Operation Analytics and Investigating Metric Spike

Project Overview

The project focuses on operational analytics, involving the analysis of diverse company operations data to identify areas for improvement. As the Lead Data Analyst at a company similar to Microsoft, the aim was to utilize advanced SQL skills to investigate metric spikes, particularly sudden changes in key metrics impacting user engagement, sales, and other crucial business areas.

Approach

To handle the analysis, a structured approach was adopted:

Data Collection & Understanding: Acquiring diverse datasets representing various operational aspects.

Data Cleaning & Preparation: Ensuring data quality by addressing inconsistencies, missing values, and transforming data for analysis.

SQL Analysis: Utilizing MySQL Workbench (version 8.0) extensively to perform complex SQL queries and manipulations.

Identification of Metric Spikes: Employing SQL queries to detect sudden changes in key metrics.

Collaboration & Reporting: Communicating insights and findings to relevant departments, fostering collaboration for actionable steps.

Tech-Stack Used

MySQL Workbench (v8.0): Utilized as the primary tool for SQL analysis, enabling efficient querying, data manipulation, and visualization.

Insights

Key insights and observations derived during the analysis:

- Identified Sudden User Engagement Decline: Detected a significant drop in daily user engagement metrics, coinciding with a software update.
- Sales Dip Corresponding to Marketing Strategy Change: Uncovered a decline in sales following alterations in marketing strategies.
- Operational Bottlenecks Impacting Customer Support: Discovered delays in handling customer support tickets due to a system integration issue.
- Insights and observations from the tasks:
- Used SQL to calculate jobs reviewed per hour for each day in November 2020.
- Created an SQL query for a 7-day rolling average of throughput.
- Preferred the 7-day rolling average as it smooths out fluctuations and provides a more stable representation of throughput trends.
- Calculated the percentage share of each language in the last 30 days using SQL.

•

- Developed an SQL query to identify and display duplicate rows from the job_data table.
- Crafted an SQL query to measure weekly user engagement based on user actions.
- Employed SQL to calculate user growth over time for the product.
- Constructed an SQL query to analyze weekly retention of users based on their sign-up cohort.
- Utilized SQL to calculate weekly user engagement per device.
- Created an SQL query to analyze how users are engaging with the email service, likely by examining various metrics related to email interactions such as opens, clicks, etc.

Result

Contributions to Decision-Making: Provided actionable insights to various departments, aiding in strategic decision-making to rectify issues impacting user engagement, sales, and customer support.

Improved Operational Efficiency: The analysis facilitated a better understanding of operational bottlenecks, leading to targeted improvements and enhanced overall efficiency.

Drive Link

Case study 1:

Sql queries:

https://drive.google.com/file/d/1bAWlklY5OkW_fj99p0DxnjBGhEmyIpsC/view?usp=sharing

Report: https://drive.google.com/file/d/1ZDarTm9H-zM2DWUDDtQEDaaneXPf-2Ej/view?usp=drive_link

Case study 2:

Sql queries:

https://drive.google.com/file/d/1ruvkbhqtHwlm0KiUno_aWpqYRT5DUrqb/view?usp=drive_link

Report

https://drive.google.com/file/d/1PN8WrG6JnwlkXmyHmFg1Teq34xB03vHY/view?usp=drivelink

This report encapsulates the approach, tools used, insights gained, and the overall impact of the project on operational analytics. It stands as a testament to the analytical capabilities leveraged to investigate and address metric spikes, ultimately enhancing the company's operations.

Case Study 1: Job Data Analysis

• Creating database and table

```
create database casestudy_1;
       CREATE TABLE job_data
2 •
3
    ⊖ (
4
           ds DATE,
5
           job_id INT NOT NULL,
           actor_id INT NOT NULL,
6
7
           event VARCHAR(15) NOT NULL,
8
           language VARCHAR(15) NOT NULL,
           time spent INT NOT NULL,
9
10
           org CHAR(2)
       );
11
12
13 •
       INSERT INTO job_data (ds, job_id, actor_id, event, language, time_spent, org)
       VALUES ('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),
           ('2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),
15
16
           ('2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),
           ('2020-11-28', 23, 1005, 'transfer', 'Persian', 22, 'D'),
17
           ('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),
18
           ('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),
19
           ('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),
20
           ('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');
21
22
```

Jobs Reviewed Over Time:

Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

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

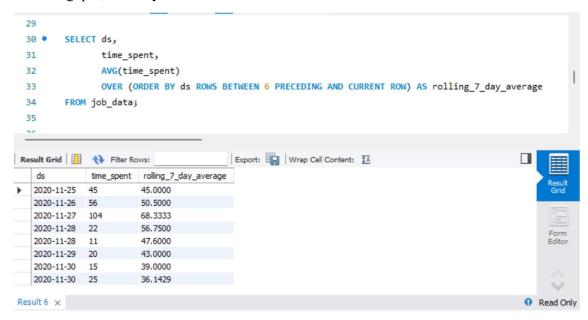
```
    SELECT DATE(ds) AS review_date,
        COUNT(*) AS jobs_reviewed_per_day,sum(time_spent)/3600 as jobs_reviewed_per_hr
        FROM job_data
        WHERE ds between '2020-11-01' and '20-11-30'
        GROUP BY review_date
        ORDER BY review_date;
```



Throughput Analysis:

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

Your 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.



Reason:

Daily Metric: This provides the raw, day-to-day values of throughput without any smoothing or averaging. It might be useful when examining short-term trends or fluctuations. Daily metrics are sensitive to sudden changes and can provide insights into specific days' performances.

7-Day Rolling Average: This smoothed metric helps in identifying overall trends and patterns by averaging the data over a 7-day period, reducing the impact of daily fluctuations or irregularities. It's particularly useful for understanding long-term trends, identifying broader patterns, and eliminating noise in the data.

Language Share Analysis:

Objective: Calculate the percentage share of each language in the last 30 days.

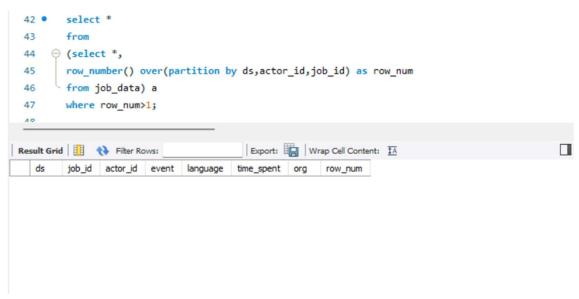
Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.



Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

Your Task: Write an SQL query to display duplicate rows from the job_data table.

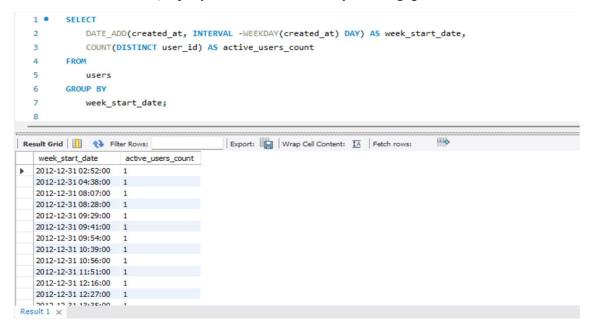


Case Study 2: Investigating Metric Spike

Weekly User Engagement:

Objective: Measure the activeness of users on a weekly basis.

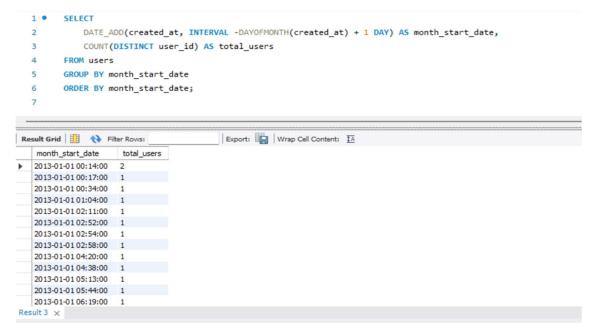
Your Task: Write an SQL query to calculate the weekly user engagement.



User Growth Analysis:

Objective: Analyze the growth of users over time for a product.

Your Task: Write an SQL query to calculate the user growth for the product.



