

# Operation Analytics and Investigating Metric Spike

## Description:

This project involves operation analytics and investigating metric spike.

- The task to us as a data analyst is we have to analyze the given data and give insights to the other working teams in the company by working on the data given to us.

We have to deal with two case studies : Case Study – 1(Job data)

Case Study - 2(Investigating metric spike)

- In the first case study we have to deal with only 1 table (job\_data) with 7 columns and 8 rows , we have to analyze it by solving the queries asked.
- In the second case study there are 3 tables - (users,events,email\_events) which are related as parent and child tables using foreign key and the data is provided in the form of csv file .
- I have imported csv files using import wizard .
- Overall goal of the project is to analyze user engagement, growth, weekly retention, weekly engagement and email engagement metrics .

## Approach:

First we will solve the case study – 1 .

- We should create a database and insert the data into the given table .
- In this case study we have 4 questions to solve .

### A . Number of jobs reviewed:

We have to calculate the no of jobs reviewed per hour per day in the month november of year 2020.

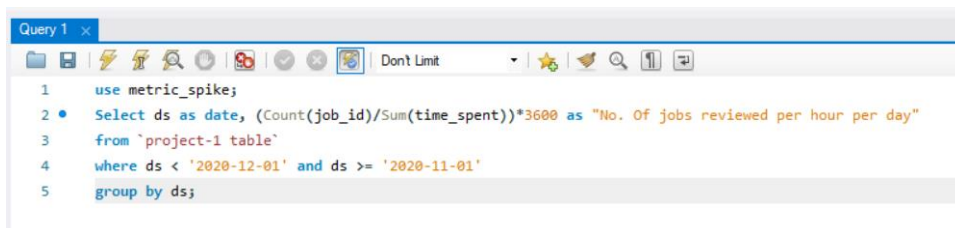
The columns involved in solving the above question are ds for the date, time\_spent and we have to multiply it by 3600 to convert it into hour , and as we require the avg we need to calculate  $\text{sum}(\text{time\_spent})/\text{count}(\text{job\_id})$  .

We have to use group by ds such that the average time for every day is required .

At last we have add where condition as we need the data of only the month of november.

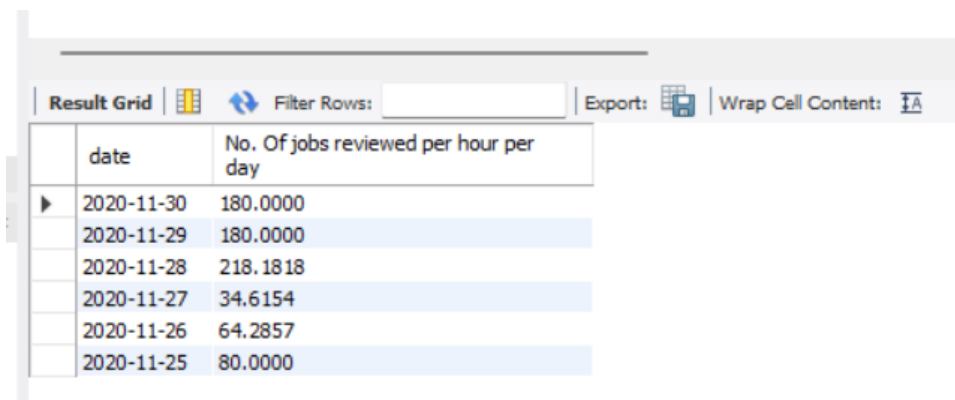
The Query is as follows :

**Select ds as date, (Count(job\_id)/Sum(time\_spent))\*3600 as "No. Of jobs reviewed per hour per day " from 'project-1 table' where ds < '2020-12-01' and ds >= '2020-11-01' group by ds;**



```
Query 1
1 use metric_spike;
2 • Select ds as date, (Count(job_id)/Sum(time_spent))*3600 as "No. Of jobs reviewed per hour per day"
3 from `project-1 table`
4 where ds < '2020-12-01' and ds >= '2020-11-01'
5 group by ds;
```

If we execute the above query we will get the following output:



	date	No. Of jobs reviewed per hour per day
▶	2020-11-30	180.0000
	2020-11-29	180.0000
	2020-11-28	218.1818
	2020-11-27	34.6154
	2020-11-26	64.2857
	2020-11-25	80.0000

## B . Throughput :

It is the no of events happening per second . We need to calculate the 7–day average of the throughput .

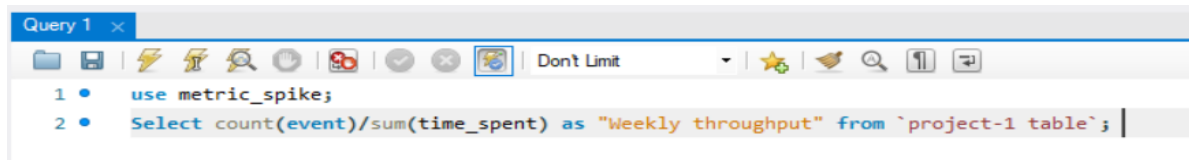
For throughput we can use daily metric for daily patterns and short-term variations and if we are dealing with long – term trends and if we cant go through each and every value we can use 7-day rolling throughput . According to our requirements we can choose between them.

In the given table there are 6 – rows of data corresponding to only one week , so we can calculate these 6-days average as weekly throughput .

The  $\text{count}(\text{event})/\text{sum}(\text{time\_spent})$  will give us the desired value.

