STEP 1: DEFINE A SIMPLE PROBLEM STATEMENT

Problem Statement:

- Analyze the spending behavior of clients in the **Transaction** table.
- Group data by clients to calculate the total amount spent and the average transaction amount using window functions.
- Implement a function to determine the highest spending merchant for each client.

Step 2: DDL Operations (Data Definition Language)

Objective: Create a new table to store analysis results.

Syntax

```
-- Create Analysis Table to Store Aggregated Data-----

CREATE TABLE Transaction_Analysis (
   AnalysisID INT PRIMARY KEY,
   UserID INT NOT NULL,
   TotalSpent DECIMAL(10, 2) NOT NULL,
   AvgTransactionAmount DECIMAL(10, 2) NOT NULL,
   HighestSpendingMerchant VARCHAR(100),
   AnalysisDate DATE NOT NULL,
   FOREIGN KEY (UserID) REFERENCES Client(UserID)
);
```

Step 3: DML Operations (Data Manipulation Language)

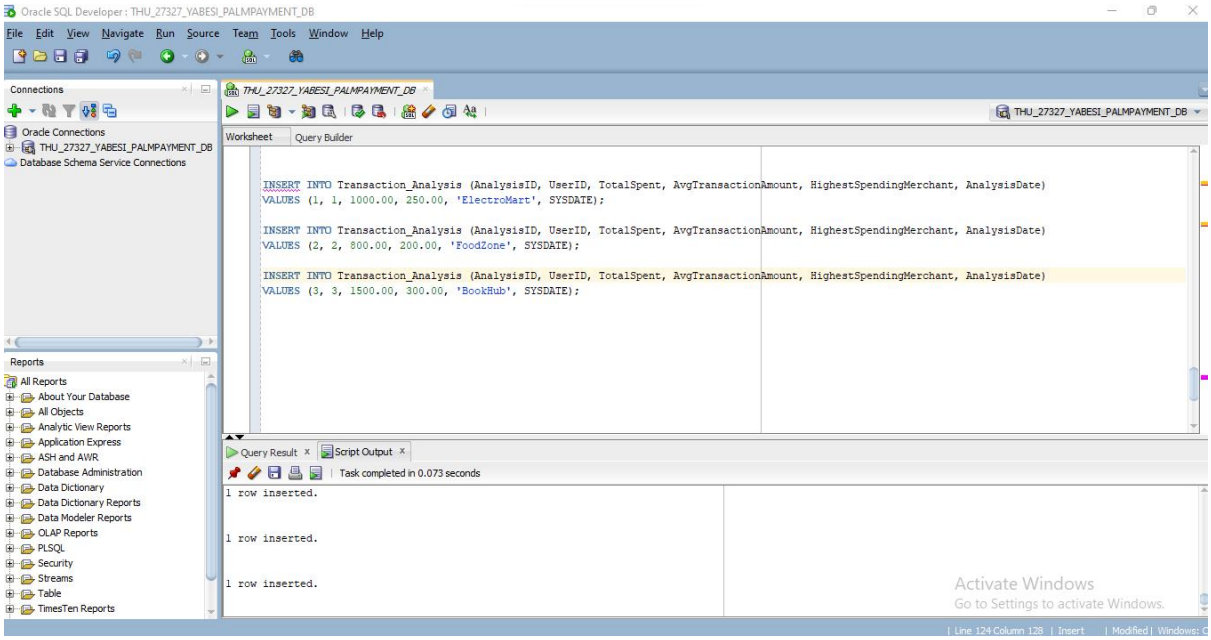
Objective: Insert, Update, and Delete data to interact with the tables.

