### 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;

A screen shot of a phone

Description automatically generated

### 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;

### 

### 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;



### 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

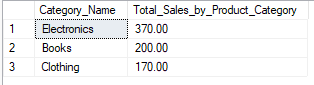
category\_name);

SELECT \*

FROM SalesReportView

ORDER BY

Total\_Sales\_by\_Product\_Category DESC;



2.

CREATE INDEX index\_prod\_category\_id

ON Products(category\_id);

select category\_id

from Products;

