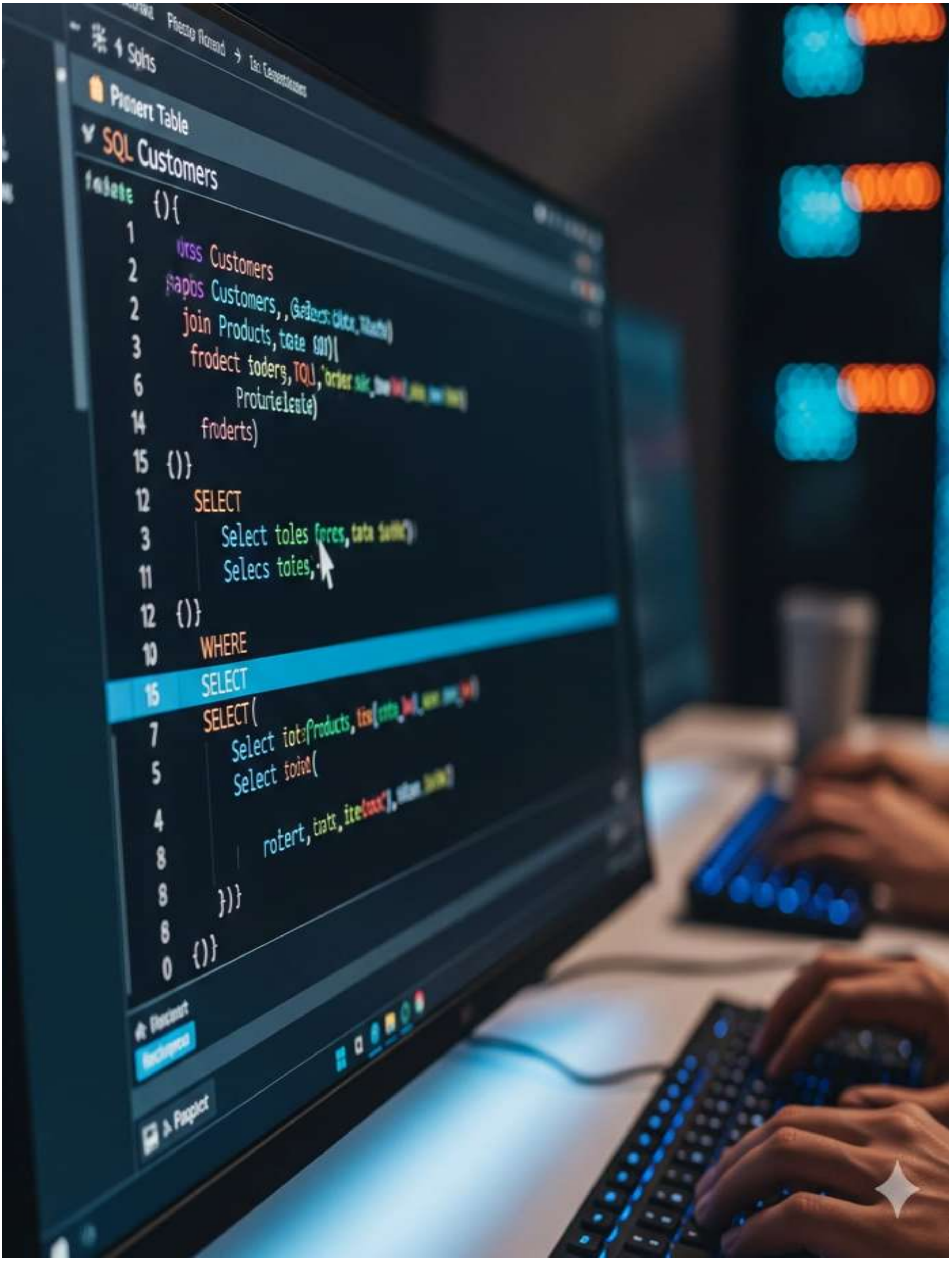
Important SQL Questions Answers

---Harsh Choudhary



Tables used in queries:

- **employees**: employee_id, employee_name, department_id, salary, hire_date
- **departments**: department_id, department_name, location

SQL Fundamentals

1. **What is the difference between SQL and MySQL?**
   - **Answer:** SQL (Structured Query Language) is a standard language for managing and querying relational databases. MySQL is a specific Relational Database Management System (RDBMS) that uses SQL. Think of SQL as the language and MySQL as the software that understands it.
   - **Example:** No query example, as this is a conceptual question.

