

Contents

| | |
|---|----|
| Install a sample dataset | 4 |
| Introduction | 4 |
| Objectives | 4 |
| Task 1: Create Customer Orders Tables | 4 |
| Task 2: Review Database Objects | 7 |
| Add columns to the products table..... | 7 |
| Introduction | 7 |
| Objectives | 8 |
| Task 1: Add Columns to the Products Table | 8 |
| Task 2: Populate the new columns | 10 |
| Task 3: Create Lookup Tables | 15 |
| Create a Database Package for Business Logic..... | 22 |
| Introduction | 22 |
| Objectives | 22 |
| Task 1: Create the Package | 22 |
| Create the Application..... | 35 |
| Create the application | 35 |
| Introduction | 35 |
| Objectives | 35 |
| Task 1: Create an Application | 35 |
| Task 2: Add a Name and Icon to the Application | 36 |
| Task 3: Add the Dashboard Page | 38 |
| Task 4: Add the Products Page | 44 |
| Task 5: Delete Original Home Page | 46 |
| Task 6: Add Multiple Reports | 47 |
| Task 7: Set Multiple Reports as Administration Pages | 49 |
| Task 8: Add Manage Products Page | 51 |
| Task 9: Set Features | 53 |
| Task 10: Finish Creating the Application | 54 |
| Task 11: Run the Application | 56 |
| Create the order page | 58 |

| | |
|--|-----|
| Introduction..... | 58 |
| Objectives..... | 59 |
| Task 1: Create a Normal Page - Order Information | 59 |
| Task 2: Add a Region..... | 61 |
| Task 3: Add Items to the Page | 62 |
| Task 4: Add Static Content Region | 64 |
| Task 5: Add Order Details Region | 66 |
| Task 6: Add Items Region..... | 70 |
| Create the shopping cart page..... | 76 |
| Introduction..... | 76 |
| Objectives..... | 77 |
| Task 1: Create Application Items..... | 78 |
| Task 2: Create Application Process..... | 79 |
| Task 3: Create a Normal Page - Shopping Cart..... | 82 |
| Task 4: Add a Cards Region..... | 84 |
| Task 5: Add an Action to the Shopping Cart..... | 91 |
| Task 6: Add Items and Buttons to the Page..... | 94 |
| Task 7: Add Validations to the Page..... | 100 |
| Task 8: Add Process to Create the Order..... | 104 |
| Task 9: Add Process to Clear the Shopping Cart..... | 107 |
| Task 10: Add Branches to the Page..... | 109 |
| Task 11: Add Dynamic Actions..... | 111 |
| Task 12: Format Products Image Size..... | 121 |
| Create the add to cart page..... | 122 |
| Introduction..... | 122 |
| Objectives..... | 123 |
| Task 1: Create a Modal Page..... | 123 |
| Task 2: Add Cards Region for Product Details..... | 124 |
| Task 3: Add Cards Region for Customer Reviews..... | 132 |
| Task 4: Add Items and Buttons..... | 139 |
| Task 5: Add Computation to Calculate the Number of Items for a Product..... | 146 |
| Task 6: Add Process to Add Products to the Shopping Cart..... | 148 |
| Task 7: Add Process to Edit Products in the Shopping Cart..... | 152 |

| | |
|---|-----|
| Task 8: Add Process to Delete Products from the Shopping Cart..... | 155 |
| Task 9: Add Process to Calculate the Shopping Cart Items..... | 157 |
| Task 10: Add Process to Close the Modal Page | 157 |
| Task 11: Enhance the Modal Page | 158 |
| Improve the products page | 160 |
| Introduction..... | 160 |
| Objectives..... | 161 |
| Task 1: Reorder Facets | 161 |
| Task 2: Enhance the Faceted Search | 164 |
| Task 3: Enhance the Cards Region..... | 165 |
| Task 4: Create Actions..... | 171 |
| Task 6: Add Dynamic Actions..... | 173 |
| Improve the application..... | 182 |
| Introduction..... | 182 |
| Objectives..... | 182 |
| Task 1: Set Pages to be Public..... | 183 |
| Task 2: Clean the Navigation Menu | 184 |
| Task 3: Enhance the Navigation Bar List..... | 185 |

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

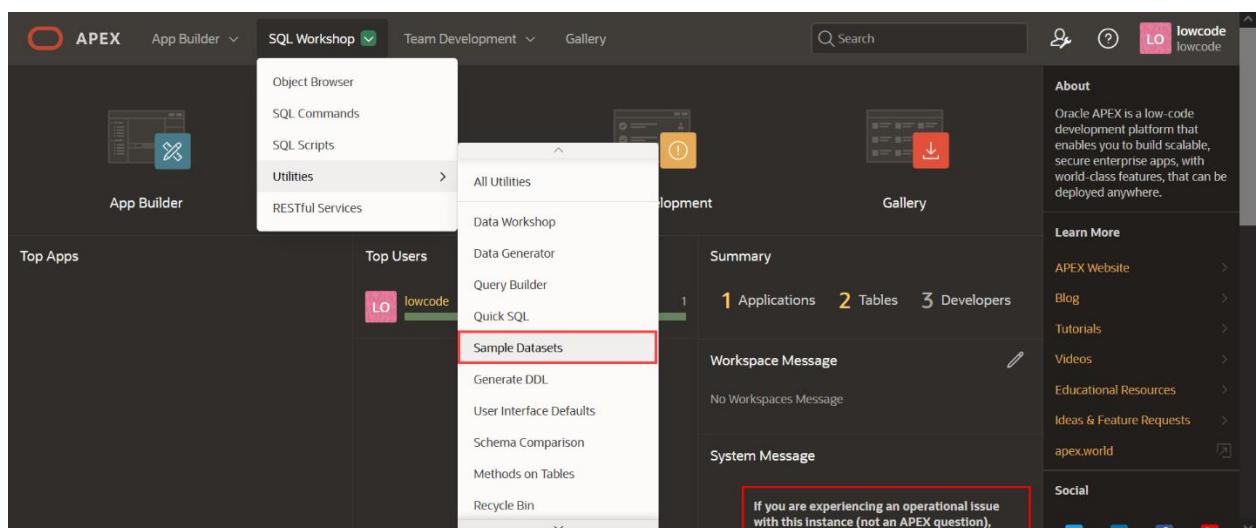
In this lab, you will:

- 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**.

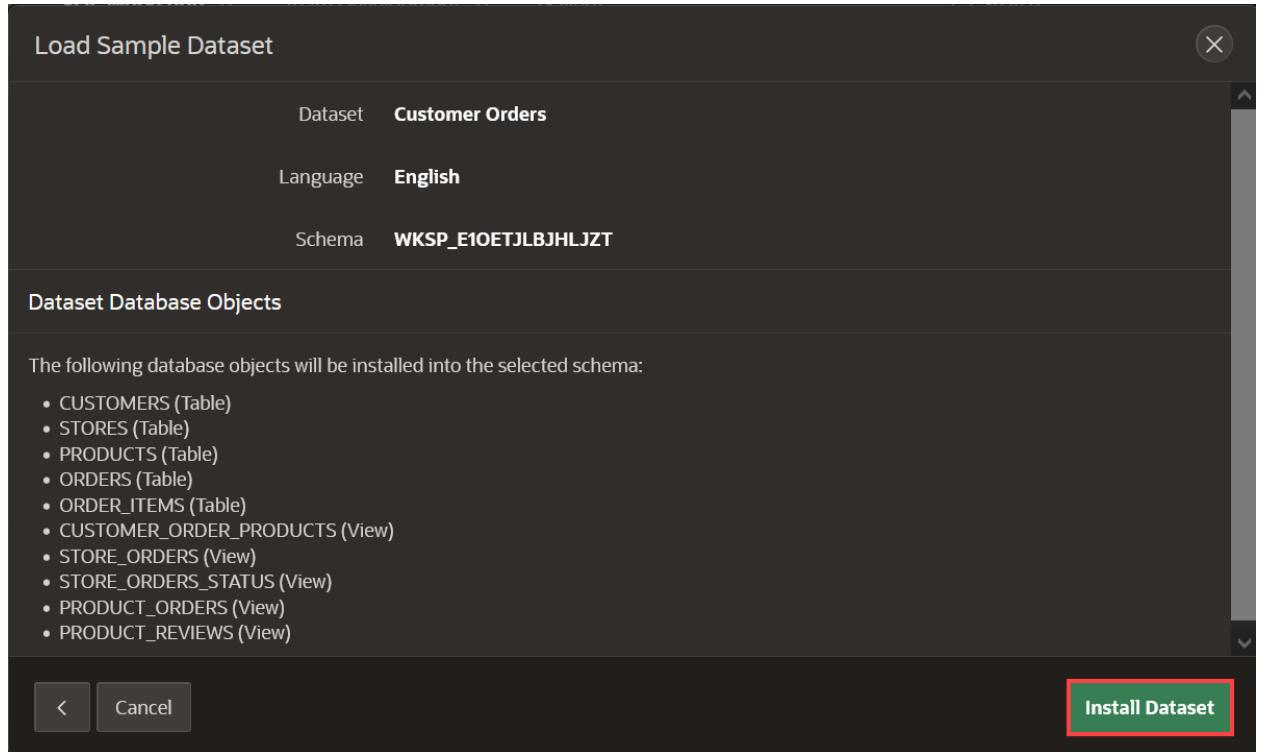
| Sample Datasets | | | | | | | About Sample Datasets |
|-----------------|-----------------|---|--|--------|----------------|-------------------|--|
| Action | Name | Languages | Description | Schema | Date Installed | Refresh Available | |
| Install | Countries | English | Listing of countries, population, and capital. | | | | |
| Install | Customer Orders | English | A collection of customers, stores, products, and orders. This dataset includes JSON data for the product description, and longitude / latitude for the stores. | | | | |
| Install | EMP / DEPT | English, Chinese, Czech, French, German, Japanese, Korean, Polish, Russian, Spanish | The generic EMP and DEPT tables. | | | | Each dataset includes various database objects and sample data. The sample data is available in one or more languages. |
| Install | HR Data | English | The generic HR tables commonly used by Oracle Education. | | | | |
| Install | Health Updates | English | Simplified patient-reported health status updates for health providers. | | | | Note: Only a single language can be loaded into the sample database objects. If you select a different language the currently loaded data will be replaced. |
| Install | Project Data | English | A collection of projects, milestones, tasks, and more. This dataset includes master-detail-detail relationships and a useful view for charting. | | | | |

4. Click Next.

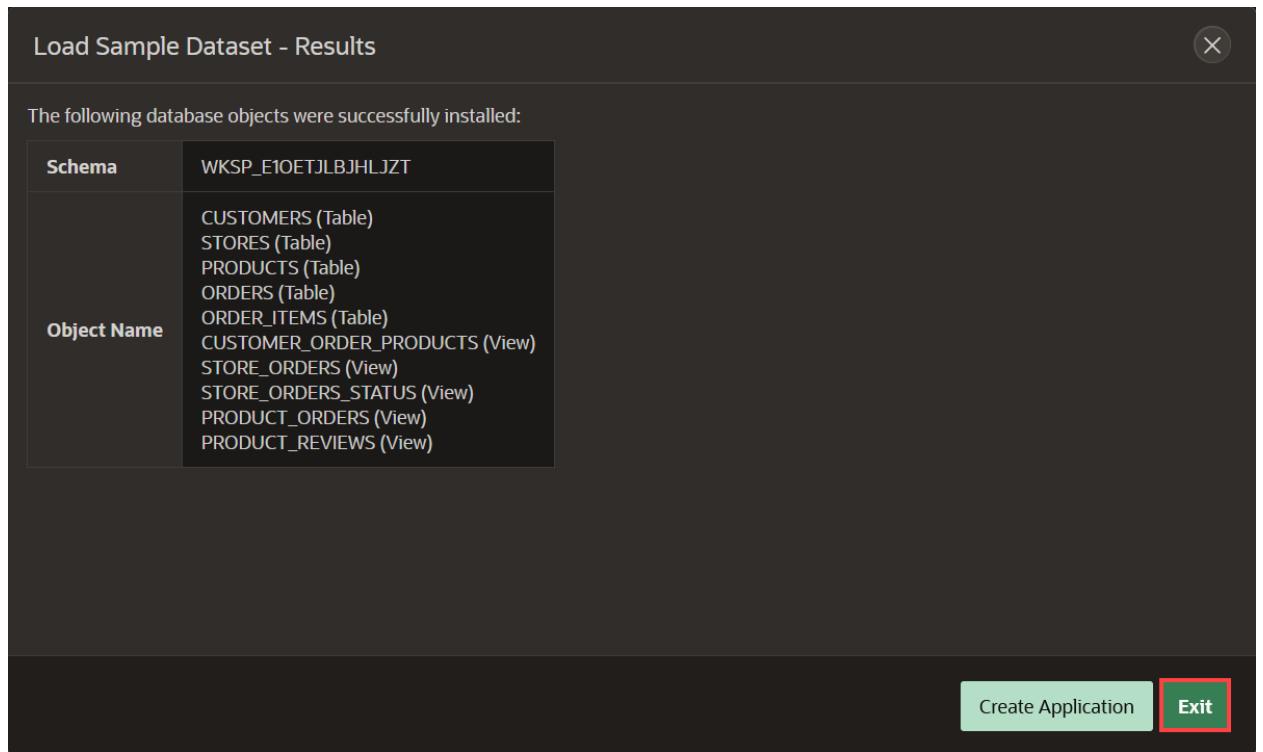
The schema name defaults to your current schema so will be different from the schema name shown below.

| Manage Sample Dataset | |
|------------------------|---|
| Dataset | Customer Orders |
| Description | A collection of customers, stores, products, and orders. This dataset includes JSON data for the product description, and longitude / latitude for the stores. |
| Change History | Installed during Oracle APEX installation. |
| Dataset Last Updated | 09-APR-2022 |
| Install Dataset | |
| Language | English |
| Schema | WKSP_E1OETJLBJHLJZT |
| Cancel | Next |

5. Click Install Dataset.



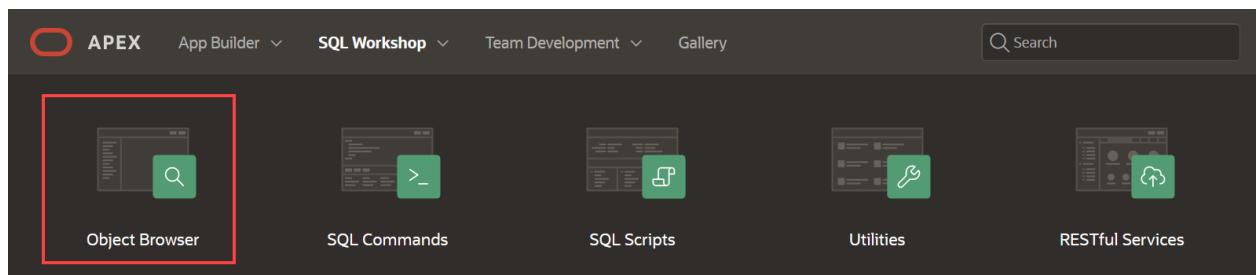
6. Click **Exit**.



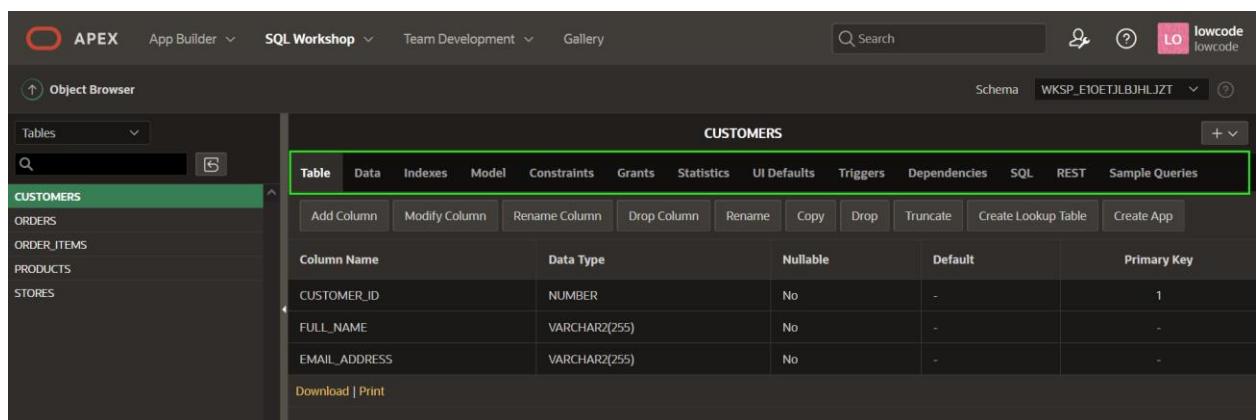
Note: You do NOT want to click Create Application, as you will manually create an application later.

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

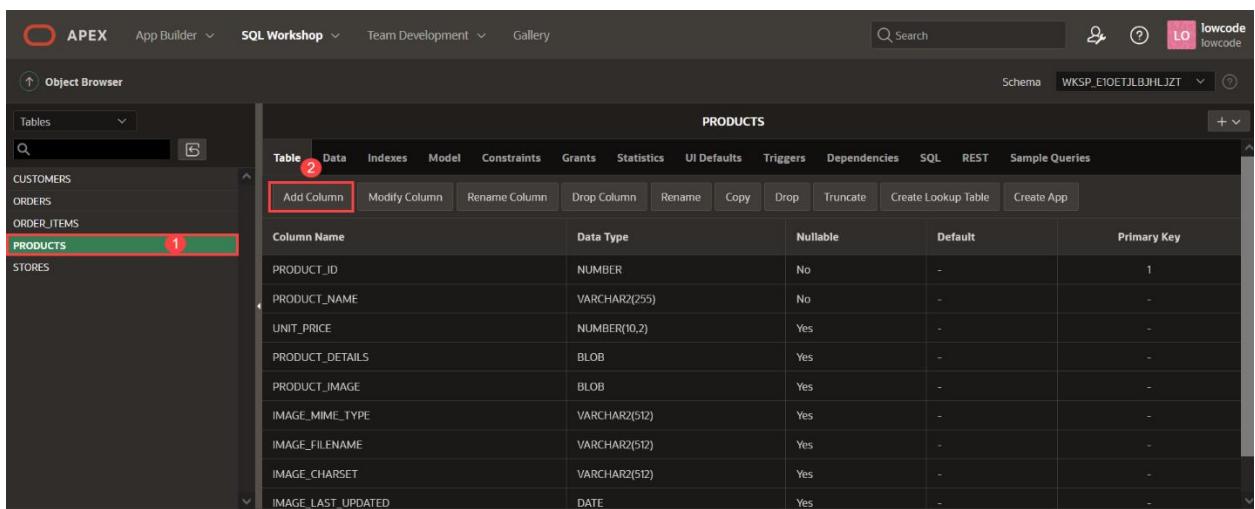
In this lab, you will:

- 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.

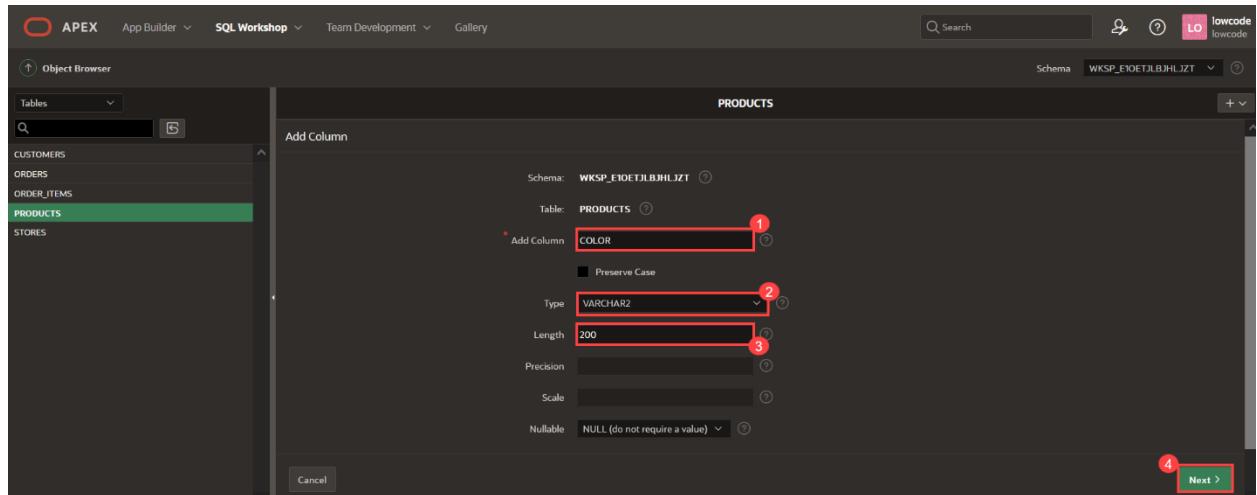


The screenshot shows the Oracle SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop' (which is currently selected), 'Team Development', and 'Gallery'. A search bar and a 'Lowcode' button are also present. The main area is titled 'PRODUCTS'. On the left, there's an 'Object Browser' pane with a 'Tables' dropdown and a list of tables: CUSTOMERS, ORDERS, ORDER_ITEMS, and PRODUCTS (which is selected and highlighted with a green box). The main panel displays the 'PRODUCTS' table structure with the following columns:

| Column Name | Data Type | Nullable | Default | Primary Key |
|--------------------|---------------|----------|---------|-------------|
| PRODUCT_ID | NUMBER | No | - | 1 |
| PRODUCT_NAME | VARCHAR2(255) | No | - | - |
| UNIT_PRICE | NUMBER(10,2) | Yes | - | - |
| PRODUCT_DETAILS | BLOB | Yes | - | - |
| PRODUCT_IMAGE | BLOB | Yes | - | - |
| IMAGE_MIME_TYPE | VARCHAR2(512) | Yes | - | - |
| IMAGE_FILENAME | VARCHAR2(512) | Yes | - | - |
| IMAGE_CHARSET | VARCHAR2(512) | Yes | - | - |
| IMAGE_LAST_UPDATED | DATE | Yes | - | - |

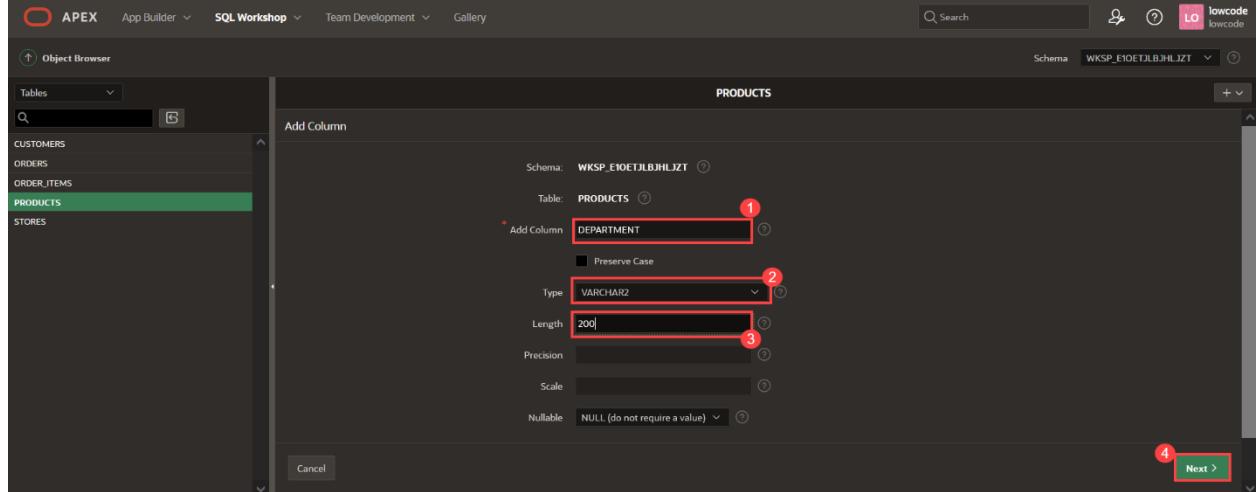
4. For **Color** column, enter the following:
 - Add Column - enter **COLOR**
 - Type - select **VARCHAR2**
 - Length - enter **200**.

Click **Next**.



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

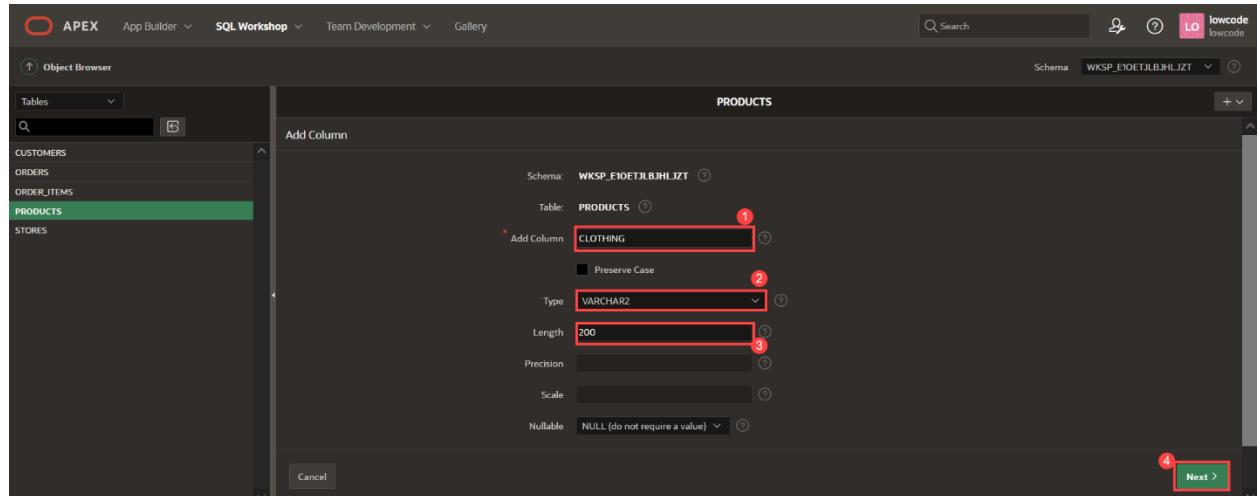
Click **Next**.



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**.

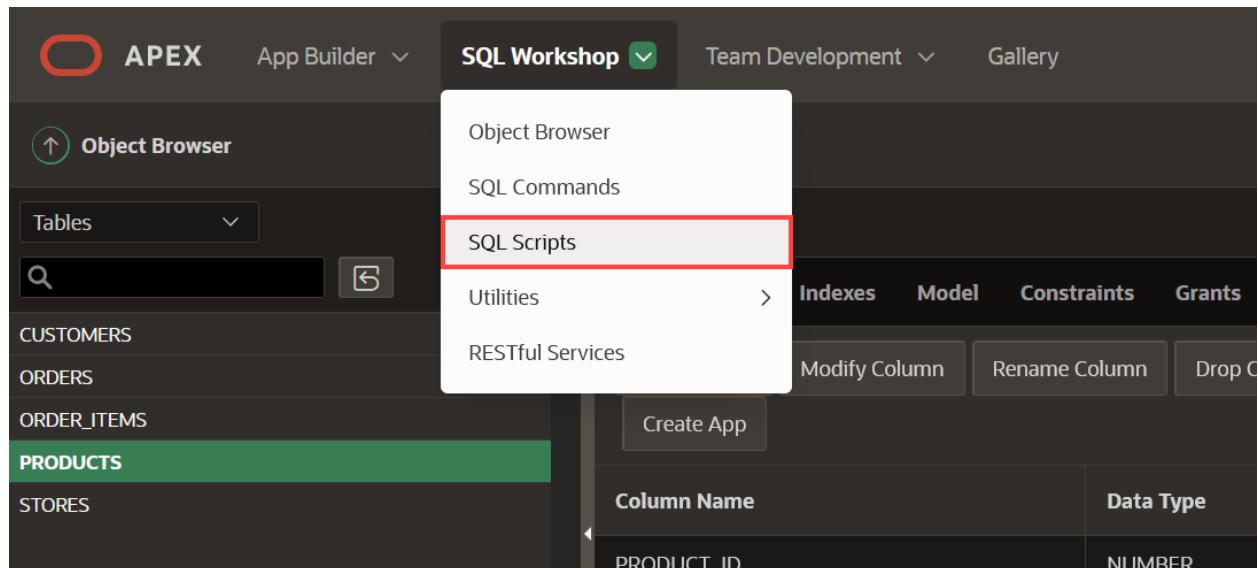
Click **Next**.



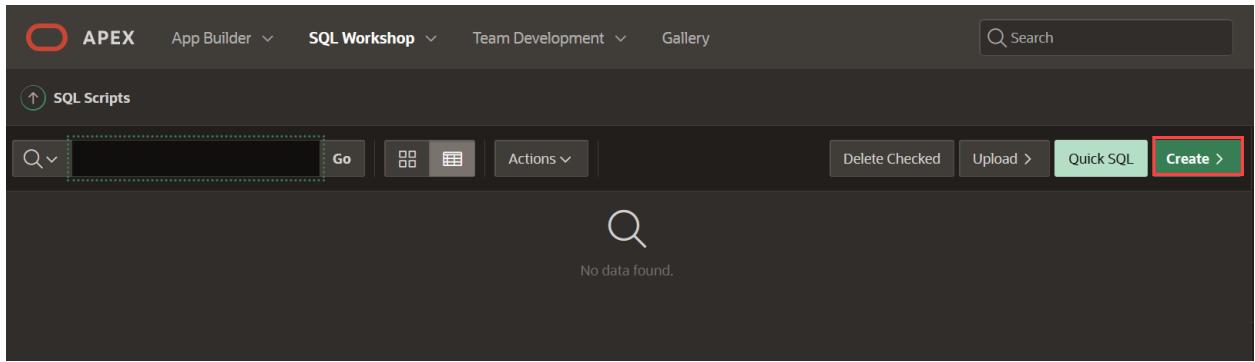
11. Click **Finish**.

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,  
15.           Instr(product_name, ' ',1,1)+1, Instr(product_name, ' ',1,  
16.           2)+1 - Instr(product_name, ' ',1,1)- 2),  
17.           p.color =
```

```

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

```

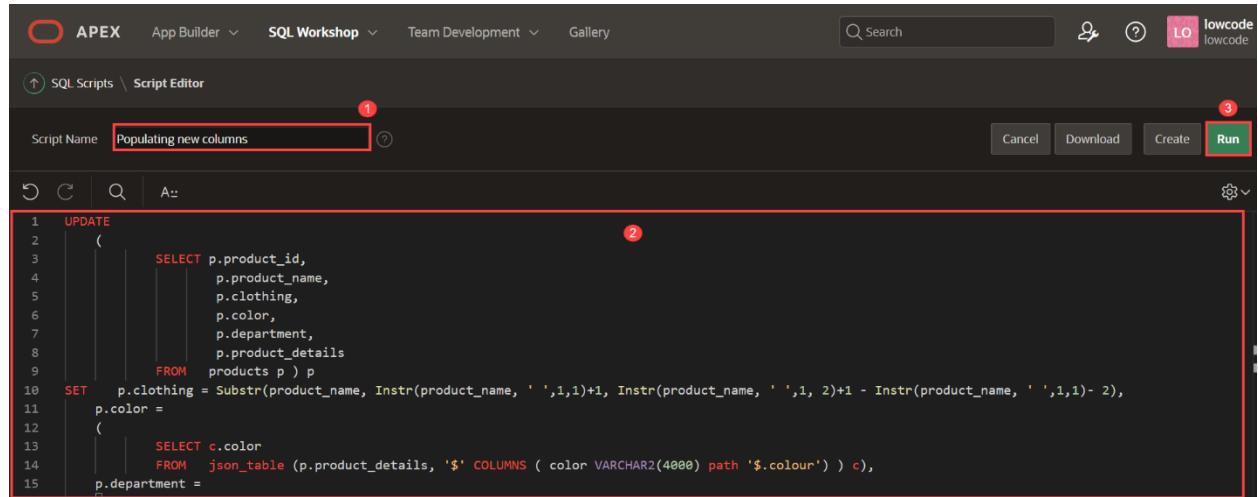
```

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

```

This script inserts the unique product type values (e.g. Shirt, Jacket, Skirt, etc.) into the CLOTHING column in the **Products** table. Similarly, 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.

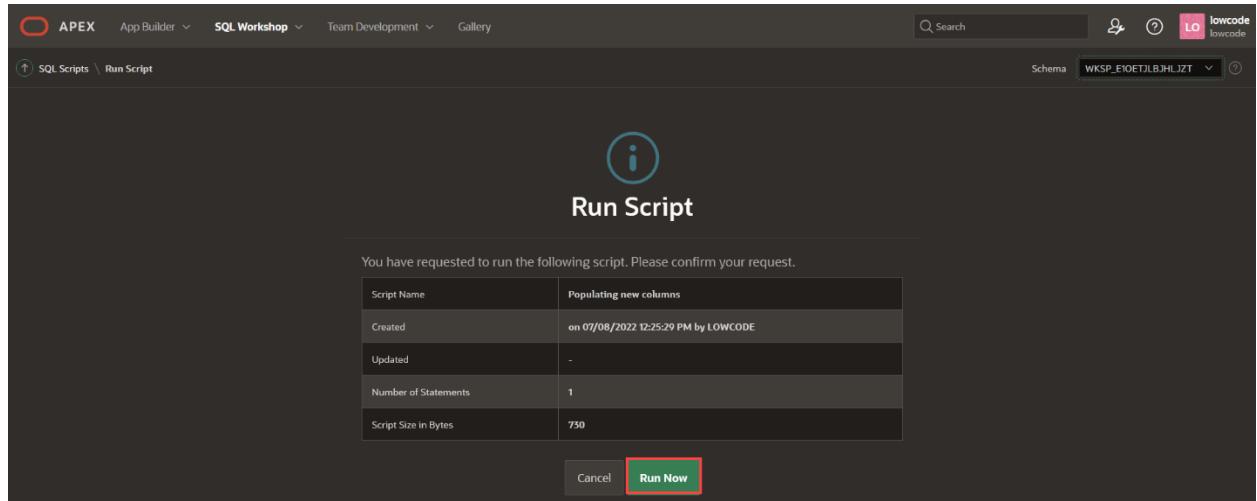


```

APEX App Builder SQL Workshop Team Development Gallery Search
SQL Scripts \ Script Editor
Script Name Populating new columns
Cancel Download Create Run
1 UPDATE
2 (
3     SELECT p.product_id,
4         p.product_name,
5         p.clothing,
6         p.color,
7         p.department,
8         p.product_details
9     FROM products p ) p
10 SET     p.clothing = Substr(product_name, Instr(product_name, ' ',1,1)+1, Instr(product_name, ' ',1, 2)+1 - Instr(product_name, ' ',1,1)- 2),
11     p.color =
12     (
13         SELECT c.color
14         FROM json_table (p.product_details, '$' COLUMNs ( color VARCHAR2(4000) path '$.colour' ) c),
15     p.department =

```

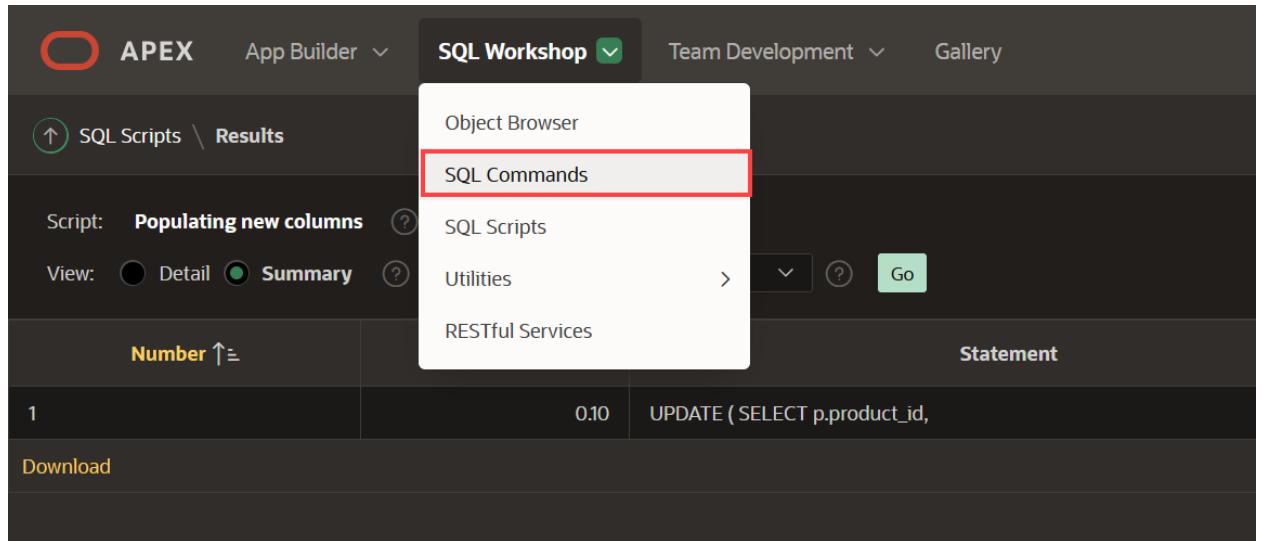
22. Click Run Now.



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

| Script: Populating new columns Status: Complete | | | | | Create App | Edit Script |
|---|---------|-------------------------------|--------------------|-------------|----------------------------|-----------------------------|
| Number ↑ | Elapsed | Statement | Feedback | Rows | | |
| 1 | 0.10 | UPDATE (SELECT p.product_id, | 46 row(s) updated. | 46 | | |
| Download | | | | | | |
| 1 | | 1 | | 0 | | row(s) 1 - 1 of 1 |
| Statements Processed | | Successful | | 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;
```

The screenshot shows the Oracle SQL Workshop interface. In the top-left, there's a code editor with a red box labeled '1' containing the following SQL query:

```

1  SELECT p.product_name,
2      p.unit_price,
3      p.color,
4      p.department,
5      p.clothing
6  FROM products p;

```

In the top-right, there are 'Save' and 'Run' buttons, with 'Run' highlighted by a red box labeled '2'. Below the code editor is a navigation bar with tabs: 'Results' (highlighted with a green box labeled '3'), 'Explain', 'Describe', 'Saved SQL', and 'History'.

The main area displays the results of the query in a table:

| PRODUCT_NAME | UNIT_PRICE | COLOR | DEPARTMENT | CLOTHING |
|-------------------------|------------|-------|------------|----------|
| Women's Socks (Grey) | 39.89 | grey | Women's | Socks |
| Women's Sweater (Brown) | 24.46 | brown | Women's | Sweater |
| Women's Jacket (Black) | 14.34 | black | Women's | Jacket |
| Men's Coat (Red) | 28.21 | red | Men's | Coat |
| Girl's Shorts (Green) | 38.34 | green | Girl's | Shorts |
| Girl's Pyjamas (White) | 39.78 | black | Girl's | Pyjamas |
| Men's Shorts (Black) | 10.35 | black | Men's | Shorts |

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.

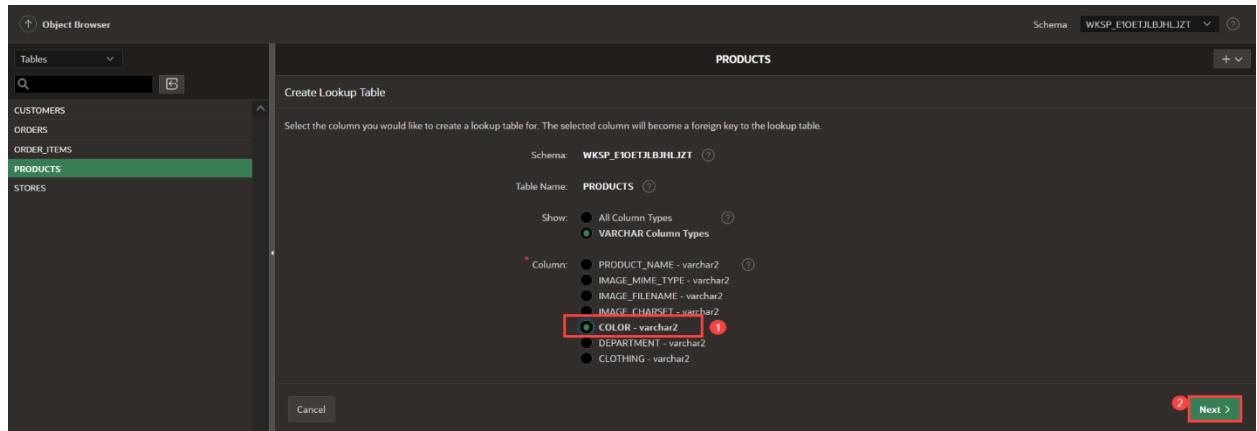
The screenshot shows the Oracle Object Browser interface. On the left, there's a sidebar with a 'Tables' dropdown and a search bar. Below it is a list of tables: CUSTOMERS, ORDERS, ORDER_ITEMS, PRODUCTS (which is selected and highlighted with a green box), and STORES.

The main area shows the 'PRODUCTS' table structure. At the top of the table view, there's a toolbar with various buttons, and the 'Create Lookup Table' button is highlighted with a red box labeled '1'.

The table structure is as follows:

| Column Name | Data Type | Nullable | Default | Primary Key |
|-----------------|---------------|----------|---------|-------------|
| PRODUCT_ID | NUMBER | No | - | 1 |
| PRODUCT_NAME | VARCHAR2(255) | No | - | - |
| UNIT_PRICE | NUMBER(10,2) | Yes | - | - |
| PRODUCT_DETAILS | BLOB | Yes | - | - |
| PRODUCT_IMAGE | BLOB | Yes | - | - |
| IMAGE_TYPE | VARCHAR2(50) | Yes | - | - |

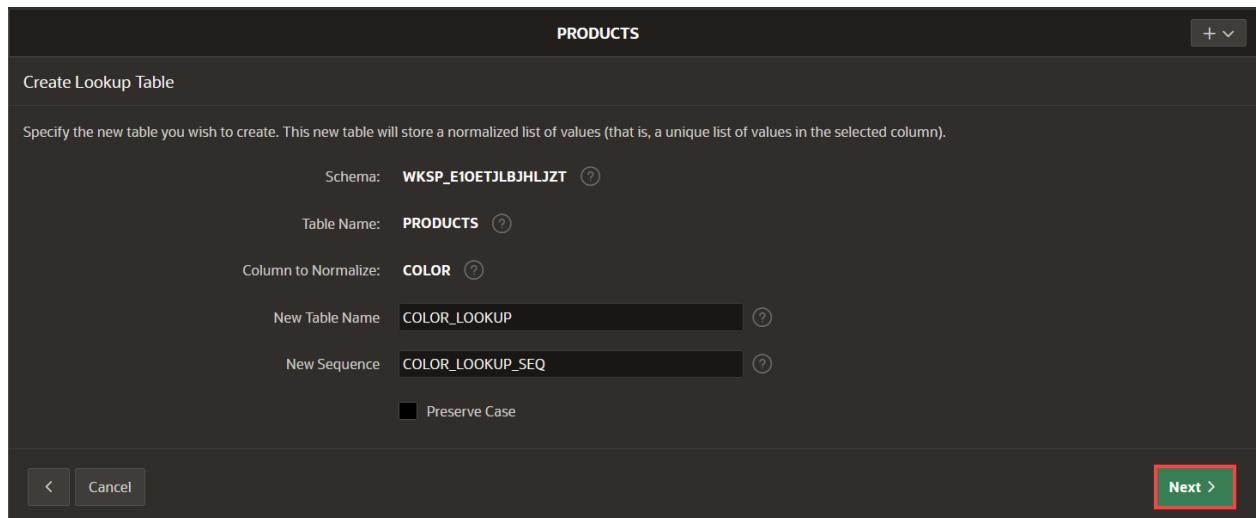
5. For Column, select **COLOR - varchar2**. Click **Next**.



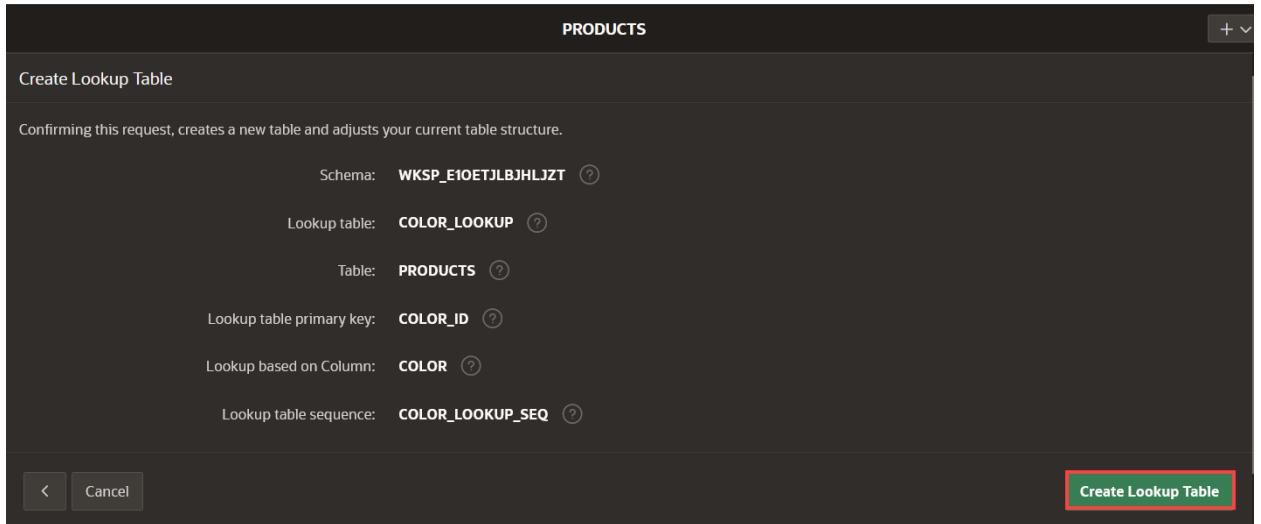
6. Leave the table and sequence name by default:

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

Click **Next**.



7. Click **Create Lookup Table**.

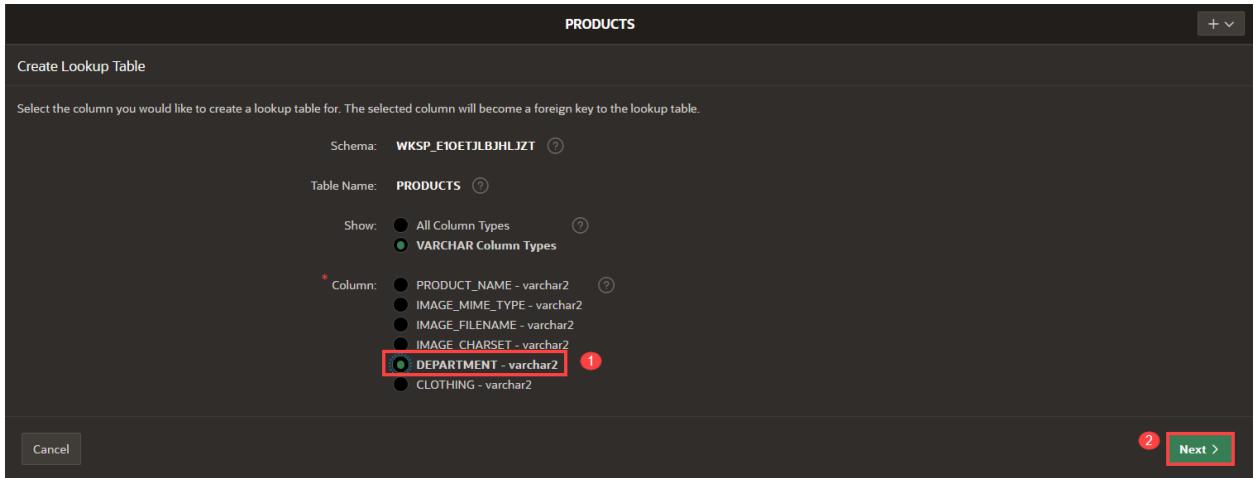


*Note: Click the **Create Lookup Table** button only once. Then you will find the new table listed in the Object Browser.*

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

| PRODUCTS | | | | | | | | | | + ▾ | | |
|--------------------|---------------|---------------|---------------|-------------|---------|------------|-------------|-------------|---------------------|------------|------|----------------|
| Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries |
| | Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table | Create App | | |
| Column Name | Data Type | | | Nullable | Default | | | Primary Key | | | | |
| PRODUCT_ID | NUMBER | | | No | - | | | 1 | | | | |
| PRODUCT_NAME | VARCHAR2(255) | | | No | - | | | - | | | | |
| UNIT_PRICE | NUMBER(10,2) | | | Yes | - | | | - | | | | |
| PRODUCT_DETAILS | BLOB | | | Yes | - | | | - | | | | |
| PRODUCT_IMAGE | BLOB | | | Yes | - | | | - | | | | |
| IMAGE_MIME_TYPE | VARCHAR2(512) | | | Yes | - | | | - | | | | |
| IMAGE_FILENAME | VARCHAR2(512) | | | Yes | - | | | - | | | | |
| IMAGE_CHARSET | VARCHAR2(512) | | | Yes | - | | | - | | | | |
| IMAGE_LAST_UPDATED | DATE | | | Yes | - | | | - | | | | |

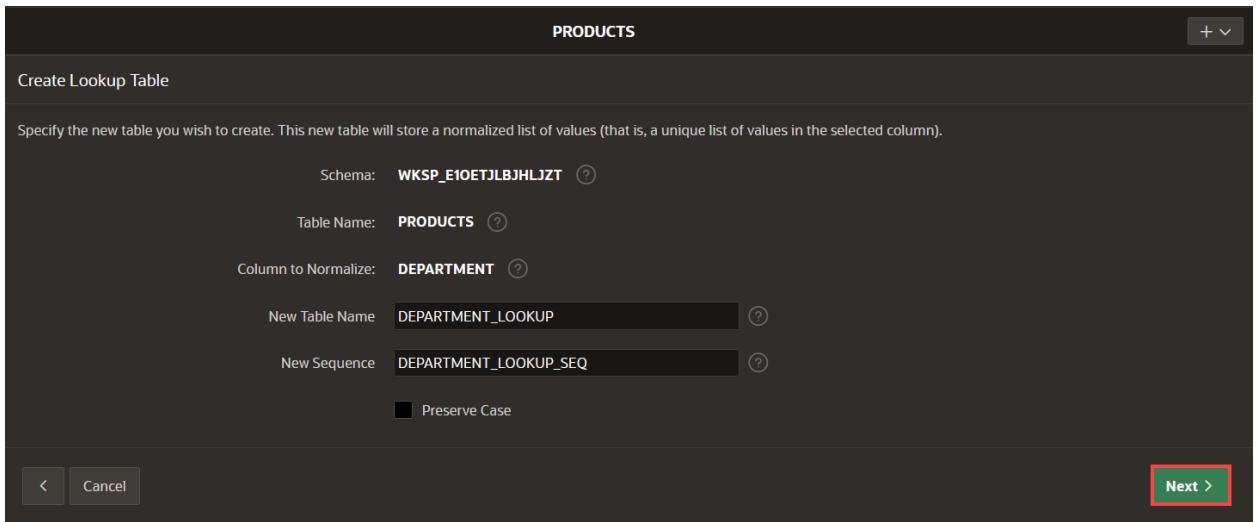
- For Column, select **DEPARTMENT - varchar2**. Click **Next**.



