**Data cleaning (call Center Data)**

In this project, we import a dataset from "Real World Fake Data," clean it up a bit, and then analyze it using MySQL Workbench.

**Step 1: Import data in MySQL**

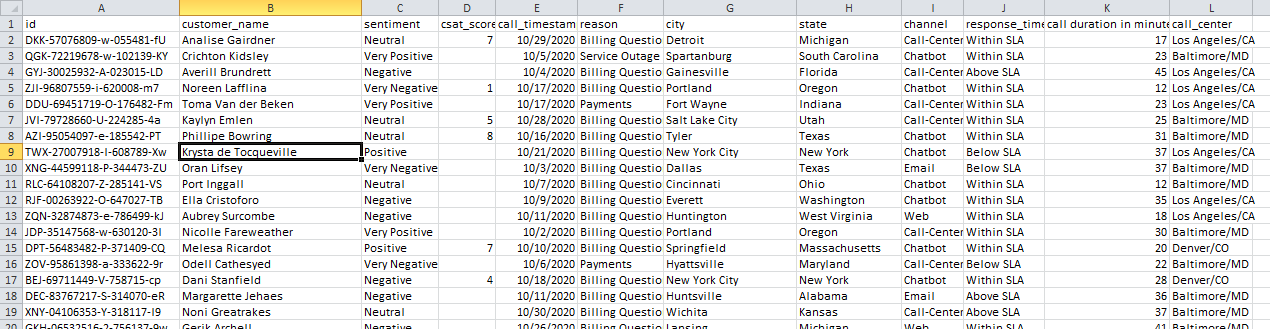
**Document type**: CSV

**Industry**: Call Center

**The link to the dataset**: <https://data.world/markbradbourne/rwfd-real-world-fake-data/workspace/file?filename=Call+Center.csv>

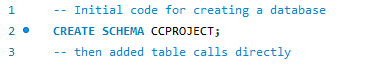
**Details**: In essence, the call-center dataset consists of about 32,941 items of information that detail calls made to multiple call centres. The caller's identity, the number of minutes the call lasted in minutes, their satisfaction level, and many more details are included.

**Snapshot from CSV file**:



You can see that there are 12 columns, with the first one, "id," containing the record's Id. Following that, we have a column for the customer's name, the sentiment for the call, and so forth.

We will create a database in MySQL and then import the CSV file into table.



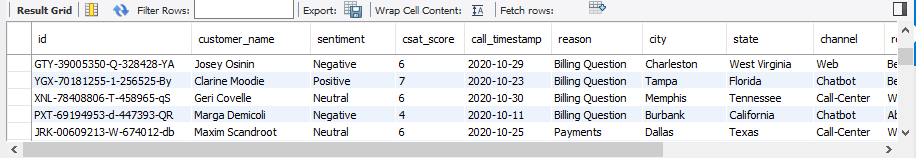
Note: The line's two dashes at the beginning indicate that it is a comment and has no bearing on our code.

After the schema CCPROJECT is created, we can check that by refreshing the Navigator SCHEMAS in MySQL. When we select the CCPROJECT scheme, we can find a table option, right click on it and select Table Data Import Wizard option. A pop up will appear and we call accept and provide the table name and proceed further to import.

I have provided the table name as ‘**calls**’.

Have a look on the newly created table.





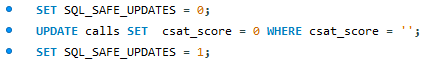
**Step 2: Data Cleaning**

When we directly imported the calls table value from CSV file, we have imported all the columns in text format except ‘call duration in minutes’ column. We can check this by,



We need to clean this data by,

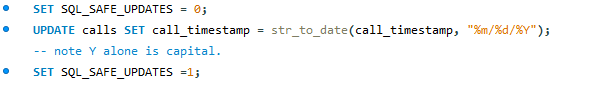
* Replace the blank cells in csat\_score column with 0.



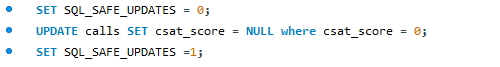
* Convert the csat\_score column to int data type.



