Aggregate Functions in SQL

Hia Al Saleh

October 28th, 2024

Contents

1 Introduction to Aggregate Functions

Aggregate functions, also called **set functions**, operate on a group of values to produce a single, summarizing value. Aggregates can be applied to:

- All rows in a table
- Only specific rows, using a `WHERE` clause
- Rows grouped by a `GROUP BY` clause

Non-aggregate queries process each row independently, while aggregate queries process the table as a whole to construct new rows.

2 Types of Aggregate Functions

2.1 COUNT

The `COUNT` function counts the number of records that match a certain criterion.

- `COUNT(*)` - returns the total number of rows in a table, including duplicates and `NULL`s.
- `COUNT(column_name)` - returns the number of non-`NULL` values in the specified column.

Example:

```
SELECT COUNT(*) AS Num_Books FROM titles;
```

2.2 MIN and MAX

- `MIN(column_name)` - returns the minimum value in the specified column.
- `MAX(column_name)` - returns the maximum value.

These functions work with character, numeric, and datetime types. Note that `DISTINCT` has no meaning in `MIN` and `MAX`. Examples:

```
SELECT MIN(dateOrdered) FROM orders;
SELECT MAX(shippedTo) FROM orders;
```

2.3 SUM

The `SUM(column_name)` function calculates the sum of numeric values in the specified column.

- It works only with numeric data types.
- `NULL` values are ignored.

- If there are no rows to sum, it returns `NULL` (not zero).

Example:

```
SELECT SUM(price) FROM carInventory WHERE qtyInStock = 1;
```

2.4 AVG

The `AVG(column_name)` function calculates the average of numeric values in the specified column.

- Works only on numeric data types.
- If no rows match, it returns `NULL`.

Example:

```
SELECT AVG(price) FROM carInventory WHERE make = 'Mazda';
```

3 Other Considerations with Aggregate Functions

- **Ignoring `NULL`s**: All aggregate functions except `COUNT(*)` ignore `NULL` values.
- **Aliases**: Use aliases to provide meaningful names for aggregate results in the `SELECT` clause.

4 Restrictions on Aggregate Functions

- Aggregate functions cannot be used in the `WHERE` clause. Attempting to use one will result in an error.
- You cannot mix non-aggregate and aggregate columns directly in the `SELECT` clause without using a `GROUP BY`.
- Nested aggregate functions are not allowed. For example:

  ```
  SELECT SUM(AVG(sales)) FROM titles;
  ```

- Subqueries cannot be used within aggregate expressions.

5 Using GROUP BY with Aggregate Functions

The GROUP BY clause is used to divide a table into groups and apply aggregate functions to each group. The database processes rows by:

- Gathering all information from the FROM clause
- Filtering by the WHERE clause, if present
- Aggregating the rows into groups

Only grouped rows can be used in the SELECT clause. Examples:

```
SELECT make, COUNT(*) AS numberOf
FROM carInventory
GROUP BY make;

SELECT make, modelYear, COUNT(*) AS numberOf
FROM carInventory
GROUP BY make, modelYear;
```

6 DISTINCT with Aggregate Functions

DISTINCT can be used to eliminate duplicate values within aggregate functions, but it's not meaningful with MIN and MAX. Example:

```
SELECT COUNT(DISTINCT make) FROM carInventory;
```

7 HAVING Clause

The HAVING clause filters results after aggregation, working similarly to the WHERE clause but applying to aggregated data. It is optional and used only with GROUP BY. Rows removed by the WHERE clause are not available to the HAVING clause.

```
SELECT make, SUM(price) AS totalPrice
FROM carInventory
GROUP BY make
HAVING SUM(price) > 2000000;
```

8 Summary of Aggregate Function Usage

Aggregate functions enable summarizing data in SQL tables:

- COUNT, SUM, and AVG for numeric summaries.
- MIN and MAX for finding boundary values.

GROUP BY and HAVING clauses allow further control and filtering of results at the group level, making aggregate functions versatile tools in data summarization.