Operation Analytics and Investigating Metric Spike

Introduction:

Operational Analytics plays a pivotal role in enhancing organizational efficiency by dissecting the entirety of a company's operations. Businesses can use this analytical method as a compass to identify areas that are ready for optimization and improvement. Your work as a data analyst is varied and requires you to work closely with various departments, including marketing, operations, and support. Utilizing the abundance of data that these teams have gathered and turning it into useful insights is your main goal.

Project Description:

Understanding and improving a company's end-to-end operations depend heavily on operational analytics. Working with a variety of departments, such as operations, support, and marketing, is the main goal of a data analyst's job to extract useful insights from data. This helps the business make well-informed decisions and optimize its processes.

Examining abrupt spikes in metrics is a crucial component of operational analytics, and it calls for a thorough grasp of data analysis methods. For this project, you will be working for a fictitious corporation that is similar to Microsoft as a Lead Data Analyst. You have to use your sophisticated SQL skills to examine the datasets that have been provided, providing operational improvement advice and insight into metric changes.

Approach:

A. 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.

job_data creation | Jobs Reviewed Over Time | Throughput Analysis | Language Share Analysis | Duplicate Rows Detection*

Limit to 1000 rows

```sql
-- Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 202
use job_db;

select * from job_data;

select ds, count(job_id) as count_jobs, sum(job_id)/3600 as reviewed_in_hours from job_data
where ds>='2020-11-01' and ds<='2020-11-30'
group by ds
```

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

ds	count_jobs	reviewed_in_hours
2020-11-30	2	0.0119
2020-11-29	1	0.0064
2020-11-28	2	0.0133
2020-11-27	1	0.0031
2020-11-26	1	0.0064
2020-11-25	1	0.0056

B. 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.
  The daily throughput

job_data creation | Jobs Reviewed Over Time | Throughput Analysis* | Language Share

Limit to 1000 rows

```
select ds,count(event)/sum(time_spent)
from job_data;


select ds,count(event)/sum(time_spent) as throughput
from job_data
group by ds
```

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

ds	throughput
2020-11-30	0.0500
2020-11-29	0.0500
2020-11-28	0.0606
2020-11-27	0.0096
2020-11-26	0.0179
2020-11-25	0.0222

And the highest being 0.0606

Rolling metrics serve as a powerful tool for monitoring and interpreting daily fluctuations in key performance indicators. By smoothing out short-term variations and highlighting longer-term trends, they offer valuable insights into the trajectory of your metrics, empowering you to make informed decisions and drive continuous improvement.

C. **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.

job_data creation | Jobs Reviewed Over Time | Throughput Analysis | Language Share Analysis* | Duplicate Rows Detection*

Limit to 1000 rows

```sql
use job_db;

select * from job_data;

select language,count(language) as count_lang, (count(language)/total_lang)*100 as percent_lang
from job_data
inner join(
    select count(language) as total_lang from job_data) as inner_table
on 1=1
where ds>='2020-11-01' and ds<='2020-11-30'
group by language,inner_table.total_lang
order by percent_lang desc;
```

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

language	count_lang	percent_lang
Persian	3	37.5000
English	1	12.5000
Arabic	1	12.5000
Hindi	1	12.5000
French	1	12.5000
Italian	1	12.5000

Here is the percentage of language used. With the highest being 37.50 by 'The Persian' language

D. 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

job_data creation | Jobs Reviewed Over Time | Throughput Analysis | Language Share Analysis* | Duplicate Rows Detection*

Limit to 1000 rows

```sql
-- duplicate rows in actor_id

select * from job_data
where actor_id in(
select actor_id from job_data
group by actor_id
having count(*)>1);
```

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

ds	job_id	actor_id	event	language	time_spent	org
2020-11-29	23	1003	decision	Persian	20	C
2020-11-25	20	1003	transfer	Italian	45	C

Actor id 1003 has 2 duplicate rows and here are the full details of those duplicate rows

Case Study 2: Investigating Metric Spike

A. Weekly User Engagement:

users-table | events-table | email_events-table | Weekly Engagement Per Device* | Weekly User Engagement*

Limit to 1000 rows

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

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

week_	no_of_users
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

Highest No_of_users in a week: 30
Lowest No_of_users in a week: 35

B. User Growth Analysis:

users-table | events-table | email_events-table | Weekly Engagement Per Device* | Weekly User Engagement* | Email Engagement

Limit to 1000 rows

```
select year_,week_,num_of_users,sum(num_of_users) over(order by year_,week_) as cumulative_users
from(
select year(created_at) as year_, week(created_at)as week_, count(distinct user_id) as num_of_users
from users
group by year_, week_) as inner_table
```

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

year_	week_	num_of_users	cumulative_users
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724

Result 1

Result Grid | Filter Rows: | Export

year_	week_	num_of_users	cumulative_users
2013	18	44	724
2013	19	57	781
2013	20	39	820
2013	21	49	869
2013	22	54	923
2013	23	50	973
2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813

The lowest was on week 35 of the year 2014 with no_of_users as 18 and the highest was on week 33 of the year 2014 with no_of_users 261.

