ADVANCED SQL

PROJECT 3

Operation Analytics and Investigating Metric Spike

RAKSHA NAYAK

Project Description: This Project involves operational analytics and investigating metric spike to provide valuable insights that can help the business grow. The insights derived from this analysis can be used by various teams like operations, support and marketing to improve the business.

Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. One of the key aspects of operational analytics is investigating metric spikes. Thus, as a data analyst, I would be answering and investigating metric spikes.

Approach: In this project, I would be using SQL and MySQL Workbench as a tool to analyze data from the database and answer questions posed by the different departments within the company and also provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.

I created a database called jd_analysis and ims_project3 for case study 1 and case study 2. Then imported the csv files provided in MySQL Workbench. Then spent time to understand the data, their datatypes and relationships with other data. This will help me to understand the question and approach in a right way to fetch the answer.

Tech stack used: SQL (Structured Query Language) is used along with MySQL Workbench 8.0.38 to perform analysis.

MySQL Workbench is a visual tool (GUI) that helps to work with MYSQL databases and servers. Therefore, it is easy to use and intuitive.

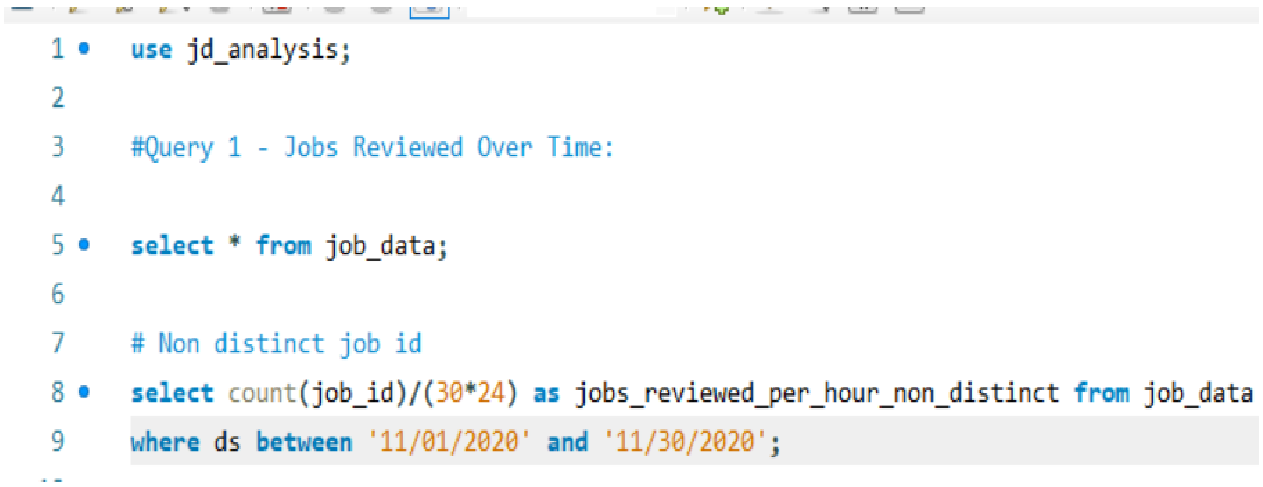
Case Study 1: Job Data Analysis

A. Jobs Reviewed Over Time:

- Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.
- My Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

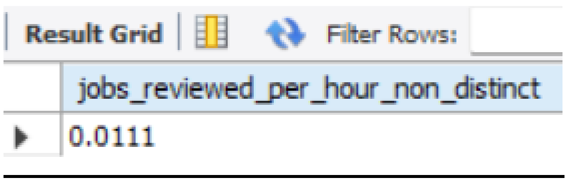
Approach: To retrieve the jobs reviewed per hour per day in the month of November 2020.

- We will use the data from job_id columns of the job_data table.
- Then we will divide the total count of job_id(distinct and non-distinct) by (30 days * 24 hours) as we need to find number of jobs reviewed per hour per day.



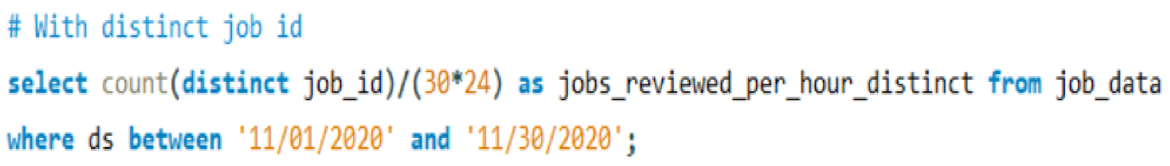
```sql
use jd_analysis;

#Query 1 - Jobs Reviewed Over Time:

select * from job_data;

# Non distinct job id
select count(job_id)/(30*24) as jobs_reviewed_per_hour_non_distinct from job_data
where ds between '11/01/2020' and '11/30/2020';
```

Output:



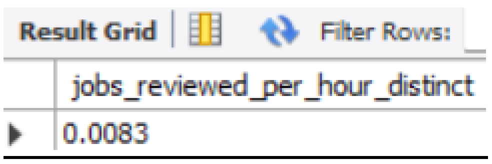
Result Grid | Filter Rows:

```
jobs_reviewed_per_hour_non_distinct
0.0111
```



```sql
# With distinct job id
select count(distinct job_id)/(30*24) as jobs_reviewed_per_hour_distinct from job_data
where ds between '11/01/2020' and '11/30/2020';
```

Output:



Result Grid | Filter Rows:

```
jobs_reviewed_per_hour_distinct
0.0083
```

Insight:

1. Number of non-distinct jobs reviewed per hour for each day in November 2020 is 0.0111.
2. Number of distinct jobs reviewed per hour for each day in November 2020 is 0.0083.

