**SQL Case Study – 1**

**‘Operation Analytics’**

By: Shrey Shah

**PROJECT DESCRIPTION:**

The case study describes the ‘Operation Analytics’ where we have a table which stores the details about the jobs as follows:

* job\_id: unique identifier of jobs
* actor\_id: unique identifier of actor
* event: decision/skip/transfer
* language: language of the content
* time\_spent: time spent to review the job in seconds
* org: organization of the actor,
* ds: date in the yyyy/mm/dd format.

Some observations:

* The job\_id and actor\_id have been specified as unique keys for both but in any table we can have only 1 primary key.
* For convenience, we have not kept any key as the primary key because we have few records and we have to perform some queries and cannot afford to lose data.
* As we clearly do not have information or resources about the event column we will consider all 3 as the categories in the event.

In this project we are going to get answers of some questions for the analysis purpose by performing some querying on the given data.

QA: Calculate the number of jobs reviewed per hour per day for November 2020?

QB: Let’s say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

QC: Calculate the percentage share of each language in the last 30 days?

QD: Let’s say you see some duplicate rows in the data. How will you display duplicates from the table?

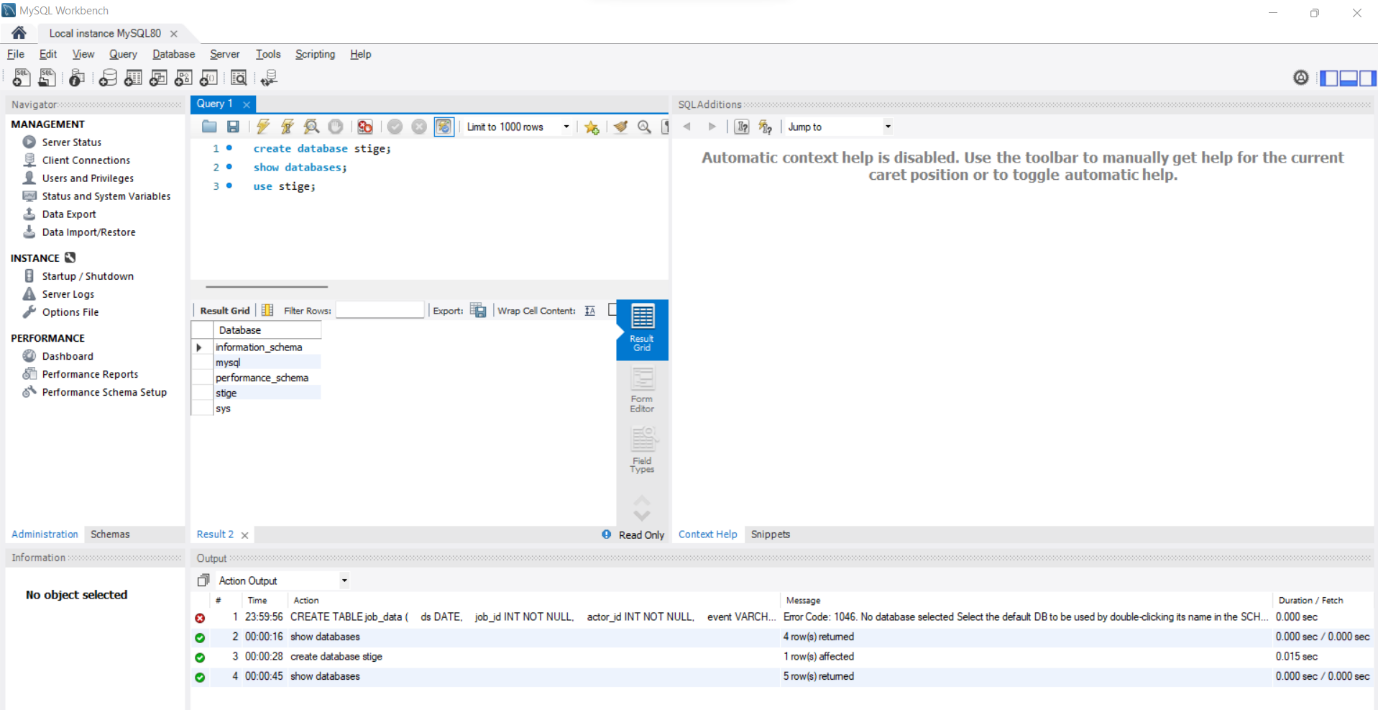
**APPROACH:**

For this project, I have used the small dataset of 8 records provided to develop queries to find answers of the questions posed above for the analytics purpose.

I have used MySQL Workbench for trying the SQL queries.

Initially, I established a local connection with the MySQL Server using the Workbench.

Next, I created a database named ‘stige’ for personal use as follows.



Once this was done, I created a table named ‘job\_data’ with all the columns as described above with the SQL query:

CREATE TABLE job\_data

(

ds DATE,

job\_id INT NOT NULL,

actor\_id INT NOT NULL,

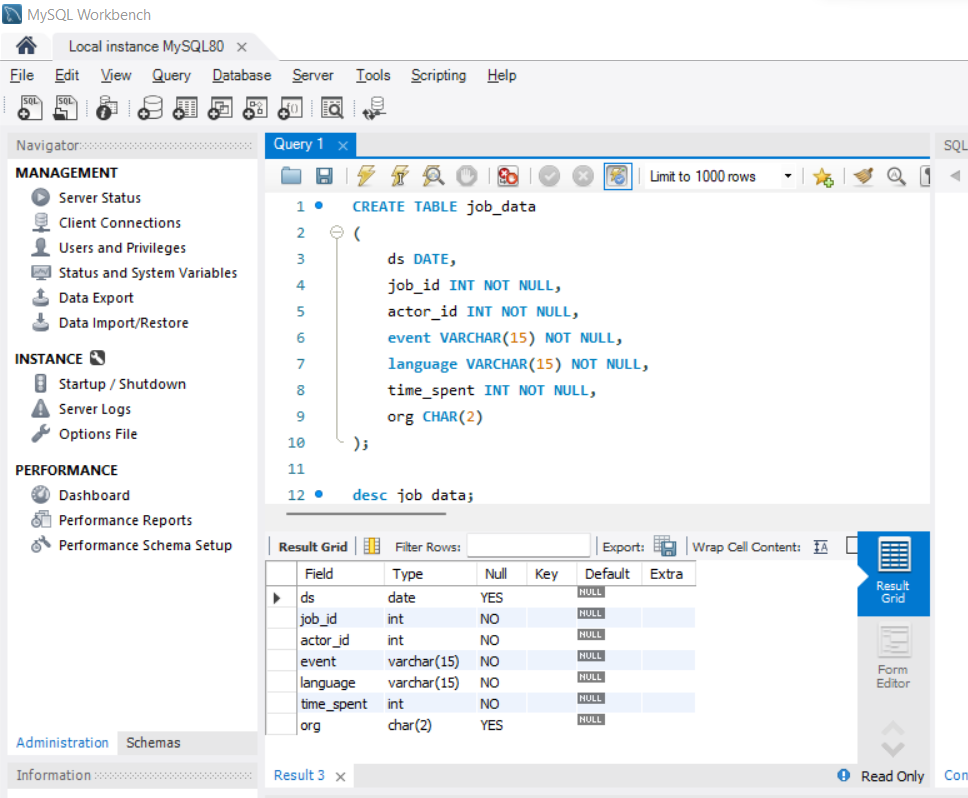
event VARCHAR(15) NOT NULL,

language VARCHAR(15) NOT NULL,

time\_spent INT NOT NULL,

org CHAR(2)

);



After this finally, I added the dataset rows with the following query and displayed the result:

INSERT INTO job\_data (ds, job\_id, actor\_id, event, language, time\_spent, org)

VALUES ('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),

('2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),

('2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),

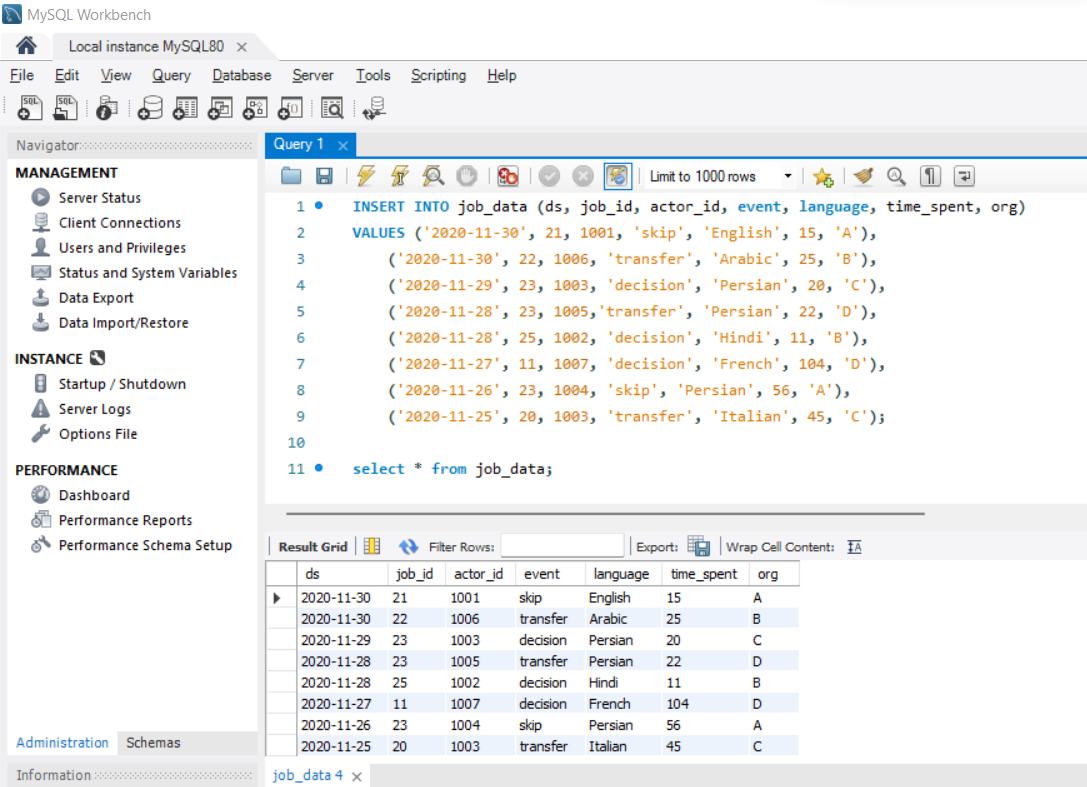
('2020-11-28', 23, 1005,'transfer', 'Persian', 22, 'D'),

('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),

('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),

('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),

('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');



**TECHSTACK USED:**

I installed the MySQL Installer for Windows.

Version: 8.0.28.0

It internally downloads the MySQL Server, MySQL Workbench and MySQL Shell all having the same version.

I have used MySQL Installer rather than MySQL Workbench (standalone) because the Installer is a compact utility which internally downloads the above mentioned applications and reduces the job of manually installing all of them.

Since we have the case study in SQL, I chose this technology and application.

**INSIGHTS:**

QA) Calculate the number of jobs reviewed per hour per day for November 2020?

