# Operation Analytics and Investigating Metric Spike

## Project Description

Operation Analytics involves a comprehensive examination of a company's end-to-end operations to identify areas for improvement. Collaborating closely with various teams such as operations, support, and marketing, a Data Analyst in this role extracts valuable insights from collected data. This analysis is pivotal for predicting the overall trajectory of a company, be it growth or decline. It facilitates enhanced automation, fosters better collaboration among cross-functional teams, and streamlines workflows for increased efficiency.

A crucial aspect of Operation Analytics is the investigation of metric spikes. As a Data Analyst Lead, the responsibility extends to understanding and helping other teams comprehend fluctuations in metrics. Investigating these metric spikes is essential for informed decision-making and ensuring the company's sustained success. In this role at a company like Microsoft, the Data Analyst Lead utilizes various datasets and tables to derive insights and respond to inquiries from different departments.

## Approach

Tackled the challenge by taking many crucial steps in approaching the results. Started by carefully going over and comprehending the project's offered dataset. This requires determining the precise activities that need to be completed in addition to examining the tables and the corresponding data.

Use MySQL as a database management system to work with the dataset. Efficiently extract the required insights from the data by querying and manipulating it in this way.

Start by carefully going over the dataset and the tasks at hand before moving on to the step-by-step analysis of the data. To get the needed data and carry out the necessary computations, required using a variety of SQL queries, aggregations, filters, and joins.

Keep a careful eye on the results throughout the analysis process to make sure they match the anticipated outcomes. Take note of any patterns, trends, or anomalies seen in the data, which enable you to extrapolate pertinent conclusions and gain insightful knowledge.

Fulfill the tasks given and gain insightful knowledge from the dataset by carefully adhering to this strategy. To meet project objectives and produce precise and significant results, the procedure comprised a thorough review, analysis, and interpretation of the data.

## Tech-Stack Used

Prepare - Process - Analyze Data using MySQL Workbench, MS Excel and Word

## Insights

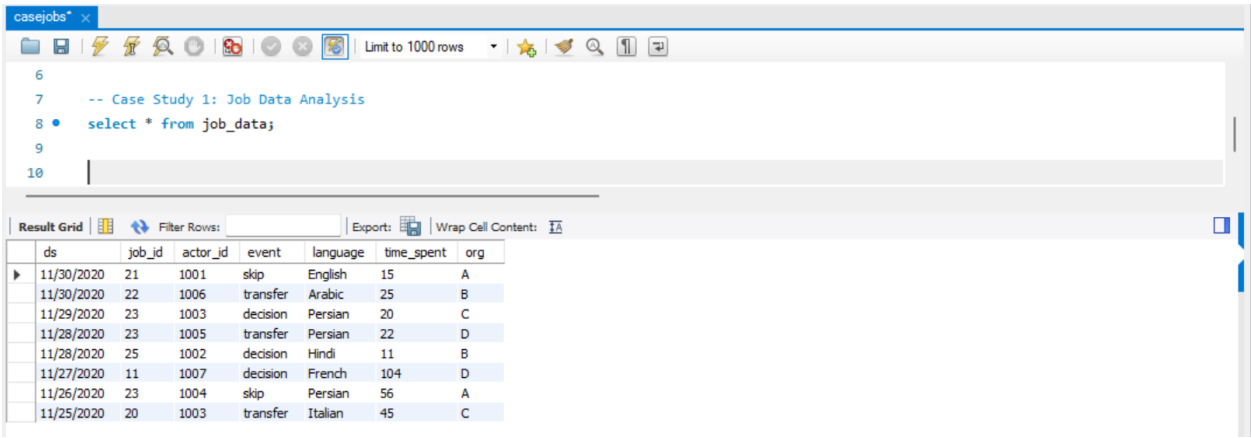
I became fully immersed in the principles of operation analysis during the project, which included a variety of essential abilities and methods. My initial focus was on putting these ideas into practice and learning as much as I could.

First, I thoroughly understood how to use SQL to do basic operations like counting, summing, calculating percentages, and working with dates and timings. I also gained practical knowledge by making tables and adding values to them. Additionally, I learned how to load files into SQL, which allowed me to easily deal with a variety of datasets and formats

I studied advanced SQL procedures to improve my analytical skills even further. I became proficient with the row number function and row counting. I investigated the potential of utilizing the group by function to aggregate data and the partition by clause to divide data. I also realized how crucial it is to give derived tables the proper aliases to enhance the readability and maintainability of the code.

To put it briefly, I became fully involved in the field of operation analysis from the beginning of the project and refined my knowledge of a variety of SQL ideas and methods.

# Case Study 1: Job Data Analysis




| ds | job_id | actor_id | event | language | time_spent | org |
|---|---|---|---|---|---|---|
| 11/30/2020 | 21 | 1001 | skip | English | 15 | A |
| 11/30/2020 | 22 | 1006 | transfer | Arabic | 25 | B |
| 11/29/2020 | 23 | 1003 | decision | Persian | 20 | C |
| 11/28/2020 | 23 | 1005 | transfer | Persian | 22 | D |
| 11/28/2020 | 25 | 1002 | decision | Hindi | 11 | B |
| 11/27/2020 | 11 | 1007 | decision | French | 104 | D |
| 11/26/2020 | 23 | 1004 | skip | Persian | 56 | A |
| 11/25/2020 | 20 | 1003 | transfer | Italian | 45 | C |

## Tasks:

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.

     **Code**

     *SELECT DS AS DATE, COUNT(DISTINCT JOB_ID) AS JOB_REVIEWED, SUM(TIME_SPENT)/(60 * 60) AS PER_HOUR_TIME_SPEND, ROUND((COUNT(DISTINCT JOB_ID)/SUM(TIME_SPENT))*(60 * 60)) AS JOBS_REVIEWED_PER_HOUR_PER_DAY FROM JOB_DATA*

     *WHERE DS >= '01-11-2020' AND DS <= '30-11-2020'*

     *GROUP BY DS ORDER BY DS*

     *DESC;*

### Output

```sql
-- 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.

SELECT DS AS DATE, COUNT(DISTINCT JOB_ID) AS JOB_REVIEWED, SUM(TIME_SPENT)/(60 * 60) AS PER_HOUR_TIME_SPEND, ROUND((COUNT(DISTINCT JOB_ID
)/SUM(TIME_SPENT))*(60 * 60)) AS JOBS_REVIEWED_PER_HOUR_PER_DAY FROM JOB_DATA
WHERE DS >= '01-11-2020' AND DS <= '30-11-2020'
GROUP BY DS ORDER BY DS
DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| DATE | JOB_REVIEWED | PER_HOUR_TIME_SPEND | JOBS_REVIEWED_PER_HOUR_PER_DAY |
|---|---|---|---|
| 11/30/2020 | 2 | 0.0111 | 180 |
| 11/29/2020 | 1 | 0.0056 | 180 |
| 11/28/2020 | 2 | 0.0092 | 218 |
| 11/27/2020 | 1 | 0.0289 | 35 |
| 11/26/2020 | 1 | 0.0156 | 64 |
| 11/25/2020 | 1 | 0.0125 | 80 |

### Insights

The maximum number of jobs reviewed per hour per day for November 2020 was 218 on “ 28 November”.

The average number of jobs reviewed per hour per day for November 2020 was 35 on “27 November”.

The average number of jobs reviewed per hour per day for November 2020 is 126.

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.

   ---> A 7-day rolling metric offers a stabilized view by averaging data over a week, whereas daily metrics capture short-term fluctuations. The preference between a daily metric and a 7-day rolling metric for throughput depends on the need for precise

daily insights or smoothed patterns. Here I've a 7-day rolling average for the number of events happening.

**Code**

*WITH DAILY_METRIC AS ( SELECT DS, COUNT(JOB_ID) AS JOB_REVIEW FROM JOB_DATA GROUP BY DS)*

*SELECT DS, JOB_REVIEW, AVG(JOB_REVIEW) OVER (ORDER BY DS ROWS BETWEEN 6 PRECEDING AND CURRENT ROW ) AS THROUGHPUT*

*FROM DAILY_METRIC ORDER BY*

*THROUGHPUT DESC ;*

**Output**

```
-- 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.

WITH DAILY_METRIC AS ( SELECT DS, COUNT(JOB_ID) AS JOB_REVIEW FROM JOB_DATA GROUP BY DS)
SELECT DS, JOB_REVIEW, AVG(JOB_REVIEW) OVER (ORDER BY DS ROWS BETWEEN 6 PRECEDING AND CURRENT ROW ) AS THROUGHPUT
FROM DAILY_METRIC ORDER BY
THROUGHPUT DESC ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| DS | JOB_REVIEW | THROUGHPUT |
|---|---|---|
| 11/30/2020 | 2 | 1.3333 |
| 11/28/2020 | 2 | 1.2500 |
| 11/29/2020 | 1 | 1.2000 |
| 11/25/2020 | 1 | 1.0000 |
| 11/26/2020 | 1 | 1.0000 |
| 11/27/2020 | 1 | 1.0000 |

**Insights**

On November 30, 2020, there were 2 job reviews with a throughput of 1.3333.

On November 28, 2020, there were also 2 job reviews with a slightly lower throughput of 1.25.

Overall, the throughput values indicate variation in the efficiency of job reviews on different dates.

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.

**Code**

*SELECT LANGUAGE,COUNT(LANGUAGE) AS LANGUAGE_COUNT,*

*(COUNT(LANGUAGE)/(SELECT COUNT(*) FROM JOB_DATA)) * 100 AS Percentage_Share*

*FROM JOB_DATA GROUP BY LANGUAGE ORDER BY LANGUAGE DESC ;*

**Output**

```
23
24      -- Language Share Analysis:
25      -- Objective: Calculate the percentage share of each language in the last 30 days.
26      -- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.
27
28 •    SELECT LANGUAGE,COUNT(LANGUAGE) AS LANGUAGE_COUNT,
29      (COUNT(LANGUAGE)/(SELECT COUNT(*) FROM JOB_DATA)) * 100 AS Percentage_Share
30      FROM JOB_DATA GROUP BY LANGUAGE ORDER BY LANGUAGE DESC ;
31
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| LANGUAGE | LANGUAGE_COUNT | Percentage_Share |
|---|---|---|
| Persian | 3 | 37.5000 |
| Italian | 1 | 12.5000 |
| Hindi | 1 | 12.5000 |
| French | 1 | 12.5000 |
| English | 1 | 12.5000 |
| Arabic | 1 | 12.5000 |

**Insights**

The Persian language has the highest percentage share at 37.5%. Italian, Hindi, French, English, and Arabic have the same percentage share at 12.5%.

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.

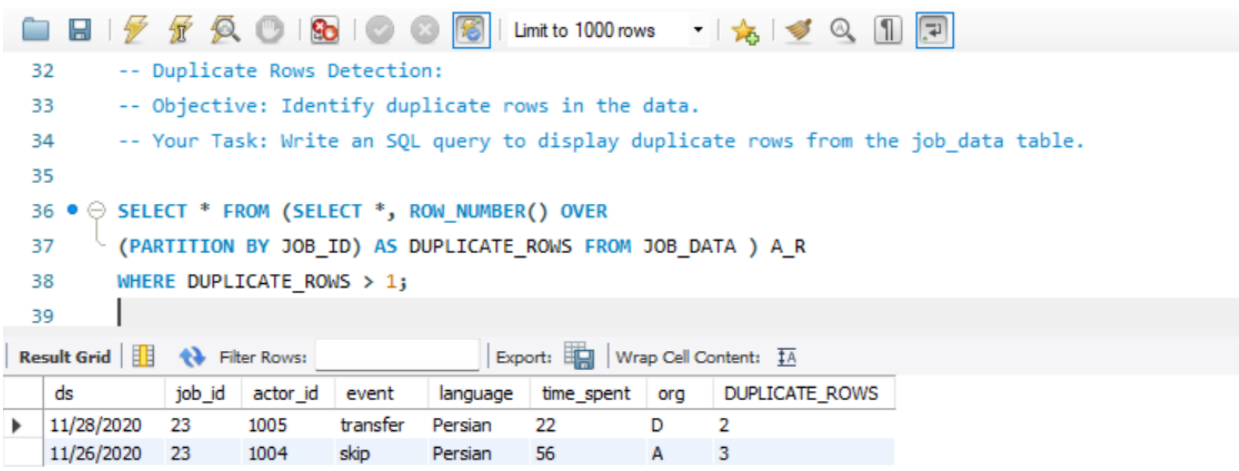
**Code**

*SELECT * FROM (SELECT *, ROW_NUMBER() OVER*

*(PARTITION BY JOB_ID) AS DUPLICATE_ROWS FROM JOB_DATA ) A_R*

*WHERE DUPLICATE_ROWS > 1;*

**Output**




| ds | job_id | actor_id | event | language | time_spent | org | DUPLICATE_ROWS |
|---|---|---|---|---|---|---|---|
| 11/28/2020 | 23 | 1005 | transfer | Persian | 22 | D | 2 |
| 11/26/2020 | 23 | 1004 | skip | Persian | 56 | A | 3 |

**Insights**

The above code creates a new column called DUPLICATE_ROWS using the ROW_NUMBER() function, which assigns unique numbers to rows within each group of JOB_ID in the JOB_DATA table.

The result is then given the alias A_R. OVER (PARTITION BY JOB_ID) defines how the rows are partitioned or grouped. In

this case, the rows are grouped based on the values in the JOB_ID column. A_R is an alias for the derived table created by the subquery. The alias A_R can be used to reference the result set in the outer query.

**Case Study 2: Investigating Metric Spike**

```
CREATE DATABASE Project3;

SHOW DATABASES;

USE Project3;


-- Table-1 users

create table users(
user_id int,
created_at varchar(100),
company_id int,
language varchar(50),
activated_at varchar(100),
state varchar(50));

SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.1/Uploads/Table-1 users.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY ""
IGNORE 1 ROWS;

select * from events;

SET SQL_SAFE_UPDATES=0;

ALTER TABLE users ADD COLUMN temp_created_at DATETIME;
```

*UPDATE users SET temp_created_at = STR_TO_DATE(created_at, '%d-%m-%Y %H:%i');*
*ALTER TABLE users DROP COLUMN created_at;*
*ALTER TABLE users CHANGE COLUMN temp_created_at created_at DATETIME;*

*ALTER TABLE users ADD COLUMN temp_activated_at DATETIME;*
*UPDATE users SET temp_activated_at = STR_TO_DATE(activated_at, '%d-%m-%Y %H:%i');*
*ALTER TABLE users DROP COLUMN activated_at;*
*ALTER TABLE users CHANGE COLUMN temp_activated_at activated_at DATETIME;*

*select * from users;*

Limit to 1000 rows

```
25 • select * from users;
26
27 • SET SQL_SAFE_UPDATES=0;
28
29 • ALTER TABLE users ADD COLUMN temp_created_at DATETIME;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| user_id | company_id | language | state | created_at | activated_at |
|---|---|---|---|---|---|
| 0 | 5737 | english | active | 2013-01-01 20:59:00 | 2013-01-01 21:01:00 |
| 3 | 2800 | german | active | 2013-01-01 18:40:00 | 2013-01-01 18:42:00 |
| 4 | 5110 | indian | active | 2013-01-01 14:37:00 | 2013-01-01 14:39:00 |
| 6 | 11699 | english | active | 2013-01-01 18:37:00 | 2013-01-01 18:38:00 |
| 7 | 4765 | french | active | 2013-01-01 16:19:00 | 2013-01-01 16:20:00 |
| 8 | 2698 | french | active | 2013-01-01 04:38:00 | 2013-01-01 04:40:00 |
| 11 | 3745 | english | active | 2013-01-01 08:07:00 | 2013-01-01 08:09:00 |
| 13 | 4025 | english | active | 2013-01-02 12:27:00 | 2013-01-02 12:29:00 |
| 15 | 4259 | english | active | 2013-01-02 15:39:00 | 2013-01-02 15:41:00 |
| 17 | 5025 | japanese | active | 2013-01-02 10:56:00 | 2013-01-02 10:57:00 |
| 19 | 326 | english | active | 2013-01-02 09:54:00 | 2013-01-02 09:55:00 |
| 20 | 7 | italian | active | 2013-01-02 09:41:00 | 2013-01-02 09:43:00 |
| 21 | 2606 | english | active | 2013-01-02 09:29:00 | 2013-01-02 09:30:00 |
| 22 | 545 | german | active | 2013-01-02 17:36:00 | 2013-01-02 17:38:00 |
| 27 | 6 | japanese | active | 2013-01-03 16:14:00 | 2013-01-03 16:15:00 |
| 30 | 4148 | english | active | 2013-01-03 08:28:00 | 2013-01-03 08:29:00 |

***-- Table-2 events***

*create table events(*
*user_id int,*
*occurred_at varchar(100),*
*event_type varchar(60),*
*event_name varchar(80),*
*location varchar(50),*

*device varchar(100),*
*user_type int);*

*LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.1/Uploads/Table-2 events.csv"*
*INTO TABLE events*
*FIELDS TERMINATED BY ','*
*ENCLOSED BY '""'*
*IGNORE 1 ROWS;*

*select * from events;*

*ALTER TABLE events ADD COLUMN temp_occurred_at DATETIME;*
*UPDATE events SET temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');*
*ALTER TABLE events DROP COLUMN occurred_at;*
*ALTER TABLE events CHANGE COLUMN temp_occurred_at occurred_at DATETIME;*

