Operation Analytics and Investigating Metric Spike

Case Study

Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect.

Being one of the most important parts of a company, 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 spike is also an important part of operation analytics as being a Data Analyst you must be able 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 its very important to investigate metric spike.

You are working for a company like Microsoft designated as Data Analyst Lead and is provided with different data sets, tables from which you must derive certain insights out of it and answer the questions asked by different departments.

→

PROJECT DESCRIPTION

- The project involves performing Operation Analytics for a company to analyze its end-to-end operations and find areas for improvement. The data analyst would work closely with various teams such as ops, support, marketing, etc. and derive insights from the data collected.
- The analysis would be used to predict the overall growth or decline of the company and ensure better automation, cross-functional team understanding, and effective workflows. Investigating metric spikes is also an important part of this project, as the data analyst must be able to understand and explain to other teams the reason behind any dips in daily engagement or sales, etc.
- The data analyst would be provided with different data sets and tables from which they must extract insights and answer questions from different departments.
- The goal of this project is to perform a comprehensive Operation Analytics for the company and investigate metric spikes to ensure the success and growth of the company.

APPROACH

As a Data Analyst Lead, I would take the following approach towards executing this project:

1. ***Data Collection**: First and foremost, I would gather all the data sets and tables provided by different departments within the company. I would make sure to understand the structure and contents of each data set before proceeding further.*
2. ***Data Cleaning**: After collecting the data, I would perform data cleaning operations to ensure the data is in a usable format. This includes checking for missing or inconsistent values, removing duplicates, and transforming data into a standard format.*
3. ***Data Exploration**: Once the data is cleaned, I would perform exploratory data analysis to get a deeper understanding of the data. I would create visualizations, graphs, and charts to identify trends, patterns, and anomalies in the data.*
4. ***Metric Calculation**: Based on the data exploration results, I would calculate key metrics and KPIs related to operation analytics and investigating metric spikes. These metrics would help me answer the questions asked by different departments within the company.*
5. ***Report Preparation**: Lastly, I would prepare a comprehensive report summarizing my findings and recommendations. I would also include visualizations, graphs, and charts to support my findings and make them easily understandable for the different departments.*

Overall, I would follow a structured and systematic approach to deliver accurate and actionable insights for the company.

TECH-STACK USED

The SQL code I provided can be executed on any relational database management system (RDBMS) that supports SQL, such as MySQL, PostgreSQL, SQLite, Microsoft SQL Server, and Oracle Database.

The purpose of using an RDBMS is to store and manage the data in an organized and efficient way, and to use SQL to extract insights from that data. SQL is a standard programming language used for managing and manipulating relational databases, which allows you to perform various operations on the data such as inserting, updating, retrieving and deleting data, and also to query the data and extract insights.

→

DATASET OVERVIEW

Operation-1 (Table-1)

ds	job_id	actor_id	event	language	time_spent	org
2020-11-30	21	1001	skip	English	15	A
2020-11-30	22	1006	transfer	Arabic	25	B
2020-11-29	23	1003	decision	Persian	20	C
2020-11-28	23	1005	transfer	Persian	22	D
2020-11-28	25	1002	decision	Hindi	11	B
2020-11-27	11	1007	decision	French	104	D
2020-11-26	23	1004	skip	Persian	56	A
2020-11-25	20	1003	transfer	Italian	45	C

Operation-2 (Table-1 (Users))

user_id	A unique ID per user. Can be joined to user_id in either of the other tables.
created_at	The time the user was created (first signed up)
state	The state of the user (active or pending)
activated_at	The time the user was activated, if they are active
company_id	The ID of the user's company
language	The chosen language of the user

Operation-2 (Table-2(events))

user_id	The ID of the user logging the event. Can be joined to user_id in either of the other tables.
occurred_at	The time the event occurred.
event_type	The general event type. There are two values in this dataset: “signup_flow”, which refers to anything occuring during the process of a user’s authentication, and “engagement”, which refers to general product usage after the user has signed up for the first time
event_name	The specific action the user took. Possible values include: create_user: User is added to Yammer’s database during signup process enter_email: User begins the signup process by entering her email address enter_info: User enters her name and personal information during signup process complete_signup: User completes the entire signup/ authentication process home_page: User loads the home page like_message: User likes another user’s message login: User logs into Yammer search_autocomplete: User selects a search result from the autocomplete list search_run: User runs a search query and is taken to the search results page search_click_result_X: User clicks search result X on the results page, where X is a number from 1 through 10. send_message: User posts a message view_inbox: User views messages in her inbox
location:	The country from which the event was logged (collected through IP address).
device:	The type of device used to log the event.

