
Practical Questions

These involve writing SQL queries, creating database objects, or implementing functionalities, ideal for hands-on practice in a MERN stack context where you'll interact with databases like PostgreSQL/MySQL via Node.js.

1. **Table Creation and Modification**
   - Create a table with fields `carname`, `color`, `price`, `model year` (1)
   - Create employee and department tables (1)
   - Create a table with a foreign key column (1)
   - Add a new column to an existing table (2)
   - Alter a table to add a foreign key constraint (1)
   - Add a column with a default value (1)
   - Modify the name of a column (1)
   - Remove a column from a table (1)
   - Delete a table in MySQL (1)
   - Rename a table/database (1)
   - Create a table with `id` auto-increment syntax (1)
   - Alter table to make a column a foreign key (1)
   - Create a table with `Age INT CHECK (Age >= 18)` (1)
   - Alter table with `AFTER` clause (1)

2. **Data Insertion**
   - Insert 5 values into a table (1)
   - Insert multiple rows with one query (1)
   - Insert five data into a table as a batch (1)
   - Create a stored procedure for insertion (6)

3. **Data Retrieval and Filtering**
   - Display all records from a table (1)

- Find cars whose price is higher than the average price (5)
- Find the car with the second highest price (3)
- Find the third largest number (1)
- Find the second highest salary (2)
- Find employees with more than 6 months of experience (1)
- Find employees joined within 2 years (1)
- Find customers who spent more than the average total spending (1)
- Find customers who haven’t placed any orders (1)
- Find the department with no employees (1)
- Find the department with the highest average salary (1)
- Find the manager with the highest number of employees (1)
- Count employees in the Engineering department (1)
- Count how many employees each manager has (1)
- Find the count of employees with the same salary (1)
- Find products with a price higher than the average price (1)
- Find products not purchased by any customer (1)
- Find the number of products purchased by each customer (1)
- Find students not in departments X and Y (1)
- Find students with the second largest age (1)
- Find the model of the car most available in count (1)
- Find the total order amount of each customer (1)
- Find department name and average salary (1)
- Show first name and department name of all employees (1)
- Name and salary of employees with the highest salary (1)
- Colors and count of cars (1)
- Cars ending with the letter ‘H’ (1)
- Names ending with ‘w’ (1)
- Remove employees whose names end with ‘n’ or ‘J’ (2)
- Remove employees with less than average salary (1)
- Write a query to group students by age and count them (1)
- Filter groups with more than 2 students (1)

- Count of each age command (1)

- SQL command to join a table and get the latest 3 employees with their department (1)

- Find duplicate rows from a table (1)

- Remove all duplicate records from a table (1)

- Find the longest full name (1)

- SQL queries with multiple joins, string functions, and aggregate functions (2)

- Join queries with subqueries, self-references, aggregate functions, date operations, window functions, rank, and CTE (2)

- Update query using joins (1)

4. **Data Updates and Deletion**

- Update records in SQL (1)

- Write a query to increase the salary of employees whose name starts with ‘D’ (1)

- Delete employees with salary > 80,000 (1)

- Delete a row from a table (1)

- Delete columns in MySQL (1)

- Delete statement without any condition (1)

- Update by even number (1)

5. **Views**

- Create a view including `carname` and `price` (4)

- Create a view with employee names starting with ‘D’ (1)

- Create a view named `car_price` and display it (1)

6. **Stored Procedures**

- Create a stored procedure (practical) (5)

- Write a function in PostgreSQL (1)

7. **Triggers**

- Create a trigger (practical) (3)

8. **Indexes**
   - Create an index (practical) (3)
   - List indexes (1)

9. **Transactions**
   - Implement a transaction (practical) (3)
   - Practice transactions with commit and rollback (1)

10. **Common Table Expressions (CTE)**
    - Practice CTEs (practical) (3)

11. **Subqueries**
    - Subquery workouts (3)
    - Correlated subquery (1)
    - Non-correlated subquery (1)
    - Exists subquery workout (1)
    - ANY vs ALL subquery workout (1)

12. **Other Practical Tasks**
    - Copy a table (1)
    - Create a user and grant read permission (1)
    - Use `EXPLAIN` and `EXPLAIN ANALYZE` commands (2)
    - Backup data using `pg_dump` (1)
    - Practice SQL injection prevention (4)
    - Write a query with `IF` condition inside `SELECT` (1)
    - Practice complex queries with workouts (2)
    - Use `CASE` and `IF` statements in queries (1)

Theoretical Questions

These focus on SQL and database concepts, crucial for designing efficient schemas and optimizing queries in MERN stack applications.

