Oracle® Database

Select AI User's Guide

Release 26ai
G35918-02
January 2026

Oracle Database Select AI User's Guide, Release 26ai

G35918-02

Primary Author: Sarika Surampudi

Contributing Authors: Dhanish Kumar

Contents

17 Private Endpoint Access for Select AI Models

18 Examples of Using Select AI

Part II Select AI Agent

19 Select AI Agent

Part I
Select AI

- Use Select AI for Natural Language Interaction with your Database
- About Select AI
  Use natural language to interact with your database and LLMs through SQL to enhance user productivity and develop AI-based applications. Select AI simplifies and automates using generative AI, whether generating, running, and explaining SQL from a natural language prompt, using retrieval augmented generation with vector stores, generating synthetic data, or chatting with the LLM.
- Select AI for Python
- Select AI Concepts
  Explores the concepts and terms related to Select AI.
- Select AI Use Cases
  Select AI enhances data interaction and enables developers to build AI-driven applications directly from SQL, transforming natural language prompts to SQL queries and text responses, supporting chat interaction with LLMs, enhancing response accuracy with current data using RAG, and generating synthetic data.
- Getting Started with Select AI
  To get started, review the prerequisites and the tasks that you need to perform to use Select AI.
- Manage AI Profiles
  You can create and manage your AI profiles through `DBMS_CLOUD_AI` package.
- Use AI Keyword to Enter Prompts
  Use `AI` as the keyword in a `SELECT` statement for interacting with the database using natural language prompts.
- Select AI Features
  Select AI enables conversations for intuitive way to work with your data, enhances response accuracy with Retrieval Augmented Generation (RAG), and generates realistic synthetic data for testing, development, and training.
- Select AI Conversations
- Select AI with Retrieval Augmented Generation (RAG)
  Select AI with RAG augments your natural language prompt by retrieving content from your specified vector store using semantic similarity search. This reduces hallucinations by using your specific and up-to-date content and provides more relevant natural language responses to your prompts.
- Use Autonomous AI Database as an AI Proxy for Select AI
- Synthetic Data Generation
  Generate synthetic data using random generators, algorithms, statistical models, and Large Language Models (LLMs) to simulate real data for developing and testing solutions effectively.
- Feedback
  Select AI enables you to provide feedback to help improve your selected LLM's ability to generate more accurate SQL queries.

- Generate a Summary with Select AI
  Select AI enables you to generate a summary of your text, especially large texts, generally supporting up to 1 GB using AI providers. You can extract key insights from texts or large files as per your specific needs. This feature uses the LLM specified in your AI profile to generate a summary for a given text.
- Translate
  With Select AI, you can use generative AI from the OCI translation service to translate your text into the language of your choice.
- Private Endpoint Access for Select AI Models
  You can enable secure, private access to generative AI models by deploying Ollama or Llama.cpp behind a private endpoint within your Virtual Cloud Network (VCN). This architecture is designed for organizations that need to keep AI processing fully private. The setup isolates both the Autonomous AI Database Serverless and your AI model servers from the public internet using private subnets, security lists, and controlled routing.
- Examples of Using Select AI
  Explore integrating Oracle's Select AI with various supported AI providers to generate, run, and explain SQL from natural language prompts or chat with the LLM.

1

Use Select AI for Natural Language Interaction with your Database

Select AI enables SQL access to generative AI using Large Language Models (LLMs) and embedding models. This includes support for natural language to SQL query generation and retrieval augmented generation, among other features.

Related Topics

-

2

About Select AI

Use natural language to interact with your database and LLMs through SQL to enhance user productivity and develop AI-based applications. Select AI simplifies and automates using generative AI, whether generating, running, and explaining SQL from a natural language prompt, using retrieval augmented generation with vector stores, generating synthetic data, or chatting with the LLM.

When you use Select AI, Autonomous AI Database manages the process of converting natural language into SQL. This means you can provide a natural language prompt instead of SQL code to interact with your data. Select AI serves as a productivity tool for SQL users and developers and enables non-expert SQL users to derive useful insights from their data, without having to understand data structures or technical languages.

Select AI also automates the retrieval augmented generation (RAG) process from generating vector embeddings to retrieving relevant content based on your prompt through semantic similarity search using your vector store. Other features include synthetic data generation, supporting chat history for conversations, and other features, all from a SQL interface.

The `DBMS_CLOUD_AI` package enables integration with a user-specified LLM for generating SQL code using natural language prompts. For natural language to SQL generation, this package provides an augmented prompt to the LLM containing the relevant database schema metadata. This enables generating, running, and explaining SQL queries based on natural language prompts. It also facilitates retrieval augmented generation using vector stores, synthetic data generation, and allows for chatting with the LLM. The `DBMS_CLOUD_AI` package works with AI providers listed in Select your AI Provider and LLMs.

> **Note**
>
> - You must have an account with the AI provider and provide the credentials through `DBMS_CLOUD_AI` objects that the Autonomous AI Database uses.
> - You can submit prompts in multiple languages. The quality of the result depends on the abilities of the specific LLM or the embedding model (transformer) being used. Check your LLM or embedding model documentation for multi-language support.

Topics

- Usage Guidelines
  Provides usage guidelines to assist in the use of Select AI for natural language to SQL generation..
- Supported Platforms
  Select AI is supported on Autonomous AI Database Serverless and Autonomous AI Database on Dedicated Exadata Infrastructure and Cloud at Customers.
- Select your AI Provider and LLMs
  Choose an AI provider and LLM that meets your security standards and aligns with your specific needs, like text or code generation.

Usage Guidelines

Provides usage guidelines to assist in the use of Select AI for natural language to SQL generation..

Intended Use

This feature generates, runs, and explains SQL queries from user-provided natural language prompts. It automates tasks that users would otherwise perform manually using their schema metadata and a large language model (LLM) of their choice. Additionally, it facilitates retrieval augmented generation with vector stores and enables chatting with the LLM.

Depending on the Select AI action you specify, you provide a prompt, whether for natural language to SQL generation, RAG, or pass-through chat, and Select AI automates the interaction with LLMs and your database using SQL and PL/SQL interfaces. Specifically, it generates SQL queries from natural language based on metadata from the specified schema and tables. Additionally, it facilitates chat-based generative AI, optionally enhanced with content from vector stores through retrieval augmented generation (RAG) for improved response quality. It also explains SQL queries based on natural language prompts and supports synthetic data generation for one or multiple schema tables. Select AI enables submitting general requests with the `chat` action.

Prompt Augmentation Data

For SQL query generation, the database augments the user-specified prompt with database metadata to mitigate hallucinations from the LLM. The augmented prompt is then sent to the user-specified LLM to produce the query. When using vector stores with retrieval augmented generation (RAG), content from the vector store is retrieved using semantic similarity search with the provided prompt. This content becomes part of the augmented prompt that is sent to the LLM.

The database augments the prompt with schema metadata only. This metadata may include schema definitions, table and column comments, and content available from the data dictionary. For the purposes of SQL generation, the database does not provide table or view contents (actual row or column values) when augmenting the prompt.

The `narrate` action, however, does provide to the LLM either:

- the result of a natural language to SQL query, which contains database data, or
- the result of semantic similarity search as retrieved from the vector store supporting retrieval augmented generation (RAG).

The LLM uses these results to generate a natural language text response.

> **Warning**
>
> Large language models (LLMs) have been trained on a broad set of text documentation and content, typically from the Internet. As a result, LLMs may have incorporated patterns from invalid or malicious content, including SQL injection. Thus, while LLMs are adept at generating useful and relevant content, they also can generate incorrect and false information including SQL queries that produce inaccurate results and/or compromise security of your data.
>
> The queries generated on your behalf by the user-specified LLM provider will be run in your database. Your use of this feature is solely at your own risk, and, notwithstanding any other terms and conditions related to the services provided by Oracle, constitutes your acceptance of that risk and express exclusion of Oracle's responsibility or liability for any damages resulting from that use.

Supported Platforms

Select AI is supported on Autonomous AI Database Serverless and Autonomous AI Database on Dedicated Exadata Infrastructure and Cloud at Customers.

- Autonomous AI Database Serverless
- Autonomous AI Database on Dedicated Exadata Infrastructure
- Autonomous AI Database on Dedicated Exadata Infrastructure Region
- Autonomous AI Database Cloud@Customer

Select your AI Provider and LLMs

Choose an AI provider and LLM that meets your security standards and aligns with your specific needs, like text or code generation.

Different LLMs excel at various tasks based on their training data and intended purpose. Some models are excellent for text generation but may not perform well in code generation, while others are specifically optimized for coding tasks. Choose an LLM that best suits your needs.

AI Provider	LLMs	Embedding Model for RAG	Purpose
OCI Generative AI	• meta.llama-3.3-70b-instruct (default) • meta.llama-3.2-90b-vision-instruct • meta.llama-3.2-11b-vision-instruct • meta.llama-3.1-70b-instruct • meta.llama-3.1-405b-instruct • cohere.command-r-08-2024 • cohere.command-r-plus-08-2024 • cohere.command-r-16k (deprecated) • cohere.command–r-plus (deprecated) • xai.grok-3 • xai.grok-3-fast • xai.grok-3-mini • xai.grok-3-mini-fast • xai.grok-4 • xai.grok-4-fast-reasoning • xai.grok-4-fast-non-reasoning See: • About the Chat Models in Generative AI • The xAI Platform for OCI Generative AI	• cohere.embed-english-v3.0 (default) • cohere.embed-multilingual-v3.0 • cohere.embed-english-light-v3.0 • cohere.embed-multilingual-light-v3.0 See About the Embedding Models in Generative AI.	The OCI Generative AI Chat models are supported for all `SELECT AI` actions such as `runsql`, `showsql`, `explainsql`, `narrate`, and `chat`. The OCI Generate text models are supported only for `SELECT AI chat` action. To configure your profile attributes, see #unique_34.
Azure OpenAI Service	• GPT-4o • GPT-4 • GPT-4 Turbo with Vision • GPT-3.5-Turbo	text-embedding-ada-002	Best suited for generating SQL from natural language prompts, `chat` action, and Select AI RAG.
OpenAI	• gpt-3.5-turbo (default) • gpt-4o • gpt-4o-mini • gpt-4 • gpt-4-0613 • gpt-4-32k • gpt-4-32k-0613 • gpt-3.5-turbo-0613 • gpt-3.5-turbo-16k • gpt-3.5-turbo-16k-0613	text-embedding-ada-002	Best suited for generating SQL from natural language prompts, `chat` action, and Select AI RAG.

AI Provider	LLMs	Embedding Model for RAG	Purpose
OpenAI-Compatible	Models from OpenAI-compatible providers such as: • Fireworks AI • xAI • Others	Embedding models from OpenAI-compaitble providers. For example, see Fireworks AI embedding models.	Supports a wide range of use cases.
Cohere	• command (default) • command-nightly (experimental) • command-r • command-r-plus • command-light • command-light-nightly (experimental) • custom models	embed-english-v2.0	Best suited for `chat` action.
Google	• gemini-1.5-flash (default) • gemini-1.5-pro • gemini-1.0-pro	text-embedding-004 (default)	Best suited for generating SQL from natural language prompts, `chat` action, and Select AI RAG.
Anthropic	• claude-3-5-sonnet-20240620 (default) • claude-3-opus-20240229 • claude-3-sonnet-20240229 • claude-3-haiku-20240307	NA	Best suited for generating SQL from natural language prompts, `chat` action, and Select AI RAG.
Hugging Face	• Mixtral-8x7B-Instruct-v0.1 (default) • Meta-Llama-3-70B-Instruct • Qwen1.5-1.8B • other chat models	NA	Best suited for generating SQL from natural language prompts, `chat` action, and Select AI RAG.
AWS	• Foundation models • Custom models	• amazon.titan-embed-text-v1 • amazon.titan-embed-text-v2:0 • cohere.embed-english-v3	Supports a wide range of use cases.

Note

- Specify OpenAI-compatible provider through `provider_endpoint` instead of `provider` parameter. See #unique_34.
- For models that accept images, use `meta.llama-3.2-90b-vision-instruct`. This model is specifically trained for vision and images. While it can be used for text and SQL generation, the model is best suited for images. To learn more, see Chat in OCI Generative AI.

  The `meta.llama-3.2-11b-vision-instruct` model provides robust multimodal capabilities.
- Embedding models are also known as transformer models.

3

Select AI for Python

Select AI for Python integrates generative AI capabilities into Autonomous AI Database workflows. Select AI for Python provides a client library, `select_ai`, that enables you to use `DBMS_CLOUD_AI` capabilities in Autonomous AI Database from Python. Select AI for Python supports enhanced generative AI workflows, summarization, feedback mechanisms, consistent metadata management, and agentic AI capabilities. It also supports Python 3.14 and includes an updated HTML documentation site (New)

What You Can Do

- Connect to the database using synchronous or asynchronous connections
- Create and manage AI profiles to enable using AI models from a wide range of AI providers
- Use natural language to query your database through AI-based SQL generation
- Describe query results in natural language
- Create and manage conversations with prompt history
- Create and update vector indexes easily for use with built-in and automated Retrieval Augmented Generation (RAG) workflows
- Generate synthetic data for testing and analysis using generative AI
- Summarize text or query results
- Record and manage model feedback
- Create autonomous and interaction AI agents. See Select AI Agent for Python for details.

Supported Platforms

Select AI for Python is certified for Autonomous Database 19*c* and Autonomous AI Database 26ai. Select AI for Python may work on other platforms, however, it is not certified.

Click https://github.com/oracle/python-select-ai/issues to report issues.

Supported Functions for Select AI Profile (Synchronous and Asynchronous)

When you send prompts through a profile, you can choose among several functions defined for AI profile objects. Some are as follows:

- **`create()`**: Create the AI profile in the database or replace if necessary.
- **`delete()`**: Remove the profile.
- **`generate()`**: Use the profile to process a prompt as per the chosen action.
- **`generate_synthetic_data()`**: Create synthetic data based on the supplied attributes.
- **`get_attributes()`**: Return the current profile attributes.
- **`run_sql()`**: generate and run SQL (default).
- **`show_sql()`**: generate SQL without running it.
- **`explain_sql()`**: provide explanation for generated SQL.

- **`narrate()`**: describe query results in natural language.
- **`chat()`**: engage in a freeform conversation.
- **`show_prompt()`**: display the constructed prompt sent to the generative AI model.
- **`summarize()`**: Produce a summary for provided content.
- **`add_positive_feedback()`**, **`add_negative_feedback()`**, **`delete_feedback()`**: Manage user feedback associated with generated queries.

For a complete list of functions, see Select AI for Python guide. See also Use AI Keyword to Enter Prompts for more information on the Select AI actions.

Supported Classes

The library includes classes to manage providers, profiles, conversations, vector indexes, and synthetic data. Both synchronous and asynchronous versions are available.

- Provider Classes: Define AI provider: `OpenAIProvider`, `AzureProvider`, `OCIGenAIProvider`, `AWSProvider`, `GoogleProvider`, `AnthropicProvider`, `CohereProvider`, `HuggingFaceProvider`.
- `Profile`: defines the generative AI profile to process prompts (provider, credentials, metadata, options) and supports synthetic data generation.
- `ProfileAttributes`: profile configuration details such as provider, credential name, max tokens, temperature, object list, or vector index.
- `ConversationAttributes`: manage conversational context across prompts.
- `VectorIndex` and `VectorIndexAttributes`: create and manage vector indexes for RAG.
- `SyntheticDataAttributes`: create synthetic datasets for testing and development.

Asynchronous equivalents exist for `Profile`, `Conversation`, and `VectorIndex` classes.

For complete API reference, see Select AI for Python guide.

API and Attribute Update Enhancements

The following enhancements are available:

- All proxy objects now support `fetch()` to retrieve existing objects.
- All proxy objects provide `set_attribute()` and `set_attributes()` for consistent updates

Privilege and HTTP Access

Privilege management is separate from HTTP access configuration.

Privilege APIs:

- `select_ai.grant_privileges`
- `select_ai.revoke_privileges`

Privileges are updated to include:

- `DBMS_CLOUD`
- `DBMS_CLOUD_AI`
- `DBMS_CLOUD_AI_AGENT`
- `DBMS_CLOUD_PIPELINE`

HTTP access APIs:

- `select_ai.grant_http_access`
- `select_ai.revoke_http_access`

Other Enhancements

- Python 3.14 support
- New HTML documentation site on GitHub using the Python docs theme: GitHub Select AI for Python Documentation

4

Select AI Concepts

Explores the concepts and terms related to Select AI.

- Vector
- Vector Database
- Vector Distance
- Vector Index
- Vector Store

Actions

An action in Select AI is a keyword that instructs Select AI to perform different behavior when acting on the prompt. By specifying an action, users can instruct Select AI to process their natural language prompt to generate SQL code, to respond to a chat prompt, narrate the output, display the SQL statement, or explain the SQL code, leveraging the LLMs to efficiently interact with the data within their database environment.

See Use AI Keyword to Enter Prompts for supported Select AI actions.

AI Agent

See Select AI Agent Concepts.

AI Model

A general term encompassing various types of artificial intelligence models, including large language models (LLMs) and transformers (also referred as embedding models), used for tasks like text generation, translation, and image recognition. An AI model is a program, trained on data, that detects patterns and makes predictions or decisions based on new inputs. Within the context of Oracle, AI model specifically refers to the various machine learning and large language models (LLMs) available through Oracle's services. See Concepts for Generative AI for more information.

AI Profile

An AI profile is a specification that includes the AI provider to use and other details regarding metadata and database objects required for generating responses to natural language prompts. See CREATE_PROFILE Procedure and Profile Attributes.

AI Provider

An AI Provider in Select AI refers to the service provider that supplies the LLM or transformer or both for processing and generating responses to natural language prompts. These providers offer models that can interpret and convert natural language for the use cases highlighted under the LLM concept. See Select your AI Provider and LLMs for the supported providers.

Chatbot

An AI-powered conversational agent designed to interact with users in natural language, often used for customer service or information retrieval. In the context of Select AI, the **Ask Oracle** chatbot helps users ask natural language questions and receive AI-generated responses backed by their database and private content. Through this UI, users can:

- Ask natural language questions and get SQL generated automatically (NL2SQL).
- Run queries against database tables and views using Select AI.
- Use Retrieval-Augmented Generation (RAG) to include private document content stored in Autonomous AI Database.
- Interact with agent teams you defined with Select AI Agent.

See Ask Oracle for more details.

Cloud Link

A cloud link establishes secure, private connectivity between Oracle Cloud Infrastructure and external cloud providers or on-premises networks, facilitating seamless data exchange. In Select AI, cloud links enable Autonomous AI Database to incorporate external data into NL2SQL interactions without public exposure, empowering users to query hybrid environments conversationally while adhering to Oracle's security standards, such as encryption and access controls, for compliant AI-driven analytics. See Use Cloud Links for Read Only Data Access on Autonomous AI Database for more details.

Conversations

Conversations in Select AI represent an interactive exchange between the user and the system, enabling users to query or interact with the database through a series of natural language prompts. Select AI incorporates session-based short-term conversations to generate context-aware responses for the current prompt based on prior interactions. Up to 10 previous prompts are incorporated into the current request with short-term conversations, creating an augmented prompt that is sent to the LLM. Select AI supports using customizable long-term conversations enabling you to use Select AI with different topics without mixing context, that can be configured through conversation APIs from the DBMS_CLOUD_AI Package. See Select AI Conversations.

Database Credentials

Database credentials are authentication credentials used to access and interact with databases. They typically consist of a user name and a password, sometimes supplemented by additional authentication factors like security tokens. These credentials are used to establish a secure connection between an application or user and a database, such that only authorized individuals or systems can access and manipulate the data stored within the database.

Database Link

A database link connects an Oracle database to remote databases, enabling transparent access to external data as if it were local. In Select AI, database links integrate with Autonomous AI Database, or on-premises Oracle AI Database to extend NL2SQL capabilities to federated sources, supporting natural language queries that span on-premises or other cloud environments securely. See CREATE DATABASE LINK and Use Database Links with Autonomous AI Database for more details.

Embedding Model

An AI model that converts input data into vector embeddings to capture semantic relationships, often used in tasks like language understanding and image recognition. Select AI uses embedding models to compute embeddings for your documents, tables, and query text. These embeddings power semantic search, RAG workflows, similarity scoring, and relevance ranking inside Autonomous AI Database.

Hallucination in LLM

Hallucination in the context of Large Language Models refers to a phenomenon where the model generates text that is incorrect, nonsensical, or unrelated to the input prompt. Despite being a result of the model's attempt to generate coherent text, these responses can contain information that is fabricated, misleading, or purely imaginative. Hallucination can occur due to biases in training data, lack of proper context understanding, or limitations in the model's training process.

IAM

Oracle Cloud Infrastructure Identity and Access Management (IAM) lets you control who has access to your cloud resources. You can control what type of access a group of users have and to which specific resources. To learn more, see Overview of Identity and Access Management.

Iterative Refinement

Iterative refinement is a process of gradually improving a solution or a model through repeated cycles of adjustments based on feedback or evaluation. It starts with an initial approximation, refines it step by step, and continues until the desired accuracy or outcome is achieved. Each iteration builds on the previous one, incorporating corrections or optimizations to move closer to the goal.

In text summary generation, iterative refinement can be useful for processing large files or documents. The process splits the text into manageable-sized chunks, for example, that fit within an LLM's token limits, generates a summary for one chunk, and then improves the summary by sequentially incorporating the following chunks.

Use cases for iterative refinement:

- Best suited for situations where contextual accuracy and coherence are critical, such as when summarizing complex or highly interconnected texts where each part builds on the previous.
- Ideal for smaller-scale tasks where sequential processing is acceptable.

See Summarization Techniques.

Large Language Model (LLM)

A Large Language Model (LLM) refers to an advanced type of artificial intelligence model that is trained on massive amounts of text data to support a range of use cases depending on their training data. This includes understanding and generating human-like language as well as software code and database queries. These models are capable of performing a wide range of

natural language processing tasks, including text generation, translation, summarization, question answering, sentiment analysis, and more. LLMs are typically based on sophisticated deep learning neural network models that learn patterns, context, and semantics from the input data, enabling them to generate coherent and contextually relevant text.

MapReduce

In general, the MapReduce programming model enables processing large-volume data by dividing tasks into two phases: Map and Reduce.

- **Map**: Processes input data and transforms it into key-value pairs.
- **Reduce**: Aggregates and summarizes the mapped data based on keys. MapReduce performs parallel processing of large data sets.

In the case of Select AI Summarize, MapReduce partitions text into multiple chunks and processes them in parallel and independently, generating individual summaries for each chunk. These summaries are then combined to form a cohesive overall summary.

Use cases for map reduce:

- Best suited for large-scale, parallel tasks where speed and scalability are priorities, such as summarizing very large data sets or documents.
- Ideal for situations where chunk independence is acceptable, and the summaries can be aggregated later.

See Summarization Techniques.

Metadata

Metadata is data that describes data. In the case of Select AI, metadata is database metadata, which refers to the data that describes the structure, organization, and properties of the database tables and views.

For database tables and views, metadata includes column names and types, constraints and keys, view definitions, relationships, lineage, quality and freshness indicators, security classifications, and access policies. Well-managed metadata enables discoverability, correct usage, performance tuning, and compliance. Select AI augments NL2SQL prompts with table metadata that include the table definition (table name, columns names and their data types), and optionally table and column comments, annotations, and constraints.

Metadata Clone

A metadata clone or an Autonomous AI Database clone creates a copy of a metadata defining the database or schema, containing only the structure, not the actual data. This clone includes tables, indexes, views, statistics, procedures, and triggers without any data rows. Developers, testers, or those building database templates find this useful. To learn more, see Cloning Moving or Upgrading Autonomous Database.

Metadata Enrichment

The practice of augmenting database schemas with high-quality descriptions, comments, and annotations so an LLM can better understand the intent for tables and columns, clarify business meaning, and generate more accurate SQL. It turns bare table or column names into well-documented assets with clear intent, relationships, and constraints.

Candidate information to include:

- Table and column descriptions: purpose, business definitions, units, and allowed value ranges
- Keys and relationships: primary/foreign keys, join paths
- Data semantics: time granularity, slowly changing dimensions, deduplication rules
- Constraints and quality: nullability, uniqueness, validation rules, data freshness
- Synonyms and aliases: common business terms that map to technical names
- Examples and patterns: sample values, common filters or aggregations

See Overview of AI Enrichment to learn more about adding such metadata using Oracle SQL Developer for VS Code through Visual Studio Code.

Natural Language Prompt

A natural language prompt consists of instructions, questions, or input statements expressed in everyday human language (such as English) that guide an LLM's response. Instead of requiring code or specialized syntax, users interact with the LLM by typing sentences or phrases that describe their intent, ask for information, or specify a task.

For example:

- "What is the revenue in the last quarter in each corporate region?"
- "What is our internal corporate policy on parental leave?"
- "Summarize this article."
- "Write an email to a customer apologizing for a delayed shipment."
- "What are the key differences between SQL and NoSQL databases?"

These prompts leverage the model’s understanding of human language to generate useful, contextually relevant outputs. Natural language prompts are central to LLM usability, making advanced AI capabilities accessible to users without technical expertise.

Network Access Control List (ACL)

A Network Access Control List is a set of rules or permissions that define what network traffic is allowed to pass through a network device, such as a router, firewall, or gateway. ACLs are used to control and filter incoming and outgoing traffic based on various criteria such as IP addresses, port numbers, and protocols. They play a crucial role in network security by enabling administrators to manage and restrict network traffic to prevent unauthorized access, potential attacks, and data breaches.

NL2SQL

Natural Language to SQL (NL2SQL) converts natural language questions into SQL statements using generative AI.

Select AI actively uses NL2SQL to interpret user prompts and generate correct, runnable SQL against your Autonomous AI Database or connected external sources. This enables business users ask questions like *“Show me last quarter’s revenue by region”* and receive accurate SQL queries and results with no SQL expertise.

ONNX

ONNX (Open Neural Network Exchange) is an open standard format for representing machine learning and deep-learning models. ONNX standardizes the representation and interchange of machine learning models across frameworks, enabling seamless deployment and interoperability. See ONNX for more details.

Select AI can use generative AI models exported in ONNX format to run AI workloads directly inside Autonomous AI Database or through supported runtimes enabling organizations to leverage pre-trained models for natural language processing tasks like query generation. By using ONNX models, you keep inference close to your data, reduce data movement, and enable consistent model processing across different tools and environments ensuring compliant AI operations.

ONNX Runtime

ONNX Runtime runs ONNX-formatted models efficiently across hardware platforms, optimizing inference for real-time AI applications.

Select AI users can specify in-database ONNX-format models in their AI profile in support of RAG. The database embeds the ONNX Runtime in Oracle AI Database 26ai and Autonomous AI Database. Using the in-database ONNX Runtime avoids sending content to an external engine to produce, for example, vector embeddings. ONNX Runtime powers the runtime evaluation of transformer-based models within Autonomous AI Database, facilitating developers to load ONNX models, fast natural language to SQL (NL2SQL) conversions, compute embeddings, classify data, or run inference inside the database engine without sending data to external service, which enhances query performance and improves security, latency, and governance. See Example: Select AI with In-database Transformer Models and ONNX Runtime for more details.

Private Endpoint

A secure and dedicated communication point that allows restricted access to specific services or resources. A private endpoint establishes a secure, dedicated connection that restricts access to specific services or resources, ensuring isolated communication. In Select AI, organizations can configure private endpoints in Oracle Cloud Infrastructure (OCI) to connect with privately hosted LLMs like Ollama or Llama.cpp on virtual machines (VMs), addressing security needs by processing AI workloads within the Oracle Virtual Cloud Network. This setup includes a public subnet with a jump server for controlled access and a private subnet housing the Autonomous AI Database and AI models, preventing internet exposure and keeping all components compliant with enterprise isolation requirements. See Private Endpoint Access for Select AI Models for more details.

Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) is a technique that involves retrieving relevant information for a user's query and supplying that information to a large language model (LLM) to improve responses and reduce hallucination.
Most commonly, RAG involves vector search, but more generally, includes augmenting a prompt of database content (either manually or automatically) such as schema metadata for SQL generation or database content explicitly queried. Other forms of augmentation can involve technologies such as graph analytics and traditional machine learning.

Semantic Similarity Search

Semantic similarity search identifies and retrieves data points that closely match a given query by comparing feature vectors in a vector store.

Sidecar

The sidecar architecture allows one database to act as the central metadata repository for both local and remote data sources, that is, Oracle and non-Oracle. Select AI uses this architecture by leveraging the metadata to build an augmented prompt that is sent to the user's chosen LLM, which then generates a federated SQL query. A key benefit of the sidecar is that it enables data to remain in its original location, eliminating the need for data duplication or complex ETL processes.

It supports federated access to diverse external systems such as, BigQuery, Redshift, multi-cloud, or on-premises databases by securely bridging these sources to Autonomous AI Database.

Similarity Threshold

A similarity threshold sets a minimum score to classify two items as related, filtering results based on their vector proximity or distance. In Select AI, the similarity threshold helps filter results that fall below a required level of semantic closeness, ensuring that only highly related document chunks, rows, or embeddings are returned.

Synthetic Data Generation

In the context of Select AI, Synthetic Data Generation is the capability to automatically generate artificial data that conforms to your database schema enabling you to populate tables for development, testing, training, or proof-of-concept scenarios without using sensitive or production data. Select AI provides the PL/SQL function `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` to produce synthetic data sets. See Synthetic Data Generation for more details.

Transformer

A type of deep learning model architecture commonly used for natural language processing tasks, such as vector embedding generation or text generation and translation. In Select AI, transformer-based LLMs drive the conversion of user queries into SQL queries that can be run within your database.

Vector

In the context of semantic similarity search, a vector is a mathematical representation that captures the semantic meaning of data points, such as words, documents, or images, in a multi-dimensional space.

In the context of Select AI, vectors support retrieval augmented generation by capturing the meaning of text content to enable fast semantic retrieval from the database.

Vector Database

A database that stores vector embeddings, which are mathematical representations of data points used in AI applications to support efficient semantic similarity search. Oracle Autonomous AI Database and Oracle AI Database serve as a vector database with optimized vector indexes.

In Select AI, the vector database component (powered by Oracle AI Vector Search) indexes embeddings generated from enterprise data. This enables natural language queries to retrieve semantically similar results, improves relevance for AI-powered search and RAG workflows, and provides seamless integration with Oracle Cloud environments.

Vector Distance

Vector distance measure the similarity or dissimilarity between feature vectors by calculating the distance between them in a multidimensional space.

Vector Index

A vector index organizes and stores vectors to enable efficient similarity search and retrieval of related data.

Vector Store

A vector store includes systems that store, manage, and enable semantic similarity search involving vector embeddings. This includes standalone vector databases and Oracle AI Database 26ai AI Vector Search.

5

Select AI Use Cases

Select AI enhances data interaction and enables developers to build AI-driven applications directly from SQL, transforming natural language prompts to SQL queries and text responses, supporting chat interaction with LLMs, enhancing response accuracy with current data using RAG, and generating synthetic data.

Use cases include:

- Generate SQL from natural language prompts

  Developer productivity: Select AI significantly enhances developer productivity by providing "starter" SQL queries quickly. Developers can input natural language prompts, and Select AI generates SQL based on your database schema tables and views. This reduces the time and effort needed to write complex queries from scratch, allowing developers to focus on refining and optimizing the generated queries for their specific needs.

  Natural language queries for end-users: Select AI empowers end-users to interact with your application's underlying data tables and views using natural language queries. This functionality allows users without SQL expertise to ask questions and retrieve data directly, making data access more intuitive and user-friendly relative to the capabilities of LLM being used and the quality of the schema metadata available.

  Other capabilities for SQL generation: The following highlighted capabilities are also supported for natural language to SQL generation:

  – Specify schema or tables or views: Select AI enables you to specify an object list consisting of schema and optionally tables or views within that schema.

  – Automatically detect relevant table metadata: Select AI automatically detects relevant tables and send metadata only for those specific tables, relevant to the query, in Oracle AI Database 26ai.

  – Restrict table access: Select AI enables you to restrict table access by considering only the tables listed in the AI profile attributes for SQL generation.

  – Specify case sensitivity for columns: Select AI enables user to specify case sensitivity such that the LLM produces case-insensitive responses from the database and LLM.

- Conversations

  Enable chatbot-like features with Select AI, enabling users to have natural conversations for querying data and performing actions. These chats can keep track of context, giving follow-up answers that clarify or expand on original questions. This scenario boosts engagement and makes complicated queries easier through conversation.

- Agentic workflows with Select AI Agent

  Use Select AI Agent to coordinate agents, tools (SQL, RAG, Websearch, Notifications), and tasks for multi-step scenarios such as data retrieval, and notification. See Select AI Agent to learn more.

- Customized media generation

  Select AI can be used to generate personalized media content such as emails tailored to individual customer details. For example, in your prompt you could instruct the LLM to create a friendly and upbeat email encouraging a customer to try a set of recommended products. These recommendations could be based on customer demographics or other

specific information available in your database. This level of customization enhances customer engagement by delivering relevant and appealing content directly to the customer.

- Code generation

  With the Select AI `chat` action, you can use Select AI to ask your specified LLM to generate code from natural language prompts. This feature supports various programming languages such as SQL, Python, R, and Java. Examples include:

  - **Python Code**: "Write the Python code to compute a confusion matrix over a DataFrame with columns ACTUAL and PREDICTED."
  - **SQL DDL**: "Write the DDL for a SQL table with columns name, age, income, and country."
  - **SQL Query**: "Write the SQL query that will use the Oracle Machine Learning in-database model named CHURN_DT_MODEL to predict which customers will churn and with what probability."

- Retrieval Augmented Generation (RAG)

  Use vector store content for semantic similarity search to enhance prompt accuracy and relevance in LLM responses.

- Synthetic data generation

  Generate synthetic data using LLMs that conforms to your schema for solution testing, proofs of concept, and other uses. Synthetic data can support better testing of your applications in the absence of real data, leading to overall quality of your application.

  Synthetic data generation can also be used to populate an Autonomous AI Database clone or a metadata clone. Select AI supports generating synthetic data for such clones. Using synthetic data helps to protect sensitive data while enabling development, testing, and validating user experiences. It's also useful for AI and machine learning projects needing sample data for model training or test data for scoring.

6

Getting Started with Select AI

To get started, review the prerequisites and the tasks that you need to perform to use Select AI.

- Configure your system to use Select AI
- Create an AI profile and enable the AI profile
- Use AI keyword in SELECT statement and enter natural language prompts
- Example of using Select AI with OCI Generative AI
- Examples of Using Select AI
- Manage AI profile, configure vector index, or drop your AI profile by reviewing DBMS_CLOUD_AI package
- Customize your AI profiles by supplying the profile attributes
- Display AI profiles, AI profile attributes, and vector index details by querying DBMS_CLOUD_AI views
- See Autonomous Database Supplied Package Reference

7

Manage AI Profiles

You can create and manage your AI profiles through `DBMS_CLOUD_AI` package.

- Use DBMS_CLOUD_AI to Configure AI Profiles
  Autonomous AI Database uses AI profiles to facilitate and configure access to an LLM and to setup for generating, running, and explaining SQL based on natural language prompts. It also facilitates retrieval augmented generation using embedding models and vector indexes and allows for chatting with the LLM.
- Perform Prerequisites for Select AI
  Before you use Select AI, here are the steps to enable `DBMS_CLOUD_AI`.
- Create and Set an AI Profile
  Describes the steps to create and enable an AI profile.

> **See Also**
>
> Use Select AI for Natural Language Interaction with your Database

Use DBMS_CLOUD_AI to Configure AI Profiles

Autonomous AI Database uses AI profiles to facilitate and configure access to an LLM and to setup for generating, running, and explaining SQL based on natural language prompts. It also facilitates retrieval augmented generation using embedding models and vector indexes and allows for chatting with the LLM.

AI profiles include database objects that are the target for natural language queries. Metadata used from these targets can include database table names, column names, column data types, and comments. You create and configure AI profiles using the `DBMS_CLOUD_AI.CREATE_PROFILE` and `DBMS_CLOUD_AI.SET_PROFILE` procedures.

In addition to specifying tables and views in the AI profile, you can also specify tables mapped with external tables, including those described in Query External Data with Data Catalog. This enables you to query data not just inside the database, but also data stored in a data lake's object store.

Perform Prerequisites for Select AI

Before you use Select AI, here are the steps to enable `DBMS_CLOUD_AI`.

The following are required to use `DBMS_CLOUD_AI`:

- Access to an Oracle Cloud Infrastructure cloud account and to an Autonomous AI Database instance.
- A paid API account of a supported AI provider, one of:

AI Provider	API Keys
OpenAI	See Use OpenAI to get your API keys.
OpenAI-compatible providers	See Use OpenAI-Compatible Providers to get your API keys and `provider_endpoint`.
Cohere	See Use Cohere to get your secret API keys.
Azure OpenAI Service	See Use Azure OpenAI Service for more information on how to configure Azure OpenAI Service.
OCI Generative AI	See Use OCI Generative AI.
Google	See Use Google to get your API keys.
Anthropic	See Use Anthropic to get your API keys.
Hugging Face	See Use Hugging Face to get your API keys.
AWS	See Use AWS to get your API keys and model ID.

- Network ACL privileges to access your external AI provider.

  > **Note**
  >
  > Network ACL privileges are not required for OCI Generative AI.

- A credential that provides access to the AI provider.
- Grant Privileges for Select AI
  To use Select AI, the administrator must grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI` package. Learn about additional privileges required for Select AI and its features.
- Examples of Privileges to Run Select AI
  Review examples of privileges required to use Select AI and its features.
- Configure Select AI to Use Supported AI Providers
  Explore how to enable your AI providers to use with Select AI.

Grant Privileges for Select AI

To use Select AI, the administrator must grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI` package. Learn about additional privileges required for Select AI and its features.

To configure `DBMS_CLOUD_AI`:

1. Grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI` package to the user who wants to use Select AI.

   By default, only the system administrator has `EXECUTE` privilege. The administrator can grant `EXECUTE` privilege to other users.

2. Grant `EXECUTE` privilege on `DBMS_CLOUD_PIPELINE` to the user who wants to use Select AI with RAG.

   > **Note**
   >
   > If the user already has the `DWROLE` role, this privilege is included and additional grant is not required.

