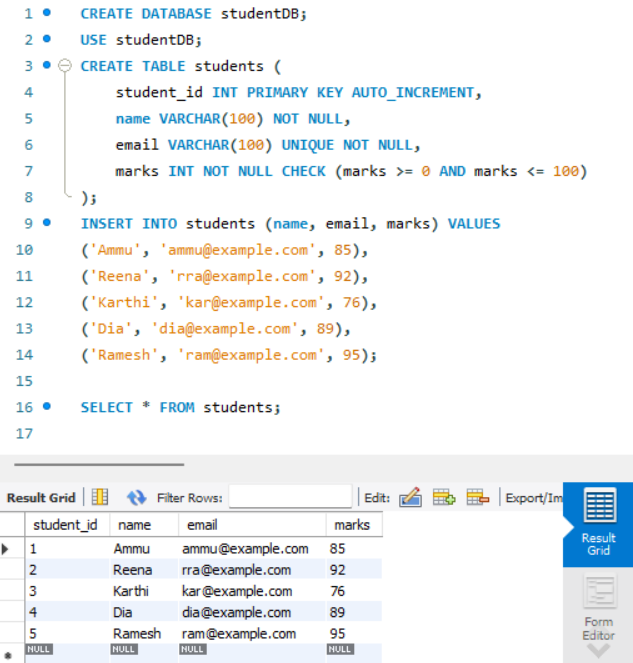
# Assignment Day 4

# Medium-Level Practical SQL Questions

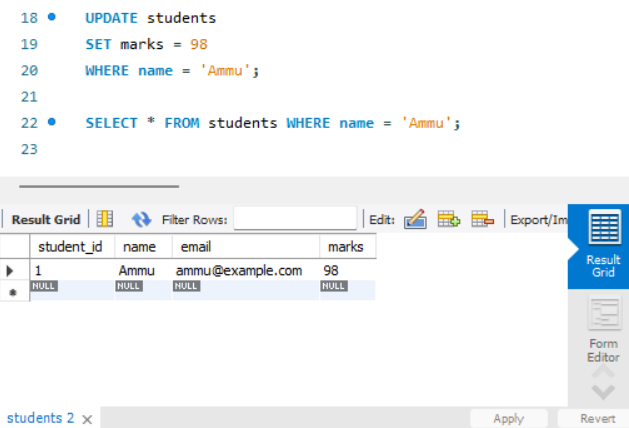
1. Insert and Update with Integrity: Create a 'students' table with constraints (NOT NULL, UNIQUE). Insert 5 records. Then, update a student's marks ensuring data integrity is maintained.

**ANSWER**

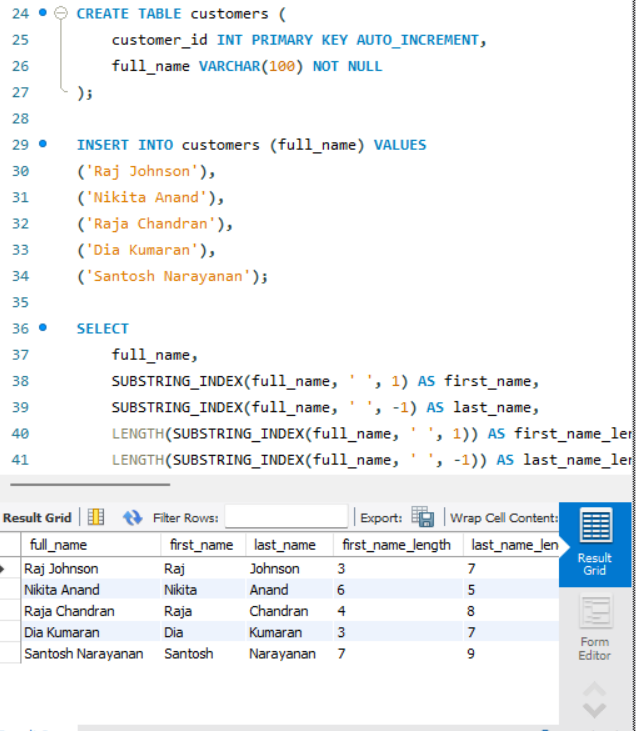
**Create a 'students' table with constraints (NOT NULL, UNIQUE). Insert 5 records.**

