**VI: Database Interaction and Transactions** for your **Movies Management System**.

## ****Objective****

You’ll write **PL/SQL programs** to interact with your database, focusing on:

DML/DDL operations

Cursors

Procedres and functions

Exception handling

Packages

### Use of ****window functions**** for analytics

### 1. ****Simple Problem Statement for Analytics****

**“Find out the top 3 most booked shows, based on the number of tickets sold.”**

2. **Create a Procedure to Fetch Booking Counts per Show  
  
CREATE OR REPLACE PROCEDURE show\_ticket\_counts IS**

**BEGIN**

**FOR rec IN (**

**SELECT s.show\_id, m.title, COUNT(\*) AS tickets\_sold**

**FROM ticket t**

**JOIN showtime s ON t.show\_id = s.show\_id**

**JOIN movie m ON s.movie\_id = m.movie\_id**

**GROUP BY s.show\_id, m.title**

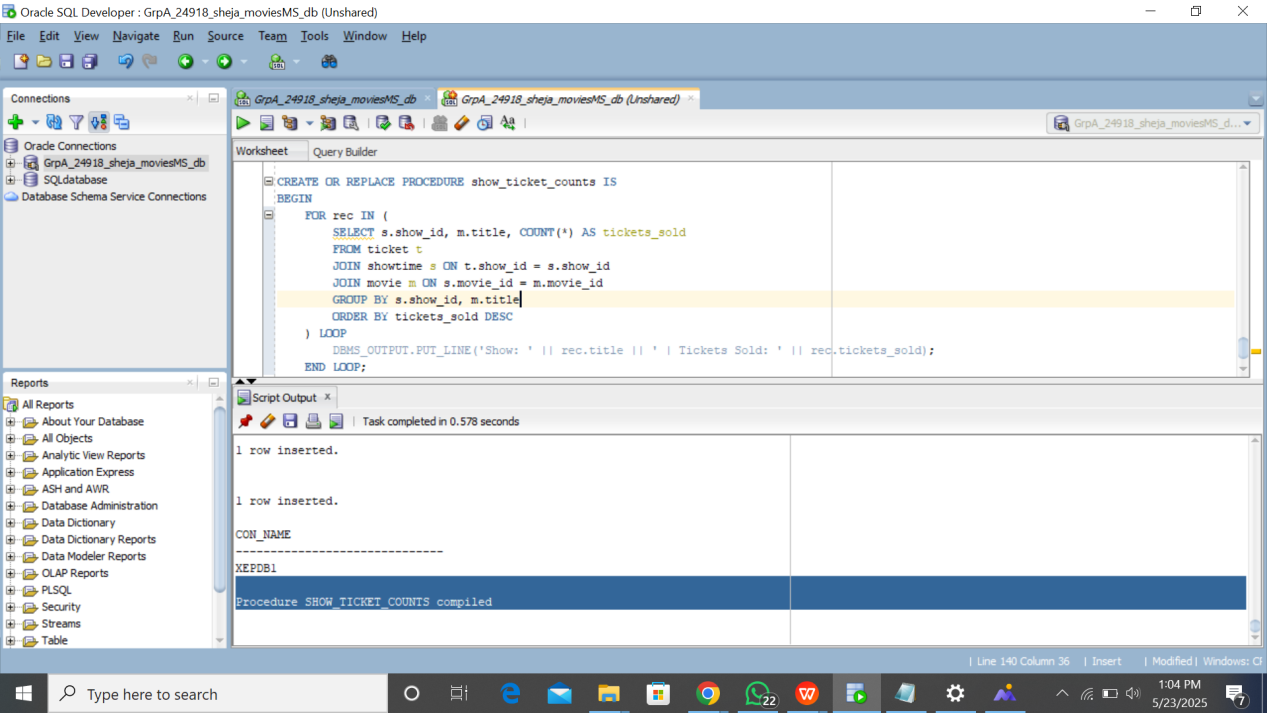
**ORDER BY tickets\_sold DESC**

**) LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Show: ' || rec.title || ' | Tickets Sold: ' || rec.tickets\_sold);**

**END LOOP;**

**END;**

**/  
  
**

3. **Function to Calculate Total Revenue for a Movie  
  
CREATE OR REPLACE FUNCTION total\_revenue\_by\_movie(p\_movie\_id NUMBER) RETURN NUMBER IS**

