ORACLE

Business / Technical Brief

Vector Map Layers in Oracle Analytics

Provides guidance on creating and using vector map layers in Oracle Analytics.

January 2026, version 1.0

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and shouldn't be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Revision History

The following revisions have been made to this document since its initial publication.

DATE	REVISION
January 2026	Initial publication

Author: Gautam Pisharam

Contributing Authors: Philippe Lions, Jon ODonnell, Anna Moat

Table of Contents

Introduction

Modern analytics often requires displaying very large geographic datasets such as postal codes, districts, parcels, and grids that can contain thousands of detailed geometries. Rendering full geometries can be resource-intensive because every shape must be downloaded and processed before being rendered on a map.

Vector tiles address this challenge by providing a compact, web-optimized format that delivers only the portion of the map needed at each zoom level. Vector tiles break the map into small, compressed units that are streamed only when needed. Only the tiles that are visible at a particular zoom level or location are requested. This significantly reduces network load, speeds up rendering, and ensures smooth zooming and panning.

When combined with Oracle REST Data Services, Oracle AI Database exposes vector tiles as standard REST endpoints that Oracle Analytics can then render in maps. Oracle Analytics retrieves and streams the vector tiles as the user zooms and navigates the map. This results in fast rendering, responsive workbooks, and consistent performance for very large spatial datasets.

This document explains the architecture and workflow for implementing vector tile layers in Oracle Analytics. It covers how tiles are generated in the database and how Oracle Analytics uses these elements to support interactive mapping.

Architecture

Oracle AI Database, together with Oracle REST Data Services, provides the spatial processing capabilities needed to generate and publish vector tiles. Oracle Analytics consumes these services to deliver interactive map visualization and analytics. Understanding this division is essential before proceeding with implementation.

Overview

The workflow for delivering vector tile layers to Oracle Analytics consists of the following steps:

1. Prepare spatial data.
2. Publish vector tiles.
3. Publish bounding box content.
4. Enable tile caching.
5. Register the layer in Oracle Analytics.
6. Visualize the layer in Oracle Analytics.

Prepare Spatial Data

Store your spatial dataset in Oracle AI Database using SDO_GEOMETRY objects for your geometries with the correct SRID. Ensure that your geometries are valid and appropriate for tiling and metadata generation. This spatial data forms the foundation for generating both vector tiles and bounding box metadata.

Publish Vector Tiles

Create an Oracle REST Data Services module, template, and handler that exposes a standard tile URL pattern such as:

```
/{module}/vt/{z}/{x}/{y}.pbf
```

The handler invokes SDO_UTIL.GET_VECTORTILE, which produces vector tiles directly from the spatial geometries.

Publishing this endpoint allows Oracle Analytics to fetch tiles dynamically as users interact with the map through zooming, filtering, or panning.

Publish Bounding Box Content

Generate a GeoJSON document that contains, for each feature:

- A minimum bounding rectangle
- An interior point or centroid (CX, CY)
- A unique identifier such as GEOID, ZIP code, or region ID

Store this metadata in a table and expose it through an Oracle REST Data Services REST endpoint.

Oracle Analytics retrieves this metadata during layer registration and uses it to determine feature extents, control zoom behavior, support filtering actions, and place labels accurately.

Enable Tile Caching

Enable caching using SDO_UTIL.ENABLE_VECTORTILE_CACHE for the table and geometry column supporting the tile service.

Caching is an essential part of a production deployment because it:

- Avoids regenerating tiles repeatedly
- Improves dashboard responsiveness
- Reduces database compute overhead
- Ensures stable performance as user load increases

Tiles are stored automatically in the cache as they're generated and are reused across different sessions and dashboards.

Register the Layer in Oracle Analytics

After the tile and metadata endpoints are available, register the vector tile layer in Oracle Analytics.

During registration, Oracle Analytics validates and stores the following:

- The vector tile endpoint URL
- The bounding box endpoint URL

The registered layer becomes part of the shared map assets available to the environment and can be used in any map visualization that references the same key attribute.

Visualize the Layer in Oracle Analytics

Once registered, the vector tile layer can be used directly with map visualizations.

When the user adds the relevant key attribute to a map, Oracle Analytics:

- Fetches vector tiles from the published Oracle REST Data Services endpoint
- Uses metadata to determine zoom levels, feature boundaries, and label positions
- Renders the map efficiently with smooth interaction and high performance

Step-by-Step Example: US Zip Codes

Step 1: Prepare Spatial Data

This example uses a sample dataset derived from the U.S. Census Bureau Shapefile for ZIP Code Tabulation Areas (ZCTAs). This dataset provides polygonal boundaries for ZIP Codes across the United States.

After you import the data into Oracle AI Database, it's stored as a spatial table named USZIPCODES, which forms the foundation for vector tile generation and bounding box metadata creation. Before using the polygon geometries in this table for vector tile generation, you should validate them and, where necessary, rectify them to ensure geometric correctness. Additionally, you should spatially index the geometry column. These steps help ensure reliable tile generation and consistent map rendering at scale.

Dataset Overview — USZIPCODES

The USZIPCODES table contains one polygon record per ZIP Code boundary.

Here's a simplified representation of key fields:

Column Name	Description
GEOID20	The ZCTA (ZIP Code) identifier from the dataset. This serves as the primary key for mapping in Oracle Analytics.
GEOM	Polygon geometry in SDO_GEOMETRY format representing the ZIP Code boundary.
STATE	Optional attribute indicating the state (derived from the ZCTA metadata).
COUNTY	Optional attribute indicating the county.
ALAND20, AWATER20	Census-provided land and water area measurements (optional).

Only GEOID20 and GEOM are required for vector tile operations.

The GEOID20 field provides the feature key used to bind Oracle Analytics datasets to spatial features.

The GEOM column is used to generate:

- Vector tiles (PBF format)
- Bounding box + centroid metadata (GeoJSON format)

Step 2: Publish Vector Tiles

Vector tiles are not precomputed artifacts, they're generated at the moment of request. Oracle REST Data Services provides the HTTP interface through which Oracle Analytics obtains these tiles. Oracle REST Data Services acts as a thin service layer above the SDO_UTIL.GET_VECTORTILE spatial function, translating tile requests (/z/x/y) into spatial tile generation calls.

