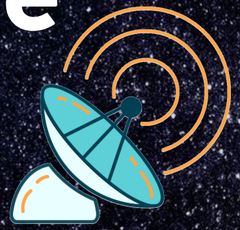
Operation Analytics and Investigating Metric Spike



A Data Analytics Project by Yash Sangwan

Trainity.co

Agenda

- Project overview
- Approach
- Tech-Stack
- Insights
- Results

Project Overview

As Data Analyst Leads for a firm like Microsoft, we are given a variety of data sets and tables to analyse in order to find particular insights and answer queries from other departments.

- Number of jobs reviewed
- Throughput
- Percentage share of each language
- Duplicate rows

- User Engagement
- User Growth
- Weekly Retention
- Weekly Engagement
- Email Engagement

Approach

To help the teams make data-driven decisions, we want to run SQL queries on the provided database.
The following insights will be provided by the SQL queries:

- We calculated the number of jobs reviewed per hour per day for November 2020.
- We calculated 7 day rolling average of throughput. For throughput, we prefer 7 day rolling average. Because, rolling averages are useful for finding long-term trends
- We calculated the percentage share of each language in the last 30 days. Percentage share of each language
- We displayed duplicates from the table.

- We calculated the weekly user engagement.
- We calculated the user growth for product.
- We calculated the weekly retention of users-sign up cohortWeekly Engagement
- We calculated the weekly engagement per device.
- We calculated the email engagement metrics.

TECH STACK

Importing a database and running queries in MySQL by Oracle Corporation to get insights

Microsoft Corporation's Excel is used for data extraction and manipulation.

Insights

Job Data

Number of jobs reviewed:

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

likes

Limit to 1000 rows

```sql
SELECT
    ds AS date,
    COUNT(job_id) AS jobs_reviewed,
    SUM(time_spent) / 3600 AS hours_spent
FROM
    job_data
WHERE
    ds >= '2020-11-01'
        AND ds <= '2020-11-30'
GROUP BY ds
ORDER BY date;
```

Result Grid | Filter Rows: | Export:

date	jobs_reviewed	hours_spent
2020-11-25	1	0.0125
2020-11-26	1	0.0156
2020-11-27	1	0.0289
2020-11-28	2	0.0092
2020-11-29	1	0.0056
2020-11-30	2	0.0111

Insights

Job Data

Throughput:

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

```
Query 1
Limit to 1000 rows
1  WITH throughput AS (SELECT ds as date, COUNT(job_id) AS jobs_reviewed,
2  SUM(time_spent)/3600 AS hours_spent
3  FROM job_data
4  WHERE ds>='2020-11-01' AND ds<= '2020-11-30'
5  GROUP BY DATE
6  ORDER BY DATE)
7  SELECT DATE, SUM(jobs_reviewed) OVER(ORDER BY DATE ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
8  /SUM(hours_spent) OVER(ORDER BY DATE ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS 7_day_rolling_avg
9  FROM throughput;
```

Result Grid | Filter Rows:

date	7_day_rolling_avg
2020-11-25	80.0000
2020-11-26	71.1744
2020-11-27	52.6316
2020-11-28	75.5287
2020-11-29	83.5655
2020-11-30	96.5018

Insights

Job Data

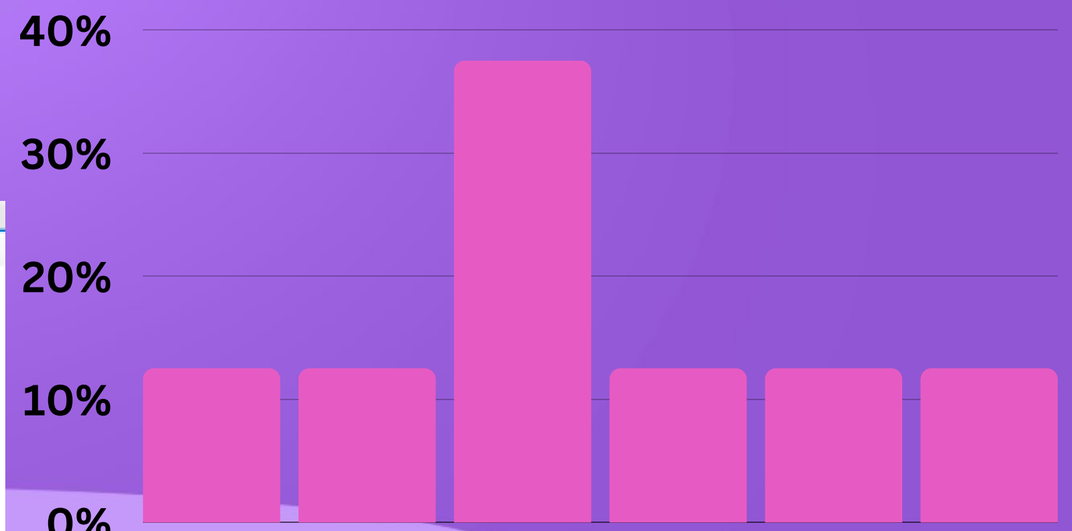
Percentage share of each language:

Our task: Calculate the percentage share of each language in the last 30 days.

Query 1 | job_data

Limit to 1000 rows

```sql
SELECT
    language,
    COUNT(job_id) * 100 / (SELECT COUNT(language) FROM job_data) AS percent_share
FROM
    job_data
GROUP BY language;
```

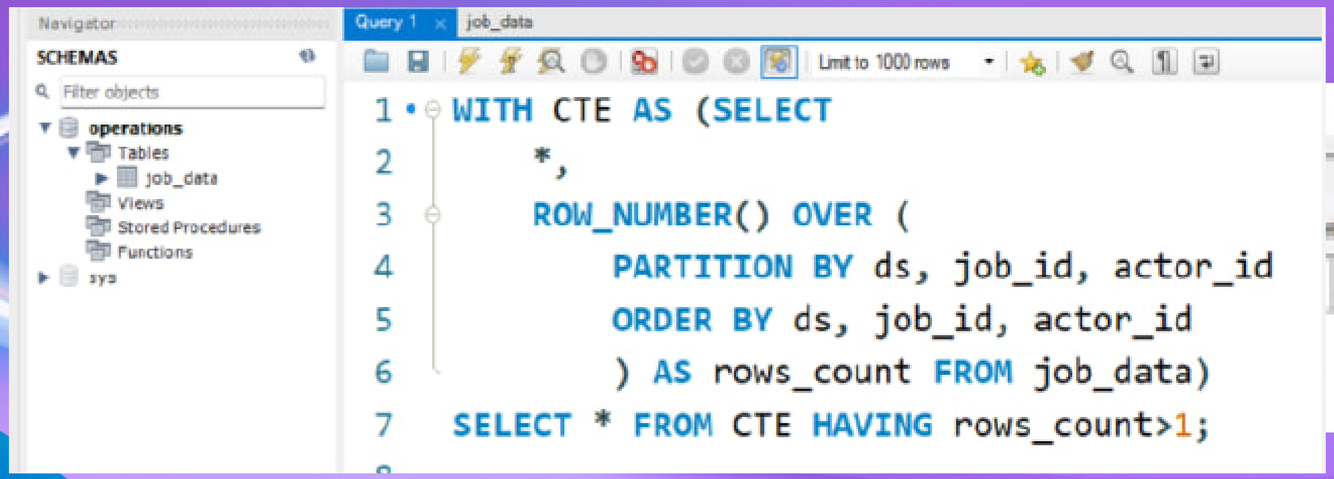


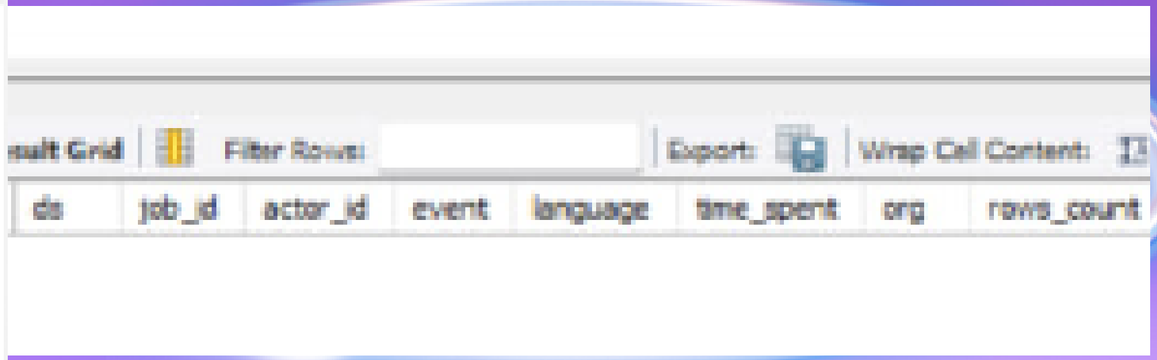
Insights

Job Data

Duplicate rows:

Our task: Let’s say you see some duplicate rows in the data. How will you display duplicates from the table.





