Contents

Install a sample dataset

Introduction

In this lab, you will learn to install sample tables and views from Sample Datasets. This particular sample dataset is a collection of customers, stores, products, and orders used to manage the shopping cart.

Estimated Time: 5 minutes

Objectives

In this lab, you will:

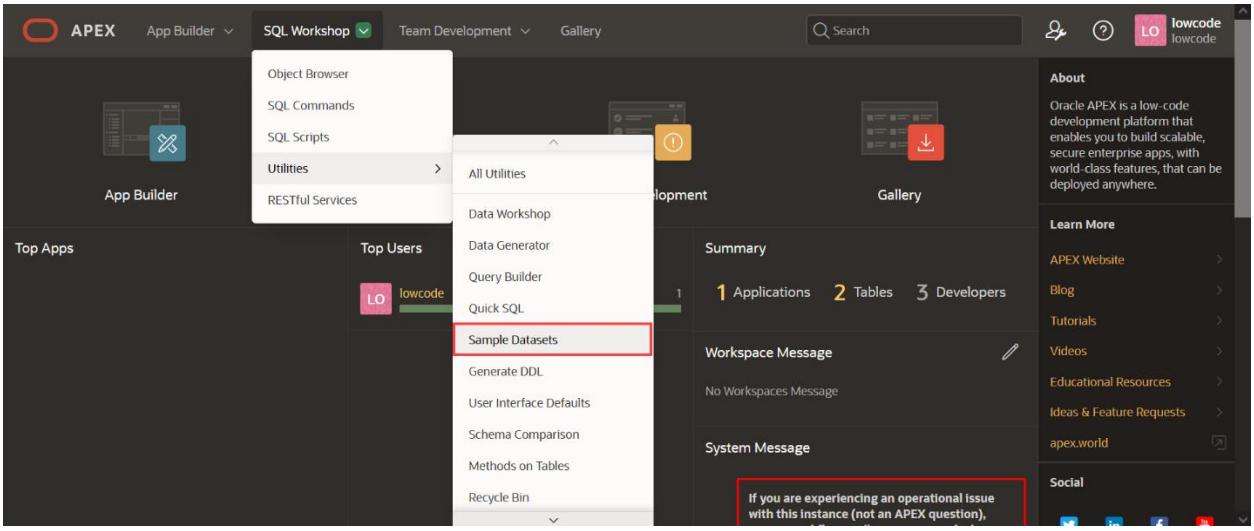
- Install a sample dataset into your Oracle APEX Workspace

Collapse All Tasks

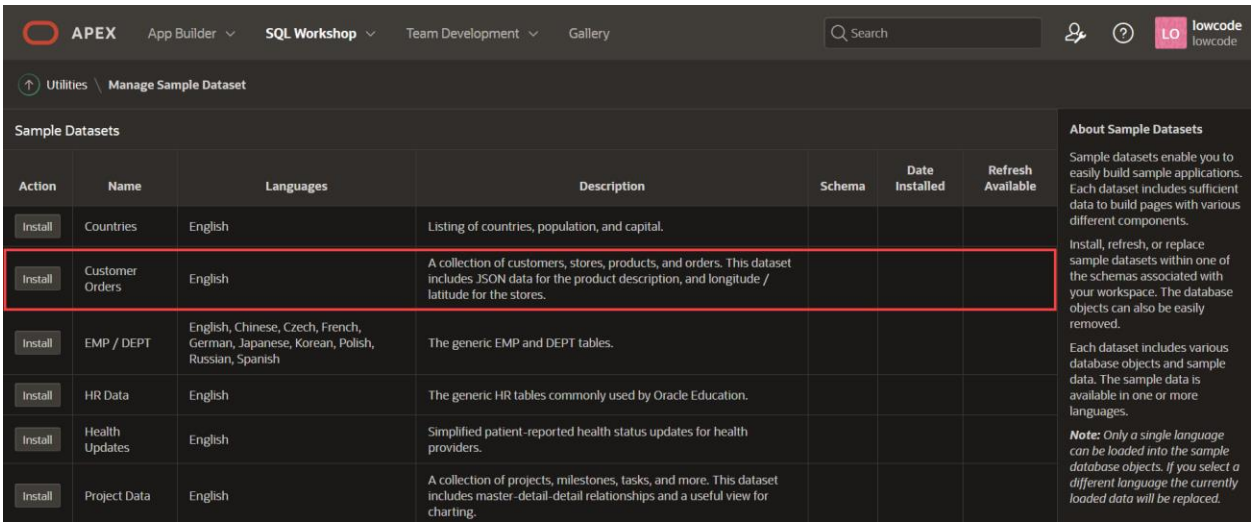
Task 1: Create Customer Orders Tables

1. Log into your workspace.

2. From your APEX workspace home page, as shown below, select the down-arrow to the right of **SQL Workshop**, then click the arrow to the right of **Utilities**, and choose **Sample Datasets**.

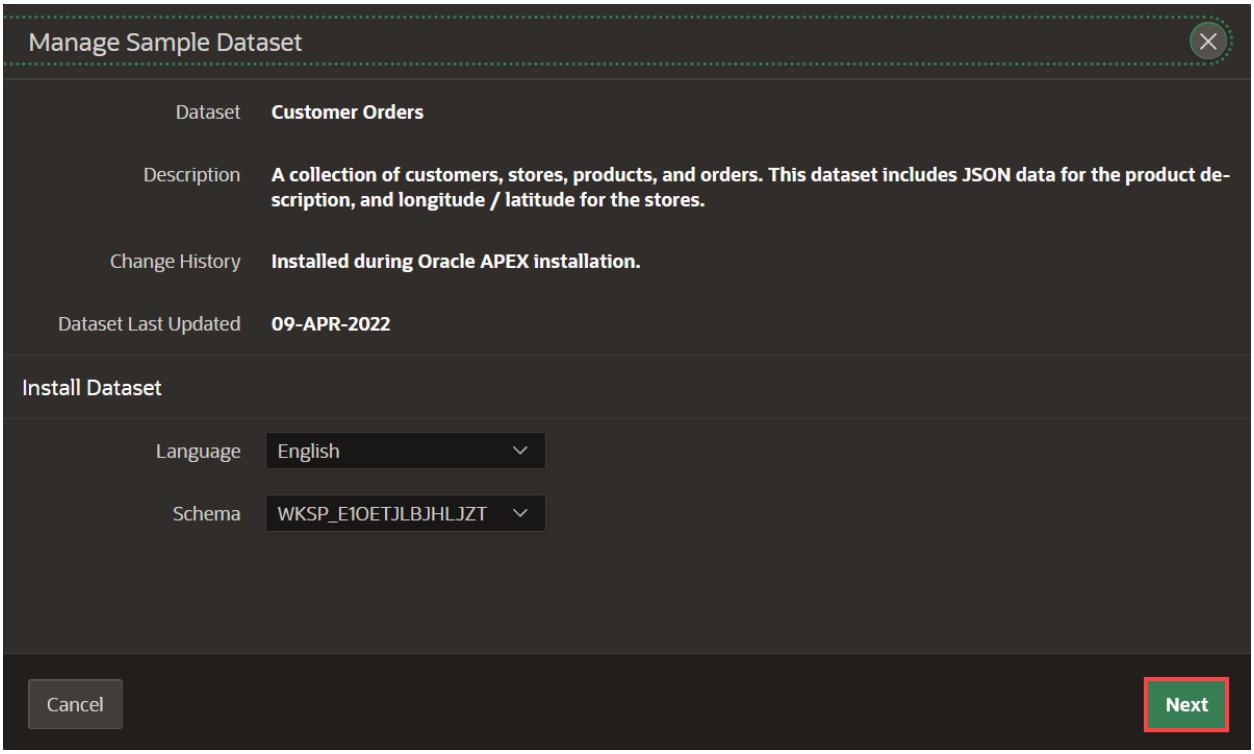


3. On the **Customer Orders** row, click Install.

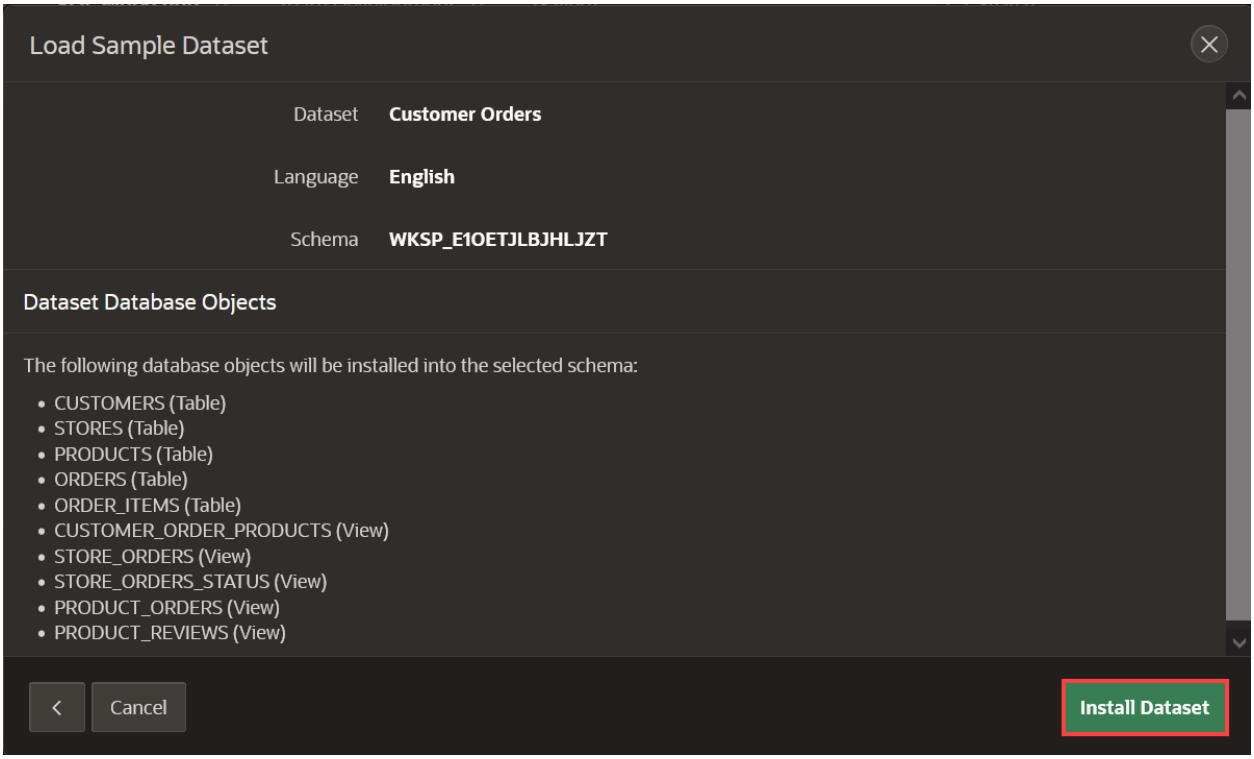


4. Click **Next**.

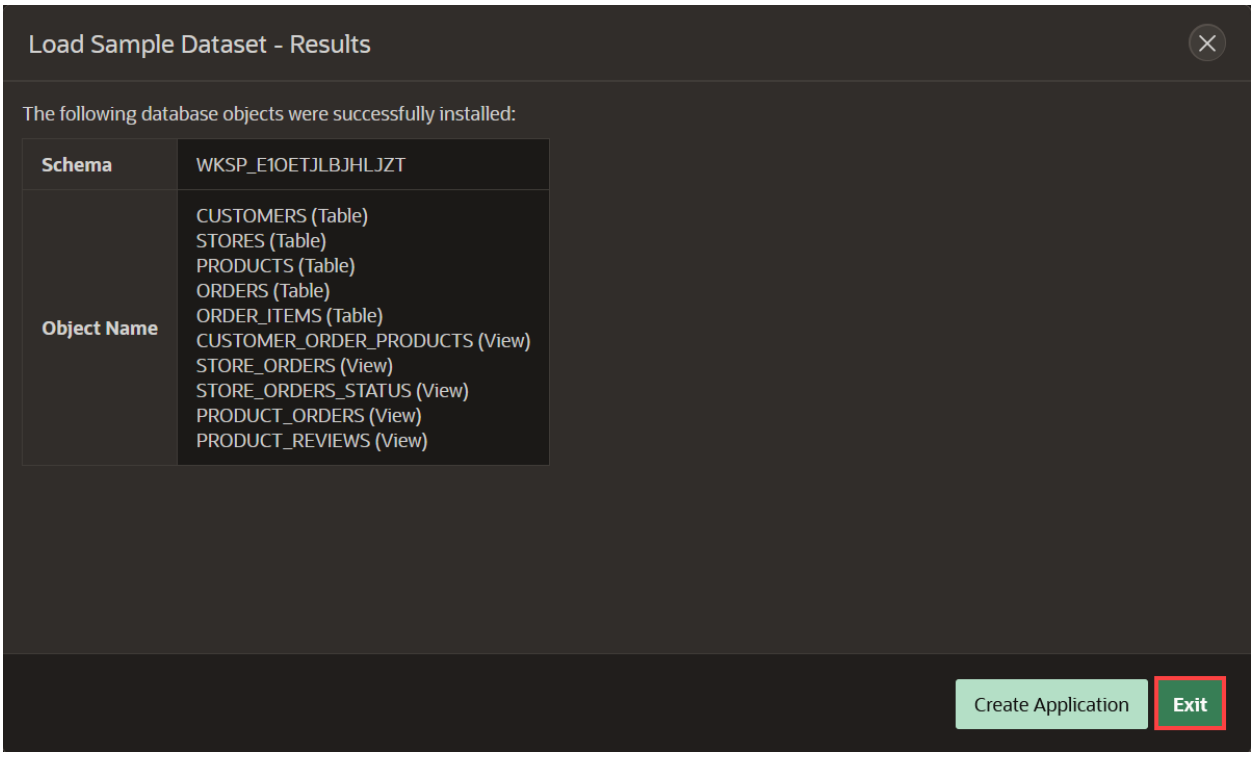
The schema name defaults to your current schema so will be different from the schema name shown below.



5. Click **Install Dataset**.



6. Click **Exit**.

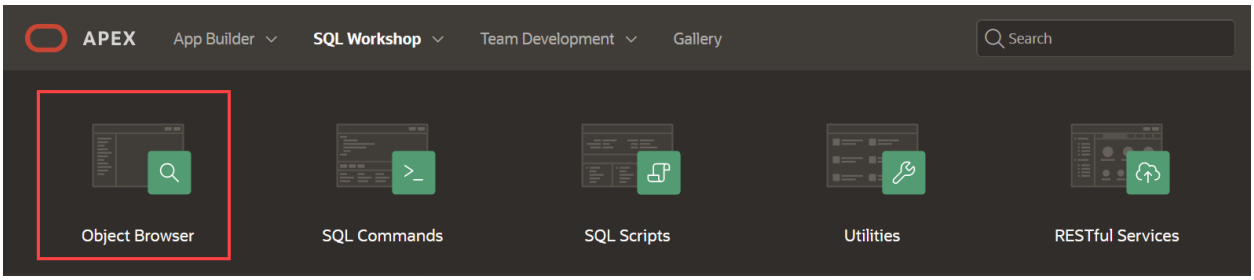


Note: You do NOT want to click Create Application, as you will manually create an application later.

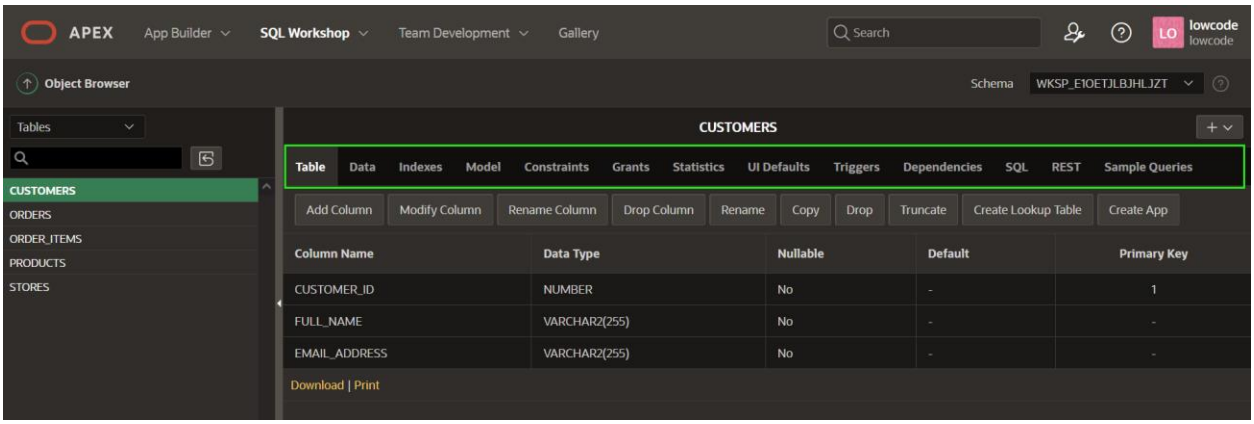
Task 2: Review Database Objects

1. In the Oracle APEX Home, click **SQL Workshop**.

2. Click **Object Browser**.



3. Click the new tables and the various tabs, such as Data, Constraints, and so forth, to review the table details.



This completes Lab 1. You now know how to install a sample dataset. You may now **proceed to the next lab**.

Add columns to the products table

Introduction

The **PRODUCTS** table includes some columns such as image, price, and details. But there are other characteristics that customers would appreciate knowing about the product, like color, type of clothing, and department. For that reason, you will add these columns to the Products table. To avoid data redundancy, you will need to create three additional tables to normalize the data. Instead of creating these three tables for yourself, you'll use the **Create Lookup Table** feature.

In this lab, you will learn how to add these three new columns to the **PRODUCTS** table and then create lookup tables for those new columns.

Estimated Time: 10 minutes

Objectives

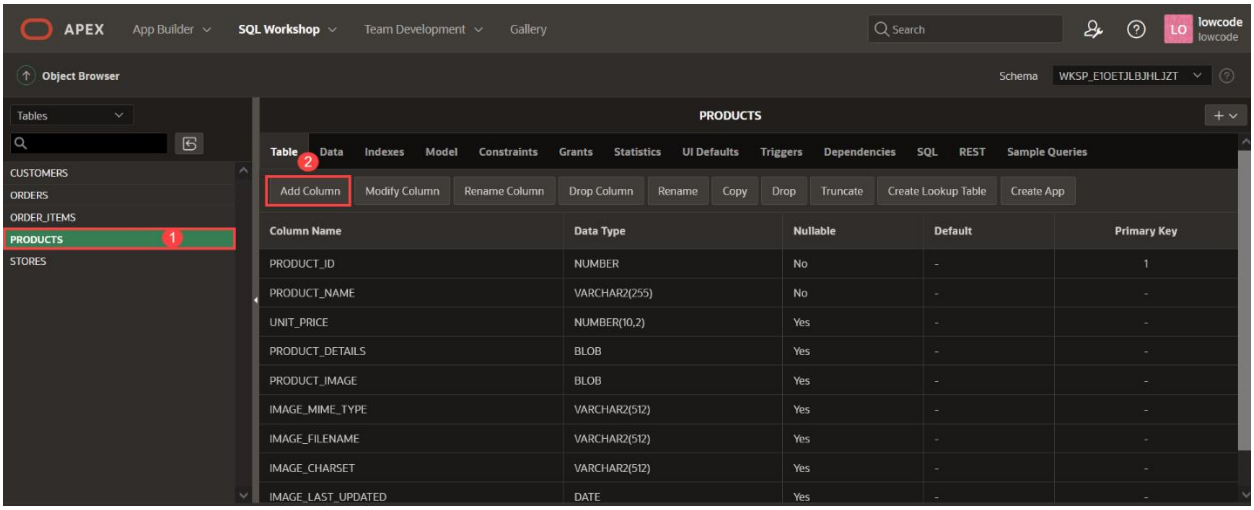
In this lab, you will:

- Add new columns to the existing Products table
- Populate the new columns
- Create lookup tables

Collapse All Tasks

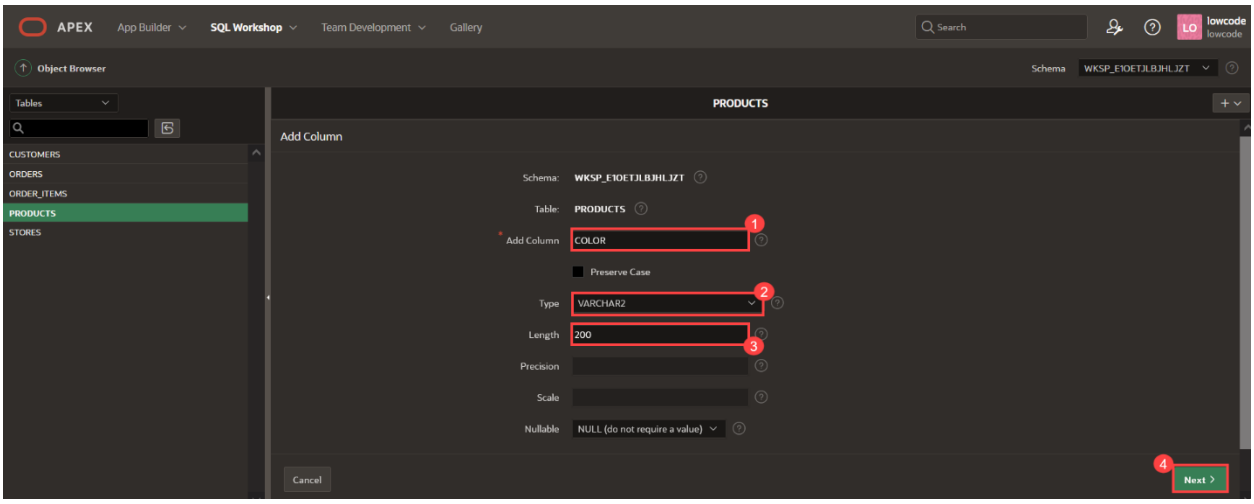
Task 1: Add Columns to the Products Table

1. From your APEX workspace home page, click **SQL Workshop**.
2. Click **Object Browser**.
3. Navigate to **PRODUCTS** table and click **Add Column** button.



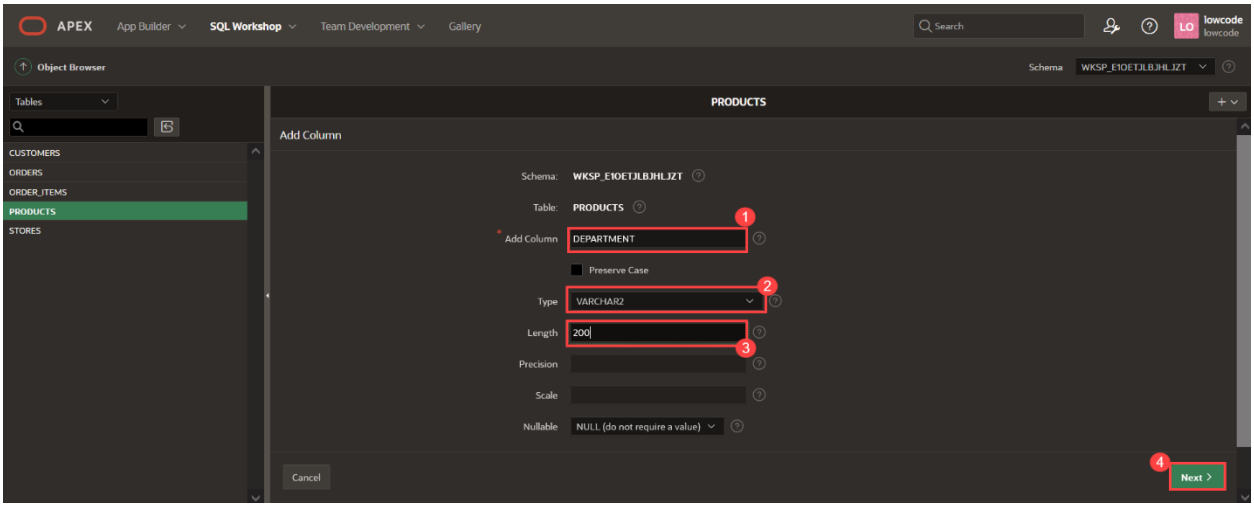
4. For **Color** column, enter the following:
    - Add Column - enter **COLOR**
    - Type - select **VARCHAR2**
    - Length - enter **200**.

    Click **Next**.



5. Click **Finish**.
6. Click **Add Column** button again.
7. For Department column, enter the following:
    - Add Column - enter **DEPARTMENT**
    - Type - select **VARCHAR2**
    - Length - enter **200**.

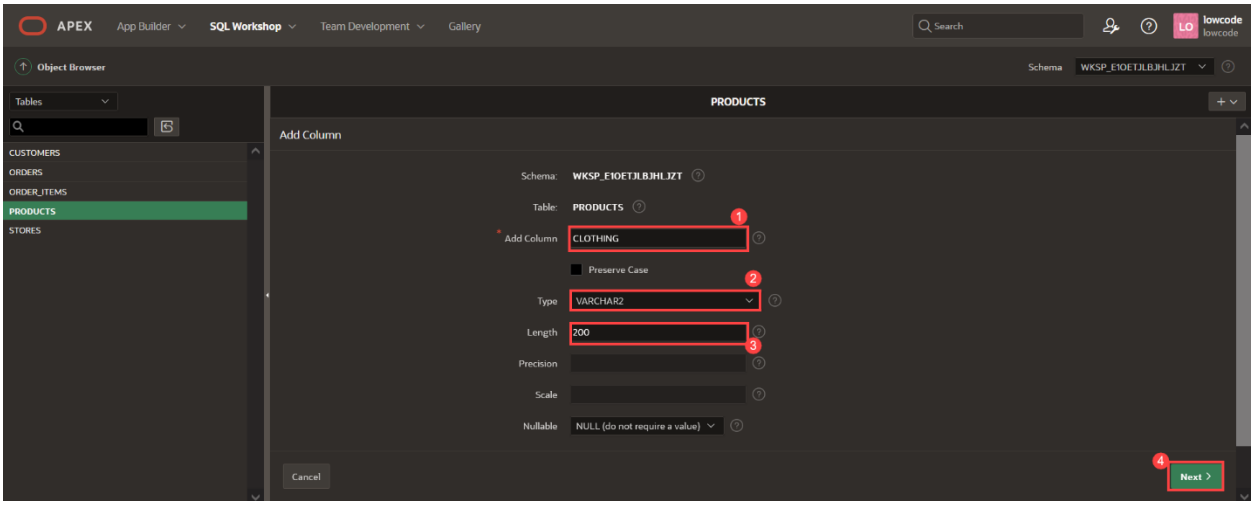
    Click **Next**.



8. Click **Finish**.
9. Click **Add Column** button again.
10. For Clothing column, enter the following:

- Add Column - enter **CLOTHING**
- Type - select **VARCHAR2**
- Length - enter **200**.

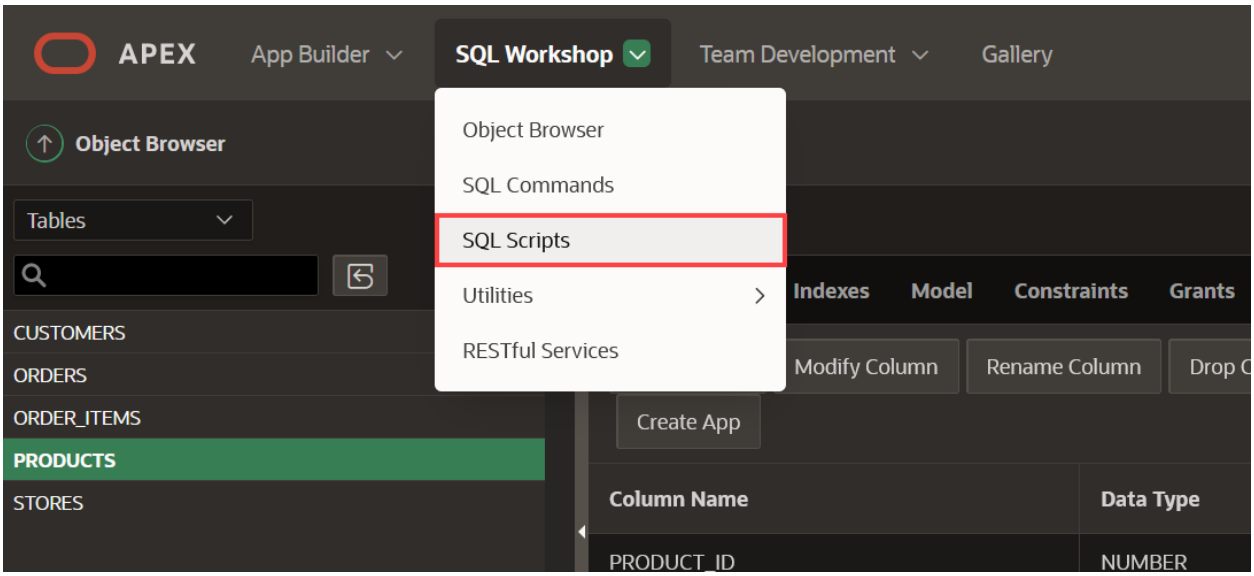
Click **Next**.



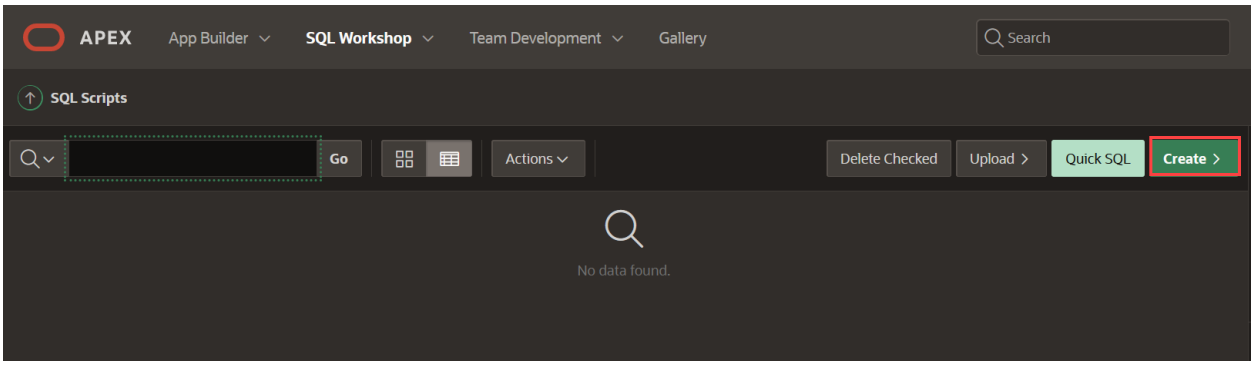
11. Click **Finish**.

Task 2: Populate the new columns

1. From the Oracle APEX Home, click **SQL Workshop** and select **SQL Scripts**.



2. Click **Create**.



3. For Script Name, enter **Populating new columns**. Copy the following script and paste into the editor.

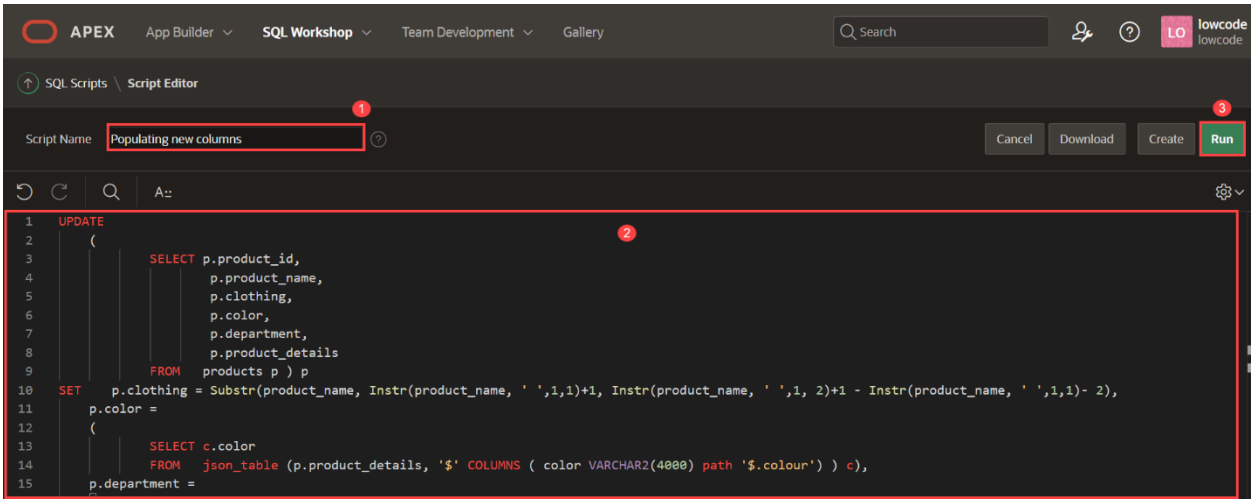
```
4. Copy

5. UPDATE

6.     (

7.             SELECT p.product_id,

8.                    p.product_name,

9.                    p.clothing,

10.                    p.color,

11.                    p.department,

12.                    p.product_details

13.             FROM   products p ) p

14. SET    p.clothing = Substr(product_name,
   Instr(product_name, ' ',1,1)+1, Instr(product_name, ' ',1,
   2)+1 - Instr(product_name, ' ',1,1)- 2),

15.     p.color =

16.     (
```

```
17.             SELECT c.color

18.             FROM   json_table (p.product_details, '$'
   COLUMNS ( color VARCHAR2(4000) path '$.colour') ) c),

19.     p.department =

20.     (

21.             SELECT g.department
```

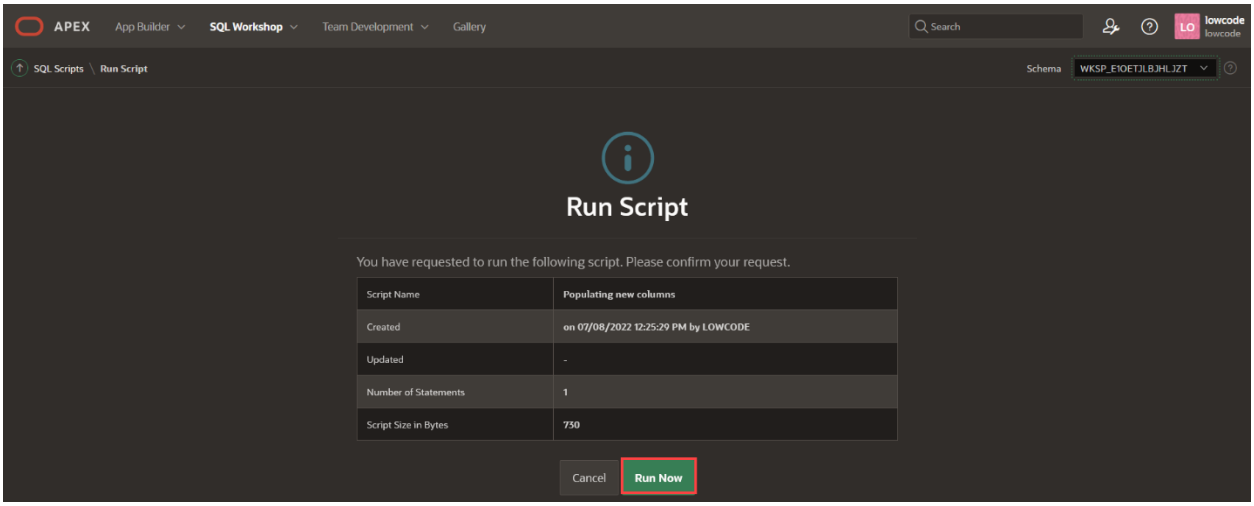
```
            FROM   json_table (p.product_details, '$'
COLUMNS ( department VARCHAR2(4000) path '$.gender') ) g)
```

This script inserts the unique product type values (e.g. Shirt, Jacket, Skirt, etc.) into the CLOTHING column in the **Products** table. Similary, it inserts the unique department names (e.g. Boy's, Girl's, Men's, Women's) and color names into the DEPARTMENT and COLOR columns respectively based on information found in the JSON product details column in the **Products** table.

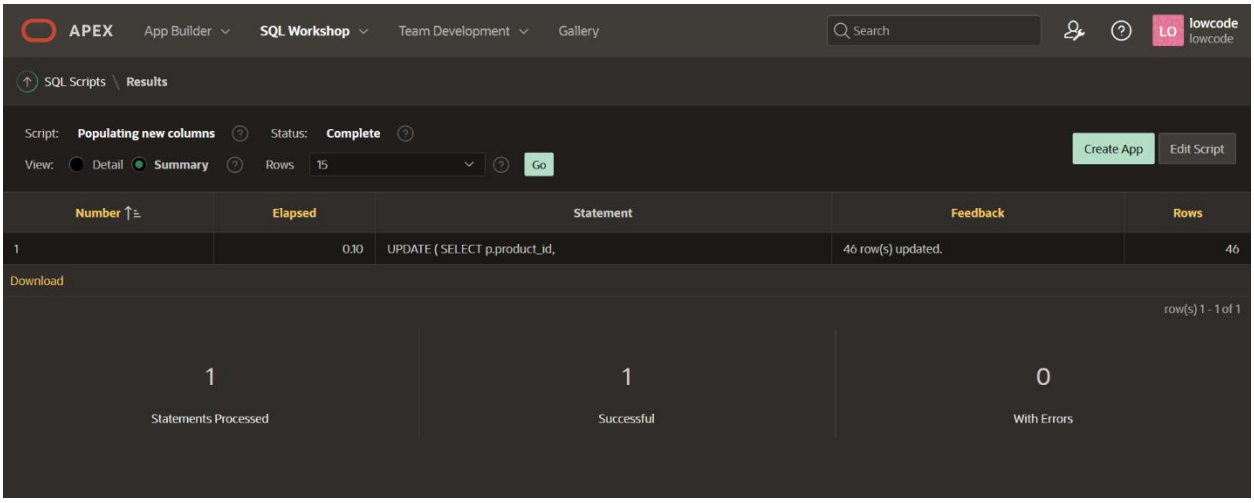
Click **Run**.



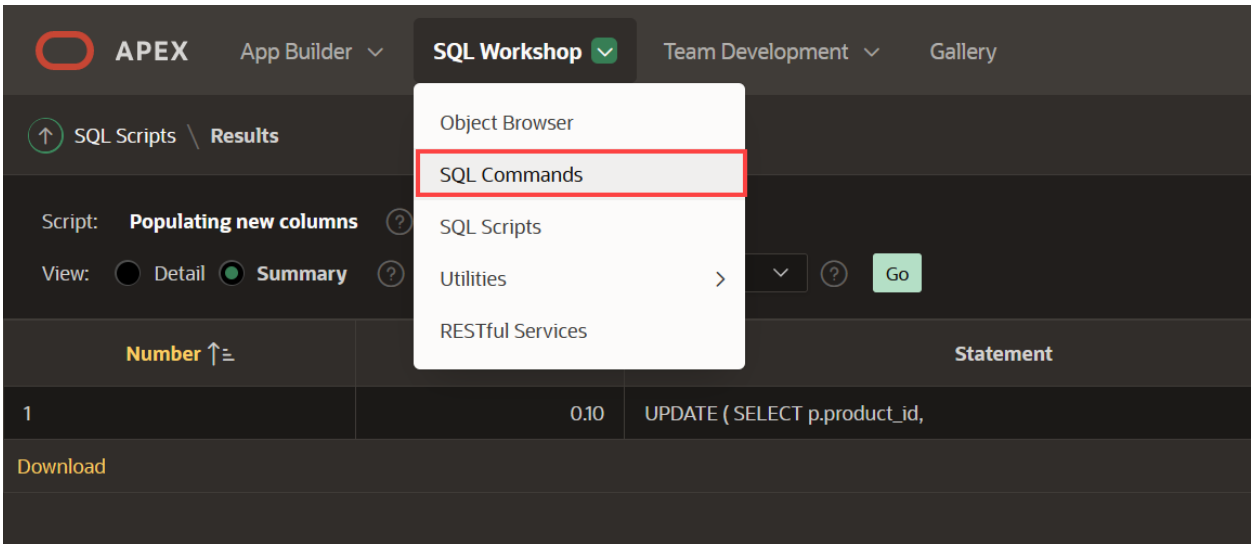
22. Click **Run Now**.



23. The Script Results page is displayed listing the number of statements processed, successful, and with errors.



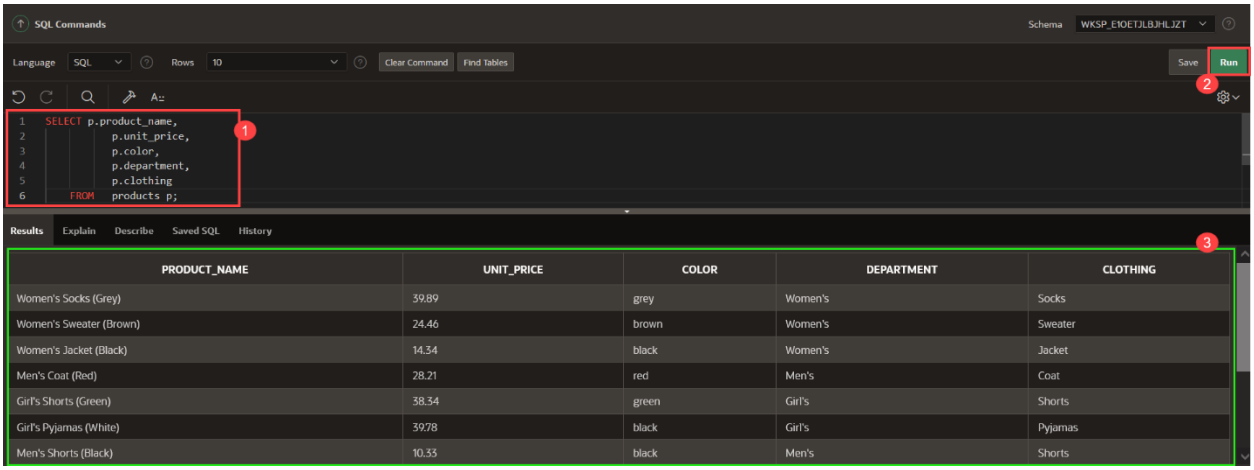
24. To check the values in the Products table, click **SQL Workshop** and click **SQL Commands**.



25. Copy the following SQL Query and click **Run**.

```
26. Copy

27. SELECT p.product_name,

28.        p.unit_price,

29.        p.color,

30.        p.department,

31.        p.clothing
```

```
FROM   products p;
```

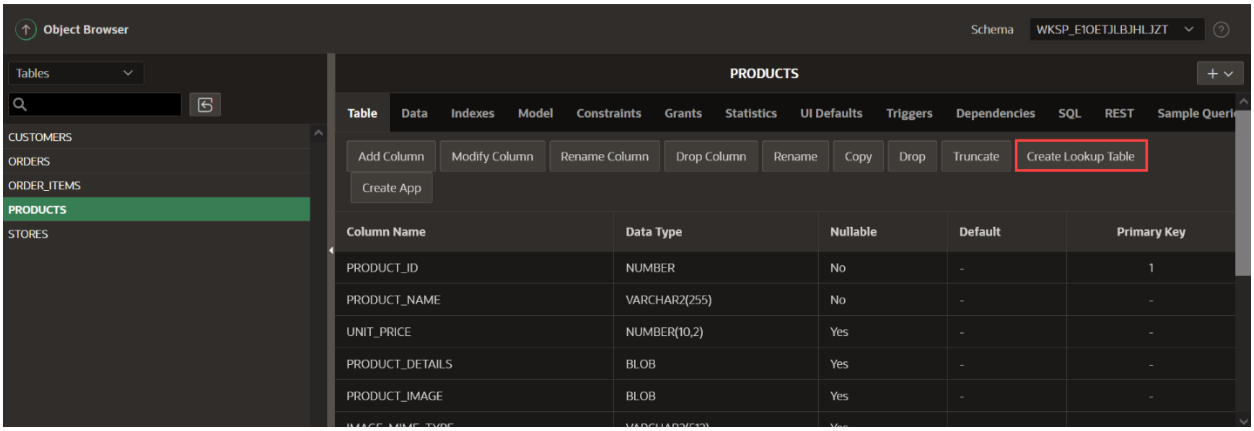


Task 3: Create Lookup Tables

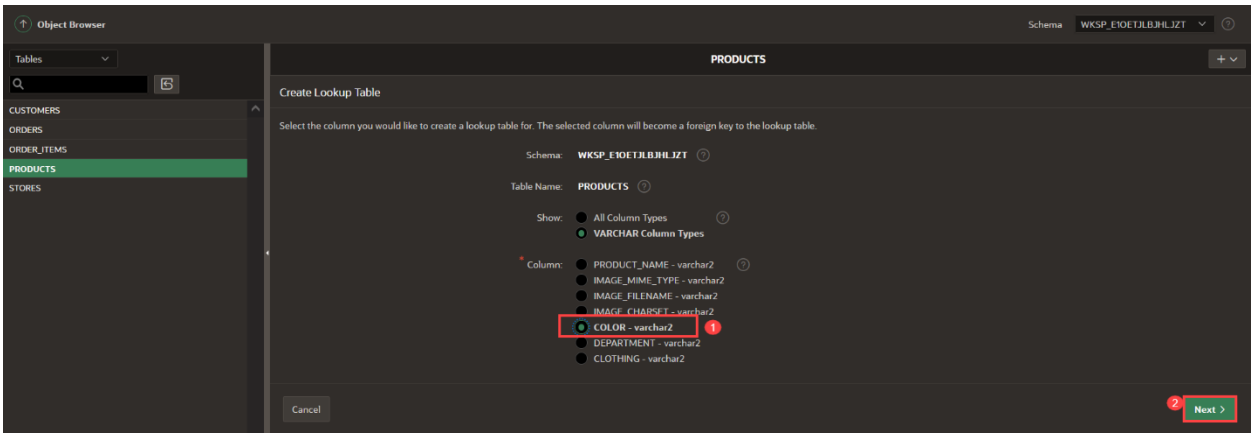
Multiple products may have the same values for Color, Department, and Clothing. Hence, to avoid repetition and make updates easy, you can create a lookup table for each. A lookup table stores the value of the available colors, departments, or clothing in a single place, and then each product can reference the value from the lookup table.

You will create lookup tables based on the new three columns, after you will have created a lookup table, you will notice that a new table was created and the column in the PRODUCTS table has been renamed and the data type was changed to NUMBER.

1. From your APEX workspace home page, click **SQL Workshop**.
2. Click **Object Browser**.
3. Navigate to **PRODUCTS** Table.
4. Click **Create Lookup Table** button.



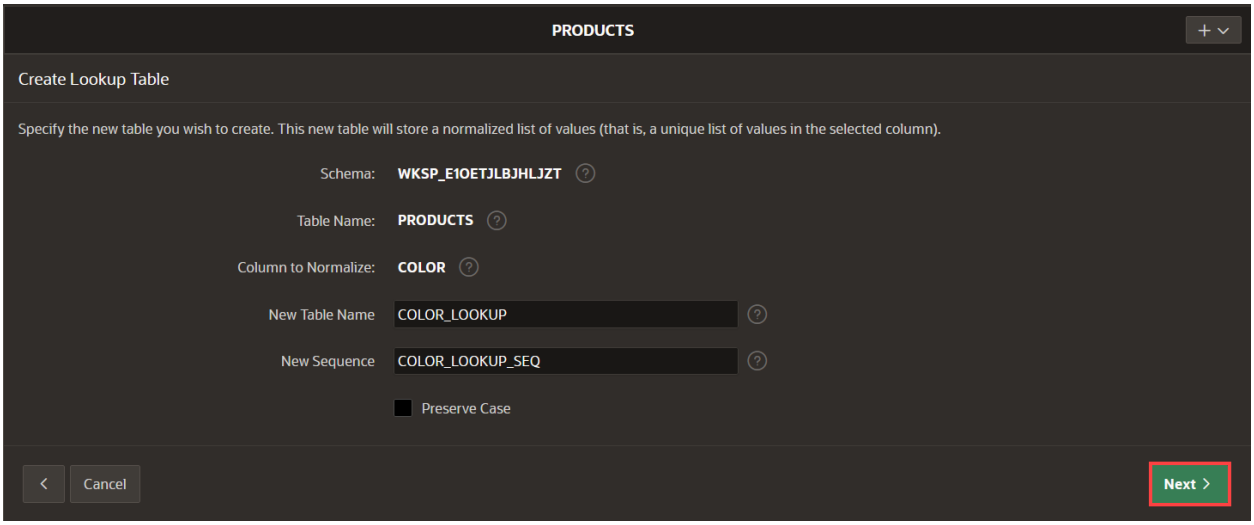
5. For Column, select **COLOR - varchar2**. Click **Next**.



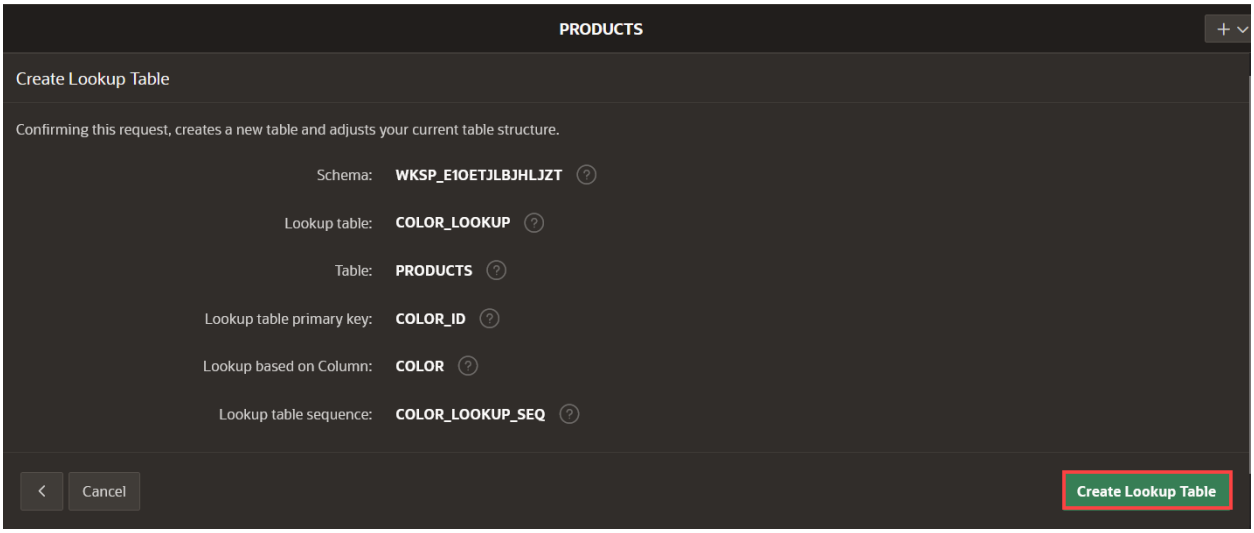
6. Leave the table and sequence name by default:

   - New Table Name: **COLOR_LOOKUP**
   - New Sequence: **COLOR_LOOKUP_SEQ**

   Click **Next**.

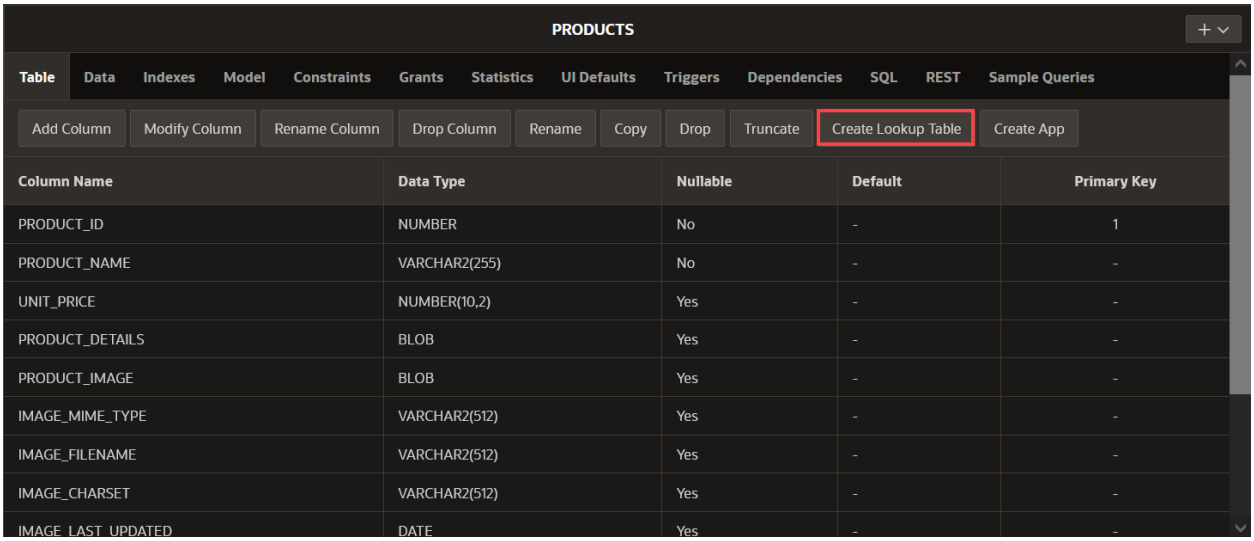


7. Click **Create Lookup Table**.

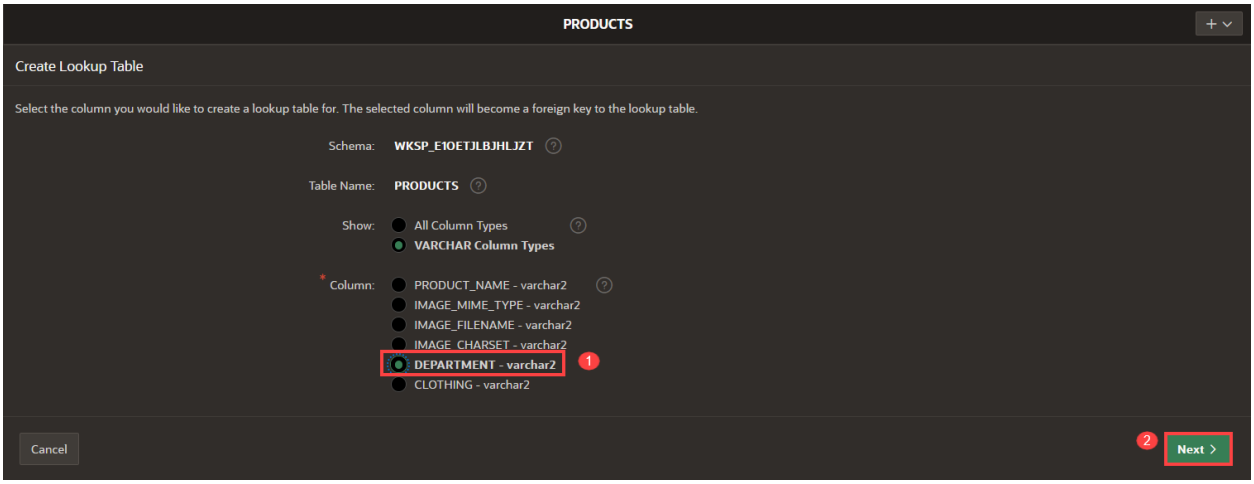


*Note: Click the **Create Lookup Table** button only once. Then you will find the new table listed in the Object Browser.*

8. To create **Department** lookup table, navigate back to the **Products** table and Click **Create Lookup Table** button.



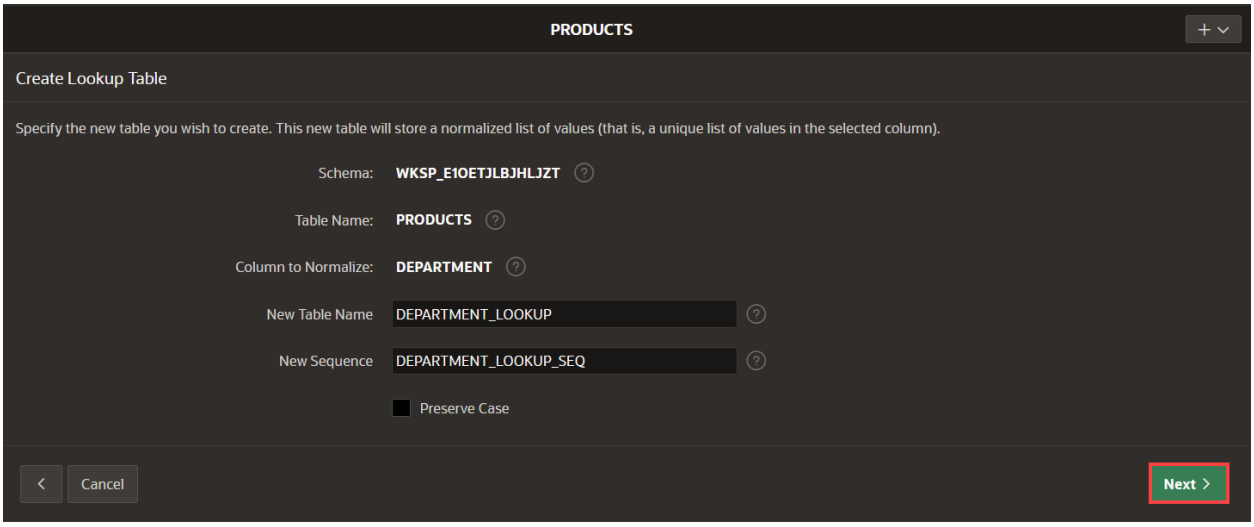
9. For Column, select **DEPARTMENT - varchar2**. Click **Next**.



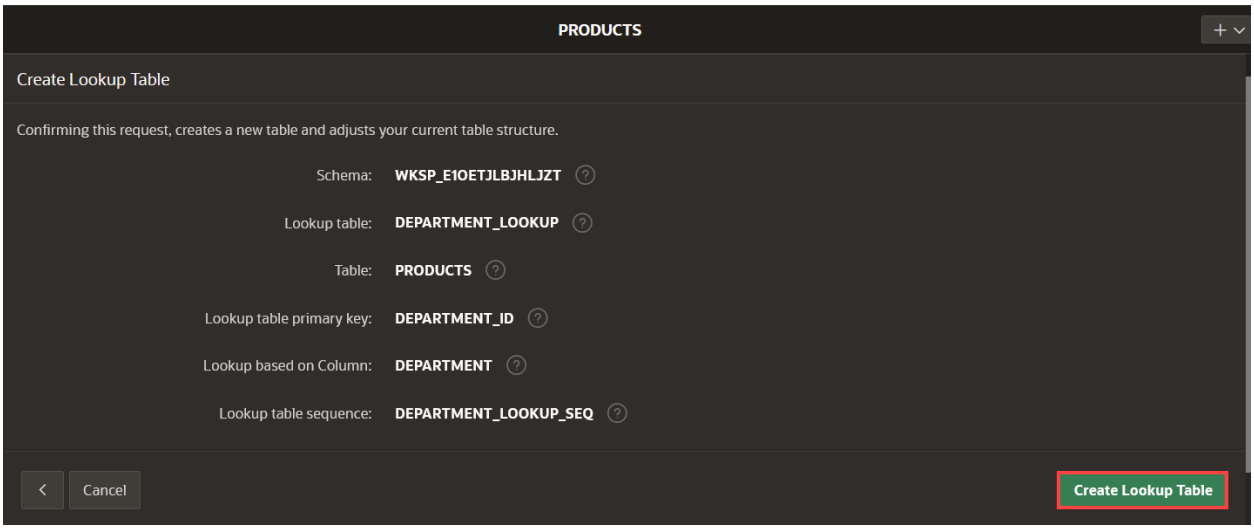
10. Leave the table and sequence name by default:

    - New Table Name: **DEPARTMENT_LOOKUP**
    - New Sequence: **DEPARTMENT_LOOKUP_SEQ**

    Click **Next**.

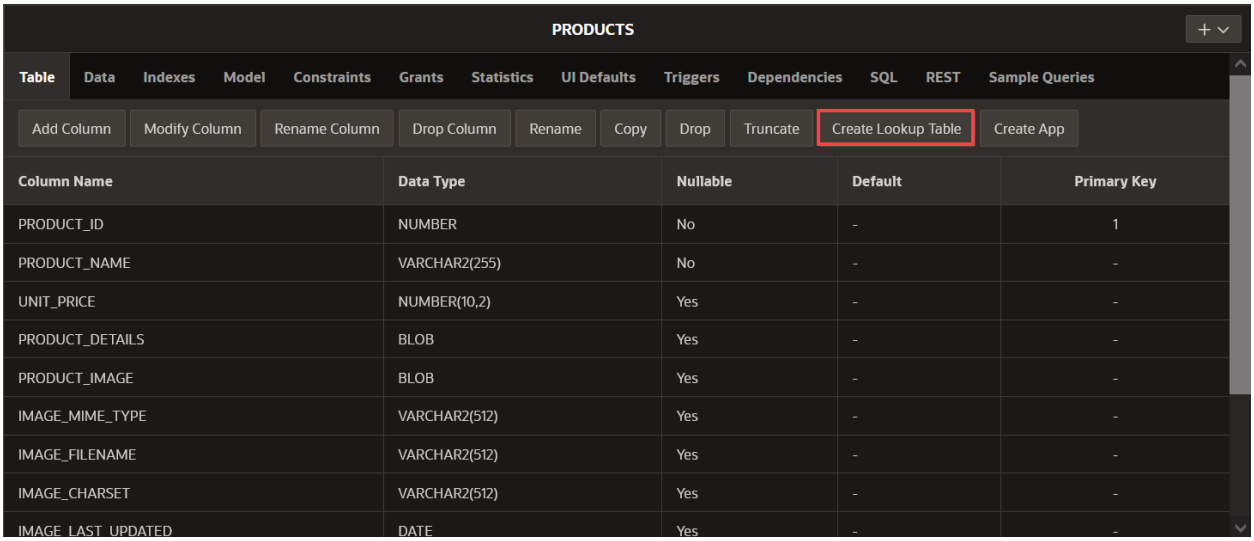


11. Click **Create Lookup Table**.

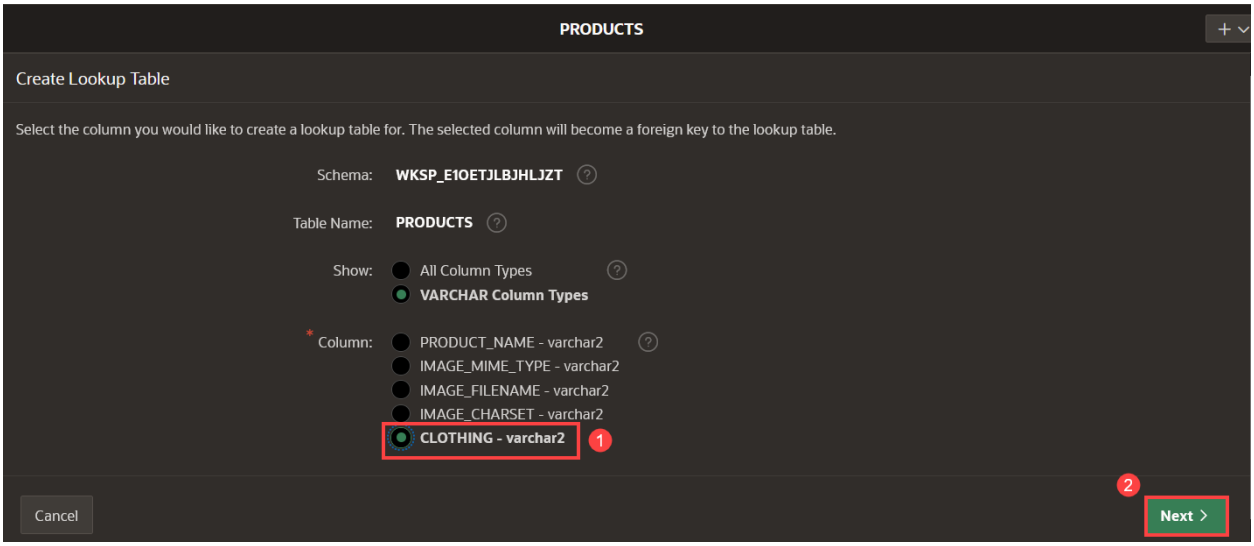


*Note: Click the **Create Lookup Table** button only once. Then you will find the new table listed in the Object Browser.*

12. To create **Clothing** lookup table, navigate back to the **Products** table and Click **Create Lookup Table** button.



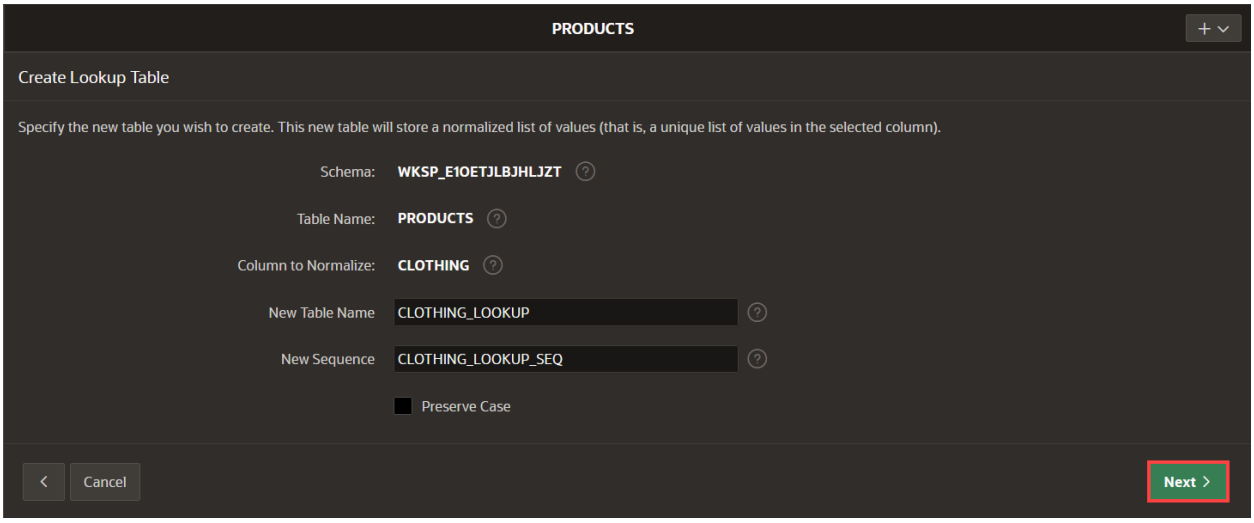
13. For Column, select **CLOTHING - varchar2**. Click **Next**.



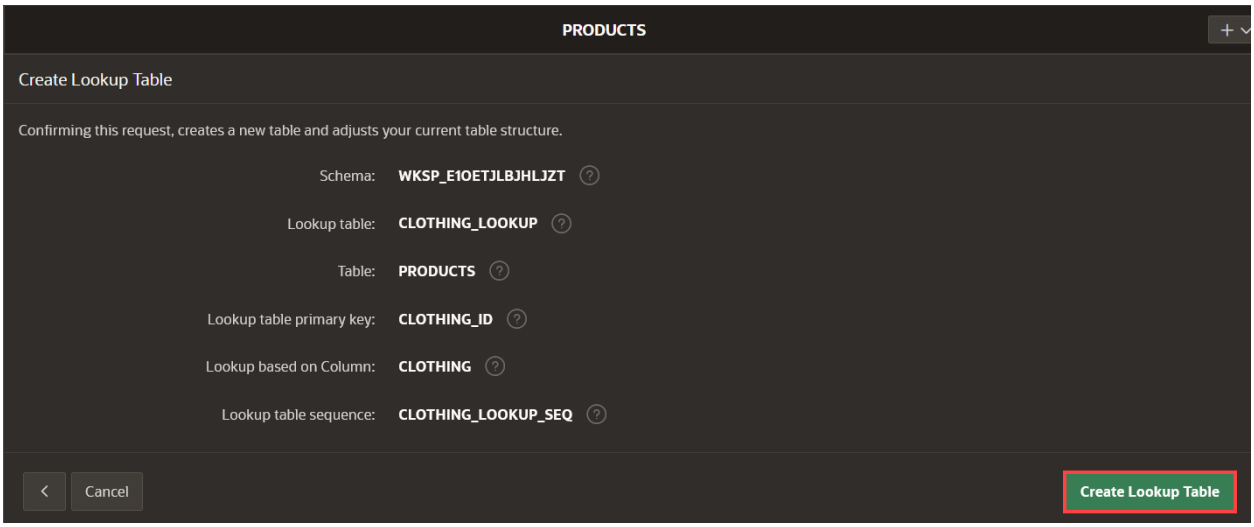
14. Leave the table and sequence name by default:

    - New Table Name: **CLOTHING_LOOKUP**
    - New Sequence: **CLOTHING_LOOKUP_SEQ**

    Click **Next**.



15. Click **Create Lookup Table**.

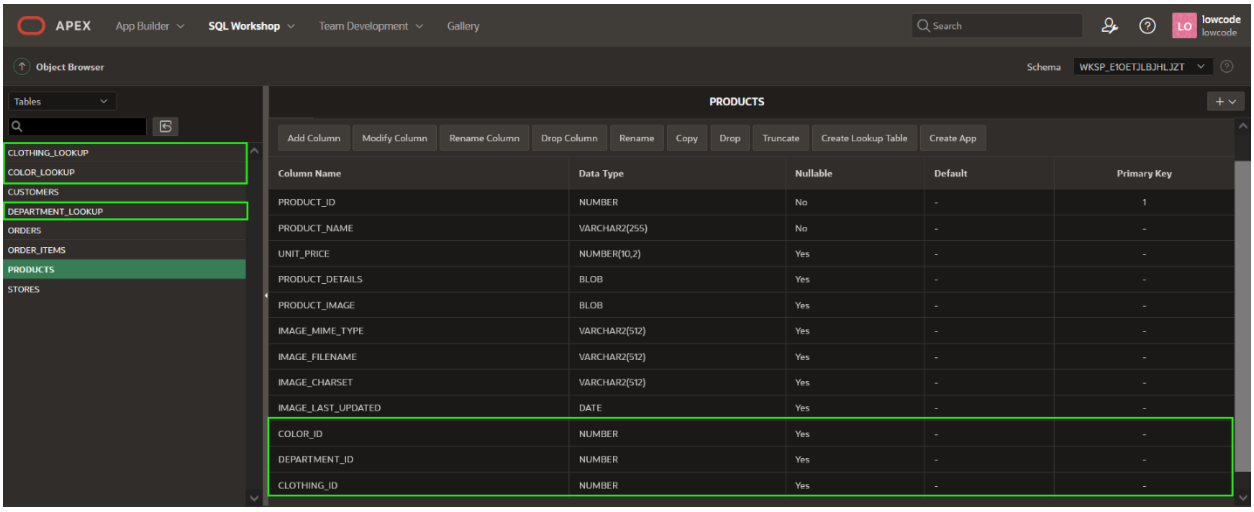


*Note: Click the **Create Lookup Table** button only once. Then you will find the new table listed in the Object Browser.*

16. The columns COLOR, DEPARTMENT, and CLOTHING in the **Products** table are renamed to COLOR_ID, DEPARTMENT_ID, and CLOTHING_ID respectively, and their data type changed to NUMBER. Also, there are new tables containing the values of the products:

    - COLOR_LOOKUP
    - DEPARTMENT_LOOKUP
    - CLOTHING_LOOKUP

The numeric value of the COLOR_ID column will now store a reference to the system-assigned unique id of a particular color, whose name is stored in the new COLOR_LOOKUP table. Similarly, the DEPARTMENT_ID column references the id of a row in the DEPARTMENT_LOOKUP table and CLOTHING_ID references the id of a row in the CLOTHING_LOOKUP table.



