Operation Analytics and Investigating Metric Spike

AGENDA

- DESCRIPTION
- APPROACH
- TECH-STACK USED
- ANALYSIS

DESCRIPTION

Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. This analysis helps identify areas for improvement within the company. As a Data Analyst, I’ll work closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect.

One of the key aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales. As a Data Analyst, I’ll answer these questions daily, making it crucial to understand how to investigate these metric spikes.

In this project, I'll take on the role of a Lead Data Analyst at a company like Microsoft. I'll be provided with various datasets and tables, and your task will be to derive insights from this data to answer questions posed by different departments within the company. My goal is to use my advanced SQL skills to analyze the data and provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.

APPROACH

<u>Role Overview: Data Analyst Lead at Microsoft</u>

- **Approach:**
  - ➢ Download all provided data.
  - ➢ Create a database with Case Study 1 (Job Data).
  - ➢ Upload datasets for Case Study 2 (Investigating metric spikes) to MySQL Workbench.
  - ➢ Gather insights, write queries, analyze data, make decisions based on findings.

TECH-STACK USED

MySQL

I used MySQL Workbench to database, store data, and make queries.

Excel

I used Microsoft Excel a lot to create visual representations of the data.

ANALYSIS

Focus on User Engagement for Metric Spike Analysis

- **Case Study Approach:**
  - **Case Study 1 (Job Data):** Created database from provided Excel sheet.
  - **Case Study 2 (Metric Spikes):** Uploaded datasets to MySQL Workbench, used SQL queries to answer questions effectively.

Case Study 1

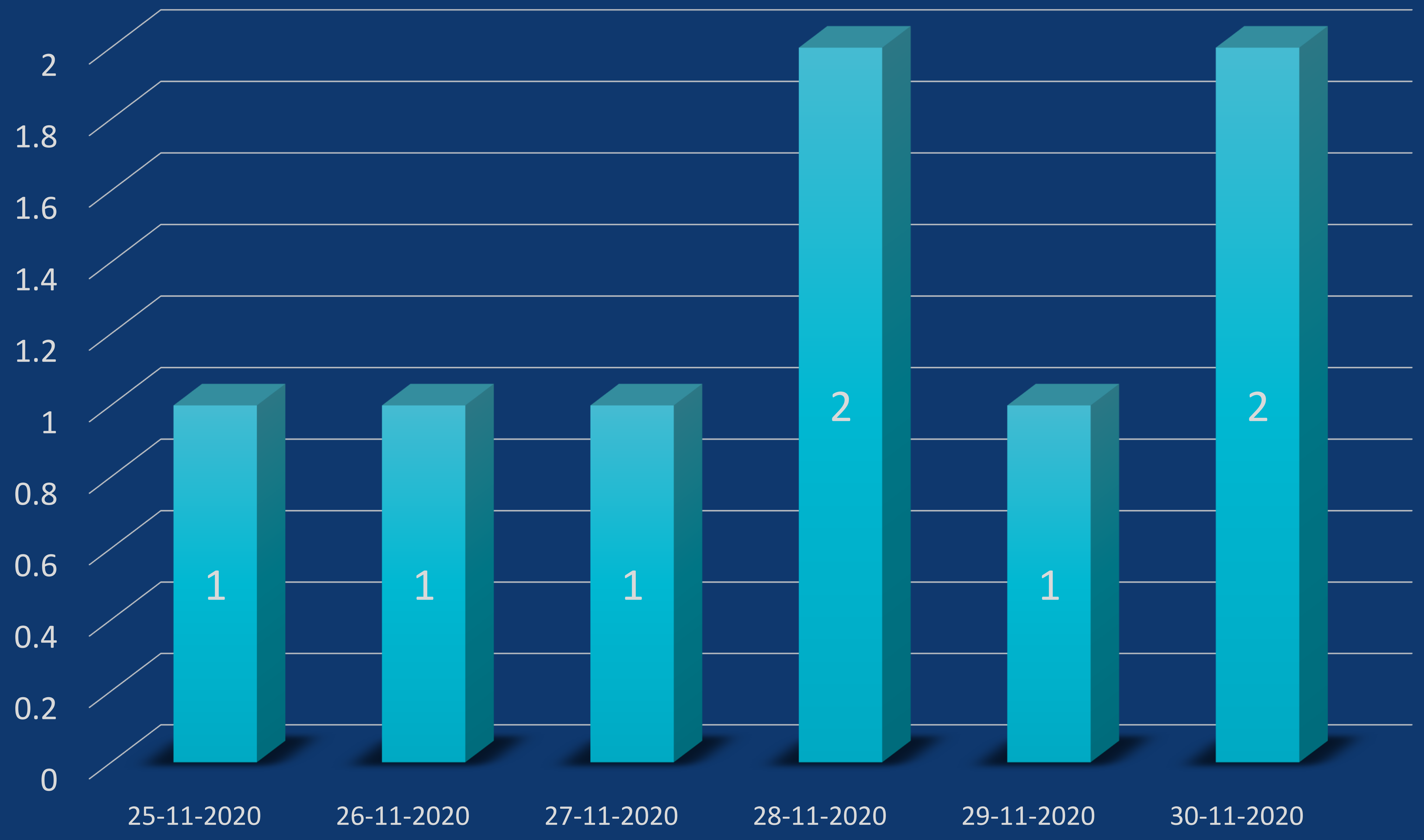
JOB DATA ANALYSIS TOOLS

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

Jobs Reviewed Over Time

```
SELECT
COUNT(job_id) AS job_reviewed,
DATE(ds) as date
FROM job_data
WHERE DATE(ds) BETWEEN '2020-11-01' AND
'2020-11-30'
GROUP BY date ;
```

job_reviewed	date
2	2020/11/30
1	2020/11/29
2	2020/11/28
1	2020/11/27
1	2020/11/26
1	2020/11/25

