Project on Operation Analytics and Investigating Metric Spike

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

One of the critical aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales. As a Data Analyst, it is our job to answer these questions daily, making it crucial to understand how to investigate these metric spikes.

- ***Case Study 1: Job Data Analysis***

Here, we will be working with a table named job_data with the following columns:

job_id: Unique identifier of jobs
actor_id: Unique identifier of actor
event: The type of event (decision/skip/transfer).
language: The Language of the content
time_spent: Time spent reviewing the job in seconds.
org: The Organization of the actor
ds: The date in the format yyyy/mm/dd (stored as text)

The problems

A)Jobs Reviewed Over Time:
Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.
Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

B)Throughput Analysis:
Objective: Calculate the 7-day rolling average of throughput (number of events per second).
Task: Write an SQL query to calculate the 7-day rolling average of throughput. Also, explain why you prefer using the daily metric or the 7-day rolling average for throughput.

C)Language Share Analysis:

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

D)Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.
Task: Write an SQL query to display duplicate rows from the job_data table.

- ***Case Study 2: Investigating Metric Spike***

Here, we will be working with three tables:

users: Contains one row per user, with descriptive information about that user's account.
events: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).
email_events: Contains events specific to the sending of emails.

1. **Weekly User engagement:**

Objective: Measure the activeness of users on a weekly basis.
Task: Write an SQL query to calculate the weekly user engagement.

2. **User Growth Analysis:**

Objective: Analyze the growth of users over time for a product.
Task: Write an SQL query to calculate the user growth for the product.

3. **Weekly Retention Analysis:**

Objective: Analyze the retention of users on a weekly basis after signing up for a product.
Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

4. **Weekly Engagement Per Device:**

Objective: Measure the activeness of users on a weekly basis per device.
Task: Write an SQL query to calculate the weekly engagement per device.

5. **Email Engagement Analysis:**

Objective: Analyze how users are engaging with the email service.
Task: Write an SQL query to calculate the email engagement metrics.

Database and tools:

- ☐ MySQL
- ☐ MySQL Workbench 8.0 CE

Software used for visualisation:

- ☐ Microsoft Excel

Queries and outputs of Case study-I

- ***Query to Create database and table:***

```sql
create database project3;
show databases;
use project3;
create table job_data (
ds datetime,
job_id int,
actor_id int,
event varchar(50),
language varchar(100),
time_spent int,
org varchar(10)
);
```

A)Jobs Reviewed Over Time:

```sql
SELECT
   ds, COUNT(job_id), sum(time_spent) / 3600 AS time_spent_in_hr
FROM
    job_data
    group by 1
    order by 1 asc;
```

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

ds	COUNT(job_id)	time_spent_in_hr
11/25/2020	1	0.0125
11/26/2020	1	0.0156
11/27/2020	1	0.0289
11/28/2020	2	0.0092
11/29/2020	1	0.0056
11/30/2020	2	0.0111

B)Throughput Analysis:

```sql
# Calculate the 7-day rolling average of throughput (number of events per second)

WITH throughput AS
( SELECT ds, COUNT(job_id) AS total_job, SUM(time_spent) AS total_time
FROM job_data
GROUP BY 1)
SELECT ds, SUM(total_job) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) /
SUM(total_time) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
AS 7_days_rolling_average FROM throughput;
```

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

ds	7_days_rolling_average
11/25/2020	0.0222
11/26/2020	0.0198
11/27/2020	0.0146
11/28/2020	0.0210
11/29/2020	0.0233
11/30/2020	0.0268

Throughput analysis by 7-day rolling average gives a general idea about the average of consecutive 7 days whereas the daily metric gives information about data on a daily basis.
In some cases, where we need an idea about how a parameter is changing over the course of time, it's better to opt for the rolling average than the daily metric.

C)Language Share Analysis:

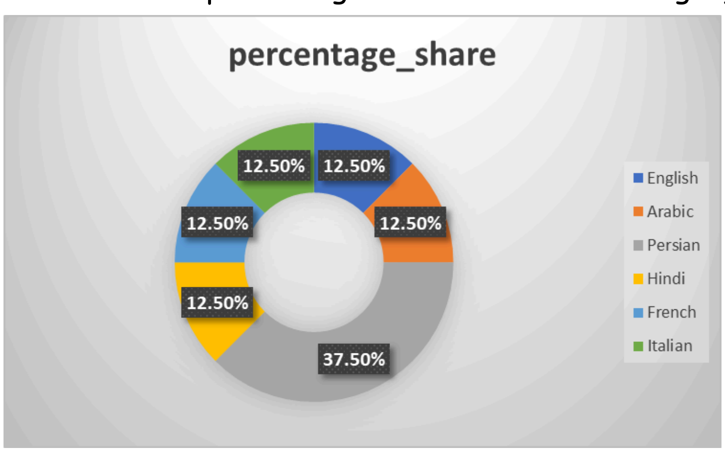
```sql
#Calculate the percentage share of each language in the last 30 days

with count_of_language as
(
select
language,
count(language) as languages
from job_data
group by 1
)
select language,
concat(languages*100/(select sum(languages) as total from count_of_language),'%') as percentage_share
from count_of_language
group by 1;
```

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

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

The percentage share of Persian is the maximum among all the languages and its value is 3 times the percentage share of all other languages.