3. Grant network ACL access to the user who wants to use Select AI and for the AI provider endpoint.

   The system administrator can grant network ACL access. See APPEND_HOST_ACE Procedure for more information.

4. Create a credential to enable access to your AI provider.
   See #unique_94 for more information.

5. Grant quotas in tablespace to manage the amount of space in a specific tablespace to the user who wants to use Select AI with RAG.

Examples of Privileges to Run Select AI

Review examples of privileges required to use Select AI and its features.

The following example grants the `EXECUTE` privilege to `ADB_USER`:

```
GRANT execute on DBMS_CLOUD_AI to ADB_USER;
```

The following example grants `EXECUTE` privilege for the `DBMS_CLOUD_PIPELINE` package required for RAG:

```
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;
```

To check the privileges granted to a user for the `DBMS_CLOUD_AI` and `DBMS_CLOUD_PIPELINE` packages, an administrator can run the following:

```
SELECT table_name AS package_name, privilege
 FROM DBA_TAB_PRIVS
 WHERE grantee = '<username>'
 AND   (table_name = 'DBMS_CLOUD_PIPELINE'
        OR table_name = 'DBMS_CLOUD_AI');
```

The following example grants `ADB_USER` the privilege to use the *api.openai.com* endpoint.

> **Note**
>
> This procedure is not applicable to OCI Generative AI.

```
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.openai.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
   );
END;
/
```

The parameters are:

- `host`: The host, which can be the name or the IP address of the host. You can use a wildcard to specify a domain or an IP subnet. The host or domain name is not case sensitive.

AI Provider	Host
OpenAI	*api.openai.com*
OpenAI-compatible providers	For example, for Fireworks AI, use *api.fireworks.ai*
Cohere	*api.cohere.ai*
Azure OpenAI Service	*<azure_resource_name>.openai.azure.com* See #unique_34 to know more about `azure_resource_name`.
Google	*generativelanguage.googleapis.com*
Anthropic	*api.anthropic.com*
Hugging Face	*api-inference.huggingface.co*
AWS	*bedrock-runtime.us-east-1.amazonaws.com*

- `ace`: The access control entries (ACE). The `XS$ACE_TYPE` type is provided to construct each ACE entry for the ACL. For more details, see Creating ACLs and ACEs.

The following example creates a credential to enable access to OpenAI.

```
EXEC
DBMS_CLOUD.CREATE_CREDENTIAL(
credential_name   => 'OPENAI_CRED',
username          =>  'OPENAI',
password          =>  '<your_api_token>');
```

The parameters are:

- `credential_name`: The name of the credential to be stored. The `credential_name` parameter must conform to Oracle object naming conventions.
- `username`: The `username` and `password` arguments together specify your AI provider credentials.

  The `username` is a user-specified user name.
- `password`: The `username` and `password` arguments together specify your AI provider credentials.

  The `password` is your AI provider secret API key, and depends on the provider:

AI Provider	API Keys
OpenAI	See Use OpenAI to get your API keys.
OpenAI-compatible providers	See Use OpenAI-Compatible Providers to get your API keys and `provider_endpoint`.
Cohere	See Use Cohere to get your API keys.

AI Provider	API Keys
Azure OpenAI Service	See Use Azure OpenAI Service to get your API keys and to configure the service. **Note** If you are using the Azure OpenAI Service principal to authenticate, you can skip the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure. See Examples of Using Select AI for an example of authenticating using Azure OpenAI Service principal.
OCI Generative AI	See Use OCI Generative AI to generate API signing keys.
Google	See Use Google to generate your API keys.
Anthropic	See Use Anthropic to generate your API keys.
Hugging Face	See Use Hugging Face to generate your API keys.
AWS	See Use AWS to get your API keys and model ID.

The following example grants quotas on tablespace to the `ADB_USER` to use Select AI with RAG:

```
ALTER USER ADB_USER QUOTA 1T ON <tablespace_name>;
```

To the check the tablespace quota granted to a user, run the following:

```
SELECT TABLESPACE_NAME, BYTES, MAX_BYTES
FROM DBA_TS_QUOTAS
WHERE USERNAME = '<username>' AND
      TABLESPACE_NAME LIKE 'DATA%';
```

The parameters are:

- `TABLESPACE_NAME`: The tablespace for which the quota is assigned. In Autonomous AI Database, tablespaces are managed automatically and have `DATA` as a prefix.
- `BYTES`: The amount of space currently used by the user in the tablespace.
- `MAX_BYTES`: The maximum quota assigned (in bytes). If `MAX_BYTES` is -1, it means the user has unlimited quota on the tablespace. The database user creating the vector index must have `MAX_BYTES` sufficiently larger than bytes to accommodate the vector index, or `MAX_BYTES` should be -1 for unlimited quota.

Configure Select AI to Use Supported AI Providers

Explore how to enable your AI providers to use with Select AI.

- Use Anthropic
  To enable Anthropic Developer Console to generate SQL and text responses to your natural language prompts, obtain API keys from your Anthropic Developer Console paid account.
- Use AWS
  To enable AWS, obtain your API key and model ID.
- Use Azure OpenAI Service
  To enable Azure OpenAI Service to generate SQL and text responses to your natural language prompts, configure and provide access to the AI provider.
- Use Cohere
  To enable Cohere to generate SQL and text responses to your natural language prompts, obtain API keys from your Cohere paid account.
- Use Google
  To enable Google AI Studio to generate SQL and text responses to your natural language prompts, obtain API keys from your Google AI Studio paid account.
- Use Hugging Face
  To enable Hugging Face as your AI provider to generate SQL and text responses to your natural language prompts, obtain API keys from your Hugging Face paid account.
- Use OCI Generative AI
  To enable OCI Generative AI to generate SQL and text responses to your natural language prompts, generate an API signing key.
- Use OpenAI
  To enable OpenAI to generate SQL and text responses to your natural language prompts, obtain API keys from your OpenAI paid account.
- Use OpenAI-Compatible Providers
  To enable providers that are compatible with OpenAI, obtain your API key.

Use Anthropic

To enable Anthropic Developer Console to generate SQL and text responses to your natural language prompts, obtain API keys from your Anthropic Developer Console paid account.

1. Go to Anthropic Developer Console.
2. Sign up for an account if you don't have one already.
3. Once logged in, navigate to the API section or the dashboard.
4. Look for an option to generate or view API keys.
5. Click to create a new API key.
6. Copy the generated API key and save it.

   The Claude API is a paid service. You'll need to add credits to your account before you can use the API key.

Use AWS

To enable AWS, obtain your API key and model ID.

Obtain your API key and use it to create credentials through `DBMS_CLOUD.CREATE_CREDENTIAL`.

Follow this process to obtain your API key and model name:

1. Sign up for an AWS account if you don't have one already.

2. Create your access keys and secret keys from AWS Bedrock Console.
3. Copy the generated keys and save it.
4. Request access to their foundation models. See Access Amazon Bedrock foundation models.
5. Obtain the model ID. You require the model ID in `DBMS_CLOUD_AI.CREATE_PROFILE` procedure. Model ID depends on the resources that you use. If you use:
   - a base model, specify the model ID or its ARN(Amazon Resource Names). For a list of model IDs for base models, see Amazon Bedrock base model IDs
   - an inference profile, specify the inference profile ID or its ARN. For a list of inference profile IDs, see Supported Regions and models for cross-region inference.
   - a provisioned model, specify the ARN of the Provisioned Throughput. For more information, see Run inference using a Provisioned Throughput.
   - a custom model, purchase Provisioned Throughput for it. Then, specify the ARN of the resulting provisioned model. For more information, see Use a custom model in Amazon Bedrock.
   - an Amazon Bedrock Marketplace model, specify the ID or ARN of the marketplace endpoint that you created, see Amazon Bedrock Marketplace.

   Note

   Imported model is not supported with Bedrock Converse API.

To use AWS as your provider, see Example: Select AI with AWS.

Use Azure OpenAI Service

To enable Azure OpenAI Service to generate SQL and text responses to your natural language prompts, configure and provide access to the AI provider.

To use Azure OpenAI Service, perform the following steps:

1. Obtain your secret API keys. You can find your API keys in the Resource Management section of your Azure portal. On your Azure OpenAI Service Resource page, click **Keys and Endpoint**. You can copy either KEY1 or KEY2.
2. Create an Azure OpenAI Service resource and deploy a model: Create and deploy an Azure OpenAI Service resource.

   Tip

   - Make note of the resource name and deployment name as those parameters are used to provide network access permission and to create your Azure OpenAI Service profile using the `DBMS_CLOUD_AI.CREATE_PROFILE` procedure.
   - To know about rate limits for token per minute on a model, see Azure OpenAI Service quotas and limits.
3. Allow access to Azure OpenAI Service:
   - You can use your secret API key to allow access to Azure OpenAI Service. To know more, see the example in Examples of Using Select AI.

- Allow service principal to access Azure OpenAI Service and grant the required permissions to the service principal: #unique_96 and Role-based access control for Azure OpenAI Service. To know more, see the example in Examples of Using Select AI.

Use Cohere

To enable Cohere to generate SQL and text responses to your natural language prompts, obtain API keys from your Cohere paid account.

1. Login to Cohere's website with your credentials. Cohere Dashboard appears by default.
2. Alternately,click **Dashboard**.
3. Click **API Keys** on the left navigation. Copy the default API key or create another key. See API-Keys for more information.

Use Google

To enable Google AI Studio to generate SQL and text responses to your natural language prompts, obtain API keys from your Google AI Studio paid account.

1. Go to Google AI Studio.
2. Click Sign In to Google AI Studio.
3. Click Get API key on the prompt screen.
4. Select all the applicable options on the next page.
5. Click Create API key.
6. Click Create API key in new project.

   The screen displays the progress and generates an API key. Copy the key and save it.

Use Hugging Face

To enable Hugging Face as your AI provider to generate SQL and text responses to your natural language prompts, obtain API keys from your Hugging Face paid account.

1. Go to Hugging Face.
2. Sign up for an account if you don't have one already.
3. Navigate to your account settings.
4. In the navigation menu locate the **Access Tokens**.
5. Click to create a new API key.
6. Copy the generated API key and save it.

Use OCI Generative AI

To enable OCI Generative AI to generate SQL and text responses to your natural language prompts, generate an API signing key.

Use the Console or command line to generate an API signing key for any Oracle Database instance. See How to Generate the API Signing Key.

Use OpenAI

To enable OpenAI to generate SQL and text responses to your natural language prompts, obtain API keys from your OpenAI paid account.

You can find your secret API key from your profile dashboard under API keys.

Use OpenAI-Compatible Providers

To enable providers that are compatible with OpenAI, obtain your API key.

OpenAI-compatible providers use bearer authentication. Obtain your API key and use it to create credentials through `DBMS_CLOUD.CREATE_CREDENTIAL`. For example, Fireworks AI is an OpenAI-compatible provider.

Follow the process to obtain your Fireworks AI API key, model base path URL, and model name:

1. Sign up for a Fireworks AI account if you don't have one already.
2. Create an API key from Fireworks AI console.
3. Copy the generated API key and save it.
4. Obtain the provider endpoint that you require in `DBMS_CLOUD_AI.CREATE_PROFILE` procedure:
   a. Select one of the available models by clicking on the available models.
   b. Scroll to the API Example and obtain the OpenAI-compatible base URL path.

      For `Llama 3.2 3B Instruct` model, the base path URL is: `https://api.fireworks.ai/inference/v1/chat/completions`. For `provider_endpoint` parameter, exclude `/v1/chat/completions`. The `provider_endpoint` for the mentioned model is `https://api.fireworks.ai/inference`.

      > **Note**
      >
      > Some providers may require a prefix for the model name, refer to the AI provider documentation for the correct format.

5. From the same API Example, copy the model name. You require the model name in `DBMS_CLOUD_AI.CREATE_PROFILE` procedure. For example, `accounts/fireworks/models/llama-v3p2-3b-instruct`.

To use Firework AI as your OpenAI-compatible provider, see Example: Select AI with OpenAI-Compatible Providers.

Create and Set an AI Profile

Describes the steps to create and enable an AI profile.

Use `DBMS_CLOUD_AI.CREATE_PROFILE` to create an AI profile. Run `DBMS_CLOUD_AI.SET_PROFILE` to enable the AI profile so that you can use `SELECT AI` with a natural language prompt.

> **Note**
>
> You must run `DBMS_CLOUD_AI.SET_PROFILE` in each new stateful database session (connection) before you use `SELECT AI`. If you are using a stateless connection, you must use the `DBMS_CLOUD_AI.GENERATE` function which enables you to specify the profile name in each invocation.

The following example with the OpenAI provider creates an `AI` profile called `OPENAI` and sets the `OPENAI` profile for the current user session.

```
-- Create AI profile
--BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
  profile_name   => 'OPENAI',
  attributes     =>'{"provider": "openai",
            "credential_name": "OPENAI_CRED",
            "object_list": [{"owner": "SH", "name": "customers"},
                      {"owner": "SH", "name": "sales"},
                      {"owner": "SH", "name": "products"},
                      {"owner": "SH", "name": "countries"}]
       }');
END;
/

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.set_profile('OPENAI');

PL/SQL procedure successfully completed.
```

8

Use AI Keyword to Enter Prompts

Use `AI` as the keyword in a `SELECT` statement for interacting with the database using natural language prompts.

The `AI` keyword in a `SELECT` statement instructs the SQL execution engine to use the LLM identified in the active AI profile to process natural language and to generate SQL.

You can use the `AI` keyword in a query with Oracle clients such as SQL Developer, OML Notebooks, and third-party tools, to interact with database in natural language.

> **Note**
>
> You cannot run PL/SQL statements, DDL statements, or DML statements using the `AI` keyword.

Syntax

The syntax for running AI prompt is:

```
SELECT AI action natural_language_prompt
```

Parameters

The following are the parameters available for the *action* parameter:

Parameter	Description
`runsql`	Runs the underlying SQL command for the natural language prompt. This is the default action and it is optional to specify this parameter.
`showsql`	Displays the SQL statement for a natural language prompt.
`explainsql`	Explains the generated SQL from the prompt in a natural language. This option sends the generated SQL to the AI provider to produce a natural language explanation.
`narrate`	Sends the result of the SQL query run by the database back to the LLM to generate a natural language description of that result. When a vector index is specified in the AI profile to enable RAG, the system uses the specified transformer (or default transformer) model to create a vector embedding from the prompt for semantic similarity search against the vector store. The system then adds the retrieved content from the vector store to the user prompt and sends it to the LLM to generate a response based on this information. If you do not want table data or vector search documents to be sent to an LLM, a user with administrator privileges can disable such access for all users of the given database. This, in effect, disables the `narrate` action.

Parameter	Description
`chat`	Passes the user prompt directly to the LLM to generate a response, which is provided to the user. If `conversation` in the `DBMS_CLOUD_AI.CREATE_PROFILE` function is set to `true`, this option includes content from prior interactions or prompts, potentially including schema metadata.

Usage Notes

- The `AI` keyword for Select AI is not supported in Database Actions or APEX Service. You can use only the `DBMS_CLOUD_AI.GENERATE` function.

  Do not use `DBMS_CLOUD_AI.SET_PROFILE` in Database Actions or APEX Service. Instead, set the AI profile in the `profile_name` argument of `DBMS_CLOUD_AI.GENERATE`.
- The `AI` keyword is supported only in a `SELECT` statement.
- You cannot run PL/SQL statements, DDL statements, or DML statements using the `AI` keyword.
- The sequence is `SELECT` followed by `AI`. These keywords are not case-sensitive. After setting your AI profile using `DBMS_CLOUD_AI.SET_PROFILE`, and optional action, the text after `SELECT AI` is a natural language prompt. If an AI profile is not set, `SELECT AI` reports the following error when running a `SELECT AI` statement:

  ```
  ORA-00923: FROM keyword not found where expected
  00923. 00000 -  "FROM keyword not found where expected"
  ```

- Special character usage rules apply according to Oracle guidelines. For example, use single quotes twice if you are using an apostrophe in a sentence.

  ```
  select ai how many customers in SF don''t own their own home
  ```

- LLMs are subject to *hallucinations* and results are not always correct:
  - It is possible that `SELECT AI` may not be able to run the generated SQL query for a specific natural language prompt.
  - It is possible that `SELECT AI` may not be able to generate a SQL query for a specific natural language prompt.
  - It is possible that the LLM may not generate a SQL query that produces a correct result given your natural language prompt.

  In such a scenario, `SELECT AI` may respond with information to assist you in generating valid SQL.
- For better results with natural language to SQL generation, use database views or tables with contextual column names or consider adding column comments explaining values stored in the columns.
- Use the `explainsql` action, as in `SELECT AI explainsql`, to get a more detailed explanation of the SQL statement, as compared to `SELECT AI showsql`.
- To access DBA or USER views, see #unique_77.

9

Select AI Features

Select AI enables conversations for intuitive way to work with your data, enhances response accuracy with Retrieval Augmented Generation (RAG), and generates realistic synthetic data for testing, development, and training.

10

Select AI Conversations

Conversations in Select AI refer to the interactive dialogue between the user and the system, where a sequence of user-provided natural language prompts are stored and managed to support long-term memory for LLM interactions.

Select AI supports short-term, session-based conversations, which are enabled in the AI profile, as well as long-term, named conversations, which are enabled using specific procedures or functions and conversation IDs.

Types of Conversations

Select AI supports session-based short-term conversations and customizable conversations.

Session-based Short-Term Conversations: Select AI includes session-based short-term conversations to generate context-aware responses for the current prompt based on prior interactions.

You can enable it by setting the `conversation` attribute to `true|false` in your AI profile. Unlike the multiple conversation feature, session-based conversations store prompts only for the duration of the session. Prompts are stored in a temporary table, automatically dropped when the session ends, and cannot be reused and switched among conversations.

Customizable Long-Term Conversations: Select AI supports creating and using customizable conversations enabling you to use Select AI with different topics without mixing context, improving both flexibility and efficiency. You can create, set, delete, and update conversations through the `DBMS_CLOUD_AI` conversation procedures and functions. When you enable conversations, Select AI retrieves prompt history and sends them to the LLM to generate a response for the current prompt. These responses are stored in a persistent table for future use.

> **Note**
>
> Select AI Conversations support the following actions: `runsql`, `showsql`, `explainsql`, `narrate`, and `chat`.

How to Use Customizable Conversations

You can use Select AI for customizable conversations in the following ways:

- Set the conversation ID in the session using the `DBMS_CLOUD_AI.SET_CONVERSATION_ID` procedure, and run `SELECT AI <ACTION> <PROMPT>`.
- Pass the `conversation_id` in the `params` argument of the `DBMS_CLOUD_AI.GENERATE` function.

> **Note**
>
> If you use both multiple conversations and the `conversation: [true|false]` setting, the system ignores the `conversation` setting.

In the context of SQL query generation (NL2SQL), conversations enable a more intuitive and accessible way to work with your data, making it easier for users to extract insights and perform tasks without needing deep technical knowledge of SQL. Conversations can also be used with chat and RAG capabilities of Select AI.

For example, conversations provide an intuitive way to analyze data:

- Ask a question such as "What are the total number of customers"
- Follow up with context aware questions like:
  - "break out count of customers by country"
  - "What age group is most common"
  - "keep the top 5 customers and their country by their purchases and include a rank in the result"

To enable conversations, see Example: Enable Conversations in Select AI for a complete example.

Session-Based Conversations vs Customizable Conversations

The following table compares the session-based conversation and the customizable conversations in Select AI:

Questions	Session-Based Short-Term Conversations	Customizable Long-Term Conversations
When do I use?	Best for quick, temporary chats where you want the model to remember recent questions and answers during a single session. It's useful when you don't need to save or reuse the conversation later.	Designed for longer or ongoing conversations that may span multiple sessions. It's useful when you want to track, review, or manage the conversation history, or when different parts of an application need to access the same conversation context over time.
How do I enable?	Set `{"conversation": true or false}` in your AI profile.	Use the DBMS_CLOUD_AI.SET_CONVERSATION_ID procedure or the DBMS_CLOUD_AI.GENERATE function.
How many conversations are permitted?	One.	You can create multiple conversations. If you explicitly specify conversation IDs, you can alternate between them to associate prompts with the appropriate conversations as needed.

Questions	Session-Based Short-Term Conversations	Customizable Long-Term Conversations
Where are prompts stored and for how long?	Prompts are stored in a temporary table and dropped when the session ends.	Prompts are stored in a permanent table. Prompts are retained in the database for the number of days specified by the `retention_days` parameter in the DBMS_CLOUD_AI.CREATE_CONVERSATION procedure. After the retention period, the conversation and its prompts are automatically deleted. You can also manually delete prompts using the DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT procedure.
How many prompts are stored and how many prompts are sent to the LLM?	A maximum of 10 prompts are stored and sent to the LLM. You cannot customize this limit.	All prompts are stored. By default, the system sends the 10 most recent prompts to the LLM. You can customize this using the `conversation_length` parameter. See #unique_104.
Can I delete individual prompts?	No, you cannot delete individual prompt manually.	You can delete specific individual prompts by using the prompt id as specified in the #unique_105 and using the DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT procedure.
Are AI profiles used for retrieval of conversations?	Yes, Select AI retrieves prompts and responses previously generated using the same AI profile.	No, Select AI tracks the AI profile used when storing prompts and responses but does not restrict their retrieval based on that profile. It sends all conversation histories to the LLM to guide response generation, regardless of the profile used to generate them.
Where can I check history of prompts?	Prompts are saved in a temporary table under CLOUD USER but are not accessible for querying.	You can query and review conversations and prompts through the `DBMS_CLOUD_AI` conversation views. See #unique_77 for details.

11

Select AI with Retrieval Augmented Generation (RAG)

Select AI with RAG augments your natural language prompt by retrieving content from your specified vector store using semantic similarity search. This reduces hallucinations by using your specific and up-to-date content and provides more relevant natural language responses to your prompts.

Select AI automates the Retrieval Augmented Generation (RAG) process. This technique retrieves data from enterprise sources using AI vector search and augments user prompts for your specified large language model (LLM). By leveraging information from enterprise data stores, RAG reduces hallucinations and generates grounded responses.

RAG uses AI vector search on a vector index to find semantically similar data for the specified question. Vector store processes vector embeddings, which are mathematical representations of various data points like text, images, and audio. These embeddings capture the meaning of the data, enabling efficient processing and analysis. For more details on vector embeddings and AI vector search, see Overview of AI Vector Search.

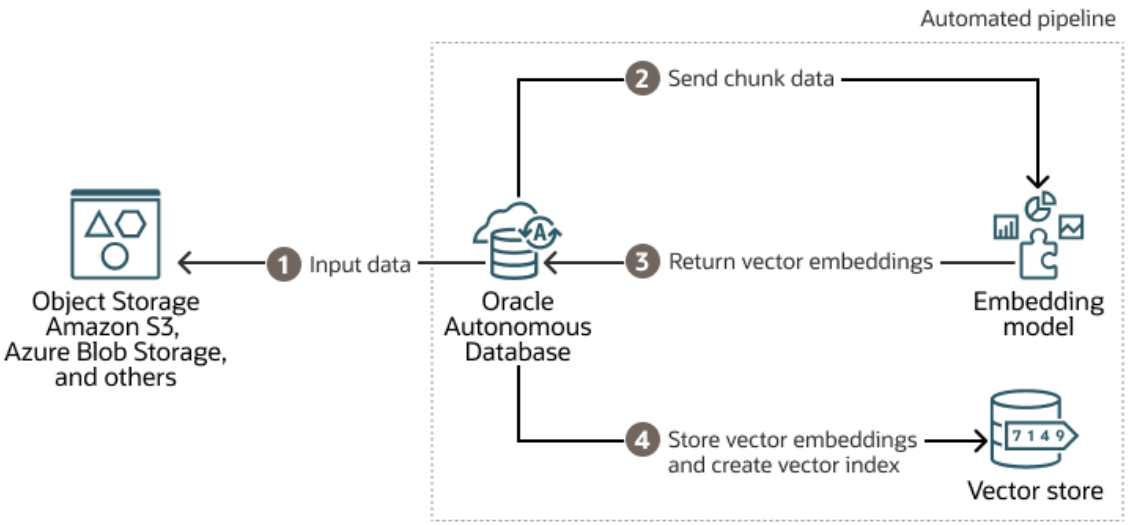
Select AI integrates with AI vector search available in Oracle Autonomous AI Database 26ai for similarity search using vector embeddings.

Topics

Build your Vector Store

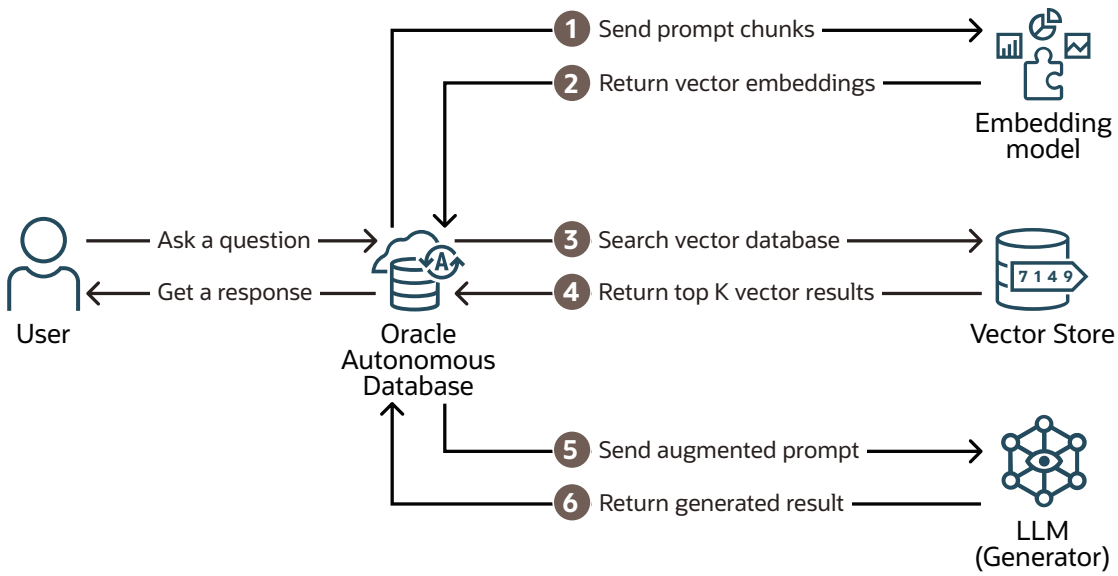
Select AI automates the creation and population of vector store by converting input documents (for example, PDF, DOC, JSON, XML, or HTML) from your object store to plain text. Oracle Text supports around 150 file types. For a complete list of all the supported document formats, see Supported Document Formats.
Select AI automatically processes documents to chunks, generates embeddings, stores them in the specified vector store, and updates the vector index as new data arrives.

Here is how the input from Object Storage is used with Select AI RAG:



1. Input: Data is initially stored in an Object Storage.
2. Oracle Autonomous Database retrieves the input data or the document, chunks it and sends the chunks to an embedding model.
3. The embedding model processes the chunk data and returns vector embeddings.
4. The vector embeddings are then stored in a vector store for use with RAG. As content is added, the vector index is automatically updated.

RAG retrieves relevant pieces of information from the enterprise database to answer a user's question. This information is provided to the specified large language model along with the user prompt. Select AI uses this additional enterprise information to enhance the prompt, improving the LLM's response. RAG can enhance response quality with update-to-date enterprise information from the vector store.



Select AI implements RAG as follows:

1. Input: User asks a question (specifies a prompt) using Select AI `narrate` action.

2. Select AI generates vector embeddings of the prompt using the embedding model specified in the AI profile.
3. The vector search index uses the vector embedding of the question to find matching content from the customer's enterprise data (searching the vector store) which has been indexed.
4. The vector search returns top K texts similar to the input to your Autonomous AI Database instance.
5. Autonomous AI Database then sends these top K query results with user question to the LLM.
6. The LLM returns its response to your Autonomous AI Database instance.
7. Autonomous AI Database Select AI provides the response to the user.

Use DBMS_CLOUD_AI to Create and Manage Vector Indexes

Use the `DBMS_CLOUD_AI` package to create and manage vector indexes and configure vector database JSON parameters.

After you create credentials and provide network access to the vector database and the AI provider, your Autonomous AI Database instance uses AI profiles to configure access to LLMs. See Example: Set Up and Use Select AI with RAG for a complete example on setting it up and using it in Select AI statements.

> **Note**
>
> If you do not want table data or vector search documents to be sent to an LLM, a user with administrator privileges can disable such access for all users of the given database. This, in effect, disables the `narrate` action for RAG.

You can configure AI profiles for providers listed in Select your AI Provider and LLMs through the `DBMS_CLOUD_AI` package.

> **See Also**
>
> - Create a vector index: #unique_111.
> - Manage vector index profiles and other AI profiles: #unique_112.
> - Query vector index views: #unique_77.

Use In-database Transformer Models

Select AI RAG enables you to use pretrained ONNX transformer models that are imported into your database in Oracle AI Database 26ai instance for generating embedding vectors from document chunks and user prompts.

To learn more about importing a pretrained ONNX model into your Oracle AI Database 26ai instance, see

- Import Pretrained Models in ONNX Format for Vector Generation Within the Database

- Alternate Method to Import ONNX Models
- Pre-built Embedding Generation model for Oracle Database 26ai

> **Note**
>
> You must import a pretrained ONNX-format transformer model into Oracle AI Database 26ai instance to use Select AI RAG with imported in-database transformer model. You can also use other transformer models from supported AI providers.

See Example: Select AI with In-database Transformer Models to explore the feature.

Benefits of Select AI RAG

Simplify querying, enhance response accuracy with current data, and gain transparency by reviewing sources used by the LLM.

Select AI RAG offers the following benefits:

- Simplify data querying and increase response accuracy: Enable users to query enterprise data using natural language and provide LLMs with detailed context from enterprise data to generate more accurate and relevant responses, reducing instances of LLM hallucinations.
- Up-to-date information: Provide LLMs access to current enterprise information using vector stores, eliminating the need for costly, time-consuming fine-tuning of LLMs trained on static data sets.
- Seamless integration: Integrate with Oracle AI Vector Search for streamlined data handling and enhanced performance.
- Automated data orchestration: Automate orchestration steps with a fully managed Vector Index pipeline, ensuring efficient processing of new data.
- Understandable contextual results: Has access and retrieves the sources used by the LLM from vector stores, ensuring transparency and confidence in results. Views and extracts data in natural language text or JSON format for easier integration and application development.

12

Use Autonomous AI Database as an AI Proxy for Select AI

Select AI runs natively inside Oracle Autonomous AI Database, which can operate as an AI Proxy Database, also referred to as "sidecar", for local and external data sources (on-premises, cloud, or third-party). Using standard Oracle federation mechanisms such as Database Links and Cloud Links, Select AI generates federated SQL from natural language prompts across Oracle and non-Oracle systems.

Autonomous AI Database hosts act as a central metadata and processing layer for both local and external data sources. The AI Proxy Database controls distributed query processing while external systems remain authoritative for their data.

What Is an AI Proxy Database

An AI Proxy Database is an Autonomous AI Database instance that runs Select AI on behalf of local or external data sources. It does not own the data. Instead, it uses metadata exposed through Database Links and Cloud Links to interpret natural language requests and generate SQL that runs across distributed systems.

In this architecture, the AI Proxy Database:

- Hosts Select AI
- Uses the user-specified LLM to interpret intent and generate SQL
- Coordinates federated query execution
- Returns results as standard SQL result sets

This approach enables AI-driven analytics across heterogeneous data stores without migrating or duplicating data.

See Use Database Links with Autonomous Database and Use Cloud Links for Read Only Data Access on Autonomous AI Database for more details.

For example, a natural language query such as “*Show pending orders for Acme Corp*” retrieves customer data from a BigQuery table on Google Cloud and order details from an Amazon Redshift database on AWS. Select AI manages the joins, data locations, and query optimization, so you do not need to write SQL or move data manually. Select AI uses AI profiles, roles, and encryption to protect data and ensure compliance across linked databases. Using a fully managed database with Real Application Security (RAS), AI Proxy Database powered by Select AI ensures secure and efficient AI-based data access and analysis.

See:

- Example: Use Select AI with Database Links to Query Another Autonomous AI Database to explore using Select AI to query data on another Autonomous AI Database instance connected through Database Links.
- #unique_114 to explore using Select AI to query data on non-Oracle database connected through Database Links.
- #unique_115 to explore using Select AI to query data on another Autonomous AI Database instance connected through Cloud Links.

Topics

How Select AI Uses the AI Proxy Database

You create Database Links or Cloud Links to expose remote tables and views to the AI Proxy Database. These objects appear as mapped schemas inside Autonomous AI Database. Select AI reads the metadata from these mapped objects and builds an augmented prompt.

Select AI sends the augmented prompt to the user-specified LLM, which generates SQL that spans local and remote data sources. The AI Proxy Database runs the federated query, pushes processing to external systems, and completes joins or aggregations as needed. Results return as standard SQL result sets.

From the user's perspective, the query behaves as if it runs against a single system.

Database Links vs Cloud Links

The AI Proxy Database uses Database Links and Cloud Links to access external data sources and generate federated SQL through Select AI. Both mechanisms expose metadata to Select AI, but they serve different connectivity and governance needs.

Database Links connect the AI Proxy Database to external databases, Oracle and non-Oracle, using direct network connectivity and credentials. They support heterogeneous environments, including on-premises systems, legacy Oracle databases, and third-party databases such as PostgreSQL, MySQL, SQL Server, Amazon Redshift, Snowflake, and others. With Database Links, Select AI reads table and view metadata from remote schemas and generates federated NL2SQL that runs across distributed systems while data remains in its source database. See CREATE DATABASE LINK for more details.

Cloud Links connect one Autonomous AI Database to another in a controlled and secure way. A data owner chooses which tables or views to share and makes them available to other Autonomous AI Databases. Once shared, other databases can find and use this data without setting up usernames, passwords, wallets, or network connections. The data remains read-only and stays in the source database, while the sharing rules are centrally managed. Cloud Links are designed specifically for sharing data between Autonomous AI Databases in a simple and governed manner. See #unique_119 for more details.

In both cases, Select AI relies on exposed metadata, not physical data movement, to interpret natural language prompts and generate SQL. The AI Proxy Database coordinates query execution, while external systems remain authoritative for their data and enforce their own security controls.

Supported External Data Sources

When used as an AI Proxy Database, Autonomous AI Database can operate on data stored in a wide range of external systems, including but not limited to:

- Oracle Autonomous AI Database (including previous versions such as 19*c*)
- PostgreSQL
- MySQL
- SQL Server
- Azure SQL
- DB2
- Teradata
- Amazon Redshift
- Snowflake
- Databricks
- Salesforce

Support depends on network accessibility and the use of Database Links or Cloud Links. See #unique_120 for a complete list of supported non-Oracle databases.

Primary Use Case

Select AI converts natural language prompts into SQL that run across multiple databases.

Federated queries: Your data platform may include on-premises relational databases, Autonomous AI Database, data lakes, legacy systems, or third-party data stores. You can use Select AI to query alongside Autonomous AI Database through Database Links or Cloud Links as a sidecar.

The AI Proxy Database coordinates query running while external systems process their data in place. This keeps data in its original location and avoids replication, synchronization, or ETL.

Key Benefits:

- Keep data in source systems
- Avoid ETL, replication, and data movement
- Query Oracle and non-Oracle databases together
- Extend generative AI and NL2SQL to legacy and cloud systems
- Use a single Select AI interface across all data sources

13

Synthetic Data Generation

Generate synthetic data using random generators, algorithms, statistical models, and Large Language Models (LLMs) to simulate real data for developing and testing solutions effectively.

Synthetic data can be a powerful tool when developing and testing solutions, especially when actual data doesn't yet exist or isn't allowed to be used. Synthetic, or artificially generated, data can have many of the characteristics of real data. Synthetic data is typically created using random generators, algorithms, or statistical models to simulate the characteristics and distributions of real data. However, this can be complex to produce or rely on tools with features of varying sophistication. With the availability of Large Language Models (LLMs), more relevant and schema-specific data may be generated that considers characteristics expressed in natural language.

Topics

Benefits of Synthetic Data Generation

Synthetic data generation enables populating database metadata clones, supporting development, testing, and machine learning projects without using sensitive data from original tables.

Synthetic Data Generation offers the following benefits:

- **Populating metadata clones with synthetic data**: A metadata clone replicates the structure of a database or schema without including actual data. Select AI allows synthetic data generation to populate these clones, protecting sensitive data while enabling development, testing, and creating templates. This approach supports performance and scalability testing.
- **Starting new projects**: When starting a new project, actual data may not be available. Synthetic data provides realistic samples to help demonstrate concepts and gain support for project proposals.
- **Validating user experience**: Synthetic data aids in testing user interfaces by providing diverse data sets to uncover design flaws, performance, and scalability issues.
- **Supporting AI and machine learning projects**: Synthetic data is useful for training AI and machine learning models when real data is unavailable or restricted. LLMs can generate data with specific patterns to facilitate model training and scoring.

Generate Synthetic Data

Use `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` function to generate synthetic data and query the data using Select AI actions.

To learn more, see #unique_124 and Example: Generate Synthetic Data.

Monitor and Troubleshoot Synthetic Data Generation

When generating large amounts of data across many tables, Select AI splits synthetic data generation tasks into smaller chunks and runs tasks in parallel. The status of each chunk is tracked in the `SYNTHETIC_DATA$<operation_id>_STATUS` table.

Synthetic data generation operations are logged in the tables `DBA_LOAD_OPERATIONS` and `USER_LOAD_OPERATIONS`. Use these tables to monitor the `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` operation. See #unique_126 for more details.

After running synthetic data generation in a given session, you can get the latest *<operation_id>* from `USER_LOAD_OPERATION` using the following:

```
SELECT max(id) FROM user_load_operations;
```

To view the synthetic data operations running in a different session, use the `DBA_LOAD_OPERATIONS` view.

