Project Description:

Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect.

Being one of the most important parts of a company, we are doing this analysis to predict the overall growth or decline of a company's fortune.

So we have to carry out metric analysis as Investigating metric spike is also an important part of operation analytics as being a Data Analyst. Here, we are supposed to be able to understand or make other teams understand questions.

We are working as a Data Analyst Lead and is provided with different data sets, tables from which we have to derive certain insights and answer the questions asked by different departments..

Here, we will be using SQL to run different commands to find the answers to the questions asked. For the same purpose, we will be using SQL help pages and the learning material provided.

Approach:

We have used SQL help pages and learning materials to make the codes. In case of any doubts, I have tried multiple trial and error and for any error I received while executing a command, I have checked the error message to understand the error and how to rectify it.

Since the data we have is in the form of spread sheet in .csv format, we have to use "Table data import Wizard" to import this data in MySQL software. Here, we have created 4 tables namely:

1. Case_study_1 - This includes basic Job data which was part of case_study_1 dataset provided.
2. Email_events – This dataset includes basic email activities done by the user. This was part of case_study_2 dataset.
3. Events - This includes events occurred along with it's type, date and device. This was part of case_study_2 dataset.
4. Users – This includes information about user that is when they started registration and when account was created. This was part of case_study_2 dataset.

Tech-Stack Used:

MySQL workbench 8.0.29

Insights:

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?

We have to identify the number of jobs per hour and per day. The column 'time_spent' has information in seconds. We have to divide the time by 3600 which keeping it in denominator. At the same time, we have kept our numerator as count of jobs.

```
SELECT
   ds,
   COUNT(job_id) * 3600 / SUM(time_spent) AS jobs_per_hour
FROM
   case_study_1
WHERE
   ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY ds
```

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?

Now we have to identify throughput which is again similar to events happening per second so we already have information on seconds required to review job in 'time_spent' column. Here, we just need to find count of jobs and divide it with sum of time spent to find out average throughput.

In order to calculate 7 days rolling throughput, we need to use 'over' clause with 6 days preceding and current day.

```
SELECT ds, throughput_per_day, sum(numjobs) over (order by ds rows between 6 preceding and current row)/sum(timespent) over (order by ds rows between 6 preceding and current row) as throughput_rolling_7d

from

(SELECT

   ds, COUNT(job_id) AS numjobs, SUM(time_spent) AS timespent, COUNT(job_id)/SUM(time_spent) as throughput_per_day

FROM

   case_study_1

WHERE

   ds BETWEEN '2020-11-01' AND '2020-11-30'

GROUP BY ds

ORDER BY ds) as a
```

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?

Here, we have to find share of each language from total number of language. So we needed to find total jobs and count of each job as well. So for that purpose, we have created 2 sub queries with aliases as ‘a’ and ‘b’. Alias ‘a’ to find out count of each language and alias ‘b’ to find out sum of total languages. Then using cross join, we provided sum of languages for each language. Then dividing number of jobs by total number of jobs gives the required output.

```
WITH a as (

SELECT

   language, COUNT(job_id) AS numjobs

FROM

   case_study_1

WHERE

   ds BETWEEN '2020-11-01' AND '2020-11-30'
```

```
GROUP BY language
),
b as (
SELECT
   language, SUM(num_jobs) AS total_jobs
FROM
   (SELECT
      language, COUNT(job_id) num_jobs
   FROM
      case_study_1
   WHERE
      ds BETWEEN '2020-11-01' AND '2020-11-30'
   GROUP BY language) b
)
SELECT
   a.language,
   (numjobs * 100 / total_jobs) AS share_of_each_language,
   total_jobs
FROM
   a
      CROSS JOIN
   b
ORDER BY share_of_each_language
```

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?

In order to identify the duplicate rows, we have taken the duplication rule as per job data that is if job data is matching, then it is a duplicate row. Here, we need to find the rank and partition the data by job_id. What it will do is it will assign rank 1 to every unique entries of Job_id except the one with repeated job_id for those contacts, job_id as 2 and 3 will be assigned. Then we can simply use where clause for ranking more than 1 to find the duplicate rows.

```
WITH duplicate_num as
(
SELECT *,
RANK() over(partition by job_id order by event) ranknum
FROM case_study_1
)
SELECT
  *
FROM
  duplicate_num
WHERE
  ranknum > 1
```

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?

We are required to identify the user's weekly engagement. So we need to fetch the weeks out of the column 'occurred_at' and then count the distinct user_ID with event equals to engagement. Then grouping it by week_num calculated will help us getting required solution.

```
SELECT
  WEEK(occurred_at) AS week_num,
  COUNT(DISTINCT user_id) AS weekly_active_user
```

```
FROM
   events
WHERE
   event_type = 'engagement'
GROUP BY week_num
ORDER BY week_num
```

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

We have to find out the number of account being created each day and total number of account present after each day. So we have to identify the count of all users grouped as per day. We used day function to find out the day of month. Keeping this in one sub query, we created original query for cumulative number of active user with the help of over clause.

```
select date_of_month, all_users, sum(all_users) over (order by date_of_month) as
Cum_all_users from
(
SELECT
   DAY(created_at) AS date_of_month, COUNT(*) AS all_users
FROM
   users
WHERE
   state = 'active'
GROUP BY date_of_month
ORDER BY date_of_month
) a
```

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

We needed to identify users who are using the site even after initial signup so thus finding out the weekly retention. In order to find it, we need to first find the difference between occurrence date and activation date as occurrence date will give us the activity day and activation date will give us when account was activated.

