Operation Analytics and Investigating Metric Spike

- **K Vikram Sundar**

Project Description :

The goal of this project is to retrieve the operational analytics report for end-to-end operations of a company's operations.

We also take a deep dive into investigation of the metric spike of user engagement on a daily and weekly basis to identify trends and patterns to make better data driven decisions.

Approach :

We gather all the relevant data and import into MySQL database and validate all the columns and their data types.
Then we use SQL to write optimized queries along with aggregation and window functions to perform the analysis and retrieve the result set which gives us a clear understanding of the metrics analysis and will be considered to further make business driven decisions.

Tools/Technologies Used

In this project , we use MySQL Workbench as the primary tool to gather relevant insights from the given dataset.
We use SQL queries to communicate with the database which contains the data from the dataset to retrieve only the data which we require based on the requirement from the marketing team.

Insights

Learnt how to use SQL queries by writing concise and optimized queries and also gained knowledge on several functions being used in MySQL to gather data effectively. The inferences made from the data are present in the below page.

Result

To identify trends and patterns in the daily or weekly metrics analysis to understand whether the overall engagement is positive or negative over a period of time to understand the areas of improvement to increase the overall engagement rate.

Overall Learning and Progress

Clear understanding on how to write optimized SQL queries by using aggregate and window functions

Case Study 1

1. **Jobs Reviewed Over Time:**

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

Query:

```
select ds as Date , round((count(job_id)/sum(time_spent))*3600) as jobs_rev_per_hour_per_day
from job_data
where ds between "11/01/2020" and "11/30/2020"
group by ds;
```

Output :

	Date	jobs_rev_per_hour_per_day
▶	11/30/2020	180
	11/29/2020	180
	11/28/2020	218
	11/27/2020	35
	11/26/2020	64
	11/25/2020	80

Outcome : The most jobs reviewed per hour was on 11/28/2020 with a count of 218.

2. **Throughput Analysis:**

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

Query:

```
-- weekly throughput value
select round((count(event)/sum(time_spent)),2) as weekly_throughput_value
from job_data;

-- daily metric value
select ds as Date , round((count(event)/sum(time_spent)),2) as daily_metric_value
from job_data
group by date;
```

Output :

	weekly_throughput_value
▶	0.03

	Date	daily_metric_value
▶	11/30/2020	0.05
	11/29/2020	0.05
	11/28/2020	0.06
	11/27/2020	0.01
	11/26/2020	0.02
	11/25/2020	0.02

Outcome : We can continue to use the 7 day rolling average as it provides a more clear view without being affected by daily up's and down's which aids in making better informed decisions in the long run.

3. **Language Share Analysis:**

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

Query :

```
select language,round((count(language)/8)*100,2) as perc_lang from job_data
where ds between "11/01/2020" and "11/30/2020"
group by language;
```

Output :

	language	perc_lang
▶	English	12.50
	Arabic	12.50
	Persian	37.50
	Hindi	12.50
	French	12.50
	Italian	12.50

Outcome : The language share in the last 30 days seems to be balanced with Persian being the highest contribution over other languages which is 37.5 %.

4. **Duplicate Rows Detection:**

   Identify duplicate rows in the data

Query :

```
select actor_id , count(actor_id)  as no_of_duplicates from job_data
group by actor_id having count(actor_id) > 1;
```

Output :

	actor_id	no_of_duplicates
▶	1003	2

Outcome : There are duplicate values present in the database for actor with actor_id 1003. Id's should always contain unique values.

Case Study 2

1. **Weekly User Engagement:**

   Measure the activeness of users on a weekly basis.

Query :

```
select extract(week from occurred_at) as weeks,
count(distinct user_id) as no_of_users from events_table
where event_type="engagement"
group by weeks order by weeks;
```

Output :

weeks	no_of_users
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

Outcome : The user engagement hits peak-level around week 30.

2. **User Growth Analysis:**

Analyze the growth of users over time for a product.