A vector tile request adheres to the pattern: `/{module}/vt/{z}/{x}/{y}.pbf`

The parameters correspond to the tile coordinate system used by the Google-style Web Mercator tiling scheme. When Oracle Analytics requests a tile for a given zoom (z) and x,y coordinates, Oracle REST Data Services passes these values to the vector tile function, which returns a binary PBF-encoded representation of the geometries that intersect that tile.

Define the Oracle REST Data Services Module

The module establishes the base URL for the vector tile service.

For example:

```
BEGIN
  ORDS.DEFINE_MODULE(
    P_MODULE_NAME => 'uszipcodespbf',
    P_BASE_PATH  => '/uszipcodespbf/',
    P_ITEMS_PER_PAGE => 25,
    P_STATUS     => 'PUBLISHED',
    P_COMMENTS   => 'Vector tile service for USZIPCODES'
  );
END;
/
```

Register the Tile URL Structure

Vector tile clients, including Oracle Analytics, expect a URL pattern that conveys zoom, X, and Y parameters.

For example:

```
BEGIN
  ORDS.DEFINE_TEMPLATE(
    P_MODULE_NAME => 'uszipcodespbf',
    P_PATTERN     => 'vt/:z/:x/:y',
    P_PRIORITY    => 0,
    P_ETAG_TYPE   => 'HASH'
  );
END;
/
```

Implement the Vector Tile Handler

The vector tile handler links incoming HTTP requests to SDO_UTIL.GET_VECTORTILE, which generates tiles in PBF format.

For example:

```
BEGIN
  ORDS.DEFINE_HANDLER(
    P_MODULE_NAME => 'uszipcodespbf',
    P_PATTERN     => 'vt/:z/:x/:y',
    P_METHOD      => 'GET',
    P_SOURCE_TYPE => ORDS.SOURCE_TYPE_MEDIA,
    P_SOURCE      => q'[
      SELECT 'application/vnd.mapbox-vector-tile' AS mediatype,
             SDO_UTIL.GET_VECTORTILE(
               TABLE_NAME    => 'USZIPCODES',
               GEOM_COL_NAME => 'GEOM',
               ATT_COL_NAMES => SDO_STRING_ARRAY('GEOID20'),
               TILE_X        => :x,
               TILE_Y_PBF    => :y,
               TILE_ZOOM     => :z
             ) AS vtile
      FROM dual
    ]'
  );
END;
/
```

Each tile request triggers on-demand spatial computation. The result is a lightweight, highly compressed representation optimized for map rendering.

Step 3: Publish Bounding Box Content

In addition to vector tiles, Oracle Analytics requires a lightweight metadata layer describing each feature's bounding box, centroid, and key identifier. This metadata is essential for analytic behaviors such as:

- Automatic focus and zoom
- Label placement
- Feature selection and highlighting
- Feature-to-dataset binding

Since vector tiles are optimized for rendering efficiency, they don't contain sufficient information for these higher-level interactions. The metadata layer steps in here.

A common approach is to compute, for each geometry:

- The minimum bounding rectangle (MBR) using SDO_GEOM_MBR
- An interior label point using SDO_UTIL.INTERIOR_POINT
- The feature key used to associate dataset attributes with map features

These elements are compiled into a GeoJSON FeatureCollection and stored in a lightweight cache table. Oracle REST Data Services then publishes this as a simple REST endpoint that Oracle Analytics accesses during layer registration.

Create a Metadata Cache Table

For example:

```
CREATE TABLE USZIPCODES_GEOJSON_CACHE (
  ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  GEOJSON_CONTENT CLOB,
  CREATED_AT TIMESTAMP DEFAULT SYSTIMESTAMP
);
```

This SQL creates the cache table to hold the bounding box metadata content.

Compute Metadata and Assemble GeoJSON

The following sample PL/SQL block computes bounding rectangles and interior points for each geometry, assembling a complete GeoJSON FeatureCollection:

```
DECLARE
  l_geojson CLOB := '{"type":"FeatureCollection","features":[';
  first BOOLEAN := TRUE;
BEGIN
  FOR r IN (
    SELECT GEOID20,
           SDO_UTIL.TO_GEOJSON(SDO_GEOM_MBR(GEOM)) AS geometry,
           (SDO_UTIL.INTERIOR_POINT(GEOM)).SDO_POINT.X AS CX,
           (SDO_UTIL.INTERIOR_POINT(GEOM)).SDO_POINT.Y AS CY
    FROM USZIPCODES
  )
  LOOP
    IF NOT first THEN
      l_geojson := l_geojson || ',';
    END IF;

    first := FALSE;

    l_geojson := l_geojson ||
      '{"type":"Feature","geometry":' || r.geometry ||
      ',"properties":{"GEOID20":"' || r.GEOID20 ||
      '","CX":' || r.CX ||
      ',"CY":' || r.CY ||
      '}}';
  END LOOP;

  l_geojson := l_geojson || ']}';

  INSERT INTO USZIPCODES_GEOJSON_CACHE (GEOJSON_CONTENT)
```

```
  VALUES (l_geojson);
END;
/
```

This metadata becomes the reference layer Oracle Analytics consults during registration.

Publish Metadata Through Oracle REST Data Services

For example:

```
BEGIN
  ORDS.DEFINE_MODULE(
    P_MODULE_NAME => 'uszipcodes_meta',
    P_BASE_PATH   => '/uszipcodes_meta/',
    P_STATUS      => 'PUBLISHED'
  );
END;
/
```

```
BEGIN
  ORDS.DEFINE_TEMPLATE(
    P_MODULE_NAME => 'uszipcodes_meta',
    P_PATTERN     => 'geojson',
    P_ETAG_TYPE   => 'HASH'
  );
END;
/
```

```
BEGIN
  ORDS.DEFINE_HANDLER(
    P_MODULE_NAME => 'uszipcodes_meta',
    P_PATTERN     => 'geojson',
    P_METHOD      => 'GET',
    P_SOURCE_TYPE => ORDS.SOURCE_TYPE_MEDIA,
    P_SOURCE      => q'[
      SELECT 'application/geo+json' AS media_type,
             GEOJSON_CONTENT
      FROM (
        SELECT GEOJSON_CONTENT
        FROM USZIPCODES_GEOJSON_CACHE
        ORDER BY CREATED_AT DESC
      )
      WHERE ROWNUM = 1
    ]'
  );
END;
/
```

This metadata endpoint acts as the descriptive layer that enables Oracle Analytics to understand how to position the tiles and interpret features. The GeoJSON Metadata content is downloaded in real time by Oracle Analytics every session when the vector layer is used by a map visualization.

