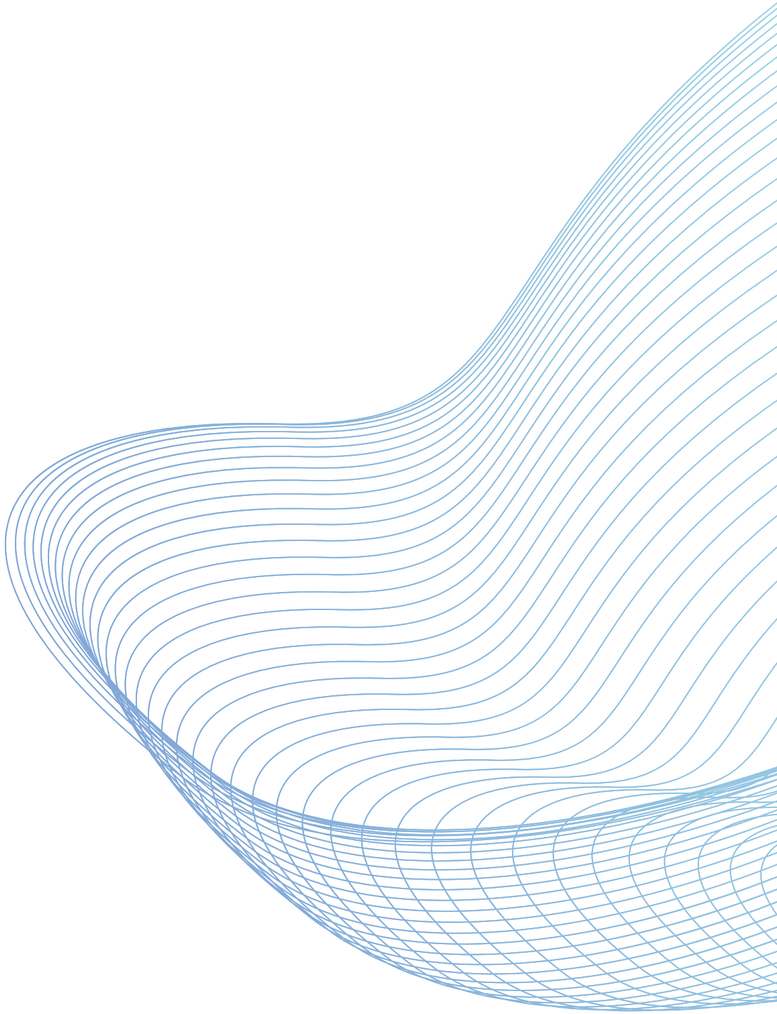
Trainity

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

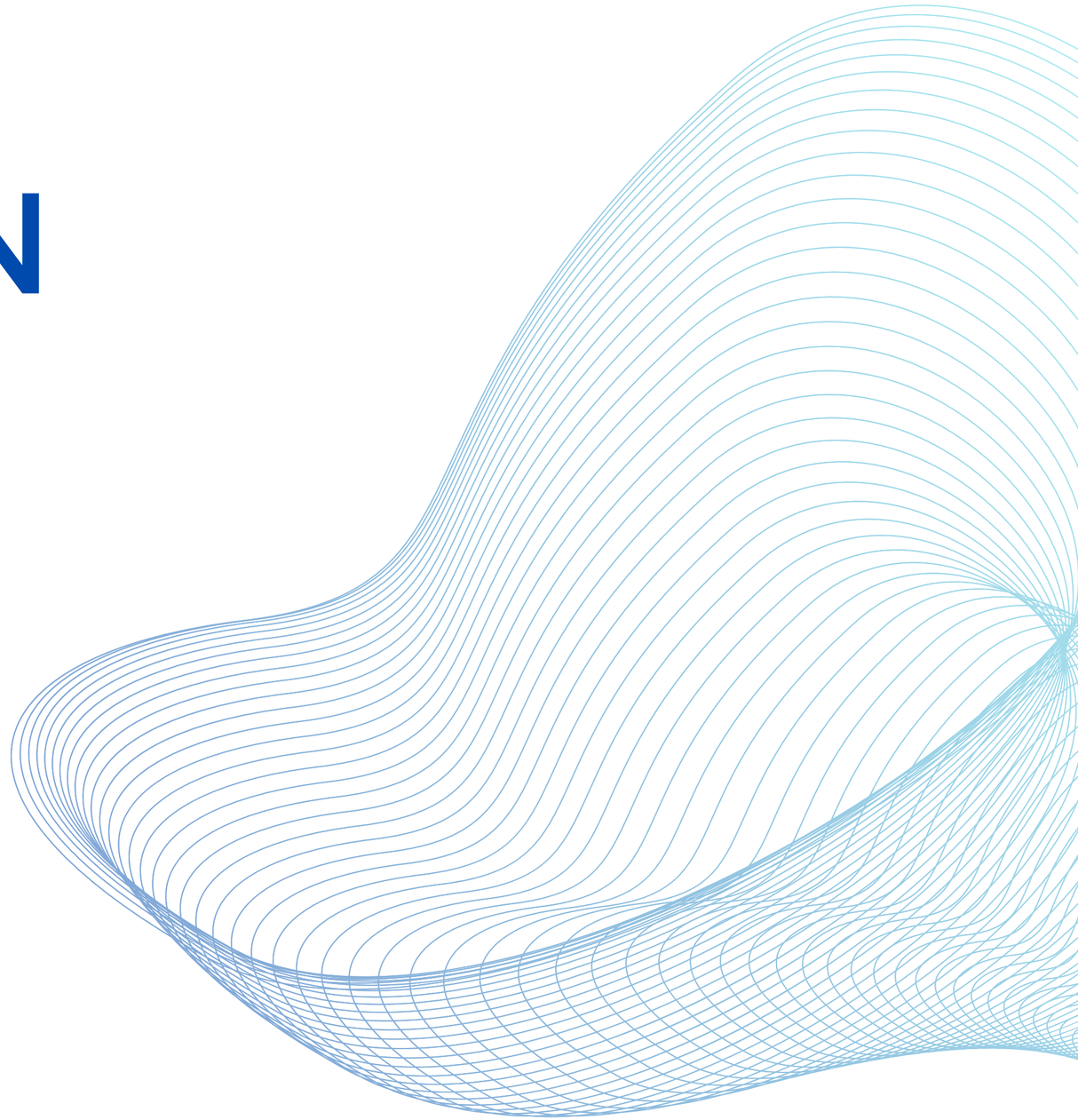


BY- SACHIN S

TABLE OF CONTENT

INTRODUCTION

This project aims to analyze the end to end operations of a company. We use dataset which contains details of users, their emails and events. Other dataset contains details of employees of a company like job_id, event, language etc.
To analyze the data we have certain queries to get data from the dataset.



DESCRIPTION

The project has the data of employees and the users along with emails and events and this data helps to analyze the data.

There are few categories under which company gets the retention rate and the monthly usage of the events of company.
We will be using SQL which gives the selected users from the dataset.
Dataset contains certain tables and values.

APPROACH

The dataset contains details of jobs in the company, users and the email and events of the users.
To analyze the data of the users of company there are certain strategies like looking for the retention rate and percentage use of each language in 30 days etc.
To find the users under certain given criteria we use SQL to get the output from the large dataset.
We use subqueries concept as well.

TECH-STACK USED

The tech-stack used to work on this project is SQL WORKBENCH 8.0 CE (version 8.0.32).
SQL helps to retrieve data from the large dataset.
Used MODE website as well for the second case study.

INSIGHTS

To analyze the dataset there is certain criteria to retrieve the data and get the result. The retention rate, jobs reviewed per hour per day etc.
We use SQL to get the retrieved data from the tables.

1. NUMBER OF JOBS REVIEWED

Number of jobs reviewed per hour taking date and the sum of time spent on the platform.

```
l> SELECT ds, SUM(time_spent) / COUNT(*) AS avg_time_spent FROM job WHERE ds BETWEEN '2020-11-01' AND '2020-11-30' GROUP BY ds ORDER BY d
---------+----------------+
         | avg_time_spent |
---------+----------------+
020-11-25 |        45.0000 |
020-11-26 |        56.0000 |
020-11-27 |       104.0000 |
020-11-28 |        16.5000 |
020-11-29 |        20.0000 |
020-11-30 |        20.0000 |
---------+----------------+
ws in set (0.01 sec)
```

