Project 3- Operation Analytics and Investigating Metric Spike

Project description:

Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect. Investigating metric spike is also an important part of operation analytics as being a Data Analyst you must be able to understand or make other teams understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that its very important to investigate metric spike.

In this project I am working as data analyst lead and am required to answer certain questions pertaining to operation analytics. I will be preparing datasets and then study and run queries on it to come up with answers.

Approach:

My approach towards the completion of this project is simple. At first I will complete the dataset on excel and save it as a csv file and then use MySQL workbench and import this file as dataset where I will attend every question and run queries on the data and come up with answers and insights.

Tech stack used:

For the project, I used:

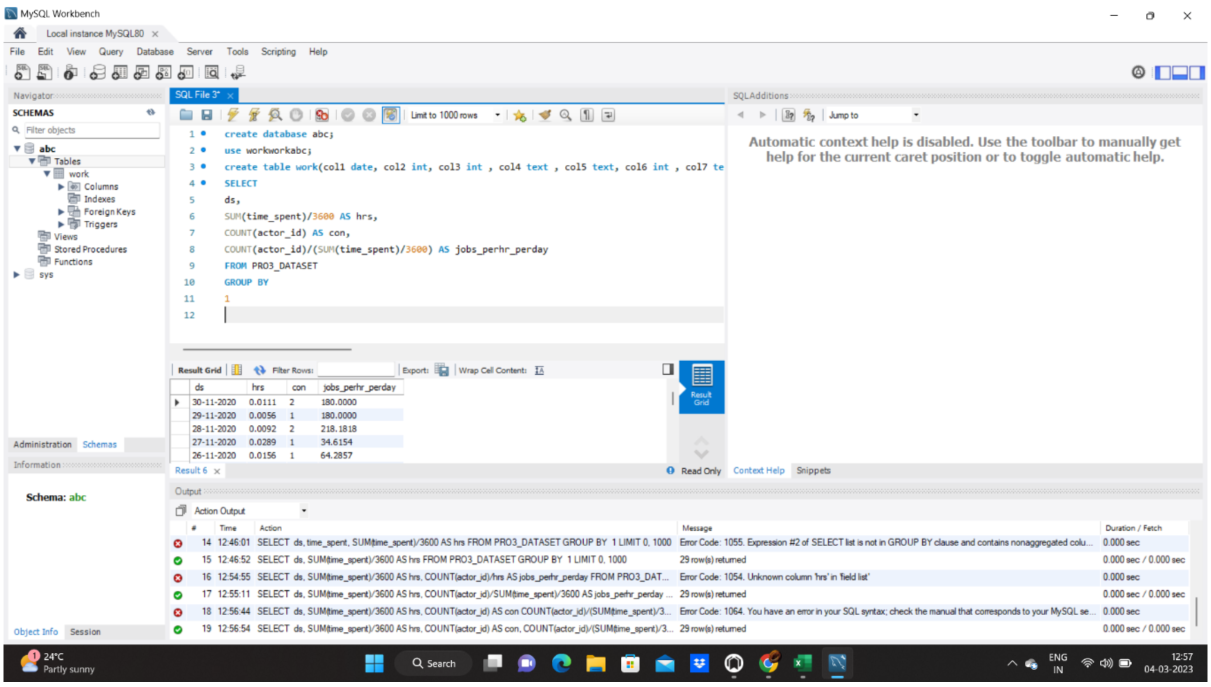
- MySQL Workbench 8.0 CE
- Excel

Insights:

Case Study 1 (Job Data)

A. **Number of jobs reviewed:** Number of jobs reviewed over time.
Your task: Calculate the number of jobs reviewed per hour per day for November 2020?

Query:



```
create database abc;
use workworkabc;
create table work(col1 date, col2 int, col3 int , col4 text , col5 text, col6 int , col7 te
SELECT
ds,
SUM(time_spent)/3600 AS hrs,
COUNT(actor_id) AS con,
COUNT(actor_id)/(SUM(time_spent)/3600) AS jobs_perhr_perday
FROM PRO3_DATASET
GROUP BY
1
```

ds	hrs	con	jobs_perhr_perday
30-11-2020	0.0111	2	180.0000
29-11-2020	0.0056	1	180.0000
28-11-2020	0.0092	2	218.1818
27-11-2020	0.0289	1	34.6154
26-11-2020	0.0156	1	64.2857

```
#   Time      Action                                                                                       Message                                                                                  Duration / Fetch
14  12:46:01  SELECT ds, time_spent, SUM(time_spent)/3600 AS hrs FROM PRO3_DATASET GROUP BY  1 LIMIT 0, 1000  Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated colu...  0.000 sec
15  12:46:52  SELECT ds, SUM(time_spent)/3600 AS hrs FROM PRO3_DATASET GROUP BY  1 LIMIT 0, 1000              29 row(s) returned                                                                       0.000 sec / 0.000 sec
16  12:54:55  SELECT ds, SUM(time_spent)/3600 AS hrs, COUNT(actor_id)/hrs AS jobs_perhr_perday FROM PRO3_DAT...  Error Code: 1054. Unknown column 'hrs' in 'field list'                                   0.000 sec
17  12:55:11  SELECT ds, SUM(time_spent)/3600 AS hrs, COUNT(actor_id)/SUM(time_spent)/3600 AS jobs_perhr_perday ...  29 row(s) returned                                                                0.000 sec / 0.000 sec
18  12:56:44  SELECT ds, SUM(time_spent)/3600 AS hrs, COUNT(actor_id) AS con COUNT(actor_id)/(SUM(time_spent)/3...  Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...  0.000 sec
19  12:56:54  SELECT ds, SUM(time_spent)/3600 AS hrs, COUNT(actor_id) AS con, COUNT(actor_id)/(SUM(time_spent)/3...  29 row(s) returned                                                               0.000 sec / 0.000 sec
```

