OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

BY : SAMIYA ALAM

PROJECT DESCRIPTION

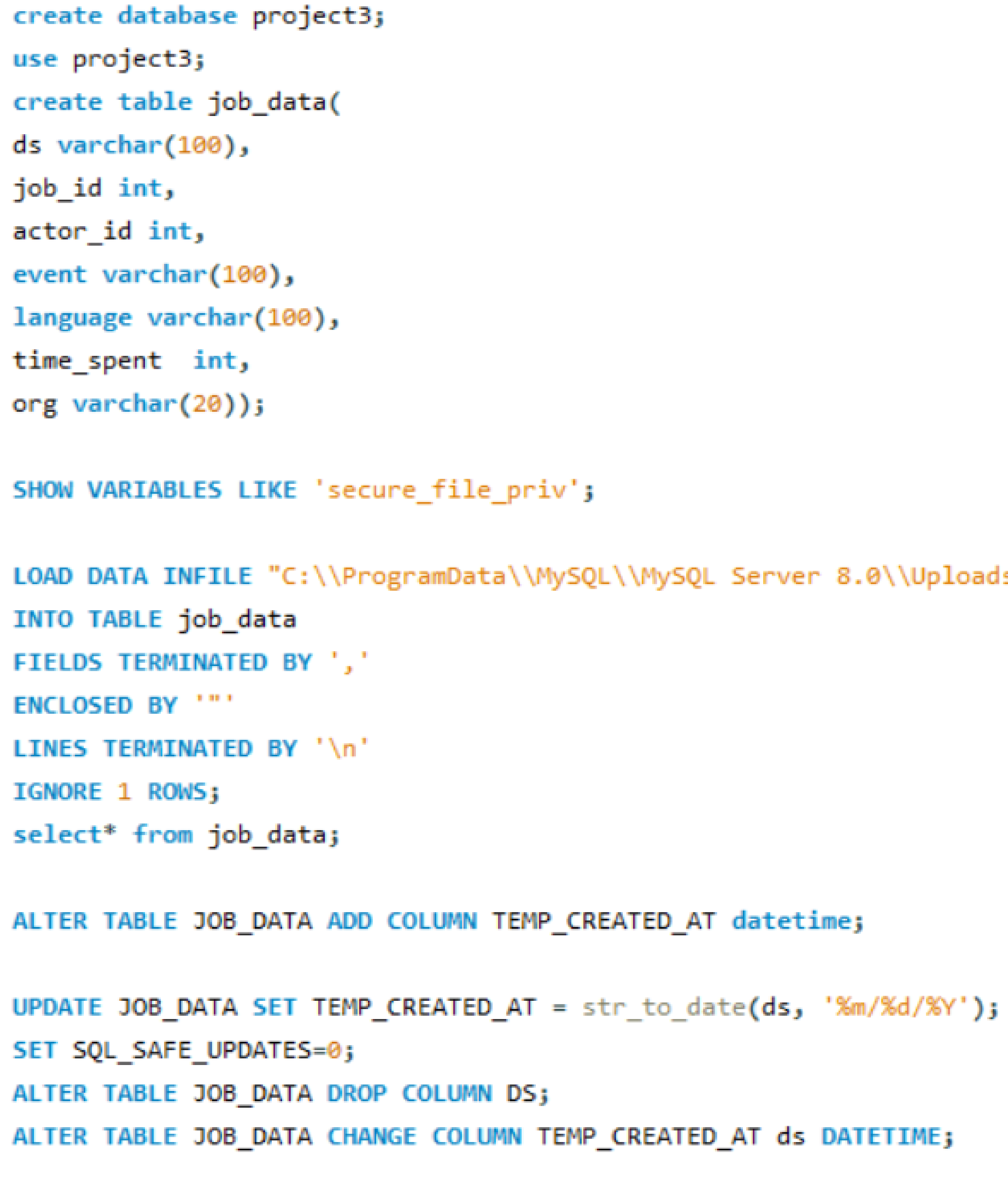
This project involves performing Operational Analytics for a hypothetical company similar to Microsoft. The primary objective is to analyze the company's end-to-end operations, identify areas for improvement, and investigate sudden changes in key metrics. As a Lead Data Analyst, I will utilize advanced SQL skills to derive valuable insights from various datasets and tables, addressing questions posed by different departments within the company. The tasks include analyzing job data, investigating metric spikes, and providing actionable insights to improve the company's operations.

TECH STACK USED IN THIS PROJECT :
MySQL Workbench for writing SQL queries
Microsoft PowerPoint for making the presentation

DATABASE CREATION

The provided SQL script method for creating and importing data into the table in the project3 database is efficient and precise. This method starts by creating the database and table, then loads data from a CSV file in program data->sql->uploads with specific formatting instructions, ensuring accurate data entry. It includes steps to transform and update data, such as converting date strings to datetime format and renaming columns. This approach offers advantages over using an import wizard and is more robust and maintainable solution for managing database imports.

Similarly other tables are created.



```
create database project3;
use project3;
create table job_data(
ds varchar(100),
job_id int,
actor_id int,
event varchar(100),
language varchar(100),
time_spent  int,
org varchar(20));

SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads
INTO TABLE job_data
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
select* from job_data;

ALTER TABLE JOB_DATA ADD COLUMN TEMP_CREATED_AT datetime;

UPDATE JOB_DATA SET TEMP_CREATED_AT = str_to_date(ds, '%m/%d/%Y');
SET SQL_SAFE_UPDATES=0;
ALTER TABLE JOB_DATA DROP COLUMN DS;
ALTER TABLE JOB_DATA CHANGE COLUMN TEMP_CREATED_AT ds DATETIME;
```

APPROACH, QUERIES AND INSIGHTS

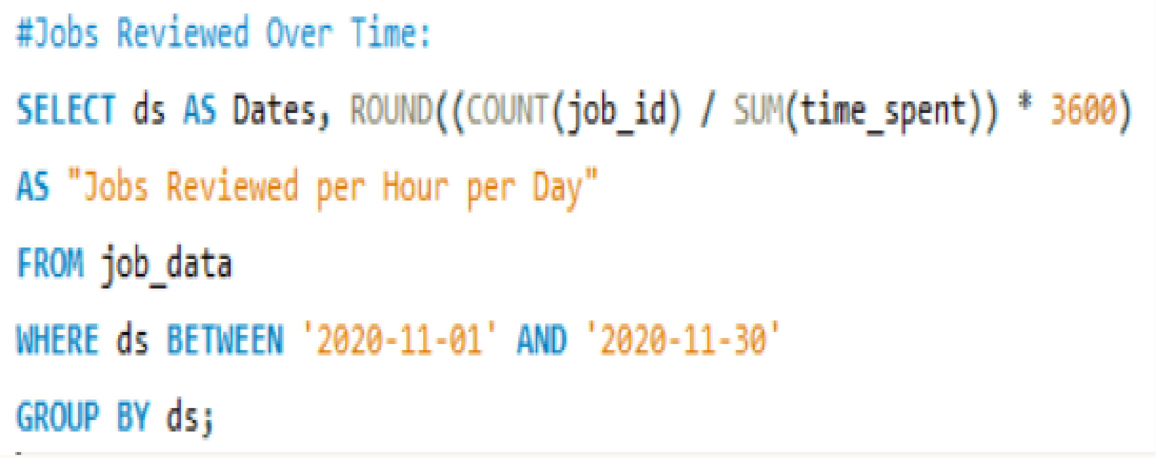
CASE STUDY - 1

TASKS:

