

Data Analysis Project

*Using SQL to Clean and
Analysis Data*

- *Ganesh Dasari*

INSTRUCTIONS

In this project we use a dataset from 'Real World Fake Data', Import it to MySQL Workbench, Clean it a bit then analysis it. At last, as always, I like to use Tableau to visualize the data.

The link to the dataset :

<https://data.world/markbradbourn/rwfd-real-world-fake-data/workspace/file?filename=Call+Center.csv>

note: They have awesome datasets ! so feel free to choose whatever dataset you want from it, I picked the call-center one.

Steps we did :

- *Download the csv file.*
- *Import the file to MySQL*
- *Clean it a bit (since it's pretty clean to start with)*
- *Perform EDA*
- *Visualize it using Tableau.*



1. DOWNLOAD THE DATASET

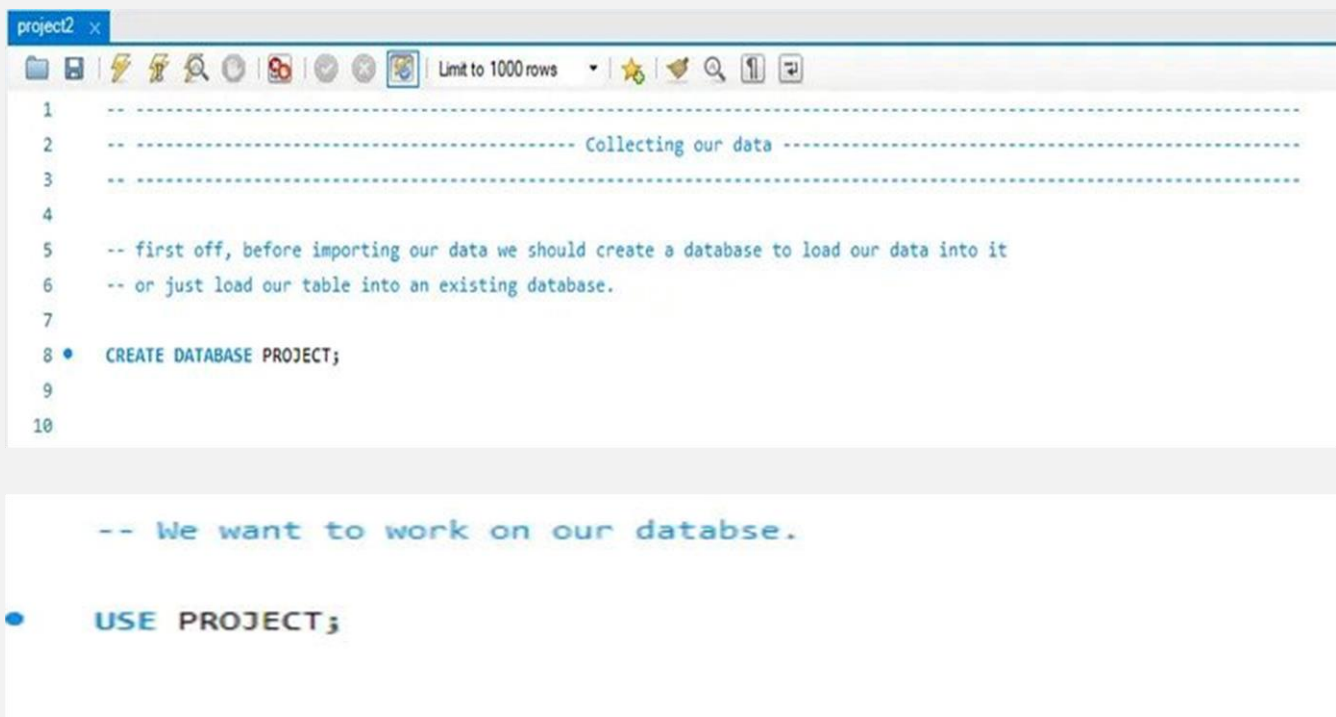
The link to the dataset : <https://data.world/markbradbourn/rwfd-real-world-fake-data/workspace/file?filename=Call+Center.csv>

2. IMPORTING THE FILE TO MYSQL

The call-center dataset basically has over 32,900 records of data that describes calls made to various call centres. It includes the ID of the call , duration of the call in minutes, the name of the person who called, their satisfaction score and many other attributes that you will see as we go.

First off, we need a database that we will import our data into it. We can either create a new one or just use an existing one. I chose the former.

Next, to get to work on a specific database we need to select it, and we do that with the keyword 'use'.



```
project2 x
Limit to 1000 rows
1  ----- Collecting our data -----
2
3
4
5  -- first off, before importing our data we should create a database to load our data into it
6  -- or just load our table into an existing database.
7
8  • CREATE DATABASE PROJECT;
9
10
-- We want to work on our databse.
• USE PROJECT;
```

After that, we need to create a table that will fit our data and match it. Let me explain that.

