



Practice: Use SQL Workshop

Practice 1: Install a Sample Data Set

Overview

In this practice, you 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.

In this practice, you will:

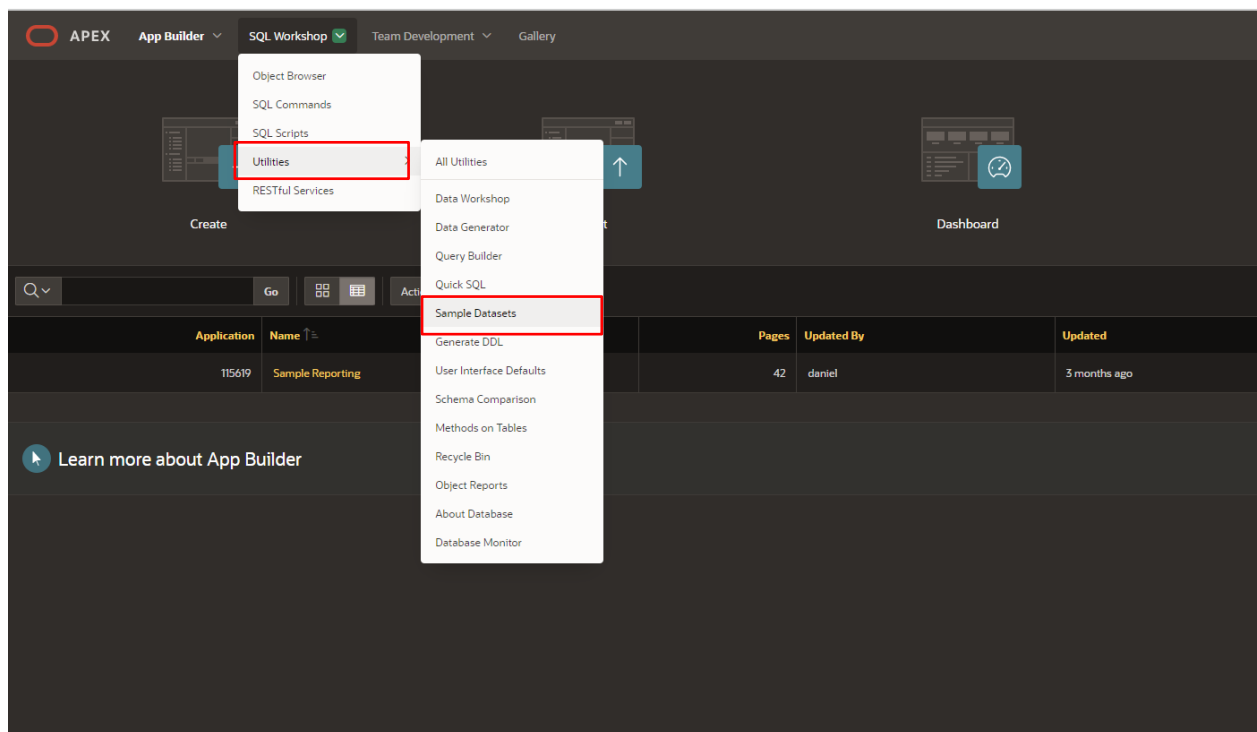
- Install a sample dataset into your Oracle APEX Workspace

Task

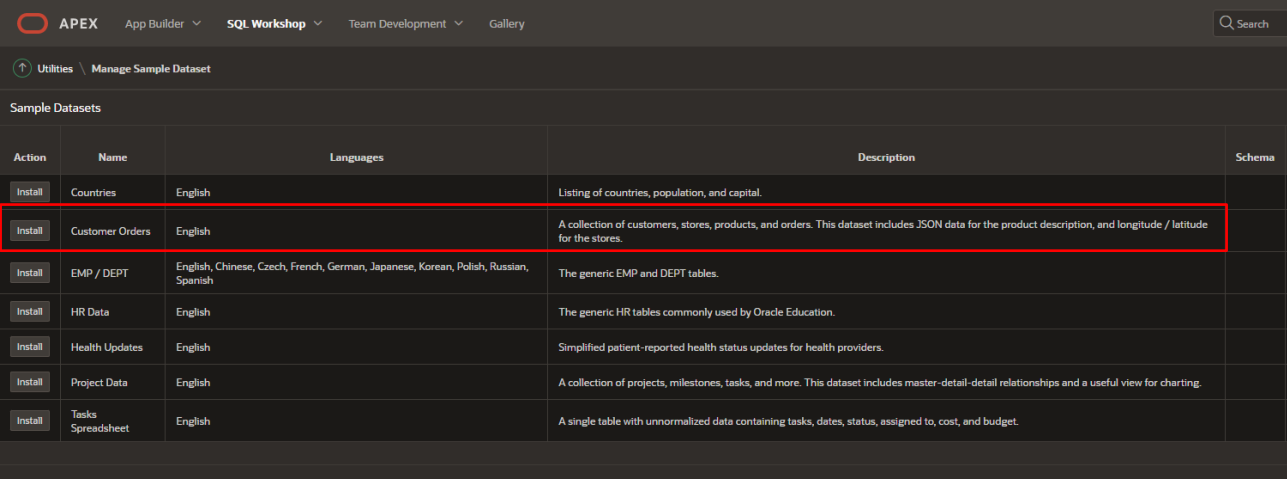
1. Log in to your workspace. If you are already logged in, click the **APEX** Logo in the upper left-hand corner of the page to return to the APEX workspace home page.

Create Customer Order Tables

2. From the APEX workspace home page, select the down arrow to the right of **SQL Workshop**, then select **Utilities** to display the options and choose **Sample Datasets**.



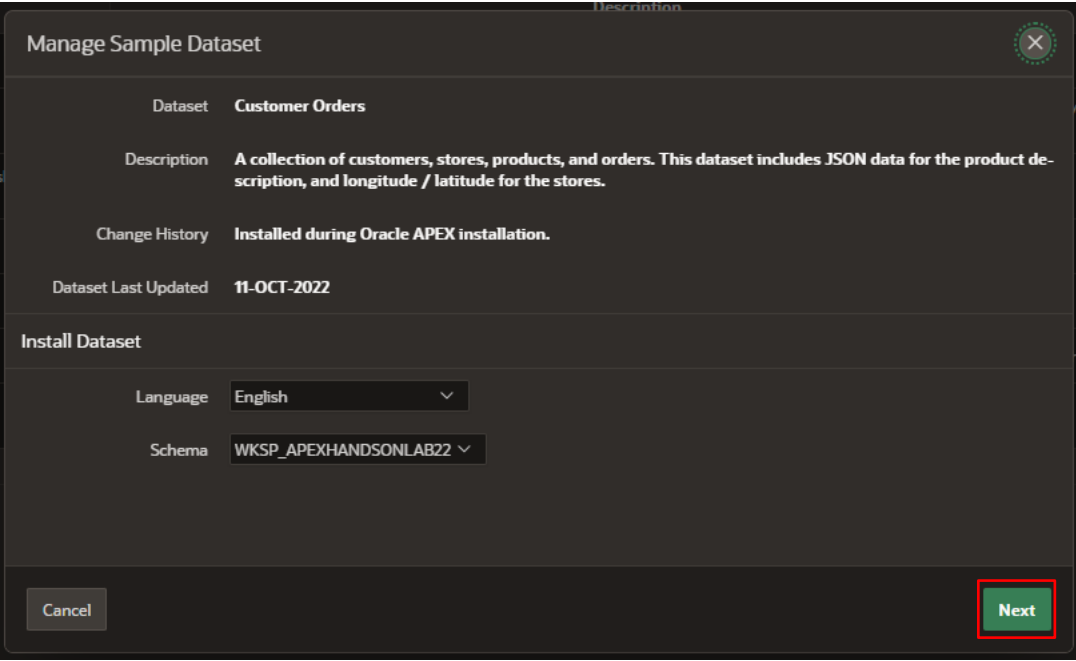
- On the **Customer Orders** row, click Install.



Action	Name	Languages	Description	Schema
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.	
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.	
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.	
Install	Tasks Spreadsheet	English	A single table with unnormalized data containing tasks, dates, status, assigned to, cost, and budget.	

- Leave the settings to default and then click **Next**.

The schema name defaults to your current schema and hence 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: **11-OCT-2022**

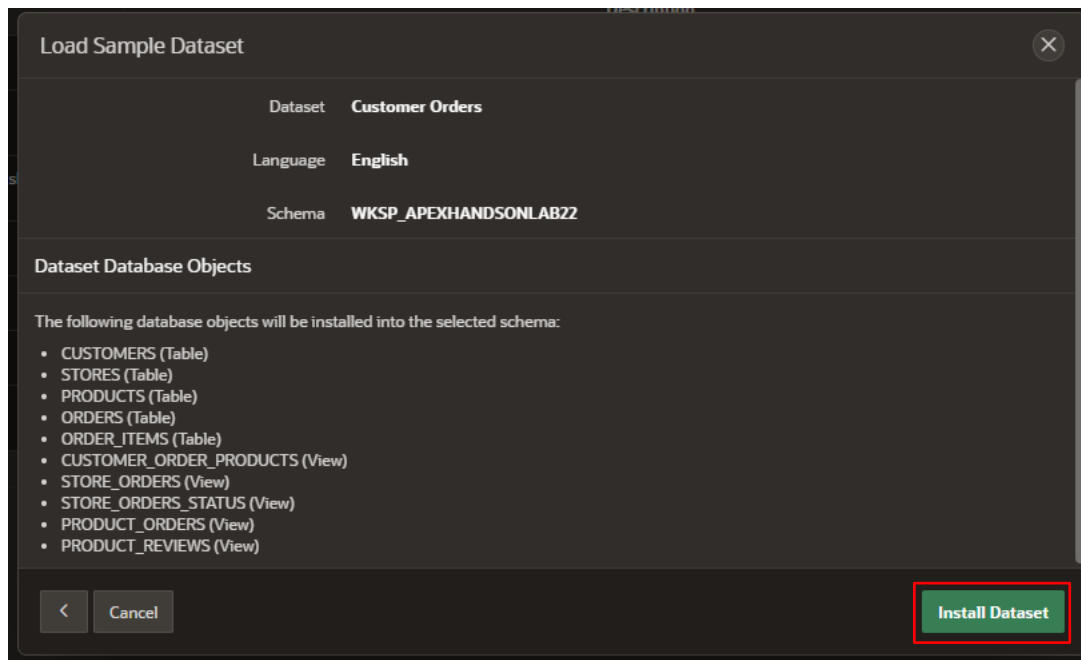
Install Dataset

Language: **English**

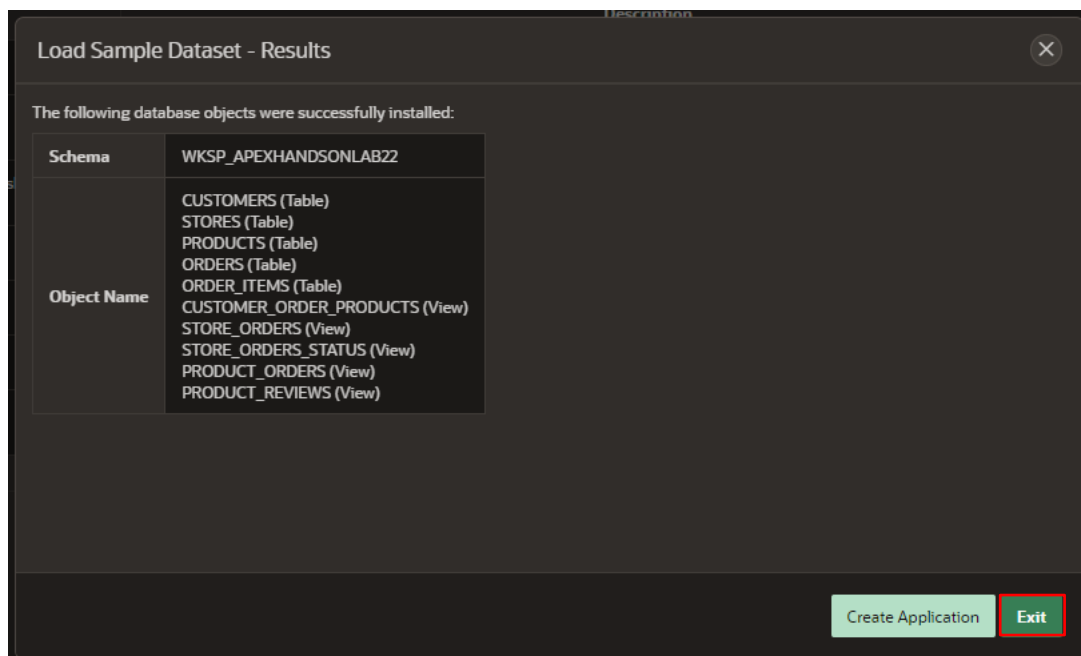
Schema: **WKSP_APEXHANDSONLAB22**

Next

5. Click **Install Dataset**.



6. Click **Exit**.



You now know how to install sample tables and views from Sample Datasets. You may now **proceed to the next practice**.

Practice 2: Add Columns to the PRODUCTS Table

Overview

The **PRODUCTS** table includes some columns such as image, price, and details. But there are other characteristics that customers would appreciate knowing about a product, such as color, type of clothing, and department. In this practice, you will add these columns to the **PRODUCTS** table.

Since many products share the same colors, clothing type, and department, to avoid data redundancy it is a best practice to create three additional tables to store the unique values of color, type, and department data. Instead of creating these three tables for yourself, you'll use the **Create Lookup Table** feature.

