Oracle Fusion Cloud Sales Automation

How do I get started with Sales Intelligence?

Oracle Fusion Cloud Sales Automation
How do I get started with Sales Intelligence?

G29768-01

Author: Oracle

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Some application pages have help icons ? to give you access to contextual help. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons. If the page has contextual help, help icons will appear.

Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

1 How do I get started with Sales Intelligence?

Sales Intelligence Features and Technologies

Sales Intelligence includes many analytics and artificial intelligence features that can help your sales organization improve sales efficiency and productivity.

The features use Oracle's machine learning and predictive analytics capabilities:

- Machine Learning

  A platform for creating sales machine learning models. Oracle provides you with sample models for Similar Accounts, Opportunity predicted win probability, and Lead scoring. As a sales administrator without prior modeling expertise, you can configure a model by duplicating predefined model templates, or you can develop completely new machine learning models of your own.
- Sales Insights

  Analyzes historical account, lead, opportunity, and contact data to provide salespeople with key insights and recommended actions. As a sales administrator, you can enable or disable individual recommendations.

Here's a list of the different Sales Intelligence features that you can enable by using the Enable Sales Intelligence Features task from Setup and Maintenance. When you enable a feature, the application determines if you've enough data to make the feature active. If the feature is available for use, you can then configure it further using the administration UIs for each technology.

Account Recommendations

Feature	Description	Technology
Account Engagement Level	Rates the level of engagement of the sales organization with each account. The rating is based on the number of active leads, the number of deals, and the frequency of interactions and activities with the account.	Sales Insights
Similar Accounts	Identifies similar accounts by such factors as industry, address, and company size. **Note:** Your administrator can configure the model to identify similarity based on your own specific criteria.	Machine Learning

Opportunity Recommendations

Feature	Description	Technology
Opportunity Predicted Win Probability	Predicts win probability using historical data.	Machine Learning

Feature	Description	Technology
Opportunity Activity Effectiveness	Suggests actions if the level of activities on an opportunity is lacking compared to similar successful opportunities.	Sales Insights

Lead Recommendations

Feature	Description	Technology
Lead Scoring	Indicates the probability of a successful conversion of sales leads to opportunities.	Machine Learning
Lead Activity Effectiveness	Suggests actions if the level of activities on a lead is lacking compared to similar successful leads.	Sales Insights

Contact Recommendations

Feature	Description	Technology
Contact Engagement Level	Rates the level of engagement of the sales organization with each contact. The rating is based on the number of active leads, the number of deals, and the frequency of interactions and activities with the contact.	Sales Insights
Lead Contact Recommendations	Recommends contacts to add to leads. Contacts are recommended based on their involvement in past successful leads, deals, and their level of interactions with the sales organization (the number of meetings, calls, and emails).	Sales Insights
Lead Contact Validation	Validates if the contacts pursued on the lead are enough to successfully convert a lead	Sales Insights
Opportunity Contact Recommendation	Recommends contacts to add to similar successful opportunities.	Sales Insights
Opportunity Contact Validation	Validates if contacts pursued on the opportunity are enough to successfully close the opportunity.	Sales Insights

Enable Sales Intelligence Features

You can enable the Sales Intelligence features provided by Oracle from the Setup and Maintenance work area. Here's how.

Before you begin:

You must have the following privileges to administer and work with Sales Intelligence, view reports, and run appropriate processes:

- ZCA_MANAGE_SALES_MACHINE_LEARNING_PRIV
- ZCA_GENERATE_SALES_INTELLIGENCE_DATA_CHECK_REPORT_PRIV
- ZCA_GENERATE_SALES_INSIGHTS_PRIV
- ZCA_VIEW_SALES_RECOMMENDATIONS_PRIV
- ZCA_VIEW_SIMILAR_RECORDS_PRIV
- ZCA_VIEW_SIMILAR_ACCOUNTS_PRIV
- ZCA_VIEW_SALES_INSIGHTS_PRIV
- ZCA_SALES_INSIGHTS_JOB_PRIV
- FUSION_APPS_CRM_ESS_APPID

1. In Setup and Maintenance, open the **Enable Sales Intelligence** task:
   - Offering: Sales
   - Functional Area: Intelligence in Sales
   - Task: Enable Sales Intelligence
2. On the Enable Sales Intelligence Features page, Select Features panel, click **Start**.
3. Select the features that you want to enable.
4. Click **Continue**.

   The application runs a process to decide if you've enough data to enable the features. This process might take a while to complete, so you can close the flow and wait for a notification email.
5. On the View Feature Eligibility Report page, you get feedback if you've enough data for each feature. Optionally, you can enable and disable features here as well.
6. Click **Continue**.
7. The Verify Recommended Features step lists the features you're enabling by technology. You must complete any extra configuration steps in the admin UIs for each technology.
8. Click **Continue**.
9. In the Configure Recommended Features page, you can click the links to open the administration UIs for technologies related to the features you enabled. Links are available for Machine Learning. There's no direct link for Sales Insights. You can also navigate to these UI pages separately.

Once enabled, users must have the following privileges to view model outcomes:

- ZCA_VIEW_SALES_RECOMMENDATIONS_PRIV
- ZCA_VIEW_SIMILAR_RECORDS_PRIV
- ZCA_VIEW_SALES_INSIGHTS_PRIV

Sales Machine Learning

Overview of Sales Machine Learning

Machine learning models recognize patterns in your data and make predictions using the attributes you select in the model. Oracle provides you with sample models for Similar Accounts, Opportunity predicted win probability, and Lead scoring. For example, the predefined Similar Accounts model recognizes similar accounts using attributes such as location, size, and industry.

> **Note:** You must have the Manage Sales Machine Learning (ZCA_MANAGE_SALES_MACHINE_LEARNING_PRIV) privilege to build sales machine learning models.

To view and access the various sales intelligence features, users must have the following privileges:

- ZCA_VIEW_SIMILAR_RECORDS_PRIV
- ZCA_VIEW_SALES_RECOMMENDATIONS_PRIV
- ZCA_VIEW_SALES_INSIGHTS_PRIV

As a sales administrator without prior training in AI, you can use the Machine Learning feature to change the existing Similar Accounts model, and you can build machine learning models of your own.

This table provides a summary of the predefined models that you can choose:

Model Name	Description
Similar Accounts	Use this model to recognize similar accounts using attributes such as location, size, and industry.
Opportunity Predicted Win Probability	Use this model to predict the win probability of opportunities using historical data.
Lead Scoring	Use this model to indicate the probability of a successful conversion of sales leads to opportunities.

This table shows the setups that you need to complete to enable the Sales Machine Learning feature and show similar accounts on the Account pages of both Oracle Sales in the Redwood User Experience and classic Oracle Sales.

Step	Applies To	Description	Navigation	Where to Get More Details
1	Both Oracle Sales and classic Sales	Create and deploy the model either by copying the Similar Accounts model or creating one from scratch.	Open the Machine Learning Models page either from the last step in the Enable Sales Intelligence Features task. You can also search for the Configure Sales Machine Learning task using the Tasks search panel of the Setup and Maintenance work area.	See the topics: • *Overview of Building Machine Learning Models* • *Create a Similar Accounts Model* • *Create a Machine Learning Model from Scratch*
2	Oracle Sales only	You can clone an existing template or create a new template from scratch to display similar records on your Oracle sales UI. Creating a new template gives you full control over the structure and parameters you want to include from the beginning. You can display explanations for machine learning predictions as a drawer in Oracle Sales by creating a	Oracle Visual Builder Studio	• *How can I display similar records identified by Machine Learning models in Oracle Sales?* • *How can I display explanations for Machine Learning predictions as a drawer in Oracle Sales?*

Step	Applies To	Description	Navigation	Where to Get More Details
		virtual field and exposing it as a hyperlinked field.		
3	Classic Sales	If you copied the Similar Accounts model, the application displays similar accounts automatically in Oracle Sales when salespeople enter **`Show similar accounts`** in the Action Bar. For classic Sales, you must display the Similar Accounts in the UI using Application Composer.	Application Composer	See the topics in the section: Display Similar Accounts on the Account pages of Classic Sales. *High-Level Setup Steps*

Overview of Building Machine Learning Models

Here’s an overview of the high-level steps for building your machine learning models. Note that you must have the Manage Sales Machine Learning (ZCA_MANAGE_SALES_MACHINE_LEARNING_PRIV) privilege to build sales machine learning models.

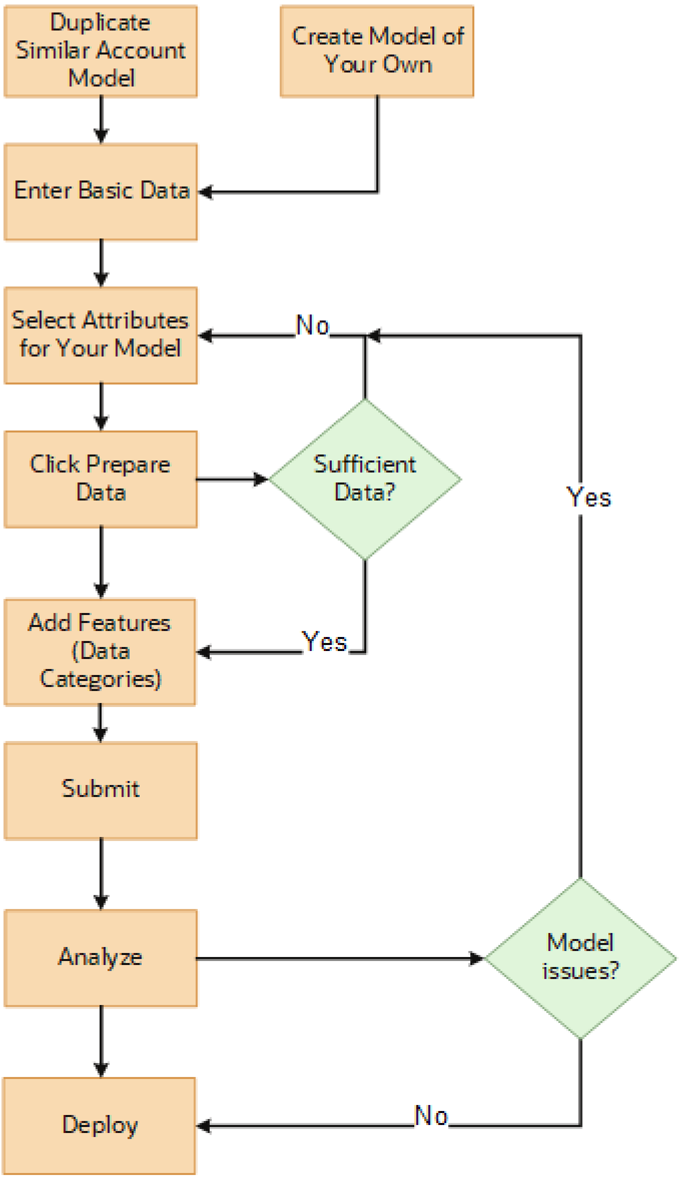
1. You've a choice of either duplicating an existing model and changing the copy, or you can create a model of your own from scratch.
2. You add some basic information. If you're creating a model from scratch, you select the object and type of model you're building: either a prediction model or a model to identify similar records. For example, select the model type you want to build such as:
   - Predict outcome - This model type is used for making predictions such as predicting if an opportunity is likely to get won. It can also be used to predict deal size (number) or time to close (days) or account score (number).
   - Identify similar records - This model type is used for finding similar records such as finding other customers similar to the one being viewed. It can also be used to find similar opportunities to the one being worked on.
3. You select the attributes you want to use for your model.
4. You can add data set filters to define your training data set for both prediction and similar records models

   - Prediction models - This is the data set where machine learning models are trained. For example, if a model needs to make a prediction on an outcome for an opportunity, it needs to learn the patterns for both a win and loss outcome by learning from past closed opportunities. During model training, the model learns from the patterns that are different for won and lost opportunities.
   - Similar records models - This is the data set where similar records are found. For example, if active accounts are used as part of the training data set, for each account, it will find only similar records from active accounts. Any inactive accounts will not surface as similar records.
5. You can specify the attributes that stores prediction outcomes for the predict outcome model type. Select what outcome would be considered a successful outcome for this use case. For example, for an opportunity prediction, an outcome of Won might be considered as success criteria.
6. You click **Prepare Data** to check if there's enough data for each attribute.
7. If your model results in the Error status, you don't have enough data for some attributes and you must select different attributes or fix the data.

8. If your model completes with the Prepared status, then you can add data categories for the different attributes in the Add Features step.
9. Click **Submit** to build the model.
10. Analyze the model results. You get tips on how to improve the model.
11. If the model needs improvement, you can go back to the previous steps to tweak it.
12. You can deploy the model to automatically run on a daily, weekly, or a monthly schedule.

> **Tip:** For prediction models, you should rebuild your model based on how often the patterns being predicted would change. For example, the training recurrence for opportunity win probability models can be determined by how often the pattern of winning versus losing opportunities change. For similar records models, you should rebuild model based on how soon you'd expect new or updated records to show up as similar records. For example, any newly created account should show up as similar within a day or within a week.

Here's a sample flowchart that shows how to build a machine learning model.



Related Topics

- Create a Similar Accounts Model
- Create a Machine Learning Model from Scratch

Navigate to the Machine Learning Models Page

You can open the Machine Learning Models page either from the last step in the Enable Sales Intelligence Features task or you can search for it using the Tasks search panel of the Setup and Maintenance work area.

1. In the Setup and Maintenance work area, select the **Sales** offering.
2. Click the **Tasks** side panel.
3. Click **Search**.

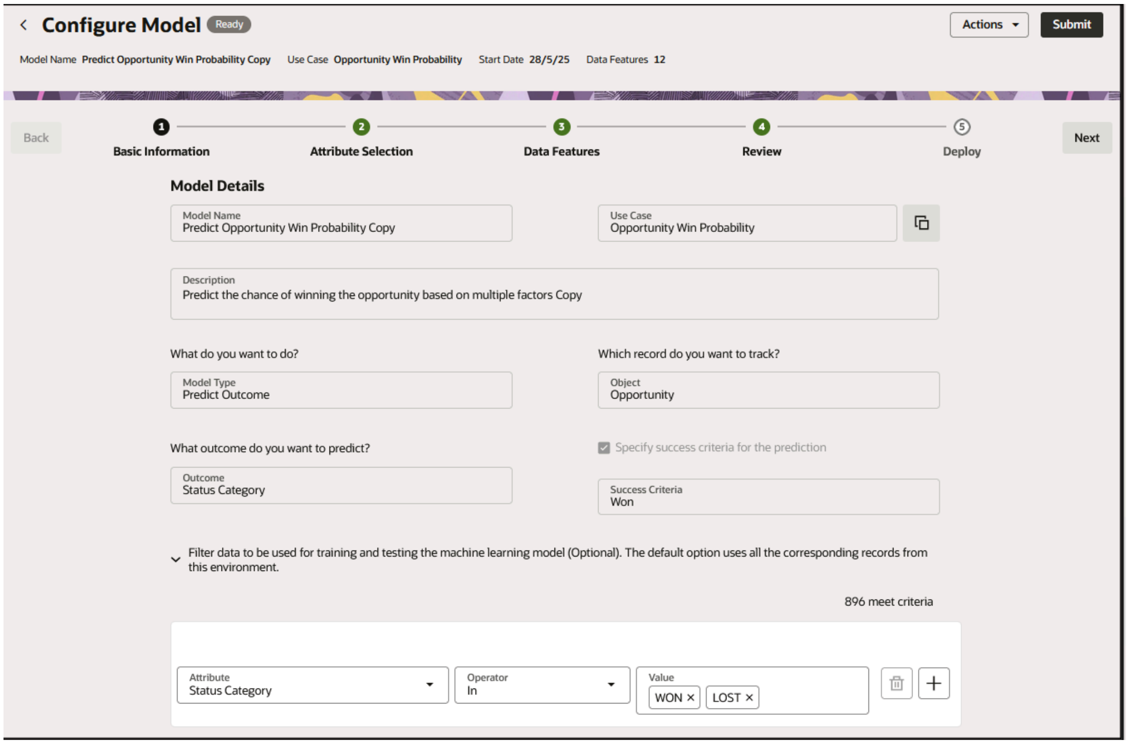
4. Search for the **Configure Sales Machine Learning** task and click the link.

Create a Similar Accounts Model

Here's how to create a similar accounts model of your own.

1. Navigate to the *Machine Learning Models page*.
2. On the Machine Learning Models page, click **Actions** (hamburger icon) and select **Duplicate** for the Similar Accounts model.

   Your action creates a copy of the predefined model in the Draft status.
3. In the Basic Information step, enter a name and description. They're visible only in the administration pages.



Model Name Predict Opportunity Win Probability Copy   Use Case Opportunity Win Probability   Start Date 28/5/25   Data Features 12

Model Details

Model Name: Predict Opportunity Win Probability Copy

Use Case: Opportunity Win Probability

Description: Predict the chance of winning the opportunity based on multiple factors Copy

What do you want to do?

Model Type: Predict Outcome

Which record do you want to track?

Object: Opportunity

What outcome do you want to predict?

Outcome: Status Category

☑ Specify success criteria for the prediction

Success Criteria: Won

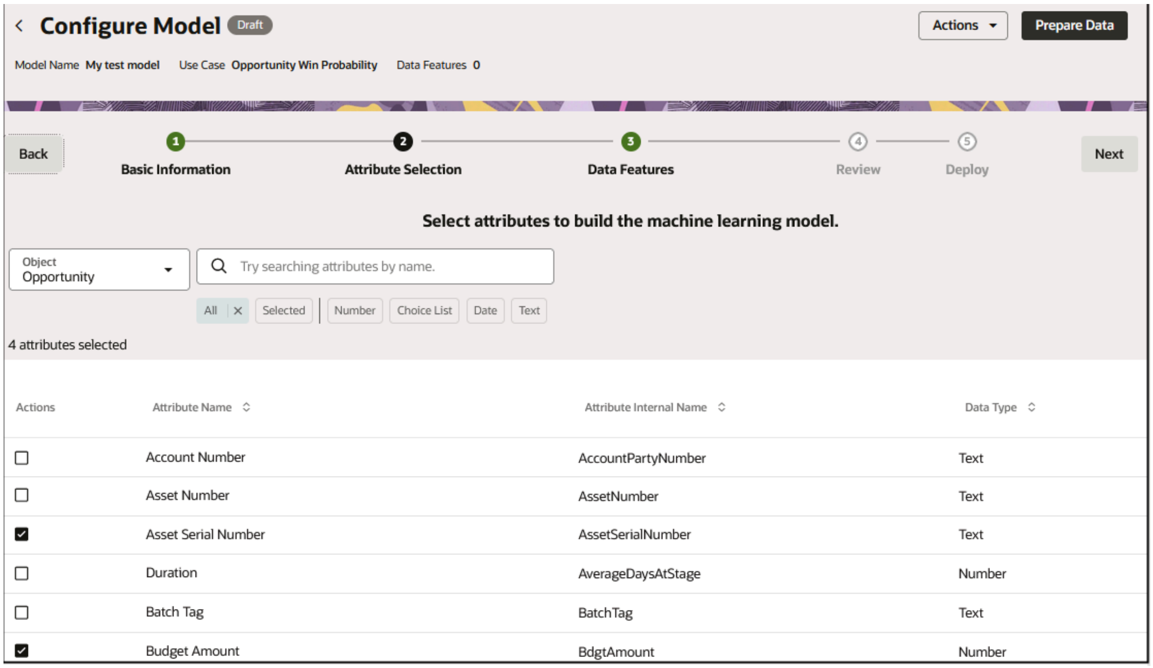
Filter data to be used for training and testing the machine learning model (Optional). The default option uses all the corresponding records from this environment.

896 meet criteria

Attribute: Status Category   Operator: In   Value: WON × LOST ×

4. Click **Next**.

5. The Attribute Selection page lists the attributes for the model provided by Oracle as shown:



Model Name My test model Use Case Opportunity Win Probability Data Features 0

Select attributes to build the machine learning model.

Object: Opportunity

Try searching attributes by name.

4 attributes selected

Actions	Attribute Name	Attribute Internal Name	Data Type
□	Account Number	AccountPartyNumber	Text
□	Asset Number	AssetNumber	Text
☑	Asset Serial Number	AssetSerialNumber	Text
□	Duration	AverageDaysAtStage	Number
□	Batch Tag	BatchTag	Text
☑	Budget Amount	BdgtAmount	Number

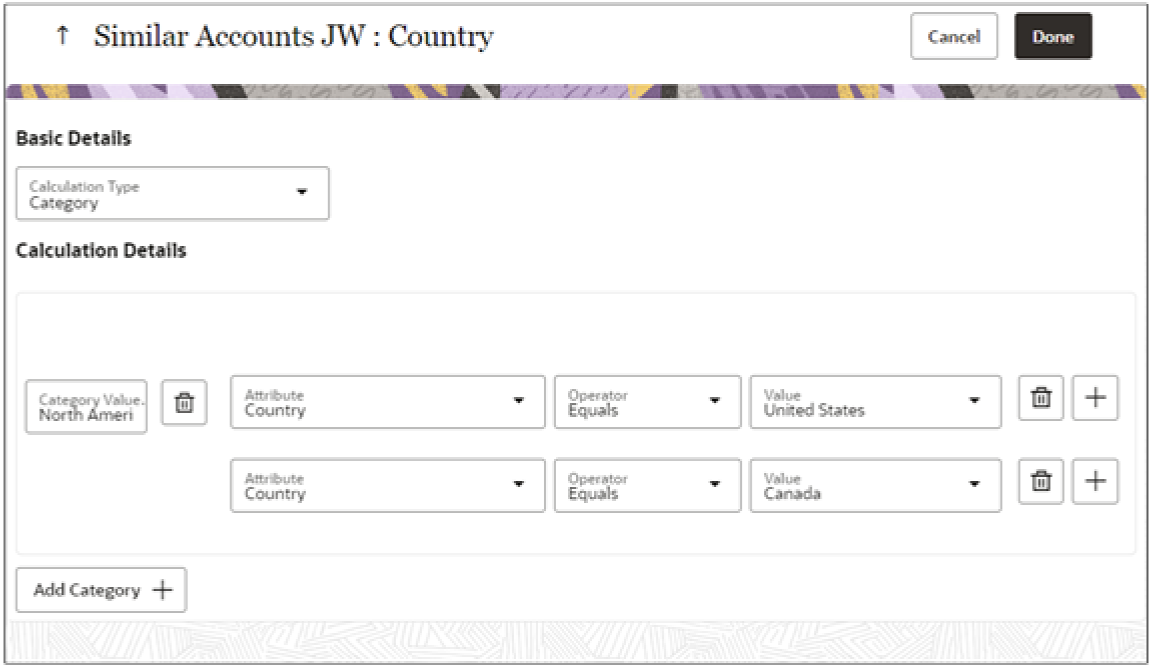
.

