# Operation Analytics and Investigating Metric Spike

## Overview:

Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. This analysis helps identify areas for improvement within the company. As a Data Analyst, you'll work closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect.

In this project, I will be using SQL and MySQL Workbench as my tool I will take on the role of a Lead Data Analyst at a company like Microsoft. I will derive insights from this data to answer questions posed by different departments within the company. My goal is to use advanced SQL skills to analyze the data and provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics

## Approach:

I used My SQL workbench 8.0 CE to execute my queries and fetch my data. First I created a database name files and then created my tables such as users,job_data,events and many other tables for both the case studies containing all the necessary constraints. Then I inserted my data into the respective tables using csv files provided. then I used my SQL queries and various

functions such as joins, group by clause, aggregate functions, nested sub queries to fetch my data.

# Tech-Stack:

I will be using My SQL workbench 8.0 CE to complete my project.

I choose this application because it is one of the best software used in analysis of data as it is fast, robust, have data security.it is free from data redundancy and data inconsistency and also it holds ACID properties.

It provides a lot of functions and modulations such ass aggregate functions, window functions, subqueries, nested subqueries and correlated subqueries which makes data fetching easier and helps us to perform our task conveniently.

# *Insights and results:*

During the course of this project I got deep knowledge of SQL uses and learnt how to use it in a huge database management system which in this case was Operation Analytics and Investigating Metric Spike. I fetched number of jobs reviewed per hour for each day , Calculate the 7-day rolling average of throughput , Analyze the growth of users over time for a product. And many more operations during the course of this

project and used my knowledge of SQL to do my task.it was a very good experience putting knowledge into use. I learned a lot in this project and deepen my SQL skills.

# Part A:

# Case Study 1: Job Data Analysis

## Query 1:

```
SELECT
    DATE(ds) AS date,
    HOUR(ds) AS hour,
    COUNT(*) AS jobs_reviewed
FROM
    job_data
WHERE
    MONTH(ds) = 11 AND YEAR(ds) = 2020
GROUP BY DATE(ds) , HOUR(ds)
```

Output:

| date | jobs_reviewed |
|---|---|
| 2020-11-25 | 1 |
| 2020-11-26 | 1 |
| 2020-11-27 | 1 |
| 2020-11-28 | 2 |
| 2020-11-29 | 1 |
| 2020-11-30 | 2 |

# Query 2:

```
SELECT
    ds, AVG(job_count) AS rolling_7_day_avg
FROM
    (SELECT
        ds,
            job_count,
            @running_sum:=@running_sum + job_count - COALESCE(@job_count_7_days_ago, 0) AS cumulative_sum,
            @job_count_7_days_ago:=job_count
    FROM
        (SELECT
        ds, COUNT(*) AS job_count
    FROM
        job_data
    GROUP BY ds
        job_data
    GROUP BY ds
    ORDER BY ds) AS daily_job_counts
    JOIN (SELECT @running_sum:=0, @job_count_7_days_ago:=NULL) AS vars) AS cumulative_counts
GROUP BY ds
ORDER BY ds;
```

## Output:

| ds | rolling_7_day_avg |
|---|---|
| 11/25/2020 | 1.0000 |
| 11/26/2020 | 1.0000 |
| 11/27/2020 | 1.0000 |
| 11/28/2020 | 2.0000 |
| 11/29/2020 | 1.0000 |
| 11/30/2020 | 2.0000 |

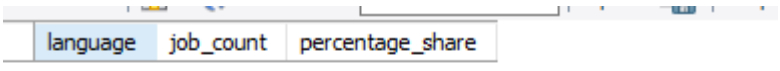
I prefer using the 7-day rolling average for throughput analysis because it provides a clearer view of underlying trends and reduces the noise from daily variations. This makes it easier to understand the general performance and identify longer-term patterns. However, it's also

important to monitor daily metrics alongside the rolling average to catch any sudden changes or specific anomalies that might require immediate attention.

# Query 3:

```
SELECT
    language,
    COUNT(*) AS job_count,
    ROUND(COUNT(*) * 100.0 / (SELECT
                    COUNT(*)
                FROM
                    job_data
                WHERE
                    ds >= CURDATE() - INTERVAL 30 DAY),
            2) AS percentage_share
FROM
    job_data
WHERE
    ds >= CURDATE() - INTERVAL 30 DAY
GROUP BY language
ORDER BY percentage_share DESC
```