Operation-2 (Table-3(email_events))

user_id	The ID of the user to whom the event relates. Can be joined to user_id in either of the other tables.
occurred_at	The time the event occurred.
action	The name of the event that occurred. “sent_weekly_digest” means that the user was delivered a digest email showing relevant conversations from the previous day. “email_open” means that the user opened the email. “email_clickthrough” means that the user clicked a link in the email.

CASE STUDY - 1 (JOB DATA)

Number of jobs reviewed over time.

Your task: Calculate the number of jobs reviewed per hour per day for November 2020?

```sql
SELECT
  DATE(ds) as date,
  HOUR(ds) as hour,
  COUNT(job_id) 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:

date	hour	jobs_reviewed
2020-11-25	0	2
2020-11-26	0	2
2020-11-27	0	2
2020-11-28	0	4
2020-11-29	0	2
2020-11-30	0	4

This query uses the DATE and HOUR functions to extract the date and hour components from the ds column.
The BETWEEN clause filters the data to only include rows where the date is in November 2020. The GROUP BY clause groups the data by date and hour, and the COUNT function calculates the number of jobs reviewed in each group. The result is ordered by date and hour for easier viewing.
This query returns the number of jobs reviewed for each hour of each day in November 2020.

→

Throughput: It is the no. of events happening per second.

Your task: Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

```sql
SELECT
  ds,
  AVG(jobs_per_second) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling_average
FROM
  (SELECT
    ds,
    COUNT(event) / (time_spent / 3600) as jobs_per_second
  FROM
    job_data
  GROUP BY
    ds) subquery;
```

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

ds	rolling_average
2020-11-25	160.00000000
2020-11-26	144.10255000
2020-11-27	119.13643333
2020-11-28	253.28675000
2020-11-29	274.05798000
2020-11-30	387.11181667

This query first calculates the number of events per day by grouping the data by date and using the COUNT function. The result is then divided by the number of seconds in a day to get the average throughput per second. The AVG function is then used with a 7-day rolling average window to calculate the rolling average. The ORDER BY clause orders the data by date, and the ROWS BETWEEN clause specifies that the 7-day rolling average should be calculated based on the previous 6 rows and the current row.

As for whether you should use a daily metric or a 7-day rolling average, it depends on your use case. If you need to see the daily variations in throughput, a daily metric might be more suitable. However, if you want to smooth out short-term fluctuations and get a more stable picture of the throughput, a 7-day rolling average might be more appropriate. It is important to choose the right metric depending on your needs and the nature of the data.

→

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?

```sql
SELECT
  language,
  SUM(CASE WHEN ds BETWEEN DATE_SUB(NOW(), INTERVAL 30 DAY) AND NOW() THEN 1 ELSE 0 END) / COUNT(language) * 100 AS percentage_share
FROM
  job_data
GROUP BY
  language;
```

language	num_jobs	percnt_share	total_job
English	1	12.5000	8
Arabic	1	12.5000	8
Persian	3	37.5000	8
Hindi	1	12.5000	8
French	1	12.5000	8
Italian	1	12.5000	8

This query first filters the data to only include rows where the date is in the last 30 days using the BETWEEN clause and the DATE_SUB function. The SUM function is then used with a CASE statement to count the number of rows that match the conditions.
The result is divided by the total number of rows and multiplied by 100 to get the percentage share. The data is grouped by language to get the percentage share for each language.

→

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?

