Question 1:

Write an SQL query to calculate the total sales amount for each product, and then display the results in descending order of total sales amount.

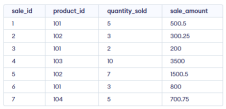
Task:

- Your query should use the SUM function to calculate the total sales amount for each product.

Table Structure:

- sale_id - Unique identifier for each sale.
- product_id - Identifier for the product sold.
- quantity_sold - Number of units of the product sold in each sale.
- sale_amount - The amount of money generated by the sale

Input Tables :



sale_id	product_id	quantity_sold	sale_amount
1	101	5	500.5
2	102	3	300.25
3	101	2	200
4	103	10	3500
5	102	7	1500.5
6	101	3	800
7	104	5	700.75

Title for Question 1: SUM of amount

Solution:

```
-- Query to calculate total sales amount for each product
SELECT product_id, SUM(sale_amount) AS total_sales_amount
FROM sales
GROUP BY product_id
ORDER BY total_sales_amount DESC;
```

TestCases:

S.No	Inputs	Outputs
1	na	product_id | total_sales_amount ---------- | ---------- 103 | 3500.00 102 | 1800.75 101 | 1500.50 104 | 700.75
2		Main.java:1: error: class, interface, or enum expected -- Create the sales table ^ Main.java:9: error: class, interface, or enum expected -- Insert sample data into the sales table ^ Main.java:20: error: class, interface, or enum expected -- Query to calculate total sales amount for each product ^ 3 errors

S.No	Inputs	Outputs
3		Main.java:1: error: class, interface, or enum expected -- Create the sales table ^ Main.java:9: error: class, interface, or enum expected -- Insert sample data into the sales table ^ Main.java:20: error: class, interface, or enum expected -- Query to calculate total sales amount for each product ^ 3 errors
4		Main.java:1: error: class, interface, or enum expected -- Create the sales table ^ Main.java:9: error: class, interface, or enum expected -- Insert sample data into the sales table ^ Main.java:20: error: class, interface, or enum expected -- Query to calculate total sales amount for each product ^ 3 errors
5		Main.java:1: error: class, interface, or enum expected -- Create the sales table ^ Main.java:9: error: class, interface, or enum expected -- Insert sample data into the sales table ^ Main.java:20: error: class, interface, or enum expected -- Query to calculate total sales amount for each product ^ 3 errors
6		Main.java:1: error: class, interface, or enum expected -- Create the sales table ^ Main.java:9: error: class, interface, or enum expected -- Insert sample data into the sales table ^ Main.java:20: error: class, interface, or enum expected -- Query to calculate total sales amount for each product ^ 3 errors

White List:

Black List:

Question 2:

Write an SQL query to calculate the total number of orders for each unique order status and display the results in ascending order of order status.

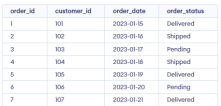
Task:

Your query should use the COUNT function to calculate the number of orders for each status.

Table Structure:

- order_id - Unique identifier for each order.
- customer_id - Identifier for the customer who placed the order.
- order_date - The date when the order was placed.
- order_status - The status of the order (e.g., 'Pending', 'Shipped', 'Delivered', etc.).

Input Table:



order_id	customer_id	order_date	order_status
1	101	2023-01-15	Delivered
2	102	2023-01-16	Shipped
3	103	2023-01-17	Pending
4	104	2023-01-18	Shipped
5	105	2023-01-19	Delivered
6	106	2023-01-20	Pending
7	107	2023-01-21	Delivered

Title for Question 2: count no of order

Solution:

```
-- Query to calculate the total number of orders for each unique order st
SELECT order_status, COUNT(*) AS total_orders
FROM orders
GROUP BY order_status
ORDER BY order_status;
```

TestCases:

S.No	Inputs	Outputs
1	na	order_status | total_orders ---------- | ---------- Delivered | 3 Pending | 2 Shipped | 2
2		
3		
4		
5		
6		

White List:

Black List:

Question 3:

Write an SQL query to calculate the average score for each subject and display the results along with the subject name.

Task:

- Your query should use the AVG function to calculate the average score for each subject.

Table Structure:

- student_id - Unique identifier for each student.
- subject - The subject for which a student has received a score.
- score - The score received by the student in the subject.

Input Table:

student_id	subject	score
1	Mathematics	90.5
2	Science	78
3	Mathematics	80
4	History	92.5
5	Science	76
6	Mathematics	82
7	History	91.5

Title for Question 3: AVG function to score

Solution:

```
-- Query to calculate the average score for each subject
SELECT
    subject,
    AVG(score) AS average_score
FROM
    student_scores
GROUP BY
    subject;
```

TestCases:

S.No	Inputs	Outputs
1	na	subject | average_score ---------- | ---------- Mathematics | 84.166667 Science | 77.000000 History | 92.000000
2		
3		
4		
5		
6		

White List:

Black List:

Question 4:

Write an SQL query to find the highest total order amount and display the order details for that order, including the order ID, order date, and total amount.