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

In this practice, you will:

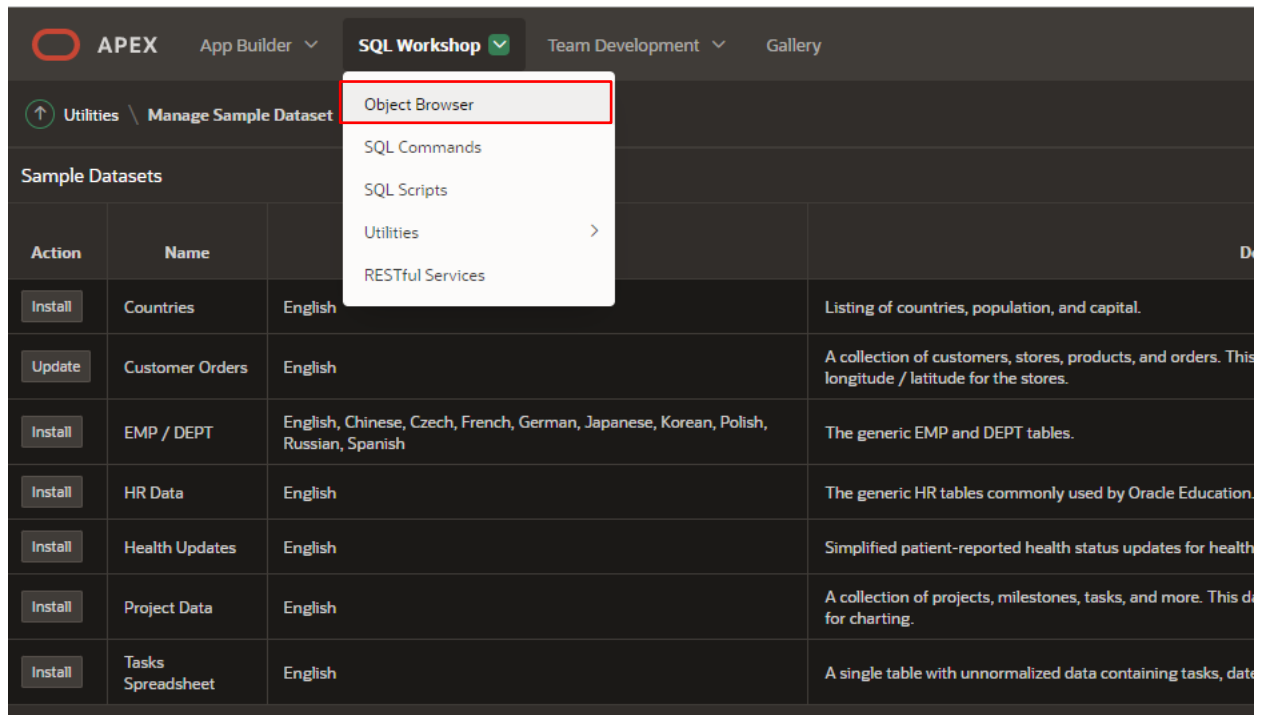
- Add new columns to the existing **PRODUCTS** table
- Populate the new columns
- Create lookup tables

Tasks

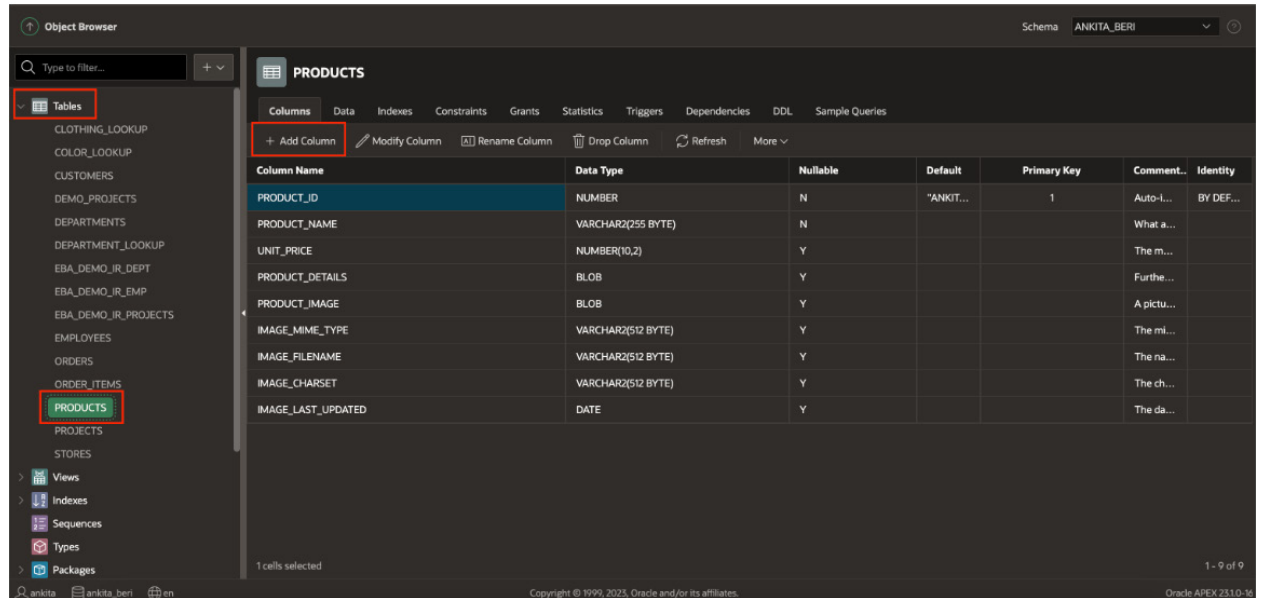
Add Column Title to the PRODUCTS Table

1. From the APEX Main Menu, click **SQL Workshop**.

2. Click **Object Browser**.



3. Navigate to **PRODUCTS**. Click the **Add Column** button.



4. For the Color column, enter the following:

- Add Column - enter **COLOR**
- Type - select **VARCHAR2**
- Length - enter **200**

Click **Apply**.

The screenshot shows the 'Add Column' dialog box in SQL Developer. The fields are as follows:

- Owner: ANKITA_BERI
- Table: PRODUCTS
- * Column: COLOR
- Data Type: VARCHAR2
- Semantics: Default
- Length: 200
- Default: (empty)
- Nullable: ☒

The 'Apply' button at the bottom right is highlighted with a red box.

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

Click **Apply**.

×

Add Column

Owner

ANKITA_BERI

Table

PRODUCTS

* Column

DEPARTMENT

?

Data Type

VARCHAR2

?

Semantics

Default

?

Length

200

?

Default

?

Nullable

☒

?

Cancel

Apply

7. Click the **Add Column** button.

APEX

App Builder

SQL Workshop

Team Development

Gallery

Search

Object Browser

Tables

CUSTOMERS

EBA_DEMO_IR_DEPT

EBA_DEMO_IR_EMP

EBA_DEMO_IR_PROJECTS

ORDERS

ORDER_ITEMS

PRODUCTS

STORES

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
PRODUCT_ID	NUMBER	No	"WKSP_APEXHANDSONLAB22"."ISEQ\$\$_173406532".nextval
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	VARCHAR2(200)	Yes	-
DEPARTMENT	VARCHAR2(200)	Yes	-

Download | Print

8. For the Clothing column, enter the following:

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

Click **Next**.

The screenshot shows the 'Add Column' dialog box in Oracle SQL Developer. The dialog has a dark background and a close button (X) in the top right corner. The fields are as follows:

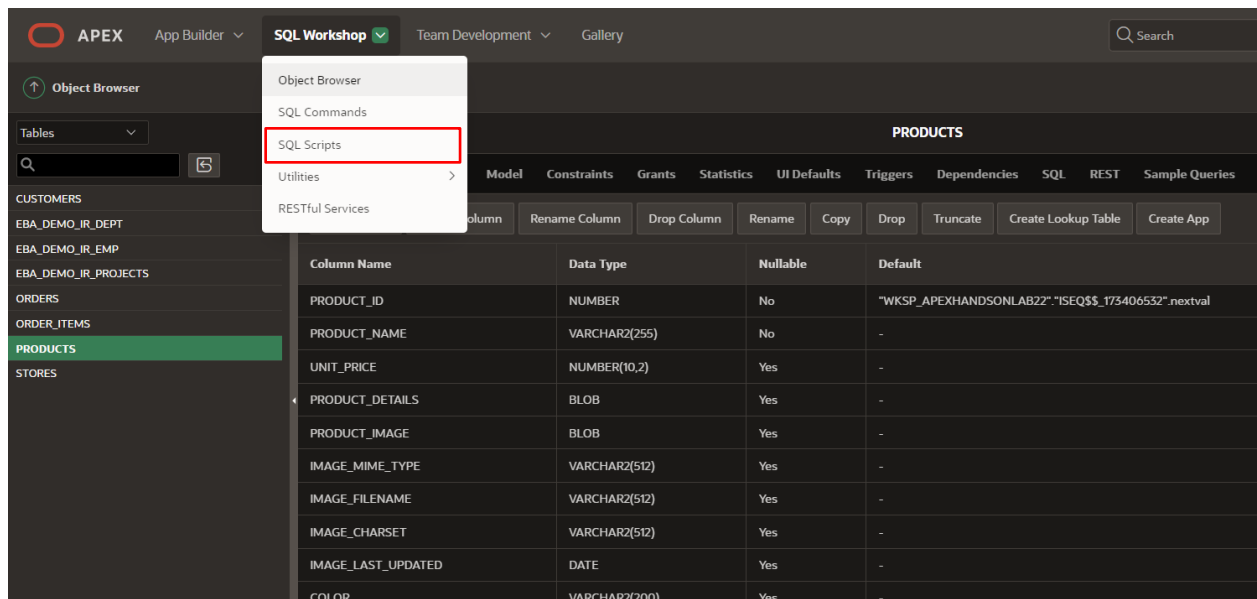
- Owner: ANKITA_BERI
- Table: PRODUCTS
- * Column: CLOTHING (highlighted with a red box)
- Data Type: VARCHAR2 (highlighted with a red box)
- Semantics: Default
- Length: 200 (highlighted with a red box)
- Default: (empty)
- Nullable: ☒ (highlighted with a red box)

At the bottom, there are two buttons: 'Cancel' on the left and 'Apply' on the right (highlighted with a red box).

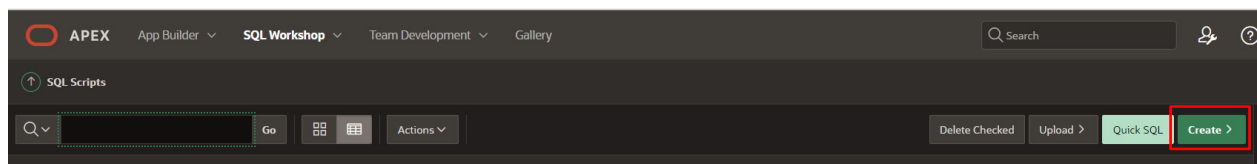
Populate New Columns

1. From the APEX main menu, click **SQL Workshop**.

2. Click **SQL Scripts**.



3. Click **Create**.



4. For Script Name, enter **Populating new columns**.

5. Copy the following script and paste it into the editor.

```
UPDATE
(
    SELECT p.product_id,
           p.product_name,
           p.clothing,
           p.color,
           p.department,
           p.product_details
    FROM   products p ) p
SET      p.clothing = Substr(product_name, Instr(product_name, '
',1,1)+1, Instr(product_name, ' ',1, 2)+1 - Instr(product_name,
' ',1,1)- 2),
        p.color =
(
    SELECT c.color
```

```

        FROM    json_table (p.product_details, '$' COLUMNS (
color VARCHAR2(4000) path '$.colour') ) c),
        p.department =
        (
            SELECT g.department
            FROM    json_table (p.product_details, '$' COLUMNS (
department VARCHAR2(4000) path '$.gender') ) g)

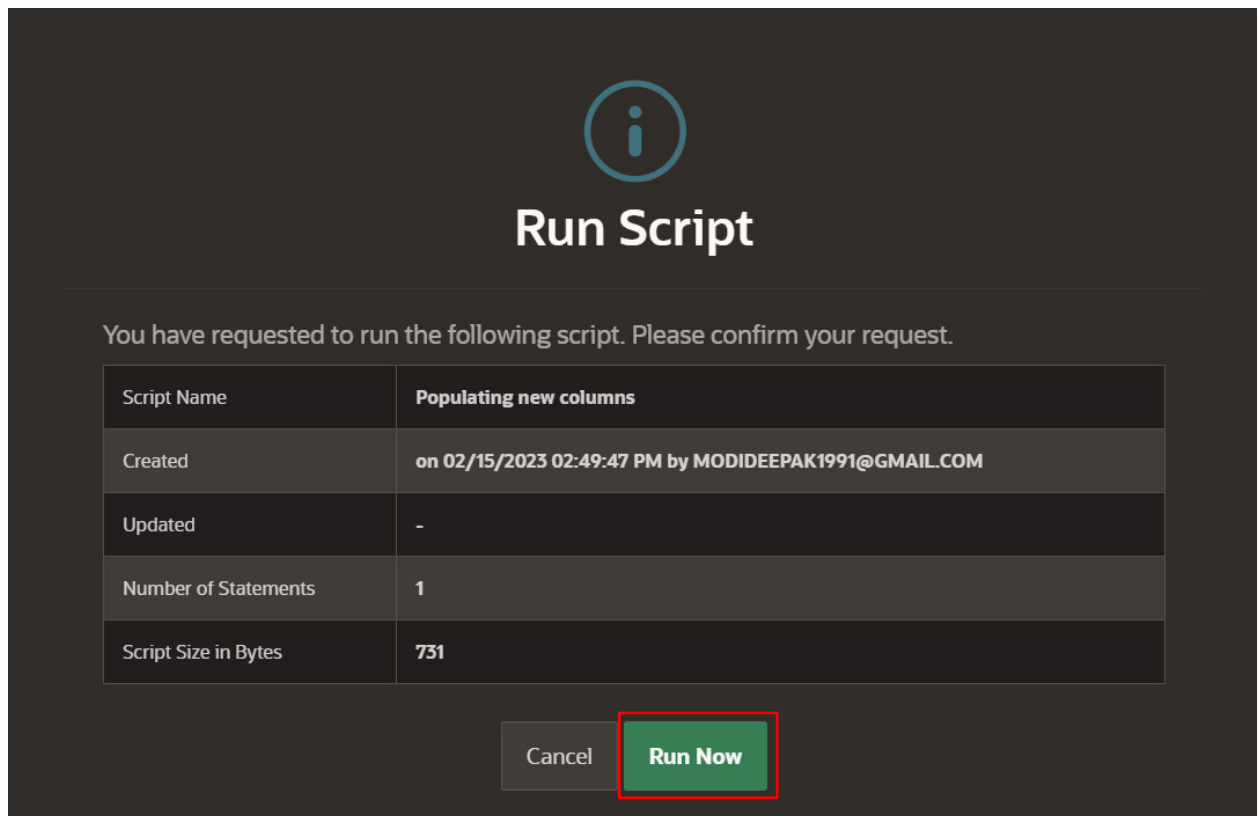
```

This script inserts the unique product type values (e.g. Shirt, Jacket, Skirt, etc.) into the CLOTHING column in the **PRODUCTS** table. Similarly, it inserts the unique department names (e.g. Boys', Girls', 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.

