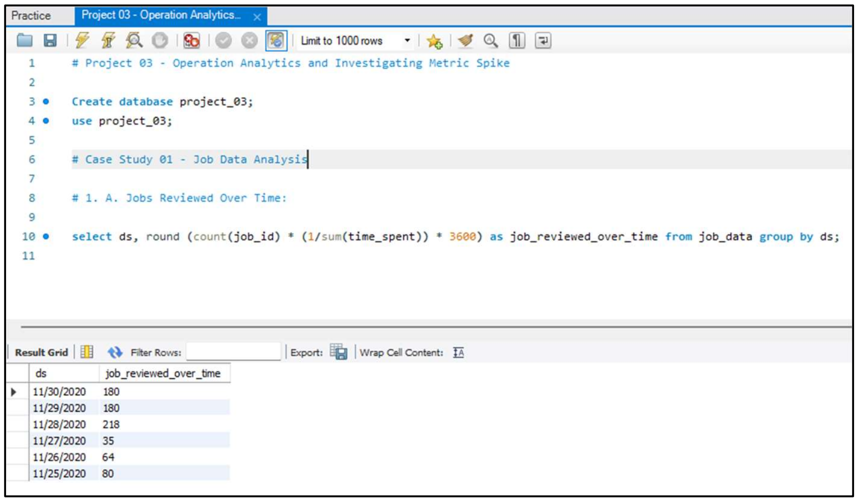
TRAINITY - OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE – PROJECT 03

Case Study 1 – Job Data Analysis:

A. Jobs Reviewed Over Time:

"SQL query to calculate the number of jobs reviewed per hour for each day in November 2020"

Query & Output: -



Practice | Project 03 - Operation Analytics...

Limit to 1000 rows

```sql
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 01 - Job Data Analysis

# 1. A. Jobs Reviewed Over Time:

select ds, round (count(job_id) * (1/sum(time_spent)) * 3600) as job_reviewed_over_time from job_data group by ds;
```

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

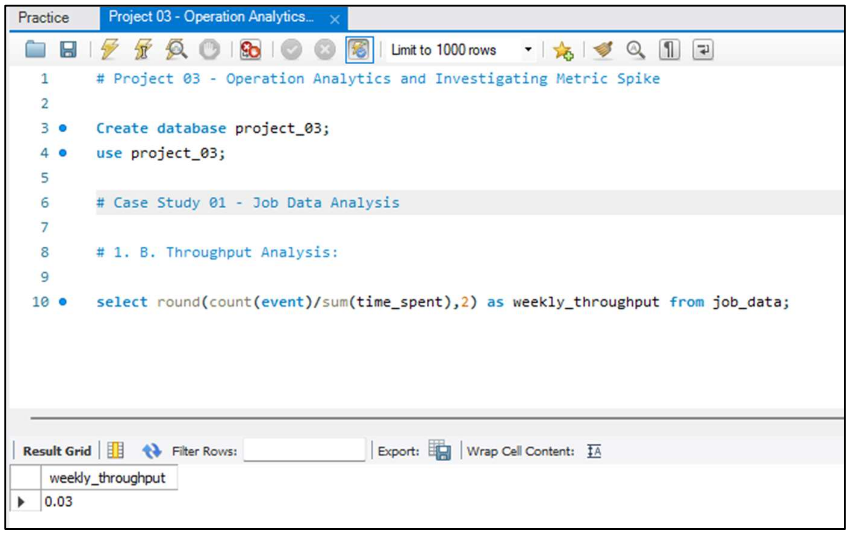
```
ds          job_reviewed_over_time
11/30/2020  180
11/29/2020  180
11/28/2020  218
11/27/2020  35
11/26/2020  64
11/25/2020  80
```

Insight: - From the obtained output it has been seen that the jobs reviewed per hour is high on **28/11/2020**.

B. Throughput Analysis:

"SQL query to calculate the 7-day rolling average of throughput"

Query & Output: - (Weekly Throughput)



Practice | Project 03 - Operation Analytics...

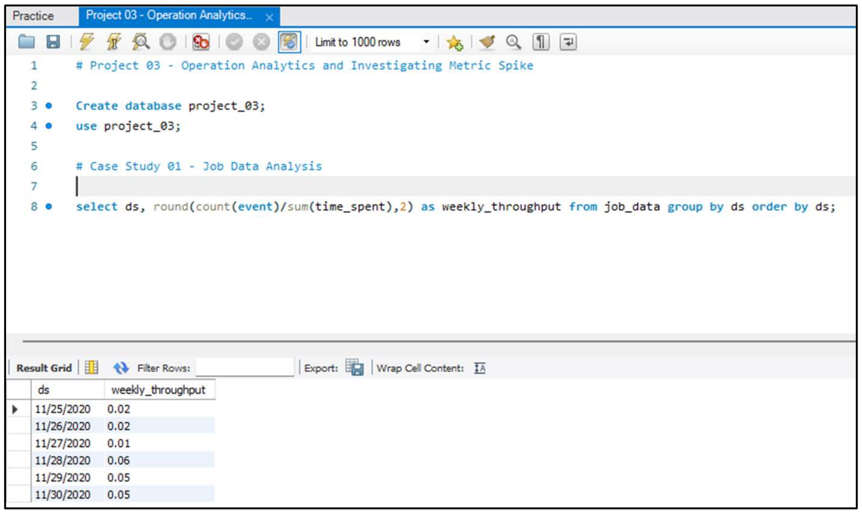
Limit to 1000 rows

```sql
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 01 - Job Data Analysis

# 1. B. Throughput Analysis:

select round(count(event)/sum(time_spent),2) as weekly_throughput from job_data;
```

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

```
weekly_throughput
0.03
```

Query & Output: - (Daily Throughput)



```sql
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 01 - Job Data Analysis

select ds, round(count(event)/sum(time_spent),2) as weekly_throughput from job_data group by ds order by ds;
```

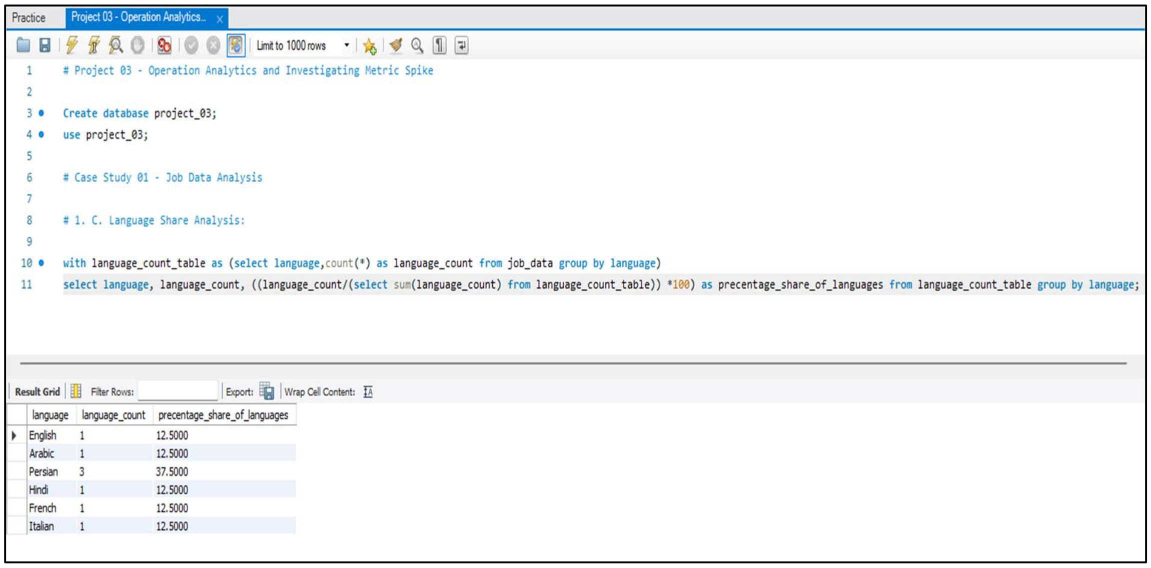
```
ds          weekly_throughput
11/25/2020  0.02
11/26/2020  0.02
11/27/2020  0.01
11/28/2020  0.06
11/29/2020  0.05
11/30/2020  0.05
```

Insight: - The weekly throughput is giving the weekly average only whereas the daily metric is giving a throughput for day-to-day update which will be more helpful for business operation and making decisions.

C. Language Share Analysis:

"SQL query to calculate the percentage share of each language over the last 30 days"

Query & Output: -



```sql
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 01 - Job Data Analysis

# 1. C. Language Share Analysis:

with language_count_table as (select language,count(*) as language_count from job_data group by language)
select language, language_count, ((language_count/(select sum(language_count) from language_count_table)) *100) as precentage_share_of_languages from language_count_table group by language;
```

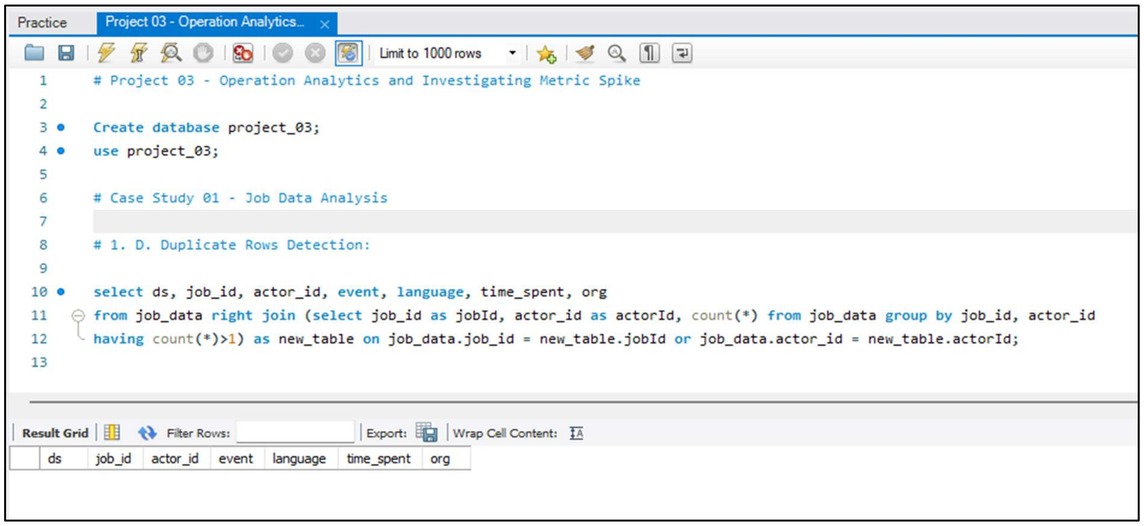
```
language  language_count  precentage_share_of_languages
English   1               12.5000
Arabic    1               12.5000
Persian   3               37.5000
Hindi     1               12.5000
French    1               12.5000
Italian   1               12.5000
```

Insight: - From the obtained output it has been seen that the language 'Persian' has maximum share percentage (35%) over all the last 30 days.

D. Duplicate Rows Detection:

"SQL query to display duplicate rows from the job_data table"

Query & Output: -



Practice | Project 03 - Operation Analytics...

Limit to 1000 rows

```
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 01 - Job Data Analysis

# 1. D. Duplicate Rows Detection:

select ds, job_id, actor_id, event, language, time_spent, org
from job_data right join (select job_id as jobId, actor_id as actorId, count(*) from job_data group by job_id, actor_id
having count(*)>1) as new_table on job_data.job_id = new_table.jobId or job_data.actor_id = new_table.actorId;
```

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

ds	job_id	actor_id	event	language	time_spent	org

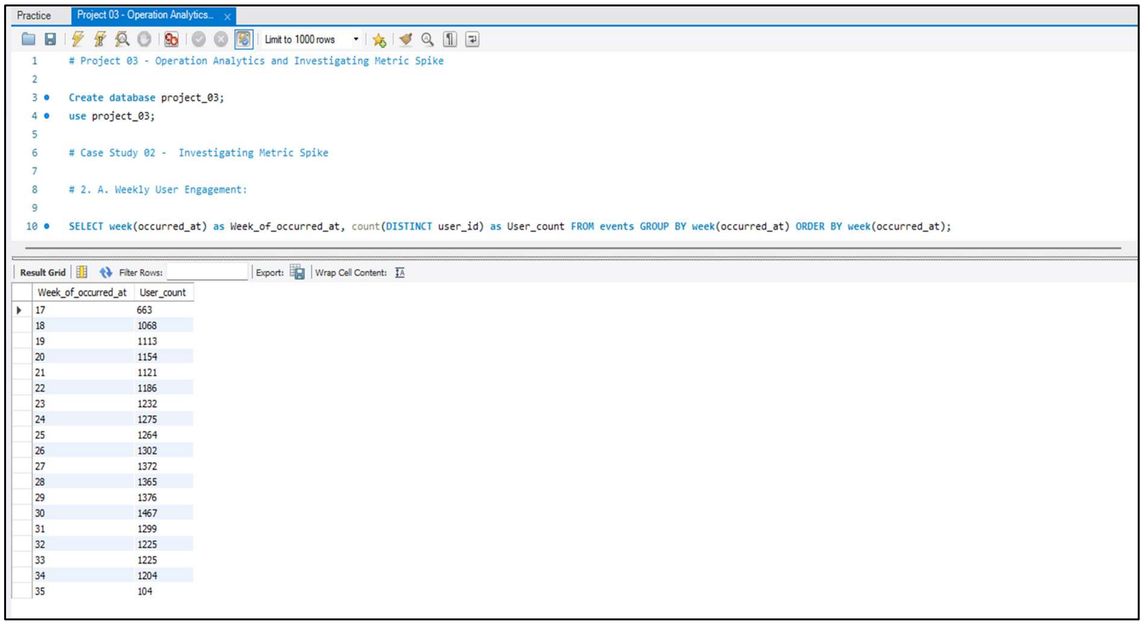
Insight: - Since the columns job_id & actor_id are unique values and there is no repetition of the same in the given data there is no duplicates were obtained in the output.

Case Study 2 – Investigating Metric Spike:

A. Weekly User Engagement:

"SQL query to calculate the weekly user engagement"

Query & Output: -



Practice | Project 03 - Operation Analytics...

Limit to 1000 rows

```
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 02 -  Investigating Metric Spike

# 2. A. Weekly User Engagement:

SELECT week(occurred_at) as Week_of_occurred_at, count(DISTINCT user_id) as User_count FROM events GROUP BY week(occurred_at) ORDER BY week(occurred_at);
```

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

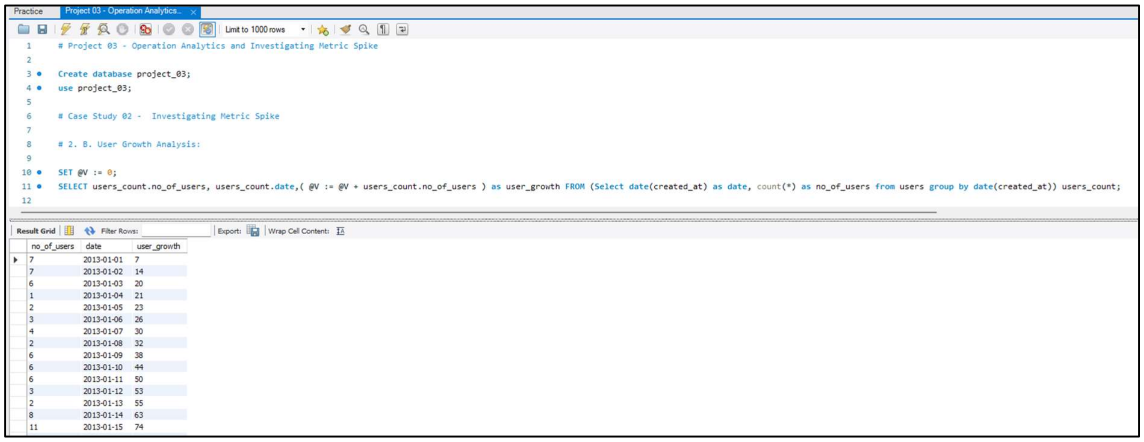
Week_of_occurred_at	User_count
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

Insight: - From the obtained output we can see the user engagement on a weekly basis.

B. User Growth Analysis:

"SQL query to calculate the user growth for the product"

Query & Output: -



```
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 02 -  Investigating Metric Spike

# 2. B. User Growth Analysis:

SET @V := 0;
SELECT users_count.no_of_users, users_count.date,( @V := @V + users_count.no_of_users ) as user_growth FROM (Select date(created_at) as date, count(*) as no_of_users from users group by date(created_at)) users_count;

```

no_of_users	date	user_growth
7	2013-01-01	7
7	2013-01-02	14
6	2013-01-03	20
1	2013-01-04	21
2	2013-01-05	23
3	2013-01-06	26
4	2013-01-07	30
2	2013-01-08	32
6	2013-01-09	38
6	2013-01-10	44
6	2013-01-11	50
3	2013-01-12	53
2	2013-01-13	55
8	2013-01-14	63
11	2013-01-15	74

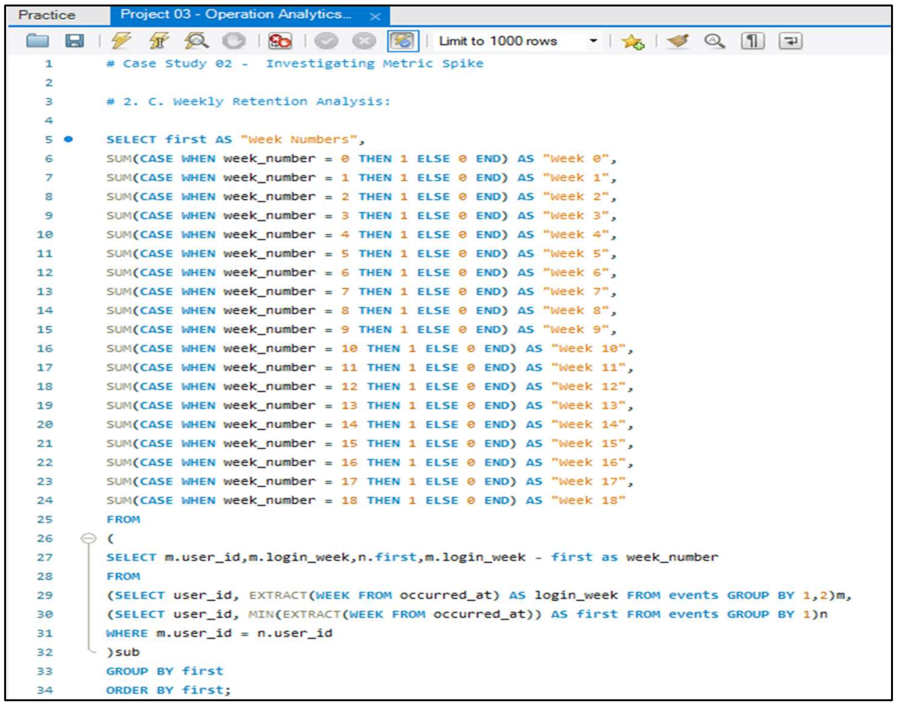
Note: - Since the output is very large, I have inserted only a front page of the output. However, the query has extracted the required output.

Insight: - From the obtained output we can see the range on how the user growth on this platform is getting increase daily.

C. Weekly Retention Analysis:

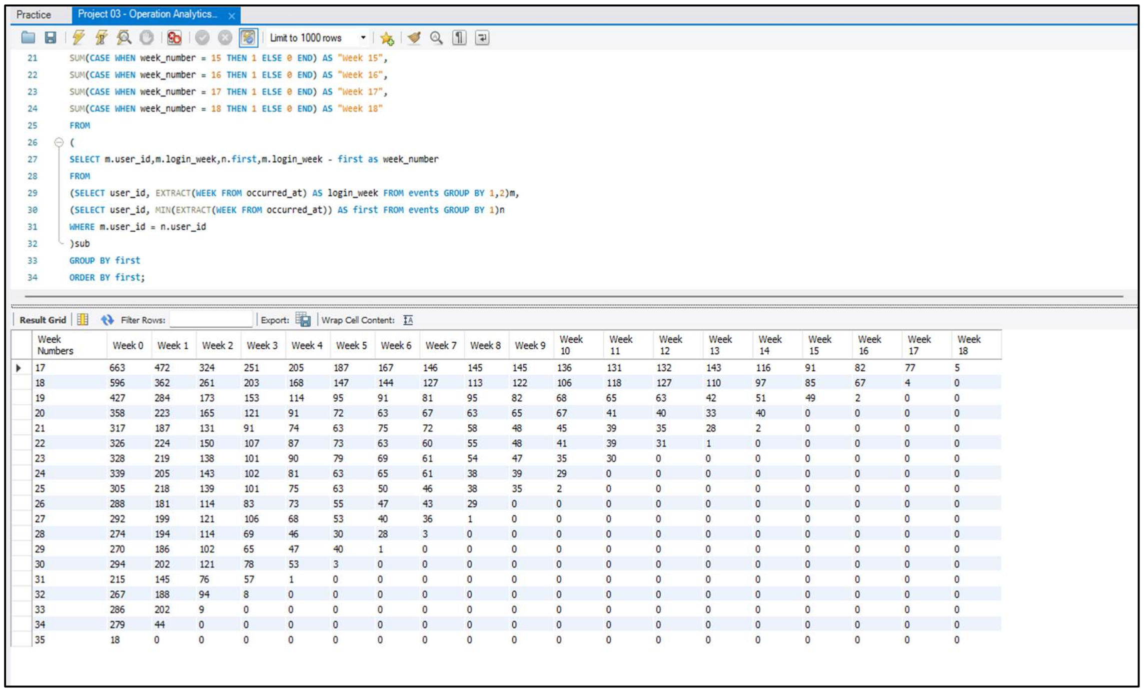
"SQL query to calculate the weekly retention of users based on their sign-up cohort"

Query: -



```
# Case Study 02 -  Investigating Metric Spike

# 2. C. Weekly Retention Analysis:

SELECT first AS "Week Numbers",
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 m.user_id,m.login_week,n.first,m.login_week - first as week_number
FROM
(SELECT user_id, EXTRACT(WEEK FROM occurred_at) AS login_week FROM events GROUP BY 1,2)m,
(SELECT user_id, MIN(EXTRACT(WEEK FROM occurred_at)) AS first FROM events GROUP BY 1)n
WHERE m.user_id = n.user_id
)sub
GROUP BY first
ORDER BY first;
```

<u>Output: -</u>



```sql
    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 m.user_id,m.login_week,n.first,m.login_week - first as week_number
    FROM
    (SELECT user_id, EXTRACT(WEEK FROM occurred_at) AS login_week FROM events GROUP BY 1,2)m,
    (SELECT user_id, MIN(EXTRACT(WEEK FROM occurred_at)) AS first FROM events GROUP BY 1)n
    WHERE m.user_id = n.user_id
    )sub
    GROUP BY first
    ORDER BY first;
```

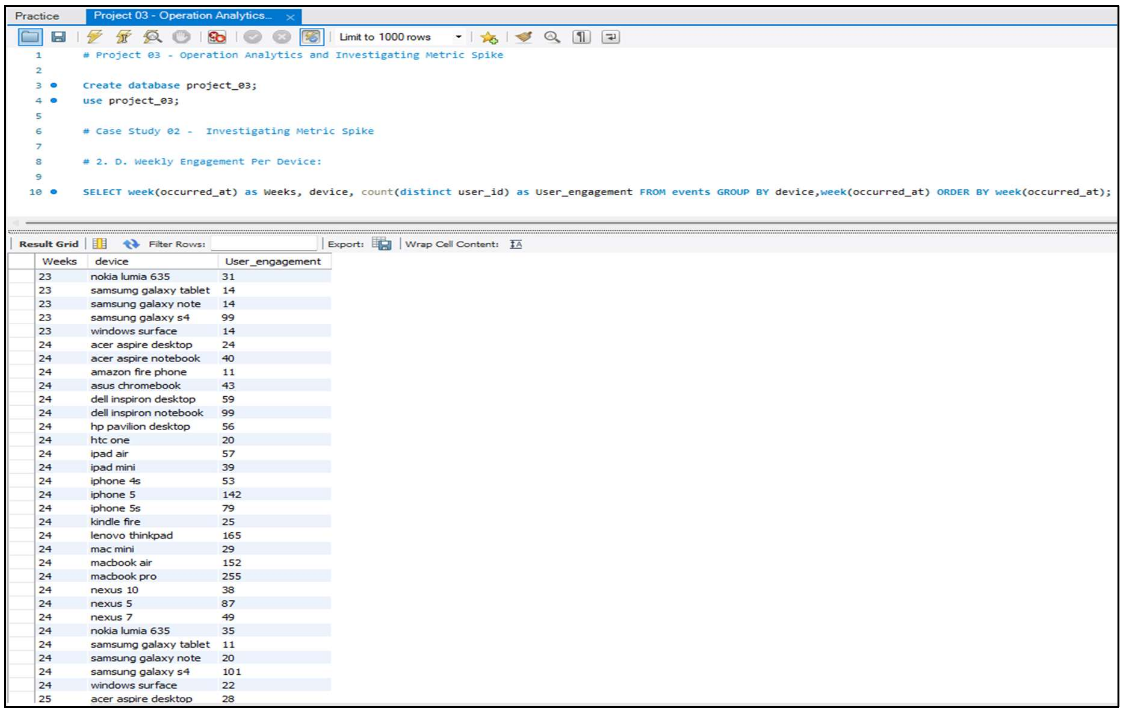
Week Numbers	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18
17	663	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77	5
18	596	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4	0
19	427	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0	0
20	358	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0	0
21	317	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0	0
22	326	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0
23	328	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0
24	339	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0
25	305	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0
26	288	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0
27	292	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0
28	274	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0
29	270	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0
30	294	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0
31	215	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	267	188	94	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	286	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	279	44	0	0	0	0	0	0	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	0	0	0	0	0	0

Insight: - From the obtained output we can see retention of users from their activated date till the present date as per the given the data set.

D. Weekly Engagement Per Device:

"SQL query to calculate the weekly engagement per device"

<u>Query & Output: -</u>



```sql
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 02 -  Investigating Metric Spike

# 2. D. Weekly Engagement Per Device:

SELECT week(occurred_at) as Weeks, device, count(distinct user_id) as User_engagement FROM events GROUP BY device,week(occurred_at) ORDER BY week(occurred_at);
```

Weeks	device	User_engagement
23	nokia lumia 635	31
23	samsumg galaxy tablet	14
23	samsung galaxy note	14
23	samsung galaxy s4	99
23	windows surface	14
24	acer aspire desktop	24
24	acer aspire notebook	40
24	amazon fire phone	11
24	asus chromebook	43
24	dell inspiron desktop	59
24	dell inspiron notebook	99
24	hp pavilion desktop	56
24	htc one	20
24	ipad air	57
24	ipad mini	39
24	iphone 4s	53
24	iphone 5	142
24	iphone 5s	79
24	kindle fire	25
24	lenovo thinkpad	165
24	mac mini	29
24	macbook air	152
24	macbook pro	255
24	nexus 10	38
24	nexus 5	87
24	nexus 7	49
24	nokia lumia 635	35
24	samsumg galaxy tablet	11
24	samsung galaxy note	20
24	samsung galaxy s4	101
24	windows surface	22
25	acer aspire desktop	28

