

ASSESSMENT-I
PL/SQL – Restaurant Application

Sudharsana Inbakumar

Date: 04-06-2025

Creation of New Connection

Step 1: Creating the new User (Schema) in SQL*Plus (Backend CMD)

1.1 Connecting to FREEPDB1 as system:

```
SQL*Plus: Release 23.0.0.0.0 - Production on Wed Jun 4 09:14:43 2025
Version 23.8.0.25.04

Copyright (c) 1982, 2025, Oracle. All rights reserved.

Enter user-name: system
Enter password:
Last Successful login time: Mon Jun 02 2025 22:01:59 +05:30

Connected to:
Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
Version 23.8.0.25.04

SQL> |
```

1.2 Creating the RESTAURANT_APP user and granting privileges:

```
SQL> ALTER SESSION SET CONTAINER = FREEPDB1;
Session altered.

SQL> SHOW CON_NAME
CON_NAME
-----
FREEPDB1
SQL> CREATE USER RESTAURANT_APP IDENTIFIED BY res123
  2 ;
User created.
```

```
SQL> GRANT CONNECT, RESOURCE TO RESTAURANT_APP;
Grant succeeded.

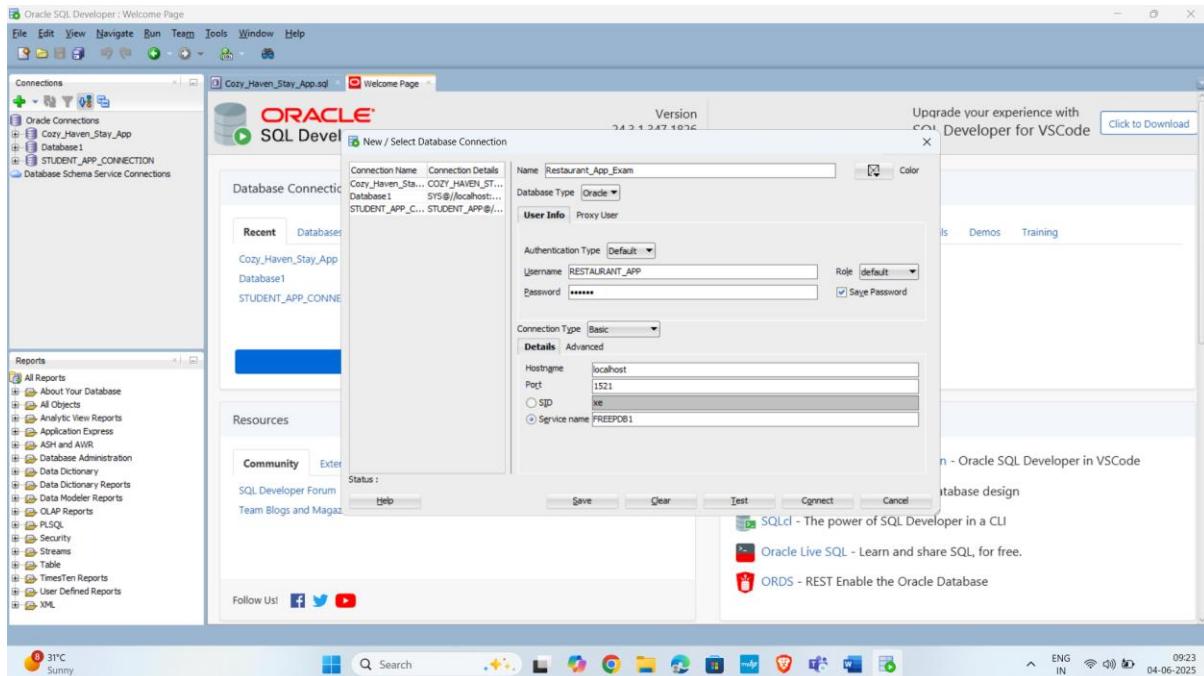
SQL> GRANT CREATE SESSION TO RESTAURANT_APP;
Grant succeeded.

SQL> GRANT CREATE TABLE TO RESTAURANT_APP;
Grant succeeded.

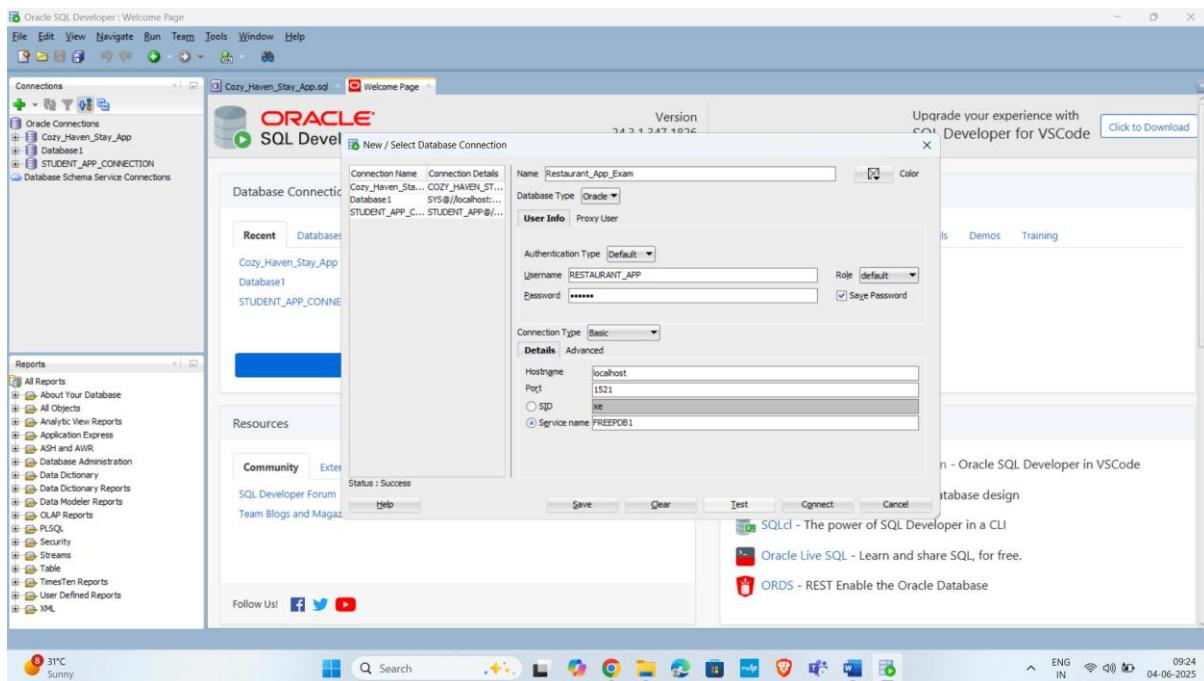
SQL> GRANT UNLIMITED TABLESPACE TO RESTAURANT_APP;
Grant succeeded.

SQL> |
```

