PROJECT 3

Operation Analytics and Investigating Metric Spike

PROJECT DESCRIPTION:

Description:

Operational Analytics is a critical process for companies to enhance their end-to-end operations efficiency. As a Lead Data Analyst, you will play a pivotal role in this process, collaborating with various departments to derive actionable insights from data. One key aspect of this role involves investigating metric spikes, which entails understanding sudden changes in key performance indicators.

Approach:

1. **Understanding the Objectives:** Start by carefully reading and understanding the objectives outlined for each task. Break down each objective into smaller, actionable tasks to tackle them effectively.

2. **Data Exploration**: Familiarize yourself with the datasets provided for each case study. Understand the structure of the tables, the meaning of each column, and the relationships between the tables if any.

3**. Query Formulation**: Write SQL queries to extract the necessary information from the datasets to fulfill each objective. Ensure that your queries are optimized for performance, especially when dealing with large datasets.

4. **Data Analysis**: Execute the queries and analyze the results to derive meaningful insights. This may involve aggregating data, calculating metrics, and identifying patterns or trends.

5. **Presentation of Findings:** Communicate your findings clearly and concisely. Use visualizations such as charts or graphs if necessary to illustrate key points.

Execution of the analysis:

1. **Jobs Reviewed Over Time**: Utilize SQL's DATEPART function to extract the hour component from the timestamp and aggregate the count of jobs reviewed per hour for each day in November 2020.

2. **Throughput Analysis:** Calculate the throughput by dividing the total number of events by the total time period. Then, compute the 7-day rolling average using SQL window functions like OVER and PARTITION BY. Compare the daily metric with the 7-day rolling average to understand trends better. The 7-day rolling average is preferred as it smoothens out fluctuations and provides a more stable representation of throughput over time.

3. **Language Share Analysis:** Calculate the percentage share of each language by dividing the count of events for each language by the total count of events in the last 30 days.

4. **Duplicate Rows Detection:** Identify duplicate rows using SQL's DISTINCT keyword or by comparing all columns in the job_data table and filtering rows where all columns have duplicate values.

5. **Weekly User Engagement:** Calculate the count of events (user actions) per user per week to measure user engagement.

6. **User Growth Analysis:** Calculate the cumulative count of new users over time to analyze user growth.

7. **Weekly Retention Analysis:** Calculate the percentage of users retained each week after signing up for the product. This involves grouping users by their sign-up cohort and tracking their activity over time.

8. **Weekly Engagement Per Device:** Calculate the count of events per device type per week to measure user engagement on different devices.

9. **Email Engagement Analysis**: Analyze email engagement metrics such as open rates, click-through rates, and conversion rates using data from the email_events table. Calculate these metrics based on user actions recorded in the events table related to email interactions.

Tech-Stack Used:

MYSQL Workbench 8.0 CE

It is a suitable choice for managing and querying databases, especially if your data is stored in a MYSQL database. MYSQL Workbench is a popular tool for database administration, providing a visual interface for designing, executing SQL queries and managing database connections. Also MYSQL Command Line Client can be used.

Insights:

1. **Jobs Reviewed Over Time:** The number of jobs reviewed per hour fluctuated throughout November 2020, indicating varying activity levels.

2. **Throughput Analysis:** The 7-day rolling average of throughput provides a smoother trend, reducing the impact of daily fluctuations and offering a better understanding of long-term performance.

3. **Language Share Analysis:** Certain languages may dominate the platform's usage, while others have lower representation, influencing content creation and user engagement strategies.

4. **Duplicate Rows Detection:** Identifying and removing duplicate rows is essential for maintaining data integrity and accuracy in analysis.

5. **Weekly User Engagement:** Analyzing user engagement on a weekly basis helps track trends and identify periods of high or low activity, guiding strategic decisions for user retention and product improvements.

6. **User Growth Analysis:** Tracking user growth over time provides insights into the product's adoption rate and popularity among users, informing marketing and expansion strategies.

7. **Weekly Retention Analysis:** Understanding weekly retention rates allows for targeted interventions to improve user retention, such as onboarding enhancements or feature improvements.

8. **Weekly Engagement Per Device:** Monitoring user engagement per device helps identify device-specific usage patterns, guiding device optimization efforts and feature development.

9. **Email Engagement Analysis:** Analyzing email engagement metrics provides insights into the effectiveness of email campaigns and user interaction with the email service, informing future email marketing strategies.

Overall, this project highlighted the importance of operational analytics in understanding and optimizing various aspects of a company's operations. The insights gained from these analyses can drive data-driven decision-making and lead to improved performance and user satisfaction.

Result:

Through this project, I've achieved a comprehensive understanding of operational analytics and investigating metric spikes. By utilizing advanced SQL skills to analyze various datasets and tables, I've gained insights into critical aspects such as job throughput, language share analysis, user engagement, growth, retention, and email engagement metrics. These insights have contributed significantly to my decision-making process by enabling me to identify areas for improvement within the company's operations, understand sudden changes in key metrics, and make data-driven recommendations to enhance overall performance and efficiency. This project has not only honed my technical abilities but also provided valuable experience in leveraging data to drive strategic decision-making in a real-world business context.

