Operation Analytics and Investigating Metric Spike

Project Description

This project centers on Operational Analytics, employing advanced SQL techniques to analyze datasets and derive insights for improving company operations. As a Lead Data Analyst, the tasks involve investigating metric spikes and providing valuable information to different departments. Case studies include Job Data Analysis and Investigating Metric Spikes, with specific SQL queries addressing tasks such as calculating job reviews per hour, throughput analysis, language share analysis, and duplicate rows detection. The second case study delves into Weekly User Engagement, User Growth Analysis, Weekly Retention Analysis, Weekly Engagement Per Device, and Email Engagement Analysis. The project concludes with a comprehensive report summarizing key findings and insights for presentation to the leadership team.

Approach

1) **Database Setup -:** Commence the project by establishing a meticulously organized database. Import essential tables using CSV files through MySQL Workbench to lay a solid foundation.
2) **Data Exploration -:** Delve into a thorough understanding of table structures, deciphering column meanings, and grasping their significance in addressing the analytical tasks for each case study.
3) **Query Execution -:** Execute SQL queries with precision and efficiency to conduct in-depth analysis, addressing specific tasks outlined in the case studies.

Tech-Stack Used

The tech-stack used included MySQL workbench v8.8.0.34 which was user friendly interface, robust SQL querying capabilities, and compatibility with MySQL databases. It provides a seamless environment for database management analysis.

Insights

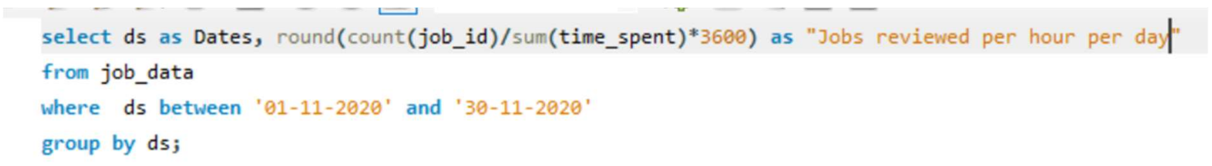
Case 1 Job Data Analysis

Worked with a table name “job_data”. Columns of table “job_data are –

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

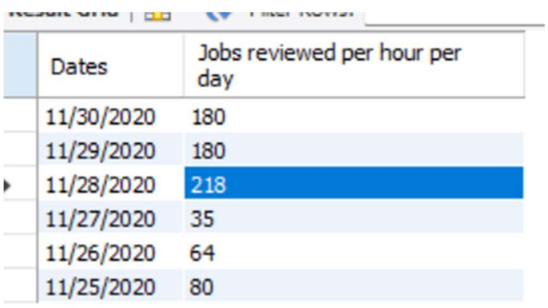
A) **Job Reviewed Over Time -:** Number of jobs reviewed per hour for each day in November 2020.
Task -: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

SQL Query -:



```
select ds as Dates, round(count(job_id)/sum(time_spent)*3600) as "Jobs reviewed per hour per day"
from job_data
where  ds between '01-11-2020' and '30-11-2020'
group by ds;
```

Output -:



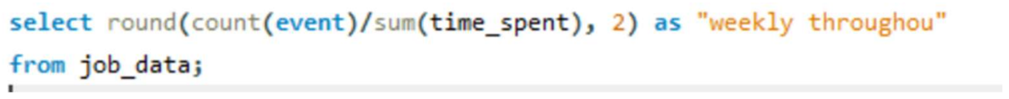
Dates	Jobs reviewed per hour per day
11/30/2020	180
11/29/2020	180
11/28/2020	218
11/27/2020	35
11/26/2020	64
11/25/2020	80

Conclusion -: On date 2020/11/28 there is a maximum number of jobs reviewed that is 218.

B) **Throughout Analysis -:** Calculate the 7-day rolling average of throughput (number of events per second).

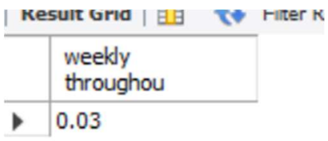
Task -: Write an SQL query to calculate the 7-day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

SQL Query -:



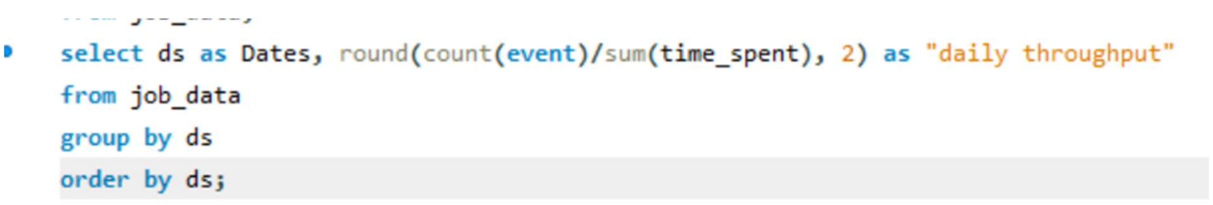
```
select round(count(event)/sum(time_spent), 2) as "weekly throughou"
from job_data;
```