View Status of Synthetic Data Generation

The status table of synthetic data generation operation shows the progress of each table and its corresponding chunk. The `STATUS_TABLE` column in `USER_LOAD_OPERATIONS` or `DBA_LOAD_OPERATIONS` shows the status table name. The table name is `SYNTHETIC_DATA$<operation_id>_STATUS` and it has following columns:

Name	Datatype	Description
`ID`	`NUMBER`	Unique identifier of the record.
`NAME`	`VARCHAR2`	Qualified name of the table, such as `"ADB_USER"."EMPLOYEES"`
`BYTES`	`NUMBER`	Desired number of records for this data generation task
`ROWS_LOADED`	`NUMBER`	Actual number of records generated.
`CHECKSUM`	`VARCHAR2`	Starting value for the primary key during this data generation task.
`LAST_MODIFIED`	`TIMESTAMP WITH TIME ZONE`	Timestamp indicating when the record was last modified.
`STATUS`	`VARCHAR2`	Status of the data generation task. The valid values are: • `COMPLETED` • `FAILED` • `PENDING` • `SKIPPED`
`ERROR_CODE`	`NUMBER`	Error code, if the data generation task fails.

Name	Datatype	Description
ERROR_MESSAGE	VARCHAR2	Error message provided if the task fails.
END_TIME	TIMESTAMP WITH TIME ZONE	Timestamp marking the end of the data generation task.

Example: Check the Number of Records Generated for Each Table
To check the number of records generated for each table, issue the following:

```
SELECT name, SUM(rows_loaded) FROM synthetic_data$<operation_id>_status group
by name;
```

Query `ROWS_LOADED` to confirm how many number of rows are loaded for each chunk, and `SUM(ROWS_LOADED)` for rows for each table.

```
BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name => 'GENAI',
        object_list => '[{"owner": "ADB_USER", "name":
"DIRECTOR","record_count":150},
                        {"owner": "ADB_USER", "name":
"MOVIE_ACTOR","record_count":300},
                        {"owner": "ADB_USER", "name": "CLASSES",
"user_prompt":"all in fall semester","record_count":5},
                        {"owner": "ADB_USER", "name":
"ACTOR","record_count":220},
                        {"owner": "ADB_USER", "name":
"MOVIE","record_count":50}]'
    );
END;
/


-- Check loaded rows for each chunk
SQL> SELECT name, rows_loaded FROM synthetic_data$141_status order by name;
NAME                           ROWS_LOADED
---------------------------------------
"ADB_USER"."ACTOR"                     188
"ADB_USER"."ACTOR"                      32
"ADB_USER"."CLASSES"                     5
"ADB_USER"."DIRECTOR"                  150
"ADB_USER"."MOVIE"                      50
"ADB_USER"."MOVIE_ACTOR"                38
"ADB_USER"."MOVIE_ACTOR"               114
"ADB_USER"."MOVIE_ACTOR"               148



-- Check loaded rows for each table
SQL> SELECT name, SUM(rows_loaded) FROM synthetic_data$141_status group by
name;

NAME                      SUM(ROWS_LOADED)
```

```
-----------------------------------------
"ADB_USER"."DIRECTOR"                    150
"ADB_USER"."MOVIE_ACTOR"                 300
"ADB_USER"."CLASSES"                       5
"ADB_USER"."ACTOR"                       220
"ADB_USER"."MOVIE"                        50
```

14

Feedback

Select AI enables you to provide feedback to help improve your selected LLM's ability to generate more accurate SQL queries.

> **Note**
>
> This feature is available only on Oracle AI Database 26ai. You can use it alongside the existing Select AI actions: `runsql`, `showsql`, and `explainsql`. Ensure that your AI profile is configured for NL2SQL generation and not RAG.

You can provide feedback to improve the accuracy of the generated SQL through the `feedback` action or the `DBMS_CLOUD_AI.FEEDBACK` procedure. Select AI creates a default vector index named `<profile_name>_FEEDBACK_VECINDEX` with default attributes when you use the feedback feature for the first time. This index helps refine future generated SQL based on the feedback provided. See #unique_127 for more information.

> **Tip**
>
> Use the `DBMS_CLOUD_AI.FEEDBACK` procedure when your specified LLM fails to generate the correct SQL query or does not return the expected result from one of the NL2SQL `SELECT AI` actions. You can also use the `DBMS_CLOUD_AI.FEEDBACK` procedure to add the correct SQL directly to the vector table. This helps guide future SQL generation by serving as a reference for similar prompts.

The following types of feedback are accepted:

- Positive feedback: You approve and confirm the accuracy of the generated SQL. The system stores the confirmed query for future reference.
- Negative feedback: If the results from your NL2SQL `SELECT AI` action fails to generate the correct SQL, you can provide the necessary SQL query improvements using `SELECT AI feedback <feedback>` or using the `DBMS_CLOUD_AI.FEEDBACK` procedure by identifying errors in the query or providing guidance as to what was expected in prose. The system refines the query using an LLM and stores the refined version for future Select AI query generation. The stored refined queries are then used as hints and sent to the LLM as part of the augmented prompt.

See #unique_128 and Example: Select AI Feedback to learn more.

Topics

- Benefits of Feedback
  The feedback action and procedure in Select AI introduces a prompt tuning mechanism that enhances the accuracy of SQL query generation.

- How Do I Use Feedback
  Select AI enables you to provide feedback either by using the `feedback` action or by calling the `DBMS_CLOUD_AI.FEEDBACK` procedure.

Benefits of Feedback

The feedback action and procedure in Select AI introduces a prompt tuning mechanism that enhances the accuracy of SQL query generation.

The owner of the AI profile can provide feedback for generated SQL queries and the Select AI system *learns* from the user interaction over time. This learning involves amassing a repository of prompts and feedback content where vector search is used to identify prompts similar to your current prompt. The top matching examples are provided as metadata to the LLM as part of the augmented prompt. The following are the benefits:

- Can improve accuracy of SQL queries.
- Enables the owner of AI profile to provide feedback by confirming the correct queries or suggesting corrections by providing the semantic intention or business definitions to produce a correct query.
- Uses historical feedback as hints for future query generation thereby adapting to your needs.

How Do I Use Feedback

Select AI enables you to provide feedback either by using the `feedback` action or by calling the `DBMS_CLOUD_AI.FEEDBACK` procedure.

> **⚠ Caution**
>
> Do not use the feedback action in applications where multiple users share database sessions under a single database user that owns the AI profile. The AI profile owner should provide feedback only after confirming that the corrected query is appropriate for all users of that profile.

You can provide feedback by including the following:

- `SQL_TEXT`: Include the feedback in quotes for your current SQL query or you can get the `SQL_TEXT` for a particular query by querying the `V$MAPPED_SQL` view.
- `SQL_ID`: You can get the `SQL_ID` by querying the `V$MAPPED_SQL` view.
- Use the last generated SQL: Provide your feedback in natural language.

> **ⓘ Note**
>
> To use last sql, be sure to set server output off in Oracle SQL*Plus or Oracle SQLcl. You must have `READ` privilege on `sys.v_$session` and `v_$mapped_sql` tables.

```
GRANT READ ON SYS.V_$MAPPED_SQL TO ADB_USER;
GRANT READ ON SYS.V_$SESSION TO ADB_USER;
```

See #unique_128 and Example: Select AI Feedback to learn more.

15

Generate a Summary with Select AI

Select AI enables you to generate a summary of your text, especially large texts, generally supporting up to 1 GB using AI providers. You can extract key insights from texts or large files as per your specific needs. This feature uses the LLM specified in your AI profile to generate a summary for a given text.

Select AI offers summarization of your content in the following ways:

- Use `summarize` as the Select AI *action*. See Use AI Keyword to Enter Prompts and Example: Select AI Actions to learn more.
- Use the `DBMS_CLOUD_AI.SUMMARIZE` function to specify customizations such as summary length, format, extraction level for a summary and so on with summarization parameters. See Example: Select AI Summarize to learn more.
- Use the `DBMS_CLOUD_AI.GENERATE` function with the `summarize` action. See #unique_98 to learn more.

> **Note**
>
> - Select AI *summarize* is applicable for Autonomous Database 19*c* and Autonomous AI Database 26ai.
> - Select AI *summarize* supports content up to 1 GB.
> - Supported file formats: Files must be encoded in *UTF-8*.
>   - For Autonomous AI Database 26ai:
>
>   Select AI supports a broader range of file formats. For details, see Oracle Text Supported Document Formats.
>   - For Autonomous Database 19c:
>
>   Select AI supports only text files. The content must be in plain text and Select AI does not support *json*, *html*, *docx*, or *pdf*.

Topics

- Summarization Techniques
  You can summarize a large document by entering the full text as a prompt for the LLM. However, since LLMs have token limits, Select AI applies different techniques to summarize large documents.

Summarization Techniques

You can summarize a large document by entering the full text as a prompt for the LLM. However, since LLMs have token limits, Select AI applies different techniques to summarize large documents.

Select AI applies the following summarization techniques:

- Iterative refinement. See Iterative Refinement.
- MapReduce. See MapReduce.

Based a given LLM's maximum token size, Select AI can break large content into smaller, easier-to-handle chunks that fit within the LLM’s limits. Select AI can then use either the iterative refinement or map reduce technique to generate the summary. Select AI uses MapReduce as the default setting. See #unique_134 and #unique_135 for more details.

16

Translate

With Select AI, you can use generative AI from the OCI translation service to translate your text into the language of your choice.

You can use this feature alongside the existing Select AI RAG. You can combine `translate` with `DBMS_CLOUD_AI.GENERATE` or `narrate` to use generative AI for producing translated outputs in your preferred language.

See #unique_136, #unique_98, and Example: Select AI Translate to learn more.

Topics

- Benefits of Translate
  The translate feature in Select AI enables you to translate input text in different languages.
- How Do I Use Translate
  Select AI enables you to translate your text input into your preferred language by using the `translate` action or by calling the `DBMS_CLOUD_AI.GENERATE` or the `DBMS_CLOUD_AI.TRANSLATE` functions.

Benefits of Translate

The translate feature in Select AI enables you to translate input text in different languages.

The following are the benefits:

- Translate improves usability by converting text into your preferred language, enabling you to work with the database more effectively.
- Translate lets you convert documents in one language into a language that may be better suited for your selected embedding model, producing better vectors for use with RAG.
- Automatically translate output into your preferred language when using `DBMS_CLOUD_AI.GENERATE` or `NARRATE`.

How Do I Use Translate

Select AI enables you to translate your text input into your preferred language by using the `translate` action or by calling the `DBMS_CLOUD_AI.GENERATE` or the `DBMS_CLOUD_AI.TRANSLATE` functions.

To use the Select AI translation feature, you must have the appropriate IAM policy permissions to access Oracle Cloud Infrastructure Language services.

Grant the permission to use `ai-service-language-family` resource in your IAM policy. An example policy statement to grant permission to a user group in a specific compartment is:

```
allow group <your group name> to use ai-service-language-family in
compartment <your_compartment>
```

- If using Resource Principal credential, assign the permission to the Dynamic Group.

- If using Private Key credential, assign the permission to the User Group.

A Dynamic Group identifies resources such as databases or functions by matching their OCIDs or tags, while a User Group contains individual IAM users.

Use a dynamic group when the policy applies to OCI resources, and use a user group when the policy applies to human users. For detailed steps to create dynamic and user groups, see Managing Dymanic Groups.

See Language Policies for more information.

You can translate text using the following:

- Use `translate` as a Select AI action. Use `select ai translate <text>`. See Example: Select AI Actions.
- Provide `translate` as an action within the `DBMS_CLOUD_AI.GENERATE` function. See GENERATE Function.
- Use the `DBMS_CLOUD_AI.TRANSLATE` function. See TRANSLATE Function.

  See also Example: Select AI Translate to learn more.

17

Private Endpoint Access for Select AI Models

You can enable secure, private access to generative AI models by deploying Ollama or Llama.cpp behind a private endpoint within your Virtual Cloud Network (VCN). This architecture is designed for organizations that need to keep AI processing fully private. The setup isolates both the Autonomous AI Database Serverless and your AI model servers from the public internet using private subnets, security lists, and controlled routing.

The setup uses a jump server in a public subnet for secure SSH access, while the database and AI models run in private subnets connected through Internet Gateway, Service Gateway, and NAT Gateway.

You create a VCN, configure subnets and gateways, and set up security rules that allow only internal traffic. See Setting up a private endpoint for AI models using Ollama and Llama.cpp for more information. The document walks you through installing Ollama and Llama.cpp, configuring a private API endpoint using Nginx as a reverse proxy, and validating connectivity from Autonomous AI Database. This configuration ensures that all AI processing occurs privately within your network boundary, enabling Select AI to integrate model capabilities while keeping sensitive data secure and fully isolated.

18

Examples of Using Select AI

Explore integrating Oracle's Select AI with various supported AI providers to generate, run, and explain SQL from natural language prompts or chat with the LLM.

- Example: Select AI Actions
  These examples illustrate common Select AI actions.
- Example: Select AI with OCI Generative AI
  These examples show how you can access OCI Generative AI using your OCI API key or Resource Principal, create an AI profile, and generate, run, and explain SQL from natural language prompts or chat using the OCI Generative AI LLMs.
- Example: Select AI with OpenAI
  This example shows how you can use OpenAI to generate SQL statements from natural language prompts.
- Example: Select AI with Cohere
  This example shows how you can use Cohere to generate SQL statements from natural language prompts.
- Example: Select AI with Azure OpenAI Service
  The following examples shows how you can enable access to Azure OpenAI Service using your API key or use Azure OpenAI Service Principal, create an AI profile, and generate SQL from natural language prompts.
- Example: Select AI with Google
- Example: Select AI with Anthropic
- Example: Select AI with Hugging Face
- Example: Select AI with AWS
- Example: Select AI with OpenAI-Compatible Providers
- Example: Enable Conversations in Select AI
  These examples illustrates enabling conversations in Select AI.
- Example: Set Up and Use Select AI with RAG
  This example guides you through setting up credentials, configuring network access, and creating a vector index for integrating OCI Generative AI vector store cloud services with OpenAI using Oracle Autonomous AI Database.
- Example: Select AI with In-database Transformer Models
  This example demonstrates how you can import a pretrained transformer model that is stored in Oracle object storage into your Oracle AI Database 26ai instance and then use the imported in-database model in Select AI profile to generate vector embeddings for document chunks and user prompts.
- Example: Improve SQL Query Generation
  These examples demonstrate how comments, annotations, foreign key, and referential integrity constraints in database tables and columns can improve the generation of SQL queries from natural language prompts.

- Example: Use Select AI with Database Links to Query Another Autonomous AI Database
  This example shows how to set up a Database Link from Autonomous AI Database to the source database and use Select AI to generate SQL from natural language prompts. Select AI uses the metadata from the source database to generate SQL.
- Example: Generate Synthetic Data
- Example: Enable or Disable Data Access
  This example illustrates how administrators can control data access and prevent Select AI from sending actual schema tables to the LLM.
- Example: Select AI Feedback
  These examples demonstrate how you can use the `DBMS_CLOUD_AI.FEEDBACK` procedure and the different scenarios of involving the `feedback` action to provide feedback and improve subsequent SQL query generation.
- Example: Select AI Summarize
  These examples show how to use the `summarize` action and `DBMS_CLOUD_AI.SUMMARIZE` function. Also, customize the summary generation for your content using the function.
- Example: Select AI Translate
  These examples demonstrate how you can use the `translate` capability.
- Example: Restrict Table Access in AI Profile
  This example demonstrates how to restrict table access and instruct the LLM to use only the tables specified in the `object_list` of the AI profile.
- Example: Specify Case Sensitivity for Columns
  This example shows how you can set case sensitivity for columns in AI profile.

Example: Select AI Actions

These examples illustrate common Select AI actions.

The following example illustrates actions such as `runsql` (the default), `showsql`, `narrate`, `chat`, `explainsql`, `feedback`, and `summarize` that you can perform with `SELECT AI`. These examples use the `sh` schema with AI provider and profile attributes specified in the `DBMS_CLOUD_AI.CREATE_PROFILE` function. Use Select AI actions after setting your AI profile by using the `DBMS_CLOUD_AI.SET_PROFILE` procedure in the current session.

To generate a summary of your text, use `SELECT AI SUMMARIZE <TEXT>`.

```
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> select ai showsql how many customers exist;

RESPONSE
--------------------------------------------------
SELECT COUNT(*) AS total_customers
FROM SH.CUSTOMERS


SQL> select ai narrate how many customers exist;

RESPONSE
```

```
--------------------------------------------------
There are a total of 55,500 customers in the database.

SQL> select ai chat how many customers exist;

RESPONSE
--------------------------------------------------------------------------------
--
It is impossible to determine the exact number of customers that exist as it
con
stantly changes due to various factors such as population growth, new
businesses
, and customer turnover. Additionally, the term "customer" can refer to
individu
als, businesses, or organizations, making it difficult to provide a specific
num
ber.


SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.

-- Feedback on SQL Text
-- Negative feedback example:
SQL > select ai feedback for query "select ai showsql how many watch
histories in total", please use sum instead of count;
-- Positive feedback example:
SQL > select ai feedback for query "select ai showsql how many watch
histories in total", the sql query generated is correct;
```

```
-- Feedback on SQL ID
-- Negative feedback example:
SQL > select ai feedback please use sum instead of count for sql_id
1v1z68ra6r9zf;
-- Positive feedback example:
SQL > select ai feedback sql query result is correct for sql_id
1v1z68ra6r9zf;

-- If not specified, use default LASTAI SQL
-- To use default LASTAI sql, make sure that set server output off;
-- Negative feedback example:
SQL > select ai feedback please use ascending sorting for ranking;
-- Positive feedback example:
SQL > select ai feedback the result is correct;

SQL> SELECT AI SUMMARIZE
Like countless other people around the globe, I stream music, and like more
than six hundred million of them I mainly use Spotify. Streaming currently
accounts for about eighty per cent of the American recording industry’s
revenue, and in recent years Spotify’s health is often consulted as a measure
for the health of the music business over all. Last spring, the International
Federation of the Phonographic Industry reported global revenues of $28.6
billion, making for the ninth straight year of growth. All of this was
unimaginable in the two-thousands, when the major record labels appeared
poorly equipped to deal with piracy and the so-called death of physical media.
On the consumer side, the story looks even rosier. Adjusted for inflation, a

... (skipped 1000 rows in the middle)

Pelly writes of some artists, in search of viral fame, who surreptitiously use
social media to effectively beta test melodies and motifs, basically putting
together songs via crowdsourcing. Artists have always fretted about the
pressure to conform, but the data-driven, music-as-content era feels
different. “You are a Spotify employee at that point,” Daniel Lopatin, who
makes abstract electronic music as Oneohtrix Point Never, told Pelly. “If your
art practice is so ingrained in the brutal reality that Spotify has outlined
for all of us, then what is the music that you’re not making? What does the
music you’re not making sound like?” Listeners might wonder something similar.
What does the music we’re not hearing sound like?;


RESPONSE
--------------------------------------------------------------------------------
The music streaming industry, led by Spotify, has revolutionized the way
people consume music, with streaming accounting for 80% of the American
recording industry's revenue. However, this shift has also complicated the
lives of artists, who struggle to survive in a hyper-abundant present where
music is often valued for its convenience rather than its artistic merit.
Spotify's algorithms prioritize popularity and profitability over artistic
diversity, leading to a homogenization of music and a devaluation of the
labor that goes into creating it. Meanwhile, the company's executives reap
enormous profits, with CEO Daniel Ek's net worth rivaling that of the
wealthiest musicians. As music critic Liz Pelly argues, the streaming economy
raises important questions about autonomy, creativity, and the value of art
in a world where everything is readily available and easily accessible.
```

Example: Select AI with OCI Generative AI

These examples show how you can access OCI Generative AI using your OCI API key or Resource Principal, create an AI profile, and generate, run, and explain SQL from natural language prompts or chat using the OCI Generative AI LLMs.

> **Note**
>
> If you do not specify the `model_name` parameter, OCI Generative AI uses the default model as per the table in Select your AI Provider and LLMs. To learn more about the parameters, see Profile Attributes.

```
-- Create Credential with OCI API key
--
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'GENAI_CRED',
    user_ocid       => 'ocid1.user.oc1..aaaa...',
    tenancy_ocid    => 'ocid1.tenancy.oc1..aaaa...',
    private_key     => '<your_api_key>',
    fingerprint     => '<your_fingerprint>'
  );
END;
/

--
-- Create AI profile
--
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name
=>'GENAI',
      attributes   =>'{"provider":
"oci",
        "credential_name": "GENAI_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name":
"countries"},
                        {"owner": "SH", "name":
"supplementary_demographics"},
                       {"owner": "SH", "name":
"profits"},
                       {"owner": "SH", "name":
"promotions"},
                       {"owner": "SH", "name": "products"}]
       }');
END;
/

PL/SQL procedure successfully completed.

--
```

```
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('GENAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"GENAI"

--
-- Use AI
--
SQL> select ai how many customers exist;
Number of Customers
-------------------
              55500

SQL> select ai how many customers in San Francisco are married;
COUNT(DISTINCTC."CUST_ID")
--------------------------
                        28

SQL> select ai showsql how many customers in San Francisco are married;
RESPONSE


--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
SELECT COUNT(DISTINCT c."CUST_ID")
FROM "SH"."CUSTOMERS" c
JOIN "SH"."COUNTRIES" co ON c."COUNTRY_ID" = co."COUNTRY_ID"
WHERE c."CUST_CITY" = 'San Francisco' AND c."CUST_MARITAL_STATUS" = 'married'

SQL> select ai explainsql how many customers in San Francisco are married;
RESPONSE


Here is the Oracle SQL query to find the number of customers in San Francisco
who are married:
```
SELECT COUNT(*)
FROM "SH"."CUSTOMERS" c
WHERE c."CUST_CITY" = 'San Francisco'
AND c."CUST_MARITAL_STATUS" = 'Married';
```
Explanation:
* We use the `COUNT(*)` aggregate function to count the number of rows that
match the conditions.
```

```
* We specify the table alias `c` for the `"SH"."CUSTOMERS"` table to make the
query more readable.
* We use the `WHERE` clause to filter the rows based on two conditions:
    + `c."CUST_CITY" = 'San Francisco'`: We filter the rows where the
customer's city is San Francisco.
    + `c."CUST_MARITAL_STATUS" = 'Married'`: We filter the rows where the
customer's marital status is Married.
* The double quotes around the table and column names are used to make the
query case-sensitive, as required in Oracle SQL.
Note: The column names and table names are assumed to be case-sensitive, so
we enclose them in double quotes. If the column names and table names are not
case-sensitive, you can remove the double quotes.
Collapse
 This snippet was truncated for display; see it in full

SQL> select ai narrate how many customers in San Francisco are married;

RESPONSE
-----------------------------------------------------------
There are 28 customers in San Francisco who are married.


SQL> select ai chat what is Autonomous AI Database;

RESPONSE



An Autonomous AI Database is a type of database that uses artificial
intelligence (AI) and machine learning (ML) to automate many of the
administrative and maintenance tasks typically performed by a database
administrator (DBA). This allows the database to manage itself, without human
intervention, to a large extent.
Autonomous AI Databases are designed to be self-driving, self-securing, and
self-repairing, which means they can:
1. **Automate administrative tasks**: Such as provisioning, patching,
upgrading, and tuning, which frees up DBAs to focus on higher-level tasks.
2. **Optimize performance**: By automatically adjusting parameters, indexing,
and caching to ensure optimal performance and efficiency.
3. **Detect and respond to security threats**: By using AI-powered security
tools to identify and respond to potential security threats in real-time.
4. **Heal itself**: By automatically detecting and repairing errors,
corruption, or other issues that may arise.
5. **Scale up or down**: To match changing workload demands, without the need
for manual intervention.
The benefits of Autonomous AI Databases include:
1. **Increased efficiency**: By automating routine tasks, DBAs can focus on
more strategic activities.
2. **Improved performance**: Autonomous AI Databases can optimize performance
in real-time, leading to faster query response times and better overall
system performance.
3. **Enhanced security**: AI-powered security tools can detect and respond to
threats more quickly and effectively than human administrators.
4. **Reduced costs**: By minimizing the need for manual intervention,
Autonomous AI Databases can help reduce labor costs and improve resource
utilization.
```

```
5. **Improved reliability**: Autonomous AI Databases can detect and repair
errors more quickly, reducing downtime and improving overall system
reliability.
Oracle Autonomous AI Database is a popular example of an Autonomous AI
Database, which was introduced in 2018. Other vendors, such as Amazon,
Microsoft, and Google, also offer Autonomous AI Database services as part of
their cloud offerings.
In summary, Autonomous AI Databases are designed to be self-managing, self-
optimizing, and self-healing, which can lead to improved performance,
security, and efficiency, while reducing costs and administrative burdens.


--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.
--
--Drop the profile
--
EXEC DBMS_CLOUD_AI.DROP_PROFILE('GENAI');

PL/SQL procedure successfully completed.
```

Example: Select AI with OCI Generative AI Resource Principal

To use resource principal with OCI Generative AI, Oracle Cloud Infrastructure tenancy administrator must grant access for Generative AI resources to a dynamic group. See #unique_150 to provide access to a dynamic group.

Set the required policies to obtain access to all Generative AI resources. See Getting Access to Generative AI to know more about Generative AI policies.

- To get access to all Generative AI resources in the entire tenancy, use the following policy:

  ```
  allow group <your-group-name> to manage generative-ai-family in tenancy
  ```

- To get access to all Generative AI resources in your compartment, use the following policy:

  ```
  allow group <your-group-name> to manage generative-ai-family in
  compartment <your-compartment-name>
  ```

Connect as an administrator and enable OCI resource principal. See #unique_151 to configure the parameters.

> **Note**
>
> If you do not specify the `model_name` parameter, OCI Generative AI uses the default model as per the table in Select your AI Provider and LLMs. To learn more about the parameters, see #unique_34.

```
-- Connect as Administrator user and enable OCI resource principal.
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(provider  => 'OCI');
END;
/

--
-- Create AI profile
--
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name
=>'GENAI',
      attributes =>'{"provider":
"oci",
        "credential_name": "OCI$RESOURCE_PRINCIPAL",
        "object_list": [{"owner": "SH", "name": "customers"},
                         {"owner": "SH", "name":
"countries"},
                         {"owner": "SH", "name":
"supplementary_demographics"},
                        {"owner": "SH", "name":
"profits"},
                        {"owner": "SH", "name":
"promotions"},
                        {"owner": "SH", "name": "products"}]
       }');
END;
/

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('GENAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"GENAI"
```

```
--
-- Use AI
--
SQL> select ai how many customers exist;
Number of Customers
-------------------
              55500

SQL> select ai how many customers in San Francisco are married;
COUNT(DISTINCTC."CUST_ID")
--------------------------
                        28

SQL> select ai showsql how many customers in San Francisco are married;
RESPONSE


--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
SELECT COUNT(DISTINCT c."CUST_ID")
FROM "SH"."CUSTOMERS" c
JOIN "SH"."COUNTRIES" co ON c."COUNTRY_ID" = co."COUNTRY_ID"
WHERE c."CUST_CITY" = 'San Francisco' AND c."CUST_MARITAL_STATUS" = 'married'

SQL> select ai explainsql how many customers in San Francisco are married;
RESPONSE


Here is the Oracle SQL query to find the number of customers in San Francisco
who are married:
```
SELECT COUNT(*)
FROM "SH"."CUSTOMERS" c
WHERE c."CUST_CITY" = 'San Francisco'
AND c."CUST_MARITAL_STATUS" = 'Married';
```
Explanation:
* We use the `COUNT(*)` aggregate function to count the number of rows that
match the conditions.
* We specify the table alias `c` for the `"SH"."CUSTOMERS"` table to make the
query more readable.
* We use the `WHERE` clause to filter the rows based on two conditions:
    + `c."CUST_CITY" = 'San Francisco'`: We filter the rows where the
customer's city is San Francisco.
    + `c."CUST_MARITAL_STATUS" = 'Married'`: We filter the rows where the
customer's marital status is Married.
* The double quotes around the table and column names are used to make the
query case-sensitive, as required in Oracle SQL.
Note: The column names and table names are assumed to be case-sensitive, so
we enclose them in double quotes. If the column names and table names are not
case-sensitive, you can remove the double quotes.
Collapse
 This snippet was truncated for display; see it in full
```

```
SQL> select ai narrate how many customers in San Francisco are married;

RESPONSE
----------------------------------------------------------
There are 28 customers in San Francisco who are married.


SQL> select ai chat what is Autonomous AI Database;

RESPONSE



An Autonomous AI Database is a type of database that uses artificial
intelligence (AI) and machine learning (ML) to automate many of the
administrative and maintenance tasks typically performed by a database
administrator (DBA). This allows the database to manage itself, without human
intervention, to a large extent.
Autonomous AI Databases are designed to be self-driving, self-securing, and
self-repairing, which means they can:
1. **Automate administrative tasks**: Such as provisioning, patching,
upgrading, and tuning, which frees up DBAs to focus on higher-level tasks.
2. **Optimize performance**: By automatically adjusting parameters, indexing,
and caching to ensure optimal performance and efficiency.
3. **Detect and respond to security threats**: By using AI-powered security
tools to identify and respond to potential security threats in real-time.
4. **Heal itself**: By automatically detecting and repairing errors,
corruption, or other issues that may arise.
5. **Scale up or down**: To match changing workload demands, without the need
for manual intervention.
The benefits of Autonomous AI Databases include:
1. **Increased efficiency**: By automating routine tasks, DBAs can focus on
more strategic activities.
2. **Improved performance**: Autonomous AI Databases can optimize performance
in real-time, leading to faster query response times and better overall
system performance.
3. **Enhanced security**: AI-powered security tools can detect and respond to
threats more quickly and effectively than human administrators.
4. **Reduced costs**: By minimizing the need for manual intervention,
Autonomous AI Databases can help reduce labor costs and improve resource
utilization.
5. **Improved reliability**: Autonomous AI Databases can detect and repair
errors more quickly, reducing downtime and improving overall system
reliability.
Oracle Autonomous AI Database is a popular example of an Autonomous AI
Database, which was introduced in 2018. Other vendors, such as Amazon,
Microsoft, and Google, also offer Autonomous AI Database services as part of
their cloud offerings.
In summary, Autonomous AI Databases are designed to be self-managing, self-
optimizing, and self-healing, which can lead to improved performance,
security, and efficiency, while reducing costs and administrative burdens.

--
--Clear profile
--
BEGIN
```

```
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
--
--Drop the profile
--
EXEC DBMS_CLOUD_AI.DROP_PROFILE('GENAI');

PL/SQL procedure successfully completed.
```

Example: Specify a Different Region for OCI Generative AI Profile
This example shows specifying an OCI Generative AI supported region in your profile. See Regions with Generative AI.

```
BEGIN


DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name => 'GENAI',
          attributes   =>'{"provider": "oci",
            "object_list": [
                        {"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}
                        ]
            "region":"eu-frankfurt-1",
            "model": "meta.llama-3.3-70b-instruct",
            "credential_name": "GENAI_CRED",
            "oci_compartment_id": "ocid1.compartment.oc1..."}');
END;

/
```

Example: Select AI with OCI Generative AI Using Grok Model
This example demonstrates how you can use xAI's Grok models with OCI Generative AI support. Review Perform Prerequisites for Select AI.

```
--Create your AI Profile

BEGIN
    DBMS_CLOUD_AI.create_profile(
            profile_name =>'grok',
            attributes   =>'{"provider": "oci",
              "credential_name": "OCI_CRED",
              "object_list": [ {"owner": "SH"}],
              "oci_compartment_id": "ocid1.compartment.oc1..aaaaa...",
              "model":"xai.grok-3"
            }');
 END;
 /

PL/SQL procedure successfully completed.
```

```
--Set Profile

exec dbms_cloud_ai.set_profile('grok');

PL/SQL procedure successfully completed.


--Use Select AI

select ai how many customers exist;

TOTAL_CUSTOMERS
---------------
          55500

select ai how many customers in San Francisco are married;

TOTAL_MARRIED_CUSTOMERS
-----------------------
                     46

select ai showsql how many customers in San Francisco are married;

RESPONSE


------------------------------------------------------------
SELECT COUNT(*) AS total_married_customers
FROM "SH"."CUSTOMERS" c
WHERE UPPER(c."CUST_CITY") = UPPER('San Francisco')
AND UPPER(c."CUST_MARITAL_STATUS") = UPPER('married')


select ai explainsql how many customers in San Francisco are married;

RESPONSE
```

```
------------------
### Oracle SQL Query
```sql
SELECT COUNT(*) AS "Total_Married_Customers"
FROM "SH"."CUSTOMERS" "cust"
WHERE UPPER("cust"."CUST_CITY") = UPPER('San Francisco')
  AND UPPER("cust"."CUST_MARITAL_STATUS") = UPPER('married')
```

Detailed Explanation
1. **Table and Schema Naming**:
   - The table `"CUSTOMERS"` is referenced with its schema name `"SH"` as `"SH"."CUSTOMERS"`. This ensures that the query explicitly points to the correct schema and table, avoiding ambiguity.
   - A table alias `"cust"` is used for the `"CUSTOMERS"` table to make the query more readable and concise when referencing columns.

2. **Column Naming**:
   - The result of the `COUNT(*)` function is aliased as `"Total_Married_Customers"` for clarity and readability. This descriptive name indicates exactly what the count represents.
   - All column names (e.g., `"CUST_CITY"`, `"CUST_MARITAL_STATUS"`) are enclosed in double quotes to maintain case sensitivity as per Oracle's naming conventions when explicitly defined.

3. **String Comparison in WHERE Clause**:
   - The strings 'San Francisco' and 'married' in the question are not enclosed in double quotes. As per the provided rules, case-insensitive comparison is required.
   - Therefore, the `UPPER()` function is applied to both the column values (`"cust"."CUST_CITY"` and `"cust"."CUST_MARITAL_STATUS"`) and the literal strings ('San Francisco' and 'married') to ensure the comparison ignores case differences.
   - This means that records with values like 'SAN FRANCISCO', 'san francisco', or any other case variation of 'San Francisco' will match, and similarly for 'married'.

4. **Purpose of the Query**:
   - The query counts the total number of customers who are located in 'San Francisco' and have a marital status of 'married'.
   - The `COUNT(*)` function is used to return the total number of rows that satisfy the conditions specified in the `WHERE` clause.

5. **Readability**:
   - The query uses consistent formatting with indentation for the `WHERE` clause conditions to improve readability.
   - The use of a meaningful alias (`"cust"`) and a descriptive result column name (`"Total_Married_Customers"`) makes the query easier to understand at a glance.

This query will return a single number representing the count of married customers in San Francisco, handling case variations in the data appropriately.
```

```
select ai narrate what are the top 3 customers in San Francisco;

RESPONSE

--------------------------------------------------------------------------------
--
I'm showing you the top 3 customers from San Francisco, based on their unique
id
entification numbers, sorted from lowest to highest. Here's who they are:

- Lyndon Baltzer
- Mike Crocker
- Milburn Klemm



select ai chat what is Autonomous AI Database;

RESPONSE

-------------------
An **Autonomous AI Database** is a cloud-based database solution that uses
artificial intelligence (AI) and machine learning (ML) to automate many of
the routine tasks associated with managing and maintaining a database. It is
designed to operate with minimal human intervention, allowing organizations
to focus on data-driven insights and application development rather than
database administration. The concept is often associated with Oracle's
Autonomous AI Database, which was one of the first widely recognized
implementations, but other cloud providers like AWS, Microsoft Azure, and
Google Cloud also offer similar self-managing database services.

### Key Features of an Autonomous AI Database:
1. **Self-Driving:** Automates tasks such as provisioning, patching, tuning,
and upgrading without manual input. It uses AI to optimize performance and
adapt to workload changes in real-time.
2. **Self-Securing:** Automatically applies security updates, detects
```

```
vulnerabilities, and protects against threats using built-in mechanisms like
encryption and threat detection, reducing the risk of human error.
3. **Self-Repairing:** Identifies and resolves issues such as system failures
or performance bottlenecks autonomously, ensuring high availability and
minimizing downtime.
4. **Scalability:** Dynamically scales resources (compute and storage) up or
down based on demand, optimizing cost and performance.
5. **Data Management:** Supports various data types and workloads, including
transactional (OLTP), analytical (OLAP), and mixed workloads, often in a
single converged database environment.

### Benefits:
- **Reduced Costs:** Minimizes the need for dedicated database
administrators, lowering operational expenses.
- **Improved Performance:** AI-driven optimization ensures efficient query
execution and resource allocation.
- **Enhanced Security:** Automated security features reduce the likelihood of
breaches due to misconfigurations or delayed updates.
- **Increased Productivity:** Frees up IT staff to focus on innovation rather
than routine maintenance.

### Use Cases:
- Businesses requiring high availability and reliability for critical
applications.
- Organizations looking to modernize IT infrastructure with cloud-native
solutions.
- Data analytics and machine learning projects needing scalable, optimized
data storage.
- Environments where security and compliance are paramount, such as finance
or healthcare.

### Example:
Oracle Autonomous AI Database, available on Oracle Cloud, offers two primary
services:
- **Autonomous Transaction Processing (ATP):** Optimized for transactional
workloads with high performance and reliability.
- **Autonomous AI Lakehouse:** Designed for analytics and big data workloads
with automated data integration and optimization.

In essence, an Autonomous AI Database represents a shift toward intelligent,
self-managing data systems that leverage automation and AI to simplify
database operations, enhance security, and improve efficiency in a cloud
environment. If you have a specific provider or context in mind, let me know,
and I can dive deeper!
```

Example: Select AI with OCI Generative AI Using LLAMA Model

This example showcases the `chat` feature from OCI Generative AI. It highlights the model's capabilities through two prompts: analyzing customer comments to gauze their sentiment and generate an introductory paragraph on rock climbing.

```
BEGIN


DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name =>
```

```
'GENAI_CRED',
        user_ocid       => 'ocid1.user.oc1..aaa',
        tenancy_ocid    => 'ocid1.tenancy.oc1..aaa',
        private_key     => '<your_api_key>',
        fingerprint     => '<your_fingerprint>'
      );


END;

    /

    PL/SQL procedure successfully completed.

BEGIN


DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name => 'GENAI',
          attributes   =>'{"provider": "oci",
            "object_list": [
                        {"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}
                        ]
            "model": "meta.llama-3.3-70b-instruct",
            "oci_apiformat":"GENERIC",
            "credential_name": "GENAI_CRED",
            "oci_compartment_id": "ocid1.compartment.oc1..."}');

END;

    /

    PL/SQL procedure successfully completed.

--
--Set profile
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('GENAI');

    PL/SQL procedure successfully completed.

SQL> set linesize
150
SQL> SELECT AI chat what is the sentiment of this comment I am not going to
waste my time filling up this three page form. Lousy idea;
SQL>
    RESPONSE

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

```
    The sentiment of this comment is strongly negative. The user is
expressing frustration and annoyance with the idea of filling out a three-
page form, an
       d is explicitly stating that they consider it a "lousy idea". The use
of the phrase "waste my time" also implies that they feel the task is
unnecessary
       and unproductive. The tone is dismissive and critical.