## Output:

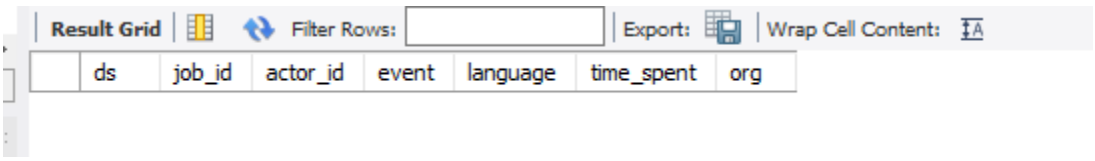


| language | job_count | percentage_share |
|---|---|---|

# Query 4:

```sql
SELECT *
FROM job_data
WHERE (job_id, actor_id, event, language, time_spent, org, ds) IN (
    SELECT job_id, actor_id, event, language, time_spent, org, ds
    FROM job_data
    GROUP BY job_id, actor_id, event, language, time_spent, org, ds
    HAVING COUNT(*) > 1
)
ORDER BY job_id, actor_id, event, language, time_spent, org, ds;
```

Output:




No dublicate rows

Part B:

**Investigating Metric Spike**

# Query 1:

```
SELECT
    DATE_FORMAT(occurred_at, '%Y-%U') AS week,
    COUNT(DISTINCT user_id) AS weekly_engagement
FROM
    events
GROUP BY week
ORDER BY week;
```

Output:

| week | weekly_engagement |
|---|---|
| 2014-17 | 168 |
| 2014-18 | 320 |
| 2014-19 | 344 |
| 2014-20 | 328 |
| 2014-21 | 335 |
| 2014-22 | 365 |
| 2014-23 | 348 |
| 2014-24 | 389 |
| 2014-25 | 372 |
| 2014-26 | 360 |
| 2014-27 | 392 |

| week | weekly_engagement |
|---|---|
| 2014-25 | 372 |
| 2014-26 | 360 |
| 2014-27 | 392 |
| 2014-28 | 398 |
| 2014-29 | 392 |
| 2014-30 | 417 |
| 2014-31 | 342 |
| 2014-32 | 367 |
| 2014-33 | 381 |
| 2014-34 | 392 |
| 2014-35 | 21 |

# Query 2:

```
SELECT
    COUNT(DISTINCT user_id) AS user_growth
FROM
    users
WHERE
    activated_at IS NOT NULL;
```

Output:

Result Grid | Filter Rows:

| user_growth |
|---|
| 9381 |

# Query 3:

```sql
SELECT
  DATE_FORMAT(u.created_at, '%Y-%U') AS cohort,
  COUNT(DISTINCT u.user_id) AS total_users,
  COUNT(DISTINCT e.user_id) AS retained_users,
  COUNT(DISTINCT e.user_id) / COUNT(DISTINCT u.user_id) AS retention_rate
FROM
  users u
  LEFT JOIN events e ON u.user_id = e.user_id
WHERE
  e.occurred_at >= u.created_at
  AND e.occurred_at <= DATE_ADD(u.created_at, INTERVAL 7 DAY)
GROUP BY
  cohort
```

# Query 4:

```sql
SELECT
    YEARWEEK(occurred_at, 1) AS week,
    device,
    COUNT(*) AS engagement_count
FROM
    events
GROUP BY YEARWEEK(occurred_at, 1) , device
ORDER BY week , device;
```

Output:

| week | device | engagement_count |
|---|---|---|
| 201418 | acer aspire desktop | 11 |
| 201418 | acer aspire notebook | 72 |
| 201418 | amazon fire phone | 12 |
| 201418 | asus chromebook | 122 |
| 201418 | dell inspiron desktop | 74 |
| 201418 | dell inspiron notebook | 112 |
| 201418 | hp pavilion desktop | 57 |
| 201418 | htc one | 9 |
| 201418 | ipad air | 102 |
| 201418 | ipad mini | 94 |
| 201418 | iphone 4s | 52 |
| 201418 | iphone 5 | 261 |

| week | device | engagement_count |
|---|---|---|
| 201418 | iphone 5 | 261 |
| 201418 | iphone 5s | 139 |
| 201418 | kindle fire | 13 |
| 201418 | lenovo thinkpad | 224 |
| 201418 | mac mini | 26 |
| 201418 | macbook air | 120 |
| 201418 | macbook pro | 337 |
| 201418 | nexus 10 | 4 |
| 201418 | nexus 5 | 134 |
| 201418 | nexus 7 | 36 |
| 201418 | nokia lumia 635 | 47 |
| 201418 | samsumg galaxy tablet | 17 |

| | | |
|---|---|---|
| 201418 | samsumg galaxy tablet | 17 |
| 201418 | samsung galaxy note | 29 |
| 201418 | samsung galaxy s4 | 104 |
| 201419 | acer aspire desktop | 105 |
| 201419 | acer aspire notebook | 65 |
| 201419 | amazon fire phone | 41 |
| 201419 | asus chromebook | 100 |
| 201419 | dell inspiron desktop | 68 |
| 201419 | dell inspiron notebook | 234 |
| 201419 | hp pavilion desktop | 98 |
| 201419 | htc one | 23 |
| 201419 | ipad air | 131 |
| 201419 | ipad mini | 100 |
| 201419 | iphone 4s | 128 |
| 201419 | iphone 5 | 323 |
| 201419 | iphone 5s | 135 |
| 201419 | kindle fire | 62 |
| 201419 | lenovo thinkpad | 516 |
| 201419 | mac mini | 25 |
| 201419 | macbook air | 368 |