Our data doesn't include any duplicate rows, as evidenced by the fact that none of the searches returned any records.

Insights

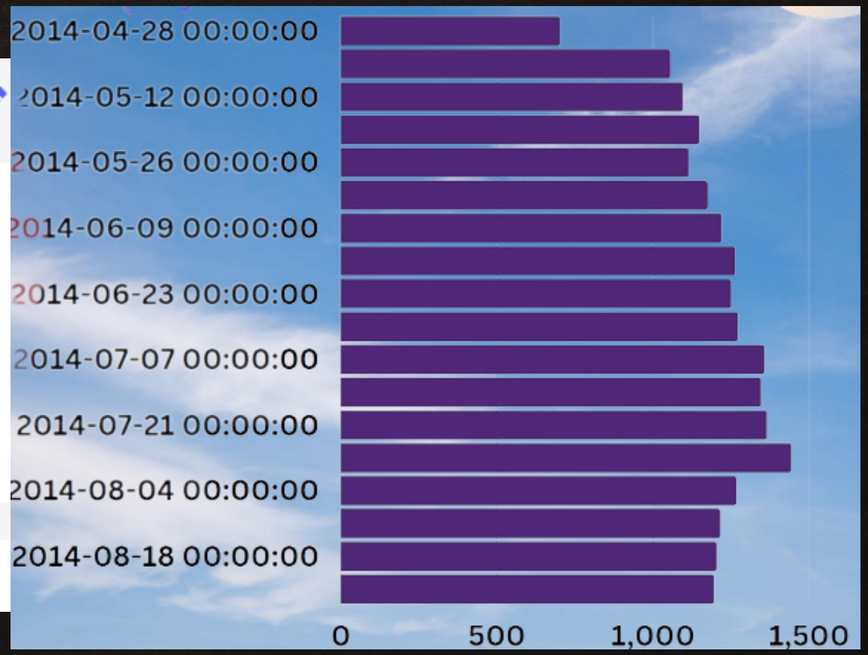
Investigating Metric Spike

User Engagement:

Our task: Calculate the weekly user engagement.

queries.sql — SQL Workbench

```sql
SELECT
DATE_TRUNC('week', e.occurred_at),
COUNT(DISTINCT e.user_id) AS weekly_active_users
FROM
events e
WHERE
e.event_type = 'engagement'
AND e.event_name = 'login'
GROUP BY 1
ORDER BY 1
```



Insights

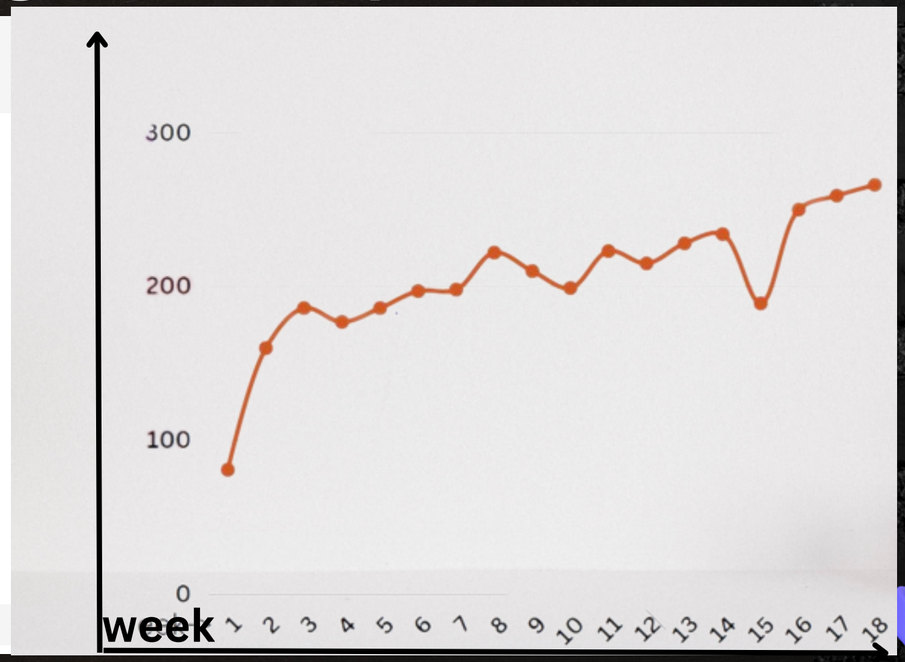
Investigating Metric Spike

User Growth:

Our task: Calculate the user growth for product

queries.sql — SQL Workbench

```sql
WITH CTE AS ( SELECT date_trunc('week', occurred_at) AS
weekly_date,
count(*) over(PARTITION by date_trunc('week', occurred_at) ROWS
BETWEEN unbounded preceding AND current ROW) AS weekly_users
FROM
events
WHERE
event_type = 'signup_flow' AND event_name = 'complete_signup')
SELECT weekly_date,max(weekly_users) AS new_users_per_week
FROM
CTE
GROUP BY
weekly_date
```



Insights

Investigating Metric Spike

Weekly Retention:

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

```sql
SELECT
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-04-28'
      AND a.occurred_at <= '2014-05-04' THEN a.user_id
      ELSE NULL
    END
  ) week_1,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-05-05'
      AND a.occurred_at <= '2014-05-11' THEN a.user_id
      ELSE NULL
    END
  ) AS week_2,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-05-12'
      AND a.occurred_at <= '2014-05-18' THEN a.user_id
      ELSE NULL
    END
  ) AS week_3,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-05-19'
      AND a.occurred_at <= '2014-05-25' THEN a.user_id
      ELSE NULL
    END
  ) AS week_4,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-05-26'
      AND a.occurred_at <= '2014-06-01' THEN a.user_id
      ELSE NULL
    END
  ) AS week_5,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-06-02'
      AND a.occurred_at <= '2014-06-08' THEN a.user_id
      ELSE NULL
    END
  ) AS week_6,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-06-09'
      AND a.occurred_at <= '2014-06-15' THEN a.user_id
      ELSE NULL
    END
  ) AS week_7,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-06-16'
      AND a.occurred_at <= '2014-06-22' THEN a.user_id
      ELSE NULL
    END
  ) AS week_8,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-06-23'
      AND a.occurred_at <= '2014-06-29' THEN a.user_id
      ELSE NULL
    END
  ) AS week_9,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-06-30'
      AND a.occurred_at <= '2014-07-06' THEN a.user_id
      ELSE NULL
    END
  ) AS week_10,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-07-07'
      AND a.occurred_at <= '2014-07-13' THEN a.user_id
      ELSE NULL
    END
  ) AS week_11,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-07-14'
      AND a.occurred_at <= '2014-07-20' THEN a.user_id
      ELSE NULL
    END
  ) AS week_12,
  count(
    DISTINCT CASE
      WHEN a.event_name = 'complete_signup'
      AND b.event_type = 'engagement'
      AND b.event_name = 'login'
      AND a.occurred_at >= '2014-07-21'
      AND a.occurred_at <= '2014-07-27' THEN a.user_id
      ELSE NULL
    END
```

Insights

Investigating Metric Spike

Weekly Retention:

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

```
131   ) AS week_13,
132   count(
133     DISTINCT CASE
134       WHEN a.event_name = 'complete_signup'
135       AND b.event_type = 'engagement'
136       AND b.event_name = 'login'
137       AND a.occurred_at >= '2014-07-28'
138       AND a.occurred_at <= '2014-08-03' THEN a.user_id
139       ELSE NULL
140     END
141   ) AS week_14,
142   count(
143     DISTINCT CASE
144       WHEN a.event_name = 'complete_signup'
145       AND b.event_type = 'engagement'
146       AND b.event_name = 'login'
147       AND a.occurred_at >= '2014-08-04'
148       AND a.occurred_at <= '2014-08-10' THEN a.user_id
149       ELSE NULL
150     END
151   ) AS week_15,
152   count(
153     DISTINCT CASE
154       WHEN a.event_name = 'complete_signup'
155       AND b.event_type = 'engagement'
156       AND b.event_name = 'login'
157       AND a.occurred_at >= '2014-08-11'
158       AND a.occurred_at <= '2014-08-17' THEN a.user_id
159       ELSE NULL
160     END
161   ) AS week_16,
162   count(
163     DISTINCT CASE
164       WHEN a.event_name = 'complete_signup'
165       AND b.event_type = 'engagement'
166       AND b.event_name = 'login'
167       AND a.occurred_at >= '2014-08-18'
168       AND a.occurred_at <= '2014-08-24' THEN a.user_id
169       ELSE NULL
170     END
171   ) AS week_17,
172   count(
173     DISTINCT CASE
174       WHEN a.event_name = 'complete_signup'
```

```
175       AND b.event_type = 'engagement'
176       AND b.event_name = 'login'
177       AND a.occurred_at >= '2014-08-25'
178       AND a.occurred_at <= '2014-08-31' THEN a.user_id
179       ELSE NULL
180     END
181   ) AS week_18
182 FROM
183   tutorial.yammer_events a
184   JOIN tutorial.yammer_events b ON a.user_id = b.user_id
```

Insights

Investigating Metric Spike

Weekly Retention:

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

Insights

Investigating Metric Spike

Weekly Engagement:

Our task: Calculate the weekl engagement per device

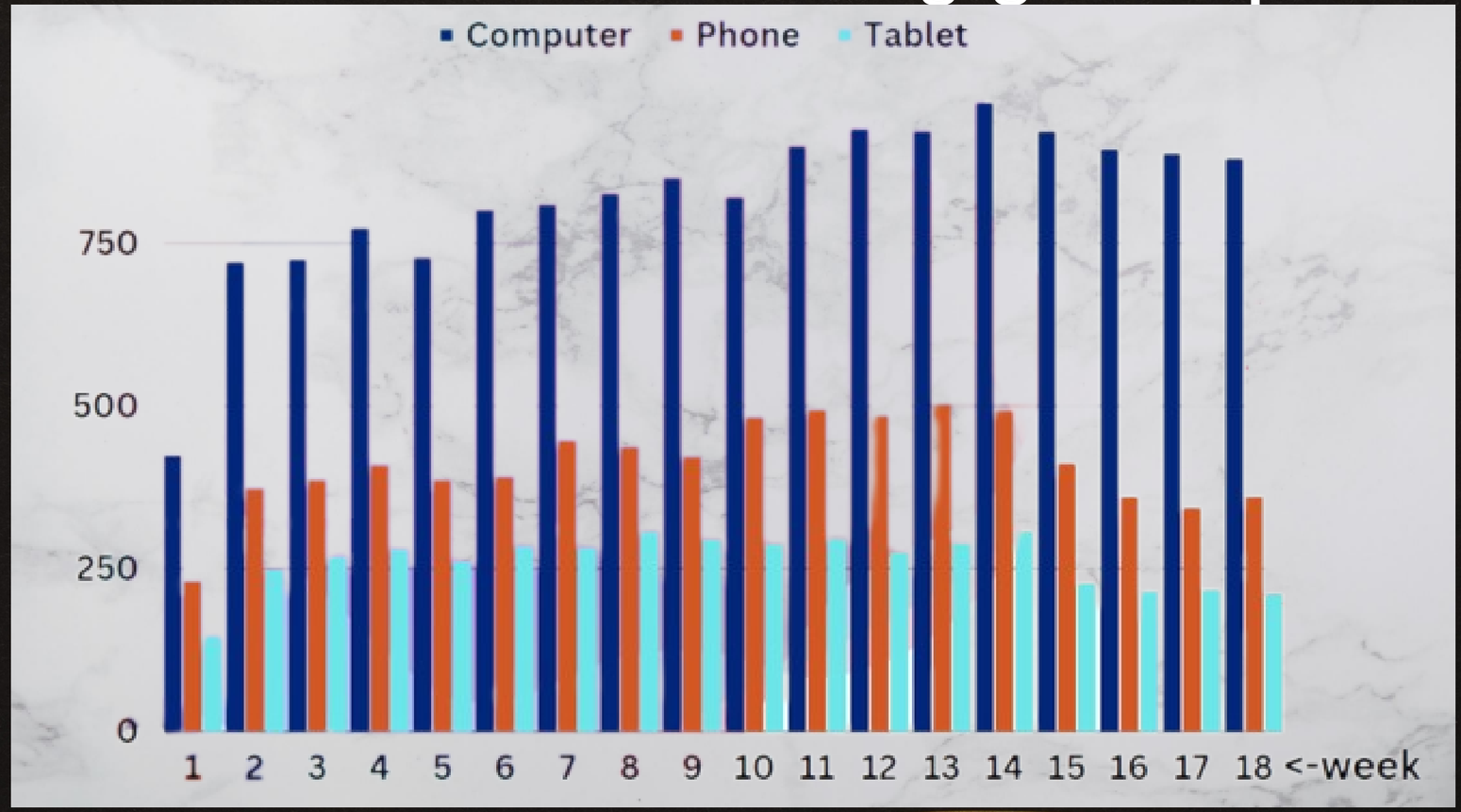
```sql
SELECT DATE_TRUNC('week', occurred_at) AS week,
COUNT(DISTINCT e.user_id) AS weekly_users,
COUNT(DISTINCT CASE WHEN e.device IN ('macbook pro', 'acer
aspire notebook','acer aspire desktop','lenovo thinkpad', 'mac mini',
'dell inspiron desktop','dell inspiron notebook','windows
surface','macbook air','asus chromebook','hp pavilion desktop') THEN
e.user_id ELSE NULL END) AS computer,
COUNT(DISTINCT CASE WHEN e.device IN ('iphone 5s','nokia lumia
635','amazon fire phone','iphone 4s','htc one','iphone 5','samsung
galaxy s4') THEN e.user_id ELSE NULL END) AS phone,
COUNT(DISTINCT CASE WHEN e.device IN ('kindle fire','samsung
galaxy note','ipad mini','nexus 7','nexus 10','samsumg galaxy
tablet','nexus 5','ipad air') THEN e.user_id ELSE NULL END) AS tablet
FROM events e
WHERE e.event_type = 'engagement'
AND e.event_name = 'login'
GROUP BY 1
order by 1
```

