Operation Analytics and Investigating Metric Spike

Project Description

Operation analytics is the analysis performed for a company's whole end-to-end operations. This helps the business identify the areas where it needs to make improvements. You collaborate closely with the operations team, the support team, the marketing team, etc. and assist them in drawing conclusions from the data they gather.

Investigating metric spikes is a crucial component of operational analytics since a data analyst has to be able to answer queries such, "Why is there a decline in daily engagement?" or at least help other teams answer these questions. Why have sales decreased? Etc. Daily answers to questions like these are required, thus it is crucial to look at metric increase.

I am going to carry out an investigation utilizing MySQL Workbench to offer precise perceptions that will support the growth Operation Analytics and Metric Spike Analysis.

In this procedure, I will learn the following things:

Case Study 1 (Job Data)

1. Calculate the number of jobs reviewed per hour per day for November 2020?
2. Let us say the above metric is called throughput. Calculate 7-day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?
3. Calculate the percentage share of each language in the last 30 days?
4. Let us say you see some duplicate rows in the data. How will you display duplicates from the table?

Case Study 2 (Investigating metric spike)

1. Calculate the weekly user engagement?
2. Calculate the user growth for product?
3. Calculate the weekly retention of users-sign up cohort?
4. Calculate the weekly engagement per device?
5. Calculate the email engagement metrics?

Approach

I have already indicated that I am engaged in a MySQL Workbench project. In order to move forward, I will first create a database on MySQL Server for my investigation. The query is then executed on this data to analyze it in order to fully comprehend what is happening.

I have two case studies for you. In the first, we looked at the table job_data and determined the number of hours per day that jobs were evaluated, the proportion of languages used, if any rows were duplicates, and the seven-day rolling average of throughput using the throughput measure. In the second case study, "Investigating Metric Spike," three tables are supplied with the metrics

for weekly user engagement, user growth for the product, weekly retention of users from the sign-up cohort, weekly engagement per device, and email engagement.

Tech-Stack Used

My SQL version is 8.0.33. Adopting MySQL 8.0.33 for analysis may help businesses run more efficiently, make wiser decisions, and gain a competitive edge in their sectors. The advantages of adopting MySQL 8.0.33 are as follows:

1. **Enhanced effectiveness:** MySQL 8.0.33 offers significant speed improvements over earlier versions, making it faster and more efficient for analyzing large datasets.
2. **Enhanced security:** MySQL 8.0.33 is safer for storing and processing sensitive data thanks to its improved encryption and password management features.
3. **Improved scalability:** MySQL 8.0.33 is the ideal solution for data analysis since it can handle enormous datasets and sustain growing data volumes thanks to its improved scalability features.
4. **Advanced analytics:** Using MySQL 8.0.33's advanced analytics capabilities, which include support for window functions, Common Table Expressions (CTEs), and JSON capability, makes it easier to complete challenging analytical tasks.
5. **Easy fusion:** Due to its widespread use and comprehensive documentation, MySQL 8.0.33 is easy to connect with other platforms and data analysis tools, such as Tableau and R.

Insights

Case Study 1:

1. **Calculate the number of jobs reviewed per hour per day for November 2020?**

   Query for distinct Job_id:

```
Select count (distinct job_id)/(30*24) as number_of_jobs_reviewed_per_day  from job;
```

   Output:

number_of_jobs_reviewed_per_day
0.0083

   Query for non-distinct Job_id:

```
Select count (job_id)/(30*24) as number_of_jobs_reviewed_per_day from job;
```

   Output:

number_of_jobs_reviewed_per_day
0.0111

Observation: Here, we explicitly state that distinct and non-distinct reviews have different values, indicating that some job_ids are identical.

2. Calculate 7-day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

Query for distinct job_id:

```
SELECT ds as Review_date, Reviewed_jobs, AVG(Reviewed_jobs) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_average FROM (select ds, count(distinct job_id) as Reviewed_jobs from job group by ds order by ds) a;
```

Output :

Review_date	Reviewed_jobs	rolling_average
2020-11-25	1	1.0000
2020-11-26	1	1.0000
2020-11-27	1	1.0000
2020-11-28	2	1.2500
2020-11-29	1	1.2000
2020-11-30	2	1.3333

Query for non-distinct job_id:

```
SELECT ds as Review_date,Reviewed_jobs, AVG(Reviewed_jobs) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_average FROM (select ds,count(job_id) as Reviewed_jobs from job group by ds order by ds) a;
```

Output :

Review_date	Reviewed_jobs	rolling_average
2020-11-25	1	1.0000
2020-11-26	1	1.0000
2020-11-27	1	1.0000
2020-11-28	2	1.2500
2020-11-29	1	1.2000
2020-11-30	2	1.3333

Observation: Here, we can see that the 7-day rolling average of throughput is the same for distinct and non-different job_ids. Yes, I like 7-day rolling metrics since they may be useful when you want to smooth out daily variations and obtain a more precise sense of the underlying trend. They offer a moving average of performance over the previous seven days and aid in determining if a steady rising or negative trend is present.