**total NUMBER := 0;**

**BEGIN**

**SELECT SUM(t.price)**

**INTO total**

**FROM ticket t**

**JOIN showtime s ON t.show\_id = s.show\_id**

**WHERE s.movie\_id = p\_movie\_id;**

**RETURN total;**

**EXCEPTION**

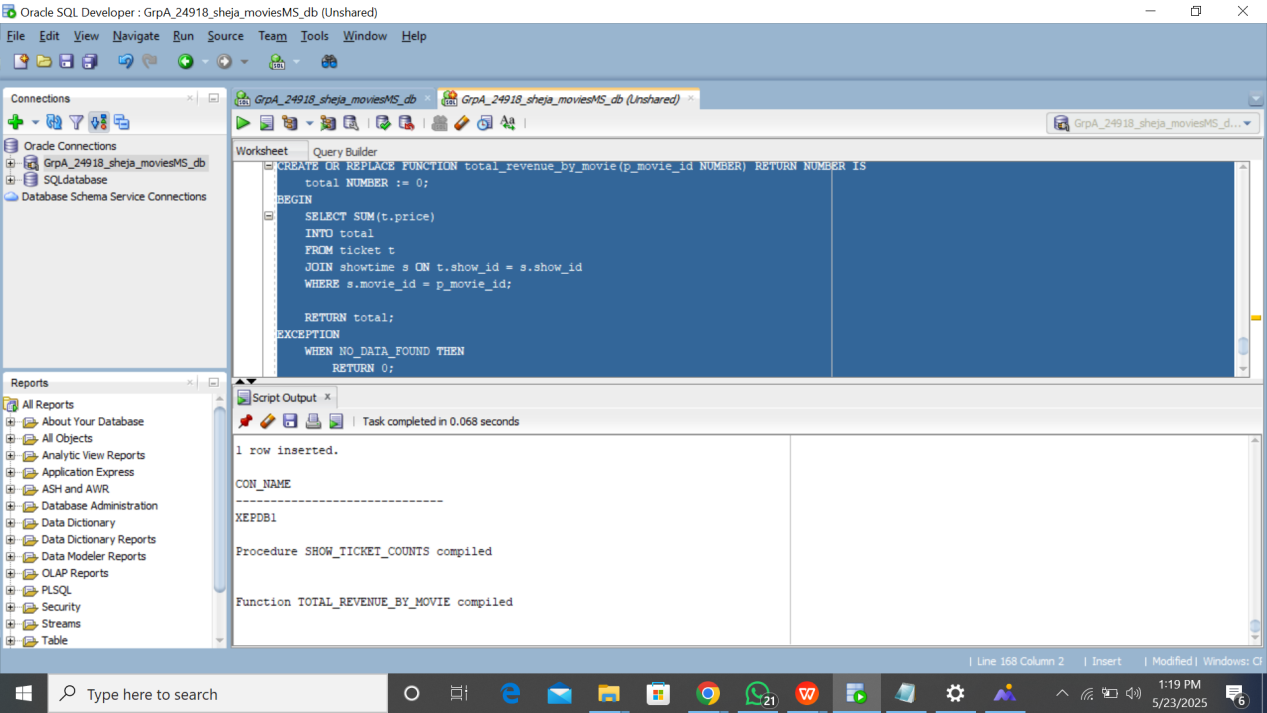
**WHEN NO\_DATA\_FOUND THEN**

**RETURN 0;**

**WHEN OTHERS THEN**

**RETURN -1;**