Conclusion -:



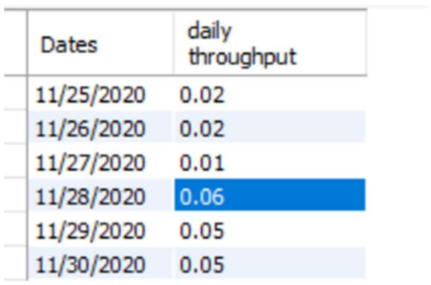
	weekly throughou
▶	0.03

SQL Query -:



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

Output -:



Dates	daily 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

Conclusion -:

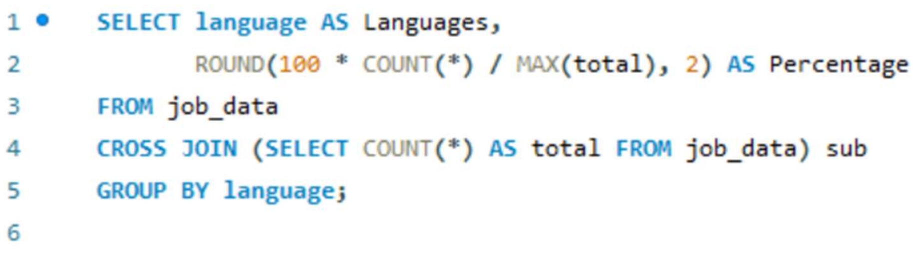
- The weekly throughput is 0.03.
- On date 28/11/2020 the throughput date is highest 0.06.

Performance metrics exhibit fluctuations on both a weekly and daily basis. Achieving quicker data updates is feasible on a daily or even minute-to-minute cadence. Consequently, rolling metrics excel in providing a nuanced representation of whether the metrics are on an upward or downward trend at a daily granularity.

C) **Language Share Analysis -:** Calculate the percentage share of each language in the last 30 days.

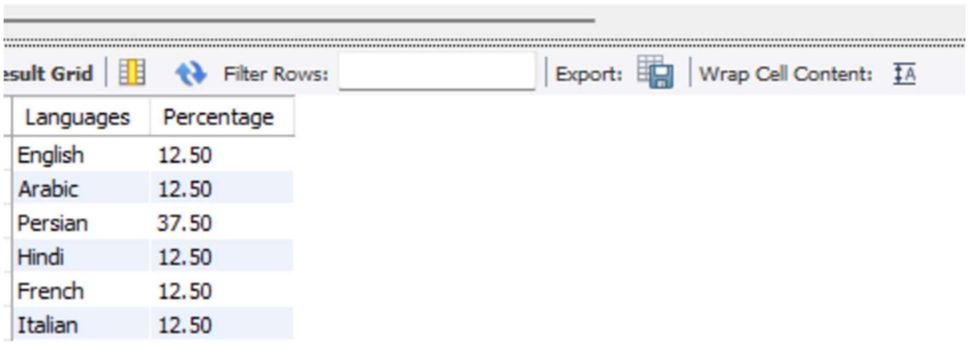
Task -: Write an SQL query to calculate the percentage share of each language over the last 30 days.

SQL Query And Output -:



```
SELECT language AS Languages,
        ROUND(100 * COUNT(*) / MAX(total), 2) AS Percentage
FROM job_data
CROSS JOIN (SELECT COUNT(*) AS total FROM job_data) sub
GROUP BY language;

```



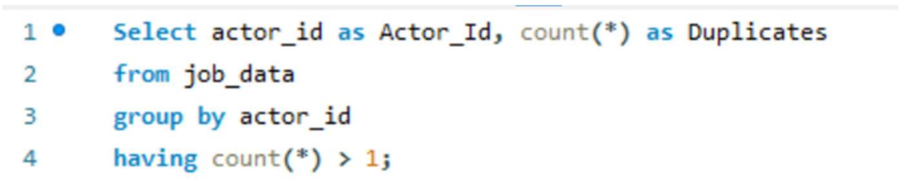
Languages	Percentage
English	12.50
Arabic	12.50
Persian	37.50
Hindi	12.50
French	12.50
Italian	12.50

Persian languages is highest with 37.5% total.

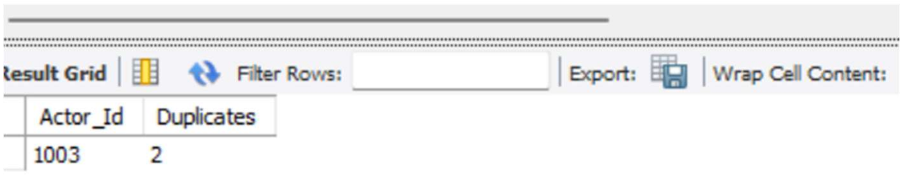
D) Duplicate Rows Detection -: Let's say you see some duplicate rows in data. How will you display duplicates rows form table?