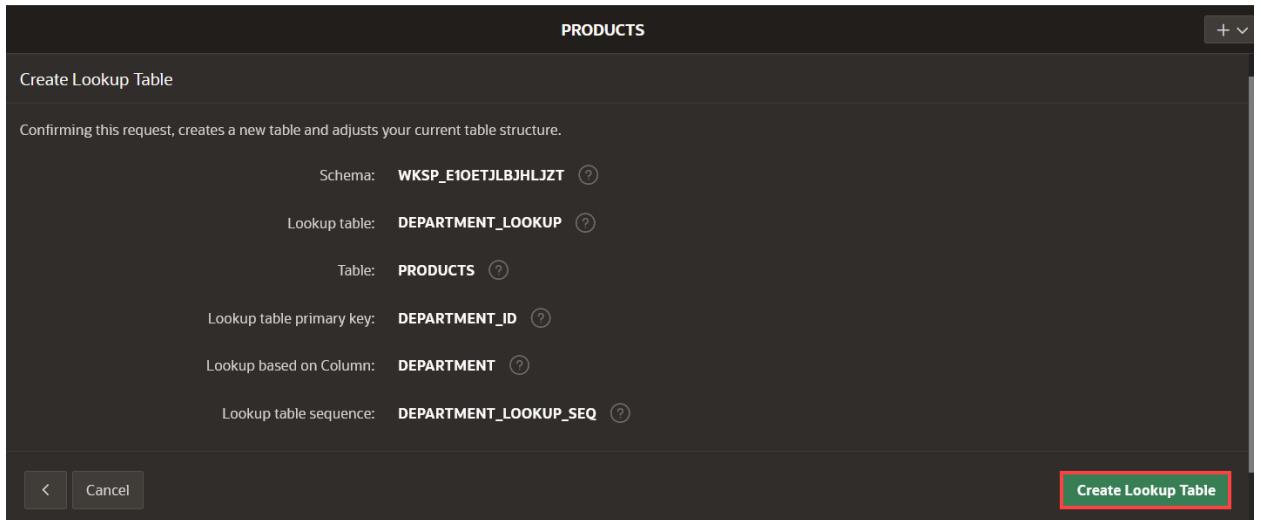
10. Leave the table and sequence name by default:

- o New Table Name: **DEPARTMENT_LOOKUP**
- o 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.

| PRODUCTS | | | | | | | | | | | | |
|--------------------|------------|---------------|------------------|---------------|--------|-----------------|-------------|----------------|---------------------|--------------------|------|----------------|
| Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries |
| | Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table | Create App | | |
| Column Name | | | Data Type | | | Nullable | | Default | | Primary Key | | |
| PRODUCT_ID | | | | NUMBER | | | No | - | - | 1 | | |
| PRODUCT_NAME | | | | VARCHAR2(255) | | | No | - | - | - | | |
| UNIT_PRICE | | | | NUMBER(10,2) | | | Yes | - | - | - | | |
| PRODUCT_DETAILS | | | | BLOB | | | Yes | - | - | - | | |
| PRODUCT_IMAGE | | | | BLOB | | | Yes | - | - | - | | |
| IMAGE_MIME_TYPE | | | | VARCHAR2(512) | | | Yes | - | - | - | | |
| IMAGE_FILENAME | | | | VARCHAR2(512) | | | Yes | - | - | - | | |
| IMAGE_CHARSET | | | | VARCHAR2(512) | | | Yes | - | - | - | | |
| IMAGE_LAST_UPDATED | | | | DATE | | | Yes | - | - | - | | |

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

PRODUCTS

Create Lookup Table

Select the column you would like to create a lookup table for. The selected column will become a foreign key to the lookup table.

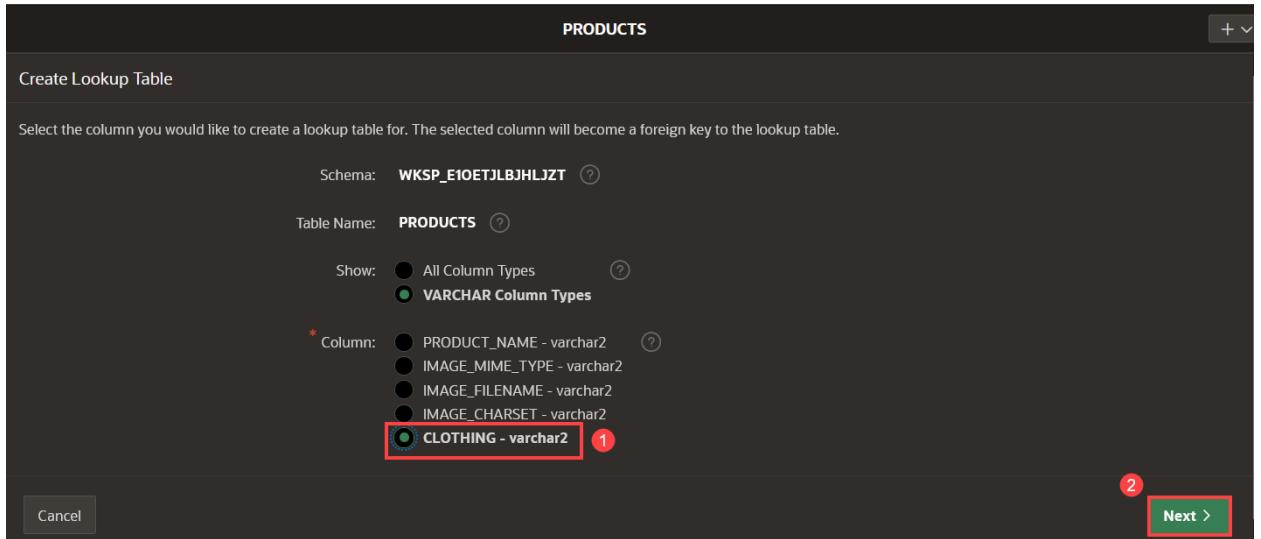
Schema: **WKSP_E1OETJLBJHLJZT** [?](#)

Table Name: **PRODUCTS** [?](#)

Show: All Column Types [?](#)
 VARCHAR Column Types [?](#)

* Column: **PRODUCT_NAME - varchar2** [?](#)
 IMAGE_MIME_TYPE - varchar2
 IMAGE_FILENAME - varchar2
 IMAGE_CHARSET - varchar2
 CLOTHING - varchar2 [?](#) **1**

[Cancel](#) [Next >](#) **2**



14. Leave the table and sequence name by default:

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

Click **Next**.

PRODUCTS

Create Lookup Table

Specify the new table you wish to create. This new table will store a normalized list of values (that is, a unique list of values in the selected column).

Schema: **WKSP_E1OETJLBJHLJZT** [?](#)

Table Name: **PRODUCTS** [?](#)

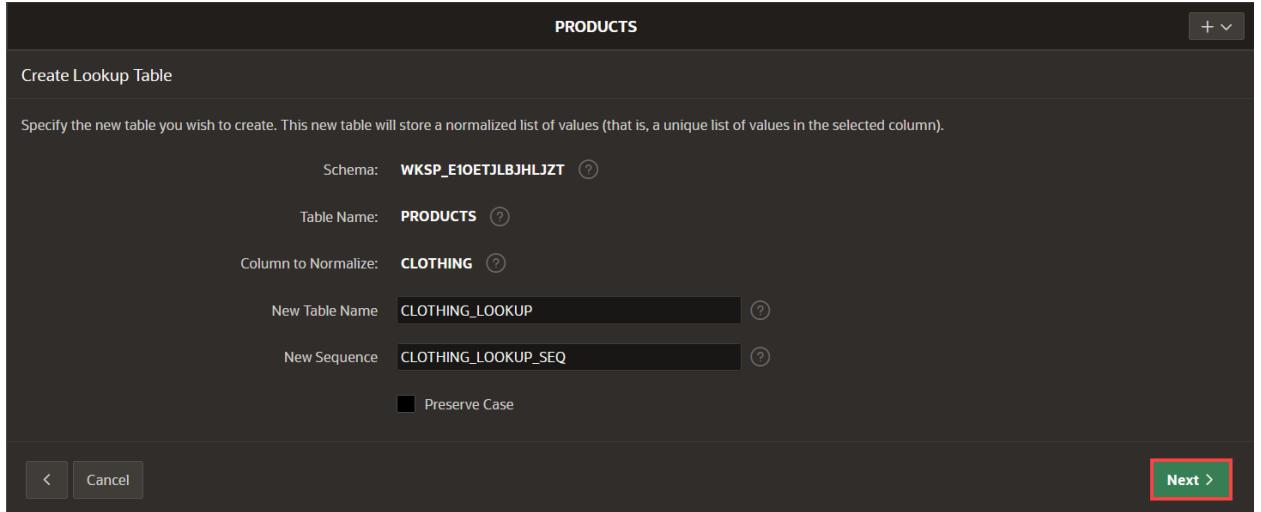
Column to Normalize: **CLOTHING** [?](#)

New Table Name: **CLOTHING_LOOKUP** [?](#)

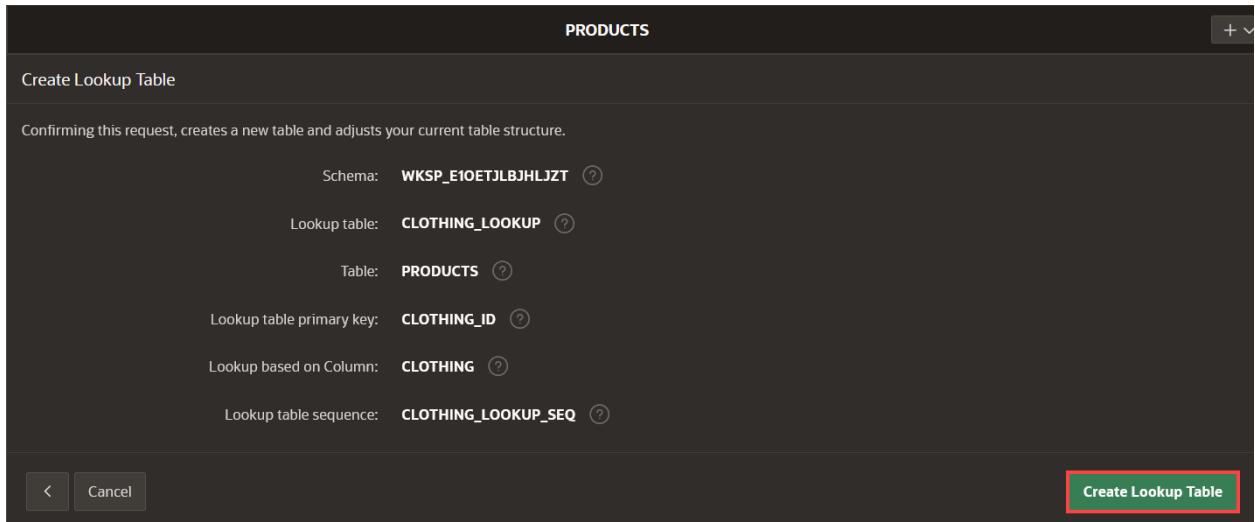
New Sequence: **CLOTHING_LOOKUP_SEQ** [?](#)

Preserve Case

[<](#) [Cancel](#) [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:

- o COLOR_LOOKUP
- o DEPARTMENT_LOOKUP
- o 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.

| Column Name | Data Type | Nullable | Default | Primary Key |
|--------------------|---------------|----------|---------|-------------|
| PRODUCT_ID | NUMBER | No | - | 1 |
| PRODUCT_NAME | VARCHAR2(255) | No | - | - |
| UNIT_PRICE | NUMBER(10,2) | Yes | - | - |
| PRODUCT_DETAILS | BLOB | Yes | - | - |
| PRODUCT_IMAGE | BLOB | Yes | - | - |
| IMAGE_MIME_TYPE | VARCHAR2(512) | Yes | - | - |
| IMAGE_FILENAME | VARCHAR2(512) | Yes | - | - |
| IMAGE_CHARSET | VARCHAR2(512) | Yes | - | - |
| IMAGE_LAST_UPDATED | DATE | Yes | - | - |
| COLOR_ID | NUMBER | Yes | - | - |
| DEPARTMENT_ID | NUMBER | Yes | - | - |
| CLOTHING_ID | NUMBER | Yes | - | - |

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:

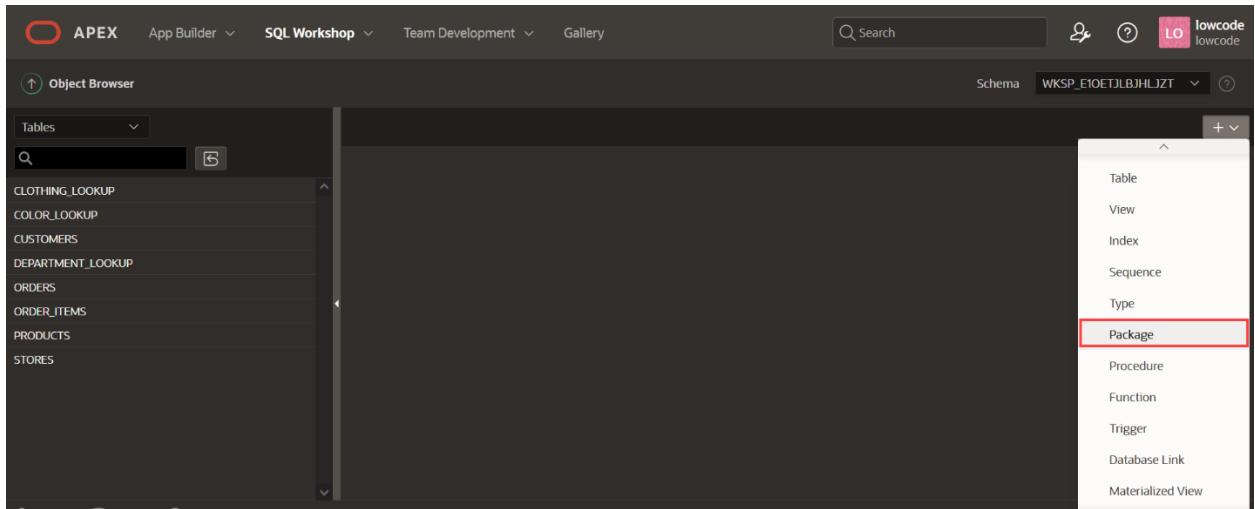
- Create a package to manage the Shopping Cart.

Collapse All Tasks

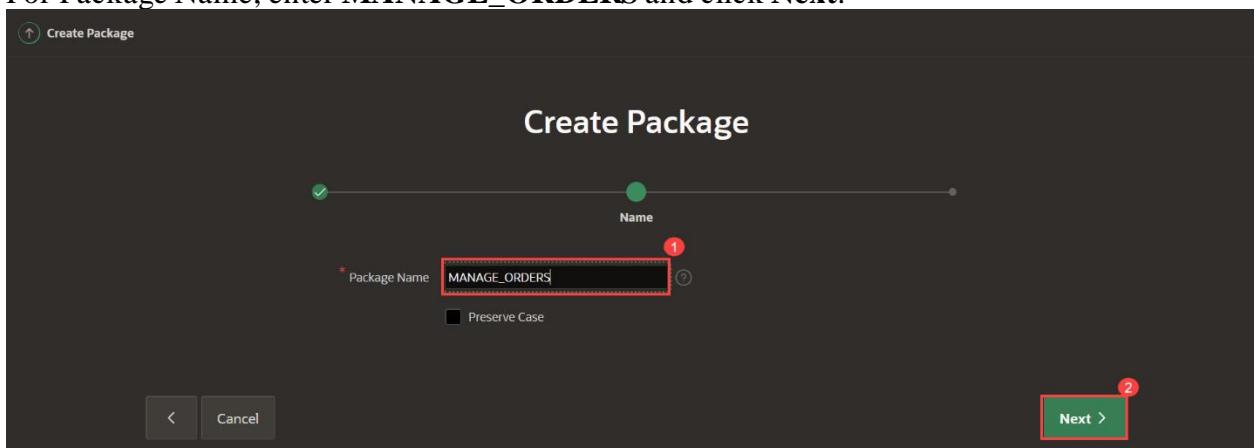
Task 1: Create the Package

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 (

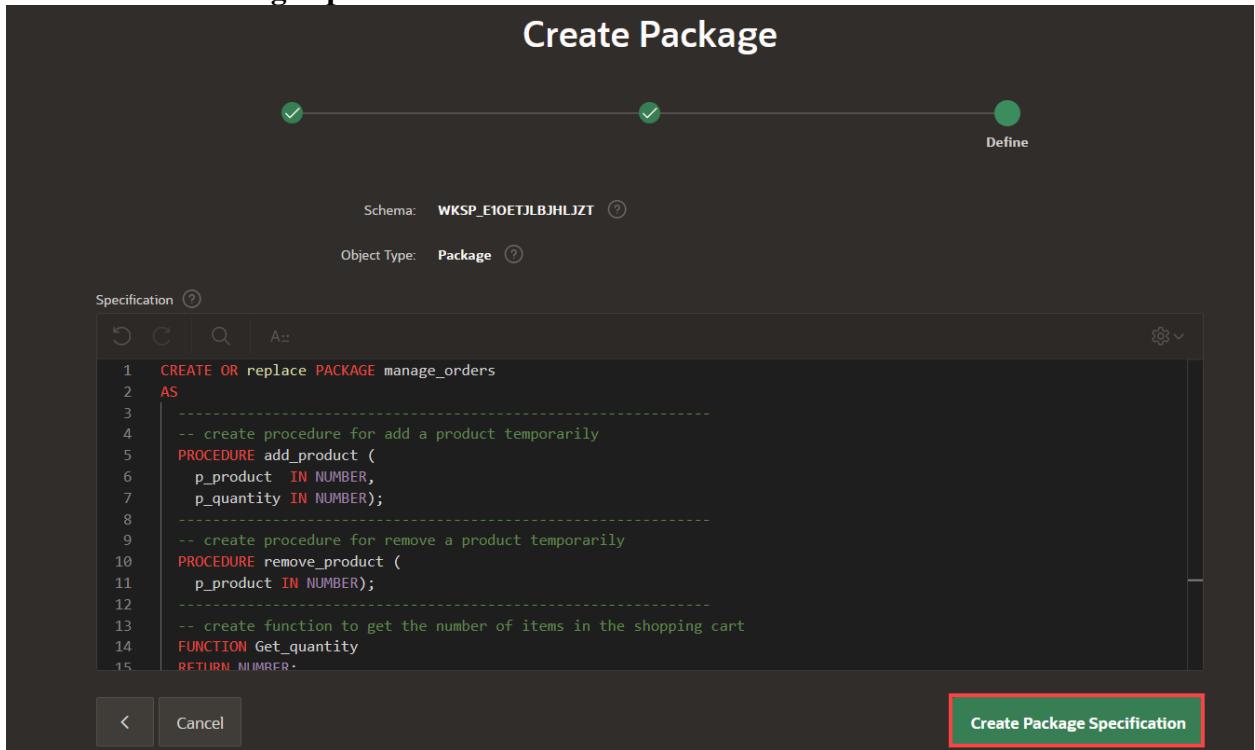
```

```
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. Copy **CREATE 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 =>
53.          'PRODUCTS') ;
54.
55.      apex_collection.add_member(p_collection_name =>
56.          'PRODUCTS',
57.          p_n001 => p_product,
58.          p_n002 => p_quantity) ;
59.
60.  PROCEDURE remove_product (p_product IN NUMBER)
61.  IS
62.      l_id NUMBER;
63.  BEGIN
64.      IF apex_collection.Collection_exists
65.          (p_collection_name => 'PRODUCTS')
66.      THEN
67.          SELECT seq_id
68.          INTO    l_id
69.          FROM    apex_collections a
70.          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,
                                p_customer_email IN
VARCHAR2,
                                p_store          IN
NUMBER,
                                p_order_id       OUT
orders.order_id%TYPE,
                                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 =>
189.      'PRODUCTS');

END create_order;

```

```
END manage_orders;
```

190. Click Save & Compile.

The screenshot shows the Oracle APEX developer interface for the 'MANAGE_ORDERS' package. The 'Body' tab is active (1). The 'Save & Compile' button is highlighted with a red box (2). The code editor displays the package specification and body (3).

```

CREATE OR replace PACKAGE BODY manage_orders
AS
    PROCEDURE add_product (p_product IN NUMBER,
                           p_quantity IN NUMBER)
    IS
        BEGIN
            IF NOT apex_collection.collection_exists (p_collection_name => 'PRODUCTS')
            THEN
                apex_collection.create_collection(p_collection_name => 'PRODUCTS');
            END IF;
            apex_collection.add_member(p_collection_name => 'PRODUCTS',
                                       p_n001 => p_product,
                                       p_n002 => p_quantity);
        END add_product;

