Lab Sheet 10: Advanced SQL, Indexing, and MongoDB

Department of Computer Science and Engineering

November 25, 2025

Lab Task

This lab sheet contains five challenging questions designed to test your understanding of advanced SQL, database indexing, and MongoDB. You are required to provide solutions for each question.

1. **Advanced SQL: Recursive CTEs for Hierarchical Data**

   Consider a table named `employees` with the following schema:

   ```sql
   CREATE TABLE employees (
       employee_id INT PRIMARY KEY,
       employee_name VARCHAR(100),
       manager_id INT,
       FOREIGN KEY (manager_id) REFERENCES employees(employee_id)
   );
   ```

   Write a SQL query using a recursive Common Table Expression (CTE) to display the entire organizational hierarchy, starting from the CEO (who has no manager). The output should clearly show the level of each employee in the hierarchy.

2. **Advanced SQL: Window Functions for Trend Analysis**

   Given a table `sales` with the schema:

   ```sql
   CREATE TABLE sales (
       sale_id INT PRIMARY KEY,
       product_id INT,
       sale_date DATE,
       sale_amount DECIMAL(10, 2)
   );
   ```

Write a SQL query that calculates the moving average of sales for each product over a 7-day window. The query should also show the percentage change in sales compared to the previous day.

3. **Indexing: Optimizing a Complex Query**

   You are given a query that is performing poorly on a large database. The query joins three tables: `orders`, `customers`, and `products`.

```sql
SELECT c.customer_name, p.product_name, o.order_date
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
JOIN products p ON o.product_id = p.product_id
WHERE c.customer_segment = 'Premium'
  AND p.product_category = 'Electronics'
  AND o.order_date BETWEEN '2023-01-01' AND '2023-12-31';
```

   Explain what indexes you would create to optimize this query and why. Describe the concept of a covering index and how it could be applied here for maximum performance.

4. **MongoDB: Aggregation Pipeline for Data Analysis**

   You have a MongoDB collection named `articles` with documents in the following format:

```json
{
    "_id": ObjectId("..."),
    "title": "...",
    "author": "...",
    "tags": ["mongodb", "database", "nosql"],
    "views": 1500,
    "comments": [
        { "user": "...", "text": "...", "likes": 10 },
        { "user": "...", "text": "...", "likes": 5 }
    ]
}
```

   Write a MongoDB aggregation pipeline that:

   4.a. Filters for articles with more than 1000 views.

   4.b. Unwinds the `tags` array.

   4.c. Groups by tag to find the average number of comments and likes per tag.

4.d. Sorts the results by the average number of likes in descending order.

5. **MongoDB: Geospatial Queries**

   Consider a collection `restaurants` where each document stores the location of a restaurant as a GeoJSON Point.

   ```
   {
       "name": "...",
       "location": {
           "type": "Point",
           "coordinates": [longitude, latitude]
       }
   }
   ```

   First, explain how you would create a 2dsphere index on the `location` field. Then, write a query to find all restaurants within a 5-kilometer radius of a given point (e.g., longitude -73.98, latitude 40.77).

Submission Instructions

- For the SQL questions, submit the complete queries.
- For the indexing question, provide a clear explanation of the indexes and the reasoning behind them.
- For the MongoDB questions, submit the complete aggregation pipeline and queries.
- All submissions should be in a single text file.