Step 2: Creating the New Connection in Oracle SQL Developer

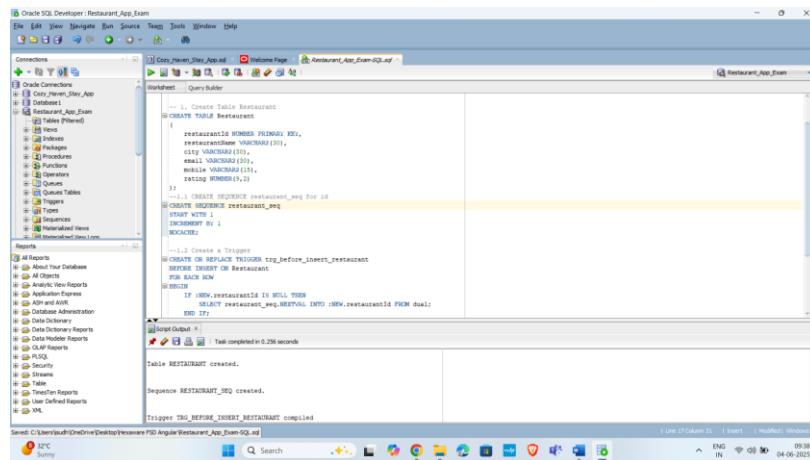


Successfully created a new connection and verified using Test



Creation of Tables:

1.Restaurant



The screenshot shows the Oracle SQL Developer interface with the connection 'Restaurant_App_Exam' selected. In the 'Worksheet' tab, a script is being run to create the 'RESTAURANT' table and a sequence 'RESTAURANT_SEQ'. The script also includes a trigger 'TRG_BEFORE_INSERT_RESTAURANT' that inserts a new record into the 'RESTAURANT_BACKUP' table before inserting into the main 'RESTAURANT' table.

```
-->1 Create Table Restaurant
CREATE TABLE RESTAURANT
(
    restaurantId NUMBER PRIMARY KEY,
    restaurantname VARCHAR2(10),
    city VARCHAR2(30),
    email VARCHAR2(30),
    mobile VARCHAR2(15),
    rating NUMBER(2),
    address VARCHAR2(50)
);
-->2 Create Sequence restaurant_seq for id
CREATE SEQUENCE RESTAURANT_SEQ
START WITH 1
INCREMENT BY 1
NOCACHE;
-->3 Create a Trigger
CREATE OR REPLACE TRIGGER trg_before_insert_restaurant
BEFORE INSERT ON Restaurant
FOR EACH ROW
BEGIN
    IF INSERTING.restaurantId IS NULL THEN
        SELECT restaurant_seq.NEXTVAL INTO :NEW.restaurantId FROM dual;
    END IF;

```

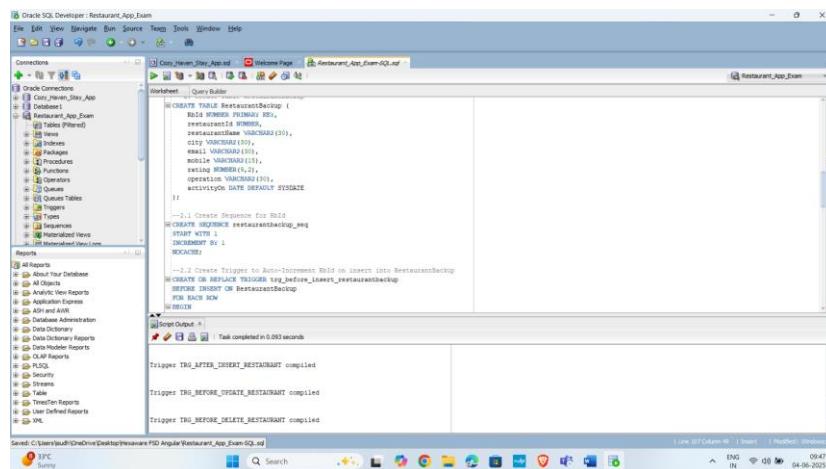
Script Output: Task completed in 0.296 seconds

Table RESTAURANT created.

Sequence RESTAURANT_SEQ created.

Triggers TRG_BEFORE_INSERT_RESTAURANT compiled

2.Restaurant Backup



The screenshot shows the Oracle SQL Developer interface with the connection 'Restaurant_App_Exam' selected. A script is being run to create the 'RESTAURANTBACKUP' table and three triggers: 'TRG_AFTER_INSERT_RESTAURANT', 'TRG_BEFORE_UPDATE_RESTAURANT', and 'TRG_BEFORE_DELETE_RESTAURANT'. The 'TRG_AFTER_INSERT_RESTAURANT' trigger inserts a new record into the 'RESTAURANTBACKUP' table after each insert into the 'RESTAURANT' table.

```
-->1 Create Table RestaurantBackup
CREATE TABLE RESTAURANTBACKUP
(
    restaurantId NUMBER PRIMKEY,
    restaurantname VARCHAR2(10),
    city VARCHAR2(30),
    email VARCHAR2(30),
    mobile VARCHAR2(15),
    rating NUMBER(2),
    address VARCHAR2(50),
    activitydate DATE DEFAULT SYSDATE
);
-->2 Create Sequence for RESTAURANTBACKUP
CREATE SEQUENCE RESTAURANTBACKUP_SEQ
START WITH 1
INCREMENT BY 1
NOCACHE;
-->3 Create Trigger to Auto-Increment Id on insert into RestaurantBackup
CREATE OR REPLACE TRIGGER trg_before_insert_restaurantbackup
BEFORE INSERT ON RESTAURANTBACKUP
FOR EACH ROW
BEGIN
    :NEW.id := RESTAURANTBACKUP_SEQ.NEXTVAL;

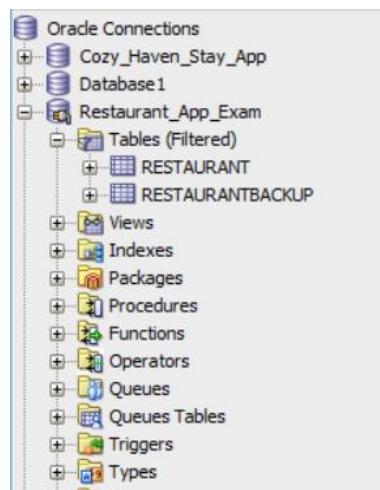
```