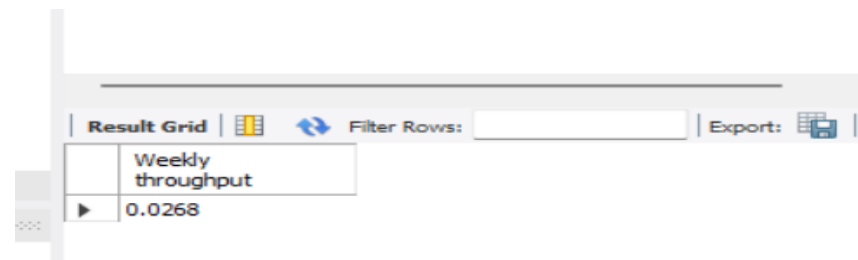
The query is as follows:

**Select  $\text{count}(\text{event})/\text{sum}(\text{time\_spent})$  as “Weekly throughput” from ‘project-1 table’;**



```
Query 1 x
1 • use metric_spike;
2 • Select count(event)/sum(time_spent) as "Weekly throughput" from `project-1 table`;
```

The output is :



Weekly throughput
0.0268

### C. Percentage share of each language :

We have to calculate the percentage share of each language in past 30 days .

For calculating percentage we will divide the count of individual languages by total no of languages , as we have to represent in percentage we will multiply by 100.

For getting individual count we can group by language and count no of rows .

The query is as follows :

**Select language,(100\*count(\*)/(select count(\*) from ‘project-1 table’)) from ‘project-1 table’ group by language;**



```
Query 1 x
1 • use metric_spike;
2 • Select language,(100*count(*)/(select count(*) from `project-1 table`))
3 • from `project-1 table` group by language;
```

The output is as follows:

Result Grid		
Filter Rows: <input type="text"/>		
Export:  Wrap C		
	language	(100*count(*)/(select count(*) from 'project-1 table'))
▶	English	12.5000
	Arabic	12.5000
	Persian	37.5000
	Hindi	12.5000
	French	12.5000
	Italian	12.5000

Result 6 x

### D . Duplicate rows:

If 2 or rows have same values int the table then duplicate rows exist .

If we use group by in each coloumn and check if count is greater than 1 then we can find the duplicate rows in the given table.

The Query is as follows :

**Select \* from 'project-1 table' group by job\_id, actor\_id, event, language, time\_spent, org, ds having Count(\*)>1;**

```

Query 1 x
1 • use metric_spike;
2 • Select * from `project-1 table`
3   group by job_id, actor_id, event, language, time_spent, org, ds
4   having Count(*)>1;

```

The result is as follows :

Result Grid						
Filter Rows: <input type="text"/>						
Export:  Wrap Cell Content:						
ds	job_id	actor_id	event	language	time_spent	org

This shows that are no duplicate rows in the given table.

Next we will solve the case study – 2 .

In the investigating metric spike there are three table involved (users,events and email\_events)

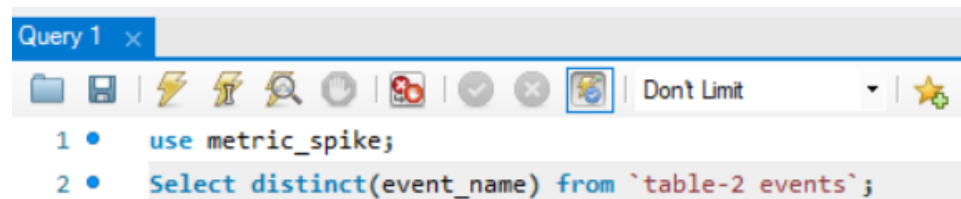
These 3 tables involve large amount of data csv files.

I have imported it using wizard .

There are 5 questions to be solved in investigating metric spike .

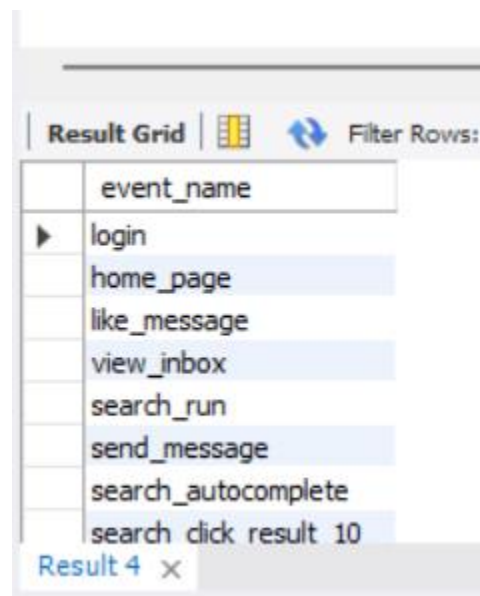
### A. User Engagement :

To calculate the Weekly User enagement we need to find the no of users who performed specific actions mentioned in the table, to know the actions I executed the following query :



```
Query 1 x
1 • use metric_spike;
2 • Select distinct(event_name) from `table-2 events`;
```

The events are:

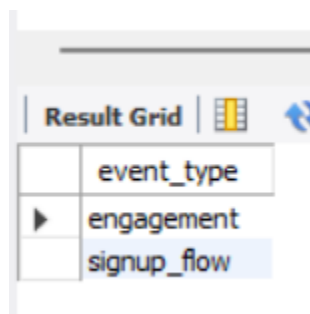


event_name
login
home_page
like_message
view_inbox
search_run
send_message
search_autocomplete
search click result 10

For the type of event\_type I executed the following query :

```
Query 1 x
1 • use metric_spike;
2 • Select distinct(event_type) from `table-2 events`;
```

The type of event\_type are :



event_type
engagement
signup_flow

We can get the week no.s from the “occured\_at” coloumn using the extract function .

