OPERATION & METRIC ANALYTICS

CONTENT

- Project Objective
- Downloaded the Data
- Import Excel Data into MySQL
- Analysis for Operation & metric Project
- Insights
- Recommendation

PROJECT OBJECTIVE

This project aims to apply advanced SQL and analytical skills to explore and understand sudden changes in key business metrics. By analyzing data from teams like operations, support, and marketing, it helps uncover trends and deliver insights that drive smarter decisions.

DOWNLOADED THE DATA

This data is downloaded from Trainity, an online learning platform for aspiring data analysts, designed to provide hands-on experience through live projects and virtual internships.

The link below is the data of MySQL, which is uploaded to GitHub.

GitHub: https://github.com/Manishtopno/Operation-Analytics-and-Investigating-Metric-Spike/blob/main/Operation%20Analytics.sql

IMPORT EXCEL DATA INTO MYSQL

- After downloaded the Excel data from Trainity.
- Load Excel data into Mysql.
- By **load data infile** statement in Mysql.

```
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;
```

This is the path of Excel data

Table name

ANALYSIS FOR OPERATION & METRIC PROJECT

To do the Analysis in the Operation & Metric Project, with the help of 2 case studies:

- Case study 1: Job Data Analysis
- Case study 2: Investigating Metric Spike

These are further subdivided into 4-5 analysis that help in finding key metrics and insights for this project.

CASE STUDY 1: JOB DATA ANALYSIS

TASK 1: Weekly User Engagement

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

```sql
SELECT
    ds AS DATE,
    COUNT(job_id),
    ROUND((SUM(time_spent) / 3600), 2) AS Total_time_spent_per_hour,
    ROUND((COUNT(job_id) / (SUM(time_spent) / 3600)),
            2) AS Job_review_per_day
FROM
    job_data
WHERE
    ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY ds
ORDER BY ds;
```

This is the SQL Query

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

DATE	COUNT(job_id)	Total_time_spent_per_hour	Job_review_per_day
2020-11-25	1	0.01	80.00
2020-11-26	1	0.02	64.29
2020-11-27	1	0.03	34.62
2020-11-28	2	0.01	218.18
2020-11-29	1	0.01	180.00
2020-11-30	2	0.01	180.00

This is the result of the above query

TASK 2: Throughput Analysis

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

```
select round(count(event)/sum(time_spent),2) as 'Daily throughput' from job_data
group by ds order by ds;
```

This is the SQL Query

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

Daily throughput
0.02
0.02
0.01
0.06
0.05
0.05

This is the result of the above Query

TASK 3: Language Share Analysis

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

```sql
select language, round(100*count(*)/total,2) as percentage, sub.total from job_data
cross join(select count(*) as total from job_data) as sub group by language, sub.total;
```

This is the SQL Query

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

language	percentage	total
English	12.50	8
Arabic	12.50	8
Persian	37.50	8
Hindi	12.50	8
French	12.50	8
Italian	12.50	8

This is the result of the above Query

TASK 4: Duplicate Rows Detection

- Identify duplicate rows in the data.

```
select actor_id, count(*) as duplicates from job_data
group by   actor_id having count(*)>1;
```

This is the SQL Query

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

	actor_id	duplicates
▶	1003	2

This is the result of the above Query

Case Study 2: Investigating Metric Spike

Task 1: Weekly User Engagement

- Measure the activeness of users on a weekly basis.

```sql
SELECT
    EXTRACT(WEEK FROM occurred_at) AS week_num,
    COUNT(DISTINCT user_id) AS active_users
FROM
    events
WHERE
    event_type = 'engagement'
GROUP BY week_num
ORDER BY week_num;
```

This is the SQL Query

Result Grid | Filter Row

week_num	active_users
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232

Result Grid | Filter Row

week_num	active_users
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467

Result Grid | Filter Row

week_num	active_users
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225

This is the result of the above Query

TASK 2: User Growth Analysis

- Analyze the growth of users over time for a product.

```sql
with Growth_of_users as(
SELECT extract(YEAR from created_at) as YEAR,
extract(WEEk from created_at) as week_number,
count(distinct user_id) as active_users
from users
group by YEAR, week_number)
select YEAR, week_number,active_users,
sum(active_users) over (order by YEAR, week_number) as cumulative_users
from Growth_of_users
order by YEAR, week_number;
```

This is the SQL Query

Result Grid | Filter Rows: | Export:

YEAR	week_number	active_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

This is the result of the above Query

Task 3: Weekly Retention Analysis

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

```sql
select extract(week from occurred_at) as weeks,
count(distinct user_id) as no_of_users
from events
where event_type="singnup_flow" or  event_name="complete_signup"
group by weeks order by weeks ;
```

This is the SQL Query

Result Grid | Filter Row

weeks	no_of_users
17	72
18	163
19	185
20	176
21	183
22	196
23	196

This is the result of the above Query

TASK 4: Weekly Engagement Per Device

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

```sql
select device, extract(week from occurred_at) as weeks,
count(distinct user_id) as no_of_users from events
where event_type="engagement"
group by device,weeks order by weeks;
```

This is the SQL Query

Result Grid | Filter Rows:

device	weeks	no_of_users
acer aspire desktop	17	9
acer aspire notebook	17	20
amazon fire phone	17	4
asus chromebook	17	21
dell inspiron desktop	17	18
dell inspiron notebook	17	46
hp pavilion desktop	17	14
htc one	17	16

This is the result of the above Query

TASK 5: Email Engagement Analysis

- Analyze how users are engaging with the email service.

```sql
select action, count(distinct user_id) as engaged_users
from email_events group by action;
```

This is the SQL Query

Result Grid | Filter Rows:

action	engaged_users
email_clickthrough	5277
email_open	5927
sent_reengagement_email	3653
sent_weekly_digest	4111

This is the result of the above Query

INSIGHTS

- Maximum number of jobs reviewed daily is 218, and the minimum is 35.
- Average jobs reviewed per day is 126.
- Average time spent reviewing a job is 0.02.
- Average daily throughout is 0.04.
- 37.50% of the language shared is in Persian.
- Highest weekly user engagement on the 30th.
- Increased by 65% of active users in 2014.
- Highest weekly retention shown in the 33rd week.
- Highest weekly engagement per device of MacBook Pro users.
- Email open action had the highest users engaged.

RECOMENDATION

The analysis shows daily job reviews vary widely, so flexible planning is needed. Review times are quick, but efficiency can still improve. Persian is the dominant language, so better support may boost engagement. User activity peaks on the 30th, and a big user increase in 2014 suggests strategies worth revisiting. Week 33 had the best retention. MacBook Pro users are most active, and emails drive strong engagement, so focusing on these areas can help improve overall performance and user satisfaction.