6. Enable or disable the attributes you want the model to use. For example, the model provided by Oracle includes Postal Code as one of the attributes. Too many distinct values will make the model less useful and slow to run, so remove Postal Code if you're selling in multiple countries, for example.
   a. From the **Actions** area, select or deselect the checkbox to enable or disable the attributes you don't need.
   b. Use the search field to add related object attributes.
7. Click **Prepare Data** to validate if enough data exists to create a model from the attributes you selected.

   The **Status** column on the Machine Learning page shows the **Running** status while the validation process is running.
8. Click **Refresh** to check for status changes.
9. A status of **Error** means that you don't have enough data for some attributes you selected for your model. At least 30 percent of the records must have a value for the attribute. Here's what to do:
   a. Click the name link of your model.
   b. Click to the **Attribute Selection** step to see a list of the errors.
   c. Delete the attributes that you can't use.
   d. Click **Prepare Data** again.
10. If your model is in the **Prepared** status, you've enough data to analyze and tweak your model further.
11. Click **Next** to go to the **Data Features** step.
12. Click **Actions** > **Edit** for any of the attributes to fine tune you model by categorizing the values.
13. On the Calculation Type page, you can provide one or more categories for the model to consider. Categories affect the way that the model learns and how it creates clusters of records. For example, if you find out that your model has too many unique values for a particular attribute, you can come here and group them. Which calculation type is available depends on the attribute:
    - **Age Date bucket**: Use this calculation type for date-time attributes such as Creation Date. You can define age groups, such as 0 to 1 year old, 1 to 2 years old, and 2 to 3 years old, instead of a date. Then, your model searches for similar accounts within the configured age groups.

- **Number bucket**: Use this calculation type for numeric attributes like Potential Revenue and Organization Size. For example, instead of searching by a deal amount, you can create number buckets of equal ranges, such as 0 to 100,000, and 100,000 to 200,000. Your model will search for similar accounts that have a deal amount within the configured number buckets.
- **Category**: You can use this calculation type to create categories based on attribute conditions. For example, you can categorize accounts by world regions, by grouping countries as Latin America, North America, Asia Pacific, and Europe. Or you can categorize accounts as Large, Medium, or Small, using Opportunity Revenue or Organization Size.

14. Here's how to classify the countries where you do business by geographical regions, for example:
    a. Click **Actions** > **Edit** for Country.
    b. From the **Calculation Type** list, select **Category**.
    c. In the **Category Value** field, enter **North America**.
    d. From the Operator list, select Equals
    e. In the Value field, search and enter one of the countries in North America.
    f. Click **Add Another Rule** (the plus sign) and add a second country.
    g. Repeat the process until you added all the countries in the category.
    h. Click **Add Category** to add more categories.
    i. Click **Done**.



Basic Details

Calculation Type: Category

Calculation Details

Category Value: North Ameri

Attribute: Country — Operator: Equals — Value: United States

Attribute: Country — Operator: Equals — Value: Canada

15. When you're done adding calculation types, click **Submit** to run the model.
16. Click **Refresh** to refresh the status.
17. When the model status changes from **Running** to **Ready**, click **Actions** > **Edit**. The Actions menu is the hamburger icon on the right side of the page.
18. Click the **Review** step to review similar accounts the model finds for any account you select:
19. From the **Account** field, select an account to display the similar accounts predicted by the model.
20. Click **Analyze** to review information about your model on the Analysis Report page.
21. The report includes tips on how you can improve the model. For example, the report can tell you that some fields have too few unique values and that others have too many.

If a field has too few values, you'll need to either import more data or eliminate that attribute from the model. If there are too many values, you'll need to group them into categories on the Features step.

Here's some other useful information:

- **Algorithm Selected**: The algorithm that was run for your model. The algorithm is auto selected to achieve best model performance optimized for your data and selected use case. You can't change the algorithm, so you can ignore this field for now.

  > **Note:** Starting from the 25D release you can manually override the selected algorithm to run the model using a specific algorithm. Each time a model is rebuilt, you can manually select an algorithm of your choice. If the data set changes significantly between two different model runs, then the best algorithm is auto selected.

- **Model Accuracy**: Shows the accuracy of your model as a percentage. A good model accuracy is above 90 percent. If the value is less, change the attributes to improve the accuracy.
- **Number of Clusters**: Number of account groupings in your model. The more clusters that you've in your model, the fewer similar accounts that you get. About 20 clusters is a good number.
- The **Data Analysis** tab shows you the number of distinct values for each attribute and the percent empty values, for example.
- The **Model Analysis** tab shows the number of clusters and a pie chart of their distribution.

22. Click **Next**.
23. On the Deploy page: enter the date and time that you want to start running the model and recurrence schedule.
    a. **Date and Time**: Enter the date and time that you want to start the schedule from.
    b. **Recurrence**: Set the frequency for rebuilding your model, depending on how often your data changes.
       - **Daily**
       - **Weekly**
       - **Monthly**
24. Click **Deploy**.

    Your model is now active.

    > **Note:** Only one model can be active at a time. If you already have an active model, you must confirm that you want to replace it.

Create a Machine Learning Model from Scratch

You can create a machine learning model completely from scratch. You can use the model to either predict similar records or predict outcomes for accounts, contacts, opportunities, and sales leads.

Creating a model from scratch involves additional setup in the Basic Information step. Otherwise the process is the same as for creating a similar accounts model from the one provided by Oracle.

1. Navigate to the *Machine Learning Models page*.
2. On the Machine Learning Models page, click **Create**.
3. In the Basic Information step, enter the following:
   - Model name
   - Name of the use case the model covers

- Optional description

4. Select an object. Options are:
    - Account
    - Contact
    - Opportunity
    - Sales Lead
    - Any custom object for Sales
5. Select the model type, either **Identify Similar Records** or **Predict Outcome**.
    - Predict Outcome - This model type is used for making predictions such as predicting if an opportunity is likely to get won. It can also be used to predict deal size (number) or time to close (days) or account score (number).
    - Identify Similar Records - This model type is used for finding similar records such as finding other customers similar to the one being viewed. It can also be used to find similar opportunities to the one being worked on.
6. If you selected **Predict Outcome**, then select the attribute you want to predict. Select what outcome would be considered a successful outcome for this use case. For example, for an opportunity prediction, an outcome of Won might be considered as success criteria. To predict if an opportunity will be won or lost, then select status category.
7. Select the records that you want to include in the data set.
    - Prediction models - This is the data set where machine learning models are trained. For example, if a model needs to make a prediction on an outcome for an opportunity, it needs to learn the patterns for both a win and loss outcome by learning from past closed opportunities. During model training, the model learns from the patterns that are different for won and lost opportunities.
    - Similar records models - This is the data set where similar records are found. For example, if active accounts are used as part of the training data set, for each account, it will find only similar records from active accounts. Any inactive accounts will not surface as similar records.
8. Click **Next**.
9. On the Attribute Selection page, add the attributes you want to use in the model.

    > **Note:** You can now add attributes from additional related records to generate more exact predictions and better insights including the relevant predictors from related records.

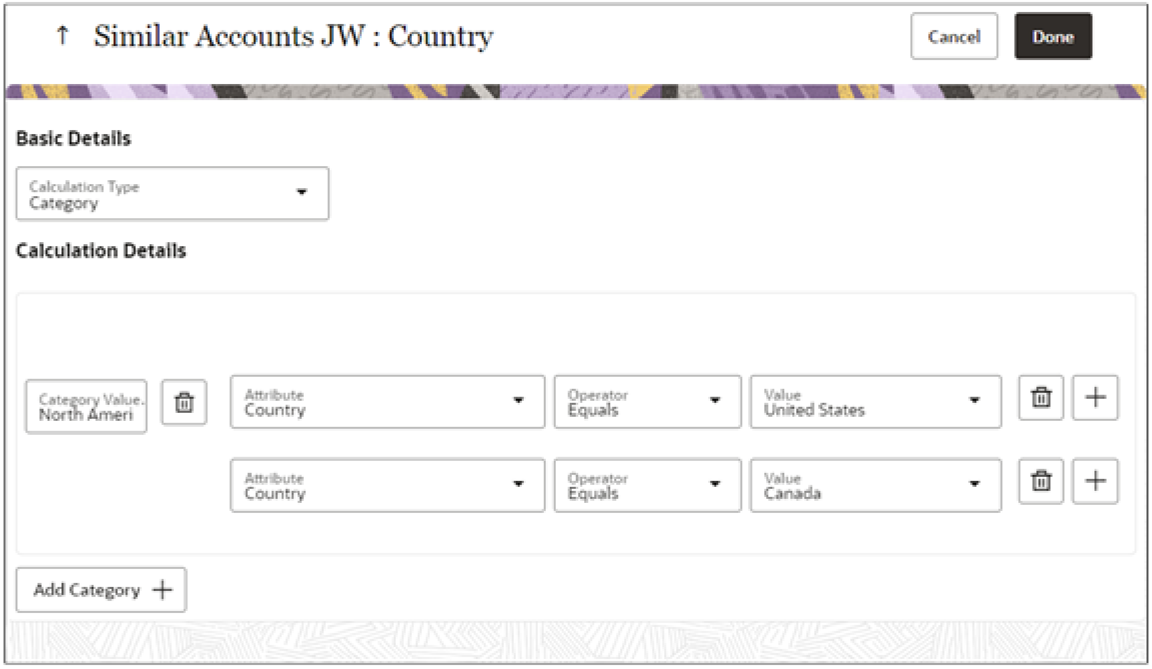
    For example, you can now build an opportunity prediction model including attributes from the account related record. The following related records are supported:
    - Opportunity account
    - Opportunity primary contact
    - Lead account
    - Lead primary contact
    - Custom object relationship for all parent objects
    - Custom object relationship for models built for custom parent objects
10. Click **Prepare Data** to validate if enough data exists to create a model from the attributes you selected.

    The **Status** column on the Machine Learning page shows the **Running** status while the validation process is running.
11. Click **Refresh** to check for status changes.
12. A status of **Error** means that you don't have enough data for some attributes you selected for your model. At least 30 percent of the records must have a value for the attribute. Here's what to do:

a. Click the name link of your model.
b. Click the **Attribute Selection** step to see a list of the errors.
c. Delete the attributes that you can't use.
d. Click **Prepare Data** again.

13. If your model is in the **Prepared** status, you've enough data to analyze and tweak your model further.
14. Click the **Features** step.
15. Click **Actions** > **Edit** for any of the attributes to fine tune you model by categorizing the values.
16. On the Calculation Type page, you can provide one or more categories for the model to consider. Categories affect the way that the model learns and how it creates clusters of records. For example, if you find out that your model has too many unique values for a particular attribute, you can come here and group them. Which calculation type is available depends on the attribute:
    - **Date range**: Calculates the number of days between two specified date attributes.
    - **Date range bucket**: Creates categories of number of days between two specified date attributes.
    - **Age Date bucket**: Use this calculation type for date-time attributes such as Creation Date. You can define age groups, such as 0 to 1 year old, 1 to 2 years old, and 2 to 3 years old, instead of a date. You model will consider age buckets when identifying the similarity or the relationship between this feature and the prediction outcome.
    - **Number bucket**: Use this calculation type for numeric attributes like Potential Revenue and Organization Size. For example, instead of searching by a deal amount, you can create number buckets of equal ranges, such as 0 to 100,000, and 100,000 to 200,000. You model will consider number buckets when identifying similarity or relationship between this feature and the prediction outcome.
    - **Category**: You can use this calculation type to create categories based on attribute conditions. For example, you can categorize accounts by world regions, by grouping countries as Latin America, North America, Asia Pacific, and Europe. Or you can categorize accounts as Large, Medium, or Small, using Opportunity Revenue or Organization Size.

17. Here's how to classify the countries where you do business by geographical regions, for example:
    a. Click **Actions** > **Edit** for Country.
    b. From the **Calculation Type** list, select **Category**.
    c. In the **Category Value** field, enter **North America**.
    d. From the Operator list, select Equals
    e. In the Value field, search and enter one of the countries in North America.
    f. Click **Add Another Rule** (the plus sign) and add a second country.
    g. Repeat the process until you added all the countries in the category.
    h. Click **Add Category** to add additional categories.
    i. Click **Done**.



Similar Accounts JW : Country

Basic Details

Calculation Type: Category

Calculation Details

Category Value: North Ameri

Attribute: Country | Operator: Equals | Value: United States

Attribute: Country | Operator: Equals | Value: Canada

18. When you're done adding calculation types, click **Submit** to run the model.
19. Click **Refresh** to refresh the status.
20. When the model status changes from **Running** to **Ready**, click **Actions** > **Edit**. The Actions menu is the hamburger icon on the right side of the page.
21. Click the **Review** step to review similar accounts the model finds for any account you select:
22. From the **Account** field, select an account to display the similar accounts predicted by the model.
23. Click **Analyze** to review information about your model on the Analysis Report page.

24. The report includes tips on how you can improve the model. For example, the report can tell you that some fields have too few unique values and that others have too many.

    If a field has too few values, you'll need to either import more data or eliminate that attribute from the model. If there are too many values, you'll need to group them into categories on the Features step.

    Here's some other useful information:

    - **Algorithm Selected**: The algorithm that was run for your model. You can't change the algorithm, so you can ignore this field.
    - **Model Accuracy**: Shows the accuracy of your model as a percentage. A good model accuracy is above 75 percent. If the value is less, change the attributes to improve the accuracy.
    - **Prediction accuracy report** : Shows per class prediction accuracy. A good model would have the desired accuracy for each predicted class. For example, an opportunity outcome prediction model will have the desired accuracy in predicting both won records and lost records.
    - **Number of Clusters**: Number of account groupings in your model. The more clusters that you have the fewer similar accounts that you get. About 20 clusters is a good number.
    - The **Data Analysis** tab shows you the number of distinct values for each attribute and the percent empty values, for example.
    - The **Model Analysis** tab shows the number of clusters and a pie chart of their distribution for the Identify Similar Records model. It also shows the feature importance and per class prediction accuracy.
25. Click **Next**.
26. On the Deploy page: enter the date and time that you want to start running the m and recurrence.
    a. **Date and Time**: Enter the date and time that you want to start the schedule from.
    b. **Recurrence**: Set the frequency for rebuilding your model, depending on how often your data changes.
        - **Daily**
        - **Weekly**
        - **Monthly**
27. Specify how to store the prediction outcome for the predict outcome model type:
    - **Scoring recurrence**: Set the frequency for making predictions and storing them, depending on how often you want to refresh (score again) the prediction.
    - **Prediction data set**: Add data set filters to identify the prediction data set. For example, make a prediction for all open opportunities or leads.
28. Click **Deploy**.

    Your model is now active.

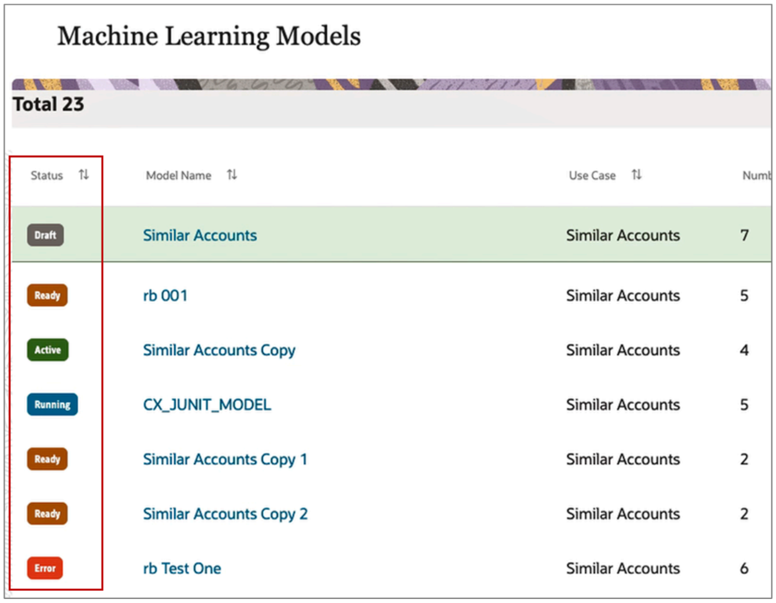
    > **Note:** Only one model can be active at a time. If you already have an active model, you must confirm that you want to replace it.

Model Statuses

This table describes the stages a model goes through and the corresponding statuses it displays while going through them:

Scenario	Status
You create a copy from an existing model, enter basic details, and save. The model appears in the list.	The model is saved as a Draft.
You add attributes and save the model.	The model is saved as a Draft.
You click **Prepare Data** to validate the attributes. The engine triggers a model run.	Whenever the engine prepares the data or builds the model, the status shows Running.
If the run is successful...	The status changes from Running to Prepared.
If the run failed...	The status changes from Running to Error.
You get the status as Error when the attributes you select don't have enough data. Review and change the attributes. Click **Prepare Data** to run the model process again. Result: The run is successful.	Status changes from Running to Prepared.
The engine prepares the data. You now want to build the model. You open the model and click **Submit**.	Status changes from Running to Ready.
Your model is now ready for deployment. Although you can have many models in the Ready status, you can deploy only one of them at any give time. Review the model, enter the deploy schedule, and click **Deploy**.	Status changes from Ready to Active. Your sales team can see the similar accounts based on the similarity identified by the active model.
You want to retire an active model. Click **Retire** from **Actions** (the hamburger icon). A confirmation message appears that the deploy schedule of the model will be deleted.	Status changes from Active to Ready.
You want to delete a model. • To remove an active model, you need to retire the model first and delete it. ○ Click **Retire** from **Actions** (the hamburger icon). ○ After the model status reverts to Ready, click **Delete** from **Actions**. • To remove an inactive model, click **Delete** from **Actions**.	The engine removes the model permanently from the list view and the Digital Sales database.

Here's a screenshot of the Machine Learning models list. It shows about seven similar account models that are in various statuses such as Draft, Ready, Running, and Error. While many models can exist in the Draft and Ready statuses, only one of them can be active. The model statuses are also indicated in different colors. For example, the Error status is shown in a red box, the Running status is shown in blue, and Active is shown in green.



Machine Learning Models

Total 23

Status	Model Name	Use Case	Numb
Draft	Similar Accounts	Similar Accounts	7
Ready	rb 001	Similar Accounts	5
Active	Similar Accounts Copy	Similar Accounts	4
Running	CX_JUNIT_MODEL	Similar Accounts	5
Ready	Similar Accounts Copy 1	Similar Accounts	2
Ready	Similar Accounts Copy 2	Similar Accounts	2
Error	rb Test One	Similar Accounts	6

Display Similar Accounts and Predictive Explanation in Oracle Sales

How can I display explanations for Machine Learning predictions as a drawer in Oracle Sales?

You can display explanations for machine learning predictions as a drawer in Oracle Sales by:

1. Creating a virtual field, which is a field that's generated dynamically and doesn't physically exist in the database.
2. Exposing it as a linked field, which when clicked, will open a drawer to display prediction explanations.