We want the event\_type to be engagement .

We should group the extracted week no.s by events and then calculate the count of distinct user\_ids .

The query is as follows:

**Select extract(week from occurred\_at) as “Week No.s”,count(distinct user\_id) as “Weekly Active users “ from ‘table-2 events’ where event\_type=’engagement’ group by extract(week from occurred\_at);**

```
Query 1 x
1 • use metric_spike;
2 • Select extract(week from occurred_at) as 'Week No.s',count(distinct user_id) as 'Weekly Active users'
3   from `table-2 events`
4   where event_type='engagement' group by extract(week from occurred_at);
```

The output is as follows :

Result Grid			Filter Rows:
	Week No.s	Weekly Active users	
▶	17	237	
	18	425	
	19	457	
	20	432	
	21	430	
	22	458	
	23	441	

Result 8 ×

Output

## B . User growth:

We should calculate the user growth after a period of time by finding the no of new users who started using the product .

We will calculate the user growth for every month .

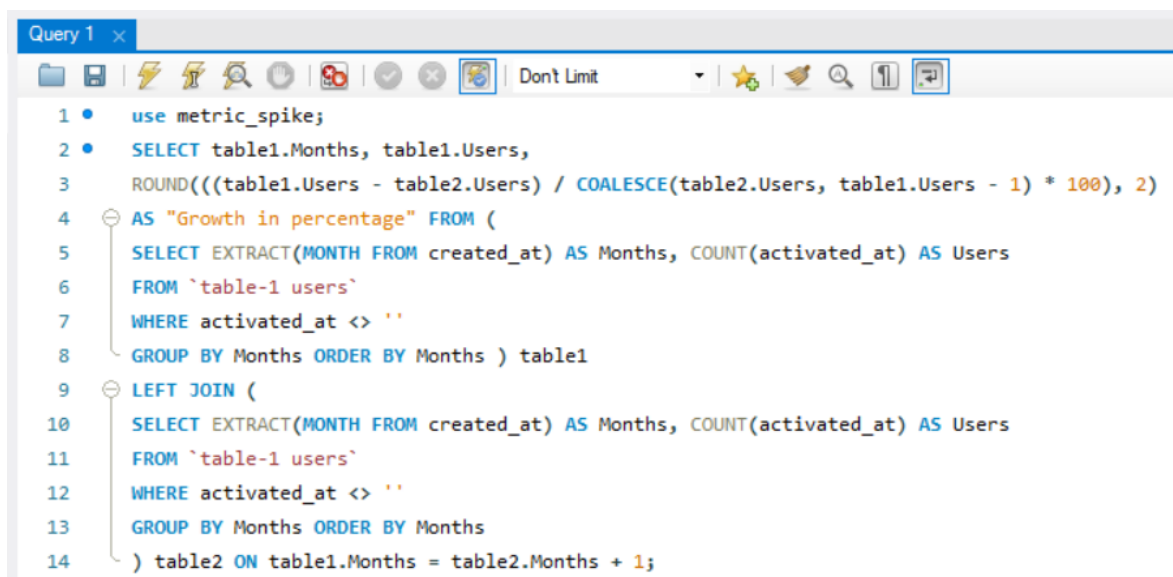
To do that we should find the no of users who started using at a particular month by the 'activated' column by using extract function we can get the month and using count of activated where activated is not null we will get the no of users who started using the product in a particular month .

We will run the same query 2 times to get the months at the individual and use the left join operation on the condition that (first query month + 1 = second query month)

And then we can subtract the users of table1 and table2 and divide it by the users of the preceding month and if the previous month count is 0 then we should replace it by table1users - 1 and to do that we will use the coalesce function we used for division by zero error .

The query is as follows :

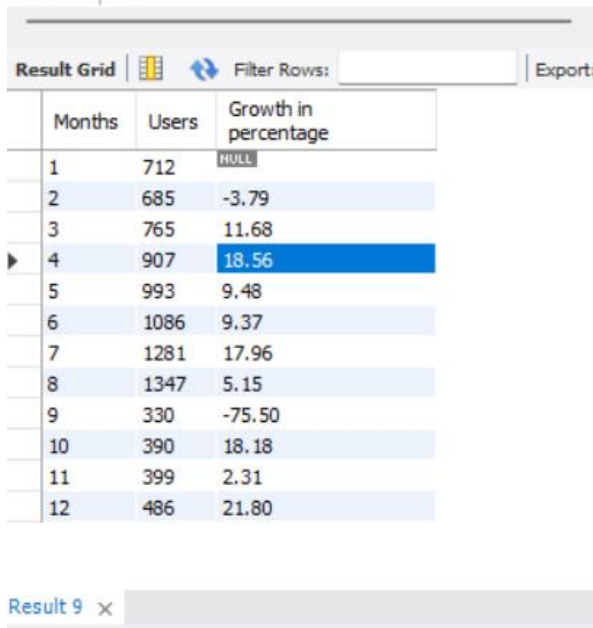
```
SELECT table1.Months, table1.Users,  
ROUND((((table1.Users - table2.Users) / COALESCE(table2.Users, table1.Users -  
1) * 100), 2) AS "Growth in percentage"  
  
FROM  
  
(SELECT EXTRACT(MONTH FROM created_at) AS Months, COUNT(activated_at)  
AS Users FROM `table-1 users` WHERE activated_at <> '' GROUP BY Months  
  
ORDER BY Months ) table1  
  
LEFT JOIN (  
  
SELECT EXTRACT(MONTH FROM created_at) AS Months, COUNT(activated_at)  
AS Users FROM `table-1 users` WHERE activated_at <> '' GROUP BY Months  
  
ORDER BY Months) table2  
  
ON table1.Months = table2.Months + 1;
```

