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?

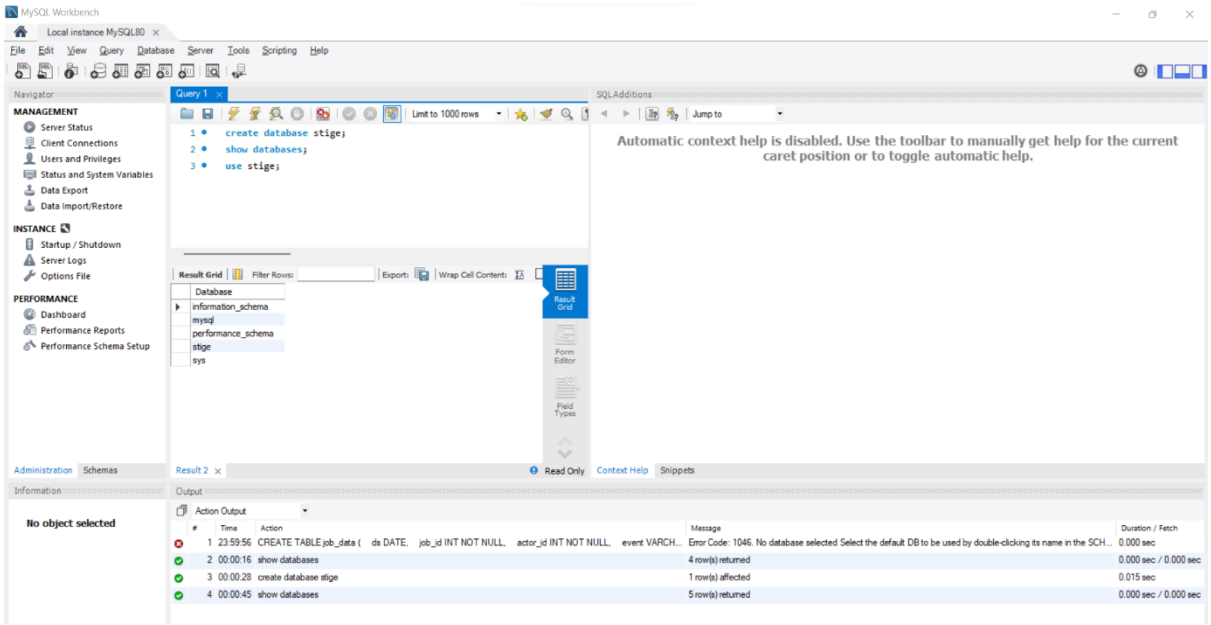
QB: Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

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?

