**Charotar University of Science and Technology**

**Faculty of Technology and Engineering**

**Department of Computer Engineering**

**Subject:** CE263 – Database Management System

**External Practical Exam (April- May 25)**

**SET-2**

**Semester: 4th (B. Tech.) Maximum Marks: 40**

**Date: 02/05/2025, Friday Time: 09:15 AM to 12:30 PM**

**Student ID – D24DCE147**

**Student Name – Nevil Dhinoja**

**Part A (20 Marks)**

Total: 40 Marks  
- 20 Marks SQL Queries  
- 20 Marks PL/SQL

**Use the given E-Commerce Management System for performing queries**

# E-Commerce Management System Schema

Tables:  
- Shoppers  
- Outlets  
- Orders  
- OrderDetails  
- Products  
- Payments  
- Staff  
- Order\_Staff

Create a table and insert the value for the following table.

## 1. Shoppers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ShopperID | Name | Email | Phone | City | MemberSincen |
| 1 | Alok Jain | alok@mail.com | 9999988888 | Delhi | 2021-02-10 |
| 2 | Riya Kapoor | riya@gmail.com | 8888877777 | Mumbai | 2022-05-14 |
| 3 | Mehul Vyas | mehul@yahoo.com | 7777766666 | Ahmedabad | 2020-09-05 |
| 4 | Sneha Iyer | sneha@mail.com | 6666655555 | Pune | 2019-11-18 |
| 5 | Vikram Das | vikram@xyz.com | 5555544444 | Chennai | 2023-01-22 |

## 2. Outlets

|  |  |  |  |
| --- | --- | --- | --- |
| OutletID | OutletName | City | Code |
| 201 | Central Plaza | Delhi | EC001 |
| 202 | Galaxy Mall | Mumbai | EC002 |
| 203 | Urban Mart | Pune | EC003 |
| 204 | Royal Arcade | Chennai | EC004 |

## 3. Orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrderID | ShopperID | OutletID | OrderDate | TotalAmount |
| 301 | 1 | 201 | 2024-02-14 | 4500.00 |
| 302 | 2 | 202 | 2024-03-01 | 7200.00 |
| 303 | 3 | 203 | 2023-12-21 | 3850.00 |
| 304 | 4 | 204 | 2022-11-12 | 5100.00 |

## 4. OrderDetails

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DetailID | OrderID | ProductID | Quantity | Price |
| 501 | 301 | 601 | 1 | 1200.00 |
| 502 | 301 | 602 | 2 | 3500.00 |

## 5. Products

|  |  |  |  |
| --- | --- | --- | --- |
| ProductID | ProductName | Category | Price |
| 601 | Laptop Bag | Accessories | 1200.00 |
| 602 | Bluetooth Speaker | Electronics | 3500.00 |
| 603 | Sneakers | Footwear | 2200.00 |

## 6. Payments

|  |  |  |  |
| --- | --- | --- | --- |
| PaymentID | OrderID | Amount | PaymentDate |
| 701 | 301 | 3000.00 | 2024-03-01 |
| 702 | 302 | 4000.00 | 2024-03-05 |

## 7. Staff

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| StaffID | Name | OutletID | Role | HireDate |
| 801 | Kunal Bhatt | 201 | Manager | 2021-04-01 |
| 802 | Neha Arora | 202 | Clerk | 2022-08-12 |