Script Output: Task completed in 0.003 seconds

Triggers TRG_AFTER_INSERT_RESTAURANT compiled

Triggers TRG_BEFORE_UPDATE_RESTAURANT compiled

Triggers TRG_BEFORE_DELETE_RESTAURANT compiled



Insertion Of Samples:

1.Restaurant

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar shows a connection to 'Cozy_Haven_Stay_App'. The 'Worksheet' tab contains a SQL script for creating a trigger and inserting sample data into the 'Restaurant' table.

```
CREATE OR REPLACE TRIGGER trg_before_insert_restaurant
BEFORE INSERT ON Restaurant
FOR EACH ROW
BEGIN
    IF :NEW.restaurantId IS NULL THEN
        SELECT restaurant_seq.NEXTVAL INTO :NEW.restaurantId FROM dual;
    END IF;
END;

-- Insertion Of Sample Records for Restaurant
INSERT ALL
INTO Restaurant (restaurantName, city, email, mobile, rating) VALUES ('Tasty Treats', 'Chennai', 'treats@food.com', '9123456789', 4.3)
INTO Restaurant (restaurantName, city, email, mobile, rating) VALUES ('Spice Hub', 'Madurai', 'spicehub@food.com', '9012345678', 4.6)
INTO Restaurant (restaurantName, city, email, mobile, rating) VALUES ('Curry Leaves', 'Coimbatore', 'curry@leaves.com', '9876543210', 4.2)
INTO Restaurant (restaurantName, city, email, mobile, rating) VALUES ('Grill House', 'Trichy', 'grill@house.com', '9998877654', 4.0)
INTO Restaurant (restaurantName, city, email, mobile, rating) VALUES ('Veg Delight', 'Salem', 'veg@delight.com', '9887766554', 4.1)

SELECT * FROM dual;
COMMIT;
```

The 'Script Output' pane shows the results of the execution:

```
Trigger TRG_BEFORE_DELETE_RESTAURANT compiled
5 rows inserted.
Commit complete.
```

The screenshot shows the Oracle SQL Developer interface with the 'RESTAURANT' table selected. The 'Data' tab displays the following data:

RESTAURANTID	RESTAURANTNAME	CITY	EMAIL	MOBILE	RATING
1	Tasty Treats	Chennai	treats@food.com	9123456789	4.3
2	Spice Hub	Madurai	spicehub@food.com	9012345678	4.6
3	Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2
4	Grill House	Trichy	grill@house.com	9998877654	4.0
5	Veg Delight	Salem	veg@delight.com	9887766554	4.1

2.Restaurant Backup

```

-- Insertion Of Sample Records to RestaurantBackup
INSERT INTO RestaurantBackup (
    restaurantId, restaurantName, city, email, mobile, rating, operation
) VALUES (
    :OLD.restaurantId, :OLD.restaurantName, :OLD.city, :OLD.email, :OLD.mobile, :OLD.rating, 'DELETE'
);
END;
/
-- Insertion Of Sample Records to RestaurantBackup
INSERT ALL
    INTO RestaurantBackup (restaurantId, restaurantName, city, email, mobile, rating, operation)
        VALUES (1, 'Tasty Treats', 'Chennai', 'treats@food.com', '9123456789', 4.3, 'INSERT')
    INTO RestaurantBackup (restaurantId, restaurantName, city, email, mobile, rating, operation)
        VALUES (2, 'Spice Hub', 'Madurai', 'spicehub@food.com', '9012345678', 4.6, 'INSERT')
    INTO RestaurantBackup (restaurantId, restaurantName, city, email, mobile, rating, operation)
        VALUES (3, 'Curry Leaves', 'Coimbatore', 'curry@leaves.com', '9876543210', 4.2, 'INSERT')
    INTO RestaurantBackup (restaurantId, restaurantName, city, email, mobile, rating, operation)
        VALUES (4, 'Grill House', 'Trichy', 'grill@house.com', '9998776655', 4.8, 'INSERT')
    INTO RestaurantBackup (restaurantId, restaurantName, city, email, mobile, rating, operation)
        VALUES (5, 'Veg Delight', 'Salem', 'veg@delight.com', '9887766554', 4.1, 'INSERT')
SELECT * FROM dual;
COMMIT;

```

Script Output x Task completed in 0.054 seconds

Commit complete.

5 rows inserted.

Commit complete.