2. **What are the different data types in SQL?**
   - **Answer:** Common data types include INT, VARCHAR, TEXT, DATE, DATETIME, DECIMAL, BOOLEAN, etc.
   - **Example:**

     SQL

     ```
     CREATE TABLE products (
         product_id INT PRIMARY KEY,
         product_name VARCHAR(100),
         price DECIMAL(10, 2),
         is_active BOOLEAN
     );
     ```

3. **Explain the purpose of the WHERE clause.**
   - **Answer:** The WHERE clause is used to filter records that a query returns, based on a specified condition.
   - **Example:**

     SQL

     ```
     SELECT * FROM employees WHERE salary > 70000;
     ```

4. **How do you use the GROUP BY clause?**
   - **Answer:** GROUP BY groups rows that have the same values into summary rows, like "find the number of employees in each department." It is often used with aggregate functions (COUNT, SUM, AVG, MIN, MAX).
   - **Example:**

     SQL

     ```
     SELECT department_id, COUNT(*) AS total_employees FROM employees
     GROUP BY department_id;
     ```

5. **What is the difference between `HAVING` and `WHERE`?**
   - **Answer:** `WHERE` filters rows *before* they are grouped. `HAVING` filters the groups *after* they have been created by `GROUP BY`.
   - **Example:**

     SQL

     ```sql
     -- Find departments with more than 5 employees
     SELECT department_id, COUNT(*) FROM employees GROUP BY department_id
     HAVING COUNT(*) > 5;
     ```

6. **Explain the `ORDER BY` clause and its usage.**
   - **Answer:** `ORDER BY` is used to sort the result set in ascending (`ASC`) or descending (`DESC`) order.
   - **Example:**

     SQL

     ```sql
     SELECT employee_name, salary FROM employees ORDER BY salary DESC;
     ```

7. **What is the purpose of the `DISTINCT` keyword?**
   - **Answer:** `DISTINCT` is used to return only unique (non-duplicate) values from a specified column.
   - **Example:**

     SQL

     ```sql
     SELECT DISTINCT department_id FROM employees;
     ```

8. **How do you insert a new record into a table?**
   - **Answer:** Using the `INSERT INTO` statement.
   - **Example:**

     SQL

     ```sql
     INSERT INTO employees (employee_id, employee_name, department_id,
     salary)
     VALUES (101, 'Jane Doe', 3, 75000);
     ```

9. **How do you update existing records in a table?**
   - **Answer:** Using the `UPDATE` statement with a `WHERE` clause to specify which records to modify.
   - **Example:**

     SQL

     ```sql
     UPDATE employees SET salary = 80000 WHERE employee_id = 101;
     ```

10. **How do you delete records from a table?**
    - **Answer:** Using the `DELETE` statement with a `WHERE` clause to specify which records to remove.

- **Example:**

  SQL

  ```
  DELETE FROM employees WHERE employee_id = 101;
  ```

Joins & Relationships

11. **Explain the different types of `JOINs`.**
    - **Answer:**
      - `INNER JOIN`: Returns records that have matching values in both tables.
      - `LEFT JOIN`: Returns all records from the left table, and the matched records from the right table.
      - `RIGHT JOIN`: Returns all records from the right table, and the matched records from the left table.
      - `FULL OUTER JOIN`: Returns all records when there is a match in either table.
    - **Example:**

      SQL

      ```
      SELECT e.employee_name, d.department_name
      FROM employees e
      INNER JOIN departments d ON e.department_id = d.department_id;
      ```
12. **What is a `SELF JOIN`?**
    - **Answer:** A join that joins a table to itself. This is useful for comparing rows within the same table.
    - **Example:** Find employees who have the same salary as another employee.

