EASTERN INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTING

AND INFORMATION TECHNOLOGY

□□□

Practice Assignment 7

Practice Assignment – Quarter 4, 2023-2024

Course Name: Database

Course Code: CSE 301

Student's Full Name: Phan Ngoc Hanh Nhi

Student ID: 2131209002

***Instruction*:**

** Students are allowing to write their answers (like SQL queries, Screen shot of outputs, etc.) in word file (Answer sheet) provided by instructor. After finishing the assignment, students must convert the word file (Answer sheet) into a PDF file. Finally, students upload the file in Moodle.*

1. Create the following tables in a new database 'Assignment3':

Clients(**Client_Number**, Client_Name, Address, City, Pincode, Province, Amount_Paid, Amount_Due)
Product(**Product_Number**, Product_Name, Quantity_On_Hand, Quantity_Sell, Sell_Price, Cost_Price)
Salesman (**Salesman_Number**, Salesman _Name, Address, City, Pincode, Province, Salary, Sales_Target, Target_Achieve, Phone)
Salesorder(**Order_Number**, Order_Date**, Client_Number, Salesman_Number**, Delivery_Status, Delivery_Date, Order_Status)
Salesorderdetails(**Order_Number, Product_Number**, Order_Quantity)

a) SQL UNION

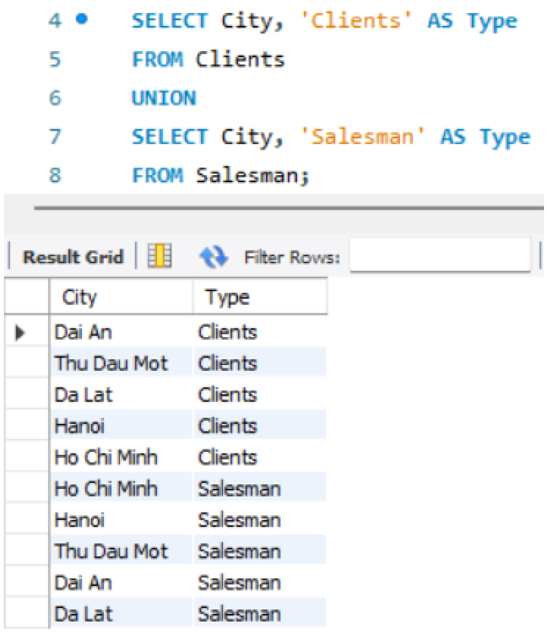
Syntax:

SELECT *column_name(s)* FROM *table1*
UNION
SELECT *column_name(s)* FROM *table2*;

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:

SELECT *column_name(s)* FROM *table1*
UNION ALL
SELECT *column_name(s)* FROM *table2*;

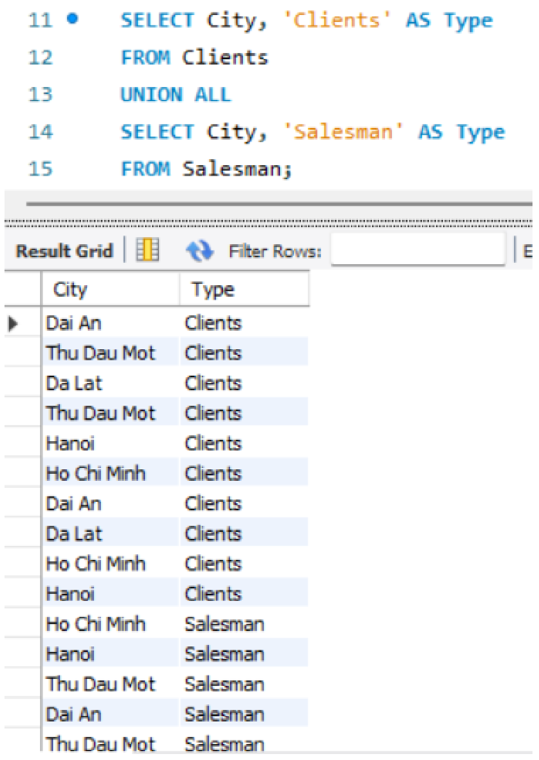
1. SQL statement returns the cities (only distinct values) from both the "Clients" and the "salesman" table.



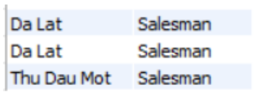
```sql
SELECT City, 'Clients' AS Type
FROM Clients
UNION
SELECT City, 'Salesman' AS Type
FROM Salesman;
```

City	Type
Dai An	Clients
Thu Dau Mot	Clients
Da Lat	Clients
Hanoi	Clients
Ho Chi Minh	Clients
Ho Chi Minh	Salesman
Hanoi	Salesman
Thu Dau Mot	Salesman
Dai An	Salesman
Da Lat	Salesman

2. SQL statement returns the cities (duplicate values also) both the "Clients" and the "salesman" table.

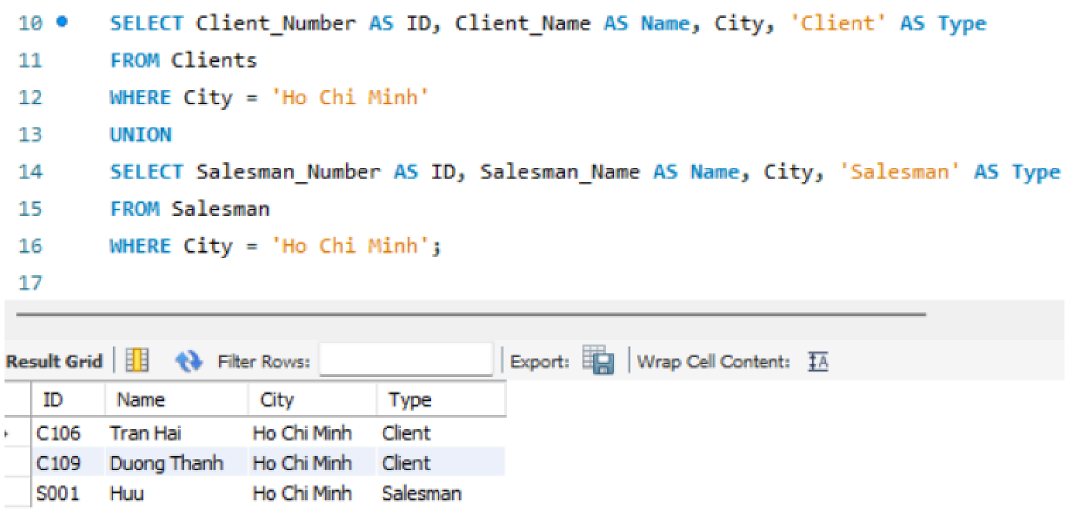


```sql
SELECT City, 'Clients' AS Type
FROM Clients
UNION ALL
SELECT City, 'Salesman' AS Type
FROM Salesman;
```



City	Type
Dai An	Clients
Thu Dau Mot	Clients
Da Lat	Clients
Thu Dau Mot	Clients
Hanoi	Clients
Ho Chi Minh	Clients
Dai An	Clients
Da Lat	Clients
Ho Chi Minh	Clients
Hanoi	Clients
Ho Chi Minh	Salesman
Hanoi	Salesman
Thu Dau Mot	Salesman
Dai An	Salesman
Thu Dau Mot	Salesman
Da Lat	Salesman
Da Lat	Salesman
Thu Dau Mot	Salesman