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

SQL Query with Output -:



```
Select actor_id as Actor_Id, count(*) as Duplicates
from job_data
group by actor_id
having count(*) > 1;
```



Actor_Id	Duplicates
1003	2

Case 2 -: Investigating Metric Spike

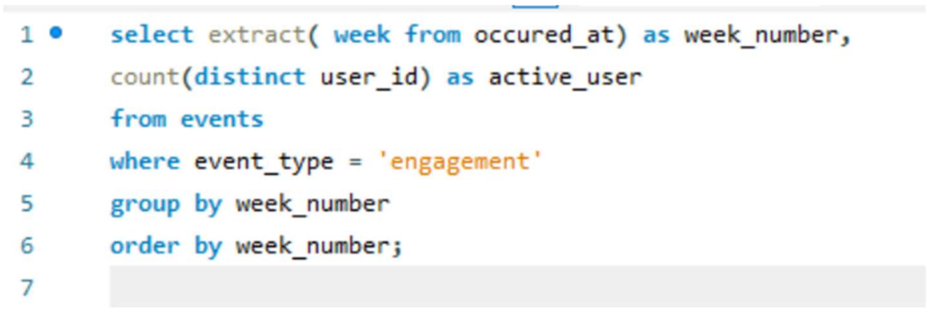
Worked with three tables

- 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).
- Emails_Events: Contains events specific to the sending of emails.

A) Weekly User Engagement -: To measure the activity of a user. Measuring if the user finds quality in a product/service.

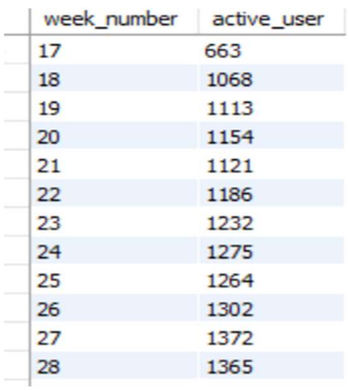
Task -: Write an SQL query to calculate the weekly user engagement.

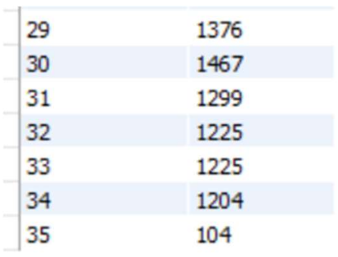
SQL Query



```
select extract( week from occured_at) as week_number,
count(distinct user_id) as active_user
from events
where event_type = 'engagement'
group by week_number
order by week_number;
```

Output



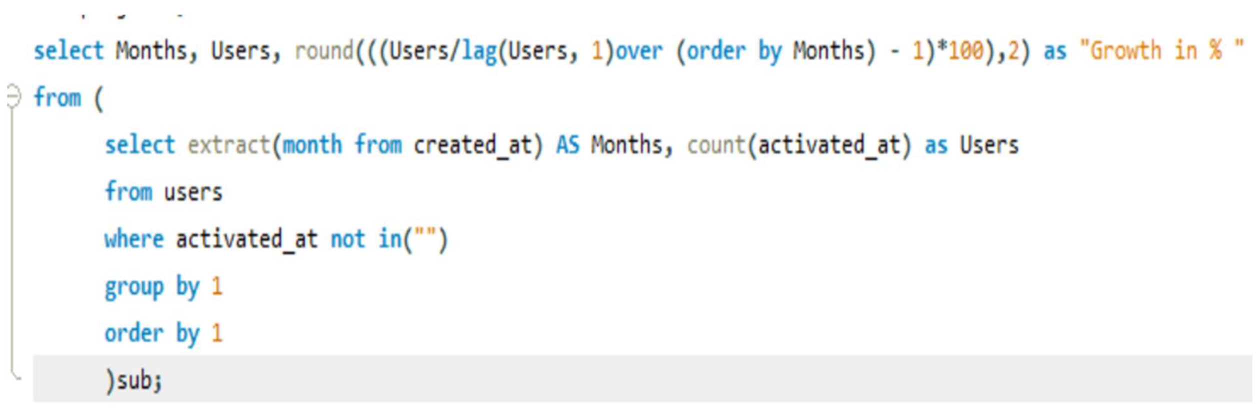


week_number	active_user
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

B)

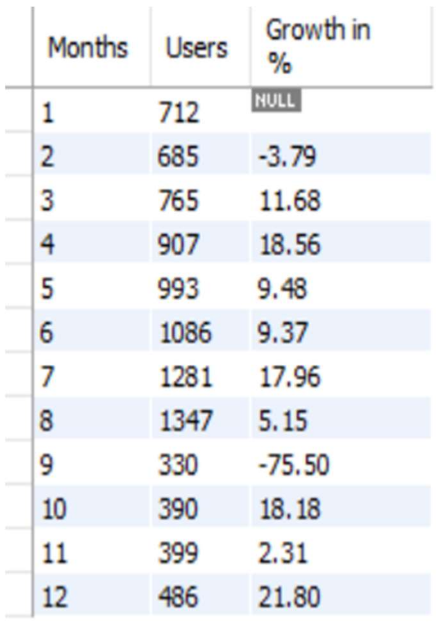
Task -: Write an SQL query to calculate the user growth for the product.