SQL> SELECT AI chat Write an enthusiastic introductory paragraph on how to
get started with rock climbing with Athletes as the target audience;
    RESPONSE

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
    Rock climbing is an exhilarating and challenging sport that's perfect for
athletes looking to push their limits and test their strength, endurance,
       and mental toughness. Whether you're a seasoned athlete or just
starting out, rock climbing offers a unique and rewarding experience that
will have
       you hooked from the very first climb. With its combination of physical
and mental challenges, rock climbing is a great way to build strength, improve
       flexibility, and develop problem-solving skills. Plus, with the
supportive community of climbers and the breathtaking views from the top of
the climb,
       you'll be hooked from the very first climb. So, if you're ready to
take on a new challenge and experience the thrill of adventure, then it's
time to
       get started with rock climbing!
```

Using OCI Generative AI with the Default Model

The following example uses the default OCI Generative AI Chat Model. If you do not specify the `model_name` parameter, OCI Generative AI uses the default model as per the table in Select your AI Provider and LLMs.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name => 'OCI_DEFAULT',
      attributes   => '{"provider": "oci",
                        "credential_name": "OCI_CRED",
                        "object_list": [{"owner": "ADB_USER"}]
                        }');
END;
/
```

Using OCI Generative AI with Chat Model

The following example uses `cohere.command-r-plus-08-2024` as the OCI Generative AI Chat Model.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name => 'OCI_COHERE_COMMAND_R_PLUS',
      attributes   => '{"provider": "oci",
                        "credential_name": "OCI_CRED",
                        "object_list": [{"owner": "ADB_USER"}],
                        "model": "cohere.command-r-plus-08-2024"
```

```
                      }');
END;
/
```

Using OCI Generative AI with Chat Model Endpoint ID
The following example demonstrates how to specify the OCI Generative AI Chat Model endpoint ID instead of `model`. If you are using Meta Llama Chat Model endpoint ID, then specify `oci_apiformat` as `GENERIC`.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name => 'OCI_CHAT_ENDPOINT',
      attributes => '{"provider": "oci",
                      "credential_name": "OCI_CRED",
                      "object_list": [{"owner": "ADB_USER"}],
                      "oci_endpoint_id": "<endpoint_id>",
                      "oci_apiformat": "GENERIC"
                     }');
END;
/
```

Using OCI Generative AI with Chat Model OCID
This example demonstrates how to specify the OCI Generative AI Cohere Chat Model endpoint ID instead of `model`. If you are using Meta Llama Chat Model endpoint ID, then specify `oci_apiformat` as `GENERIC`.

```
BEGIN

  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name => 'OCI_CHAT_OCID',
      attributes   => '{"provider": "oci",
                        "credential_name": "OCI_CRED",
                        "object_list": [{"owner": "ADB_USER"}],
                        "model": "<model_ocid>",
                        "oci_apiformat": "COHERE"
                       }');
END;
/
```

Example: Select AI with OpenAI

This example shows how you can use OpenAI to generate SQL statements from natural language prompts.

> **Note**
>
> Only a DBA can run `EXECUTE` privileges and network ACL procedure.

```
--Grants EXECUTE privilege to ADB_USER
--
```

```
SQL> grant execute on DBMS_CLOUD_AI to ADB_USER;

-- Grant Network ACL for OpenAI endpoint
--
SQL> BEGIN
     DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.openai.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
     );
    END;
    /

PL/SQL procedure successfully completed.

--
-- Create Credential for AI provider
--
EXEC
DBMS_CLOUD.CREATE_CREDENTIAL(
CREDENTIAL_NAME   => 'OPENAI_CRED',
username          =>  'OPENAI',
password          =>  '<your_api_token>');

PL/SQL procedure successfully completed.

--
-- Create AI profile
--
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
  profile_name   => 'OPENAI',
  attributes     =>'{"provider":
"openai",
        "credential_name": "OPENAI_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
        "conversation": "true"
       }');
END;
/

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI');

PL/SQL procedure successfully completed.
```

```
--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"OPENAI"

--
-- Use AI
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


SQL> select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'


SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
```

```
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.

SQL> select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


SQL> select ai chat what is Autonomous AI Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous AI Database is a cloud-based database service provided by Oracle.
It is designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous AI Database uses machine
learning and automation to optimize performance, security, and availability,
allowing
users to focus on their applications and data rather than database
administration tasks. It offers both Autonomous Transaction Processing (ATP)
for
transactional workloads and Autonomous AI Lakehouse for analytical workloads.
Autonomous AI Database provides high performance, scalability, and
reliability, making it an ideal choice for modern cloud-based applications.

--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.

--
--Drop the profile
--
SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('OPENAI');

PL/SQL procedure successfully completed.
```

The following example shows specifying a different model in your AI profile:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
  profile_name   => 'OPENAI',
  attributes     =>'{"provider":
"openai",
        "credential_name": "OPENAI_CRED",
        "model": "gpt-3.5-turbo",
     "object_list": [{"owner": "SH", "name": "customers"},
                         {"owner": "SH", "name": "countries"},
                         {"owner": "SH", "name": "supplementary_demographics"},
                         {"owner": "SH", "name": "profits"},
                         {"owner": "SH", "name": "promotions"},
                         {"owner": "SH", "name": "products"}],
        "conversation": "true"
       }');
END;
/
```

Example: Select AI with Cohere

This example shows how you can use Cohere to generate SQL statements from natural language prompts.

> **Note**
>
> Only a DBA can run `EXECUTE` privileges and network ACL procedure.

```
--Grants EXECUTE privilege to ADB_USER
--
SQL>GRANT execute on DBMS_CLOUD_AI to ADB_USER;
--
-- Create Credential for AI provider
--
EXEC
DBMS_CLOUD.CREATE_CREDENTIAL(
CREDENTIAL_NAME   => 'COHERE_CRED',
username          =>  'COHERE',
password          =>  'your_api_token');

PL/SQL procedure successfully completed.

--
-- Grant Network ACL for Cohere endpoint
--
SQL> BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.cohere.ai',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
```

```
                                principal_type => xs_acl.ptype_db)
    );
     END;
     /
    /

PL/SQL procedure successfully completed.

--
-- Create AI profile
--
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
  profile_name   => 'COHERE',
  attributes     =>'{"provider": "cohere",
     "credential_name": "COHERE_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "sales"},
                        {"owner": "SH", "name": "products"},
                        {"owner": "SH", "name": "countries"}]
      }');
END;
/

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('COHERE');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"COHERE"

--
-- Use AI
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

--
--Clear the profile
--
BEGIN
```

```
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.
--
--Drop the profile
--

SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('COHERE');

PL/SQL procedure successfully completed.
```

The following example shows specifying a different model and custom attributes in your AI profile:

```
BEGIN
DBMS_CLOUD_AI.CREATE_PROFILE(
     profile_name => 'COHERE',
     attributes =>
      '{"provider": "cohere",
        "credential_name": "COHERE_CRED",
    "model": "cohere.command-a-03-2025",
        "object_list": [{"owner": "ADB_USER"}],
        "max_tokens":512,
        "stop_tokens": [";"],
        "temperature": 0.5,
        "comments": true
       }');
END;
/
```

Example: Select AI with Azure OpenAI Service

The following examples shows how you can enable access to Azure OpenAI Service using your API key or use Azure OpenAI Service Principal, create an AI profile, and generate SQL from natural language prompts.

```
-- Create Credential for AI integration
--
EXEC
DBMS_CLOUD.CREATE_CREDENTIAL(
CREDENTIAL_NAME   => 'AZURE_CRED',
username          =>  'AZUREAI',
password          =>  'your_api_token');

PL/SQL procedure successfully completed.

--
-- Grant Network ACL for OpenAI endpoint
--
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => '<azure_resource_name>.openai.azure.com',
```

```
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
       );
END;
/

PL/SQL procedure successfully completed.

--
-- Create AI profile
--
BEGIN

DBMS_CLOUD_AI.CREATE_PROFILE(
     profile_name=>
'AZUREAI',
     attributes=> '{"provider": "azure",
        "azure_resource_name": "<azure_resource_name>",
        "azure_deployment_name":
"<azure_deployment_name>"

        "credential_name": "AZURE_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
        "conversation": "true"
       }');
END;
/

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"AZUREAI"


--
-- Use AI
```

```
--
select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'


select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.
```

```
select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


select ai chat what is Autonomous AI Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous AI Database is a cloud-based database service provided by Oracle.
It is designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous AI Database uses machine
learning and automation to optimize performance, security, and availability,
allowing
users to focus on their applications and data rather than database
administration tasks. It offers both Autonomous Transaction Processing (ATP)
for transactional
workloads and Autonomous AI Lakehouse for analytical workloads. Autonomous AI
Database provides high performance, scalability, and reliability, making it
an ideal choice for modern cloud-based applications.

--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/

PL/SQL procedure successfully completed.
--
--Drop the profile
--
EXEC DBMS_CLOUD_AI.DROP_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.
```

The following example shows specifying a different model in your AI profile:

```
BEGIN

DBMS_CLOUD_AI.CREATE_PROFILE(

profile_name=>'AZUREAI',

      attributes=>'{"provider":
```

```
"azure",
        "credential_name": "AZURE$PA",
        "model": "gpt-3.5-turbo",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
    "azure_resource_name":
"<azure_resource_name>",
        "azure_deployment_name": "<azure_deployment_name>"
       }');
END;
/
```

Example: Select AI with Azure OpenAI Service Principal
Connect as a database administrator to provide access to Azure service principal authentication and then grant the network ACL permissions to the user (`ADB_USER`) who wants to use Select AI. To provide access to Azure resources, see #unique_96.

> **Note**
>
> Only a DBA user can run `EXECUTE` privileges and network ACL procedure.

```
-- Connect as ADMIN user and enable Azure service principal authentication.
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(provider  => 'AZURE',
                                         params    =>
JSON_OBJECT('azure_tenantid' value 'azure_tenantid'));
END;
/

-- Copy the consent url from cloud_integrations view and consents the ADB-S
application.
SQL> select param_value from CLOUD_INTEGRATIONS where param_name =
'azure_consent_url';
PARAM_VALUE
--------------------------------------------------------------------------------
--
https://login.microsoftonline.com/<tenant_id>/oauth2/v2.0/authorize?
client_id=<client_id>&response_type=code&scope=User.read

-- On the Azure OpenAI IAM console, search for the Azure application name and
assign the permission to the application.
-- You can get the application name in the cloud_integrations view.
SQL> select param_value from CLOUD_INTEGRATIONS where param_name =
'azure_app_name';
PARAM_VALUE
--------------------------------------------------------------------------------
--
ADBS_APP_DATABASE_OCID
```

```
--
-- Grant Network ACL for Azure OpenAI endpoint
--SQL> BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'azure_resource_name.openai.azure.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
       );
       END;
       /

PL/SQL procedure successfully completed.

--
-- Create AI profile
--
BEGIN

DBMS_CLOUD_AI.CREATE_PROFILE(

profile_name=>'AZUREAI',

      attributes=>'{"provider":
"azure",
        "credential_name": "AZURE$PA",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
        "azure_resource_name":
"<azure_resource_name>",
        "azure_deployment_name": "<azure_deployment_name>"
       }');
END;
/

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
```

```
"AZUREAI"

--
-- Use AI
--
select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'


select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.
```

```
Feel free to ask if you have more questions related to SQL or database in
general.

select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


select ai chat what is Autonomous AI Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous AI Database is a cloud-based database service provided by Oracle.
It is designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous AI Database uses machine
learning and automation to optimize performance, security, and availability,
allowing us
ers to focus on their applications and data rather than database
administration tasks. It offers both Autonomous Transaction Processing (ATP)
for transactional
workloads and Autonomous AI Lakehouse for analytical workloads. Autonomous AI
Database provides high performance, scalability, and reliability, making it
an ideal choice for modern cloud-based applications.

EXEC DBMS_CLOUD_AI.DROP_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.
```

Example: Select AI with Google

This example shows how you can use Google to generate, run, and explain SQL from natural language prompts or chat using the Google Gemini LLM.

The following example demonstrates using Google as your AI provider. The example demonstrates using your Google API signing key to provide network access, creating an AI profile, using Select AI actions to generate SQL queries from natural language prompts and chat responses.

```
--Grants EXECUTE privilege to ADB_USER
--
SQL> grant EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--
-- Create Credential for AI provider
--
SQL> BEGIN
```

```
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'GOOGLE_CRED',
        username    => 'GOOGLE',
        password    => '<your_api_key>'
      );
     END;
    /

PL/SQL procedure successfully completed.

--
-- Grant Network ACL for Google endpoint
--
SQL>
SQL> BEGIN
      DBMS_NETWORK_ACL_ADB_USER.APPEND_HOST_ACE(
        host => 'generativelanguage.googleapis.com',
        ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                    principal_name => 'ADB_USER',
                    principal_type => xs_acl.ptype_db)
       );
     END;
    /

PL/SQL procedure successfully completed.

--
-- Create AI profile
--
SQL> BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'GOOGLE',
        attributes   =>'{"provider": "google",
          "credential_name": "GOOGLE_CRED",
          "object_list": [{"owner": "SH", "name":
"customers"},
                          {"owner": "SH", "name": "countries"},
                          {"owner": "SH", "name": "supplementary_demographics"},
                          {"owner": "SH", "name": "profits"},
                          {"owner": "SH", "name": "promotions"},
                          {"owner": "SH", "name": "products"}]
         }');
     END;
   /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('GOOGLE');

PL/SQL procedure successfully completed.

--
-- Use AI
```

```
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


SQL> select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'


SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.

SQL> select ai narrate what are the top 3 customers in San Francisco;
```

```
RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


SQL> select ai chat what is Autonomous AI Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous AI Database is a cloud-based database service provided by Oracle.
It is designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous AI Database uses machine
learning and automation to optimize performance, security, and availability,
allowing
users to focus on their applications and data rather than database
administration tasks. It offers both Autonomous Transaction Processing (ATP)
for transactional
workloads and Autonomous Lakehouse for analytical workloads. Autonomous AI
Database provides high performance, scalability, and reliability, making it
an ideal choice for modern cloud-based applications.

--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.

--
--Drop the profile
--
EXEC DBMS_CLOUD_AI.DROP_PROFILE('GOOGLE');

PL/SQL procedure successfully
completed.
```

The following example shows specifying a different model in your AI profile:

```
BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'GOOGLE',
        attributes   =>'{"provider": "google",
          "credential_name": "GOOGLE_CRED",
          "model": "gemini-1.5-pro",
```

```
          "object_list": [{"owner": "SH", "name":
"customers"},
                         {"owner": "SH", "name": "countries"},
                         {"owner": "SH", "name": "supplementary_demographics"},
                         {"owner": "SH", "name": "profits"},
                         {"owner": "SH", "name": "promotions"},
                         {"owner": "SH", "name": "products"}]
                       }');
END;
/
```

Example: Select AI with Anthropic

This example shows how you can use Anthropic to generate, run, and explain SQL from natural language prompts or chat using the Anthropic Claude LLM.

The following example demonstrates using Anthropic as your AI provider. The example demonstrates using your Anthropic API signing key to provide network access, creating an AI profile, and using Select AI actions to generate SQL queries from natural language prompts and chat using the Anthropic Claude LLM.

See #unique_34 to supply the profile attributes.

```
--Grant EXECUTE privilege to ADB_USER

SQL>GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--
-- Create Credential for AI provider
--

SQL>BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'ANTHROPIC_CRED',
        username    => 'ANTHROPIC',
        password    => '<your api key>'
      );
    END;
     /

PL/SQL procedure successfully completed.


--
-- Grant Network ACL for Anthropic endpoint
--
SQL>BEGIN
      DBMS_NETWORK_ACL_ADB_USER.APPEND_HOST_ACE(
        host => 'api.anthropic.com',
        ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                    principal_name => 'ADB_USER',
                    principal_type => xs_acl.ptype_db)
       );
     END;
    /
```

```
PL/SQL procedure successfully completed.

--
-- Create AI profile
--
SQL>BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'ANTHROPIC',
        attributes   =>'{"provider": "anthropic",
          "credential_name": "ANTHROPIC_CRED",
          "object_list": [{"owner": "SH", "name":
"customers"},
                          {"owner": "SH", "name": "countries"},
                          {"owner": "SH", "name": "supplementary_demographics"},
                          {"owner": "SH", "name": "profits"},
                          {"owner": "SH", "name": "promotions"},
                          {"owner": "SH", "name": "products"}]
         }');
    END;
     /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL>EXEC DBMS_CLOUD_AI.SET_PROFILE('ANTHROPIC');

PL/SQL procedure successfully completed.

--
-- Use AI
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


SQL> select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'
```

```
SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.

SQL> select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


SQL> select ai chat what is Autonomous AI Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous AI Database is a cloud-based database service provided by Oracle.
It is designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous AI Database uses machine
learning and automation to optimize performance, security, and availability,
allowing
users to focus on their applications and data rather than database
```

```
administration tasks. It offers both Autonomous Transaction Processing (ATP)
for transactional
workloads and Autonomous AI Lakehouse for analytical workloads. Autonomous AI
Database provides high performance, scalability, and reliability, making it
an ideal choice for modern cloud-based applications.

--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.

--
--Drop the profile
--
EXEC DBMS_CLOUD_AI.DROP_PROFILE('ANTHROPIC');

PL/SQL procedure successfully completed.
```

The following example shows specifying a different model in your AI profile:

```
BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'ANTHROPIC',
        attributes   =>'{"provider": "anthropic",
          "credential_name": "ANTHROPIC_CRED",
          "model": "claude-3-opus-20240229",
          "object_list": [{"owner": "SH", "name":
"customers"},
                          {"owner": "SH", "name": "countries"},
                          {"owner": "SH", "name": "supplementary_demographics"},
                          {"owner": "SH", "name": "profits"},
                          {"owner": "SH", "name": "promotions"},
                          {"owner": "SH", "name": "products"}]
         }');
END;
/
```

Example: Select AI with Hugging Face

This example shows how you can use Hugging Face to generate, run, and explain SQL from natural language prompts or chat using the Hugging Face LLM.

The following example demonstrates using Hugging Face as your AI provider. The example demonstrates using your Hugging Face API signing key to provide network access, creating an AI profile, and using Select AI actions to generate SQL queries from natural language prompts and chat using the Hugging Face LLM.

```
--Grant EXECUTE privilege to ADB_USER

SQL>GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;
```

```
--
-- Create Credential for AI provider
--
SQL>BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'HF_CRED',
        username    => 'HF',
        password    => '<your_api_key>'
      );
    END;
     /

PL/SQL procedure successfully completed.

--
-- Grant Network ACL for Hugging Face endpoint
--
SQL>BEGIN
      DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'api-inference.huggingface.co',
        ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                    principal_name => 'ADB_USER',
                    principal_type => xs_acl.ptype_db)
       );
    END;
     /

PL/SQL procedure successfully completed.



--
-- Create AI profile
--
SQL>BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'HF',
        attributes   =>'{"provider": "huggingface",
          "credential_name": "HF_CRED",
          "object_list": [{"owner": "SH", "name":
"customers"},
                          {"owner": "SH", "name": "countries"},
                          {"owner": "SH", "name": "supplementary_demographics"},
                          {"owner": "SH", "name": "profits"},
                          {"owner": "SH", "name": "promotions"},
                          {"owner": "SH", "name": "products"}],
           "model" : "Qwen/Qwen2.5-72B-Instruct"
         }');
    END;
     /


PL/SQL procedure successfully completed.
```

```
--
-- Enable AI profile in current session
--
SQL>EXEC DBMS_CLOUD_AI.SET_PROFILE('HF');

PL/SQL procedure successfully completed.

--
-- Use AI
--SQL> select ai how many customers exist;

Customer_Count
--------------
         55500

SQL> select ai how many customers in San Francisco are married;

Married_Customers
-----------------
               46

SQL> select ai showsql how many customers in San Francisco are married;

RESPONSE
------------------------------------------------------------
SELECT COUNT("CUST_ID") AS "Married_Customers"
FROM "SH"."CUSTOMERS" "C"
WHERE "CUST_CITY" = 'San Francisco' AND "CUST_MARITAL_STATUS
" = 'Married'

SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
------------------------------------------------------------
To answer the question "How many customers in San Francisco are married?", we
need to query the "SH"."CUSTOMERS" table and filter the results based on the
city and marital status.
We will use table aliases to make the query more readable and ensure that the
string comparisons follow the specified rules.

Here is the Oracle SQL query:

```sql
SELECT COUNT("CUST_ID") AS "Married_Customers_in_San_Francisco"
FROM "SH"."CUSTOMERS" c
WHERE c."CUST_CITY" = 'San Francisco' -- Case insensitive comparison
  AND UPPER(c."CUST_MARITAL_STATUS") = UPPER('Married'); --
Case insensitive comparison
```

Explanation:

1. **Table Alias**:

RESPONSE

```

```
   - We use the alias `c` for the "SH"."CUSTOMERS" table to make the query
more readable.

2. **Column Selection**:
   - We select the count of "CUST_ID" to get the number of customers who meet
the criteria. We alias this count as "Married_Customers_in_San_Francisco" for
clarity.

3. **WHERE Clause**:
   - **City Filter**: `c."CUST_CITY" = 'San Francisco'`
     - Since 'San Francisco' is not in double quotes, we use a case-
insensitive comparison.
   - **Marital Status Filter**: `UPPER(c."CUST_MARITAL_STATUS") =
UPPER('Married')`
     - Since 'Married' is not in double quotes, we use a case-insensitive
comparison by applying the `UPPER` function to both sides of the comparison.

This query will return the number of customers in San Francisco who are
married.



SQL> select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
------------------------------------------------------------
The top 3 customers in San Francisco are:

- Hector Colven, with a customer ID of 8866.
- Milburn Klemm, with a customer ID of 1944.
- Gavin Xie, with a customer ID of 2579.

These customers have the highest total sales among all customers in San
Francisco.


SQL> select ai chat what is Autonomous AI Database;

RESPONSE
------------------------------------------------------------
An Autonomous AI Database is a type of database management system that uses
artificial intelligence (AI) and machine learning (ML) to automate many of
the routine tasks typically performed by database administrators (DBAs).
These tasks include provisioning, patching, backups, tuning, and security.
The goal of an Autonomous AI Database is to reduce the need for human
intervention, minimize errors, and optimize performance,thereby allowing
organizations to focus on more strategic activities.

### Key Features of Autonomous AI Databases:

RESPONSE
------------------------------------------------------------
1. **Automated Provisioning**: The database can be set up and configured
automatically, reducing the time and effort required to get it up and running.
2. **Self-Patching and Updates**: The database automatically applies security
patches and updates without downtime, ensuring that the system remains secure
```

```
and up-to-date.
3. **Self-Tuning**: The database continuously monitors its performance and
adjusts settings to optimize query execution and resource utilization.
4. **Self-Backup and Recovery**: Automated backup and recovery processes
ensure that data is protected and can be restored quickly in the event of a
failure.
5. **Security**: Advanced security features, including threat detection and
response, are built into the database to protect against cyber threats.
6. **Scalability**: The database can automatically scale resources up or down
based on demand, ensuring optimal performance and cost efficiency.
7. **Monitoring and Diagnostics**: Real-time monitoring and diagnostics help
identify and resolve issues before they impact performance.


RESPONSE
------------------------------------------------------------
### Benefits of Autonomous AI Databases:
- **Reduced Operational Costs**: By automating routine tasks, the need for
dedicated DBAs is reduced, lowering operational costs.
- **Improved Reliability**: Automated processes reduce the risk of human
error, leading to more reliable and consistent performance.
- **Enhanced Security**: Continuous monitoring and automated security
measures help protect against threats.
- **Faster Time to Market**: Automated provisioning and tuning allow
applications to be deployed more quickly.

RESPONSE
------------------------------------------------------------
- **Scalability and Flexibility**: The ability to scale resources
automatically ensures that the database can handle varying workloads
efficiently.

### Use Cases:
- **Cloud Applications**: Autonomous AI Databases are particularly useful in
cloud environments where scalability and reliability are critical.
- **Data Warehousing**: They can handle large volumes of data and complex
queries, making them ideal for data warehousing and analytics.

RESPONSE
------------------------------------------------------------
- **IoT and Real-Time Data Processing**: They can process and analyze real-
time data from IoT devices efficiently.
- **E-commerce**: They can handle high transaction volumes and ensure fast
response times for online shopping platforms.


### Examples of Autonomous AI Databases:
- **Oracle Autonomous AI Database**: One of the first and most well-known
Autonomous AI Databases, offering both transactional
 and data warehousing capabilities.
- **Amazon Aurora**: A managed relational database service that includes
automated scaling, patching, and backups.
- **Microsoft Azure SQL Database Managed Instance**: Provides a high level of
automation and management for SQL Server databases in the cloud.
- **Google Cloud Spanner**: A globally distributed, horizontally scalable
relational database that is highly available and consistent.
```

```
Autonomous AI Databases represent a significant advancement in database
technology, offering organizations a more efficient, secure, and cost-
effective way to manage their data.

--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.
--
--Drop the profile
--
SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('HF');

PL/SQL procedure successfully completed.
```

Example: Select AI with AWS

This example shows how you can use AWS to generate, run, and explain SQL from natural language prompts or chat using the models available with AWS.

The following example shows how to use AWS as the AI provider with Amazon Bedrock and its foundation models. The example shows creating AWS credentials, provide network access, creating an AI profile, and using Select AI actions to generate SQL queries from natural language prompts and chat using the AWS foundation models.

To use AWS, obtain access key, secret keys, and model ID. See Use AWS. Use the model ID as the `model` attribute in the `DBMS_CLOUD_AI.CREATE_PROFILE` procedure. You must specify the `model` attribute explicitly, as no default model is provided.

```
--Grant EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--
-- Create Credential for AI provider
--
BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'AWS_CRED',
        username    => '<your_AWS_access_key>',
        password    => '<your_AWS_secret_key>'
      );
END;
/

PL/SQL procedure successfully completed.


--
-- Grant Network ACL for AWS
--
```

```
BEGIN
      DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'bedrock-runtime.us-east-1.amazonaws.com',
        ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                    principal_name => 'ADB_USER',
                    principal_type => xs_acl.ptype_db)
       );
END;
/

PL/SQL procedure successfully completed.



--
-- Create AI profile
--
BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'AWS',
        attributes   =>'{"provider": "aws",
          "credential_name": "AWS_CRED",
          "object_list": [{"owner": "SH", "name":
"customers"},
                          {"owner": "SH", "name": "countries"},
                          {"owner": "SH", "name": "supplementary_demographics"},
                          {"owner": "SH", "name": "profits"},
                          {"owner": "SH", "name": "promotions"},
                          {"owner": "SH", "name": "products"}],
           "model" : "anthropic.claude-v2",
           "conversation" : "true"
          }');
END;
/


PL/SQL procedure successfully completed.


--
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('AWS');

PL/SQL procedure successfully completed.

--
-- Use AI
--

SELECT AI how many customers exist;
"RESPONSE"
"COUNT(*)"
55500
```

```
SELECT AI how many customers in San Francisco are married;
"RESPONSE"
"COUNT(*)"
46

SELECT AI showsql how many customers in San Francisco are married;
"RESPONSE"
"SELECT COUNT(*) AS "Number of Married Customers in San Francisco"
FROM "SH"."CUSTOMERS" C
WHERE UPPER(C."CUST_CITY") = UPPER('San Francisco')
AND UPPER(C."CUST_MARITAL_STATUS") = UPPER('Married')"

SELECT AI explainsql how many customers in San Francisco are married;

"RESPONSE""SELECT
COUNT(*) AS "Number of Married Customers in San Francisco"
FROM "SH"."CUSTOMERS" C
WHERE C."CUST_CITY" = 'San Francisco'
AND C."CUST_MARITAL_STATUS" = 'Married'

Explanation:
- Used table alias C for CUSTOMERS table
- Used easy to read column names like CUST_CITY, CUST_MARITAL_STATUS
- Enclosed table name, schema name and column names in double quotes
- Compared string values in WHERE clause without UPPER() since the values are
not in double quotes
- Counted number of rows satisfying the condition and aliased the count as
"Number of Married Customers in San Francisco""

SELECT AI narrate what are the top 3 customers in San Francisco;
"RESPONSE"
The top 3 customers in San Francisco ordered by credit limit in descending
order are:

1. Bert Katz
2. Madallyn Ladd
3. Henrietta Snodgrass

SELECT AI chat what is Autonomous AI Database;
"RESPONSE"
"An Autonomous AI Database is a cloud database service provided by Oracle
Corporation. Some key features of Oracle Autonomous AI Database include:

- Fully automated and self-driving - The database automatically upgrades,
patches, tunes, and backs itself up without any human intervention required.

- Self-securing - The database uses machine learning to detect threats and
automatically apply security updates.

- Self-repairing - The database monitors itself and automatically recovers
from failures and errors without downtime.

- Self-scaling - The database automatically scales compute and storage
resources up and down as needed to meet workload demands.

- Serverless - The database is accessed as a cloud service without having to
```

```
manually provision any servers or infrastructure.

- High performance - The database uses Oracle's advanced automation and
machine learning to continuously tune itself for high performance.

- Multiple workload support - Supports transaction processing, analytics,
graph processing, etc in a single converged database.

- Fully managed - Oracle handles all the management and administration of the
database. Users just load and access their data.

- Compatible - Supports common SQL and Oracle PL/SQL for easy migration from
on-prem Oracle databases.

So in summary, an Oracle Autonomous AI Database is a fully automated, self-
driving, self-securing, and self-repairing database provided as a simple
cloud service. The automation provides high performance, elasticity, and
availability with minimal human labor required."

--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.

--
--Drop the profile
--
EXEC DBMS_CLOUD_AI.DROP_PROFILE('AWS');

PL/SQL procedure successfully completed.
```

Example: Select AI with OpenAI-Compatible Providers

This example shows how you can use OpenAI-compatible providers to generate, run, and explain SQL from natural language prompts or chat using the models available with OpenAI-compatible providers.

The following example shows how to use Fireworks AI as an OpenAI-compatible provider. It demonstrates how to create credentials using your Fireworks AI API signing key, configure network access, create an AI profile, and use Select AI actions to generate SQL queries from natural language prompts and chat using the Fireworks AI LLM model.

To use Fireworks AI, specify `provider_endpoint` as an attribute in the `DBMS_CLOUD_AI.CREATE_PROFILE` procedure instead of the `provider` attribute. See Use OpenAI-Compatible Providers to obtain the attribute. You must specify the `model` attribute explicitly, as no default model is provided.

```
--Grant EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--
```

```
-- Create Credential for AI provider
--
BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'FIREWORKS_CRED',
        username    => 'FIREWORKS',
        password    => '<your_fireworksaiapi_key>'
      );
END;
/

PL/SQL procedure successfully completed.


--
-- Grant Network ACL for Fireworks AI endpoint
--
BEGIN
      DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'api.fireworks.ai',
        ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                    principal_name => 'ADB_USER',
                    principal_type => xs_acl.ptype_db)
       );
END;
/

PL/SQL procedure successfully completed.



--
-- Create AI profile
--
BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'FIREWORKS',
        attributes   =>'{
          "credential_name": "FIREWORKS_CRED",
          "object_list": [{"owner": "SH", "name":
"customers"},
                          {"owner": "SH", "name": "countries"},
                          {"owner": "SH", "name": "supplementary_demographics"},
                          {"owner": "SH", "name": "profits"},
                          {"owner": "SH", "name": "promotions"},
                          {"owner": "SH", "name": "products"}],
           "model" : "accounts/fireworks/models/llama-v3p1-405b-instruct",
           "provider_endpoint" : "api.fireworks.ai/inference",
           "conversation" : "true"
          }');
END;
/


PL/SQL procedure successfully completed.
```

```
--
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('FIREWORKS');

PL/SQL procedure successfully completed.

--
-- Use AI
--

select ai how many customers exist;
"RESPONSE"
"COUNT(*)"
55500


select ai how many customers in San Francisco are married;
"RESPONSE"
"COUNT(*)"
46

select ai showsql how many customers in San Francisco are married;

"RESPONSE"
"SELECT COUNT(*)
FROM ""SH"".""CUSTOMERS"" ""c""
WHERE UPPER(""c"".""CUST_CITY"") = UPPER('San Francisco')
  AND UPPER(""c"".""CUST_MARITAL_STATUS"") = UPPER('Married')"

select ai explainsql how many customers in San Francisco are married;

"RESPONSE"
"Here is the Oracle SQL query:

```sql
SELECT COUNT(*)
FROM ""SH"".""CUSTOMERS"" ""c""
WHERE UPPER(""c"".""CUST_CITY"") = UPPER('San Francisco')
  AND UPPER(""c"".""CUST_MARITAL_STATUS"") = UPPER('Married')
```

Explanation:

* We use the table alias `""c""` to refer to the `""CUSTOMERS""` table.
* We enclose the schema name `""SH""`, table name `""CUSTOMERS""`, and column
names `""CUST_CITY""` and `""CUST_MARITAL_STATUS""` in double quotes to
ensure case sensitivity.
* In the `WHERE` clause, we use the `UPPER()` function to perform case-
insensitive comparisons for the strings `'San Francisco'` and `'Married'`,
since they are not enclosed in double quotes.
* The `COUNT(*)` function returns the number of rows in the result set, which
represents the number of customers in San Francisco who are married."

select ai narrate what are the top 3 customers in San Francisco;
```

```
"RESPONSE"
The top 3 customers in San Francisco are:

1. Lyndon Baltzer
2. Mike Crocker
3. Milburn Klemm

These customers are identified by their unique customer IDs, which are used
to track their information and interactions. The list shows the customer's
first and last names.

select ai chat what is Autonomous AI Database;
"RESPONSE"
"An Autonomous AI Database is a type of database that uses artificial
intelligence (AI) and machine learning (ML) to automate many of the tasks
traditionally performed by database administrators (DBAs). The goal of an
Autonomous AI Database is to provide a self-managing, self-securing, and self-
repairing database that can operate with minimal human intervention.

Autonomous AI Databases use advanced algorithms and machine learning
techniques to:

1. **Automate database administration**: Tasks such as patching, backups, and
performance tuning are automated, freeing up DBAs to focus on higher-level
tasks.
2. **Predict and prevent issues**: The database uses machine learning to
predict and prevent issues such as downtime, data breaches, and performance
problems.
3. **Optimize performance**: The database continuously monitors and optimizes
its own performance, ensuring that it is running at peak efficiency.
4. **Enhance security**: Autonomous AI Databases use advanced security
features, such as encryption and access controls, to protect data from
unauthorized access.
5. **Improve data management**: Autonomous AI Databases can automatically
manage data, including data ingestion, processing, and storage.

The benefits of Autonomous AI Databases include:

1. **Increased efficiency**: By automating routine tasks, Autonomous AI
Databases can reduce the workload of DBAs and improve overall efficiency.
2. **Improved security**: Autonomous AI Databases can detect and respond to
security threats in real-time, reducing the risk of data breaches.
3. **Enhanced performance**: Autonomous AI Databases can optimize their own
performance, ensuring that applications run quickly and efficiently.
4. **Reduced costs**: By automating routine tasks and improving efficiency,
Autonomous AI Databases can help reduce costs associated with database
management.

Examples of Autonomous AI Databases include:

1. Oracle Autonomous AI Database
2. Microsoft Azure SQL Database
3. Amazon Aurora
4. Google Cloud SQL

Overall, Autonomous AI Databases represent a significant shift in the way
```

```
databases are managed and maintained, using AI and ML to automate many of the
tasks traditionally performed by DBAs."

--
--Clear the profile
--
BEGIN
   DBMS_CLOUD_AI.CLEAR_PROFILE;
END;
/
PL/SQL procedure successfully completed.

--
--Drop the profile
--
EXEC DBMS_CLOUD_AI.DROP_PROFILE('FIREWORKS');

PL/SQL procedure successfully completed.
```

Example: Enable Conversations in Select AI

These examples illustrates enabling conversations in Select AI.

Before You Begin
Review Perform Prerequisites for Select AI.

> **Note**
>
> A user with administrator privileges (ADMIN) must grant `EXECUTE` and enable network access control list (ACL).

Session-Based Conversations

Create your AI profile. Set the `conversation` attribute to `true` in the profile, this action includes content from prior interactions or prompts, potentially including schema metadata, and set your profile. Once the profile is enabled, you can begin having conversations with your data. Use natural language to ask questions and follow up as needed.

```
--Grants EXECUTE privilege to ADB_USER
--
SQL> grant execute on DBMS_CLOUD_AI to ADB_USER;

-- Grant Network ACL for OpenAI endpoint
--
SQL> BEGIN
     DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.openai.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
     );
    END;
```

```
    /

PL/SQL procedure successfully completed.

--
-- Create Credential for AI provider
--
EXEC
DBMS_CLOUD.CREATE_CREDENTIAL(
CREDENTIAL_NAME   => 'OPENAI_CRED',
username          =>  'OPENAI',
password          =>  '<your_api_token>');

PL/SQL procedure successfully completed.

