# Operation Analytics and Investigating Metric Spike

**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, you'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, you'll need to answer these questions daily, making it crucial to understand how to investigate these metric spikes.

In this project, you'll take on the role of a Lead Data Analyst at a company like Microsoft. You'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. Your goal is to use your 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.

**Case Study 1: Job Data Analysis**

**You will be working with a table named job_data with the following columns:**

- **job_id:** Unique identifier of jobs
- **actor_id:** Unique identifier of actor
- **event:** The type of event (decision/skip/transfer).
- **language:** The Language of the content
- **time_spent:** Time spent to review the job in seconds.
- **org:** The Organization of the actor
- **ds:** The date in the format yyyy/mm/dd (stored as text).

**Tasks:**

A. **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.

B. **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.

C. **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.

D. **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.

**Case Study 2: Investigating Metric Spike**

**You will be working with three tables:**

- **users**: Contains one row per user, with descriptive information about that user's account.
- **events**: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).
- **email_events**: Contains events specific to the sending of emails.

**Tasks:**

A. **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.

B. **User Growth Analysis:**

- Objective: Analyze the growth of users over time for a product.
- Your Task: Write an SQL query to calculate the user growth for the product.

C. **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.

D. **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.

E. **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.

# Solution:

To complete this project, I analysed the provided tables and data. The project includes two case studies: one with a single table and another with three interconnected tables. The data is clear and accurate, with verified sources to ensure reliability.

The project's tech-stack includes:

- SQL for querying client-requested metrics

- MySQL Workbench as the database technology, chosen for its simplicity and advantages such as:

    - High security
    - Multiple storage engines
    - High efficiency

- ## Case Study 1: Job Data Analysis

    **A. Jobs Reviewed Over Time:**

- **Objective:** Calculate jobs reviewed per hour for each day in November 2020.

- **Approach:**

    1. Filter data for November 2020 using ds.
    2. Group by ds (day) and hour extracted from ds.
    3. Count the number of jobs (job_id) per hour.

```
#Calculate jobs reviewed per hour for each day in November 2020.
select * from job_data;

select ds, count(job_id) as Nos_of_jobs, sum(time_spent)/3600 as hours
from job_data
group by ds
order by ds;
```

| ds | Nos_of_jobs | hours |
|---|---|---|
| 2020-11-25 00:00:00 | 1 | 0.0125 |
| 2020-11-26 00:00:00 | 1 | 0.0156 |
| 2020-11-27 00:00:00 | 1 | 0.0289 |
| 2020-11-28 00:00:00 | 2 | 0.0092 |
| 2020-11-29 00:00:00 | 1 | 0.0056 |
| 2020-11-30 00:00:00 | 2 | 0.0111 |

B. **Throughput Analysis:** The rolling average is more useful than the daily metric because it smooths out short-term fluctuations and provides a clearer trend over time.

```sql
#Calculate the 7-day rolling average of throughput (number of events per second).
select * from events;
SELECT
    ds,
    COUNT(job_id) AS daily_throughput,
    AVG(COUNT(job_id)) OVER
    (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_avg_throughpu
    FROM  job_data
GROUP BY ds
ORDER BY ds;
```

| ds | daily_throughput | rolling_avg_throughput |
|---|---|---|
| 2020-11-25 00:00:00 | 1 | 1.0000 |
| 2020-11-26 00:00:00 | 1 | 1.0000 |
| 2020-11-27 00:00:00 | 1 | 1.0000 |
| 2020-11-28 00:00:00 | 2 | 1.2500 |
| 2020-11-29 00:00:00 | 1 | 1.2000 |
| 2020-11-30 00:00:00 | 2 | 1.3333 |

C. **Language Share Analysis:** This task requires calculating the percentage distribution of languages over a 30-day period. The results show that Persian dominates with a 37.5% share, surpassing all other languages.

```sql
#Calculate the percentage share of each language in the last 30 days.
select language, round(cnt*100/sum(cnt)over(),2) as percentage_share
from
(select language, count(language) as cnt
from job_data
group by language)d
group by language;
```

| language | percentage_share |
|---|---|
| English | 12.50 |
| Arabic | 12.50 |
| Persian | 37.50 |
| Hindi | 12.50 |
| French | 12.50 |
| Italian | 12.50 |

D. **Duplicate Rows Detection:** This task aims to identify and report duplicate records in the table. To do this, three scenarios were considered:

1. Both job_id and actor_id as primary keys: No duplicates found.
2. job_id as primary key: One duplicate record detected.
3. actor_id as primary key: One duplicate record detected.

