**Basic SQL Questions**

1. **What is SQL?**
   * **Answer:** SQL (Structured Query Language) is a standard programming language used to manage and manipulate relational databases.
2. **What is the difference between SQL and MySQL?**
   * **Answer:** SQL is the language used for database queries, while MySQL is an open-source database management system that uses SQL to interact with data.
3. **What are the different types of SQL statements?**
   * **Answer:**
     + **DDL (Data Definition Language):** CREATE, ALTER, DROP
     + **DML (Data Manipulation Language):** SELECT, INSERT, UPDATE, DELETE
     + **DCL (Data Control Language):** GRANT, REVOKE
     + **TCL (Transaction Control Language):** COMMIT, ROLLBACK, SAVEPOINT
4. **What is a primary key?**
   * **Answer:** A primary key is a unique identifier for a record in a table. It cannot be NULL and must be unique for each row.
5. **What is a foreign key?**
   * **Answer:** A foreign key is a column (or a set of columns) in a table that establishes a link between data in two tables. It refers to the primary key in another table.
6. **What is the difference between INNER JOIN and LEFT JOIN?**
   * **Answer:**
     + INNER JOIN: Returns only rows with matching values in both tables.
     + LEFT JOIN: Returns all rows from the left table and matching rows from the right table, or NULL if no match exists.
7. **What is a unique constraint?**
   * **Answer:** A unique constraint ensures that all values in a column are distinct. It allows one NULL value unless defined otherwise.
8. **How do you retrieve all records from a table?**
   * **Answer:**

SELECT \* FROM table\_name;

1. **What is a JOIN in SQL?**
   * **Answer:** A JOIN is used to combine rows from two or more tables based on a related column. Common types include INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN.
2. **What does the DISTINCT keyword do?**
   * **Answer:** The DISTINCT keyword is used to return unique values in a query result, eliminating duplicate records.

**Intermediate SQL Questions**

1. **What is the difference between WHERE and HAVING?**
   * **Answer:**
     + WHERE is used to filter records before grouping.
     + HAVING is used to filter records after grouping, typically with aggregate functions.
2. **What is an aggregate function in SQL?**
   * **Answer:** Aggregate functions perform calculations on multiple rows of a table, such as COUNT(), SUM(), AVG(), MIN(), and MAX().
3. **What is a subquery, and when would you use it?**
   * **Answer:** A subquery is a query within another query. It is typically used to retrieve a single value or set of values that will be used by the outer query.
4. **How do you update a record in SQL?**
   * **Answer:**

UPDATE table\_name  
SET column\_name = new\_value  
WHERE condition;

1. **What is a GROUP BY clause used for?**
   * **Answer:** The GROUP BY clause is used to group rows that have the same values into summary rows, often used with aggregate functions.
2. **How do you delete a record from a table?**
   * **Answer:**

DELETE FROM table\_name  
WHERE condition;

1. **What is the LIMIT clause used for?**
   * **Answer:** The LIMIT clause is used to specify the number of records returned by a query. For example, LIMIT 10 returns the first 10 records.
2. **What is the difference between DELETE and TRUNCATE?**
   * **Answer:**
     + DELETE removes rows from a table with the option to rollback and can include a WHERE clause.
     + TRUNCATE removes all rows from a table and cannot be rolled back.
3. **What is a CASE statement in SQL?**
   * **Answer:** The CASE statement is used for conditional logic, similar to an IF statement in programming. It allows you to return different values based on conditions.
4. **What is the difference between UNION and UNION ALL?**
   * **Answer:**
     + UNION: Combines results of two queries and removes duplicates.
     + UNION ALL: Combines results of two queries and includes duplicates.