Step 4: Enable Tile Caching

Although vector tiles can be generated dynamically, many tiles are requested repeatedly, particularly in workbooks where users explore the same regions. Oracle AI Database includes a built-in caching mechanism that stores previously generated tiles and retrieves them instantly on subsequent requests.

Enabling caching significantly reduces generation time and improves workbook responsiveness. Once activated, the cache automatically manages tile lifecycle and consistency during updates to the underlying data.

For example:

```
EXEC SDO_UTIL.ENABLE_VECTORTILE_CACHE('USZIPCODES', 'GEOM');
```

This caching layer allows the database to function as an efficient tile service even under heavy analytical workloads. Administrators can also purge, disable, or adjust zoom-level constraints for the cache.

Step 5: Register the layer in Oracle Analytics

Once the vector tile endpoint and metadata endpoint are available, Oracle Analytics can incorporate them as a reusable map layer. During registration, Oracle Analytics fetches the metadata to understand feature extents, labeling points, and the primary key used to bind dataset attributes to spatial features.

The registration process results in a new entry within your Oracle Analytics Map Layers catalog. From this point onward, the layer becomes available across datasets and map visualizations.

This step transitions the solution from backend preparation to frontend analytical use, completing the integration between Oracle AI Database, Oracle REST Data Services, and Oracle Analytics.

Go to Data Layers in Oracle Analytics

This is where custom vector tile services are configured. Only this area allows the addition of new tile layers and metadata definitions.

1. Navigate to your Oracle Analytics home page.
2. Click **Navigator**, then click **Console**.
3. Click **Maps**.
4. Open the **Data Layers** tab, which lists all GeoJSON and vector-based layers available to the system.

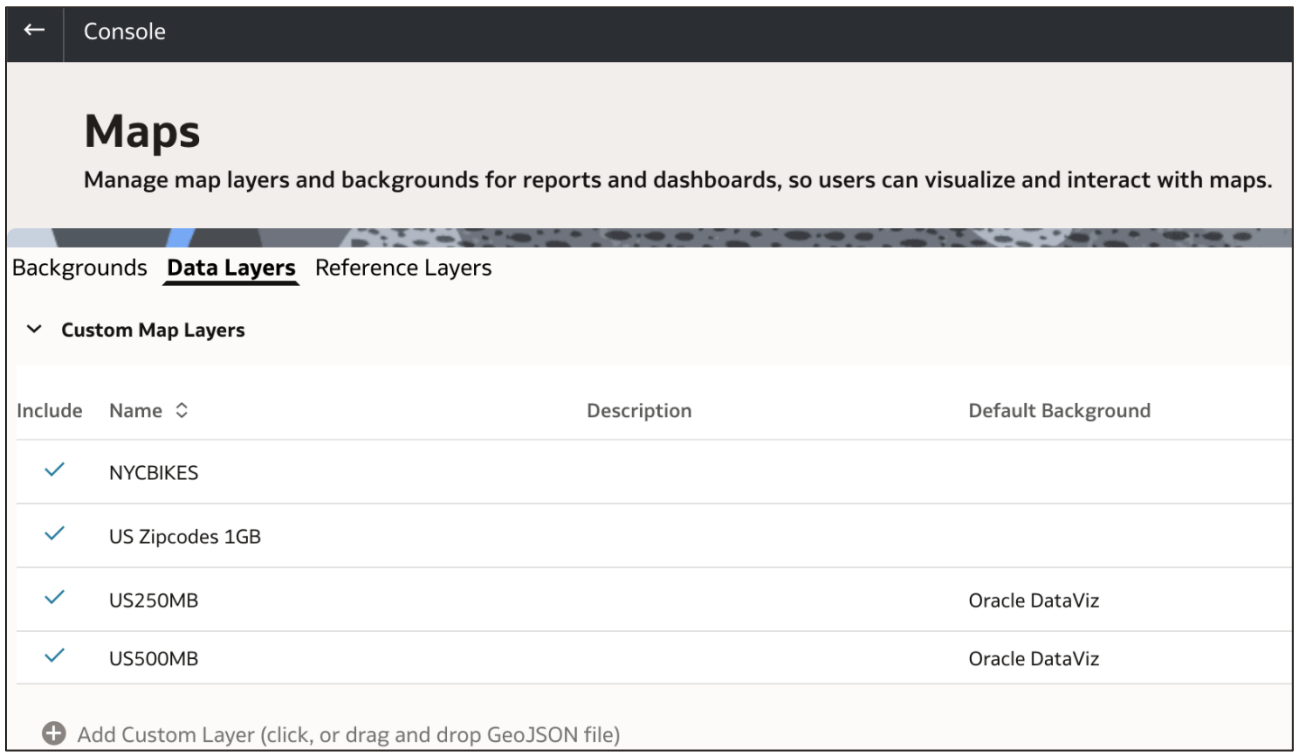

Figure 1: In the Maps area of the Console in Oracle Analytics, the Data Layers tab lists custom map layers such as vector tiles.

Create a New Layer

Creating a new layer binds the vector tile service and metadata service into a single map layer definition. Oracle Analytics retrieves the metadata immediately to confirm that feature keys, bounding boxes, and centroids are valid, and stores the resulting information internally.

