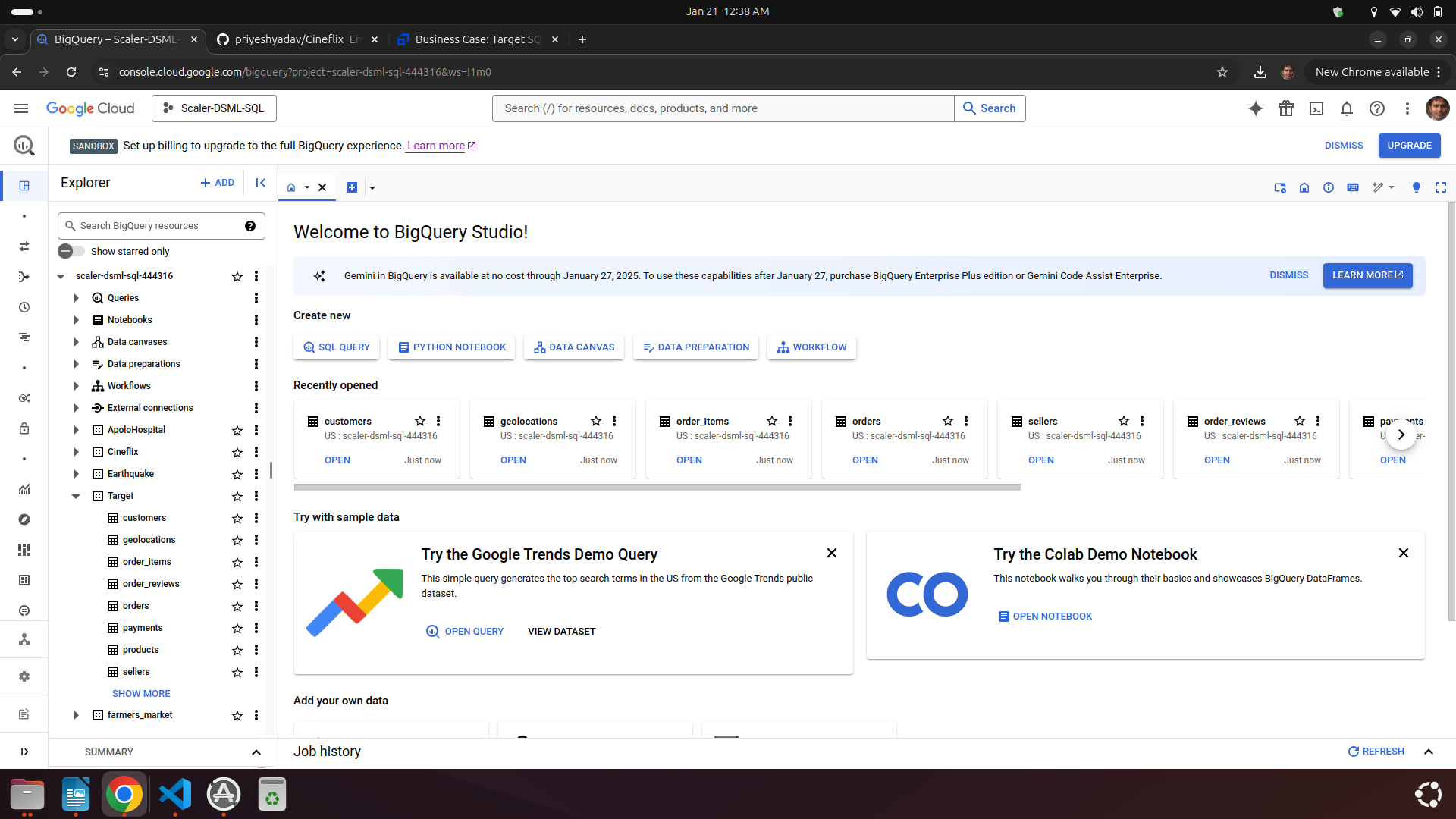
TARGET SQL – BUSINESS CASE SOLUTION

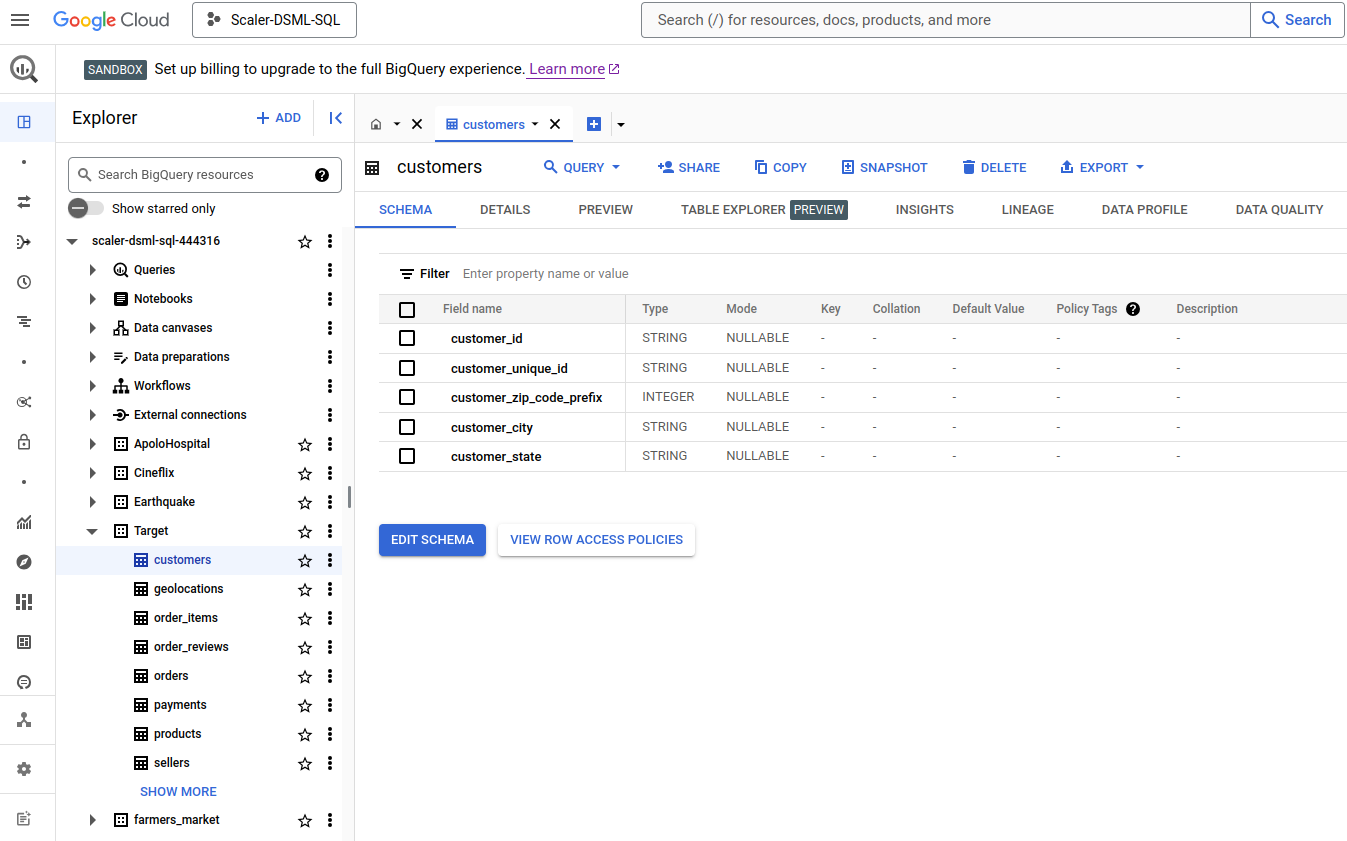
## Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

* Download all(8) CSV files from Google Drive.
* Create a *New Dataset* called ***Target*** in Google BigQuery.
* Create tables under the Target dataset by uploading each CSV one by one, using ***Upload*** & ***Auto Detect*** schema options.