B. Throughput Analysis:

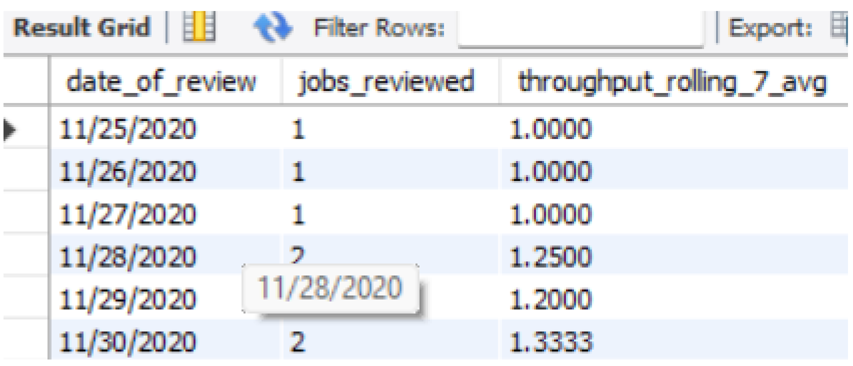
- Objective: Calculate the 7-day rolling average of throughput (number of events per second).
- My Task: Write an SQL query to calculate the 7-day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

Approach:

- We will use select statement along with aggregate functions like avg() and count().
- Rows() helps to consider the rows between previous 6 rows and current row.
- We are also going to group by and order by ds

```sql
#Query 2 - Throughput Analysis

select * from job_data;

select
ds as date_of_review,
count(job_id) as jobs_reviewed,
avg(count(job_id)) over(order by ds rows between 6 preceding and current row) as throughput_rolling_7_avg
from job_data
group by ds
order by ds;
```

Output:



date_of_review	jobs_reviewed	throughput_rolling_7_avg
11/25/2020	1	1.0000
11/26/2020	1	1.0000
11/27/2020	1	1.0000
11/28/2020	2	1.2500
11/29/2020	[illegible]	1.2000
11/30/2020	2	1.3333

Insight: Produced the 7-day rolling average for throughput, offering a consistent metric for thorough performance analysis.

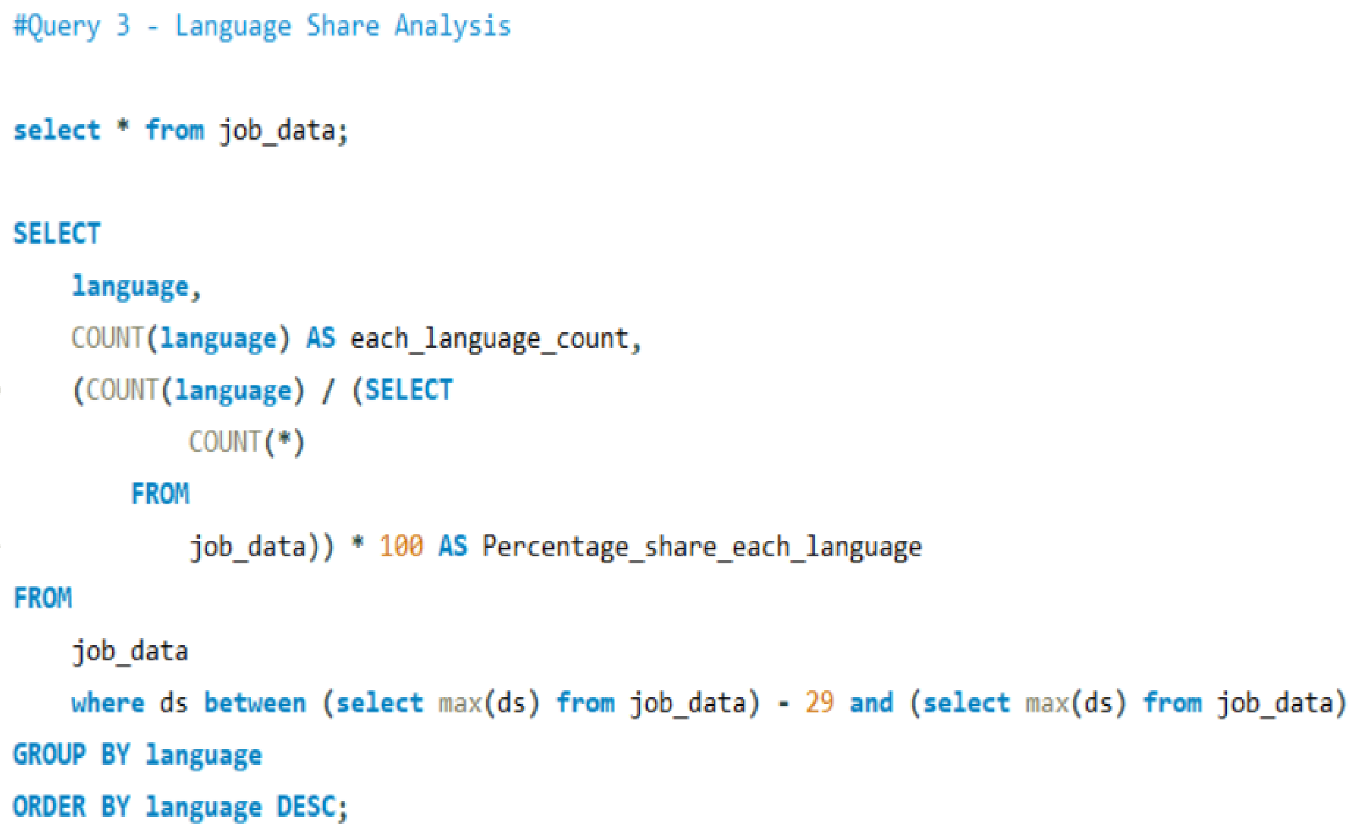
I prefer using the 7-day rolling average for throughput as it yields a more stable trend, more precise and mitigating daily variances. 7-day rolling average is more reliable than daily metric.

C. Language Share Analysis:

- Objective: Calculate the percentage share of each language in the last 30 days.
- My Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

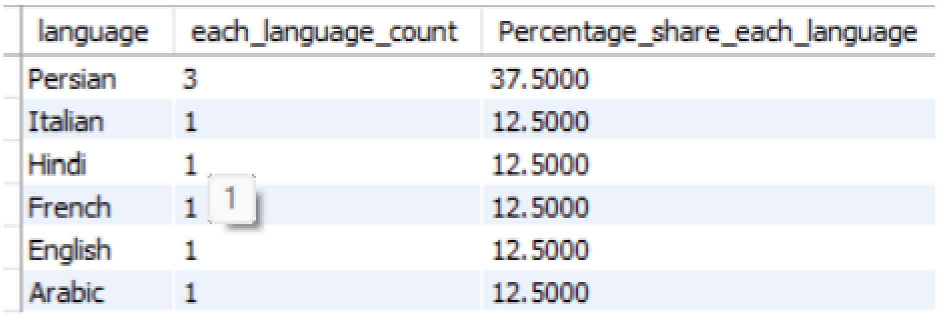
Approach:

- Let's use select statement to fetch language, each language count and their percentage share.
- To calculate Percentage, each language count is divided by total records and multiplied by 100.
- Group by is done on language column to get percentage share of each column
- Finally, have sorted language percentage in descending order.



