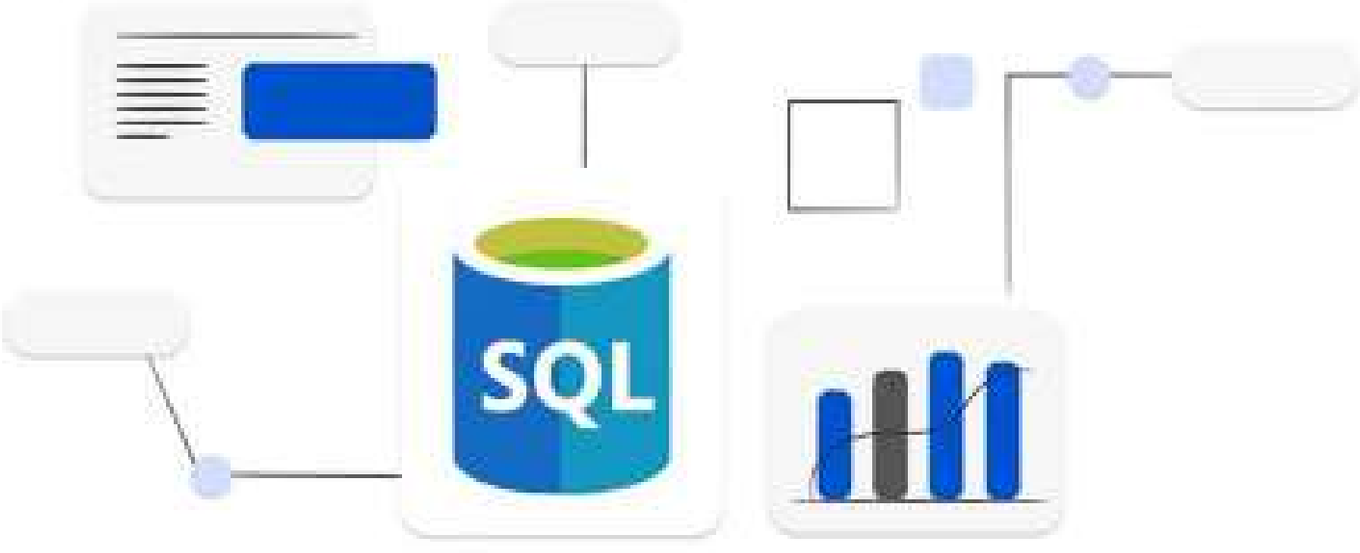
ADVANCED SQL CONCEPTS



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;
```

This will group the rows by specified columns and count the occurrences. Rows with a count greater than 1 are duplicates.

Using `ROW_NUMBER()` with Common Table Expressions (CTE):

```sql
WITH CTE AS (
    SELECT *,
           ROW_NUMBER() OVER (PARTITION BY col1, col2, ..., colN ORDER BY (SELECT 0)) AS rn
    FROM your_table
)
DELETE FROM CTE WHERE rn > 1;
```

This method uses the ROW_NUMBER() window function to assign a unique number to each row within a partition. Rows with rn > 1 are duplicates.

Using `INNER JOIN`:

```sql
DELETE t1
FROM your_table t1
INNER JOIN your_table t2
WHERE t1.id > t2.id
AND t1.col1 = t2.col1
```

```
AND t1.col2 = t2.col2
AND ...;
```

This query deletes duplicates based on specified columns, keeping the row with the lowest ID.

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

Inner Join:

- Returns only the rows with matching values in both tables.
- Filters out rows with no match.

SQL Query:

```SQL
SELECT employees.name, departments.department_name
FROM employees
INNER JOIN departments ON employees.department_id =
departments.id;
```

Output:

name	department_name
Alice	HR
Bob	IT
Charlie	HR
David	Marketing

Left Join (Left Outer Join):

- Returns all rows from the left table and the matched rows from the right table.
- If there is no match in the right table, NULL values are returned.

SQL Query:

```SQL
SELECT employees.name, departments.department_name
FROM employees
LEFT JOIN departments ON employees.department_id =
departments.id;
```

Output:

name	department_name
Alice	HR
Bob	IT
Charlie	HR
David	Marketing

Right Join (Right Outer Join):

- Returns all rows from the right table and the matched rows from the left table.
- If there is no match in the left table, NULL values are returned.

SQL Query:

```SQL
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments ON employees.department_id =
departments.id;
```

Output:

name	department_name
Alice	HR
Bob	IT
Charlie	HR
David	Marketing
NULL	Sales

Full Outer Join:

- Returns all rows when there is a match in either the left or right table.
- Includes rows with no match in either table with NULL values.

SQL Query:

```SQL
SELECT employees.name, departments.department_name
FROM employees
FULL OUTER JOIN departments ON
employees.department_id = departments.id;
```

Output:

name	department_name
Alice	HR
Bob	IT
Charlie	HR
David	Marketing
NULL	Sales

Self Join:

- Combines rows from a single table, treating it as two separate tables.
- Often used for hierarchical data.

SQL Query:

```SQL
SELECT e1.name, e2.name AS manager
FROM employees e1
LEFT JOIN employees e2 ON e1.manager_id = e2.id;
```

Output:

name	manager
Alice	NULL
Bob	NULL
Charlie	Alice
David	NULL