

# **OPERATION ANALYTICS AND INVESTING METRIC SPIKE**

## **(ADVANCED SQL)**

### **PROJECT DESCRIPTION**

In this project we'll do work as a Lead Data Analyst. We'll do Analyze the data and provide valuable insights that can improve the company's operations and understand changes the key metrics.

In this project, The Operational Analytics is a crucial process that involves analyzing a company's end – to – end operations. This analysis helps to identify areas improvement within the company.

As a Data Analyst, we'll work closely with various teams such as operations team, supports team, and marketing to helping them derive the insights from the data.

This Operational Analytics involves with understanding and explaining the sudden changes in key metrics and

also involve with in daily user engagement or a drop in sales.

As a Data Analyst, we'll analyze the data and extract the valuable output from dataset.

## **APPROACH**

Firstly, I'll take approach to upload the dataset in MySQL then create and insert the values in the database by using the DDL (Data Definition Language) , DML (Data Manipulation Language) by using MySQL workbench.

## **TECH – STACK USED**

The MySQL Workbench Community and MySQL and MySQL community server and MySQLWorkbench release for versions 8.0 through 8.0.34.

## **INSIGHTS**

## **CASE STUDY 1 : Job Data Analysis**

In this case study we'll working with the table job\_data with the following columns.

- job\_id: Unique identifier of jobs
- actor\_id: Unique identifier of actor
- event: The type of event (decision/skip/transfer).
- language: The Language of the content
- time\_spent: Time spent to review the job in seconds.
- org: The Organization of the actor
- ds: The date in the format yyyy/mm/dd (stored as text).

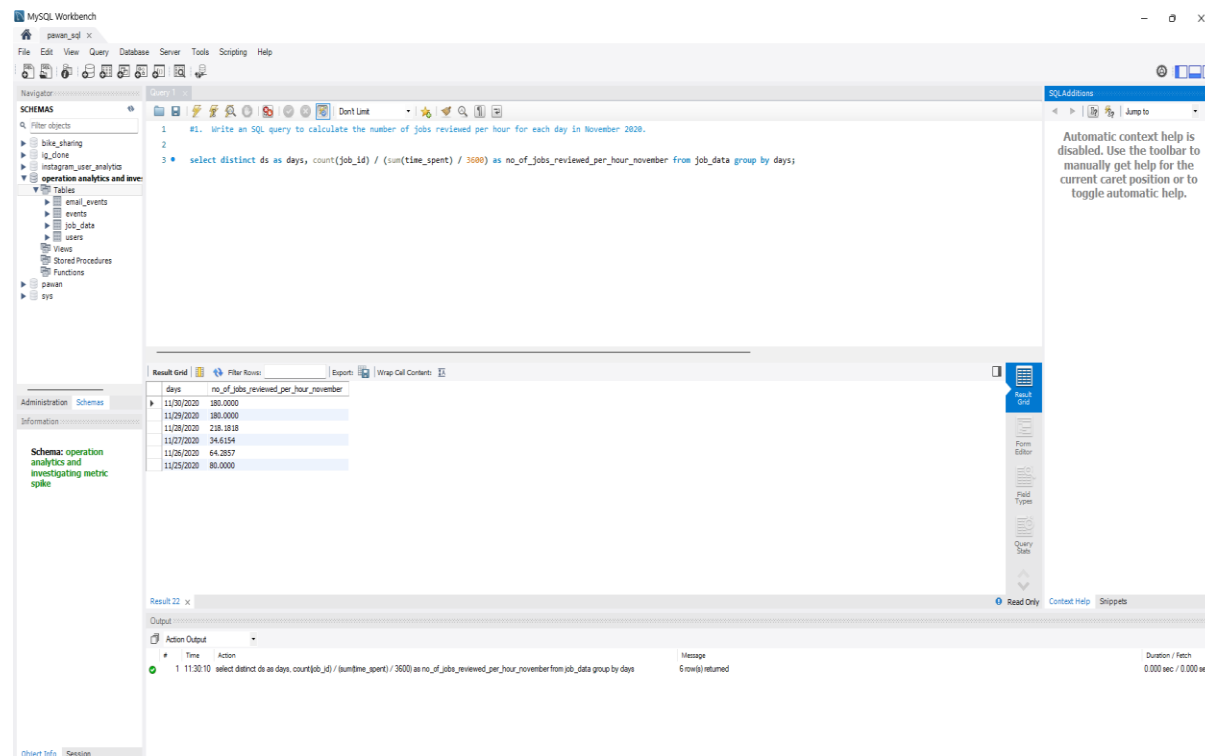
### **A. Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.**