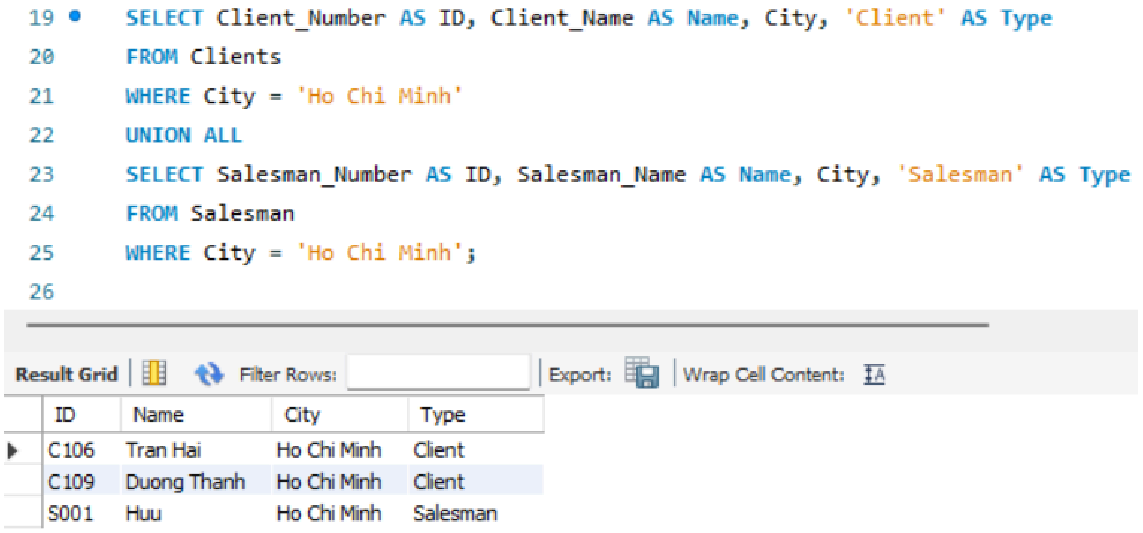
3. SQL statement returns the Ho Chi Minh cities (only distinct values) from the "Clients" and the "salesman" table.



```sql
SELECT Client_Number AS ID, Client_Name AS Name, City, 'Client' AS Type
FROM Clients
WHERE City = 'Ho Chi Minh'
UNION
SELECT Salesman_Number AS ID, Salesman_Name AS Name, City, 'Salesman' AS Type
FROM Salesman
WHERE City = 'Ho Chi Minh';
```

ID	Name	City	Type
C106	Tran Hai	Ho Chi Minh	Client
C109	Duong Thanh	Ho Chi Minh	Client
S001	Huu	Ho Chi Minh	Salesman

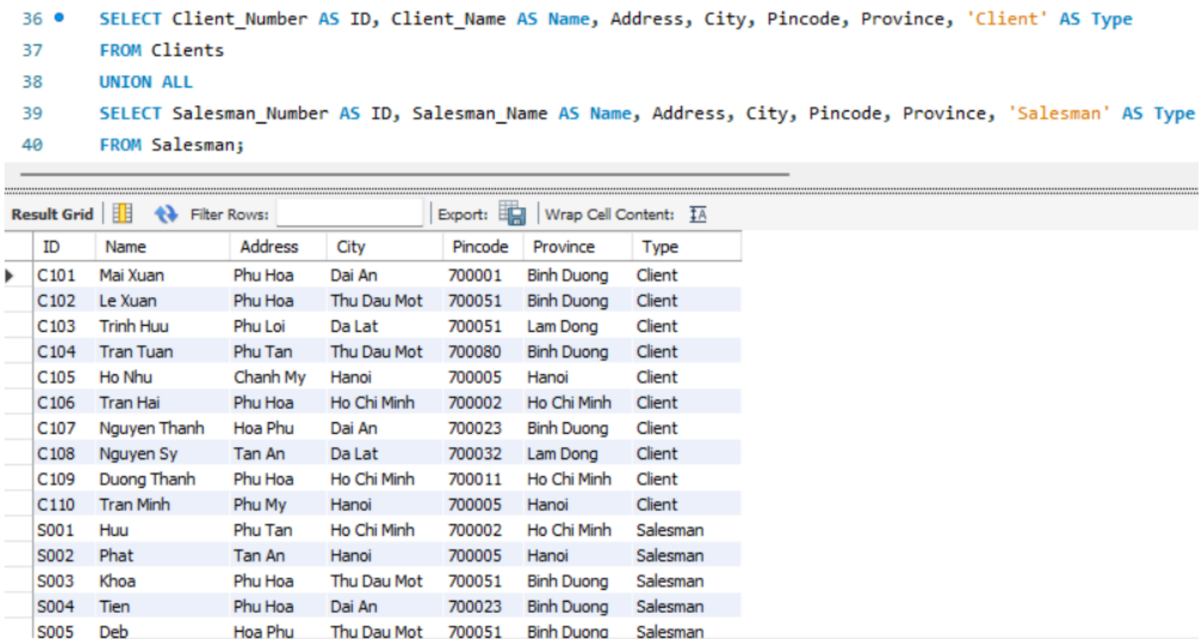
4. SQL statement returns the Ho Chi Minh cities (duplicate values also) from the "Clients" and the "salesman" table.



```sql
SELECT Client_Number AS ID, Client_Name AS Name, City, 'Client' AS Type
FROM Clients
WHERE City = 'Ho Chi Minh'
UNION ALL
SELECT Salesman_Number AS ID, Salesman_Name AS Name, City, 'Salesman' AS Type
FROM Salesman
WHERE City = 'Ho Chi Minh';
```

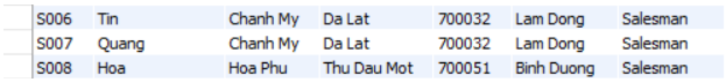
ID	Name	City	Type
C106	Tran Hai	Ho Chi Minh	Client
C109	Duong Thanh	Ho Chi Minh	Client
S001	Huu	Ho Chi Minh	Salesman

5. SQL statement lists all Clients and salesman.



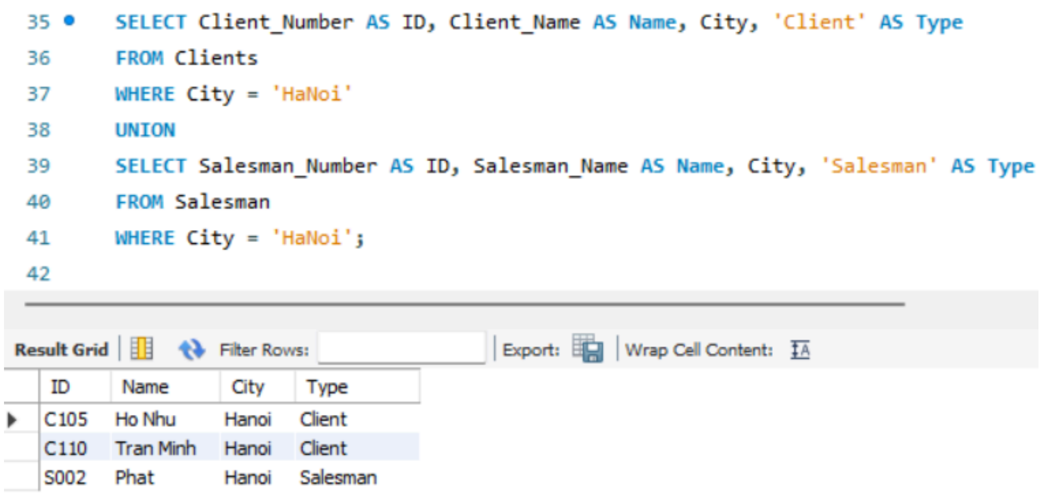
```sql
SELECT Client_Number AS ID, Client_Name AS Name, Address, City, Pincode, Province, 'Client' AS Type
FROM Clients
UNION ALL
SELECT Salesman_Number AS ID, Salesman_Name AS Name, Address, City, Pincode, Province, 'Salesman' AS Type
FROM Salesman;
```

ID	Name	Address	City	Pincode	Province	Type
C101	Mai Xuan	Phu Hoa	Dai An	700001	Binh Duong	Client
C102	Le Xuan	Phu Hoa	Thu Dau Mot	700051	Binh Duong	Client
C103	Trinh Huu	Phu Loi	Da Lat	700051	Lam Dong	Client
C104	Tran Tuan	Phu Tan	Thu Dau Mot	700080	Binh Duong	Client
C105	Ho Nhu	Chanh My	Hanoi	700005	Hanoi	Client
C106	Tran Hai	Phu Hoa	Ho Chi Minh	700002	Ho Chi Minh	Client
C107	Nguyen Thanh	Hoa Phu	Dai An	700023	Binh Duong	Client
C108	Nguyen Sy	Tan An	Da Lat	700032	Lam Dong	Client
C109	Duong Thanh	Phu Hoa	Ho Chi Minh	700011	Ho Chi Minh	Client
C110	Tran Minh	Phu My	Hanoi	700005	Hanoi	Client
S001	Huu	Phu Tan	Ho Chi Minh	700002	Ho Chi Minh	Salesman
S002	Phat	Tan An	Hanoi	700005	Hanoi	Salesman
S003	Khoa	Phu Hoa	Thu Dau Mot	700051	Binh Duong	Salesman
S004	Tien	Phu Hoa	Dai An	700023	Binh Duong	Salesman
S005	Deb	Hoa Phu	Thu Dau Mot	700051	Binh Duong	Salesman



