Name: Naman Rawal

Email: rawalnaman49@gmail.com

Operation Analytics and Investigating Metric Spike

Project Title and Overview:

- ### Project Title:

Analysis of Job Data and User Engagement Metrics

- ### Project Overview:

This project aimed to analyze job review data and user engagement metrics to extract actionable insights for better decision-making. By applying various data analysis techniques, we examined trends, user behavior, and platform performance.

Project Description:

- ### Project Purpose:

The goal of this project was to analyze job review data for November 2020, identify trends in user engagement, and evaluate key metrics related to job data and platform performance. The project focused on understanding job review patterns, calculating throughput, analyzing language distribution, and investigating user behavior and retention.

- **Analysis Plan:**

1. Analyze job review trends over time.
2. Calculate rolling averages for throughput.
3. Assess language usage in job reviews.
4. Detect and remove duplicate records from the dataset.
5. Investigate user engagement, growth, retention, and device usage.

Approach:

- **Approach to Analysis:**

Data Collection & Preparation:

- Extracted relevant data from the job_data and events tables.
- Filtered data to focus on the required timeframe and relevant columns.

Data Segmentation:

- Grouped data by time (hour, day, week) to analyze patterns over different intervals.
- Segmented user data by cohorts and device types for deeper insights.

Data Analysis:

- Used SQL queries to perform calculations and identify trends.
- Applied rolling averages and cohort analysis to smooth data and reveal patterns.
- Performed language share analysis and duplicate row detection.

Result Interpretation:

- Summarized findings into actionable insights.
- Visualized data trends to support decision-making.

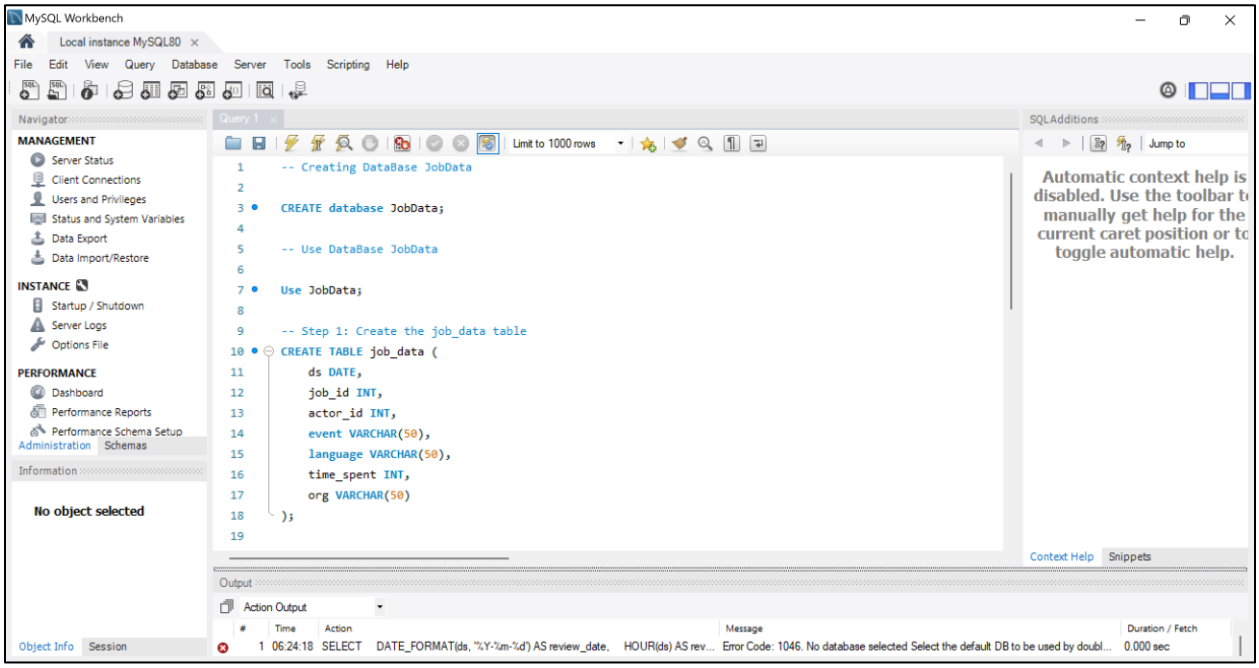
Tech-Stack Used:

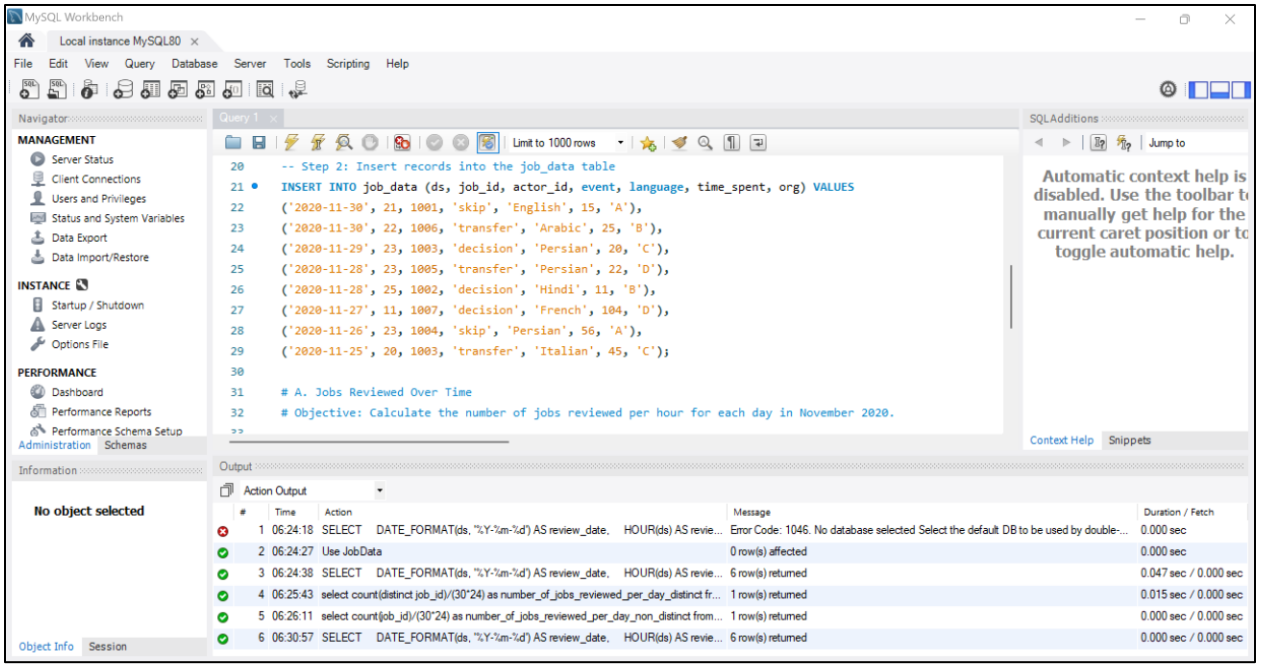
- **Software & Versions:**

MySQL Workbench 8.0 CE:

- **Purpose:** Used for executing SQL queries, managing databases, and performing data analysis.
- **Role in Analysis:** Facilitated data extraction, filtering, aggregation, and visualization. Enabled the efficient computation of metrics like throughput, user retention, and growth.

Case Study 1: Job Data





A. Jobs Reviewed Over Time:

- **Objective:** Calculate the number of jobs reviewed per hour for each day in November 2020.

Approach:

1. **Data Selection**: Utilize the job_id column from the job_data table.
2. **Time Segmentation**: Filter the data for November 2020.
3. **Calculation**: Count the number of job reviews per hour using the COUNT function and group the results by each day.