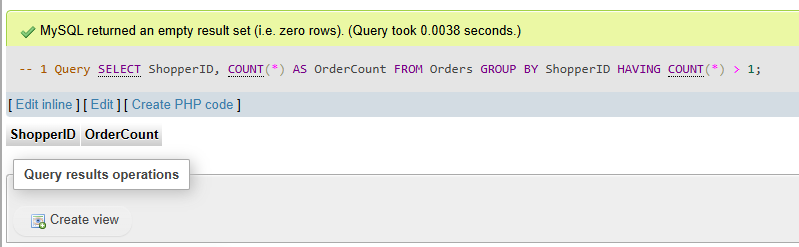
## 8. Order\_Staff

|  |  |
| --- | --- |
| OrderID | StaffID |
| 301 | 801 |
| 302 | 802 |

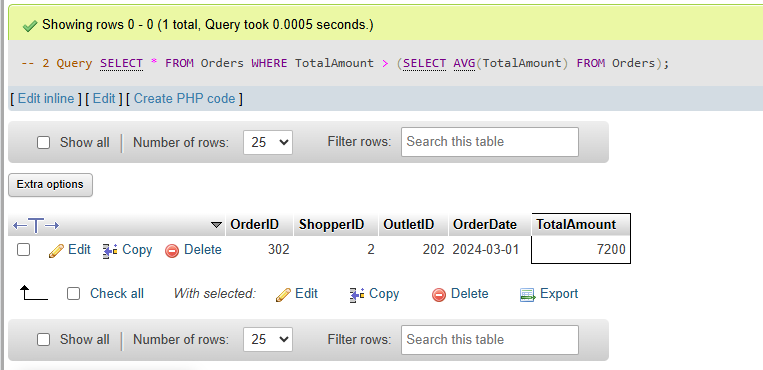
**Perform following queries based on the given schema:**

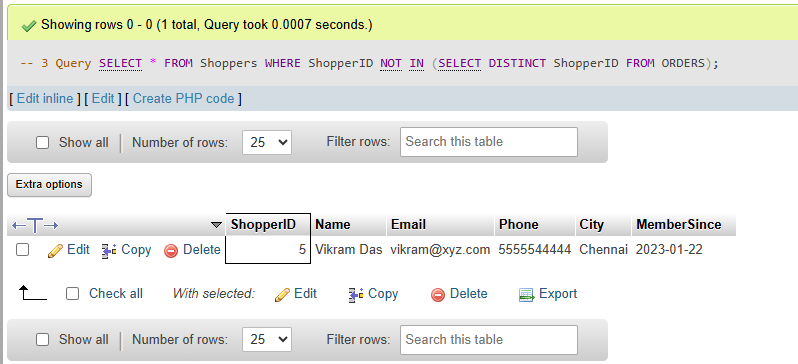
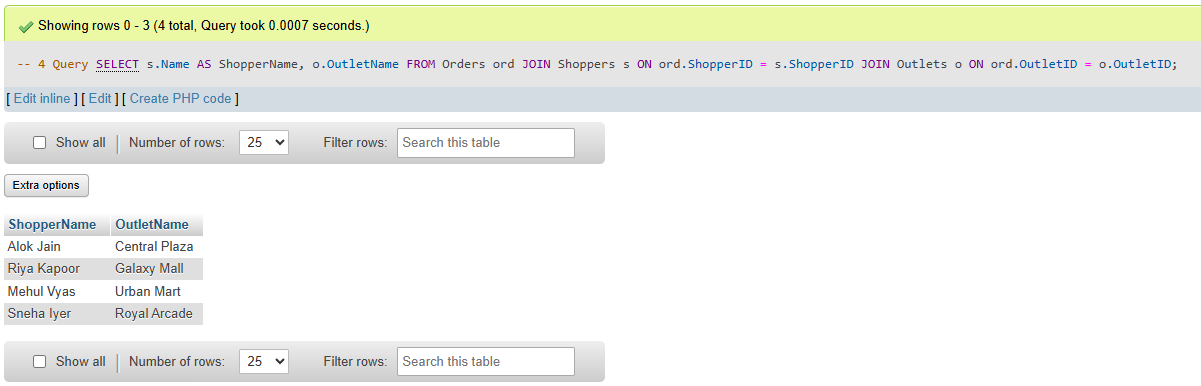
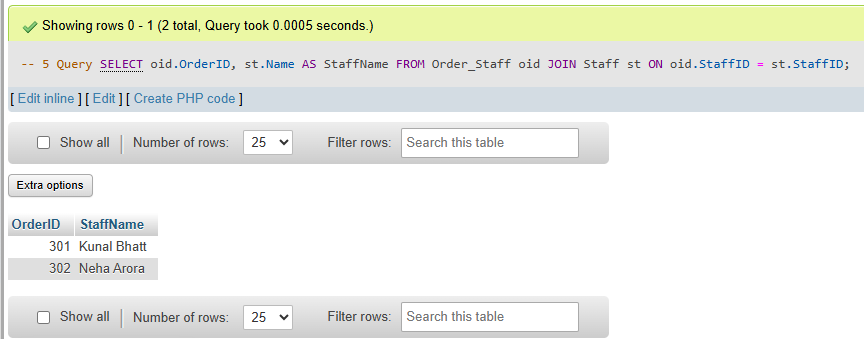
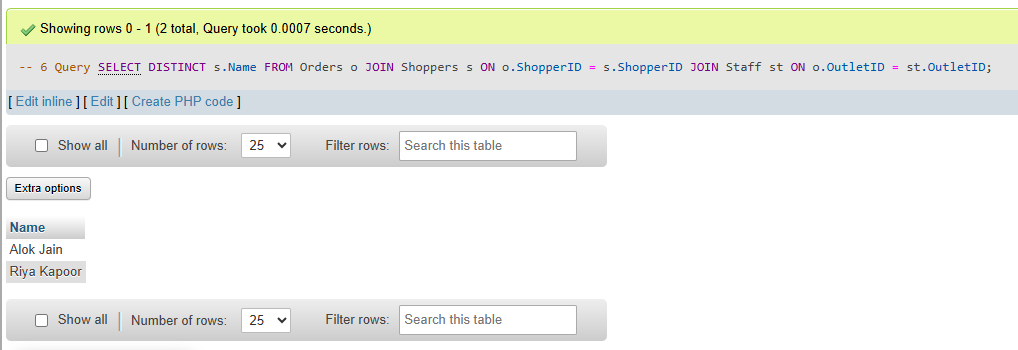
1. List shoppers who have placed more than one order. (2 marks)

