Operation Analytics and Investigating Metric Spike

By Pratiksha Shetty

Project Description

In the "Operational Analytics and Investigating Metric Spike" project, as a Lead Data Analyst, I will use advanced SQL skills to analyse diverse datasets, collaborating with teams like operations and marketing. Focused on end-to-end operations, the project emphasizes on identifying improvement areas through daily investigation of metric spikes. My role involves interpreting sudden changes in vital metrics, addressing queries from various departments, and delivering insights crucial for enhancing company operations. This project mirrors real-world scenarios, allowing us to provide valuable insights that contribute to strategic decision-making and continuous improvement within the organization.

Approach

Initially, I dedicated my time to understand the given data/table. I addressed questions, clarifying the meanings of fields like job_id, actor_id, and event, and considered essential aspects for data review. Using SQL, I extracted diverse insights from the management team's provided dataset. I initiated by establishing a database named "operation_analytics" and subsequently created tables based on the team's provided structure and connections. The ensuing analysis aimed to produce valuable insights, contributing to the company's understanding and strategic decision-making.

Analysis:

Case Study 1: Job Data Analysis

A) **Jobs Reviewed Over Time:** Calculate the number of jobs reviewed per hour for each day of November 2020

Query:

```
SELECT
   COUNT(DISTINCT job_id) / (30 * 24) AS num_jobs_reviewed
FROM
   job_data
WHERE
   ds BETWEEN '2020-11-01' AND '2020-11-30';
```

Output:

num_jobs_reviewed
0.0083

The above query calculates the average number of jobs reviewed per hour during the November 2020. It divides the count of distinct job IDs by the total hours in the given month (30 days * 24 hours/day).

B) **Throughput Analysis:** Calculate the 7-day rolling average of throughput (number of events per second)

Query:

```
with CTE as (
            select ds, count(job_id) as num_jobs, sum(time_spent) as total_time
            from job_data
            where event NOT IN ('skip')
            AND ds between '2020-11-01' AND '2020-11-30'
            GROUP BY ds
   )
select ds,
```

```
ROUND(SUM(num_jobs) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW)
/ SUM(total_time) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT
ROW),2) AS throughput_7d
FROM CTE;
```

Output:

ds	throughput_7d
2020-11-25	0.02
2020-11-27	0.01
2020-11-28	0.02
2020-11-29	0.02
2020-11-30	0.03

I prefer using the weekly metrics to compare the number of events as it evens out daily fluctuations, creating a more consistent metric. This approach helps in recognizing trends over a long period, facilitating the discrepancy of genuine changes from short-term variations.

C) **Language Share Analysis:** Calculate the percentage share of each language in the last 30 days

```
select language,
count(language) as "Language Count",
count(*)*100/sum(count(*))
over() as percentage
from job_data
group by language
order by percentage desc;
```

Output:

language	Language Count	percentage
Persian	3	37.5000
English	1	12.5000
Arabic	1	12.5000
Hindi	1	12.5000
French	1	12.5000
Italian	1	12.5000

D) **Duplicate Rows Detection:** Identify duplicate rows in the data

```
WITH cte AS
(
   SELECT *, ROW_NUMBER() OVER (PARTITION BY job_id) AS row_num
   FROM job_data
)
SELECT * FROM cte WHERE row_num > 1;
```

Output:

ds	job_id	actor_id	event	language	time_spent	org	row_num
2020-11-28	23	1005	transfer	Persian	22	D	2
2020-11-26	23	1004	skip	Persian	56	A	3

Case Study 2: Investigating Metric Spike

A) **Weekly User Engagement:**

```
select week(occurred_at) as week_num,
count(distinct user_id) as user_count
from events where event_type='Engagement'
```

```
group by week_num
order by week_num;
```

Output:

week_num	user_count
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

B) User Growth Analysis: Analyse the growth of users over time for a product

```
WITH weekly_user_counts AS (
  SELECT
   EXTRACT(year FROM created_at) AS year,
     week(created_at) AS week_num, state, user_id,
       COUNT(DISTINCT user_id) AS user_count
  FROM users
  -- WHERE state = "active"
  GROUP BY year, week_num,state, user_id
)
SELECT year, week_num, user_count, SUM(user_count) OVER (ORDER BY year, week_num) AS
cumulative_users
FROM weekly_user_counts
ORDER BY year, week_num;
```