    PROCEDURE remove_product (p_product IN NUMBER)
    IS
        l_id NUMBER;
    BEGIN
        IF apex_collection.Collection_exists (p_collection_name => 'PRODUCTS')
        THEN
            apex_collection.remove_member(p_collection_name => 'PRODUCTS',
                                         l_id);
        END IF;
    END remove_product;
END;

```

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

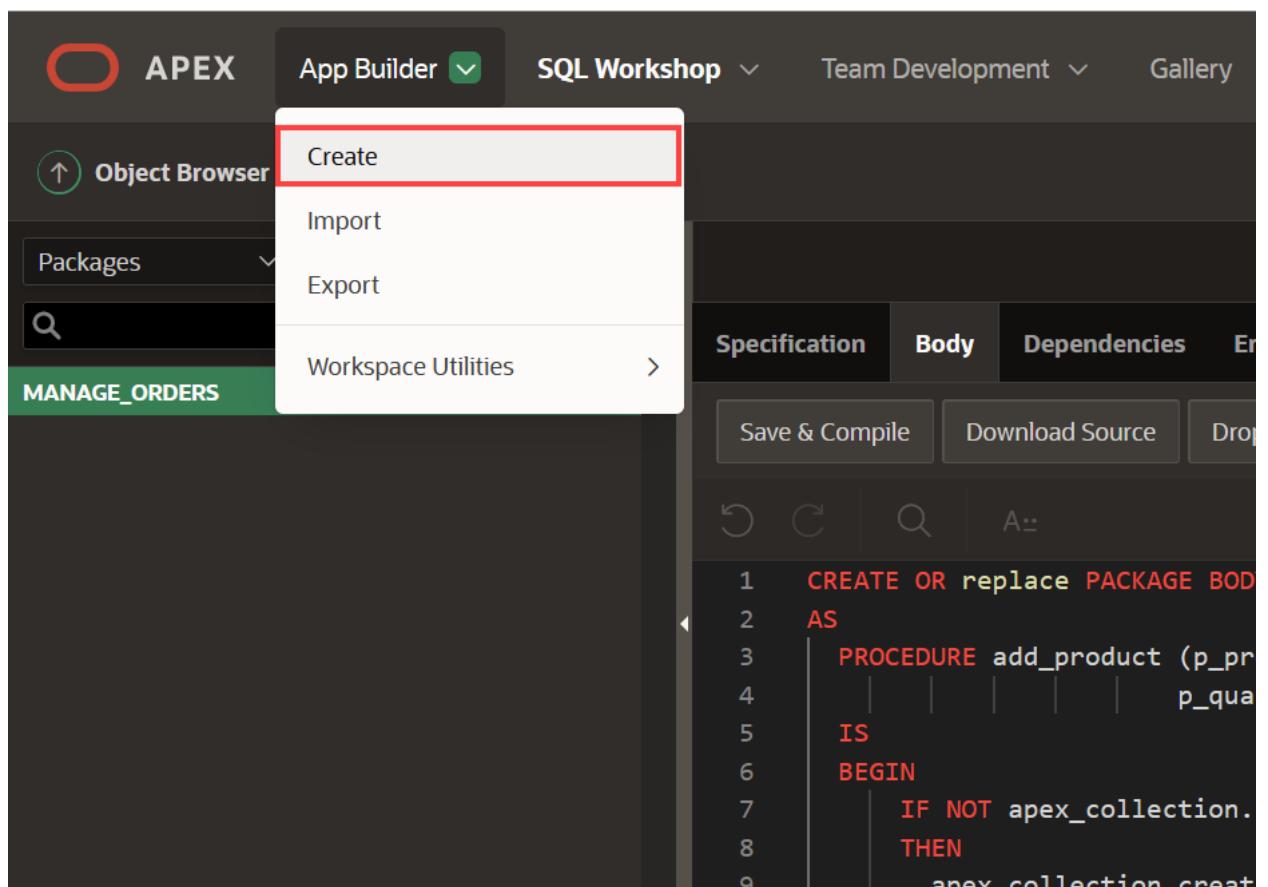
In this lab, you will:

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

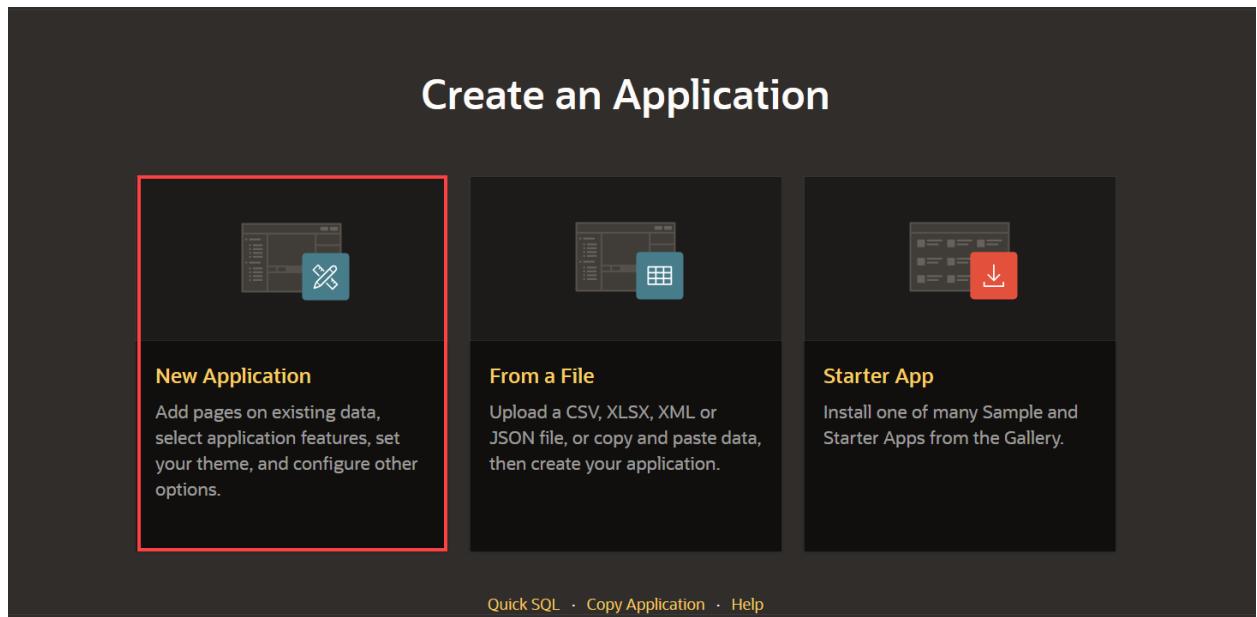
Collapse All Tasks

Task 1: Create an Application

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



2. Click **New Application**.

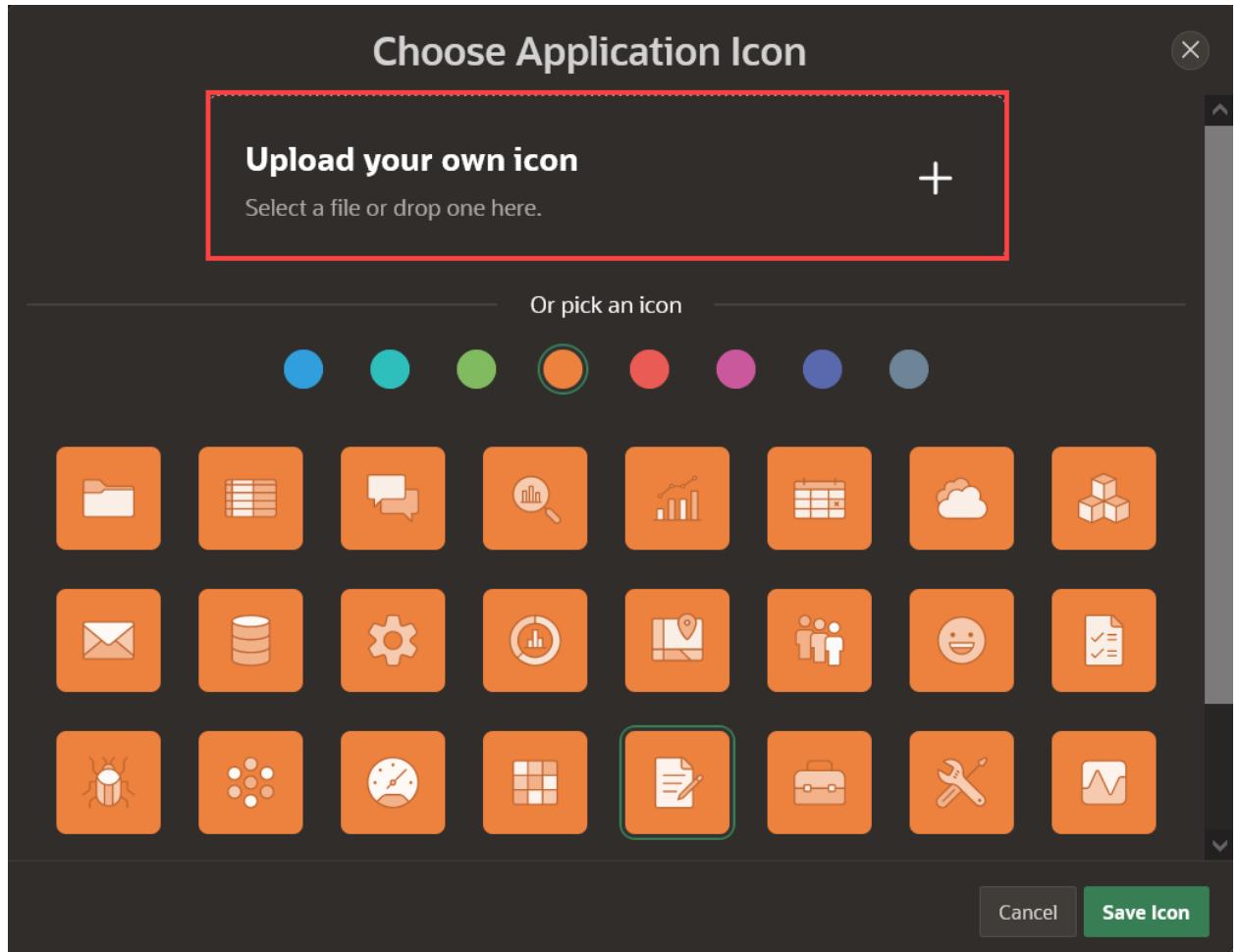


Task 2: Add a Name and Icon to the Application

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](#).



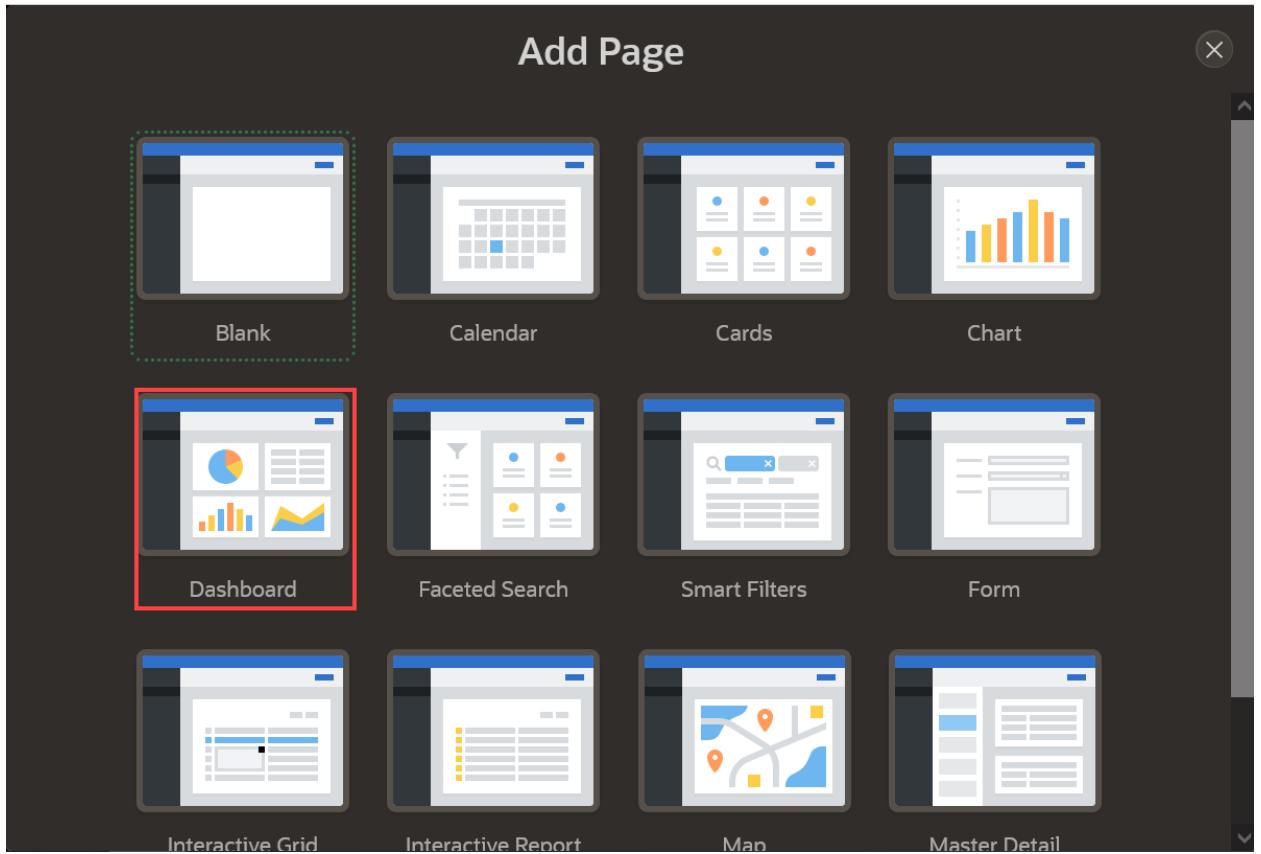
3. Adjust the icon if needed and click **Save Icon**.



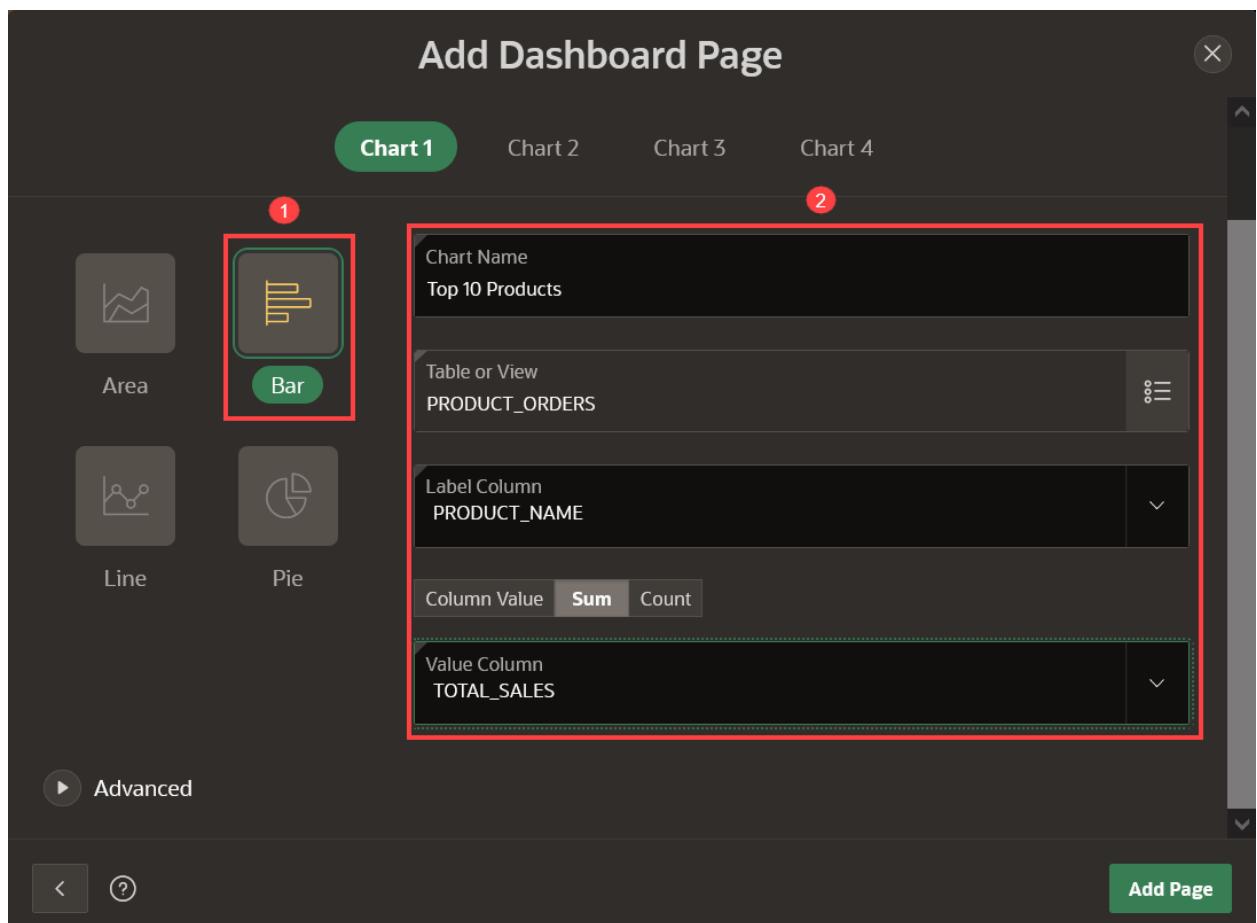
Task 3: Add the Dashboard Page

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**.

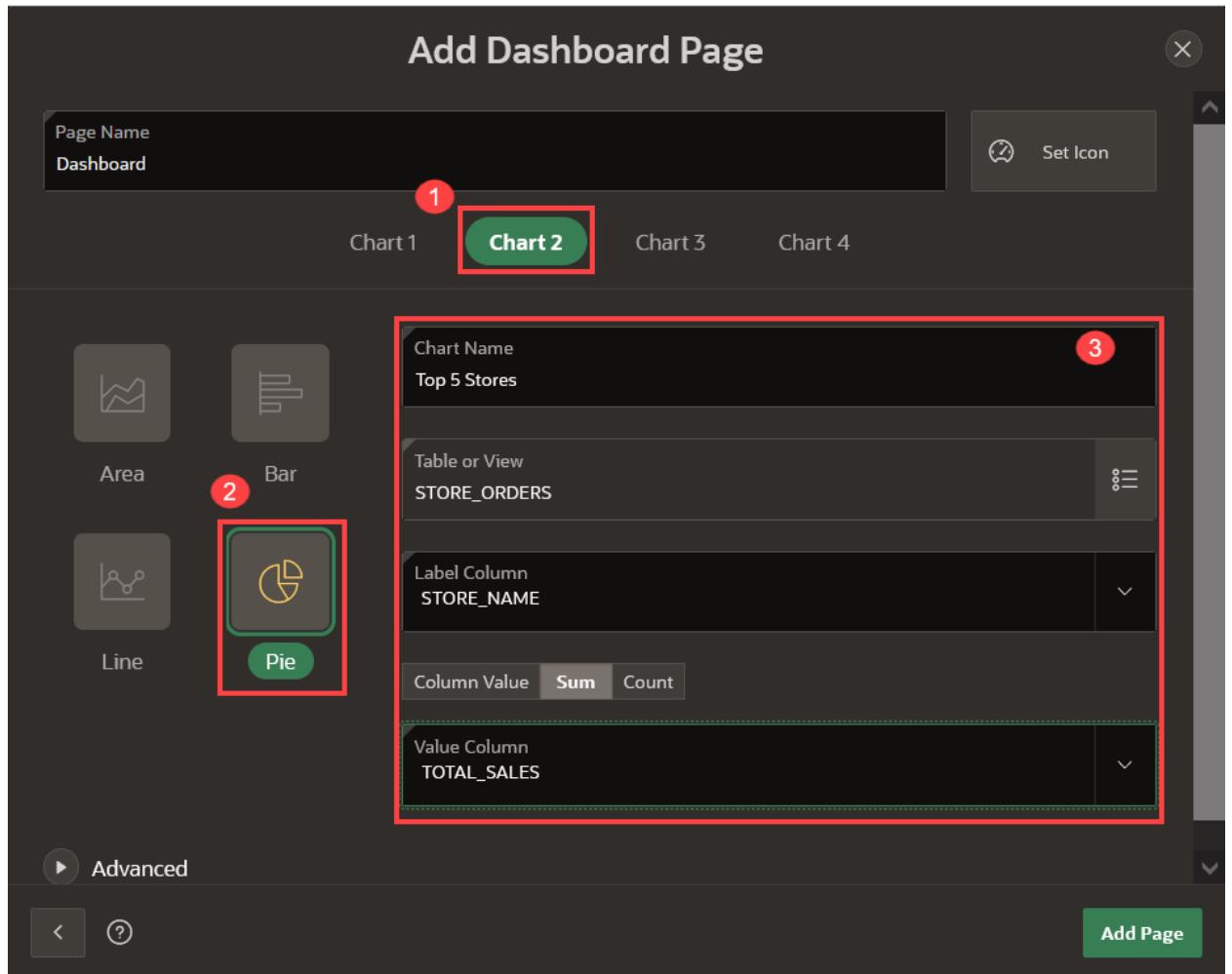


3. For Chart 1, enter the following:
 - o Chart Type – select **Bar**
 - o Chart Name – enter **Top 10 Products**
 - o Table or View – select **PRODUCT_ORDERS**
 - o Label Column – select **PRODUCT_NAME**
 - o Type – select **Sum**
 - o Value Column – select **TOTAL_SALES**.



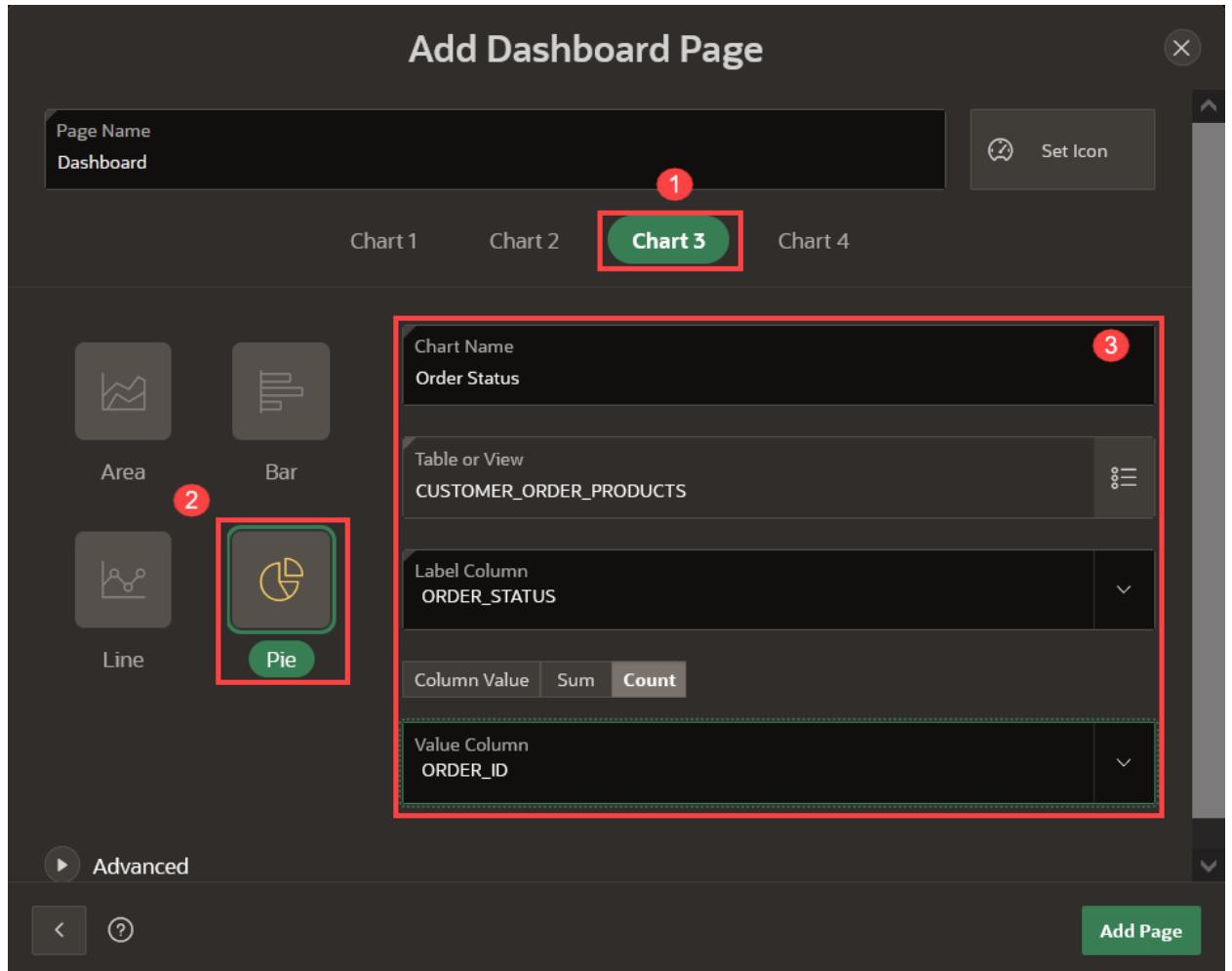
4. Click Chart 2, and enter the following:

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



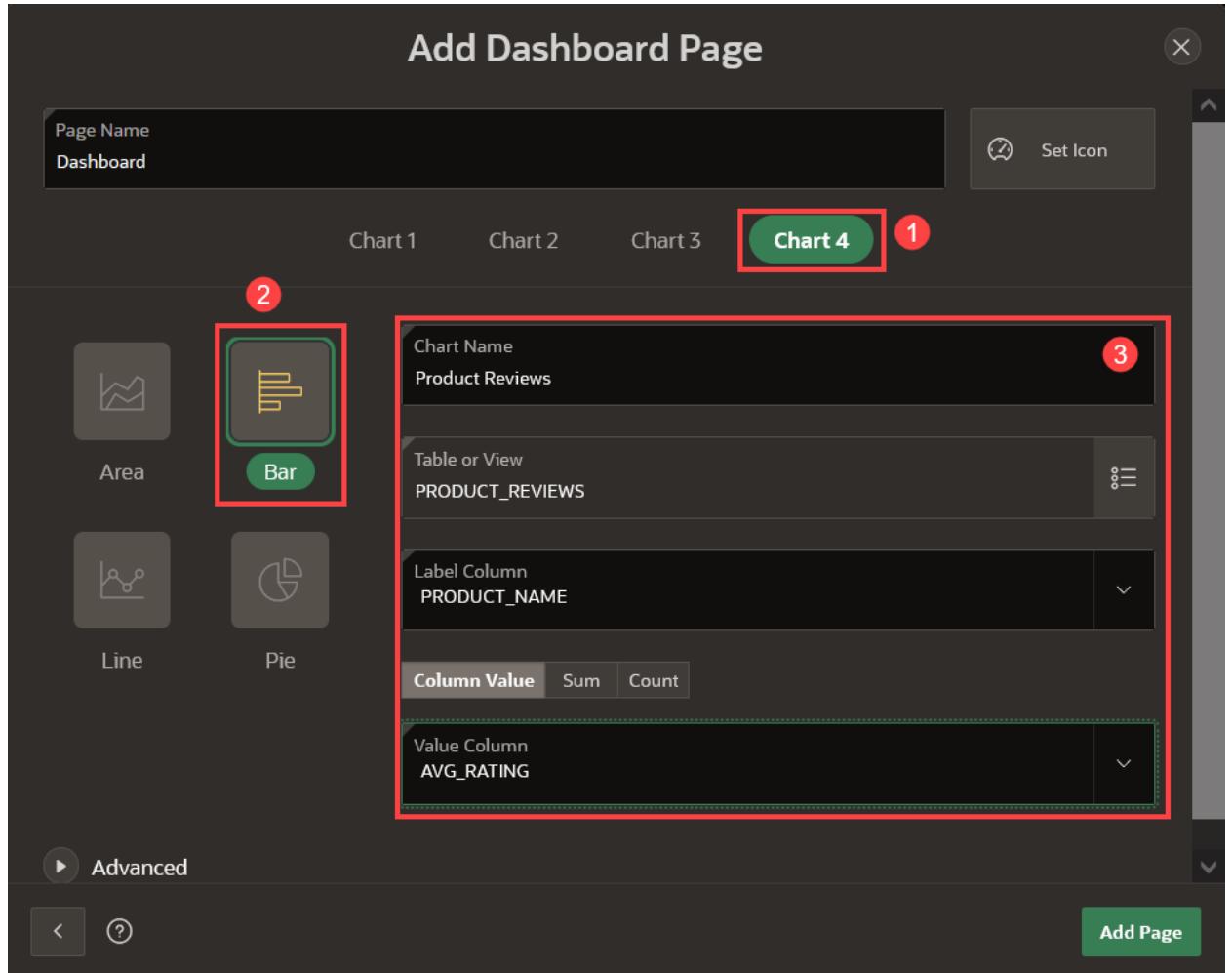
5. Click Chart 3, and enter the following:

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



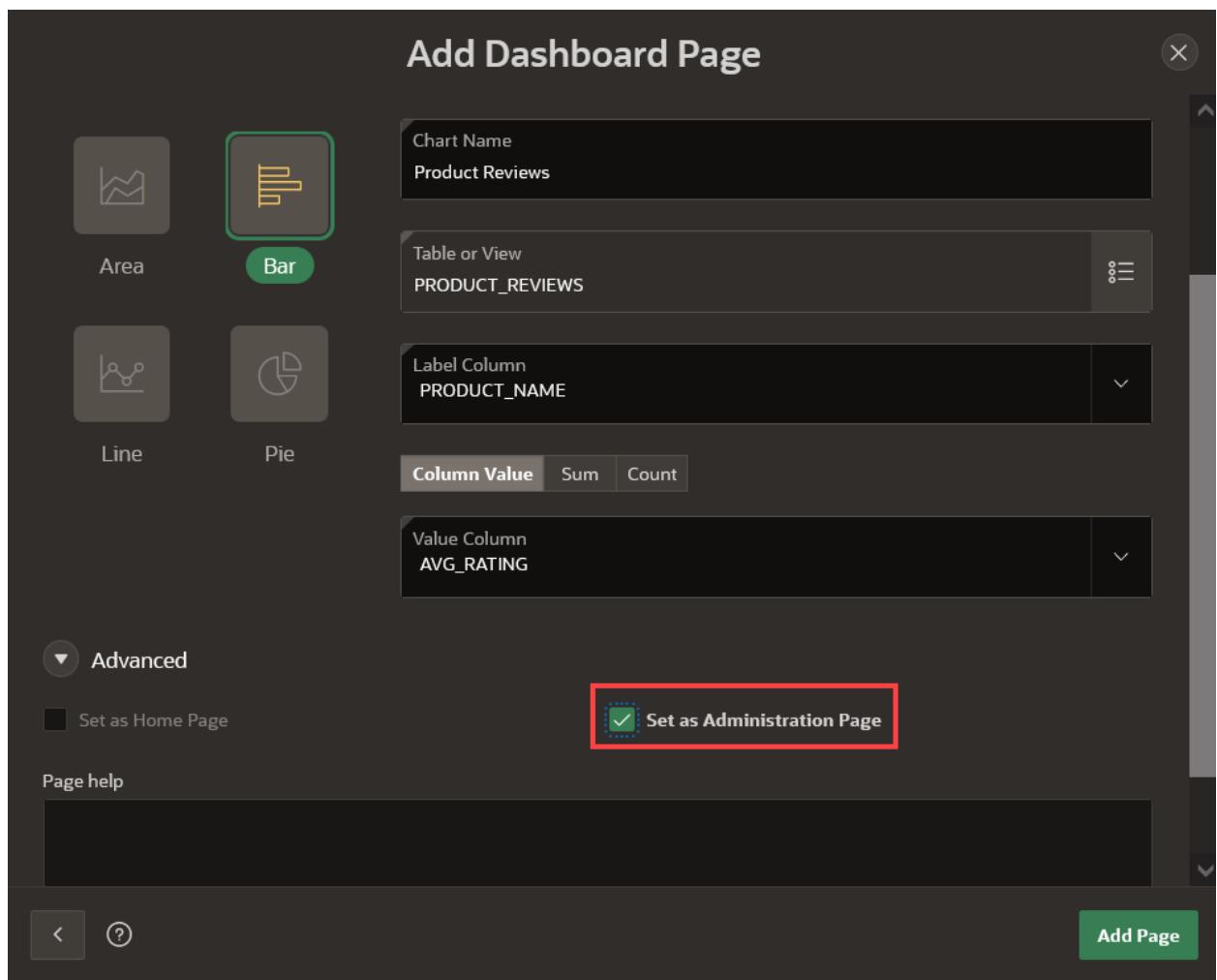
6. Click Chart 4, and enter the following:

- 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.

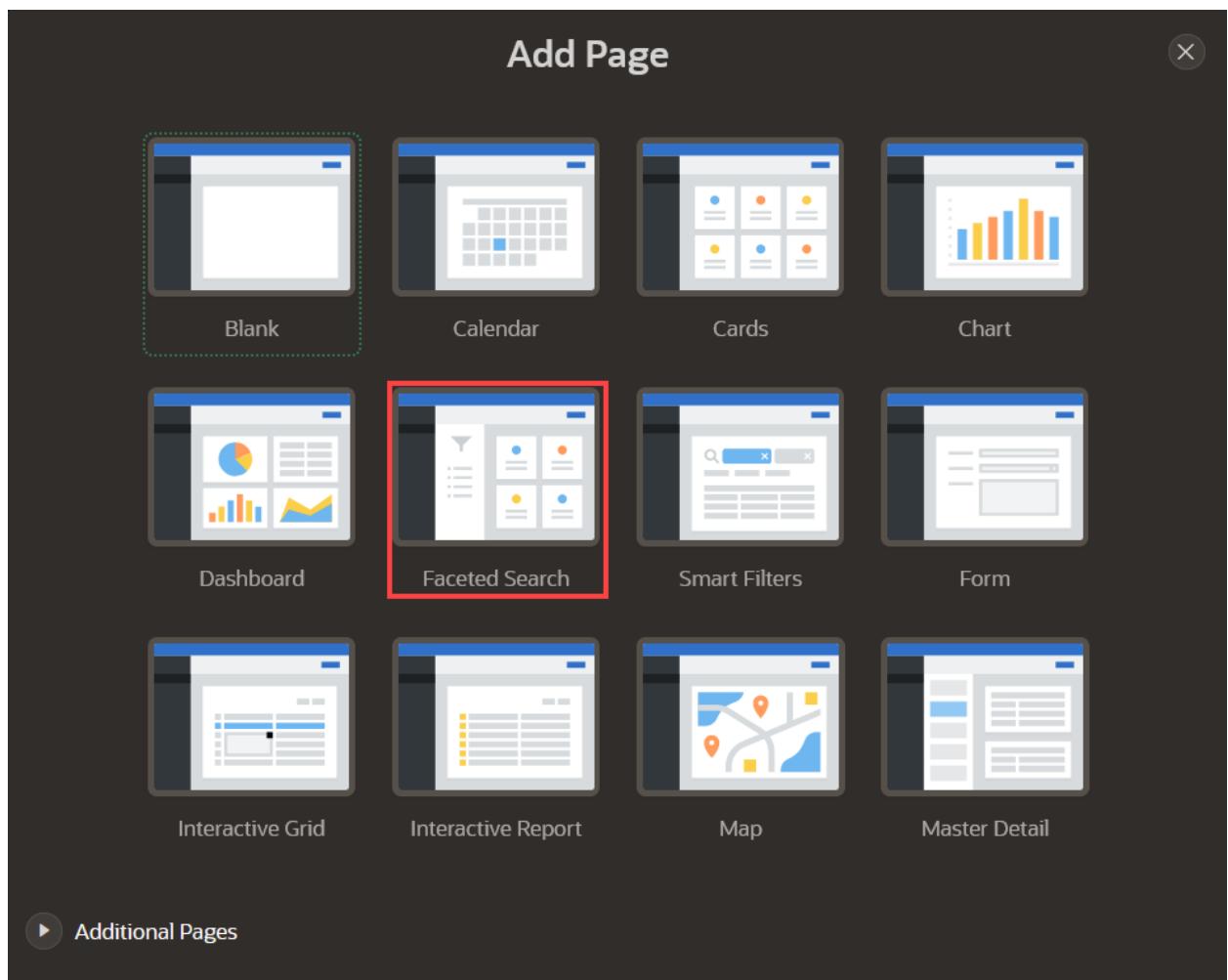
Setting a page as Administration Page will make that only certain users can run the page. In this case, only users with Administration Rights will be able to run the Dashboard page, which means that this page will require authentication.



8. Click **Add Page**.

Task 4: Add the Products Page

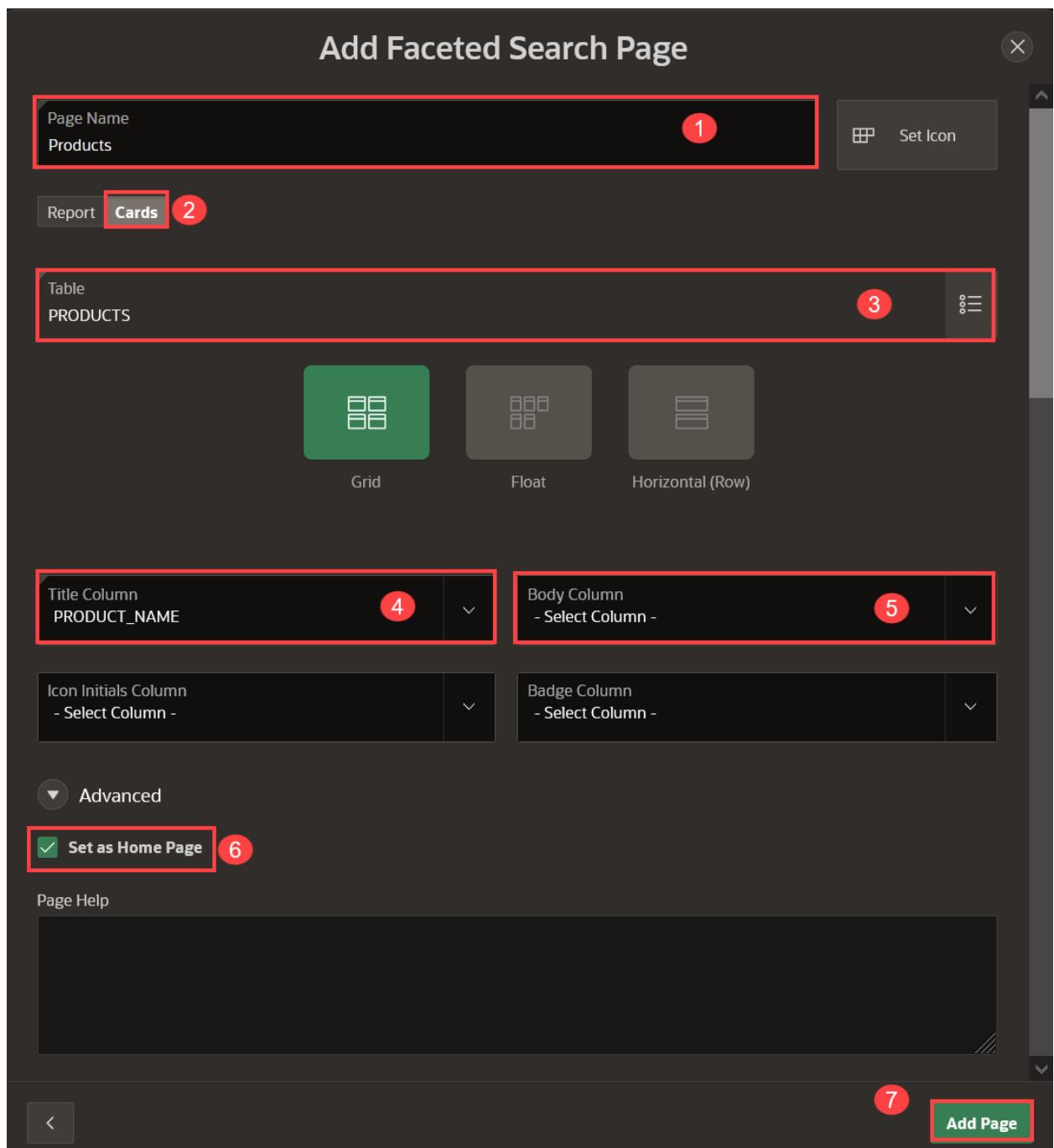
1. In the Create Application wizard, click **Add Page**.
2. Click **Faceted Search**.



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**

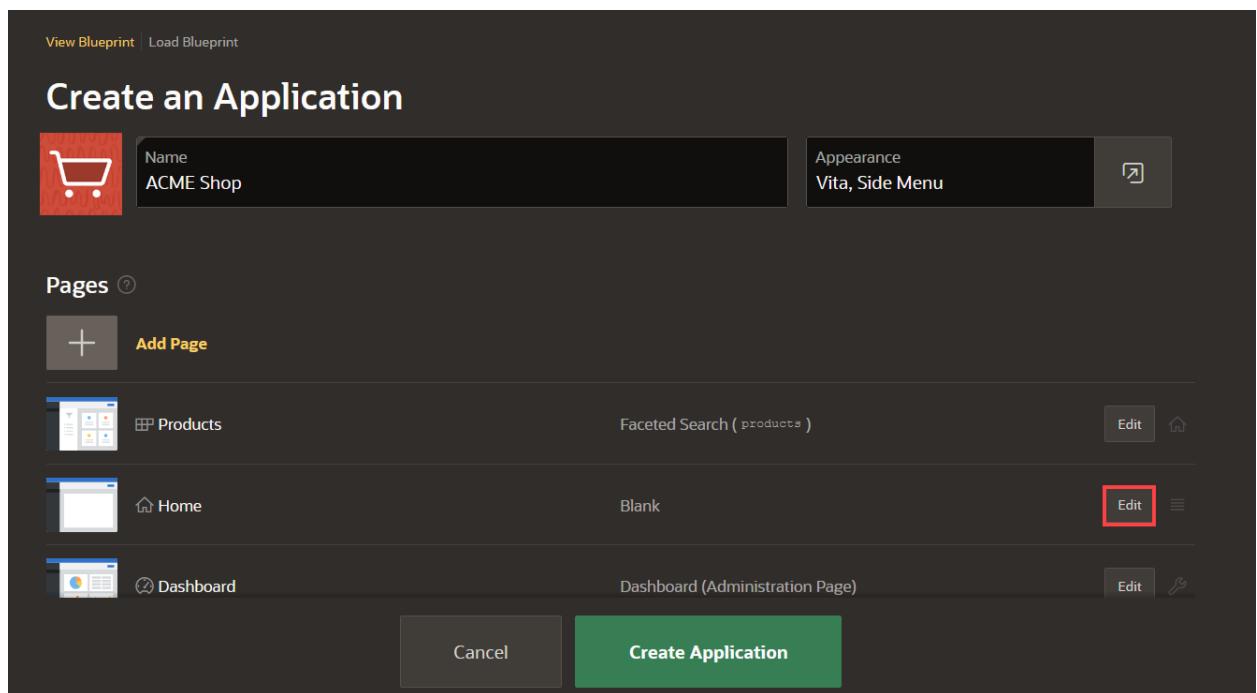
Click **Add Page**.



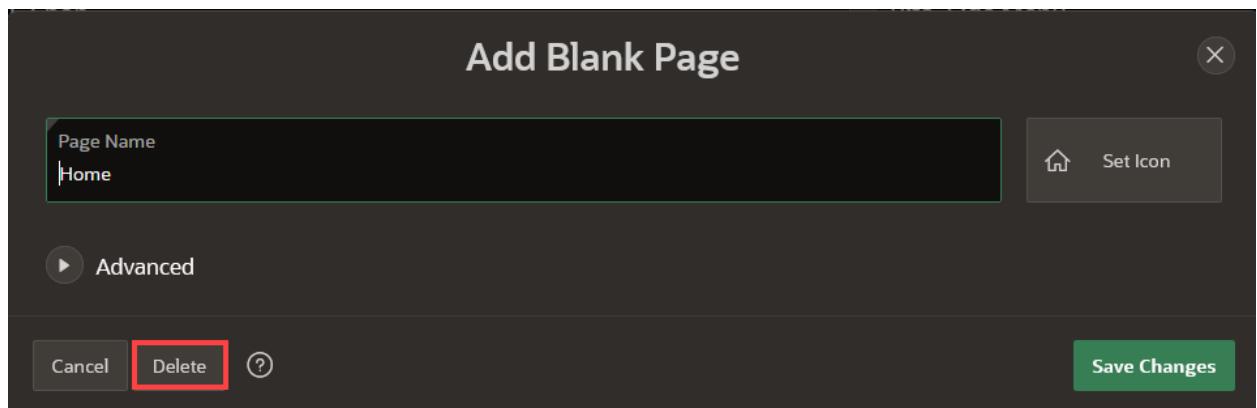
Task 5: Delete Original Home Page

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**.



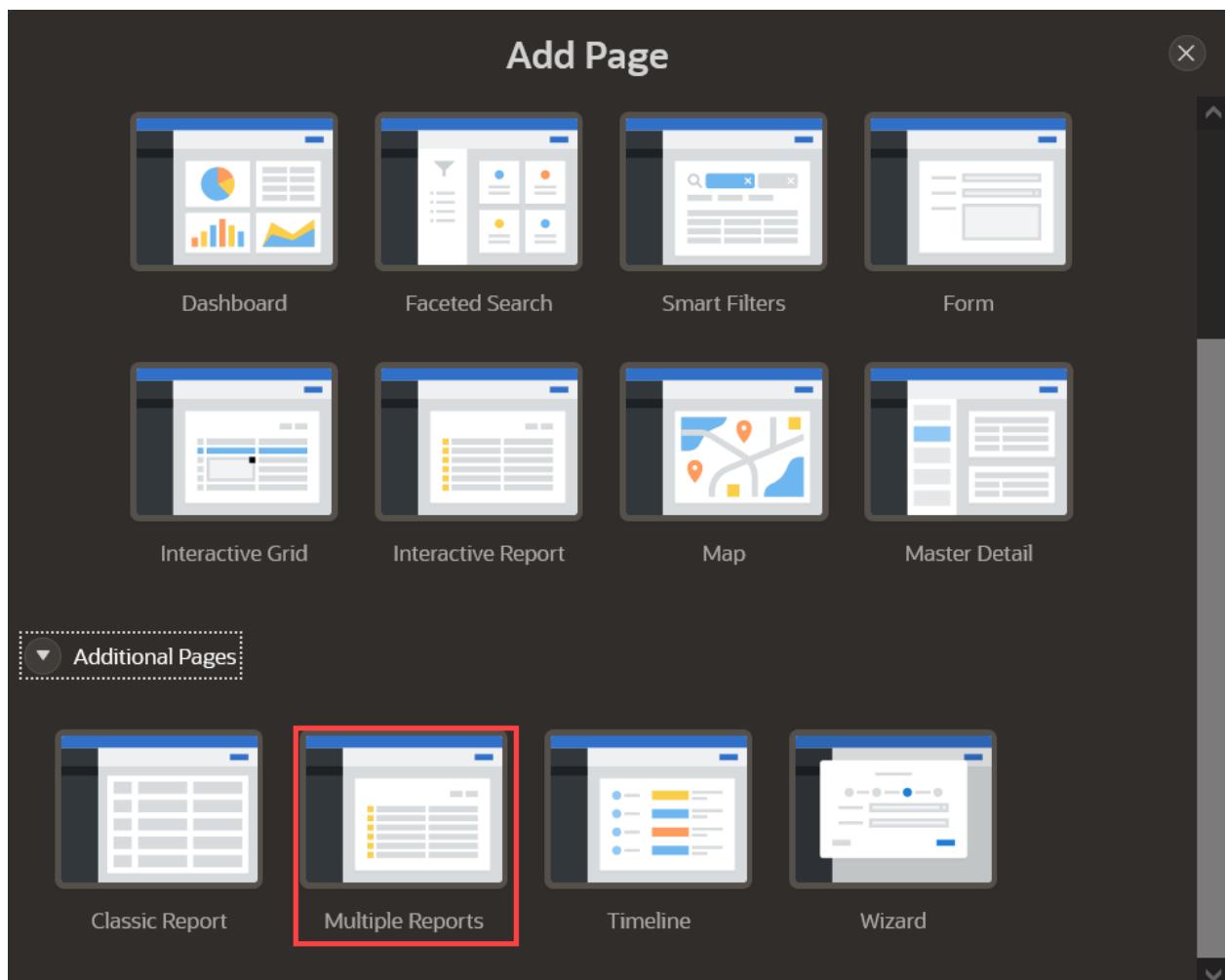
2. Click **Delete**.



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**.

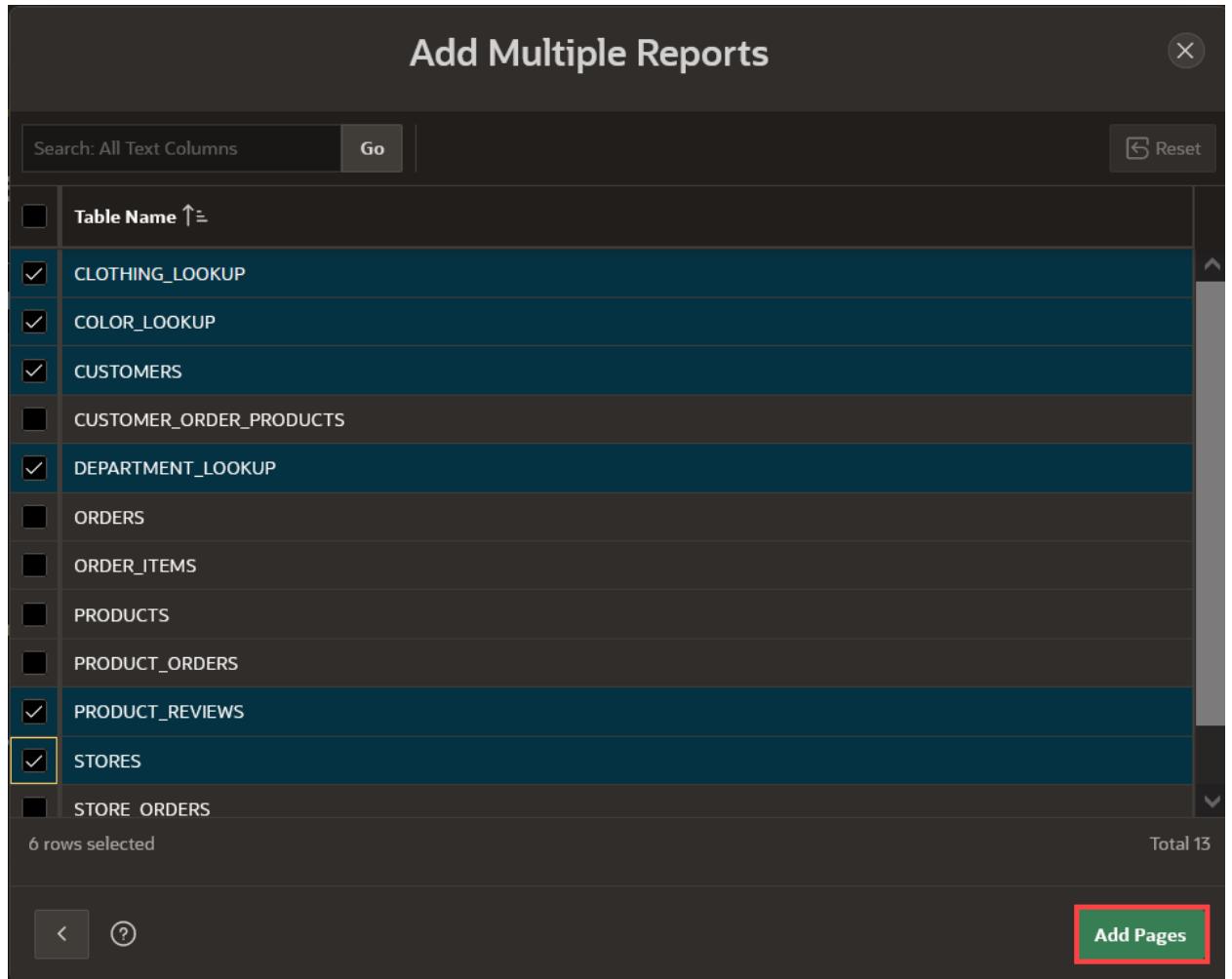
Add Multiple Reports

Search: All Text Columns **Go** [Reset](#)

| <input type="checkbox"/> | Table Name ↑↓ |
|-------------------------------------|-------------------------|
| <input checked="" type="checkbox"/> | CLOTHING_LOOKUP |
| <input checked="" type="checkbox"/> | COLOR_LOOKUP |
| <input checked="" type="checkbox"/> | CUSTOMERS |
| <input type="checkbox"/> | CUSTOMER_ORDER_PRODUCTS |
| <input checked="" type="checkbox"/> | DEPARTMENT_LOOKUP |
| <input type="checkbox"/> | ORDERS |
| <input type="checkbox"/> | ORDER_ITEMS |
| <input type="checkbox"/> | PRODUCTS |
| <input type="checkbox"/> | PRODUCT_ORDERS |
| <input checked="" type="checkbox"/> | PRODUCT_REVIEWS |
| <input checked="" type="checkbox"/> | STORES |
| <input type="checkbox"/> | STORE ORDERS |

6 rows selected Total 13

[**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

[View Blueprint](#) [Load Blueprint](#)

Create an Application

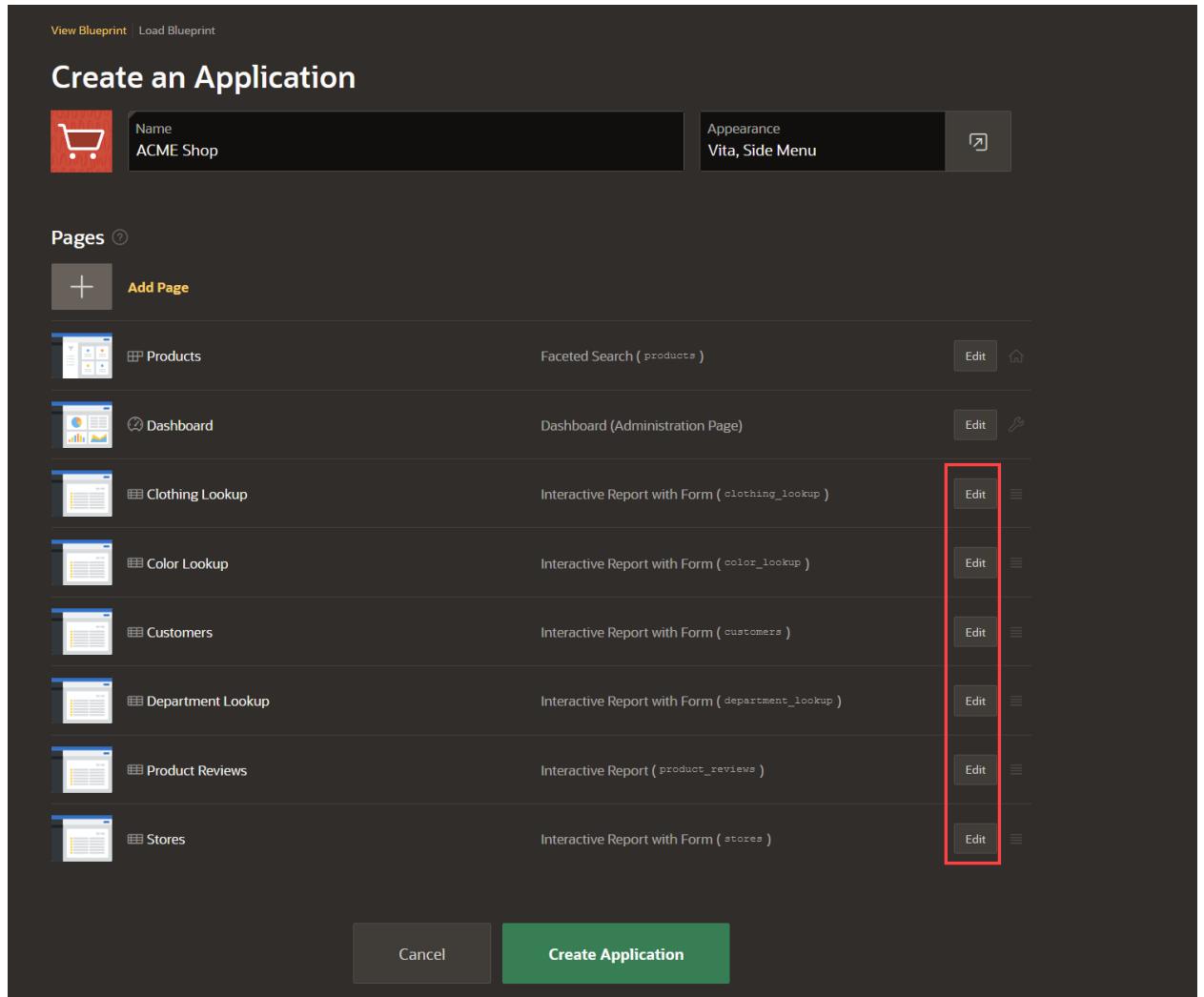
Name: ACME Shop Appearance: Vita, Side Menu

Pages (7)

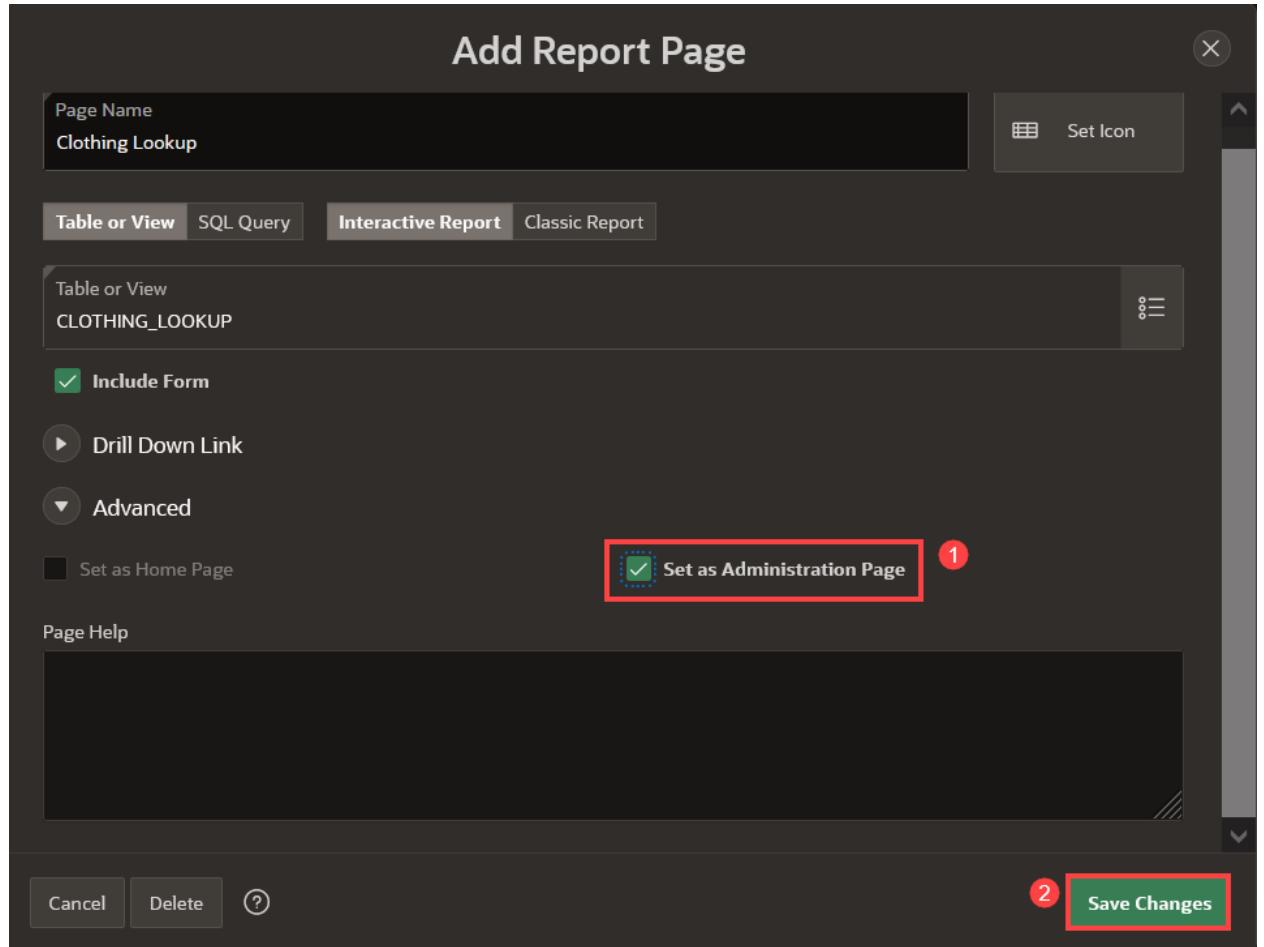
[Add Page](#)

| Page | Description | Actions |
|-------------------|--|---|
| Products | Faceted Search (products) | Edit Home |
| Dashboard | Dashboard (Administration Page) | Edit Copy |
| Clothing Lookup | Interactive Report with Form (clothing_lookup) | Edit More |
| Color Lookup | Interactive Report with Form (color_lookup) | Edit More |
| Customers | Interactive Report with Form (customers) | Edit More |
| Department Lookup | Interactive Report with Form (department_lookup) | Edit More |
| Product Reviews | Interactive Report (product_reviews) | Edit More |
| Stores | Interactive Report with Form (stores) | Edit More |

[Cancel](#) [Create Application](#)

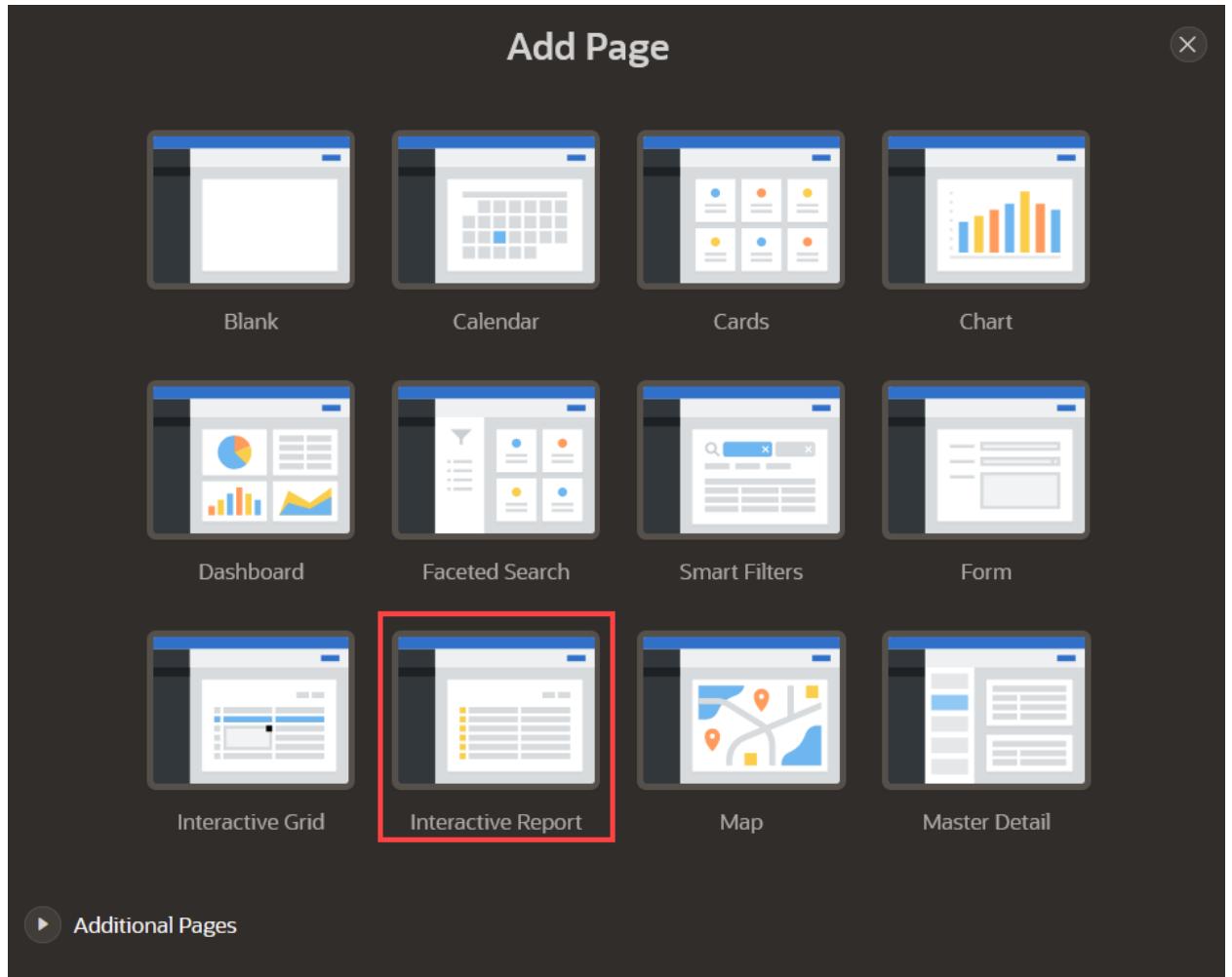


2. For each page you edit, click **Advanced** and check **Set as Administration Page**.
3. Click **Save Changes**.



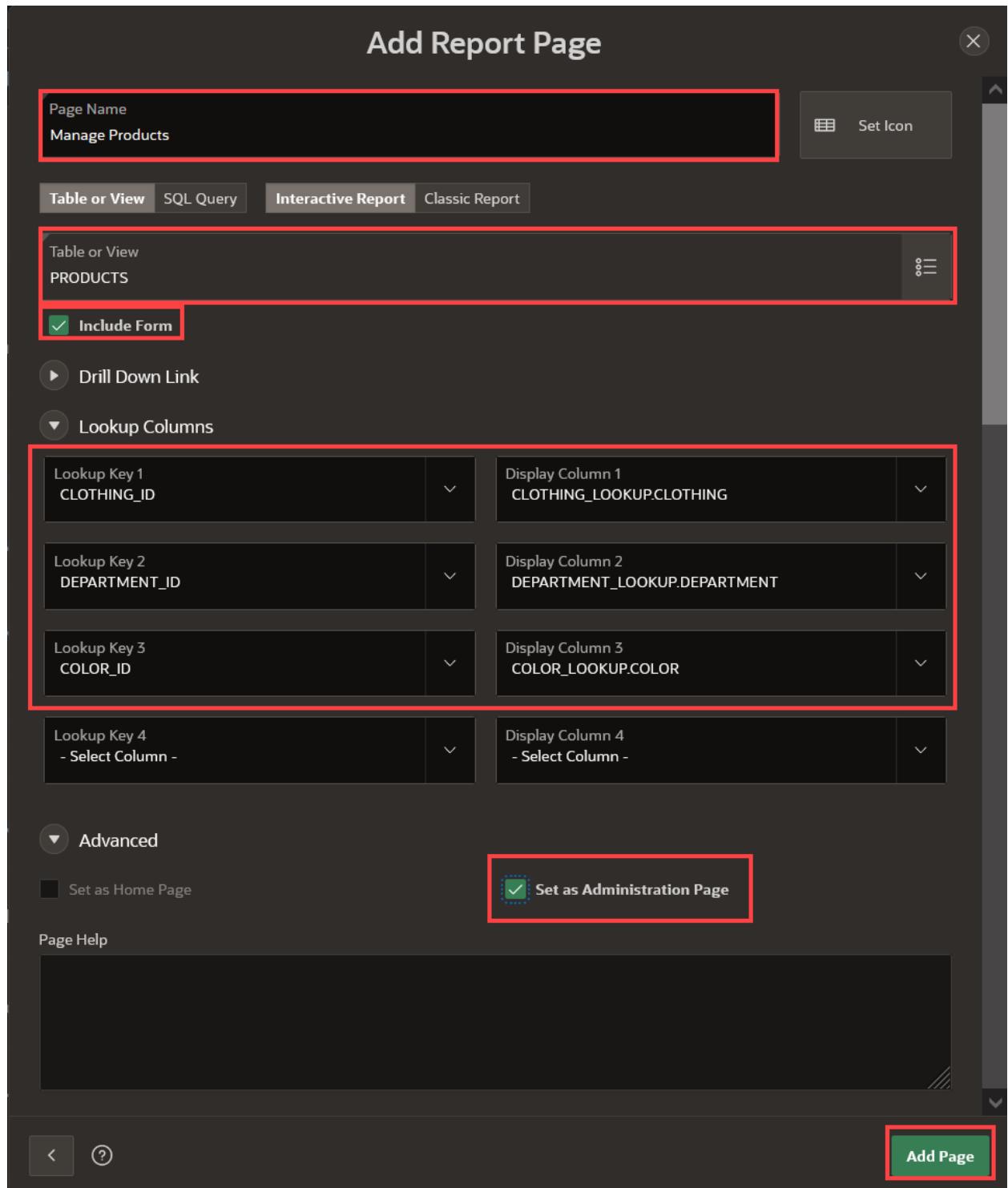
Task 8: Add Manage Products Page

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**.

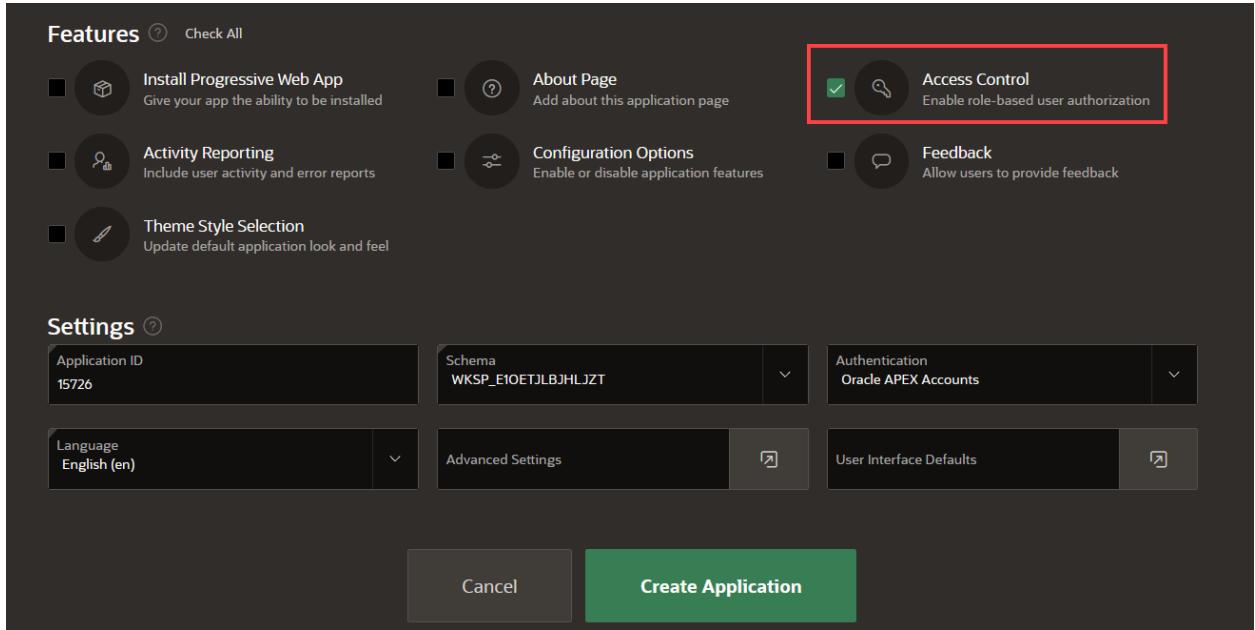
6. Click Add Page.



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

| Icon | Page Name | Description | Edit | Home |
|------|-------------------|--|----------------------|----------------------|
| | Products | Faceted Search (products) | Edit | Home |
| | Dashboard | Dashboard (Administration Page) | Edit | Home |
| | Clothing Lookup | Interactive Report with Form (clothing_lookup) | Edit | Home |
| | Color Lookup | Interactive Report with Form (color_lookup) | Edit | Home |
| | Customers | Interactive Report with Form (customers) | Edit | Home |
| | Department Lookup | Interactive Report with Form (department_lookup) | Edit | Home |
| | Product Reviews | Interactive Report (product_reviews) | Edit | Home |
| | Stores | Interactive Report with Form (stores) | Edit | Home |
| | Manage Products | Interactive Report with Form (products) | Edit | Home |

Features

[Check All](#)

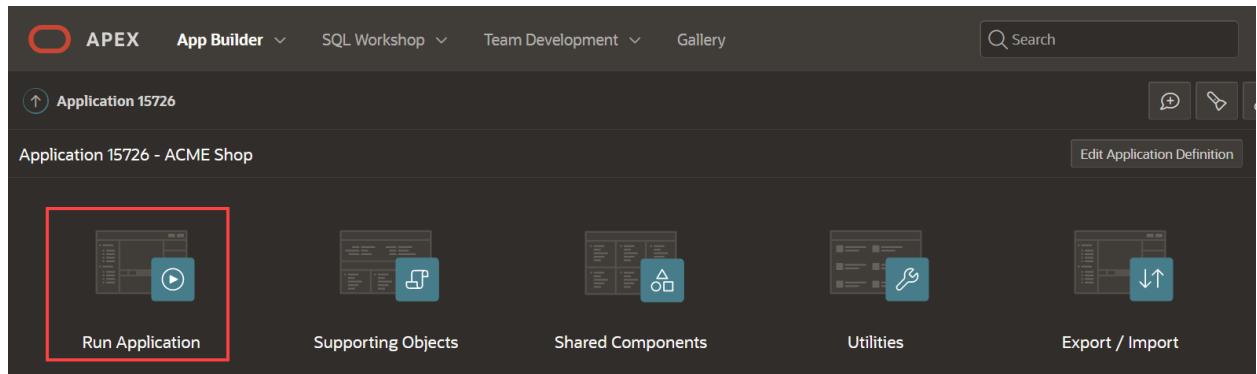
| | | | | | |
|--------------------------|--|--------------------------|---|-------------------------------------|--|
| <input type="checkbox"/> | Install Progressive Web App Give your app the ability to be installed | <input type="checkbox"/> | About Page Add about this application page | <input checked="" type="checkbox"/> | Access Control Enable role-based user authorization |
| <input type="checkbox"/> | Activity Reporting Include user activity and error reports | <input type="checkbox"/> | Configuration Options Enable or disable application features | <input type="checkbox"/> | Feedback Allow users to provide feedback |
| <input type="checkbox"/> | Theme Style Selection Update default application look and feel | | | | |

Settings

| | | |
|--------------------------|-------------------------------|--|
| Application ID 15726 | Schema WKSP_EIOETJLBJHLJZT | Authentication Oracle APEX Accounts |
| Language English (en) | Advanced Settings | User Interface Defaults |

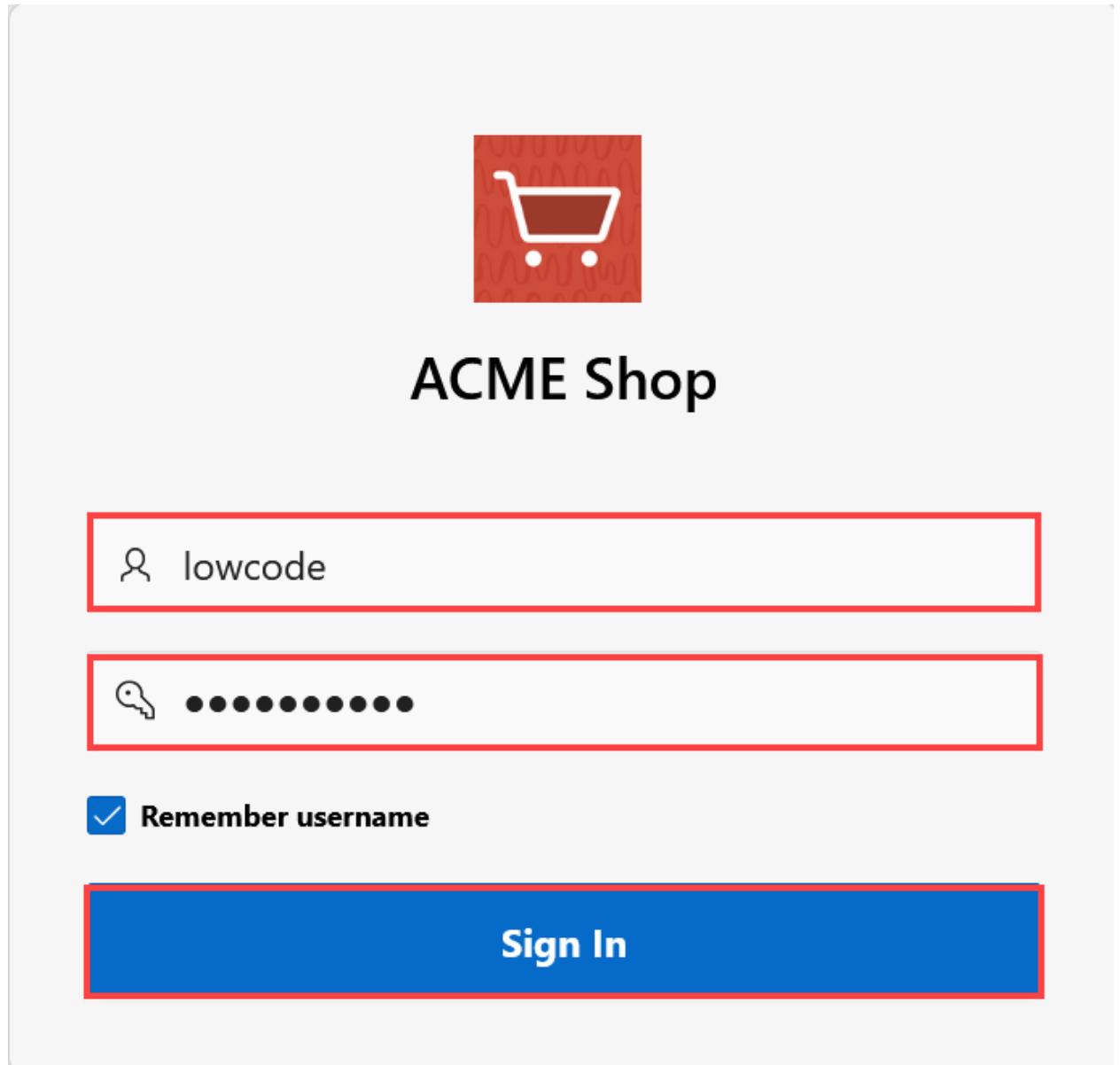
[Cancel](#) Create Application

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



Task 11: Run the Application

- 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:

The screenshot shows a web page from 'ACME Shop' with a green success message at the top right. The message reads: 'Order successfully created: 268825160859771512879186327887983976716'. Below this, a red heart icon and the text 'Thank you for your order!' are displayed. At the top left, there's a blue header bar with the shop name. The main content area shows an order summary and two product items.

Order: 268825160859771512879186327887983976716

Order Placed: 8/23/2021
Status: OPEN
Total: 19.94

Girl's Jeans (Grey)

Quantity: 1
Unit Price: 9.7

Boy's Coat (Blue)

Quantity: 1
Unit Price: 10.24

Estimated Time: 15 minutes

Objectives

In this lab, you will:

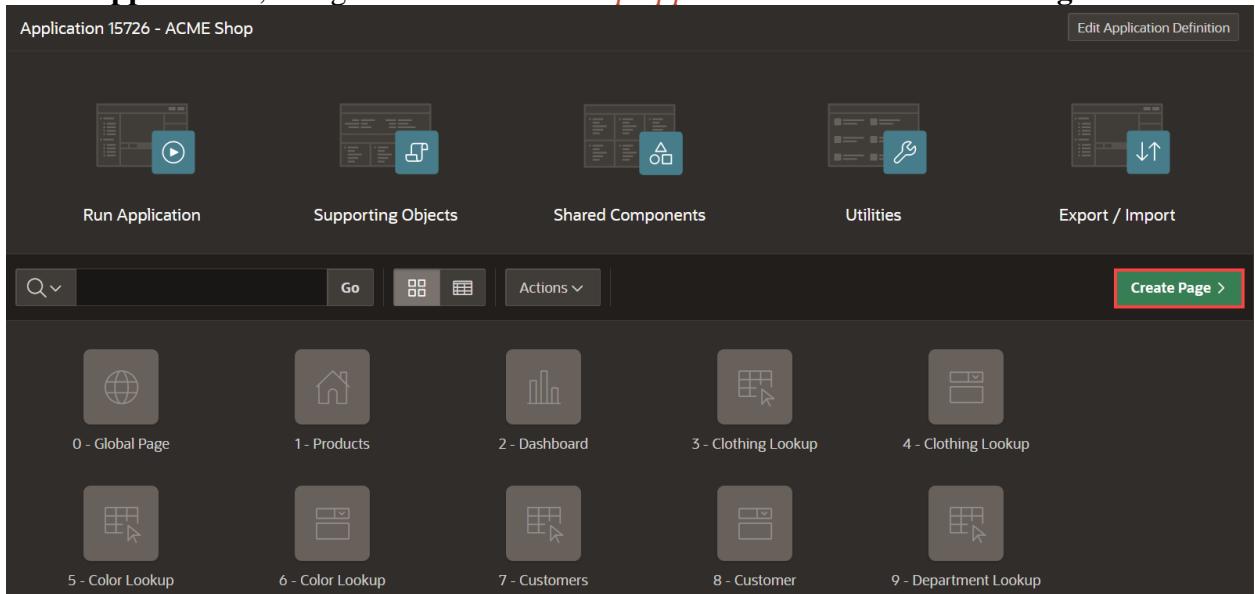
- Create a page to review the items that the customer recently bought.

[Collapse All Tasks](#)

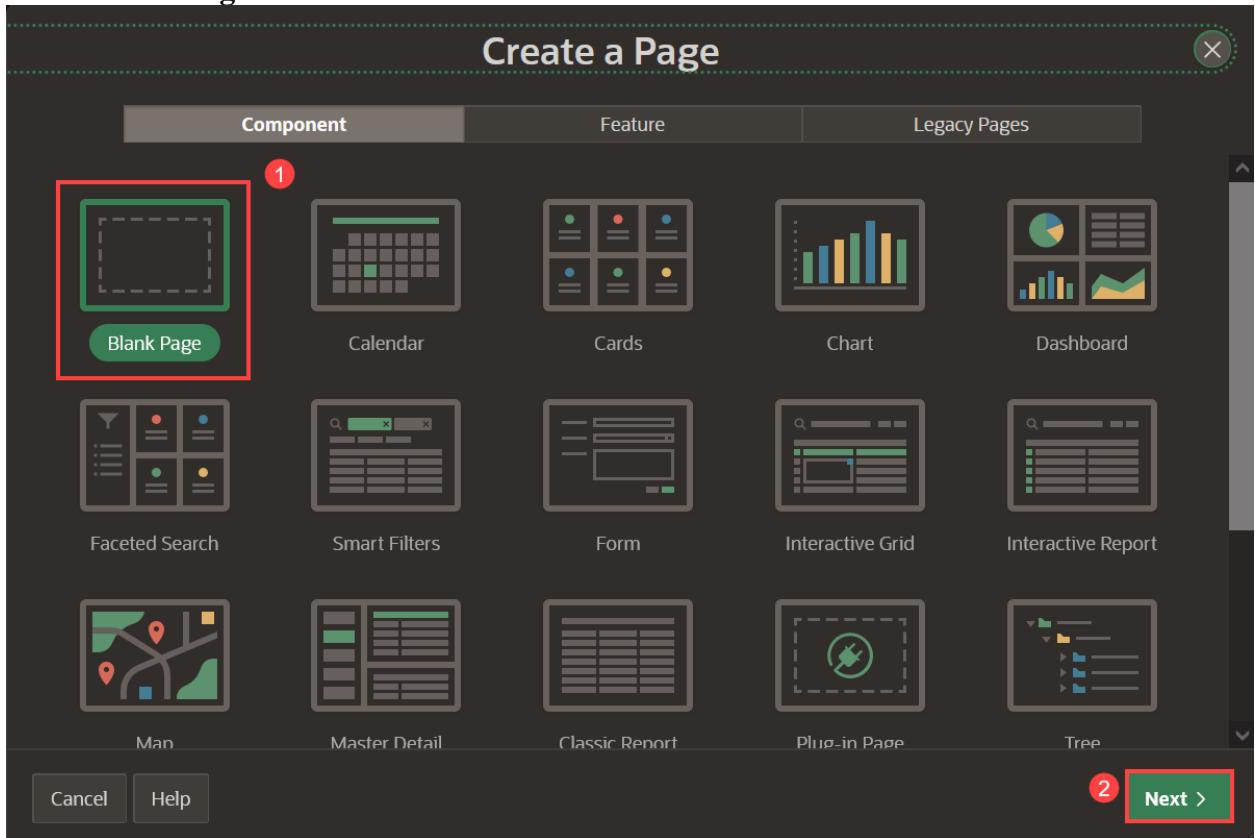
Task 1: Create a Normal Page - Order Information

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**.



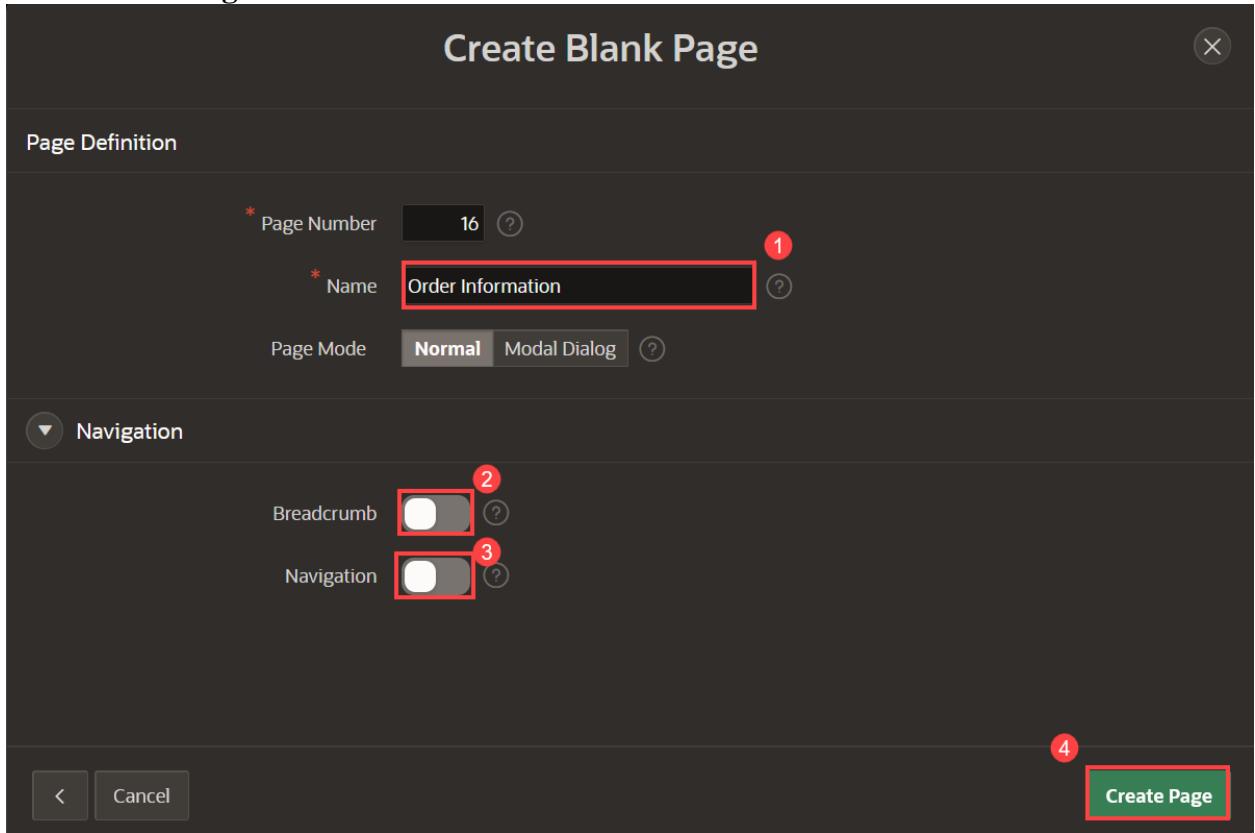
2. Select **Blank Page** and click **Next**.



3. Enter the following and click **Next**.

- o Page Number - enter **16**
- o Name - enter **Order Information**

- Under Navigation, deselect **Breadcrumb** and **Navigation**.
4. Click **Create Page**.

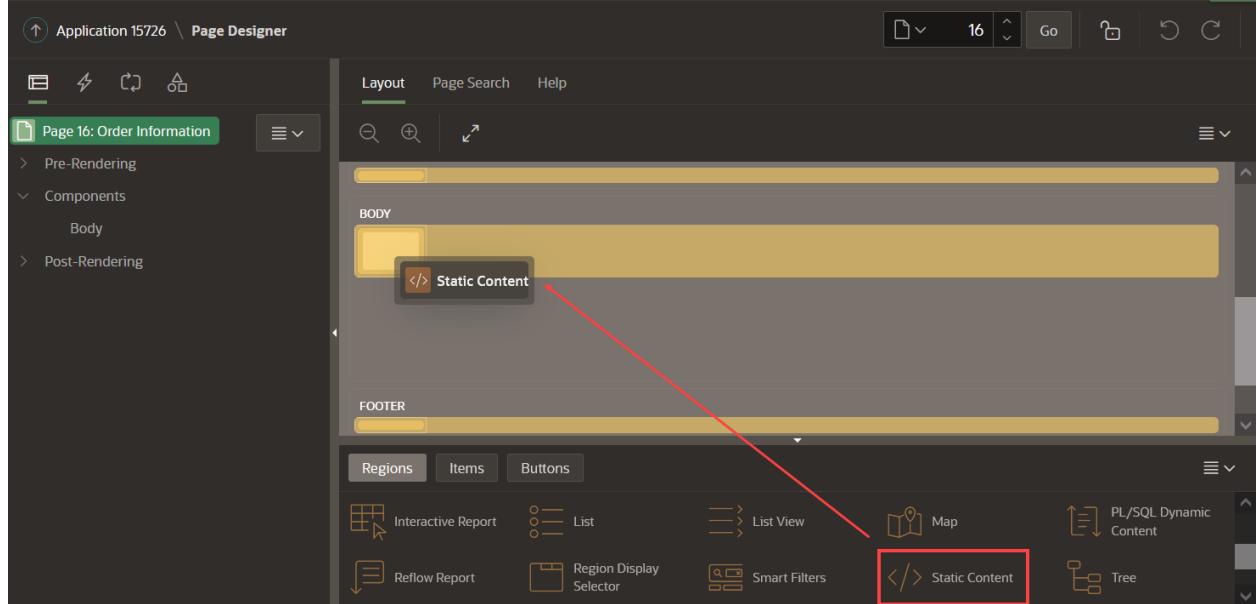


Task 2: Add a Region

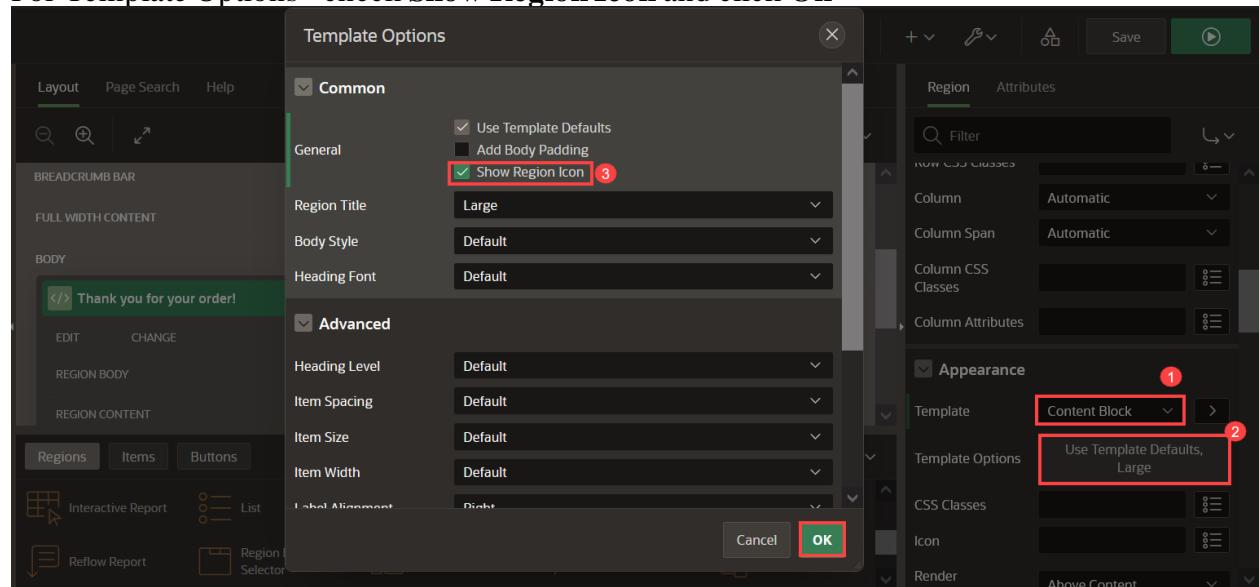
Add a region to the page to display order details.

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.

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



- 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**



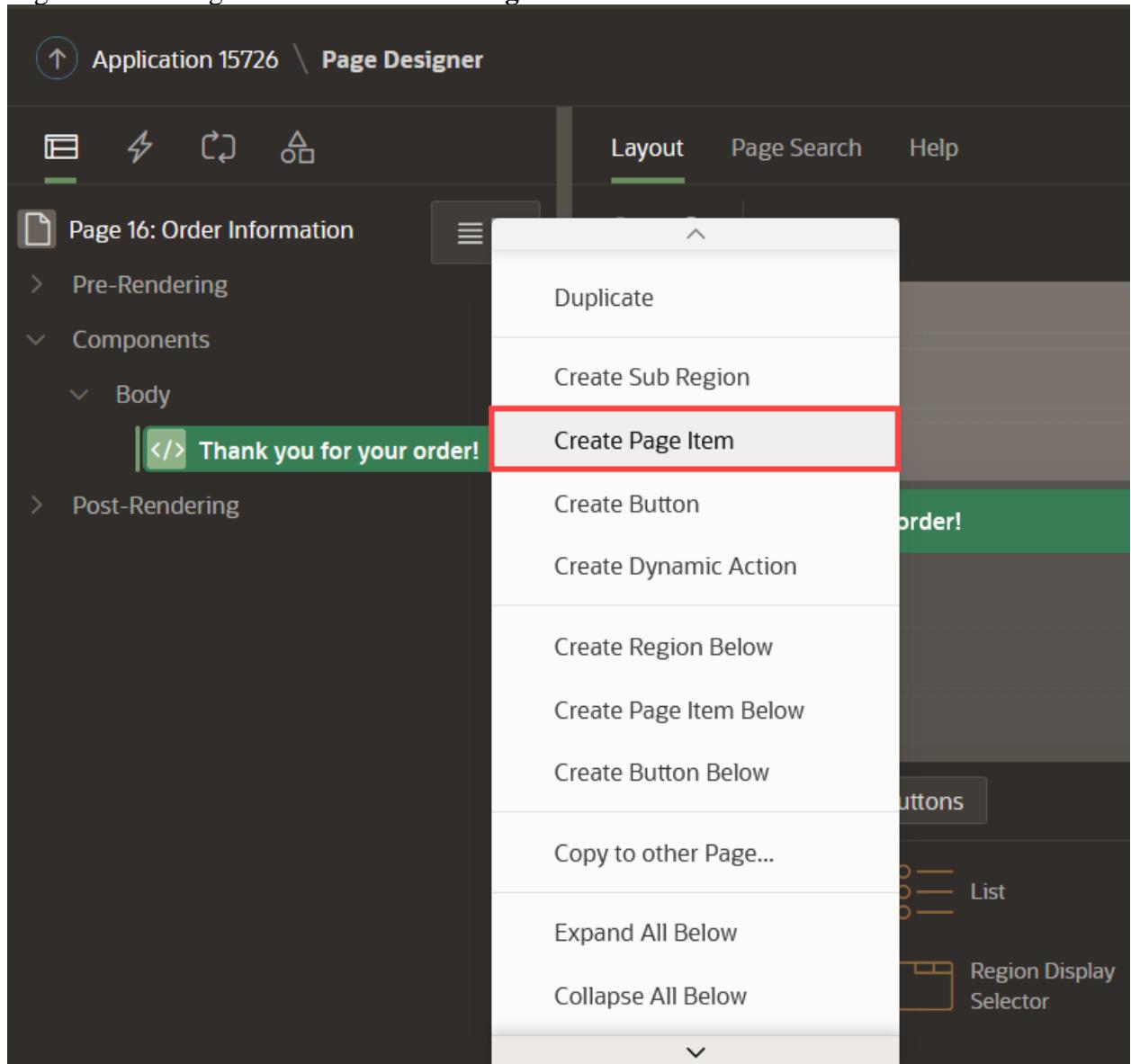
- For Icon, enter **fa-heart**

Task 3: Add Items to the Page

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

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

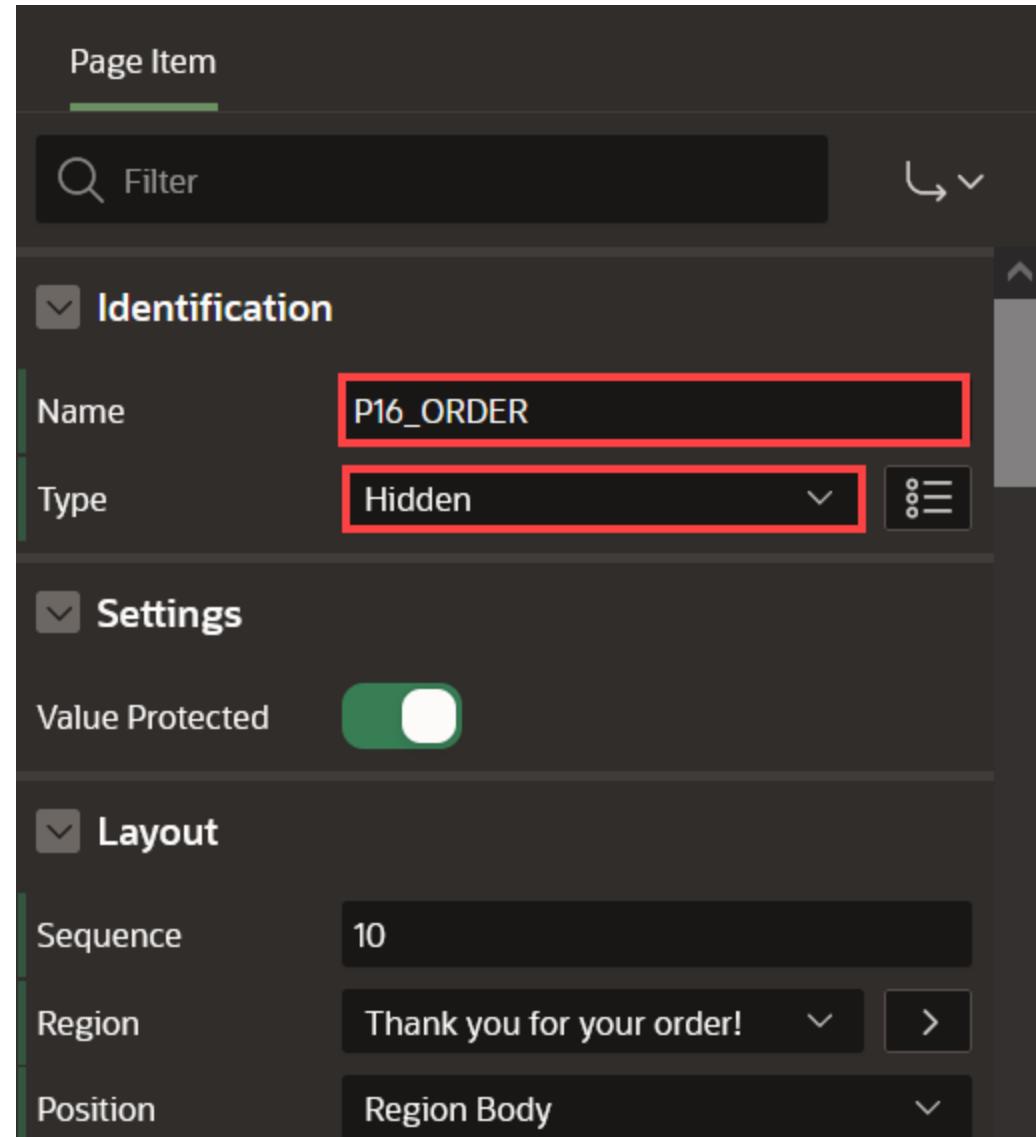
2. Right-click the region and click **Create Page Item**.



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

Table 1: Page Item values

| Name | Type |
|-----------|--------|
| P16_ORDER | Hidden |

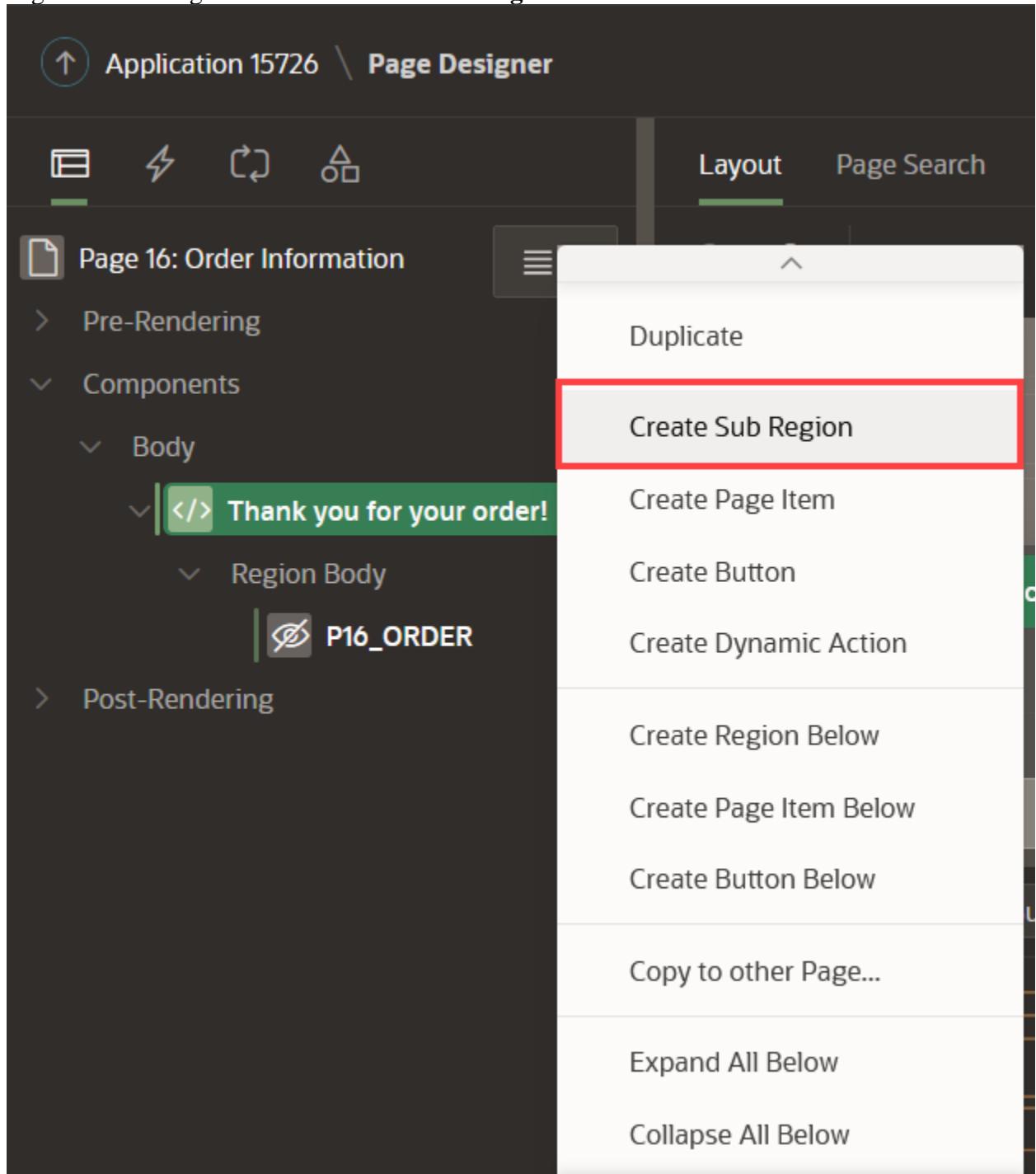


Task 4: Add Static Content Region

Add a region to contain Order details and items.

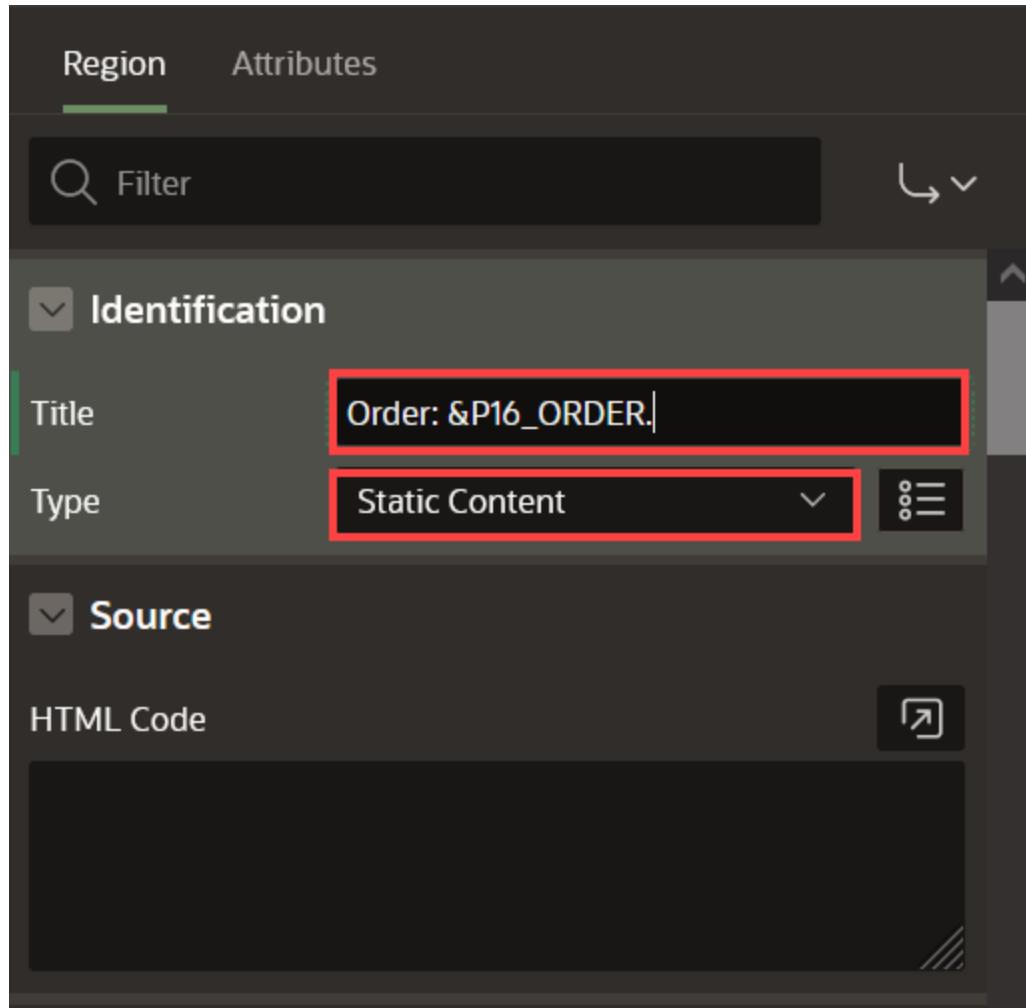
1. In the Rendering tree (left pane), navigate to the **Thank you for your order!** region.

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**

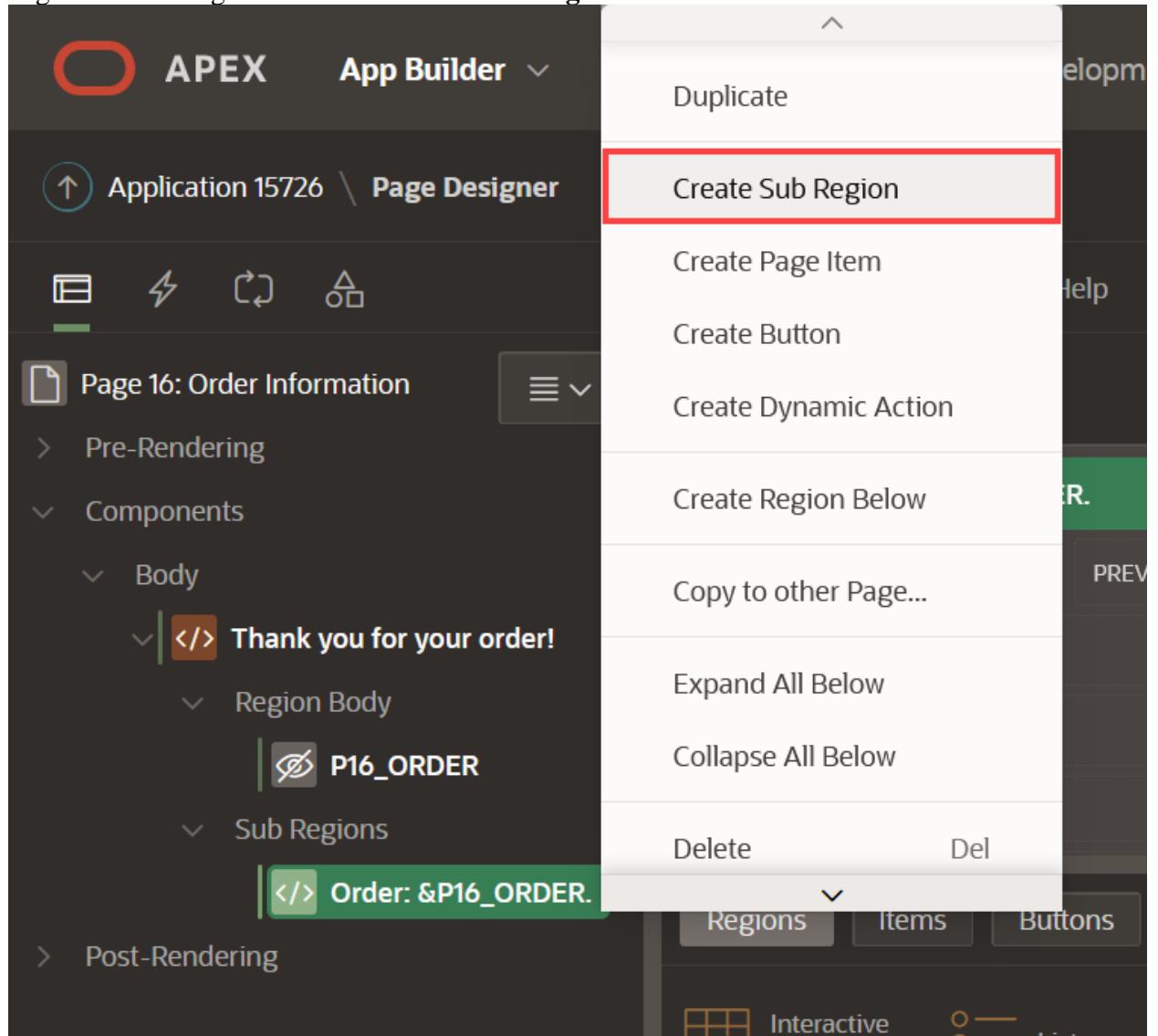


Task 5: Add Order Details Region

Add a region to display Order details.

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

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



- 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
```