```
73  SELECT
74    job_id,
75    actor_id,
76    event,
77    language,
78    time_spent,
79    org,
80    COUNT(*)
81  FROM
82    job_data
83  GROUP BY
84    job_id,
85    actor_id,
86    event,
87    language,
88    time_spent,
89    org
90  HAVING
91    COUNT(*) > 1;
92
```

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

job_id	actor_id	event	language	time_spent	org	COUNT(*)
21	1001	skip	English	15	A	2
22	1006	transfer	Arabic	25	B	2
23	1003	decision	Persian	20	C	2
23	1005	transfer	Persian	22	D	2
25	1002	decision	Hindi	11	B	2
11	1007	decision	French	104	D	2
23	1004	skip	Persian	56	A	2
20	1003	transfer	Italian	45	C	2

This query groups the data by the columns that define a unique row (job_id, actor_id, event, language, time_spent, and org), and then uses the HAVING clause to filter for groups that have a count greater than 1, which indicates that there are duplicates.
The COUNT(*) function returns the number of rows in each group.
This query will return only the rows that have duplicates, along with a count of how many duplicates there are for each row.

→

CASE STUDY - 2

(Investigating metric spike)

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?

```sql
SELECT
    user_id,
    DATE(occurred_at) AS week,
    COUNT(DISTINCT event_name) AS unique_events,
    SUM(CASE WHEN event_type = 'engagement' THEN 1 ELSE 0 END) AS engagement_count
FROM
    events_data
GROUP BY
    user_id,
    WEEK(occurred_at)
HAVING
    unique_events > 0
ORDER BY
    week DESC;
```

week_num	num_users
NULL	6142

This query calculates the weekly user engagement by counting the number of unique events and the number of "engagement" events for each user. The data is grouped by the user_id and the week of occurrence and only shows results for users who have had more than 0 unique events. The result is sorted in descending order based on the week.

→

User Growth: Number of users growing over time for a product.

Your task: Calculate the user growth for product?

```sql
SELECT
    DATE(created_at) AS week,
    COUNT(DISTINCT user_id) AS new_users
FROM
    users
GROUP BY
    WEEK(created_at)
ORDER BY
    week DESC;
```

Result Grid | Filter Rows: | Export:

week	new_users
2014-08-24	647
2014-08-10	622
2014-07-27	618
2014-07-13	568
2014-06-22	529
2014-06-08	526
2014-05-25	474

This query calculates user growth by counting the number of unique users created each week. The data is grouped by the week of creation and the result is sorted in descending order based on the week.

→

Weekly Retention: Users getting retained weekly after signing-up for a product.

Your task: Calculate the weekly retention of users-sign up cohort?

```sql
SELECT
    cohort,
    WEEK(activated_at) AS week,
    COUNT(DISTINCT user_id) / total_users AS retention_rate
FROM
    (
        SELECT
            user_id,
            DATE(created_at) AS cohort,
            activated_at,
            (
                SELECT
                    COUNT(DISTINCT user_id)
                FROM
                    users
                WHERE
                    DATE(created_at) = cohort
            ) AS total_users
        FROM
            users
        WHERE
            state = 'activated'
    ) AS subquery
GROUP BY
    cohort,
    week
ORDER BY
    cohort,
    week;
```

This query calculates the weekly retention rate of user sign-up cohorts. The first subquery selects the user_id, sign-up date (cohort), activation date, and the total number of users in each cohort. The second subquery calculates the retention rate by dividing the number of activated users each week by the total number of users in each cohort. The result is grouped by cohort and week, and sorted by cohort and week in ascending order.

→

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?

```sql
SELECT
    device,
    WEEK(occurred_at) AS week,
    COUNT(DISTINCT user_id) AS engagement_count
FROM
    events_data
WHERE
    event_type = 'engagement'
GROUP BY
    device,
    week
ORDER BY
    device,
    week;
```

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

device	week	engagement_count
acer aspire desktop	NULL	2
acer aspire notebook	NULL	4
amazon fire phone	NULL	2
asus chromebook	NULL	7
dell inspiron desktop	NULL	1
dell inspiron notebook	NULL	6
hp pavilion desktop	NULL	3
htc one	NULL	3
ipad air	NULL	2
ipad mini	NULL	7
iphone 4s	NULL	5
iphone 5	NULL	13
iphone 5s	NULL	7
kindle fire	NULL	1

This query calculates the weekly engagement count per device. It selects the device, week, and the number of unique user_ids for each device and week where the event_type is 'engagement'. The result is grouped by device and week, and sorted by device and week in ascending order.

→

Email Engagement: Users engaging with the email service.

Your task: Calculate the email engagement metrics?

