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. Working 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.

Case Study 1: Job Data Analysis

1.Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

```
SELECT
   COUNT(DISTINCT job_id) / (30 * 24) AS no_reviewed
FROM
   job_data
WHERE
   ds BETWEEN '11/1/2020' AND '11/30/2020';
```

Throughput Analysis:

2.Objective: Calculate the 7-day rolling average of throughput (number of events per second). 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.

```
SELECT
    ds, jobs_reviewed,
    AVG(jobs_reviewed)
         OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS throuhput_rolling
FROM (SELECT
   ds, COUNT(DISTINCT job_id) AS jobs_reviewed
FROM
   job_data
WHERE
   ds BETWEEN '11/1/2020' AND '11/30/2020'
GROUP BY ds
ORDER BY ds)a;
```

Language Share Analysis:

3.Objective: Calculate the percentage share of each language in the last 30 days.
Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

```
SELECT
   language,num_jobs,
   num_jobs * 100.0 / total_jobs AS percentage_share
FROM (
   SELECT
      language,
      COUNT( job_id) AS num_jobs
   FROM
      job_data
      group by language)a
      cross join
   (
     select count( job_id) as total_jobs
     from job_data)b;
```

4.Duplicate Rows Detection:
Objective: Identify duplicate rows in the data.
Task: Write an SQL query to display duplicate rows from the job_data table.

```
SELECT *
FROM (
    SELECT * ,
           row_number()over (partition by job_id) as row_num
     FROM job_data
     )a
WHERE row_num>1;
```

Case Study 2: Investigating Metric Spike

1.Weekly User Engagement:
Objective: Measure the activeness of users on a weekly basis.
Task: Write an SQL query to calculate the weekly user engagement.

```
SELECT
   EXTRACT(WEEK FROM occurred_at) AS week_start_date,
   COUNT(DISTINCT user_id) AS active_users
FROM
   events
GROUP BY
   week_start_date
ORDER BY
   week_start_date;
```

2.User Growth Analysis:
Objective: Analyze the growth of users over time for a product.
Task: Write an SQL query to calculate the user growth for the product.

```
SELECT
   extract(month from created_at) AS month,
   COUNT(DISTINCT user_id) AS new_users
FROM
   users
GROUP BY
   month
ORDER BY
   month;
```

3.Weekly Retention Analysis:
Objective: Analyze the retention of users on a weekly basis after signing up for a product.

Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

```
WITH user_ret AS (
   SELECT
      distinct user_id,
      MIN(created_at) AS signup_date
   FROM
      users
   GROUP BY
      user_id
)
SELECT
   extract(week from signup_date) AS signup_week,
   extract(week from e.occurred_at) AS engagement_week,
   COUNT(DISTINCT e.user_id) AS retained_users
FROM
   user_ret c
JOIN
   events e ON c.user_id = e.user_id
WHERE
   e.occurred_at BETWEEN c.signup_date AND DATE_ADD(c.signup_date, INTERVAL 7 DAY)
GROUP BY
   signup_week,
   engagement_week
ORDER BY
   signup_week,
   engagement_week;
```

4.Weekly Engagement Per Device:
Objective: Measure the activeness of users on a weekly basis per device.
Task: Write an SQL query to calculate the weekly engagement per device.

```
SELECT
   EXTRACT(YEAR FROM occurred_at) AS year_num,
   EXTRACT(WEEK FROM occurred_at) AS week_num_device,
   COUNT(DISTINCT user_id) AS no_of_users
FROM
   events
WHERE
   event_type = 'engagement'
GROUP BY 1 , 2
```

```
ORDER BY 1 , 2 , 3;
```

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

```
SELECT
   action,
   COUNT(DISTINCT user_id) AS unique_users,
   COUNT(*) AS total_actions
FROM
   email_events
GROUP BY
   action
ORDER BY
   action;
```

TECH STACK USED

MySQL Workbench (Version 8.0 CE): MySQL Workbench provides data modelling. SQL development, and various administration tools for configuration. It also offers a graphical interface to work with the databases in a structured way. It is easy and free to use MySQL create a database and perform analysis answering the questions given in the description.

Microsoft Word 2021: It is used to make a report (PDF) to be presented to the leadership team.

Microsoft Excel 2021: it is used to make visualizations to be presented alongside the insights

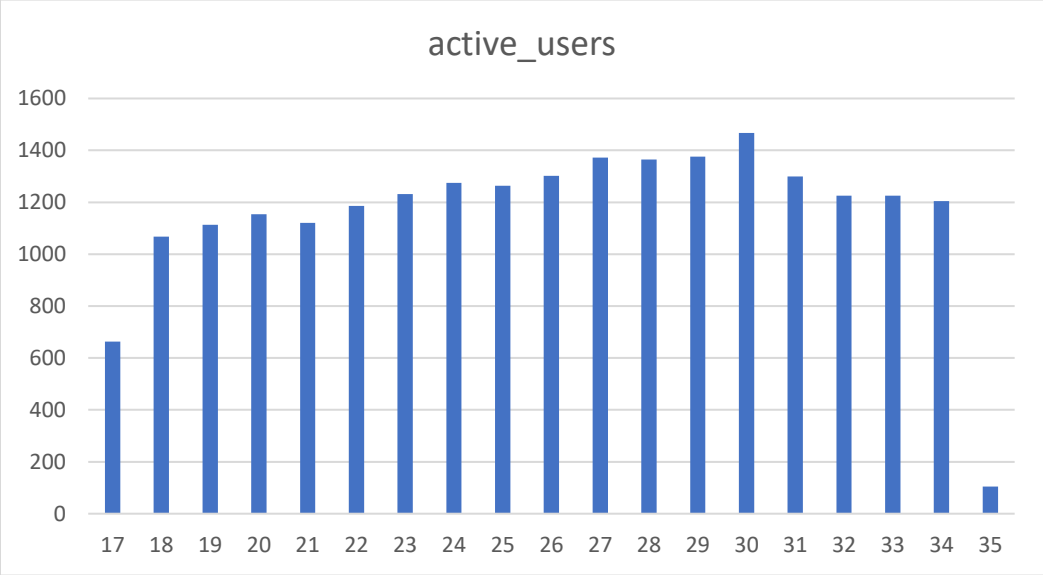
Insights Driven From The case study

Case Study 1 (Job Data):

- The number of distinct jobs reviewed per hour per day for November 2020 is 83%.
- We used the 7-day rolling average of throughput as it gives the average for all the days rig from day 1 to day 7 whereas, daily metric gives the average for only that particular day its
- The percentage share of Persian language is the most (37.5%).
- There are two duplicate rows if we partition the data by job_id. But if we look the overall columns, all the rows are unique.

Case Study 2 (Investigating metric spike):

- The weekly user engagement increased from week 18th to week 31st and then start declining from then onwards. This means that some of the users do not find much quality the product/service in the last of the weeks.



- The number of users increases towards the 8 th month then fall down and keep going constant towards the end of the year