
Some basic SQL Theory questions for your final mock.

1. What is SQL?

Answer:
SQL (Structured Query Language) is a standard language used to interact with relational databases. It allows for **data manipulation** (insert, update, delete), **data retrieval** (queries), and **management of database structures** (create, alter, drop tables). SQL is crucial in managing data-driven applications.

---

2. What are the types of SQL commands?

Answer:

1. **DDL (Data Definition Language)**: Commands like CREATE, ALTER, and DROP are used to define or modify database structures.
2. **DML (Data Manipulation Language)**: Commands like INSERT, UPDATE, DELETE modify the data in the database.
3. **DQL (Data Query Language)**: The SELECT statement retrieves data.
4. **DCL (Data Control Language)**: Commands like GRANT, REVOKE control permissions.
5. **TCL (Transaction Control Language)**: Commands like COMMIT, ROLLBACK, SAVEPOINT manage database transactions.

---

3. What is a primary key?

Answer:
A **primary key** uniquely identifies each record in a table. It:

- Must contain unique values.
- Cannot contain NULL values.
  For example:

```
CREATE TABLE students (
   student_id INT PRIMARY KEY,
   name VARCHAR(50)
);
```

---

4. What is a foreign key?

Answer:
A **foreign key** establishes a relationship between two tables. It references the primary key in another table.
Example:

```
CREATE TABLE orders (
   order_id INT PRIMARY KEY,
   customer_id INT,
   FOREIGN KEY (customer_id) REFERENCES
customers(customer_id)
);
```

---

5. What is a unique key?

Answer:
A **unique key** ensures that all values in a column are distinct. Unlike primary keys, it can include NULL values.
Example:

```
CREATE TABLE employees (
   emp_id INT UNIQUE,
   email VARCHAR(100) UNIQUE
);
```

---

6. What is the difference between WHERE and HAVING clauses?

Answer:

- **WHERE** filters rows before any grouping happens.
- **HAVING** filters grouped data after the GROUP BY clause.

Example:

```
SELECT department, AVG(salary)
FROM employees
WHERE salary > 50000
GROUP BY department
HAVING AVG(salary) > 60000;
```

---

7. What are joins in SQL?

Answer:
Joins combine rows from multiple tables based on related columns.
Types:

1. **INNER JOIN**: Returns rows with matching values in both tables.
2. **LEFT JOIN**: Returns all rows from the left table and matching rows from the right table.
3. **RIGHT JOIN**: Returns all rows from the right table and matching rows from the left table.
4. **FULL JOIN**: Returns rows with matches in either table.
5. **CROSS JOIN**: Returns the Cartesian product of two tables.
6. **SELF JOIN**: Joins a table with itself.

---

8. Explain the ACID properties.

Answer:

1. **Atomicity**: Ensures all parts of a transaction are completed or none are applied.
2. **Consistency**: Maintains database integrity before and after a transaction.
3. **Isolation**: Transactions do not interfere with each other.
4. **Durability**: Once committed, data changes are permanent.

---

9. What is normalization?

Answer:
Normalization is a process to organize database data to reduce redundancy and dependency and enhance data integrity.

- **1NF**: Remove duplicate columns; ensure atomicity, single cell cannot hold multiple values.
- **2NF**: Eliminate partial dependency.
- **3NF**: Remove transitive dependency.
-

---

10. What is denormalization?

Answer:
Denormalization adds redundancy to improve performance. Often used in OLAP systems to optimize query speeds.

 Reduced Joins:
By storing redundant data in one table, the need for joins across multiple tables is minimized, improving query performance.

 Improved Query Speed:
Since all the necessary data for a query is often in one table, queries run faster.

 Increased Storage Use:
Because data is duplicated, denormalization increases the storage requirements.

---

11. What is an index?

Answer:
An index improves data retrieval speed by maintaining a data structure for quick lookups.
Example:

```
CREATE INDEX idx_employee_name ON employees (name);
```

---

12. What is a clustered index?

Answer:
A clustered index sorts the actual table rows based on the key column values.

