Operation Analytics and Investigating Metric Spike

By:- Vishal Singh Sangral

Project Description

The project involves conducting Operational Analytics at a company, akin to Microsoft, focusing on end-to-end operations analysis.

As a Lead Data Analyst, the goal is to collaboratively work with diverse teams like operations, support, and marketing.

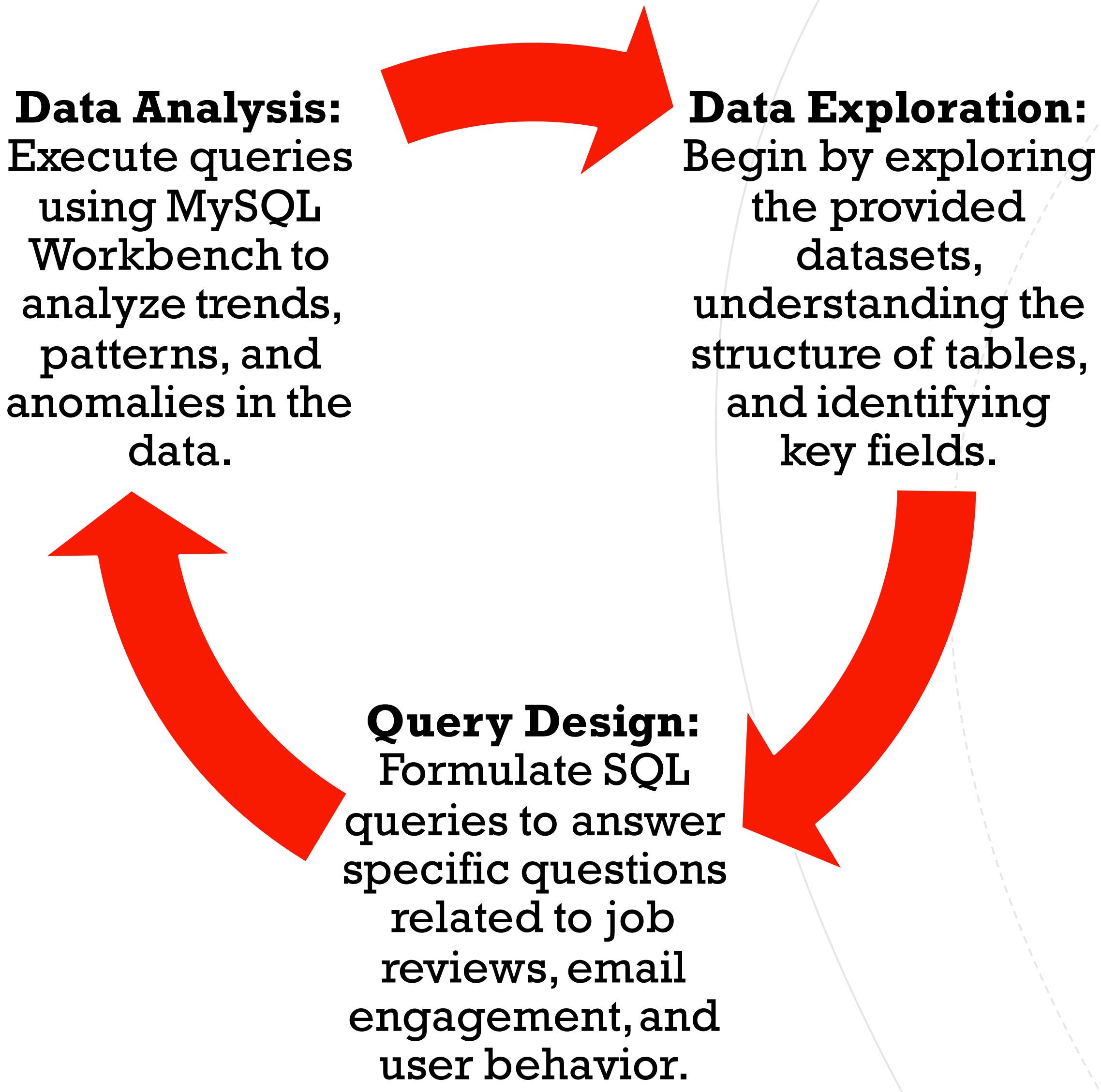
The primary purpose is to identify areas for improvement by analyzing provided datasets.

A key aspect involves investigating metric spikes to understand and explain sudden changes in key metrics.

The analysis aims to provide valuable insights to different departments, aiding in daily decision-making and contributing to the overall enhancement of the company's operations.

The approach involves leveraging advanced SQL skills to derive meaningful conclusions and recommendations from the data.

Approach



Data Exploration: Begin by exploring the provided datasets, understanding the structure of tables, and identifying key fields.

Query Design: Formulate SQL queries to answer specific questions related to job reviews, email engagement, and user behavior.

Data Analysis: Execute queries using MySQL Workbench to analyze trends, patterns, and anomalies in the data.

Excel

MySQL® workbench

Tech-Stack Used

INSIGHTS

Job Data Analysis:

Analyzed job reviews over time and identified patterns in throughput.

Investigated language share and detected duplicate rows in the dataset.

Metric Spike Investigation:

Explored sudden changes in job metrics, providing valuable insights into potential issues or improvements needed.

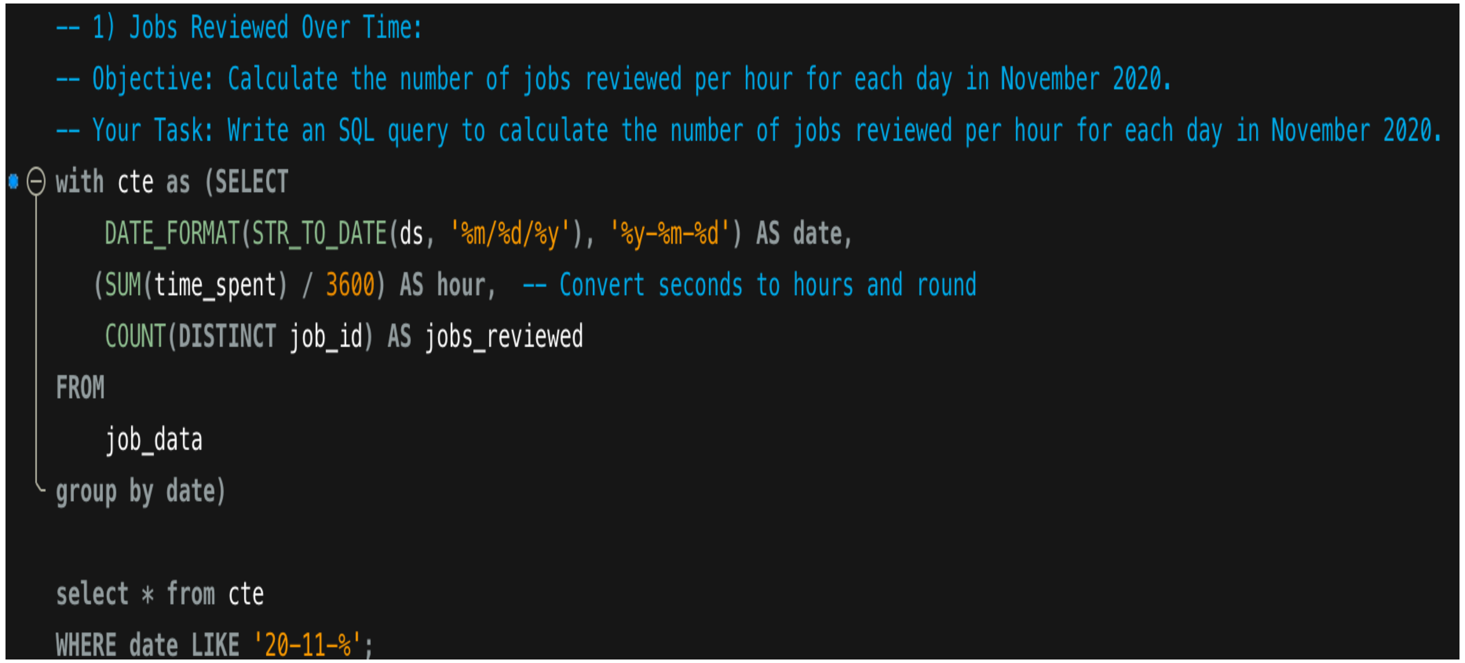
Email Engagement Analysis:

Evaluated user engagement metrics, including opens, clicks, and unsubscribe rates.

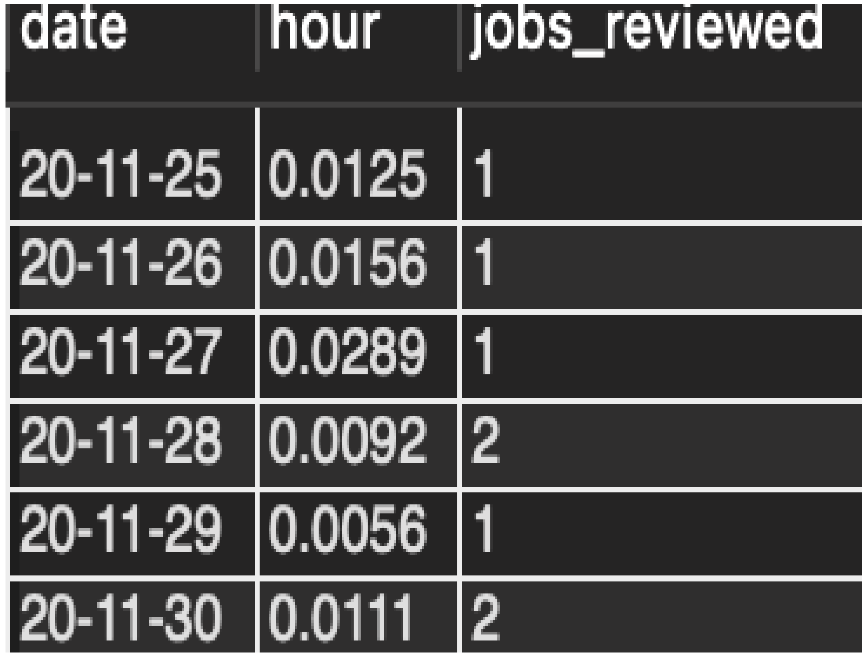
Provided a breakdown of engagement metrics based on specific email actions.

Results

Case study 1: Job Data Analysis



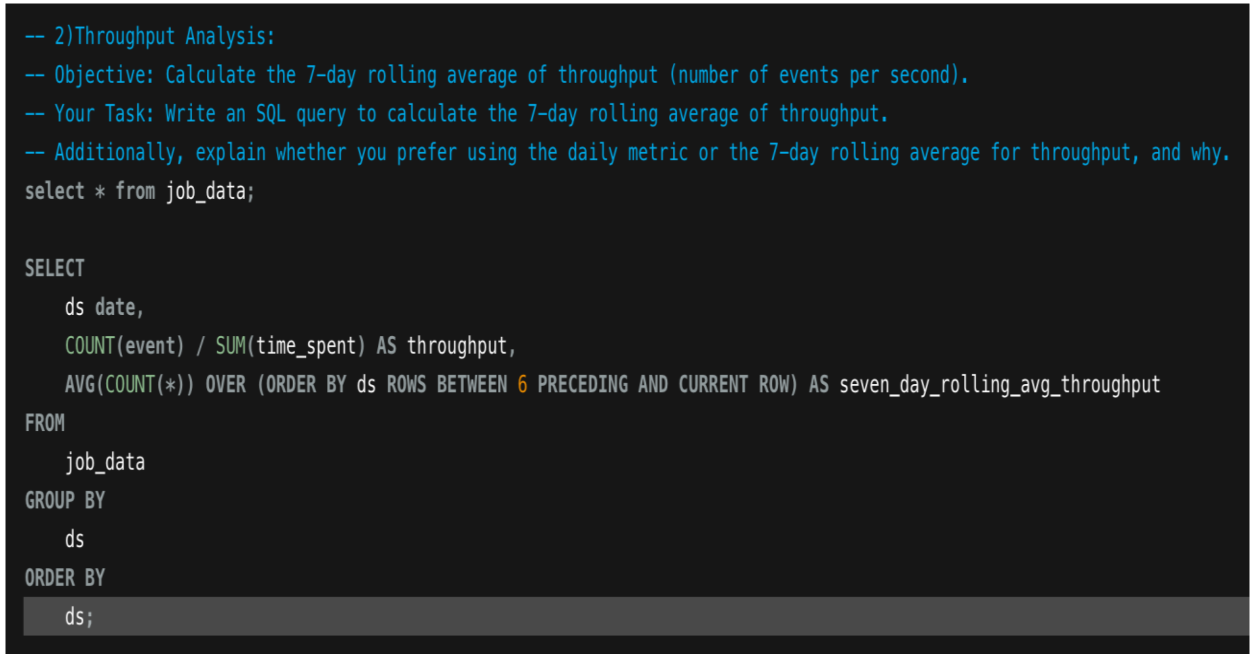
```
-- 1) Jobs Reviewed Over Time:
-- Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.
-- Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.
with cte as (SELECT
    DATE_FORMAT(STR_TO_DATE(ds, '%m/%d/%y'), '%y-%m-%d') AS date,
   (SUM(time_spent) / 3600) AS hour,  -- Convert seconds to hours and round
    COUNT(DISTINCT job_id) AS jobs_reviewed
FROM
    job_data
group by date)

select * from cte
WHERE date LIKE '20-11-%';
```



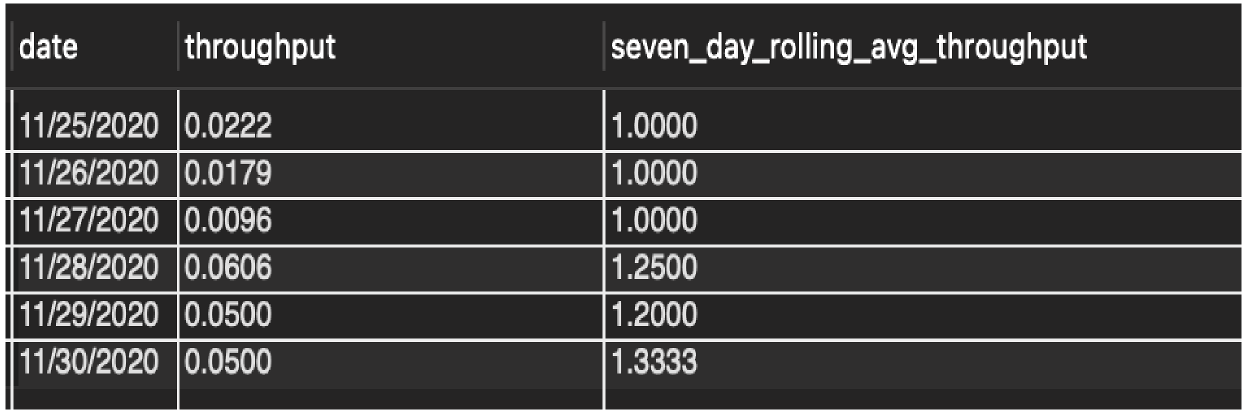
date	hour	jobs_reviewed
20-11-25	0.0125	1
20-11-26	0.0156	1
20-11-27	0.0289	1
20-11-28	0.0092	2
20-11-29	0.0056	1
20-11-30	0.0111	2

Jobs Reviewed Over Time

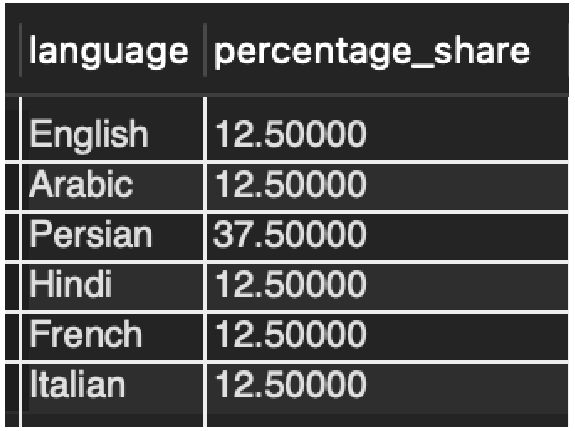
Throughput Analysis



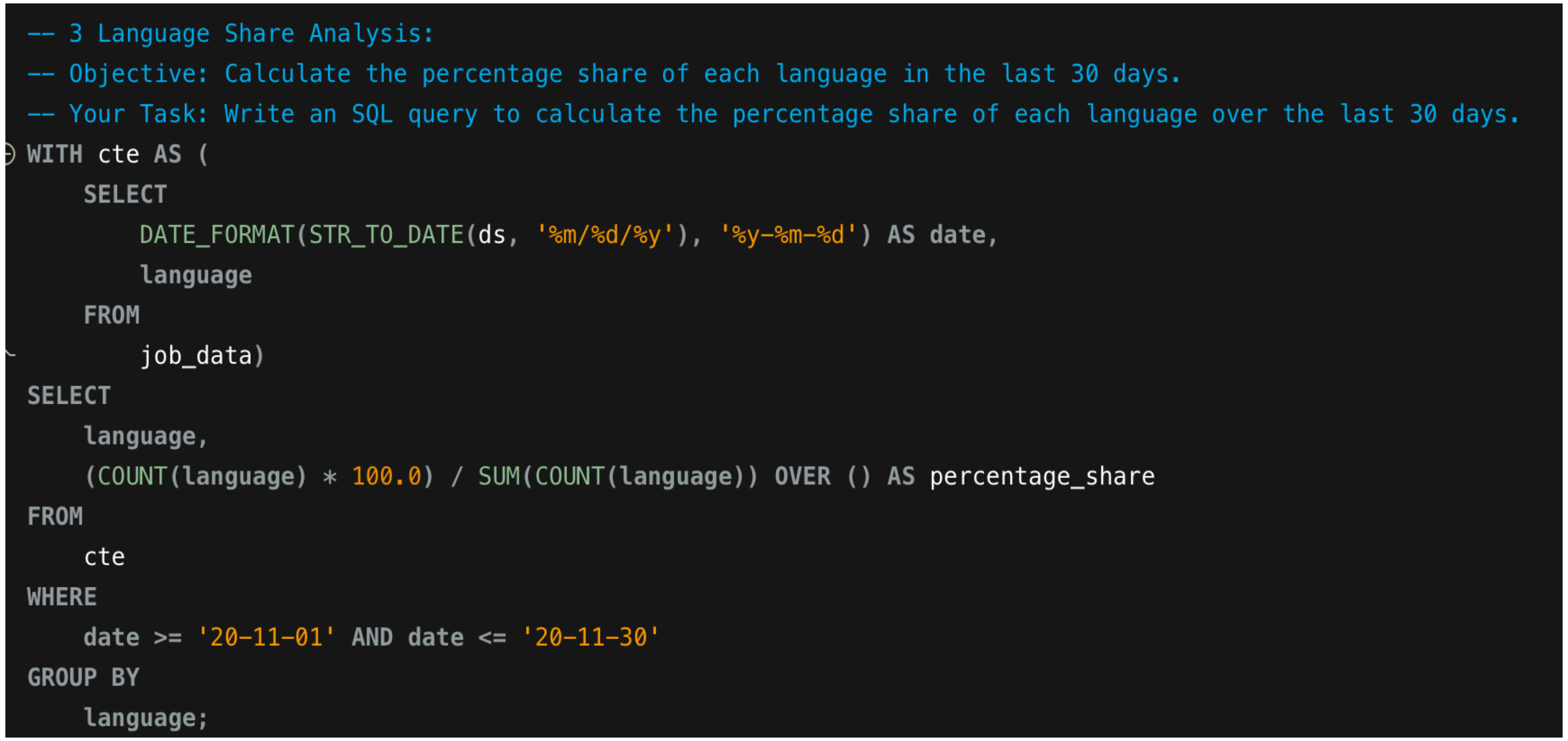
```
-- 2)Throughput Analysis:
-- Objective: Calculate the 7-day rolling average of throughput (number of events per second).
-- Your Task: Write an SQL query to calculate the 7-day rolling average of throughput.
-- Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.
select * from job_data;

SELECT
    ds date,
    COUNT(event) / SUM(time_spent) AS throughput,
    AVG(COUNT(*)) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS seven_day_rolling_avg_throughput
FROM
    job_data
GROUP BY
    ds
ORDER BY
    ds;
```



date	throughput	seven_day_rolling_avg_throughput
11/25/2020	0.0222	1.0000
11/26/2020	0.0179	1.0000
11/27/2020	0.0096	1.0000
11/28/2020	0.0606	1.2500
11/29/2020	0.0500	1.2000
11/30/2020	0.0500	1.3333

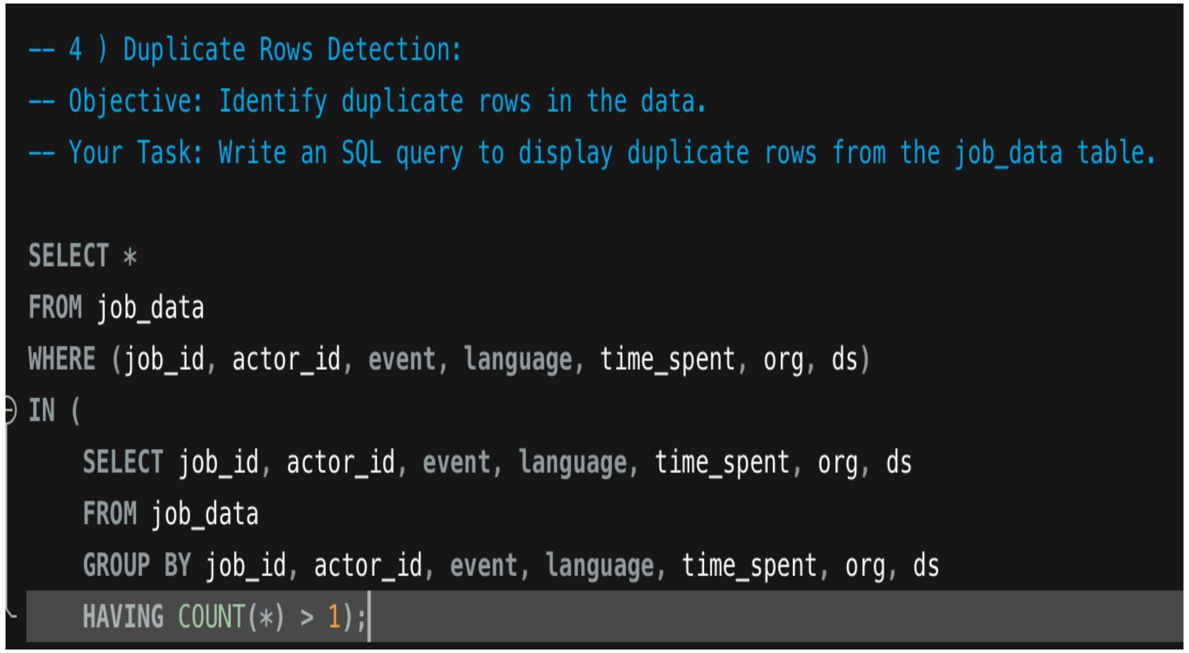


language	percentage_share
English	12.50000
Arabic	12.50000
Persian	37.50000
Hindi	12.50000
French	12.50000
Italian	12.50000

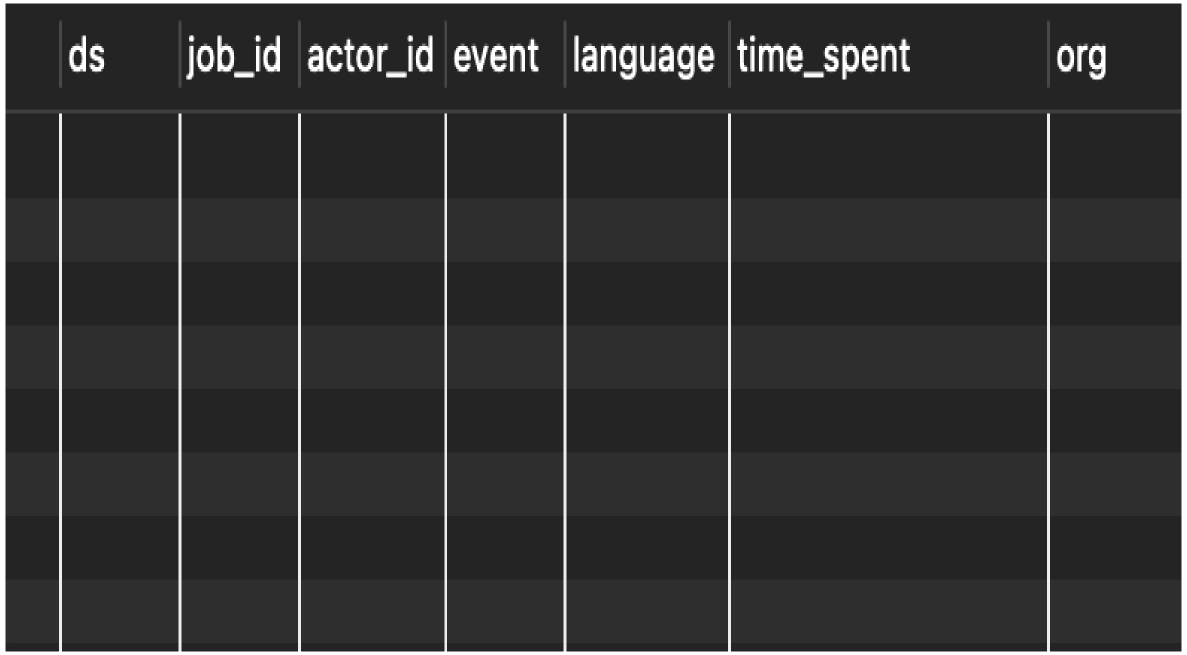


```
-- 3 Language Share Analysis:
-- Objective: Calculate the percentage share of each language in the last 30 days.
-- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.
WITH cte AS (
    SELECT
        DATE_FORMAT(STR_TO_DATE(ds, '%m/%d/%y'), '%y-%m-%d') AS date,
        language
    FROM
        job_data)
SELECT
    language,
    (COUNT(language) * 100.0) / SUM(COUNT(language)) OVER () AS percentage_share
FROM
    cte
WHERE
    date >= '20-11-01' AND date <= '20-11-30'
GROUP BY
    language;
```

Language Share Analysis



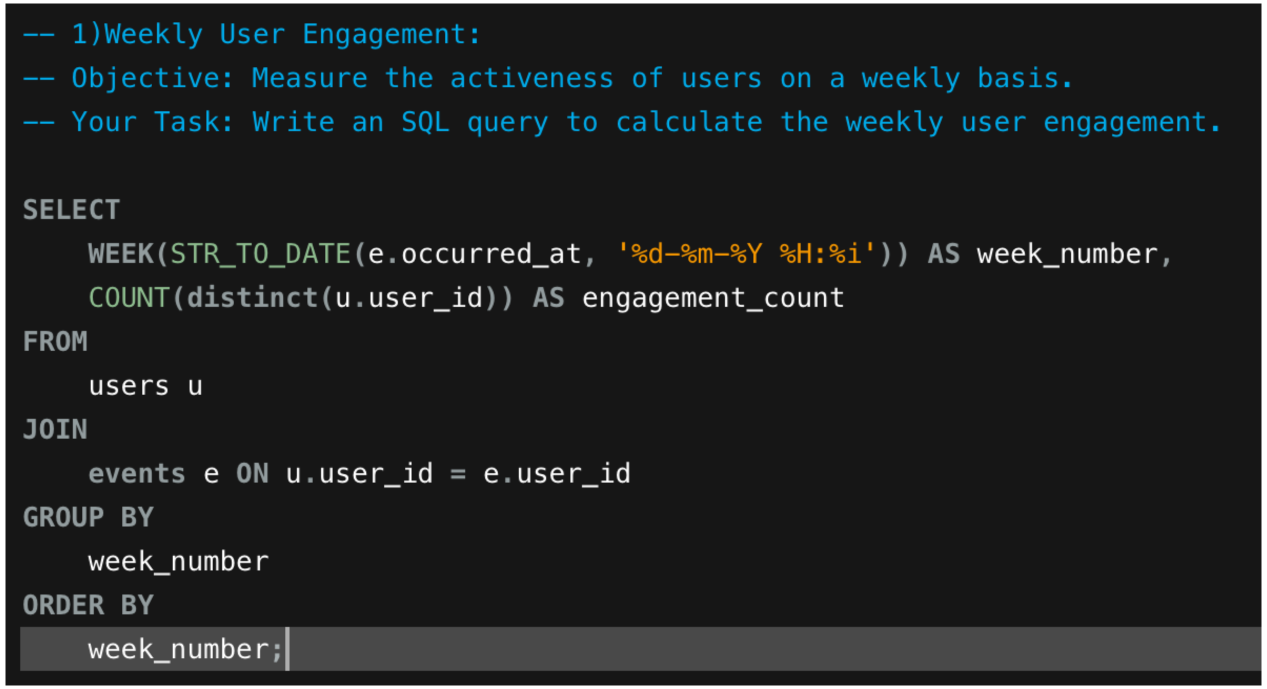
```
-- 4 ) Duplicate Rows Detection:
-- Objective: Identify duplicate rows in the data.
-- Your Task: Write an SQL query to display duplicate rows from the job_data table.

SELECT *
FROM job_data
WHERE (job_id, actor_id, event, language, time_spent, org, ds)
IN (
    SELECT job_id, actor_id, event, language, time_spent, org, ds
    FROM job_data
    GROUP BY job_id, actor_id, event, language, time_spent, org, ds
    HAVING COUNT(*) > 1);
```



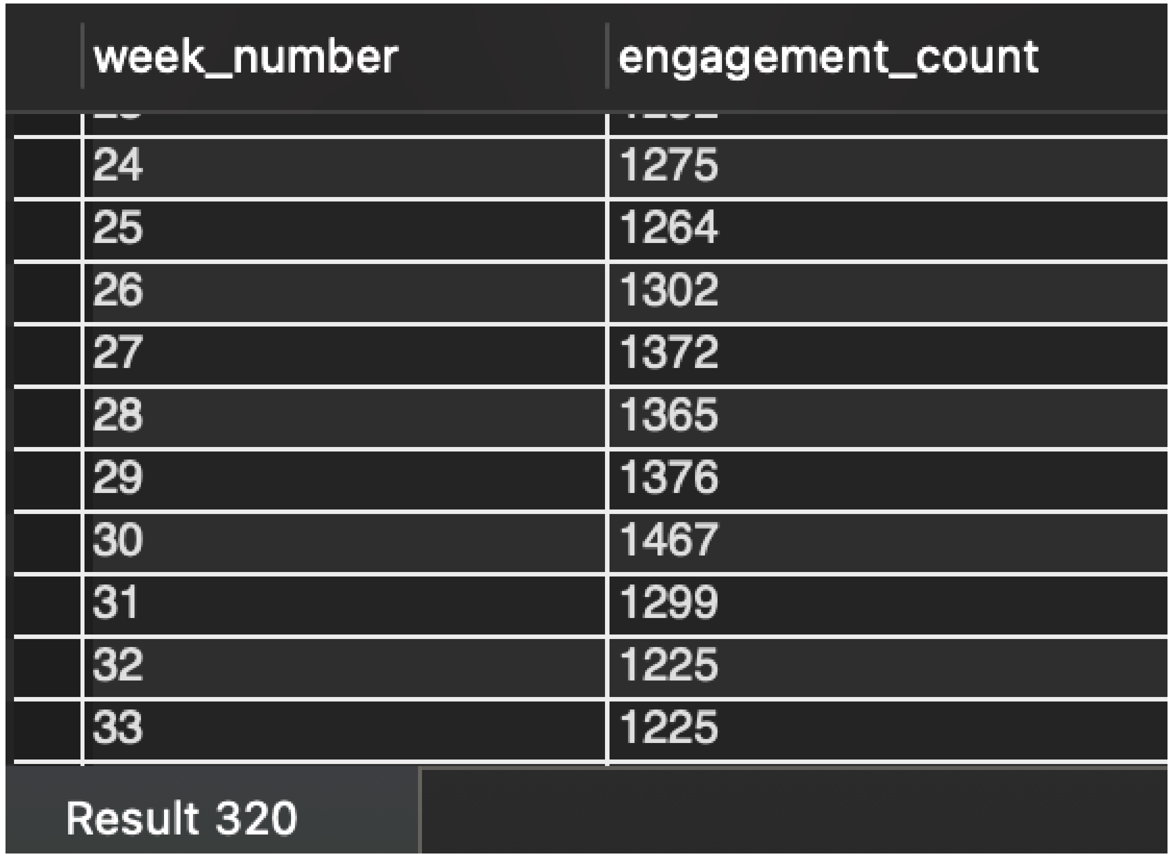
ds	job_id	actor_id	event	language	time_spent	org

Duplicate Rows Detection

Case study 2: Investigating Metric Spike



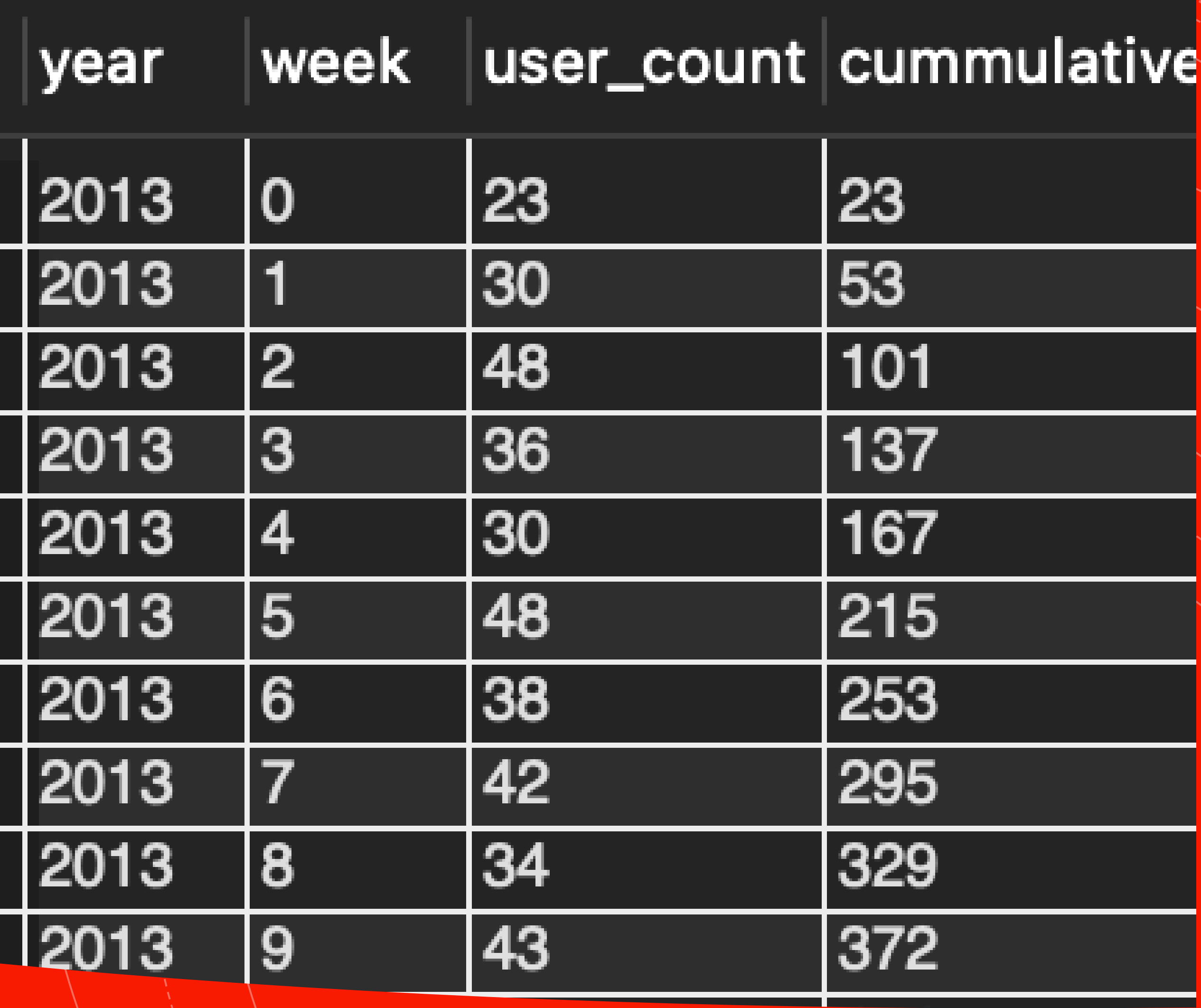
```sql
-- 1)Weekly User Engagement:
-- Objective: Measure the activeness of users on a weekly basis.
-- Your Task: Write an SQL query to calculate the weekly user engagement.

SELECT
    WEEK(STR_TO_DATE(e.occurred_at, '%d-%m-%Y %H:%i')) AS week_number,
    COUNT(distinct(u.user_id)) AS engagement_count
FROM
    users u
JOIN
    events e ON u.user_id = e.user_id
GROUP BY
    week_number
ORDER BY
    week_number;
```



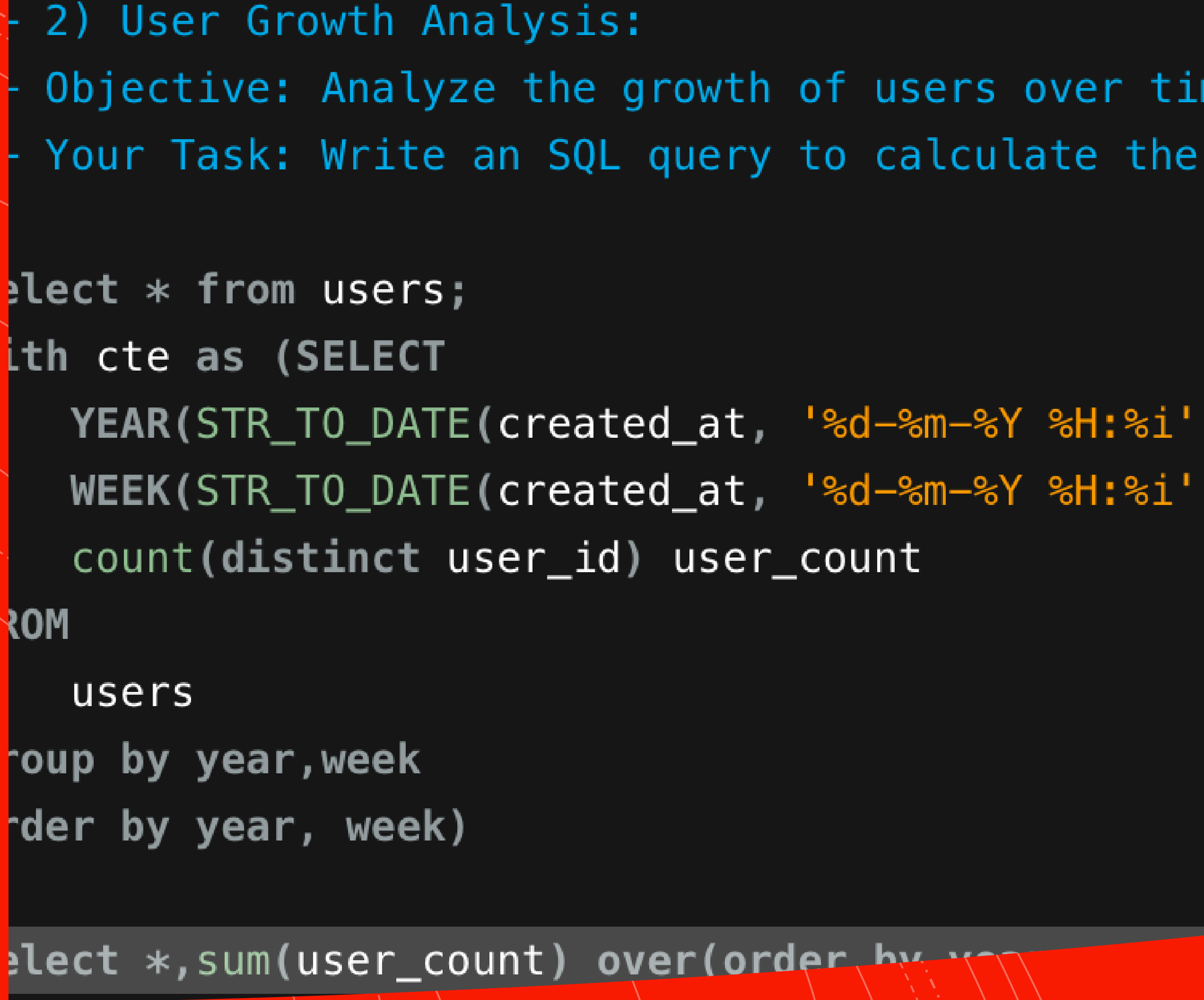
week_number	engagement_count
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225

Result 320

Weekly User Engagement



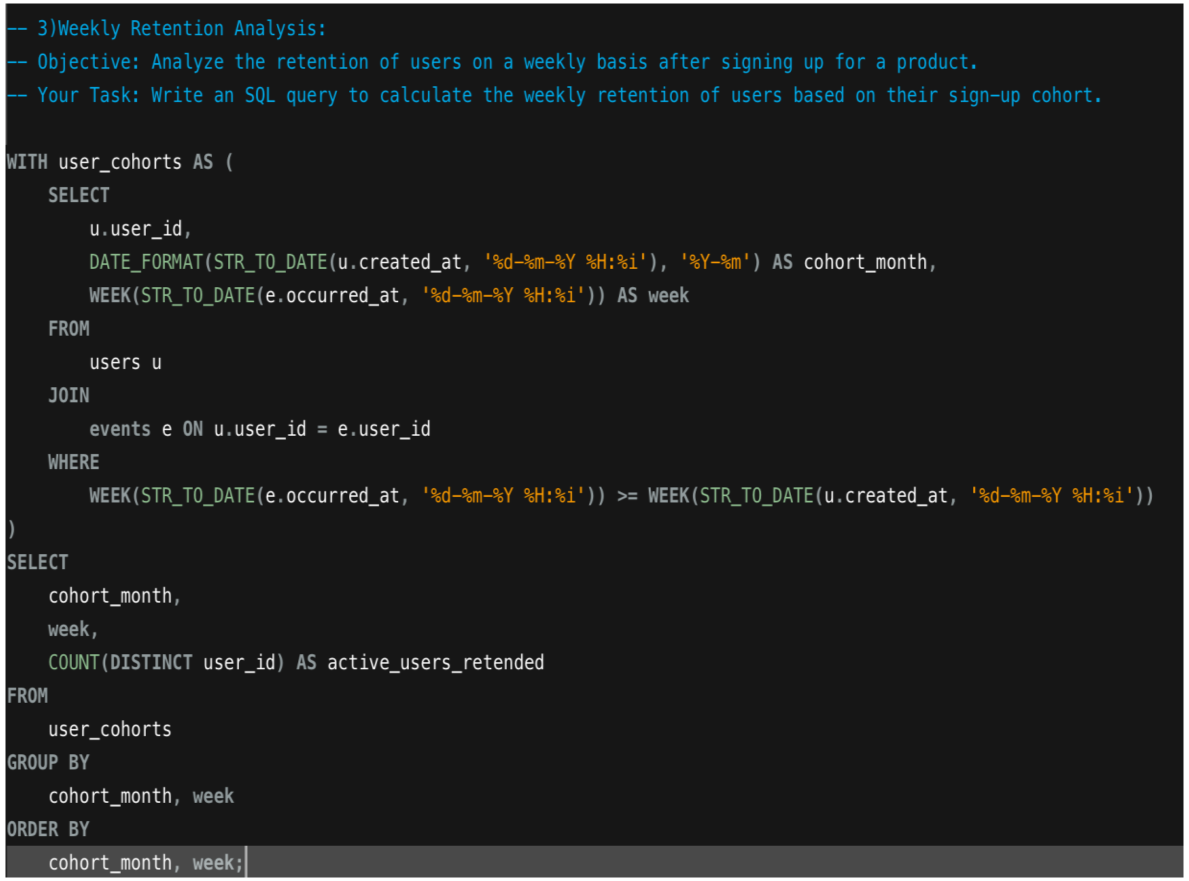
year	week	user_count	cummulative
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372



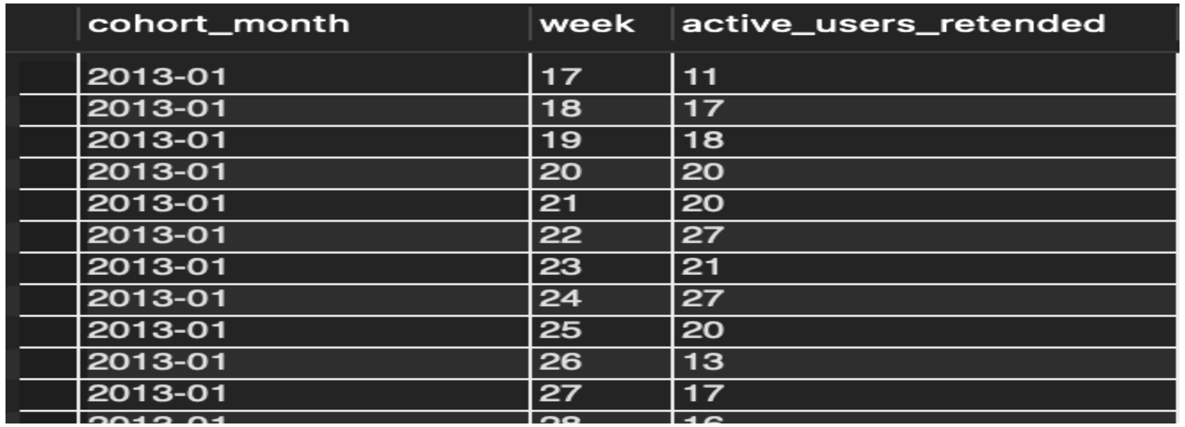
```
- 2) User Growth Analysis:
- Objective: Analyze the growth of users over ti
- Your Task: Write an SQL query to calculate the

elect * from users;
ith cte as (SELECT
   YEAR(STR_TO_DATE(created_at, '%d-%m-%Y %H:%i'
   WEEK(STR_TO_DATE(created_at, '%d-%m-%Y %H:%i'
   count(distinct user_id) user_count
ROM
   users
roup by year,week
rder by year, week)

elect *,sum(user_count) over(order by
```