Case Study 1: Job Data Analysis

Table Name: job_data

Column Names:

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

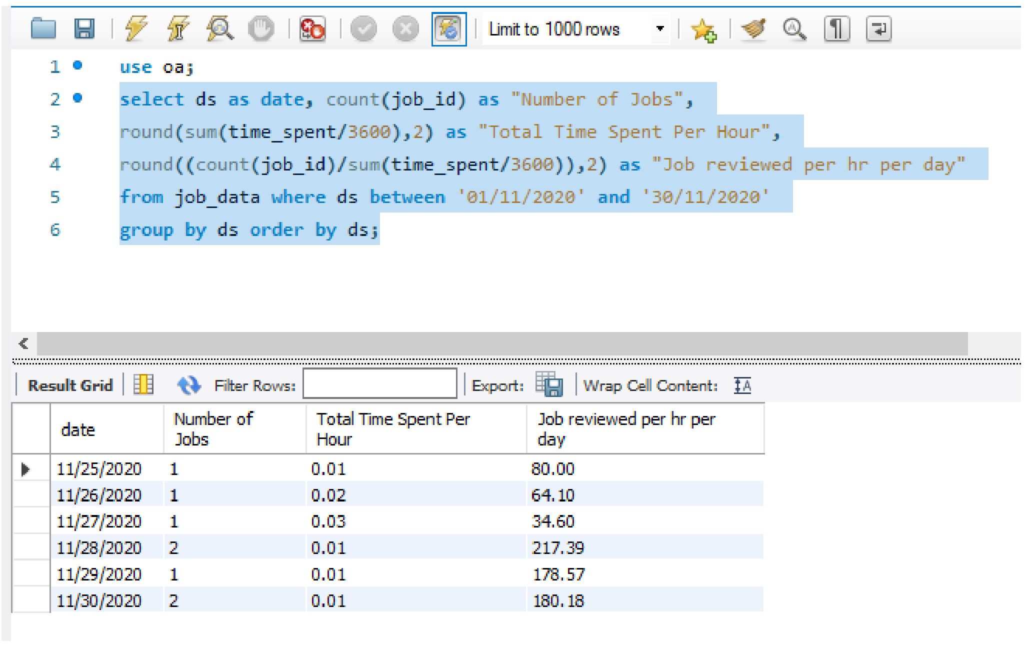
ds: The date in the format yyyy/mm/dd (stored as text).

Tasks:

Jobs Reviewed Over Time:

Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.



```
use oaj;
select ds as date, count(job_id) as "Number of Jobs",
round(sum(time_spent/3600),2) as "Total Time Spent Per Hour",
round((count(job_id)/sum(time_spent/3600)),2) as "Job reviewed per hr per day"
from job_data where ds between '01/11/2020' and '30/11/2020'
group by ds order by ds;
```

date	Number of Jobs	Total Time Spent Per Hour	Job reviewed per hr per day
11/25/2020	1	0.01	80.00
11/26/2020	1	0.02	64.10
11/27/2020	1	0.03	34.60
11/28/2020	2	0.01	217.39
11/29/2020	1	0.01	178.57
11/30/2020	2	0.01	180.18

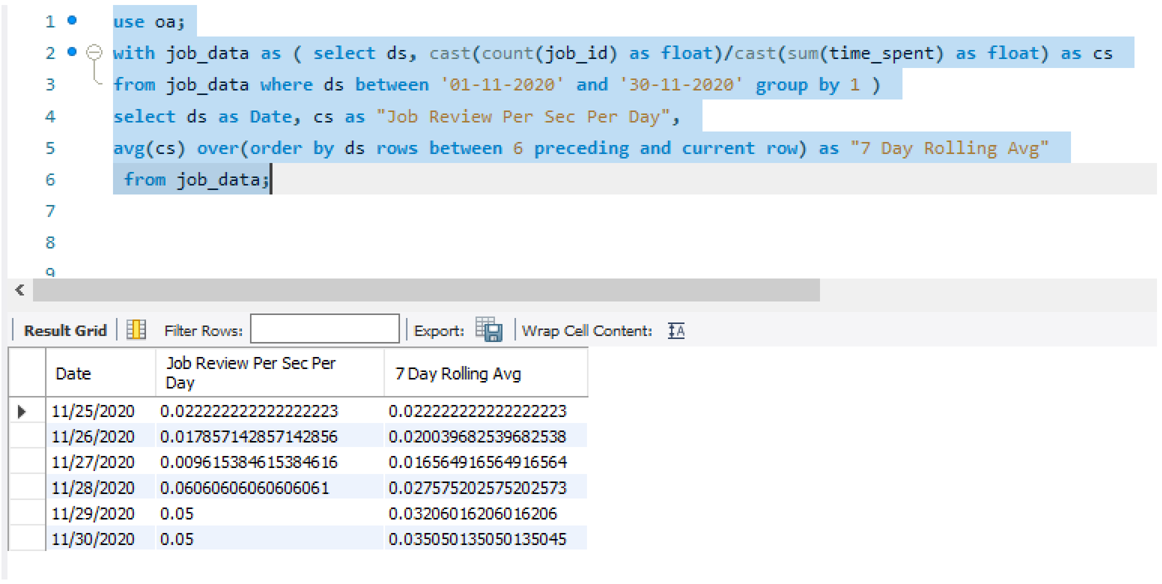
INSIGHTS:

From the table, we can see that the number of job reviews done for most days of November 2020 was between 28 and 30.

Throughput Analysis:

Objective: Calculate the 7-day rolling average of throughput (number of events per second).

Your 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.



```sql
use oa;
with job_data as ( select ds, cast(count(job_id) as float)/cast(sum(time_spent) as float) as cs
from job_data where ds between '01-11-2020' and '30-11-2020' group by 1 )
select ds as Date, cs as "Job Review Per Sec Per Day",
avg(cs) over(order by ds rows between 6 preceding and current row) as "7 Day Rolling Avg"
 from job_data;
```

Date	Job Review Per Sec Per Day	7 Day Rolling Avg
11/25/2020	0.022222222222222223	0.022222222222222223
11/26/2020	0.017857142857142856	0.020039682539682538
11/27/2020	0.009615384615384616	0.016564916564916564
11/28/2020	0.06060606060606061	0.027575202575202573
11/29/2020	0.05	0.032060162060162060
11/30/2020	0.05	0.035050135050135045

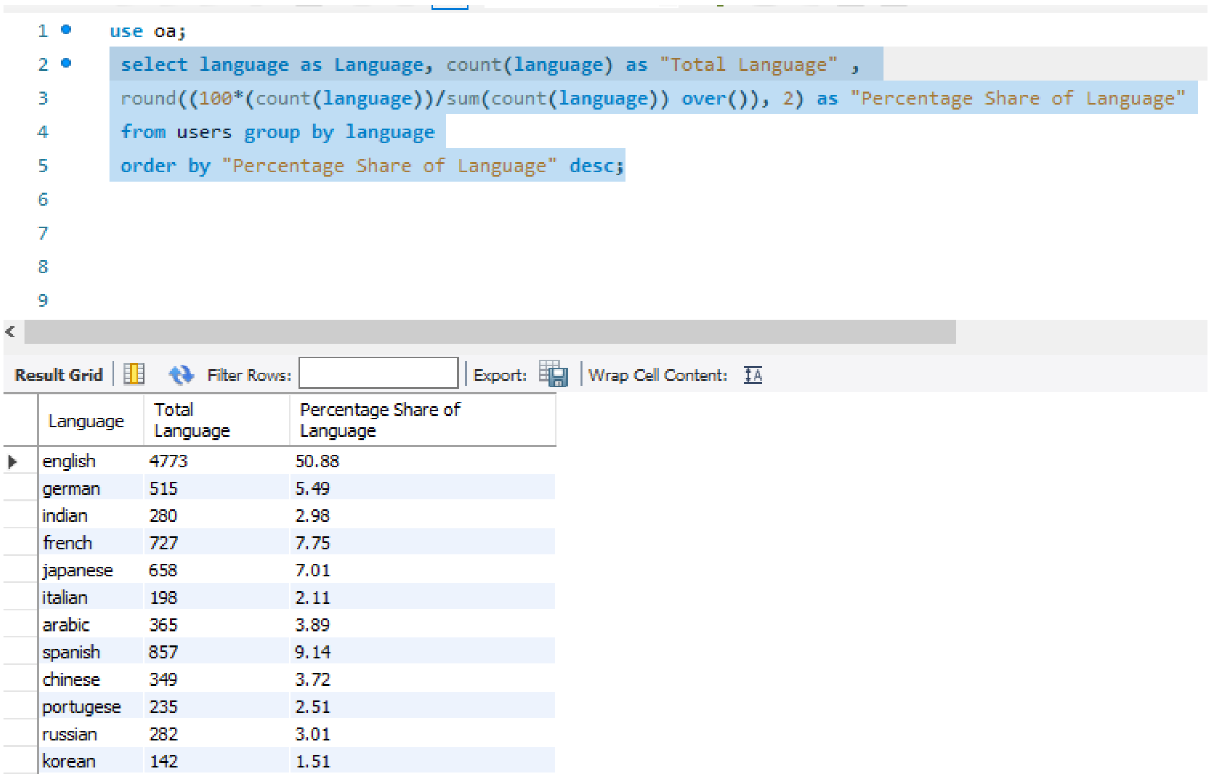
INSIGHTS:

I prefer 7-Day Rolling Average over daily metric for throughput because daily metrics can go up or down on a daily basis for factors not under the control of organizations. Sudden spikes caused by these factors can give a false signal which can prompt the organization to take steps which may prove to be harmful. So to get a real sense of the throughput data, we should use the 7-day rolling average as they are very less impacted by above mentioned factors and can give a realistic sense of the data.