--
-- Create AI profile
--
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
  profile_name   => 'OPENAI',
  attributes     =>'{"provider":
"openai",
        "credential_name": "OPENAI_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
        "conversation": "true"
       }');

END;
     /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"OPENAI"

--
```

```
-- Use AI
--
what are the total number of customers;

CUSTOMER_COUNT
--------------
         55500

break out count of customers by country;

RESPONSE
-----------------
COUNTRY_NAME             CUSTOMER_COUNT
Italy                 7780
Brazil                 832
Japan                 624
United Kingdom             7557
Germany            8173
United States of America     18520
France                 3833
Canada                 2010
Spain                 2039
China                 712
Singapore             597
New Zealand             244
Poland     708
Australia             831
Argentina             403
Denmark            383
South Africa             88
Saudi Arabia             75
Turkey                 91


what age group is most common;

RESPONSE
--------------------------------------------------------------------------------
--
AGE_GROUP     CUSTOMER_COUNT
65+         28226

select ai keep the top 5 customers and their country by their purchases and
include a rank in the result;

RESPONSE
--------------------------------------------------------------------------------
--
RANK     CUSTOMER_NAME         COUNTRY     PURCHASES
1     Abigail Ruddy         Japan         276
2     Abigail Ruddy         Italy         168
3     Abigail Ruddy         Japan         74
3     Abner Robbinette     Germany     74
5     Abner Everett         France         68
```

```
SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('OPENAI');

PL/SQL procedure successfully completed.
```

Customizable Conversations

The following examples demonstrate using the conversation management API supporting customizable conversations. To use Select AI for multiple conversations:

1. Create a conversation
2. Set the conversation in the current user session
3. Use `Select AI <action> <prompt>`

- Use `DBMS_CLOUD_AI.CREATE_CONVERSATION` function and then set the conversation using `DBMS_CLOUD_AI.SET_CONVERSATION_ID`.
- Call the `DBMS_CLOUD_AI.CREATE_CONVERSATION` procedure directly to create and set the conversation in one step.

Example: Create and Set Customizable Conversations
The following example demonstrates how to create a conversation using `DBMS_CLOUD_AI.CREATE_CONVERSATION` function and set it using the `DBMS_CLOUD_AI.SET_CONVERSATION_ID` procedure.

```
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION;  -- in 19c, run SELECT
DBMS_CLOUD_AI.create_conversation FROM dual;


CREATE_CONVERSATION
-----------------------------------
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92


EXEC DBMS_CLOUD_AI.SET_CONVERSATION_ID('30C9DB6E-EA4D-AFBA-
E063-9C6D46644B92');

PL/SQL procedure successfully completed
```

The following example demonstrates running the `DBMS_CLOUD_AI.CREATE_CONVERSATION` procedure to create and set the `conversation_id` directly.

```
EXEC DBMS_CLOUD_AI.create_conversation;

PL/SQL procedure successfully completed.
```

You can also customize the conversation attributes such as `title`, `description`, `retention_days`, and `conversation_length` attributes.

```
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION(
            attributes => '{"title":"My first conversation",
            "description":"this is my first conversation",
            "retention_days":5,
            "conversation_length":5}');
```

```
CREATE_CONVERSATION
------------------------------------
38F8B874-7687-2A3F-E063-9C6D4664EC3A
```

You can view if a certain conversation exists by querying `DBA/USER_CLOUD_AI_CONVERSATIONS` view.

```
-- Verify the setup
SELECT conversation_id, conversation_title, description, retention_days,
conversation_length FROM DBA_CLOUD_AI_CONVERSATIONS WHERE
conversation_id = '38F8B874-7687-2A3F-E063-9C6D4664EC3A';

CONVERSATION_ID
CONVERSATION_TITLE
DESCRIPTION                         RETENTION_DAYS
CONVERSATION_LENGTH
------------------------------------
-------------------------------------------
--------------------------------- -----------------------------
------------------
38F8B874-7687-2A3F-E063-9C6D4664EC3A     My first
conversation                            this is my first conversation
+00005 00:00:00.000000         5
```

You can also verify if a conversation is set by calling the `DBMS_CLOUD_AI.GET_CONVERSATION_ID` function.

```
SELECT DBMS_CLOUD_AI.GET_CONVERSATION_ID;


--------------------------------------------------------------------------------
--
30C9DB6E-EA4F-AFBA-E063-9C6D46644B92
```

Example: Use Customizable Conversations with Select AI

After you create and set the conversation and enable your AI profile, you can start interacting with your data. Use natural language to ask questions and follow up as needed.

Use `SELECT AI <ACTION> <PROMPT>`.

```
SELECT AI CHAT What is the difference in weather between Seattle and San
Francisco?;

RESPONSE
--------------------------------------------------------------------------------
--
Seattle and San Francisco are both located on the West Coast of the United
State
s, but they have distinct weather patterns due to their unique geography and
cli
mate conditions. Here are the main differences:

1. **Rainfall**: Seattle is known for its rainy reputation, with an average
annu
```

```
al rainfall of around 37 inches (94 cm). San Francisco, on the other hand,
recei
ves significantly less rainfall, with an average of around 20 inches (51 cm)
per
 year.
2. **Cloud Cover**: Seattle is often cloudy, with an average of 226 cloudy
days
per year. San Francisco is also cloudy, but to a lesser extent, with an
average
of 165 cloudy days per year.

......


SELECT AI CHAT Explain the difference again in one paragraph only.;

RESPONSE
--------------------------------------------------------------------------------
--
Seattle and San Francisco have different weather patterns despite both
experienc
ing a mild oceanic climate. San Francisco tends to be slightly warmer, with
aver
age temperatures ranging from 45?F to 67?F, and receives less rainfall,
around 2
0 inches per year, mostly during winter. In contrast, Seattle is cooler, with
te
mperatures ranging from 38?F to 64?F, and rainier, with around 37 inches of
rain
fall per year, distributed throughout the year. San Francisco is also known
for
its fog, particularly during summer, and receives more sunshine, around 160
sunn
y days per year, although it's often filtered through the fog. Overall, San
Fran
cisco's weather is warmer and sunnier, with more pronounced seasonal
variations,
 while Seattle's is cooler and rainier, with more consistent temperatures
throug
hout the year.
```

Example: Use GENERATE Function to Compare Two Conversations

The following example show how two conversations are used interchangeably to ask questions and verify accurate responses. Each conversation begins with a different question focused on comparison. Later, when you ask the same follow-up question in both conversations, each returns a different answer based on its prior context.

```
-- First conversation
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION;


CREATE_CONVERSATION
----------------------------------
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92
```

```
-- Second conversation
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION;


CREATE_CONVERSATION
-----------------------------------
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92


-- Call generate using the first conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
        prompt       =>  'What is the difference in weather between Seattle
and San Francisco?',
        profile_name =>  'GENAI',
        action       =>  'CHAT',
        params       =>  '{"conversation_id":"30C9DB6E-EA4D-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;


RESPONSE
--------------------------------------------------------------------------------
--
Seattle and San Francisco, both located in the Pacific Northwest and Northern
Ca
lifornia respectively, experience a mild oceanic climate. However, there are
som
e notable differences in their weather patterns:

1. **Temperature**: San Francisco tends to be slightly warmer than Seattle,
espe
cially during the summer months. San Francisco's average temperature ranges
from
 45?F (7?C) in winter to 67?F (19?C) in summer, while Seattle's average
temperat
ure ranges from 38?F (3?C) in winter to 64?F (18?C) in summer.

2. **Rainfall**: Seattle is known for its rainy reputation, with an average
annu
al rainfall of around 37 inches (94 cm). San Francisco receives less
rainfall, w
ith an average of around 20 inches (51 cm) per year. However, San Francisco's
ra
infall is more concentrated during the winter months, while Seattle's
rainfall i
s more evenly distributed throughout the year.

......

-- Call generate using the second conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
        prompt       =>  'How does the cost of living compare between New
York and Los Angeles?',
        profile_name =>  'GENAI',
        action       =>  'CHAT',
```

```
        params       =>  '{"conversation_id":"30C9DB6E-EA4E-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;


RESPONSE
--------------------------------------------------------------------------------
--
The cost of living in New York and Los Angeles is relatively high compared to
ot
her cities in the United States. However, there are some differences in the
cost
 of living between the two cities. Here's a comparison of the cost of living
in
New York and Los Angeles:

1. Housing: The cost of housing is significantly higher in New York than in
Los
Angeles. The median home price in New York is around $999,000, while in Los
Ange
les it's around $849,000. Rent is also higher in New York, with the average
rent
 for a one-bedroom apartment being around $3,000 per month, compared to
around $
2,400 per month in Los Angeles.

2. Food: The cost of food is relatively similar in both cities, with some
variat
ion in the cost of certain types of cuisine. However, eating out in New York
can
 be more expensive, with the average cost of a meal at a mid-range restaurant
be
ing around $15-20 per person, compared to around $12-18 per person in Los
Angele
s.

......


-- Call generate using the first conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
        prompt       =>  'Explain the difference again in one paragraph
only.',
        profile_name =>  'GENAI',
        action       =>  'CHAT',
        params       =>  '{"conversation_id":"30C9DB6E-EA4D-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;


RESPONSE
--------------------------------------------------------------------------------
--
Seattle and San Francisco have different weather patterns despite both
experienc
ing a mild oceanic climate. San Francisco tends to be slightly warmer, with
aver
age temperatures ranging from 45?F to 67?F, and receives less rainfall,
```

```
around 2
0 inches per year, mostly during winter. In contrast, Seattle is cooler, with
te
mperatures ranging from 38?F to 64?F, and rainier, with around 37 inches of
rain
fall per year, distributed throughout the year. San Francisco is also known
for
its fog, particularly during summer, and receives more sunshine, around 160
sunn
y days per year, although it's often filtered through the fog. Overall, San
Fran
cisco's weather is warmer and sunnier, with more pronounced seasonal
variations,
 while Seattle's is cooler and rainier, with more consistent temperatures
throug
hout the year.


-- Call generate using the second conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
        prompt       =>  'Explain the difference again in one paragraph
only.',
        profile_name =>  'GENAI',
        action       =>  'CHAT',
        params       =>  '{"conversation_id":"30C9DB6E-EA4E-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;


RESPONSE
--------------------------------------------------------------------------------
--
The cost of living in New York is approximately 20-30% higher than in Los
Angele
s, mainly due to the higher cost of housing and transportation. New York has
a m
edian home price of around $999,000 and average rent of $3,000 per month for
a o
ne-bedroom apartment, compared to Los Angeles' median home price of $849,000
and
 average rent of $2,400 per month. While the cost of food and utilities is
relat
ively similar in both cities, the cost of transportation is higher in Los
Angele
s due to its car-centric culture, but the cost of public transportation is
highe
r in New York. Overall, the total monthly expenses for a single person in New
Yo
rk can range from $4,600, compared to around $4,050 in Los Angeles, making
New Y
ork the more expensive city to live in.
```

You may call the DBMS_CLOUD_AI.GENERATE function without specifying a conversation; however, in such cases, a meaningful response should not be expected.

```
-- Ask SELECT AI using the second conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
        prompt       =>  'Explain the difference again in one paragraph
only.',
        profile_name =>  'GENAI',
        action       =>  'CHAT') AS RESPONSE;


RESPONSE
--------------------------------------------------------------------------------
--
There is no previous explanation to draw from, as this is the beginning of
our c
onversation. If you would like to ask a question or provide a topic, I would
be
happy to explain the differences related to it in one paragraph.
```

Example: Verify Conversations through DBMS_CLOUD_AI Views
You can query the DBMS_CLOUD_AI conversation views to review conversation and prompt details. See #unique_77 for more details.

> **Note**
>
> The
> Views with the DBA_ prefix are available only to users with administrator privileges (ADMIN).

```
SELECT conversation_id, conversation_title, description FROM
dba_cloud_ai_conversations;

CONVERSATION_ID
----------------------------------
CONVERSATION_TITLE
--------------------------------------------------------------------------------
--------------------
DESCRIPTION
--------------------------------------------------------------------------------
--
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92
Seattle vs San Francisco Weather
The conversation discusses the comparison of weather patterns between Seattle
an
d San Francisco, focusing on the differences in temperature, rainfall, fog,
suns
hine, and seasonal variation between the two cities.

30C9DB6E-EA4E-AFBA-E063-9C6D46644B92
NY vs LA Cost Comparison
The conversation discusses and compares the cost of living in New York and
Los A
```

```
ngeles, covering housing, food, transportation, utilities, and taxes to
provide
an overall view of the expenses in both cities.

SELECT conversation_id, count(*) FROM dba_cloud_ai_conversation_prompts
     GROUP BY conversation_id;

CONVERSATION_ID                COUNT(*)
---------------------------------- ----------
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92          2
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92          2
```

Example: Update Conversation Details

You can update the `title`, `description`, and `retention_days` of a conversation using the `DBMS_CLOUD_AI.UPDATE_CONVERSATION` procedure. You can verify the update by querying the `DBMS_CLOUD_AI` conversation view.

```
-- Update the second conversation's title, description and retention_days
SQL> EXEC DBMS_CLOUD_AI.update_conversation(conversation_id => '30C9DB6E-EA4E-
AFBA-E063-9C6D46644B92',
                                            attributes =>
'{"retention_days":20,
                                                        "description":"This a
description",
                                                        "title":"a title",

"conversation_length":20}');

PL/SQL procedure successfully completed.


-- Verify the information for the second conversation
SQL> SELECT conversation_title, description, retention_days
FROM dba_cloud_ai_conversations
WHERE conversation_id = '30C9DB6E-EA4E-AFBA-E063-9C6D46644B92';

CONVERSATION_TITLE         DESCRIPTION
RETENTION_DAYS         LENGTH
------------------------ ------------------------------------
------------- -------------
a title                    This a description
20                         20
```

Example: Delete a Prompt

You can delete an individual prompt from your conversations and verify the modification by querying the `DBMS_CLOUD_AI` conversation view.

```
-- Find the latest prompt for first conversation
SELECT conversation_prompt_id FROM dba_cloud_ai_conversation_prompts
     WHERE conversation_id = '30C9DB6E-EA4D-AFBA-E063-9C6D46644B92'
     ORDER BY created DESC
     FETCH FIRST ROW ONLY;

CONVERSATION_PROMPT_ID
----------------------------------
```

```
30C9DB6E-EA61-AFBA-E063-9C6D46644B92


-- Delete the prompt
EXEC DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT('30C9DB6E-EA61-AFBA-
E063-9C6D46644B92');

PL/SQL procedure successfully completed.


-- Verify if the prompt is deleted
SELECT conversation_prompt_id FROM dba_cloud_ai_conversation_prompts
WHERE conversation_id = '30C9DB6E-EA4D-AFBA-E063-9C6D46644B92';

-- Only one prompt now
CONVERSATION_PROMPT_ID
----------------------------------
30C9DB6E-EA5A-AFBA-E063-9C6D46644B92
```

Example: Drop a Conversation
You can delete the entire conversation, which also removes all prompts associated with it.

```
-- Delete the first conversation
EXEC DBMS_CLOUD_AI.DROP_CONVERSATION('30C9DB6E-EA4D-AFBA-E063-9C6D46644B92');

PL/SQL procedure successfully completed.


-- Verify if the conversation and its prompts are removed
SELECT conversation_id FROM dba_cloud_ai_conversations;

-- We only have the second conversation now
CONVERSATION_ID
----------------------------------
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92


SELECT conversation_id, count(*) FROM dba_cloud_ai_conversation_prompts GROUP
BY conversation_id;

-- We only have prompts in the second conversation
CONVERSATION_ID                  COUNT(*)
---------------------------------- ----------
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92          2
```

Example: Set Up and Use Select AI with RAG

This example guides you through setting up credentials, configuring network access, and creating a vector index for integrating OCI Generative AI vector store cloud services with OpenAI using Oracle Autonomous AI Database.

The setup concludes with creating an AI profile that uses the vector index to enhance LLM responses. Finally, this example uses the Select AI `narrate` action, which returns a response that has been enhanced using information from the specified vector database.

The following example demonstrates building and querying vector index in Oracle AI Database 26ai.

```
--Grants EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--Grants EXECUTE privilege DBMS_CLOUD_PIPELINE to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Create the OpenAI credential
BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OPENAI_CRED',
        username => 'OPENAI_CRED',
        password => '<your_api_key>'
      );
END;
/

PL/SQL procedure successfully completed.

 -- Append the OpenAI endpoint
BEGIN
        DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
             host => 'api.openai.com',
             ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                     principal_name => 'ADB_USER',
                     principal_type => xs_acl.ptype_db)
       );
END;
/

PL/SQL procedure successfully completed.


-- Create the object store credential
BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OCI_CRED',
        username => '<your_username>',
        password => '<OCI_profile_password>'
      );
END;
/

PL/SQL procedure successfully completed.

 -- Create the profile with the vector index.

BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name =>'OPENAI_ORACLE',
          attributes   =>'{"provider": "openai",
            "credential_name": "OPENAI_CRED",
            "vector_index_name": "MY_INDEX",
            "temperature": 0.2,
```

```
            "max_tokens": 4096,
            "model": "gpt-3.5-turbo-1106"
          }');
END;
/

PL/SQL procedure successfully completed.

-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI_ORACLE');

PL/SQL procedure successfully
completed.

-- create a vector index with the vector store name, object store location and
-- object store credential
BEGIN
       DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
         index_name  => 'MY_INDEX',
         attributes  => '{"vector_db_provider": "oracle",
                          "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
                          "object_storage_credential_name": "OCI_CRED",
                          "profile_name": "OPENAI_ORACLE",
                          "vector_dimension": 1536,
                          "vector_distance_metric": "cosine",
                          "chunk_overlap":128,
                          "chunk_size":1024
      }');
END;
/
PL/SQL procedure successfully completed.


-- After the vector index is populated, we can now query the index.




-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI_ORACLE');

PL/SQL procedure successfully completed.

-- Select AI answers the question with the knowledge available in the vector
database.

set pages 1000
set linesize 150
SELECT AI narrate how can I deploy an oracle machine learning model;
RESPONSE
To deploy an Oracle Machine Learning model, you would first build your model
within the Oracle database. Once your in-database models are built, they
become immediately available for use, for instance, through a SQL query using
the prediction operators built into the SQL language.
```

```
The model scoring, like model building, occurs directly in the database,
eliminating the need for a separate engine or environment within which the
model and corresponding algorithm code operate. You can also use models from
a different schema (user account) if the appropriate permissions are in place.

Sources:
  - Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-
Database.txt (https://objectstorage.../v1/my_namespace/my_bucket/
my_data_folder/Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-
Database.txt)
  - Develop-and-deploy-machine-learning-models-using-Oracle-Autonomous-
Database-Machine-Learning-and-APEX.txt (https://objectstorage.../v1/
my_namespace/my_bucket/my_data_folder/Develop-and-deploy-machine-learning-
models-using-Oracle-Autonomous-Database-Machine-Learning-and-APEX.txt)
```

Example: Select AI with In-database Transformer Models

This example demonstrates how you can import a pretrained transformer model that is stored in Oracle object storage into your Oracle AI Database 26ai instance and then use the imported in-database model in Select AI profile to generate vector embeddings for document chunks and user prompts.

To use in-database transformer models in your Select AI profile, be sure you have:

- your pretrained model imported in your Oracle AI Database 26ai instance.
- optionally, access to Oracle object storage.

Import a Pretrained Transformer Model into your Oracle AI Database 26ai From Oracle Object Storage
Review the steps in Import Pretrained Models in ONNX Format for Vector Generation Within the Database and the blog Pre-built Embedding Generation model for Oracle AI Database 26ai to import a pretrained transformer model into your database.

The following example shows how to import a pretained transformer model from Oracle object storage into your database and then view the imported model.

```
- Create a Directory object, or use an existing directory object
CREATE OR REPLACE DIRECTORY ONNX_DIR AS 'onnx_model';

-- Object storage bucket
VAR location_uri VARCHAR2(4000);
EXEC :location_uri := 'https://adwc4pm.objectstorage.us-
ashburn-1.oci.customer-oci.com/p/
eLddQappgBJ7jNi6Guz9m9LOtYe2u8LWY19GfgU8flFK4N9YgP4kTlrE9Px3pE12/n/adwc4pm/b/
OML-Resources/o/';

-- Model file name
VAR file_name VARCHAR2(512);
EXEC :file_name := 'all_MiniLM_L12_v2.onnx';

-- Download ONNX model from object storage into the directory object
BEGIN
  DBMS_CLOUD.GET_OBJECT(
        credential_name => NULL,
        directory_name  => 'ONNX_DIR',
```

```
        object_uri      => :location_uri || :file_name);
END;
/

-- Load the ONNX model into the database
BEGIN
  DBMS_VECTOR.LOAD_ONNX_MODEL(
        directory  => 'ONNX_DIR',
        file_name  => :file_name,
        model_name => 'MY_ONNX_MODEL');
END;
/

-- Verify
SELECT model_name, algorithm, mining_function
FROM user_mining_models
WHERE  model_name='MY_ONNX_MODEL';
```

Use In-database Transformer Models in Select AI Profiles

These examples illustrate how to use in-database transformer models within a Select AI profile. One profile is configured only for generating vector embeddings, while the other supports both Select AI actions and vector index creation.

Review Perform Prerequisites for Select AI to complete the prerequisites.

The following is an example for generating vector embeddings only:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
     profile_name => 'EMBEDDING_PROFILE',
     attributes   => '{"provider" : "database",
                       "embedding_model": "MY_ONNX_MODEL"}'
  );
END;
/
```

The following is an example for general Select AI actions and vector index generation where you can specify a supported AI provider. This example uses OCI Gen AI profile and credentials. See Select your AI Provider and LLMs for list of supported providers. However, if you want to use in-database transformer model for generating vector embeddings, then use `"database: <MY_ONNX_MODEL>"` in `embedding_model` attribute:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'GENAI_CRED',
    user_ocid       => 'ocid1.user.oc1..aaaa...',
    tenancy_ocid    => 'ocid1.tenancy.oc1..aaaa...',
    private_key     => '<your_api_key>',
    fingerprint     => '<your_fingerprint>'
  );
END;
/

BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
```

```
    profile_name => 'OCI_GENAI',
    attributes   => '{"provider": "oci",
                      "model": "meta.llama-3.3-70b-instruct",
                      "credential_name": "GENAI_CRED",
                      "vector_index_name": "MY_INDEX",
                      "embedding_model": "database: MY_ONNX_MODEL"}'
  );
END;
/
```

Use Select AI with an In-database Transformer Model from Another Schema
This example demonstrates how to use Select AI with an in-database transformer model if another schema owner owns the model. Specify `schema_name.object_name` as the fully qualified name of the model in `embedding_model` attribute. If the current user is the schema owner or owns the model, you can omit the schema name.

Be sure to have the following privileges if a different schema owner owns the model:

- `CREATE ANY MINING MODEL` system privilege
- `SELECT ANY MINING MODEL` system privilege
- `SELECT MINING MODEL` object privilege on the specific model

To grant a system privilege, you must either have been granted the system privilege with the `ADMIN OPTION` or have been granted the `GRANT ANY PRIVILEGE` system privilege.

See System Privileges for Oracle Machine Learning for SQL to review the privileges.

The following statements allow `ADB_USER1` to score data and view model details in any schema as long as `SELECT` access has been granted to the data. However, `ADB_USER1` can only create models in the `ADB_USER1` schema.

```
GRANT CREATE MINING MODEL TO ADB_USER1;
GRANT SELECT ANY MINING MODEL TO ADB_USER1;


BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'OCI_GENAI',
    attributes   => '{"provider": "oci",
                      "credential_name": "GENAI_CRED",
                      "vector_index_name": "MY_INDEX",
                      "embedding_model": "database:
ADB_USER1.MY_ONNX_MODEL"}'
  );
END;
/
```

The following example shows how you can specify case sensitive model object name:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'OCI_GENAI',
    attributes   => '{"provider": "oci",
                      "credential_name": "GENAI_CRED",
                      "model": "meta.llama-3.3-70b-instruct",
```

```
                      "vector_index_name": "MY_INDEX",
                      "embedding_model": "database:
\"adb_user1\".\"my_model\""}'
  );
END;
/
```

End-to-end Examples with Different AI Providers

These examples demonstrate end-to-end steps for using in-database transformer model with Select AI RAG. One profile uses *database* as the `provider` exclusively created for generating embedding vectors while the other profile uses *oci* as the `provider` created for Select AI actions as well as vector index.

Review Perform Prerequisites for Select AI to provide the necessary privileges.

```
--Grant create any directory privilege to the user
GRANT CREATE ANY DIRECTORY to ADB_USER;

- Create a Directory object, or use an existing directory object
CREATE OR REPLACE DIRECTORY ONNX_DIR AS 'onnx_model';

-- Object storage bucket
VAR location_uri VARCHAR2(4000);
EXEC :location_uri := 'https://adwc4pm.objectstorage.us-
ashburn-1.oci.customer-oci.com/p/
eLddQappgBJ7jNi6Guz9m9LOtYe2u8LWY19GfgU8flFK4N9YgP4kTlrE9Px3pE12/n/adwc4pm/b/
OML-Resources/o/';

-- Model file name
VAR file_name VARCHAR2(512);
EXEC :file_name := 'all_MiniLM_L12_v2.onnx';

-- Download ONNX model from object storage into the directory object
BEGIN
  DBMS_CLOUD.GET_OBJECT(
        credential_name => NULL,
        directory_name  => 'ONNX_DIR',
        object_uri      => :location_uri || :file_name);
END;
/

-- Load the ONNX model into the database
BEGIN
  DBMS_VECTOR.LOAD_ONNX_MODEL(
        directory  => 'ONNX_DIR',
        file_name  => :file_name,
        model_name => 'MY_ONNX_MODEL');
END;
/

-- Verify
SELECT model_name, algorithm, mining_function
FROM user_mining_models
WHERE  model_name='MY_ONNX_MODEL';
```

```
--Administrator grants EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--Administrator grants EXECUTE privilege DBMS_CLOUD_PIPELINE to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Create the object store credential
BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OCI_CRED',
        username => '<your_username>',
        password => '<OCI_profile_password>'
      );
END;
/

PL/SQL procedure successfully completed.

 -- Create the profile with Oracle Database.

BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name =>'EMBEDDING_PROFILE',
          attributes   =>'{"provider": "database",
            "embedding_model": "MY_ONNX_MODEL"
          }');
END;
/

PL/SQL procedure successfully completed.

-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('EMBEDDING_PROFILE');

PL/SQL procedure successfully
completed.
```

This example uses *oci* as the `provider`.

```
--Grant create any directory privilege to the user
GRANT CREATE ANY DIRECTORY to ADB_USER;

- Create a Directory object, or use an existing directory object
CREATE OR REPLACE DIRECTORY ONNX_DIR AS 'onnx_model';

-- Object storage bucket
VAR location_uri VARCHAR2(4000);
EXEC :location_uri := 'https://adwc4pm.objectstorage.us-
ashburn-1.oci.customer-oci.com/p/
eLddQappgBJ7jNi6Guz9m9LOtYe2u8LWY19GfgU8flFK4N9YgP4kTlrE9Px3pE12/n/adwc4pm/b/
OML-Resources/o/';

-- Model file name
VAR file_name VARCHAR2(512);
```

```
EXEC :file_name := 'all_MiniLM_L12_v2.onnx';

-- Download ONNX model from object storage into the directory object
BEGIN
  DBMS_CLOUD.GET_OBJECT(
        credential_name => NULL,
        directory_name  => 'ONNX_DIR',
        object_uri      => :location_uri || :file_name);
END;
/

-- Load the ONNX model into the database
BEGIN
  DBMS_VECTOR.LOAD_ONNX_MODEL(
        directory  => 'ONNX_DIR',
        file_name  => :file_name,
        model_name => 'MY_ONNX_MODEL');
END;
/

-- Verify
SELECT model_name, algorithm, mining_function
FROM user_mining_models
WHERE  model_name='MY_ONNX_MODEL';


--Administrator Grants EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--Administrator Grants EXECUTE privilege DBMS_CLOUD_PIPELINE to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Create the object store credential
BEGIN
      DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OCI_CRED',
        username => '<your_username>',
        password => '<OCI_profile_password>'
      );
END;
/
--Create GenAI credentials
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'GENAI_CRED',
    user_ocid       => 'ocid1.user.oc1..aaaa...',
    tenancy_ocid    => 'ocid1.tenancy.oc1..aaaa...',
    private_key     => '<your_api_key>',
    fingerprint     => '<your_fingerprint>'
  );
END;
/
--Create OCI AI profile
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
     profile_name => 'OCI_GENAI',
```

```
     attributes   => '{"provider": "oci",
                      "model": "meta.llama-3.3-70b-instruct",
                      "credential_name": "GENAI_CRED",
                      "vector_index_name": "MY_INDEX",
                      "embedding_model": "database: MY_ONNX_MODEL"}'
  );
END;
/

-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_GENAI');

PL/SQL procedure successfully
completed.

-- create a vector index with the vector store name, object store location and
-- object store credential
BEGIN
       DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
         index_name  => 'MY_INDEX',
         attributes  => '{"vector_db_provider": "oracle",
                          "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
                          "object_storage_credential_name": "OCI_CRED",
                          "profile_name": "OCI_GENAI",
                          "vector_dimension": 384,
                          "vector_distance_metric": "cosine",
                          "chunk_overlap":128,
                          "chunk_size":1024
      }');
END;
/
PL/SQL procedure successfully completed.



-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_GENAI');

PL/SQL procedure successfully completed.

-- Select AI answers the question with the knowledge available in the vector
database.

set pages 1000
set linesize 150
SELECT AI narrate how can I deploy an oracle machine learning model;
RESPONSE
To deploy an Oracle Machine Learning model, you would first build your model
within the Oracle database. Once your in-database models are
built, they become immediately available for use, for instance, through a SQL
query using the prediction operators built into the SQL
language.

The model scoring, like model building, occurs directly in the database,
eliminating the need for a separate engine or environment within
```

```
which the model and corresponding algorithm code operate. You can also use
models from a different schema (user account) if the appropriate
permissions are in place.

Sources:
  - Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-
Database.txt (https://objectstorage.../v1/my_namespace/my_bucket/
my_data_folder/Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-
Database.txt)
  - Develop-and-deploy-machine-learning-models-using-Oracle-Autonomous-
Database-Machine-Learning-and-APEX.txt
(https://objectstorage.../v1/my_namespace/my_bucket/my_data_folder/Develop-
and-deploy-machine-learning-models-using-Oracle-Autonomous-
Database-Machine-Learning-and-APEX.txt)
```

Example: Improve SQL Query Generation

These examples demonstrate how comments, annotations, foreign key, and referential integrity constraints in database tables and columns can improve the generation of SQL queries from natural language prompts.

Example: Improve SQL Generation with Table and Column Comments
If you have table and column comments in your database tables, enable `"comments":"true"` parameter in `DBMS_CLOUD_AI.CREATE_PROFILE` function to retrieve table level and column level comments. The comments are added to the metadata of the LLM for a better SQL generation.

```
-- Adding comments to table 1, table 2, and table 3. Table 1 has 3 columns,
table 2 has 7 columns, table 3 has 2 columns.

-- TABLE1
COMMENT ON TABLE table1 IS 'Contains movies, movie titles and the year it was
released';
COMMENT ON COLUMN table1.c1 IS 'movie ids. Use this column to join to other
tables';
COMMENT ON COLUMN table1.c2 IS 'movie titles';
COMMENT ON COLUMN table1.c3 IS 'year the movie was released';
-- TABLE2
COMMENT ON TABLE table2 IS 'transactions for movie views - also known as
streams';
COMMENT ON COLUMN table2.c1 IS 'day the movie was streamed';
COMMENT ON COLUMN table2.c2 IS 'genre ids. Use this column to join to other
tables';
COMMENT ON COLUMN table2.c3 IS 'movie ids. Use this column to join to other
tables';
COMMENT ON COLUMN table2.c4 IS 'customer ids. Use this column to join to
other tables';
COMMENT ON COLUMN table2.c5 IS 'device used to stream, watch or view the
movie';
COMMENT ON COLUMN table2.c6 IS 'sales from the movie';
COMMENT ON COLUMN table2.c7 IS 'number of views, watched, streamed';

-- TABLE3
COMMENT ON TABLE table3 IS 'Contains the genres';
COMMENT ON COLUMN table3.c1 IS 'genre id. use this column to join to other
tables';
```

```
COMMENT ON COLUMN table3.c2 IS 'name of the genre';


BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'myprofile',
    attributes =>
        '{"provider": "azure",
          "azure_resource_name": "my_resource",
          "azure_deployment_name": "my_deployment",
          "credential_name": "my_credential",
          "comments":"true",
          "object_list": [
            {"owner": "moviestream", "name": "table1"},
            {"owner": "moviestream", "name": "table2"},
            {"owner": " moviestream", "name": "table3"}
          ]
          }'
    );

    DBMS_CLOUD_AI.SET_PROFILE(
        profile_name => 'myprofile'
    );

END;
/

--Prompts
select ai what are our total views;
RESPONSE
--------------------------------------------------
TOTAL_VIEWS
-----------
   97890562

select ai showsql what are our total views;

RESPONSE
--------------------------------------------------------------------------------
SELECT SUM(QUANTITY_SOLD) AS total_views
FROM "moviestream"."table"

select ai what are our total views broken out by device;

DEVICE                    TOTAL_VIEWS
------------------------- -----------
mac                          14719238
iphone                       20793516
ipad                         15890590
pc                           14715169
galaxy                       10587343
pixel                        10593551
lenovo                        5294239
fire                          5296916

8 rows selected.
```

```
select ai showsql what are our total views broken out by device;

RESPONSE

--------------------------------------------------------------------------------
---------
SELECT DEVICE, COUNT(*) AS TOTAL_VIEWS
FROM "moviestream"."table"
GROUP BY DEVICE
```

Example: Improve SQL Query Generation with Table and Column Annotations
This example demonstrates the integration of annotations in Select AI, applicable in Oracle AI Database 26ai. The annotations are added to the metadata that is sent to the LLM.

If you have a table with annotations in your schema, enable `"annotations":"true"` in the `DBMS_CLOUD_AI.CREATE_PROFILE` function to instruct Select AI to add annotations to the metadata.

```
--
-- Annotations
--

CREATE TABLE emp2 (
    empno NUMBER,
    ename VARCHAR2(50) ANNOTATIONS (display 'lastname'),
    salary NUMBER ANNOTATIONS ("person_salary", "column_hidden"),
    deptno NUMBER ANNOTATIONS (display 'department')
)ANNOTATIONS (requires_audit 'yes', version '1.0', ówner 'HR Organization');

Table created.

 BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
   profile_name => 'GOOGLE_ANNOTATIONS',
   attributes   => '{"provider": "google",
      "credential_name": "GOOGLE_CRED",
      "object_list": [{"owner": "ADB_USER", "name": "emp2"}],
      "annotations" : "true"
      }');
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI.SET_PROFILE('GOOGLE_ANNOTATIONS');

PL/SQL procedure successfully completed.
```

Example: Improve SQL Query Generation with Foreign Key and Referential Key Constraints
This example demonstrates the ability of the LLM to generate accurate `JOIN` conditions by retrieving the foreign key and referential key constraints into the metadata of the LLM. The foreign key and referential key constraints provide structured relationship data between the tables to the LLM.

Enable `"constraints":"true"` in the `DBMS_CLOUD_AI.CREATE_PROFILE` function for Select AI to retrieve foreign key and referential key.

```
--
-- Referential Constraints
--
CREATE TABLE dept_test (
    deptno NUMBER PRIMARY KEY,
    dname VARCHAR2(50)
);

Table created.

CREATE TABLE emp3 (
    empno NUMBER PRIMARY KEY,
    ename VARCHAR2(50),
    salary NUMBER,
    deptno NUMBER,
    CONSTRAINT emp_dept_fk FOREIGN KEY (deptno) REFERENCES dept_test(deptno)
);

Table created.


BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name=>'GOOGLE_CONSTRAINTS',
    attribues   =>'{"provider": "google",
      "credential_name": "GOOGLE_CRED",
      "object_list": [{"owner": "ADB_USER", "name": "dept_test"},
                      {"owner": "ADB_USER", "name": "emp3"}],
      "constraints" : "true"
      }');
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI.SET_PROFILE('GOOGLE_CONSTRAINTS');

PL/SQL procedure successfully completed.
```

Example: Automatically Detect Relevant Table Metadata
These examples shows how Select AI automatically detects relevant tables and sends metadata only for those specific tables relevant to the query in Oracle AI Database 26ai. To enable this feature, set `object_list_mode` to *automated*. This automatically creates a vector index named `<profile_name>_OBJECT_LIST_VECINDEX`. The vector index is initialized with default attributes and values such as `refresh_rate`, `similarity_threshold`, and `match_limit`. You can modify some of the attributes through `DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX`. See #unique_152 for more information.

One profile is configured to use `object_list` to specify the schema or the objects in the schema while the other does not specify `object_list`. However, the same SQL construct is expected.

Review Perform Prerequisites for Select AI to provide access to the DBMS_CLOUD_AI package and provide network access to the AI provider.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name=>'OCI_AUTO',
    attributes=>'{"provider":
"oci",
      "credential_name": "GENAI_CRED",
      "object_list": [{"owner": "SH"}],
      "oci_compartment_id": "ocid1.compartment.oc1..aaaa...",
      "model" : "meta.llama-3.3-70b-instruct"
      }');
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_AUTO');

PL/SQL procedure successfully completed.

select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------
SELECT COUNT(DISTINCT c."CUST_ID") AS "NUMBER_OF_CUSTOMERS"
FROM "SH"."CUSTOMERS" c
WHERE UPPER(c."CUST_CITY") = UPPER('San Francisco')
AND UPPER(c."CUST_MARITAL_STATUS") = UPPER('married')
```

The following example compares the same scenario without using object_list. When you don't specify object_list, Select AI automatically chooses all objects available to the current schema.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name=>'OCI_AUTO1',
    attributes=>'{"provider":
"oci",
      "credential_name": "GENAI_CRED",
      "oci_compartment_id": "ocid1.compartment.oc1..aaaa...",
      "object_list_mode": "automated",
      "model" : "meta.llama-3.3-70b-instruct"
      }');
END;
/
PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_AUTO1');

PL/SQL procedure successfully completed.
```

```
select ai showsql how many customers in San Francisco are married?;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------
SELECT COUNT(c."CUST_ID") AS "Number_of_Customers"
FROM "SH"."CUSTOMERS" c
WHERE UPPER(c."CUST_CITY") = UPPER('San Francisco')
AND UPPER(c."CUST_MARITAL_STATUS") = UPPER('Married')
```

Example: Use Select AI with Database Links to Query Another Autonomous AI Database

This example shows how to set up a Database Link from Autonomous AI Database to the source database and use Select AI to generate SQL from natural language prompts. Select AI uses the metadata from the source database to generate SQL.

Before You Begin
Review

- Perform Prerequisites for Select AI
- Download your cloud wallet credentials and upload to an Object Storage bucket:
  - Download the wallet (`cwallet.sso`) from your source database through OCI Console or Cloud Shell. See Download Database Connection Information for more details.
  - Upload the wallet file to an Object Storage bucket. See Creating an Object Storage Bucket for more details.