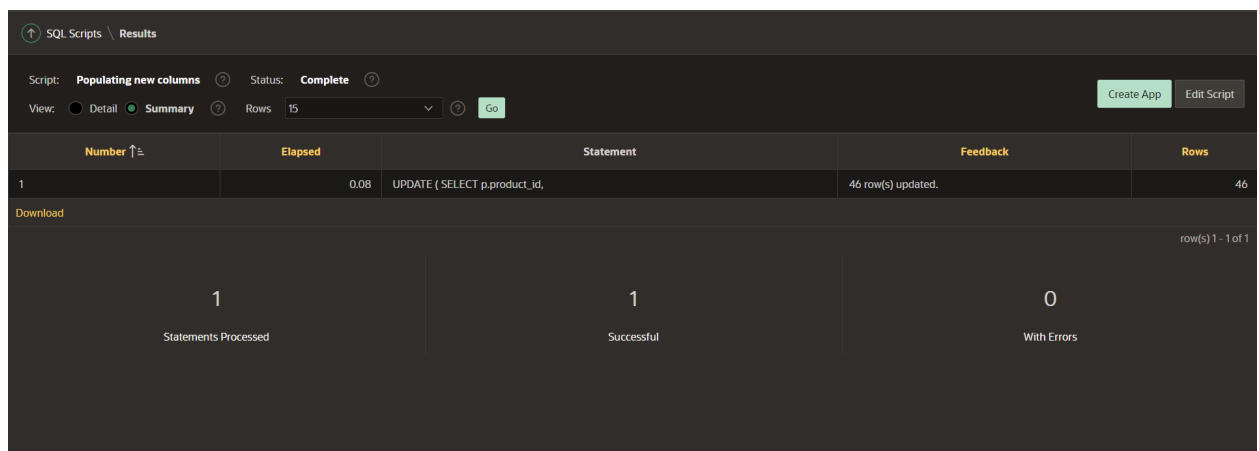
6. Click **Run**.



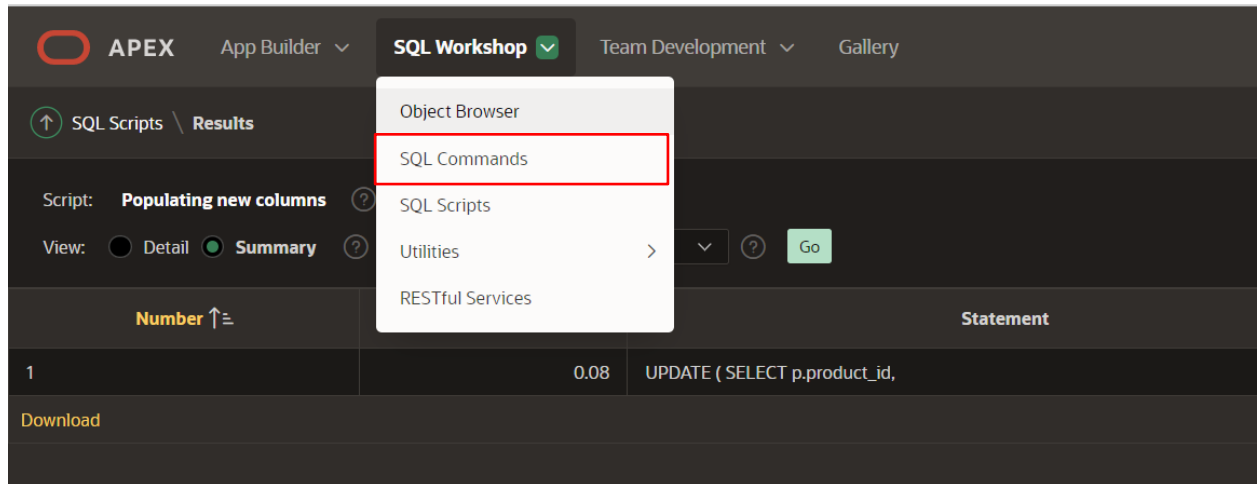
- Click **Run Now**.



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



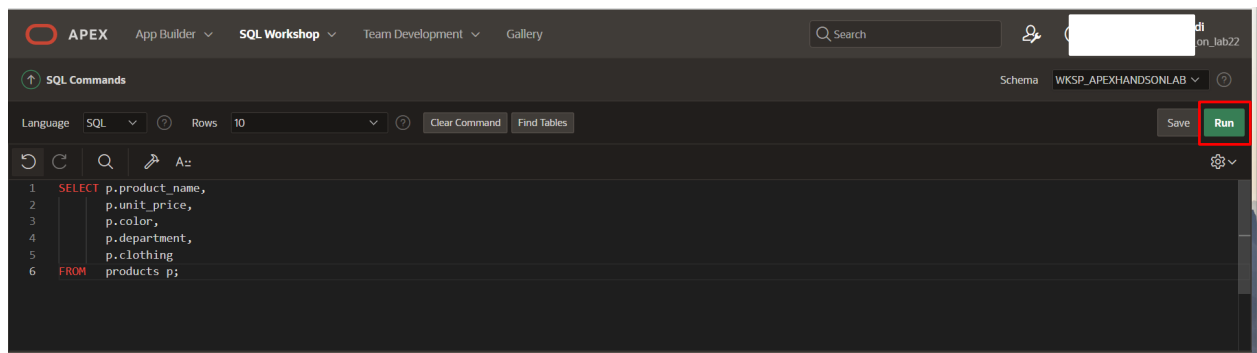
9. To check the values in the **PRODUCTS** table, click **SQL Workshop** and click **SQL Commands**.



10. Copy the following SQL query.

```
SELECT p.product_name,  
       p.unit_price,  
       p.color,  
       p.department,  
       p.clothing  
FROM   products p;
```

11. Click **Run**.

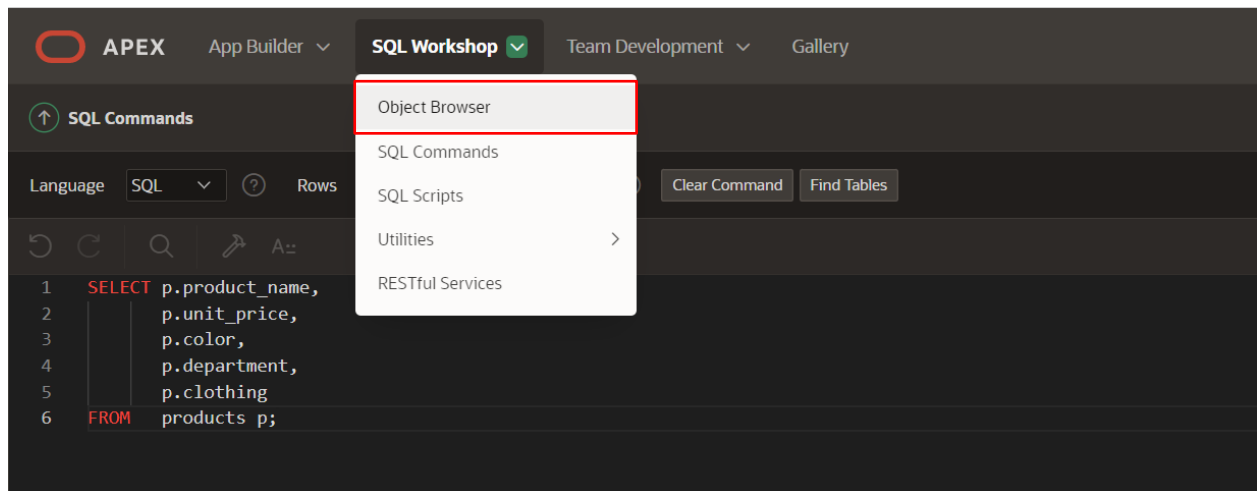


Create Lookup Tables

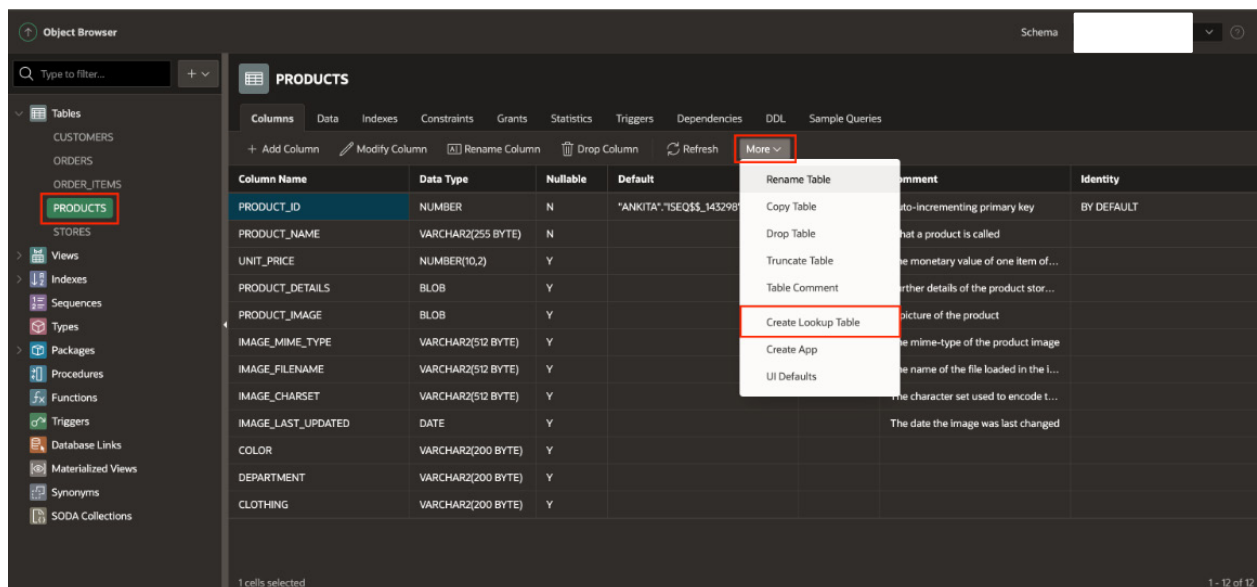
Since multiple products may have the same values for Color, Department, and Clothing, to avoid repetition and make updates easy, you can create a lookup table for each. A lookup table stores the value of the available colors, departments, or clothing in a single place, and then each product can reference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

In this practice, you create lookup tables based on the new three columns. After you create a lookup table, this allows the `PRODUCTS` table to reference the new color, clothing type, and department lookup value by its new numeric primary key value in the lookup table.

1. From the APEX main menu, click **SQL Workshop**.
2. Click **Object Browser**.



3. Navigate to the `PRODUCTS` table.
4. Click the **Create Lookup Table** button.



- For **Text Column**, select **COLOR** and Click **Create Lookup Table**.

Create Lookup Table

Table **PRODUCTS**

* Text Column **COLOR**

* Lookup Table Name **COLOR_LOOKUP**

SQL Script

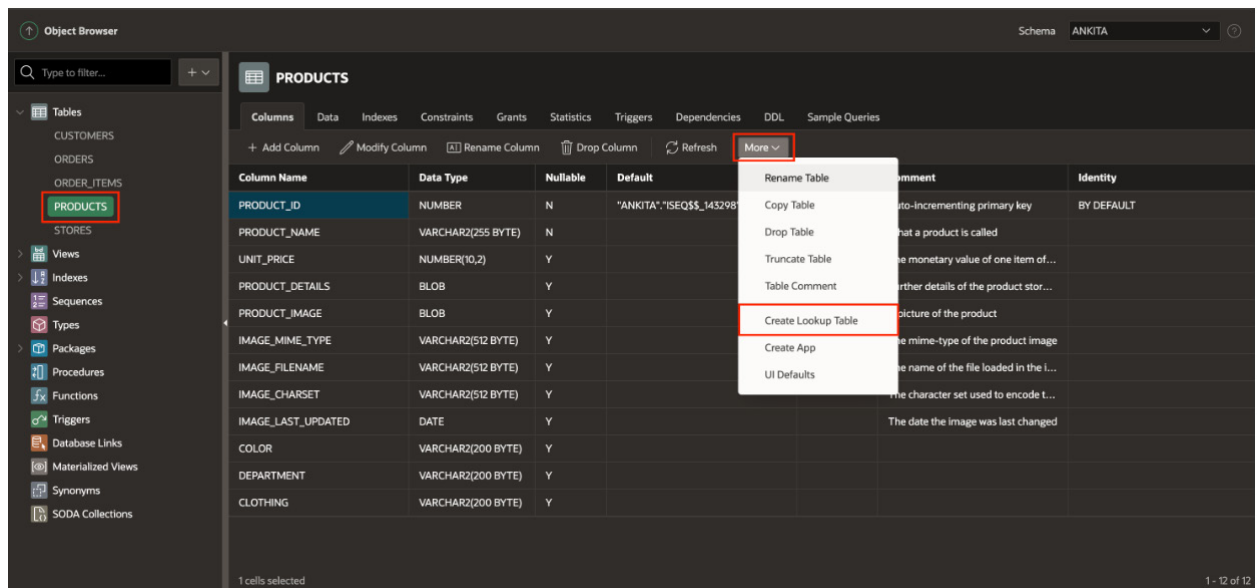
```
create table "COLOR_LOOKUP" (  
  "COLOR_ID" number generated by default on null as identity,  
  "COLOR" varchar2(4000 byte) not null,  
  constraint "COLOR_LOOKUP_PK" primary key ("COLOR_ID")  
);  
  
insert into "COLOR_LOOKUP" ("COLOR")  
select distinct "COLOR"  
from "PRODUCTS"  
where "COLOR" is not null;  
  
alter table "PRODUCTS" add "COLOR_ID" number;  
  
update "PRODUCTS" x  
set "COLOR_ID" = (select "COLOR_ID" from "COLOR_LOOKUP" where "COLOR" = x."COLOR");  
  
alter table "PRODUCTS" drop column "COLOR";  
  
alter table "PRODUCTS"  
add constraint "PRODUCTS_COLOR_FK"  
foreign key ("COLOR_ID")  
references "COLOR_LOOKUP"("COLOR_ID");
```

Cancel

Create Lookup Table

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

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



- For Text Column, select **DEPARTMENT** and Click **Create Lookup Table**.