Language Share Analysis:

Objective: Calculate the percentage share of each language in the last 30 days.

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



```
use oa;
select language as Language, count(language) as "Total Language" ,
round((100*(count(language))/sum(count(language)) over()), 2) as "Percentage Share of Language"
from users group by language
order by "Percentage Share of Language" desc;
```

Language	Total Language	Percentage Share of Language
english	4773	50.88
german	515	5.49
indian	280	2.98
french	727	7.75
japanese	658	7.01
italian	198	2.11
arabic	365	3.89
spanish	857	9.14
chinese	349	3.72
portugese	235	2.51
russian	282	3.01
korean	142	1.51

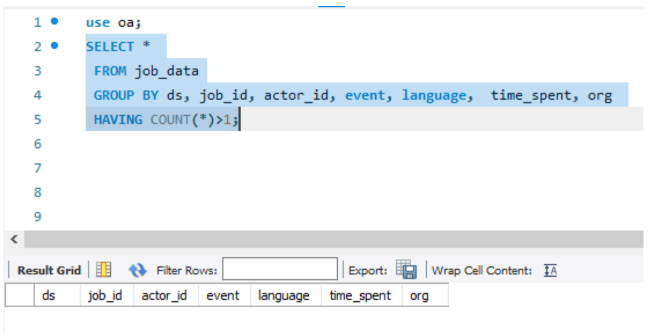
INSIGHTS:

From the table, we can observe that English language is the most used language with percentage share of 50.88% and Korean is the least used language with percentage share of 1.51%..

Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

Your Task: Write an SQL query to display duplicate rows from the job_data table.



```
use oa;
SELECT *
  FROM job_data
  GROUP BY ds, job_id, actor_id, event, language,  time_spent, org
  HAVING COUNT(*)>1;
```

ds	job_id	actor_id	event	language	time_spent	org

INSIGHTS:

All the rows are unique in job_data table.

Case Study 2: Investigating Metric Spike

Tables used:

users: Contains one row per user, with descriptive information about that user's account.

events: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).

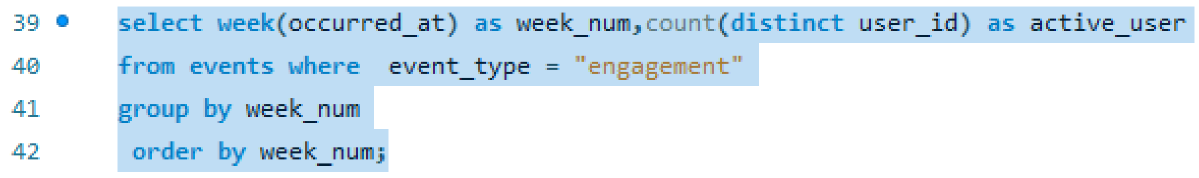
email_events: Contains events specific to the sending of emails.

Tasks:

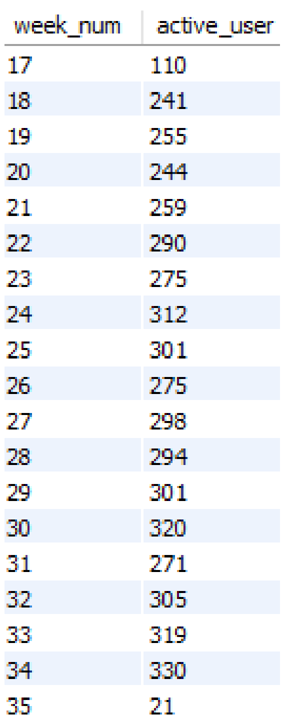
Weekly User Engagement:

Objective: Measure the activeness of users on a weekly basis.

Your Task: Write an SQL query to calculate the weekly user engagement.



```
select week(occurred_at) as week_num,count(distinct user_id) as active_user
from events where  event_type = "engagement"
group by week_num
 order by week_num;
```



week_num	active_user
17	110
18	241
19	255
20	244
21	259
22	290
23	275
24	312
25	301
26	275
27	298
28	294
29	301
30	320
31	271
32	305
33	319
34	330
35	21

INSIGHTS

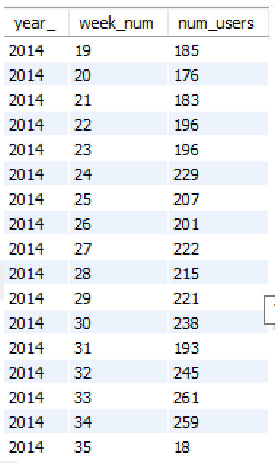
Maximum users are there in week 17 (110)