**Real-World SQL Scenario Questions**

1. **You are given two tables: Customers and Orders. How would you retrieve a list of all customers who have placed at least one order?**
   * **Answer:**

SELECT DISTINCT Customers.CustomerID, Customers.Name  
FROM Customers  
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

1. **How would you find all employees who have a salary greater than the average salary in the company?**
   * **Answer:**

SELECT EmployeeID, Name, Salary  
FROM Employees  
WHERE Salary > (SELECT AVG(Salary) FROM Employees);

1. **You need to find the top 3 highest-paid employees. How would you do this?**
   * **Answer:**

SELECT EmployeeID, Name, Salary  
FROM Employees  
ORDER BY Salary DESC  
LIMIT 3;

1. **How would you find the total sales made by each salesperson in a Sales table?**
   * **Answer:**

SELECT SalespersonID, SUM(SaleAmount) AS TotalSales  
FROM Sales  
GROUP BY SalespersonID;

1. **How would you retrieve a list of employees who have not completed any orders, assuming you have an Orders table and an Employees table?**
   * **Answer:**

SELECT Employees.EmployeeID, Employees.Name  
FROM Employees  
LEFT JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID  
WHERE Orders.OrderID IS NULL;

1. **You have two tables: Products and Sales. How would you find the number of products sold for each product?**
   * **Answer:**

SELECT Products.ProductID, Products.Name, COUNT(Sales.ProductID) AS NumberSold  
FROM Products  
LEFT JOIN Sales ON Products.ProductID = Sales.ProductID  
GROUP BY Products.ProductID;

1. **How would you calculate the average sales for each month in a Sales table?**
   * **Answer:**

SELECT YEAR(SaleDate) AS Year, MONTH(SaleDate) AS Month, AVG(SaleAmount) AS AverageSales  
FROM Sales  
GROUP BY YEAR(SaleDate), MONTH(SaleDate);

1. **What SQL query would you write to find the second-highest salary from an Employees table?**
   * **Answer:**

SELECT MAX(Salary) AS SecondHighestSalary  
FROM Employees  
WHERE Salary < (SELECT MAX(Salary) FROM Employees);

1. **How would you join three tables: Customers, Orders, and Products, to find out which products were ordered by customers?**
   * **Answer:**

SELECT Customers.Name, Products.Name  
FROM Customers  
JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
JOIN Products ON Orders.ProductID = Products.ProductID;

1. **You are given a Sales table with columns SaleDate and SaleAmount. How would you find the total sales for each year?**
   * **Answer:**

SELECT YEAR(SaleDate) AS Year, SUM(SaleAmount) AS TotalSales  
FROM Sales  
GROUP BY YEAR(SaleDate);

**Advanced SQL Questions**

1. **What is a window function in SQL? Provide an example.**
   * **Answer:** A window function performs calculations across a set of table rows related to the current row. For example:

SELECT EmployeeID, Salary,   
       RANK() OVER (ORDER BY Salary DESC) AS SalaryRank  
FROM Employees;

1. **What is indexing in SQL, and why is it important?**
   * **Answer:** Indexing improves the speed of data retrieval operations on a table. It is especially useful for large datasets.
2. **What is normalization in SQL, and why is it necessary?**
   * **Answer:** Normalization is the process of organizing data to reduce redundancy and dependency. It ensures that data is stored efficiently and minimizes anomalies.
3. **Explain the concept of ACID properties in SQL databases.**
   * **Answer:** ACID stands for Atomicity, Consistency, Isolation, and Durability, which are properties that ensure reliable transaction processing in a database.
4. **What is a self join in SQL?**
   * **Answer:** A self join is a join where a table is joined with itself. This is useful for hierarchical data.

SELECT E1.EmployeeID, E1.Name, E2.Name AS Manager  
FROM Employees E1  
LEFT JOIN Employees E2 ON E1.ManagerID = E2.EmployeeID;

1. **What is a CROSS JOIN in SQL, and when would you use it?**
   * **Answer:** A CROSS JOIN returns the Cartesian product of two tables, i.e., it combines every row from the first table with every row from the second table.