Create Lookup Table

Table **PRODUCTS**

* Text Column **DEPARTMENT**

* Lookup Table Name **DEPARTMENT_LOOKUP**

SQL Script

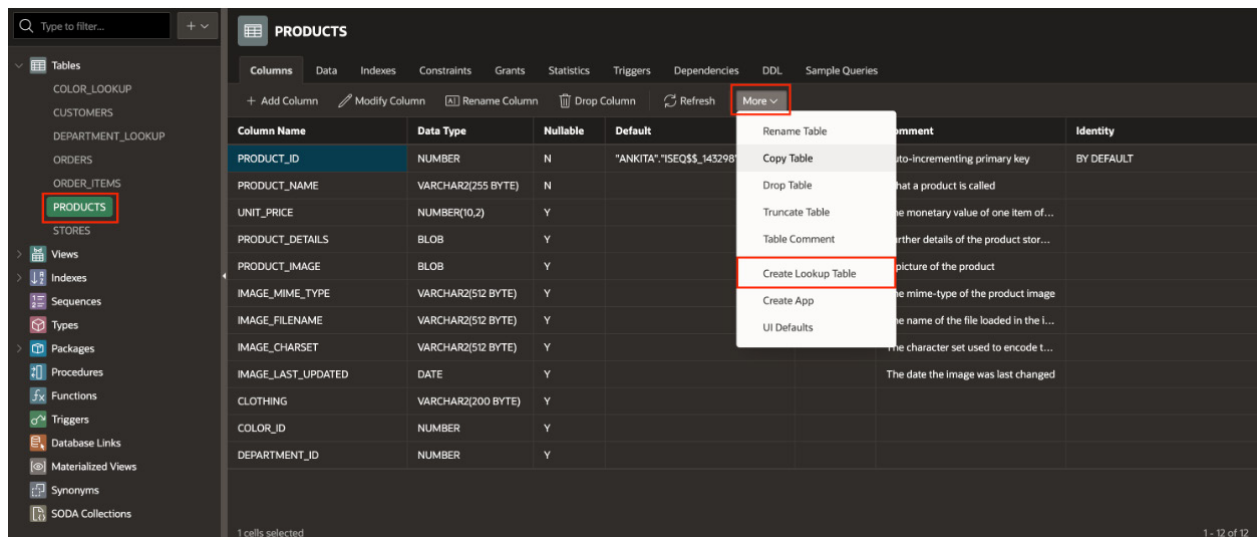
```
create table "DEPARTMENT_LOOKUP" (  
  "DEPARTMENT_ID" number generated by default on null as identity,  
  "DEPARTMENT" varchar2(4000 byte) not null,  
  constraint "DEPARTMENT_LOOKUP_PK" primary key ("DEPARTMENT_ID")  
);  
  
insert into "DEPARTMENT_LOOKUP" ("DEPARTMENT")  
select distinct "DEPARTMENT"  
from "PRODUCTS"  
where "DEPARTMENT" is not null;  
  
alter table "PRODUCTS" add "DEPARTMENT_ID" number;  
  
update "PRODUCTS" x  
set "DEPARTMENT_ID" = (select "DEPARTMENT_ID" from "DEPARTMENT_LOOKUP" where "DEPARTMENT" =  
x."DEPARTMENT");  
  
alter table "PRODUCTS" drop column "DEPARTMENT";  
  
alter table "PRODUCTS"  
add constraint "PRODUCTS_DEPARTMENT_FK"  
foreign key ("DEPARTMENT_ID")  
references "DEPARTMENT_LOOKUP"("DEPARTMENT_ID");
```

Cancel

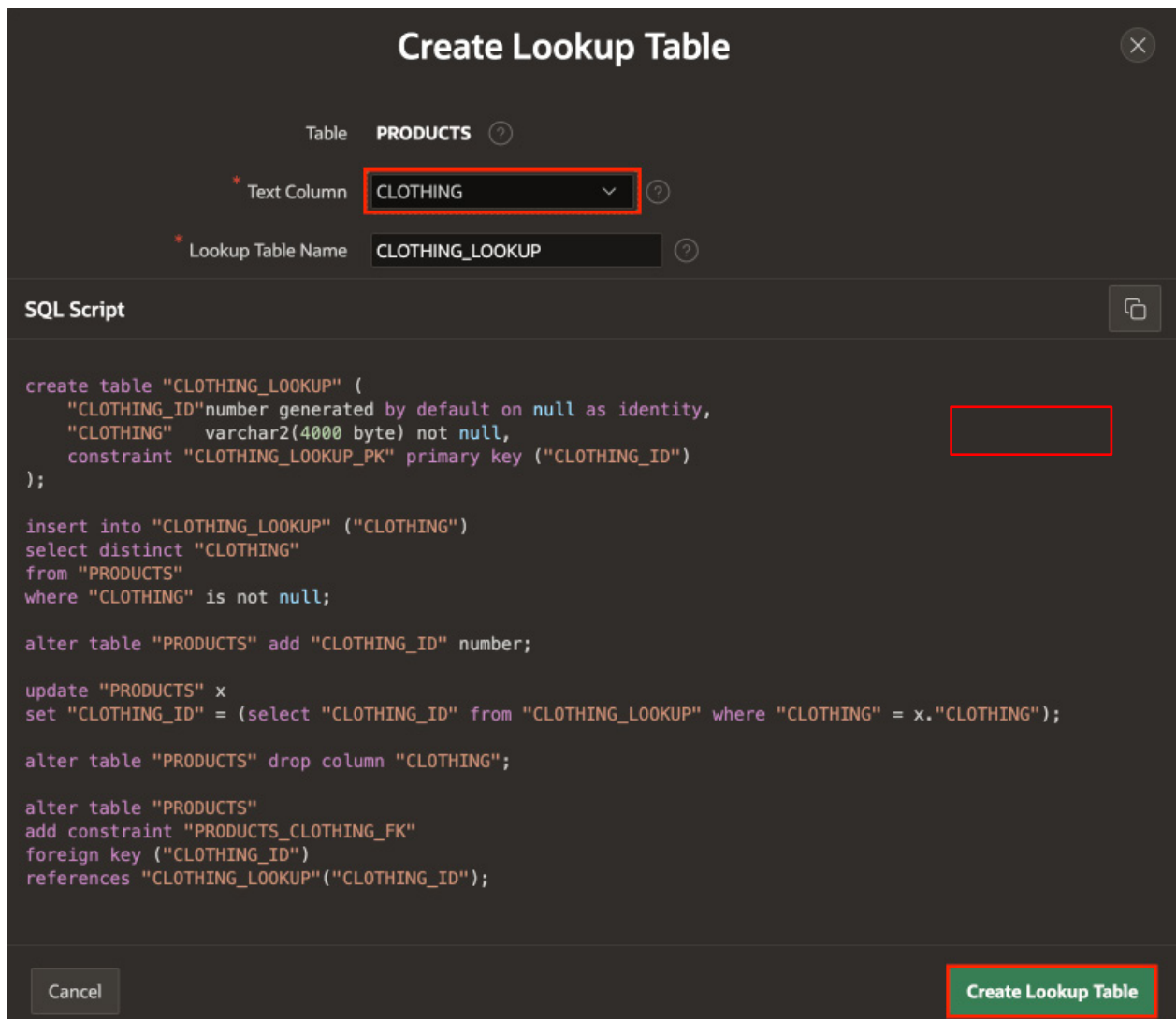
Create Lookup Table

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

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



9. For Text Column, select **CLOTHING** and 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.*

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

- COLOR_LOOKUP
- DEPARTMENT_LOOKUP
- CLOTHING_LOOKUP

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

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									
PRODUCT_ID	NUMBER	No	"WKSP_APEXHANDSONLAB22"."ISEQ\$\$_173406532".nextval									
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	-									
Download Print												

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 practice**.

Practice 3: Create a Database Package for Business Logic

Overview

In this practice, 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 an order, clear the cart, and more.

To manage items in the cart, you 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.

Business logic in APEX applications can be written using PL/SQL, Oracle's procedural language extension to SQL. PL/SQL offers a handy program unit called a "package" that allows you to cleanly separate the API signatures for a set of reusable procedures and functions, from their implementation.

In this practice, you will:

- Create a package to manage the Shopping Cart

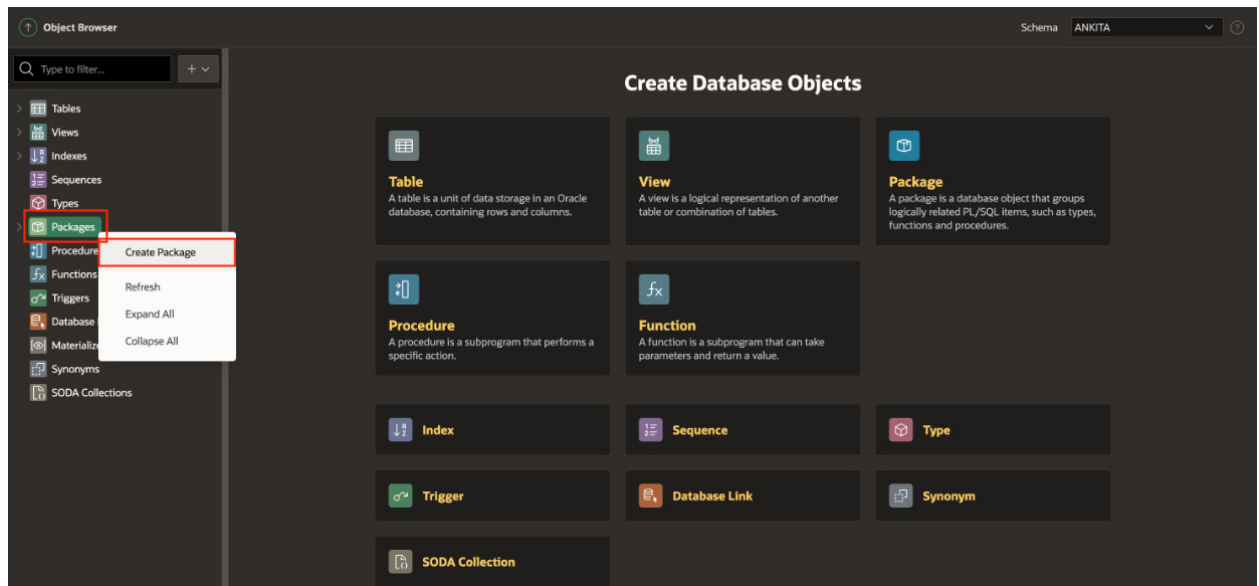
Task

Create the Package

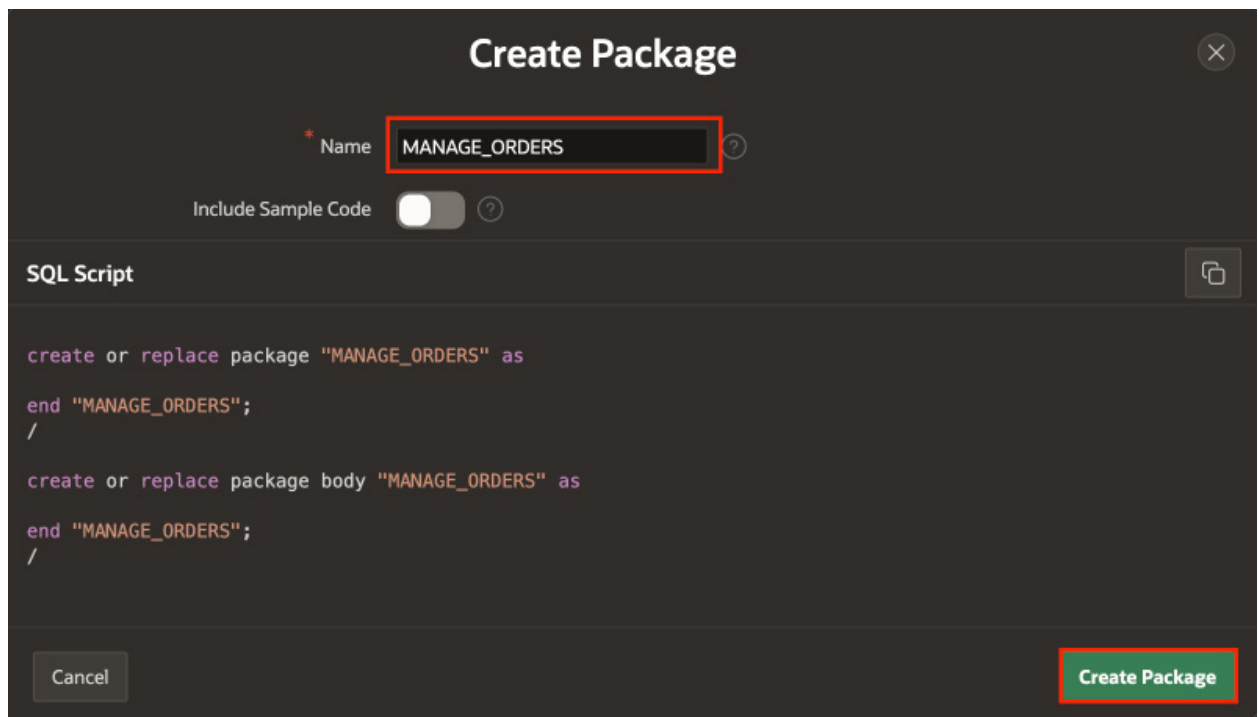
Create specification and body for the package.