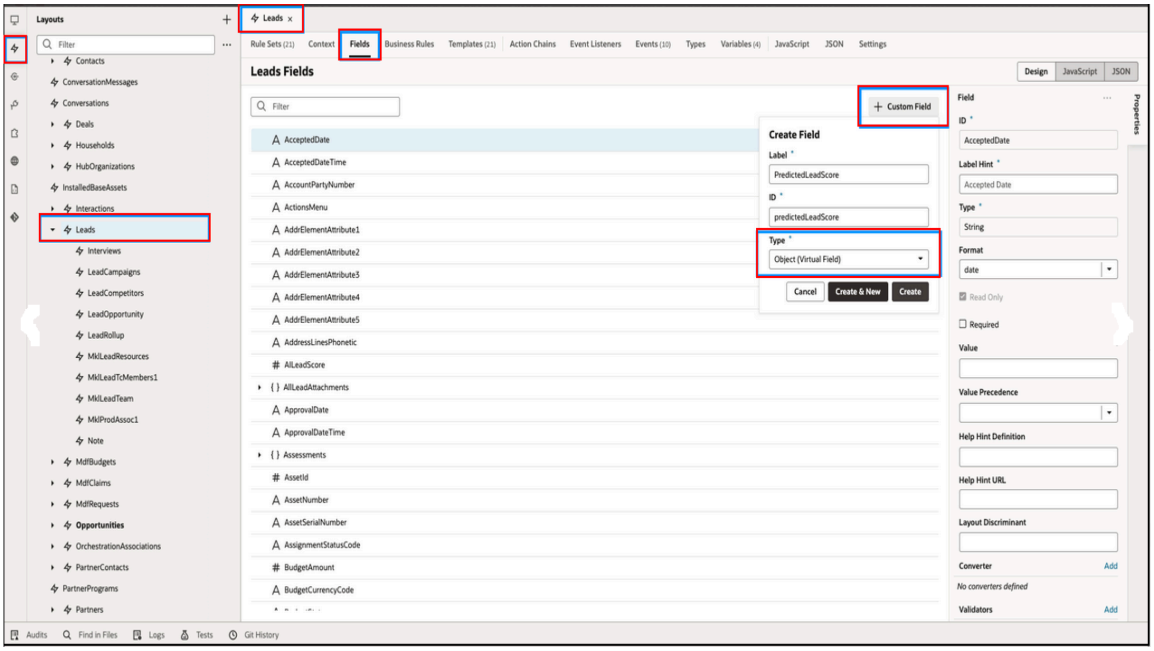
Explanations of machine learning predictions provides salespeople with added insight into the highest impacting factors that can help their sales organization improve sales efficiency and productivity. For example, salespeople can take appropriate actions to mitigate any outcomes that negatively impact the prediction. They can also use the insights from positive factors to devise a sales strategy that plays to their unique strengths.

Create a Virtual Field Using Visual Builder Studio (VBS)

Here's an example that shows how to create a virtual field for the Leads object:

1. Open Oracle Visual Builder Studio.
2. Click the **Layouts** tab and select **Leads** > **Fields**.

3. Click **Create Field** and because this field is expected to be a link, ensure that it follows the format as highlighted in the Type field as shown:



Create Field
Label * PredictedLeadScore
ID * predictedLeadScore
Type * Object (Virtual Field)

Field
ID * AcceptedDate
Label Hint * Accepted Date
Type * String
Format date
☑ Read Only
☐ Required
Value
Value Precedence
Help Hint Definition
Help Hint URL
Layout Discriminant
Converter Add
No converters defined
Validators Add

For example, enter the following details:

- **Label**: `PredictedLeadScore`
- **ID**: `predictedLeadScore`
- **Type**: Select Object (Virtual Field) from the drop-down list.

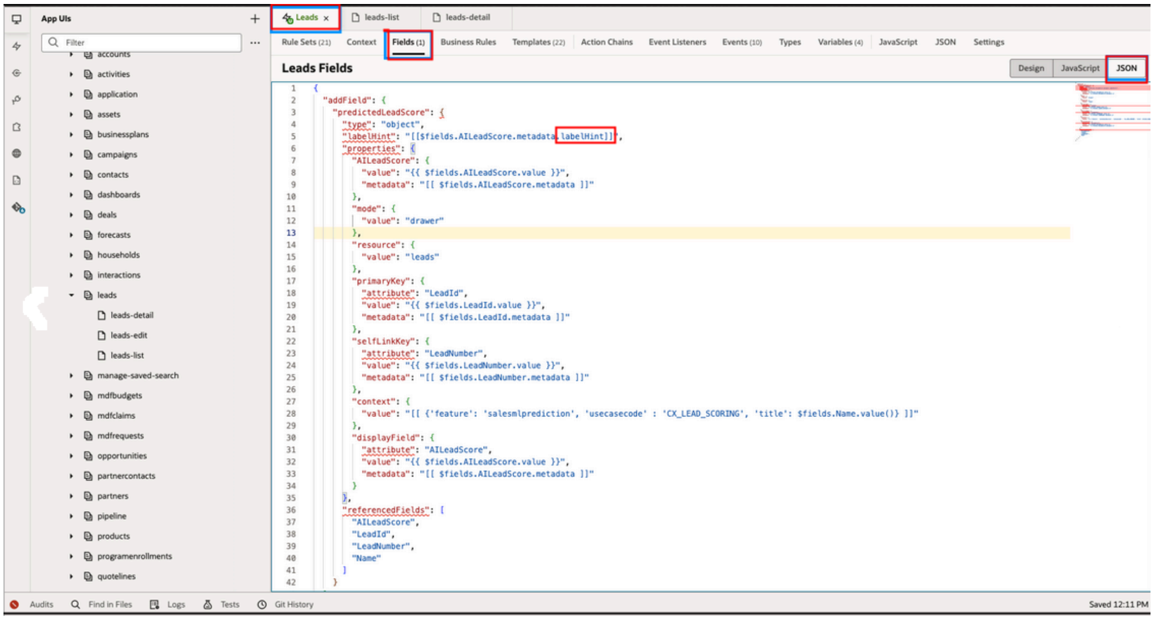
4. Click **Create**.

After creating the virtual field, you must manually edit its configuration to include the following attributes:

- `labelHint`
- `value`
- `displayField`
- `primaryKey`
- `selfLinkKey`
- `mode`

- `referencedFields`

1. Click the **JSON** tab as shown:



```
{
  "addField": {
    "predictedLeadScore": {
      "type": "object",
      "labelHint": "[[$fields.AILeadScore.metadata.labelHint]]",
      "properties": {
        "AILeadScore": {
          "value": "{{ $fields.AILeadScore.value }}",
          "metadata": "[[ $fields.AILeadScore.metadata ]]"
        },
        "mode": {
          "value": "drawer"
        },
        "resource": {
          "value": "leads"
        },
        "primaryKey": {
          "attribute": "LeadId",
          "value": "{{ $fields.LeadId.value }}",
          "metadata": "[[ $fields.LeadId.metadata ]]"
        },
        "selfLinkKey": {
          "attribute": "LeadNumber",
          "value": "{{ $fields.LeadNumber.value }}",
          "metadata": "[[ $fields.LeadNumber.metadata ]]"
        },
        "context": {
          "value": "[[ {'feature': 'salesmlprediction', 'usecasecode' : 'CX_LEAD_SCORING', 'title': $fields.Name.value()} ]]"
        },
        "displayField": {
          "attribute": "AILeadScore",
          "value": "{{ $fields.AILeadScore.value }}",
          "metadata": "[[ $fields.AILeadScore.metadata ]]"
        }
      },
      "referencedFields": [
        "AILeadScore",
        "LeadId",
        "LeadNumber",
        "Name"
      ]
    }
```

2. Find the virtual field entry created in the previous task.
3. Change the JSON to include the required attributes

Here are sample configurations for the **Opportunity Predicted Win Probability** and **Lead Scoring** predefined models. These examples assume that:

- The Predicted Win Probability model uses the `PredWinProb` attribute in the Opportunity object.
- The Lead Scoring model uses the `AILeadScore` attribute in the Lead object.

Sample Code Snippet for: Opportunity `PredWinProb` Model

```
"PredictedWinProb": {
 "type": "object",
 "labelHint": "[[$fields.PredWinProb.metadata.labelHint]]",
 "value" : "[[ $fields.PredWinProb.value ]]",
 "properties": {
 "resource": {
 "value": "opportunities"
 },
 "displayField": {
 "attribute": "PredWinProb",
 "value": "{{ $fields.PredWinProb.value }}",
 "metadata": "[[ $fields.PredWinProb.metadata ]]"
 },
 "primaryKey": {
 "attribute": "OptyId",
 "value": "{{ $fields.OptyId.value }}",
 "metadata": "[[ $fields.OptyId.metadata ]]"
 },
 "selfLinkKey": {
 "attribute": "OptyNumber",
 "value": "{{ $fields.OptyNumber.value }}",
 "metadata": "[[ $fields.OptyNumber.metadata ]]"
 },
 "mode": {
 "value" : "drawer"
 },
 "context" : {
 "value" : "[[ {'feature': 'salesmlprediction',
 'usecasecode' : 'CX_OPPORTUNITY_WIN_PROBABILITY',
 'title': $fields.Name.value()} ]]"
 }
 },
 "referencedFields": [
 "Name",
 "OptyId",
 "OptyNumber",
 "PredWinProb",
 "StatusCode"
 ]
}
```

Sample Code Snippet for: Lead Scoring Model

```
{
 "addField": {
 "predictedLeadScore": {
 "type": "object",
 "labelHint": "[[$fields.AILeadScore.metadata.labelHint]]",
 "value" : "[[ $fields.AILeadScore.value ]]",
 "properties": {
 "AILeadScore": {
 "value": "{{ $fields.AILeadScore.value }}",
 "metadata": "[[ $fields.AILeadScore.metadata ]]"
 },
 "mode": {
 "value": "drawer"
 },
 "resource": {
 "value": "leads"
 },
 "primaryKey": {
 "attribute": "LeadId",
 "value": "{{ $fields.LeadId.value }}",
 "metadata": "[[ $fields.LeadId.metadata ]]"
 },
 "selfLinkKey": {
 "attribute": "LeadNumber",
 "value": "{{ $fields.LeadNumber.value }}",
 "metadata": "[[ $fields.LeadNumber.metadata ]]"
 },
 "context": {
 "value": "[[ {'feature': 'salesmlprediction', 'usecasecode' :
 'CX_LEAD_SCORING', 'title': $fields.Name.value()} ]]"
 },
 "displayField": {
 "attribute": "AILeadScore",
 "value": "{{ $fields.AILeadScore.value }}",
 "metadata": "[[ $fields.AILeadScore.metadata ]]"
 }
 },
 "referencedFields": [
 "AILeadScore",
 "LeadId",
 "LeadNumber",
 "Name"
 ]
 }
 }
```

```
    }
```

When defining a virtual field, the following properties must be configured:

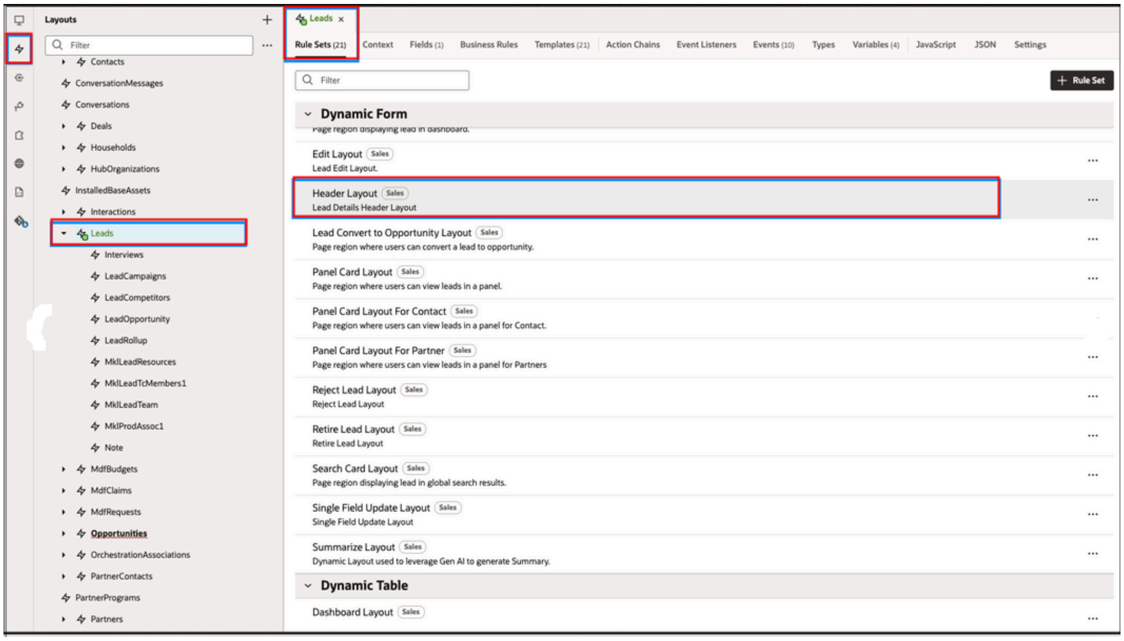
Name	Description
`labelHint`	Represents the label (field name) that will be displayed in the UI.
`value`	Holds the value of the field for that record (Field Value) that you want to expose.

`properties`	• `resource` Contains information about the object the field belongs to, for example: If the field belongs to an Opportunity object, resource will reference that.
	• `displayField` Specifies the field being exposed in the UI, for example `displayField`: `PredWinProb`.
	• `primaryKey` The primary identifier for the object. This value is fixed for each object type. For example `primaryKey": "OpportunityId"` for an Opportunity object.
	• `selfLinkKey` A fixed value used to reference the object's self-link. For example: `selfLinkKey : self .`
	• `mode` This value is fixed and indicates that the field should open a drawer when clicked. For example `mode : drawer`.
	• `context` Used for sending custom context information to the hyperlink listener and accepts a JSON object where you can pass multiple key-value pairs.
	• To ensure the drawer displays explanations for **SalesML predictions**, the `context` must contain three key-value pairs: as follows:
	○ `feature` – Must always be `'salesmlprediction'`. If this is missing, the explanation layout will not load.
	○ `usecasecode` – Represents the active model's use case code for fetching explanations via REST API.
	- This value can be conditional also.
	- Example If `StatusCode` is "OPEN", `usecasecode` is set to "OPEN_ALANTHO", otherwise, it defaults to "CX_OPPORTUNITY_WIN_PROBABILITY"
	`"context" : { "value" : "[[ {'feature': 'salesmlprediction',`
	`'usecasecode' : $fields.StatusCode.value() == 'OPEN' ? 'OPEN_ALANTHO' :`
	`'CX_OPPORTUNITY_WIN_PROBABILITY', 'title': $fields.Name.value()}`
	○ `title` – The heading of the drawer that opens. Typically set to the record name.
	• `referencedFields` Must include all distinct fields referenced in the configuration.

Exposing Your Virtual Field as a Link Field

Once you've created your virtual field, you need to edit the header layout to display this field in the UI as follows:

1. Navigate to the object's **Rule Sets** and select the **Header Layout** of the relevant object such as Lead.



Page region displaying lead in dashboard.

Edit Layout (Sales)
Lead Edit Layout.

Header Layout (Sales)
Lead Details Header Layout

Lead Convert to Opportunity Layout (Sales)
Page region where users can convert a lead to opportunity.

Panel Card Layout (Sales)
Page region where users can view leads in a panel.

Panel Card Layout For Contact (Sales)
Page region where users can view leads in a panel for Contact.

Panel Card Layout For Partner (Sales)
Page region where users can view leads in a panel for Partners

Reject Lead Layout (Sales)
Reject Lead Layout

Retire Lead Layout (Sales)
Retire Lead Layout

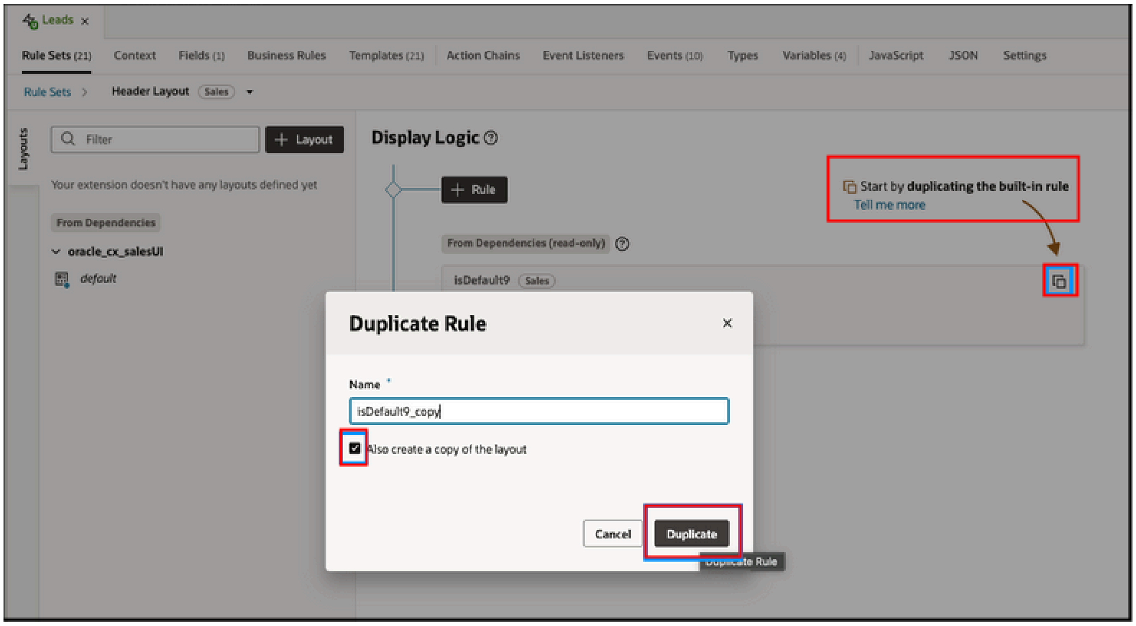
Search Card Layout (Sales)
Page region displaying lead in global search results.

Single Field Update Layout (Sales)
Single Field Update Layout

Summarize Layout (Sales)
Dynamic Layout used to leverage Gen AI to generate Summary.

Dashboard Layout (Sales)

2. Duplicate the rule and clone the layout as follows:



Your extension doesn't have any layouts defined yet

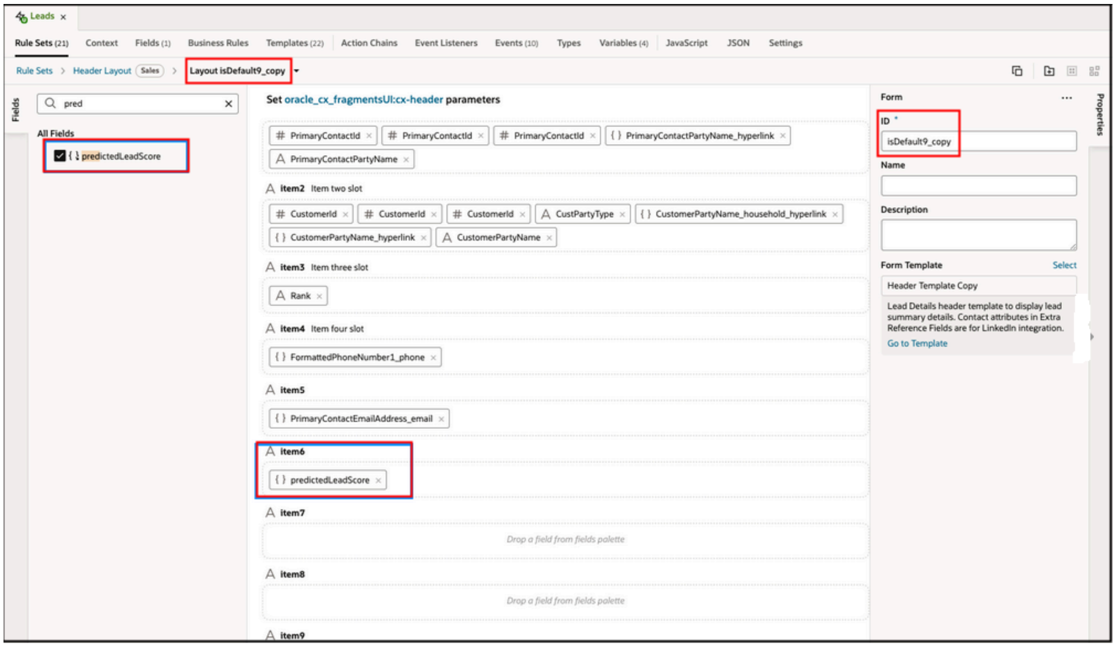
Start by **duplicating the built-in rule**
Tell me more

Duplicate Rule

Name *: isDefault9_copy

☑ Also create a copy of the layout

3. Add the virtual field and drag into any available slot on the layout header.



ID *: isDefault9_copy

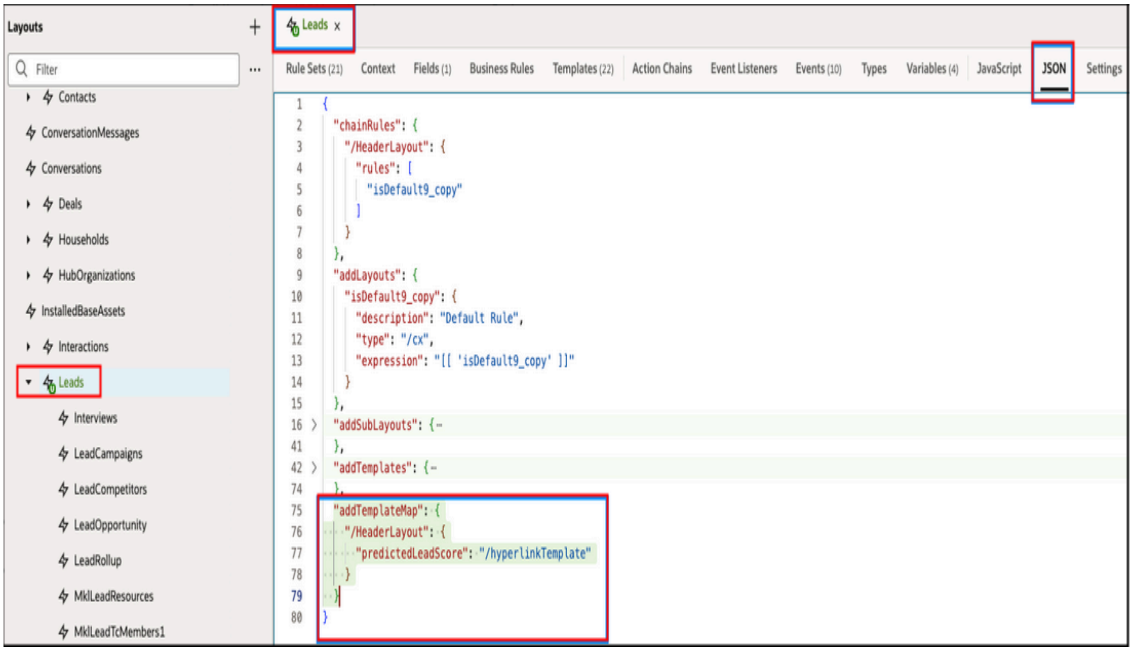
Form Template: Header Template Copy

Lead Details header template to display lead summary details. Contact attributes in Extra Reference Fields are for LinkedIn integration.

