Operation Analytics and Investigating Metric Spike

Project Description:

Case Study 1 (Job Data):

This project focuses on analyzing a job_data table to answer specific questions using SQL fundamentals. The table contains information about jobs, including job_id, actor_id, event, language, time_spent, org, and ds columns. The questions include calculating the number of jobs reviewed per hour per day, analyzing throughput, determining the percentage share of each language, and identifying duplicate rows.

Case Study 2 (Investigating Metric Spike):

This project involves analyzing three tables: users, events, and email_events. The goal is to calculate metrics related to user engagement, user growth, weekly retention, weekly engagement per device, and email engagement. SQL fundamentals will be used to extract insights from the dataset and provide valuable information for marketing campaigns and decision-making.

Approach:

As an individual working on this project, I followed a structured approach to analyze the job_data table. I began by understanding the table structure and column definitions. Using SQL queries and functions, I calculated the number of jobs reviewed per hour per day for November 2020, analyzed throughput by calculating events per second and 7-day rolling average, determined the percentage share of each language in the last 30 days, and identified duplicate rows. I prioritized data accuracy, optimized queries for efficiency, and maintained documentation of my workflow. The project aimed to provide valuable insights for marketing and investor assessments, achieved through the successful application of SQL fundamentals.

Tech-Stack Used:

For this project, I utilized MySQL Workbench 8.0 as the primary software tool. MySQL Workbench is an integrated development environment (IDE) for MySQL databases, providing a graphical interface for designing, querying, and managing databases.

Insights:

Case Study 1(Job Data):

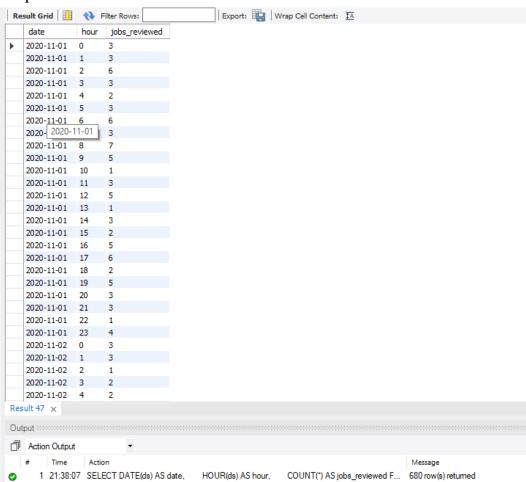
Number of jobs reviewed: The number of jobs reviewed per day per hour indicates the level of activity in job review processes.

To calculate the number of jobs reviewed per hour per day for November 2020, I utilized SQL queries to filter the data for the specified time period and grouped the results by hour and day.

Query:

SELECT DATE(ds) AS date,
HOUR(ds) AS hour,
COUNT(*) AS jobs_reviewed
FROM sheet1
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY DATE(ds), HOUR(ds)

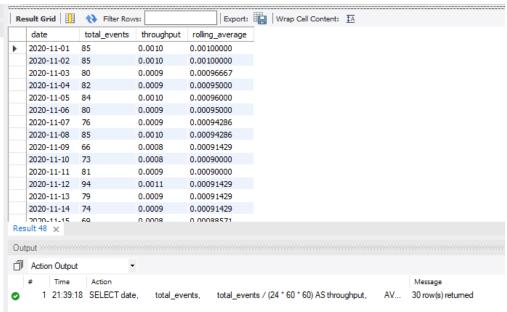
ORDER BY DATE(ds), HOUR(ds);



Throughput: Throughput refers to the number of events happening per second, representing the system's processing capacity.

To calculate the 7-day rolling average of throughput, I used SQL queries to aggregate the events per second and then calculated the average over a rolling window of 7 days. This helps identify any trends or variations in the throughput metric. I prefer the 7-day rolling average because it provides a smoother representation of the metric, reducing the impact of daily fluctuations and offering a more comprehensive view of the system's performance.

```
Query:
SELECT date,
total_events,
total_events / (24 * 60 * 60) AS throughput,
AVG(total_events / (24 * 60 * 60)) OVER (ORDER BY date ROWS BETWEEN
6 PRECEDING AND CURRENT ROW) AS rolling_average
FROM (
SELECT DATE(ds) AS date, COUNT(*) AS total_events
FROM sheet1
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY DATE(ds)
) AS subquery
GROUP BY date, total_events
ORDER BY date;
```



Percentage share of each language: The percentage share of each language in different contents provides insights into language preferences and content distribution. To calculate the percentage share of each language in the last 30 days, I used SQL queries to filter the data for the specified time period and performed calculations to determine the language distribution. By dividing the count of each language by the total count of contents, I obtained the percentage share for each language.