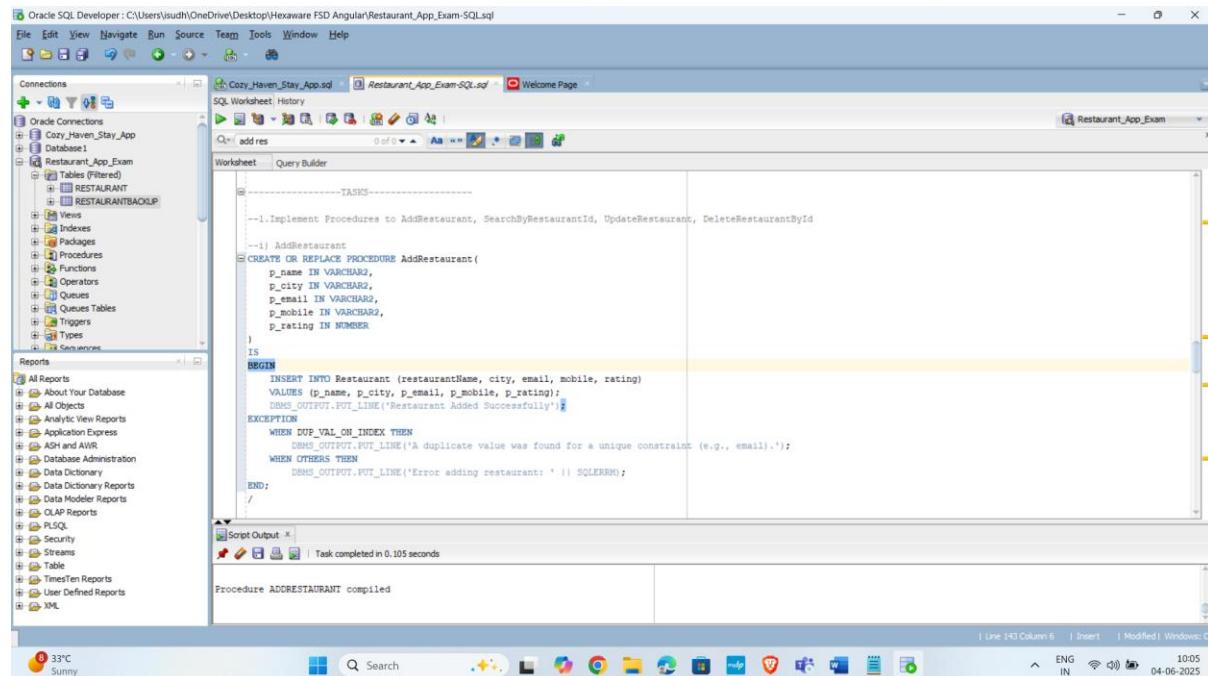
ID	RESTAURANTID	RESTAURANTNAME	CITY	EMAIL	MOBILE	RATING	OPERATION	ACTIVITYON
1	1	1Tasty Treats	Chennai	treats@food.com	9123456789	4.3	INSERT	04-06-25
2	2	2Spice Hub	Madurai	spicehub@food.com	9012345678	4.6	INSERT	04-06-25
3	3	3Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2	INSERT	04-06-25
4	4	4Grill House	Trichy	grill@house.com	9998776655	4.8	INSERT	04-06-25
5	5	5Veg Delight	Salem	veg@delight.com	9887766554	4.1	INSERT	04-06-25
6	6	1Tasty Treats	Chennai	treats@food.com	9123456789	4.3	INSERT	04-06-25
7	7	2Spice Hub	Madurai	spicehub@food.com	9012345678	4.6	INSERT	04-06-25
8	8	3Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2	INSERT	04-06-25
9	9	4Grill House	Trichy	grill@house.com	9998776655	4.8	INSERT	04-06-25
10	10	5Veg Delight	Salem	veg@delight.com	9887766554	4.1	INSERT	04-06-25

Tasks:

1. Implement Procedures to

AddRestaurant, SearchByRestaurantId, UpdateRestaurant, DeleteRestaurantById

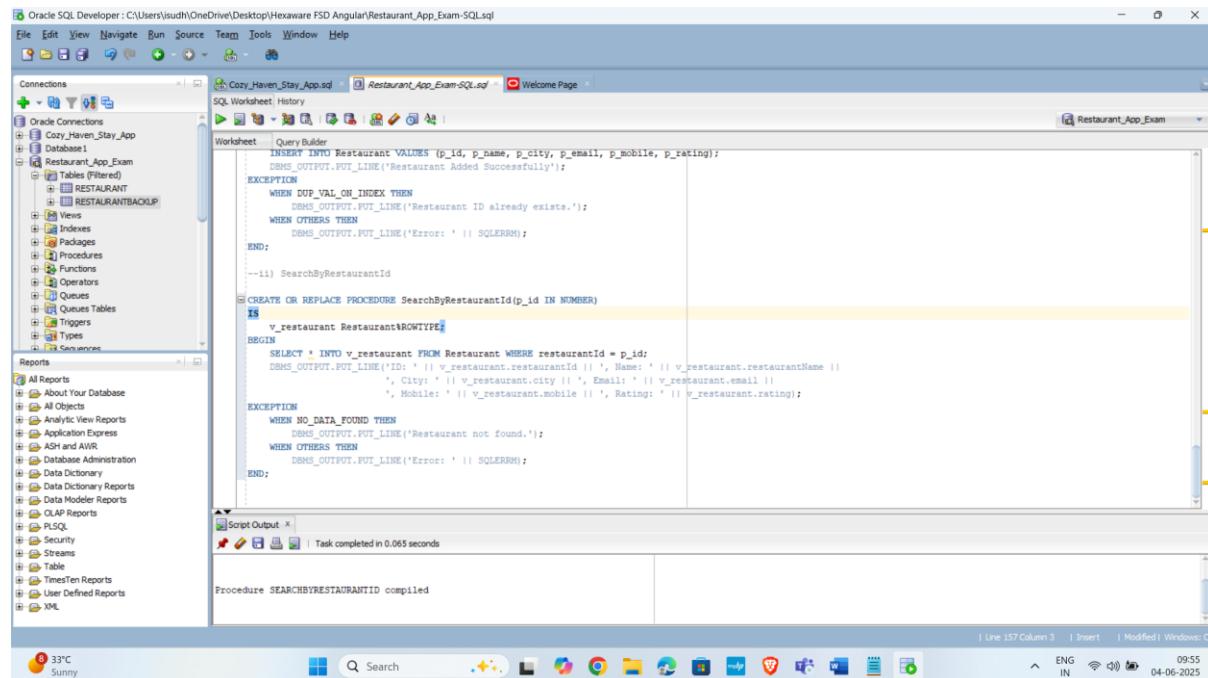
i) Add Restaurant



The screenshot shows the Oracle SQL Developer interface with the 'Restaurant_App_Exam' connection selected. The 'Script Output' window at the bottom indicates that the procedure was compiled successfully.

```
-->1. Implement Procedures to AddRestaurant, SearchByRestaurantId, UpdateRestaurant, DeleteRestaurantById
-->1) AddRestaurant
CREATE OR REPLACE PROCEDURE AddRestaurant(
    p_name IN VARCHAR2,
    p_city IN VARCHAR2,
    p_email IN VARCHAR2,
    p_mobile IN VARCHAR2,
    p_rating IN NUMBER
)
IS
BEGIN
    INSERT INTO Restaurant (restaurantName, city, email, mobile, rating)
    VALUES (p_name, p_city, p_email, p_mobile, p_rating);
    DBMS_OUTPUT.PUT_LINE('Restaurant Added Successfully');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('A duplicate value was found for a unique constraint (e.g., email).');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error adding restaurant: ' || SQLEERR);
END;
/
Procedure ADDERESTAURANT compiled
```