A screenshot of a SQL query editor window titled "Query 1". The window has a toolbar with various icons for file operations, execution, and search. The SQL code is displayed in a monospaced font with syntax highlighting. The code is as follows:

```
1 • use metric_spike;  
2 • SELECT table1.Months, table1.Users,  
3   ROUND((((table1.Users - table2.Users) / COALESCE(table2.Users, table1.Users - 1) * 100), 2)  
4   AS "Growth in percentage" FROM (  
5     SELECT EXTRACT(MONTH FROM created_at) AS Months, COUNT(activated_at) AS Users  
6     FROM `table-1 users`  
7     WHERE activated_at <> ''  
8     GROUP BY Months ORDER BY Months ) table1  
9   LEFT JOIN (  
10    SELECT EXTRACT(MONTH FROM created_at) AS Months, COUNT(activated_at) AS Users  
11    FROM `table-1 users`  
12    WHERE activated_at <> ''  
13    GROUP BY Months ORDER BY Months  
14  ) table2 ON table1.Months = table2.Months + 1;
```



The output is as follows :



	Months	Users	Growth in percentage
1	712	HULL	
2	685	-3.79	
3	765	11.68	
4	907	18.56	
5	993	9.48	
6	1086	9.37	
7	1281	17.96	
8	1347	5.15	
9	330	-75.50	
10	390	18.18	
11	399	2.31	
12	486	21.80	

### C. Weekly Retention :

We need to find the no of users who continue to engage with the product after they started using the product .

The query is as follows :

```
SELECT first AS "Week Number",  
SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS "Week 0",  
SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS "Week 1",  
SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS "Week 2",  
SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS "Week 3",  
SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS "Week 4",  
SUM(CASE WHEN week_number = 5 THEN 1 ELSE 0 END) AS "Week 5",  
SUM(CASE WHEN week_number = 6 THEN 1 ELSE 0 END) AS "Week 6",
```

SUM(CASE WHEN week\_number = 7 THEN 1 ELSE 0 END) AS "Week 7",  
SUM(CASE WHEN week\_number = 8 THEN 1 ELSE 0 END) AS "Week 8",  
SUM(CASE WHEN week\_number = 9 THEN 1 ELSE 0 END) AS "Week 9",  
SUM(CASE WHEN week\_number = 10 THEN 1 ELSE 0 END) AS "Week 10",  
SUM(CASE WHEN week\_number = 11 THEN 1 ELSE 0 END) AS "Week 11",  
SUM(CASE WHEN week\_number = 12 THEN 1 ELSE 0 END) AS "Week 12",  
SUM(CASE WHEN week\_number = 13 THEN 1 ELSE 0 END) AS "Week 13",  
SUM(CASE WHEN week\_number = 14 THEN 1 ELSE 0 END) AS "Week 14",  
SUM(CASE WHEN week\_number = 15 THEN 1 ELSE 0 END) AS "Week 15",  
SUM(CASE WHEN week\_number = 16 THEN 1 ELSE 0 END) AS "Week 16",  
SUM(CASE WHEN week\_number = 17 THEN 1 ELSE 0 END) AS "Week 17",  
SUM(CASE WHEN week\_number = 18 THEN 1 ELSE 0 END) AS "Week 18"

from

( SELECT table1.user\_id, table1.login\_week, table2.first, table1.login\_week - first  
AS week\_number

FROM

(SELECT user\_id, EXTRACT(WEEK FROM occurred\_at) AS login\_week FROM  
`table-2 events` GROUP BY 1, 2) table1,

(SELECT user\_id, MIN(EXTRACT(WEEK FROM occurred\_at)) AS first FROM `table-  
2 events` GROUP BY 1) table2

WHERE table1.user\_id=table2.user\_id ) sub GROUP BY first ORDER BY first;

```

Query 1 x
[Icons] Don't Limit

2 • SELECT first AS "Week Number",
3     SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS "Week 0",
4     SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS "Week 1",
5     SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS "Week 2",
6     SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS "Week 3",
7     SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS "Week 4",
8     SUM(CASE WHEN week_number = 5 THEN 1 ELSE 0 END) AS "Week 5",
9     SUM(CASE WHEN week_number = 6 THEN 1 ELSE 0 END) AS "Week 6",
10    SUM(CASE WHEN week_number = 7 THEN 1 ELSE 0 END) AS "Week 7",
11    SUM(CASE WHEN week_number = 8 THEN 1 ELSE 0 END) AS "Week 8",
12    SUM(CASE WHEN week_number = 9 THEN 1 ELSE 0 END) AS "Week 9",
13    SUM(CASE WHEN week_number = 10 THEN 1 ELSE 0 END) AS "Week 10",
14    SUM(CASE WHEN week_number = 11 THEN 1 ELSE 0 END) AS "Week 11",
15    SUM(CASE WHEN week_number = 12 THEN 1 ELSE 0 END) AS "Week 12",
16    SUM(CASE WHEN week_number = 13 THEN 1 ELSE 0 END) AS "Week 13",
17    SUM(CASE WHEN week_number = 14 THEN 1 ELSE 0 END) AS "Week 14",
18    SUM(CASE WHEN week_number = 15 THEN 1 ELSE 0 END) AS "Week 15",
19    SUM(CASE WHEN week_number = 16 THEN 1 ELSE 0 END) AS "Week 16",
20    SUM(CASE WHEN week_number = 17 THEN 1 ELSE 0 END) AS "Week 17",
21    SUM(CASE WHEN week_number = 18 THEN 1 ELSE 0 END) AS "Week 18"
22
23    FROM
24    ( SELECT table1.user_id, table1.login_week, table2.first, table1.login_week - first AS week_number
25      FROM
26      (SELECT user_id, EXTRACT(WEEK FROM occurred_at) AS login_week FROM `table-2 events` GROUP BY 1, 2) table1,
27      (SELECT user_id, MIN(EXTRACT(WEEK FROM occurred_at)) AS first FROM `table-2 events` GROUP BY 1) table2
28      WHERE table1.user_id=table2.user_id ) sub GROUP BY first ORDER BY first;