Query:

SELECT language,

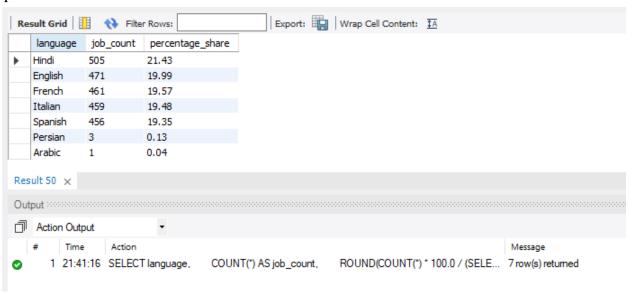
COUNT(*) AS job_count,

ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM sheet1 WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'), 2) AS percentage_share FROM sheet1

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

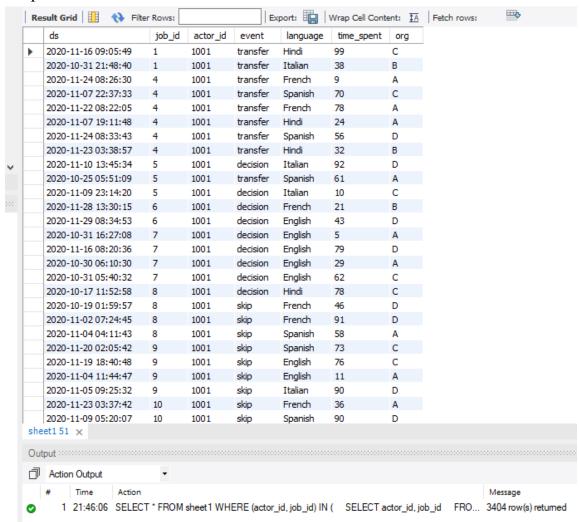
GROUP BY language

ORDER BY percentage share DESC;



Duplicate rows: Duplicate rows refer to rows in the dataset that have the same values. If actor_id and job_id are the same for an entry, then that entry is considered to be duplicate. Comparing the values across columns, I can identify rows with identical values and retrieve them from the table.

```
Query:
SELECT *
FROM sheet1
WHERE (actor_id, job_id) IN (
SELECT actor_id, job_id
FROM sheet1
GROUP BY actor_id, job_id
HAVING COUNT(*) > 1
)
ORDER BY actor_id, job_id;
```



Case Study 2 (Investigating Metric Spike):

User Engagement: User engagement is a measure of how active users are and indicates their satisfaction with a product or service.

To calculate the weekly user engagement, I utilized SQL queries to analyze unique user engagement events within the specified week.

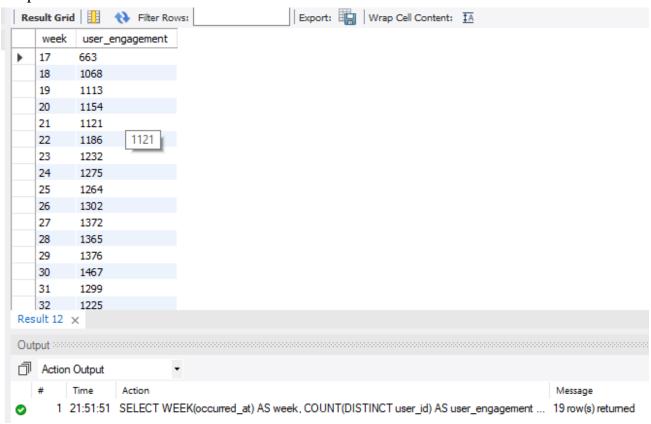
Query:

SELECT WEEK(occurred_at) AS week, COUNT(DISTINCT user_id) AS user_engagement

FROM events

WHERE event_type = 'engagement'

GROUP BY week;



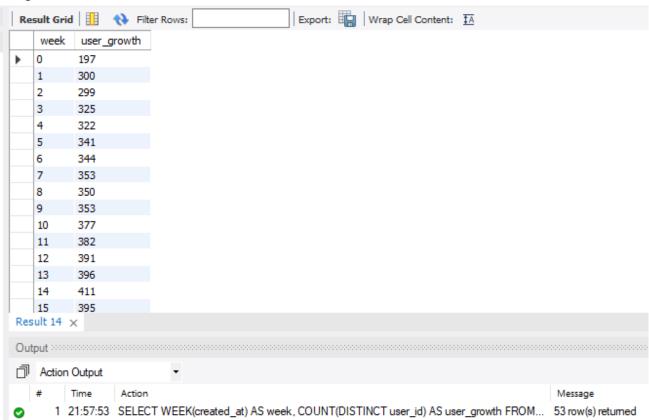
User Growth: User growth measures the increase in the number of users over a specific period, reflecting the product's adoption and popularity.