ii) SearchByRestaurantId



The screenshot shows the Oracle SQL Developer interface with the 'Restaurant_App_Exam' connection selected. The 'Script Output' window at the bottom indicates that the procedure was compiled successfully.

```
-->1) SearchByRestaurantId
CREATE OR REPLACE PROCEDURE SearchByRestaurantId(p_id IN NUMBER)
IS
    v_restaurant Restaurant%ROWTYPE;
BEGIN
    SELECT * INTO v_restaurant FROM Restaurant WHERE restaurantId = p_id;
    DBMS_OUTPUT.PUT_LINE('ID: ' || v_restaurant.restaurantId || ', Name: ' || v_restaurant.restaurantName ||
                         ', City: ' || v_restaurant.city || ', Email: ' || v_restaurant.email ||
                         ', Mobile: ' || v_restaurant.mobile || ', Rating: ' || v_restaurant.rating);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Restaurant not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLEERR);
END;
/
Procedure SEARCHBYRESTAURANTID compiled
```

iii) UpdateRestaurant

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar shows a connection to 'Cozy_Haven_Stay_App'. The 'Worksheet' tab contains the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE UpdateRestaurant(
    p_id IN NUMBER,
    p_name IN VARCHAR2,
    p_city IN VARCHAR2,
    p_email IN VARCHAR2,
    p_mobile IN VARCHAR2,
    p_rating IN NUMBER
)
IS
BEGIN
    UPDATE Restaurant
    SET restaurantName = p_name, city = p_city, email = p_email, mobile = p_mobile, rating = p_rating
    WHERE restaurantId = p_id;

    IF SQLROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Restaurant not found.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Restaurant updated.');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLEERRM);
END;
```

The 'Script Output' pane at the bottom shows the message: "Procedure UPDATERESTAURANT compiled". The status bar at the bottom right indicates "0956 04-06-2025".

iv) DeleteRestaurantById

The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar shows a connection to 'Cozy_Haven_Stay_App'. The 'Worksheet' tab contains the following PL/SQL code:

```
DBMS_OUTPUT.PUT_LINE('Restaurant not found.');
ELSE
    DBMS_OUTPUT.PUT_LINE('Restaurant updated.');
END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLEERRM);
END;

--iv)DeleteRestaurantById
CREATE OR REPLACE PROCEDURE DeleteRestaurantById(p_id IN NUMBER)
IS
BEGIN
    DELETE FROM Restaurant WHERE restaurantId = p_id;
    IF SQLROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Restaurant not found.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Restaurant deleted.');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLEERRM);
END;
```

The 'Script Output' pane at the bottom shows the message: "Procedure DELETERESTAURANTBYID compiled". The status bar at the bottom right indicates "0957 04-06-2025".

2. Write cursor to print all restaurant Details

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database connections and schema for 'Cozy_Haven_Stay_App'. The central workspace contains the following PL/SQL code:

```
DBMS_OUTPUT.PUT_LINE('Restaurant not found.');
ELSE
    DBMS_OUTPUT.PUT_LINE('Restaurant deleted.');
END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;

-- 2. Write cursor to print all restaurant Details

DECLARE
    CURSOR rest_cursor IS SELECT * FROM Restaurant;
    v_restaurant Restaurant%ROWTYPE;
BEGIN
    OPEN rest_cursor;
    LOOP
        FETCH rest_cursor INTO v_restaurant;
        EXIT WHEN rest_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Id: ' || v_restaurant.restaurantId || ', Name: ' || v_restaurant.restaurantName ||
                            ', City: ' || v_restaurant.city || ', Email: ' || v_restaurant.email ||
                            ', Mobile: ' || v_restaurant.mobile || ', Rating: ' || v_restaurant.rating);
    END LOOP;
    CLOSE rest_cursor;
END;
```

The script output window at the bottom shows the message: "PL/SQL procedure successfully completed."

3. Write a procedure with output parameters to searchRestaurantById

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database connections and schema for 'Cozy_Haven_Stay_App'. The central workspace contains the following PL/SQL code:

```
END;

---3. Write a procedure with output parameters to searchRestaurantById

CREATE OR REPLACE PROCEDURE searchRestaurantById(
    p_id IN NUMBER,
    p_name OUT VARCHAR2,
    p_city OUT VARCHAR2,
    p_email OUT VARCHAR2,
    p_mobile OUT VARCHAR2,
    p_rating OUT NUMBER
)
IS
BEGIN
    SELECT restaurantName, city, email, mobile, rating
    INTO p_name, p_city, p_email, p_mobile, p_rating
    FROM Restaurant
    WHERE restaurantId = p_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Restaurant not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

The script output window at the bottom shows the message: "Procedure SEARCHRESTAURANTBYID compiled".

4. Write 3 Triggers for Insert, Update and Delete as you need to store the records in backup table and write operation as "INSERT", "DELETE", "UPDATE" //done after table creation

1. If update old values to be stored in Backup Table
2. if delete old values to be stored in Backup Table
3. if insert new values to be stored in backup table

Trigger Name	Timing	Event	Captures	operation
trg_after_insert_restaurant	AFTER INSERT	Insert	NEW values	'INSERT'
trg_before_update_restaurant	BEFORE UPDATE	Update	OLD values	'UPDATE'
trg_before_delete_restaurant	BEFORE DELETE	Delete	OLD values	'DELETE'

1. trg_before_update_restaurant

```
--2.3.2 Before Update on Restaurant — Log old data before update
CREATE OR REPLACE TRIGGER trg_before_update_restaurant
BEFORE UPDATE ON Restaurant
FOR EACH ROW
BEGIN
    INSERT INTO RestaurantBackup (
        restaurantId, restaurantName, city, email, mobile, rating, operation
    ) VALUES (
        :OLD.restaurantId, :OLD.restaurantName, :OLD.city, :OLD.email, :OLD.mobile, :OLD.rating, 'UPDATE'
    );
END;
/
```

2. trg_before_delete_restaurant

```
--2.3.3 Before Delete on Restaurant — Log old data before delete
CREATE OR REPLACE TRIGGER trg_before_delete_restaurant
BEFORE DELETE ON Restaurant
FOR EACH ROW
BEGIN
    INSERT INTO RestaurantBackup (
        restaurantId, restaurantName, city, email, mobile, rating, operation
    ) VALUES (
        :OLD.restaurantId, :OLD.restaurantName, :OLD.city, :OLD.email, :OLD.mobile, :OLD.rating, 'DELETE'
    );
END;
/
```

3. trg_after_insert_restaurant

```
--2.3.1 After Insert on Restaurant — Log new inserted row
CREATE OR REPLACE TRIGGER trg_after_insert_restaurant
AFTER INSERT ON Restaurant
FOR EACH ROW
BEGIN
    INSERT INTO RestaurantBackup (
        restaurantId, restaurantName, city, email, mobile, rating, operation
    ) VALUES (
        :NEW.restaurantId, :NEW.restaurantName, :NEW.city, :NEW.email, :NEW.mobile, :NEW.rating, 'INSERT'
    );
END;
/
```

