Operation And Metric Analytics

By
Moksh Jaiswal

Project Description:

This project explores the concept of Operation Analytics, which encompasses the comprehensive analysis of a company's complete end-to-end operations. The task of investigating metric spike is a daily endeavour that revels the reasons behind variations in business performance. Using Operation Analytics, the following tasks were achieved:

Case Study 1: Job data analysis

Tasks:

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

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

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

D. **Duplicate Rows Detection:**
   - Objective: Identify duplicate rows in the data.

Case Study 2: Investigating metric spike

Tasks:

A. **Weekly User Engagement:**
   - Objective: Measure the activeness of users on a weekly basis.

B. **User Growth Analysis:**
   - Objective: Analyse the growth of users over time for a product.

C. **Weekly Retention Analysis:**
   - Objective: Analyse the retention of users on a weekly basis after signing up for a product.

D. **Weekly Engagement Per Device:**
   - Objective: Measure the activeness of users on a weekly basis per device.

E. **Email Engagement Analysis:**
   - Objective: Analyse how users are engaging with the email service.

Approach:

The approach towards the project is kept simple. Certain tables were created to import the datasets and were later altered as per suitability using SQL. By means of these tables, several queries were executed to generate the required results. The SQL Queries are given at the end.

Tech-Stack Used:

MySQL installer community 8.0.30.0.msi was used for this project and it contains: -MySQL Workbench 8.0 CE -MySQL Shell -MySQL Command Line Client.

Insights:

Case Study 1: Job data analysis

1. The number of jobs reviewed per hour for each day in November 2020 are:

ds	jobs_reviewed_per_day	hours_spent_reviewing
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

2. The 7-day rolling average of throughput (number of events per second) is:

ds	rolling_7D_average
2020-11-25	0.02
2020-11-26	0.02
2020-11-27	0.01
2020-11-28	0.02
2020-11-29	0.02
2020-11-30	0.03

- I personally believe that the choice between the daily metric and the 7-day rolling average depends on the objectives of your analysis and the specific needs of your organization:
- If one needs to make quick decisions based on recent changes, daily metrics are more suitable, as they provide real-time data.
- If your focus is on identifying long-term trends and making strategic decisions, the 7-day rolling average is better suited, as it offers a more stable and insightful perspective.
- A combination of both approaches could be valuable. Daily metrics can be used for short-term monitoring and immediate actions, while the rolling average can provide an extensive perspective for understanding overall performance trends.

3. The percentage share of each language in the last 30 days is:

Language	Percentage
English	12.50
Arabic	12.50
Persian	37.50
Hindi	12.50
French	12.50
Italian	12.50

4. The duplicate rows in the data are:

ds	job_id	actor_id	event	language	Time_spent	org	RowNumber
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

Tasks:

1. The activeness of users on a weekly basis.

Week_Num	Active_Users
17	887
18	1985
19	2030
20	2093
21	1986
23	2188
22	2157

2. The growth of users over time for a product.

month	user_count	next_month_users
1	712	685
2	685	765
3	765	907
4	907	993
5	993	1086
6	1086	1281
7	1281	1347
8	1347	330
9	330	390
10	390	399
11	399	486
12	486	

3. The retention of users on a weekly basis after signing up for a product.

signup_week	event_week	cohort_size	retention_count	retention_rate
17	17	663	1	0.15
17	18	472	1	0.21
17	19	324	1	0.31

17	20	251	1	0.40
17	21	205	1	0.49
17	22	187	1	0.53
17	23	167	1	0.60
17	24	146	1	0.68
18	18	596	1	0.17
18	19	362	1	0.28
18	20	261	1	0.38
18	21	203	1	0.49
18	22	168	1	0.60
18	23	147	1	0.68
18	24	144	1	0.69
18	25	127	1	0.79
19	19	427	1	0.23
19	20	284	1	0.35
19	21	173	1	0.58
19	22	153	1	0.65
19	23	114	1	0.88
19	24	95	1	1.05
19	25	91	1	1.10
19	26	81	1	1.23
20	20	358	1	0.28
20	21	223	1	0.45
20	22	165	1	0.61
20	23	121	1	0.83
20	24	91	1	1.10
20	25	72	1	1.39
20	26	63	1	1.59
20	27	67	1	1.49
21	21	317	1	0.32
21	22	187	1	0.53
21	23	131	1	0.76
21	24	91	1	1.10
21	25	74	1	1.35
21	26	63	1	1.59
21	27	75	1	1.33
21	28	72	1	1.39
22	22	326	1	0.31
22	23	224	1	0.45
22	24	150	1	0.67
22	25	107	1	0.93
22	26	87	1	1.15
22	27	73	1	1.37
22	28	63	1	1.59
22	29	60	1	1.67

