Operation Analytics and Investigating Metric Spike

Advanced SQL

Project Description

Operational Analytics is a powerful tool that enables data analysts to optimize and improve the performance of any company. By analyzing the data collected from various sources, such as user behavior, events, emails, and more, data analysts can identify patterns, trends, and anomalies that can help them make better decisions and recommendations.

In this project, we will learn how to use **advanced SQL** skills to perform operational analytics on real-world datasets. We will be working on **two** case studies that will challenge us to apply our SQL knowledge and analytical thinking to solve real business problems. We will also learn how to use **MySQL Workbench**, a popular software for working with databases and SQL queries. By the end of this project, we will have a solid understanding of how to use SQL for operational analytics and how to present the findings in a clear and concise report.

Approach

To perform the analysis and answer the questions, I followed these steps:

- I downloaded and installed **MySQL Workbench** on my computer. I chose this tool because it is easy to use, has a user-friendly interface, and supports various features such as syntax highlighting, auto-completion, and query execution.

- I created a **database** and **tables** using the MySQL Workbench. I used the provided table structures and links to import the csv files into my tables. I verified that the data was imported correctly and that there were no errors or missing values.

- I performed analysis using **SQL queries** to answer the questions mentioned in the case studies. I used appropriate functions, clauses, and joins to manipulate the data and obtain the desired results. I also used comments to explain my logic and reasoning behind each query.

- I submitted a report to present my findings to the leadership team. I used a **PDF** format to create a professional and appealing report. I included the following details in my report:

1. **Project description**: I provided a brief overview of the project, explaining its purpose and how I planned to handle the analysis.
2. **Approach**: I described my approach towards the project and explained how I executed the analysis.
3. **Tech-stack used**: I mentioned the software and versions used for the project, such as MySQL Workbench, and explained their purpose in the analysis.
4. **Insights**: I summarized the insights and knowledge I gained during the project. I highlighted key observations and inferences I made from the data, keeping them concise and to the point.
5. **Result**: I discussed what I achieved through the project and explained how it contributed to my understanding and decision-making.

Tech-Stack Used

To complete this project, I used the following software:

- **MySQL Workbench 8.0.34**: This is a graphical user interface (GUI) tool that allows me to create and manage databases, tables, and queries using SQL. I used this tool to import the CSV files into my tables, write and execute SQL queries, and export the results as CSV files.
- **Microsoft Word 365**: This is a word processing software that allows me to create and edit documents using text, images, and formatting. I used this tool to create a PDF report for my project, using the results exported from MySQL Workbench and the text written by me.

These are the main software I used for the project. I chose them because they are widely used, reliable, and compatible with each other. They also have many features and functions that make data analysis easier and more efficient.

Insights

Here are some of the insights and knowledge I gained while working on this project:

1. Case Study 1: Job Data Analysis:

a) Task 1: Jobs Reviewed Over Time:

One of the tasks that the operations team assigned me was to measure the **number of jobs reviewed per hour** for each day in November 2020. This is important because it helps us understand the workload, availability, and productivity of the actors who review the jobs.

To measure the jobs reviewed over time, I used the following SQL query and generated the output:

```sql
SELECT DATE(ds) AS date, HOUR(ds) AS hour, COUNT(*) AS jobs_reviewed
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY date, hour
ORDER BY date, hour;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

date	hour	jobs_reviewed
2020-11-25	0	1
2020-11-26	0	1
2020-11-27	0	1
2020-11-28	0	2
2020-11-29	0	1
2020-11-30	0	2

From this output, we can see that the number of jobs reviewed per hour for each day in November 2020 showed a clear pattern of daily and weekly fluctuations. The highest number of jobs reviewed was on ***November 28*** and ***November 30***, with ***2*** jobs, and the other days in November had only ***1*** job reviewed. This could be related to the workload, availability, and productivity of the actors.

b) Task 2: Throughput Analysis:

One of the tasks that the operations team assigned me was to calculate the **7-day rolling average of throughput** (number of events per second) for the last 30 days. This is important because it helps us measure the performance and efficiency of the system that processes the jobs.

To calculate the throughput analysis, I used the following SQL query and generated the output:

```sql
SELECT ds, AVG(throughput) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling_avg
FROM (
  SELECT ds, COUNT(*) / SUM(time_spent) AS throughput
  FROM job_data
  GROUP BY ds
) AS subquery;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ds	rolling_avg
2020-11-25	0.02220000
2020-11-26	0.02005000
2020-11-27	0.01656667
2020-11-28	0.02757500
2020-11-29	0.03206000
2020-11-30	0.03505000

From this output, we can see that the 7-day rolling average of throughput showed a smooth and steady trend over time, with some minor spikes and dips. The highest throughput was on ***November 30***, with ***0.035*** events per second, and the lowest throughput was on ***November 27***, with ***0.016*** events per second. This could be related to the complexity, quality, and volume of the jobs.

c) Task 3: Language Share Analysis:

One of the tasks that the operations team assigned me was to calculate the **percentage share of each language** in the last 30 days. This is important because it helps us understand the demand, supply, and preference of the customers and the actors for different languages.

To calculate the language share analysis, I used the following SQL query and generated the output:

```sql
SELECT language, CAST(COUNT(*) AS DECIMAL) / SUM(COUNT(*)) OVER () * 100 AS percentage
FROM job_data
WHERE ds BETWEEN DATE_SUB('2020-11-30', INTERVAL 30 DAY) AND '2020-11-30'
GROUP BY language;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

language	percentage
English	12.5000
Arabic	12.5000
Persian	37.5000
Hindi	12.5000
French	12.5000
Italian	12.5000

From this output, we can see that ***Persian*** was the dominant language in the last 30 days, with ***37.5%*** of the total jobs, followed by other languages with ***12.5%*** each. This could be related to the demand, supply, and preference of the customers and the actors for different languages.

d) Task 4: Duplicate Rows Detection:

One of the tasks that the operations team assigned me was to identify **duplicate rows** in the data. This is important because duplicate rows can cause errors and inconsistencies in the data and affect the accuracy and reliability of the analysis.

To identify duplicate rows in the data, I used the following SQL query and generated the output:

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ds	job_id	actor_id	event	language	time_spent	org

From this output, we can see that there were ***no*** duplicate rows in the data. If any duplicate rows were present, it could be due to human error, system glitch, or data corruption.

2. Case Study 2: Investigating Metric Spike:

a) Task 1: Weekly User Engagement:

One of the tasks that the product team assigned me was to measure the **activeness of users on a weekly basis** for a product. This is important because active users are more likely to use the product's features, interact with other users, and generate more revenue for the business.

To measure the weekly user engagement, I used the following SQL query and generated the output:

```sql
SELECT DATE_FORMAT(MIN(occurred_at), '%Y-%m-%d') as week_start,
COUNT(DISTINCT user_id) AS active_users,
COUNT(DISTINCT user_id) * 100.0 / (SELECT COUNT(DISTINCT user_id) FROM users) AS weekly_user_engagement
FROM events
GROUP BY WEEK(occurred_at), YEAR(occurred_at)
ORDER BY WEEK(occurred_at), YEAR(occurred_at);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

week_start	active_users	weekly_user_engagement
2014-05-01	663	7.06748
2014-05-04	1068	11.38471
2014-05-11	1113	11.86441
2014-05-18	1154	12.30146
2014-05-25	1121	11.94969
2014-06-01	1186	12.64258
2014-06-08	1232	13.13293
2014-06-15	1275	13.59130
2014-06-22	1264	13.47404
2014-06-29	1302	13.87912
2014-07-06	1372	14.62531
2014-07-13	1365	14.55069
2014-07-20	1376	14.66795
2014-07-27	1467	15.63799
2014-08-03	1299	13.84714
2014-08-10	1225	13.05831
2014-08-17	1225	13.05831
2014-08-24	1204	12.83445
2014-08-31	104	1.10862