You now know how to add new columns to your existing tables, how to create lookup tables for reference information, and how to create and run a SQL script to populate your tables. You may now **proceed to the next lab**.

Create a Database Package for Business Logic

Introduction

In this lab, you learn to create database objects to use in your APEX application. This package contains functions and procedures to add products to the cart, remove products, create the order, clear the cart, and more.

To manage items in the cart, you will use collections, which enable you to temporarily store products currently in session state so they can be accessed, manipulated, or processed during a user's specific session.

Estimated Time: 5 minutes

Objectives

In this lab, you will:

- Create a package to manage the Shopping Cart.

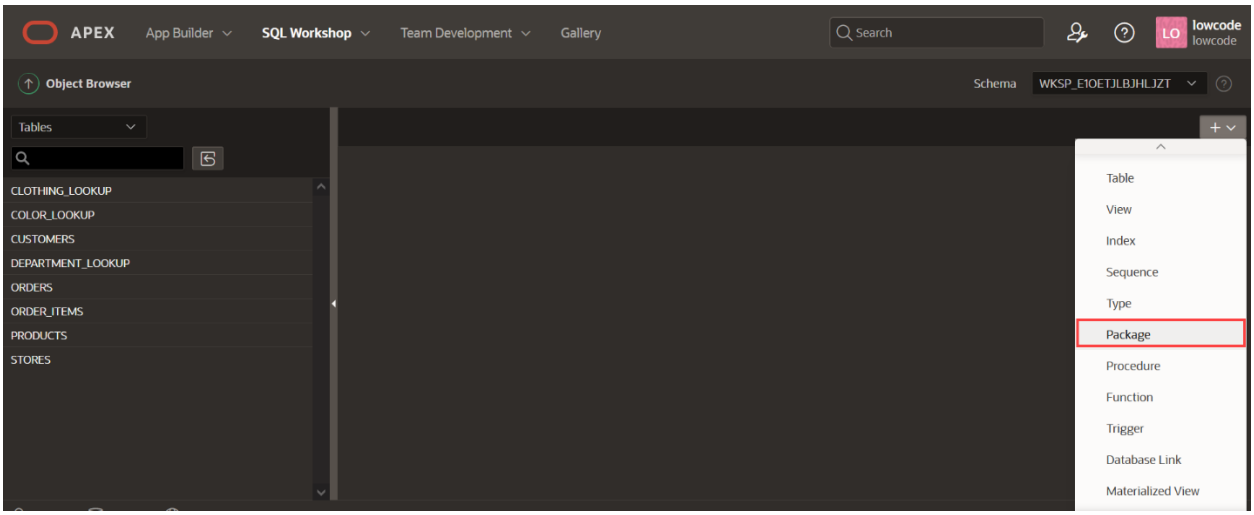
Collapse All Tasks

Task 1: Create the Package

Create specification and body for the package.

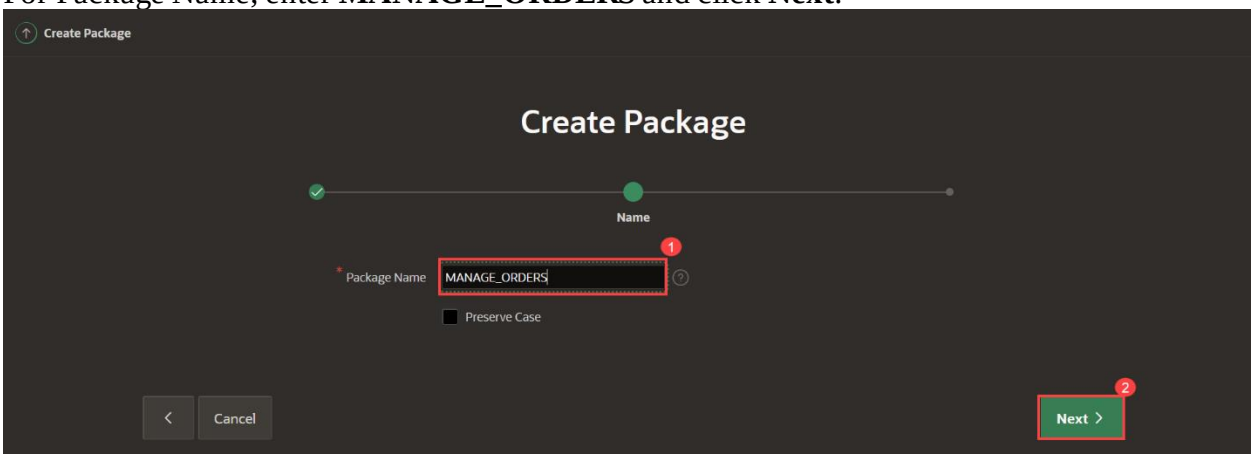
1. Navigate to **SQL Workshop**, click **Object Browser**.

2. Navigate to the + button on the right side, and click **Package**.



3. Select **Specification** and click **Next**.

4. For Package Name, enter **MANAGE_ORDERS** and click **Next**.



5. For Specification, enter the following:

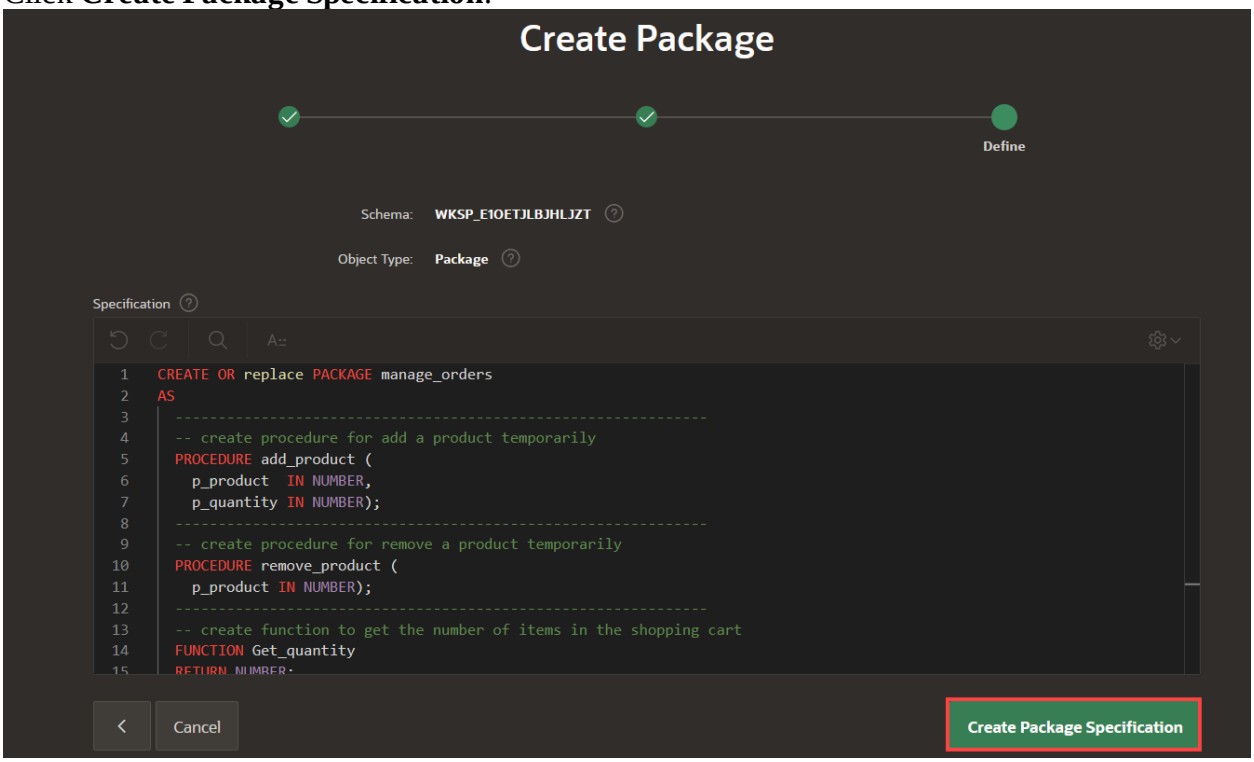
```
6. CopyCREATE OR replace PACKAGE manage_orders

7. AS

8.   ------------------------------------------------------------
   -----

9.   -- create procedure for add a product temporarily

10.   PROCEDURE add_product (
```

```
11.     p_product  IN NUMBER,
12.     p_quantity IN NUMBER);
13.   ------------------------------------------------------
  ------
14.   -- create procedure for remove a product temporarily
15.   PROCEDURE remove_product (
16.     p_product IN NUMBER);
17.   ------------------------------------------------------
  ------
18.   -- create function to get the number of items in the
  shopping cart
19.   FUNCTION Get_quantity
20.   RETURN NUMBER;
21.   ------------------------------------------------------
  ------
22.   -- create procedure for validate if a product exists in
  the shopping cart
23.   FUNCTION Product_exists(
24.     p_product IN NUMBER)
25.   RETURN NUMBER;
26.   ------------------------------------------------------
  ------
27.   -- create procedure for clear the cart
28.   PROCEDURE clear_cart;
```

```
29.   ------------------------------------------------------------
  ------

30.   -- create function to validate a customer

31.   FUNCTION Customer_exists(

32.     p_customer_email IN VARCHAR2)

33.   RETURN NUMBER;

34.   ------------------------------------------------------------
  ------

35.   -- create procedure to insert orders

36.   PROCEDURE create_order (

37.     p_customer       IN VARCHAR2 DEFAULT NULL,

38.     p_customer_email IN VARCHAR2,

39.     p_store          IN NUMBER,

40.     p_order_id       OUT orders.order_id%TYPE,

41.     p_customer_id    OUT NUMBER );
```

```
END manage_orders;
```

42. Click **Create Package Specification**.



43. Navigate to body part of the package by clicking on Body tab and enter the following:

```
44. CopyCREATE OR replace PACKAGE BODY manage_orders

45. AS

46.    PROCEDURE add_product (p_product   IN NUMBER,

47.                           p_quantity IN NUMBER)

48.    IS

49.    BEGIN

50.        IF NOT apex_collection.collection_exists
   (p_collection_name => 'PRODUCTS')

51.        THEN
```

```
52.
  apex_collection.create_collection(p_collection_name =>
  'PRODUCTS');

53.       END IF;

54.

55.       apex_collection.add_member(p_collection_name =>
  'PRODUCTS',

56.       p_n001 => p_product,

57.       p_n002 => p_quantity);

58.   END add_product;

59.

60.   PROCEDURE remove_product (p_product IN NUMBER)

61.   IS

62.     l_id NUMBER;

63.   BEGIN

64.       IF apex_collection.Collection_exists
  (p_collection_name => 'PRODUCTS')

65.       THEN

66.         SELECT seq_id

67.         INTO   l_id

68.         FROM   apex_collections a

69.         WHERE  collection_name = 'PRODUCTS'
```

```
70.                AND a.n001 = p_product;
71.
72.         apex_collection.delete_member(p_collection_name =>
   'PRODUCTS',
73.                                       p_seq => l_id);
74.       END IF;
75.   END remove_product;
76.
77.   FUNCTION get_quantity
78.   RETURN NUMBER
79.   IS
80.     l_items NUMBER := 0;
81.   BEGIN
82.       IF apex_collection.collection_exists
   (p_collection_name => 'PRODUCTS')
83.       THEN
84.         SELECT SUM(n002)
85.         INTO   l_items
86.         FROM   apex_collections a
87.         WHERE  collection_name = 'PRODUCTS';
88.       END IF;
```

```
89.
90.       RETURN l_items;
91.   END get_quantity;
92.
93.   FUNCTION product_exists(p_product IN NUMBER)
94.   RETURN NUMBER
95.   IS
96.     l_quantity NUMBER;
97.   BEGIN
98.       IF apex_collection.collection_exists
  (p_collection_name => 'PRODUCTS')
99.       THEN
100.              SELECT a.n002
101.              INTO   l_quantity
102.              FROM   apex_collections a
103.              WHERE  collection_name = 'PRODUCTS'
104.                    AND a.n001 = p_product;
105.
106.              RETURN l_quantity;
107.            ELSE
108.              RETURN 0;
```

```
109.            END IF;
110.        EXCEPTION
111.          WHEN OTHERS THEN
112.                    RETURN 0;
113.        END product_exists;
114.
115.        PROCEDURE clear_cart
116.        IS
117.        BEGIN
118.            IF apex_collection.collection_exists
  (p_collection_name => 'PRODUCTS')
119.            THEN
120.
  apex_collection.truncate_collection(p_collection_name =>
  'PRODUCTS');
121.            END IF;
122.        END clear_cart;
123.
124.        FUNCTION customer_exists(p_customer_email IN
  VARCHAR2)
125.        RETURN NUMBER
126.        IS
```

```
127.          l_customer customers.customer_id%TYPE;
128.        BEGIN
129.            SELECT customer_id
130.            INTO   l_customer
131.            FROM   customers
132.            WHERE  email_address = p_customer_email;
133.
134.            RETURN l_customer;
135.        EXCEPTION
136.          WHEN no_data_found THEN
137.                    RETURN 0;
138.        END customer_exists;
139.
140.        PROCEDURE create_order (p_customer       IN
  VARCHAR2,
141.                                p_customer_email IN
  VARCHAR2,
142.                                p_store          IN
  NUMBER,
143.                                p_order_id       OUT
  orders.order_id%TYPE,
144.                                p_customer_id    OUT
  NUMBER)
```

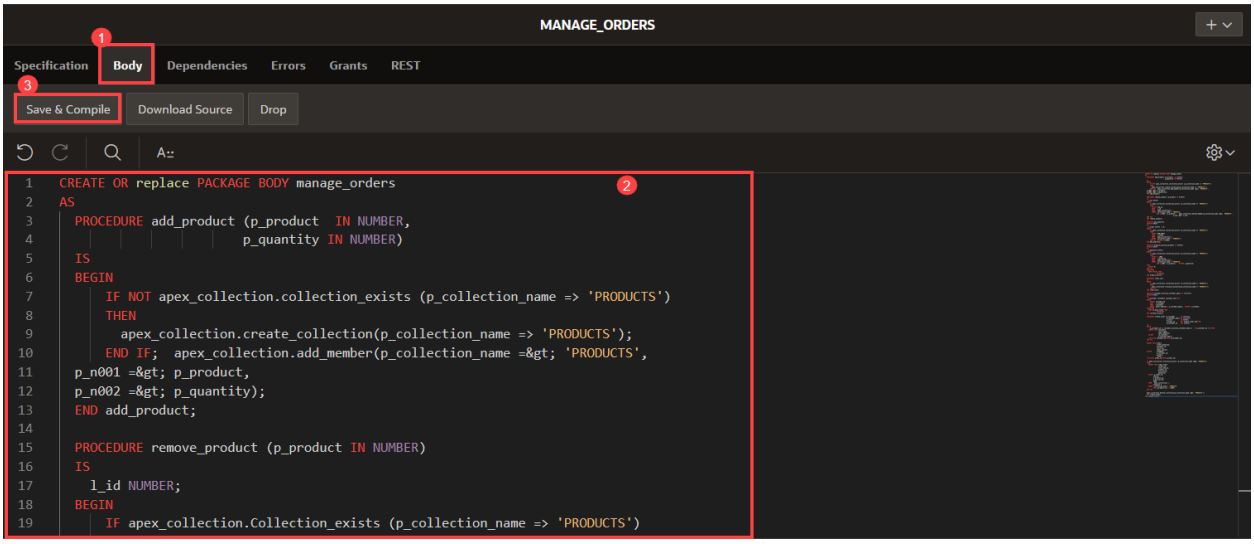
```
145.        IS
146.        BEGIN
147.            p_customer_id :=
  customer_exists(p_customer_email);
148.
149.            IF p_customer_id = 0 THEN
150.              INSERT INTO customers
151.                          (full_name,
152.                          email_address)
153.              VALUES      (p_customer,
154.                          p_customer_email)
155.              returning customer_id INTO p_customer_id;
156.            END IF;
157.
158.            INSERT INTO orders
159.                        (order_datetime,
160.                        customer_id,
161.                        store_id,
162.                        order_status)
163.            VALUES      (SYSDATE,
164.                        p_customer_id,
```

```
165.                         p_store,
166.                         'OPEN')
167.             returning order_id INTO p_order_id;
168.
169.             IF apex_collection.collection_exists
   (p_collection_name => 'PRODUCTS')
170.             THEN
171.               INSERT INTO order_items
172.                           (order_id,
173.                           line_item_id,
174.                           product_id,
175.                           unit_price,
176.                           quantity)
177.               SELECT p_order_id,
178.                     seq_id,
179.                     p.product_id,
180.                     p.unit_price,
181.                     n002
182.               FROM   apex_collections a,
183.                     products p
184.               WHERE  collection_name = 'PRODUCTS'
```

```
185.                     AND p.product_id = a.n001;

186.             END IF;

187.

188.
   apex_collection.delete_collection(p_collection_name =>
   'PRODUCTS');

189.         END create_order;
```

```
END manage_orders;
```

190. Click **Save & Compile**.



While you don't have to understand the code to complete the workshop successfully, know that the functions and procedures you've defined in this lab are using a built-in feature of Oracle APEX to handle the user's shopping cart by managing a collection of product id and quantity values specific to the current user, and automatically create new row in the **Customers** table during order creation if it's the first time the user is placing an order.

You now know how to create a package to manage the shopping cart. In the following labs, you will call these procedures and functions when it is required. You may now **proceed to the next lab**.

Create the Application

Create the application

Introduction

In this lab, you build an application based on the data structures you built in previous labs.

Estimated Time: 15 minutes

Objectives

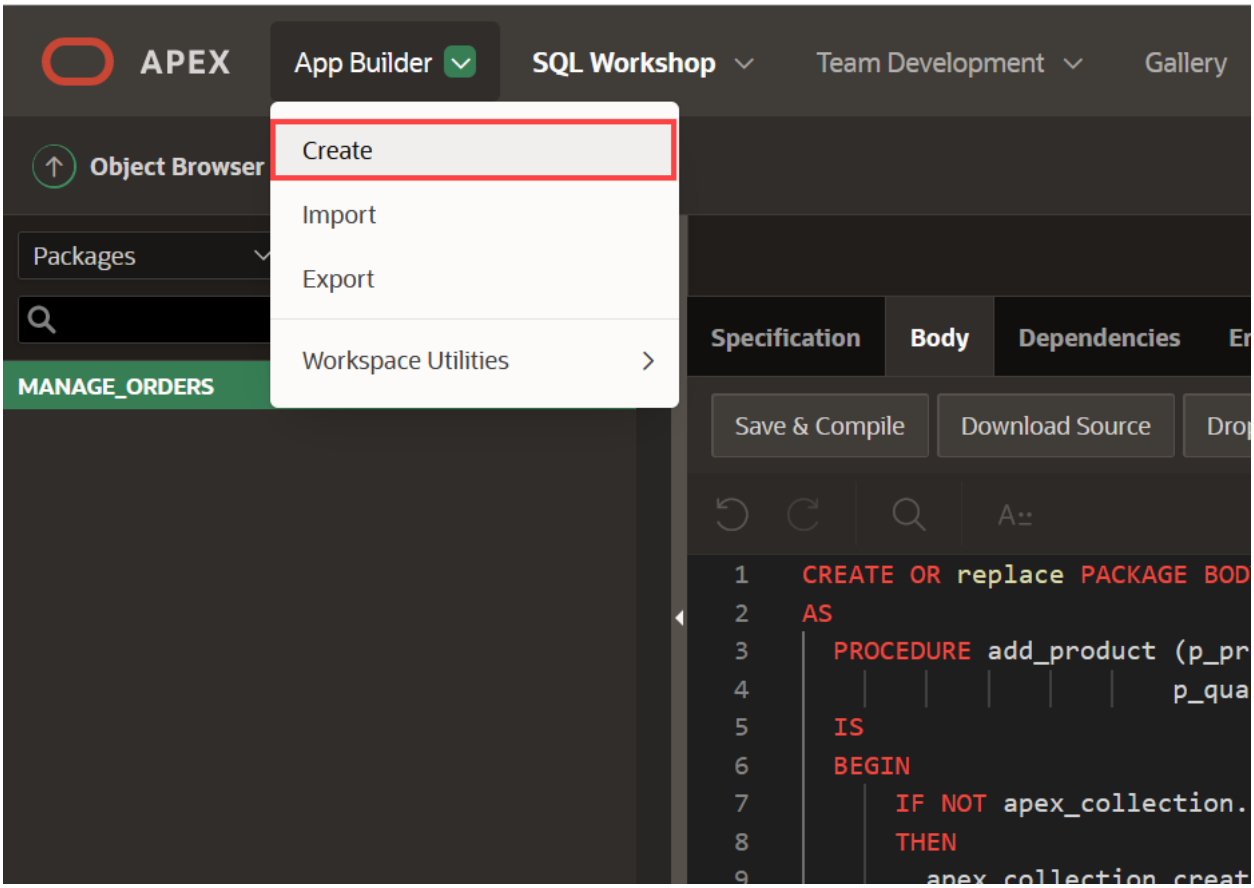
In this lab, you will:

- Create an application using the tables and data that you already have installed.

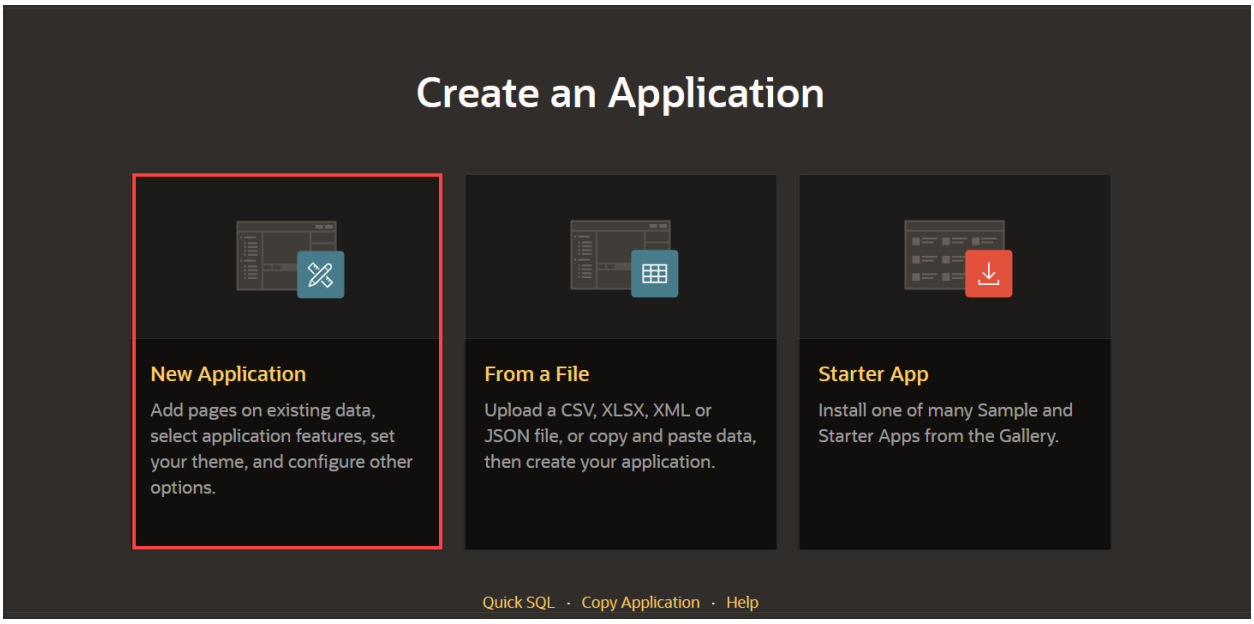
Collapse All Tasks

Task 1: Create an Application

1. In the App Builder menu, click **App Builder** and then, click **Create**.

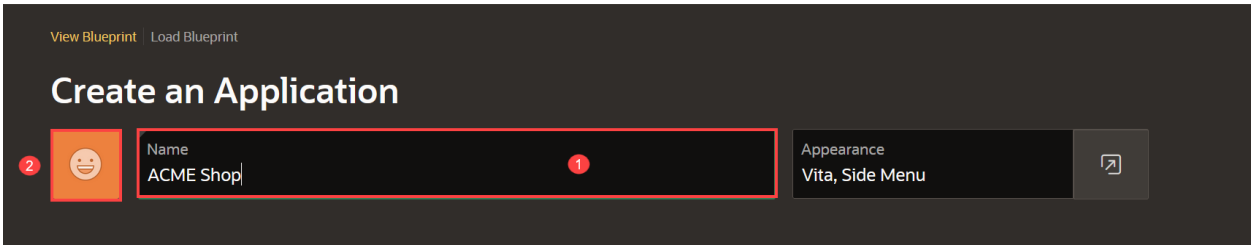


2. Click **New Application**.

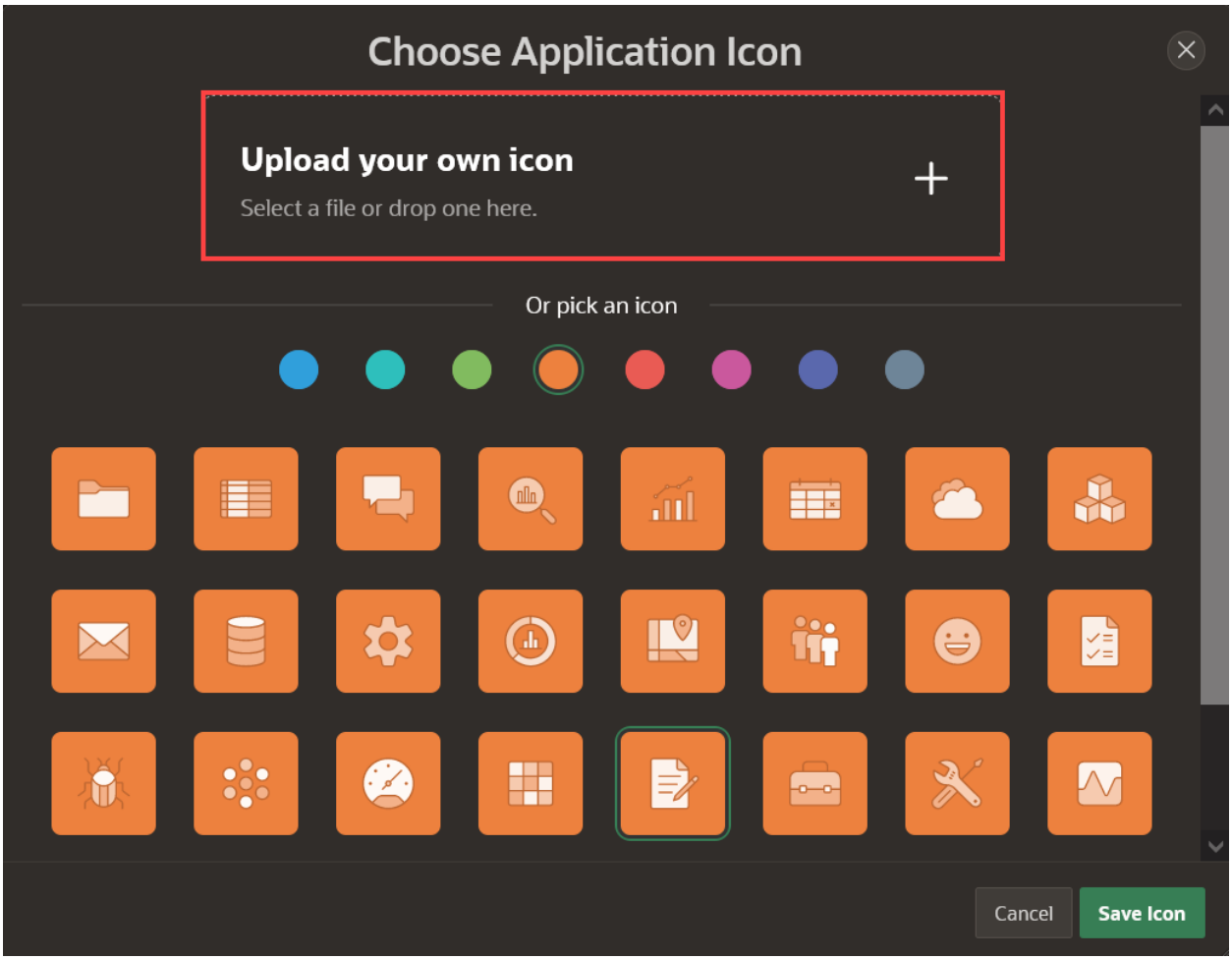


Task 2: Add a Name and Icon to the Application

1. In the Create Application wizard, for Name, enter **ACME Shop** and click the application icon.



2. Click **Upload your own icon** to select an icon or simply drag and drop the image. Download a sample icon here.



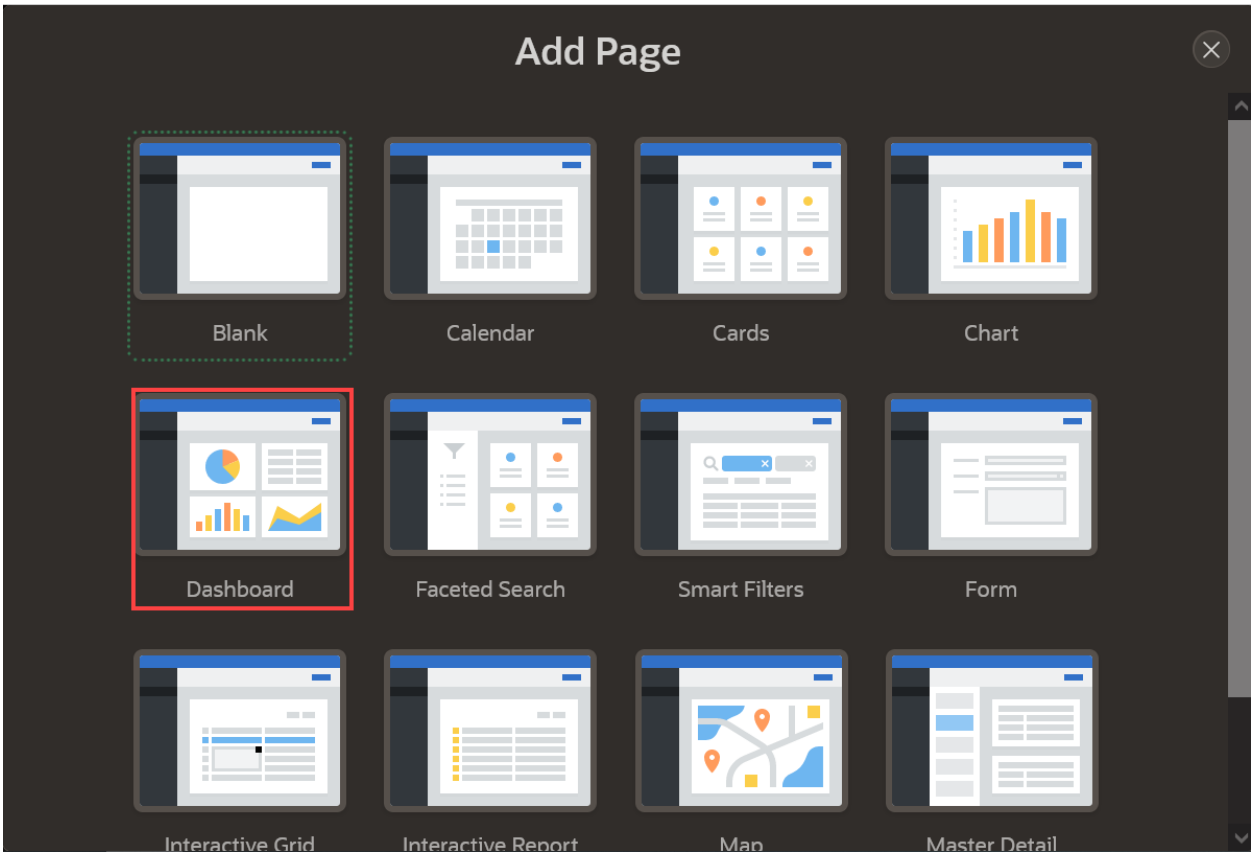
3. Adjust the icon if needed and click **Save Icon**.

Task 3: Add the Dashboard Page

A dashboard page is a great way to show important information using various charts. When you installed the Sample Dataset, it also created a number of views, which join data from various tables. These views are ideal as the basis for the dashboard charts.

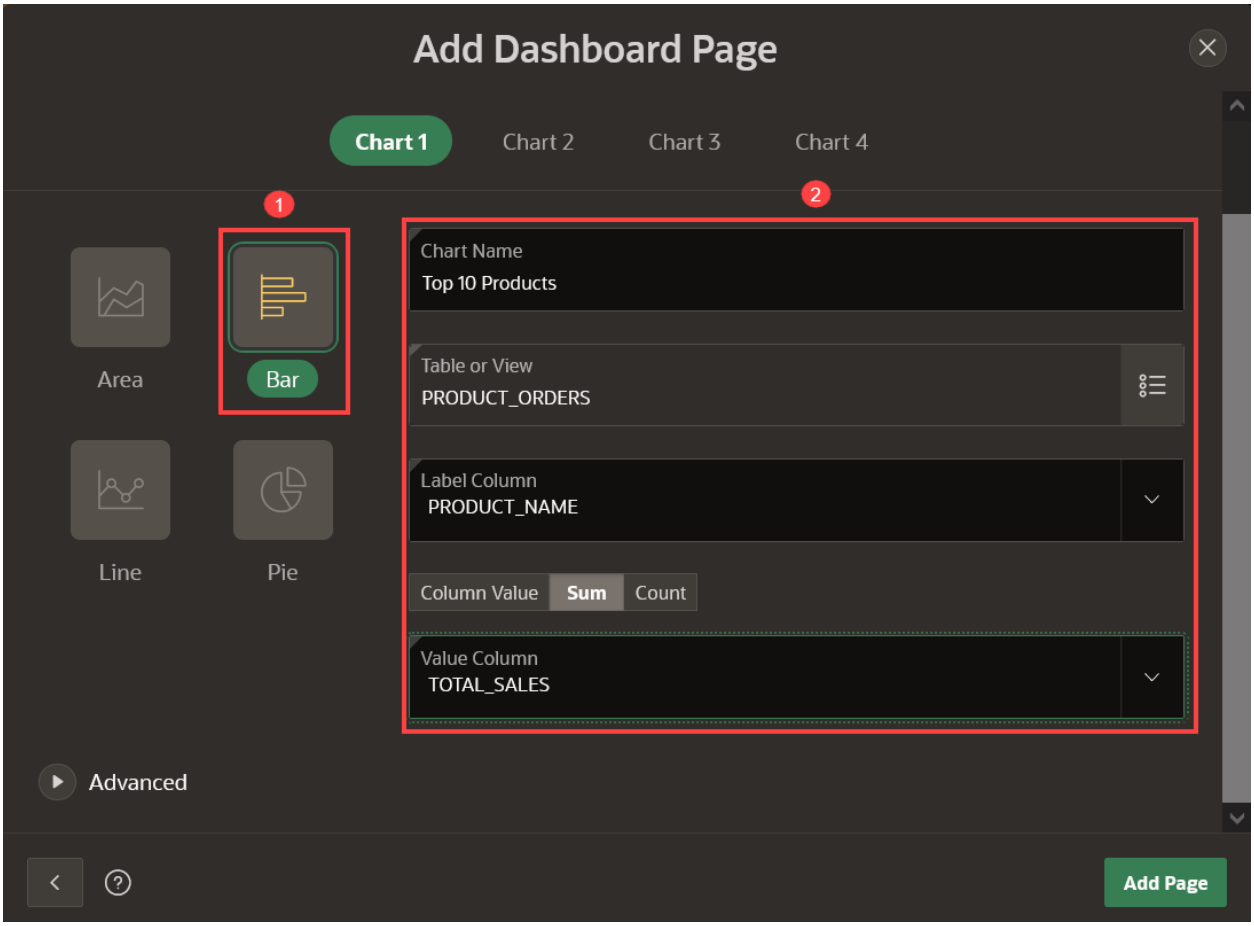
1. In the Create Application wizard, click **Add Page**.

2. Click **Dashboard**.



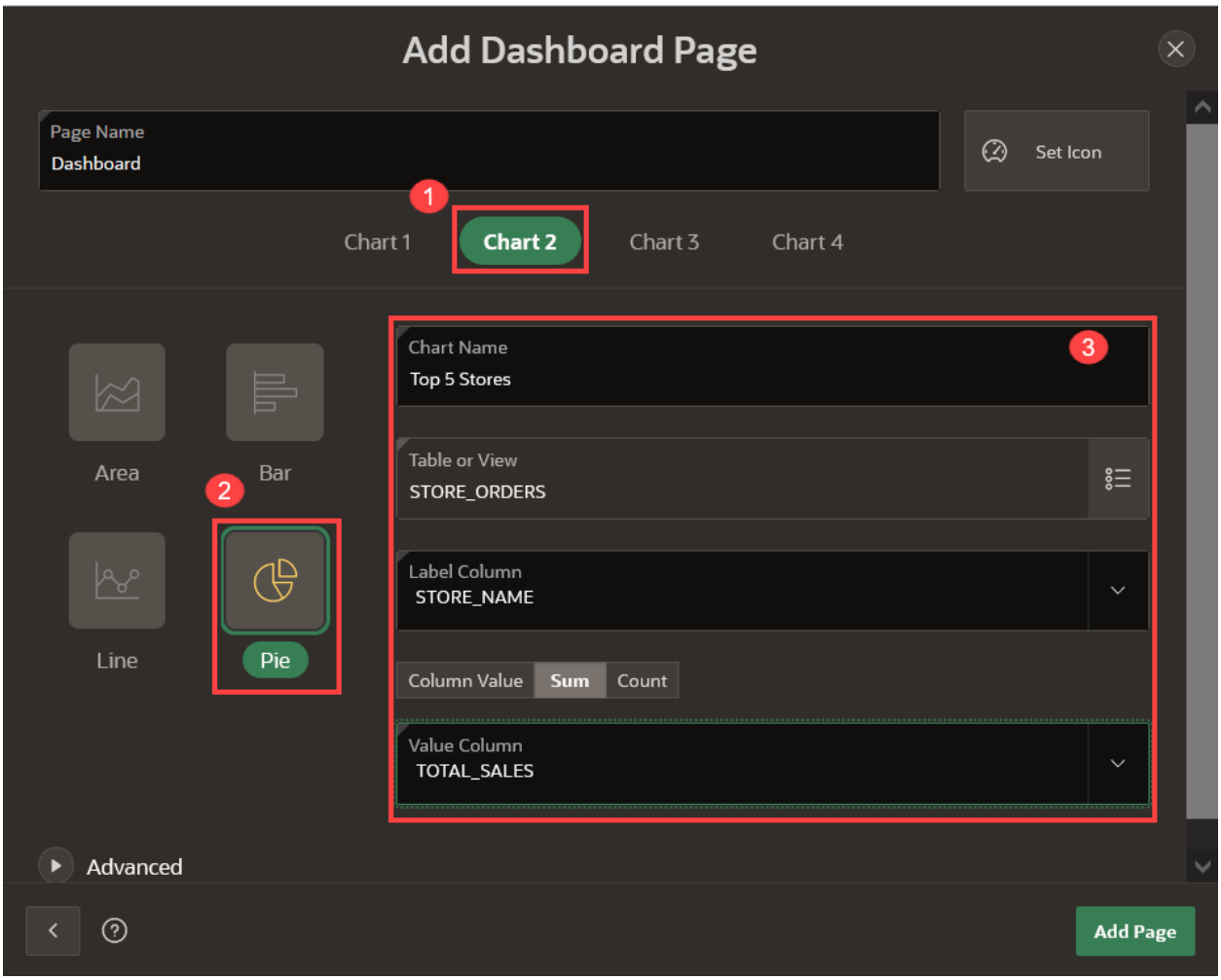
3. For Chart 1, enter the following:

    - Chart Type – select **Bar**
    - Chart Name – enter **Top 10 Products**
    - Table or View – select **PRODUCT_ORDERS**
    - Label Column – select **PRODUCT_NAME**
    - Type – select **Sum**
    - Value Column – select **TOTAL_SALES**.



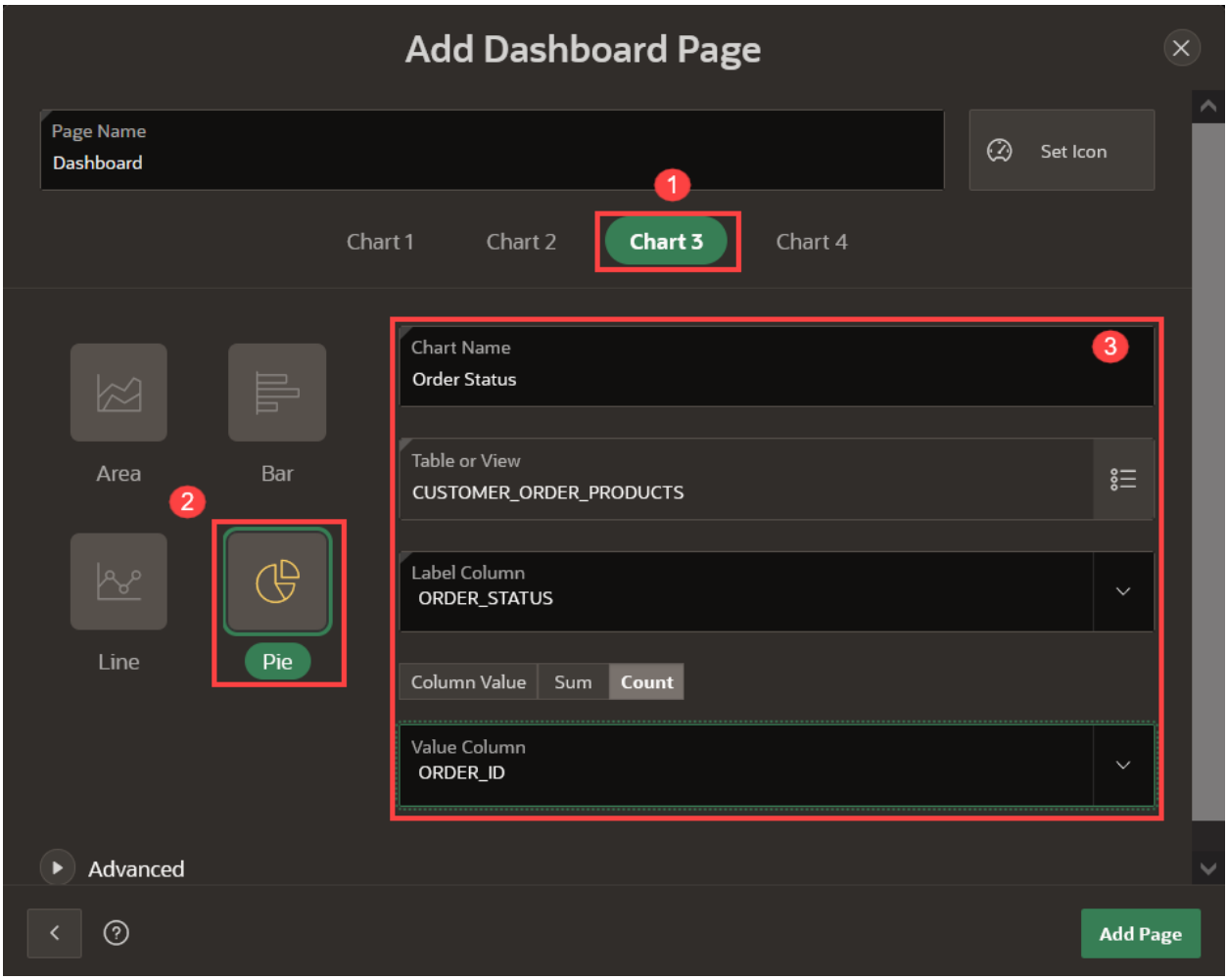
4. Click Chart 2, and enter the following:

   - Chart Type – select **Pie**
   - Chart Name – enter **Top 5 Stores**
   - Table or View – select **STORE_ORDERS**
   - Label Column – select **STORE_NAME**
   - Type – select **Sum**
   - Value Column – select **TOTAL_SALES**.



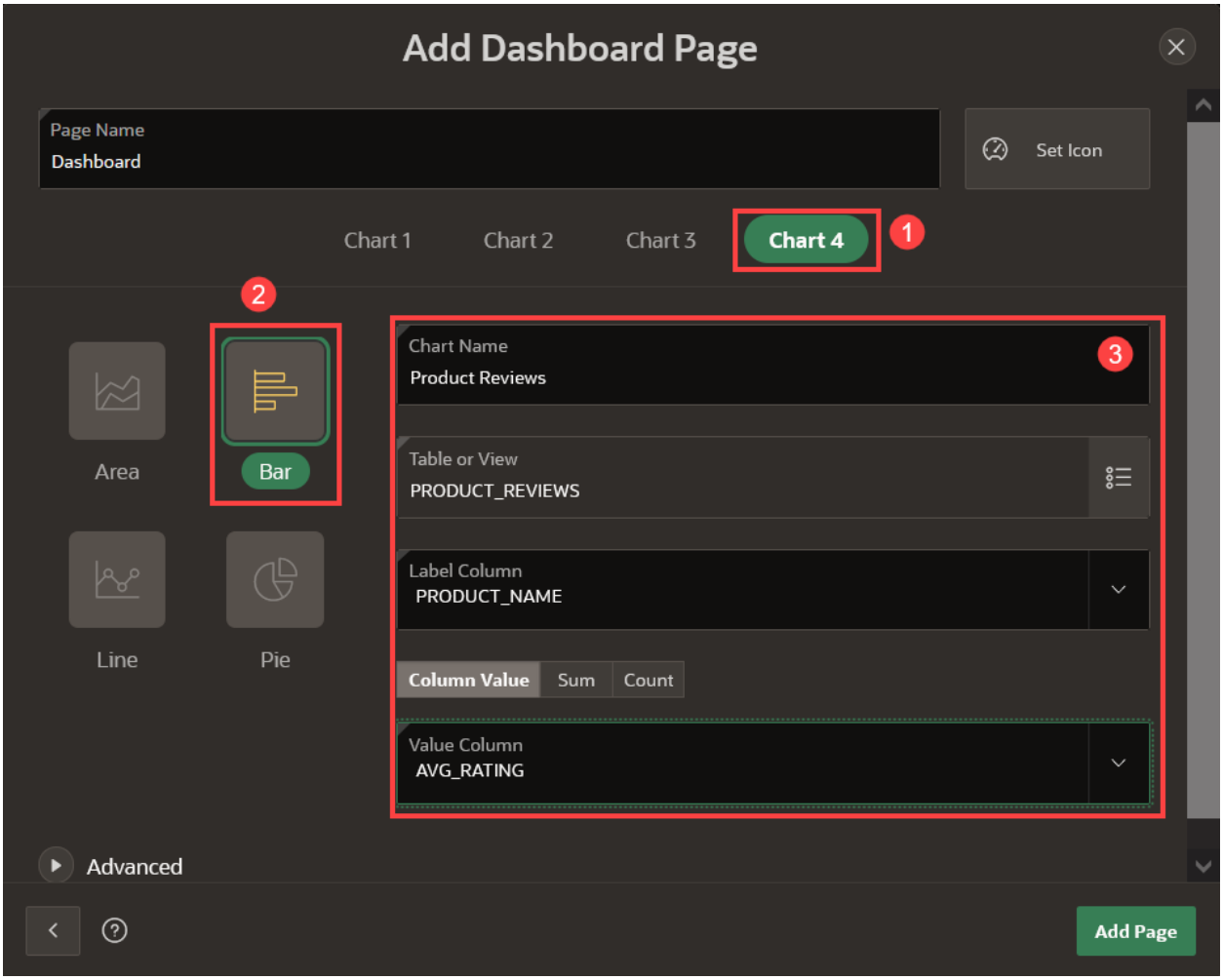
5. Click Chart 3, and enter the following:

   - Chart Type – select **Pie**
   - Chart Name – enter **Order Status**
   - Table or View – select **CUSTOMER_ORDER_PRODUCTS**
   - Label Column – select **ORDER_STATUS**
   - Type – select **Count**
   - Value Column – select **ORDER_ID**.



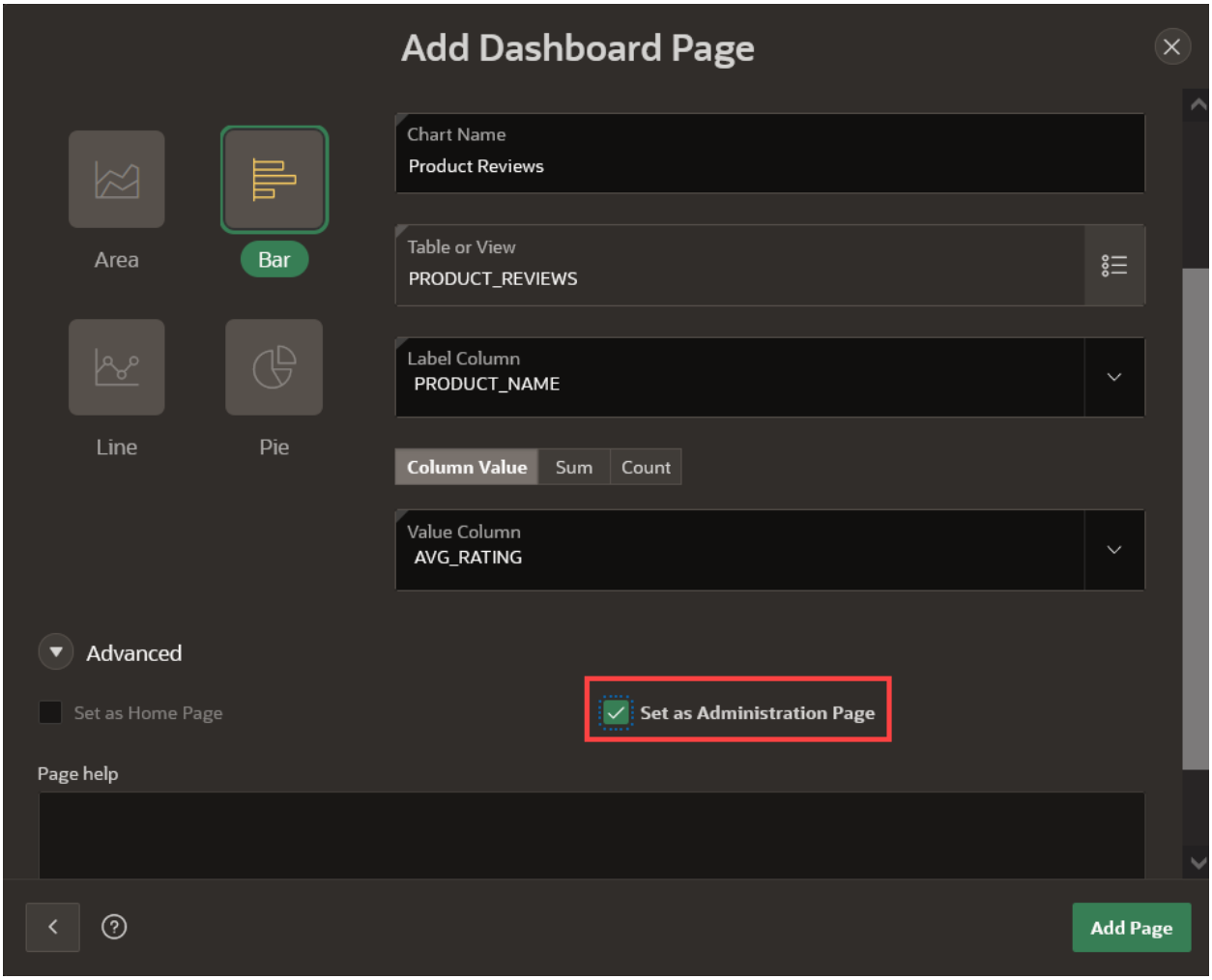
6. Click Chart 4, and enter the following:

   - Chart Type – select **Bar**
   - Chart Name – enter **Product Reviews**
   - Table or View – select **PRODUCT_REVIEWS**
   - Label Column – select **PRODUCT_NAME**
   - Type – select **Column Value**
   - Value Column – select **AVG_RATING**.



7. Click Advanced and check **Set as Administration Page**.

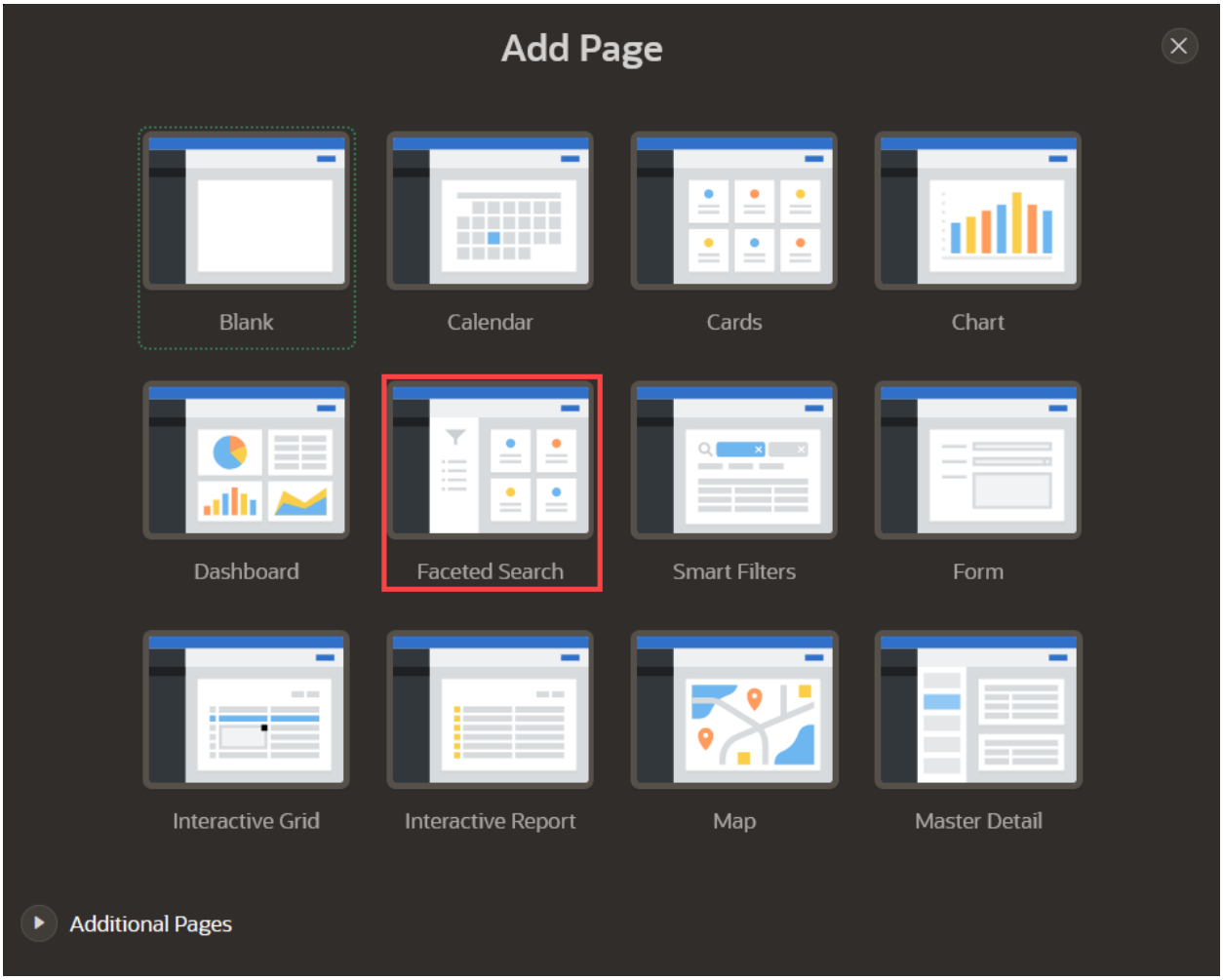
   Setting a page as Administration Page will make that only certain users can run the page. In this case, only users with Administration Rights will be able to run the Dashboard page, which means that this page will require authentication.



8. Click **Add Page**.

Task 4: Add the Products Page

1. In the Create Application wizard, click **Add Page**.
2. Click **Faceted Search**.

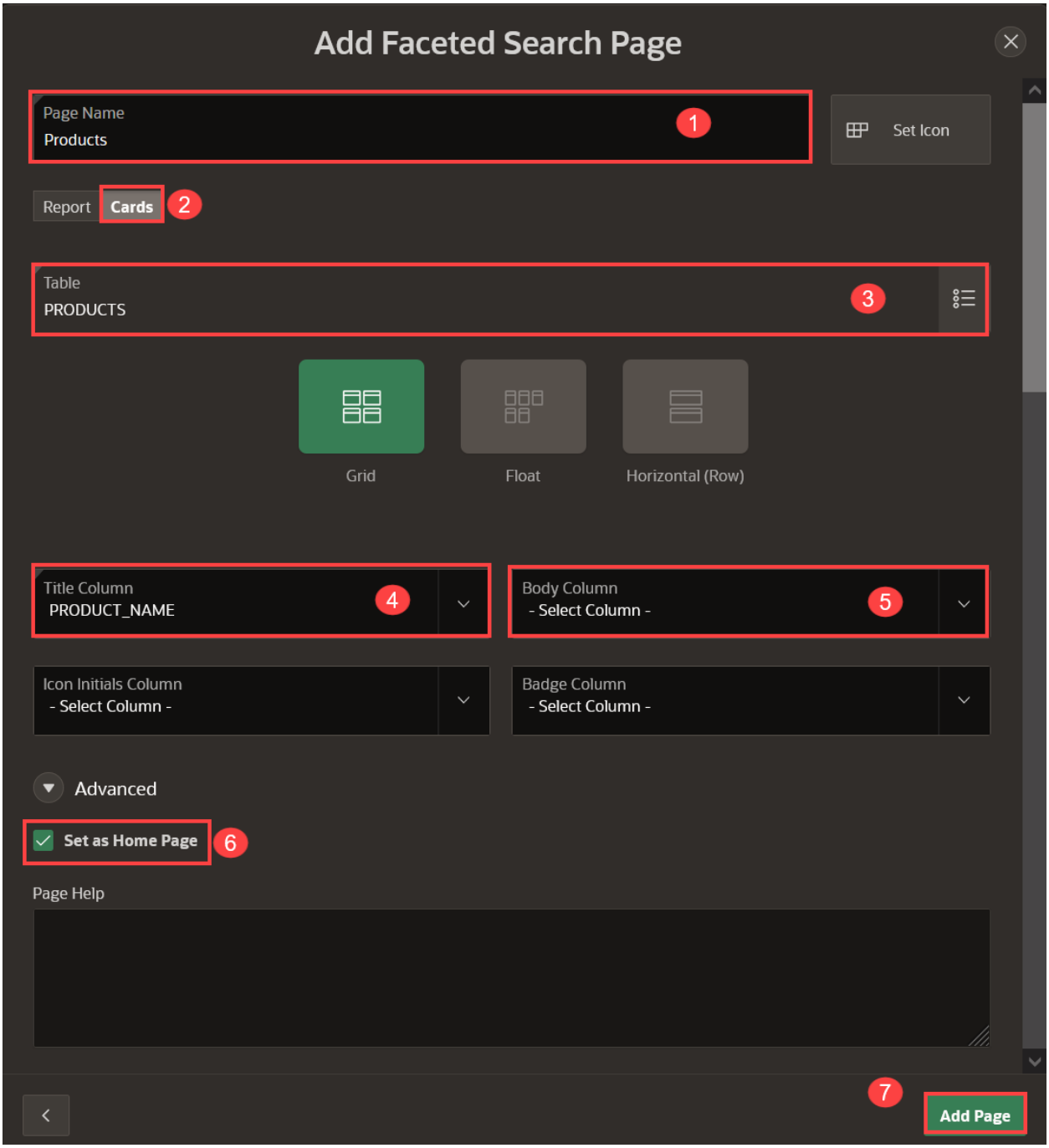


3. On the Faceted Search Page, enter the following:

   - Page Name - enter **Products**
   - Select **Cards**
   - Table - select **PRODUCTS**
   - Select **Grid**
   - Title Column - select **PRODUCT_NAME**
   - Body Column - select **- Select Column -** (to unselect the default column chosen)

   Expand Advanced Section and check **Set as Home Page**

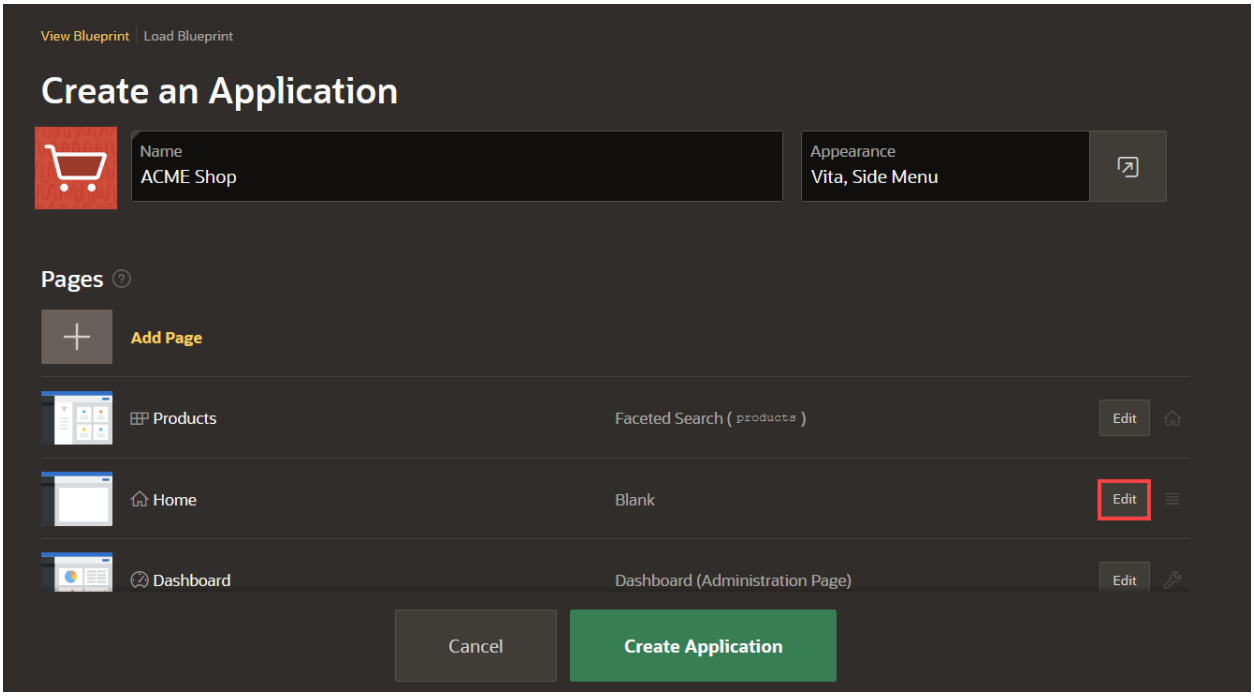
   Click **Add Page**.



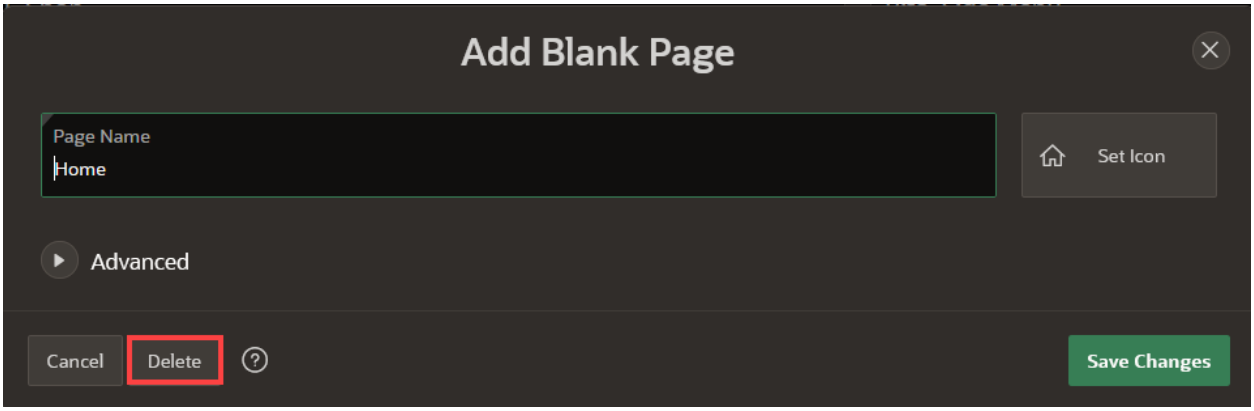
Task 5: Delete Original Home Page

Since we designated the product faceted search page as the application's home page, we no longer need the default home page that the wizard included in the list of pages. Therefore, in this task, we will remove the original home page.

1. Navigate to the original page named "Home" and click **Edit**.



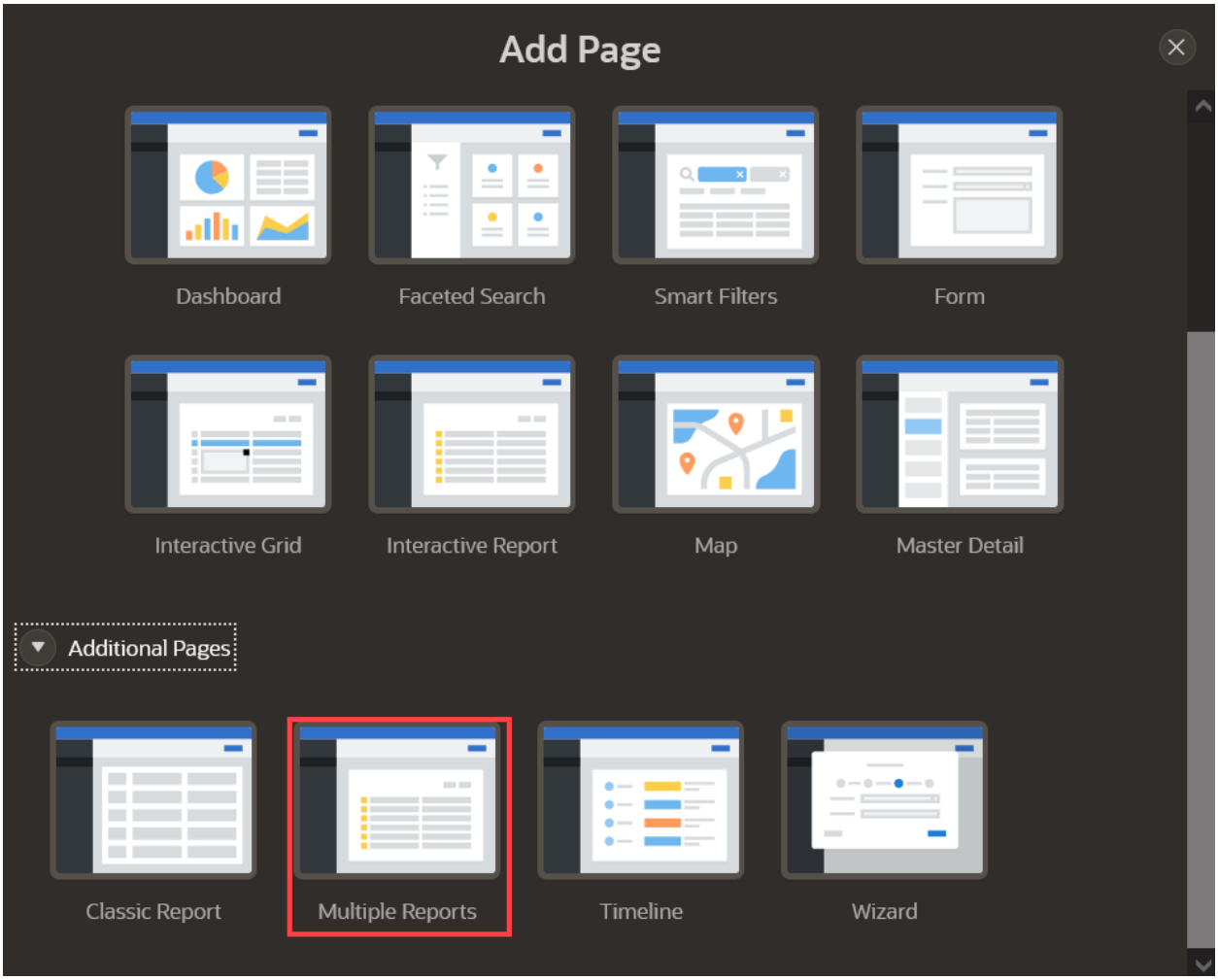
2. Click **Delete**.