S006	Tin	Chanh My	Da Lat	700032	Lam Dong	Salesman
S007	Quang	Chanh My	Da Lat	700032	Lam Dong	Salesman
S008	Hoa	Hoa Phu	Thu Dau Mot	700051	Binh Duong	Salesman

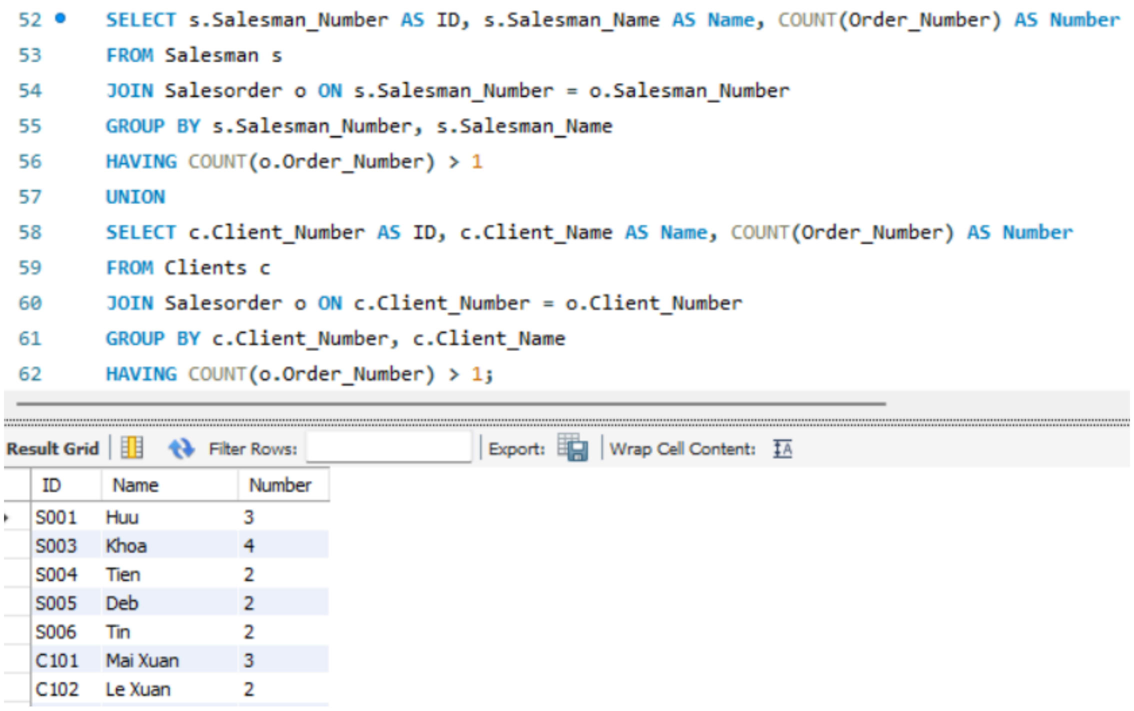
6. Write a SQL query to find all salesman and clients located in the city of Ha Noi on a table with information: ID, Name, City and Type.



```sql
SELECT Client_Number AS ID, Client_Name AS Name, City, 'Client' AS Type
FROM Clients
WHERE City = 'HaNoi'
UNION
SELECT Salesman_Number AS ID, Salesman_Name AS Name, City, 'Salesman' AS Type
FROM Salesman
WHERE City = 'HaNoi';
```

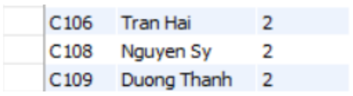
ID	Name	City	Type
C105	Ho Nhu	Hanoi	Client
C110	Tran Minh	Hanoi	Client
S002	Phat	Hanoi	Salesman

7. Write a SQL query to find those salesman and clients who have placed more than one order. Return ID, name and order by ID.



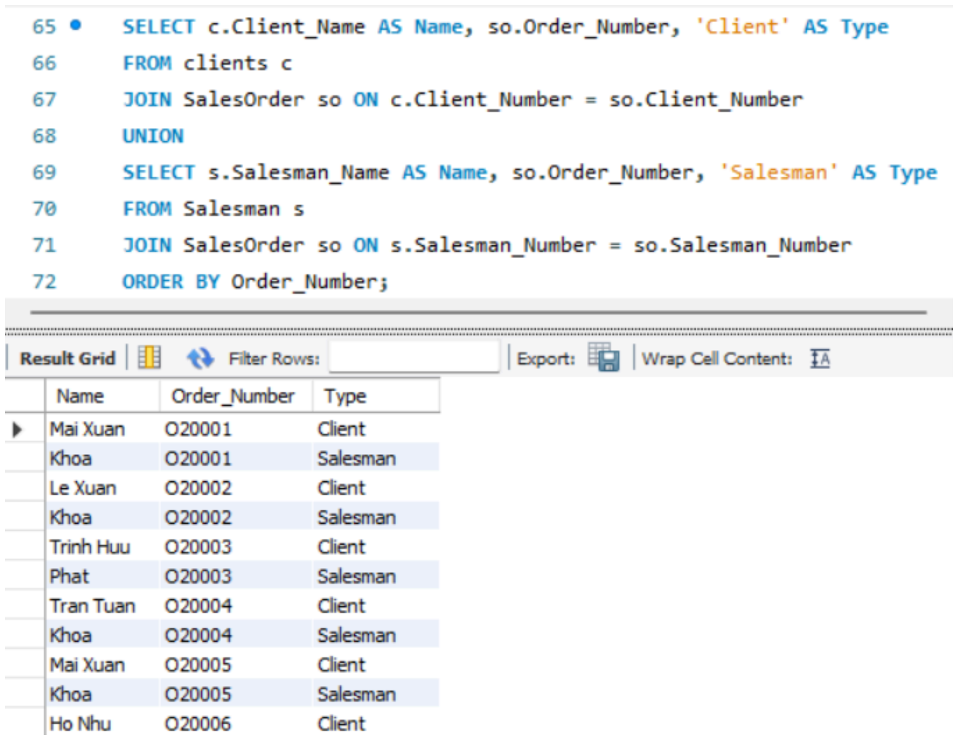
```sql
SELECT s.Salesman_Number AS ID, s.Salesman_Name AS Name, COUNT(Order_Number) AS Number
FROM Salesman s
JOIN Salesorder o ON s.Salesman_Number = o.Salesman_Number
GROUP BY s.Salesman_Number, s.Salesman_Name
HAVING COUNT(o.Order_Number) > 1
UNION
SELECT c.Client_Number AS ID, c.Client_Name AS Name, COUNT(Order_Number) AS Number
FROM Clients c
JOIN Salesorder o ON c.Client_Number = o.Client_Number
GROUP BY c.Client_Number, c.Client_Name
HAVING COUNT(o.Order_Number) > 1;
```

Result Grid
Filter Rows:
Export:
Wrap Cell Content:

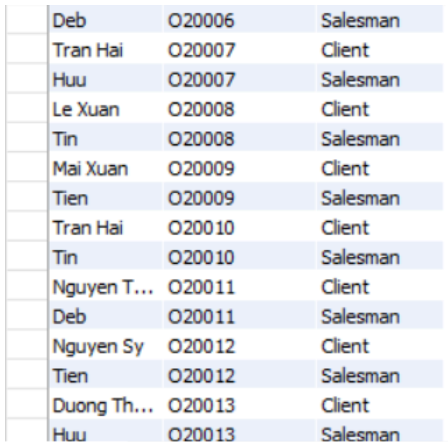
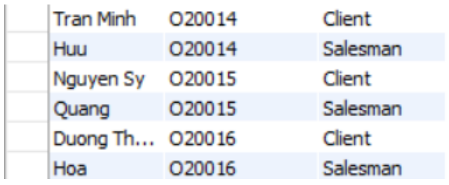


ID	Name	Number
S001	Huu	3
S003	Khoa	4
S004	Tien	2
S005	Deb	2
S006	Tin	2
C101	Mai Xuan	3
C102	Le Xuan	2
C106	Tran Hai	2
C108	Nguyen Sy	2
C109	Duong Thanh	2

8. Retrieve Name, Order Number (order by order number) and Type of client or salesman with the client names who placed orders and the salesman names who processed those orders.