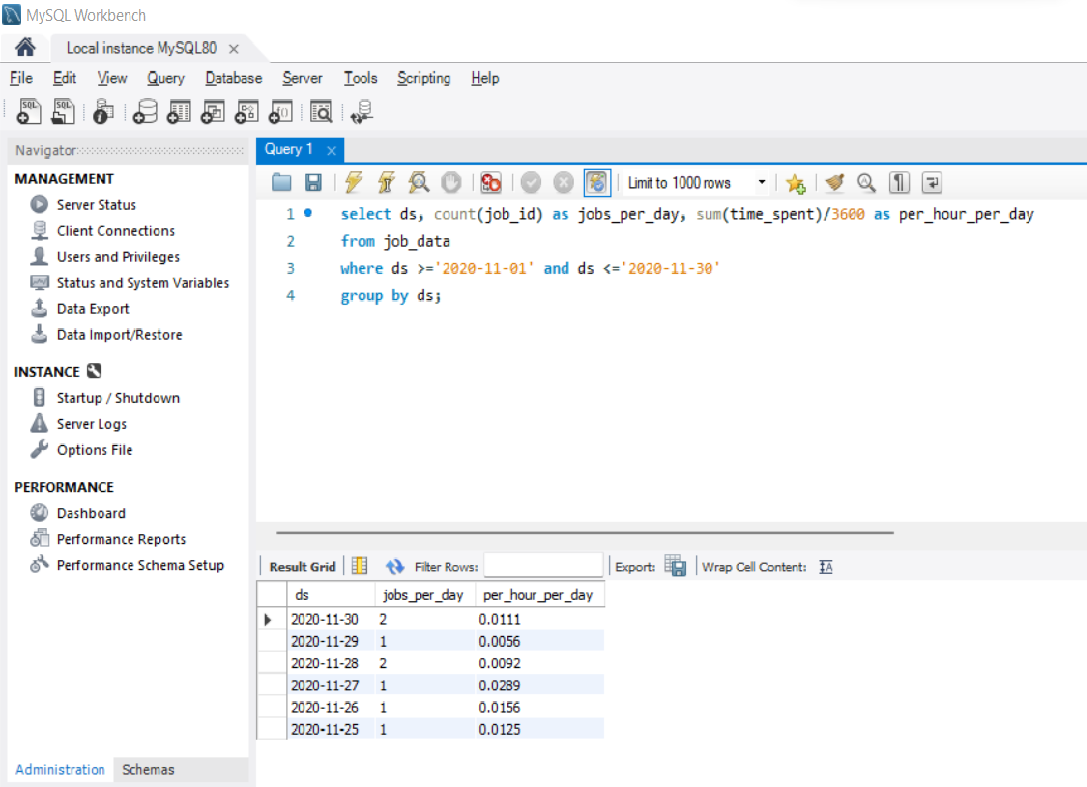
select ds, count(job\_id) as jobs\_per\_day, sum(time\_spent)/3600 as per\_hour\_per\_day

from job\_data

where ds >='2020-11-01' and ds <='2020-11-30'

group by ds;

In this query, we have calculated the number of jobs grouped by their dates and the time spent per hour per day based on the time in seconds / 3600.



QB) Let’s say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

with 7day\_rolling\_avg as(select ds, COUNT(job\_id) as number\_of\_jobs, SUM(time\_spent) as total\_time

from job\_data

where ds >= '2020-11-01' AND ds <= '2020-11-30'

group by ds

)

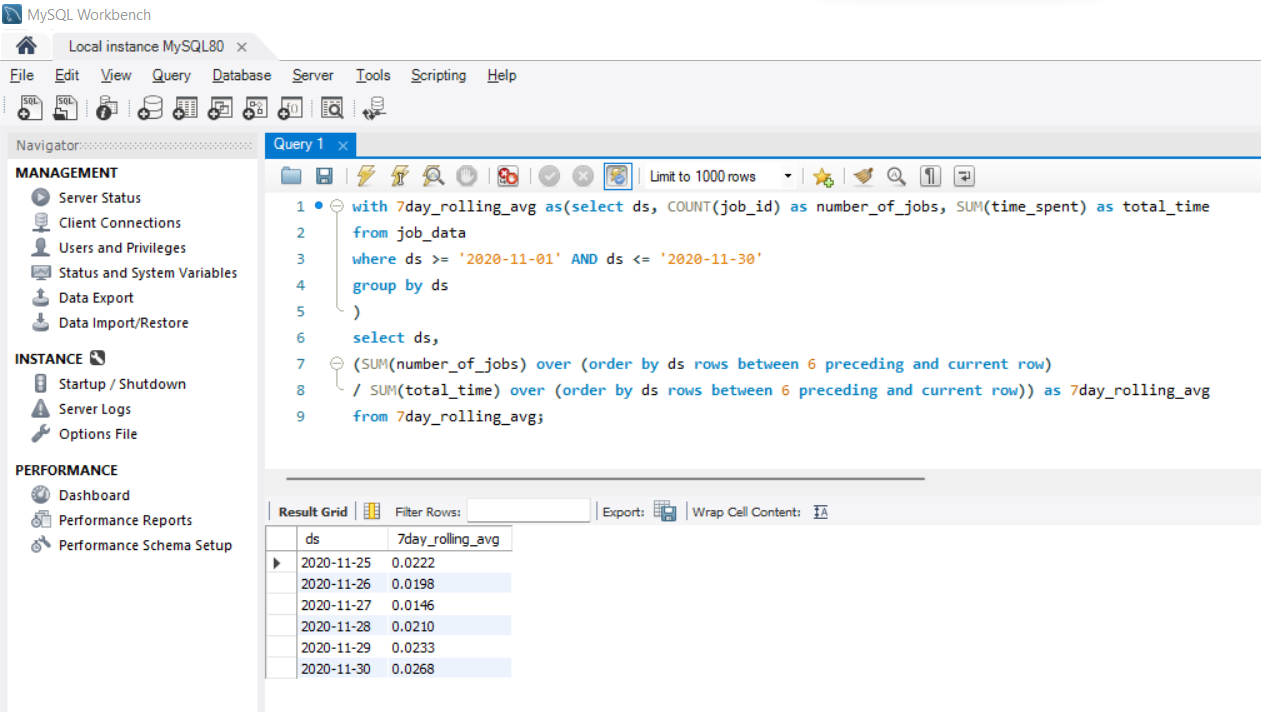
select ds,

(SUM(number\_of\_jobs) over (order by ds rows between 6 preceding and current row)

/ SUM(total\_time) over (order by ds rows between 6 preceding and current row)) as 7day\_rolling\_avg

from 7day\_rolling\_avg;

In this query, we have mixed 2 queries. Firstly we find out the number of jobs and the total time of the jobs grouped by date. Using this output as the input table for the second query, we have found out the cumulative average using the 6 rows above the current row and the current row to find the 7 day cumulative / running average.