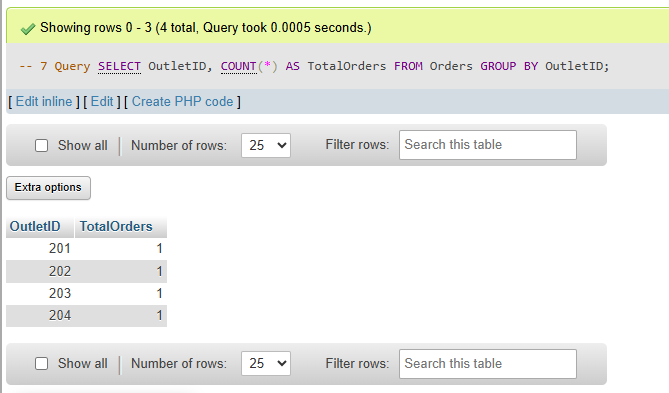
* Because no one has placed more than ’1’ Order – hence no result.



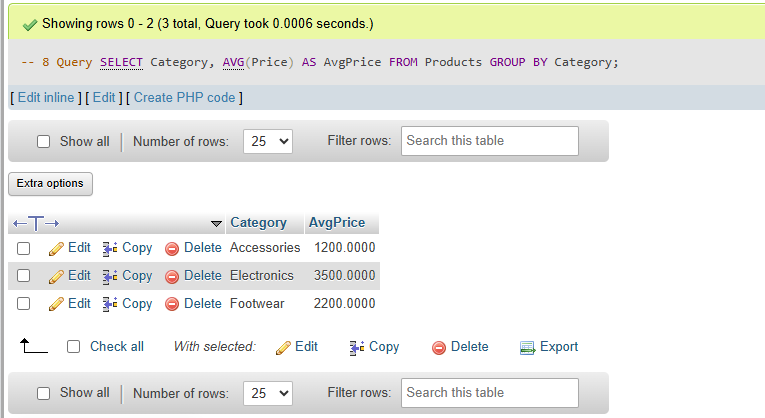
1. Find orders where the total amount exceeds the average order amount. (2 marks)



