**Advanced SQL techniques for beginners**

**On a scale from 1 to 10 how good are your data warehousing skills?**

[Github](https://gist.github.com/638f8f09346769ad480d11764f11975c)

<https://gist.github.com/mshakhomirov/638f8f09346769ad480d11764f11975c>



Photo by [Jess Bailey](https://unsplash.com/@jessbaileydesigns?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

On a scale from 1 to 10 how good are your data warehousing skills? Want to go above 7/10? This article is for you.

Want to get ready for a data analyst job interview asap? This blog post explains some intricate data warehouse SQL techniques in detail. I will use BigQuery standard SQL dialect to scribble down a few thoughts on this topic.

**1. Incremental tables and MERGE**

Updating table effectively in the right way is important. It is important indeed. Ideal situation is when you have transactions that are a PRIMARY key, unique integers and auto increment. Table update in this case is simple:

That is not always the case when working with denormalized star-schema datasets in modern data warehouses. You might be tasked to create **sessions** with SQL and/or incrementally update datasets with just a portion of data. transaction\_id might not exist but instead you will have to deal with data model where unique key depends on the latest transaction\_id (or timestamp) known. For example, user\_id in last\_online dataset depends on the latest known connection timestamp. In this case you would want to update existing users and insert the new ones.

**MERGE and incremental updates**

You can use **MERGE** or you can split the operation into two actions. One to update existing records with new ones and one to insert completely new ones that don’t exits (LEFT JOIN situation).

**MERGE** is a statement that is generally used in relational databases. Google BigQuery MERGE Command is one of the Data Manipulation Language (DML) statements. It is often used to perform three main functions atomically in one single statement. These functions are UPDATE, INSERT, and DELETE.

* UPDATE or DELETE clause can be used when two or more data match.
* INSERT clause can be used when two or more data are different and do not match.
* The UPDATE or DELETE clause can also be used when the given data does not match the source.

This means that the Google BigQuery MERGE command let you to merge Google BigQuery data by updating, inserting, and deleting data from your Google BigQuery tables.

**Consider this SQL:**

Here we did UPDATE and INSERT in just one operation.

**2. Counting words**

Doing UNNEST() and check if the word you need is in the list you need migth be useful in many situation, i.e. data warehouse sentiment analysis:

**3. Using IF() statement outside of the SELECT statement**

This gives us an opportunity to save some lines of code and be more eloquent code-wise. Normally you would want to put this into a sub-query, and add a filter in the **where** clause but you can do **this** instead:

Another example how **NOT** to use it with **partitioned** tables. **Don’t do this**. This is bad example because since the matching table suffixes are probably determined dynamically (based on something in your table) you **will be charged for a full table scan.**

You can also use it in HAVING clause and AGGREGATE functions.

**4. Using GROUP BY ROLLUP**

The ROLLUP function is used to perform aggregation at multiple levels. This is useful when you have to work with dimension graphs.

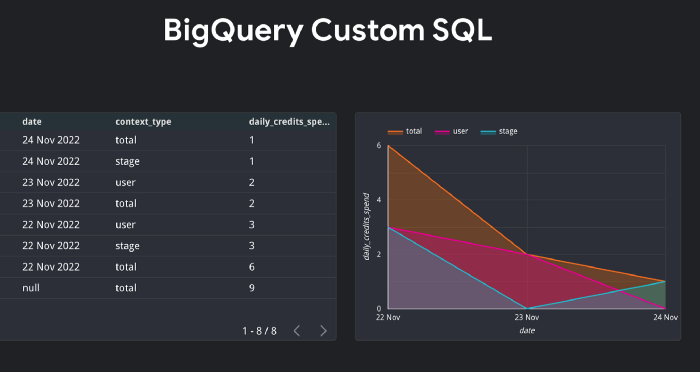


Image by author

The following query returns the total credit spend per day by the transaction type (is\_gift) specified in the **where** clause, and it also shows the total spend for each day and the total spend in all the dates available.

**5. Convert a table into JSON**

Imagine you are required to convert your table into JSON object where each record is an element of nested array. This is where to\_json\_string() function becomes useful:

Then you can use it anywhere: dates, marketing funnels, indices, histogram graphs, etc.

**6. Using PARTITION BY**

Given user\_id, date and total\_cost columns. For EACH date, how do you show the total revenue value for EACH customer while keeping all the rows? You can achieve this like so:

**7. Moving average**

Very often BI developers are tasked to add a moving average to reports and their fantastic dashboards. This might be 7, 14, 30 day/month or even year MA line graph. So how do we do it?

**8. Date arrays**

Becomes really handy when you work with **user retention** or want to check some dataset for missing values, i.e. dates. BigQuery has a function called GENERATE\_DATE\_ARRAY:

I previously wrote about it here:

**[Retention and Daily Active Users Explained.](https://towardsdatascience.com/retention-and-daily-active-users-explained-79c5dab1bf84" \t "_blank)**

**[Complete Data Studio guide and BigQuery tutorial for Firebase users, Machine Learning enthusiasts and Marketers. All…](https://towardsdatascience.com/retention-and-daily-active-users-explained-79c5dab1bf84" \t "_blank)**

[towardsdatascience.com](https://towardsdatascience.com/retention-and-daily-active-users-explained-79c5dab1bf84" \t "_blank)

**9. Row\_number()**

This is useful to get something latest from your data, i.e. latest updated record, etc. or even to remove duplicates:

**10. NTILE()**

Another numbering function. Really useful to monitor things like Login duration in seconds if you have a mobile app. For example, I have my App connected to Firebase and when users login I can see how long it took for them.