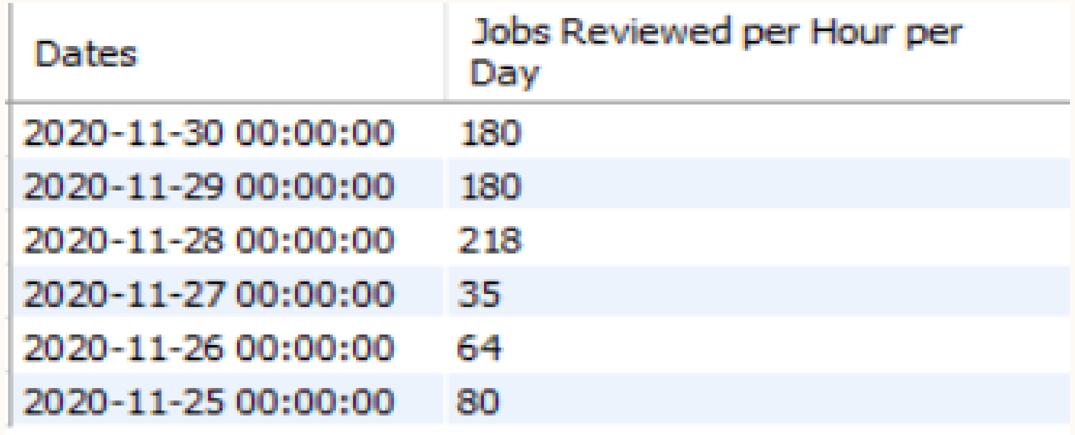
- Jobs Reviewed Over Time
- Throughput Analysis
- Language Share Analysis
- Duplicate Rows Detection

JOBS REVIEWED OVER TIME

SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.



```
#Jobs Reviewed Over Time:

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 '2020-11-01' AND '2020-11-30'

GROUP BY ds;
```



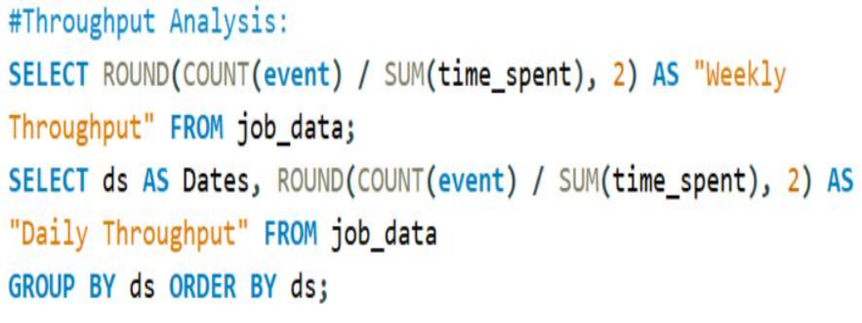
Dates	Jobs Reviewed per Hour per Day
2020-11-30 00:00:00	180
2020-11-29 00:00:00	180
2020-11-28 00:00:00	218
2020-11-27 00:00:00	35
2020-11-26 00:00:00	64
2020-11-25 00:00:00	80

INSIGHTS:

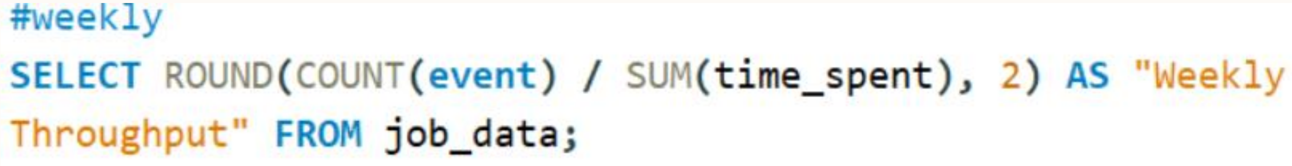
Most jobs are reviewed on November 28, 2020, the number of jobs reviewed that day was 218. This information is useful for optimizing staffing and resources during high-activity periods.

THROUGHPUT ANALYSIS

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

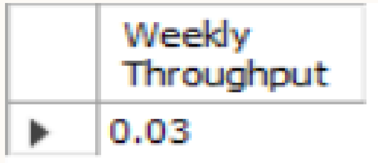


```
#Throughput Analysis:
SELECT ROUND(COUNT(event) / SUM(time_spent), 2) AS "Weekly
Throughput" FROM job_data;
SELECT ds AS Dates, ROUND(COUNT(event) / SUM(time_spent), 2) AS
"Daily Throughput" FROM job_data
GROUP BY ds ORDER BY ds;
```



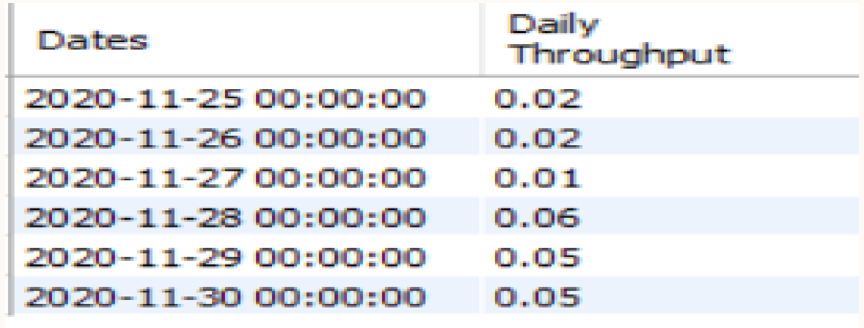
```
#weekly
SELECT ROUND(COUNT(event) / SUM(time_spent), 2) AS "Weekly
Throughput" FROM job_data;
```

Weekly Throughput



	Weekly Throughput
▶	0.03

Daily throughput

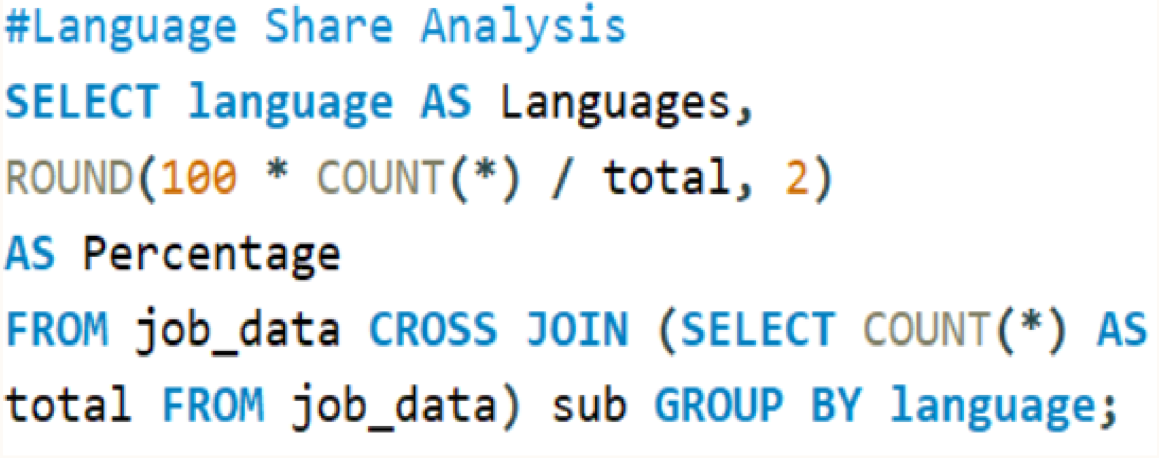


Dates	Daily Throughput
2020-11-25 00:00:00	0.02
2020-11-26 00:00:00	0.02
2020-11-27 00:00:00	0.01
2020-11-28 00:00:00	0.06
2020-11-29 00:00:00	0.05
2020-11-30 00:00:00	0.05