*select * from events;*

```
56 •  select * from events;
57
58 •  ALTER TABLE events ADD COLUMN temp_occurred_at DATETIME;
59 •  UPDATE events SET temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');
60 •  ALTER TABLE events DROP COLUMN occurred_at;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| user_id | event_type | event_name | location | device | user_type | occurred_at |
|---|---|---|---|---|---|---|
| 10522 | engagement | login | Japan | dell inspiron notebook | 3 | 2014-05-02 11:02:00 |
| 10522 | engagement | home_page | Japan | dell inspiron notebook | 3 | 2014-05-02 11:02:00 |
| 10522 | engagement | like_message | Japan | dell inspiron notebook | 3 | 2014-05-02 11:03:00 |
| 10522 | engagement | view_inbox | Japan | dell inspiron notebook | 3 | 2014-05-02 11:04:00 |
| 10522 | engagement | search_run | Japan | dell inspiron notebook | 3 | 2014-05-02 11:03:00 |
| 10522 | engagement | search_run | Japan | dell inspiron notebook | 3 | 2014-05-02 11:03:00 |
| 10612 | engagement | login | Netherlands | iphone 5 | 1 | 2014-05-01 09:59:00 |
| 10612 | engagement | like_message | Netherlands | iphone 5 | 1 | 2014-05-01 10:00:00 |
| 10612 | engagement | send_message | Netherlands | iphone 5 | 1 | 2014-05-01 10:00:00 |
| 10612 | engagement | home_page | Netherlands | iphone 5 | 1 | 2014-05-01 10:01:00 |
| 10612 | engagement | like_message | Netherlands | iphone 5 | 1 | 2014-05-01 10:01:00 |
| 10612 | engagement | home_page | Netherlands | iphone 5 | 1 | 2014-05-01 10:02:00 |
| 10612 | engagement | view_inbox | Netherlands | iphone 5 | 1 | 2014-05-01 10:02:00 |
| 10612 | engagement | like_message | Netherlands | iphone 5 | 1 | 2014-05-01 10:03:00 |
| 10612 | engagement | home_page | Netherlands | iphone 5 | 1 | 2014-05-01 10:03:00 |
| 10612 | engagement | send_message | Netherlands | iphone 5 | 1 | 2014-05-01 10:04:00 |

***-- Table-3 email_events***

*create table email_events(*
*user_id int,*

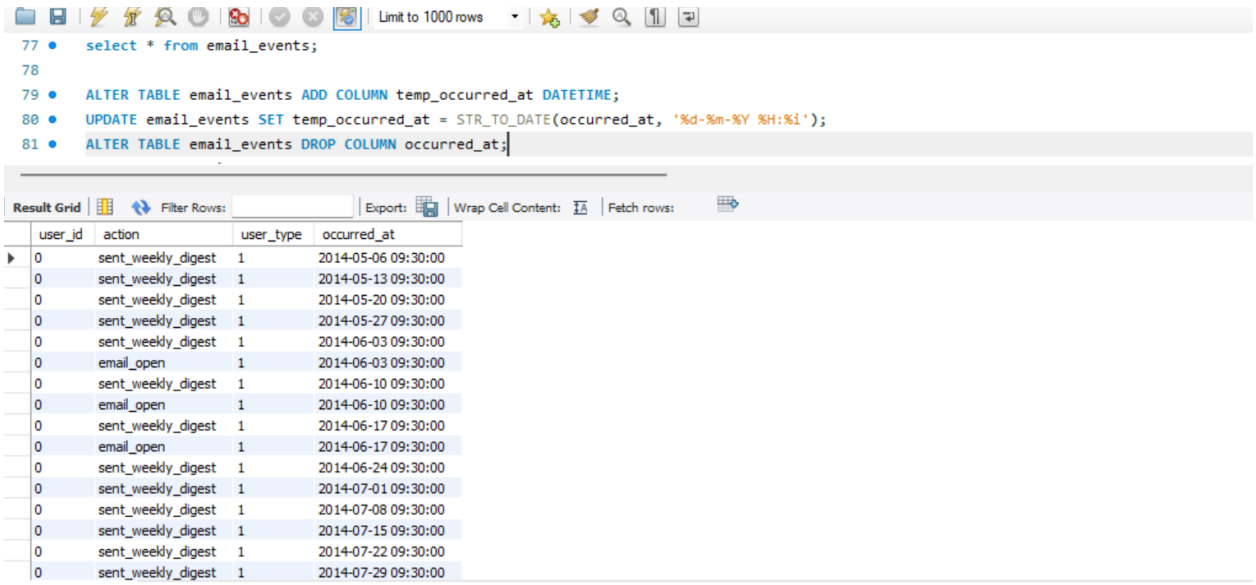
*occurred_at varchar(100),*
*action varchar(100),*
*user_type int);*

*LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.1/Uploads/Table-3 email_events.csv"*
*INTO TABLE email_events*
*FIELDS TERMINATED BY ','*
*ENCLOSED BY ""*
*IGNORE 1 ROWS;*

*select * from email_events;*

*ALTER TABLE email_events ADD COLUMN temp_occurred_at DATETIME;*
*UPDATE email_events SET temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');*
*ALTER TABLE email_events DROP COLUMN occurred_at;*
*ALTER TABLE email_events CHANGE COLUMN temp_occurred_at occurred_at DATETIME;*

*select * from email_events;*



```
77 • select * from email_events;
78
79 • ALTER TABLE email_events ADD COLUMN temp_occurred_at DATETIME;
80 • UPDATE email_events SET temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');
81 • ALTER TABLE email_events DROP COLUMN occurred_at;
```

| user_id | action | user_type | occurred_at |
|---|---|---|---|
| 0 | sent_weekly_digest | 1 | 2014-05-06 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-05-13 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-05-20 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-05-27 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-06-03 09:30:00 |
| 0 | email_open | 1 | 2014-06-03 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-06-10 09:30:00 |
| 0 | email_open | 1 | 2014-06-10 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-06-17 09:30:00 |
| 0 | email_open | 1 | 2014-06-17 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-06-24 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-07-01 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-07-08 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-07-15 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-07-22 09:30:00 |
| 0 | sent_weekly_digest | 1 | 2014-07-29 09:30:00 |

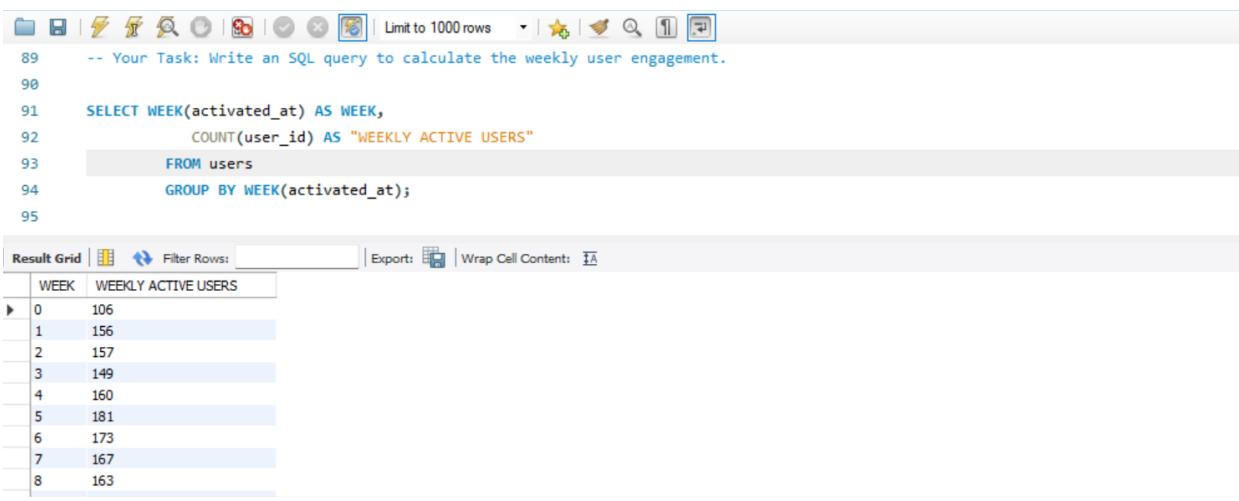
**Tasks:**

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.

**Code**

