Operation Analytics and Investigating Metric Spike

Project Description: In this project, we have performed operation analytics and investigated metric spike which is used to perform end-to-end operations for company growth and which areas to improve on. In this project, I have found out the –

1. The number of jobs reviewed per hour per day for November 2020.
2. 7-day rolling average of throughput.
3. The Percentage share of each language in the last 30 days.
4. Rows that have the same value present in them.
5. The weekly user engagement.
6. The user growth for the product.
7. The weekly retention of users-sign up cohort.
8. The weekly engagement per device.
9. The email engagement metrics. From the provided dataset.

This kind of analysis is further used to predict the overall growth or decline of a company's fortune. It means better automation, better understanding between cross-functional teams, and more effective workflows. Investigating metric spikes is also an important part of operation analytics to understand or make other teams understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that, it is very important to investigate metric spikes. This analysis helps the company to look into the ongoing business's growth and downfall too. Through this, the company gets a view of why there is a certain loss in sales. What should we improve to get the business on the proper track? Etc. A company may get a proper understanding about what is the current state of its sales, if any loss what measures can be taken to overcome such conditions, operational analytics is a more specific term for a type of business analytics that focuses on improving existing operations.

Approach:

1. Initially I downloaded the provided datasets to the device. Both datasets were in the form of CSV (Comma Separate Value) format.
2. The datasets are imported into MySQL Workbench 8.0.31 for performing queries.
3. At times, I have used CMD (Command Prompt) for performing operations on the databases.
4. After execution of proper SQL queries the snapshot of the answers to the questions is given in the result section before being properly reviewed.

Execution :

Case Study 1 (Job Data)

A. **Number of jobs reviewed:** Amount of jobs reviewed over time.
Your task: Calculate the number of jobs reviewed per hour per day for November 2020?

```
SELECT COUNT(job_id)/(30*24) AS num_jobs_reviewed
FROM `sqlproject-1`
WHERE
ds BETWEEN '2020-11-01' AND '2020-11-30';
```

B. **Throughput:** It is the no. of events happening per second.
Your task: Let's 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?

```
SELECT ds,
jobs_reviewed,
AVG (jobs_reviewed) OVER (ORDER BY ds rows BETWEEN 6 preceding and current row) AS
rolling_average
FROM
( SELECT ds,
COUNT(DISTINCT(job_id) ) AS jobs_reviewed
FROM `sqlproject-1`
WHERE ds BETWEEN "2020-11-01" AND "2020-11-30"
GROUP BY ds
ORDER BY ds )a;
```

C. **Percentage share of each language:** Share of each language for different contents.
Your task: Calculate the percentage share of each language in the last 30 days?

```
SELECT language,
num_jobs,
100.0* (num_jobs/total_jobs) AS pct_share
FROM
( SELECT language, COUNT(job_id) AS num_jobs
FROM `sqlproject-1`
GROUP BY language ) a
CROSS JOIN
( SELECT COUNT(job_id) AS total_jobs
FROM `sqlproject-1` ) b;
```

D. **Duplicate rows:** Rows that have the same value present in them.
Your task: Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

```
SELECT *
FROM
( SELECT *,
row_number() OVER (PARTITION BY job_id) AS rownum FROM  `sqlproject-1`) a
WHERE rownum>1;
```

Case Study 2 (Investigating Metrics Spikes)

A. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.
Your task: Calculate the weekly user engagement?

```
SELECT EXTRACT(WEEK FROM occurred_at) AS weeknum, COUNT(DISTINCT user_id)
FROM events GROUP BY weeknum;
```

B. **User Growth:** Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

```
SELECT year, weeknum, num_active_users,
SUM(num_active_users) OVER (ORDER BY year, weeknum ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_users
FROM
(SELECT EXTRACT( YEAR FROM activated_at) AS year,
EXTRACT(WEEK FROM activated_at) AS weeknum,
COUNT(DISTINCT user_id) AS num_active_users
FROM users a
WHERE state='active'
GROUP BY year, weeknum
ORDER BY year, weeknum ) a;
```

C. **Weekly Retention:** Users getting retained weekly after signing-up for a product.
Your task: Calculate the weekly retention of users-sign up cohort?

```
SELECT COUNT(user_id) AS num_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 1,2,3
ORDER BY user_id) c
```

D. **Weekly Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.
Your task: Calculate the weekly engagement per device?

```
SELECT EXTRACT(YEAR FROM occurred_at) AS year,
EXTRACT(WEEK FROM occurred_at) AS week,
device,
COUNT(DISTINCT user_id )
FROM events
WHERE event_type = “engagement”
GROUP BY 1,2,3
ORDER BY 1,2,3;
```