4. Map the virtual field to the Global Hyperlink Template

    - Ensure that the virtual field is mapped to the global hyperlink template, which is predefined and ready-to-use.

      The global hyperlink template ensures that clicking the virtual field opens a drawer with the explanations of the Machine Learning predictions.
    - Follow these steps to link to the hyperlink template:
        - Open the **JSON** tab of the layout configuration.
        - Scroll through the JSON and find `"addTemplates"` section as follows:



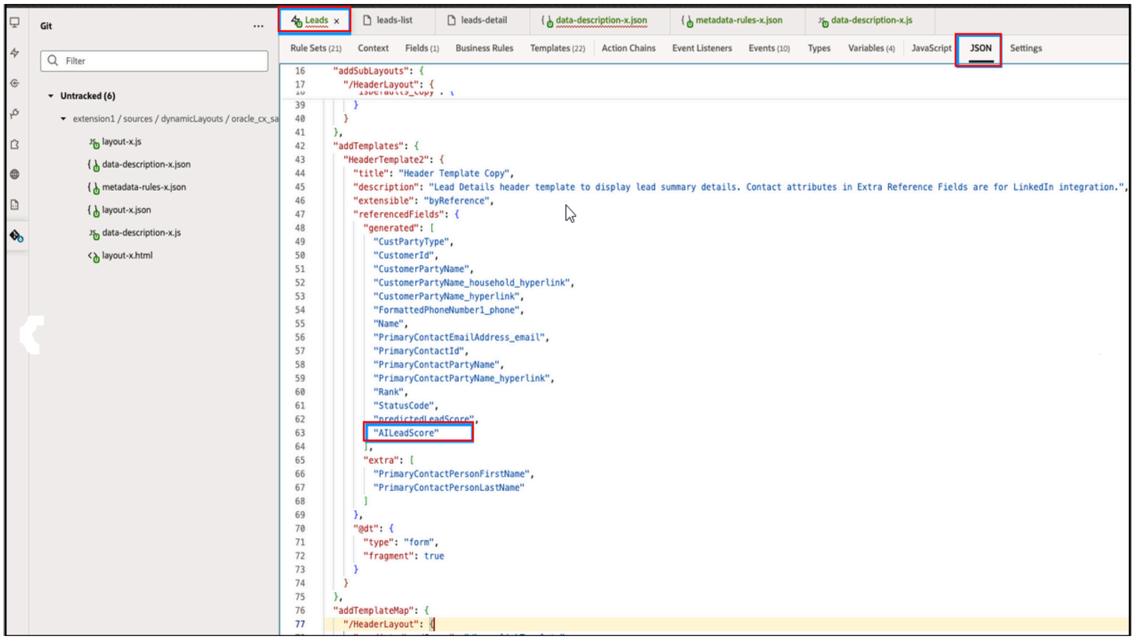
```
{
  "chainRules": {
    "/HeaderLayout": {
      "rules": [
        "isDefault9_copy"
      ]
    }
  },
  "addLayouts": {
    "isDefault9_copy": {
      "description": "Default Rule",
      "type": "/cx",
      "expression": "[[ 'isDefault9_copy' ]]"
    }
  },
  "addSubLayouts": {⋯
  },
  "addTemplates": {⋯
  },
  "addTemplateMap": {
    "/HeaderLayout": {
      "predictedLeadScore": "/hyperlinkTemplate"
    }
  }
}
```

- Add the following code to the `"addTemplates"` section:

> **Note:** Replace the virtual field ID in the code snippet with your Virtual Field ID.

-
```
"addTemplateMap": {
 "/HeaderLayout": {
 "predictedLeadScore":  "/hyperlinkTemplate"
 }
 }
```

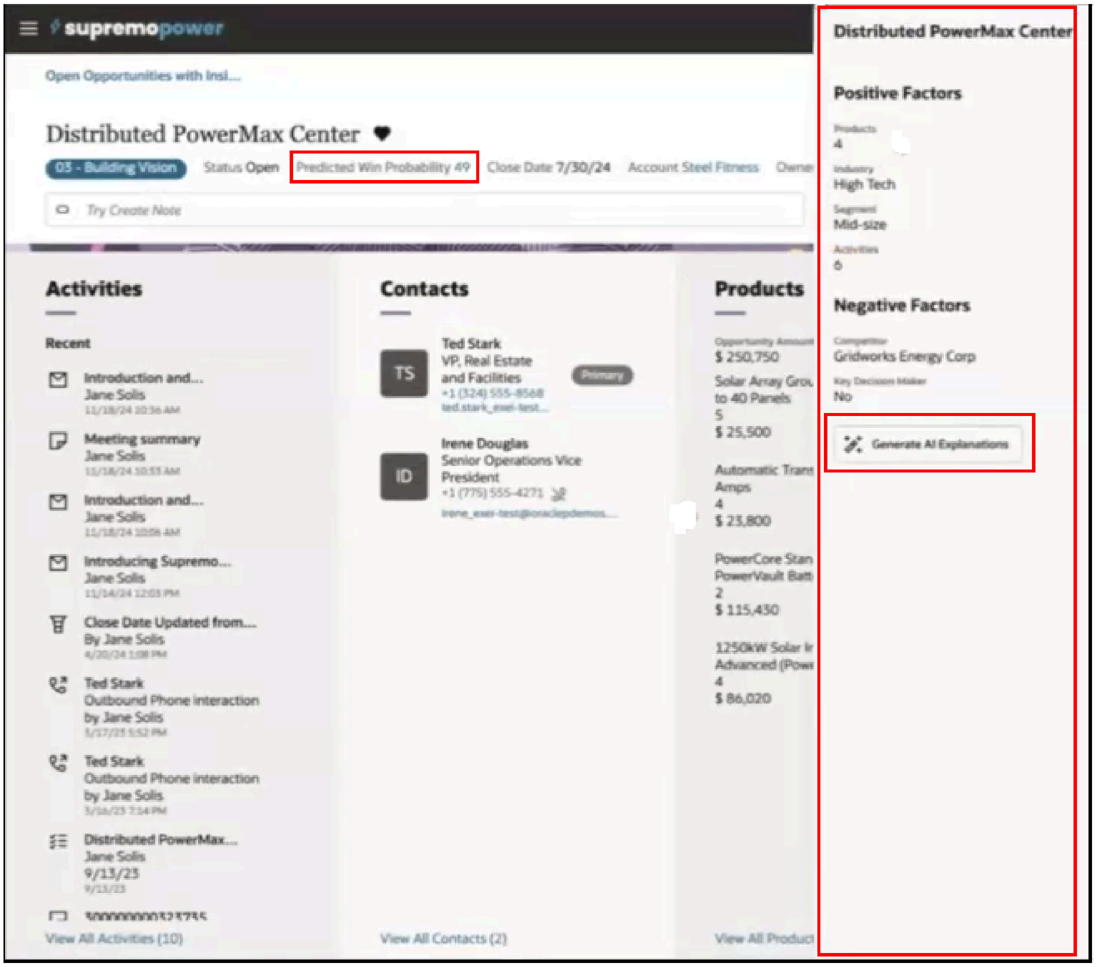
5. Add the attribute referenced as `"value"` in the Virtual Field to the cloned template.



```
"addSubLayouts": {
  "/HeaderLayout": {
    }
  }
},
"addTemplates": {
  "HeaderTemplate2": {
    "title": "Header Template Copy",
    "description": "Lead Details header template to display lead summary details. Contact attributes in Extra Reference Fields are for LinkedIn integration.",
    "extensible": "byReference",
    "referencedFields": {
      "generated": [
        "CustPartyType",
        "CustomerId",
        "CustomerPartyName",
        "CustomerPartyName_household_hyperlink",
        "CustomerPartyName_hyperlink",
        "FormattedPhoneNumber1_phone",
        "Name",
        "PrimaryContactEmailAddress_email",
        "PrimaryContactId",
        "PrimaryContactPartyName",
        "PrimaryContactPartyName_hyperlink",
        "Rank",
        "StatusCode",
        "predictedLeadScore",
        "AILeadScore"
      ],
      "extra": [
        "PrimaryContactPersonFirstName",
        "PrimaryContactPersonLastName"
      ]
    },
    "@dt": {
      "type": "form",
      "fragment": true
    }
  }
},
"addTemplateMap": {
  "/HeaderLayout": {
```

6. Test and publish your changes. When complete, click the virtual field link to open a drawer that displays the explanations of the Machine Learning predictions.

   Here's an example of what this will look like when the link is created and the panel displays.



Distributed PowerMax Center

03 - Building Vision · Status Open · Predicted Win Probability 49 · Close Date 7/30/24 · Account Steel Fitness · Owne

Activities

Recent

Introduction and...
Jane Solis
11/18/24 10:36 AM

Meeting summary
Jane Solis
11/18/24 10:33 AM

Introduction and...
Jane Solis
11/18/24 10:06 AM

Introducing Supremo...
Jane Solis
11/14/24 12:03 PM

Close Date Updated from...
By Jane Solis
4/20/24 1:08 PM

Ted Stark
Outbound Phone interaction
by Jane Solis
3/17/23 5:52 PM

Ted Stark
Outbound Phone interaction
by Jane Solis
3/16/23 7:14 PM

Distributed PowerMax...
Jane Solis
9/13/23
9/13/23

300000000323735

View All Activities (10)

Contacts

TS Ted Stark
VP, Real Estate and Facilities
Primary
+1 (324) 555-8568
ted.stark_exei-test...

ID Irene Douglas
Senior Operations Vice President
+1 (775) 555-4271
irene_exei-test@oraclepdemos...

View All Contacts (2)

Products

Opportunity Amount
$ 250,750

Solar Array Grou
to 40 Panels
5
$ 25,500

Automatic Trans
Amps
4
$ 23,800

PowerCore Stan
PowerVault Batt
2
$ 115,430

1250kW Solar Ir
Advanced (Powe
4
$ 86,020

View All Product

Distributed PowerMax Center

Positive Factors

Products
4

Industry
High Tech

Segment
Mid-size

Activities
6

Negative Factors

Competitor
Gridworks Energy Corp

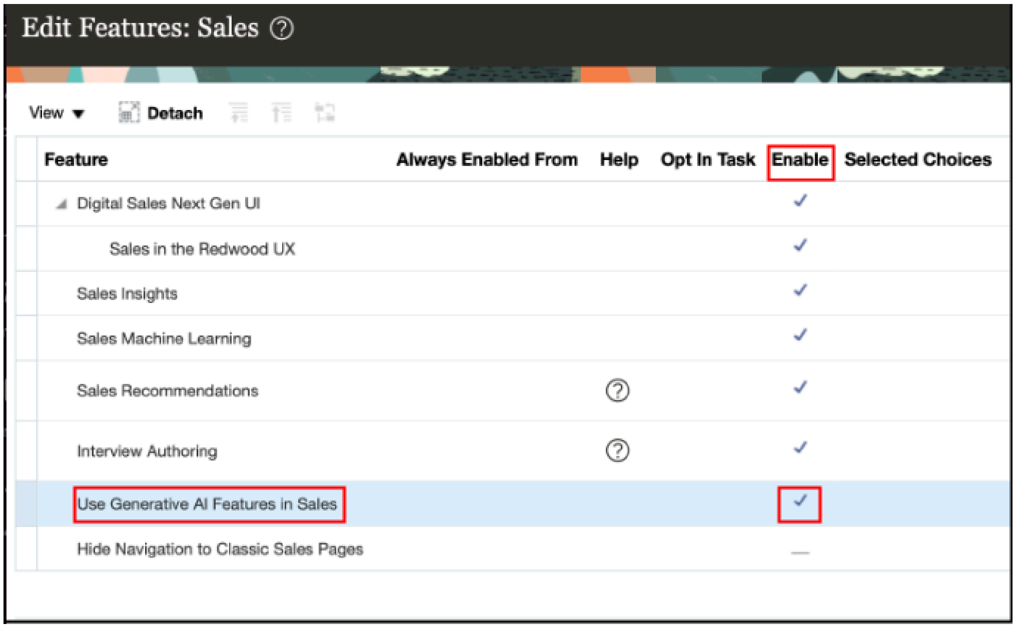
Key Decision Maker
No

Enable Generative AI Explanations for Sales

You can enable the Generative AI explanations for the Sales features provided by Oracle from the Setup and Maintenance work area. Here's how.

1. In Setup and Maintenance, open the **Enable Sales Intelligence** task:
   - Offering: Sales
   - Functional Area: Intelligence in Sales
   - Task: Enable Sales Intelligence
2. On the Enable Sales Intelligence Features page, Select Features panel, click **Start**.

3. Select the **Use Generative AI for Sales** checkbox that you want to enable as shown.

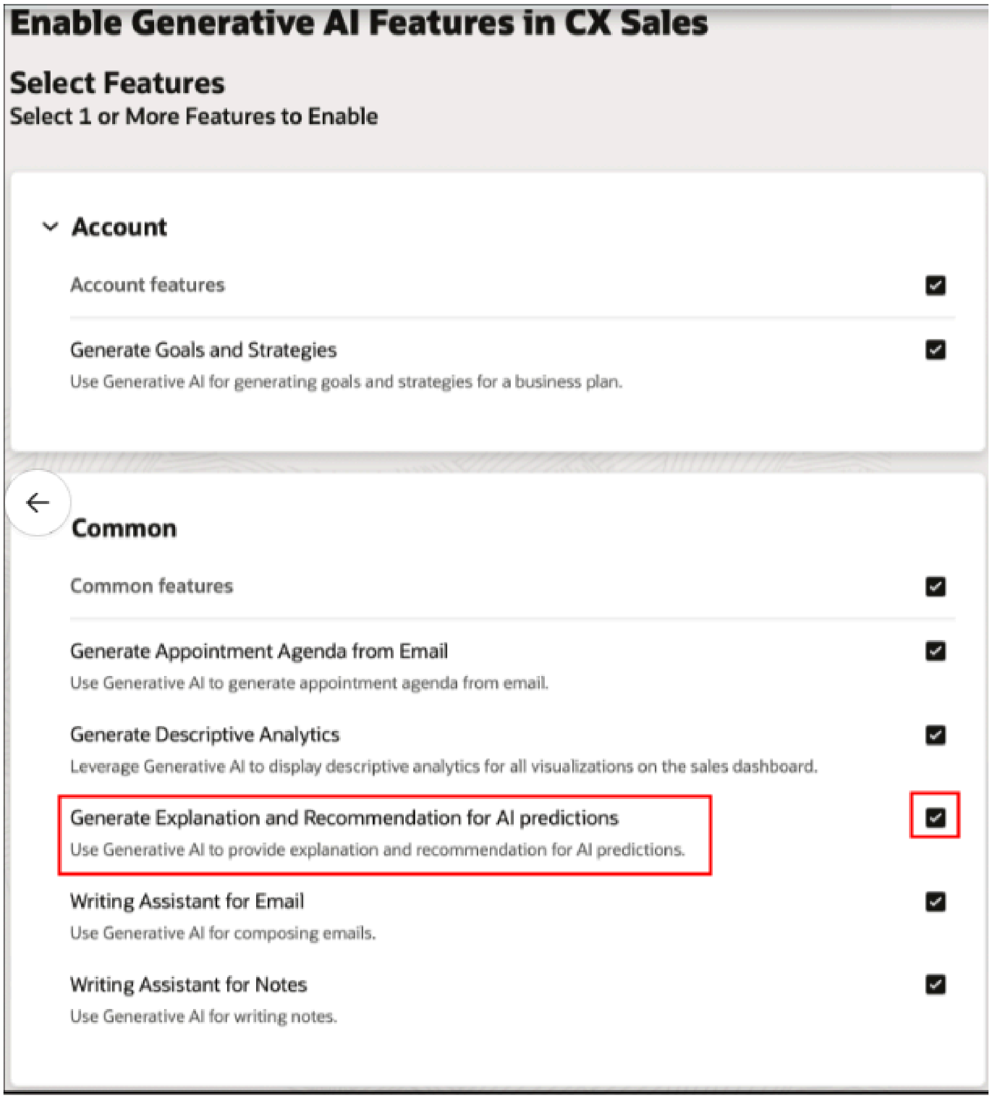


Edit Features: Sales

Feature	Always Enabled From	Help	Opt In Task	Enable	Selected Choices
Digital Sales Next Gen UI				✓	
Sales in the Redwood UX				✓	
Sales Insights				✓	
Sales Machine Learning				✓	
Sales Recommendations		?		✓	
Interview Authoring		?		✓	
Use Generative AI Features in Sales				✓	
Hide Navigation to Classic Sales Pages				—	

4. Click **Save**.
5. In the Configure Recommended Features page, you can click the links to open the administration UIs for technologies related to the features you enabled. Links are available for Machine Learning. There's no direct link for Sales Insights. You can also navigate to these UI pages separately.

6. From the **Enable Generative AI for Sales** page, select the **Generate Explanation and Explanation for AI predictions** checkbox as shown:



Enable Generative AI Features in CX Sales

Select Features

Select 1 or More Features to Enable

Account

Account features	☑
Generate Goals and Strategies Use Generative AI for generating goals and strategies for a business plan.	☑

Common

Common features	☑
Generate Appointment Agenda from Email Use Generative AI to generate appointment agenda from email.	☑
Generate Descriptive Analytics Leverage Generative AI to display descriptive analytics for all visualizations on the sales dashboard.	☑
Generate Explanation and Recommendation for AI predictions Use Generative AI to provide explanation and recommendation for AI predictions.	☑
Writing Assistant for Email Use Generative AI for composing emails.	☑
Writing Assistant for Notes Use Generative AI for writing notes.	☑

After this feature is enabled, the generative AI explanation button shows up in drawer panel enabled for explanations. See *How can I display explanations for Machine Learning predictions as a drawer in Oracle Sales?* for more information.

How can I display similar records identified by Machine Learning models in Oracle Sales?

As a sales administrator, you can display similar records identified by your own custom Machine Learning model in Oracle Sales by using a ready to use template model to display records details in a similar records fold-out panel.

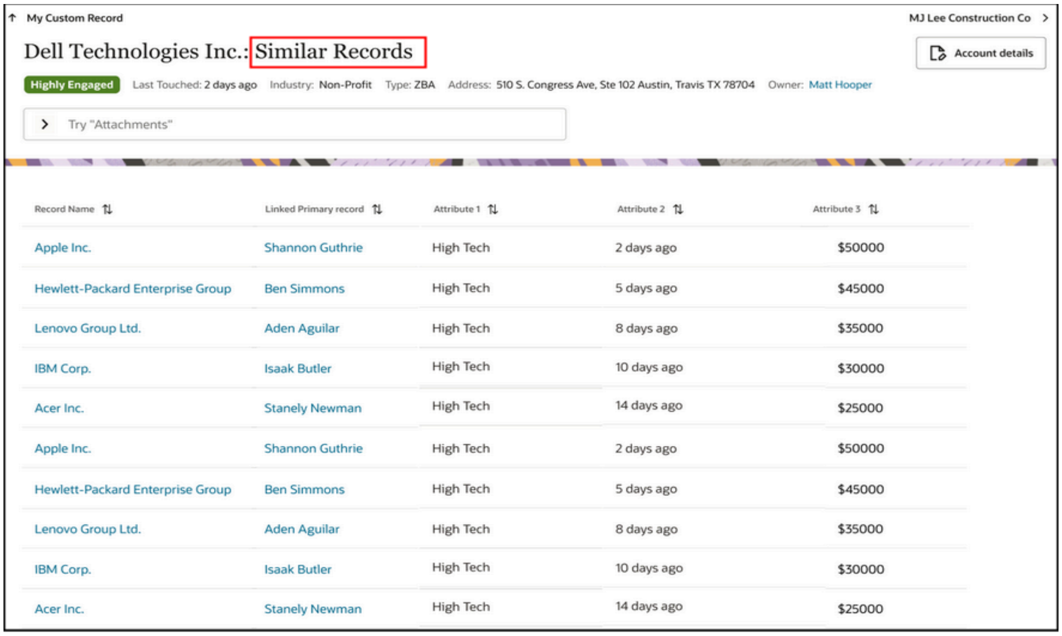
You can also add custom subview templates for any Sales Machine Learning use case predictions for both standard and custom objects using the following predefined configurations:

- Template `cx-subview`

  Template definition is available for accounts, or a new one can be created if needed. For other objects you need to create a new subview.

- Callback Function
  Fetches predicted record IDs.
- Dynamic Table Layout `subviewLayout`
  Defines which attributes of the predicted records to display.