Output:

Weekly_retention Query.csv

C) Weekly Retention Analysis: Analyse the retention of users on a weekly basis after signing up for a product

```
SELECT COUNT (user_id) AS total_users, SUM (CASE WHEN retention_week = 1 THEN 1 ELSE 0
END) AS per_week_retention
FROM (
   SELECT a.user_id, a.sign_up_week, b.engagement_week, b.engagement_week - a.sign_up_week
AS retention_week
   FROM (
      (SELECT DISTINCT user_id, EXTRACT (week FROM occurred_at) AS sign_up_week
       FROM events
```

```
        WHERE event_type = 'signup_flow' AND event_name = 'complete_signup' AND
EXTRACT(week FROM occurred_at) = 18) a
        LEFT JOIN
        (SELECT DISTINCT user_id, EXTRACT(week FROM occurred_at) AS engagement_week
         FROM events
         WHERE event_type = 'engagement') b
        ON a.user_id = b.user_id
    )
    GROUP BY a.user_id, a.sign_up_week, b.engagement_week
    ORDER BY a.user_id, a.sign_up_week
) subquery;
```

Output:

total_users	per_week_retention
615	114

D) Weekly Engagement Per Device: Measure the activeness of users on a weekly basis per device

```
select
extract(year from occurred_at) as year_num,
extract(week from occurred_at) as week_num,
device,
count(distinct user_id) as no_of_users
from events
where event_type = 'Engagement'
group by year_num,week_num,device
order by year_num,week_num,device;
```

Output:

Weekly_Eng_Per_Device.csv

E) Email Engagement Analysis:

```
with CTE as (
    SELECT *,
    CASE
        WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email_sent'
        WHEN action IN ('email_open') THEN 'email_opened'
        WHEN action IN ('email_clickthrough') THEN 'email_clicked'
    END AS email_cat
    FROM email_events
)  select 100.0 * SUM(email_cat = 'email_opened') / SUM(email_cat = 'email_sent') AS
email_opening_rate,
    100.0 * SUM(email_cat = 'email_clicked') / SUM(email_cat = 'email_sent') AS email_clicking_rate
FROM CTE;
```

Output:

email_opening_rate	email_clicking_rate
33.58339	14.78989

Tech-Stack Used

As advised, I have used MySQL Workbench 8.0 for my project to create the databases and perform all the analysis.

Insights

Case Study 1 (Job Data):

1) In November 2020, 0.83% of unique jobs were reviewed per day
2) The weekly metrics to compare the number of events was better compare to daily as it evens out daily fluctuations, creating a more consistent metric
3) The percentage of Persian Language was the most i.e. 37.5%
4) We see there were 2 duplicate rows when we partitioned it based on job_id

Case Study 2 (Metrics Spike Investigation)

1) The weekly user engagement inclined between week 17 to 30 and there was a drop after 30th week suggesting the product quality reduce in those weeks
2) The total number of users who joined in 18th Week was 615 & Weekly retention was 114 users
3) Engagement based on Devices are highest among MacBook & iPhone users
4) The Email Opening rate is 33.5 % and the Email clicking rate is 14.7% showing that there is a positive user engagement across the platform

Results

Throughout this project, I gained proficiency in applying advanced SQL concepts, including Windows Functions, and deepened my understanding of real-world industry practices. This experience significantly contributed to mastering my SQL skills. Learning to pose insightful questions tailored to specific circumstances became a key takeaway. I learnt the ability to discern which columns are essential for deriving valuable insights, ultimately aiding business growth. Exploring how companies identify areas for improvement was helpful. Additionally, exploring the metric spikes provided valuable insights into understanding both the surges and declines in metrics.

Drive Link

SQL Query Link (Case Study 1) : Job_Data_Query

SQL Query Link (Case Study 2) : Metrics_Spike_Query