**Project Description:**

Working as a Data Analyst in Microsoft, I have to extract useful insights from the dataset provided. These insights will depict how the end users are engaging with the product and fluctuations in the product’s performance. The approach and the corresponding insights and results will be listed down in detail in the coming pages of this report.

**Tech-Stack Used:**

To implement this project, I have used the MySQL Workbench community version 8.0. There are two reasons for this:

* MySQL’s community version allows news users to utilize their platform for analytics free of cost.
* I am familiar with the UI of this tool as I had practiced with the same tool in my college years.

**CASE STUDY 1:**

**Preparing the data:**

Create a database and table for the job data analysis and use SQL commands to insert into the table, the data given in .csv files.



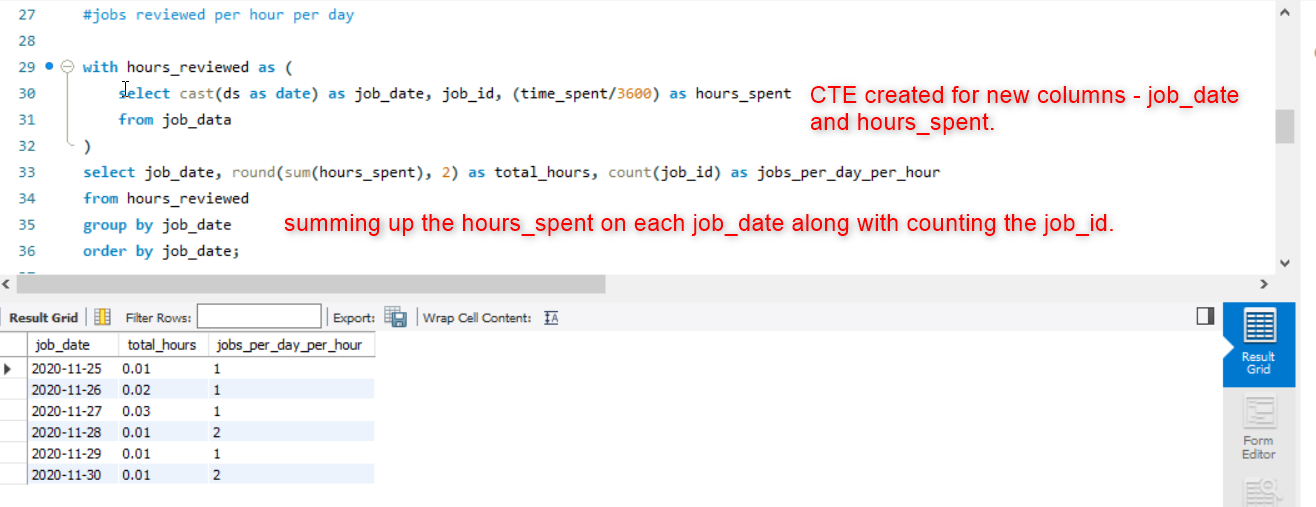
Problem Description No. 1:

Jobs Reviewed Over Time:

Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

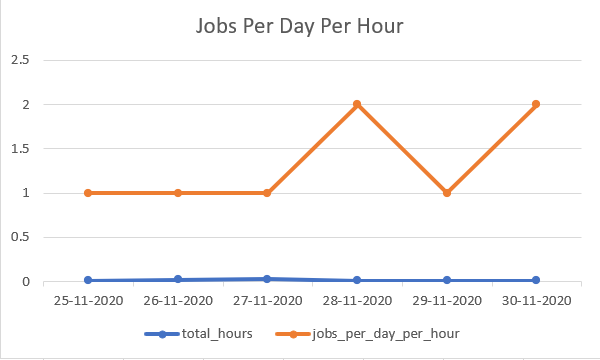
**Approach:**

1. Break down the problem description into the keywords – hour and day.
2. Convert the time spent attribute given to us in seconds into hours.
3. Convert the date given in attribute ‘ds’ to date format using **cast** function and store as **job\_date**.
4. Create a temporary table – hours\_reviewed with these two new attributes – hours\_spent and job\_date.
5. Now we count the job\_id attribute for each day of November as shown below:



**Insights:**

If we visualize the result, we can see that on 28th and 30th in 1 hour more than 1 job has been reviewed as compared to the other days. Further we can see that on all other days less than 1 hour has been spent on each job.

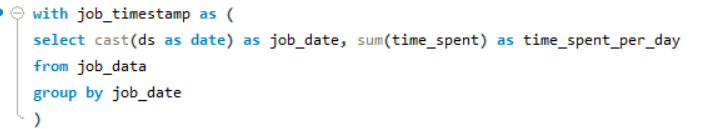
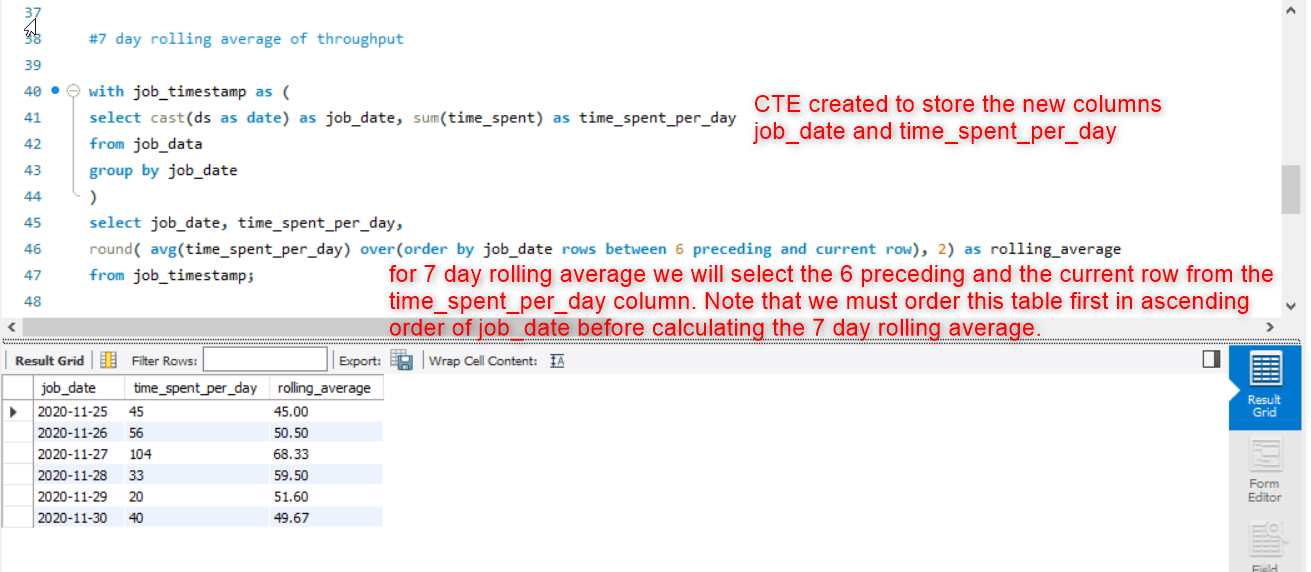


Problem Description No. 2:

Throughput Analysis:

Objective: Calculate the 7-day rolling average of throughput (number of events per second).