QC) Calculate the percentage share of each language in the last 30 days?

with percent\_share\_language as (

select language, count(language) as count\_of\_language

from job\_data

where ds >= '2020-11-01' and ds <= '2020-11-30'

group by language),

total as (

select count(language) as total\_language

from job\_data

where ds >= '2020-11-01' and ds <= '2020-11-30'

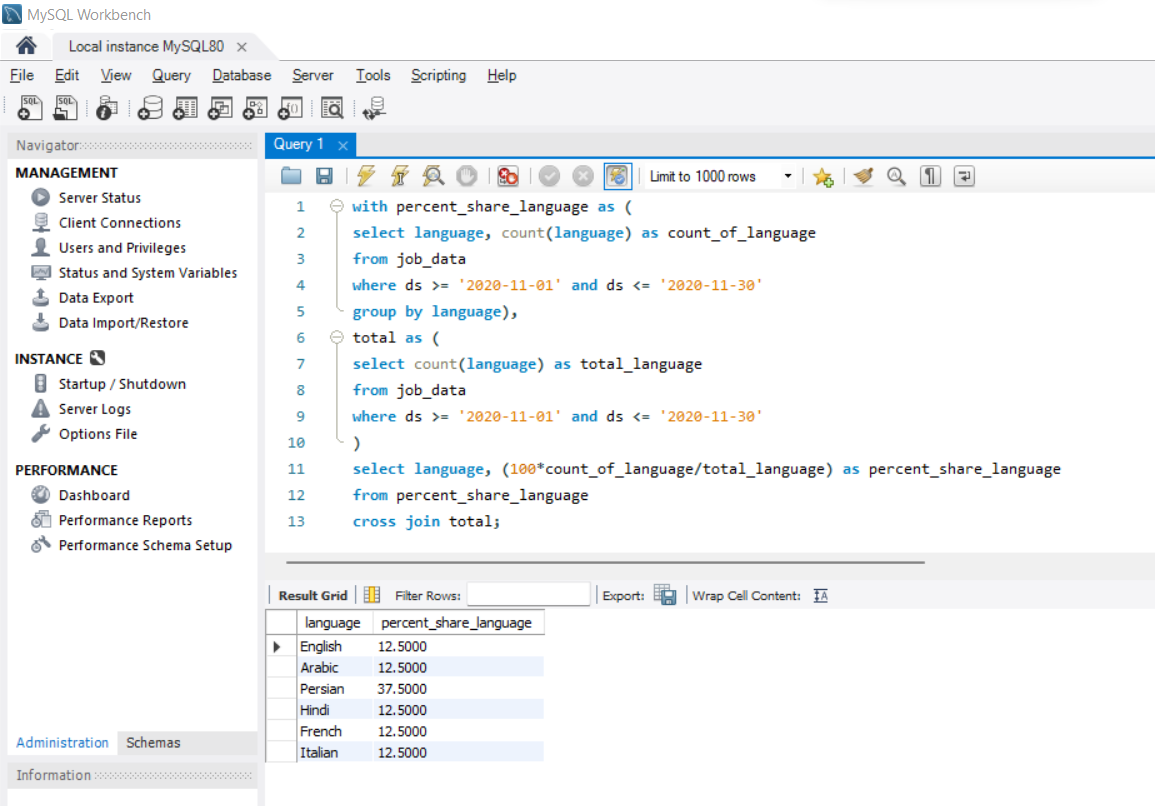
)

select language, (100\*count\_of\_language/total\_language) as percent\_share\_language

from percent\_share\_language

cross join total;

In this query, we are again using 3 queries. Firstly, we are counting the number of individual languages / jobs grouped by the languages. Secondly, we are calculating the sum of all languages / jobs as the total. Lastly, we are performing the ratio of the individual language counts by the total and multiply by 100 to get the percentage of language share.



QD) Let’s say you see some duplicate rows in the data. How will you display duplicates from the table?

with duplicate\_rows as (

select \*, row\_number() over (partition by ds, job\_id, actor\_id) as rownum

from job\_data)

delete from duplicate\_rows

where rownum > 1;

In this query, we are performing the display and deletion or the duplicate records. We are creating a new column ‘rownum’ which stores the count. If a row is repeated then the rownum will hold the value > 1 and that indicates that it is a duplicate record which can be deleted. In our dataset, we did not have any such duplicate records and hence there will be no result.

**RESULTS:**

* Working on this SQL project, has helped me understand the very new and complex, advanced functions and SQL writing techniques.
* I did trial and error many times with new functions like Windows functions, and writing multiple queries within a single query.
* This project has helped me improve my SQL querying skills to the next level from earlier when I just was able to work with basic SQL clauses.

**SQL Case Study – 2**

**‘Investigating Metric Spike’**

By: Shrey Shah

**PROJECT DESCRIPTION:**

The case study deals with the dip in the weekly engagement and the need to investigate to identify the root cause behind the dip. The head of the Product team walks over to your desk and asks you to investigate the dip in weekly engagement.

Table-1: users

This table includes one row per user, with descriptive information about that user’s account.

|  |  |
| --- | --- |
| **user\_id** | A unique ID per user. Can be joined to user\_id in either of the other tables. |
| **created\_at** | The time the user was created (first signed up) |
| **state** | The state of the user (active or pending) |
| **activated\_at** | The time the user was activated, if they are active |
| **company\_id** | The ID of the user’s company |
| **language** | The chosen language of the user |

**Table-2: events**

This table includes one row per event, where an event is an action that a user has taken. These events include login events, messaging events, search events, events logged as users progress through a signup funnel, events around received emails.

|  |  |
| --- | --- |
| **user\_id** | The ID of the user logging the event. Can be joined to user\\_id in either of the other tables. |
| **occurred\_at** | The time the event occurred. |
| **event\_type** | The general event type. There are two values in this dataset: “signup\_flow”, which refers to anything occuring during the process of a user’s authentication, and “engagement”, which refers to general product usage after the user has signed up for the first time. |
| **event\_name** | The specific action the user took. Possible values include: create\_user: User is added to Yammer’s database during signup process enter\_email: User begins the signup process by entering her email address enter\_info: User enters her name and personal information during signup process complete\_signup: User completes the entire signup/authentication process home\_page: User loads the home page like\_message: User likes another user’s message login: User logs into Yammer search\_autocomplete: User selects a search result from the autocomplete list search\_run: User runs a search query and is taken to the search results page search\_click\_result\_X: User clicks search result X on the results page, where X is a number from 1 through 10. send\_message: User posts a message view\_inbox: User views messages in her inbox |
| **location:** | The country from which the event was logged (collected through IP address). |
| **device:** | The type of device used to log the event. |

**Table-3: email\_events**

This table contains events specific to the sending of emails. It is similar in structure to the events table above.

|  |  |
| --- | --- |
| **user\_id** | The ID of the user to whom the event relates. Can be joined to user\_id in either of the other tables. |
| **occurred\_at** | The time the event occurred. |
| **action** | The name of the event that occurred. “sent\_weekly\_digest” means that the user was delivered a digest email showing relevant conversations from the previous day. “email\_open” means that the user opened the email. “email\_clickthrough” means that the user clicked a link in the email. |

In this project we are going to get answers of some questions for the analysis purpose by performing some querying on the given data.

