

# Operation Analytics and Investigating Metric Spike

A Data Analytics Project by Yash Sangwan



# Agenda



Project overview

Approach

Tech-Stack

Insights

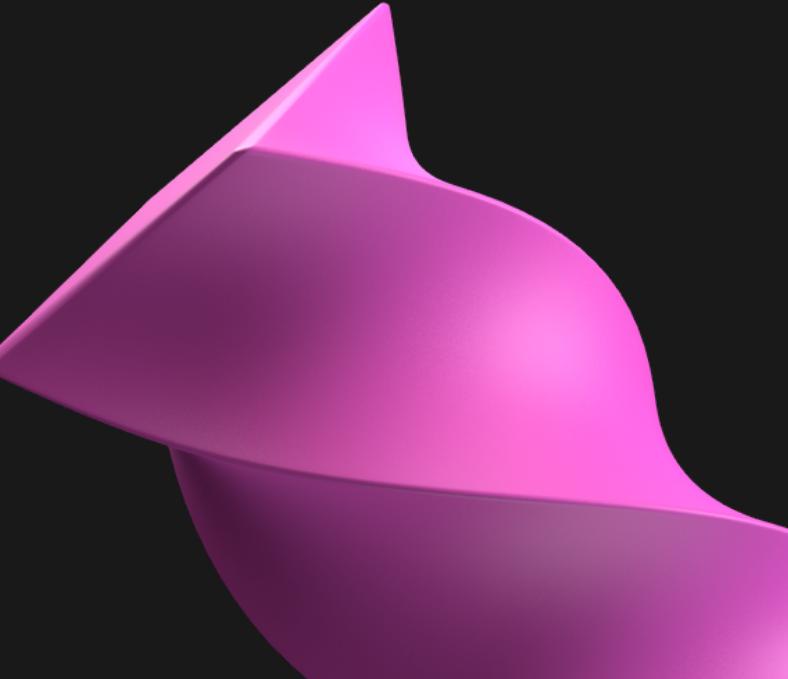
Results

# 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



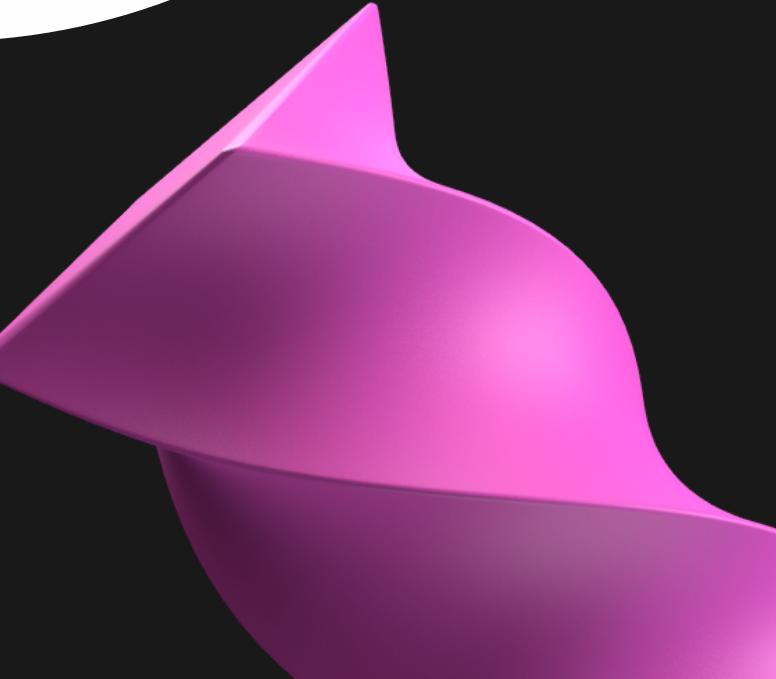
# Approach

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 x
1 • SELECT
2     ds AS date,
3     COUNT(job_id) AS jobs_reviewed,
4     SUM(time_spent) / 3600 AS hours_spent
5 FROM
6     job_data
7 WHERE
8     ds >= '2020-11-01'
9         AND ds <= '2020-11-30'
10 GROUP BY ds
11 ORDER BY date;
```

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.