SQL Query -:



```
select Months, Users, round(((Users/lag(Users, 1)over (order by Months) - 1)*100),2) as "Growth in % "
from (
    select extract(month from created_at) AS Months, count(activated_at) as Users
    from users
    where activated_at not in("")
    group by 1
    order by 1
    )sub;
```

Output -:

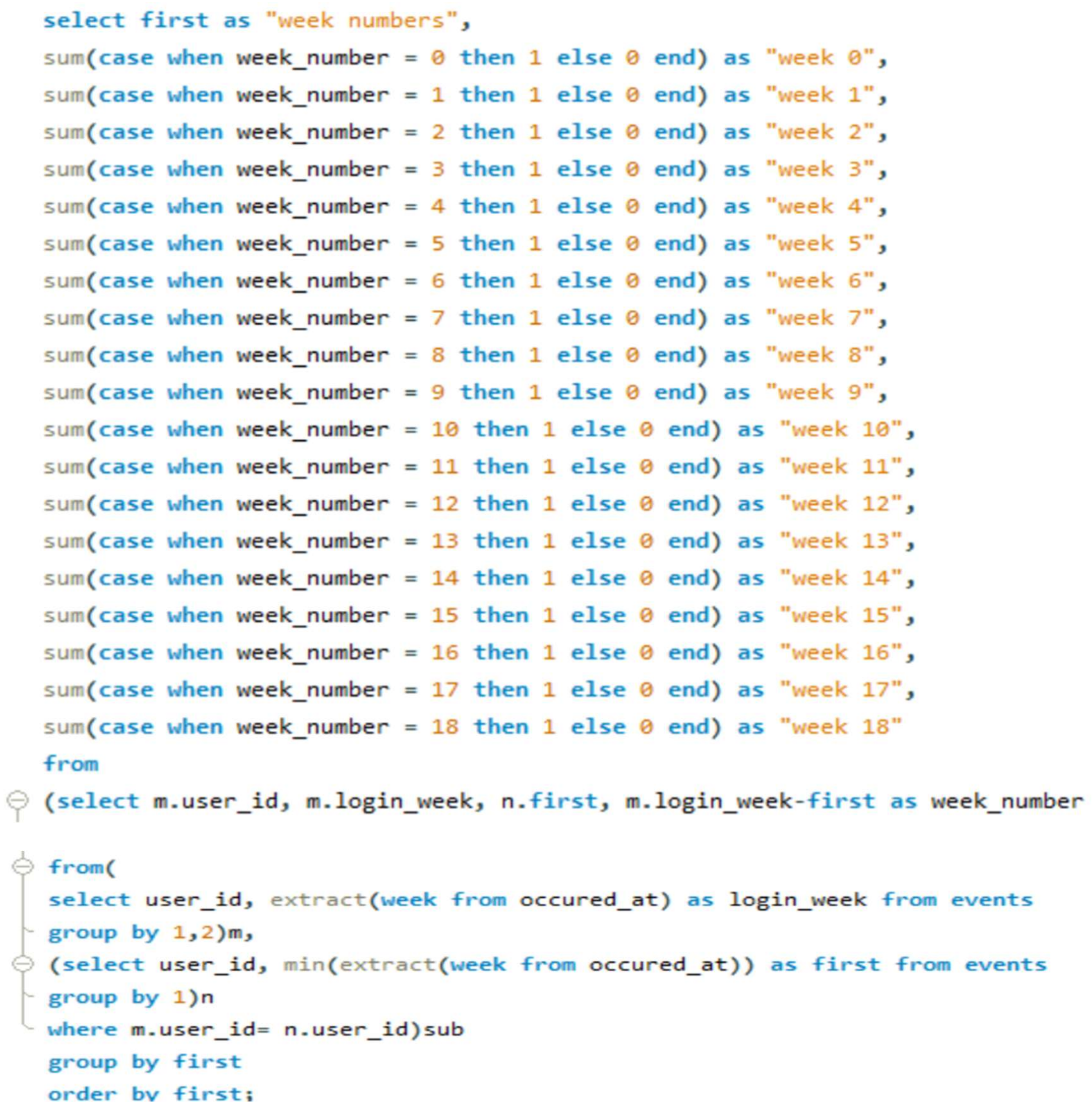


Months	Users	Growth in %
1	712	NULL
2	685	-3.79
3	765	11.68
4	907	18.56
5	993	9.48
6	1086	9.37
7	1281	17.96
8	1347	5.15
9	330	-75.50
10	390	18.18
11	399	2.31
12	486	21.80