```
SELECT WEEK(activated_at) AS WEEK,
        COUNT(user_id) AS "WEEKLY ACTIVE USERS"
    FROM users
    GROUP BY WEEK(activated_at);
```

**Output**




| WEEK | WEEKLY ACTIVE USERS |
|---|---|
| 0 | 106 |
| 1 | 156 |
| 2 | 157 |
| 3 | 149 |
| 4 | 160 |
| 5 | 181 |
| 6 | 173 |
| 7 | 167 |
| 8 | 163 |

**Insights**

The dataset represents the weekly activity levels of users, with each pair indicating the week number and corresponding user activity count. User activity varies throughout the weeks, with a notable increase around week 14. The overall trend suggests

fluctuating user engagement, possibly influenced by external factors or events.

2. **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.

**Code**

*SELECT Months, Users, ROUND(((Users / LAG(Users, 1) OVER (ORDER BY Months) - 1) * 100), 2) AS "Growth in %"*

*FROM ( SELECT EXTRACT(MONTH FROM created_at) AS Months, COUNT(activated_at) AS Users FROM users*

*WHERE activated_at IS NOT NULL -- Exclude rows where activated_at is NULL*

*GROUP BY 1*

*ORDER BY 1*

*) sub;*

**Output**

```
-- User Growth Analysis: B
-- 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.

SELECT Months, Users, ROUND(((Users / LAG(Users, 1) OVER (ORDER BY Months) - 1) * 100), 2) AS "Growth in %"
FROM ( SELECT EXTRACT(MONTH FROM created_at) AS Months, COUNT(activated_at) AS Users FROM users
WHERE activated_at IS NOT NULL  -- Exclude rows where activated_at is NULL
GROUP BY 1
ORDER BY 1
) sub;
```

Result Grid

| Months | Users | Growth in % |
|---|---|---|
| 1 | 712 | NULL |
| 2 | 685 | -3.79 |
| 3 | 765 | 11.68 |
| 4 | 907 | 18.56 |
| 5 | 993 | 9.48 |
| 6 | 1086 | 9.37 |
| 7 | 1281 | 17.96 |
| 8 | 1347 | 5.15 |
| 9 | 330 | -75.50 |
| 10 | 390 | 18.18 |
| 11 | 399 | 2.31 |
| 12 | 486 | 21.80 |

**Insights**

The data reveals fluctuations in user growth over the months, with a noticeable surge in months 4, 7, and 12, indicating potential positive developments or marketing impacts. However, a significant drop in user count in month 9 suggests a possible issue or decline in user engagement during that period. Overall, understanding the underlying factors contributing to these fluctuations can help optimize strategies for sustained user growth.

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.

**Code**

```
SELECT first AS "Week Numbers",

SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS "Week 0",

SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS "Week 1",

SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS "Week 2",

SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS "Week 3",

SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS "Week 4",
```

```
SUM(CASE WHEN week_number = 5 THEN 1 ELSE 0 END) AS "Week 5",

SUM(CASE WHEN week_number = 6 THEN 1 ELSE 0 END) AS "Week 6",

SUM(CASE WHEN week_number = 7 THEN 1 ELSE 0 END) AS "Week 7",

SUM(CASE WHEN week_number = 8 THEN 1 ELSE 0 END) AS "Week 8",

SUM(CASE WHEN week_number = 9 THEN 1 ELSE 0 END) AS "Week 9",

SUM(CASE WHEN week_number = 10 THEN 1 ELSE 0 END) AS "Week 10",

SUM(CASE WHEN week_number = 11 THEN 1 ELSE 0 END) AS "Week 11",

SUM(CASE WHEN week_number = 12 THEN 1 ELSE 0 END) AS "Week 12",

SUM(CASE WHEN week_number = 13 THEN 1 ELSE 0 END) AS "Week 13",

SUM(CASE WHEN week_number = 14 THEN 1 ELSE 0 END) AS "Week 14",

SUM(CASE WHEN week_number = 15 THEN 1 ELSE 0 END) AS "Week 15",

SUM(CASE WHEN week_number = 16 THEN 1 ELSE 0 END) AS "Week 16",

SUM(CASE WHEN week_number = 17 THEN 1 ELSE 0 END) AS "Week 17",

SUM(CASE WHEN week_number = 18 THEN 1 ELSE 0 END) AS "Week 18"

FROM
```

*(*

*SELECT m.user_id, m.login_week, n.first, m.login_week - first AS week_number FROM*

*(SELECT user_id, EXTRACT(WEEK FROM occurred_at) AS login_week FROM events GROUP BY 1, 2) m,*

*(SELECT user_id, MIN(EXTRACT(WEEK FROM occurred_at)) AS first FROM events GROUP BY 1) n*

*WHERE m.user_id = n.user_id*

*) sub*

*GROUP BY first ORDER BY first;*

## Output

```
-- Weekly Retention Analysis: C
-- 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.