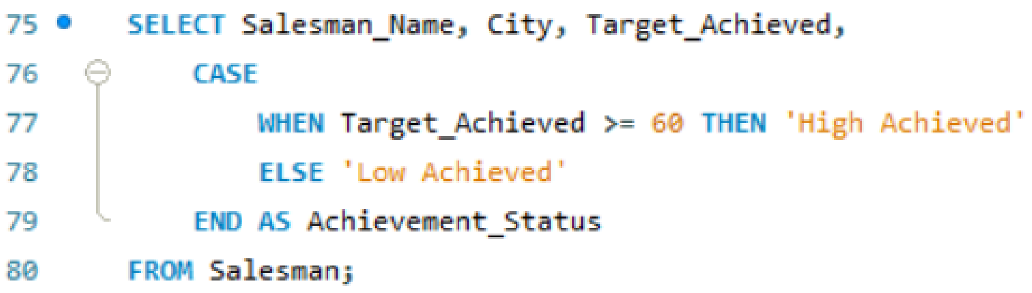


```sql
SELECT c.Client_Name AS Name, so.Order_Number, 'Client' AS Type
FROM clients c
JOIN SalesOrder so ON c.Client_Number = so.Client_Number
UNION
SELECT s.Salesman_Name AS Name, so.Order_Number, 'Salesman' AS Type
FROM Salesman s
JOIN SalesOrder so ON s.Salesman_Number = so.Salesman_Number
ORDER BY Order_Number;
```

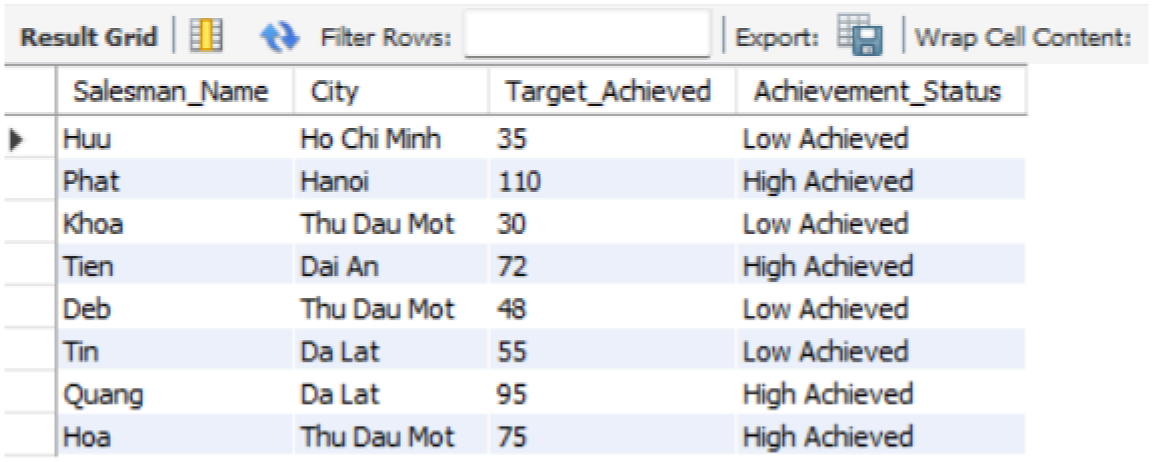



Name	Order_Number	Type
Mai Xuan	O20001	Client
Khoa	O20001	Salesman
Le Xuan	O20002	Client
Khoa	O20002	Salesman
Trinh Huu	O20003	Client
Phat	O20003	Salesman
Tran Tuan	O20004	Client
Khoa	O20004	Salesman
Mai Xuan	O20005	Client
Khoa	O20005	Salesman
Ho Nhu	O20006	Client
Deb	O20006	Salesman
Tran Hai	O20007	Client
Huu	O20007	Salesman
Le Xuan	O20008	Client
Tin	O20008	Salesman
Mai Xuan	O20009	Client
Tien	O20009	Salesman
Tran Hai	O20010	Client
Tin	O20010	Salesman
Nguyen T...	O20011	Client
Deb	O20011	Salesman
Nguyen Sy	O20012	Client
Tien	O20012	Salesman
Duong Th...	O20013	Client
Huu	O20013	Salesman
Tran Minh	O20014	Client
Huu	O20014	Salesman
Nguyen Sy	O20015	Client
Quang	O20015	Salesman
Duong Th...	O20016	Client
Hoa	O20016	Salesman

9. Write a SQL query to create a union of two queries that shows the salesman, cities, and target_Achieved of all salesmen. Those with a target of 60 or greater will have the words 'High Achieved', while the others will have the words 'Low Achieved'.

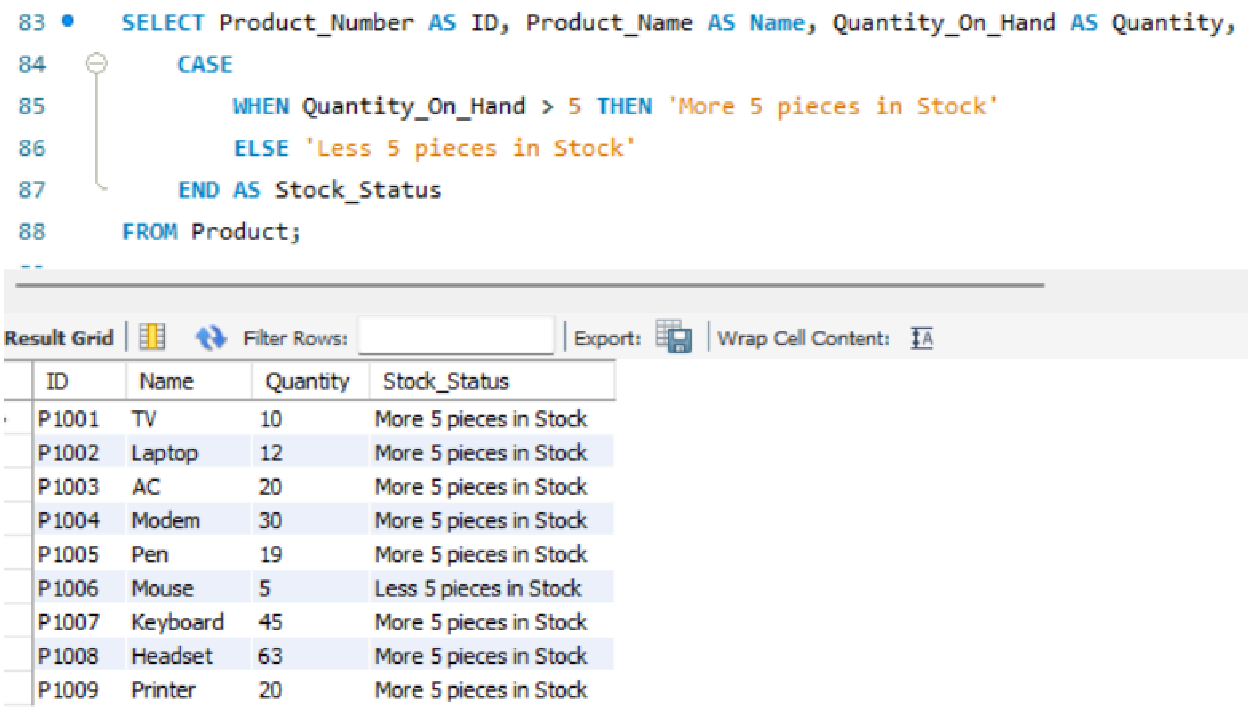


```sql
SELECT Salesman_Name, City, Target_Achieved,
    CASE
        WHEN Target_Achieved >= 60 THEN 'High Achieved'
        ELSE 'Low Achieved'
    END AS Achievement_Status
FROM Salesman;
```