QA: Calculate the weekly user engagement?

QB: Calculate the user growth for product?

QC: Calculate the weekly retention of users-sign up cohort?

QD: Calculate the weekly engagement per device?

QE: Calculate the email engagement metrics

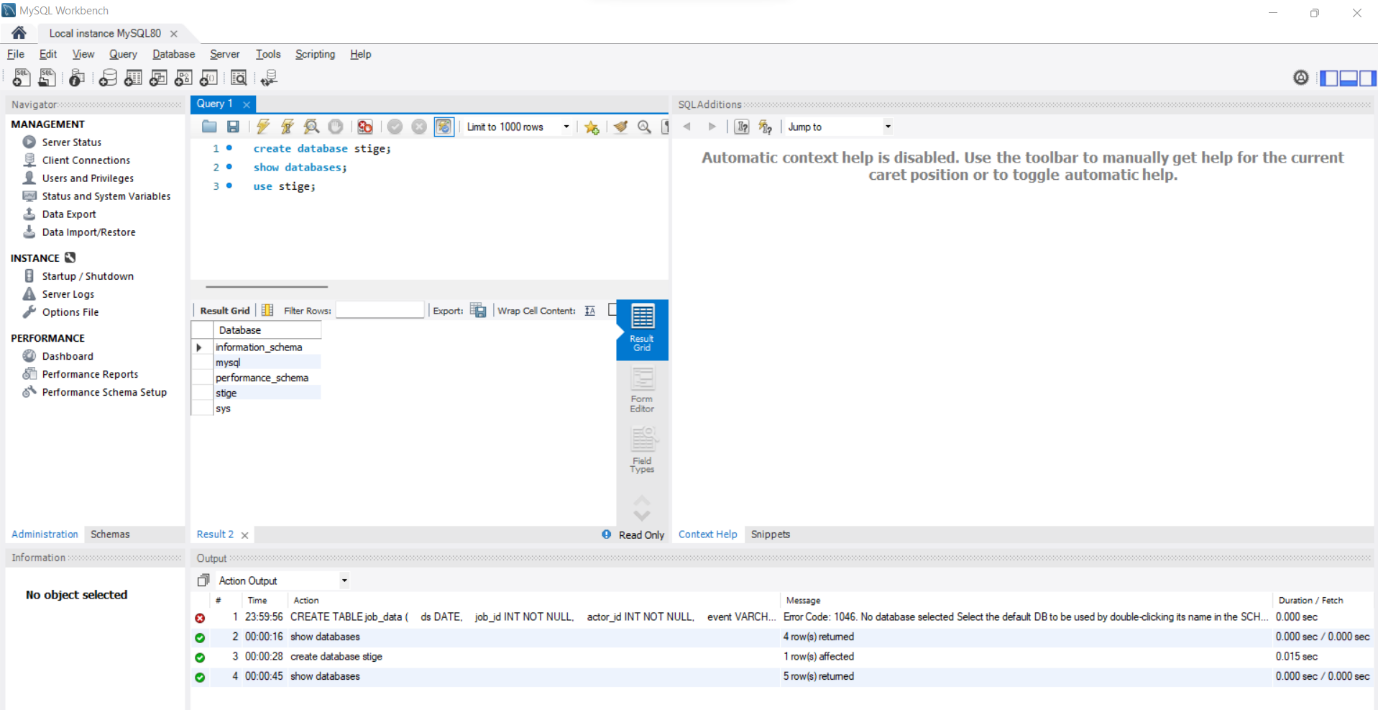
**APPROACH:**

For this project, I have used the attached dataset provided to develop queries to find answers of the questions posed above for the analytics purpose.

I have used MySQL Workbench for trying the SQL queries.

Initially, I established a local connection with the MySQL Server using the Workbench.

Next, I created a database named ‘stige’ for personal use as follows.



* Next, I chose the schema ‘stige’
* Select ‘Tables’
* Right-Click > Table Data Import Wizard
* Specify the path of the .csv files where they are located
* Choose the Create New Table option and check the ‘Drop if table exists’ option.
* Click Next and import will start.

All the records from users and email\_events table were loaded. But due to CPU and MySQL constrains only 248806 records were loaded from 380000 records in the events table. So results may vary.

**TECHSTACK USED:**

I installed the MySQL Installer for Windows.

Version: 8.0.28.0

It internally downloads the MySQL Server, MySQL Workbench and MySQL Shell all having the same version.

I have used MySQL Installer rather than MySQL Workbench (standalone) because the Installer is a compact utility which internally downloads the above mentioned applications and reduces the job of manually installing all of them.

Since we have the case study in SQL, I chose this technology and application.

