Paul Damein K

05-01-2025

PROJECT- 3 OPERATIONAL ANALYTICS AND INVESTIGATING METRIC SPIKE

trainity

MySQL™

PROJECT DESCRIPTION:

trainity

- Operational analytics is crucial for improving a company's operations by extracting valuable insights from collected data. As a lead Data Analyst at a company like Microsoft, I collaborated with the operations, support, and marketing teams. With the help of my advanced SQL skills, I examine their data to pinpoint areas for improvement and provided the company understand with insights into unexpected changes in key metrics.
- My project focuses on two main case studies. The first, Job Data Analysis, aims to analyse operational metrics related to job reviews. The second, Investigating Metric Spike, seeks to understand user engagement and activity patterns. Both case studies leverage advanced SQL techniques to derive valuable insights and address specific business questions. This project will provide actionable recommendations to optimize operations and enhance user experience.

PROJECT APPROACH:

trainity

1. DATABASE SETUP:

- **Create Database and Table: A database is created, and tables are defined according to the given structure. Also, import CSV files into MySQL Workbench.**

2. ANALYSIS EXECUTION:

In the analysis phase, I am going to use advanced SQL techniques to answer the questions asked for case studies.

- **Job Data Analysis: In this part, using SQL aggregation functions and series analysis, I worked on calculating average scores, identification trends, and anomaly detection in job reviews.**
- **Investigating Metric Spike: User engagement trends are analyzed by aggregating activity data. Spikes are detected by comparing current data against historical averages.**

3. REPORT PREPARATION:

- **Compile Findings: The analysis results were summarized in a detailed report. This report, created in PPT presentation format, summarized the key findings by table trends and making actionable recommendations. The clear and crisp presentation of insights, driven by data, had to be given to the leadership team for meaningful decision-making and improvements in operations and user experience.**

TECH-STACK USED:

trainity

- The tech-stack used for the project is MySQL workbench,
- community server version 8.0.
- We also used Microsoft Excel for cleaning the data as shown in the tutorial on how to load large data to MySQL.
- The data files we in csv. Format

1) CASE STUDY 1: JOB DATA ANALYSIS

trainity

Creating 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)

QUERY FOR CREATING DB AND LOADING DATA INTO MYSQL

trainity

Create Database for core project 3:

```
CREATE DATABASE core_project3;

USE core_project3;
```

Loading Data into MySQL server:

```
CREATE TABLE job_data (
    ds DATE,
    job_id INT,
    actor_id INT,
    event VARCHAR(10),
    language VARCHAR(20),
    time_spent INT,
    org VARCHAR(2)
);
```

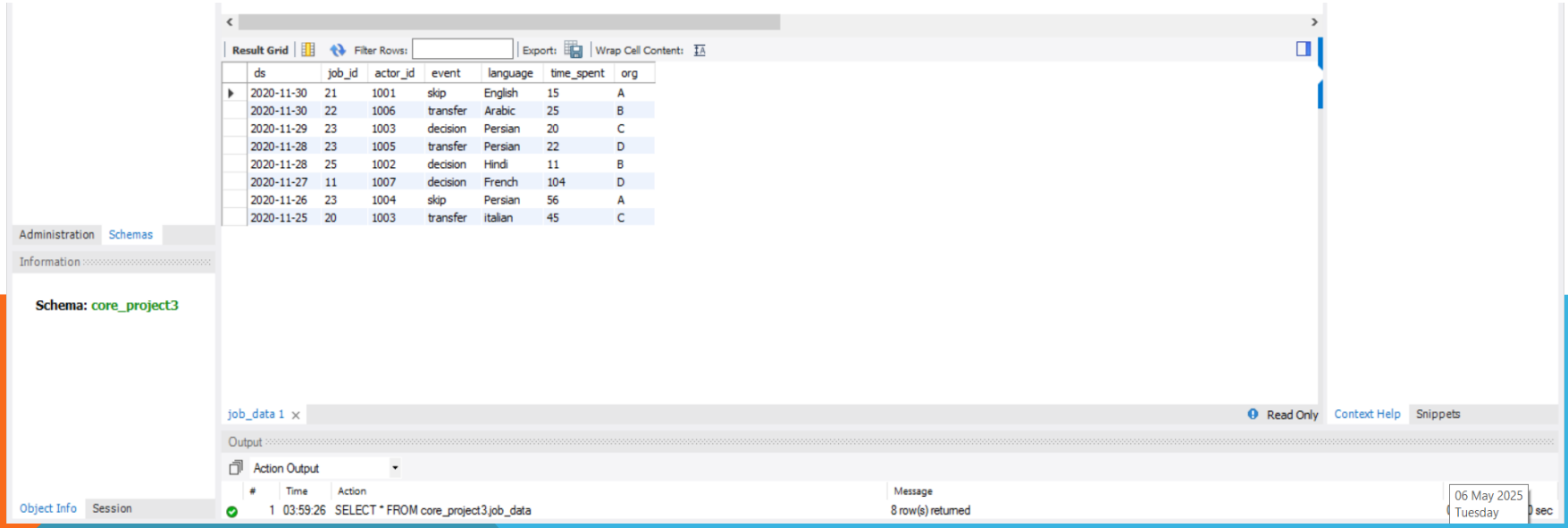
Query for loading data into the table Job_data:

```
SELECT *
FROM   job_data;

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');
```

RESULT:

Query:

Output:



CASE STUDY 1: TASKS

trainity

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.

SQL Query:

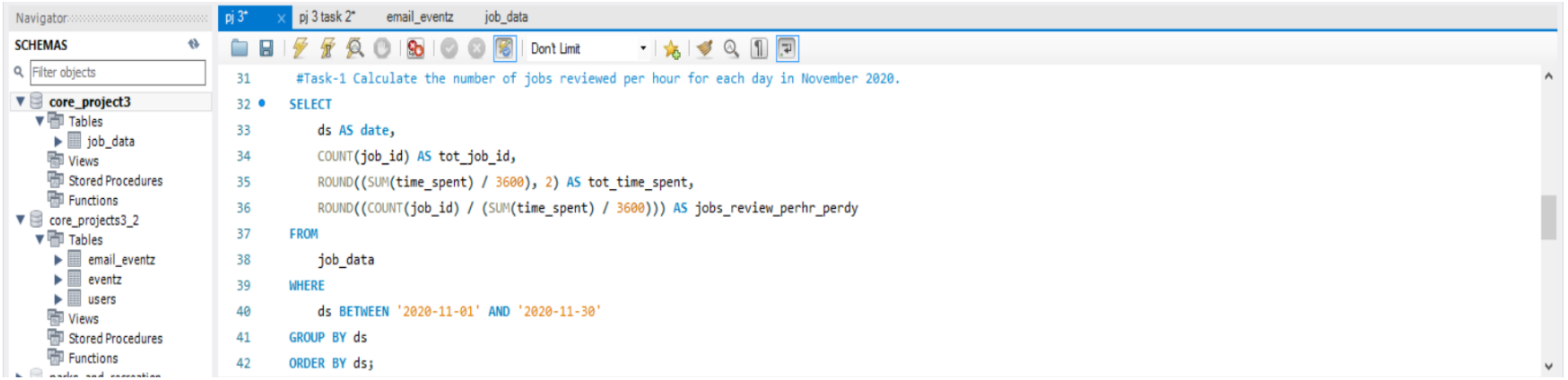
```
SELECT ds
    AS date,
       Count(job_id)
    AS tot_job_id,
       Round(( Sum(time_spent) / 3600 ), 2)
    AS tot_time_spent,
       Round(( Count(job_id) / ( Sum(time_spent) / 3600 )
 )) AS
       jobs_review_perhr_perdy
FROM   job_data
WHERE  ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP  BY ds
ORDER  BY ds;
```

Insight:

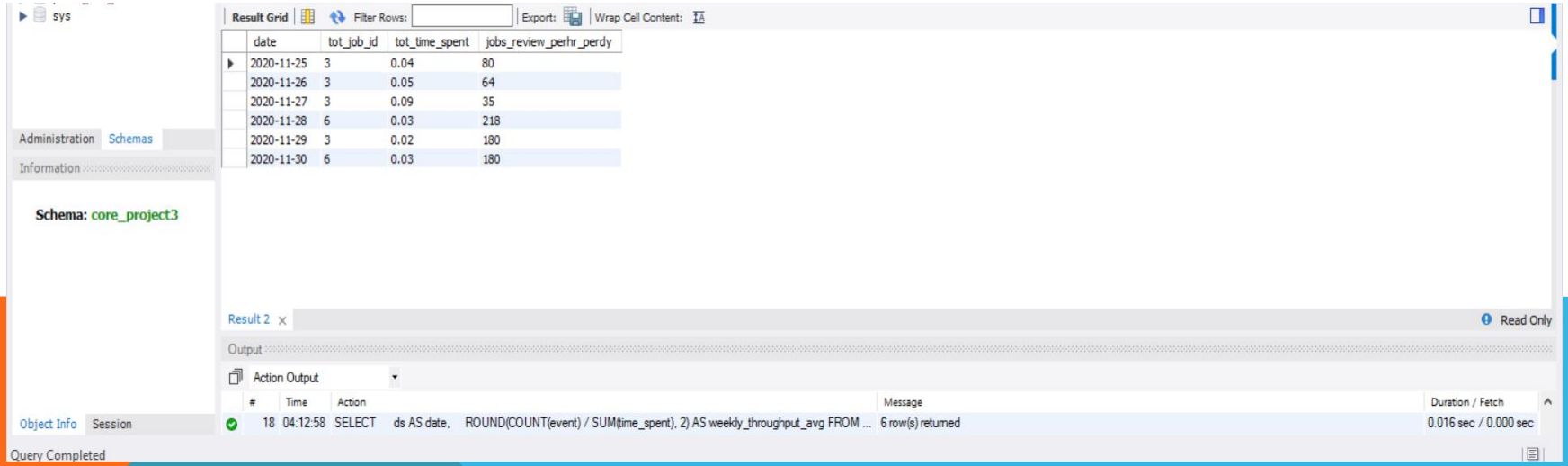
- **From the table, we can observe that 0.01 jobs reviewed per hour for each day in November 2020.**
- **The highest jobs reviewed is on 28th November 2020 with 218.16 per hour.**

RESULT:

Query:



Output:



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.

SQL Query:

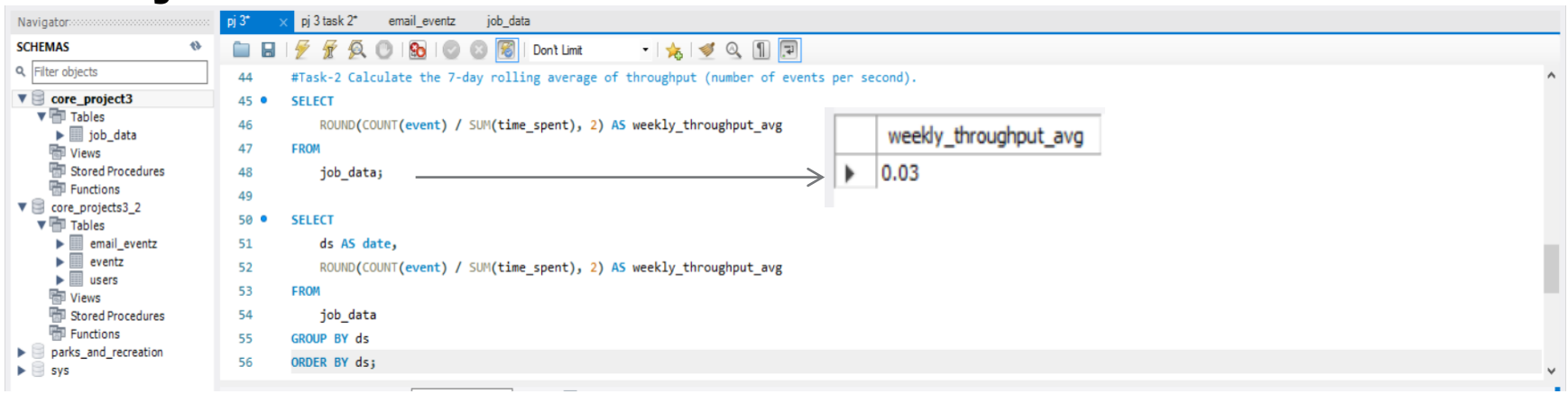
```
SELECT Round(Count(event) / Sum(time_spent), 2) AS weekly
_throughput_avg
FROM   job_data;

SELECT ds                                AS date,
     Round(Count(event) / Sum(time_spent), 2) AS weekly_thro
ughput_avg
FROM   job_data
GROUP  BY ds
ORDER  BY ds;
```

Insights:

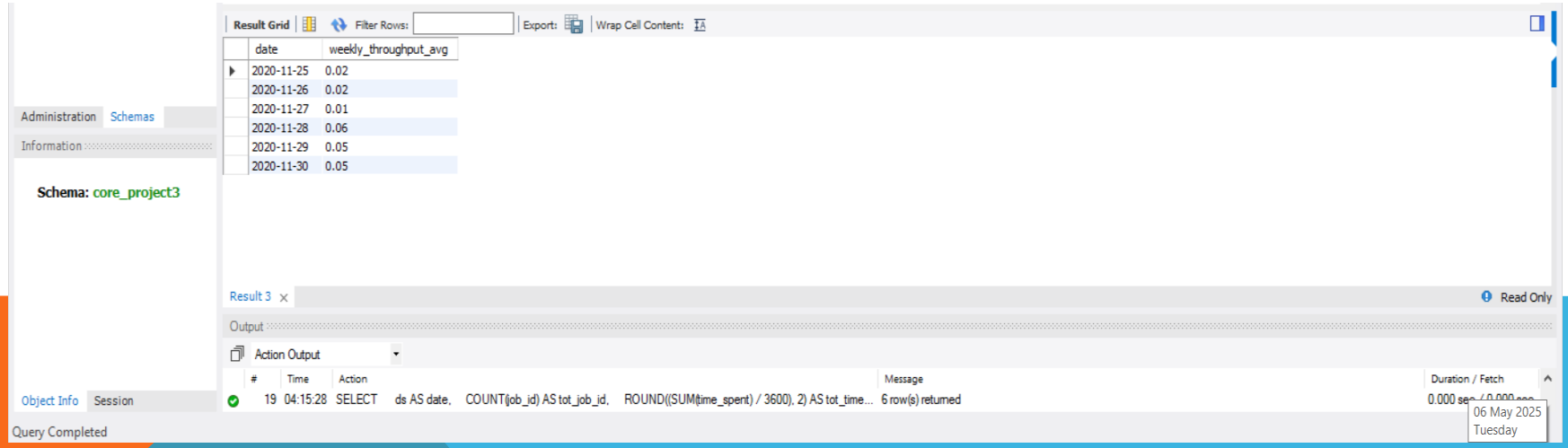
- The 7-day rolling average of throughput is between 0.01 to 0.06.
- The weekly average throughput is 0.03 events per second.
- 7 day rolling average will normally preferable for analyzing throughput to avoid fluctuating trends.

RESULT:

Query:



Output:



DAILY METRIC VS 7-DAY ROLLING AVERAGE:

The 7-day rolling average is often better for analysing throughput.

Here's a simpler way to put it in pointers:

- **Here's a simpler way to put it in pointers:**
- **Daily data can jump around a lot due to short-term things.**
- **A 7-day rolling average smooth these ups and downs.**
- **This gives a clearer picture of the overall trend in throughput over time.**
- **So, the rolling average provides a more stable and comprehensive view of performance than daily numbers.**

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.

SQL Query:

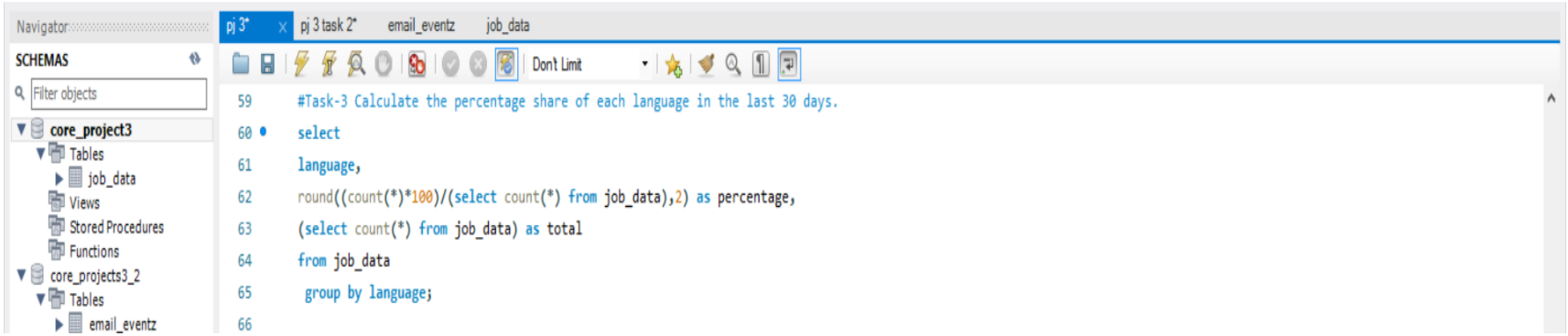
```
SELECT    language,
          Round((Count(*)*100)/
          (
                SELECT Count(*)
                FROM   job_data),2) AS percentage,
          (
                SELECT Count(*)
                FROM   job_data) AS total
FROM      job_data
GROUP BY language;#Task-
4 identify duplicate rows IN the dataSELECT   actor_id,
          Count(*) AS duplicate
FROM      job_data
GROUP BY actor_id
HAVING    Count(*)>1;
```

Insights:

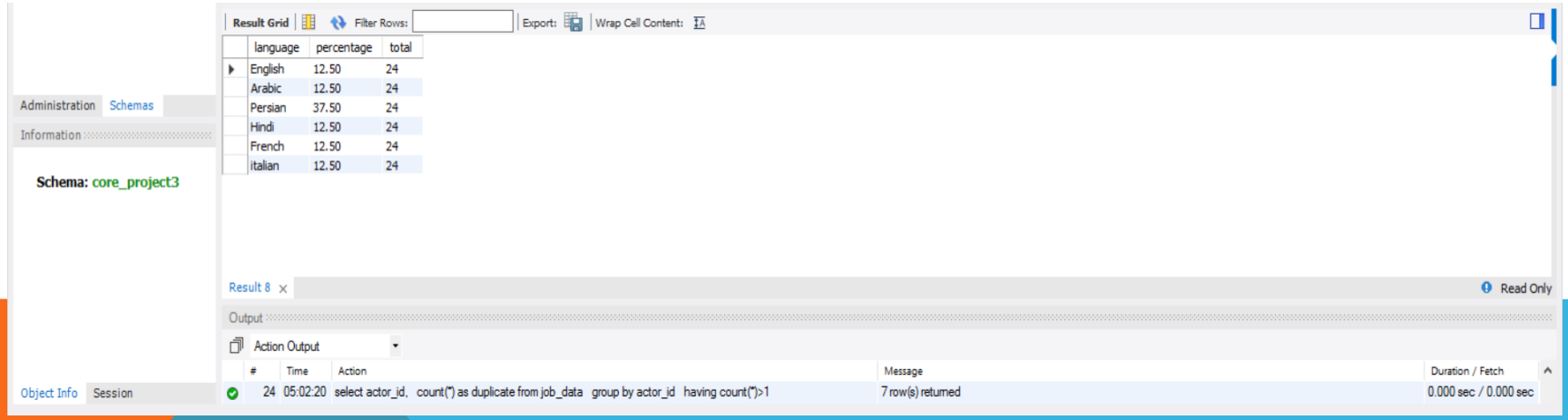
- Persian is the most frequent language, representing 37.50% of the data.
- English, Arabic, Hindi, French, and Italian are used equally, each at 12.50%.
- This analysis is based on a total of 8 data points.

RESULT:

Query:



Output:



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.

SQL Query:

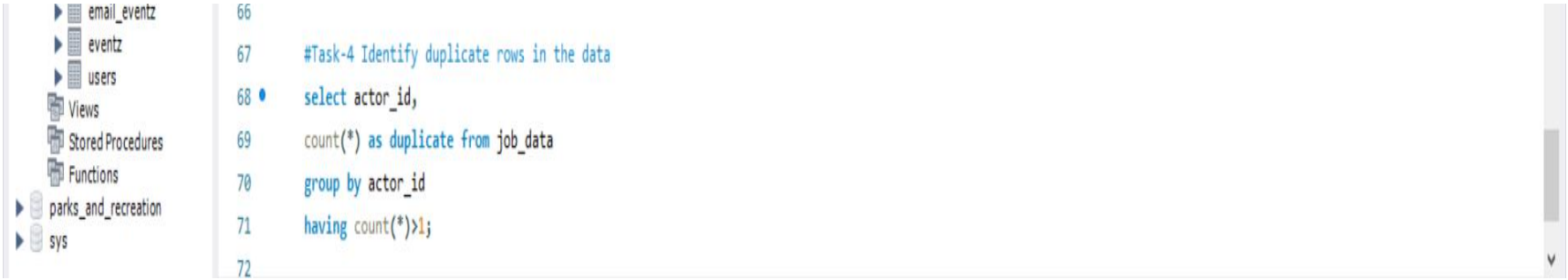
```
SELECT    language,
          Round((Count(*)*100)/
          (
                    SELECT Count(*)
                    FROM   job_data),2) AS percentage,
          (
                    SELECT Count(*)
                    FROM   job_data) AS total
FROM      job_data
GROUP BY language;#Task-
4 identify duplicate rows IN the dataSELECT   actor_id,
          Count(*) AS duplicate
FROM      job_data
GROUP BY actor_id
HAVING    Count(*)>1;
```

Insights:

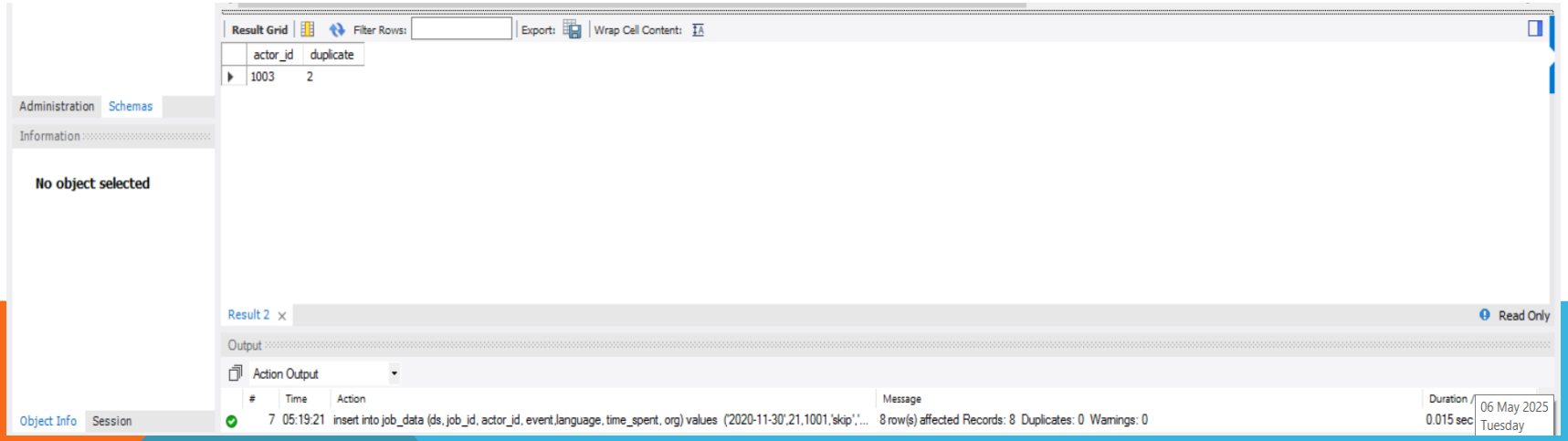
- **Out of 8 total rows, there are 2 duplicate rows.**
- **The actor_id 1003 is duplicated.**

RESULT:

Query:



Output:



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.

QUERY FOR CREATING DB AND LOADING DATA INTO MYSQL

trainity

Create Database for core project 3:

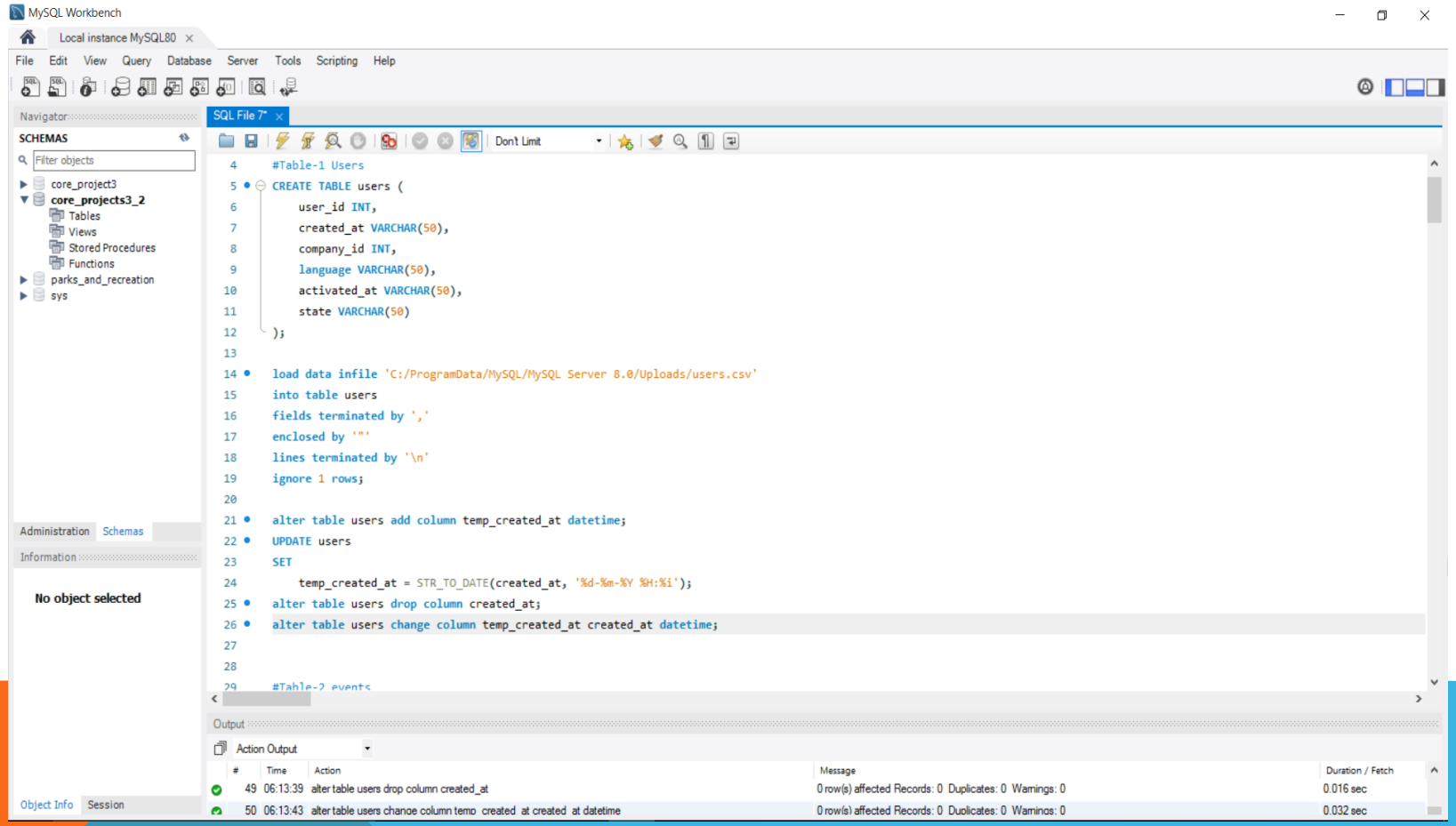
```
CREATE DATABASE core_projects3_2;

USE core_projects3_2;
```