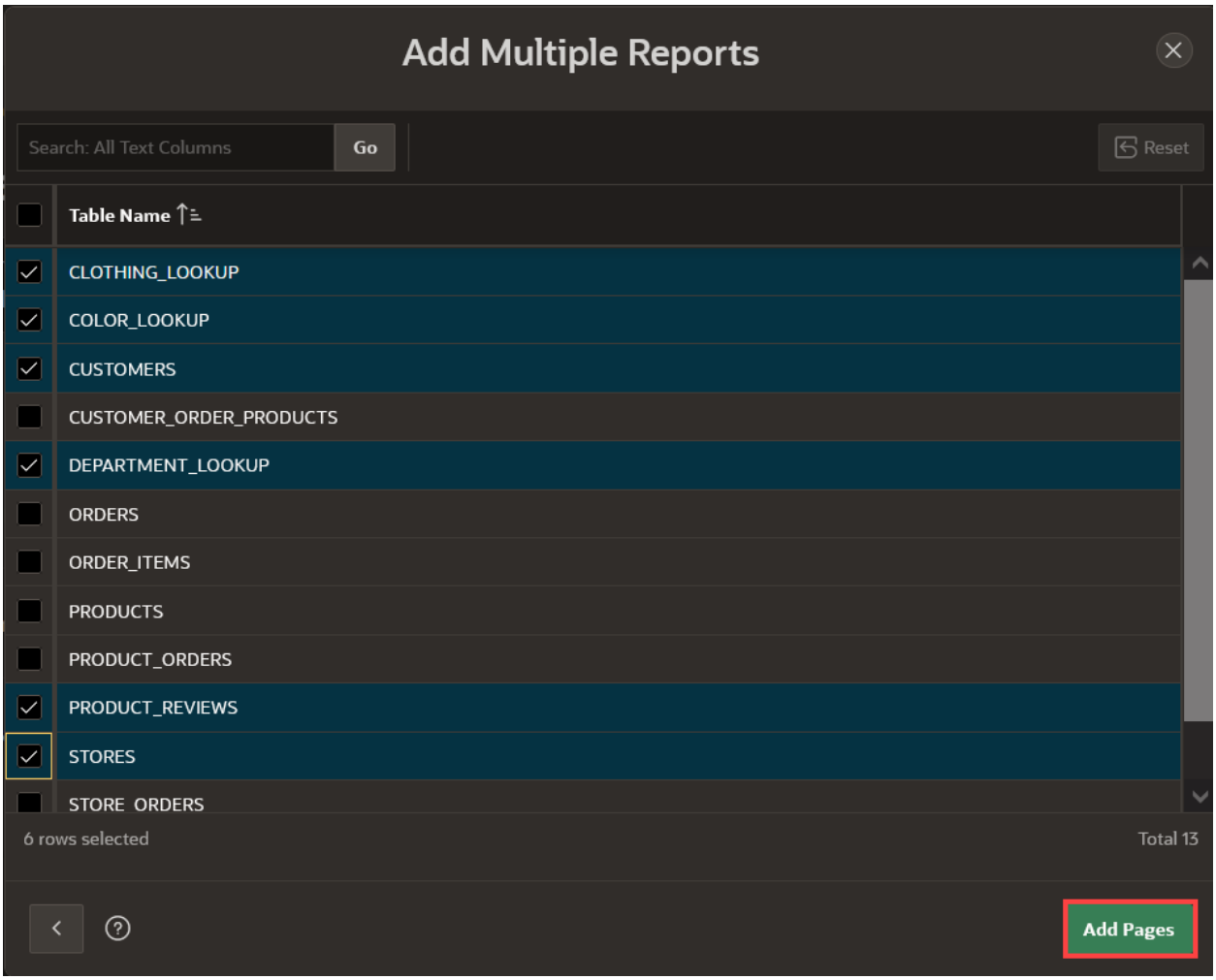
3. Confirm the deletion by clicking **OK** on the dialog.

Task 6: Add Multiple Reports

1. In the Create Application wizard, click **Add Page**.
2. Click the arrow to the left of **Additional Pages** to see additional page types.
3. Click **Multiple Reports**.

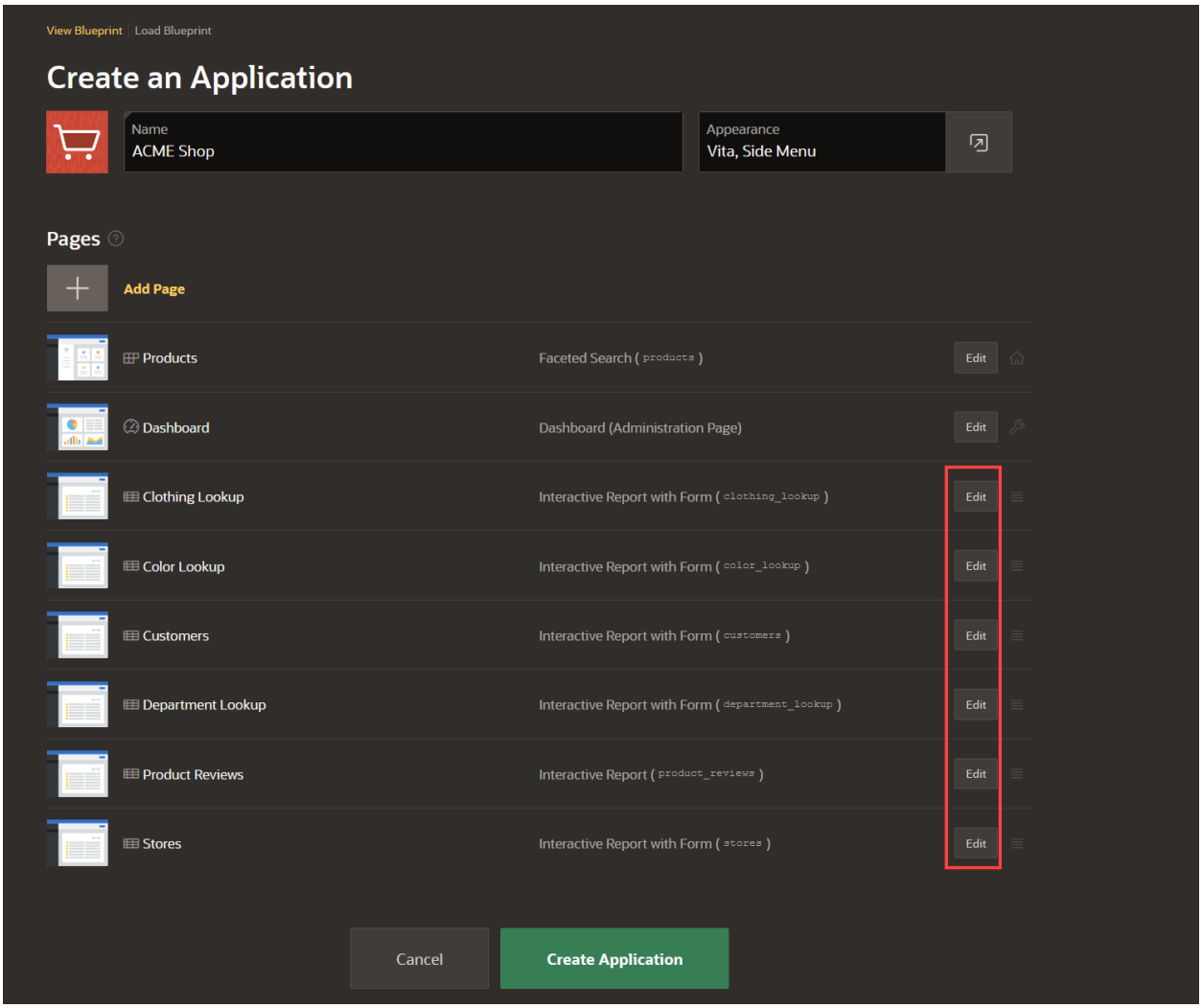


4. On the Create Multiple Reports Page, select the following tables:
    - CLOTHING_LOOKUP
    - COLOR_LOOKUP
    - CUSTOMERS
    - DEPARTMENT_LOOKUP
    - PRODUCT_REVIEWS
    - STORES
5. Click **Add Pages**.



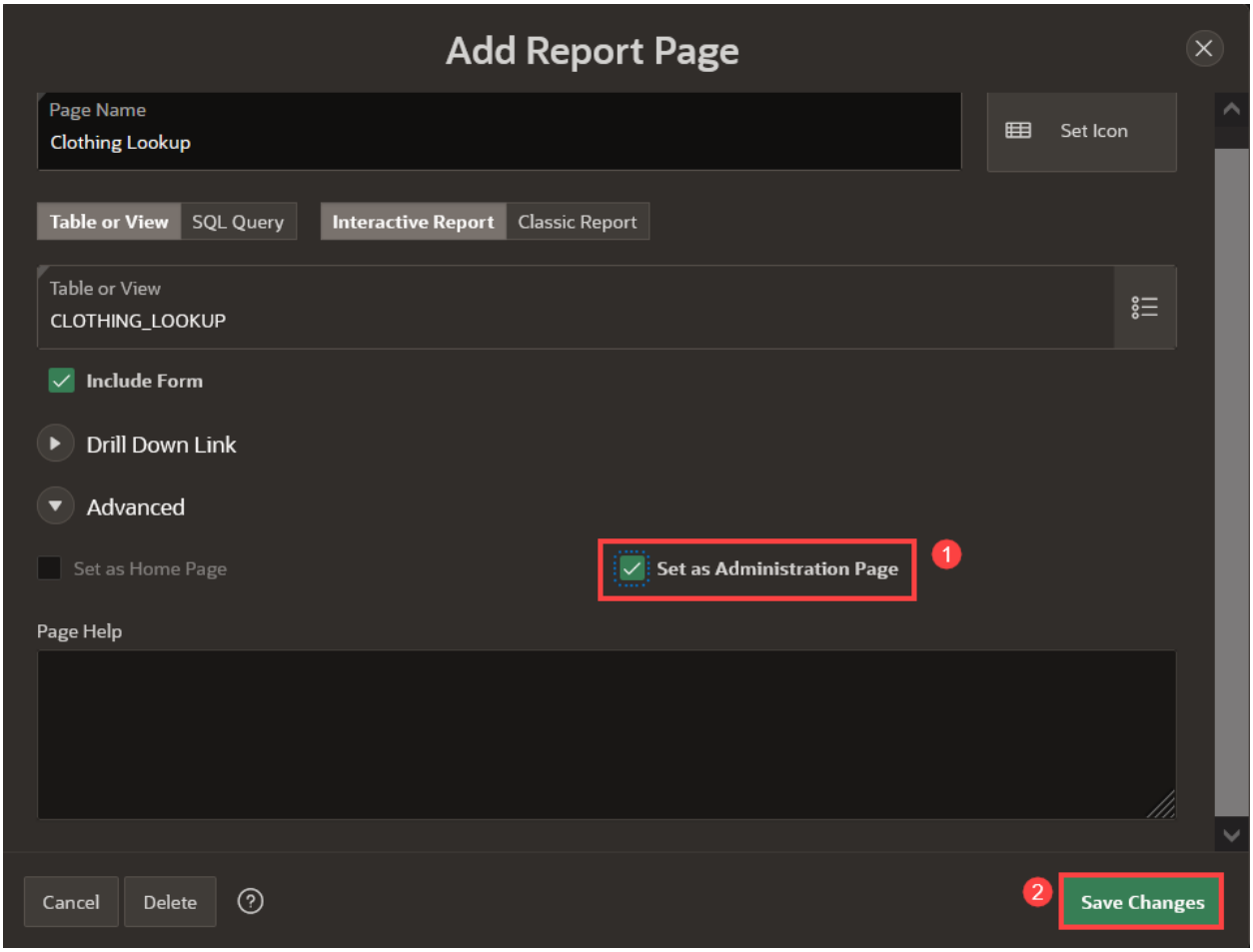
Task 7: Set Multiple Reports as Administration Pages

1. Edit each of the following pages to set it as an Administration Page:
    - CLOTHING_LOOKUP
    - COLOR_LOOKUP
    - CUSTOMERS
    - DEPARTMENT_LOOKUP
    - PRODUCT_REVIEWS
    - STORES



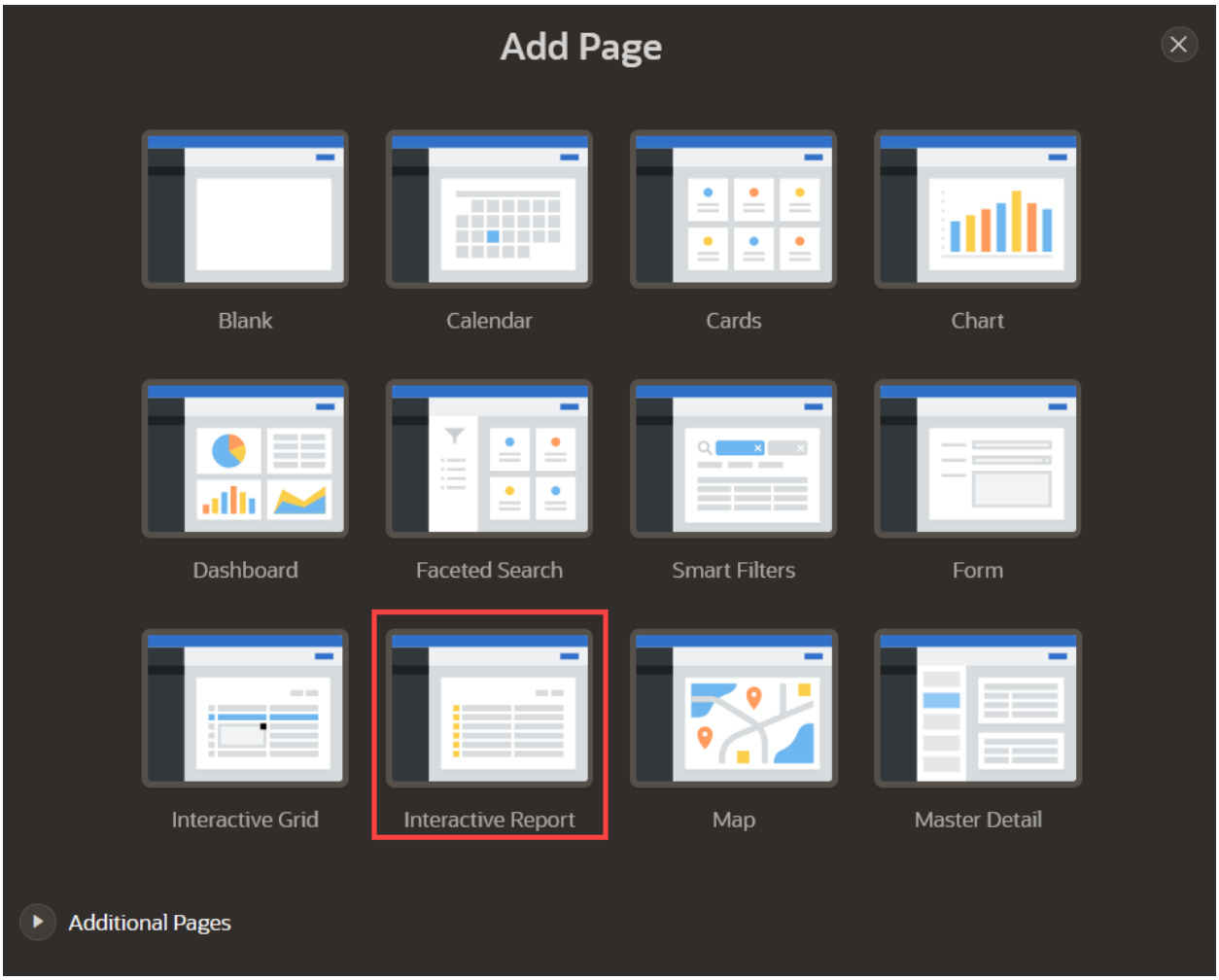
2. For each page you edit, click **Advanced** and check **Set as Administration Page**.

3. Click **Save Changes**.



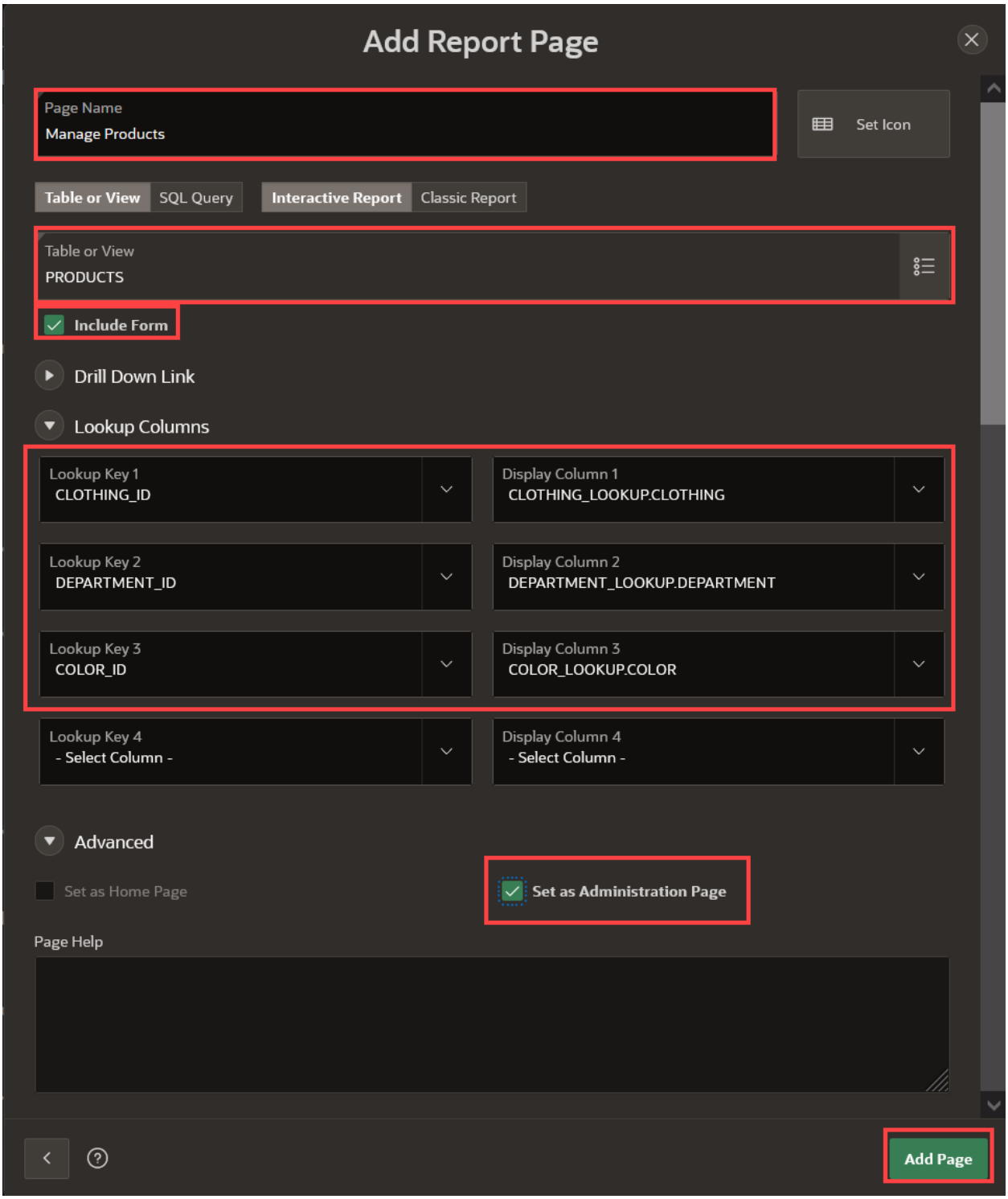
Task 8: Add Manage Products Page

1. In the Create Application wizard, click **Add Page**.
2. Click **Interactive Report**.



3. On the Report Page, enter the following:
    - Page Name - enter **Manage Products**
    - Table - select **PRODUCTS**
    - Check **Include Form**
4. For Lookup Columns, enter the following:
    - Lookup Key 1 - select **CLOTHING_ID**
    - Display Column 1 - select **CLOTHING_LOOKUP.CLOTHING**
    - Lookup Key 2 - select **DEPARTMENT_ID**
    - Display Column 2 - select **DEPARTMENT_LOOKUP.DEPARTMENT**
    - Lookup Key 3 - select **COLOR_ID**
    - Display Column 3 - select **COLOR_LOOKUP.COLOR**
5. Click **Advanced** and check **Set as Administration Page**.

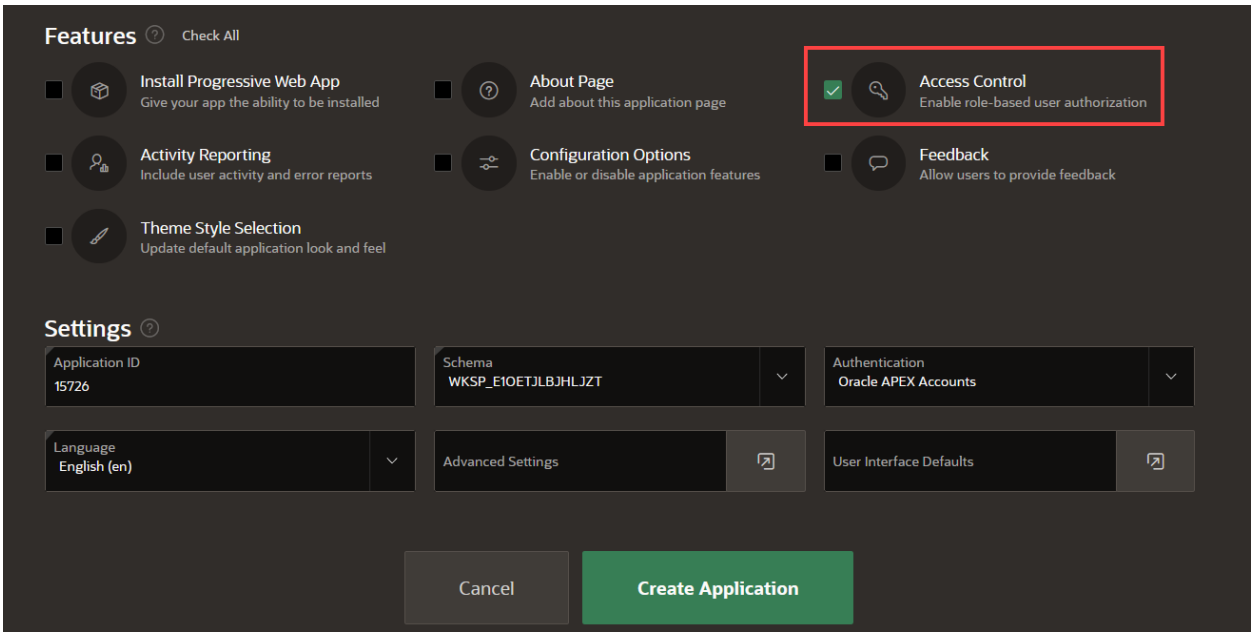
6. Click **Add Page**.



Task 9: Set Features

Features are a set of optional application capabilities Oracle APEX can include for your new application. Access Control enables role-based user authorization with a single click.

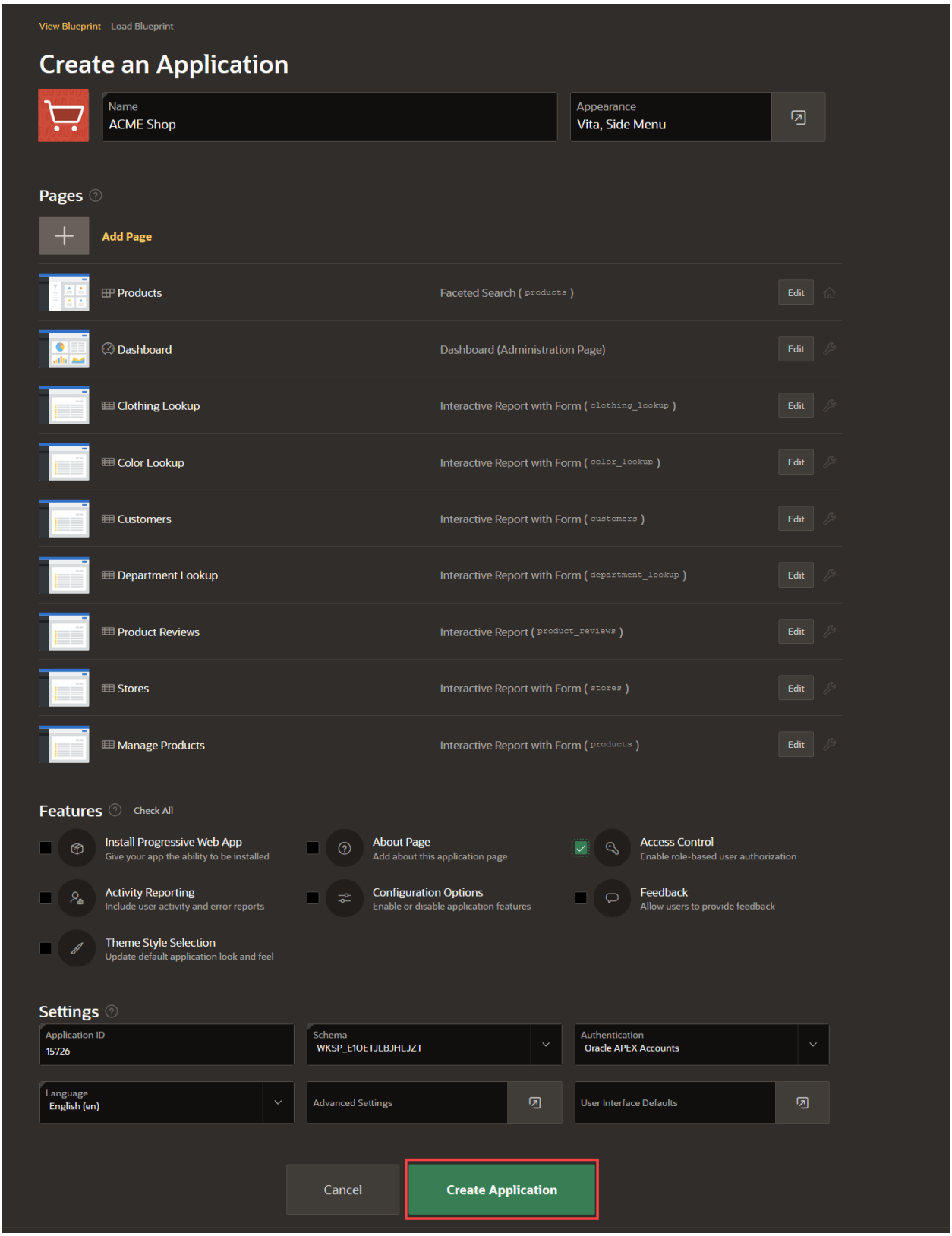
1. Under **Features** section, check **Access Control**.



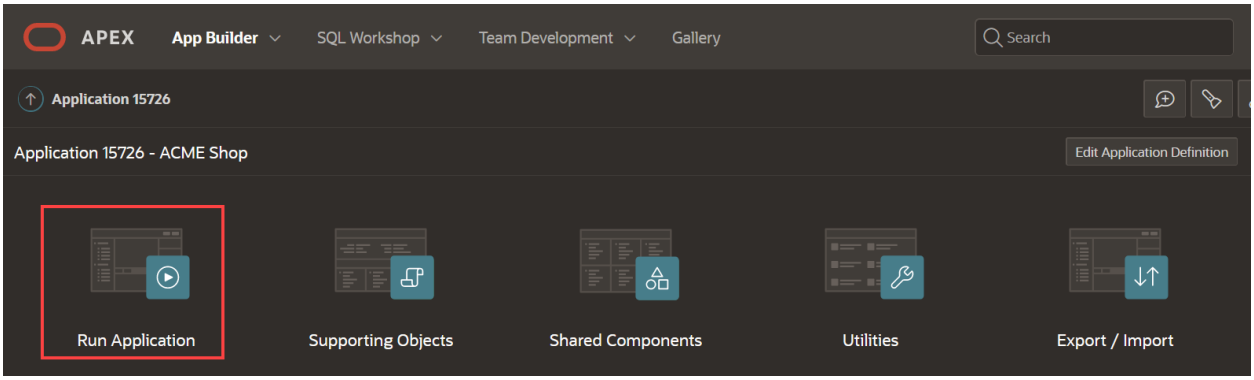
Task 10: Finish Creating the Application

Now that you have added all the pages, it is time to generate the app and review it.

1. Scroll to the bottom of the page, and click **Create Application**.

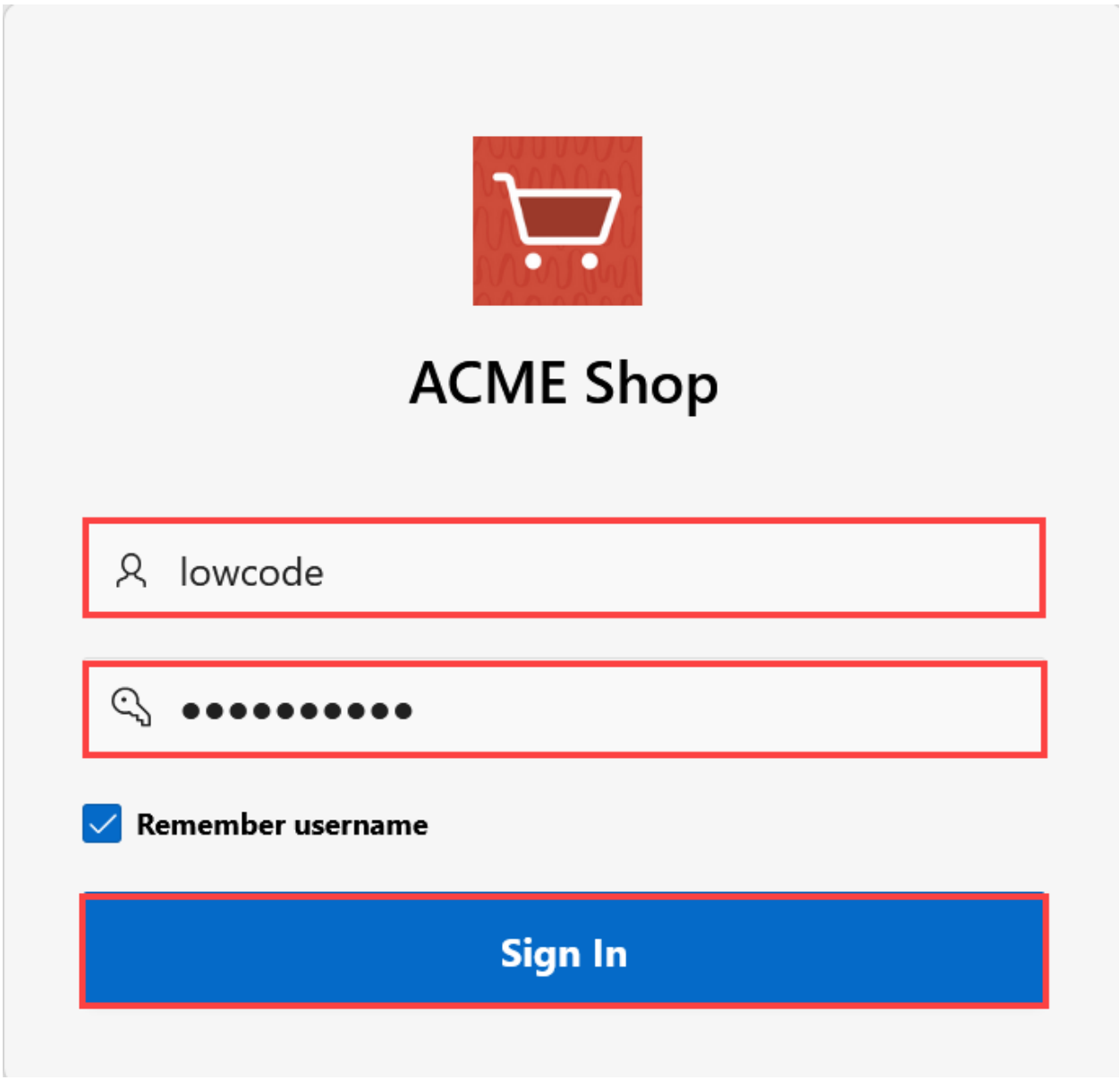

View Blueprint | Load Blueprint
Create an Application
Name
ACME Shop
Appearance
Vita, Side Menu
Pages
Add Page
Products
Faceted Search (products)
Edit
Dashboard
Dashboard (Administration Page)
Edit
Clothing Lookup
Interactive Report with Form (clothing_lookup)
Edit
Color Lookup
Interactive Report with Form (color_lookup)
Edit
Customers
Interactive Report with Form (customers)
Edit
Department Lookup
Interactive Report with Form (department_lookup)
Edit
Product Reviews
Interactive Report (product_reviews)
Edit
Stores
Interactive Report with Form (stores)
Edit
Manage Products
Interactive Report with Form (products)
Edit
Features
Check All
Install Progressive Web App
Give your app the ability to be installed
About Page
Add about this application page
Access Control
Enable role-based user authorization
Activity Reporting
Include user activity and error reports
Configuration Options
Enable or disable application features
Feedback
Allow users to provide feedback
Theme Style Selection
Update default application look and feel
Settings
Application ID
15726
Schema
WKSP_E1OETJLBJHLJZT
Authentication
Oracle APEX Accounts
Language
English (en)
Advanced Settings
User Interface Defaults
Cancel
Create Application

2. Once the application is created, you will find the new app on the application home page. Click **Run Application**.

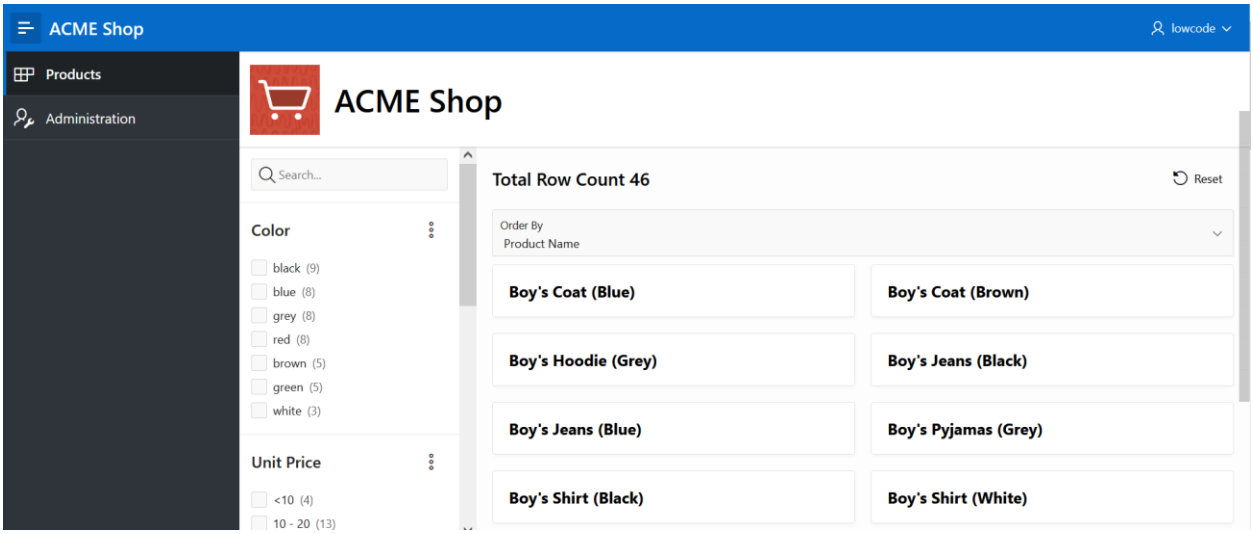


Task 11: Run the Application

1. Enter your user credentials. Click **Sign In**.



2. The new application will be displayed. Explore the pages that you just created by clicking on the navigation menu.



You now know how to create an application with a number of different page types based on existing database objects. You may now **proceed to the next lab**.

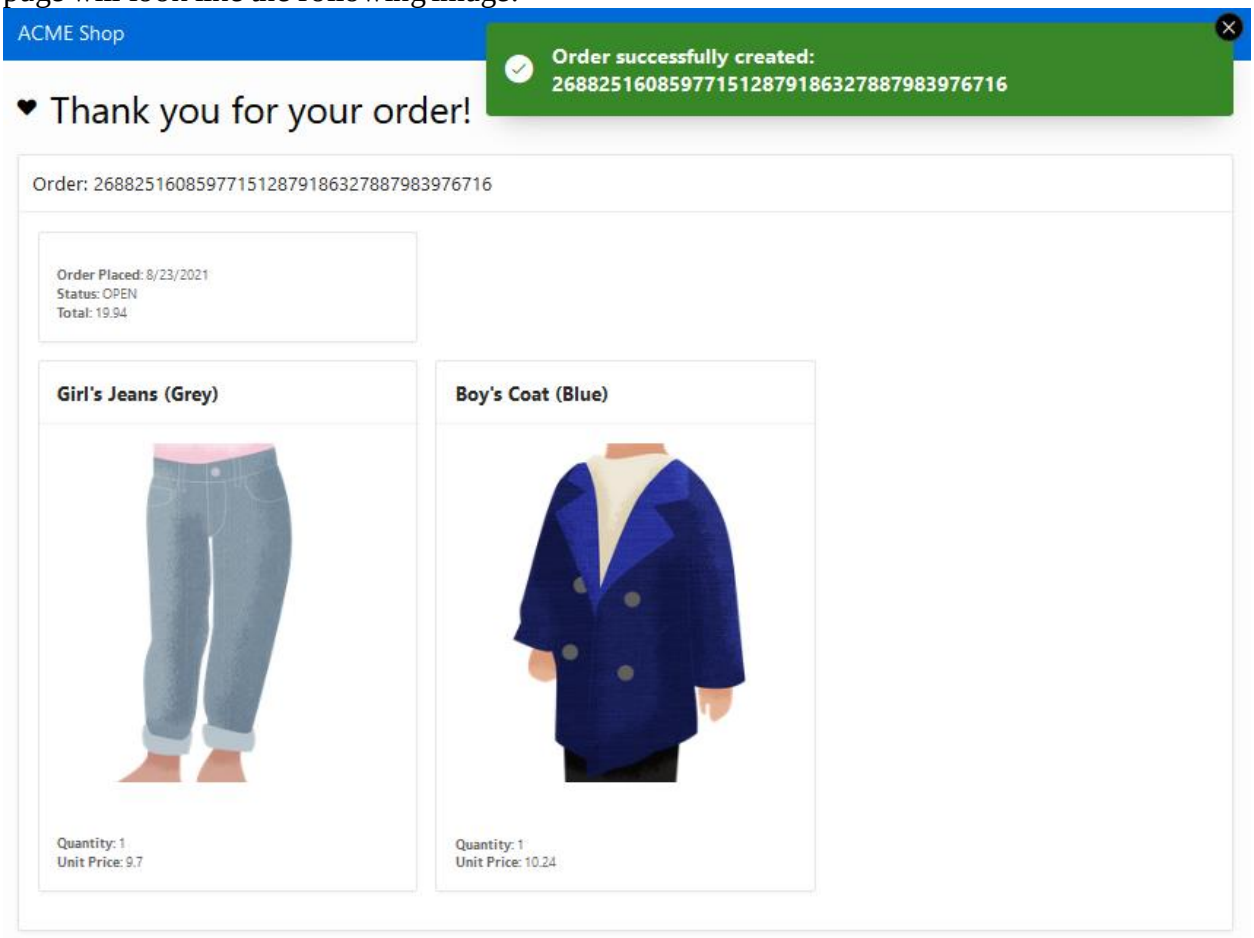
Create the order page

Introduction

In this lab, you will create a new page that will allow customers to view the details of their recent order. Customers will find the following details of the order:

- Order number
- Order date
- Status
- Total price
- Quantity and price of the items.

Once you have finished the workshop and updated all the products as described in the steps, your page will look like the following image:



Estimated Time: 15 minutes

Objectives

In this lab, you will:

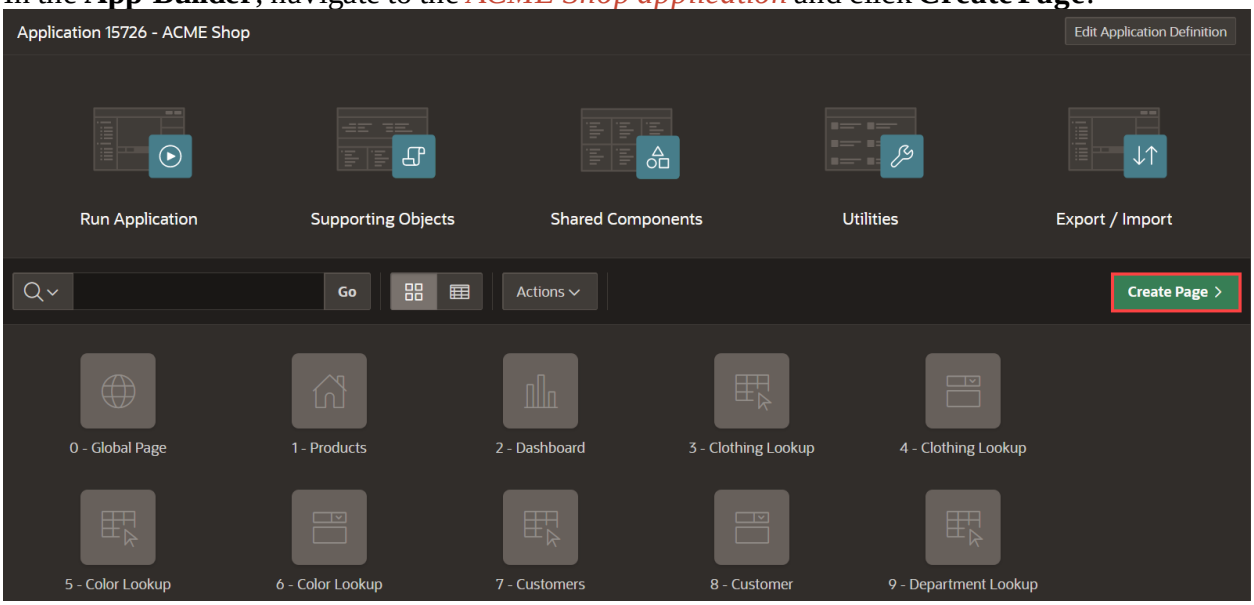
- Create a page to review the items that the customer recently bought.

Collapse All Tasks

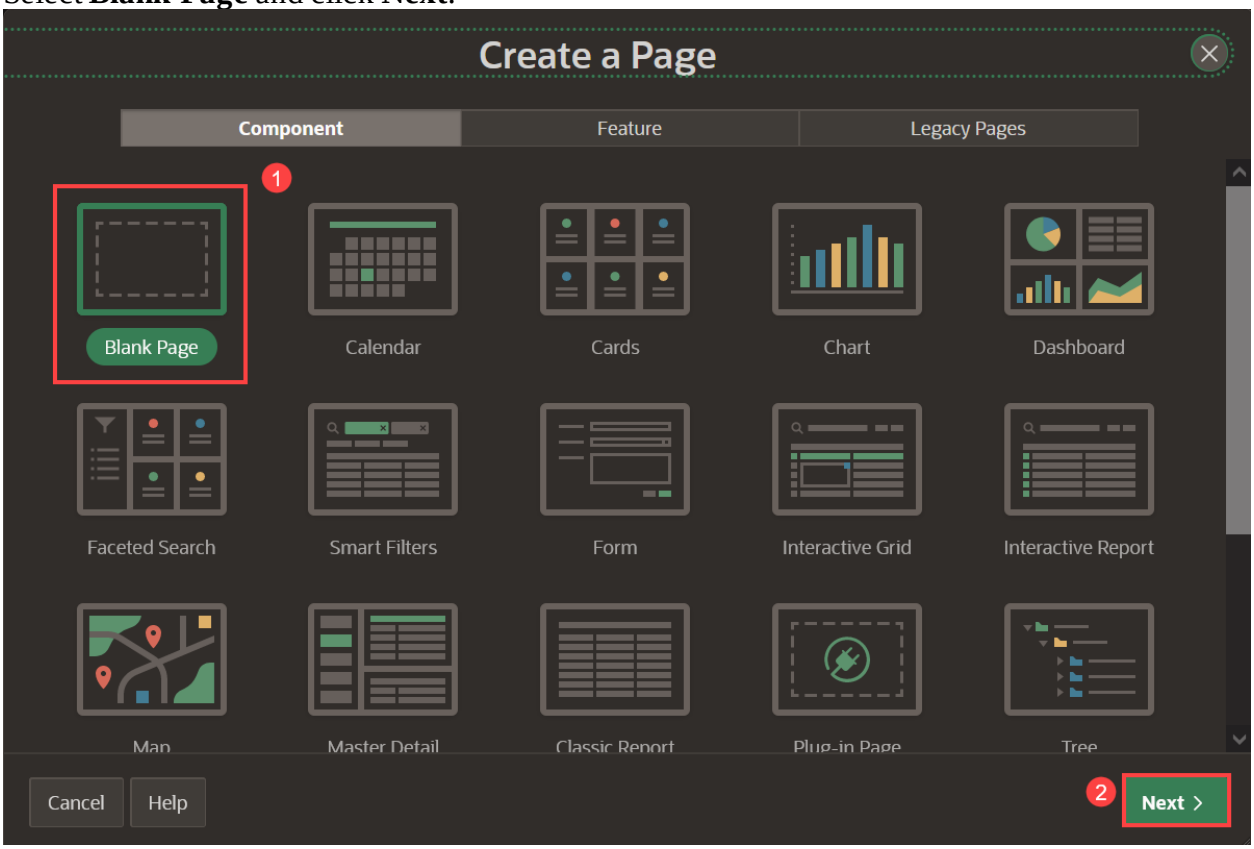
Task 1: Create a Normal Page - Order Information

Create a Normal Page to review the Order that customer has made.

1. In the **App Builder**, navigate to the *ACME Shop application* and click **Create Page**.



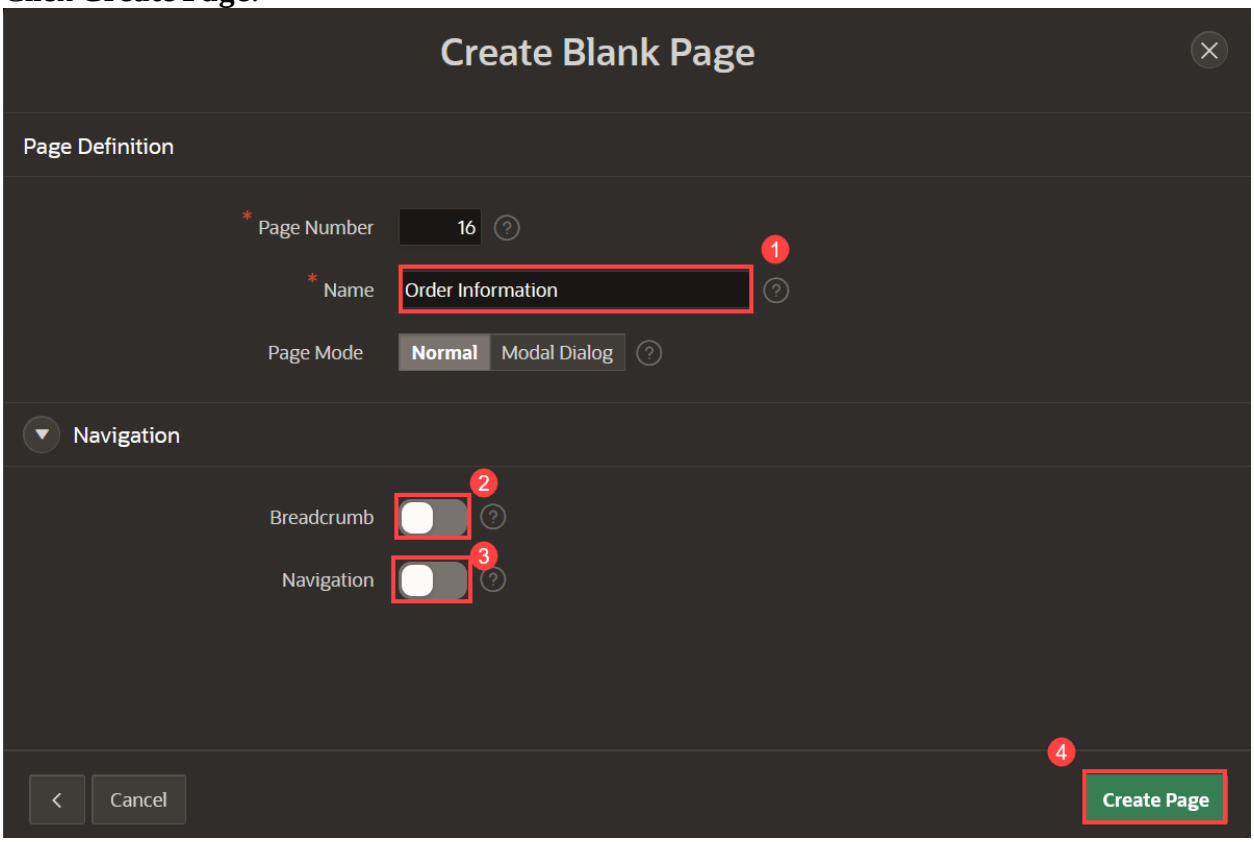
2. Select **Blank Page** and click **Next**.



3. Enter the following and click **Next**.
    - Page Number - enter **16**
    - Name - enter **Order Information**

- Under Navigation, deselect **Breadcrumb** and **Navigation**.

4. Click **Create Page**.

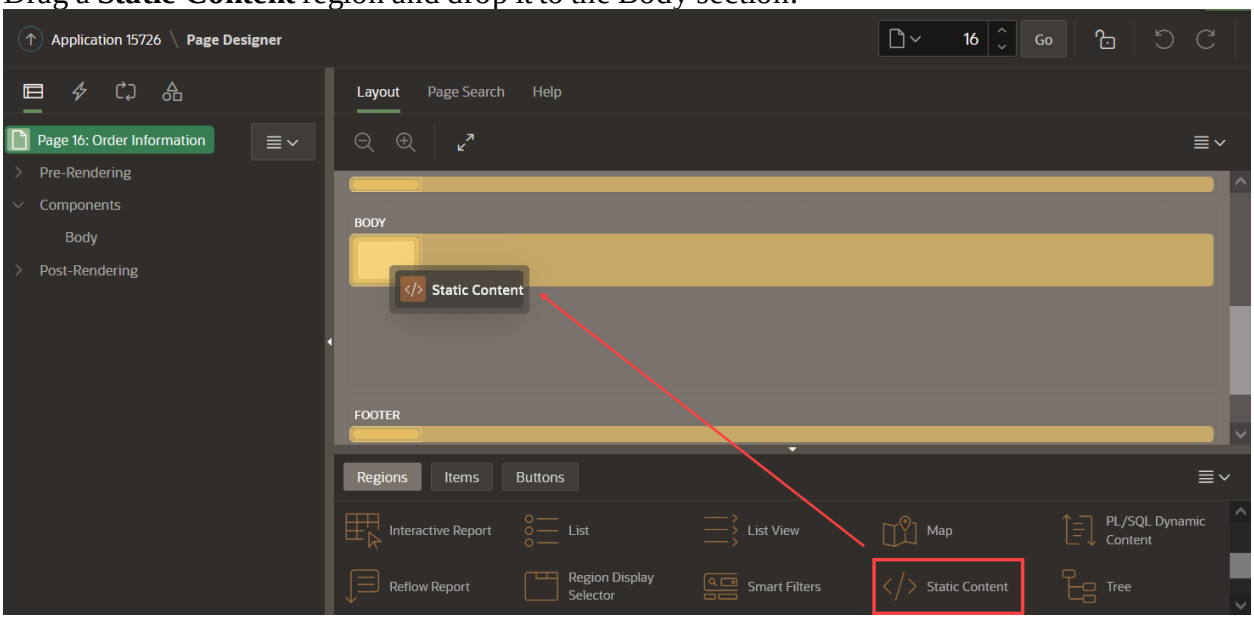


Task 2: Add a Region

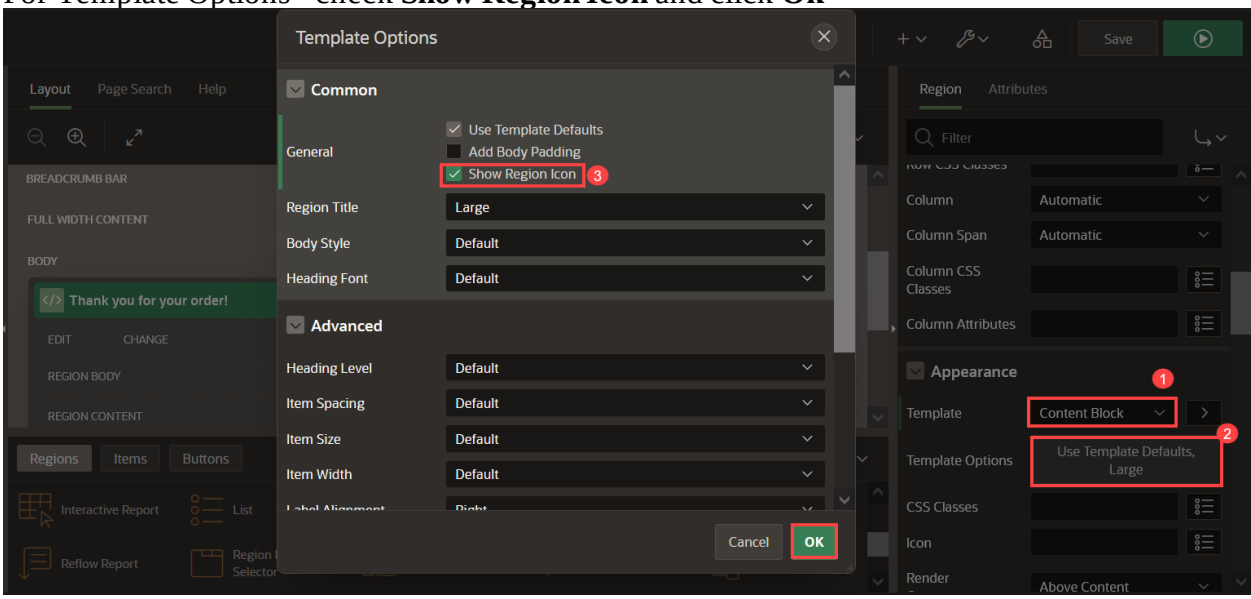
Add a region to the page to display order details.

1. In the newly created page, navigate to the **Gallery Menu** at the bottom of the page showing Regions, Items, and Buttons categories and ensure that **Regions** is selected.

2. Drag a **Static Content** region and drop it to the Body section.



3. In the Property Editor, enter the following:
   - For Title - enter **Thank you for your order!**
   - For Template - select **Content Block**
   - For Template Options - check **Show Region Icon** and click **Ok**



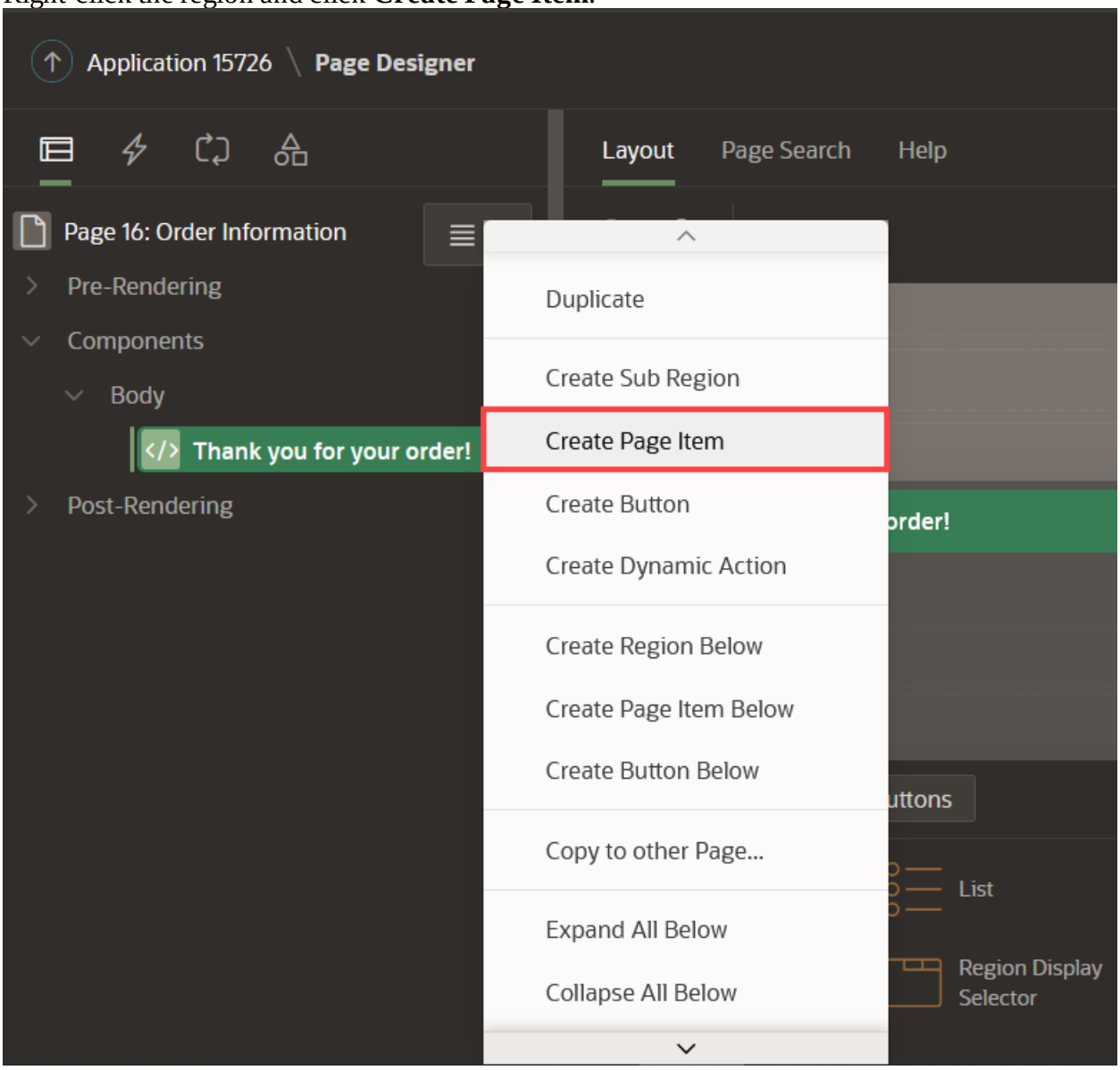
   - For Icon, enter **fa-heart**

Task 3: Add Items to the Page

Add a hidden item to store the order ID without the user being able to see it.

1. In the Rendering tree (left pane), navigate to the **Thank you for your order!** region.

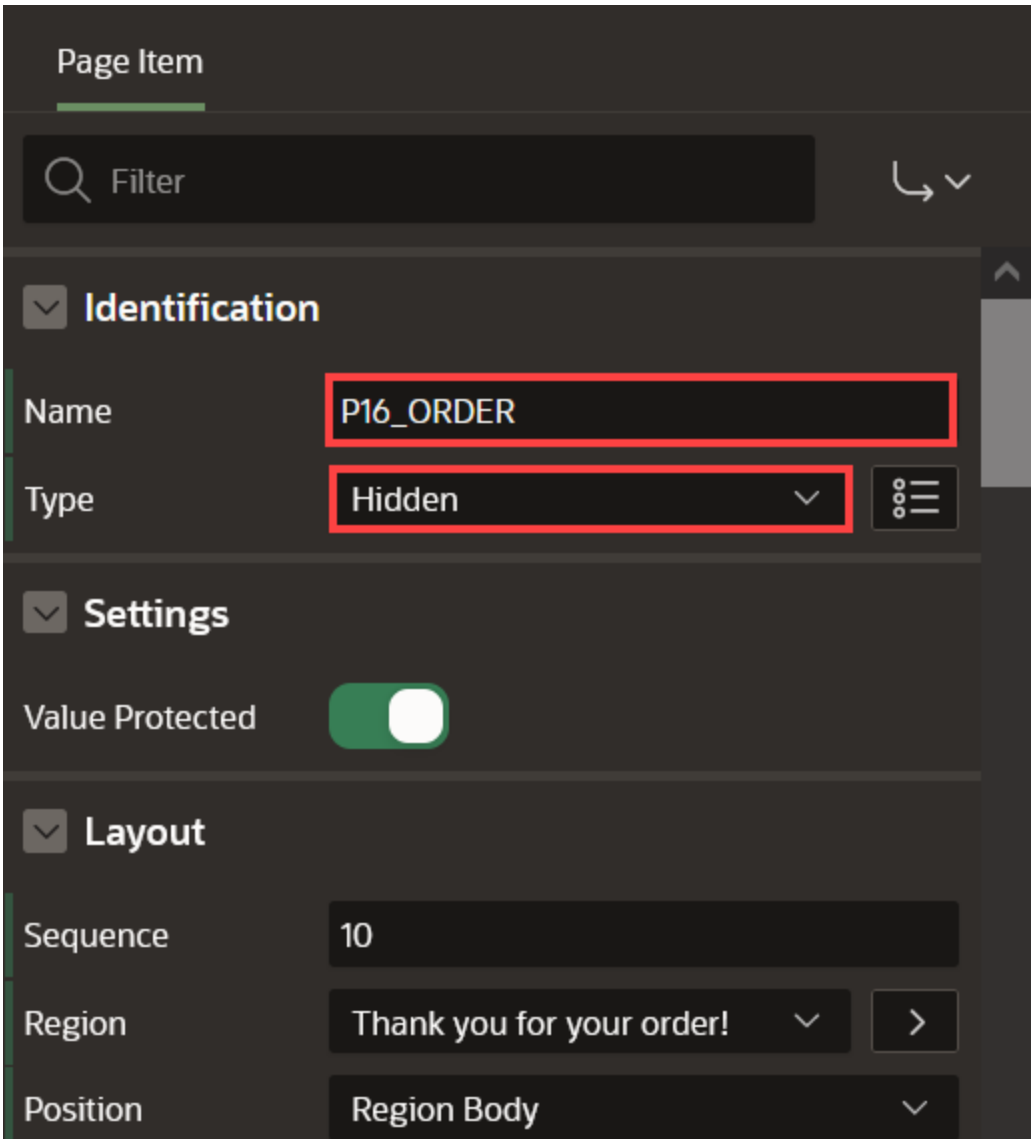
2. Right-click the region and click **Create Page Item**.



3. In the property editor, set the name and type as follows:

Table 1: Page Item values

Name	Type
P16_ORDER	Hidden

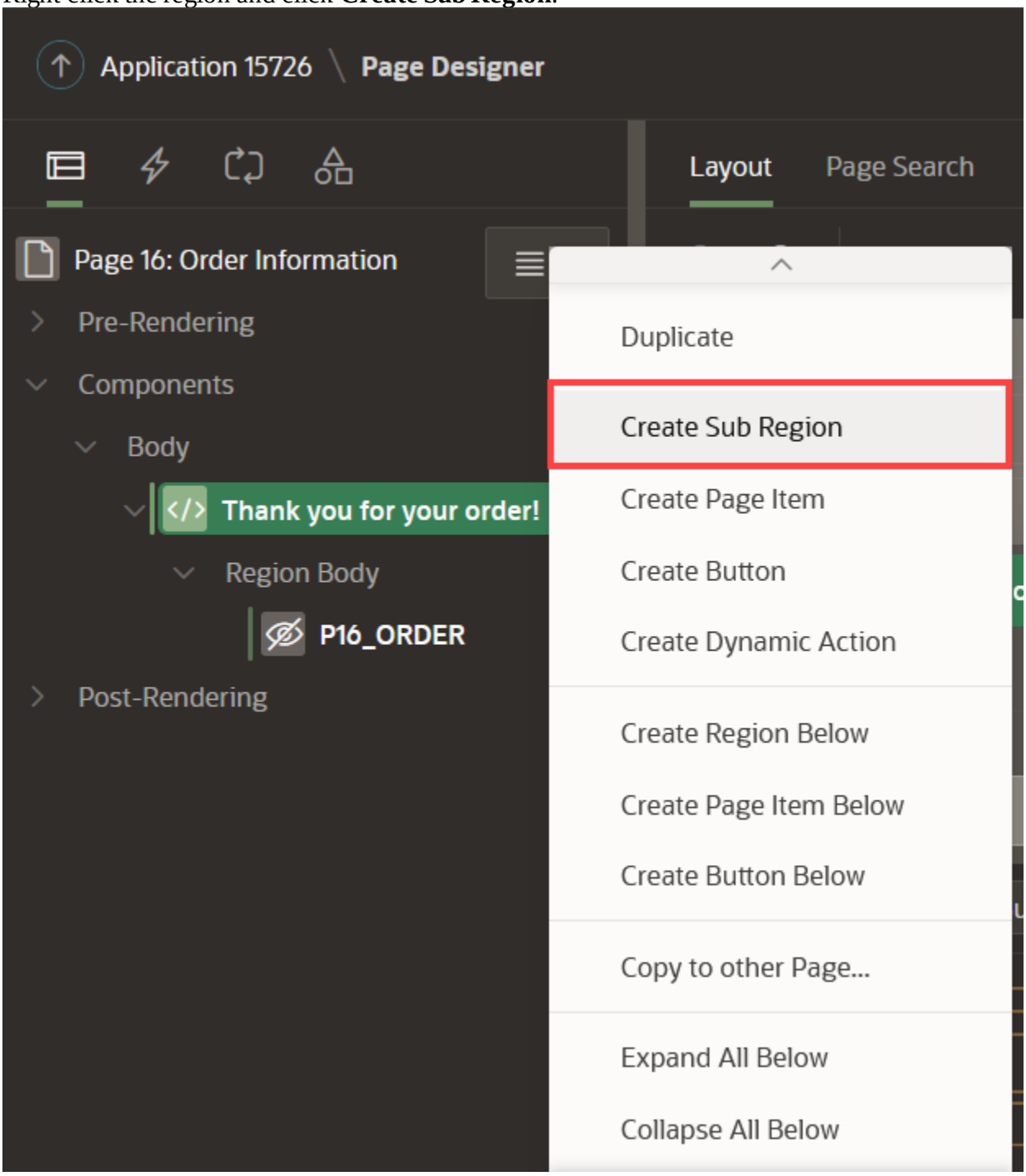
4. 

Task 4: Add Static Content Region

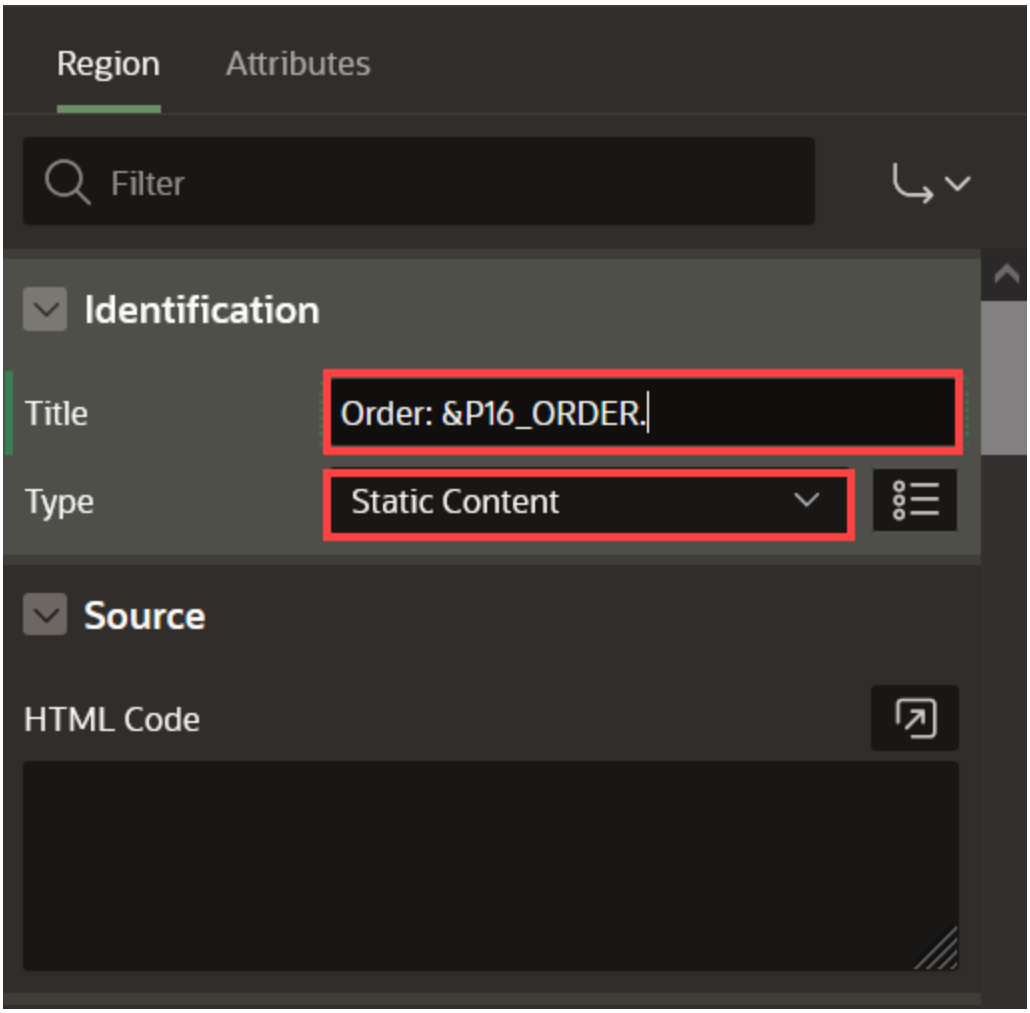
Add a region to contain Order details and items.