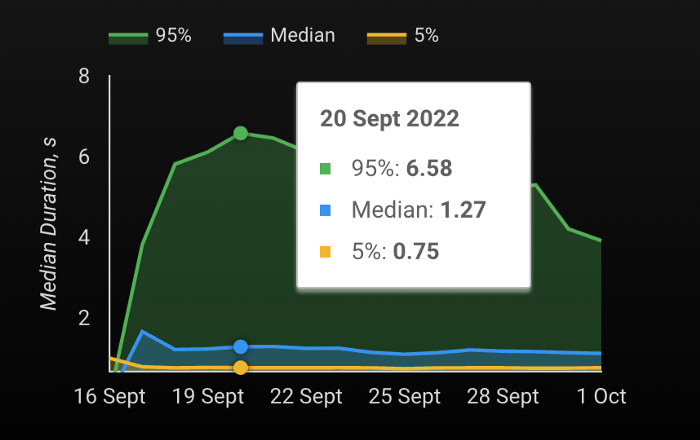


Image by author

This function divides the rows into constant\_integer\_expression buckets based on row ordering and returns the 1-based bucket number that is assigned to each row. The number of rows in the buckets can differ by at most 1. The remainder values (the remainder of number of rows divided by buckets) are distributed one for each bucket, starting with bucket 1. If constant\_integer\_expression evaluates to NULL, 0 or negative, an error is provided.

In this article you can find a tutorial how to create custom user activity and aperformance dashboards using Firebase and Google Analytics 4 data:

**[I stopped using Firebase Dashboards. I’ve built my own instead.](https://towardsdatascience.com/i-stopped-using-firebase-dashboards-ive-built-my-own-instead-299631185b41" \t "_blank)**

**[How would you explain Firebase figures? Here is the answers. Neat template included as well as sample Firebase datasets…](https://towardsdatascience.com/i-stopped-using-firebase-dashboards-ive-built-my-own-instead-299631185b41" \t "_blank)**

[towardsdatascience.com](https://towardsdatascience.com/i-stopped-using-firebase-dashboards-ive-built-my-own-instead-299631185b41" \t "_blank)

**11. Rank / dense\_rank**

They are also called **numbering** functions. I tend to use DENSE\_RANK **as default ranking function** as it doesn't skip the next available ranking whereas RANK would. It returns consecutive rank values. You can use it with a partition which divides the results into distinct buckets. Rows in each partition receive the same ranks if they have the same values. **Example:**

**Another example with product prices:**

**12. Pivot / unpivot**

Pivot changes rows to columns. It’s all it does. Unpivot does the [opposite](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#unpivot_operator). And finally it is available in BigQuery:

**13. First\_value / last\_value**

That’s another useful function which helps to get a delta for each row against the first / last value in that particular partition.

**14. Convert a table into Array of structs and pass them to UDF**

This is useful when you need to apply a user defined function (UDF) with some complex logic to each row or a table. You can always consider your table as an array of TYPE STRUCT objects and then pass each one of them to UDF. It depends on your logic. For example, I use it to calculate purchase expire times:

In a similar way you can create tables with no need to use **UNION ALL**. For example, I use it to mock some test data for unit tests. This way you can do it very fast just by using Alt+ Shift+ Down in your editor.

**[SQL Unit Testing in BigQuery? Here is a tutorial.](https://towardsdatascience.com/sql-unit-testing-in-bigquery-here-is-a-tutorial-64fc7a6b377" \t "_blank)**

**[Complete guide for scripting and UDF testing.](https://towardsdatascience.com/sql-unit-testing-in-bigquery-here-is-a-tutorial-64fc7a6b377" \t "_blank)**

[towardsdatascience.com](https://towardsdatascience.com/sql-unit-testing-in-bigquery-here-is-a-tutorial-64fc7a6b377" \t "_blank)

**15. Creating event funnels using FOLLOWING AND UNBOUNDED FOLLOWING**

Good example might be marketing funnels. Your dataset might contain continiously repeating events of the same type but ideally you would want to chain each event with next one of a different type. This might be useful when you need to get a list of something, i.e. events, purchases, etc. in order to build a funnels dataset. Working with PARTITION BY it gives you the opportunity to group all the follwoing events no matter how many of them exists ineach partition.

**16. Regexp**

You would to use it if you need to extract something from unstructured data, i.e. fx rates, custom groupings, etc.

**Working with currency exchange rates using regexp**

Consider this example with exchange rates data:

**Working with App versions using regexp**

Sometimes you might want to use regexp to get **major**, **release** or **mod** versions for your app and a create a custom report:

**Conclusion**

SQL is a powerful tool that helps to manipulate data. Hopefuly these SQL use cases from digital marketing will be useful for you. It’s a handy skill indeed and can help you with many projects. These SQL snippets made my life a lot easier and I use at work alomost every day. More, SQL and modern data warehouses are essentials tools for data science. Its robust dialect features allow to model and visualize data with ease. Because SQL is the language that data warehouses and business intelligence professionals use, it’s an excellent selection if you want to share data with them. It is the most common way to communicate with almost every data warehouse / lake solution in the market.

**Recommended read:**

[1] <https://cloud.google.com/bigquery/docs/reference/standard-sql/dml-syntax#merge_statement>

[6] <https://towardsdatascience.com/how-to-compute-a-moving-average-in-bigquery-using-sql-15f3fedd7489>

[11] <https://cloud.google.com/bigquery/docs/reference/standard-sql/window-function-calls>

[12] <https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#unpivot_operator>

[14] <https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators#first_value>