Loading Data into MySQL server, TABLE-1:

```
CREATE TABLE users
  (
     user_id       INT,
     created_at    VARCHAR(50),
     company_id    INT,
     language      VARCHAR(50),
     activated_at  VARCHAR(50),
     state         VARCHAR(50)
  );

ALTER TABLE users
  ADD COLUMN temp_created_at DATETIME;

UPDATE users
SET    temp_created_at = Str_to_date(created_at, '%d-%m-%Y %H:%i');

ALTER TABLE users
  DROP COLUMN created_at;

ALTER TABLE users
  CHANGE COLUMN temp_created_at created_at DATETIME;
```

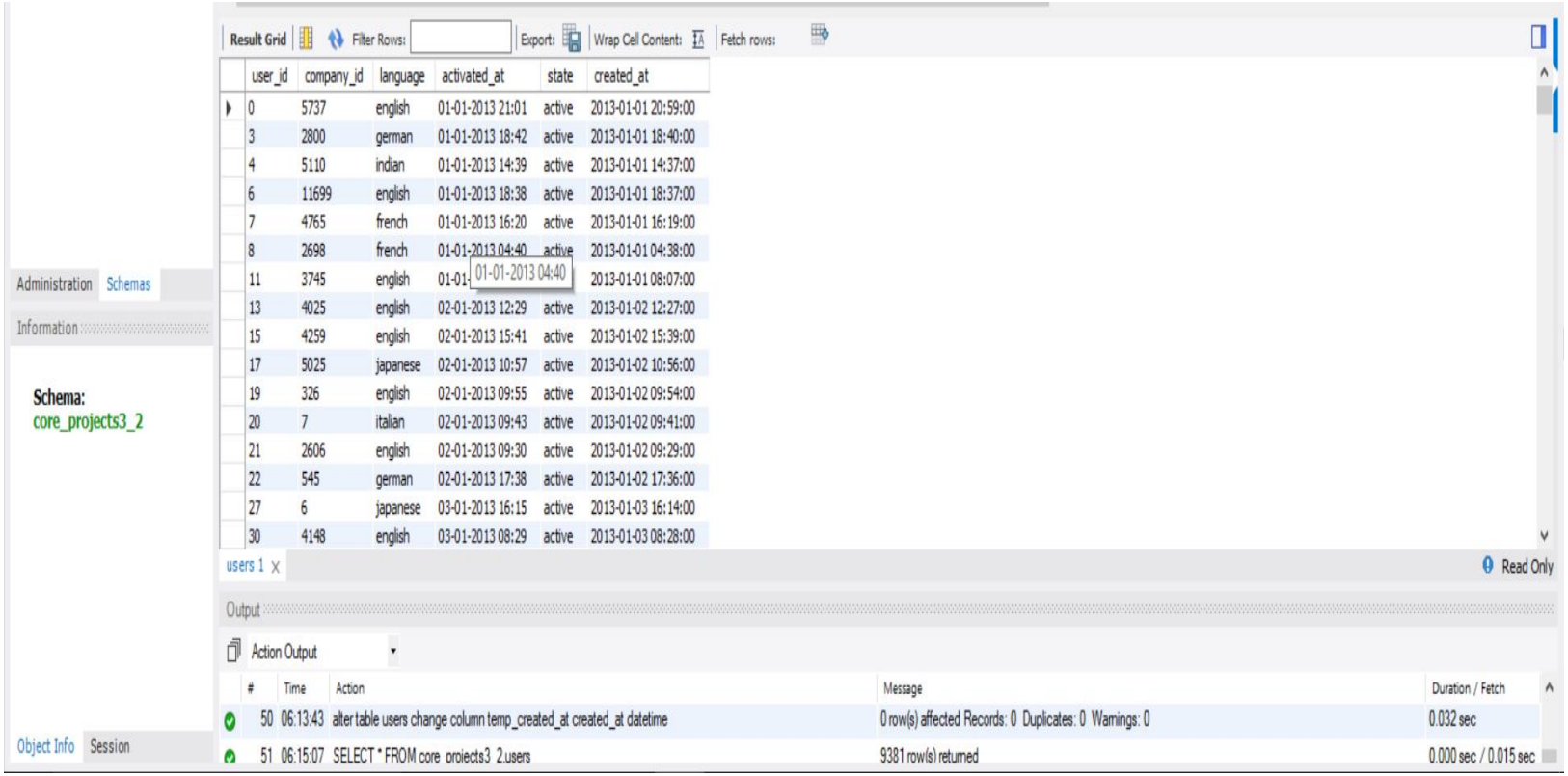
RESULT:

Query:



trainity

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

user_id	company_id	language	activated_at	state	created_at
0	5737	english	01-01-2013 21:01	active	2013-01-01 20:59:00
3	2800	german	01-01-2013 18:42	active	2013-01-01 18:40:00
4	5110	indian	01-01-2013 14:39	active	2013-01-01 14:37:00
6	11699	english	01-01-2013 18:38	active	2013-01-01 18:37:00
7	4765	french	01-01-2013 16:20	active	2013-01-01 16:19:00
8	2698	french	01-01-2013 04:40	active	2013-01-01 04:38:00
11	3745	english	01-01-2013 04:40		2013-01-01 08:07:00
13	4025	english	02-01-2013 12:29	active	2013-01-02 12:27:00
15	4259	english	02-01-2013 15:41	active	2013-01-02 15:39:00
17	5025	japanese	02-01-2013 10:57	active	2013-01-02 10:56:00
19	326	english	02-01-2013 09:55	active	2013-01-02 09:54:00
20	7	italian	02-01-2013 09:43	active	2013-01-02 09:41:00
21	2606	english	02-01-2013 09:30	active	2013-01-02 09:29:00
22	545	german	02-01-2013 17:38	active	2013-01-02 17:36:00
27	6	japanese	03-01-2013 16:15	active	2013-01-03 16:14:00
30	4148	english	03-01-2013 08:29	active	2013-01-03 08:28:00

01-01-2013 04:40

Administration Schemas

Information

Schema:
core_projects3_2

users 1 X

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
50	06:13:43	alter table users change column temp_created_at created_at datetime	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
51	06:15:07	SELECT * FROM core_projects3_2.users	9381 row(s) returned	0.000 sec / 0.015 sec

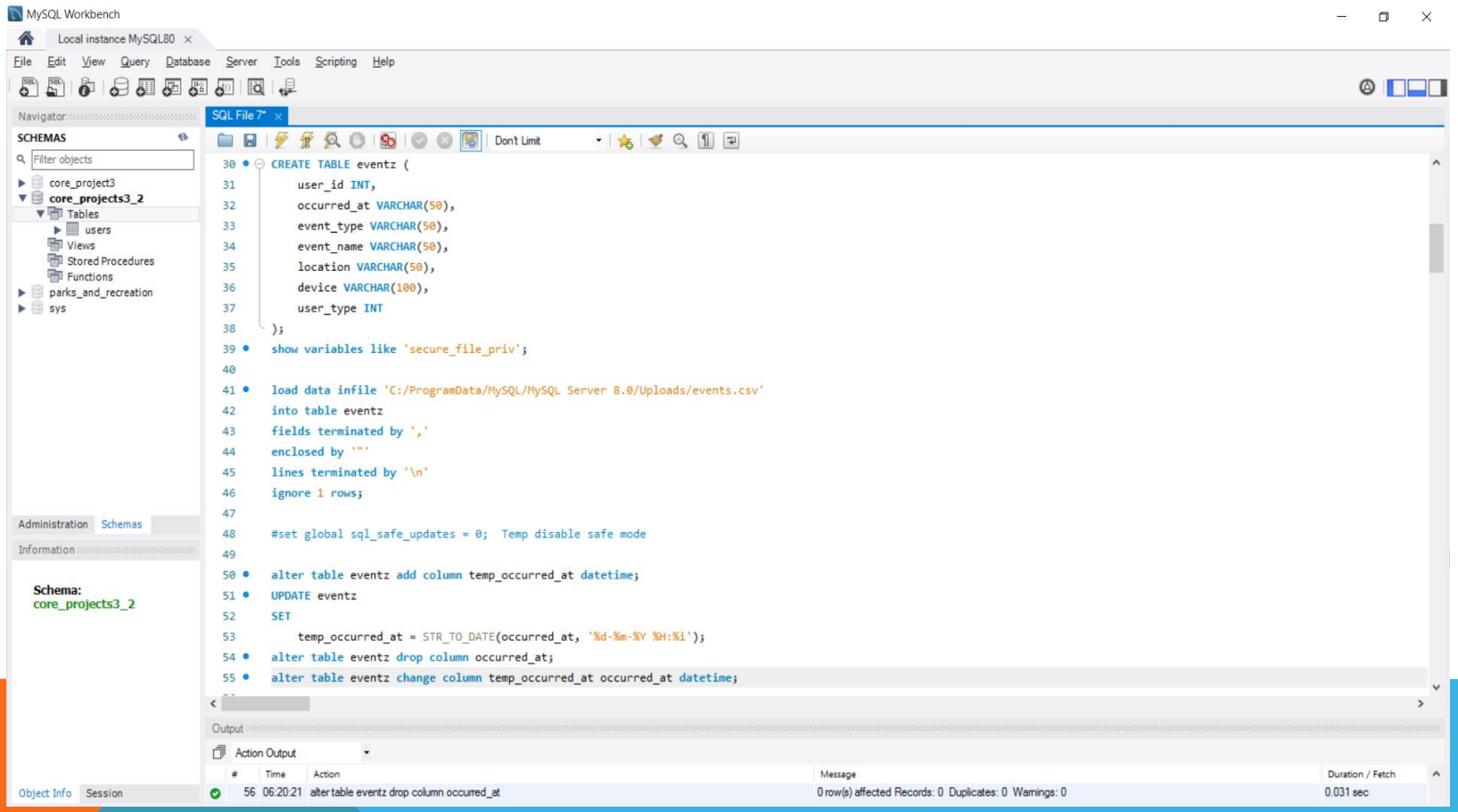
Object Info Session

TABLE-2:

```
CREATE TABLE eventz
(
user_id      INT,
occurred_at  VARCHAR(50),
event_type   VARCHAR(50),
event_name   VARCHAR(50),
location     VARCHAR(50),
device       VARCHAR(100),
user_type    INT
);SHOW variables LIKE 'secure_file_priv';LOAD data INFILE 'C:/ProgramData/MySQL/
MySQL Server 8.0/Uploads/events.csv INTO TABLE eventz FIELDS TERMINATED BY ',
ENCLOSED BY '"' LINES TERMINATED BY '\n
IGNORE 1 rows;
/* # SET global sql_safe_updates = 0;*/Temp disable safe mode
ALTER TABLE eventz ADD COLUMN temp_occurred_at datetime;UPDATE eventz
SET    temp_occurred_at = Str_to_date(occurred_at, '%d-%m-
%Y %H:%i');ALTER TABLE eventz DROP COLUMN occurred_at;ALTER TABLE eventz CHANGE
COLUMN temp_occurred_at occurred_At DATETIME;
```

RESULT:

Query:



trainity

Output:

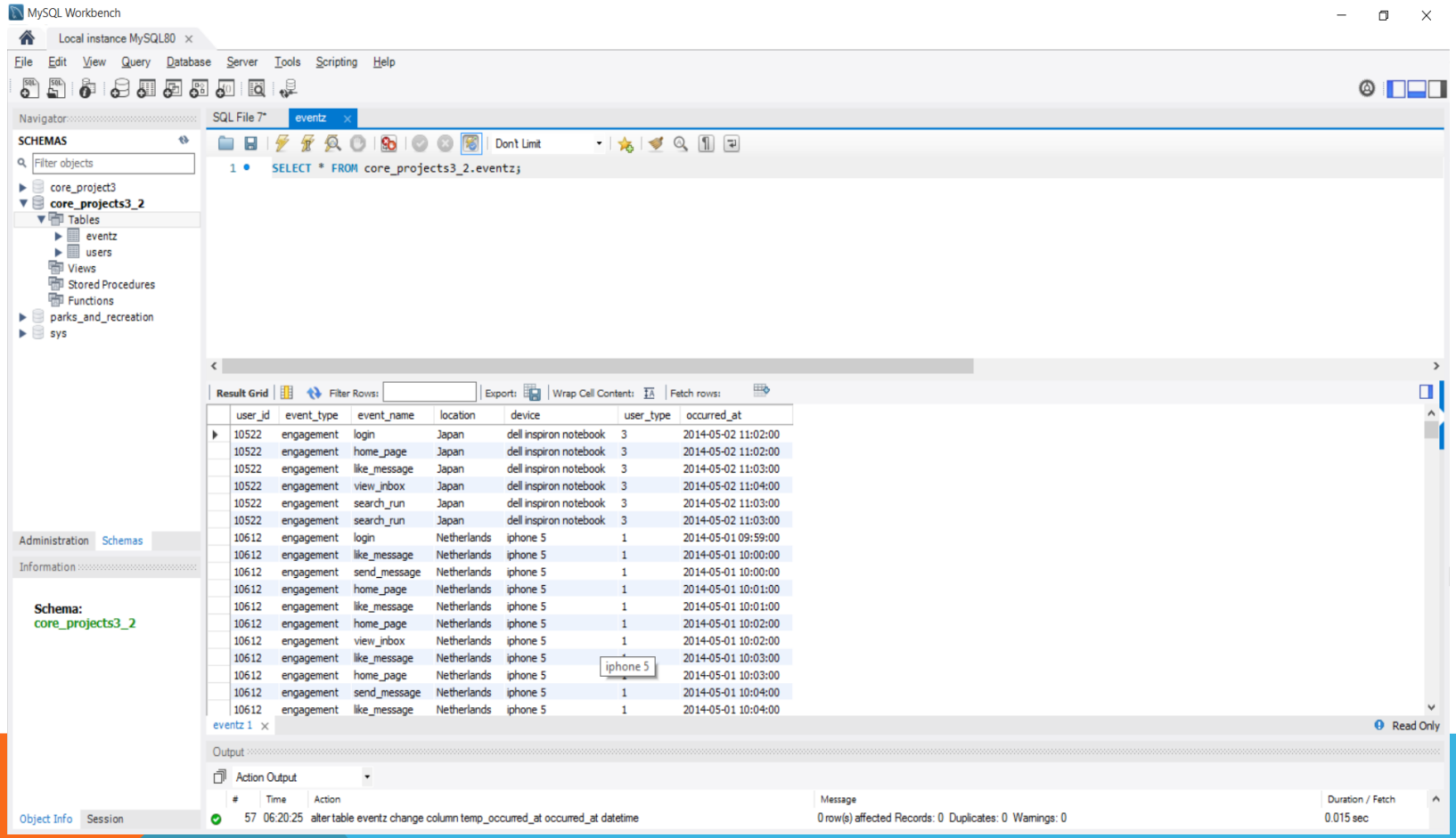
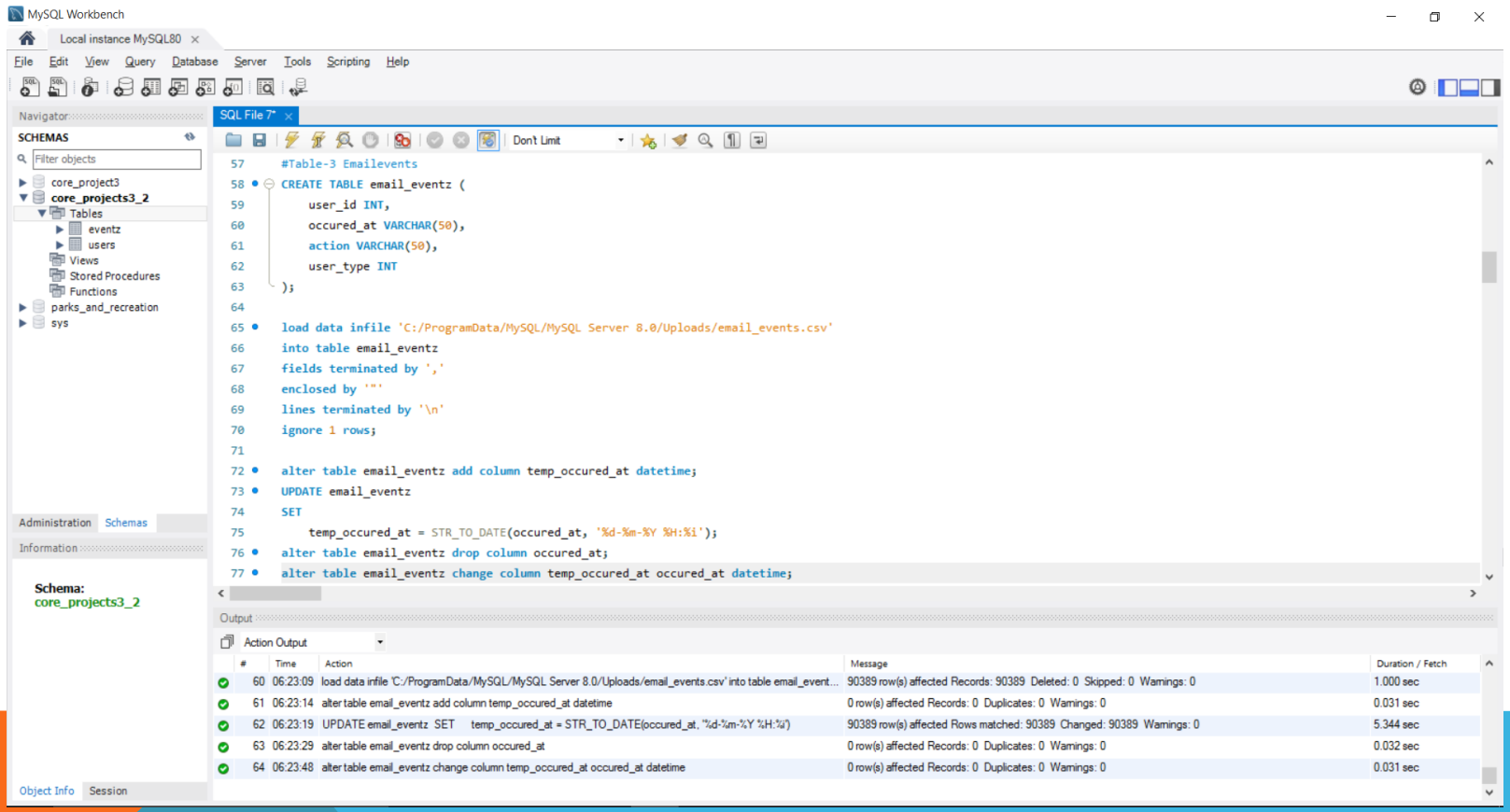


TABLE-3:

```
CREATE TABLE email_eventz
(
user_id     INT,
occured_at VARCHAR(50),
action      VARCHAR(50),
user_type   INT
);load data INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_eve
nts.csv INTO TABLE email_eventz fields terminated BY ',
ENCLOSED BY '"' lines terminated BY '\n'
IGNORE 1 rows;

ALTER TABLE email_eventz ADD COLUMN temp_occured_at DATETIME;UPDATE email_e
ventz
SET    temp_occured_at = Str_to_date(occured_at, '%d-%m-%Y %H:%i');

ALTER TABLE email_eventz DROP COLUMN occured_at;

ALTER TABLE email_eventz CHANGE COLUMN temp_occured_at occured_at DATETIME;
```

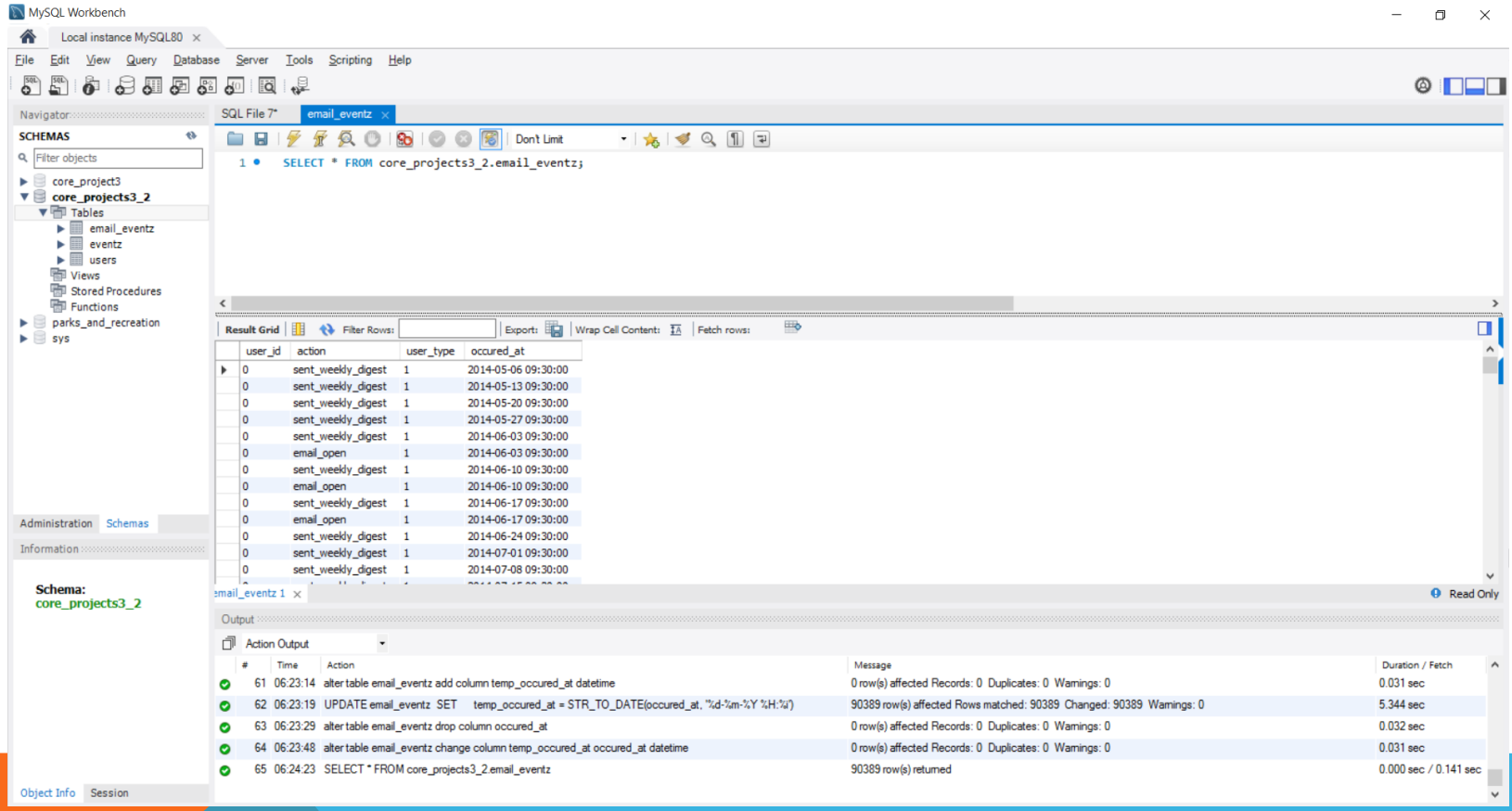
RESULT:

Query:



trainity

Output:



CASE STUDY 2: TASKS

trainity

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.

SQL Query:

```
SELECT
    EXTRACT(WEEK FROM occurred_at) AS week_num,
    COUNT(DISTINCT user_id) AS active_users
FROM
    eventz
WHERE
    event_type = 'engagement'
GROUP BY week_num
ORDER BY week_num;
```

Insight:

- The Highest User Week: 30th Week, with 1467 users.
- The Minimum User Week: 35th Week, with 104 users.