Salesman_Name	City	Target_Achieved	Achievement_Status
Huu	Ho Chi Minh	35	Low Achieved
Phat	Hanoi	110	High Achieved
Khoa	Thu Dau Mot	30	Low Achieved
Tien	Dai An	72	High Achieved
Deb	Thu Dau Mot	48	Low Achieved
Tin	Da Lat	55	Low Achieved
Quang	Da Lat	95	High Achieved
Hoa	Thu Dau Mot	75	High Achieved

10. Write query to creates lists all products (Product_Number AS ID, Product_Name AS Name, Quantity_On_Hand AS Quantity) and their stock status. Products with a positive quantity in stock are labeled as 'More 5 pieces in Stock'. Products with zero quantity are labeled as 'Less 5 pieces in Stock'.



```
SELECT Product_Number AS ID, Product_Name AS Name, Quantity_On_Hand AS Quantity,
    CASE
        WHEN Quantity_On_Hand > 5 THEN 'More 5 pieces in Stock'
        ELSE 'Less 5 pieces in Stock'
    END AS Stock_Status
FROM Product;
```

ID	Name	Quantity	Stock_Status
P1001	TV	10	More 5 pieces in Stock
P1002	Laptop	12	More 5 pieces in Stock
P1003	AC	20	More 5 pieces in Stock
P1004	Modem	30	More 5 pieces in Stock
P1005	Pen	19	More 5 pieces in Stock
P1006	Mouse	5	Less 5 pieces in Stock
P1007	Keyboard	45	More 5 pieces in Stock
P1008	Headset	63	More 5 pieces in Stock
P1009	Printer	20	More 5 pieces in Stock

b) *STORE PROCEDURES*

Statements:

1. Create a procedure stores

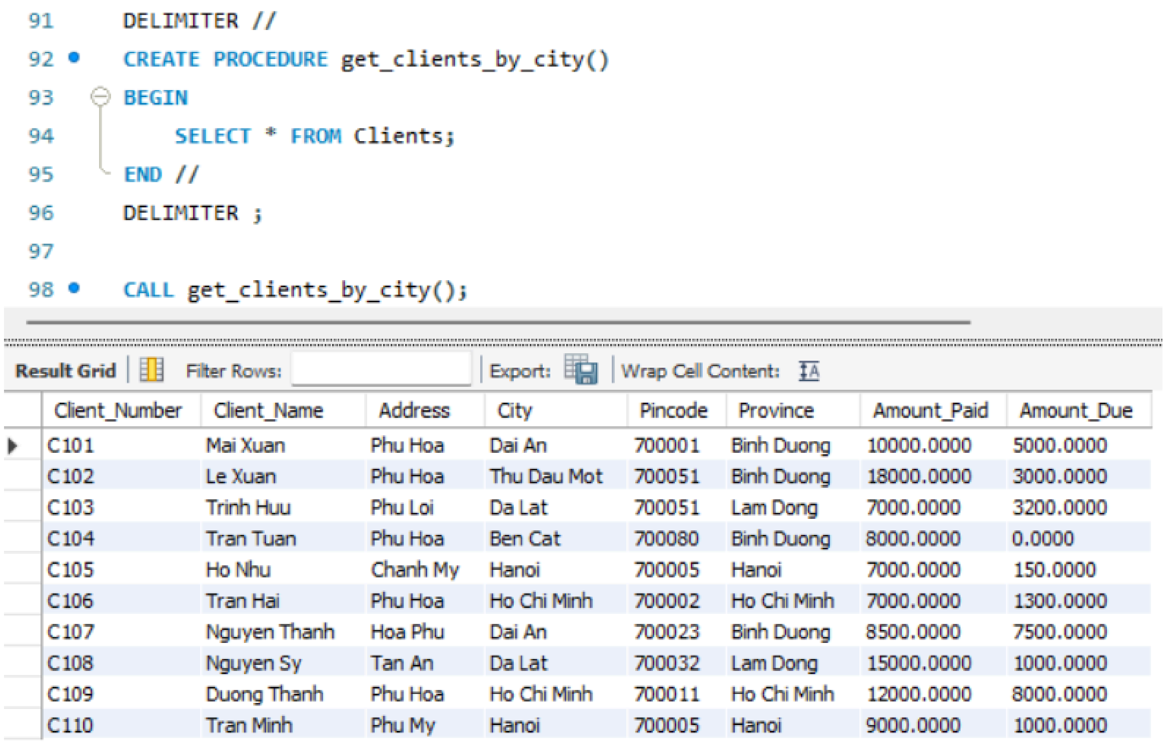
   Delimiter $$

   CREATE PROCEDURE sp_name () .../
   CREATE PROCEDURE sp_name ([IN] param_name type).../
   CREATE PROCEDURE sp_name ([OUT] param_name type).../
   CREATE PROCEDURE sp_name ([INOUT] param_name type)...
   Begin

Select statements;
End$$
Delimiter ;

2. Call a procedure stores
   Call *name_store* (*value of parameter if have*);
3. Drop a procedure stores
   Drop procedure *name_store*;

11. Create a procedure stores get_clients _by_city () saves the all Clients in table. Then Call procedure stores.



```
DELIMITER //
CREATE PROCEDURE get_clients_by_city()
BEGIN
    SELECT * FROM Clients;
END //
DELIMITER ;

CALL get_clients_by_city();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_Number	Client_Name	Address	City	Pincode	Province	Amount_Paid	Amount_Due
C101	Mai Xuan	Phu Hoa	Dai An	700001	Binh Duong	10000.0000	5000.0000
C102	Le Xuan	Phu Hoa	Thu Dau Mot	700051	Binh Duong	18000.0000	3000.0000
C103	Trinh Huu	Phu Loi	Da Lat	700051	Lam Dong	7000.0000	3200.0000
C104	Tran Tuan	Phu Hoa	Ben Cat	700080	Binh Duong	8000.0000	0.0000
C105	Ho Nhu	Chanh My	Hanoi	700005	Hanoi	7000.0000	150.0000
C106	Tran Hai	Phu Hoa	Ho Chi Minh	700002	Ho Chi Minh	7000.0000	1300.0000
C107	Nguyen Thanh	Hoa Phu	Dai An	700023	Binh Duong	8500.0000	7500.0000
C108	Nguyen Sy	Tan An	Da Lat	700032	Lam Dong	15000.0000	1000.0000
C109	Duong Thanh	Phu Hoa	Ho Chi Minh	700011	Ho Chi Minh	12000.0000	8000.0000
C110	Tran Minh	Phu My	Hanoi	700005	Hanoi	9000.0000	1000.0000

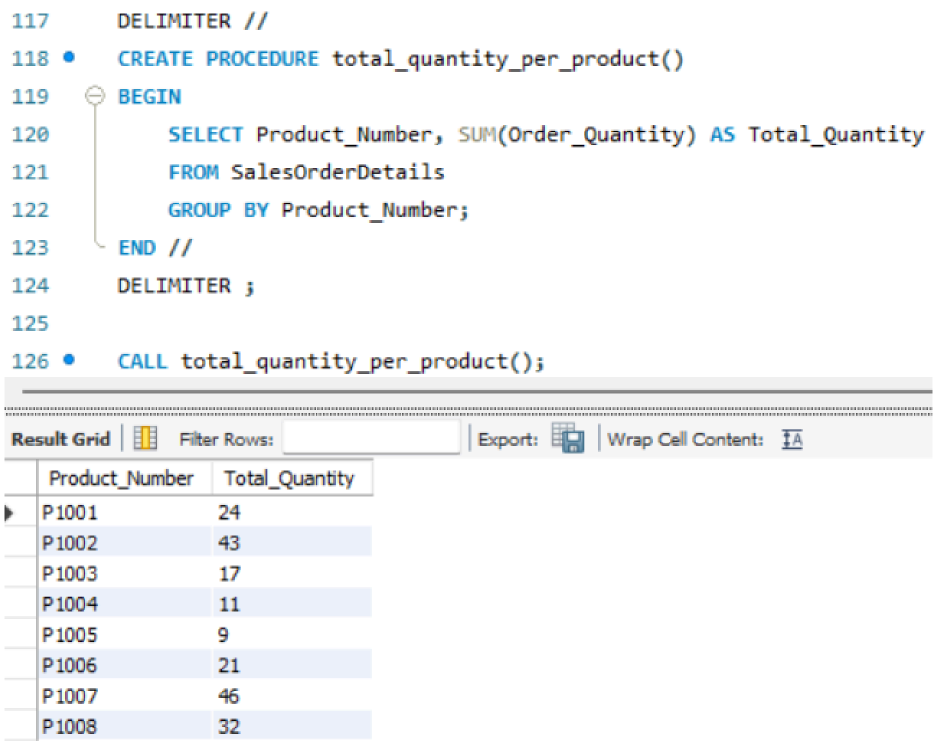
12. Drop get_clients _by_city () procedure stores.