### Data type of all columns in the "customers" table.

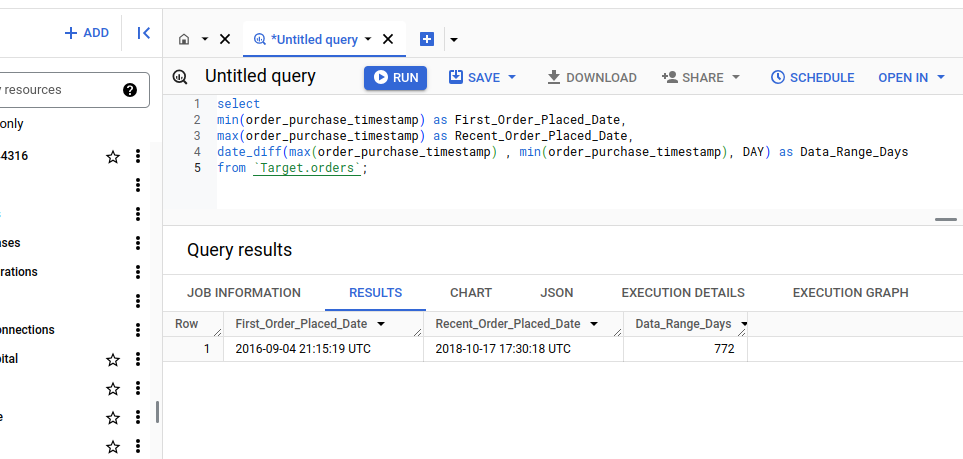
* Google BigQuery provides a convenient way to check all columns of a table & their data type.
* Click on Custome Table & Big Query will open a tab with Table Schema Details.
* Another way is to use *Information\_Schema.Columns*
* ***We observe that there are total 5 columns, 4 have string data type while 1 has timestamp***