RESULT:

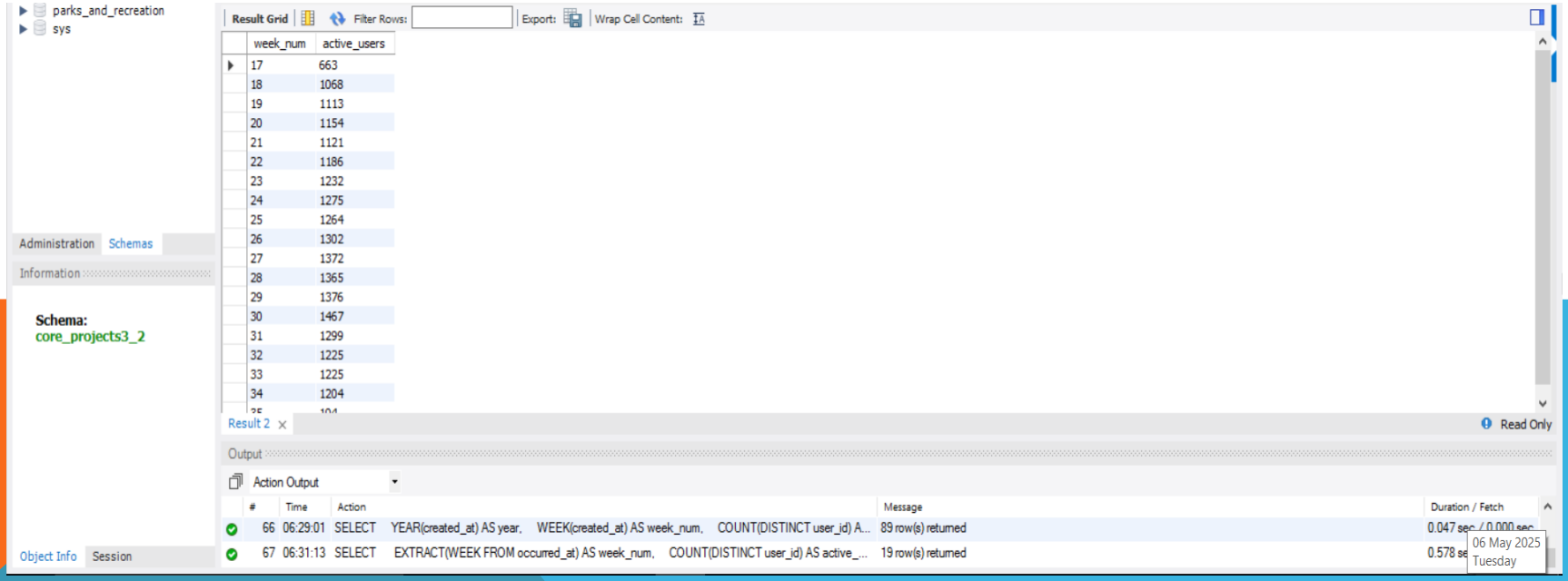
trainity

week_num	active_users
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

RESULT:

Query:

Output:



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.

SQL Query:

```
SELECT
    YEAR(created_at) AS year,
    WEEK(created_at) AS week_num,
    COUNT(DISTINCT user_id) AS weekly_users,
    SUM(COUNT(DISTINCT user_id)) OVER (ORDER BY
YEAR(created_at), WEEK(created_at)) AS cumulative_users
FROM
    users
GROUP BY
    YEAR(created_at),
    WEEK(created_at)
ORDER BY
    year,
    week_num;
```

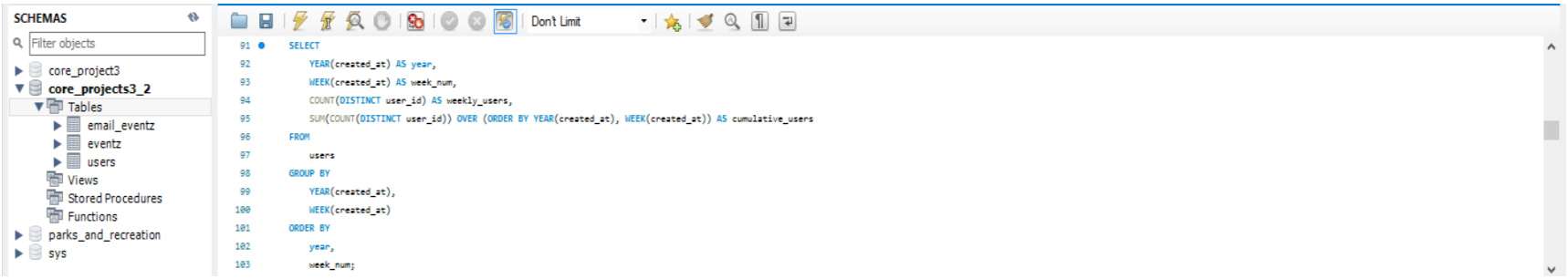
RESULT:

trainity

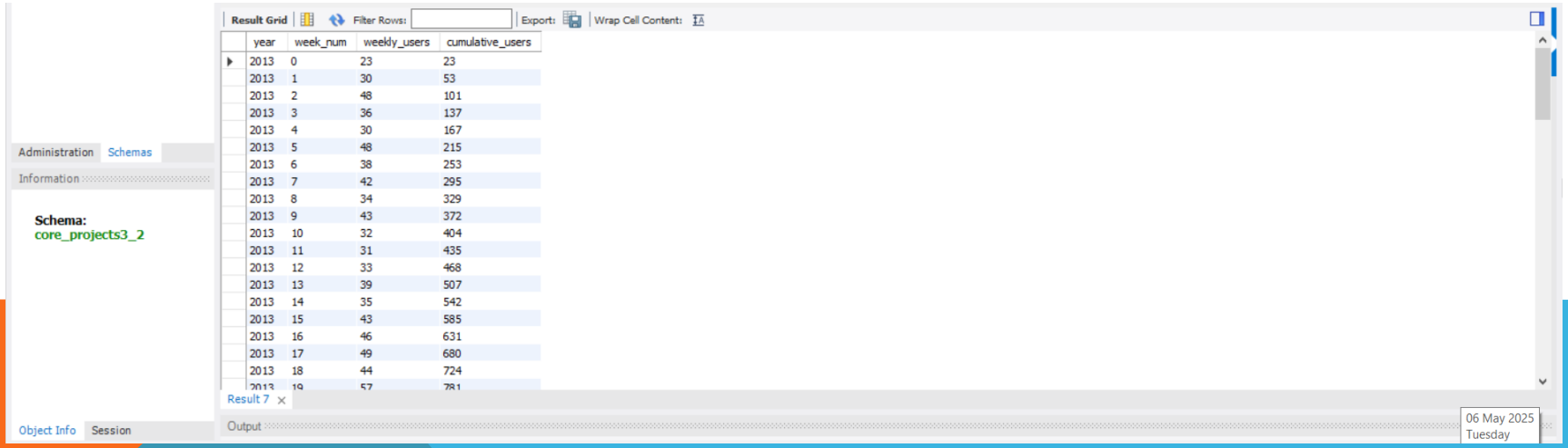
year	week_num	weekly_users	cumulative_users
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724
2013	19	57	781

RESULT:

Query:



Output:



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.
- SQL Query:

```
SELECT
    week_numbs,
    SUM(CASE WHEN week_diff = 0 THEN 1 ELSE 0 END) AS week_0,
    SUM(CASE WHEN week_diff = 1 THEN 1 ELSE 0 END) AS week_1,
    SUM(CASE WHEN week_diff = 2 THEN 1 ELSE 0 END) AS week_2,
    SUM(CASE WHEN week_diff = 3 THEN 1 ELSE 0 END) AS week_3,
    SUM(CASE WHEN week_diff = 4 THEN 1 ELSE 0 END) AS week_4,
    SUM(CASE WHEN week_diff = 5 THEN 1 ELSE 0 END) AS week_5,
    SUM(CASE WHEN week_diff = 6 THEN 1 ELSE 0 END) AS week_6,
    SUM(CASE WHEN week_diff = 7 THEN 1 ELSE 0 END) AS week_7,
    SUM(CASE WHEN week_diff = 8 THEN 1 ELSE 0 END) AS week_8,
    SUM(CASE WHEN week_diff = 9 THEN 1 ELSE 0 END) AS week_9,
    SUM(CASE WHEN week_diff = 10 THEN 1 ELSE 0 END) AS week_10,
    SUM(CASE WHEN week_diff = 11 THEN 1 ELSE 0 END) AS week_11,
    SUM(CASE WHEN week_diff = 12 THEN 1 ELSE 0 END) AS week_12,
    SUM(CASE WHEN week_diff = 13 THEN 1 ELSE 0 END) AS week_13,
    SUM(CASE WHEN week_diff = 14 THEN 1 ELSE 0 END) AS week_14,
    SUM(CASE WHEN week_diff = 15 THEN 1 ELSE 0 END) AS week_15,
    SUM(CASE WHEN week_diff = 16 THEN 1 ELSE 0 END) AS week_16,
    SUM(CASE WHEN week_diff = 17 THEN 1 ELSE 0 END) AS week_17,
    SUM(CASE WHEN week_diff = 18 THEN 1 ELSE 0 END) AS week_18
FROM (
    SELECT
        user_id,
        EXTRACT(WEEK FROM occurred_at) AS login_week,
        MIN(EXTRACT(WEEK FROM occurred_at)) OVER (PARTITION BY user_id) AS week_numbs,
        (EXTRACT(WEEK FROM occurred_at) - MIN(EXTRACT(WEEK FROM occurred_at)) OVER (PARTITION BY user_id)) AS week_diff
    FROM
        eventz
) AS sub
GROUP BY
    week_numbs
ORDER BY
    week_numbs;SELECT
    YEAR(created_at) AS year,
    WEEK(created_at) AS week_num,
    COUNT(DISTINCT user_id) AS weekly_users,
    SUM(COUNT(DISTINCT user_id)) OVER (ORDER BY YEAR(created_at), WEEK(created_at)) AS cumulative_users
FROM
    users
GROUP BY
    YEAR(created_at),
    WEEK(created_at)
ORDER BY
    year,
    week_num;
```

Insights:

- The data shown in the table is from the year 2014 and includes data for 35 weeks.
- The users who joined in the 17th week show the longest period of user retention (18 weeks).
- The 17th week was the largest user sign-up week, with retention up to 18 weeks for 63 users.
- The lowest number of users joined in the 35th week, with 145 users.
- We can come to a conclusion that there is a flaw relating to customer retention maybe the software, website, product, etc. Which made customers retention rate crash and increased dissatisfaction.

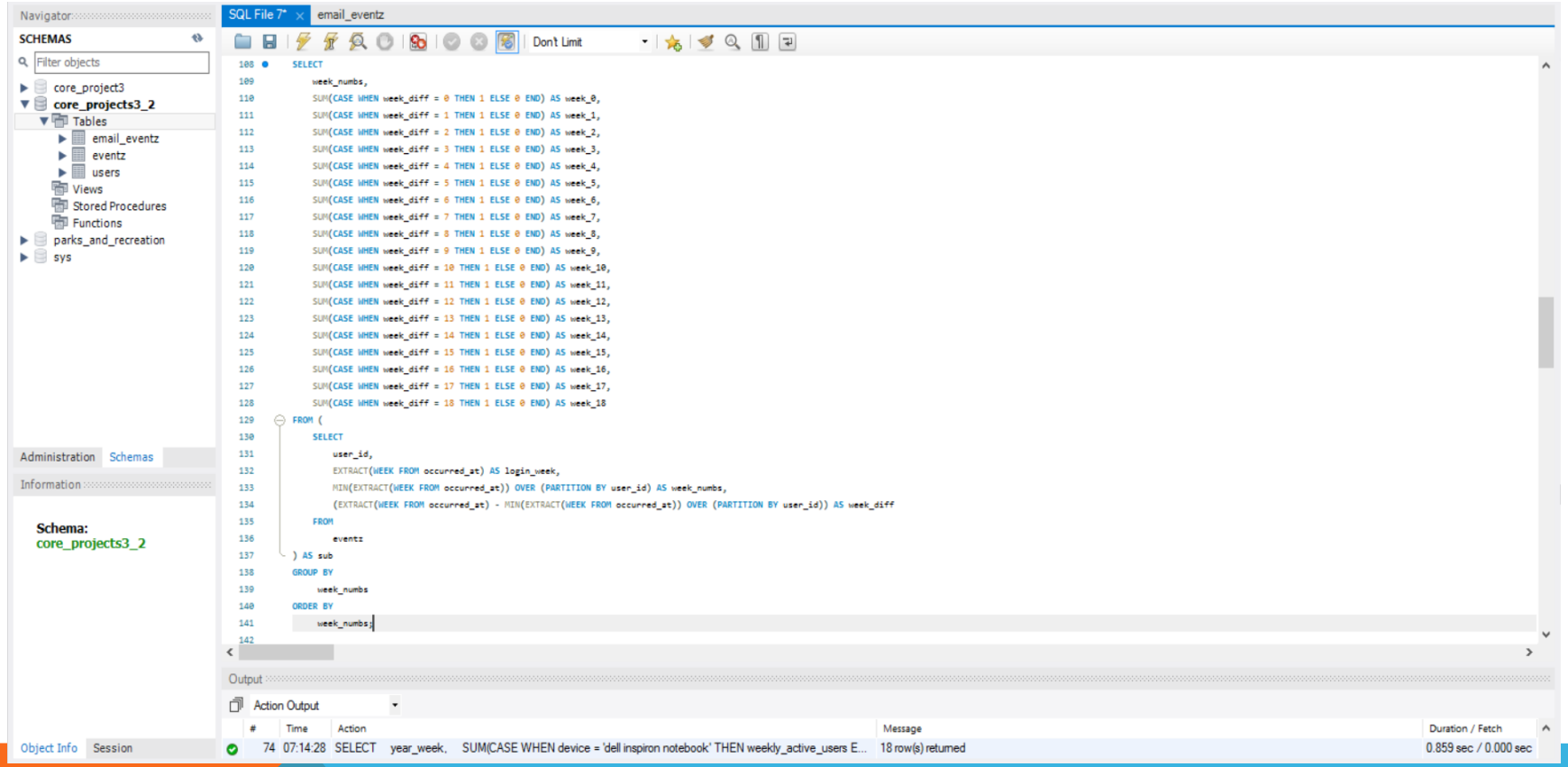
RESULT:

trainity

week_numbs	week_0	week_1	week_2	week_3	week_4	week_5	week_6	week_7	week_8	week_9	week_10	week_11	week_12	week_13	week_14	week_15	week_16	week_17	week_18
17	8091	9291	6100	4890	3910	3376	2900	2782	2692	2743	2390	2395	2391	2574	2044	1464	1474	1527	40
18	8213	5952	4598	3391	2719	2223	2329	2008	1861	1823	1836	1916	2093	1954	1400	1412	1135	37	0
19	5357	4326	2756	2303	1905	1658	1355	1484	1409	1483	1158	917	826	669	785	805	23	0	0
20	4273	3340	2840	1884	1413	1252	953	835	1029	989	876	549	441	463	450	0	0	0	0
21	3937	3141	2213	1342	1145	963	955	1275	822	804	616	567	555	441	17	0	0	0	0
22	4230	3378	2506	1642	1263	942	1001	901	851	694	791	679	467	8	0	0	0	0	0
23	3973	3369	2064	1522	1143	1041	1065	824	799	674	412	410	0	0	0	0	0	0	0
24	3882	3252	2067	1613	1289	911	1143	913	678	430	384	0	0	0	0	0	0	0	0
25	3439	3087	2280	1639	1212	816	577	723	536	537	23	0	0	0	0	0	0	0	0
26	3319	3029	1552	1296	933	714	662	548	345	0	0	0	0	0	0	0	0	0	0
27	3684	3195	1693	1553	1057	806	558	560	12	0	0	0	0	0	0	0	0	0	0
28	3256	3026	1854	873	564	388	323	20	0	0	0	0	0	0	0	0	0	0	0
29	2908	3111	1688	966	619	639	5	0	0	0	0	0	0	0	0	0	0	0	0
30	3422	2995	1727	1124	696	14	0	0	0	0	0	0	0	0	0	0	0	0	0
31	2450	2112	906	777	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	2613	2304	1085	67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	3213	2727	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	3078	326	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	145	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