D)Duplicate Rows Detection:

```
# Duplicate rows:

with duplicate_value as
(
select ds, job_id, actor_id,
row_number() over(partition by job_id order by job_id) as row_no
from job_data
)
select*
from duplicate_value
where row_no > 1;
```

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

ds	job_id	actor_id	row_no
11/28/2020	23	1005	2
11/26/2020	23	1004	3

Query and Output of Case study-II

- ***Query to Create database, table and loading data into the table***

```sql
use project3;
create table users
(
user_id int,
created_at varchar(100),
company_id int,
language varchar(50),
activated_at varchar(100),
state varchar(50)
);


SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
alter table users add column temp_created_at datetime;
update users set temp_created_at = str_to_date(created_at, '%d-%m-%Y %H:%i:');
alter table users drop column created_at;
alter table users change temp_created_at created_at datetime;
alter table users add column temp_activated_at datetime;
update users set temp_activated_at = str_to_date(activated_at, '%d-%m-%Y %H:%i:');
alter table users drop column activated_at;
alter table users change temp_activated_at activated_at datetime;

create table events(
user_id int,
occurred_at varchar(100),
event_type varchar(100),
event_name varchar(50),
location varchar(50),
device  varchar(50),
user_type int
);
LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/events.csv"
INTO TABLE events
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
```

```
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;


alter table events add column temp_created_at datetime;
update events set temp_created_at = str_to_date(occurred_at, '%d-%m-%Y %H:%i:');
alter table events drop column occurred_at;
alter table events change temp_created_at occurred_at datetime;

create table email_events
(
user_id int,
occurred_at varchar(100),
action  varchar(100),
user_type int
);

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv"
INTO TABLE email_events
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

alter table email_events add column temp_created_at datetime;
update email_events set temp_created_at = str_to_date(occurred_at, '%d-%m-%Y %H:%i:');
alter table email_events drop column occurred_at;
alter table email_events change temp_created_at occurred_at datetime;
```

1. Weekly User engagement:

```sql
# weekly user engagement

select extract(week from occurred_at) as week_no,
count(distinct user_id)
from events
where event_type='engagement'
group by 1
order by 1;
```

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

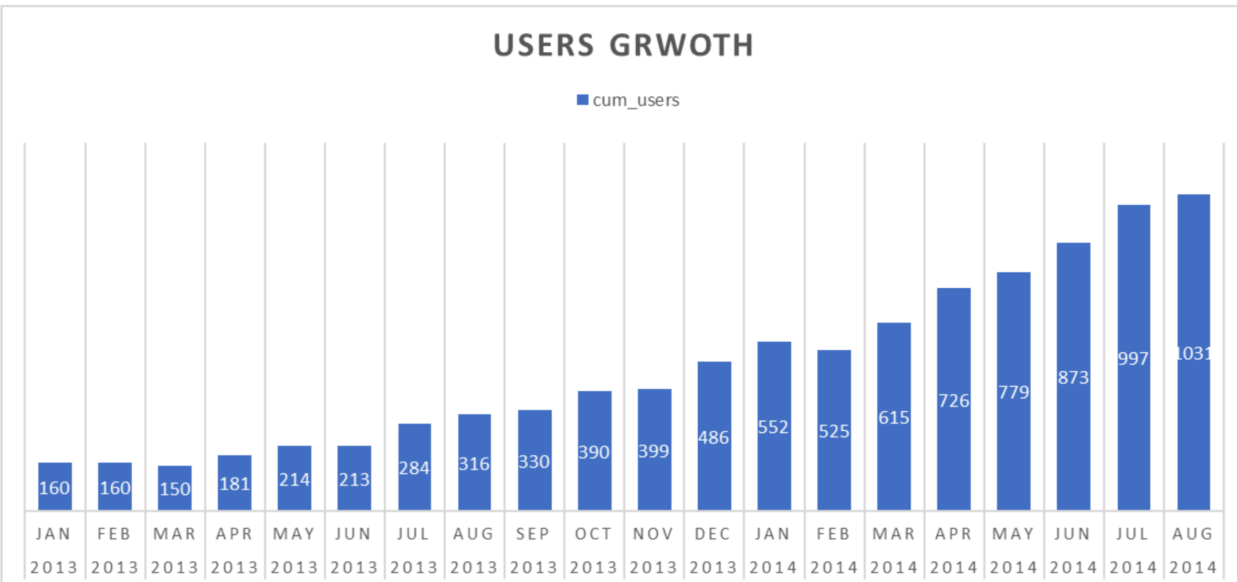
week_no	count(distinct user_id)
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376