SELECT first AS "Week Numbers",
```

| Week Numbers | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | 663 | 472 | 324 | 251 | 205 | 187 | 167 | 146 | 145 | 145 | 136 | 131 | 132 | 143 | 116 |
| 18 | 596 | 362 | 261 | 203 | 168 | 147 | 144 | 127 | 113 | 122 | 106 | 118 | 127 | 110 | 97 |
| 19 | 427 | 284 | 173 | 153 | 114 | 95 | 91 | 81 | 95 | 82 | 68 | 65 | 63 | 42 | 51 |
| 20 | 358 | 223 | 165 | 121 | 91 | 72 | 63 | 67 | 63 | 65 | 67 | 41 | 40 | 33 | 40 |
| 21 | 317 | 187 | 131 | 91 | 74 | 63 | 75 | 72 | 58 | 48 | 45 | 39 | 35 | 28 | 2 |
| 22 | 326 | 224 | 150 | 107 | 87 | 73 | 63 | 60 | 55 | 48 | 41 | 39 | 31 | 1 | 0 |
| 23 | 328 | 219 | 138 | 101 | 90 | 79 | 69 | 61 | 54 | 47 | 35 | 30 | 0 | 0 | 0 |
| 24 | 339 | 205 | 143 | 102 | 81 | 63 | 65 | 61 | 38 | 39 | 29 | 0 | 0 | 0 | 0 |
| 25 | 305 | 218 | 139 | 101 | 75 | 63 | 50 | 46 | 38 | 35 | 2 | 0 | 0 | 0 | 0 |
| 26 | 288 | 181 | 114 | 83 | 73 | 55 | 47 | 43 | 29 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 292 | 199 | 121 | 106 | 68 | 53 | 40 | 36 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | 274 | 194 | 114 | 69 | 46 | 30 | 28 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | 270 | 186 | 102 | 65 | 47 | 40 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 294 | 202 | 121 | 78 | 53 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Insights

| Week Numbers | 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 | Week 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | 663 | 472 | 324 | 251 | 205 | 187 | 167 | 146 | 145 | 145 | 136 | 131 | 132 | 143 | 116 | 91 | 82 | 77 | 5 |
| 18 | 596 | 362 | 261 | 203 | 168 | 147 | 144 | 127 | 113 | 122 | 106 | 118 | 127 | 110 | 97 | 85 | 67 | 4 | 0 |
| 19 | 427 | 284 | 173 | 153 | 114 | 95 | 91 | 81 | 95 | 82 | 68 | 65 | 63 | 42 | 51 | 49 | 2 | 0 | 0 |
| 20 | 358 | 223 | 165 | 121 | 91 | 72 | 63 | 67 | 63 | 65 | 67 | 41 | 40 | 33 | 40 | 0 | 0 | 0 | 0 |
| 21 | 317 | 187 | 131 | 91 | 74 | 63 | 75 | 72 | 58 | 48 | 45 | 39 | 35 | 28 | 2 | 0 | 0 | 0 | 0 |
| 22 | 326 | 224 | 150 | 107 | 87 | 73 | 63 | 60 | 55 | 48 | 41 | 39 | 31 | 1 | 0 | 0 | 0 | 0 | 0 |
| 23 | 328 | 219 | 138 | 101 | 90 | 79 | 69 | 61 | 54 | 47 | 35 | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 | 339 | 205 | 143 | 102 | 81 | 63 | 65 | 61 | 38 | 39 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 305 | 218 | 139 | 101 | 75 | 63 | 50 | 46 | 38 | 35 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | 288 | 181 | 114 | 83 | 73 | 55 | 47 | 43 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 292 | 199 | 121 | 106 | 68 | 53 | 40 | 36 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | 274 | 194 | 114 | 69 | 46 | 30 | 28 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | 270 | 186 | 102 | 65 | 47 | 40 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 294 | 202 | 121 | 78 | 53 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | 215 | 145 | 76 | 57 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | 267 | 188 | 94 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | 286 | 202 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 34 | 279 | 44 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 35 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The provided data represents user counts for each week after signing up for the product. In analyzing user retention, it is evident that retention rates decline over time, with a significant drop in active users by Week 5. This suggests a challenge in sustaining user engagement beyond the initial weeks. Understanding and addressing factors affecting retention, especially during critical periods like Week 5, can be crucial for improving the product's long-term user retention strategy.

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.

**Code**

```
SELECT EXTRACT(WEEK FROM occurred_at) AS "Week Numbers",

COUNT(DISTINCT CASE WHEN device IN('dell inspiron notebook') THEN user_id ELSE NULL END) AS "Dell Inspiron Notebook",

COUNT(DISTINCT CASE WHEN device IN('iphone 5') THEN user_id ELSE NULL END) AS

"iPhone 5",

COUNT(DISTINCT CASE WHEN device IN('iphone 4s') THEN user_id ELSE NULL END) AS

"iPhone 4S",

COUNT(DISTINCT CASE WHEN device IN('windows surface') THEN user_id ELSE NULL END) AS "Windows Surface",

COUNT(DISTINCT CASE WHEN device IN('macbook air') THEN user_id ELSE NULL END) AS "Macbook Air",
```

*COUNT(DISTINCT CASE WHEN device IN('iphone 5s') THEN user_id ELSE NULL END) AS*

*"iPhone 5S",*

*COUNT(DISTINCT CASE WHEN device IN('macbook pro') THEN user_id ELSE NULL END) AS "Macbook Pro",*

*COUNT(DISTINCT CASE WHEN device IN('kindle fire') THEN user_id ELSE NULL END) AS "Kindle Fire",*

*COUNT(DISTINCT CASE WHEN device IN('ipad mini') THEN user_id ELSE NULL END) AS "iPad Mini",*

*COUNT(DISTINCT CASE WHEN device IN('nexus 7') THEN user_id ELSE NULL END) AS*

*"Nexus 7",*

*COUNT(DISTINCT CASE WHEN device IN('nexus 5') THEN user_id ELSE NULL END) AS*

*"Nexus 5",*

*COUNT(DISTINCT CASE WHEN device IN('samsung galaxy s4') THEN user_id ELSE NULL END) AS "Samsung Galaxy S4",*

*COUNT(DISTINCT CASE WHEN device IN('lenovo thinkpad') THEN user_id ELSE NULL END) AS "Lenovo Thinkpad",*

*COUNT(DISTINCT CASE WHEN device IN('samsumg galaxy tablet') THEN user_id ELSE NULL END) AS "Samsumg Galaxy Tablet",*

*COUNT(DISTINCT CASE WHEN device IN('acer aspire notebook') THEN user_id ELSE NULL END) AS "Acer Aspire Notebook",*

*COUNT(DISTINCT CASE WHEN device IN('asus chromebook') THEN user_id ELSE NULL END) AS "Asus Chromebook",*

*COUNT(DISTINCT CASE WHEN device IN('htc one') THEN user_id ELSE NULL END) AS "HTC One",*

```
COUNT(DISTINCT CASE WHEN device IN('nokia lumia 635') THEN
user_id ELSE NULL END) AS "Nokia Lumia 635",

COUNT(DISTINCT CASE WHEN device IN('samsung galaxy note')
THEN user_id ELSE NULL END) AS "Samsung Galaxy Note",

COUNT(DISTINCT CASE WHEN device IN('acer aspire desktop')
THEN user_id ELSE NULL END) AS "Acer Aspire Desktop",

COUNT(DISTINCT CASE WHEN device IN('mac mini') THEN
user_id ELSE NULL END) AS "Mac Mini",