To calculate user growth for a product, I used SQL queries to track the number of new users added over time. By comparing the count of users added in different weeks, we can identify growth of product.

Query:

SELECT WEEK(created_at) AS week, COUNT(DISTINCT user_id) AS user_growth FROM user

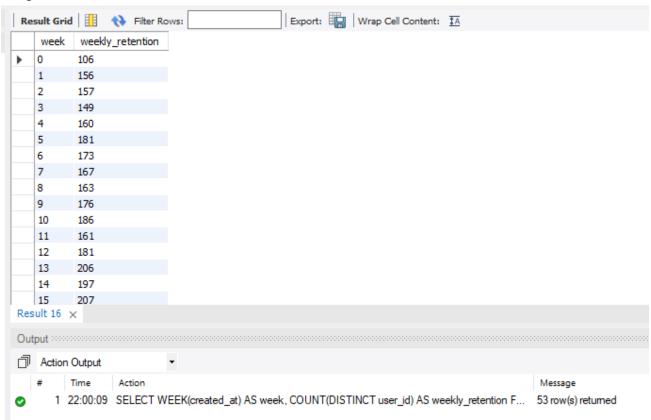
GROUP BY week;



Weekly Retention: Weekly retention evaluates the percentage of users who continue to use a product or service after signing up, indicating its ability to retain users. To calculate the weekly retention of users, I employed SQL queries to track the state of users and identify users who remained active in consecutive weeks. By comparing the count of retained users to the initial sign-up cohort, I determined the retention rate for each week.

Query:

SELECT WEEK(created_at) AS week, COUNT(DISTINCT user_id) AS weekly_retention
FROM user
WHERE state = 'active'
GROUP BY week;



Weekly Engagement: Weekly engagement measures the level of user activity and satisfaction with a product or service on a weekly basis.

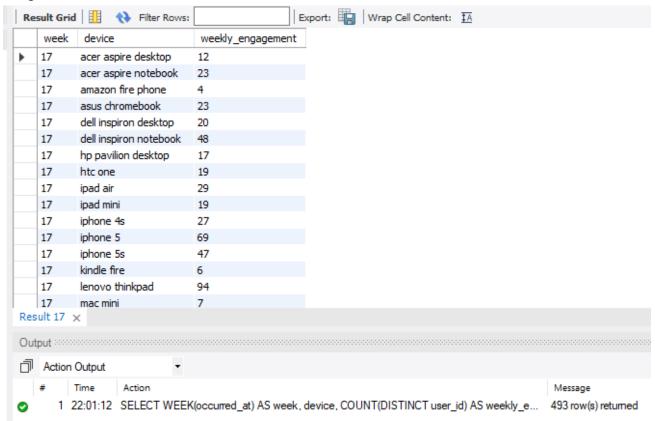
To calculate the weekly engagement per device, I utilized SQL queries to analyze user interactions and activities categorized by device type. By selecting distinct users and their devices and grouping by week and device, I assessed the level of engagement for each device category.

Query:

SELECT WEEK(occurred_at) AS week, device, COUNT(DISTINCT user_id) AS weekly engagement

FROM events

GROUP BY week, device;

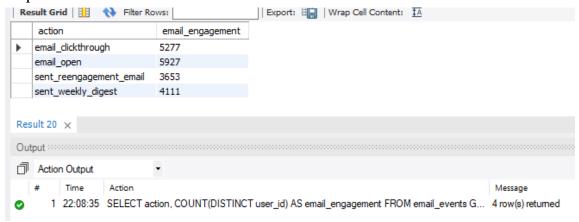


Email Engagement: Email engagement reflects user involvement and interaction with the email service.

To calculate email engagement metrics, I used SQL queries to analyze user email activities such as opens, clicks, and responses.

Query:

SELECT action, COUNT(DISTINCT user_id) AS email_engagement FROM email_events GROUP BY action;



Results:

While working on this project, I have gained a better understanding of user analytics and SQL fundamentals. By analyzing Operation Analytics and Investigating Metric Spike, I was able to provide insights on various aspects such as number of jobs analyzed per hour per day, 7 day rolling average of throughput, percentage share of each language, user engagement, user growth, weekly retention, weekly engagement per device, and email engagement.

This project has helped me enhance my SQL skills, particularly in querying and manipulating data to derive meaningful insights. It has also improved my ability to interpret data and provide actionable recommendations based on the analysis. Overall, this project has deepened my understanding of user behavior analysis and its application in making informed decisions for product development and marketing strategies.