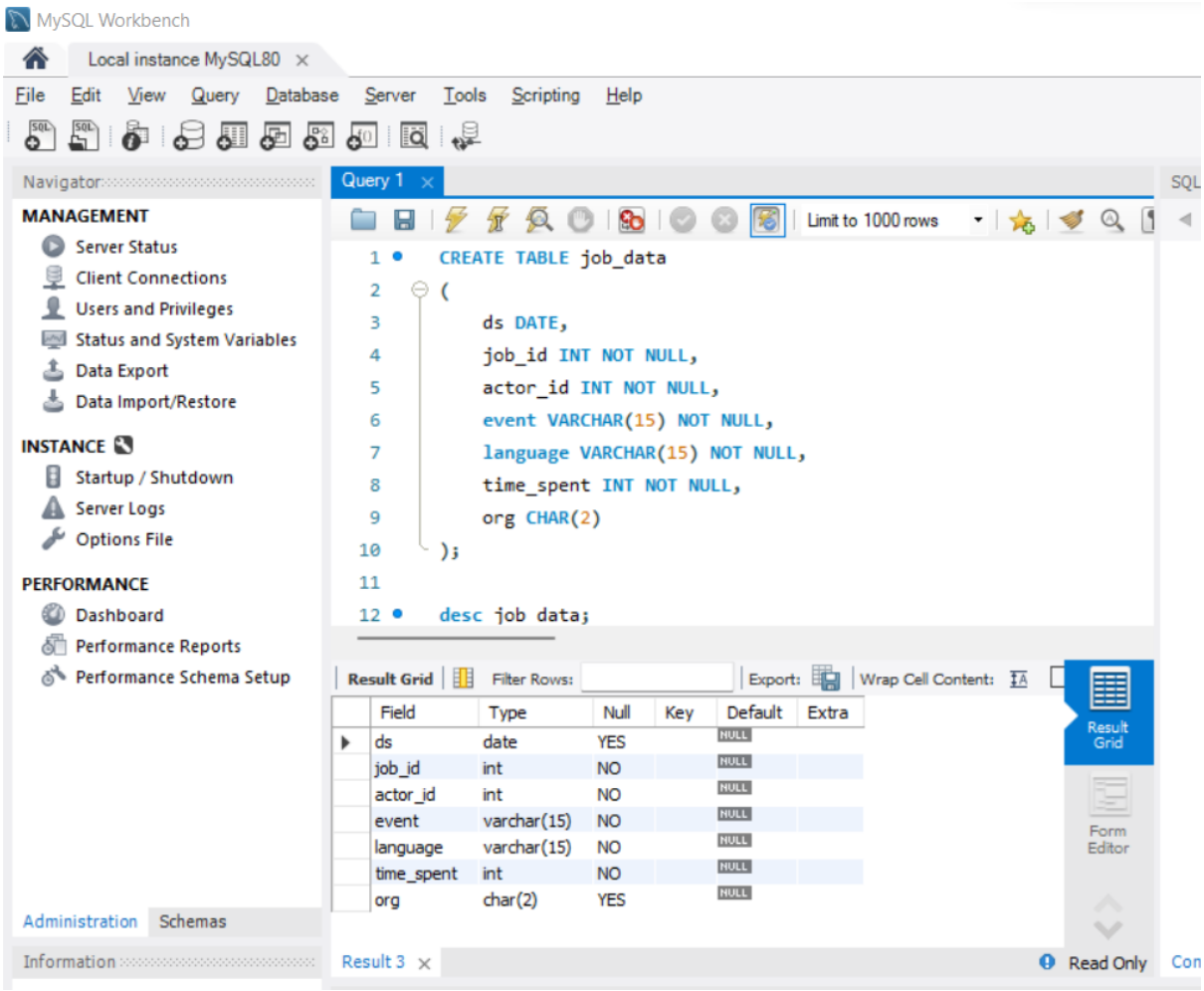
APPROACH:

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');
```

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas

Information

Query 1

Limit to 1000 rows

```
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');