In this question, we have to calculate the numbers of jobs reviewed per hour for each day in November 2020

Firstly, we select the ds column because we have to extract the output with the help of this column consider ds as a days after this we are using the ratio to find the numbers of jobs

reviewed per hour from November from job\_data.

Here's our output :



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 #1. Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.
2
3 select distinct ds as days, count(job_id) / (sum(time_spent) / 3600) as no_of_jobs_reviewed_per_hour_november from job_data group by days;
```

The Results Grid shows the following data:

days	no_of_jobs_reviewed_per_hour_november
11/20/2020	180.0000
11/29/2020	180.0000
11/28/2020	218.1818
11/27/2020	346.154
11/26/2020	64.2857
11/25/2020	80.0000

The Action Output pane shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	11:30:10	select distinct ds as days, count(job_id) / (sum(time_spent) / 3600) as no_of_jobs_reviewed_per_hour_november from job_data group by days	6 row(s) returned	0.000 sec / 0.000 sec

**B. Write an SQL query to calculate the 7 – day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.**

In this question. We have to calculate the 7-day rolling average of throughput.

So, Firstly we select ds column as a day and **throughput** with alias **a**. In this query we are taking average of **ds** row and **a.throughput** and we're giving name as a **7\_day\_avg\_of\_throughput** and we're taking average from the **select ds , count (job\_id)/sum(time spent) as throughput from job\_data** and we're doing **group by ds**.

Here's our output :

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
1 # 2. Write an SQL query to calculate the 7-day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.
2
3 select a.ds as day, a.throughput, avg(a.throughput) over ( order by ds rows between 6 preceding and current row ) AS 7_day_avg_of_throughput
4 from ( select ds, count(job_id) / sum(time_spent) as throughput from job_data group by ds ) as a group by ds;
5
```

The results grid shows the following data:

day	throughput	7_day_avg_of_throughput
11/25/2020	0.0222	0.02220000
11/26/2020	0.0179	0.02095000
11/27/2020	0.0096	0.01656667
11/28/2020	0.0606	0.02757500
11/29/2020	0.0500	0.02060000
11/30/2020	0.0500	0.03595000

The bottom panel shows the action output with the following details:

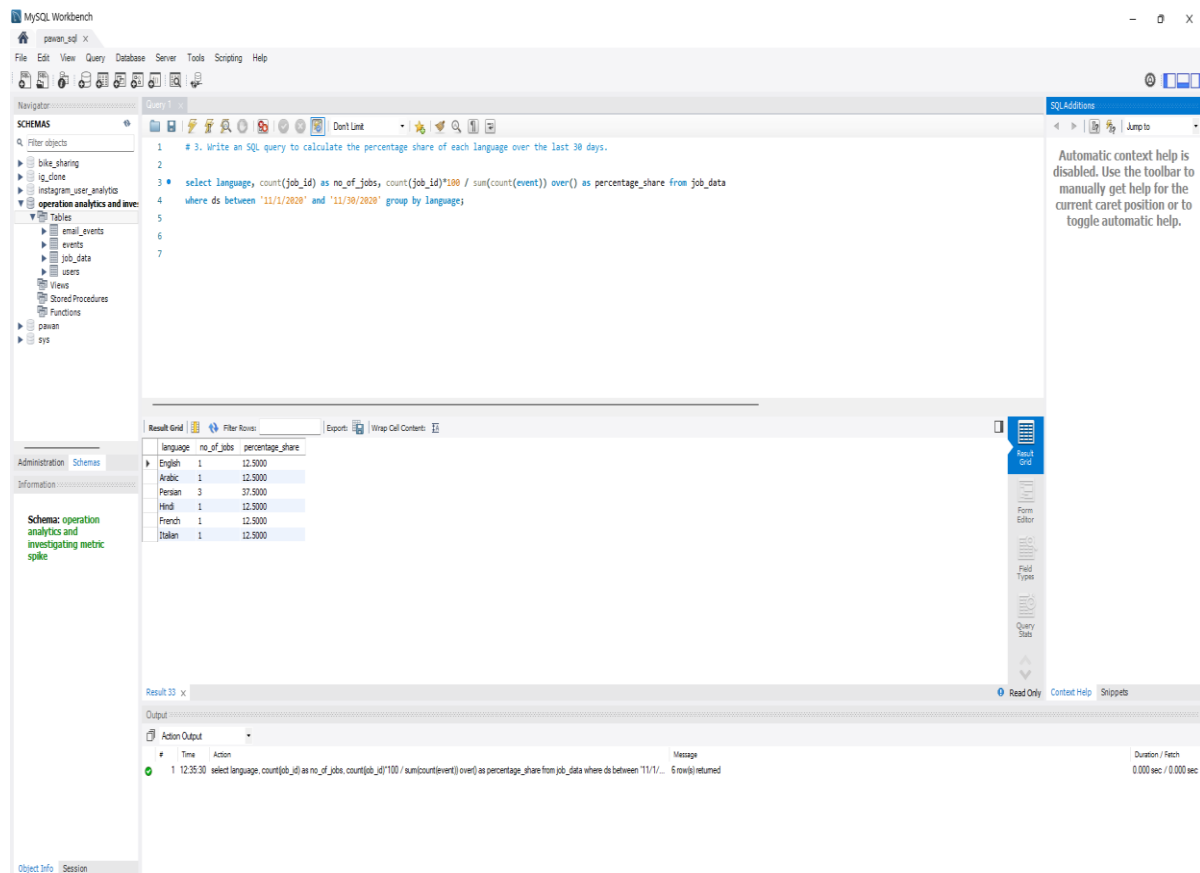
#	Time	Action	Message	Duration / Tech
1	11:30:10	select distinct ds as days, count(job_id) / (sum(time_spent) / 3600) as no_of_jobs_reviewed_per_hour_november from job_data group by days	6 row(s) returned	0.000 sec / 0.000 sec
2	11:40:32	select a.ds as day, a.throughput, avg(a.throughput) over ( order by ds rows between 6 preceding and current row ) AS 7_day_avg_of_throughput from ( s...	6 row(s) returned	0.000 sec / 0.000 sec

**C. Write an SQL query to calculate the percentage share of each language over the last 30 days.**

In this question, we have to calculate the percentage share of each language over the last 30 days.

So. Firstly, we select the language and we're taking ratio of count(job\_id) as no\_of\_jobs,  $\text{count(job\_id)*100 / sum(count(event)) OVER()}$  as percentage\_share. After this giving some constraints of dates to get required output and using the group by languages.

Here's our output :



## D. Write an SQL query to display duplicate rows from the job\_data table.

In this question we have to display the duplicate rows from the job\_data table.

Firstly, we select all the rows by partitioning by various columns and then we add a duplicate column in the main select query defining whether there are duplicates for that row in the dataset.

MySQL Workbench

Navigator

SCHEMAS

Filter objects

- bike\_sharing
- ig\_clone
- instagram\_user\_analytics
- operation\_analytics and investigating metric spike
  - email\_events
  - events
  - job\_data
  - users
  - Views
  - Stored Procedures
  - Functions
- paawan
- sys

Query 1

```

1 # 4. Write an SQL query to display duplicate rows from the job_data table.
2
3 * SELECT a.ds, a.job_id, a.actor_id, a.event, a.language, a.time_spent, a.org, case when a.duplicates = 1 then "No Duplicate" else "Duplicate" end as Duplicate
4 from ( select *, row_number() over (partition by ds, job_id, actor_id, event, language, time_spent, org) as duplicates from job_data ) as a ;
5
6
7
8

