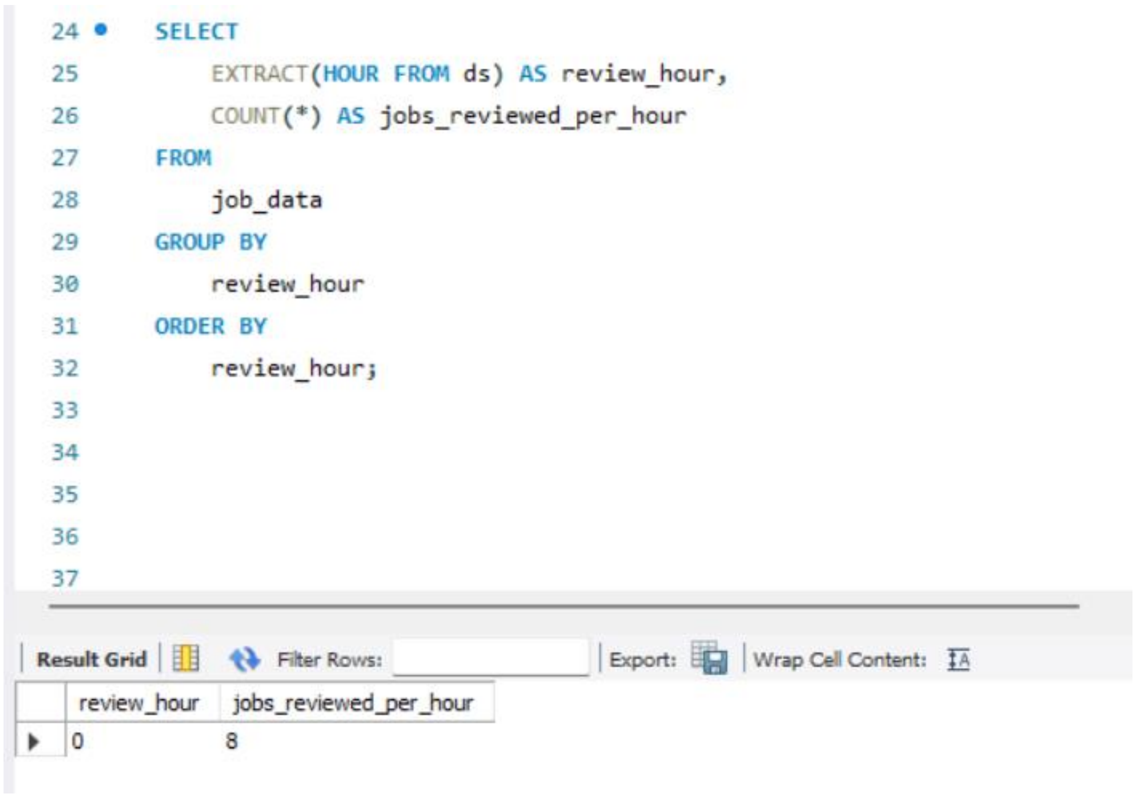
TASK 3 :- Operation Analytics and Investigating Metric Spike.

Case Study 1: Job Data Analysis

A:- Jobs Reviewed Over Time:

```
SELECT
    EXTRACT(HOUR FROM ds) AS review_hour,
    COUNT(*) AS jobs_reviewed_per_hour
FROM
    job_data
GROUP BY
    review_hour
ORDER BY
    review_hour;
```



```
SELECT
    EXTRACT(HOUR FROM ds) AS review_hour,
    COUNT(*) AS jobs_reviewed_per_hour
FROM
    job_data
GROUP BY
    review_hour
ORDER BY
    review_hour;
```

review_hour	jobs_reviewed_per_hour
0	8

B:-Throughput Analysis

SELECT ds as date_of_review, jobs_reviewed, AVG(jobs_reviewed)

OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS

throughput_7_rolling_average

FROM

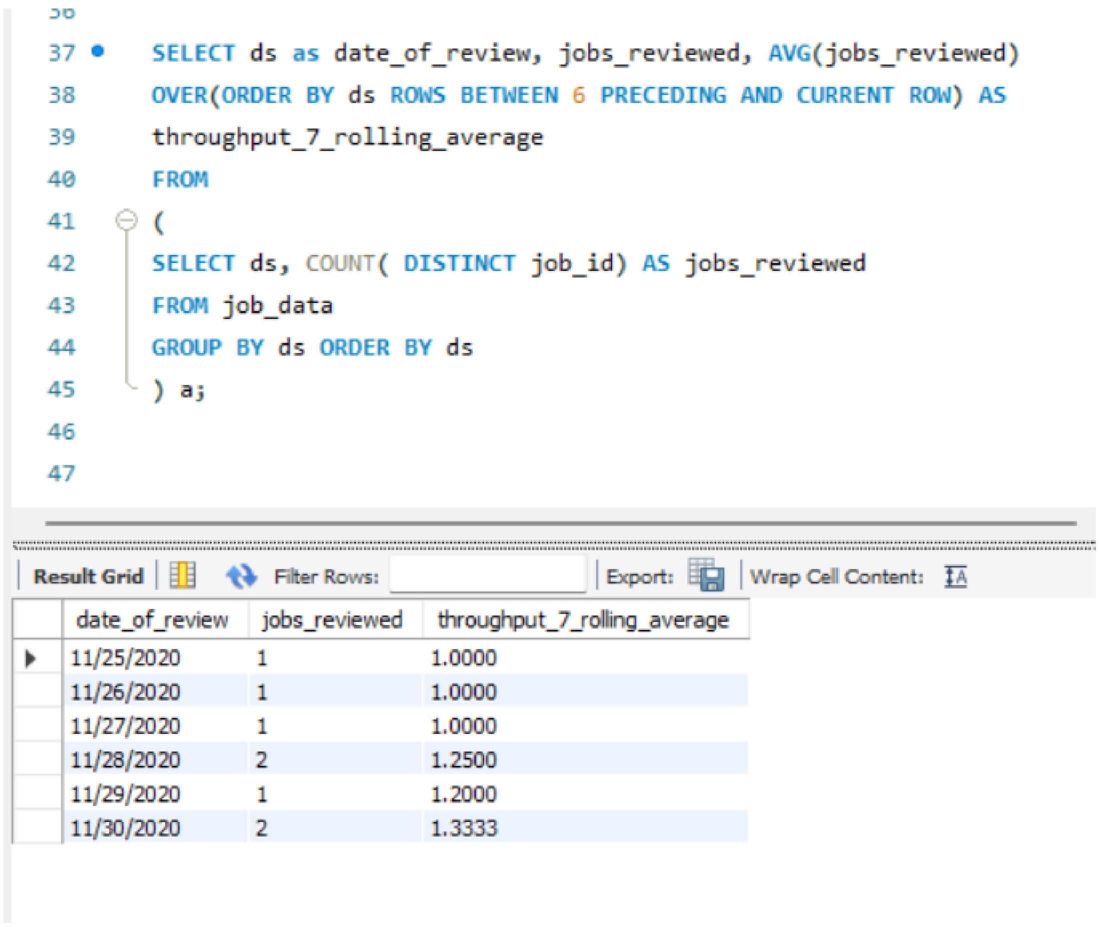
(

SELECT ds, COUNT(DISTINCT job_id) AS jobs_reviewed

FROM job_data

GROUP BY ds ORDER BY ds

) a;



```
SELECT ds as date_of_review, jobs_reviewed, AVG(jobs_reviewed)
OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS
throughput_7_rolling_average
FROM
(
SELECT ds, COUNT( DISTINCT job_id) AS jobs_reviewed
FROM job_data
GROUP BY ds ORDER BY ds
) a;
```