Region Attributes

Filter

Identification

Title ① Order Details

Type ② Cards

Source

Location Local Database

Type ③ SQL Query

SQL Query

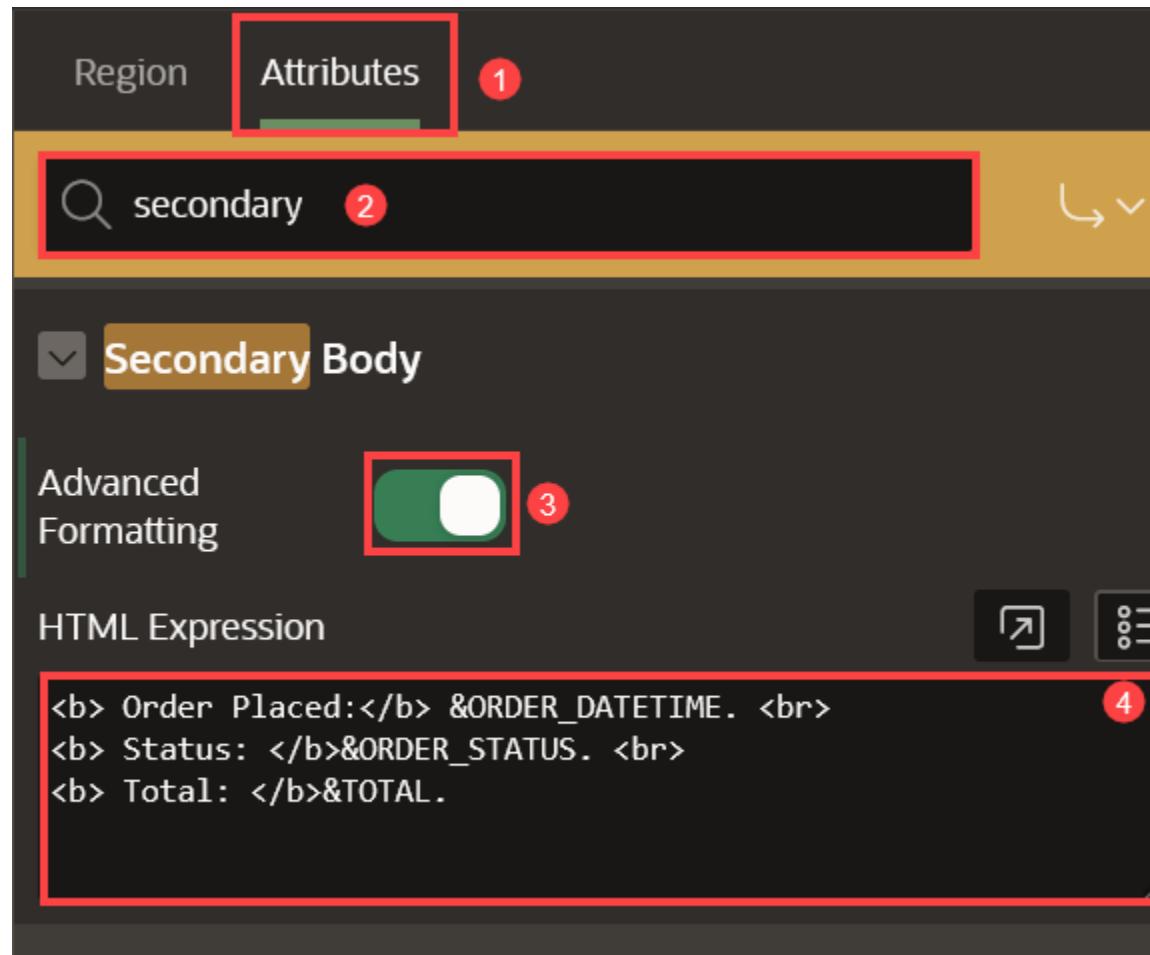
```
④
SELECT o.order_id,
       o.order_datetime,
       o.customer_id,
       o.order_status,
       o.store_id,
       (SELECT Sum(unit_price * quantity)
            FROM   order_items i
            WHERE  i.order_id = o.order_id) total
  FROM   orders o
 WHERE  order_id = :P16_ORDER
```

4. Click **Attributes**.

- o Search for Secondary Body in the filter and do the following:

- Set Advanced Formatting to **On**
- For HTML Expression - enter:
 - Copy ** Order Placed: &ORDER_DATETIME.
**
 - ** Status: &ORDER_STATUS.
**

** Total: &TOTAL.**

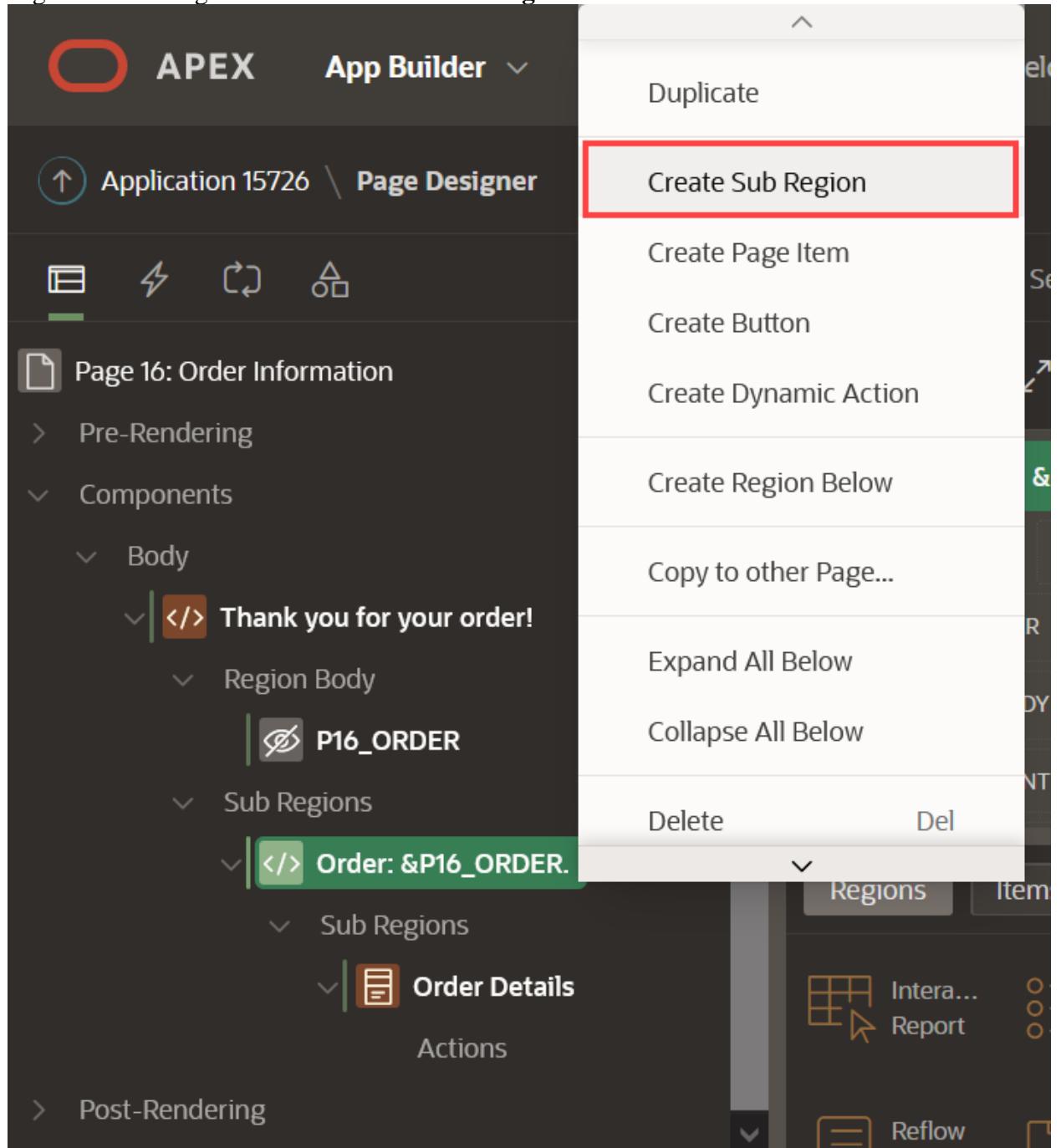


Task 6: Add Items Region

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:

- Copy

```
SELECT      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

Region Attributes

Filter ↻

Identification

Title 1 Items

Type 2 Cards

Source

Location Local Database ↻

Type 3 SQL Query ↻

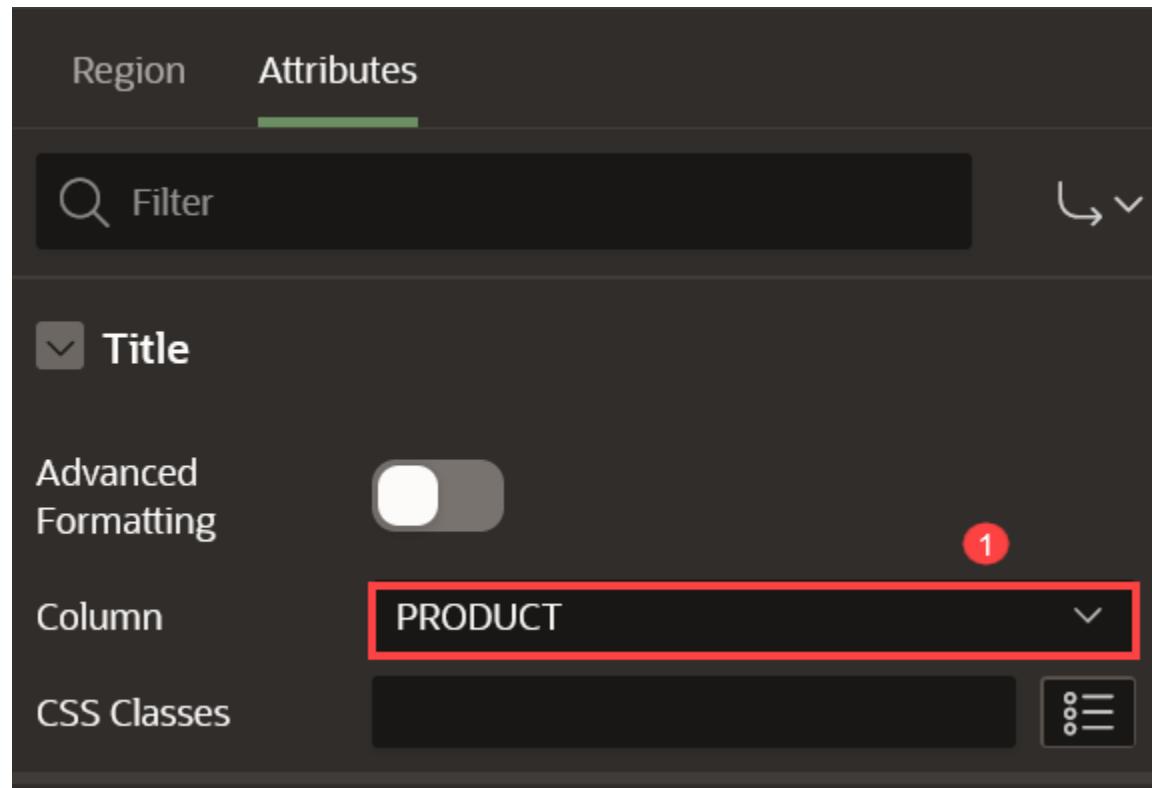
SQL Query ↗

```
4
SELECT 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:

- o 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  

```
<b>Quantity: </b> &QUANTITY. <br>
<b>Unit Price: </b>&UNIT_PRICE.
```

Icon and Badge

Icon Source No Icon

Badge Column - Select -

Media

Advanced Formatting 

Source BLOB Column 4

BLOB Column PRODUCT_IMAGE 5

Position Body

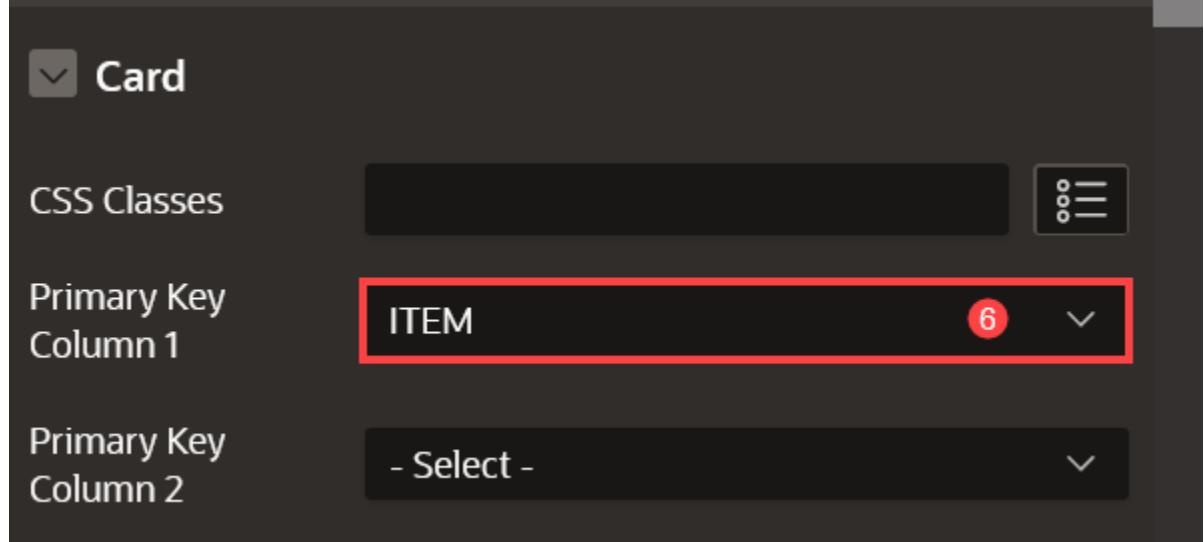
Appearance Auto

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:

The screenshot shows a shopping cart interface for 'ACME Shop'. At the top, there are links for 'Shopping Cart' (with 2 items) and 'Administration'. The main content area displays two items in a grid:

- Girl's Jeans (Grey)** by KINETICUT. Quantity: 1. Unit Price: 9.7. Subtotal: 9.7. Includes an 'Edit' button.
- Boy's Coat (Blue)** by GRONK. Quantity: 1. Unit Price: 10.24. Subtotal: 10.24. Includes an 'Edit' button.

To the right of the items is a sidebar titled 'Order Information' containing fields for 'Email Address', 'Full Name', and 'Store' (with a dropdown menu showing '- Select a Store -'). At the bottom right are buttons for 'Clear Shopping Cart' and 'Proceed to Checkout'.

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

The screenshot shows the Oracle APEX Page Designer interface. In the center, there's a 'Regions' tab selected in the ribbon. On the left, the page structure tree shows a 'Body' section with a 'Region Body' containing a placeholder 'Thank you for your order!'. Below it is a 'Sub Regions' section with a 'Order Details' region. Under 'Actions', there is a green-bordered 'Items' button, which is highlighted with a red box. The right side of the screen displays the 'Identification' and 'Source' sections of the item definition. The 'Identification' section has 'Title' set to 'Items' and 'Type' set to 'Cards'. The 'Source' section shows 'Location' as 'Local Database' and 'Type' as 'SQL Query'. The SQL query is defined as:

```
SELECT o.line_item_id, p.product_name, o.unit_price, o.quantity, (o.unit_price * o.quantity) Subtotal,
```

2. Under Application Logic, click Application Items.

The screenshot shows the 'Application Logic' section of the Oracle APEX application configuration. The 'Application Items' button is highlighted with a red box. Other options in this section include 'Application Definition', 'Application Processes', 'Application Computations', 'Application Settings' (with a value of 1), and 'Build Options' (with a value of 2). To the right, there are sections for 'Security' (with 'Security Attributes', 'Authentication Schemes' (1), 'Authorization Schemes' (3), 'Application Access Control' (3), and 'Session State Protection'), 'User Interface' (with 'User Interface Attributes', 'Progressive Web App', and 'Themes' (1)), and 'Other Components' (with 'Lists of Values' (5), 'Plug-ins', 'Component Settings', and 'Shortcuts'). At the bottom, there are sections for 'Navigation' (with 'Lists' (4), 'Navigation Menu', and 'Breadcrumbs' (1)), 'Files and Reports' (with 'Static Application Files' (3), 'Static Workspace Files', and 'Report Queries'), and 'General' (with 'Page Properties' (1)).

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.

The screenshot shows the 'Create Application Item' dialog. At the top, there's a breadcrumb navigation: Application 15726 \ Shared Components \ Application Items \ Create / Edit. On the right, there are three buttons: 'Cancel', 'Create Application Item' (which is highlighted with a red box and labeled '3'), and another button with a question mark icon. Below the buttons, there are tabs: Show All (selected), Name, Security, Configuration, and Comments. The 'Name' tab is active. In the 'Name' section, there's a label 'Application:' followed by '15726 ACME Shop' with a help icon. Below it, there are two input fields: '* Name' containing 'SHOPPING_CART_ICON' (highlighted with a red box and labeled '1') and '* Scope' containing 'Application' (highlighted with a red box and labeled '2').

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**.

The screenshot shows the 'Shared Components' page with the 'Application Items' tab selected. At the top, there's a breadcrumb navigation: Application 15726 \ Shared Components \ Application Items. Below the navigation, a message says 'Item processed.' with a checkmark icon. There are three tabs: Application Items (selected), Utilization, and History. At the top right, there are three buttons: 'Reset', 'Create' (which is highlighted with a red box), and another button with a question mark icon. Below the tabs is a search bar with a magnifying glass icon and a 'Go' button. The main area is a table with columns: Name, Computed On, Updated, Updated By, Protection Level, Escape Special Characters, and Scope. Two rows are listed: 'SHOPPING_CART_ICON' and 'SHOPPING_CART_ITEMS'. Both rows have '-' in the 'Computed On' column and '21 seconds ago' or '1 seconds ago' in the 'Updated' column. 'LOWCODE' is listed in both 'Updated By' columns. 'Protected - May not be set from browser' is listed in both 'Protection Level' columns. 'Yes' is listed in both 'Escape Special Characters' columns. 'Application' is listed in both 'Scope' columns. At the bottom right of the table, there's a label '1 - 2'.