```

Result Grid

ds	job_id	actor_id	event	language	time_spent	org	Duplicate
11/25/2020	20	1003	transfer	Italian	45	C	No Duplicate
11/26/2020	23	1004	skip	Persian	56	A	No Duplicate
11/27/2020	11	1007	decision	French	104	D	No Duplicate
11/28/2020	23	1005	transfer	Persian	22	D	No Duplicate
11/28/2020	25	1002	decision	Hindi	11	B	No Duplicate
11/29/2020	23	1003	decision	Persian	20	C	No Duplicate
11/30/2020	21	1001	skip	English	15	A	No Duplicate
11/30/2020	22	1006	transfer	Arabic	25	B	No Duplicate

Result 38 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:40:42	SELECT a.ds, a.job_id, a.actor_id, a.event, a.language, a.time_spent, a.org, case when a.duplicates = 1 then "No Duplicate" else "Duplicate" end as D...	8 row(s) returned	0.000 sec / 0.000 sec

## CASE STUDY 2 : Investing Metric Spike

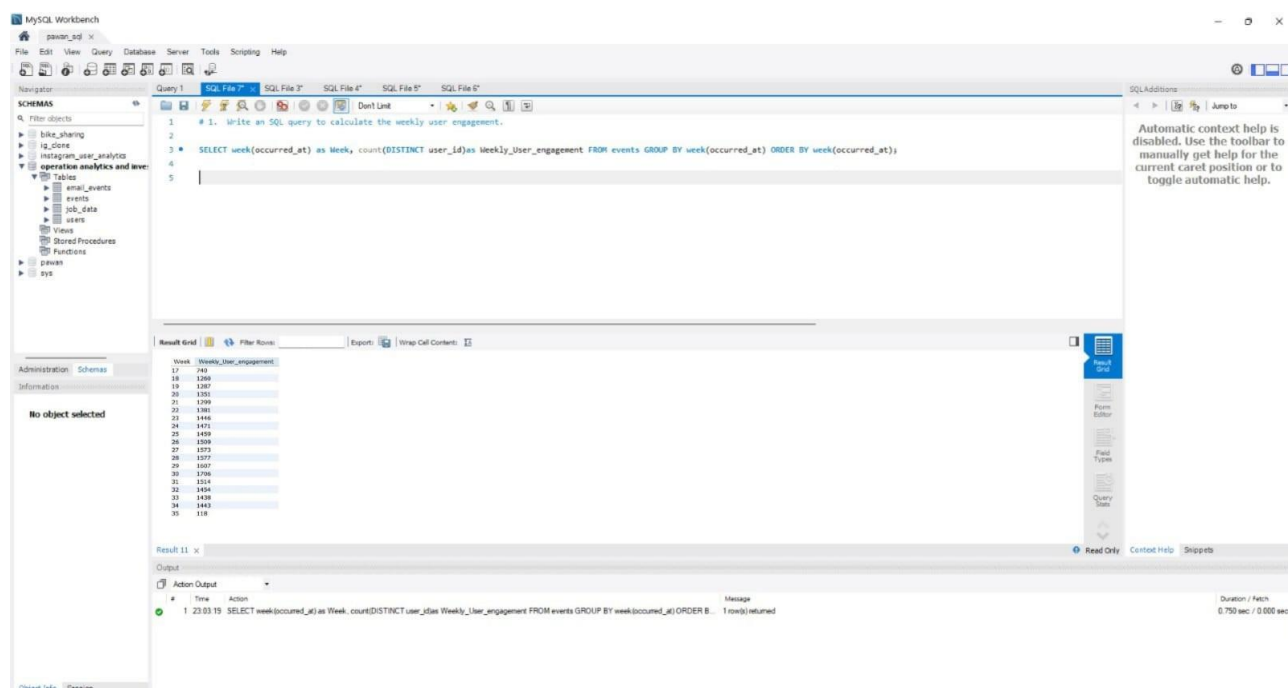
In this case study we're using three tables which are users, events and email\_events.

**A. Write an SQL query to calculate the weekly user engagement.**



In this question we have to calculate the weekly user engagement from the given data.

