ChatGPT ⌄

> In which order sql query is executed, who has join, where, orderby
> and my have other attributes, explain with example

In SQL, the order of execution for a query that includes various clauses such as `JOIN`, `WHERE`, `ORDER BY`, and others follows a specific logical sequence, which is different from the written order. The typical order of execution is as follows:

1. **FROM** (including `JOIN`)
2. **WHERE**
3. **GROUP BY**
4. **HAVING**
5. **SELECT**
6. **ORDER BY**
7. **LIMIT/OFFSET**

Here's a detailed explanation of each step with an example query:

Example Query

```sql
SELECT department, COUNT(*) as employee_count
FROM employees
JOIN departments ON employees.department_id = departments.id
WHERE employees.salary > 50000
GROUP BY department
HAVING COUNT(*) > 5
ORDER BY employee_count DESC
LIMIT 10;
```

Execution Order

1. **FROM (including JOIN)**:

   - The `FROM` clause identifies the tables to be queried.
   - Any `JOIN` operations to combine tables occur in this step.
   - Example: Combine `employees` and `departments` tables based on the `department_id`.

   ```sql
   FROM employees
   JOIN departments ON employees.department_id = departments.id
   ```

2. **WHERE**:

   - The `WHERE` clause filters the rows based on specified conditions.
   - Example: Select rows where `employees.salary` is greater than 50,000.

   ```sql
   WHERE employees.salary > 50000
   ```

3. **GROUP BY**:

   - The `GROUP BY` clause groups the rows that have the same values in specified columns into aggregated data.
   - Example: Group rows by `department`.

   ```sql
   GROUP BY department
   ```

4. **HAVING**:

- The `HAVING` clause filters groups based on aggregate functions.
- Example: Filter groups where the count of employees is greater than 5.

```sql
HAVING COUNT(*) > 5
```

5. **SELECT**:

- The `SELECT` clause specifies the columns to be returned in the result set.
- Example: Select the `department` and the count of employees as `employee_count`.

```sql
SELECT department, COUNT(*) as employee_count
```

6. **ORDER BY**:

- The `ORDER BY` clause sorts the result set based on one or more columns.
- Example: Order the results by `employee_count` in descending order.

```sql
ORDER BY employee_count DESC
```

7. **LIMIT/OFF