Here is a snapshot from the CSV file we have:

	A	B	C	D	E	F	G	H	I	J	K	L
1	id	customer_name	sentiment	csat_score	call_timestamp	reason	city	state	channel	response_time	call duration in minutes	call_center
2	DKK-57076809-w-055481-fu	Analise Gairdner	Neutral	7	10/29/2020	Billing Question	Detroit	Michigan	Call-Center	Within SLA		17 Los Angeles/CA
3	QGX-72219678-w-102139-KY	Crichton Kidsler	Very Positive		10/05/2020	Service Outage	Spartanburg	South Carolina	Chatbot	Within SLA		23 Baltimore/MD
4	GJY-30025932-A-023015-LD	Averill Brundrett	Negative		10/04/2020	Billing Question	Gainesville	Florida	Call-Center	Above SLA		45 Los Angeles/CA
5	ZJI-96807559-l-620008-m7	Noreen Lafflina	Very Negative	1	10/17/2020	Billing Question	Portland	Oregon	Chatbot	Within SLA		12 Los Angeles/CA
6	DDU-69451719-O-176482-Fm	Toma Van der Beken	Very Positive		10/17/2020	Payments	Fort Wayne	Indiana	Call-Center	Within SLA		23 Los Angeles/CA
7	JVI-79728660-U-224285-4a	Kaylyn Emlen	Neutral	5	10/28/2020	Billing Question	Salt Lake City	Utah	Call-Center	Within SLA		25 Baltimore/MD
8	AZI-95054097-e-185542-PT	Philippe Bowring	Neutral	8	10/16/2020	Billing Question	Tyler	Texas	Chatbot	Within SLA		31 Baltimore/MD
9	TWX-27007918-l-608789-Xw	Krysta de Tocqueville	Positive		10/21/2020	Billing Question	New York City	New York	Chatbot	Below SLA		37 Los Angeles/CA
10	XNG-44599118-P-344473-ZU	Oran Lifsey	Very Negative		10/03/2020	Billing Question	Dallas	Texas	Email	Below SLA		37 Baltimore/MD
11	RLC-64108207-Z-285141-VS	Port Inggall	Neutral		10/07/2020	Billing Question	Cincinnati	Ohio	Chatbot	Within SLA		12 Baltimore/MD
12	RJF-00263922-O-647027-TB	Elia Cristoforo	Negative		10/09/2020	Billing Question	Everett	Washington	Chatbot	Within SLA		35 Los Angeles/CA
13	ZQN-32874873-e-786499-kj	Aubrey Surcombe	Negative		10/11/2020	Billing Question	Huntington	West Virginia	Web	Within SLA		18 Los Angeles/CA
14	JDP-35147568-w-630120-Jl	Nicolas Fareweather	Very Positive		10/02/2020	Billing Question	Portland	Oregon	Call-Center	Within SLA		30 Baltimore/MD
15	DPT-56483482-P-371409-CQ	Melesa Ricardot	Positive	7	10/10/2020	Billing Question	Springfield	Massachusetts	Chatbot	Within SLA		20 Denver/CO
16	ZOV-95861398-a-333622-9r	Odell Cathesied	Very Negative		10/06/2020	Payments	Hyattsville	Maryland	Call-Center	Below SLA		22 Baltimore/MD
17	BEJ-69711449-V-758715-cp	Dani Stanfield	Negative	4	10/18/2020	Billing Question	New York City	New York	Chatbot	Within SLA		28 Denver/CO
18	DEC-83767217-S-314070-eR	Margarette Jehaes	Negative		10/11/2020	Billing Question	Huntsville	Alabama	Email	Above SLA		36 Baltimore/MD
19	XNY-04106353-Y-318117-19	Noni Greatrakes	Neutral		10/30/2020	Billing Question	Wichita	Kansas	Call-Center	Above SLA		37 Baltimore/MD
20	GKH-06532516-Z-756137-9w	Gerik Archell	Negative		10/26/2020	Billing Question	Lansing	Michigan	Web	Within SLA		41 Baltimore/MD
21	DJU-19977844-M-356042-cQ	Tammie Bettinson	Very Negative		10/11/2020	Payments	Lansing	Michigan	Call-Center	Within SLA		9 Chicago/IL
22	ADD-82219259-r-882390-EG	Errol Follos	Neutral		10/12/2020	Billing Question	Fort Wayne	Indiana	Chatbot	Below SLA		35 Baltimore/MD
23	YOB-40492230-M-009287-T8	Nanni Doy	Negative	5	10/08/2020	Billing Question	Hayward	California	Email	Within SLA		27 Baltimore/MD
24	GZD-50459522-O-178569-D2	Sophie Kleinerman	Very Negative	2	10/03/2020	Billing Question	Santa Barbara	California	Chatbot	Within SLA		20 Chicago/IL
25	FQX-24118867-H-358169-3N	Timotheus Menlove	Negative		10/28/2020	Billing Question	Memphis	Tennessee	Call-Center	Within SLA		43 Baltimore/MD
26	SVH-32880745-c-681066-4a	Allayne Lednor	Negative		10/06/2020	Service Outage	Murfreesboro	Tennessee	Chatbot	Below SLA		41 Baltimore/MD
27	ISK-94965442-x-233388-Vz	Bethina Fazioli	Positive		10/22/2020	Billing Question	Lubbock	Texas	Chatbot	Below SLA		45 Denver/CO
28	WUJ-90727821-m-177169-j0	Stanwood Esley	Very Negative		10/01/2020	Billing Question	New York City	New York	Call-Center	Within SLA		19 Baltimore/MD
29	PKG-51691289-6-484895-mg	Anissa Kinrade	Very Positive		10/05/2020	Payments	Oklahoma City	Oklahoma	Call-Center	Within SLA		40 Chicago/IL

You can see that we have 12 columns: 'id' containing the, well, Id of the record. Then we have a customer name column, the sentiment that describes the call and so on..