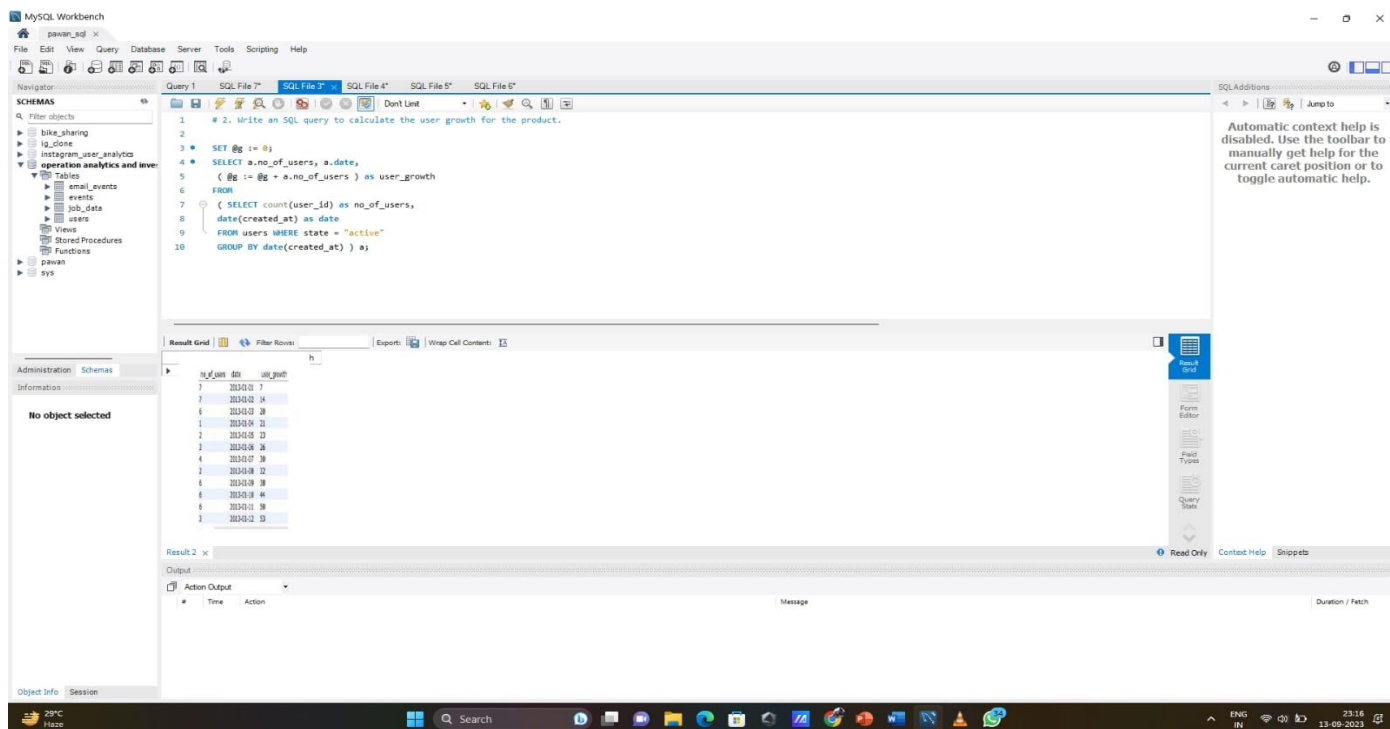
In this query, we firstly extract week from the datetime column we have and also we take count of distinct user\_id for the output. Finally we group by the extracted week column.



## B. Write an SQL query to calculate the user growth for the product.

In this question, we have to calculate the user growth for the product.