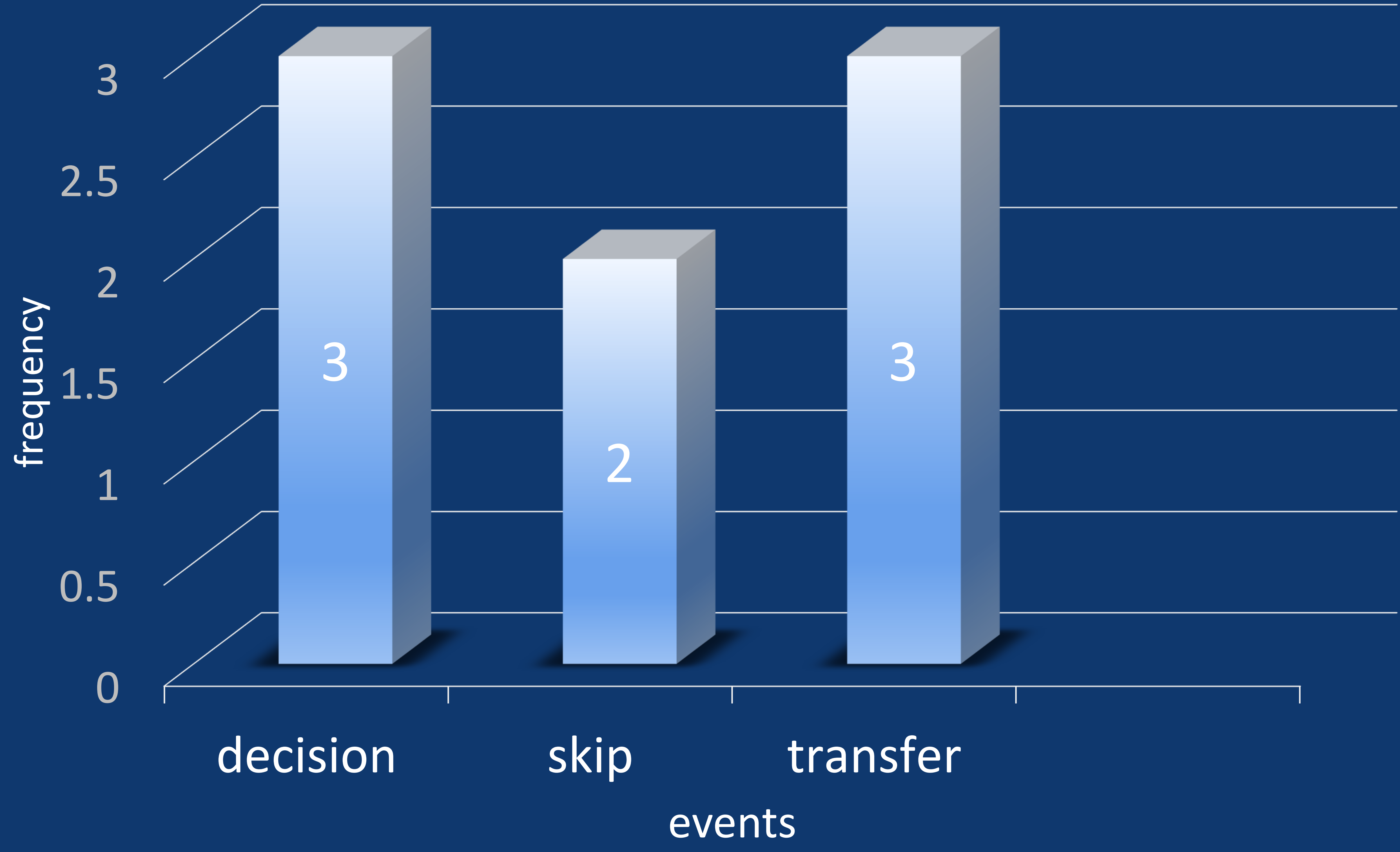


This query calculates the number of jobs reviewed per day in November 2020. It counts the number of unique job IDs for each date within the specified date range.

Throughput Analysis

```
SELECT    event,
COUNT(*) AS frequency
FROM    job_data
GROUP BY    event;
```

event	frequency
decision	3
skip	2
transfer	3



I prefer using the 7-day rolling average for throughput because it smooths out daily fluctuations and provides a more stable metric for analyzing trends over time.