* In the original csv file, the call timestamp was written in the following format: mm-dd-YYYY, which, when translated into human terms, means two numbers for the month, two digits for the day, and four digits for the year. Since MySQL's default format is yyyy-MM-dd, this is unacceptable. Because of this, we converted it to a string and later fix it in MySQL.



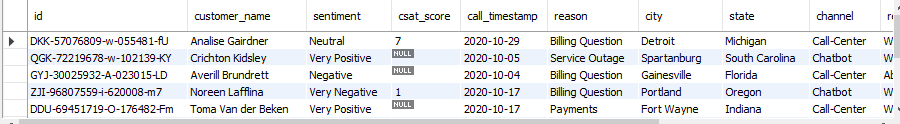
* In the csat\_score column, we must substitute null for zero as the value. If not, our aggregates will be messed up because the minimum score is 1 and not 0.



Note: The SQL\_SAFE\_UPDATES must be disabled before making the column update. The fact that we don't define a where clause that uses a KEY column is the culprit. We turn it off prior to the query and then turn it back on as a result.

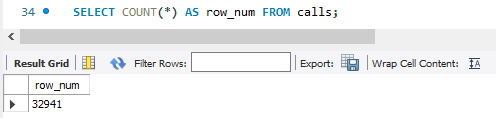
Let’s have a look on the cleaned table,

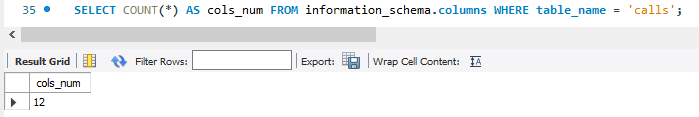




**Step 3: EDA**

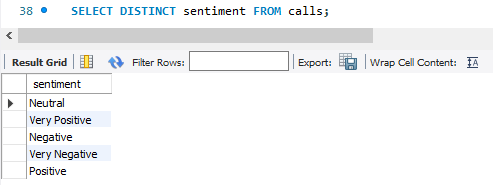
* I'm curious about the size and number of columns and rows of our table. We'll see:



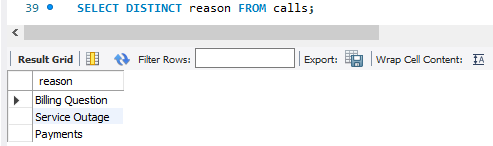


* Next, lets checkup for the unique values of required columns,

**Distinct data in ‘sentiment’ column:**

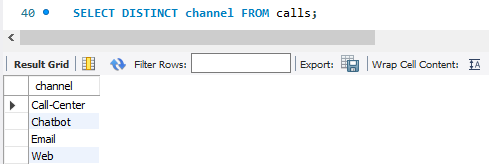


So, there are 5 types of sentiments behind each call.

**Distinct data in ‘reason’ column:**

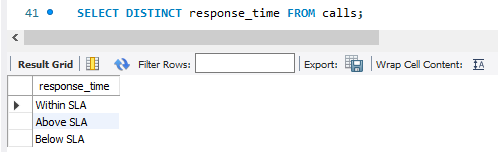
So, customer make enquiry regarding 3 resons.

**Distinct data in ‘channel’ column:**



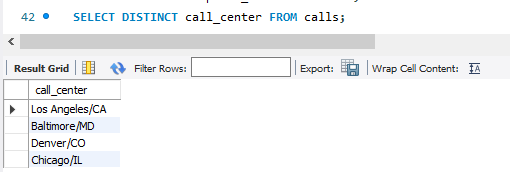
So, there are 4 mode of communication.

**Distinct data in ‘response\_time’ column:**



So, we have 3 categories of response time.

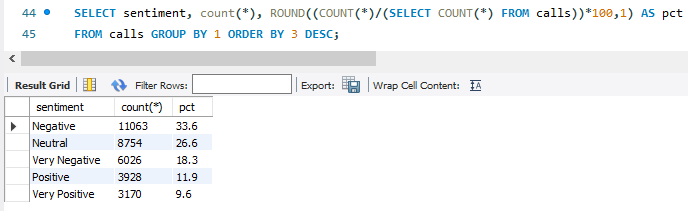
**Distinct data in ‘call\_center’ column:**



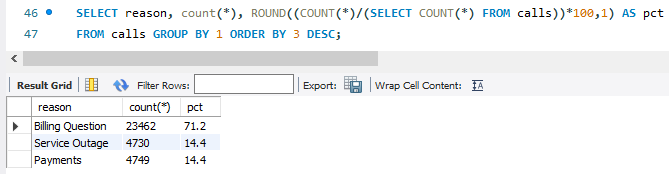
So, we have 4 call centers in total.

