OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

PROJECT DONE BY PRACHI RANJAN

PROJECT DESCRIPTION

The project is about finding out valuable insights that can help improve the company's operations and understand sudden changes in key metrics. We analyze this data on the following points:

Case Study 1: Job Data Analysis

- **Jobs Reviewed Over Time**
- **Throughput Analysis**
- **Language Share Analysis**
- **Duplicate Rows Detection**

PROJECT DESCRIPTION

Case Study 2: Investigating Metric Spike

- **Weekly User Engagement**
- **User Growth Analysis**
- **Weekly Retention Analysis**
- **Weekly Engagement Per Device**
- **Email Engagement Analysis**

Software used:-

- **MySQL Workbench 8.0 CE**

CASE STUDY 1: JOB DATA ANALYSIS

JOBS REVIEWED OVER TIME: CALCULATE THE NUMBER OF JOBS REVIEWED PER HOUR FOR EACH DAY IN NOVEMBER 2020.

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

- We will use **select** statement from job_data table.
- Then we will use **count** function in **distinct** job_id column and divide by (30 days * 24 hours) to get number of jobs reviewed per hour for each day.

QUERY:-

```
select count(distinct job_id)/(30*24) as no_of_jobs_reviewed
from job_data;
```

CASE STUDY 1: JOB DATA ANALYSIS

JOBS REVIEWED OVER TIME: CALCULATE THE NUMBER OF JOBS REVIEWED PER HOUR FOR EACH DAY IN NOVEMBER 2020.

Output/Results:-

no_of_jobs_reviewed
0.0083

The number of jobs reviewed per hour for each day in November 2020 is 0.0083

CASE STUDY 1: JOB DATA ANALYSIS

THROUGHPUT ANALYSIS: CALCULATE THE 7-DAY ROLLING AVERAGE OF THROUGHPUT (NUMBER OF EVENTS PER SECOND).

Task B: 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.

- We will use **select** statement , **count** function on **distinct** job_id column and **avg** function in count(distinct job_id) from job_data table.
- By using **ROWS** function we will be considering the rows between 6 preceding and current row.
- Then we will get 7-day rolling average of throughput.
- We will use **group by** clause in ds column.
- By using **order by** clause in ds column we will sort the order.

CASE STUDY 1: JOB DATA ANALYSIS

THROUGHPUT ANALYSIS: CALCULATE THE 7-DAY ROLLING AVERAGE OF THROUGHPUT (NUMBER OF EVENTS PER SECOND).

QUERY:-

```
select ds,
count(distinct job_id) as jobs_reviewed,
avg(count(distinct job_id)) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW) as throughput_7_rolling_avg
from job_data
group by ds
order by ds;
```

CASE STUDY 1: JOB DATA ANALYSIS

THROUGHPUT ANALYSIS: CALCULATE THE 7-DAY ROLLING AVERAGE OF THROUGHPUT (NUMBER OF EVENTS PER SECOND).

Output/Results:-

ds	jobs_reviewed	throughput_7_rolling_avg
25-11-2020	1	1
26-11-2020	1	1
27-11-2020	1	1
28-11-2020	2	1.25
29-11-2020	1	1.2
30-11-2020	2	1.3333

CASE STUDY 1: JOB DATA ANALYSIS

LANGUAGE SHARE ANALYSIS: CALCULATE THE PERCENTAGE SHARE OF EACH LANGUAGE IN THE LAST 30 DAYS.

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

- We will use **select** statement to select job_id, language column from job_data table.
- Then we will use **count** function in language column and divide by total using **sum(count(*)) over().**
- Using **group by** in language column we will get percentage share of each language.

QUERY:-

```
select job_id, language, (count(language)/ sum(count(*)) over())*100 as
percentage_share_of_each_language
from job_data
group by language;
```

CASE STUDY 1: JOB DATA ANALYSIS

LANGUAGE SHARE ANALYSIS: CALCULATE THE PERCENTAGE SHARE OF EACH LANGUAGE IN THE LAST 30 DAYS.

Output/Results:-

job_id	language	percentage_share_of_each_language
21	English	12.5
22	Arabic	12.5
23	Persian	37.5
25	Hindi	12.5
11	French	12.5
20	Italian	12.5

CASE STUDY 1: JOB DATA ANALYSIS

DUPLICATE ROWS DETECTION: IDENTIFY DUPLICATE ROWS IN THE DATA.

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

- First we will decide in which column we need to find duplicate rows.
- Then we will use **row_number()** function to find the row numbers which are having the same value.
- We will use **partition** on row_number function over the column which we decided i.e job_id.
- Then we will use **where** function to find the row_num having value greater than 1.

QUERY:-

```
select * from (select *, row_number() over (partition by job_id) as no_of_rowsfrom job_data)
awhere no_of_rows>1;
select * from (select *, row_number() over (partition by job_id) as no_of_rows
from job_data) a
where no_of_rows>1;
```

CASE STUDY 1: JOB DATA ANALYSIS

DUPLICATE ROWS DETECTION: IDENTIFY DUPLICATE ROWS IN THE DATA.

Output/Results:-

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

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY USER ENGAGEMENT: MEASURE THE ACTIVENESS OF USERS ON A WEEKLY BASIS.

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

- We will use **select** statement, **week** function in occurred_at column to extract number of weeks and **count** function in **distinct** user_id column to get number of users from events table.
- Using **group by** clause in no_of_week we will get weekly user engagement.

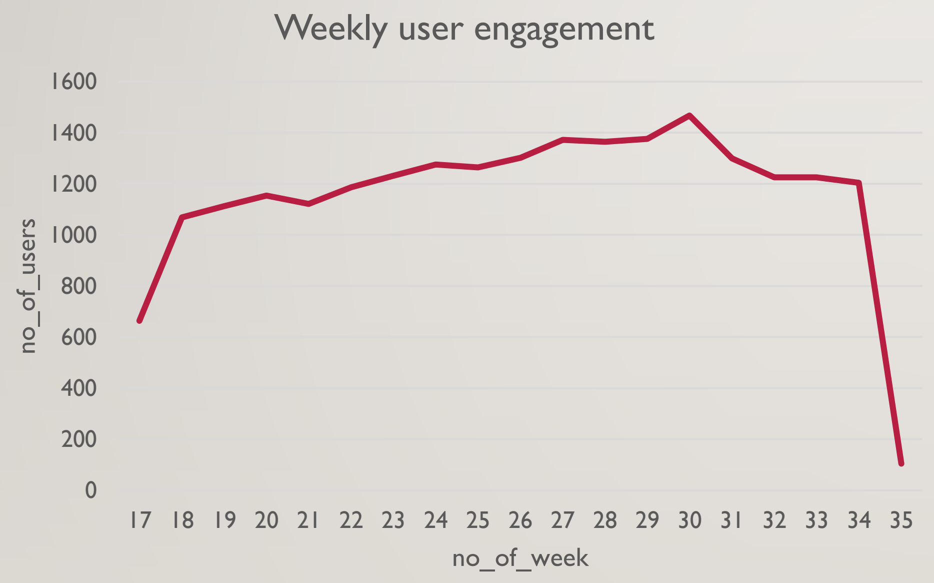
QUERY:-

```
select week(occurred_at) as no_of_week, count(distinct user_id) as no_of_users
from events
group by no_of_week;
```

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY USER ENGAGEMENT: MEASURE THE ACTIVENESS OF USERS ON A WEEKLY BASIS.

Output/Results:-



no_of_week	no_of_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

CASE STUDY 2: INVESTIGATING METRIC SPIKE

USER GROWTH ANALYSIS: ANALYZE THE GROWTH OF USERS OVER TIME FOR A PRODUCT.

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

- We will use **extract** to extract year and week from activated_at column from users table.
- Using **group by** clause we will group extracted year and week on the basis of year and week number.
- Then we will use **order by** to sort the output based on extracted year and week
- We will use **sum, over** and **row** function **between unbounded preceding and current row** to find cumm_active_users.

CASE STUDY 2: INVESTIGATING METRIC SPIKE

USER GROWTH ANALYSIS: ANALYZE THE GROWTH OF USERS OVER TIME FOR A PRODUCT.

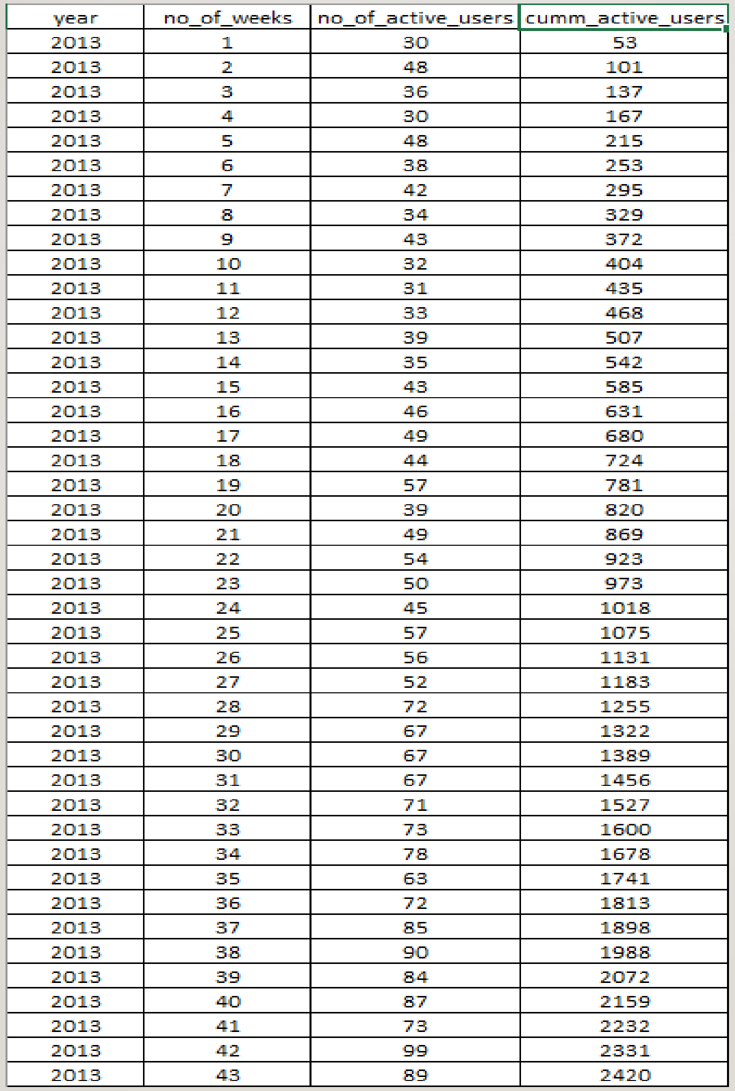
QUERY:-

```
select year, no_of_weeks, no_of_active_users, sum(no_of_active_users) over(order by
year, no_of_weeks ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as
cumm_active_users
from (select extract(year from activated_at) as year,
extract(week from activated_at) as no_of_weeks,
count(distinct user_id) as no_of_active_users
from users
group by year, no_of_weeks
order by year, no_of_weeks) a;
```

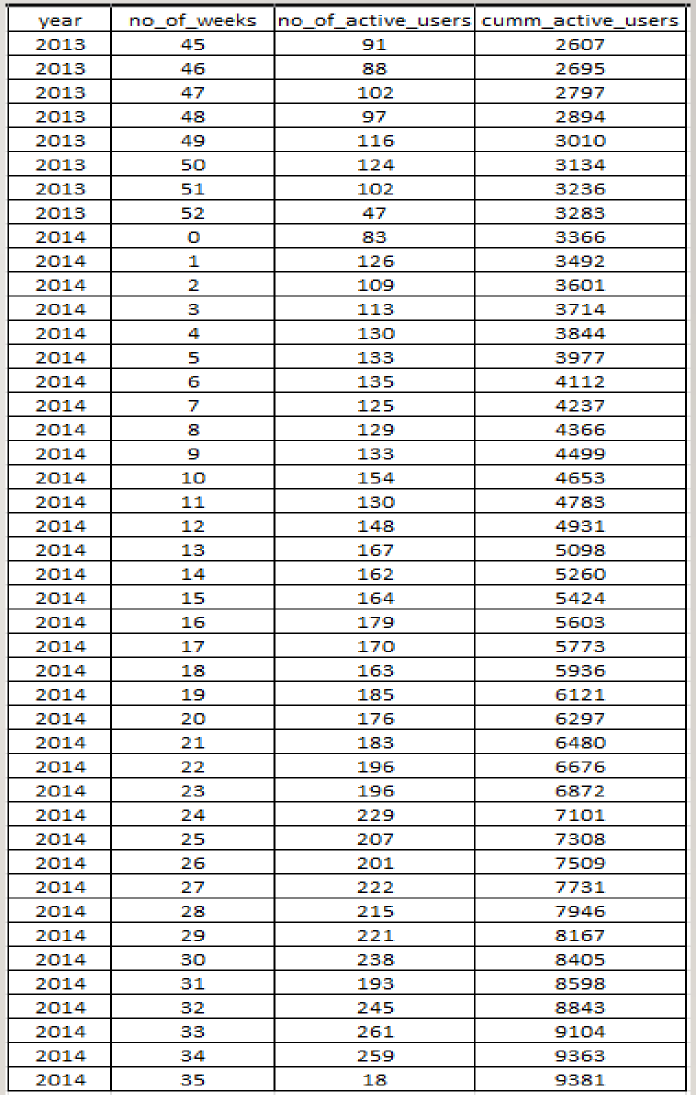
CASE STUDY 2: INVESTIGATING METRIC SPIKE

USER GROWTH ANALYSIS: ANALYZE THE GROWTH OF USERS OVER TIME FOR A PRODUCT.

Output/Results:-



year	no_of_weeks	no_of_active_users	cumm_active_users
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724
2013	19	57	781
2013	20	39	820
2013	21	49	869
2013	22	54	923
2013	23	50	973
2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813
2013	37	85	1898
2013	38	90	1988
2013	39	84	2072
2013	40	87	2159
2013	41	73	2232
2013	42	99	2331
2013	43	89	2420



year	no_of_weeks	no_of_active_users	cumm_active_users
2013	45	91	2607
2013	46	88	2695
2013	47	102	2797
2013	48	97	2894
2013	49	116	3010
2013	50	124	3134
2013	51	102	3236
2013	52	47	3283
2014	0	83	3366
2014	1	126	3492
2014	2	109	3601
2014	3	113	3714
2014	4	130	3844
2014	5	133	3977
2014	6	135	4112
2014	7	125	4237
2014	8	129	4366
2014	9	133	4499
2014	10	154	4653
2014	11	130	4783
2014	12	148	4931
2014	13	167	5098
2014	14	162	5260
2014	15	164	5424
2014	16	179	5603
2014	17	170	5773
2014	18	163	5936
2014	19	185	6121
2014	20	176	6297
2014	21	183	6480
2014	22	196	6676
2014	23	196	6872
2014	24	229	7101
2014	25	207	7308
2014	26	201	7509
2014	27	222	7731
2014	28	215	7946
2014	29	221	8167
2014	30	238	8405
2014	31	193	8598
2014	32	245	8843
2014	33	261	9104
2014	34	259	9363
2014	35	18	9381

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY RETENTION ANALYSIS: ANALYZE THE RETENTION OF USERS ON A WEEKLY BASIS AFTER SIGNING UP FOR A PRODUCT.

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

The weekly retention of users-sign up cohort can be calculated by two means i.e. either for the entire column of occurred_at of the events table or by specifying the week number (18 to 35)

- First we will use **extract** function to extract week from occurred_at column from events table.
- Then we will select the rows in which **event_type = 'signup_flow' and event_name = 'complete_signup'.**
- After that we will use **left join on** user_id to join the tables in which event_type = 'engagement'.
- Using **group by** clause in user_id we will get weekly retention for each user.
- Then we will use **order by** to sort the output on the basis of user_id.

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY RETENTION ANALYSIS: ANALYZE THE RETENTION OF USERS ON A WEEKLY BASIS AFTER SIGNING UP FOR A PRODUCT.

QUERY:- (entire column of occurred_at)

```
Select distinct user_id, count(user_id) as no_of_user,
sum(case when retention_week = 1 then 1 else 0 end) as per_week_retention
from (select a.user_id,
        a.signup_week,  b.engagement_week,
        b.engagement_week - a.signup_week as retention_week
from  ((select distinct user_id, extract(week from occurred_at) as signup_week
          from events
          where event_type = 'signup_flow' and event_name = 'complete_signup') a
           left join (select distinct user_id, extract(week from occurred_at) as engagement_week
                         from events
                         where event_type = 'engagement') b
            on a.user_id = b.user_id)) d
group by user_id
order by user_id;
```

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY RETENTION ANALYSIS: ANALYZE THE RETENTION OF USERS ON A WEEKLY BASIS AFTER SIGNING UP FOR A PRODUCT.

Output/Results:- (entire column of occurred_at)

Link for the result

https://drive.google.com/file/d/19oYzGzQm7nZNv8209cRSXLsFswLTQUd0/view?usp=sharing

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY RETENTION ANALYSIS: ANALYZE THE RETENTION OF USERS ON A WEEKLY BASIS AFTER SIGNING UP FOR A PRODUCT.

QUERY:- (week number as 18)

```
Select distinct user_id, count(user_id) as no_of_user,
sum(case when retention_week = 1 then 1 else 0 end) as per_week_retention
from (select a.user_id,
          a.signup_week,  b.engagement_week,
          b.engagement_week - a.signup_week as retention_week
from  ((select distinct user_id, extract(week from occurred_at) as signup_week
             from events
             where event_type = 'signup_flow’
            and event_name = 'complete_signup’
             and extract(week from occurred_at) = 18) a
              left join (select distinct user_id, extract(week from occurred_at) as engagement_week
                              from events
                              where event_type = 'engagement’) b
               on a.user_id = b.user_id)) d
group by user_id
order by user_id;
```

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY RETENTION ANALYSIS: ANALYZE THE RETENTION OF USERS ON A WEEKLY BASIS AFTER SIGNING UP FOR A PRODUCT.

Output/Results:- (week number as 18)

Link for the result

https://drive.google.com/file/d/19oYzGzQm7nZNv8209cRSXLsFswLTQUd0/view?usp=sharing

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY ENGAGEMENT PER DEVICE: MEASURE THE ACTIVENESS OF USERS ON A WEEKLY BASIS PER DEVICE.

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

- We will extract year and week from occurred_at column from events table.
- Then we will select device column and use count function to get number of users.
- Using **where** clause we will select rows where **event_type='engagement'**
- We will use **group by** and **order by** function to group and order the output based on year, no_of_weeks, device

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY ENGAGEMENT PER DEVICE: MEASURE THE ACTIVENESS OF USERS ON A WEEKLY BASIS PER DEVICE.

QUERY:-

```
select year(occurred_at) as year,
week(occurred_at) as no_of_weeks,
device,
count(distinct user_id) as no_of_user
from events
where event_type='engagement’
group by 1,2,3
order by 1,2,3;
```

CASE STUDY 2: INVESTIGATING METRIC SPIKE

WEEKLY ENGAGEMENT PER DEVICE: MEASURE THE ACTIVENESS OF USERS ON A WEEKLY BASIS PER DEVICE.

Output/Results:-

Link for the result

https://drive.google.com/file/d/1MSLN2L7Tp5s22GruL0hRZJOGbOqIzpZh/view?usp=sharing

CASE STUDY 2: INVESTIGATING METRIC SPIKE

EMAIL ENGAGEMENT ANALYSIS: ANALYZE HOW USERS ARE ENGAGING WITH THE EMAIL SERVICE.

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

- First we will categorize the action into **'email_opened', 'email_sent', 'email_clicked'** using **when, case, then** functions.
- We will divide **sum** of category 'email_opened' and sum of category 'email_sent' and multiply by 100 and put the name as **email_opening_rate.**
- Then we will divide **sum** of category 'email_clicked' and sum of category 'email_sent' and multiply by 100 and put the name as **email_clicking_rate**.

Categorizing of action:-

- **email_opened** = ('email_open')
- **email_sent** = ('sent_weekly_digest','sent_reengagement_email')
- **email_clicked** = ('email_clickthrough')

CASE STUDY 2: INVESTIGATING METRIC SPIKE

EMAIL ENGAGEMENT ANALYSIS: ANALYZE HOW USERS ARE ENGAGING WITH THE EMAIL SERVICE.

QUERY:-

```
Select 100*SUM(CASE when email_action = 'email_opened' then 1 else 0 end) / SUM(CASE when email_action =
'email_sent' then 1 else 0 end) as email_opening_rate,
100*SUM(CASE when email_action = 'email_clicked' then 1 else 0 end) / SUM(CASE when email_action = 'email_sent'
then 1 else 0 end) as email_clicking_rate
from
(select *,
CASE WHEN action in ('email_open’)
then 'email_opened’
action in ('sent_weekly_digest','sent_reengagement_email’)
then 'email_sent’
WHEN action in ('email_clickthrough’)
then 'email_clicked’
end as email_action
from email_events) a;
```

CASE STUDY 2: INVESTIGATING METRIC SPIKE

EMAIL ENGAGEMENT ANALYSIS: ANALYZE HOW USERS ARE ENGAGING WITH THE EMAIL SERVICE.

Output/Results:-

email_opening_rate	email_clicking_rate
33.5834	14.7899