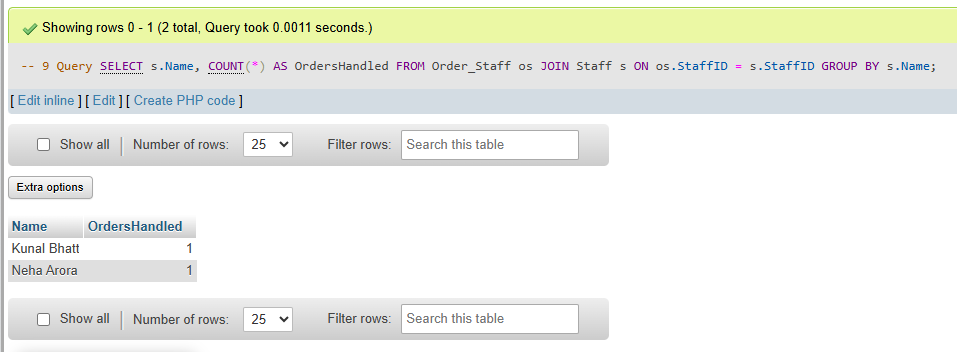
1. Show shoppers who haven’t placed any orders. (2 marks)
2. List each shopper’s name along with the outlet they placed orders at. (2 marks)
3. Display order IDs and the names of staff members assigned to them. (2 marks)
4. Show all shoppers who placed orders from the same outlet where a staff member works. (2 marks)
5. Find the total number of orders in each outlet. (2 marks)

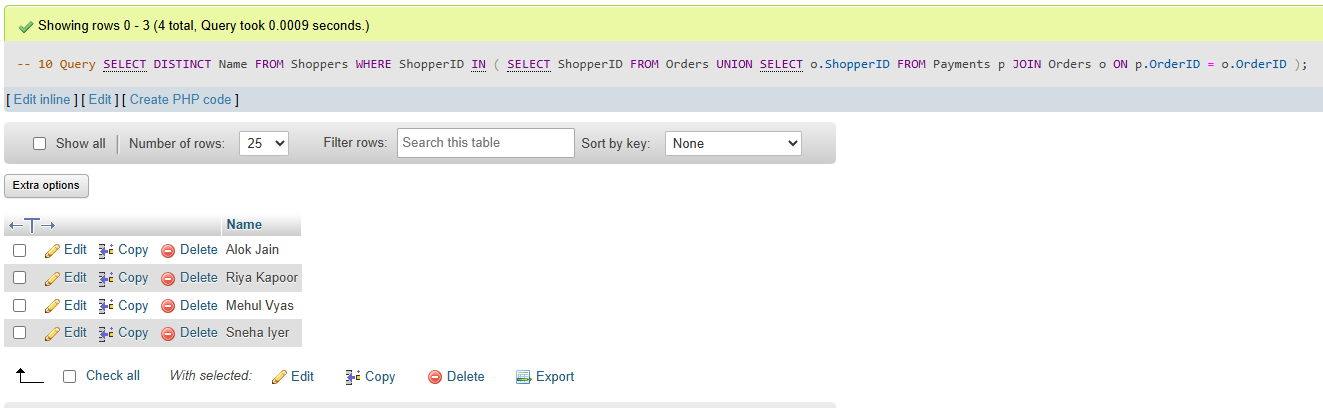


1. Calculate the average product price by category. (2 marks)



1. Display the number of orders each staff member handled. (2 marks)



1. Retrieve a list of all shoppers who have either placed an order or made a payment. (2 marks)

**Part B (20 Marks)**

# Q11. Update Old Orders Using Explicit Cursor (5 Marks)

Write a PL/SQL block that uses an explicit cursor to find orders placed before the year 2023 and update their status to 'Archived'.  
Assumption: The Orders table contains a column Status.