```

The result is as follows :

Week Numbers	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
17	237	121	85	74	64	52	46	48	41	39	43	35	34
18	304	85	59	46	39	34	39	30	29	35	34	33	36
19	287	72	53	46	34	36	32	31	36	31	27	24	26
20	227	41	36	24	20	15	15	11	11	14	12	7	5
21	226	43	28	19	16	13	13	9	12	9	10	6	4
22	242	40	27	22	15	13	15	15	10	10	9	8	7
23	235	39	25	16	12	14	12	13	12	11	7	7	0
24	277	51	35	27	24	20	21	19	11	7	10	0	0
25	244	35	25	26	20	17	13	12	8	10	0	0	0
26	234	30	24	13	16	10	5	6	1	0	10	0	0
27	247	30	15	14	10	5	6	8	0	0	0	0	0
28	234	34	14	8	2	4	2	0	0	0	0	0	0
29	236	28	18	7	4	1	0	0	0	0	0	0	0
30	255	24	17	10	7	0	0	0	0	0	0	0	0

Result Grid														
Filter Rows:														
Export:														
Wrap Cell Contents:														
Week Numbers	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	
22	242	40	27	22	15	13	15	15	10	10	9	8	7	
23	235	39	25	16	12	14	12	13	12	11	7	7	0	
24	277	51	35	27	24	20	21	19	11	7	10	0	0	
25	244	35	25	26	20	17	13	12	8	10	0	0	0	
26	234	30	24	13	16	10	5	6	1	0	0	0	0	
27	247	30	15	14	10	5	6	8	0	0	0	0	0	
28	234	34	14	8	2	4	2	0	0	0	0	0	0	
29	236	28	18	7	4	1	0	0	0	0	0	0	0	
30	255	24	17	10	7	0	0	0	0	0	0	0	0	
31	195	23	8	8	0	0	0	0	0	0	0	0	0	
32	247	15	10	0	0	0	0	0	0	0	0	0	0	
33	269	29	1	0	0	0	0	0	0	0	0	0	0	
34	272	3	0	0	0	0	0	0	0	0	0	0	0	
35	18	0	0	0	0	0	0	0	0	0	0	0	0	

### Subquery "table1":

This subquery retrieves the user\_id and the week number of each login occurrence from the `table-2 events` table.

The EXTRACT(WEEK FROM occurred\_at) function extracts the week number from the "occurred\_at" timestamp.

The GROUP BY clause groups the results by the user\_id and login\_week.

### Subquery "table2":

This subquery retrieves the user\_id and the minimum (first) week number for each user from the `table-2 events` table.

The MIN(EXTRACT(WEEK FROM occurred\_at)) function retrieves the earliest week number for each user.

The GROUP BY clause groups the results by the user\_id.

The main query:

The main query joins the "table1" and "table2" subqueries using the user\_id column to associate the user's login week with their first login week.

The column first represents the first login week for each user.

The column week\_number calculates the difference between the login\_week and the first login week, representing the week number relative to the user's first login.

The SUM(CASE WHEN week\_number = X THEN 1 ELSE 0 END) expressions count the occurrences where the week\_number matches a specific value (0-18) and

assigns a 1 to that week number and 0 otherwise. This effectively counts the occurrences for each week number.

The result is grouped by the first column (representing the first login week) and ordered in ascending order.

The query provides a breakdown of the number of occurrences for each week number relative to the user's first login week. The week numbers are represented as columns, and the counts are calculated using conditional aggregation with the CASE statement and the SUM function. The final result set displays the "Week Number" (first login week) and the occurrences for each week number from 0 to 18.

#### **D . Weekly Engagement :**

The Weekly engagement aims to analyze the number of distinct users using specific devices for each week based on engagement events stored in the 'table-2 events'.

