

Project Description

As a Lead Data Analyst, you'll delve into Operational Analytics to enhance business operations by working with teams like operations, support, and marketing to mine actionable insights from data. Your daily challenge will be to decode metric spikes—sudden shifts in key performance indicators—to maintain operational health. Armed with advanced SQL skills, you'll analyze datasets and tables to provide answers and strategies that improve Microsoft-like company operations and explain metric variations succinctly. This project's aim is to gather data and insights from given dataset to provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.

Approach

Values were generated and entered into the database by executing the DDL (Data Definition Language) and DML (Data Manipulation Language) SQL queries supplied by the product manager, in accordance with the project requirements, utilizing the MySQL Workbench for the MySQL database.

The process entailed the following steps:

1. Database creation was initiated.
2. A table named 'job_data' was established.
3. A 'job_id' column was created as a unique identifier of jobs.
4. A column called 'actor_id' has been created as a unique identifier of actor.
5. An event column has been created with the name 'event' which stores the type of event which can either be decision/skip/transfer.
6. A column by the name 'language' has been created that keeps track of the language of the content.
7. Column 'time_spent' has been created which keeps track of the time spent to review the job in seconds.
8. A column by the name 'org' notes the organization the actor belongs to.
9. Column 'ds' tracks the date in the format of yyyy/mm/dd and is stored as text.
10. Values were populated into all tables using the provided datasets using Insert – Values command.
11. Upon database completion, necessary insights were extracted from the database tables through SQL queries executed in MySQL Workbench.

Tech-Stack Used

- MySQL Workbench
- MySQL Community Server – GPL Version 8.0.29
- SQL Server Management Studio (SSMS)
- Microsoft Excel

In Operations Analytics, tools such as MySQL Workbench, MySQL Community Server, SQL Server Management Studio (SSMS), and Microsoft Excel are critical for their specialized functions: database design, data storage, SQL infrastructure management, and data analysis visualization, respectively. They streamline the entire data process from creation to reporting, ensuring efficient and effective insights for operational enhancement.

Insights

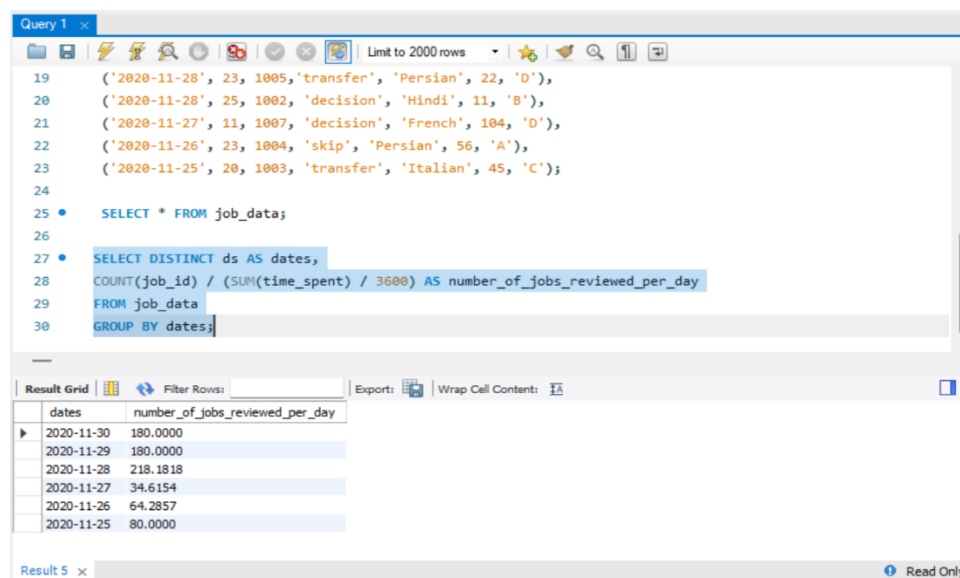
A snapshot of both the query as well as the obtained output has been listed for each question posed by the management team. Each insight observed is highlighted in blue colour. Following insights can be inferred from each query:

Case Study 1 - Job Data Analysis:

- Jobs Reviewed Over Time:

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

Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.



```
19 ('2020-11-28', 23, 1005, 'transfer', 'Persian', 22, 'D'),
20 ('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),
21 ('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),
22 ('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),
23 ('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');
24
25 • SELECT * FROM job_data;
26
27 • SELECT DISTINCT ds AS dates,
28 COUNT(job_id) / (SUM(time_spent) / 3600) AS number_of_jobs_reviewed_per_day
29 FROM job_data
30 GROUP BY dates;
```

dates	number_of_jobs_reviewed_per_day
2020-11-30	180.0000
2020-11-29	180.0000
2020-11-28	218.1818
2020-11-27	34.6154
2020-11-26	64.2857
2020-11-25	80.0000

As per the output, number of jobs reviewed per hour for each day in November continuously varies. The highest job reviews of 218 are done on 28th November and the lowest job reviews of 34 are done on 27th November.

- Throughput Analysis:

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

Your Task: Write an SQL query to calculate the 7-day rolling average of throughput.

Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

Query 1

```

26
27 • SELECT DISTINCT ds AS dates,
28 COUNT(job_id) / (SUM(time_spent) / 3600) AS number_of_jobs_reviewed_per_day
29 FROM job_data
30 GROUP BY dates;
31
32 • SELECT a.ds AS DAY,
33 a.throughput,
34 AVG(a.throughput) OVER ( ORDER BY ds rows BETWEEN 6 PRECEDING AND CURRENT row ) AS 7_day_avg_of_throughput
35 FROM
36 ( SELECT ds, COUNT(job_id) / SUM(time_spent) AS throughput FROM job_data GROUP BY ds ) AS a
37 GROUP BY ds;

```

Result Grid

DAY	throughput	7_day_avg_of_throughput
2020-11-25	0.0222	0.02220000
2020-11-26	0.0179	0.02005000
2020-11-27	0.0096	0.01656667
2020-11-28	0.0606	0.02757500
2020-11-29	0.0500	0.03206000
2020-11-30	0.0500	0.03505000

Result 7

Daily Metric provides the raw throughput values for each day. It's useful when you need the most up-to-date data or when analyzing specific events that might affect throughput on a particular day.

7-Day Rolling Average smooths out the daily fluctuations and provides a clearer view of the trend over time. It's beneficial for identifying the underlying trends and patterns, especially in volatile data where daily metrics might be misleading due to short-term spikes or dips.

Personally, I find the 7-day rolling average more informative for strategic decision-making as it reduces the "noise" in the data and highlights longer-term movements, which are often more actionable than daily variations. However, for real-time monitoring or response to immediate issues, the daily metric would be more appropriate.

- Language Share Analysis:

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

Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

Query 1

```

32 • SELECT a.ds AS DAY,
33 a.throughput,
34 AVG(a.throughput) OVER ( ORDER BY ds rows BETWEEN 6 PRECEDING AND CURRENT row ) AS 7_day_avg_of_throughput
35 FROM
36 ( SELECT ds, COUNT(job_id) / SUM(time_spent) AS throughput FROM job_data GROUP BY ds ) AS a
37 GROUP BY ds;
38
39 • SELECT language,
40 COUNT(job_id)*100 / SUM(COUNT(*)) OVER() AS percentage_share
41 FROM job_data
42 WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
43 GROUP BY language;

```

Result Grid

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

Result 9

According to the output, Persian is a popularly used language for content with a percentage share of 37.5%, whereas, the rest of the languages like English, Arabic, Hindi, French, Italian have a percentage share of 12.5%.

- Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

Your Task: Write an SQL query to display duplicate rows from the job_data table.

```
56
57 • SELECT ds, job_id, actor_id, event, language, time_spent, org, COUNT(*) AS row_occurrence
58 FROM job_data
59 GROUP BY ds, job_id, actor_id, event, language, time_spent, org
60 HAVING COUNT(*) > 1;
```

Result Grid

ds	job_id	actor_id	event	language	time_spent	org	row_occurrence
----	--------	----------	-------	----------	------------	-----	----------------

Result 11 x Read Only

According to the output, there are no duplicate rows in the data.