Task:

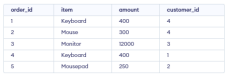
- Your query should use the MAX function to find the highest total order amount.

Table Structure:

- order_id - Unique identifier for each order.

- order_date - The date when the order was placed.
- total_amount - The total amount of the order.

Input Table:



order_id	item	amount	customer_id
1	Keyboard	400	4
2	Mouse	300	4
3	Monitor	12000	3
4	Keyboard	400	1
5	Mousepad	250	2

Title for Question 4: MAX function to amount

Solution:

```
-- Query to find the highest total order amount and its details
SELECT
    order_id,
    order_date,
    total_amount
FROM
    orders
WHERE
total_amount = (SELECT MAX(total_amount) FROM orders);
```

TestCases:

S.No	Inputs	Outputs
1	na	order_id | order_date | total_amount -------- | ---------- | ------------ 102 | 2023-05-15 | 1250.75
2		
3		
4		
5		
6		

White List:

Black List:

Question 5:

Write an SQL query to find the product with the lowest stock quantity and display its details, including the product ID, product name, and stock quantity.

Task:

- Your query should use the MIN function to find the product with the lowest stock quantity.

Table Structure:

- product_id - Unique identifier for each product.
- product_name - The name of the product.
- stock_quantity - The quantity of the product in stock.

Input Table:

product_id	product_name	stock_quantity
101	Laptop	20
102	Smartphone	35
103	Tablet	15
104	Paper Towels	50
105	Headphones	10

Title for Question 5: Min Function 2

Solution:

```
-- Query to find the product with the lowest stock quantity
SELECT
    product_id,
    product_name,
    stock_quantity
FROM
    products
WHERE
    stock_quantity = (SELECT MIN(stock_quantity) FROM products);
```

TestCases:

S.No	Inputs	Outputs
1	na	product_id | product_name | stock_quantity ---------- | ---------- | ---------- 105 | Headphones | 10
2		
3		
4		
5		
6		

White List:

Black List:

Question 6:

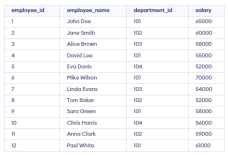
Write an SQL query to retrieve the following information:

- List the employee_name, salary, and department_name for employees who earn a salary greater than $50,000.
- Order the results by salary in descending order.
- Display only the first 10 rows of the result.

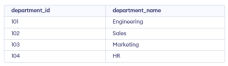
Table Structure:

- employee_id - Unique identifier for each employee.
- employee_name - The name of the employee.
- department_id - Identifier for the department to which the employee belongs.
- salary - The monthly salary of the employee.
- department_id - Unique identifier for each department.
- department_name - The name of the department.

Input Table:



employee_id	employee_name	department_id	salary
1	John Doe	101	65000
2	Jane Smith	102	60000
3	Alice Brown	103	58000
4	David Lee	101	55000
5	Eva Davis	104	52000
6	Mike Wilson	101	70000
7	Linda Evans	103	54000
8	Tom Baker	102	52000
9	Sara Green	101	58000
10	Chris Harris	104	56000
11	Anna Clark	102	59000
12	Paul White	101	61000



department_id	department_name
101	Engineering
102	Sales
103	Marketing
104	HR

Title for Question 6: order the result

Solution:

```
-- Insert sample data into the employees table
INSERT INTO employees (employee_id, employee_name, department_id, salary)
VALUES
    (1, 'John Doe', 101, 65000.0),
    (2, 'Jane Smith', 102, 60000.0),
    (3, 'Alice Brown', 103, 58000.0),
    (4, 'David Lee', 101, 55000.0),
    (5, 'Eva Davis', 104, 52000.0),
    (6, 'Mike Wilson', 101, 70000.0),
    (7, 'Linda Evans', 103, 54000.0),
    (8, 'Tom Baker', 102, 52000.0),
    (9, 'Sara Green', 101, 58000.0),
    (10, 'Chris Harris', 104, 56000.0),
    (11, 'Anna Clark', 102, 59000.0),
    (12, 'Paul White', 101, 61000.0);
```

```
-- Query to retrieve employee details meeting the criteria
SELECT
    e.employee_name,
    e.salary,
    d.department_name
FROM
    employees e
JOIN
    departments d
ON
    e.department_id = d.department_id
WHERE
    e.salary > 50000.0
ORDER BY
    e.salary DESC
LIMIT 10;
```

TestCases:

S.No	Inputs	Outputs
1	na	employee_name | salary | department_name ---------- | -------- | ---------- Mike Wilson | 70000.00 | Engineering John Doe | 65000.00 | Engineering Paul White | 61000.00 | Engineering Jane Smith | 60000.00 | Sales Anna Clark | 59000.00 | Sales Alice Brown | 58000.00 | Marketing Sara Green | 58000.00 | Engineering Chris Harris | 56000.00 | HR David Lee | 55000.00 | Engineering Linda Evans | 54000.00 | Marketing
2		
3		
4		
5		
6		

White List:

Black List:
