6 SQL Queries Every Data Engineer Should Be Aware of

It might be more than 45 years old, but SQL still gets the job done

[[](https://cinto-sunny.medium.com/?source=post_page-----2d0a2cc5986e--------------------------------)](https://cinto-sunny.medium.com/?source=post_page-----2d0a2cc5986e--------------------------------)

[Cinto](https://cinto-sunny.medium.com/?source=post_page-----2d0a2cc5986e--------------------------------)

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F80f4993f79be%2F2d0a2cc5986e&operation=register&redirect=https%3A%2F%2Fbetterprogramming.pub%2F6-sql-queries-every-data-engineer-should-be-aware-of-2d0a2cc5986e&user=Cinto&userId=80f4993f79be&source=post_page-80f4993f79be----2d0a2cc5986e---------------------follow_byline-----------)

[Oct 7](https://betterprogramming.pub/6-sql-queries-every-data-engineer-should-be-aware-of-2d0a2cc5986e?source=post_page-----2d0a2cc5986e--------------------------------) · 4 min read

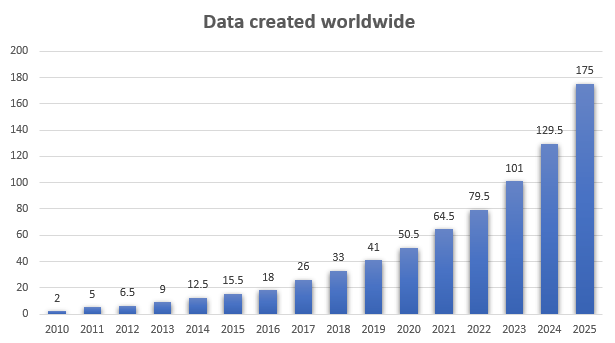
Obsah obrázku text

Popis byl vytvořen automaticky

Photo by [Markus Spiske](https://unsplash.com/@markusspiske?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/software-engineer?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

Whether you are a beginner starting your engineering career, or you are an experienced data engineer or data analyst, knowledge of advanced SQL syntax is a must.

With exponential growth in data, it has become more important to analyze these data super quickly.



Source: [*Statista*](https://www.statista.com/statistics/871513/worldwide-data-created/)

The units in this graph are zettabytes.

1 zettabyte = 1 trillion gigabytes

And people may say that SQL is dead, but the reality is that there is no system currently to replace it currently. There are many very capable NoSQL stores that do their jobs very well, supporting massive scale out with low costs. However, they don’t replace high-quality SQL-based stores — they complement them. The ACID properties of SQL make it a highly reliable way to model data relatively naturally.

As a data engineer myself, I have been using SQL for a while, and I know the importance of writing complex queries faster. So, here is some advanced SQL syntax that will surely come in handy.

For the below examples, I have used the below table content. The table is called “*bill*”.

Obsah obrázku stůl

Popis byl vytvořen automaticky

Image by author

Running Totals

You often come across scenarios where you have to calculate a running total from a table. This is to know what each value is, against a running total.

A running total refers to the sum of values in all cells of a column that precedes the next cell in that particular column.

Here is a query to do that.

|  |
| --- |
| SELECT id,month |
|  | , Amount |
|  | , SUM(Amount) OVER (ORDER BY id) as total\_sum |
|  | FROM bill |

And, this is how the output will look like:

Obsah obrázku stůl

Popis byl vytvořen automaticky

Image by author

Common Table Expressions

The Common Table Expressions or CTE’s for short are used to simplify the readability of complex joins and subqueries.

It is basically a temporary named result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement.

Consider this simple query,

|  |
| --- |
| SELECT \* |
|  | FROM bill |
|  | WHERE id in |
|  | (SELECT DISTINCT id |
|  | FROM id |
|  | WHERE country = "US" |
|  | AND status = "Y" |
|  | ) |

Now imagine if we are using this subquery multiple times in the subsequent query. Won’t it be easier if we can use that as a temporary table? CTE solves this exact problem.

|  |
| --- |
| WITH idtempp as ( |
|  | SELECT id as id |
|  | FROM id |
|  | WHERE country = "US" |
|  | AND status = "Y" |
|  | ) |
|  |  |
|  | SELECT \* |
|  | FROM bill |
|  | WHERE id in (SELECT id from idtempp) |

This is a small example, but this generally can be really useful for larger and complex subqueries.

Ranking the Data

Data engineers and analysts would agree that it is very common to rank values based on some parameter like salaries or expense, etc. And having the knowledge of ranking data at your fingertips can save you a lot of time finding out the exact query.

|  |
| --- |
| SELECT |
|  | id, |
|  | Amount, |
|  | RANK() OVER (ORDER BY Amount desc) |
|  | FROM bill |

In the below query, I have ranked the dataset based on the amount column.

You can also use DENSE\_RANK() which is similar to RANK() except that it doesn't skip the subsequent rank if two rows have the same value.

Adding Subtotals

Again, a super important query for data engineers and analysts. In my 10 year career working as a business/data analyst, I have used this query for a lot of analysis. Having a subtotal helps you put the data in perspective of the total.

It’s an extension of a GROUP BY clause with the ability to add subtotals and grand totals to your data.

|  |
| --- |
| SELECT |
|  | Type, |
|  | id, |
|  | SUM (Amount) AS total\_amount |
|  | FROM bill |
|  | GROUP BY Type,id WITH ROLLUP |

Obsah obrázku stůl

Popis byl vytvořen automaticky

Image by author

*Note:*the above query is in MySQL. The rollup syntax may vary for others.

In the above query, the rows where both type and id are null is the one that is the total. You also have subtotals irrespective of the id column. That is represented by the 4th row and the second last row.

Temporary Functions

Temporary functions allow you to modify the data easily without writing huge case statements.

In the below example, a temporary function is used to convert type to gender. This can be done using case statement inline in the query, but it would have been messy to read

|  |
| --- |
| CREATE TEMPORARY FUNCTION get\_gender(type varchar) AS ( |
|  | CASE WHEN type = "M" THEN "male" |
|  | WHEN type = "F" THEN "female" |
|  | ELSE "n/a" |
|  | END |
|  | ) |
|  | SELECT |
|  | name, |
|  | get\_gender(Type) as gender |
|  | FROM bill |

Variance and Standard Deviation

For data scientists and analysts, having the ability to get the variance and the standard deviation is crucial. Thankfully, there are functions to get these values.

The VARIANCE, VAR\_POP, and VAR\_SAMP are aggregate functions i.e they group the data. These are utilized to determine the variance, group variance, and sample variance of a collection of data individually.

|  |
| --- |
| SELECT |
|  | VARIANCE(amount) AS var\_amount, |
|  | VAR\_POP(amount) AS var\_pop\_amount, |
|  | VAR\_SAMP(amount) AS var\_samp\_amount, |
|  | STDDEV\_SAMP(amount) as stddev\_sample\_amount, |
|  | STDDEV\_POP(amount) as stddev\_pop\_amount, |
|  | FROM bill |

VAR\_POP: This is the population variance  
VAR\_SAMP: This is the sample variance  
STDDEV\_SAMP: This is the sample standard deviation  
STDDEV\_POP: This is the population standard deviation

These are some of the top SQL commands that I have constantly used in my data engineering career. These have come in very handy for solving a lot of business problems. [Stats](https://thenewstack.io/sql-is-dead-right/) say that the ecosystem of SQL tools that includes anything from Excel and Tableau to SparkSQL is used by [more than 60%](https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/) of organizations. A very impressive feat, especially considering its age.

So, if you are a data engineer, I am sure you will find these commands useful. Let me know in the comments section if I have missed anything.