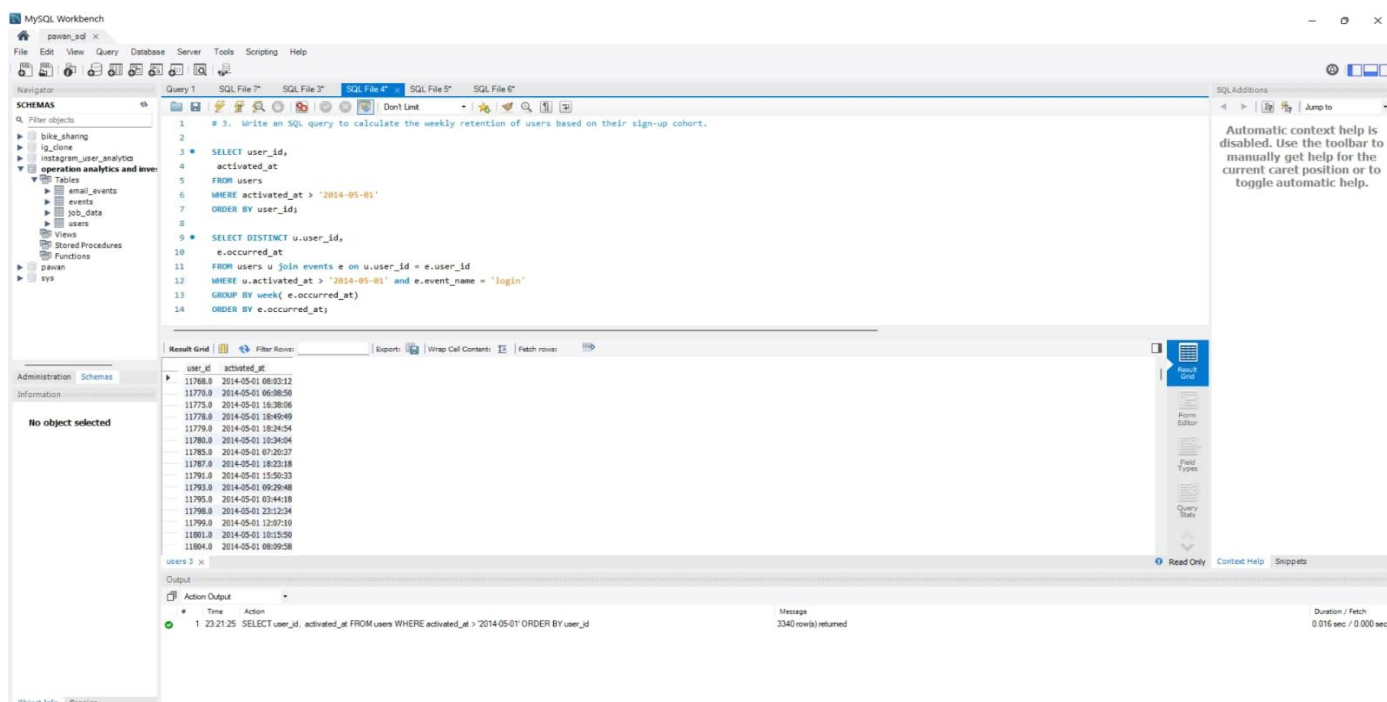
We would use an incremental approach wherein we set a variable to 0 at first and then keep on incrementing that value according to the condition we define according to date and an active status.



### C. Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

In this question, we have to calculate the weekly retention of users based on their sign-up cohort