```
Query1 x
WITH throughput AS (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 DATE
ORDER BY DATE)
SELECT DATE, SUM(jobs_reviewed) OVER(ORDER BY DATE ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
/SUM(hours_spent) OVER(ORDER BY DATE ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS 7_day_rolling_avg
FROM throughput;
```

	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

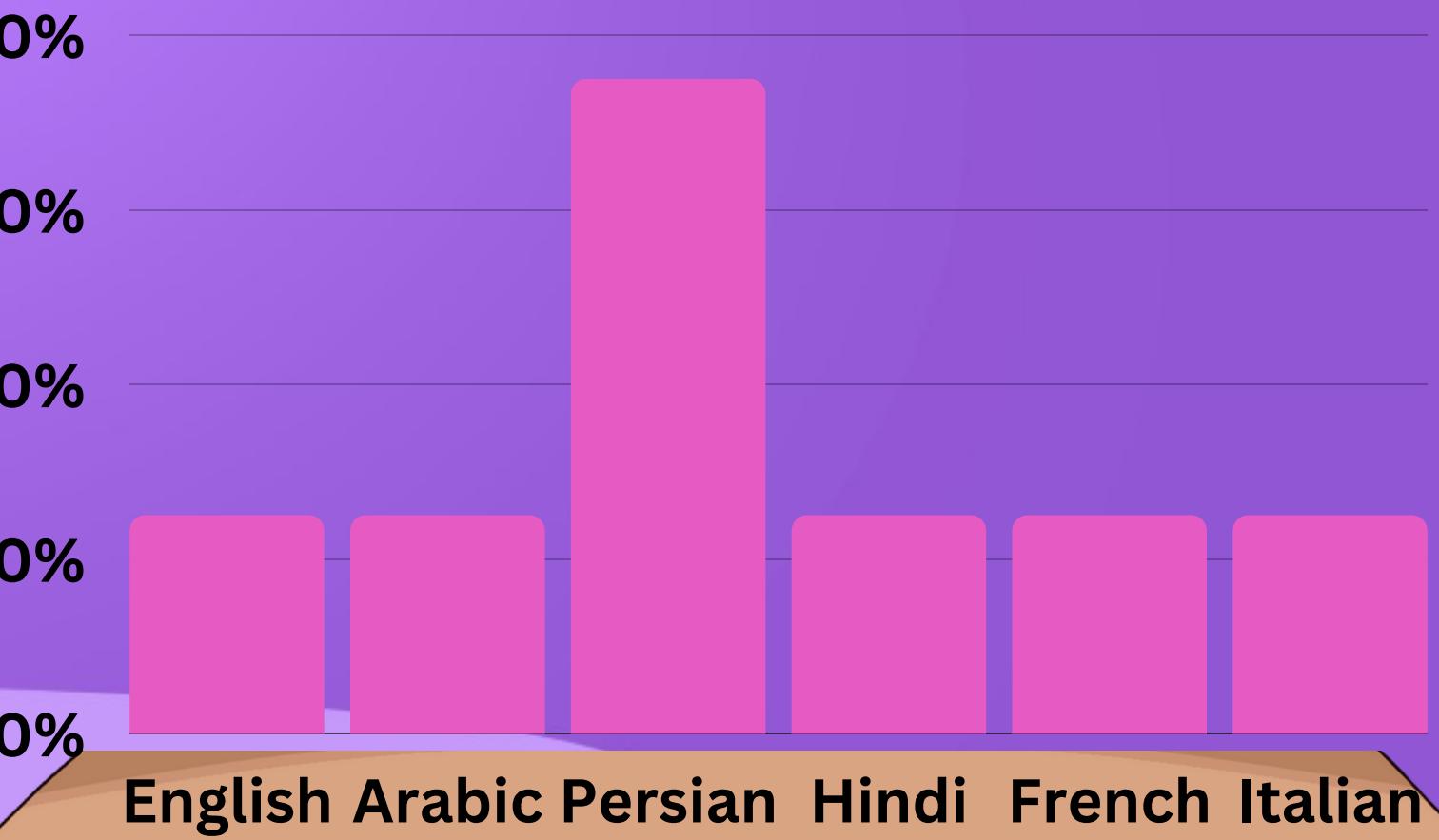
# Insights

## Job Data

Percentage share of each language:

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

```
Query 1 x job_data
1 • SELECT
2     language,
3     COUNT(job_id) * 100 / (SELECT COUNT(language) FROM job_data) AS percent_share
4 FROM
5     job_data
6 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.