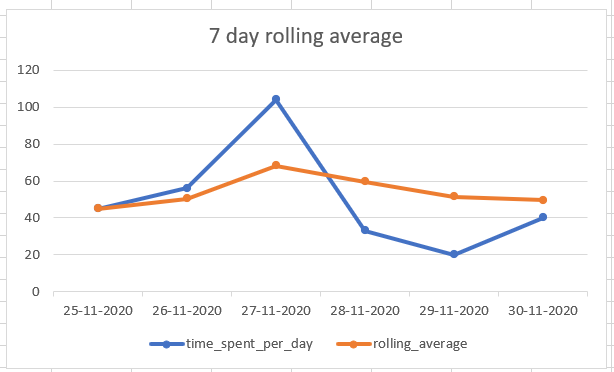
**Approach:**

1. First, calculate the time spent on each date as time\_spent\_per\_day and **cast** the ds attribute to job\_date attribute.
2. Store these new columns in a temporary table job\_timestamp.  
   
3. To calculate rolling average, we will use the **avg() window function** on this new column time\_spent\_per\_day and take the 6 previous and 1 current row as we move down the column.  
   

**Insights:**

If we visualize the result obtained, we can easily see that by calculating the rolling average of throughput, we are able to smooth out the blue spike into a much less steep line and provide a trend over a period of 7 days.  
On the 27th, the time spent is 104 seconds which is almost more than twice of what is spent on the other days while on the 29th it is only 20 seconds.

Therefore, if we want to make a prediction for the month of December, we can say that on an average, the time taken to review each job would require somewhere between 45 and 70 seconds.

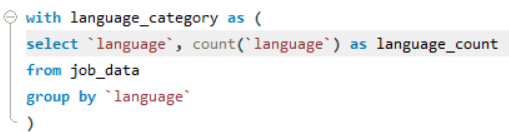
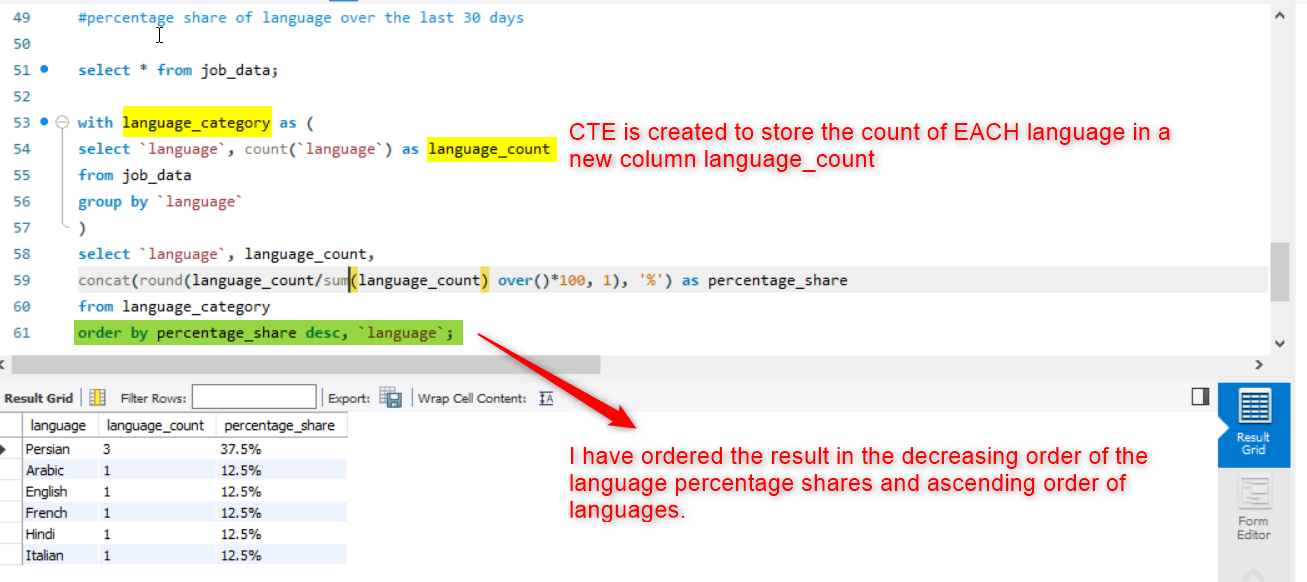


Problem Description No. 3:

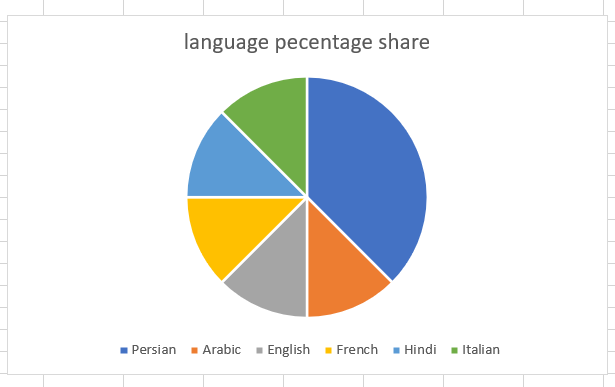
Language Share Analysis:

Objective: Calculate the percentage share of each language in the last 30 days.

**Approach:**

1. Create count of each language from the job\_data table and store it in a temporary table.  
   
2. To find percentage share of each language, we will divide the count of each language with the total share of all the languages. In the query below, I have rounded the percentage value to 1 decimal place with the help of **round()** function and afterwards added a percentage sign to this value with the help of **concat()** function.  
   

**Insights:**

By visualizing the result in a pie chart, we can see that Persian language takes the greatest area which means that most job openings prefer Persian language. With this insight, we can inform our job candidates that Persian language is highly in demand and can give them an upper hand in their job application journey.  


Problem Description No. 4:

Duplicate Rows Detection:  
Objective: Identify duplicate rows in the data.