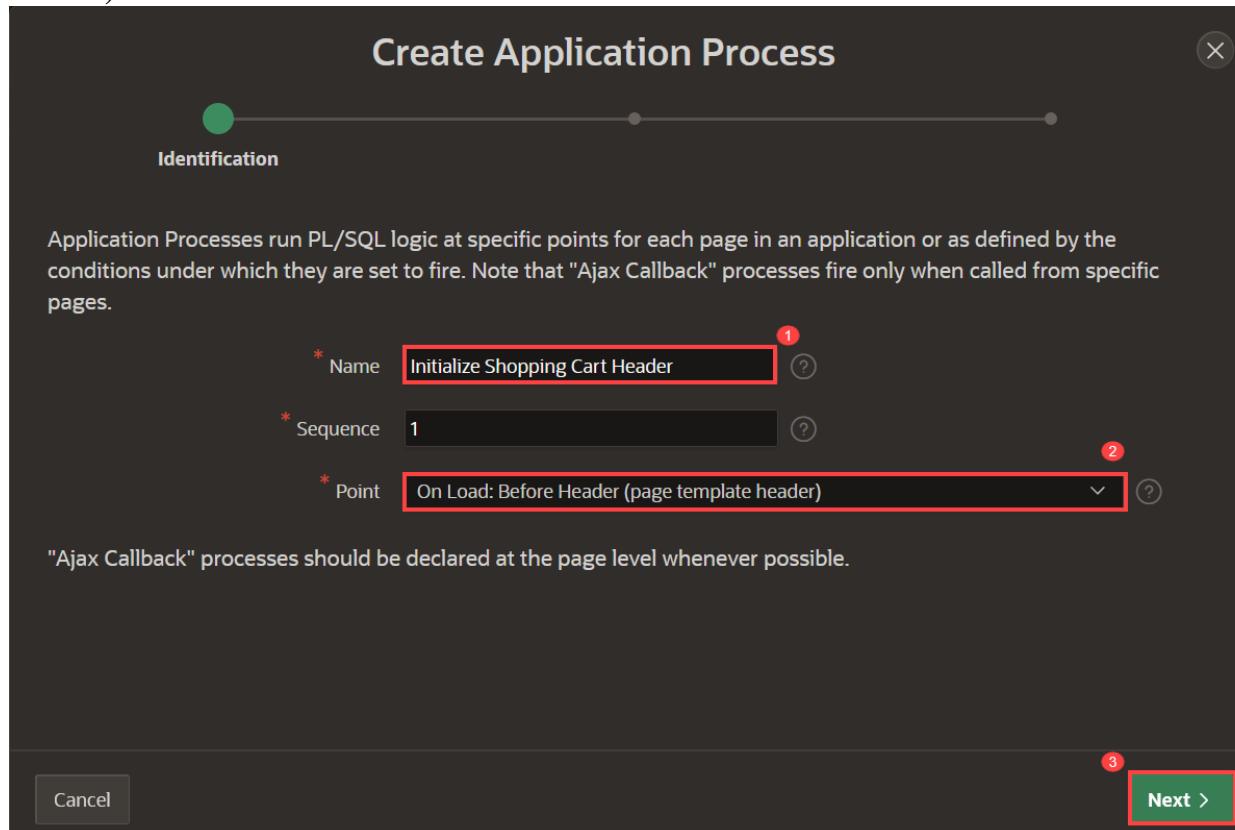
2. Under Application Logic, click **Application Processes**.

The screenshot shows the 'Shared Components' page in Oracle APEX. The 'Application Logic' section contains several items: 'Application Definition', 'Application Items' (with a count of 2), 'Application Processes' (highlighted with a red box), 'Application Computations', 'Application Settings' (with a count of 1), and 'Build Options' (with a count of 2). The 'Security' section contains: 'Security Attributes', 'Authentication Schemes' (with a count of 1), 'Authorization Schemes' (with a count of 3), 'Application Access Control', and 'Session State Protection'. The 'Other Components' section contains: 'Lists of Values' (with a count of 5), 'Plug-ins', 'Component Settings', and 'Shortcuts'. The 'Navigation' section contains: 'Lists' (with a count of 4), 'Navigation Menu', 'Breadcrumbs' (with a count of 1), and 'Navigation Bar List'. The 'User Interface' section contains: 'User Interface Attributes', 'Progressive Web App', 'Themes' (with a count of 1), 'Templates' (with a count of 67), and 'Email Templates'. The 'Files and Reports' section contains: 'Static Application Files' (with a count of 3), 'Static Workspace Files', 'Report Queries', and 'Report Layouts'.

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

```

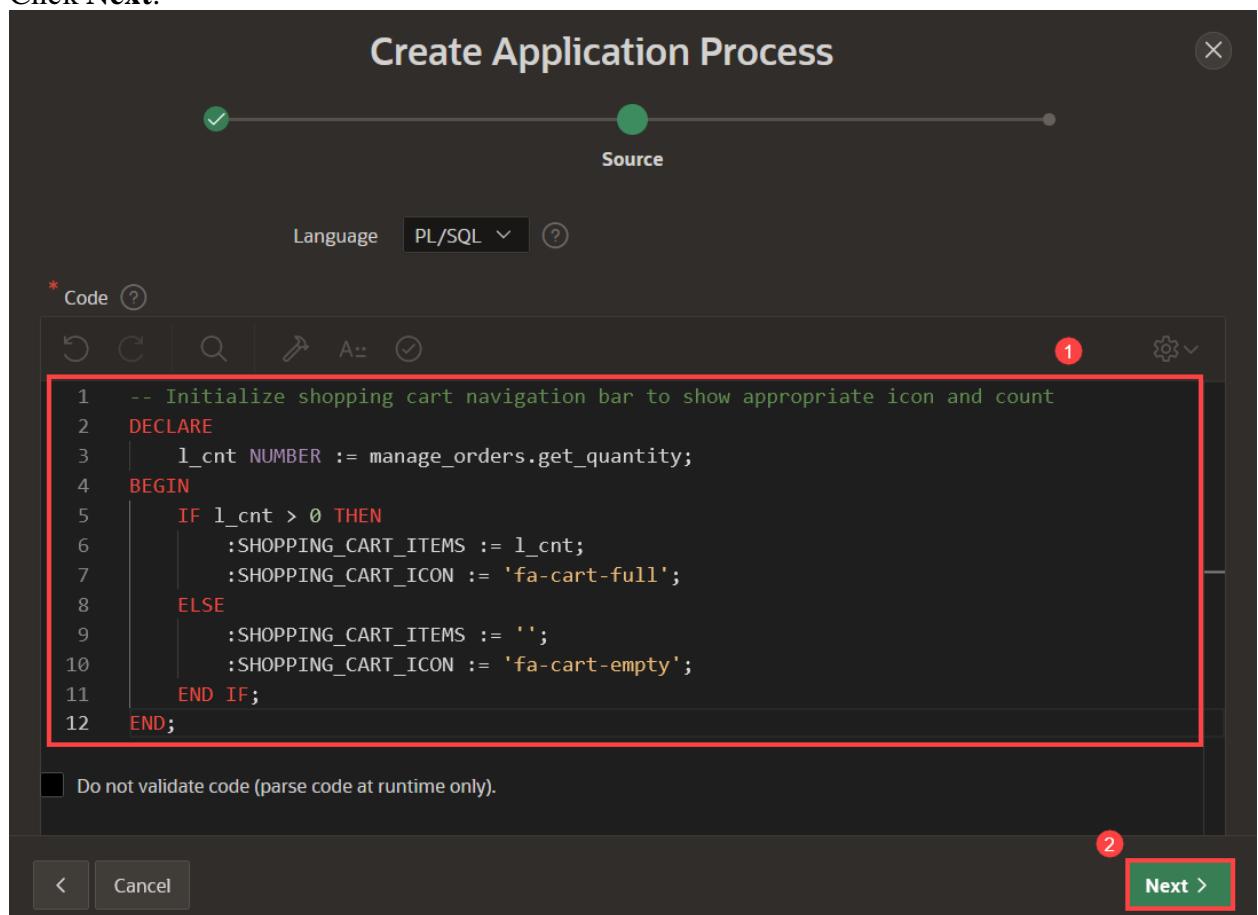
```

13.      :SHOPPING_CART_ITEMS := '';
14.      :SHOPPING_CART_ICON := 'fa-cart-empty';
15.  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.***

The screenshot shows the Oracle Application Home interface. The top navigation bar displays 'Application 15726 \ Shared Components \ Application Processes'. Below the navigation is a toolbar with search, go, and actions buttons. A table lists application processes, with one row highlighted:

| Sequence | Name | Point | Text | Authorization Scheme |
|----------|---------------------------------|---|--|----------------------|
| 1 | Initialize Shopping Cart Header | On Load: Before Header (page template header) | -- Initialize shopping cart navigation bar to show appropriate icon and count DECLARE l_cnt NUMBER := manage_orders.get_quantity; BEGIN IF l_cnt > 0 THEN :SHOPPING_CART_ITEMS := i ... | - |

2. Click Create Page.

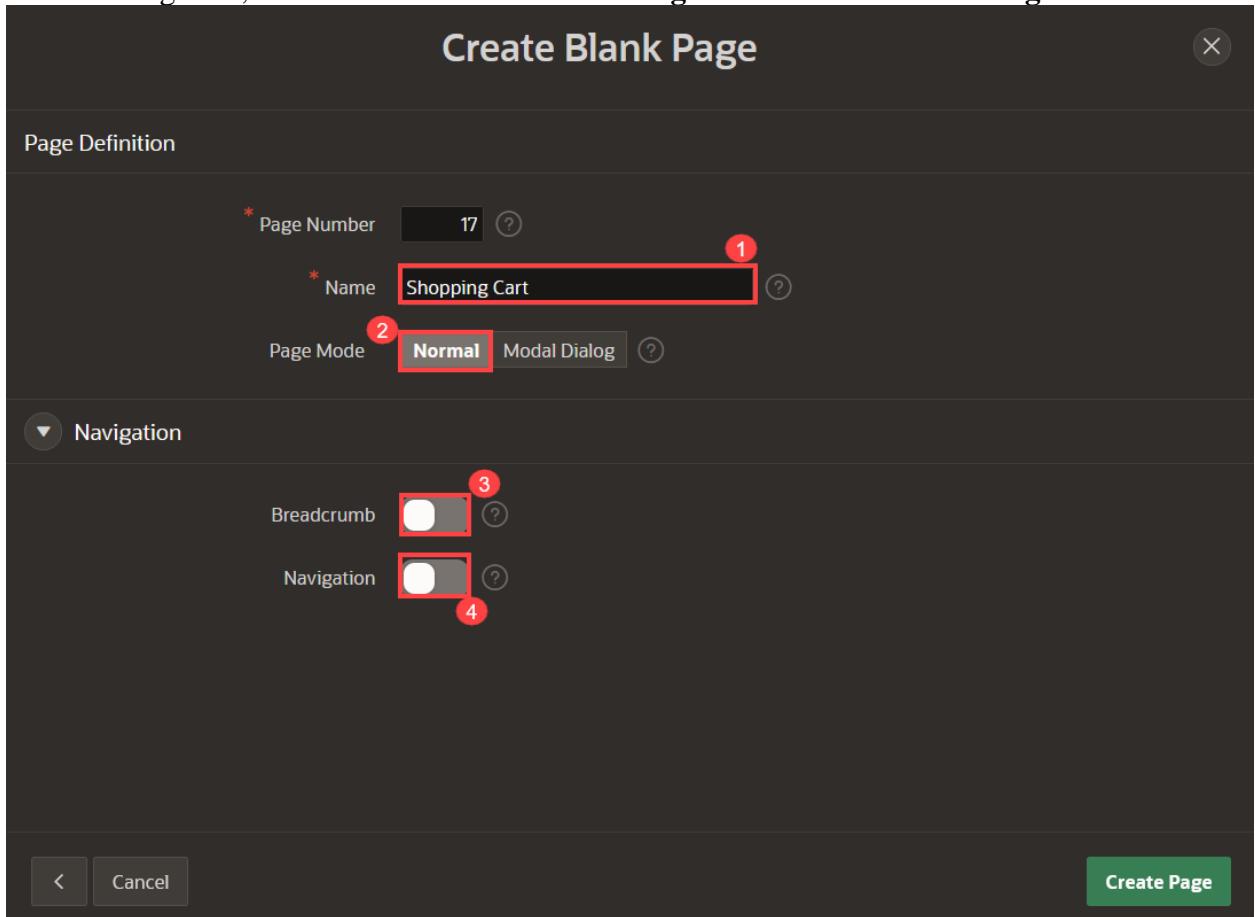
The screenshot shows the Oracle Application Home interface for 'Application 15726 - ACME Shop'. The top navigation bar shows the application name. Below it is a toolbar with various icons for running the application, managing objects, and shared components. A main area displays several icons for different pages and utilities. At the bottom right of the toolbar, there is a green button labeled 'Create Page >' which is highlighted with a red box.

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

4. Enter the following and click **Next**.

- o Page Number - enter **17**
- o For Name - enter **Shopping Cart**
- o For Page Mode - select **Normal**

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

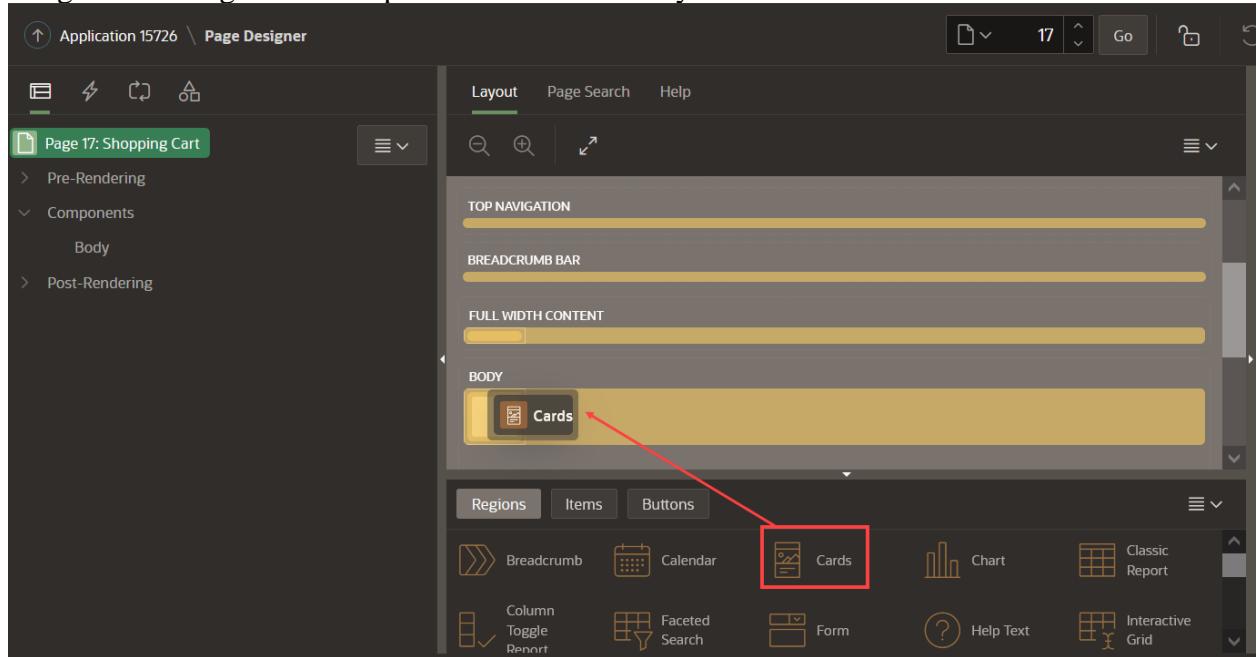


Task 4: Add a Cards Region

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:

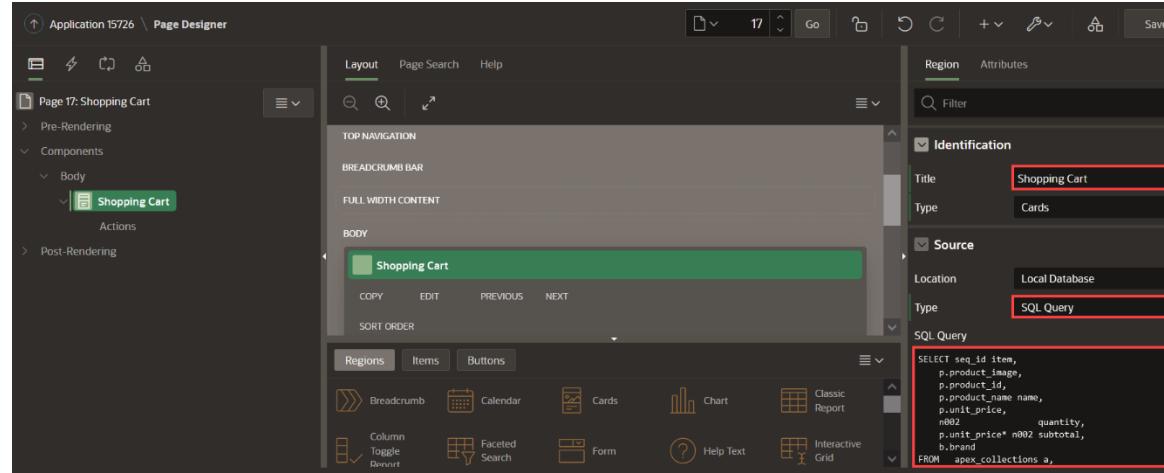
- o For Title - enter **Shopping Cart**
- o Under Source section:
 - For Type - select **SQL Query**
 - For SQL Query - enter the following SQL Query:

```

▪ CopySELECT 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'

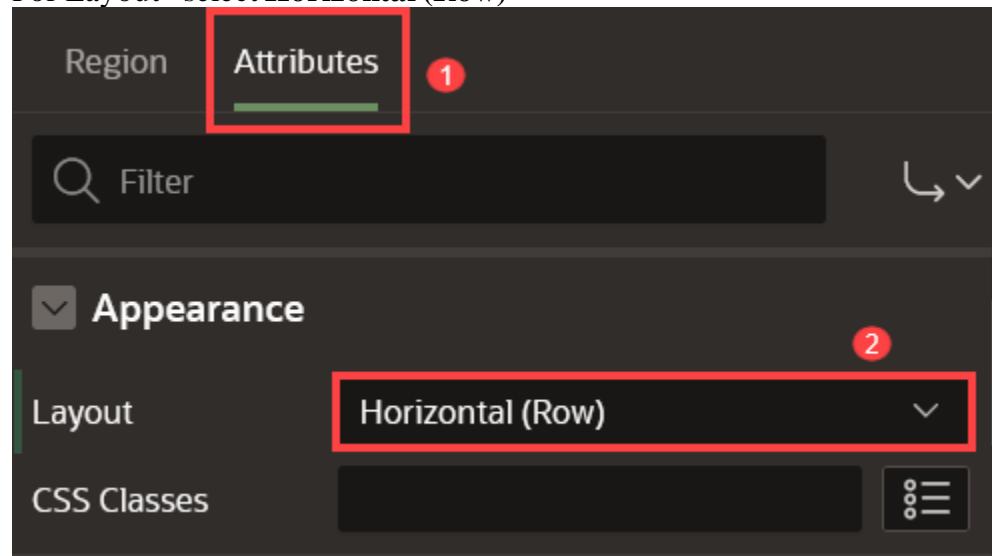
AND p.product_id = a.n001



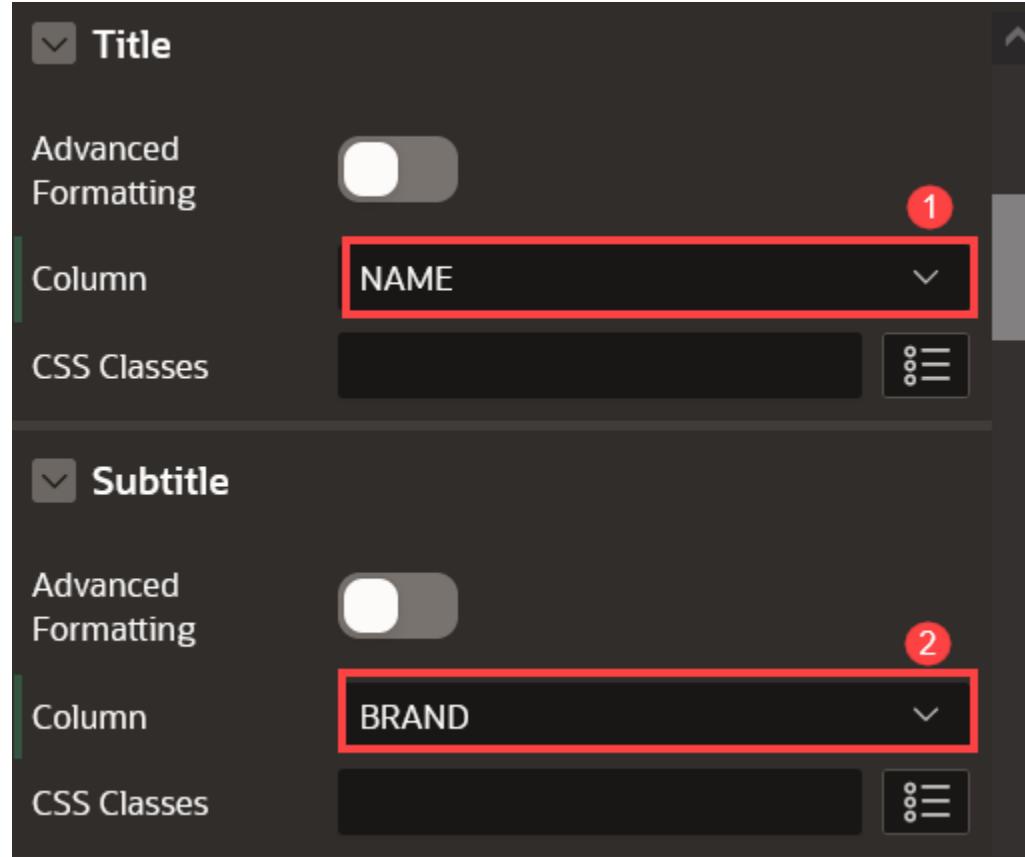
4. Click on **Attributes** and enter the following:

- 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>
```

Region Attributes

Filter

Body

Advanced Formatting  1

HTML Expression

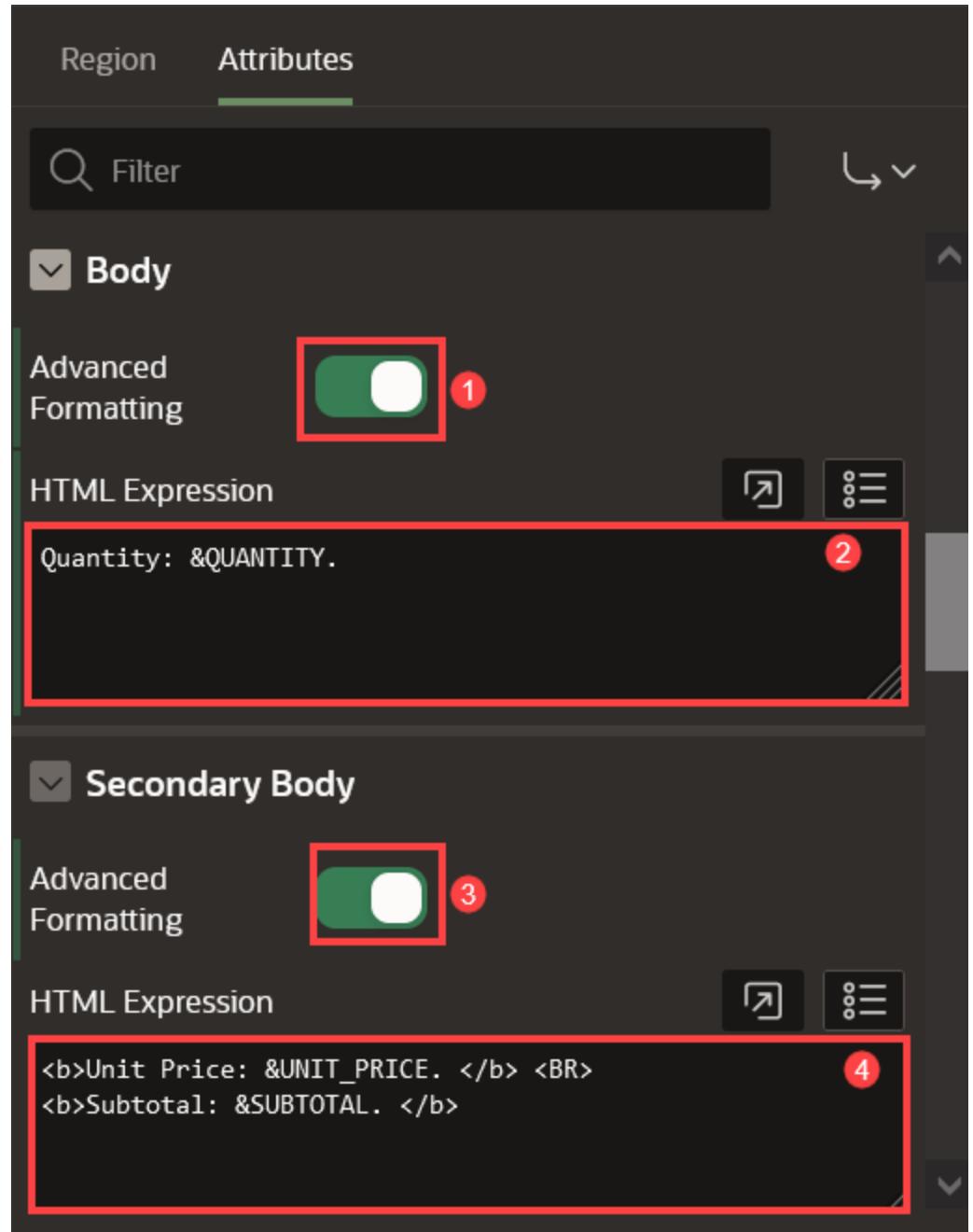
Quantity: &QUANTITY. 2

Secondary Body

Advanced Formatting  3

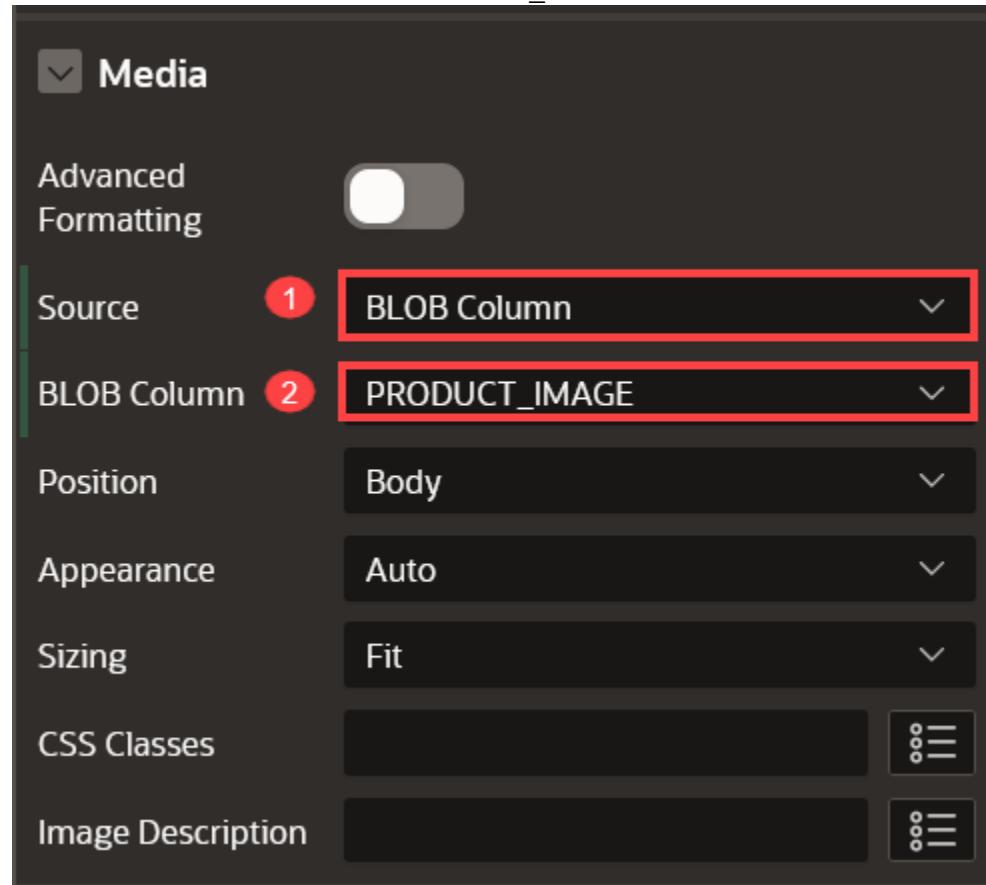
HTML Expression

Unit Price: &UNIT_PRICE.
Subtotal: &SUBTOTAL. 4



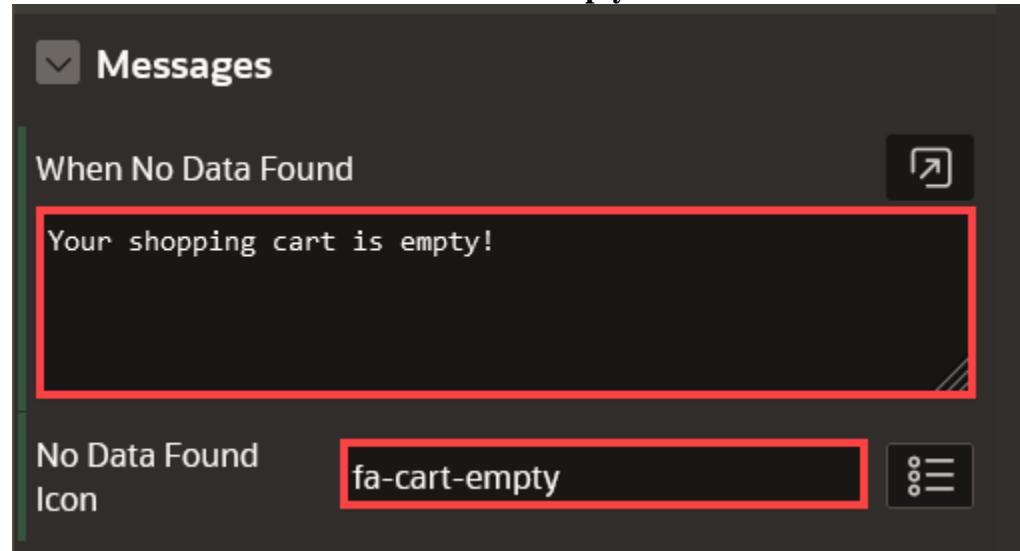
- Under Media section:
 - For Source - select **BLOB Column**

- For BLOB Column - select **PRODUCT_IMAGE**



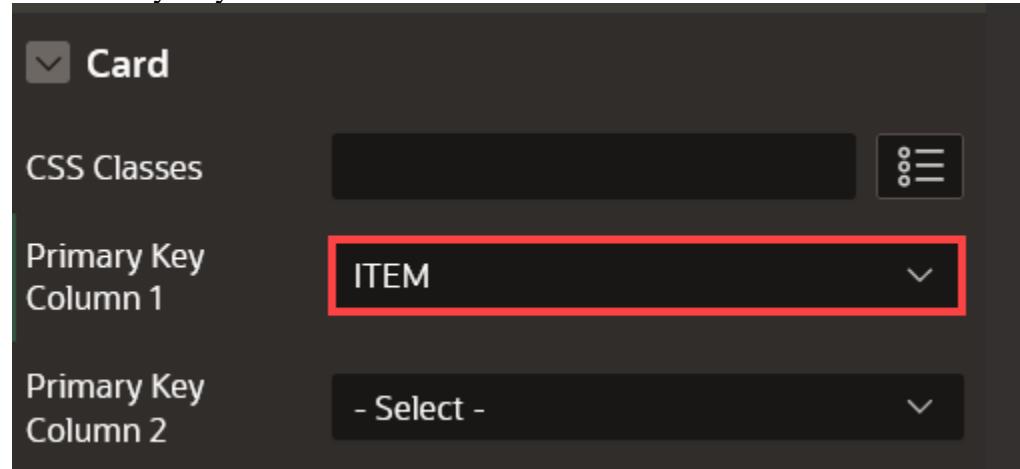
- Under Messages section:

- 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**

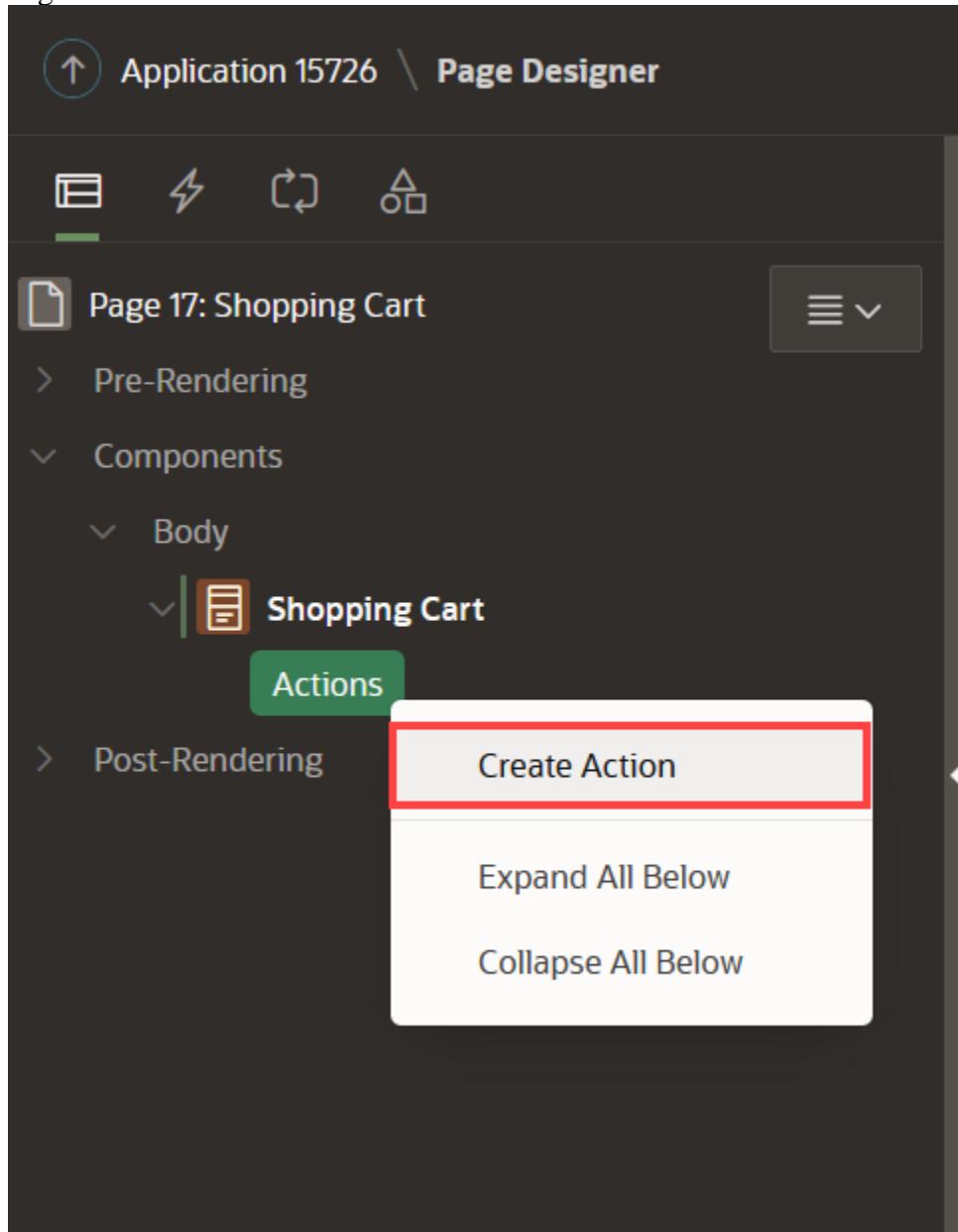


Task 5: Add an Action to the Shopping Cart

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**.



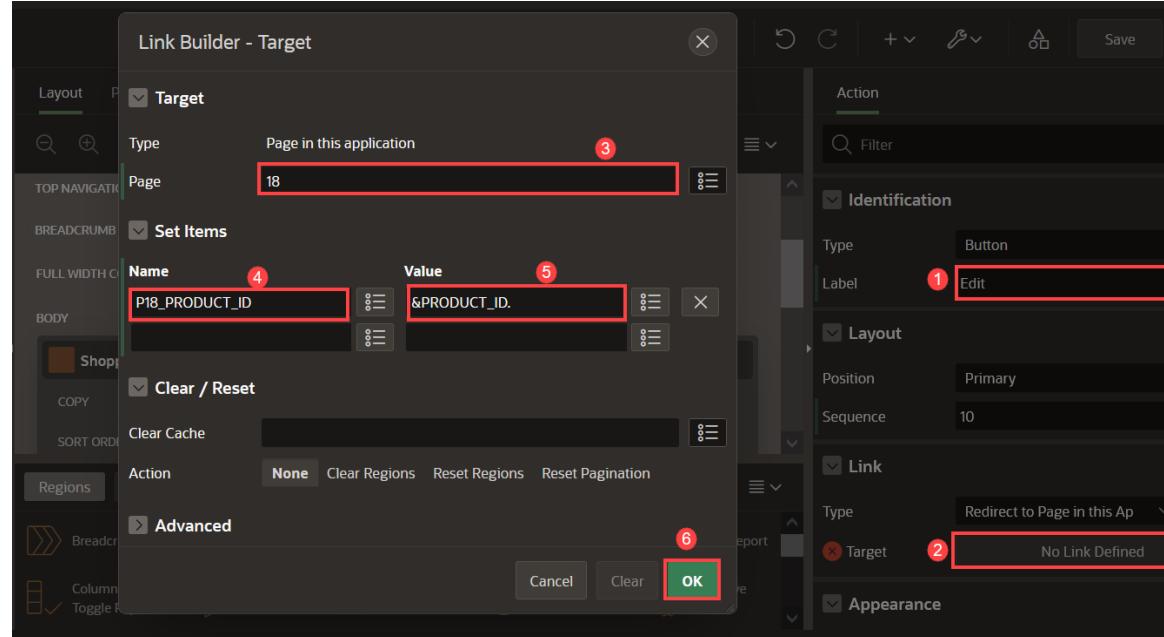
3. In the Property Editor, enter the following:

- 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

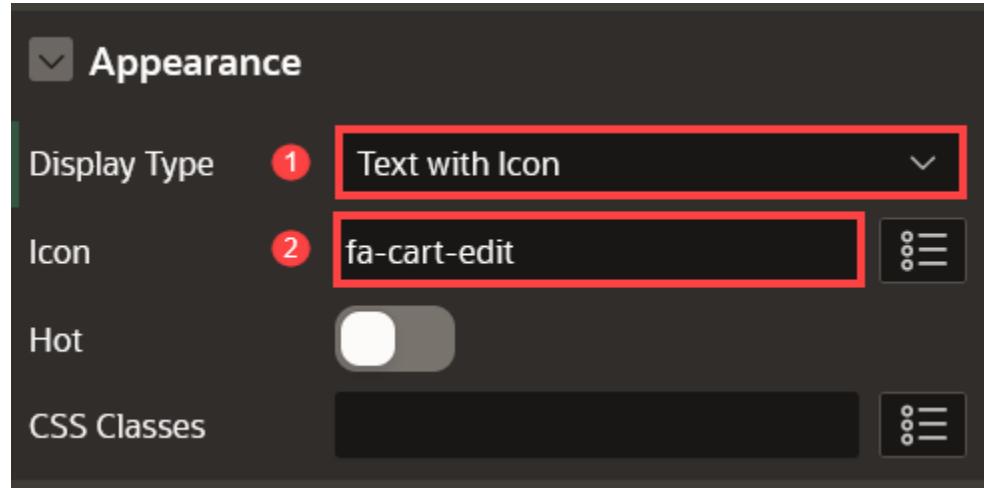
| Name | Value |
|----------------|--------------|
| P18_PRODUCT_ID | &PRODUCT_ID. |

- Click Ok.



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

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

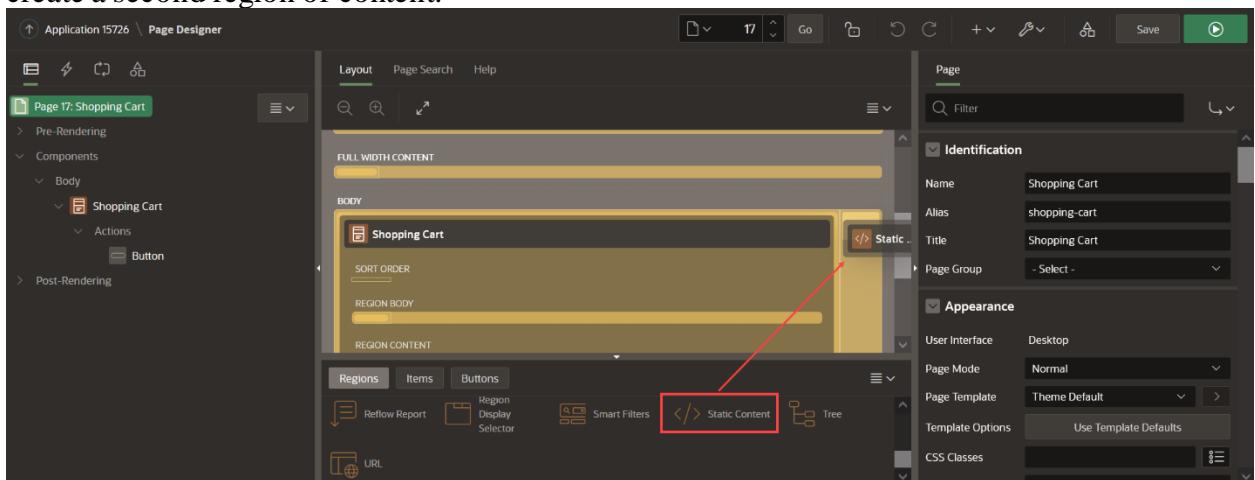


This

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.

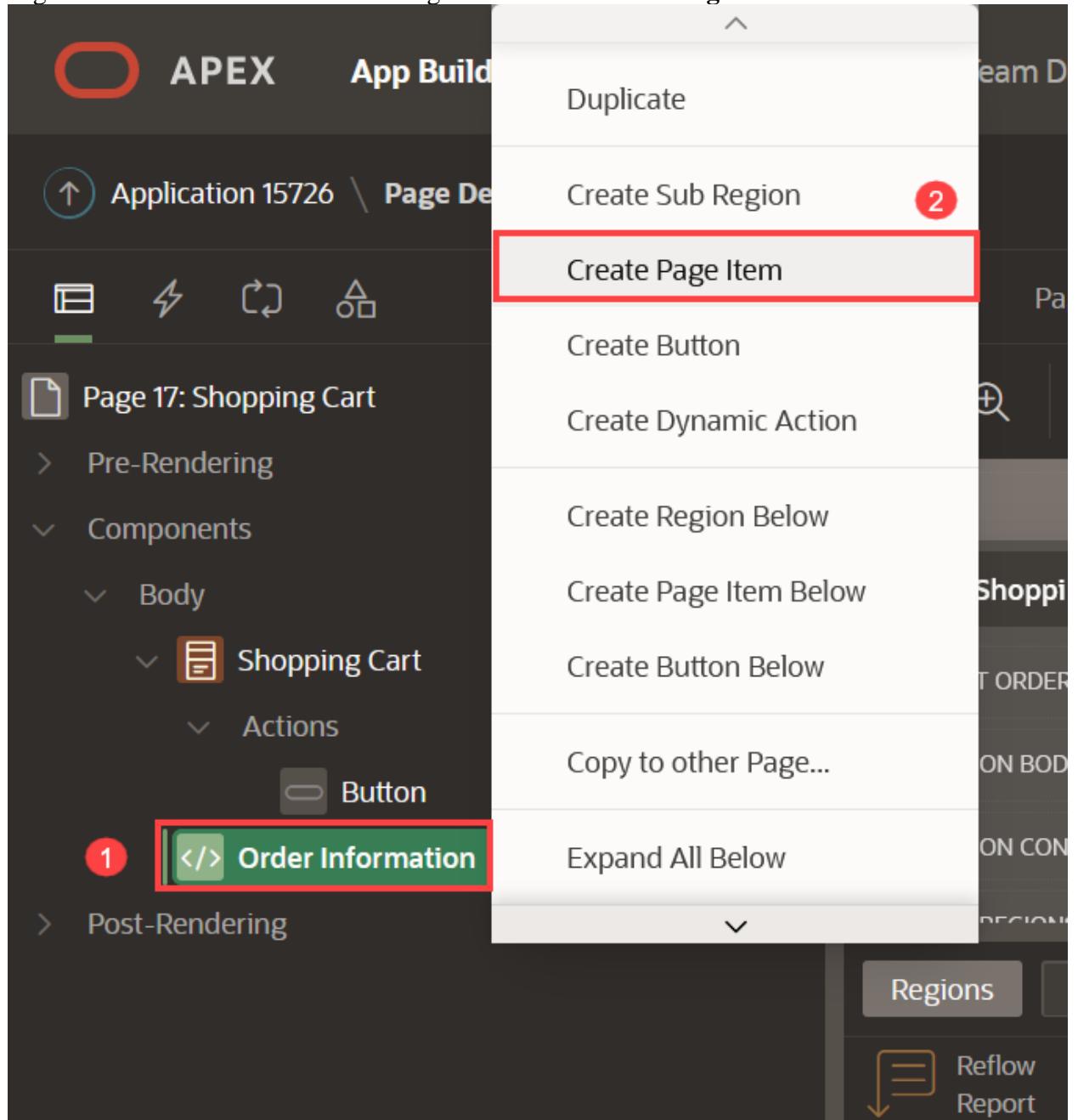
Task 6: Add Items and Buttons to the Page

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:

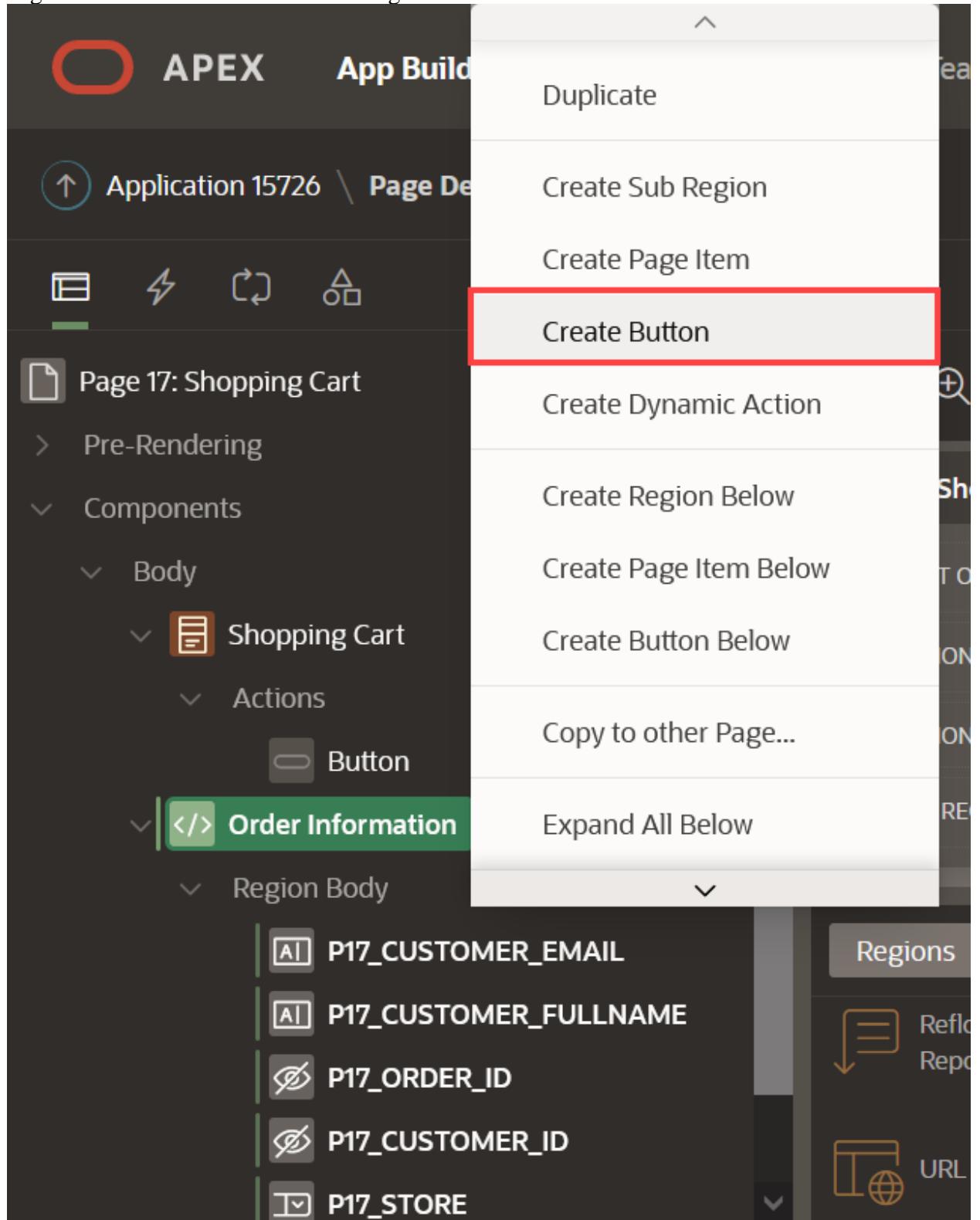
- For Type - select **SQL Query**
- For SQL Query - enter the following SQL Query:

- Copy **select 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:

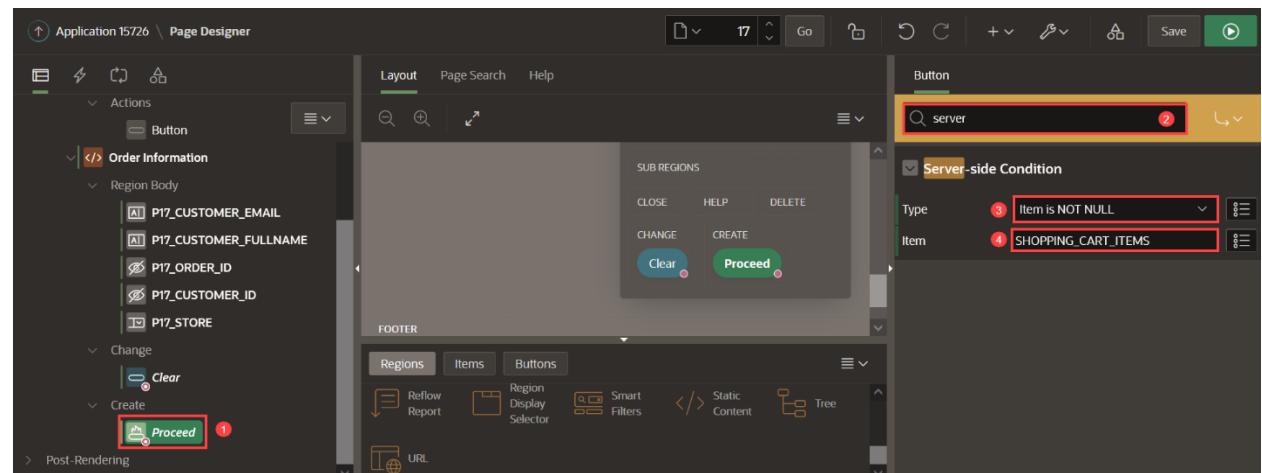
Table 4: Details of the 2 Buttons

| 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

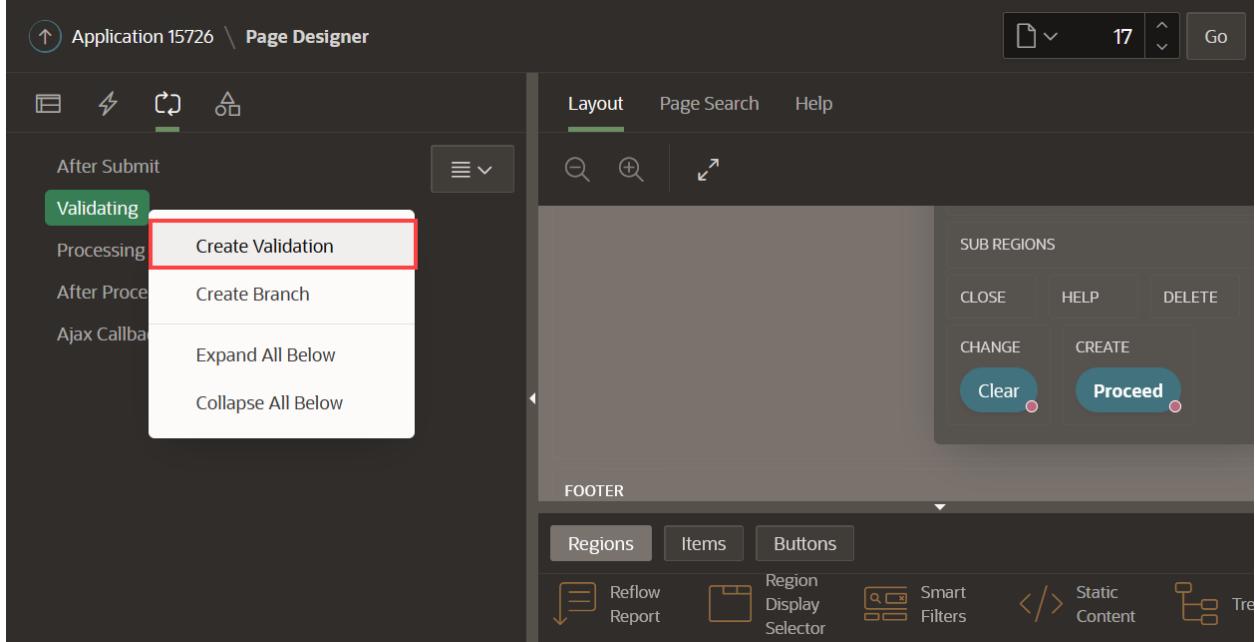
| Button Name | Type | Item |
|--------------------|------------------|---------------------|
| Proceed | Item is NOT NULL | SHOPPING_CART_ITEMS |
| Clear | Item is NOT NULL | SHOPPING_CART_ITEMS |



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**

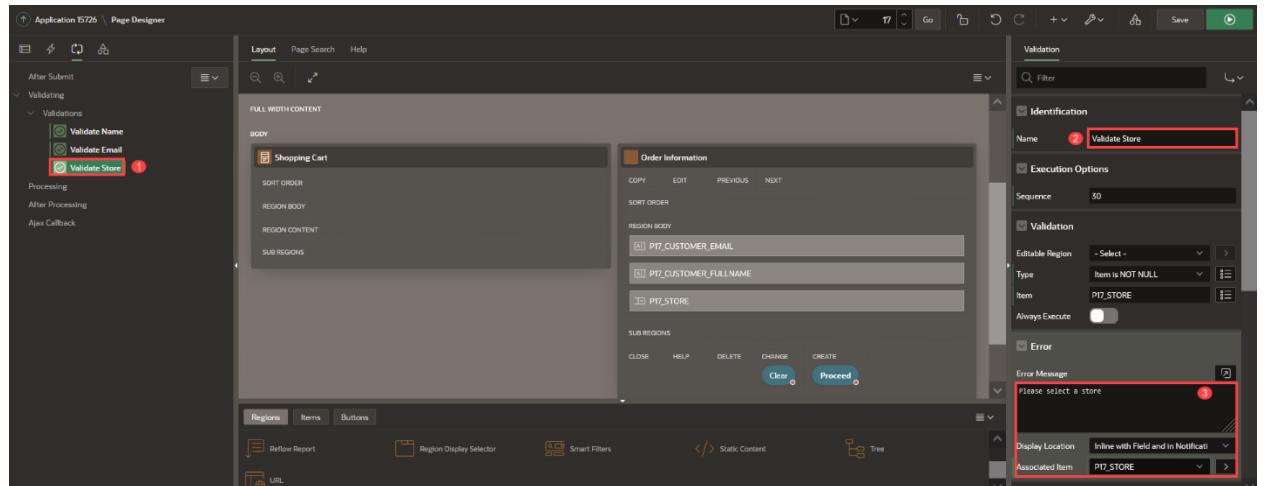
Table 6: Validations for the Items

| 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.

