PROJECT-3

OPERATION & METRIC ANALYTICS

Project Description :-

This project involved using MySQL to analyze operational data and investigate metric spikes. The aim was to generate insights and answer queries related to the organization's operations.

CASE STUDY 01:

A. Number of jobs reviewed: Amount of jobs reviewed over time. Your task: Calculate the number of jobs reviewed per hour per day for November 2020?
B. Throughput: It is the no. of events happening per second. Your task: Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?
C. 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?
D. 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?

CASE STUDY 2:

A. 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?

B. User Growth: Amount of users growing over time for a product. Your task: Calculate the user growth for product?

C. Weekly Retention: Users getting retained weekly after signing-up for a product. Your task: Calculate the weekly retention of users-sign up cohort?

D. 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?

E. Email Engagement: Users engaging with the email service. Your task: Calculate the email engagement metrics?

Approach:
Use SQL to create a database that can store the data and support the queries that we need to perform the analysis and load the data into the database. Use SQL queries to perform the analysis and retrieve the insights you want to extract.

Tech-Stack Used :
MySQL Workbench 8.0 CE

Insights:
The queries were performed and recorded the observations based on the questions that were asked and the followings were results.

Case Study 1 (Job Data)

1.Number of jobs reviewed:

```
use cs1;

###Number of jobs reviewed
select ds, count(job_id) as 'Jobs Reviewed per day', sum(time_spen
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	Jobs Reviewed per day	Hours spent per Day
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

Result 3

2.Throughput:

```sql
###Throughput
SELECT ds,
sum(jobs) over (order by ds rows between 6 preceding and current row) /
sum(total_time) as throughput_7day_rolling_avg
from
(SELECT ds, COUNT(job_id) as jobs, SUM(time_spent) as total_time
FROM job_data where ds >= '2020-11-01' and ds <='2020-11-30'
GROUP BY ds
) d
group by ds;
```

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

ds	throughput_7d_rolling_avg
2020-11-25	0.0222
2020-11-26	0.0357
2020-11-27	0.0288
2020-11-28	0.1515
2020-11-29	0.3000
2020-11-30	0.2000

0.1515

3.Percentage share of each language:

```
###Percentage share of each language
select language, (count(*)/(select count(*) from job_data))*100 as percentage
from job_data
group by language;
```

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

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

4.Duplicate rows:

Limit to 50000 rows

```
###Duplicate rows
select job_id, count(job_id) as duplicate_rows
from job_data
group by job_id
having count(*) > 1;
```

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

job_id	duplicate_rows
23	3

Case Study 2 (Investigating metric spike)

1.User Engagement:

```sql
SELECT
    extract(year from occurred_at) as year,
    extract(week from occurred_at) as weeknum,
    count(distinct user_id) as user_enagagement
FROM
    events
GROUP BY
    year, weeknum
order by
    year, weeknum
```

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

year	weeknum	user_enagagement
2014	17	663
2014	18	1068
2014	19	1113
2014	20	1154
2014	21	1121

2.User Growth:

```sql
select
year_num,
week_num,
num_active_users,
SUM(num_active_users)OVER(ORDER BY year_num, week_num ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_users
from
(
select
extract(year from activated_at) as year_num,
extract(week from activated_at) as week_num,
count(distinct user_id) as num_active_users
from
users
WHERE
state = 'active'
group by year_num,week_num
order by year_num,week_num
) a;
```

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

year_num	week_num	num_active_users	cum_active_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

3.Weekly Retention:

```sql
SELECT
distinct user_id,
COUNT(user_id),
SUM(CASE WHEN retention_week = 1 Then 1 Else 0 END) as per_week_retention
FROM
(
SELECT
a.user_id,
a.signup_week,
b.engagement_week,
b.engagement_week - a.signup_week as retention_week
FROM
(
(SELECT distinct user_id, extract(week from occurred_at) as signup_week from events_data
WHERE event_type = 'signup_flow'
and event_name = 'complete_signup'
and extract(week from occurred_at) = 18
)a
LEFT JOIN
(SELECT distinct user_id, extract(week from occurred_at) as engagement_week FROM events_data
where event_type = 'engagement'
)b
```

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

user_id	COUNT(user_id)	per_week_retention
11926	1	0
11928	1	0
11929	1	0
11931	1	0
11933	1	0

```sql
SELECT
a.user_id,
a.signup_week,
b.engagement_week,
b.engagement_week - a.signup_week as retention_week
FROM
(
(SELECT distinct user_id, extract(week from occurred_at) as signup_week from events_data
WHERE event_type = 'signup_flow'
and event_name = 'complete_signup'
and extract(week from occurred_at) = 18
)a
LEFT JOIN
(SELECT distinct user_id, extract(week from occurred_at) as engagement_week FROM events_data
where event_type = 'engagement'
)b
on a.user_id = b.user_id
)
)d
group by user_id
order by user_id;
```

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

user_id	COUNT(user_id)	per_week_retention
11936	1	0
11939	3	1
11940	1	0
11942	1	0
11944	1	0

4.Weekly Engagement:

```
SELECT
extract(year from occurred_at) as year_num,
extract(week from occurred_at) as week_num,
device,
COUNT(distinct user_id) as no_of_users
FROM
events_data
where event_type = 'engagement'
GROUP by 1,2,3
order by 1,2,3;
```

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

year_num	week_num	device	no_of_users
2014	17	acer aspire desktop	2
2014	17	acer aspire notebook	2
2014	17	amazon fire phone	1
2014	17	asus chromebook	3
2014	17	dell inspiron desktop	1

5.Email Engagement:

```
SELECT
engagements,
total_users,
engagements/total_users*100 AS engagement_rate
from (  SELECT
            count(distinct(user_id)) AS total_users,
COUNT(DISTINCT
CASE
WHEN action = 'email_open' THEN user_id
WHEN action = 'email_clickthrough' THEN user_id END)  AS engagements
        FROM email_events) as counts;
```

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

engagements	total_users	engagement_rate
5927	6179	95.9217

Result :

Extracted many useful analysis that could help in making data driven decisions making for the business.

In this task, I have many basics as well as advanced concepts related to SQL and this project helped me to understand those concepts experimentally.

All the tables and data that were extracted where attached in the following sheet:

Project-3 Results