These findings indicate that duplicates exist under certain assumptions.

```sql
#By asuming both job_id and actor_id as primary key:
select job_id, actor_id
from job_data
group by job_id, actor_id
having count(*)>1;
```

Result Grid | Filter

| | job_id | actor_id |
|---|---|---|

```sql
#By asuming job_id as primary key:
select job_id, count(job_id) as dup_job_id
from job_data
group by job_id
having dup_job_id>1;
```

| | job_id | dup_job_id |
|---|---|---|
| ▶ | 23 | 3 |

```sql
#By asuming actor_id as primary key:
select actor_id, count(actor_id) as dup_actor_id
from job_data
group by actor_id
having dup_actor_id >1;
```

| actor_id | dup_actor_id |
|---|---|
| 1003 | 2 |

- **Case Study 2: Investigating Metric Spike**

A. **Weekly User Engagement:** The goal of this task is to measure the activeness of users on a weekly basis. The SQL query calculates two key metrics:

- Weekly Active Users: The number of unique users who performed any action (event) in a given week.
- Average Daily Active Users: The average number of unique users active per day in a week, calculated by dividing the weekly active users by 7.

```sql
#Weekly user engagement
select
concat(year(occurred_at), '-W', LPAD(week(occurred_at),2,'0')) as week,
count(distinct user_id) as active_users,
count(distinct user_id)/7 as Avg_daily_active_users
from events
group by  week
order by active_users desc;
```

| week | active_users | Avg_daily_active_users |
|---|---|---|
| 2014-W30 | 1467 | 209.5714 |
| 2014-W29 | 1376 | 196.5714 |
| 2014-W27 | 1372 | 196.0000 |
| 2014-W28 | 1365 | 195.0000 |
| 2014-W26 | 1302 | 186.0000 |
| 2014-W31 | 1299 | 185.5714 |
| 2014-W24 | 1275 | 182.1429 |
| 2014-W25 | 1264 | 180.5714 |
| 2014-W23 | 1232 | 176.0000 |
| 2014-W32 | 1225 | 175.0000 |
| 2014-W33 | 1225 | 175.0000 |
| 2014-W34 | 1204 | 172.0000 |
| 2014-W22 | 1186 | 169.4286 |
| 2014-W20 | 1154 | 164.8571 |
| 2014-W21 | 1121 | 160.1429 |
| 2014-W19 | 1113 | 159.0000 |
| 2014-W18 | 1068 | 152.5714 |
| 2014-W17 | 663 | 94.7143 |
| 2014-W35 | 104 | 14.8571 |

B. **User Growth Analysis:** The goal is to analyse and track the growth of users over time for a product.
This query analyses user growth by calculating monthly signups, cumulative users, and growth rates. It groups users by signup month and provides insights into user acquisition trends, overall growth, and rate changes over time, showing how the user base is growing and changing each month.

```
#User Growth Analysis

WITH monthly_users AS (
    SELECT
        DATE_FORMAT(activated_at, '%Y-%m') AS signup_month,
        COUNT(user_id) AS new_users
    FROM users
    GROUP BY signup_month
)
SELECT
    signup_month,
    new_users,
    SUM(new_users) OVER (ORDER BY signup_month) AS cumulative_users,
    ROUND(
        (new_users - LAG(new_users) OVER (ORDER BY signup_month))
        / NULLIF(LAG(new_users) OVER (ORDER BY signup_month), 0) * 100, 2
    ) AS growth_rate
FROM monthly_users
ORDER BY signup_month;
```

| signup_month | new_users | cumulative_users | growth_rate |
|---|---|---|---|
| 2013-01 | 160 | 160 | NULL |
| 2013-02 | 160 | 320 | 0.00 |
| 2013-03 | 150 | 470 | -6.25 |
| 2013-04 | 181 | 651 | 20.67 |
| 2013-05 | 214 | 865 | 18.23 |
| 2013-06 | 213 | 1078 | -0.47 |
| 2013-07 | 284 | 1362 | 33.33 |
| 2013-08 | 316 | 1678 | 11.27 |
| 2013-09 | 330 | 2008 | 4.43 |
| 2013-10 | 390 | 2398 | 18.18 |
| 2013-11 | 399 | 2797 | 2.31 |
| 2013-12 | 486 | 3283 | 21.80 |
| 2014-01 | 552 | 3835 | 13.58 |
| 2014-02 | 525 | 4360 | -4.89 |
| 2014-03 | 615 | 4975 | 17.14 |
| 2014-04 | 726 | 5701 | 18.05 |
| 2014-05 | 779 | 6480 | 7.30 |
| 2014-06 | 873 | 7353 | 12.07 |
| 2014-07 | 997 | 8350 | 14.20 |
| 2014-08 | 1031 | 9381 | 3.41 |