23	23	328	1	0.30
23	24	219	1	0.46
23	25	138	1	0.72
23	26	101	1	0.99
23	27	90	1	1.11
23	28	79	1	1.27
23	29	69	1	1.45
23	30	61	1	1.64
24	24	339	1	0.29
24	25	205	1	0.49
24	26	143	1	0.70
24	27	102	1	0.98
24	28	81	1	1.23
24	29	63	1	1.59
24	30	65	1	1.54
24	31	61	1	1.64
25	25	305	1	0.33
25	26	218	1	0.46
25	27	139	1	0.72
25	28	101	1	0.99
25	29	75	1	1.33
25	30	63	1	1.59
25	31	50	1	2.00
25	32	46	1	2.17
26	26	288	1	0.35
26	27	181	1	0.55
26	28	114	1	0.88
26	29	83	1	1.20
26	30	73	1	1.37
26	31	55	1	1.82
26	32	47	1	2.13
26	33	43	1	2.33
27	27	292	1	0.34
27	28	199	1	0.50
27	29	121	1	0.83
27	30	106	1	0.94
27	31	68	1	1.47
27	32	53	1	1.89
27	33	40	1	2.50
27	34	36	1	2.78
28	28	274	1	0.36
28	29	194	1	0.52
28	30	114	1	0.88
28	31	69	1	1.45
28	32	46	1	2.17

28	33	30	1	3.33
28	34	28	1	3.57
28	35	3	1	33.33
29	29	270	1	0.37
29	30	186	1	0.54
29	31	102	1	0.98
29	32	65	1	1.54
29	33	47	1	2.13
29	34	40	1	2.50
29	35	1	1	100.00
30	30	294	1	0.34
30	31	202	1	0.50
30	32	121	1	0.83
30	33	78	1	1.28
30	34	53	1	1.89
30	35	3	1	33.33
31	31	215	1	0.47
31	32	145	1	0.69
31	33	76	1	1.32
31	34	57	1	1.75
31	35	1	1	100.00
32	32	267	1	0.37
32	33	188	1	0.53
32	34	94	1	1.06
32	35	8	1	12.50
33	33	286	1	0.35
33	34	202	1	0.50
33	35	9	1	11.11
34	34	279	1	0.36
34	35	44	1	2.27
35	35	18	1	5.56

4. The way users are engaging with the email service.

Result:

While performing the project, I learned how as a Data Analyst specializing in Operation Analytics, it is vital to examine abrupt surges in metrics. This involves comprehending the reasons for declines in metrics like daily engagement or sales. Investigating metric

spikes is a daily task that helps uncover the reasons behind fluctuations in business performance. The ultimate objective is to address the inquiries raised by various departments and extract meaningful insights from the diverse data sets and tables provided.

SQL Queries:

Case Study 1:

```
CREATE DATABASE Project3;


USE Project3;


ALTER TABLE job_data

ADD COLUMN temp_ds DATE

AFTER ds;


UPDATE job_data

SET temp_ds = STR_TO_DATE(ds, '%m/%d/%Y');


ALTER TABLE job_data

DROP COLUMN ds;


ALTER TABLE job_data

CHANGE COLUMN temp_ds ds DATE;
```

```
SELECT * FROM job_data;


#A.) No. of jobs reviewed per hour for each day

SELECT ds, COUNT(*) AS jobs_reviewed_per_day, SUM(time_spent)/3600 AS
hours_spent_reviewing

FROM job_data

group by ds;


# B. Throughput Analysis: Using two different types of subqueries

# type 1.

SELECT ds, ROUND(SUM(num_event) 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 rolling_7D_average

FROM (

        SELECT ds, COUNT(*) AS num_event , SUM(time_spent) AS total_time

        FROM job_data

        GROUP BY ds) AS totals;

# type 2.

WITH CTE AS (

  SELECT

    ds,

    COUNT(*) AS num_events,

    SUM(time_spent) AS total_time

  FROM job_data
```

```
  GROUP BY ds
)
SELECT
  ds,
  ROUND(SUM(num_events) 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 rolling_avg
FROM CTE;


# C. Language Share Analysis.
SELECT language, ROUND(100*COUNT(*)/(SELECT COUNT(*) FROM job_data),2) AS
Percentage
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY language;


# Duplicate Rows Detection: Using 2 different ways
SET SQL_SAFE_UPDATES = 0;
# Method 1
SELECT * FROM job_data;


WITH Duplicates AS(
        SELECT *, ROW_NUMBER() OVER(PARTITION BY job_id) AS RowNumber
```

```
        FROM job_data
)
SELECT * FROM Duplicates
WHERE RowNumber NOT IN
(SELECT MIN(RowNumber) FROM DUPLICATES GROUP BY job_id);


# Method 2
WITH Duplicates AS(
        SELECT *, ROW_NUMBER() OVER(PARTITION BY job_id) AS RowNumber
        FROM job_data
)
SELECT * FROM Duplicates
WHERE RowNumber > 1;
```

Case Study 2:

This dataset comprises of 3 tables and only one of which I have mentioned here to reduce the wastage of time. The other 2 tables were created, data was imported and altered in the same manner.

```
USE project3;
CREATE TABLE email_events(
user_id INT,
occured_at VARCHAR(100),
action VARCHAR(100),
user_type INT
);
```

```
LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv"
INTO TABLE email_events
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

SELECT * FROM email_events;

ALTER TABLE email_events ADD COLUMN temp_occured_at DATETIME AFTER user_id;

SET SQL_SAFE_UPDATES = 0;

UPDATE email_events SET temp_occured_at = STR_TO_DATE(occured_at, '%d-%m-%Y
%H:%i');

ALTER TABLE email_events DROP occured_at;

ALTER TABLE email_events CHANGE COLUMN temp_occured_at occured_at DATETIME;

# A. Weekly user engagement
SELECT EXTRACT(WEEK FROM occured_at) AS Week_Num, COUNT(user_id) AS Active_Users
FROM events
```

```
WHERE event_type = 'engagement'
AND event_name = 'login'
GROUP BY Week_Num;

# B. Write an SQL query to calculate the user growth for the product.
SELECT EXTRACT(MONTH FROM created_at) AS month,
COUNT(user_id) AS user_count,
LEAD(COUNT(user_id),1) OVER(ORDER BY MIN(created_at)) AS next_month_users
FROM users
GROUP BY month;

# C. Weekly Retention Analysis
WITH cohort AS(
        SELECT user_id, MIN(occured_at) AS signup_date
        FROM events
   WHERE event_type = 'engagement' AND event_name = 'login'
   GROUP BY user_id
),
user_week AS(
        SELECT e.user_id,
   EXTRACT(WEEK FROM occured_at) AS event_week,
   EXTRACT(WEEK FROM signup_date) AS signup_week
   FROM events e
JOIN cohort c
```

```
ON e.user_id = c.user_id

),

weekly_cohort AS(

        SELECT event_week, signup_week, COUNT(DISTINCT(uw.user_id)) AS cohort_size

    FROM user_week uw

    WHERE event_week <= signup_week + 7

    GROUP BY event_week, signup_week

)

SELECT wc.signup_week, wc.event_week, wc.cohort_size,

        SUM(CASE WHEN wc.event_week - wc.signup_week <= 7 THEN 1 ELSE 0 END) AS
retention_count,

    ROUND(SUM(CASE WHEN wc.event_week - wc.signup_week <= 7 THEN 1 ELSE 0 END) /
wc.cohort_size * 100, 2) AS retention_rate

FROM weekly_cohort wc

GROUP BY wc.signup_week, wc.event_week, wc.cohort_size

ORDER BY wc.signup_week, wc.event_week;



# D. Weekly Engagement Per Device

SELECT COUNT(DISTINCT user_id) AS num_users, EXTRACT(WEEK FROM occured_at) AS week,
device

FROM events

WHERE event_type = 'engagement'

GROUP BY week,device

ORDER BY week,device;
```

E. Email Engagement Analysis: