12/9/25

Task-6 : Procedures, functions and loops

Case study : online food ordering system

Objective : TO objective of this task is to design, implement and execute PL/SQL procedures, functions and loops to handle real-world bussiness 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 :

Before running the procedure and functions, create the required tables in your oracle Database

```
DROP TABLE Ordertable PURGE;
DROP TABLE Delivery PURGE;
DROP TABLE Menu-Item PURGE;
CREATE TABLE Ordertable (order-ID Number primary key, Cost-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 2 (20), foregin key (order_ID) References order table (order-ID));
CREATE TABLE Menu-Item (Item-ID number primary key, Item-name varchar2(100) price number(10,2));
INSERT into Order-table values (1, 101, TO_date('2024-20-01', 'yyyy-MM-DD'), 250.50, 'pending');
Insert into ordertable values (2, 102, 50-Date('2024-03-03', yyyy-MM-DD)' 400.75, 'Paid');
Insert Into orderable value (3, TO-Date ('2024-02-05', 'yyyy-MM-DD'), 15000, 'pending');
Insert into Delivery values (1, 'pending');
Insert into Delivery values (2, 'Delivered');
Insert into Delivery values (3, 'pending');
Insert into menu-Item values (1, 'pizza', 500);
Insert into menu-Item values (2, 'Burger', 300);
Insert into menu-Item values (3, 'Pasta', 400);
```

```
Order ID: 1, Date: 01-FEB-24, Total:
250.5, Status: Paid
Statement processed.
```

```
Discount Applied: 10%
Statement processed.
```

```
Payment status updated successfully
for order ID: 1
Statement processed.
```

```
GET_TOTAL_REVENUE()
--------------------
            801.25
```

1. Procedure to update payment status

Step 1: Create a procedure.

```
Create OR Replace procedure update_payment_status(
P_order_ID IN Number, P_new_status IN varcha
AS Begin Update ordertable set payment_status=P_New
_status where order_ID=P_order_ID;
Commit;
DBMS_output put_line('payment status updated
for order ID :'||P_order_ID);      successfully)
End;
/
```

Expected output:

procedure created

Step 2: Execution:

```
Begin
    update_payment_status(1, 'paid');
    End;
/
```

Expected output:

payment status updated successfully for order ID:1

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 fro
  Order table;
Return v_total_revenue;
END;
/
```

Expected Output:

Function created

Step 2: Execution

GET_TOTAL_REVENDE()

801.25

Query 3: loop: mark All undelivered orders as "Delayed"

```
Declare v_order_ID order table.order_ID%Type;
```

order_ID	Cust_ID	order_Date	order_total	payment_status
1	101	2024-02-01	250.50	Paid
2	102	2024-02-02	400.75	Paid
3	103	2024-02-03	150.00	Pending

order_ID	Delivery_status
1	Delayed
2	Delivered
3	Delayed

```
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%NOT FOUND;

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;
/
```

Expected output :- 1 row(s) update

Query 4 :- Procedure to get order details by customer ID

Step 1 :- create a procedure

```
CREATE OR REPLACE PROCEDURE Get_customer_orders(
  P_WST_ID IN NUMBER
  ) AS
  BEGIN
    FOR order_rec IN (SELECT order_ID, order_date,
                      order_total, payment_status
    FROM order table WHERE cost_ID = P_cost_ID) LOOP
DBMS_OUTPUT.PUT_LINE('Order ID: ' || order_rec.order_ID ||
                     ', Date: ' || order_rec.order_Date ||
                     ', Total: ' || order_rec.order_Total ||
                     ', Status: ' || order_rec.payment_status);
    END LOOP;
  END;
/
```

Expected output :-
Procedure created

Step 2 :- Executed

```
BEGIN
  Get_order_Details_By_customer(1);
END;
/
```

Expected output :- Order ID : 1, Date : 2024-02-01, Total : 250.5, Payment : paid Statement processed.

Item_ID	Item_name	price
1	Pizza	450.00
2	Burger	450.00
3	Pasta	405.00

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_Items
Set price = price - (price * discount_percent/100);
commit;
DBMS_output.put_line('Discount Applied :'|| discount_
percent || '%.');
End;
/
```

Excepected output :-

procedure created

Step 2 : Execution

```
Begin Apply_Discount (10);
    End;
/
```

Expected output :-

Discount Applied : 10%.
Statement processed.

Result :- Thus, the PL/SQL procedures, functions and loops on number theory bussiness scenario experiment was successfully Complemented and results are verified.

VIT TECH - CSE	
EX NO.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (20)	19
SIGN WITH DATE	19/9/25