**INSIGHTS:**

QA) Calculate the weekly user engagement?

select extract(week from e.occurred\_at) as week\_number,

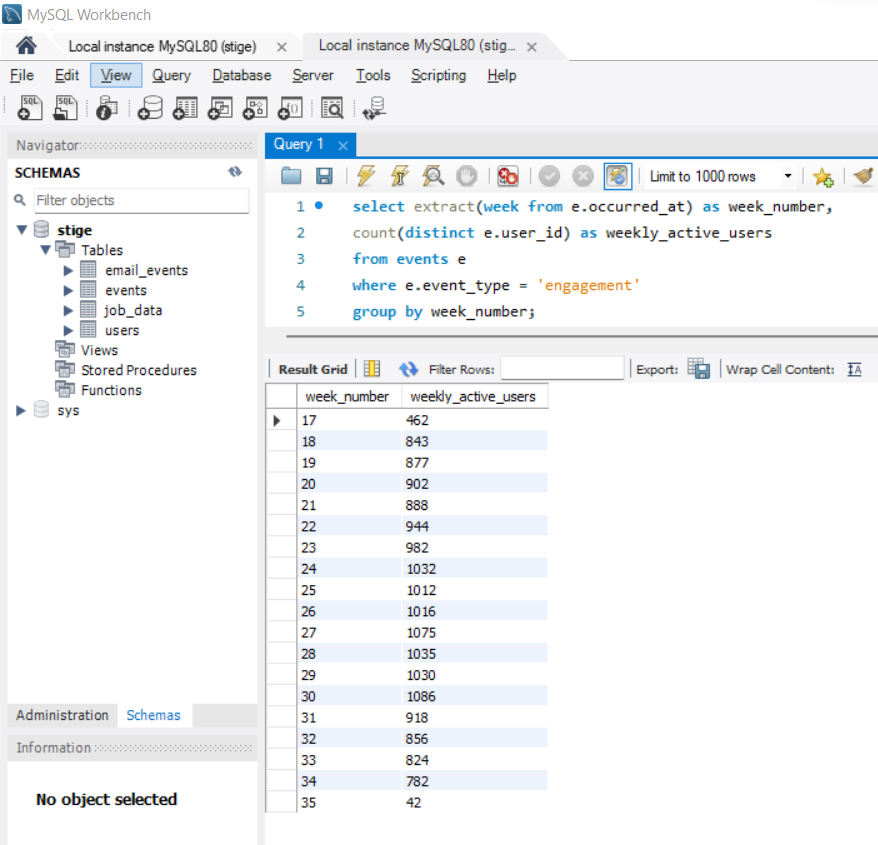
count(distinct e.user\_id) as weekly\_active\_users

from events e

where e.event\_type = 'engagement'

group by week\_number;

In this query, we have used the extract() function to find the week number of the occurred\_at date in the events table. Then using the user\_id column in the events table we have found the distinct users for whom the event\_type is engagement and the results are grouped by the week number.



QB) Calculate the user growth for product?

select extract(day from created\_at) as day,

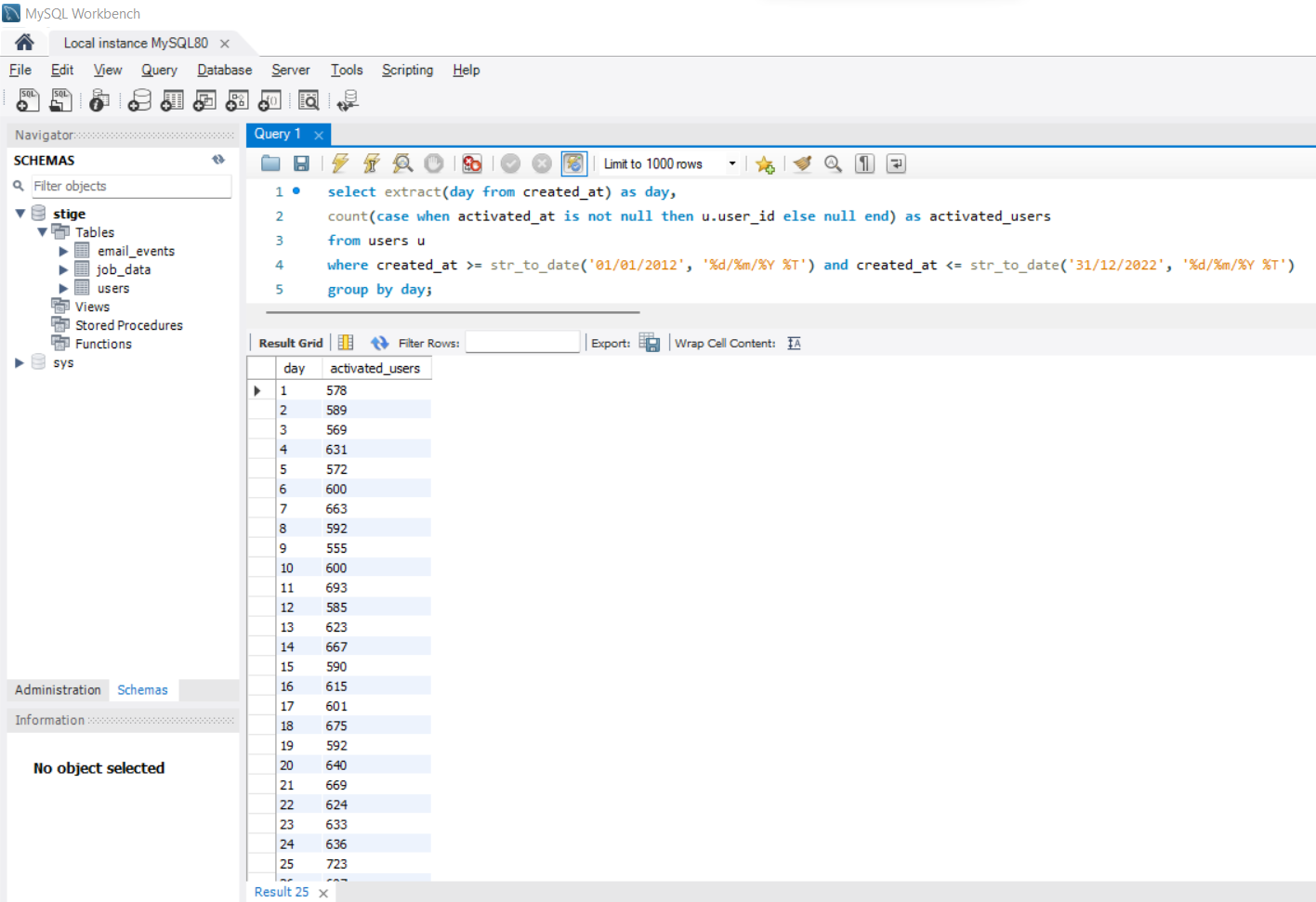
count(case when activated\_at is not null then u.user\_id else null end) as activated\_users

from users u

where created\_at >= str\_to\_date('01/01/2012', '%d/%m/%Y %T') and created\_at <= str\_to\_date('31/12/2022', '%d/%m/%Y %T')

group by day;

In this query, we have calculated the daily users who are active which indicates the user growth on a daily basis. The date range is from 2012 to 2022 to include all the years in between even though we don’t have data.