2. User Growth Analysis:

```sql
# USER GROWTH ANALYSIS

with growth as
(
select
extract(year from activated_at) as year_no,
extract(month from activated_at) as month_no,
count(distinct user_id) as cum_users
from users
group by 1,2
)
select*, sum(cum_users) over(order by year_no, month_no rows between unbounded preceding and current row) as cumulative_users
from growth;
```

year_no	month_no	cum_users	cumulative_users
2013	1	160	160
2013	2	160	320
2013	3	150	470
2013	4	181	651
2013	5	214	865
2013	6	213	1078
2013	7	284	1362
2013	8	316	1678
2013	9	330	2008
2013	10	390	2398
2013	11	399	2797
2013	12	486	3283
2014	1	552	3835
2014	2	525	4360
2014	3	615	4975
2014	4	726	5701
2014	5	779	6480
2014	6	873	7353
2014	7	997	8350
2014	8	1031	9381



3. Weekly Retention Analysis:

```sql
# WEEKLY USER RETENTION

SELECT
a.cohort_week,
b.engagement_week,
count(distinct a.user_id) as engagement
FROM
((SELECT distinct user_id, extract(week from occurred_at) as cohort_week from events
WHERE event_type = 'signup_flow'
and event_name = 'complete_signup') as a
LEFT JOIN
(SELECT distinct user_id, extract(week from occurred_at) as engagement_week FROM events
where event_type = 'engagement') as b
on a.user_id = b.user_id)
group by 1,2
order by 1,2
```

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

cohort_week	engagement_week	engagement
17	17	72
17	18	59
17	19	24
17	20	16
17	21	11
17	22	16
17	23	11
17	24	9
17	25	6
17	26	8
17	27	8
17	28	8

4. Weekly Engagement Per Device:

```sql
# weekly engagement per device
select
extract(year from occurred_at) as year_no,
extract(week from occurred_at) as week_no,
device,
count(distinct user_id)
from events
where event_type='engagement'
group by 1,2,3
order by 1;
```

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

year_no	week_no	device	count(distinct user_id)
2014	17	acer aspire desktop	9
2014	17	acer aspire notebook	20
2014	17	amazon fire phone	4
2014	17	asus chromebook	21
2014	17	dell inspiron desktop	18
2014	17	dell inspiron notebook	46
2014	17	hp pavilion desktop	14
2014	17	htc one	16
2014	17	ipad air	27
2014	17	ipad mini	19
2014	17	iphone 4s	21
2014	17	iphone 5	65
2014	17	iphone 5s	42
2014	17	kindle fire	6
2014	17	lenovo thinkpad	86
2014	17	mac mini	6
2014	17	macbook air	54

Knowing the engagement per device can help us make inferences like what device most of the users are using and if a certain device has fewer users then it can be an indication that we need to improve our services for those devices.

5. Email Engagement Analysis:

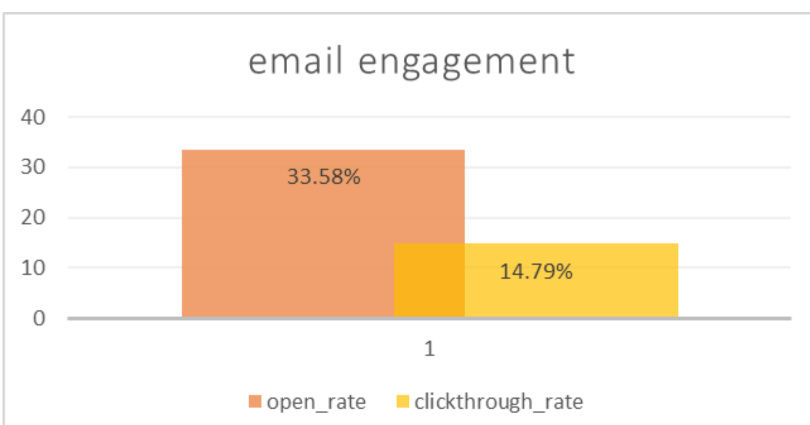
```sql
#email engagement

select
100.0*sum(case when emails='opened' then 1 else 0 end)/ sum(case when emails='sent' then 1 else 0 end ) as open_rate,
100.0*sum(case when emails='clicked' then 1 else 0 end)/ sum(case when emails='sent' then 1 else 0 end ) as clickthrough_rate
from
(select
case
when action='email_clickthrough' then 'clicked'
when action='email_open' then 'opened'
when action in ('sent_reengagement_email' , 'sent_weekly_digest' ) then 'sent'
end as emails
from email_events)x;
```

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

open_rate	clickthrough_rate
33.58339	14.78989

Here, it shows that the opening rate of emails is 33.58%, i.e. 33 out of 100 users have opened the email. The Click-through rate (CTR) is 14.78%, which is the percentage of users who click on the link in the email. This CTR here, indicates that we need to have a more catchy subject line and captivating body of our email personalized for each user to make our users click the link and boost our CTR.



Conclusion

We can conclude that analyzing operation analytics and investigating metric spikes is crucial for understanding business growth so that we can work on our areas of improvement to provide a better experience to users/clients.

So, it is very important that any organization analyze it on a periodic basis(weekly or monthly or quarterly or yearly) based on their needs for making better decisions for their company/business growth.