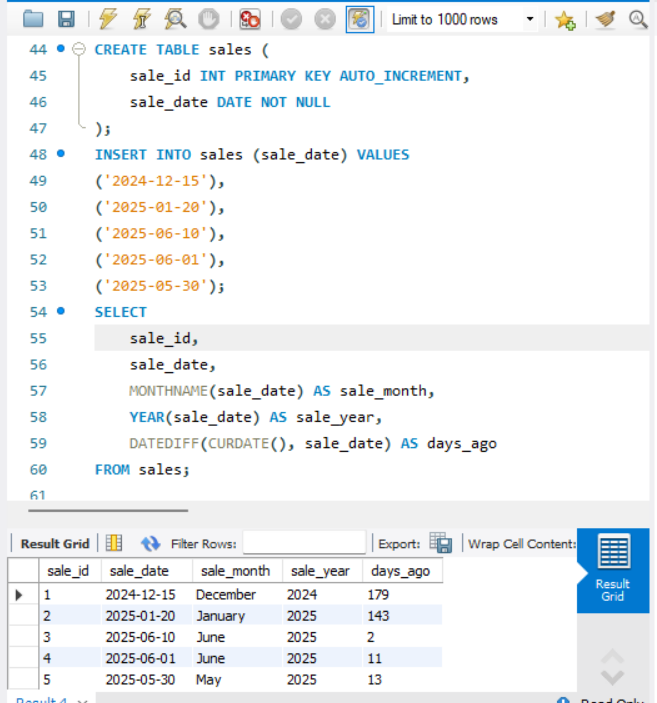


**Update a student's marks ensuring data integrity is maintained.**

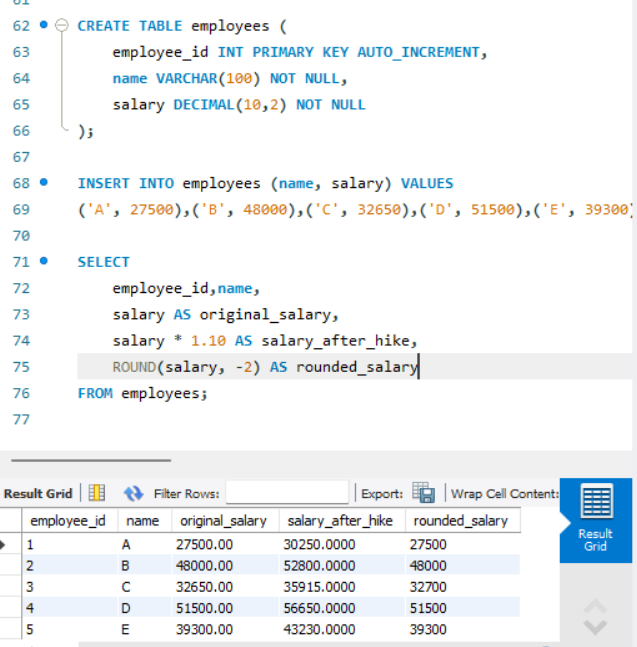


2. String Function Challenge: Given a 'customers' table with a 'full\_name' column, write a query to display: - First name - Last name - Length of each name