C. **Weekly Retention Analysis:** The goal of Weekly Retention Analysis measures user retention over time by tracking activity in the weeks following sign-up. It calculates the percentage of active users in each cohort, helping identify retention trends and improve engagement strategies.
This query analyses weekly retention by tracking user activity from their first login. It shows how many users remain active over 17 weeks, grouped by their first login week, providing insights into user retention over time.

```sql
#   Weekly Retention Analysis:
select occurred_at from events;
WITH formatted_events AS (
    SELECT
        user_id,
        occurred_at
    FROM events
    WHERE event_type = 'engagement'
),
user_first_week AS (
    SELECT
        user_id,
        MIN(EXTRACT(WEEK FROM occurred_at)) AS first_week
    FROM formatted_events
    GROUP BY user_id
),
user_activity_weeks AS (
    SELECT
        fe.user_id,
        EXTRACT(WEEK FROM fe.occurred_at) AS activity_week,
        ufw.first_week,
        (EXTRACT(WEEK FROM fe.occurred_at) - ufw.first_week) AS weeks_since_first
    FROM formatted_events fe
    JOIN user_first_week ufw
        ON fe.user_id = ufw.user_id
)
```

```sql
SELECT
    first_week AS first_login_week,
    COUNT(CASE WHEN weeks_since_first = 0 THEN 1 END) AS week_0,
    COUNT(CASE WHEN weeks_since_first = 1 THEN 1 END) AS week_1,
    COUNT(CASE WHEN weeks_since_first = 2 THEN 1 END) AS week_2,
    COUNT(CASE WHEN weeks_since_first = 3 THEN 1 END) AS week_3,
    COUNT(CASE WHEN weeks_since_first = 4 THEN 1 END) AS week_4,
    COUNT(CASE WHEN weeks_since_first = 5 THEN 1 END) AS week_5,
    COUNT(CASE WHEN weeks_since_first = 6 THEN 1 END) AS week_6,
    COUNT(CASE WHEN weeks_since_first = 7 THEN 1 END) AS week_7,
    COUNT(CASE WHEN weeks_since_first = 8 THEN 1 END) AS week_8,
    COUNT(CASE WHEN weeks_since_first = 9 THEN 1 END) AS week_9,
    COUNT(CASE WHEN weeks_since_first = 10 THEN 1 END) AS week_10,
    COUNT(CASE WHEN weeks_since_first = 11 THEN 1 END) AS week_11,
    COUNT(CASE WHEN weeks_since_first = 12 THEN 1 END) AS week_12,
    COUNT(CASE WHEN weeks_since_first = 13 THEN 1 END) AS week_13,
    COUNT(CASE WHEN weeks_since_first = 14 THEN 1 END) AS week_14,
    COUNT(CASE WHEN weeks_since_first = 15 THEN 1 END) AS week_15,
    COUNT(CASE WHEN weeks_since_first = 16 THEN 1 END) AS week_16,
    COUNT(CASE WHEN weeks_since_first = 17 THEN 1 END) AS week_17
FROM user_activity_weeks
GROUP BY first_week
ORDER BY first_week;
```

