Operation Analytics and Investigating Metric Spike

Project Description:

This project aims at doing job data analysis for operation analytics and finding metric spikes. Through this analysis, it would be possible to identify the trends in jobs and operations' metrics to provide insight into decision-making and improvement of efficiency. The approach is to examine historical job data, probe anomalies in key metrics, and perform statistical analysis to spot trends and insights that would streamline operations.

Approach:

I approached this project by first gathering relevant job data and performing initial data cleaning and preprocessing. Then, I zeroed in on finding metric spikes by isolating periods of significant deviations in the data. Statistical techniques such as anomaly detection and trend analysis were employed to understand the factors contributing to the spikes. The analysis was done using SQL queries and Python for data manipulation and visualization.

Tech-Stack Used:

MySQL Workbench: It is used to manage and query the job data stored in a MySQL database.

Python (Pandas, Matplotlib, Seaborn): Used for data processing, analysis, and visualization to find trends, correlations, and anomalies.

Jupyter Notebooks: Used to document and share the process and results of the analysis.

Google Sheets: Used to visualize and share some key findings with the team.

Insights:

Metric Spikes: There were several metrics that had spikes at certain times, probably due to external events or changes in operations.

Data Trends: By looking at the job performance data over time, I could see seasonal fluctuations in performance metrics, which helped me predict when spikes might occur.

Efficiency Opportunities: There were operations that consistently had low metrics, so there was room for improvement.

Case Study 1: Job Data Analysis

Worked 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).

QUERIES:

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

```sql
SELECT
    DATE(ds) AS review_date,
    HOUR(ds) AS review_hour,
    COUNT(DISTINCT job_id) AS jobs_reviewed
FROM
    job_data
WHERE
    MONTH(ds) = 11 AND YEAR(ds) = 2020
GROUP BY
    DATE(ds), HOUR(ds)
ORDER BY
    review_date, review_hour;
```

Result Grid | Filter Rows: | Export:

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

2. **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.**

```sql
SELECT
    ds,
    ROUND(SUM(events_per_second) OVER (
        ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
    ), 2) AS rolling_avg_throughput
FROM (
    SELECT
        DATE(ds) AS ds,
        COUNT(*) AS total_events,
        SUM(time_spent) AS total_time_spent,
        COUNT(*) / SUM(time_spent) AS events_per_second
    FROM
        job_data
    GROUP BY
        DATE(ds)
) daily_throughput;
```

ds	rolling_avg_throughput
2020-11-25	0.02
2020-11-26	0.04
2020-11-27	0.05
2020-11-28	0.11
2020-11-29	0.16
2020-11-30	0.21

I prefer using the 7-day rolling average for throughput as it smooths out daily fluctuations and provides a clearer view of trends over time. It reduces the impact of short-term anomalies or outliers, offering a more stable and reliable metric. This method helps identify underlying performance patterns, making it easier to detect issues or improvements. The daily metric, while more granular, can be too volatile and less effective for long-term analysis. Overall, the 7-day rolling average offers a better balance between sensitivity and stability in throughput analysis.

3. **Write an SQL query to calculate the percentage share of each language over the last 30 days.**

```
SELECT
    language,
    ROUND((COUNT(*) * 100.0) / SUM(COUNT(*)) OVER (), 2) AS language_percentage
FROM
    job_data
WHERE
    ds >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
GROUP BY
    language
ORDER BY
    language_percentage DESC;
```

Result Grid | Filter Rows:

language	language_percentage

4. **Write an SQL query to display duplicate rows from the job_data table.**

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

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

ds	job_id	actor_id	event	language	time_spent	org	duplicate_count

Toggle wrapping of cell content

Case Study 2: Investigating Metric Spike

Worked 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.

1. Write an SQL query to calculate the weekly user engagement.

```
SELECT
    DATE_FORMAT(occurred_at, '%Y-%u') AS event_week,
    COUNT(DISTINCT user_id) AS active_users
FROM
    events_1
GROUP BY
    event_week
ORDER BY
    event_week
LIMIT 1000;
```

Result Grid | Filter Rows: | E

event_week	active_users
NULL	774

2. Write an SQL query to calculate the user growth for the product.

```
SELECT
    DATE_FORMAT(created_at, '%Y-%u') AS signup_week,
    COUNT(user_id) AS new_users,
    SUM(COUNT(user_id)) OVER (ORDER BY DATE_FORMAT(created_at, '%Y-%u')) AS cumulative_users
FROM
    users
GROUP BY
    signup_week
ORDER BY
    signup_week;
```

Result Grid | Filter Rows: | Export: | Wr

signup_week	new_users	cumulative_users
NULL	9381	9381

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

```
WITH signup_and_events AS (
    SELECT
        u.user_id,
        DATE_FORMAT(u.created_at, '%Y-%u') AS signup_week,
        DATE_FORMAT(e.occurred_at, '%Y-%u') AS event_week
    FROM
        users u
    JOIN
        events_1 e
    ON
        u.user_id = e.user_id
)
SELECT
    signup_week,
    event_week,
    COUNT(DISTINCT user_id) AS retained_users,
    COUNT(DISTINCT user_id) * 100.0 /
        SUM(COUNT(DISTINCT user_id)) OVER (PARTITION BY signup_week) AS retention_rate
FROM
    signup_and_events
GROUP BY
FROM
    signup_and_events
GROUP BY
    signup_week, event_week
ORDER BY
    signup_week, event_week;
```

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

signup_week	event_week	retained_users	retention_rate
NULL	NULL	774	100.00000

4. **Write an SQL query to calculate the weekly engagement per device.**

```
SELECT
    DATE_FORMAT(occurred_at, '%Y-%u') AS event_week,
    device,
    COUNT(DISTINCT user_id) AS active_users_per_device
FROM
    events_1
GROUP BY
    event_week, device
ORDER BY
    event_week, device;
```

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

event_week	device	active_users_per_device
NULL	acer aspire desktop	13
NULL	acer aspire notebook	19
NULL	amazon fire phone	8
NULL	asus chromebook	22
NULL	dell inspiron desktop	19
NULL	dell inspiron notebook	40
NULL	hp pavilion desktop	22
NULL	htc one	16
NULL	ipad air	35
NULL	ipad mini	19
NULL	iphone 4s	27
NULL	iphone 5	71
NULL	iphone 5s	43
NULL	kindle fire	13
NULL	lenovo thinkpad	95
NULL	mac mini	6
NULL	macbook air	72
NULL	macbook pro	145

5. Write an SQL query to calculate the email engagement metrics.

```
SELECT
    DATE_FORMAT('week', occurred_at) AS email_week,
    COUNT(*) AS email_events_count
FROM
    email_events
GROUP BY
    email_week
ORDER BY
    email_week;
```

| Result Grid | Filter Rows: | Export:

email_week	email_events_count
NULL	5924

Result

This project has successfully identified key drivers of metric spikes, which can be used to anticipate future performance fluctuations and optimize operational strategies. The insights gathered have been instrumental in enhancing my understanding of job data patterns and improving decision-making in operations management.