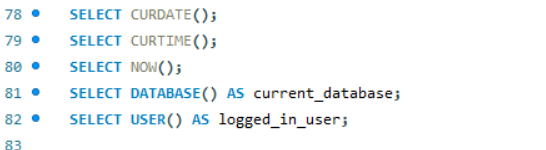


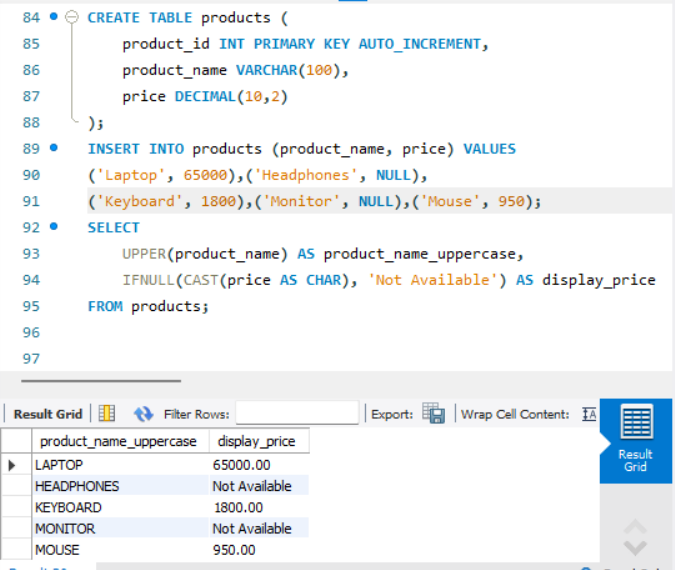
3. Date Function Usage: From a 'sales' table with a 'sale\_date' column, write a query to: - Extract the month name and year - Display how many days ago the sale happened

4. Mathematical Functions on Salary: In an 'employees' table, calculate: - Salary after a 10% hike - Round the salary to the nearest hundred

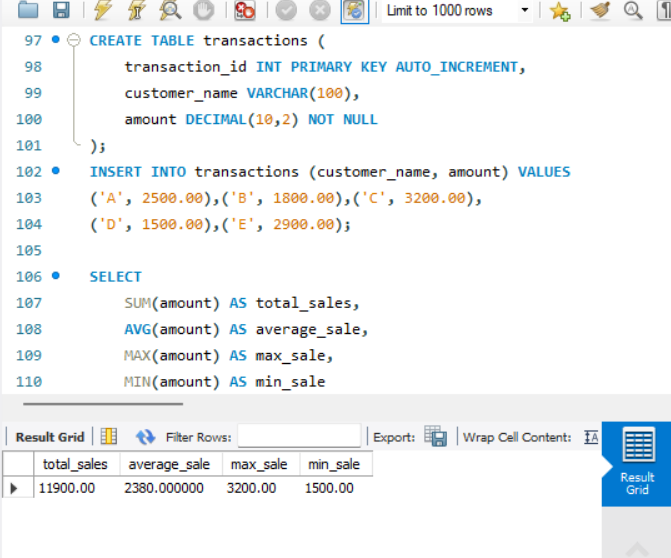


5. System Function Check: Retrieve: - Current date and time - Database name and logged-in user

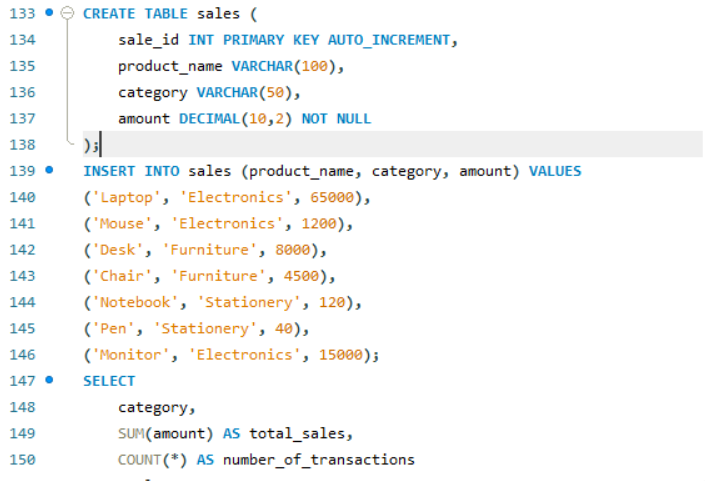


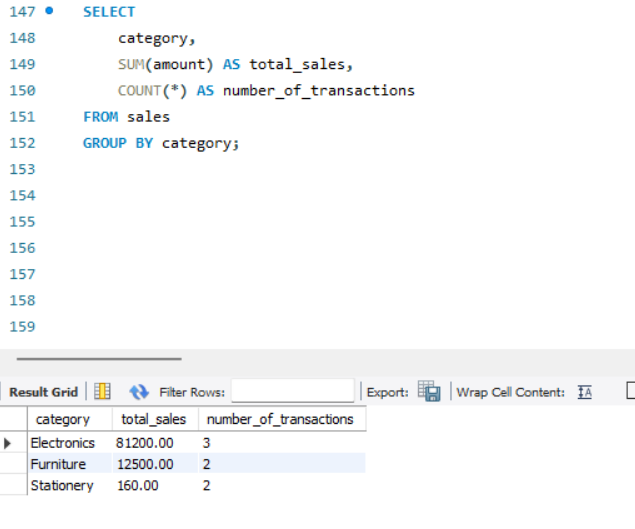
6. Demo: Custom Result Set: From the 'products' table, write a query that: - Returns product name in uppercase - Replaces any NULL prices with 'Not Available'

7. Aggregate Functions Practice: From a 'transactions' table, get: - Total sales - Average sale value - Maximum and minimum sale on a single transaction

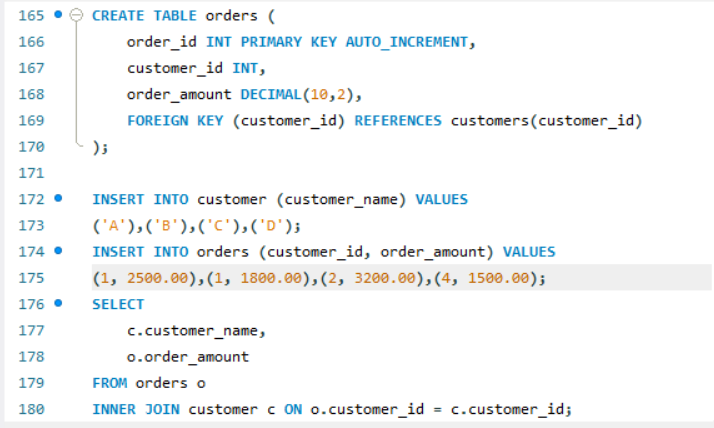


8. Grouping with Aggregation: From a 'sales' table: - Group by product category - Show total sales and number of transactions in each category