This example shows how to set up a Database Link (DB Link) in an Autonomous AI Database to securely connect with another Autonomous AI Database. However, you can create DB Links to non-Autonomous AI Databases and third-party databases. Database links enable Select AI to query across remote data sets without replicating data through a wallet, credentials, and linked views.

You first create a credential to store your username and password to authenticate the source database. Create a directory to store the wallet files used for authentication when you are connecting to another Autonomous AI Database. Download the source database wallet credentials using `GET_OBJECT` procedure. Create a secure Database Link from Autonomous AI Database to the source Autonomous Database. You then create views on the remote tables. Create an AI profile with `object_list` attribute specifying the views as JSON objects. Finally, issue any NL2SQL Select AI actions such as `runsql`, `showsql`, `explainsql`, `narrate`, or `chat`. This example uses `showsql`.

```
--Create Cloud Credential (run in Autonomous AI Database)

BEGIN
DBMS_CLOUD.DROP_CREDENTIAL(credential_name => 'DB_LINK_CRED');
EXCEPTION WHEN OTHERS THEN NULL;
END;
/
BEGIN
DBMS_CLOUD.CREATE_CREDENTIAL(
credential_name => 'DB_LINK_CRED',
```

```
username => 'DB_USER',     -- Username on source database
password => '<password>'          -- Password for source database
);
END;
/

--Create Directory (run in Autonomous AI Database)

CREATE DIRECTORY dblink_wallet_dir AS 'DATA_PUMP_DIR';

--Prepare and Upload Source Database Wallet in Object Storage bucket and run
in Autonomous AI Database:
BEGIN
DBMS_CLOUD.GET_OBJECT(
credential_name => 'DB_LINK_CRED',
object_uri => 'https://objectstorage.ca-toronto-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/data_folder/cwallet.sso/cwallet.sso',
directory_name => 'DBLINK_WALLET_DIR'
);
END;
/

--Create Database Link (Drop dblink if it exists) to Source Database (run in
Autonomous AI Database)


BEGIN
DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(db_link_name => 'MY_DATA_LINK');
EXCEPTION WHEN OTHERS THEN NULL;
END;
/

BEGIN
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
db_link_name => 'MY_DATA_LINK',
hostname => 'adb.<region>-1.oraclecloud.com',             -- Source database
hostname
port => '1522',                                           -- Source database
port
service_name => 'your_service_name.adb.oraclecloud.com',  -- Source database
service
credential_name => 'DB_LINK_CRED',
directory_name => 'DBLINK_WALLET_DIR'
);
END;
/

--Create Views (run in Autonomous AI Database)

CREATE VIEW customer_view AS SELECT * FROM customer@MY_DATA_LINK;
CREATE VIEW streams_view AS SELECT * FROM streams@MY_DATA_LINK;

--Create an AI Profile (run in Autonomous AI Database)

BEGIN
DBMS_CLOUD_AI.CREATE_PROFILE(
```

```
profile_name => 'MY_AI_PROFILE',
attributes => JSON_OBJECT(
'provider' => 'openai',
'credential_name' => 'OPENAI_CRED',
'object_list' => JSON_ARRAY(
JSON_OBJECT('owner' => 'SELECT_AI_USER', 'name' => 'CUSTOMER_VIEW'),
JSON_OBJECT('owner' => 'SELECT_AI_USER', 'name' => 'STREAMS_VIEW')
)
)
);
END;
/

--Showsql test:

SELECT AI SHOWSQL how many customers are there;

--Run on Source Database
```

Copy the generated SQL, remove `@MY_DATA_LINK` and run the query on your source database to verify.

Example: Generate Synthetic Data

This example explores how you can generate synthetic data mimicking the characteristics and distribution of real data.

The following example shows how to create a few tables in your schema, use OCI Generative AI as your AI provider to create an AI profile, synthesize data into those tables using the `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` function, and query or generate responses to natural language prompts with Select AI.

```
--Create tables or use cloned tables

CREATE TABLE ADB_USER.Director (
    director_id     INT PRIMARY KEY,
    name            VARCHAR(100)
);
CREATE TABLE ADB_USER.Movie (
    movie_id        INT PRIMARY KEY,
    title           VARCHAR(100),
    release_date    DATE,
    genre           VARCHAR(50),
    director_id     INT,
    FOREIGN KEY (director_id) REFERENCES ADB_USER.Director(director_id)
);
CREATE TABLE ADB_USER.Actor (
    actor_id        INT PRIMARY KEY,
    name            VARCHAR(100)
);
CREATE TABLE ADB_USER.Movie_Actor (
    movie_id        INT,
    actor_id        INT,
    PRIMARY KEY (movie_id, actor_id),
    FOREIGN KEY (movie_id) REFERENCES ADB_USER.Movie(movie_id),
```

```
    FOREIGN KEY (actor_id) REFERENCES ADB_USER.Actor(actor_id)
);

-- Create the GenAI credential
BEGIN
  DBMS_CLOUD.create_credential(
    credential_name => 'GENAI_CRED',
    user_ocid       => 'ocid1.user.oc1....',
    tenancy_ocid    => 'ocid1.tenancy.oc1....',
    private_key     => 'vZ6cO...',
    fingerprint     => '86:7d:...'
  );
END;
/

-- Create a profile
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name
=>'GENAI',
      attributes  =>'{"provider":
"oci",
        "credential_name": "GENAI_CRED",
        "object_list": [{"owner": "ADB_USER",
        "oci_compartment_id": "ocid1.compartment.oc1...."}]
       }');
END;
/


EXEC DBMS_CLOUD_AI.set_profile('GENAI');

-- Run the API for single table
BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name => 'GENAI',
        object_name  => 'Director',
        owner_name   => 'ADB_USER',
        record_count => 5
    );
END;
/
PL/SQL procedure successfully completed.


-- Query the table to see results
SQL> SELECT * FROM ADB_USER.Director;

DIRECTOR_ID NAME
-----------
--------------------------------------------------------------------------------
----------------------
          1 John Smith
          2 Emily Chen
          3 Michael Brown
          4 Sarah Taylor
```

```
         5 David Lee


-- Or ask select ai to show the results
SQL> select ai how many directors are there;

NUMBER_OF_DIRECTORS
-------------------
                  5
```

Example: Generate Synthetic Data for Multiple Tables

After you create and set your AI provider profile, use the `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` to generate data for multiple tables. You can query or use Select AI to respond to the natural language prompts.

```
BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name => 'GENAI',
        object_list => '[{"owner": "ADB_USER", "name":
"Director","record_count":5},
                        {"owner": "ADB_USER", "name":
"Movie_Actor","record_count":5},
                        {"owner": "ADB_USER", "name":
"Actor","record_count":10},
                        {"owner": "ADB_USER", "name":
"Movie","record_count":5,"user_prompt":"all movies released in 2009"}]'
    );
END;
/
PL/SQL procedure successfully completed.


-- Query the table to see results
SQL> select * from ADB_USER.Movie;

 MOVIE_ID
TITLE                                                    RELEASE_D
                 GENRE                                  DIRECTOR_ID
---------- ---------------------------------------------------------- ---------
------------------------------------------------------------------
-----------
         1 The Dark Knight                                        15-
JUL-09                           Action                            8
         2 Inglourious Basterds                                   21-
AUG-09                           War                               3
         3 Up in the Air                                          04-
SEP-09                           Drama                             6
         4 The Hangover                                           05-
JUN-09                           Comedy                            1
         5 District 9                                             14-
AUG-09                           Science Fiction                  10



-- Or ask select ai to show the results
```

```
SQL> select ai how many actors are there;

Number of Actors
----------------
              10
```

Example: Interrupting Synthetic Data Generation Task
When you start generating large synthetic data sets, the system splits the task into smaller subtasks and runs them in parallel. If you interrupt the session (for example, using `Ctrl + C`), the process may continue in the background because the subtasks do not end automatically.

If you want to end a running task, take the following steps:

```
-- Find the operation ID of the running Synthetic Data Generation (SDG)
process
SELECT * FROM user_load_operations WHERE operation_type = 'SYNTHETIC_DATA';

-- Delete a specific SDG operation
EXEC dbms_cloud.delete_operation(<operation_id>);

-- Delete all SDG operations
EXEC dbms_cloud.delete_all_operations('SYNTHETIC_DATA');
```

If the commands above do not stop the background processes, end the session manually:

```
SELECT sid, serial# FROM v$session WHERE audsid = userenv('sessionid');
ALTER SYSTEM KILL SESSION '<sid>,<serial#>' IMMEDIATE;
```

Example: Guide Synthetic Data Generation with Sample Rows
To guide AI service in generating synthetic data, you can randomly select existing records from a table. For instance, by adding `{"sample_rows": 5}` to the `params` argument, you can send 5 sample rows from a table to the AI provider. This example generates 10 additional rows based on the sample rows from the `Transactions` table.

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'GENAI',
    object_name  => 'Transactions',
    owner_name   => 'ADB_USER',
    record_count => 10,
    params       => '{"sample_rows":5}'
  );
END;
/
```

Example: Customize Synthetic Data Generation with User Prompts
The `user_prompt` argument enables you to specify additional rules or requirements for data generation. This can be applied to a single table or as part of the `object_list` argument for multiple tables. For example, in the following calls to `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA`, the prompt instructs the AI to generate synthetic data on movies released in 2009.

```
-- Definition for the Movie table CREATE TABLE Movie
```

```
CREATE TABLE Movie (
    movie_id        INT PRIMARY KEY,
    title           VARCHAR(100),
    release_date    DATE,
    genre           VARCHAR(50),
    director_id     INT,
    FOREIGN KEY (director_id) REFERENCES Director(director_id)
);



BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name      => 'GENAI',
    object_name       => 'Movie',
    owner_name        => 'ADB_USER',
    record_count      => 10,
    user_prompt       => 'all movies are released in 2009',
    params            => '{"sample_rows":5}'
  );
END;
/

BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name => 'GENAI',
        object_list => '[{"owner": "ADB_USER", "name":
"Director","record_count":5},
                        {"owner": "ADB_USER", "name":
"Movie_Actor","record_count":5},
                        {"owner": "ADB_USER", "name":
"Actor","record_count":10},
                        {"owner": "ADB_USER", "name":
"Movie","record_count":5,"user_prompt":"all movies are released in 2009"}]'
    );
END;
/
```

Example: Improve Synthetic Data Quality by Using Table Statistics

If a table has column statistics or is cloned from a database that includes metadata, Select AI can use these statistics to generate data that closely resembles or is consistent with the original data.

For `NUMBER` columns, the high and low values from the statistics guide the value range. For instance, if the `SALARY` column in the original `EMPLOYEES` table ranges from 1000 to 10000, the synthetic data for this column will also fall within this range.

For columns with distinct values, such as a `STATE` column with values *CA*, *WA*, and *TX*, the synthetic data will use these specific values. You can manage this feature using the `{"table_statistics": true/false}` parameter. By default, the table statistics are enabled.

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name      => 'GENAI',
    object_name       => 'Movie',
    owner_name        => 'ADB_USER',
```

```
    record_count      => 10,
    user_prompt => 'all movies released in 2009',
    params            => '{"sample_rows":5,"table_statistics":true}'
  );
END;
/
```

Example: Use Column Comments to Guide Data Generation

If column comments exist, Select AI automatically includes them to provide additional information for the LLM during data generation. For example, a comment on the `Status` column in a Transaction table might list allowed values such as *successful*, *failed*, *pending*, *canceled*, and *need manual check*. You can also add comments to further explain the column, giving AI services more precise instructions or hints for generating accurate data. By default, comments are disabled. See Optional Parameters for more details.

```
-- Use comment on column
COMMENT ON COLUMN Transaction.status IS 'the value for state should either be
''successful'', ''failed'', ''pending'' or ''canceled''';
/

BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name  => 'GENAI',
        object_name   => 'employees',
        owner_name    => 'ADB_USER',
        record_count  => 10
        params        => '{"comments":true}'

    );
END;
/
```

Example: Set Unique Values in Synthetic Data Generation

When generating large amounts of synthetic data with LLMs, duplicate values are likely to occur. To prevent this, set up a unique constraint on the relevant column. This ensures that Select AI ignores rows with duplicate values in the LLM response. Additionally, to restrict values for certain columns, you can use the `user_prompt` or add comments to specify the allowed values, such as limiting a `STATE` column to *CA*, *WA*, and *TX*.

```
-- Use 'user_prompt'
BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name  => 'GENAI',
        object_name   => 'employees',
        owner_name    => 'ADB_USER',
        user_prompt   => 'the value for state should either be CA, WA, or TX',
        record_count  => 10
    );
END;
/


-- Use comment on column
COMMENT ON COLUMN EMPLOYEES.state IS 'the value for state should either be
```

```
CA, WA, or TX'
/
```

Example: Enhance Synthetic Data Generation by Parallel Processing
To reduce runtime, Select AI splits synthetic data generation tasks into smaller chunks for tables without primary keys or with numeric primary keys. These tasks run in parallel, interacting with the AI provider to generate data more efficiently. The Degree of Parallelism (DOP) in your database, influenced by your Autonomous AI Database service level and ECPU or OCPU settings, determines the number of records each chunk processes. Running tasks in parallel generally improves performance, especially when generating large amounts of data across many tables. To manage the parallel processing of synthetic data generation, set `priority` as an optional parameter. See Optional Parameters.

Example: Enable or Disable Data Access

This example illustrates how administrators can control data access and prevent Select AI from sending actual schema tables to the LLM.

Disabling Data Access
To restrict access to schema tables, log in as an administrator and run the following procedure.

```
EXEC DBMS_CLOUD_AI.DISABLE_DATA_ACCESS;

PL/SQL procedure successfully completed.
```

Disabling data access limits Select AI's `narrate` action and Synthetic Data Generation. The `narrate` action and synthetic data generation raise an error.

Log in as database user, create and configure your AI profile. Review Perform Prerequisites for Select AI to configure your AI profile.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name =>'DATA_ACCESS',
          attributes   =>'{"provider": "openai",
            "credential_name": "OPENAI_CRED",
            "object_list": [{"owner":"SH"}]
          }');
END;
/

EXEC DBMS_CLOUD_AI.SET_PROFILE('DATA_ACCESS');


select ai how many customers;

NUM_CUSTOMERS
55500

select ai narrate what are the top 3 customers in San Francisco;

ORA-20000: Data access is disabled for SELECT AI.
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD", line 2228
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD_AI", line 13157
ORA-06512: at line 1 https://docs.oracle.com/error-help/db/ora-20000/
```

```
The stored procedure 'raise_application_error' was called which causes this
error to be generated
Error at Line: 1 Column: 6
```

The following example shows the errors that are triggered when you try to generate synthetic data.

```
BEGIN
DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
profile_name => 'DATA_ACCESS_SDG',
object_name => 'CUSTOMERS_NEW',
owner_name => 'ADB_USER,
record_count => 5
);
END;
/

ERROR at line 1:

ORA-20000: Data access is disabled for SELECT AI.
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD", line 2228
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD_AI", line 13401

ORA-06512: at line 2
```

Enabling Data Access

The following example shows enabling data access. Log in as an administrator and run the following procedure:

```
EXEC DBMS_CLOUD_AI.ENABLE_DATA_ACCESS;

PL/SQL procedure successfully completed.
```

Log in as database user, create and configure your AI profile. Review Perform Prerequisites for Select AI to configure your AI profile. Run `narrate` action and separately generate synthetic data.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name =>'DATA_ACCESS_NEW',
          attributes   =>'{"provider": "openai",
            "credential_name": "OPENAI_CRED",
            "object_list": [{"owner":"SH"}]
          }');
   END;
   /

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI.SET_PROFILE('DATA_ACCESS_NEW');

PL/SQL procedure successfully completed.
```

```
select ai how many customers;

NUM_CUSTOMERS
55500

select ai narrate what are the top 3 customers in San Francisco;

"RESPONSE"
"The top 3 customers in San Francisco are Cody Seto, Lauren Yaskovich, and
Ian Mc"
```

The following example shows successful synthetic data generation after enabling data access.

```
BEGIN
DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
profile_name => 'DATA_ACCESS_SDG',
object_name => 'CUSTOMERS_NEW',
owner_name => 'ADB_USER',
record_count => 5
);
END;
/

PL/SQL procedure successfully completed.
```

Example: Select AI Feedback

These examples demonstrate how you can use the `DBMS_CLOUD_AI.FEEDBACK` procedure and the different scenarios of involving the `feedback` action to provide feedback and improve subsequent SQL query generation.

Before You Begin
Review Perform Prerequisites for Select AI.

Example: Provide Negative Feedback
The following example demonstrates providing corrections to the generated SQL as feedback (negative feedback) using `feedback_type` as *negative* and providing your SQL query.

You add your feedback to the AI profile named `OCI_FEEDBACK1` by calling the `DBMS_CLOUD_AI.FEEDBACK` procedure with the `sql_text` parameter containing the prompt. See #unique_128 to learn about the attributes. Then, you retrieve the `content` and `attributes` columns from the `<profile_name>_FEEDBACK_VECINDEX$VECTAB` table, which is linked to that specific SQL query. Select AI automatically creates this vector table when you first use the feedback feature. See #unique_127 for more information.

```
SQL> select ai showsql how many movies;

RESPONSE
--------------------------------------------------------------------------------
SELECT COUNT(m."MOVIE_ID") AS "Number of Movies" FROM "ADB_USER"."MOVIES" m

SQL> exec DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1', sql_text=>
'select ai showsql how many movies', feedback_type=> 'negative',
response=>'SELECT SUM(1) FROM "ADB_USER"."MOVIES"');
```

```
PL/SQL procedure successfully completed.
SQL> select CONTENT, ATTRIBUTES from OCI_FEEDBACK1_FEEDBACK_VECINDEX$VECTAB
where JSON_VALUE(attributes, '$.sql_text') = 'select ai showsql how many
movies';

CONTENT
--------------------------------------------------------------------------------
----------------------
how many movies
ATTRIBUTES
--------------------------------------------------------------------------------
--------------------

{"response":"SELECT SUM(1) FROM
\"ADB_USER\".\"MOVIES\"","feedback_type":"negative","sql_id":null,"sql_text":"
select ai showsql how many movies","feedback_content":null}
```

Example: Provide Positive Feedback

The following example demonstrates providing your approval that you agree and confirm the generated SQL (positive feedback) using `feedback_type` as *positive*.

In this example, the query retrieves the `sql_id` from the `v$mapped_sql` view for the given prompt. See V_MAPPED_SQL for more information.

You add your feedback to the AI profile named `OCI_FEEDBACK1` by calling the `DBMS_CLOUD_AI.FEEDBACK` procedure with the `sql_id` parameter. Then, you retrieve the `content` and `attributes` columns from the `<profile_name>_FEEDBACK_VECINDEX$VECTAB` table, which is linked to that specific SQL query. Select AI automatically creates this vector table when you first use the feedback feature. See #unique_127 for more information.

```
SQL> select ai showsql how many distinct movie genres?;

RESPONSE
--------------------------------------------------------------------------------
-----------
SELECT COUNT(DISTINCT g."GENRE_NAME") AS "Number of Movie Genres" FROM
"ADB_USER"."GENRES" g

SQL> SELECT sql_id FROM v$mapped_sql WHERE sql_text = 'select ai showsql how
many distinct movie genres?';

SQL_ID
-------------
852w8u83gktc1

SQL> exec DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1', sql_id=>
'852w8u83gktc1', feedback_type=>'positive', operation=>'add');

PL/SQL procedure successfully completed.

SQL> SELECT content, attributes FROM OCI_FEEDBACK1_FEEDBACK_VECINDEX$VECTAB
WHERE JSON_VALUE(attributes, '$.sql_id') ='852w8u83gktc1';

 CONTENT
--------------------------------------------------------------------------------
----------------------
```

```
how many distinct movie genres?
ATTRIBUTES
--------------------------------------------------------------------------------
--------------------
{"response":"SELECT COUNT(DISTINCT g.\"GENRE_NAME\") AS \"Number of Movie
Genres\" FROM \"ADB_USER\".\"GENRES\"
g","feedback_type":"positive","sql_id":"852w8u83gktc1","sql_text":"select ai
showsql how many distinct movie genres?","feedback_content":null}
```

Example: Add or Delete Your Feedback for the Generated SQL
The following example demonstrates adding or deleting your feedback for the generated SQL by specifying the `DBMS_CLOUD_AI.FEEDBACK` procedure parameters. This example demonstrates using `sql_id` and `sql_text` along with other parameters.

> **Note**
>
> Select AI allows only a single feedback entry for each `sql_id`. If you provide additional feedback for the same `sql_id`, Select AI replaces the previous entry with the new one.

See #unique_128 for more details on the parameters.

```
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',
                                    sql_id=> '852w8u83gktc1',
                                    feedback_type=>'positive',
                                    operation=>'add');
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',
                                    sql_text=> 'select ai showsql how many
movies',
                                    feedback_type=> 'negative',
                                    response=>'SELECT SUM(1) FROM
"ADB_USER"."MOVIES"',
                                    feedback_content=>'Use SUM instead of
COUNT');
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',
                                    sql_id=> '852w8u83gktc1',
                                    operation=>'delete');
```

Example: Use Feedback Action with the Last AI SQL to Provide Negative Feedback
This example demonstrates using `feedback` action to improve the generated SQL by suggesting the modifications using natural language.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name=>'OCI_FEEDBACK1',
    attributes=>'{"provider":
"oci",
      "credential_name": "GENAI_CRED",
      "oci_compartment_id": "ocid1.compartment.oc1..aaaa...",
      "object_list": [{"owner": "ADB_USER", "name": "users"},
                      {"owner": "ADB_USER", "name": "movies"},
                      {"owner": "ADB_USER", "name": "genres"},
                      {"owner": "ADB_USER", "name": "watch_history"},
                      {"owner": "ADB_USER", "name": "movie_genres"},
```

```
                        {"owner": "ADB_USER", "name": "employees1"},
                        {"owner": "ADB_USER", "name": "employees2"}
                          ]
      }');
END;
/

EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_FEEDBACK1');


PL/SQL procedure successfully completed.


select ai showsql rank movie duration;

RESPONSE
--------------------------------------------------------------------------------
-
SELECT "DURATION" AS "Movie Duration" FROM "ADB_USER"."MOVIES" ORDER BY
"DURATION"

select ai feedback use ascending sorting;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------
Based on your feedback, the SQL query for prompt "rank movie duration" is
successfully refined. The refined SQL query as following:
SELECT m."DURATION" AS "Movie Duration" FROM "ADB_USER."MOVIES" m ORDER BY
m."DURATION" ASC


select ai showsql rank the movie duration;

RESPONSE
--------------------------------------------------------------------------------
-----------
SELECT m."DURATION" AS "Movie Duration" FROM "ADB_USER."MOVIES" m ORDER BY
m."DURATION" ASC
```

Example: Use Feedback Action with the Last AI SQL to Provide Positive Feedback
This example demonstrates using the `feedback` action to accept the generated SQL using natural language.

```
--Positive feedback

select ai showsql which movies are comedy?;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------
SELECT DISTINCT m."TITLE" AS "Movie Title" FROM "ADB_USER"."MOVIES" m INNER
JOIN "ADB_USER"."MOVIE_GENRES" mg ON m."MOVIE_ID" = mg."MOVIE_ID" INNER JOIN
```

```
"ADB_USER"."GENRES" g ON mg."GENRE_ID" = g."GENRE_ID" WHERE g."GENRE_NAME" =
'comedy'

select ai feedback this is correct;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------
Thank you for your positive feedback. The SQL query for prompt "which movies
are comedy?" is correctly implemented and delivering the expected results. It
will be referenced for future optimizations and improvements.


Select AI Feedback Action Referring SQL_ID
```

Example: Use Feedback Action with SQL_ID to Provide Feedback
This example demonstrates using `SQL_ID` with the `feedback` action to provide feedback for a particular generated SQL query. You can obtain the `SQL_ID` by querying the `v$MAPPED_SQL` table.

```
-- Query mentioned with SQL_ID

select ai showsql how many movies are in each genre;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
SELECT g."GENRE_NAME" AS "Genre Name", COUNT(m."MOVIE_ID") AS "Number of
Movies" FROM "ADB_USER"."MOVIES" m INNER JOIN "ADB_USER"."MOVIE_GENRES" mg ON
m."MOVIE_ID" = mg."MOVIE_ID" INNER JOIN "ADB_USER"."GENRES" g ON
mg."GENRE_ID" = g."GENRE_ID" GROUP BY g."GENRE_NAME"

select sql_id from v$cloud_ai_sql where sql_text = 'select ai showsql how
many movies are in each genre';

SQL_ID
-------------
8azkwc0hr87ga

select ai feedback for query with sql_id = '8azkwc0hr87ga', rank in
descending sorting;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------------------------------
Based on your feedback, the SQL query for prompt "how many movies are in each
genre" is successfully refined. The refined SQL query as following:
SELECT g."GENRE_NAME" AS "Genre Name", COUNT(m."MOVIE_ID") AS "Number of
Movies"
FROM "ADB_USER"."MOVIES" m
```

```
INNER JOIN "ADB_USER"."MOVIE_GENRES" mg ON m."MOVIE_ID" = mg."MOVIE_ID"
INNER JOIN "ADB_USER"."GENRES" g ON mg."GENRE_ID" = g."GENRE_ID"
GROUP BY g."GENRE_NAME"
ORDER BY COUNT(m."MOVIE_ID") DESC
```

Example:Use Feedback Action with Query Text
This example shows the `feedback` action for a specific Select AI query by including the Select AI prompt in quotes followed by your feedback.

```
-Query mentioned with SQL_TEXT

select ai showsql how many watch history in total;

RESPONSE
--------------------------------------------------------------------------------
----
SELECT COUNT(w."WATCH_ID") AS "Total Watch History" FROM
"ADB_USER"."WATCH_HISTORY" w

select ai feedback for query "select ai showsql how many watch history in
total", name the column as total_watch;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------
Based on your feedback, the SQL query for prompt "how many watch history in
total" is successfully refined. The refined SQL query as following:
SELECT COUNT(w."WATCH_ID") AS "total_watch" FROM "ADB_USER"."WATCH_HISTORY" w
```

Example: Select AI Summarize

These examples show how to use the `summarize` action and `DBMS_CLOUD_AI.SUMMARIZE` function. Also, customize the summary generation for your content using the function.

Before You Begin
Review Perform Prerequisites for Select AI.

Example: Use Summarize Action on SQL Command Line

The following example uses `SUMMARIZE` as a Select AI action. Use `SELECT AI SUMMARIZE <TEXT>` in the SQL command line to generate a summary of input text.

```
SELECT AI SUMMARIZE
Like countless other people around the globe, I stream music, and like more
than six hundred million of them I mainly use Spotify. Streaming currently
accounts for about eighty per cent of the American recording industry’s
revenue, and in recent years Spotify’s health is often consulted as a measure
for the health of the music business over all. Last spring, the International
Federation of the Phonographic Industry reported global revenues of $28.6
billion, making for the ninth straight year of growth. All of this was
unimaginable in the two-thousands, when the major record labels appeared
poorly equipped to deal with piracy and the so-called death of physical media.
On the consumer side, the story looks even rosier. Adjusted for inflation, a
```

```
... (skipped 1000 rows in the middle)

Pelly writes of some artists, in search of viral fame, who surreptitiously use
social media to effectively beta test melodies and motifs, basically putting
together songs via crowdsourcing. Artists have always fretted about the
pressure to conform, but the data-driven, music-as-content era feels
different. “You are a Spotify employee at that point,” Daniel Lopatin, who
makes abstract electronic music as Oneohtrix Point Never, told Pelly. “If your
art practice is so ingrained in the brutal reality that Spotify has outlined
for all of us, then what is the music that you’re not making? What does the
music you’re not making sound like?” Listeners might wonder something similar.
What does the music we’re not hearing sound like?;


RESPONSE
--------------------------------------------------------------------------------
The music streaming industry, led by Spotify, has revolutionized the way
people consume music, with streaming accounting for 80% of the American
recording industry's revenue. However, this shift has also complicated the
lives of artists, who struggle to survive in a hyper-abundant present where
music is often valued for its convenience rather than its artistic merit.
Spotify's algorithms prioritize popularity and profitability over artistic
diversity, leading to a homogenization of music and a devaluation of the
labor that goes into creating it. Meanwhile, the company's executives reap
enormous profits, with CEO Daniel Ek's net worth rivaling that of the
wealthiest musicians. As music critic Liz Pelly argues, the streaming economy
raises important questions about autonomy, creativity, and the value of art
in a world where everything is readily available and easily accessible.
```

Tip

In SQL*Plus, a single quotation mark (') is treated as a string delimiter. If your text contains single quotes, either escape the quote by doubling it (' to ''), or enclose the text using the `q'[]'` quoting mechanism. If your text contains empty double quotes (""), enclose the text using `q'[]'` mechanism. For example:

```
SELECT AI SUMMARIZE q'[this's a text]';
```

Example: Use DBMS_CLOUD_AI.SUMMARIZE Procedure to Generate a Summary

These examples demonstrate generating a summary by using different parameters from the `DBMS_CLOUD_AI.SUMMARIZE` procedure.

You can generate a summary from 3000+ word text stored in an OCI object storage by specifying the object storage link as the `location_uri` parameter and your cloud account credentials as `credential_name` using the `DBMS_CLOUD_AI.SUMMARIZE`

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
                location_uri => 'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/' ||
                   'namespace-string/b/bucketname/o/data_folder/' ||
                   'summary/test_4000_words.txt',
```

```
                credential_name => 'STORE_CRED',
                profile_name => 'GENAI')
from DUAL;
```

Another way to generate a summary from a text that is stored in an OCI object storage is by using the `content` parameter to call the `DBMS_CLOUD.GET_OBJECT` procedure.

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
                content => TO_CLOB(
                            DBMS_CLOUD.GET_OBJECT(
                                credential_name => 'STORE_CRED',
                                location_uri => 'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/' ||
                    'namespace-string/b/bucketname/o/data_folder/' ||
                    'summary/test_4000_words.txt')),
                profile_name => 'GENAI'>)
from DUAL;
```

Example: Generate a Summary by Specifying User Prompt, Minimum Words, and Maximum Words

The following example demonstrates generating a summary of a 3000+ word text by specifying the following parameters:

- `user_prompt`: *The summary should start with 'The summary of the article is: '*
- `min_words`: *50*
- `max_words`: *100*

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
                content => TO_CLOB(
                             DBMS_CLOUD.GET_OBJECT(
                             credential_name =>'STORE_CRED',
                             location_uri =>'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/' ||
                                 'namespace-string/b/bucketname/o/
data_folder/' ||
                                 'summary/test_4000_words.txt')),
                profile_name    => 'GENAI',
                user_prompt     => 'The summary should start with ''The
summary of ' ||
                                   'the article is: ''',
                params          => '{"min_words":50,"max_words":100}')
As response FROM dual;


RESPONSE
--------------------------------------------------------------------------------
--
The summary of the article is: The music streaming industry, led by Spotify,
has
 revolutionized the way people consume music, with streaming accounting for
abou
t eighty per cent of the American recording industry's revenue. However, this
sh
ift has also raised concerns about the impact on artists, with many
```

```
struggling t
o make a living due to low royalty rates and the dominance of playlists. The
art
icle explores the history of music streaming, from the early days of Napster
to
the current landscape, and how it has changed the way people listen to music.
It
 also delves into the issues of autonomy and creativity in the music
industry, w
ith some artists feeling pressured to conform to certain styles or formulas
to s
ucceed on platforms like Spotify. The article cites examples of artists who
have
 spoken out against the streaming economy, including Taylor Swift and Neil
Young
, and discusses the rise of alternative platforms like Bandcamp and Nina.
Ultima
tely, the article suggests that the streaming economy has created a perverse
vis
ion for art, where music is valued for its ability to be ignored rather than
app
reciated, and that this has significant implications for the future of music
and
 creativity. With the rise of AI-generated music and the increasing
importance o
f data-driven decision making in the music industry, the article asks what
the m
usic we're not hearing sounds like, and what the consequences of this shift
will
 be for artists and listeners alike. The article concludes by highlighting
the n
eed for a more nuanced understanding of the music industry and the impact of
str
eaming on artists and listeners, and for alternative models that prioritize
crea
tivity and autonomy over profit and convenience.
```

Example: Generate a Summary by Specifying User Prompt, Maximum Words, and Summary Style

The following example demonstrates generating a summary of a 12000+ word text by specifying the following parameters:

- `user_prompt`: *The summary should start with 'The summary of the article is: '*
- `max_words`: *100*
- `summary_style`: *list*

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
                location_uri    => 'https://objectstorage.ca-toronto-1.' ||
                                   'oraclecloud.com/n/namespace-string/b/' ||
                                   '/bucketname/o/data_folder/' ||
                                   'summary/dreams.txt',
                credential_name => 'STORE_CRED',
                profile_name    => 'GENAI',
                user_prompt     => 'The summary should start with ''The
summary of ' ||
```

```
                                   'the article is: ''',
                params          => '{"max_words":100,
"summary_style":"list"}')
As response FROM dual;


RESPONSE
--------------------------------------------------------------------------------
--
The summary of the article is:
- The book "Dreams" by Henri Bergson explores the concept of dreams and their
si
gnificance in understanding human consciousness.
- Bergson argues that dreams are not just random thoughts, but rather a way
for
our unconscious mind to process and consolidate memories.
- He suggests that dreams are a result of the relaxation of our mental
faculties
, which allows our unconscious mind to freely associate and create new
connectio
ns between memories.
- The book also discusses the role of sensations, such as visual and auditory
im
pressions, in shaping our dreams.
- Bergson's theory of dreams is compared to other theories, including those
of F
reud and Jung, and is seen as a unique and insightful contribution to the
field
of psychology.
- The book concludes by highlighting the importance of studying dreams in
order
to gain a deeper understanding of human consciousness and the workings of the
mi
nd.
```

Example: Generate a Summary of a Book

This example demonstrates passing a 35.66 MiB file as an input to generate a summary. The `DBMS_CLOUD_AI.SUMMARIZE` function uses iterative refinement method to process the chunks. See Iterative Refinement for more information.

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
       location_uri    => 'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/namespace-string/b/' ||
                          'bucketname/o/data_folder/summary/
Descartes_An_Intellectual_Biography.pdf',
       credential_name => 'STORE_CRED',
       profile_name    => 'GENAI',
       params          =>
'{"chunk_processing_method":"iterative_refinement"}')
AS response FROM dual;


RESPONSE
--------------------------------------------------------------------------------
--
```

```
Stephen Gaukroger's intellectual biography of Rene Descartes provides a
detailed
 examination of the philosopher's crucial role in shaping modern thought,
placin
g him within the cultural, religious, and scientific context of the early
sevent
eenth century. It traces Descartes' intellectual journey from his education
at L
a Fleche, where he rejected Aristotelian logic, to his influential
interactions
with figures like Isaac Beeckman, which shaped his mechanistic worldview
evident
 in works like his hydrostatics manuscript and *Compendium Musicae*. The
biograp
hy underscores Descartes' dual commitment to philosophy and science,
highlightin
g his social status among the gentry, mathematical innovations such as
solving t
he Pappus problem through algebraic geometry, and his epistemology based on
clea
r and distinct ideas. It explores his mechanistic explanations of bodily
functio
ns, challenging traditional soul-body distinctions, and his extensive natural
ph
ilosophy in texts like *Le Monde* and *L'Homme*. Gaukroger also delves into
Desc
artes' cosmological theories, including the vortex theory and laws of motion
lin
ked to divine immutability, as well as his nuanced perspectives on animal
cognit
ion versus human consciousness. Central to the narrative is Descartes' use of
hy
perbolic doubt to combat skepticism and establish metaphysical foundations
throu
gh the *cogito*, alongside his classification of ideas and theological proofs
of
 God's existence. The complex relationship between his natural philosophy and
me
taphysics, especially in defining motion as a mode, and his innovative
approach
to the passions in *Passions of the Soul*, rejecting Stoic views for a mind-
body
 union, are key themes. This portrayal captures Descartes' struggle with
traditi
onal paradigms during a transformative era, emphasizing his enduring impact
on p
hilosophy and science.
```

Example: Select AI Translate

These examples demonstrate how you can use the `translate` capability.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- #unique_34
- Use AI Keyword to Enter Prompts

Example: Use Translate Action on the SQL Command Line
The following example shows using the `translate` action on the SQL command line.

> **Note**
>
> Your AI profile must specify the target language. This feature is supported only for the provider OCI.

```
--Create an AI profile with language parameters
BEGIN
DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name
=>'GENAI_NEW',
      attributes   =>'{"provider":
"oci",
        "credential_name": "GENAI_CRED",
        "target_language": "french",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}]
       }');
END;
/
PL/SQL procedure successfully completed.

SQL> exec DBMS_CLOUD_AI.SET_PROFILE('GENAI_NEW');

PL/SQL procedure successfully completed.

SQL> select ai translate I need to translate this;

RESPONSE
--------------------
Je dois traduire ceci
```

Example: Use Translate in DBMS_CLOUD_AI.GENERATE Function
The following examples show using translate as a Select AI action within the `DBMS_CLOUD_AI.GENERATE` function. See #unique_98 for more information.

> **Note**
>
> The AI profile can skip specifying the target language parameter if it is passed as an attribute in `DBMS_CLOUD_AI.GENERATE`.

The `translate` action is supplied in the `DBMS_CLOUD_AI.GENERATE` function along with `target_language` and `source_language`. This example uses generative AI translation. The input text `this is a document` in English (`source_language: "en"`) is translated into French (`target_language: "fr"`).

```
SELECT DBMS_CLOUD_AI.GENERATE('select ai translate text to be translated')
          FROM dual;

      DECLARE
         l_attributes  clob := '{"target_language": "fr", "source_language":
"en"}';
         output clob;
      BEGIN
         output := DBMS_CLOUD_AI.GENERATE(
                        prompt           => 'this is a document',
                        profile_name     => 'oci_translate',
                        action           => 'translate',
                        attributes       => l_attributes
                     );
```

Example: Use DBMS_CLOUD_AI.TRANSLATE Function for Translation
This example calls the `DBMS_CLOUD_AI.TRANSLATE` function to use generative AI translation, converting the input text from English (`source_language`) into French (`target_language`) using the specified AI profile.

