Theory: Aggregate functions

19 minutes 0 / 5 problems solved

Imagine you are a business analyst in a financial consulting company. You start your day by preparing a report about what happened with the stock market yesterday. Information about the committed deals is stored in tables with hundreds of thousands rows in an SQL-compatible database. Simply selecting all data from the columns won't work since performing the exact computation over such volume of data is unrealistic. Regular SQL functions won't help much since they are applied to each cell individually and provide no means of combining their results. What you can do instead is apply **aggregate functions**, and this topic is all about them.

§1. General form

To execute an aggregate function and pass all values from a column to it, use the following syntax:

```
SELECT AGG_FUNCTION(column_name) FROM table_name;
```

The function will take all values from the specified column and produce a single cell as a result. Consequently, running *n* aggregate functions simultaneously produces *n* cells.

Vendors of different SQL-compatible database management systems provide their users with slightly different sets of aggregate functions. Here is a list of aggregate functions according to the ANSI SQL standard:

- MIN
- MAX
- AVG
- COUNT
- SUM

There are also a few more elaborate functions related to statistics such as `STDDEV_POP` for the population standard deviation or `CORR` for the correlation coefficient. It may be worth your while to check out the full list of aggregate functions.

§2. Example

Consider this table named *stocks*:

stock_name	price	yesterday_deals
WTI	89.8	NULL
NVAX	26.3	5
GSPC	18.9	20
DJI	40	2
NYSE	15.6	13
TCHENY	52.7	5
FB	63.7	20

Using the `MAX` function, we can easily find the highest price among all the stocks:

```
SELECT MAX(price) FROM stocks;
```

This query will produce `89.8`. Likewise, the `MIN` function for the same column would give us `15.6`. If we want to know the count of deals made yesterday, we can use this query:

Current topic:

Aggregate functions

Topic depends on:

SELECT FROM WHERE statement

Topic is required for:

GROUP BY statement

Table of contents:

```
SELECT SUM(yesterday_deals) FROM stocks;
```

It will return `65`.

Numeric types like INT or REAL are a natural fit for most aggregate functions. However, in some cases it also makes sense to use aggregate functions with other data types. For example, the `MIN` and `MAX` functions can be used to find the lexicographically smallest and largest strings respectively. We encourage you to experiment and see how different aggregate functions behave with other types of data!

§3. Adding WHERE

It is also possible to use `WHERE` to choose a subset of rows on which we want to run our aggregation functions. For example, let's find the average price and average count of deals for all stocks that cost more than 40:

```
SELECT AVG(price) as avg_price, AVG(yesterday_deals) as avg_deals
FROM stocks
WHERE price > 40;
```

The answer will be:

avg_price	avg_deals
68.733333333	12.5

§4. DISTINCT keyword

When working with large amounts of data, you might be interested in omitting all duplicate values. To do that, place the `DISTINCT` keyword inside the brackets of your aggregate function:

```
SELECT COUNT(DISTINCT yesterday_deals) FROM stocks;
```

This query will return `4` as there are only 4 distinct numeric values in the column *yesterday_deals*.

Determining the set of unique values can be costly in terms of computation complexity, so don't overuse `DISTINCT`. Always think about whether you really need to omit the duplicates and how it can affect the final result.

§5. COUNT(*)

A regular call of the `COUNT` function with a column name as an argument will simply count the total amount of values in the column. If you call `COUNT` with an asterisk, then you're telling the function to count *all* rows that exist in the table. The final result won't be affected by the particular types of columns or the values that their cells store. For our *stocks* table, `COUNT(*)` will return `7`.

All aggregate functions except COUNT(*) ignore the NULL values.

§6. Conclusion

In this topic, we went through some standard aggregate functions and examined their special features. It is hard to imagine working in a serious production environment without knowing the aggregate functions well. Let's move on to practical tasks and check how you've grasped them!

Report a typo

51 users liked this theory. **0** didn't like it. **What about you?**

Start practicing

Comments (2) Hints (0) Useful links (0) Show discussion