Case Study 2 - Investigating Metric Spike:

1. Weekly User Engagement:

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

Your Task: Write an SQL query to calculate the weekly user engagement.

```
94
95 • SELECT week(occurred_at) as Week,
96 COUNT(DISTINCT user_id) as Weekly_User_engagement
97 FROM events
98 GROUP BY week(occurred_at)
99 ORDER BY week(occurred_at);
```

Result Grid

Week	Weekly_User_engagement
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

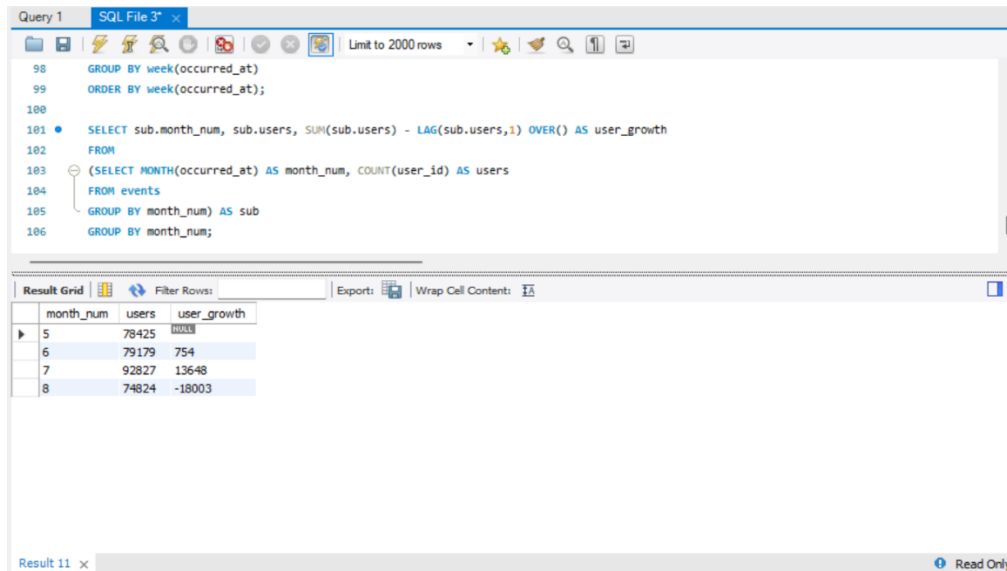
Result 9 x Read Only

The average weekly user engagement is 1349. The highest engagement was seen on week 30, with 1706 users. The lowest engagement was seen on week 17, with 740 users.

2. User Growth Analysis:

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

Your Task: Write an SQL query to calculate the user growth for the product.



The screenshot shows a SQL query editor with a query that calculates user growth by month. The query uses a subquery to group users by month and then uses a window function to calculate the growth from the previous month. The result grid shows the following data:

month_num	users	user_growth
5	78425	NULL
6	79179	754
7	92827	13648
8	74824	-18003

I've ascertained the user count for the product in May (5th), June (6th), July (7th), and August (8th) of 2014, allowing us to deduce the monthly user growth. On June 14th, there was a decrease of 3,034 users. However, the following month saw an upturn, with a rise of 2,725 users on July 14th. Yet, on August 14th, there was a sharp decline in user growth, plummeting by 13,418 users.

3. Weekly Retention Analysis:

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

Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Since the events dataset started from May 2014, I used only users who signed up on or after May 2014.

```
SELECT user_id,  
activated_at  
FROM users  
WHERE activated_at > '2014-05-01'  
ORDER BY user_id;
```

• Then the user activity was extracted from events column for user who signed up after May 2014

```
SELECT DISTINCT u.user_id,  
e.occurred_at  
FROM users u join events e on u.user_id = e.user_id  
WHERE u.activated_at > '2014-05-01' and e.event_name = 'login'  
GROUP BY week( e.occurred_at)  
ORDER BY e.occurred_at;
```

Then, the retention rate was calculated in Tableau to create the user sign up cohort chart

Upon conducting SQL queries in the workbench editor, I determined the weekly user count throughout 2014. This information reveals the weekly retention rates of users post-sign-up for a product. The data shows numerous instances of user retention (positive figures in the user_retention column) and also instances of user drop-off (negative figures in the user_retention column).