See #unique_136 for more details.

```
BEGIN
   output_text := DBMS_CLOUD_AI.TRANSLATE(
   profile_name    => 'GENAI_NEW'
   text            => 'text to be translated',
   source_language => 'English',
   target_language => 'French');
END;
/
```

Example: Display Supported Languages for a Provider

Query the `AI_TRANSLATION_LANGUAGES` view to see a list of languages that your AI provider supports. See #unique_153 for details.

```
SELECT* FROM AI_TRANSLATION_LANGUAGES;

LANGUAGE_NAME        LANGUAGE_CODE        PROVIDER
-------------------- -------------------- ---------------
ARABIC               ar                   OCI
CROATIAN             hr                   OCI
CZECH                cs                   OCI
DANISH               da                   OCI
GERMAN               de                   OCI
GREEK                el                   OCI
ENGLISH              en                   OCI
SPANISH              es                   OCI
```

```
FINNISH              fi                  OCI
FRENCH               fr                  OCI
FRENCH CANADA        fr-CA               OCI
HEBREW               he                  OCI
HUNGARIAN            hu                  OCI
ITALIAN              it                  OCI
```

Example: Use a Different Profile for Natural Language to SQL Generation
This example shows setting up a different profile named `openai` with OpenAI as the provider. You can use this profile for generating SQL among other Select AI features and actions from your prompts.

> **Note**
>
> The profile in this example does not support `translate` capability.

```
-- OpenAI profile used for sql generation
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name = > 'openai',
      attributes => '{"provider": "openai",
         "credential_name": "OPENAI_CRED",
          "object_list": [{"owner": "ADB_USER", "name": "GENRE"},
                    {"owner": "ADB_USER", "name": "CUSTOMER"},
                    {"owner": "ADB_USER", "name": "PIZZA_SHOP"},
                    {"owner": "ADB_USER", "name": "STREAMS"},
                    {"owner": "ADB_USER", "name": "MOVIES"},
                    {"owner": "ADB_USER", "name": "ACTIONS"}]
        }');
END;
/

EXEC DBMS_CLOUD_AI.SET_PROFILE('openai');
SELECT AI SHOWSQL Give me the total number of customers;

RESPONSE
------------
SELECT COUNT("Customer_ID") AS "Total_Customers" FROM "Customers"
```

Example: Restrict Table Access in AI Profile

This example demonstrates how to restrict table access and instruct the LLM to use only the tables specified in the `object_list` of the AI profile.

Set `enforce_object_list` to *true* to restrict table access to the LLM.

As a database user, create and configure your AI profile. Review Perform Prerequisites for Select AI to configure your AI profile.

```
BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'GOOGLE_ENFORCED',
        attributes   =>'{"provider": "google",
```

```
          "credential_name": "GOOGLE_CRED",
          "object_list": [{"owner": "ADB_USER", "name": "GENRE"},
                  {"owner": "ADB_USER", "name": "CUSTOMER"},
                  {"owner": "ADB_USER", "name": "PIZZA_SHOP"},
                  {"owner": "ADB_USER", "name": "STREAMS"},
                  {"owner": "ADB_USER", "name": "MOVIES"},
                  {"owner": "ADB_USER", "name": "ACTIONS"}],
          "enforce_object_list" : "true"
          }');
END;
/

PL/SQL procedure successfully completed.


EXEC DBMS_CLOUD_AI.set_profile('GOOGLE_ENFORCED');

PL/SQL procedure successfully completed.

select ai showsql please list the user tables;

RESPONSE
--------------------------------------------------------------------------------
--------------
SELECT 'ADB_USER.GENRE' AS TABLE_NAME FROM DUAL UNION ALL SELECT
'ADB_USER.CUSTOMER' AS
TABLE_NAME FROM DUAL UNION ALL SELECT 'ADB_USER.PIZZA_SHOP' AS TABLE_NAME
FROM DUAL UNION
 ALL SELECT 'ADB_USER.STREAMS' AS TABLE_NAME FROM DUAL UNION ALL SELECT
'ADB_USER.MOVIES'
AS TABLE_NAME FROM DUAL
--
```

Setting `enforce_object_list` to *false* instructs the LLM to use other tables and views based on its prior knowledge.

```
BEGIN
      DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name =>'GOOGLE_ENFORCED1',
        attributes   =>'{"provider": "google",
          "credential_name": "GOOGLE_CRED",
          "object_list": [{"owner": "ADB_USER", "name": "GENRE"},
                  {"owner": "ADB_USER", "name": "CUSTOMER"},
                  {"owner": "ADB_USER", "name": "PIZZA_SHOP"},
                  {"owner": "ADB_USER", "name": "STREAMS"},
                  {"owner": "ADB_USER", "name": "MOVIES"},
                  {"owner": "ADB_USER", "name": "ACTIONS"}],
          "enforce_object_list" : "false"
          }');
END;
/

PL/SQL procedure successfully completed.
```

```
EXEC DBMS_CLOUD_AI.set_profile('GOOGLE_ENFORCED1');

PL/SQL procedure successfully completed.

select ai showsql please list the user tables;
RESPONSE
--------------------------------
SELECT TABLE_NAME FROM USER_TABLES
```

Example: Specify Case Sensitivity for Columns

This example shows how you can set case sensitivity for columns in AI profile.

Set `case_sensitive_values` to *false* to retrieve queries that are not case sensitive.

As a database user, create and configure your AI profile. Review Perform Prerequisites for Select AI to configure your AI profile.

```
BEGIN
      DBMS_CLOUD_AI.create_profile(
        profile_name   =>'GOOGLE',
        attributes     =>'{"provider": "google",
          "credential_name": "GOOGLE_CRED",
          "object_list": [{"owner": "ADB_USER", "name": "GENRE"},
                  {"owner": "ADB_USER", "name": "CUSTOMER"},
                  {"owner": "ADB_USER", "name": "PIZZA_SHOP"},
                  {"owner": "ADB_USER", "name": "STREAMS"},
                 {"owner": "ADB_USER", "name": "MOVIES"},
                 {"owner": "ADB_USER", "name": "ACTIONS"}],
         "case_sensitive_values" : "false"
         }');
 END;
 /

PL/SQL procedure successfully completed.

-- With "case_sensitive_values" set to "false", LLM will give back case
insensitive query.

select ai showsql how many people watch Inception;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------------
SELECT COUNT(DISTINCT c.CUSTOMER_ID) AS "COUNT"
FROM "ADB_USER"."CUSTOMER" c
JOIN "ADB_USER"."STREAMS" s ON c.CUSTOMER_ID = s.CUSTOMER_ID
JOIN "ADB_USER"."MOVIES" m ON s.MOVIE_ID = m.MOVIE_ID
WHERE UPPER(m.TITLE) = UPPER('Inception')
```

You can specify case sensitive query using double quotes even though the `case_sensitive_values` is set to *false*.

```
select ai showsql how many people watch "Inception";

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------
SELECT COUNT(DISTINCT c.CUSTOMER_ID) AS "COUNT"
FROM "ADB_USER"."CUSTOMER" c JOIN "ADB_USER"."STREAMS" s ON
c.CUSTOMER_ID = s.CUSTOMER_ID JOIN "ADB_USER"."MOVIES" m ON
s.MOVIE_ID = m.MOVIE_ID WHERE m.TITLE = 'Inception'
```

Part II
Select AI Agent

- Select AI Agent
  Select AI Agent (autonomous agent framework) enables you to build interactive and autonomous agents inside Autonomous AI Database, combining planning, tool use, reflection, and memory to deliver multi-turn workflows.
- Select AI Agent for Python
- Select AI Agent Concepts
  Explores the concepts and terms related to Select AI Agent (autonomous agent framework).
- Select AI Agent Use Cases
  Select AI Agent supports practical scenarios across roles: connect external services, build task-focused automation with PL/SQL tools, and enable conversational access to data using natural-language interactions.
- Prerequisites for Using Select AI Agent
  Before you use Select AI Agent, you must have privileges to use the `DBMS_CLOUD_AI_AGENT` package and all other privileges that are required for Select AI.
- Getting Started with Select AI Agent
  To get started, review the prerequisites and the tasks that you need to perform to use Select AI Agent.
- How Do I Use Select AI Agent
  You can use Select AI Agent by using the `agent` action or the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function.
- Examples of Using Select AI Agent
  Explore examples that show how to build, configure, and interact with Select AI Agent for common tasks such as movie analysis, log analysis and customer support.

19

Select AI Agent

Select AI Agent (autonomous agent framework) enables you to build interactive and autonomous agents inside Autonomous AI Database, combining planning, tool use, reflection, and memory to deliver multi-turn workflows.

- About Select AI Agent
  Select AI Agent (autonomous agent framework) is a program for creating and managing interactive and autonomous agents inside Autonomous AI Database. Agents reason about requests, call tools, reflect on results, and maintain context with short and long-term memory powered by an AI profile specified LLM with the ReAct (Reasoning and Acting) agentic pattern.

About Select AI Agent

Select AI Agent (autonomous agent framework) is a program for creating and managing interactive and autonomous agents inside Autonomous AI Database. Agents reason about requests, call tools, reflect on results, and maintain context with short and long-term memory powered by an AI profile specified LLM with the ReAct (Reasoning and Acting) agentic pattern.

Select AI Agent enables the use of built-in tools such as RAG and Natural Language to SQL (NL2SQL), custom PL/SQL procedures, and external REST APIs to complete tasks. The framework preserves multi-turn memory, maintaining context across conversations. Together, these capabilities support scalable, context-aware generative AI that integrates with enterprise data and workflows.

The `DBMS_CLOUD_AI_AGENT` package encapsulates management, orchestration, and security boundaries. See #unique_163 for details.

Topics

- Features of Select AI Agents
  The key features include integrated intelligence, flexible tooling, context-aware conversations, and faster deployment.
- ReAct Agentic Pattern
  Select AI Agent uses ReAct (Reasoning and Acting) agentic pattern where the agent reasons about the request, chooses tools, performs actions, and evaluates results to accomplish a goal.
- Select AI Agent Architecture
  Select AI Agent organizes work into four layers: Planning, Tool Use, Reflection, and Memory Management. These layers coordinate reasoning, tool runs, evaluation, and context multi-turn interactions.

Features of Select AI Agents

The key features include integrated intelligence, flexible tooling, context-aware conversations, and faster deployment.

- Integrated Intelligence:

Combines planning, tool use, and reflection so agents can reason about tasks, choose and run tools, observe outcomes, adjust plans, and improve responses throughout conversation. Agents plan steps, run tools, evaluate observations, and update their approach when outcomes miss expectations. This loop strengthens accuracy, reduces rework, and keeps conversations on track.

- Flexible Tooling:

  Support and use built-in RAG and NL2SQL, custom PL/SQL procedures, and external REST services, without orchestration components or separate infrastructure, so you can keep core logic in the database while integrating external capabilities as needed.

- Context-Aware Conversations:

  Maintain short-term and long-term memory to keep context across turns, personalize responses, store preferences, and support human-in-the-loop control for corrections and confirmations during multi-turn sessions. Short-term memory keeps the current dialogue coherent. Long-term memory records preferences and prior outcomes, supporting follow-up interactions and oversight by human reviewers.

- Scalable and Secure:

  Run inside Autonomous AI Database, inherit its security controls, auditing, and performance, reduce data movement, and standardize governance for enterprise deployments and regulated environments at scale. Agents benefit from database security, auditing, and performance characteristics. Keeping processing close to data reduces movement and aligns with governance practices.

- Faster Development:

  Define agents, tasks, and tools with familiar SQL and PL/SQL, reuse existing procedures, and ship features faster while keeping logic close to operational data and teams without building separate infrastructure.

ReAct Agentic Pattern

Select AI Agent uses ReAct (Reasoning and Acting) agentic pattern where the agent reasons about the request, chooses tools, performs actions, and evaluates results to accomplish a goal.

ReAct combines reasoning and action in a loop. The agent thinks, chooses a tool, observes results, and repeats until it can present a confident answer. The user's AI profile specified LLM alternates between reasoning and actions through the tools. The database processes those actions and returns the observations.

The following is the pattern for each iteration:

1. Query: The user asks a question or states a request. The agent reads it, extracts key details, and prepares to plan the next steps.
2. Thought and Action: The agent reasons about options, picks a tool, and runs it to gather data or change state as needed for the task.
3. Observation: Observations include tool or query results, confirmation messages, and errors. These become inputs to the agent’s next round of reasoning. The agent records observations and checks whether the results support the next step or the final response.
4. Final Response: After enough successful thought-action and observations, the agent composes a clear answer, explains important decisions, and shares any next steps or follow-up actions.

Select AI Agent Architecture

Select AI Agent organizes work into four layers: Planning, Tool Use, Reflection, and Memory Management. These layers coordinate reasoning, tool runs, evaluation, and context multi-turn interactions.

Planning: Planning interprets the user request, breaks it into ordered actions, selects candidate tools, and drafts a plan using session context, prior outcomes, and relevant knowledge. The agent analyzes the request, identifies missing details, and proposes an ordered sequence of actions. It chooses tools that fit policy, data scope, and expected outcomes.

Tool Use: Tool Use selects and runs the tool for each action. Supported types include RAG, NL2SQL, custom PL/SQL procedures that can be added when you create a tool, and external REST services such as web search and email. Each step calls a tool with parameters. Built-in tools handle retrieval and SQL generation. Custom PL/SQL encapsulates domain logic. REST tools connect to external services.

Reflection: Reflection evaluates tool results against expectations and proceeds to final response. The agent compares observations to the goal. If results look wrong or if there are tool call errors or user disapproved results, the agent revises reasoning, chooses another tool, or updates the plan before trying again. When results do not fit, it adjusts the plan, selects different tools, or may ask clarifying questions before proceeding. Select AI Agent thoughts can be queried using `USER_CLOUD_AI_CONVERSATION_PROMPTS`. See #unique_77 for more information.

Memory Management: Memory Management stores session context and knowledge per agent team. Short-term memory holds recent messages and intermediate results per agent team. Long-term memory records preferences, history, and strategies, improving continuity, personalization, and planning. Long-term memory persists useful knowledge across sessions, improving guidance and response quality over time across agent teams.

20

Select AI Agent for Python

Select AI Agent for Python builds on the Select AI for Python client library, `select_ai`, enabling you to leverage `DBMS_CLOUD_AI` features within Autonomous AI Database directly from Python. The module extends support for advanced generative AI workflows and agent-based automation through the `select_ai.agent` submodule.

What You Can Do

You can now design and orchestrate agentic workflows in Python with the following classes:

- `select_ai.agent.Tool`
- `select_ai.agent.Task`
- `select_ai.agent.Agent`
- `select_ai.agent.Team`

These classes enable you to programmatically define tools, construct tasks, configure agents, and assemble multi-agent teams in Python. This approach closely mirrors the structure and capabilities provided by the `DBMS_CLOUD_AI_AGENT` package in the database, giving you flexible control and seamless integration with Autonomous AI Database AI operations.

Async Select AI Agent Support

The `select_ai.agent` module also includes asynchronous versions of its core classes, enabling you to build and run agent workflows using Python's `async` and `await`. These async classes are designed for co-routine-based applications and enable non-blocking interaction with the database.

You can use the following async classes:

- `select_ai.agent.AsyncTool`
- `select_ai.agent.AsyncTask`
- `select_ai.agent.AsyncAgent`
- `select_ai.agent.AsyncTeam`

See "*Async AI agent examples*" in *Select AI for Python* to explore.

These asynchronous classes support the same core functionality as their synchronous counterparts, including:

- Creating tools for natural-language-to-SQL generation (NLSQL), web search, retrieval-augmented generation (RAG), PL/SQL, notifications, and custom functions
- Configuring task logic and tool usage
- Assigning agent roles and profiles
- Assembling and running agent teams programmatically

This enables you to build scalable AI pipelines that integrate naturally with Python async applications.

For complete API reference, see Select AI for Python guide.

21

Select AI Agent Concepts

Explores the concepts and terms related to Select AI Agent (autonomous agent framework).

- Agentic Action
  In the context of agents, an action is an instruction that triggers a tool with parameters. The platform runs the tool, handles errors, and returns an observation for the next reasoning step loop.
- Agent
  An agent performs tasks for a defined purpose. An agent is a configured worker that reasons about requests, selects tools, runs steps, evaluates results, and produces responses grounded in database context.
- Agent Team
  One or more agents that perform an agentic workflow. A team coordinates responsibilities, shares context, and runs agent-task pairs to complete multi-step interactions reliably.
- MCP Server
- Observation
  The data returned from a tool run: rows, messages, or errors, that the agent records and evaluates during reflection. Observations guide the next reasoning step and provide evidence for the final response.
- Task
  A task represents a unit of work. It guides tool selection, parameter mapping, and execution policy, producing results that downstream steps can read and summarize.
- Tool

Agentic Action

In the context of agents, an action is an instruction that triggers a tool with parameters. The platform runs the tool, handles errors, and returns an observation for the next reasoning step loop.

Actions connect planning to execution. They define which tool to run and with which inputs.

Agent

An agent performs tasks for a defined purpose. An agent is a configured worker that reasons about requests, selects tools, runs steps, evaluates results, and produces responses grounded in database context.

An agent encapsulates behavior, policies, and available tools for a particular job, such as returns processing or knowledge retrieval.

Agent Team

One or more agents that perform an agentic workflow. A team coordinates responsibilities, shares context, and runs agent-task pairs to complete multi-step interactions reliably.

Teams divide work among specialized agents and sequence their contributions. Teams keep shared context and produce a unified response.

You can view agent and team run details using the following history views: `USER_AI_AGENT_TEAM_HISTORY`, `USER_AI_AGENT_TASK_HISTORY`, and `USER_AI_AGENT_TOOL_HISTORY`. These views record prompts, responses, and tool runs for each agent, task, and team, providing transparency and traceability across agentic workflows. For more information about these history views, see the DBMS_CLOUD_AI_AGENT History Views.

MCP Server

An MCP Server (Model Context Protocol Server) provides a standardized way for applications to interact with generative AI models and tools by exposing model capabilities, context retrieval, and extensions through a consistent protocol.
Using Select AI Agent, you can define tools that can be exposed through the Autonomous AI Database MCP Server. These tools can include built-in tools for SQL generation, RAG, notifications (email and Slack), and websearch, as well as custom tools. Custom tools can expose functions defined in your database.

Observation

The data returned from a tool run: rows, messages, or errors, that the agent records and evaluates during reflection. Observations guide the next reasoning step and provide evidence for the final response.

Task

A task represents a unit of work. It guides tool selection, parameter mapping, and execution policy, producing results that downstream steps can read and summarize.

Tasks specify the goal, inputs, tool choices, and guardrails. They return structured outputs that later steps can consume.

You can query the `USER_AI_AGENT_TASK_HISTORY` view to inspect how tasks are defined, run, and completed within an agentic run. See the BMS_CLOUD_AI_AGENT History Views for more information.

Tool

A tool performs an action such as updating data, retrieving documents, or calling external services. Tools take parameters, can run deterministically or non-deterministically, and return observations for reasoning.
Tools encapsulate repeatable operations. They keep side effects controlled and observable, supporting auditing and debugging.

In the Select AI Agent framework, tools represent functional building blocks that agents can use during task processing. A tool might be:

- A deterministic tool, such as a PL/SQL function, which returns the same output for the same input.
- A non-deterministic tool, such as a web search or LLM-based summarization, where outputs may vary depending on timing, network results, or randomness.

Agents select tools based on the task's needs and use the output (known as an observation) to reason, decide next steps, or generate responses. Tool metadata and run history are logged to support debugging, observability, and security.

You can query the `USER_AI_AGENT_TOOL_HISTORY` view to monitor tool usage and review run outcomes for each agent task. See the DBMS_CLOUD_AI_AGENT History Views for more information.

22

Select AI Agent Use Cases

Select AI Agent supports practical scenarios across roles: connect external services, build task-focused automation with PL/SQL tools, and enable conversational access to data using natural-language interactions.

Use cases include:

Integrate external APIs for enriched responses:

Connect REST services such as shipping status, knowledge articles, or messaging. The agent blends external results with database facts and returns timely, actionable guidance.

Build task-specific agents using PL/SQL tools:

Create focused agents that call PL/SQL tools for updates, validations, or transformations, keeping logic close to data, improving performance, and simplifying maintenance across environments consistently. Wrap existing PL/SQL for updates and checks. Agents coordinate tasks, enforce parameters, and record outcomes while staying within transactional controls.

Enable conversational access to enterprise data:

Expose data through natural-language questions grounded on database objects. Use NL2SQL for retrieval, apply policies, and present findings clearly while respecting security and governance requirements. NL2SQL translates questions to SQL on approved objects. Agents summarize results and next steps while following object lists and access rules.

23

Prerequisites for Using Select AI Agent

Before you use Select AI Agent, you must have privileges to use the `DBMS_CLOUD_AI_AGENT` package and all other privileges that are required for Select AI.

To use Select AI Agent, you must have the following:

- Perform Prerequisites for Select AI
- Grant Privileges for Select AI
- Grant Privileges for Select AI Agent
- At least one agent team defined. See #unique_175.
- Optionally, a `conversation_id` is created to keep track of multi-turn context.

Topics

- Grant Privileges for Select AI Agent
  To use Select AI Agent, the administrator must grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI_AGENT` package.

Grant Privileges for Select AI Agent

To use Select AI Agent, the administrator must grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI_AGENT` package.

To configure `DBMS_CLOUD_AI_AGENT`:

Grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI_AGENT` package to the user who wants to use Select AI Agent.

By default, only the system administrator has `EXECUTE` privilege. The administrator can grant `EXECUTE` privilege to other users.

Example of Privileges to Run Select AI Agent

The following example grants the `EXECUTE` privilege to `ADB_USER`:

```
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
```

See also Examples of Privileges to Run Select AI.

24

Getting Started with Select AI Agent

To get started, review the prerequisites and the tasks that you need to perform to use Select AI Agent.

- Configure your system to use Select AI Agent
- Create an AI profile and enable the AI profile
- Configure your Select AI Agent by reviewing DBMS_CLOUD_AI_AGENT Package
- Use Select AI action and a prompt or call DBMS_CLOUD_AI_AGENT.RUN_TEAM
- Examples of Using Select AI Agent
- Review Select AI Agent views and history by querying DBMS_CLOUD_AI_AGENT Views for troubleshooting or diagnostics

25

How Do I Use Select AI Agent

You can use Select AI Agent by using the `agent` action or the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function.

You can use Select AI Agent in the following ways:

- Use `agent` as Select AI action: `Select AI agent <prompt>`.
- Call the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` and supply a prompt within the function along with other parameters.

Both options route the prompt to the chosen agent team and return the team's reply.

See Also:

Examples of Using Select AI Agent to learn more.

26

Examples of Using Select AI Agent

Explore examples that show how to build, configure, and interact with Select AI Agent for common tasks such as movie analysis, log analysis and customer support.

- Example: Create an Agent
- Example: Create Built-In Tools
- Example: Create a Task
- Example: Create an Agent Team
- Example: Create a Movie Analysis Agent with Built-In Tools
- Example: Create a Product Return Agent
- Example: Fetch and Analyze Log Reports
- Example: Create a Customized HTTP Tool
- Example: View Agent Prompts and Responses from the Latest Team Run
- Example: Resume an Agent Team Run from WAITING_FOR_HUMAN State

Example: Create an Agent

Create an agent to perform a defined task.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- #unique_175

Example: Create an Agent
This example creates an agent named `Customer_Return_Agent` responsible for handling product return-related conversations.

This example illustrates using Google as the AI provider as specified in the AI profile named `GOOGLE`. The AI profile identifies the LLM the agent uses for reasoning and responses. The `role` attribute provides instructions to guide the agent.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name => 'CustomerAgent',
    attributes =>'{
       "profile_name": "GOOGLE",
                       "role": "You are an experienced customer agent who
deals with customers return request."
     }'
  );
```

```
END;
/
```

> **Note**
>
> Each agent in a multi-agent team can have a distinct AI profile and each profile may use a different AI provider and/or LLM.

Example: Create Built-In Tools

Create built-in tools such as SQL, RAG, Websearch, Email, and Slack. These tools enable agents and tasks to query data, retrieve knowledge, search the web, and send notifications.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- #unique_187
- #unique_188

Select AI Agent accepts the following types of tools:

- `SQL`
- `RAG`
- `WEBSEARCH`
- `NOTIFICATION`
  - `EMAIL`
  - `SLACK`

> **Note**
>
> Built-in tools (for example, SQL and RAG) include internal instructions. You can optionally append user-provided instructions to tailor behavior for your use case. Appending an instruction can add task-specific context (for example, scope queries to a specific schema or help enforce a response format). This can improve relevance for specialized scenarios. Poorly written or overly prescriptive instructions can degrade tool performance. Keep instructions short, specific, and consistent with the tool's purpose.

Example: SQL Tool
This example creates a `SQL` tool that translates natural language queries into SQL statements. SQL tool helps the agents answer data-related questions by mapping prompts into SQL queries.

This example shows using OCI as the AI provider as specified in the AI profile named `nl2sql_profile`. The AI profile identifies the LLM the agent uses for reasoning and responses. In this example, the AI profile `nl2sql_profile` defines the set of SH schema tables that the

agent can query, enabling natural language access to commonly used sales history data such as customers, products, promotions, and countries. The SQL tool uses the database objects that are specified in the AI profile, ensuring that generated SQL statements remain accurate and relevant to the SH data set.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      profile_name => 'nl2sql_profile',
      attributes   => '{"provider": "oci",
        "credential_name": "GEN1_CRED",
        "oci_compartment_id": "ocid1.compartment.oc1..aaaa...",
    object_list": [{"owner": "SH", "name": "customers"},
                    {"owner": "SH", "name": "countries"},
                    {"owner": "SH", "name": "supplementary_demographics"},
                    {"owner": "SH", "name": "profits"},
                    {"owner": "SH", "name": "promotions"},
                    {"owner": "SH", "name": "products"}]

       }');
end;
/

EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('SQL');
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'SQL',
    attributes => '{"tool_type": "SQL",
                    "tool_params": {"profile_name": "nl2sql_profile"}}'
  );
END;
/
```

Example: RAG Tool
This example creates a RAG (Retrieval Augmented Generation) tool. The RAG tool lets agents retrieve and ground responses in enterprise documents, improving accuracy for knowledge-based answers.

This example illustrates defining a `RAG_PROFILE` with credentials, a vector index, and specifying profile parameters. Then, creating a vector index `RAG_INDEX` in Object Storage for document embeddings and creating the `RAG_TOOL` linked to your profile.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name =>'RAG_PROFILE',
          attributes   =>'{"provider": "oci",
            "credential_name": "GENAI_CRED",
            "vector_index_name": "RAG_INDEX",
            "oci_compartment_id": "ocid1.compartment.oc1..aaaa..",
            "temperature": 0.2,
            "max_tokens": 3000
          }');
END;
/

BEGIN
```

```
   DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
           index_name  => 'RAG_INDEX',
           attributes  => '{"vector_db_provider": "oracle",
                           "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
                           "object_storage_credential_name": "OCI_CRED",
                           "profile_name": "RAG_PROFILE",
                           "chunk_overlap":128,
                           "chunk_size":1024
   }');
END;
/

EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('RAG_TOOL');
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'RAG_TOOL',
    attributes => '{"tool_type": "RAG",
                    "tool_params": {"profile_name": "RAG_PROFILE"}}'
  );
END;
/
```

Example: Websearch Tool
This example creates a Websearch tool for retrieving details from the internet. The Websearch tool enables agents to look up information from the web, such as product prices or descriptions.

This example illustrates adding an ACL entry for the OpenAI provider. Creating credentials `OPENAI_CRED` with your API key and creating the Websearch tool, describing its purpose, linking it to the credential.

> **Note**
>
> OpenAI is the only Websearch AI provider supported.

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'api.openai.com',
    ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                        principal_name => 'ADB_USER',
                        principal_type => xs_acl.ptype_db)
   );
END;
/

BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OPENAI_CRED',
    username        => 'OPENAI',
    password        => '<OPENAI_API_KEY>'
  );
END;
/
```

```
EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('Websearch');
BEGIN
  DBMS_CLOUD_AI_AGENT.create_tool(
    tool_name  => 'Websearch',
    attributes => '{"instruction": "This tool can be used for searching the
details about topics mentioned in notes and prepare a summary about prices,
details on web",
                   "tool_type": "WEBSEARCH",
                   "tool_params": {"credential_name": "OPENAI_CRED"}}'
  );
END;
/
```

Example: Notification Tool with Email Type
This example creates an Email notification tool. The Email tool enables agents to send notification emails as part of their workflow.

This example illustrates creating credentials `EMAIL_CRED` with your password, allowing SMTP access for the database user, and creating a notification tool with type `EMAIL`, including SMTP details, sender, and recipient addresses.

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'EMAIL_CRED',
    username => '<username>',
    password => '<password>');
END;
/
-- Allow SMTP access for user
BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
     host => 'smtp.email.us-ashburn-1.oci.oraclecloud.com',
     lower_port => 587,
     upper_port => 587,
     ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
                        principal_name => 'ADB_USER',
                        principal_type => xs_acl.ptype_db));
END;
/

EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('Email');
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'EMAIL',
    attributes => '{"tool_type": "NOTIFICATION",
                    "tool_params": {"notification_type" : "EMAIL",
                                    "credential_name": "EMAIL_CRED",
                                    "recipient":
"example_recipient@oracle.com",
                                    "smtp_host": "smtp.email.us-
ashburn-1.oci.oraclecloud.com",
                                    "sender": "example_sender@oracle.com"}}'
  );
```

```
END;
/
```

Example: Notification Tool with Slack Type
This example creates a Slack notification tool. The Slack tool enables agents to deliver notifications directly to a Slack workspace channel.

This example illustrates adding an ACL entry for Slack and creating a notification tool with type `SLACK` linking it to `SLACK_CRED` and the target channel.

```
BEGIN
 DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
  host         => 'slack.com',
  lower_port   => 443,
  upper_port   => 443,
  ace          => xs$ace_type(
      privilege_list => xs$name_list('http'),
      principal_name => 'ADB_USER',
      principal_type => xs_acl.ptype_db));
END;
/

BEGIN
  DBMS_CLOUD_AI_AGENT.create_tool(
    tool_name  => 'slack',
    attributes => '{"tool_type": "SLACK",
                    "tool_params": {"credential_name": "SLACK_CRED",
                                    "channel": "<channel_number>"}}'
  );
END;
/
```

Example: Create a Task

Create a task an agent can perform.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- #unique_189

> **Note**
>
> Only a DBA can grant `EXECUTE` privileges and network ACL procedure.

Example: Create an Email Task
This example creates `Generate_Email_Task` that instructs the LLM to produce a standard confirmation email using structured data.

The following example illustrates using the `instruction` attribute and providing instructions on what the email should include.

```
BEGIN DBMS_CLOUD_AI_AGENT.DROP_TASK('Generate_Email_Task');
EXCEPTION WHEN OTHERS THEN NULL; END;
/
BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'Generate_Email_Task',
    attributes => '{"instruction": "Use the customer information and product
details to generate an email in a professional format. The email should:' ||
                    '1. Include a greeting to the customer by name' ||
                    '2. Specify the item being returned, the order number,
and the reason for the return' ||
                    '3. If it is a refund, state the refund will be issued to
the credit card on record.' ||
                    '4. If it is a replacement, state that the replacement
will be shipped in 3-5 business days."}'
    );
END;
```

Example: Create a Fetch Log Task
This example creates a `FETCH_LOGS_TASK` that retrieves logs based on the request.

This example illustrates creating a task that uses the `log_fetcher` tool to retrieve logs. This custom tool specifies the PL/SQL procedure `fetch_logs`. The task instruction specifies what the task needs accomplish. The attribute `enable_human_tool` is set to true so a person can step in to guide or approve the task flow if needed. See Example: Fetch and Analyze Log Reports for a complete example.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name  =>'FETCH_LOGS_TASK',
    attributes =>'{
       "instruction": "Fetch the log entries from the database based on user
request: {query}",
       "tools": ["log_fetcher"],
       "enable_human_tool" : "true"
     }'
  );
END;
/
```

Example: Create a Log Analysis Task
This example creates a `ANALYZE_LOG_TASK` that analyzes the fetched logs.

The following example illustrates using `FETCH_LOGS_TASK` as the `input` attribute. The task named `ANALYZE_LOG_TASK` starts after logs have been gathered to have `FETCH_LOGS_TASK` output as an input. The task instruction specifies what the task needs accomplish. The attribute `enable_human_tool` is set to false indicating the absence of human-in-the-loop review.

```
BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
      task_name  =>'ANALYZE_LOG_TASK',
      attributes =>'{"instruction": "Analyze the fetched log entries
```

```
retrieved based on user request: {query} ' ||
                      'Generate a detailed report include issue analysis and
possible solution.",
         "input" : "FETCH_LOGS_TASK",
         "enable_human_tool" : "false"
       }'
    );
END;
/
```

Example: Create an Agent Team

Create agent teams to accomplish your tasks.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- #unique_190

Example: Create an Agent Team
This example creates the `ReturnAgency` team and includes a single agent `Customer_Return_Agent`. The task `Return_And_Price_Match` is assigned to the agent. This task manages return requests by asking for the reason and updating the order status in your database.. The `process` is set to `sequential` to run the tasks in a defined order.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TEAM(
    team_name  =>
'ReturnAgency',
    attributes => '{"agents": [{"name":"Customer_Return_Agent","task" :
"Return_And_Price_Match"}],
                   "process":
"sequential"}');

END;
/
```

See Example: Create a Product Return Agent for a complete example.

Example: Create Multi-Agent Team
This example creates the `Ops_Issues_Solution_Team` team and includes two agents to handle log analysis and troubleshooting: `Data_Engineer` and `Ops_Manager`. The `Data_Engineer` agent runs the task `fetch_logs_task` and `Ops_Manager` agent runs the task `analyze_log_task`. The `process` is set to `sequential` to run the tasks in a defined order.

```
BEGIN
      DBMS_CLOUD_AI_AGENT.create_team(
        team_name  => 'Ops_Issues_Solution_Team',
        attributes => '{"agents": [{"name":"Data_Engineer","task" :
"fetch_logs_task"},
                                   {"name":"Ops_Manager","task" :
"analyze_log_task"}],
```

```
                "process": "sequential"
                }');
END;
/
```

See Example: Fetch and Analyze Log Reports for a complete example.

Example: Create a Movie Analysis Agent with Built-In Tools

This example shows how you can create a movie analysis agent using Select AI Agent. In this example, you set up a movie analysis agent that retrieves data, answers questions, searches the web, and emails the analysis or sends Slack notifications.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- Example: Select AI with OCI Generative AI
- #unique_187
- #unique_188

The following example assumes that the data is available to you.

This example creates a `MOVIE_ANALYST` agent and uses multiple built-in tools such as `SQL`, `RAG`, `WEBSEARCH`, and `NOTIFICATION` tool of type: `EMAIL` and `SLACK`. The agent answers movie related questions using natural language prompts.

In this example, after your DBA grants `EXECUTE` privileges for: the `DBMS_CLOUD_AI_AGENT`, `DBMS_CLOUD_AI`, `DBMS_CLOUD_PIPELINE` packages, ACL access for your AI provider, SMTP access, and Slack access, you begin by creating the `MOVIE_ANALYST` agent.

Create an Agent

You create an agent called `MOVIE_ANALYST` with a profile (`GROK`) and role of answering questions about movies, actors, and genres.

Create Task

You create a task called `ANALYZE_MOVIE_TASK` with instructions to answer movie-related queries.

Create Team

You create a `MOVIE_AGENT_TEAM` team with `MOVIE_ANALYST` as the agent and the task as `ANALYZE_MOVIE_TASK` and set it as an active team.

You can later attach tools such as SQL, RAG, Websearch, or Notification to extend its capabilities.

Run the Select AI Agent Team

You now run the agent team by using `select ai agent` as a prefix to your prompts.

Create Tools

You then attach different built-in agent tools:

- `SQL`: Uses an NL2SQL profile to translate questions into SQL queries and other supported Select AI actions.
- `RAG`: Retrieve knowledge-based context from stored documents.
- `WEBSEARCH`: Gather movie details or prices online.
- `NOTIFICATION`:
  - `EMAIL`: Send answers to user prompts, movie reports, or reviews to a recipient.
  - `SLACK`: Send answers to user prompts, summaries, or updates directly to Slack.

The complete example is as follows:

```
--Grants EXECUTE privilege to ADB_USER
--
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Websearch tool accesses OPENAI endpoint, allow ACL access
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'api.openai.com',
    ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                        principal_name => 'ADB_USER',
                        principal_type => xs_acl.ptype_db)
   );
END;
/

-- To allow Email tool in Autonomous Database, allow SMTP access

BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
     host => 'smtp.email.us-ashburn-1.oci.oraclecloud.com',
     lower_port => 587,
     upper_port => 587,
     ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
                        principal_name => 'ADB_USER,
                        principal_type => xs_acl.ptype_db));
END;
/

-- Allow ACL access to Slack

BEGIN
 DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
  host         => 'slack.com',
  lower_port   => 443,
  upper_port   => 443,
  ace          => xs$ace_type(
      privilege_list => xs$name_list('http'),
      principal_name => 'ADB_USER',
      principal_type => xs_acl.ptype_db));
END;
/
```

```
PL/SQL procedure successfully completed.

--Create an agent
BEGIN
   DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
     agent_name => 'MOVIE_ANALYST',
     attributes => '{"profile_name": "GROK",
                     "role": "You are an AI Movie Analyst. Your can help
answer a variety of questions related to movies. "
             }'
   );
END;
/

PL/SQL procedure successfully completed.

BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
      task_name  => 'ANALYZE_MOVIE_TASK',
      attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}",
         "enable_human_tool" : "true"
       }'
    );
END;
/

PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TEAM(
    team_name  =>
'MOVIE_AGENT_TEAM',

    attributes => '{"agents": [{"name":"MOVIE_ANALYST","task" :
"ANALYZE_MOVIE_TASK"}],
                   "process": "sequential"
                   }');

END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');

PL/SQL procedure successfully completed.

select ai agent who are you?;

RESPONSE
--------------------------------------------------------------------------------
--
I'm MOVIE_ANALYST, an AI Movie Analyst here to assist with any questions or
topi
cs related to movies. Whether you need information on films, actors,
```

```
directors,
genres, or recommendations, I'm ready to help. What can I assist you with
regard
ing movies?

