OPERATIONAL ANALYSIS AND INVESTIGATING METRIC SPIKE:

Project Overview:

This project focuses on Operational Analytics and Investigating Metric Spikes at a company like Microsoft. The goal is to analyze user interactions and operational data to provide actionable insights that improve company operations and understand sudden changes in key metrics.

Description:

The purpose of this project is to utilize advanced SQL skills to derive valuable insights from operational and user data. These insights will support various departments, such as operations, support, and marketing, in making informed decisions to enhance user engagement, identify growth opportunities, and improve overall efficiency.

Approach:

I created the database and tables using MySQL Workbench and imported CSV files, ensuring data accuracy. For Job Data Analysis, I calculated hourly job reviews for November 2020, computed a 7-day rolling average of throughput, determined language shares over the last 30 days, and identified duplicate rows in `job_data`.

In Investigating Metric Spikes, I measured weekly user engagement, analyzed user growth, calculated retention rates, assessed engagement per device, and evaluated email interactions. Summarized results provided key business insights, enhanced with visualizations where necessary.

Finally, I compiled SQL queries and outputs, documented the approach, analysis, and insights in a structured report, saved it as a PDF, and uploaded it to Google Drive for stakeholder access.

Tech-Stack used:

For this project, I utilized the following software and tools:

MySQL Workbench (Version 8.0): This tool was used for creating and managing the database, writing and executing SQL queries, and visualizing data structures. MySQL Workbench provided a user-friendly interface to efficiently handle data import, perform complex queries, and analyze results.

Microsoft Excel (Version 2019): Excel was used for initial data inspection, cleaning, and preparation before importing the data into MySQL Workbench. It also helped in creating preliminary visualizations and performing basic statistical analysis.

Insights:

During this project, several key insights and observations were made:

1. User Activity Trends: Analyzing job reviews over time revealed peak activity hours and days, helping to identify when users are most engaged.

2. Throughput Consistency: The 7-day rolling average for throughput provided a smoother and more reliable metric compared to daily values, highlighting the importance of using aggregated metrics for operational stability.

3. Language Preferences: The language share analysis showed the dominant languages used on the platform, guiding content and localization strategies.

4. Duplicate Data Detection: Identifying duplicate rows in the `job_data` table ensured data quality and highlighted the need for data cleansing procedures.

5. User Engagement: Weekly user engagement metrics indicated which devices and activities were most popular, helping to tailor user experience improvements.

6. Retention Rates: Retention analysis pinpointed the drop-off points in user activity, suggesting areas where user retention strategies could be strengthened.

7. Email Interaction: Email engagement metrics provided insights into user responses to email campaigns, aiding in optimizing communication strategies.

These insights will guide operational improvements, enhance user engagement strategies, and inform future development decisions.

Results:

Case Study 1: Job Data Analysis:

A) The number of jobs reviewed per hour for each day in November 2020.

Conclusion: Following are the jobs reviewed per hour for each day in November 2020.

2020-11-30 0 2
2020-11-29 0 1
2020-11-28 0 2
2020-11-27 0 1
2020-11-26 0 1
2020-11-25 0 1

Code:

```
SELECT DATE(ds) as date, HOUR(ds) as hour, COUNT(job_id) as jobs_reviewed

FROM job_data

WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'

GROUP BY DATE(ds), HOUR(ds);
```

Insight: Calculating the number of jobs reviewed per hour for each day in November 2020 revealed the peak times when users are most active.

Interpretation: This information can help optimize system performance and plan maintenance during low-activity periods.

B) 7-day rolling average of throughput:

Conclusion: This is the rolling average of throughput of 7- days.

2020-11-25 0.00000000
2020-11-26 0.00000000
2020-11-27 0.00000000
2020-11-28 0.00000000
2020-11-29 0.00000000
2020-11-30 0.00000000

Preference: I prefer using the 7-day rolling average for throughput because it provides a more consistent and reliable view of the data. While the daily metric is useful for identifying immediate changes, it can be too volatile and misleading due to short-term anomalies. The rolling average helps in identifying overall trends and patterns, making it easier to derive actionable insights without being distracted by daily noise.