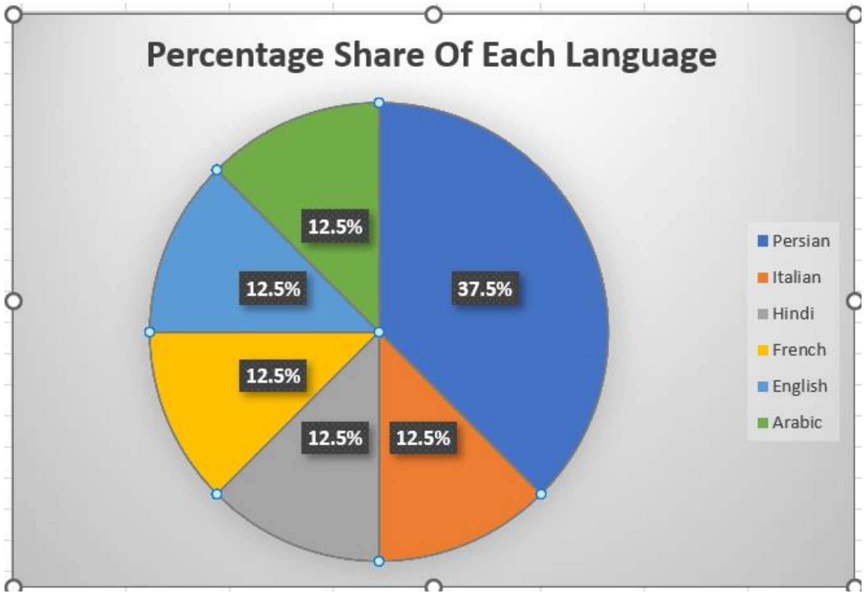
```
#Query 3 - Language Share Analysis

select * from job_data;

SELECT
    language,
    COUNT(language) AS each_language_count,
    (COUNT(language) / (SELECT
            COUNT(*)
        FROM
            job_data)) * 100 AS Percentage_share_each_language
FROM
    job_data
    where ds between (select max(ds) from job_data) - 29 and (select max(ds) from job_data)
GROUP BY language
ORDER BY language DESC;
```

Output:



language	each_language_count	Percentage_share_each_language
Persian	3	37.5000
Italian	1	12.5000
Hindi	1	12.5000
French	1	12.5000
English	1	12.5000
Arabic	1	12.5000

Visual Representation using Pie Chart in Excel:



Insight: According to derived table and pie chart, it is clear that Persian language is having highest percentage share of 37.5 and the rest of the languages, i.e., Italian, Hindi, French, English and Arabic have same percentage share of 12.5%.

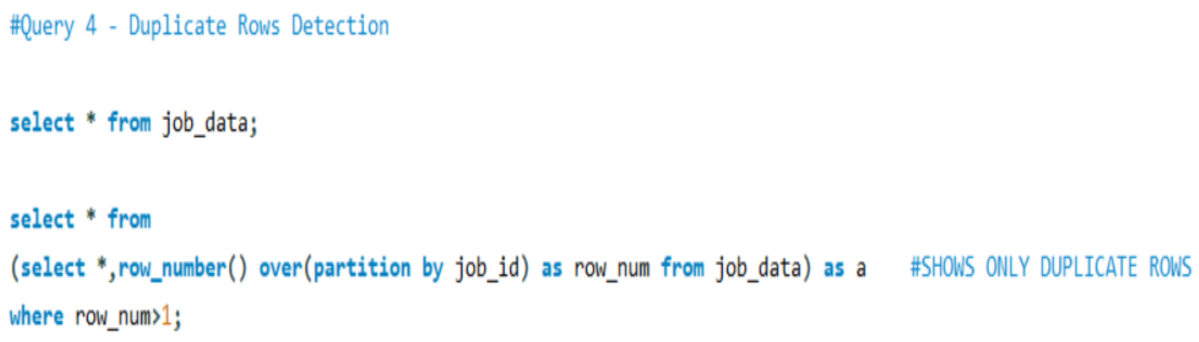
D. Duplicate Rows Detection:

- Objective: Identify duplicate rows in the data.
- My Task: Write an SQL query to display duplicate rows from the job_data table.

Approach:

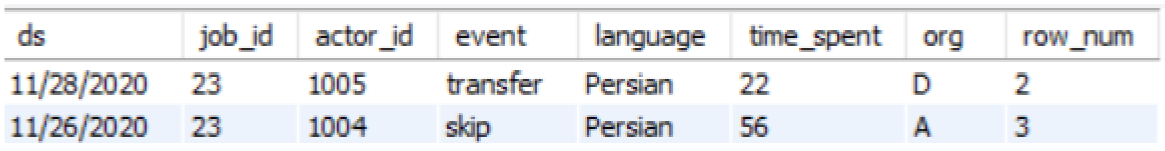
- Let's first decide in which column are we looking for duplicates. Here, let's consider job_id(As this table deals with job_data, no 2 records can have the same job_id. Thus, instead of looking for duplicates in entire row, I am looking for duplicates in job_id column)
- We will then use row_number() function to display the number of duplicate values.

- We will then group by job_id using over () function and use WHERE clause to look for records who have row_number greater than 1.
- This displays only duplicate copy/rows and not the original copy.