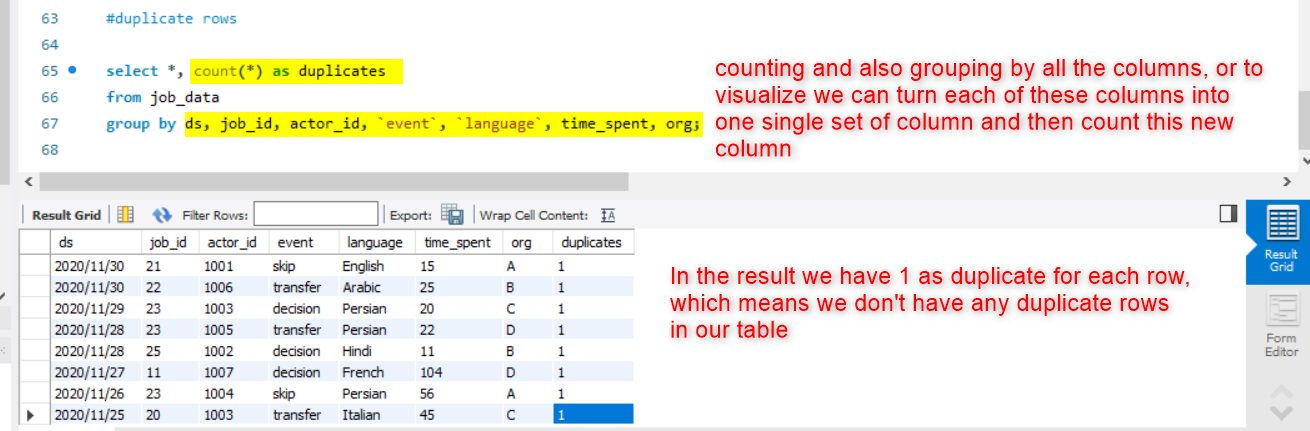
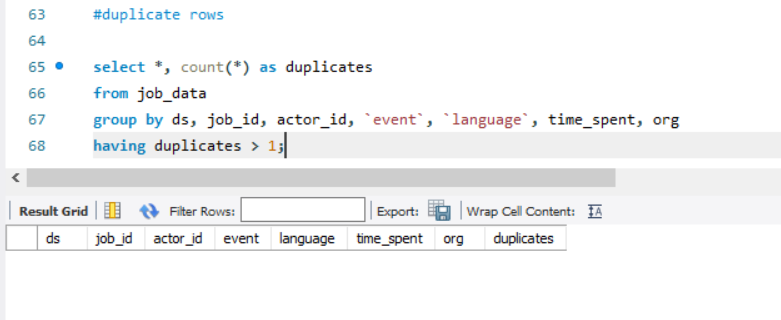
**Approach:**

1. To identify duplicate rows from the job\_data table, we know that for any 2 rows, **all** columns should have the same value. For example, in the table below, the first 2 rows are duplicates because **all** columns have the same value:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Letter** | **Name** | **Category** |
| 1 | A | Apple | fruit |
| 1 | A | Apple | Fruit |
| 2 | B | Banana | fruit |

1. Now, to count the 2 rows as 1 we will group **all** columns into one single column. In other words, we will create a new column to group the 2 rows together. This new column will have all the attributes. We can call this column {ID\_Letter\_Name\_Cateogry}.
2. This means our table will look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Letter** | **Name** | **Category** | **ID\_Letter\_Name\_Category** |
| 1 | A | Apple | fruit | 1\_A\_Apple\_fruit |
| 1 | A | Apple | Fruit | 1\_A\_Apple\_fruit |
| 2 | B | Banana | fruit | 2\_B\_Banana\_fruit |

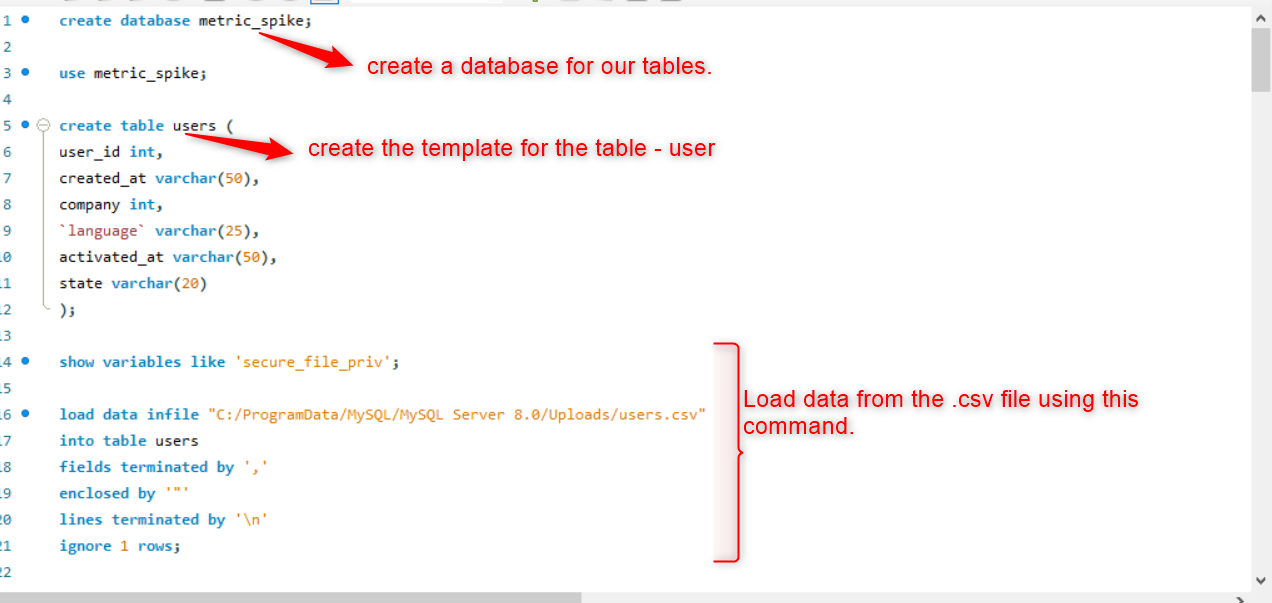
1. Now, to find the duplicate row we can simply count the ID\_Letter\_Name\_Cateogry and check for duplicates.
2. With the above example, we can write our SQL query for the **job\_data** table:  
   
3. Now to filter only the rows which are duplicates, we will use the **having** clause with the duplicates column where the value is greater than 1. We get no result for this query because we do not have any duplicate rows as can be seen in step 5.  
   

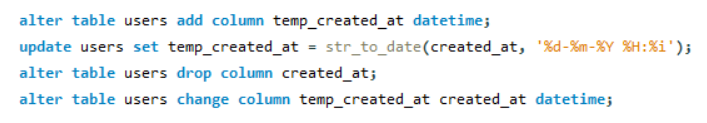
**Insights:**From the result we can see that there are no duplicates in our table. Finding the duplicates will help us clean our data and make it more valuable for providing the correct insights such as average time spent on each job, language share, etc.  
It is important to note here that we consider **all** provided attributes to find out the duplicates as **same** actor can apply for multiple jobs on the **same** date. Similarly, 2 actors can apply for the **same** job or the **same** language or on **same** dates and can event spend the **same amount of time** to review the job.

**CASE STUDY 2:**

**Preparing the data:**

To load the tables provided in .csv in the MySQL Workbench, we first create the **template** of these tables using SQL queries where we define the attributes’ datatypes and size. After this we will write the command below to **import** the data from the .csv files into the tables we have created:



After this we will convert the created\_at attribute’s datatype from varchar to timestamp using the following set of commands:  
Here we have altered the table to create a new column with datatype datetime, and added the values to it from the created\_at column after converting it to timestamp using the str\_to\_date() function. Later we drop this created\_at column since it’s no longer needed and lastly, we change the new column we created to take the name “created\_at”.  