### 

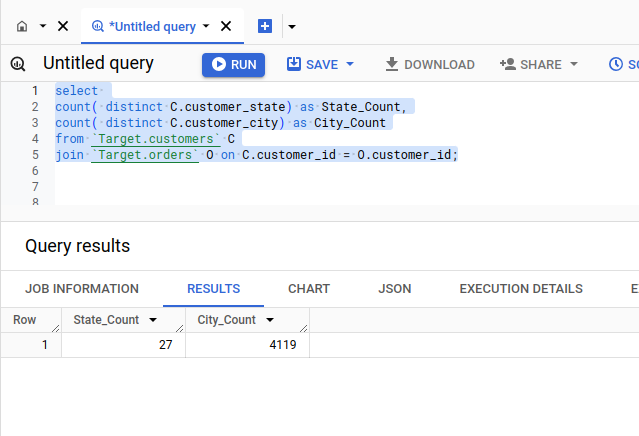
### Get the time range between which the orders were placed

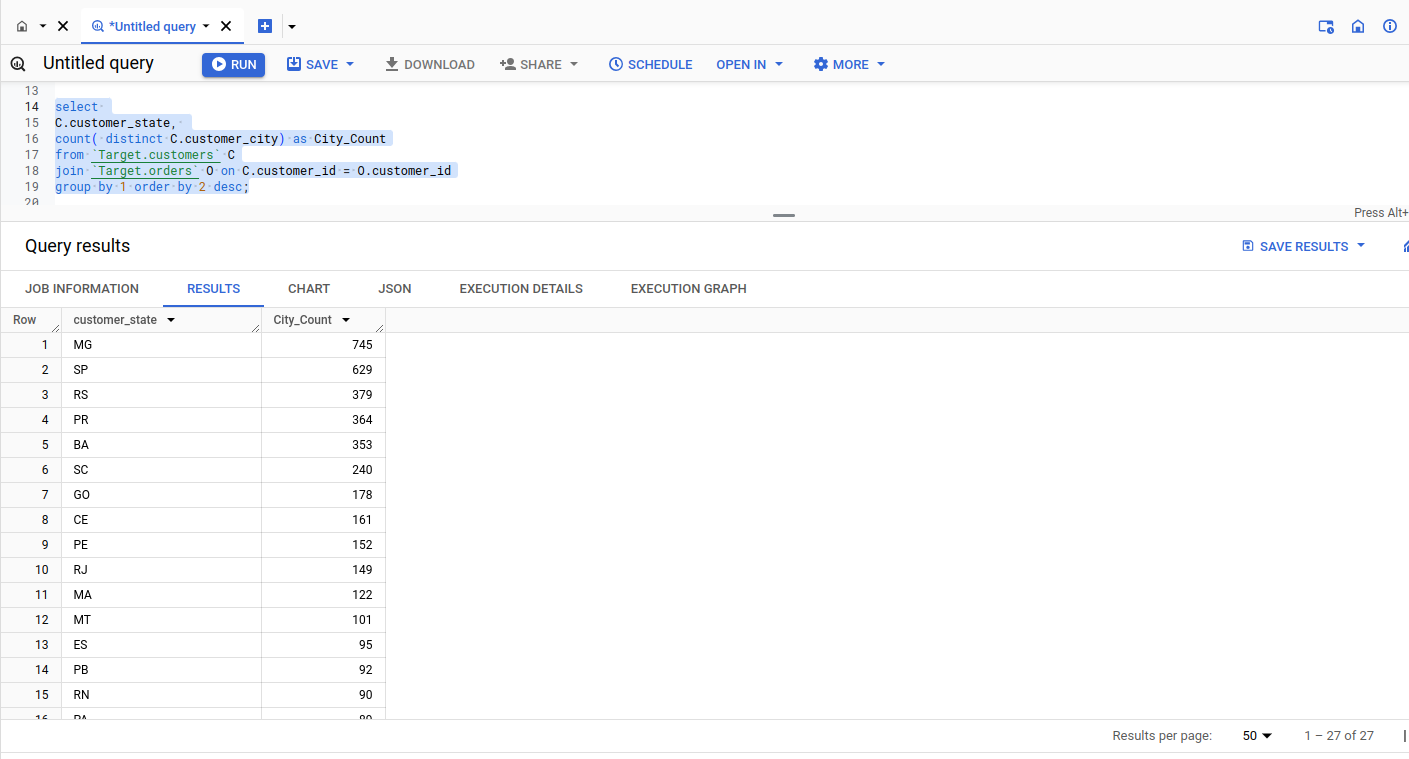
* We are looking for Oldest & Recent order datetime, Calculating difference between both of them as well.
* Since the same record is updated in the orders table, we do not filter on order\_status
* ***We Observe that We have 772 days of Range.***



