OPERATION
&
METRIC ANALYTICS

By Thotamchetty Ranjith

Description:

- Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc. and help them derive insights out of the data they collect.

- Being one of the most important parts of a company, this kind of analysis is further used to predict the overall growth or decline of a company’s fortune. It means better automation, better understanding between cross-functional teams, and more effective workflows.

- Investigating metric spike is also an important part of operation analytics as being a Data Analyst you must be able to understand or make other teams understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that its very important to investigate metric spike.

Approach:

During these case studies, we approached the data carefully and examined the data thoroughly. We understood the questions and brainstormed how to get different answers for the questions provided. Some of the questions were complex and took time to understand but we worked hard to understand and deliver the required solutions for the both case studies.

Tech Stack Used:

We used MySQL Work Bench 8.0.32 for the querying process

Microsoft Excel was used to create the graphs and columnar data for the insights.

Case Study 1 (Job Data)

Below is the structure of the table with the definition of each column that you must work on:

Table-1: job_data

job_id: unique identifier of jobs

actor_id: unique identifier of actor

event: decision/skip/transfer

language: language of the content

time_spent: time spent to review the job in seconds

org: organization of the actor

ds: date in the yyyy/mm/dd format.

1. ***Number of jobs reviewed: Amount of jobs reviewed over time.***

Our task: Calculate the number of jobs reviewed per hour per day for November 2020?

Query:

SELECT COUNT(job_id)/(30*24) AS number_of_jobs_reviewed_per_day

FROM job_data;

Output:

number_of_jobs_reviewed_per_day
0.0111

Query:

SELECT count(distinct job_id)/(30*24) AS number_of_jobs_reviewed_per_day_distinct

FROM job_data;

Output:

number_of_jobs_reviewed_per_day_distinct
0.0083

Query for throughput 7 day rolling average

2. Throughput: It is the no. of events happening per second.

Our task: Let’s say the above metric is called throughput. Calculate 7 day rolling average of throughput.

Query:

```
SELECT ds,
 jobs_reviewed,
AVG(jobs_reviewed) OVER (ORDER BY ds rows between 6 preceding and current row) AS
throughput_7_day_rolling_average
FROM
(
SELECT ds,
COUNT(DISTINCT job_id) AS jobs_reviewed
FROM
job_data
WHERE ds BETWEEN "2020-11-01" AND "2020-11-30“
GROUP BY ds
ORDER BY ds
)b;
```

For throughput, do you prefer daily metric or 7-day rolling and why?

To calculate the 7-day rolling average of throughput, you would need to sum up the number of events happening in the past 7 days and divide by 7. Then, you would need to shift the window by 1 day and repeat the process until you have computed the rolling average for each day in your dataset.

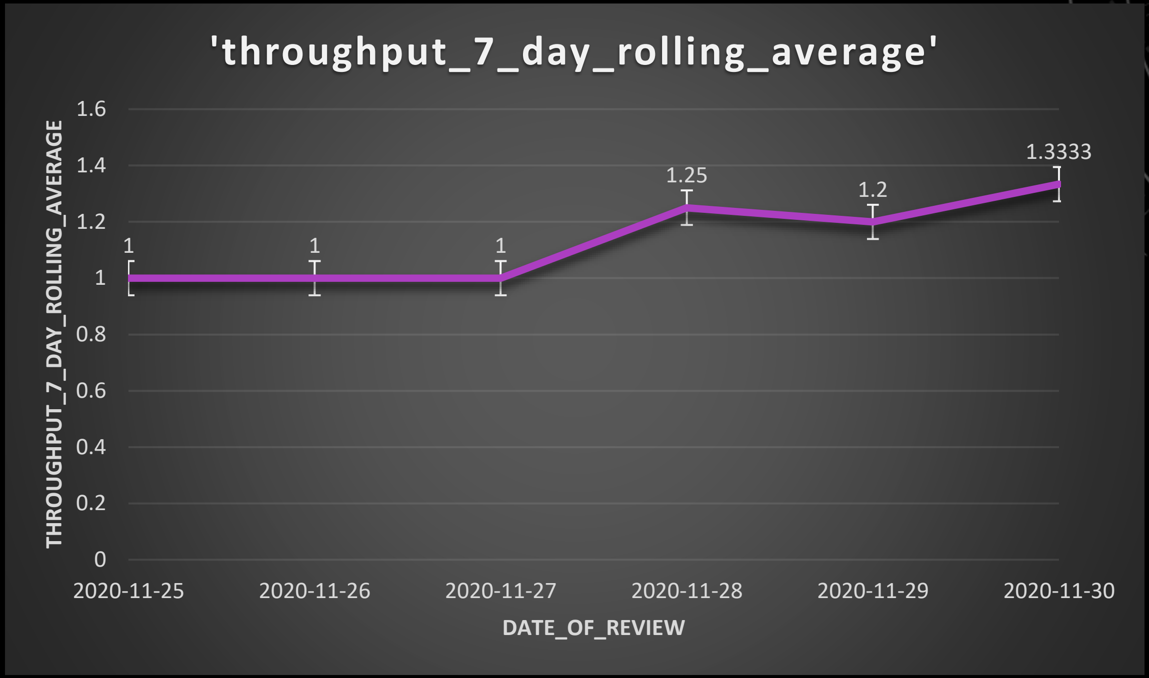
Regarding the preference for daily metric or 7-day rolling, it depends on the specific use case and the level of granularity needed.