To load that data into a table in the database we need to create a table that will match it. Like this:

```
-- Create a table that will contain the csv file we want to import.
```

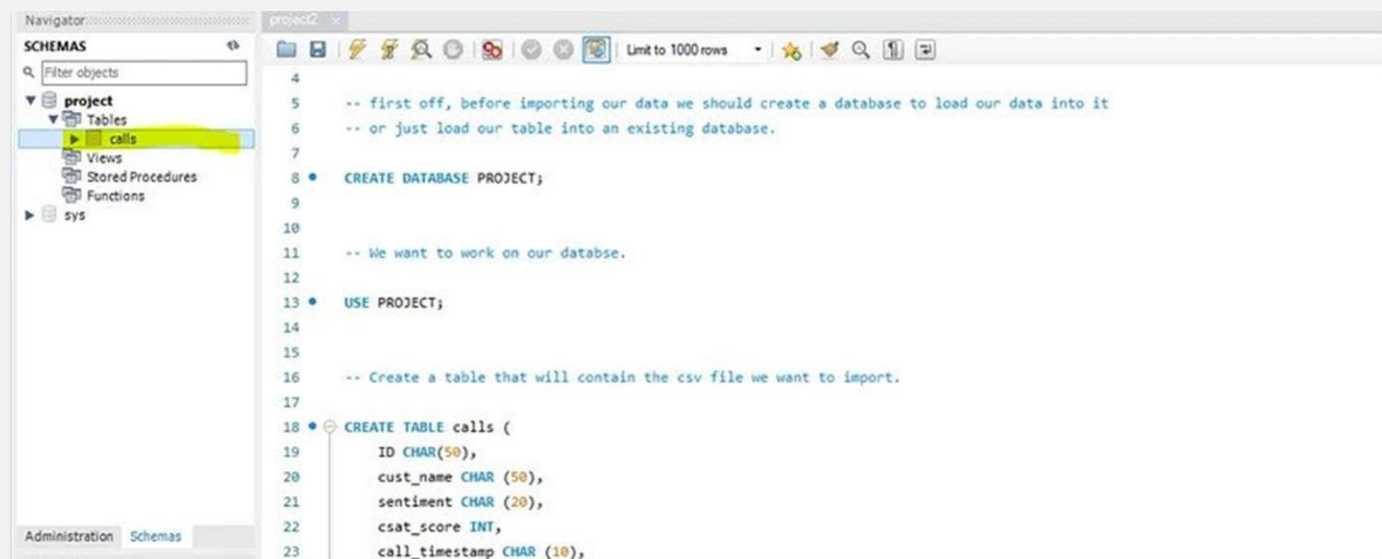
```
CREATE TABLE calls (
    ID CHAR(50),
    cust_name CHAR (50),
    sentiment CHAR (20),
    csat_score INT,
    call_timestamp CHAR (10),
    reason CHAR (20),
    city CHAR (20),
    state CHAR (20),
    channel CHAR (20),
    response_time CHAR (20),
    call_duration_minutes INT,
    call_center CHAR (20)
);
```

Here we are creating a table called 'calls', and we are designing it in a way to fit our data by matching the columns and data types. We go column by column in our csv file and in the same order we build our calls table. It is not that critical as we can change that while we import our data.

We create columns in this way (at least in this example): first off we specify the column name, then we add the data type of that column and in the parenthesis we choose the size of the variable.

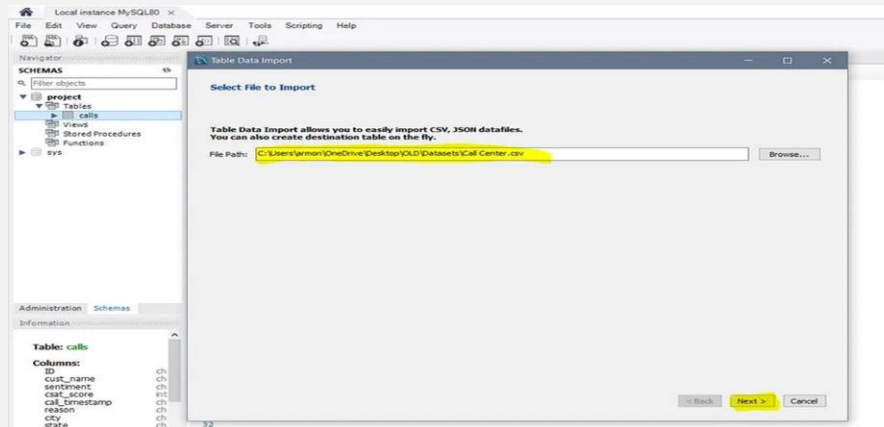
Specifying the size of the variables without checking the CSV file before proved to be stupid. By specifying that city, or state , or whatever it is , is of max size of 20 meant that rows that will be more than 20 characters will not be imported. It wasn't that critical though, I lost almost 20 rows out of 32,900+ records that was too lazy to fix, no biggie. But I hope you will double check everything you do before creating a table.