### Count the Cities & States of customers who ordered during the given period

* If we inner join both *Customers* & *Orders* table, than we get only customers who has placed a order, & it will automatically fall into timeframe of Orders.
* Here also we can provide data in 2 ways
  + Individual count of distinct cities & states.
  + Count of cities per state
* ***We observe that all customer who ordered something belong to 27 different states which has 4119 distinct cities, We also identified number of cities per state.***

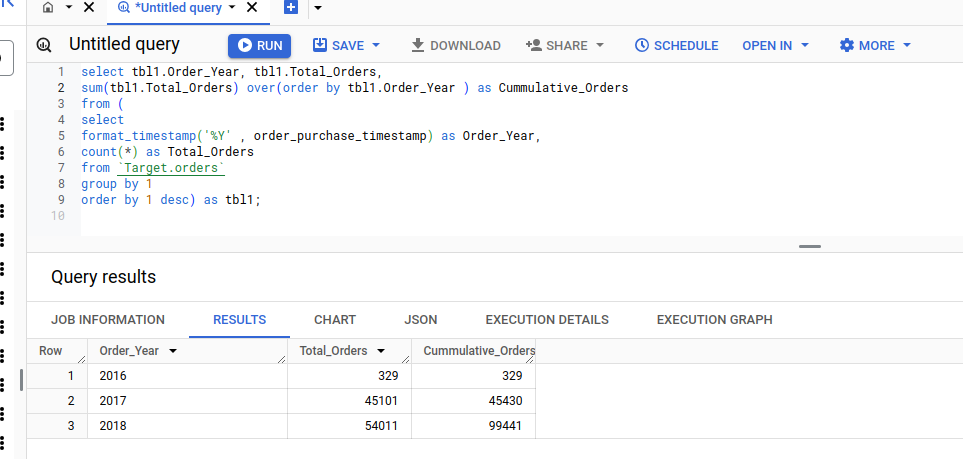




## In-depth Exploration

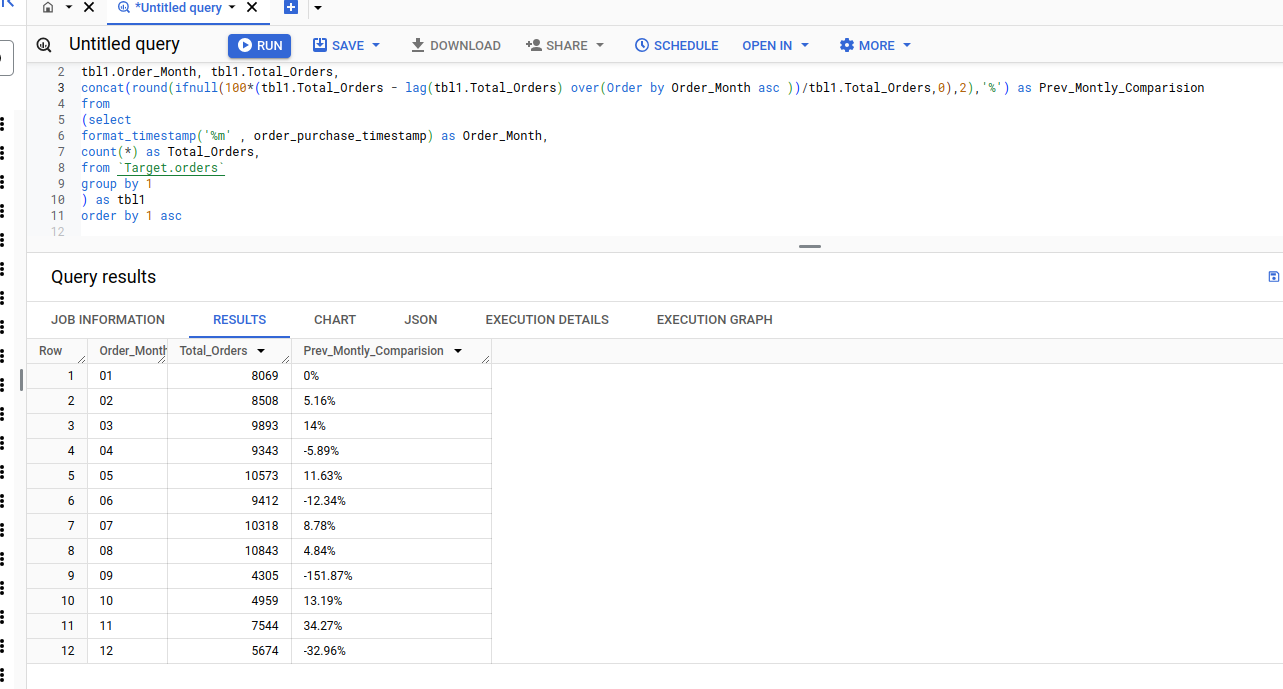
### Is there a growing trend in the no. of orders placed over the past years

* To identify yearly tread we should get the number of orders per year.
* Sort the Orders & calculate thier cumulative sum year on year
* Since we are considering trend of order placed , we should even consider canceled(625) and unavailable(609) orders
* ***We observe a Posotive Yearly Orders count pattern.***



### Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

* To identify monthly seasonality , We should group order bases on month.
* Since seasons ( e.g. winter/summer ) are based on months only , we should not consider Year-Month combination.
* Again we are considering all order irrespective of order\_status
* ***We observe that Monthly Seasonality***
  + ***Months May, June, July & August are best in terms of orders placed.***
  + ***Month Sepember, October, November & December are worst in terms of orders placed.***
  + ***Orders decline to lowest in November, While we receive most orders in August.***