1. **Database Fundamentals**

- What is a database? (1)
- SQL vs RDBMS (2)
- SQL vs PostgreSQL (3)
- DBMS vs RDBMS (2)
- SQL vs NoSQL (2)
- Use cases of SQL vs NoSQL (1)
- Types of NoSQL databases (1)
- Relationship between SQL and RDBMS (1)
- Database vs table (1)
- Database concepts (needs clarity) (1)
- Pros and cons of SQL (2)
- Advantages and disadvantages of SQL (1)
- 3-schema architecture (3)
- Advantages of 3-schema architecture (1)
- External schema (1)
- Physical data storage (1)
- Entity (2)
- ER diagram (Entity-Relationship Diagram) (1)
- DBMS responsibility (1)

2. **SQL Commands and Categories**

- Types of SQL commands (1)
- DDL (Data Definition Language) (3)
- DML (Data Manipulation Language) (3)
- DCL (Data Control Language) (4)
- TCL (Transaction Control Language) (5)
- Differences between DDL, DML, DCL, and TCL (2)

- Examples of SQL commands (1)
- Is `SELECT` in DML? (1)
- Order of execution of SQL queries (2)
- Follow conventions when writing SQL queries (1)

3. **Keys and Constraints**
   - Primary key vs unique key (5)
   - Primary key vs foreign key (2)
   - Foreign key vs primary key (1)
   - Can foreign key accept null values? (1)
   - Super key (3)
   - Candidate key (4)
   - Composite key (4)
   - Natural key (1)
   - Which datatypes can be used as primary key? (1)
   - Is NULL value possible in primary key? (1)
   - Constraints in SQL (needs clarity) (2)
   - CHECK constraint (1)
   - DEFAULT constraint (1)
   - Auto-increment (1)
   - Cascading (2)
   - Functional dependency (1)
   - Partial dependency and transitive dependency (2)

4. **Data Types**
   - CHAR vs VARCHAR (4)
   - TEXT vs VARCHAR (2)
   - BLOB (2)
   - Array datatype (1)
   - Datatype for storing date and time (1)
   - UUID (1)

- SERIAL vs BIGSERIAL (1)
- Data types in PostgreSQL (1)

5. **Normalization and Denormalization**
- Normalization (needs clarity) (5)
- Normal forms (1NF, 2NF, 3NF, BCNF) (7)
- Disadvantages of normalization/over-normalization (2)
- Denormalization (2)
- Full form of BCNF (1)

6. **Relationships**
- Types of relationships (3)
- Many-to-many relationship (example) (1)
- Entities and relationships (2)
- Types of relations (1)

7. **Joins**
- Difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN (1)
- LEFT JOIN vs INNER JOIN (1)
- CROSS JOIN vs SELF JOIN (3)
- Natural join (1)
- Join vs UNION (2)
- Joins don’t require foreign keys (1)
- When to use GROUP BY, ORDER BY, COUNT, subqueries, joins, UNION, etc. (1)

8. **Indexes**
- What is an index and why is it useful? (1)
- How indexing works (internally) (2)
- Types of indexes (4)
- Clustered vs non-clustered index (2)
- Disadvantages of indexing (needs clarity) (2)

- How to decide which column needs indexing (2)
- Cons of indexes (1)

9. **Views**
   - What is a view in SQL? (1)
   - View vs table (1)
   - View vs stored procedure (1)
   - Can a view be created from multiple tables? (1)
   - Advantages of views (1)
   - Pros of materialized views (2)

10. **Stored Procedures**
    - What is a stored procedure? (2)
    - Advantages of stored procedures (1)

11. **Triggers**
    - What is a trigger in SQL? (2)
    - Cons of triggers (1)
    - Trigger syntax (1)

12. **Transactions**
    - What are transactions and why are they important? (1)
    - Properties of transactions (ACID) (3)
    - ACID properties (needs clarity) (4)
    - Atomicity (1)
    - Isolation (ACID) (2)
    - Durability (ACID) (2)
    - Sample scenarios/examples for ACID (1)
    - Database transaction usage (1)
    - Postgres concurrent transactions (1)
    - ROLLBACK, COMMIT, Savepoint (1)