```
DROP PROCEDURE IF EXISTS get_clients_by_city;
```

13. Create a stored procedure to update the delivery status for a given order number. Change value delivery status of order number “O20006” and “O20008” to “On Way”.

```
DELIMITER //
CREATE PROCEDURE update_delivery_status(IN p_order_number VARCHAR(15), IN p_status CHAR(15))
BEGIN
    UPDATE SalesOrder
    SET Delivery_Status = p_status
    WHERE Order_Number = p_order_number;
END //
DELIMITER ;

CALL update_delivery_status('O20006', 'On Way');
CALL update_delivery_status('O20008', 'On Way');
```

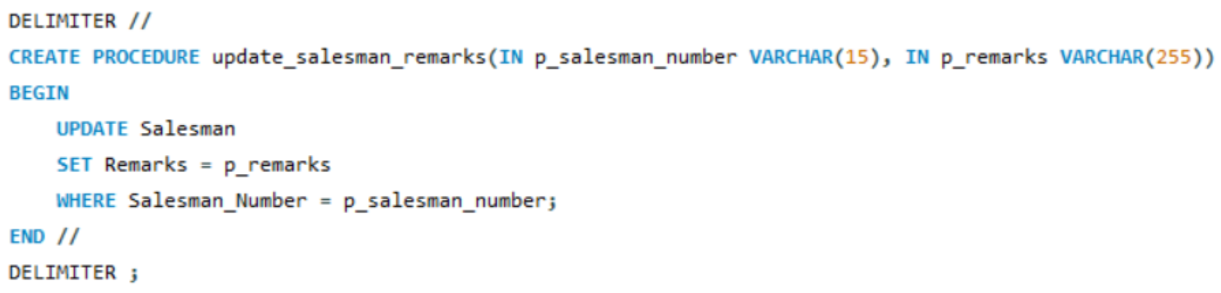
14. Create a stored procedure to retrieve the total quantity for each product.



```
DELIMITER //
CREATE PROCEDURE total_quantity_per_product()
BEGIN
    SELECT Product_Number, SUM(Order_Quantity) AS Total_Quantity
    FROM SalesOrderDetails
    GROUP BY Product_Number;
END //
DELIMITER ;

CALL total_quantity_per_product();
```

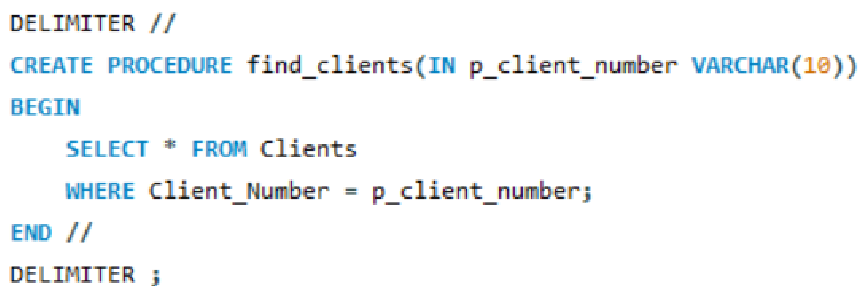
Product_Number	Total_Quantity
P1001	24
P1002	43
P1003	17
P1004	11
P1005	9
P1006	21
P1007	46
P1008	32

15. Create a stored procedure to update the remarks for a specific salesman.



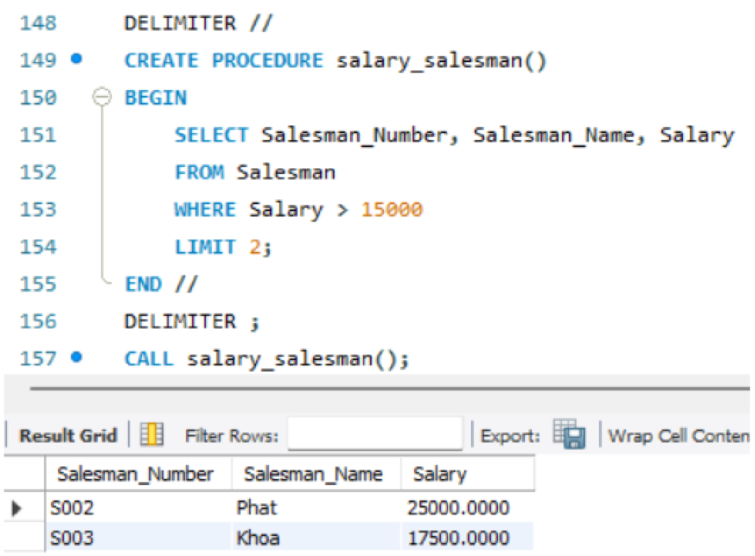
```
DELIMITER //
CREATE PROCEDURE update_salesman_remarks(IN p_salesman_number VARCHAR(15), IN p_remarks VARCHAR(255))
BEGIN
    UPDATE Salesman
    SET Remarks = p_remarks
    WHERE Salesman_Number = p_salesman_number;
END //
DELIMITER ;
```

16. Create a procedure that stores find_clients() saves all of clients and can call each client by client_number.



```
DELIMITER //
CREATE PROCEDURE find_clients(IN p_client_number VARCHAR(10))
BEGIN
    SELECT * FROM Clients
    WHERE Client_Number = p_client_number;
END //
DELIMITER ;
```

17. Creating a procedure stores salary_salesman() saves all of the clients (salesman_number, salesman_name, salary) having a salary >15000. Then execute the first 2 rows or the first 4 rows from the salesman table.

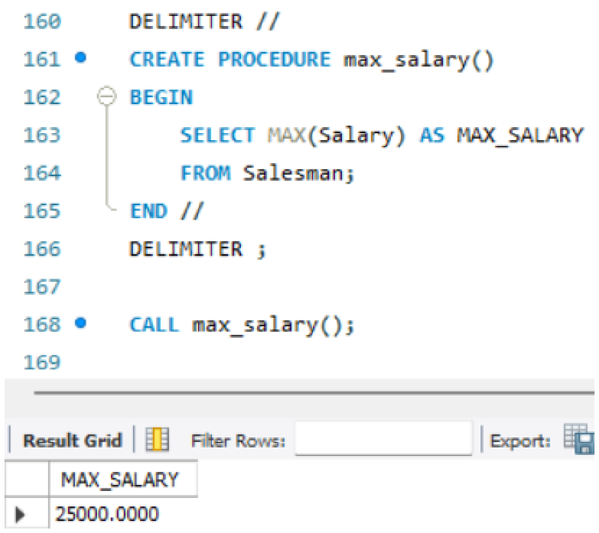


```sql
DELIMITER //
CREATE PROCEDURE salary_salesman()
BEGIN
    SELECT Salesman_Number, Salesman_Name, Salary
    FROM Salesman
    WHERE Salary > 15000
    LIMIT 2;
END //
DELIMITER ;
CALL salary_salesman();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