select * from job_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ds	job_id	actor_id	event	language	time_spent	org
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

job_data 4

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.

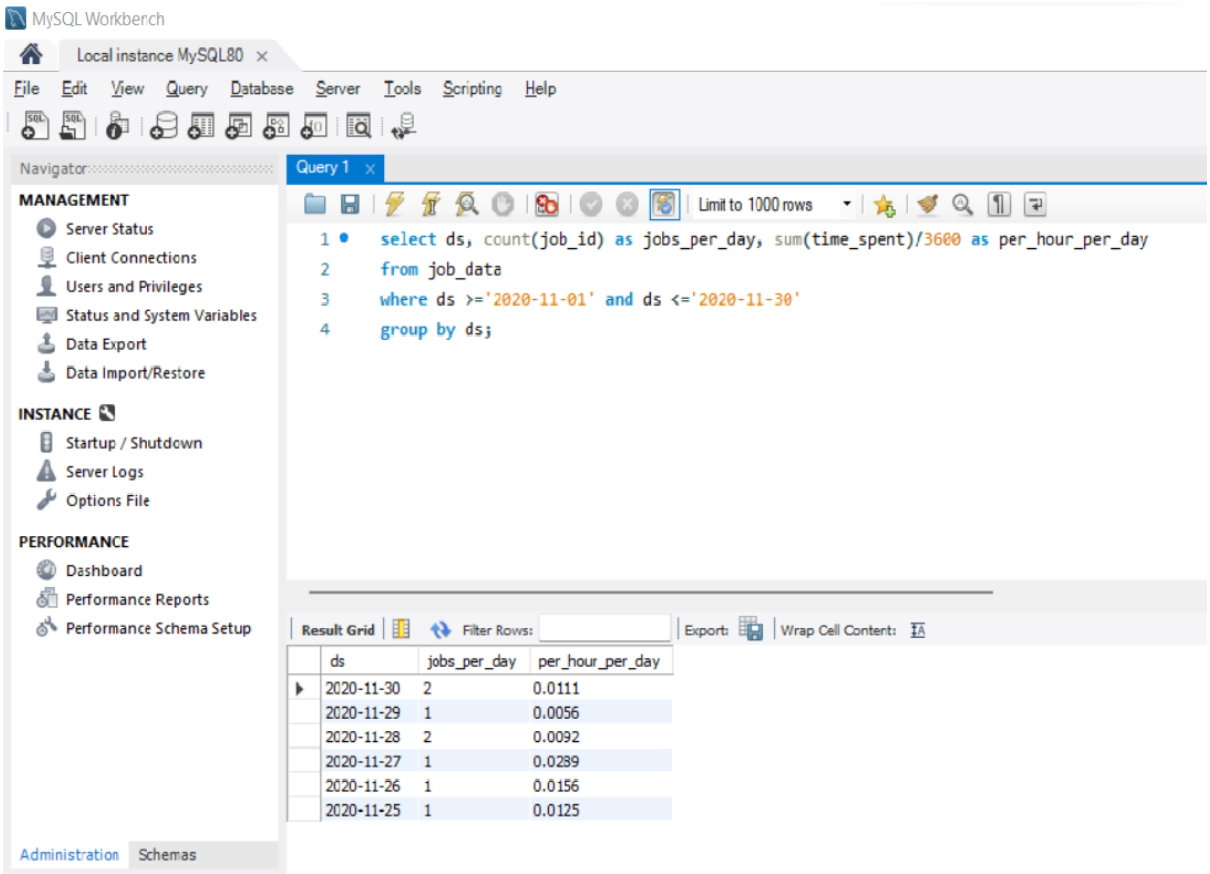
Since we have the case study in SQL, I chose this technology and application.

