Database Questions

Theory Questions

1. **Database Fundamentals**

   - What is a database?
   - DBMS vs RDBMS: Differences.
   - SQL vs RDBMS: Relationship.
   - SQL vs NoSQL: Use cases.
   - SQL vs PostgreSQL: Differences.
   - Pros and cons of SQL.
   - SQL vs NoSQL: Advantages.
   - When to prefer relational databases?

2. **Three Schema Architecture**

   - Explain three-schema architecture (Internal, Conceptual, External).
   - Advantages of three-schema architecture.

3. **ACID Properties**

   - What are ACID properties? Explain each.
   - Sample scenario for ACID.
   - Explain durability in ACID.
   - Explain isolation in ACID.

4. **Normalization and Denormalization**

   - What is normalization? Explain 1NF, 2NF, 3NF, BCNF.
   - Partial and transitive dependencies.
   - What is denormalization? Use cases.
   - Disadvantages of over-normalization.

5. **Constraints**

   - What are SQL constraints? (NOT NULL, UNIQUE, PK, FK, CHECK)
   - Primary key vs unique key.
   - Primary key vs foreign key.
   - Can foreign key accept NULL?
   - Can primary key have NULL?
   - CHECK constraint (e.g., Age >= 18).
   - DEFAULT constraints.

6. **Keys**

   - Super keys, candidate keys, composite keys.
   - Natural key.
   - Primary vs unique vs candidate key.

7. **Data Types**
    - CHAR vs VARCHAR.
    - VARCHAR vs TEXT.
    - BLOB: Use cases.
    - Data types for primary key.
    - Array data type.
    - Date and time data type.
    - JSON fields in PostgreSQL.
8. **Indexing**
    - What is an index? How it works.
    - Types of indexes (clustered vs non-clustered).
    - Pros and cons of indexing.
    - Choosing columns for indexing.
    - Covered query.
    - Clustered vs non-clustered index.
9. **Joins and Relationships**
    - Types of relationships (one-to-one, one-to-many, many-to-many).
    - INNER vs LEFT vs RIGHT vs FULL vs CROSS vs SELF JOIN.
    - Natural join.
    - Join vs union.
    - Joins without foreign key.
10. **SQL Commands**
    - DDL, DML, DCL, TCL: Examples.
    - DELETE vs TRUNCATE vs DROP.
    - WHERE vs HAVING.
    - SQL query order of execution.
11. **Views**
    - What is a view? View vs table.
    - Pros and cons of views.
    - Materialized view vs regular view.
    - Views from multiple tables.
12. **Stored Procedures**
    - What is a stored procedure? Benefits.
    - Advantages of stored procedures.
13. **Triggers**
    - What is a trigger?
    - Drawbacks of triggers.
14. **Transactions**

- What are transactions? Importance.
- COMMIT, ROLLBACK, SAVEPOINT.
- Transaction properties.
- Multi-version concurrency control (MVCC).
- Deadlocks.

15. **Functions**

- Scalar functions: Examples.
- Scalar vs aggregate functions.
- Aggregate functions (COUNT, SUM, AVG, MIN, MAX).
- CONCAT function.

16. **Subqueries**

- What are subqueries? Correlated vs non-correlated.
- ANY vs ALL operators.
- EXISTS vs IN.

17. **Common Table Expressions**

- What is a CTE? Usage.
- CTE vs subquery.

18. **Window Functions**

- What are window functions? Examples.
- RANK vs DENSE_RANK.

19. **Operators**

- Comparison operators (=, !=, <, >, <=, >=).
- IN, BETWEEN, LIKE, ILIKE.
- Wildcards (% and _).
- UNION vs UNION ALL vs INTERSECT vs MINUS.

20. **Security**

- What is SQL injection? Prevention.
- Securing SQL databases.
- Ensuring data integrity.

21. **Performance Optimization**

- Query optimization.
- Performance improvement techniques.
- EXPLAIN and EXPLAIN ANALYZE.

22. **Scaling and Architecture**

- Horizontal vs vertical scaling.
- Sharding: Types of shard keys.
- Partitioning: Types.
- CAP theorem applicability.

23. **Backup and Restore**

    - Backup in PostgreSQL (pg_dump).
    - Backup and restore processes.