**END;**

**/  
  
**4. **Cursor Example: Loop Over Customers  
  
DECLARE**

**CURSOR c\_customers IS**

**SELECT full\_name, email FROM customer;**

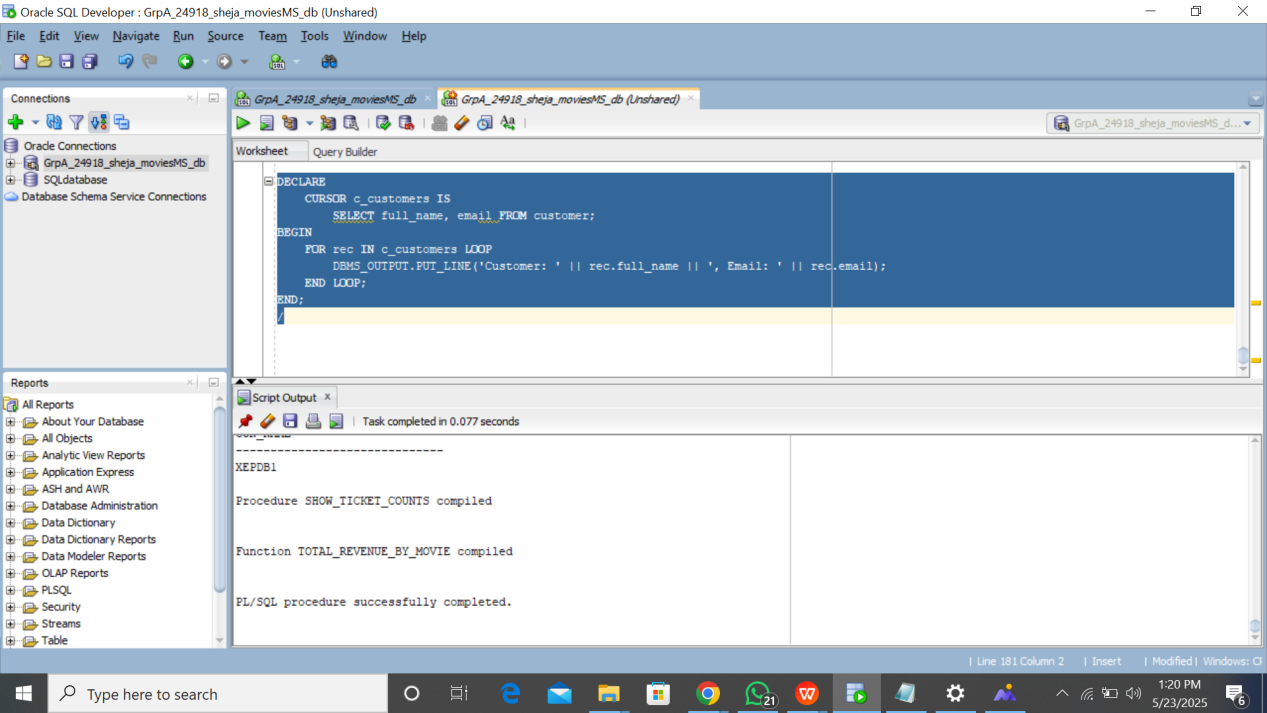
**BEGIN**

**FOR rec IN c\_customers LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Customer: ' || rec.full\_name || ', Email: ' || rec.email);**