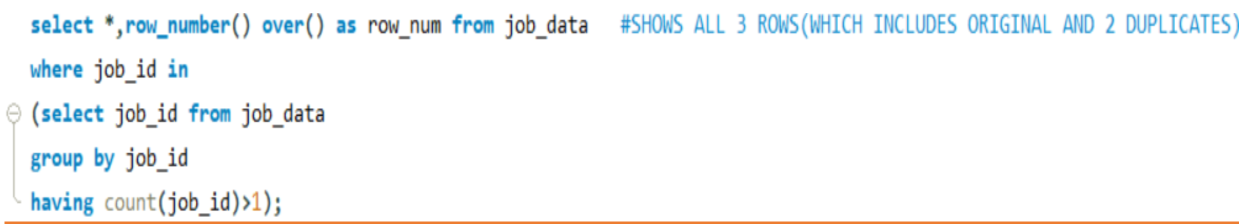
```
#Query 4 - Duplicate Rows Detection

select * from job_data;

select * from
(select *,row_number() over(partition by job_id) as row_num from job_data) as a     #SHOWS ONLY DUPLICATE ROWS
where row_num>1;
```

Output:



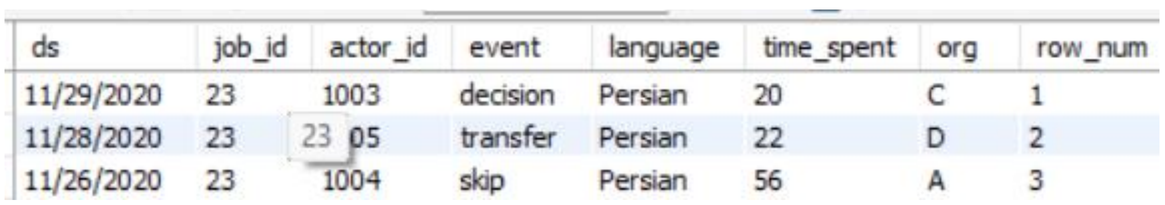
ds	job_id	actor_id	event	language	time_spent	org	row_num
11/28/2020	23	1005	transfer	Persian	22	D	2
11/26/2020	23	1004	skip	Persian	56	A	3

Insight: There are 2 records, which are a copy of an original record with job_id as 23. To get original record/copy, below query can be executed, which shows copies along with original. Revealed and showcased duplicate attributes of an rows highlighting potential threat of data integrity. Further, data cleaning can be facilitated for upholding data integrity.



```
select *,row_number() over() as row_num from job_data   #SHOWS ALL 3 ROWS(WHICH INCLUDES ORIGINAL AND 2 DUPLICATES)
where job_id in
(select job_id from job_data
group by job_id
having count(job_id)>1);
```

Output:



ds	job_id	actor_id	event	language	time_spent	org	row_num
11/29/2020	23	1003	decision	Persian	20	C	1
11/28/2020	23	23 05	transfer	Persian	22	D	2
11/26/2020	23	1004	skip	Persian	56	A	3

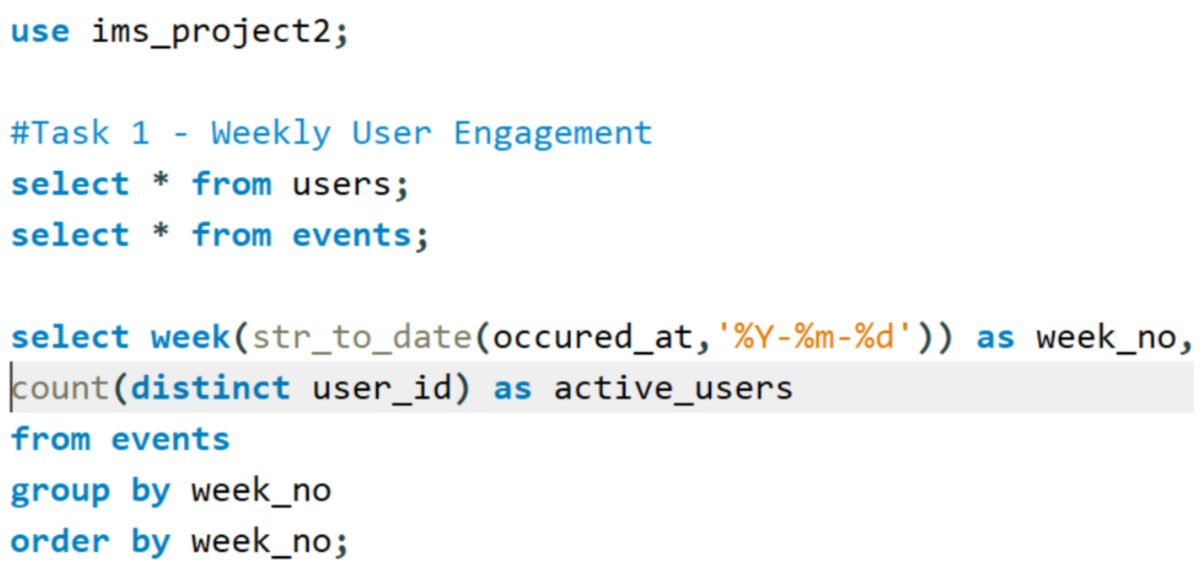
Case Study 2: Investigating Metric Spike

A. Weekly User Engagement:

- Objective: Measure the activeness of users on a weekly basis.
- My Task: Write an SQL query to calculate the weekly user engagement.

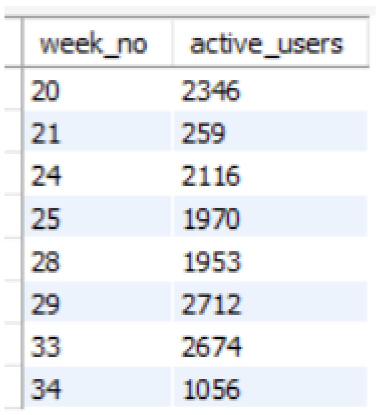
Approach: To find the weekly user engagement,

- We will extract week from the occurred_at coulumn by changing its datatype to datetime.
- Then we will take the count of unique active users using user_id attribute.
- To extract active users of the respective weeks, we need to group by and sort by week number.