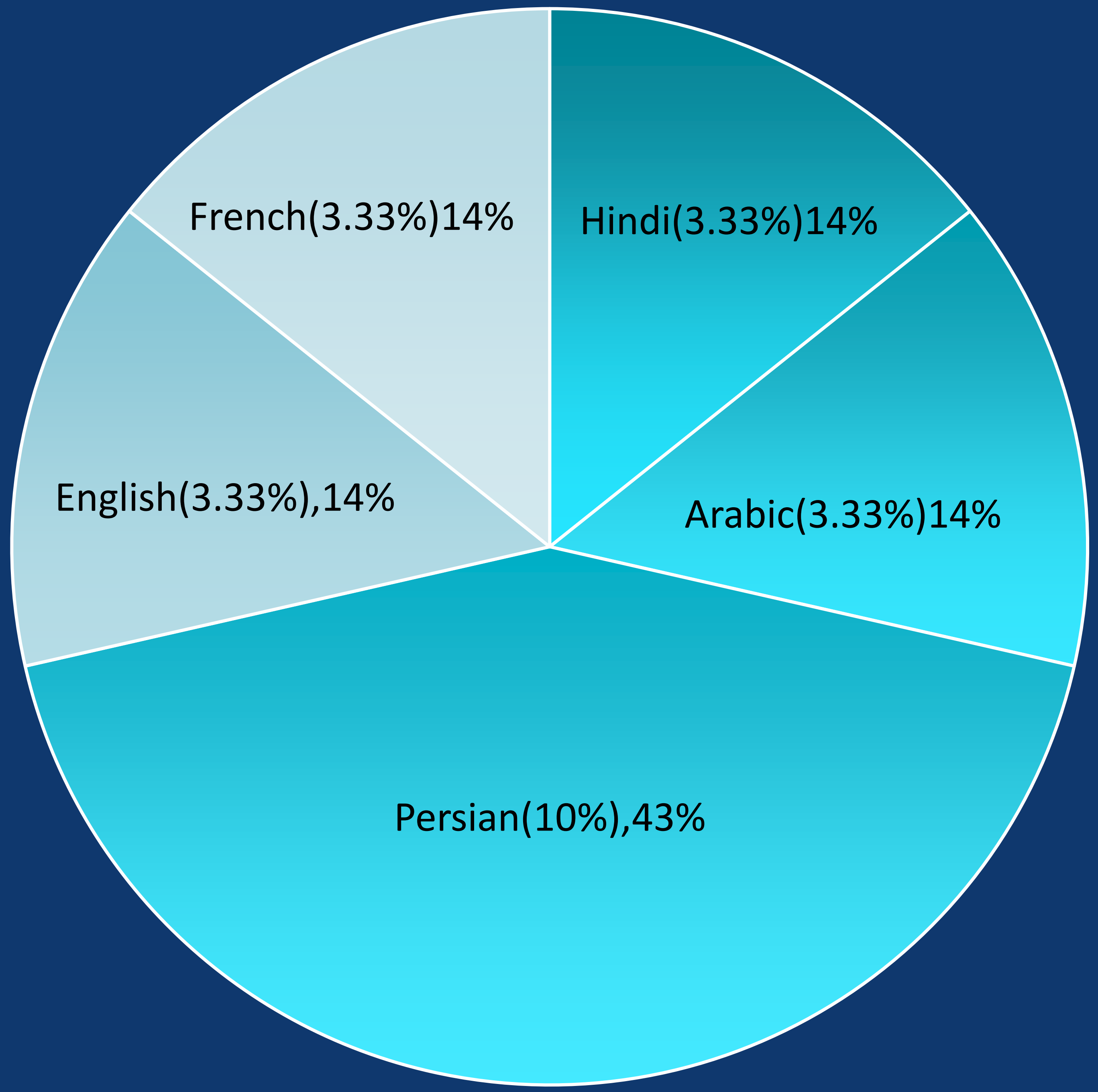
The query returns a result set with columns for date (ds) and avg_throughput, which represents the 7-day rolling average of throughput.

Language Share Analysis

```
SELECT language,
CONCAT(ROUND((COUNT(*) / 30) * 100, 2), "%")
AS percentage_share
FROM    job_data
GROUP BY    language;
```

language	percentage_share
English	3.33%
Arabic	3.33%
Persian	10.00%
Hindi	3.33%
French	3.33%

By Calculating the Persian has highest percentage share (10%) and rest each has percentage share (3.33%) in the last 30 days.



The query returns a result set with columns for language, events_count (the number of events for each language in the last 30 days), and percentage_share (the percentage share of each language in the last 30 days)

Duplicate Rows Detection

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

0

Ds	job_id	actor_id	event	language	time_spent	org

No duplicate rows in the data.

The query returns duplicate rows from the job_data table, showing the duplicated records based on the specified columns.

Case Study 2

INVESTIGATING METRIC SPIKE

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

Weekly User Engagement

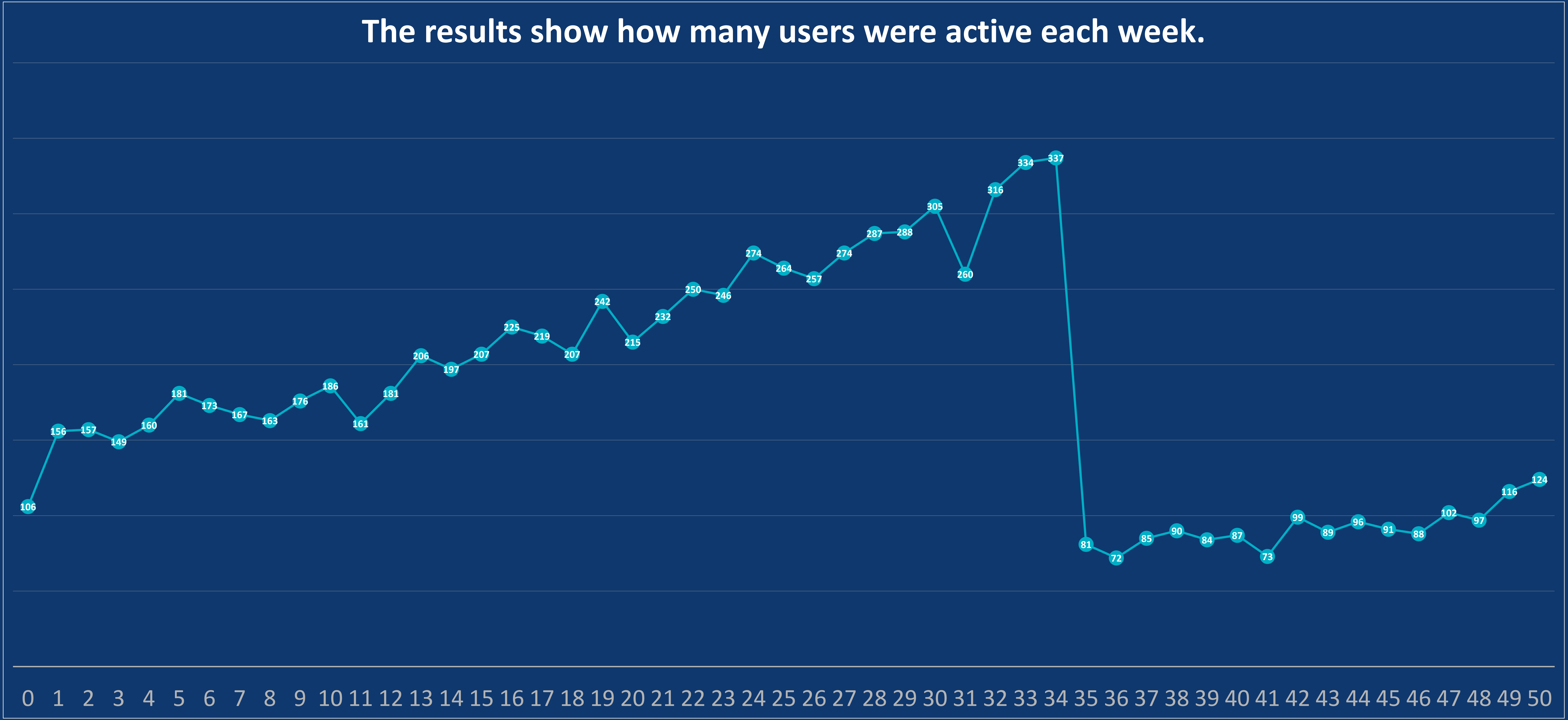
SQL Query:

```
SELECT WEEK(created_at)
AS week_number,
COUNT(DISTINCT user_id)
AS active_users
FROM users
GROUP BY week_number
ORDER BY week_number;
```

