# SQL CODE CHALLENGE

**By,**

**Shamini Puthooppallil**

Contents

[SQL CODE CHALLENGE 1](#_Toc40350399)

[Problem Statement 3](#_Toc40350400)

[Input data 4](#_Toc40350401)

[Platform 5](#_Toc40350402)

[BigQuery 5](#_Toc40350403)

[Google S3 Bucket 6](#_Toc40350404)

[Data Import 6](#_Toc40350405)

[Solution 8](#_Toc40350406)

[STEP 1 8](#_Toc40350407)

[STEP 2 8](#_Toc40350408)

[SQL Query Explanation 9](#_Toc40350409)

[Output and Code Save 10](#_Toc40350410)

[Appendix 12](#_Toc40350411)

[1. Input Data 12](#_Toc40350412)

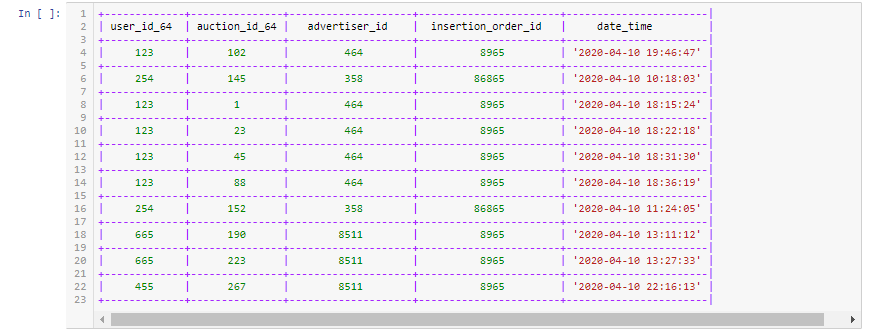
[2. Query 12](#_Toc40350413)

[3. Output 12](#_Toc40350414)

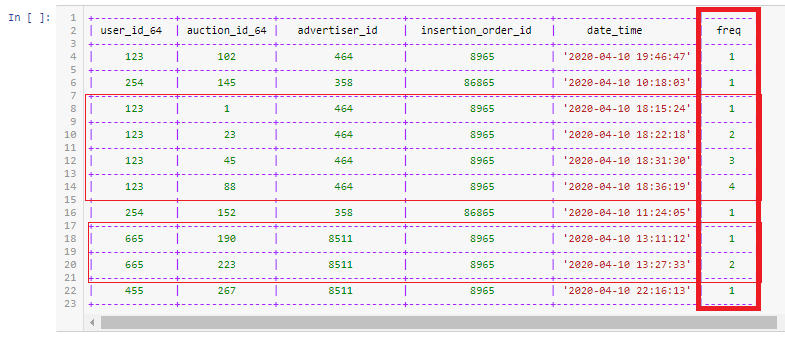
## Problem Statement

Identify and rank impressions in the table where any given user\_id\_64 received more than 1 impression in any given 60 minutes interval for the same advertiser\_id and insertion\_order\_id:

Input



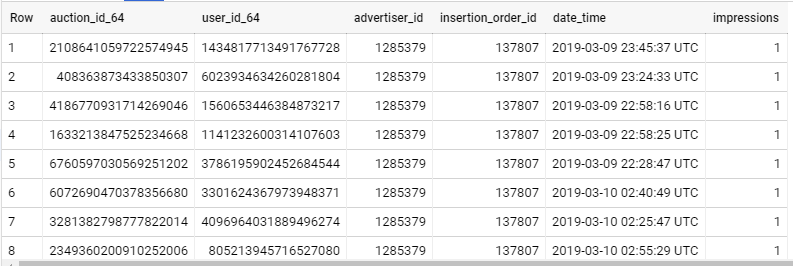
Expected output



Counting doesn’t necessarily start at clock hours (18:00:00 or 21:00:00) but a rolling 60 minutes interval of any impression for any user\_id\_64, advertiser\_id and insertion\_order\_id.

## Input data

Here is the sample data



Field Description

auction\_id\_64 -Primary key in this table

user\_id\_64 -User identification number which is unique for each user

advertiser\_id - Indicates each advertisement ,unique for each advertisement.

date\_time -Time in which the impression received

Impressions -Flag which indicates impression for particular advertisement

Please find the data attached at [Appendix 1](#_Input_Data)

## Platform

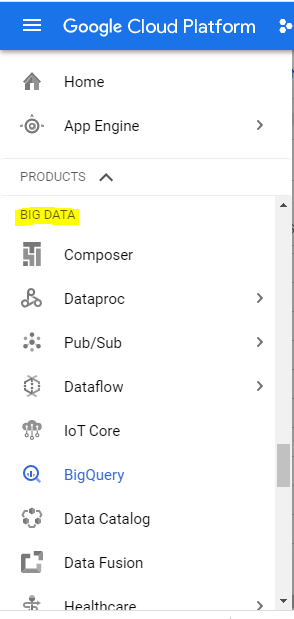
Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail and YouTube

In this analysis I am using some of the Google cloud services to store and analyse and save the desired output. Before starting the analysis, I sign up with GCP to use some of the essential services they provide.

### ****BigQuery****

The **BigQuery** service replaces the typical hardware setup for a traditional data warehouse. That is, it serves as a collective home for all analytical data in an organization.

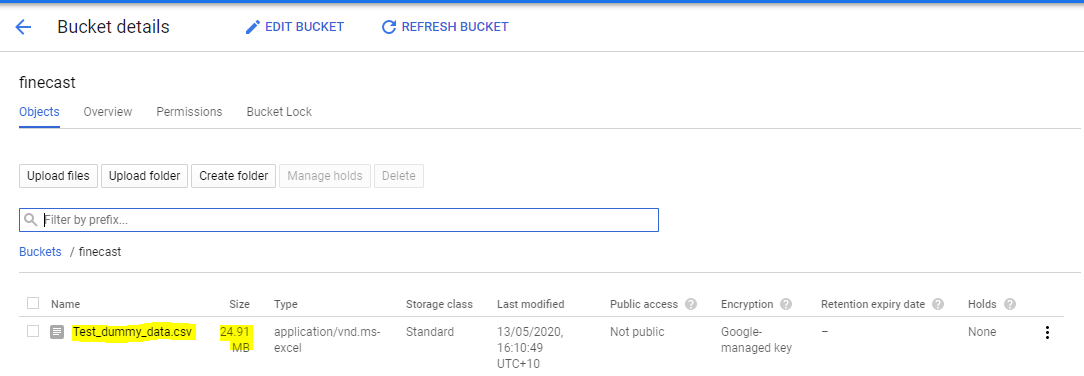
We can find BigQuery under GCP console’s ‘Big data’ heading



### Google S3 Bucket

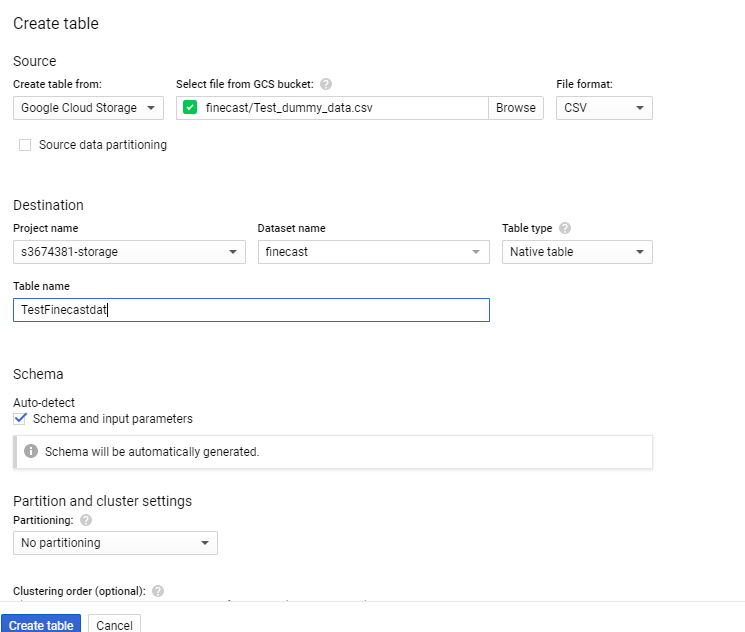
Since the size of the input data is more than 10 MB, we cannot upload data directly to the Bigquery service. In this case we can create a S3 bucket and upload the input data there.S3 bucket is GCP’s storage service. We can refer this stored data from S3 bucket when using BigQuery.

I have created a bucket for this analysis and uploaded the input data to the bucket as shown below.