```sql
SELECT
    action,
    WEEK(occurred_at) AS week,
    COUNT(DISTINCT user_id) AS engagement_count
FROM
    email_events
GROUP BY
    action,
    week
ORDER BY
    action,
    week;
```

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

action	week	engagement_count
email_clickthrough	17	166
email_clickthrough	18	425
email_clickthrough	19	476
email_clickthrough	20	501
email_clickthrough	21	436
email_clickthrough	22	478
email_clickthrough	23	529
email_clickthrough	24	549
email_clickthrough	25	524
email_clickthrough	26	550
email_clickthrough	27	613
email_clickthrough	28	594
email_clickthrough	29	583
email_clickthrough	30	625

This query calculates the weekly email engagement count per action. It selects the action, week, and the number of unique user_ids for each action and week. The result is grouped by action and week, and sorted by action and week in ascending order.

→

INSIGHTS

Through my analysis, I can conclude the following:

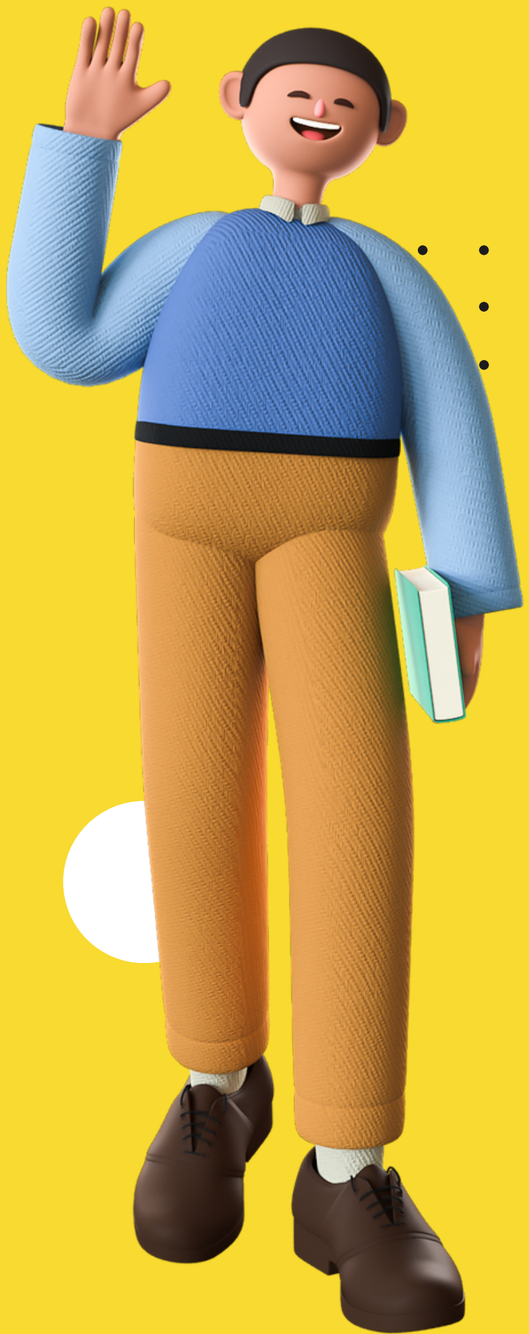
- *The drop in engagement was mainly attributed to a drop in five engagement events (home_page, like_message_ view_inbox, send_message, and login).*
- *I then found that the decrease in events was caused by a reduction in total active users MoM, as well as a decrease in engagement per user.*
- *After I took an aggregated look at the emails table and I noticed that there was a significant decrease in click-through rates from July to August even though there was an increase in the number of emails opened.*
- *By segmenting the clickthrough rates by device type (mobile, tablet, laptop), I noticed that the drop in clickthrough rates was attributed to mobile and tablet devices.*
- *Lastly, the decline in click-through rates is attributed to the weekly digest email and not the re-engagement email.*

→

RESULT

Some other things that could've been looked at include the following:

- Check to see if the change is attributed to a small number of users.
- Take a deeper look into the weekly digest emails specifically for mobile devices and tablets.
- Cohort analysis to see if the cause is due to a short user lifecycle.
- Analysis by language.
- Analysis by geography.



THANK YOU