Salesman_Number	Salesman_Name	Salary
S002	Phat	25000.0000
S003	Khoa	17500.0000

18. Procedure MySQL MAX() function retrieves maximum salary from MAX_SALARY of salary table.

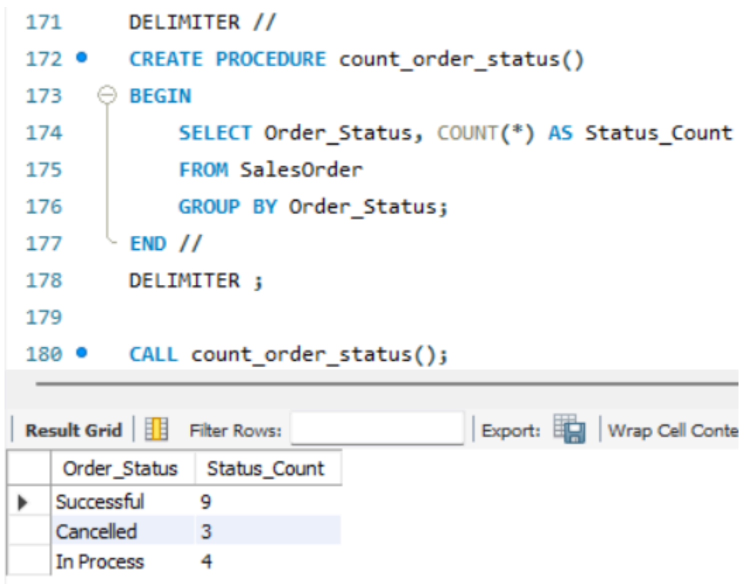


```sql
DELIMITER //
CREATE PROCEDURE max_salary()
BEGIN
    SELECT MAX(Salary) AS MAX_SALARY
    FROM Salesman;
END //
DELIMITER ;

CALL max_salary();

```

Result Grid | Filter Rows: | Export:

MAX_SALARY
25000.0000

19. Create a procedure stores execute finding the amount of order_status by values order status of sales order table.

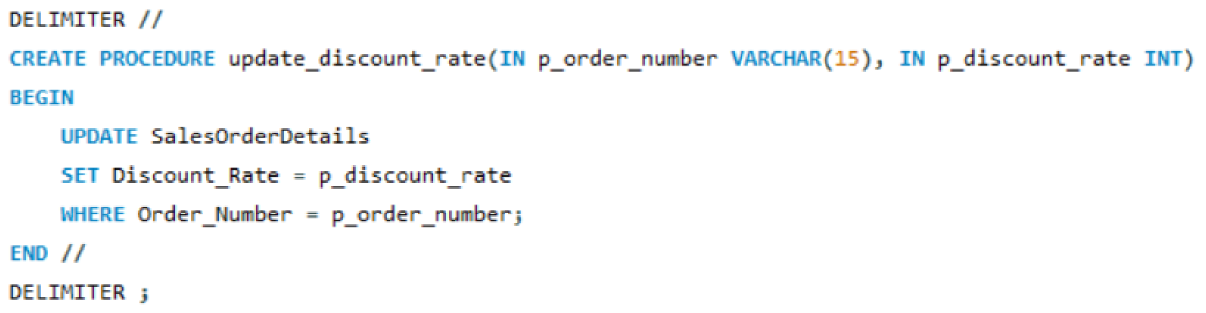


```sql
DELIMITER //
CREATE PROCEDURE count_order_status()
BEGIN
    SELECT Order_Status, COUNT(*) AS Status_Count
    FROM SalesOrder
    GROUP BY Order_Status;
END //
DELIMITER ;

CALL count_order_status();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

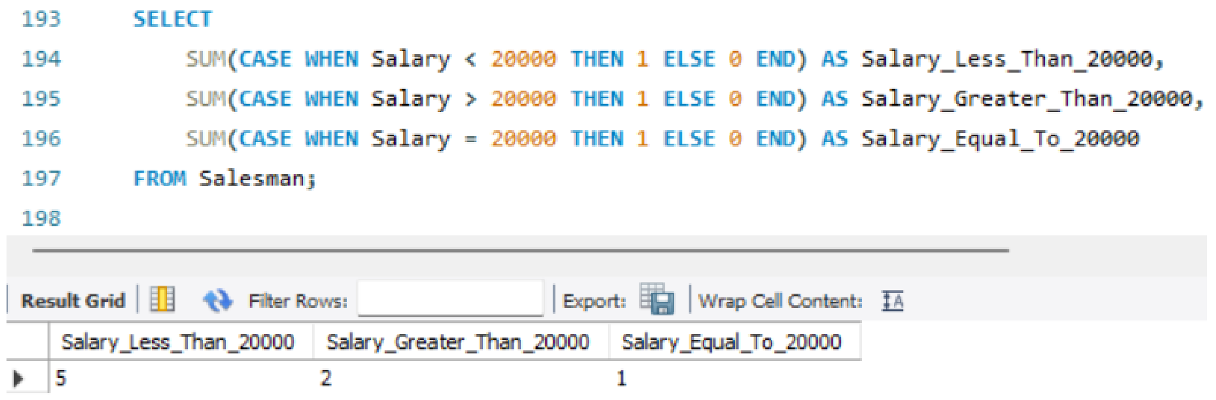
Order_Status	Status_Count
Successful	9
Cancelled	3
In Process	4

20. Create a stored procedure to calculate and update the discount rate for orders.



```
DELIMITER //
CREATE PROCEDURE update_discount_rate(IN p_order_number VARCHAR(15), IN p_discount_rate INT)
BEGIN
    UPDATE SalesOrderDetails
    SET Discount_Rate = p_discount_rate
    WHERE Order_Number = p_order_number;
END //
DELIMITER ;
```

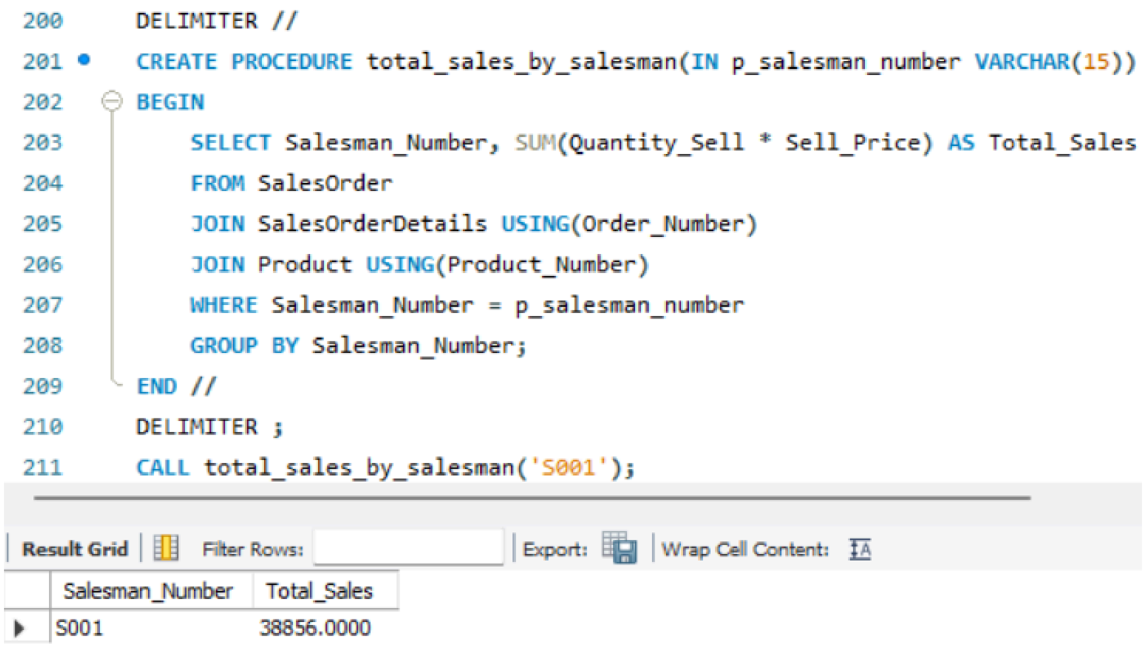
21. Count the number of salesmen with following conditions : SALARY < 20000; SALARY > 20000; SALARY = 20000.