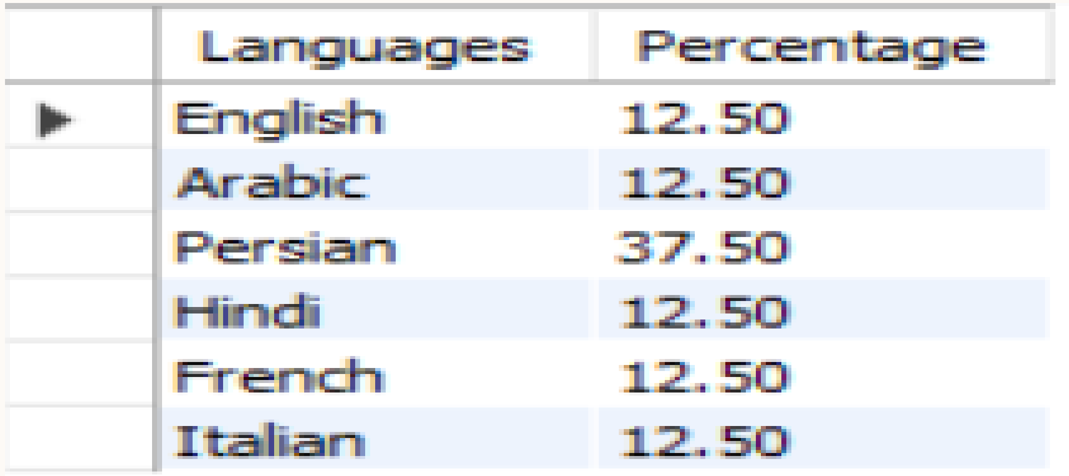
MY PREFERENCE:- I prefer using the 7-day rolling average for throughput over the daily metric because it smooths out short-term fluctuations and provides a clearer view of long-term trends. This approach reduces the noise from daily variations, allowing for better detection of significant patterns and trends in the data. This makes it easier to identify underlying issues or successes and supports more informed decision-making.

LANGUAGE SHARE ANALYSIS

Compute the percentage share of each language in the last 30 days



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



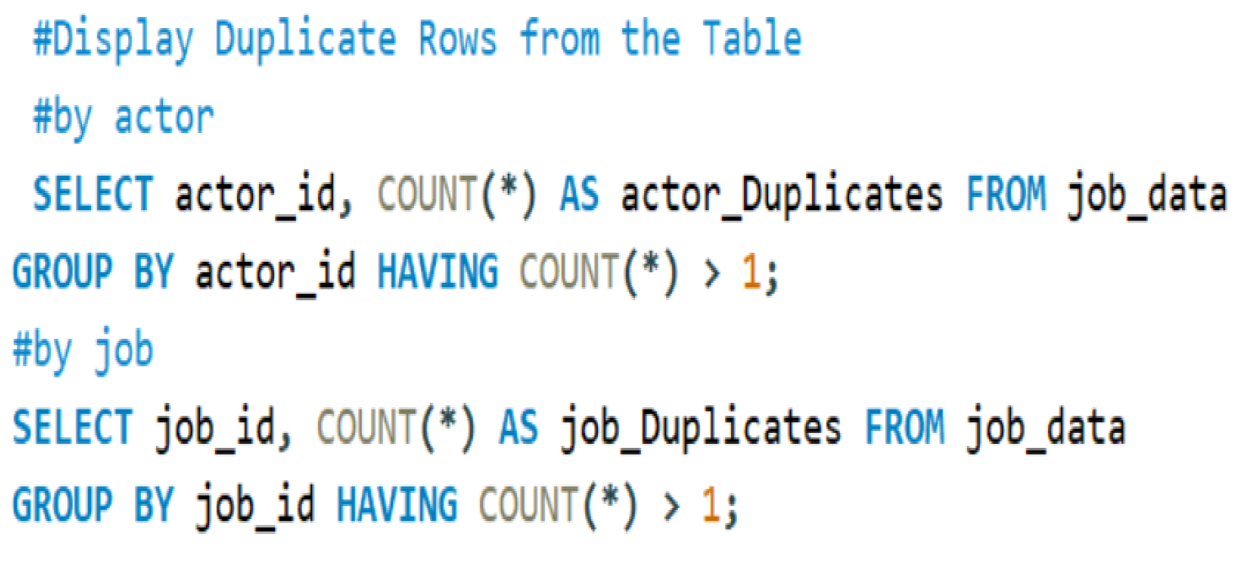
	Languages	Percentage
▶	English	12.50
	Arabic	12.50
	Persian	37.50
	Hindi	12.50
	French	12.50
	Italian	12.50

INSIGHTS:

The Persian language holds the largest proportion, comprising 37.5% of the overall total.

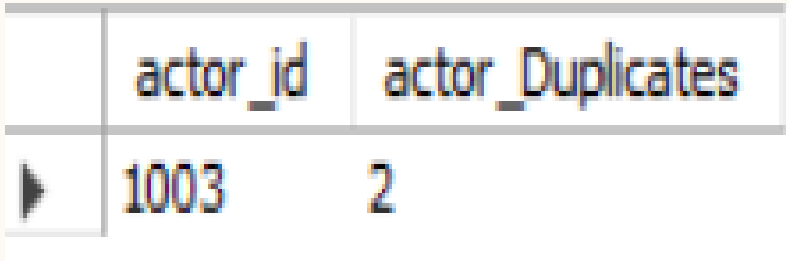
DUPLICATE ROWS DETECTION:

Identified duplicate entries in the job data, underscoring the need for data integrity checks



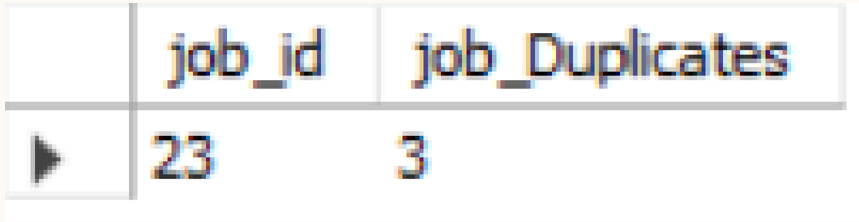
```
 #Display Duplicate Rows from the Table
 #by actor
 SELECT actor_id, COUNT(*) AS actor_Duplicates FROM job_data
GROUP BY actor_id HAVING COUNT(*) > 1;
#by job
SELECT job_id, COUNT(*) AS job_Duplicates FROM job_data
GROUP BY job_id HAVING COUNT(*) > 1;
```

ACTOR DUPLICATES



	actor_id	actor_Duplicates
▶	1003	2

JOB DUPLICATES



	job_id	job_Duplicates
▶	23	3

INSIGHTS:
3 duplicates of job id
2 Duplicates of actor id

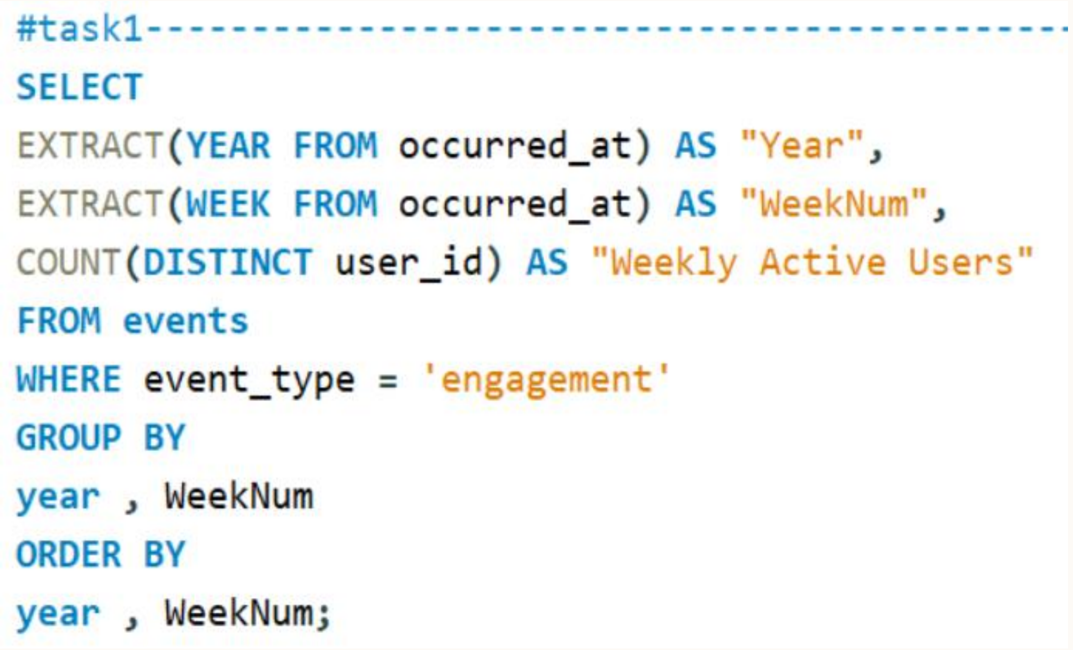
CASE STUDY - 2

TASKS:

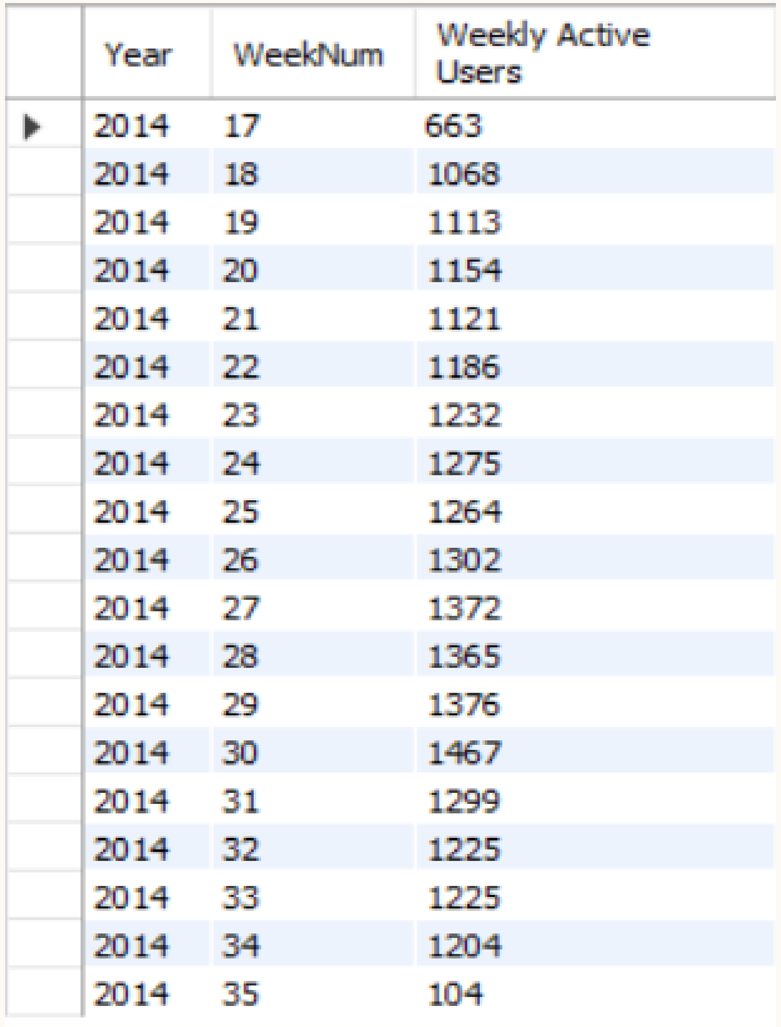
- Weekly User Engagement
- User Growth Analysis
- Weekly Retention Analysis
- Weekly Engagement Per Device
- Email Engagement Analysis

WEEKLY USER ENGAGEMENT

Measured the activeness of users on a weekly basis and SQL query to calculate the weekly user engagement.



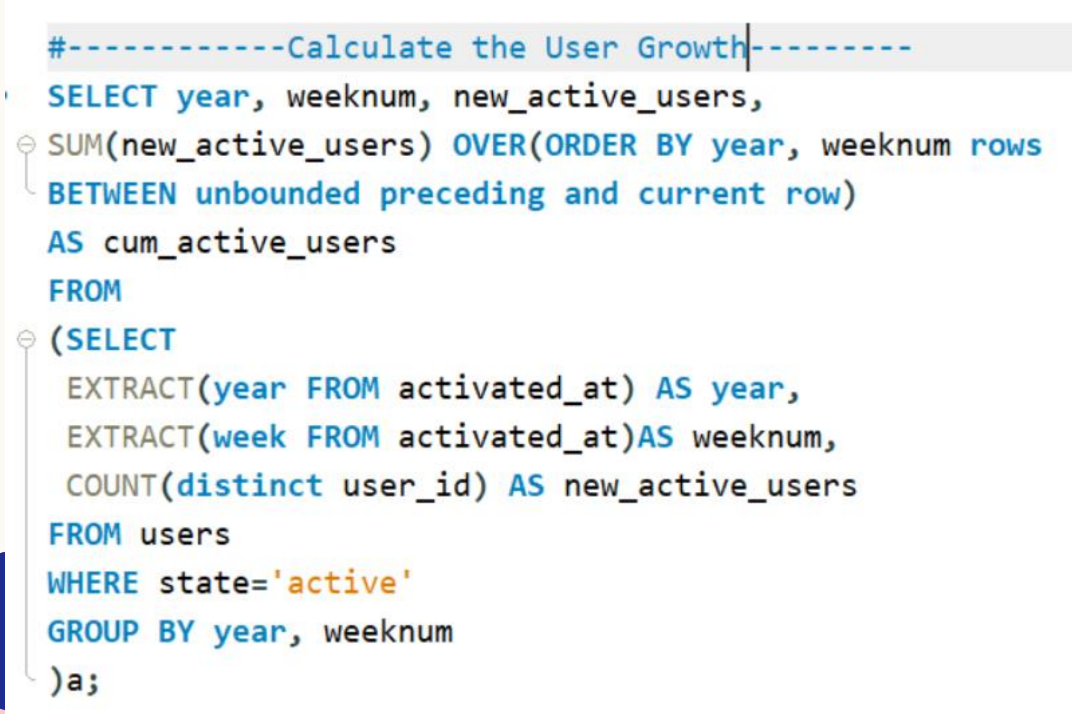
```
#task1----------------------------------------
SELECT
EXTRACT(YEAR FROM occurred_at) AS "Year",
EXTRACT(WEEK FROM occurred_at) AS "WeekNum",
COUNT(DISTINCT user_id) AS "Weekly Active Users"
FROM events
WHERE event_type = 'engagement'
GROUP BY
year , WeekNum
ORDER BY
year , WeekNum;
```



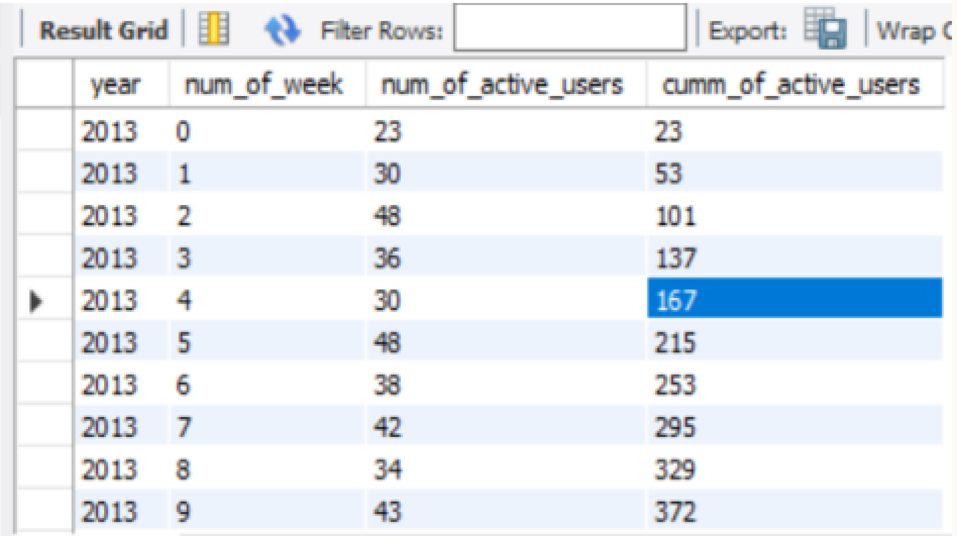
Year	WeekNum	Weekly Active Users
2014	17	663
2014	18	1068
2014	19	1113
2014	20	1154
2014	21	1121
2014	22	1186
2014	23	1232
2014	24	1275
2014	25	1264
2014	26	1302
2014	27	1372
2014	28	1365
2014	29	1376
2014	30	1467
2014	31	1299
2014	32	1225
2014	33	1225
2014	34	1204
2014	35	104

USER GROWTH ANALYSIS

Analyzed the growth of users over time for a product. SQL query to calculate the user growth for the product.



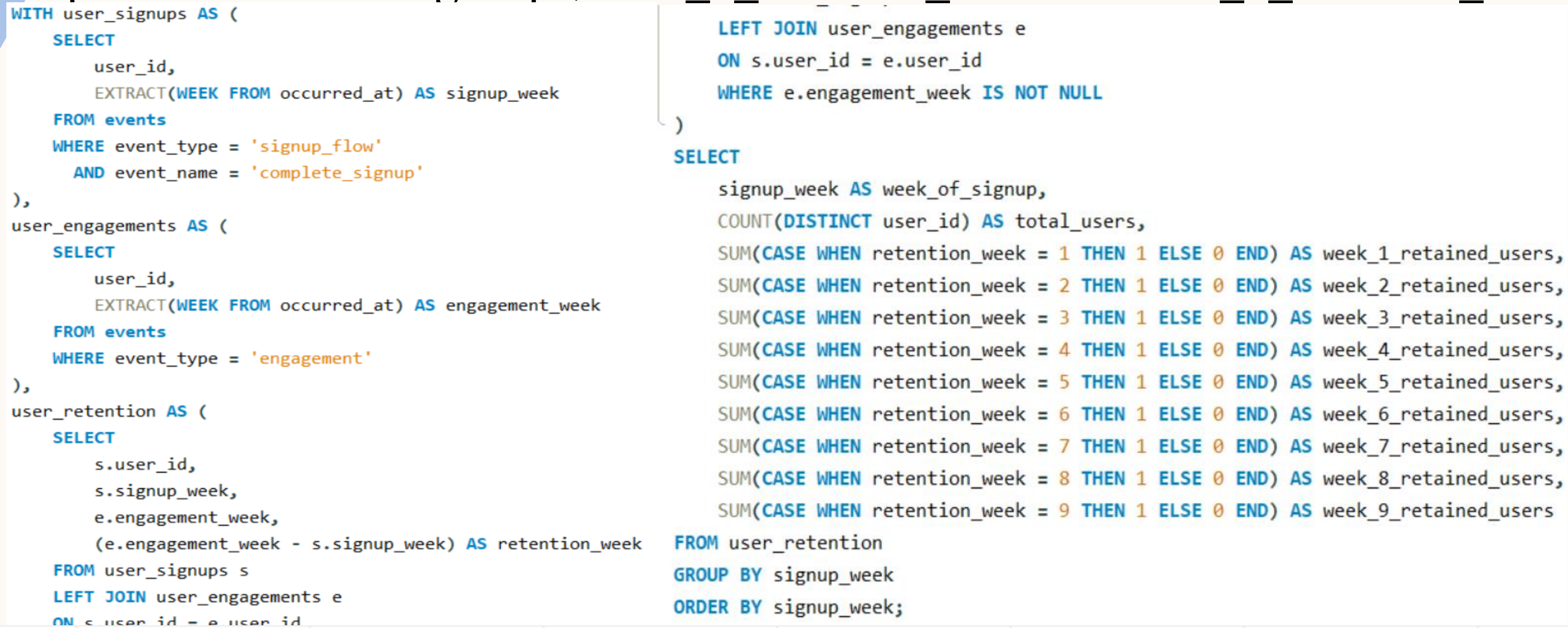
```
#-----------Calculate the User Growth---------
SELECT year, weeknum, new_active_users,
SUM(new_active_users) OVER(ORDER BY year, weeknum rows
BETWEEN unbounded preceding and current row)
AS cum_active_users
FROM
(SELECT
 EXTRACT(year FROM activated_at) AS year,
 EXTRACT(week FROM activated_at)AS weeknum,
 COUNT(distinct user_id) AS new_active_users
FROM users
WHERE state='active'
GROUP BY year, weeknum
)a;
```



year	num_of_week	num_of_active_users	cumm_of_active_users
2013	0	23	23
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

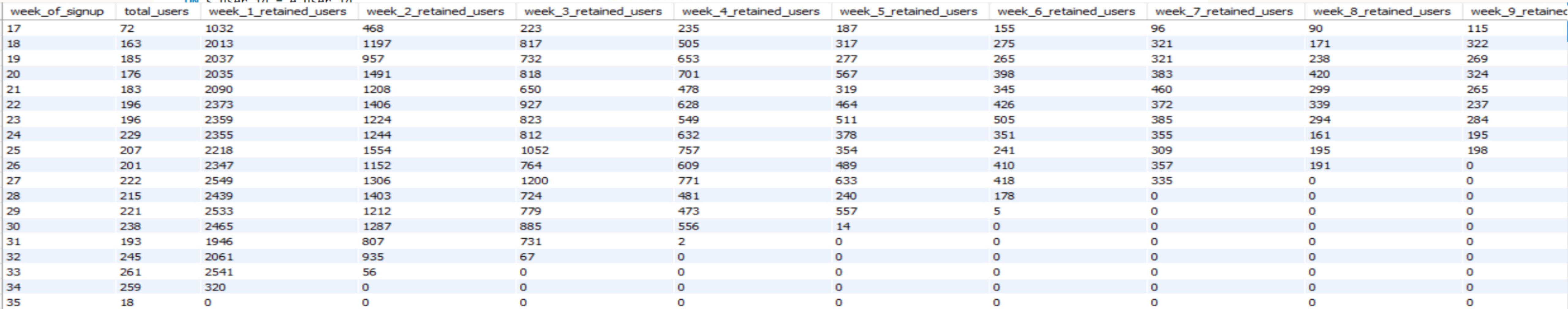
WEEKLY RETENTION ANALYSIS

The query Groups the retention data by the week of sign-up (signup_week) and Counts the total number of users who signed up in each week and Calculates the number of users retained in each subsequent week after sign-up (week_1_retained_users to week_9_retained_users).



```sql
WITH user_signups AS (
    SELECT
        user_id,
        EXTRACT(WEEK FROM occurred_at) AS signup_week
    FROM events
    WHERE event_type = 'signup_flow'
      AND event_name = 'complete_signup'
),
user_engagements AS (
    SELECT
        user_id,
        EXTRACT(WEEK FROM occurred_at) AS engagement_week
    FROM events
    WHERE event_type = 'engagement'
),
user_retention AS (
    SELECT
        s.user_id,
        s.signup_week,
        e.engagement_week,
        (e.engagement_week - s.signup_week) AS retention_week
    FROM user_signups s
    LEFT JOIN user_engagements e
    ON s.user_id = e.user_id
```

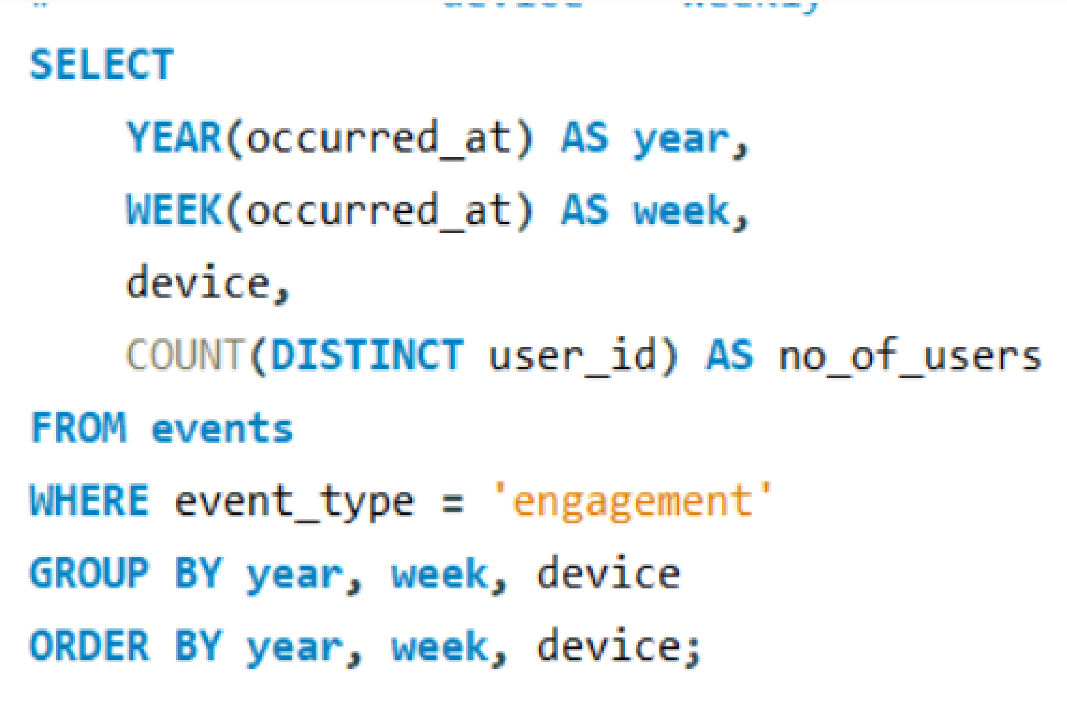
```sql
    LEFT JOIN user_engagements e
    ON s.user_id = e.user_id
    WHERE e.engagement_week IS NOT NULL
)
SELECT
    signup_week AS week_of_signup,
    COUNT(DISTINCT user_id) AS total_users,
    SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS week_1_retained_users,
    SUM(CASE WHEN retention_week = 2 THEN 1 ELSE 0 END) AS week_2_retained_users,
    SUM(CASE WHEN retention_week = 3 THEN 1 ELSE 0 END) AS week_3_retained_users,
    SUM(CASE WHEN retention_week = 4 THEN 1 ELSE 0 END) AS week_4_retained_users,
    SUM(CASE WHEN retention_week = 5 THEN 1 ELSE 0 END) AS week_5_retained_users,
    SUM(CASE WHEN retention_week = 6 THEN 1 ELSE 0 END) AS week_6_retained_users,
    SUM(CASE WHEN retention_week = 7 THEN 1 ELSE 0 END) AS week_7_retained_users,
    SUM(CASE WHEN retention_week = 8 THEN 1 ELSE 0 END) AS week_8_retained_users,
    SUM(CASE WHEN retention_week = 9 THEN 1 ELSE 0 END) AS week_9_retained_users
FROM user_retention
GROUP BY signup_week
ORDER BY signup_week;
```