24. **Other Concepts**

    - Cursor in SQL.
    - Closure in databases.
    - Functional dependency.
    - Entity in databases.
    - ER diagram: Purpose.
    - Data redundancy vs integrity.
    - Database vs table.
    - Global vs local temp tables.
    - UUID as primary key: When?
    - SERIAL vs BIGSERIAL.
    - Table inheritance in PostgreSQL.
    - Replica set architecture.
    - Maximum table size.
    - LIKE vs REGEXP.

Practical Questions

1. **Table Creation and Modification**

    - Create table: carname, color, price, model year.
    - Create employee & department tables with FK.
    - Create table with auto-increment ID.
    - Add new column to existing table.
    - Alter table to add FK.
    - Change column data type.
    - Rename table/database.
    - Delete column in MySQL.
    - Drop table in MySQL.
    - Add DEFAULT constraint.
    - Create table with CHECK (e.g., Age >= 18).
    - Alter table to make column FK.

2. **Data Insertion**

    - Insert 5 values into a table.
    - Insert multiple rows in one query.
    - Batch insert five data.
    - Create stored procedure for insertion.
    - Use UPSERT (ON CONFLICT DO NOTHING).

3. **Data Deletion**

    - Delete all records without condition.

- Delete employees: salary > 80,000.
- Remove employees: names end with 'J' or 'n'.
- Remove all duplicate records.
- Truncate a table.
- Delete a row.

4. **Data Updates**

- Increase salary for names starting with 'D'.
- Update records using joins.
- Update column with even-numbered values.
- Write update query for specific condition.

5. **Select Queries**

- Display all table records.
- Cars with price > avg price.
- Car with 2nd/3rd highest price.
- Cars with names ending 'H' or 'w'.
- Count cars by color.
- Model with highest count.
- 2nd largest student age.
- Students not in dept X/Y.
- Employees joined in last 2 years.
- Customers spending > avg.
- Customers with no orders.
- Products not purchased.
- Count products per customer.
- Employee with highest salary.
- Name & dept name of employees.
- Dept with highest avg salary.
- Dept with no employees.
- Count employees with same salary.
- Longest full name.
- Manager with most employees.
- Count employees in Engineering.
- Count employees per manager.
- Group students by age, count > 2.
- Count each age.
- Total order amount per customer.
- Latest 3 employees with dept.
- Remove employees: salary < avg.

6. **Joins**

- Join tables, filter by date.
- Query with multiple joins, string/aggregate functions.
- Joins with subqueries, self-references, window functions, rank.
- Name & dept name using JOIN.
- Combine JOIN and HAVING.

7. **Views**
    - Create view: carname, price.
    - Create view: employee names start with 'D'.
    - Create & display view: car_price.
8. **Stored Procedures**
    - Create stored procedure for insertion.
    - Write PostgreSQL function.
9. **Triggers**
    - Create a trigger.
10. **Indexes**
    - Create index (syntax).
    - List all indexes.
11. **Transactions**
    - Implement transaction with COMMIT/ROLLBACK.
    - Use SAVEPOINT.
12. **Subqueries**
    - Subquery: products price > avg.
    - Correlated subquery.
    - Use EXISTS in subquery.
    - Use ANY/ALL in subquery.
13. **Window Functions**
    - Query using window functions (RANK, DENSE_RANK).
14. **Common Table Expressions**
    - Query using CTE.
15. **Aggregate Functions**
    - Query with GROUP BY and aggregation.
    - Query with GROUP BY and HAVING.
16. **Security**
    - Create user, grant read permission.
    - Prevent SQL injection in query.
17. **Miscellaneous**
    - Copy a table.
    - Use EXPLAIN to analyze query.
    - Query with CASE/IF.

- Query with wildcards (%/_).
- Query with OFFSET/LIMIT.
- Increase score by 10.
- Query with ILIKE.
- Backup data with pg_dump.
- Alter table with AFTER.

18. **Specific Query** Fetch highest-paid employees per department:

```
WITH RankedEmployees AS (
    SELECT e.*, DENSE_RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS rnk
    FROM employees e
)
SELECT e.employee_name, e.salary, d.department_name
FROM RankedEmployees e
JOIN departments d ON e.department_id = d.department_id
WHERE rnk = 1;
```