
Exercise 9-3: Configure Range-Based Attribute Pricing

Scenario



The standard attribute pricing matrix is great for calculating prices based on specific values, but at Infiwave the Home Hub Modem product uses range-based pricing. If someone buys up to 4 Home Hub Modems in their order, they're charged \$65 for each. However, if they buy 5 - 9 they're charged only \$50 for each modem. You've got this covered with the RangeAttributePricingMatrix and a new expression set.

Goals

- Add data to a standard attribute-based pricing lookup table
- Create a new expression set to price the line item in the cart
- Add a new step to the default pricing plan

Tasks

1. Create the AttributePricingMatrix lookup table
2. Create the RangeAttributePricingProcedure expression set
3. Create a new pricing plan step for range-attribute pricing

Task 1: Create the AttributePricingMatrix lookup table

As you did with standard attribute-based pricing, you'll set up a decision matrix, this time called RangeAttributePricingMatrix, that will be used in calculating the price based on the quantity that the customer orders.

Instructions

1. Create a new version of the RangeAttributePricingMatrix, which is the sample decision matrix for range attribute-based pricing.
 - a. Navigate to the **Lookup Tables** tab. You may need to search for it again using the **App Launcher**.
 - b. From the **All Decision Matrices** list view, click the **RangeAttributePricingMatrix**, which was installed from the DataPack.
 - c. Select the **Related** tab and click **RangeAttributePricingMatrix V1**.
 - d. In the decision matrix header, click **New Version**.
 - e. Add the following details to the New Decision Matrix Version dialog:

Field	Value
Version Data Selection	Include column headers
Name	RangeAttributePricingMatrix V2
Start Date Time	[Yesterday's date]
End Date Time	[leave blank]
Rank	1
Version Number	2
Group Key Value and Sub Group Key Value	[leave blank]

- f. Click **Save**.
2. Configure the RangeAttributePricingMatrix v2 to hold your product pricing information.

- a. On the Decision Matrix Version screen, click **Add Row** two times to add two rows to the matrix.
- b. Enter the data shown below in the new matrix rows. Notice the MRC values are blank. You're going to leave them in there and ensure they're included in the calculations because maybe in the future Infiwave might introduce monthly charges for the modem - and it's best to be prepared!

SourceProductNa me (Text)	SourceProd uctCode (Text)	Characteristic Name (Text)	Characteristic Value (Text)	Quantity (Text)	MRC (Curr ency)	NRC (Curre ncy)
Home Hub Modem	C-MOD-002	Grade	Best	1-4	0	65.00
Home Hub Modem	C-MOD-002	Grade	Best	5-9	0	50.00



Notice that the Quantity column uses a Text column type instead of a Number Range column type. While either format is acceptable, it's recommended that you use Text column types for numeric ranges because they're easier to manage and maintain consistency when using several ranges in a single matrix.

- c. Select **Save**.
- d. In the RangeAttributePricingMatrix V2 header, click **Edit**. Check the **Enabled** checkbox and ensure the **End Date Time** fields are blank before selecting **Save**.

Decision Matrix Version
RangeAttributePricingMatrix V2 [Edit](#) [New Version](#)

Version Number	Enabled	Start Date Time	End Date Time	Rank	API Name
2	<input checked="" type="checkbox"/>	10/19/2022, 3:47 AM		20	RangeAttributePricingMatrix_V2

Matrix Related Details

2 records - Sorted by SourceProductName
[Upload CSV File](#) [Generate CSV File](#) [Add Column](#) [Add Row](#) [Delete Rows](#) [Refresh](#) [Save](#)

Input Data					Output Data	
SourceProductName (Text) ↑	SourceProductCode (Text)	CharacteristicName (Text)	CharacteristicValue (Text)	Quantity (Text)	MRC (Currency)	NRC (Currency)
Home Hub Modem	C-MOD-002	Grade	Best	5-9	\$0	\$50
Home Hub Modem	C-MOD-002	Grade	Best	1-4	\$0	\$65

Task 2: Create the RangeAttributePricingProcedure expression set

Now the range-attribute-based-pricing decision matrix is set up, it's time to create the associated expression set, to perform the calculations.