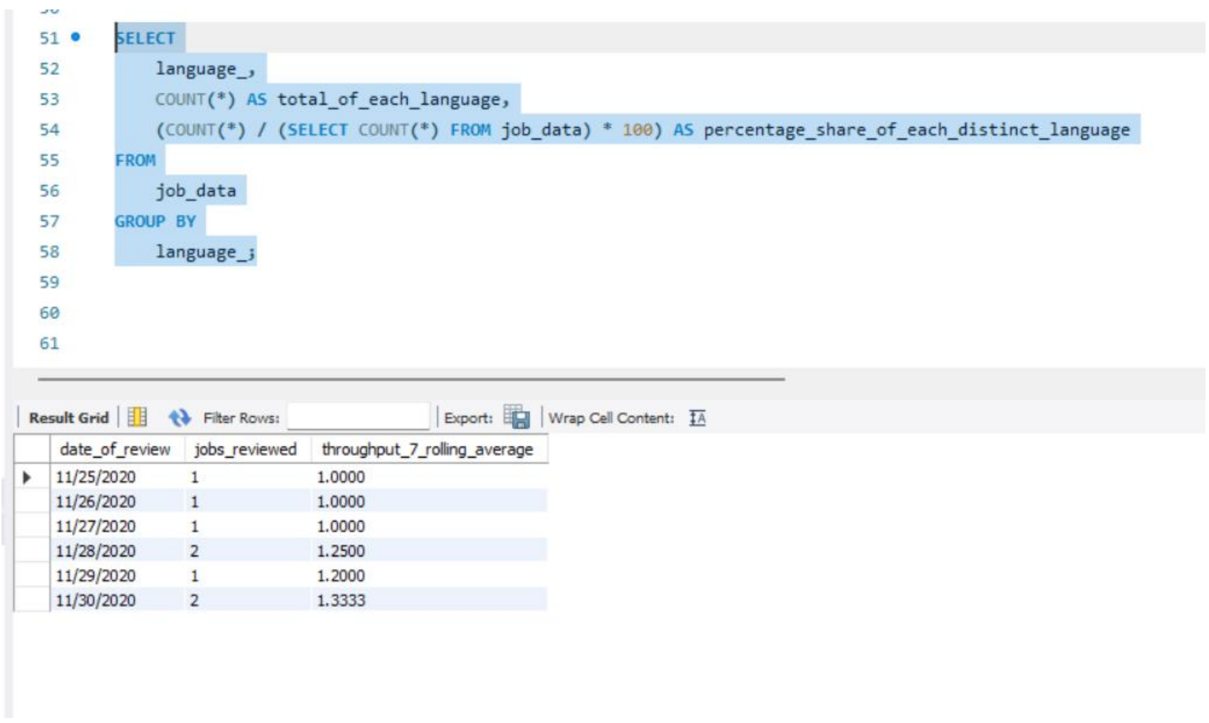
date_of_review	jobs_reviewed	throughput_7_rolling_average
11/25/2020	1	1.0000
11/26/2020	1	1.0000
11/27/2020	1	1.0000
11/28/2020	2	1.2500
11/29/2020	1	1.2000
11/30/2020	2	1.3333

For throughput I prefer using 7-day rolling average than the daily metric for throughput for the following reasons.

1. Smoothens out short term fluctuations, which offers a clear view of trend.
2. Its helps us to identify more stable and sustained patterns.

C:- Language Share Analysis:

```
SELECT
    language_,
    COUNT(*) AS total_of_each_language,
    (COUNT(*) / (SELECT COUNT(*) FROM job_data) * 100) AS
percentage_share_of_each_distinct_language
FROM
    job_data
GROUP BY
    language_;
```