* Let's continue by finding the count and percentage from total of each of the distinct values we got:

**Sentiment wise,**

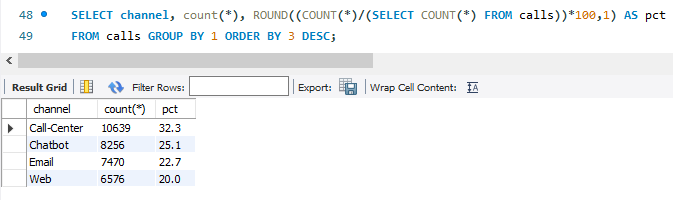


Negative sentiment based calls tops the list followed by neutral sentiment. So there is a need for improvement, mainly because positive and very positive sentiments percentage were very low.

**Reason wise,** 

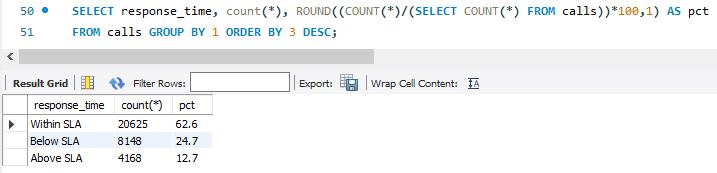
Here, we can observe that calls about billing account for a staggering 71% of all calls, while calls about service outages and payments account for 14.4% of all calls.

**Channel wise,**



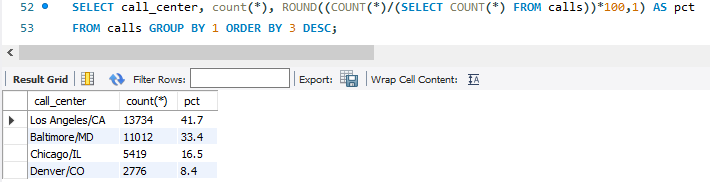
Call center takes the top position in Channel. And the Web based customer interaction is having the lowest count.

**Response time wise,**



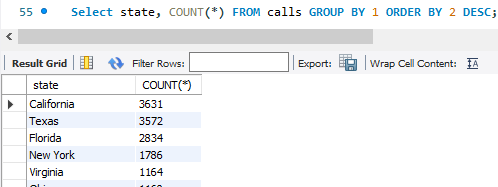
More than 80% of the tickets were handled within or below SLA. Only 12.7% calls were responded above SLA time.

**Call center wise,**



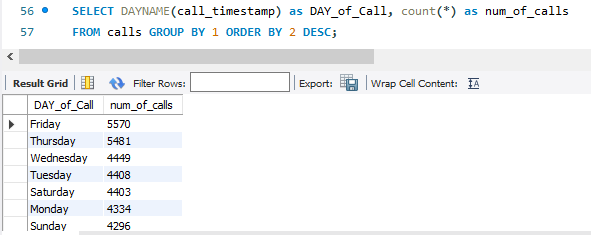
Los Angeles call center stand in the top by handling 41.7% of calls. Baltimore, Chicago, Denver handle 33.4%, 16.5%, and 8.4% of calls respectively.

* We can also take the state wise count of calls,



We can see that most calls are made by California and then Texas is in second place. And this shows the reason for Los Angeles call center handling high count of calls.

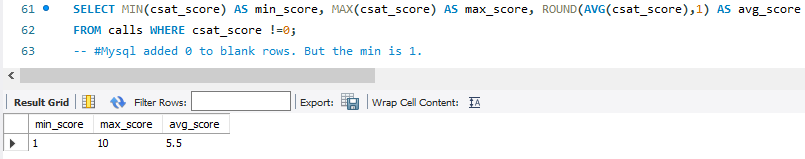
* Which day call centers received most of the calls?