- 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 |

Validation

Filter ↻

Identification

| | |
|------|----------------|
| Name | Validate Store |
|------|----------------|

Execution Options

| | |
|----------|----|
| Sequence | 30 |
|----------|----|

Validation

| | | |
|-----------------|--------------------------|---|
| Editable Region | - Select - | > |
| Type | Item is NOT NULL | ⋮ |
| Item | P17_STORE | ⋮ |
| Always Execute | <input type="checkbox"/> | |

Error

| | |
|-----------------------|---|
| Error Message | ↗ |
| Please select a store | |

Display Location

| | | |
|-----------------|---------------------------------------|---|
| Associated Item | Inline with Field and in Notification | > |
|-----------------|---------------------------------------|---|

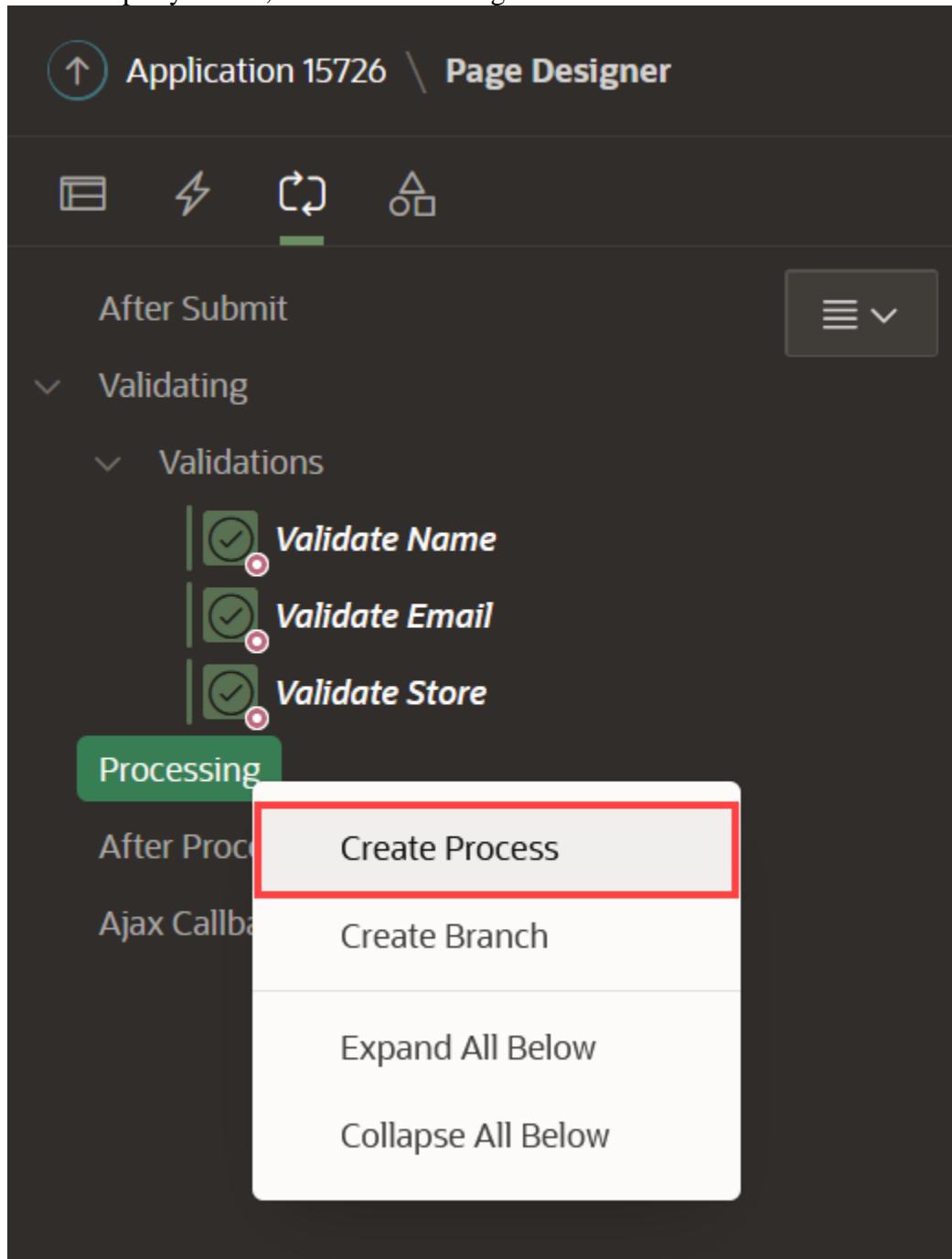
Server-side Condition

| | | |
|---------------------|------------|---|
| When Button Pressed | Proceed | > |
| Type | - Select - | ⋮ |

7.

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:

```

○ CopyBEGIN

○      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   =>  
        :P17_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:
 - o For Name - enter **Clear Shopping Cart**
 - o For Type - select **Execute Code**
 - o For PL/SQL Code - enter the following PL/SQL code:

```
4. CopyBEGIN  
5.      manage_orders.clear_cart;
```

```
END;
```

- o 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**

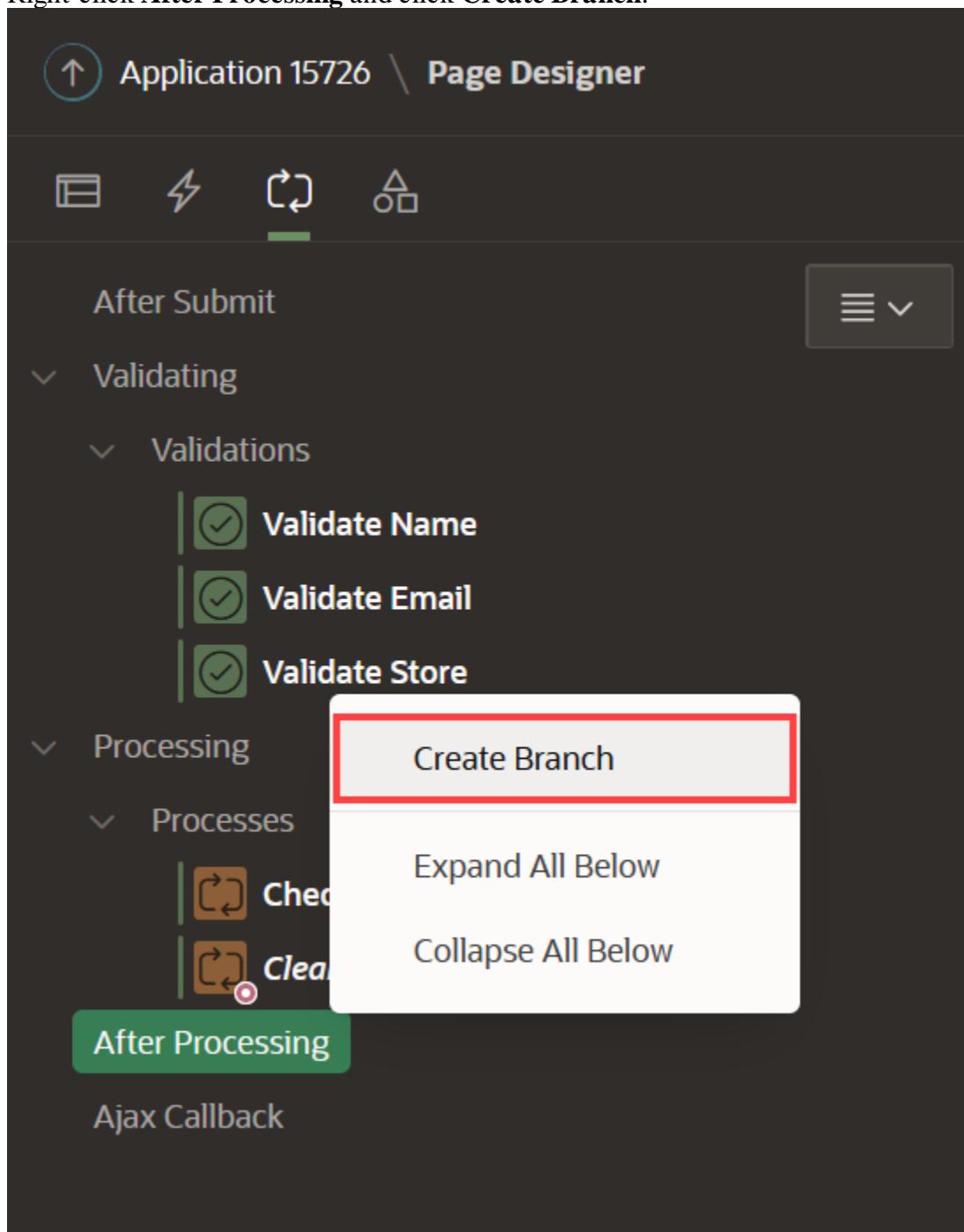
When Button Pressed Clear

Type - Select -

The screenshot shows the configuration interface for an APEX item. Three specific fields are highlighted with red boxes and numbered circles: 'Name' (highlighted at 1) containing 'Clear Shopping Cart', 'PL/SQL Code' (highlighted at 2) containing the code block, and 'When Button Pressed' (highlighted at 3) containing 'Clear'. The interface includes sections for Identification, Source, Execution Options, Success Message, Error, and Server-side Condition.

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:

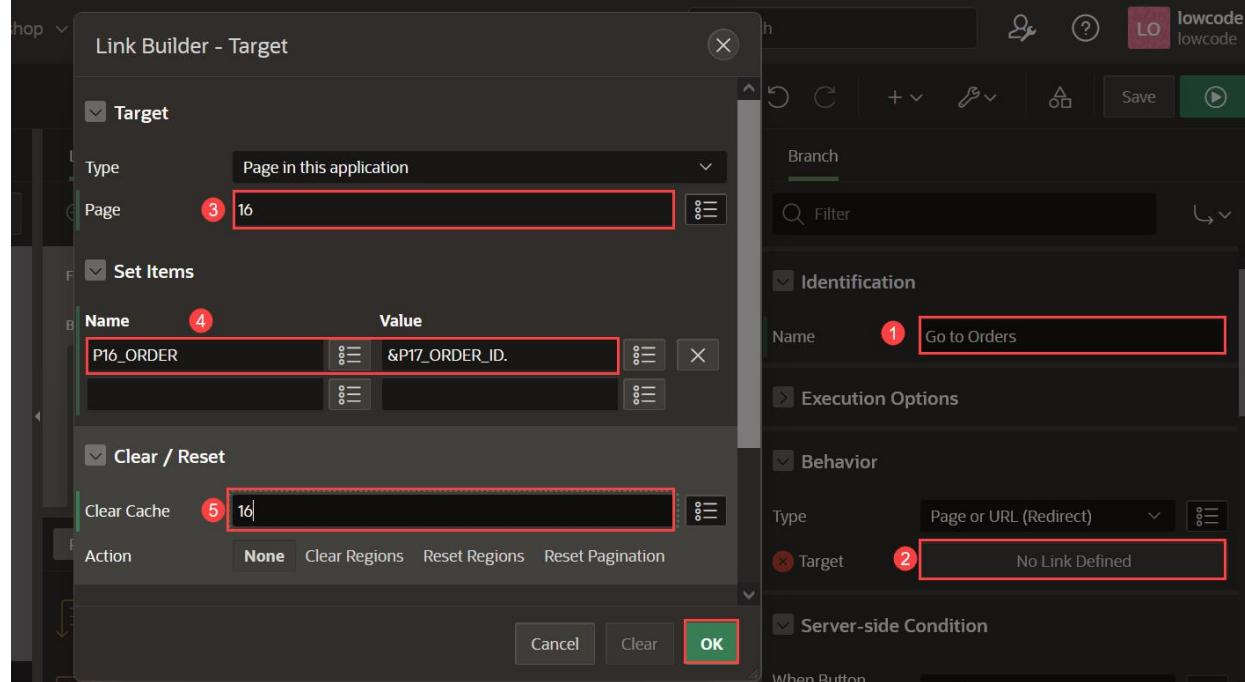
- For Name - enter **Go to Orders**
- Navigate to Target attribute and click **No Link Defined.**
 - For Type - select **Page in this application**
 - For Page - enter **16**
 - For Set Items - enter:

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**.



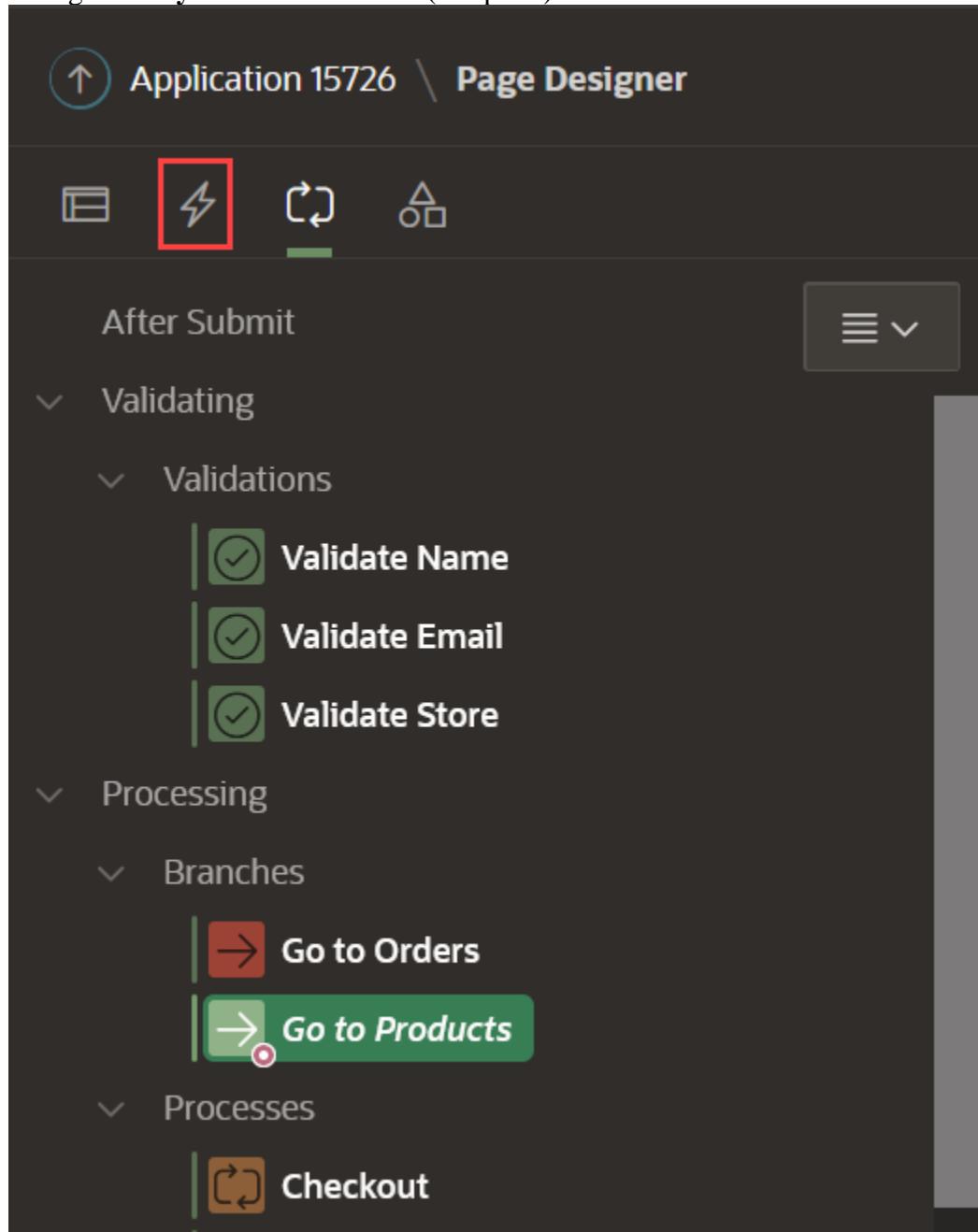
4. Create a second branch when user clears the shopping cart. Right-click on **After Processing** and click **Create Branch**.
5. In the Property Editor, enter the following:
 - For Name - enter **Go to Products**
 - Navigate to Target attribute and click **No Link Defined**
 - For Type - select **Page 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

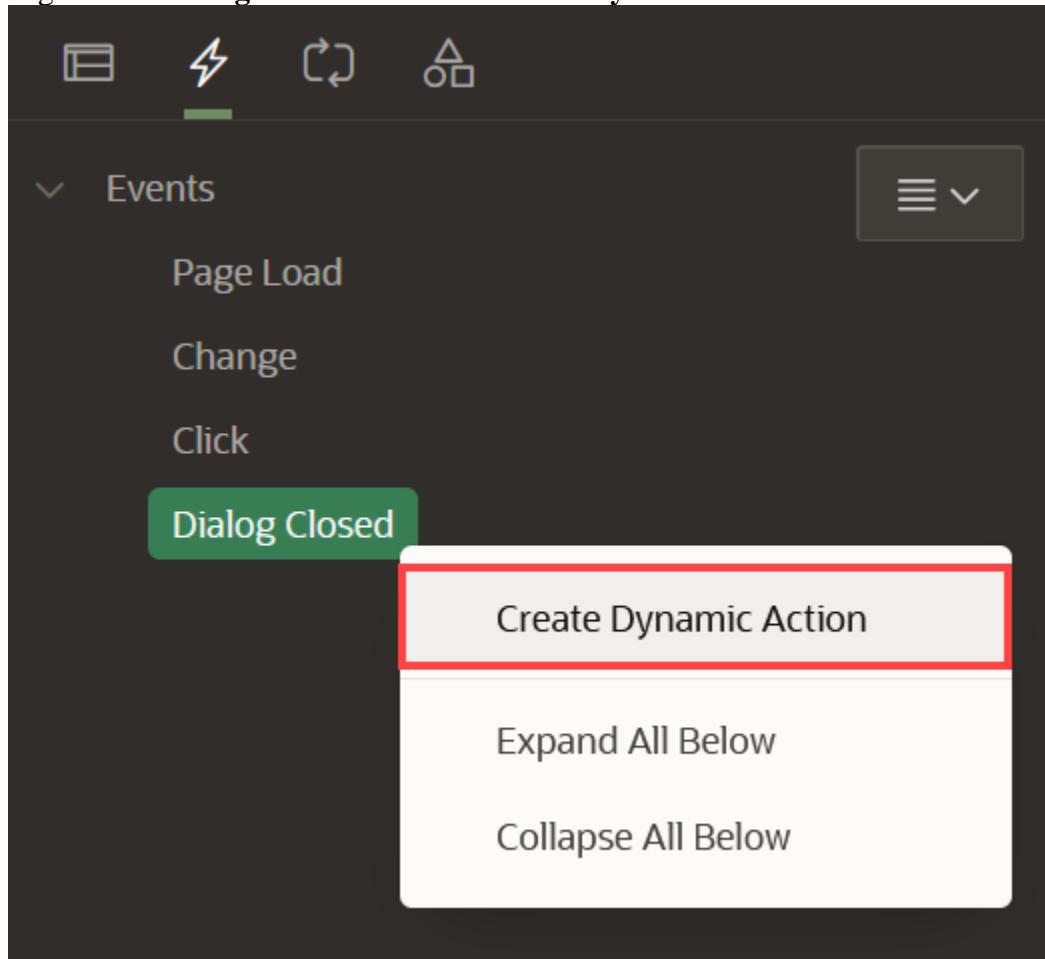
In this task, you will create a dynamic action to:

- 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:

```
CopyParseInt(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 ShoppingCart

Client-side Condition

Type ④ JavaScript expression

JavaScript Expression

```
parseInt(this.data.P18_SHOPPING_CART_ITEMS) > 0
```

⑤

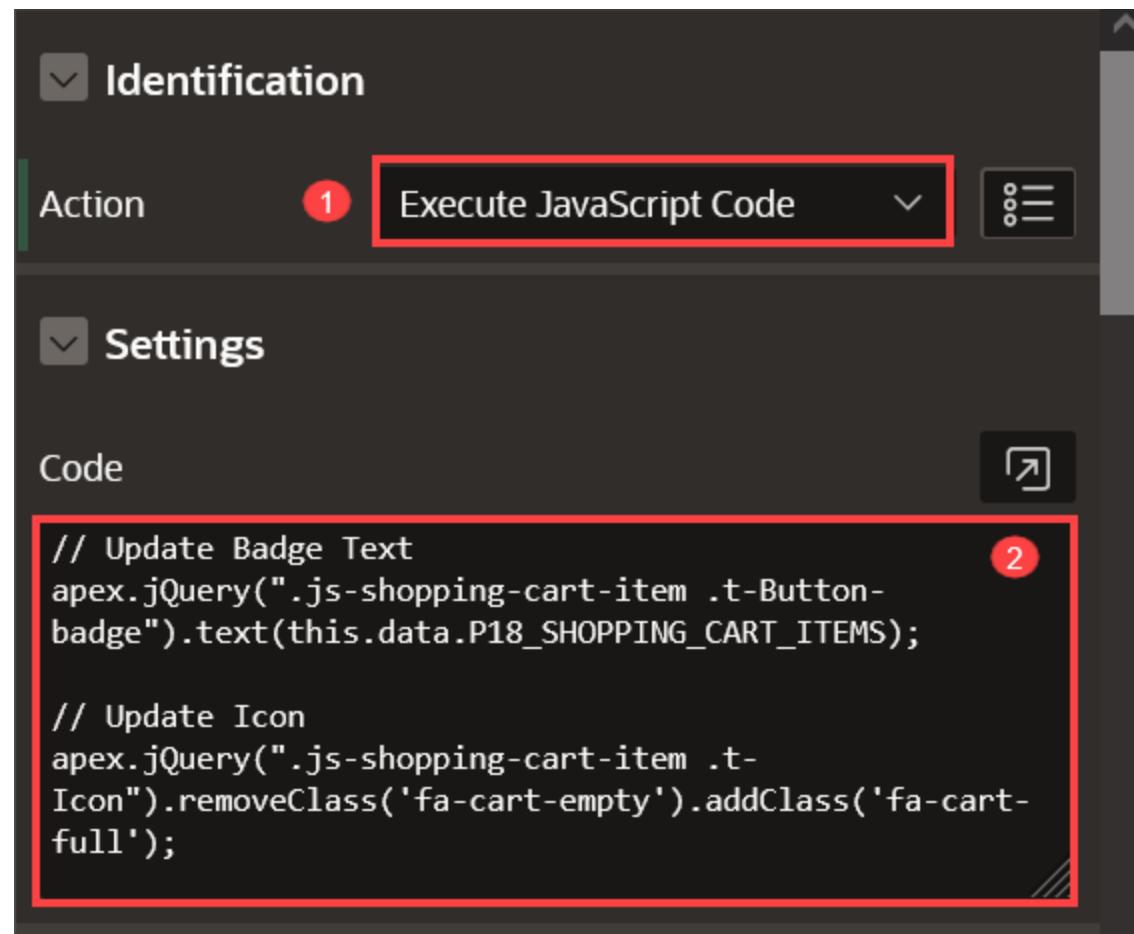
Advanced

This screenshot shows the configuration of a Dynamic Action in Oracle APEX. The action is named "Update Shopping Cart Header". It is triggered by the "Dialog Closed" event, specifically targeting the "Region" selection type and the "Shopping Cart" region. The client-side condition is defined using a JavaScript expression: `parseInt(this.data.P18_SHOPPING_CART_ITEMS) > 0`. The entire configuration is highlighted with a red border.

4. Navigate to **Refresh** Action.

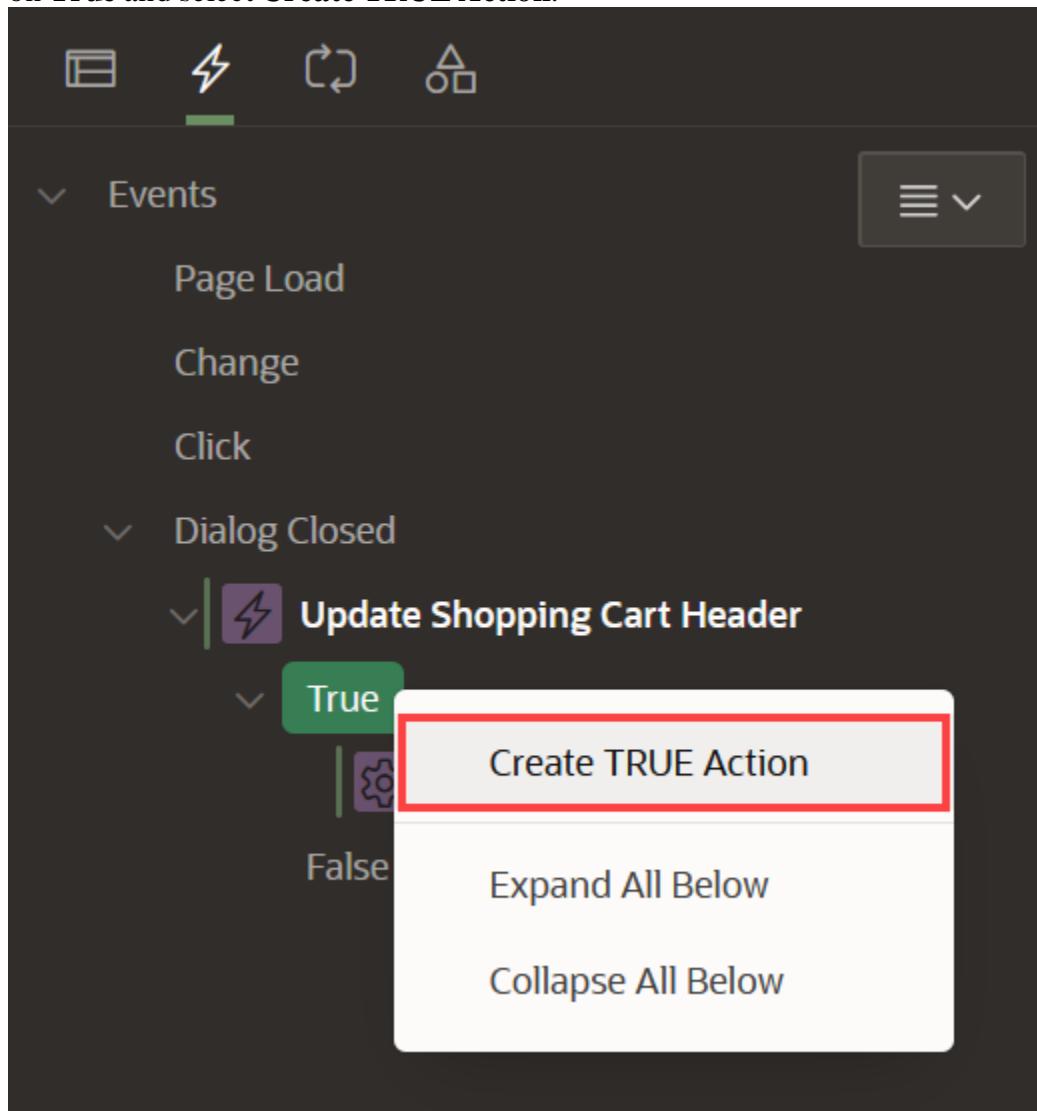
- o Under Identification section:
 - For Action - select **Execute JavaScript Code**
- o Under Settings section:
 - For Code - enter the following JavaScript Code:
 - Copy// Update Badge Text
 - `apex.jQuery(".js-shopping-cart-item .t-Button-badge") .text(this.data.P18_SHOPPING_CART_ITEMS);`
 -
 - // 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

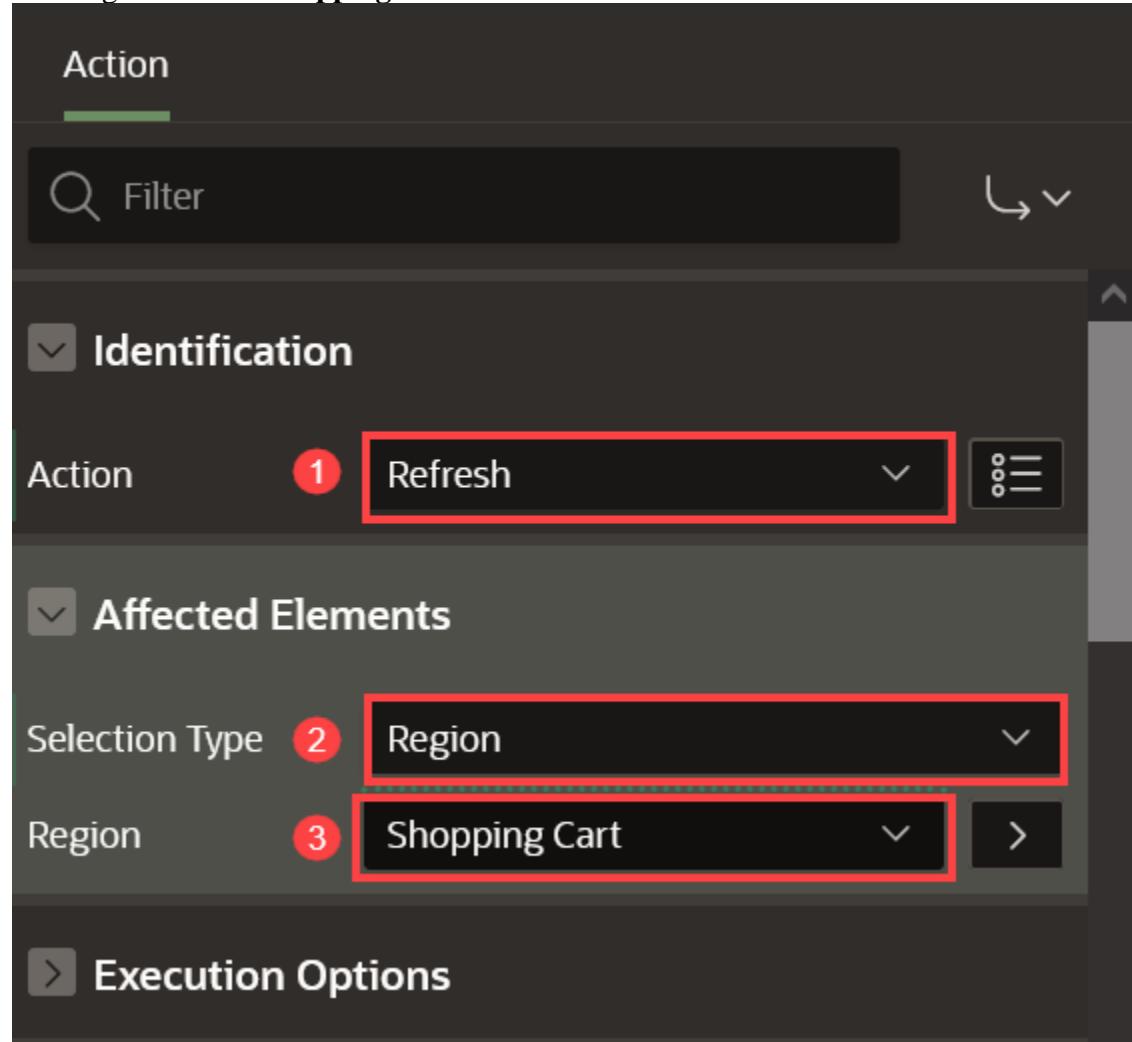
on **True** and select **Create TRUE Action**.



6. In the Property Editor, enter the following:

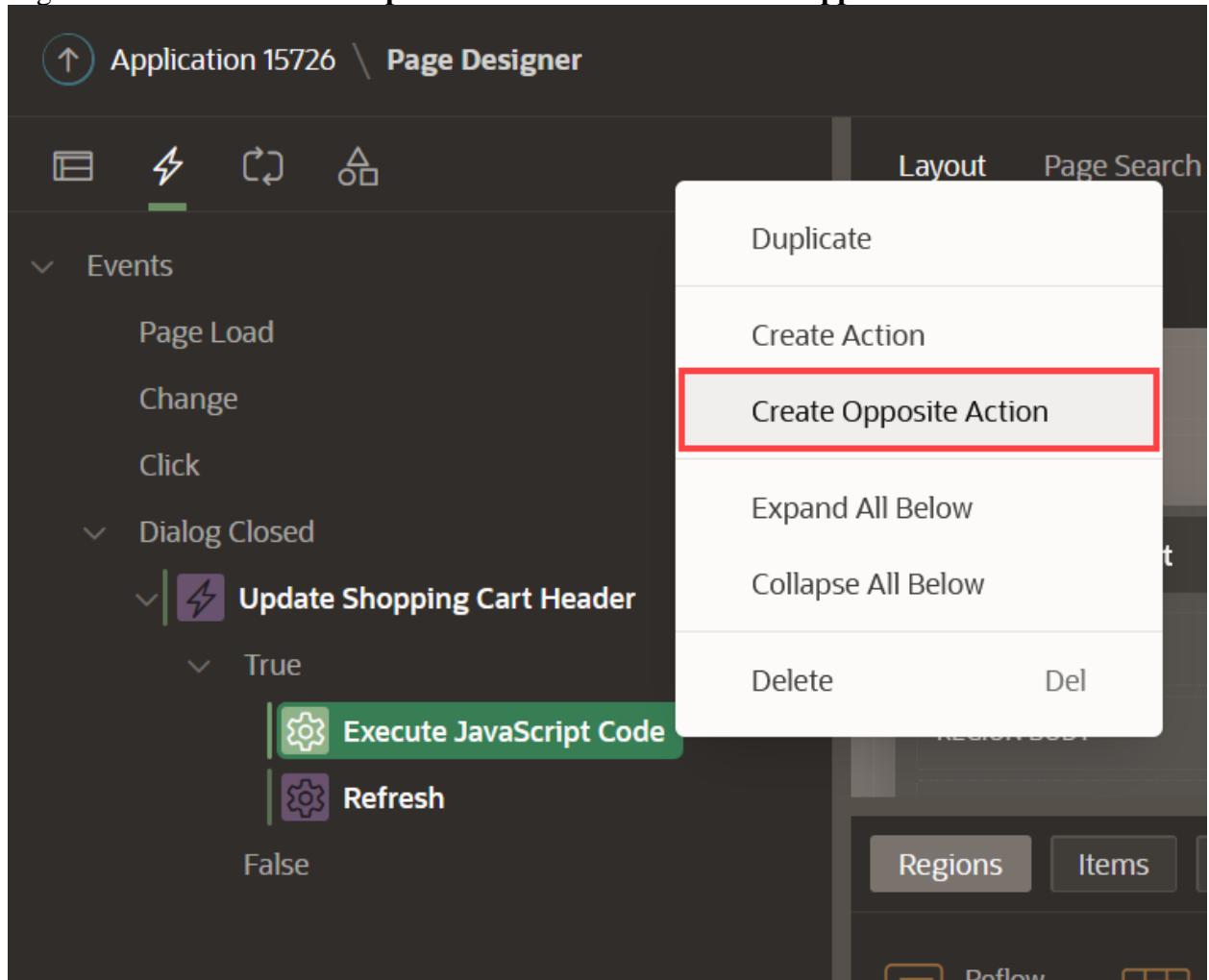
- Under Identification section:
 - For Action - select **Refresh**
- Under Affected Elements section:
 - For Selection Type - select **Region**

- 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 FALSE 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**

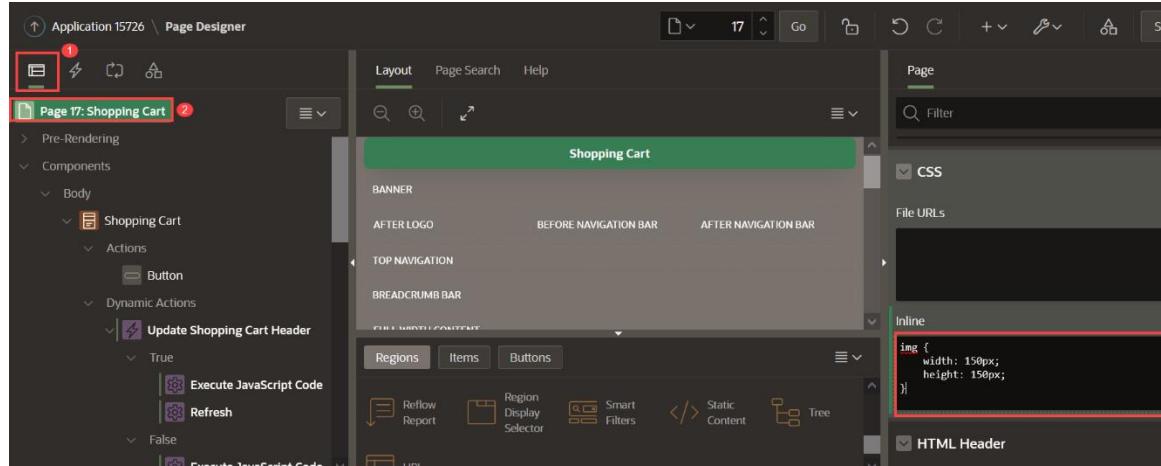
Task 12: Format Products Image Size

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:

```
▪ Copy {  
▪ 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:

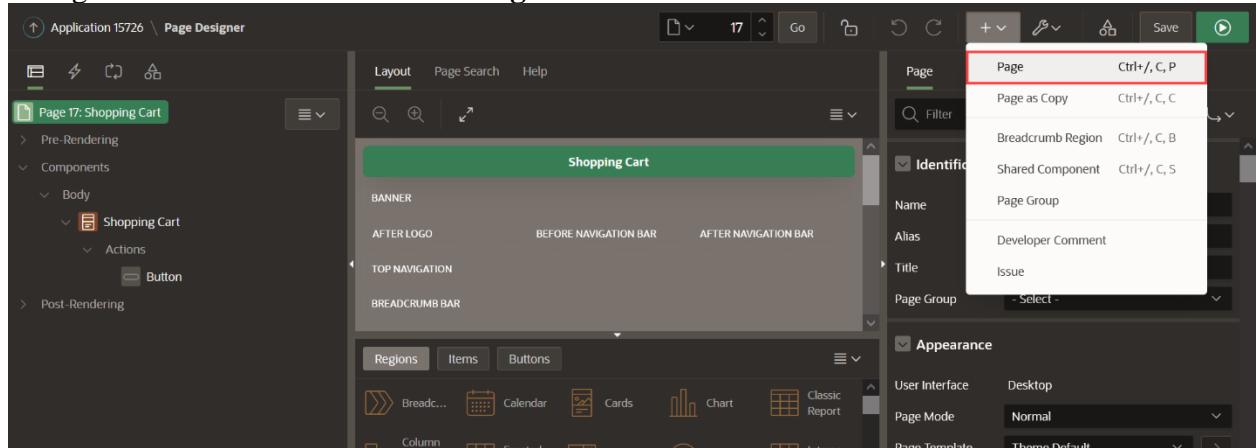
- Create a page that allows users to add and edit products in the Shopping Cart

[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**.

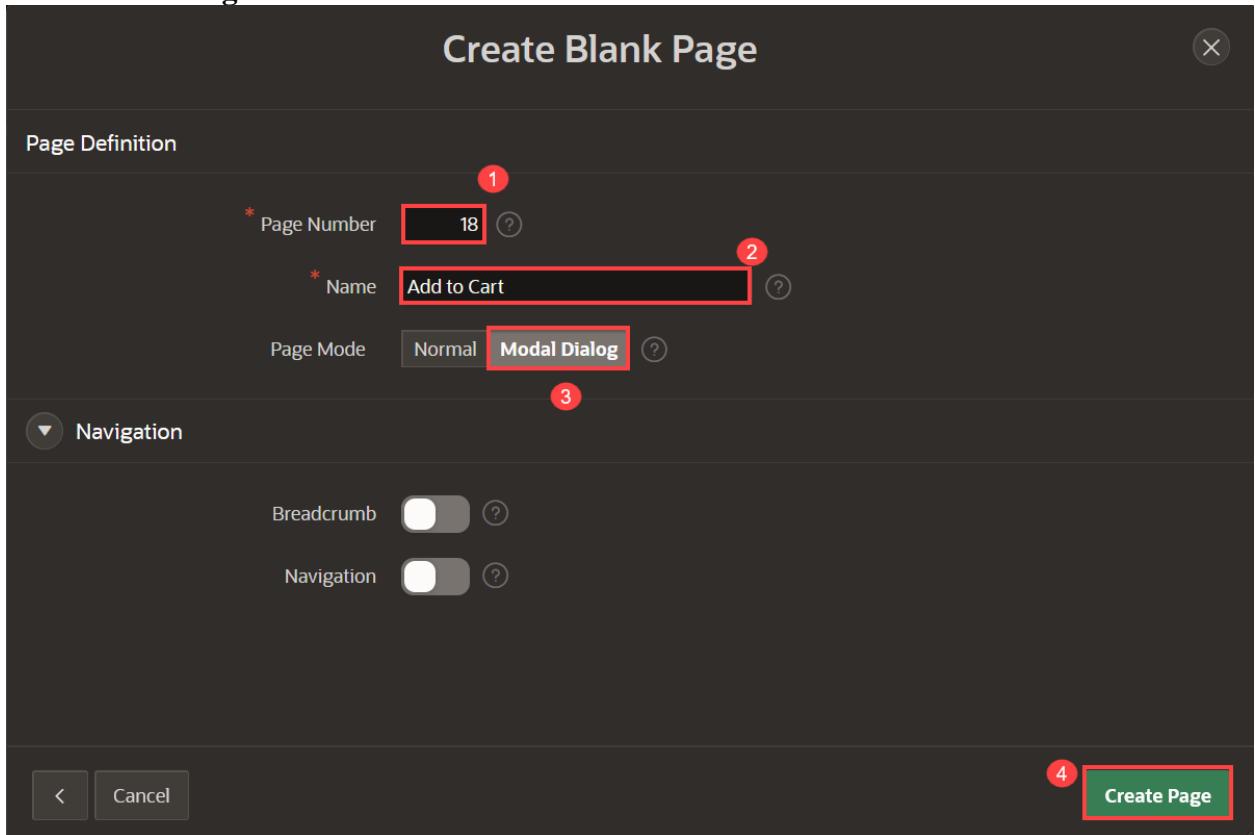


2. Select **Blank Page** and click **Next**.

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**.

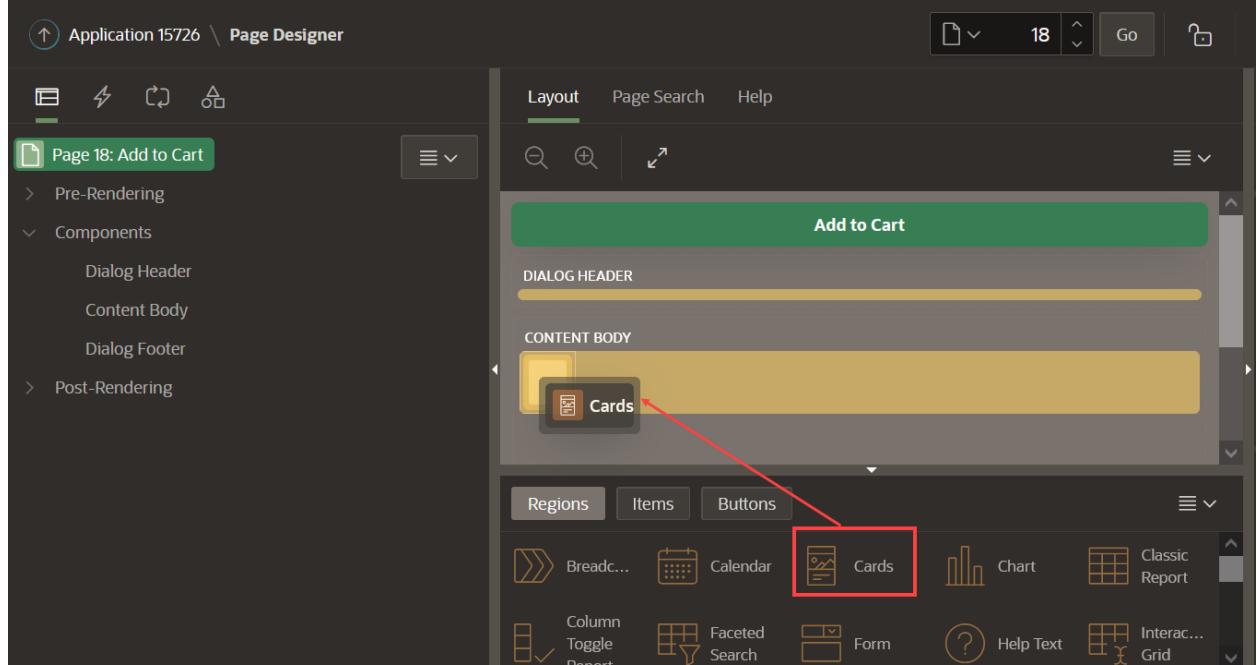


Task 2: Add Cards Region for Product Details

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:

```

      ▪ CopySELECT product_id,
      ▪       product_name,
      ▪       unit_price,
      ▪       product_details,
      ▪       product_image,
      ▪       image_mime_type,
      ▪       image_filename,
```

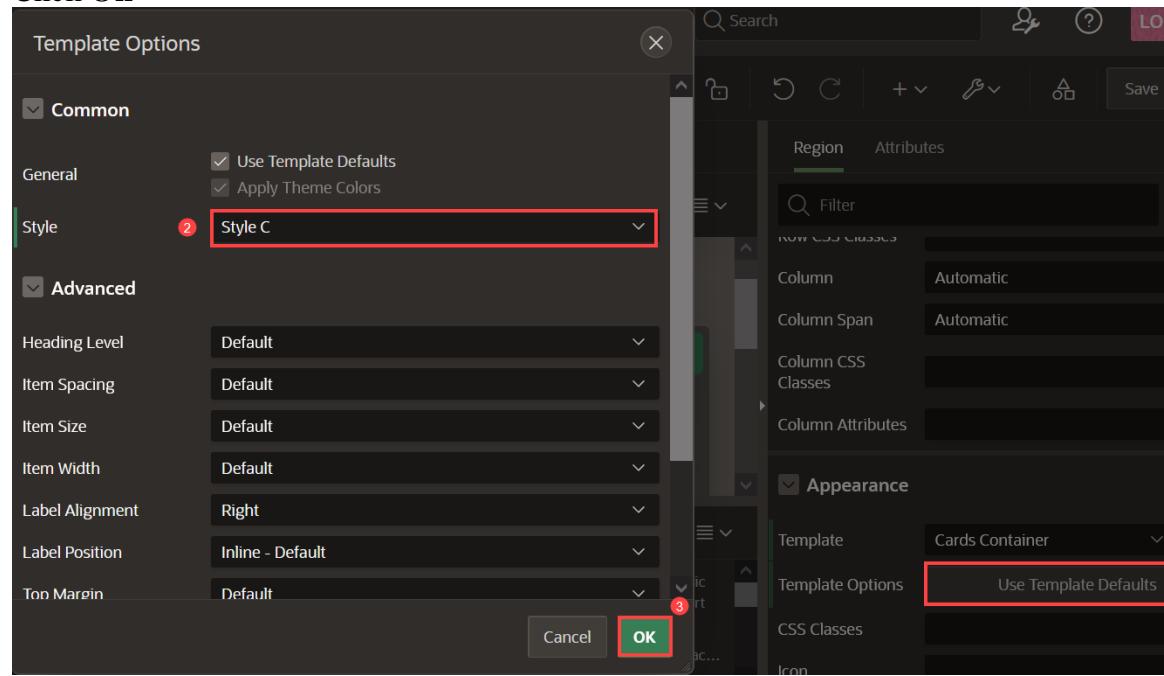
```
▪      image_charset,  
▪      image_last_updated,  
▪      color_id,  
▪      department_id,  
▪      clothing_id,  
▪      d.description,  
▪      b.brand  
▪ FROM    products p,  
▪      json_table (p.product_details, '$' columns (  
description varchar2(4000) path '$.description')  
) d,  
▪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