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;
```

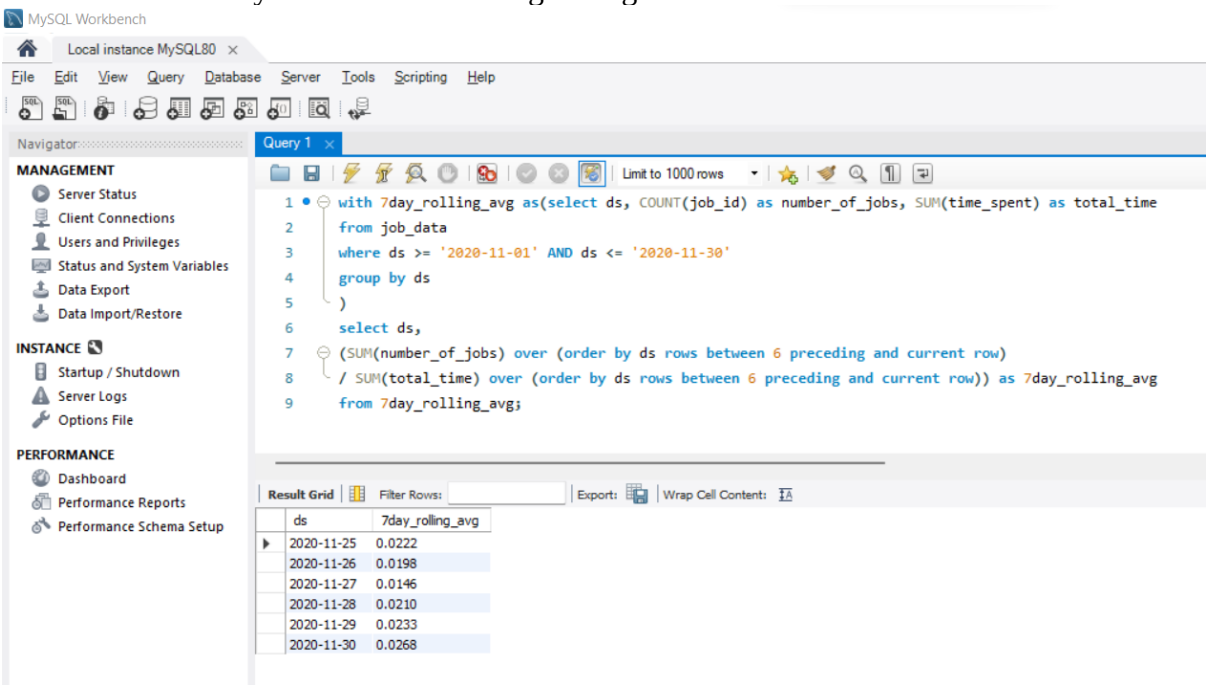
In this query, we have calculated the number of jobs grouped by their dates and the time spent per hour per day based on the time in seconds / 3600.



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;
```