E. **Email Engagement:** Users engaging with the email service.
Your task: Calculate the email engagement metrics?

```
SELECT EXTRACT(WEEK FROM occurred_at) AS week,
COUNT(CASE WHEN e.action = 'sent_weekly_digest' THEN e.user_id ELSE NULL END) AS
weekly_emails,
COUNT(CASE WHEN e.action = 'sent_reengagement_email' THEN e.user_id ELSE NULL
END) AS reengagement_emails,
COUNT(CASE WHEN e.action = 'email_open' THEN e.user_id ELSE NULL END) AS
email_opens,
COUNT(CASE WHEN e.action = 'email_clickthrough' THEN e.user_id  ELSE NULL END) AS
email_clickthroughs
FROM email_events e
GROUP BY 1;
```

Tech-Stack Used: For providing the information, operations are performed by installing MySQL 8.0.28 Server and MySQL Workbench 8.0.31. The provided datasets are loaded in MySQL Workbench 8.0.31 which helped me to execute queries more efficiently, also MySQL Workbench 8.0.31 provides a superb view of databases. After connecting the CMD (Command Prompt) to the MySQL 8.0.28 Server, SQL operations are performed in Command Prompt platform and some of them are also performed in MySQL Workbench 8.0.31.

Insights: First of all, I have studied the provided datasets and their columns before operations, which helped me to understand what exactly the dataset is about and what information it holds. From the tables, I discovered the number of jobs reviewed per hour per day for November 2020 i.e. **0.0083** jobs got reviewed in the month of November 2020 per hour per day. Then calculated 7-day rolling average of throughput (between 25-11-2020 and 30-11-2020) is displayed in the result section. And evaluated the percentage share of each language in the last 30 days in which the Persian language has the highest amount of contribution i.e. **50.00%** meanwhile English, Italian and French have **16.67%** each. Lastly displayed duplicate rows (**job_ id 23**) from the table dataset.
Subsequently, investigating metrics spike is carried out where at first calculated the weekly user engagement is about **2774** and the number of users growing over time for a product is evaluated (**9381**). Then calculated the weekly retention of users sign-up which results in **0** and the weekly engagement of users per device where MacBook pro sits at the top with **484** users. Last, weekly emails(**15688**), re-engagement emails(**0**), emails opens(**4692**) and email clickthrough (**1662**) have also been evaluated.

Result: In this project, I learned how the Operation analysis and Investigating of Spike Metrics take place. I got to know how to use Advanced SQL concepts like window functions (over(), row_number(), etc.), Date and Time functions to perform operations, and how the companies perform the Operation analysis and Investigating of Spike Metrics to stabilize the ups and downs in the sales. I also determined exactly when we need sub-queries and why. This project helped me to clear more concepts of SQL and assisted in understanding SQL problems.
The outputs of the executed SQL queries are mentioned below:

Case Study 1 (Job Data)

A. **Number of jobs reviewed:** Amount of jobs reviewed over time.
Your task: Calculate the number of jobs reviewed per hour per day for November 2020?

```
mysql> select count(job_id)/(30*24) as num_jobs_reviewed from `sqlproject-1` where ds between "2020-11-01" and "2020-11-30";
+-------------------+
| num_jobs_reviewed |
+-------------------+
|            0.0083 |
+-------------------+
1 row in set (0.00 sec)
```

B. **Throughput:** It is the no. of events happening per second.
Your task: Let's 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?

```
mysql> select ds,  jobs_reviewed,  avg(jobs_reviewed) over (order by ds rows between 6 preceding and current row) as rolling_average  from (select ds,count(distinct job_id)
 as jobs_reviewed from`sqlproject-1` where ds between "2020-11-01" and "2020-11-30" group by ds order by ds) a;
+------------+--------------+-----------------+
| ds         | jobs_reviewed | rolling_average |
+------------+--------------+-----------------+
| 2020-11-25 |            1 |          1.0000 |
| 2020-11-26 |            1 |          1.0000 |
| 2020-11-27 |            1 |          1.0000 |
| 2020-11-28 |            1 |          1.0000 |
| 2020-11-29 |            1 |          1.0000 |
| 2020-11-30 |            1 |          1.0000 |
+------------+--------------+-----------------+
6 rows in set (0.00 sec)
```

C. **Percentage share of each language:** Share of each language for different contents.
Your task: Calculate the percentage share of each language in the last 30 days?

```
mysql> select language,num_jobs,100*(num_jobs/total_jobs) as pct_share from (select language,count(job_id) as num_jobs from `sqlproject-1` group by language)a cross join(se
lect count(job_id) as total_jobs from `sqlproject-1`)b;
+----------+----------+-----------+
| language | num_jobs | pct_share |
+----------+----------+-----------+
| English  |        1 |   16.6667 |
| Persian  |        3 |   50.0000 |
| French   |        1 |   16.6667 |
| Italian  |        1 |   16.6667 |
+----------+----------+-----------+
4 rows in set (0.00 sec)
```