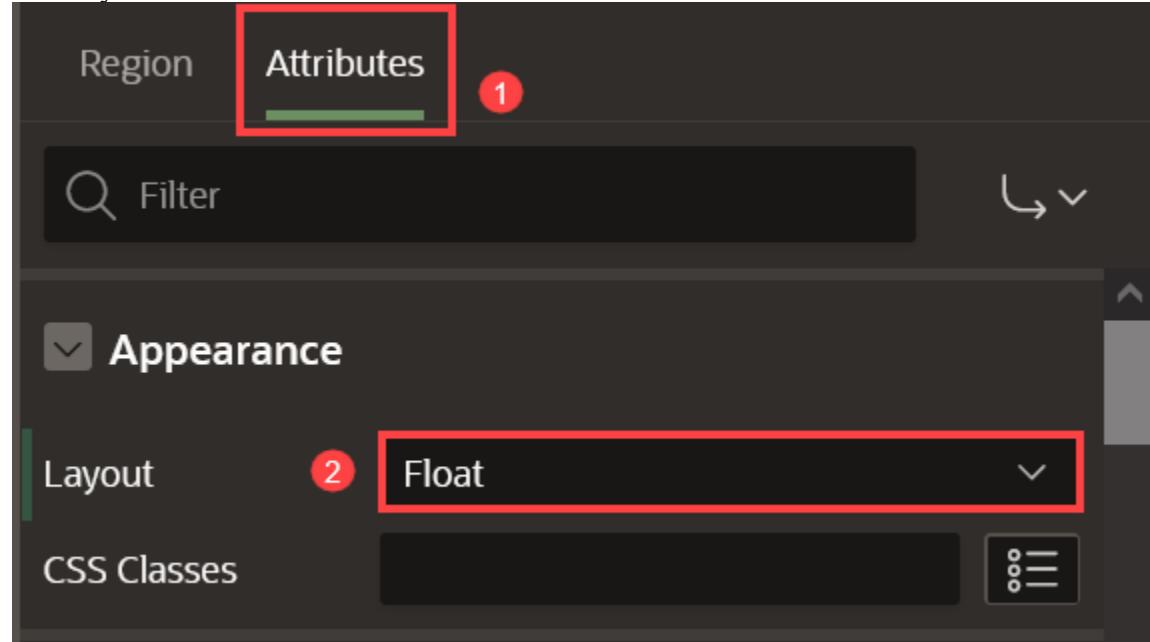
- Click **Ok**



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

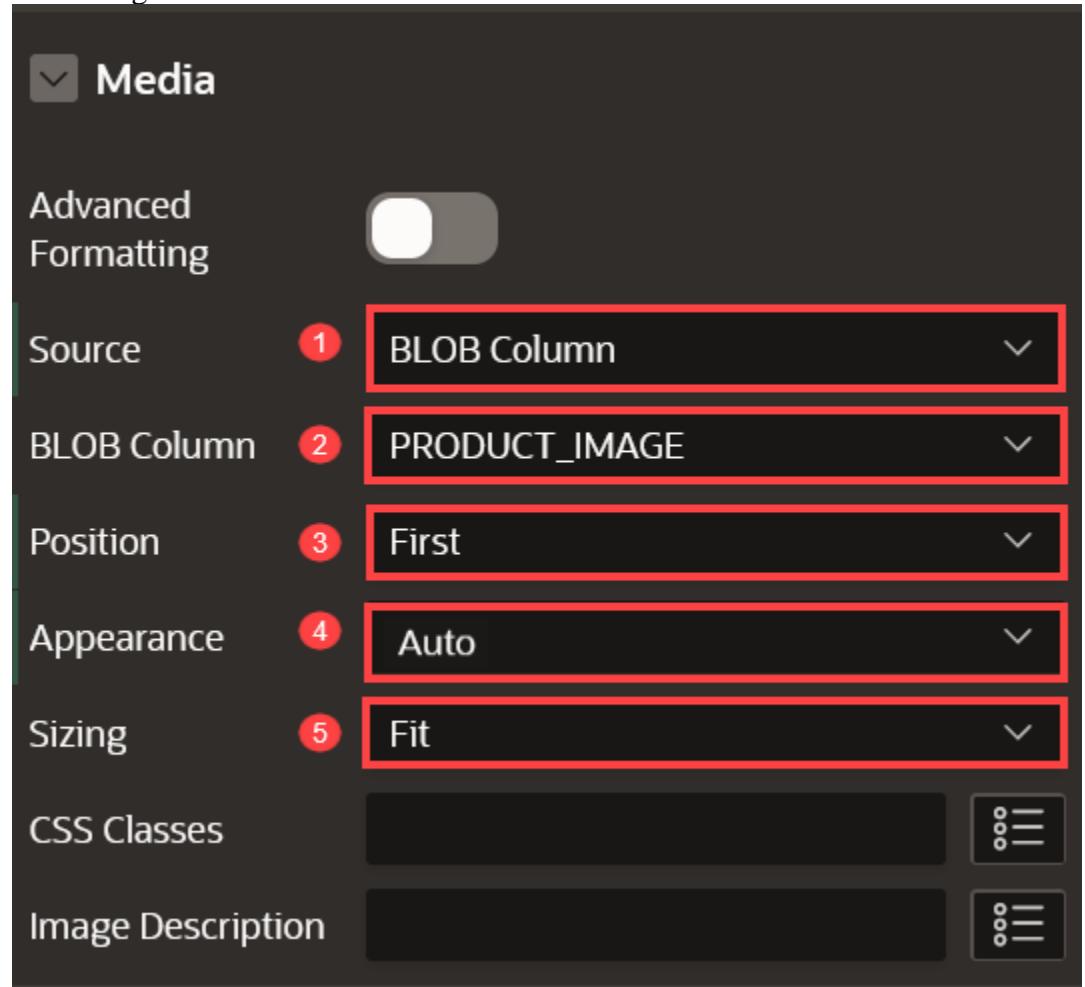
Advanced Formatting 4

HTML Expression

Price: &UNIT_PRICE. 5

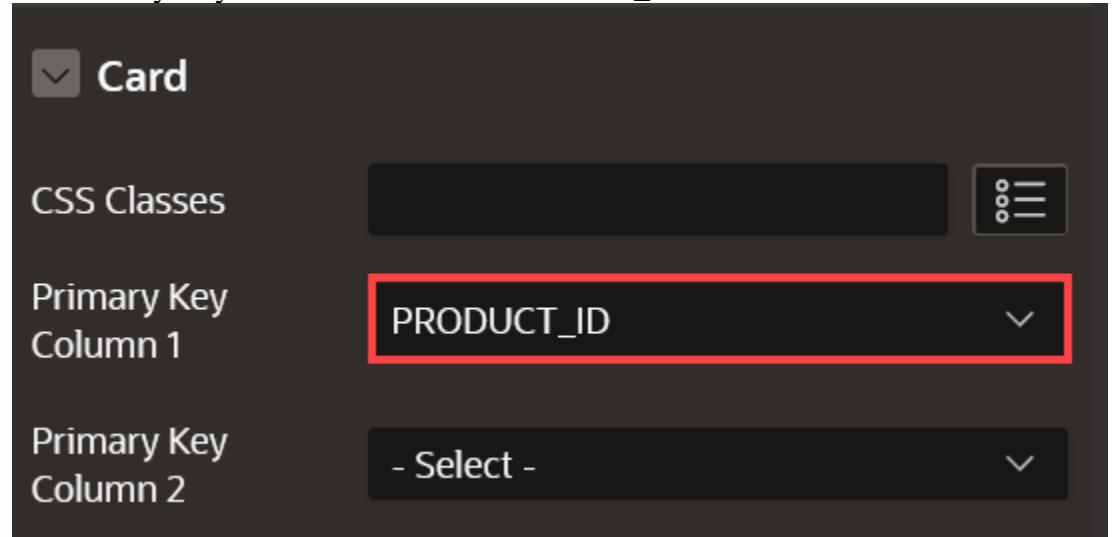
- Under Media section:

- 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:

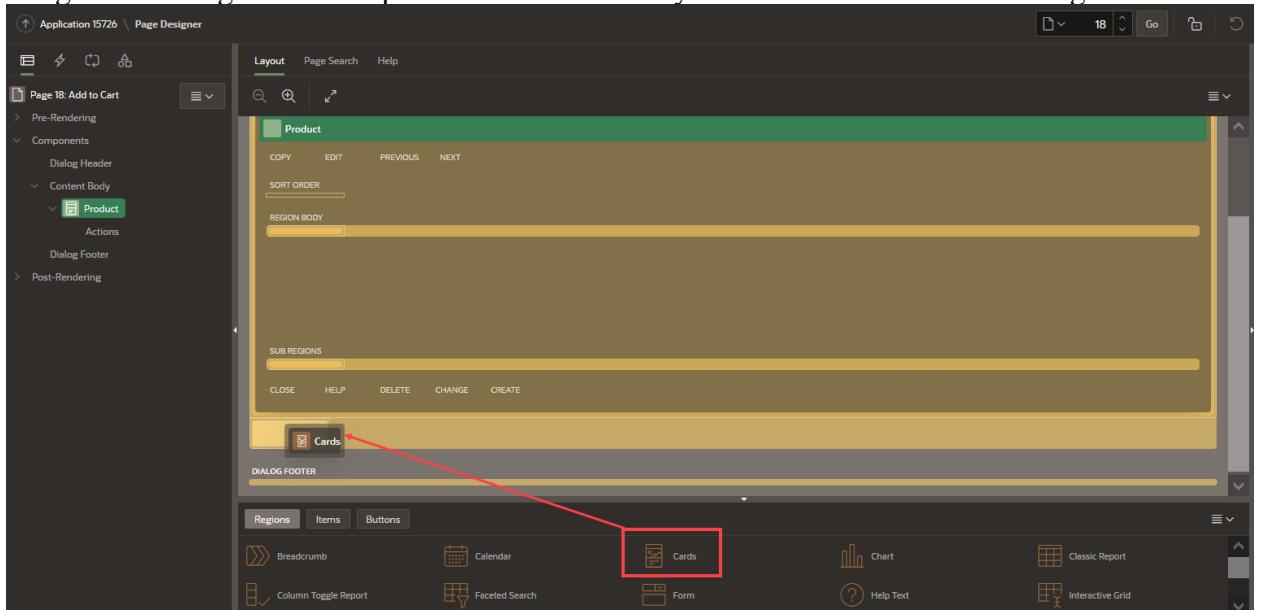
- For Primary Key Column 1 - select **PRODUCT_ID**



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 **Customer Reviews** 1

Type Cards

Source

Location Local Database

Type **SQL Query** 2

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

The screenshot shows the configuration interface for a region in Oracle APEX. The 'Identification' section has 'Customer Reviews' set as the title (marked 1). The 'Source' section has 'SQL Query' selected (marked 2) and contains the following SQL code (marked 3):

```
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
```

The 'Appearance' section has 'Standard' selected as the template (marked 4).

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

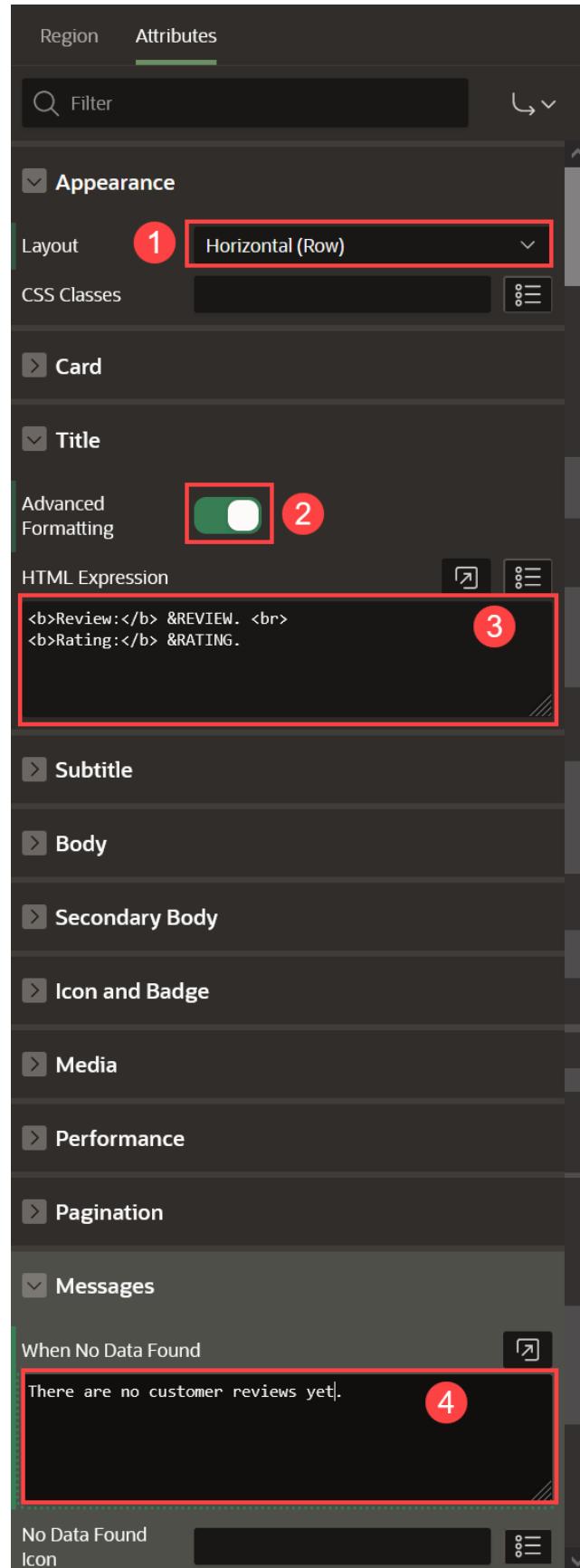
- Under Appearance section:
 - For Layout - select **Horizontal (Row)**
- Under Title section:
 - Set Advanced Formatting to **On**
 - For HTML Expression - enter the following:

▪ Copy `Review: &REVIEW.
`

`Rating: &RATING.`

- Under Messages:

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



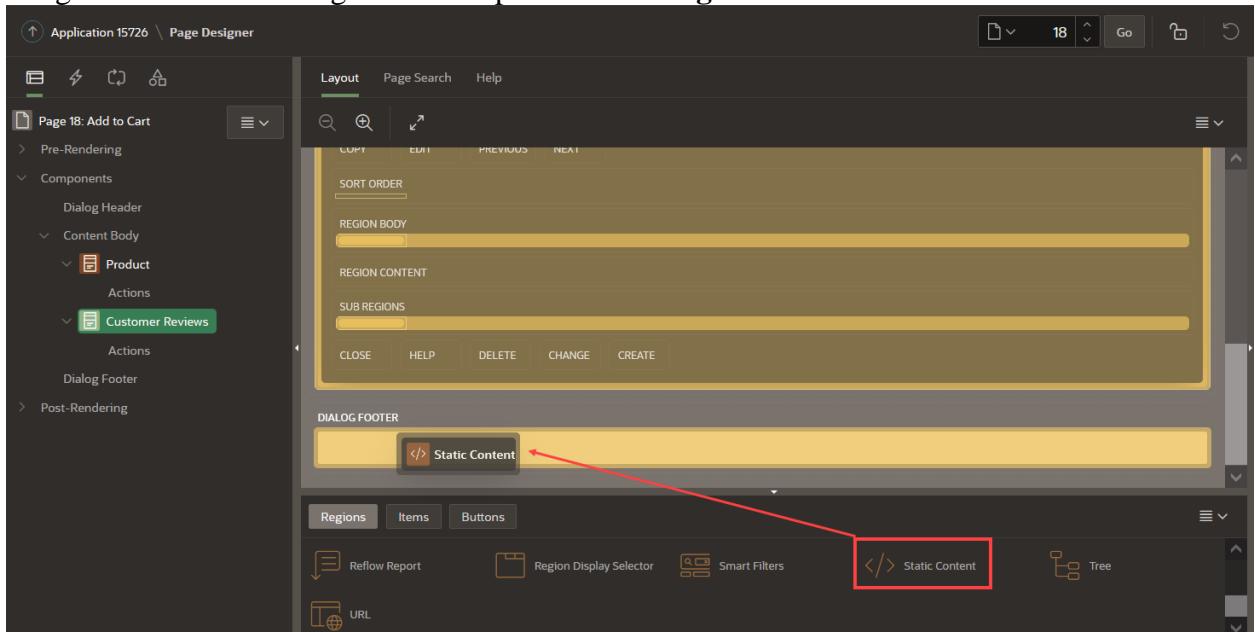
Task 4: Add Items and Buttons

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**.

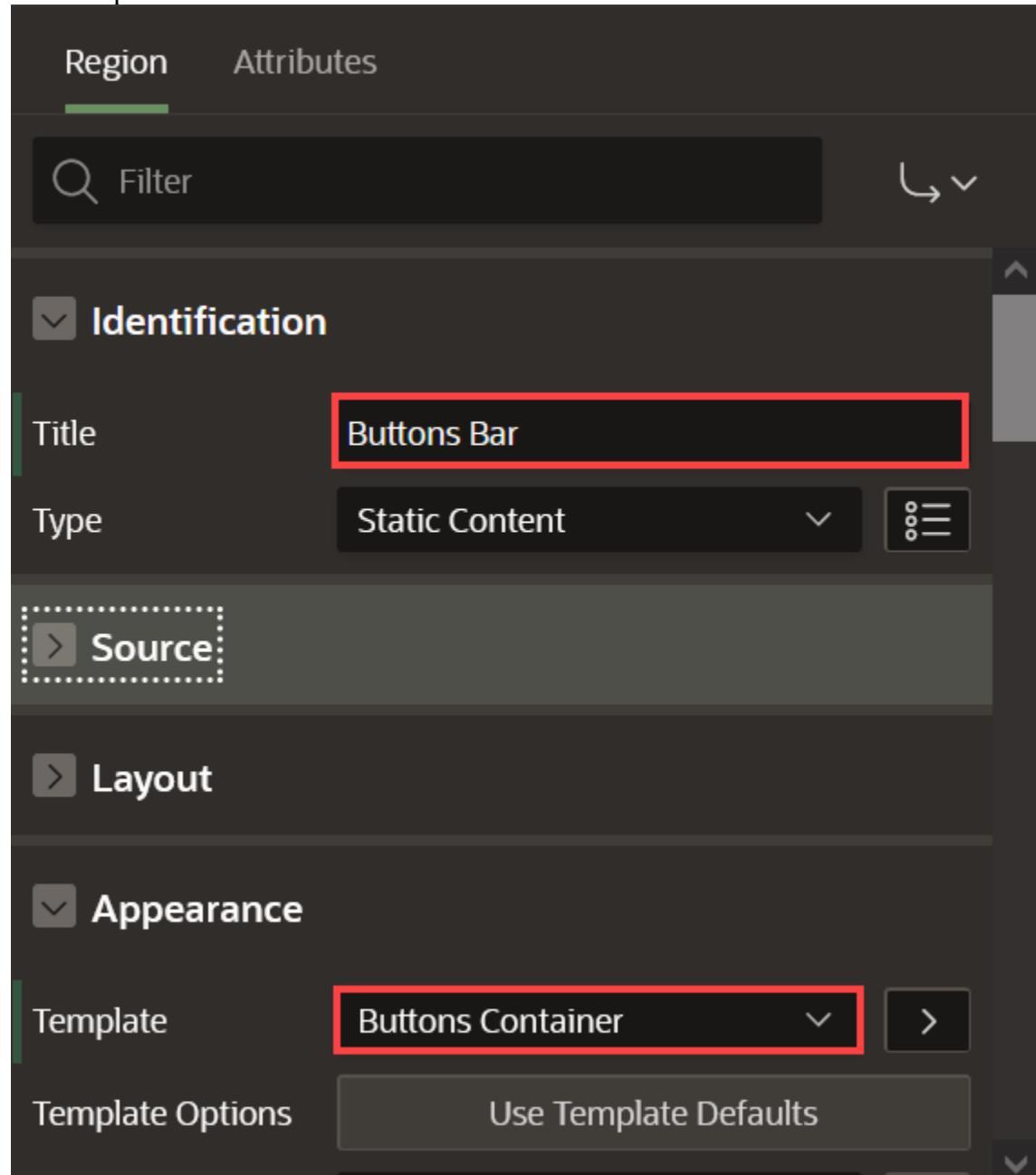
2. Drag a **Static Content** region and drop it to the **Dialog Footer**.



3. In the Property Editor, enter the following:

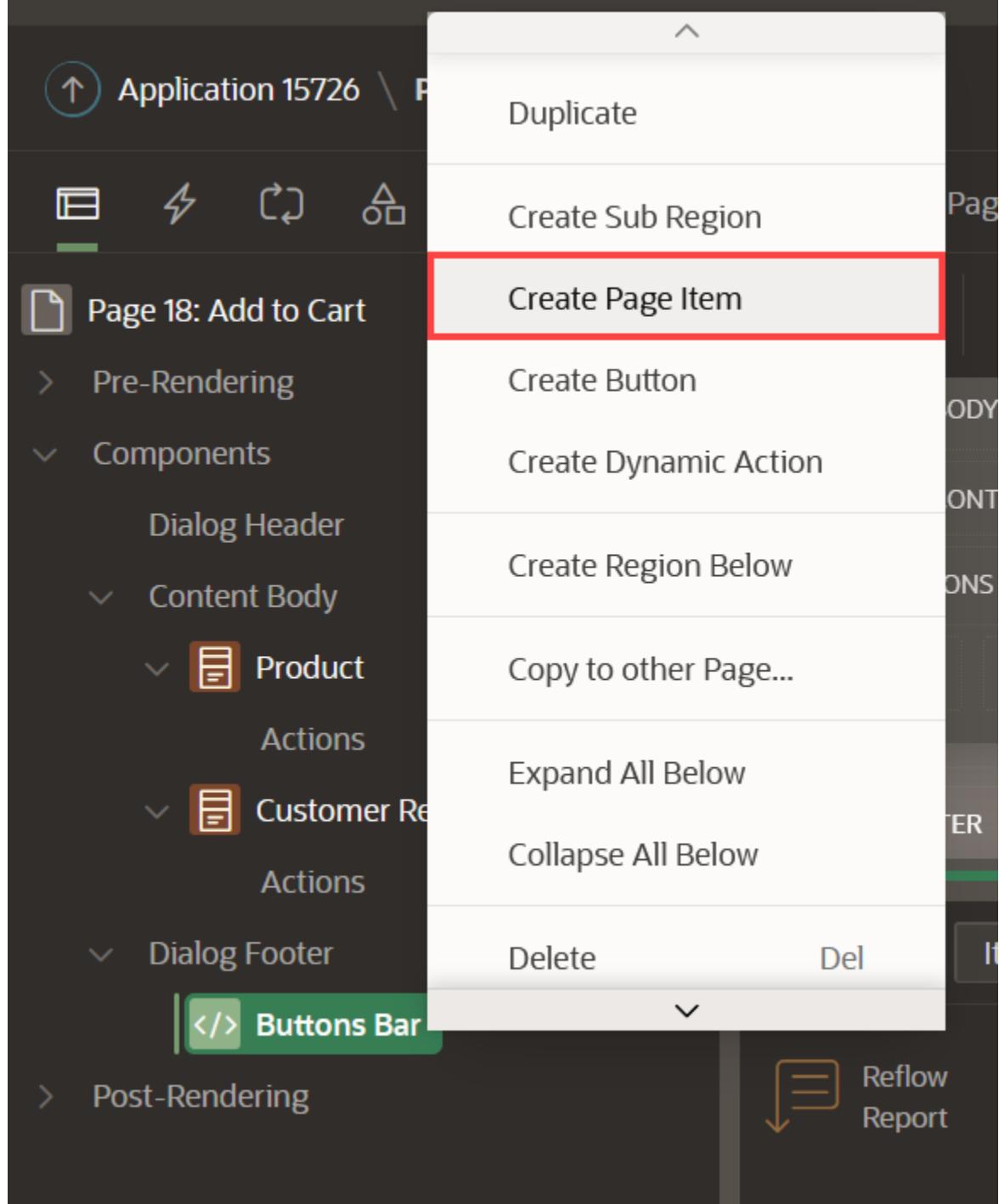
- For Title - enter **Buttons Bar**

- 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:

- o 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

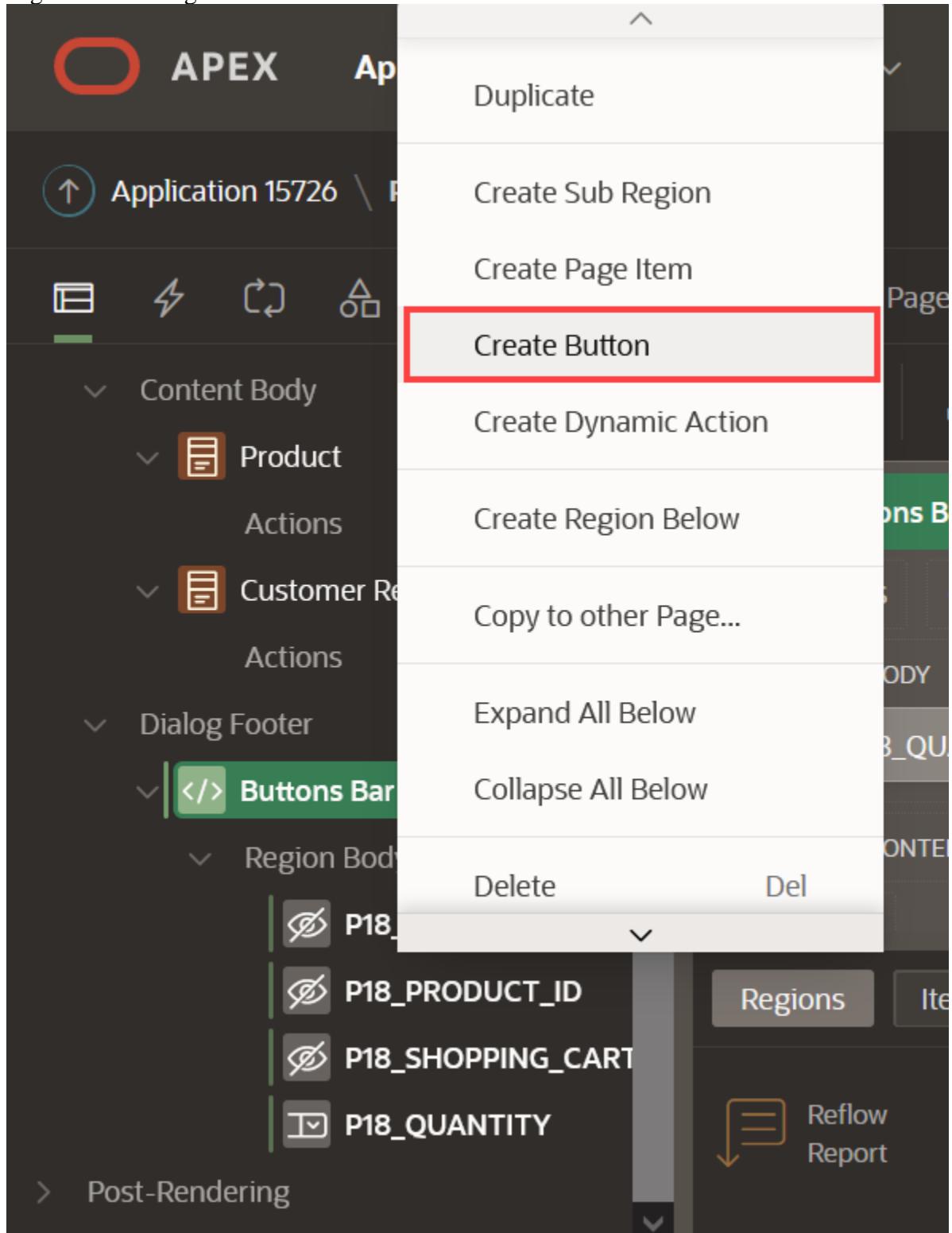
| Display Value | Return Value |
|---------------|--------------|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

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:

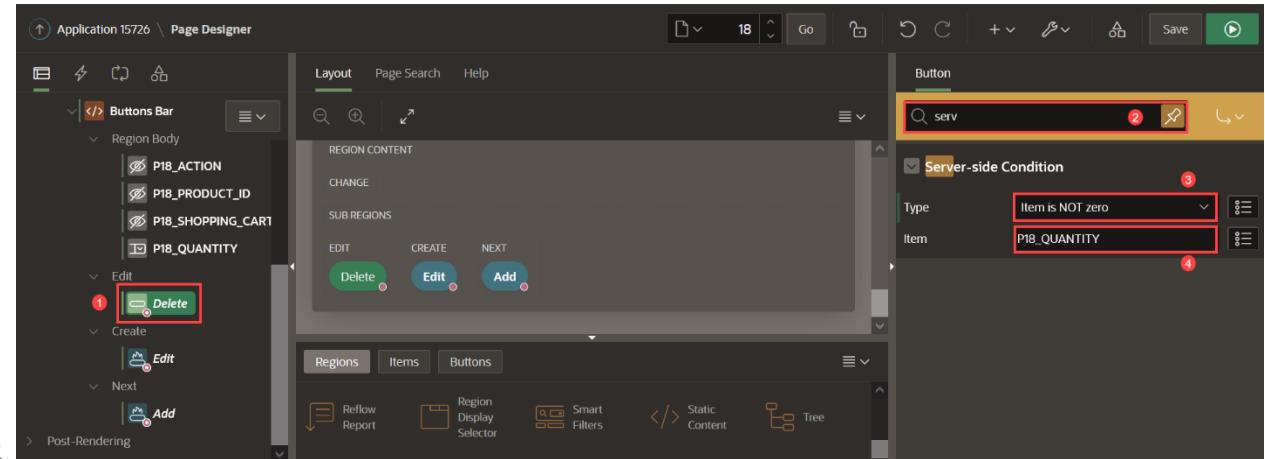
Table 3: Details of the Buttons

| 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

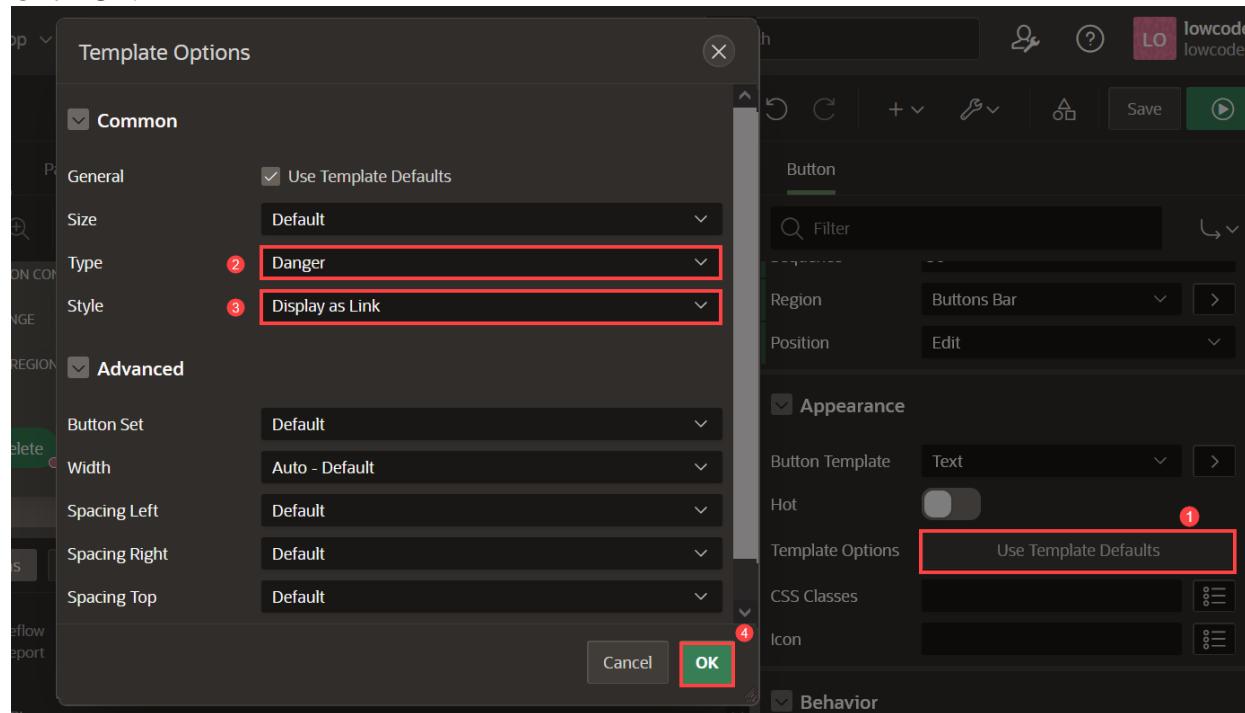
| Name | Type | Item |
|--------|------------------|--------------|
| Add | Item is zero | P18_QUANTITY |
| Edit | Item is NOT zero | P18_QUANTITY |
| Delete | Item is NOT zero | P18_QUANTITY |



12.

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

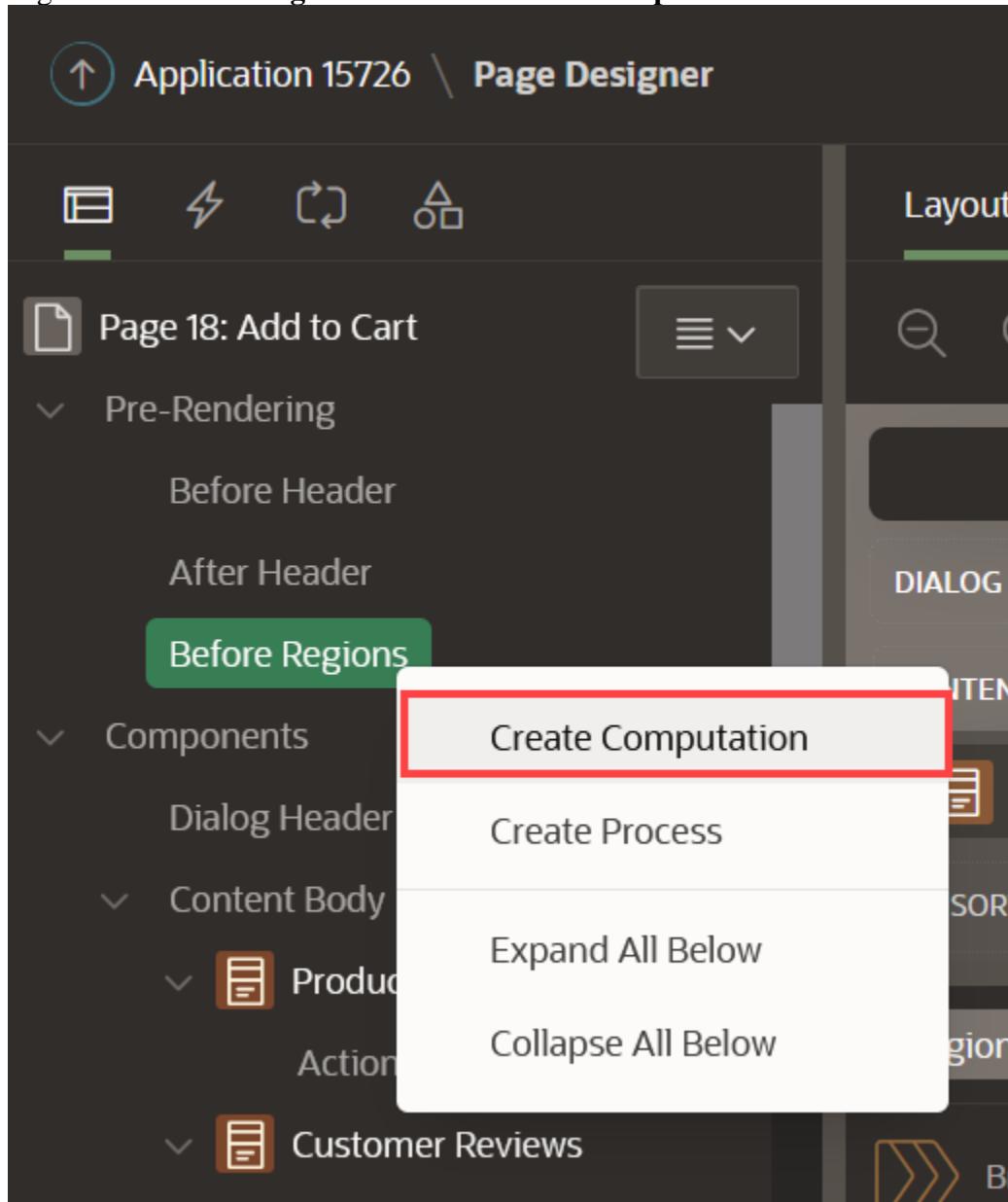
- Under Appearance section, click **Use Template Defaults**:
 - For Type - select **Danger**
 - For Style -select **Display as Link**
 - For Spacing Right, select **Large**
- Click **Ok**.



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:

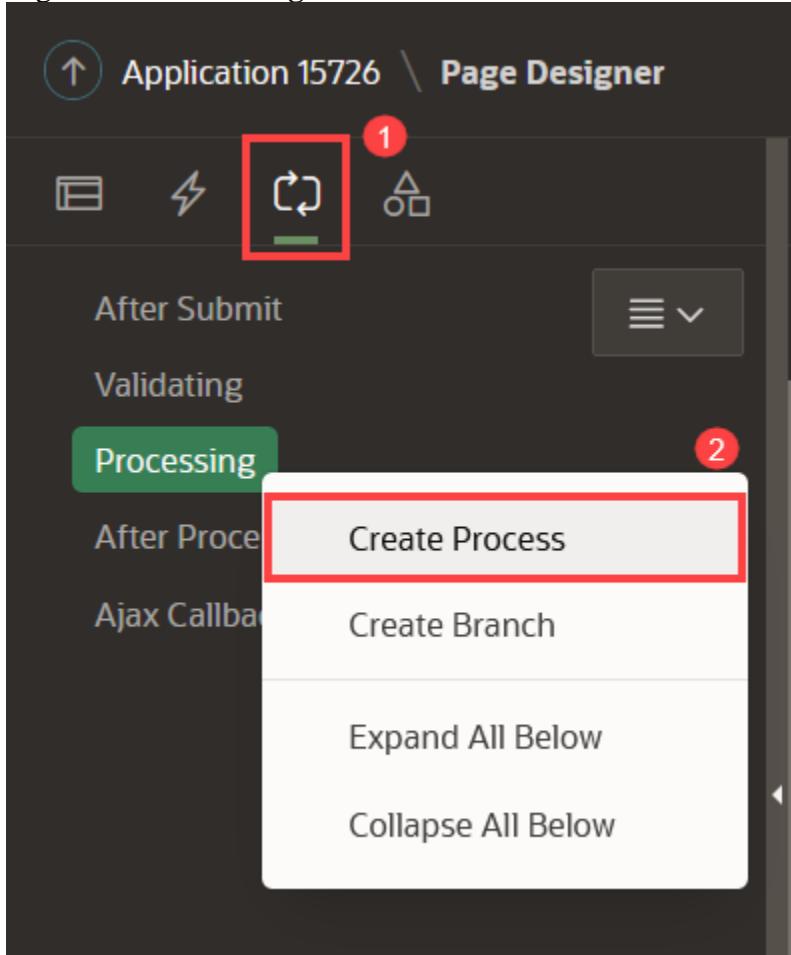
```
CopyRETURN manage_orders.product_exists(p_product =>
:P18_PRODUCT_ID);
```

The screenshot shows the 'Computation' configuration page. It includes sections for 'Identification', 'Execution Options', and 'Computation'. The 'Identification' section has an 'Item Name' set to 'P18_QUANTITY'. The 'Execution Options' section has a 'Sequence' of 10 and a 'Point' of 'Before Regions'. The 'Computation' section has a 'Type' of 'Function Body' and a 'Language' of 'PL/SQL'. The 'PL/SQL Function Body' field contains the code: 'RETURN manage_orders.product_exists(p_product => :P18_PRODUCT_ID);'. The entire 'Computation' section is highlighted with a red box, and specific fields are numbered: 1 for the 'Item Name', 2 for the 'Type', and 3 for the 'PL/SQL Function Body' code.

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:

```

○ CopyBEGIN

○      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 -

The screenshot shows the configuration of an Oracle APEX process. The process is named 'Add product' (1). The code type is 'Execute Code' (1). The PL/SQL code (2) checks if a product exists and adds it if not. It also sets the action to 'ADD'. The condition for execution is when the 'Add' button is pressed (3).

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

```
o CopyBEGIN

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

o          manage_orders.remove_product(p_product =>
:P18_PRODUCT_ID);

o          manage_orders.add_product (p_product =>
:P18_PRODUCT_ID,
                                p_quantity =>
:P18_QUANTITY);

o      END IF;

o      :P18_ACTION := 'EDIT';
```

```
END;
```

- o 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:
 - o For Name - enter **Delete product**
 - o For Type - select **Execute Code**
 - o For PL/SQL Code - enter the following code:
 - o Copy**BEGIN**
 - o **IF** *manage_orders.product_exists*(*p_product* => :P18_PRODUCT_ID) > 0 **THEN**
 - o *manage_orders.remove_product*(*p_product* => :P18_PRODUCT_ID);
 - o **END IF;**
 - o :P18_ACTION := 'DELETE';