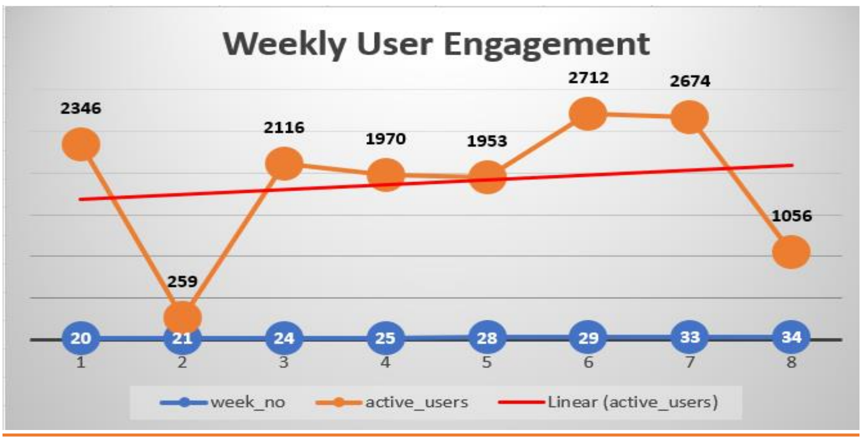
```sql
use ims_project2;

#Task 1 - Weekly User Engagement
select * from users;
select * from events;

select week(str_to_date(occured_at,'%Y-%m-%d')) as week_no,
count(distinct user_id) as active_users
from events
group by week_no
order by week_no;
```

Output:



week_no	active_users
20	2346
21	259
24	2116
25	1970
28	1953
29	2712
33	2674
34	1056

Visual Representation using Line Chart in Excel:



Insight: This query computes the count of unique active users participating in the event on a weekly basis. From the above graph, we can track the broader spectrum of user engagement over an extended period.

Observation: Week 29 had highest users active and week 21 has the least user engagement.

B. User Growth Analysis:

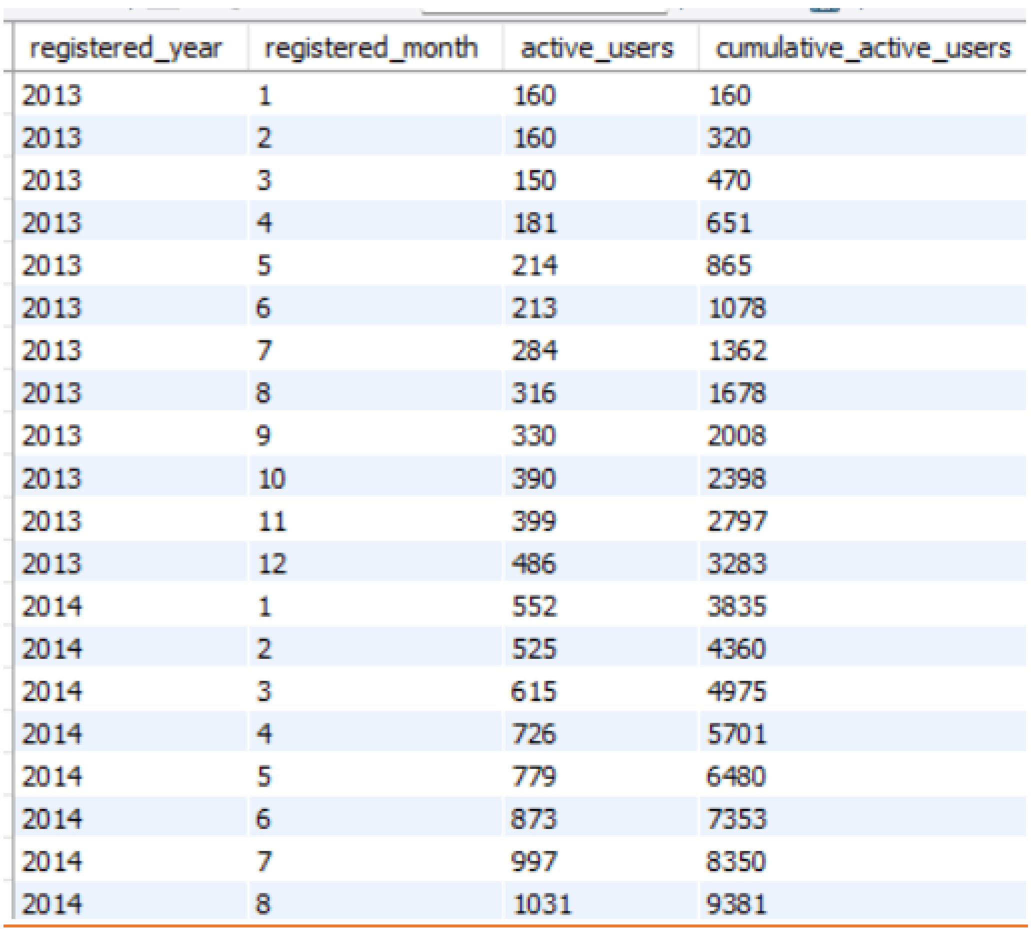
- Objective: Analyze the growth of users over time for a product.
- My Task: Write an SQL query to calculate the user growth for the product.

Approach: To find the user growth (active users per month),

- First, we are grouping them by year and months as our dataset is quite huge.
- To extract year and month, we are going to convert activated_at attributes datatype from varchar to date.
- We are also displaying cumulative active users by taking sum of previous row and current row's active users using over and row function.
- Lastly, we are grouping and sorting by year and then month.

```
#Task 2 - User Growth Analysis

select * from users;

select extract(year from (str_to_date(activated_at,'%d-%m-%Y %H:%i'))) as registered_year,
extract(month from (str_to_date(activated_at,'%d-%m-%Y %H:%i'))) as registered_month,
count(distinct user_id) as active_users,
sum(count(distinct user_id)) over(order by
(select extract(year from (str_to_date(activated_at,'%d-%m-%Y %H:%i'))) as registered_year),
(select extract(month from (str_to_date(activated_at,'%d-%m-%Y %H:%i'))) as registered_month)
rows between unbounded preceding and current row) as cumulative_active_users
from users
group by registered_year,registered_month
order by registered_year,registered_month;
```