1. In the Rendering tree (left pane), navigate to the **Thank you for your order!** region.

2. Right click the region and click **Create Sub Region**.



3. In the Property Editor, enter the following:
    - For Title - enter the expression (including the period) **Order: &P16_ORDER.**
    - For Type - select **Static Content**

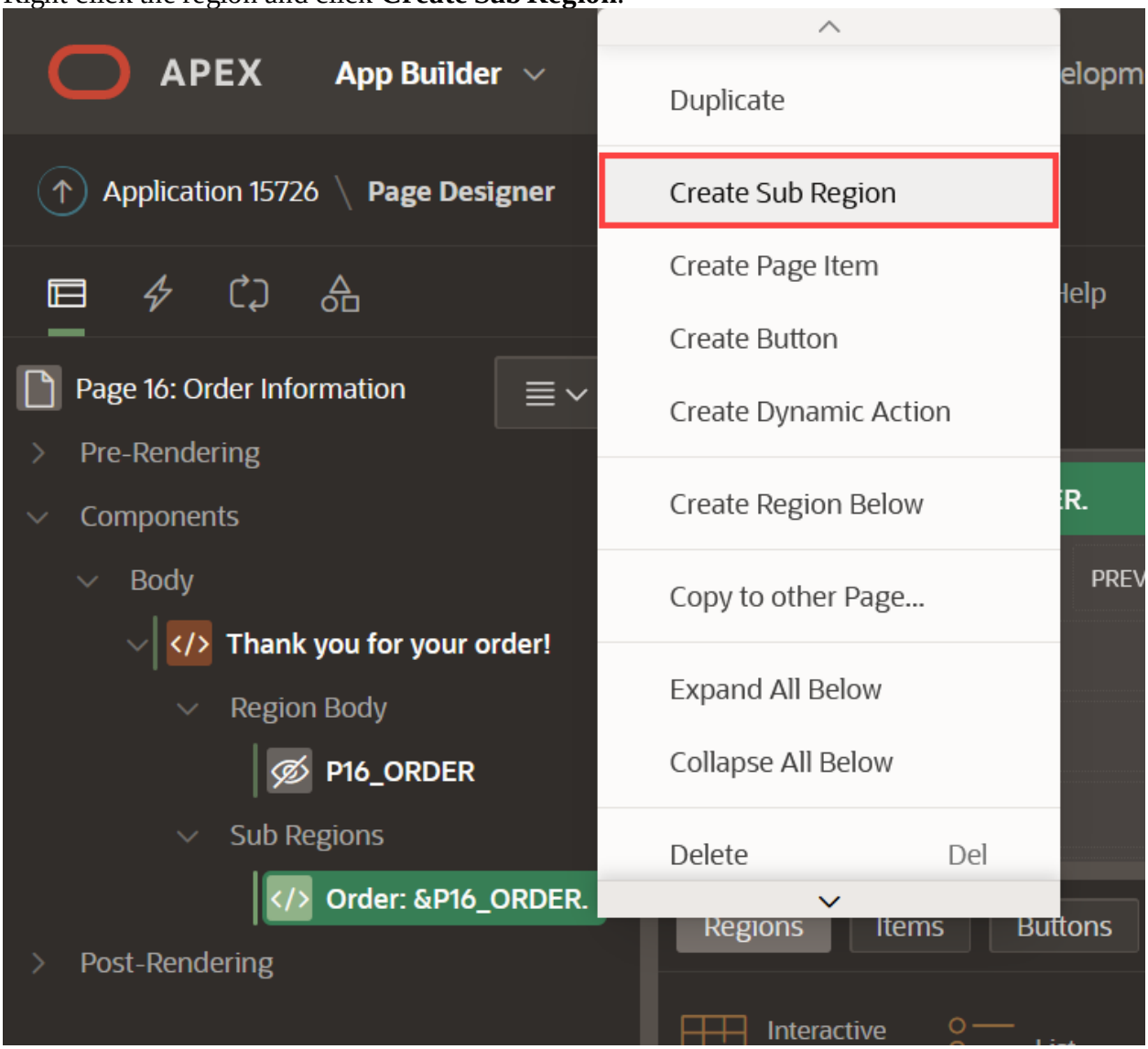


Task 5: Add Order Details Region

Add a region to display Order details.

1. In the Rendering tree (left pane), navigate to the **Order: &P16_ORDER.** region.

2. Right click the region and click **Create Sub Region**.

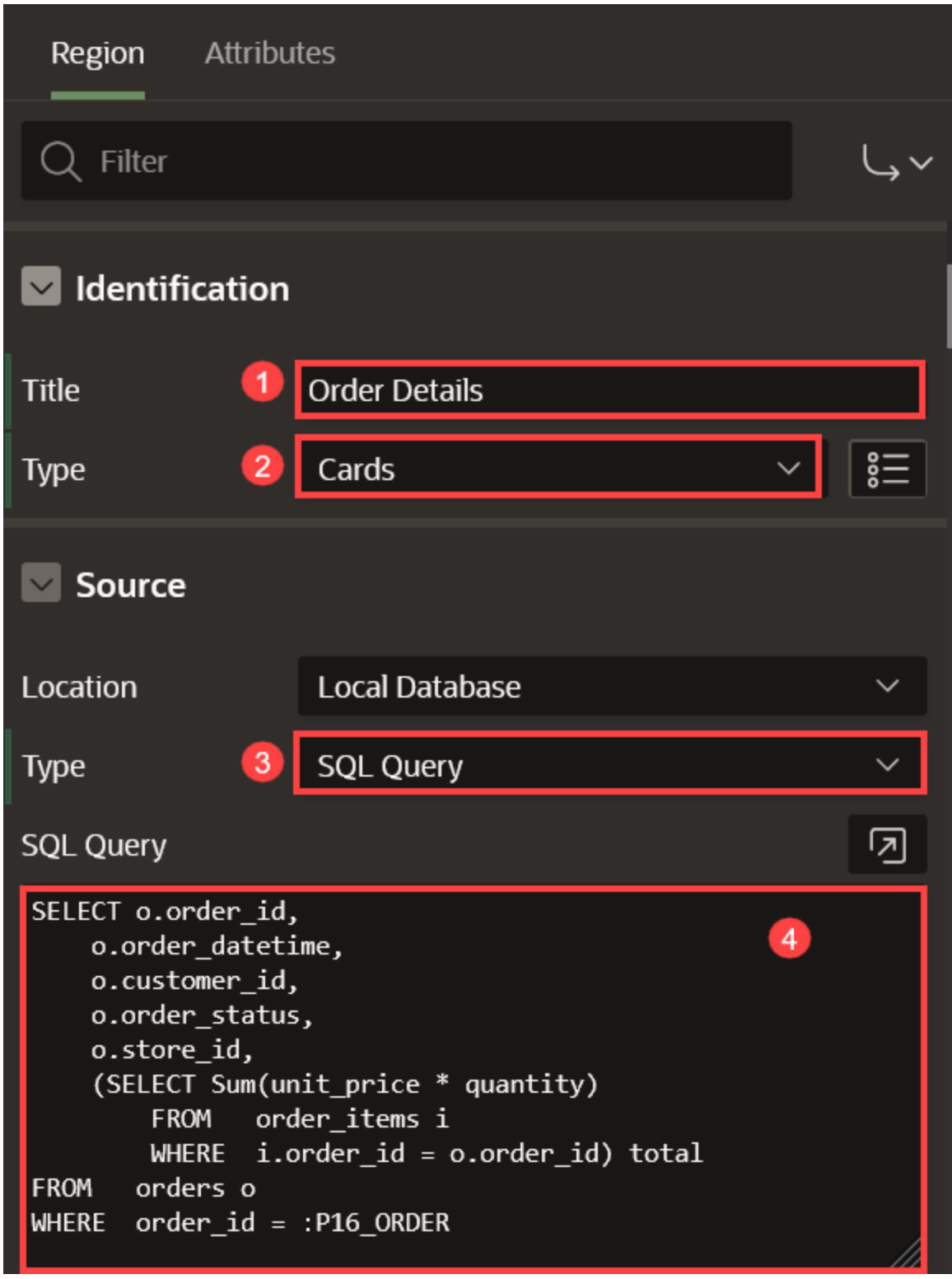


3. In the Property Editor, enter the following:
    - For Title - enter **Order Details**
    - For Type - select **Cards**
    - Under Source section:
        - For Type - select **SQL Query**
        - For SQL Query - enter the following SQL Query:

```
Copy
SELECT o.order_id,
```

```
        o.order_datetime,
        o.customer_id,
        o.order_status,
        o.store_id,
        (SELECT Sum(unit_price * quantity)
            FROM   order_items i
            WHERE  i.order_id = o.order_id) total
    FROM   orders o
```

```
WHERE  order_id = :P16_ORDER
```

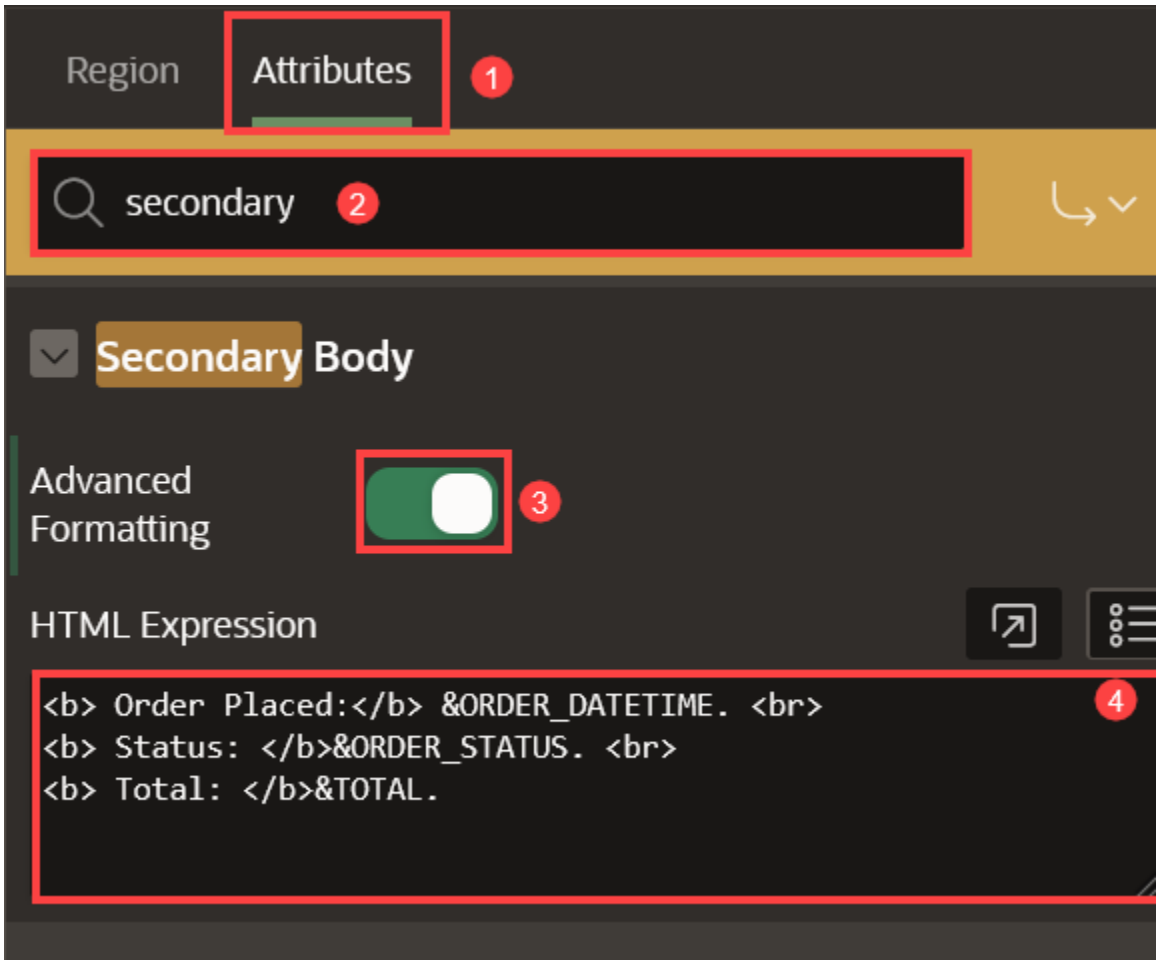


4. Click **Attributes**.
    - Search for Secondary Body in the filter and do the following:

- Set Advanced Formatting to **On**
- For HTML Expression - enter:

```
Copy<b> Order Placed:</b> &ORDER_DATETIME. <br>
<b> Status: </b>&ORDER_STATUS. <br>
```

```
<b> Total: </b>&TOTAL.
```

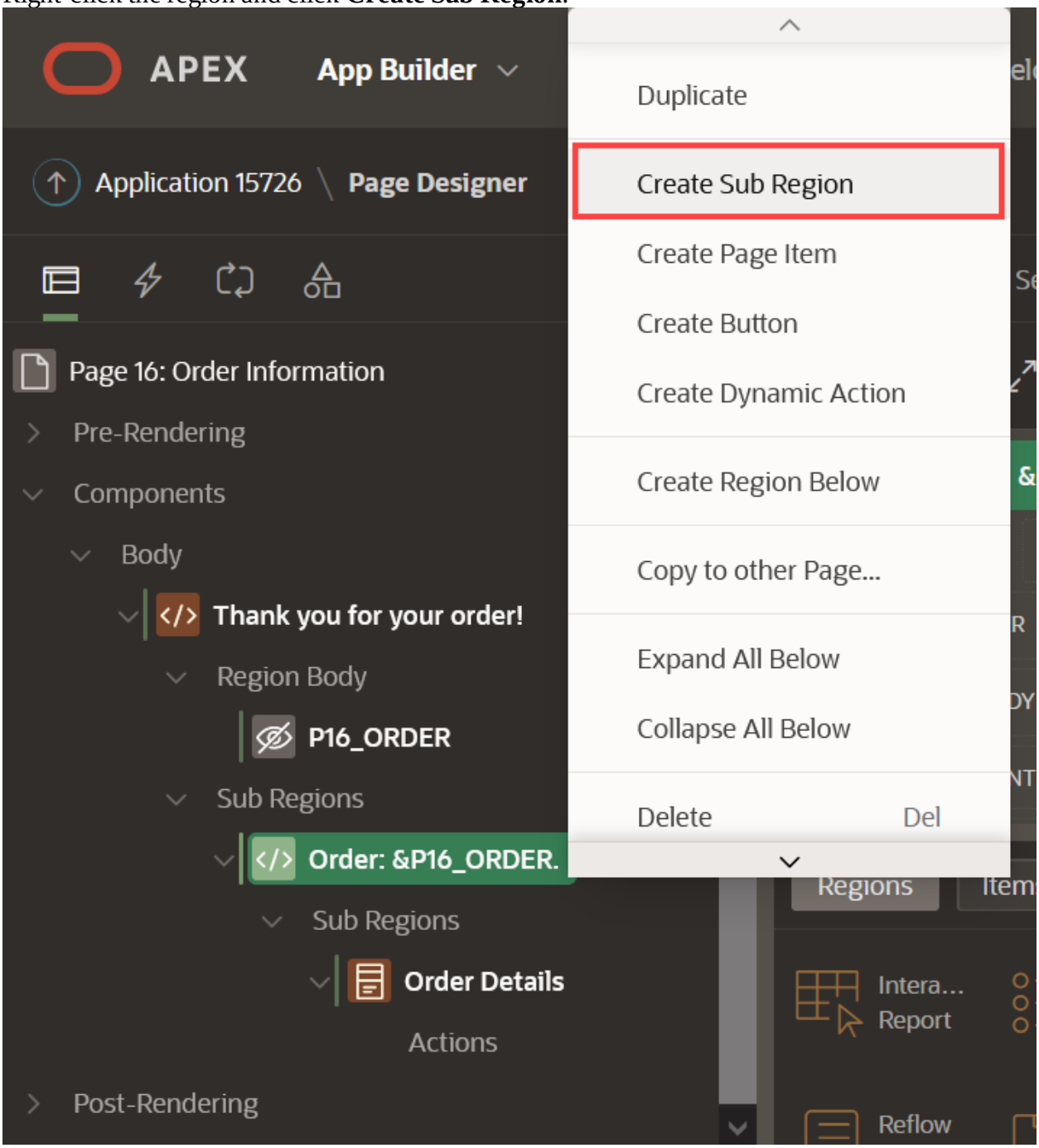


Task 6: Add Items Region

Add a region to display items in the order.

1. In the Rendering tree (left pane), navigate to the **Order: &P16_ORDER.** region.

2. Right-click the region and click **Create Sub Region**.

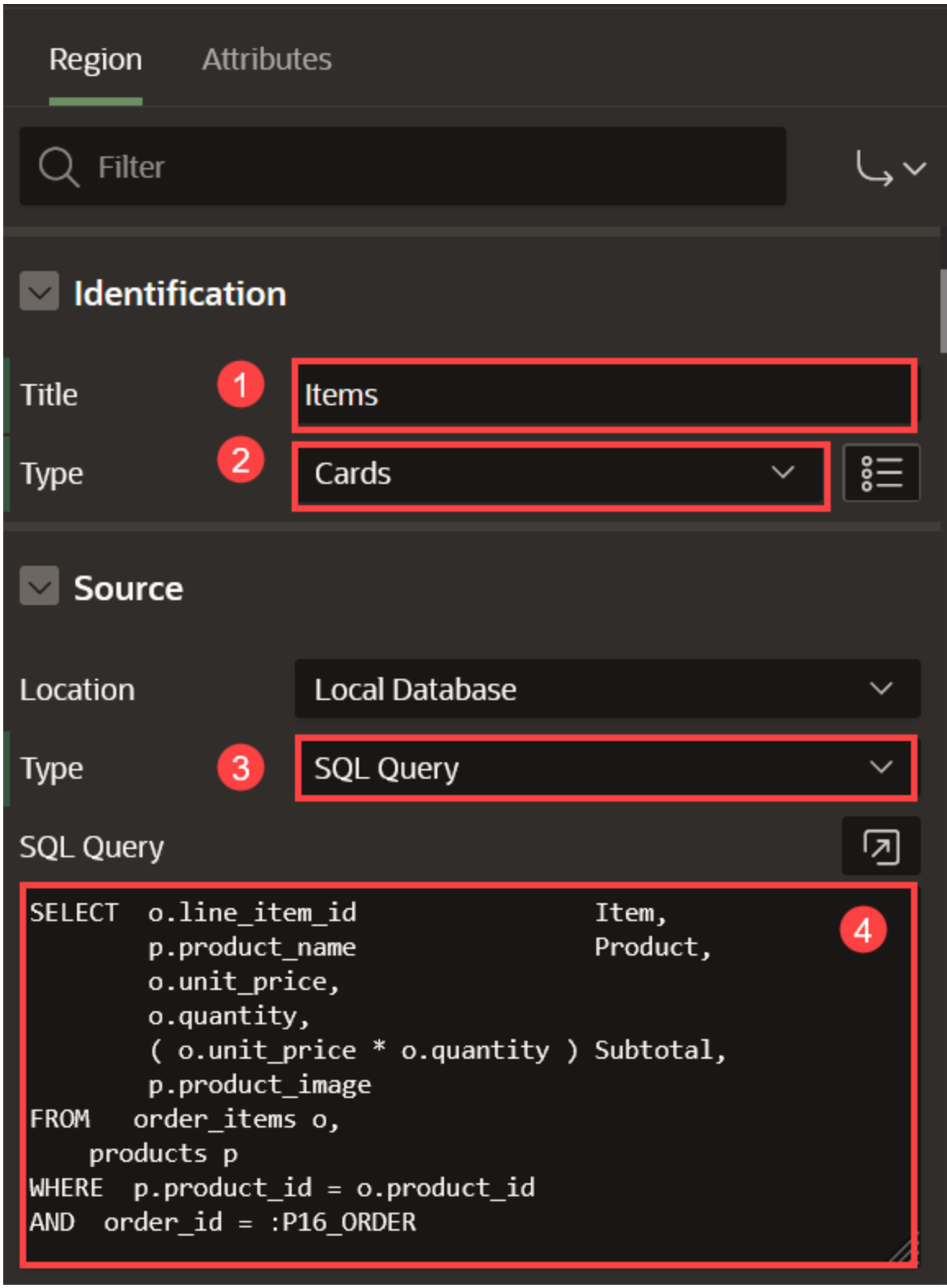


3. In the Property Editor, enter the following:
    - For Title - enter **Items**
    - For Type - select **Cards**
    - Under Source section:

- For Type - select **SQL Query**
- For SQL Query - enter the following SQL Query:

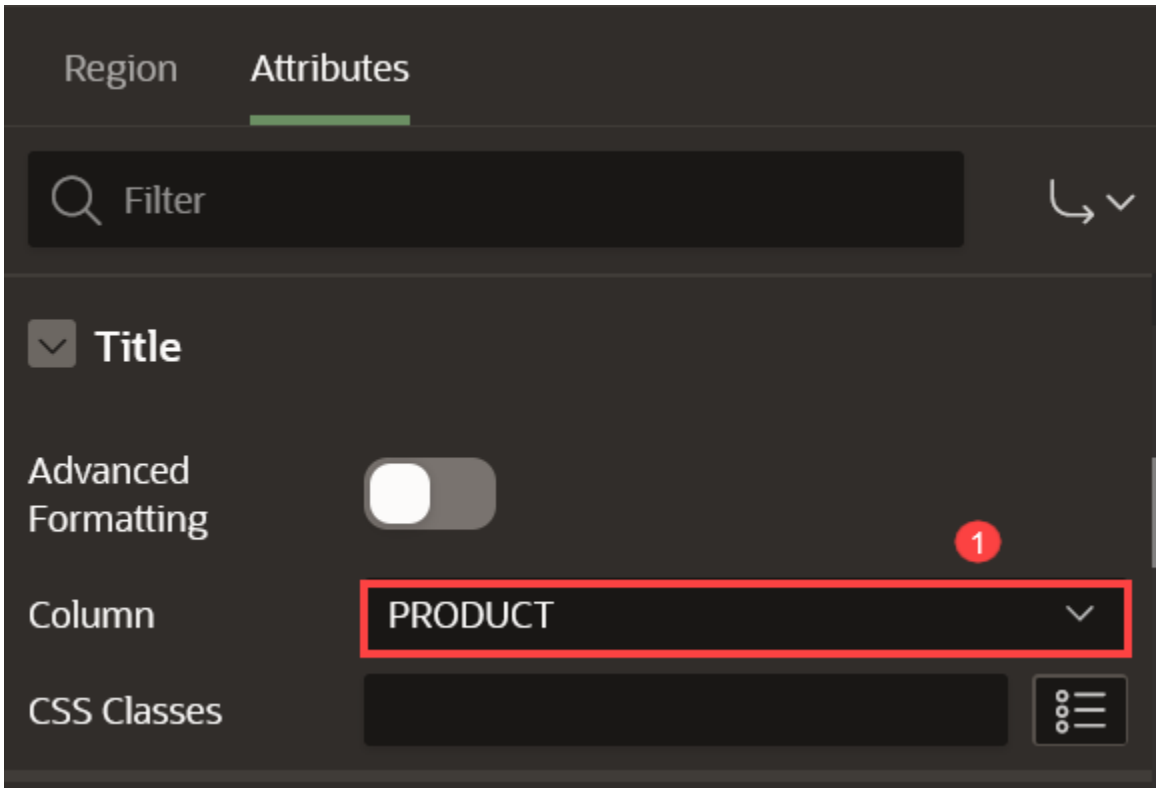
```
CopySELECT  o.line_item_id                  Item,
        p.product_name                  Product,
        o.unit_price,
        o.quantity,
        ( o.unit_price * o.quantity ) Subtotal,
        p.product_image
FROM   order_items o,
    products p
WHERE  p.product_id = o.product_id
```

```
AND  order_id = :P16_ORDER
```



4. Click **Attributes** and do the following:
    - Under Title section:

- For Column - select **PRODUCT**

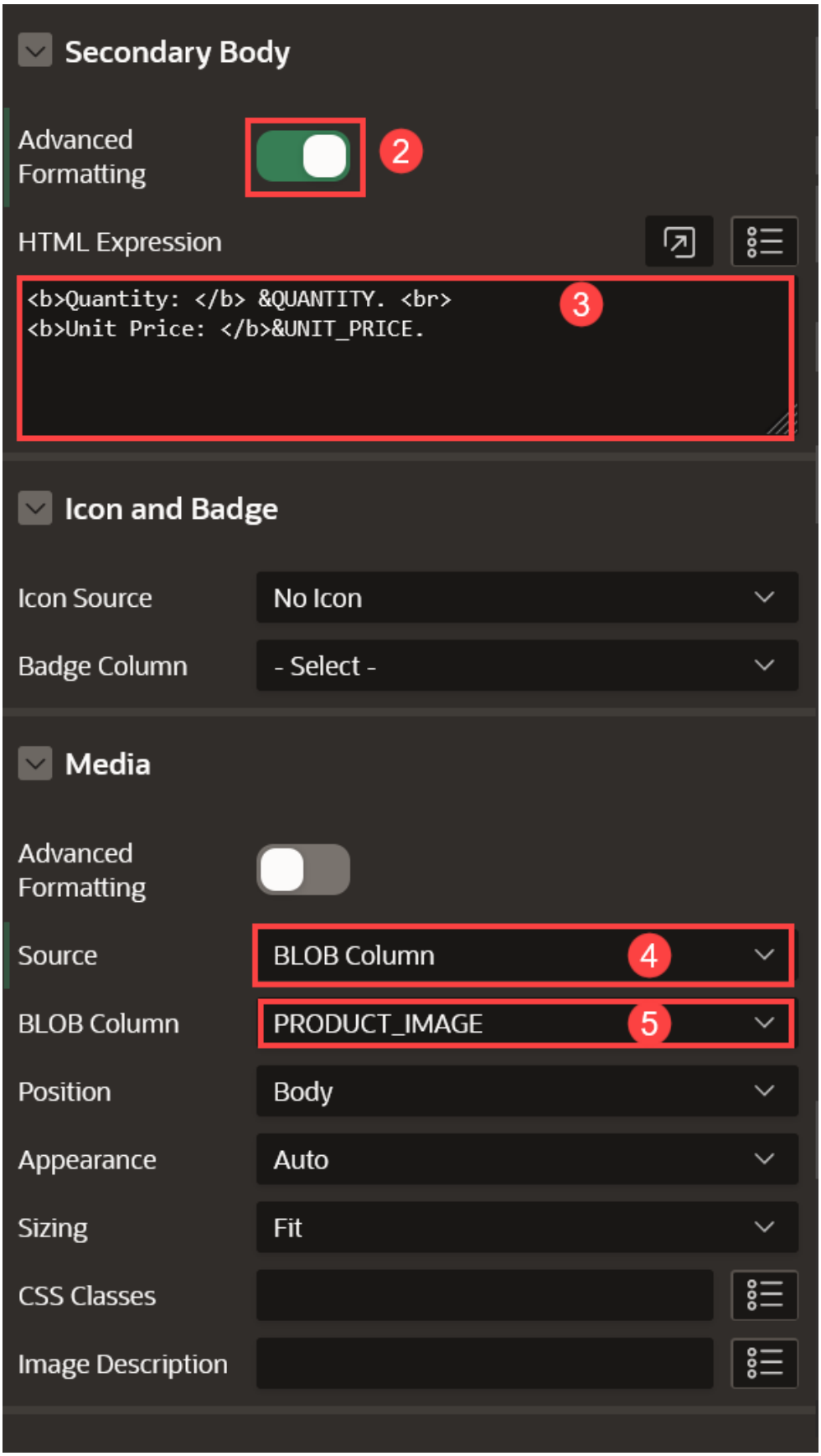


- Under Secondary Body:
  - Set Advanced Formatting to **On**
  - For HTML Expression - enter:

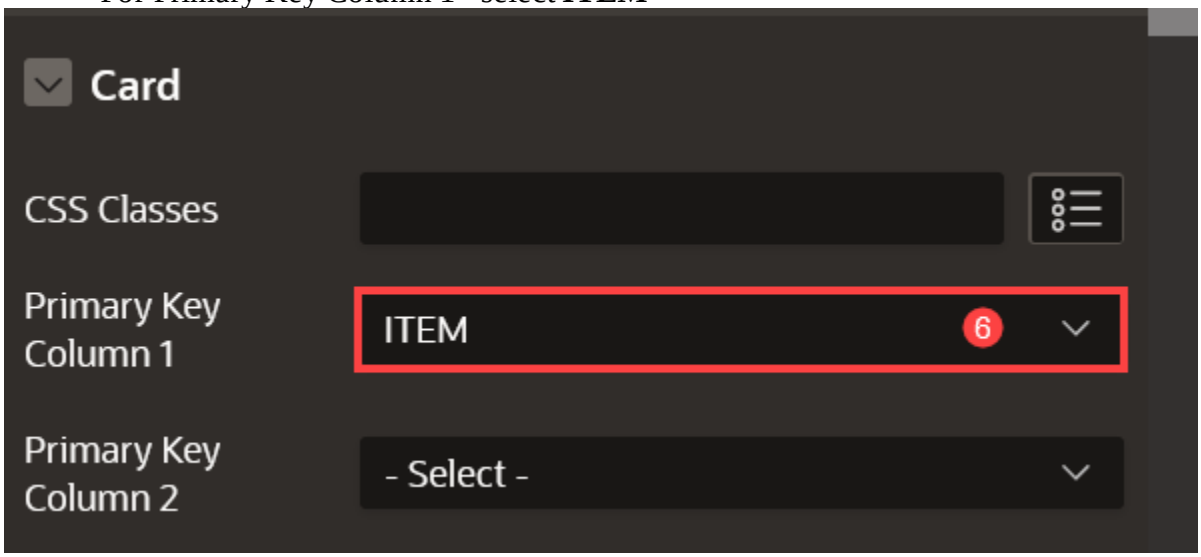
```
Copy<b>Quantity: </b> &QUANTITY. <br>
```

```
<b>Unit Price: </b>&UNIT_PRICE.
```

- Under Media section:
  - For Source - select **BLOB Column**
  - For BLOB Column - select **PRODUCT_IMAGE**
  - For Position - select **Body**
  - For Appearance - select **Auto**
  - For Sizing - select **Fit**


Secondary Body
Advanced Formatting
2
HTML Expression
Quantity: &QUANTITY.
Unit Price: &UNIT_PRICE.
3
Icon and Badge
Icon Source
No Icon
Badge Column
- Select -
Media
Advanced Formatting
Source
BLOB Column
4
BLOB Column
PRODUCT_IMAGE
5
Position
Body
Appearance
Auto
Sizing
Fit
CSS Classes
Image Description

- Under Card:
  - For Primary Key Column 1 - select **ITEM**



5. Click **Save**.

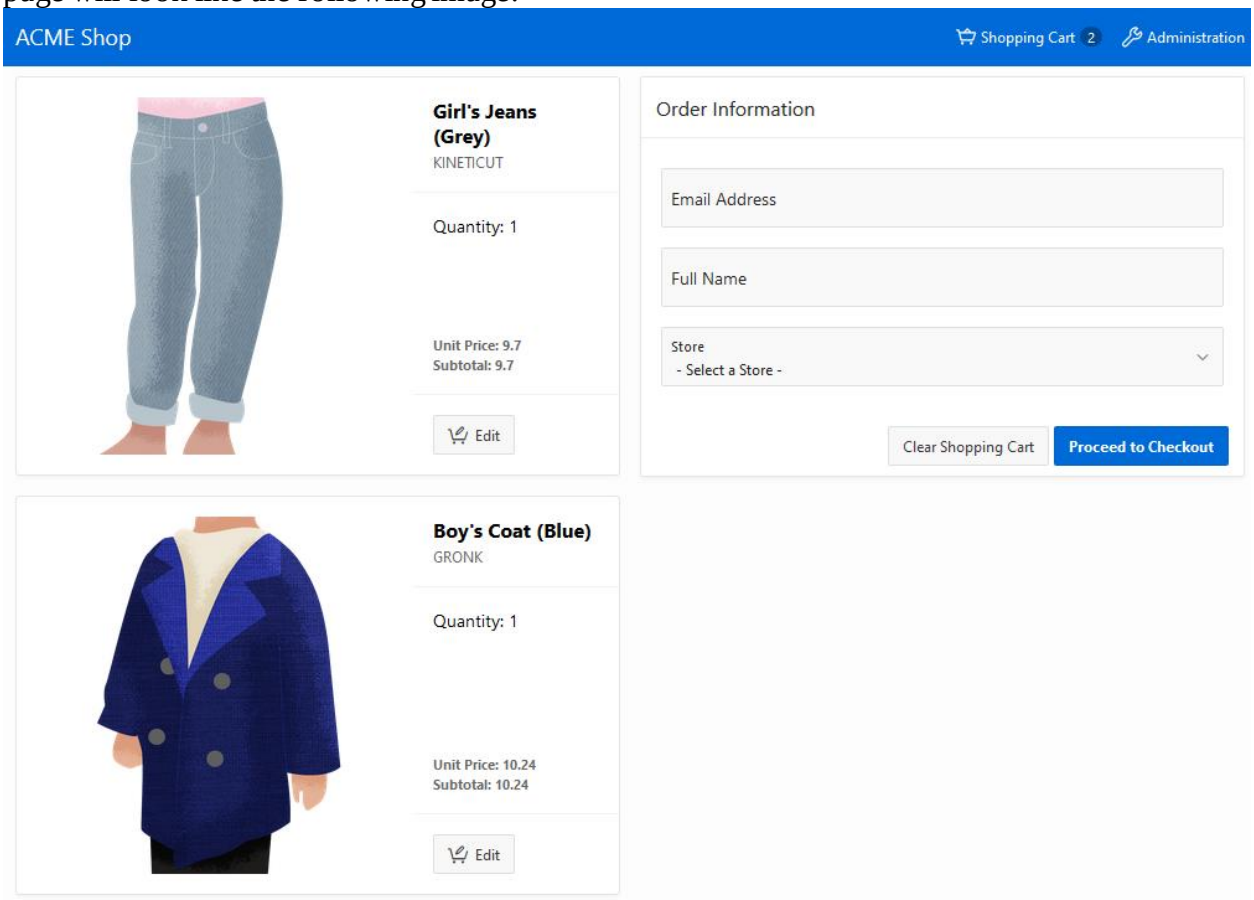
You now know how to add a new page to your existing APEX Application and add regions to define the page's content using the Page Designer. You may now **proceed to the next lab**.

Create the shopping cart page

Introduction

In this lab, you will create a new page to review the items added to the Shopping Cart.

Once you have finished the workshop and updated all the products as described in the steps, your page will look like the following image:



Customers will be able to:

- Review the items in the shopping cart
- Edit the quantity of the items
- Remove an item
- Clear the shopping cart
- Proceed to checkout

Estimated Time: 20 minutes

Objectives

In this lab, you will:

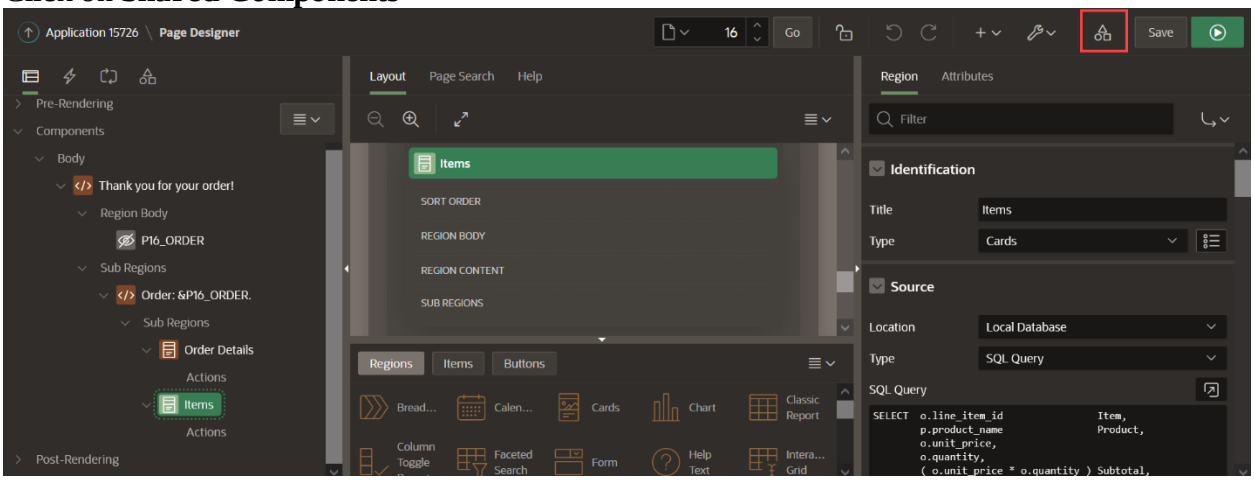
- Create a page to list the products in the Shopping Cart
- Create Application Items, Application Processes, and Dynamic Actions to manage the Shopping Cart

Collapse All Tasks

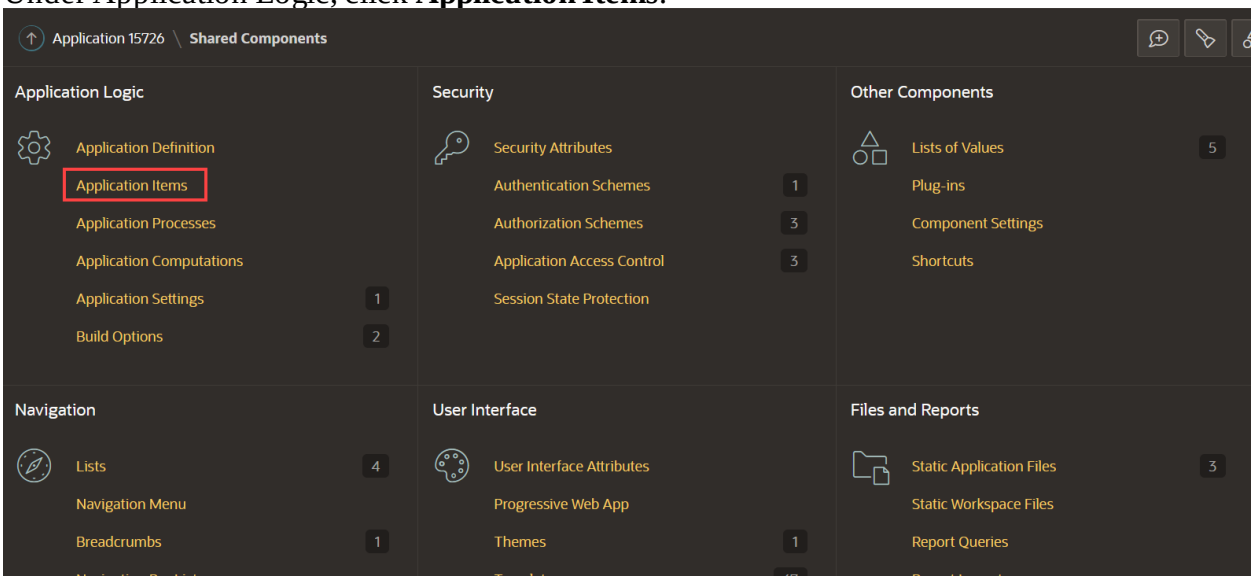
Task 1: Create Application Items

These items are needed to count the number of items in the shopping cart and the icon to display in the Navigation Bar.

1. Click on **Shared Components**



2. Under Application Logic, click **Application Items**.



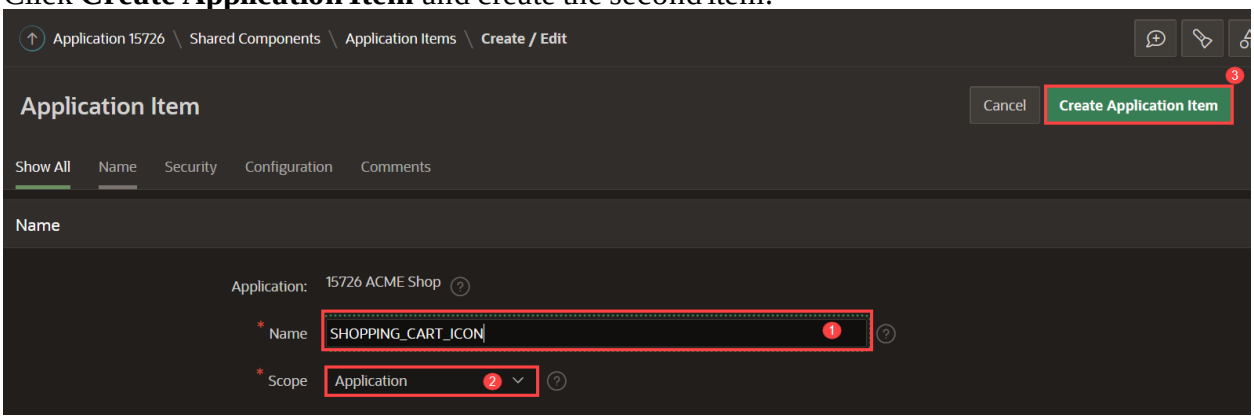
3. Click **Create**.

4. Create two items as follow:

Table 1: Application items

Name	Scope
SHOPPING_CART_ICON	Application
SHOPPING_CART_ITEMS	Application

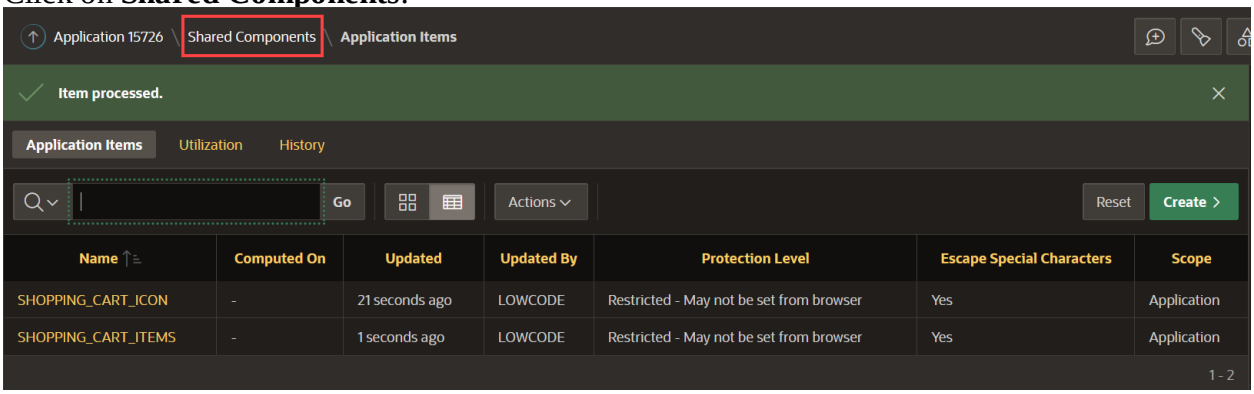
5. Click **Create Application Item** and create the second item.



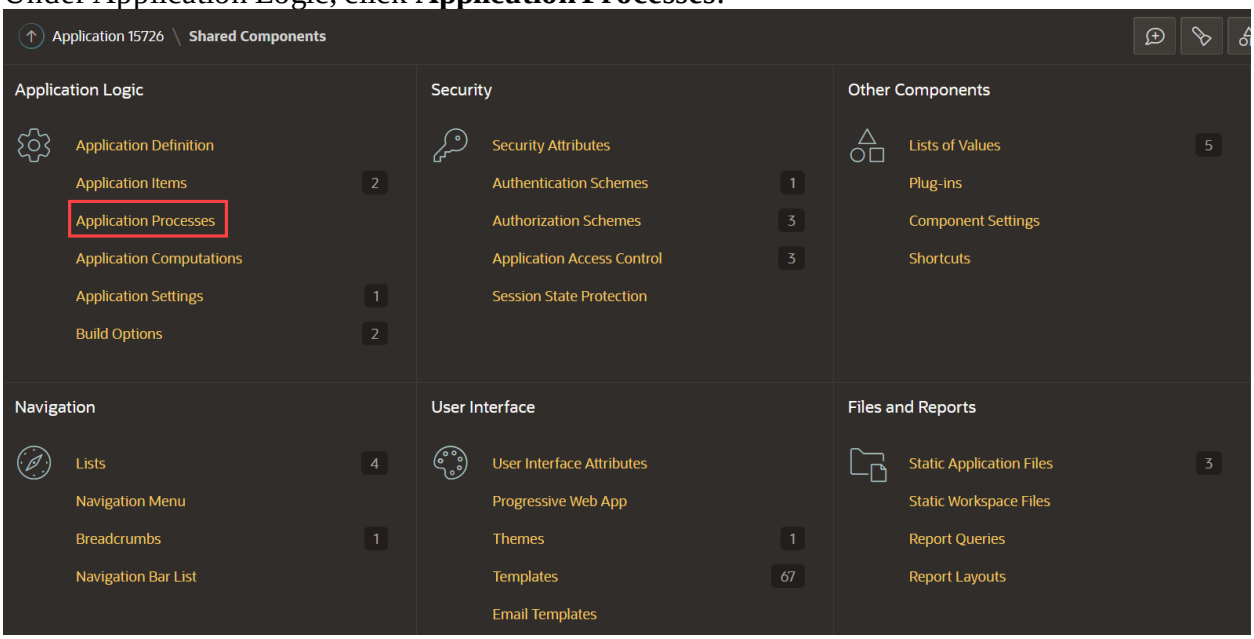
Task 2: Create Application Process

This process is needed to refresh the number of items in the Shopping Cart, which will be shown in the navigation bar.

1. Click on **Shared Components**.

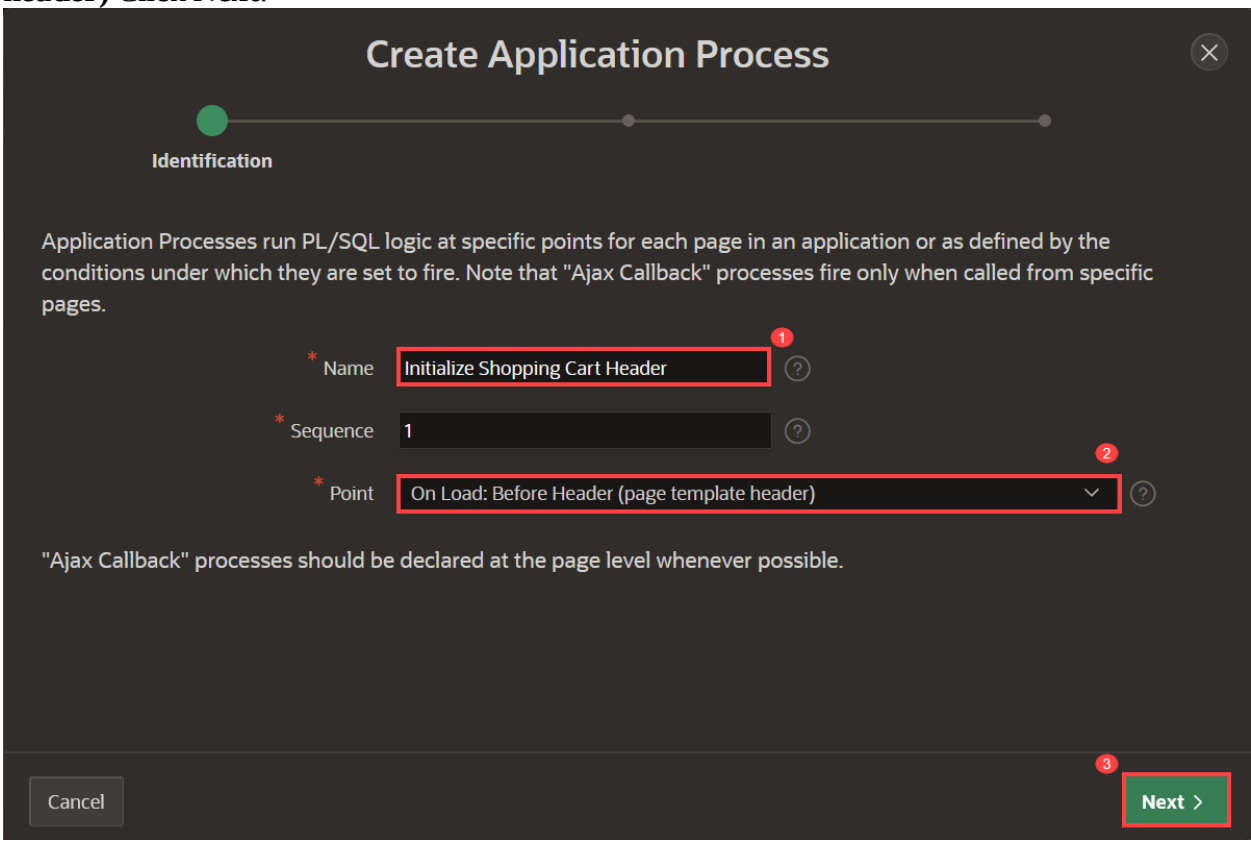


2. Under Application Logic, click **Application Processes**.



3. Click **Create** and enter the following:
    - For Name - enter **Initialize Shopping Cart Header**

- For Process Point - select **On Load: Before Header (page template header)** Click **Next**.



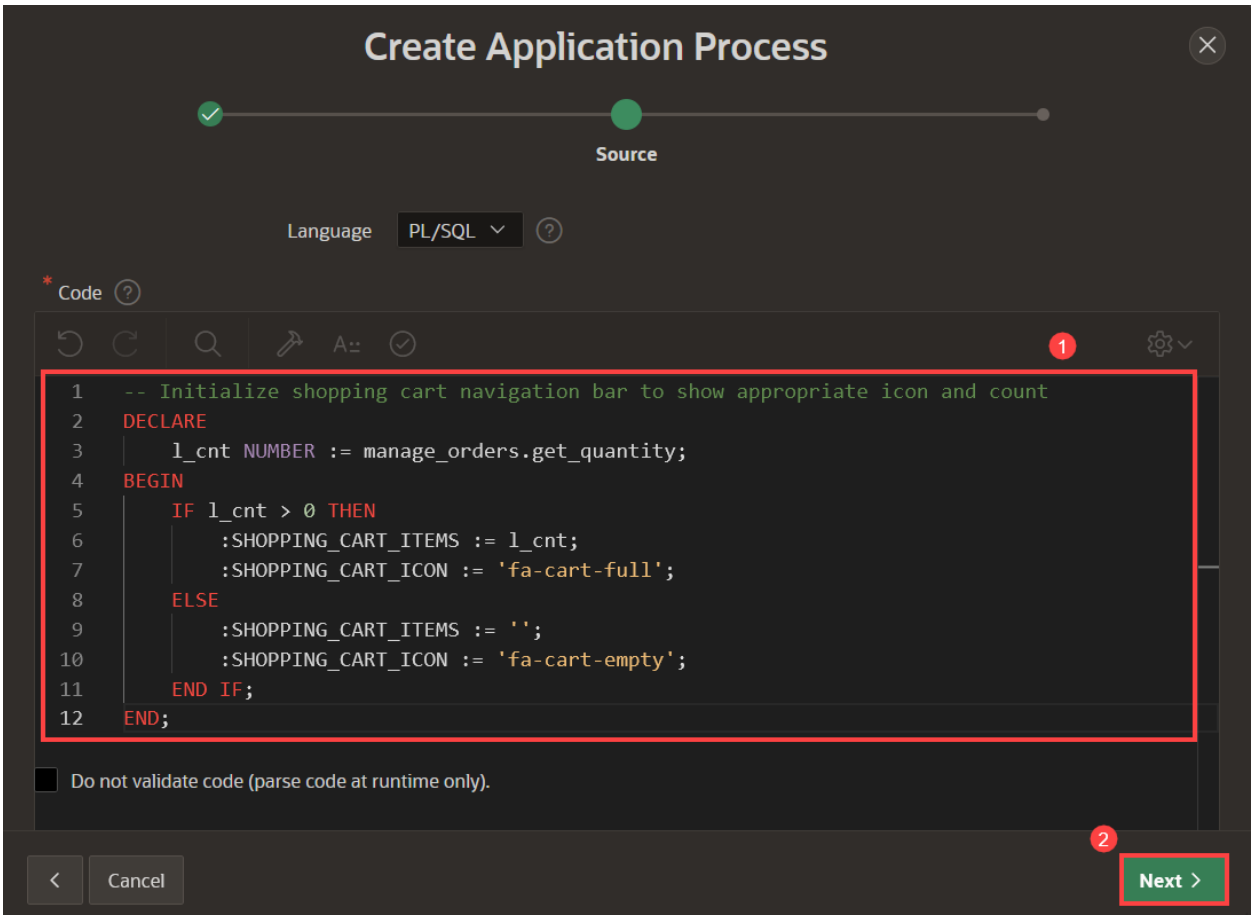
4. For Code, enter:

```
5. Copy-- Initialize shopping cart navigation bar to show
   appropriate icon and count

6. DECLARE

7.     l_cnt NUMBER := manage_orders.get_quantity;

8. BEGIN

9.     IF l_cnt > 0 THEN

10.         :SHOPPING_CART_ITEMS := l_cnt;

11.         :SHOPPING_CART_ICON := 'fa-cart-full';

12.     ELSE
```

```
        :SHOPPING_CART_ITEMS := '';
        :SHOPPING_CART_ICON := 'fa-cart-empty';
    END IF;
```

```
END;
```

Click **Next**.

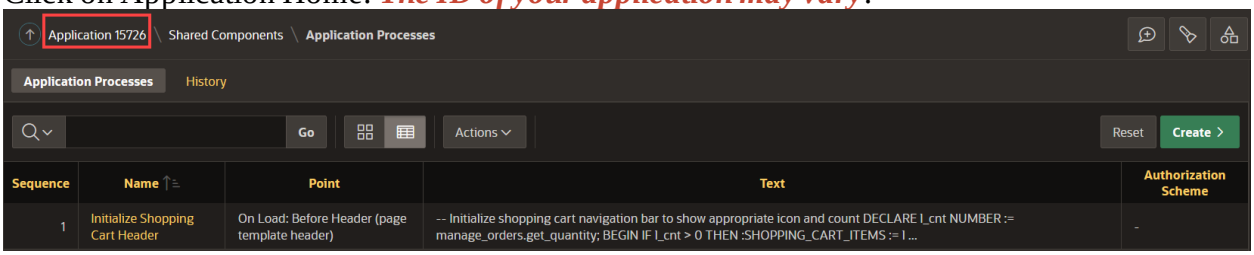


16. Click **Create Process**.

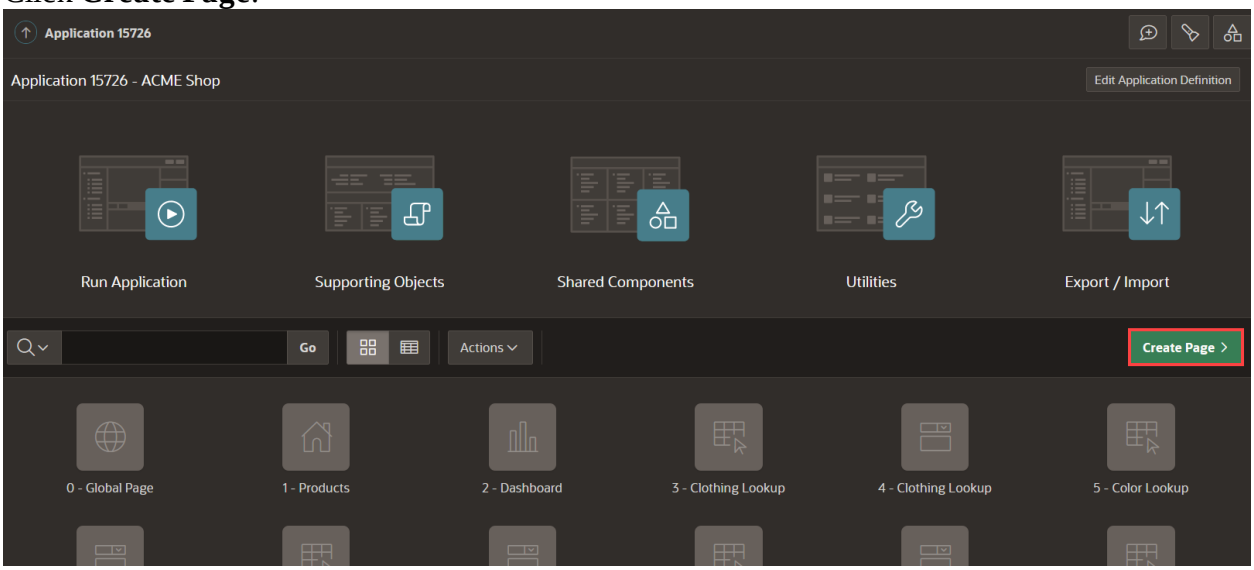
Task 3: Create a Normal Page - Shopping Cart

The shopping cart page allows users to review and edit the products in the cart. Additionally, users can create the order or clear the shopping cart.

1. Click on Application Home. ***The ID of your application may vary***.

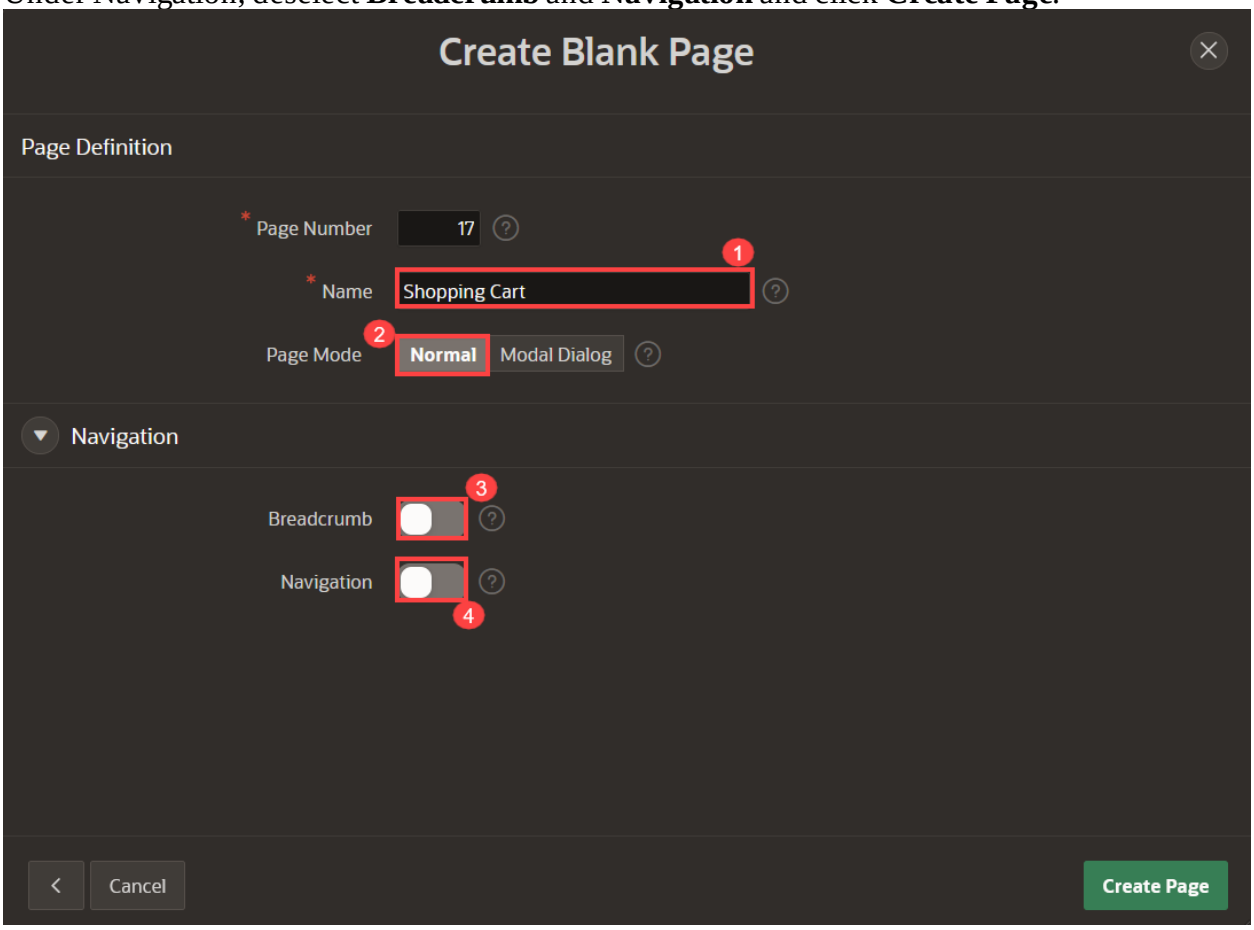


2. Click **Create Page**.



3. Select **Blank Page** and click **Next**.
4. Enter the following and click **Next**.
    - Page Number - enter **17**
    - For Name - enter **Shopping Cart**
    - For Page Mode - select **Normal**

5. Under Navigation, deselect **Breadcrumb** and **Navigation** and click **Create Page**.

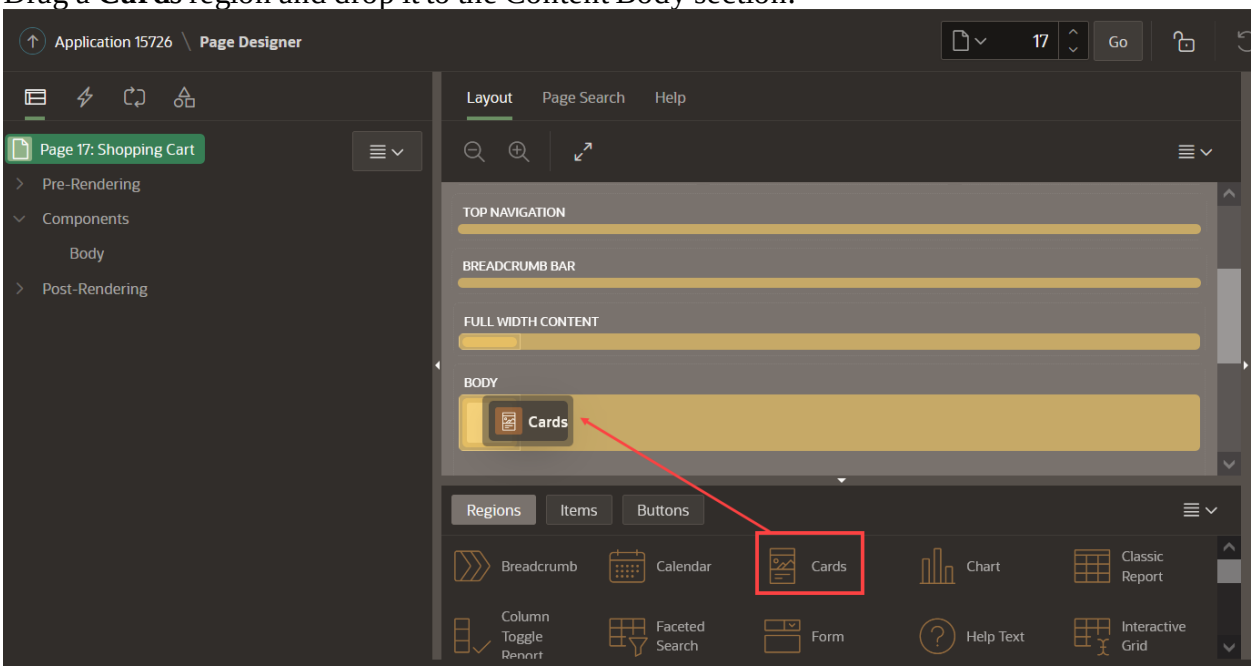


Task 4: Add a Cards Region

This region will list the items that have been added temporarily to the shopping cart.