C) Weekly Retention Analysis -:

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

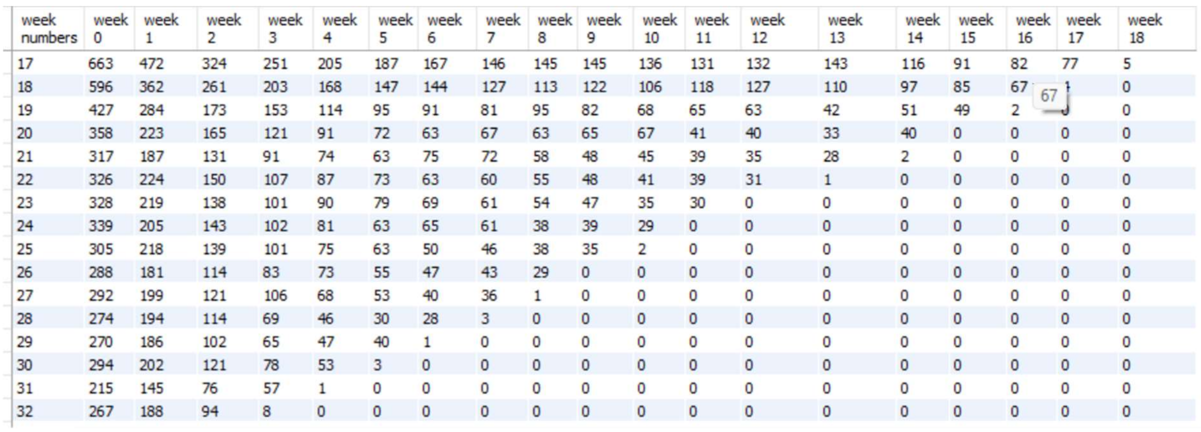
SQL Query-:



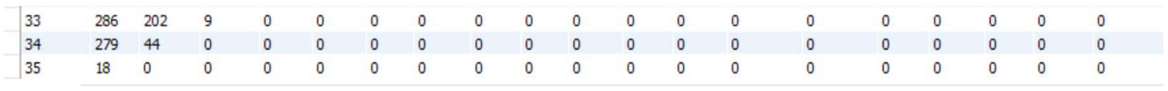
```
select first as "week numbers",
sum(case when week_number = 0 then 1 else 0 end) as "week 0",
sum(case when week_number = 1 then 1 else 0 end) as "week 1",
sum(case when week_number = 2 then 1 else 0 end) as "week 2",
sum(case when week_number = 3 then 1 else 0 end) as "week 3",
sum(case when week_number = 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 occured_at) as login_week from events
group by 1,2)m,
(select user_id, min(extract(week from occured_at)) as first from events
group by 1)n
where m.user_id= n.user_id)sub
group by first
order by first;
```

Output



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

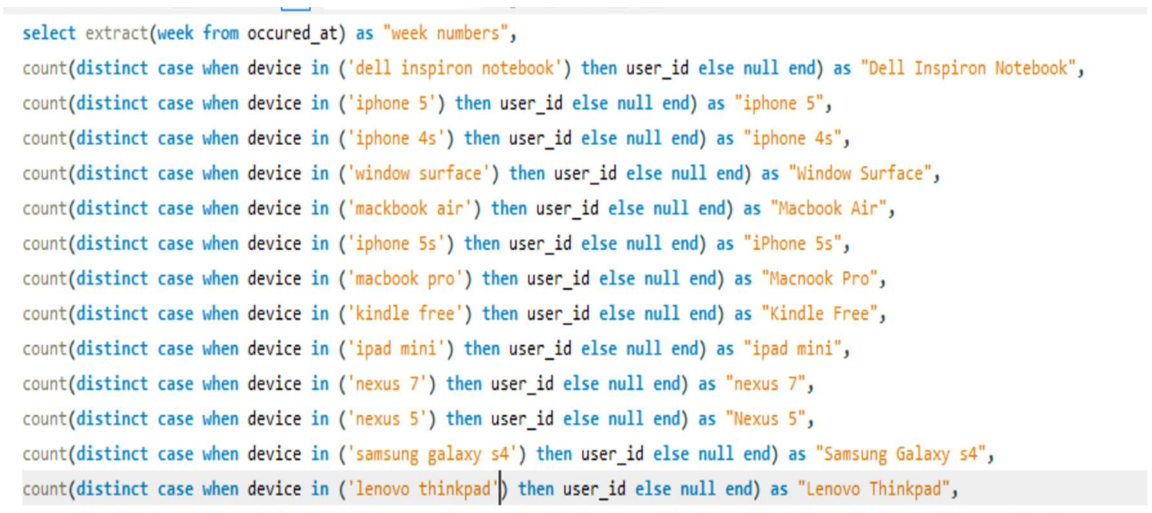


33	286	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	279	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