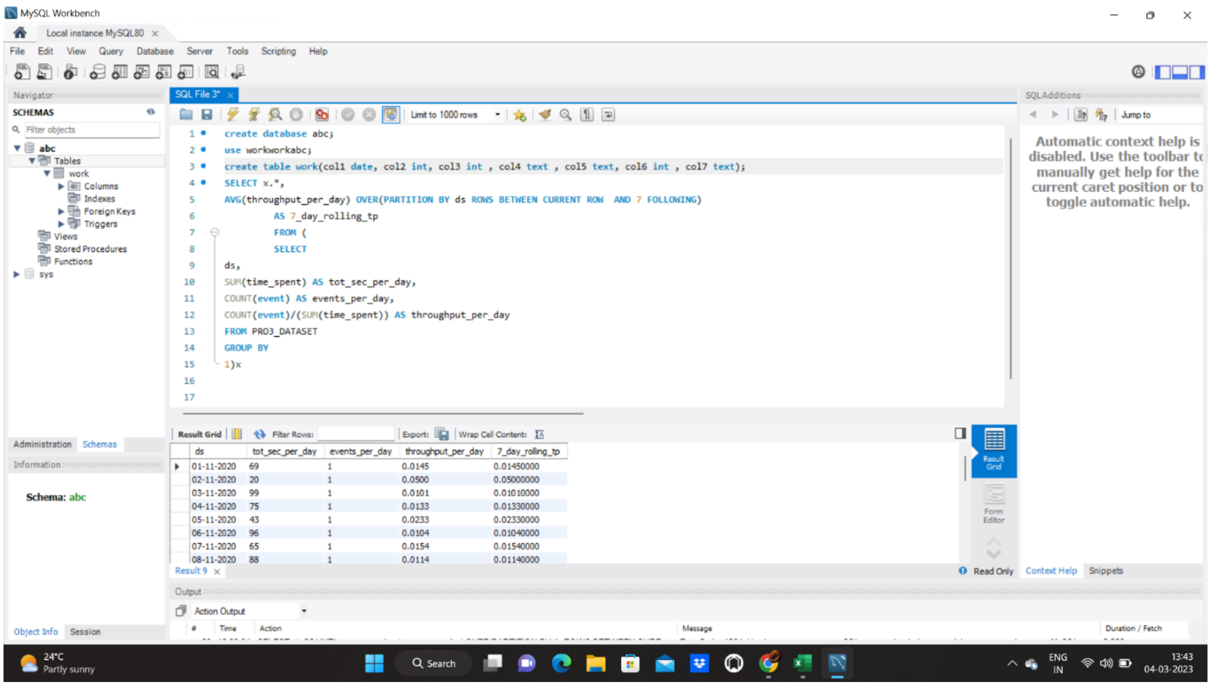
According to the results, the number of jobs reviewed per hour per day is very high since it took only certain seconds to review a single job and not many jobs were reviewed on a single date i.e. the rate is high but not the work.

B. **Throughput:** It is the no. of events happening per second.
Your task: 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?

Query:

```
SELECT x.*,
AVG(throughput_per_day) OVER(PARTITION BY ds ROWS
BETWEEN CURRENT ROW  AND 7 FOLLOWING)
        AS 7_day_rolling_tp
        FROM (
        SELECT
ds,
SUM(time_spent) AS tot_sec_per_day,
```

COUNT(event) AS events_per_day,
COUNT(event)/(SUM(time_spent)) AS throughput_per_day
FROM PRO3_DATASET
GROUP BY
1)x



```
create database abc;
use workworkabc;
create table work(col1 date, col2 int, col3 int , col4 text , col5 text, col6 int , col7 text);
SELECT x.*,
AVG(throughput_per_day) OVER(PARTITION BY ds ROWS BETWEEN CURRENT ROW  AND 7 FOLLOWING)
        AS 7_day_rolling_tp
        FROM (
        SELECT
ds,
SUM(time_spent) AS tot_sec_per_day,
COUNT(event) AS events_per_day,
COUNT(event)/(SUM(time_spent)) AS throughput_per_day
FROM PRO3_DATASET
GROUP BY
1)x
```

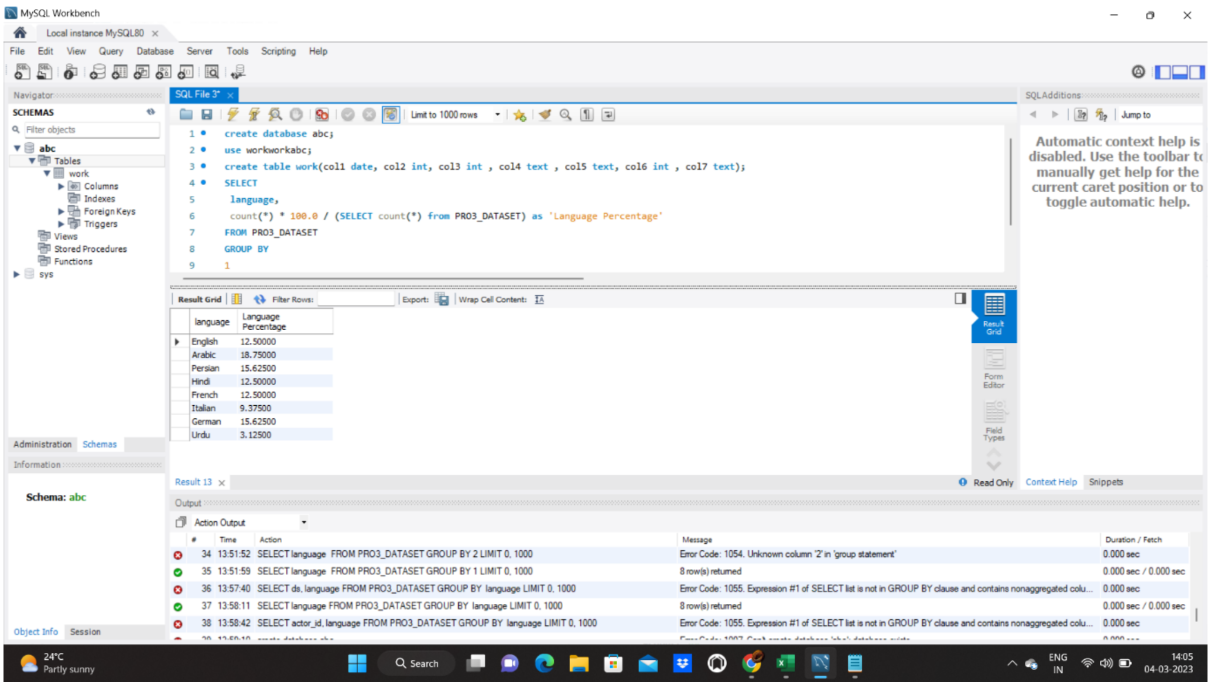
ds	tot_sec_per_day	events_per_day	throughput_per_day	7_day_rolling_tp
01-11-2020	69	1	0.0145	0.01450000
02-11-2020	20	1	0.0500	0.05000000
03-11-2020	99	1	0.0101	0.01010000
04-11-2020	75	1	0.0133	0.01330000
05-11-2020	43	1	0.0233	0.02330000
06-11-2020	96	1	0.0104	0.01040000
07-11-2020	65	1	0.0154	0.01540000
08-11-2020	88	1	0.0114	0.01140000