So, most of the calls were received on Friday. Monday and Sunday have least calls.

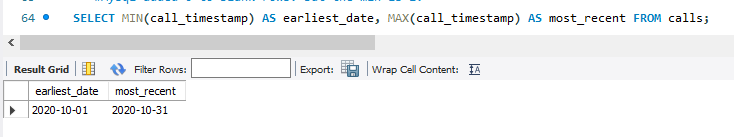
* Moving on, let’s do aggregations:

**Customer score**



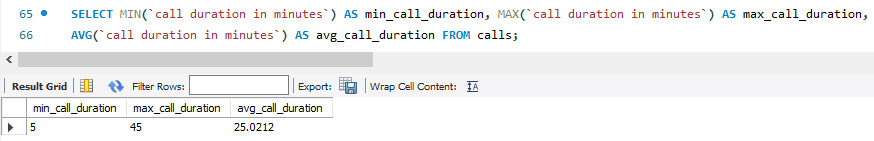
There is need for improvement in customer service as the average score is just 5.5 while the maximum score is 10.

**Call Time Stamp**



This shows us that our records belong to a time line of 1st October to 31st October of 2020.

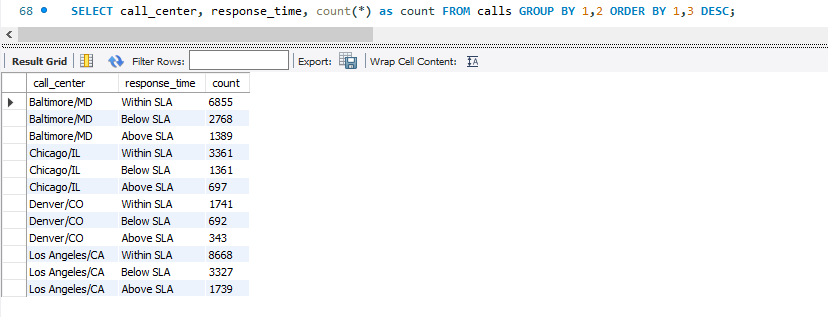
**Call duration**



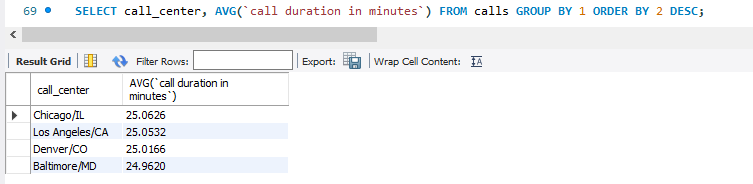
So, the minimum and maximum call duration for the month of October in 2022 is 5 and 45 minutes respectively.

Note: we use `` when there is space in column name.

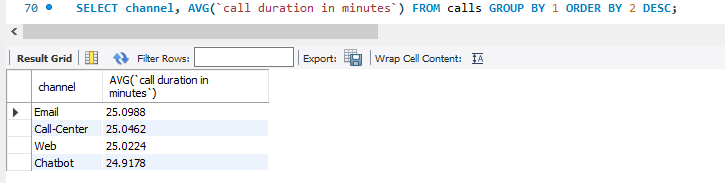
**Call center wise response time count**



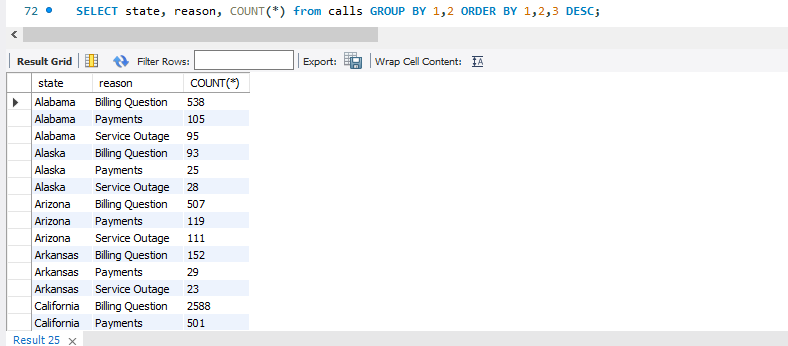
**Average call duration in minutes for each center**