The query is as follows :

```
SELECT EXTRACT(WEEK FROM occurred_at) AS "Week Numbers",  
COUNT(DISTINCT CASE WHEN device IN('dell inspiron notebook') THEN user_id  
ELSE NULL END) AS "Dell Inspiron Notebook",  
COUNT(DISTINCT CASE WHEN device IN('iphone 5') THEN user_id ELSE NULL  
END) AS "iPhone 5",  
COUNT(DISTINCT CASE WHEN device IN('iphone 4s') THEN user_id ELSE NULL  
END) AS "iPhone 4S",  
COUNT(DISTINCT CASE WHEN device IN('windows surface') THEN user_id ELSE  
NULL END) AS "Windows Surface",  
COUNT(DISTINCT CASE WHEN device IN('macbook air') THEN user_id ELSE NULL  
END) AS "Macbook Air",  
COUNT(DISTINCT CASE WHEN device IN('iphone 5s') THEN user_id ELSE NULL  
END) AS "iPhone 5S",
```

COUNT(DISTINCT CASE WHEN device IN('macbook pro') THEN user\_id ELSE NULL  
END) AS "Macbook Pro",

COUNT(DISTINCT CASE WHEN device IN('kindle fire') THEN user\_id ELSE NULL  
END) AS "Kindle Fire",

COUNT(DISTINCT CASE WHEN device IN('ipad mini') THEN user\_id ELSE NULL  
END) AS "iPad Mini",

COUNT(DISTINCT CASE WHEN device IN('nexus 7') THEN user\_id ELSE NULL END)  
AS "Nexus 7",

COUNT(DISTINCT CASE WHEN device IN('nexus 51') THEN user\_id ELSE NULL  
END) AS "Nexus 5",

COUNT(DISTINCT CASE WHEN device IN('samsung galaxy s4') THEN user\_id ELSE  
NULL END) AS "Samsung Galaxy S4",

COUNT(DISTINCT CASE WHEN device IN('lenovo thinkpad') THEN user\_id ELSE  
NULL END) AS "Lenovo Thinkpad",

COUNT(DISTINCT CASE WHEN device IN('Csamsune galaxy tablet') THEN user\_id  
ELSE NULL END) AS "Samsung Galaxy Tabler",

COUNT(DISTINCT CASE WHEN device IN('Nacer aspire notebook') THEN user\_id  
ELSE NULL END) AS "Acer Aspire Notebook",

COUNT(DISTINCT CASE WHEN device IN ('Camus chromebook') THEN user\_id  
ELSE NULL END) AS "Ass Chromebook",

COUNT(DISTINCT CASE WHEN device IN ('Chtc one') THEN user\_id ELSE NULL  
END) AS "HTC One",

COUNT(DISTINCT CASE WHEN device IN('nokia lumia 635') THEN user\_id ELSE  
NULL END) AS "Nokia Lumia 635",

COUNT(DISTINCT CASE WHEN device IN('samsung galaxy note') THEN user\_id  
ELSE NULL END) AS "Samsung Galaxy Note",

COUNT(DISTINCT CASE WHEN device IN('acer aspire desktop') THEN user\_id  
ELSE NULL END) AS "Acer Aspire Desktop",

COUNT(DISTINCT CASE WHEN device IN('mac mini') THEN user\_id ELSE NULL  
END) AS "Mac Mini",

COUNT(DISTINCT CASE WHEN device IN('Chp pavilion desktop') THEN user\_id  
ELSE NULL END) AS "HP Pavilion Desktop",

COUNT(DISTINCT CASE WHEN device IN('dell inspiron desktop') THEN user\_id  
ELSE NULL END) AS "Dell Inspiron Desktop",

COUNT(DISTINCT CASE WHEN device IN('ipad air') THEN user\_id ELSE NULL END)  
AS "iPad Air",

COUNT(DISTINCT CASE WHEN device IN('amazon fire phone') THEN user\_id ELSE  
NULL END) AS "Amazon Fire Phone",

COUNT(DISTINCT CASE WHEN device IN('nexus 10') THEN user\_id ELSE NULL  
END) AS "Nexus 10"

FROM `table-2 events`

WHERE event\_type = 'engagement'