2. THROUGHPUT

Number of events happening per second. We take the date, events per day and taking the average of distinct events happening per day.

```
mysql> SELECT ds, event_per_day, AVG(event_per_day)OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND  CURRENT ROW)AS 7_day_rolling_avg FROM  (SELECT ds, COUNT(DISTINCT event) AS event_per_day  FROM job WHERE ds BETW
EEN '2020-11-01' and '2020-11-30'  GROUP BY ds  ORDER BY ds)a;
+------------+---------------+-------------------+
| ds         | event_per_day | 7_day_rolling_avg |
+------------+---------------+-------------------+
| 2020-11-25 |             1 |            1.0000 |
| 2020-11-26 |             1 |            1.0000 |
| 2020-11-27 |             1 |            1.0000 |
| 2020-11-28 |             2 |            1.2500 |
| 2020-11-29 |             1 |            1.2000 |
| 2020-11-30 |             2 |            1.3333 |
+------------+---------------+-------------------+
6 rows in set (0.01 sec)
```

3. PERCENTAGE SHARE OF EACH LANGUAGE

To find the percentage use of each language for the last 30 days. We use the total language and all the rows among the table.

```
mysql> select  count(job.language)  as  total_of_each_language,((count(job.language)/(select count(*) from job))*100)percentage_share_of_each_language from job group by job.language;
+-------------------------+-----------------------------------+