**Solution**

BEGIN

FOR rec IN (SELECT OrderID FROM Orders WHERE OrderDate < TO\_DATE('2023-01-01', 'YYYY-MM-DD')) LOOP

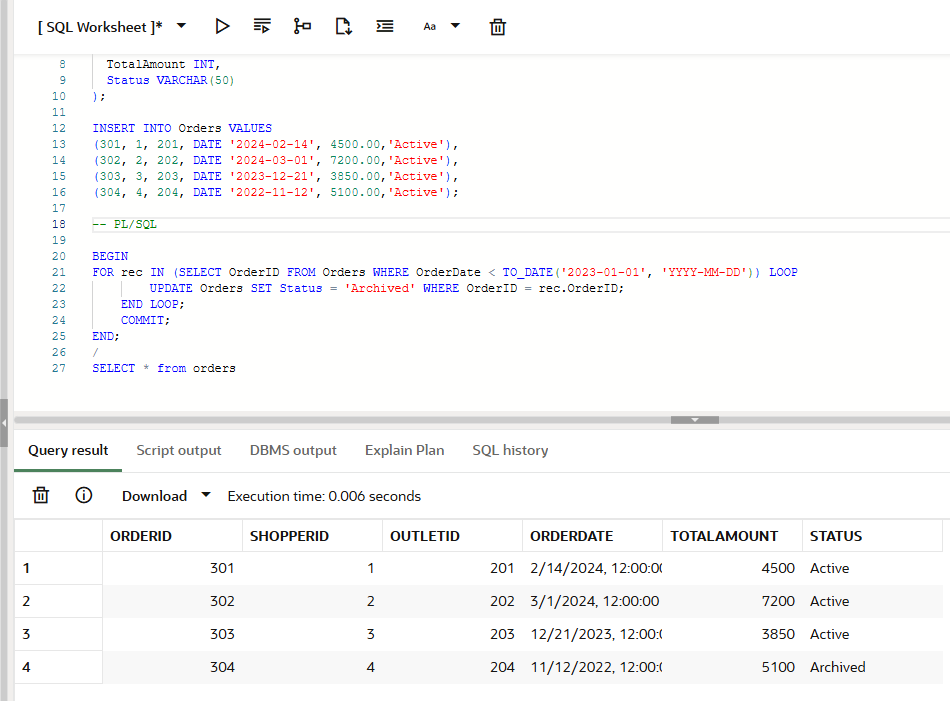
UPDATE Orders SET Status = 'Archived' WHERE OrderID = rec.OrderID;

END LOOP;

COMMIT;

END;

/



# Q12. Shopper Spend Check (5 Marks)

Write a PL/SQL block that checks the total order amount for each shopper.  
Requirements:

- If the total amount is less than ₹5000, insert a record into a new table Low\_Spenders with:  
 - ShopperID  
 - Name  
 - TotalAmount  
  
- If the total is ₹5000 or more, display a message like:  
 'Shopper [Name] has sufficient spend: ₹[TotalAmount]'

**Solution –**

CREATE TABLE Low\_Spenders ( ShopperID NUMBER,

Name VARCHAR2(50),

TotalAmount NUMBER );

DECLARE

CURSOR c IS

SELECT s.ShopperID, s.Name, SUM(o.TotalAmount) AS Total

FROM Shoppers s

JOIN Orders o ON s.ShopperID = o.ShopperID

GROUP BY s.ShopperID, s.Name;

BEGIN

FOR record IN c LOOP

IF record.Total < 5000 THEN

INSERT INTO Low\_Spenders VALUES (record.ShopperID, record.Name, record.Total);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Shopper ' || record.Name || ' has sufficient spend: ₹' || record.Total);

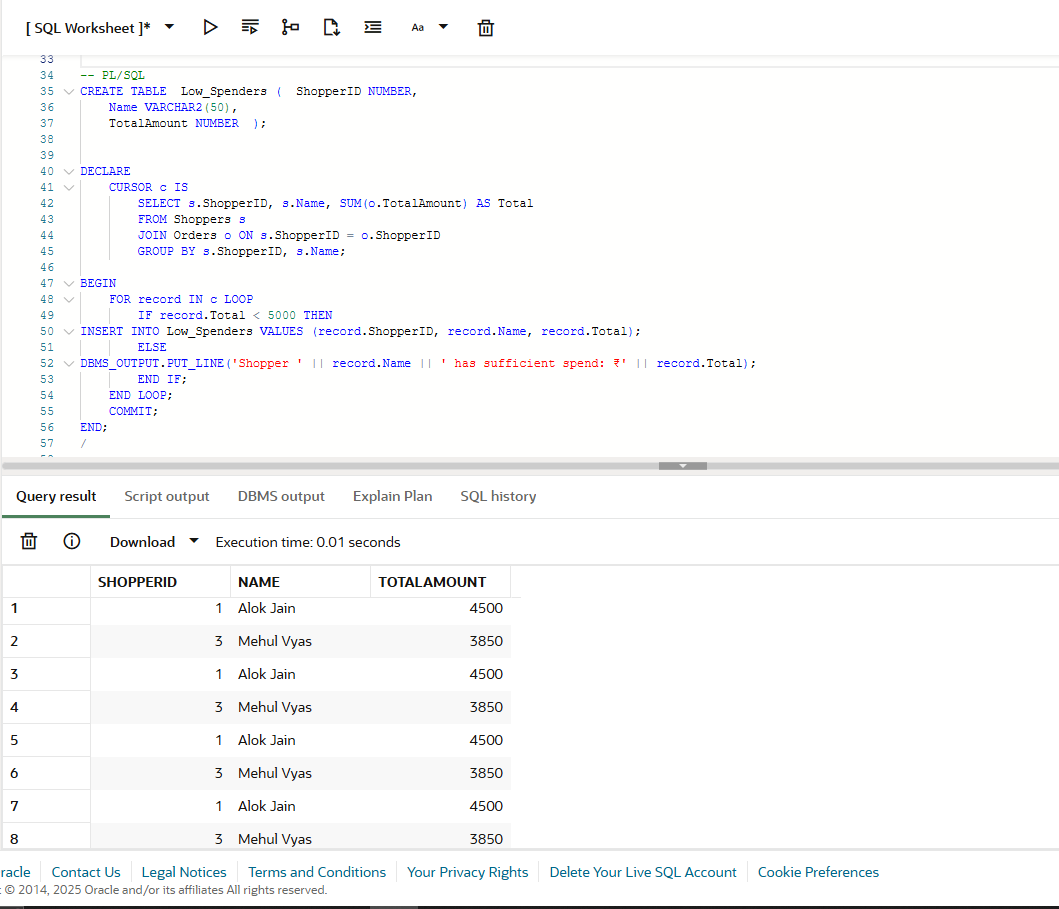
END IF;

END LOOP;

COMMIT;

END;

/



**Q13. Apply Tiered Discount to Orders (5 Marks)**

Write a PL/SQL block to apply a discount to orders based on the following total amount ranges:

|  |  |
| --- | --- |
| Order Amount Range | Discount Percentage |
| ₹1–5000 | 5% |
| ₹5001–10000 | 7% |
| ₹10001 and above | 10% |

The block should update the discount amount for each order according to this logic.