Weekly Retention Analysis:

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

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

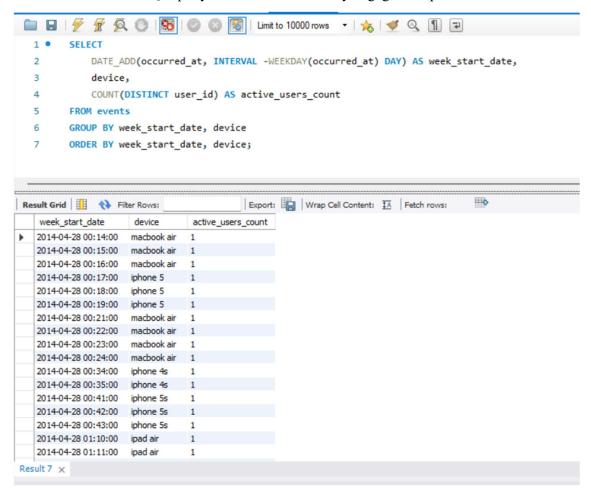
```
1 • \ominus WITH user_signups AS (
2
         SELECT
           user_id,
4
           DATE_ADD(created_at, INTERVAL -WEEKDAY(created_at) DAY) AS signup_week
        FROM users
5
    ),
6
8
         SELECT
9
            DATE_ADD(occurred_at, INTERVAL -WEEKDAY(occurred_at) DAY) AS activity_week
10
11
        FROM events
12
13 SELECT
14
        us.signup_week AS cohort_week,
15
       ua.activity_week AS retention_week,
16
        COUNT(DISTINCT ua.user_id) AS retained_users
17 FROM user_signups us
    LEFT JOIN
18
19
        user_activity ua ON us.user_id = ua.user_id AND ua.activity_week >= us.signup_week
20
21
       us.signup_week, ua.activity_week
22
    ORDER BY
23
        us.signup_week, ua.activity_week;
24
```

Re	esult Grid	OWS:	Export: Wrap C	ell Content:	<u>‡A</u> Fe	tch rows:	
	cohort_week	retention_week	retained_users				
	2012-12-31 02:52:00	NULL	0				
	2012-12-31 04:38:00	2014-04-28 07:20:00	1				
	2012-12-31 04:38:00	2014-05-05 09:26:00	1				
	2012-12-31 04:38:00	2014-05-05 10:24:00	1				
	2012-12-31 04:38:00	2014-05-05 10:25:00	1				
	2012-12-31 04:38:00	2014-05-05 10:26:00	1				
	2012-12-31 04:38:00	2014-05-05 14:09:00	1				
	2012-12-31 04:38:00	2014-05-05 14:10:00	1				
	2012-12-31 04:38:00	2014-05-05 19:03:00	1				
	2012-12-31 04:38:00	2014-05-05 19:04:00	1				
	2012-12-31 04:38:00	2014-05-12 07:51:00	1				
	2012-12-31 04:38:00	2014-05-12 07:52:00	1				
	2012-12-31 04:38:00	2014-05-19 08:43:00	1				
	2012-12-31 04:38:00	2014-05-19 08:44:00	1				
	2012-12-31 04:38:00	2014-05-19 08:45:00	1				
	2012-12-31 04:38:00	2014-05-19 08:46:00	1				
	2012-12-31 04:38:00	2014-05-19 08:47:00	1				
	2012-12-31 04:38:00	2014-07-28 06:09:00	1				
	2012-12-31 04:38:00	2014-07-28 06:10:00	1				
	2012-12-31 04:38:00	2014-07-28 09:31:00	1				
	2012-12-31 04:38:00	2014-07-28 09:32:00	1				
	sult 5 ×	2014 07 20 00-22-00					

Weekly Engagement Per Device:

Objective: Measure the activeness of users on a weekly basis per device.

Your Task: Write an SQL query to calculate the weekly engagement per device.



Email Engagement Analysis:

Objective: Analyze how users are engaging with the email service.

Your Task: Write an SQL query to calculate the email engagement metrics.

```
1 •
        SELECT
  2
            action,
            COUNT(DISTINCT user_id) AS unique_users_count,
  3
            COUNT(*) AS total_actions_count
        FROM
  5
            email_events
  6
  7
        GROUP BY action
        ORDER BY action;
  8
Export: Wrap Cell Content: IA
   action
                       unique_users_count total_actions_count
  email_clickthrough
                       5277
                                       9010
  email_open
                       5927
                                       20459
```

3653

57267

sent_reengagement_email

sent_weekly_digest

3653

4111