GROUP BY 1

ORDER BY 1;

```
Query 1 x
1 • use metric_spike;
2 • SELECT EXTRACT(WEEK FROM occurred_at) AS "Week Numbers",
3 COUNT(DISTINCT CASE WHEN device IN('dell inspiron notebook') THEN user_id ELSE NULL END) AS "Dell Inspiron Notebook",
4 COUNT(DISTINCT CASE WHEN device IN('iphone 5') THEN user_id ELSE NULL END) AS "iPhone 5",
5 COUNT(DISTINCT CASE WHEN device IN('iphone 4s') THEN user_id ELSE NULL END) AS "iPhone 4S",
6 COUNT(DISTINCT CASE WHEN device IN('windows surface') THEN user_id ELSE NULL END) AS "Windows Surface",
7 COUNT(DISTINCT CASE WHEN device IN('macbook air') THEN user_id ELSE NULL END) AS "Macbook Air",
8 COUNT(DISTINCT CASE WHEN device IN('iphone 5s') THEN user_id ELSE NULL END) AS "iPhone 5S",
9 COUNT(DISTINCT CASE WHEN device IN('macbook pro') THEN user_id ELSE NULL END) AS "Macbook Pro",
10 COUNT(DISTINCT CASE WHEN device IN('kindle fire') THEN user_id ELSE NULL END) AS "Kindle Fire",
11 COUNT(DISTINCT CASE WHEN device IN('ipad mini') THEN user_id ELSE NULL END) AS "iPad Mini",
12 COUNT(DISTINCT CASE WHEN device IN('nexus 7') THEN user_id ELSE NULL END) AS "Nexus 7",
13 COUNT(DISTINCT CASE WHEN device IN('nexus 5l') THEN user_id ELSE NULL END) AS "Nexus 5",
14 COUNT(DISTINCT CASE WHEN device IN('samsung galaxy s4') THEN user_id ELSE NULL END) AS "Samsung Galaxy S4",
15 COUNT(DISTINCT CASE WHEN device IN('lenovo thinkpad') THEN user_id ELSE NULL END) AS "Lenovo Thinkpad",
16 COUNT(DISTINCT CASE WHEN device IN('Samsung galaxy tablet') THEN user_id ELSE NULL END) AS "Samsung Galaxy Tabler",
17 COUNT(DISTINCT CASE WHEN device IN('Nacer aspire notebook') THEN user_id ELSE NULL END) AS "Acer Aspire Notebook",
18 COUNT(DISTINCT CASE WHEN device IN ('Camus chromebook') THEN user_id ELSE NULL END) AS "Ass Chromebook",
19 COUNT(DISTINCT CASE WHEN device IN ('Chct one') THEN user_id ELSE NULL END) AS "HTC One",
20 COUNT(DISTINCT CASE WHEN device IN('nokia lumia 635') THEN user_id ELSE NULL END) AS "Nokia Lumia 635",
21 COUNT(DISTINCT CASE WHEN device IN('samsung galaxy note') THEN user_id ELSE NULL END) AS "Samsung Galaxy Note",
```



```

17 COUNT(DISTINCT CASE WHEN device IN(' Nacer aspire notebook') THEN user_id ELSE NULL END) AS "Acer Aspire Notebook",
18 COUNT(DISTINCT CASE WHEN device IN ('Camus chromebook') THEN user_id ELSE NULL END) AS "Ass Chromebook",
19 COUNT(DISTINCT CASE WHEN device IN ('Chtc one') THEN user_id ELSE NULL END) AS "HTC One",
20 COUNT(DISTINCT CASE WHEN device IN('nokia lumia 635') THEN user_id ELSE NULL END) AS "Nokia Lumia 635",
21 COUNT(DISTINCT CASE WHEN device IN('samsung galaxy note') THEN user_id ELSE NULL END) AS "Samsung Galaxy Note",
22 COUNT(DISTINCT CASE WHEN device IN('acer aspire desktop') THEN user_id ELSE NULL END) AS "Acer Aspire Desktop",
23 COUNT(DISTINCT CASE WHEN device IN('mac mini') THEN user_id ELSE NULL END) AS "Mac Mini",
24 COUNT(DISTINCT CASE WHEN device IN('Chp pavilion desktop') THEN user_id ELSE NULL END) AS "HP Pavilion Desktop",
25 COUNT(DISTINCT CASE WHEN device IN('dell inspiron desktop') THEN user_id ELSE NULL END) AS "Dell Inspiron Desktop",
26 COUNT(DISTINCT CASE WHEN device IN('ipad air') THEN user_id ELSE NULL END) AS "iPad Air",
27 COUNT(DISTINCT CASE WHEN device IN('amazon fire phone') THEN user_id ELSE NULL END) AS "Amazon Fire Phone",
28 COUNT(DISTINCT CASE WHEN device IN('nexus 10') THEN user_id ELSE NULL END) AS "Nexus 10"
29 FROM `table-2 events`
30 WHERE event_type = 'engagement'
31 GROUP BY 1
32 ORDER BY 1;

```

The output is :

Week Numbers	Dell Inspiron Notebook	iPhone 5	iPhone 4S	Windows Surface	Macbook Air	iPhone 5S	Macbook Pro	Kindle Fire	iPad Mini	Nexus 7	Nexus 5	Samsung Galaxy S4
17	15	29	7	2	16	21	52	2	5	6	0	22
18	29	37	16	2	46	23	104	13	12	10	0	33
19	33	47	16	4	46	28	103	6	10	14	0	34
20	29	37	18	7	39	28	106	6	11	8	0	26
21	31	53	21	8	37	25	91	11	10	7	0	27
22	37	35	15	4	54	31	84	7	9	18	0	36
23	33	53	22	3	44	24	81	11	10	8	0	21
24	35	59	25	13	53	29	92	12	9	19	0	39
25	43	50	15	9	38	19	102	6	7	13	0	40
26	32	49	21	7	47	32	94	9	9	14	0	36
27	22	51	34	10	49	23	117	12	9	12	0	39
28	31	47	25	11	61	25	97	11	7	12	0	53
29	38	42	18	6	54	31	102	12	8	14	0	37
30	40	51	25	7	40	36	101	2	8	17	0	30
31	26	43	16	7	39	19	105	2	6	10	0	33
32	26	41	12	4	34	24	113	6	5	9	0	24
33	27	33	13	7	38	26	109	5	11	12	0	28
34	32	29	18	9	43	26	97	3	5	13	0	26
35	4	0	3	1	2	0	4	0	1	0	0	1

The main query selects the week number extracted from the "occurred\_at" timestamp using the EXTRACT(WEEK FROM occurred\_at) function as "Week Numbers".

The query then counts the number of distinct user IDs for each device type using the COUNT(DISTINCT CASE WHEN device IN .. THEN user\_id ELSE NULL END) expressions.

Each device type has its own COUNT(DISTINCT CASE WHEN device IN .. THEN user\_id ELSE NULL END) expression to calculate the number of distinct users for that device type.

The device types are listed in the query with their corresponding expressions to count the distinct users.