      SQL

      ```
      SELECT e1.employee_name, e2.employee_name, e1.salary
      FROM employees e1, employees e2
      WHERE e1.salary = e2.salary AND e1.employee_id <> e2.employee_id;
      ```
13. **What is a `CROSS JOIN`?**
    - **Answer:** A join that returns the Cartesian product of the tables, meaning every row from the first table is combined with every row from the second table.
    - **Example:**

      SQL

      ```
      SELECT * FROM employees CROSS JOIN departments;
      ```
14. **What is the difference between a `PRIMARY KEY` and a `FOREIGN KEY`?**
    - **Answer:** A `PRIMARY KEY` uniquely identifies each record in a table. It cannot be `NULL`. A `FOREIGN KEY` links to the `PRIMARY KEY` of another table, establishing a relationship.
    - **Example:**

SQL

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY, -- Primary Key
    ...
);

CREATE TABLE projects (
    project_id INT PRIMARY KEY,
    manager_id INT,
    FOREIGN KEY (manager_id) REFERENCES employees(employee_id) --
Foreign Key
);
```

15. **What is the purpose of a `FOREIGN KEY` constraint?**
    - **Answer:** It ensures referential integrity, preventing actions that would destroy the link between tables. For example, it prevents you from deleting a department that has employees assigned to it.
    - **Example:** The `FOREIGN KEY` constraint in the `projects` table above ensures that `manager_id` must exist in the `employees` table.

16. **What is a `UNION` and `UNION ALL`?**
    - **Answer:** Both combine the result sets of two or more `SELECT` statements. `UNION` removes duplicate rows, while `UNION ALL` includes all rows, including duplicates.
    - **Example:**

      SQL

      ```
      SELECT employee_name FROM employees
      UNION
      SELECT department_name FROM departments;
      ```

17. **What is the difference between `JOIN` and `UNION`?**
    - **Answer:** `JOIN` combines columns from different tables based on a related column. `UNION` combines rows from different tables, with the columns being identical.
    - **Example:** (See `JOIN` example above for `JOIN` and `UNION` example for `UNION`).

18. **What is a `NULL` value in SQL?**
    - **Answer:** A `NULL` value represents a missing or unknown value. It is not an empty string or zero.
    - **Example:**

      SQL

      ```
      SELECT * FROM employees WHERE hire_date IS NULL;
      ```

19. **How do you handle `NULL` values when performing calculations?**
    - **Answer:** Using functions like `COALESCE` (returns the first non-NULL value in a list) or `ISNULL` (specific to SQL Server) to replace `NULL` values.
    - **Example:**

      SQL

      ```
      -- Assuming a `bonus` column might be NULL
      SELECT employee_name, salary + COALESCE(bonus, 0) AS
      total_compensation
      FROM employees;
      ```

20. **What is a one-to-many relationship?**
    - **Answer:** A relationship where one record in a table can be associated with one or more records in another table. For example, one department can have many employees.
    - **Example:** The `department_id` in the `employees` table is the foreign key that creates a one-to-many relationship with the `departments` table.

Advanced SQL & Optimization

21. **Explain the concept of a `Subquery`.**
    - **Answer:** A query nested inside another query. It can be used in `SELECT`, `FROM`, `WHERE`, or `HAVING` clauses.
    - **Example:** Find all employees whose salary is greater than the average salary.

      SQL

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

22. **What is a `Common Table Expression (CTE)`?**
    - **Answer:** A temporary named result set that you can reference within a `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement. It improves readability and makes complex queries easier to manage.
    - **Example:**

      SQL

      ```
      WITH DepartmentAverage AS (
        SELECT department_id, AVG(salary) AS avg_salary FROM employees
      GROUP BY department_id
      )
      SELECT e.employee_name, e.salary, da.avg_salary
      FROM employees e
      JOIN DepartmentAverage da ON e.department_id = da.department_id
      WHERE e.salary > da.avg_salary;
      ```

23. **What are `Window Functions`? Provide an example.**

- **Answer:** Functions that perform a calculation across a set of table rows that are somehow related to the current row. They don't group rows into a single output row like aggregate functions.
- **Example:** Find the employee with the highest salary in each department.

  SQL