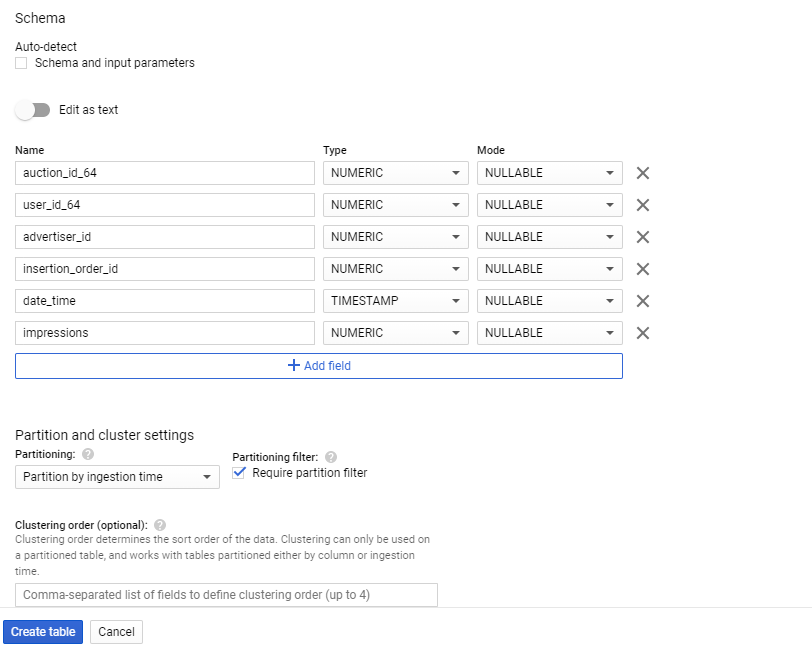


### Data Import

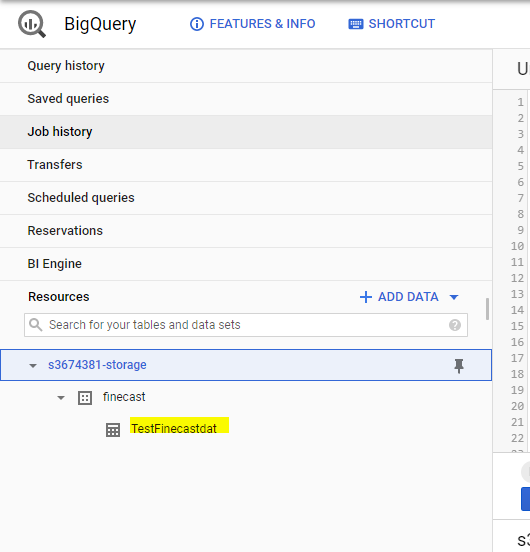
I have created a table in BigQuey for this analysis and imported data from my s3 bucket easily with the below characteristics.



We can check Source data partitioning if he dataset is too big and this will improve the performance in terms of space and time.

Here, I used the default set up to create table structure by itself. We can create the table structure as per our convenience shown below:

Once the upload complete, we can see the loaded table in the left panel of BigQuery window.



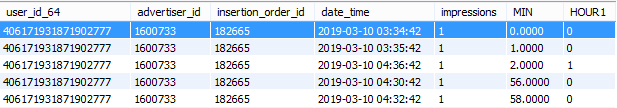
## Solution

I approached this problem in two steps:

### STEP 1

The first is to separate each record into its 1 hour(60 min) window, by getting the number of seconds difference from the minimum time for each user id having same advertisement id and insertion order. In this step we will get the hour value and minute value for each record.

To illustrate more, consider these records shown below,



Here the highlighted record is the first entry with same (user id, advertisement id, insertion order id) combination. We are taking this record as the reference for next entries within 60 minute time frame (from 3:34 to 4:33)

To understand the concept, I have added minute and Hour fields to the table. Let’s take the highlighted record, this says MIN=0 AND HOUR1=0 which means that is the first record and the reference record for 1 hour window. Similarly, for the second row MIN=2,HOUR1=0.That means the second record came ‘0 hour and 2 minute’ after the reference record and so on.

For the third row, the entry is little bit different i.e. Min=2 and hour=1.this indicates that this record has a difference of ‘1 hour and 2 min’ with the reference record. According to our expected output, this record will go under the second hour cycle (Frequency should be 1 not 3 in this case).So, we need this ‘HOUR1’ field to check the window of the record insertion.

### STEP 2

Once we get the Step 1 results. Step 2 is simple. Here we can do a DENSE\_RANK() ordering by incorporating newly created hour field along with user id, advertisement id, and insertion order id.

### SQL Query Explanation

Please find the query which I used below:



Please find the query in text format at [Appendix 2](#_Query)

As explained in the previous session, Step 2 is to fetch minute and Hour window from the timestamp fields. Here, line number 11 to 18 explains the corresponding code.