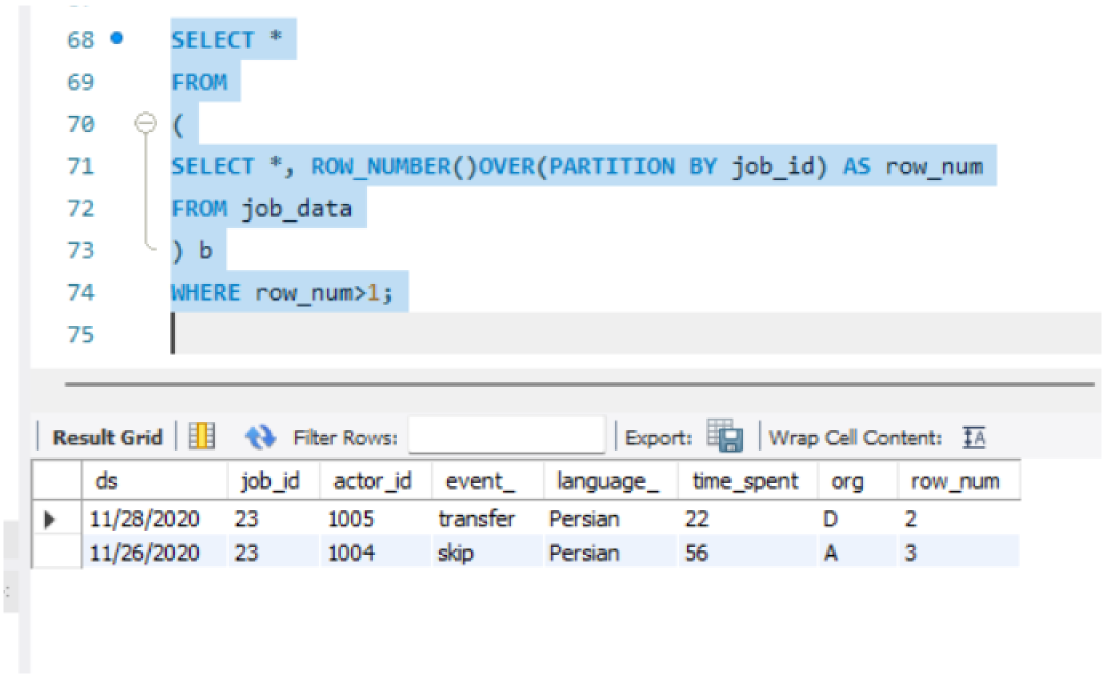


```
SELECT
    language_,
    COUNT(*) AS total_of_each_language,
    (COUNT(*) / (SELECT COUNT(*) FROM job_data) * 100) AS percentage_share_of_each_distinct_language
FROM
    job_data
GROUP BY
    language_;
```

date_of_review	jobs_reviewed	throughput_7_rolling_average
11/25/2020	1	1.0000
11/26/2020	1	1.0000
11/27/2020	1	1.0000
11/28/2020	2	1.2500
11/29/2020	1	1.2000
11/30/2020	2	1.3333

D:- Duplicate Rows Detection

```
SELECT *

FROM

(

SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num

FROM job_data

) b

WHERE row_num>1;
```



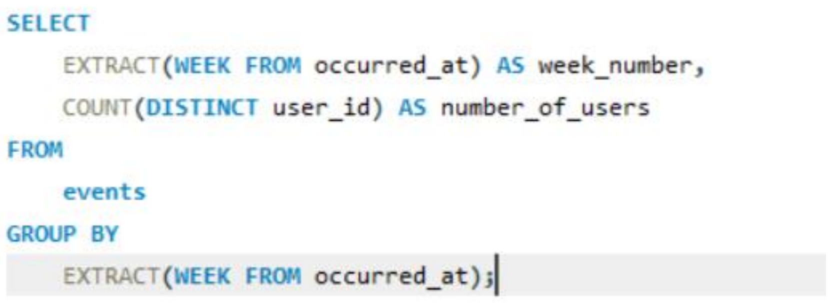
```
SELECT *
FROM
(
SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num
FROM job_data
) b
WHERE row_num>1;
```

ds	job_id	actor_id	event_	language_	time_spent	org	row_num
11/28/2020	23	1005	transfer	Persian	22	D	2
11/26/2020	23	1004	skip	Persian	56	A	3