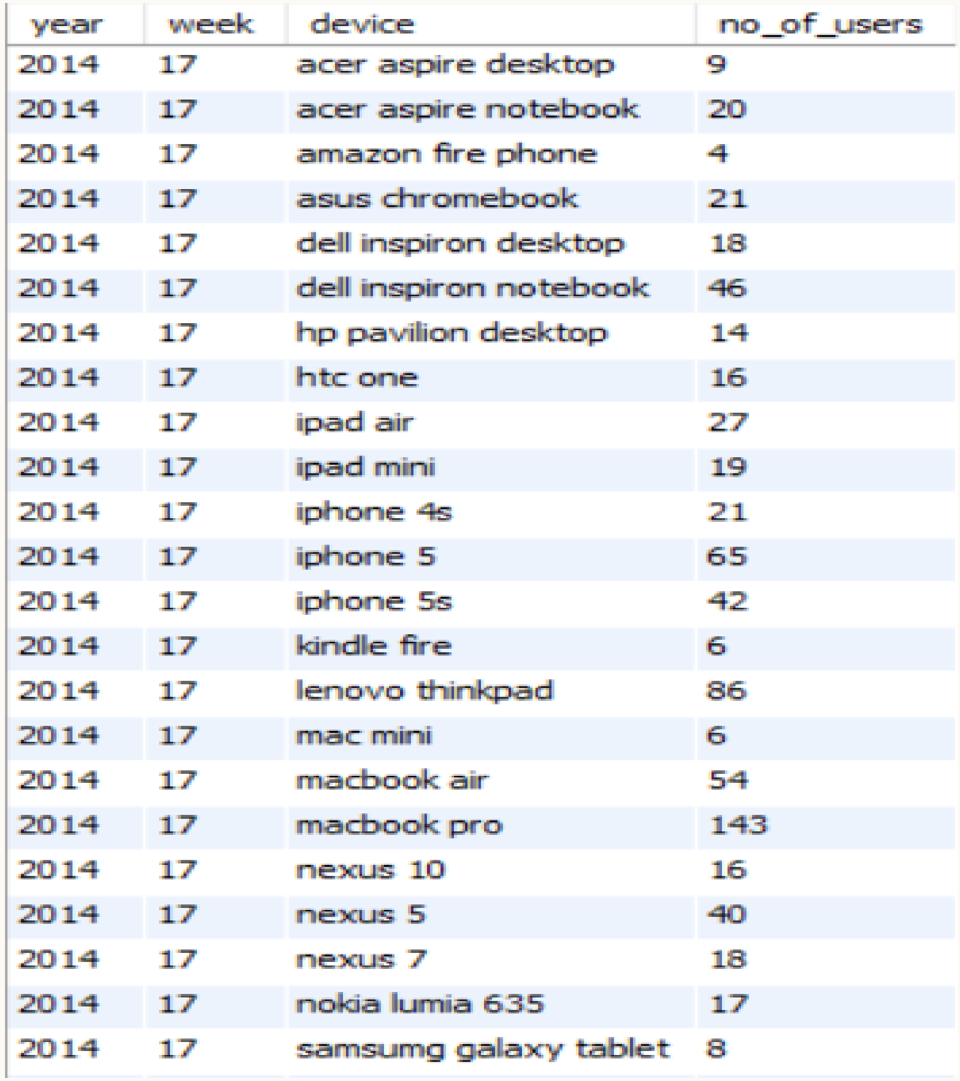
week_of_signup	total_users	week_1_retained_users	week_2_retained_users	week_3_retained_users	week_4_retained_users	week_5_retained_users	week_6_retained_users	week_7_retained_users	week_8_retained_users	week_9_retained
17	72	1032	468	223	235	187	155	96	90	115
18	163	2013	1197	817	505	317	275	321	171	322
19	185	2037	957	732	653	277	265	321	238	269
20	176	2035	1491	818	701	567	398	383	420	324
21	183	2090	1208	650	478	319	345	460	299	265
22	196	2373	1406	927	628	464	426	372	339	237
23	196	2359	1224	823	549	511	505	385	294	284
24	229	2355	1244	812	632	378	351	355	161	195
25	207	2218	1554	1052	757	354	241	309	195	198
26	201	2347	1152	764	609	489	410	357	191	0
27	222	2549	1306	1200	771	633	418	335	0	0
28	215	2439	1403	724	481	240	178	0	0	0
29	221	2533	1212	779	473	557	5	0	0	0
30	238	2465	1287	885	556	14	0	0	0	0
31	193	1946	807	731	2	0	0	0	0	0
32	245	2061	935	67	0	0	0	0	0	0
33	261	2541	56	0	0	0	0	0	0	0
34	259	320	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0

WEEKLY ENGAGEMENT PER DEVICE

The activeness of users on a weekly basis per device and SQL query to calculate the weekly engagement per device.



```
SELECT
    YEAR(occurred_at) AS year,
    WEEK(occurred_at) AS week,
    device,
    COUNT(DISTINCT user_id) AS no_of_users
FROM events
WHERE event_type = 'engagement'
GROUP BY year, week, device
ORDER BY year, week, device;
```

