Trainity project 3

Operation Analytics and Investigating Metric Spike

Project Description:

Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. This analysis helps identify areas for improvement within the company. As a Data Analyst, you'll work closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect.

Approach:

First we extracted the data given into tables and analyzed user engagement, event counts, and distribution patterns, comparing them across time periods to pinpoint irregularities etc.

Tech-stacks used:

We extracted and cleaned the table data in excel and loaded the table data to mysql to write the sql queries to analyze the data.

Insights:

Identified a significant spike in event generation during specific weeks.

Observed that the average events per user and possible bot activity.

Weekly user engagement trends revealed periods of high variability correlating with specific marketing campaigns.

Result:

The analysis successfully pinpointed the source, enabling corrective measures to be taken. This project enhanced our understanding of operational metrics, user behavior, and how external factors influence engagement, driving more informed decision-making.

Case Study 1: Job Data Analysis

You will be working with a table named job_data with the following columns:

- job_id: Unique identifier of jobs
- actor_id: Unique identifier of actor
- event: The type of event (decision/skip/transfer).
- language: The Language of the content
- time_spent: Time spent to review the job in seconds.
- org: The Organization of the actor
- ds: The date in the format yyyy/mm/dd (stored as text).

To create the table job_data from the given data

Initial code:

```
create table job_data
(ds date,
job_id int not null,
actor_id int not null,
event varchar(50) not null,
language varchar(50) not null,
time_spent int not null,
org char(2)
);

INSERT INTO job_data(ds, job_id, actor_id, event, language, time_spent, org)
VALUES
('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),
('2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),
('2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),
('2020-11-28', 23, 1005, 'transfer', 'Persian', 22, 'D'),
('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),
('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),
```

('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),

('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');

Tasks:

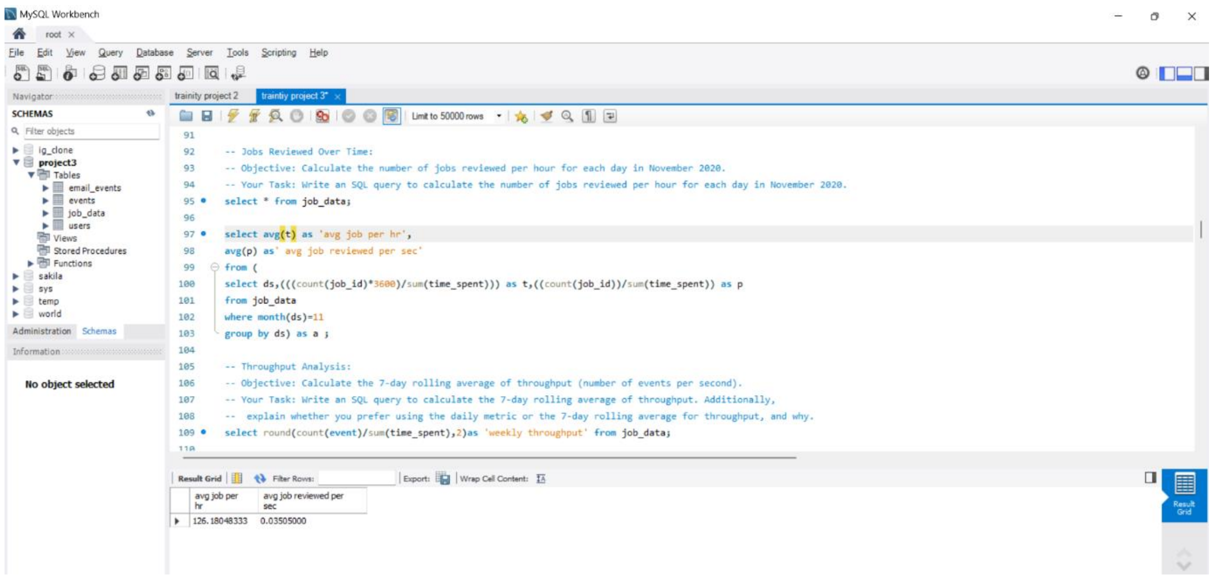
A. **Jobs Reviewed Over Time:**

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

Code:

```
select avg(t) as 'avg job per hr',
avg(p) as' avg job reviewed per sec'
from (
select ds,(((count(job_id)*3600)/sum(time_spent))) as t,((count(job_id))/sum(time_spent)) as p
from job_data
where month(ds)=11
group by ds) as a ;
```

Output:



```
-- Jobs Reviewed Over Time:
-- Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.
-- Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.
select * from job_data;

select avg(t) as 'avg job per hr',
avg(p) as' avg job reviewed per sec'
from (
select ds,(((count(job_id)*3600)/sum(time_spent))) as t,((count(job_id))/sum(time_spent)) as p
from job_data
where month(ds)=11
group by ds) as a ;

-- Throughput Analysis:
-- Objective: Calculate the 7-day rolling average of throughput (number of events per second).
-- Your 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.
select round(count(event)/sum(time_spent),2)as 'weekly throughput' from job_data;
```

avg job per hr	avg job reviewed per sec
126.18048333	0.03505000

B. **Throughput Analysis:**

- Objective: Calculate the 7-day rolling average of throughput (number of events per second).
- Your 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.