D. **Duplicate rows:** Rows that have the same value present in them.
Your task: Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

```
mysql> select * from(select *,row_number()over(partition by job_id) as rownum from `sqlproject-1`)a where rownum>1;
+------------+--------+----------+----------+----------+------------+------+--------+
| ds         | job_id | actor_id | event    | language | time_spent | org  | rownum |
+------------+--------+----------+----------+----------+------------+------+--------+
| 2020-11-28 |     23 |     1005 | transfer | Persian  |         22 | D    |      2 |
| 2020-11-26 |     23 |     1004 | skip     | Persian  |         56 | A    |      3 |
+------------+--------+----------+----------+----------+------------+------+--------+
2 rows in set (0.00 sec)
```

Case Study 2 (Investigating Metrics Spikes)

A. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.
Your task: Calculate the weekly user engagement?

```
mysql> select extract(week from occurred_at) as weeknum,count(distinct user_id) from events
    -> group by weeknum;
+---------+-------------------------+
| weeknum | count(distinct user_id) |
+---------+-------------------------+
|    NULL |                    2774 |
+---------+-------------------------+
1 row in set, 55248 warnings (0.13 sec)
```

B. **User Growth:** Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

```
mysql> select year,weeknum,num_active_user,sum(num_active_user) over(order by year,weeknum rows between unbounded preceding and current row) as cum_active_users from(select
 extract(year from activated_at) as year,extract(week from activated_at)as weeknum,count(distinct user_id) as num_active_user from users a where state="active" group by yea
r,weeknum order by year,weeknum)a;
+------+---------+----------------+-----------------+
| year | weeknum | num_active_user | cum_active_users |
+------+---------+----------------+-----------------+
| NULL |    NULL |           9381 |            9381 |
+------+---------+----------------+-----------------+
1 row in set, 37536 warnings (0.12 sec)
```

C. **Weekly Retention:** Users getting retained weekly after signing-up for a product.
Your task: Calculate the weekly retention of users-sign up cohort?

num_users	per_week_retention
0	NULL

D. **Weekly Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.
Your task: Calculate the weekly engagement per device?

```
mysql> select extract(year from occurred_at)as year,extract(week from occurred_at)as week,device,count(distinct user_id)from events where event_type="engagement" group by 1
,2,3 order by 1,2,3;
+------+------+-----------------------+-------------------------+
| year | week | device                | count(distinct user_id) |
+------+------+-----------------------+-------------------------+
| NULL | NULL | acer aspire desktop   |                      49 |
| NULL | NULL | acer aspire notebook  |                      77 |
| NULL | NULL | amazon fire phone     |                      24 |
| NULL | NULL | asus chromebook       |                      89 |
| NULL | NULL | dell inspiron desktop |                      90 |
| NULL | NULL | dell inspiron notebook|                     165 |
| NULL | NULL | hp pavilion desktop   |                      80 |
| NULL | NULL | htc one               |                      40 |
| NULL | NULL | ipad air              |                     111 |
| NULL | NULL | ipad mini             |                      56 |
| NULL | NULL | iphone 4s             |                     101 |
| NULL | NULL | iphone 5              |                     256 |
| NULL | NULL | iphone 5s             |                     154 |
| NULL | NULL | kindle fire           |                      46 |
| NULL | NULL | lenovo thinkpad       |                     336 |
| NULL | NULL | mac mini              |                      32 |
| NULL | NULL | macbook air           |                     267 |
| NULL | NULL | macbook pro           |                     484 |
| NULL | NULL | nexus 10              |                      58 |
| NULL | NULL | nexus 5               |                     149 |
| NULL | NULL | nexus 7               |                      83 |
| NULL | NULL | nokia lumia 635       |                      60 |
| NULL | NULL | samsumg galaxy tablet |                      22 |
| NULL | NULL | samsung galaxy note   |                      31 |
| NULL | NULL | samsung galaxy s4     |                     186 |
| NULL | NULL | windows surface       |                      48 |
+------+------+-----------------------+-------------------------+
26 rows in set, 65535 warnings (0.23 sec)
```

E. **Email Engagement:** Users engaging with the email service.
Your task: Calculate the email engagement metrics?

```
mysql> SELECT EXTRACT(week FROM occurred_at) AS week, COUNT(CASE WHEN e.action = 'sent_weekly_digest' THEN e.user_id ELSE NULL END) AS weekly_emails, COUNT(CASE WHEN e.acti
on = 'sent_reengagement_email' THEN e.user_id ELSE NULL END) AS reengagement_emails, COUNT(CASE WHEN e.action = 'email_open' THEN e.user_id ELSE NULL END) AS email_opens, C
OUNT(CASE WHEN e.action = 'email_clickthrough' THEN e.user_id  ELSE NULL END) AS email_clickthroughs  FROM email_events e GROUP BY 1;
+------+---------------+----------------------+---------------+--------------------+
| week | weekly_emails | reengagement_emails | email_opens | email_clickthroughs |
+------+---------------+----------------------+---------------+--------------------+
| NULL |         15688 |                    0 |          4692 |               1662 |
+------+---------------+----------------------+---------------+--------------------+
1 row in set, 25173 warnings (0.16 sec)
```