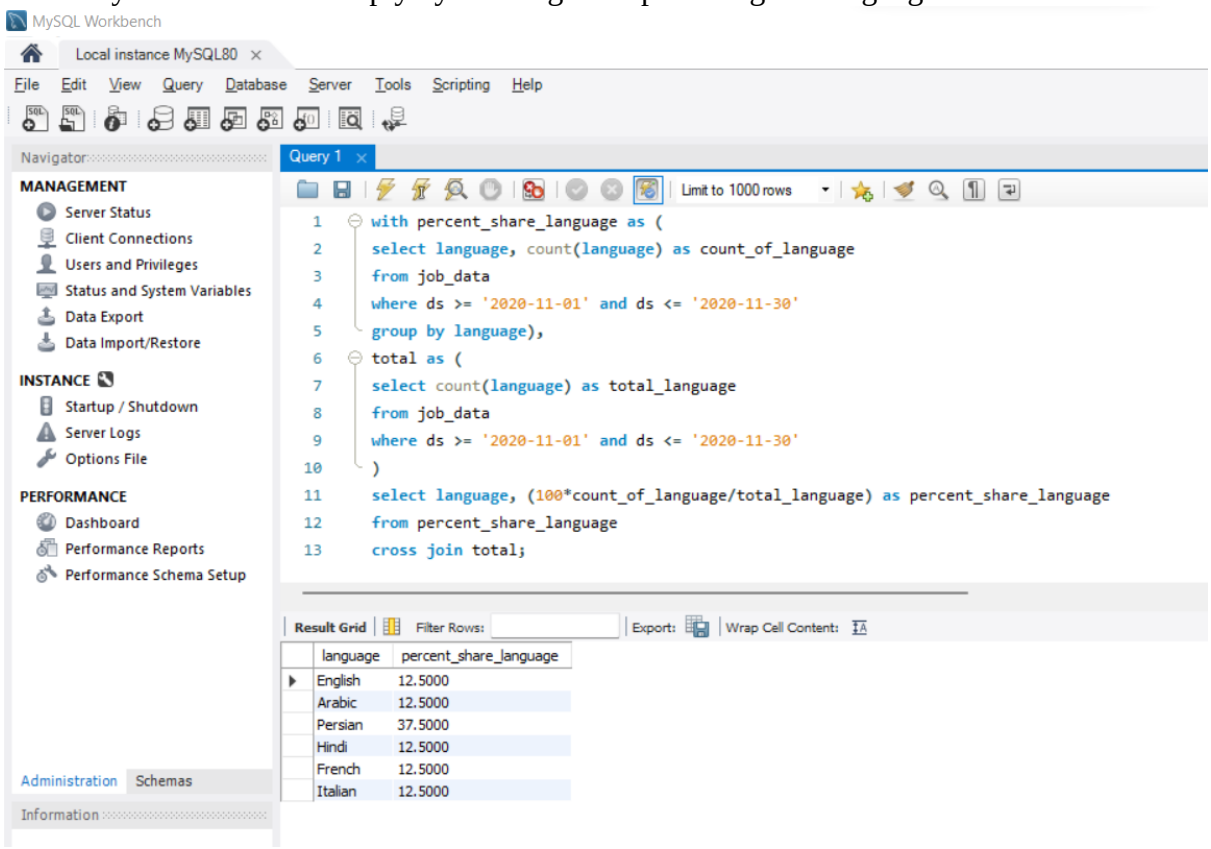
In this query, we have mixed 2 queries. Firstly we find out the number of jobs and the total time of the jobs grouped by date. Using this output as the input table for the second query, we have found out the cumulative average using the 6 rows above the current row and the current row to find the 7 day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