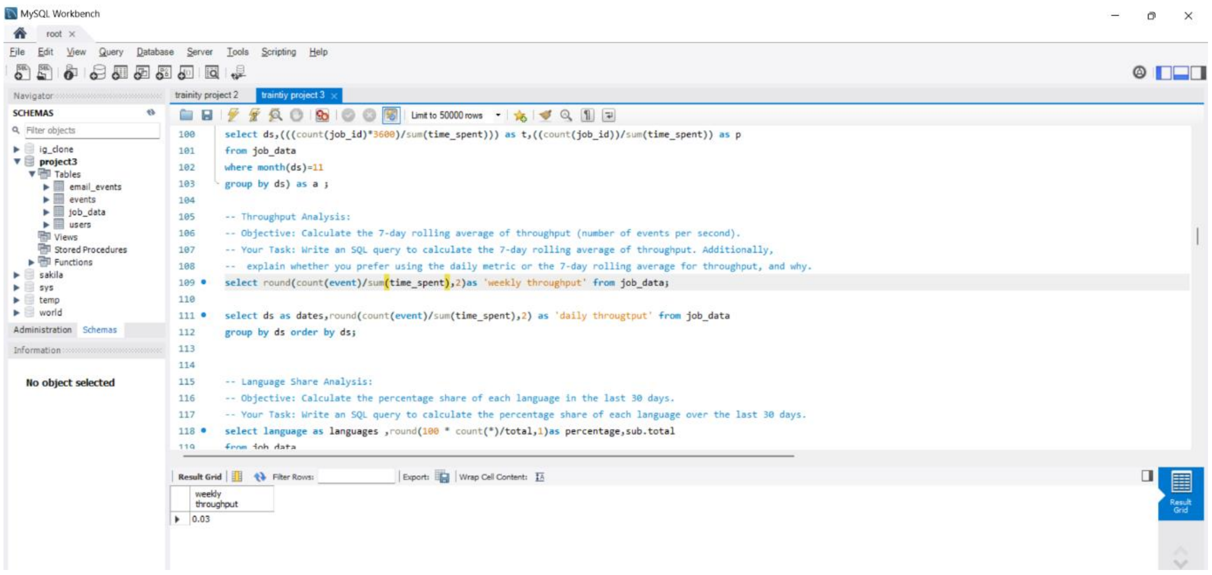
Code:

select round(count(event)/sum(time_spent),2)as 'weekly throughput' from job_data;

select ds as dates,round(count(event)/sum(time_spent),2) as 'daily througtput' from job_data

group by ds order by ds;

Output:



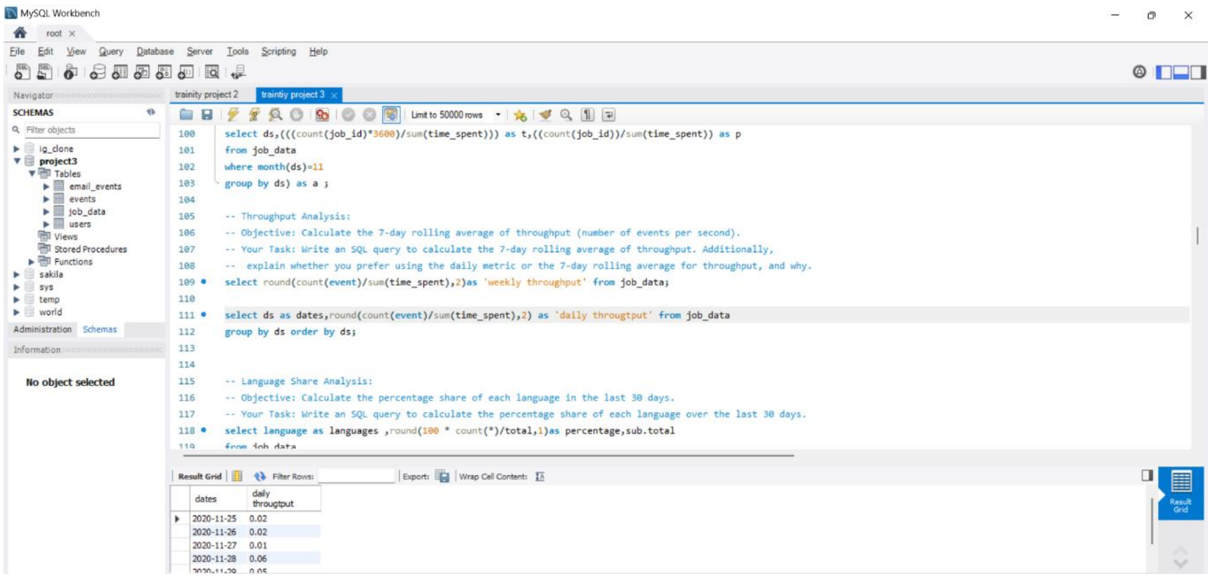
```
select ds,(((count(job_id)*3600)/sum(time_spent))) as t,((count(job_id))/sum(time_spent)) as p
from job_data
where month(ds)=11
group by ds) as a ;

-- Throughput Analysis:
-- Objective: Calculate the 7-day rolling average of throughput (number of events per second).
-- Your 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.
select round(count(event)/sum(time_spent),2)as 'weekly throughput' from job_data;

select ds as dates,round(count(event)/sum(time_spent),2) as 'daily througtput' from job_data
group by ds order by ds;


-- Language Share Analysis:
-- Objective: Calculate the percentage share of each language in the last 30 days.
-- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.
select language as languages ,round(100 * count(*)/total,1)as percentage,sub.total
from job data
```

```
weekly throughput
0.03
```



```
select ds,(((count(job_id)*3600)/sum(time_spent))) as t,((count(job_id))/sum(time_spent)) as p
from job_data
where month(ds)=11
group by ds) as a ;

-- Throughput Analysis:
-- Objective: Calculate the 7-day rolling average of throughput (number of events per second).
-- Your 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.
select round(count(event)/sum(time_spent),2)as 'weekly throughput' from job_data;

select ds as dates,round(count(event)/sum(time_spent),2) as 'daily througtput' from job_data
group by ds order by ds;


-- Language Share Analysis:
-- Objective: Calculate the percentage share of each language in the last 30 days.
-- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.
select language as languages ,round(100 * count(*)/total,1)as percentage,sub.total
from job_data
```

```
dates        daily
             througtput
2020-11-25   0.02
2020-11-26   0.02
2020-11-27   0.01
2020-11-28   0.06
2020-11-29   0.05
```

C. **Language Share Analysis:**

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

Code:

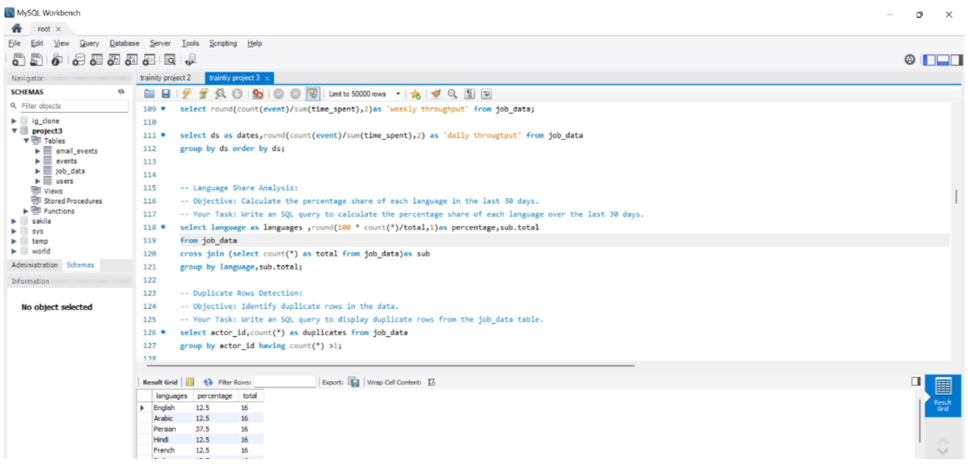
select language as languages ,round(100 * count(*)/total,1)as percentage,sub.total

from job_data

cross join (select count(*) as total from job_data)as sub

group by language,sub.total;

