Operation Analytics and Investigating Metric Spike

Project Description :-

- The project is about operational analytics where this analysis done for the complete end to end operations of the company.
- Where company collects the data of the operations and with the help of this data company analyze the area’s where it is needs to work to improve the performance of the company.
- As a data analyst we need to derive meaningful insights from the data provided by the company operations.
- In this project we basically work on two datasets
- i) Job Data.
- ii) Investigating metric spike.

My Approach :-

- My approach for this project would be first of all reviewing the data provided by the company like how many table’s it has and is there error are present in the data.
- After reviewing the data going through the questions provided by the company and brain storming which functions to use and how to derive the answer’s for that particular questions.
- Optimal ways of writing query for that questions provided by the company.

Tech-Stack Used :-

- To Analyze the data and gain insights from the data I have used MYSQL workbench.

Insights :-

- The completion of this project has helped me to gain very insights about the company operations by analyzing the data we can gain the various useful insights about company which are used to improve the underperforming areas of the company.
- We derived the very useful insights from data like Duplicate Rows , User Engagement, User Growth, Email Engagement and Percentage Share of Each Language .. Etc.

JOB DATA

NUMBER OF JOBS REVIEWED NOVEMBER

Jobs reviewed per hour per day
0.0111

- The jobs reviewed per hour per day is derived by the dividing the count(job_id) and total hour's present in the November month.

MYSQL Queries Used :-

```
SELECT

COUNT(JOB_ID)/(30*28) AS “JOBS REVIEWED PER
HOUR PER DAY”

FROM

PROJECT_1_TABLE
```

7-DAY ROLLING AVERAGE OF THROUGHPUT

DS	JOBS_REVIEWED	7 DAYS ROLLING AVG
25-11-2020	1	1
26-11-2020	1	1
27-11-2020	1	1
28-11-2020	2	1.25
29-11-2020	1	1.2
30-11-2020	2	1.3333

- For Finding the 7-day rolling average of throughput we need to find the jobs_reviewed by counting the job_id in an sub query then by using the windows function over clause we calculate the 7-day rolling average of throughput.

MYSQL Queries Used :-

```
SELECT

DS,

JOBS_REVIEWED,

AVG(JOBS_REVIEWED) OVER(ORDER BY DS ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS "7 DAYS
ROLLING AVG“

FROM PROJECT_1_TABLE

FROM(SELECT DS, COUNT(JOB_ID) AS JOBS_REVIEWED

FROM PROJECT_1_TABLE

GROUP BY DS ORDER BY DS) A
```

PERCENTAGE SHARE OF EACH LANGUAGE

JOB_ID	LANGUAGE	TOTAL OF EACH LANGUAGE	% OF EACH LANGUAGE
21	English	1	12.5%
22	Arabic	1	12.5%
23	Persian	3	37.5%
25	Hindi	1	12.5%
11	French	1	12.5%
20	Italian	1	12.5%

- For Finding the percentage share of each language we calculate the count(language) then dividing it with the occurrence of unique language and then multiplying with the 100 value for percentage share value of that language in all language.

MYSQL Queries Used :-

```
SELECT
JOB_ID,
LANGUAGE,
COUNT(LANGUAGE) AS "TOTAL OF EACH LANGUAGE" ,
((COUNT(LANGUAGE)/(SELECT COUNT(*)
FROM PROJECT_1_TABLE))*100) AS “% OF EACH
 LANGUAGE“
FROM PROJECT_1_TABLE
GROUP BY LANGUAGE, JOB_ID
```

DUPLICATE ROWS

DS	JOB_ID	ACTOR_ID	EVENT	LANGUAGE	TIMESPENT	ORG	ROW_NUM
28-11-2020	23	1005	transfer	Persian	22	D	2
26-11-2020	23	1004	skip	Persian	56	A	3

- For finding the duplicate rows we can use the job_id (or) language as reference for this task.
- In this case I have used the row_number window function partition by job_id to find row_num in a sub-query.
- If the row_num has more than 1 then those row are known be duplicate rows.