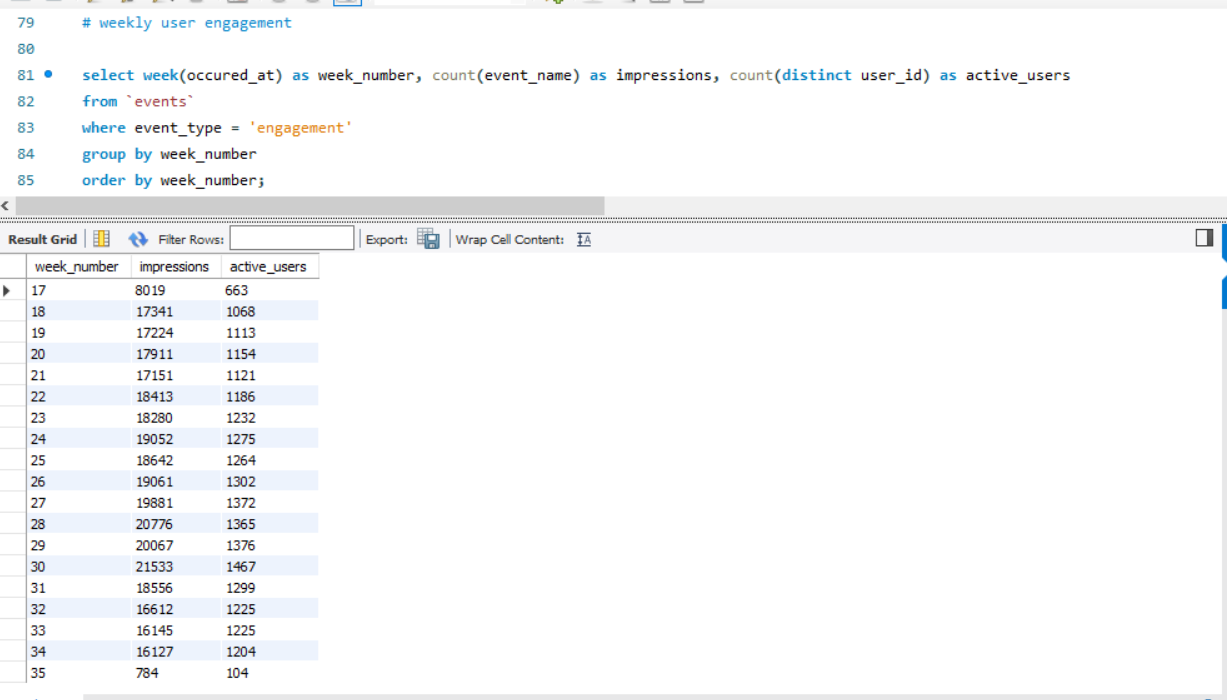
Likewise, we can do the above steps for the other two tables.

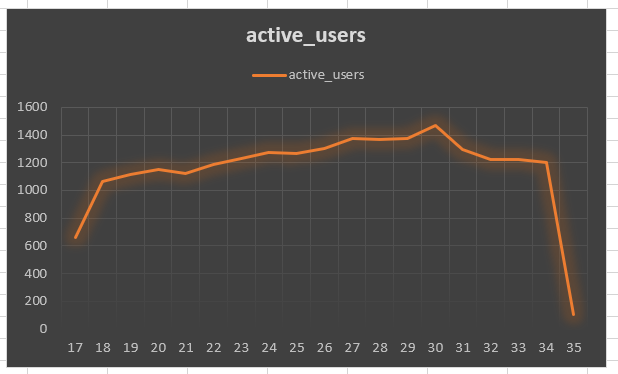
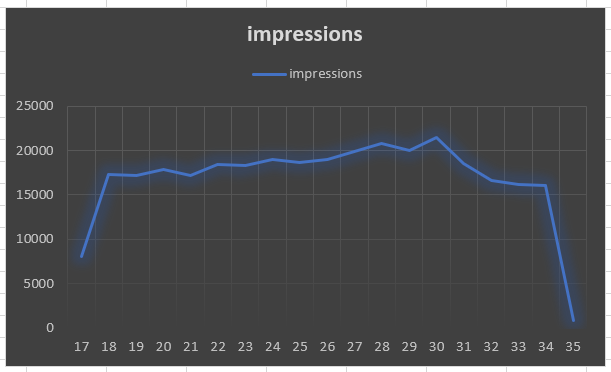
Problem Description No. 1:

Weekly User Engagement:

Objective: Measure the activeness of users on a weekly basis.  
Task: Write an SQL query to calculate the weekly user engagement.

**Approach:**

1. Breakdown the problem into the keywords – active users, weekly basis, engagement.
2. Table identified from the above keywords – events.
3. Find the week number for each row in occurred\_at column using the **week()** function.
4. We will use this week\_number column to group the user engagement on a **weekly** basis.
5. Filter only those events where event\_type is ‘engagement’.
6. Next, count the event\_name column and store as ‘impressions’.
7. Count **distinct** user\_id and store as active\_users.
8. Order the result in ascending order of week\_number.  
   