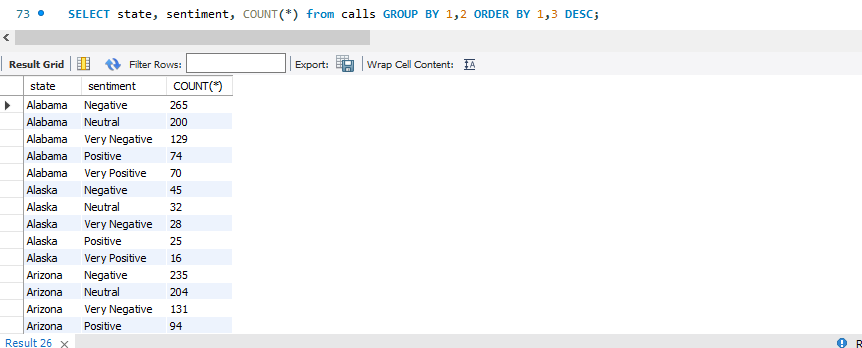
**Channel wise average call duration**



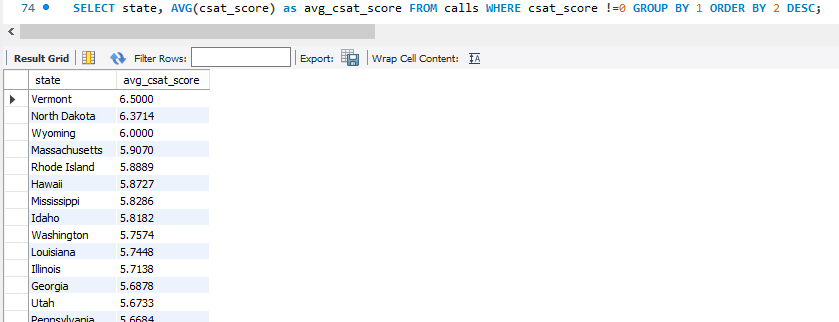
**Count of calls based on state and reason**



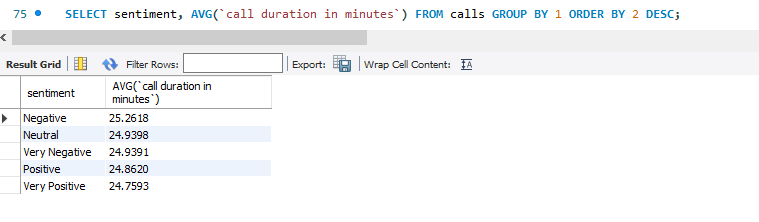
**Count of calls based on state and sentiment**



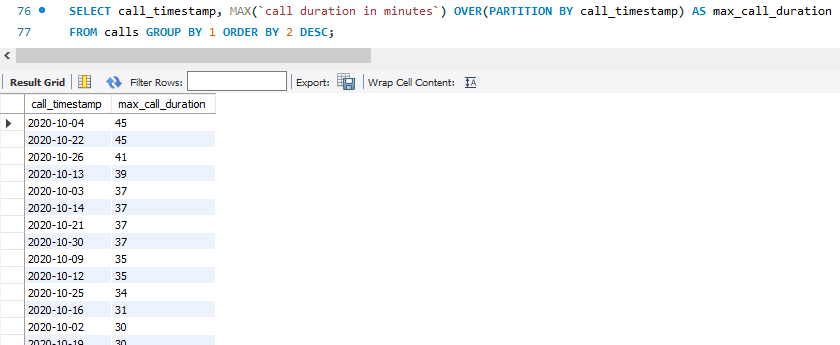
**State wise customer score**



**Sentiment wise call duration**



I just added a window function at the end to query the **daily maximum call duration** and sort by it.



We have used basic SQL statements and functions to create, clean and analyze the Call center Data. We could perform certain joins and more window operations if our database contained a number of connected tables, but I suppose we can do that later. With this cleaned data and EDA analysis output we can move forward and create attractive dashboards using Power BI.