17/9/25

Task-6: Procedures, functions and loops

Aim:

To Produres, functions and loops

Objective:

To Objective this task is to design, implement and execute PL/SQL Procedures, functions and loops to handle real-world business scenarious related to an online food ordering system. This will help in automating transctions, improving database efficiency and enforcing business rules in a structured manner.

Step 1:

Before running the prosedure and functions, create the required tables in your oracle database.

```
DROP TABLE ordertable PURGE;

DROP TABLE Delivery PURGE;

DROP TABLE Meno_Item PURGE;
CREATE TABLE Delivery (order_ID Number Primary key,
Delivery_status VARCHAR(20), foregin key (order_ID) References
ordertable (order_ID));
CREATE TABLE ordertable (order_ID Number Primary key,
```

Task-6: Procedures function and loops

Objective: To objective of two task is to design, implement and execute all SQL Procedurces, functions and loops to handle real-world bussiness the scenario as related to an online food ordering system. This will help in automating tranctions, improving database efficiency and enforcing business roles actual and menmer.

Step 1: Ensure the neccessory tables exists:

Before running the produce and functions create the required table in your oracle database.

```
DROP TABLE order table PURGE;
DROP TABLE Delivery PURGE;
DROP TABLE Menu_Item PURGE;
CREATE TABLE order table order_ID Number
Primary key cost_ID Number, order_Data DATE,
order_total Number (10,2), Payment_status varchar
(20));
CREATE TABLE Delivery (order_ID Number Primary
key Delivery_status varchar(20), foriga key
(order_ID) refereces order table (order_ID);
INSERT into order table values (1, 101, To_date,
'2024-20-01', 'YYYY-MMM-DD'), 250.25.
```

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

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

`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 order table set Payment_
status = P_New_status where order_ID = P_order_ID;
Commit;
DBMS_output Put_line (`Payment status updated for
                                    Successfully);
order ID: // P_order_ID);
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.

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

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

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
  order table;
 Return V_total_Revenue;
 END;
/
```

Expected output

Function Created.

Step 2: Execution

```
GET_TOTAL_REVENUE()
```

801.25

Query 3:- loop; mark All underlined orders as "Delayed".

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

order_ID	Delivery_status
1	Delayed
2	Delivered
3	Delayed

```
Dbms_output.put_line('Order ' || v_order || ' marked
as Delayed');
END loop;
Close cur;
Commit;
END;
/
```

Expected output: / row(s) update

Query 4: Procedure to get order Details by the 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 order ~~and~~ 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.

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

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

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.

Query 5:- Procedure to apply Discount on menu_Items

Step-1: Create a Proceduce.

```
Create or Replace Proceduce Apply_Discount (discount_
Percent IN number) is
Begin
update Menu_Item
set Prince = Price - (Price * discount_Percent /100);
Commit;
DBMS_output.PUT_line(`Discount Applied:'|| discount Percent
||'%');
END;
/
```

Expected output:

Procedure created.

Result: Thus, the PL/SQL Procedures, functions and loops on number theory bussiness scenario experment was successfully completed and results are verified.

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