```
SELECT
    SUM(CASE WHEN Salary < 20000 THEN 1 ELSE 0 END) AS Salary_Less_Than_20000,
    SUM(CASE WHEN Salary > 20000 THEN 1 ELSE 0 END) AS Salary_Greater_Than_20000,
    SUM(CASE WHEN Salary = 20000 THEN 1 ELSE 0 END) AS Salary_Equal_To_20000
FROM Salesman;
```

Salary_Less_Than_20000	Salary_Greater_Than_20000	Salary_Equal_To_20000
5	2	1

22. Create a stored procedure to retrieve the total sales for a specific salesman.

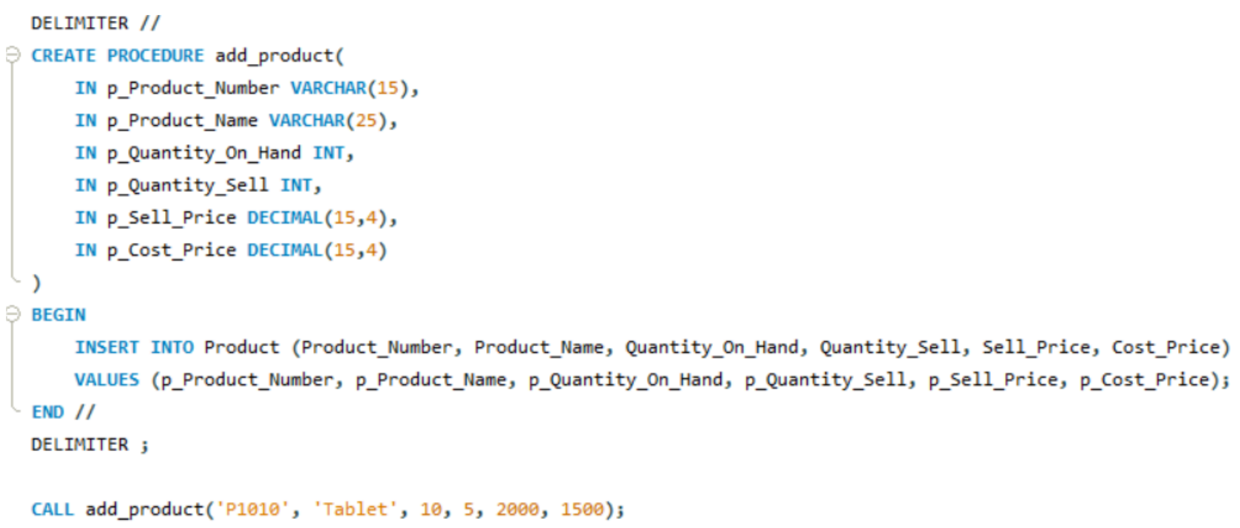


```
DELIMITER //
CREATE PROCEDURE total_sales_by_salesman(IN p_salesman_number VARCHAR(15))
BEGIN
    SELECT Salesman_Number, SUM(Quantity_Sell * Sell_Price) AS Total_Sales
    FROM SalesOrder
    JOIN SalesOrderDetails USING(Order_Number)
    JOIN Product USING(Product_Number)
    WHERE Salesman_Number = p_salesman_number
    GROUP BY Salesman_Number;
END //
DELIMITER ;
CALL total_sales_by_salesman('S001');
```

Salesman_Number	Total_Sales
S001	38856.0000

23. Create a stored procedure to add a new product:

Input variables: Product_Number, Product_Name, Quantity_On_Hand, Quantity_Sell, Sell_Price, Cost_Price.



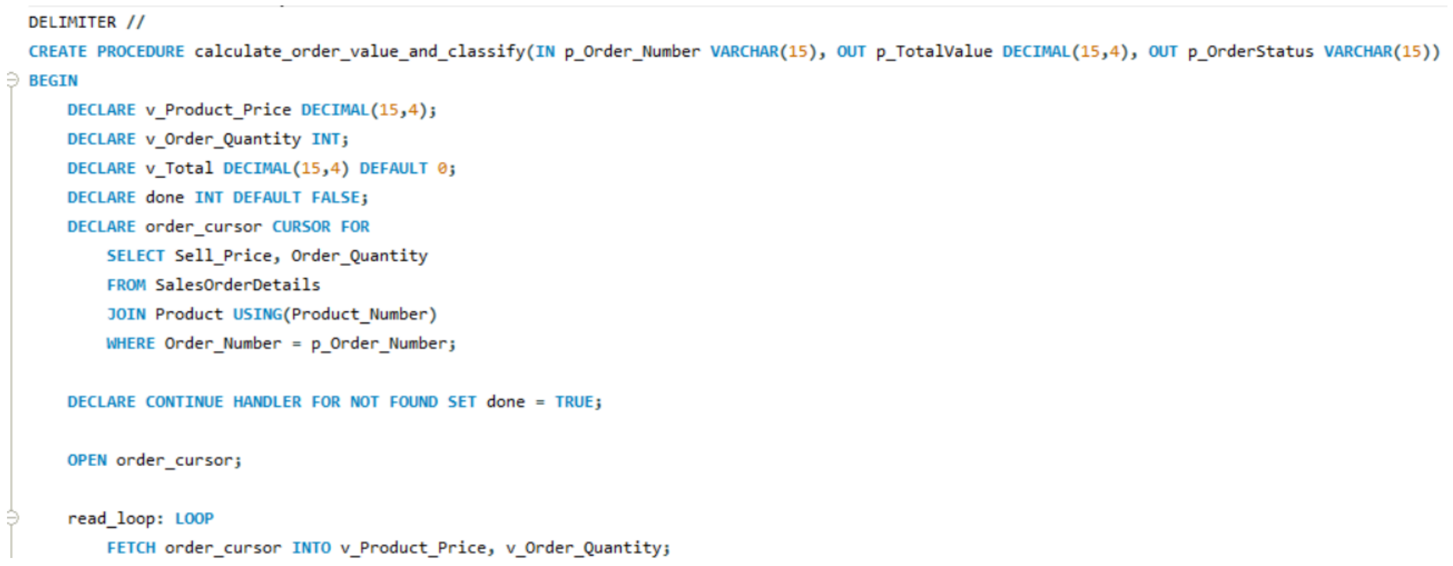
```
DELIMITER //
CREATE PROCEDURE add_product(
    IN p_Product_Number VARCHAR(15),
    IN p_Product_Name VARCHAR(25),
    IN p_Quantity_On_Hand INT,
    IN p_Quantity_Sell INT,
    IN p_Sell_Price DECIMAL(15,4),
    IN p_Cost_Price DECIMAL(15,4)
)
BEGIN
    INSERT INTO Product (Product_Number, Product_Name, Quantity_On_Hand, Quantity_Sell, Sell_Price, Cost_Price)
    VALUES (p_Product_Number, p_Product_Name, p_Quantity_On_Hand, p_Quantity_Sell, p_Sell_Price, p_Cost_Price);
END //
DELIMITER ;

CALL add_product('P1010', 'Tablet', 10, 5, 2000, 1500);
```

24. Create a stored procedure for calculating the total order value and classification:
    - This stored procedure receives the order code (p_Order_Number) và return the total value (p_TotalValue) and order classification (p_OrderStatus).
    - Using the cursor (CURSOR) to browse all the products in the order (SalesOrderDetails).
    - LOOP/While: Browse each product and calculate the total order value.
    - CASE WHEN: Classify orders based on total value:

      Greater than or equal to 10000: "Large"

      Greater than or equal to 5000: "Medium"

      Less than 5000: "Small"



```
DELIMITER //
CREATE PROCEDURE calculate_order_value_and_classify(IN p_Order_Number VARCHAR(15), OUT p_TotalValue DECIMAL(15,4), OUT p_OrderStatus VARCHAR(15))
BEGIN
    DECLARE v_Product_Price DECIMAL(15,4);
    DECLARE v_Order_Quantity INT;
    DECLARE v_Total DECIMAL(15,4) DEFAULT 0;
    DECLARE done INT DEFAULT FALSE;
    DECLARE order_cursor CURSOR FOR
        SELECT Sell_Price, Order_Quantity
        FROM SalesOrderDetails
        JOIN Product USING(Product_Number)
        WHERE Order_Number = p_Order_Number;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN order_cursor;

    read_loop: LOOP
        FETCH order_cursor INTO v_Product_Price, v_Order_Quantity;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SET v_Total = v_Total + (v_Product_Price * v_Order_Quantity);
    END LOOP;

    CLOSE order_cursor;

    SET p_TotalValue = v_Total;

    CASE
        WHEN v_Total >= 10000 THEN SET p_OrderStatus = 'Large';
        WHEN v_Total >= 5000 THEN SET p_OrderStatus = 'Medium';
        ELSE SET p_OrderStatus = 'Small';
    END CASE;
END //
DELIMITER ;

CALL calculate_order_value_and_classify('O20001', @total_value, @order_status);
SELECT @total_value AS TotalValue, @order_status AS OrderStatus;
```