From this output, we can see that there was a constant rise in the number of active users till the week starting from ***July 27***, reaching a record high of ***1,467*** users. This could be due to a successful marketing campaign, a viral word-of-mouth effect, or a new feature launch that attracted more users to the product. Then there was a constant fall and then a sudden dip in the week starting from ***August 31***, reaching a record low of ***104*** active users.

b) Task 2: User Growth Analysis:

One of the tasks that the product team assigned me was to analyze the **growth of users over time** for a product. This is important because user growth indicates the success and potential of the product in the market.

To analyze the user growth for the product, I used the following SQL query and generated the output:

```sql
SELECT MONTH(created_at) AS month,
YEAR(created_at) AS year,
COUNT(user_id) AS new_users,
ROUND((COUNT(user_id) - LAG(COUNT(user_id), 1) OVER (ORDER BY YEAR(created_at),
MONTH(created_at))) * 100.0 / LAG(COUNT(user_id), 1) OVER (ORDER BY YEAR(created_at),
MONTH(created_at)), 2) AS user_growth
FROM users
GROUP BY month, year
ORDER BY year, month
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month	year	new_users	user_growth
1	2013	160	NULL
2	2013	160	0.00
3	2013	150	-6.25
4	2013	181	20.67
5	2013	214	18.23
6	2013	213	-0.47
7	2013	284	33.33
8	2013	316	11.27
9	2013	330	4.43
10	2013	390	18.18
11	2013	399	2.31
12	2013	486	21.80
1	2014	552	13.58
2	2014	525	-4.89
3	2014	615	17.14
4	2014	726	18.05
5	2014	779	7.30
6	2014	873	12.07
7	2014	997	14.20
8	2014	1031	3.41

From this output, we can see that there was a steady increase in the number of new users per month, reaching a peak of ***1,031*** new users in ***August 2014***. This could be due to a growing market demand, a strong brand reputation, or a competitive advantage that attracted more customers to the product.

c) Task 3: Weekly Retention Analysis:

One of the tasks that the product team assigned me was to analyze the **retention of users on a weekly basis** after signing up for a product. This is important because retention measures how loyal and satisfied users are with the product.

To analyze the weekly retention of users based on their sign-up cohort, I used the following SQL query and generated the output:

```sql
WITH user_cohort AS ( SELECT u.user_id,
    WEEK(u.created_at) AS sign_up_week,
    YEAR(u.created_at) AS sign_up_year,
    WEEK(e.occurred_at) AS login_week,
    YEAR(e.occurred_at) AS login_year
    FROM users u
    JOIN events e
ON u.user_id = e.user_id AND e.event_name = 'login')
SELECT sign_up_week, sign_up_year, login_week, login_year,
COUNT(DISTINCT user_id) AS retained_users,
COUNT(DISTINCT user_id) * 100.0 / (SELECT COUNT(DISTINCT user_id)
FROM user_cohort uc2
WHERE uc2.sign_up_week = uc1.sign_up_week AND uc2.sign_up_year = uc1.sign_up_year) AS weekly_retention
FROM user_cohort uc1
GROUP BY sign_up_week, sign_up_year, login_week, login_year
ORDER BY sign_up_year, sign_up_week, login_year, login_week;
```

sign_up_week	sign_up_year	login_week	login_year	retained_users	weekly_retention
0	2013	17	2014	2	22.22222
0	2013	18	2014	3	33.33333
0	2013	19	2014	3	33.33333

[CSV file for user retention]

From this output, we can see that there was a gradual increase in the percentage of users who remained active after signing up for the product. There were many instances of high retention rates, where it reached a peak of ***100%***. The lowest retention rate reached was ***0.45249%***. This could be due to a lack of user satisfaction, engagement, or loyalty with the product.

d) Task 4: Weekly Engagement Per Device:

One of the tasks that the product team assigned me was to measure the **activeness of users on a weekly basis per device** for a product. This is important because it helps us understand how users prefer to access and use the product on different devices.

To measure the weekly engagement per device, I used the following SQL query and generated the output:

```sql
SELECT WEEK(occurred_at) AS week,
YEAR(occurred_at) AS year,
device,
COUNT(DISTINCT user_id) AS active_users,
COUNT(DISTINCT user_id) * 100.0 / (SELECT COUNT(DISTINCT user_id)
FROM events e2
WHERE e2.device = e1.device) AS weekly_engagement_per_device
FROM events e1
GROUP BY week, year, device
ORDER BY year, week, device;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

week	year	device	active_users	weekly_engagement_per_device
17	2014	acer aspire desktop	9	4.54545
17	2014	acer aspire notebook	20	5.91716
17	2014	amazon fire phone	4	4.49438
17	2014	asus chromebook	21	5.91549
17	2014	dell inspiron desktop	18	5.00000
17	2014	dell inspiron notebook	46	6.79468
17	2014	hp pavilion desktop	14	4.12979
17	2014	htc one	16	8.16327
17	2014	ipad air	27	5.64854
17	2014	ipad mini	19	6.50685
17	2014	iphone 4s	21	5.13447
17	2014	iphone 5	65	6.34146
17	2014	iphone 5s	42	6.70927
17	2014	kindle fire	6	2.92683
17	2014	lenovo thinkpad	86	6.56990
17	2014	mac mini	6	4.00000
17	2014	macbook air	54	5.68421
17	2014	macbook pro	143	7.32582
17	2014	nexus 10	16	5.86081
17	2014	nexus 5	40	6.44122
17	2014	nexus 7	18	5.07042

CSV file for user engagement

From this output, we can see that there was a significant difference in the number of active users per device type for each week. The most popular device type was ***MacBook pro***, which had an average of ***322*** active users per week, followed by ***Lenovo ThinkPad*** with an average of ***220*** active users per week. This could be due to a preference for larger screens, better performance, or more functionality on desktop devices.

e) Task 5: Email Engagement Analysis:

One of the tasks that the product team assigned me was to analyze how **users are engaging with the email service** for a product. This is important because email is one of the main channels for communicating and marketing with users.

To analyze the email engagement metrics for each user type (existing or new), I used the following SQL query and generated the output:

```sql
SELECT action,
COUNT(DISTINCT user_id) AS users,
COUNT(DISTINCT user_id) * 100.0 / (SELECT COUNT(DISTINCT user_id)
FROM email_events) AS email_engagement_metric
FROM email_events
GROUP BY action;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

action	users	email_engagement_metric
email_clickthrough	5277	85.40217
email_open	5927	95.92167
sent_reengagement_email	3653	59.11960
sent_weekly_digest	4111	66.53180

From this output, we can see that there was a ***high*** response rate for both existing and new users for the email service. The average open rate for emails was **95.92%**, and the average click rate for emails was **85.4%**. This could be due to a good email design, content, or timing that succeeded in capturing the attention or interest of the users.

Result

Through this project, I achieved the following results:

- I created a database and tables using MySQL Workbench. I imported the csv files into my tables and verified that the data was correct and complete.
- I performed analysis using SQL queries to answer the questions mentioned in the case studies. I used advanced SQL functions, clauses, and joins to manipulate the data and obtain the desired results. I also used comments to explain my logic and reasoning behind each query.
- I submitted a report to present my findings to the leadership team. I used a PDF format to create a professional and appealing report. I included all the details required in the report, such as project description, approach, tech-stack used, insights, and result.

By completing this project, I improved my SQL skills and learned how to use SQL for operational analytics. I also gained valuable insights and knowledge about the data and the company. I demonstrated my ability to handle complex data analysis tasks and present them in a clear and concise manner. I also created a portfolio-ready project that showcases my SQL skills and analytical thinking.