If you need to monitor the performance of a system in real-time, a daily metric may provide a more accurate picture of the current state. However, if you are interested in identifying trends and patterns over a longer period of time, a 7-day rolling average may be more suitable as it can smooth out any fluctuations and provide a clearer view of the overall trend.

In general, both daily and 7-day rolling metrics have their own advantages and limitations, and the choice of which to use should be based on the specific requirements of the analysis.

Output and insights for the previous query:

date_of_review	jobs_reviewed	throughput_7_day_rolling_average
2020-11-25	1	1
2020-11-26	1	1
2020-11-27	1	1
2020-11-28	2	1.25
2020-11-29	1	1.2
2020-11-30	2	1.3333



3. Percentage share of each language: Share of each language for different contents.

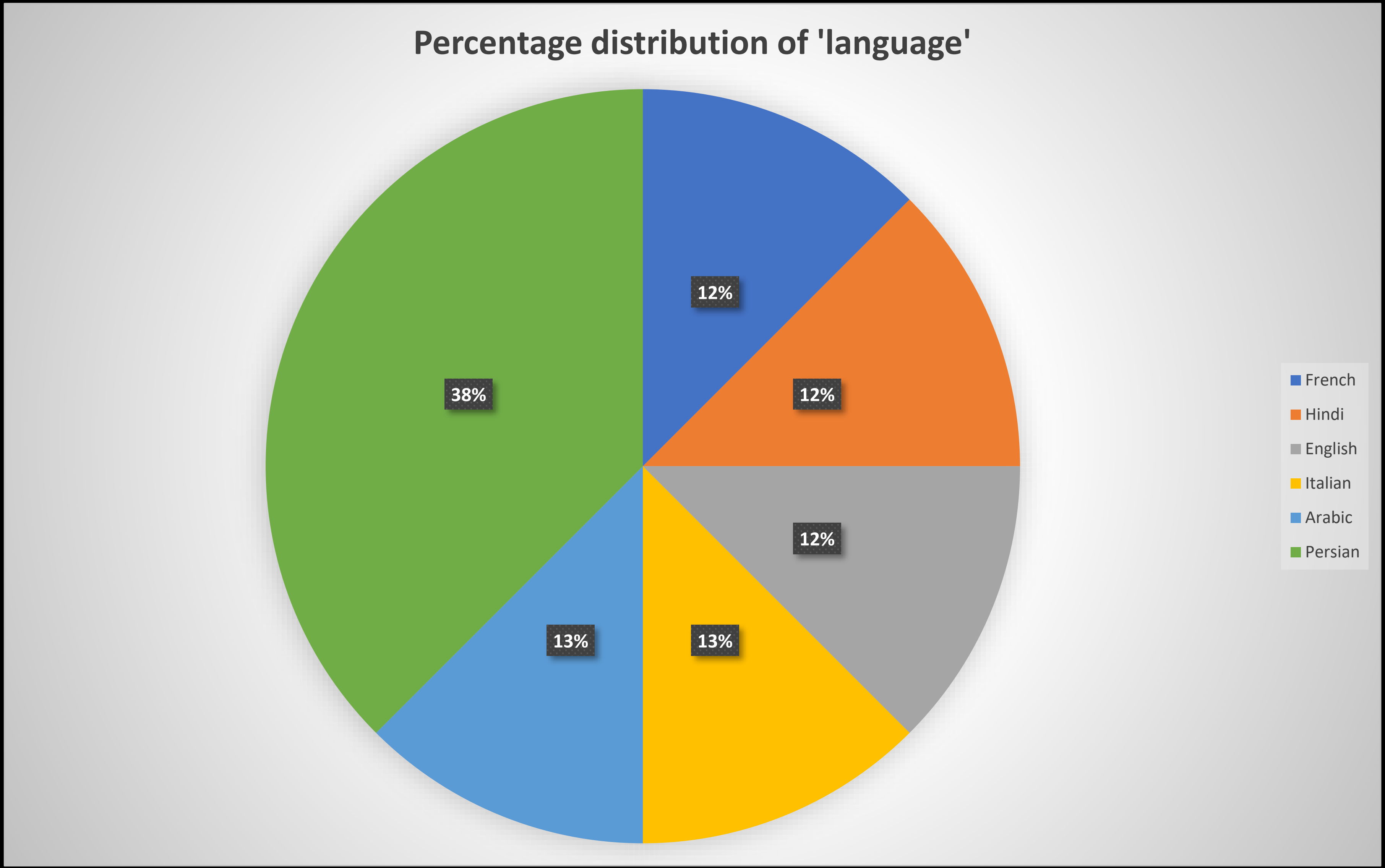
Your task: Calculate the percentage share of each language in the last 30 days?