**Solution –**

ALTER TABLE Orders ADD Discount NUMBER;

BEGIN

FOR rec IN (SELECT OrderID, TotalAmount FROM Orders) LOOP

IF rec.TotalAmount <= 5000 THEN

UPDATE Orders SET Discount = rec.TotalAmount \* 0.05 WHERE OrderID = rec.OrderID;

ELSIF rec.TotalAmount <= 10000 THEN

UPDATE Orders SET Discount = rec.TotalAmount \* 0.07 WHERE OrderID = rec.OrderID;

ELSE

UPDATE Orders SET Discount = rec.TotalAmount \* 0.10 WHERE OrderID = rec.OrderID;

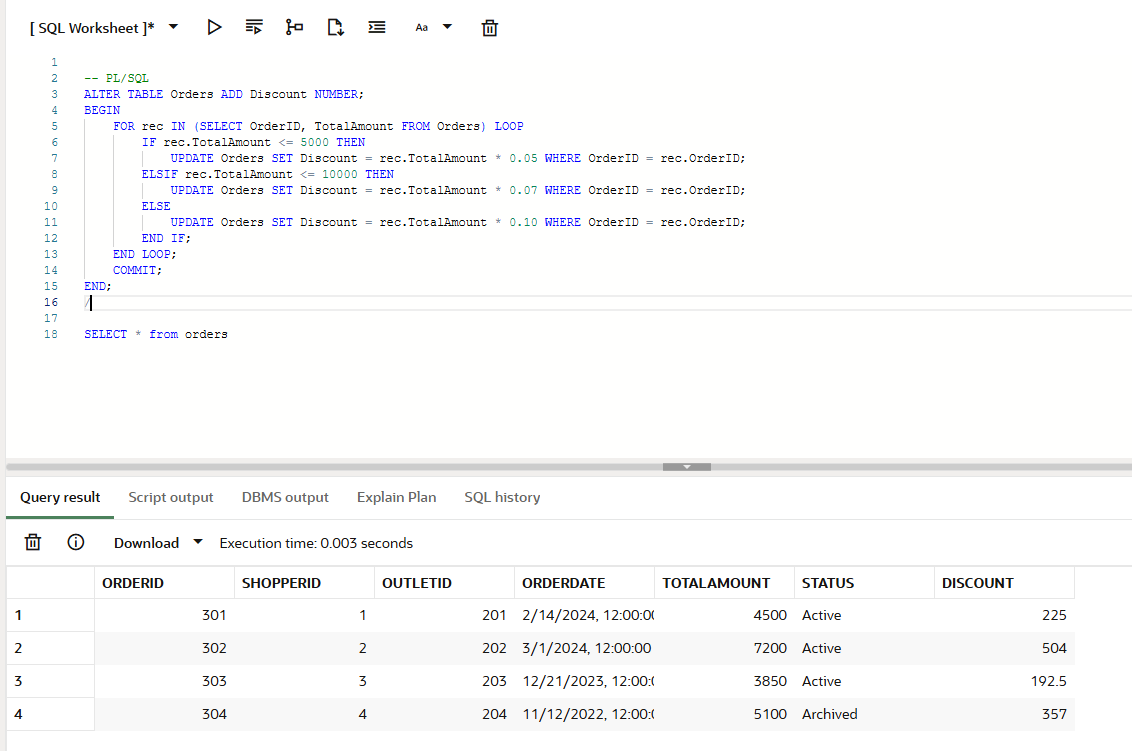
END IF;

END LOOP;

COMMIT;

END;

**/**



# Q14. Payment Management Function & Procedure (5 Marks)

Create a function and a procedure to handle order payments.

Function: GetOutstandingAmount  
Input: order\_id (NUMBER)  
- Output: Returns the remaining payment for the given order.

Procedure: MakePayment  
- Inputs: order\_id (NUMBER), amount (NUMBER)  
- Logic:  
 - Call the GetOutstandingAmount function.  
 - If the amount exceeds the outstanding amount, raise an error using RAISE\_APPLICATION\_ERROR.  
 - Otherwise, insert the payment into the Payments table.  
 - Display the new outstanding amount using DBMS\_OUTPUT.