Result 15

| | | |
|---|---|---|
| 201419 | nexus 5 | 218 |
| 201419 | nexus 7 | 77 |
| 201419 | nokia lumia 635 | 47 |
| 201419 | samsumg galaxy tablet | 22 |
| 201419 | samsung galaxy note | 59 |
| 201419 | samsung galaxy s4 | 228 |
| 201419 | windows surface | 28 |
| 201420 | acer aspire desktop | 32 |
| 201420 | acer aspire notebook | 145 |
| 201420 | amazon fire phone | 28 |
| 201420 | asus chromebook | 64 |
| 201420 | dell inspiron desktop | 65 |
| 201420 | dell inspiron notebook | 252 |
| 201420 | hp pavilion desktop | 73 |
| 201420 | htc one | 56 |
| 201420 | ipad air | 166 |
| 201420 | ipad mini | 85 |
| 201420 | iphone 4s | 119 |
| 201420 | iphone 5 | 340 |
| 201420 | iphone 5s | 210 |

esult 15 ×

| | | |
|---|---|---|
| 201420 | macbook pro | 686 |
| 201420 | nexus 10 | 49 |
| 201420 | nexus 5 | 310 |
| 201420 | nexus 7 | 116 |
| 201420 | nokia lumia 635 | 67 |
| 201420 | samsumg galaxy tablet | 29 |
| 201420 | samsung galaxy note | 42 |
| 201420 | samsung galaxy s4 | 229 |
| 201420 | windows surface | 41 |
| 201421 | acer aspire desktop | 14 |
| 201421 | acer aspire notebook | 59 |
| 201421 | amazon fire phone | 12 |
| 201421 | asus chromebook | 85 |
| 201421 | dell inspiron desktop | 182 |
| 201421 | dell inspiron notebook | 236 |
| 201421 | hp pavilion desktop | 52 |
| 201421 | htc one | 46 |
| 201421 | ipad air | 153 |
| 201421 | ipad mini | 51 |
| 201421 | iphone 4s | 103 |

esult 15 ×

.....

# Query 5:

```sql
SELECT
    DATE(occurred_at) AS date,
    COUNT(*) AS total_emails_sent,
    COUNT(CASE WHEN action = 'open' THEN 1 END) AS total_emails_opened,
    COUNT(CASE WHEN action = 'click' THEN 1 END) AS total_emails_clicked,
    ROUND(COUNT(CASE WHEN action = 'open' THEN 1 END) / COUNT(*) * 100, 2) AS open_rate,
    ROUND(COUNT(CASE WHEN action = 'click' THEN 1 END) / COUNT(*) * 100, 2) AS click_rate
FROM
    email_events
GROUP BY
    DATE(occurred_at)
ORDER BY
    date;
```

Output:

| date | total_emails_sent | total_emails_opened | total_emails_clicked | open_rate | click_ra |
|---|---|---|---|---|---|
| 2014-05-01 | 72 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-02 | 61 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-05 | 98 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-06 | 52 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-07 | 46 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-08 | 67 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-09 | 58 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-12 | 103 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-13 | 65 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-14 | 48 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-15 | 67 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-16 | 60 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-19 | 103 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-20 | 60 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-21 | 48 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-22 | 63 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-23 | 61 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-26 | 112 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-27 | 56 | 0 | 0 | 0.00 | 0.00 |
| 2014-05-28 | 46 | 0 | 0 | 0.00 | 0.00 |

Result 16 ×

| date | total_emails_sent | total_emails_opened | total_emails_clicked | open_rate | click_rate |
|---|---|---|---|---|---|
| 2014-08-07 | 65 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-08 | 61 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-11 | 97 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-12 | 51 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-13 | 44 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-14 | 62 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-15 | 57 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-18 | 94 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-19 | 60 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-20 | 52 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-21 | 68 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-22 | 59 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-25 | 100 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-26 | 58 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-27 | 53 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-28 | 72 | 0 | 0 | 0.00 | 0.00 |
| 2014-08-29 | 53 | 0 | 0 | 0.00 | 0.00 |