The screenshot shows a database interface with a query editor and a results grid. The query editor contains the following SQL code:

```
1 • 1 WITH CTE AS (SELECT
2     *
3     ROW_NUMBER() OVER (
4         PARTITION BY ds, job_id, actor_id
5         ORDER BY ds, job_id, actor_id
6     ) AS rows_count FROM job_data)
7     SELECT * FROM CTE HAVING rows_count>1;
```

The results grid shows columns: ds, job\_id, actor\_id, event, language, time\_spent, org, rows\_count. There are no rows present in the grid.

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

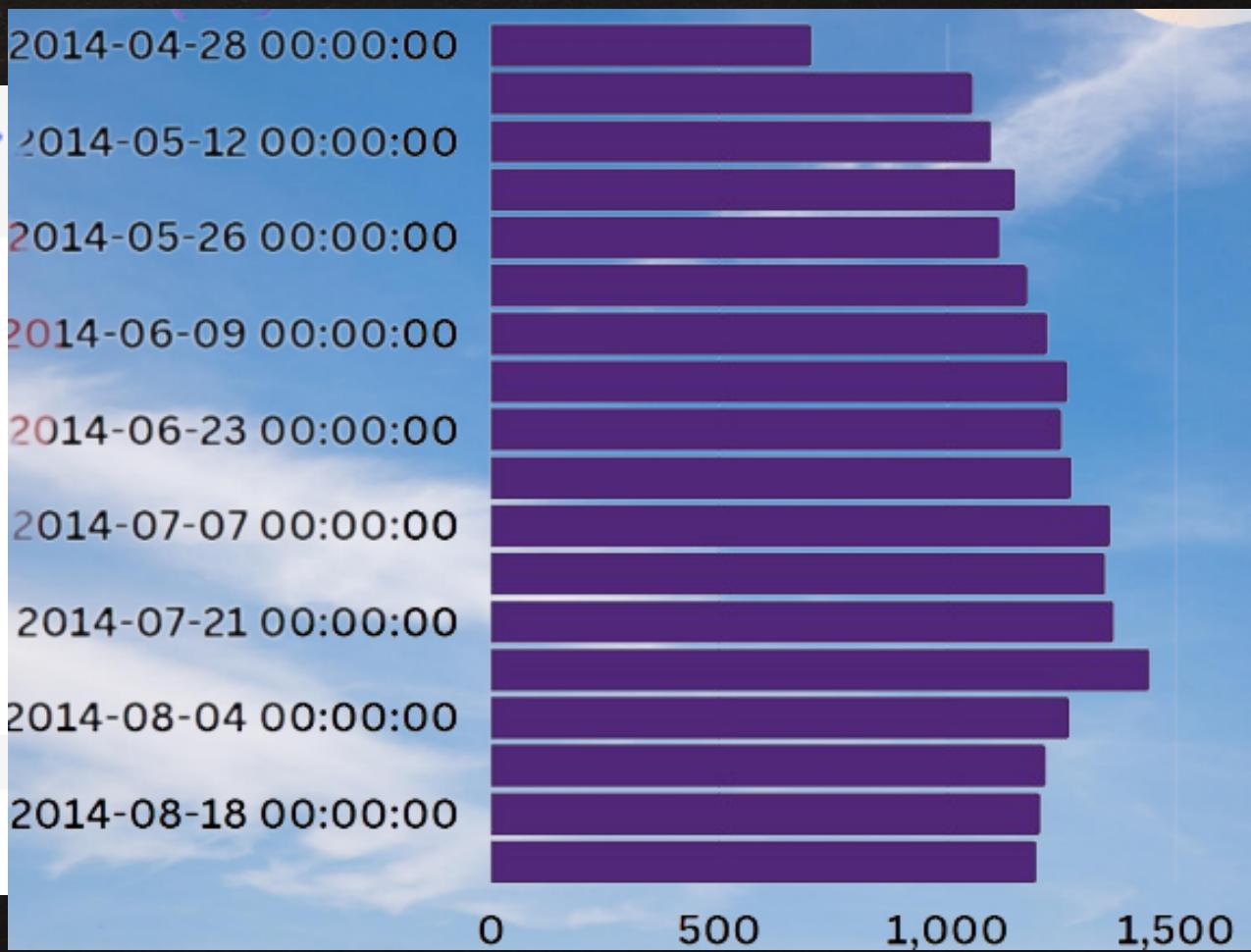
### User Engagement:

Our task: Calculate the weekly user engagement.

queries.sql

```
1 SELECT
2 DATE_TRUNC('week', e.occurred_at),
3 COUNT(DISTINCT e.user_id) AS weekly_active_users
4 FROM
5 events e
6 WHERE
7 e.event_type = 'engagement'
8 AND e.event_name = 'login'
9 GROUP BY 1
10 ORDER BY 1
```

SQL Workbench



# Insights

## Investigating Metric Spike

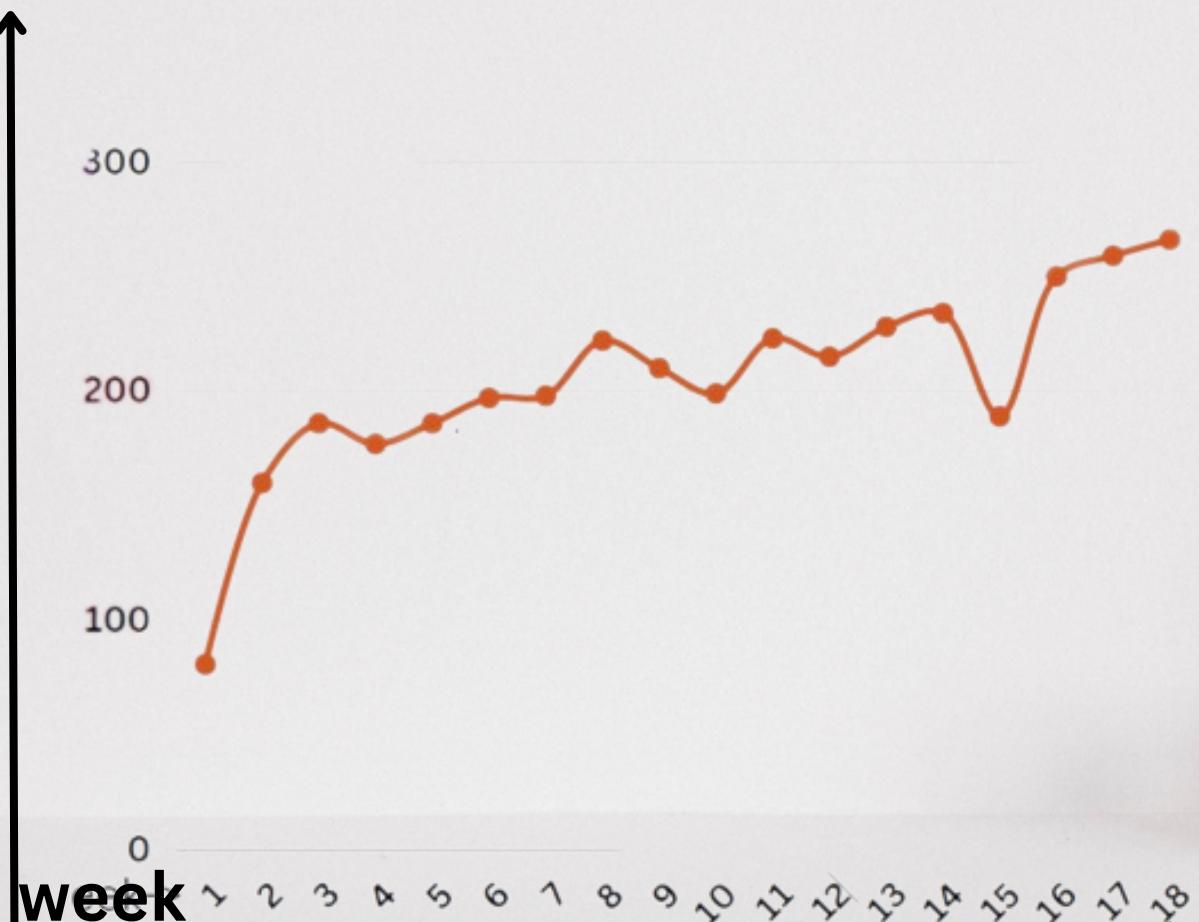
User Growth:

Our task: Calculate the user growth for product

queries.sql

```
1 WITH CTE AS ( SELECT date_trunc('week', occurred_at) AS
2 weekly_date,
3 count(*) over(PARTITION by date_trunc('week', occurred_at) ROWS
4 BETWEEN unbounded preceding AND current ROW) AS weekly_users
5 FROM
6 events
7 WHERE
8 event_type = 'signup_flow' AND event_name = 'complete_signup')
9 SELECT weekly_date,max(weekly_users) AS new_users_per_week
10 FROM
11 CTE
12 GROUP BY
13 weekly_date
```

SQL Workbench



# Insights

## Investigating Metric Spike

---

### Weekly Retention:

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

```
1  SELECT
2    count(
3      DISTINCT CASE
4        WHEN a.event_name = 'complete_signup'
5          AND b.event_type = 'engagement'
6          AND b.event_name = 'login'
7          AND a.occurred_at >= '2014-04-28'
8          AND a.occurred_at <= '2014-05-04' THEN a.user_id
9        ELSE NULL
10      END
11    ) AS week_1,
12    count(
13      DISTINCT CASE
14        WHEN a.event_name = 'complete_signup'
15          AND b.event_type = 'engagement'
16          AND b.event_name = 'login'
17          AND a.occurred_at >= '2014-05-05'
18          AND a.occurred_at <= '2014-05-11' THEN a.user_id
19        ELSE NULL
20      END
21    ) AS week_2,
22    count(
23      DISTINCT CASE
24        WHEN a.event_name = 'complete_signup'
25          AND b.event_type = 'engagement'
26          AND b.event_name = 'login'
27          AND a.occurred_at >= '2014-05-12'
28          AND a.occurred_at <= '2014-05-18' THEN a.user_id
29        ELSE NULL
30      END
31    ) AS week_3,
32    count(
33      DISTINCT CASE
34        WHEN a.event_name = 'complete_signup'
35          AND b.event_type = 'engagement'
36          AND b.event_name = 'login'
37          AND a.occurred_at >= '2014-05-19'
38          AND a.occurred_at <= '2014-05-25' THEN a.user_id
39        ELSE NULL
40      END
41    ) AS week_4,
42    count(
43      DISTINCT CASE
44        WHEN a.event_name = 'complete_signup'
```

```
43      AND b.event_type = 'engagement'
44      AND b.event_name = 'login'
45      AND a.occurred_at >= '2014-05-26'
46      AND a.occurred_at <= '2014-06-01' THEN a.user_id
47      ELSE NULL
48    END
49  ) AS week_5,
50  count(
51    DISTINCT CASE
52      WHEN a.event_name = 'complete_signup'
53        AND b.event_type = 'engagement'
54        AND b.event_name = 'login'
55        AND a.occurred_at >= '2014-06-02'
56        AND a.occurred_at <= '2014-06-08' THEN a.user_id
57        ELSE NULL
58    END
59  ) AS week_6,
60  count(
61    DISTINCT CASE
62      WHEN a.event_name = 'complete_signup'
63        AND b.event_type = 'engagement'
64        AND b.event_name = 'login'
65        AND a.occurred_at >= '2014-06-09'
66        AND a.occurred_at <= '2014-06-15' THEN a.user_id
67        ELSE NULL
68    END
69  ) AS week_7,
70  count(
71    DISTINCT CASE
72      WHEN a.event_name = 'complete_signup'
73        AND b.event_type = 'engagement'
74        AND b.event_name = 'login'
75        AND a.occurred_at >= '2014-06-16'
76        AND a.occurred_at <= '2014-06-22' THEN a.user_id
77        ELSE NULL
78    END
79  ) AS week_8,
80  count(
81    DISTINCT CASE
82      WHEN a.event_name = 'complete_signup'
83        AND b.event_type = 'engagement'
84        AND b.event_name = 'login'
85        AND a.occurred_at >= '2014-06-23'
86        AND a.occurred_at <= '2014-06-29' THEN a.user_id
87        ELSE NULL
88    END
89  ) AS week_9,
90  count(
91    DISTINCT CASE
92      WHEN a.event_name = 'complete_signup'
93        AND b.event_type = 'engagement'
94        AND b.event_name = 'login'
95        AND a.occurred_at >= '2014-06-30'
96        AND a.occurred_at <= '2014-07-06' THEN a.user_id
97        ELSE NULL
98    END
99  ) AS week_10,
100  count(
101    DISTINCT CASE
102      WHEN a.event_name = 'complete_signup'
103        AND b.event_type = 'engagement'
104        AND b.event_name = 'login'
105        AND a.occurred_at >= '2014-07-07'
106        AND a.occurred_at <= '2014-07-13' THEN a.user_id
107        ELSE NULL
108    END
109  ) AS week_11,
110  count(
111    DISTINCT CASE
112      WHEN a.event_name = 'complete_signup'
113        AND b.event_type = 'engagement'
114        AND b.event_name = 'login'
115        AND a.occurred_at >= '2014-07-14'
116        AND a.occurred_at <= '2014-07-20' THEN a.user_id
117        ELSE NULL
118    END
119  ) AS week_12,
120  count(
121    DISTINCT CASE
122      WHEN a.event_name = 'complete_signup'
123        AND b.event_type = 'engagement'
124        AND b.event_name = 'login'
125        AND a.occurred_at >= '2014-07-21'
126        AND a.occurred_at <= '2014-07-27' THEN a.user_id
127        ELSE NULL
128    END
129  )
```

# 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 weekly engagement per device

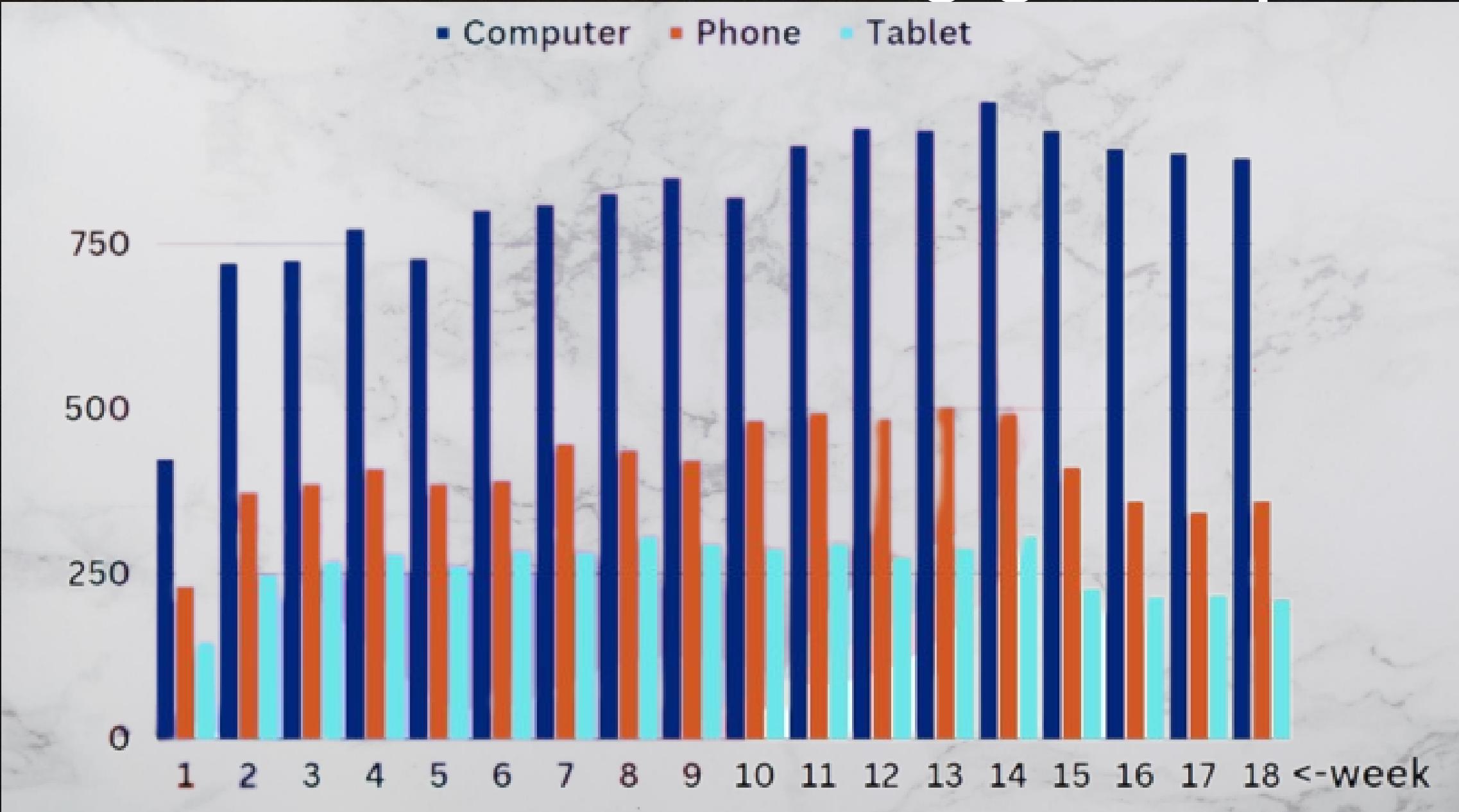
```
1  SELECT DATE_TRUNC('week', occurred_at) AS week,
2  COUNT(DISTINCT e.user_id) AS weekly_users,
3  COUNT(DISTINCT CASE WHEN e.device IN ('macbook pro', 'acer
4  aspire notebook', 'acer aspire desktop', 'lenovo thinkpad', 'mac mini',
5  'dell inspiron desktop', 'dell inspiron notebook', 'windows
6  surface', 'macbook air', 'asus chromebook', 'hp pavilion desktop') THEN
7  e.user_id ELSE NULL END) AS computer,
8  COUNT(DISTINCT CASE WHEN e.device IN ('iphone 5s', 'nokia lumia
9  635', 'amazon fire phone', 'iphone 4s', 'htc one', 'iphone 5', 'samsung
10 galaxy s4') THEN e.user_id ELSE NULL END) AS phone,
11 COUNT(DISTINCT CASE WHEN e.device IN ('kindle fire', 'samsung
12 galaxy note', 'ipad mini', 'nexus 7', 'nexus 10', 'samsung galaxy
13 tablet', 'nexus 5', 'ipad air') THEN e.user_id ELSE NULL END) AS tablet
14 FROM events e
15 WHERE e.event_type = 'engagement'
16 AND e.event_name = 'login'
17 GROUP BY 1
18 order by 1
```

# Insights

## Investigating Metric Spike

Weekly Engagement:

Our task: Calculate the weekly engagement per device



# Insights

## Investigating Metric Spike

### Email Engagement:

Our task: Calculate the email engagement metrics

```
1 SELECT DATE_TRUNC('week', occurred_at) AS week,
2   COUNT(CASE WHEN e.action = 'sent_weekly_digest' THEN
3     e.user_id ELSE NULL END) AS weekly_emails,
4   COUNT(CASE WHEN e.action = 'sent_reengagement_email'
5     THEN e.user_id ELSE NULL END) AS reengagement_emails,
6   COUNT(CASE WHEN e.action = 'email_open' THEN e.user_id
7     ELSE NULL END) AS emailOpens,
8   COUNT(CASE WHEN e.action = 'email_clickthrough' THEN
9     e.user_id ELSE NULL END) AS emailClickthroughs
10  FROM emails e
11  GROUP BY 1
12  ORDER BY 1
```

	week	weekly_emails	reengagement_emails	emailOpens	emailClickthroughs
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