This query calculates the week number based on the created_at date and counts the distinct user_ids for each week.

week_number	active_users	week_number	active_users	week_number	active_users
0	106	18	207	36	72
1	156	19	242	37	85
2	157	20	215	38	90
3	149	21	232	39	84
4	160	22	250	40	87
5	181	23	246	41	73
6	173	24	274	42	99
7	167	25	264	43	89
8	163	26	257	44	96
9	176	27	274	45	91
10	186	28	287	46	88
11	161	29	288	47	102
12	181	30	305	48	97
13	206	31	260	49	116
14	197	32	316	50	124
15	207	33	334	51	102
16	225	34	337	52	47
17	219	35	81		

Weekly User Engagement



Weekly User Engagement

Interpretation

The results show how many users were active each week.

Insight

I can see if user activity is going up, down, or staying the same over time. This helps me understand how engaging our platform is and if our efforts to attract users are working.

SQL Query:

```
SELECT AVG(active_users) AS
avg_active_usersFROM (SELECT
WEEK(created_at)
AS week_number,COUNT(DISTINCT
user_id) AS active_users
FROM users GROUP BY week_number)
AS weekly_engagement;
```

avg_active_users
177.0000

User Growth Analysis

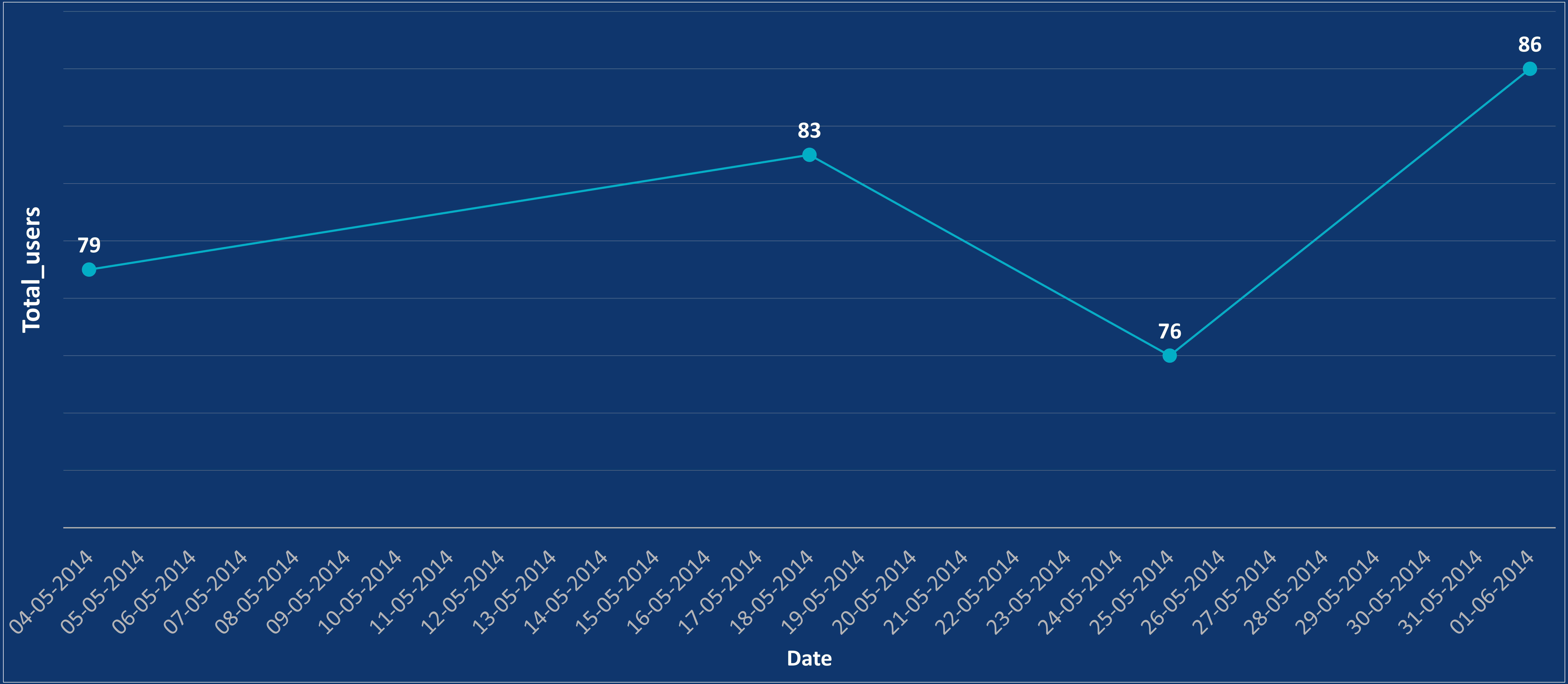
```
SELECT DATE(occurred_at)
AS date,COUNT(DISTINCT user_id)
AS total_users FROM events
GROUP BY DATE(occurred_at)
ORDER BY DATE(occurred_at);
```

