Operation Analytics and Investigating Metric Spike

Project Description:

As a Data Analyst Lead at Microsoft, my primary responsibility is to perform Operation Analytics on the company's end-to-end operations. This involves closely working with various departments such as operations, support, marketing, etc., to derive insights from the data they collect.

My role also includes investigating metric spikes, identifying areas for improvement, and predicting the overall growth or decline of the company. To do this, I analyze various data sets and tables and provide insights to different departments based on their specific questions.

Overall, my objective is to help Microsoft achieve better automation, streamline workflows, and enhance cross-functional team communication by providing actionable insights based on data analysis.

Approach:

First, I learnt the concepts of windowing functions and their uses, and how to use them to our advantage from the recorded videos. The project was very challenging and at the same time I had fun learning different concepts and implementing them. I read the problem statements clearly before writing the query in-order to properly understand what are the necessary insights that should be given. Since I am a beginner, I faced lots of issues with errors, but I made sure to watch the videos again to get a better understanding to implement the work in a right way.

Tech-Stack used:

I used SQL queries to give the insights for the problems that were assigned. I used PopSQL

throughout this project in-order to easily visualize the data and to write queries with ease. It was

very helpful and user friendly by changing according to user preferences.

Insights:

A) Case study 1 (Job Data):

1) Number of jobs reviewed:

The task was to calculate the number of jobs reviewed per hour per day for November 2020. The SQL query that I used to give the output for above requirement is given below:

```
SELECT
   DATE_FORMAT(ds, '%Y-%m-%d %H:00') AS date_hour,
   COUNT(*) AS job_count,
   COUNT(*) / COUNT(DISTINCT HOUR(ds)) AS job_count_per_hour
FROM
```

```
    langtable
WHERE
    ds BETWEEN '2020-11-01' AND '2020-11-30'
    AND event = 'decision'
GROUP BY
    DATE_FORMAT(ds, '%Y-%m-%d %H:00'),
    DATE(ds)
ORDER BY
    DATE(ds) ASC
LIMIT
    1000;
```

2) Calculating throughput:

The task was to find the 7-day rolling average of throughput. Regarding the preference of daily metric or 7-day rolling for throughput, it depends on the use case and the level of granularity required for the analysis. If we want to understand the daily fluctuations in throughput, then the daily metric would be more appropriate. However, if we are more interested in the overall trend and want to smooth out any daily variations, then the 7-day rolling average would be more useful. The SQL query that I used to give the output for above requirement is given below:

```
SELECT ds, COUNT(event) as events_per_day
FROM langtable
GROUP BY ds;
SELECT ds, AVG(events_per_day) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling_avg_throughput
FROM (
  SELECT ds, COUNT(event) as events_per_day
  FROM langtable
  GROUP BY ds
) t;
```

3) Percentage share of each language:

The task was to calculate the percentage share of each language in the last 30 days. The SQL query that I used to give the output for above requirement is given below:

```
SELECT language,
    event,
    COUNT(*) AS job_count,
    ROUND(COUNT(*) / SUM(COUNT(*)) OVER(PARTITION BY language) * 100, 2) AS
percentage_share
FROM langtable
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY language, event
ORDER BY language, event;
```

4) Duplicate rows:

The task was to display the duplicate data from the table. The SQL query that I used to give the output for above requirement is given below:

```
SELECT ds, job_id, actor_id, event, language, time_spent, org, COUNT(*) as count
FROM langtable
GROUP BY ds, job_id, actor_id, event, language, time_spent, org
HAVING COUNT(*) > 1;
```

B) Case study 2 (Investigating metric spike):

1) User engagement:

The task was to find the weekly user engagement by seeing their activeness. The SQL query that I used to give the output for above requirement is given below:

```
SELECT
  'events' AS source,
  event_type AS event,
  COUNT(*) AS count
FROM
  events
GROUP BY
  event_type
UNION ALL
SELECT
```

```
    'email_events' AS source,
    action AS event,
    COUNT(*) AS count
FROM
    email_events
GROUP BY
    action;
```

2) User growth:

The task was to calculate the user growth for the product. The SQL query that I used to give the output for above requirement is given below:

```
SELECT
    DATE_FORMAT(created_at, '%Y-%m') AS month,
    COUNT(DISTINCT user_id) AS user_count
FROM
    users
GROUP BY
    month
ORDER BY
    month ASC;
```

3) Weekly retention:

The task was to find out the weekly retention of users-sign up cohort after signing up for the product. The SQL query that I used to give the output for above requirement is given below:

```
SELECT
  DATE_FORMAT(created_at, '%x-%v') AS cohort_day,
  DATE_FORMAT(events.occurred_at, '%x-%v') AS event_day,
  COUNT(DISTINCT users.user_id) AS total_users,
  COUNT(DISTINCT events.user_id) AS retained_users,
  COUNT(DISTINCT events.user_id) / COUNT(DISTINCT users.user_id) AS retention_rate
FROM users
JOIN (
```

```
  SELECT user_id, MIN(occurred_at) AS first_event
  FROM events
  WHERE event_type = 'engagement'
  GROUP BY user_id
) AS first_events
  ON users.user_id = first_events.user_id
JOIN events
  ON events.user_id = users.user_id
  AND events.occurred_at >= first_events.first_event
WHERE events.event_type = 'engagement'
GROUP BY 1, 2
ORDER BY 1, 2
LIMIT 1000;
```

4) Weekly engagement:

The task was to find the weekly engagement of the users by activeness of the devices. The SQL query that I used to give the output for above requirement is given below:

```
SELECT
  DATE_FORMAT(events.occurred_at, '%x-%v') AS week,
  events.device,
  COUNT(DISTINCT events.user_id) AS weekly_engaged_users
FROM events
JOIN users ON events.user_id = users.user_id
WHERE events.event_type = 'engagement'
GROUP BY 1, 2
ORDER BY 1, 2
LIMIT 1000;
```

5) Email engagement:

The task was to calculate the email engagement metrics. The SQL query that I used to give the output for above requirement is given below:

```
SELECT
```

```
    'email_events' AS source,
    action AS event,
    COUNT(*) AS count
FROM
    email_events
GROUP BY
    action;
```

Result:

This project has helped me a lot in understanding the advanced SQL concepts and the art of writing queries for fetching minute details from huge databases. This project taught me to decode the errors that I get without getting frustrated and made me fall in love with data analysis. I realized the importance of data analysis and the demand for it out there in the world. I would like to thank the **Trainity** team for having provided this kind of learning in which the project and implementing is giving more importance. I look forward to learning many things out of this learning journey.