SQL Test # 01

Task # 1

Hospital Database Query Task

Given the following tables:

- **Patients**: patient_id, patient_name
- **Doctors**: doctor_id, doctor_name
- **Appointments**: appointment_id, patient_id, doctor_id, appointment_date, treatment_id
- **Treatments**: treatment_id, treatment_name, treatment_cost

Task: Write an SQL query to retrieve the following information:

- patient_name
- doctor_name
- appointment_date
- treatment_name
- treatment_cost

Requirements:

1. Include only appointments from the year 2024.
2. Include only treatments with a cost greater than 500.
3. Sort the results by patient_name in ascending order and then by treatment_cost in descending order.

Patients

patient_id	patient_name
1	Alice Brown
2	Bob Smith
3	Carol White

Doctors

doctor_id	doctor_name
1	Dr. Emily Clark
2	Dr. John Doe
3	Dr. Sarah Lee

Appointments

appointment_id	patient_id	doctor_id	appointment_date	treatment_id
101	1	1	2024-01-15	201
102	2	2	2024-03-10	202
103	1	3	2024-05-22	203
104	3	1	2024-07-10	204

Treatments

treatment_id	treatment_name	treatment_cost
201	MRI Scan	700
202	Blood Test	300
203	X-Ray	600
204	Physical Therapy	450

```
SELECT
    patient_name,
    doctor_name,
    appointment_date,
    treatment_name,
    treatment_cost
FROM
    Patients p
INNER JOIN Appointments a
ON p.patient_id = a.patient_id
INNER JOIN Doctors d
ON d.doctor_id = a.doctor_id
INNER JOIN Treatments t
ON t.treatment_id = a.treatment_id
WHERE YEAR(appointment_date) = 2024
AND treatment_cost > 500
ORDER BY patient_name ASC, treatment_cost DESC;
```

	patient_name	doctor_name	appointment_date	treatment_name	treatment_cost
1	Alice Brown	Dr. Emily Clark	2024-01-15	MRI Scan	700.00
2	Alice Brown	Dr. Sarah Lee	2024-07-10	X-Ray	600.00

Task # 2

Scenario: Most Liked Post

Context: You need to find the post with the highest number of likes overall.

Tables Involved:

1. **Posts**
2. **Likes**

Task

Write an SQL query using a single CTE to find the post with the highest number of likes overall, including the number of likes and the post date.

Posts

post_id	user_id	post_date
101	1	1/10/2024
102	2	3/15/2024
103	1	5/20/2024
104	3	7/25/2024
105	1	8/5/2024

Liked

like_id	post_id
201	101
202	101
203	103
204	105
205	105
206	105

```
WITH CTE (post, likes, post_date) AS (
        SELECT
                p.post_id post,
                count(*) likes,
                p.post_date post_date
        FROM
                Posts p
        INNER JOIN Likes l
```

```
        ON l.post_id = p.post_id
        GROUP BY
                p.post_id,
                p.post_date
)

SELECT TOP 1
        post,
        likes,
        post_date
FROM
        CTE
ORDER BY
        post desc;
```

	post	likes	post_date
1	105	3	2024-08-05

Task # 3

Scenario: Total Sales per Product per Category

Context: You manage a retail database and need to generate a report that shows the total sales for each product, grouped by product category. The report should also be ordered by total sales in descending order to highlight the best-selling products.

Tables Involved:

1. **Sales**
2. **Products**
3. **Categories**

Create a stored procedure named GenerateSalesReportByCategory that:

1. **Calculates Total Sales**: Computes the total sales for each product.
2. **Groups Data**: Groups the results by product and category.
3. **Orders Results**: Orders the results by total sales in descending order.

Sales Table

sale_id	product_id	sale_amount
1	101	150
2	101	100
3	102	200
4	103	120
5	104	170

Products Table

product_id	product_name	category_id
101	Product A	1
102	Product B	2
103	Product C	1
104	Product D	3

Categories Table

category_id	category_name
1	Electronics
2	Books
3	Clothing

```
CREATE PROCEDURE GenerateSalesReportByCategory AS

BEGIN

SELECT

        category_name as Category_Name,

        sum(sale_amount) Total_Sales_by_Product_Category

FROM

        Categories c

        INNER JOIN Products p ON c.category_id = p.category_id

        INNER JOIN Sales s ON p.product_id = s.product_id

GROUP BY

        category_name

ORDER BY

        Total_Sales_by_Product_Category DESC

END;


EXECUTE GenerateSalesReportByCategory;
```

	Category_Name	Total_Sales_by_Product_Category
1	Electronics	370.00
2	Books	200.00
3	Clothing	170.00

Task # 4

Create a view and an index to optimize query performance for generating sales reports. The task involves:

1. **Creating a View**: Define a view named SalesReportView that provides a summary of total sales for each product, grouped by category.
2. **Creating an Index**: Add an index on the Products table to improve the performance of queries involving category_id. Show Query Execution Plan.

1.

```
CREATE VIEW SalesReportView AS (
SELECT
        category_name as Category_Name,
        sum(sale_amount) Total_Sales_by_Product_Category
FROM
        Categories c
        INNER JOIN Products p ON c.category_id = p.category_id
        INNER JOIN Sales s ON p.product_id = s.product_id
GROUP BY
        category_name);


SELECT *
FROM SalesReportView
ORDER BY
        Total_Sales_by_Product_Category DESC;
```

	Category_Name	Total_Sales_by_Product_Category
1	Electronics	370.00
2	Books	200.00
3	Clothing	170.00

2.

CREATE INDEX index_prod_category_id

ON Products(category_id);

select category_id

from Products;