MYSQL Queries Used :-

```
SELECT  *
FROM
(SELECT  *, ROW_NUMBER()OVER(PARTITION BY JOB_ID)
 AS
ROW_NUM
FROM PROJECT.SHEET)A
WHERE ROW_NUM >1
```

Investigating Metric Spike

WEEK	USERS
17	85
18	194
19	208
20	195
21	208
22	230
23	224
24	252
25	245
26	230

USER ENGAGEMENT

- For finding the user engagement we need to extract the particular week of that from event occurred_at.
- Need to count the unique users from user_id provided in the data then we need to group them by using the week in the given data.

MYSQL Queries Used :-

```
select
extract(week from occurred_at) as week,
count( distinct user_id) as users
from table_2_events
group by Week
```

USER GROWTH

YEARS	WEEKS	ACTIVE_USERS	CUM_ACTIVE_USERS
2013	1	23	23
2013	2	30	53
2013	3	48	101
2013	4	36	137
2013	5	30	167
2013	6	48	215
2013	7	38	253
2013	8	42	295
2013	9	34	329
2013	10	43	372
2013	11	32	404
2013	12	31	435
2013	13	33	468

MYSQL Queries Used :-

```
Select

 years, weeks, active_users, SUM(active_users)OVER(ORDER BY years, weeks ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_users

from(select extract(year from activated_at) as years, extract(week from activated_at)+1 as weeks, --
Add 1 so that weeks counting start with 1 otherwise it start with 0

count(distinct user_id) as active_users

from table_1_users

WHERE state = 'active’

group by years, weeks

order by years, weeks) a

-- For total active users

select count(*) from table_1_users

where state = 'active'
```

WEEKLY RETENTION

USER_ID	COUNT(USER_ID)	PER_WEEK_RETENTION
11768	1	0
11770	1	0
11775	2	1
11778	3	0
11779	1	0
11780	1	0
11785	1	0
11787	3	1
11791	1	0

MYSQL Queries Used :-

```
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 table_2_events

WHERE event_type = 'signup_flow' And  event_name = 'complete_signup') a LEFT JOIN

(SELECT distinct user_id, extract(week from occurred_at) as engagement_week

FROM table_2_events

where event_type = 'engagement') b on a.user_id = b.user_id)) d

group by user_id

order by user_id
```

WEEKLY ENGAGEMENT

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
2014	17	DELL INSPIRON NOTEBOOK	4
2014	17	HP PAVILION DESKTOP	2
2014	17	HTC ONE	2
2014	17	IPAD AIR	1
2014	17	IPAD MINI	3

MYSQL Queries Used :-

```
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 table_2_events
where event_type = 'engagement’
GROUP by 1,2,3
order by 1,2,3
```

EMAIL ENGAGEMENT

Email Opening Rate	Email Clicking Rate
33.58339	14.78989

MYSQL Queries Used :-

```
SELECT
100.0*SUM(CASE when email = 'email_opened' then 1 else 0 end)/SUM(CASE when email = 'email_sent' then 1 else 0 end) as email_opening_rate,

100.0*SUM(CASE when email = 'email_clicked' then 1 else 0 end)/SUM(CASE when email = 'email_sent' then 1 else 0 end) as email_clicking_rate

FROM

(SELECT *,

CASE

WHEN action in ('sent_weekly_digest','sent_reengagement_email')

then 'email_sent'

WHEN action in ('email_open')

then 'email_opened'

WHEN action in ('email_clickthrough')

then 'email_clicked'end as email

from table_3_email_events) a
```

RESULTS :-

- This project has helped me to understand various analytics metrics and how to use database operations in MYSQL.
- Where I applied different functions like window functions and very different ways of solving a problem using sub-queries and MYSQL functions of various types.
- Working on different types of databases my analytical thinking and way of solving a problem is improved using MYSQL.
- Gained more experience on various function in MYSQL and different ways of using in it.
- Extraction of useful insights from different databases using MYSQL has really improved my database skills

THANKING YOU