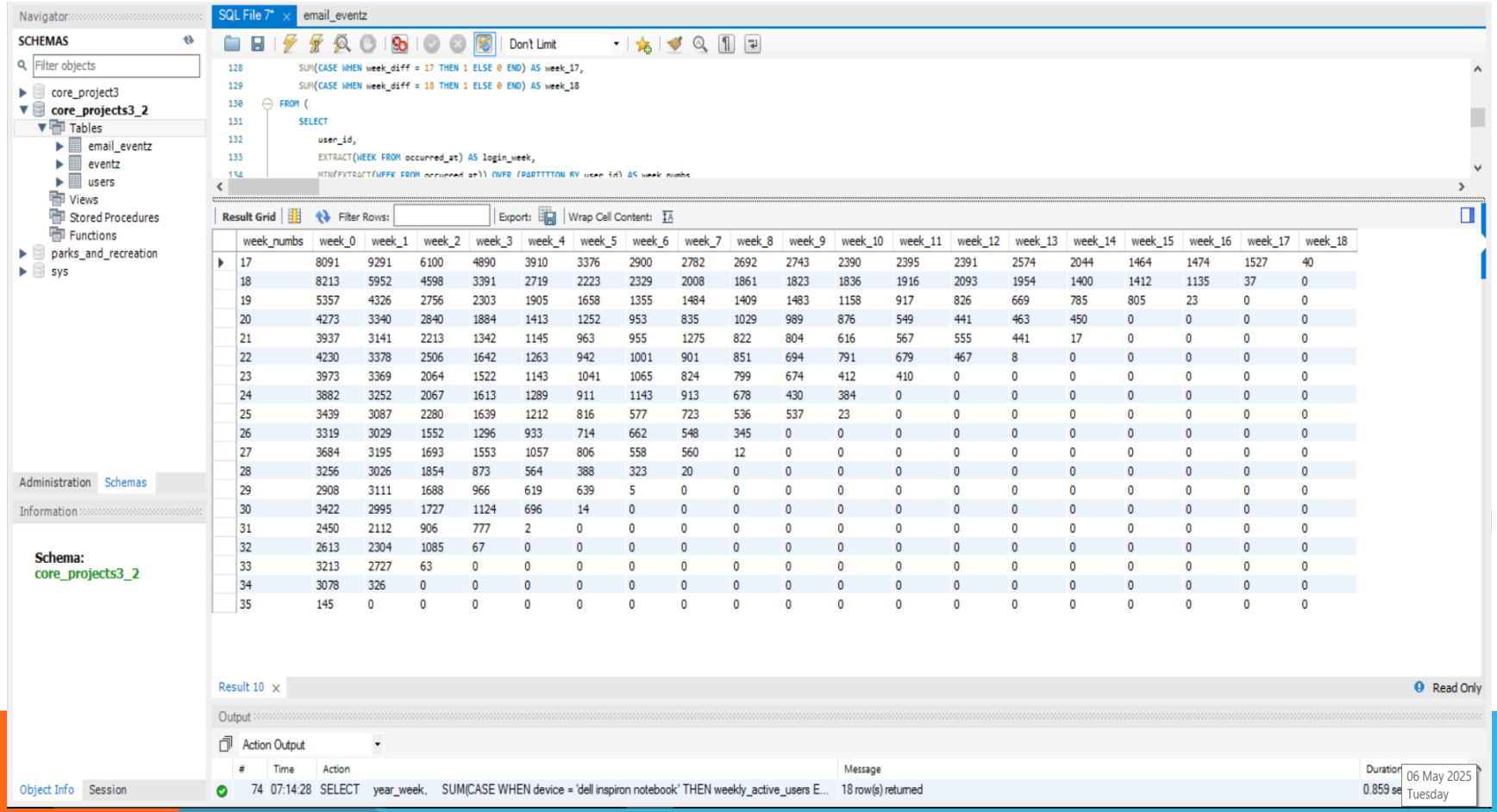
RESULT:

Query:



RESULT:

Output:



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.

SQL Query:

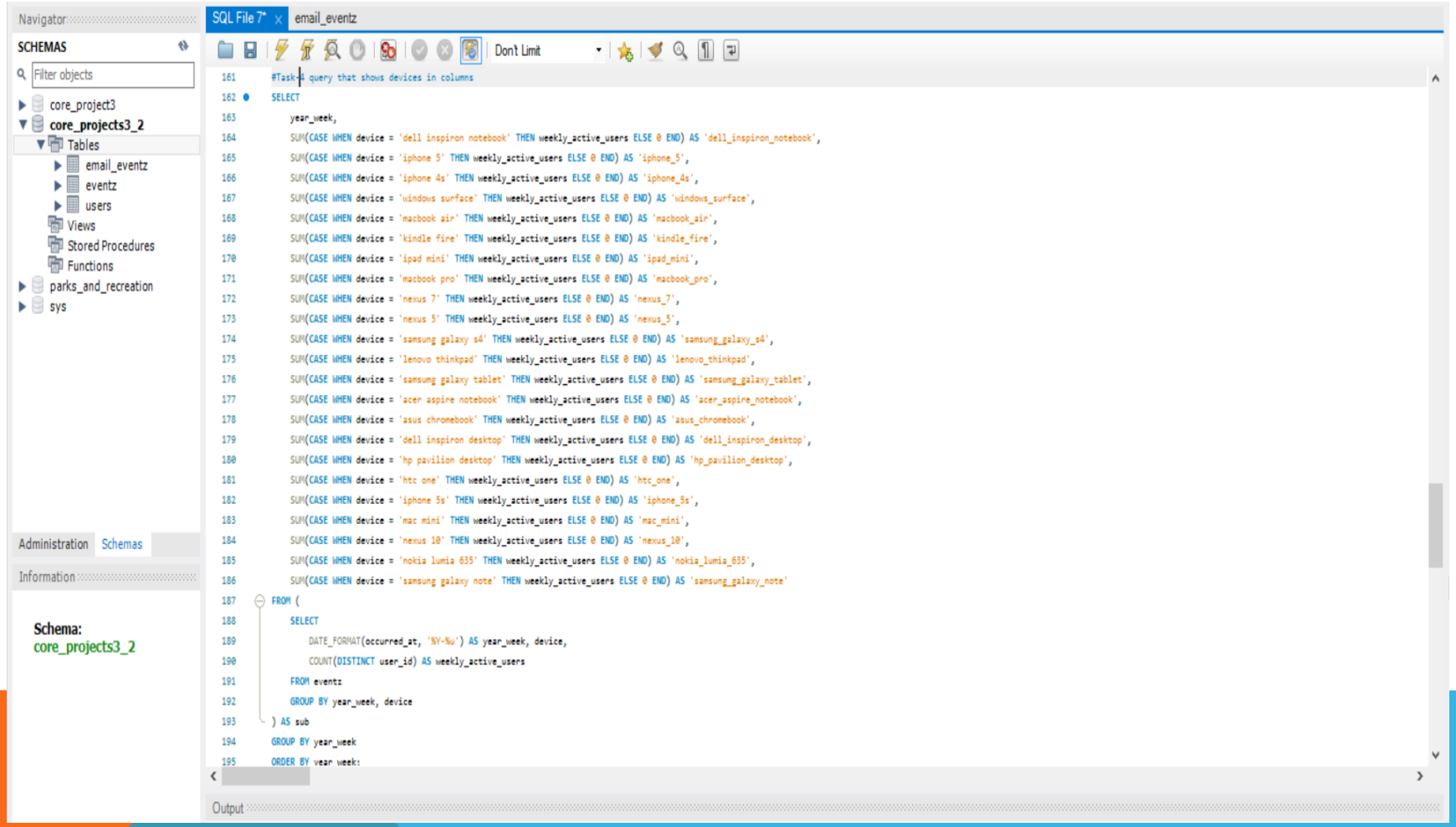
```
SELECT
    year_week,
    SUM(CASE WHEN device = 'dell inspiron notebook' THEN weekly_active_users ELSE 0 END) AS 'dell_inspiron_notebook',
    SUM(CASE WHEN device = 'iphone 5' THEN weekly_active_users ELSE 0 END) AS 'iphone_5',
    SUM(CASE WHEN device = 'iphone 4s' THEN weekly_active_users ELSE 0 END) AS 'iphone_4s',
    SUM(CASE WHEN device = 'windows surface' THEN weekly_active_users ELSE 0 END) AS 'windows_surface',
    SUM(CASE WHEN device = 'macbook air' THEN weekly_active_users ELSE 0 END) AS 'macbook_air',
    SUM(CASE WHEN device = 'kindle fire' THEN weekly_active_users ELSE 0 END) AS 'kindle_fire',
    SUM(CASE WHEN device = 'ipad mini' THEN weekly_active_users ELSE 0 END) AS 'ipad_mini',
    SUM(CASE WHEN device = 'macbook pro' THEN weekly_active_users ELSE 0 END) AS 'macbook_pro',
    SUM(CASE WHEN device = 'nexus 7' THEN weekly_active_users ELSE 0 END) AS 'nexus_7',
    SUM(CASE WHEN device = 'nexus 5' THEN weekly_active_users ELSE 0 END) AS 'nexus_5',
    SUM(CASE WHEN device = 'samsung galaxy s4' THEN weekly_active_users ELSE 0 END) AS 'samsung_galaxy_s4',
    SUM(CASE WHEN device = 'lenovo thinkpad' THEN weekly_active_users ELSE 0 END) AS 'lenovo_thinkpad',
    SUM(CASE WHEN device = 'samsumg galaxy tablet' THEN weekly_active_users ELSE 0 END) AS 'samsumg_galaxy_tablet',
    SUM(CASE WHEN device = 'acer aspire notebook' THEN weekly_active_users ELSE 0 END) AS 'acer_aspire_notebook',
    SUM(CASE WHEN device = 'asus chromebook' THEN weekly_active_users ELSE 0 END) AS 'asus_chromebook',
    SUM(CASE WHEN device = 'dell inspiron desktop' THEN weekly_active_users ELSE 0 END) AS 'dell_inspiron_desktop',
    SUM(CASE WHEN device = 'hp pavilion desktop' THEN weekly_active_users ELSE 0 END) AS 'hp_pavilion_desktop',
    SUM(CASE WHEN device = 'htc one' THEN weekly_active_users ELSE 0 END) AS 'htc_one',
    SUM(CASE WHEN device = 'iphone 5s' THEN weekly_active_users ELSE 0 END) AS 'iphone_5s',
    SUM(CASE WHEN device = 'mac mini' THEN weekly_active_users ELSE 0 END) AS 'mac_mini',
    SUM(CASE WHEN device = 'nexus 10' THEN weekly_active_users ELSE 0 END) AS 'nexus_10',
    SUM(CASE WHEN device = 'nokia lumia 635' THEN weekly_active_users ELSE 0 END) AS 'nokia_lumia_635',
    SUM(CASE WHEN device = 'samsung galaxy note' THEN weekly_active_users ELSE 0 END) AS 'samsung_galaxy_note'
FROM (SELECT
        DATE_FORMAT(occurred_at, '%Y-%u') AS year_week, device,
        COUNT(DISTINCT user_id) AS weekly_active_users
    FROM eventz
    GROUP BY year_week, device) AS sub GROUP BY year_week ORDER BY year_week;
```

Insights:

- Most people used a Mac Pro (322 users in the 19th week).
- Lenovo Thinkpad was the next most popular (220 users in the 28th week).
- iPhone 5 was used by 163 people in the 27th week.
- Samsung Galaxy Note had the least number of users that is 299 in total.
- Kindle fire started off with low users but show a considerable growth compared to Samsung Galaxy Note.
- From this we can understand something interesting that focusing on delivering a seamless experience would increase the trust of customers while retaining and show growth.