C. 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

Here's the week by retention

```sql
select week(occurred_at) as week_by,
count(case when events.event_type='signup_flow' then events.user_id else null end) as signup_cohort,
count(case when events.event_type ='engagement' then events.user_id else null end) as engagement
from events
group by week_by;
```

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

week_by	signup_cohort	engagement
17	72	8019
18	163	17341
19	185	17224
20	176	17911
21	183	17151
23	196	18280
22	196	18413
24	229	19052
25	207	18642
29	221	20067
26	201	19061
30	238	21533
28	215	20776
27	222	19881
31	193	18556
32	245	16612
33	261	16145
34	259	16127
35	18	784

D. Weekly Engagement Per Device

```
select concat(year(occurred_at),'-',week(occurred_at)) as week_by,device,count(distinct user_id) as weekly
from events
where event_type='engagement'
group by week_by,device
order by week_by;
```

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

week_by	device	weekly_eng
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
2014-17	macbook pro	143
2014-17	nexus 10	16

Week 30 of the year 2014 has had the highest user engagement of 322 users, the device being used was “MacBook Pro” for the week.

E. Email Engagement Analysis

users-table | events-table | email_events-table | Weekly Engagement Per Device* | Weekly User Engagement* | Email Engagement Analysis*

Limit to 1000 rows

```
select (count_open/email_sent)*100 as per_open,(count_click/email_sent)*100 as per_click
from(
select count(case when action='email_open' then 1 end) as count_open
 from email_events) as open_table
join(
select count(case when action='email_clickthrough' then 1 end) as count_click,
count(case when action in('sent_weekly_digest', 'sent_reengagement_email') then 1 end) as email_sent
from email_events) click_table;
```

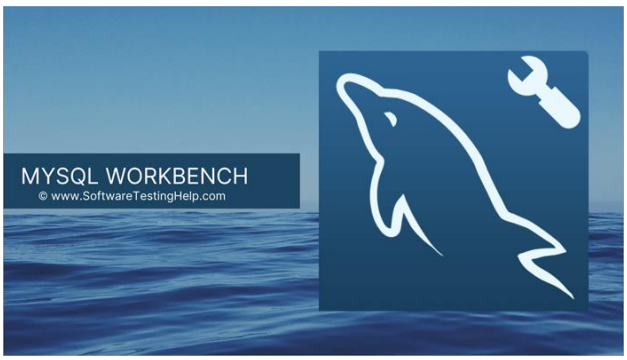
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

per_open	per_click
33.5834	14.7899

Out of the total emails sent 33.5834% of the emails were opened and 14.7899% were clicked This query calculates the percentage of emails opened and clicks based on the total number of emails sent.

Tech-Stack Used:

For this project, MySQL Workbench is utilized for data analysis. MySQL Workbench provides a user-friendly interface for querying databases, making it suitable for executing complex SQL queries. Its functionalities allow for efficient data manipulation, extraction, and visualization, facilitating a comprehensive analysis of the datasets provided.



Insights:

Through the analysis, several key insights and observations can be drawn from the data. These include patterns in user behavior, trends in metric fluctuations, and correlations between different operational variables. Insights gained from the analysis can shed light on areas for improvement, identify potential bottlenecks, and highlight opportunities for optimization within the company's operations.

Result:

This project helped me with SQL Proficiency and also made me more comfortable with writing complex SQL queries to extract valuable insights from diverse datasets. And I have also learned a lot about Time-Series Analysis, Investigating metrics over periods, such as hourly, daily, and weekly. The project aims to achieve a comprehensive understanding of the company's operations through data-driven insights. By successfully executing the provided tasks and generating meaningful analyses, the project contributes to informed decision-making processes within the organization. Ultimately, the insights gleaned from the data analysis endeavor to enhance operational efficiency and drive business growth

Conclusion:

This operational analytics project has been a worthwhile and rewarding experience that has provided a thorough immersion into the complexities of assessing a company's end-to-end operations. I have explored a variety of datasets, addressed a wide range of analytical issues, and produced useful insights to promote business progress throughout the project. I have improved our ability to query, aggregate, and analyze data by using advanced SQL techniques.

We have also become more adept at data visualization, time-series analysis, and problem-solving.