### During what time of the day, do the Brazilian customers mostly place their orders

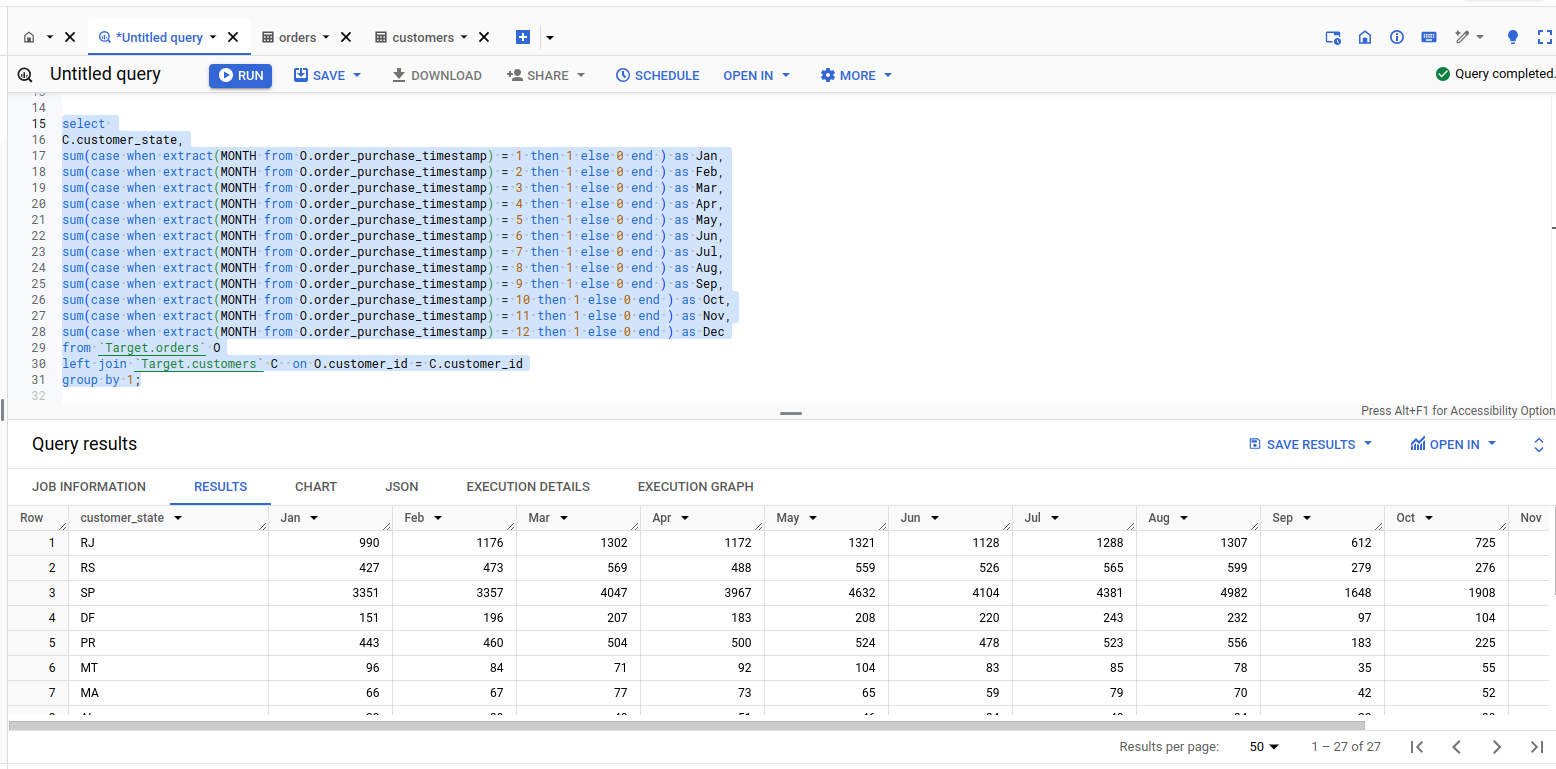
* Since the whole data is for Brazilian customer, There is no need to identify Brazilian Customer.
* First we need to extract the Hour part form the Order\_Purchase\_Timestamp.
* Next, Create categories with Case statement with given range of hours.
* Next, We count order placed in each category.
* As our final output requirement is 1 row, we limit 1 rows after sorting data based on order placed in each category in descending order.
* ***We observe that Brazilian customer mostly place thier orders in Afternoon.***