  ```sql
  SELECT * FROM (
    SELECT
      employee_name,
      salary,
      department_id,
      ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY salary
  DESC) as rn
    FROM employees
  ) AS ranked
  WHERE rn = 1;
  ```

24. **What is the difference between `ROW_NUMBER()`, `RANK()`, and `DENSE_RANK()`?**
    - **Answer:** All are window functions for ranking.
      - `ROW_NUMBER()`: Assigns a unique, sequential rank to each row (e.g., 1, 2, 3, 4).
      - `RANK()`: Assigns the same rank to rows with the same values, but skips the next rank (e.g., 1, 2, 2, 4).
      - `DENSE_RANK()`: Assigns the same rank to rows with the same values and does not skip ranks (e.g., 1, 2, 2, 3).
    - **Example:**

      SQL

      ```sql
      SELECT
        employee_name,
        salary,
        RANK() OVER (ORDER BY salary DESC) AS rank,
        DENSE_RANK() OVER (ORDER BY salary DESC) AS dense_rank
      FROM employees;
      ```

25. **Explain the purpose of a `VIEW`.**
    - **Answer:** A virtual table based on the result-set of an SQL statement. It simplifies complex queries and can be used for security by restricting access to certain columns or rows.
    - **Example:**

      SQL

      ```sql
      -- Create a view for HR to see employee salaries without their IDs
      CREATE VIEW employee_salaries_view AS
      SELECT employee_name, department_id, salary FROM employees;

      SELECT * FROM employee_salaries_view;
      ```

26. **What is an `INDEX` and why is it important?**

- **Answer:** An index is a data structure that improves the speed of data retrieval operations on a database table. It is similar to an index in a book.
- **Example:**

  SQL

```sql
CREATE INDEX idx_salary ON employees (salary);
```

27. **What is the difference between a `Clustered Index` and a `Non-Clustered Index`?**
    - **Answer:**
      - `Clustered Index`: Dictates the physical order of data in the table. A table can have only one.
      - `Non-Clustered Index`: Does not change the physical order of data. It creates a separate list of pointers to the data rows. A table can have multiple.
    - **Example:** No query, as this is a conceptual difference in `CREATE INDEX` syntax.

28. **How would you optimize a slow-running SQL query?**
    - **Answer:** Use `EXPLAIN` (or `EXPLAIN PLAN`) to understand the query execution plan, create appropriate indexes, rewrite the query to use `JOIN`s efficiently, avoid `SELECT *`, etc.
    - **Example:**

      SQL

```sql
-- Use this to analyze a query's performance
EXPLAIN SELECT * FROM employees WHERE hire_date > '2020-01-01';
```

29. **What is a `Stored Procedure`?**
    - **Answer:** A prepared SQL code that you can save and reuse. It can contain control flow statements and parameters.
    - **Example:**

      SQL

```sql
-- SQL Server syntax example
CREATE PROCEDURE GetEmployeesByDept
    @dept_id INT
AS
BEGIN
    SELECT employee_name, salary
    FROM employees
    WHERE department_id = @dept_id;
END;

EXEC GetEmployeesByDept 101;
```

30. **What is a `Trigger`?**
    - **Answer:** A special type of stored procedure that automatically executes when a specified event (like `INSERT`, `UPDATE`, or `DELETE`) occurs on a table.

- **Example:**

  SQL

```
-- Create a log table for employee salary changes
CREATE TABLE employee_salary_history (
  change_id INT PRIMARY KEY AUTO_INCREMENT,
  employee_id INT,
  old_salary DECIMAL(10, 2),
  new_salary DECIMAL(10, 2),
  change_date DATETIME
);

-- Create a trigger that fires after an UPDATE on salary
CREATE TRIGGER after_salary_update
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
  IF NEW.salary <> OLD.salary THEN
    INSERT INTO employee_salary_history (employee_id, old_salary,
new_salary, change_date)
    VALUES (OLD.employee_id, OLD.salary, NEW.salary, NOW());
  END IF;
END;
```

Database Concepts & Design

31. **Explain what `Normalization` is.**
    - **Answer:** A process of organizing data in a database to reduce data redundancy and improve data integrity. It involves breaking down a large table into smaller, related tables.
    - **Example:** Conceptual; no query.
32. **Describe the different `Normal Forms` (1NF, 2NF, 3NF).**
    - **Answer:**
      - 1NF: Each column contains atomic values, and there are no repeating groups of columns.
      - 2NF: It is in 1NF, and all non-key attributes are fully dependent on the primary key.
      - 3NF: It is in 2NF, and all non-key attributes are not transitively dependent on the primary key.
    - **Example:** Conceptual; no query.

33. **What is `Denormalization`?**
    - **Answer:** Intentionally introducing redundancy to a database to improve performance for specific query patterns. This is often used in data warehousing.
    - **Example:** No query.

34. **Explain the `ACID` properties of a transaction.**

- **Answer:**
    - Atomicity: All or nothing.
    - Consistency: A transaction brings the database from one valid state to another.
    - Isolation: Concurrent transactions do not interfere with each other.
    - Durability: Once a transaction is committed, it will persist.
- **Example:**

  SQL

```sql
-- A transaction to transfer money
BEGIN TRANSACTION;
UPDATE accounts SET balance = balance - 100 WHERE account_id = 1;
UPDATE accounts SET balance = balance + 100 WHERE account_id = 2;
COMMIT;
```

35. **What is a `Transaction`?**
    - **Answer:** A single unit of work composed of one or more SQL statements. It's either fully committed or fully rolled back.
    - **Example:** (See above).
36. **What is the difference between `TRUNCATE`, `DELETE`, and `DROP`?**
    - **Answer:**
        - `TRUNCATE`: Deletes all rows quickly. Cannot be rolled back. DDL command.
        - `DELETE`: Deletes rows based on a condition. Can be rolled back. DML command.
        - `DROP`: Deletes the entire table structure and all data. Cannot be rolled back. DDL command.
    - **Example:**

      SQL

```sql
-- Delete some rows
DELETE FROM employees WHERE salary < 50000;

-- Delete all rows
TRUNCATE TABLE employees;

-- Delete the entire table
DROP TABLE employees;
```

37. **What is a `Schema` in a database?**
    - **Answer:** A logical container for database objects (tables, views, stored procedures). It is used to group objects logically and to manage user permissions.
    - **Example:**

      SQL