Ok, now we have our table calls that is waiting to be fed with data. We go to the left panel of the MySQL workbench, go to the database we use> the table we just created:

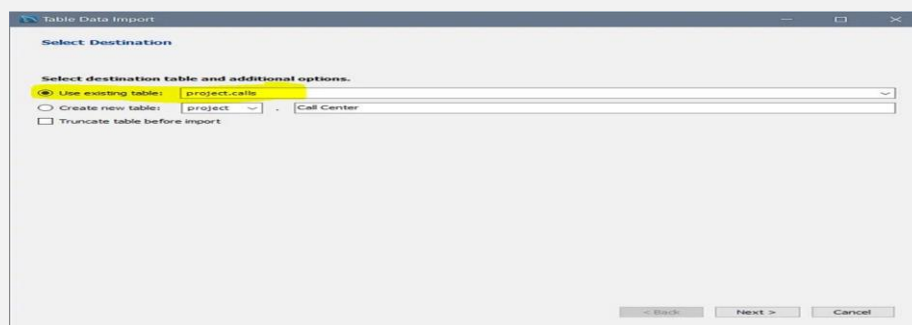


If you can't find the database or the table, make sure to refresh by pressing the little arrows button next to schemas.

When you right click, there is an option called 'Table Data Import Wizard'. That is the wizard that will help us ! click on it, it will open a window for you to browse for the csv file you want to import, search for it and then we you finish press next:

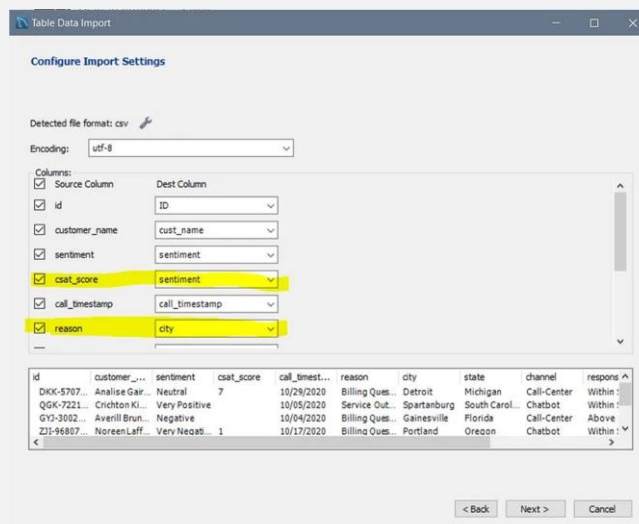


After you press next, it will ask you for the destination where you want the data to go:



We want the data to go to the table we created so choose the 'Use existing table' and then press next.

Now we need to open our eyes very well here, because here we are finalizing the process by making sure everything is good:



As you can see, it shows us the CSV columns and their destination columns in our table. But Lo and behold , in the highlighted areas it shows us that the `csat_score` will go to sentiment column in our table, and that the reason column will go to the city column *face palms*. So make sure you select the right column from the dropdown menu ! Ah right, at the end we can see a preview of our beloved csv file. Then after making sure the columns are matching press next and continue. I can't do that since I already imported my data so, continue :)

To make sure everything went smoothly, let's take a look at our table:

```
SELECT * FROM calls LIMIT 10;
```

The table is way too big to see all of it, so let's see first 10 rows. This is the result:

ID	cust_name	sentiment	csat_score	call_timestamp	reason	city	state	channel	response_time	call_duration_minutes	call_center
DKK-57076809-w-055481-fU	Analise Gairdner	Neutral	7	10/29/2020	Billing Question	Detroit	Michigan	Call-Center	Within SLA	17	Los Angeles/CA
QGK-72219678-w-102139-KY	Crichton Kiddley	Very Positive	0	10/05/2020	Service Outage	Spartanburg	South Carolina	Chatbot	Within SLA	23	Baltimore/MD
GYJ-30025932-A-023015-LD	Averill Brundrett	Negative	0	10/04/2020	Billing Question	Gainesville	Florida	Call-Center	Above SLA	45	Los Angeles/CA
ZJI-96807559-4-620008-m7	Noreen Laffina	Very Negative	1	10/17/2020	Billing Question	Portland	Oregon	Chatbot	Within SLA	12	Los Angeles/CA
DDU-69451719-O-176482-Fm	Toma Van der Beken	Very Positive	0	10/17/2020	Payments	Fort Wayne	Indiana	Call-Center	Within SLA	23	Los Angeles/CA
JVI-79728660-U-224285-4a	Kaylyn Emlen	Neutral	5	10/28/2020	Billing Question	Salt Lake City	Utah	Call-Center	Within SLA	25	Baltimore/MD
AZI-95054097-e-185542-PT	Philipe Bowring	Neutral	8	10/16/2020	Billing Question	Tyler	Texas	Chatbot	Within SLA	31	Baltimore/MD
TWX-27007918-1-608789-Xw	Krysta de Tocqueville	Positive	0	10/21/2020	Billing Question	New York City	New York	Chatbot	Below SLA	37	Los Angeles/CA
XNG-44599118-P-344473-ZU	Oran Lifsey	Very Negative	0	10/03/2020	Billing Question	Dallas	Texas	Email	Below SLA	37	Baltimore/MD
RLC-64108207-Z-285141-VS	Port Inggall	Neutral	0	10/07/2020	Billing Question	Cincinnati	Ohio	Chatbot	Within SLA	12	Baltimore/MD

3. CLEAN THE DATA.

When we created the table, you might've wondered: Why is the `call_timestamp` , a date, using a char datatype ? Back in the csv file, the call timestamp was in this format: `mm-dd-YYYY`. which in human words mean: two digits for month, two digits for day and four digits for year. This is not acceptable in MySQL which it's default format is: `yyyy-MM-dd`. That is why we made it into a string and then fix it up later in MySQL.