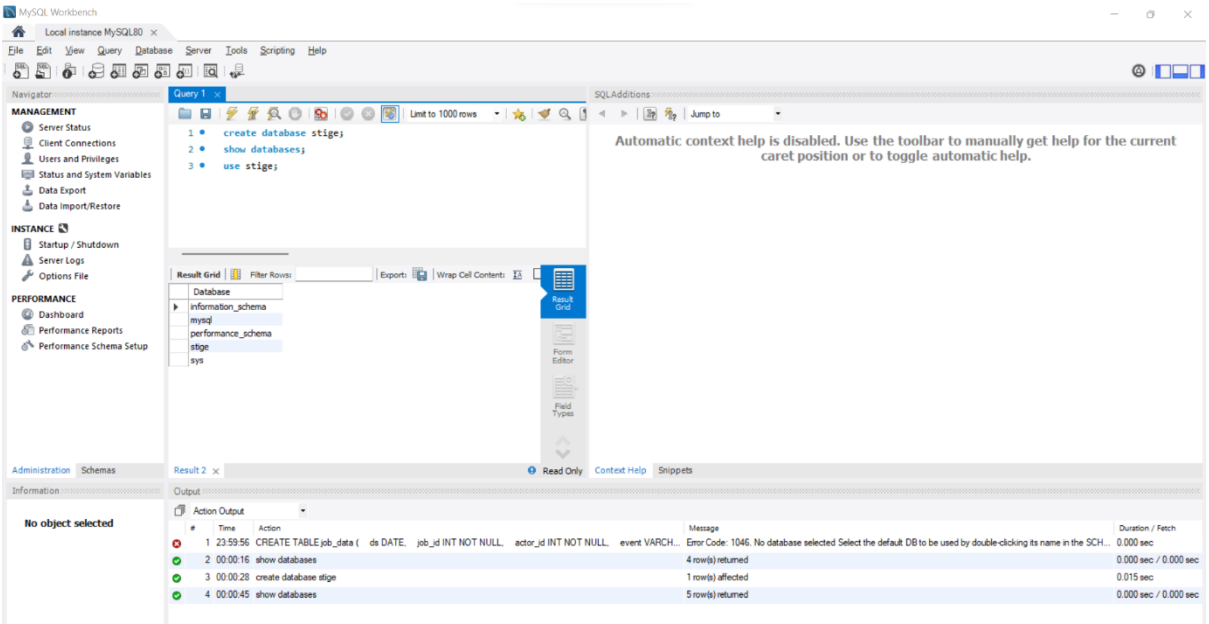
QC: Calculate the weekly retention of users-sign up cohort?

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.

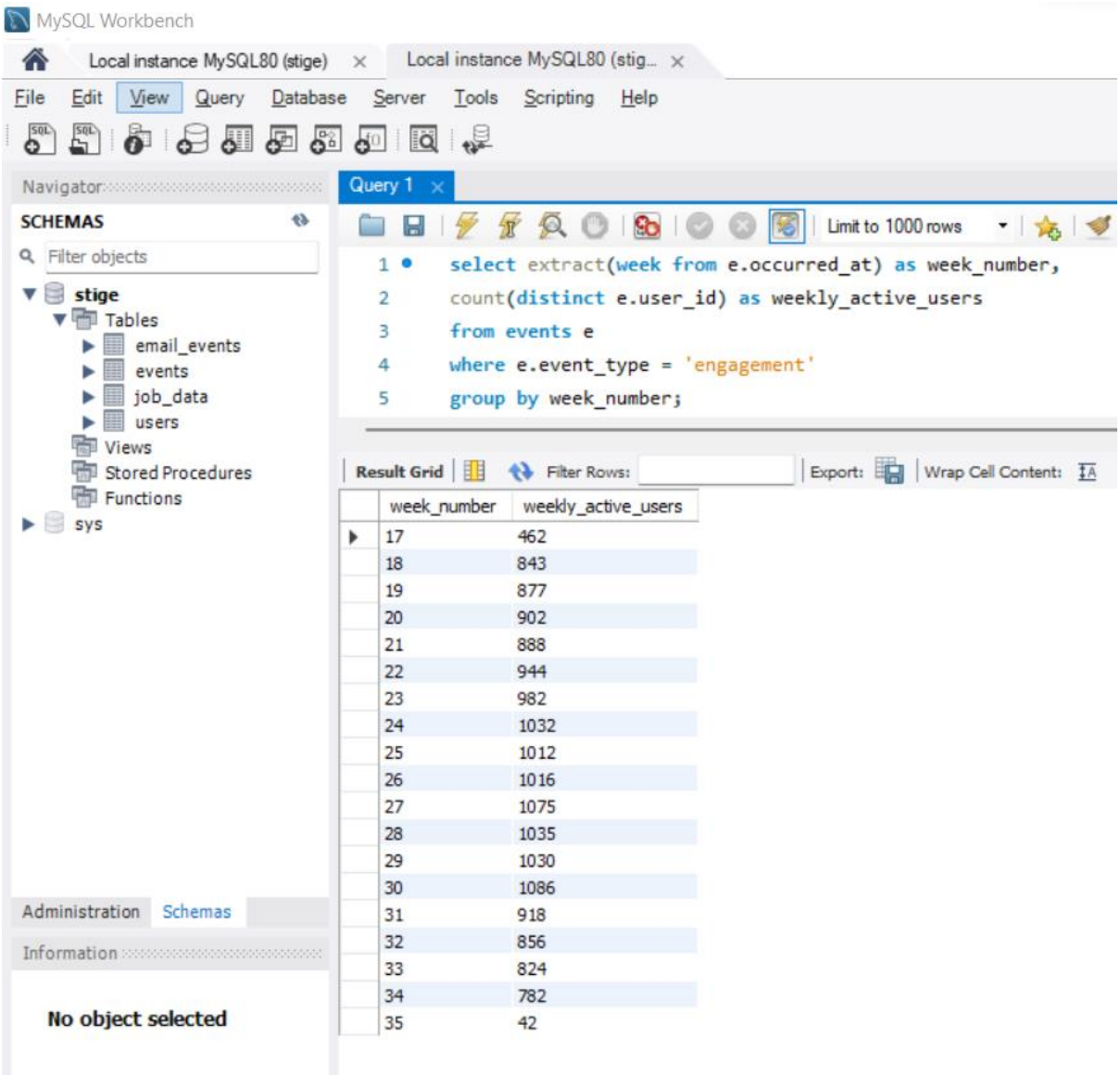
Since we have the case study in SQL, I chose this technology and application.