1. In the new page created, navigate to the **Gallery Menu**.

2. Drag a **Cards** region and drop it to the Content Body section.

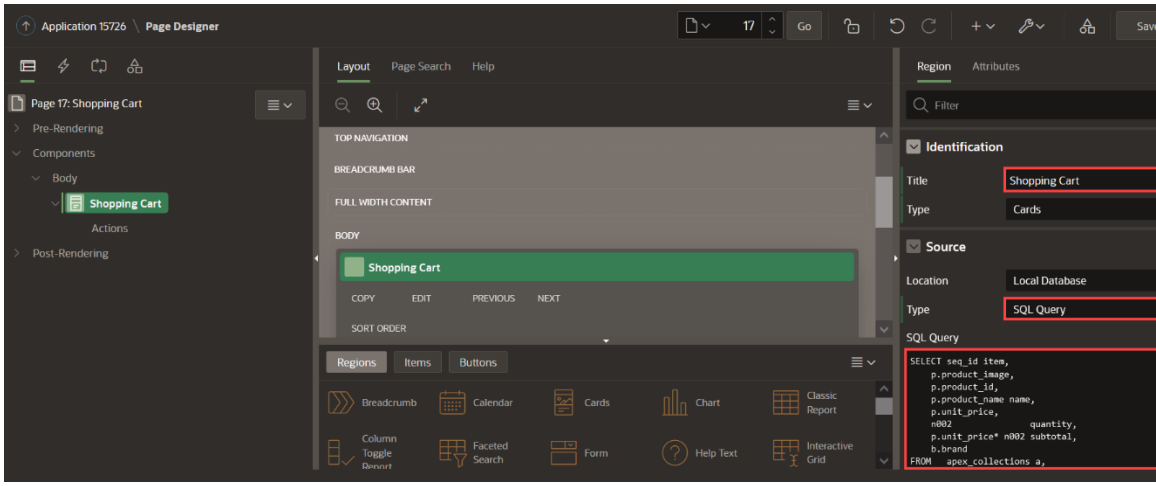


3. In the Property Editor, enter the following:
    - For Title - enter **Shopping Cart**
    - Under Source section:
        - For Type - select **SQL Query**
        - For SQL Query - enter the following SQL Query:

```
Copy
SELECT seq_id item,
    p.product_image,
    p.product_id,
    p.product_name name,
    p.unit_price,
    n002 quantity,
    p.unit_price* n002 subtotal,
```

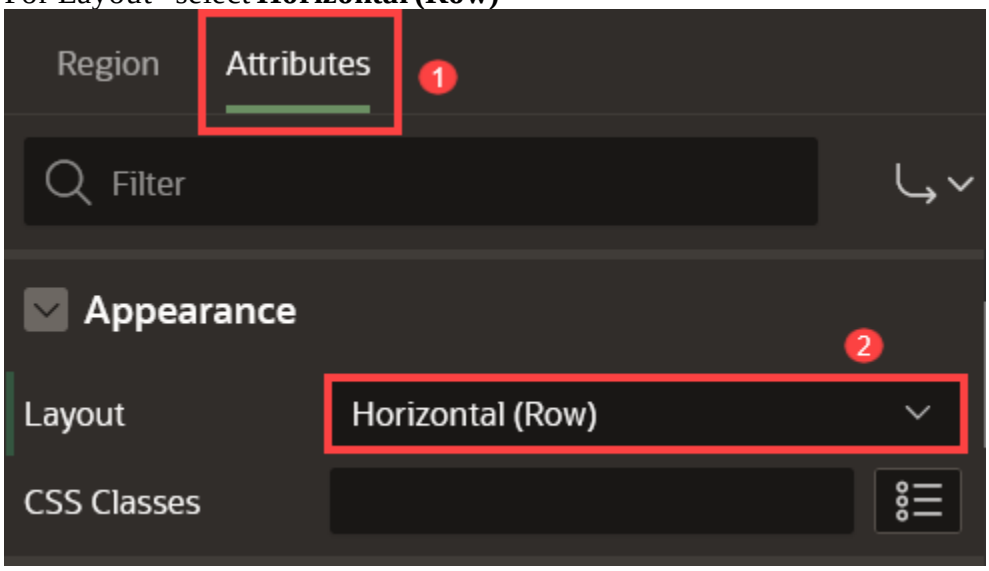
```
    b.brand
FROM   apex_collections a,
    products p,
    json_table (p.product_details, '$' columns (
brand varchar2(4000) path '$.brand') ) b
WHERE  collection_name = 'PRODUCTS'
```

```
AND    p.product_id = a.n001
```

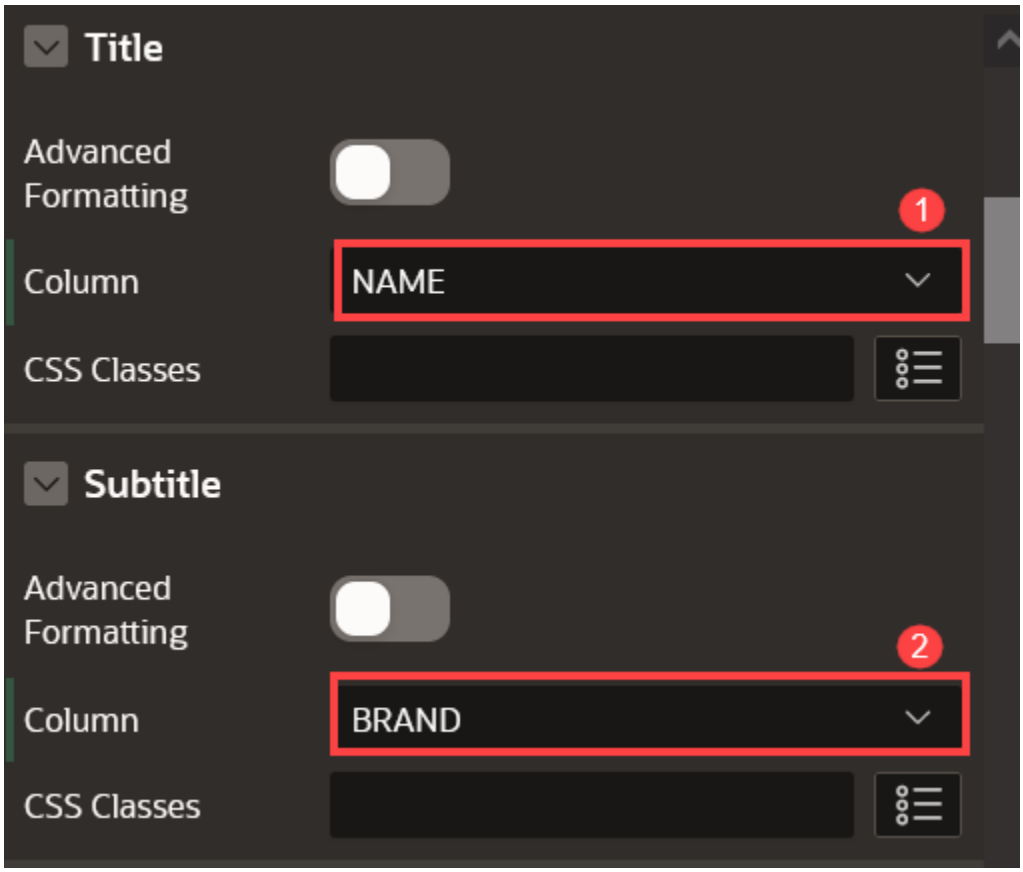


4. Click on **Attributes** and enter the following:
    - Under Appearance section:

- For Layout - select **Horizontal (Row)**



- Under Title section:
  - For Column - select **NAME**
- Under Subtitle section:
  - For Column - select **BRAND**



- Under Body section:

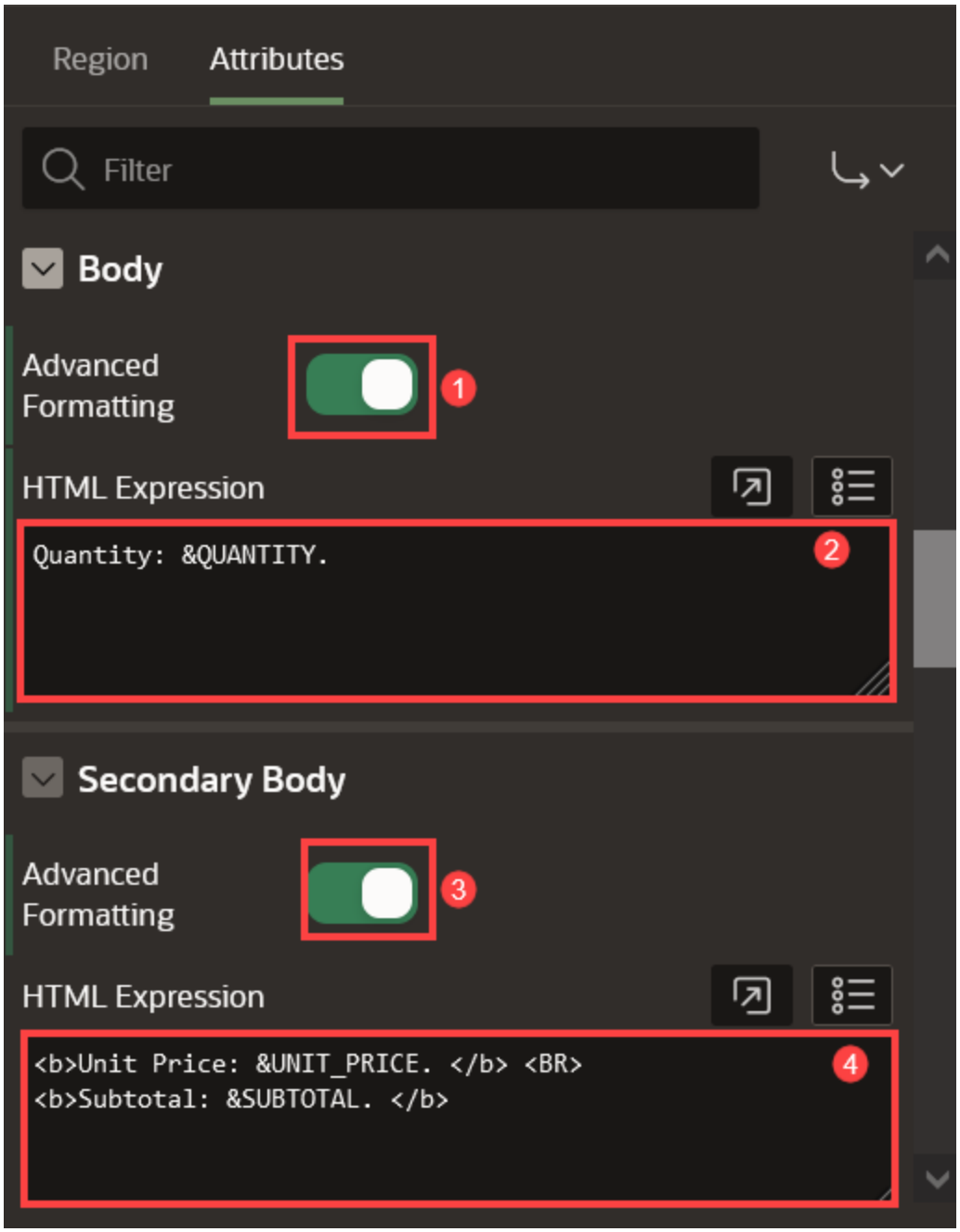
- Set Advanced Formatting to **On**
- For HTML Expression - enter the following:

```
CopyQuantity: &QUANTITY.
```

- Under Secondary Body section:
  - Set Advanced Formatting to **On**
  - For HTML Expression - enter the following:

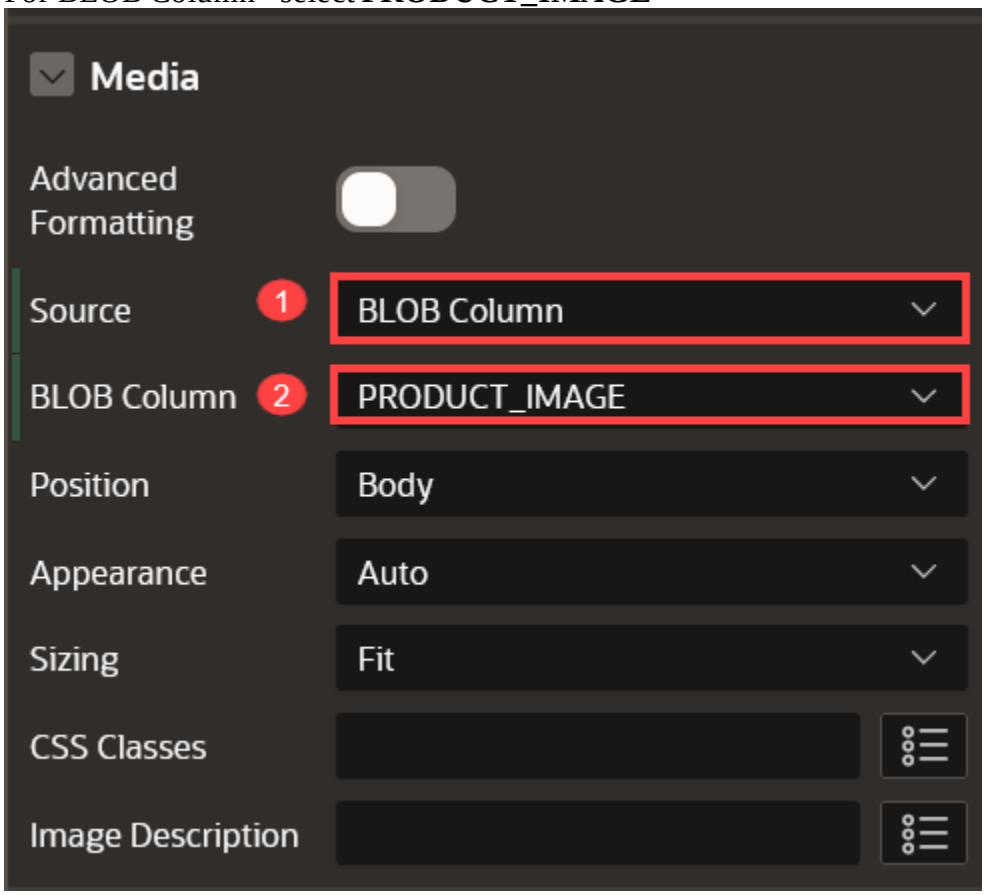
```
Copy<b>Unit Price: &UNIT_PRICE. </b> <BR>
```

```
<b>Subtotal: &SUBTOTAL. </b>
```

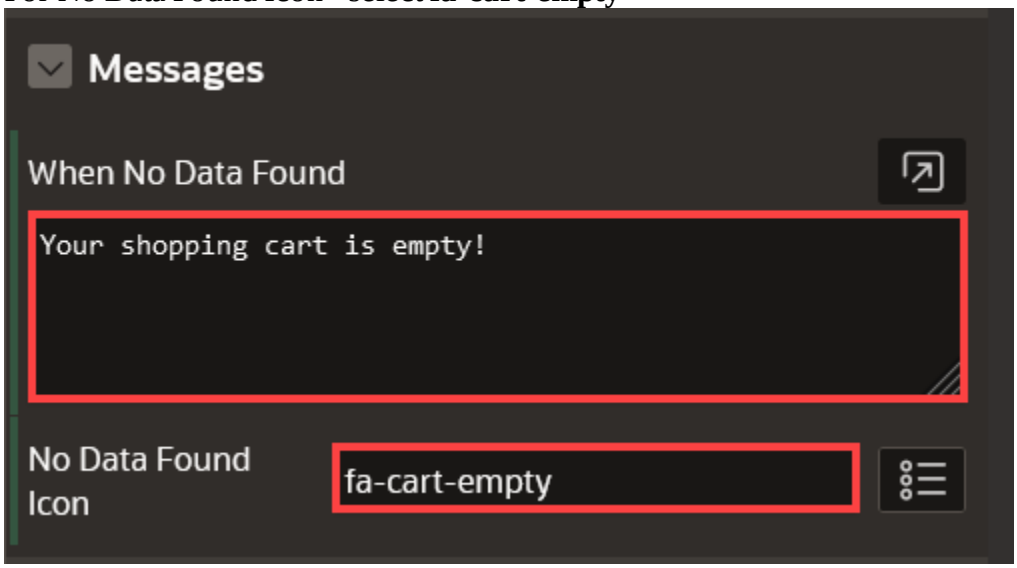


- Under Media section:
  - For Source - select **BLOB Column**

- For BLOB Column - select **PRODUCT_IMAGE**

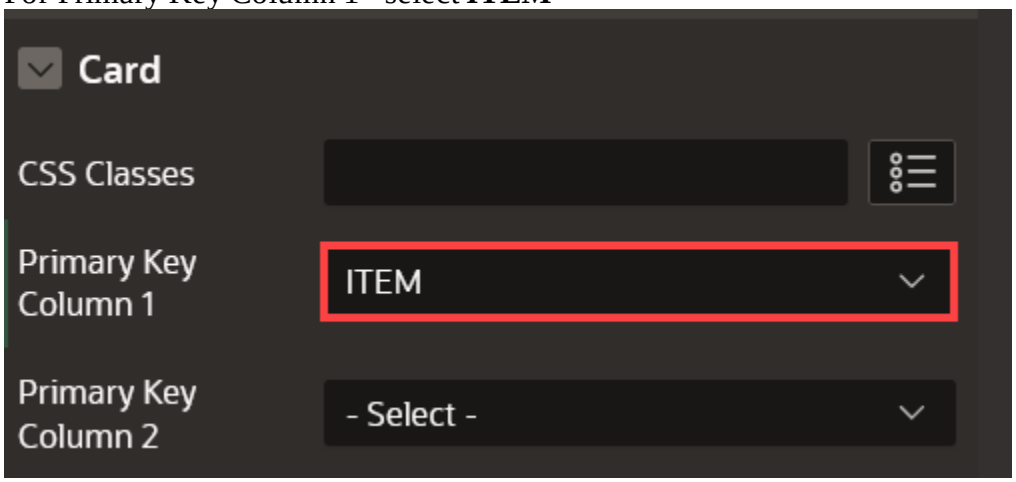


- Under Messages section:
  - For When No Data Found - enter: **Your shopping cart is empty!**
  - For No Data Found Icon - select **fa-cart-empty**



- Under Card section:

- For Primary Key Column 1 - select **ITEM**

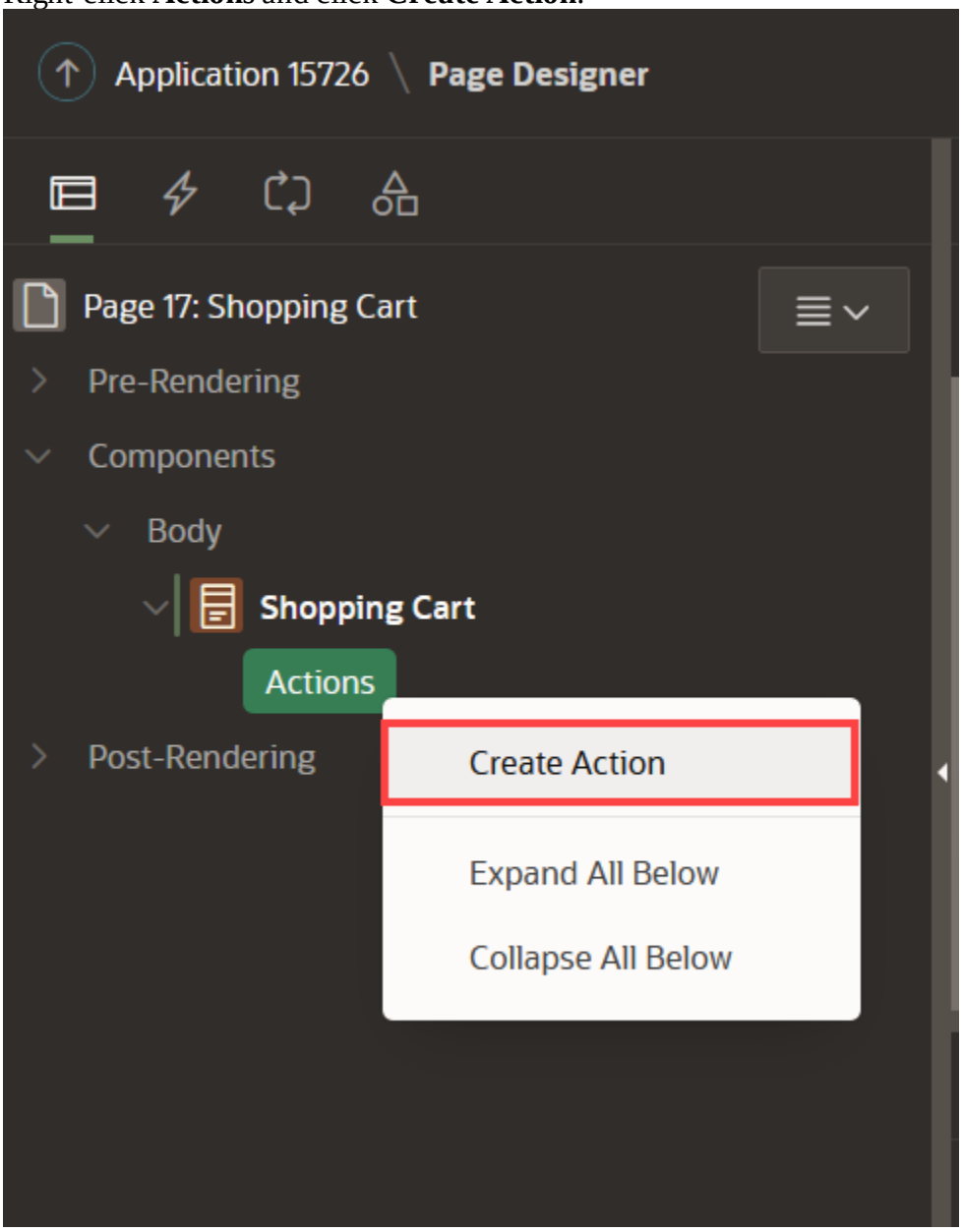


Task 5: Add an Action to the Shopping Cart

This action allows customers to open a page to edit a particular item in the shopping cart.

1. In the Rendering tree (left pane), navigate to **Actions** under **Shopping Cart**.

2. Right-click **Actions** and click **Create Action**.

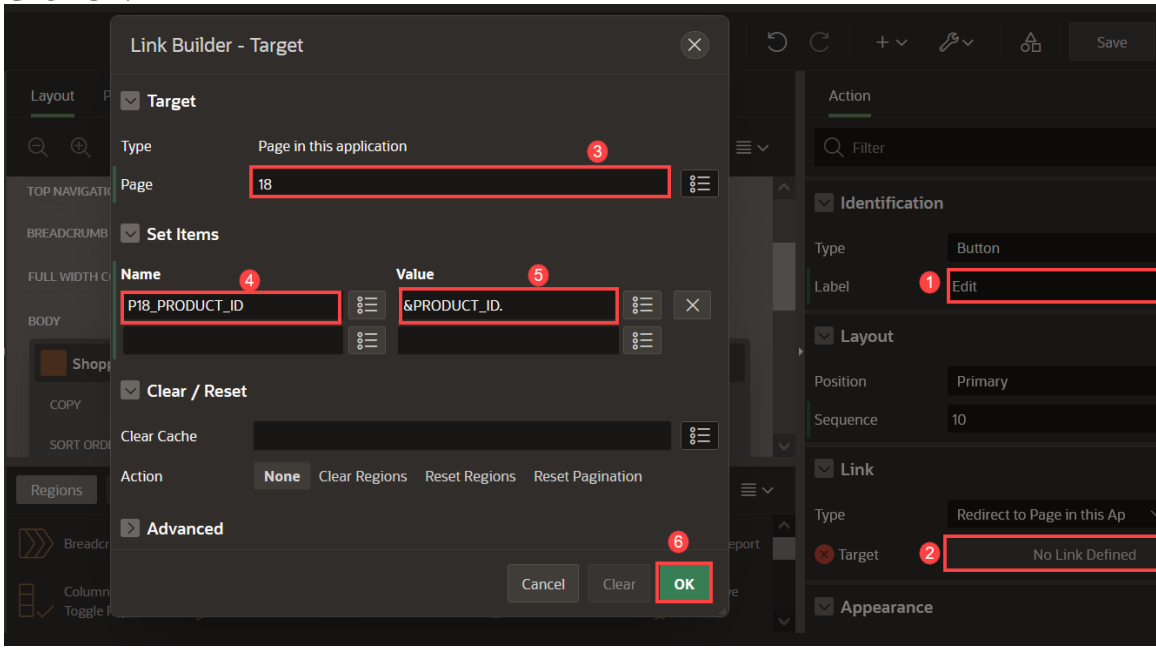


3. In the Property Editor, enter the following:
    - For Label - enter **Edit**
    - For Target - click **No Link Defined**:
        - For Page - enter **18**
          {Note: Page 18 will be created in the next lab}
        - Set items as follows:

Table 2: Build a Starter Online Shopping App with Oracle APEX! | Lab 6: Create the Shopping Cart Page

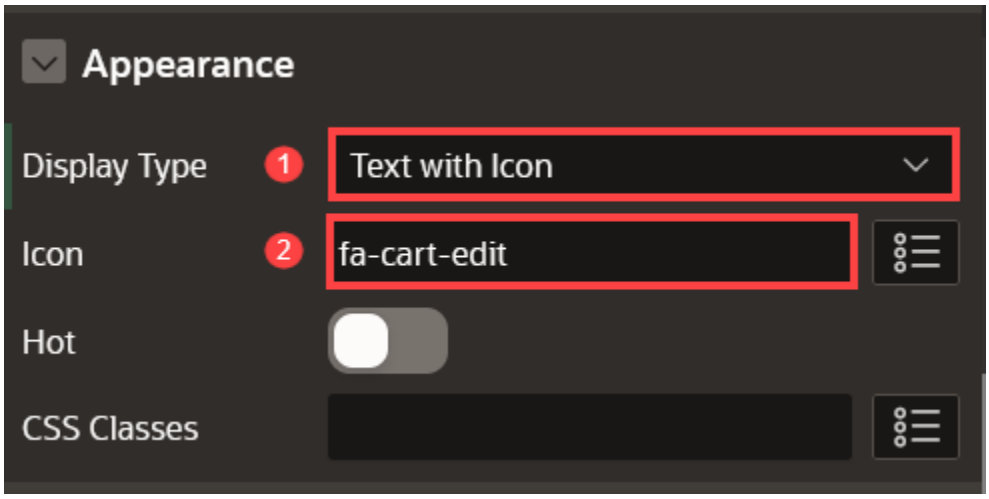
Name	Value
P18_PRODUCT_ID	&PRODUCT_ID.

- Click **Ok**.



  - For Display Type - select **Text with Icon**

- For Icon - enter **fa-cart-edit**

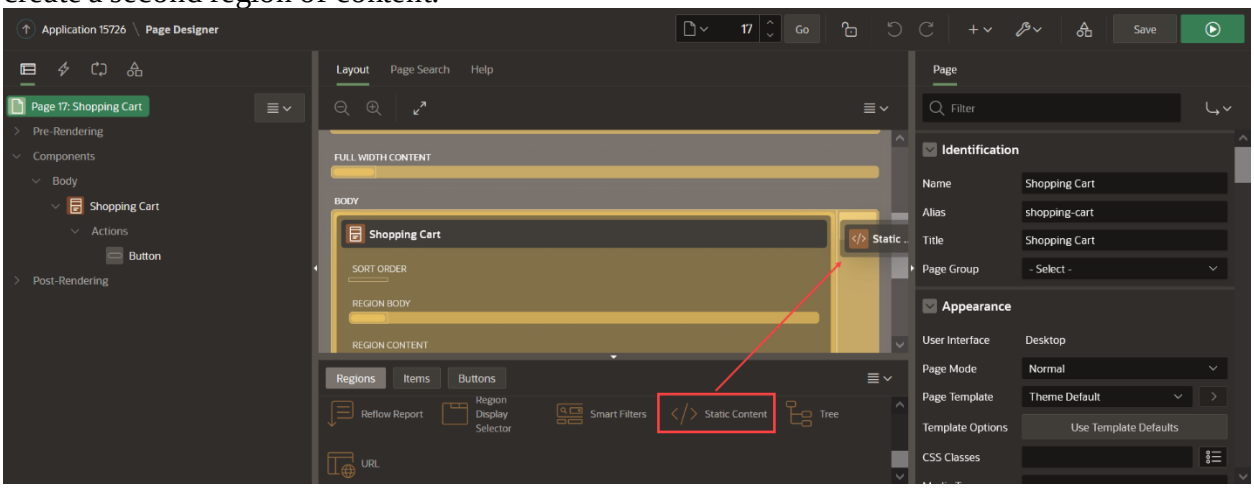


This configures the (Edit) button to open page 18, passing the value of the PRODUCT_ID column of the current card as the value for the page item P18_PRODUCT_ID in that called page.

Task 6: Add Items and Buttons to the Page

1. Navigate to the **Gallery Menu**.

2. Drag a **Static Content** region and drop it to the right of the Shopping Cart region to create a second region of content.

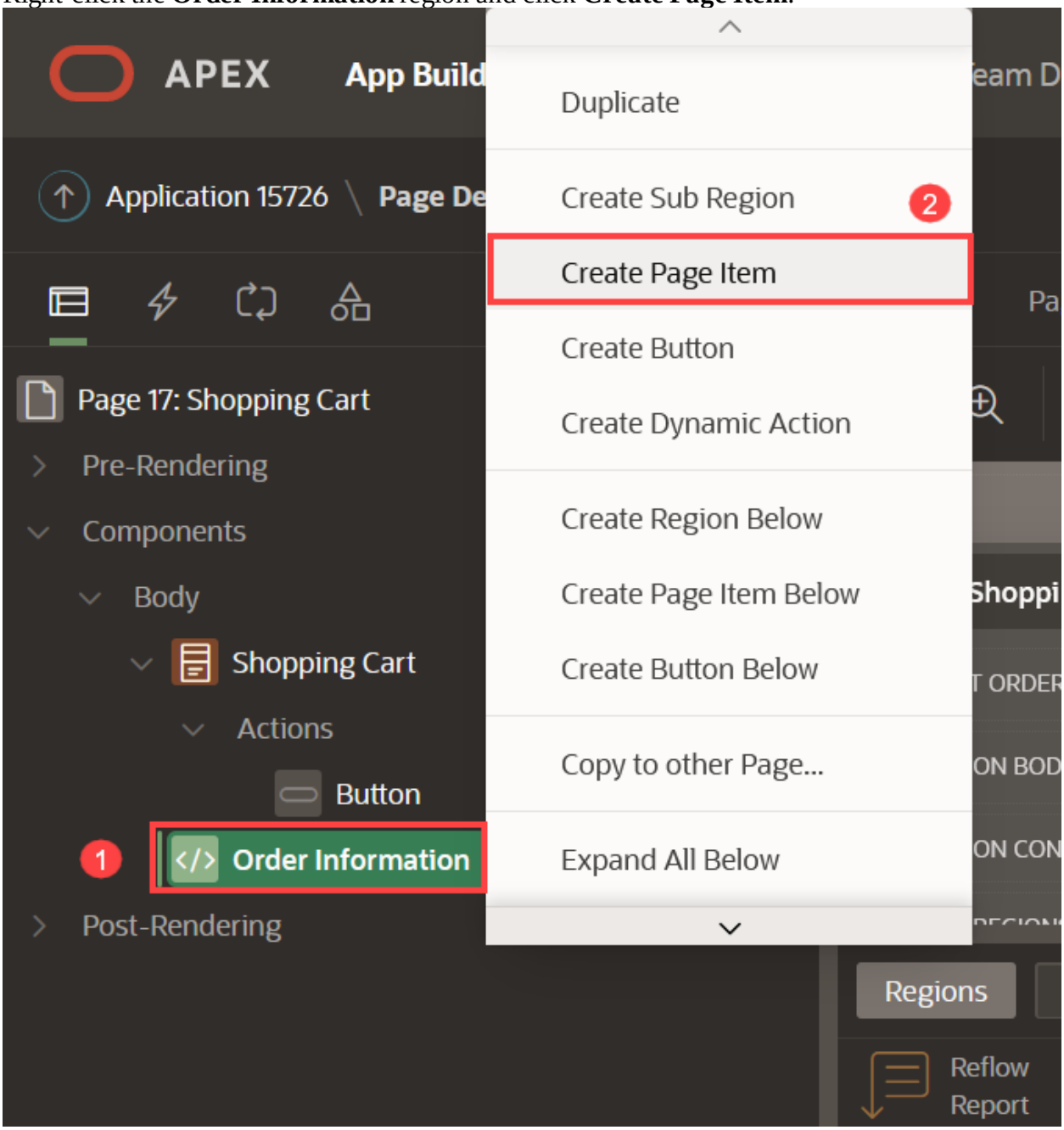


3. In the Property Editor, enter the following:

   - For Name - enter **Order Information**

4. Navigate to the **Order Information** (left pane) region.

5. Right-click the **Order Information** region and click **Create Page Item**.



6. Create five items as follows:

Table 3: Details of the Page Items

Name	Type	Label	Template	Value Required
P17_CUSTOMER_EMAIL	Text Field	Email Address	Optional - Floating	Off
P17_CUSTOMER_FULLNAME	Text Field	Full Name	Optional - Floating	Off
P17_ORDER_ID	Hidden			
P17_CUSTOMER_ID	Hidden			
P17_STORE	Select List	Store	Optional - Floating	Off

7. For **P17_STORE** item, in the list of values section, configure the type as follows:
    - For Type - select **SQL Query**
    - For SQL Query - enter the following SQL Query:

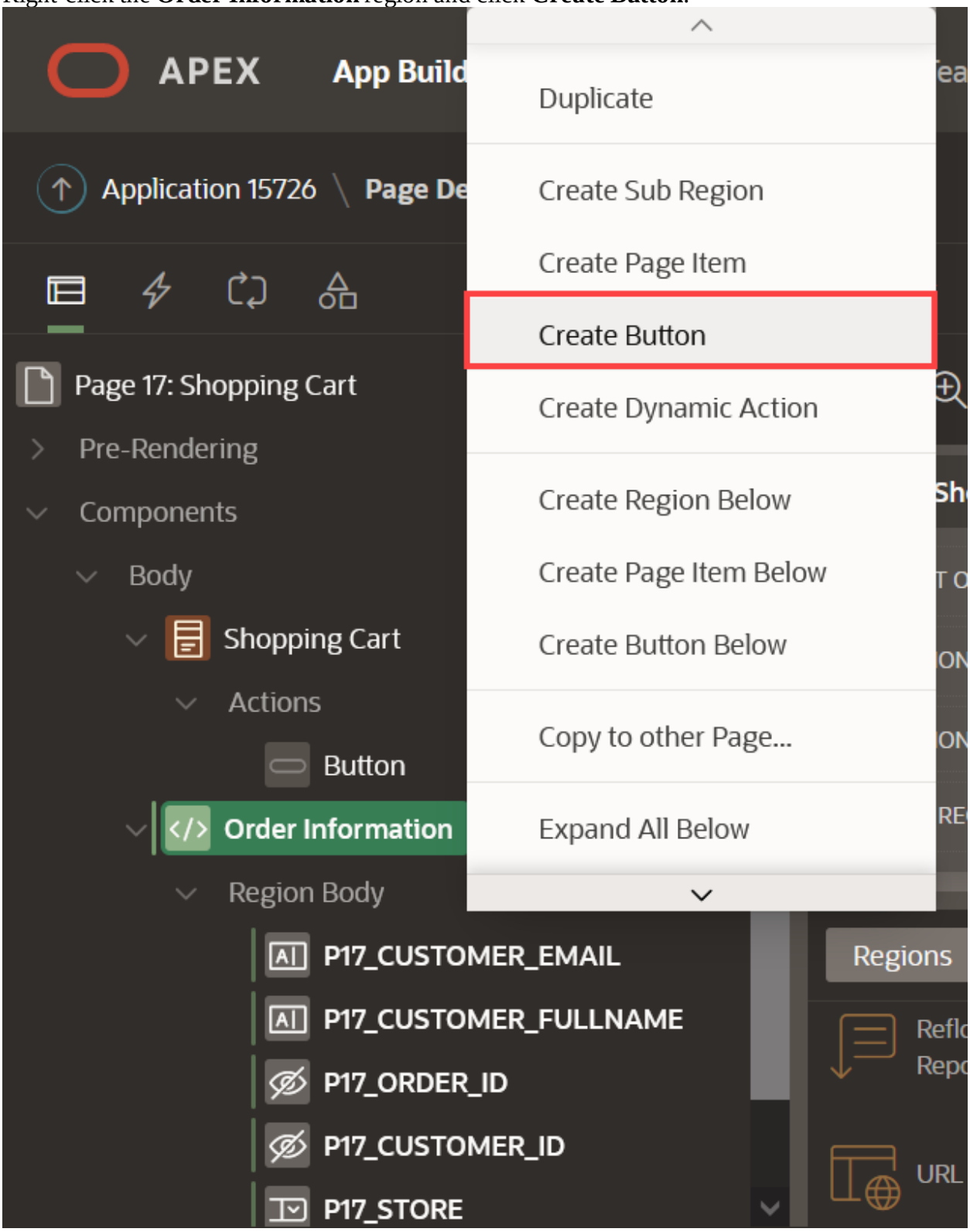
```
Copyselect STORES.STORE_NAME as STORE_NAME,
    STORES.STORE_ID as STORE_ID
```

```
from STORES STORES
```

- Set Display Extra Values - to **Off**
- For Null Display Value - enter **- Select a Store -**

8. Navigate to the **Order Information** (left pane) region.

9. Right-click the **Order Information** region and click **Create Button**.



10. Create two buttons as follows:

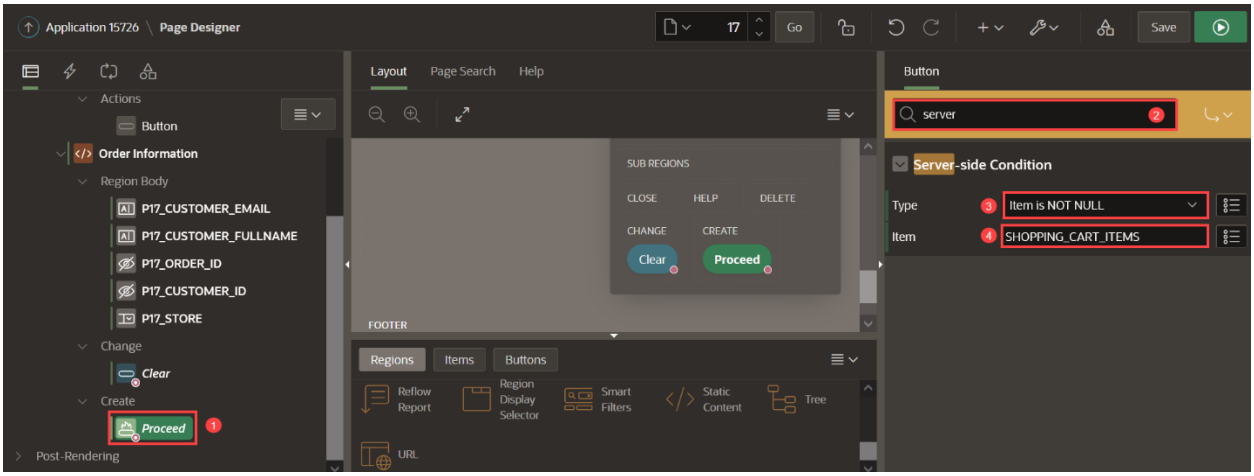
Table 4: Details of the 2 Buttons

Button Name	Label	Position	Button Template	Hot	Icon
Proceed	Proceed to Checkout	Create	Text	On	
Clear	Clear Shopping Cart	Change	Text with Icon	Off	fa-cart-empty

11. Under Server-side Condition:

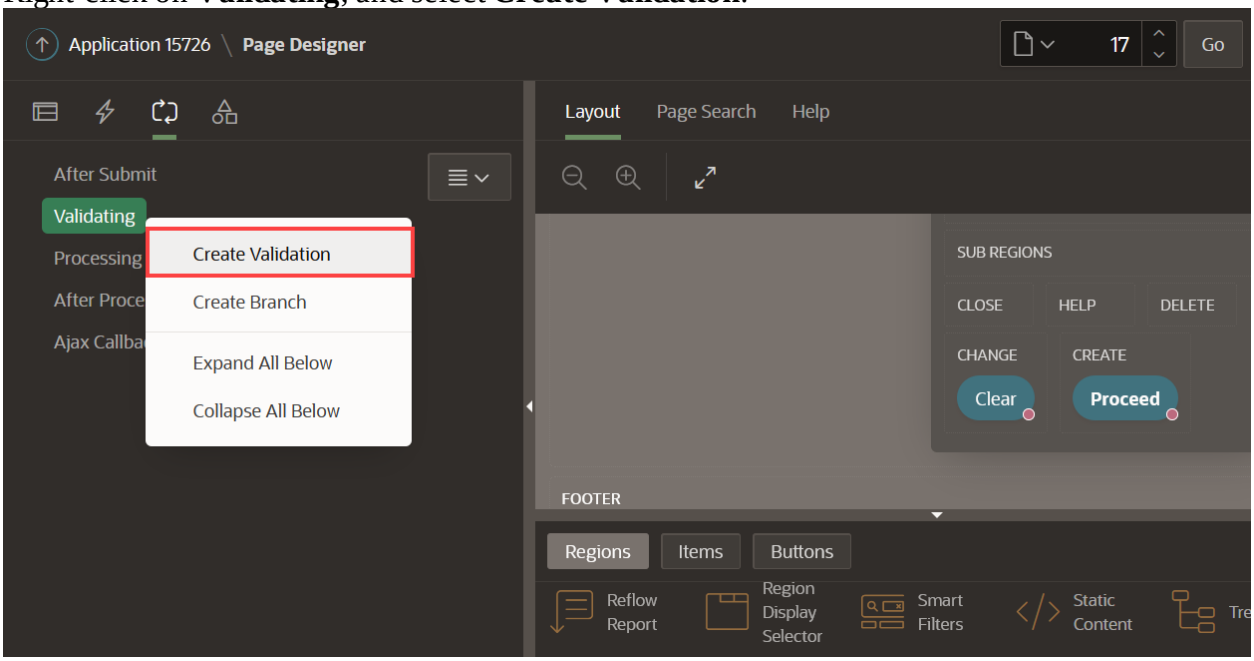
Table 5: Server Side Condition for the 2 Buttons

Button Name	Type	Item
Proceed	Item is NOT NULL	SHOPPING_CART_ITEMS
Clear	Item is NOT NULL	SHOPPING_CART_ITEMS

12. 

Task 7: Add Validations to the Page

1. In the Rendering tree (left pane), click **Processing** tab.
2. Right-click on **Validating**, and select **Create Validation**.



3. Create three validations for the following items: **Name**, **Email**, and **Store**

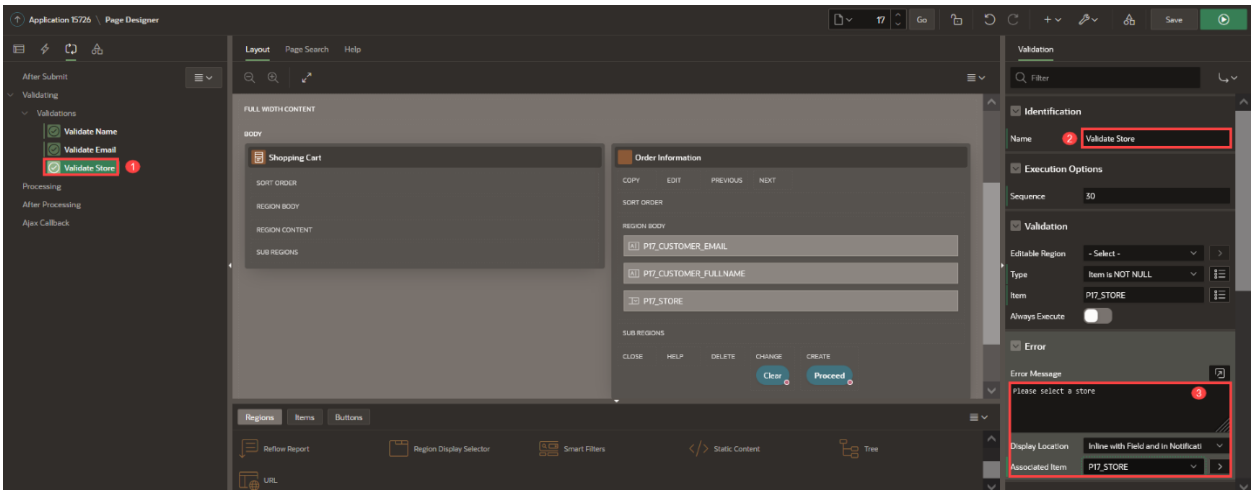
Table 6: Validations for the Items

Name	Type (under Validation)	Item
Validate Name	Item is NOT NULL	P17_CUSTOMER_FULLNAME
Validate Email	Item is NOT NULL	P17_CUSTOMER_EMAIL
Validate Store	Item is NOT NULL	P17_STORE

4. Under Error:

Table 7: Error Messages for the Validations

Error Message	Display Location	Associated Item
Please enter your name	Inline with Field and in Notification	P17_CUSTOMER_FULLNAME
Please enter your email address	Inline with Field and in Notification	P17_CUSTOMER_EMAIL
Please select a store	Inline with Field and in Notification	P17_STORE

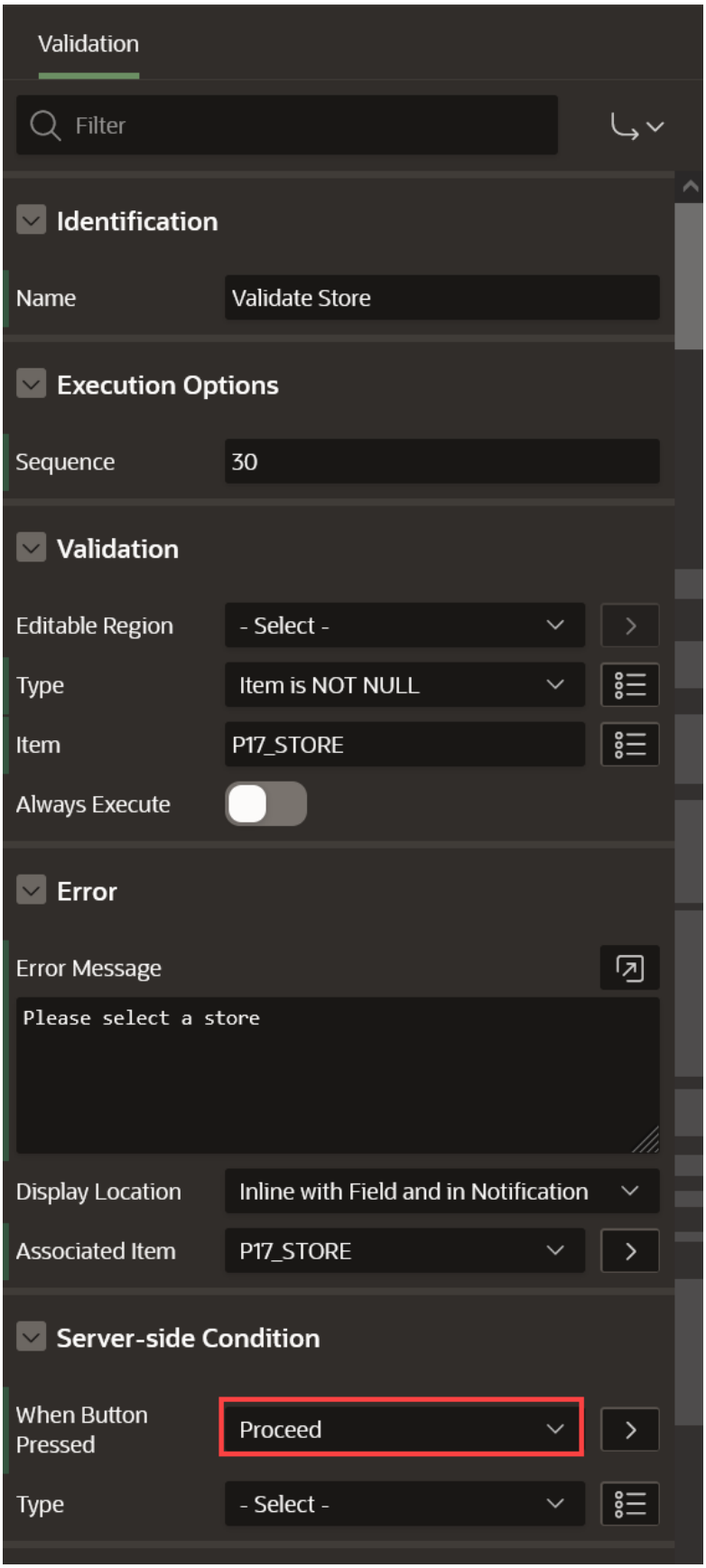
5. 

6. As these validations only apply when user proceeds to checkout, let's create that condition. Under Server-side Condition, set the following:

Table 8: Server-side Conditions

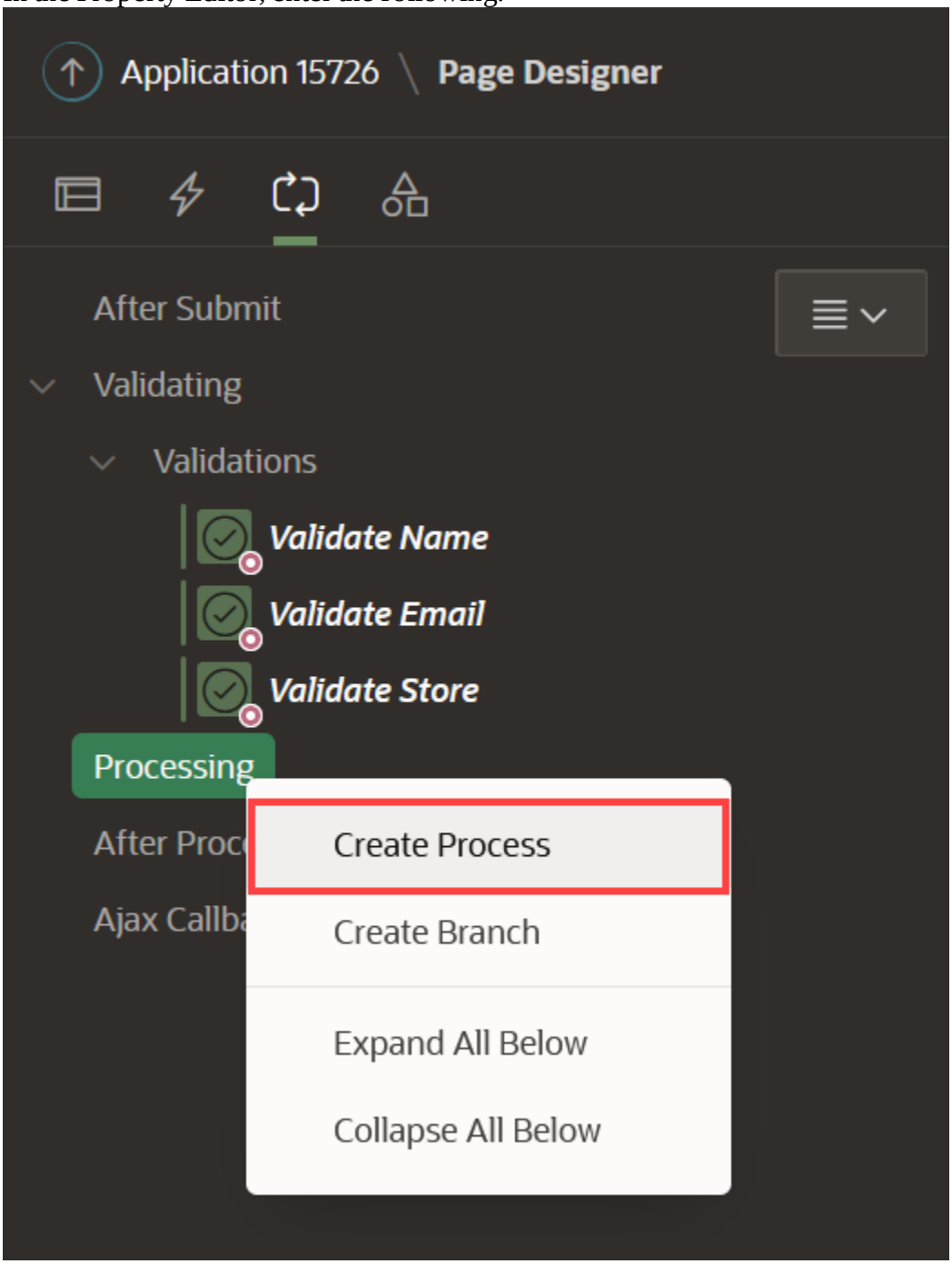
Name	When Button Pressed
Validate Name	Proceed
Validate Email	Proceed
Validate Store	Proceed

7.


Validation
Filter
Identification
Name
Validate Store
Execution Options
Sequence
30
Validation
Editable Region
- Select -
Type
Item is NOT NULL
Item
P17_STORE
Always Execute
Error
Error Message
Please select a store
Display Location
Inline with Field and in Notification
Associated Item
P17_STORE
Server-side Condition
When Button Pressed
Proceed
Type
- Select -

Task 8: Add Process to Create the Order

1. On the **Processing** tab (left pane).
2. Right-click **Processing** and click **Create Process**.
3. In the Property Editor, enter the following:

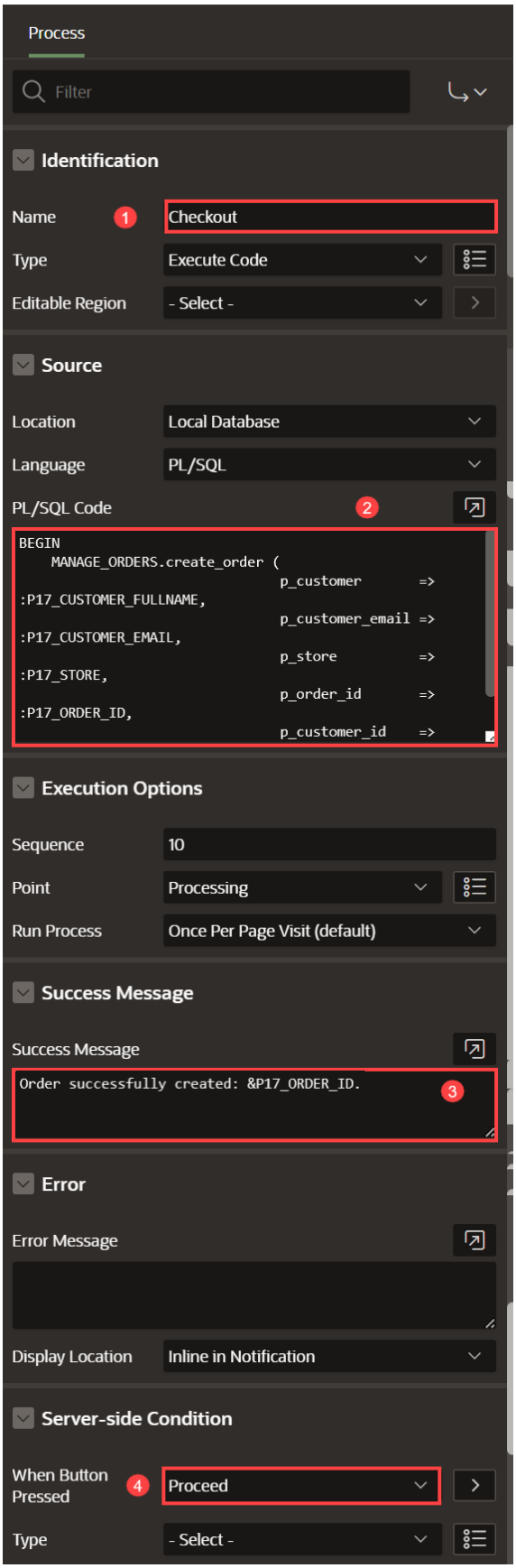


- For Name - enter **Checkout**
- For Type -select **Execute Code**
- For PL/SQL Code - enter the following PL/SQL code:

```
Copy
```

```
BEGIN
    MANAGE_ORDERS.create_order (
                                p_customer       => :P17_CUSTOMER_FULLNAME,
                                p_customer_email => :P17_CUSTOMER_EMAIL,
                                p_store          => :P17_STORE,
                                p_order_id       => :P17_ORDER_ID,
                                p_customer_id    => :P17_CUSTOMER_ID);
END;
```

- For Success Message, enter **Order successfully created: &P17_ORDER_ID.**
- Under Server-side conditions, for When Button Pressed, select **Proceed**


Process
Filter
Identification
Name
1
Checkout
Type
Execute Code
Editable Region
- Select -
Source
Location
Local Database
Language
PL/SQL
PL/SQL Code
2
BEGIN
MANAGE_ORDERS.create_order (
p_customer =>
:P17_CUSTOMER_FULLNAME,
p_customer_email =>
:P17_CUSTOMER_EMAIL,
p_store =>
:P17_STORE,
p_order_id =>
:P17_ORDER_ID,
p_customer_id =>
Execution Options
Sequence
10
Point
Processing
Run Process
Once Per Page Visit (default)
Success Message
Success Message
Order successfully created: &P17_ORDER_ID.
3
Error
Error Message
Display Location
Inline in Notification
Server-side Condition
When Button Pressed
4
Proceed
Type
- Select -

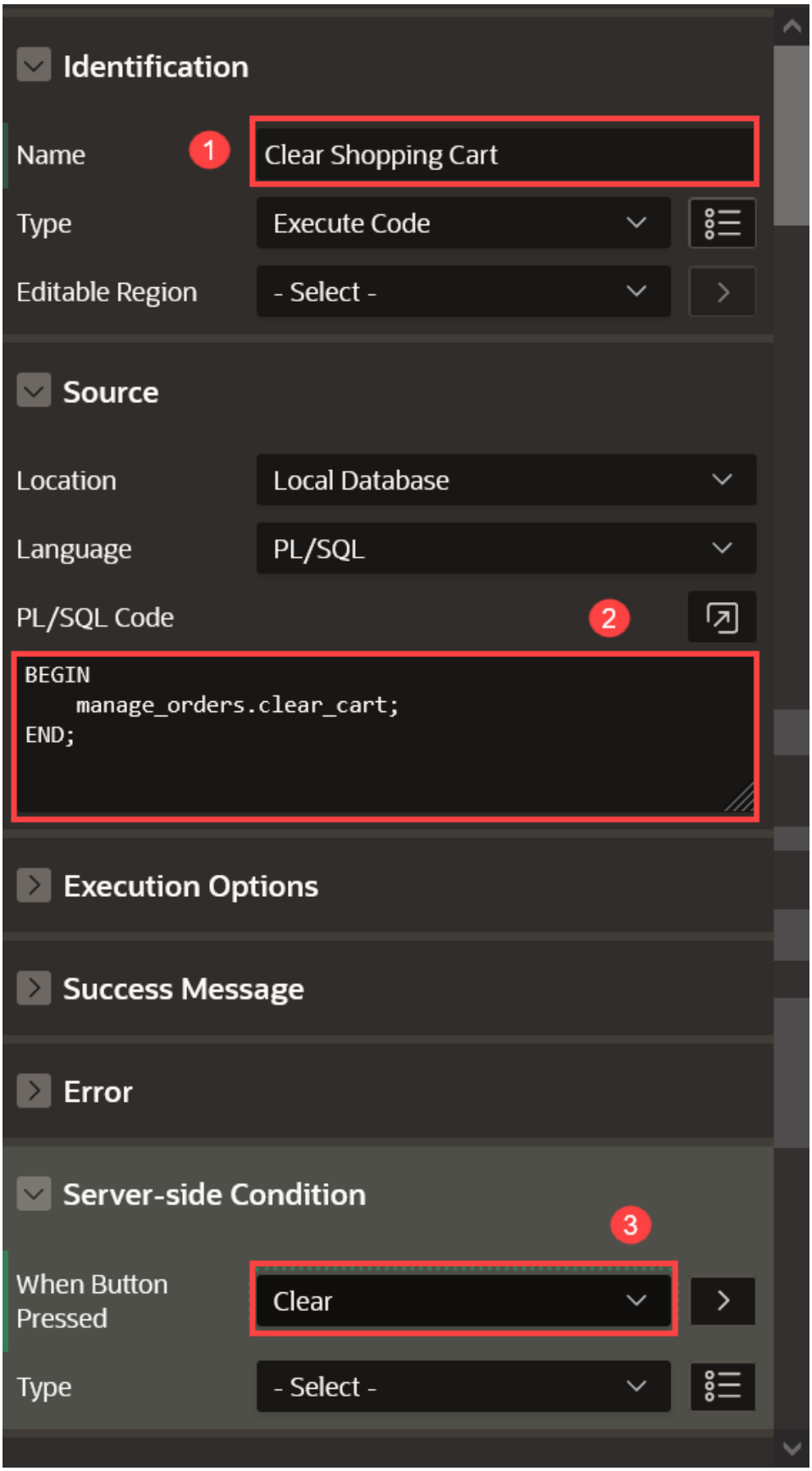
Task 9: Add Process to Clear the Shopping Cart

1. On the **Processing** tab (left pane).
2. Right-click **Processing** and click **Create Process**.
3. Create a second process to clear the shopping cart. In the Property Editor, enter the following:
    - For Name - enter **Clear Shopping Cart**
    - For Type - select **Execute Code**
    - For PL/SQL Code - enter the following PL/SQL code:

```
4. CopyBEGIN

5.     manage_orders.clear_cart;
```

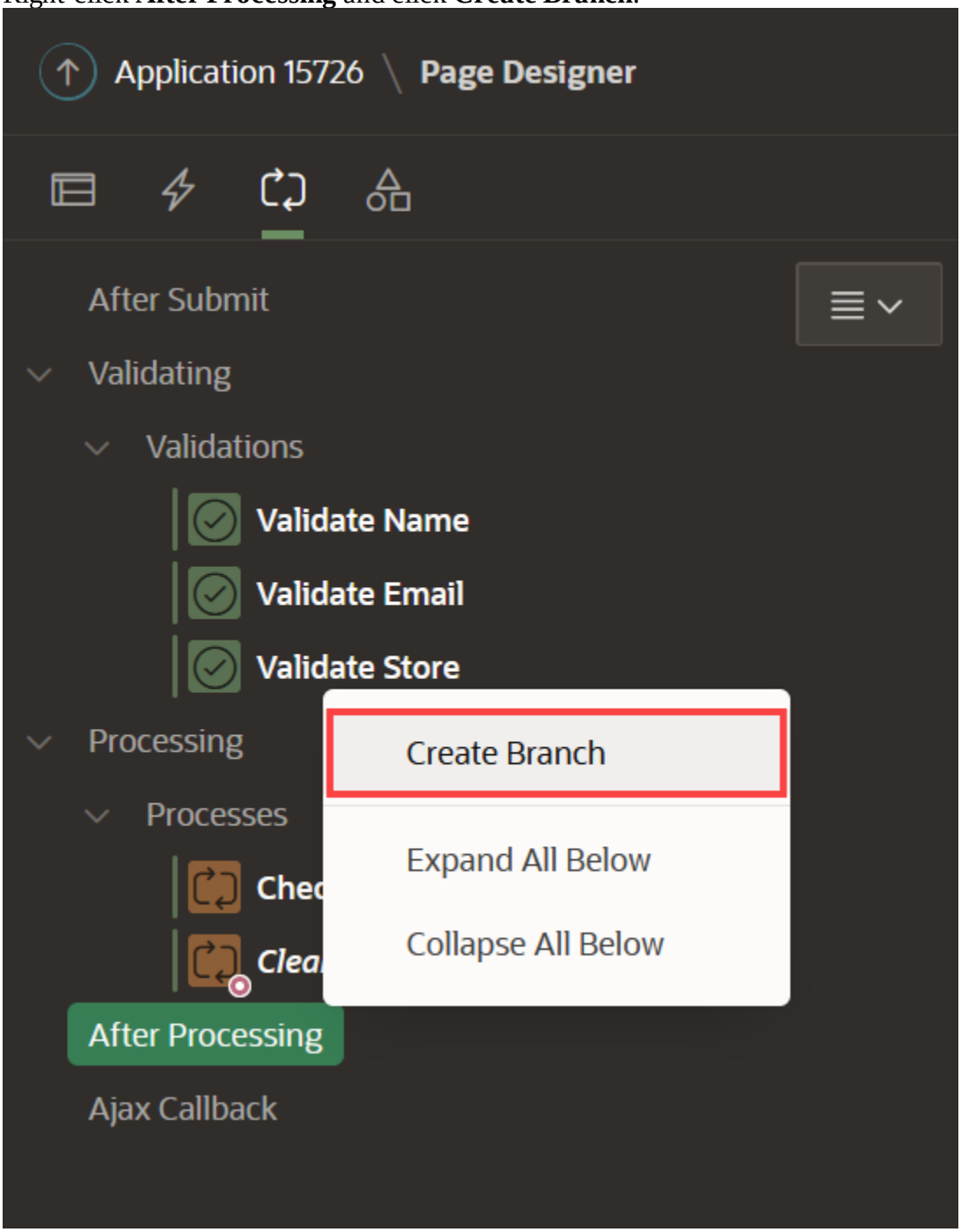
```
END;
```

    - For When Button Pressed, select **Clear**


Identification
Name
1
Clear Shopping Cart
Type
Execute Code
Editable Region
- Select -
Source
Location
Local Database
Language
PL/SQL
PL/SQL Code
2
BEGIN
manage_orders.clear_cart;
END;
Execution Options
Success Message
Error
Server-side Condition
3
When Button Pressed
Clear
Type
- Select -

Task 10: Add Branches to the Page

1. On the **Processing** tab (left pane).
2. Right-click **After Processing** and click **Create Branch**.



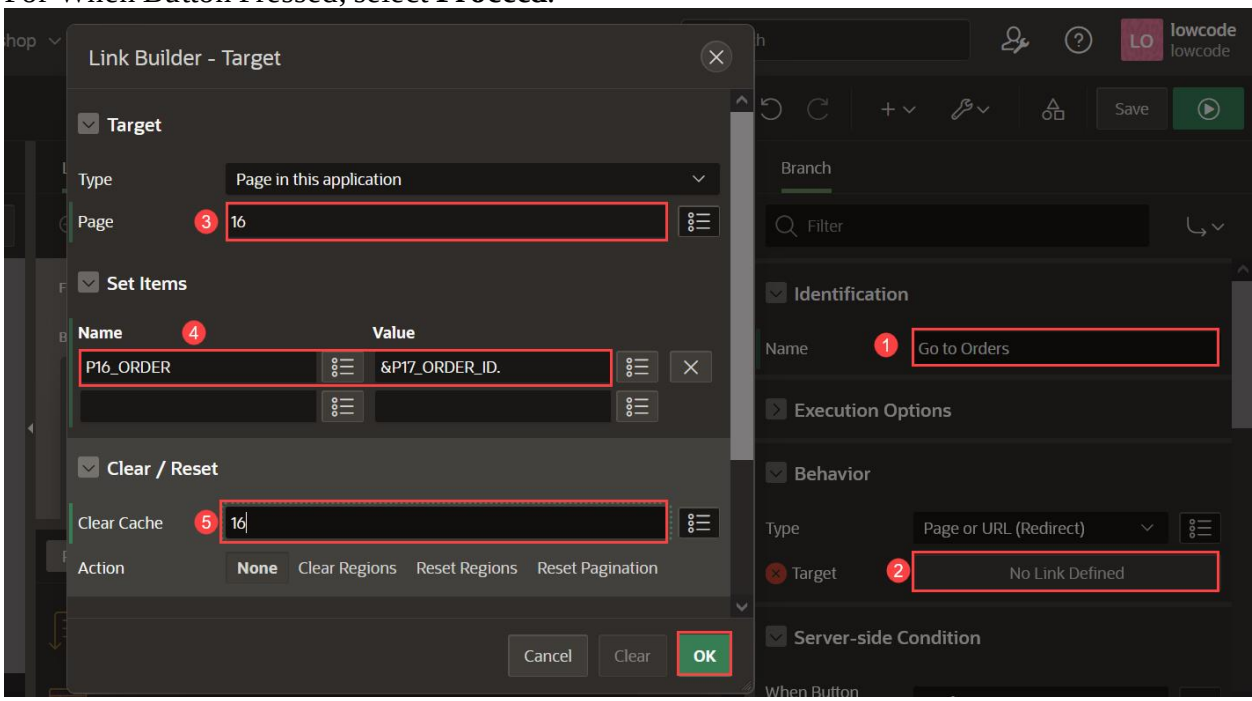
3. In the Property Editor, enter the following:

- For Name - enter **Go to Orders**
- Navigate to Target attribute and click **No Link Defined**.
  - For Type - select **Page in this application**
  - For Page - enter **16**
  - For Set Items - enter:

Table 9: Build a Starter Online Shopping App with Oracle APEX! | Lab 6: Create the Shopping Cart Page

Name	Value
P16_ORDER	&P17_ORDER_ID.

  - For Clear Cache - enter **16**.
  - Click **OK**.
- For When Button Pressed, select **Proceed**.



4. Create a second branch when user clears the shopping cart. Right-click on **After Processing** and click **Create Branch**.