| first_login_week | week_0 | week_1 | week_2 | week_3 | week_4 | week_5 | week_6 | week_7 | week_8 | week_9 | week_10 | week_11 | week_12 | week_13 | week_14 | week_15 | week_16 | week_17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | 8019 | 9291 | 6100 | 4890 | 3910 | 3376 | 2900 | 2782 | 2692 | 2743 | 2390 | 2395 | 2391 | 2574 | 2044 | 1464 | 1474 | 1527 |
| 18 | 8050 | 5952 | 4598 | 3391 | 2719 | 2223 | 2329 | 2008 | 1861 | 1823 | 1836 | 1916 | 2093 | 1954 | 1400 | 1412 | 1135 | 37 |
| 19 | 5172 | 4326 | 2756 | 2303 | 1905 | 1658 | 1355 | 1484 | 1409 | 1483 | 1158 | 917 | 826 | 669 | 785 | 805 | 23 | 0 |
| 20 | 4097 | 3340 | 2840 | 1884 | 1413 | 1252 | 953 | 835 | 1029 | 989 | 876 | 549 | 441 | 463 | 450 | 0 | 0 | 0 |
| 21 | 3754 | 3141 | 2213 | 1342 | 1145 | 963 | 955 | 1275 | 822 | 804 | 616 | 567 | 555 | 441 | 17 | 0 | 0 | 0 |
| 22 | 4034 | 3378 | 2506 | 1642 | 1263 | 942 | 1001 | 901 | 851 | 694 | 791 | 679 | 467 | 8 | 0 | 0 | 0 | 0 |
| 23 | 3777 | 3369 | 2064 | 1522 | 1143 | 1041 | 1065 | 824 | 799 | 674 | 412 | 410 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 3653 | 3252 | 2067 | 1613 | 1289 | 911 | 1143 | 913 | 678 | 430 | 384 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 3232 | 3087 | 2280 | 1639 | 1212 | 816 | 577 | 723 | 536 | 537 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | 3118 | 3029 | 1552 | 1296 | 933 | 714 | 662 | 548 | 345 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 3462 | 3195 | 1693 | 1553 | 1057 | 806 | 558 | 560 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | 3041 | 3026 | 1854 | 873 | 564 | 388 | 323 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | 2687 | 3111 | 1688 | 966 | 619 | 639 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 3184 | 2995 | 1727 | 1124 | 696 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | 2257 | 2112 | 906 | 777 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | 2368 | 2304 | 1085 | 67 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | 2952 | 2727 | 63 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 34 | 2819 | 326 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 35 | 127 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

D. **Weekly Engagement Per Device:** The goal is to calculate the number of unique active users per device on a weekly basis to measure user engagement trends.

The query extracts weekly active users per device for weeks W17 to W35 in 2014. First, it formats occurred_at into YYYY-WW format and counts unique users per device. Then, it pivots data using CASE WHEN conditions to create separate columns for each week. Finally, it aggregates values, grouping by device for easy analysis.

```sql
#Weekly Engagement Per Device
SELECT device,
    SUM(CASE WHEN week = '2014-W17' THEN active_users ELSE 0 END) AS week_2014_W17,
    SUM(CASE WHEN week = '2014-W18' THEN active_users ELSE 0 END) AS week_2014_W18,
    SUM(CASE WHEN week = '2014-W19' THEN active_users ELSE 0 END) AS week_2014_W19,
    SUM(CASE WHEN week = '2014-W20' THEN active_users ELSE 0 END) AS week_2014_W20,
    SUM(CASE WHEN week = '2014-W21' THEN active_users ELSE 0 END) AS week_2014_W21,
    SUM(CASE WHEN week = '2014-W22' THEN active_users ELSE 0 END) AS week_2014_W22,
    SUM(CASE WHEN week = '2014-W23' THEN active_users ELSE 0 END) AS week_2014_W23,
    SUM(CASE WHEN week = '2014-W24' THEN active_users ELSE 0 END) AS week_2014_W24,
    SUM(CASE WHEN week = '2014-W25' THEN active_users ELSE 0 END) AS week_2014_W25,
    SUM(CASE WHEN week = '2014-W26' THEN active_users ELSE 0 END) AS week_2014_W26,
    SUM(CASE WHEN week = '2014-W27' THEN active_users ELSE 0 END) AS week_2014_W27,
    SUM(CASE WHEN week = '2014-W28' THEN active_users ELSE 0 END) AS week_2014_W28,
    SUM(CASE WHEN week = '2014-W29' THEN active_users ELSE 0 END) AS week_2014_W29,
    SUM(CASE WHEN week = '2014-W30' THEN active_users ELSE 0 END) AS week_2014_W30,
    SUM(CASE WHEN week = '2014-W31' THEN active_users ELSE 0 END) AS week_2014_W31,
    SUM(CASE WHEN week = '2014-W32' THEN active_users ELSE 0 END) AS week_2014_W32,
    SUM(CASE WHEN week = '2014-W33' THEN active_users ELSE 0 END) AS week_2014_W33,
    SUM(CASE WHEN week = '2014-W34' THEN active_users ELSE 0 END) AS week_2014_W34,
    SUM(CASE WHEN week = '2014-W35' THEN active_users ELSE 0 END) AS week_2014_W35
FROM ( SELECT device,
        CONCAT(YEAR(occurred_at), '-W', LPAD(WEEK(occurred_at, 2), 2, '0')) AS week,
        COUNT(DISTINCT user_id) AS active_users
    FROM events
    WHERE event_type = 'engagement'
    GROUP BY device, week
) AS weekly_engagement
GROUP BY device
ORDER BY device;
```