```
END;
```

- o 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 | <input type="button" value="↗"/> |

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

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

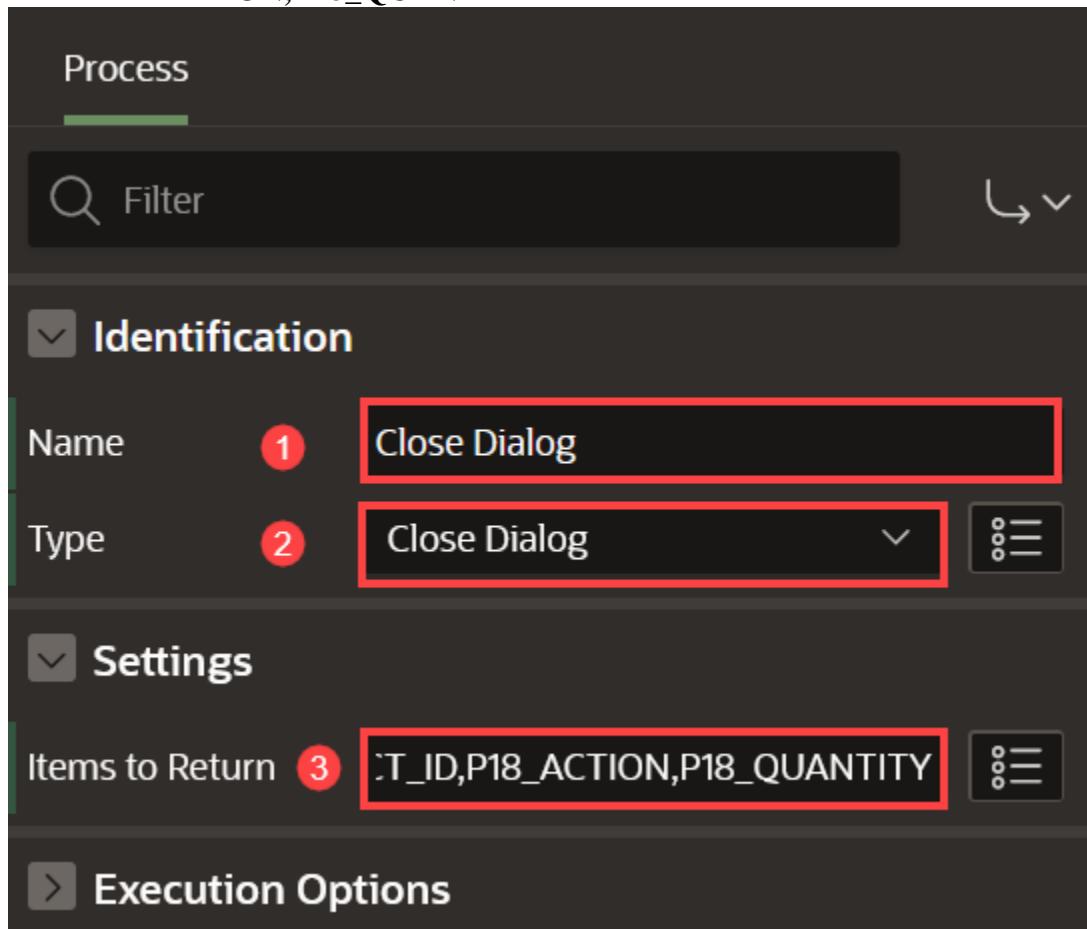
```
END ;
```

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.

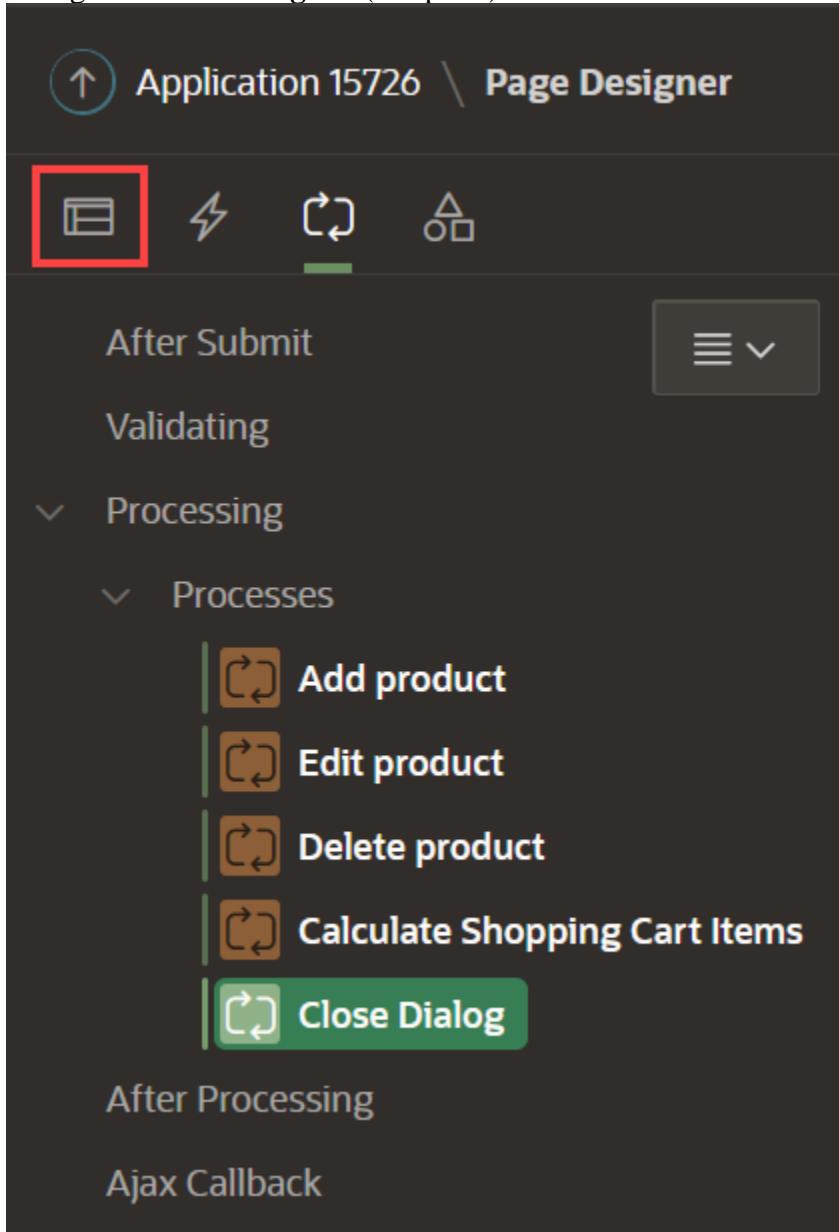
1. In the **Processing** tab.
2. Right click **Processing** and click **Create Process**.
3. In the Property Editor, enter the following:
 - o Under Identification section:
 - For Name - enter **Close Dialog**
 - For Type - select **Close Dialog**
 - o Under Settings section:

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



Task 11: Enhance the Modal Page

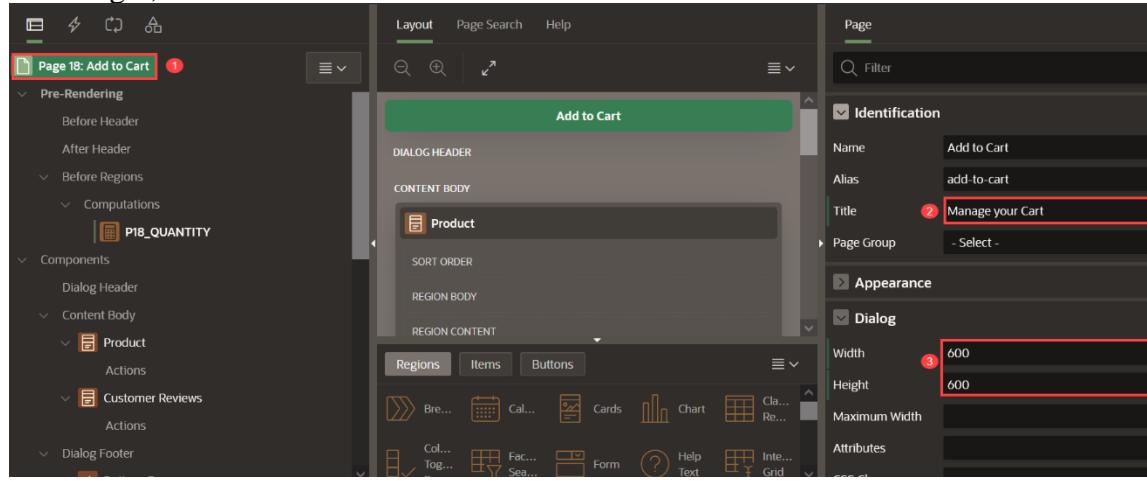
1. Navigate to **Rendering** tab (left pane).



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:

The screenshot shows the ACME Shop storefront. At the top, there is a blue header bar with the text "ACME Shop" on the left, a "Shopping Cart" icon with the number "3" in the middle, and an "Administration" link on the right. Below the header is the main content area. On the left side, there is a sidebar with a search bar ("Search...") and a "Go" button. The sidebar also contains several facets: "Department" (Boy's 17, Women's 14, Girl's 10, Men's 5), "Clothing" (Jeans 6, Pyjamas 6, Coat 5, Trousers 5, Shorts 4), "Color" (black 9, red 9, blue 8, grey 7, white 5), and "Unit Price". The main content area displays four products in a grid:

- Girl's Jeans (Grey)** by KINETICUT, \$9.7. Status: **1 in cart**.
- Girl's Hoodie (White)** by PROXSOFT, \$39.91.
- Boy's Coat (Blue)** by GRONK, \$10.24. Status: **1 in cart**.
- Men's Hoodie (Red)** by FANGOLD, \$24.71.

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

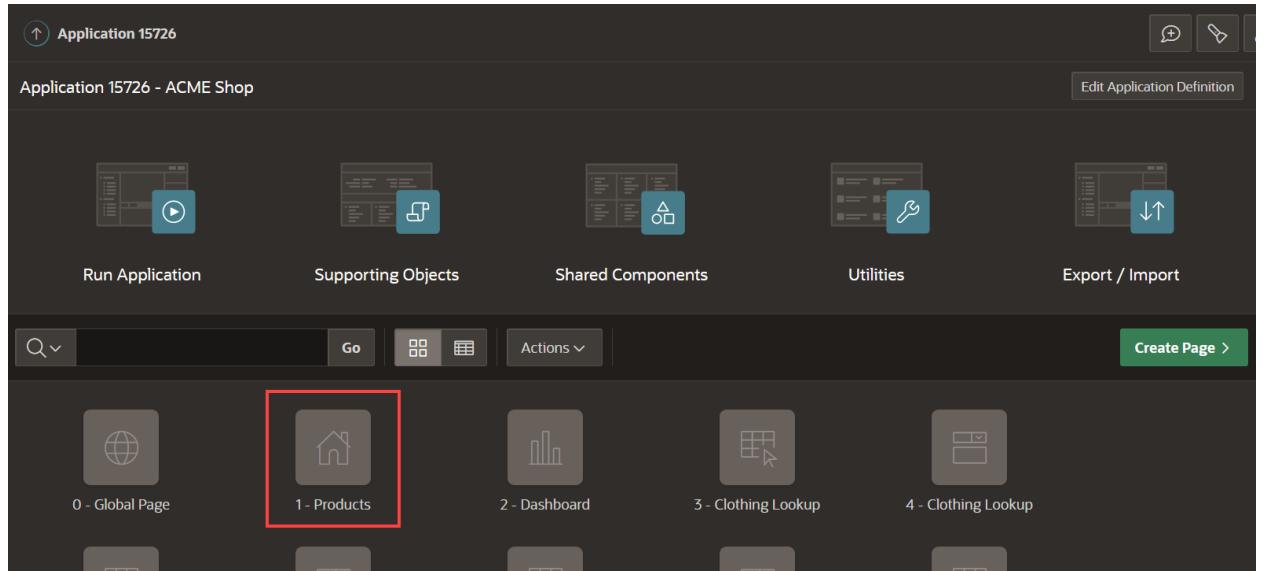
The **Products** page is where your customers can explore the products and select what they wish to buy. As you can see, it's hard to find the products and it would be beneficial to see additional details related to the products.

- 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.}*

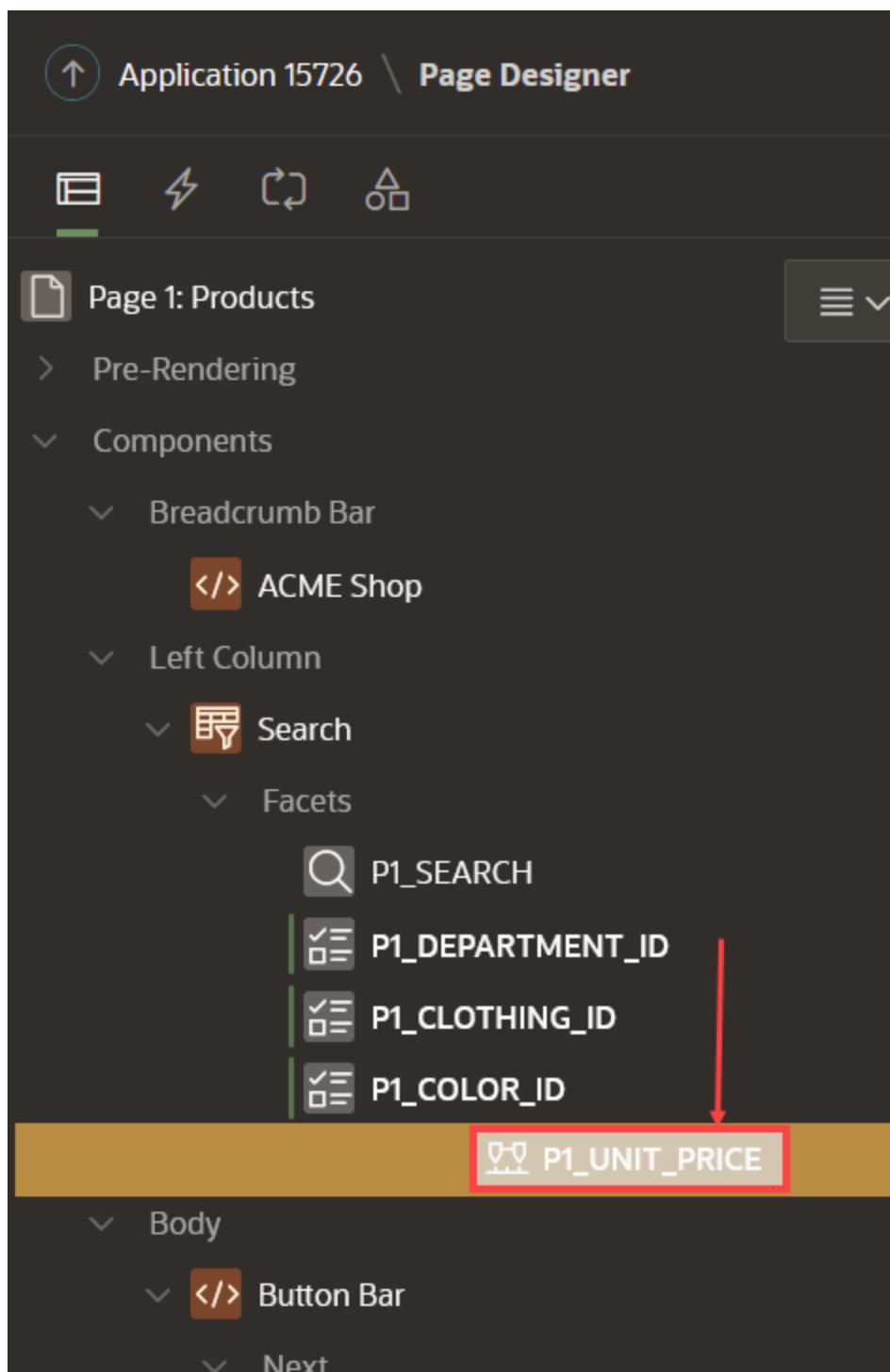
In the Developer Toolbar click **Edit Page 1**.

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.

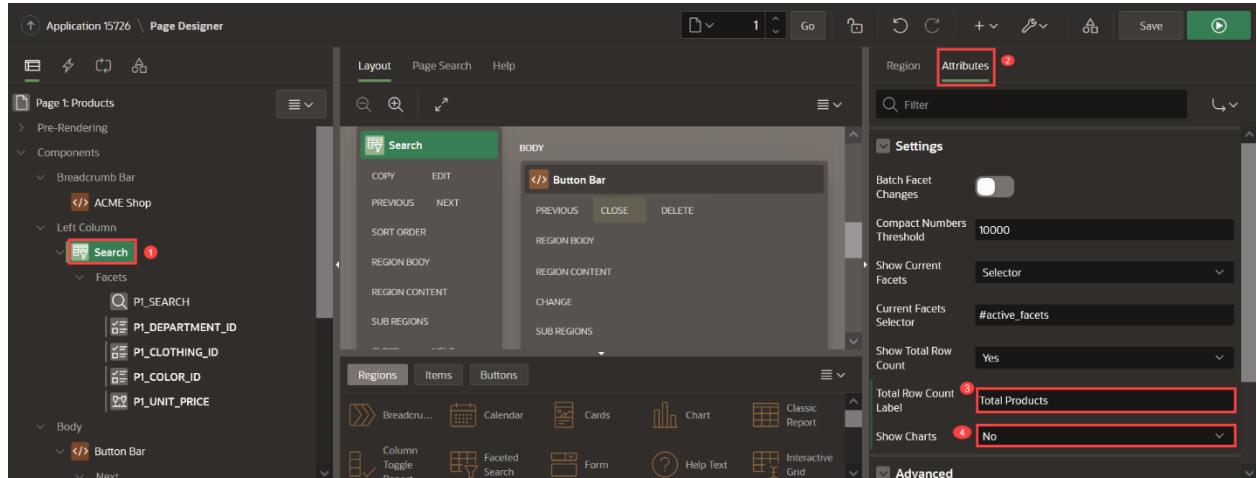
2. Unit price is not a common search criteria, so you want to put this facet at the bottom. In the Rendering tree (left pane), under Search, within Facets, click and hold **P1_UNIT_PRICE** and drag it down until it is under **P1_COLOR_ID**, then release the mouse. Reorder the facets to display as in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

```

○ CopySELECT "PRODUCT_ID",
○      "PRODUCT_NAME",
○      "UNIT_PRICE",
○      "PRODUCT_DETAILS",
○      "PRODUCT_IMAGE",
○      "IMAGE_MIME_TYPE",
○      "IMAGE_FILENAME",
○      "IMAGE_CHARSET",
○      "IMAGE_LAST_UPDATED",
○      "COLOR_ID",

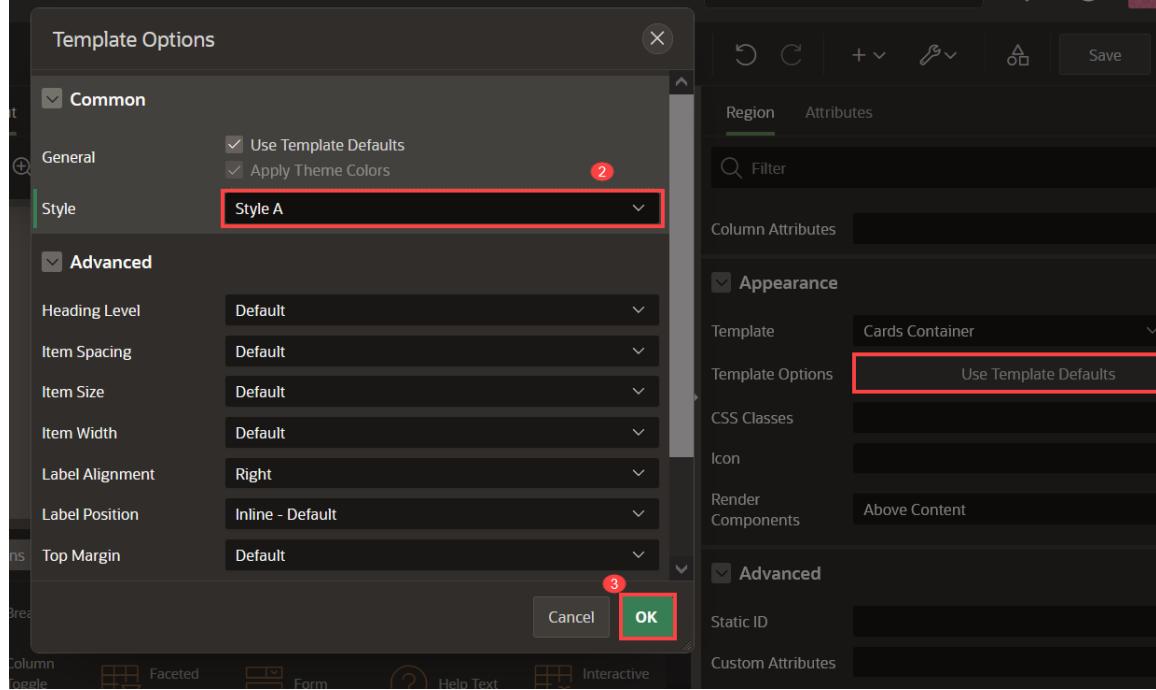
```

- o (
- o **SELECT** 11."COLOR"
- o **FROM** "COLOR_LOOKUP" 11
- o **WHERE** 11."COLOR_ID" = m."COLOR_ID")
"COLOR_ID_L\$1",
- o "DEPARTMENT_ID",
- o (
- o **SELECT** 12."DEPARTMENT"
- o **FROM** "DEPARTMENT_LOOKUP" 12
- o **WHERE** 12."DEPARTMENT_ID" =
m."DEPARTMENT_ID") "DEPARTMENT_ID_L\$2",
- o "CLOTHING_ID",
- o (
- o **SELECT** 13."CLOTHING"
- o **FROM** "CLOTHING_LOOKUP" 13
- o **WHERE** 13."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
```

- o Under Appearance section:

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

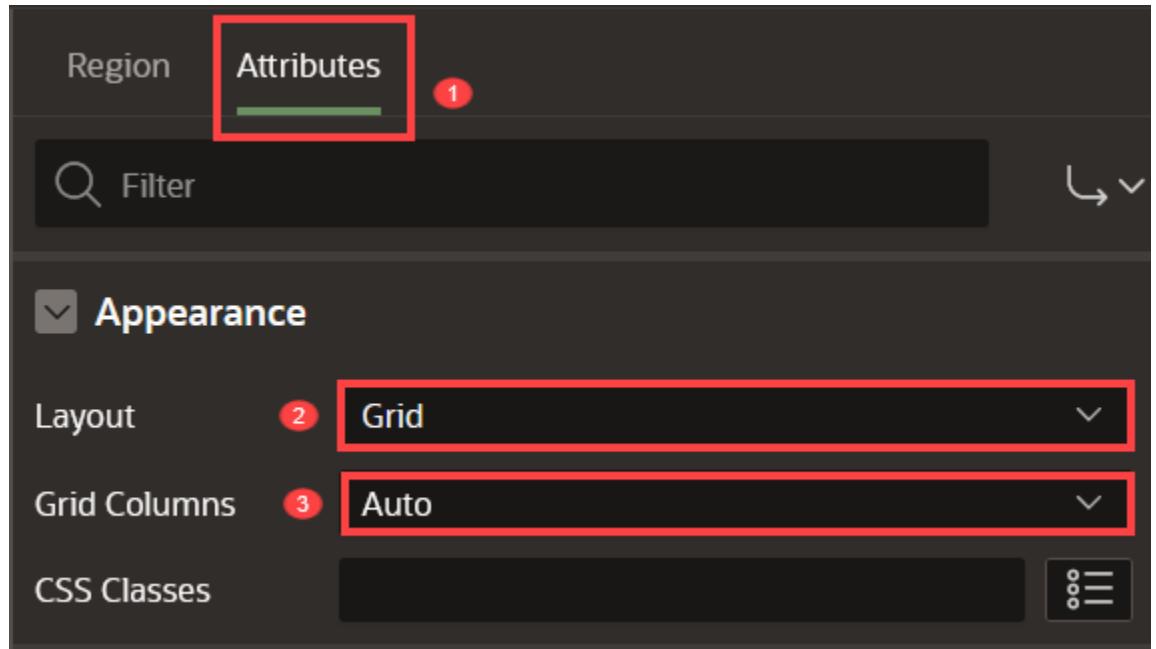


- Click **Ok**

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

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

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

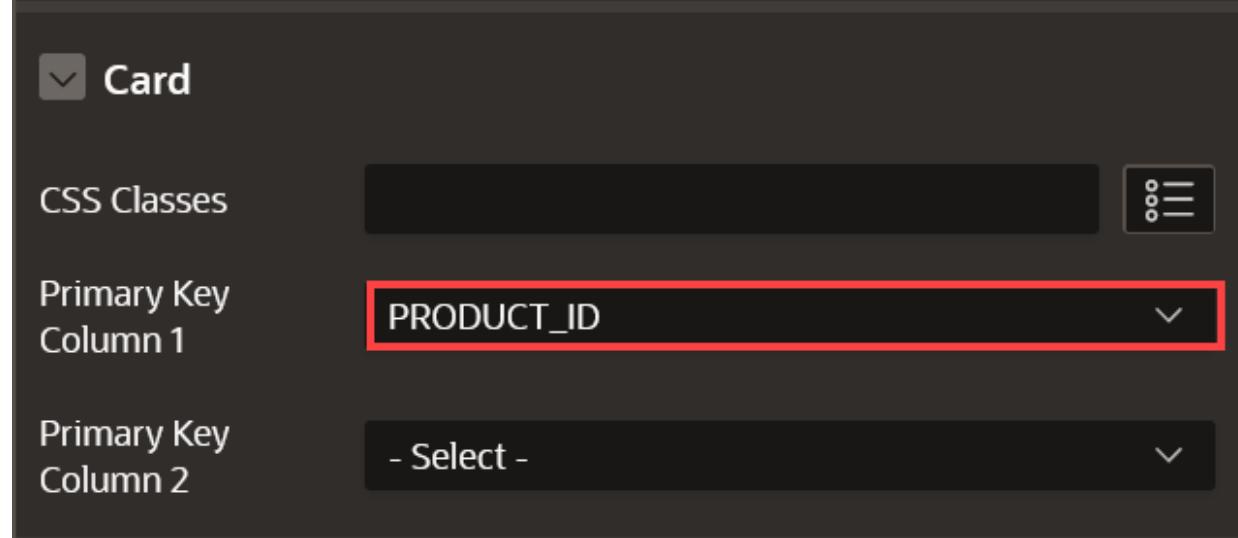
Appearance 7 Widescreen

Sizing 8 Fit

CSS Classes

Image Description

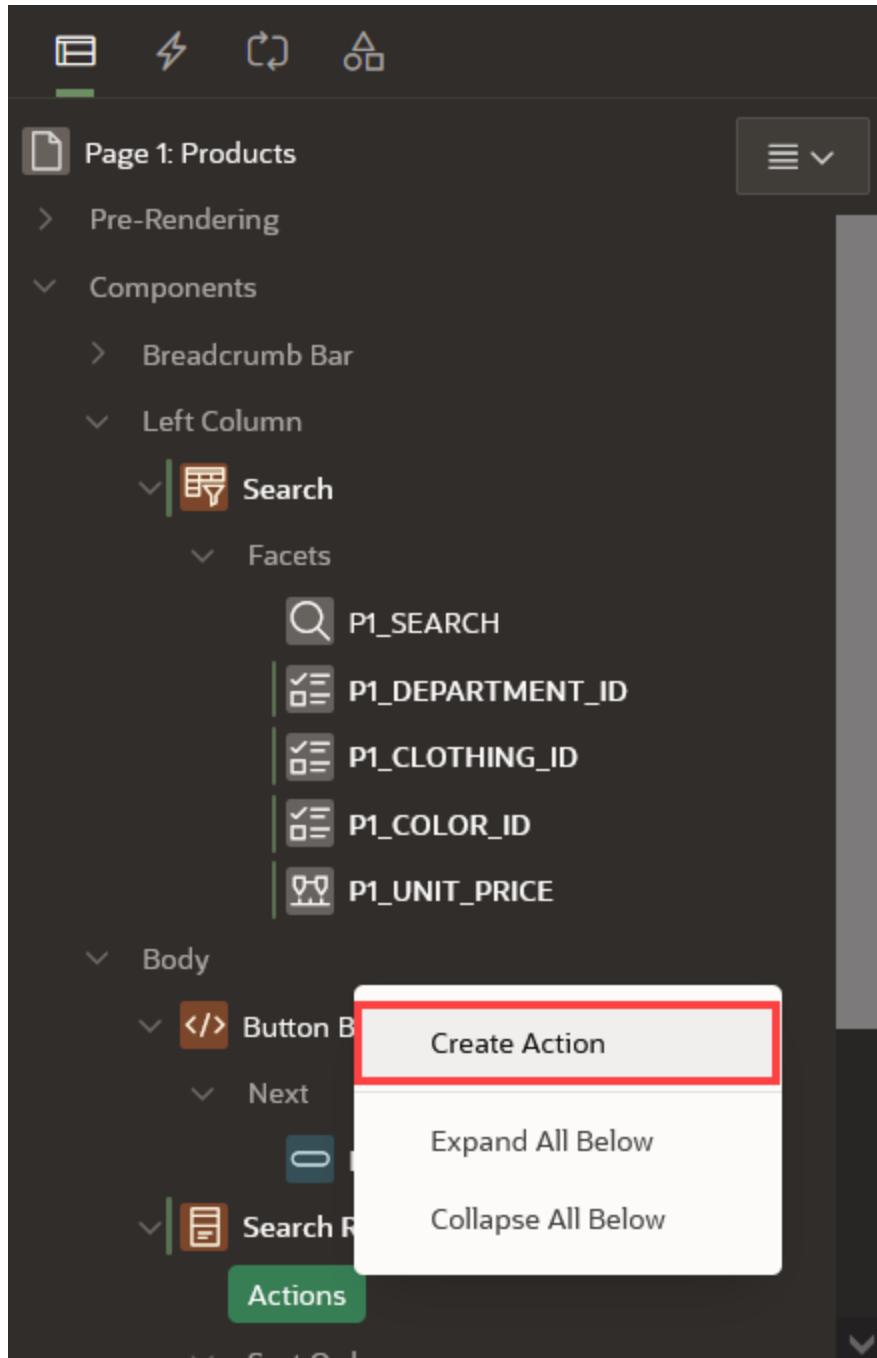
- Under Card section:
 - For Primary Key Column 1 - select **PRODUCT_ID**



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**.



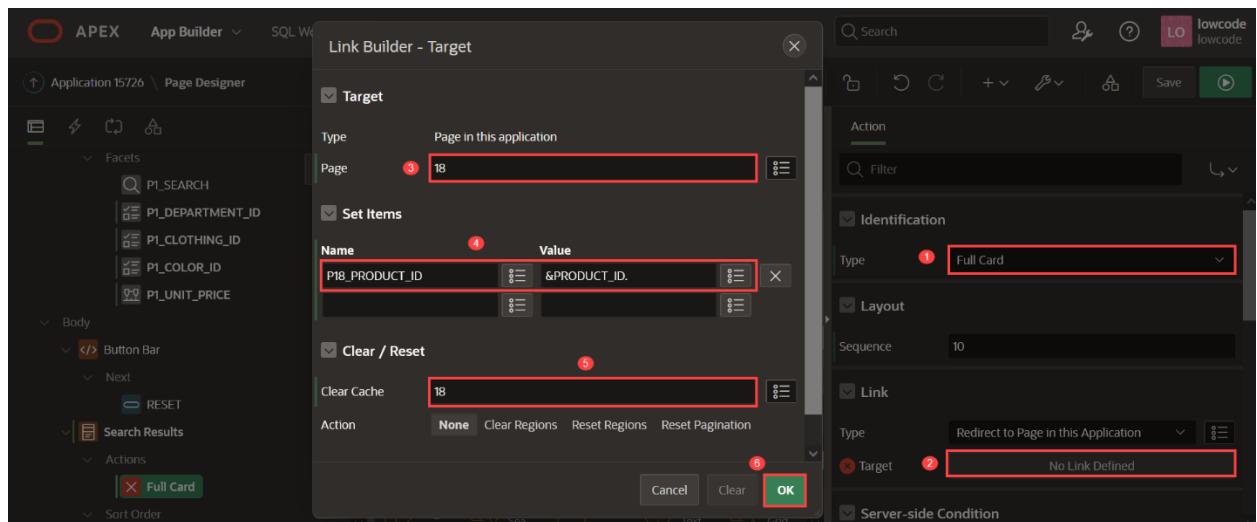
3. In the Property Editor (right pane), enter the following:

- 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**.

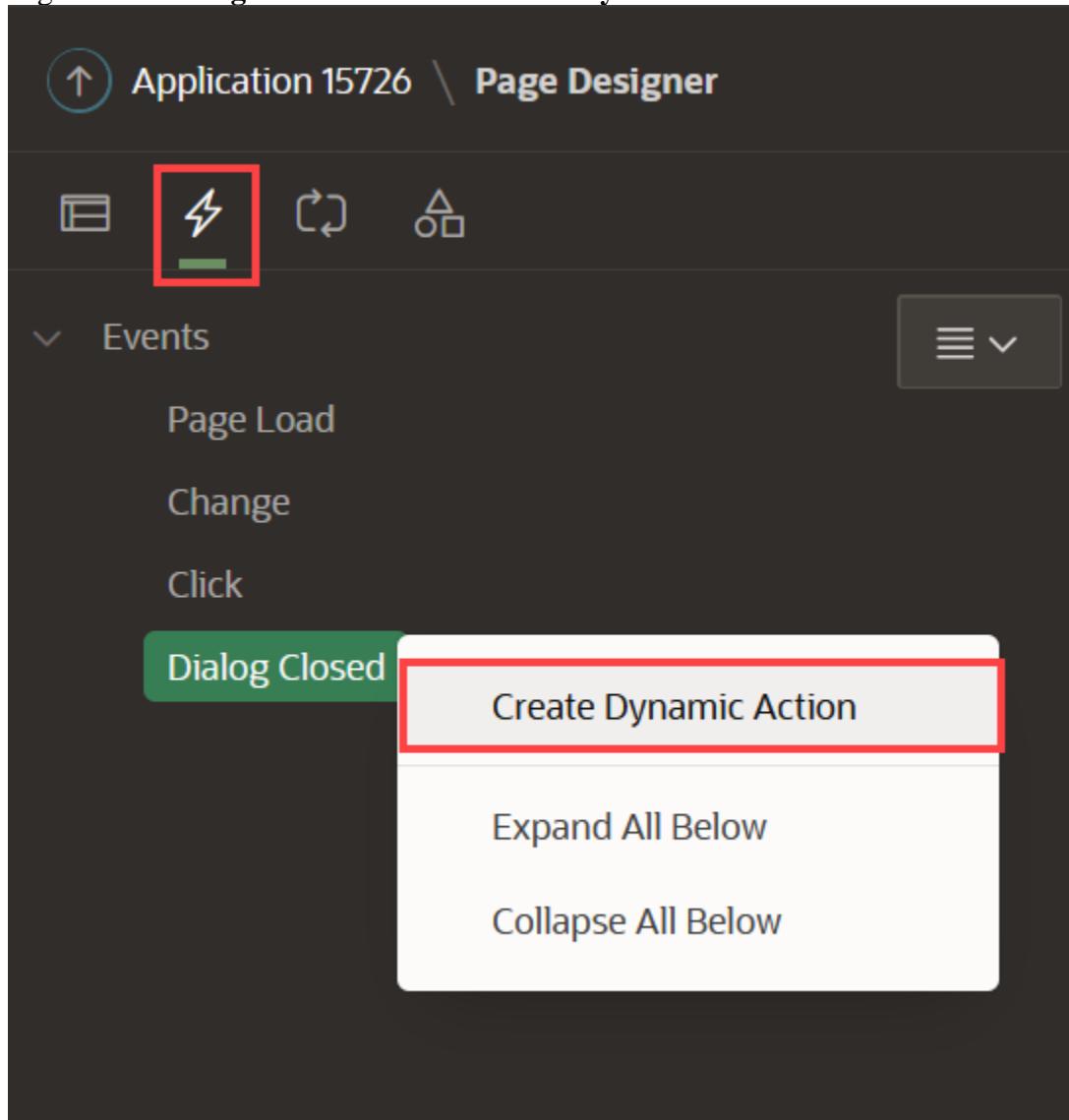


Task 6: Add Dynamic Actions

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**

Dynamic Action

Identification

| | |
|------|------------------------|
| Name | ① Show Success Message |
|------|------------------------|

Execution Options

| | |
|----------|----|
| Sequence | 10 |
|----------|----|

When

| | |
|----------------|------------------|
| Event | ② Dialog Closed |
| Selection Type | ③ Region |
| Region | ④ Search Results |

4. Navigate to **Refresh Action**.

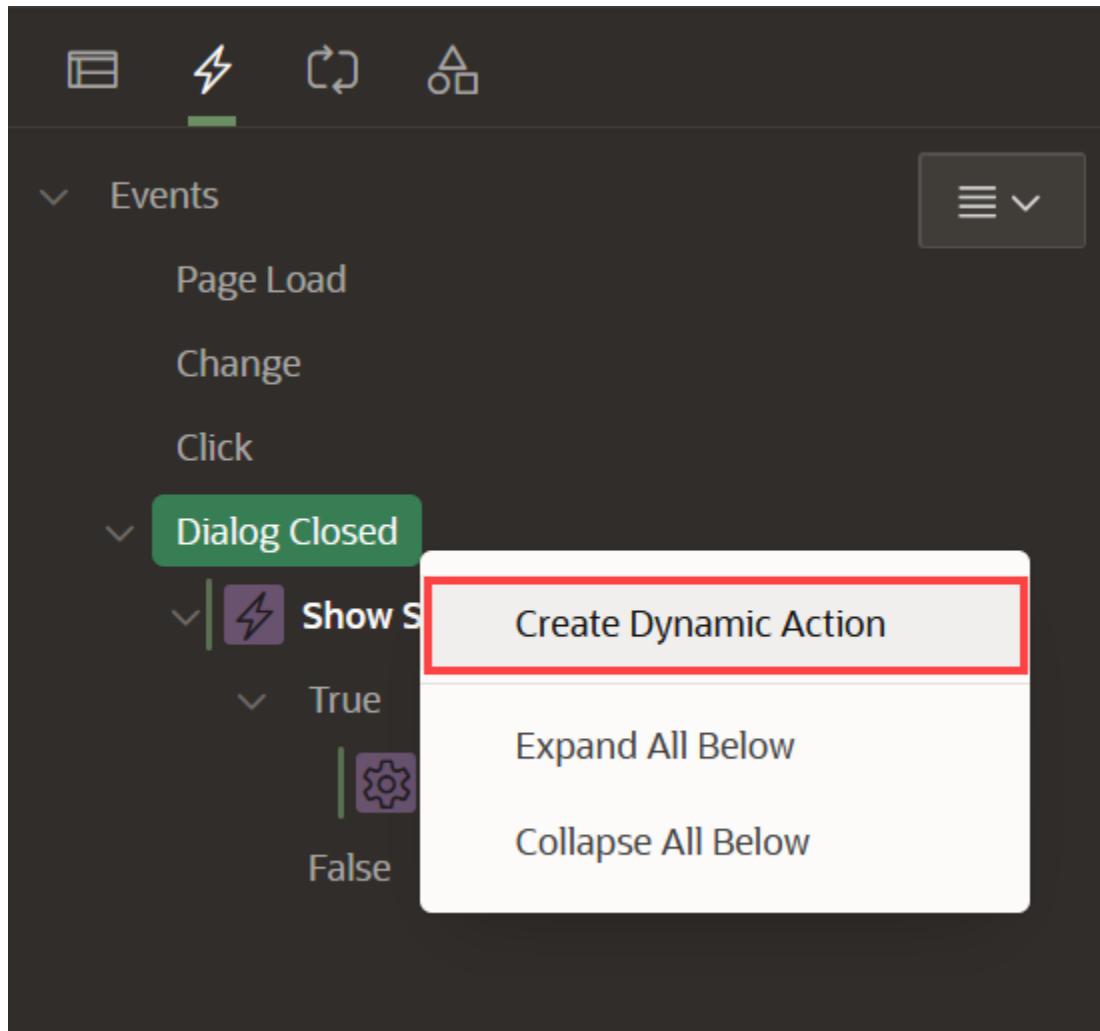
- Under Identification section:
 - For Action - select **Execute JavaScript Code**
- Under Settings section:
 - For Code - enter the following JavaScript Code:

```
▪ 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:

```
CopyParseInt(this.data.P18_SHOPPING_CART_ITEMS) >  
0
```

Dynamic Action

Filter ↻

Identification

Name ① Update Shopping Cart Header

Execution Options

Sequence 20

When

Event ② Dialog Closed

Selection Type ③ Region

Region ④ Search Results

Client-side Condition

Type ⑤ JavaScript expression

JavaScript Expression

```
parseInt(this.data.P18_SHOPPING_CART_ITEMS) > 0
```

⑥

7. Navigate to Refresh Action.

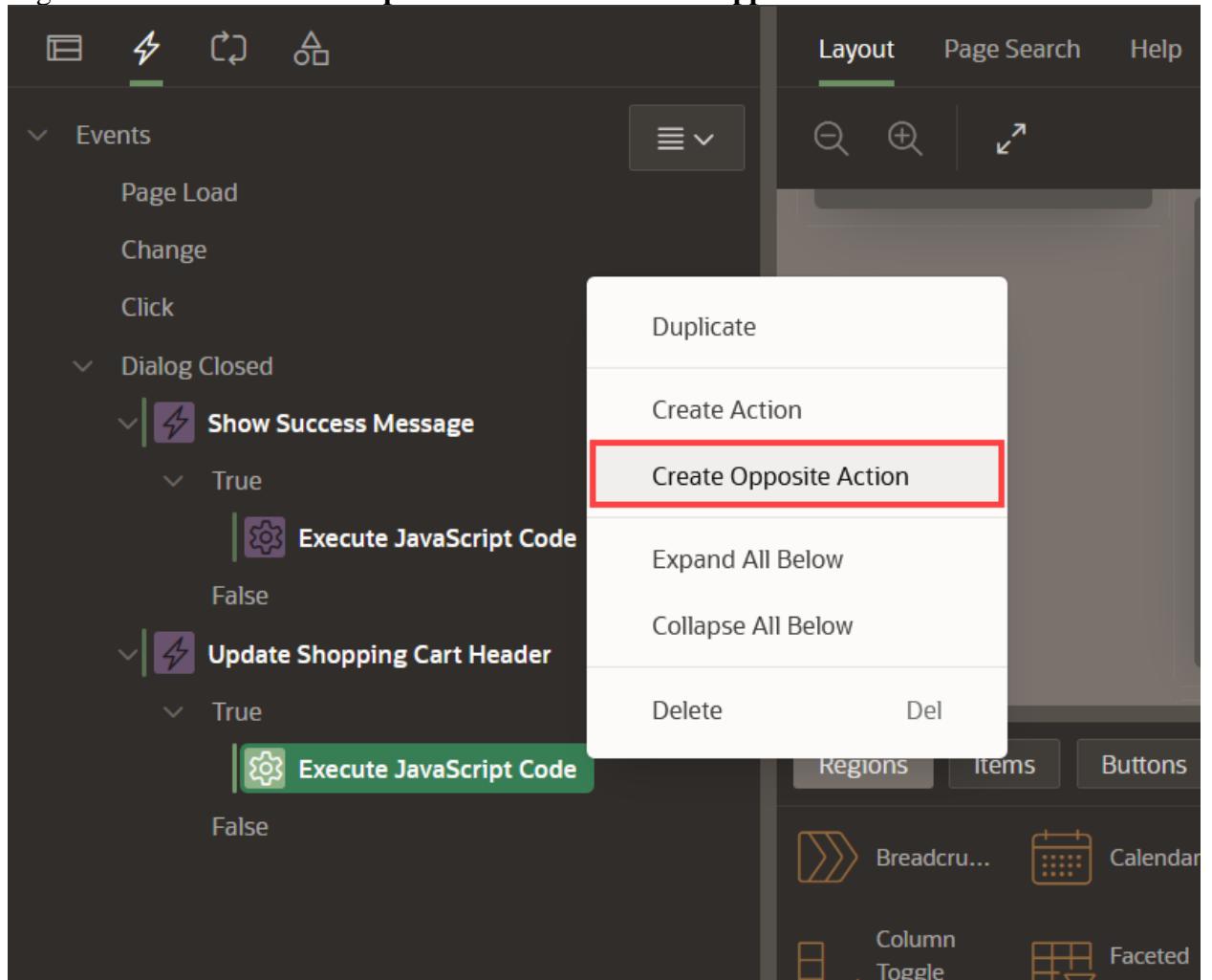
- Under Identification section:
 - For Action - select **Execute JavaScript Code**
- Under Settings section:
 - For Code - enter the following JavaScript Code:

- 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.

The screenshot shows the ACME Shop application interface. At the top, there is a blue header bar with the text "ACME Shop". Below the header, there is a search bar labeled "Search..." and a shopping cart icon. On the left side, there are two filter panels: "Department" and "Clothing". The "Department" panel shows categories like Boy's (17), Women's (14), Girl's (10), and Men's (5). The "Clothing" panel shows categories like Jeans (6), Pyjamas (6), Coat (5), Trousers (5), and Shirts (8). In the center, there is a main content area titled "Total Products 46" with a "Reset" button. The products are displayed in a grid format. The first product is "Boy's Coat (Blue)" by GRONK at \$10.24. The second product is "Boy's Coat (Brown)" by KOFFEE at \$21.16. The third product is "Boy's Hoodie (Grey)" by SUNCIPSE at \$26.14. Each product card includes an image, the name, the brand, and the price.

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:

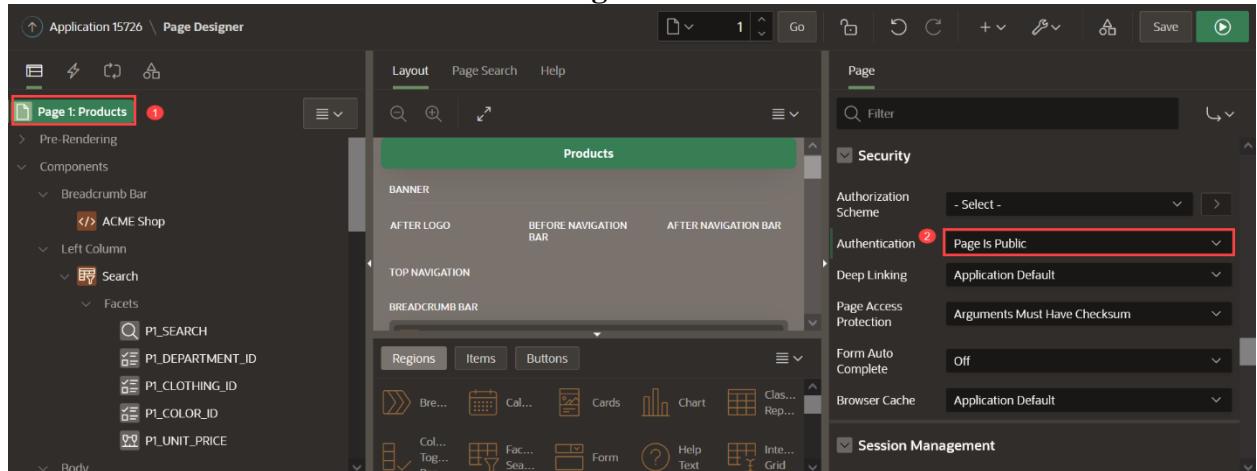
1. From the runtime application, navigate to the page.

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**.

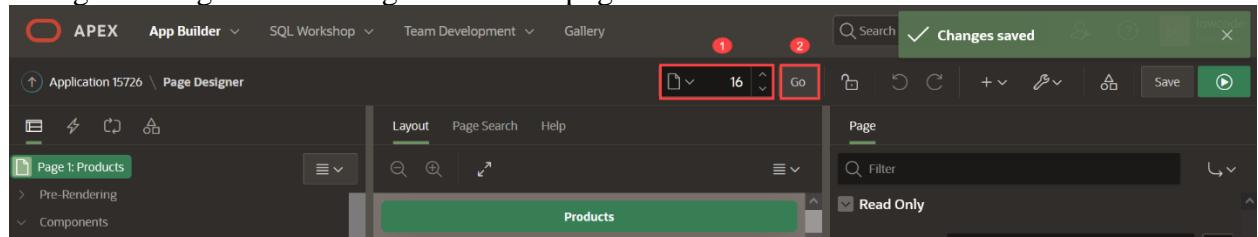
In the Property Editor (right pane), apply the following change:

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



3. Click **Save**.

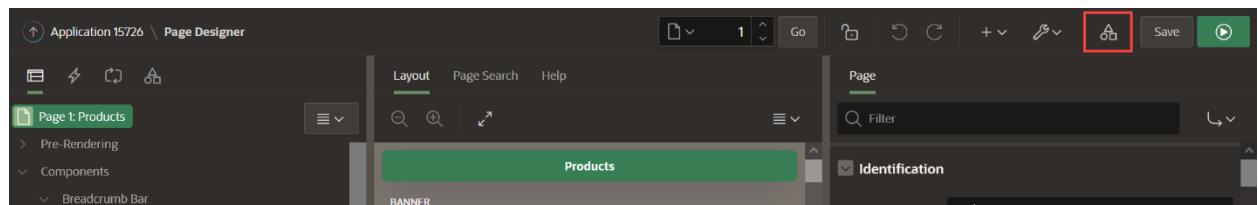
4. Navigate to Page Finder and go to the next page.



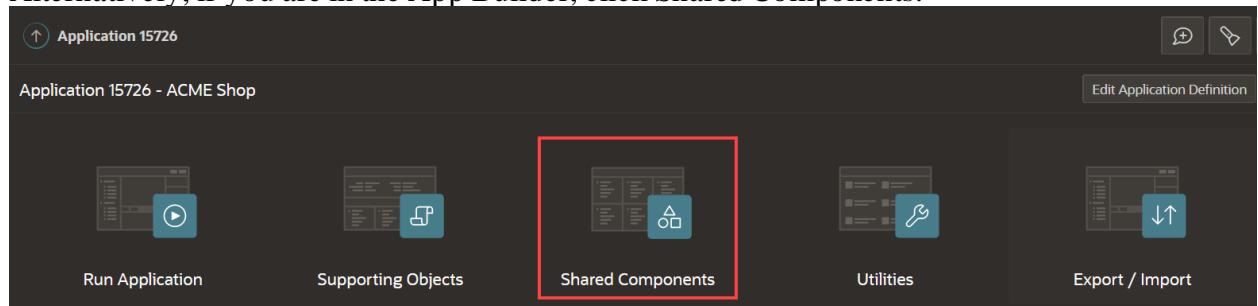
Task 2: Clean the Navigation Menu

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**.

The screenshot shows the 'Shared Components' page for Application 15726. The 'User Interface' category is selected. Within 'User Interface', the 'User Interface Attributes' item is highlighted with a red box and a red number '1' above it. Other items in this section include 'Progressive Web App', 'Themes', 'Templates', and 'Email Templates'. The 'Navigation' section contains 'Lists', 'Navigation Menu', 'Breadcrumbs', and 'Navigation Bar List'. The 'Security' section includes 'Security Attributes', 'Authentication Schemes', 'Authorization Schemes', 'Application Access Control', and 'Session State Protection'. The 'Other Components' section lists 'Lists of Values', 'Plug-ins', 'Component Settings', and 'Shortcuts'. The 'Files and Reports' section contains 'Static Application Files', 'Static Workspace Files', 'Report Queries', and 'Report Layouts'.

3. Click **Navigation Menu**.

4. Set Display Navigation to Off.

The screenshot shows the 'User Interface' settings for Application 15726. The 'Navigation Menu' tab is selected, indicated by a red box and a red number '1' above it. Below this, the 'Display Navigation' toggle switch is shown in its off position, with a red box and a red number '2' next to it. The 'Apply Changes' button is highlighted with a red box and a red number '3' above it. Other tabs visible include 'Definition', 'Security', 'Globalization', 'User Interface', and 'Progressive Web App'. The 'Cancel' button is also visible.

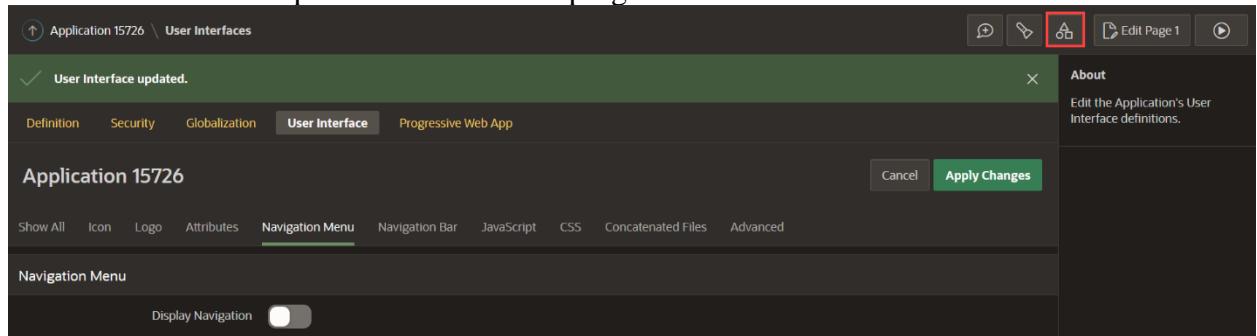
5. Click **Apply Changes**.

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**.

The screenshot shows the 'Navigation' category under 'Shared Components'. The 'Navigation Bar List' item is highlighted with a red box. Other items in the list include 'Lists', 'Navigation Menu', and 'Breadcrumbs'.

| Name | Type | Entries | References | Updated | Navigation Bar | Navigation Menu | Subscribed From | Subscribers |
|----------------|--------|---------|------------|------------|----------------|-----------------|-----------------|-------------|
| Navigation Bar | Static | 3 | 1 | 8 days ago | Yes | No | - | - |

3. Click **Navigation Bar**.

The screenshot shows the 'Lists' page with the 'Navigation Bar' entry highlighted with a red box. The table below shows the details of the 'Navigation Bar' list.

| Name | Type | Entries | References | Updated | Navigation Bar | Navigation Menu | Subscribed From | Subscribers |
|----------------|--------|---------|------------|------------|----------------|-----------------|-----------------|-------------|
| Navigation Bar | Static | 3 | 1 | 8 days ago | Yes | No | - | - |

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

The screenshot shows a software interface titled "List: Navigation Bar". At the top right are buttons for "Cancel", "Delete", and "Apply Changes". Below the title are tabs: "Show All" (highlighted in green), "Name", "List Entries", "Subscription", "Configuration", and "Comments". The main area is titled "Name" and contains a search bar with the placeholder "* Name" and a value "Navigation Bar". To the right of the search bar is a help icon (question mark). Below this is a table titled "List Entries". The table has columns: Sequence ↑=2, Name, Target, Icon, Authorization Scheme, Build Option, and Level. There are three rows in the table:

| Sequence ↑=2 | Name | Target | Icon | Authorization Scheme | Build Option | Level |
|--------------|------------|--------------|-------------|----------------------|--------------|-------|
| 10 | &APP_USER. | # | fa-user | | | |
| 20 | --- | separator | - | | | |
| 30 | Sign Out | &LOGOUT_URL. | fa-sign-out | | | |

 The bottom of the table shows "1 rows selected" and "Total 3".

- 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**

The screenshot shows the 'List Entry' configuration screen. The top navigation bar includes 'List Entry', 'Cancel', 'Create and Create Another', and 'Create List Entry'. Below the navigation is a toolbar with tabs: 'Show All', 'Entry' (which is selected), 'Target', 'Current List Entry', 'Conditions', 'Authorization', 'Configuration', 'Click Counting', 'User Defined Attributes', and 'Developer Resources'. The main area is divided into two sections: 'Entry' and 'Target'.

Entry Section:

- List: **Navigation Bar**
- Parent List Entry: - No Parent List Item -
- Sequence: **1**
- Image/Class: **&SHOPPING_CART_ICON.**
- Attributes: [empty]
- Alt Attribute: [empty]
- * List Entry Label: **Shopping Cart**

Target Section:

- Target type: Page in this Application
- * Page: **17**
- reset pagination for this page
- Printer Friendly
- Request: [empty]
- Clear Cache: **17**
- Set these items: [empty]
- With these values: [empty]
- URL Target: [empty]

Fields highlighted in red include: Sequence (1), Image/Class (&SHOPPING_CART_ICON.), List Entry Label (Shopping Cart), Page (17), and Clear Cache (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**

User Defined Attributes

Substitution strings #A01# - #A10# are optional attributes within a list template, which can be used to reference list entry user defined attributes 1 - 10, respectively. Click **List Template Attributes** to view a more meaningful description of the substitution strings in use in the list templates available in the current application.

Theme: Universal Theme - 42 | ?

List Template: Navigation Bar (1 references) | ? View List Utilization

Translatable

1. Badge Value: &SHOPPING_CART_ITEMS. | ?

2. List Item CSS Classes: js-shopping-cart-item | ?

3. | ?

6. Click Create List Entry.

Application 15726 \ Shared Components \ Lists \ List Details \ Create / Edit

< > List Entry Cancel Create and Create Another **Create List Entry**

Show All Entry Target Current List Entry Conditions Authorization Configuration Click Counting User Defined Attributes Developer Resources

Entry

List: **Navigation Bar**

Parent List Entry: - No Parent List Item - | ?

Sequence: 1 | ?

Image/Class: &SHOPPING_CART_ICON. | ?

Attributes: | ?

Alt Attribute: | ?

7. Click &APP_USER.

| Sequence | Name | Target | Icon | Authorization Scheme | Build Option | Level |
|----------|---------------|--|--------------|----------------------|--------------|-------|
| 1 | Shopping Cart | f?p=&APP_ID.:17:&SESSION.:&DEBUG.:17:: | &SHOPPING... | | | |
| 10 | &APP_USER. | # | fa-user | | | |
| 20 | --- | separator | - | | | |
| 30 | Sign Out | &LOGOUT_URL. | fa-sign-out | | | |

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.

The screenshot shows the 'List Entry' creation screen in Oracle APEX. The top navigation bar includes 'Cancel', 'Create and Create Another', and a green 'Create List Entry' button with a red border and a red number 5. Below the header are tabs: 'Show All' (highlighted in green), 'Entry' (selected), 'Target', 'Current List Entry', 'Conditions', 'Authorization', 'Configuration', 'Click Counting', 'User Defined Attributes', and 'Developer Resources'. The main area is divided into 'Entry' and 'Target' sections. In the 'Entry' section, the 'List' is set to 'Navigation Bar'. The 'Parent List Entry' dropdown is set to '- No Parent List Item -'. The 'Sequence' field contains '5' with a red box and a red number 1. The 'Image/Class' field contains 'fa-wrench' with a red box and a red number 2. The 'Attributes' and 'Alt Attribute' fields are empty. The 'List Entry Label' field contains 'Administration' with a red box and a red number 3. In the 'Target' section, the 'Target type' is set to 'Page in this Application' and the 'Page' field contains '10000' with a red box and a red number 4.

12. Click Run Page and view the updated application.

The screenshot shows the 'List Details' page after saving changes. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', 'Gallery', and user information 'lowcode'. The left sidebar shows the application structure: 'Application 15726 \ Shared Components \ Lists \ List Details'. A green banner at the top says 'Action processed.'. The main content area shows the 'List: Navigation Bar' entry with a 'Page' value of '10000'. There are 'Cancel', 'Delete', and a green 'Apply Changes' button. To the right, there is a note: 'List can be static or dynamic. When the list is static, the list may contain any number of list entries. For dynamic lists, the number of list entries depends on the application logic.'

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