Here's an example of a custom subview displaying similar records:



Dell Technologies Inc.: Similar Records

Highly Engaged   Last Touched: 2 days ago   Industry: Non-Profit   Type: ZBA   Address: 510 S. Congress Ave, Ste 102 Austin, Travis TX 78704   Owner: Matt Hooper

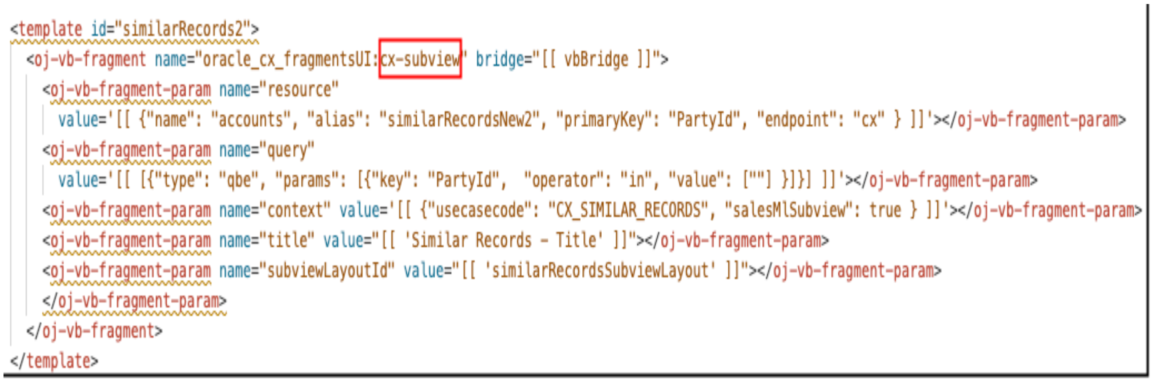
Record Name ⇅	Linked Primary record ⇅	Attribute 1 ⇅	Attribute 2 ⇅	Attribute 3 ⇅
Apple Inc.	Shannon Guthrie	High Tech	2 days ago	$50000
Hewlett-Packard Enterprise Group	Ben Simmons	High Tech	5 days ago	$45000
Lenovo Group Ltd.	Aden Aguilar	High Tech	8 days ago	$35000
IBM Corp.	Isaak Butler	High Tech	10 days ago	$30000
Acer Inc.	Stanely Newman	High Tech	14 days ago	$25000
Apple Inc.	Shannon Guthrie	High Tech	2 days ago	$50000
Hewlett-Packard Enterprise Group	Ben Simmons	High Tech	5 days ago	$45000
Lenovo Group Ltd.	Aden Aguilar	High Tech	8 days ago	$35000
IBM Corp.	Isaak Butler	High Tech	10 days ago	$30000
Acer Inc.	Stanely Newman	High Tech	14 days ago	$25000

Clone Existing Template to Display Similar Records

There are two approaches you can take:

1. Clone an existing template: This involves copying an existing subview template **Similar Records Template** and changing it to suit your needs. This can be faster if the existing template is already close to your requirements and can only be used for Accounts
2. Add a new template: You can create a new template from scratch, giving you full control over the structure and parameters from the beginning.

Regardless of the approach, the final template would look something like this:



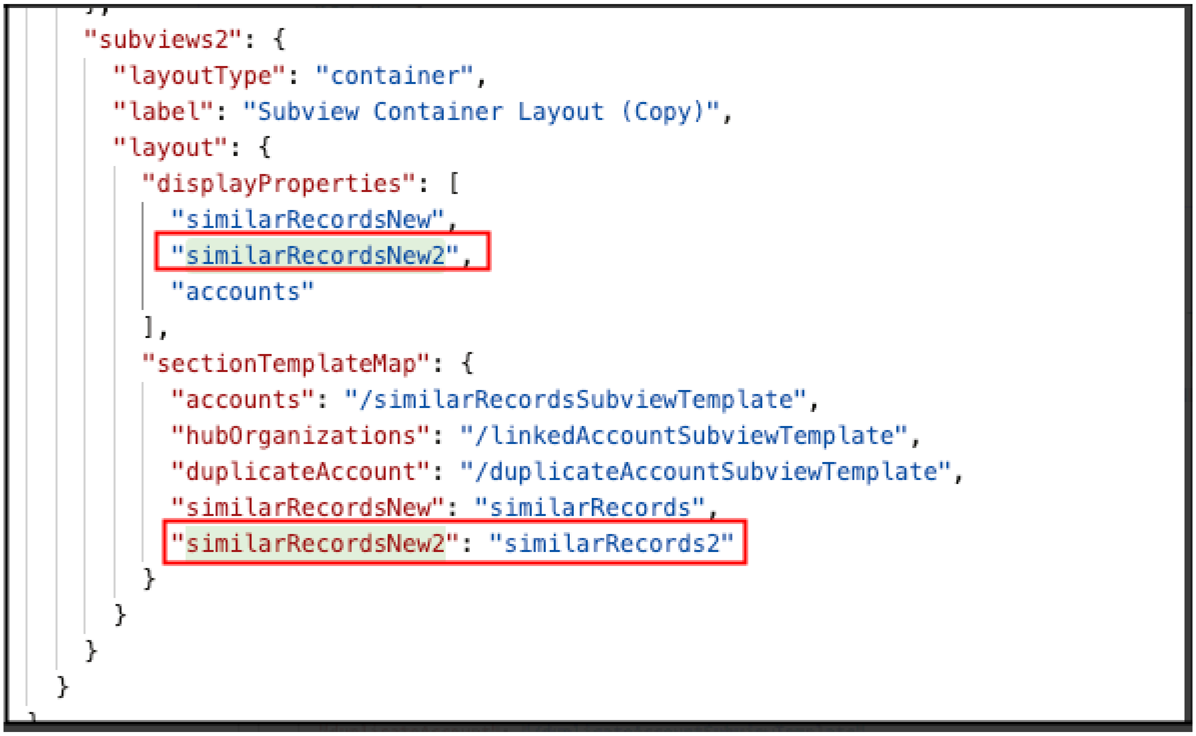
```
<template id="similarRecords2">
  <oj-vb-fragment name="oracle_cx_fragmentsUI:cx-subview" bridge="[[ vbBridge ]]">
    <oj-vb-fragment-param name="resource"
      value='[[ {"name": "accounts", "alias": "similarRecordsNew2", "primaryKey": "PartyId", "endpoint": "cx" } ]]'></oj-vb-fragment-param>
    <oj-vb-fragment-param name="query"
      value='[[ [{"type": "qbe", "params": [{"key": "PartyId",  "operator": "in", "value": [""] }]}] ]]'></oj-vb-fragment-param>
    <oj-vb-fragment-param name="context" value='[[ {"usecasecode": "CX_SIMILAR_RECORDS", "salesMlSubview": true } ]]'></oj-vb-fragment-param>
    <oj-vb-fragment-param name="title" value="[[ 'Similar Records - Title' ]]"></oj-vb-fragment-param>
    <oj-vb-fragment-param name="subviewLayoutId" value="[[ 'similarRecordsSubviewLayout' ]]"></oj-vb-fragment-param>
    </oj-vb-fragment-param>
  </oj-vb-fragment>
</template>
```

If creating a new template, the following properties must be configured:

Name	Description
`resource`	This is the name of the object you're working with (for example, Account, Lead, Custom Object). The alias ensures that your subview definition is unique because multiple sub views can use the same resource (object). The primary key should correspond to the object's identifier, such as `PartyId` for Account, `LeadId` for Lead, or `Id` for custom objects.
`query`	This should remain unchanged, except for the `key` which must be set to the object identifier such as `PartyId` for Account, `LeadId` for Lead, or `Id` for custom objects. The value will be empty, as it will be populated with the predicted record IDs. This is handled by the callback function, which fetches and assigns the recommended ID). **Note:** To use Elastic endpoint to call the record details you need to remove `provider: "adfRest"` in the `value` parameter and in `params` make sure that operator key exist.
`context`	The `usecasecode` should contain the use case ID of the model you want to display. The `salesMlSubview` parameter must always be set to true for any subview defined for a similar records template. This helps identify the subview and triggers the callback function to retrieve the predicted record ID(s).
`title`	The string value entered here will be used as the subheading when accessing the subview.
`subviewLayoutId`	It's best practice to add your applications extension ID and a slash (/) after as a prefix to your `subviewLayoutId.` For example `site_cxsales_Extension/similarRecordsSubviewLayout`

If you added the subview template manually in the template-page (rather than clone an existing subview), the JSON code block which exposes the subview template you created won't get generated. You must add it manually as follows:

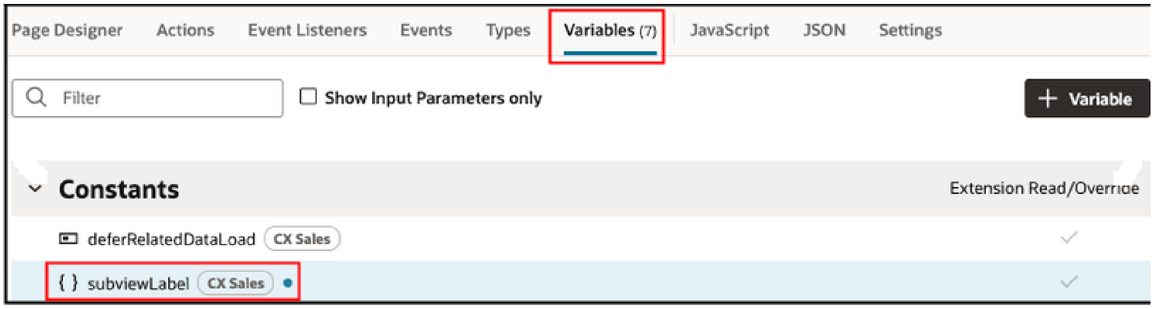
1. Add the following entry to the JSON file. You must also add the alias mentioned in the subview template as the key for exposing your subview template.



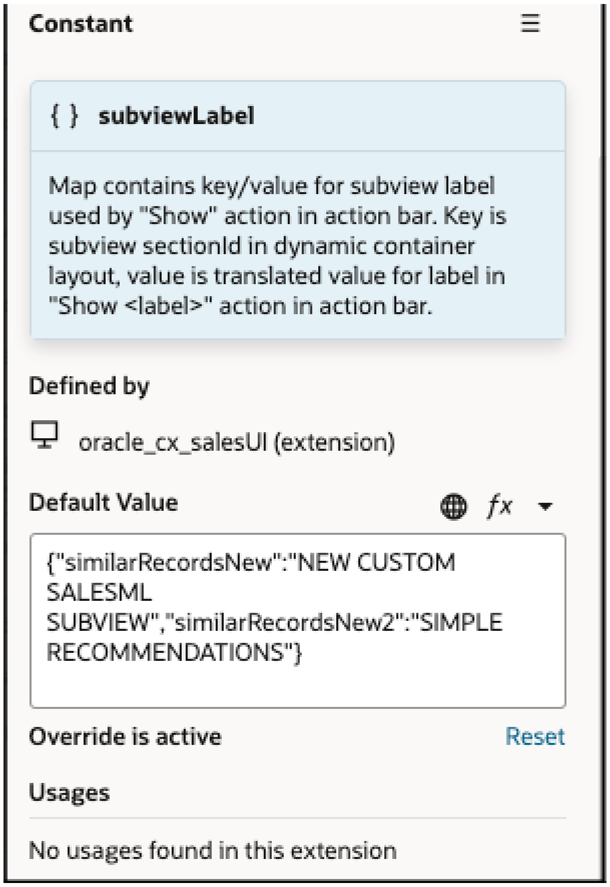
```
    "subviews2": {
      "layoutType": "container",
      "label": "Subview Container Layout (Copy)",
      "layout": {
        "displayProperties": [
          "similarRecordsNew",
          "similarRecordsNew2",
          "accounts"
        ],
        "sectionTemplateMap": {
          "accounts": "/similarRecordsSubviewTemplate",
          "hubOrganizations": "/linkedAccountSubviewTemplate",
          "duplicateAccount": "/duplicateAccountSubviewTemplate",
          "similarRecordsNew": "similarRecords",
          "similarRecordsNew2": "similarRecords2"
        }
      }
    }
  }
```

2. Add an entry as follows for the subview label variable.

   > **Note:** An existing variable called `subviewLabel` already exists for standard objects.



The entry in the subview label will need your key and value. Key would be the key used to expose your subview template. Value would decide the action name (prefixing with 'Show') that shows up in action bar.



Constant

{ } subviewLabel

Map contains key/value for subview label used by "Show" action in action bar. Key is subview sectionId in dynamic container layout, value is translated value for label in "Show <label>" action in action bar.

Defined by

oracle_cx_salesUI (extension)

Default Value

{"similarRecordsNew":"NEW CUSTOM SALESML SUBVIEW","similarRecordsNew2":"SIMPLE RECOMMENDATIONS"}

Override is active Reset

Usages

No usages found in this extension

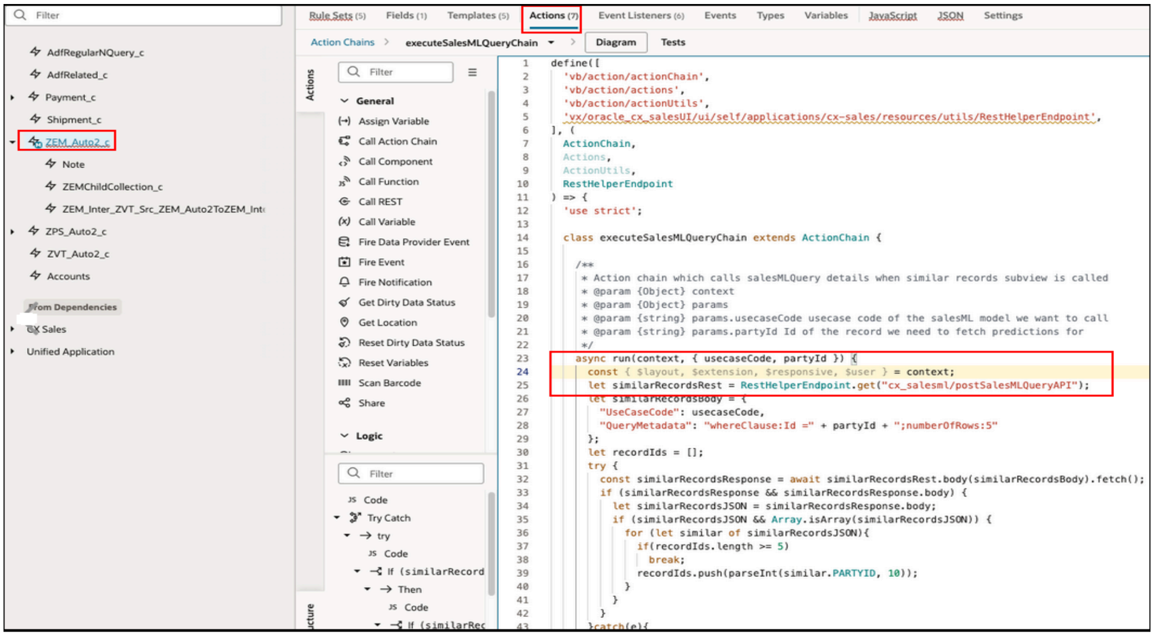
3. Create call back function. You should already have a `callbackhelper` registered before calling a subview.

   > **Note:** You can track the action chain this callback calls from the listener registered.

   You can update the action chain to call the machine learning specific chain to fetch the predicted records IDs and update the query parameter of the subview. Here's a sample of the code:

```
define([
 'vb/action/actionChain',
 'vb/action/actions',
 'vb/action/actionUtils',
], (
 ActionChain,
 Actions,
 ActionUtils
) => {
 'use strict';

 class onBeforeInvokeSubviewCallbackChain extends ActionChain {

 /**
 * @param {Object} context
 */
 async run(context, { event }) {
 const { $layout, $extension, $responsive, $user } = context;
 const { query, parentResource, parentRow, fragmentContext } = event;


 if (fragmentContext.salesMlSubview) {
 // Calling our subview specific action chain.
 const callChainExecuteSalesMLQueryChainResult = await Actions.callChain(context,
 {
 chain: 'executeSalesMLQueryChain',
 params: {
 usecaseCode: fragmentContext.usecasecode,
 partyId: parentRow.Id
 },
 });
 if(callChainExecuteSalesMLQueryChainResult.length > 0){
 query[0].params[0].value = callChainExecuteSalesMLQueryChainResult;
 }
 }

 return event.detail.done(query);
 }
 }

 return onBeforeInvokeSubviewCallbackChain;
});
```

4. Create an action chain which calls your endpoint and returns a list of predicted record IDs. This function should be defined from the dynamic layouts of the custom object (thunder sign) as shown:



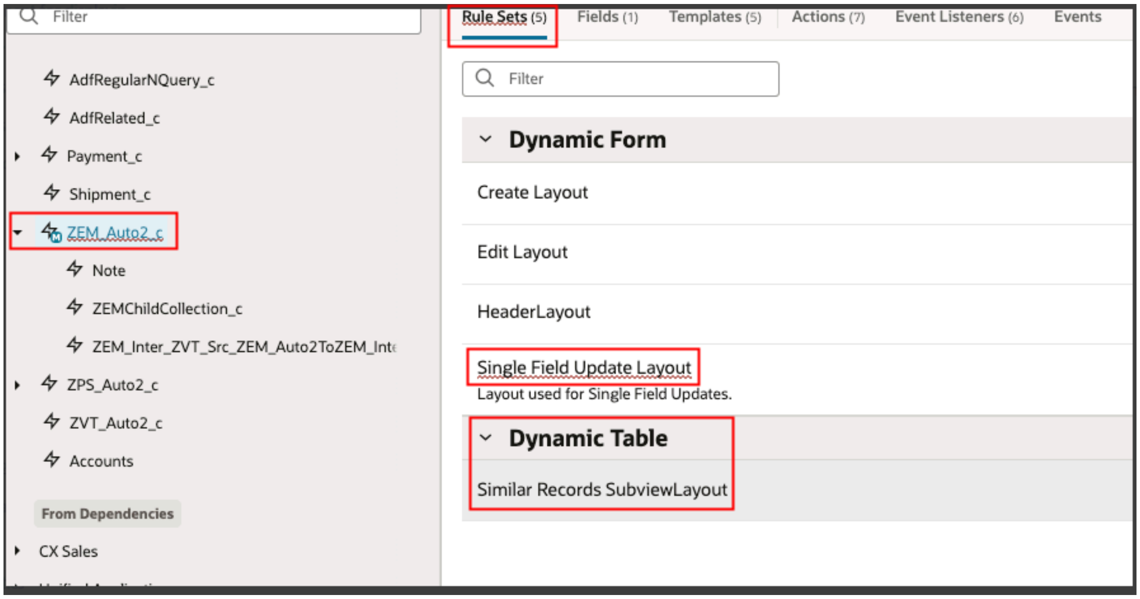
```
define([
  'vb/action/actionChain',
  'vb/action/actions',
  'vb/action/actionUtils',
  'vx/oracle_cx_salesUI/ui/self/applications/cx-sales/resources/utils/RestHelperEndpoint',
], (
  ActionChain,
  Actions,
  ActionUtils,
  RestHelperEndpoint
) => {
  'use strict';

  class executeSalesMLQueryChain extends ActionChain {

    /**
     * Action chain which calls salesMLQuery details when similar records subview is called
     * @param {Object} context
     * @param {Object} params
     * @param {string} params.usecaseCode usecase code of the salesML model we want to call
     * @param {string} params.partyId Id of the record we need to fetch predictions for
     */
    async run(context, { usecaseCode, partyId }) {
      const { $layout, $extension, $responsive, $user } = context;
      let similarRecordsRest = RestHelperEndpoint.get("cx_salesml/postSalesMLQueryAPI");
      let similarRecordsBody = {
        "UseCaseCode": usecaseCode,
        "QueryMetadata": "whereClause:Id =" + partyId + ";numberOfRows:5"
      };
      let recordIds = [];
      try {
        const similarRecordsResponse = await similarRecordsRest.body(similarRecordsBody).fetch();
        if (similarRecordsResponse && similarRecordsResponse.body) {
          let similarRecordsJSON = similarRecordsResponse.body;
          if (similarRecordsJSON && Array.isArray(similarRecordsJSON)) {
            for (let similar of similarRecordsJSON){
              if(recordIds.length >= 5)
                break;
              recordIds.push(parseInt(similar.PARTYID, 10));
            }
          }
        }
      }catch(e){
```

Here's a sample of the code:

```
define([
 'vb/action/actionChain',
 'vb/action/actions',
 'vb/action/actionUtils',
 'vx/oracle_cx_salesUI/ui/self/applications/cx-sales/resources/utils/RestHelperEndpoint',
], (
 ActionChain,
 Actions,
 ActionUtils,
 RestHelperEndpoint
) => {
 'use strict';

 class executeSalesMLQueryChain extends ActionChain {

 /**
 * Action chain which calls salesMLQuery details when similar records subview is called
 * @param {Object} context
 * @param {Object} params
 * @param {string} params.usecaseCode usecase code of the salesML model we want to call
 * @param {string} params.partyId Id of the record we need to fetch predictions for
 */
 async run(context, { usecaseCode, partyId }) {
 const { $layout, $extension, $responsive, $user } = context;
 let similarRecordsRest = RestHelperEndpoint.get("cx_salesml/postSalesMLQueryAPI");
 let similarRecordsBody = {
 "UseCaseCode": usecaseCode,
 "QueryMetadata": "whereClause:Id =" + partyId + ";numberOfRows:5"
 };
 let recordIds = [];
 try {
 const similarRecordsResponse = await similarRecordsRest.body(similarRecordsBody).fetch();
 if (similarRecordsResponse && similarRecordsResponse.body) {
 let similarRecordsJSON = similarRecordsResponse.body;
 if (similarRecordsJSON && Array.isArray(similarRecordsJSON)) {
```

```
 for (let similar of similarRecordsJSON){
 if(recordIds.length >= 5)
 break;
 recordIds.push(parseInt(similar.PARTYID, 10));
 }
 }
 }
 }catch(e){
 //Logger.info('similar-records-subview exception: ' + e);
 }
 return recordIds;
 }
 }

 return executeSalesMLQueryChain;
});
```

5. Create a Subview Layout as shown:



Display Similar Accounts on the Account Pages of Classic Sales

High-Level Setup Steps

Here's a summary of the setups you need to make to display Similar Accounts in the classic Sales account pages. No setup is required to make Similar Accounts display in Oracle Sales UIs.