QC) Calculate the weekly retention of users-sign up cohort?

select extract(week from z.occurred\_at) as week,

AVG(z.age\_at\_event) as 'Average age durig week',

count(distinct case when z.user\_age > 70 then z.user\_id ELSE

NULL END) as '10+ weeks',

count(distinct case when z.user\_age < 70 and z.user\_age >=63

then z.user\_id else null end) as '9 weeks',

count(distinct case when z.user\_age < 63 and z.user\_age >=56

then z.user\_id else null end) as '8 weeks',

count(distinct case when z.user\_age < 56 and z.user\_age >=49

then z.user\_id else null end) as '7 weeks',

count(distinct case when z.user\_age < 49 and z.user\_age >=42

then z.user\_id else null end) as '6 weeks',

count(distinct case when z.user\_age < 42 and z.user\_age >=35

then z.user\_id else null end) as '5 weeks',

count(distinct case when z.user\_age < 35 and z.user\_age >=28

then z.user\_id else null end) as '4 weeks',

count(distinct case when z.user\_age < 28 and z.user\_age >=21

then z.user\_id else null end) as '3 weeks',

count(distinct case when z.user\_age < 21 and z.user\_age >=14

then z.user\_id else null end) as '2 weeks',

count(distinct case when z.user\_age < 14 and z.user\_age >=7

then z.user\_id else null end) as '1 weeks',

count(distinct case when z.user\_age < 7 and z.user\_age >=63

then z.user\_id else null end) as 'Less than a week'

from(

select e.occurred\_at, u.user\_id, extract(week from u.activated\_at) as activation\_week,

extract(day from datediff(e.occurred\_at, u.activated\_at)) as age\_at\_event,

extract(day from datediff(u.created\_at, u.activated\_at)) as user\_age

from users u

join events e

on e.user\_id = u.user\_id

and e.event\_type = 'engagement'

and e.event\_name= 'login'

and e.occurred\_at >= '2012-05-01'

and e.occurred\_at < '2022-09-01'

where u.activated\_at is not null

) z

group by week;

QD) Calculate the weekly engagement per device?

select extract(week from occurred\_at) as week,

count(distinct e.user\_id) as weekly\_active\_users,

count(distinct case when e.device in('macbook pro','lenovo thinkpad','macbook air','dell inspiron notebook','asus chromebook', 'dell inspiron desktop','acer aspire notebook','hp pavilion desktop','acer aspire desktop','mac mini')

then e.user\_id else null end) as computer,

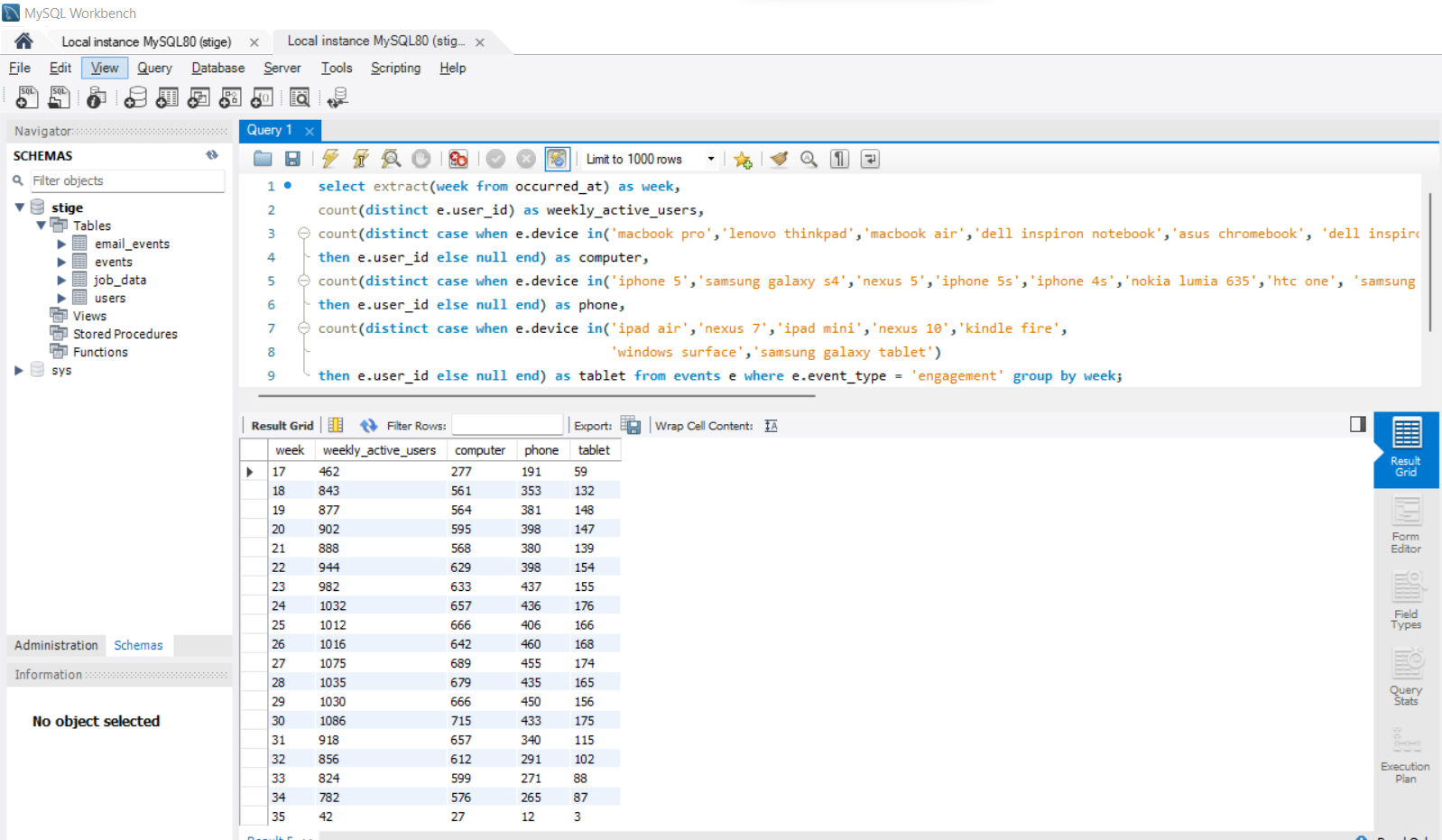
count(distinct case when e.device in('iphone 5','samsung galaxy s4','nexus 5','iphone 5s','iphone 4s','nokia lumia 635','htc one', 'samsung galaxy note','amazon fire phone')