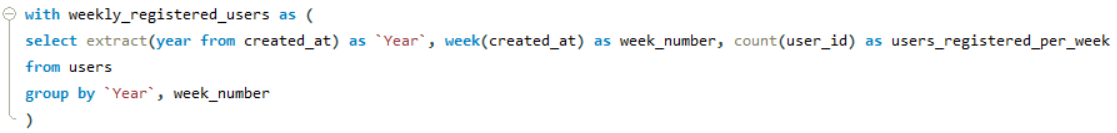
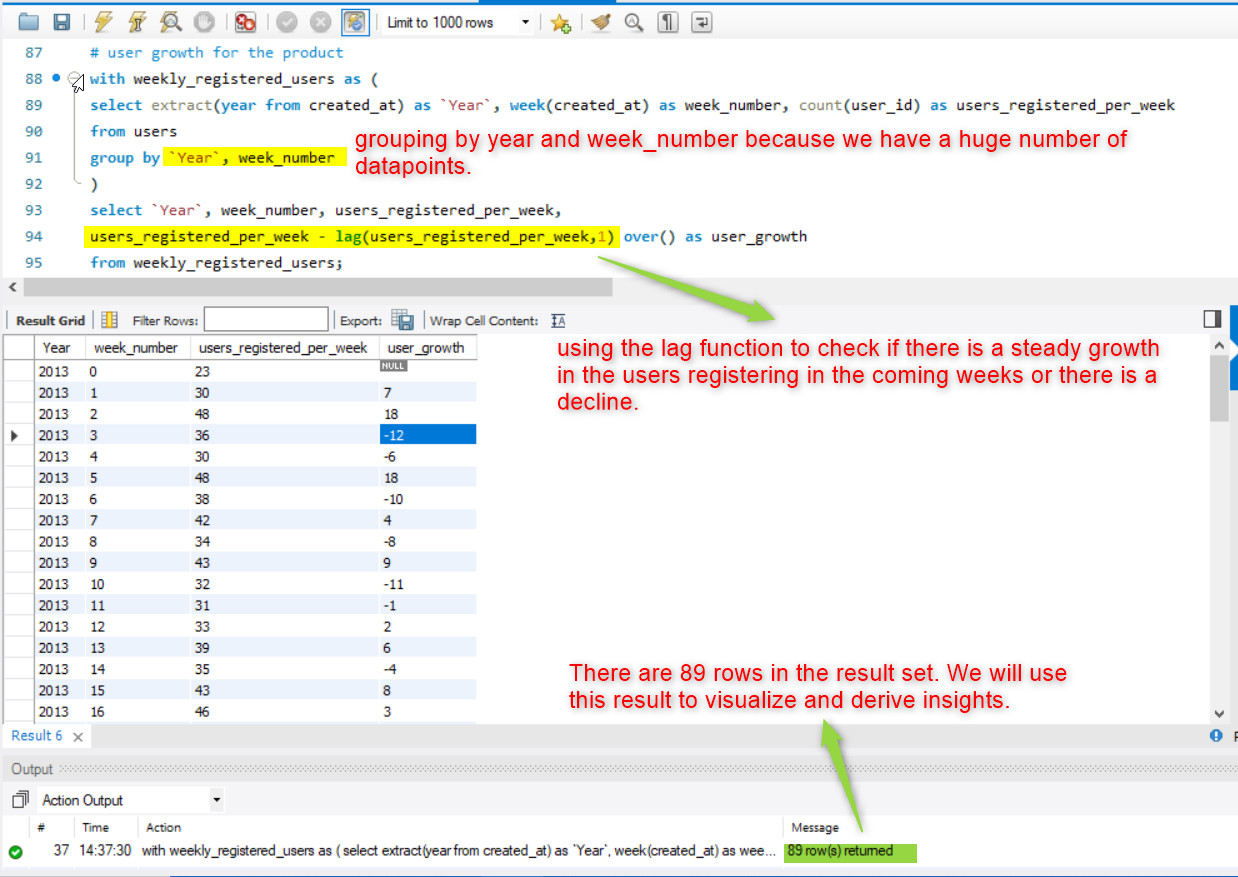
**Insights:**  
Looking at the active users we can see that there is a drop in week 35, whereas week 30 shows the highest active users. Until week 30 there is a steady increase in the number of active users but after week 30, they start declining, however there is a sharp decline in week 35 from week 34 – from around 1200 users to less than 200. We have a similar visual in terms of impressions. This drop can be tracked by the relevant team to find out the reasons and prevent it in the future. Similarly, they can find out what was particularly interesting in week 30 that it recorded highest number of active users.  
 

Problem Description No. 2:

User Growth Analysis:

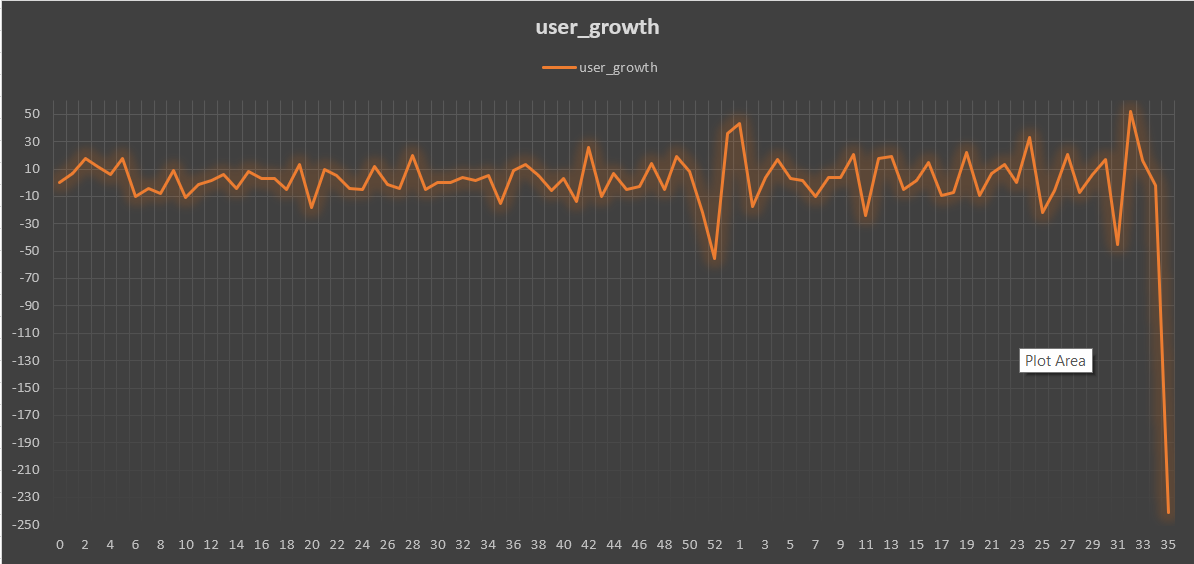
Objective: Analyze the growth of users over time for a product.

**Approach:**

1. Keywords – users and time.
2. In the users table we see that we have the data over two years – 2013 and 2014. So, we will extract both year and week from the created\_at column using the functions **extract (year from <date>)** and **week ()**. These new columns we will store in a temporary table.
3. Next, we need to find out the growth so we must choose a period over which we wish to check the growth. Since the datapoints are too large we will group them together on weekly basis over the two years.
4. After this we will count the users registered per week over these two years by counting the user\_id column.  
   
5. The above query gives us the distinct count for the weeks separately and does not allow us to check the growth or the increase in the users over the weeks. Thus, we need to find the growth with respect to the previous week as we move forward each week. Here we use **lag ()** function as shown below:  
   

**Insights:**

When we look at the user growth, we see that it is not steady but in the last week 52 of 2013 we see a sharp decline in users who newly register (-55) and in 2014, week 35 this is even more (-241) as compared to the previous week. This can help the relevant team to find out the reasons for such a sharp decline – maybe the ads need to be more engaging, or more frequent, rewards system, geographical findings, etc.