Code:

```
SELECT ds, AVG(events_per_second) OVER (ORDER BY ds ROWS 6 PRECEDING) AS rolling_avg_throughput
FROM (
  SELECT ds, COUNT(event) / 86400 AS events_per_second
  FROM job_data
  GROUP BY ds
) as daily_throughput;
```

Insight: The 7-day rolling average of throughput showed a smoother trend compared to daily metrics.

C) Language Share Analysis:

Conclusion: The percentage share of each language in the last 30 days.

Code:

```
SELECT language, COUNT(*) * 100.0 / (SELECT COUNT(*) FROM job_data WHERE ds
>= DATE_SUB(CURDATE(), INTERVAL 30 DAY)) AS language_share

FROM job_data

WHERE ds >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)

GROUP BY language;
```

Insight: The percentage share of each language over the last 30 days highlighted the most used languages on the platform.

Interpretation: This data can guide content localization efforts and prioritize language support for the most popular languages among users.

D) Duplicate Rows Detection:

Conclusion: These are the duplicates rows in the data.

Code:

```
SELECT job_id, actor_id, event, language, time_spent, org, ds, COUNT(*) AS Duplicates

FROM job_data

GROUP BY job_id, actor_id, event, language, time_spent, org, ds

HAVING COUNT(*) > 1;
```

Insight: Identifying duplicate rows in the job_data table ensured the integrity and accuracy of the data.

Interpretation: Removing duplicates improves data quality, leading to more accurate analysis and insights. It also highlights the need for better data entry controls.

Case Study 2: Investigating metric spike:

A) Weekly user engagement:

Conclusion: Activeness of users on a weekly basis

Code:

```
SELECT YEARWEEK(occurred_at) as week, user_id, COUNT(*) as engagement_count
FROM events
GROUP BY YEARWEEK(occurred_at), user_id;
```

Insight: Weekly user engagement metrics indicated periods of high and low activity, showing how users interact with the platform over time.

Interpretation: This information can be used to enhance user experience by focusing on features and content that drive engagement during peak times.

B) User Growth Analysis:

Conclusion: The growth of users over time for a product.

Code:

```
SELECT DATE(created_at) as date, COUNT(user_id) as new_users
FROM users
GROUP BY DATE(created_at);
```

Insight: Analyzing user growth over time showed trends in user acquisition and retention.

Interpretation: Identifying periods of significant growth or decline can inform marketing strategies and help in understanding the impact of product changes or campaigns.

C) Weekly Retention Analysis:

Conclusion: The retention of users on a weekly basis after signing up for a product.

Code:

```
SELECT signup_week, retention_week, COUNT(DISTINCT cohorts.user_id) as
retained_users
FROM (
  SELECT users.user_id, YEARWEEK(users.created_at) as signup_week,
YEARWEEK(events.occurred_at) as retention_week
  FROM users
```

```
  JOIN events ON users.user_id = events.user_id
) as cohorts
WHERE retention_week >= signup_week
GROUP BY signup_week, retention_week;
```

Insight: Calculating weekly retention rates revealed how well the platform retains users after they sign up.
Interpretation: High retention rates indicate user satisfaction and engagement, while low rates may signal issues that need addressing to improve user retention.

D) Weekly engagement per device:

Conclusion: The activeness of users on a weekly basis per device.

Code:

```
SELECT YEARWEEK(occurred_at) as week, device, COUNT(*) as engagement_count
FROM events
GROUP BY YEARWEEK(occurred_at), device;
```

Insight: Assessing weekly engagement per device type provided insights into which devices users prefer.

Interpretation: Understanding device preferences can guide development efforts, ensuring optimal performance and user experience across different devices.

E) Email engagement Analysis:

Conclusion: This is how users are engaging with queries.

Code:

```
SELECT action, COUNT(*) as engagement_count
FROM emailEvents
GROUP BY action;
```

Insight: Evaluating email engagement metrics showed how users interact with email campaigns.
Interpretation: Insights into email engagement can help optimize email marketing strategies, improve open and click-through rates, and enhance overall communication effectiveness.