This query calculates the total number of distinct users for each date based on the events table.

date	total_users
2014-05-01	293
2014-05-02	358
2014-05-03	145
2014-05-04	79
2014-05-05	257
2014-05-06	310
2014-05-07	323
2014-05-08	312
2014-05-09	352
2014-05-10	135
2014-05-11	92
2014-05-12	259
2014-05-13	313
2014-05-14	300

123 row(s) returned

User Growth Analysis



User Growth Analysis

Interpretation

- I identify the trends and spikes in user growth, which can help in understanding the impact of marketing campaigns or product updates.
- This analysis can assist in making informed decisions to further accelerate user growth.

Insight

The user count in 2014 shows a gradual increase over time, with the highest number of users recorded on July 18th (455 users) and the lowest on May 25th (76 users). This indicates a growth trend in user numbers, peaking in mid-July and experiencing a gradual decline thereafter.

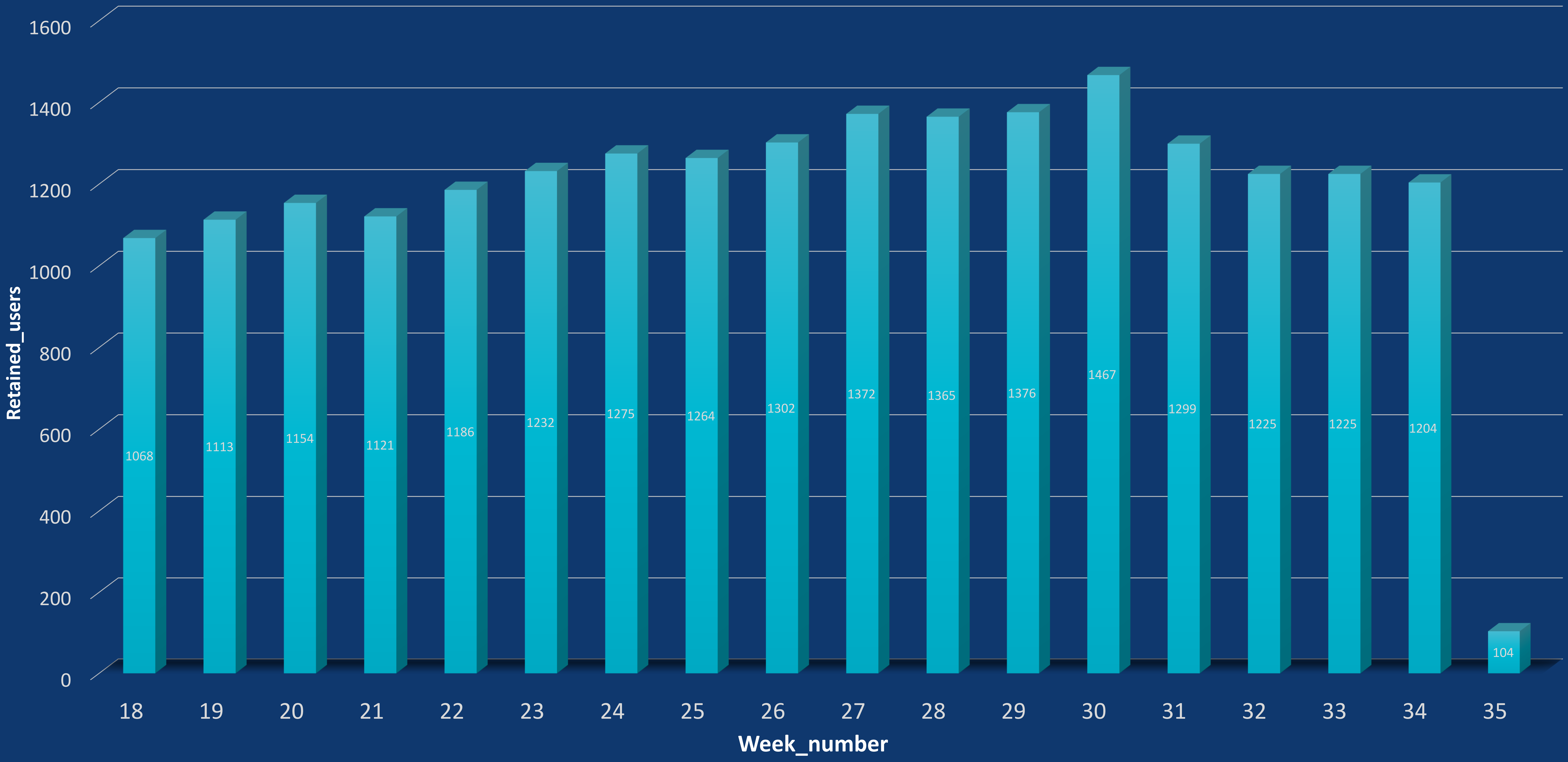
Weekly Retention Analysis

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

This query calculates the number of retained users per week based on their sign-up cohort.

week_number	retained_users	week_number	retained_users
17	663	25	1264
18	1068	26	1302
19	1113	27	1372
20	1154	28	1365
21	1121	29	1376
22	1186	30	1467
23	1232	31	1299
24	1275	32	1225

Weekly Retention Analysis



Weekly Retention Analysis

Interpretation:

The fluctuation in retained users per week suggests varying levels of user engagement or retention strategies. Week 30's peak could indicate successful initiatives or product updates, while the drop in week 17 might indicate a need for improvement in retaining users.

Insight

The number of retained users per week varies, with a peak of 1467 users in week 30 and a low of 663 users in week 17.

Weekly Engagement Per Device

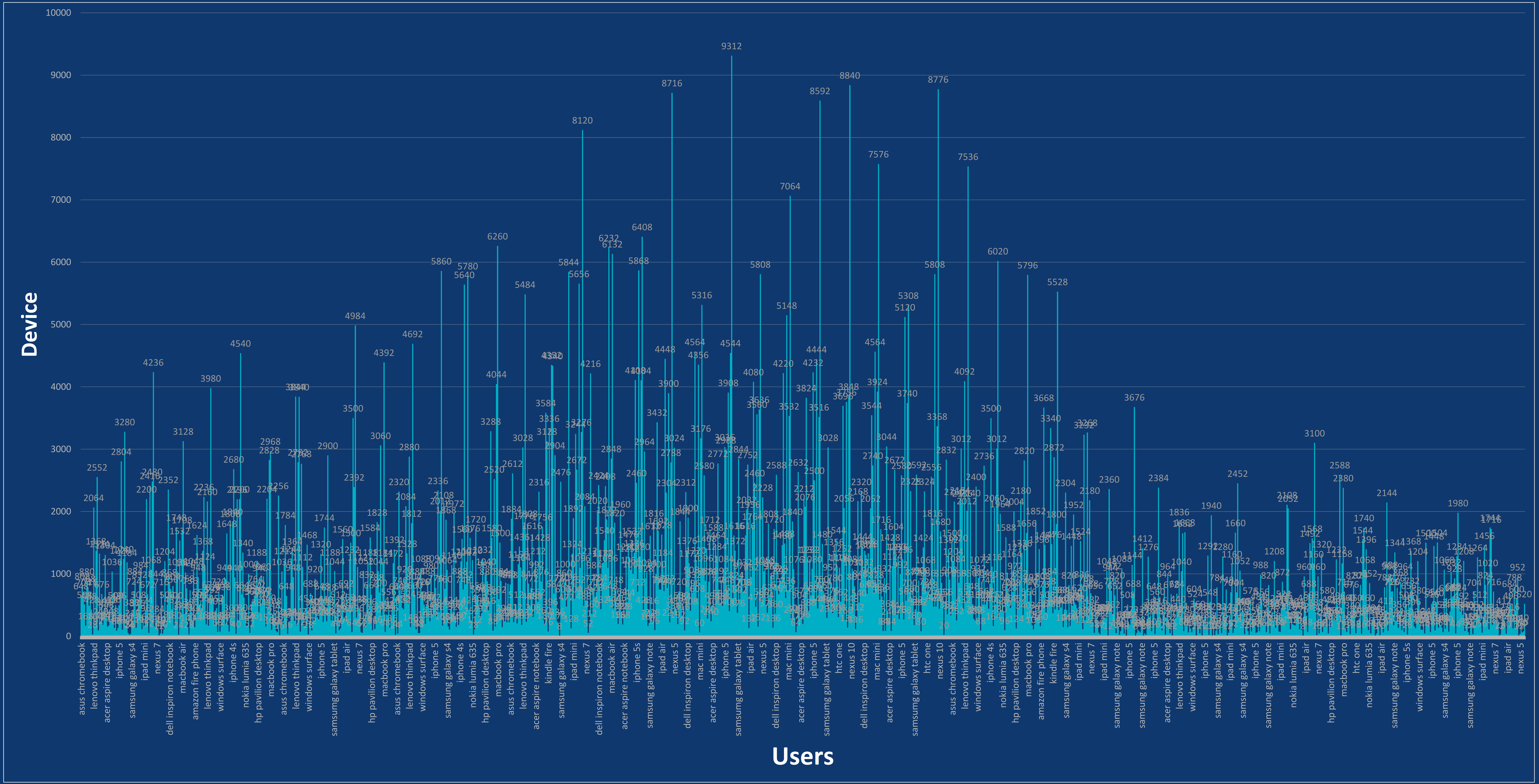
```
select weekofyear(u.created_at)
as weekly , e.device ,
count(u.user_id) as usersfrom
events e
right join users u
on
e.user_id = u.user_idwhere
e.event_type = 'engagement'group
by 1,2order by 1;
```

This query calculates the number of distinct active users per week per device based on the events table.

weekly	device	users
1	acer aspire notebook	220
1	asus chromebook	648
1	dell inspiron desktop	808
1	dell inspiron notebook	704
1	hp pavilion desktop	508
1	htc one	168
1	ipad air	880
1	ipad mini	768
1	iphone 4s	488
1	iphone 5	716
1	iphone 5s	428
1	kindle fire	28
1	lenovo thinkpad	2064
1	mac mini	164

1260 row(s) returned

Weekly Engagement Per Device



Weekly Retention Analysis

Insight

- **Popular Devices**: iPhone 5, MacBook Pro, and iPad Air consistently have high user engagement over the weeks.
- **Varied Engagement:** Devices like Kindle Fire and Nokia Lumia 635 show fluctuating user engagement.
- **Seasonal Trends:** Some devices, like Acer Aspire Notebook, have peaks and dips, possibly due to seasonal factors or product releases.
- **Consistent Growth:** Nexus 5 and Lenovo ThinkPad show a steady increase in user engagement over time.