Instructions

1. Create the expression set.
 - a. Navigate to the **Expression Sets** tab. You may need to search for it again using the **App Launcher**.
 - b. Click **New**.
 - c. Enter `RangeAttributePricingProcedure` for the **Name** and select **Save**.
 - d. On the **Related** tab, from the menu ▼ next to **RangeAttributePricingProcedure V1**, select **Edit**.
 - e. In the **Rank** field, enter `1` and click **Save**.
2. Next, configure the expression set using the Expression Set Builder. Start by adding the lookup table and configuring it to include output variables for the one-time and recurring charges.
 - a. Click **RangeAttributePricingProcedure V1** to open the record.
 - b. In the Expression Set header, select **Open in Expression Set Builder**.
 - c. In the Expression Set Builder workspace, click the plus + symbol and select **Lookup Table**.
 - d. In the **Lookup Table Details** field, search for and select **RangeAttributePricingMatrix**.
 - e. With the new Lookup Table element selected, click the **Element Details** tab and check the **Include in Output** checkbox.
 - f. Click the **Resource Manager** tab and select **Add Resource**.
 - g. As you did in the last exercise, add the one-time and recurring standard price variables:

Recurring Monthly Standard Price

Field	Entry
Resource Type	Variable
Resource Name	REC_MNTH_STD_PRC

Data Type	Currency
Decimal	2

One-Time Standard Price Variable

Field	Entry
Resource Type	Variable
Resource Name	OT_STD_PRC
Data Type	Currency
Decimal	2

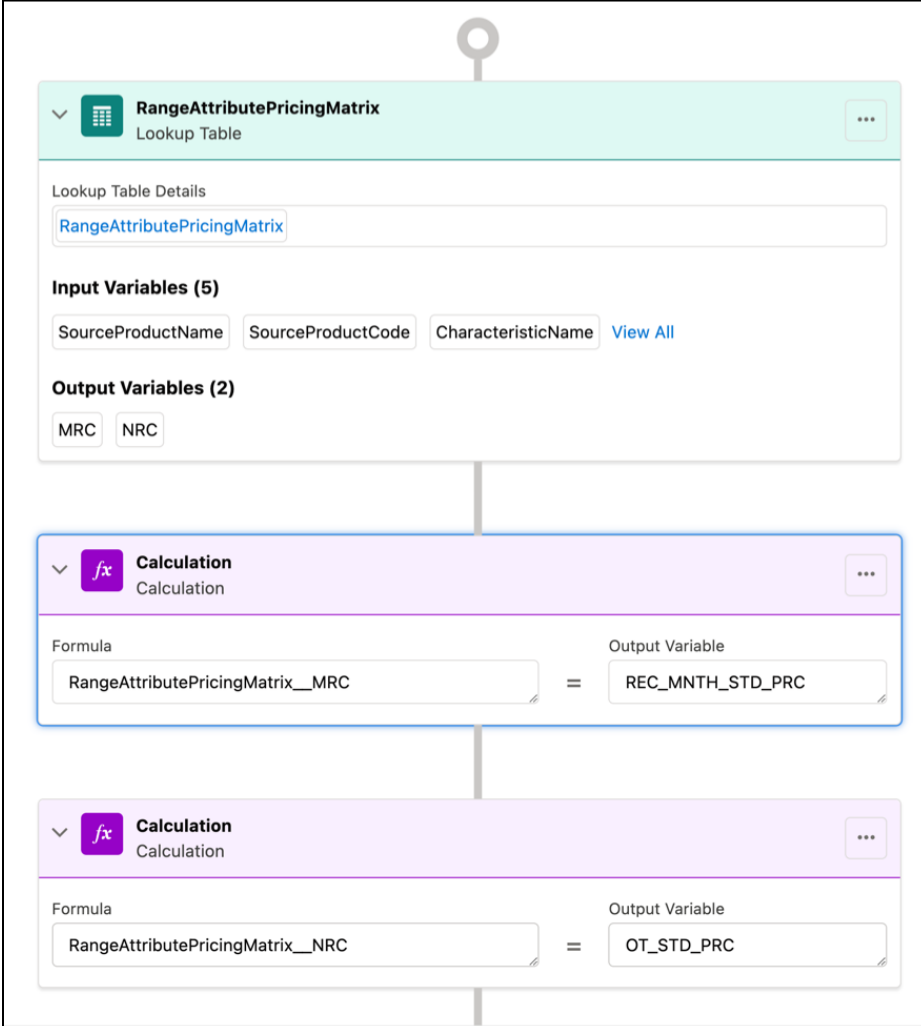
3. Next, add the calculation elements for the monthly recurring charge and the one-time charge.
 - a. In the workspace, click the + and add a new **Calculation** element for the monthly recurring charge with the following information:

Field	Entry
Formula	RangeAttributePricingMatrix__MRC
Output Variable	REC_MNTH_STD_PRC

- b. Add another **Calculation** element for the one-time charge and enter the following information:

Field	Entry
Formula	RangeAttributePricingMatrix__NRC
Output Variable	OT_STD_PRC

4. Check that all the elements you've added have the **Include in Output** checkbox selected in the **Element Details** tab.



The screenshot displays a configuration interface for a pricing matrix. It features three main components connected by a vertical line:

- RangeAttributePricingMatrix** (Lookup Table):
 - Lookup Table Details: RangeAttributePricingMatrix
 - Input Variables (5): SourceProductName, SourceProductCode, CharacteristicName, View All
 - Output Variables (2): MRC, NRC
- Calculation** (Calculation):
 - Formula: RangeAttributePricingMatrix__MRC
 - Output Variable: REC_MNTH_STD_PRC
- Calculation** (Calculation):
 - Formula: RangeAttributePricingMatrix__NRC
 - Output Variable: OT_STD_PRC

5. Select **Save** and then click **Activate**.

6. Now let's test your expression set to make sure everything is working ok.
 - a. Click **Simulate** and enter the following details. Be careful to enter the values in the correct field, as they may appear in a different order than listed here.

Input Variable	Test Value
SourceProductName	Home Hub Modem
CharacteristicName	Grade
SourceProductCode	C-MOD-002
Quantity	1-4
CharacteristicValue	Best

- b. Click **Simulate** to run the simulation. What price is returned? It should be \$65.

To help troubleshoot any potential errors, try simulating the expression set by clicking **Simulate** and entering corresponding values from the connected decision matrix.



When simulating the RangeAttributePricingProcedure, you must enter the literal range values from the Quantity field of the RangeAttributePricingMatrix (e.g. "1-4") to return an accurate calculation. This is because the PricingPlanHelper.cls, which identifies the corresponding range based on the Quantity entered, is not called from the expression set. You'll test this calculation in the cart in a later exercise step.

- c. Switch back to the **Input Details** in the simulation and change the **Quantity** to 5-9 before clicking **Simulate**. What price is returned? It should be \$50.
7. Close the **Expression Set Builder** browser tab.

Task 3: Create a new pricing plan step for range-attribute pricing

Next, you'll create a new pricing plan step that calls the range-attribute lookup table and expression set you just created.

Instructions

1. Create a new pricing plan step for Range Attribute Pricing.
 - a. Click **New Item** and enter the following information:

Field	Entry
Name	Calculate Range Attribute Pricing
Implementation Name	CustomPricingPlanStepImpl
Method Name	GetMatrixPrice
Sequence	6
Active	✓

- b. Add the following parameters to the step:

First Box	Second Box
ProcedureName	RangeAttributePricingProcedure
MatrixName	RangeAttributePricingMatrix
RangeFields	Quantity
DecisionMatrix	true
IncludeAttrInfoInRangeKeys	false



When you want to use range attribute pricing, you specify the API names of fields or attributes using the RangeFields or RangeAttributes parameters. The fields and attributes listed need to have corresponding columns in the RangeAttributePricingMatrix.

- c. Click **Done**.
- d. Click **Save** in the General Properties pane.

Parameters	
ProcedureName	RangeAttributePricingProcedure
MatrixName	RangeAttributePricingMatrix
RangeFields	Quantity
DecisionMatrix	true
IncludeAttrInfoInRangeKeys	false

+

[Done](#)



The IncludeAttrInfoInRangeKeys parameter controls how the range column of the decision matrix is handled based on the conditional attributes in the other columns. In the range attribute matrix you created, the attributes are the same across all of the rows, and the quantity fields alone control the price of the product:

C-MOD-002 | Grade | Best | 1-4 | \$65
C-MOD-002 | Grade | Best | 5-9 | \$50

But what if you wanted to repeat the ranges and include another characteristic value (e.g. “Good”) as a condition, as shown below?

C-MOD-002 | Grade | Best | 1-4 | \$65
C-MOD-002 | Grade | Good | 1-4 | \$50

In this case, you’d set the IncludeAttrInfoInRangeKeys parameter to ‘true’ in the pricing plan step so that each attribute value is calculated along with the repeated range.



Yay! All done!