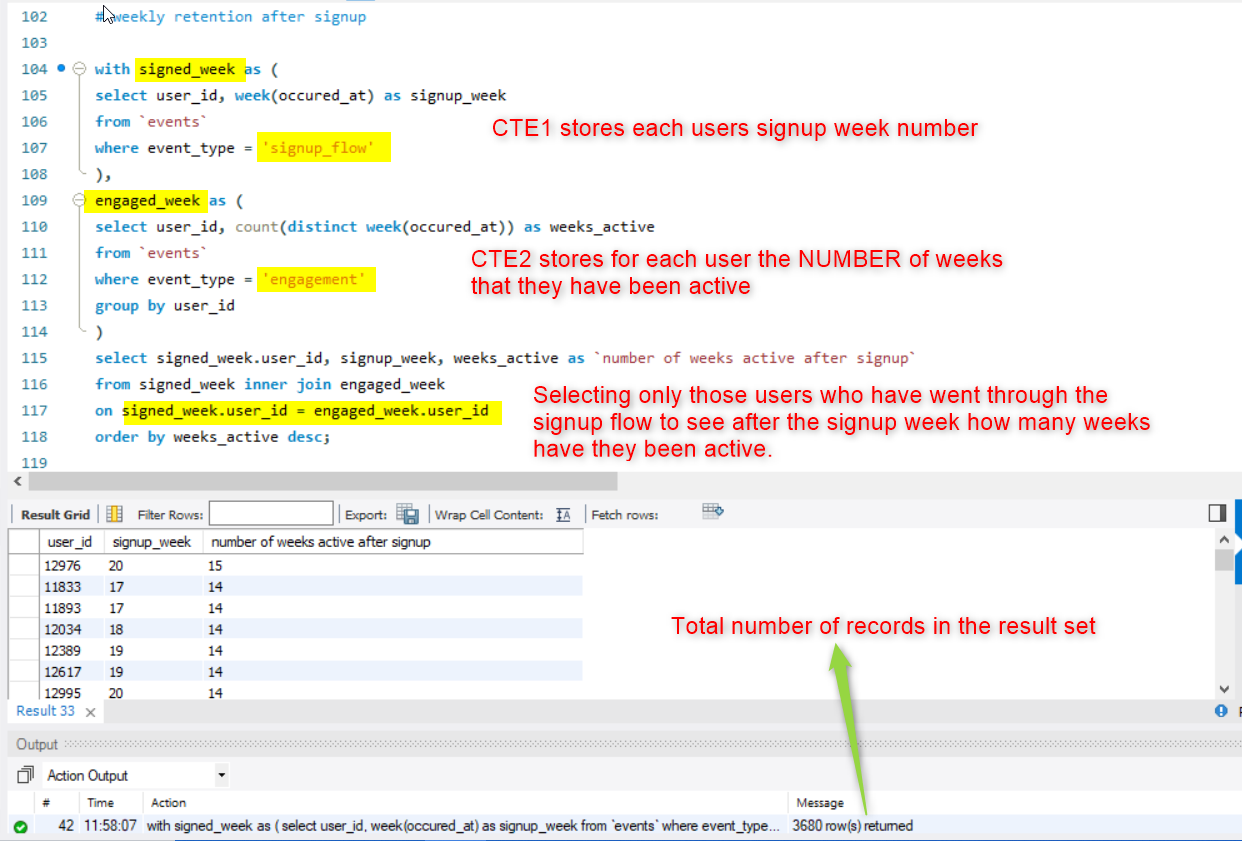


Problem Description No. 3:

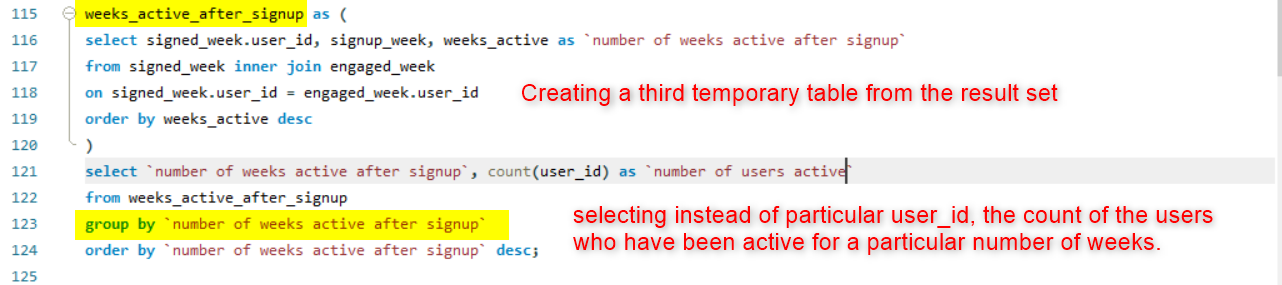
Weekly Retention Analysis:

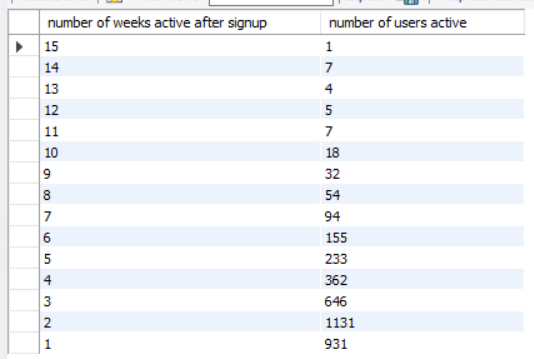
Objective: Analyze the retention of users on a weekly basis after signing up for a product.

**Approach:**

1. We first find out the signup week number of each user\_id. Let’s call this user\_id U and the week that U has signed up as W.
2. After that, our task is to find out the count of the number of weeks U has been engaging after W.
3. Create a temporary table signed\_week and store the user\_id where event\_type is signup\_flow.
4. Create a temporary table engaged\_week and for each user store the distinct count of weeks where event\_type is engagement.
5. Finally join the two tables on the condition that user\_ids are the same because we only want to find out the engagement for the users who have completed the signup flow.  
   

**Insights:**

From the result set we can see that the user 12976 has been active for the most number of weeks (15) after signing up on week 20. With this result set we can also count the number of users who have been active for a particular number of weeks after signup.   
Our query will look like this: In this query we can remove the line number 119 to save processing time since we need to order the result based on the new attribute derived later.  




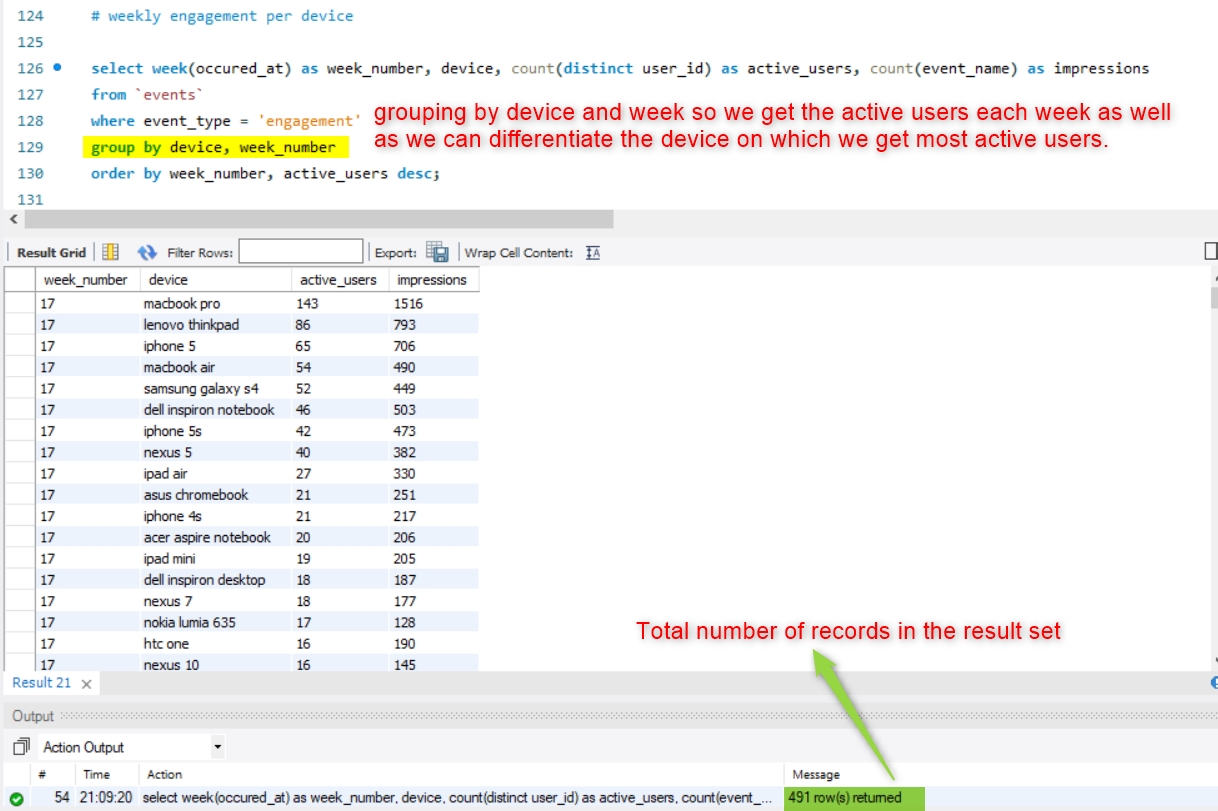
Now, we can see that only 1 user has been active for the most number of weeks, i.e., 15 whereas there are 931 users who have been active for only 1 week after signup. Most number of users – 1131 – have been active for 2 weeks after signup.

