Operation-Analytics-and-Investigating-Metric-Spike

Project Overview:

This project involves analysing datasets provided by the company to derive insights and answer questions posed by various departments such as operations, support, and marketing. The goal is to use data analysis to predict the overall growth or decline of the company's fortunes. This includes improving automation, enhancing understanding between cross-functional teams, and optimizing workflows.

Case Study 1: Job Data Analysis: In this case study, we'll work with a table named job_data. This table contains information such as job IDs, actor IDs, event types, time spent on tasks, organization details, and dates. We'll analyze this data to gain insights into job-related activities and performance metrics.

Case Study 2: Investigating Metric Spikes: For this case study, we have three tables: users, events, and email_events. These tables contain information about users, their interactions (events), and specific email-related events. We'll analyze these datasets to investigate sudden changes or spikes in key metrics such as user engagement, email open rates, etc.

The insights derived from these analyses will be instrumental in helping the company make informed decisions, improve operational efficiency, and drive overall growth.

Approach:

- Understand the provided datasets and their schema.
- Identify key metrics to analyse, such as user engagement, email open rates, etc.
- Write SQL queries to analyse trends in the metrics over time.
- Detect spikes or sudden changes in the metrics using statistical techniques or threshold-based approaches.
- Investigate the causes of spikes by examining related events or user behaviours.
- Provide insights and recommendations based on the analysis to relevant stakeholders.

Tech-Stack Used:

- MySQL Workbench: Used for writing and executing SQL queries on the provided datasets.
- Google Drive: Used for storing and sharing the project report in PDF format.

To investigate metric spikes in the provided case study, we'll need to perform advanced SQL queries on the given tables **users**, **events**, and **email_events**. Here's a general approach we can take:

1) **Identify the Metric to Investigate**: Determine which key metric we need to investigate for any sudden changes or spikes. This could be user engagement, email open rates, login frequency, etc.
2) **Analyze Data Trends**: Use SQL queries to analyze trends in the metric over time. This involves aggregating and summarizing data from the relevant tables, possibly using functions like COUNT, SUM, AVG, etc., and grouping by time intervals (e.g., day, week).
3) **Detect Spikes**: Look for sudden changes or spikes in the metric using statistical techniques or threshold-based approaches. This may involve comparing current values to historical averages or identifying outliers.
4) **Investigate Causes**: Once a spike is detected, delve deeper into the data to understand the underlying causes. This could involve examining related events or user behaviours leading up to the spike.
5) **Provide Insights**: Based on the analysis, provide insights and recommendations to relevant stakeholders within the company. This could include suggestions for further investigation, potential actions to address issues, or strategies to capitalize on positive trends.

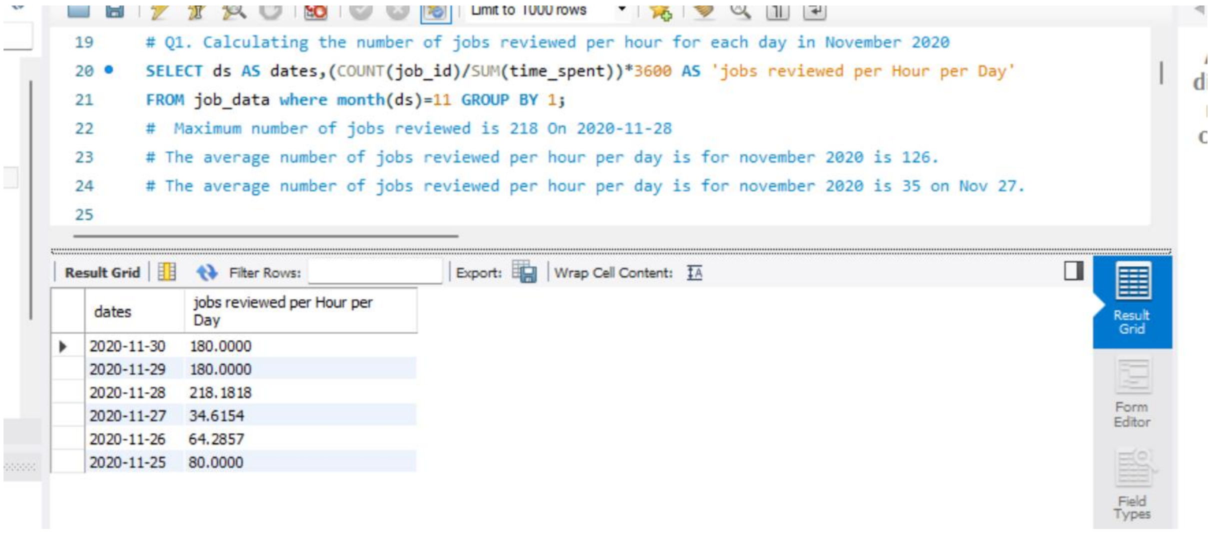
Let's start by writing SQL queries to analyse and investigate metric spikes in the **events** and **email_events** tables. We'll focus on specific metrics such as user engagement, email open rates, etc., depending on the requirements provided in the case study.

Case study 1 (Operational Analytics)

Q1. Calculating the number of jobs reviewed per hour for each day in November 2020

Syntax used:

SELECT ds AS dates,(COUNT(job_id)/SUM(time_spent))*3600 AS 'jobs reviewed per Hour per Day' FROM job_data where month(ds)=11 GROUP BY 1;



```
# Q1. Calculating the number of jobs reviewed per hour for each day in November 2020
SELECT ds AS dates,(COUNT(job_id)/SUM(time_spent))*3600 AS 'jobs reviewed per Hour per Day'
FROM job_data where month(ds)=11 GROUP BY 1;
#  Maximum number of jobs reviewed is 218 On 2020-11-28
# The average number of jobs reviewed per hour per day is for november 2020 is 126.
# The average number of jobs reviewed per hour per day is for november 2020 is 35 on Nov 27.
```

dates	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

Insights:

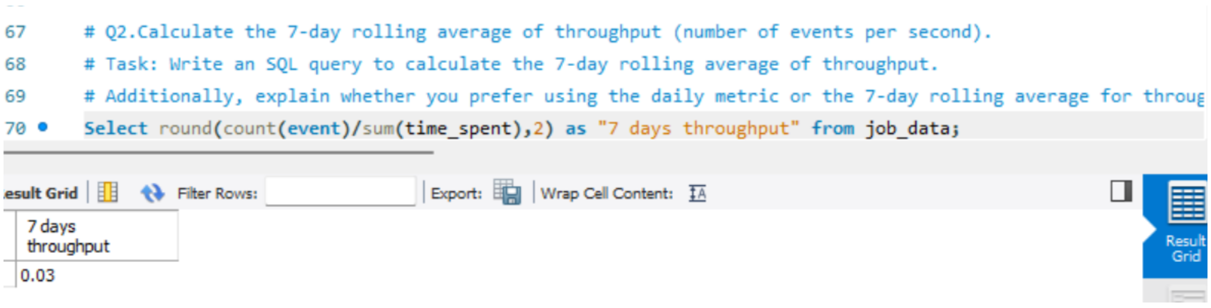
Maximum number of jobs reviewed is 218 On 2020-11-28

The average number of jobs reviewed per hour per day is for November 2020 is 126.

The average number of jobs reviewed per hour per day is for November 2020 is 35 on Nov 27.

Q2. Calculate the 7-day rolling average of throughput (number of events per second).
Task: 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?
Syntax used: To find weekly throughput

Select round(count(event)/sum(time_spent),2) as "7 days throughput" from job_data;



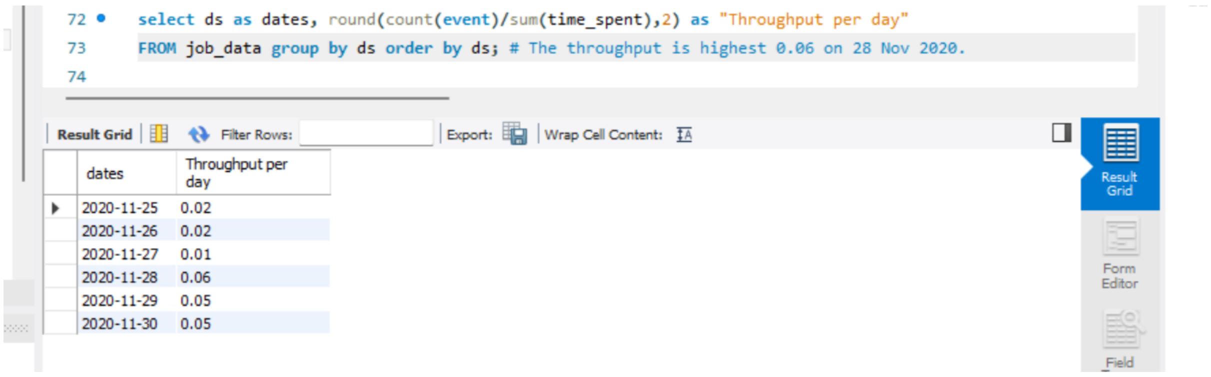
```
# Q2.Calculate the 7-day rolling average of throughput (number of events per second).
# Task: Write an SQL query to calculate the 7-day rolling average of throughput.
# Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throug
Select round(count(event)/sum(time_spent),2) as "7 days throughput" from job_data;
```

7 days throughput
0.03

Insights: 7 day throughput is 0.03

Syntax used: to find throughput per day

select ds as dates, round(count(event)/sum(time_spent),2) as "Throughput per day" FROM job_data group by ds order by ds;



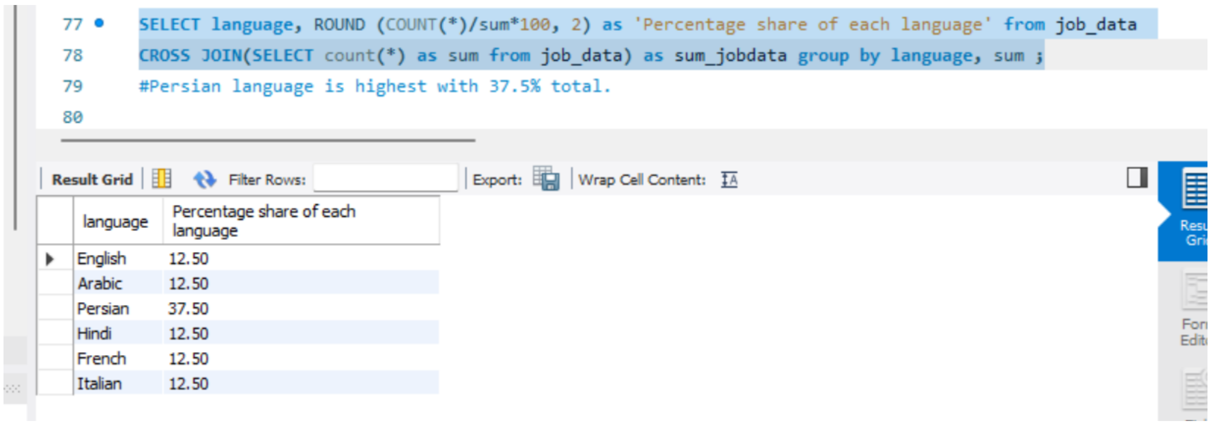
```
select ds as dates, round(count(event)/sum(time_spent),2) as "Throughput per day"
FROM job_data group by ds order by ds; # The throughput is highest 0.06 on 28 Nov 2020.
```

dates	Throughput per day
2020-11-25	0.02
2020-11-26	0.02
2020-11-27	0.01
2020-11-28	0.06
2020-11-29	0.05
2020-11-30	0.05

Insights: The throughput is highest 0.06 on 28 Nov 2020

Q3. Calculate the percentage share of each language in the last 30 days.
Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

Syntax used:
SELECT language, ROUND (COUNT(*)/sum*100, 2) as 'Percentage share of each language' from job_data
CROSS JOIN(SELECT count(*) as sum from job_data) as sum_jobdata group by language, sum ;



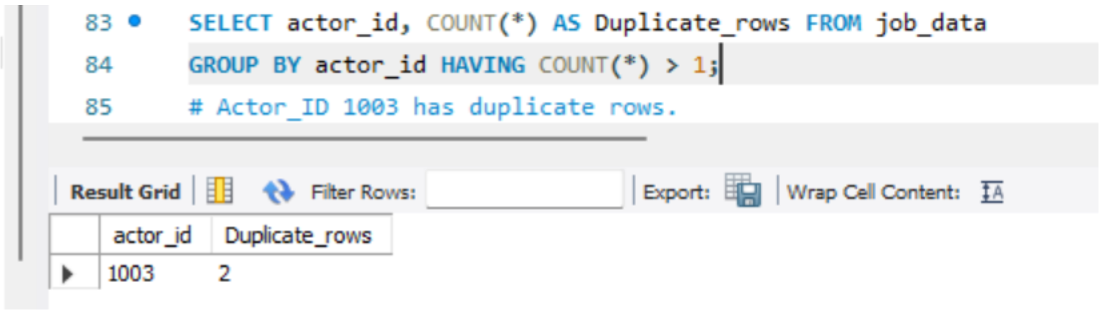
```
SELECT language, ROUND (COUNT(*)/sum*100, 2) as 'Percentage share of each language' from job_data
CROSS JOIN(SELECT count(*) as sum from job_data) as sum_jobdata group by language, sum ;
#Persian language is highest with 37.5% total.
```

language	Percentage share of each language
English	12.50
Arabic	12.50
Persian	37.50
Hindi	12.50
French	12.50
Italian	12.50

Insights: Persian language is highest with 37.5% total.

Q4. Identify duplicate rows in the data.
Task: Write an SQL query to display duplicate rows from the job_data table.

Syntax used:
SELECT actor_id, COUNT(*) AS Duplicate_rows FROM job_data
GROUP BY actor_id HAVING COUNT(*) > 1;



```
SELECT actor_id, COUNT(*) AS Duplicate_rows FROM job_data
GROUP BY actor_id HAVING COUNT(*) > 1;
# Actor_ID 1003 has duplicate rows.
```

actor_id	Duplicate_rows
1003	2

Insights: Actor_ID 1003 has duplicate rows.

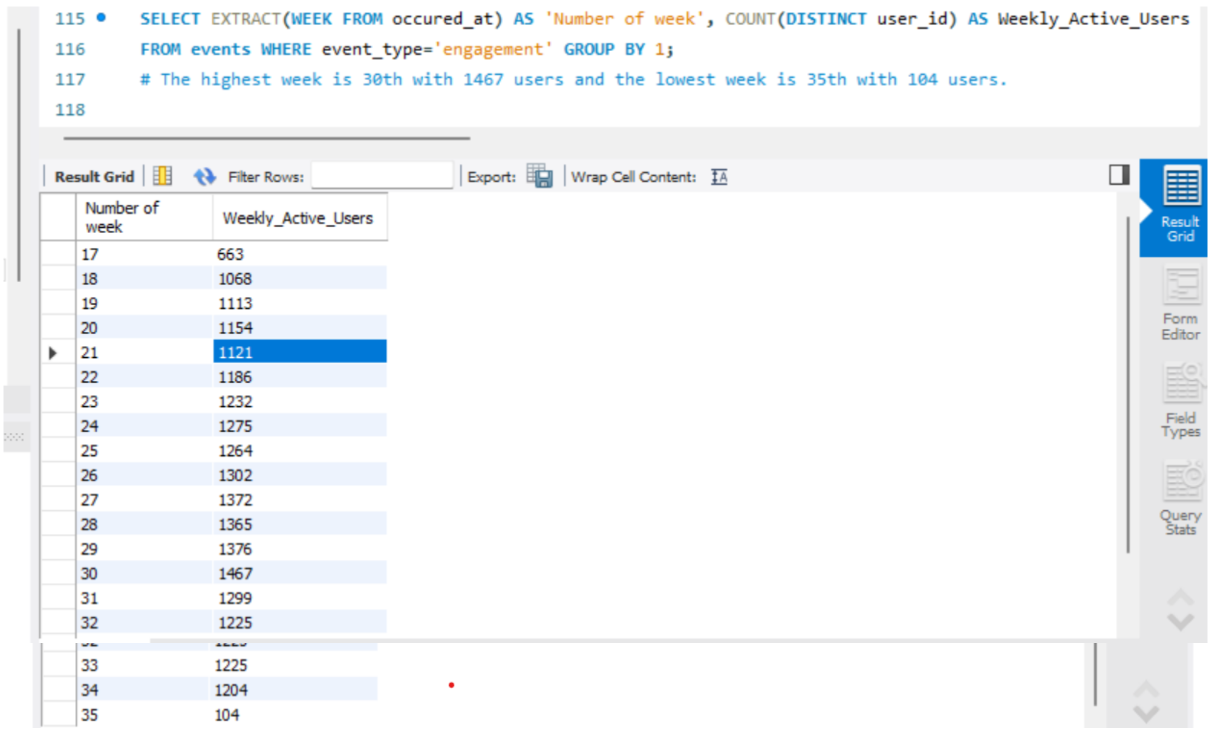
****** Meanwhile the format of date time was changed as it was in the varchar format. So converting it into DATETIME OR TIMESTAMP format**

```
select * from users;
alter table users Add COLUMN temp_created_at datetime;
UPDATE users SET temp_created_at = STR_TO_DATE(created_at, '%d-%m-%Y %H:%i');
ALTER TABLE users DROP COLUMN created_at;
ALTER TABLE users CHANGE COLUMN temp_created_at created_at DATETIME;

desc events;
select * from events;
alter table events Add COLUMN temp_occured_at datetime;
UPDATE events SET temp_occured_at = STR_TO_DATE(occured_at, '%d-%m-%Y %H:%i');
ALTER TABLE events DROP COLUMN occured_at;
ALTER TABLE events CHANGE COLUMN temp_occured_at occured_at DATETIME;

select * from email_events;
alter table email_events Add COLUMN temp_occured_at datetime;
UPDATE email_events SET temp_occured_at = STR_TO_DATE(occurred_at, '%d-%m-%Y
%H:%i');
ALTER TABLE email_events DROP COLUMN occurred_at;
ALTER TABLE email_events CHANGE COLUMN temp_occured_at occured_at DATETIME;
```

Case Study 2 (Investigating Metric Spike)

Q5. Measure the activeness of users on a weekly basis.
Task: Write an SQL query to calculate the weekly user engagement.
Syntax used:

SELECT EXTRACT(WEEK FROM occured_at) AS 'Number of week', COUNT(DISTINCT user_id) AS Weekly_Active_Users
FROM events WHERE event_type='engagement' GROUP BY 1;



```
SELECT EXTRACT(WEEK FROM occured_at) AS 'Number of week', COUNT(DISTINCT user_id) AS Weekly_Active_Users
FROM events WHERE event_type='engagement' GROUP BY 1;
# The highest week is 30th with 1467 users and the lowest week is 35th with 104 users.
```

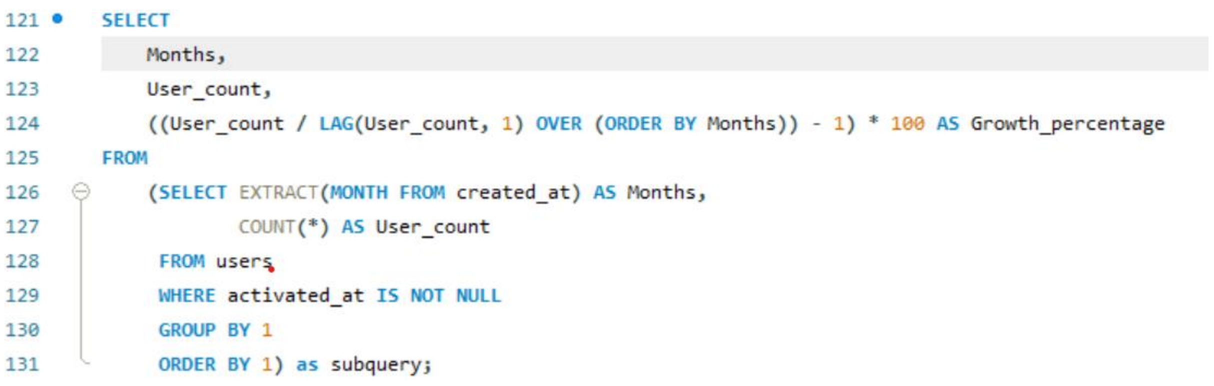
Number of week	Weekly_Active_Users
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

Insights: The highest week is 30th with 1467 users and the lowest week is 35th with 104 users.

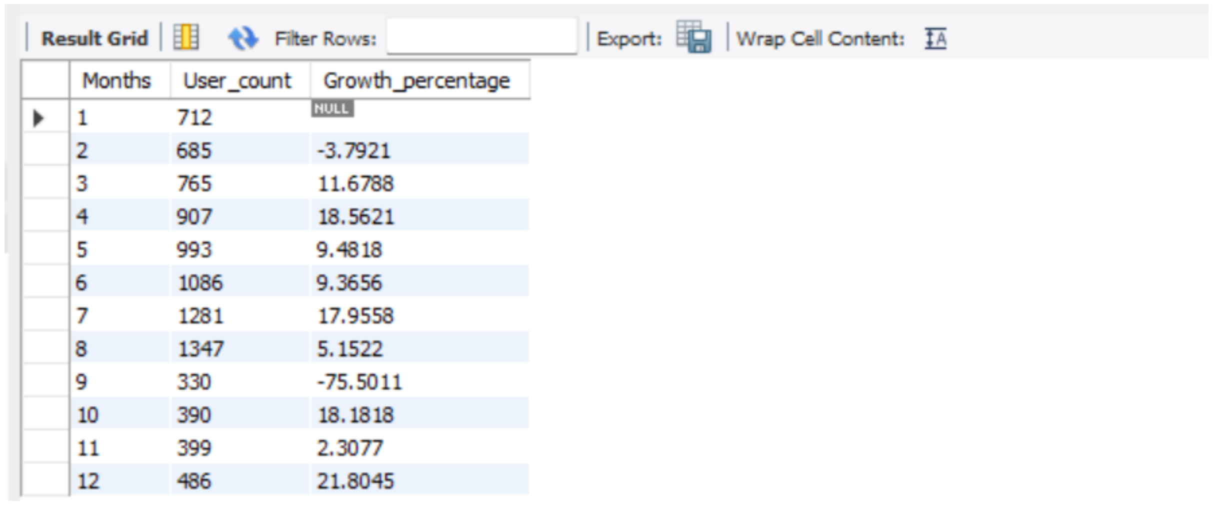
Q6. Analyse the growth of users over time for a product.
Task: Write an SQL query to calculate the user growth for the product.

Syntax used:

```
SELECT
   Months,
   User_count,
   ((User_count / LAG(User_count, 1) OVER (ORDER BY Months)) - 1) * 100 AS
Growth_percentage
FROM
   (SELECT EXTRACT(MONTH FROM created_at) AS Months,
         COUNT(*) AS User_count
    FROM users
    WHERE activated_at IS NOT NULL
    GROUP BY 1
    ORDER BY 1) as subquery;
```



```
SELECT
    Months,
    User_count,
    ((User_count / LAG(User_count, 1) OVER (ORDER BY Months)) - 1) * 100 AS Growth_percentage
FROM
    (SELECT EXTRACT(MONTH FROM created_at) AS Months,
            COUNT(*) AS User_count
     FROM users
     WHERE activated_at IS NOT NULL
     GROUP BY 1
     ORDER BY 1) as subquery;
```



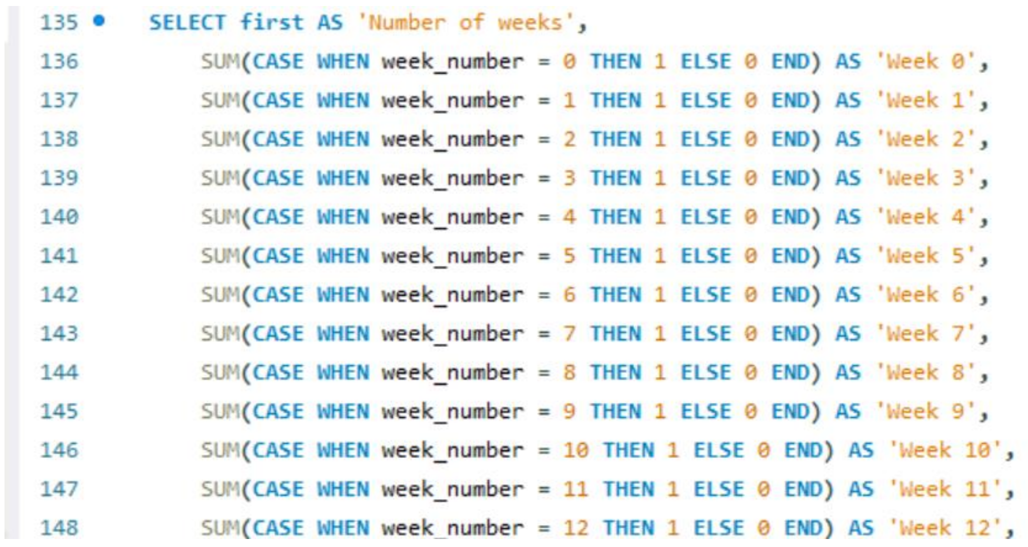
Months	User_count	Growth_percentage
1	712	NULL
2	685	-3.7921
3	765	11.6788
4	907	18.5621
5	993	9.4818
6	1086	9.3656
7	1281	17.9558
8	1347	5.1522
9	330	-75.5011
10	390	18.1818
11	399	2.3077
12	486	21.8045

Insights: There was a positive increase in the percentage growth in the users from JAN TO APRIL and then fluctuating.

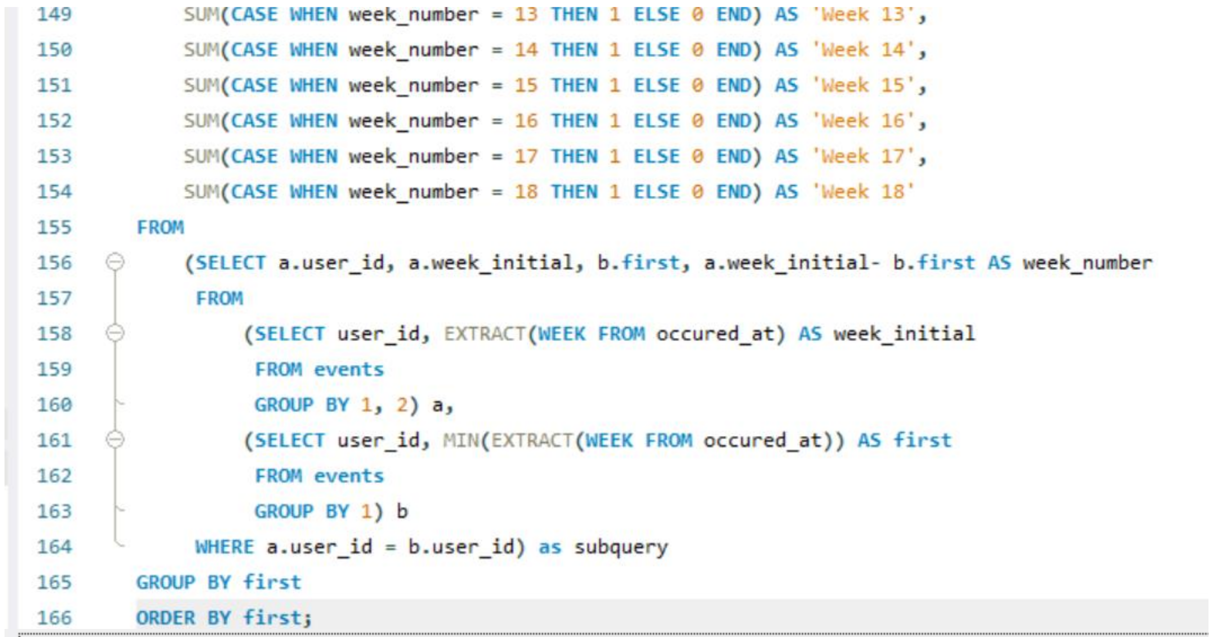
Q7. Analyse the retention of users on a weekly basis after signing up for a product.
Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Syntax used:

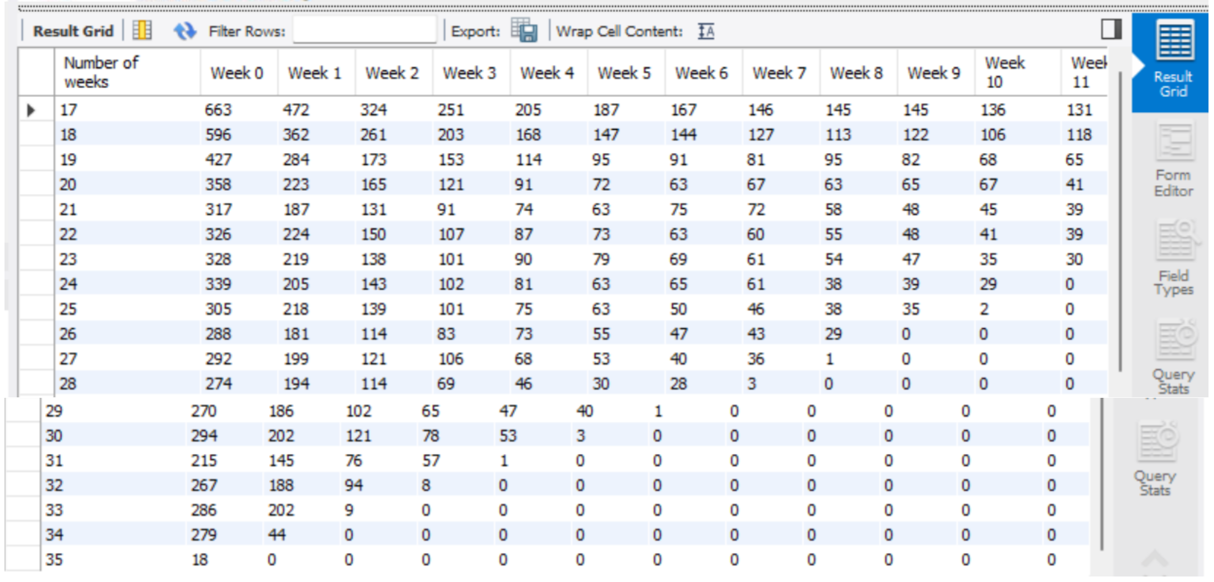
```
SELECT first AS 'Number of weeks',
   SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS 'Week 0',
   SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS 'Week 1',
   SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS 'Week 2',
   SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS 'Week 3',
   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 a.user_id, a.week_initial, b.first, a.week_initial- b.first AS week_number
    FROM
       (SELECT user_id, EXTRACT(WEEK FROM occured_at) AS week_initial
        FROM events
        GROUP BY 1, 2) a,
       (SELECT user_id, MIN(EXTRACT(WEEK FROM occured_at)) AS first
        FROM events
        GROUP BY 1) b
    WHERE a.user_id = b.user_id) as subquery
GROUP BY first
ORDER BY first;
```



```sql
SELECT first AS 'Number of weeks',
       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',
```

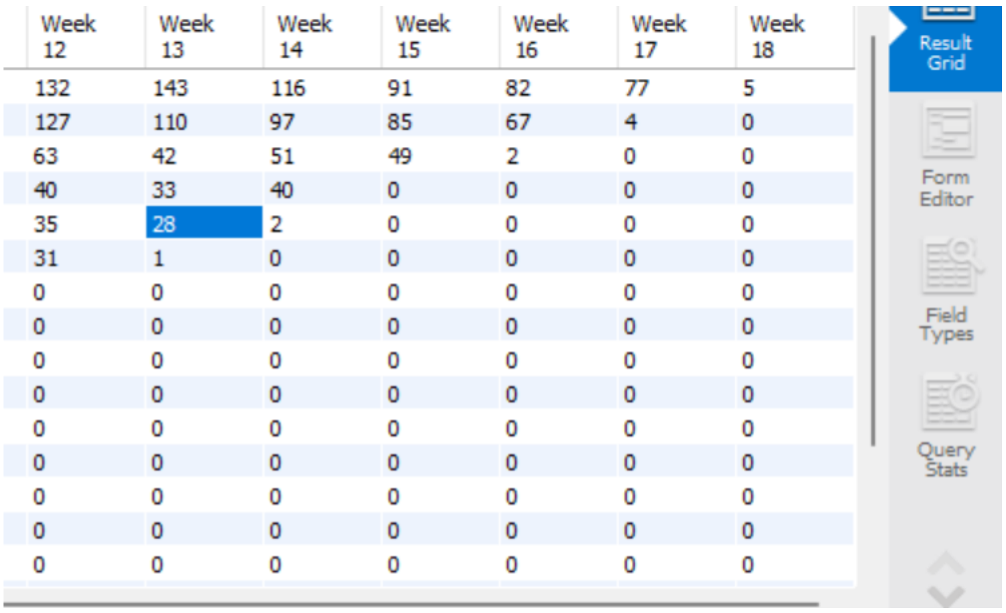


```sql
       SUM(CASE WHEN week_number = 13 THEN 1 ELSE 0 END) AS 'Week 13',
       SUM(CASE WHEN week_number = 14 THEN 1 ELSE 0 END) AS 'Week 14',
       SUM(CASE WHEN week_number = 15 THEN 1 ELSE 0 END) AS 'Week 15',
       SUM(CASE WHEN week_number = 16 THEN 1 ELSE 0 END) AS 'Week 16',
       SUM(CASE 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 a.user_id, a.week_initial, b.first, a.week_initial- b.first AS week_number
     FROM
        (SELECT user_id, EXTRACT(WEEK FROM occured_at) AS week_initial
         FROM events
         GROUP BY 1, 2) a,
        (SELECT user_id, MIN(EXTRACT(WEEK FROM occured_at)) AS first
         FROM events
         GROUP BY 1) b
     WHERE a.user_id = b.user_id) as subquery
GROUP BY first
ORDER BY first;
```



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

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



Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18
132	143	116	91	82	77	5
127	110	97	85	67	4	0
63	42	51	49	2	0	0
40	33	40	0	0	0	0
35	28	2	0	0	0	0
31	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

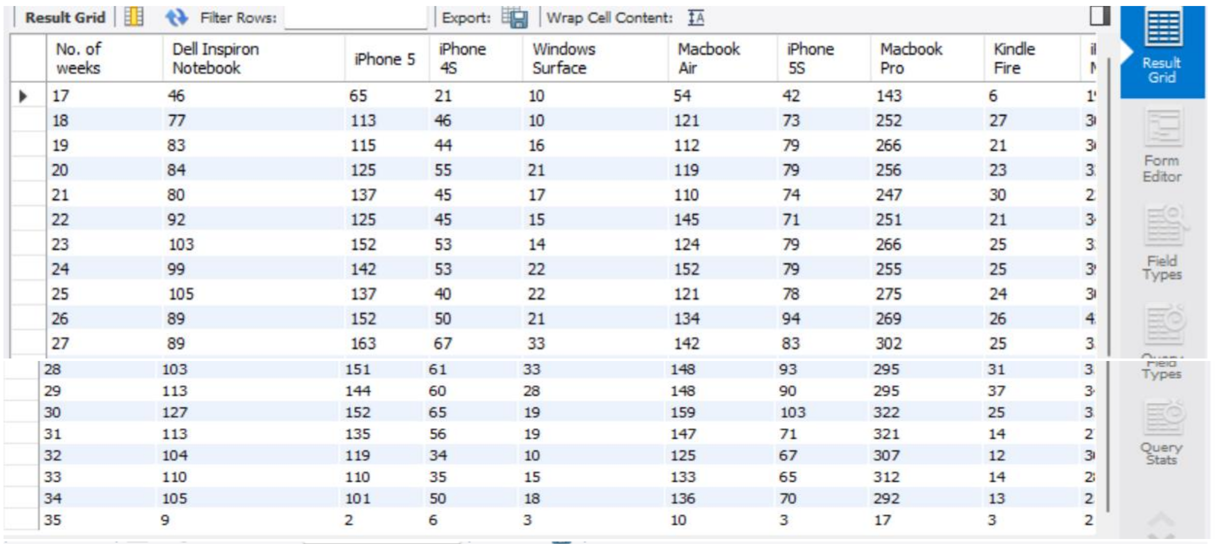
Insights: It is observed that once the customers sign-up there is a drastic drop in the weekly retention of customers. Necessary and effective strategies should be adopted to keep to customers engaged.

Q8. Measure the activeness of users on a weekly basis per device.
Task: Write an SQL query to calculate the weekly engagement per device.

Syntax used:

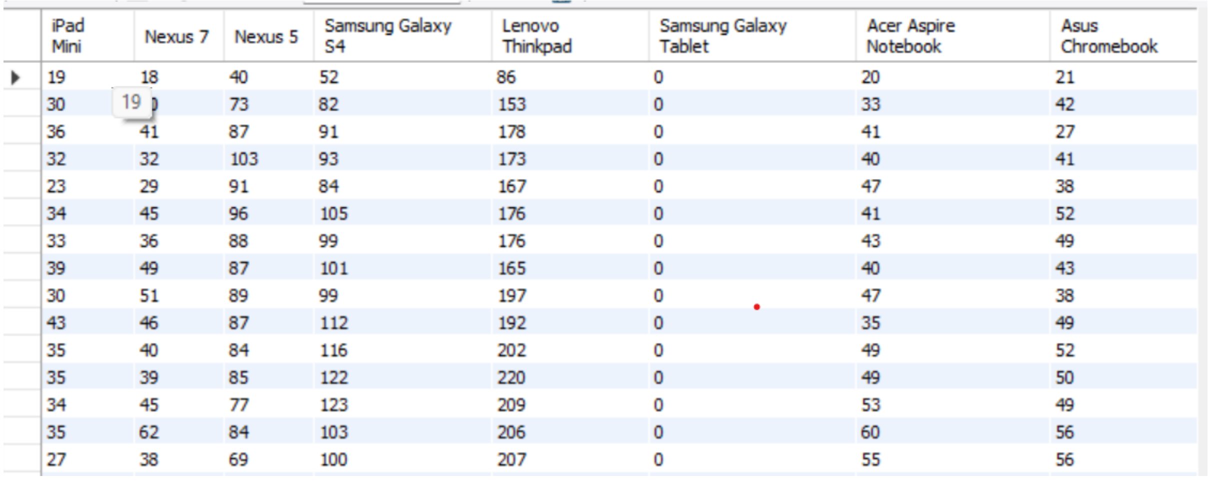
```
Select EXTRACT(WEEK FROM occured_at) AS "No. of weeks",
  COUNT(DISTINCT CASE WHEN device IN ('dell inspiron notebook') THEN user_id ELSE NULL
END) AS "Dell Inspiron Notebook",
  COUNT(DISTINCT CASE WHEN device IN ('iphone 5') THEN user_id ELSE NULL END) AS
"iPhone 5",
  COUNT(DISTINCT CASE WHEN 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",
```

```sql
   COUNT(DISTINCT CASE WHEN device IN ('nexus 7') THEN user_id ELSE NULL END) AS "Nexus 7",
   COUNT(DISTINCT CASE WHEN device IN ('nexus 5') 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 ('samsung galaxy tablet') THEN user_id ELSE NULL END) AS "Samsung Galaxy Tablet",
   COUNT(DISTINCT CASE WHEN device IN ('acer aspire notebook') THEN user_id ELSE NULL END) AS "Acer Aspire Notebook",
   COUNT(DISTINCT CASE WHEN device IN ('asus chromebook') THEN user_id ELSE NULL END) AS "Asus Chromebook",
   COUNT(DISTINCT CASE WHEN device IN ('htc 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 ('hp 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",
   7
FROM events
WHERE event_type = 'engagement'
GROUP BY 1
ORDER BY 1;
```



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

No. of weeks	Dell Inspiron Notebook	iPhone 5	iPhone 4S	Windows Surface	Macbook Air	iPhone 5S	Macbook Pro	Kindle Fire
17	46	65	21	10	54	42	143	6
18	77	113	46	10	121	73	252	27
19	83	115	44	16	112	79	266	21
20	84	125	55	21	119	79	256	23
21	80	137	45	17	110	74	247	30
22	92	125	45	15	145	71	251	21
23	103	152	53	14	124	79	266	25
24	99	142	53	22	152	79	255	25
25	105	137	40	22	121	78	275	24
26	89	152	50	21	134	94	269	26
27	89	163	67	33	142	83	302	25
28	103	151	61	33	148	93	295	31
29	113	144	60	28	148	90	295	37
30	127	152	65	19	159	103	322	25
31	113	135	56	19	147	71	321	14
32	104	119	34	10	125	67	307	12
33	110	110	35	15	133	65	312	14
34	105	101	50	18	136	70	292	13
35	9	2	6	3	10	3	17	3



iPad Mini	Nexus 7	Nexus 5	Samsung Galaxy S4	Lenovo Thinkpad	Samsung Galaxy Tablet	Acer Aspire Notebook	Asus Chromebook
19	18	40	52	86	0	20	21
30	19	73	82	153	0	33	42
36	41	87	91	178	0	41	27
32	32	103	93	173	0	40	41
23	29	91	84	167	0	47	38
34	45	96	105	176	0	41	52
33	36	88	99	176	0	43	49
39	49	87	101	165	0	40	43
30	51	89	99	197	0	47	38
43	46	87	112	192	0	35	49
35	40	84	116	202	0	49	52
35	39	85	122	220	0	49	50
34	45	77	123	209	0	53	49
35	62	84	103	206	0	60	56
27	38	69	100	207	0	55	56

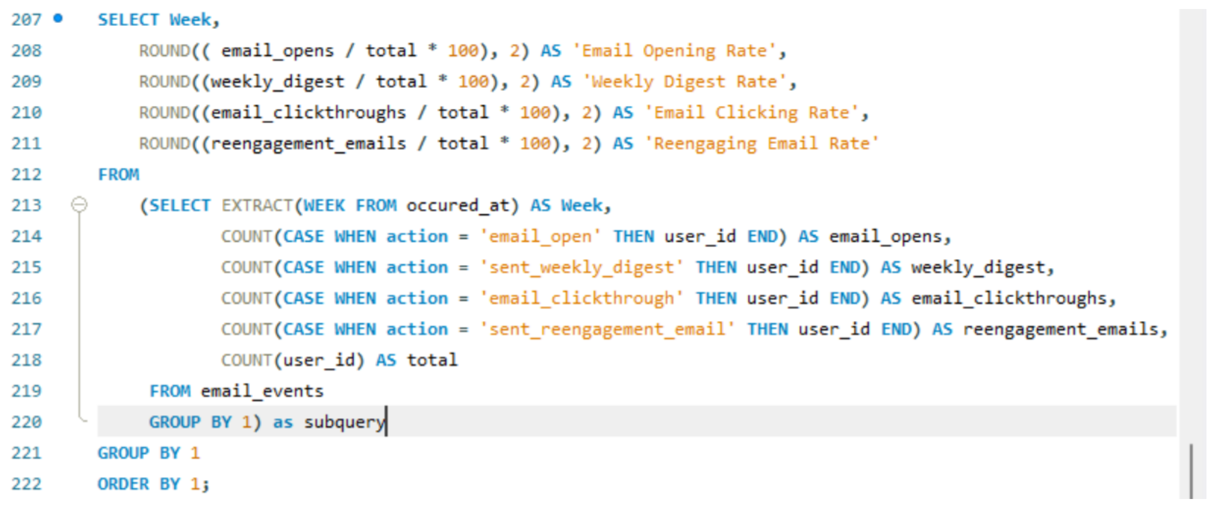
Insights:

We can observe that the most widely used device for engagement on weekly basis is Macbook Pro followed by Lenovo thinkpad and Macbook Air

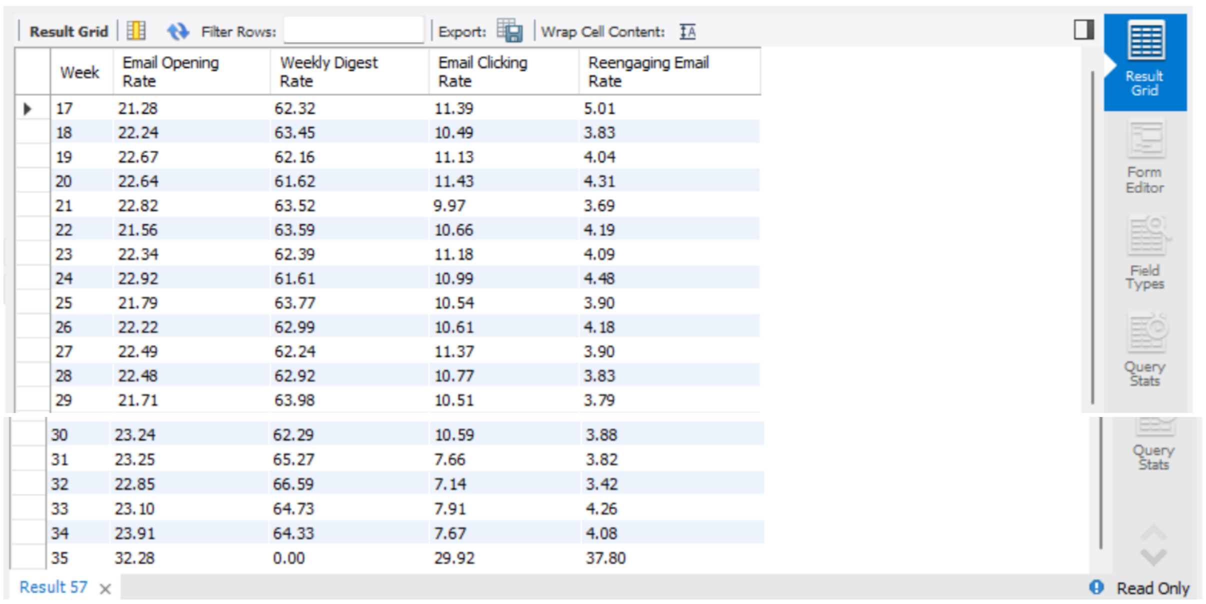
Q9 Analyze how users are engaging with the email service.
Task: Write an SQL query to calculate the email engagement metrics.

Syntax used:

```
SELECT Week,
   ROUND(( email_opens / total * 100), 2) AS 'Email Opening Rate',
   ROUND((weekly_digest / total * 100), 2) AS 'Weekly Digest Rate',
   ROUND((email_clickthroughs / total * 100), 2) AS 'Email Clicking Rate',
   ROUND((reengagement_emails / total * 100), 2) AS 'Reengaging Email Rate'
FROM
   (SELECT EXTRACT(WEEK FROM occured_at) AS Week,
         COUNT(CASE WHEN action = 'email_open' THEN user_id END) AS email_opens,
         COUNT(CASE WHEN action = 'sent_weekly_digest' THEN user_id END) AS weekly_digest,
         COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id END) AS email_clickthroughs,
         COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id END) AS reengagement_emails,
         COUNT(user_id) AS total
    FROM email_events
    GROUP BY 1) as subquery
GROUP BY 1
ORDER BY 1;
```



```
SELECT Week,
    ROUND(( email_opens / total * 100), 2) AS 'Email Opening Rate',
    ROUND((weekly_digest / total * 100), 2) AS 'Weekly Digest Rate',
    ROUND((email_clickthroughs / total * 100), 2) AS 'Email Clicking Rate',
    ROUND((reengagement_emails / total * 100), 2) AS 'Reengaging Email Rate'
FROM
    (SELECT EXTRACT(WEEK FROM occured_at) AS Week,
            COUNT(CASE WHEN action = 'email_open' THEN user_id END) AS email_opens,
            COUNT(CASE WHEN action = 'sent_weekly_digest' THEN user_id END) AS weekly_digest,
            COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id END) AS email_clickthroughs,
            COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id END) AS reengagement_emails,
            COUNT(user_id) AS total
     FROM email_events
     GROUP BY 1) as subquery
GROUP BY 1
ORDER BY 1;
```



Week	Email Opening Rate	Weekly Digest Rate	Email Clicking Rate	Reengaging Email Rate
17	21.28	62.32	11.39	5.01
18	22.24	63.45	10.49	3.83
19	22.67	62.16	11.13	4.04
20	22.64	61.62	11.43	4.31
21	22.82	63.52	9.97	3.69
22	21.56	63.59	10.66	4.19
23	22.34	62.39	11.18	4.09
24	22.92	61.61	10.99	4.48
25	21.79	63.77	10.54	3.90
26	22.22	62.99	10.61	4.18
27	22.49	62.24	11.37	3.90
28	22.48	62.92	10.77	3.83
29	21.71	63.98	10.51	3.79
30	23.24	62.29	10.59	3.88
31	23.25	65.27	7.66	3.82
32	22.85	66.59	7.14	3.42
33	23.10	64.73	7.91	4.26
34	23.91	64.33	7.67	4.08
35	32.28	0.00	29.92	37.80

Insights: The email opening rate is around 21.82%, email clicking rate is around 11.15%. The customers are continuously engaged with email services.

Insights:

- User Engagement: Analysed user engagement metrics such as login frequency, time spent on tasks, etc., to understand patterns and identify spikes in activity.
- Email Events: Investigated email-related metrics such as open rates, click-through rates, etc., to assess the effectiveness of email campaigns and detect any anomalies.
- Operational Efficiency: Identified areas for improvement in company operations based on the analysis of various metrics and trends.
- Data-Driven Decision Making: Empowered stakeholders with actionable insights derived from the analysis, facilitating better decision-making processes.

Result:

Through the project, I have gained a deeper understanding of operational analytics and the importance of investigating metric spikes in identifying areas for improvement within a company. The analysis has provided valuable insights that can help optimize operations, enhance user experiences, and drive business growth.