Output:



```
select round(count(event)/sum(time_spent),2)as 'weekly throughput' from job_data;

select ds as dates,round(count(event)/sum(time_spent),2) as 'daily througtput' from job_data
group by ds order by ds;


-- Language Share Analysis:
-- Objective: Calculate the percentage share of each language in the last 30 days.
-- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.
select language as languages ,round(100 * count(*)/total,1)as percentage,sub.total
from job_data
cross join (select count(*) as total from job_data)as sub
group by language,sub.total;

-- Duplicate Rows Detection:
-- Objective: Identify duplicate rows in the data.
-- Your Task: Write an SQL query to display duplicate rows from the job_data table.
select actor_id,count(*) as duplicates from job_data
group by actor_id having count(*) >1;
```

```
languages   percentage   total
English     12.5         16
Arabic      12.5         16
Persian     37.5         16
Hindi       12.5         16
French      12.5         16
```

D. **Duplicate Rows Detection:**

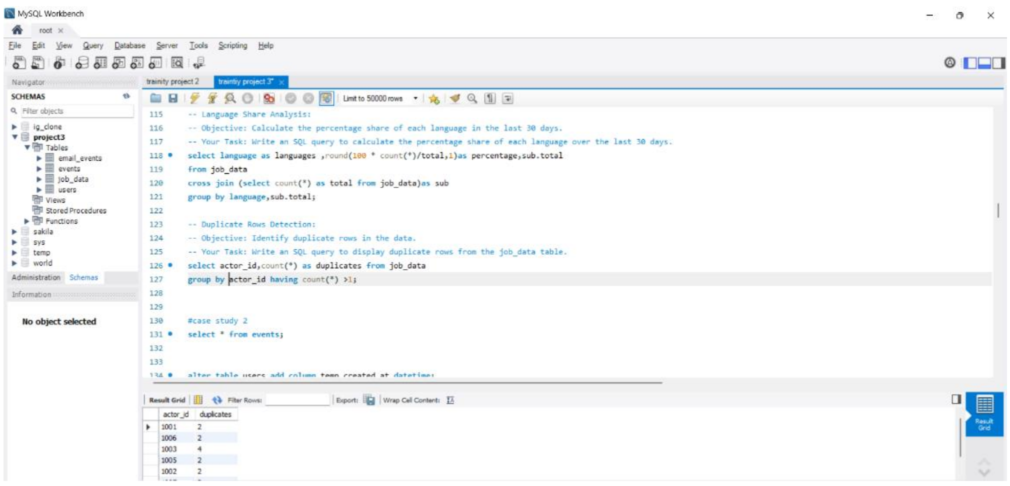
- Objective: Identify duplicate rows in the data.
- Your Task: Write an SQL query to display duplicate rows from the job_data table.

Code:

select actor_id,count(*) as duplicates from job_data

group by actor_id having count(*) >1;

Output:



```
-- Language Share Analysis:
-- Objective: Calculate the percentage share of each language in the last 30 days.
-- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.
select language as languages ,round(100 * count(*)/total,1)as percentage,sub.total
from job_data
cross join (select count(*) as total from job_data)as sub
group by language,sub.total;

-- Duplicate Rows Detection:
-- Objective: Identify duplicate rows in the data.
-- Your Task: Write an SQL query to display duplicate rows from the job_data table.
select actor_id,count(*) as duplicates from job_data
group by actor_id having count(*) >1;


#case study 2
select * from events;


alter table users add column temp_created_at datetime;
```

actor_id	duplicates
1001	2
1006	2
1003	4
1005	2
1002	2

Case Study 2: Investigating Metric Spike

You will be working with three tables:

- users: Contains one row per user, with descriptive information about that user's account.
- events: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).
- email_events: Contains events specific to the sending of emails.

After cleaning the tables data in excel we have to load the tables in mysql

Initial code:

create table users(

user_id int,

created_at varchar(100),

company_id int,

language varchar(100),

```
activated_at varchar(100),
state varchar(100));

show variables like 'secure_file_priv';

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv"
into table users
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;

select * from users;

create table events(
user_id int,
occurred_at varchar(100),
event_type varchar(100),
event_name varchar(100),
location varchar(100),
device varchar(100),
user_type int);

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/events.csv"
into table events
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;

select * from events;
```

```
create table email_events(
user_id int,
occurred_at     varchar(100),
action varchar(100),
user_type int);

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv"
into table email_events
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```

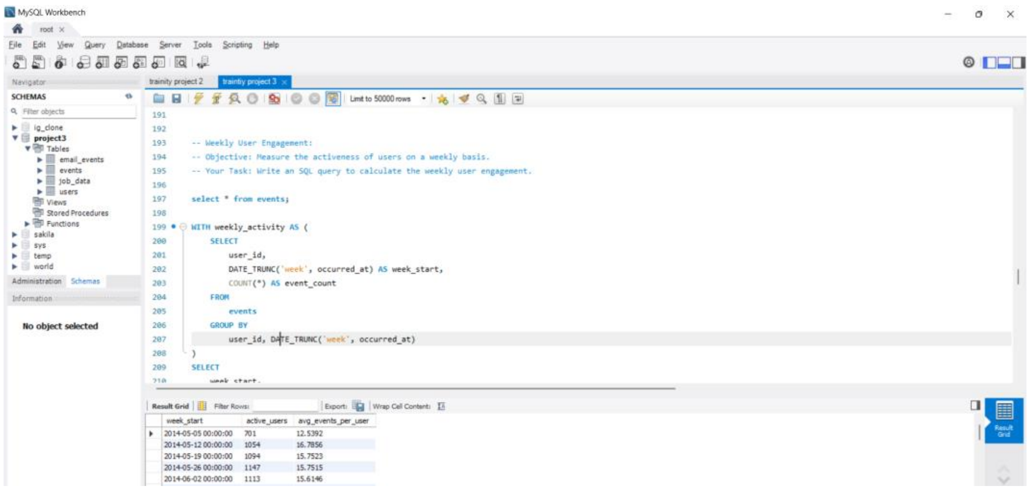
Tasks:

A. **Weekly User Engagement:**
   - Objective: Measure the activeness of users on a weekly basis.
   - Your Task: Write an SQL query to calculate the weekly user engagement.

Code:

```
WITH weekly_activity AS (
  SELECT
    user_id,
    DATE_TRUNC('week', occurred_at) AS week_start,
    COUNT(*) AS event_count
  FROM
    events
  GROUP BY
    user_id, DATE_TRUNC('week', occurred_at)
)
SELECT
    week_start,
    COUNT(DISTINCT user_id) AS active_users,
    AVG(event_count) AS avg_events_per_user
FROM
    weekly_activity
GROUP BY
    week_start
ORDER BY
    week_start;
```

Output:



```
-- Weekly User Engagement:
-- Objective: Measure the activeness of users on a weekly basis.
-- Your Task: Write an SQL query to calculate the weekly user engagement.

select * from events;

WITH weekly_activity AS (
    SELECT
        user_id,
        DATE_TRUNC('week', occurred_at) AS week_start,
        COUNT(*) AS event_count
    FROM
        events
    GROUP BY
        user_id, DATE_TRUNC('week', occurred_at)
)
SELECT
```

week_start	active_users	avg_events_per_user
2014-05-05 00:00:00	701	12.5392
2014-05-12 00:00:00	1054	16.7856
2014-05-19 00:00:00	1094	15.7523
2014-05-26 00:00:00	1147	15.7515
2014-06-02 00:00:00	1113	15.6146

B. **User Growth Analysis:**

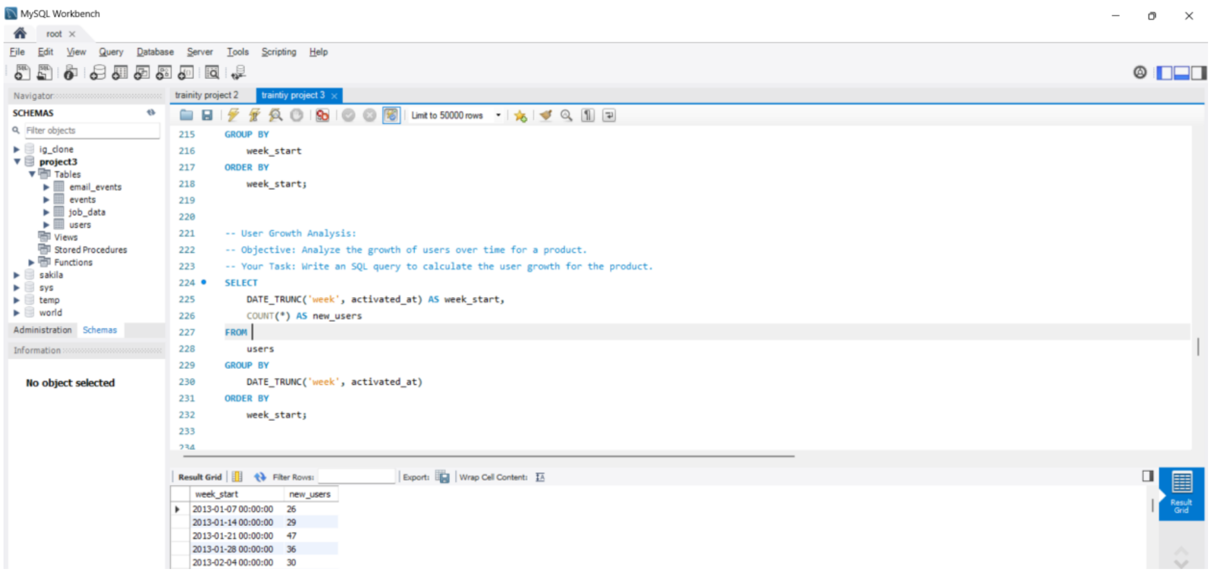
- Objective: Analyze the growth of users over time for a product.
- Your Task: Write an SQL query to calculate the user growth for the product.

Code:

```
SELECT
    DATE_TRUNC('week', activated_at) AS week_start,
    COUNT(*) AS new_users
FROM
    users
```

```
GROUP BY
    DATE_TRUNC('week', activated_at)
ORDER BY
    week_start;
```

Output:



```
GROUP BY
    week_start
ORDER BY
    week_start;


-- User Growth Analysis:
-- Objective: Analyze the growth of users over time for a product.
-- Your Task: Write an SQL query to calculate the user growth for the product.
SELECT
    DATE_TRUNC('week', activated_at) AS week_start,
    COUNT(*) AS new_users
FROM
    users
GROUP BY
    DATE_TRUNC('week', activated_at)
ORDER BY
    week_start;
```

week_start	new_users
2013-01-07 00:00:00	26
2013-01-14 00:00:00	29
2013-01-21 00:00:00	47
2013-01-28 00:00:00	36
2013-02-04 00:00:00	30

C. **Weekly Retention Analysis:**

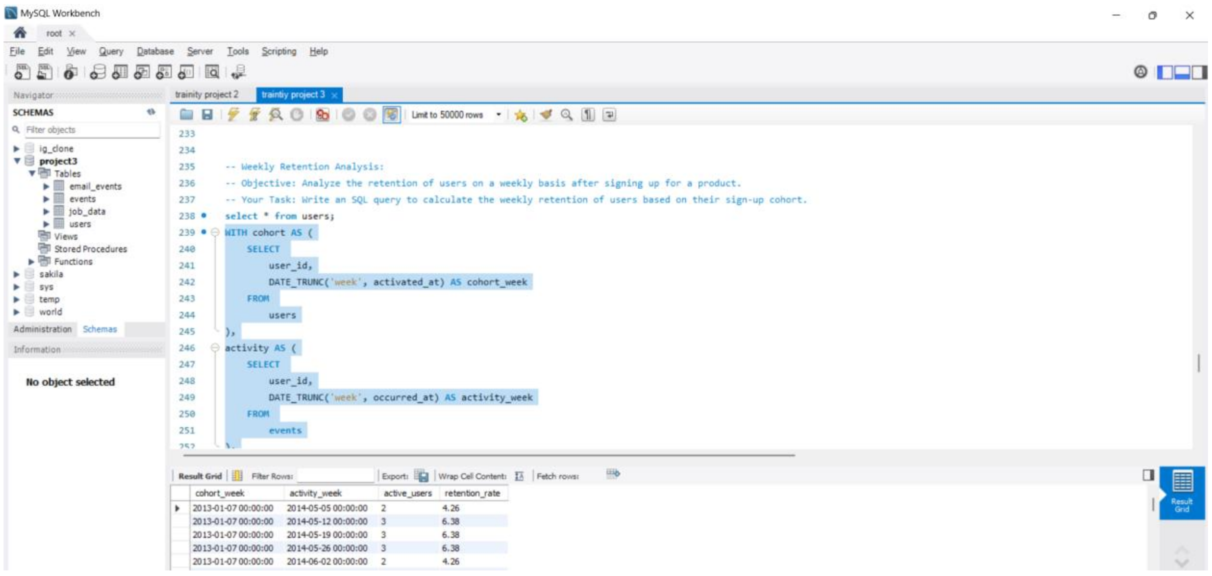
- Objective: Analyze the retention of users on a weekly basis after signing up for a product.
- Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Code:

```
WITH cohort AS (
    SELECT
        user_id,
        DATE_TRUNC('week', activated_at) AS cohort_week
    FROM
        users
),
activity AS (
    SELECT
```