1. Click **Add Custom Layer**.
2. Select **Vector Tile** from the layer type options.

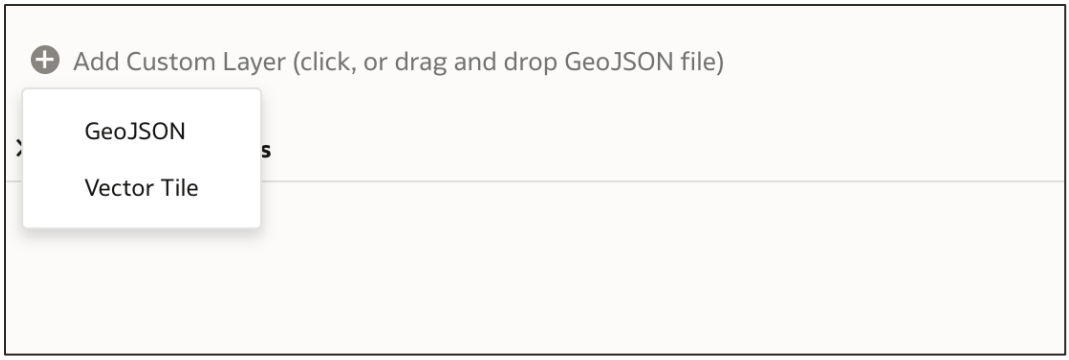

Figure 2: The Add Custom Layer options include Vector Tile.

3. In the Vector Tile Layer dialog, enter your previously published Oracle REST Data Services endpoints:
   A. Vector Source Endpoint (vector tile PBF endpoint URL):

      https://<host>/ords/<schema>/<pbf_module_name>/vt/{z}/{x}/{y}.pbf

   B. Bounding Box Endpoint (GeoJSON metadata endpoint URL):

      https://<host>/ords/<schema>/<bounding_box_module_name>/geojson

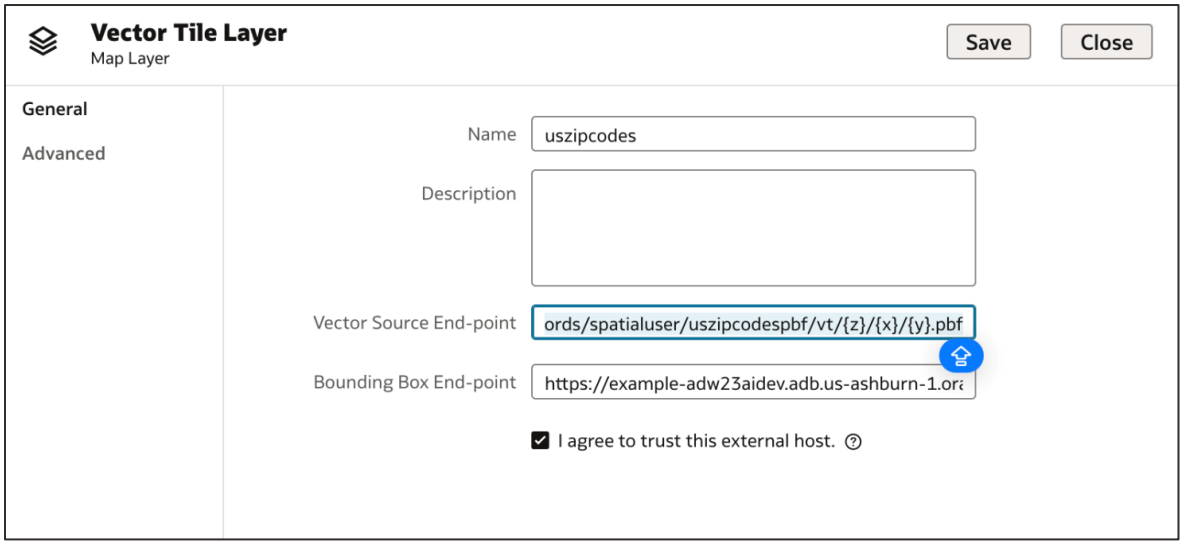

Figure 3: In the Vector Tile Layer dialog, you add your vector tile PBF endpoint URL to the Vector Source End-point field. In this example, https://<host>/ords/<schema>/<pbf_ords_endpoint>/vt/{z}/{x}/{y}.pbf.

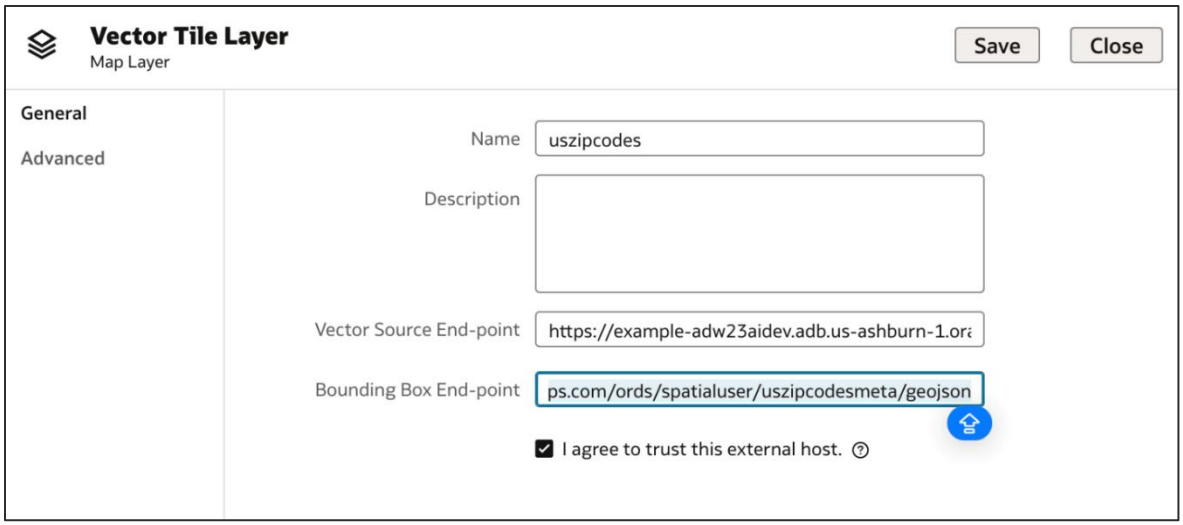

Figure 4: In the Vector Tile Layer dialog, you add your GeoJSON metadata endpoint URL to the Bounding Box End-point field. In this example, https://<host>/ords/<schema>/<bounding_box_ords_endpoint>/geojson.