Interpretation

The data provides insights into user engagement trends with different devices over time. It can help understand user preferences, identify popular devices, and track changes in user behavior. The analysis can inform decision-making regarding product development, marketing strategies, and customer support efforts.

Email Engagement Analysis

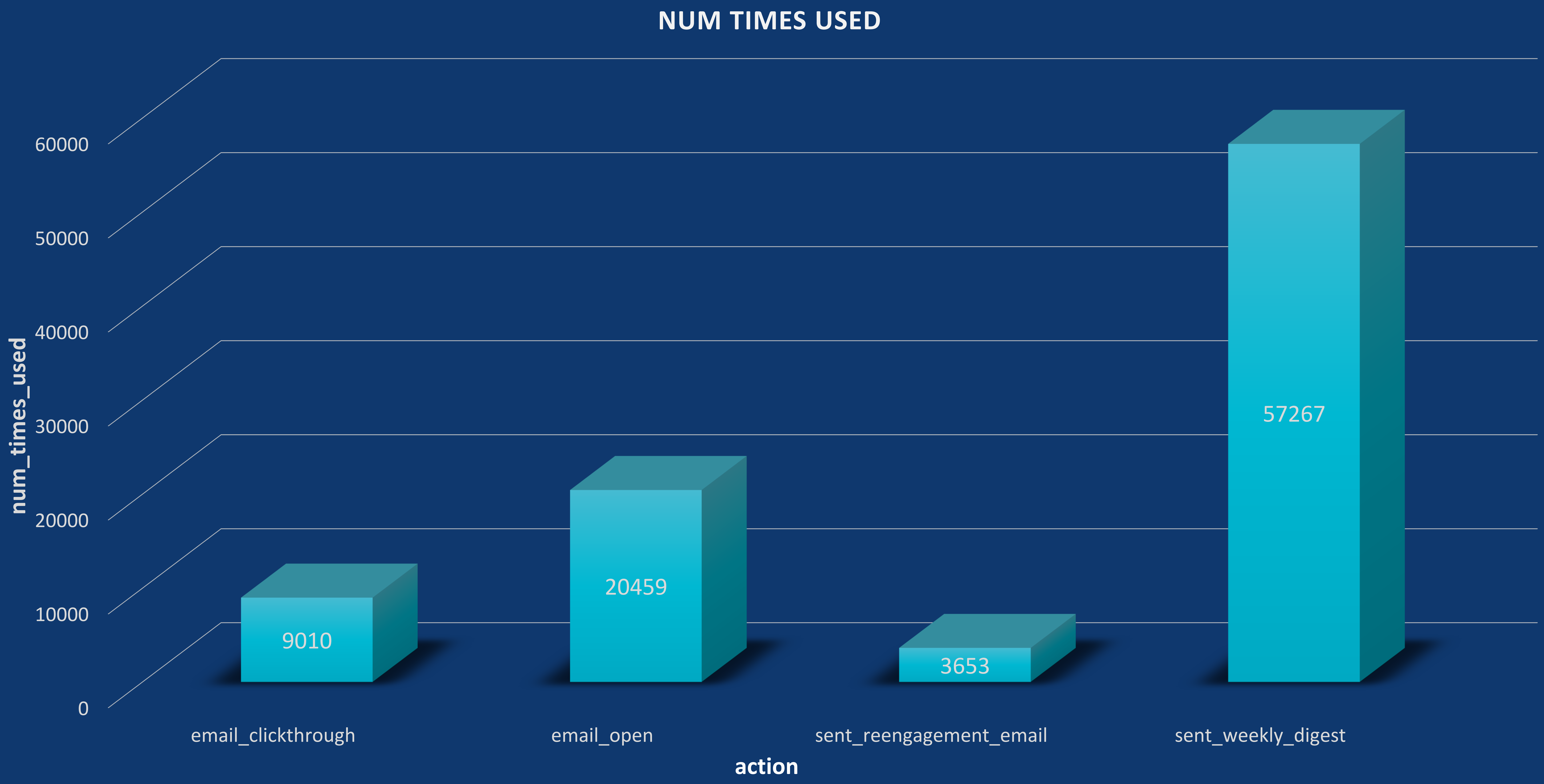
```
SELECT
    u.user_id,
    COUNT(CASE WHEN ee.action = 'sent_weekly_digest' THEN 1 ELSE NULL END) AS
num_sent_weekly_digest,
    COUNT(CASE WHEN ee.action = 'email_open' THEN 1 ELSE NULL END) AS num_email_open,
    COUNT(CASE WHEN ee.action = 'email_clickthrough' THEN 1 ELSE NULL END) AS
num_email_clickthrough,
    COUNT(CASE WHEN ee.action = 'sent_reengagement_email' THEN 1 ELSE NULL END) AS
num_sent_reengagement_email,
    AVG(CASE
            WHEN ee.action = 'sent_weekly_digest' THEN 1
            WHEN ee.action = 'email_open' THEN 1
            WHEN ee.action = 'email_clickthrough' THEN 1
            WHEN ee.action = 'sent_reengagement_email' THEN 1
            ELSE 0
        END) AS avg_email_engagement
FROM
    users u
LEFT JOIN
    email_events ee ON u.user_id = ee.user_id
GROUP BY
    u.user_id;

WITH q1 AS (
    SELECT
        action,
        TIMESTAMPDIFF(WEEK, '2013-01-01 04:40:10', occurred_at) AS wk,
        COUNT(user_id) AS Cnt
    FROM
        email_events
    GROUP BY
        action,
        wk
)
SELECT
    q1.action,
    COUNT(ee.user_id) AS num_times_used,
    ROUND(AVG(q1.Cnt), 2) AS avg_email_engagement
FROM
    q1
LEFT JOIN
    email_events ee ON q1.action = ee.action AND TIMESTAMPDIFF(WEEK, '2013-01-01 04:40:10',
ee.occurred_at) = q1.wk
GROUP BY
    q1.action
ORDER BY
    avg_email_engagement DESC;
```

