Project: **Operation Analytics and Investigating Metric Spike**

Project Description

This Project involves analyzing a company's end-to-end operations by working closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect.

As a data analyst, first I will understand the project requirements and purpose and then I will understand the data which is provided in the project attachment and will perform analysis to get the meaningful insights.

Analysis: involves understanding and explaining sudden changes in key metrics.

For example, in case study 1 results by calculating jobs reviewed per hour for each day shows inconsistency in jobs reviewed per day. With this information operations team can look in to challenges and plans for day-to-day activities to bring the consistency. 7 day rolling averages analysis provides overall performance over time and long-term trends and patterns that can be used for strategic business decisions.

In case study 2 calculation of weekly user engagement showed that there is a raise or increase in user engagement in initial weeks and a steady decline in user engagement after that

Approach

1. **Understanding the data**: The first step was to understand the questions given to me in both the case studies and understand structure of the data and created the database “operation_analytics”. The attachment provided the csv files for 4 tables, one for case study 1 and three for case study 2 including job_data, users, events and email_events. I analyzed the data types and values in each column to understand what kind of information each table contains.

2. **Formulating SQL Queries**: Based on your questions, I wrote SQL queries to calculate the metrics you were interested in. This involved using various SQL functions and clauses such as COUNT(), GROUP BY, ORDER BY, DATE_FORMAT(), WEEK(), INTERVAL, and CASE WHEN. I also used JOIN operations to combine data from different tables when necessary.

3. **Analyzing Data**: Once I had the SQL queries, I executed the queries to retrieve the data from CSV files. For this I used CREATE, LOAD, ALTER and UPDATE queries and added additional functions to my SQL queries like CONCAT(), DATE_ADD() to get the required data.

4. **Interpreting Results**: After retrieving the data, I interpreted the results to answer the questions of both case studies. For example, I calculated weekly user engagement by counting the number of events per user per week, and I

calculated email engagement metrics by counting the number of each action per user.

5. **Providing Insights**: Based on the results of the data analysis, I provided insights that could be useful for decision-making. Throughout this process, my goal was to provide clear, accurate, and helpful insights and answers to questions posed by management team.

Tech-Stack Used

The software and versions you used for the project:

MySQL Workbench 8.0 CE – To write SQL queries to analyze Instagram user data and answer questions posed by the management team

MYSQL Server 8.0 – To create and store database

Pandas – to create graph visualisation

Insights:

I will provide my insights in the form of answers for the questions posed by management team

CASE STUDY 1: Job Data Analysis

1. **Jobs Reviewed Over Time:**

- Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

SQL Query:
To calculate the number of jobs reviewed per hour for each day in November 2020, you would first need to calculate the total time spent on jobs each day. Then, divide the number of jobs by the total time spent in hours.

```
SELECT datestamp, COUNT(job_id) / SUM(time_spent / 60.0) as jobs_per_hour
FROM operration_analytics.job_data
GROUP BY datestamp;
```

Output:

Below is the result for above query which shows number of jobs reviewed per hour for each day.

Result Grid | Filter Rows:

datestamp	jobs_per_hour
2020-11-30	2.9999
2020-11-29	3.0003
2020-11-28	3.6364
2020-11-27	0.5769
2020-11-26	1.0715
2020-11-25	1.3333

2. **Throughput Analysis**:

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

SQL Query:

```sql
/*B. Throughput Analysis:
Objective: Calculate the 7-day rolling average of throughput (number of events per second). */
SELECT datestamp,
        AVG(time_spent) OVER (ORDER BY datestamp ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling_avg_throughput
FROM operration_analytics.job_data
WHERE MONTH(datestamp) = 11 AND YEAR(datestamp) = 2020;
```

Output:

Above query give the average time_spent for each day and the six preceding days in November 2020. It assumes that time_spent is a measure of throughput.

Result Grid | Filter Rows:

datestamp	rolling_avg_throughput
2020-11-25	45.0000
2020-11-26	50.5000
2020-11-27	68.3333
2020-11-28	56.7500
2020-11-28	47.6000
2020-11-29	43.0000
2020-11-30	39.0000
2020-11-30	36.1429

calculating jobs reviewed per hour for each day shows inconsistency in jobs reviewed per day. With this information operations team can look in to challenges and plans for day-to-day activities to bring the consistency. 7 day rolling averages analysis provides overall performance over time and long-term trends and patterns that can be used for strategic business decisions.

So, the choice between these two would depend on whether you're more interested in short-term operational insights or long-term strategic trends.