Insights

Investigating Metric Spike

Weekly Engagement:

Our task: Calculate the weekl engagement per device



Insights

Investigating Metric Spike

Email Engagement:

Our task: Calculate the email engagement metrics

```sql
SELECT DATE_TRUNC('week', occurred_at) AS week,
COUNT(CASE WHEN e.action = 'sent_weekly_digest' THEN
e.user_id ELSE NULL END) AS weekly_emails,
COUNT(CASE WHEN e.action = 'sent_reengagement_email'
THEN e.user_id ELSE NULL END) AS reengagement_emails,
COUNT(CASE WHEN e.action = 'email_open' THEN e.user_id
ELSE NULL END) AS email_opens,
COUNT(CASE WHEN e.action = 'email_clickthrough' THEN
e.user_id ELSE NULL END) AS email_clickthroughs
FROM emails e
GROUP BY 1
ORDER BY 1
```

	week	weekly_emails	reengagement_emails	email_opens	email_clickthroughs
1	2014-04-28 00:00:00	908	98	332	187
2	2014-05-05 00:00:00	2602	164	919	434
3	2014-05-12 00:00:00	2665	175	971	479
4	2014-05-19 00:00:00	2733	179	995	498
5	2014-05-26 00:00:00	2822	179	1026	453
6	2014-06-02 00:00:00	2911	199	993	492
7	2014-06-09 00:00:00	3003	190	1070	533
8	2014-06-16 00:00:00	3105	234	1161	563
9	2014-06-23 00:00:00	3207	187	1090	524
10	2014-06-30 00:00:00	3302	222	1168	559
11	2014-07-07 00:00:00	3399	214	1230	622
12	2014-07-14 00:00:00	3499	226	1260	607
13	2014-07-21 00:00:00	3592	206	1211	584
14	2014-07-28 00:00:00	3706	230	1386	633
15	2014-08-04 00:00:00	3793	206	1336	432
16	2014-08-11 00:00:00	3897	224	1357	430
17	2014-08-18 00:00:00	4012	257	1421	487
18	2014-08-25 00:00:00	4111	263	1533	493

Results

This project's foundation in advanced SQL ideas enabled me to gain a more accurate understanding of data analytics. We made an effort to use graphics like charts and tables to make the results more illustrative.

Business leaders may utilize the spike data to improve or change the product because they are plainly visible in the graphs.


Thank You