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

2. Navigate to the **Packages** on the left side, Right click on it and Select **Create Package**.



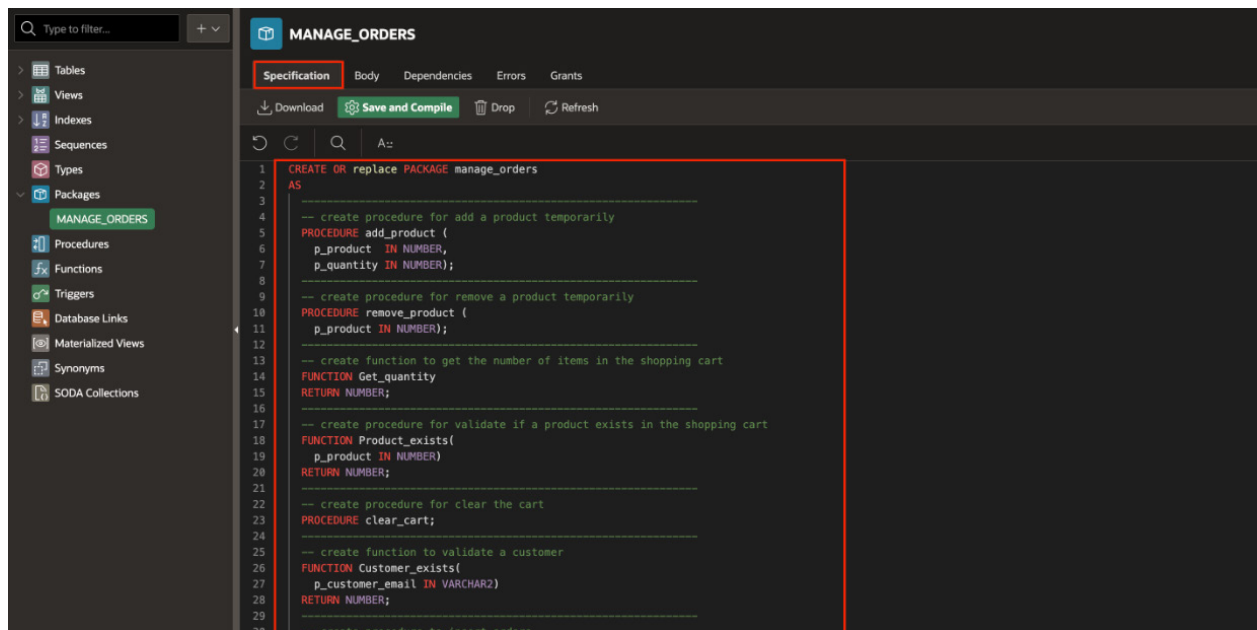
3. For Package Name, enter **MANAGE_ORDERS** and click **Create Package**.



4. For Specification, replace the contents of the code editor with the following:

```
CREATE OR replace PACKAGE manage_orders
AS
-----
-- create procedure for add a product temporarily
PROCEDURE add_product (
    p_product  IN NUMBER,
    p_quantity IN NUMBER);
-----
-- create procedure for remove a product temporarily
PROCEDURE remove_product (
    p_product IN NUMBER);
-----
-- create function to get the number of items in the shopping
cart
FUNCTION Get_quantity
RETURN NUMBER;
-----
-- create procedure for validate if a product exists in the
shopping cart
FUNCTION Product_exists(
    p_product IN NUMBER)
RETURN NUMBER;
-----
-- create procedure for clear the cart
PROCEDURE clear_cart;
-----
-- create function to validate a customer
FUNCTION Customer_exists(
    p_customer_email IN VARCHAR2)
RETURN NUMBER;
-----
-- create procedure to insert orders
PROCEDURE create_order (
    p_customer          IN VARCHAR2 DEFAULT NULL,
    p_customer_email IN VARCHAR2,
    p_store              IN NUMBER,
    p_order_id          OUT orders.order_id%TYPE,
    p_customer_id       OUT NUMBER );
END manage_orders;
```

5. Click **Save and Compile** to save the changes.



6. Navigate to the body part of the package by clicking the Body tab and replace the contents of the code editor with the following:

```
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
            SELECT seq_id
            INTO    l_id
            FROM    apex_collections a
            WHERE   collection_name = 'PRODUCTS'
                    AND a.n001 = p_product;

            apex_collection.delete_member(p_collection_name =>
            'PRODUCTS',
                                           p_seq => l_id);

        END IF;
    END remove_product;

    FUNCTION get_quantity
    RETURN NUMBER
    IS
```

```

        l_items NUMBER := 0;
BEGIN
    IF apex_collection.collection_exists (p_collection_name =>
'PRODUCTS')
    THEN
        SELECT SUM(n002)
        INTO    l_items
        FROM    apex_collections a
        WHERE   collection_name = 'PRODUCTS';
    END IF;

    RETURN l_items;
END get_quantity;

FUNCTION product_exists(p_product IN NUMBER)
RETURN NUMBER
IS
    l_quantity NUMBER;
BEGIN
    IF apex_collection.collection_exists (p_collection_name =>
'PRODUCTS')
    THEN
        SELECT a.n002
        INTO    l_quantity
        FROM    apex_collections a
        WHERE   collection_name = 'PRODUCTS'
                AND a.n001 = p_product;

        RETURN l_quantity;
    ELSE
        RETURN 0;
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RETURN 0;
END product_exists;

PROCEDURE clear_cart
IS
BEGIN
    IF apex_collection.collection_exists (p_collection_name =>
'PRODUCTS')
    THEN

```

```

        apex_collection.truncate_collection(p_collection_name =>
'PRODUCTS');
    END IF;
END clear_cart;

FUNCTION customer_exists(p_customer_email IN VARCHAR2)
RETURN NUMBER
IS
    l_customer customers.customer_id%TYPE;
BEGIN
    SELECT customer_id
    INTO    l_customer
    FROM    customers
    WHERE   email_address = p_customer_email;

    RETURN l_customer;
EXCEPTION
    WHEN no_data_found THEN
        RETURN 0;
END customer_exists;

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)
IS
BEGIN
    p_customer_id := customer_exists(p_customer_email);

    IF p_customer_id = 0 THEN
        INSERT INTO customers
            (full_name,
             email_address)
        VALUES
            (p_customer,
             p_customer_email)
        returning customer_id INTO p_customer_id;
    END IF;

    INSERT INTO orders
        (order_datetime,
         customer_id,

```

```

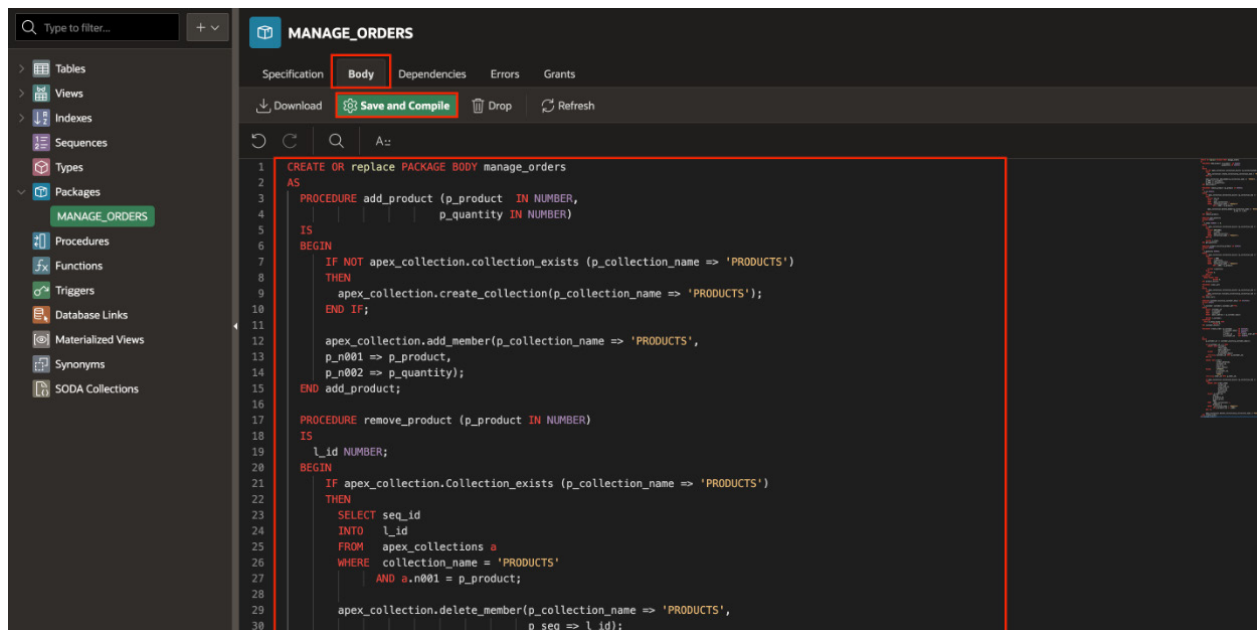
        store_id,
        order_status)
VALUES      (SYSDATE,
            p_customer_id,
            p_store,
            'OPEN')
returning order_id INTO p_order_id;

IF apex_collection.collection_exists (p_collection_name =>
'PRODUCTS')
THEN
    INSERT INTO order_items
        (order_id,
         line_item_id,
         product_id,
         unit_price,
         quantity)
    SELECT p_order_id,
           seq_id,
           p.product_id,
           p.unit_price,
           n002
    FROM   apex_collections a,
           products p
    WHERE  collection_name = 'PRODUCTS'
           AND p.product_id = a.n001;
END IF;

apex_collection.delete_collection(p_collection_name =>
'PRODUCTS');
END create_order;
END manage_orders;

```

7. Click **Save & Compile**.



While you don't have to understand the code to complete the workshop successfully, know that the functions and procedures you've defined in this practice 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 a 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 practices, you will call these procedures and functions when it is required. You may now **proceed to the next practice**.

Practice 4: Create Database Objects Using Quick SQL

Overview

Memorizing the SQL to create and maintain database objects can be difficult. However, there are tools that can generate the code for you based on a shorthand syntax or a graphical user interface. In this practice, you will create database objects using Quick SQL and then view the objects in Database Actions. Then you will then use the Data Workshop page to load data from a file into your workspace.

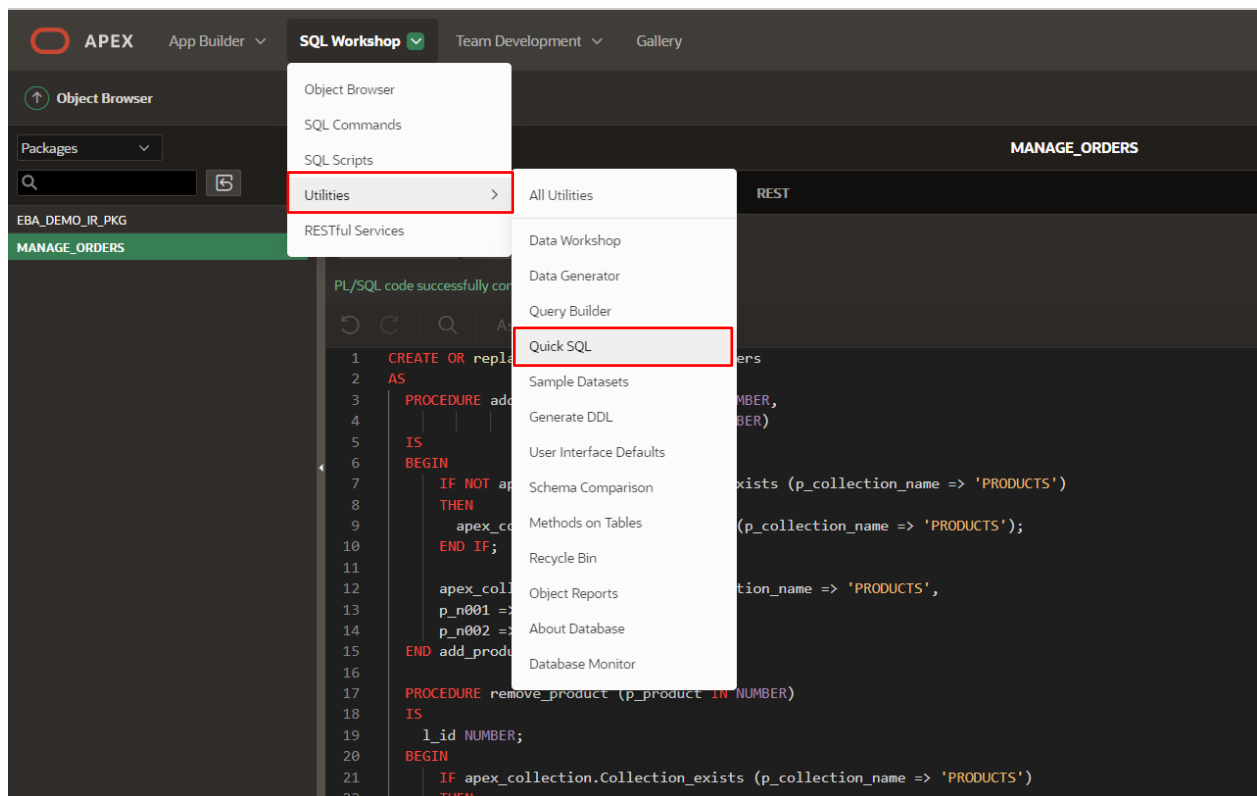
In this practice, you'll:

- Create database objects using Quick SQL
- Load data using the Data Workshop page

Tasks

Create database objects using Quick SQL

1. Return to your APEX Workspace and click 'v' next to the **SQL Workshop** tab, click **Utilities**, and then select **Quick SQL**.