-- SQL TOOL
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name
=>'nl2sql_profile',

    attributes => '{"provider":
"oci",
      "credential_name": "GENAI_CRED",
      "oci_compartment_id" :
"ocid1.compartment.oc1..aaaaa...",
      "object_list": [{"owner": "ADB_USER", "name": "GENRE"},
                      {"owner": "ADB_USER", "name": "CUSTOMER"},
                      {"owner": "ADB_USER", "name": "WATCH_HISTORY"},
                      {"owner": "ADB_USER", "name": "STREAMS"},
                      {"owner": "ADB_USER", "name": "MOVIES"},
                      {"owner": "ADB_USER", "name": "ACTORS"}]
      }');
END;
/

PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'SQL',
    attributes => '{"tool_type": "SQL",
                    "tool_params": {"profile_name": "nl2sql_profile"}}'
  );
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.drop_task('ANALYZE_MOVIE_TASK');

PL/SQL procedure successfully completed.

BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
      task_name => 'ANALYZE_MOVIE_TASK',
      attributes =>'{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                      'You can use SQL tool to search the data from
database",
        "tools": ["SQL"],
         "enable_human_tool" : "true"
       }'
    );
END;
/
```

```
PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');

PL/SQL procedure successfully completed.

-- SQL tool retrieves the movie with the highest popularity view count from
the watch_history table
select ai agent what is the most popular movie?;

RESPONSE
--------------------------------------------------------------
The most popular movie is "Laugh Out Loud" released in 2008.

-- RAG TOOL
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
         profile_name => 'RAG_PROFILE',
         attributes =>'{"provider": "oci",
            "credential_name": "GENAI_CRED",
            "vector_index_name": "RAG_INDEX",
            "oci_compartment_id": "ocid1.compartment.oc1..aaaaa...",
            "temperature": 0.2,
            "max_tokens": 3000
          }');
END;
/

PL/SQL procedure successfully completed.

BEGIN
   DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
           index_name  => 'RAG_INDEX',
           attributes  => '{"vector_db_provider": "oracle",
                            "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
                            "object_storage_credential_name": "MY_OCI_CRED",
                            "profile_name": "RAG_PROFILE",
                            "vector_dimension": 1024,
                            "vector_distance_metric": "cosine",
                            "chunk_overlap":128,
                            "chunk_size":1024
   }');
END;
/
PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'RAG_TOOL',
    attributes => '{"tool_type": "RAG",
```

```
                      "tool_params": {"profile_name": "RAG_PROFILE"}}'
  );
END;
/

PL/SQL procedure successfully completed.

BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
      task_name =>'ANALYZE_MOVIE_TASK',
      attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                       'You can use RAG tool to search the information from
the knowledge base user give.",
        "tools": ["RAG_TOOL"],
         "enable_human_tool" : "true"
       }'
    );
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');

PL/SQL procedure successfully completed.

-- Rag seach the object store to find review or comments of Movie Laugh out
Loud
select ai agent Please find the comments of Movie Laugh out Loud;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------
The comments for the movie "Laugh Out Loud" (2008) are as follows:
1. A lighthearted comedy that delivers plenty of laughs, though the plot is
fairly predictable.
2. The performances are fun, especially from the lead actor who keeps the
energy high.
3. Some jokes feel a bit outdated, but overall it is an enjoyable watch for a
casual movie night.
4. Good chemistry between the cast members, which makes the humor more
natural.
5. Not a groundbreaking comedy, but it does what it promises makes you laugh
out loud.

-- WEBSEARCH TOOL
```

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OPENAI_CRED',
    username        => 'OPENAI',
    password        => '<API_KEY>'
  );
END;
/

PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'WEBSEARCH_TOOL',
    attributes => '{"instruction": "This tool can be used for searching the
details about topics mentioned in notes and prepare a summary about prices,
details on web",
                       "tool_type": "WEBSEARCH",
                       "tool_params": {"credential_name": "OPENAI_CRED"}}'
  );
END;
/

PL/SQL procedure successfully completed.

BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
      task_name => 'ANALYZE_MOVIE_TASK',
      attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                        'You can use WEBSEARCH_TOOL tool to search the
information from internet.",
        "tools": ["WEBSEARCH_TOOL"],
         "enable_human_tool" : "true"
       }'
    );
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');

PL/SQL procedure successfully completed.

select ai agent What is the most popular movie of 2023?;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------------------------
Determining the most popular movie of 2023 can depend on various metrics such
as box office earnings, streaming viewership, critical acclaim, or audience
reception. Based on global box office data, which is often a strong indicator
of popularity, the most successful movie of 2023 is "Barbie," directed by
Greta Gerwig. Released in July 2023, it grossed over $1.4 billion worldwide,
making it the highest-grossing film of the year and one of the biggest
cultural phenomena, often discussed alongside "Oppenheimer" due to the
"Barbenheimer" trend. Its widespread appeal, marketing, and social media buzz
further solidify its status as the most popular movie of 2023. However, if
you are looking for popularity based on a different metric (like streaming
numbers or awards), please let me know, and I can adjust the analysis
accordingly.

-- NOTIFICATION TOOL WITH EMAIL TYPE

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'EMAIL',
    attributes => q'[{"tool_type": "Notification",
                      "tool_params": {"notification_type" : "EMAIL",
                                      "credential_name": "EMAIL_CRED",
                                      "recipient": "example@oracle.com",
                                      "smtp_host": "smtp.email.us-
ashburn-1.oci.oraclecloud.com",
                                      "sender": "example@oracle.com"}}]'
  );
END;
/

PL/SQL procedure successfully completed.

BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
      task_name => 'ANALYZE_MOVIE_TASK',
      attributes =>'{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                    'You can use EMAIL TOOL tool to send email to the
user.",
        "tools": ["EMAIL"],
         "enable_human_tool" : "true"
       }'
    );
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;
```

```
PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');

PL/SQL procedure successfully completed.

select ai agent Please help me write a review of Movie "Barbie" and send the
review to my email;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------
I have written a review for the movie "Barbie" (2023) and sent it to your
email. Please check your inbox for the detailed review. If you have any
additional feedback or would like me to revise the review, let me know!

-- NOTIFICATION TOOL WITH SLACK TYPE

DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name  => 'SLACK_TOOL',
    attributes => '{"tool_type": "SLACK",
                    "tool_params": {"credential_name": "SLACK_CRED",
                                    "channel": "<channel_number>"}}'
  );
END;
/

PL/SQL procedure successfully completed.

BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
      task_name  => 'ANALYZE_MOVIE_TASK',
      attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                    'You can use SLACK TOOL to send the notification to
the user.",
        "tools": ["SLACK_TOOL"],
         "enable_human_tool" : "true"
       }'
    );
END;
/

PL/SQL procedure successfully completed.

select ai agent Please help me find the top 3 most-watched movies of 2023 and
send them to me on slack;

RESPONSE
--------------------------------------------------------------------------------
------------------------------------------------------
I have sent the list of the top 3 most-watched movies of 2023 to you via
Slack. Please check your Slack notifications for the details.
```

Example: Create a Product Return Agent

This example shows how you can create a multi-turn conversational agent using Select AI Agent. In this example, you set up a customer-service agent that handles product returns and updates the return status in your database.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent

This example creates an agent named *Customer_Return_Agent* and a tool named *Update_Order_Status_Tool* and then defines a task and a team to handle product return.

In this example, after your DBA grants `EXECUTE` privileges for: the `DBMS_CLOUD_AI_AGENT` and `DBMS_CLOUD_AI`, you begin by creating a sample data of customers, customer order status, and a function to update the customer order status. Then, create an agent named `Customer_Return_Agent`.

Create an Agent

You create an agent called `Customer_Return_Agent` with a profile (`OCI_GENAI_GROK`) and role to manage return requests.

Create Tools

You then create an agent tool named `Update_Order_Status_Tool` to update the status of the order in your database.

Create Task

You create a task called `Handle_Product_Return_Task` to guide the flow: ask for the reason (no longer needed, arrived too late, box broken, or defective). Proceed with a defective return flow.

Create Team

You create an agent team called `Return_Agency_Team` with `Customer_Return_Agent` as the agent and set it as an active team.

Run the Select AI Agent Team

You now run the agent team by using `select ai agent` as a prefix to your prompts.

The complete example is as follows:

```
--Grants EXECUTE privilege to ADB_USER
--
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name => 'Customer_Return_Agent',
    attributes => '{"profile_name": "OCI_GENAI_GROK",
                    "role": "You are an experienced customer return agent who
deals with customers return requests."}');
END;
```

```
/

PL/SQL procedure successfully completed.



--Sample customer data
CREATE TABLE CUSTOMERS (
    customer_id  NUMBER(10) PRIMARY KEY,
    name         VARCHAR2(100),
    email        VARCHAR2(100),
    phone        VARCHAR2(20),
    state        VARCHAR2(2),
    zip          VARCHAR2(10)
);

INSERT INTO CUSTOMERS (customer_id, name, email, phone, state, zip) VALUES
(1, 'Alice Thompson', 'alice.thompson@example.com', '555-1234', 'NY',
'10001'),
(2, 'Bob Martinez', 'bob.martinez@example.com', '555-2345', 'CA', '94105'),
(3, 'Carol Chen', 'carol.chen@example.com', '555-3456', 'TX', '73301'),
(4, 'David Johnson', 'david.johnson@example.com', '555-4567', 'IL', '60601'),
(5, 'Eva Green', 'eva.green@example.com', '555-5678', 'FL', '33101');

--create customer order status
CREATE TABLE CUSTOMER_ORDER_STATUS (
    customer_id     NUMBER(10),
    order_number    VARCHAR2(20),
    status          VARCHAR2(30),
    product_name    VARCHAR2(100)

);

INSERT INTO CUSTOMER_ORDER_STATUS (customer_id, order_number, status,
product_name) VALUES
(2, '7734', 'delivered', 'smartphone charging cord'),
(1, '4381', 'pending_delivery', 'smartphone protective case'),
(2, '7820', 'delivered', 'smartphone charging cord'),
(3, '1293', 'pending_return', 'smartphone stand (metal)'),
(4, '9842', 'returned', 'smartphone backup storage'),
(5, '5019', 'delivered', 'smartphone protective case'),
(2, '6674', 'pending_delivery', 'smartphone charging cord'),
(1, '3087', 'returned', 'smartphone stand (metal)'),
(3, '7635', 'pending_return', 'smartphone backup storage'),
(4, '3928', 'delivered', 'smartphone protective case'),
(5, '8421', 'pending_delivery', 'smartphone charging cord'),
(1, '2204', 'returned', 'smartphone stand (metal)'),
(2, '7031', 'pending_delivery', 'smartphone backup storage'),
(3, '1649', 'delivered', 'smartphone protective case'),
(4, '9732', 'pending_return', 'smartphone charging cord'),
(5, '4550', 'delivered', 'smartphone stand (metal)'),
(1, '6468', 'pending_delivery', 'smartphone backup storage'),
(2, '3910', 'returned', 'smartphone protective case'),
(3, '2187', 'delivered', 'smartphone charging cord'),
(4, '8023', 'pending_return', 'smartphone stand (metal)'),
(5, '5176', 'delivered', 'smartphone backup storage');
```

```
--Create a update customer order status function
CREATE OR REPLACE FUNCTION UPDATE_CUSTOMER_ORDER_STATUS (
    p_customer_name IN VARCHAR2,
    p_order_number  IN VARCHAR2,
    p_status        IN VARCHAR2
) RETURN CLOB IS
    v_customer_id  customers.customer_id%TYPE;
    v_row_count    NUMBER;
BEGIN
    -- Find customer_id from customer_name
    SELECT customer_id
    INTO v_customer_id
    FROM customers
    WHERE name = p_customer_name;

    UPDATE customer_order_status
    SET status = p_status
    WHERE customer_id = v_customer_id
      AND order_number = p_order_number;

    v_row_count := SQL%ROWCOUNT;

    IF v_row_count = 0 THEN
        RETURN 'No matching record found to update.';
    ELSE
        RETURN 'Update successful.';
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RETURN 'Error: ' || SQLERRM;
END;

--Create Tool
BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
        tool_name => 'Update_Order_Status_Tool',
        attributes => '{"instruction": "This tool updates the database to
reflect return status change. Always confirm user name and order number with
user before update status",
                       "function" : "update_customer_order_status"}',
        description => 'Tool for updating customer order status in database
table.'
    );
END;
/
--Create Task
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'Handle_Product_Return_Task',
    attributes => '{"instruction": "Process a product return request from a
customer:{query}' ||
                  '1. Ask customer the order reason for return (no longer
needed, arrived too late, box broken, or defective)' ||
                  '2. If no longer needed:' ||
                  '   a. Inform customer to ship the product at their
```

```
expense back to us.' ||
                    '   b. Update the order status to return_shipment_pending
using Update_Order_Status_Tool.' ||
                     '3. If it arrived too late:' ||
                    '   a. Ask customer if they want a refund.' ||
                    '   b. If the customer wants a refund, then confirm
refund processed and update the order status to refund_completed' ||
                    '4. If the product was defective or the box broken:' ||
                    '   a. Ask customer if they want a replacement or a
refund' ||
                    '   b. If a replacement, inform customer replacement is
on its way and they will receive a return shipping label for the defective
product, then update the order status to replaced' ||
                    '   c. If a refund, inform customer to print out the
return shipping label for the defective product, return the product, and
update the order status to refund' ||
                    '5. After the completion of a return or refund, ask if
you can help with anything else.' ||
                    '   End the task if user does not need help on anything
else",
                    "tools": ["Update_Order_Status_Tool"]}'
  );
END;
/
SQL> BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TEAM(
    team_name  =>
'Return_Agency_Team',

    attributes => '{"agents": [{"name":"Customer_Return_Agent","task" :
"Handle_Product_Return_Task"}],
                    "process":
"sequential"}');

END;
/

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('Return_Agency_Team');

PL/SQL procedure successfully completed.

--Validate that the agent team is set in the current session with get_team
SQL> SELECT DBMS_CLOUD_AI_AGENT.GET_TEAM from DUAL;

--Interact with the agent
SQL> select ai agent I want to return a smartphone case;

RESPONSE
Could you please tell me the reason for returning the smartphone case? Is it
no longer needed, arrived too late, box broken, or defective?

SQL> select ai agent the item is defective

RESPONSE
```

```
I'm sorry to hear that the smartphone case is defective. Would you like a
replacement or a refund?

SQL> select ai agent I will need a replacement

RESPONSE
Can you please confirm your name and order number so I can update the status
and process the replacement?

SQL> select ai agent I''m Bob Martinez and my order number is 7820

RESPONSE
Is there anything else I can help you with?

SQL> select ai agent No, I''m all set. Thanks

RESPONSE
Thank you for reaching out, Bob. I'm glad I could assist with the return of
your smartphone case and stand. If you have any further questions or need
assistance in the future, feel free to contact us. Have a great day!
```

Example: Use Select AI Agent with Database Actions and APEX
If you are using web SQL clients such as **Database Actions**, run the agent team by using the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function and supply your prompt within the function.

> **Note**
>
> Do not use `DBMS_CLOUD_AI_AGENT.SET_TEAM` in Database Actions or APEX Service. Instead, specify the agent team using the `team_name` argument of `DBMS_CLOUD_AI_AGENT.RUN_TEAM`.

The following example creates a conversation ID to track the customer conversation history and then uses the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function to interact with the product return agent.

See #unique_191 for more information.

```
CREATE OR REPLACE PACKAGE my_globals IS
  l_team_cov_id varchar2(4000);
END my_globals;
/
-- Create conversation
DECLARE
  l_team_cov_id varchar2(4000);
BEGIN
  l_team_cov_id := DBMS_CLOUD_AI.create_conversation();
  my_globals.l_team_cov_id := l_team_cov_id;
  DBMS_OUTPUT.PUT_LINE('Created conversation with ID: ' ||
my_globals.l_team_cov_id);
END;

--Interact with the agent
DECLARE
```

```
  v_response VARCHAR2(4000);
BEGIN
  v_response :=  DBMS_CLOUD_AI_AGENT.RUN_TEAM(
    team_name   => 'Return_Agency_Team',
    user_prompt => 'I want to return a smartphone case',
    params      => '{"conversation_id": "' || my_globals.l_team_cov_id || '"}'
  );
  DBMS_OUTPUT.PUT_LINE(v_response);
END;
```

Example: Fetch and Analyze Log Reports

This example shows how you can create an agent team using two agents-task pairs to retrieve logs and analyze them.

Before You Begin
Review:

- Perform Prerequisites for Select AI
  - Create an AI profile with Google as the provider. See Example: Select AI with Google.
  - Create an AI profile with OpenAI as the provider. See Example: Select AI with OpenAI.
- Prerequisites for Using Select AI Agent
- #unique_192

This example creates:

- a two-agent workflows for troubleshooting: *Data_Engineer* and *Ops_Manager*
- one tool: `log-fetcher`

and then defines tasks and team to fetch logs and analyze them.

> **Note**
>
> Only a DBA can run `EXECUTE` privileges and network ACL procedure.

In this example, after your DBA grants `EXECUTE` privileges for: the `DBMS_CLOUD_AI_AGENT` package, `DBMS_CLOUD_AI` package, and ACL access for your AI providers, you begin by creating the `Data_Engineer` and `Ops_Manager` agents.

Create an Agent

You create the agent called `Data_Engineer` with a profile (`GOOGLE`) that uses Google as the AI provider and role to retrieve and process complex data.

Create the agent called `Ops_Manager` with a profile (`OPENAI`) that uses OpenAI as the AI provider and role to analyze the data.

Create Tool

You then create an agent tool:`log_fetcher`: returns log entries after a given date. This uses a custom procedure `fetch_logs`.

Create Tasks

You define two tasks `fetch_logs_task` and `analyze_log_task` to guide the flow. The `fetch_logs_task` calls `log_fetcher` to retrieve logs based on the request. The `analyze_log_task` analyzes the fetched logs.

Create Team

You create `Ops_Issues_Solution_Team` team with `Data_Engineer` and `Ops_Manager` to run sequentially.

Run the Select AI Agent

You now set the active team and run the agent team with `select ai agent` as a prefix to your prompts.

The complete example is as follows:

```
--Grants EXECUTE privilege to ADB_USER
--
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

-- Grant Network ACL for OpenAI endpoint
BEGIN
     DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.openai.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
     );
END;
/
--Grant Network ACL for Google endpoint
BEGIN
      DBMS_NETWORK_ACL_ADB_USER.APPEND_HOST_ACE(
        host => 'generativelanguage.googleapis.com',
        ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                    principal_name => 'ADB_USER',
                    principal_type => xs_acl.ptype_db)
       );
END;
/
PL/SQL procedure successfully completed.

--Create a table with Database logs and insert sample data

SQL> CREATE TABLE app_logs (
  log_id         NUMBER GENERATED BY DEFAULT AS IDENTITY,
  log_timestamp  DATE,
  log_message    VARCHAR2(4000)
);

Table created.

SQL> INSERT INTO app_logs (log_timestamp, log_message) VALUES (
  TO_DATE('2025-03-22 03:15:45', 'YYYY-MM-DD HH24:MI:SS'),
  'INFO: Database Cluster: Failover completed successfully. Standby promoted
to primary. Downtime duration: 33 seconds. Correlation ID: dbfailover102.'
```

```
);

1 row created.

SQL> INSERT INTO app_logs (log_timestamp, log_message) VALUES (
  TO_DATE('2025-03-23 08:44:10', 'YYYY-MM-DD HH24:MI:SS'),
  'INFO: Switchover Process: Synchronization restored. Performing scheduled
switchover. Correlation ID: dbswitchover215.'
);

1 row created.

SQL> INSERT INTO app_logs (log_timestamp, log_message) VALUES (
  TO_DATE('2025-03-24 03:15:12', 'YYYY-MM-DD HH24:MI:SS'),
  'ERROR: Database Cluster: Primary database unreachable, initiating failover
to standby. Correlation ID: dbfailover102.'
);

1 row created.

SQL> COMMIT;

Commit complete.

-- create data engineer agent
SQL> BEGIN
      DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
        agent_name => 'Data_Engineer',
        attributes => '{"profile_name": "GOOGLE",
                        "role": "You are a specialized data ingestion
engineer with expertise in   ' ||
                         'retrieving and processing data from complex
database systems."
                }'
      );
    END;
    /
PL/SQL procedure successfully completed.

-- create ops manager agent
SQL> BEGIN
      DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
        agent_name => 'Ops_Manager',
        attributes => '{"profile_name": "OPENAI",
                        "role": "You are an experienced Ops manager who
excels at analyzing ' ||
                'complex log data, diagnosing the issues."
         }'
      );
    END;
    /


PL/SQL procedure successfully completed.

-- create log_fetcher tool
```

```
-- fetch_logs is a customized procedure.
-- Please make sure you have created your own procedure before use it in the
tool

--Create a customized fetch_logs procedure
SQL> CREATE OR REPLACE FUNCTION fetch_logs(since_date IN DATE) RETURN CLOB IS
  l_logs CLOB;
BEGIN
  SELECT JSON_ARRAYAGG(log_message RETURNING CLOB)
    INTO l_logs
    FROM app_logs
   WHERE log_timestamp >= since_date
   ORDER BY log_timestamp;

  RETURN l_logs;
EXCEPTION
  WHEN OTHERS THEN
    RETURN 'Error fetching logs: ' || SQLERRM;
END fetch_logs;
/

Function created.

SQL> BEGIN
      DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
        tool_name => 'log_fetcher',
        attributes => '{"instruction": "retrieves and returns all log
messages with a LOG_TIMESTAMP greater than or equal to the input date.",
                       "function": "fetch_logs"}'
      );
    END;
    /

PL/SQL procedure successfully completed.

-- create task with log fetcher tool

SQL> BEGIN
      DBMS_CLOUD_AI_AGENT.CREATE_TASK(
        task_name  => 'fetch_logs_task',
        attributes => '{"instruction": "Fetch the log entries from the
database based on user request: {query}}.",
                       "tools": ["log_fetcher"]}'
      );
    END;
    /

PL/SQL procedure successfully completed.

-- create task with predefined rag and slack tool

SQL> BEGIN
      DBMS_CLOUD_AI_AGENT.create_task(
        task_name => 'analyze_log_task',
        attributes => '{"instruction": "Analyze the fetched log entries
retrieved based on user request: {query} ' ||
```

```
                      'Generate a detailed report include issue analysis and
possible solution.",
           "input" : "fetch_logs_task"
        }'
     );
    END;
    /

PL/SQL procedure successfully completed.

-- create team and set agent team
SQL> BEGIN
      DBMS_CLOUD_AI_AGENT.create_team(
        team_name  => 'Ops_Issues_Solution_Team',
        attributes => '{"agents": [{"name":"Data_Engineer","task" :
"fetch_logs_task"},
                                    {"name":"Ops_Manager","task" :
"analyze_log_task"}],
                "process": "sequential"
                }');
    END;
    /

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('Ops_Issues_Solution_Team');


PL/SQL procedure successfully completed.


SQL> select ai agent fetch and analyze the logs after 03/15 2025;

RESPONSE
--------------------------------------------------------------------------------
--------------------------------------------------------------------
1. Issue: High volume of 500 Internal Server Errors between 03/22 and 03/24.
   Solution: Review server application logs to identify failing components;
add better exception handling and fallback mechanisms to prevent service
crashes.

2. Issue: Increased response time on /api/v1/user and /checkout, peaking on
03/25.
   Solution: Profile backend queries and services, apply caching where
applicable, and offload static content to a CDN.

3. Issue: Detected brute-force login attack with over 500 failed POST
attempts from a single IP.
   Solution: Add rate-limiting, temporary IP bans, and CAPTCHA on the /login
endpoint to prevent credential stuffing.

4. Issue: Suspicious User-Agent headers such as curl/7.58.0 or empty headers
mimicking mobile devices.
   Solution: Block malformed or uncommon User-Agent strings and log them for
threat intelligence analysis.
```

```
5. Issue: DDoS-like traffic spike from distributed IPs observed on 03/20.
   Solution: Enable DDoS protection at the CDN or cloud provider level, and
configure autoscaling to absorb burst traffic.
```

Example: Create a Customized HTTP Tool

This example defines a PL/SQL function that sends an `HTTP GET` request and returns the response text. The function is then added in the Select AI Agent tool.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- Example: Create Built-In Tools

You can create a custom tool that sends a REST request using the `DBMS_CLOUD.SEND_REQUEST` procedure and returns the response as text. This example shows how to write the PL/SQL function and add it as a tool for use in agent tasks.

The function `get_url_content` sends a GET request to the target URL and returns the response as a `CLOB`. The `DBMS_CLOUD_AI_AGENT.CREATE_TOOL` uses the `get_url_content` function and creates a tool named `HTTP_TOOL` that Select AI agent can call when a task requires fetching content from a URL.

```
CREATE OR REPLACE FUNCTION get_url_content (
    url IN CLOB
) RETURN CLOB
AS
    l_resp   DBMS_CLOUD_TYPES.RESP;
    l_result CLOB;
BEGIN
    l_resp := DBMS_CLOUD.SEND_REQUEST(
                credential_name => NULL,
                method          => 'GET',
                uri             => url
              );
    l_result :=
DBMS_VECTOR_CHAIN.UTL_TO_TEXT(DBMS_CLOUD.GET_RESPONSE_RAW(l_resp);
    RETURN l_result;
END get_url_content;
/

BEGIN
  DBMS_CLOUD_AI_AGENT.create_tool(
    tool_name  => 'HTTP_TOOL',
    attributes => '{
      "instruction": "This tool fetches and returns the plain text content
from the specified URL. ",
      "function": "get_url_content"
    }'
  );
END;
/
```

Example: View Agent Prompts and Responses from the Latest Team Run

This example helps debug recent Select AI Agent activity by displaying prompts, responses, and agent thoughts from the most recent agent team run using historical views and conversation data.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- DBMS_CLOUD_AI_AGENT History Views

This example retrieves and displays detailed logs from the most recent Select AI Agent team run. The example uses `USER_AI_AGENT_TEAM_HISTORY`, `USER_AI_AGENT_TASK_HISTORY`, and `USER_CLOUD_AI_CONVERSATION_PROMPTS` views to show each agent's team name, task, associated prompts, and responses. Use this query to troubleshoot issues by reviewing both the input prompt and the generated responses from the latest team run.

```
WITH latest_team AS (
  SELECT team_exec_id, team_name, start_date
  FROM user_ai_agent_team_history
  ORDER BY start_date DESC
  FETCH FIRST 1 ROW ONLY
),
latest_task AS (
    SELECT team_exec_id, task_name, agent_name, conversation_params,
start_date,
           ROW_NUMBER() OVER (PARTITION BY team_exec_id, task_name,
agent_name
                              ORDER BY start_date DESC) as rn
  FROM user_ai_agent_task_history
)
SELECT
  team.team_name,
  task.task_name,
  task.agent_name,
  p.prompt,
  p.prompt_response
FROM latest_team team
JOIN latest_task task
  ON team.team_exec_id = task.team_exec_id
 AND task.rn = 1
LEFT JOIN user_cloud_ai_conversation_prompts p
  ON p.conversation_id = JSON_VALUE(task.conversation_params,
'$.conversation_id')
ORDER BY task.start_date DESC NULLS LAST,
         p.created     DESC NULLS LAST;
```

```
USER_AI_AGENT_TOOL_HISTORYUSER_AI_AGENT_TASK_HISTORYUSER_AI_AGENT_TASK_HISTORY

--View the tool history

select * from USER_AI_AGENT_TOOL_HISTORY
order by START_DATE desc

--View the task history

select * from USER_AI_AGENT_TASK_HISTORY
order by START_DATE desc

--View the team history
select * from USER_AI_AGENT_TEAM_HISTORY
order by START_DATE desc
```

Example: Resume an Agent Team Run from WAITING_FOR_HUMAN State

This example shows how you can resume an agent team run when the task history status is `WAITING_FOR_HUMAN`. When an agent team enters the `WAITING_FOR_HUMAN` state, it pauses until a user provides the next input or confirmation. You can resume the run depending on how you use the agent team: on SQL command line or through `DBMS_CLOUD_AI_AGENT.RUN_TEAM`.

Before You Begin
Review:

- Perform Prerequisites for Select AI
- Prerequisites for Using Select AI Agent
- #unique_193

Example: Resume an Agent Team Run Using SQL Command Line
If you are using `SELECT AI AGENT <prompt>` on the SQL command line, the Select AI Agent automatically resumes the paused team run. The conversation context, including prior prompts and reasoning, is maintained without requiring additional parameters.

Example: Resume an Agent Team Run Using DBMS_CLOUD_AI_AGENT.RUN_TEAM
If you are using the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` procedure, you can resume a paused agent team by passing the same `conversation_id` that was used in the initial run. The agent team run state, including pending actions, and context, is preserved and the agent team continues exactly from where it stopped.

See Example: Create a Product Return Agent for an example using the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function.

```
--Interact with the agent
DECLARE
  v_response VARCHAR2(4000);
BEGIN
  v_response :=  DBMS_CLOUD_AI_AGENT.RUN_TEAM(
    team_name   => '<same initial team name>',
    user_prompt => 'response to request',
    params      => '{"conversation_id": "<same initial conversation_id>"}'
  );
```

```
  DBMS_OUTPUT.PUT_LINE(v_response);
END;
```

Part III
Reference

27

Select AI PL/SQL Package

The `DBMS_CLOUD_AI` package is available to support Select AI.

Subprogram	Description
CREATE_PROFILE Procedure	This procedure creates a new AI profile for translating natural language prompts to SQL statements.
Profile Attributes	Provides AI profile attributes that you can configure.
CLEAR_PROFILE Procedure	This procedure clears an active AI profile in the current session.
DISABLE_PROFILE Procedure	This procedure disables an AI profile in the current database.
DROP_PROFILE Procedure	This procedure drops an existing AI profile.
ENABLE_PROFILE	This procedure enables an AI profile to use in the current database.
GENERATE Function	This function enables you to use Select AI in a stateless manner with your existing profile.
GENERATE_SYNTHETIC_DATA Function	This function generates synthetic data.
GET_PROFILE Function	This function returns the profile name used in the current session.
SET_ATTRIBUTE Procedure	This procedure sets AI profile attributes.
SET_ATTRIBUTES Procedure	This procedure enables you to set one or more AI profile attributes by accepting attribute name and value pairs in JSON format.
SET_PROFILE Procedure	This procedure sets AI profile for the current database.
ENABLE_DATA_ACCESS Procedure	Use this procedure to enable sending data to your LLM.
DISABLE_DATA_ACCESS Procedure	Use this procedure to disable sending data to your LLM.
FEEDBACK Procedure	Use this procedure to potentially improve query generation accuracy by providing a feedback to Select AI.
Vector Index FEEDBACK	This is a default vector index created when you first use `feedback`.
CREATE_CONVERSATION Procedure	This procedure helps you to create a conversation.
CREATE_CONVERSATION Function	This function helps you to create a conversation and use the conversation ID in other procedures.
CREATE_CONVERSATION Attributes	Use the conversation attributes to customize your conversations.
UPDATE_CONVERSATION Procedure	This procedure updates an existing procedure with user-specified parameters.
SET_CONVERSATION_ID Procedure	This procedure sets conversation support in the current session.
GET_CONVERSATION_ID Function	This procedure helps you to get the `conversation_id` parameter.

Subprogram	Description
CLEAR_CONVERSATION_ID Function	This procedure helps you to clear any `conversation_id` set in the current session.
DELETE_CONVERSATION_PROMPT Procedure	This procedure deletes a particular prompt.
DROP_CONVERSATION Procedure	This procedure deletes an entire conversation and its metadata.
SUMMARIZE Function	This function summarizes your content based on the parameters.
SUMMARIZE Parameters	Use the summarize attributes to customize summary generation.
CREATE_VECTOR_INDEX Procedure	This procedure creates a vector index in the specified vector database, and populates it with data from an object store using an asynchronous scheduler job.
DROP_VECTOR_INDEX Procedure	This procedure removes a vector store index. It normally removes the vector store index object and deletes the vector database.
DISABLE_VECTOR_INDEX Procedure	This procedure disables a vector index object in the current database. When disabled, an AI profile cannot use the vector index, and the system does not load data into the vector store.
ENABLE_VECTOR_INDEX Procedure	This procedure enables or activates a previously disabled vector index object.
UPDATE_VECTOR_INDEX Procedure	This procedure updates an existing vector store index with a specified value of the vector index attribute.
Vector Index Attributes	Provides vector index profile attributes that you can configure.

28

DBMS_CLOUD_AI Views

The `DBMS_CLOUD_AI` package uses the following views.

View Name	Description
DBA_CLOUD_AI_PROFILES	Displays all AI profiles created in the current database so you can review and manage them.
USER_CLOUD_AI_PROFILES	Displays AI profiles created within your own schema so you can review and manage them.
DBA_CLOUD_AI_PROFILE_ATTRIBUTES	Displays configuration attributes for all AI profiles in the database to show how each profile behaves.
USER_CLOUD_AI_PROFILE_ATTRIBUTES	Displays configuration attributes for AI profiles created in your schema.
DBA_CLOUD_VECTOR_INDEXES	Lists all vector indexes created in the current database and shows their details.
USER_CLOUD_VECTOR_INDEXES	Lists vector indexes that you created in your schema.
DBA_CLOUD_VECTOR_INDEX_ATTRIBUTES	Displays attributes and configuration details for all vector indexes in the database.
USER_CLOUD_VECTOR_INDEX_ATTRIBUTES	Displays attributes for vector indexes created within your schema.
USER_CLOUD_AI_CONVERSATIONS	Shows details of AI conversations that occurred within your schema.
USER_CLOUD_AI_CONVERSATION_PROMPTS	Shows prompts and related details for each conversation in your schema.
AI_TRANSLATION_LANGUAGES	Displays available translation languages that Select AI Translation can use.

29

Select AI Agent PL/SQL Package

The `DBMS_CLOUD_AI_AGENT` package is available to support Select AI Agent.

Subprogram	Description
CREATE_AGENT Procedure	This procedure creates an agent.
CREATE_AGENT Attributes	Provides attributes for creating an agent.
ENABLE_AGENT Procedure	This procedure enables an agent.
DISABLE_AGENT Procedure	This procedure disables an agent.
DROP_AGENT Procedure	This procedure drops an existing agent.
CREATE_TASK Procedure	This procedure creates a task that an agent and agent team can include.
CREATE_TASK Attributes	Provides attributes for creating an agent task.
ENABLE_TASK Procedure	This procedure enables an agent task.
DISABLE_TASK Procedure	This procedure disables an agent task.
DROP_TASK Procedure	This procedure drops an existing agent task.
CREATE_TOOL Procedure	This procedure creates custom tools that an agent can include.
CREATE_TOOL Attributes	Provides attributes for creating tools.
ENABLE_TOOL Procedure	This procedure enables a registered tool.
DISABLE_TOOL Procedure	This procedure disables a registered tool.
DROP_TOOL Procedure	This procedure drops an existing tool.
CREATE_TEAM Procedure	This procedure creates an agent team and includes agents and tasks.
CREATE_TEAM Attributes	Provides attributes for creating an agent team.
SET_TEAM Function	This procedure sets the agent team in the current session.
GET_TEAM Function	This procedure returns the agent team set in the current session.
CLEAR_TEAM Procedure	This procedure clears the agent team set in the current session.
RUN_TEAM Function	This procedure creates a new agent or runs a paused one.
ENABLE_TEAM Procedure	This procedure enables an agent team.
DISABLE_TEAM Procedure	This procedure disables an agent team.
DROP_TEAM Procedure	This procedure drops an existing agent team.

30

DBMS_CLOUD_AI_AGENT Views

The `DBMS_CLOUD_AI_AGENT` package uses the following views.

View Name	Description
DBA_AI_AGENTS	Displays all agents created by users in the current database.
DBA_AI_AGENTS_ATTRIBUTES	Displays attributes for all agents created by users in the database.
USER_AI_AGENTS	Displays details of agents created in your schema.
USER_AI_AGENTS_ATTRIBUTES	Displays attributes of agents created by the current user in their schema.
DBA_AI_AGENT_TOOLS	Displays all tools created by users in the database.
DBA_AI_AGENT_TOOL_ATTRIBUTES	Displays attributes for all tools created by users in the database.
USER_AI_AGENT_TOOLS	Displays details of tools created by the current user in their schema.
USER_AI_AGENT_TOOL_ATTRIBUTES	Displays attributes of tools created by the current user in their schema.
DBA_AI_AGENT_TASKS	Displays all tasks created by users in the current database.
DBA_AI_AGENT_TASK_ATTRIBUTES	Displays attributes for all tasks created by users in the database.
USER_AI_AGENT_TASKS	Displays details of tasks created by the current user in their schema.
USER_AI_AGENT_TASK_ATTRIBUTES	Displays attributes of tasks created by the current user in their schema.
DBA_AI_AGENT_TEAMS	Displays all agent teams created by users in the database.
DBA_AI_AGENT_TEAM_ATTRIBUTES	Displays attributes for all agent teams created by users in the database.
USER_AI_AGENT_TEAMS	Displays details of agent teams created by the current user in their schema.
USER_AI_AGENT_TEAM_ATTRIBUTES	Displays attributes of agent teams created by the current user in their schema.
DBMS_CLOUD_AI_AGENT History Views	Groups the agent history views into team, task, and tool history categories for monitoring and analysis.

View Name	Description
DBA_AI_AGENT_TEAM_HISTORY	Displays all agent team runs across the system, including team names, run states, timestamps, and errors.
USER_AI_AGENT_TEAM_HISTORY	Displays all agent team runs for teams owned by the current user, including run states and execution details.
DBA_AI_AGENT_TASK_HISTORY	Displays task-level parameters and results within agent teams across the database for all users.

View Name	Description
USER_AI_AGENT_TASK_HISTORY	Displays task parameters, inputs, and results for agent teams owned by the current user.
DBA_AI_AGENT_TOOL_HISTORY	Displays all tool calls across the database, including inputs, outputs, and diagnostic messages.
USER_AI_AGENT_TOOL_HISTORY	Displays tool call details, inputs, and outputs for tools owned by the current user.