```sql
-- PostgreSQL syntax
CREATE SCHEMA hr_schema;
CREATE TABLE hr_schema.employees ( ... );
```

38. **What is the difference between `DDL`, `DML`, and `DCL`?**
   - **Answer:**
     - `DDL` (Data Definition Language): Used to define the database schema (`CREATE`, `ALTER`, `DROP`).
     - `DML` (Data Manipulation Language): Used to manipulate data (`INSERT`, `UPDATE`, `DELETE`, `SELECT`).
     - `DCL` (Data Control Language): Used to control permissions (`GRANT`, `REVOKE`).
   - **Example:** No query.
39. **What are `Constraints`? Name a few examples.**
   - **Answer:** Rules enforced on data columns to limit what can be inserted into a table. Examples: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`.
   - **Example:**

     SQL

     ```
     CREATE TABLE students (
         student_id INT PRIMARY KEY,
         age INT CHECK (age >= 18) -- CHECK constraint
     );
     ```
40. **What is a `VIEW`? Can you update data through a view?**
   - **Answer:** A virtual table based on a query. You can update data through a view if it is simple (e.g., based on a single table with a primary key).
   - **Example:** (See Q.25)

System Design & General Tech

41. **How would you design a database schema for an e-commerce platform?**
   - **Answer:** Involves tables for `users`, `products`, `orders`, `order_items`, `payments`, etc., with appropriate primary and foreign keys.
   - **Example:** No query.

42. **Explain the difference between `OLAP` and `OLTP`.**
   - **Answer:**
     - `OLTP` (Online Transaction Processing): Handles a large number of small, short transactions (e.g., bank transactions). Focuses on fast, concurrent, and reliable data processing.
     - `OLAP` (Online Analytical Processing): Handles complex analytical queries on large data sets for business intelligence. Focuses on read operations and is often denormalized.
   - **Example:** No query.

43. **What is `Replication` in a database?**
    - **Answer:** The process of copying and distributing data from a primary database to one or more secondary databases. Used for high availability and load balancing.
    - **Example:** No query.

44. **What is a `Singleton` design pattern?**
    - **Answer:** A design pattern that restricts the instantiation of a class to a single object.
    - **Example:** Not a SQL concept.

45. **Explain the `CAP` theorem.**
    - **Answer:** States that a distributed data store can only provide two of three guarantees: Consistency, Availability, and Partition Tolerance.
    - **Example:** Not a SQL concept.

46. **What is the difference between `Symmetric` and `Asymmetric` encryption?**
    - **Answer:**
        - `Symmetric`: Uses the same key for both encryption and decryption.
        - `Asymmetric`: Uses a public key for encryption and a private key for decryption.
    - **Example:** Not a SQL concept.

47. **How would you handle a large number of concurrent users accessing a database?**
    - **Answer:** Techniques include connection pooling, read replicas, database sharding, and optimizing queries and indexes.
    - **Example:** No query.

48. **What is `Big Data`? Name some technologies used to handle it.**
    - **Answer:** Extremely large data sets that are too complex for traditional data processing. Technologies: Hadoop, Spark, NoSQL databases (e.g., Cassandra, MongoDB).
    - **Example:** No query.

49. **What is the purpose of a `Load Balancer`?**
    - **Answer:** Distributes network traffic across multiple servers to ensure no single server is overworked.

- **Example:** No query.

50. **What is the difference between `Authentication` and `Authorization`?**
    - **Answer:**
        - `Authentication`: Verifies who the user is (e.g., username and password).
        - `Authorization`: Determines what the user is allowed to do (e.g., `GRANT` and `REVOKE` permissions).
    - **Example:**

      SQL

      ```
      -- Grant select privilege on the employees table to a user named
      'analyst'
      GRANT SELECT ON employees TO analyst;

      -- Revoke that privilege
      REVOKE SELECT ON employees FROM analyst;
      ```

General & Advanced Concepts

51. **What is a `Singleton` design pattern?**
    - **Answer:** A design pattern that restricts the instantiation of a class to a single object. This is useful when you need a single, global point of access to a resource, like a database connection or a logger.
    - **Code:**

      Python

      ```
      class Logger:
          _instance = None
          def __new__(cls):
              if cls._instance is None:
                  cls._instance = super(Logger, cls).__new__(cls)
              return cls._instance

      logger1 = Logger()
      logger2 = Logger()
      print(logger1 is logger2)  # Output: True
      ```

52. **Explain `SOLID` principles.**
    - **Answer:** A mnemonic for five design principles intended to make software designs more understandable, flexible, and maintainable.
        - **S**ingle Responsibility Principle
        - **O**pen/Closed Principle
        - **L**iskov Substitution Principle
        - **I**nterface Segregation Principle
        - **D**ependency Inversion Principle

53. **What is `Dependency Injection`?**
   - **Answer:** A design pattern in which a class receives its dependencies from an external source rather than creating them itself. This promotes loose coupling and makes code easier to test.
   - **Code:**

     Python

     ```
     class Database:
         def connect(self):
             # Connection logic
             pass

     # Without Dependency Injection
     class UserRepository:
         def __init__(self):
             self.db = Database() # Tightly coupled

     # With Dependency Injection
     class UserRepositoryDI:
         def __init__(self, db: Database):
             self.db = db # Dependency is injected
     ```

54. **What is the difference between `JWT` and `OAuth`?**
   - **Answer:** `OAuth` is an authorization protocol that allows a user to grant a third-party application access to their information without sharing their password. `JWT` (JSON Web Token) is a standard for creating tokens that contain claims (e.g., user ID, roles) that can be verified securely by a server. `JWT` is often used as the access token in an `OAuth` flow.

55. **Explain `Microservices` architecture.**
   - **Answer:** An architectural style where a complex application is composed of small, independent services that communicate with each other, often via APIs. Each service runs in its own process and is independently deployable.

56. **What is `Docker`?**
   - **Answer:** A platform that uses OS-level virtualization to deliver software in packages called containers. These containers are isolated from each other and the host system, ensuring an application runs consistently across different environments.
   - **Code:**

     Dockerfile

```
# Simple Dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

57. **What is the purpose of `Kubernetes`?**
    - **Answer:** An open-source container orchestration system that automates the deployment, scaling, and management of containerized applications. It helps manage multiple Docker containers across a cluster of machines.