1. Create a REST web service for Similar Records.
2. Create a REST web service for GET Account.
3. Create a global function for `invokeSalesMLSimilarRecordsQuery` API.
4. Use Formula fields to create a similar accounts field.
5. Add the similar accounts field to the Account landing page layout.

See Similar Accounts for an Account in Oracle Sales

If your administrator has enabled a Similar Accounts Machine Learning model, you can see similar accounts for an account in Oracle Sales.

Similar accounts are the accounts that have something in common with the main account you're looking at. The commonality could be in organization size, revenue, location, account score, or any other account attribute.

Your sales administrator decides the attributes when building a machine learning model for similar accounts. See *Create a Similar Accounts Model* to learn more about building the models.

Here's how you can see the similar accounts:

1. Navigate to **Redwood Sales** > **Accounts**.
2. On the Accounts landing page, click an account.

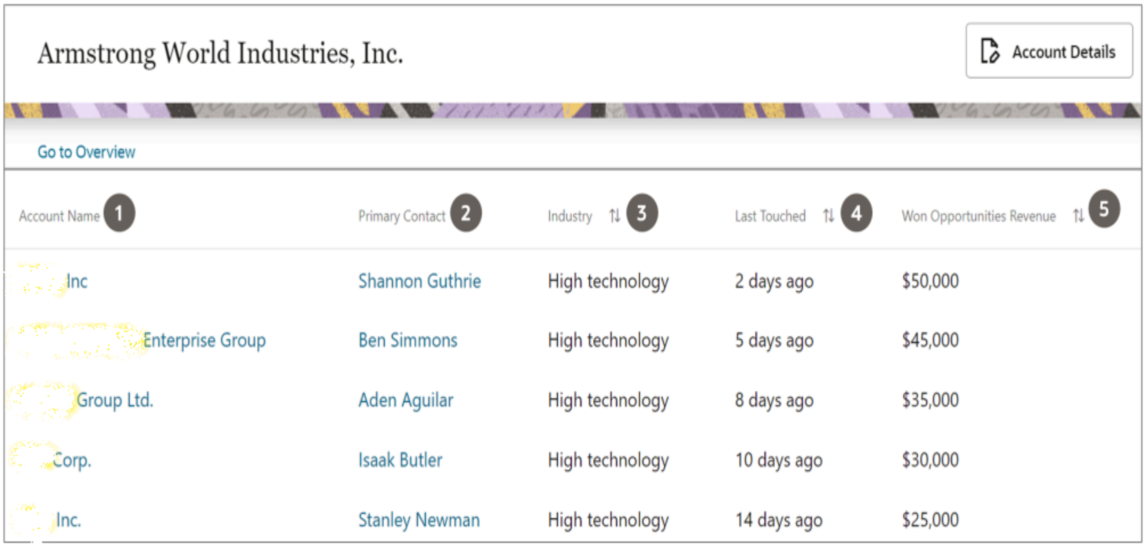
   The account foldout view appears.
3. In the Action Bar, type **Show Similar Accounts**.

   The similar accounts list appears.

Similar Accounts for an Account

Here's a screenshot that shows the similar accounts found by the Sales Machine Learning engine for a given account. Callouts from 1 to 5 in the screenshot highlight the columns that show details of the similar accounts. These columns are the attributes on which the model is built. Hence, you might not see the same columns every time.

If your administrator builds a new model with different attributes, the columns will change.



Armstrong World Industries, Inc.

Account Name 1	Primary Contact 2	Industry ⇅ 3	Last Touched ⇅ 4	Won Opportunities Revenue ⇅ 5
[redacted] Inc	Shannon Guthrie	High technology	2 days ago	$50,000
[redacted] Enterprise Group	Ben Simmons	High technology	5 days ago	$45,000
[redacted] Group Ltd.	Aden Aguilar	High technology	8 days ago	$35,000
[redacted] Corp.	Isaak Butler	High technology	10 days ago	$30,000
[redacted] Inc.	Stanley Newman	High technology	14 days ago	$25,000

The table here describes the columns highlighted in the sample similar accounts. This will give you an idea of the data you get from the model.

Callout	Column	Description

1	Account Name	Shows the name of the account that has some similarity with the given account. Click the account name to get to its foldout view and learn more about the account.
2	Primary Contact	Shows the primary contact associated with the account. Click the primary contact to get to the contact's foldout view and learn more about the contact.
3	Industry	This column appears if Industries is selected as one of the attributes for the model. Shows the industry of the account.
4	Last Touched	This column appears if Last Touched is selected as one of the attributes for the model. Shows the date or number of days when the account was last contacted either by phone or email. The calculation type for a date attribute can be in days, as an age bucket, number bucket, category, and so on. If your administrator used an age bucket, you'll see the last touched in the x days ago, format, as depicted in the screenshot above. Your administrator can show this in dates as well. For example, 02-04-2022 to represent Feb 4, 2022.
5	Won Opportunity Revenue	This column appears if Won Opportunity Revenue is selected as one of the attributes for the model. Shows the revenue from the opportunities that are in Won status.

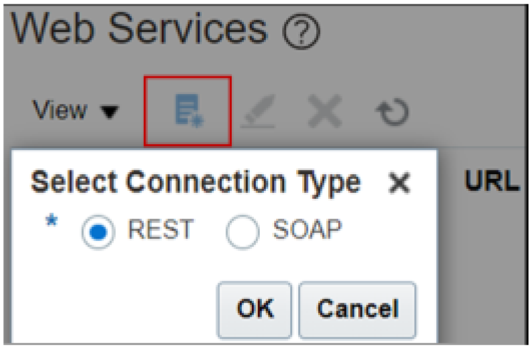
Related Topics

- Overview of Sales Machine Learning

Create a REST Web Service for Similar Records API

Here's how you create a REST Web Service for Similar Records API:

1. Enter an active sandbox. From the navigator, go to **Configurations** > **Application Composer** > **Common Setup** > **Web Services**.
2. On the Web Services page, click the new web service icon.
3. On the Select Connection Type window, select **REST** and click **OK**.



Select Connection Type

* ● REST ○ SOAP

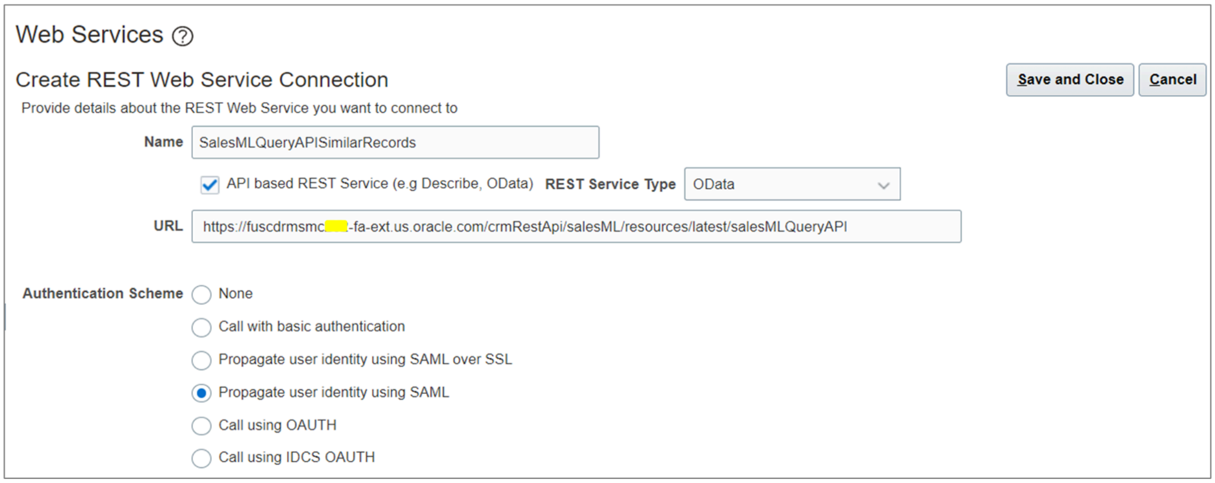
OK Cancel

4. On the Create REST Web Service Connection page, enter a **Connection Name**. For example, `SalesMLQueryAPISimilarRecords.`

   Ensure that you use same connection name in Global Function Groovy as well. See the **Create a Global Function for SalesMLQueryAPI Similar Records** section in this topic.
5. Leave the **API-based REST Service** checkbox blank.
6. Select **OData** for REST Service Type.
7. Enter a **URL** in the format, `https://<host>/crmRestApi/salesML/resources/latest/salesMLQueryAPI` For example, `https://fuscdrmsmcxxx-fa-ext.oracle.com/crmRestApi/salesML/resources/latest/salesMLQueryAP.`
8. Select any of these values for **Authentication Scheme**:
   - Call with Basic Authentication.

- Propagate user identity using SAML.
- Propagate user identity using SAML over SSL.

Here's a screenshot of the Create REST Web Service Connection page. The page shows the API name, REST service related settings selected, and a sample URL entered.



Web Services ?

Create REST Web Service Connection

Provide details about the REST Web Service you want to connect to

Name: SalesMLQueryAPISimilarRecords

☑ API based REST Service (e.g Describe, OData) REST Service Type: OData

URL: https://fuscdrmsmc[redacted]-fa-ext.us.oracle.com/crmRestApi/salesML/resources/latest/salesMLQueryAPI

Authentication Scheme:
○ None
○ Call with basic authentication
○ Propagate user identity using SAML over SSL
● Propagate user identity using SAML
○ Call using OAUTH
○ Call using IDCS OAUTH

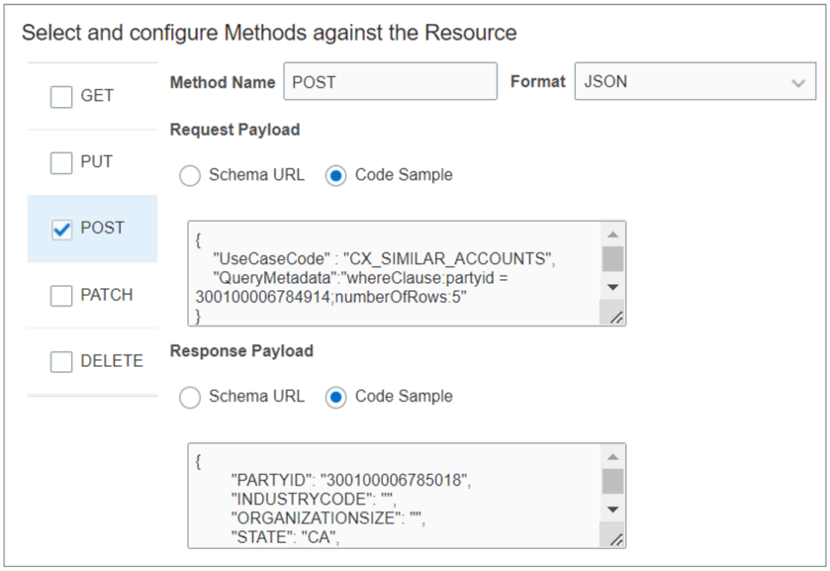
9. Next, enter information in the Select and Configure Methods against the Resource section:
   a. Select the **POST** checkbox.
   b. Let the **Method Name** be POST and **Format** be JSON.
   c. Select Code Sample for **Request Payload** and enter this code into the text field:

```
{
 "UseCaseCode" : "CX_SIMILAR_ACCOUNTS",
 "QueryMetadata":"whereClause:partyid = 300100006784914;numberOfRows:5"
}
```

   d. Select Code Sample for **Response Payload** and enter this code into the text field:

```
{
 "PARTYID": "300100006785018",
 "INDUSTRYCODE": "",
 "ORGANIZATIONSIZE": "",
 "STATE": "CA",
 "COUNTRY": "US",
 "POSTALCODE": "94065",
 "CITY": "Redwood Shores",
 "EMPLOYEESTOTAL": "1",
 "PROB": "0.8740334165546894",
 "CLUSTER_ID": "370",
 "RNK_CLUS2": "6"
```

```
        }
```



Select and configure Methods against the Resource

□ GET
□ PUT
☑ POST
□ PATCH
□ DELETE

Method Name: POST   Format: JSON

Request Payload

○ Schema URL ● Code Sample

```
{
   "UseCaseCode" : "CX_SIMILAR_ACCOUNTS",
   "QueryMetadata":"whereClause:partyid =
300100006784914;numberOfRows:5"
}
```

Response Payload

○ Schema URL ● Code Sample

```
{
     "PARTYID": "300100006785018",
     "INDUSTRYCODE": "",
     "ORGANIZATIONSIZE": "",
     "STATE": "CA",
```

10. Click **Save and Close**.

    The application creates a web service for the similar records API.

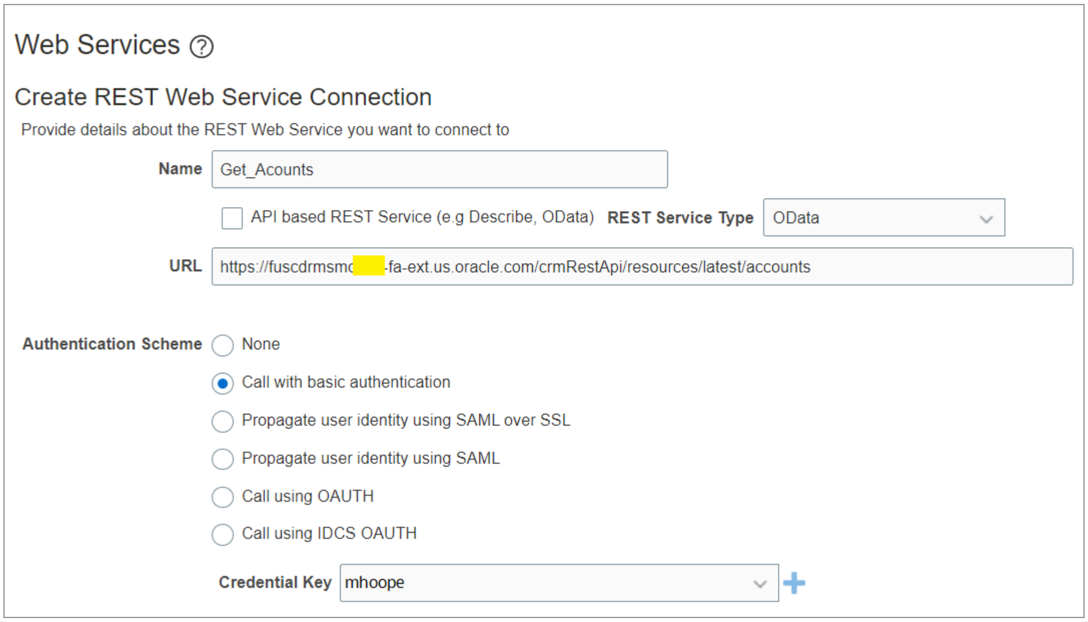
Create a REST Web Service for GET Account API

After creating the Similar Accounts REST web service, create another REST web service for GET Account API. Here's how you create it:

1. On the Web Services page, click the new web service icon.
2. On the Select Connection Type window, select REST and click **OK**.
3. On the Create REST Web Service Connection page, enter a **Connection Name**. For example, GET_Accounts.

   Ensure that you use same connection name in Global Function Groovy. See the Create a Global Function for SalesMLQueryAPI Similar Records section in this topic.
4. Leave the **API-based REST Service** checkbox blank.
5. Select **OData** for REST Service Type.
6. Enter a URL in the format, `https://<host>/crmRestApi/resources/latest/accounts`. For example, `https://fuscdrmsmcxxx-fa-ext.oracle.com/crmRestApi/resources/latest/accounts`.
7. Select Call with Basic Authentication for **Authentication Scheme**.

8. Add your **Credential Key**.



Web Services ⓘ

Create REST Web Service Connection

Provide details about the REST Web Service you want to connect to

Name: Get_Acounts

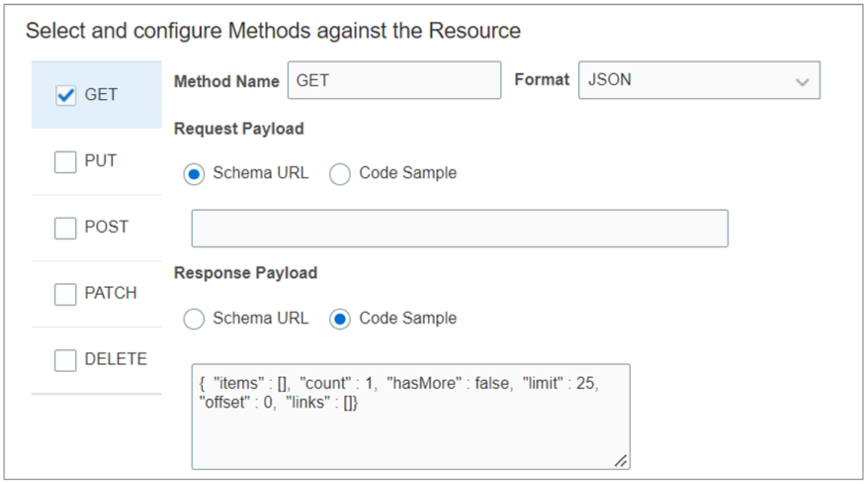
□ API based REST Service (e.g Describe, OData) **REST Service Type**: OData

URL: https://fuscdrmsmc[redacted]-fa-ext.us.oracle.com/crmRestApi/resources/latest/accounts

Authentication Scheme:
○ None
● Call with basic authentication
○ Propagate user identity using SAML over SSL
○ Propagate user identity using SAML
○ Call using OAUTH
○ Call using IDCS OAUTH

Credential Key: mhoope +

9. Next, enter information in the Select and Configure Methods against the Resource section:
   a. Select the **GET** checkbox.
   b. Let the **Method Name** be GET and **Format** be JSON.
   c. Leave **Request Payload** blank.
   d. Select Code Sample for **Response Payload** and enter the following code into the text field: `{ "items" : [], "count" : 1, "hasMore" : false, "limit" : 25, "offset" : 0, "links" : []}`



Select and configure Methods against the Resource

☑ GET
□ PUT
□ POST
□ PATCH
□ DELETE

Method Name: GET   Format: JSON

Request Payload
● Schema URL ○ Code Sample

Response Payload
○ Schema URL ● Code Sample

```
{ "items" : [], "count" : 1, "hasMore" : false, "limit" : 25,
"offset" : 0, "links" : []}
```

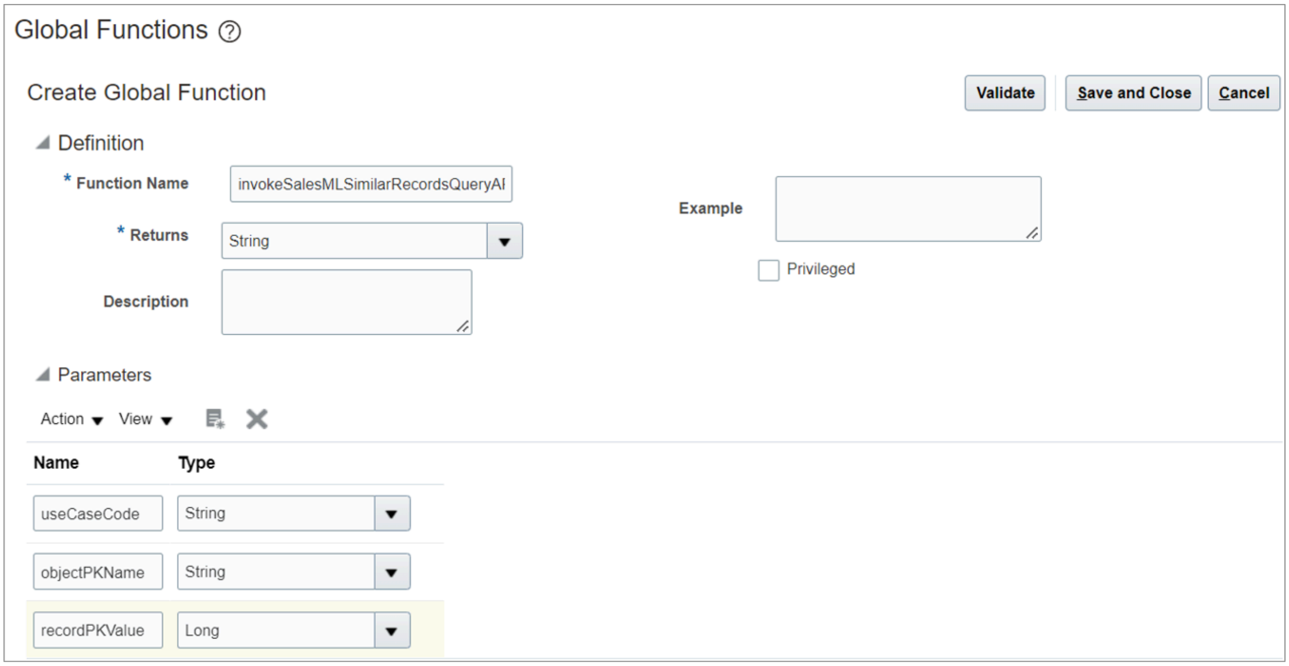
10. Save and close.