---

13. What is a non-clustered index?

Answer:
A non-clustered index creates a separate structure to store pointers to the data rows.

---

14. What is a subquery?

Answer:
A query inside another query, typically in SELECT, WHERE, or FROM.
Example:

```
SELECT name
```

FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);

---

15. What are correlated and non-correlated subqueries?

Answer:

- **Correlated Subquery**: References columns from the outer query.
- **Non-correlated Subquery**: Independent of the outer query.

---

16. What is the difference between DELETE and TRUNCATE?

Answer:

- **DELETE**: Removes rows selectively using WHERE. Can be rolled back.
- **TRUNCATE**: Removes all rows. Cannot be rolled back.

---

17. What is the difference between DROP and TRUNCATE?

Answer:

- **DROP**: Deletes the table and its structure.
- **TRUNCATE**: Clears data but retains the table structure.

---

18. What are constraints in SQL?

Answer:
Rules applied to columns to ensure data integrity:

- NOT NULL: Prevents NULL values.
- UNIQUE: Ensures uniqueness.
- CHECK: Validates values against a condition.

- FOREIGN KEY Constraint
- PRIMARY KEY Constraint
- DEFAULT Constraint

---

19. What is the difference between UNION and UNION ALL?

Answer:

- **UNION**: Combines and removes duplicates.
- **UNION ALL**: Combines without removing duplicates.

---

20. What is a stored procedure?

Answer:
A reusable set of SQL statements.

A stored procedure is a set of SQL statements that are stored in the database and can be executed repeatedly. It provides code reusability and better performance.

---

21. What are triggers in SQL?

Answer:
Triggers execute automatically in response to events (INSERT, UPDATE, DELETE).
The TRIGGER statement is used to associate a set of SQL statements with a specific event in the database. It is executed automatically when the event occurs.

---

22. What is a view?

Answer:
A view is a virtual table created using a SQL query. It doesn't store data.

CREATE VIEW employee_view AS
SELECT name, salary FROM employees;

---

23. What is the difference between CHAR and VARCHAR?

Answer:

- **CHAR**:
  - Fixed-length storage.
  - Always consumes the defined space, even for shorter data.
  - Faster for small, fixed-length data.
    Example:
- CHAR(10) // Always uses 10 bytes even if "Test" is stored.
- **VARCHAR**:
  - Variable-length storage.
  - Uses space based on the actual data size (plus 1-2 bytes for length).
  - Efficient for variable-length data.
    Example:
- VARCHAR(10) // Uses 4 bytes for "Test" plus length info.

---

24. What are the different types of relationships in SQL?

Answer:

1. **One-to-One**: Each row in Table A is linked to one row in Table B.
   Example: A user and their profile.
2. **One-to-Many**: One row in Table A is linked to multiple rows in Table B.
   Example: A customer and their orders.

3. **Many-to-Many**: Many rows in Table A link to many rows in Table B, typically via a junction table.
   Example: Students and courses.

---

25. How do you retrieve the current date in SQL?

Answer:

- In **SQL Server**: GETDATE()
- In **MySQL**: CURRENT_DATE()
  Example:

```
SELECT GETDATE();  -- SQL Server
SELECT CURRENT_DATE();  -- MySQL
```

** 26. How do you find the second-highest salary?

Answer:
Using a subquery:

```
SELECT MAX(salary)
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

Alternatively, using LIMIT:

```
SELECT salary
FROM employees
ORDER BY salary DESC
LIMIT 1 OFFSET 1;
```

---

27. What is the difference between RANK() and DENSE_RANK()?

Answer:

- **RANK()**: Skips ranks if there are ties.
- **DENSE_RANK()**: Does not skip ranks in case of ties.
  Example:

---

28. What are window functions in SQL?

Answer:
Window functions perform calculations across a set of rows related to the current row but do not group data.
Examples:

- ROW_NUMBER(): Assigns a unique number to each row.
- SUM(), AVG(): Aggregate functions over a window.

SELECT name, salary, ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_num
FROM employees;

---

29. What is a CTE (Common Table Expression)?

Answer:
A **CTE** is a temporary result set that can be referenced within a SQL query using WITH.

---

30. What are aggregate functions in SQL?

Answer:
Aggregate functions perform calculations on a set of values:

- SUM(): Total.
- COUNT(): Number of rows.
- AVG(): Average value.
- MIN(): Minimum value.

- MAX(): Maximum value.
  Example:

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department;
```

---

31. What is the difference between GROUP BY and ORDER BY?

Answer:

- **GROUP BY**: Groups rows with the same values into summary rows.
- **ORDER BY**: Sorts query results.
  Example:

```
SELECT department, COUNT(*)
FROM employees
GROUP BY department
ORDER BY COUNT(*) DESC;
```

---

32. What is a self-join?

Answer:
A self-join joins a table to itself. It's used to compare rows within the same table.
Example:

```
SELECT e1.name AS employee, e2.name AS manager
FROM employees e1
JOIN employees e2 ON e1.manager_id = e2.employee_id;
```

---

33. What is the difference between INNER JOIN and OUTER JOIN?

Answer:

- **INNER JOIN**: Returns only matching rows from both tables.
- **OUTER JOIN**: Includes rows from one or both tables even if no match exists (LEFT, RIGHT, FULL).
  Example:

```
SELECT e.name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

---

34. How do you handle NULL values in SQL?

Answer:

- Use IS NULL or IS NOT NULL to check for NULL.
- Use COALESCE to replace NULL with a default value.
  Example:

```
SELECT name, COALESCE(phone, 'No Phone') AS phone_number
FROM employees;
```

---

35. What is a transaction?

Answer:
A transaction is a sequence of SQL operations performed as a single logical unit.
Example:

```
BEGIN TRANSACTION;
UPDATE accounts SET balance = balance - 100 WHERE id = 1;
UPDATE accounts SET balance = balance + 100 WHERE id = 2;
COMMIT;
```

---

36. What is the difference between a database and a schema?

Answer:

A database is a container that holds multiple objects, such as tables, views, indexes, and procedures. It represents a logical grouping of related data.
A schema, on the other hand, is a container within a database that holds objects and defines their ownership. It provides a way to organize and manage database objects.

---

37. What is a deadlock?

Answer:
A deadlock occurs when two or more transactions are waiting for each other to release resources, resulting in a circular dependency. As a result, none of the transactions can proceed, and the system may become unresponsive.

---

38. What is a composite key?

Answer:
A composite key is a primary key made up of two or more columns.
Example:

```
CREATE TABLE orders (
   order_id INT,
   product_id INT,
   PRIMARY KEY (order_id, product_id)
);
```

---

39. What are scalar functions?

Answer:
Scalar functions return a single value based on input. Examples:

- LEN(): Length of a string.
- ROUND(): Rounds a number.
- UPPER(): Converts text to uppercase.
  Example:

SELECT UPPER(name) AS uppercase_name FROM employees;

40.What is the flow of execution for SQL commands,?

Answer:

SQL Execution Order (Logical vs. Written)

Logical Execution	**Query Syntax Order**
1. FROM	1. SELECT
2. WHERE	2. FROM
3. GROUP BY	3. WHERE
4. HAVING	4. GROUP BY
5. SELECT	5. HAVING
6. ORDER BY	6. ORDER BY
7. LIMIT/OFFSET	7. LIMIT/OFFSET

Some basic SQL query questions for your final mock

---

1. Question:

Write an SQL query to retrieve all columns from the table students where the student's age is greater than 18.

Answer:

SELECT * FROM students WHERE age > 18;

This query selects all records from the students table where the age is greater than 18.

---

2. Question:

Write an SQL query to find the names and ages of students who have the name "John" from the table students.

Answer:
SELECT name, age FROM students WHERE name = 'John';

This query selects the name and age columns from the students table where the name is exactly "John."

---

3. Question:

Write an SQL query to find the total number of students in the students table.

Answer:
SELECT COUNT(*) FROM students;

This query counts and returns the total number of records (students) in the students table.

---

4. Question:

Write an SQL query to retrieve all distinct ages from the students table.

Answer:
SELECT DISTINCT age FROM students;

This query retrieves all distinct values in the age column, without duplicates.

---

5. Question:

Write an SQL query to find the average age of students in the students table.

Answer:
SELECT AVG(age) FROM students;

This query calculates the average (AVG) age from the students table.

---

6. Question:

Write an SQL query to list the students whose age is between 20 and 25.

Answer:

SELECT * FROM students WHERE age BETWEEN 20 AND 25;

This query retrieves all records from the students table where the age is between 20 and 25, inclusive.

---

7. Question:

Write an SQL query to retrieve the first 5 records from the students table.

Answer:

```
SELECT * FROM students LIMIT 5;
```

This query selects the first 5 rows from the students table.

---

8. Question:

Write an SQL query to retrieve the names of students sorted alphabetically in ascending order.

Answer:

```
SELECT name FROM students ORDER BY name ASC;
```

This query retrieves the name column from the students table, sorting the results in ascending order alphabetically.

---

9. Question:

Write an SQL query to update the age of a student with student_id = 10 to 22.

Answer:

```
UPDATE students SET age = 22 WHERE student_id = 10;
```

This query updates the age to 22 for the student whose student_id is 10.

---

10. Question:

Write an SQL query to delete a student with student_id = 5.

Answer:

DELETE FROM students WHERE student_id = 5;

This query deletes the record of the student with student_id equal to 5 from the students table.

---

11. Question:

Write an SQL query to find all students who have the age less than or equal to 16.

Answer:

SELECT * FROM students WHERE age <= 16;

This query selects all records from the students table where the age is less than or equal to 16.

---

12. Question:

Write an SQL query to retrieve all students, ordered by age in descending order.

Answer:

SELECT * FROM students ORDER BY age DESC;

This query retrieves all records from the students table, sorted by the age column in descending order (oldest first).

---

13. Question:

Write an SQL query to count the number of students whose age is exactly 18.

Answer:

SELECT COUNT(*) FROM students WHERE age = 18;

This query counts the number of students in the students table whose age is exactly 18.

---

14. Question:

Write an SQL query to select the minimum age of students from the students table.

Answer:

SELECT MIN(age) FROM students;

This query retrieves the minimum age from the students table.

---

15. Question:

Write an SQL query to find the students with ages greater than 25, ordered by student_id in ascending order.

Answer:

SELECT * FROM students WHERE age > 25 ORDER BY student_id ASC;

This query retrieves students whose age is greater than 25 and orders the results by student_id in ascending order.

---

16. Question:

Write an SQL query to find the average age of students who are older than 20.

Answer:

SELECT AVG(age) FROM students WHERE age > 20;

This query calculates the average age of students whose age is greater than 20.

---

17. Question:

Write an SQL query to retrieve the top 3 youngest students.

Answer:

SELECT * FROM students ORDER BY age ASC LIMIT 3;

This query retrieves the first 3 students with the lowest ages, sorting them by age in ascending order.

---

18. Question:

Write an SQL query to find the student(s) with the highest age.

Answer:

```
SELECT * FROM students WHERE age = (SELECT MAX(age) FROM students);
```

This query retrieves the student(s) whose age matches the highest age in the students table.

---

19. Question:

Write an SQL query to select all students who are either 18 or 22 years old.

Answer:

```
SELECT * FROM students WHERE age IN (18, 22);
```

This query retrieves all students whose age is either 18 or 22.

---

20. Question:

Write an SQL query to find the number of students in each age group (grouped by age).

Answer:

```
SELECT age, COUNT(*) FROM students GROUP BY age;
```

This query counts the number of students for each distinct age in the students table and groups the results by age.