The table shows both the daily throughput and the 7 day rolling throughput. Not saying for this particular data in hand because both are quite small and doesn’t make that much difference but 7 day rolling throughput is better than the daily because calculating and analysing rates on a daily basis doesn’t makes sense not much difference occurs in a single day. It is better to derive insights on a substantial amount of data.

C. **Percentage share of each language:** Share of each language for different contents.
Your task: Calculate the percentage share of each language in the last 30 days?

Query:
SELECT
 language,
 count(*) * 100.0 / (SELECT count(*) from PRO3_DATASET) as 'Language Percentage'
FROM PRO3_DATASET
GROUP BY

1



```
create database abc;
use workworkabc;
create table work(col1 date, col2 int, col3 int , col4 text , col5 text, col6 int , col7 text);
SELECT
 language,
 count(*) * 100.0 / (SELECT count(*) from PRO3_DATASET) as 'Language Percentage'
FROM PRO3_DATASET
GROUP BY
1
```

	language	Language Percentage
▶	English	12.50000
	Arabic	18.75000
	Persian	15.62500
	Hindi	12.50000
	French	12.50000
	Italian	9.37500
	German	15.62500
	Urdu	3.12500

```
#   Time      Action                                                            Message                                                                              Duration / Fetch
34  13:51:52  SELECT language  FROM PRO3_DATASET GROUP BY 2 LIMIT 0, 1000        Error Code: 1054. Unknown column '2' in 'group statement'                            0.000 sec
35  13:51:59  SELECT language  FROM PRO3_DATASET GROUP BY 1 LIMIT 0, 1000        8 row(s) returned                                                                    0.000 sec / 0.000 sec
36  13:57:40  SELECT ds, language FROM PRO3_DATASET GROUP BY  language LIMIT 0, 1000   Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated colu...   0.000 sec
37  13:58:11  SELECT language FROM PRO3_DATASET GROUP BY  language LIMIT 0, 1000  8 row(s) returned                                                                    0.000 sec / 0.000 sec
38  13:58:42  SELECT actor_id, language FROM PRO3_DATASET GROUP BY  language LIMIT 0, 1000   Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated colu...   0.000 sec
```

The results shows each language and the percentage of each language used in the month of November. The result is pretty straight forward.

D. **Duplicate rows:** Rows that have the same value present in them.
Your task: Let’s say you see some duplicate rows in the data. How will you display duplicates from the table?

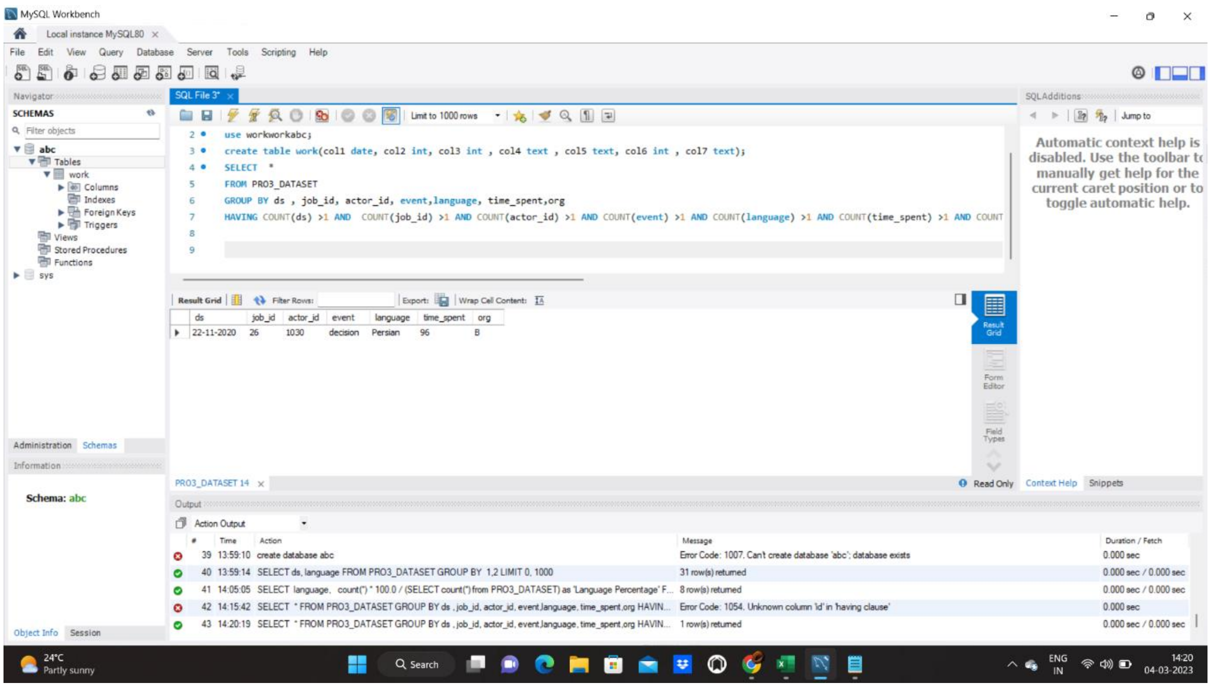
Query:

SELECT *

FROM PRO3_DATASET

GROUP BY ds , job_id, actor_id, event,language, time_spent,org

HAVING COUNT(ds) >1 AND COUNT(job_id) >1 AND COUNT(actor_id) >1 AND COUNT(event) >1 AND COUNT(language) >1 AND COUNT(time_spent) >1 AND COUNT(org) >1



```
use workworkabc;
create table work(col1 date, col2 int, col3 int , col4 text , col5 text, col6 int , col7 text);
SELECT  *
FROM PRO3_DATASET
GROUP BY ds , job_id, actor_id, event,language, time_spent,org
HAVING COUNT(ds) >1 AND  COUNT(job_id) >1 AND COUNT(actor_id) >1 AND COUNT(event) >1 AND COUNT(language) >1 AND COUNT(time_spent) >1 AND COUNT
```

	ds	job_id	actor_id	event	language	time_spent	org
▶	22-11-2020	26	1030	decision	Persian	96	B

```
#   Time      Action                                                                              Message                                                         Duration / Fetch
39  13:59:10  create database abc                                                                 Error Code: 1007. Can't create database 'abc'; database exists  0.000 sec
40  13:59:14  SELECT ds, language FROM PRO3_DATASET GROUP BY 1,2 LIMIT 0, 1000                    31 row(s) returned                                              0.000 sec / 0.000 sec
41  14:05:05  SELECT language, count(*) * 100.0 / (SELECT count(*) from PRO3_DATASET) as 'Language Percentage' F...  8 row(s) returned                      0.000 sec / 0.000 sec
42  14:15:42  SELECT * FROM PRO3_DATASET GROUP BY ds , job_id, actor_id, event,language, time_spent,org HAVIN...  Error Code: 1054. Unknown column 'id' in 'having clause'  0.000 sec
43  14:20:19  SELECT * FROM PRO3_DATASET GROUP BY ds , job_id, actor_id, event,language, time_spent,org HAVIN...  1 row(s) returned                       0.000 sec / 0.000 sec
```

The result showed there were two rows present in the dataset that were duplicates of each other that is both had every value of every column as the same.

Case Study 2 (Investigating metric spike)

A. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.
Your task: Calculate the weekly user engagement?

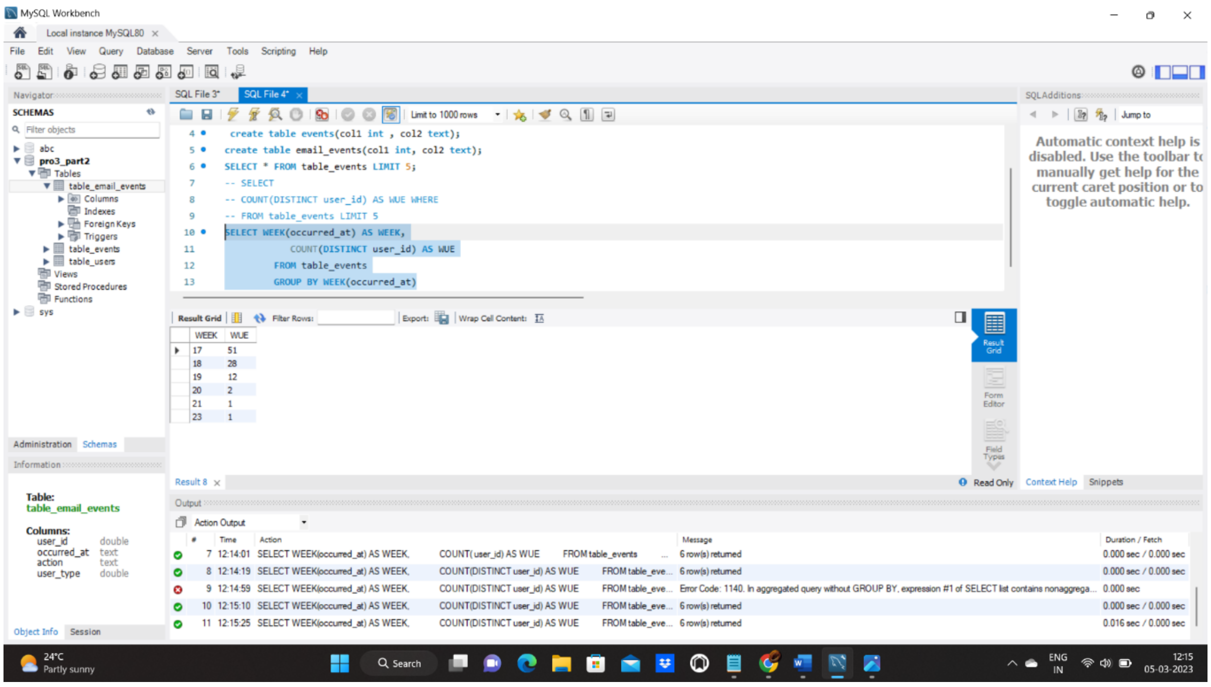
Query:

```
SELECT WEEK(occurred_at) AS WEEK,

          COUNT(DISTINCT user_id) AS WUE

       FROM table_events

       GROUP BY WEEK(occurred_at)
```