Create a Global Function for salesMLQuerySimilarRecords API

Here's how you create a global function for the `salesMLQuerySimilarRecords` API:

1. Continue in the active sandbox and navigate to **Configurations** > **Application Composer** > **Common Setup** > **Global Functions**.
2. Click the global function icon.
3. On the Create Global Function page, enter a **Function Name**. For example, `invokeSalesMLSimilarRecordsQueryAPI`.

   Ensure that you use same global function name in the Groovy for the formula type field.
4. Select String for **Returns**.
5. Add these parameters:
   - **Name**: `useCaseCode` **Type**: String.
   - **Name**: `objectPKName` **Type**: String.
   - **Name**: `recordPKValue` **Type**: Long (Integer).



Global Functions

Create Global Function

Definition

* Function Name	invokeSalesMLSimilarRecordsQueryAI
* Returns	String
Description	
Example	
□ Privileged	

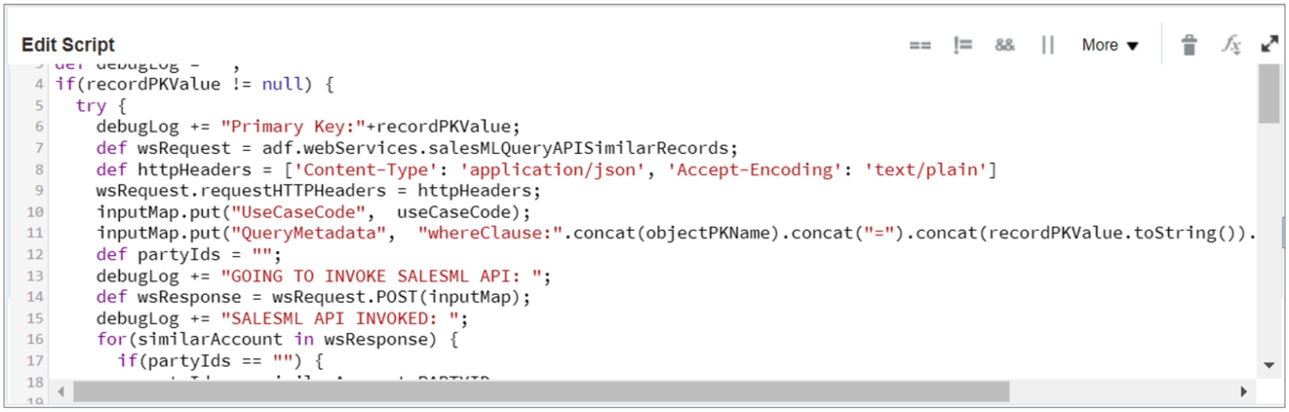
Parameters

Name	Type
useCaseCode	String
objectPKName	String
recordPKValue	Long

6. Add this Groovy code in the Edit Script field:

```
def inputMap = [:];
def accountNames = "";
def debugLog = "";
if(recordPKValue != null) {
 try {
 debugLog += "Primary Key:"+recordPKValue;
 def wsRequest = adf.webServices.salesMLQueryAPISimilarRecords;
 def httpHeaders = ['Content-Type': 'application/json', 'Accept-Encoding': 'text/plain']
 wsRequest.requestHTTPHeaders = httpHeaders;
 inputMap.put("UseCaseCode", useCaseCode);
 inputMap.put("QueryMetadata",
 "whereClause:".concat(objectPKName).concat("=").concat(recordPKValue.toString()).concat(";numberOfRows:5"));
 def partyIds = "";
 debugLog += "GOING TO INVOKE SALESML API: ";
 def wsResponse = wsRequest.POST(inputMap);
 debugLog += "SALESML API INVOKED: ";
 for(similarAccount in wsResponse) {
 if(partyIds == "") {
 partyIds += similarAccount.PARTYID;
 } else {
 partyIds += ","+similarAccount.PARTYID;
 }
 }
 if(partyIds != "") {
 debugLog += "START ACCOUNTS GET PREP: ";
 def wsRequest2 = adf.webServices.Acconts_GET;
 def httpHeaders2 = ['REST-Framework-Version': '2'];
 wsRequest2.requestHTTPHeaders = httpHeaders2;
 def queryParams = ['q':'PartyId IN ('+partyIds+')'];
 wsRequest2.dynamicQueryParams = queryParams;
 debugLog += "GOING TO INVOKE ACCOUNTS GET: ";
 def accounts = wsRequest2.GET();
 debugLog += "INVOKED ACCOUNTS GET: ";
 def count = 0;
 for(account in accounts.items) {
 if(count == 0) {
 accountNames += account.OrganizationName;
 } else {
 if(count<5){
 accountNames += ",\n"+account.OrganizationName;
 } else {
 break;
 }
 }
 count++;
 }
 }
 } catch (e) {
 debugLog += e.getMessage();
 accountNames = debugLog;
 }
} else {
 accountNames = "Not Available";
}

if(accountNames == "") {
 accountNames = "Not Available";
}

return accountNames;
```

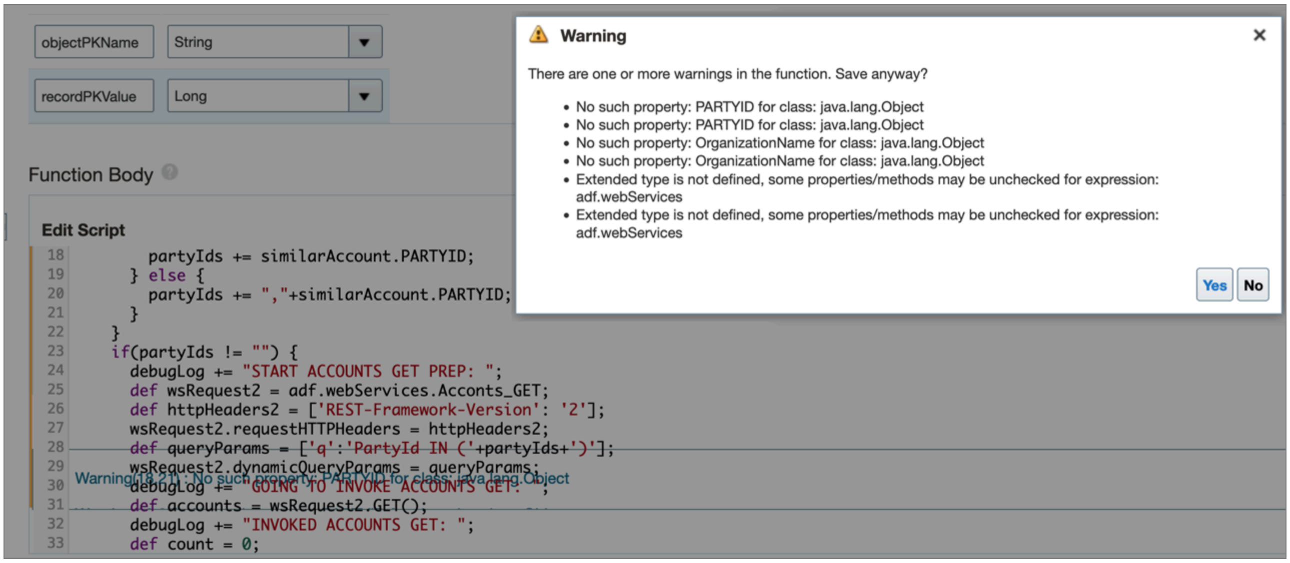
7. Click **Validate**.



```
if(recordPKValue != null) {
  try {
    debugLog += "Primary Key:"+recordPKValue;
    def wsRequest = adf.webServices.salesMLQueryAPISimilarRecords;
    def httpHeaders = ['Content-Type': 'application/json', 'Accept-Encoding': 'text/plain']
    wsRequest.requestHTTPHeaders = httpHeaders;
    inputMap.put("UseCaseCode",  useCaseCode);
    inputMap.put("QueryMetadata",  "whereClause:".concat(objectPKName).concat("=").concat(recordPKValue.toString()).
    def partyIds = "";
    debugLog += "GOING TO INVOKE SALESML API: ";
    def wsResponse = wsRequest.POST(inputMap);
    debugLog += "SALESML API INVOKED: ";
    for(similarAccount in wsResponse) {
      if(partyIds == "") {
```

8. Click **Save and Close**.

   The Groovy might show some warnings. Ignore those warnings and click **Yes** to save the script.



There are one or more warnings in the function. Save anyway?

- No such property: PARTYID for class: java.lang.Object
- No such property: PARTYID for class: java.lang.Object
- No such property: OrganizationName for class: java.lang.Object
- No such property: OrganizationName for class: java.lang.Object
- Extended type is not defined, some properties/methods may be unchecked for expression: adf.webServices
- Extended type is not defined, some properties/methods may be unchecked for expression: adf.webServices

```
          partyIds += similarAccount.PARTYID;
        } else {
          partyIds += ","+similarAccount.PARTYID;
        }
      }
      if(partyIds != "") {
        debugLog += "START ACCOUNTS GET PREP: ";
        def wsRequest2 = adf.webServices.Acconts_GET;
        def httpHeaders2 = ['REST-Framework-Version': '2'];
        wsRequest2.requestHTTPHeaders = httpHeaders2;
        def queryParams = ['q':'PartyId IN ('+partyIds+')'];
        wsRequest2.dynamicQueryParams = queryParams;
        debugLog += "GOING TO INVOKE ACCOUNTS GET: ";
        def accounts = wsRequest2.GET();
        debugLog += "INVOKED ACCOUNTS GET: ";
        def count = 0;
```

Warning(18,21) : No such property: PARTYID for class: java.lang.Object

Use a Formula Field to Render a Similar Accounts Field

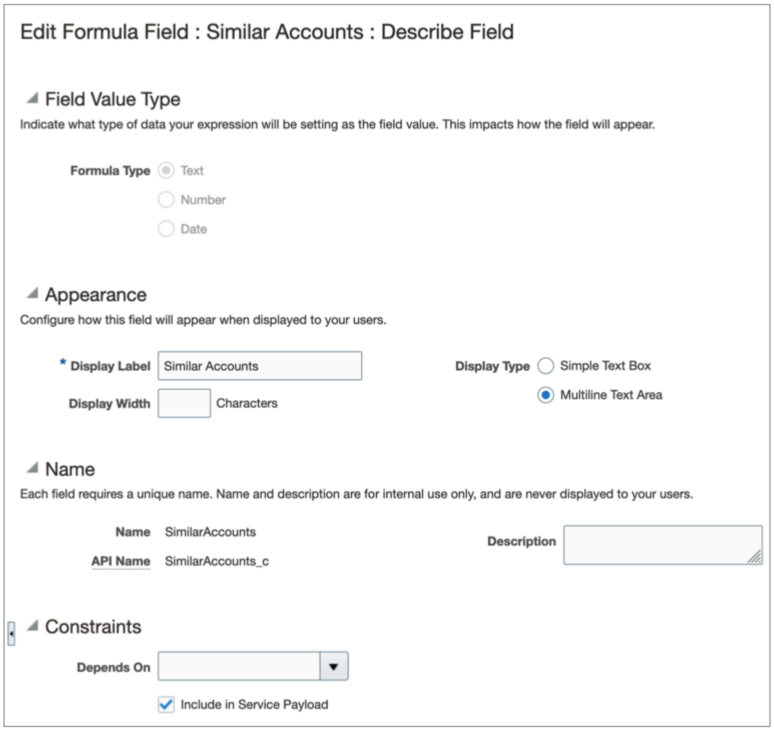
Create a multiline text formula field to show the similar accounts you get from the model. Here's how you create the field:

1. Enter an active sandbox.
2. Navigate to **Configuration** > **Application Composer** > **Standard Objects** > **Account** > **Fields**.
3. On the Fields page, click **Actions** > **Create**.
4. Select **Formula** and click **OK**.

   To learn more about the formula fields, read the *Formula Fields* topic in the Configuring Applications Using Application Composer guide.
5. On the Create Formula Field page, enter a **Display Label**. Enter the same name but without spaces, in the **Name** field.

6. Select **Display Type** as Multiline Text Area.

   The formula field will appear as a multiline text area to display the similar accounts.



Edit Formula Field : Similar Accounts : Describe Field

Field Value Type

Indicate what type of data your expression will be setting as the field value. This impacts how the field will appear.

Formula Type ● Text ○ Number ○ Date

Appearance

Configure how this field will appear when displayed to your users.

* Display Label: Similar Accounts

Display Width: Characters

Display Type: ○ Simple Text Box ● Multiline Text Area

Name

Each field requires a unique name. Name and description are for internal use only, and are never displayed to your users.

Name: SimilarAccounts

API Name: SimilarAccounts_c

Description:

Constraints

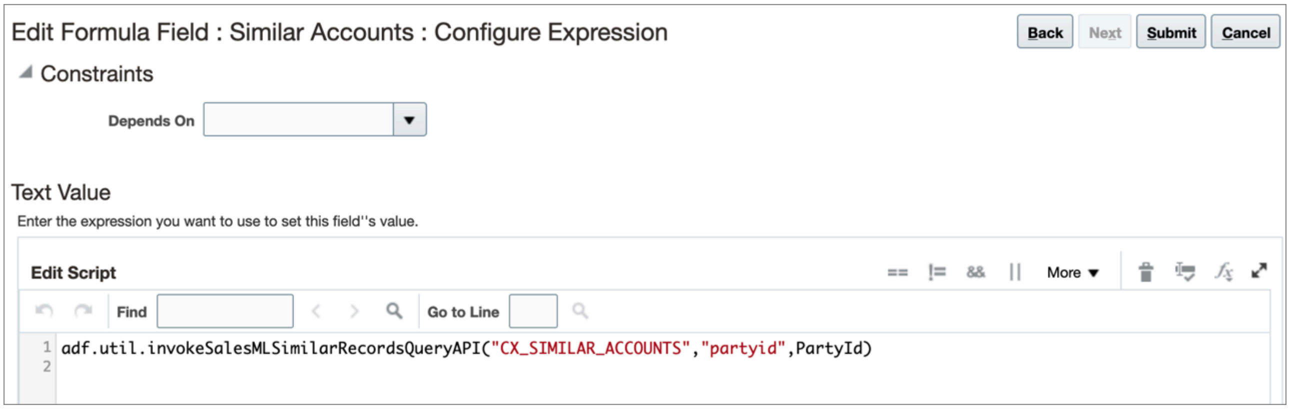
Depends On:

☑ Include in Service Payload

7. Click **Next**.
8. On the Configure Expression page, enter the formula value in the following format:

```
adf.util.InvokeSalesMLQueryAPI("<Prediction Model Usecase
 Code>","<Primary Key Field Name>",<Primary
 Field Value>)
```

For example, `adf.util.invokeSalesMLSimilarRecordsQueryAPI("CX_SIMILAR_ACCOUNTS","partyid",PartyId).`



Edit Formula Field : Similar Accounts : Configure Expression

Constraints

Depends On:

Text Value

Enter the expression you want to use to set this field''s value.

```
adf.util.invokeSalesMLSimilarRecordsQueryAPI("CX_SIMILAR_ACCOUNTS","partyid",PartyId)
```

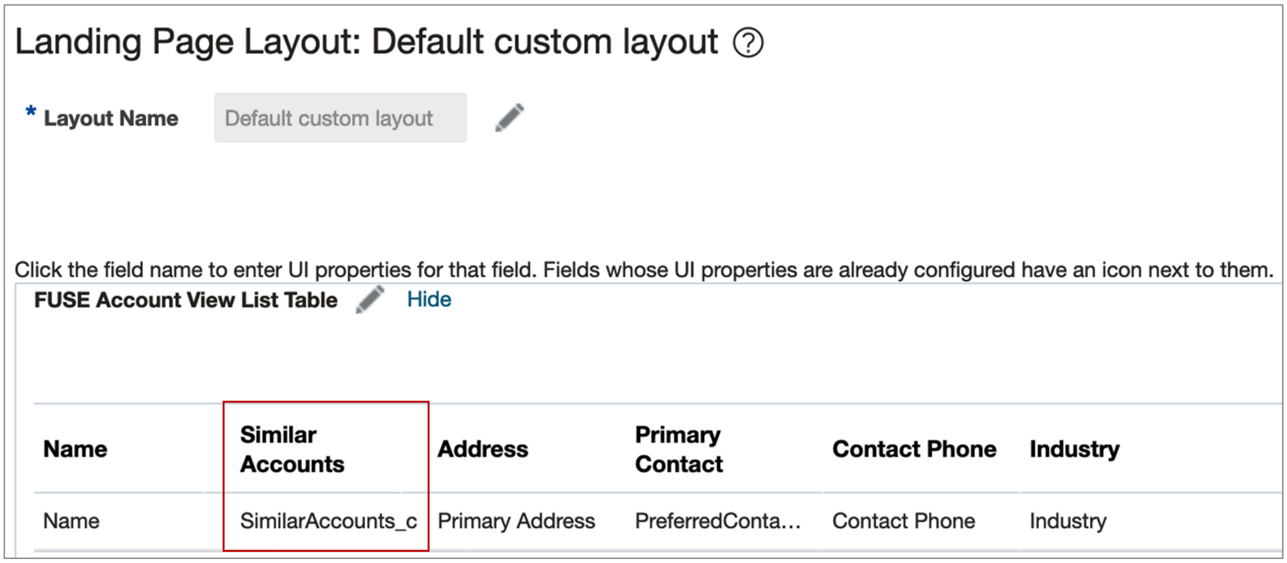
9. Click **Submit**.

   You can now add the similar account field to an object page.

Add the Field to the Landing Page Layout

Add the multiline text formula field you created, to the landing page so that similar account results render in that field. Here's how you add:

1. Enter an active sandbox.
2. Navigate to **Configuration** > **Application Composer** > **Standard Objects** > **Account** > **Pages** > **Application Pages** > **Landing Page Layouts**.
3. Create a copy of the Landing Page layout and add the Similar Accounts formula field you created.
4. Save your changes.



Landing Page Layout: Default custom layout

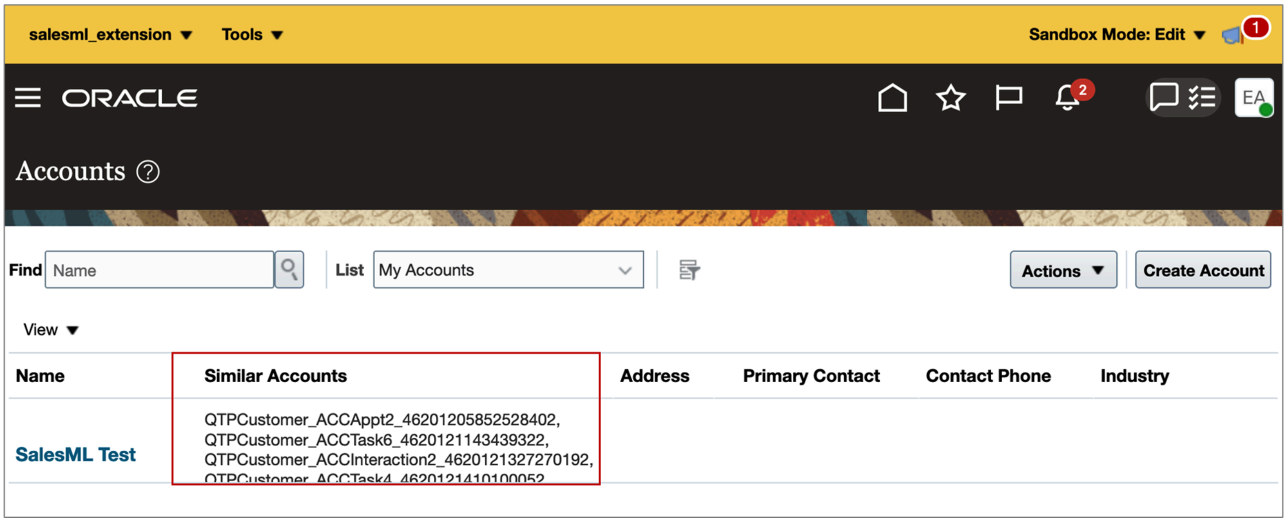
* Layout Name: Default custom layout

Click the field name to enter UI properties for that field. Fields whose UI properties are already configured have an icon next to them.

FUSE Account View List Table

Name	Similar Accounts	Address	Primary Contact	Contact Phone	Industry
Name	SimilarAccounts_c	Primary Address	PreferredConta...	Contact Phone	Industry

At the end of the configuration, you can see the similar accounts showing up on the Accounts page.



Name	Similar Accounts	Address	Primary Contact	Contact Phone	Industry
SalesML Test	QTPCustomer_ACCAppt2_46201205852528402, QTPCustomer_ACCTask6_4620121143439322, QTPCustomer_ACCInteraction2_4620121327270192, QTPCustomer_ACCTask4_4620121410100052				

Sales Insights

Overview of Sales Insights

Sales Insights help provide an objective understanding to salespeople working with accounts, leads, and opportunities.

Based on detailed analysis of account interaction and historical exchanges such as sales activities, status, duration, contacts, and so on, salespeople can gain key insights and take appropriate corrective actions on records to further increase their sales pipeline.