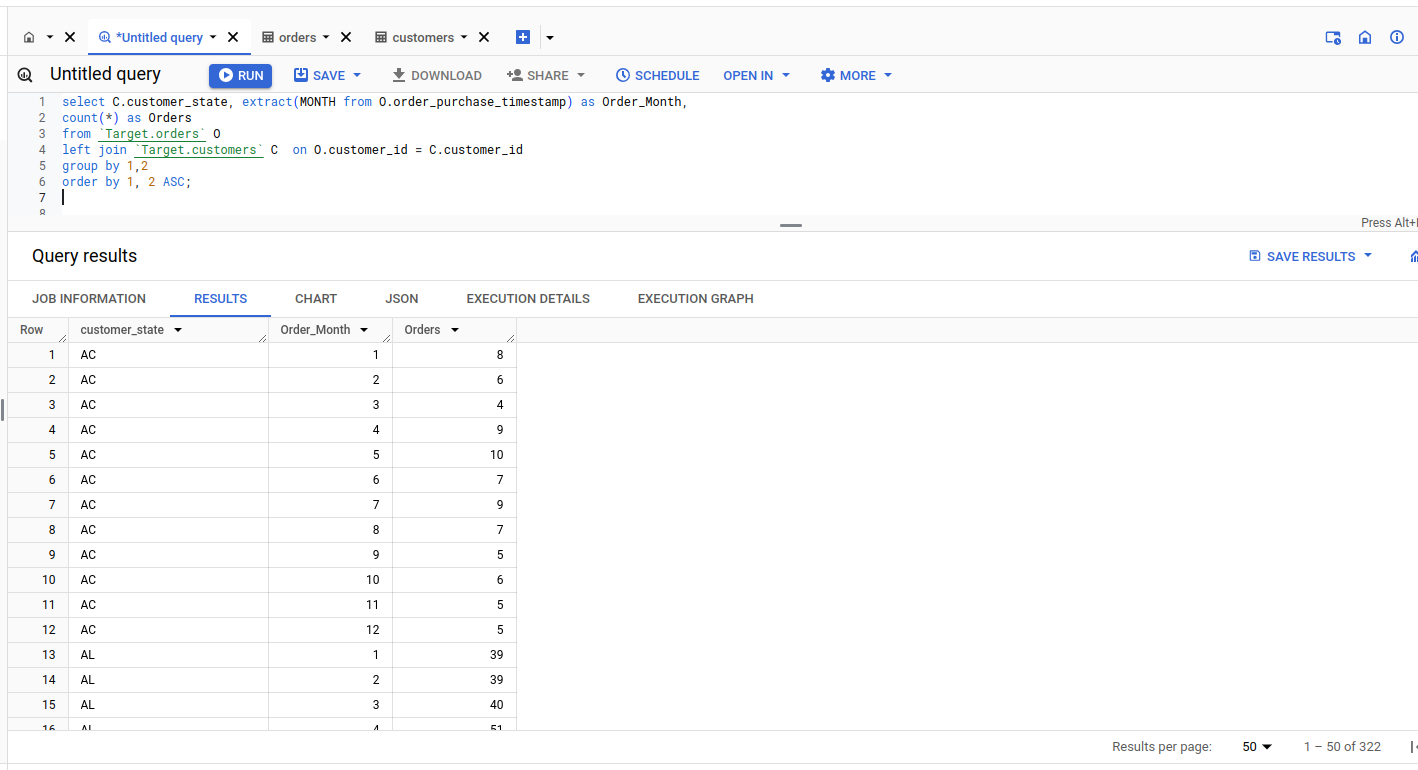


## Evolution of E-commerce orders in the Brazil region

### Get the month on month no. of orders placed in each state

* Orders table do not have *State* column.
* We should *left join* with *customers* table , so that we get State details, also retain data for orders.
* We can work with null State column separately. (*Curretly we dont have such data.)*
* We can represent this data in two format.
* ***We are able to observe the month on month orders placed in each state, however First solution provides a good pivotal view.***