Case Study 2: Investigating Metric Spike

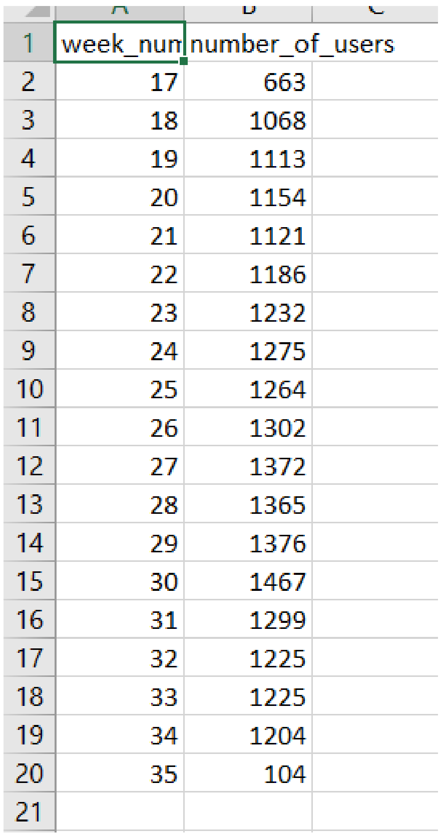
A:- Weekly User Engagement

```
SELECT
   EXTRACT(WEEK FROM occurred_at) AS week_number,
   COUNT(DISTINCT user_id) AS number_of_users
FROM
   events
GROUP BY
   EXTRACT(WEEK FROM occurred_at);
```



```
SELECT
    EXTRACT(WEEK FROM occurred_at) AS week_number,
    COUNT(DISTINCT user_id) AS number_of_users
FROM
    events
GROUP BY
    EXTRACT(WEEK FROM occurred_at);
```

OUTPUT:-

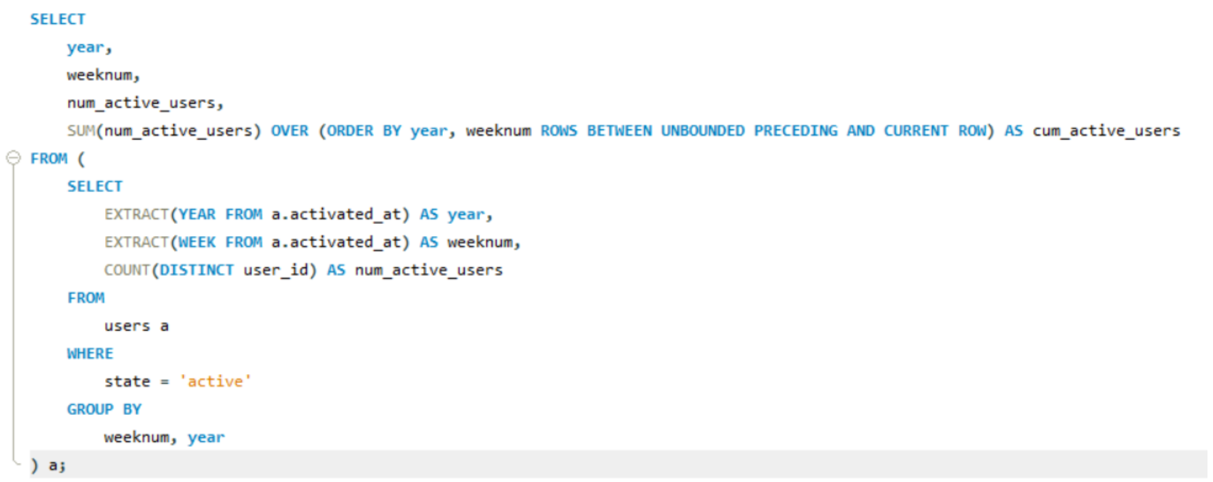


	A	B	C
1	week_num	number_of_users	
2	17	663	
3	18	1068	
4	19	1113	
5	20	1154	
6	21	1121	
7	22	1186	
8	23	1232	
9	24	1275	
10	25	1264	
11	26	1302	
12	27	1372	
13	28	1365	
14	29	1376	
15	30	1467	
16	31	1299	
17	32	1225	
18	33	1225	
19	34	1204	
20	35	104	
21			

B:- User Growth Analysis:

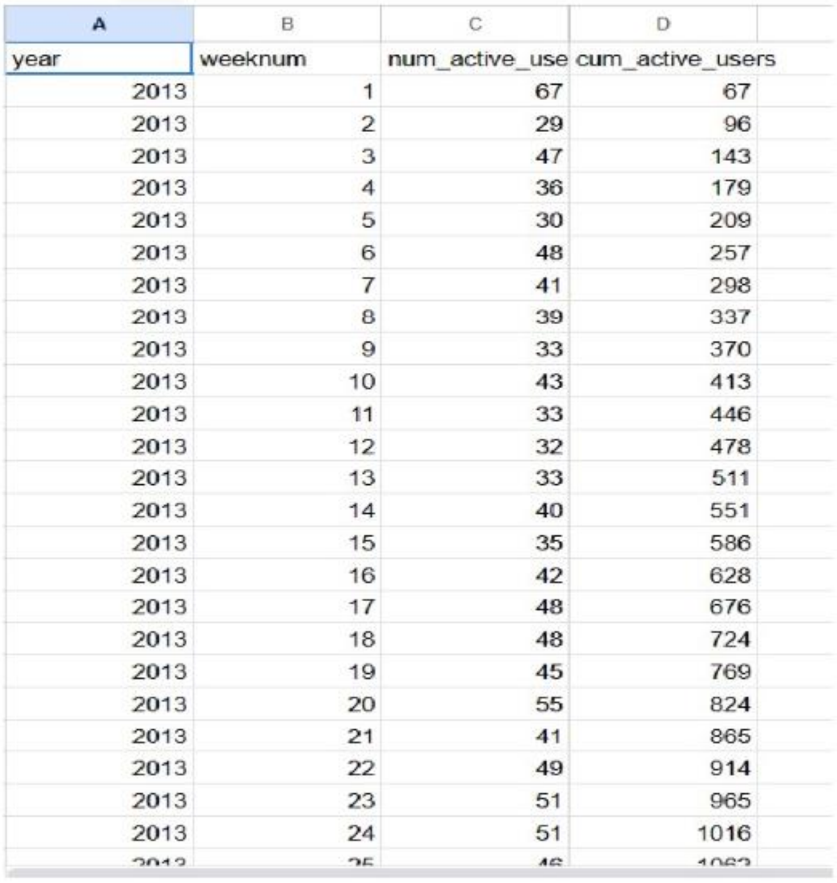
```
SELECT
    year,
    weeknum,
    num_active_users,
    SUM(num_active_users) OVER (ORDER BY year, weeknum ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW) AS cum_active_users
FROM (
    SELECT
```

```
        EXTRACT(YEAR FROM a.activated_at) AS year,
        EXTRACT(WEEK FROM a.activated_at) AS weeknum,
        COUNT(DISTINCT user_id) AS num_active_users
    FROM
        users a
    WHERE
        state = 'active'
    GROUP BY
        weeknum, year
) a;
```



```
SELECT
    year,
    weeknum,
    num_active_users,
    SUM(num_active_users) OVER (ORDER BY year, weeknum ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_users
FROM (
    SELECT
        EXTRACT(YEAR FROM a.activated_at) AS year,
        EXTRACT(WEEK FROM a.activated_at) AS weeknum,
        COUNT(DISTINCT user_id) AS num_active_users
    FROM
        users a
    WHERE
        state = 'active'
    GROUP BY
        weeknum, year
) a;
```

OUTPUT:-



A	B	C	D
year	weeknum	num_active_use	cum_active_users
2013	1	67	67
2013	2	29	96
2013	3	47	143
2013	4	36	179
2013	5	30	209
2013	6	48	257
2013	7	41	298
2013	8	39	337
2013	9	33	370
2013	10	43	413
2013	11	33	446
2013	12	32	478
2013	13	33	511
2013	14	40	551
2013	15	35	586
2013	16	42	628
2013	17	48	676
2013	18	48	724
2013	19	45	769
2013	20	55	824
2013	21	41	865
2013	22	49	914
2013	23	51	965
2013	24	51	1016
2013	25	46	1062