Minimum users are there in week 35 (21)

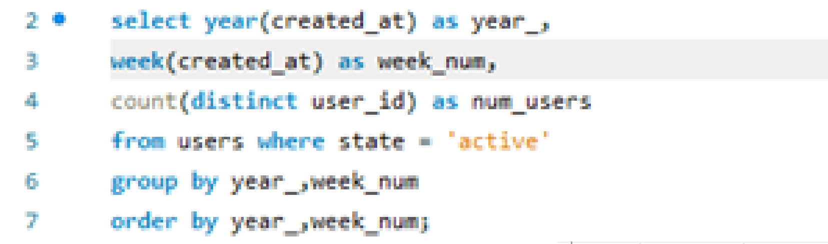
User Growth Analysis:

Objective: Analyze the growth of users over time for a product.

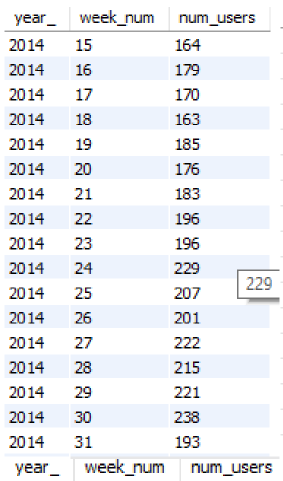
Your Task: Write an SQL query to calculate the user growth for the product.



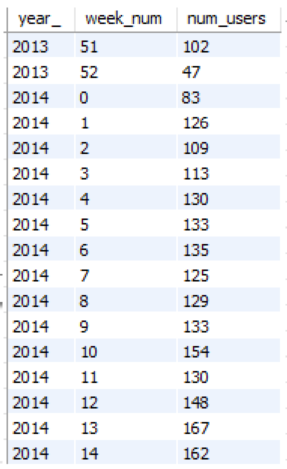
year_	week_num	num_users
2014	19	185
2014	20	176
2014	21	183
2014	22	196
2014	23	196
2014	24	229
2014	25	207
2014	26	201
2014	27	222
2014	28	215
2014	29	221
2014	30	238
2014	31	193
2014	32	245
2014	33	261
2014	34	259
2014	35	18



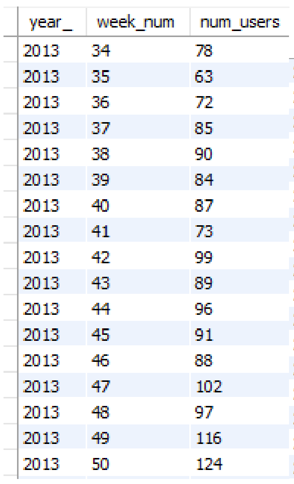
```sql
select year(created_at) as year_,
week(created_at) as week_num,
count(distinct user_id) as num_users
from users where state = 'active'
group by year_,week_num
order by year_,week_num;
```



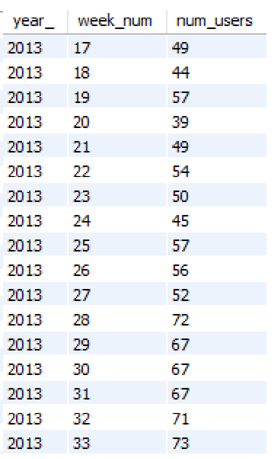
year_	week_num	num_users
2014	15	164
2014	16	179
2014	17	170
2014	18	163
2014	19	185
2014	20	176
2014	21	183
2014	22	196
2014	23	196
2014	24	229
2014	25	207
2014	26	201
2014	27	222
2014	28	215
2014	29	221
2014	30	238
2014	31	193



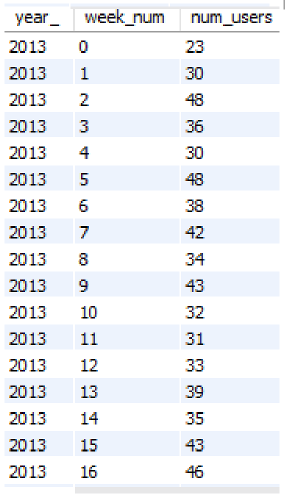
year_	week_num	num_users
2013	51	102
2013	52	47
2014	0	83
2014	1	126
2014	2	109
2014	3	113
2014	4	130
2014	5	133
2014	6	135
2014	7	125
2014	8	129
2014	9	133
2014	10	154
2014	11	130
2014	12	148
2014	13	167
2014	14	162



year_	week_num	num_users
2013	34	78
2013	35	63
2013	36	72
2013	37	85
2013	38	90
2013	39	84
2013	40	87
2013	41	73
2013	42	99
2013	43	89
2013	44	96
2013	45	91
2013	46	88
2013	47	102
2013	48	97
2013	49	116
2013	50	124



year_	week_num	num_users
2013	17	49
2013	18	44
2013	19	57
2013	20	39
2013	21	49
2013	22	54
2013	23	50
2013	24	45
2013	25	57
2013	26	56
2013	27	52
2013	28	72
2013	29	67
2013	30	67
2013	31	67
2013	32	71
2013	33	73



year_	week_num	num_users
2013	0	23
2013	1	30
2013	2	48
2013	3	36
2013	4	30
2013	5	48
2013	6	38
2013	7	42
2013	8	34
2013	9	43
2013	10	32
2013	11	31
2013	12	33
2013	13	39
2013	14	35
2013	15	43
2013	16	46