- Deadlock (1)

13. **Functions**
    - Scalar functions (needs clarity) (4)
    - Scalar function examples (1)
    - Aggregate functions (3)
    - String operations/functions (1)
    - CONCAT function (1)
    - Window functions (2)
    - EXTRACT function (1)

14. **Operators and Clauses**
    - Comparison operators (`=`, `!=`, `<`, `>`, etc.) (1)
    - IN operator (1)
    - BETWEEN operator (1)
    - LIKE vs ILIKE (2)
    - LIKE vs REGEXP (1)
    - Wildcards (`%`, `_`) (2)
    - CASE statement syntax (2)
    - CASE WHEN (1)
    - WHERE vs HAVING (needs clarity) (2)
    - GROUP BY clause (needs clarification) (2)
    - GROUP BY vs WHERE vs HAVING (1)
    - EXISTS operator (2)
    - ANY vs ALL (1)
    - NOT IN clause (1)
    - UNION vs UNION ALL (5)
    - UNION vs INTERSECT (2)
    - INTERSECT and MINUS operators (1)
    - Conditions of UNION (1)
    - DISTINCT (2)

- ORDER BY (1)
- LIMIT vs OFFSET (1)
- OFFSET (1)

15. **Subqueries**
    - Subqueries (types and practical use) (2)
    - Types of subqueries (1)

16. **Common Table Expressions (CTE)**
    - Explain CTE (Common Table Expression) (2)
    - WITH clause (1)

17. **Cursors**
    - What is a cursor in SQL? (1)
    - Cursor (2)

18. **SQL Injection**
    - What is SQL injection and how to prevent it? (3)
    - Preventing SQL injection (2)

19. **Performance and Optimization**
    - Query optimization (1)
    - Performance improvement (1)
    - Maximum size for a table (1)
    - Partitioning and types of partitioning (2)
    - Types of shard keys (sharding) (1)
    - How large-scale databases can be optimized (1)
    - Scaling in SQL databases (horizontal vs vertical) (2)
    - Why prefer vertical scaling in SQL? (1)
    - Database migration (1)

20. **Concurrency and Locking**
    - Multi-version concurrency control (MVCC) (1)
    - Locks in databases (1)

21. **ACID and CAP Theorem**
    - CAP theorem (1)
    - CAP theorem applicability (1)
    - ACID applicability (1)

22. **Backup and Restore**
    - Backup in PostgreSQL (1)
    - Backup and restore (1)

23. **Miscellaneous**
    - Data redundancy vs data integrity (1)
    - Data integrity (how to ensure) (1)
    - Difference between MongoDB and SQL (1)
    - PostgreSQL vs MySQL (1)
    - Features of PostgreSQL (1)
    - pgAdmin (1)
    - JSON fields in SQL/PostgreSQL (1)
    - Table inheritance in SQL/PostgreSQL (1)
    - Global temp vs local temp tables (1)
    - Closure (2)
    - Dense RANK vs RANK (1)
    - When relational databases are preferred (1)
    - How to secure SQL databases (1)
    - Types of queries in SQL (1)
    - Arithmetic operations in SQL (1)
    - Combination of columns in SQL (1)
    - DROP vs TRUNCATE vs DELETE (3)

- UPDATE vs ALTER (1)

- What is UNION key in SQL? (1)

- PUT vs PATCH (1)

Recommendations for MERN Stack Context

- **Practical Focus**: Focus on high-occurrence practical topics like creating stored procedures (6), finding cars above average price (5), and creating views (4). These are directly relevant for backend development in MERN apps, where you'll write SQL queries in Node.js to interact with PostgreSQL/MySQL.

- **Theoretical Focus**: Prioritize frequently mentioned concepts like normal forms (7), primary key vs unique key (5), and UNION vs UNION ALL (5) to design efficient schemas and optimize queries, especially when deciding between SQL (e.g., PostgreSQL) and NoSQL (e.g., MongoDB) for your MERN project.

- **Practice**: Use a local PostgreSQL/MySQL instance or a cloud database (e.g., AWS RDS) with Node.js (via `pg` or `mysql2` libraries) to execute these queries. Test high-occurrence queries like finding duplicates or second-highest values in a sample MERN app.