- Copy the following SQL shorthand code and paste it into the left pane in Quick SQL.

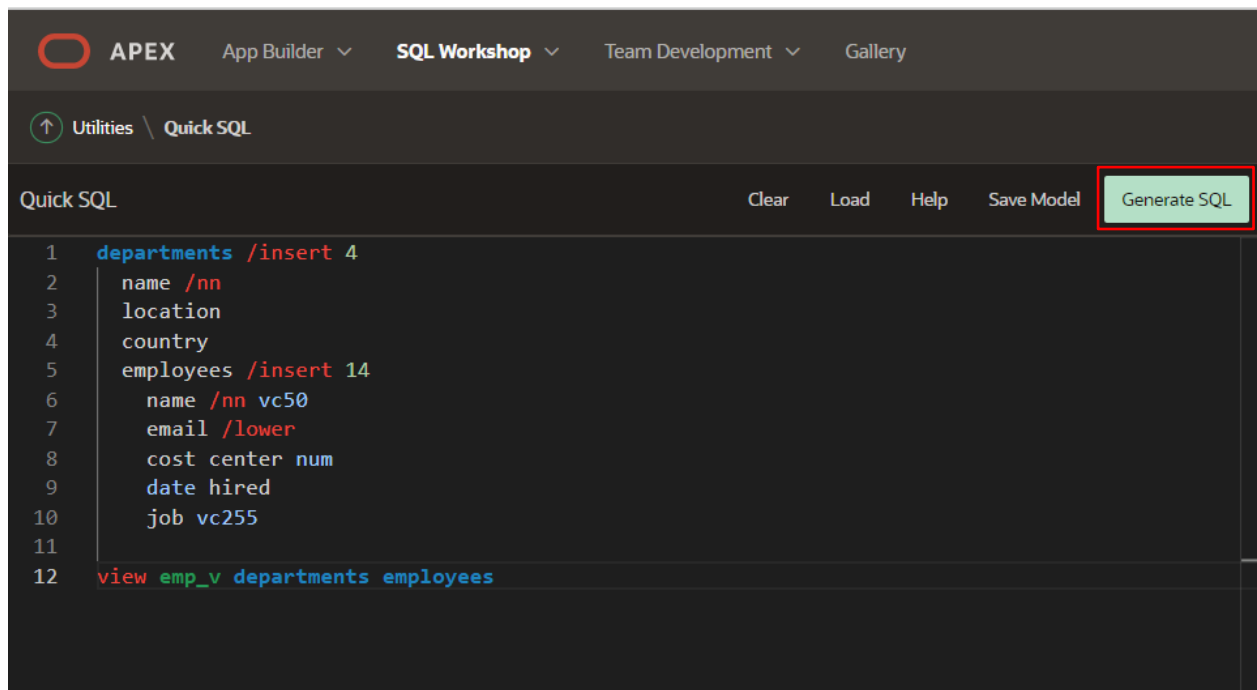
```
departments /insert 4
  name /nn
  location
  country
employees /insert 14
  name /nn vc50
  email /lower
  cost center num
  date hired
  job vc255

view emp_v departments employees
```

Note that the Indentation from the code you copied should be similar to the one we have in the following screenshot.

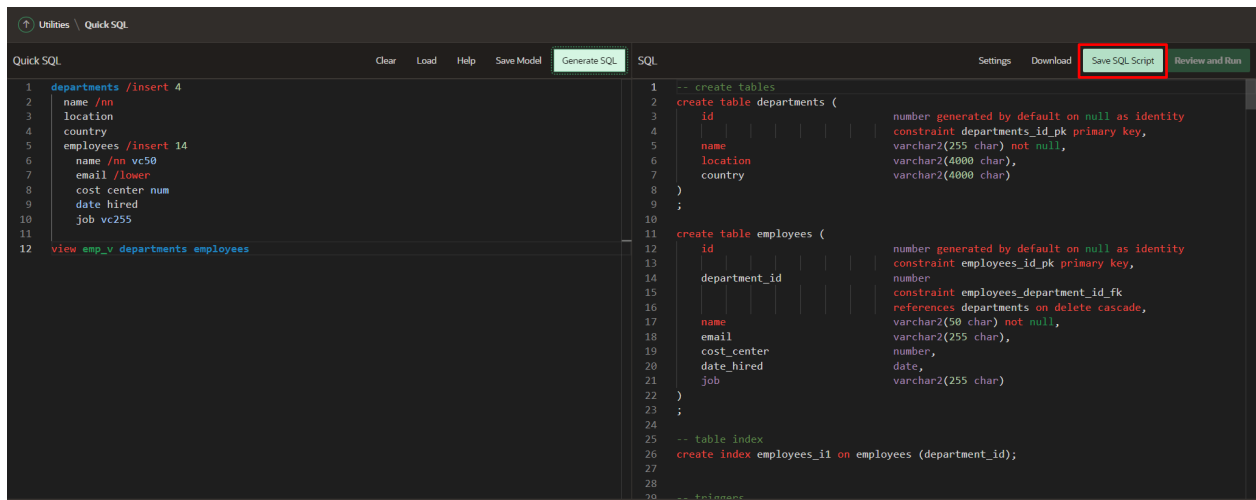
- Review the shorthand code. How many tables will be created and how are they related? What types of columns and constraints will be created for the tables? Note that the **Help** button provides details on the shorthand syntax along with several examples of its usage.

When ready, in the Quick SQL (left pane) toolbar, click **Generate SQL**.

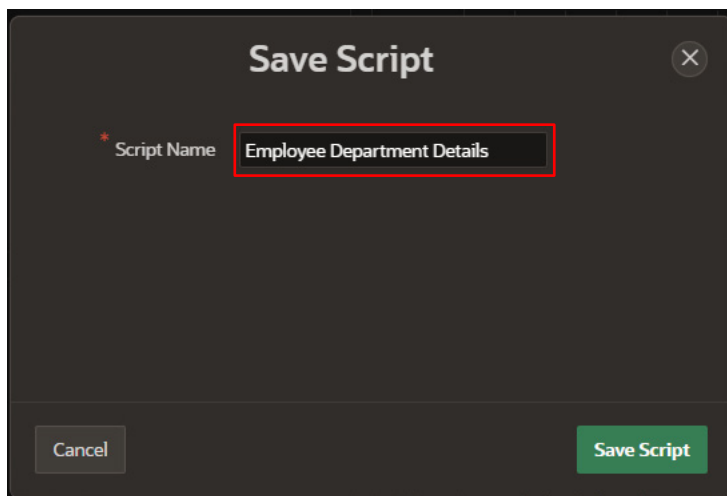


- Review the SQL that is generated in the right-hand pane, and then click **Settings** to make some changes to the resulting code.

It is important to save the SQL Shorthand as a script so that it can be run in your database. In the SQL (right pane) toolbar, click **Save SQL Script**.

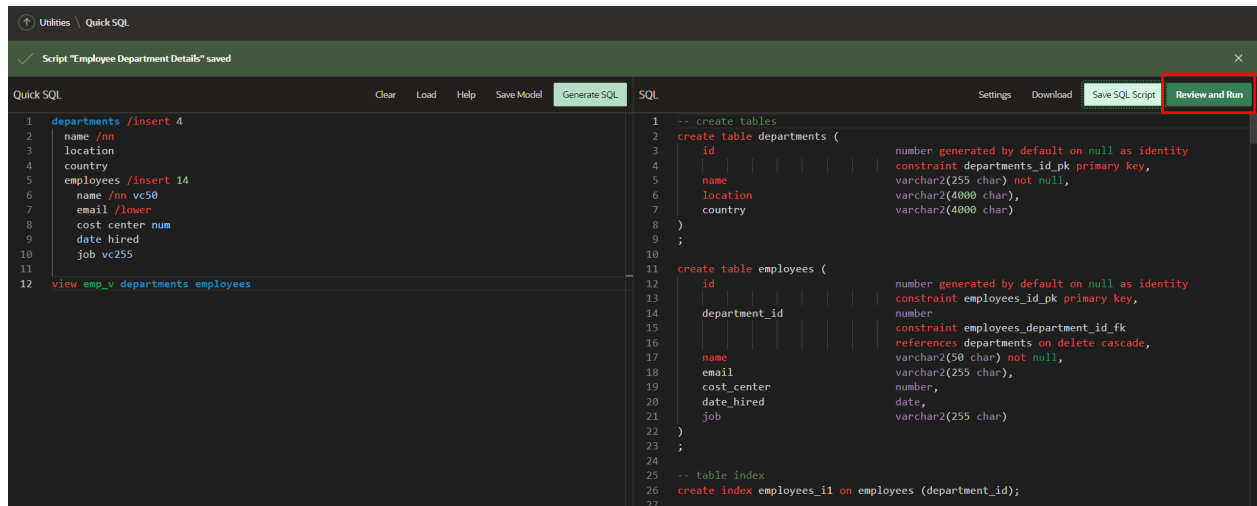


- In the Save Script dialog, for Script Name enter **Employee Department Details**, and click **Save Script**.

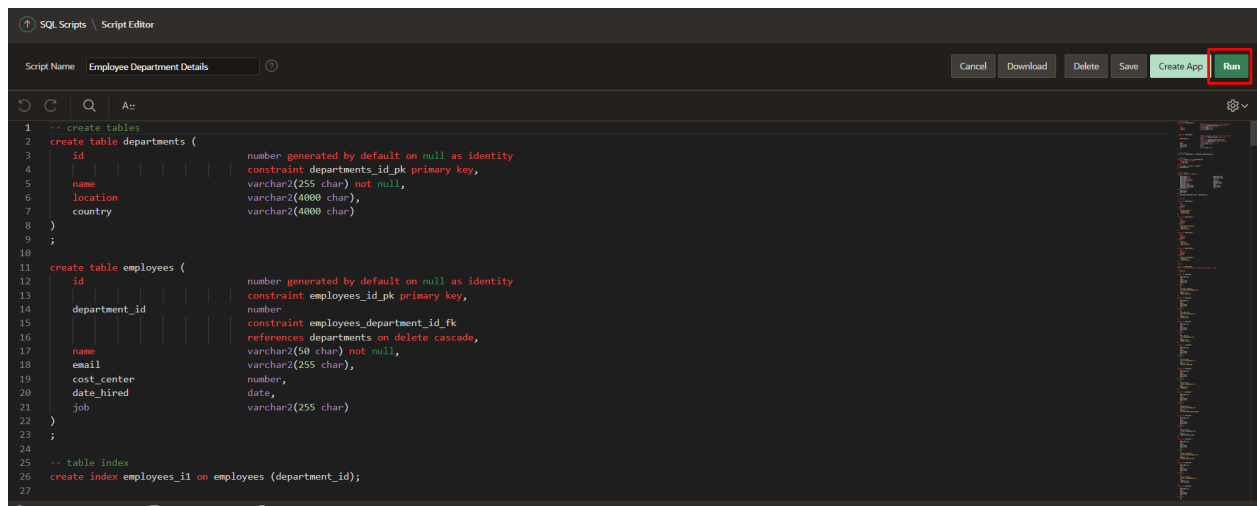


- Now that you saved the script you can run it to create the specified database objects.

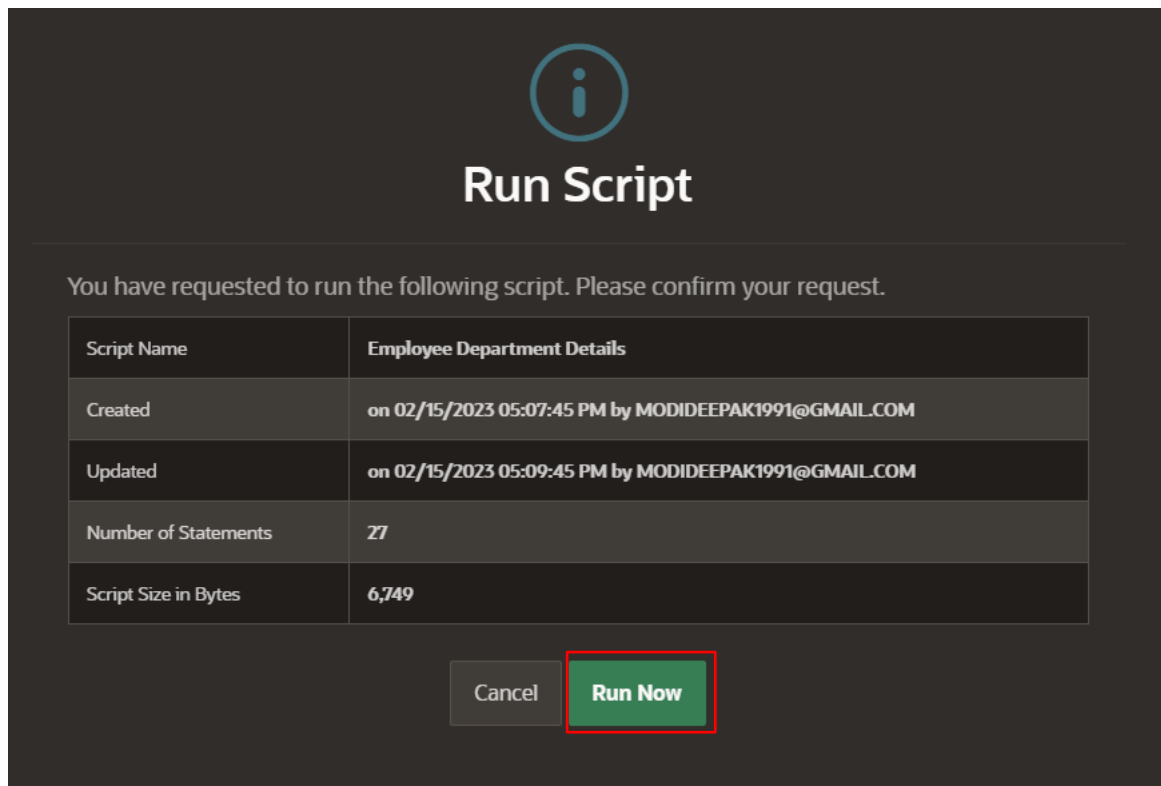
In the SQL (right pane) toolbar, click **Review and Run**.