**END LOOP;**

**END;**

**/  
  
**5. **Error Handling Example  
  
BEGIN**

**INSERT INTO ticket (show\_id, customer\_id, seat\_number, price)**

**VALUES (999, 1, 'X1', 5000); -- 999 is a non-existent show\_id**

**EXCEPTION**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('Error inserting ticket: ' || SQLERRM);**

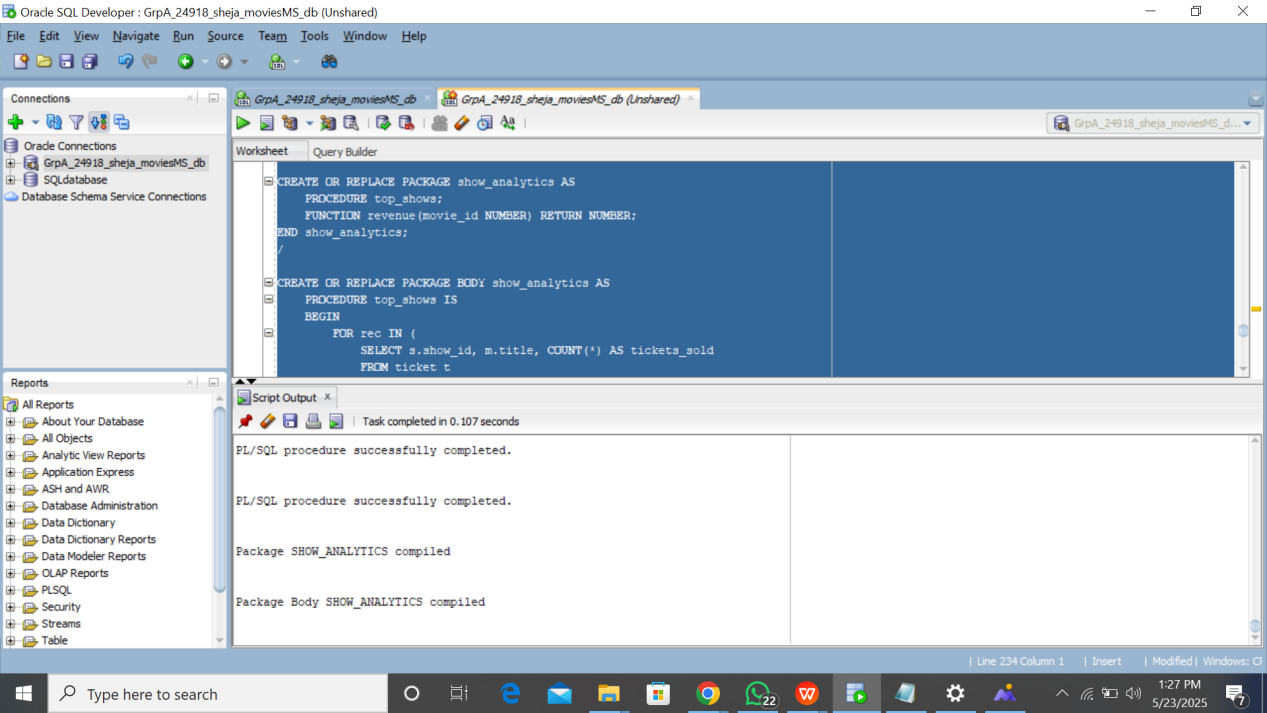
**END;**

**/**6. **Create a Package for Show Analytics  
  
CREATE OR REPLACE PACKAGE show\_analytics AS**

**PROCEDURE top\_shows;**

**FUNCTION revenue(movie\_id NUMBER) RETURN NUMBER;**

**END show\_analytics;**

**/  
  
**

**CREATE OR REPLACE PACKAGE BODY show\_analytics AS**

**PROCEDURE top\_shows IS**

**BEGIN**

**FOR rec IN (**

**SELECT s.show\_id, m.title, COUNT(\*) AS tickets\_sold**

**FROM ticket