**

### How are the customers distributed across all the states

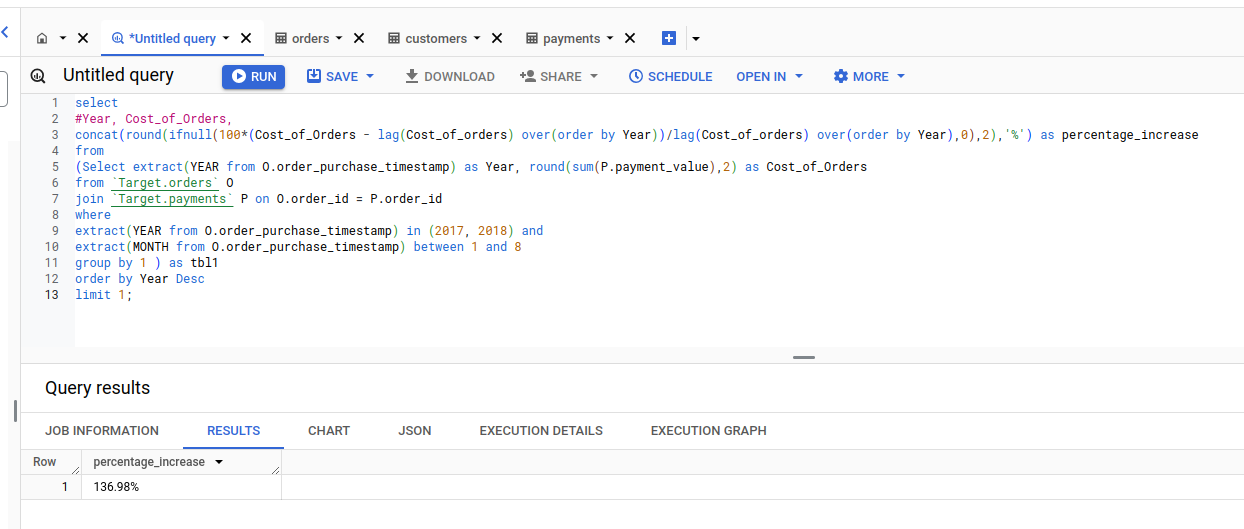
* There are two columns, *customer\_id* and *customer\_unique\_id,* But upon checking with *customers* & *orders*, we identify that each customer\_unique\_id has multiple customer\_id which is directly related to order\_id.
* Hence taking the count of customer\_unique\_id group by will provide the correct result.
* ***We Observe Customer distribution across state. SP , RJ & MG are top 3 contributing states.***