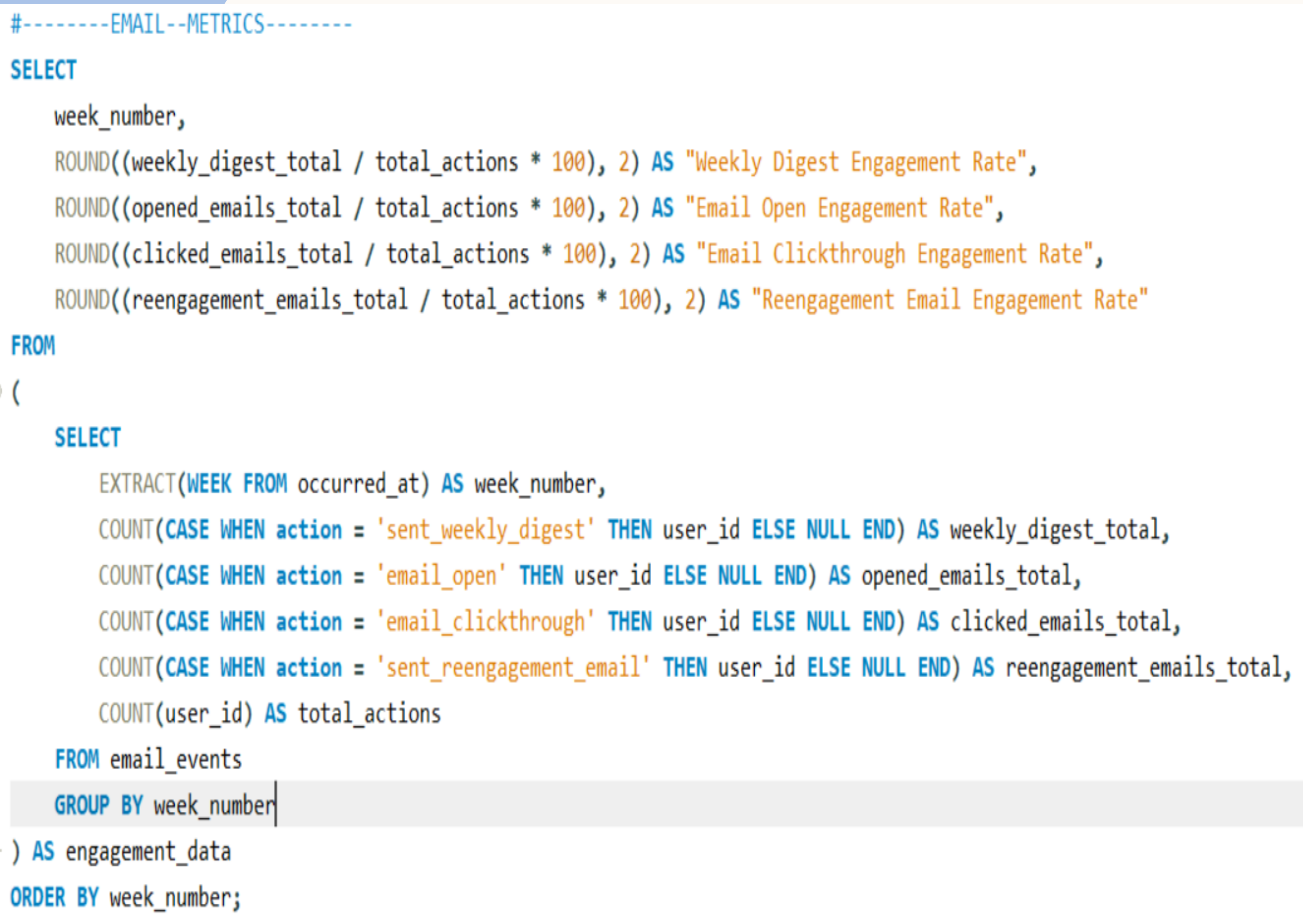


year	week	device	no_of_users
2014	17	acer aspire desktop	9
2014	17	acer aspire notebook	20
2014	17	amazon fire phone	4
2014	17	asus chromebook	21
2014	17	dell inspiron desktop	18
2014	17	dell inspiron notebook	46
2014	17	hp pavilion desktop	14
2014	17	htc one	16
2014	17	ipad air	27
2014	17	ipad mini	19
2014	17	iphone 4s	21
2014	17	iphone 5	65
2014	17	iphone 5s	42
2014	17	kindle fire	6
2014	17	lenovo thinkpad	86
2014	17	mac mini	6
2014	17	macbook air	54
2014	17	macbook pro	143
2014	17	nexus 10	16
2014	17	nexus 5	40
2014	17	nexus 7	18
2014	17	nokia lumia 635	17
2014	17	samsumg galaxy tablet	8

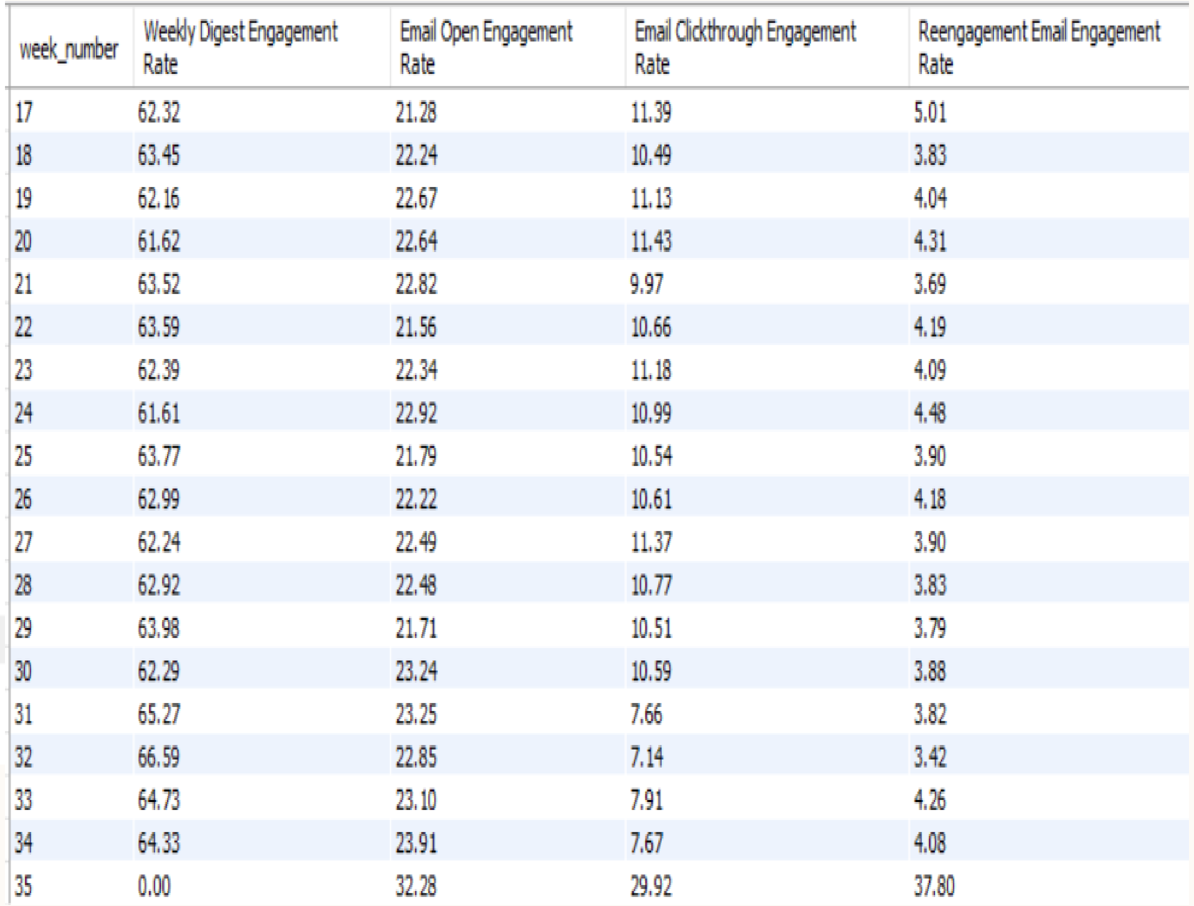
And so on…

EMAIL ENGAGEMENT ANALYSIS

Analyze how users are engaging with the email service and SQL query to calculate the email engagement metrics.



```
#--------EMAIL--METRICS--------
SELECT
    week_number,
    ROUND((weekly_digest_total / total_actions * 100), 2) AS "Weekly Digest Engagement Rate",
    ROUND((opened_emails_total / total_actions * 100), 2) AS "Email Open Engagement Rate",
    ROUND((clicked_emails_total / total_actions * 100), 2) AS "Email Clickthrough Engagement Rate",
    ROUND((reengagement_emails_total / total_actions * 100), 2) AS "Reengagement Email Engagement Rate"
FROM
(
    SELECT
        EXTRACT(WEEK FROM occurred_at) AS week_number,
        COUNT(CASE WHEN action = 'sent_weekly_digest' THEN user_id ELSE NULL END) AS weekly_digest_total,
        COUNT(CASE WHEN action = 'email_open' THEN user_id ELSE NULL END) AS opened_emails_total,
        COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id ELSE NULL END) AS clicked_emails_total,
        COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id ELSE NULL END) AS reengagement_emails_total,
        COUNT(user_id) AS total_actions
    FROM email_events
    GROUP BY week_number
) AS engagement_data
ORDER BY week_number;
```



week_number	Weekly Digest Engagement Rate	Email Open Engagement Rate	Email Clickthrough Engagement Rate	Reengagement Email Engagement 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

Through this project, I successfully analyzed various aspects of the company's operations and user engagement, providing actionable insights that can drive operational improvements and strategic decisions. The analysis enhanced my understanding of data trends and user behavior, enabling me to contribute effectively to decision-making processes within the company. The project demonstrated the importance of thorough data analysis in identifying opportunities for optimization and growth.

THANK YOU

Samiya Alam
samiyaalam1710@gmail.com