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;
```

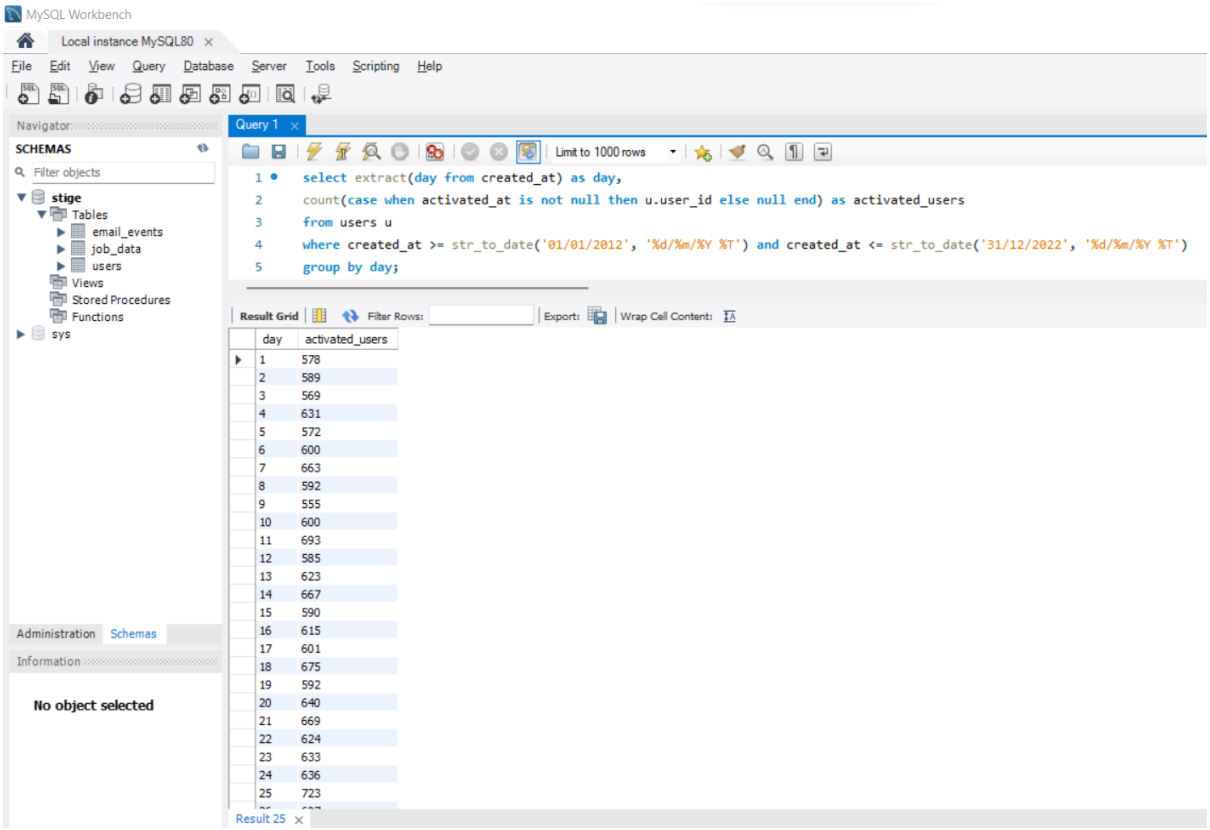
In this query, we have used the extract() function to find the week number of the occurred_at date in the events table. Then using the user_id column in the events table we have found the distinct users for whom the event_type is engagement and the results are grouped by the week number.



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.



day	activated_users
1	578
2	589
3	569
4	631
5	572
6	600
7	663
8	592
9	555
10	600
11	693
12	585
13	623
14	667
15	590
16	615
17	601
18	675
19	592
20	640
21	669
22	624
23	633
24	636
25	723

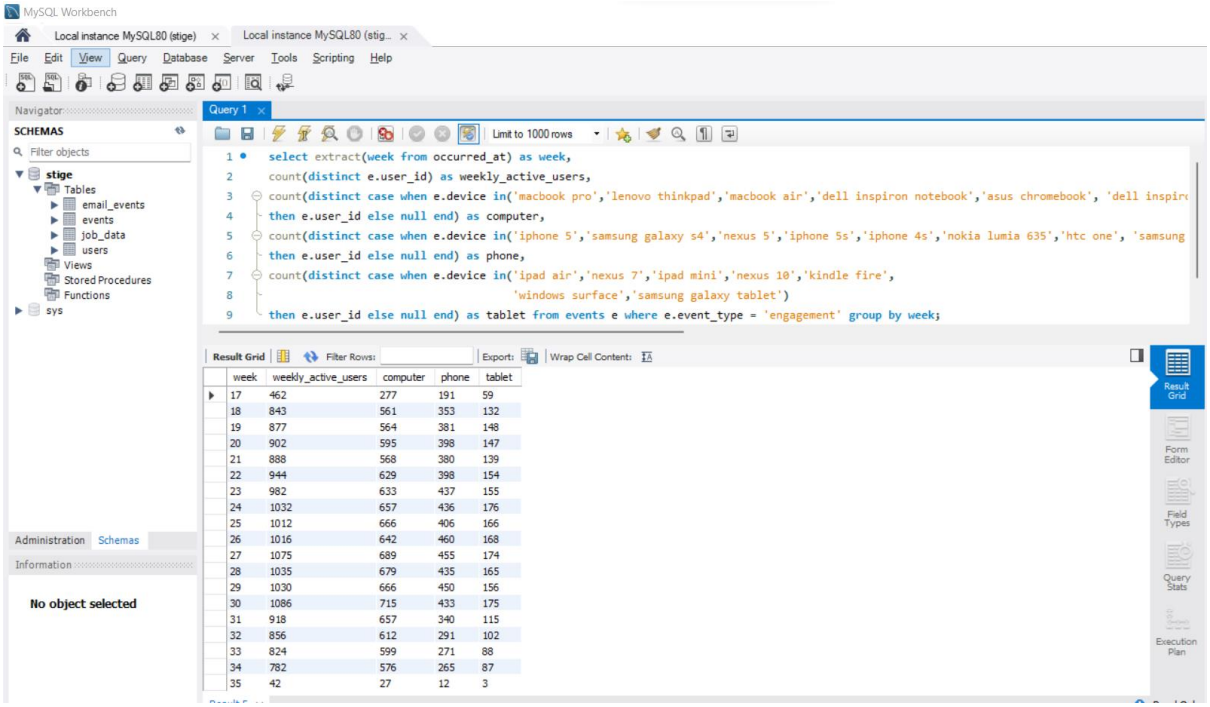
QC) Calculate the weekly retention of users-sign up cohort?