User Growth Analysis

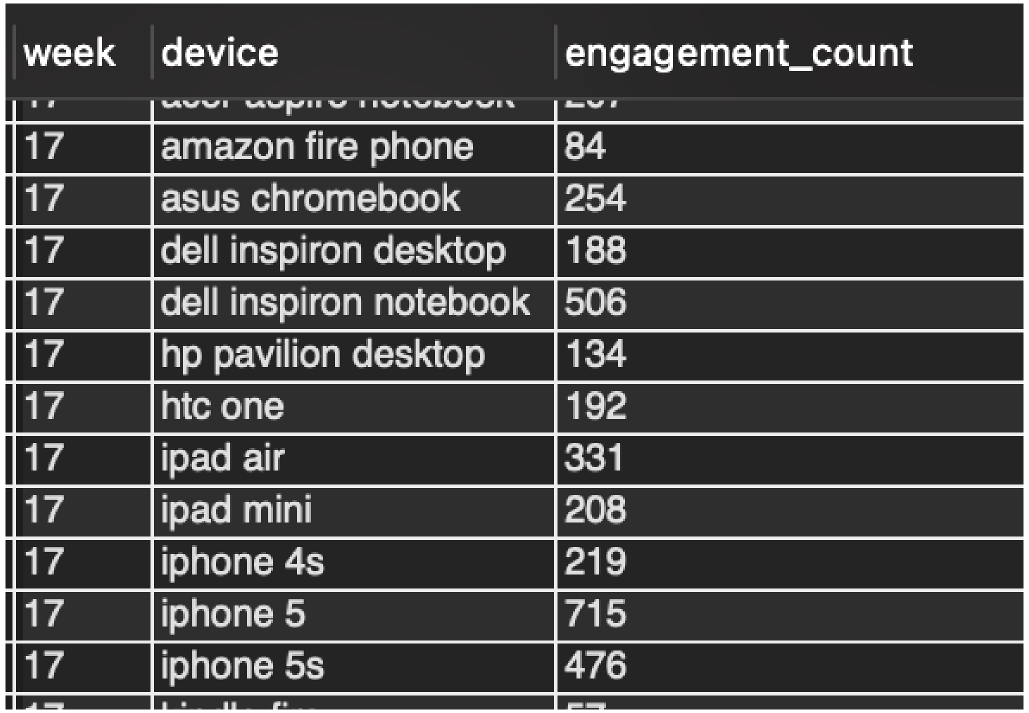
Weekly Retention Analysis



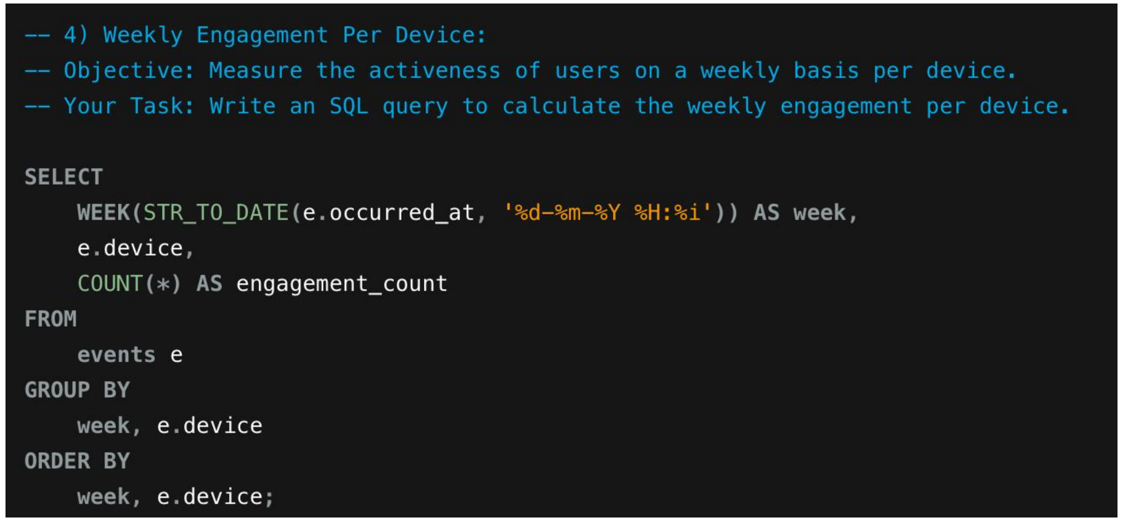
```
-- 3)Weekly Retention Analysis:
-- Objective: Analyze the retention of users on a weekly basis after signing up for a product.
-- Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

WITH user_cohorts AS (
    SELECT
        u.user_id,
        DATE_FORMAT(STR_TO_DATE(u.created_at, '%d-%m-%Y %H:%i'), '%Y-%m') AS cohort_month,
        WEEK(STR_TO_DATE(e.occurred_at, '%d-%m-%Y %H:%i')) AS week
    FROM
        users u
    JOIN
        events e ON u.user_id = e.user_id
    WHERE
        WEEK(STR_TO_DATE(e.occurred_at, '%d-%m-%Y %H:%i')) >= WEEK(STR_TO_DATE(u.created_at, '%d-%m-%Y %H:%i'))
)
SELECT
    cohort_month,
    week,
    COUNT(DISTINCT user_id) AS active_users_retended
FROM
    user_cohorts
GROUP BY
    cohort_month, week
ORDER BY
    cohort_month, week;
```



cohort_month	week	active_users_retended
2013-01	17	11
2013-01	18	17
2013-01	19	18
2013-01	20	20
2013-01	21	20
2013-01	22	27
2013-01	23	21
2013-01	24	27
2013-01	25	20
2013-01	26	13
2013-01	27	17

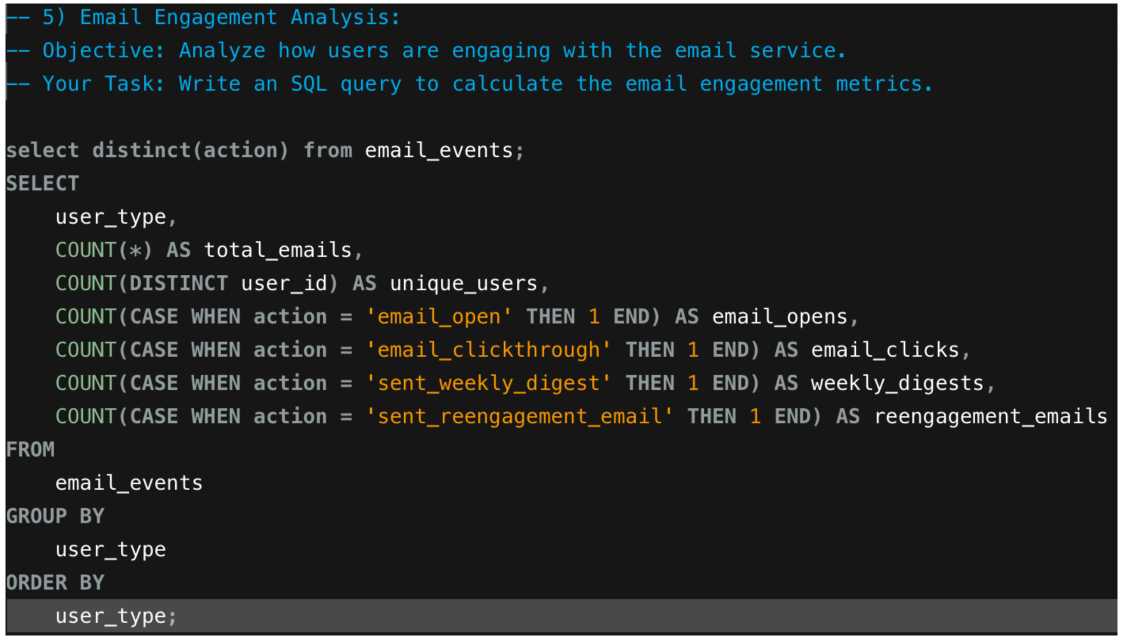


week	device	engagement_count
17	amazon fire phone	84
17	asus chromebook	254
17	dell inspiron desktop	188
17	dell inspiron notebook	506
17	hp pavilion desktop	134
17	htc one	192
17	ipad air	331
17	ipad mini	208
17	iphone 4s	219
17	iphone 5	715
17	iphone 5s	476

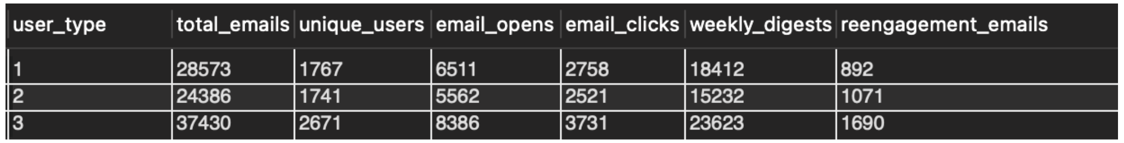


```
-- 4) Weekly Engagement Per Device:
-- Objective: Measure the activeness of users on a weekly basis per device.
-- Your Task: Write an SQL query to calculate the weekly engagement per device.

SELECT
    WEEK(STR_TO_DATE(e.occurred_at, '%d-%m-%Y %H:%i')) AS week,
    e.device,
    COUNT(*) AS engagement_count
FROM
    events e
GROUP BY
    week, e.device
ORDER BY
    week, e.device;
```

Weekly Engagement Per Device



```
-- 5) Email Engagement Analysis:
-- Objective: Analyze how users are engaging with the email service.
-- Your Task: Write an SQL query to calculate the email engagement metrics.

select distinct(action) from email_events;
SELECT
    user_type,
    COUNT(*) AS total_emails,
    COUNT(DISTINCT user_id) AS unique_users,
    COUNT(CASE WHEN action = 'email_open' THEN 1 END) AS email_opens,
    COUNT(CASE WHEN action = 'email_clickthrough' THEN 1 END) AS email_clicks,
    COUNT(CASE WHEN action = 'sent_weekly_digest' THEN 1 END) AS weekly_digests,
    COUNT(CASE WHEN action = 'sent_reengagement_email' THEN 1 END) AS reengagement_emails
FROM
    email_events
GROUP BY
    user_type
ORDER BY
    user_type;
```



user_type	total_emails	unique_users	email_opens	email_clicks	weekly_digests	reengagement_emails
1	28573	1767	6511	2758	18412	892
2	24386	1741	5562	2521	15232	1071
3	37430	2671	8386	3731	23623	1690

Email Engagement Analysis

Results

- **Improved Operations:** Identified areas for improvement in job review processes through metric spike investigations.
- **Enhanced User Engagement:** Provided actionable insights to optimize email campaigns based on user engagement metrics.

Thank You