| device | week_2014_W17 | week_2014_W18 | week_2014_W19 | week_2014_W20 | week_2014_W21 | week_2014_W22 | week_2014_W23 | week_2014_W24 | week_2014_W25 | week_2014_W26 | week_2014_W: |
|---|---|---|---|---|---|---|---|---|---|---|---|
| acer aspire desktop | 9 | 26 | 23 | 23 | 29 | 25 | 22 | 24 | 28 | 29 | 29 |
| acer aspire notebook | 20 | 33 | 41 | 40 | 47 | 41 | 43 | 40 | 47 | 35 | 49 |
| amazon fire phone | 4 | 9 | 12 | 11 | 5 | 5 | 16 | 11 | 13 | 13 | 10 |
| asus chromebook | 21 | 42 | 27 | 41 | 38 | 52 | 49 | 43 | 38 | 49 | 52 |
| dell inspiron desktop | 18 | 58 | 36 | 52 | 41 | 52 | 53 | 59 | 52 | 60 | 53 |
| dell inspiron notebook | 46 | 77 | 83 | 84 | 80 | 92 | 103 | 99 | 105 | 89 | 89 |
| hp pavilion desktop | 14 | 37 | 40 | 30 | 44 | 38 | 54 | 56 | 52 | 46 | 56 |
| htc one | 16 | 19 | 30 | 29 | 21 | 24 | 20 | 20 | 21 | 23 | 27 |
| ipad air | 27 | 52 | 55 | 59 | 51 | 58 | 41 | 57 | 57 | 56 | 55 |
| ipad mini | 19 | 30 | 36 | 32 | 23 | 34 | 33 | 39 | 30 | 43 | 35 |
| iphone 4s | 21 | 46 | 44 | 55 | 45 | 45 | 53 | 53 | 40 | 50 | 67 |
| iphone 5 | 65 | 113 | 115 | 125 | 137 | 125 | 152 | 142 | 137 | 152 | 163 |
| iphone 5s | 42 | 73 | 79 | 79 | 74 | 71 | 79 | 79 | 78 | 94 | 83 |
| kindle fire | 6 | 27 | 21 | 23 | 30 | 21 | 25 | 25 | 24 | 26 | 25 |
| lenovo thinkpad | 86 | 153 | 178 | 173 | 167 | 176 | 176 | 165 | 197 | 192 | 202 |
| mac mini | 6 | 13 | 18 | 26 | 18 | 25 | 18 | 29 | 21 | 11 | 15 |
| macbook air | 54 | 121 | 112 | 119 | 110 | 145 | 124 | 152 | 121 | 134 | 142 |
| macbook pro | 143 | 252 | 266 | 256 | 247 | 251 | 266 | 255 | 275 | 269 | 302 |
| nexus 10 | 16 | 30 | 25 | 22 | 25 | 27 | 45 | 38 | 29 | 29 | 37 |
| nexus 5 | 40 | 73 | 87 | 103 | 91 | 96 | 88 | 87 | 89 | 87 | 84 |
| nexus 7 | 18 | 30 | 41 | 32 | 29 | 45 | 36 | 49 | 51 | 46 | 40 |
| nokia lumia 635 | 17 | 33 | 23 | 22 | 25 | 25 | 31 | 35 | 37 | 42 | 31 |
| samsumg galaxy tablet | 8 | 11 | 6 | 9 | 6 | 10 | 14 | 11 | 12 | 12 | 15 |
| samsung galaxy note | 7 | 15 | 11 | 18 | 20 | 19 | 14 | 20 | 14 | 9 | 15 |