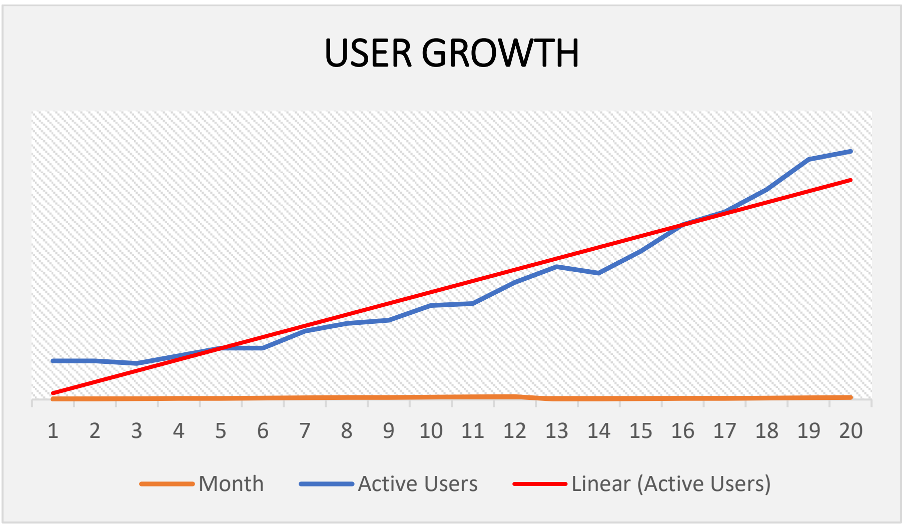
Output:



registered_year	registered_month	active_users	cumulative_active_users
2013	1	160	160
2013	2	160	320
2013	3	150	470
2013	4	181	651
2013	5	214	865
2013	6	213	1078
2013	7	284	1362
2013	8	316	1678
2013	9	330	2008
2013	10	390	2398
2013	11	399	2797
2013	12	486	3283
2014	1	552	3835
2014	2	525	4360
2014	3	615	4975
2014	4	726	5701
2014	5	779	6480
2014	6	873	7353
2014	7	997	8350
2014	8	1031	9381

Insight: There are 9381 active users in total from 1st month of the year 2013 to 8th month of the year 2014. Therefore, as per the analysis there is a drastic increase in active users count on monthly basis.

Visual Representation using Line Chart in Excel:

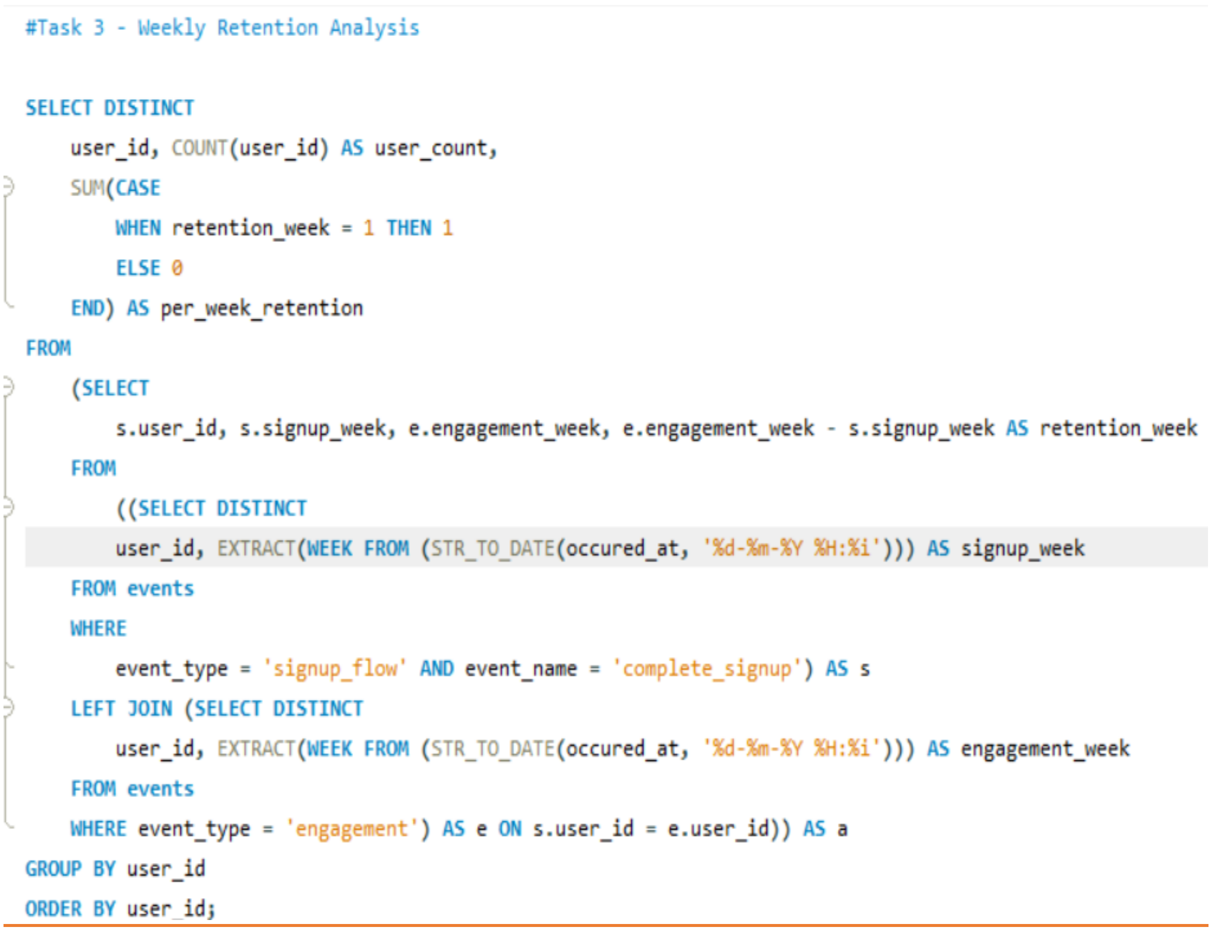


C. Weekly Retention Analysis:

- Objective: Analyze the retention of users on a weekly basis after signing up for a product.
- My Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Approach: For weekly retention of users after signing up for a product,

