Operation Analytics and Investigating Metric Spike

Project Description

The project involves operational analysis for complete end to end operations of the company working with ops team, support team, marketing team, etc and help them derive insights out of the data they collect. We have two operations that we will be tackling as mentioned below.

Approach

The first step in the project was to create a database and tables using the data provided. Then using MySQL we perform a set of analysis answering the questions asked and presenting the insights obtained.

Tech-Stack Used

For this project, we used MySQL Workbench 8.0 CE for creating the databases and tables and running the required queries to get the insights. MySQL Workbench offers great configuration and connection parameters to MySQL database servers but we imported all the data locally for the project.

Insights

The analysis of Instagram database provided several insights which are as follows:-

CASE 1

A. Calculate the number of jobs reviewed per hour per day for November 2020.

```
SELECT
     date, total_time_spent,
    (total_time_spent/60) / (COUNT(DISTINCT DATE) * 24) AS avg_reviews_per_hour
FROM (
    SELECT
    DATE(ds) AS date,
    SUM(time_spent) AS total_time_spent,
    COUNT(*) AS jobs_reviewed
FROM
    case1.table1
    WHERE ds >= '2020-11-25' AND ds < '2020-12-01'
GROUP BY
    date
    ) AS reviews_by_hour
Group BY
    date, total_time_spent
```

The above query was used to find the total time spent on reviewing jobs on every date and the average reviews per hour per day for November, 2020. The details are as follows -

date	total_time_spent	avg_reviews_per_hour
2020-11-25	45	0.03125000
2020-11-26	56	0.03888889
2020-11-27	104	0.07222222
2020-11-28	33	0.02291667
2020-11-29	20	0.01388889
2020-11-30	40	0.02777778

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

```
SELECT
  DATE(ds) AS date,
  COUNT(*) as event_count,
  AVG(COUNT(*)) OVER (ORDER BY DATE(ds) ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling_average
FROM case1.table1
GROUP BY date;
```

From the above query we can find the number of events happening as the days go by and present a rolling average of the total number of events as they happen. The results are as follows –

date	event_count	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

Choosing a rolling average over daily average gives us a more comprehensive view of throughput by averaging the data over a longer period, smoothing out the effects of daily fluctuations. This approach helps us identify trends and make data-driven decisions based on more stable and reliable data.

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

```
SELECT language, language_count * 100.0 / total_count as percentage_share
FROM (
    SELECT language, COUNT(*) as language_count,
        (SELECT COUNT(*) FROM case1.table1 ) as total_count
    FROM case1.table1
    GROUP BY language
) as counts
GROUP BY language, language_count, total_count
ORDER BY percentage_share DESC;
```

Using the above query we get the list of languages used from the table and the share of percentage of each languages in the data. The results were obtained as follows –

language	percentage_share
Persian	37.50000
English	12.50000
Arabic	12.50000
Hindi	12.50000
French	12.50000
Italian	12.50000

From the data we can clearly see that the language Persian had the most percentage share of all the languages.

D. Display duplicate rows from the data.

To present the duplicate rows with same data in the table we used the following query-

```
SELECT job_id,
    COUNT(*) AS Num
    FROM case1.table1
GROUP BY job_id
HAVING COUNT(*) > 1;
```

From the above query we got the following result when checked for duplicate rows having same job IDs-

job_id	Num
23	3

Using the same query we can find duplicate rows for any other desired attributes of the data table by replacing the selected attribute in the query.

Results

Through this project, we were able to provide insights on jobs review average, calculate a weekly rolling average of throughput, share of languages and duplicate rows in the given data. The insights can be used by the management team to review and make data-driven decisions that will help them make better decisions for the company.

Case 2

A. To measure the activeness of a user. Measuring if the user finds quality in a product/service. Calculate the weekly user engagement?

```
SELECT
  EXTRACT( WEEK FROM occurred_at) AS Weeknum,
  COUNT(DISTINCT user_id) AS Usercount
  FROM case2.table2_events
  GROUP BY Weeknum
```

Running the above query gives us the number of times the users engaged on the platform in each week shown by week number as below-

Weeknum	Usercount
17	85
18	194
19	208
20	195
21	173
22	26
23	10
24	4
25	3

As we can see from the result, the number of users engaging on the platform gradually increases and peaks in week-19 and then slowly decreases. After week-22 we can see a sharp decline in the user engagement on the platform.

B. To calculate the amount of users growing over time for a product i.e., the user growth for product.

```
SELECT
  Weeknum, Num_active_users,
  SUM(Num_active_users) OVER(ORDER BY Weeknum ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS Cum_active_users
  FROM
    (
     SELECT
        EXTRACT(week FROM a.activated_at) AS Weeknum,
        COUNT(DISTINCT user_id) AS num_active_users
     FROM case2.`table1_ users` a
     WHERE state='active'
     GROUP BY Weeknum
     ) b
   GROUP BY Weeknum
```

Using the above query, we can obtain the number of active users in each week denoted by 'Weeknum' and cumulative active users by 'Cum_active_users'. The results obtained are as follows –

Weeknum	Num_active_users	Cum_active_users
0	106	106
1	156	262
2	157	419
3	149	568
4	160	728
5	181	909
6	173	1082
7	167	1249
8	163	1412
9	176	1588
10	186	1774
11	161	1935
12	181	2116
13	206	2322
14	197	2519
15	207	2726
16	225	2951
17	219	3170
18	207	3377
19	242	3619
20	215	3834
21	232	4066
22	250	4316
23	246	4562
24	274	4836
25	264	5100
26	257	5357
27	274	5631
28	287	5918
29	288	6206
30	305	6511
31	260	6771
32	316	7087
33	334	7421
34	337	7758
35	81	7839
36	72	7911
37	85	7996
38	90	8086
39	84	8170
40	87	8257
41	73	8330
42	99	8429
43	89	8518
44	96	8614
45	91	8705
46	88	8793
47	102	8895
48	97	8992
49	116	9108
50	124	9232
51	102	9334
52	47	9381

So, from this table we can see that the growth of users has grown from just a few hundreds to 9000 by the 52nd week.