1. <u>Insert Data into the Transaction_Analysis Table:</u>

INSERT INTO Transaction_Analysis (AnalysisID, UserID, TotalSpent, AvgTransactionAmount, HighestSpendingMerchant, AnalysisDate)
VALUES (1, 1, 1000.00, 250.00, 'ElectroMart', SYSDATE);

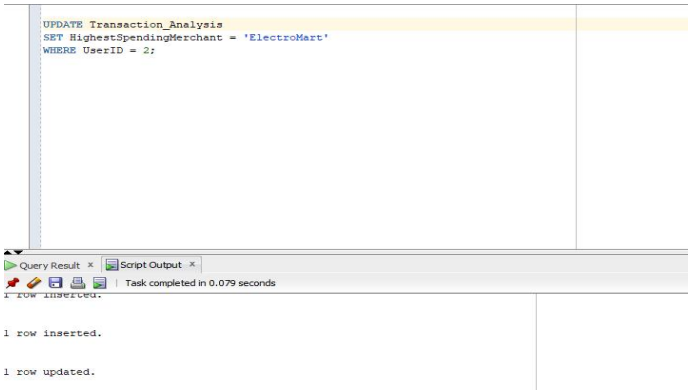
INSERT INTO Transaction_Analysis (AnalysisID, UserID, TotalSpent, AvgTransactionAmount, HighestSpendingMerchant, AnalysisDate)
VALUES (2, 2, 800.00, 200.00, 'FoodZone', SYSDATE);

INSERT INTO Transaction_Analysis (AnalysisID, UserID, TotalSpent, AvgTransactionAmount, HighestSpendingMerchant, AnalysisDate)
VALUES (3, 3, 1500.00, 300.00, 'BookHub', SYSDATE);



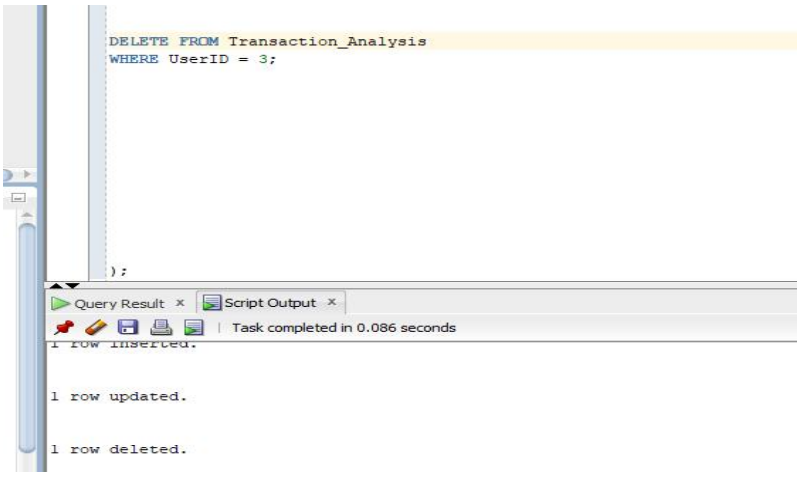
2. Update Data in the Transaction_Analysis Table:

UPDATE Transaction_Analysis
SET HighestSpendingMerchant = 'ElectroMart'
WHERE UserID = 2;



3. Delete Data from the Transaction_Analysis Table:

```
DELETE FROM Transaction_Analysis
WHERE UserID = 3;
```



Step 4: Implement a Simple Function Using Window Functions

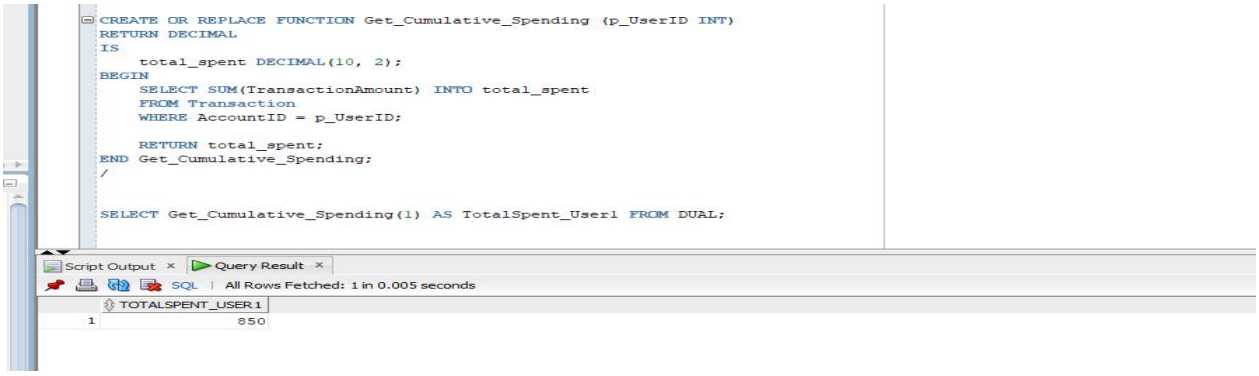
Objective: Calculate the cumulative spending for each user using the **SUM()** window function.

```
CREATE OR REPLACE FUNCTION Get_Cumulative_Spending (p_UserID INT)
RETURN DECIMAL
IS
   total_spent DECIMAL(10, 2);
BEGIN
   SELECT SUM(TransactionAmount) INTO total_spent
   FROM Transaction
   WHERE AccountID = p_UserID;

   RETURN total_spent;
END Get_Cumulative_Spending;
/
```

Test the Function:

```
SELECT Get_Cumulative_Spending(1) AS TotalSpent_User1 FROM DUAL;
```



Step 5: Implement a Procedure for Data Analysis

Objective: Analyze the spending behavior of all clients and store the results in the **Transaction_Analysis** table.

Syntax

```
CREATE OR REPLACE PROCEDURE Analyze_Spending

IS
   CURSOR trans_cursor IS
      SELECT
         t.AccountID,
         SUM(t.TransactionAmount) AS TotalSpent,
         AVG(t.TransactionAmount) AS AvgSpent,
         (SELECT m.MerchantName
          FROM Transaction t2
          JOIN Merchant m ON t2.MerchantID = m.MerchantID
          WHERE t2.AccountID = t.AccountID
          GROUP BY m.MerchantName
          ORDER BY SUM(t2.TransactionAmount) DESC
          FETCH FIRST 1 ROW ONLY) AS HighestSpendingMerchant
      FROM Transaction t
      GROUP BY t.AccountID;

   rec trans_cursor%ROWTYPE;
BEGIN
   FOR rec IN trans_cursor LOOP
      -- Insert the analysis data with sequence-generated AnalysisID
      INSERT INTO Transaction_Analysis
      (AnalysisID, UserID, TotalSpent, AvgTransactionAmount, HighestSpendingMerchant, AnalysisDate)
      VALUES
      (AnalysisID_Seq.NEXTVAL, rec.AccountID, rec.TotalSpent, rec.AvgSpent,
rec.HighestSpendingMerchant, SYSDATE);
   END LOOP;
 COMMIT;
END Analyze_Spending;
/
```

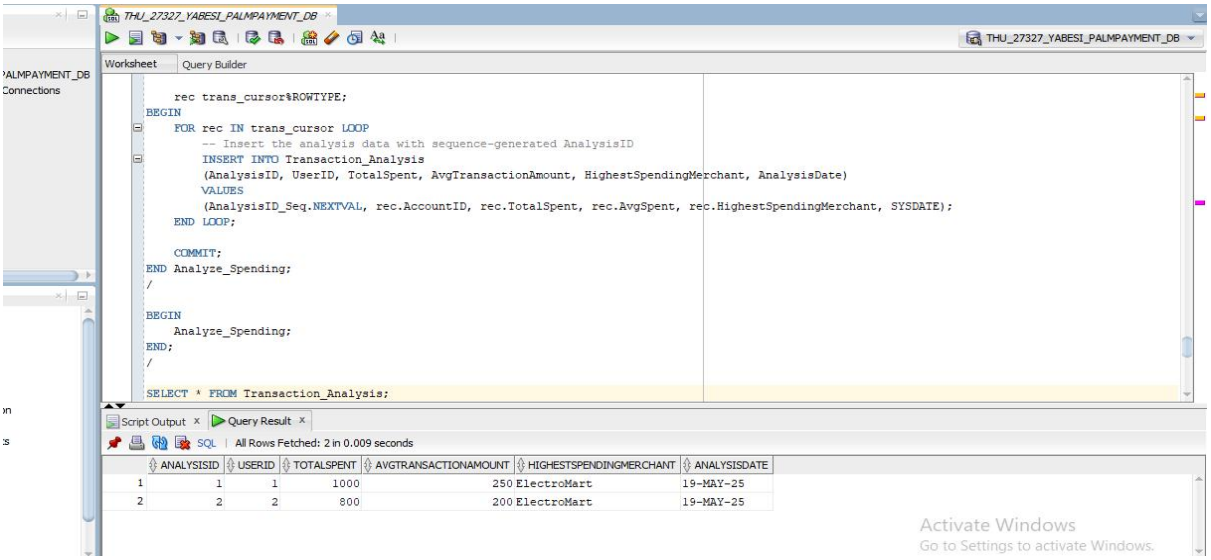
Execute the Procedure:

```
BEGIN
   Analyze_Spending;
END;
/
```

To show output

SELECT * FROM Transaction_Analysis;

output



Step 6: Create a Package for Data Analysis and Reporting

A package will consolidate the function and procedure for reusability.

1. ### Package Specification

```
CREATE OR REPLACE PACKAGE Data_Analysis_Pkg AS
   -- Procedure to analyze spending
   PROCEDURE Analyze_Spending;

   -- Function to get cumulative spending for a user
   FUNCTION Get_Cumulative_Spending(p_UserID INT) RETURN DECIMAL;
END Data_Analysis_Pkg;
/
```

2. ### Package Body

```
CREATE OR REPLACE PACKAGE BODY Data_Analysis_Pkg AS

  -- Procedure to analyze total and average spending
  PROCEDURE Analyze_Spending IS
    CURSOR trans_cursor IS
      SELECT AccountID, SUM(TransactionAmount) AS TotalSpent,
          AVG(TransactionAmount) AS AvgSpent
      FROM Transaction
      GROUP BY AccountID;

    rec trans_cursor%ROWTYPE;
    analysis_id INT := 1;
  BEGIN
    -- Delete previous analysis results
    DELETE FROM Transaction_Analysis;

    -- Loop through aggregated data and insert new records
    FOR rec IN trans_cursor LOOP
      INSERT INTO Transaction_Analysis (
        AnalysisID, UserID, TotalSpent, AvgTransactionAmount, AnalysisDate
      ) VALUES (
        analysis_id, rec.AccountID, rec.TotalSpent, rec.AvgSpent, SYSDATE
      );

      analysis_id := analysis_id + 1;
    END LOOP;

    COMMIT;
  END Analyze_Spending;

  -- Function to get cumulative spending for a user
  FUNCTION Get_Cumulative_Spending(p_UserID INT) RETURN DECIMAL IS
    total_spent DECIMAL(10, 2);
  BEGIN
    SELECT SUM(TransactionAmount) INTO total_spent
    FROM Transaction
    WHERE AccountID = p_UserID;

    RETURN NVL(total_spent, 0); -- Return 0 if null
  END Get_Cumulative_Spending;

END Data_Analysis_Pkg;
/
```

3. Test and Show Output

Making sure server output is enabled:

```
SET SERVEROUTPUT ON;
```

Testing

```
BEGIN
   -- Run the spending analysis procedure
   Data_Analysis_Pkg.Analyze_Spending;
   DBMS_OUTPUT.PUT_LINE('Spending analysis completed.');

   -- Test the cumulative spending function for a specific user
   DBMS_OUTPUT.PUT_LINE('Total spent by user 1: ' ||
Data_Analysis_Pkg.Get_Cumulative_Spending(1));

   DBMS_OUTPUT.PUT_LINE('Total spent by user 2: ' ||
Data_Analysis_Pkg.Get_Cumulative_Spending(2));
END;
/
```

OUTPUT

```
CREATE OR REPLACE PACKAGE BODY Data_Analysis_Pkg AS


   PROCEDURE Analyze_Spending IS
       CURSOR trans_cursor IS
           SELECT AccountID, SUM(TransactionAmount) AS TotalSpent,
                  AVG(TransactionAmount) AS AvgSpent
           FROM Transaction
           GROUP BY AccountID;

       rec trans_cursor%ROWTYPE;
       analysis_id INT := 1;
   BEGIN

       DELETE FROM Transaction_Analysis;
```

Script Output × | Query Result ×

Task completed in 0.126 seconds

```
Package DATA_ANALYSIS_PKG compiled


Package Body DATA_ANALYSIS_PKG compiled

✓ Spending analysis completed.
Total spent by user 1: 850
Total spent by user 2: 150


PL/SQL procedure successfully completed.
```