**Solution –**

CREATE OR REPLACE FUNCTION GetOutstandingAmount(order\_id IN NUMBER)

RETURN NUMBER IS

total\_amt NUMBER;

paid\_amt NUMBER;

BEGIN

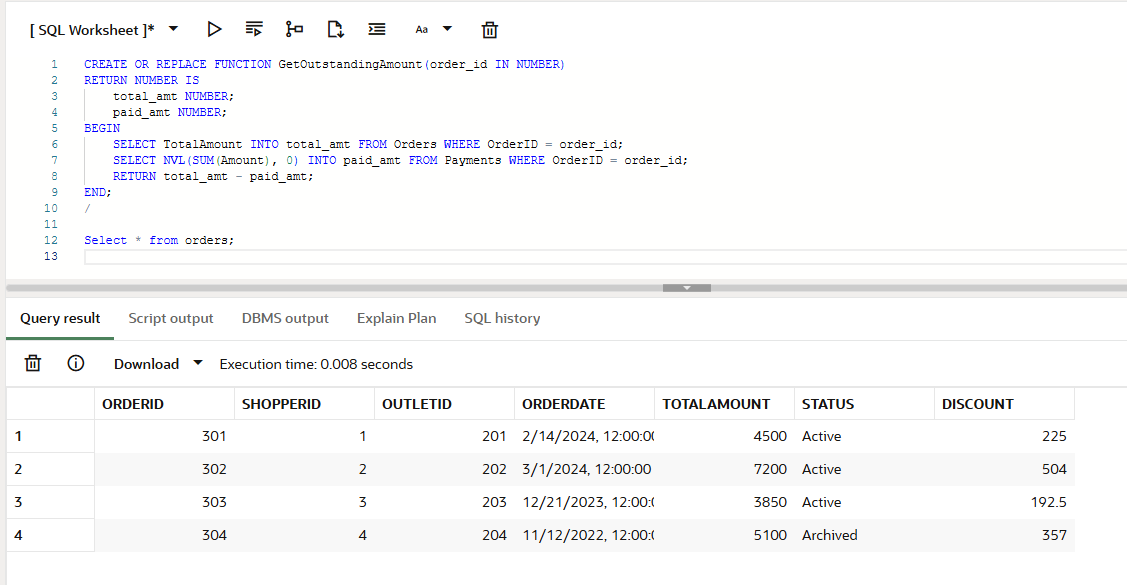
SELECT TotalAmount INTO total\_amt FROM Orders WHERE OrderID = order\_id;

SELECT NVL(SUM(Amount), 0) INTO paid\_amt FROM Payments WHERE OrderID = order\_id;

RETURN total\_amt - paid\_amt;

END;

/



**Procedure -**

CREATE OR REPLACE PROCEDURE MakePayment(order\_id IN NUMBER, amount IN NUMBER) IS

outstanding NUMBER;

BEGIN

outstanding := GetOutstandingAmount(order\_id);

IF amount > outstanding THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Payment exceeds outstanding amount.');

ELSE

INSERT INTO Payments (PaymentID, OrderID, Amount, PaymentDate)

VALUES ((SELECT NVL(MAX(PaymentID), 700) + 1 FROM Payments), order\_id, amount, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Remaining outstanding amount: ' || (outstanding - amount));

END IF;

COMMIT;

END;

/

## Marks Distribution

|  |  |  |
| --- | --- | --- |
| Question No. | Topic | Marks |
| Q11 | Update Old Orders Using Cursor | 5 |
| Q12 | Shopper Spend Check & Conditional Insert | 5 |
| Q13 | Apply Tiered Discount to Orders | 5 |
| Q14 | Function & Procedure for Payments | 5 |
| Total | --- | 20 |