Problem Description No. 4:

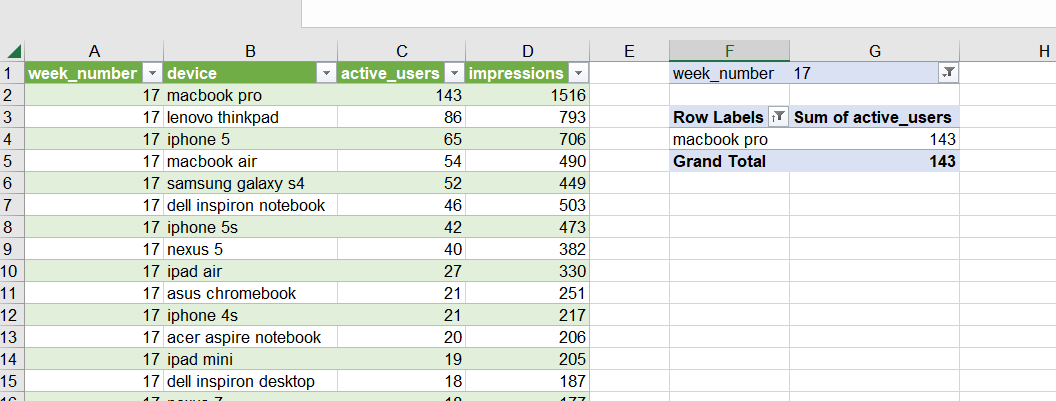
Weekly Engagement Per Device:

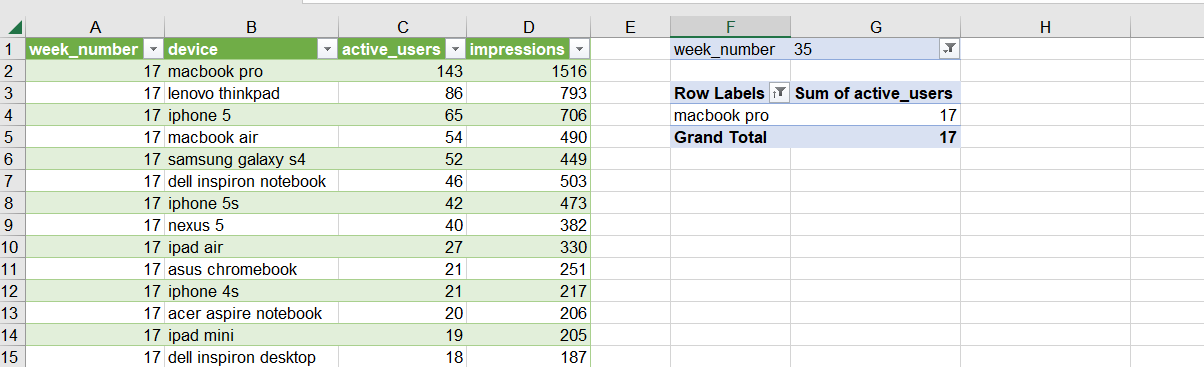
Objective: Measure the activeness of users on a weekly basis per device.

**Approach:**

This problem is similar to weekly user engagement except here we have one additional attribute “device” to group the records.  


**Insights:**

When we visualize the results, we can see that for every week the device **macbook pro** has reported highest number of active users. I have used pivot chart to select the top device for each week.  




Problem Description No. 5:

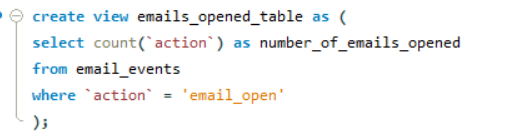
Email Engagement Analysis:

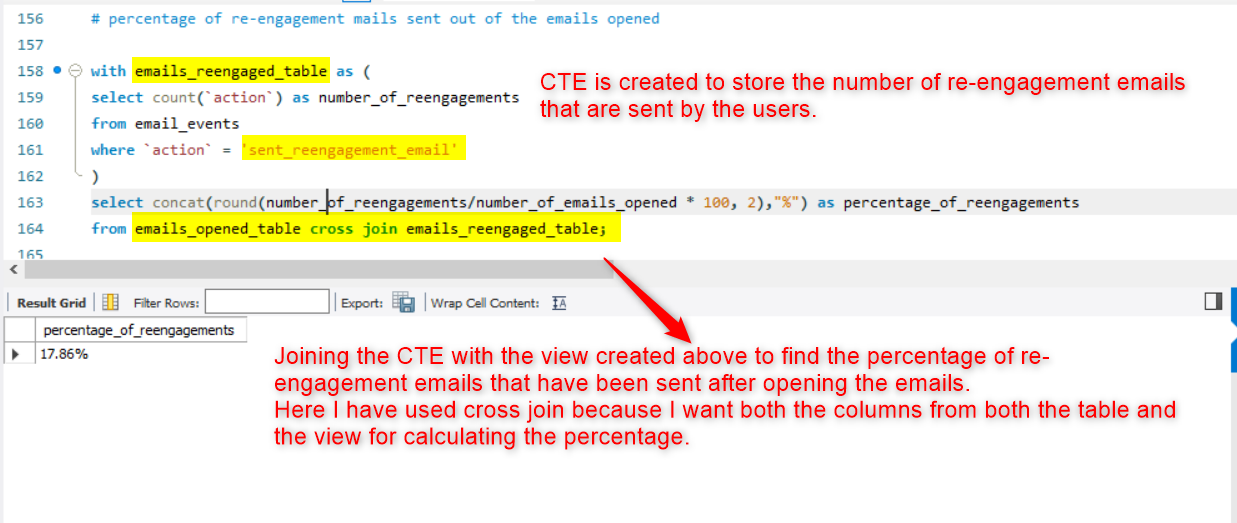
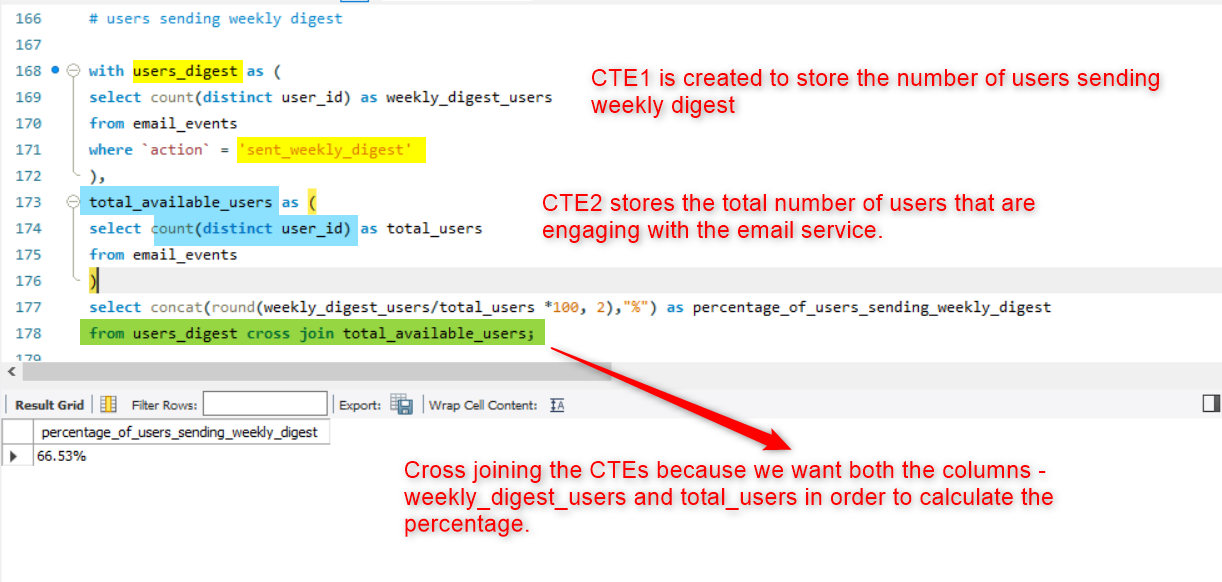
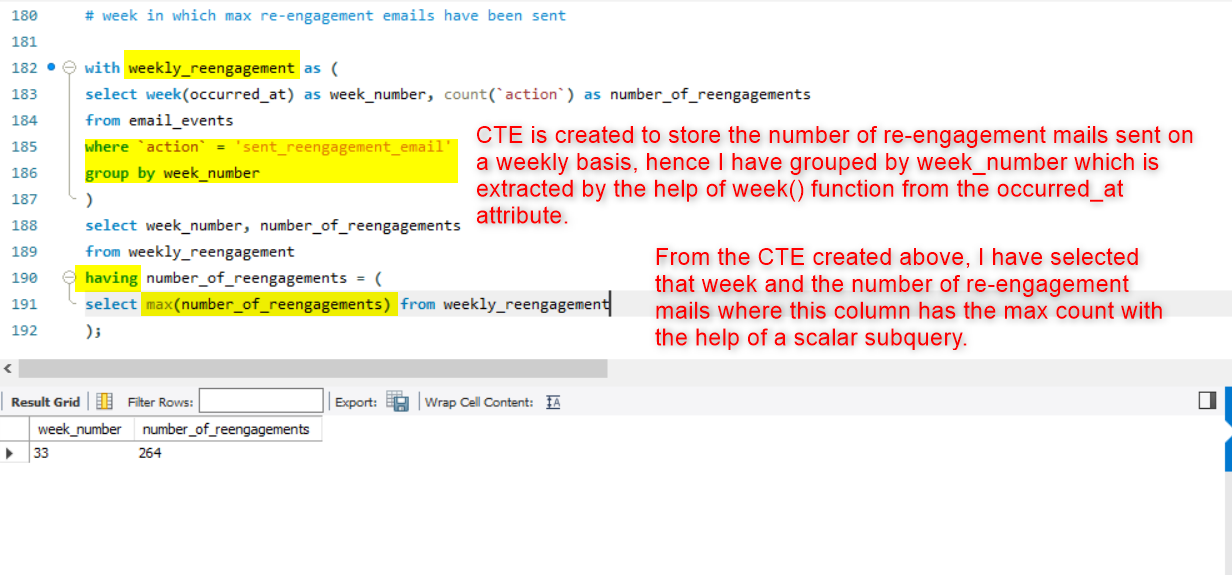
Objective: Analyze how users are engaging with the email service.

**Approach:**

To analyze the engagement with the email service we will calculate 4 things from the **email\_events** table provided:

1. Percentage of emails clicked through out of the emails opened.
2. Percentage of re-engagement mails sent out of the emails opened.
3. Percentage of users sending weekly digest.
4. Identify the week in which maximum number of re-engagement emails have been sent.

We will create a **view** this time called ‘emails\_opened\_table’ because we will use this in multiple queries for identifying the first 2 points. The two above points will be based on the number of emails that have been opened so this view will simply have the count of the emails that have been opened.  


1. Query for problem number 1. is shown below:  
   
2. Query for problem number 2. is shown below:  
   
3. Query for problem number 3.  
   
4. Query for problem number 4.:  
   

**Insights:**

1. From the results obtained above we can deduce, that out of the number of emails opened, only 44.04% are clicked- through whereas only 17.86% have been re-engaged with. This percentage shows that the emails that are sent to users are not as engaging as should be. The percentage showing clicked-through count suggests that while the **subject line or extract of the email could be improved**, the percentage showing re-engagement directly points out to the fact that **major attention and priority needs to be given to the matter of the emails**.
2. Further we can see that in **week number 33 maximum number of re-engagement emails (264)** have been sent. This can be analyzed by the respective team to check what type of content was sent in the emails sent in this week and similar efforts can be made to improve the engagements in the future.
3. Out of the total users engaging with the email service only 66.53% are sending out weekly digests which is a good number. This insight can be combined with the above two to improve the percentage of users sending weekly digest even more. If more users send re-engagement mails, they will be motivated to send their own weekly digests as well.

**Result:**

With this project, my grip on SQL has strengthened even more. But more specifically I have learnt how to visualize the results of my analysis and provide meaningful insights that can help the business grow. Most of these insights have proven that I will be closely in contact with the various teams that are involved in a business. It has also shown me how as a data analyst, I will take the pivotal role in directing a business’s future – analyzing trends continuously and paying keen attention to details that might otherwise be overlooked while making valuable decisions.