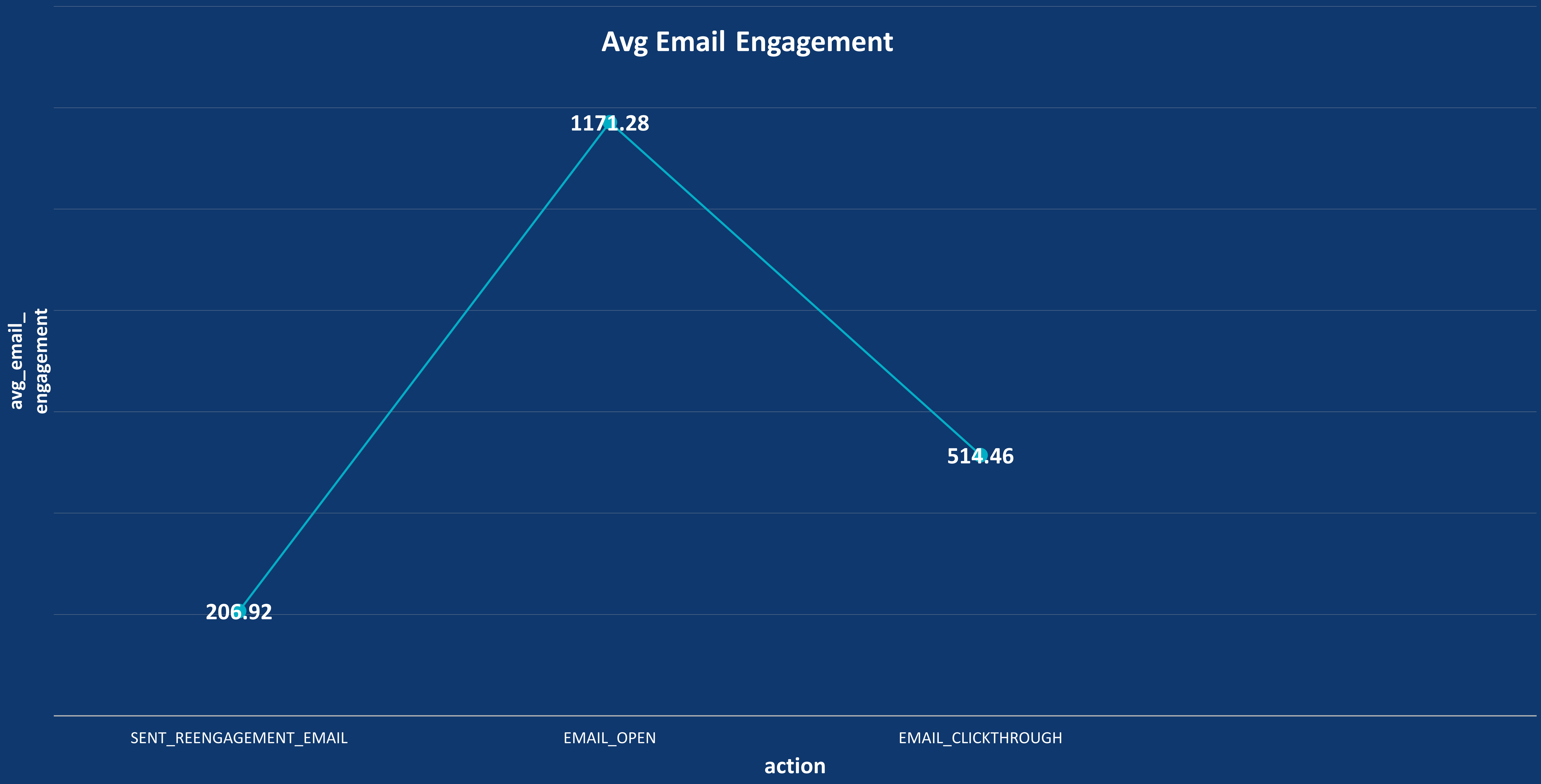
action	num_times_used	avg_email_engagement
email_clickthrough	9010	514.46
email_open	20459	1171.28
sent_reengagement_email	3653	206.92
sent_weekly_digest	57267	3281.11

This query calculates the average number of email engagements per week for each action based on the events in the email_events table.

Email Engagement Analysis



Email Engagement Analysis



Conclusion

In this project, I conducted operational analytics and investigated metric spikes using advanced SQL skills. I analyzed job data to calculate the number of jobs reviewed per hour, the 7-day rolling average of throughput, and the percentage share of each language over the last 30 days. Additionally, I identified duplicate rows in the data.

For the investigation of metric spikes, I analyzed user engagement on a weekly basis, user growth over time, weekly user retention, weekly engagement per device, and email engagement metrics. These analyses provided valuable insights for improving the company's operations and understanding sudden changes in key metrics.

Through this project, I gained a deeper understanding of operational analytics and how to use SQL to derive insights from data. I also improved my skills in data analysis, query writing, and interpretation of results. Overall, this project has contributed significantly to my analytical capabilities and decision-making processes.

Thank You

By Ayush Mahanta