SELECT \* FROM A CROSS JOIN B;

1. **What is the purpose of WITH clauses (Common Table Expressions) in SQL?**
   * **Answer:** CTEs are used to simplify complex queries by defining temporary result sets that can be referred to within a SELECT, INSERT, UPDATE, or DELETE statement.
2. **What is a stored procedure in SQL?**
   * **Answer:** A stored procedure is a set of SQL statements that can be executed as a single unit. It can be called to perform operations like inserts, updates, deletes, and selects.
3. **What is a trigger in SQL, and when is it used?**
   * **Answer:** A trigger is a set of SQL statements that are automatically executed when certain events (such as INSERT, UPDATE, or DELETE) occur on a table.
4. **What is the difference between ROW\_NUMBER() and RANK() window functions in SQL?**
   * **Answer:**
     + ROW\_NUMBER() assigns a unique sequential number to rows.
     + RANK() assigns a rank to each row, but rows with the same values will receive the same rank, with gaps in ranking.

**Scenario-Based SQL Questions**

1. **You need to find the customers who have purchased products more than twice. How would you do that?**
   * **Answer:**

SELECT CustomerID, COUNT(\*) AS PurchaseCount  
FROM Orders  
GROUP BY CustomerID  
HAVING COUNT(\*) > 2;

1. **How would you calculate the difference between two dates (e.g., the number of days between StartDate and EndDate)?**
   * **Answer:**

SELECT DATEDIFF(EndDate, StartDate) AS DateDifference  
FROM TableName;

1. **How would you delete all records in a table except for the most recent one?**
   * **Answer:**

DELETE FROM TableName  
WHERE DateColumn < (SELECT MAX(DateColumn) FROM TableName);

1. **You need to find the earliest order date for each customer. How would you do this?**
   * **Answer:**

SELECT CustomerID, MIN(OrderDate) AS EarliestOrderDate  
FROM Orders  
GROUP BY CustomerID;

1. **You are given a table Transactions with columns TransactionDate and Amount. How would you calculate the running total of transactions?**
   * **Answer:**

SELECT TransactionDate, Amount,   
       SUM(Amount) OVER (ORDER BY TransactionDate) AS RunningTotal  
FROM Transactions;

**Advanced Performance and Optimization**

1. **How do you optimize slow SQL queries?**
   * **Answer:**
     + Use indexes for frequently queried columns.
     + Avoid SELECT \* and only retrieve necessary columns.
     + Use joins efficiently, and consider using EXPLAIN to analyze query execution plans.
2. **What is query execution plan, and how do you read it?**
   * **Answer:** A query execution plan outlines how a database engine will execute a query. It helps in understanding the performance aspects and can be used to optimize queries.
3. **What is the difference between UNION and INTERSECT?**
   * **Answer:**
     + UNION: Combines results of two queries and removes duplicates.
     + INTERSECT: Returns only rows that are common between two queries.
4. **How do you ensure data consistency in a multi-user environment?**
   * **Answer:**
     + Use transactions with ACID properties (Atomicity, Consistency, Isolation, Durability).
     + Implement proper locking mechanisms.
5. **What are some strategies for handling large datasets in SQL?**
   * **Answer:**
     + Use indexing, partitioning, and caching to improve performance.
     + Optimize queries by reducing unnecessary operations, like joins and subqueries.
     + Use pagination to retrieve data in chunks rather than all at once.

### Basic Concepts

1. \*\*What is MySQL?\*\*

- Explain what MySQL is and its primary use.

2. \*\*Explain the ACID properties.\*\*

- Describe Atomicity, Consistency, Isolation, and Durability.

3. \*\*What are the different types of joins in MySQL?\*\*

- List and briefly describe INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN.

4. \*\*What is the difference between CHAR and VARCHAR data types?\*\*

- Explain the main differences in storage and usage.

5. \*\*What is a primary key?\*\*

- Define a primary key and its purpose in a database table.

6. \*\*What is a foreign key?\*\*

- Explain what a foreign key is and its role in relational databases.

7. \*\*What is normalization?\*\*

- Describe the process and purpose of normalization in database design.

8. \*\*What are indexes in MySQL?\*\*

- Explain what indexes are and how they improve query performance.

9. \*\*What is the difference between DELETE, TRUNCATE, and DROP commands?\*\*

- Describe how each command works and when to use them.

10. \*\*What is a stored procedure?\*\*

- Define stored procedures and their advantages.

### Basic SQL Queries

11. \*\*Write a query to create a table named `employees` with columns `id`, `name`, `position`, `salary`, and `hire\_date`.\*\*

```sql

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

position VARCHAR(50),

salary DECIMAL(10, 2),

hire\_date DATE

);

```

12. \*\*Write a query to insert a new employee into the `employees` table.\*\*

```sql

INSERT INTO employees (name, position, salary, hire\_date) VALUES ('John Doe', 'Manager', 75000, '2023-06-01');

```

13. \*\*Write a query to retrieve all employees with a salary greater than $50,000.\*\*

```sql

SELECT \* FROM employees WHERE salary > 50000;

```

14. \*\*Write a query to update the salary of an employee with `id` = 1 to $80,000.\*\*

```sql

UPDATE employees SET salary = 80000 WHERE id = 1;

```

15. \*\*Write a query to delete an employee with the name 'John Doe'.\*\*

```sql

DELETE FROM employees WHERE name = 'John Doe';

```

### Advanced SQL Queries

16. \*\*Write a query to find the average salary of all employees.\*\*

```sql

SELECT AVG(salary) AS average\_salary FROM employees;

```

17. \*\*Write a query to count the number of employees in each position.\*\*

```sql

SELECT position, COUNT(\*) AS num\_employees FROM employees GROUP BY position;

```

18. \*\*Write a query to find the highest salary in the `employees` table.\*\*

```sql

SELECT MAX(salary) AS highest\_salary FROM employees;

```

19. \*\*Write a query to retrieve employees who were hired in the year 2023.\*\*

```sql

SELECT \* FROM employees WHERE YEAR(hire\_date) = 2023;

```

20. \*\*Write a query to list the first 5 employees ordered by their hire date in descending order.\*\*

```sql

SELECT \* FROM employees ORDER BY hire\_date DESC LIMIT 5;

```

21. \*\*Write a query to join the `employees` table with a `departments` table on `department\_id` and retrieve employee names and department names.\*\*

```sql

SELECT e.name, d.department\_name

FROM employees e

JOIN departments d ON e.department\_id = d.id;

```

22. \*\*Write a query to find the total salary expense for each department.\*\*

```sql

SELECT d.department\_name, SUM(e.salary) AS total\_salary

FROM employees e

JOIN departments d ON e.department\_id = d.id

GROUP BY d.department\_name;

```

23. \*\*Write a query to find all employees whose name starts with 'J'.\*\*

```sql

SELECT \* FROM employees WHERE name LIKE 'J%';

```

24. \*\*Write a query to find duplicate entries in the `employees` table based on the `name` column.\*\*

```sql

SELECT name, COUNT(\*) AS num\_occurrences

FROM employees

GROUP BY name

HAVING COUNT(\*) > 1;

```

25. \*\*Write a query to create an index on the `salary` column of the `employees` table.\*\*

```sql

CREATE INDEX idx\_salary ON employees(salary);

```

These questions and queries cover a range of topics from basic concepts to more advanced SQL queries, providing a comprehensive test of MySQL knowledge and skills.

Sure! Here are 25 questions that include JOINs, aggregation functions, DISTINCT, LIMIT, WHERE, GROUP BY, and HAVING:

### Basic Concepts

1. \*\*What is MySQL?\*\*

- Explain what MySQL is and its primary use.

2. \*\*Explain the ACID properties.\*\*

- Describe Atomicity, Consistency, Isolation, and Durability.

3. \*\*What are the different types of joins in MySQL?\*\*

- List and briefly describe INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN.

4. \*\*What is the difference between CHAR and VARCHAR data types?\*\*

- Explain the main differences in storage and usage.

5. \*\*What is a primary key?\*\*

- Define a primary key and its purpose in a database table.

6. \*\*What is a foreign key?\*\*

- Explain what a foreign key is and its role in relational databases.

7. \*\*What is normalization?\*\*

- Describe the process and purpose of normalization in database design.

8. \*\*What are indexes in MySQL?\*\*

- Explain what indexes are and how they improve query performance.

9. \*\*What is the difference between DELETE, TRUNCATE, and DROP commands?\*\*

- Describe how each command works and when to use them.

10. \*\*What is a stored procedure?\*\*

- Define stored procedures and their advantages.

### Basic SQL Queries

11. \*\*Write a query to create a table named `employees` with columns `id`, `name`, `position`, `salary`, and `hire\_date`.\*\*

```sql

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

position VARCHAR(50),

salary DECIMAL(10, 2),

hire\_date DATE

);

```

12. \*\*Write a query to insert a new employee into the `employees` table.\*\*

```sql

INSERT INTO employees (name, position, salary, hire\_date) VALUES ('John Doe', 'Manager', 75000, '2023-06-01');

```

13. \*\*Write a query to retrieve all employees with a salary greater than $50,000.\*\*

```sql

SELECT \* FROM employees WHERE salary > 50000;

```

14. \*\*Write a query to update the salary of an employee with `id` = 1 to $80,000.\*\*

```sql

UPDATE employees SET salary = 80000 WHERE id = 1;

```

15. \*\*Write a query to delete an employee with the name 'John Doe'.\*\*

```sql

DELETE FROM employees WHERE name = 'John Doe';

```

### Advanced SQL Queries

16. \*\*Write a query to find the average salary of all employees.\*\*

```sql

SELECT AVG(salary) AS average\_salary FROM employees;

```

17. \*\*Write a query to count the number of employees in each position.\*\*

```sql

SELECT position, COUNT(\*) AS num\_employees FROM employees GROUP BY position;

```

18. \*\*Write a query to find the highest salary in the `employees` table.\*\*

```sql

SELECT MAX(salary) AS highest\_salary FROM employees;

```

19. \*\*Write a query to retrieve employees who were hired in the year 2023.\*\*

```sql

SELECT \* FROM employees WHERE YEAR(hire\_date) = 2023;

```

20. \*\*Write a query to list the first 5 employees ordered by their hire date in descending order.\*\*

```sql

SELECT \* FROM employees ORDER BY hire\_date DESC LIMIT 5;

```

### Queries with JOINs

21. \*\*Write a query to join the `employees` table with a `departments` table on `department\_id` and retrieve employee names and department names.\*\*

```sql

SELECT e.name, d.department\_name

FROM employees e

JOIN departments d ON e.department\_id = d.id;

```

22. \*\*Write a query to find the total salary expense for each department.\*\*

```sql

SELECT d.department\_name, SUM(e.salary) AS total\_salary

FROM employees e

JOIN departments d ON e.department\_id = d.id

GROUP BY d.department\_name;

```

23. \*\*Write a query to find the name and position of employees who work in the 'Sales' department.\*\*

```sql

SELECT e.name, e.position

FROM employees e

JOIN departments d ON e.department\_id = d.id

WHERE d.department\_name = 'Sales';

```

24. \*\*

\*\*Write a query to find the number of employees in each department.\*\*

```sql

SELECT d.department\_name, COUNT(e.id) AS num\_employees

FROM employees e

JOIN departments d ON e.department\_id = d.id

GROUP BY d.department\_name;

```

25. \*\*Write a query to retrieve the distinct positions from the `employees` table.\*\*

```sql

SELECT DISTINCT position FROM employees;

```

### Advanced Queries with Aggregation and Filtering

26. \*\*Write a query to find the average salary of employees in each department.\*\*

```sql

SELECT d.department\_name, AVG(e.salary) AS average\_salary

FROM employees e

JOIN departments d ON e.department\_id = d.id

GROUP BY d.department\_name;

```

27. \*\*Write a query to find departments with an average salary greater than $60,000.\*\*

```sql

SELECT d.department\_name, AVG(e.salary) AS average\_salary

FROM employees e

JOIN departments d ON e.department\_id = d.id

GROUP BY d.department\_name

HAVING AVG(e.salary) > 60000;

```

28. \*\*Write a query to find the top 3 highest-paid employees.\*\*

```sql

SELECT \* FROM employees ORDER BY salary DESC LIMIT 3;

```

29. \*\*Write a query to count the number of distinct positions in the `employees` table.\*\*

```sql

SELECT COUNT(DISTINCT position) AS distinct\_positions FROM employees;

```

30. \*\*Write a query to find employees who were hired in the last 30 days.\*\*

```sql

SELECT \* FROM employees WHERE hire\_date >= DATE\_SUB(CURDATE(), INTERVAL 30 DAY);

```

31. \*\*Write a query to find the total number of employees hired each year.\*\*

```sql

SELECT YEAR(hire\_date) AS hire\_year, COUNT(\*) AS num\_employees

FROM employees

GROUP BY YEAR(hire\_date);

```

32. \*\*Write a query to find the second highest salary in the `employees` table.\*\*

```sql

SELECT DISTINCT salary

FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 1;

```

33. \*\*Write a query to find the total salary of employees whose name starts with 'A'.\*\*

```sql

SELECT SUM(salary) AS total\_salary

FROM employees

WHERE name LIKE 'A%';

```

34. \*\*Write a query to find the department with the highest total salary expense.\*\*

```sql

SELECT d.department\_name, SUM(e.salary) AS total\_salary

FROM employees e

JOIN departments d ON e.department\_id = d.id

GROUP BY d.department\_name

ORDER BY total\_salary DESC

LIMIT 1;

```

35. \*\*Write a query to retrieve employees who have the same position in the same department as another employee.\*\*

```sql

SELECT e1.name, e1.position, e1.department\_id

FROM employees e1

JOIN employees e2 ON e1.position = e2.position AND e1.department\_id = e2.department\_id AND e1.id != e2.id;

```

36. \*\*Write a query to find the minimum salary in each department.\*\*

```sql

SELECT d.department\_name, MIN(e.salary) AS min\_salary

FROM employees e

JOIN departments d ON e.department\_id = d.id

GROUP BY d.department\_name;

```

37. \*\*Write a query to find all employees who have a higher salary than the average salary of their department.\*\*

```sql

SELECT e.name, e.salary, d.department\_name

FROM employees e

JOIN departments d ON e.department\_id = d.id

WHERE e.salary > (SELECT AVG(e2.salary) FROM employees e2 WHERE e2.department\_id = e.department\_id);

```

38. \*\*Write a query to list all department names and the total number of employees in each, including departments with zero employees.\*\*

```sql

SELECT d.department\_name, COUNT(e.id) AS num\_employees

FROM departments d

LEFT JOIN employees e ON d.id = e.department\_id

GROUP BY d.department\_name;

```

39. \*\*Write a query to find the third highest salary in each department.\*\*

```sql

SELECT department\_id, salary

FROM (

SELECT e.department\_id, e.salary, DENSE\_RANK() OVER (PARTITION BY e.department\_id ORDER BY e.salary DESC) AS rank

FROM employees e

) AS ranked\_salaries

WHERE rank = 3;

```

40. \*\*Write a query to find the name of the employee with the highest salary in each department.\*\*

```sql

SELECT e.name, d.department\_name, e.salary

FROM employees e

JOIN departments d ON e.department\_id = d.id

WHERE e.salary = (

SELECT MAX(e2.salary)

FROM employees e2

WHERE e2.department\_id = e.department\_id

);

```

These questions cover a wide range of MySQL topics, including basic concepts, joins, aggregation functions, DISTINCT, LIMIT, WHERE, GROUP BY, and HAVING, providing a comprehensive test of SQL knowledge and skills.