AND ALSO,

You might've noticed that all the empty values which were in the original/csv dataset that were in the csat_score (ah forgot to say, it means customer satisfaction score), got converted to zero's Instead of null values, which will mess up our aggregations because the minimum score is 1 and not 0.

Let's fix them:

*It's pretty straightforward to be honest, we just call the function str_to_date() and give it our column and the way the date is formatted in it (pay attention here, not the format **we** want, but the format that the string is already in it):*

```
-- The call_timestamp is a string, so we need to convert it to a date.

SET SQL_SAFE_UPDATES = 0;

UPDATE calls SET call_timestamp = str_to_date(call_timestamp, "%m/%d/%Y");

SET SQL_SAFE_UPDATES = 1;

SELECT * FROM calls LIMIT 10;
```

We need to set the SQL_SAFE_UPDATES off before we do change the column. reason is because we dont specify a where clause that uses a KEY column. That is why we set it off before the query, and then set it back on after. and the results?

ID	cust_name	sentiment	csat_score	call_timestamp	reason	city	state	channel	response_time	call_duration_minutes	call_center
DKK-57076809-w-055481-fU	Analise Gairdner	Neutral	7	2020-10-29	Billing Question	Detroit	Michigan	Call-Center	Within SLA	17	Los Angeles,CA
QGK-72219678-w-102139-KY	Crichton Kiddley	Very Positive	NULL	2020-10-05	Service Outage	Spartanburg	South Carolina	Chatbot	Within SLA	23	Baltimore,MD
GYJ-30025932-A-023015-LD	Averill Brundrett	Negative	NULL	2020-10-04	Billing Question	Gainesville	Florida	Call-Center	Above SLA	45	Los Angeles,CA
ZJI-96807559-i-620008-m7	Noreen Laffina	Very Negative	1	2020-10-17	Billing Question	Portland	Oregon	Chatbot	Within SLA	12	Los Angeles,CA
DDU-69451719-O-176482-Fm	Toma Van der Beken	Very Positive	NULL	2020-10-17	Payments	Fort Wayne	Indiana	Call-Center	Within SLA	23	Los Angeles,CA
JVI-79728660-U-224285-4a	Kaylyn Emlen	Neutral	5	2020-10-28	Billing Question	Salt Lake City	Utah	Call-Center	Within SLA	25	Baltimore,MD
AZI-95054097-e-185542-PT	Phillipe Bowring	Neutral	8	2020-10-16	Billing Question	Tyler	Texas	Chatbot	Within SLA	31	Baltimore,MD
TWX-27007918-1-608789-Xw	Krysta de Tocqueville	Positive	NULL	2020-10-21	Billing Question	New York City	New York	Chatbot	Below SLA	37	Los Angeles,CA
XNG-44599118-P-344473-ZU	Oran Lifsey	Very Negative	NULL	2020-10-03	Billing Question	Dallas	Texas	Email	Below SLA	37	Baltimore,MD
RLC-64108207-Z-285141-VS	Port Inggall	Neutral	NULL	2020-10-07	Billing Question	Cincinnati	Ohio	Chatbot	Within SLA	12	Baltimore,MD

The `call_timestamp` is finally date format .

Now to our next problem. The zero's in the `csat_score`. There are two options: either we set them to `NULL`, or we just leave them be and then when we query the table we just add the clause `WHERE csat_score != 0`. But I will set them to nulls in this way:

```
-- The call_timestamp is a string, so we need to convert it to a date.

SET SQL_SAFE_UPDATES = 0;

UPDATE calls SET call_timestamp = str_to_date(call_timestamp, "%m/%d/%Y");

UPDATE calls SET csat_score = NULL WHERE csat_score = 0;

SET SQL_SAFE_UPDATES = 1;

SELECT * FROM calls2 LIMIT 10;
```

(Make sure to add set the safe updates off and then on, just like we did earlier).

Now let's see the table:

ID	cust_name	sentiment	csat_score	call_timestamp	reason	city	state	channel	response_time	call_duration_minutes	call_center
DKK-57076809-w-055481-fU	Analise Gairdner	Neutral	7	2020-10-29	Billing Question	Detroit	Michigan	Call-Center	Within SLA	17	Los Angeles,CA
QGK-72219678-w-102139-KY	Crichton Kidsley	Very Positive	0	2020-10-05	Service Outage	Spartanburg	South Carolina	Chatbot	Within SLA	23	Baltimore,MD
GYJ-30025932-A-023015-LD	Averil Brundrett	Negative	0	2020-10-04	Billing Question	Gainesville	Florida	Call-Center	Above SLA	45	Los Angeles,CA
ZJI-96807559-i-620008-m7	Noreen Laffina	Very Negative	1	2020-10-17	Billing Question	Portland	Oregon	Chatbot	Within SLA	12	Los Angeles,CA
DDU-69451719-O-176482-Fm	Toma Van der Beken	Very Positive	0	2020-10-17	Payments	Fort Wayne	Indiana	Call-Center	Within SLA	23	Los Angeles,CA
JVI-79728660-U-224285-4a	Kaylyn Emilen	Neutral	5	2020-10-28	Billing Question	Salt Lake City	Utah	Call-Center	Within SLA	25	Baltimore,MD
AZI-95054097-e-185542-PT	Philine Bowring	Neutral	8	2020-10-16	Billing Question	Tyler	Texas	Chatbot	Within SLA	31	Baltimore,MD
TWX-27007918-I-608789-Xw	Krysta de Tocqueville	Positive	0	2020-10-21	Billing Question	New York City	New York	Chatbot	Below SLA	37	Los Angeles,CA
XNG-44599118-P-344473-ZU	Oran Lifsey	Very Negative	0	2020-10-03	Billing Question	Dallas	Texas	Email	Below SLA	37	Baltimore,MD
RLC-64108207-Z-285141-vS	Port Inggall	Neutral	0	2020-10-07	Billing Question	Cincinnati	Ohio	Chatbot	Within SLA	12	Baltimore,MD