## 

## Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

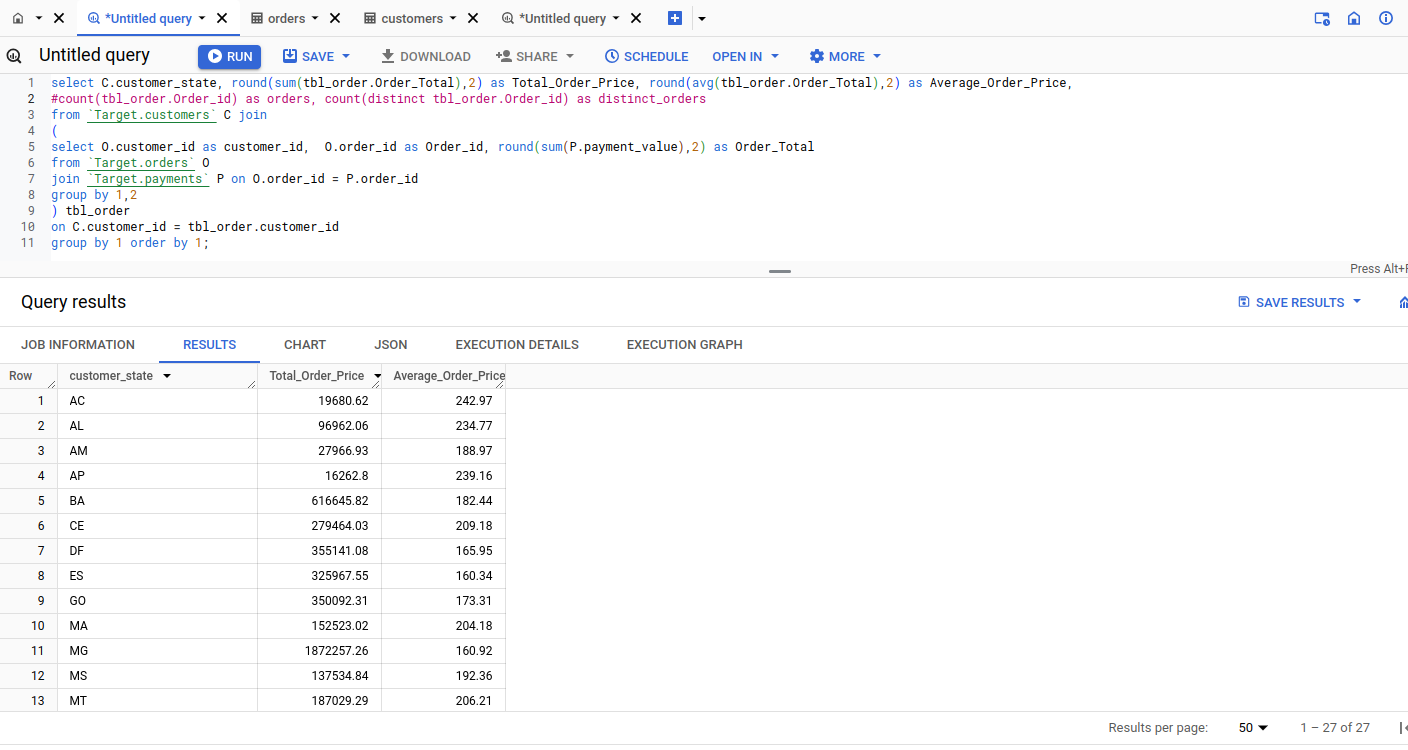
### Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only)

* We identify that Orders & Purchase table has 1 to many relationshiop.
* Purchase table do not have any date column, we can get purchase date details by combining the Orders & Purchase table over order\_id.
* We aggregate the purchase\_amount over given year & month commbination in a inner query.
* We use lag function to get next value & calculate differnce. Since we are dealing with only 2 year, One of the output for lag will be null.
* We limit our Rows to only 1 to showcase percentage increase between 2017 & 2018.
* ***We observe Percentage Increase for Cost of Orders , Between January to Auguest months of 2017 & 2018 is almost 138%.***

******

### Calculate the Total & Average value of order price for each state.

* To get order price statewirse, we need to join Payment, Orders & Customers.
* Since one order can have multiple payment, Hence calculating average using payments table directly is not a good options.
* First we create a inner query to calculate Order value per order. Combine it with Statedata & then calculate Statewise Total & Average.
* ***We observe total & average order price for each state.***



Calculate the Total & Average value of order price for each state.

### Calculate the Total & Average value of order freight for each state.

* Freight\_value is presnet in order\_items table, which has multiple rows for a single order, So we should first create a total freight\_value for each order.
* We further join it with orders & customer table to get to State for each order.
* Finally we can aggregate required values at state level.
* ***We observe total & average order freight value per state.***