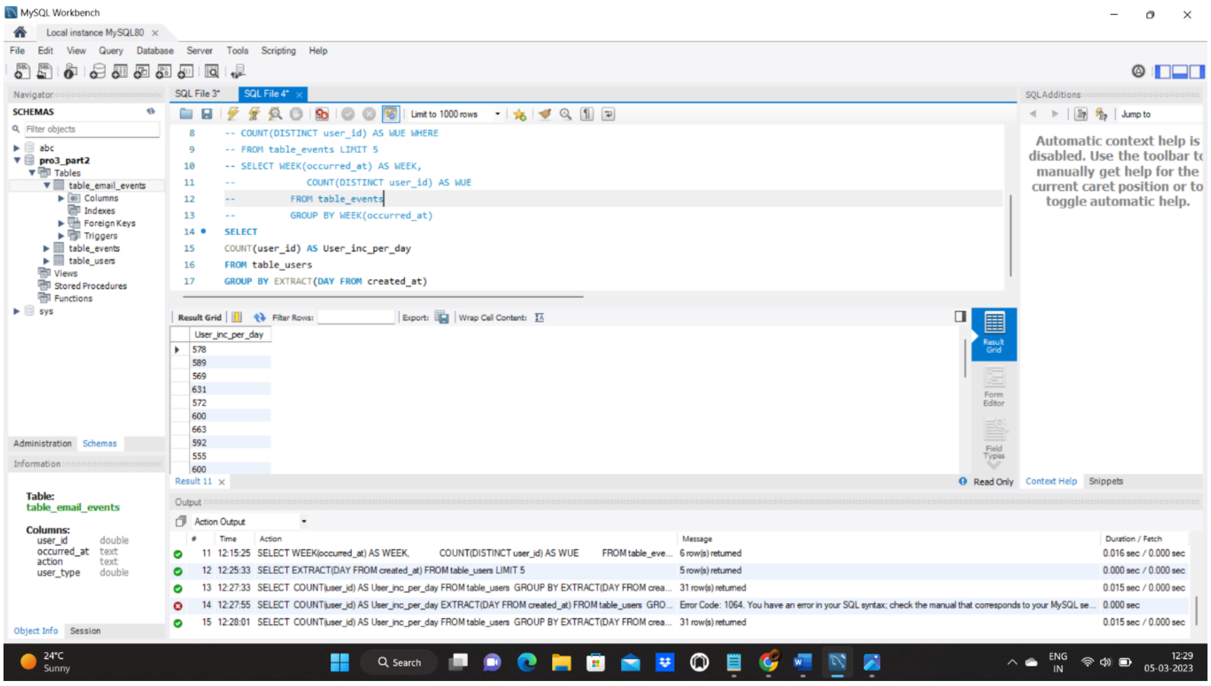
```
 create table events(col1 int , col2 text);
create table email_events(col1 int, col2 text);
SELECT * FROM table_events LIMIT 5;
-- SELECT
-- COUNT(DISTINCT user_id) AS WUE WHERE
-- FROM table_events LIMIT 5
SELECT WEEK(occurred_at) AS WEEK,
            COUNT(DISTINCT user_id) AS WUE
        FROM table_events
        GROUP BY WEEK(occurred_at)
```

WEEK	WUE
17	51
18	28
19	12
20	2
21	1
23	1

```
#   Time      Action                                                                          Message                                                                                    Duration / Fetch
7   12:14:01  SELECT WEEK(occurred_at) AS WEEK,   COUNT( user_id) AS WUE   FROM table_events  ...  6 row(s) returned                                                                 0.000 sec / 0.000 sec
8   12:14:19  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE   FROM table_eve...  6 row(s) returned                                                              0.000 sec / 0.000 sec
9   12:14:59  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE   FROM table_eve...  Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggrega...  0.000 sec
10  12:15:10  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE   FROM table_eve...  6 row(s) returned                                                              0.000 sec / 0.000 sec
11  12:15:25  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE   FROM table_eve...  6 row(s) returned                                                              0.016 sec / 0.000 sec
```

B. **User Growth:** Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

Query:
SELECT
COUNT(user_id) AS User_inc_per_day
FROM table_users
GROUP BY EXTRACT(DAY FROM created_at)



```
-- COUNT(DISTINCT user_id) AS WUE WHERE
-- FROM table_events LIMIT 5
-- SELECT WEEK(occurred_at) AS WEEK,
--              COUNT(DISTINCT user_id) AS WUE
--          FROM table_events
--          GROUP BY WEEK(occurred_at)
SELECT
COUNT(user_id) AS User_inc_per_day
FROM table_users
GROUP BY EXTRACT(DAY FROM created_at)
```

User_inc_per_day
578
589
569
631
572
600
663
592
555
600

```
#   Time      Action                                                                                          Message                                                                                      Duration / Fetch
11  12:15:25  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE   FROM table_eve...              6 row(s) returned                                                                    0.016 sec / 0.000 sec
12  12:25:33  SELECT EXTRACT(DAY FROM created_at) FROM table_users LIMIT 5                                       5 row(s) returned                                                                    0.000 sec / 0.000 sec
13  12:27:33  SELECT COUNT(user_id) AS User_inc_per_day FROM table_users GROUP BY EXTRACT(DAY FROM crea...     31 row(s) returned                                                                   0.015 sec / 0.000 sec
14  12:27:55  SELECT COUNT(user_id) AS User_inc_per_day EXTRACT(DAY FROM created_at) FROM table_users GRO...  Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL se...  0.000 sec
15  12:28:01  SELECT COUNT(user_id) AS User_inc_per_day FROM table_users GROUP BY EXTRACT(DAY FROM crea...     31 row(s) returned                                                                   0.015 sec / 0.000 sec
```

The result shows the number of users that joined the product on the daily basis.

c. **Weekly Retention:** Users getting retained weekly after signing-up for a product.
Your task: Calculate the weekly retention of users-sign up cohort?

Query:

Select m.user_id,m.login_week,n.first as first,

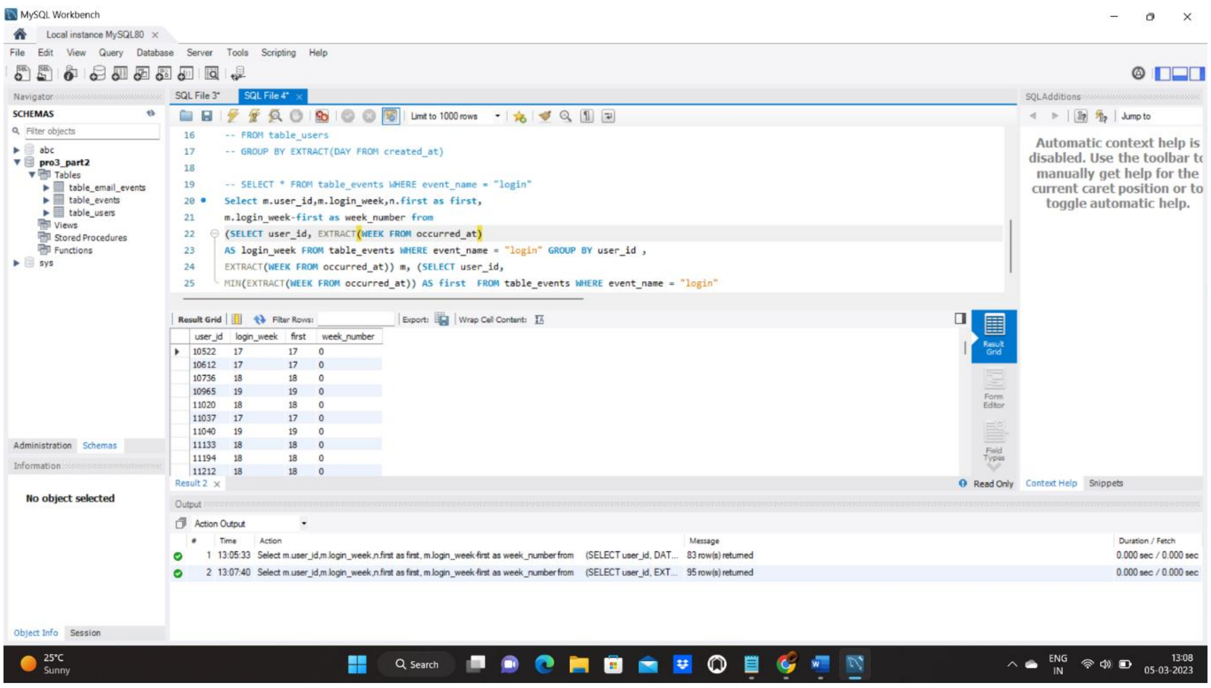
m.login_week-first as week_number from

(SELECT user_id, EXTRACT(WEEK FROM occurred_at)

AS login_week FROM table_events WHERE event_name = "login" GROUP BY user_id ,

EXTRACT(WEEK FROM occurred_at)) m, (SELECT user_id,

MIN(EXTRACT(WEEK FROM occurred_at)) AS first FROM table_events WHERE event_name = "login"



```
-- FROM table_users
-- GROUP BY EXTRACT(DAY FROM created_at)

-- SELECT * FROM table_events WHERE event_name = "login"
Select m.user_id,m.login_week,n.first as first,
m.login_week-first as week_number from
(SELECT user_id, EXTRACT(WEEK FROM occurred_at)
AS login_week FROM table_events WHERE event_name = "login" GROUP BY user_id ,
EXTRACT(WEEK FROM occurred_at)) m, (SELECT user_id,
MIN(EXTRACT(WEEK FROM occurred_at)) AS first  FROM table_events WHERE event_name = "login"
```

user_id	login_week	first	week_number
10522	17	17	0
10612	17	17	0
10736	18	18	0
10965	19	19	0
11020	18	18	0
11037	17	17	0
11040	19	19	0
11133	18	18	0
11194	18	18	0
11212	18	18	0

#	Time	Action	Message	Duration / Fetch
1	13:05:33	Select m.user_id,m.login_week,n.first as first, m.login_week-first as week_number from (SELECT user_id, DAT...	83 row(s) returned	0.000 sec / 0.000 sec
2	13:07:40	Select m.user_id,m.login_week,n.first as first, m.login_week-first as week_number from (SELECT user_id, EXT...	95 row(s) returned	0.000 sec / 0.000 sec

The results show excellent user retention and that most of the users returned to login in the same week with a very exceptions that returned the next week.

Weekly Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.

Query:

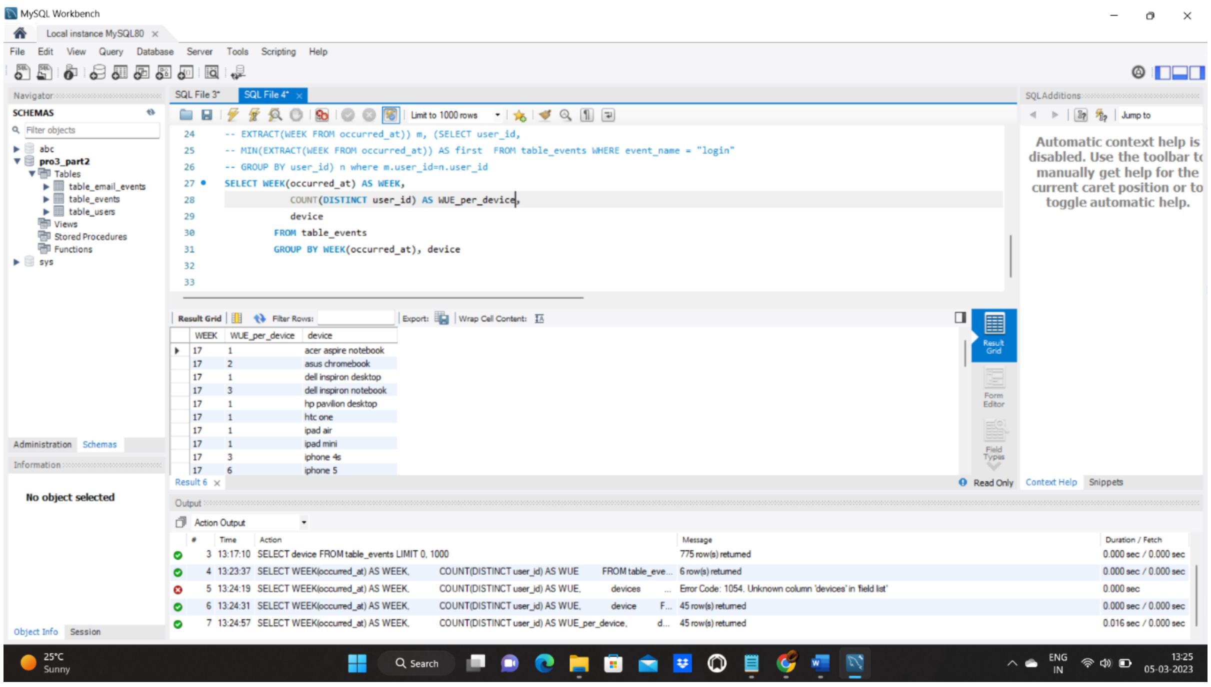
```
SELECT WEEK(occurred_at) AS WEEK,

            COUNT(DISTINCT user_id) AS WUE_per_device,

            device

        FROM table_events

        GROUP BY WEEK(occurred_at), device
```



```
-- EXTRACT(WEEK FROM occurred_at)) m, (SELECT user_id,
-- MIN(EXTRACT(WEEK FROM occurred_at)) AS first  FROM table_events WHERE event_name = "login"
-- GROUP BY user_id) n where m.user_id=n.user_id
SELECT WEEK(occurred_at) AS WEEK,
            COUNT(DISTINCT user_id) AS WUE_per_device,
            device
        FROM table_events
        GROUP BY WEEK(occurred_at), device
```

WEEK	WUE_per_device	device
17	1	acer aspire notebook
17	2	asus chromebook
17	1	dell inspiron desktop
17	3	dell inspiron notebook
17	1	hp pavilion desktop
17	1	htc one
17	1	ipad air
17	1	ipad mini
17	3	iphone 4s
17	6	iphone 5

```
#  Time      Action                                                                                 Message                                               Duration / Fetch
3  13:17:10  SELECT device FROM table_events LIMIT 0, 1000                                          775 row(s) returned                                   0.000 sec / 0.000 sec
4  13:23:37  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE    FROM table_eve... 6 row(s) returned                                     0.000 sec / 0.000 sec
5  13:24:19  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE,   devices   ...    Error Code: 1054. Unknown column 'devices' in 'field list'  0.000 sec
6  13:24:31  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE,   device    F...   45 row(s) returned                                    0.000 sec / 0.000 sec
7  13:24:57  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE_per_device,  d...   45 row(s) returned                                    0.016 sec / 0.000 sec
```

A. **Email Engagement:** Users engaging with the email service.
   Your task: Calculate the email engagement metrics?

   Query:

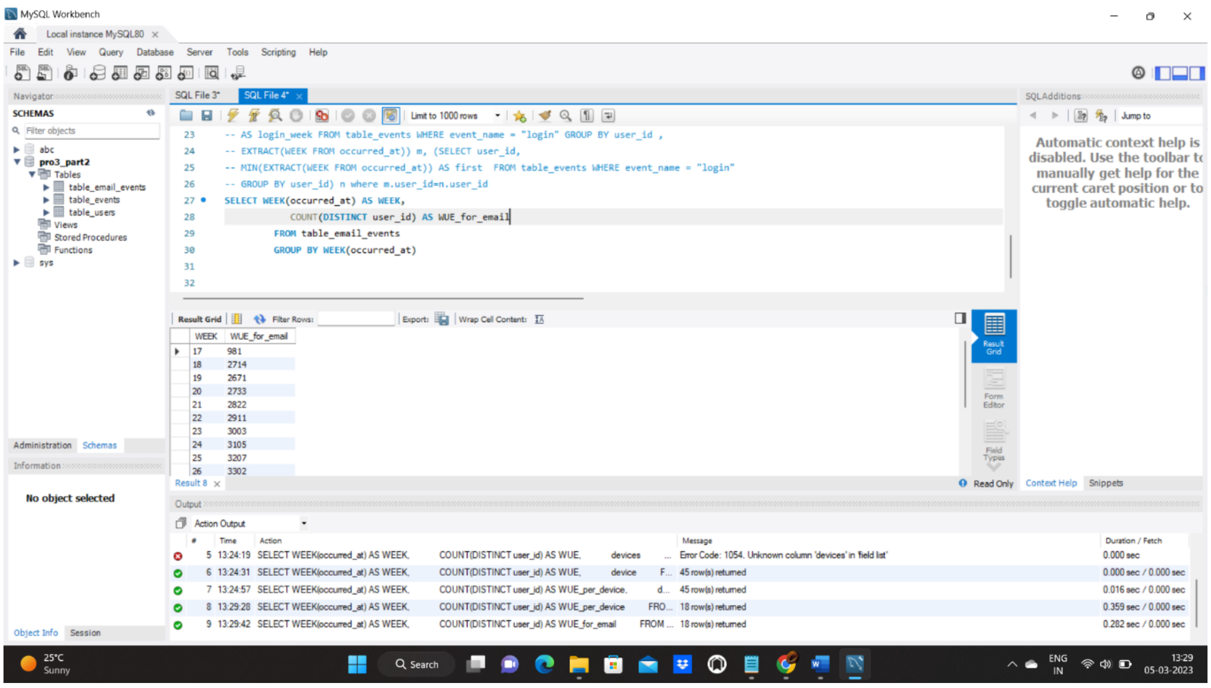
Calculating weekly user engagement for email services

SELECT WEEK(occurred_at) AS WEEK,

COUNT(DISTINCT user_id) AS WUE_for_email

FROM table_email_events

GROUP BY WEEK(occurred_at)



```
-- AS login_week FROM table_events WHERE event_name = "login" GROUP BY user_id ,
-- EXTRACT(WEEK FROM occurred_at)) m, (SELECT user_id,
-- MIN(EXTRACT(WEEK FROM occurred_at)) AS first  FROM table_events WHERE event_name = "login"
-- GROUP BY user_id) n where m.user_id=n.user_id
SELECT WEEK(occurred_at) AS WEEK,
            COUNT(DISTINCT user_id) AS WUE_for_email
        FROM table_email_events
        GROUP BY WEEK(occurred_at)
```

WEEK	WUE_for_email
17	981
18	2714
19	2671
20	2733
21	2822
22	2911
23	3003
24	3105
25	3207
26	3302

```
#  Time      Action                                                                                Message                                              Duration / Fetch
5  13:24:19  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE,   devices      ...  Error Code: 1054. Unknown column 'devices' in 'field list'  0.000 sec
6  13:24:31  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE,   device       F... 45 row(s) returned                                   0.000 sec / 0.000 sec
7  13:24:57  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE_per_device,     d... 45 row(s) returned                                   0.016 sec / 0.000 sec
8  13:29:28  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE_per_device    FRO... 18 row(s) returned                                   0.359 sec / 0.000 sec
9  13:29:42  SELECT WEEK(occurred_at) AS WEEK,   COUNT(DISTINCT user_id) AS WUE_for_email    FROM ... 18 row(s) returned                                   0.282 sec / 0.000 sec
```

We now are calculating email service retention Select m.user_id,m.login_week,n.first as first,

m.login_week-first as week_number from

(SELECT user_id, EXTRACT(WEEK FROM occurred_at)

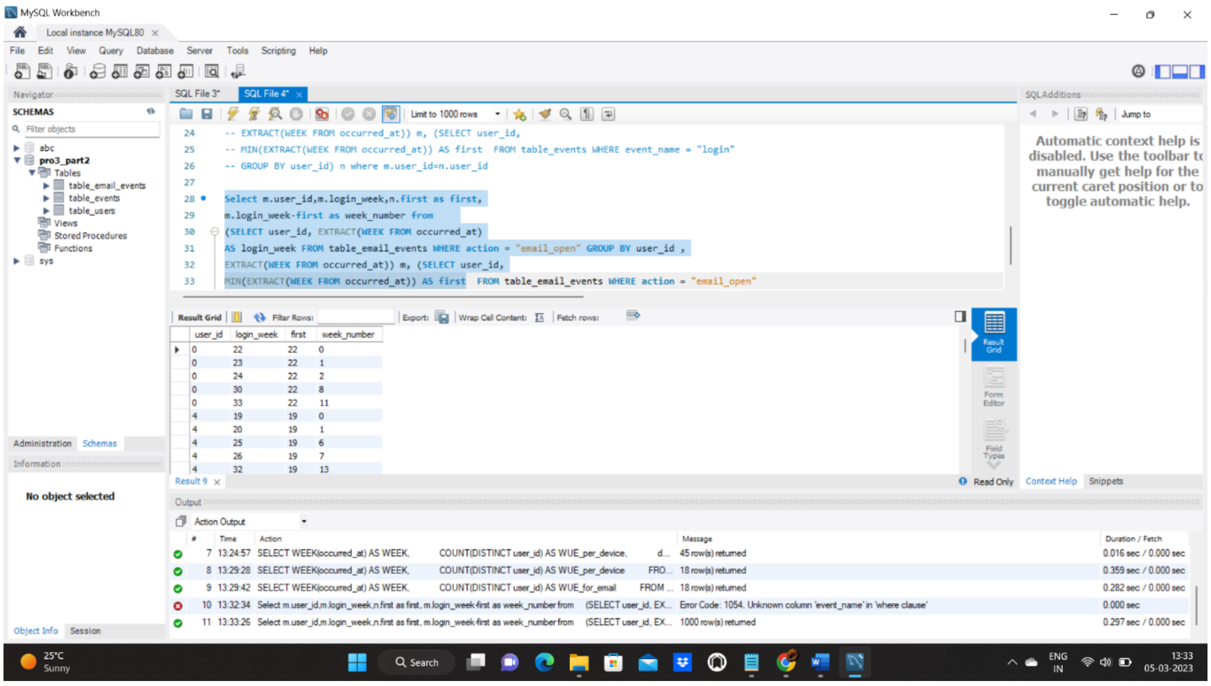
AS login_week FROM table_email_events WHERE action = "email_open" GROUP BY user_id ,

EXTRACT(WEEK FROM occurred_at)) m, (SELECT user_id,

MIN(EXTRACT(WEEK FROM occurred_at)) AS first FROM table_email_events WHERE action = "email_open"

GROUP BY user_id) n where m.user_id=n.user_id

in users.



```
-- EXTRACT(WEEK FROM occurred_at)) m, (SELECT user_id,
-- MIN(EXTRACT(WEEK FROM occurred_at)) AS first  FROM table_events WHERE event_name = "login"
-- GROUP BY user_id) n where m.user_id=n.user_id

Select m.user_id,m.login_week,n.first as first,
m.login_week-first as week_number from
(SELECT user_id, EXTRACT(WEEK FROM occurred_at)
AS login_week FROM table_email_events WHERE action = "email_open" GROUP BY user_id ,
EXTRACT(WEEK FROM occurred_at)) m, (SELECT user_id,
MIN(EXTRACT(WEEK FROM occurred_at)) AS first  FROM table_email_events WHERE action = "email_open"
```

user_id	login_week	first	week_number
0	22	22	0
0	23	22	1
0	24	22	2
0	30	22	8
0	33	22	11
4	19	19	0
4	20	19	1
4	25	19	6
4	26	19	7
4	32	19	13

```
#   Time      Action                                                                                              Message                                                       Duration / Fetch
7   13:24:57  SELECT WEEK(occurred_at) AS WEEK,      COUNT(DISTINCT user_id) AS WUE_per_device,      d...        45 row(s) returned                                            0.016 sec / 0.000 sec
8   13:29:28  SELECT WEEK(occurred_at) AS WEEK,      COUNT(DISTINCT user_id) AS WUE_per_device     FRO...        18 row(s) returned                                            0.359 sec / 0.000 sec
9   13:29:42  SELECT WEEK(occurred_at) AS WEEK,      COUNT(DISTINCT user_id) AS WUE_for_email      FROM ...      18 row(s) returned                                            0.282 sec / 0.000 sec
10  13:32:34  Select m.user_id,m.login_week,n.first as first, m.login_week-first as week_number from  (SELECT user_id, EX...  Error Code: 1054. Unknown column 'event_name' in 'where clause'  0.000 sec
11  13:33:26  Select m.user_id,m.login_week,n.first as first, m.login_week-first as week_number from  (SELECT user_id, EX...  1000 row(s) returned                                          0.297 sec / 0.000 sec
```

Results:

While making this project I faced many challenges and had to search and for answering many of them but because of this I now have a much clear understanding of operation metrics and how is it used for not only gaining but also retaining the users in a service.