Here are some points which are specific to BigQuery editing window:

MOD()and DENSE\_RANK() functions cannot accept FLOAT values as input. So, I have change the result to NUMERIC values with the help of CAST function.

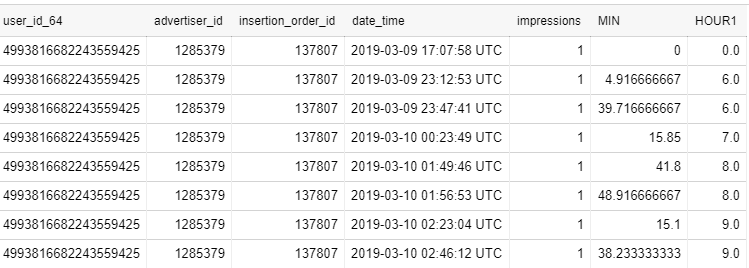
MOD function will fetch the minute remaining from the hour window by finding the first timestamp and the other insertion timestamps. Here, I used 60 as deviser to convert seconds to minutes



To fetch the hour window from the timestamp we can use’ divide’ arithmetic function and round the result with the help of FLOOR() function available in BigQuery.

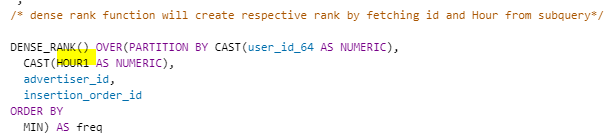


The sample output from the above is shown below:



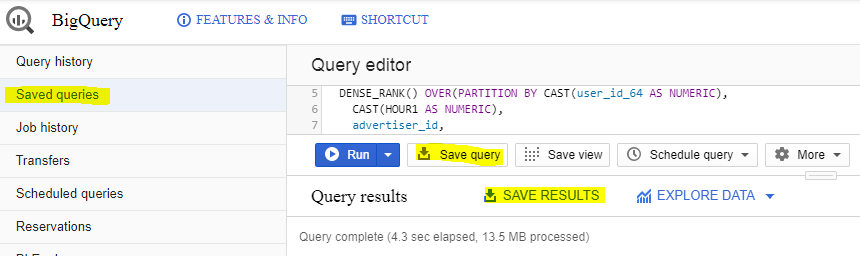
Here we are creating an ordering of the records with same user id, advertiser id and insertion order id with respect to the first timestamp.

Once this step is done, I called a DENSE\_RANK function from the outer query by incorporating the newly created HOUR 1 field and placed Step 1 as inner query



## Output and Code Save

In BigQuery, we can save the output of the code in Google cloud storage or Google Drive. Also, the code can save for later execution in S3 bucket.

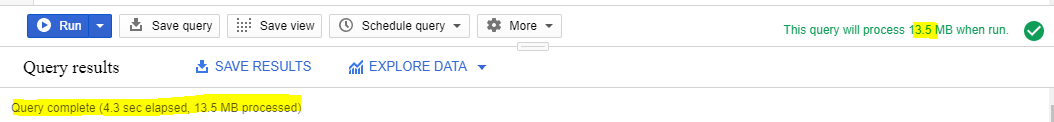


Please Find the output saved in [Appendix 3](#_Output)

I have included Minute and Hour Attributes in the output for better understanding. We can remove it by selecting only the relevant attributes from the outside query.

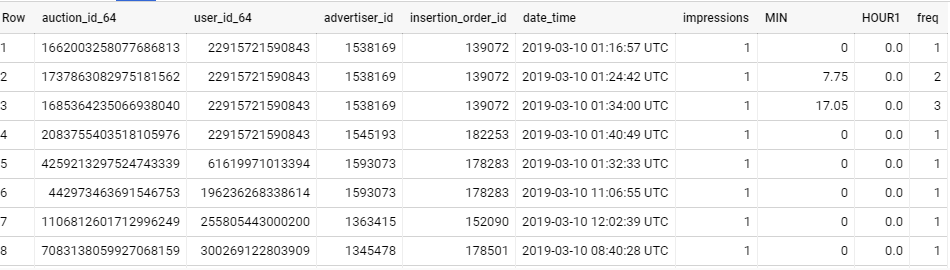
BigQuery also shows the execution time and the memory used to process the query in the output window as below:

Also, We can validate the query before the time of execution.



The input data size is around 25 MB and it took 4.3 seconds to process.

Please see the sample output below:



## Appendix

### Input Data



### Query



### Output

