Task 6

Procedure, Functions, and Loops in PL/SQL.

Casestudy : Online Food Ordering System.

Objective :

The objective of this task is to design, implement and execute PL/SQL procedures, functions, and loops to handle real-world business scenarios related to an online food ordering system. This will help in automating transactions, improving database efficiency, and enforcing business rules in a structured manner.

Step 1 : Ensure the Necessary Tables Exist

```
DROP TABLE OrderTable PURGE;
DROP TABLE Delivery PURGE;
DROP TABLE Menu_Item PURGE;
CREATE TABLE OrderTable (
        Order_Id Number PRIMARY KEY,
        Cust_Id NUMBER,
        Order_date DATE,
        Order_Total NUMBER(10,2),
        Payment_status VARCHAR(20)
);
CREATE TABLE DELIVERY (
    Order_ID Number PRIMARY KEY,
    Delivery_status VARCHAR(20),
    FOREIGN KEY (order_Id) REFERENCES OrderTable (order_Id)
);
```

```
CREATE TABLE Menu_Item(
   Item_Id Number Primary key,
   Item_Name VARCHAR(100),
   Price Number(10,2)
);
Insert into OrderTable VALUES (1, 101, To_date('2024-02-01',
        'YYYY-MM-DD'), 250.50, 'Pending');
Insert into OrderTable VALUES (2, 102, To_date('2024-02-02','YYYY-
        MM-DD'), 400.75, 'Paid');
Insert into OrderTable VALUES(3,103,To_date('2024-02-03',
        'YYYY-MM-DD'), 150.00, 'Pending');

INSERT INTO Delivery VALUES (1, 'Pending');
INSERT INTO Delivery VALUES (2, 'Delivered');
INSERT INTO Delivery VALUES(3, 'Pending');
```

1. Procedure to update Payment status

step1 : Create a Procedure

```
CREATE OR REPLACE PROCEDURE Update_Payment_status(
      P_order_ID IN NUMBER,
      P_new_status. IN VARCHAR,
  ) AS
BEGIN
UPDATE OrderTable
  SET Payment_status = P_New_status
```

Expected Output:

Procedure Created.

Step 2 : Execution

```
BEGIN
    Update_Payment_status (1, 'Paid');
END;
/
```

Expected Output :

Statement processed.

Query 2: Function to calculate Total Revenue

Step 1 : Create a Function

```
CREATE OR REPLACE FUNCTION Get_Total_Revenue RETURN NUMBER AS
    v_Total_Revenue NUMBER;
BEGIN
    SELECT SUM(Order_Total) INTO v_Total_Revenue FROM OrderTable;
    RETURN v_Total_Revenue;
END;
/
```

Expected Output :

Function created

Step 2 : Execution

801.25

Query 3 : Mark all undelivered Orders as "Delayed ":

```
DECLARE
  v_Order_ID OrderTable.Order_ID%TYPE;
  CURSOR cur IS SELECT Order_ID FROM DELIVERY WHERE Delivery-
   status = 'Pending';
BEGIN
   OPEN cur;
  LOOP
    FETCH cur INTO v_Order_ID;
     EXIT WHEN cur%NOTFOUND;
  UPDATE Delivery
  SET DELIVERY_status = 'Delayed'
  WHERE Order_ID = v_Order_ID;
 DBMS_Output.PUT_LINE ('Order'|| v_Order_ID || 'marked as
                Delayed');
END LOOP;
CLOSE cur;
   COMMIT;
END;
/
```

Query 4 : Procedure to get order details by Customer ID

Step 1 : Create a Procedure

```
CREATE OR REPLACE PROCEDURE GET_Customer_Orders(
    P_cust_ID IN NUMBER
) AS
BEGIN
  FOR Order_rec IN (SELECT Order_ID, Order_date, Order_total,
                    Payment_status FROM OrderTable WHERE Cust_ID
                    = P_Cust_ID) LOOP
  DBMS_OUTPUT.PUT_LINE ('OrderID:' || Order_rec.Order_ID ||
                        ', Date:' || Order_rec.Order_date ||
                        ', Total:' || order_rec.Order_Total ||
                        ', status:' || order_rec.Payment_status);
END LOOP;
END;
/
```

Query 5 : Procedure to Apply Discount on Menu Items

Step 1 : Create a Procedure

```
CREATE OR REPLACE PROCEDURE Apply_discount(
   discount_percent IN NUMBER
)
 IS
BEGIN
  UPDATE Menu_Item
  SET PRICE = Price - (Price * discount_percent/100));
  COMMIT;
```