| total_of_each_language | percentage_share_of_each_language |
+-------------------------+-----------------------------------+
|                      1 |                           12.5000 |
|                      1 |                           12.5000 |
|                      3 |                           37.5000 |
|                      1 |                           12.5000 |
|                      1 |                           12.5000 |
|                      1 |                           12.5000 |
+-------------------------+-----------------------------------+
6 rows in set (0.01 sec)
```

4. DUPLICATE ROWS

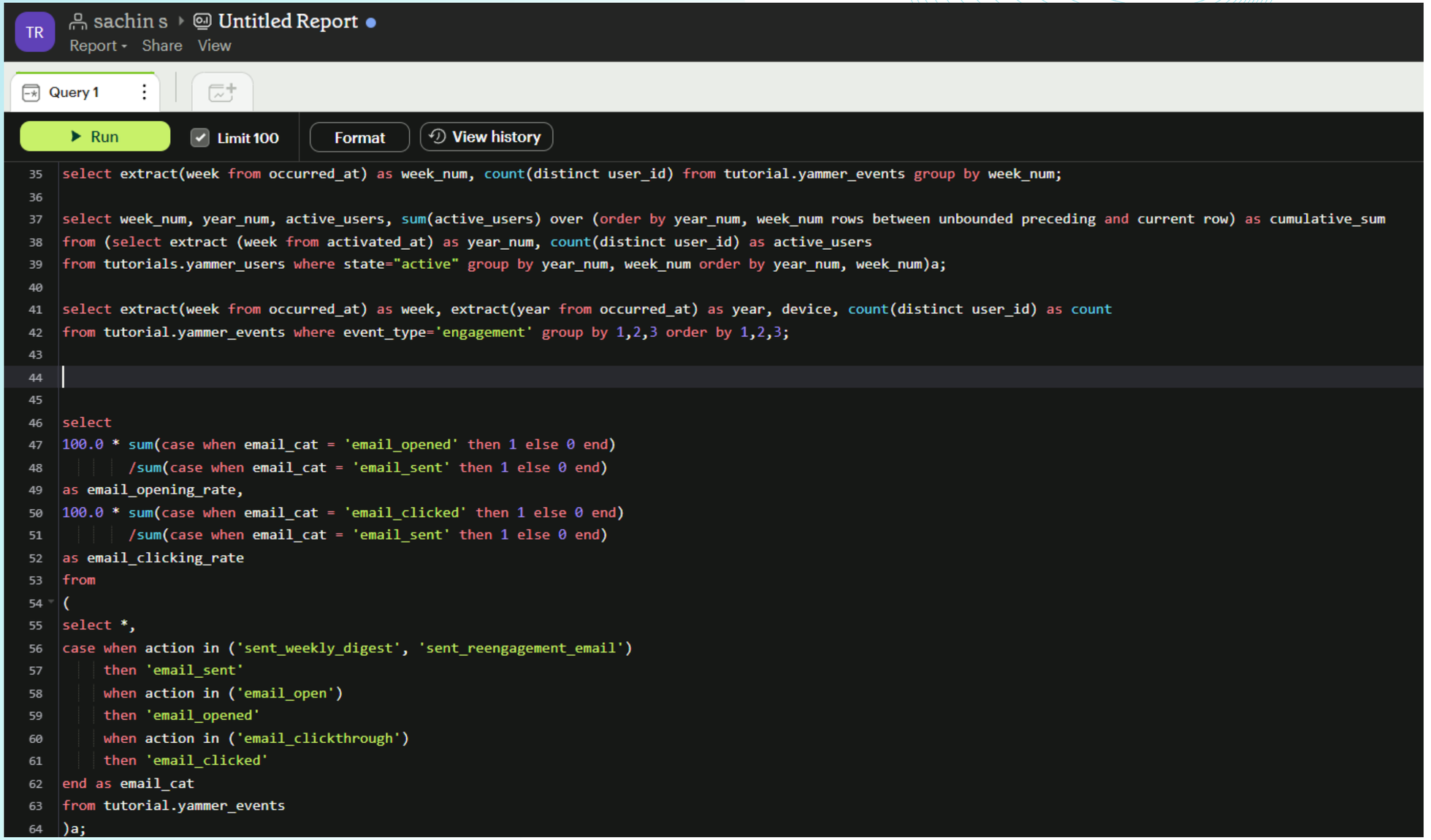
To check if we have the same values for multiple entries. We just check the count if greater than 1 that row repeats.

```
mysql> SELECT * FROM job WHERE job_id IN (SELECT job_id FROM job GROUP BY job_id HAVING COUNT(*) > 1);
+------------+--------+----------+----------+----------+------------+------+
| ds         | job_id | actor_id | event    | language | time_spent | org  |
+------------+--------+----------+----------+----------+------------+------+
| 2020-11-29 |     23 |     1003 | decision | Persian  |         20 | C    |
| 2020-11-28 |     23 |     1005 | transfer | Persian  |         22 | D    |
| 2020-11-26 |     23 |     1004 | skip     | Persian  |         56 | A    |
+------------+--------+----------+----------+----------+------------+------+
3 rows in set (0.00 sec)
```

