step1:- ensure the necessary tables Exist:

```
DROP TABLE order Table 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 order Table VALUES(1,101,TO-DATE(
'2024-02-01', 'YYYY-MM-DD),250.50, 'Pending');
INSERT INTO order Table VALUES(2,102, TO_DATE
('2024-02-02', 'YYYY-MM-DD'),400.75, 'paid');
INSERT INTO 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:-

step1:- create a procedure.

```
CREATE OR REPLACE PROCEDURE update_payment
_status(P_order_ID IN Number, P_New_status
IN VARCHAR2)
AS BEGIN
    UPDATE order_Table SET payment_status=
```

TASK-6:

Procedures, Functions, and loops in PL/SQL

Q case study:- 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.

step2:-

```
Execution:- BEGIN
        update_payment_status(1,'paid');
        END;
```

Expected output:-

payment status update successfully for order ID: 1

statement processed.

P_New_status WHERE order_ID = P_Order_ID;

COMMIT;

DBMS_output.put_LINE ('payment status updated successfully for order ID: ' || P_order_ID);

END;

Expected output:-

Procedure created.

Step1:- 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_orderID;

Exit WHEN CUR%.NOTFOUND;

UPDATE Delivery

SET Delivery-status = 'Delayed';

Query 2:- Function to calculate Total revenue

Expected output:-

Function created.

Step2: 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.

Step2:-

Execution

Begin

Get_order_details_By_customer(1);

END;

Expected output:-

OrderID: 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_cust
_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_cust_ID) Loop DBMS_output.put LINE
('order ID: '|| 'Total: '||order_ID||', Date: '||order_rec
||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/
commit;
DBMS_output.put_LINE('Discount applied: '||discount
_percent || %.');
END;

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: 10.1.
statement processed.

VELTECH	
EX No.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	15
SIGN WITH DATE	

Result:- Thus the study of procedures, Functions, and loops in SQL is successfully verified.