- In the Script Editor page toolbar, click **Run**.



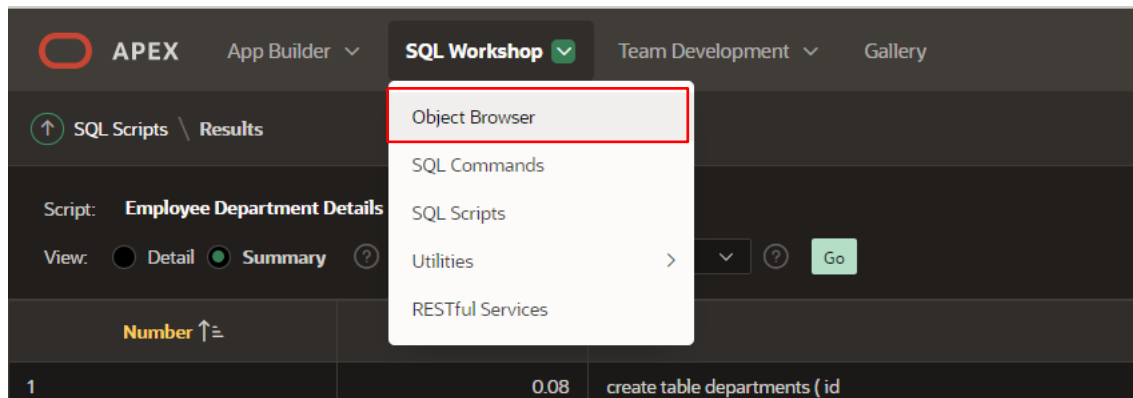
8. On the Run Script page, click **Run Now**.



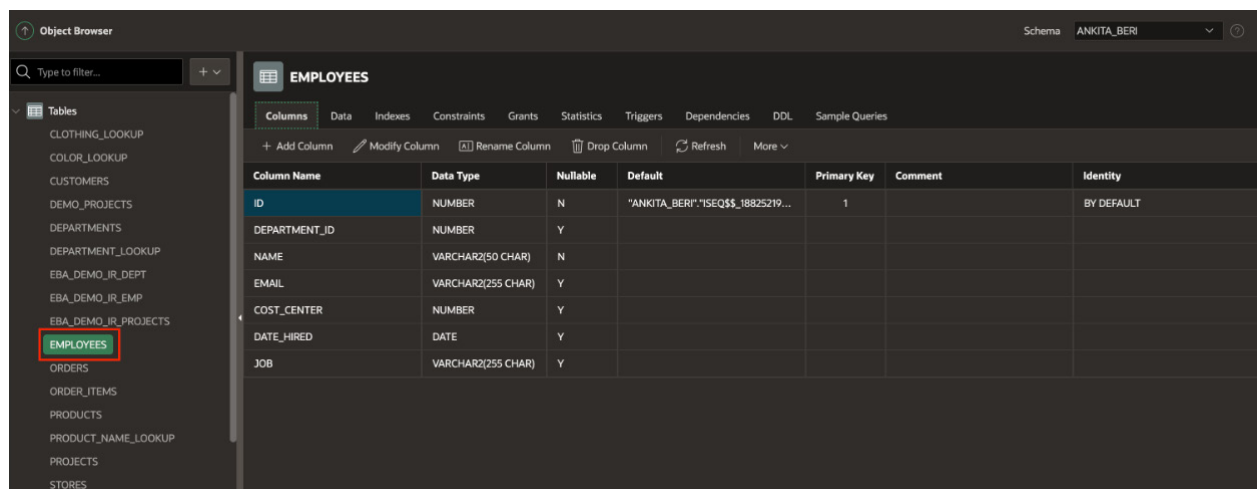
9. The Results page shows the results of running the script. Scroll to the bottom to see a summary. You should not see any errors.

SQL Scripts \ Results				
Script: Employee Department Details		Status: Complete	Create App Edit Script	
View: Detail Summary		Rows: 15	Go	
Number ↑	Elapsed	Statement	Feedback	Rows
1	0.08	create table departments (id	Table created.	0
2	0.03	create table employees (id	Table created.	0
3	0.02	create index employees_i1 on employees (department_id)	Index created.	0
4	0.04	create or replace trigger employees_biu before insert or	Trigger created.	0
5	0.06	create or replace view emp_v as select departments.id	View created.	0
6	0.02	insert into departments (id, name, location,	1 row(s) inserted.	1
7	0.01	insert into departments (id, name, location,	1 row(s) inserted.	1
8	0.00	insert into departments (id, name, location,	1 row(s) inserted.	1
9	0.00	insert into departments (id, name, location,	1 row(s) inserted.	1
10	0.00	commit	Statement processed.	0
11	0.07	alter table departments modify id generated always as identi	Table altered.	0
12	0.08	insert into employees (id, department_id, name,	1 row(s) inserted.	1
13	0.00	insert into employees (id, department_id, name,	1 row(s) inserted.	1
14	0.00	insert into employees (id, department_id, name,	1 row(s) inserted.	1
15	0.00	insert into employees (id, department_id, name,	1 row(s) inserted.	1

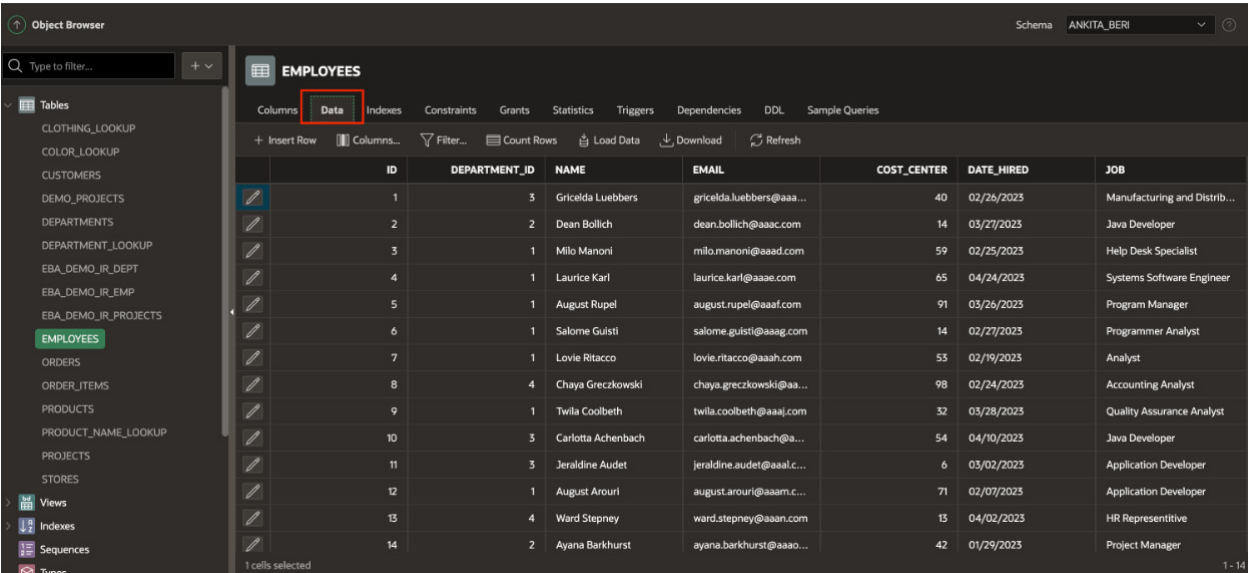
10. To view the database objects that were created, click the arrow 'v' in the **SQL Workshop** tab and then select **Object Browser**.



11. Select the **EMPLOYEES** table and review its columns.



12. Click the **Data** tab to see the data that was loaded.



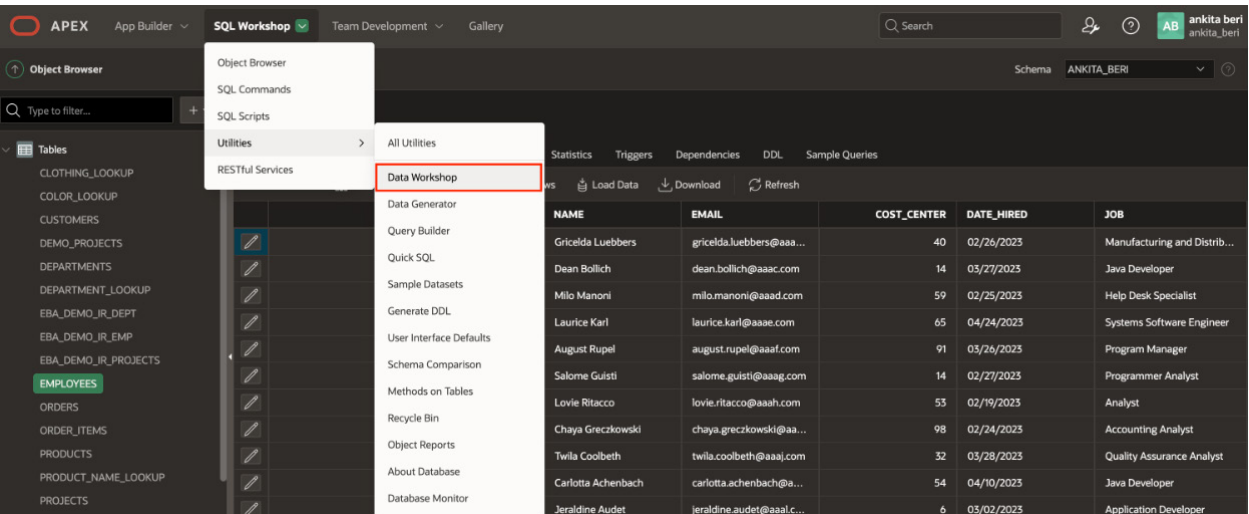
The screenshot shows the Oracle SQL Developer Object Browser. The left pane displays a tree of database objects, with 'EMPLOYEES' selected under the 'Tables' category. The right pane shows the 'Data' tab for the 'EMPLOYEES' table. The table has columns: ID, DEPARTMENT_ID, NAME, EMAIL, COST_CENTER, DATE_HIRED, and JOB. The data is displayed in a grid with 14 rows. The 'Data' tab is highlighted with a red box.

ID	DEPARTMENT_ID	NAME	EMAIL	COST_CENTER	DATE_HIRED	JOB
1	3	Gricelda Luebbbers	gricelda.luebbbers@aaa...	40	02/26/2023	Manufacturing and Distrib...
2	2	Dean Bollich	dean.bollich@aaac.com	14	03/27/2023	Java Developer
3	1	Milo Manoni	milo.manoni@aaad.com	59	02/25/2023	Help Desk Specialist
4	1	Laurice Karl	laurice.karl@aaae.com	65	04/24/2023	Systems Software Engineer
5	1	August Rupel	august.rupel@aaaf.com	91	03/26/2023	Program Manager
6	1	Salome Gulsti	salome.gulsti@aaag.com	14	02/27/2023	Programmer Analyst
7	1	Lovie Ritacco	lovie.ritacco@aaah.com	53	02/19/2023	Analyst
8	4	Chaya Greczkowski	chaya.greczkowski@aa...	98	02/24/2023	Accounting Analyst
9	1	Twila Coolbeth	twila.coolbeth@aaaj.com	32	03/28/2023	Quality Assurance Analyst
10	3	Carlotta Achenbach	carlotta.achenbach@a...	54	04/10/2023	Java Developer
11	3	Jeraldine Audet	geraldine.audet@aaal.c...	6	03/02/2023	Application Developer
12	1	August Arouri	august.arouri@aaam.c...	71	02/07/2023	Application Developer
13	4	Ward Stepney	ward.stepney@aaan.com	13	04/02/2023	HR Representative
14	2	Ayana Barkhurst	ayana.barkhurst@aaao...	42	01/29/2023	Project Manager

Load Data Using the Data Workshop Utility

In this step, you will learn how to create a table and Load data using the Data Workshop utility.

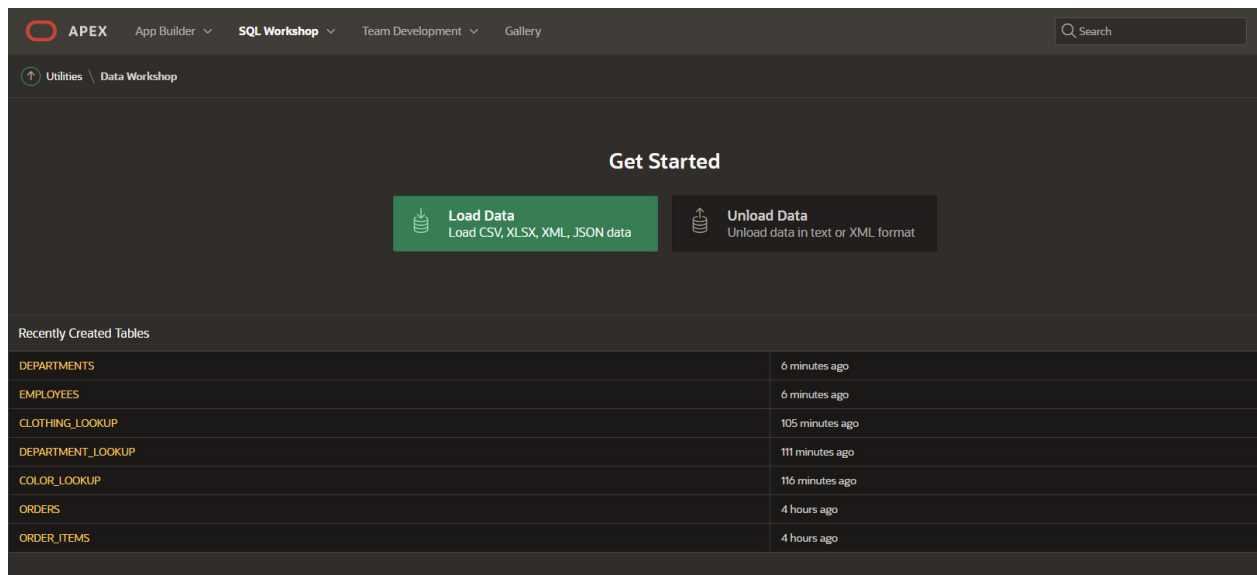
1. On the Workspace home page, click the 'v' next to **SQL Workshop** tab, click **Utilities**, and then select **Data Workshop**.