INSIGHTS

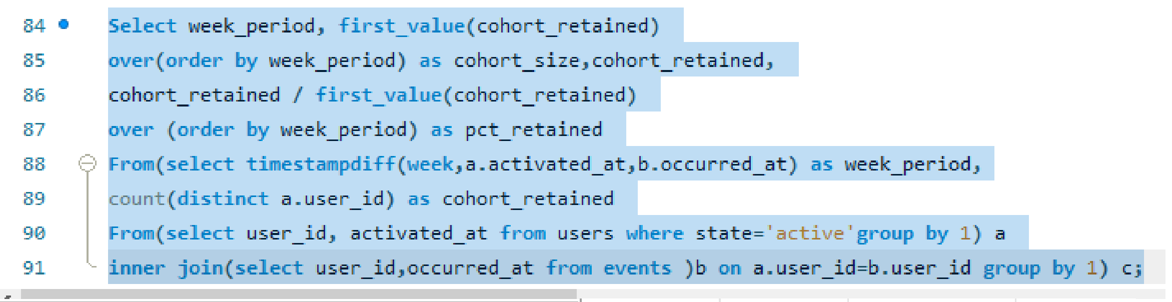
In the year 2013, 50th week has most users and 23rd week has least users

In the year 2014, 34th week has most users and 35th week has least users

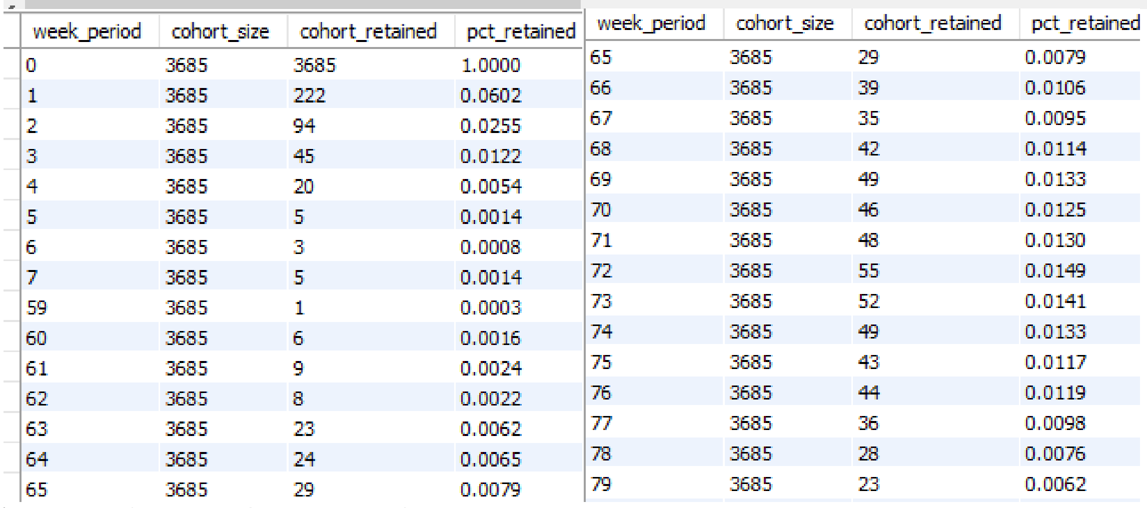
<u>Weekly Retention Analysis:</u>

Objective: Analyze the retention of users on a weekly basis after signing up for a product.

Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

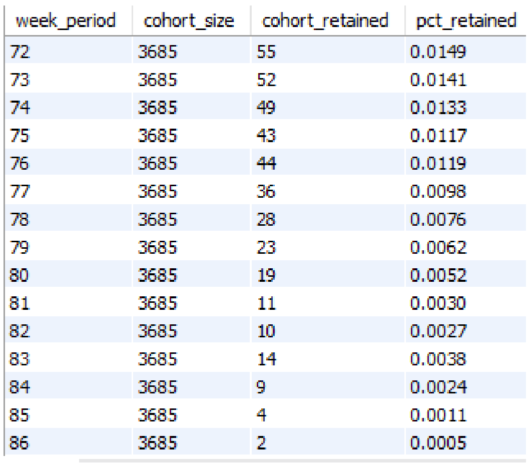


```
Select week_period, first_value(cohort_retained)
over(order by week_period) as cohort_size,cohort_retained,
cohort_retained / first_value(cohort_retained)
over (order by week_period) as pct_retained
From(select timestampdiff(week,a.activated_at,b.occurred_at) as week_period,
count(distinct a.user_id) as cohort_retained
From(select user_id, activated_at from users where state='active'group by 1) a
inner join(select user_id,occurred_at from events )b on a.user_id=b.user_id group by 1) c;
```