```
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'
```

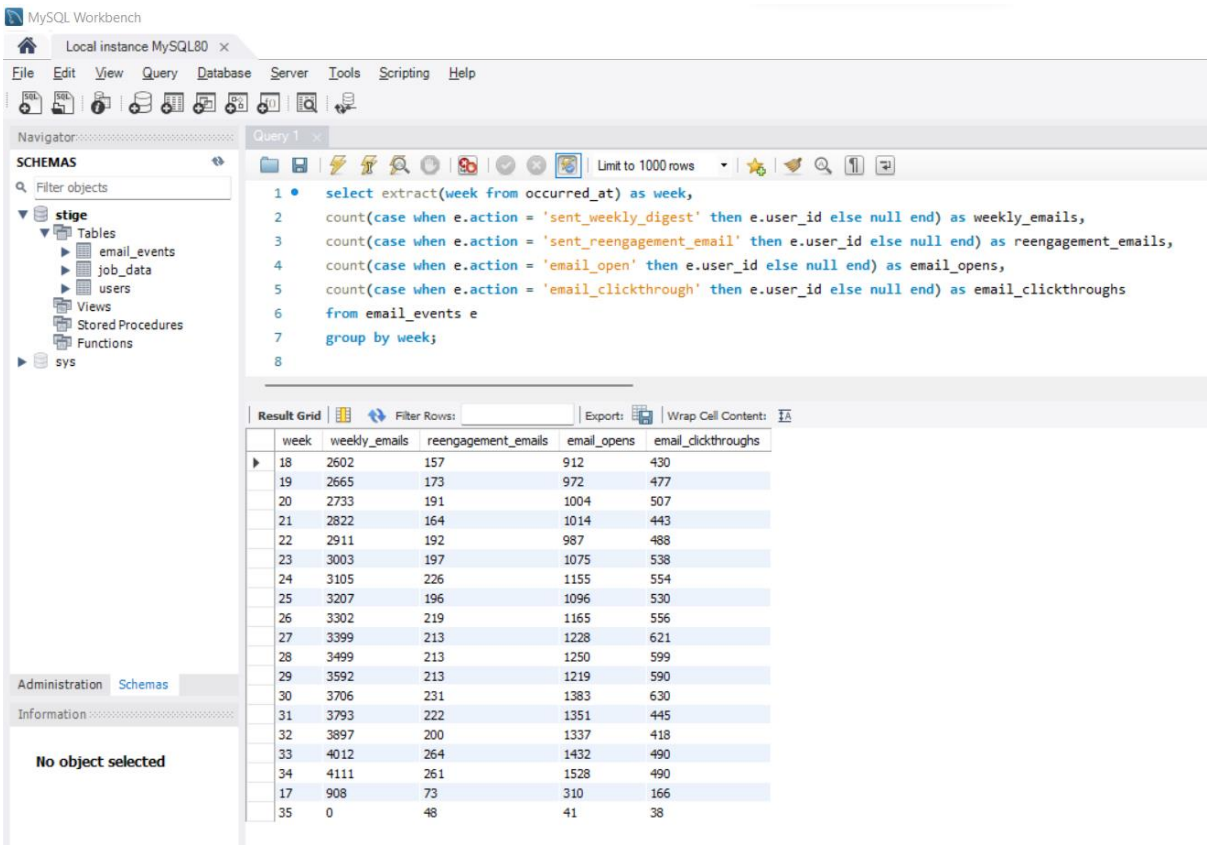
In this query, we have calculated the weekly active users and their distribution based on the devices we have and we have applied the case clause to check if the device belongs either to the computer, phone or tablet category. The results are grouped by week.



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.



week	weekly_emails	reengagement_emails	email_opens	email_clickthroughs
18	2602	157	912	430
19	2665	173	972	477
20	2733	191	1004	507
21	2822	164	1014	443
22	2911	192	987	488
23	3003	197	1075	538
24	3105	226	1155	554
25	3207	196	1096	530
26	3302	219	1165	556
27	3399	213	1228	621
28	3499	213	1250	599
29	3592	213	1219	590
30	3706	231	1383	630
31	3793	222	1351	445
32	3897	200	1337	418
33	4012	264	1432	490
34	4111	261	1528	490
17	908	73	310	166
35	0	48	41	38

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.