Sample Inputs and Corresponding Output

Inputs that I used :

Procedure	Sample Input
AddRestaurant	('Ocean Bites', 'Pondicherry', ' ocean@bites.com ', '9876543211', 4.5)
SearchById	1 (existing ID), 999 (non-existing ID)
UpdateRestaurant	(2, 'Spice Hub Deluxe', 'Madurai', ' spicehub@newmail.com ', '9012345678', 4.7)
DeleteRestaurantById	3 (existing ID), 999 (non-existing ID)
searchRestaurantById	1

1) AddRestaurant

```
EXEC AddRestaurant('Ocean Bites', 'Pondicherry', 'ocean@bites.com', '9876543211', 4.5);
```

```
Restaurant Added Successfully
```

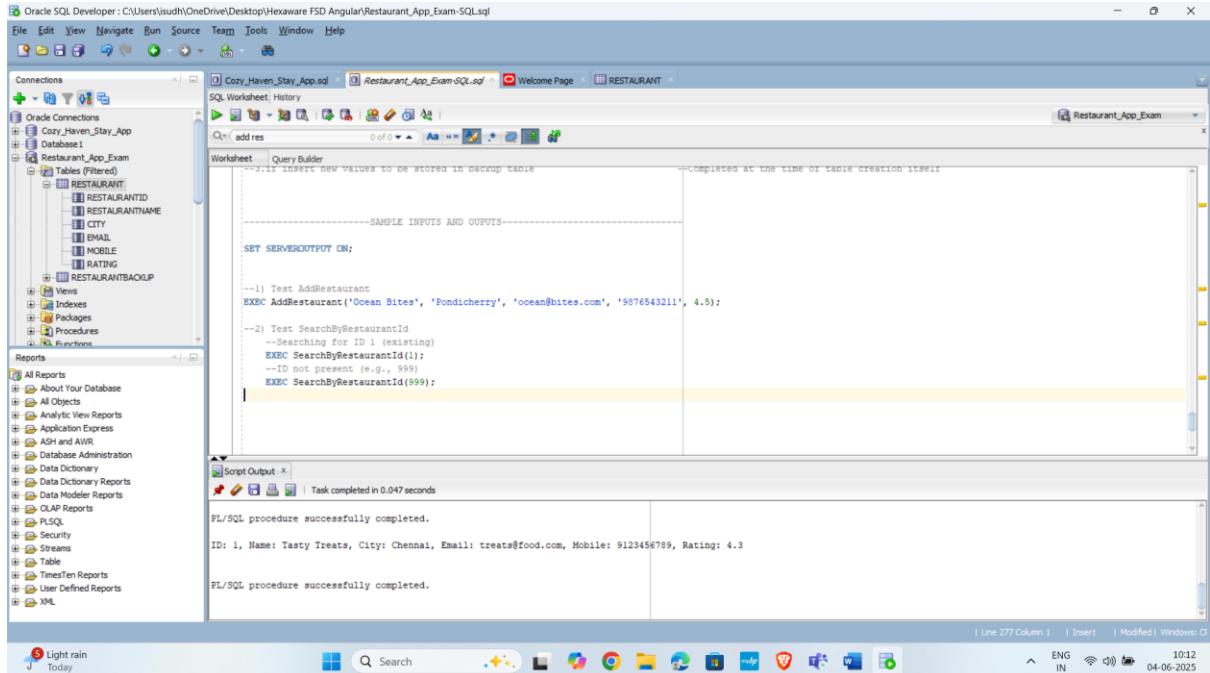
```
PL/SQL procedure successfully completed.
```

The screenshot shows the Oracle SQL Developer interface with the RESTAURANT table selected. The table has columns: RESTAURANTID, RESTAURANTNAME, CITY, EMAIL, MOBILE, and RATING. The data grid shows the following rows:

RESTAURANTID	RESTAURANTNAME	CITY	EMAIL	MOBILE	RATING
1	Ocean Bites	Pondicherry	ocean@bites.com	9876543211	4.5
2	Tasty Treats	Chennai	treats@food.com	9123456789	4.3
3	Spice Hub	Madurai	spicehub@food.com	9012345678	4.6
4	Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2
5	Grill House	Trichy	grill@house.com	9998877665	4.8
6	Veg Delight	Salem	veg@delight.com	9887766554	4.1

2) SearchByRestaurantId

2.1 Searching for ID 1 (existing)



The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database structure for the 'Cozy_Haven_Stay_App' schema, specifically the 'RESTAURANT' table which includes columns like RESTAURANTID, RESTAURANTNAME, CITY, EMAIL, MOBILE, and RATING. The central workspace contains a SQL Worksheet with the following PL/SQL code:

```
--1) Test AddRestaurant
EXEC AddRestaurant('Ocean Bites', 'Pondicherry', 'ocean@bites.com', '9876543211', 4.5);
--2) Test SearchByRestaurantId
EXEC SearchByRestaurantId(1);
--ID not present (e.g., 999)
EXEC SearchByRestaurantId(999);
```