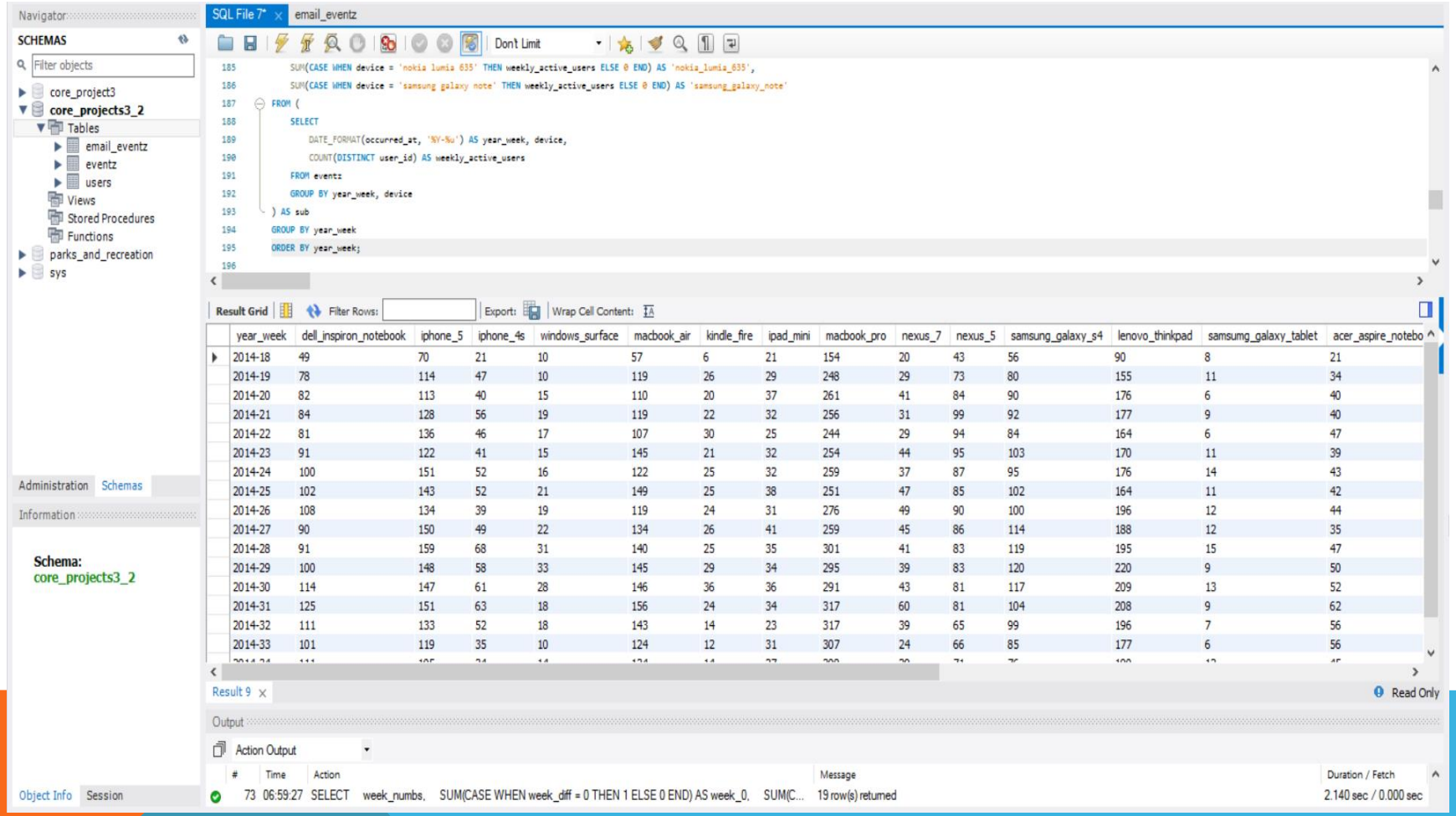
RESULT:

Query:



RESULT:

Output:



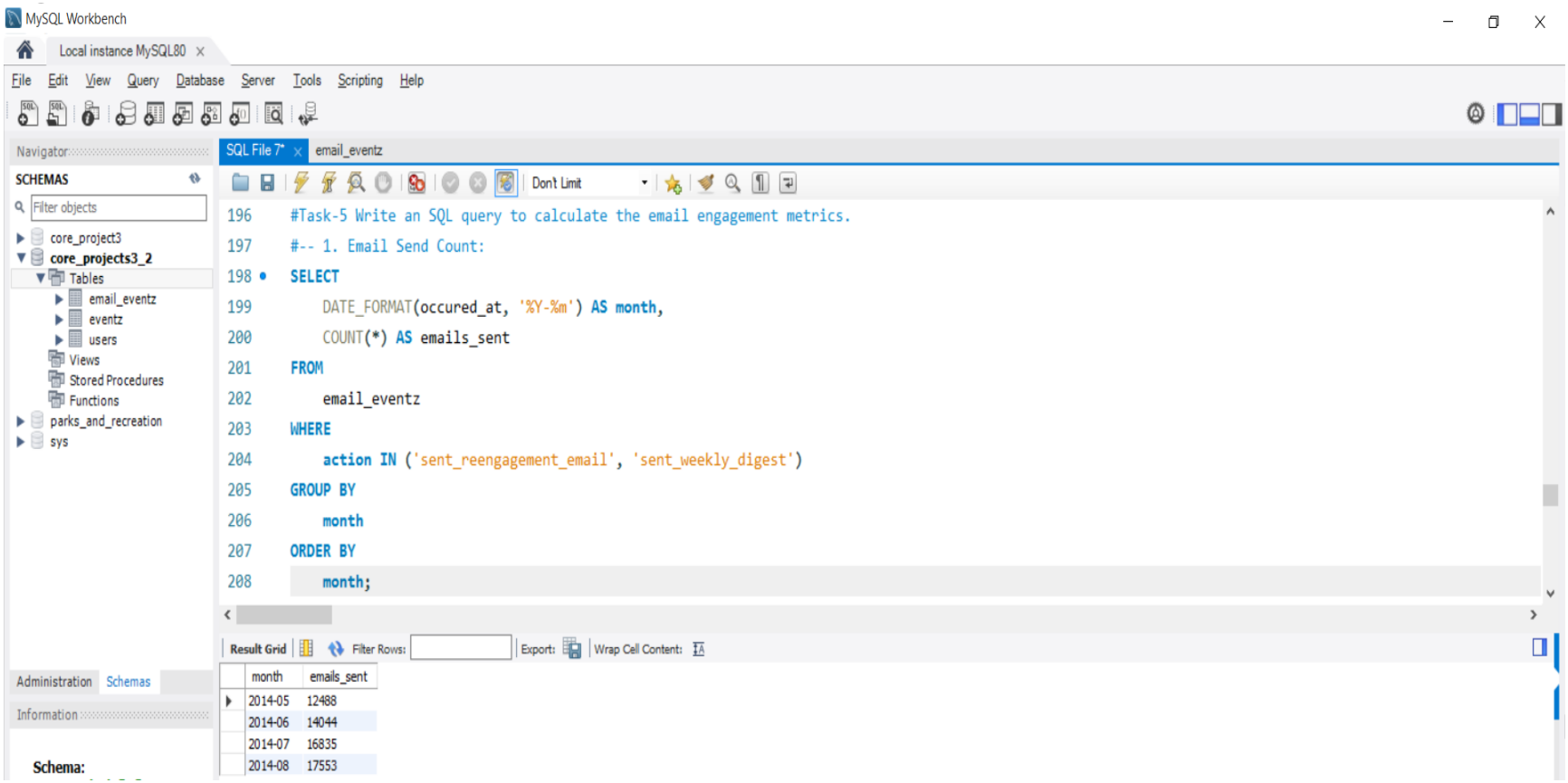
E. Email Engagement Analysis:

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

SQL Query:

```
#-- 1. Email Send Count:
SELECT
    DATE_FORMAT(occurred_at, '%Y-%m') AS month,
    COUNT(*) AS emails_sent
FROM
    email_eventz
WHERE
    action IN ('sent_reengagement_email', 'sent_weekly_digest')
GROUP BY
    month
ORDER BY
    month;
```

RESULT:

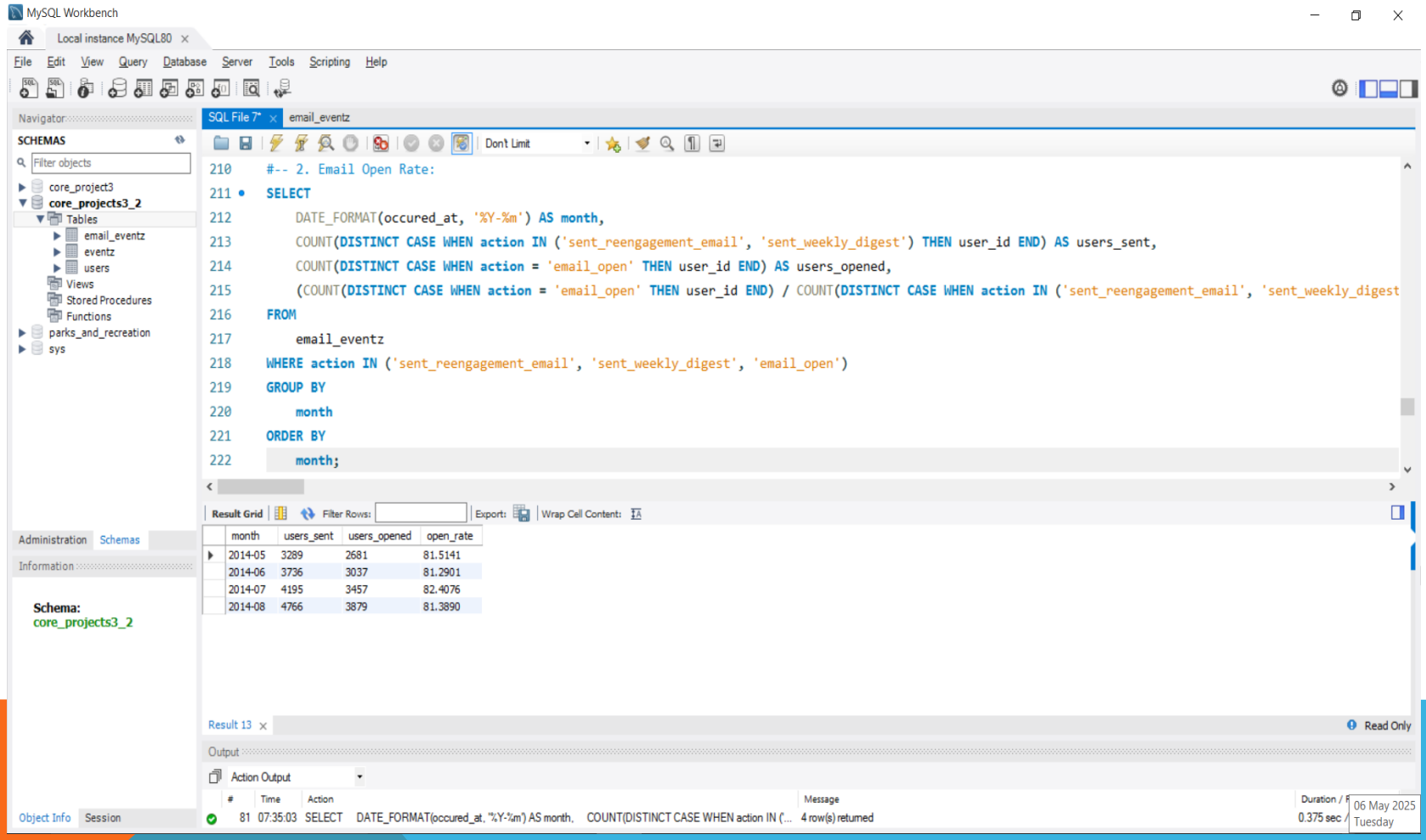

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
core_project3
core_projects3_2
Tables
email_eventz
eventz
users
Views
Stored Procedures
Functions
parks_and_recreation
sys
Administration Schemas
Information
Schema:
SQL File 7*
email_eventz
Don't Limit
196 #Task-5 Write an SQL query to calculate the email engagement metrics.
197 #-- 1. Email Send Count:
198 SELECT
199 DATE_FORMAT(occured_at, '%Y-%m') AS month,
200 COUNT(*) AS emails_sent
201 FROM
202 email_eventz
203 WHERE
204 action IN ('sent_reengagement_email', 'sent_weekly_digest')
205 GROUP BY
206 month
207 ORDER BY
208 month;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
month emails_sent
2014-05 12488
2014-06 14044
2014-07 16835
2014-08 17553

SQL Query:

```
#-- 2. Email Open Rate:
SELECT
    DATE_FORMAT(occurred_at, '%Y-%m') AS month,
    COUNT(DISTINCT CASE WHEN action IN ('sent_reengagement_email',
'sent_weekly_digest') THEN user_id END) AS users_sent,
    COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) AS
users_opened,
    (COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) /
COUNT(DISTINCT CASE WHEN action IN ('sent_reengagement_email',
'sent_weekly_digest') THEN user_id END)) * 100 AS open_rate
FROM
    email_eventz
WHERE action IN ('sent_reengagement_email', 'sent_weekly_digest', 'email_open')
GROUP BY
    month
ORDER BY
    month;
```