COUNT(DISTINCT CASE WHEN device IN('hp pavilion desktop')
THEN user_id ELSE NULL END) AS "HP Pavilion Desktop",

COUNT(DISTINCT CASE WHEN device IN('dell inspiron desktop')
THEN user_id ELSE NULL END) AS "Dell Inspiron Desktop",

COUNT(DISTINCT CASE WHEN device IN('ipad air') THEN user_id
ELSE NULL END) AS "iPad Air",


COUNT(DISTINCT CASE WHEN device IN('amazon fire phone')
THEN user_id ELSE NULL END) AS "Amazon Fire Phone",

COUNT(DISTINCT CASE WHEN device IN('nexus 10') THEN
user_id ELSE NULL END) AS

"Nexus 10" FROM events

WHERE event_type = 'engagement' GROUP BY 1

ORDER BY 1;
```

## Output

```
-- Weekly Engagement Per Device: D
-- 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 EXTRACT(WEEK FROM occurred_at) AS "Week Numbers",
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Week Numbers | Dell Inspiron Notebook | iPhone 5 | iPhone 4S | Windows Surface | Macbook Air | iPhone 5S | Macbook Pro | Kindle Fire | iPad Mini | Nexus 7 | Nexus 5 | Samsung Galaxy S4 | Lenc Thinl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 92 | 125 | 45 | 15 | 145 | 71 | 251 | 21 | 34 | 45 | 96 | 105 | 176 |
| 23 | 103 | 152 | 53 | 14 | 124 | 79 | 266 | 25 | 33 | 36 | 88 | 99 | 176 |
| 24 | 99 | 142 | 53 | 22 | 152 | 79 | 255 | 25 | 39 | 49 | 87 | 101 | 165 |
| 25 | 105 | 137 | 40 | 22 | 121 | 78 | 275 | 24 | 30 | 51 | 89 | 99 | 197 |
| 26 | 89 | 152 | 50 | 21 | 134 | 94 | 269 | 26 | 43 | 46 | 87 | 112 | 192 |
| 27 | 89 | 163 | 67 | 33 | 142 | 83 | 302 | 25 | 35 | 40 | 84 | 116 | 202 |
| 28 | 103 | 151 | 61 | 33 | 148 | 93 | 295 | 31 | 35 | 39 | 85 | 122 | 220 |
| 29 | 113 | 144 | 60 | 28 | 148 | 90 | 295 | 37 | 34 | 45 | 77 | 123 | 209 |
| 30 | 127 | 152 | 65 | 19 | 159 | 103 | 322 | 25 | 35 | 62 | 84 | 103 | 206 |
| 31 | 113 | 135 | 56 | 19 | 147 | 71 | 321 | 14 | 27 | 38 | 69 | 100 | 207 |
| 32 | 104 | 119 | 34 | 10 | 125 | 67 | 307 | 12 | 30 | 25 | 67 | 82 | 179 |
| 33 | 110 | 110 | 35 | 15 | 133 | 65 | 312 | 14 | 28 | 30 | 70 | 80 | 191 |
| 34 | 105 | 101 | 50 | 18 | 136 | 70 | 292 | 13 | 25 | 33 | 70 | 90 | 193 |
| 35 | 9 | 2 | 6 | 3 | 10 | 3 | 17 | 3 | 2 | 2 | 4 | 6 | 16 |

## Insights

The provided data represents the activity levels of users every week per device. Analyzing the dataset reveals varying levels of user engagement across weeks and devices. Weeks with higher activity counts may indicate increased user interaction, while lower counts may suggest reduced engagement. Further investigation into the specific activities or features associated with each device can provide insights into user preferences and guide strategies to enhance overall user satisfaction and retention. Additionally, monitoring fluctuations in activity levels over time can help identify trends and patterns for targeted improvements or interventions.