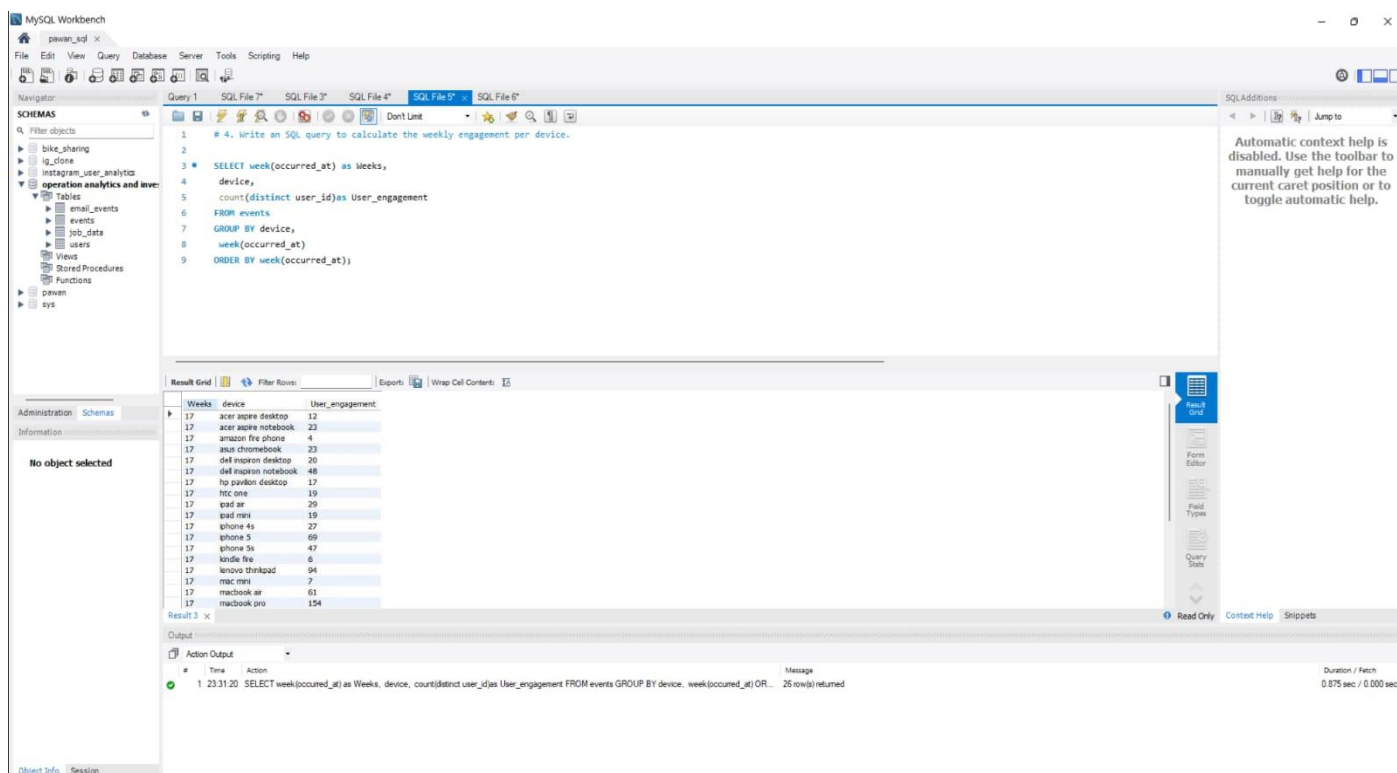
Firstly, We observe from the dataset that the events start from May 2014 onwards so we select all those records where activated date is after the date we mentioned. Then the user activity was extracted from events column for user who signed up after May 2014.



## D. Write an SQL query to calculate the weekly engagement per device.

In this question, we have to calculate the weekly engagement per device.

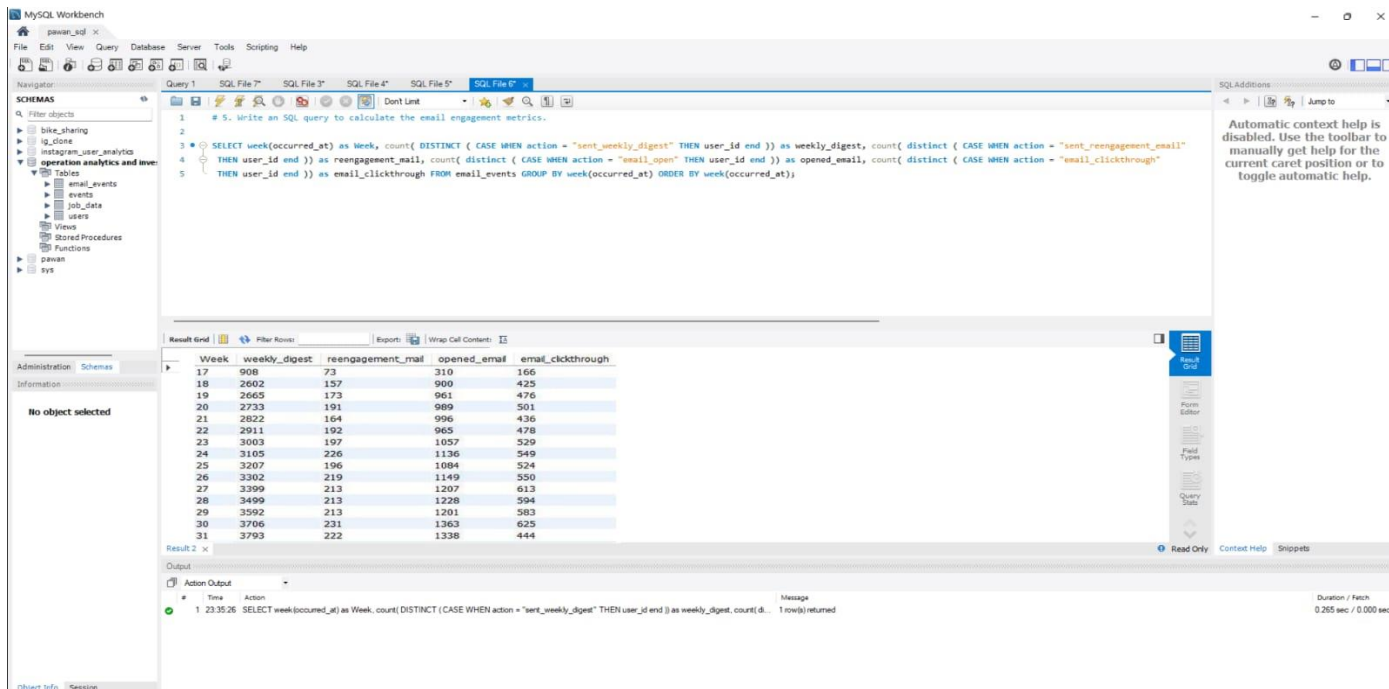
Firstly, for this problem we extract week from the datetime column along with count of distinct user id and we group by these records to get the desired output.