t**

**JOIN showtime s ON t.show\_id = s.show\_id**

**JOIN movie m ON s.movie\_id = m.movie\_id**

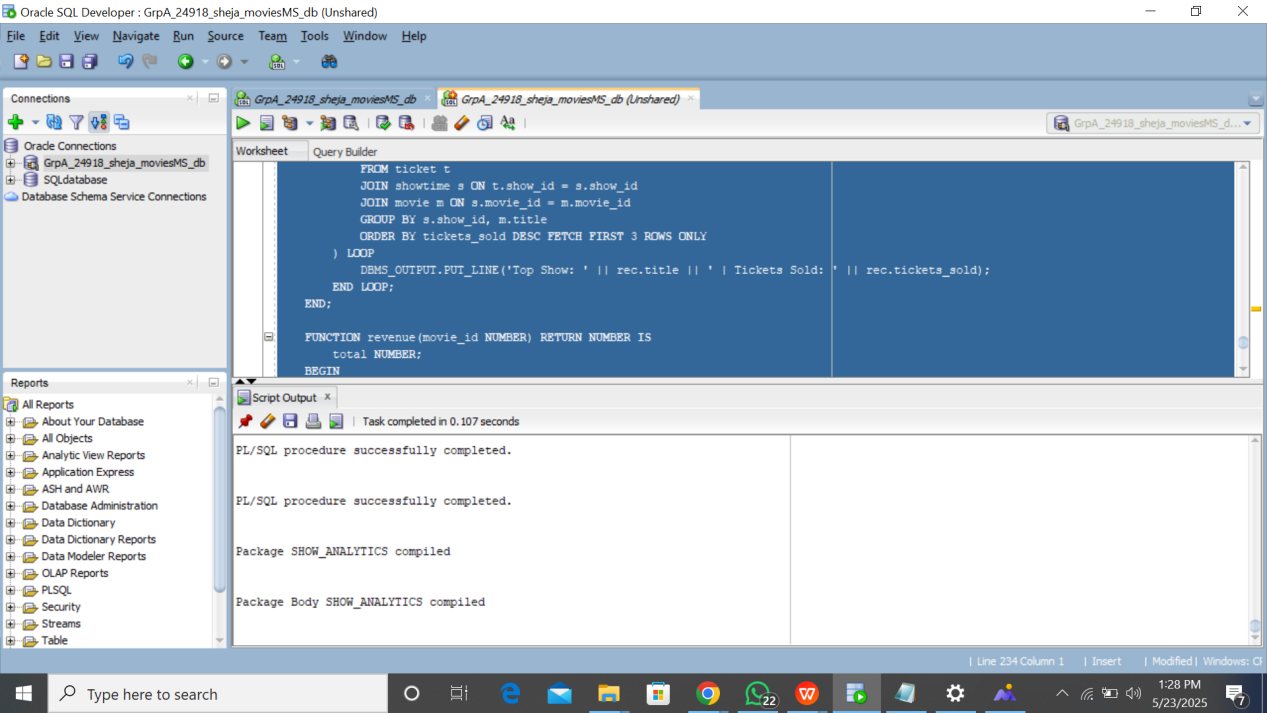
**GROUP BY s.show\_id, m.title**

**ORDER BY tickets\_sold DESC FETCH FIRST 3 ROWS ONLY**

**) LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Top Show: ' || rec.title || ' | Tickets Sold: ' || rec.tickets\_sold);**

**END LOOP;**

**END;  
  
**

**FUNCTION revenue(movie\_id NUMBER) RETURN NUMBER IS**

**total NUMBER;**

**BEGIN**

**SELECT SUM(price)**

**INTO total**

**FROM ticket t**

**JOIN showtime s ON t.show\_id = s.show\_id**

**WHERE s.movie\_id = movie\_id;**

**RETURN total;**

**EXCEPTION**

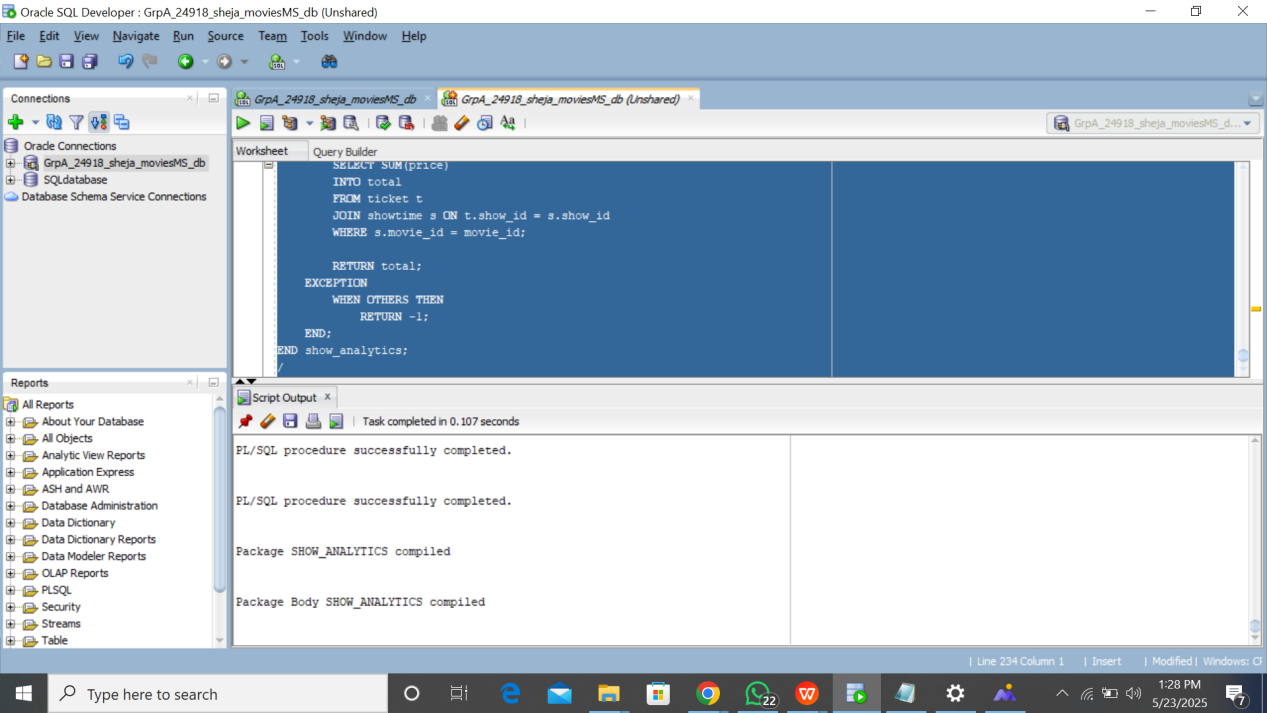
**WHEN OTHERS THEN**

**RETURN -1;**

**END;**

**END show\_analytics;**

**/**

****