4. EXPLORATORY DATA ANALYSIS (EDA):

Let's see, what is the shape of our table, i.e, the number of columns and rows:

```

----- Exploring our data -----
-- lets see the shape pf our data, i.e, the number of columns and rows

SELECT COUNT(*) AS rows_num FROM calls;
SELECT COUNT(*) AS cols_num FROM information_schema.columns WHERE table_name = 'calls' ;

```

Running the line gives us:

Result Grid	
	rows_num
▶	32918

Result Grid	
	cols_num
▶	12

So we 32,918 records and 12 columns.

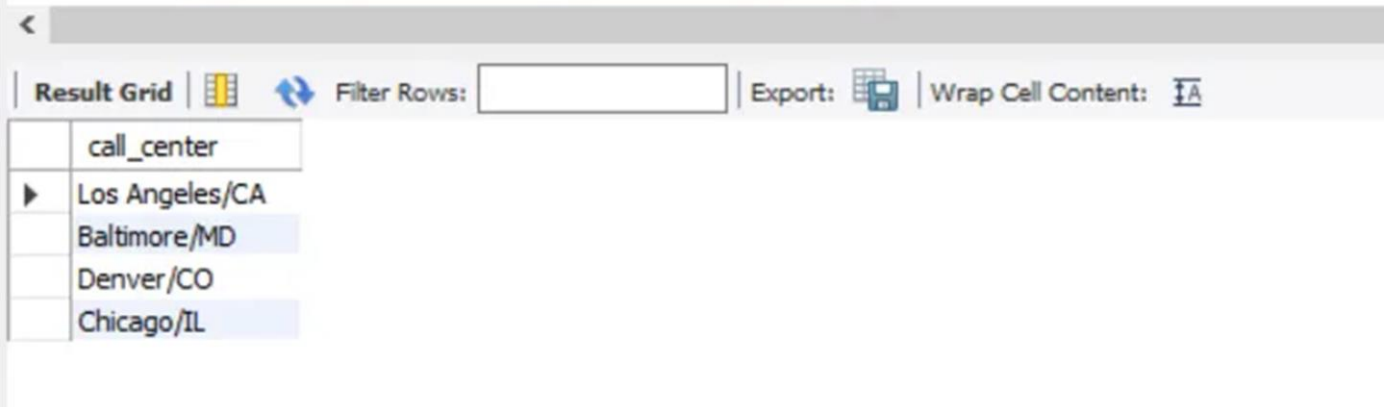
Next up: Distinct values.

```
-- Checking the distinct values of some columns:
```

- `SELECT DISTINCT sentiment FROM calls;`
- `SELECT DISTINCT reason FROM calls;`
- `SELECT DISTINCT channel FROM calls;`
- `SELECT DISTINCT response_time FROM calls;`
- `SELECT DISTINCT call_center FROM calls;`

We are checking the different values possible for the rows we selected. Let's run a random one of these , since they are all similar:

```
70 • SELECT DISTINCT sentiment FROM calls;
71 • SELECT DISTINCT reason FROM calls;
72 • SELECT DISTINCT channel FROM calls;
73 • SELECT DISTINCT response_time FROM calls;
74 • SELECT DISTINCT call_center FROM calls;
```



The screenshot shows a SQL query result grid. The top bar includes a 'Result Grid' tab, a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The main table has one column, 'call_center', and four rows of data.

call_center
Los Angeles/CA
Baltimore/MD
Denver/CO
Chicago/IL

So looks like we have only 4 call-centres. Let's continue :

```
-- The count and precentage from total of each of the distinct values we got:
```

- ```
SELECT sentiment, count(*), ROUND((COUNT(*) / (SELECT COUNT(*) FROM calls)) * 100, 1) AS pct
FROM calls GROUP BY 1 ORDER BY 3 DESC;
```
- ```
SELECT reason, count(*), ROUND((COUNT(*) / (SELECT COUNT(*) FROM calls)) * 100, 1) AS pct
FROM calls GROUP BY 1 ORDER BY 3 DESC;
```
- ```
SELECT channel, count(*), ROUND((COUNT(*) / (SELECT COUNT(*) FROM calls)) * 100, 1) AS pct
FROM calls GROUP BY 1 ORDER BY 3 DESC;
```
- ```
SELECT response_time, count(*), ROUND((COUNT(*) / (SELECT COUNT(*) FROM calls)) * 100, 1) AS pct
FROM calls GROUP BY 1 ORDER BY 3 DESC;
```
- ```
SELECT call_center, count(*), ROUND((COUNT(*) / (SELECT COUNT(*) FROM calls)) * 100, 1) AS pct
FROM calls GROUP BY 1 ORDER BY 3 DESC;
```
- ```
SELECT state, COUNT(*) FROM calls GROUP BY 1 ORDER BY 2 DESC;
```