5. CASE STUDY 2
INVESTIGATING METRIC SPIKE

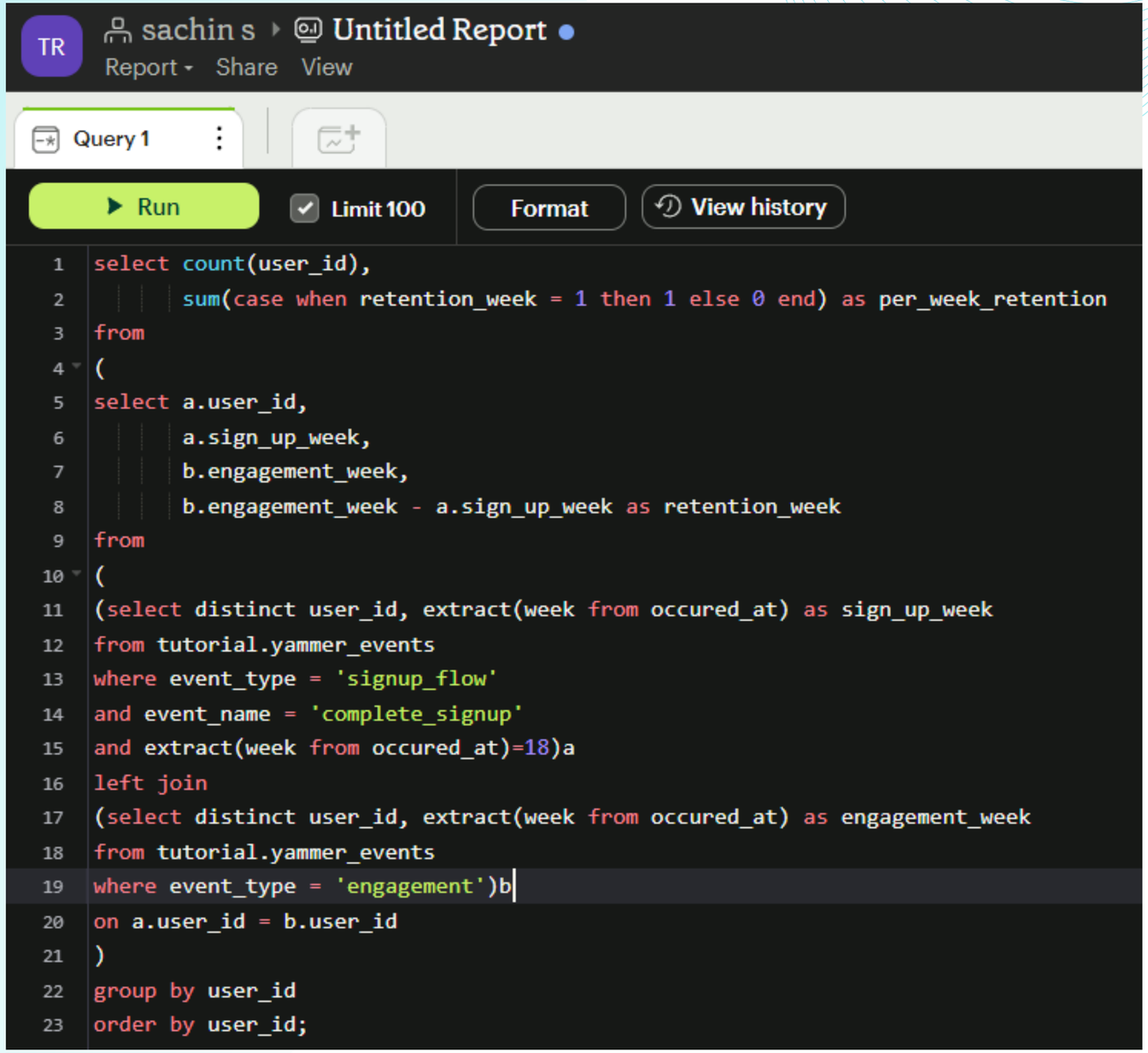
- **User Engagement -** weekly user usage, to find the weekly usage of the product
- **User Growth -** user growth of product, how many users are using product for long time
- **Weekly Retention -** weekly retention, how many users use the product after a week
- **Weekly Engagement -** weekly engagement per device
- **Email Engagement -** how many users use the email services

6. QUERY 1,2,3,4




```sql
select extract(week from occurred_at) as week_num, count(distinct user_id) from tutorial.yammer_events group by week_num;

select week_num, year_num, active_users, sum(active_users) over (order by year_num, week_num rows between unbounded preceding and current row) as cumulative_sum
from (select extract (week from activated_at) as year_num, count(distinct user_id) as active_users
from tutorials.yammer_users where state="active" group by year_num, week_num order by year_num, week_num)a;

select extract(week from occurred_at) as week, extract(year from occurred_at) as year, device, count(distinct user_id) as count
from tutorial.yammer_events where event_type='engagement' group by 1,2,3 order by 1,2,3;



select
100.0 * sum(case when email_cat = 'email_opened' then 1 else 0 end)
        /sum(case when email_cat = 'email_sent' then 1 else 0 end)
as email_opening_rate,
100.0 * sum(case when email_cat = 'email_clicked' then 1 else 0 end)
        /sum(case when email_cat = '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_cat
from tutorial.yammer_events
)a;
```

7. QUERY 5




```sql
select count(user_id),
        sum(case when retention_week = 1 then 1 else 0 end) as per_week_retention
from
(
select a.user_id,
        a.sign_up_week,
        b.engagement_week,
        b.engagement_week - a.sign_up_week as retention_week
from
(
(select distinct user_id, extract(week from occured_at) as sign_up_week
from tutorial.yammer_events
where event_type = 'signup_flow'
and event_name = 'complete_signup'
and extract(week from occured_at)=18)a
left join
(select distinct user_id, extract(week from occured_at) as engagement_week
from tutorial.yammer_events
where event_type = 'engagement')b
on a.user_id = b.user_id
)
group by user_id
order by user_id;
```

RESULT

While working on this project I used SQL WORKBENCH which helped me to understand new topics like subqueries, limit etc.
The main use of SQL was to retrieve the data according to the requirement.
This project helped me how to analyze the data and use WORKBENCH.

THANK YOU