- First, we will extract week from occurred_at column of events table
- Then we will select the rows in which event_type='signup_flow' and event_name='complete_signup'.
- Then we will perform left join on user_id, whose event_type='engagement'
- We are also subtraction engagement week with signup week and if value is 1, we are showing 1 under per_week_retention column else 0.
- Finally grouping and sorting by user_id



```sql
#Task 3 - Weekly Retention Analysis

SELECT DISTINCT
    user_id, COUNT(user_id) AS user_count,
    SUM(CASE
        WHEN retention_week = 1 THEN 1
        ELSE 0
    END) AS per_week_retention
FROM
    (SELECT
        s.user_id, s.signup_week, e.engagement_week, e.engagement_week - s.signup_week AS retention_week
    FROM
        ((SELECT DISTINCT
        user_id, EXTRACT(WEEK FROM (STR_TO_DATE(occured_at, '%d-%m-%Y %H:%i'))) AS signup_week
    FROM events
    WHERE
        event_type = 'signup_flow' AND event_name = 'complete_signup') AS s
    LEFT JOIN (SELECT DISTINCT
        user_id, EXTRACT(WEEK FROM (STR_TO_DATE(occured_at, '%d-%m-%Y %H:%i'))) AS engagement_week
    FROM events
    WHERE event_type = 'engagement') AS e ON s.user_id = e.user_id)) AS a
GROUP BY user_id
ORDER BY user_id;
```

Output: Kindly check the output at below provided link

https://drive.google.com/file/d/12ot8aPZvxwB-qf_vvqJidleEs4aWHTx3/view?usp=sharing

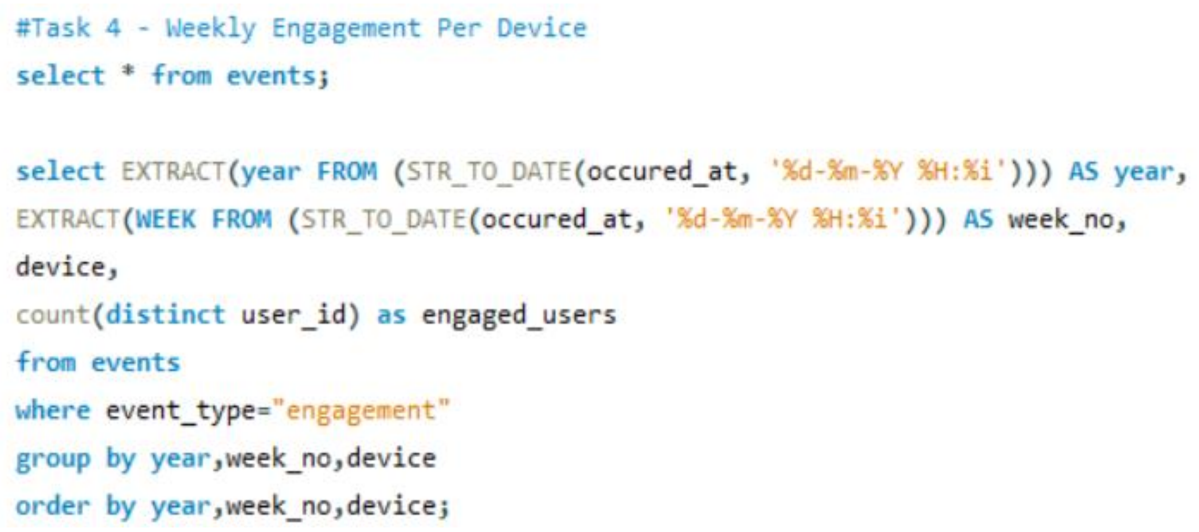
Insight: Out of 1001 records, 723 have a weekly retention of 1. Thus, this helps us to compute weekly retention of users by assessing the relationship between the week of sign up and engagement period. This analysis helps widely to aim for the desired result of user retention after signup.

D. Weekly Engagement Per Device:

- Objective: Measure the activeness of users on a weekly basis per device.
- My Task: Write an SQL query to calculate the weekly engagement per device.

Approach: For measuring weekly engagement per device,

- We will extract year and week from occured_at column of events
- We will check for a condition event_type='engagement' and retain the year, week, device and engaged_users by grouping and sorting by year, week and device



```
#Task 4 - Weekly Engagement Per Device
select * from events;

select EXTRACT(year FROM (STR_TO_DATE(occured_at, '%d-%m-%Y %H:%i'))) AS year,
EXTRACT(WEEK FROM (STR_TO_DATE(occured_at, '%d-%m-%Y %H:%i'))) AS week_no,
device,
count(distinct user_id) as engaged_users
from events
where event_type="engagement"
group by year,week_no,device
order by year,week_no,device;
```

Output:

https://drive.google.com/file/d/1K_mLtbJn1mThILg2XkcpRWUQ6VRq8tbJ/view?usp=sharing

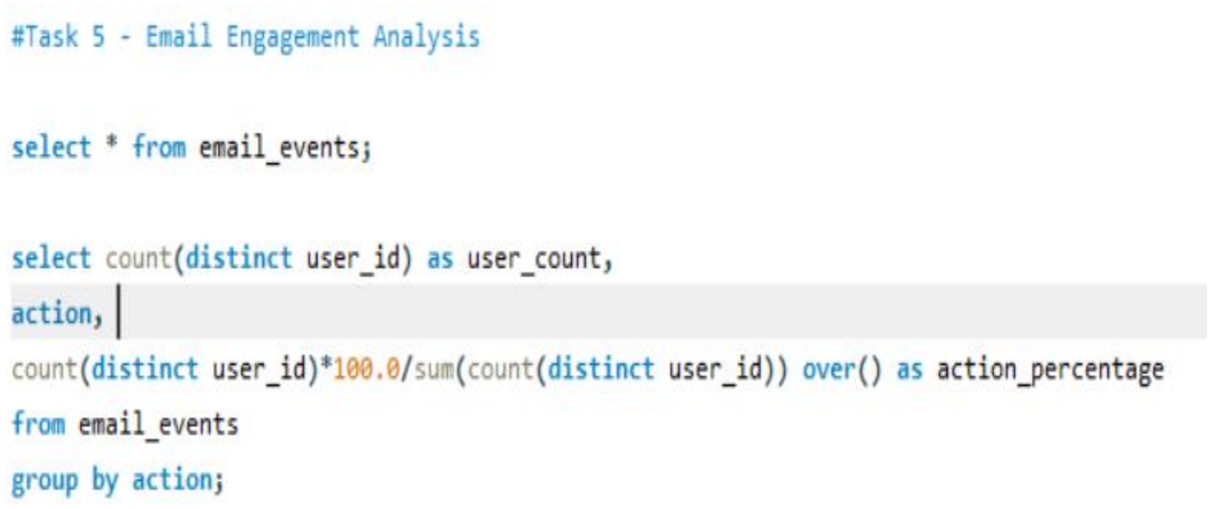
Insight: Macbook Pro was used widely during 30th week of 2014 by 322 users. Samsung galaxy note was used just once during 35th week of 2014. Thus, this helps in identifying devices that significantly contribute to user activity.