5. In the Property Editor, enter the following:

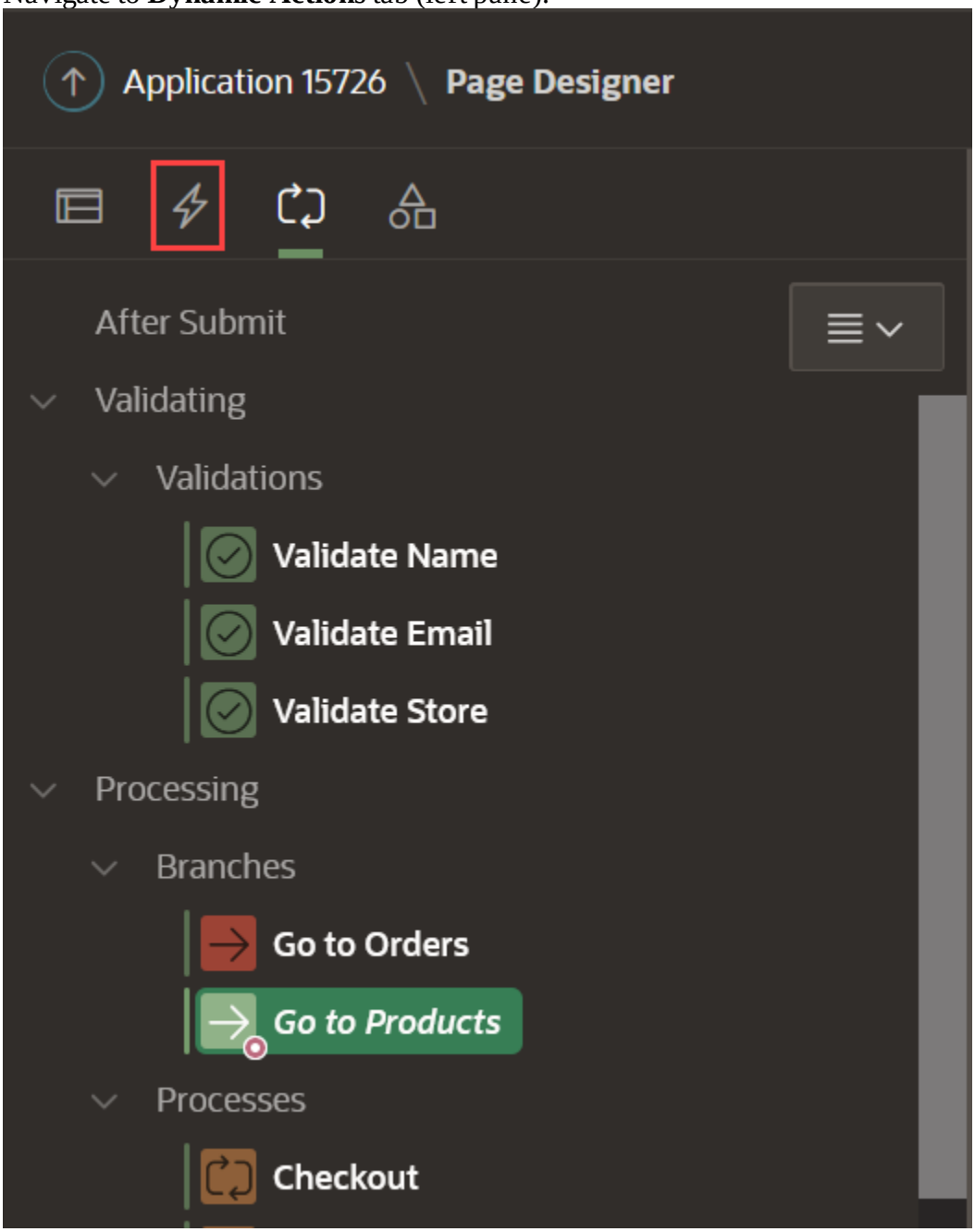
    - For Name - enter **Go to Products**
    - Navigate to Target attribute and click **No Link Defined**
        - For Type - select **Page in this application**
        - For Page - enter **1**
        - For Clear Cache - enter **1**
        - Click **OK**
    - For When Button Pressed, select **Clear**

Task 11: Add Dynamic Actions

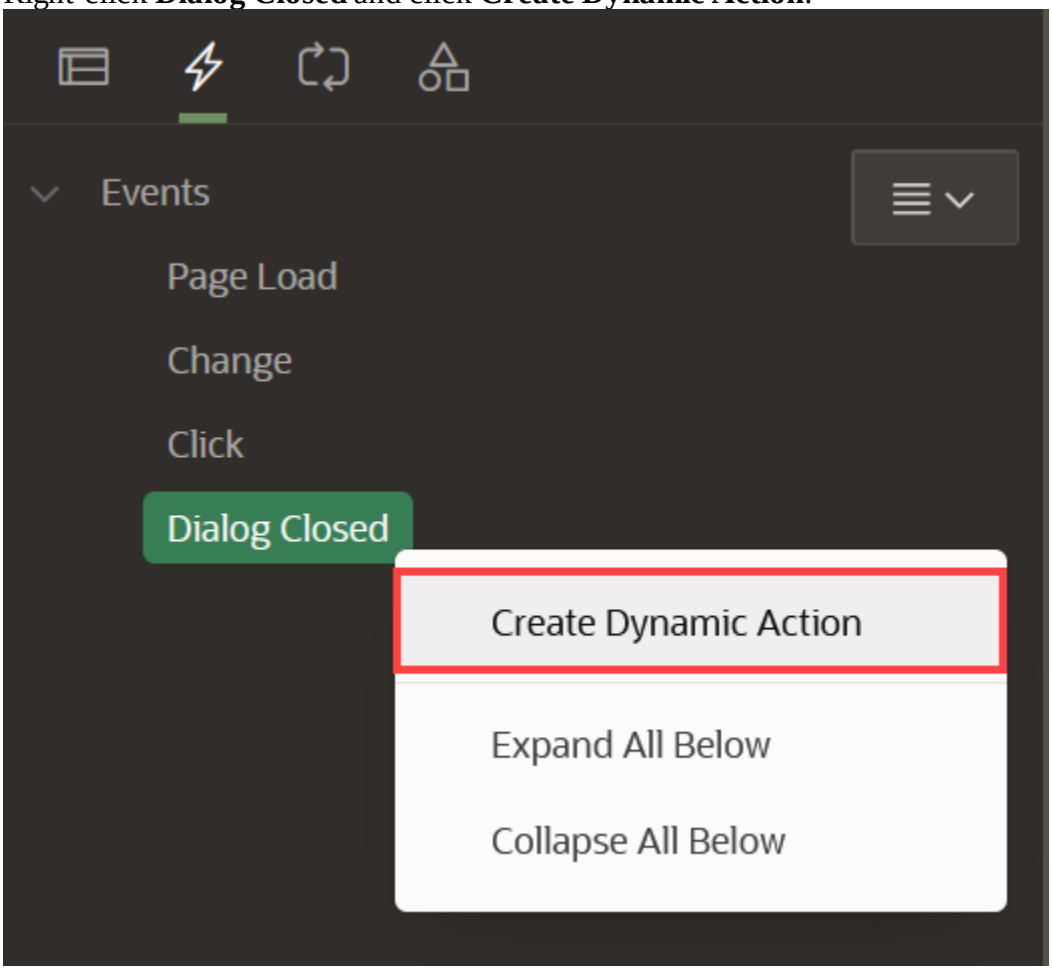
In this task, you will create a dynamic action to:

- Update the badge and icon shown in the navigation bar after the customer has added / edited / removed a product from the shopping cart
- Refresh the shopping cart region

1. Navigate to **Dynamic Actions** tab (left pane).

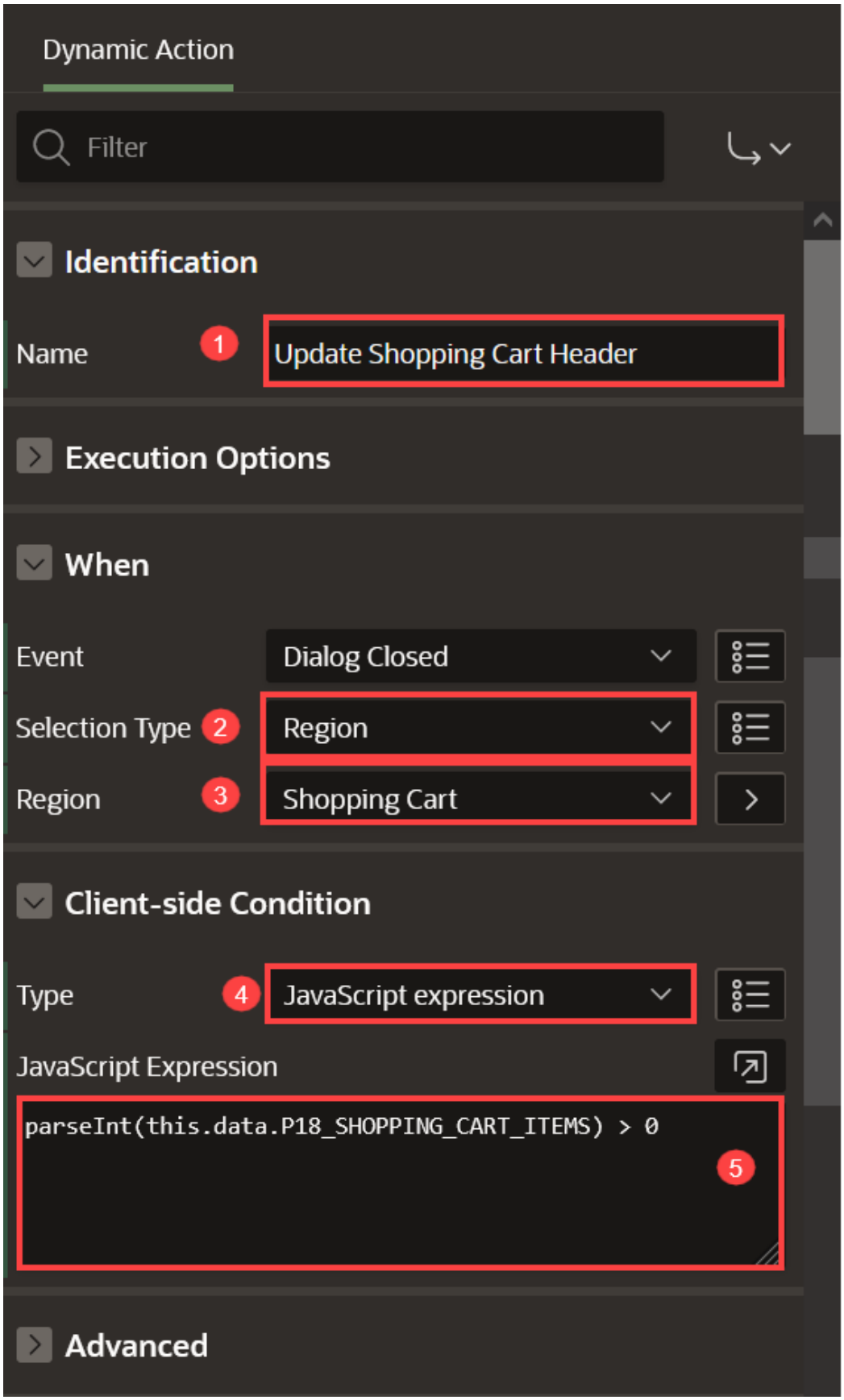


2. Right-click **Dialog Closed** and click **Create Dynamic Action**.



3. In the Property Editor, enter the following:
    - Under Identification section:
        - For Name - enter **Update Shopping Cart Header**
    - Under When section:
        - For Event - select **Dialog Closed**
        - For Selection Type - select **Region**
        - For Region - select **Shopping Cart**
    - Under Client-side Condition:
        - For Type - select **JavaScript expression**
        - For JavaScript Expression, enter the following:

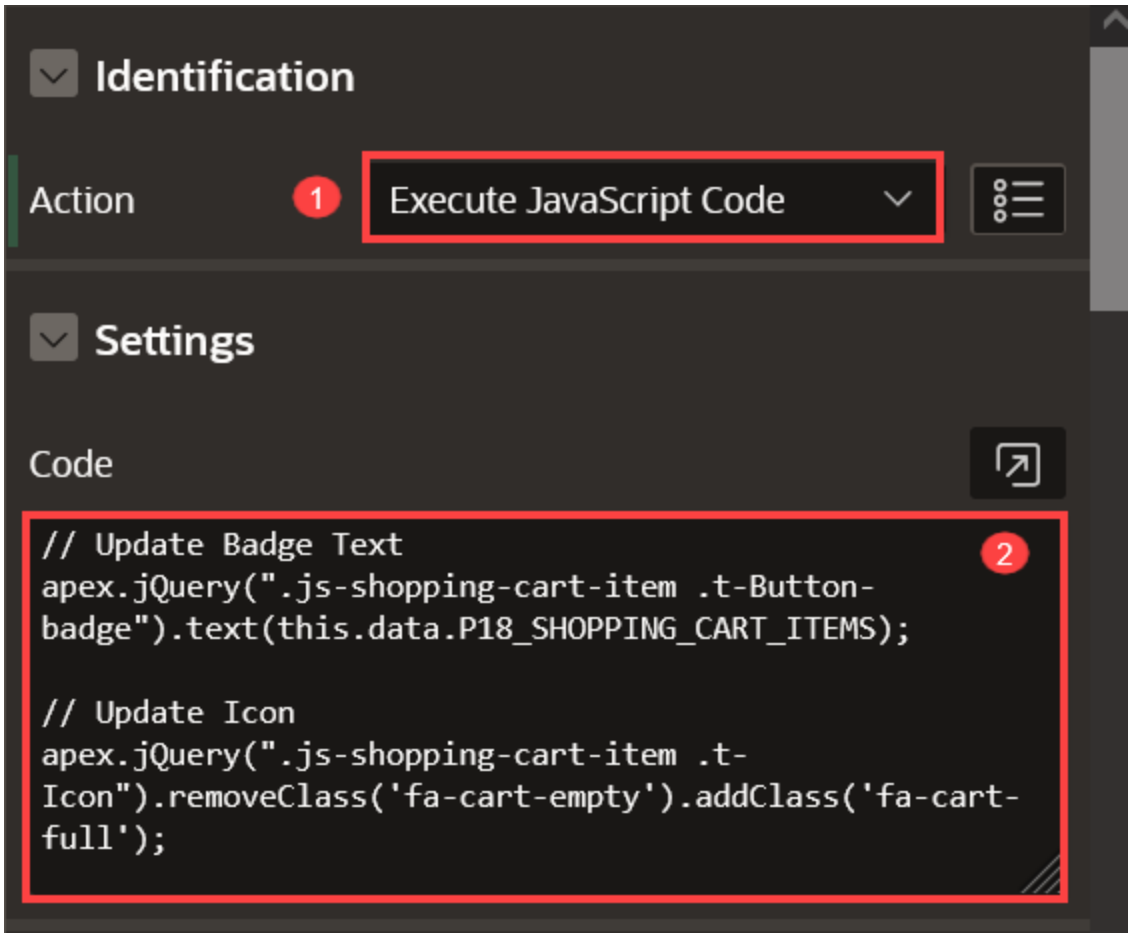
```
Copyparse Int(this.data.P18_SHOPPING_CART_ITEMS) >
0
```


Dynamic Action
Filter
Identification
Name
1
Update Shopping Cart Header
Execution Options
When
Event
Dialog Closed
Selection Type
2
Region
Region
3
Shopping Cart
Client-side Condition
Type
4
JavaScript expression
JavaScript Expression
parseInt(this.data.P18_SHOPPING_CART_ITEMS) > 0
5
Advanced

4. Navigate to **Refresh** Action.
    - Under Identification section:
        - For Action - select **Execute JavaScript Code**
    - Under Settings section:
        - For Code - enter the following JavaScript Code:

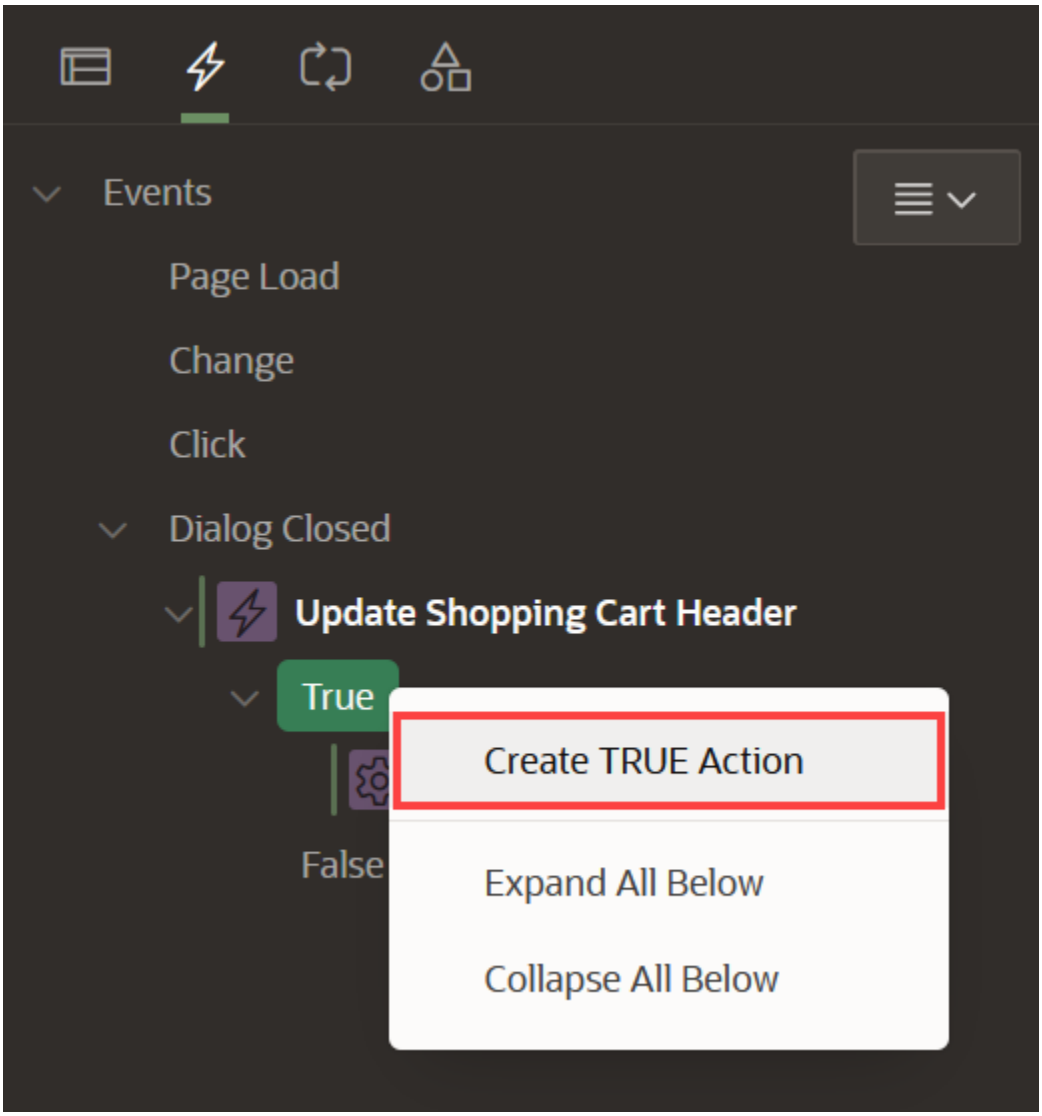
```
Copy// Update Badge Text

apex.jQuery(".js-shopping-cart-item .t-Button-badge").text(this.data.P18_SHOPPING_CART_ITEMS);


// Update Icon
```

```
apex.jQuery(".js-shopping-cart-item .t-Icon").removeClass('fa-cart-empty').addClass('fa-cart-full');
```



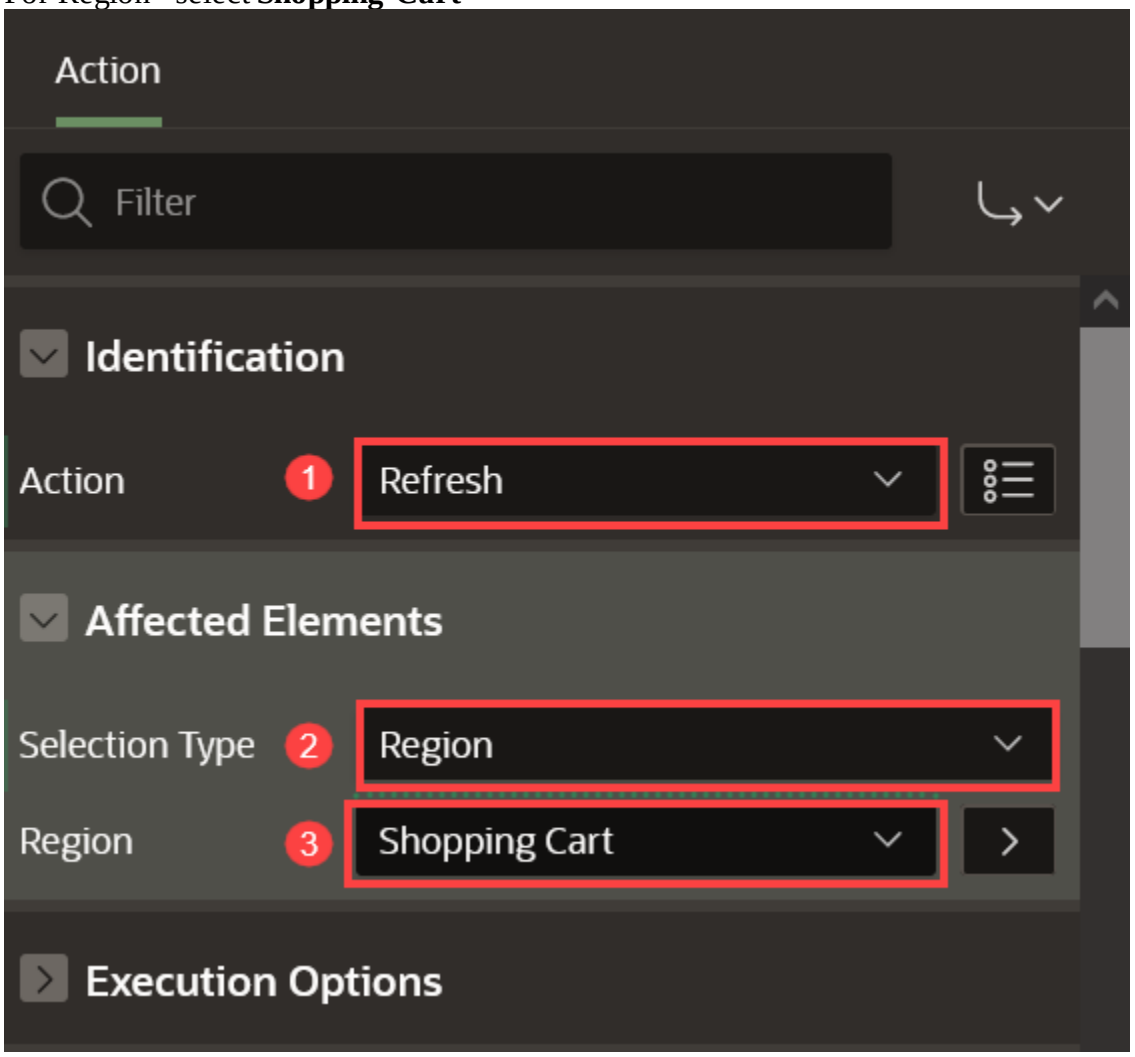
5. Create a second action. In the Dynamic Actions tab (left pane), navigate to **True** under **Update Shopping Cart Header** Dynamic Action. Right-click

on **True** and select **Create TRUE Action**.



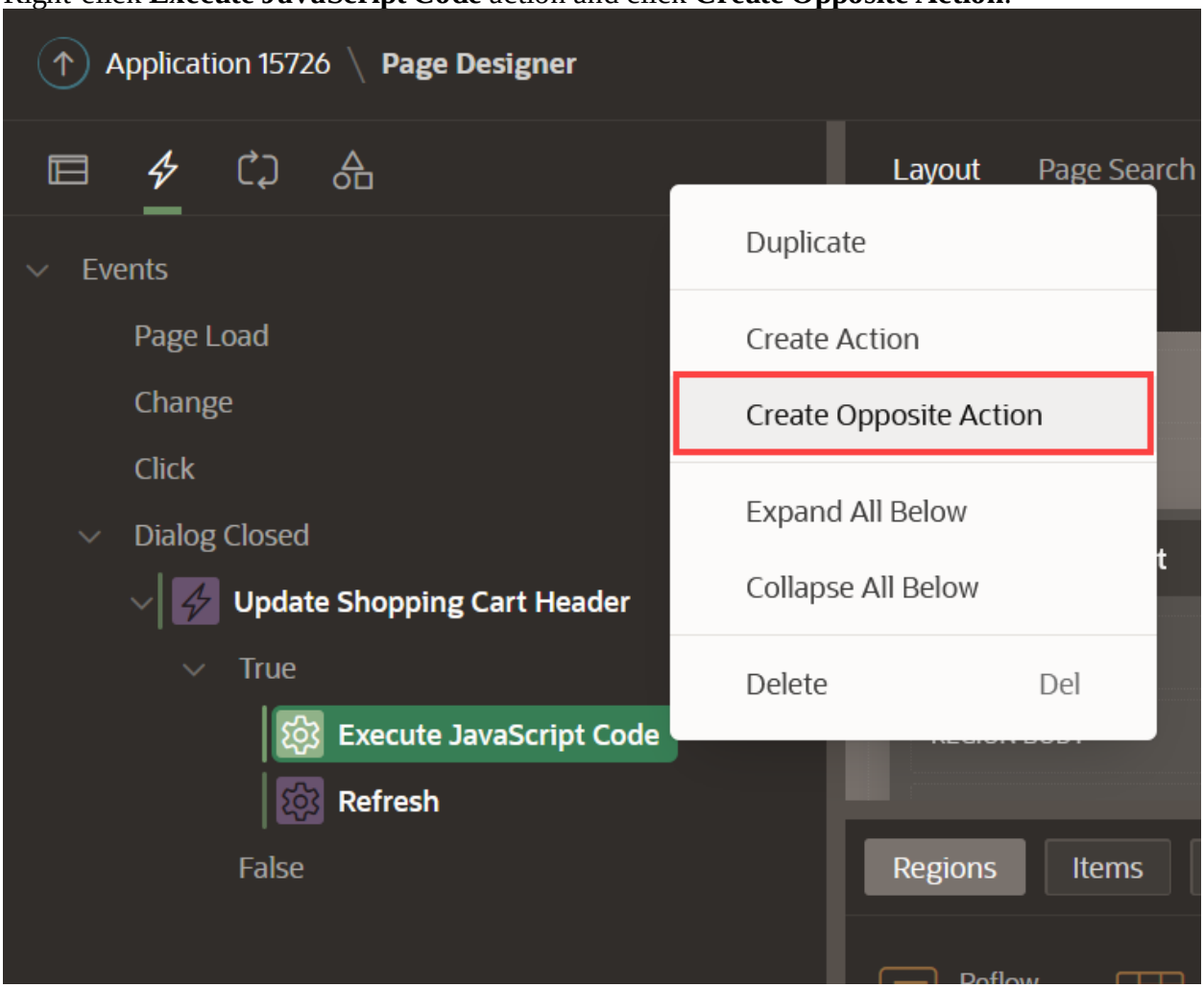
6. In the Property Editor, enter the following:
    - Under Identification section:
        - For Action - select **Refresh**
    - Under Affected Elements section:
        - For Selection Type - select **Region**

- For Region - select **Shopping Cart**



7. Create an opposite action. In the Dynamic Actions tab (left pane), navigate to **Execute JavaScript Code** action.

8. Right-click **Execute JavaScript Code** action and click **Create Opposite Action**.



9. Navigate to **Execute JavaScript Code** Action.
    - Under Identification section:
        - For Action - select **Execute JavaScript Code**
    - Under Settings section:
        - For Code - enter the following JavaScript Code:

```
Copy// Update Badge Text

apex.jQuery(".js-shopping-cart-item .t-Button-badge").text('');

```

```
// Update Icon
```

```
apex.jQuery(".js-shopping-cart-item .t-Icon").removeClass('fa-cart-full').addClass('fa-cart-empty');
```

10. Create a second action. In the Dynamic Actions tab (left pane), navigate to **False** under **Update Shopping Cart Header** Dynamic Action. Right-click on **False** and select **Create FLASE Action**.

11. In the Property Editor, enter the following:

    - Under Identification section:
      - For Action - select **Refresh**
    - Under Affected Elements section:
      - For Selection Type - select **Region**
      - For Region - select **Shopping Cart**

Task 12: Format Products Image Size

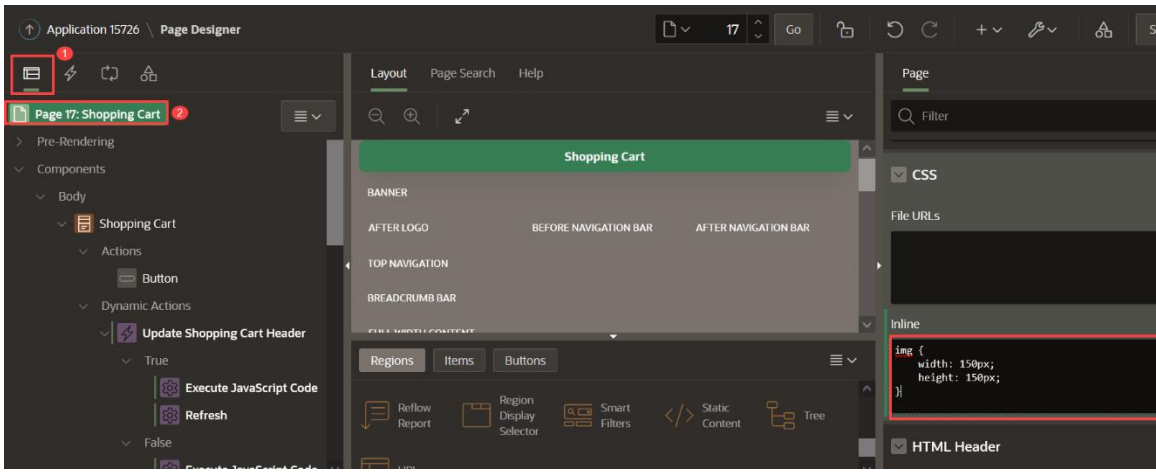
1. In the Rendering tree (left pane), navigate to **Page 17: Shopping Cart**.

2. In the Property Editor (right pane), do the following:

    - Under CSS section.
      - For Inline - enter the following:

```
Copyimg {
    width: 150px;
    height: 150px;
```

```
}
```



3. Click **Save**.

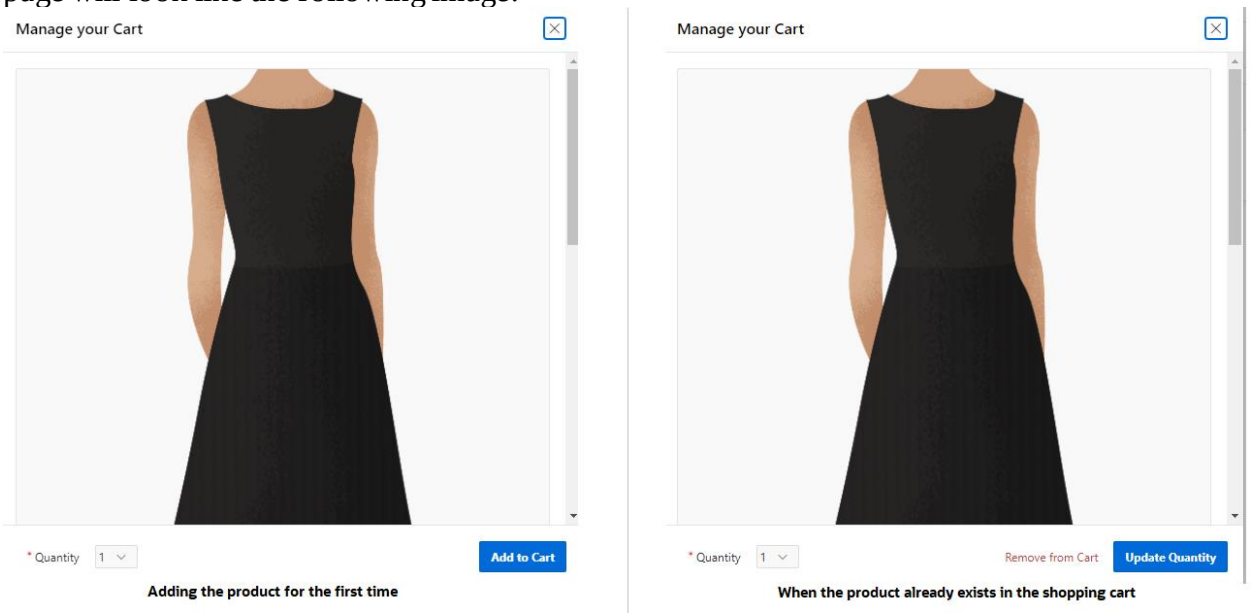
You now know how to add validations, processes, branches, and dynamic actions to your APEX page. You may now **proceed to the next lab**.

Create the add to cart page

Introduction

In this lab, you will create a new modal page to add or edit existing items in the Shopping Cart.

Once you have finished the workshop and updated all the products as described in the steps, your page will look like the following image:



Customers will be able to:

- Review the product details
- Add, edit, or remove the product from the shopping cart
- Read the customer reviews

Estimated Time: 20 minutes

Objectives

In this lab, you will:

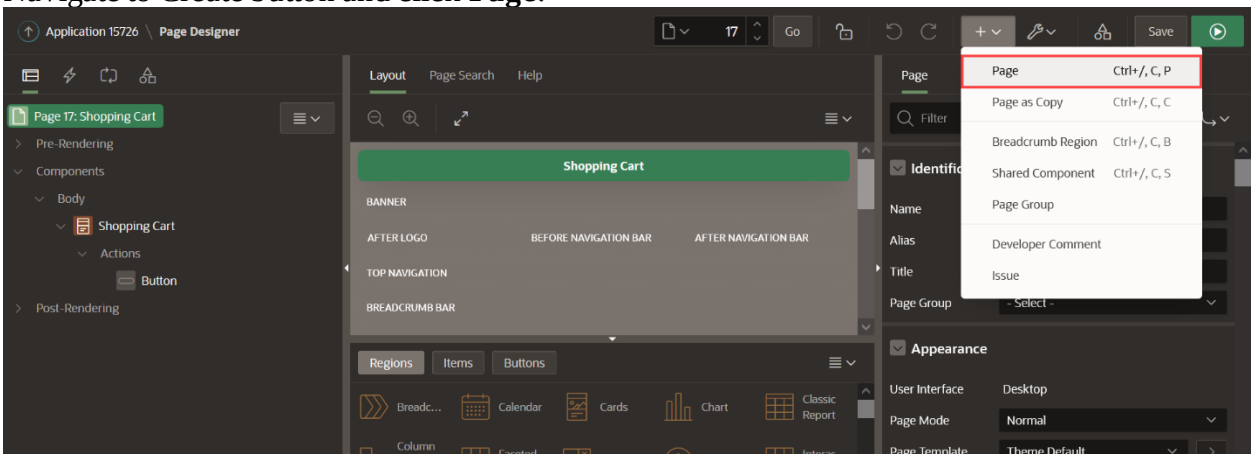
- Create a page that allows users to add and edit products in the Shopping Cart

Collapse All Tasks

Task 1: Create a Modal Page

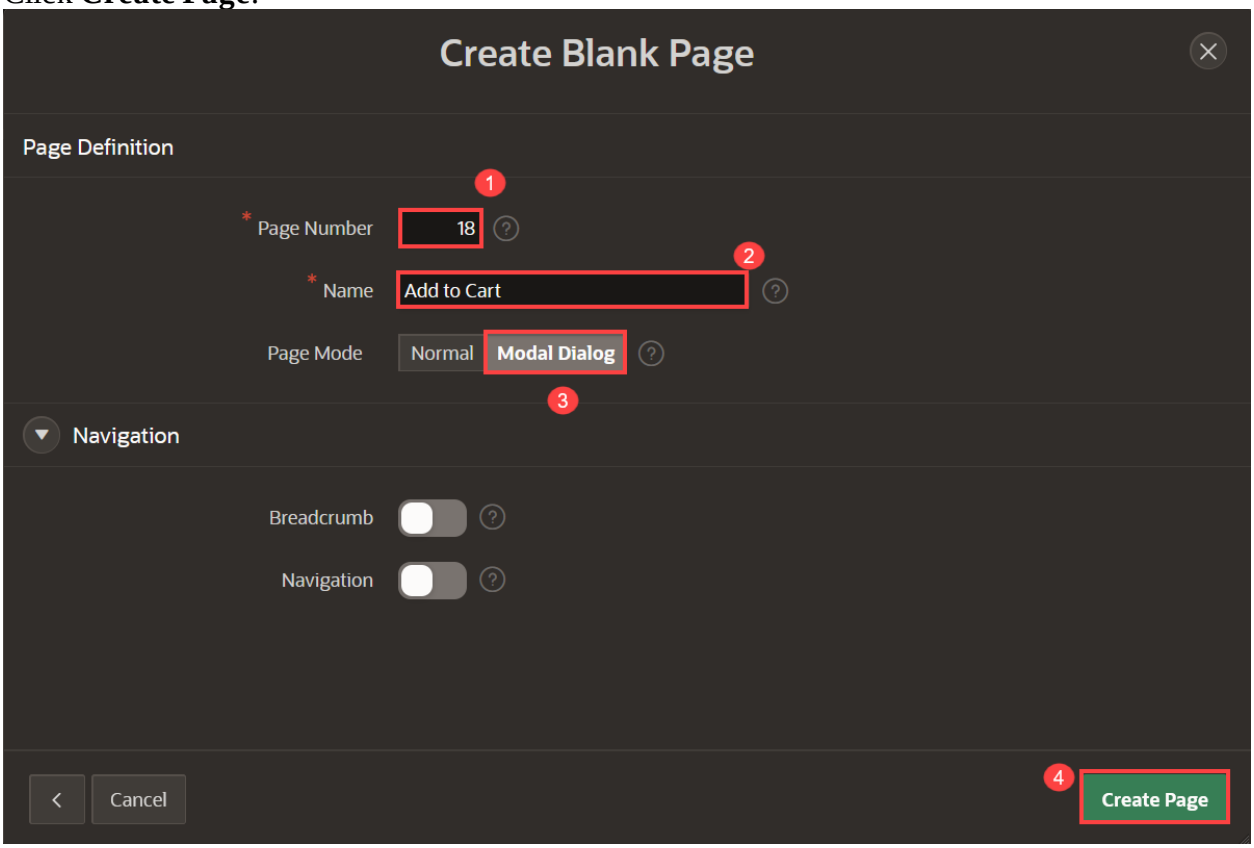
Create a Modal Page to add products to the cart.

1. Navigate to Create button and click **Page**.



2. Select **Blank Page** and click **Next**.
3. Enter the following and click **Next**.
    - Page Number - enter **18**
    - For Name - enter **Add to Cart**
    - For Page Mode - select **Modal Dialog**

4. Click **Create Page**.

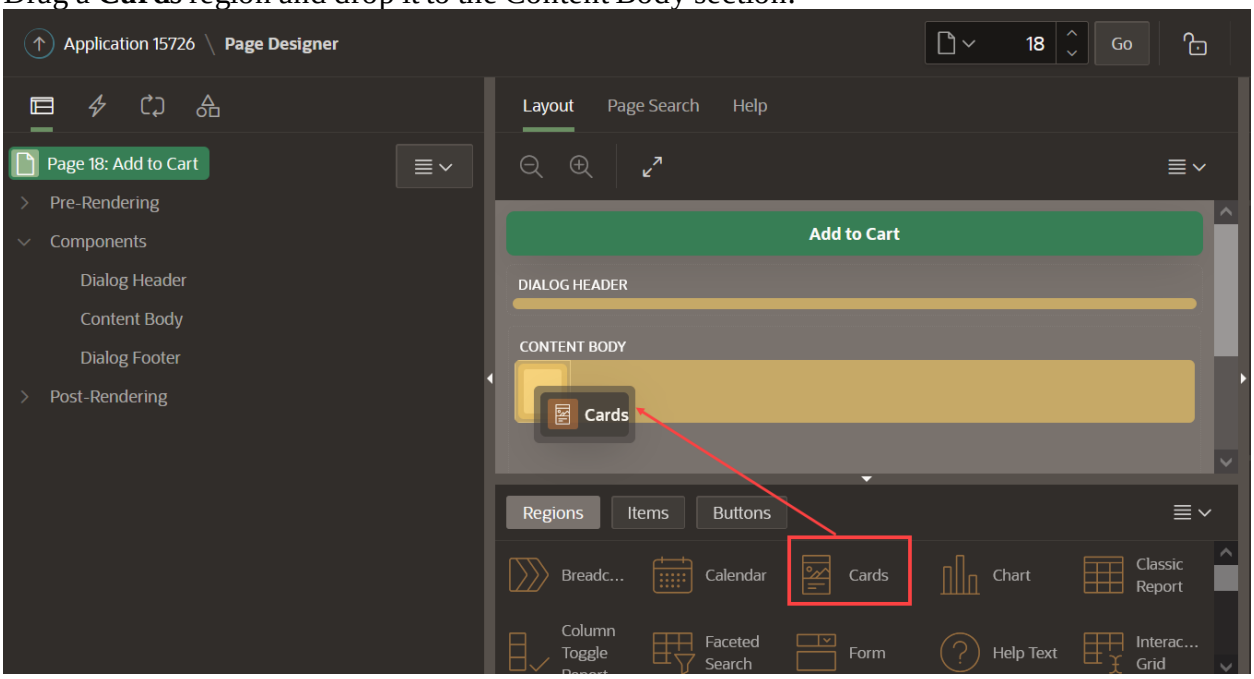


Task 2: Add Cards Region for Product Details

This region allows users to review the details of the product, such as brand, price, description, and more.

1. In the new modal page created, navigate to the **Gallery Menu**.

2. Drag a **Cards** region and drop it to the Content Body section.



3. In the Property Editor, enter the following:
    - For Title, enter **Product**
    - Under Source section:
        - For Type - select **SQL Query**
        - For SQL Query - enter the following SQL Code:

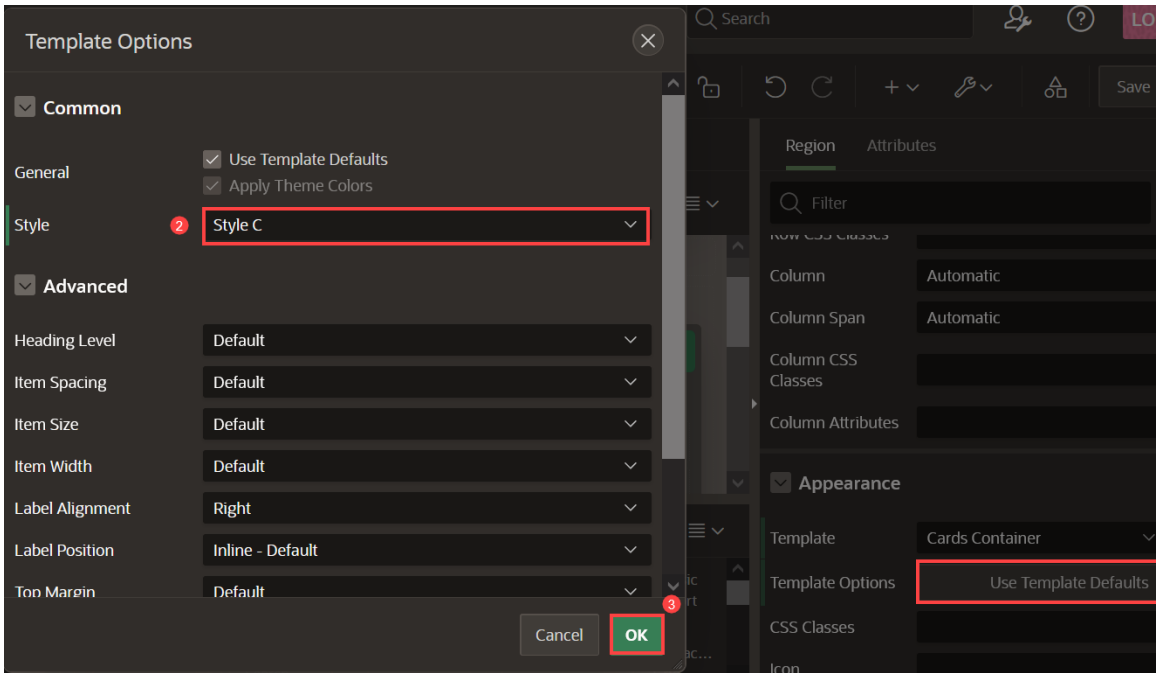
```
Copy SELECT product_id,
       product_name,
       unit_price,
       product_details,
       product_image,
       image_mime_type,
       image_filename,
```

```
    image_charset,
    image_last_updated,
    color_id,
    department_id,
    clothing_id,
    d.description,
    b.brand
FROM   products p,
    json_table (p.product_details, '$' columns (
description varchar2(4000) path '$.description')
) d,
    json_table (p.product_details, '$' columns (
brand       varchar2(4000) path '$.brand') ) b
```

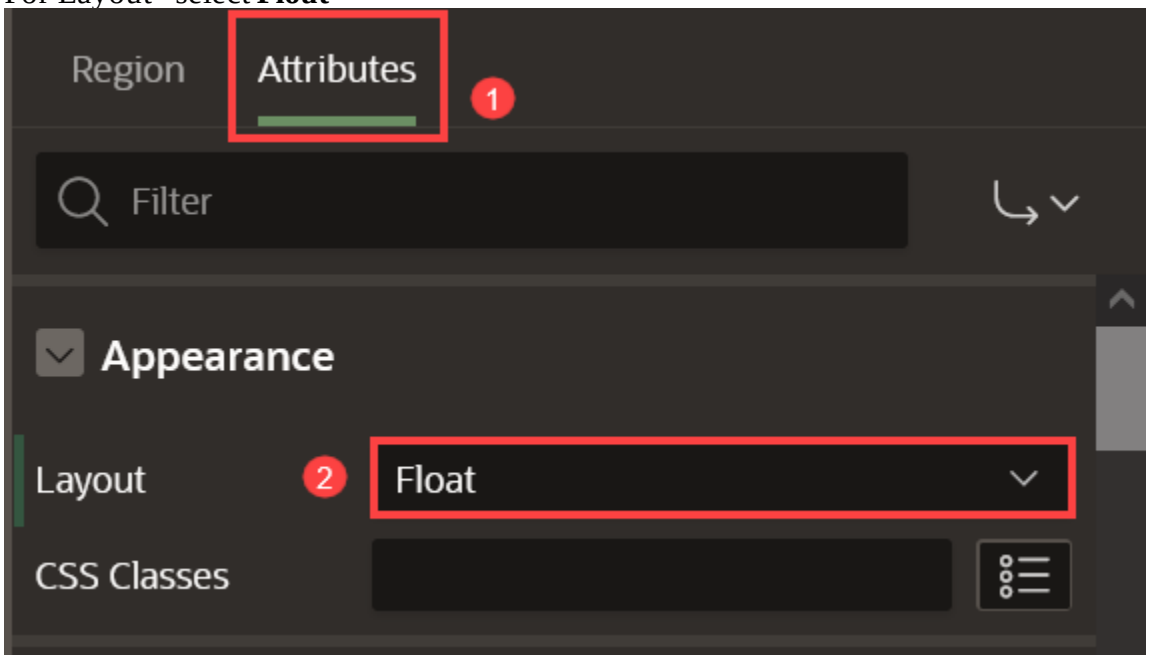
```
WHERE  product_id = :p18_product_id
```

- For Template Options - click **Use Template Defaults**
  - For **Style** - select **Style C**

- Click **Ok**



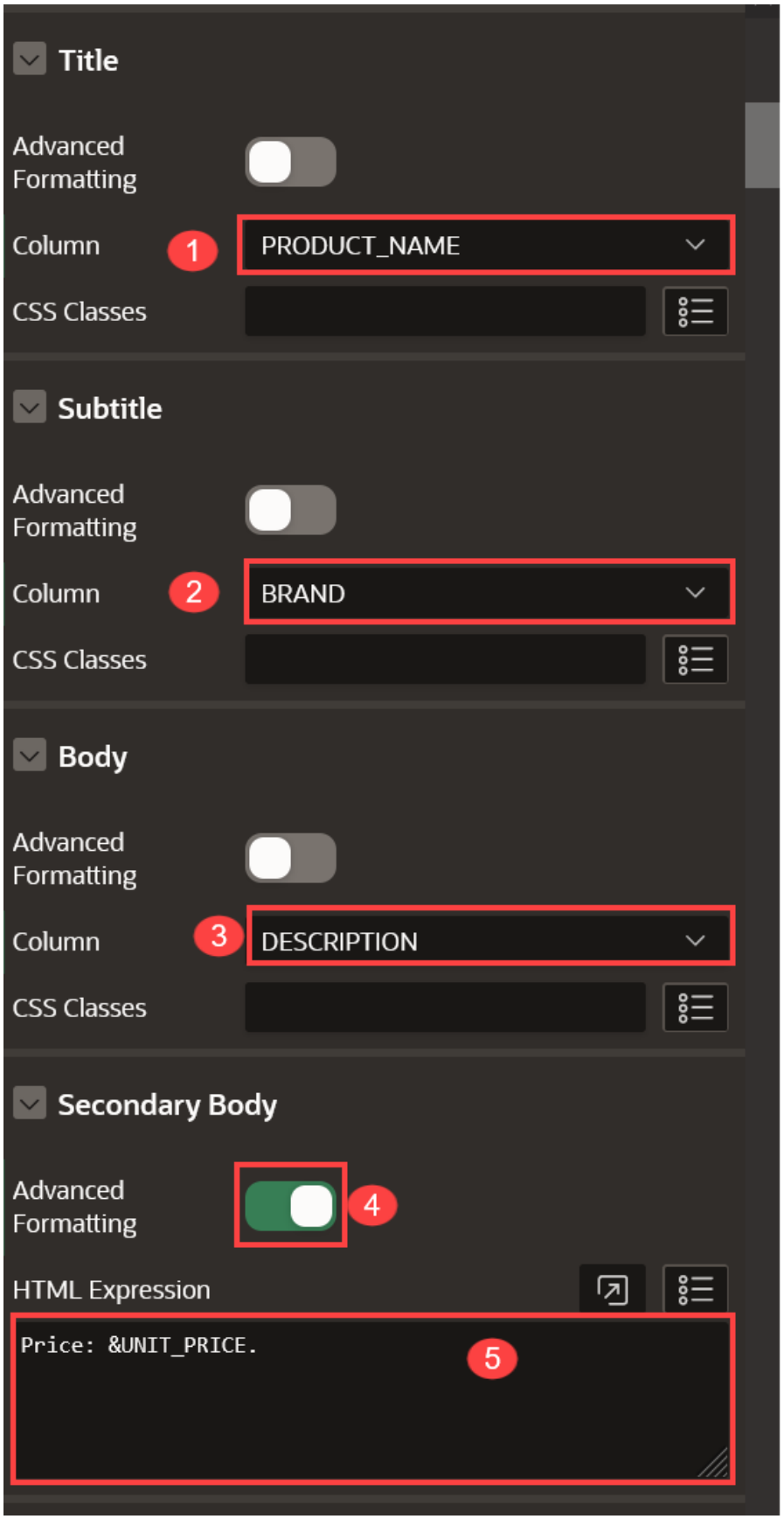
4. Click **Attributes** and enter the following:
    - Under Appearance section:
        - For Layout - select **Float**



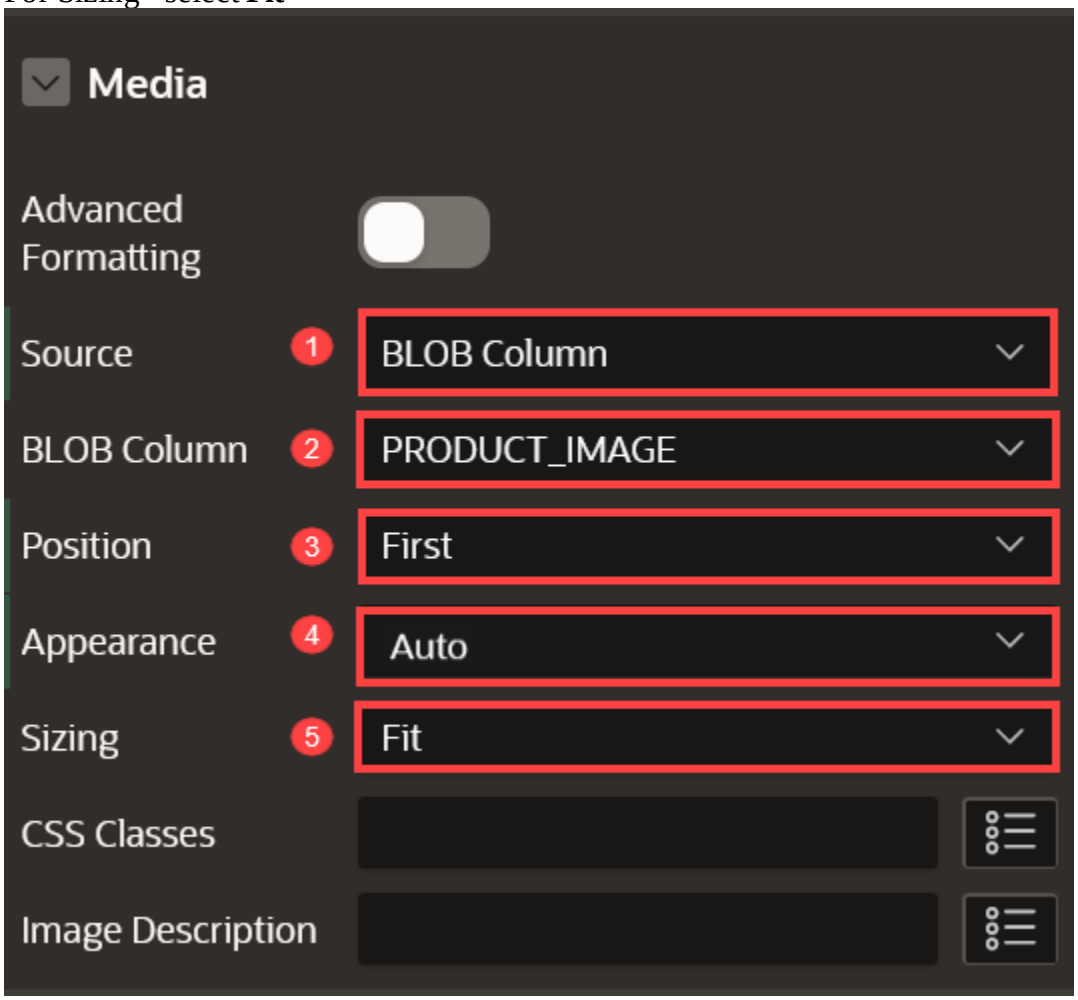
    - Under Title section:
        - For Column - select **PRODUCT_NAME**
    - Under Subtitle section:

        - For Column - select **BRAND**
    - Under Body section:
        - For Column - select **DESCRIPTION**
    - Under Secondary Body section:
        - Set Advanced Formatting to **On**.
        - For HTML Expression - enter the following:

```
CopyPrice: &UNIT_PRICE.
```

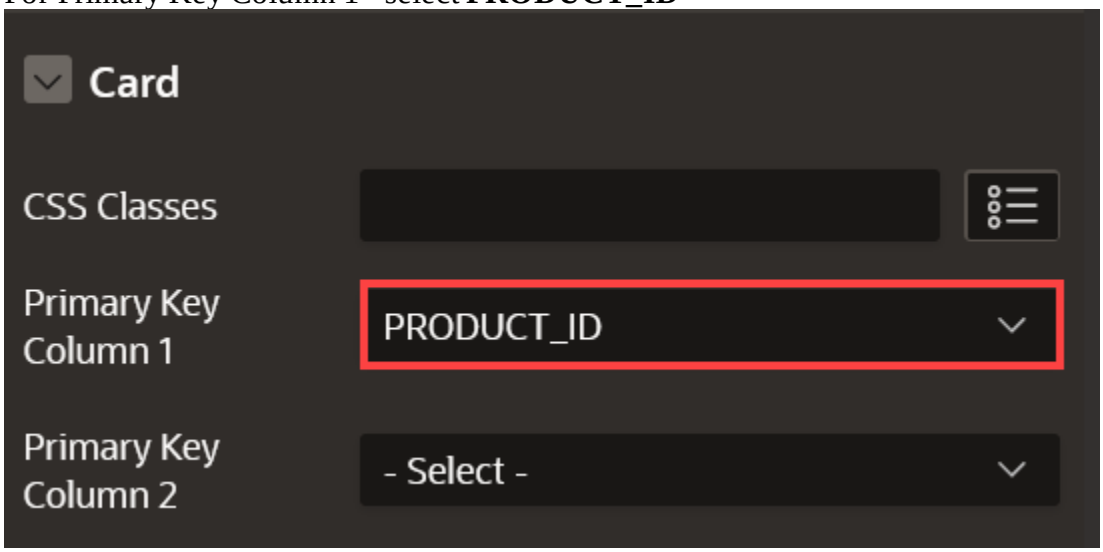

Title
Advanced Formatting
Column
1
PRODUCT_NAME
CSS Classes
Subtitle
Advanced Formatting
Column
2
BRAND
CSS Classes
Body
Advanced Formatting
Column
3
DESCRIPTION
CSS Classes
Secondary Body
Advanced Formatting
4
HTML Expression
Price: &UNIT_PRICE.
5

- Under Media section:
    - For Source - select **BLOB Column**
    - For BLOB Column - select **PRODUCT_IMAGE**
    - For Position - select **First**
    - For Appearance - select **Auto**
    - For Sizing - select **Fit**



- Under Card section:

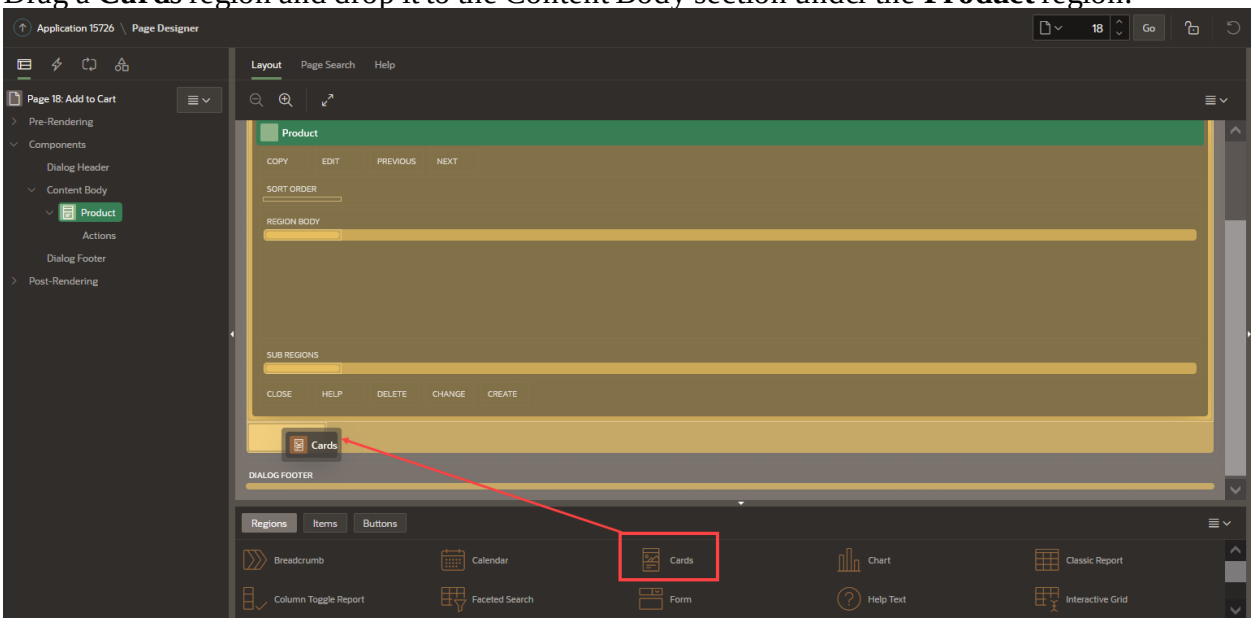
- For Primary Key Column 1 - select **PRODUCT_ID**



Task 3: Add Cards Region for Customer Reviews

This region lets users read the customer reviews for a product.

1. Navigate to the **Gallery Menu**.
2. Drag a **Cards** region and drop it to the Content Body section under the **Product** region.



3. In the Property Editor, enter the following:
    - For Title - enter **Customer Reviews**
    - Under Source section:

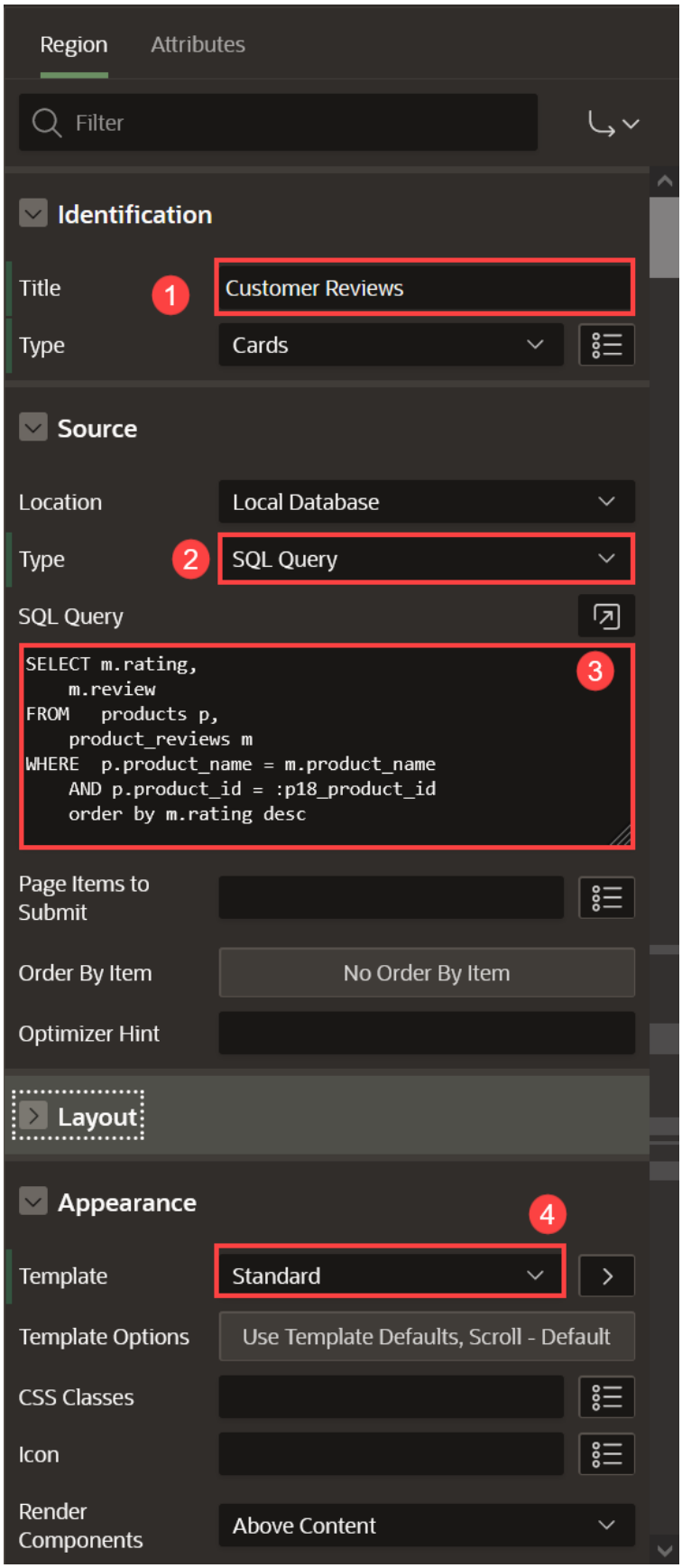
- For **Type** - select **SQL Query**
- For **SQL Query** - enter the following SQL Code:

```
Copy
SELECT m.rating,
    m.review
FROM   products p,
    product_reviews m
WHERE  p.product_name = m.product_name
    AND p.product_id = :p18_product_id
```

```
    order by m.rating desc
```

- Under Appearance section:

- For Template - select **Standard**


Region
Attributes
Filter
Identification
Title
1
Customer Reviews
Type
Cards
Source
Location
Local Database
Type
2
SQL Query
SQL Query
SELECT m.rating,
m.review
FROM products p,
product_reviews m
WHERE p.product_name = m.product_name
AND p.product_id = :p18_product_id
order by m.rating desc
3
Page Items to Submit
Order By Item
No Order By Item
Optimizer Hint
Layout
Appearance
4
Template
Standard
Template Options
Use Template Defaults, Scroll - Default
CSS Classes
Icon
Render Components
Above Content

4. Click **Attributes** and enter the following:

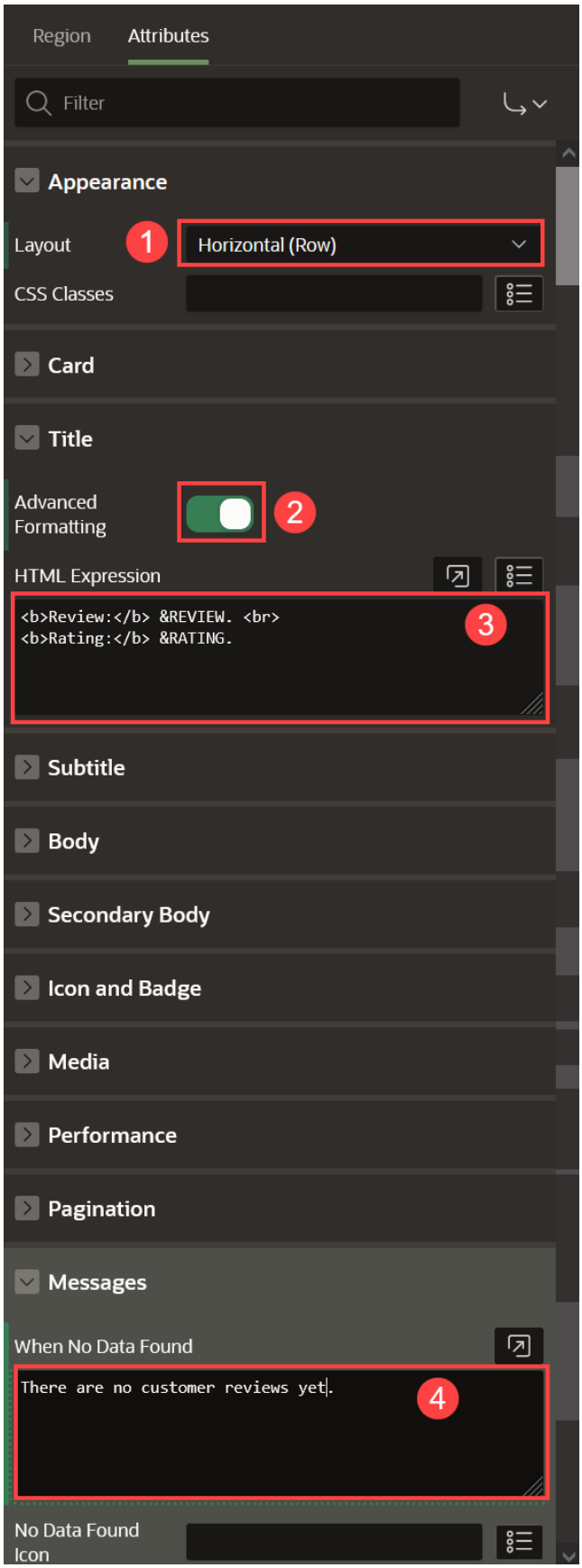
    - Under Appearance section:
        - For Layout - select **Horizontal (Row)**
    - Under Title section:
        - Set Advanced Formatting to **On**
        - For HTML Expression - enter the following:

```
Copy<b>Review:</b> &REVIEW. <br>
```

```
<b>Rating:</b> &RATING.
```

    - Under Messages:

- For When No Data Found - enter **There are no customer reviews yet.**


Region
Attributes
Filter
Appearance
Layout
1
Horizontal (Row)
CSS Classes
Card
Title
Advanced Formatting
2
HTML Expression
Review: &REVIEW.
Rating: &RATING.
3
Subtitle
Body
Secondary Body
Icon and Badge
Media
Performance
Pagination
Messages
When No Data Found
There are no customer reviews yet.
4
No Data Found Icon

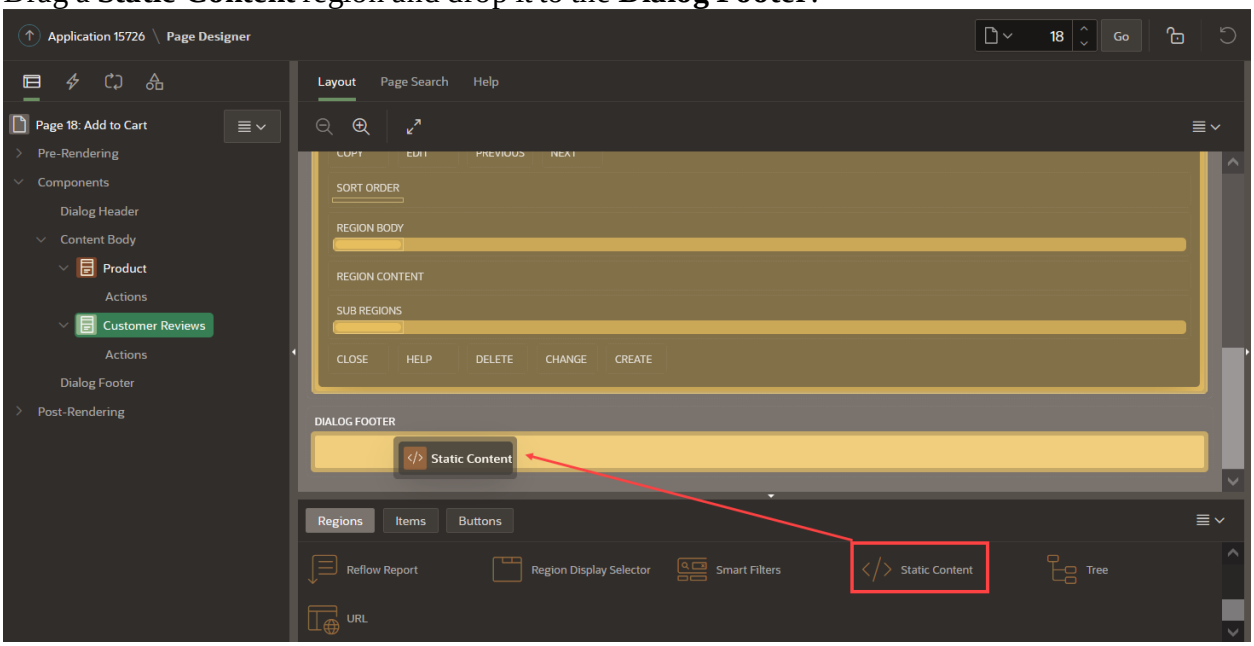
Task 4: Add Items and Buttons

In this task, you will create four-page items:

- **PRODUCT_ID**: To get the product ID
- **ACTION**: To identify the action (Add / Edit / Delete) made for the customer
- **QUANTITY**: To permit customers to select the number of items to add or edit in the shopping cart
- **SHOPPING_CART_ITEMS**: To get the number of items (total) in the shopping cart after an action is made

1. Navigate to the **Gallery Menu**.

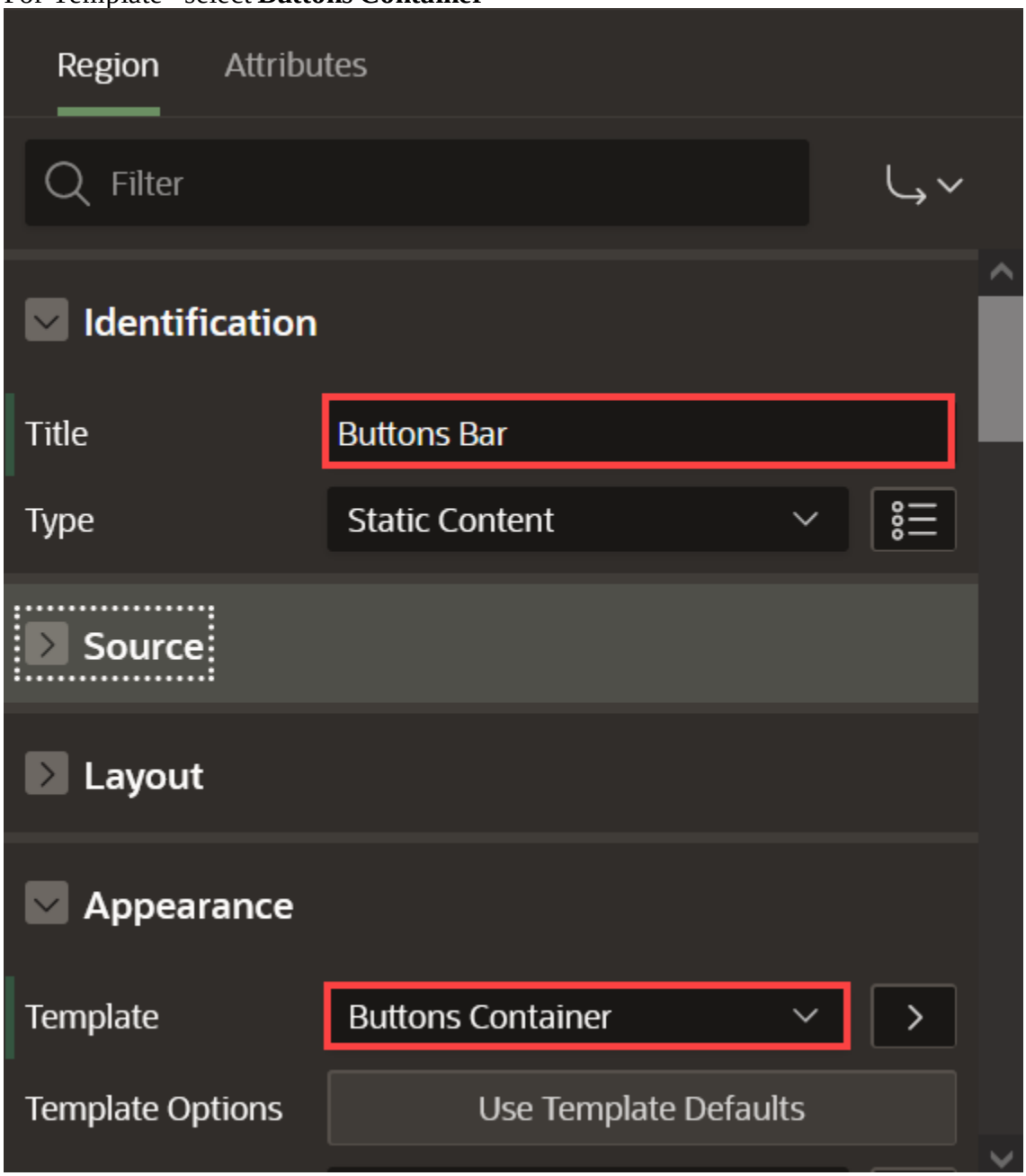
2. Drag a **Static Content** region and drop it to the **Dialog Footer**.