Once we identify the age of their account in days, then we can classify the data based on the fact that whether how many users were retained for more than week and how many users for just a week using ‘case’ clause.

```
SELECT
  COUNT(DISTINCT CASE
      WHEN z.age_of_user > 35 THEN z.user_id
      ELSE NULL
    END) AS '5+ weeks',
  COUNT(DISTINCT CASE
      WHEN
        z.age_of_user < 35
          AND z.age_of_user <= 28
      THEN
        z.user_id
      ELSE NULL
    END) AS '5 weeks',
  COUNT(DISTINCT CASE
      WHEN
        z.age_of_user < 28
          AND z.age_of_user <= 21
      THEN
```

```
            z.user_id
        ELSE NULL
    END) AS '4 weeks',
  COUNT(DISTINCT CASE
        WHEN
            z.age_of_user < 21
                AND z.age_of_user <= 14
        THEN
            z.user_id
        ELSE NULL
    END) AS '3 weeks',
  COUNT(DISTINCT CASE
        WHEN
            z.age_of_user < 14
                AND z.age_of_user <= 7
        THEN
            z.user_id
        ELSE NULL
    END) AS '2 weeks',
  COUNT(DISTINCT CASE
        WHEN z.age_of_user < 7 AND z.age_of_user <= 0 THEN z.user_id
        ELSE NULL
    END) AS 'Less than a week'
FROM
  (SELECT
    u.user_id,
```

```
        DATEDIFF(e.occurred_at, u.activated_at) AS age_of_user
    FROM
      users u
    LEFT JOIN events e ON u.user_id = e.user_id
    WHERE
      e.event_type = 'engagement'
    GROUP BY user_id
    ORDER BY user_id) z
```

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?

Since we were required to find out weekly engagement per device, we identify count of users where event equals engagement. For this, we then grouped the date by device and then date so that we find out for each device, number of engagement activities carried out on each date.

```
SELECT
        device,
    WEEK(occurred_at) AS week_num,
    COUNT(DISTINCT user_id) AS weekly_active_user
FROM
    events
WHERE
    event_type = 'engagement'
GROUP BY device, week_num
ORDER BY week_num
```

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

As we were required to identify the email engagement metrics, we have to divide the date as per weeks in 4 different even outputs of emails. So we took the count of action is grouped it on action and then week.

```
SELECT
   WEEK(occurred_at) AS week_num, action, COUNT(action) as No_of_instances
FROM
   email_events
GROUP BY week_num , action
order by week_num
```

Result:

1. **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?**

ds	jobs_per_hour
2020-11-30	180.0000
2020-11-29	180.0000
2020-11-28	218.1818
2020-11-27	34.6154
2020-11-26	64.2857
2020-11-25	80.0000

2. **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?**

Result Grid | Filter Rows: | Export:

ds	throughput_per_day	throughput_rolling_7d
2020-11-25	0.0222	0.0222
2020-11-26	0.0179	0.0198
2020-11-27	0.0096	0.0146
2020-11-28	0.0606	0.0210
2020-11-29	0.0500	0.0233
2020-11-30	0.0500	0.0268

3. **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?**

language	share_of_each_language	total_jobs
English	12.5000	8
Arabic	12.5000	8
Hindi	12.5000	8
French	12.5000	8
Italian	12.5000	8
Persian	37.5000	8

4. **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?**

ds	job_id	actor_id	event	language	time_spent	org	ranknum
2020-11-26	23	1004	skip	Persian	56	A	2
2020-11-28	23	1005	transfer	Persian	22	D	3

5. **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?**

week_num	weekly_active_user
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

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

date_of_month	all_users	Cum_all_users
1	280	280
2	296	576
3	283	859
4	322	1181
5	268	1449
6	277	1726
7	331	2057
8	283	2340
9	274	2614
10	292	2906
11	327	3233
12	291	3524
13	313	3837
14	330	4167
15	312	4479
16	319	4798
17	301	5099
18	342	5441
19	281	5722
20	319	6041
21	326	6367
22	310	6677
23	299	6976
24	298	7274

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

5+ weeks	5 weeks	4 weeks	3 weeks	2 weeks	Less than a week
2139	3947	3881	3812	3743	3680

8. **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?**

device	week_num	weekly_active_user
mac mini	17	6
macbook air	17	54
macbook pro	17	143
nexus 10	17	16
nexus 5	17	40
nexus 7	17	18
nokia lumia 635	17	17
samsumg galaxy tablet	17	8
samsung galaxy note	17	7
samsung galaxy s4	17	52
windows surface	17	10
acer aspire desktop	18	26
acer aspire notebook	18	33
amazon fire phone	18	9
asus chromebook	18	42
dell inspiron desktop	18	58
dell inspiron notebook	18	77
hp pavilion desktop	18	37
htc one	18	19
ipad air	18	52
ipad mini	18	30
iphone 4s	18	46

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

week_num	action	No_of_instances
17	email_clickthrough	166
17	email_open	310
17	sent_reengagement_email	73
17	sent_weekly_digest	908
18	email_clickthrough	430
18	email_open	912
18	sent_reengagement_email	157
18	sent_weekly_digest	2602
19	email_clickthrough	477
19	email_open	972
19	sent_reengagement_email	173
19	sent_weekly_digest	2665
20	email_clickthrough	507
20	email_open	1004
20	sent_reengagement_email	191
20	sent_weekly_digest	2733
21	email_clickthrough	443
21	email_open	1014
21	sent_reengagement_email	164
21	sent_weekly_digest	2822
22	email_clickthrough	488
22	email_open	987