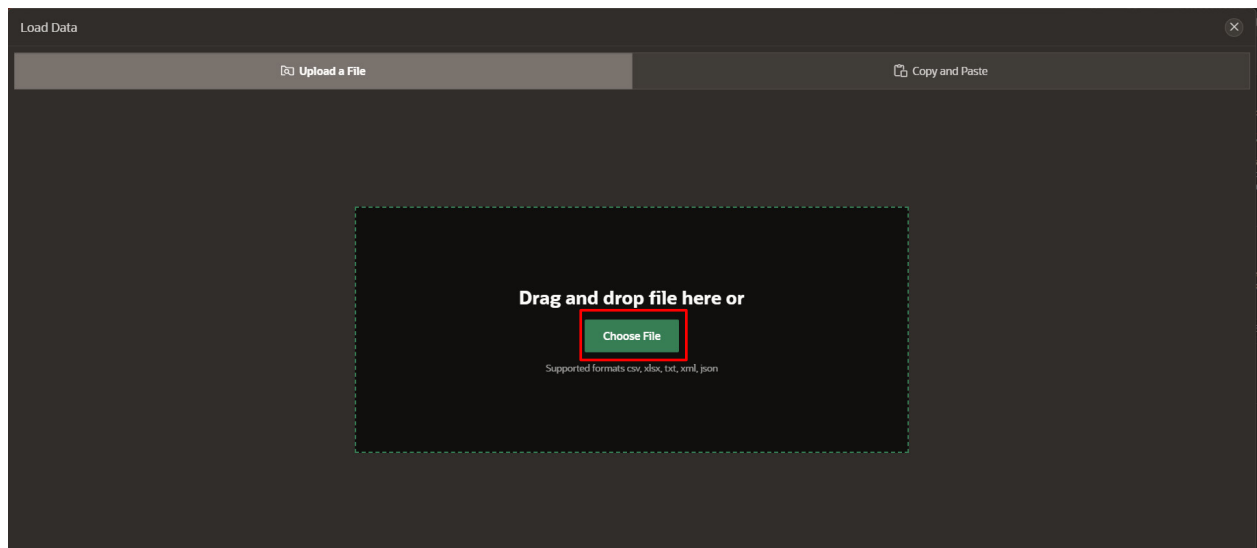
The screenshot shows the Oracle SQL Developer Workspace home page. The 'SQL Workshop' tab is selected, and the 'Utilities' menu is open. The 'Data Workshop' option is highlighted with a red box. The background shows the 'EMPLOYEES' table data.

NAME	EMAIL	COST_CENTER	DATE_HIRED	JOB
Gricelda Luebbbers	gricelda.luebbbers@aaa...	40	02/26/2023	Manufacturing and Distrib...
Dean Bollich	dean.bollich@aaac.com	14	03/27/2023	Java Developer
Milo Manoni	milo.manoni@aaad.com	59	02/25/2023	Help Desk Specialist
Laurice Karl	laurice.karl@aaae.com	65	04/24/2023	Systems Software Engineer
August Rupel	august.rupel@aaaf.com	91	03/26/2023	Program Manager
Salome Gulsti	salome.gulsti@aaag.com	14	02/27/2023	Programmer Analyst
Lovie Ritacco	lovie.ritacco@aaah.com	53	02/19/2023	Analyst
Chaya Greczkowski	chaya.greczkowski@aa...	98	02/24/2023	Accounting Analyst
Twila Coolbeth	twila.coolbeth@aaaj.com	32	03/28/2023	Quality Assurance Analyst
Carlotta Achenbach	carlotta.achenbach@a...	54	04/10/2023	Java Developer
Jeraldine Audet	geraldine.audet@aaal.c...	6	03/02/2023	Application Developer

- Under Get Started, click **Load Data**. The Load Data Wizard appears.



- Under Upload a File, click **Choose File**.



- Download [projects.csv](#) and select the file.

Lab 1 - Getting Started with Oracle APEX on ...	2/15/2023 5:14 PM	Microsoft Word Doc...	3,721 KB
Lab 2 - Use SQL Workshop.docx	2/15/2023 9:27 PM	Microsoft Word Doc...	3,665 KB
Lab 3 - Creating a Database Application.docx	2/14/2023 5:33 PM	Microsoft Word Doc...	220 KB
projects.csv	2/15/2023 10:48 PM	Microsoft Excel Com...	7 KB
Template.docx	2/14/2023 2:55 PM	Microsoft Word Doc...	251 KB

Note: If the uploaded XLSX file contains multiple worksheets, the Load Data wizard picks the first sheet by default. To load another sheet, pick it from the Select Sheet select list. The First lines contains headers checkbox works similar to CSV files. The same is true for the Configure dialog, the behavior is the same as it is for CSV files. Uploading XLSX files is limited to 20 MB for each file.

- Under **Load Data**, edit the Table Name field and provide an appropriate name. In this practice, you will set the Table Name as **Projects**. Then click **Load Data**.

Load Data

Where do you want to load this data?

Load To: **New Table** Existing Table

Table Owner: WKSP_APEXHANDSONLAB22

Table Name: **PROJECTS**

Please select the columns to load. [Configure](#)

Primary Keys: SYS_GUID **Identity Column**

☒ Use Column Data Types

Settings

Column Headers: ☒ First line contains headers

Column Delimiter: , | # tab

Enclosed By: None " ' .

File Encoding: Western European ISO-8859-1

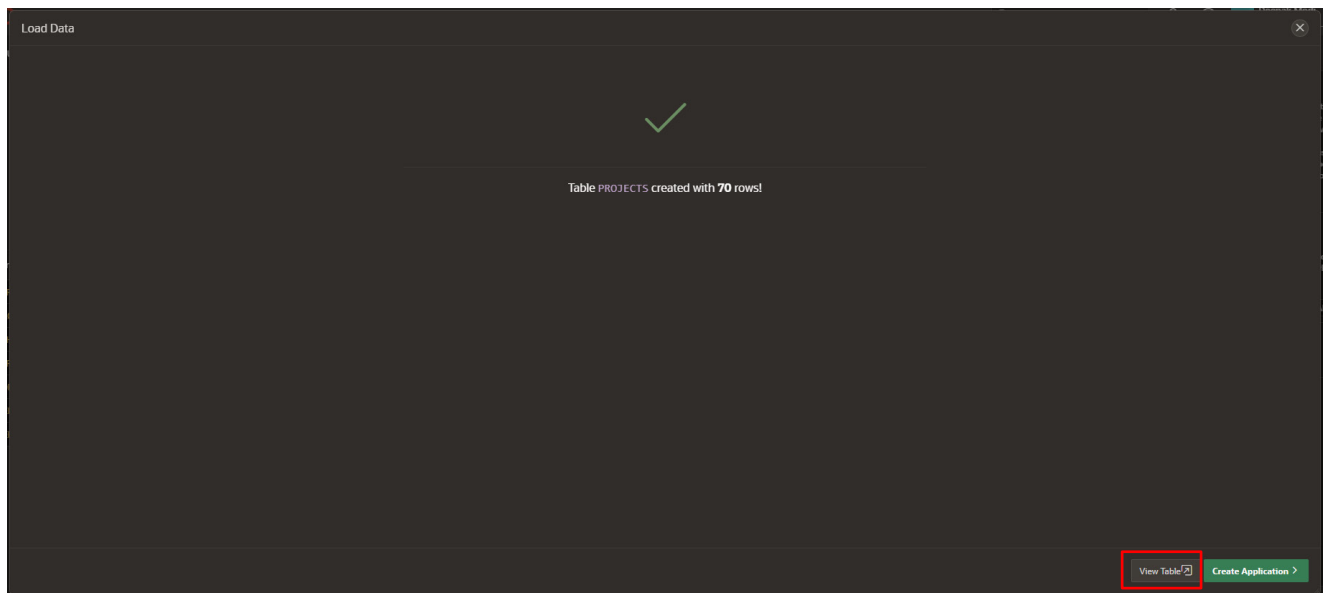
Preview

Parsed first 71 rows to sample the column types. The preview below only displays the first 8 columns and 5 rows. Only up to 500 characters of column content are shown. To view the full preview and configure data load settings, please click Preview button. [Preview](#)

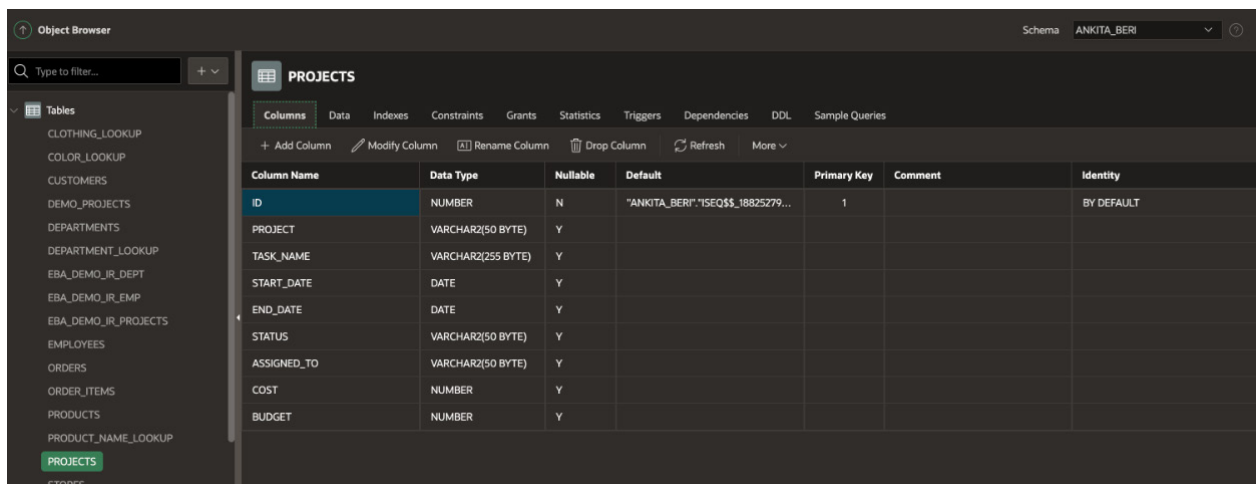
	PROJECT	TASK_NAME	START_DATE	END_DATE	STATUS	ASSIGNED_TO	COST	BUDGET
1	ACME Web Configuration	Identify server requirements	11/19/2021	12/04/21	Closed	John Watson	300	300
2	Maintain Support Systems	HR software upgrades	12/04/21	01/01/22	On-Hold	Pam King	8000	7000

[Cancel](#) [Load Data](#)

- Once Data Loading is finished, a success message is displayed. If some rows cannot be loaded to the target table, they will be stored to the error table and error table is displayed. Then click **View Table**.



- Select the **PROJECTS** table and review its columns.



- Click the **Data** tab to see the data that was loaded.

The screenshot shows the Oracle SQL Developer interface. On the left is the 'Object Browser' pane with a search bar and a tree view of database objects. The 'PROJECTS' table is selected and highlighted in green. The main pane displays the 'Data' tab for the 'PROJECTS' table. Above the table grid are tabs for 'Columns', 'Data', 'Indexes', 'Constraints', 'Grants', 'Statistics', 'Triggers', 'Dependencies', 'DDL', and 'Sample Queries'. The 'Data' tab is active, showing a toolbar with '+ Insert Row', 'Columns...', 'Filter...', 'Count Rows', 'Load Data', 'Download', and 'Refresh'. The table grid has 10 columns: ID, PROJECT, TASK_NAME, START_DATE, END_DATE, STATUS, ASSIGNED_TO, COST, and BUDGET. It contains 14 rows of data.

	ID	PROJECT	TASK_NAME	START_DATE	END_DATE	STATUS	ASSIGNED_TO	COST	BUDGET
	1	ACME Web Con...	Identify server r...	11/19/2021	12/06/2021	Closed	John Watson	300	300
	2	Maintain Suppor...	HR software upg...	12/04/2021	01/01/2022	On-Hold	Pam King	8000	7000
	3	Maintain Suppor...	Apply Billing Sys...	01/07/2022	01/29/2022	On-Hold	Russ Sanders	9500	7000
	4	ACME Web Con...	Determine Web I...	12/16/2021	12/17/2021	Closed	James Cassidy	100	100
	5	ACME Web Con...	Specify security ...	01/02/2022	01/04/2022	Closed	John Watson	200	300
	6	ACME Web Con...	Select servers fo...	11/24/2021	11/26/2021	Closed	James Cassidy	200	600
	7	Email Integration	Complete plan	11/17/2021	12/15/2021	Closed	Mark Nile	3000	1500
	8	ACME Web Con...	Configure Works...	12/20/2021	01/04/2022	Closed	John Watson	200	100
	9	ACME Web Con...	Create pilot wor...	01/15/2022	02/08/2022	Closed	John Watson	100	100
	10	ACME Web Con...	Run installation	11/22/2021	12/12/2021	Closed	James Cassidy	100	100
	11	Bug Tracker	Implement bug t...	01/05/2022	01/10/2022	Closed	Myra Sutcliff	100	100
	12	Bug Tracker	Review automat...	11/25/2021	11/27/2021	On-Hold	Myra Sutcliff	2750	1500
	13	Train Developers	Publish develop...	12/03/2021	12/13/2021	On-Hold	John Watson	1000	2000
	14	Train Developers	Publish links to s...	12/28/2021	01/03/2022	Closed	John Watson	100	100

This completes the practice. At this point, you know how to use Quick SQL to generate database objects using a SQL shorthand. You also know how to use Data Workshop utility to load data.