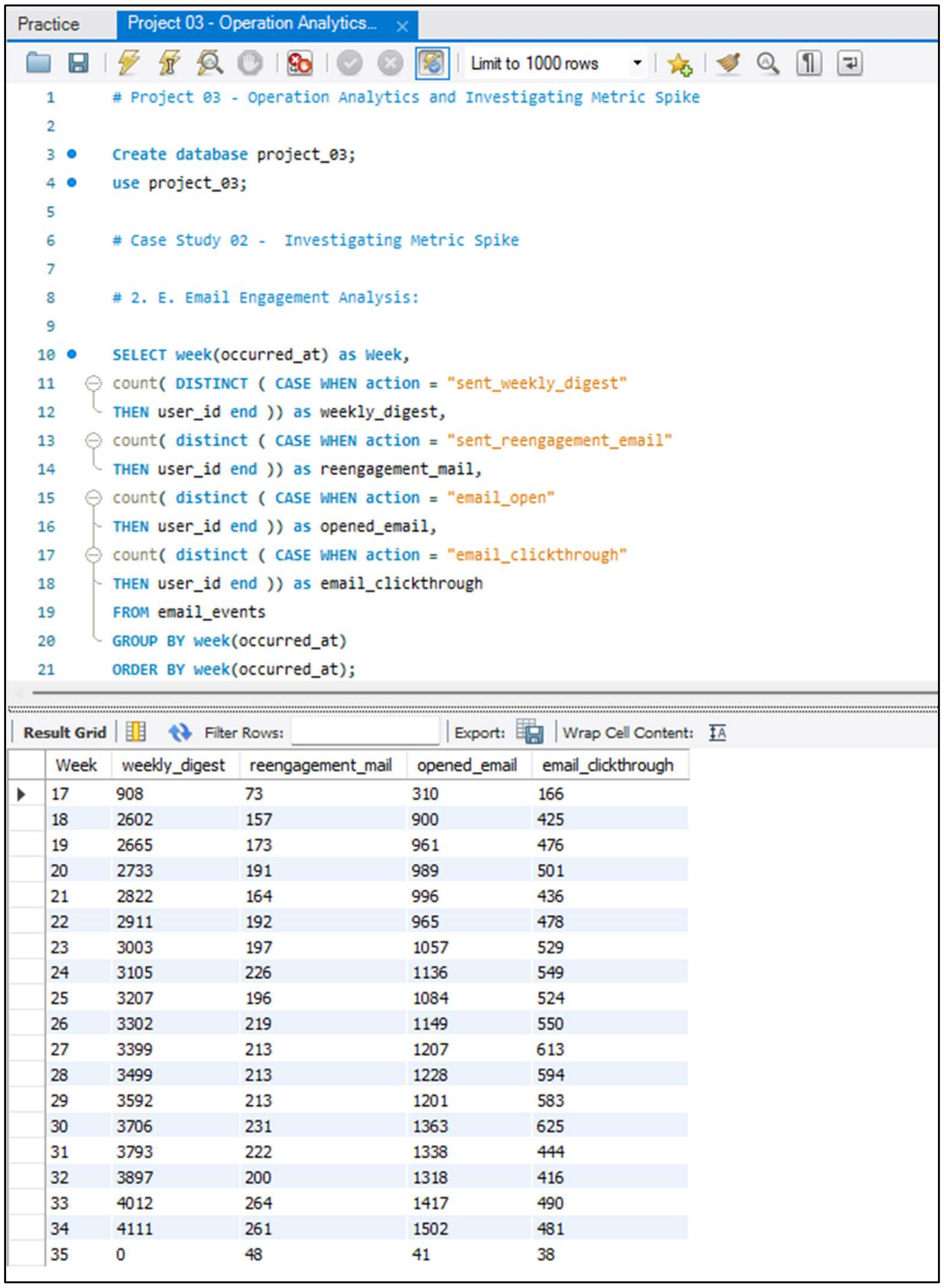
Note: - Since the output is very large, I have inserted only a front page of the output. However, the query has extracted the required output.

Insight: - From the obtained output we can see the user engagement on weekly basis per device.

E. Email Engagement Analysis:

"SQL query to calculate the email engagement metrics"

Query & Output: -



Practice | Project 03 - Operation Analytics...

Limit to 1000 rows

```
# Project 03 - Operation Analytics and Investigating Metric Spike

Create database project_03;
use project_03;

# Case Study 02 -  Investigating Metric Spike

# 2. E. Email Engagement Analysis:

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"
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"
THEN user_id end )) as email_clickthrough
FROM email_events
GROUP BY week(occurred_at)
ORDER BY week(occurred_at);
```

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

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
32	3897	200	1318	416
33	4012	264	1417	490
34	4111	261	1502	481
35	0	48	41	38

Insight: - From the obtained output we can see email engagement metrices on weekly basis.

FINAL SUMMARY: -

Sr No	Question	Answer
1	Project Description	Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. This analysis helps identify areas for improvement within the company. As a Data Analyst, you'll work closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect. One of the key aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales. As a Data Analyst, you'll need to answer these questions daily, making it crucial to understand how to investigate these metric spikes. In this project, you'll take on the role of a Lead Data Analyst at a company like Microsoft. You'll be provided with various datasets and tables, and your task will be to derive insights from this data to answer questions posed by different departments within the company. Your goal is to use your advanced SQL skills to analyze the data and provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.
2	Approach	The steps that I approached on this project are explained below. 1. Read the data set given and understand the details of the data in it. 2. Created a new data base to create and store tables as per the given data set. 3. Created job_data table as per the data set for case study 1 and inserted the values 4. Started to work on the different questions on the case study 1 and extracted the answers. 5. Created three tables’ users, events, email_events as per the given data set. 6. Started to work on the different questions on the case study 2 and extracted the answers.
3	Tech-Stack Used	MySQL Workbench 8.0.36

4	Insights	Through this project, learned on how to approach the project, understanding the given data sets and questions, creating the tables as per the data sets, applying different functions and their sequence etc.
5	Result	This project has helped us to improve our knowledge on some advanced My SQL functions.