Sure! Here are 25 questions covering basic and query topics in MySQL for a test:

### 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.