3. Language Share Analysis:

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

SQL Query:

```sql
/*C. Language Share Analysis:
Objective: Calculate the percentage share of each language in the last 30 days.*/
SELECT language,
        COUNT(*) * 100.0 / (SELECT COUNT(*) FROM job_data WHERE datestamp >= (SELECT MAX(datestamp) FROM job_data) - INTERVAL 30 DAY) as percentage
FROM job_data
WHERE datestamp >= (SELECT MAX(datestamp) FROM job_data) - INTERVAL 30 DAY
GROUP BY language
Order by percentage desc;
```

Output: Above query shows percentage share of each language in the last 30 days. Persian language has more share percentage

Result Grid | Filter Rows:

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

4. Duplicate Rows Detection:

- Objective: Identify duplicate rows in the data.

SQL Query:

```sql
/* D. Duplicate Rows Detection:
Objective: Identify duplicate rows in the data.*/
SELECT job_id, actor_id, event, language, time_spent, org, datestamp, COUNT(*)
FROM job_data
GROUP BY job_id, actor_id, event, language, time_spent, org, datestamp
HAVING COUNT(*) > 1;
```

Output:
above query gave no result as there is not duplicate rows in the default data provided in the project attachment.
We can create dummy data by adding additional rows to the table by changing some information like year in date column or by copying existing rows then we will get duplicate rows in the results.

CASE STUDY 2: Investigating Metric Spike

1. Weekly User Engagement:

- Objective: Measure the activeness of users on a weekly basis

SQL Query:

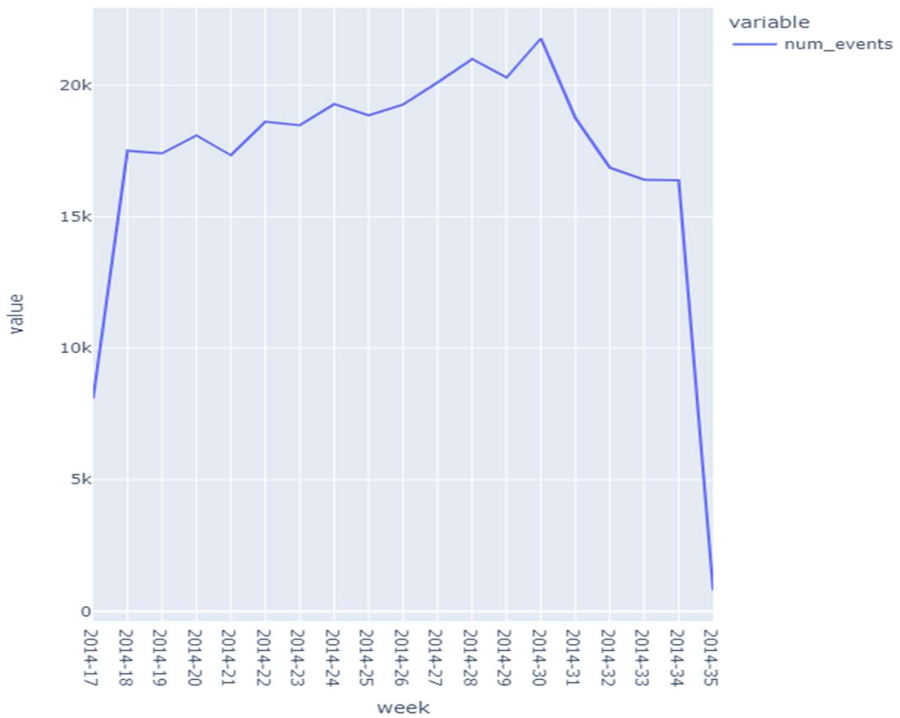
```sql
/*Weekly User Engagement:
•   Objective: Measure the activeness of users on a weekly basis
*/
SELECT CONCAT(YEAR(occured_at), ' Week -', WEEK(occured_at)) as week, COUNT(*) as num_events
FROM events
GROUP BY week
ORDER BY week DESC, num_events DESC;
```

Output: As per below result we can see user engagement was steadily increasing but later from last four weeks there is rapid decline in user activity

Result Grid | Filter Rows:

week	num_events
2014 Week -35	802
2014 Week -34	16386
2014 Week -33	16406
2014 Week -32	16857
2014 Week -31	18749
2014 Week -30	21771
2014 Week -29	20288
2014 Week -28	20991
2014 Week -27	20103

Result 10



2. User Growth Analysis:

- Objective: Analyze the growth of users over time for a product.