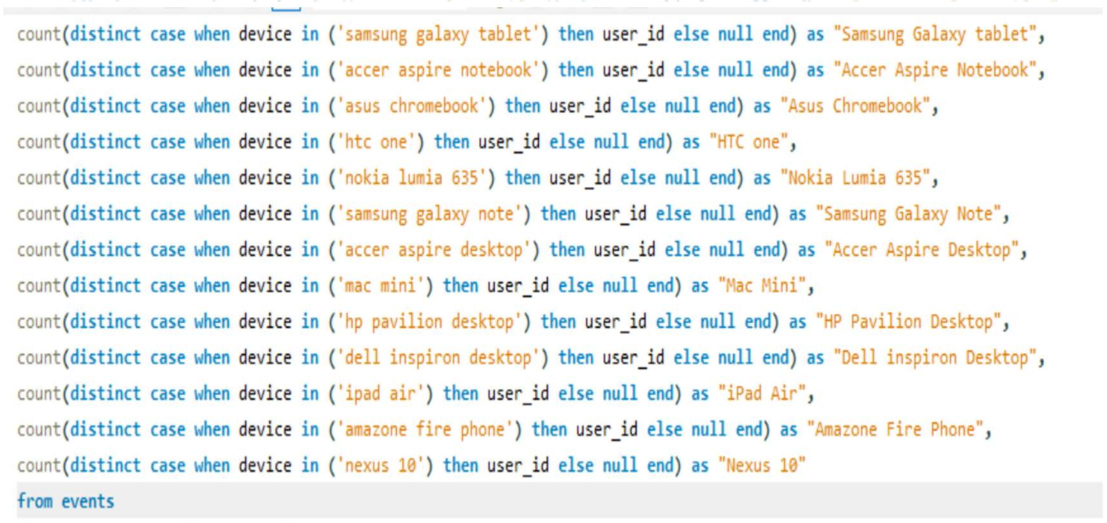
D) Weekly Engagement Per Device-:

Task-: Write an SQL Query to calculate weekly engagement per device.

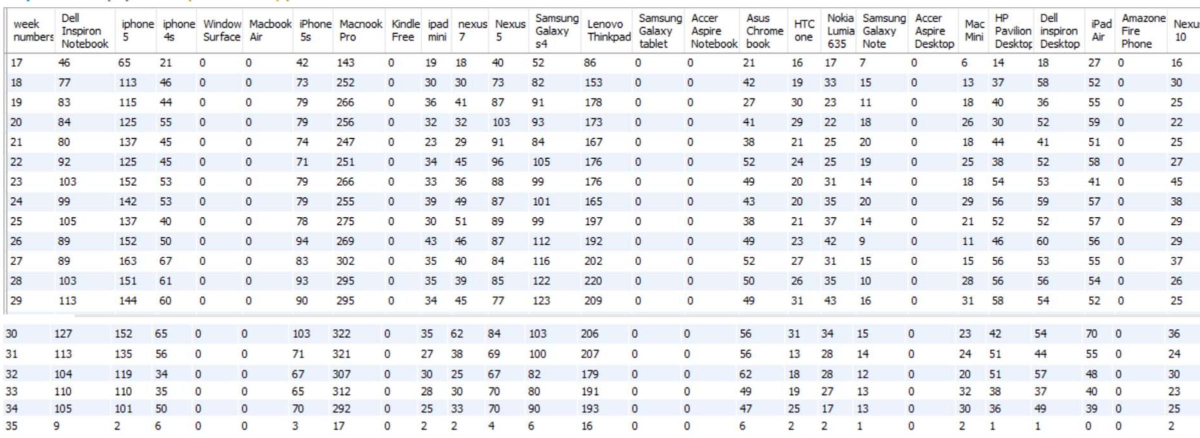
SQL Query-:



```
select extract(week from occured_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 ('window surface') then user_id else null end) as "Window Surface",
count(distinct case when device in ('mackbook 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 "Macnook Pro",
count(distinct case when device in ('kindle free') then user_id else null end) as "Kindle Free",
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 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 ('accer aspire notebook') then user_id else null end) as "Accer 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 ('accer aspire desktop') then user_id else null end) as "Accer 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 ('amazone fire phone') then user_id else null end) as "Amazone Fire Phone",
count(distinct case when device in ('nexus 10') then user_id else null end) as "Nexus 10"
from events
```