4. Optional: Click the Advanced tab and choose a map background to visually pair the layer with a base map.

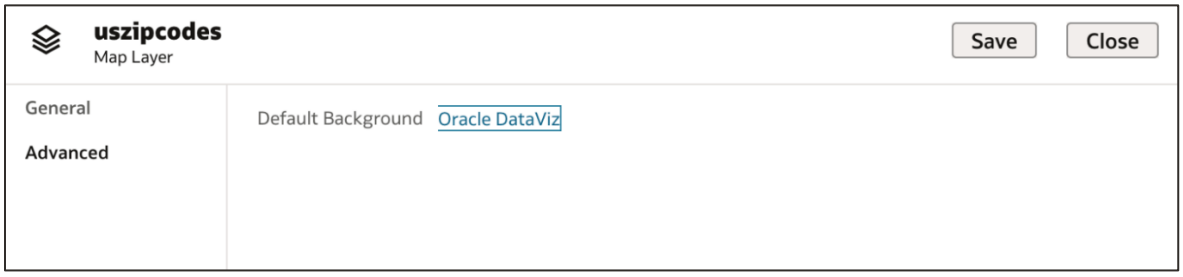

Figure 5: The Advanced tab has a Default Background option for associating a default background map, such as Oracle DataViz, to the vector layer.

5. In the General tab, select **I Agree to trust this external host** to add the host to Oracle Analytics' trusted domain list.

   This enables Oracle Analytics to securely request vector tiles from your database environment.

Save the Layer

Saving the layer completes the integration. Going forward, Oracle Analytics can stream tiles on demand, apply metadata-driven behaviors (zooming, labeling, focusing), and link dataset attributes to the geography through the feature key (GEOID20).

1. Provide a meaningful layer name (e.g., US ZIP Codes – Vector Layer).
2. Optional: Provide a description for governance and catalogue documentation.
3. Click **Save** to register the layer.

After saving, the new vector tile layer appears in the Data Layers list under Custom Map Layers and is available for use in any workbook.

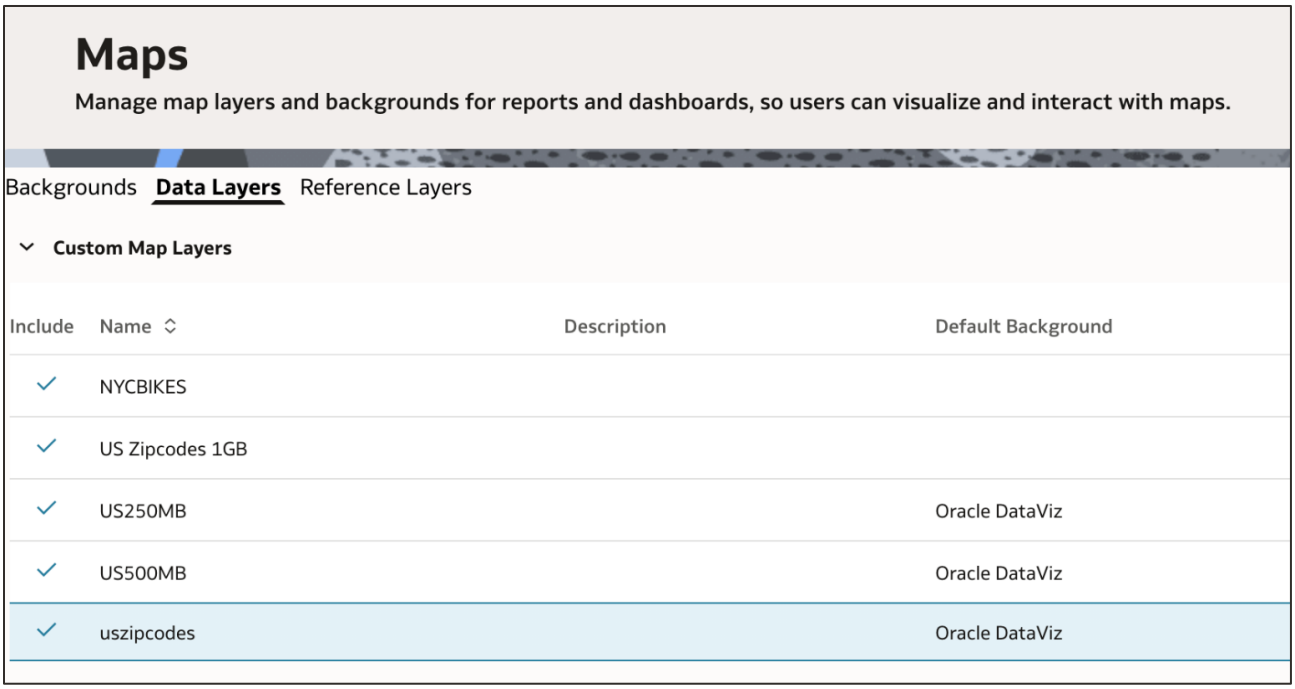

Figure 6: The newly created vector layer called uszipcodes is available under Custom Map Layers in the Data Layers tab.

Step 6: Visualize the Layer in Oracle Analytics

At this stage, Oracle Analytics acts as a tile client. It retrieves vector tiles dynamically, applies metadata-driven interactions such as zooming and labeling, and binds dataset attributes (such as ZIP Codes) to the geography defined in the vector layer.

You can now integrate the new layer into your datasets and visualizations.

Add the Vector Layer to a Map

1. In your workbook, create a new Map visualization.
2. In the Data pane, drag the attribute key column or GEOID20 to **Category (Location)** in the Grammar pane.