RESULT:

Output:



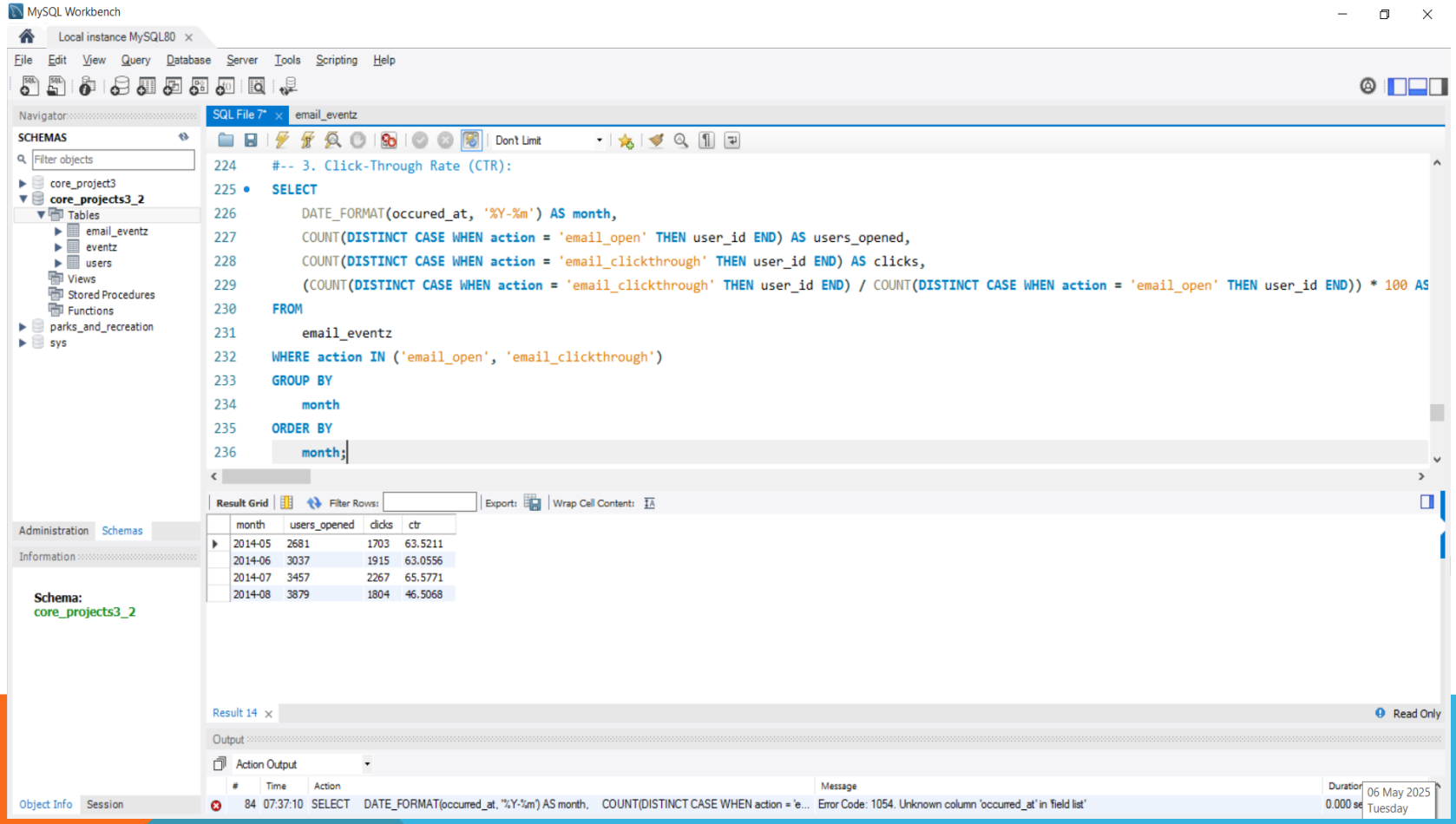
SQL Query:

```
#-- 3. Click-Through Rate (CTR):

SELECT

    DATE_FORMAT(occured_at, '%Y-%m') AS month,

    COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) AS
users_opened,

    COUNT(DISTINCT CASE WHEN action = 'email_clickthrough' THEN user_id END)
AS clicks,

    (COUNT(DISTINCT CASE WHEN action = 'email_clickthrough' THEN user_id END)
/ COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END)) * 100 AS
ctr

FROM

    email_eventz

WHERE action IN ('email_open', 'email_clickthrough')

GROUP BY

    month

ORDER BY

    month;
```

trainity

RESULT:

Output:

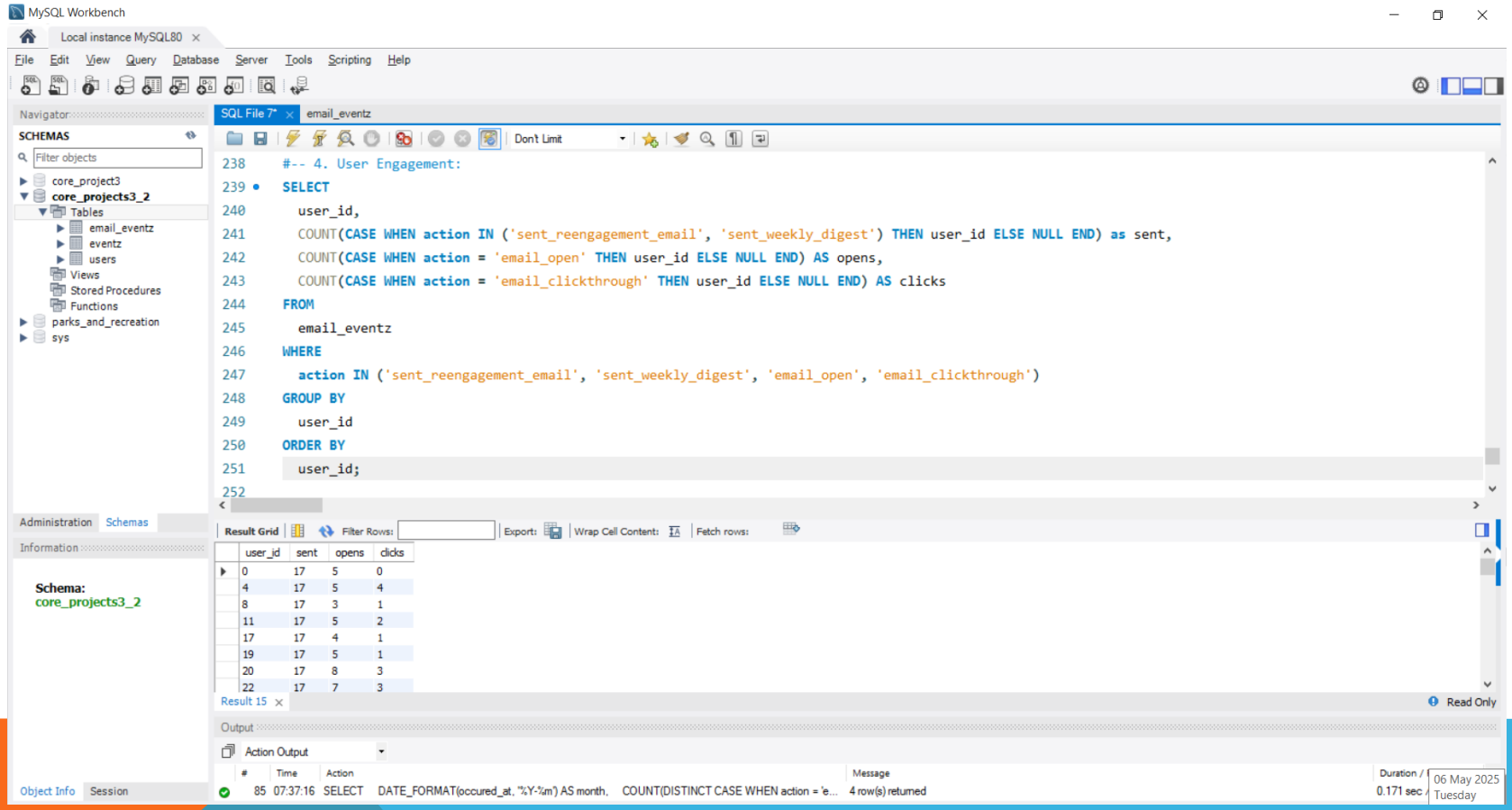


SQL Query:

```
#-- 4. User Engagement:

SELECT

  user_id,

  COUNT(CASE WHEN action IN ('sent_reengagement_email', 'sent_weekly_digest')
THEN user_id ELSE NULL END) as sent,

  COUNT(CASE WHEN action = 'email_open' THEN user_id ELSE NULL END) AS
opens,

  COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id ELSE NULL END)
AS clicks

FROM

  email_eventz

WHERE

  action IN ('sent_reengagement_email', 'sent_weekly_digest', 'email_open',
'email_clickthrough')

GROUP BY

  user_id

ORDER BY

  user_id;
```

RESULT:

Output:

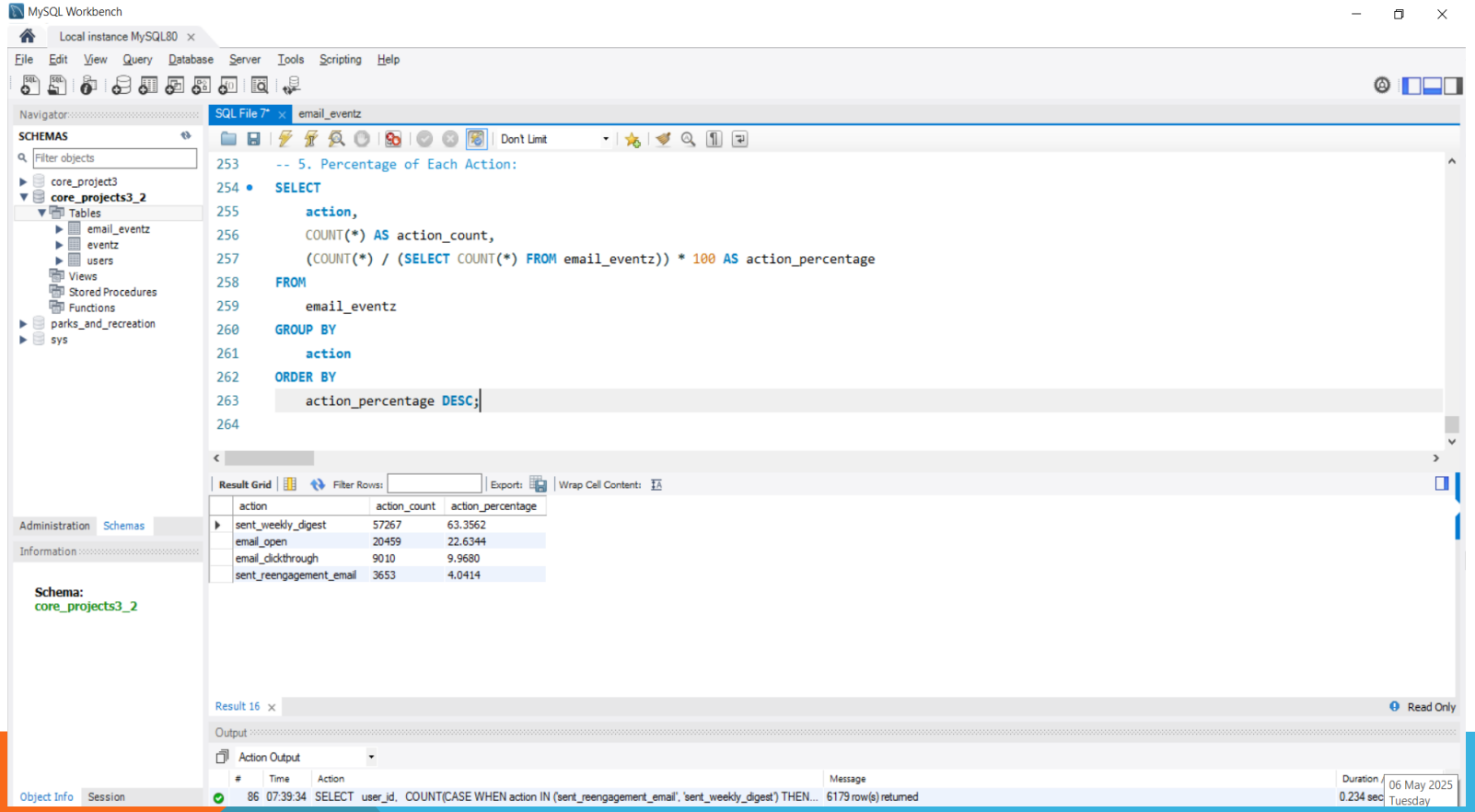


SQL Query:

```
#-- 5. Percentage of Each Action:
SELECT
    action,
    COUNT(*) AS action_count,
    (COUNT(*) / (SELECT COUNT(*) FROM email_eventz)) * 100 AS action_percentage
FROM
    email_eventz
GROUP BY
    action
ORDER BY
    action_percentage DESC;
```

RESULT:

Output:



THANK YOU