SQL - Aggregations & Sorting

Kodkod 4

Overview

- Use aggregation functions to summarize data.
- Group rows using to analyze data by category.
- Filter grouped results.
- Sort query results in ascending or descending order.

COUNT() - Counting Rows

- So far, we've queried tables and retrieved data based on specific columns and conditions.
- Now we can aggregate data — summarize multiple rows into a single value.

```
SELECT COUNT(*)
FROM users
WHERE age > 100;
```

Other Common Aggregate Functions

- Just like COUNT(), SQL provides additional functions to perform calculations on numeric data.
  - `MIN()` → finds the lowest value
  - `MAX()` → finds the highest value
  - `AVG()` → calculates the average value

```sql
-- Lowest payment
SELECT MIN(amount)
FROM payments;
```

```sql
-- Highest payment
SELECT MAX(amount)
FROM payments;
```

```sql
-- Average payment
SELECT AVG(amount)
FROM payments;
```

Class Exercises

1. How many customers live in France or Spain?
2. How many employees have first names that start with A or B?
3. Find the minimum payment amount in the payments table.
4. Find the maximum payment amount in the payments table.
5. Find the average payment amount in the payments table.

GROUP BY

- When we want to summarize data per category - for example, count how many orders each customer has - we use `GROUP BY`.
- `GROUP BY` groups rows that have the same value in one or more columns.
- Aggregate functions like `COUNT()`, `SUM()`, `AVG()` run on each group separately.

```
SELECT customerNumber, COUNT(*) AS total_orders
FROM orders
GROUP BY customerNumber;
```

ORDER BY

- After retrieving or grouping data, you can sort the results using `ORDER BY`.
- `ORDER BY` sorts the output by one or more columns.
- By default it sorts ascending (`ASC`) - from smallest to largest. To reverse the order, use `DESC` (descending).

```
SELECT customerNumber, COUNT(*) AS total_orders
FROM orders
GROUP BY customerNumber
ORDER BY total_orders DESC;
```

HAVING

- `HAVING` is used to filter groups that were created with `GROUP BY`.
- It applies conditions after the grouping is done.
- Use `HAVING` when your condition involves an aggregate function such as `COUNT()`, `SUM()`, `AVG()`, `MIN()`, or `MAX()`.

```sql
SELECT customerNumber, COUNT(*) AS total_orders
FROM orders
GROUP BY customerNumber
HAVING COUNT(*) > 5;
```

Class Exercises

1. Find how many orders each customer has.
2. Sort the results so that the customers with the most orders appear first.
3. Show only customers who have placed more than 5 orders.

Summary

- Learned to use aggregate functions to summarize data.
- Used GROUP BY to organize rows into groups and calculate values per group.
- Applied HAVING to filter groups after aggregation (used with functions like COUNT or SUM).
- Used ORDER BY to sort results ascending (ASC) or descending (DESC).