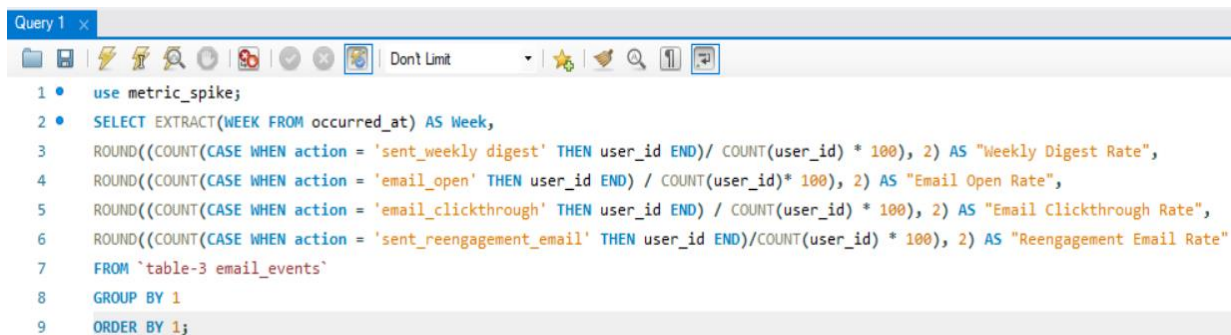
The result set is grouped by the week numbers ("Week Numbers") and ordered in ascending order.

#### E . Weekly Engagement :

Weekly Engagement aims to analyze the rates of different email actions for each week based on email events stored in the table email events .

The query is as follows :

```
SELECT EXTRACT(WEEK FROM occurred_at) AS Week,  
ROUND((COUNT(CASE WHEN action = 'sent_weekly digest' THEN user_id END)/  
COUNT(user_id) * 100), 2) AS "Weekly Digest Rate",  
ROUND((COUNT(CASE WHEN action = 'email_open' THEN user_id END) /  
COUNT(user_id)* 100), 2) AS "Email Open Rate",  
ROUND((COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id END) /  
COUNT(user_id) * 100), 2) AS "Email Clickthrough Rate",  
ROUND((COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id  
END)/COUNT(user_id) * 100), 2) AS "Reengagement Email Rate"  
FROM `table-3 email_events`  
GROUP BY 1  
ORDER BY 1;
```

A screenshot of a SQL query editor interface. The top bar shows 'Query 1' and a 'Don't Limit' button. Below the toolbar, the SQL query is displayed with line numbers 1 through 9. The query is identical to the one shown in the text block above. The editor has a light blue background and a dark blue toolbar.

```
1 • use metric_spike;  
2 • SELECT EXTRACT(WEEK FROM occurred_at) AS Week,  
3   ROUND((COUNT(CASE WHEN action = 'sent_weekly digest' THEN user_id END)/ COUNT(user_id) * 100), 2) AS "Weekly Digest Rate",  
4   ROUND((COUNT(CASE WHEN action = 'email_open' THEN user_id END) / COUNT(user_id)* 100), 2) AS "Email Open Rate",  
5   ROUND((COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id END) / COUNT(user_id) * 100), 2) AS "Email Clickthrough Rate",  
6   ROUND((COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id END)/COUNT(user_id) * 100), 2) AS "Reengagement Email Rate"  
7   FROM `table-3 email_events`  
8   GROUP BY 1  
9   ORDER BY 1;
```

The output is :

Result Grid					
		Filter Rows:	Export:	Wrap Cell Content:	
	Week	Weekly Digest Rate	Email Open Rate	Email Clickthrough Rate	Reengagement Email Rate
▶	17	0.00	21.28	11.39	5.01
	18	0.00	22.24	10.49	3.83
	19	0.00	22.67	11.13	4.04
	20	0.00	22.64	11.43	4.31
	21	0.00	22.82	9.97	3.69
	22	0.00	21.56	10.66	4.19
	23	0.00	22.34	11.18	4.09
	24	0.00	22.92	10.99	4.48
	25	0.00	21.79	10.54	3.90
	26	0.00	22.22	10.61	4.18
	27	0.00	22.49	11.37	3.90
	28	0.00	22.48	10.77	3.83
	29	0.00	21.71	10.51	3.79
	30	0.00	23.24	10.59	3.88
	31	0.00	23.25	7.66	3.82
	32	0.00	22.85	7.14	3.42
	33	0.00	23.10	7.91	4.26
	34	0.00	23.01	7.67	4.08

The main query selects the week number extracted from the "occurred\_at" timestamp using the `EXTRACT(WEEK FROM occurred_at)` function as "Week".

The query calculates the rate of different email actions for each week:

"Weekly Digest Rate" is the percentage of users who performed the "sent\_weekly\_digest" action out of all users, "Email Open Rate" is the percentage of users who performed the "email\_open" action out of all users, "Email Clickthrough Rate" is the percentage of users who performed the "email\_clickthrough" action out of all users, "Reengagement Email Rate" is the percentage of users who performed the "sent\_reengagement\_email" action out of all users.

The `COUNT(user_id)` counts the total number of users.

The rates are calculated by dividing the count of users who performed the action by the total count of users and multiplying by 100.

The `ROUND()` function is used to round the rates to two decimal places.

The result set is grouped by the week numbers ("Week") and ordered in ascending order.

**TECH STACK USED :**

I have used mysql workbench to run the above sql queries , I have used this because I am familiar to this environment and I learnt how to import large csv files using wizard .

MySQL Workbench provides data modeling, SQL development, and various administration tools for configuration.

I preferred mysql to do this project .

**Insights:**

From this project I learnt to analyze large amount of data .

I learnt coalesce function for the first time which avoids division by zero problem and extract function , round function and used group by clause , where condition.

I have never worked this hard for any project, its my first experience .

**RESULT:**

This project helped me to understand the importance of operational analytics and who it provides insights to the company for its growth . I have learnt about User engagement , user growth , weekly retention , weekly growth , email engagement and throughput . Happy to do this project very challenging one .