58. **Explain the `DRY` principle.**
    - **Answer:** `Don't Repeat Yourself`. A software development principle aimed at reducing repetition of software patterns, replacing them with abstractions or using data normalization.
    - **Code:**

      Python

      ```
      # Bad (violation of DRY)
      def calculate_total_price(item1, item2):
          total = item1.price + item2.price
          return total

      # Good (DRY)
      def calculate_total_price_list(items):
          total = sum(item.price for item in items)
          return total
      ```

59. **What is `Regression Testing`?**
    - **Answer:** A type of software testing that verifies that recently added code or changes have not adversely affected existing functionalities.

60. **What is the difference between a `compiler` and an `interpreter`?**
    - **Answer:** A `compiler` translates the entire source code into machine code before execution. An `interpreter` translates and executes code line by line.

61. **What is a `session` vs `cookie`?**
    - **Answer:** A `cookie` is a small text file stored on the client's browser. A `session` is data stored on the server. The cookie often contains a session ID that the server uses to retrieve the session data.

62. **What is `CORS`?**

- **Answer:** `Cross-Origin Resource Sharing`. A mechanism that allows resources on a web page to be requested from another domain outside the domain from which the first resource was served. It's a security feature enforced by browsers.

63. **What is the difference between `SQL` and `NoSQL`?**
    - **Answer:** `SQL` databases are relational, use a structured schema, and are good for complex queries. `NoSQL` databases are non-relational, have a flexible schema, and are often used for large-scale, unstructured data.
    - **Code:**

      JavaScript

      ```javascript
      // NoSQL (MongoDB) query example
      db.employees.find({ salary: { $gt: 70000 } });
      ```

64. **What is `Docker Compose`?**
    - **Answer:** A tool for defining and running multi-container Docker applications. It uses a YAML file to configure the services, networks, and volumes for an application.
    - **Code:**

      YAML

      ```yaml
      # docker-compose.yml
      services:
        web:
          image: "nginx:latest"
          ports:
            - "80:80"
        db:
          image: "postgres:13"
          environment:
            POSTGRES_USER: user
            POSTGRES_PASSWORD: password
      ```

65. **Explain `Load Balancing`.**
    - **Answer:** The process of distributing network traffic across multiple servers to ensure no single server is overworked, improving application availability and responsiveness.

66. **What is the difference between `HTTP` and `HTTPS`?**
    - **Answer:** `HTTP` is the standard protocol for web data transfer. `HTTPS` is the secure version of HTTP, which uses `SSL/TLS` to encrypt communication between the browser and the server.

67. **Explain `REST vs SOAP`.**

- **Answer:** `REST` is an architectural style, while `SOAP` is a protocol. `REST` is more flexible and uses standard HTTP methods. `SOAP` is more rigid, relies on XML, and has stricter security and transaction standards.

68. **What is a `unit test`?**
    - **Answer:** A software testing method where individual units of source code, like functions or methods, are tested to ensure they work as expected.
    - **Code:**

      Python

      ```
      # Python unit test example using unittest
      import unittest

      def add(a, b):
          return a + b

      class TestAddition(unittest.TestCase):
          def test_positive_numbers(self):
              self.assertEqual(add(2, 3), 5)

      if __name__ == '__main__':
          unittest.main()
      ```

69. **What is `refactoring`?**
    - **Answer:** The process of restructuring existing computer code without changing its external behavior. It improves non-functional attributes of the software, such as readability and maintainability.

70. **What is `CAP Theorem`?**
    - **Answer:** A fundamental principle in distributed computing stating that it is impossible for a distributed data store to simultaneously provide more than two out of the three guarantees: Consistency, Availability, and Partition tolerance.