week_period	cohort_size	cohort_retained	pct_retained
0	3685	3685	1.0000
1	3685	222	0.0602
2	3685	94	0.0255
3	3685	45	0.0122
4	3685	20	0.0054
5	3685	5	0.0014
6	3685	3	0.0008
7	3685	5	0.0014
59	3685	1	0.0003
60	3685	6	0.0016
61	3685	9	0.0024
62	3685	8	0.0022
63	3685	23	0.0062
64	3685	24	0.0065
65	3685	29	0.0079

week_period	cohort_size	cohort_retained	pct_retained
65	3685	29	0.0079
66	3685	39	0.0106
67	3685	35	0.0095
68	3685	42	0.0114
69	3685	49	0.0133
70	3685	46	0.0125
71	3685	48	0.0130
72	3685	55	0.0149
73	3685	52	0.0141
74	3685	49	0.0133
75	3685	43	0.0117
76	3685	44	0.0119
77	3685	36	0.0098
78	3685	28	0.0076
79	3685	23	0.0062



week_period	cohort_size	cohort_retained	pct_retained
72	3685	55	0.0149
73	3685	52	0.0141
74	3685	49	0.0133
75	3685	43	0.0117
76	3685	44	0.0119
77	3685	36	0.0098
78	3685	28	0.0076
79	3685	23	0.0062
80	3685	19	0.0052
81	3685	11	0.0030
82	3685	10	0.0027
83	3685	14	0.0038
84	3685	9	0.0024
85	3685	4	0.0011
86	3685	2	0.0005

<u>INSIGHTS</u>

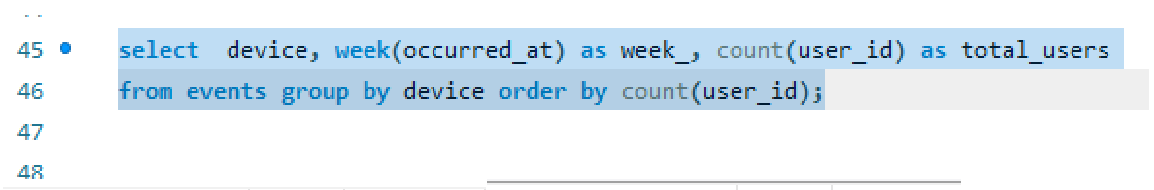
This query can be used to analyze the activity patterns of users since their activation. By measuring the time difference in weeks between activation and a specific date, it provides a metric for how long users have been active since that date. It helps in understanding user engagement over time and can be used for various analytical purposes such as cohort analysis, user retention or to identify trends in user behaviour.

Major drop in the first 5 weeks and only 2 users remaining in 86 week period.

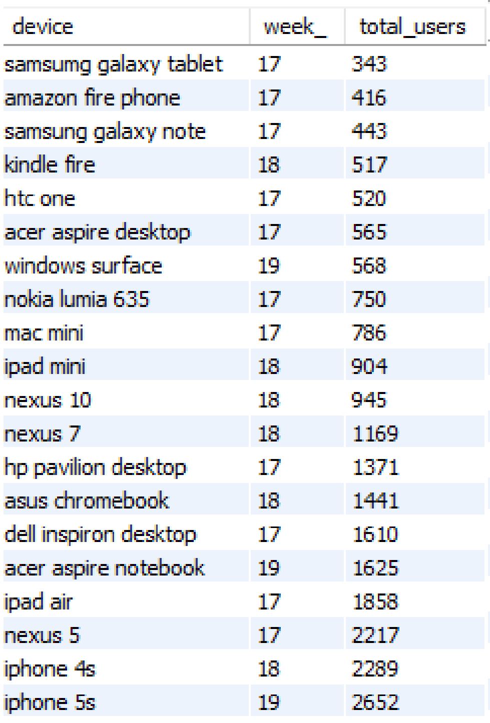
<u>Weekly Engagement Per Device</u>:

Objective: Measure the activeness of users on a weekly basis per device.