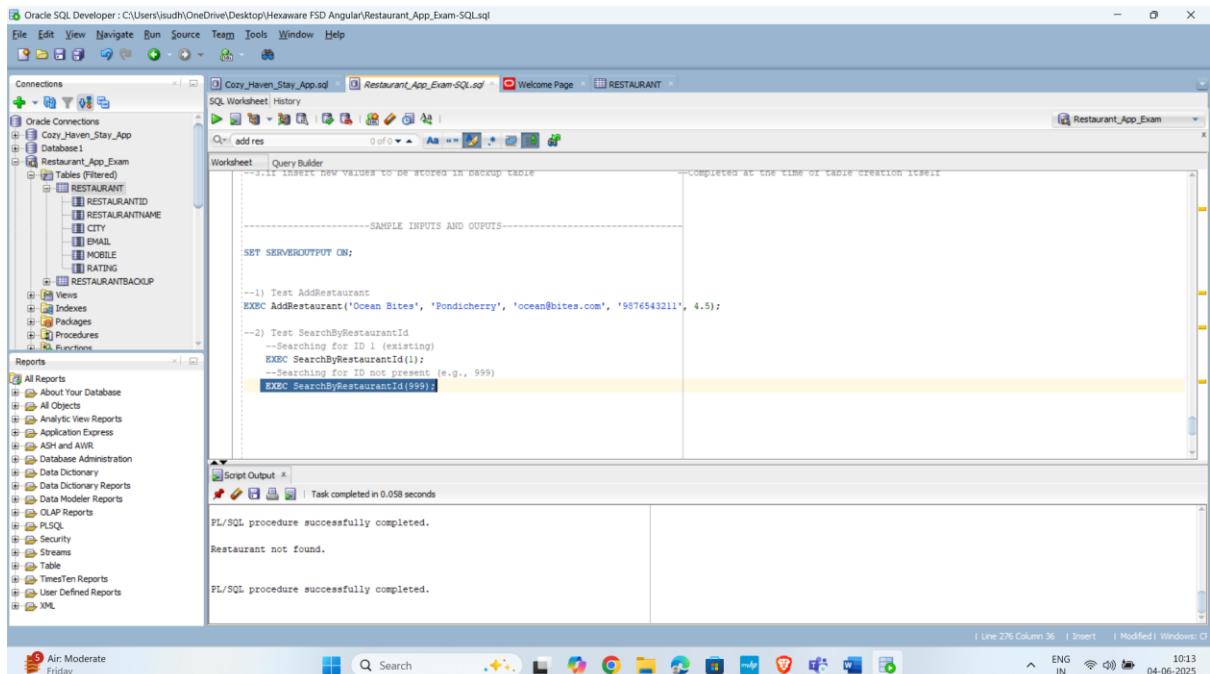
The 'Script Output' pane at the bottom shows the results of the executed code:

```
PL/SQL procedure successfully completed.

ID: 1, Name: Tasty Treats, City: Chennai, Email: treats@food.com, Mobile: 9123456789, Rating: 4.3

PL/SQL procedure successfully completed.
```

2.2 Searching for ID not present (e.g., 999)



This screenshot is identical to the previous one, showing the Oracle SQL Developer interface. The central workspace contains the same PL/SQL code as before:

```
--1) Test AddRestaurant
EXEC AddRestaurant('Ocean Bites', 'Pondicherry', 'ocean@bites.com', '9876543211', 4.5);
--2) Test SearchByRestaurantId
EXEC SearchByRestaurantId(1);
--Searching for ID not present (e.g., 999)
EXEC SearchByRestaurantId(999);
```

The 'Script Output' pane at the bottom shows the results of the executed code:

```
PL/SQL procedure successfully completed.

Restaurant not found.

PL/SQL procedure successfully completed.
```

3.Update Restaurant

Before Updation

The screenshot shows the Oracle SQL Developer interface with the RESTAURANT table selected. The table has columns: RESTAURANTID, RESTAURANTNAME, CITY, EMAIL, MOBILE, and RATING. The data is as follows:

RESTAURANTID	RESTAURANTNAME	CITY	EMAIL	MOBILE	RATING
1	Ocean Bites	Pondicherry	ocean@bites.com	9876543211	4.5
2	Tasty Treats	Chennai	treats@food.com	9123456789	4.3
3	Spice Hub	Madurai	spicehub@food.com	9012345678	4.6
4	Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2
5	Grill House	Trichy	grill@house.com	9998877665	4.8
6	Veg Delight	Salem	veg@delight.com	9887766554	4.1

After Updation

The screenshot shows the Oracle SQL Developer interface with the RESTAURANT table selected. The table has columns: RESTAURANTID, RESTAURANTNAME, CITY, EMAIL, MOBILE, and RATING. The data is as follows:

RESTAURANTID	RESTAURANTNAME	CITY	EMAIL	MOBILE	RATING
1	Ocean Bites	Pondicherry	ocean@bites.com	9876543211	4.5
2	Tasty Treats	Chennai	treats@food.com	9123456789	4.3
3	Spice Hub Deluxe	Madurai	spicehub@newmail.com	9012345678	4.7
4	Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2
5	Grill House	Trichy	grill@house.com	9998877665	4.8
6	Veg Delight	Salem	veg@delight.com	9887766554	4.1

The script output shows the execution of the update query:

```
--3) Test UpdateRestaurant
EXEC UpdateRestaurant(2, 'Spice Hub Deluxe', 'Madurai', 'spicehub@newmail.com', '9012345678', 4.7);
```

PL/SQL procedure successfully completed.

Restaurant updated.

PL/SQL procedure successfully completed.

3.1 Non Existant ID test

The screenshot shows the Oracle SQL Developer interface with the RESTAURANT table selected. The table has columns: RESTAURANTID, RESTAURANTNAME, CITY, EMAIL, MOBILE, and RATING. The data is as follows:

RESTAURANTID	RESTAURANTNAME	CITY	EMAIL	MOBILE	RATING
1	Ocean Bites	Pondicherry	ocean@bites.com	9876543211	4.5
2	Tasty Treats	Chennai	treats@food.com	9123456789	4.3
3	Spice Hub Deluxe	Madurai	spicehub@newmail.com	9012345678	4.7
4	Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2
5	Grill House	Trichy	grill@house.com	9998877665	4.8
6	Veg Delight	Salem	veg@delight.com	9887766554	4.1

The script output shows the execution of the update query:

```
--3) Test UpdateRestaurant
EXEC UpdateRestaurant(2, 'Spice Hub Deluxe', 'Madurai', 'spicehub@newmail.com', '9012345678', 4.7);

--3.1 Non Existant ID test
EXEC UpdateRestaurant(999, 'No Name', 'Nowhere', 'no@mail.com', '0000000000', 0);
```

PL/SQL procedure successfully completed.

Restaurant not found.

PL/SQL procedure successfully completed.

4.Deletion

Before

	RESTAURANTID	RESTAURANTNAME	CITY	EMAIL	MOBILE	RATING
1	6	Ocean Bites	Pondicherry	ocean@bites.com	9876543211	4.5
2	1	Tasty Treats	Chennai	treats@food.com	9123456789	4.3
3	2	Spice Hub Deluxe	Madurai	spicehub@newmail.com	9012345678	4.7
4	3	Curry Leaves	Coimbatore	curry@leaves.com	9876543210	4.2
5	4	Grill House	Trichy	grill@house.com	9998877665	4.8
6	5	Veg Delight	Salem	veg@delight.com	9887766554	4.1

After

The screenshot shows the Oracle SQL Developer interface. On the left, the Object Navigator displays various database objects like Indexes, Procedures, and Functions. The central workspace contains a script editor window with the following PL/SQL code:

```
--4) Test DeleteRestaurantById
EXEC DeleteRestaurantById(3);

--4.1 Deletion of Non Existent ID test
EXEC DeleteRestaurantById(999);
```

Below the script editor is a "Script Output" window showing the results of the execution:

```
PL/SQL procedure successfully completed.

Restaurant deleted.

PL/SQL procedure successfully completed.
```

At the bottom, a grid-based table view displays the same data as the initial screenshot, but it only contains 5 rows, indicating that the record with ID 3 has been deleted.