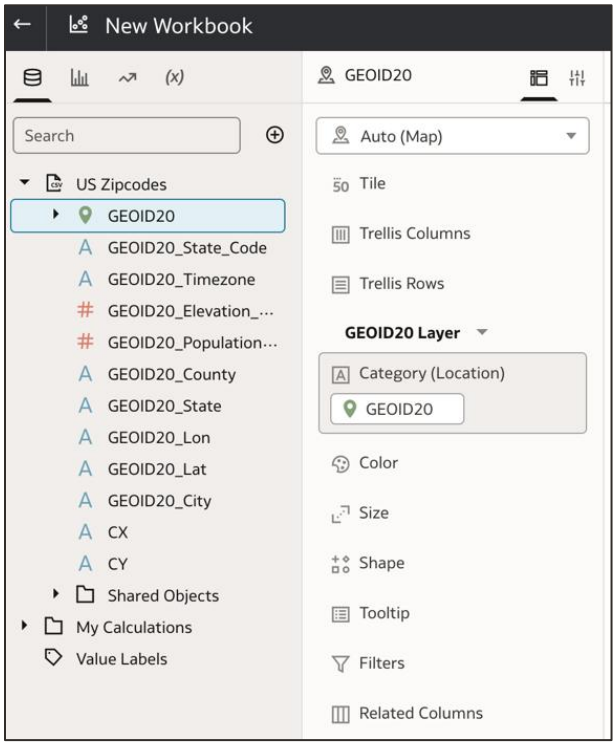

Figure 7: The column GEOID20 in the dataset is added to the Category (Location) drop target under GEOID20 Layer in the Grammar pane.

3. Click **Properties** and go to the Layers tab.
4. In the Map Layer row, click **Auto** and select the custom vector tile layer from the list.

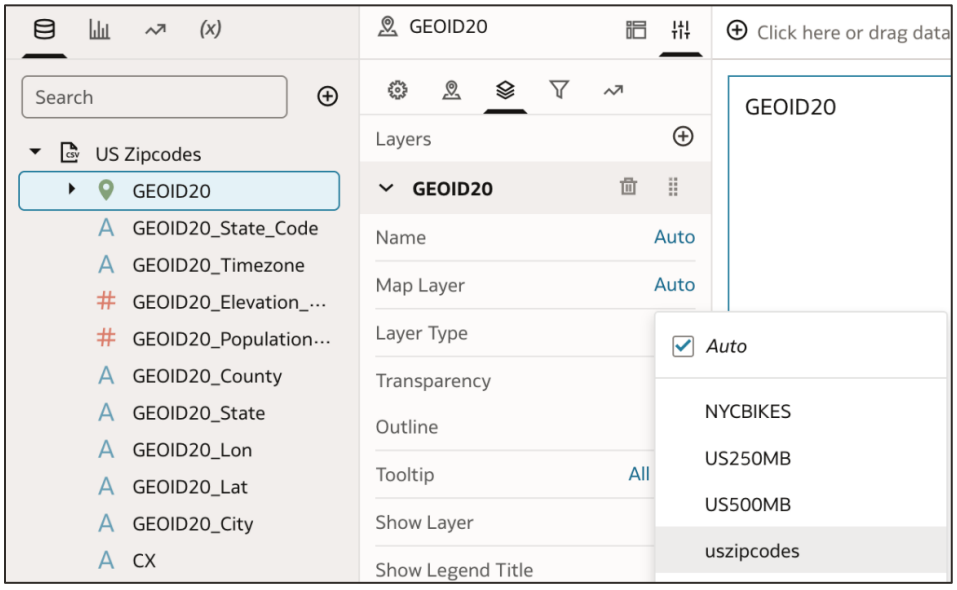

Figure 8: In the Layers tab of the Properties pane, the key column is mapped to the vector layer in the Map Layer row for the GEOID20 layer.

This binds the dataset values to the geographic features defined in the vector tile service. Every distinct key value associates with a feature in the metadata layer previously retrieved from Oracle REST Data Services. Once the vector layer is selected, Oracle Analytics loads the metadata, positions the layer, and begins requesting vector tiles based on the visible map region.

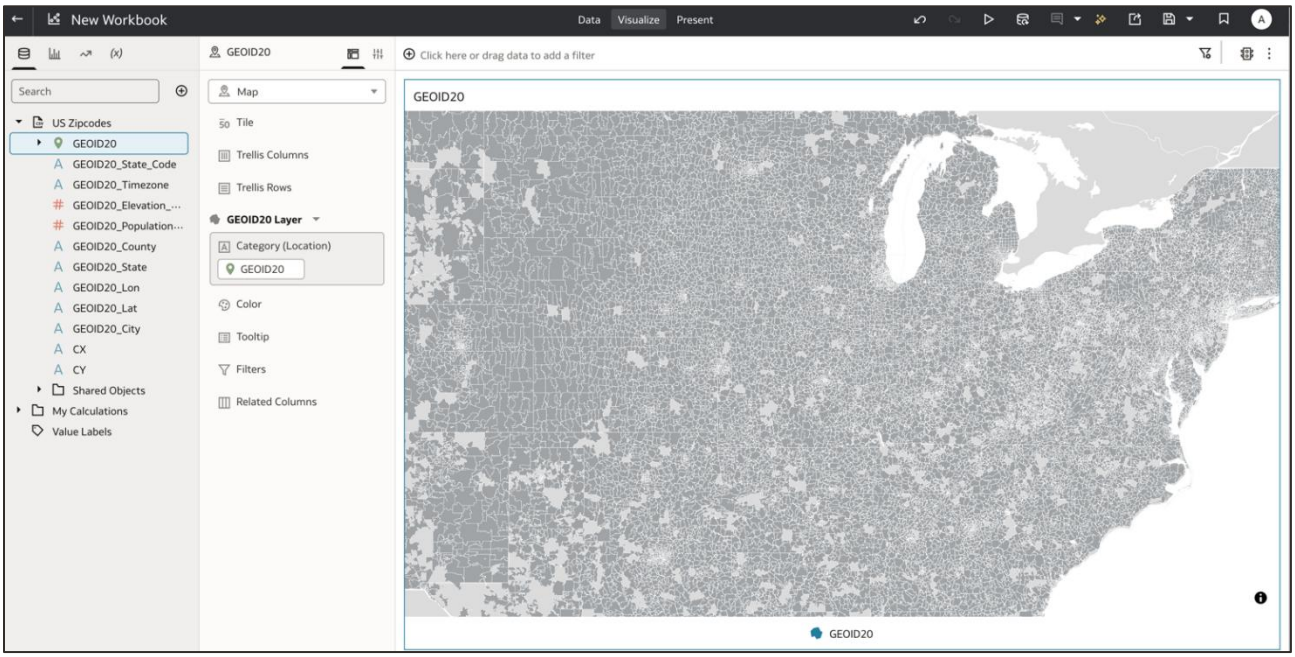

Figure 9: Vector tiles are streamed from the database as the user zooms or pans across the map visualization.