Query:

```
SELECT language, count(language)*100/(SELECT count(*) FROM job_data) AS lang_percentage
FROM job_data
GROUP BY language;
```

Output:

language	lang_percentage
English	12.5000
Arabic	12.5000
Persian	37.5000
Hindi	12.5000
French	12.5000
Italian	12.5000


Percentage distribution of 'language'
12%
12%
12%
13%
13%
38%
French
Hindi
English
Italian
Arabic
Persian

4. Duplicate rows: Rows that have the same value present in them.

our task: Let's say you see some duplicate rows in the data.

Query:

```
SELECT *
FROM
(
SELECT *,
row_number( ) over(partition BY job_id) AS
rownum FROM job_data
)a
WHERE rownum>1;
```

Output:

ds	job_id	actor_id	event	language	time_spent	org	rownum
28-11-2020	23	1005	transfer	Persian	22	D	2
26-11-2020	23	1004	skip	Persian	56	A	3

INVESTIGATING METRIC SPIKE

Case Study 2 (Investigating metric spike)

The structure of the table with the definition of each column that you must work on is present in the project image

Table-1: users

This table includes one row per user, with descriptive information about that user’s account.

Table-2: events

This table includes one row per event, where an event is an action that a user has taken. These events include login events, messaging events, search events, events logged as users progress through a signup funnel, events around received emails.

Table-3: email_events

This table contains events specific to the sending of emails. It is similar in structure to the events table above.

In the upcoming slides we will be performing the tasks given to us

1. User Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service.

Our task: Calculate the weekly user engagement

Here we use login as user engagement as to engage in the our analysis.

Query:

SELECT WEEK(occurred_at) AS week,

COUNT(distinct user_id) AS login_count

FROM events

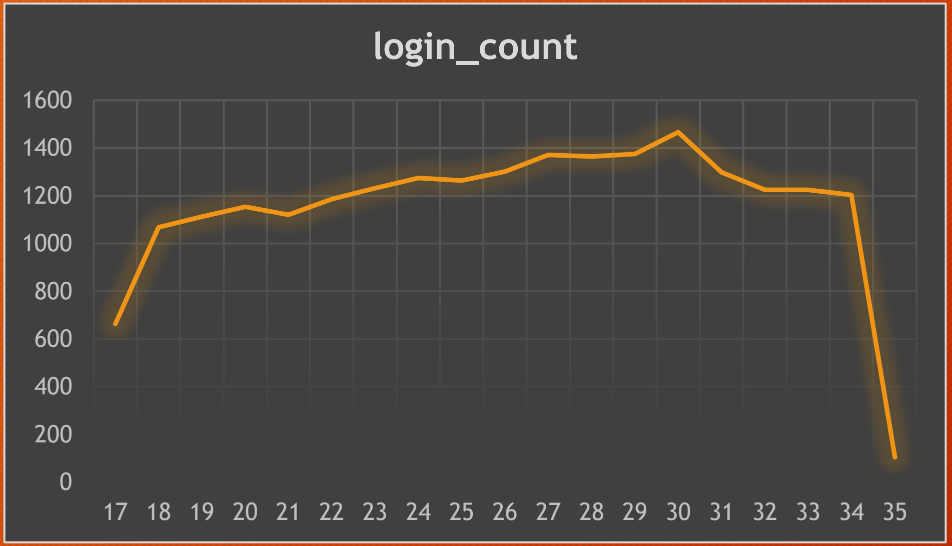
WHERE event_name = 'login’

GROUP BY week;

Output:

week	login_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

Insights:



2. User Growth: Amount of users growing over time for a product.

Our task: Calculate the user growth for product?

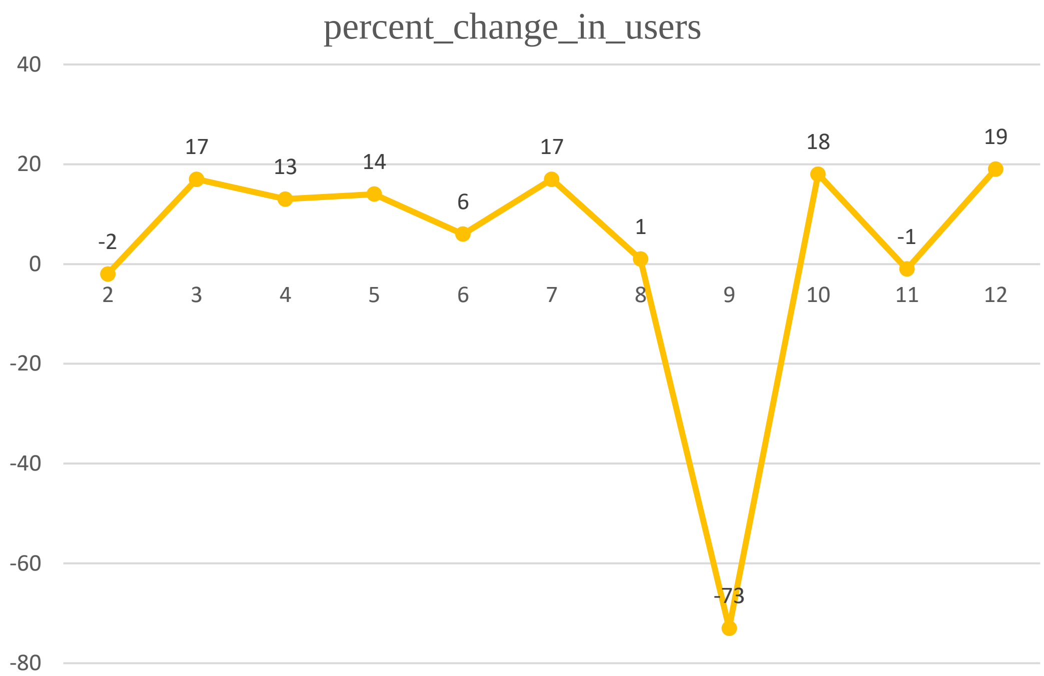
Query:

```
SELECT month,
no_of_created_users,
prev_month_users,
change_in_users_created,
round((change_in_users_created*100)/prev_month_users) AS percent_change
FROM (SELECT month,
no_of_created_users, lag(no_of_created_users,1) over(ORDER BY month) AS prev_month_users,
(no_of_created_users - LAG(no_of_created_users,1) OVER (ORDER BY month)) AS change_in_users_created
FROM (SELECT MONTH(created_at) AS month,
COUNT(user_id) AS no_of_created_users
FROM users
GROUP BY MONTH(created_at)
)AS subquery
) AS subquery2;
```

Output:

month	no_of_created_users	prev_month_users	change_in_users_created	percent_change
1	1415	NULL	NULL	NULL
2	1382	1415	-33	-2
3	1614	1382	232	17
4	1829	1614	215	13
5	2083	1829	254	14
6	2213	2083	130	6
7	2591	2213	378	17
8	2626	2591	35	1
9	699	2626	-1927	-73
10	826	699	127	18
11	816	826	-10	-1
12	972	816	156	19

Insights:



3. Weekly Retention: Users getting retained weekly after signing-up for a product.

Our task: Calculate the weekly retention of users-sign up cohort

Query:

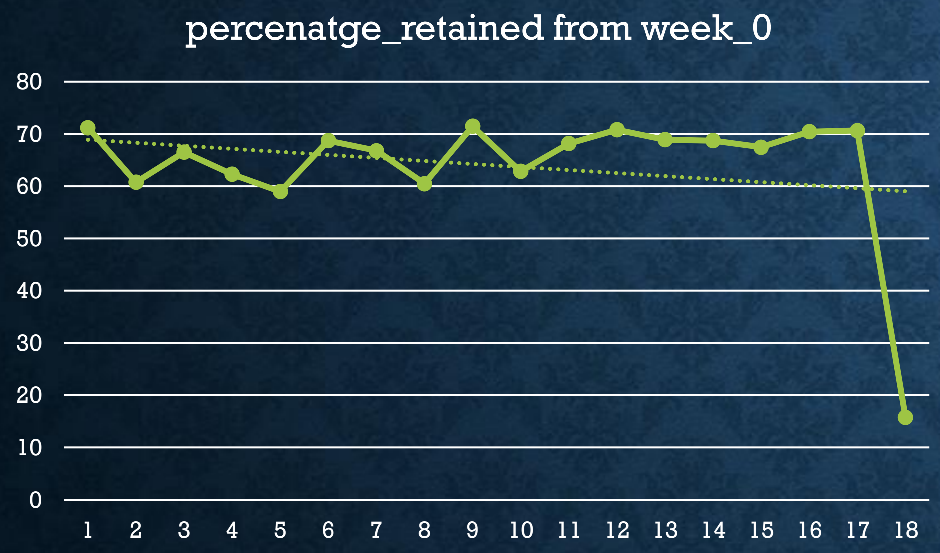
```
SELECT
first_login,
COUNT(CASE WHEN login_week = first_login THEN 1 END) AS week_0,
COUNT(CASE WHEN login_week = first_login + 1 THEN 1 END) AS week_1,
COUNT(CASE WHEN login_week = first_login + 2 THEN 1 END) AS week_2,
COUNT(CASE WHEN login_week = first_login + 3 THEN 1 END) AS week_3,
COUNT(CASE WHEN login_week = first_login + 4 THEN 1 END) AS week_4,
COUNT(CASE WHEN login_week = first_login + 5 THEN 1 END) AS week_5,
COUNT(CASE WHEN login_week = first_login + 6 THEN 1 END) AS week_6,
COUNT(CASE WHEN login_week = first_login + 7 THEN 1 END) AS week_7,
COUNT(CASE WHEN login_week = first_login + 8 THEN 1 END) AS week_8,
COUNT(CASE WHEN login_week = first_login + 9 THEN 1 END) AS week_9,
COUNT(CASE WHEN login_week = first_login + 10 THEN 1 END) AS week_10,
COUNT(CASE WHEN login_week = first_login + 11 THEN 1 END) AS week_11,
```

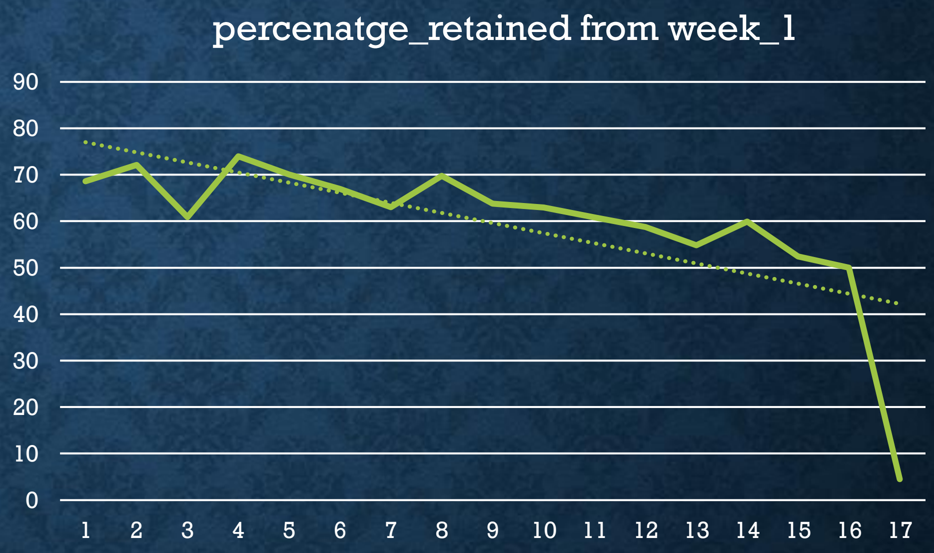
```
COUNT(CASE WHEN login_week = first_login + 12 THEN 1 END) AS week_12,
COUNT(CASE WHEN login_week = first_login + 13 THEN 1 END) AS week_13,
COUNT(CASE WHEN login_week = first_login + 14 THEN 1 END) AS week_14,
COUNT(CASE WHEN login_week = first_login + 15 THEN 1 END) AS week_15,
COUNT(CASE WHEN login_week = first_login + 16 THEN 1 END) AS week_16,
COUNT(CASE WHEN login_week = first_login + 17 THEN 1 END) AS week_17
FROM
(
SELECT distinct user_id,
week(occurred_at) AS login_week,
MIN(week(occurred_at)) OVER (PARTITION BY user_id) AS first_login
FROM events
WHERE event_name = 'login’
) subquery
GROUP BY first_login
ORDER BY first_login;
```

Output:

first_login	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
17	663	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77
18	596	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4
19	427	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0
20	358	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0
21	317	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0
22	326	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0
23	328	219	138	101	90	79	69	61	54	47	35	30	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
25	305	218	139	101	75	63	50	46	38	35	2	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
27	292	199	121	106	68	53	40	36	1	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
29	270	186	102	65	47	40	1	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
31	215	145	76	57	1	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
33	286	202	9	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
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Insights:


percenatge_retained from week_0
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18


percenatge_retained from week_1
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

4. Weekly Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.

Our task: Calculate the weekly engagement per device?

Query:

```
SELECT week(occurred_at) AS weeknum,
COUNT(DISTINCT user_id) AS weekly_users, device
FROM events
WHERE event_type = 'engagement’
AND event_name = 'login’
GROUP BY weeknum, device
ORDER BY 1;
```

Output:

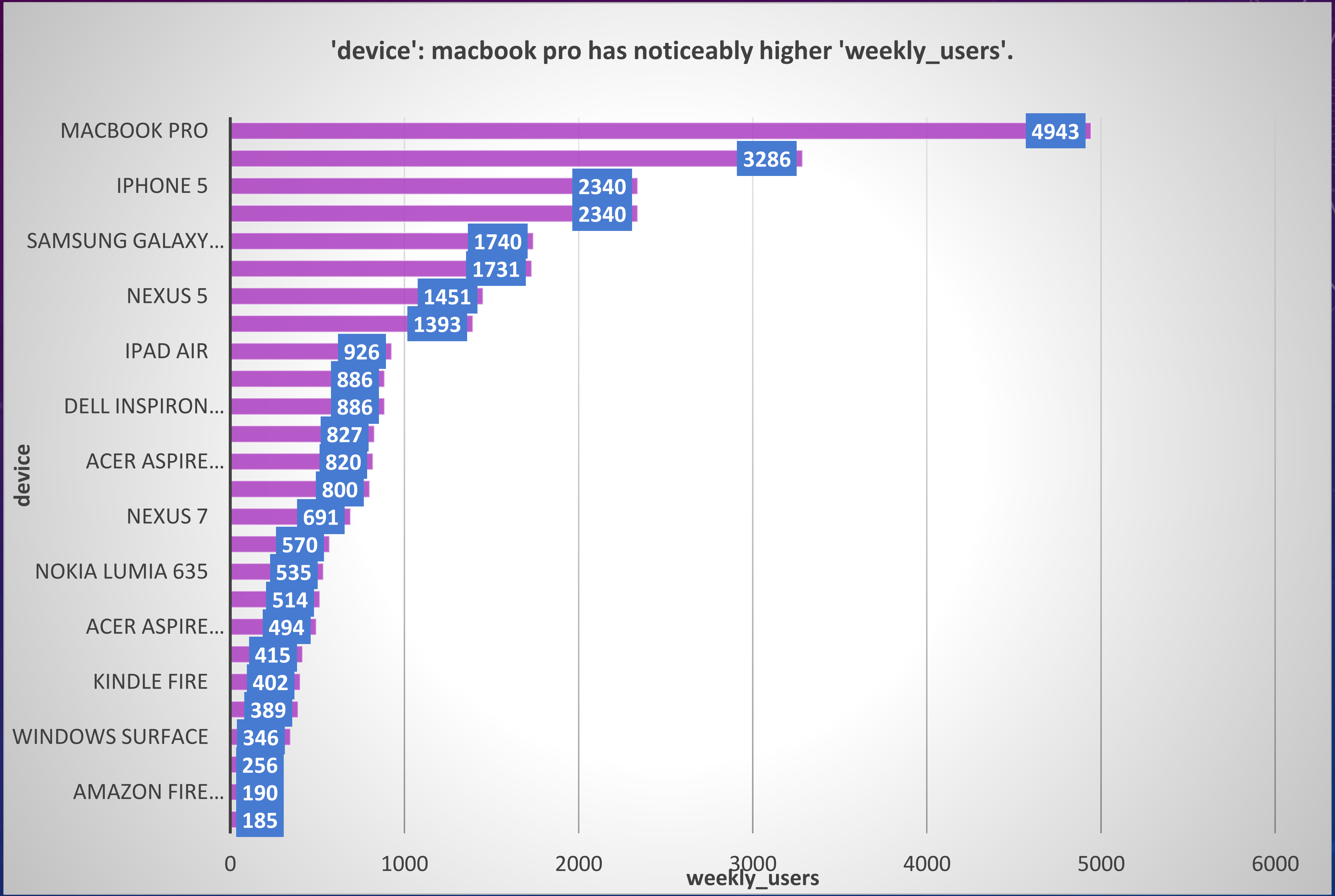
casestudy-2 task-4.xlsx

The output is too big to be fit in the slide so the link is provided to view the output

In the upcoming slides we can see the insights derived from the above output data

Showing insights for 'weekly_users'.	
device	**Sum of weekly_users**
macbook pro	4943
lenovo thinkpad	3286
iphone 5	2340
macbook air	2340
samsung galaxy s4	1740
dell inspiron notebook	1731
nexus 5	1451
iphone 5s	1393
ipad air	926
iphone 4s	886
dell inspiron desktop	886
asus chromebook	827
acer aspire notebook	820
hp pavilion desktop	800
nexus 7	691
ipad mini	570
nokia lumia 635	535
nexus 10	514
acer aspire desktop	494
htc one	415
kindle fire	402
mac mini	389
windows surface	346
samsung galaxy note	256
amazon fire phone	190
samsumg galaxy tablet	185
Grand Total	**29356**

device	*User percentage of Each Device*
macbook pro	16.84%
lenovo thinkpad	11.19%
iphone 5	7.97%
macbook air	7.97%
samsung galaxy s4	5.93%
dell inspiron notebook	5.90%
nexus 5	4.94%
iphone 5s	4.75%
ipad air	3.15%
iphone 4s	3.02%
dell inspiron desktop	3.02%
asus chromebook	2.82%
acer aspire notebook	2.79%
hp pavilion desktop	2.73%
nexus 7	2.35%
ipad mini	1.94%
nokia lumia 635	1.82%
nexus 10	1.75%
acer aspire desktop	1.68%
htc one	1.41%
kindle fire	1.37%
mac mini	1.33%
windows surface	1.18%
samsung galaxy note	0.87%
amazon fire phone	0.65%
samsumg galaxy tablet	0.63%
Grand Total	100.00%



'device': macbook pro has noticeably higher 'weekly_users'.
MACBOOK PRO
IPHONE 5
SAMSUNG GALAXY...
NEXUS 5
IPAD AIR
DELL INSPIRON...
ACER ASPIRE...
NEXUS 7
NOKIA LUMIA 635
ACER ASPIRE...
KINDLE FIRE
WINDOWS SURFACE
AMAZON FIRE...
4943
3286
2340
2340
1740
1731
1451
1393
926
886
886
827
820
800
691
570
535
514
494
415
402
389
346
256
190
185
device
0
1000
2000
3000
4000
5000
6000
weekly_users

5. Email Engagement: Users engaging with the email service.

Our task: Calculate the email engagement metrics?

Query:

```
SELECT  week,  users_who_receive_weekly_digest, no_of_users_opened,
ROUND((no_of_users_opened) * (100)/users_who_receive_weekly_digest) AS percentage_of_users_opened,  no_of_users_clickthrough,
ROUND((no_of_users_clickthrough) * (100)/users_who_receive_weekly_digest) AS percentage_of_users_clickthrough,
no_of_users_who_reengaged,
ROUND((no_of_users_who_reengaged)*(100)/users_who_receive_weekly_digest) AS percentage_of_users_who_reengaged
FROM
(SELECT  WEEK(occurred_at) AS week,
COUNT(CASE WHEN action = 'sent_weekly_digest' THEN 1 END) AS users_who_receive_weekly_digest,
COUNT(CASE WHEN action = 'email_open' THEN 1 END) AS no_of_users_opened,
COUNT(CASE WHEN action = 'email_clickthrough' THEN 1 END) AS no_of_users_clickthrough,
COUNT(CASE WHEN action = 'sent_reengagement_email' THEN 1 END) AS no_of_users_who_reengaged
FROM
email_events
GROUP BY week
ORDER BY week) subquery;
```

Output:

week	users_who_receive_weekly_digest	No_of_users_opened	percentage_of_users_opened	no_of_users_clickthrough	percentage_of_users_clickthrough	no_of_users_who_reengaged	percentage_of_users_who_reengaged
17	908	310	34	166	18	73	8
18	2602	912	35	430	17	157	6
19	2665	972	36	477	18	173	6
20	2733	1004	37	507	19	191	7
21	2822	1014	36	443	16	164	6
22	2911	987	34	488	17	192	7
23	3003	1075	36	538	18	197	7
24	3105	1155	37	554	18	226	7
25	3207	1096	34	530	17	196	6
26	3302	1165	35	556	17	219	7
27	3399	1228	36	621	18	213	6
28	3499	1250	36	599	17	213	6
29	3592	1219	34	590	16	213	6
30	3706	1383	37	630	17	231	6
31	3793	1351	36	445	12	222	6
32	3897	1337	34	418	11	200	5
33	4012	1432	36	490	12	264	7
34	4111	1528	37	490	12	261	6
35	0	41	NULL	38	NULL	48	NULL

Insights:

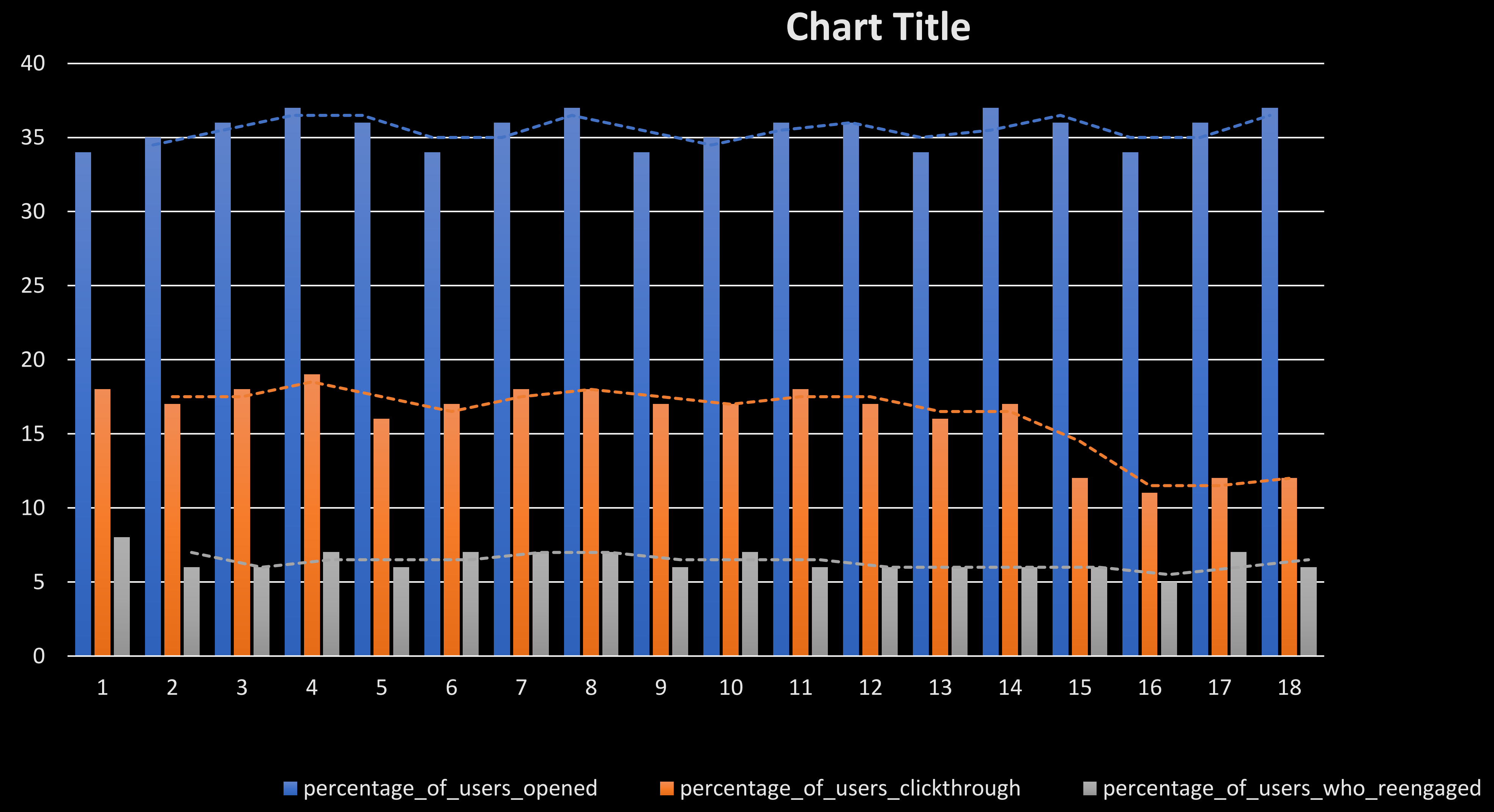


Chart Title
40
35
30
25
20
15
10
5
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
percentage_of_users_opened
percentage_of_users_clickthrough
percentage_of_users_who_reengaged

Result:

We have successfully analysed the data from each case study and provided insights. During this project, we gained knowledge of windows functions, subqueries and how to apply them for the required solutions. We also used Excel and gained the knowledge of charts and tables. We also learned how to use INFILE command to insert csv files as table into a database.

Thank You