4.1 Non existent ID deletion

This screenshot is identical to the one above, showing the Oracle SQL Developer interface. The script editor contains the same PL/SQL code for deleting a non-existent ID:

```
--4) Test DeleteRestaurantById
EXEC DeleteRestaurantById(3);

--4.1 Deletion of Non Existent ID test
EXEC DeleteRestaurantById(999);
```

The "Script Output" window shows the results:

```
PL/SQL procedure successfully completed.

Restaurant not found.

PL/SQL procedure successfully completed.
```

The table view at the bottom also shows the same 5-row dataset as the previous screenshot, confirming that no records were deleted.

5. Cursor block to print all restaurants

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with tables like RESTAURANT, CITY, EMAIL, MOBILE, and RATING. The central workspace contains a query builder with the following PL/SQL code:

```
--5) Cursor block to print all restaurants
DECLARE
    CURSOR rest_cursor IS SELECT * FROM Restaurant;
    v_restaurant Restaurant%ROWTYPE;
BEGIN
    OPEN rest_cursor;
    LOOP
        FETCH rest_cursor INTO v_restaurant;
        EXIT WHEN rest_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('ID: ' || v_restaurant.restaurantId || ', Name: ' || v_restaurant.restaurantName ||
                            ', City: ' || v_restaurant.city || ', Email: ' || v_restaurant.email ||
                            ', Mobile: ' || v_restaurant.mobile || ', Rating: ' || v_restaurant.rating);
    END LOOP;
    CLOSE rest_cursor;
END;
```

The script output window shows the results of the query:

```
Restaurant not found.

PL/SQL procedure successfully completed.

ID: 6, Name: Ocean Bites, City: Pondicherry, Email: ocean@bites.com, Mobile: 9876543211, Rating: 4.5
ID: 1, Name: Tasty Treats, City: Chennai, Email: treats@food.com, Mobile: 9123456789, Rating: 4.3
ID: 3, Name: Spice Zone Delight, City: Madurai, Email: spices@spicezone.com, Mobile: 9012345678, Rating: 4.7
ID: 4, Name: Grill House, City: Kochi, Email: grillhouse.com, Mobile: 9998776665, Rating: 4.8
ID: 5, Name: Veg Delight, City: Salem, Email: vegdelight.com, Mobile: 9887766554, Rating: 4.1
```

6. SearchRestaurantById with output params

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with tables like RESTAURANT, CITY, EMAIL, MOBILE, and RATING. The central workspace contains a query builder with the following PL/SQL code:

```
--6 Test searchRestaurantById with output params
DECLARE
    v_name VARCHAR2(30);
    v_city VARCHAR2(30);
    v_email VARCHAR2(30);
    v_mobile VARCHAR2(15);
    v_rating NUMBER(5,2);
BEGIN
    searchRestaurantById(v_name, v_city, v_email, v_mobile, v_rating);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_name);
    DBMS_OUTPUT.PUT_LINE('City: ' || v_city);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
    DBMS_OUTPUT.PUT_LINE('Mobile: ' || v_mobile);
    DBMS_OUTPUT.PUT_LINE('Rating: ' || v_rating);
END;
```

The script output window shows the results of the query:

```
Task completed in 0.062 seconds

PL/SQL procedure successfully completed.

Name: Tasty Treats
City: Chennai
Email: treats@food.com
Mobile: 9123456789
Rating: 4.3
```

7. Verifying Triggers working by checking RestaurantBackup

Oracle SQL Developer : C:\Users\usudh\OneDrive\Desktop\Hexaware FSD Angular\Restaurant_App_Exam-SQL.sql

File Edit View Navigate Run Source Team Tools Window Help

Connections

Cozy_Haven_Stay_App.sql Restaurant_App_Exam-SQL.sql Welcome Page RESTAURANTBACKUP

Tables (Filtered)

- RESTAURANT
 - RbId
 - RestaurantId
 - RestaurantName
 - City
 - Email
 - Mobile
 - Rating
- RESTAURANTBACKUP
 - RbId
 - RestaurantId
 - RestaurantName
 - City
 - Email
 - Mobile
 - Rating
 - ActivityOn

Reports

- All Reports
- About Your Database
- All Objects
- Analytic View Reports
- Application Express
- ASH and AWR
- Database Administration
- Data Dictionary
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- PLSQL
- Security
- Streams
- Table
- TimesTen Reports
- User Defined Reports
- XML

Query Builder

```

v_name VARCHAR2(30);
v_email VARCHAR2(15);
v_mobile VARCHAR2(15);
v_rating NUMBER(5,2);

BEGIN
    searchRestaurantId(l, v_name, v_city, v_email, v_mobile, v_rating);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_name);
    DBMS_OUTPUT.PUT_LINE('City: ' || v_city);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
    DBMS_OUTPUT.PUT_LINE('Mobile: ' || v_mobile);
    DBMS_OUTPUT.PUT_LINE('Rating: ' || v_rating);
END;
/

```

--7 Verifying Triggers working by checking RestaurantBackup

SELECT RbId, restaurantId, restaurantName, operation, activityOn FROM RestaurantBackup ORDER BY RbId;

Script Output X Query Result X

SQL All Rows Fetched: 13 in 0.005 seconds

RbId	RestaurantId	RestaurantName	Operation	ActivityOn
5	5	5 Veg Delight	INSERT	04-06-25
6	6	1 Tasty Treats	INSERT	04-06-25
7	7	2 Spice Hub	INSERT	04-06-25
8	8	3 Curry Leaves	INSERT	04-06-25
9	9	4 Grill House	INSERT	04-06-25
10	10	5 Veg Delight	INSERT	04-06-25
11	11	6 Ocean Bites	INSERT	04-06-25
12	12	2 Spice Hub	UPDATE	04-06-25
13	13	3 Curry Leaves	DELETE	04-06-25

Line 330 Column 1 Insert Modified Windows 10:24 ENG IN 04-06-2023