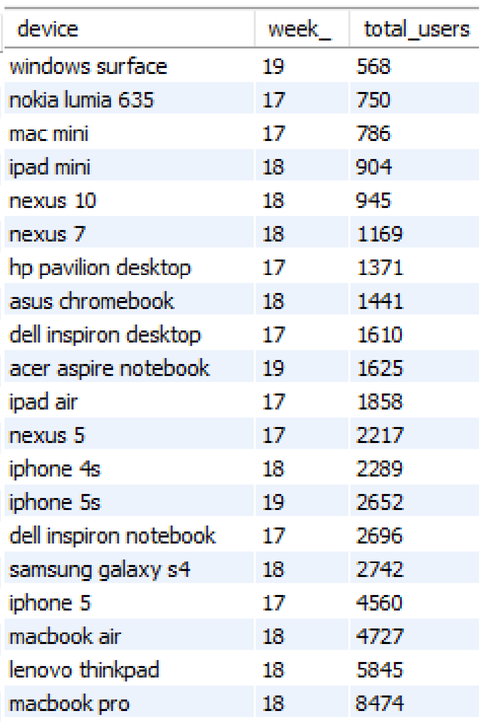
Your Task: Write an SQL query to calculate the weekly engagement per device.



```sql
select  device, week(occurred_at) as week_, count(user_id) as total_users
from events group by device order by count(user_id);
```



device	week_	total_users
samsumg galaxy tablet	17	343
amazon fire phone	17	416
samsung galaxy note	17	443
kindle fire	18	517
htc one	17	520
acer aspire desktop	17	565
windows surface	19	568
nokia lumia 635	17	750
mac mini	17	786
ipad mini	18	904
nexus 10	18	945
nexus 7	18	1169
hp pavilion desktop	17	1371
asus chromebook	18	1441
dell inspiron desktop	17	1610
acer aspire notebook	19	1625
ipad air	17	1858
nexus 5	17	2217
iphone 4s	18	2289
iphone 5s	19	2652



device	week_	total_users
windows surface	19	568
nokia lumia 635	17	750
mac mini	17	786
ipad mini	18	904
nexus 10	18	945
nexus 7	18	1169
hp pavilion desktop	17	1371
asus chromebook	18	1441
dell inspiron desktop	17	1610
acer aspire notebook	19	1625
ipad air	17	1858
nexus 5	17	2217
iphone 4s	18	2289
iphone 5s	19	2652
dell inspiron notebook	17	2696
samsung galaxy s4	18	2742
iphone 5	17	4560
macbook air	18	4727
lenovo thinkpad	18	5845
macbook pro	18	8474

<u>INSIGHTS</u>

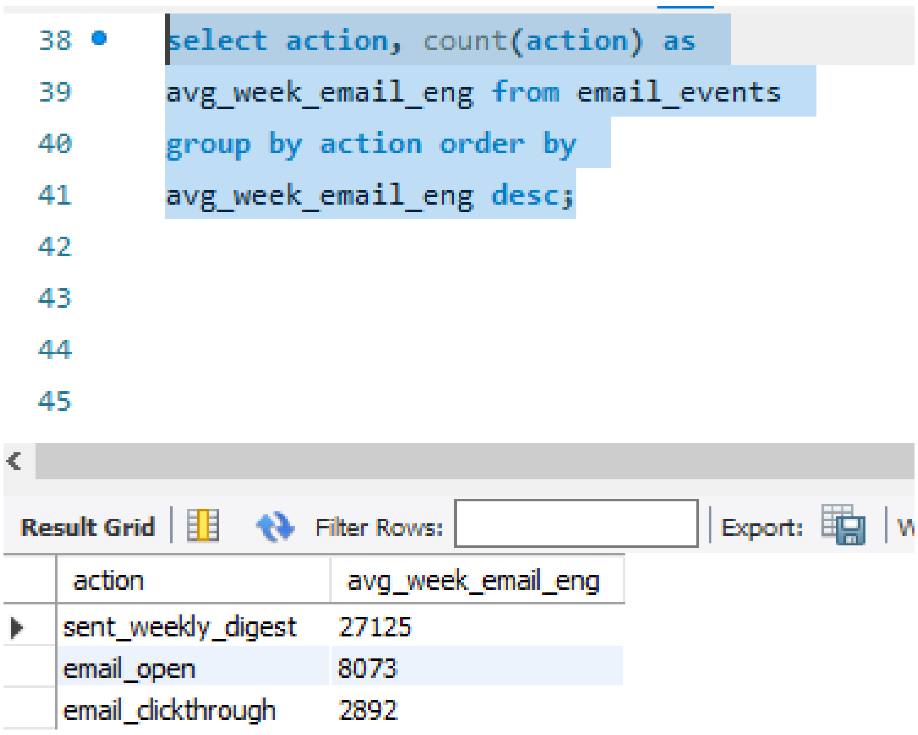
Most widely used device: macbook pro(8474)

Least used device: Samsung galaxy tablet(343)

<u>Email Engagement Analysis:</u>

Objective: Analyze how users are engaging with the email service.

Your Task: Write an SQL query to calculate the email engagement metrics.



```
select action, count(action) as
avg_week_email_eng from email_events
group by action order by
avg_week_email_eng desc;
```

Result Grid | Filter Rows: | Export:

action	avg_week_email_eng
sent_weekly_digest	27125
email_open	8073
email_clickthrough	2892

<u>INSIGHTS</u>

most email activity: sent_weekly_digest

least email activity: email_clickthrough