Apply Metrics, Styling, and Interactivity

Drag a measure (for example, Population or Elevation) to the **Color** drop target in the Grammar pane.

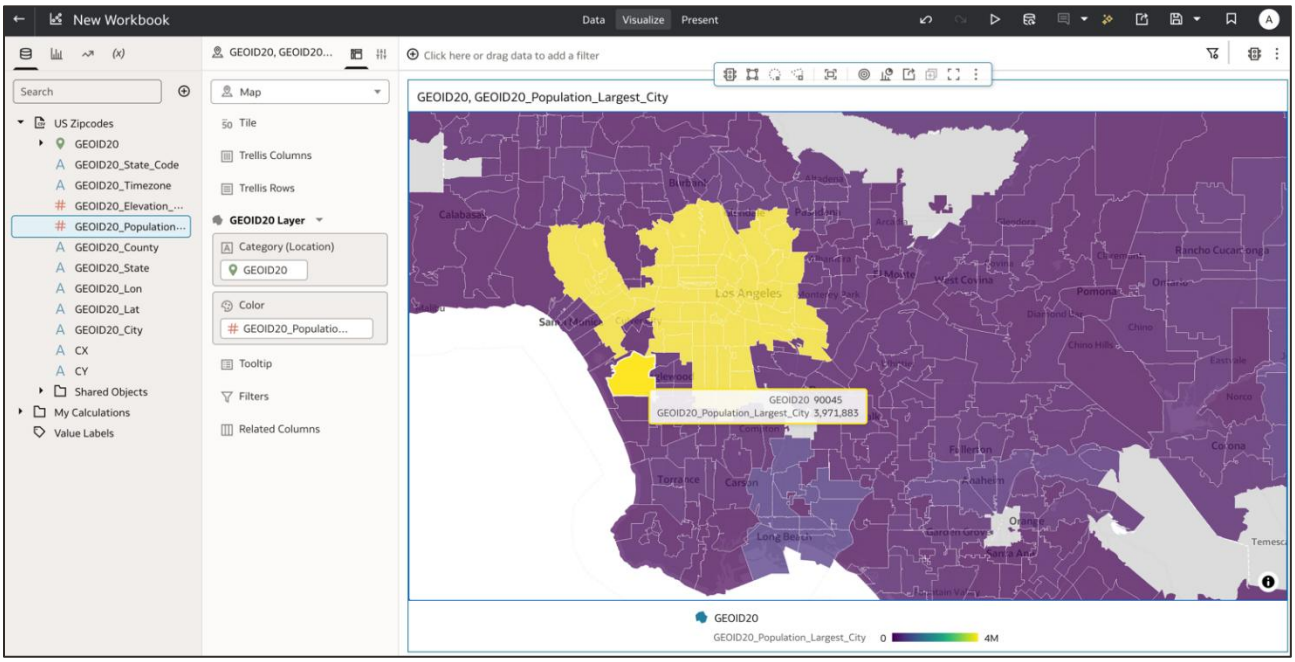

Figure 10: The measure GEOID20_Population_Largest_City is added to Color in the Grammar pane to apply a color scheme to the map.

Use the Properties pane to adjust the transparency, thematic colors, borders, and legends for the visualization. Use interactive filters to explore the data.