Identifying those accounts, leads, and opportunities that are either highly or moderately engaged and by observing certain accounts helps salespeople:

- Prioritize the next set of actions for records and tasks
- Improve productivity by moving the sales pipeline along
- Identify quickly what's wrong with a lead or opportunity and take the necessary corrective action

As sales insights evolve over time, context-based recommendations are made available to salespeople to help them evaluate and decide on what actions to take on appropriate next best lead and opportunity records or tasks for an account.

Sales data is mined, evaluated and analyzed through data analytical and statistical modules. Adaptive search finds key metrics, calculate standard deviations, averages, and margins to deliver insights. Standard query language programs sorts and classifies the data so that key sales metrics are readily available to salespeople to help them focus on those records that need attention.

Access Sales Insights from Classic Sales

Here are the UI paths for salespeople to access Sales Insight details from the classic Sales UI:

- Accounts: Use the Profile tab on the Accounts page

- Leads: Use the Summary tab on the Edit Lead page
- Opportunities: Use the Summary tab on the Edit Opportunity page

Access Sales Insights from Oracle Sales

To access sales insights from Oracle Sales in the Redwood User Experience, you must first enable the sales recommendation feature to show recommendations based on sales insights. Here are the steps:

1. Open the **Setup and Maintenance** work area and select the Sales offering.
2. In Setup: Sales, click the **Change Feature Opt In** link.
3. In the Opt In: Sales page, click **Features** for Sales.
4. Select **Enable** for **Sales Recommendations**.
5. Click **Done**.

Enable Sales Insights

Sales Insights data is mined, evaluated, and analyzed through data analytical and statistical modules. Adaptive Search finds key metrics and calculates standard deviations, averages, and margins to deliver insights.

You must have the following privileges to work with and run Sales Insights processes:

- ZCA_VIEW_SALES_INSIGHTS_PRIV
- ZCA_SALES_INSIGHTS_JOB_PRIV
- FUSION_APPS_CRM_ESS_APPID

You can also enable sales recommendations for Sales Insights for the Redwood User Experience. The sales recommendation feature analyzes text of notes and call logs and recommends follow-up tasks and appointments. When enabled, this feature replaces the automatic creation of a task after call wrap ups.

You enable the feature using the opt-in task. If you've permission to configure offerings, then you can use the New Features page to opt in to Sales Insights feature:

1. Click **Navigator** > **My Enterprise** > **Offerings** work area.
2. On the Offerings page, select the Sales offering.
3. Click **Change Feature Opt In**.
4. For the Sales root node, click the edit link (pencil icon).
5. From the Edit Features page, find **Sales Insights** and select the checkbox in the **Enable** column.
6. On the Opt In page, click **Done** to save the changes and return to the Offerings page.

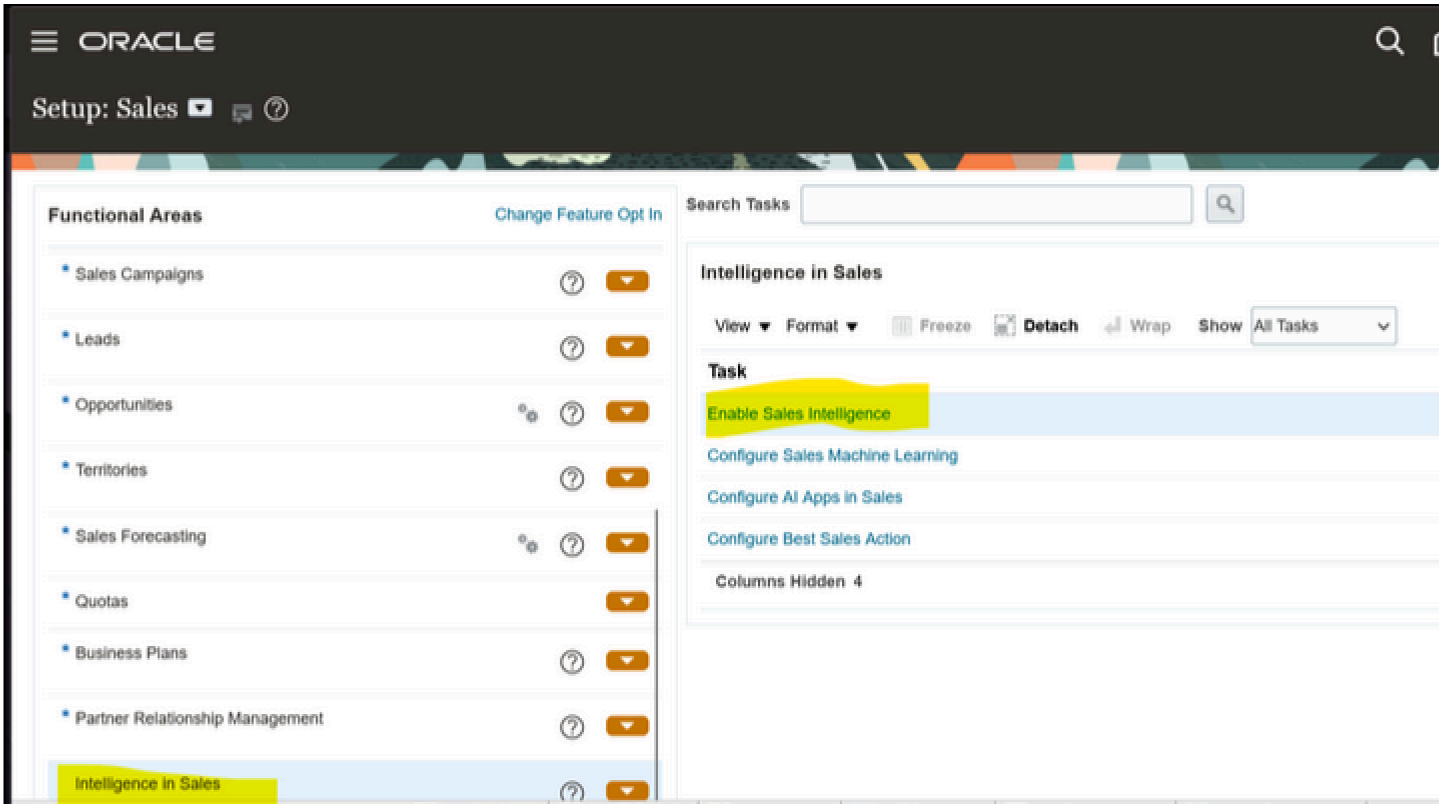
For more information, see the New Feature Opt-In section of the Oracle Applications Cloud - Using Functional Setup Manager guide on the Oracle Help Center (docs.oracle.com).

Enable Sales Intelligence for Sales Insights

Here are the high level steps that needs to be done to ensure you enable Sales Intelligence and get access to sales insights.

> **Note:** The Employee (Abstract Role) is required to view Sales Insights.

1. Sign in to Oracle CX Sales using Administrator role.
2. In the Setup and Maintenance work area, go to the following:
   - Offering: Sales
   - Functional Area: Intelligence in Sales
   - Task: Enable Sales Intelligence



3. From the **Enable Sales Intelligence Features** page, click **Start** and select the features you want.

   When you've selected the features, a message will display that an eligibility report will run and the application will close. Wait to receive a confirmation email before proceeding.
4. Once you receive the confirmation email, navigate to the **Enable Sales Intelligence** again as per step 2.
5. Click **Resume** to display the **View Features Eligibility Report**.

   This report lists the thresholds and data usage of factors that contribute to the overall data sufficiency for each of the sales intelligence features selected.
6. Expand the report and make sure the data for your enabled insight feature is sufficient and that the **Overall Status** contains the **Sufficient** value.

   The overall status can contain one of these values for the sales intelligence feature:
   - Insufficient : Your application has insufficient data to analyze and activate the feature
   - Emerging : Your application has some data to analyze but that isn't sufficient to activate the feature
   - Sufficient : Your application has sufficient data to analyze and activate the feature.
7. From the **Verify Recommended Features** section, select all the features that you require.
8. Click **Continue**.

9. Click **Done**.

> **Note:** To support opt in with custom layouts for the Account, Lead, and Opportunity objects, you must go to Application Composer and click **Show** in the Sales Insights section for your custom layout object. See the *Enable Sales Insights for Custom Layouts in Classic Sales UI* topic for more details.

Schedule Sales Insights Data Processes

Next, you must schedule Sales Insights data processes. For instructions about how to schedule the jobs, see the related links.

You must have the following privileges to run Sales Insights processes:

- ZCA_VIEW_SALES_INSIGHTS_PRIV
- ZCA_SALES_INSIGHTS_JOB_PRIV
- FUSION_APPS_CRM_ESS_APPID

This table outlines the name and frequency of when you must run the sales insights processes:

Process	Frequency
Generate Sales Insight Metrics	Once a week
Generate Sales Insight Facts	Once a week
Generate Sales Insight Facts Using Signals	Daily
Purge Redundant Sales Insight Facts	Once a month

You must ensure that both the **Generate Sales Insight Facts Using Signals** and the **Generate Sales Insight Facts** onboarding processes for the insights features have fully completed with a successful data sufficiency status of **Sufficient**.

Related Topics

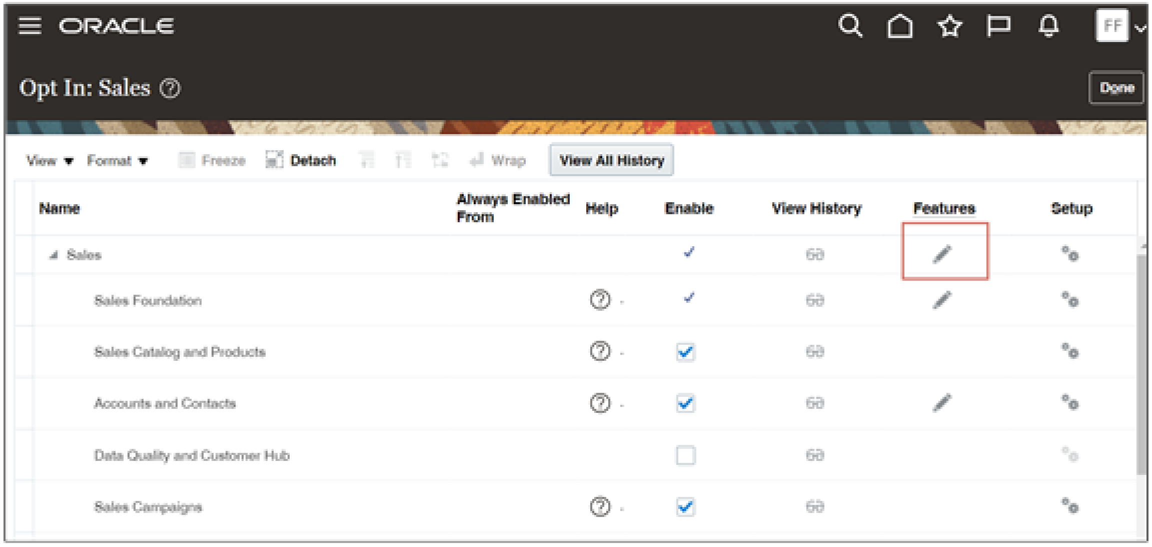
- Submit Scheduled Processes and Process Sets

Enable Sales Recommendations for Sales Insights

Here's how to enable sales recommendations. The sales recommendation feature analyzes text of notes and call logs and recommends follow-up tasks and appointments. When enabled, this feature replaces the automatic creation of a task after call wrap ups.

1. Open the **Setup and Maintenance** work area and select the Sales offering.

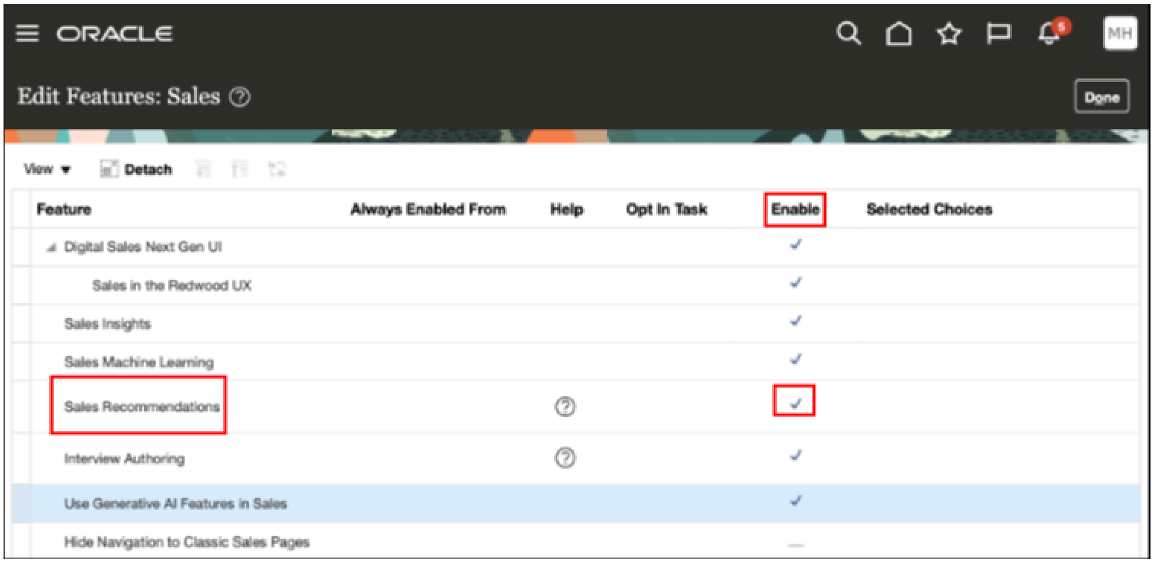
2. In Setup: Sales, click the **Change Feature Opt In** link.
3. In the Opt In: Sales page, click **Features** for Sales (the pencil icon in the topmost row highlighted in the screenshot).



Opt In: Sales

Name	Always Enabled From	Help	Enable	View History	Features	Setup
Sales			✓	👓	✎	⚙
Sales Foundation		? -	✓	👓	✎	⚙
Sales Catalog and Products		? -	☑	👓		⚙
Accounts and Contacts		? -	☑	👓	✎	⚙
Data Quality and Customer Hub			□	👓		⚙
Sales Campaigns		? -	☑	👓		⚙

4. Select **Enable** for **Sales Recommendations** as highlighted:



Edit Features: Sales

Feature	Always Enabled From	Help	Opt In Task	Enable	Selected Choices
Digital Sales Next Gen UI				✓	
Sales in the Redwood UX				✓	
Sales Insights				✓	
Sales Machine Learning				✓	
Sales Recommendations		?		✓	
Interview Authoring		?		✓	
Use Generative AI Features in Sales				✓	
Hide Navigation to Classic Sales Pages				—	

5. Click **Done**.

Recommendations Based on Sales Insights

Machine learning models, sales insights and historical sales analysis further enable salespeople to increase their sales win rates by offering data-driven pipeline management and recommendations. For example, setting win probability variance thresholds and providing multiple recommendations per opportunity help salespeople optimize their sales opportunities.

Providing recommended actions lets salespeople know which opportunities to focus on, which ones are at risk to close, and specifically, what's the best recommended action to take to close opportunities. The following predefined sales insights are available and your administrator can display these insights as filters on the object list views for your accounts, opportunities and leads. See the section **Make the Field Searchable in Its Business Object** in the topic *How do I make a field searchable in the UI?* for more information.

- Account engagement level

  Identifying those accounts that are either highly or moderately engaged or have a low level of engagement (lonely) and by observing certain accounts helps salespeople to:

  - Prioritize the next set of actions for records and tasks
  - Improve productivity with moving the sales pipeline along
  - Identify quickly what's wrong with a lead or opportunity and take the necessary corrective action

- Opportunity activity effectiveness

  By knowing if the activities associated with a given opportunity are enough and exact helps you effectively optimize your opportunities and move them along the sales pipeline. For example, you might want some insights to check if you're doing enough to nurture opportunities while sales managers might want to gain an objective understanding of:

  - Which opportunities aren't being nurtured effectively and need more attention?
  - Who are our most engaged customers?
  - Which accounts can be reached again to increase the sales pipeline?

- Lead activity effectiveness

  Sales insight details about lead tracking activities associated with a given sales lead helps you effectively optimize your leads and move them along the sales pipeline. For example, you might want to check if you're doing enough to nurture leads while sales managers might want to recommendations or insights into which leads aren't being nurtured effectively and need more attention?

Factors that Influence the Account Engagement Levels

Actions for the above insights might also be based on insights that recommends when:

- There's a lonely account with low engagements levels.
- An opportunity activity effectiveness needs attention.
- There's a contact validation that shows an alert for insufficient contacts to pursue a lead or a key decision maker is missing.

Engagement level are based on factors such as the number of active leads, the number of deals, the frequency of interactions and activities with the account and so on.

Here's a table that outlines some factors that influence the 3 account engagement levels with some sample scenarios:

Level	Criteria	Scenario
Highly	Accounts that have a high number of records and activities recently updated or newly created.	If you've many accounts to manage, then you can use this information to prioritize your sales efforts. For example, you might decide to focus on accounts with a high number of open activities or touches, as these accounts are more likely to be ready to move forward in the sales process.
Moderate	The number of accounts that have several records and activities recently updated or newly created is between the Highly and Lonely totals of engagement levels.	If you note that an account has a moderate number of interactions, you might infer that the account is in the early stages of the buying process and that you should focus on building relationships with the contacts. Or you might identify potential opportunities for these accounts. By seeing the number of interactions and activities that have taken place with different contacts within an account, you get a better sense of which opportunities are most likely to close.
Lonely	Accounts that have no records and activities recently updated or newly created since X days/weeks. For example, {activity/interaction/lead/opty/SR since X days/weeks} where X denotes a quarter of the average sales cycle time.	You can identify which accounts need more attention. If you see that an account has a low number of open activities, touches, or emails sent, then this insight level might alert you to increase your engagement with the account.

Enable Sales Insights for Custom Layouts in Classic Sales UI

Here are the steps to display the Sales Insights section for a custom account layout page for your classic sales UI. You can also follow similar steps to show the Sales Insights section for custom sales lead and opportunity layout pages.

> **Note:** When you first enable Sales Insights for a business or custom object, then all other objects that support the Sales Insights feature are automatically enabled and ready for use.

1. Ensure you're working in an active sandbox.
2. Click **Navigator** > **Configuration** > **Application Composer**.
3. In the Objects navigation tree, expand **Standard Objects** and then expand the object whose pages you want to change. For example, select the **Account** object.

   > **Note:** Select the **Sales Lead** object to display Sales Insights section for the leads UI and select **Opportunity** object to display Sales Insights section for the opportunities UI.

4. Click the **Pages** node.

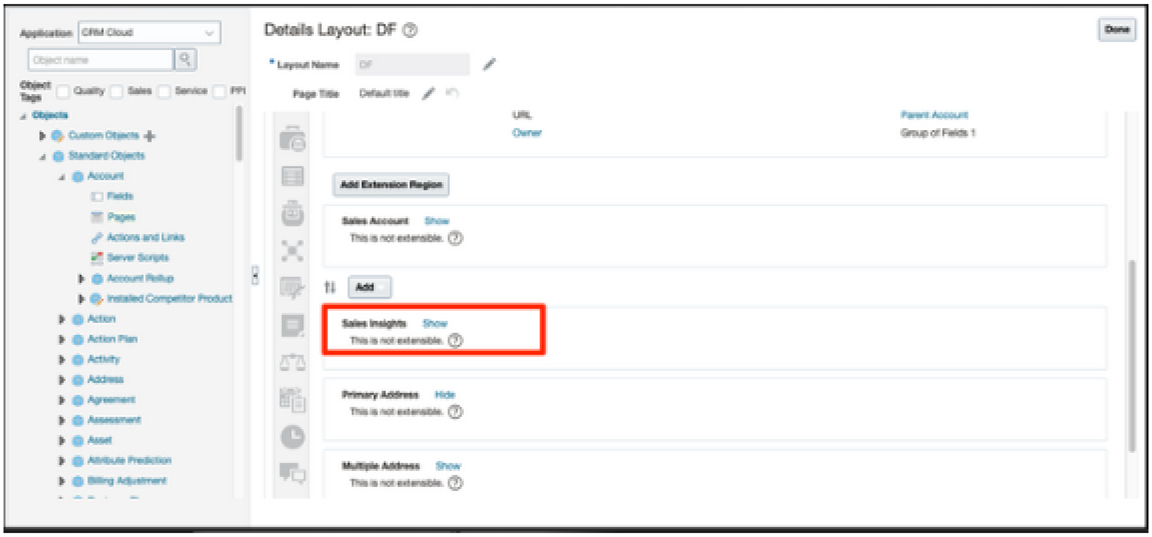
5. Ensure that the **Application Pages** tab is selected for either the account, contact, household, or partner object.
6. In the Details Page Layouts region, edit the relevant layout.

   If none exists, then duplicate the standard layout using the duplicate layout icon, and edit the resulting layout.
7. In the Details Layout page for Account, click the **Profile** tab.

   > **Note:** Click the **Summary** tab if you're displaying the Sales Insights section for leads and opportunities.

8. In the Sales Insights region for the selected object, click **Show** to show the Sales Insights region at runtime.

   Here's an example screen shot for the Account object.



Details Layout: DF

* Layout Name: DF

Page Title: Default title

Sales Account Show
This is not extensible.

Sales Insights Show
This is not extensible.

Primary Address Hide
This is not extensible.

Multiple Address Show
This is not extensible.

9. Click **Save and Close**, then **Done**.
10. Publish the sandbox according to your company's business practices.