then e.user\_id else null end) as phone,

count(distinct case when e.device in('ipad air','nexus 7','ipad mini','nexus 10','kindle fire', 'windows surface','samsung galaxy tablet')

then e.user\_id else null end) as tablet from events e where e.event\_type = 'engagement'

In this query, we have calculated the weekly active users and their distribution based on the devices we have and we have applied the case clause to check if the device belongs either to the computer, phone or tablet category. The results are grouped by week.



QE: Calculate the email engagement metrics?

select extract(week from occurred\_at) as week,

count(case when e.action = 'sent\_weekly\_digest' then e.user\_id else null end) as weekly\_emails,

count(case when e.action = 'sent\_reengagement\_email' then e.user\_id else null end) as reengagement\_emails,

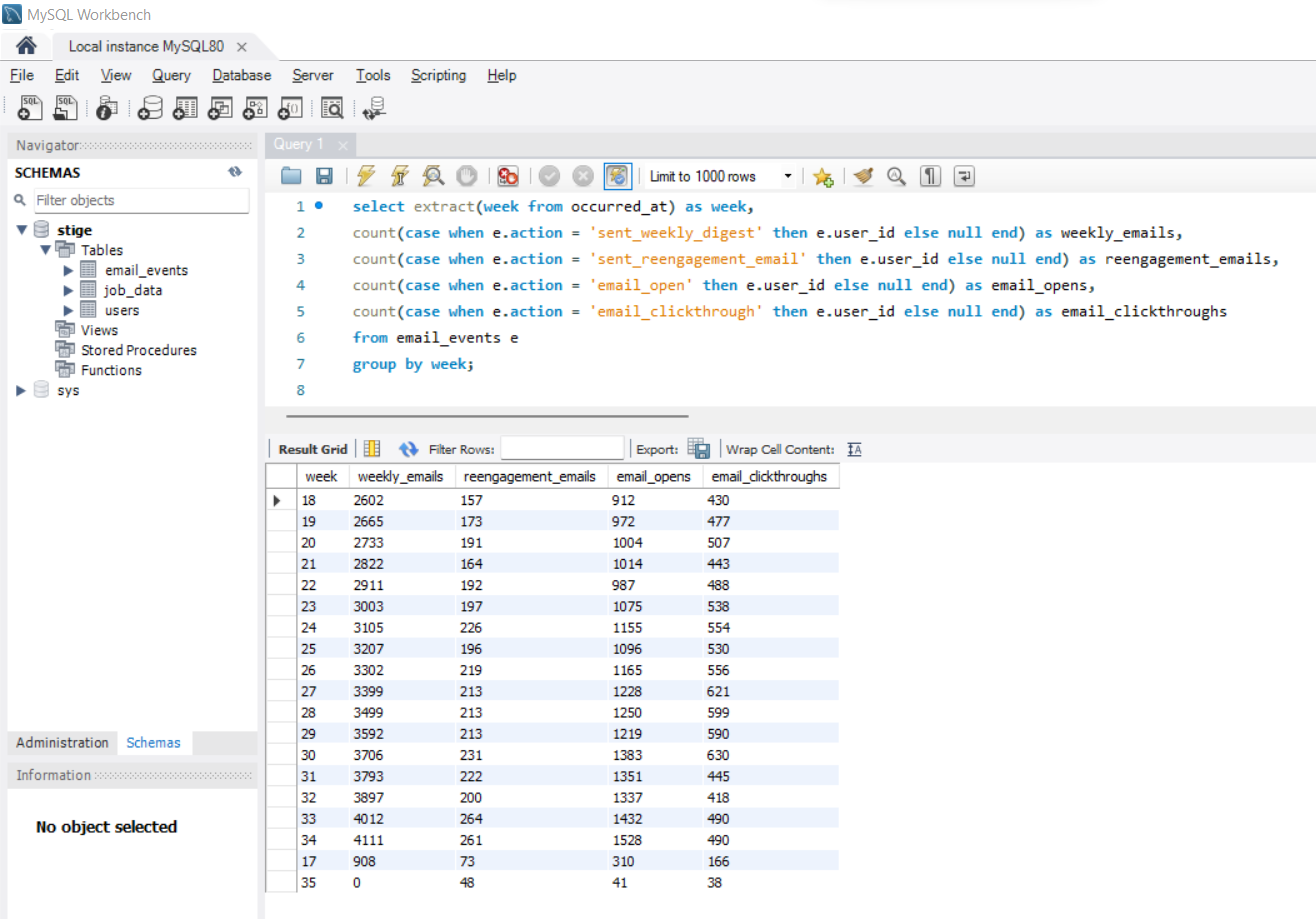
count(case when e.action = 'email\_open' then e.user\_id else null end) as email\_opens,

count(case when e.action = 'email\_clickthrough' then e.user\_id else null end) as email\_clickthroughs

from email\_events e

group by week;

In this query, we have used the email\_events table, for finding out the email engagement metrics. We have counted the users who have responded to different email actions like opened the email, clicked the email, sent the reengagement mail and sent the weekly digest and these are grouped on a weekly basis.



**RESULTS:**

* Working on this SQL project, has helped me understand the very new and complex, advanced functions and SQL writing techniques.
* I did trial and error many times with new functions like Windows functions, and writing multiple queries within a single query.
* This project has helped me improve my SQL querying skills to the next level from earlier when I just was able to work with basic SQL clauses.