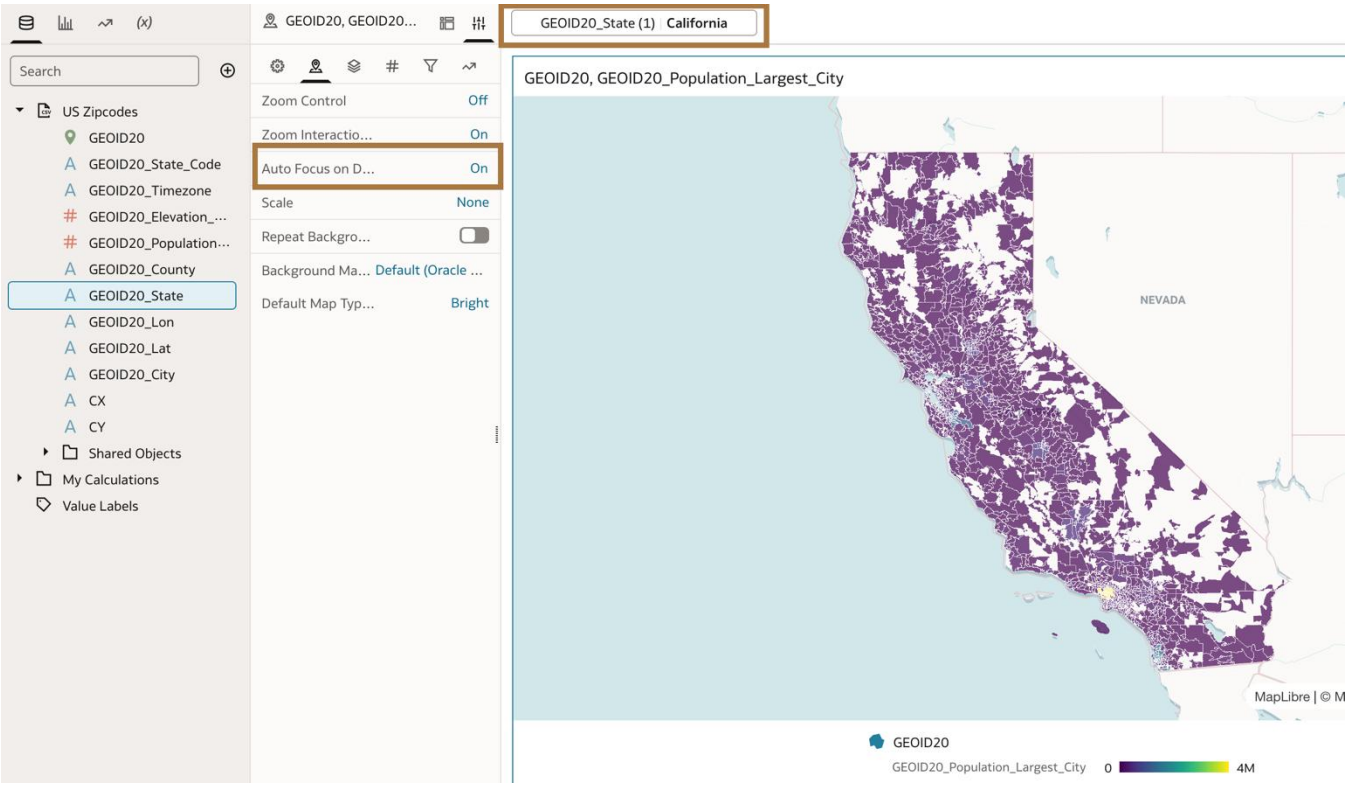

Figure 11: In the Properties pane, the Auto Focus on Data Property is turned on. In the filter bar, the column GEOID20_State is added as a filter with the state of California selected. The map visualization is filtered based on that filter selection.

Implementation Considerations and Performance Notes

While vector tiles provide significant performance and scalability advantages, it's important to understand certain characteristics of the current implementation and how they influence end-to-end behavior in Oracle Analytics.

Geometry Type Support

The present vector tile generation workflow through Oracle Analytics supports only polygon geometries. This includes boundaries such as ZIP codes, administrative regions, districts, parcels, service zones, and other closed-shape features. Support for point and line geometries as vector tiles in Oracle Analytics is planned for future iterations of the vector tile engine. Until then, point layers (such as store locations or incident markers) and line layers (such as roads or network traces) should continue to be delivered through conventional GeoJSON or direct geometry-based approaches.

Geometry Validation

Spatial datasets may contain invalid or inconsistent geometries, including self-intersections, duplicate vertices, or minor boundary gaps. These conditions can affect spatial operations used during vector tile generation. Oracle Spatial provides validation and rectification utilities that can be used to identify and correct such issues prior to publishing vector tile services. Valid geometries help ensure consistent tile generation and predictable map behavior.

For example, check geometry validity using a SQL query like this:

```
SELECT COUNT(*) AS invalid_count
FROM USZIPCODES
WHERE SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(GEOM, 0.005) <> 'TRUE';
```

For example, rectify invalid shapes using a SQL query like this:

```
UPDATE USZIPCODES
SET GEOM = SDO_UTIL.RECTIFY_GEOMETRY(GEOM, 0.005)
WHERE SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(GEOM, 0.005) <> 'TRUE';
```

Corrected geometries avoid rendering gaps and ensure that each feature can be reliably processed into vector tiles.

Geometry Metadata Registration

Oracle Spatial relies on geometry metadata to correctly interpret coordinate systems, dimensional extents, and tolerance values. Oracle AI Database Version 26ai automatically manages this metadata during spatial data loading. For traditional database deployments, explicit registration using the USER_SDO_GEOM_METADATA table may be required.

This metadata enables accurate spatial indexing and ensures predictable behavior during vector tile generation.

For example, a SQL query for spatial metadata registration (only traditional database deployments) might look like this:

```
INSERT INTO user_sdo_geom_metadata (
  table_name, column_name, diminfo, srid
)
VALUES (
  'USZIPCODES',
  'GEOM',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('LONG', -180, 180, 0.05),
    SDO_DIM_ELEMENT('LAT', -90, 90, 0.05)
  ),
  4326
);
```

This metadata informs Oracle Spatial how to interpret the geometry during indexing and tile generation.

Spatial Indexing

Vector tile generation relies on spatial intersection queries to determine which features intersect a given tile envelope. For this reason, a Spatial Index V2 on the geometry column is recommended.

Spatial Index V2 is designed for modern spatial workloads like tile generation and improves the efficiency of spatial filtering operations used during tile generation. This directly influences tile response times when users interact with maps in Oracle Analytics

For example, create a spatial index with a SQL query like this:

```
CREATE INDEX USZIPCODES_SIDX
ON USZIPCODES(GEOM)
INDEXTYPE IS MDSYS.SPATIAL_INDEX_V2;
```

Tile generation queries depend heavily on spatial filtering. The index accelerates this filtering process, contributing directly to faster tile production during Oracle Analytics interaction.

Initial Tile Generation and Caching Behavior

For datasets containing high-resolution, high precision, or large volumes of polygon geometries, the first visualization load may take longer than subsequent interactions.

This is expected because:

- The database must generate the vector tiles dynamically for the first time.
- The tile cache is initially empty.
- The geometry may include thousands of vertices or complex boundaries that require computational processing.

Once this first load completes, database-level vector tile caching ensures that all further tile requests for the same geographic regions are served immediately from cache rather than recomputed.

This results in:

- Faster dashboard refreshes
- Smooth panning and zooming
- Stable performance even for multi-region or nationwide datasets

The overall user experience improves substantially after the cache is populated.

Bounding Box Retrieval

Oracle Analytics retrieves the bounding box metadata GeoJSON endpoint at registration time and during layer initialization per session.

For datasets with a large number of features, metadata retrieval may result in small one-time bandwidth usage. However, this metadata is lightweight compared to the geometries themselves and is typically loaded only once per session.

SQL Execution Considerations

For executing the SQL statements in this workflow, Oracle SQL Developer Extension for VS Code or Oracle SQL Developer Desktop Edition is the recommended approach, particularly for large or complex spatial datasets.

You can download Oracle SQL Developer Extension for VS Code or Oracle SQL Developer (Desktop Edition) from https://www.oracle.com/database/sqldeveloper/.