C. Calculate number of users getting retained weekly after signing-up for a product.
To get this data we had to run this complex query which is as shown below –

```
SELECT
 user_id,
 COUNT(user_id) AS user_count,
 signup_week, engagement_week,
 SUM(CASE WHEN retention_week=1 THEN 1
        WHEN retention_week=2 THEN 2
        WHEN retention_week=3 THEN 3
        WHEN retention_week=4 THEN 4
        WHEN retention_week=5 THEN 5 ELSE 0 END) AS week_count
```

```
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 case2.table2_events
  WHERE event_type='signup_flow'
  AND event_name='complete_signup'
  AND EXTRACT(week FROM occurred_at)
  ) a
  LEFT JOIN
  (
  SELECT DISTINCT user_id,
   EXTRACT( week FROM occurred_at) AS engagement_week
   FROM case2.table2_events
   WHERE event_type='engagement'
  ) b
  ON a.user_id = b.user_id
  )
  ORDER BY a.user_id
  ) d
  GROUP BY user_id, signup_week, engagement_week
```

This gave us a list of all the users, their signup week, engagement week and also their week count. A small sample of the data obtained is shown below.

user_id	user_count	signup_week	engagement_week	week_count
11768	1	17	17	0
11770	1	17	17	0
11775	1	17	17	0
11775	1	17	18	1
11778	1	17	17	0
11778	1	17	21	4
11778	1	17	23	0
11779	1	17	17	0
11780	1	17	17	0
11785	1	17	17	0
11787	1	17	17	0
11787	1	17	18	1
11787	1	17	19	2
11791	1	17	17	0
11793	1	17	17	0
11795	1	17	17	0
11795	1	17	18	1
11798	1	17	17	0
11799	1	17	17	0
11799	1	17	20	3

The data spans from user ID 11768 to 13293 with a few instances of repeated IDs and missing IDs as well. The result clearly shows the weeks the users were active and engaged on the platform.

D. Measure the activeness of a user. Measuring if the user finds quality in a product/service weekly. To calculate the weekly engagement per device.

```
SELECT
  EXTRACT( YEAR FROM occurred_at) AS Year,
  EXTRACT( WEEK FROM occurred_at) AS Weeknum,
  device,
  COUNT(DISTINCT user_id) AS Usercount
  FROM case2.table2_events
  GROUP BY Year, Weeknum, device
  ORDER BY Year, Weeknum, device
```

From the above query we can obtain the week number, device type and the user count on these devices. The result obtained for year 2014 week 17 is as follows –

Year	Weeknum	device	Usercount
2014	17	acer aspire desktop	2
2014	17	acer aspire notebook	2
2014	17	amazon fire phone	1
2014	17	asus chromebook	3
2014	17	dell inspiron desktop	1
2014	17	dell inspiron notebook	4
2014	17	hp pavilion desktop	2
2014	17	htc one	2
2014	17	ipad air	1
2014	17	ipad mini	3
2014	17	iphone 4s	3
2014	17	iphone 5	11
2014	17	iphone 5s	5
2014	17	lenovo thinkpad	8
2014	17	mac mini	1
2014	17	macbook air	4
2014	17	macbook pro	13
2014	17	nexus 5	4
2014	17	nexus 7	4
2014	17	nokia lumia 635	2
2014	17	samsumg galaxy tablet	2
2014	17	samsung galaxy note	1
2014	17	samsung galaxy s4	7

Similarly, the data is also obtained for rest of the weeks of the year available from the raw data tables.

E. Users engaging with the email service. Calculate the email engagement metrics?
For this metric, we run a small case clause query conditioning the email category and using it to generate a percentage of email opening rate and email clicked rate. The query is as mentioned below –

```sql
SELECT
 100*SUM(CASE WHEN email_cat='email_open' THEN 1 ELSE 0 END)/SUM(CASE WHEN email_cat='email_sent' THEN 1 ELSE 0 END) AS email_open_rate,
 100*SUM(CASE WHEN email_cat='email_clickthrough' THEN 1 ELSE 0 END)/SUM(CASE WHEN email_cat='email_sent' THEN 1 ELSE 0 END) AS email_clicked_rate
 FROM
  (
  SELECT *,
   CASE
     WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email')
      THEN 'email_sent'
     WHEN action IN ('email_open')
      THEN 'email_open'
      WHEN action IN ('email_clickthrough')
      THEN 'email_clickthrough'
    END AS email_cat
   FROM case2.table3_email_events
   ) a
```

This query gave us the following result –

email_open_rate	email_clicked_rate
33.5834	14.7899

As we can observe the email opening rate remains greater than the email clicked rate as many target users avoided clicking on the links in the mail.

Results

From this case, we got a very good insight into the data and metric spikes for various questions relating to the engagement, growth and retention of users on the platform. These results can help the operations team, marketing team and support team to make more effective workflow and better data-driven decisions to help the company progress further and grow.