3. In the Property Editor, enter the following:

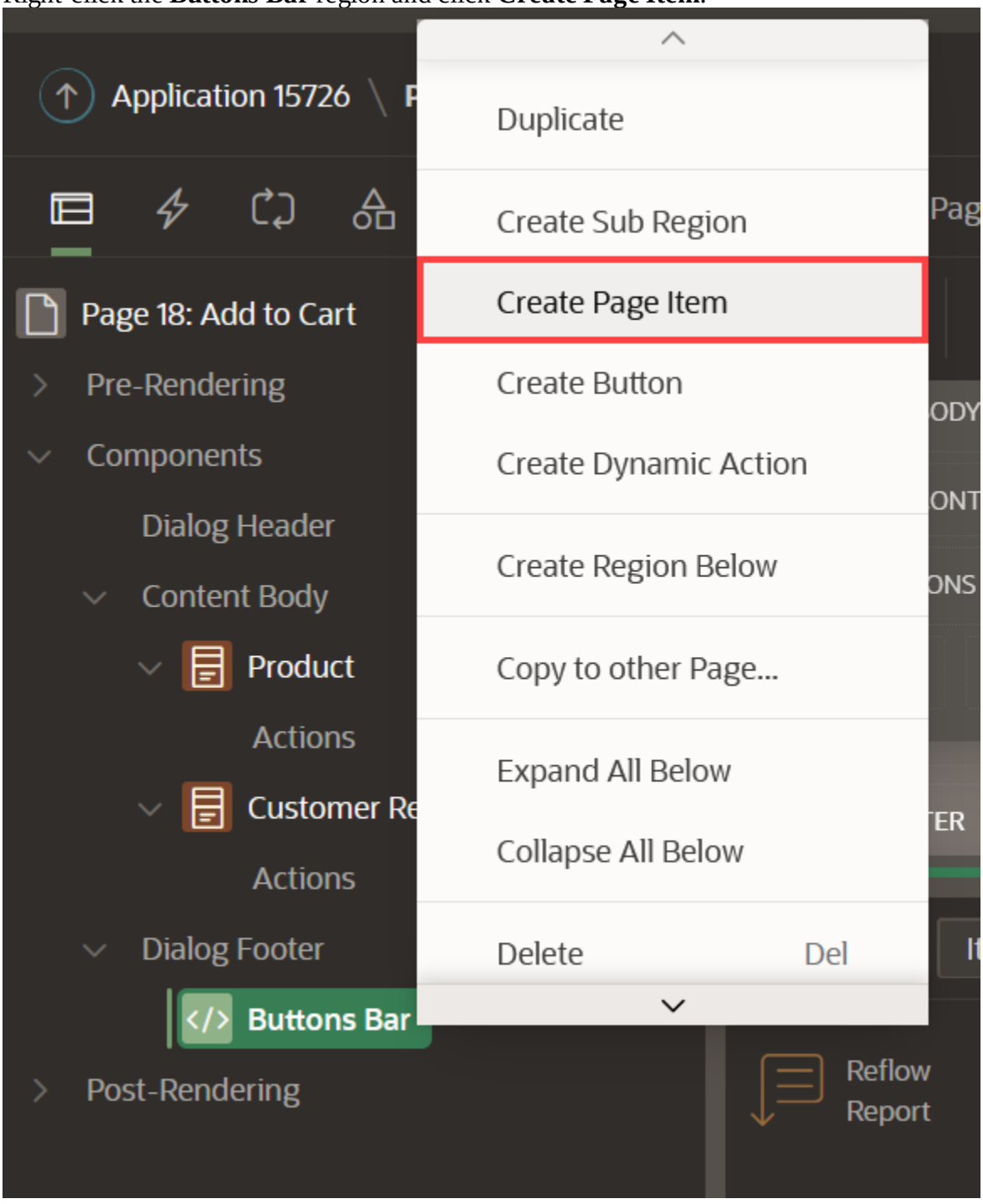
    - For Title - enter **Buttons Bar**

- For Template - select **Buttons Container**



4. In the Rendering tree (left pane), navigate to **Buttons Bar** region.

5. Right-click the **Buttons Bar** region and click **Create Page Item**.



6. Create four items as follows. In the Property Editor, do the following:

Table 1: Details of the Page Items

Name	Type	Label	Template
P18_ACTION	Hidden		
P18_PRODUCT_ID	Hidden		
P18_SHOPPING_CART_ITEMS	Hidden		
P18_QUANTITY	Select List	Quantity	Required

7. For **P18_QUANTITY** item, do the following:
    - Under List of Values section:
        - For Type - select **Static Values**
        - For Static Values - click **Display1, Display2** and enter the following:

Table 2: List of Static Values

Display Value	Return Value
1	1
2	2
3	3

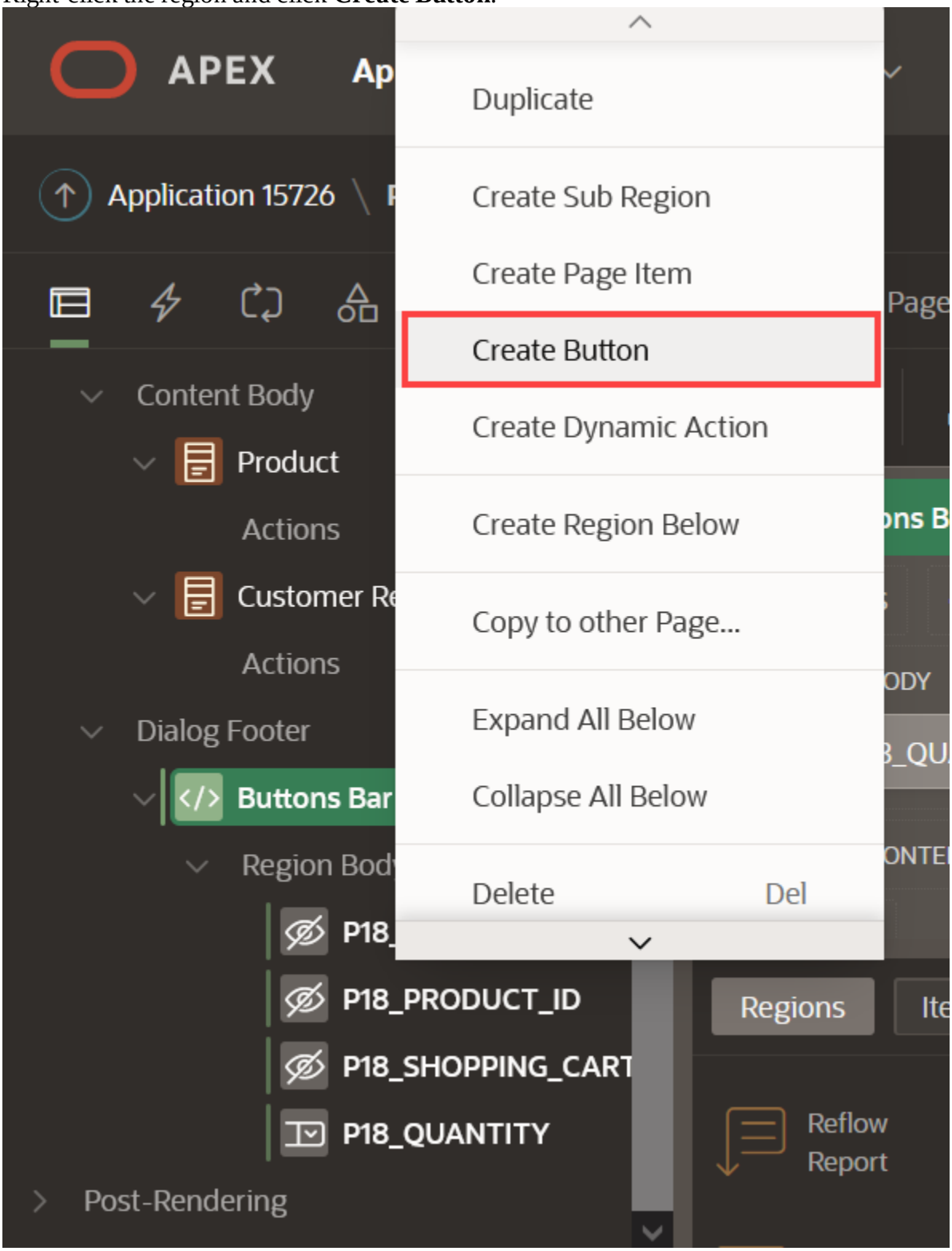
Table 2: List of Static Values

Display Value	Return Value
4	4
5	5

- Click **Ok**
- Set Display Extra Values to **Off**
- Set Display Null Value to **Off**

8. Navigate to **Buttons Bar** region (left side).

9. Right-click the region and click **Create Button**.



10. Create three buttons as follows:

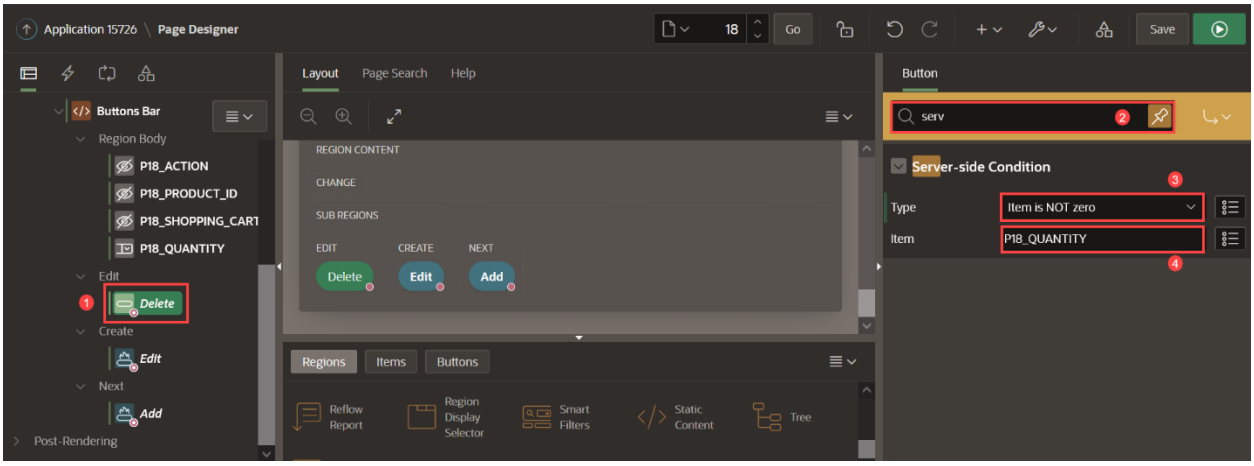
Table 3: Details of the Buttons

Name	Label	Button Position	Button Template	Hot
Add	Add to Cart	Next	Text	On
Edit	Update Quantity	Create	Text	On
Delete	Remove from Cart	Edit	Text	Off

11. Under Server-side Condition section:

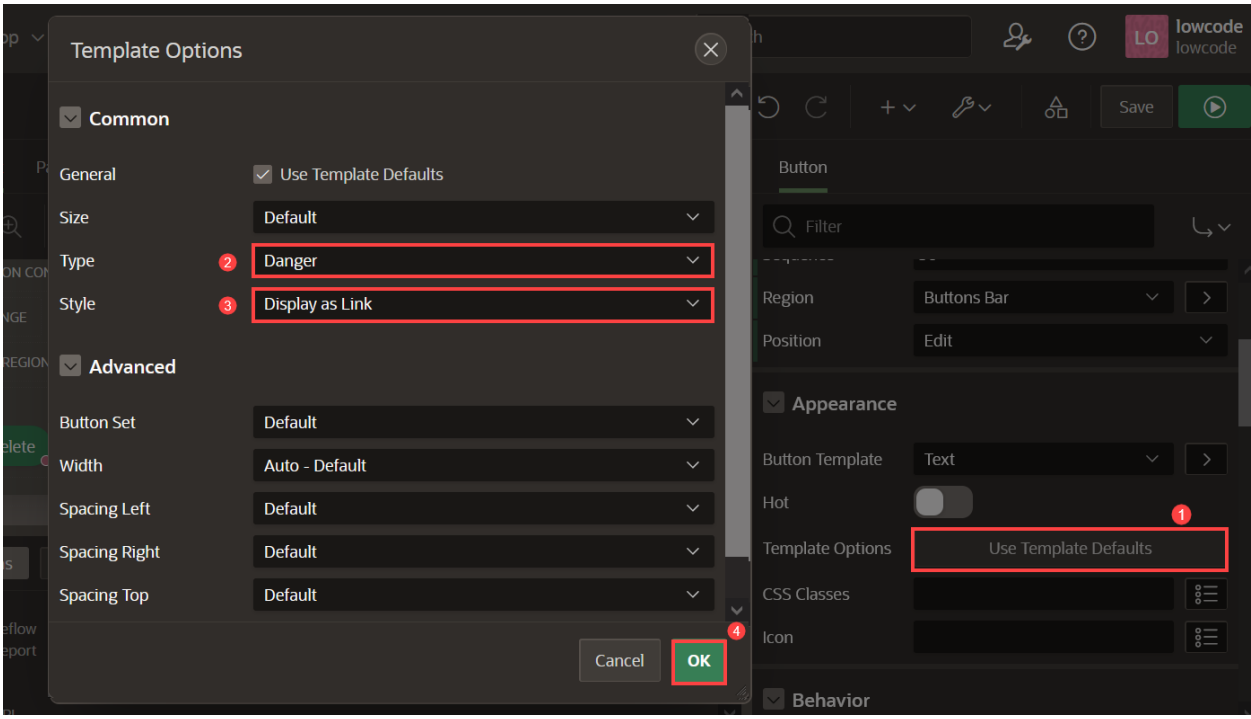
Table 4: Server-side conditions for the Buttons

Name	Type	Item
Add	Item is zero	P18_QUANTITY
Edit	Item is NOT zero	P18_QUANTITY
Delete	Item is NOT zero	P18_QUANTITY

12. 

13. For **Delete** button, apply the following changes:

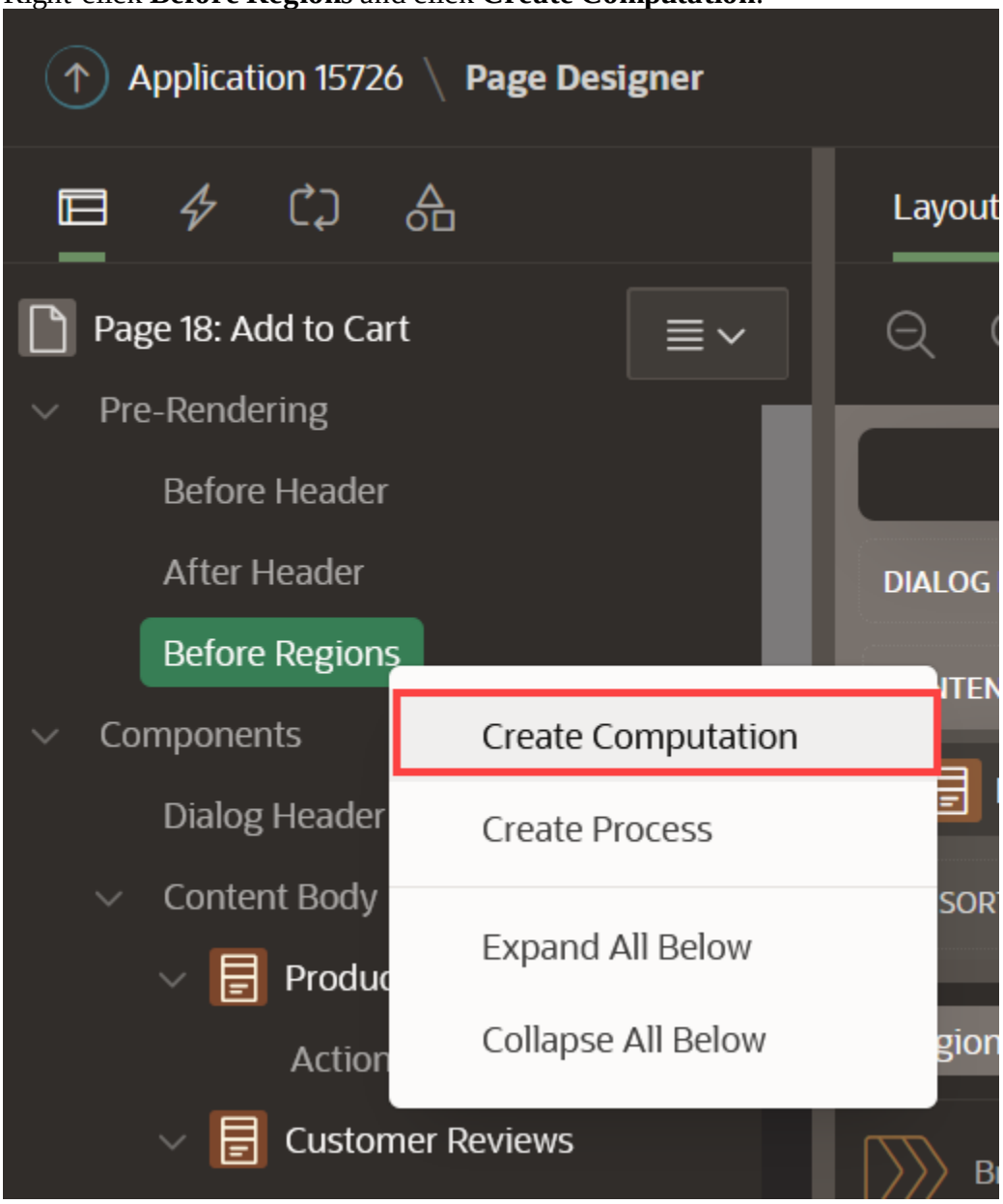
    - Under Appearance section, click **Use Template Defaults**:
        - For Type - select **Danger**
        - For Style -select **Display as Link**
        - For Spacing Right, select **Large**
    - Click **Ok**.



Task 5: Add Computation to Calculate the Number of Items for a Product

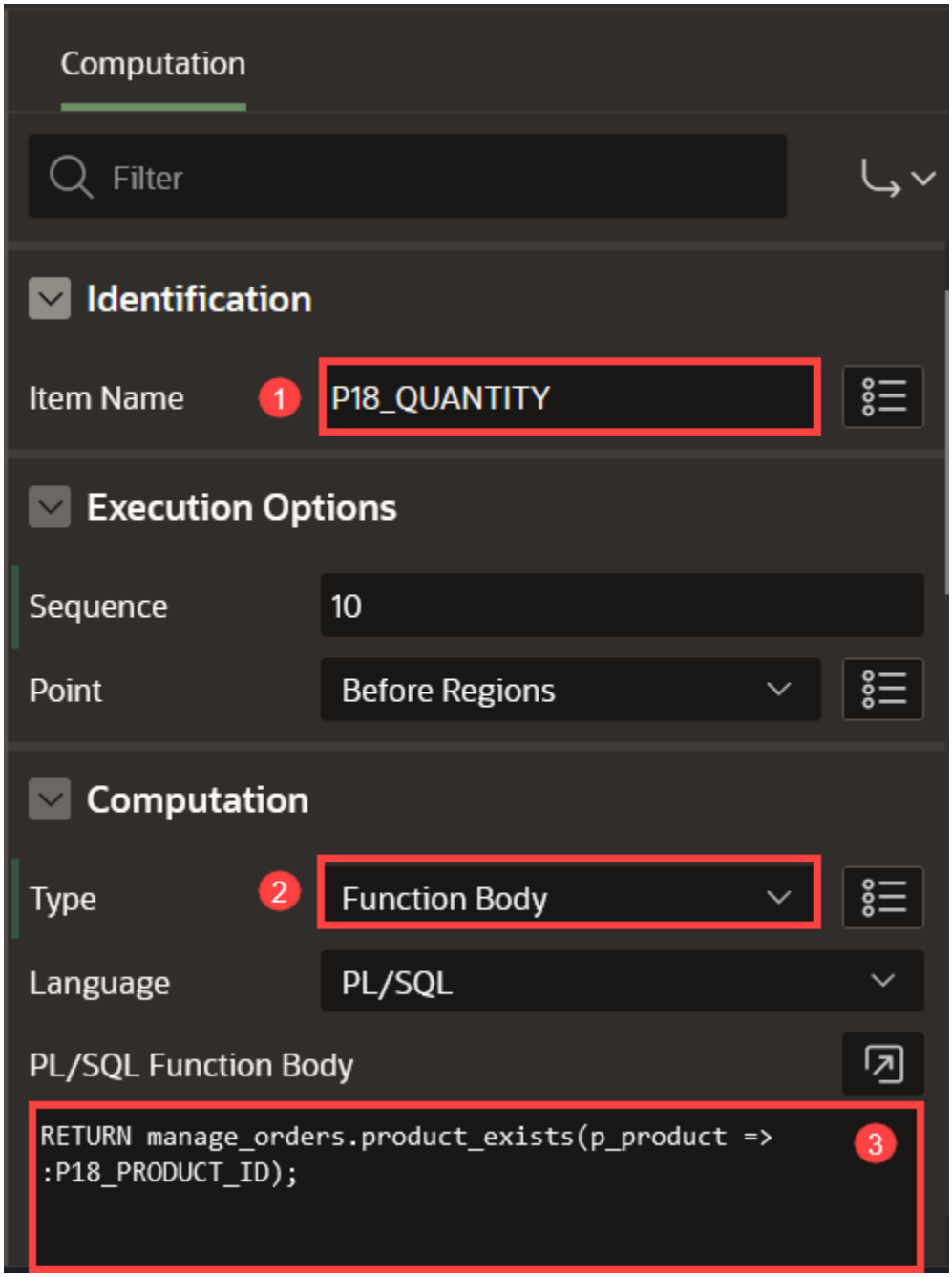
1. In the Rendering tree (left pane), expand the **Pre-Rendering**.

2. Right-click **Before Regions** and click **Create Computation**.



3. In the Property Editor, enter the following:
    - Under Identification section:
        - For Item Name - select **P18_QUANTITY**
    - Under Computation:
        - For Type - select **Function Body**
        - For PL/SQL Function Body - enter the following PL/SQL Code:

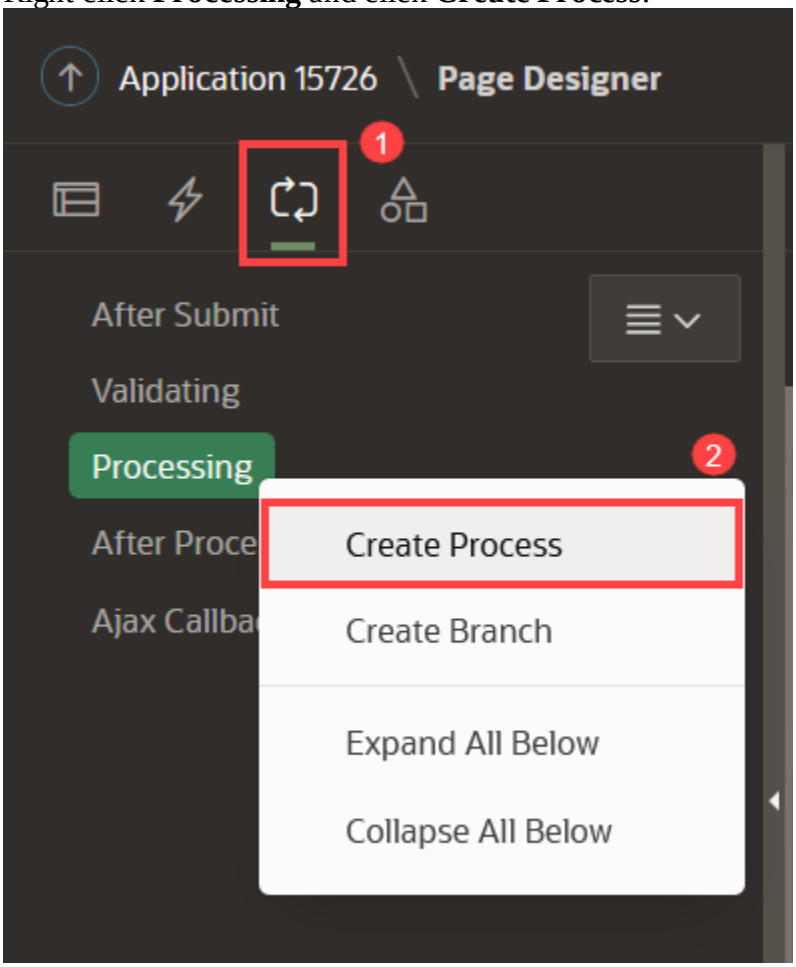
```
CopyRETURN manage_orders.product_exists(p_product =>
:P18_PRODUCT_ID);
```



Task 6: Add Process to Add Products to the Shopping Cart

In this Task, you will call the *manage_orders.add_product* procedure that will add the product temporarily in the APEX collection.

1. In the Rendering tree (left pane), navigate to **Processing** tab.
2. Right click **Processing** and click **Create Process**.



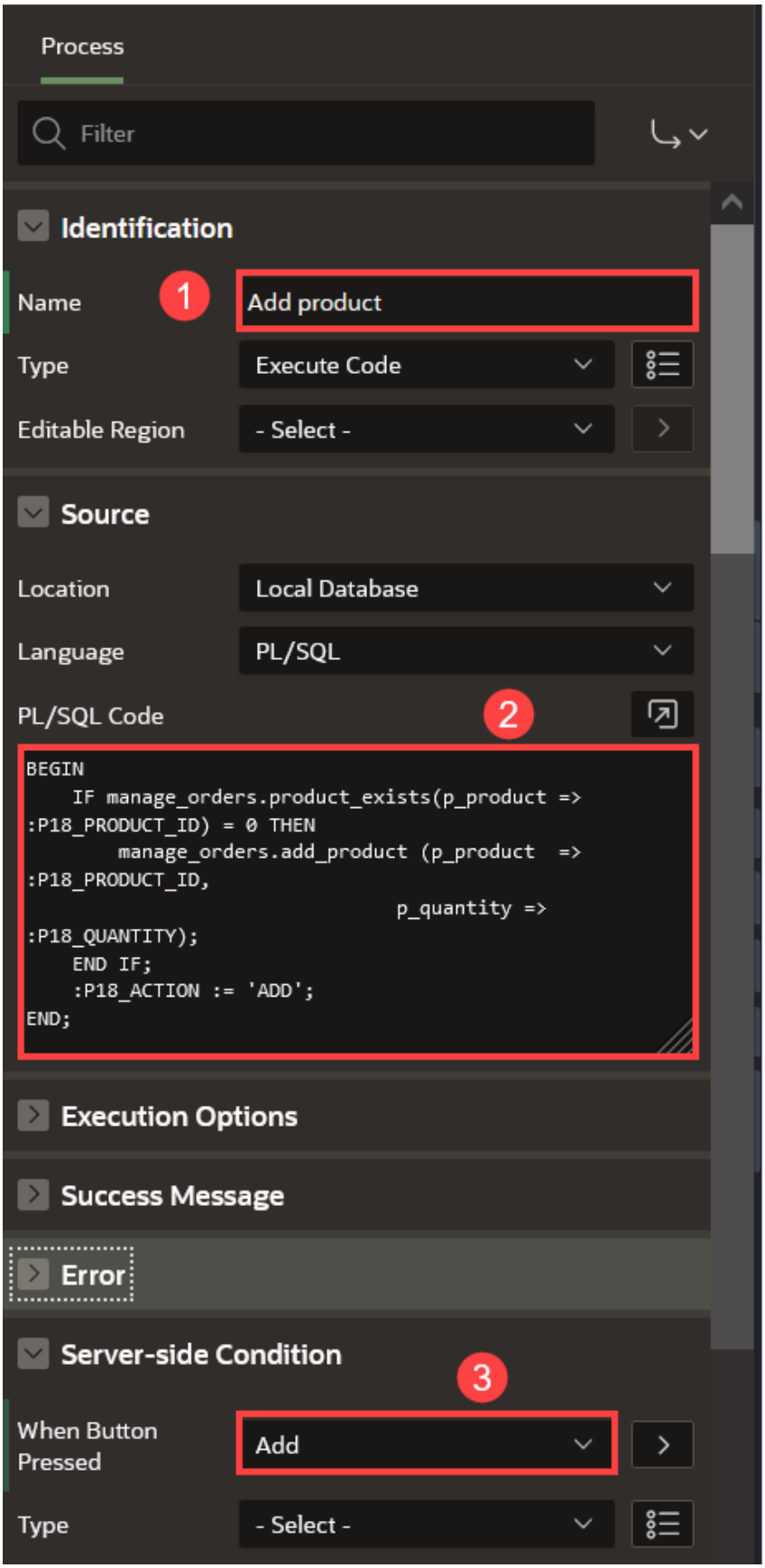
3. In the Property Editor, enter the following:
    - For Name - enter **Add product**
    - For Type - select **Execute Code**
    - For PL/SQL Code - enter the following code:

```
Copy
BEGIN

    IF manage_orders.product_exists(p_product =>
:P18_PRODUCT_ID) = 0 THEN

        manage_orders.add_product (p_product  =>
:P18_PRODUCT_ID,
```

```
                                        p_quantity =>
:P18_QUANTITY);

    END IF;

    :P18_ACTION := 'ADD';
```

```
END;
```

- Under Server-side Condition section:
  - For When Button Pressed - select **Add**


Process
Filter
Identification
Name
1
Add product
Type
Execute Code
Editable Region
- Select -
Source
Location
Local Database
Language
PL/SQL
PL/SQL Code
2
BEGIN
IF manage_orders.product_exists(p_product =>
:P18_PRODUCT_ID) = 0 THEN
manage_orders.add_product (p_product =>
:P18_PRODUCT_ID,
p_quantity =>
:P18_QUANTITY);
END IF;
:P18_ACTION := 'ADD';
END;
Execution Options
Success Message
Error
Server-side Condition
3
When Button Pressed
Add
Type
- Select -

Task 7: Add Process to Edit Products in the Shopping Cart

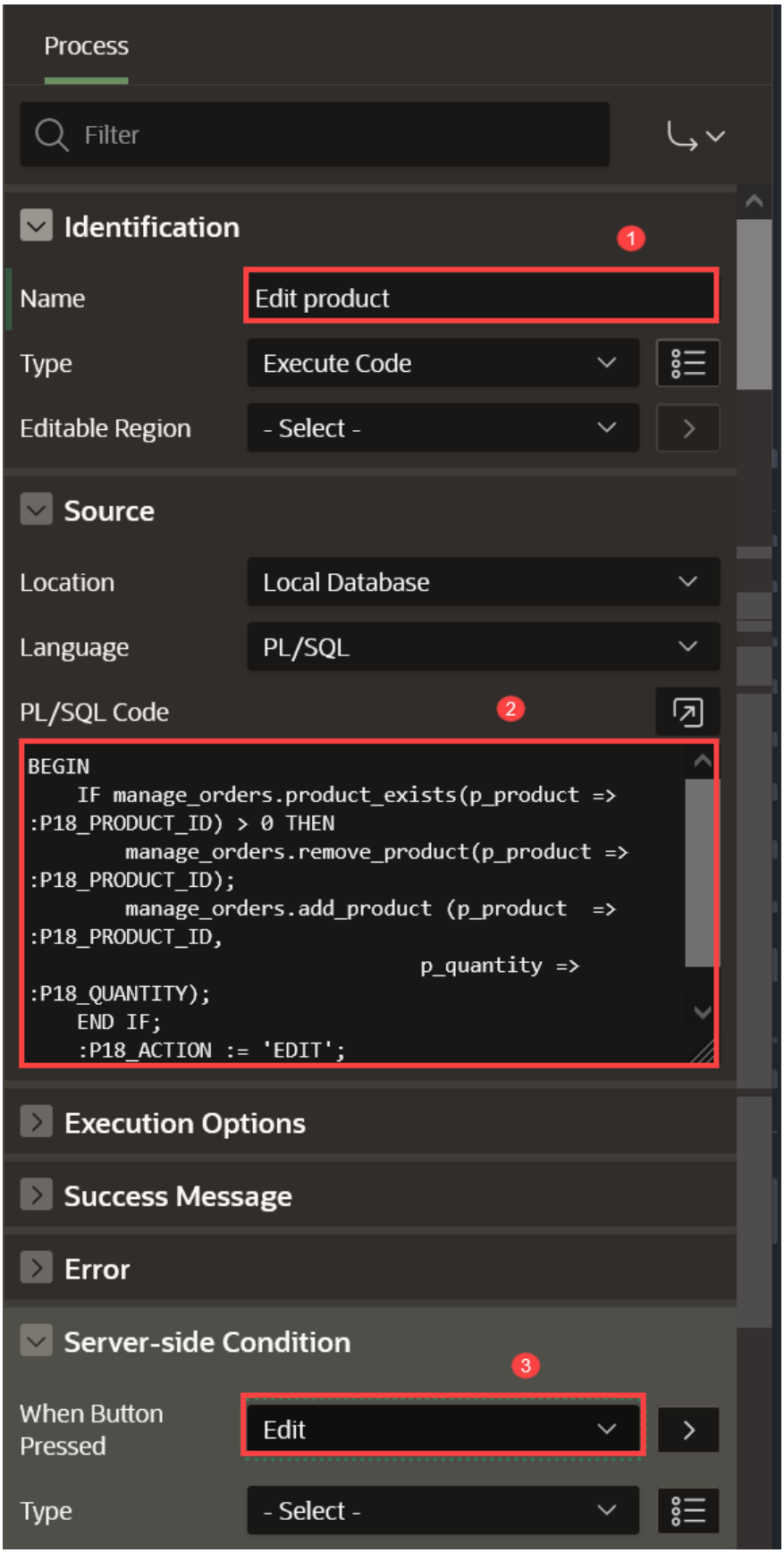
In this Task, you will call the *manage_orders.remove_product* and *manage_orders.add_product* procedures to remove the product from the shopping cart and add it again with the updated quantity.

1. In the **Processing** tab.
2. Right click **Processing** and click **Create Process**.
3. In the Property Editor, enter the following:
    - For Name - enter **Edit product**
    - For Type - select **Execute Code**
    - For PL/SQL Code - enter the following PL/SQL code:

```
Copy
BEGIN
    IF manage_orders.product_exists(p_product => :P18_PRODUCT_ID) > 0 THEN
        manage_orders.remove_product(p_product => :P18_PRODUCT_ID);
        manage_orders.add_product (p_product  => :P18_PRODUCT_ID,
                                   p_quantity => :P18_QUANTITY);
    END IF;
    :P18_ACTION := 'EDIT';
END;
```

    - Under Server-side Condition section:

- For When Button Pressed, select **Edit**


Process
Filter
Identification
1
Name
Edit product
Type
Execute Code
Editable Region
- Select -
Source
Location
Local Database
Language
PL/SQL
PL/SQL Code
2
BEGIN
IF manage_orders.product_exists(p_product =>
:P18_PRODUCT_ID) > 0 THEN
manage_orders.remove_product(p_product =>
:P18_PRODUCT_ID);
manage_orders.add_product (p_product =>
:P18_PRODUCT_ID,
p_quantity =>
:P18_QUANTITY);
END IF;
:P18_ACTION := 'EDIT';
Execution Options
Success Message
Error
Server-side Condition
3
When Button Pressed
Edit
Type
- Select -

Task 8: Add Process to Delete Products from the Shopping Cart

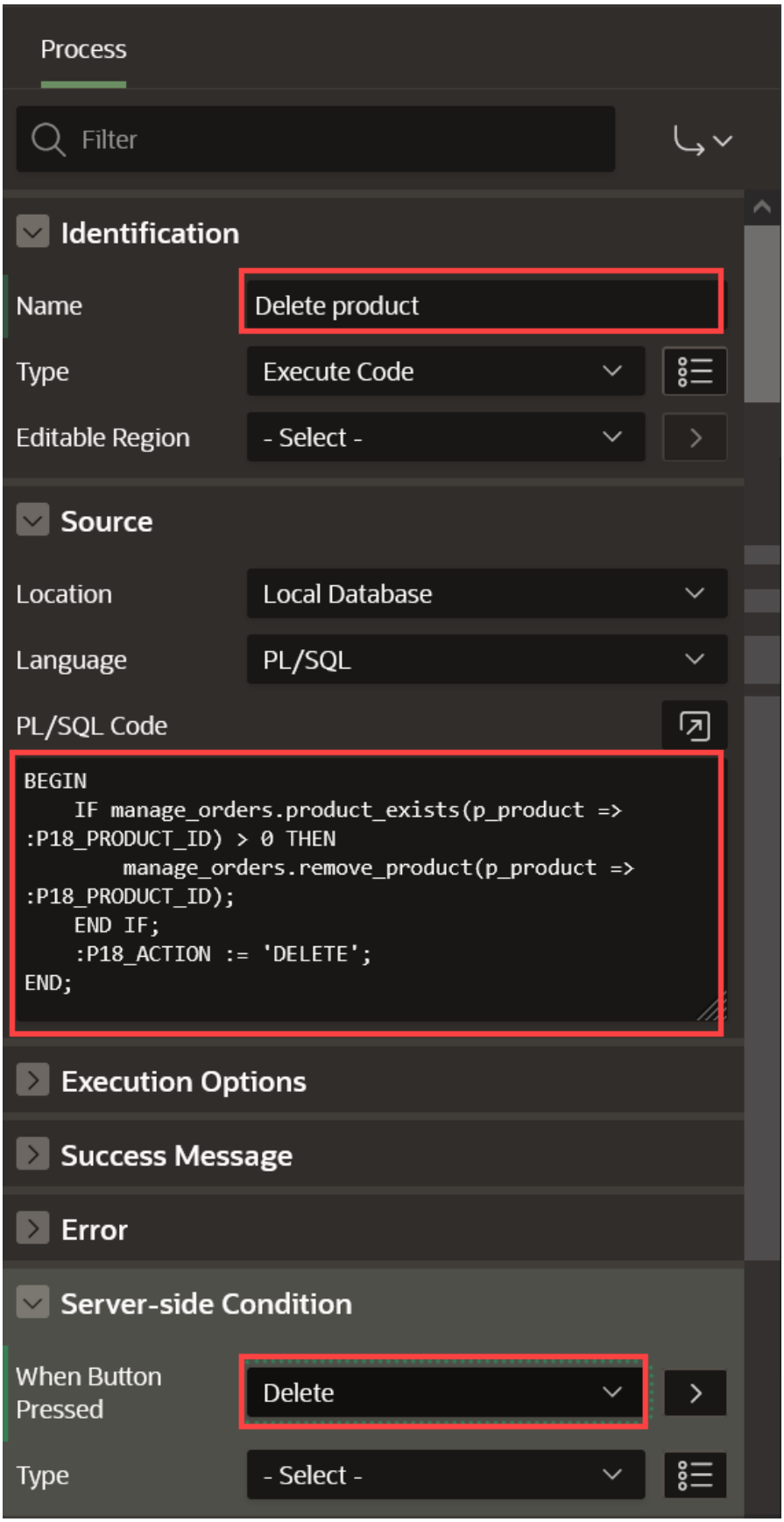
In this Task, you will call the *manage_orders.remove_product* to remove the product from the shopping cart.

1. In the **Processing** tab.
2. Right click **Processing** and click **Create Process**.
3. In the Property Editor, enter the following:
    - For Name - enter **Delete product**
    - For Type - select **Execute Code**
    - For PL/SQL Code - enter the following code:

```
Copy
BEGIN

    IF manage_orders.product_exists(p_product => :P18_PRODUCT_ID) > 0 THEN

        manage_orders.remove_product(p_product => :P18_PRODUCT_ID);

    END IF;

    :P18_ACTION := 'DELETE';
```

```
END;
```

    - Under Server-side Condition section:
        - For When Button Pressed - select **Delete**


Process
Filter
Identification
Name
Delete product
Type
Execute Code
Editable Region
- Select -
Source
Location
Local Database
Language
PL/SQL
PL/SQL Code
BEGIN
IF manage_orders.product_exists(p_product =>
:P18_PRODUCT_ID) > 0 THEN
manage_orders.remove_product(p_product =>
:P18_PRODUCT_ID);
END IF;
:P18_ACTION := 'DELETE';
END;
Execution Options
Success Message
Error
Server-side Condition
When Button Pressed
Delete
Type
- Select -

Task 9: Add Process to Calculate the Shopping Cart Items

In this task, you will call the *manage_orders.get_quantity* to get the total of products in the shopping cart.

1. In the **Processing** tab.

2. Right click **Processing** and click **Create Process**.

3. In the Property Editor, enter the following:

    - For Name - enter **Calculate Shopping Cart Items**
    - For Type - select **Execute Code**
    - For PL/SQL Code - enter the following PL/SQL code:

```
Copy
```

```
BEGIN
```

```
    :P18_SHOPPING_CART_ITEMS := manage_orders.get_quantity;
```

```
END;
```

Task 10: Add Process to Close the Modal Page

After the customer has taken action (add/edit/delete) about the product, the modal page will close and continue the shopping process.

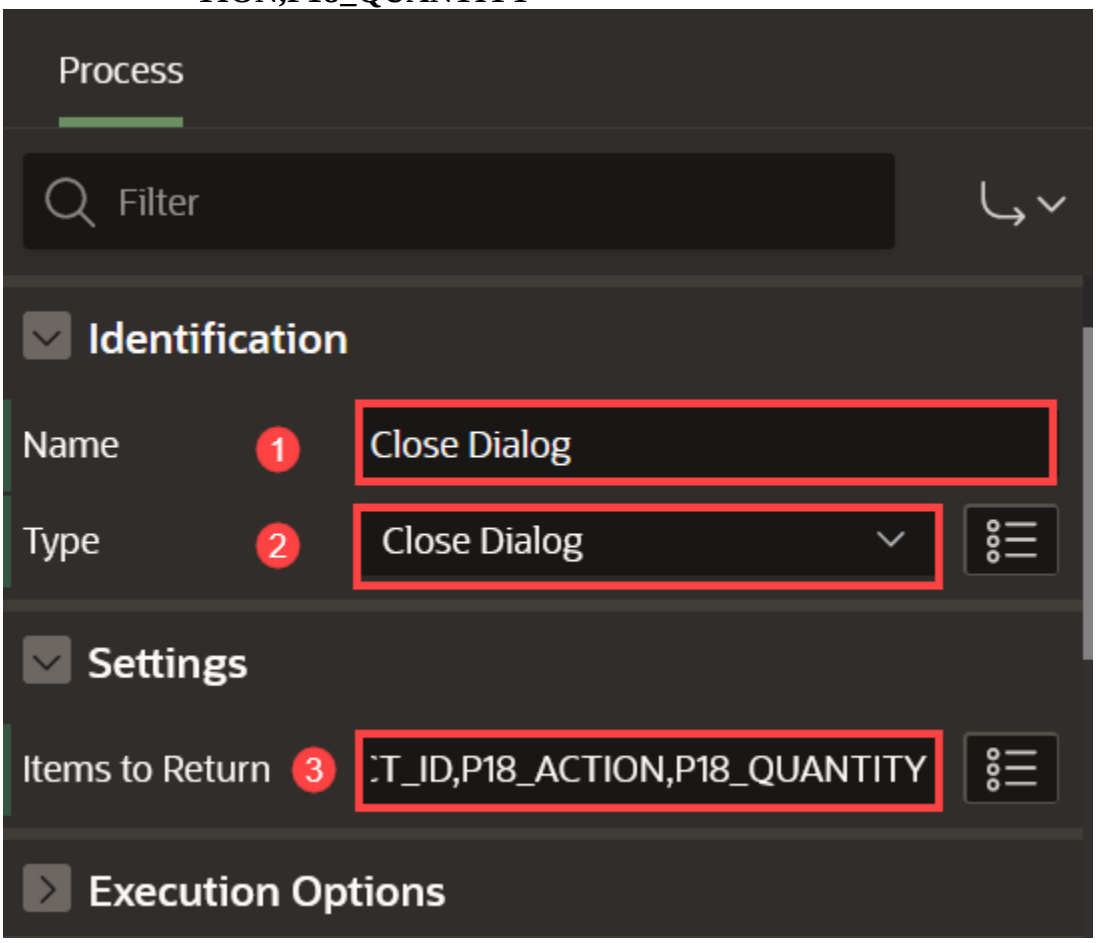
1. In the **Processing** tab.

2. Right click **Processing** and click **Create Process**.

3. In the Property Editor, enter the following:

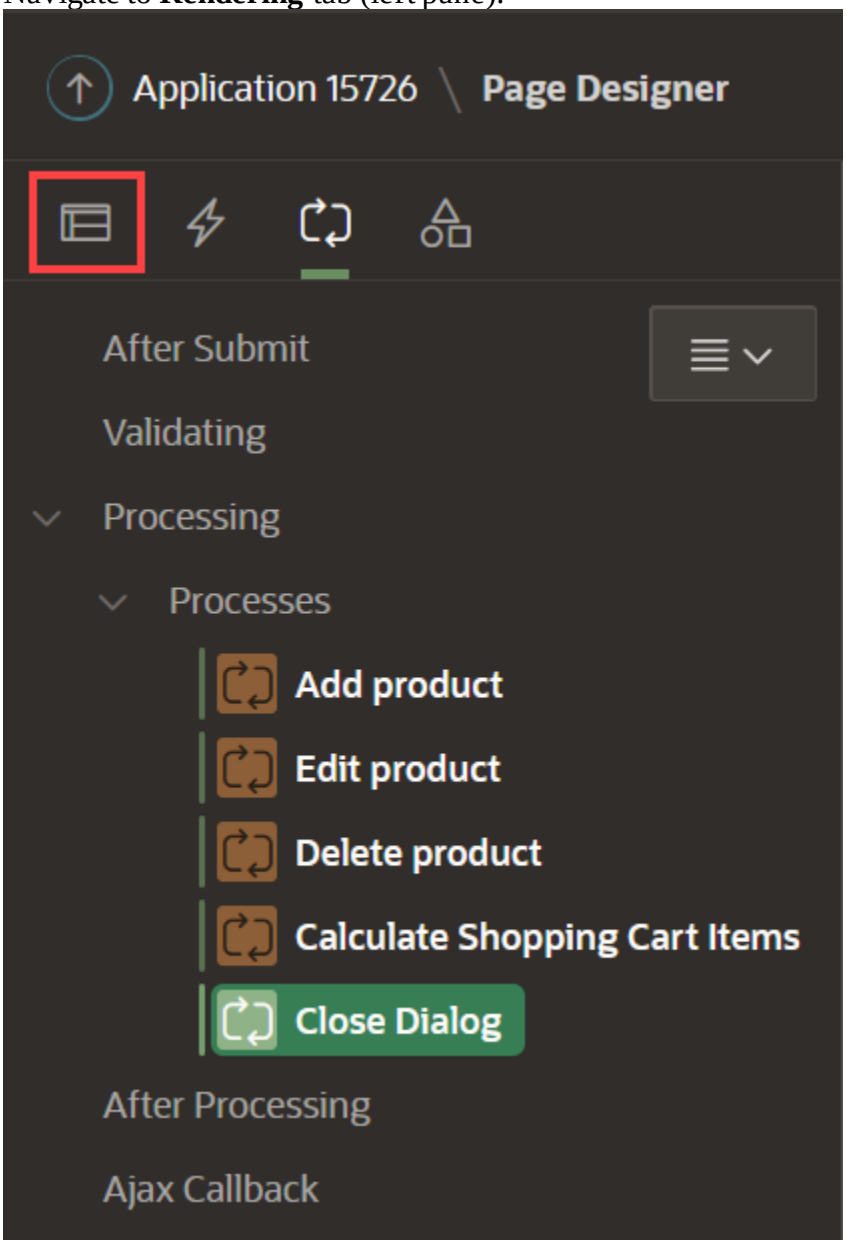
    - Under Identification section:
        - For Name - enter **Close Dialog**
        - For Type - select **Close Dialog**
    - Under Settings section:

- For Items to Return - enter **P18_SHOPPING_CART_ITEMS,P18_PRODUCT_ID,P18_ACTION,P18_QUANTITY**



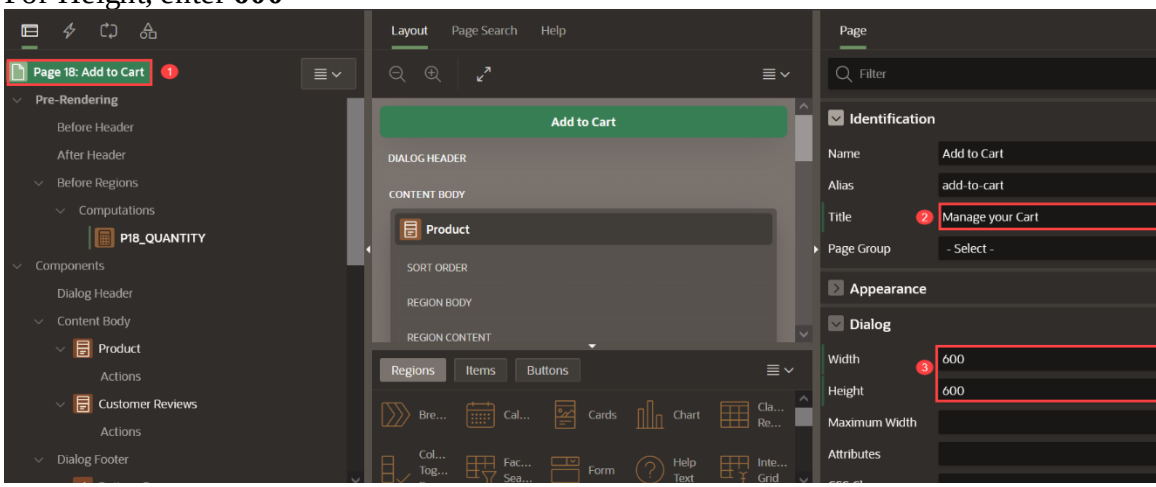
Task 11: Enhance the Modal Page

1. Navigate to **Rendering** tab (left pane).



2. Navigate to **Page 18: Add to Cart**
3. In the Property Editor, do the following changes:
    - Under Identification section: For Title, enter **Manage your Cart**
    - Under Dialog section:
        - For Width, enter **600**

- For Height, enter **600**



4. Click **Save**.

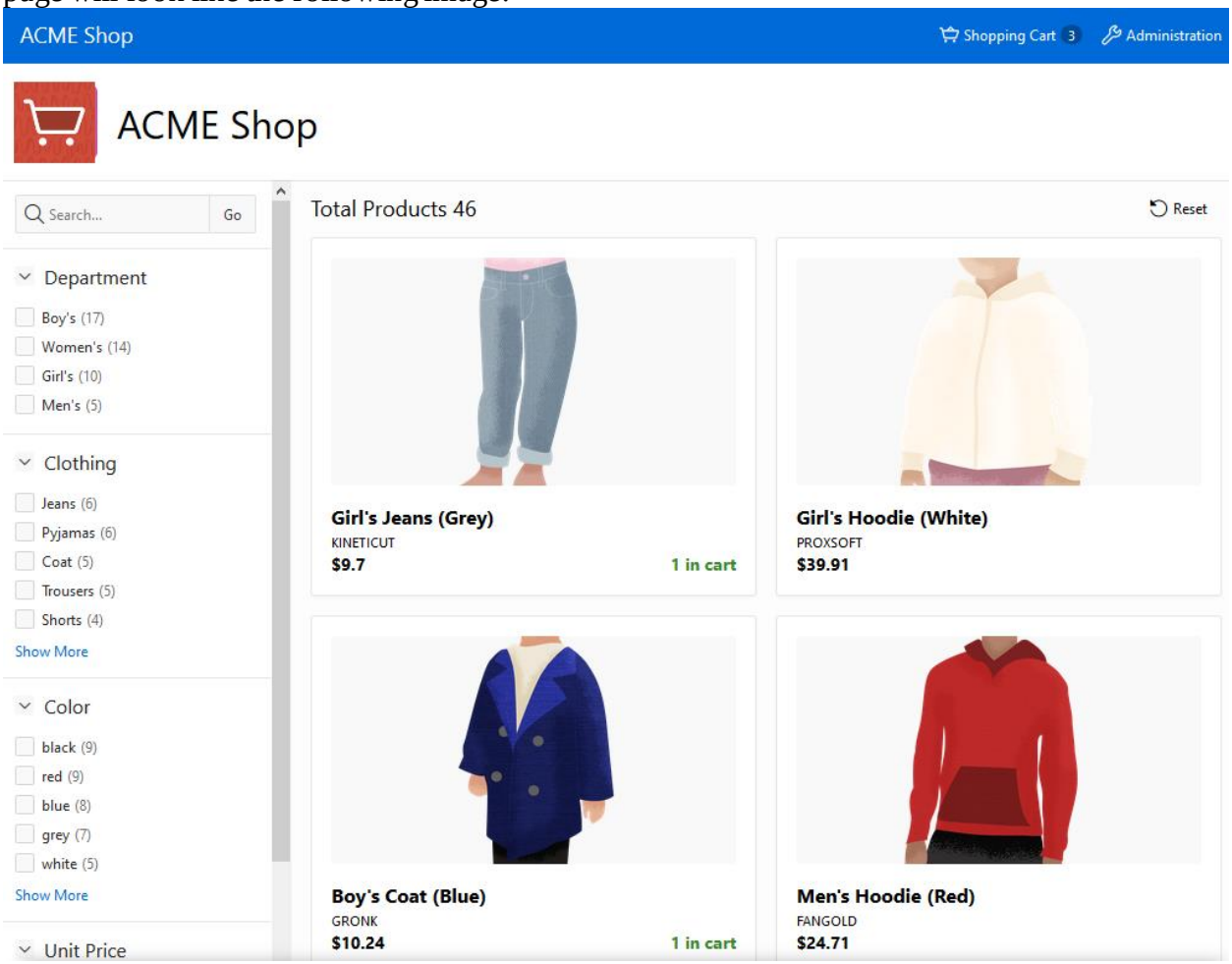
You now know how to customize and enhance an APEX page. You may now **proceed to the next lab**.

Improve the products page

Introduction

In this lab, you will learn how to improve the Products page by adding new facets and customizing the cards.

Once you have finished the workshop and updated all the products as described in the steps, your page will look like the following image:



Please note that customer can quickly identify the products that already have been added to the shopping cart.

Estimated Time: 20 minutes

Objectives

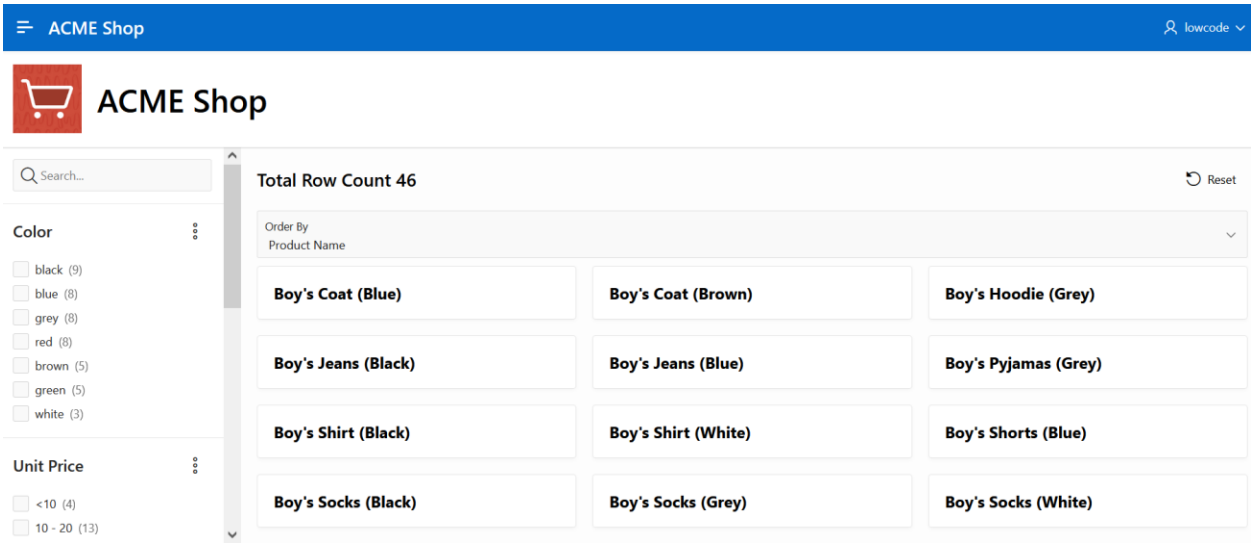
In this lab, you will:

- Improve both Faceted Search and Cards region
- Add Dynamic Actions to the page

Collapse All Tasks

Task 1: Reorder Facets

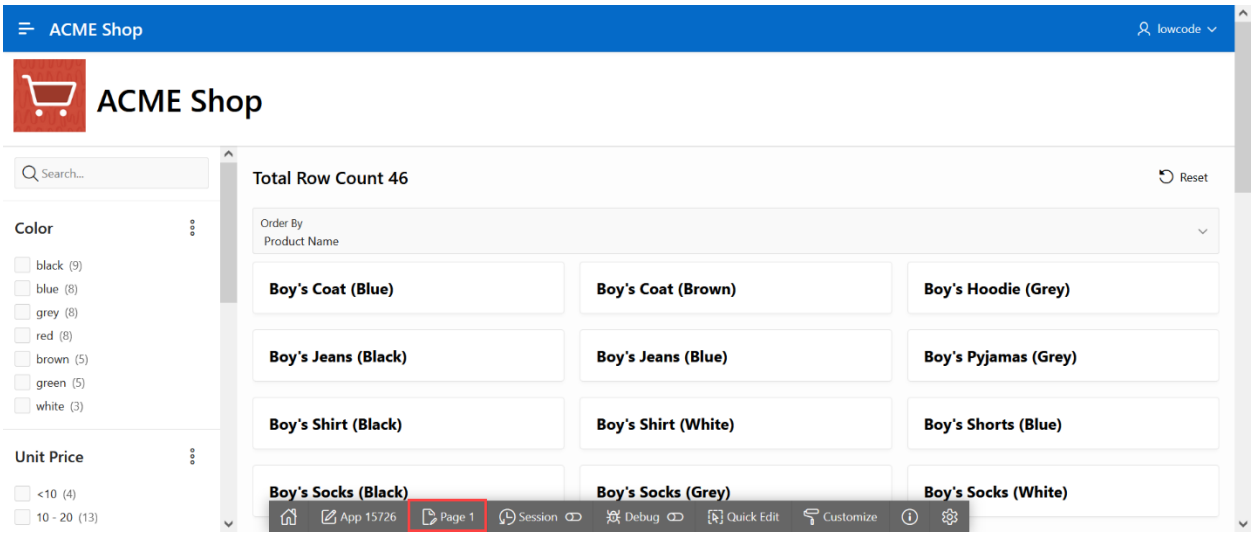
The **Products** page is where your customers can explore the products and select what they wish to buy. As you can see, it's hard to find the products and it would be beneficial to see additional details related to the products.



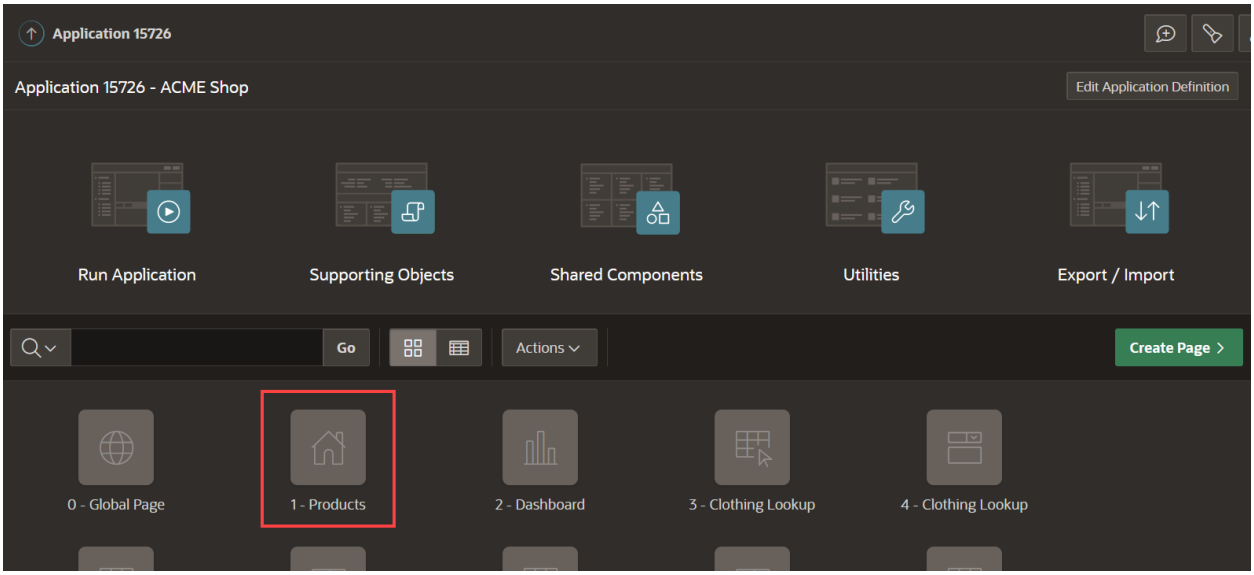
1. From the runtime application, navigate to the **Products** page in **Page Designer**.

   Given that you run this app from the APEX App Builder, you will find the Developer Toolbar at the bottom of the screen. *{Note: End users who log directly into the app will not see this toolbar.}*

   In the Developer Toolbar click **Edit Page 1**.

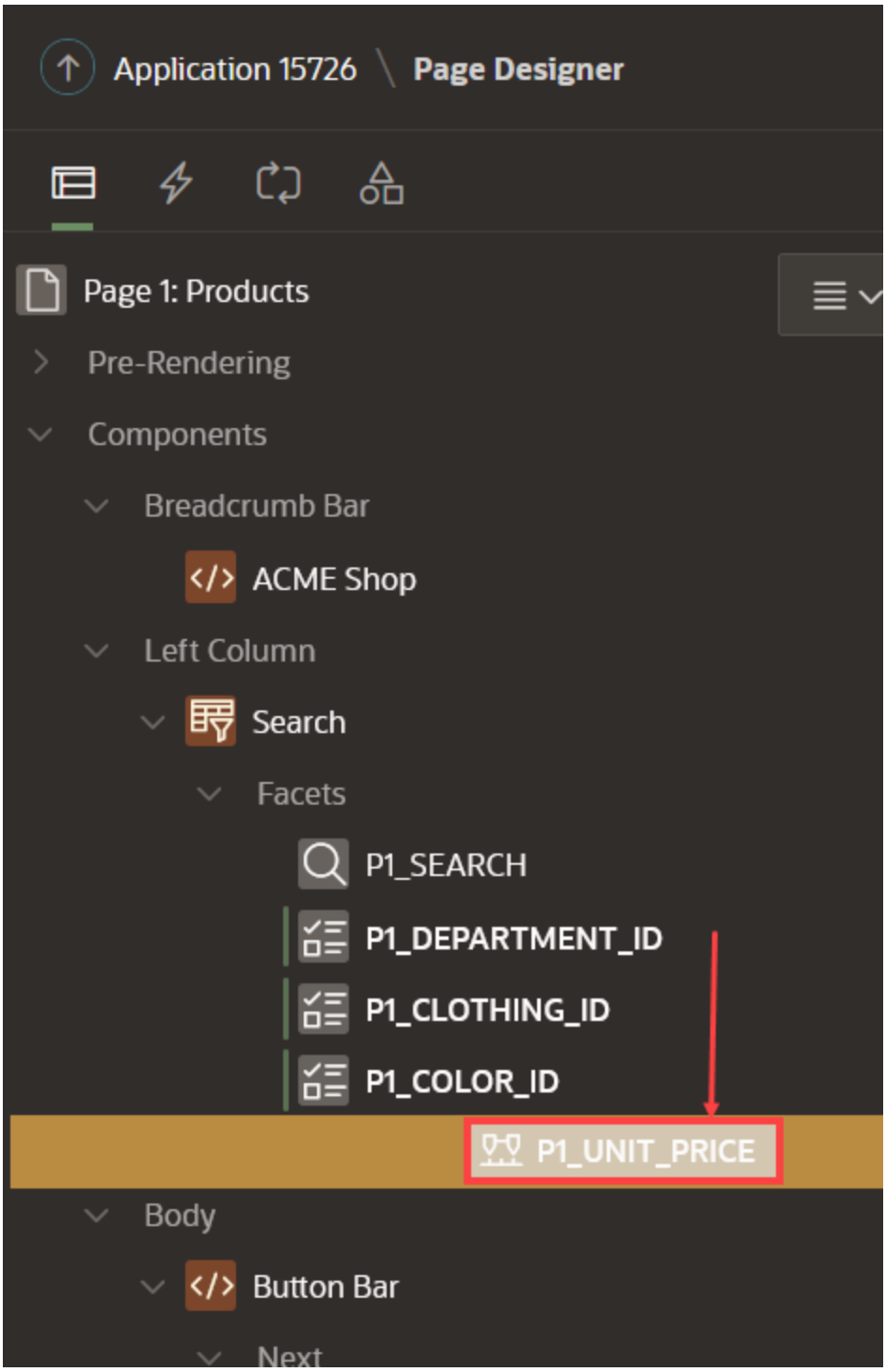


   Alternatively, you can also navigate back to the APEX App Builder tab in your browser manually by selecting the appropriate browser tab or window.
   Once in the App Builder click **1 - Products**.



You should now be in Page Designer with **Page 1: Products** loaded.

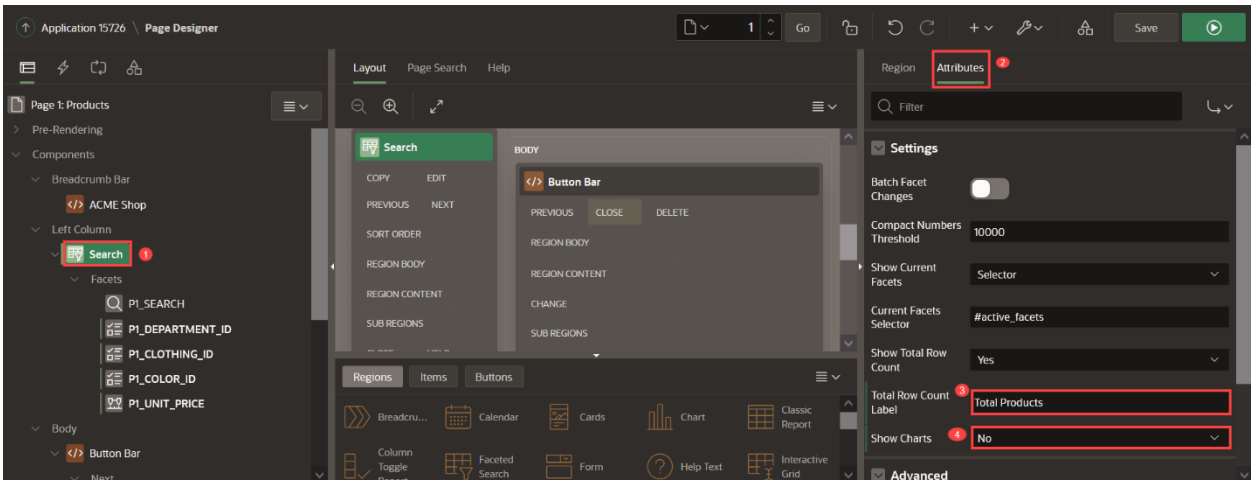
2. Unit price is not a common search criteria, so you want to put this facet at the bottom. In the Rendering tree (left pane), under Search, within Facets, click and hold **P1_UNIT_PRICE** and drag it down until it is under **P1_COLOR_ID**, then release the mouse. Reorder the facets to display as in this image.



Task 2: Enhance the Faceted Search

1. In the Rendering tree (left pane), navigate to **Search**.
2. In the Property Editor (right pane), click **Attributes** and do the following:

- For Total Row Count Label - enter **Total Products**
- For Show Charts - select **No**



Task 3: Enhance the Cards Region

1. In the Rendering tree (left pane), navigate to **Search Results** and in the Property Editor (right pane), do the following:
   - For SQL Query - enter the following SQL code:

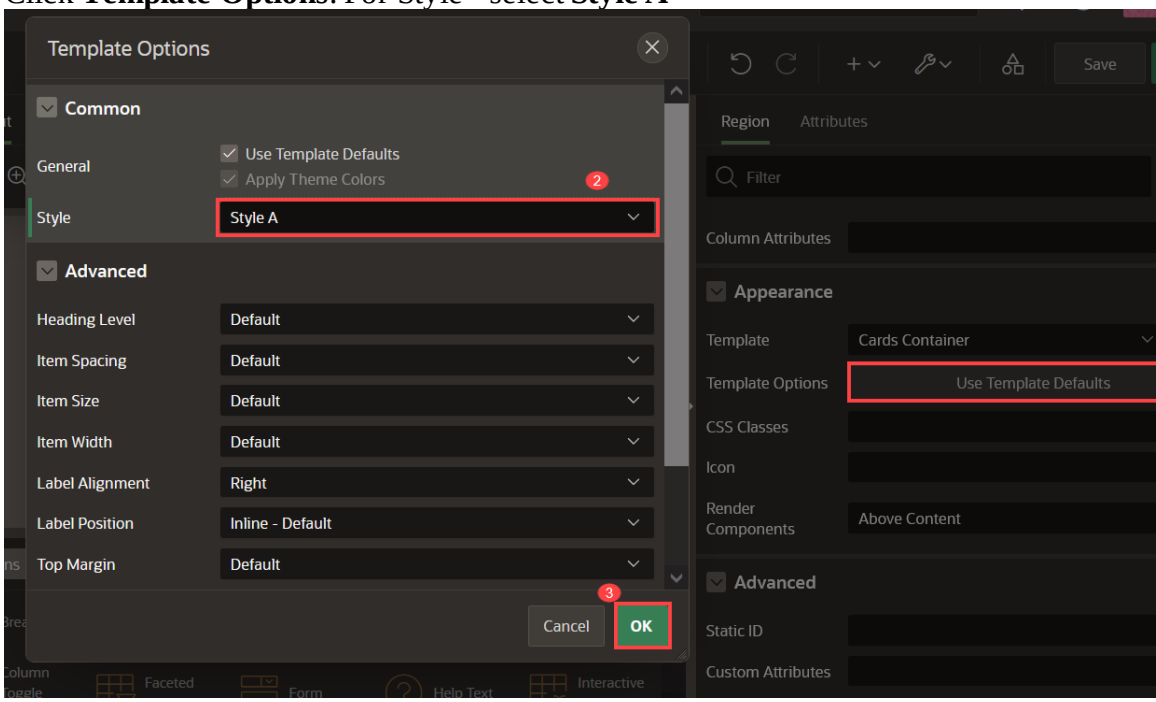
```
Copy SELECT "PRODUCT_ID",
       "PRODUCT_NAME",
       "UNIT_PRICE",
       "PRODUCT_DETAILS",
       "PRODUCT_IMAGE",
       "IMAGE_MIME_TYPE",
       "IMAGE_FILENAME",
       "IMAGE_CHARSET",
       "IMAGE_LAST_UPDATED",
       "COLOR_ID",
```

```
o      (
o              SELECT l1."COLOR"
o              FROM   "COLOR_LOOKUP" l1
o              WHERE  l1."COLOR_ID" = m."COLOR_ID")
  "COLOR_ID_L$1",
o      "DEPARTMENT_ID",
o      (
o              SELECT l2."DEPARTMENT"
o              FROM   "DEPARTMENT_LOOKUP" l2
o              WHERE  l2."DEPARTMENT_ID" =
  m."DEPARTMENT_ID") "DEPARTMENT_ID_L$2",
o      "CLOTHING_ID",
o      (
o              SELECT l3."CLOTHING"
o              FROM   "CLOTHING_LOOKUP" l3
o              WHERE  l3."CLOTHING_ID" = m."CLOTHING_ID")
  "CLOTHING_ID_L$3",
o      b.brand
o  FROM   "PRODUCTS" m,
```

```
    json_table (m.product_details, '$' columns ( brand
varchar2(4000) path '$.brand') ) b
```

- Under Appearance section:

- Click **Template Options**. For Style - select **Style A**

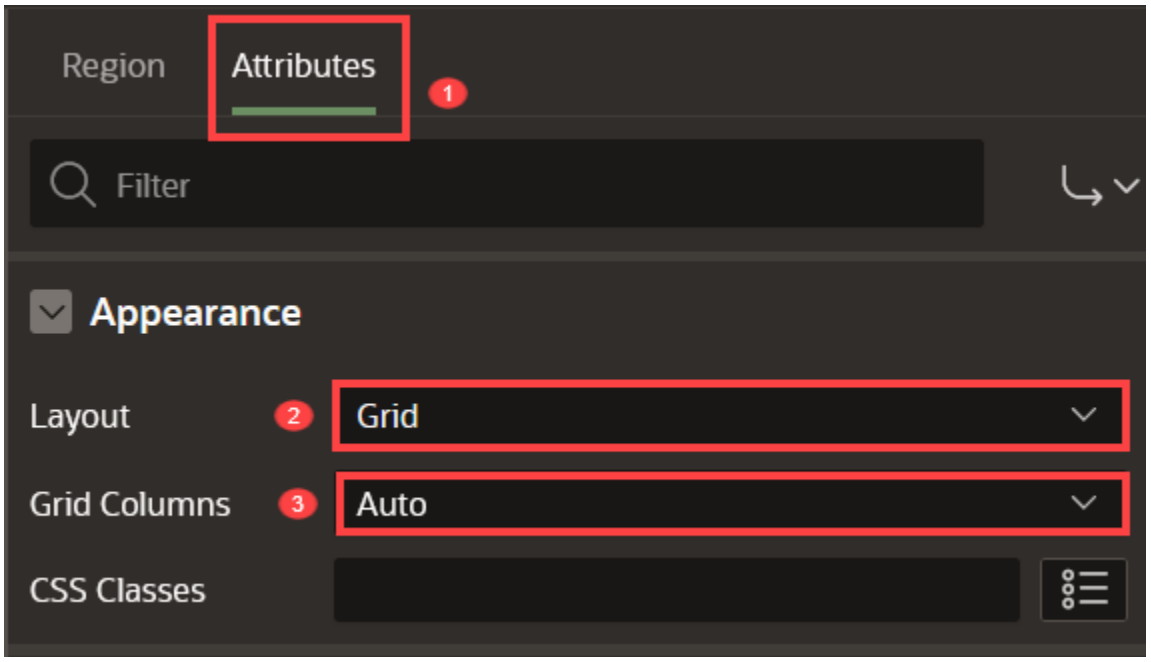


- Click **Ok**

2. Click **Attributes** and apply the following changes:

   - Under Appearance section:

     - For Layout - select **Grid**
     - For Grid Columns - select **Auto**



   - Under Title section:

- For Column - select **PRODUCT_NAME**

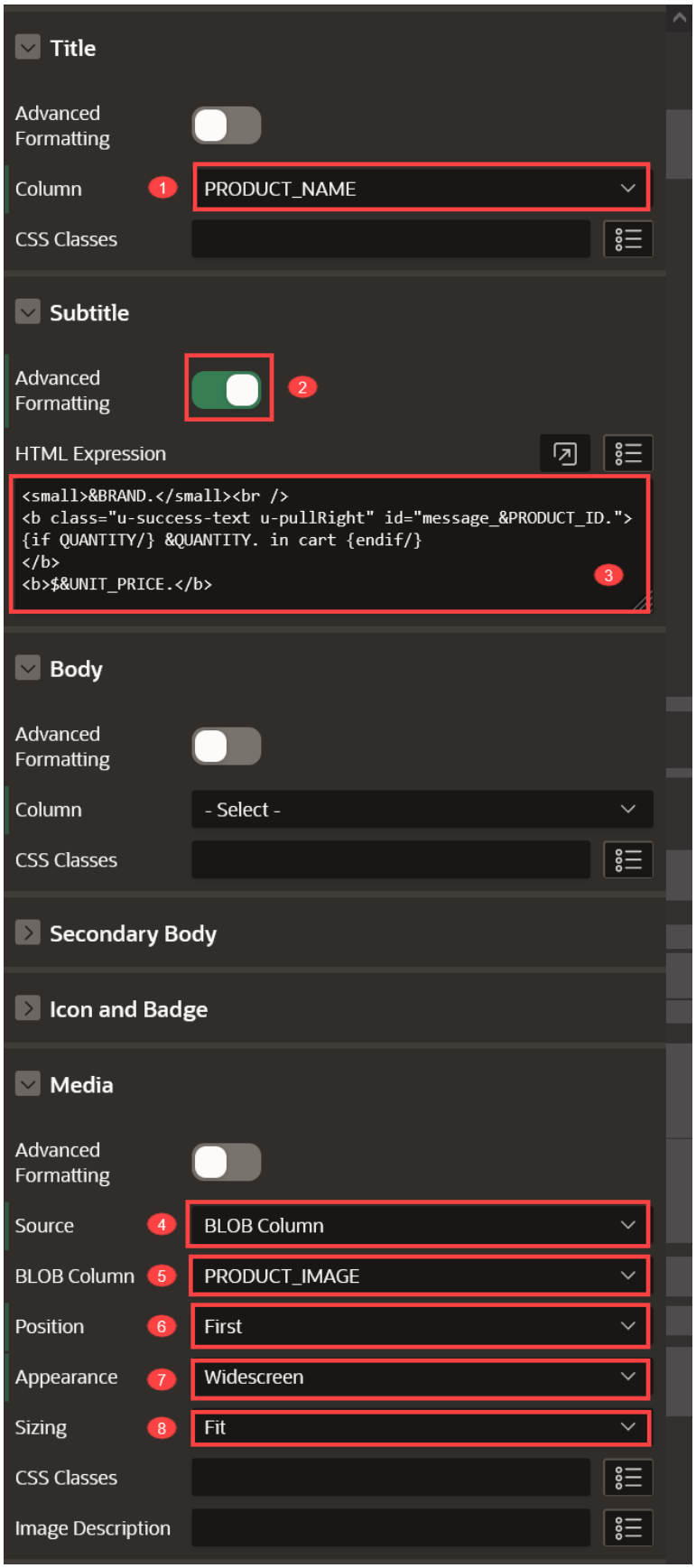
- Under Subtitle section:
  - Set Advanced Formatting to **On**
  - For HTML Expression - enter the following:

```
Copy<small>&BRAND.</small><br />
<b class="u-success-text u-pullRight"
id="message_&PRODUCT_ID.">
{if QUANTITY/} &QUANTITY. in cart {endif/}
</b>
```

```
<b>$&UNIT_PRICE.</b>
```

- Under Media section:
  - For Source - select **BLOB Column**
  - For BLOB Column - select **PRODUCT_IMAGE**
  - For Position - select **First**
  - For Appearance - select **Widescreen**

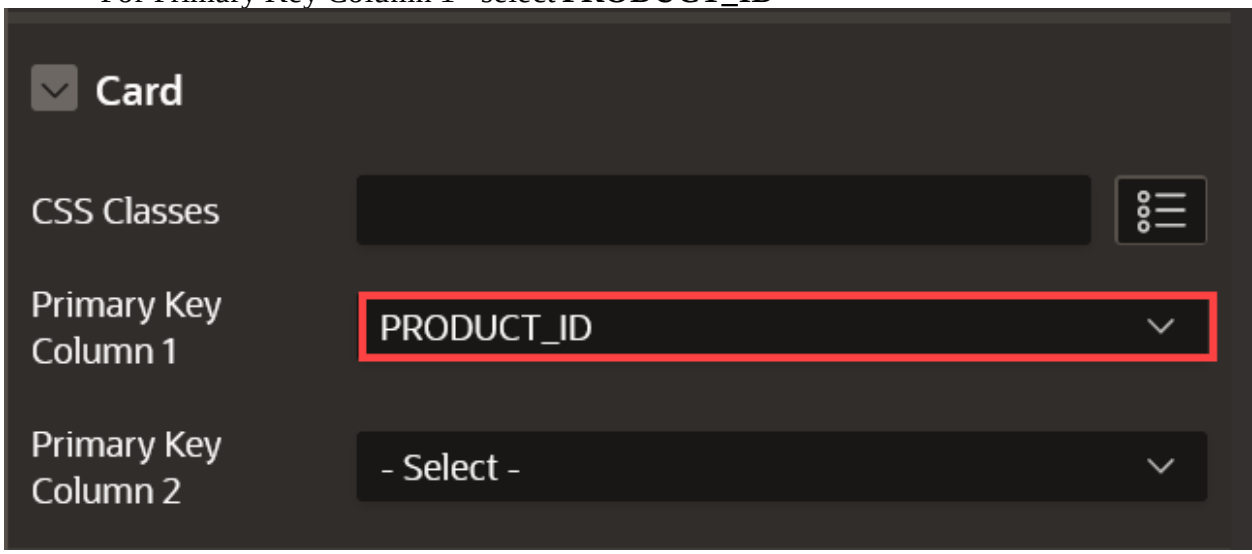
- For Sizing - select **Fit**


Title
Advanced Formatting
Column
1
PRODUCT_NAME
CSS Classes
Subtitle
Advanced Formatting
2
HTML Expression
<small>&BRAND.</small>

{if QUANTITY/} &QUANTITY. in cart {endif/}

$&UNIT_PRICE.
3
Body
Advanced Formatting
Column
- Select -
CSS Classes
Secondary Body
Icon and Badge
Media
Advanced Formatting
Source
4
BLOB Column
BLOB Column
5
PRODUCT_IMAGE
Position
6
First
Appearance
7
Widescreen
Sizing
8
Fit
CSS Classes
Image Description

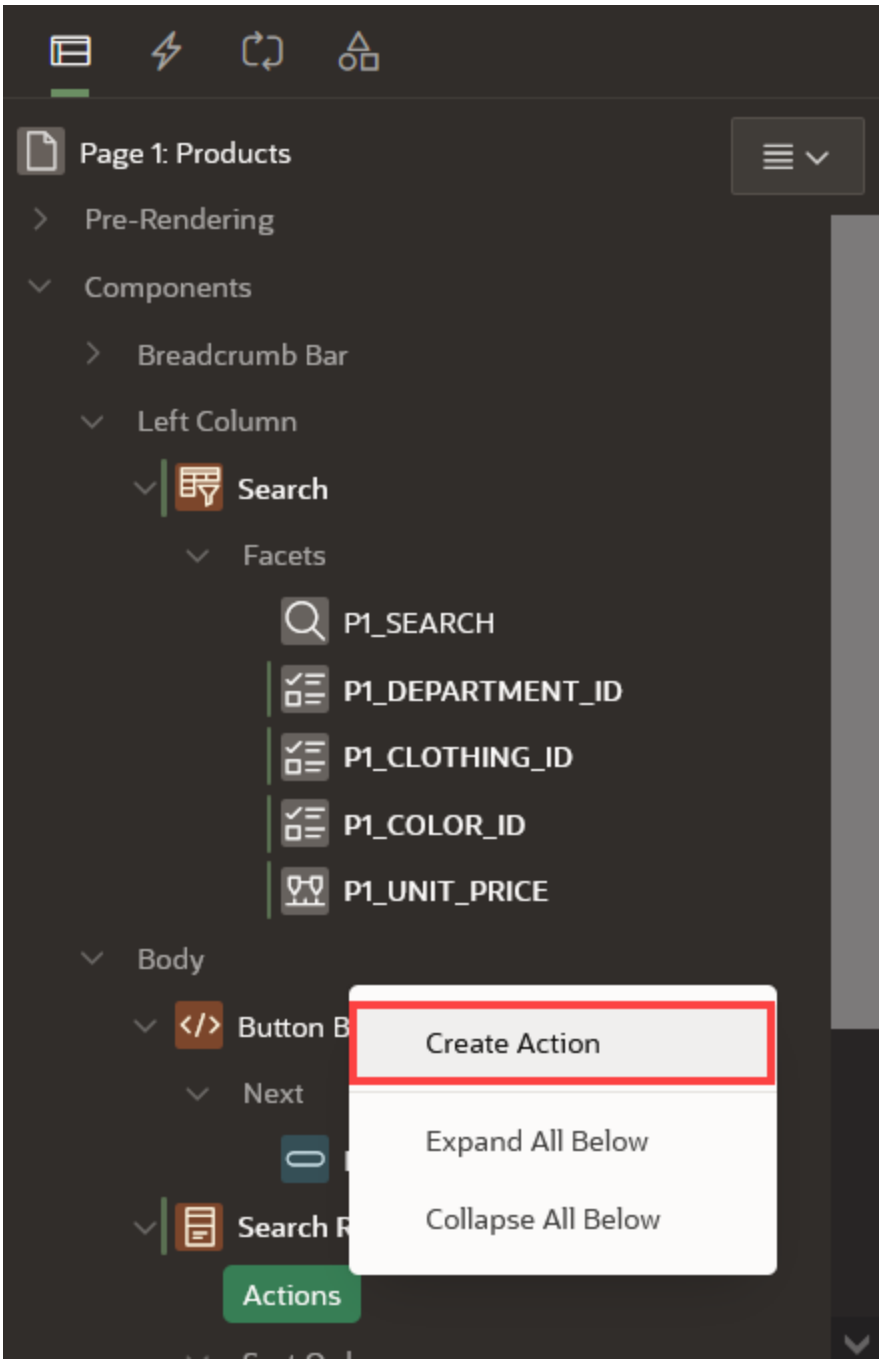
- Under Card section:
  - For Primary Key Column 1 - select **PRODUCT_ID**



Task 4: Create Actions

Customers need a way to shop the products, so in this task you will add an action to allow customers to learn more about the product.

1. Navigate to **Search Results** (left pane).
2. On Actions, right-click **Create Action**.

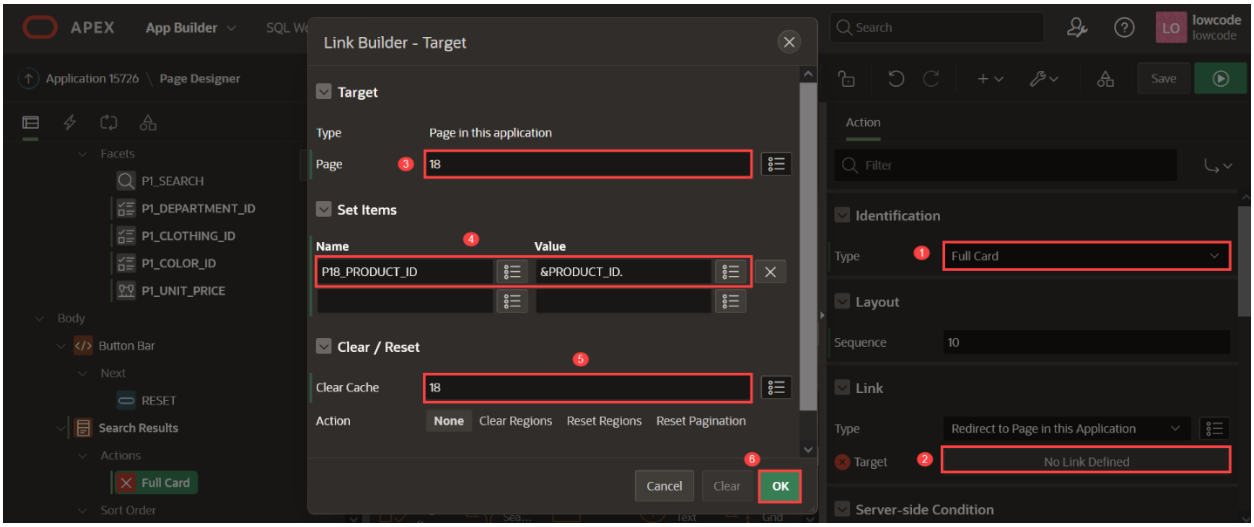


3. In the Property Editor (right pane), enter the following:
    - For Type - select **Full Card**
    - For Target - Click **No Link Defined** and do the following:
        - For Page - enter **18**.
        - For Set Items, enter:

Table 1: Build a Starter Online Shopping App with Oracle APEX! | Lab 8: Improve the Products Page

Name	Value
P18_PRODUCT_ID	&PRODUCT_ID.

- For Clear Cache, enter **18**
- Click **Ok**.



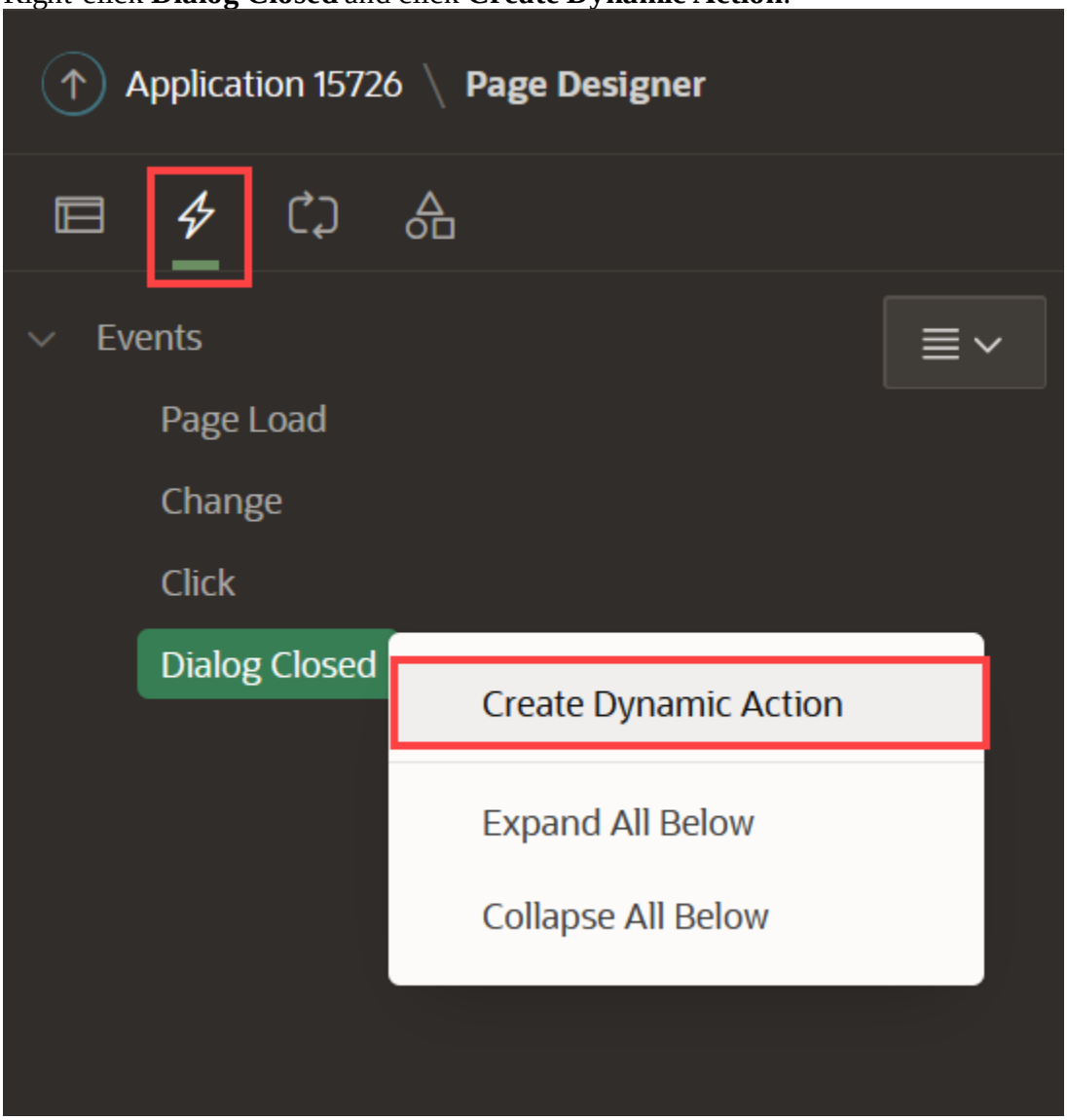
Task 6: Add Dynamic Actions

In this task, you will create two dynamic actions:

- To show a success message when a product is added/edited/removed from the shopping cart.
- To update the badge and icon shown in the navigation bar after the customer has added/edited/removed a product from the shopping cart.

1. Navigate to **Dynamic Actions** tab (left pane).

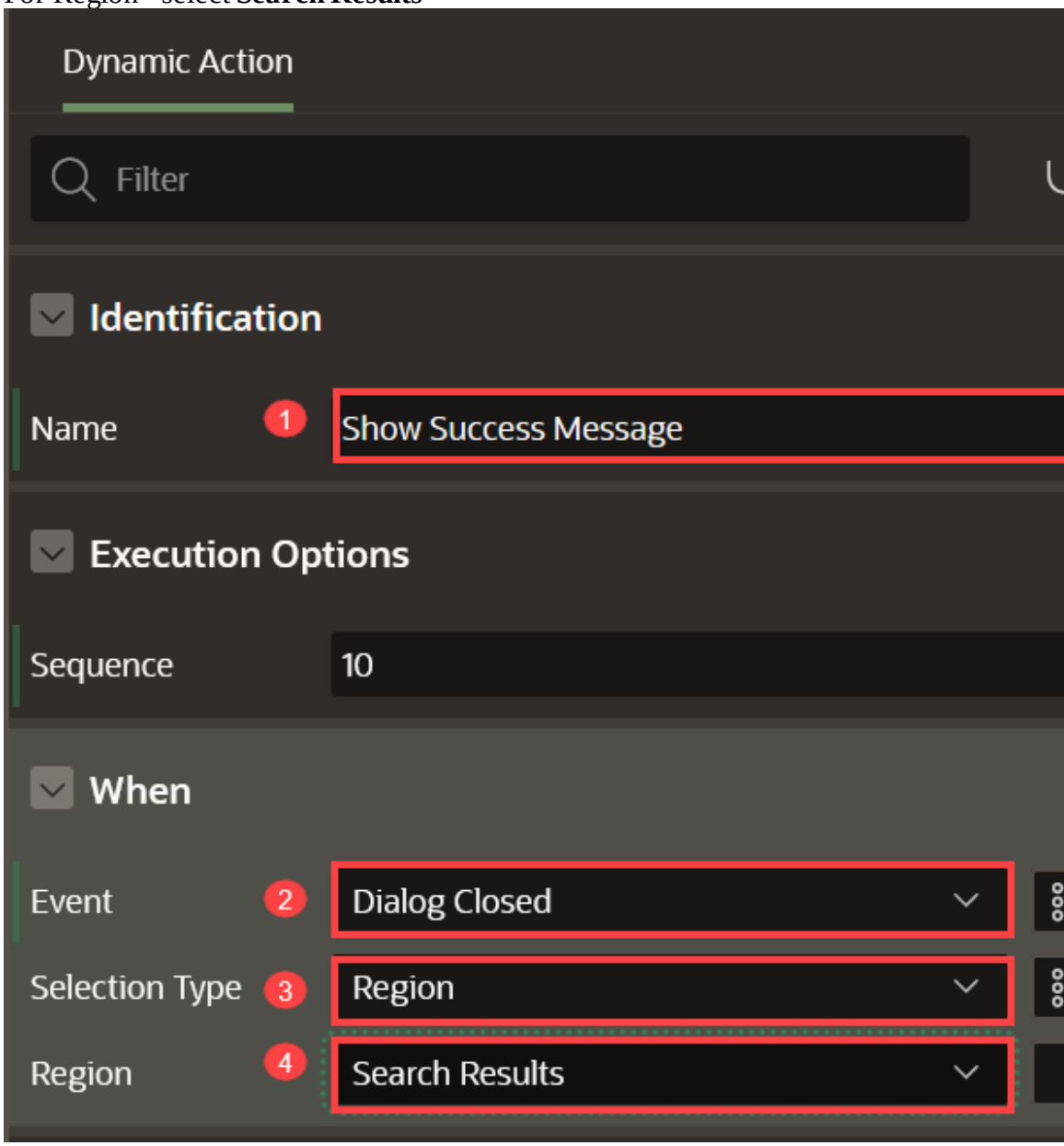
2. Right-click **Dialog Closed** and click **Create Dynamic Action**.



3. In the Property Editor, enter the following:

    - Under Identification section:
        - For Name - enter **Show Success Message**
    - Under When section:
        - For Event - select **Dialog Closed**
        - For Selection Type - select **Region**

- For Region - select **Search Results**



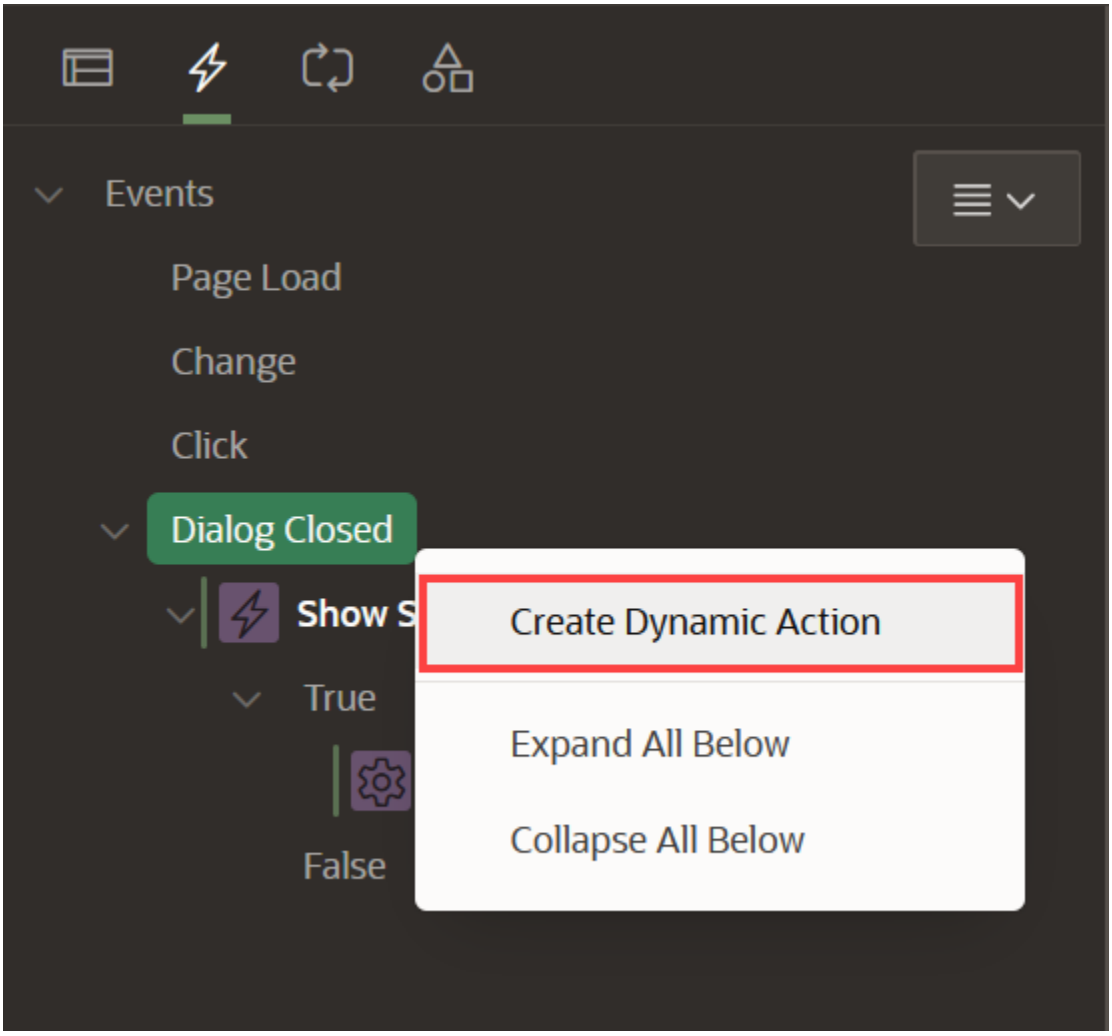
4. Navigate to **Refresh** Action.
    - Under Identification section:
        - For Action - select **Execute JavaScript Code**
    - Under Settings section:
        - For Code - enter the following JavaScript Code:

```
Copyvar productAction   = this.data.P18_ACTION,
```

```
    productQuantity = this.data.P18_QUANTITY,
    productCard$  = apex.jQuery("#message_" + this.data.P18_PRODUCT_ID);

if (productAction === 'ADD') {
    productCard$.text("Added " + productQuantity + " to cart!");
} else if (productAction === 'EDIT') {
    productCard$.text("Updated quantity to " + productQuantity + "!");
} else if (productAction === 'DELETE') {
    productCard$.text("Removed from cart!");
```

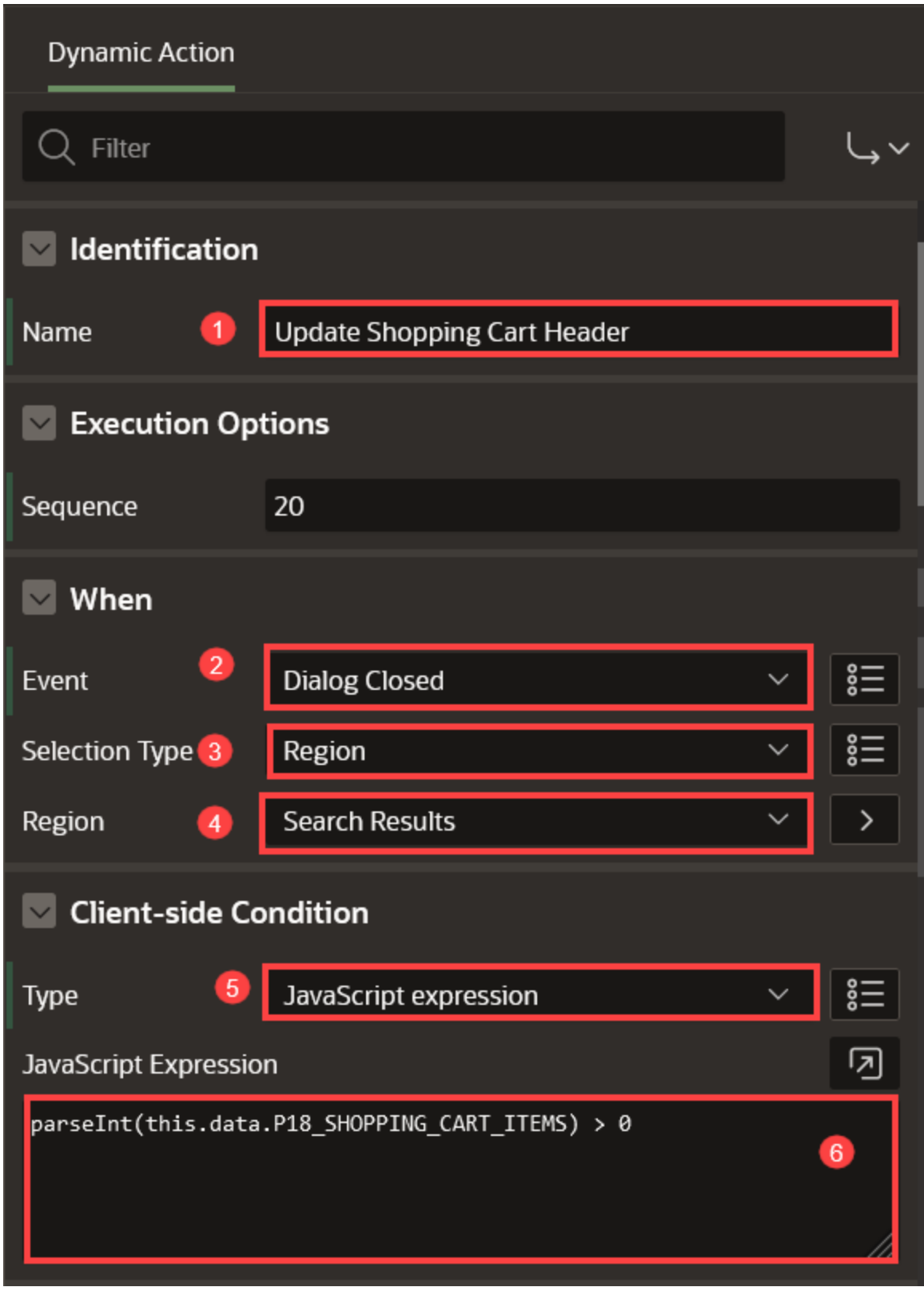
```
}
```

5. Create a second dynamic action. Right-click **Dialog Closed** and click **Create Dynamic Action**.



6. In the Property Editor, enter the following:
    - Under Identification section:
        - For Name - enter **Update Shopping Cart Header**
    - Under When section:
        - For Event - select **Dialog Closed**
        - For Selection Type - select **Region**
        - For Region - select **Search Results**
    - Under Client-side Condition:
        - For Type - select **JavaScript expression**
        - For JavaScript Expression, enter the following:

```
Copyparse Int(this.data.P18_SHOPPING_CART_ITEMS) >
0
```



7. Navigate to **Refresh** Action.

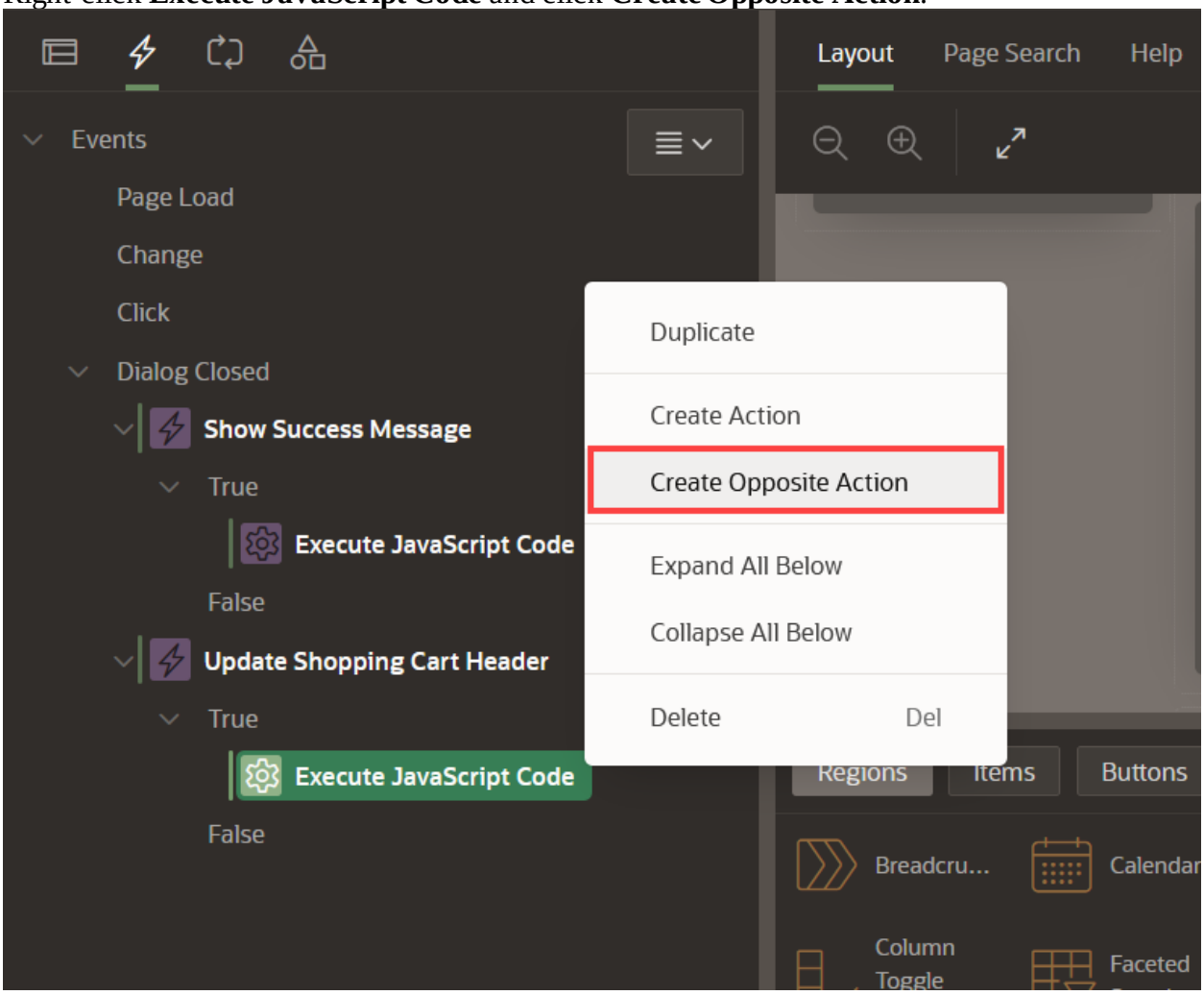
- Under Identification section:
  - For Action - select **Execute JavaScript Code**
- Under Settings section:
  - For Code - enter the following JavaScript Code:

```
Copy// Update Badge Text

apex.jQuery(".js-shopping-cart-item .t-Button-badge").text(this.data.P18_SHOPPING_CART_ITEMS);



// Update Icon
```

```
apex.jQuery(".js-shopping-cart-item .t-Icon").removeClass('fa-cart-empty').addClass('fa-cart-full');
```

8. Create an opposite action. In the Dynamic Actions tab (left pane), navigate to the newly dynamic action.

9. Right-click **Execute JavaScript Code** and click **Create Opposite Action**.



10. Navigate to **Execute JavaScript Code** Action.

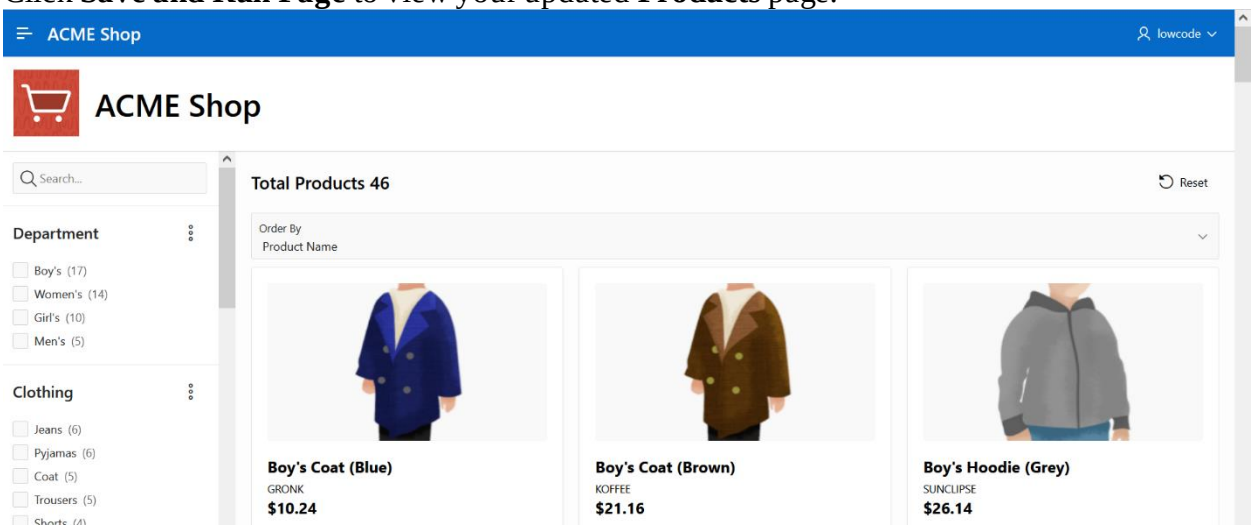
    - Under Identification section:
        - For Action - select **Execute JavaScript Code**
    - Under Settings section:
        - For Code - enter the following JavaScript Code:

```
Copy// Update Badge Text

apex.jQuery(".js-shopping-cart-item .t-Button-badge").text('');

```

```
// Update Icon
```

```
apex.jQuery(".js-shopping-cart-item .t-Icon").removeClass('fa-cart-full').addClass('fa-cart-empty');
```

11. Click **Save and Run Page** to view your updated **Products** page.



You now know how to enhance faceted search and cards region. You may now **proceed to the next lab**.

Improve the application

Introduction

In this lab, you will learn how to make some pages publicly accessible.

Estimated Time: 10 minutes

Objectives

In this lab, you will:

- Set the following pages as public pages:
    - Products
    - Shopping Cart
    - Order Information

- Disable the Navigation Menu
- Enhance the Navigation Bar

Collapse All Tasks

Task 1: Set Pages to be Public

Your customers don't need to log in the app to shop the products, so let's set the following pages as public:

- Page 1: Products
- Page 16: Order Information
- Page 17: Shopping Cart
- Page 18: Add to Cart

Follow these steps for the four listed pages:

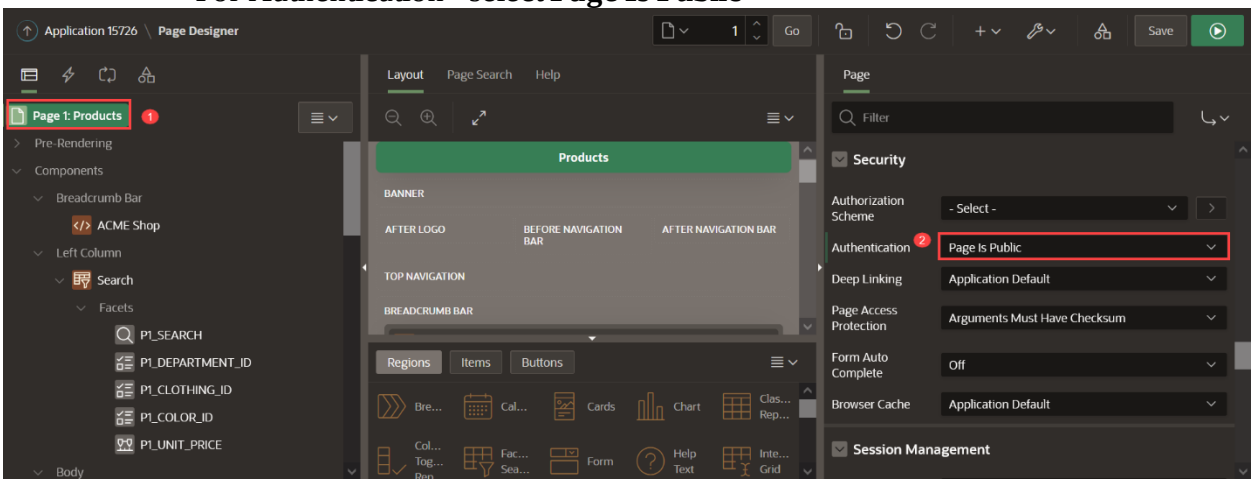
1. From the runtime application, navigate to the page.

   In the Developer Toolbar click **Edit Page x**.

2. Within Page Designer, in the Rendering tree (left pane), navigate to **Page x: Name of the Page**.

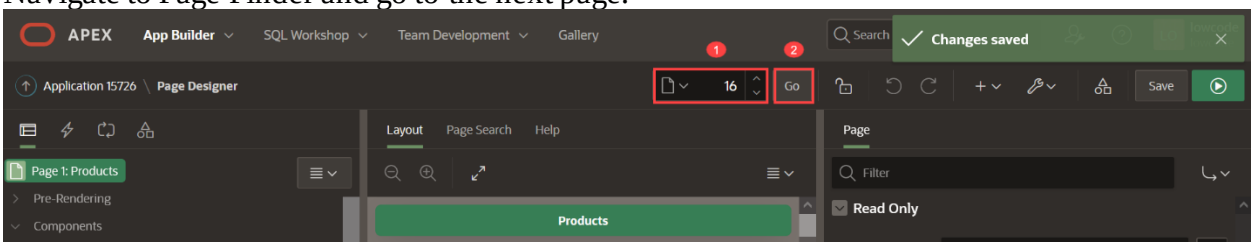
   In the Property Editor (right pane), apply the following change:

   - Under Security section:
     - For Authentication - select **Page Is Public**



3. Click **Save**.

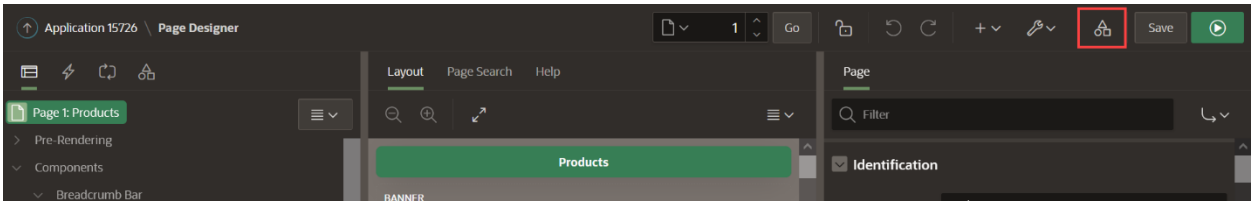
4. Navigate to Page Finder and go to the next page.



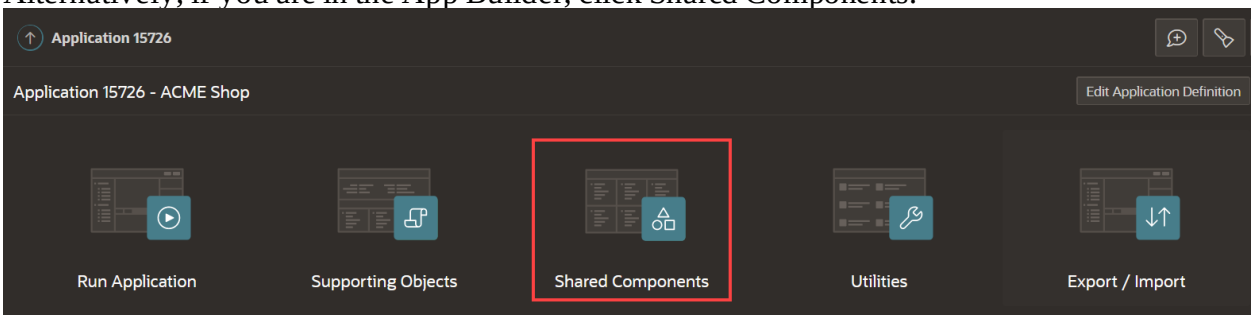
Task 2: Clean the Navigation Menu

Since the home page is the Products page and this is a public access page as some others too, it's not needed to have a navigation menu. In this task, you will turn off displaying the navigation menu.

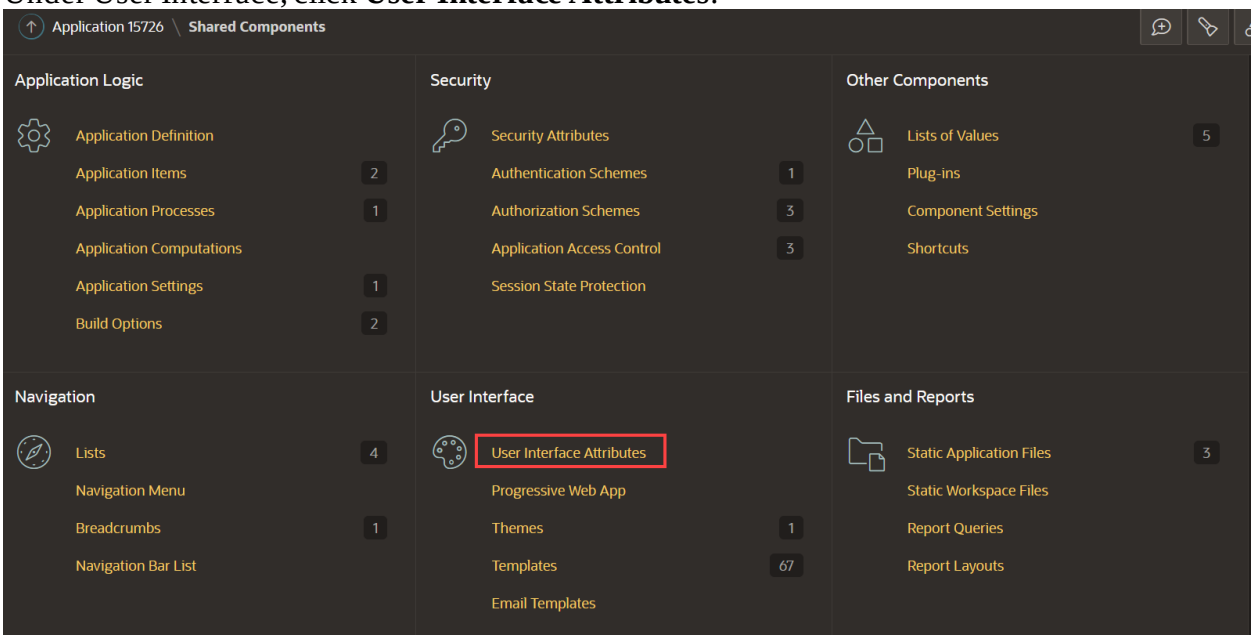
1. Within Page Designer, click the Shared Components icon at the top right.



Alternatively, if you are in the App Builder, click Shared Components.

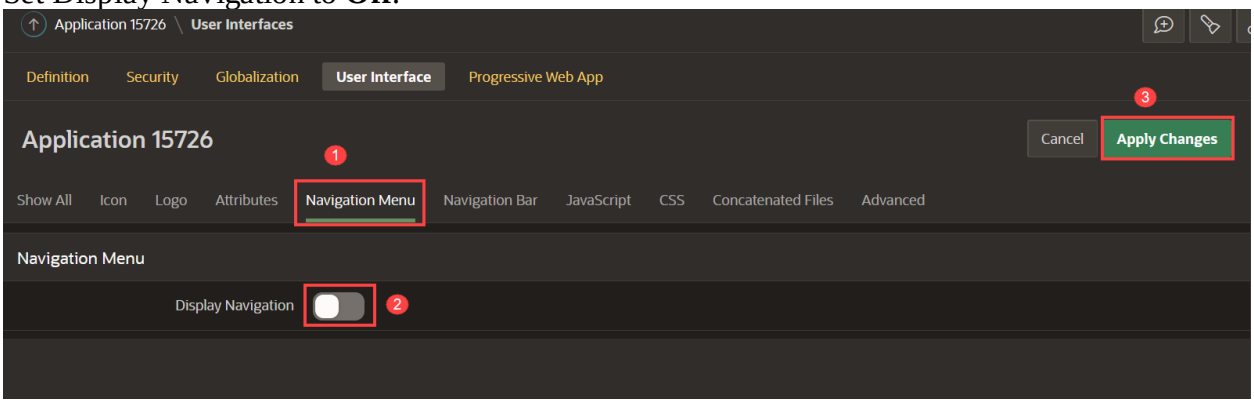


2. Under User Interface, click **User Interface Attributes**.



3. Click **Navigation Menu**.

4. Set Display Navigation to **Off**.



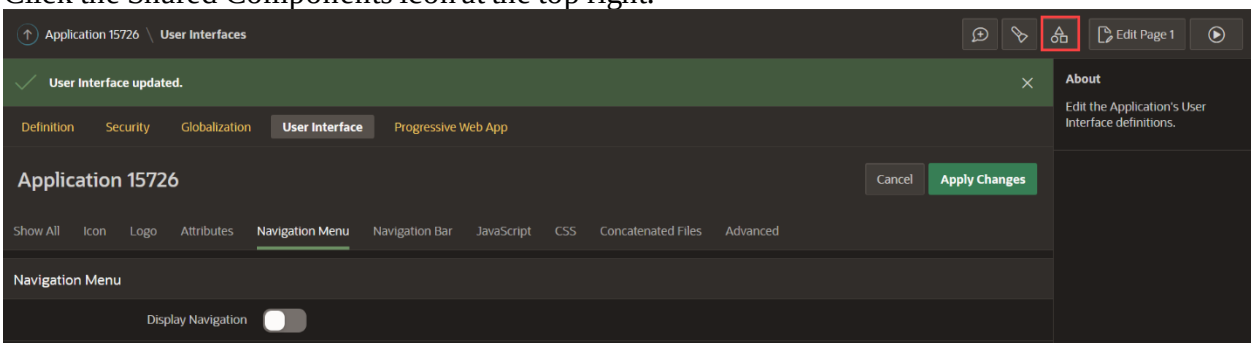
5. Click **Apply Changes**.

Task 3: Enhance the Navigation Bar List

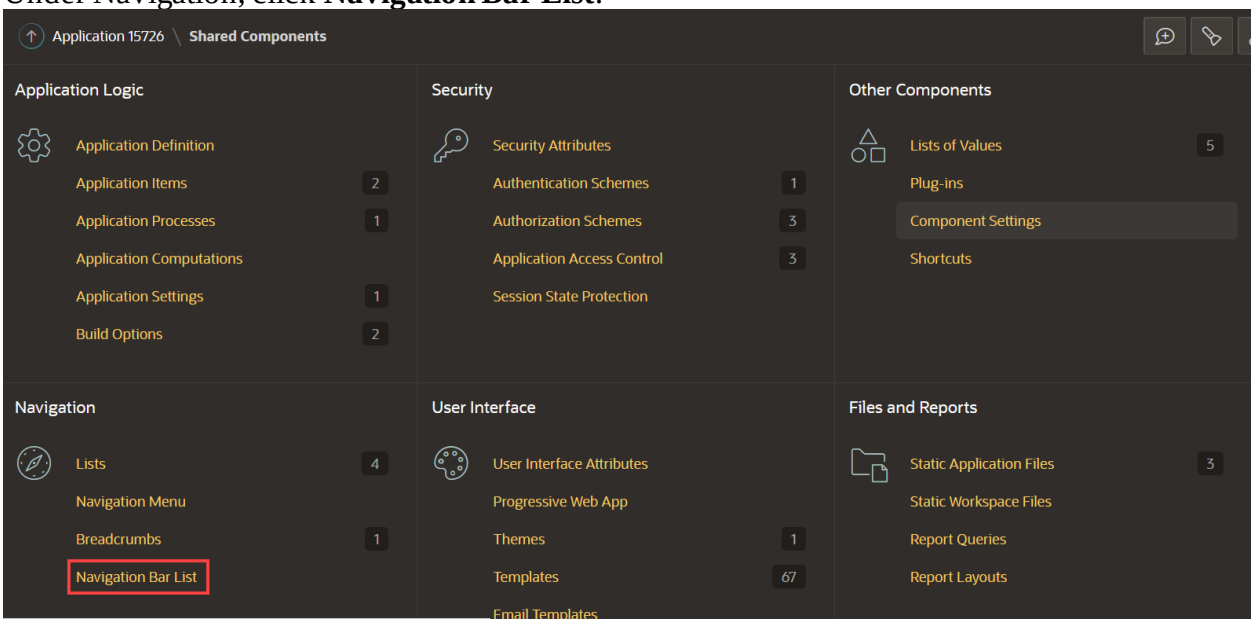
Add a new navigation menu entry to allow:

- Customers to go directly to the Shopping Cart
- Administrators to login and access to administration page

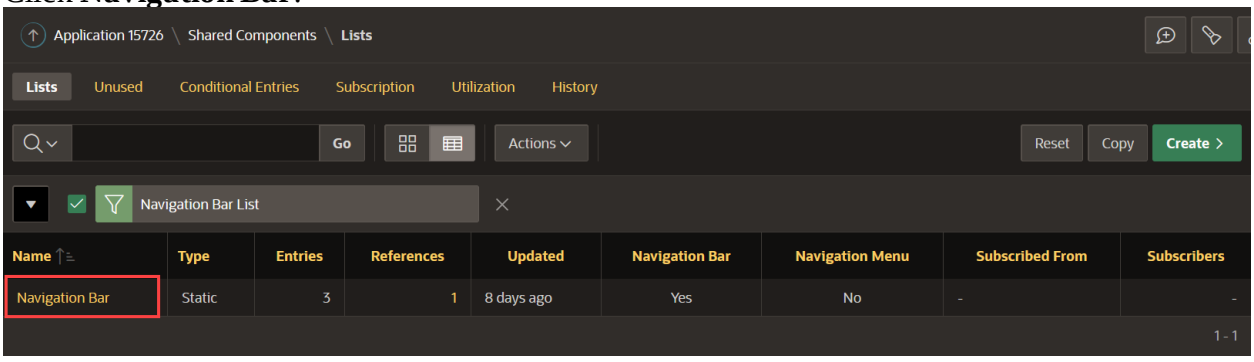
1. Click the Shared Components icon at the top right.



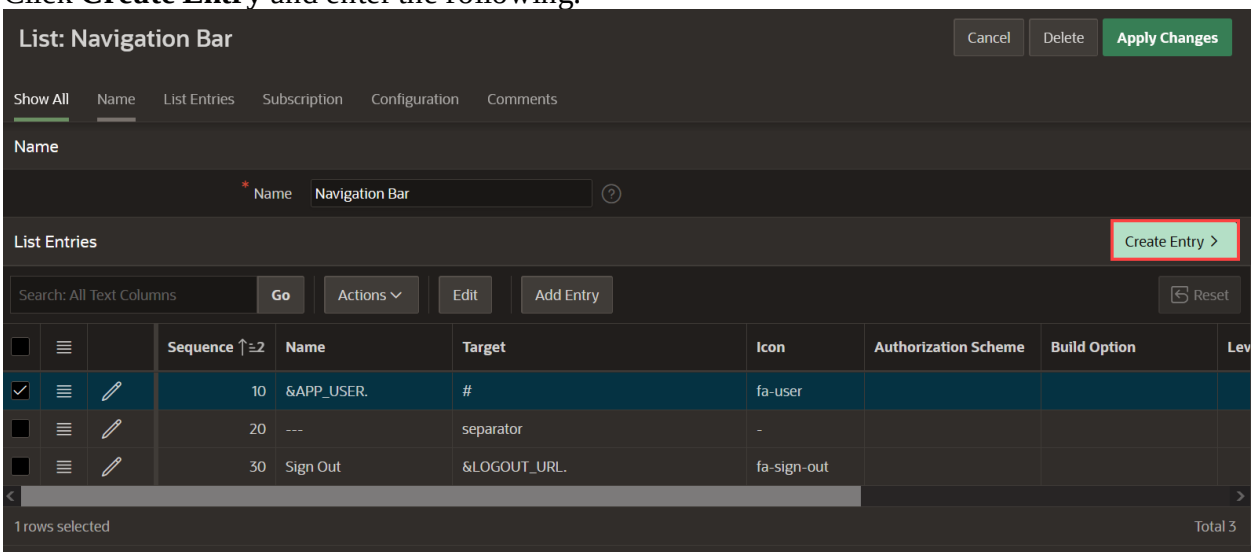
2. Under Navigation, click **Navigation Bar List**.



3. Click **Navigation Bar**.

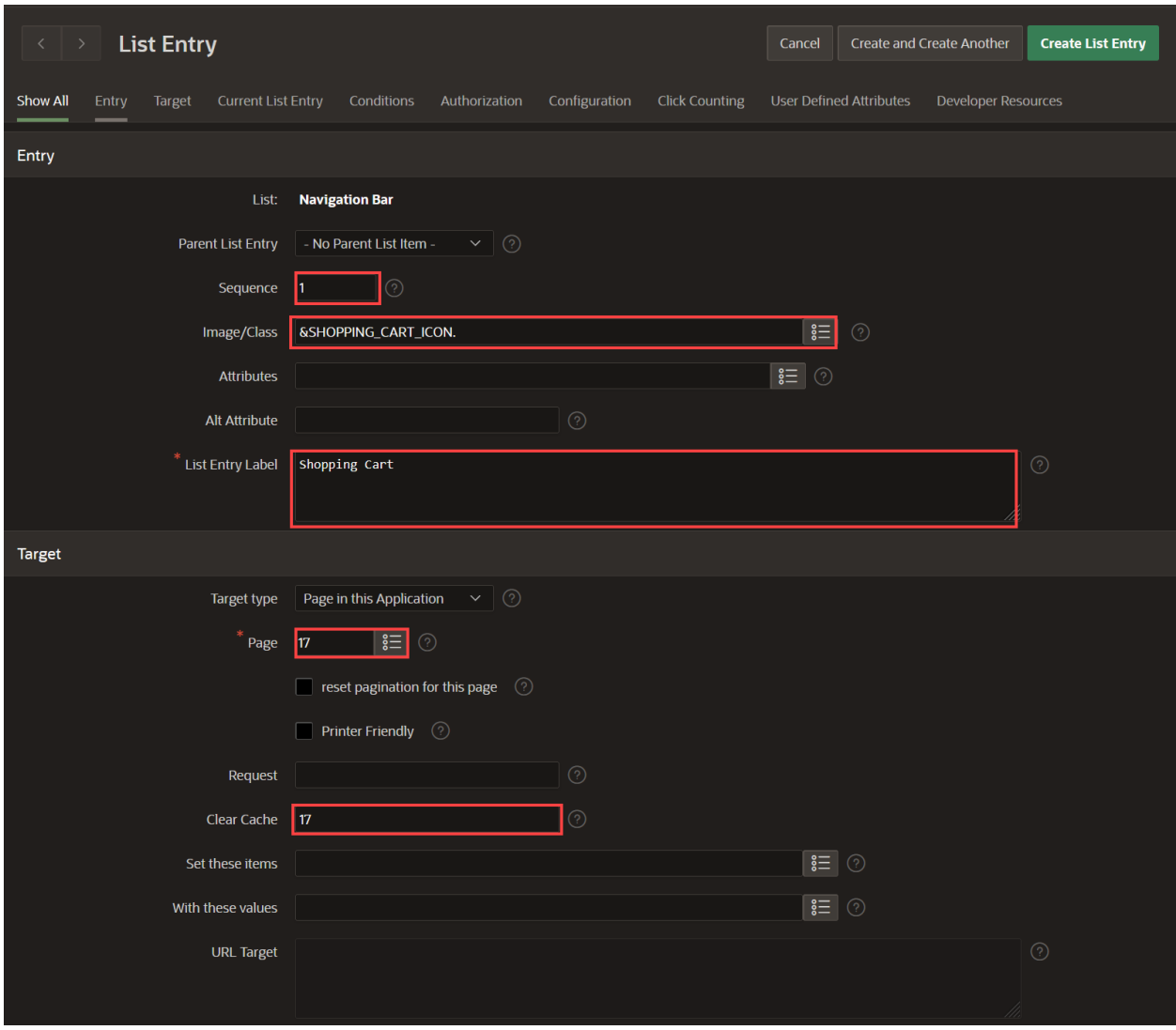


4. Click **Create Entry** and enter the following:



- For Sequence - enter **1**
- For Image/Class - **&SHOPPING_CART_ICON.**
- For List Entry Label - **Shopping Cart**
- For Page - select **17**

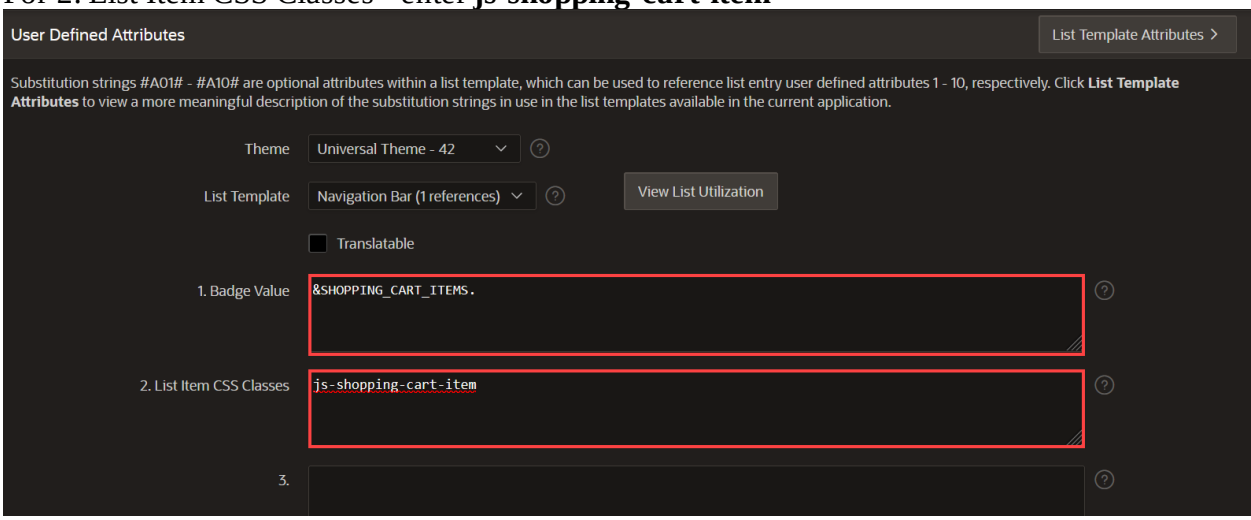
- For Clear cache - enter **17**



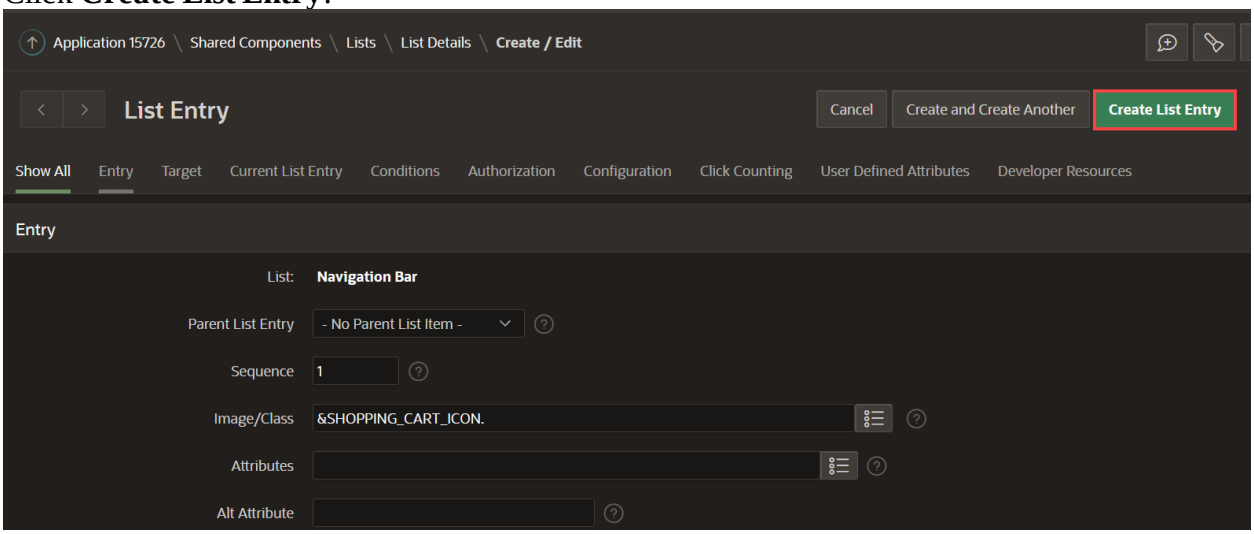
5. Scroll under **User Defined Attributes** and enter the following:

   - For 1. Badge Value - enter **&SHOPPING_CART_ITEMS.**

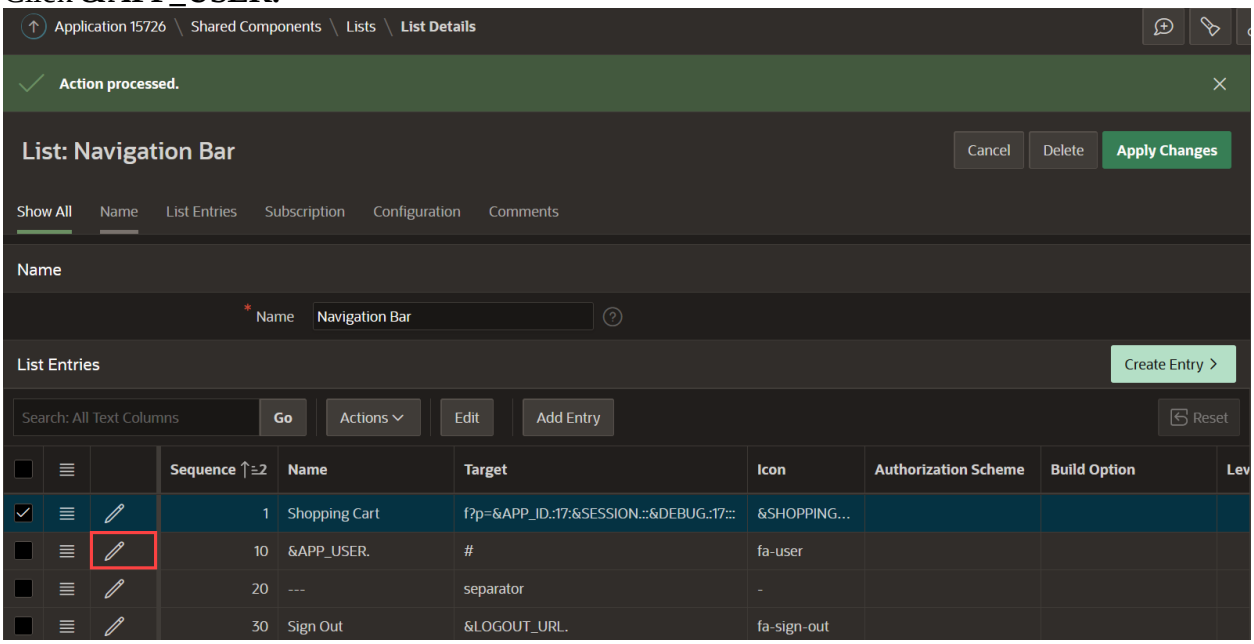
- For 2. List Item CSS Classes - enter **js-shopping-cart-item**



6. Click **Create List Entry**.

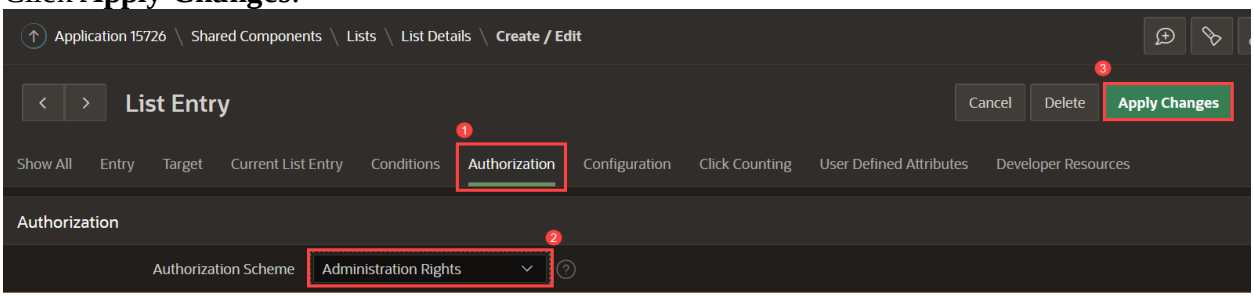


7. Click **&APP_USER.**



8. Under Authorization, for Authorization Scheme, select **Administration Rights**.

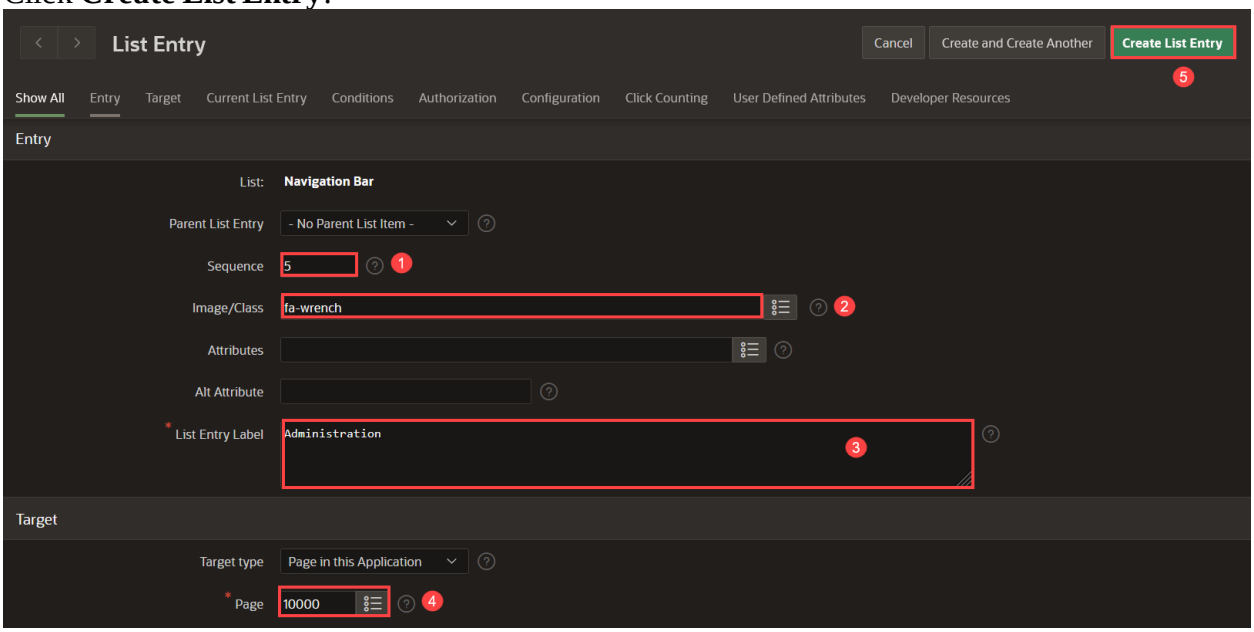
9. Click **Apply Changes**.



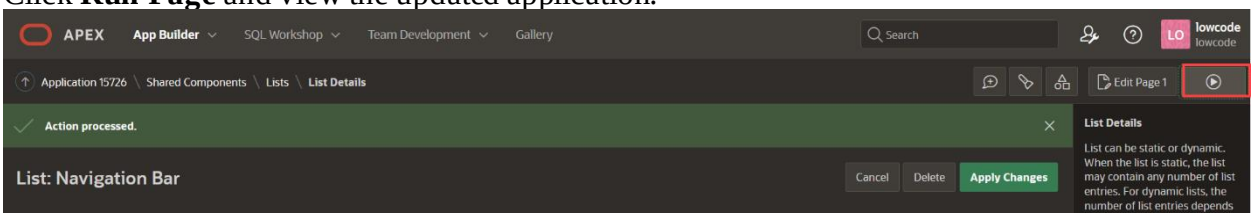
10. Click **Create Entry** and enter the following:

    - For Sequence - enter **5**
    - For Image/Class - enter **fa-wrench**
    - For List Entry Label - enter **Administration**
    - For Page - select **10000**

11. Click **Create List Entry**.



12. Click **Run Page** and view the updated application.



You now know how to enhance and maintain both navigation menu and navigation bar.