Query :

```
select week_num, year_num,
sum(active_users) over (order by week_num, year_num
rows between unbounded preceding and current row) as cumulative_sum
from (
select extract(week from activated_at) as week_num,
extract(year from activated_at) as year_num,
count(distinct user_id) as active_users from users
where state= "active"
group by year_num, week_num
order by year_num, week_num) as alias;
```

Output :

week_num	year_num	cumulative_sum
0	2013	23
0	2014	106
1	2013	136
1	2014	262
2	2013	310
2	2014	419
3	2013	455
3	2014	568
4	2013	598
4	2014	728
5	2013	776
5	2014	909
6	2013	947
6	2014	1082
7	2013	1124
7	2014	1249
8	2013	1283
8	2014	1412
9	2013	1455
9	2014	1588
10	2013	1620
10	2014	1774
11	2013	1805
11	2014	1935
12	2013	1968
12	2014	2116
13	2013	2155
13	2014	2322
14	2013	2357
14	2014	2519
15	2013	2562
15	2014	2726
16	2013	2772
16	2014	2951
17	2013	3000
17	2014	3170
18	2013	3214
18	2014	3377
19	2013	3434
19	2014	3619
20	2013	3658
20	2014	3834
21	2013	3883
21	2014	4066
22	2013	4120
22	2014	4316
23	2013	4366
23	2014	4562
24	2013	4607
24	2014	4836
25	2013	4893
25	2014	5100
26	2013	5156
26	2014	5357
27	2013	5409
27	2014	5631
28	2013	5703
28	2014	5918
29	2013	5985
29	2014	6206
30	2013	6273
30	2014	6511
31	2013	6578
31	2014	6771
32	2013	6842
32	2014	7087
33	2013	7160
33	2014	7421
34	2013	7499
34	2014	7758
35	2013	7821
35	2014	7839
36	2013	7911
37	2013	7996
38	2013	8086
39	2013	8170
40	2013	8257
41	2013	8330
42	2013	8429
43	2013	8518
44	2013	8614
45	2013	8705
46	2013	8793
47	2013	8895
48	2013	8992
49	2013	9108
50	2013	9232
51	2013	9334
52	2013	9381

Outcome : User Growth seems to be on the positive side throughout the year when compared with the weekly count.

3. **Weekly Retention Analysis:**

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

Query :

```
select extract(week from occurred_at) as weeks,
count(distinct user_id) as no_of_users from events
where event_type="signup_flow" and event_name="complete_signup"
group by weeks order by weeks;
```

Output :

weeks	no_of_users
17	72
18	163
19	185
20	176
21	183
22	196
23	196
24	229
25	207
26	201
27	222
28	215
29	221
30	238
31	193
32	245
33	261
34	259
35	18

Outcome : As per the weekly retention count , the count seems to decrease overtime.

4. **Weekly Engagement Per Device:**

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

Query :

```
select device, extract(week from occurred_at) as weeks,
count(distinct user_id) as no_of_users from events
where event_type="engagement"
group by device, weeks order by weeks;
```

Output :

device	weeks	no_of_users
acer aspire desktop	17	9
acer aspire notebook	17	20
amazon fire phone	17	4
asus chromebook	17	21
dell inspiron desktop	17	18
dell inspiron notebook	17	46
hp pavilion desktop	17	14
htc one	17	16
ipad air	17	27
ipad mini	17	19
iphone 4s	17	21
iphone 5	17	65
iphone 5s	17	42
kindle fire	17	6
lenovo thinkpad	17	86
mac mini	17	6
macbook air	17	54
macbook pro	17	143
nexus 10	17	16
nexus 5	17	40
nexus 7	17	18
nokia lumia 635	17	17

Outcome : Engagement consists of good number of users across various devices over the weeks. Some devices have a higher count of engagement from users when compared to the other devices.

5. **Email Engagement Analysis:**

Analyze how users are engaging with the email service.

Query :

```
select
(sum(case when
email_category="email_opened" then 1 else 0 end)/sum(case when email_category="email_sent" then 1 else 0 end))*100 as open_rate,
(sum(case when
email_category="email_clickthrough" then 1 else 0 end)/sum(case when email_category="email_sent" then 1 else 0 end))*100 as click_rate
from (
    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_clickthrough")
    end as email_category
    from email_events) as alias;
```

Output :

	open_rate	click_rate
▶	33.5834	14.7899

Outcome : The email engagement metrics shows that the open rate of emails is around 33.5% and the click rate of emails is around 14.7%