BOSSCODER ACADEMY

ADVANCED SQL CONCEPTS

Most Asked Interview Questions

01 QUESTION

How do you find the last ID in a SQL table?

In SQL, you can find the last ID in a table using the MAX function along with the column that represents the ID. Assuming you have an "id" column in your table, you can use the following query:

```SQL
SELECT MAX(id) AS last_id FROM your_table_name;
```

This query selects the maximum (largest) value in the "id" column and aliases it as "last_id." The result will be a single row with the highest ID value in the specified table.

Otherwise, in many SQL versions, we can use the following syntax:

```SQL
SELECT id
FROM your_table_name
ORDER BY id DESC
LIMIT 1;
```

This query selects the maximum (largest) value in the "id" column and aliases it as "last_id." The result will be a single row with the highest ID value in the specified table.

02 QUESTION

How do you remove duplicates from a table?

Using `DISTINCT`:

```SQL
SELECT DISTINCT * FROM your_table;
```

This will retrieve distinct rows from the table based on all columns. Keep in mind that this doesn't actually remove duplicates from the table; it just returns a result set with distinct values.

This query selects the maximum (largest) value in the "id" column and aliases it as "last_id." The result will be a single row with the highest ID value in the specified table.

Otherwise, in many SQL versions, we can use the following syntax:

Using `GROUP BY`:

```SQL
SELECT col1, col2, ..., colN, COUNT(*)
FROM your_table
GROUP BY col1, col2, ..., colN
HAVING COUNT(*) > 1;
```

03 QUESTION

Give the resulting tables arising from applying Joins on the following tables in SQL

Employees Table:

id	name	department_id
1	Alice	101
2	Bob	102
3	Charlie	101
4	David	103

Departments Table:

id	department_id
101	HR
102	IT
103	Marketing
104	Sales

04 QUESTION

What is difference between HAVING and WHERE in SQL?

GROUP BY Clause:

The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows, like categories.

It is often used in conjunction with aggregate functions (e.g., COUNT, SUM, AVG, MAX, MIN) to perform calculations on each group of rows.

It helps to create a result set where rows are grouped based on the values in specified columns.

Example:

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

In this example, the result set will have one row for each distinct department, and the average salary for each department will be calculated.

05 QUESTION

Explain about Auto Increment in SQL?

In SQL, the auto-increment feature is often used to generate unique, sequential numeric values automatically for a column. This is typically used for primary key columns to ensure each row has a unique identifier. The specific implementation of auto-increment may vary slightly between different database management systems (DBMS). Here are examples for some popular SQL database systems:

MySQL:

In MySQL, the auto-increment feature is achieved using the `AUTO_INCREMENT` attribute.

```SQL
CREATE TABLE example_table (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50)
);
INSERT INTO example_table (name) VALUES ('John');
INSERT INTO example_table (name) VALUES ('Jane');
-- The 'id' column will be automatically incremented
for each new record.
```

06 QUESTION

Explain about set operators in SQL

SQL provides several set operators that allow you to combine the results of multiple queries or tables. The main set operators include `UNION`, `UNION ALL`, `INTERSECT`, and `MINUS` (or `EXCEPT` in some database systems). Here's an explanation of each along with examples:

UNION:

The `UNION` operator is used to combine the result sets of two or more SELECT statements. It removes duplicate rows from the result set.

```SQL
SELECT column1, column2 FROM table1
UNION
SELECT column1, column2 FROM table2;
```

Example:

```SQL
-- Assuming two tables with similar structure
SELECT employee_id, employee_name FROM employees
UNION
SELECT vendor_id, vendor_name FROM vendors;
```

07 QUESTION

Give SQL query to retrieve nth record from an employee table

Method 1: Using LIMIT and OFFSET

```SQL
SELECT * FROM Employee
ORDER BY <column_name> -- Here, <column_name> is the column according to which the rows are ordered (e.g., ID).
OFFSET 5 ROWS            -- N - 1 = 6 - 1 = 5, so we skip the first 5 rows
FETCH NEXT 1 ROWS ONLY; -- Retrieve the next 1 row
```

Explanation:

- **ORDER BY:** It specifies the column based on which the ordering of rows is done.
- **OFFSET 5 ROWS:** It skips the first 5 rows, so the starting point is the 6th row.
- **FETCH NEXT 1 ROWS ONLY:** It retrieves the next 1 row after the offset, giving you the 6th row.

08 QUESTION

Explain how to get unique records without using the DISTINCT keyword.

Let's consider an example with a hypothetical table called `employee` that contains information about employees, including their `employee_id` and `employee_name`. In this example, we want to retrieve distinct rows without using the DISTINCT clause, and we'll demonstrate four different approaches.

Assuming the `employee` table looks like this:

SQL

```sql
CREATE TABLE employee (
    employee_id INT,
    employee_name VARCHAR(255)
);

INSERT INTO employee VALUES (1, 'John Doe');
INSERT INTO employee VALUES (2, 'Jane Doe');
INSERT INTO employee VALUES (3, 'John Doe');
INSERT INTO employee VALUES (4, 'Bob Smith');
INSERT INTO employee VALUES (5, 'Jane Doe');
```