| Week Numbers | Dell Inspiron Notebook | iPhone 5 | iPhone 4S | Windows Surface | Macbook Air | iPhone 5S | Macbook Pro | Kindle Fire | iPad Mini | Nexus 7 | Nexus 5 | Samsung Galaxy S4 | Lenovo Thinkpad | Samsung Galaxy Tablet | Acer Aspire Notebook | Asus Chromebook | HTC One | Nokia Lumia 635 | Samsung Galaxy Note | Acer Aspire Desktop | Mac Mini | HP Pavilion Desktop | Dell Inspiron Desktop | iPad Air | Amazon Fire Phone | Nexus 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 17 | 46 | 65 | 21 | 10 | 54 | 42 | 143 | 6 | 19 | 18 | 40 | 52 | 86 | 8 | 20 | 21 | 16 | 17 | 7 | 9 | 6 | 14 | 18 | 27 | 4 | 16 |
| 18 | 77 | 113 | 46 | 10 | 121 | 73 | 252 | 27 | 30 | 30 | 73 | 82 | 153 | 11 | 33 | 42 | 19 | 33 | 15 | 26 | 13 | 37 | 58 | 52 | 9 | 30 |
| 19 | 83 | 115 | 44 | 16 | 112 | 79 | 266 | 21 | 36 | 41 | 87 | 91 | 178 | 6 | 41 | 27 | 30 | 23 | 11 | 23 | 18 | 40 | 36 | 55 | 12 | 25 |
| 20 | 84 | 125 | 55 | 21 | 119 | 79 | 256 | 23 | 32 | 52 | 103 | 93 | 173 | 9 | 40 | 41 | 29 | 22 | 18 | 23 | 26 | 30 | 52 | 59 | 11 | 22 |
| 21 | 80 | 137 | 45 | 17 | 110 | 74 | 247 | 30 | 23 | 29 | 91 | 84 | 167 | 6 | 47 | 38 | 21 | 25 | 20 | 29 | 18 | 44 | 41 | 51 | 5 | 25 |
| 22 | 92 | 125 | 45 | 15 | 145 | 71 | 251 | 21 | 34 | 45 | 96 | 105 | 176 | 10 | 41 | 52 | 24 | 25 | 19 | 25 | 25 | 38 | 52 | 58 | 5 | 27 |
| 23 | 103 | 152 | 53 | 14 | 124 | 79 | 266 | 25 | 33 | 36 | 88 | 99 | 176 | 14 | 43 | 49 | 20 | 31 | 14 | 22 | 18 | 54 | 53 | 41 | 16 | 45 |
| 24 | 99 | 142 | 53 | 22 | 152 | 79 | 255 | 25 | 39 | 49 | 87 | 101 | 165 | 11 | 40 | 43 | 20 | 35 | 20 | 24 | 29 | 56 | 59 | 57 | 11 | 38 |
| 25 | 105 | 137 | 40 | 22 | 121 | 78 | 275 | 24 | 30 | 51 | 89 | 99 | 197 | 12 | 47 | 38 | 21 | 37 | 14 | 28 | 21 | 52 | 52 | 57 | 13 | 29 |
| 26 | 89 | 152 | 50 | 21 | 134 | 94 | 269 | 26 | 43 | 46 | 87 | 112 | 192 | 12 | 35 | 49 | 23 | 42 | 9 | 29 | 11 | 46 | 60 | 56 | 13 | 29 |
| 27 | 89 | 163 | 67 | 33 | 142 | 83 | 302 | 25 | 35 | 40 | 84 | 116 | 202 | 15 | 49 | 52 | 27 | 31 | 15 | 29 | 15 | 56 | 53 | 55 | 10 | 37 |
| 28 | 103 | 151 | 61 | 33 | 148 | 93 | 295 | 31 | 35 | 39 | 85 | 122 | 220 | 9 | 49 | 50 | 26 | 35 | 10 | 30 | 28 | 56 | 56 | 54 | 6 | 26 |
| 29 | 113 | 144 | 60 | 28 | 148 | 90 | 295 | 37 | 34 | 45 | 77 | 123 | 209 | 13 | 53 | 49 | 31 | 43 | 16 | 28 | 31 | 58 | 54 | 52 | 12 | 25 |
| 30 | 127 | 152 | 65 | 19 | 159 | 103 | 322 | 25 | 35 | 62 | 84 | 103 | 206 | 9 | 60 | 56 | 31 | 34 | 15 | 33 | 23 | 42 | 54 | 70 | 12 | 36 |
| 31 | 113 | 135 | 56 | 19 | 147 | 71 | 321 | 14 | 27 | 38 | 69 | 100 | 207 | 8 | 55 | 56 | 13 | 28 | 14 | 31 | 24 | 51 | 44 | 55 | 14 | 24 |
| 32 | 104 | 119 | 34 | 10 | 125 | 67 | 307 | 12 | 30 | 25 | 67 | 82 | 179 | 6 | 55 | 62 | 18 | 28 | 12 | 35 | 20 | 51 | 57 | 48 | 12 | 30 |
| 33 | 110 | 110 | 35 | 15 | 133 | 65 | 312 | 14 | 28 | 30 | 70 | 80 | 191 | 12 | 46 | 49 | 19 | 27 | 13 | 39 | 32 | 38 | 37 | 40 | 14 | 23 |
| 34 | 105 | 101 | 50 | 18 | 136 | 70 | 292 | 13 | 25 | 33 | 70 | 90 | 193 | 14 | 63 | 47 | 25 | 17 | 13 | 30 | 30 | 36 | 49 | 39 | 11 | 25 |
| 35 | 9 | 2 | 6 | 3 | 10 | 3 | 17 | 3 | 2 | 2 | 4 | 6 | 16 | 0 | 3 | 6 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 0 | 0 | 2 |

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.

**Code**

```
SELECT Week,

ROUND((weekly_digest/total*100),2) AS "Weekly Digest Rate", ROUND((email_opens/total*100),2) AS "Email Open Rate", ROUND((email_clickthroughs/total*100),2) AS "Email Clickthrough Rate", ROUND((reengagement_emails/total*100),2) AS "Reengagement Email Rate" FROM

( SELECT EXTRACT(WEEK FROM occurred_at) AS Week,

COUNT(CASE WHEN action = 'sent_weekly_digest' THEN user_id ELSE NULL END) AS weekly_digest,

COUNT(CASE WHEN action = 'email_open' THEN user_id ELSE NULL END) AS email_opens,

COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id ELSE NULL END) AS email_clickthroughs,

COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id ELSE NULL END) AS reengagement_emails,

COUNT(user_id) AS total FROM email_events GROUP BY 1

) sub

GROUP BY 1

ORDER BY 1;
```

## Output

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

SELECT Week,
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Week | Weekly Digest Rate | Email Open Rate | Email Clickthrough Rate | Reengagement Email Rate |
|---|---|---|---|---|
| 17 | 62.32 | 21.28 | 11.39 | 5.01 |
| 18 | 63.45 | 22.24 | 10.49 | 3.83 |
| 19 | 62.16 | 22.67 | 11.13 | 4.04 |
| 20 | 61.62 | 22.64 | 11.43 | 4.31 |
| 21 | 63.52 | 22.82 | 9.97 | 3.69 |
| 22 | 63.59 | 21.56 | 10.66 | 4.19 |
| 23 | 62.39 | 22.34 | 11.18 | 4.09 |
| 24 | 61.61 | 22.92 | 10.99 | 4.48 |
| 25 | 63.77 | 21.79 | 10.54 | 3.90 |
| 26 | 62.99 | 22.22 | 10.61 | 4.18 |
| 27 | 62.24 | 22.49 | 11.37 | 3.90 |
| 28 | 62.92 | 22.48 | 10.77 | 3.83 |
| 29 | 63.98 | 21.71 | 10.51 | 3.79 |
| 30 | 62.29 | 23.24 | 10.59 | 3.88 |
| 31 | 65.27 | 23.25 | 7.66 | 3.82 |

## Insights

The provided data represents engagement metrics for an email service across different weeks. The percentages indicate the distribution of user engagement in various categories, such as opening, clicking, and responding to emails. Analyzing the data reveals that users generally show consistent engagement, with variations in the distribution across specific activities. For instance, Week 35 stands out with a significant increase in the "clicked" category, suggesting a potential shift in user behavior or an anomaly that requires further investigation. Understanding these patterns can help optimize email content and strategies to enhance user interaction and overall email service effectiveness.

**Result**

How this project benefited me: It has made it easier for me to see how crucial operation analytics is. I can now comprehend how businesses employ metric spikes as a covert weapon thanks to this initiative. By being proactive and well-informed, they may use insights to make data-driven decisions that optimize their approach and increase return on investment.

In conclusion, real-time data synchronization is how Operational Analytics addresses the issue. Operational analytics may combine data from several sources to produce a cumulative, well-organized, and actionable solution. It can also provide analytical models in real-time, allowing businesses to build personalized customer profiles and an overall picture of their operations.