3. Calculate the percentage share of each language in the last 30 days?

Query for distinct language:

```
Select

job_id, language,count(distinct language) as total_of_each_language,((count(language)/(select count(*) from job))*100) as percentage_share_of_each_language

from job group by job_id,language;
```

Output:

job_id	language	total_of_each_language	percentage_share_of_each_language
11	French	1	12.5000
20	Italian	1	12.5000
21	English	1	12.5000
22	Arabic	1	12.5000
23	Persian	1	37.5000
25	Hindi	1	12.5000

Query for non-distinct language:

```
select

job_id,language,count(language) as total_of_each_language,((count(language)/(select count(*) from job))*100) as percentage_share_of_each_language

from job group by job_id,language;
```

Output:

job_id	language	total_of_each_language	percentage_share_of_each_language
21	English	1	12.5000
22	Arabic	1	12.5000
23	Persian	3	37.5000
25	Hindi	1	12.5000
11	French	1	12.5000
20	Italian	1	12.5000

Observation: Here we can see that the language share percentage is the same for both distinct and non-distinctive language selections, and we see that Persian is used more than other languages.

4. Let us say you see some duplicate rows in the data. How will you display duplicates from the table?

Query:

```
SELECT  * FROM (SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num FROM job) a 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

Observation: We discovered two duplicate job_id here.

Investigating metric spike

Case Study 2:

1. Calculate the weekly user engagement

Query:

```
SELECT EXTRACT(WEEK FROM occurred_at) AS week, COUNT(DISTINCT user_id) AS num_of_weekly_engaged_users

FROM `table-2 events` GROUP BY week;
```

Output:

week	num_of_weekly_engaged_users
18	791
19	1244
20	1270
21	1341
22	1293
23	1366
24	1434
25	1462
26	1443
27	1477
28	1556
29	1556
30	1593
31	1685

32	1483
33	1438
34	1412
35	1442

Observation: Here, we retrieve weekly engaged users. As you can see, there are more engaged users in the 31st week than there are in the 18th.

2. Calculate the user growth for product

Query:

```
WITH active_users AS (SELECT YEAR(activated_at) AS year_num,WEEK(activated_at) AS week_num,COUNT(DISTINCT user_id) AS num_active_users FROM `table-1 users`

WHERE state = 'active' GROUP BY year_num, week_num ORDER BY year_num, week_num)

SELECT a.year_num, a.week_num, a.num_active_users, SUM(b.num_active_users) AS cum_active_users FROM active_users AS a JOIN active_users AS b ON a.year_num >=

b.year_num AND a.week_num >= b.week_num GROUP BY a.year_num,a.week_num, a.num_active_users ORDER BY  a.year_num, a.week_num;
```

Output:

year_num	week_num	user_id	cum_active_users
2013	1	67	67
2013	2	29	96
2013	3	47	143
2013	4	36	179
2013	5	30	209
2013	6	48	257
2013	7	41	298
2013	8	39	337
2013	9	33	370
2013	10	43	413
2013	11	33	446
2013	12	32	478
2013	13	33	511
2013	14	40	551
2013	15	35	586
2013	16	42	628
2013	17	48	676
2013	18	48	724

2013	19	45	769
2013	20	55	824
2013	21	41	865
2013	22	49	914
2013	23	51	965
2013	24	51	1016
2013	25	46	1062
2013	26	57	1119
2013	27	57	1176
2013	28	52	1228
2013	29	71	1299
2013	30	66	1365
2013	31	69	1434
2013	32	66	1500
2013	33	73	1573
2013	34	70	1643
2013	35	80	1723
2013	36	65	1788
2013	37	71	1859
2013	38	84	1943
2013	39	92	2035
2013	40	81	2116
2013	41	88	2204
2013	42	74	2278
2013	43	97	2375
2013	44	92	2467
2013	45	97	2564
2013	46	94	2658
2013	47	82	2740
2013	48	103	2843
2013	49	96	2939
2013	50	117	3056
2013	51	123	3179
2013	52	104	3283
2014	1	91	3374
2014	2	122	3496
2014	3	112	3608
2014	4	113	3721
2014	5	130	3851
2014	6	132	3983
2014	7	135	4118

2014	8	127	4245
2014	9	127	4372
2014	10	135	4507
2014	11	152	4659
2014	12	132	4791
2014	13	151	4942
2014	14	161	5103
2014	15	166	5269
2014	16	165	5434
2014	17	176	5610
2014	18	172	5782
2014	19	160	5942
2014	20	186	6128
2014	21	177	6305
2014	22	186	6491
2014	23	197	6688
2014	24	198	6886
2014	25	222	7108
2014	26	210	7318
2014	27	199	7517
2014	28	223	7740
2014	29	215	7955
2014	30	228	8183
2014	31	234	8417
2014	32	189	8606
2014	33	250	8856
2014	34	259	9115
2014	35	266	9381

Observation: Here we can see that most active users are more in 31st week as compare to other weeks.

Total active users

Query:

```
select count(*) as total_active_user  from `table-1 users` where state = 'active';
```

total_active_user
9381

3. Calculate the weekly retention of users-sign up cohort

Query without week number:

```
SELECT distinct user_id,COUNT(user_id),SUM(CASE WHEN retention_week = 1 Then 1 Else 0 END) as per_week_retention FROM(SELECT a.user_id,a.signup_week,b.engagement_week,

b.engagement_week - a.signup_week as retention_week FROM((SELECT distinct user_id, extract(week from occurred_at) as signup_week from `table-2 events` WHERE event_type = 'signup_flow' and event_name = 'complete_signup')a LEFT JOIN (SELECT distinct user_id, extract(week from occurred_at) as engagement_week FROM `table-2 events` where event_type = 'engagement')b on a.user_id = b.user_id))d

group by user_id order by user_id;
```

Output: Without week number query output link kindly press to see result

Observation: Here we get user_id 11833, 11893, 12034, 12389, 12976, 12995, and 13247, which has a maximum count of 14, which is an output without a week number.

Query with week number:

```
SELECT distinct user_id,COUNT(user_id),SUM(CASE WHEN retention_week = 1 Then 1 Else 0 END) as per_week_retention FROM(SELECT a.user_id,a.signup_week,b.engagement_week,

b.engagement_week - a.signup_week as retention_week FROM((SELECT distinct user_id, extract(week from occurred_at) as signup_week from `table-2 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 `table-2 events` where event_type = 'engagement')b on a.user_id = b.user_id))d

group by user_id

order by user_id;
```

Output: With week number kindly press to see result

Observation: Here we get user_id 11833, and 11893 has a maximum count of 14, which is an output without a week number.

4. Calculate the weekly engagement per device.

```
SELECT YEAR(occurred at) as year_num,WEEK(occurred_at) as week_num, device,
COUNT(DISTINCT user_id) as no_of_users FROM `table-2 events` WHERE event_type =
'engagement' GROUP BY YEAR(occurred_at), WEEK(occurred_at), device ORDER BY
YEAR(occurred_at), WEEK(occurred_at), device;
```

Output: User engagement per device kindly press to see result

Observation: Here we found out that MacBook Pro users are more in 2014 and week 32, which is 359, and Amazon Fire Phone users are less in 2014 and week 18, which is four.

5. Calculate the email engagement metrics.

Query:

```
WITH email_actions AS (SELECT *,
      CASE
          WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email')
              THEN 'email_sent'
          WHEN action = 'email_open'
              THEN 'email_opened'
          WHEN action = 'email_clickthrough'
              THEN 'email_clicked'
      END AS email_cat
   FROM `table-3 email_events`)
SELECT
   100.0 * SUM(CASE WHEN email_cat = 'email_opened' THEN 1 ELSE 0 END) / SUM(CASE
WHEN email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_opening_rate,
   100.0 * SUM(CASE WHEN email_cat = 'email_clicked' THEN 1 ELSE 0 END) / SUM(CASE WHEN
email_cat = 'email_sent' THEN 1 ELSE 0 END) AS email_clicking_rate
FROM email_actions;
```

Output :

email_opening_rate	email_clicking_rate
33.58339	14.78989

Observation: Here we can see that the email opening rate and email clicking rate are not so good, so we can improve those.

Result

- We explicitly state that distinct and non-distinct reviews have different values, indicating that some job_ids are identical.
- We can observe that both distinct and non-different job_ids have the same 7-day rolling average throughput. Yes, I do appreciate 7-day rolling metrics since they may be helpful for smoothing out daily fluctuations and getting a better feel of the underlying trend. They help identify if a continuous increasing or declining trend is there by providing a moving average of performance over the past seven days.
- Persian is used more frequently than other languages, and the language share percentage is the same for both distinct and non-distinctive language choices.
- Here, we can observe that the 31st week has a higher percentage of active users than the other weeks.
- This result lacks a week number and includes user_ids 11833, 11893, 12034, 12389, 12976, 12995, and 13247, with a maximum count of 14.
- Here, we receive user_id 11833, and 11893, which is an output without a week number, has a maximum count of 14.
- Here, we discovered that there were more MacBook Pro users in 2014 and week 32, which was 359, and less Amazon Fire Phone users in 2014 and week 18, which was four.
- We can enhance those as we can see that the email opening and clicking rates are not great.