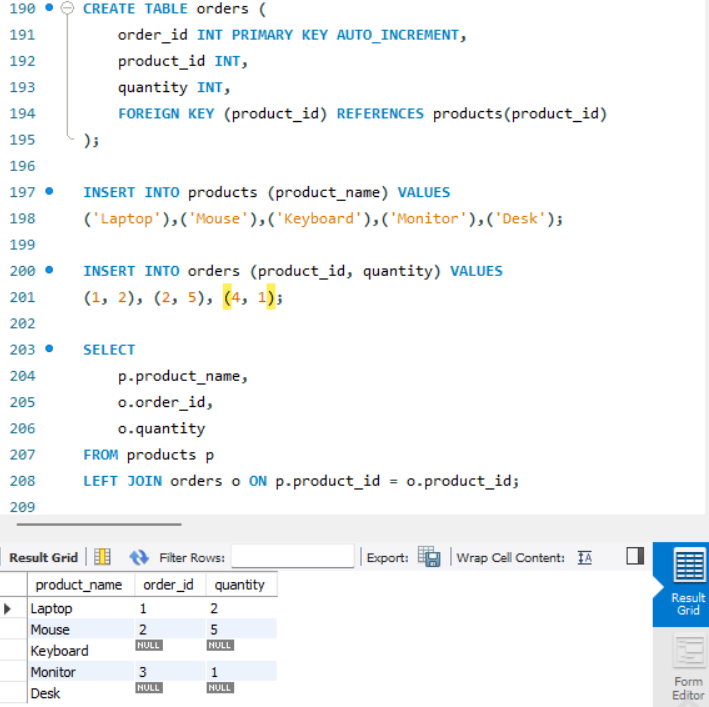




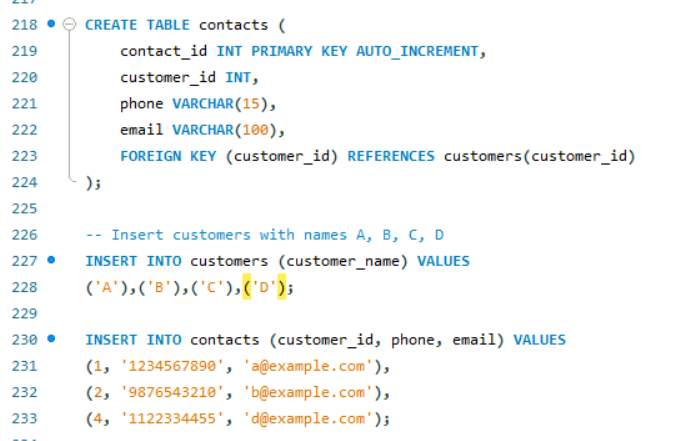
9. Inner Join for Orders and Customers: Join 'orders' and 'customers' to show: - Customer name - Order amount - Only for customers who made orders

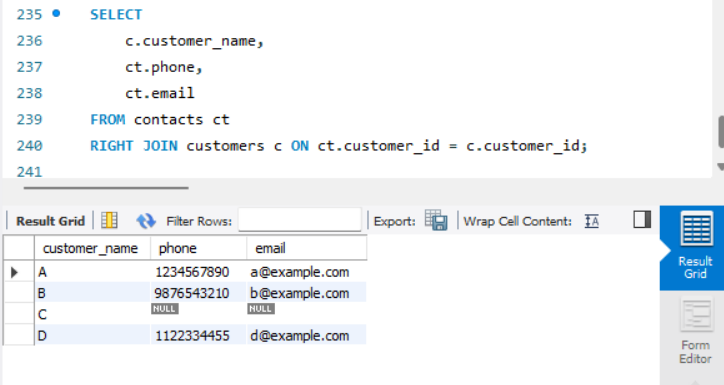


10. Left Join for Products with or without Orders: Show all products with: - Their order details (if available) - Use LEFT JOIN

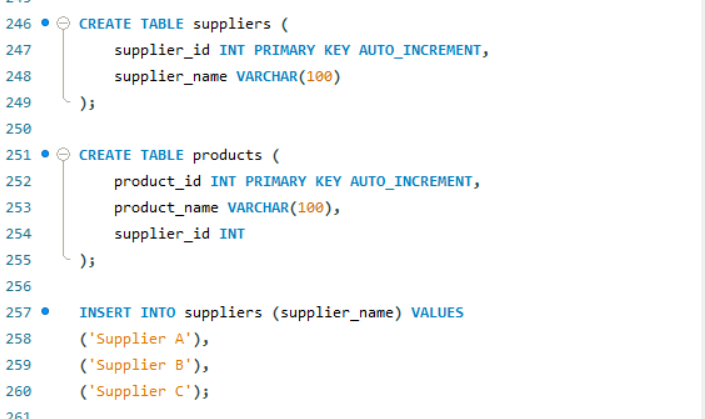


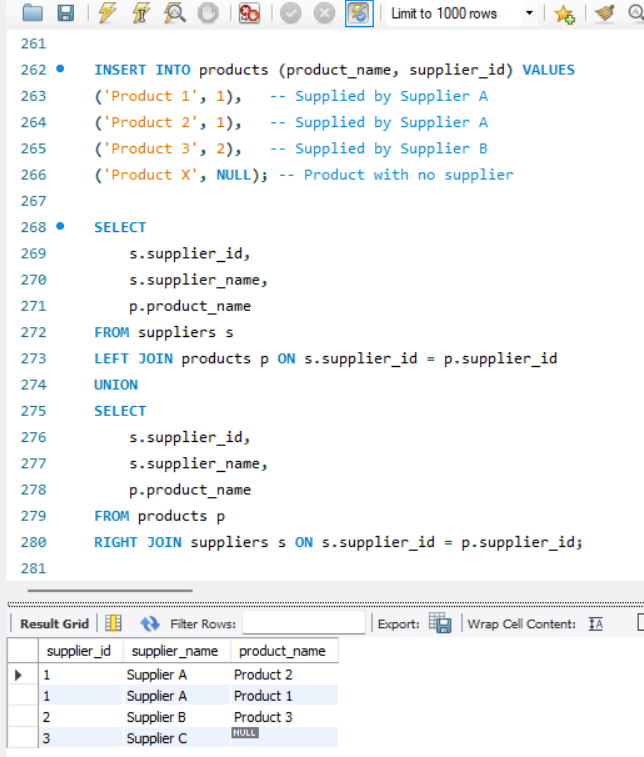
11. Right Join for Customer Contacts: Use a RIGHT JOIN between 'contacts' and 'customers' to display: - All customers, even if they don't have contact info



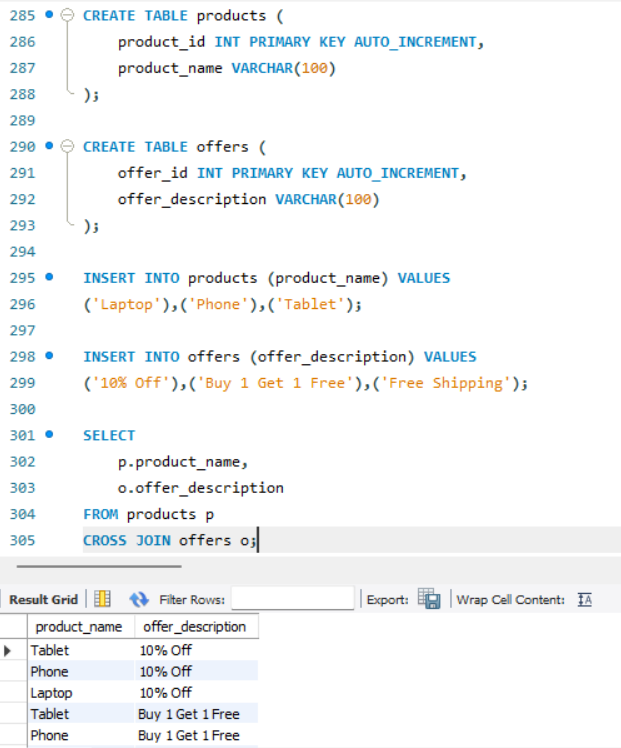


12. Full Outer Join for Suppliers and Products: Use a FULL OUTER JOIN to list: - All suppliers and products - Match supplier to product, or show NULLs where not available

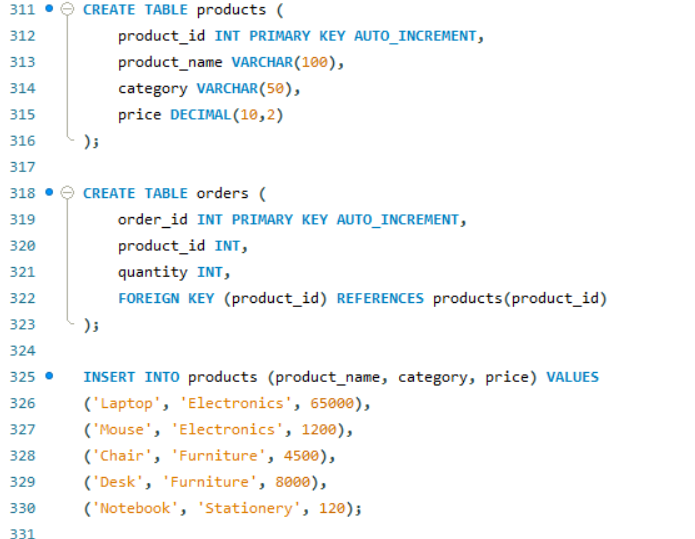


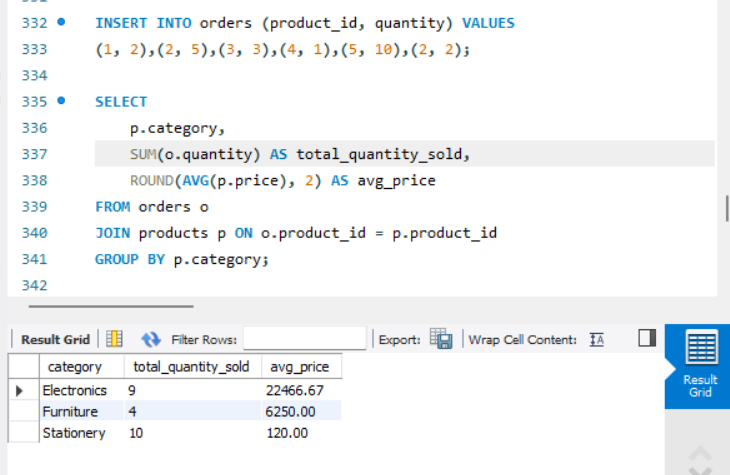


13. Cross Join for Offers: Suppose you have tables 'products' and 'offers'. Write a CROSS JOIN to show: - All possible combinations of products and offers

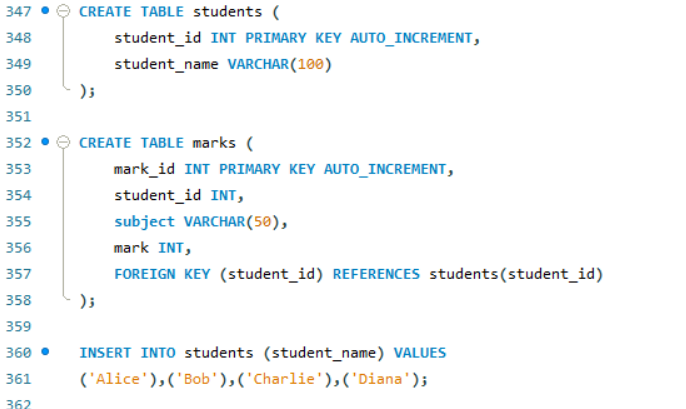


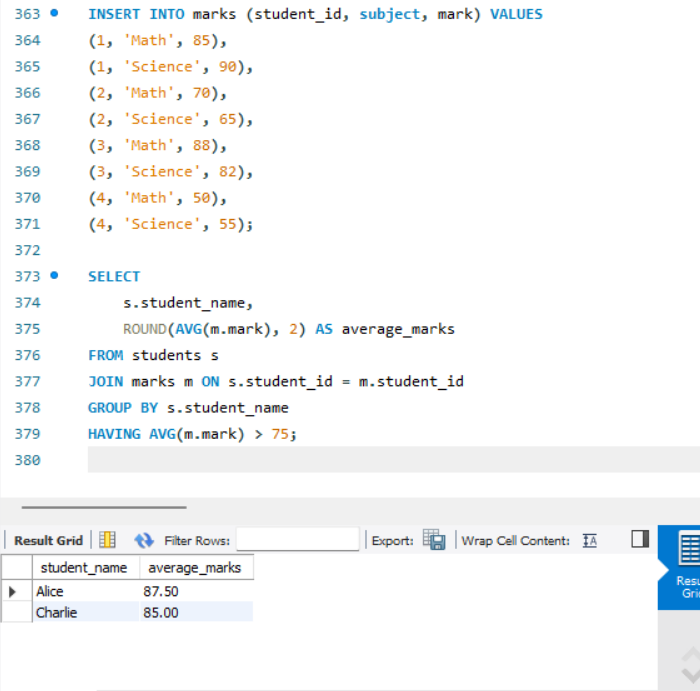
14. Join with Aggregation: Join 'orders' and 'products', then group by product category and: - Show total quantity sold and average price per category