```
        user_id,
        DATE_TRUNC('week', occurred_at) AS activity_week
    FROM
        events
),
cohort_activity AS (
    SELECT
        c.cohort_week,
        a.activity_week,
        COUNT(DISTINCT a.user_id) AS active_users
    FROM
        cohort c
    JOIN
        activity a ON c.user_id = a.user_id
    GROUP BY
        c.cohort_week, a.activity_week
)
SELECT
    cohort_week,
    activity_week,
    active_users,
    ROUND(100.0 * active_users / SUM(active_users) OVER (PARTITION BY cohort_week), 2) AS
retention_rate
FROM
    cohort_activity
ORDER BY
    cohort_week, activity_week;
```

Output:



```
-- Weekly Retention Analysis:
-- Objective: Analyze the retention of users on a weekly basis after signing up for a product.
-- Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.
select * from users;
WITH cohort AS (
    SELECT
        user_id,
        DATE_TRUNC('week', activated_at) AS cohort_week
    FROM
        users
),
activity AS (
    SELECT
        user_id,
        DATE_TRUNC('week', occurred_at) AS activity_week
    FROM
        events
```

cohort_week	activity_week	active_users	retention_rate
2013-01-07 00:00:00	2014-05-05 00:00:00	2	4.26
2013-01-07 00:00:00	2014-05-12 00:00:00	3	6.38
2013-01-07 00:00:00	2014-05-19 00:00:00	3	6.38
2013-01-07 00:00:00	2014-05-26 00:00:00	3	6.38
2013-01-07 00:00:00	2014-06-02 00:00:00	2	4.26

D. **Weekly Engagement Per Device:**

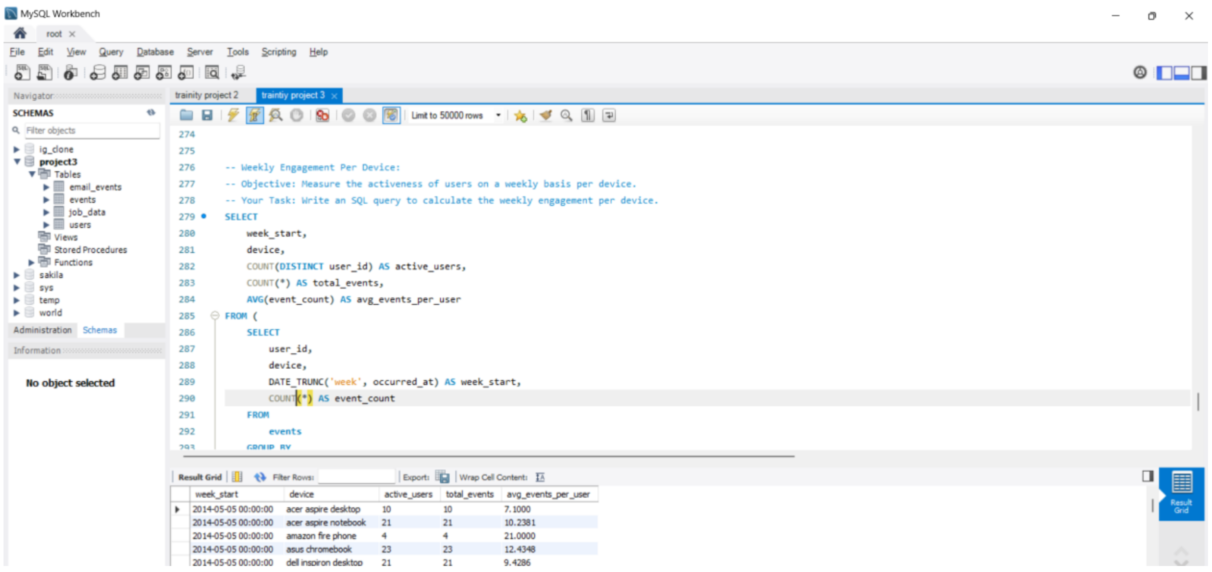
- Objective: Measure the activeness of users on a weekly basis per device.
- Your Task: Write an SQL query to calculate the weekly engagement per device.

Code:

```
SELECT
  week_start,
  device,
  COUNT(DISTINCT user_id) AS active_users,
  COUNT(*) AS total_events,
  AVG(event_count) AS avg_events_per_user
FROM (
  SELECT
    user_id,
    device,
    DATE_TRUNC('week', occurred_at) AS week_start,
    COUNT(*) AS event_count
  FROM
    events
  GROUP BY
```

```
        user_id, device, DATE_TRUNC('week', occurred_at)
) AS user_device_activity
GROUP BY
    week_start, device
ORDER BY
    week_start, device
LIMIT 50000;
```

Output:



```
-- Weekly Engagement Per Device:
-- Objective: Measure the activeness of users on a weekly basis per device.
-- Your Task: Write an SQL query to calculate the weekly engagement per device.
SELECT
    week_start,
    device,
    COUNT(DISTINCT user_id) AS active_users,
    COUNT(*) AS total_events,
    AVG(event_count) AS avg_events_per_user
FROM (
    SELECT
        user_id,
        device,
        DATE_TRUNC('week', occurred_at) AS week_start,
        COUNT(*) AS event_count
    FROM
        events
    GROUP BY
```

week_start	device	active_users	total_events	avg_events_per_user
2014-05-05 00:00:00	acer aspire desktop	10	10	7.1000
2014-05-05 00:00:00	acer aspire notebook	21	21	10.2381
2014-05-05 00:00:00	amazon fire phone	4	4	21.0000
2014-05-05 00:00:00	asus chromebook	23	23	12.4348
2014-05-05 00:00:00	dell inspiron desktop	21	21	9.4286

E. **Email Engagement Analysis:**

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

Code:

```
SELECT
    DATE_TRUNC('week', occurred_at) AS week_start,
    action,
    COUNT(*) AS event_count,
    COUNT(DISTINCT user_id) AS unique_users
FROM
    email_events
```

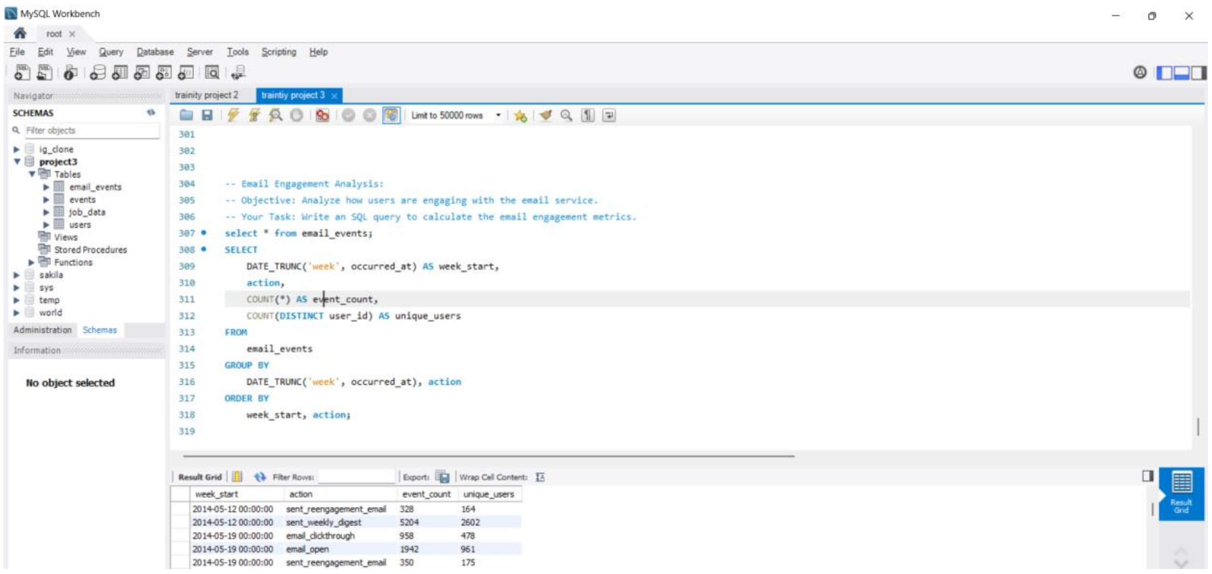
GROUP BY

    DATE_TRUNC('week', occurred_at), action

ORDER BY

    week_start, action;

Output:



```
-- Email Engagement Analysis:
-- Objective: Analyze how users are engaging with the email service.
-- Your Task: Write an SQL query to calculate the email engagement metrics.
select * from email_events;
SELECT
    DATE_TRUNC('week', occurred_at) AS week_start,
    action,
    COUNT(*) AS event_count,
    COUNT(DISTINCT user_id) AS unique_users
FROM
    email_events
GROUP BY
    DATE_TRUNC('week', occurred_at), action
ORDER BY
    week_start, action;
```

week_start	action	event_count	unique_users
2014-05-12 00:00:00	sent_reengagement_email	328	164
2014-05-12 00:00:00	sent_weekly_digest	5204	2602
2014-05-19 00:00:00	email_clickthrough	958	478
2014-05-19 00:00:00	email_open	1942	961
2014-05-19 00:00:00	sent_reengagement_email	350	175

Thank you