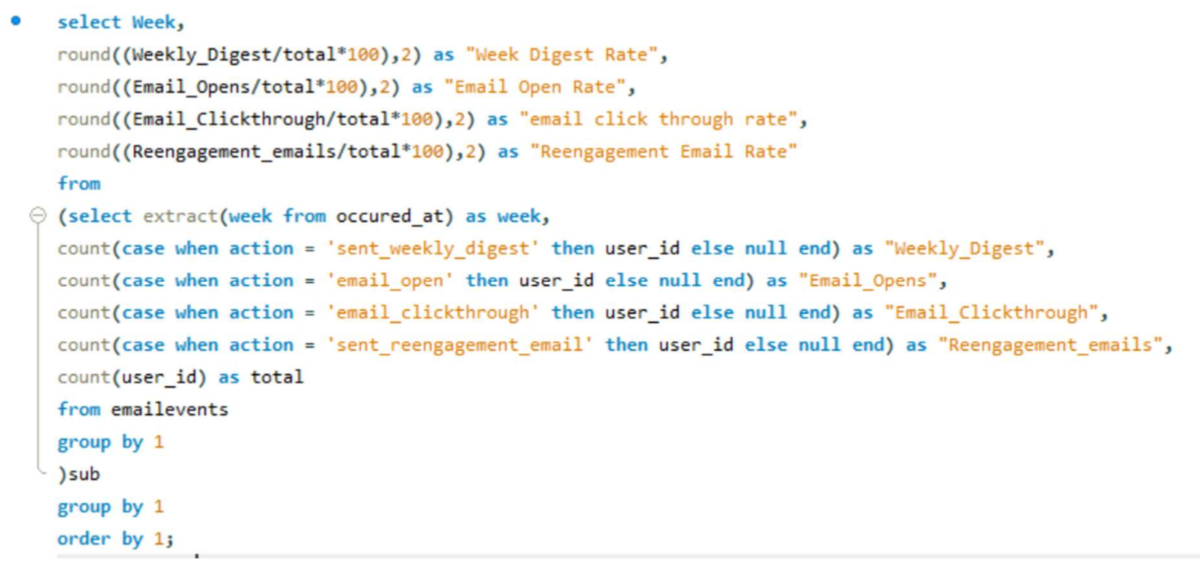


week numbers	Dell Inspiron Notebook	iphone 5	iphone 4s	Window Surface	Macbook Air	iPhone 5s	Macnook Pro	Kindle Free	ipad mini	nexus 7	Nexus 5	Samsung Galaxy s4	Lenovo Thinkpad	Samsung Galaxy tablet	Accer Aspire Notebook	Asus Chrome book	HTC one	Nokia Lumia 635	Samsung Galaxy Note	Accer Aspire Desktop	Mac Mini	HP Pavilion Desktop	Dell inspiron Desktop	iPad Air	Amazone Fire Phone	Nexus 10
17	46	65	21	0	0	42	143	0	19	18	40	52	86	0	0	21	16	17	7	0	6	14	18	27	0	16
18	77	113	46	0	0	73	252	0	30	30	73	82	153	0	0	42	19	33	15	0	13	37	58	52	0	30
19	83	115	44	0	0	79	266	0	36	41	87	91	178	0	0	27	30	23	11	0	18	40	36	55	0	25
20	84	125	55	0	0	79	256	0	32	32	103	93	173	0	0	41	29	22	18	0	26	30	52	59	0	22
21	80	137	45	0	0	74	247	0	23	29	91	84	167	0	0	38	21	25	20	0	18	44	41	51	0	25
22	92	125	45	0	0	71	251	0	34	45	96	105	176	0	0	52	24	25	19	0	25	38	52	58	0	27
23	103	152	53	0	0	79	266	0	33	36	88	99	176	0	0	49	20	31	14	0	18	54	53	41	0	45
24	99	142	53	0	0	79	255	0	39	49	87	101	165	0	0	43	20	35	20	0	29	56	59	57	0	38
25	105	137	40	0	0	78	275	0	30	51	89	99	197	0	0	38	21	37	14	0	21	52	52	57	0	29
26	89	152	50	0	0	94	269	0	43	46	87	112	192	0	0	49	23	42	9	0	11	46	60	56	0	29
27	89	163	67	0	0	83	302	0	35	40	84	116	202	0	0	52	27	31	15	0	15	56	53	55	0	37
28	103	151	61	0	0	93	295	0	35	39	85	122	220	0	0	50	26	35	10	0	28	56	56	54	0	26
29	113	144	60	0	0	90	295	0	34	45	77	123	209	0	0	49	31	43	16	0	31	58	54	52	0	25
30	127	152	65	0	0	103	322	0	35	62	84	103	206	0	0	56	31	34	15	0	23	42	54	70	0	36
31	113	135	56	0	0	71	321	0	27	38	69	100	207	0	0	56	13	28	14	0	24	51	44	55	0	24
32	104	119	34	0	0	67	307	0	30	25	67	82	179	0	0	62	18	28	12	0	20	51	57	48	0	30
33	110	110	35	0	0	65	312	0	28	30	70	80	191	0	0	49	19	27	13	0	32	38	37	40	0	23
34	105	101	50	0	0	70	292	0	25	33	70	90	193	0	0	47	25	17	13	0	30	36	49	39	0	25
35	9	2	6	0	0	3	17	0	2	2	4	6	16	0	0	6	2	2	1	0	2	1	1	0	0	2