E. Email Engagement Analysis:

- Objective: Analyze how users are engaging with the email service.
- My Task: Write an SQL query to calculate the email engagement metrics.

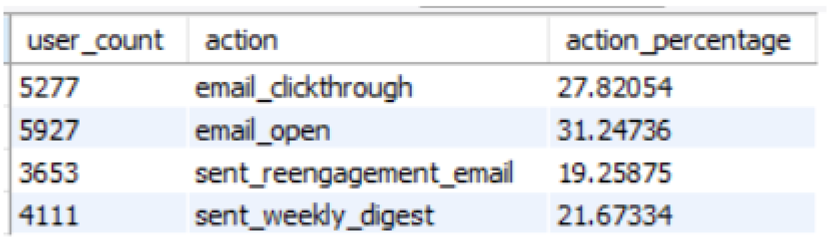
Approach: Analyzing email engagement can be done in a simple or more logical ways.

- Simple way is to fetch user_id counts based on the email actions they are performing (i.e., grouping by action). To make it more interactive, percentage of each action can be added by taking user count for each action*100/total(sum) of user count.
- Logical way is to categorize the action into "email_opened"," email_sent"," email_clicked". This can be achieved by using when, case and then functions.



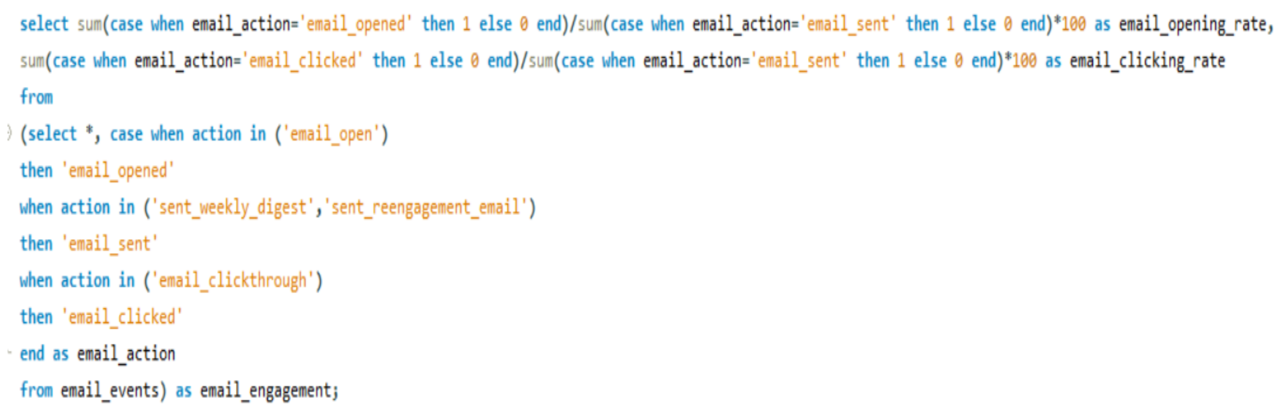
```
#Task 5 - Email Engagement Analysis

select * from email_events;

select count(distinct user_id) as user_count,
action,
count(distinct user_id)*100.0/sum(count(distinct user_id)) over() as action_percentage
from email_events
group by action;
```

Output:

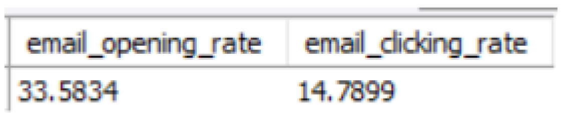


user_count	action	action_percentage
5277	email_clickthrough	27.82054
5927	email_open	31.24736
3653	sent_reengagement_email	19.25875
4111	sent_weekly_digest	21.67334

Other method:


```
select sum(case when email_action='email_opened' then 1 else 0 end)/sum(case when email_action='email_sent' then 1 else 0 end)*100 as email_opening_rate,
sum(case when email_action='email_clicked' then 1 else 0 end)/sum(case when email_action='email_sent' then 1 else 0 end)*100 as email_clicking_rate
from
(select *, case when action in ('email_open')
then 'email_opened'
when action in ('sent_weekly_digest','sent_reengagement_email')
then 'email_sent'
when action in ('email_clickthrough')
then 'email_clicked'
end as email_action
from email_events) as email_engagement;
```

Output:



email_opening_rate	email_clicking_rate
33.5834	14.7899

Insight: The email engagement actions has happened with email opening at 31%, email click through at 28%, sending weekly digest at 22% and sending reengagement mails at 19%.

Thus, this has helped us to access user engagement concerning various email actions and helped to understand popular email interactions among users.

Insights: Providing Insights is one of the most important parts of the project. Insights help the various departments of the organization to make informed decisions and improve the business.

I have already provided insights for each query after their execution for better understanding. Please refer under specific queries.

Result: This project helped me to understand

- How to create a database (tables, attributes, datatypes, constraints etc) and import huge csv files as tables into it.
- Has made me work on a huge dataset.
- Has made me think and apply logic and use the best out of them with proper explanation so that the audience can relate with it.

- How to retrieve/extract the useful information/records as per the requirements along with what I think would help more better.
- How to frame the query, which function or clause to use, at the same time making sure not to make it very complicated.
- Has made me think like an analyst, considering what makes the entire thing perfect with relevant and valuable insights.
- Have tried providing visual representations where ever it seems fit and easy to understand. Thus, gave me a chance to use my excel knowledge of graphs and made the entire process more engaging.
- The whole process has made me more confident with the huge dataset and advanced SQL now.