| 3 | week_2014_W24 | week_2014_W25 | week_2014_W26 | week_2014_W27 | week_2014_W28 | week_2014_W29 | week_2014_W30 | week_2014_W31 | week_2014_W32 | week_2014_W33 | week_2014_W34 | week_2014_W35 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 24 | 28 | 29 | 29 | 30 | 28 | 33 | 31 | 35 | 39 | 30 | 1 |
| | 40 | 47 | 35 | 49 | 49 | 53 | 60 | 55 | 55 | 46 | 63 | 3 |
| | 11 | 13 | 13 | 10 | 6 | 12 | 12 | 14 | 12 | 14 | 11 | 0 |
| | 43 | 38 | 49 | 52 | 50 | 49 | 56 | 56 | 62 | 49 | 47 | 6 |
| | 59 | 52 | 60 | 53 | 56 | 54 | 54 | 44 | 57 | 37 | 49 | 1 |
| | 99 | 105 | 89 | 89 | 103 | 113 | 127 | 113 | 104 | 110 | 105 | 9 |
| | 56 | 52 | 46 | 56 | 56 | 58 | 42 | 51 | 51 | 38 | 36 | 1 |
| | 20 | 21 | 23 | 27 | 26 | 31 | 31 | 13 | 18 | 19 | 25 | 2 |
| | 57 | 57 | 56 | 55 | 54 | 52 | 70 | 55 | 48 | 40 | 39 | 0 |
| | 39 | 30 | 43 | 35 | 35 | 34 | 35 | 27 | 30 | 28 | 25 | 2 |
| | 53 | 40 | 50 | 67 | 61 | 60 | 65 | 56 | 34 | 35 | 50 | 6 |
| | 142 | 137 | 152 | 163 | 151 | 144 | 152 | 135 | 119 | 110 | 101 | 2 |
| | 79 | 78 | 94 | 83 | 93 | 90 | 103 | 71 | 67 | 65 | 70 | 3 |
| | 25 | 24 | 26 | 25 | 31 | 37 | 25 | 14 | 12 | 14 | 13 | 3 |
| | 165 | 197 | 192 | 202 | 220 | 209 | 206 | 207 | 179 | 191 | 193 | 16 |
| | 29 | 21 | 11 | 15 | 28 | 31 | 23 | 24 | 20 | 32 | 30 | 2 |
| | 152 | 121 | 134 | 142 | 148 | 148 | 159 | 147 | 125 | 133 | 136 | 10 |
| | 255 | 275 | 269 | 302 | 295 | 295 | 322 | 321 | 307 | 312 | 292 | 17 |
| | 38 | 29 | 29 | 37 | 26 | 25 | 36 | 24 | 30 | 23 | 25 | 2 |
| | 87 | 89 | 87 | 84 | 85 | 77 | 84 | 69 | 67 | 70 | 70 | 4 |
| | 49 | 51 | 46 | 40 | 39 | 45 | 62 | 38 | 25 | 30 | 33 | 2 |
| | 35 | 37 | 42 | 31 | 35 | 43 | 34 | 28 | 28 | 27 | 17 | 2 |
| | 11 | 12 | 12 | 15 | 9 | 13 | 9 | 8 | 6 | 12 | 14 | 0 |
| | 20 | 14 | 9 | 15 | 10 | 16 | 15 | 14 | 12 | 13 | 13 | 1 |

E. **Email Engagement Analysis:** The goal is to measure the average weekly frequency of different email engagement actions to understand user interaction trends over time.

This query measures weekly email engagement by tracking actions over time, calculating their average frequency, and ranking them to identify the most popular interactions.

```
#Email Engagement Analysis:
SELECT action,
    ROUND(AVG(d1.frequency), 2) AS avg_week
FROM (SELECT action,
        TIMESTAMPDIFF(WEEK, '2014-05-01 00:00:00', occurred_at) AS wk,
        COUNT(user_id) AS frequency
    FROM email_events
    GROUP BY action, wk
) d1
GROUP BY action
ORDER BY avg_week DESC;
```

| action | avg_week |
|---|---|
| sent_weekly_digest | 3181.50 |
| email_open | 1136.61 |
| email_clickthrough | 500.56 |
| sent_reengagement_email | 202.94 |

- ## **Result** :

  This project gave me valuable insights into **Operational Analytics** and **Metric Spike Investigation**. The findings can help businesses **optimize operations**, **understand user behavior**, and **improve engagement strategies**. By analysing trends in job data and user interactions, companies can make **data-driven decisions**, enhance **marketing efforts**, and refine **product development** to boost overall efficiency and user satisfaction.