C:- Weekly Retention Analysis

```
SELECT
  user_id,
  COUNT(user_id) AS total_events,
  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
      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
      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;
```

OUTPUT:-

https://drive.google.com/file/d/1Eo2VDa03-6OtlEWidWr3vkH2nG1u_7HX/view?usp=drive_link

D:-Weekly Engagement Per Device:

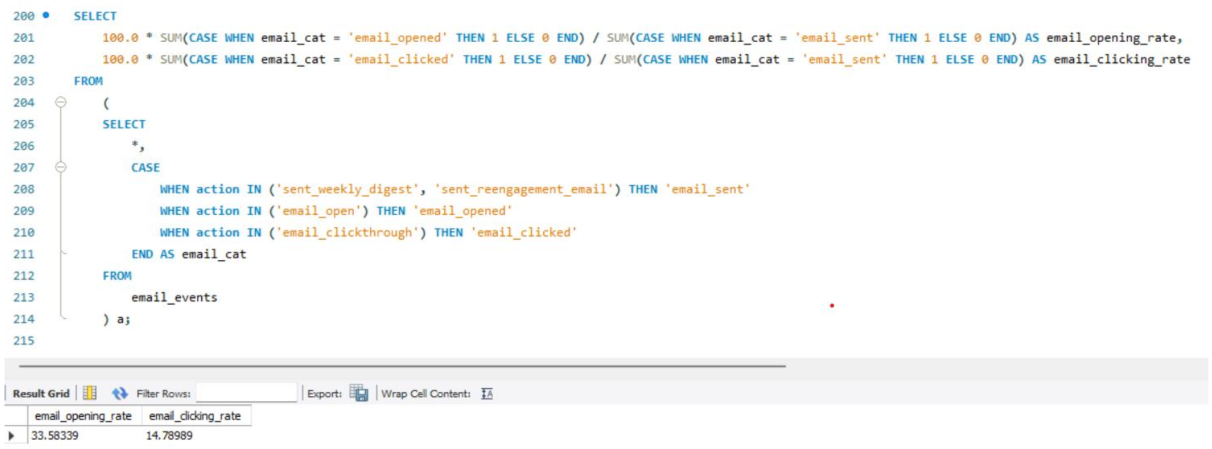
```
SELECT
   EXTRACT(YEAR FROM occurred_at) AS year_num,
   EXTRACT(WEEK FROM occurred_at) AS week_num,
   device,
   COUNT(DISTINCT user_id) AS no_of_users
FROM
   events
WHERE
   event_type = 'engagement'
GROUP BY
   1, 2, 3
ORDER BY
   1, 2, 3;
```

OUTPUT:-

https://drive.google.com/file/d/1BbKFOUYzrrStbSSjfslhcyLQxTeUuWoE/view?usp=drive_link

E:- Email Engagement Analysis

```
SELECT
  100.0 * SUM(CASE WHEN email_cat = 'email_opened' THEN 1 ELSE 0 END) / SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_opening_rate,
  100.0 * SUM(CASE WHEN email_cat = 'email_clicked' THEN 1 ELSE 0 END) / SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_clicking_rate
FROM
  (
  SELECT
    *,
    CASE
      WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email_sent'
      WHEN action IN ('email_open') THEN 'email_opened'
      WHEN action IN ('email_clickthrough') THEN 'email_clicked'
    END AS email_cat
  FROM
    email_events
  ) a;
```



```
SELECT
    100.0 * SUM(CASE WHEN email_cat = 'email_opened' THEN 1 ELSE 0 END) / SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_opening_rate,
    100.0 * SUM(CASE WHEN email_cat = 'email_clicked' THEN 1 ELSE 0 END) / SUM(CASE WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_clicking_rate
FROM
    (
    SELECT
        *,
        CASE
            WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email_sent'
            WHEN action IN ('email_open') THEN 'email_opened'
            WHEN action IN ('email_clickthrough') THEN 'email_clicked'
        END AS email_cat
    FROM
        email_events
    ) a;
```

email_opening_rate	email_clicking_rate
33.58339	14.78989