To see the distribution of our calls among different columns. Let's see the reason column:

```
70 • SELECT DISTINCT sentiment FROM calls;
71 • SELECT DISTINCT reason FROM calls;
72 • SELECT DISTINCT channel FROM calls;
73 • SELECT DISTINCT response_time FROM calls;
74 • SELECT DISTINCT call_center FROM calls;
```

call_center
Los Angeles/CA
Baltimore/MD
Denver/CO
Chicago/IL

Here we can see that Billing Questions amount to a whopping 71% of all calls, with service outage and payment related calls both are 14.4% of all calls.

Moving on, which day has the most calls?

```
96 • SELECT DAYNAME(call_timestamp) as Day_of_call, COUNT(*) num_of_calls FROM calls GROUP BY 1 ORDER BY 2 DESC;
97
```

Day_of_call	num_of_calls
Friday	5566
Thursday	5479
Wednesday	4445
Tuesday	4407
Saturday	4397
Monday	4332
Sunday	4292

Friday has the most number of calls while Sunday has the least.

Now let's move to some aggregations:

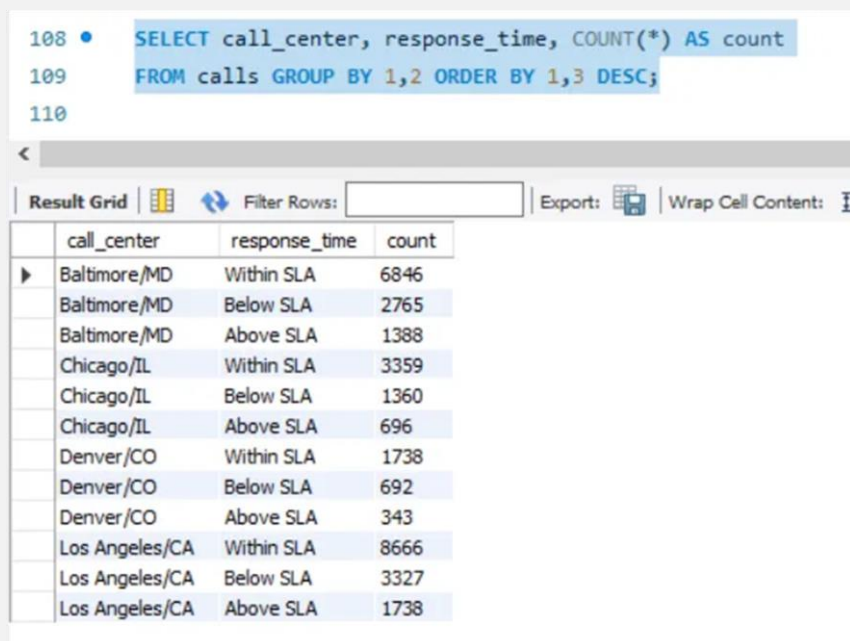
```
99 -- Aggregations :
100
101 • SELECT MIN(csat_score) AS min_score, MAX(csat_score) AS max_score, ROUND(AVG(csat_score),1) AS avg_score
102 FROM calls WHERE csat_score != 0; # MySQL added 0 to blank rows. But the min is 1.
103
104 • SELECT MIN(call_timestamp) AS earliest_date, MAX(call_timestamp) AS most_recent FROM calls;
105
106 • SELECT MIN(call_duration_minutes) AS min_call_duration, MAX(call_duration_minutes) AS max_call_duration, AVG(call_duration_minutes) AS avg_call_duration FROM calls;
107
108 • SELECT call_center, response_time, COUNT(*) AS count
109 FROM calls GROUP BY 1,2 ORDER BY 1,3 DESC;
110
111 • SELECT call_center, AVG(call_duration_minutes) FROM calls GROUP BY 1 ORDER BY 2 DESC;
112
113 • SELECT channel, AVG(call_duration_minutes) FROM calls GROUP BY 1 ORDER BY 2 DESC;
114
115 • SELECT state, COUNT(*) FROM calls GROUP BY 1 ORDER BY 2 DESC;
116
117 • SELECT state, reason, COUNT(*) FROM calls GROUP BY 1,2 ORDER BY 1,2,3 DESC;
118
119 • SELECT state, sentiment, COUNT(*) FROM calls GROUP BY 1,2 ORDER BY 1,3 DESC;
120
121 • SELECT state, AVG(csat_score) AS avg_csat_score FROM calls WHERE csat_score != 0 GROUP BY 1 ORDER BY 2 DESC;
122
123 • SELECT sentiment, AVG(call_duration_minutes) FROM calls GROUP BY 1 ORDER BY 2 DESC;
124
```

Again, most of them follow the same logic with minor changes. Let's run a few of them , starting with the one at line 106, querying the min, max and average call duration in minutes:



	min_call_duration	max_call_duration	avg_call_duration
▶	5	45	25.0175

Then let's check line 108 and 109:

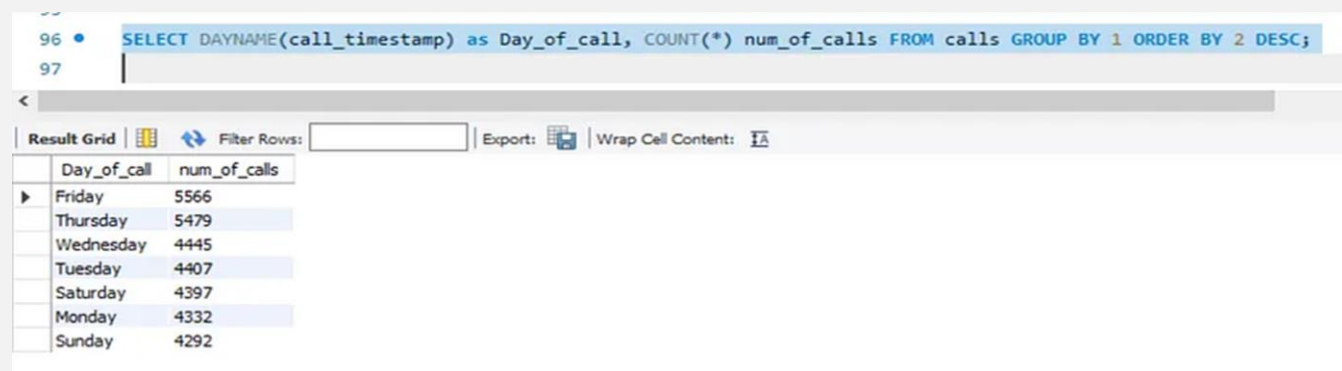


```
108 • SELECT call_center, response_time, COUNT(*) AS count
109 FROM calls GROUP BY 1,2 ORDER BY 1,3 DESC;
110
```

	call_center	response_time	count
▶	Baltimore/MD	Within SLA	6846
	Baltimore/MD	Below SLA	2765
	Baltimore/MD	Above SLA	1388
	Chicago/IL	Within SLA	3359
	Chicago/IL	Below SLA	1360
	Chicago/IL	Above SLA	696
	Denver/CO	Within SLA	1738
	Denver/CO	Below SLA	692
	Denver/CO	Above SLA	343
	Los Angeles/CA	Within SLA	8666
	Los Angeles/CA	Below SLA	3327
	Los Angeles/CA	Above SLA	1738

Here we are checking how many calls are within, below or above the Service-Level - Agreement time. For example we see that Chicago/IL call center has around 3359 calls Within SLA , and then Denver/CO has 692 calls below SLA. you get it.

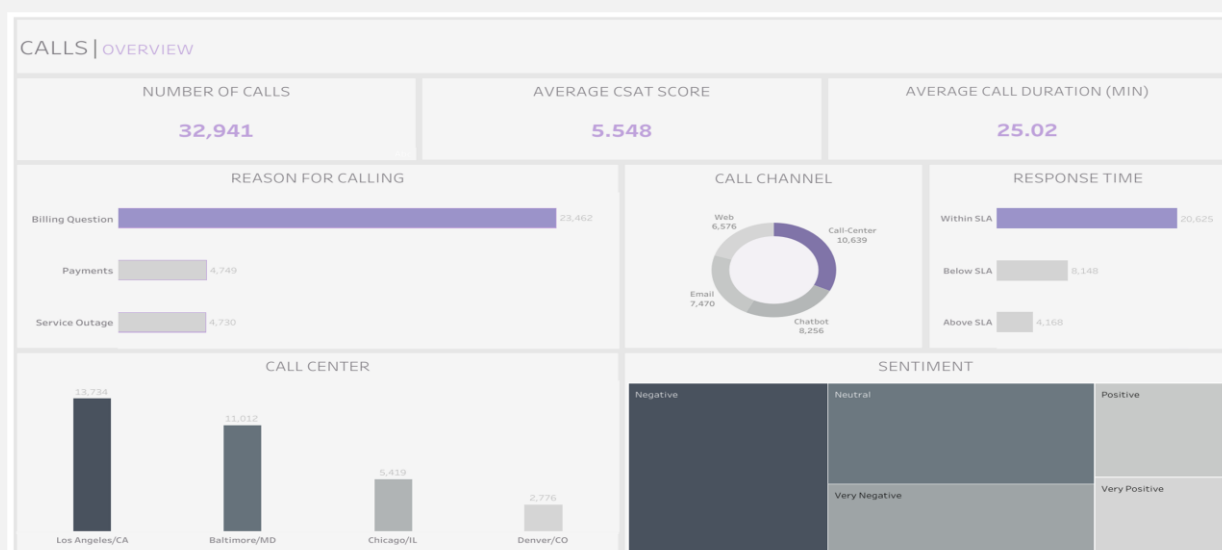
At the end I just added one Window Function to query the maximum call duration each day and then sort by it:



Here we see that for example on Oct 4th the maximum call duration was 45 minutes long while on Oct 8th it was 27 minutes long.

That's it for SQL ! You can make as many queries as you want and change to suit your dataset. One thing to be said though, is I wished we had a database with multiple related tables so we can run some joins and more window functions, but I guess we can do that next time!

BEFORE WRAPPING UP, I MADE A DASHBOARD USING TABLEAU !



Thank You.