15. Demo: Join with Grouping and Filter: Join 'students' and 'marks' tables. Display: - Student name - Average marks - Filter to show only students with average marks > 75





### **1) Querying Data Using Subqueries**

**Table: employees**

# Assignment Day 5

1) Querying Data by Using Subqueries

2) Querying Data by Using Subqueries Using the EXISTS,

3) Querying Data by Using Subqueries using ANY,

4) Querying Data by Using Subqueries using ALL Keywords

5) Querying Data by Using Subqueries using Using Nested Subqueries

6) Querying Data by Using Subqueries Using Correlated Subqueries

7) Querying Data by Using Subqueries Using UNION,

8) Querying Data by Using Subqueries using INTERSECT,

9) Querying Data by Using Subqueries using EXCEPT,

10)Querying Data by Using Subqueries using MERGE

### **1) Querying Data Using Subqueries**

**Table: employees**

CREATE TABLE employees(

id INT, name VARCHAR(10), dept\_id INT, salary INT

);

INSERT INTO employees VALUES

(1,'Alice',10,5000),(2,'Bob',20,6000),

(3,'Carol',10,7000),(4,'Dave',30,4000);

**Query**: Get employees earning more than the overall average.

SELECT \* FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);

### **2) Using EXISTS**

**Table: departments**

CREATE TABLE departments(

dept\_id INT, dept\_name VARCHAR(10)

);

INSERT INTO departments VALUES (10,'IT'),(20,'HR'),(30,'Sales');

**Query**: Show departments with at least one employee.

SELECT dept\_name FROM departments d

WHERE EXISTS (

SELECT 1 FROM employees e WHERE e.dept\_id = d.dept\_id

);

### **3) Using ANY**

**Query**: Employees whose salary is higher than any IT employee.

SELECT \* FROM employees

WHERE salary > ANY (

SELECT salary FROM employees WHERE dept\_id = 10

);

### **4) Using ALL**

**Query**: Employees who earn more than all employees in dept 10.

SELECT \* FROM employees

WHERE salary > ALL (

SELECT salary FROM employees WHERE dept\_id = 10

);

### **5) Nested Subqueries**

**Query**: Employees earning more than the minimum salary of the highest-paying department.

SELECT \* FROM employees

WHERE salary > (

SELECT MIN(salary) FROM employees WHERE dept\_id =

(SELECT dept\_id FROM employees GROUP BY dept\_id ORDER BY AVG(salary) DESC LIMIT 1)

);

### **6) Correlated Subqueries**

**Query**: Employees earning above their department’s average.

SELECT \* FROM employees e

WHERE salary > (

SELECT AVG(salary) FROM employees WHERE dept\_id = e.dept\_id

);

### **7) Using UNION**

**Query**: List all names of employees and departments.

SELECT name AS item FROM employees

UNION

SELECT dept\_name AS item FROM departments;

### **8) Using INTERSECT**

**Assume** a table of names test:

CREATE TABLE test\_names(name VARCHAR(10));

INSERT INTO test\_names VALUES ('Alice'),('IT'),('X');

**Query**: Names present in both employees and test\_names.

SELECT name FROM employees

INTERSECT

SELECT name FROM test\_names;

### **9) Using EXCEPT**

**Query**: Employee names not in test\_names.

SELECT name FROM employees

EXCEPT

SELECT name FROM test\_names;

### **10) Using MERGE**

**Assume** a target table emp\_archive(id, name, dept\_id, salary).

**Query**: Sync employees table into emp\_archive.

MERGE INTO emp\_archive AS t

USING employees AS s

ON t.id = s.id

WHEN MATCHED THEN

UPDATE SET name = s.name, dept\_id = s.dept\_id, salary = s.salary

WHEN NOT MATCHED THEN

INSERT (id,name,dept\_id,salary) VALUES (s.id,s.name,s.dept\_id,s.salary);