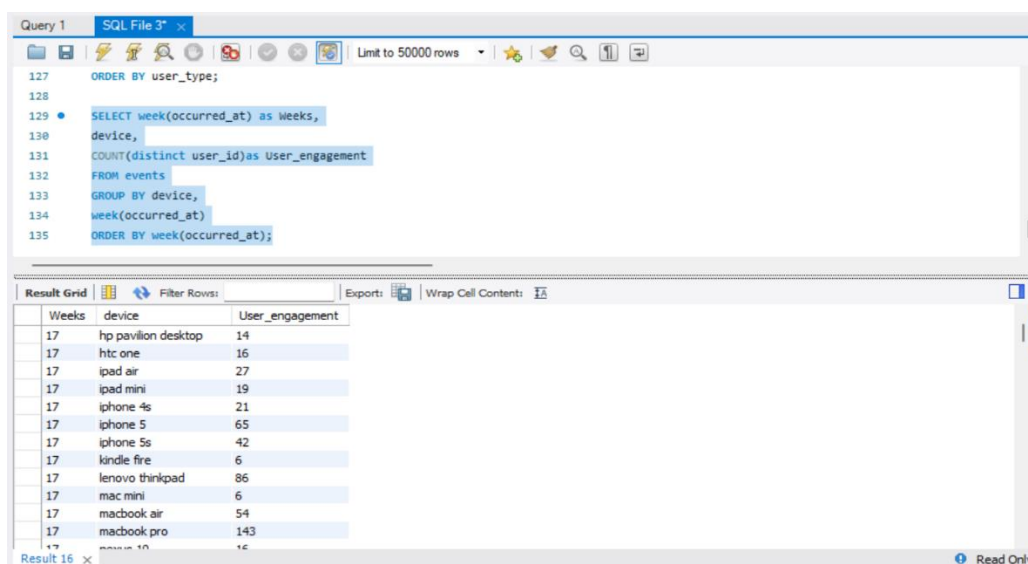
4. Weekly Engagement Per Device:

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

Your Task: Write an SQL query to calculate the weekly engagement per device.

I've tracked user engagement with the product across all devices from the 17th to the 35th week of the year. Users engage in various activities, such as visiting the homepage, liking messages, and logging in. This data, sourced from the diverse IP addresses of users' devices, categorizes devices—and consequently users—into three main types.

This study is instrumental in assessing weekly user activeness and their perception of the product's quality. It appears users remain active weekly only if the product meets their quality expectations. The data on weekly user engagement (user_engagement_device column) and device-specific weekly engagement (user_engagement_wk column) provide insights into the extent and device preference of user interaction. Such analysis is valuable for enhancing the product's user experience (UX).



The screenshot shows a SQL IDE interface. The top pane displays a SQL query (lines 127-135) that calculates weekly user engagement per device. The bottom pane shows the 'Result Grid' with the following data:

Weeks	device	User_engagement
17	hp pavilion desktop	14
17	htc one	16
17	ipad air	27
17	ipad mini	19
17	iphone 4s	21
17	iphone 5	65
17	iphone 5s	42
17	kindle fire	6
17	lenovo thinkpad	86
17	mac mini	6
17	macbook air	54
17	macbook pro	143
17	msi gt72	16

5. Email Engagement Analysis:

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

Your Task: Write an SQL query to calculate the email engagement metrics.

I've determined the count of users linked to various actions tied to email-sending events, categorized by user types. This measure is instrumental in gauging the engagement levels of users within identical user types—those utilizing comparable devices—with each distinct action related to email dispatch.

Query 1 SQL File 3"

Limit to 2000 rows

```

120
121 • SELECT email_events.user_type,
122       action,
123       COUNT(email_events.user_id) AS users
124 FROM
125       email_events
126 GROUP BY action, user_type
127 ORDER BY user_type;

```

Result Grid Filter Rows: Export: Wrap Cell Contents: Read Only

	user_type	action	users
1	1	sent_weekly_digest	18412
1	1	email_open	6511
1	1	email_clickthrough	2758
1	1	sent_reengagement_email	892
2	2	sent_weekly_digest	15232
2	2	email_open	5562
2	2	email_clickthrough	2521
2	2	sent_reengagement_email	1071
3	3	sent_weekly_digest	23623
3	3	email_open	8386
3	3	email_clickthrough	3731
3	3	sent_reengagement_email	1690

Result 14 x

Result

This project deepened my comprehension of the critical role data analysis plays in enabling organizations to make informed data-driven decisions. Through this project, which utilized Operational data, I gained insights into a variety of queries. I learned how to use MySQL Workbench on my local machine and its features and functionalities. Consecutively, I have learnt the importance of user engagement and how it can be increased with the help of data analytics