TASK-6

Procedures, Functions, and loops in PL/SQL:

Case Study: Online food ordering System

Objective:- The objective of this task is to design, imple-
-ment, and execute PL/SQL Procedures, functions and loops
to handle real-world business scenarios related to an
online food ordering System.

Step 2:

Execution:-
```
BEGIN
        Update_Payement_status (1,'Paid');
END;
```

Expected output:-

Payment status updated Successfully for order ID : 1
Statement Processed.

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,
Item_Name VARCHAR (100), Price Number (10,2) );
INSERT INTO OrderTable VALUES (1,101,TO-DATE ('2024-02-01',
 'YYY-MM-DD'), 250.50, 'Pending');
INSERT INTO OrderTable VALUES(2,102,TO_DATE('2024-02-02',
 'YYYY-MM-DD', 400.75 , 'Paid');
INSERTINTO Delivery values (1, 'Pending');
INSERT INTO Delivery values (2', 'Delivered');
INSERT INTO Menu_Item VALUES (1, 'Pizza', 500);
INSERT INTO Menu_Item Values (2,'Burger', 300);
```

1. Procedure to update Payment Status:-

Step 1: Create a Procedure

```
CREATE OR REPLACE PROCEDURE Update_Payement_Status(
   P_order_ID IN Number, P_New_Status IN VARCHAR2)
AS BEGIN
        UPDATE order_Table SET Payment_Status = P_New_St
```

```
atus WHERE Order_ID = P_order_ID;
COMMIT;
 DBMS_output.PUT_LINE ('Payment Status updated Success-
-fully for order ID:'|| P_order_ID);
END;
```

Expected output:-

Procedure Created.

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
Order Table;
    RETURN V_Total_Revenue;
 END;
```

Query-3:-

```
 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"
```

Query 2:- Function to Calculate Total Revenue.

Expected Output:-

Function Created.

Step 2: Execution.

```
  GET_TOTAL_REVENUE()
  801.25
```

Query 3:- Loop : Mark All Undelivered orders as "Delayed!

Expected output:-

1 row(s) updated.

Query 4:- Procedure to Get order Details by Customer ID.

Expected output:-

Procedure Created.

Step 2 : Execution

```
BEGIN
     Get_order_Details_BY_Customer(1);
 END;
```

Expected output:-

```
order ID : 1, Date : 2024-02-01, Total : 250.5, Payment : Paid
Statement Processed.
```

```
WHERE Order_ID = V_order_ID;
  DBMS_Output_Line ('order'||v_order_ID||' marked as Delayed');
END Loop;
Close Cur;
COMMIT;
END;
```

Query 4:- Create a Procedure.

```
CREATE OR REPLACE PROCEDURE Get_Customer_Orders (
 P_cost_ID IN NUMBER
 ) AS
 BEGIN
    FOR order_rec IN (SELECT order_ID, Order_Date, Order_
Total, Payment_Status FROM orderTable WHERE cost_ID =
 P_Cost_ID) LOOP DBMS_OUTPUT.PUT_LINE
 ('order ID:'|| order_vec.Order_ID||', Date:'|| order_vec.
 order_ID||', Total:'|| order_vec.Order_Total||', Status:'
 || order_rec.Payment_Status);
 END LOOP;
 END;
```

Query 5:-

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);
```

Query 5:- Procedure to Apply Discount on Menu Items

Expected output:-

Procedure Created.

Step 2:- Execution.

```
 BEGIN
     Apply_Discount (10);
  END;
```

Expected Output:-

Discount Applied: 10%.

Statement Processed.

VEL TECH	
EX No.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	15
SIGN WITH DATE	[illegible] 15/9/25

Result:- Hence, Implementing Procedures, functions and loops in PL/SQL has been done successfully.

[illegible] 15/9/25