E) Email Engagement Analysis -:

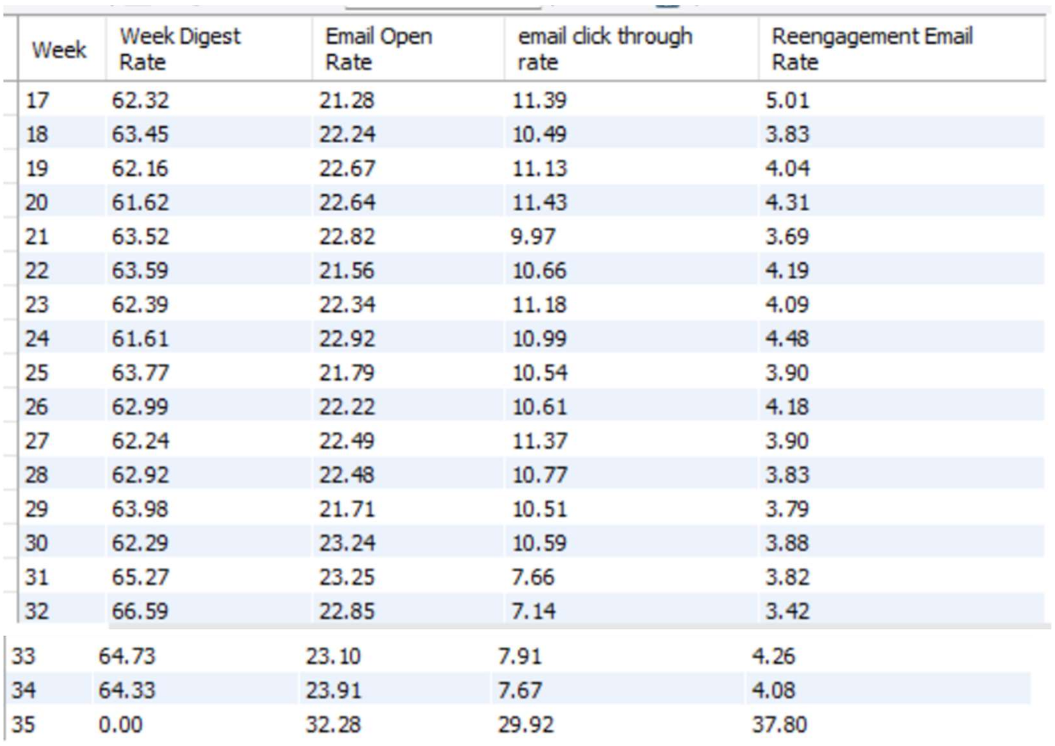
Task -: Write an SQL Query to calculate the email engagement metrics.

SQL Query -:



```
select Week,
round((Weekly_Digest/total*100),2) as "Week Digest Rate",
round((Email_Opens/total*100),2) as "Email Open Rate",
round((Email_Clickthrough/total*100),2) as "email click through rate",
round((Reengagement_emails/total*100),2) as "Reengagement Email Rate"
from
(select extract(week from occured_at) as week,
count(case when action = 'sent_weekly_digest' then user_id else null end) as "Weekly_Digest",
count(case when action = 'email_open' then user_id else null end) as "Email_Opens",
count(case when action = 'email_clickthrough' then user_id else null end) as "Email_Clickthrough",
count(case when action = 'sent_reengagement_email' then user_id else null end) as "Reengagement_emails",
count(user_id) as total
from emailevents
group by 1
)sub
group by 1
order by 1;
```

Output-:



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

Result

This project has helped me understand the importance of operational analytics. Through this project, I have come to understand how companies utilize metric spikes as a secret weapon. With an informed and proactive approach, they can leverage insights to make data-backed decisions that optimize their strategy and boost ROI.
The challenge I faced in this project was with the data in case study 2, which was very huge. Due to the large amount of data, importing it into SQL Workbooks was very slow. To tackle this situation, I had to use LOAD DATA statements.
Another problem arose in the "user type" column in the events table, which had a data type of int, causing issues with the import process. First, I needed to change the data type to text and then restart the process of loading the data into the events table.
In conclusion, operational analysis tackles problems by synchronizing real-time data. Operational Analytics can aggregate data from multiple sources into a cumulative, organized, actionable solution capable of delivering analytical models in real-time, creating individual customer profiles and a holistic view of operations for a company. This ensures that operational routines and systems are used efficiently.

THE END