SQL Query:

First calculate the number users registered in each week

```
SELECT DATE_FORMAT(created_at, '%Y-%u') as week, COUNT(*) as new_users
FROM users
GROUP BY week
ORDER BY week;
```

Next calculate the cumulative number of users up to each week to find the growth of users over time

```sql
SELECT
    curr.week,
    curr.new_users,
    curr.new_users + IFNULL(SUM(prev.new_users), 0) as cumulative_users
FROM
    (SELECT DATE_FORMAT(created_at, '%Y-%u') as week, COUNT(*) as new_users
     FROM users
     GROUP BY week) curr
LEFT JOIN
    (SELECT DATE_FORMAT(created_at, '%Y-%u') as week, COUNT(*) as new_users
     FROM users
     GROUP BY week) prev
ON curr.week > prev.week
GROUP BY curr.week, curr.new_users
ORDER BY curr.week;
```

Output:

This second query will list the number of new users for each week and also growth of users for each week by adding previous weeks users to the current week

Result Grid | Filter Rows:

week	new_users	cumulative_users
2013-01	26	26
2013-02	29	55
2013-03	47	102
2013-04	36	138
2013-05	30	168
2013-06	48	216
2013-07	41	257
2013-08	39	296
2013-09	33	329

3. Weekly Retention Analysis:

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

SQL Query:

```sql
SELECT
    DATE_FORMAT(u.created_at, '%Y-%u') as sign_up_week,
    COUNT(DISTINCT u.user_id) as sign_ups,
    COUNT(DISTINCT CASE WHEN e.occured_at BETWEEN u.created_at AND DATE_ADD(u.created_at, INTERVAL 7 DAY) THEN u.user_id END) as week_1,
    COUNT(DISTINCT CASE WHEN e.occured_at BETWEEN DATE_ADD(u.created_at, INTERVAL 7 DAY) AND DATE_ADD(u.created_at, INTERVAL 14 DAY) THEN u.user_id END) as week_2,
    COUNT(DISTINCT CASE WHEN e.occured_at BETWEEN DATE_ADD(u.created_at, INTERVAL 14 DAY) AND DATE_ADD(u.created_at, INTERVAL 21 DAY) THEN u.user_id END) as week_3
FROM
    users u
LEFT JOIN
    events e ON u.user_id = e.user_id
GROUP BY
    sign_up_week
ORDER BY
    sign_up_week;
```

Output:

to calculate the weekly retention of users based on their sign-up cohort. Above SQL query calculates the number of users who signed up each week and how many of them had an event in the following weeks

Result Grid | Filter Rows: | Export:

sign_up_week	sign_ups	week_1	week_2	week_3
2014-28	223	223	118	90
2014-29	215	215	108	70
2014-30	228	228	104	70
2014-31	234	234	106	75
2014-32	189	189	78	56
2014-33	250	250	100	48
2014-34	259	259	81	0
2014-35	266	266	0	0

4. Weekly Engagement Per Device:

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

SQL Query:

```sql
/*Weekly Engagement Per Device:
Objective: Measure the activeness of users on a weekly basis per device.*/
SELECT DATE_FORMAT(occured_at, '%Y-%u') as week, device, COUNT(*) as num_events
FROM events
GROUP BY week, device
ORDER BY week DESC, num_events DESC;
```

Output:

Above query counts the number of events per device per week to find weekly engagement per device

Result Grid | Filter Rows:

week	device	num_events
2014-35	macbook pro	3179
2014-35	lenovo thinkpad	1956
2014-35	macbook air	1446
2014-35	dell inspiron notebook	1087
2014-35	iphone 5	957

5. Email Engagement Analysis:

- Objective: Analyze how users are engaging with the email service.

SQL Query:

```sql
/*Email Engagement Analysis:
Objective: Analyze how users are engaging with the email service.*/
SELECT user_id,
       COUNT(CASE WHEN action = 'sent_weekly_digest' THEN 1 END) as sent_weekly_digest,
       COUNT(CASE WHEN action = 'email_open' THEN 1 END) as email_open
FROM email_events
GROUP BY user_id;
```

Output:

Result Grid | Filter Rows:

user_id	sent_weekly_digest	email_open
0	17	5
4	17	5
8	17	3
11	17	5
17	17	4
19	17	5
20	17	8
22	17	7
30	18	6

Result

This project definitely helps me rebrushing my SQL skills and I regained my confidence on using SQL for data analytics. I was relying mostly on Excel and Power BI for data analysis but with this project I feel SQL is easier to use.

In this project based on the above analysis we can find how is the user engagement and new users growth over time, and which language and device

most of the users are preferring to interact with the applications. Based on this company can take decisions on the changes or modifications to bring to make the application more user friendly.

Drive Link

Case Study 1

https://drive.google.com/file/d/1zoSiB2sknV0xB960MIXwkt725wAzkYIO/view?usp=sharing

Case Study 2

https://drive.google.com/file/d/1X_2v3Wir3znmo66e0Z8KFL8YwRkMugoj/view?usp=sharing