## E. Write an SQL query to calculate the email engagement metrics.

In this question, we have to calculate the email engagement metrics.

For this we select count of distinct user ids which contain different email action using case statements according to the week and finally we group by week to get the desired output.



The screenshot displays the MySQL Workbench interface. The central pane shows a SQL query (Query 1) designed to calculate email engagement metrics by week. The query uses CASE WHEN statements to categorize different email actions into four metrics: weekly\_digest, reengagement\_mail, opened\_email, and email\_clickthrough. The results are grouped by week (occurred\_at) and ordered by week.

The query text is as follows:

```
1 # 5. Write an SQL query to calculate the email engagement metrics.
2
3 * SELECT week(occurred_at) as Week, count( DISTINCT ( CASE WHEN action = "sent_weekly_digest" THEN user_id end )) as weekly_digest, count( distinct ( CASE WHEN action = "sent_reengagement_email"
4 THEN user_id end )) as reengagement_mail, count( distinct ( CASE WHEN action = "email_open" THEN user_id end )) as opened_email, count( distinct ( CASE WHEN action = "email_clickthrough"
5 THEN user_id end )) as email_clickthrough FROM email_events GROUP BY week(occurred_at) ORDER BY week(occurred_at);
```

The Results grid below the query shows the output of the query. The columns are Week, weekly\_digest, reengagement\_mail, opened\_email, and email\_clickthrough. The data is grouped by week, with rows 17 through 31 displayed.

Week	weekly_digest	reengagement_mail	opened_email	email_clickthrough
17	908	73	310	166
18	2602	157	900	425
19	2665	173	961	476
20	2733	191	989	501
21	2822	164	996	436
22	2911	192	965	478
23	3003	197	1057	529
24	3105	226	1136	549
25	3207	196	1084	524
26	3302	219	1149	550
27	3399	213	1207	613
28	3499	213	1228	594
29	3592	213	1201	583
30	3706	231	1363	625
31	3793	222	1338	444

The Output pane at the bottom shows the execution of the query, indicating it was successful and returned 1 row(s).

## CONCLUSION

- From this project, I learnt how to deal with advanced SQL queries and how to solve and how to apply the functions to extract the valuable and required outcome from the dataset.

- I also learnt from project how the Lead Data Analyst work in this kind of projects and how they deal with it.
- Also, they are dealing and working with various teams such as operationals, support team , marketing team and the Data Analyst helping and giving them valuable insights from the data.
- I gained my advanced SQL skills to analyze the data and improve the company's operations and their key metrics