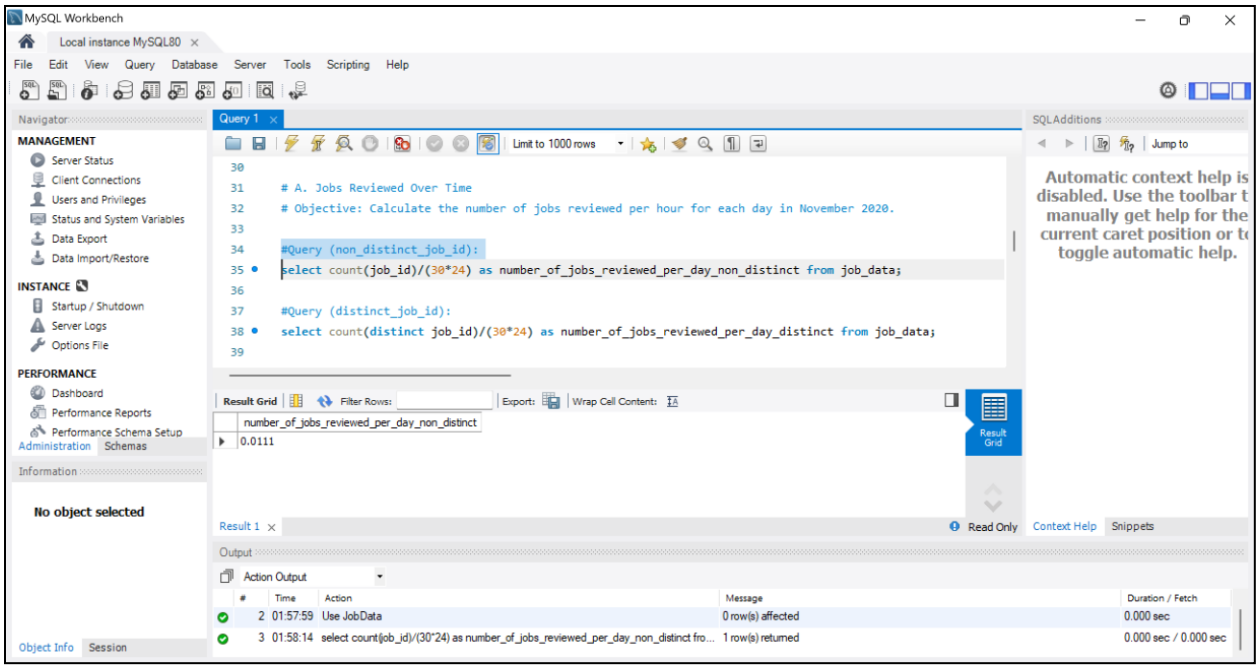
Query:

Query (non_distinct_job_id):

```
select count(job_id)/(30*24) as number_of_jobs_reviewed_per_day_non_distinct
from job_data;
```

Result:

number_of_jobs_reviewed_per_day_non_distinct
0.0111

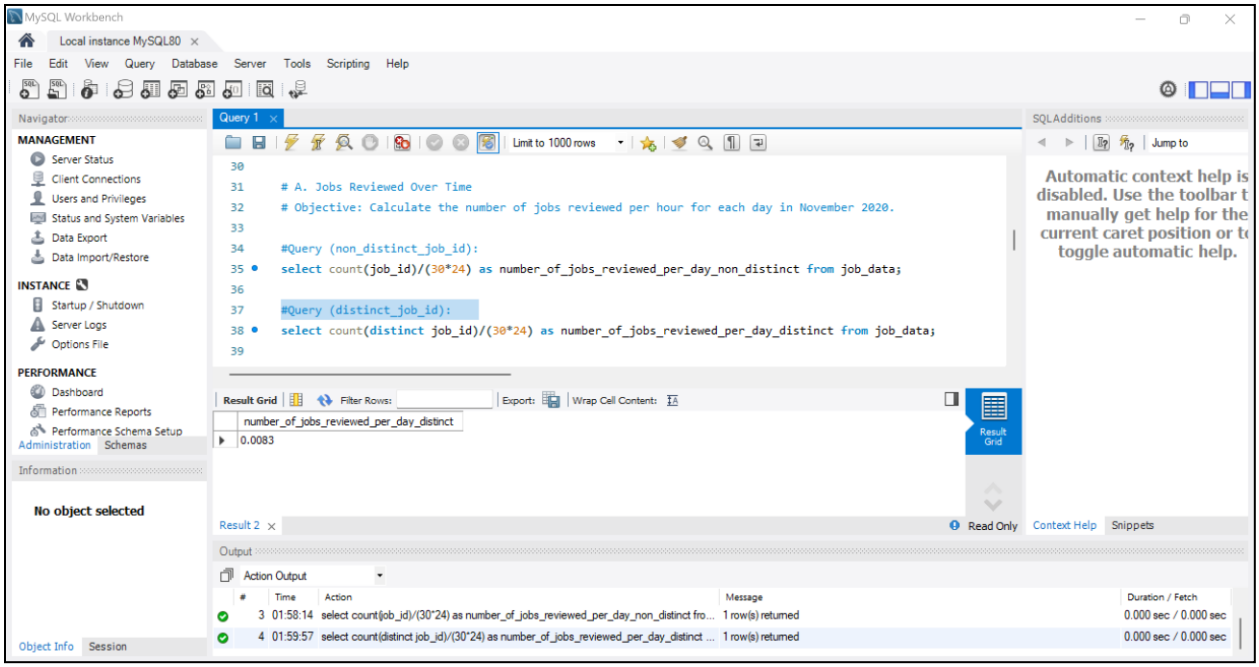


Query (distinct_job_id):

select count(distinct job_id)/(30*24) as
number_of_jobs_reviewed_per_day_distinct from job_data;

Result:

number_of_jobs_reviewed_per_day_distinct
0.0083



Insight: This query calculates the total number of jobs reviewed per hour each day in November 2020, providing a detailed view of review activity across the month.

B. Throughput Analysis:

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

- **Approach**:

1. **Metric Selection:** Throughput is calculated as the count of events occurring per second.
2. **Rolling Average**: Implement a 7-day rolling average using window functions.
3. **Preference**: The 7-day rolling average is preferred over daily metrics for smoother trend analysis and to avoid daily fluctuations.

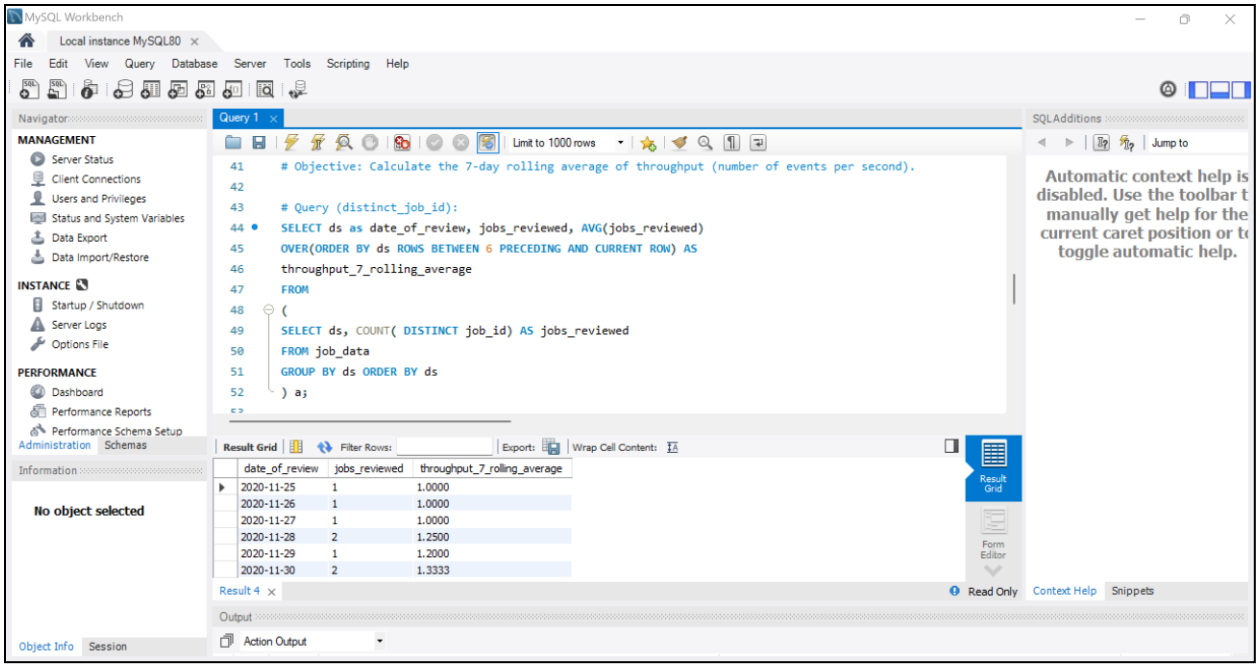
Query:

Query (distinct_job_id):

```
SELECT ds as date_of_review, jobs_reviewed, AVG(jobs_reviewed)

OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT
ROW) AS

throughput_7_rolling_average

FROM

(

SELECT ds, COUNT( DISTINCT job_id) AS jobs_reviewed

FROM job_data

GROUP BY ds ORDER BY ds

) a;
```

Result:

date_of_review	jobs_reviewed	throughput_7_rolling_average
25-11-2020	1	1
26-11-2020	1	1
27-11-2020	1	1
28-11-2020	2	1.25
29-11-2020	1	1.2
30-11-2020	2	1.3333



Query (non_distinct_job_id):

SELECT ds as date_of_review, jobs_reviewed, AVG(jobs_reviewed)

OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS

throughput_7_rolling_average_non_distinct_job_id

FROM

(

SELECT ds, COUNT(job_id) AS jobs_reviewed

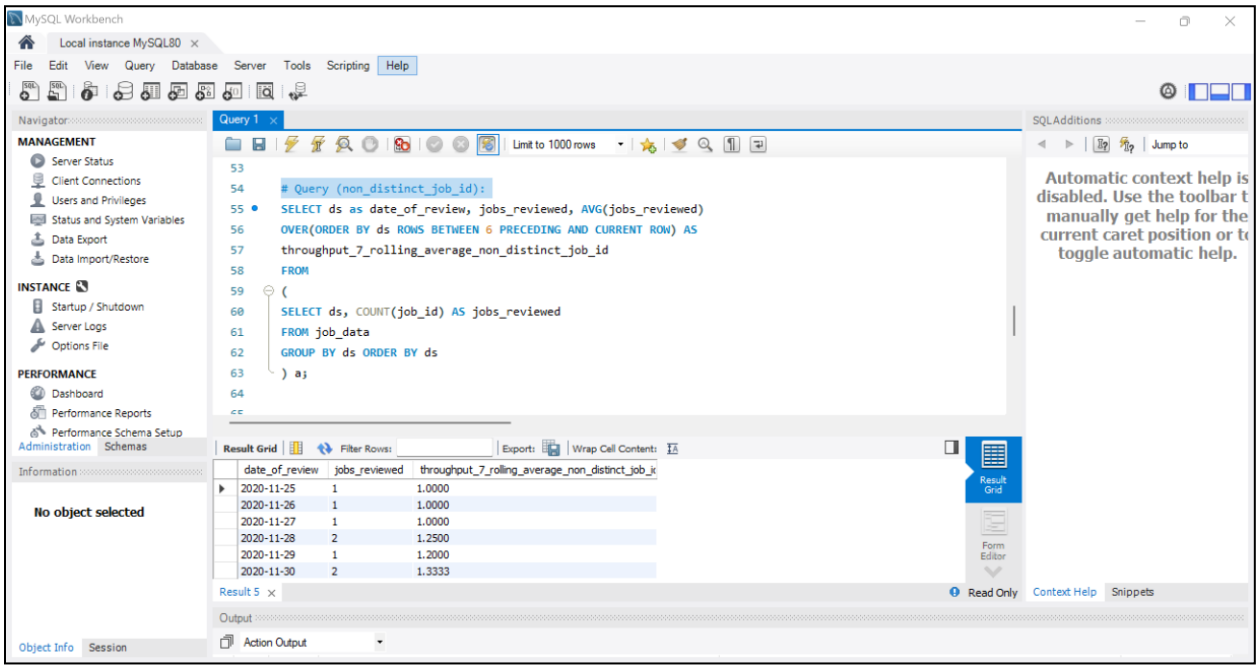
FROM job_data

GROUP BY ds ORDER BY ds

) a;

Result:

date_of_review	jobs_reviewed	throughput_7_rolling_average_non_distinct_job_id
25-11-2020	1	1
26-11-2020	1	1
27-11-2020	1	1
28-11-2020	2	1.25
29-11-2020	1	1.2
30-11-2020	2	1.3333



Insight: The 7-day rolling average helps in understanding long-term trends and smoothens out day-to-day variations, offering a more stable metric for analysis.

C. Language Share Analysis

Objective: Determine the percentage share of each language used in job reviews over the last 30 days.

Approach:

1. **Data Segmentation**: Filter data for the last 30 days.
2. **Percentage Calculation**: Calculate the count of reviews in each language and express it as a percentage of the total reviews.

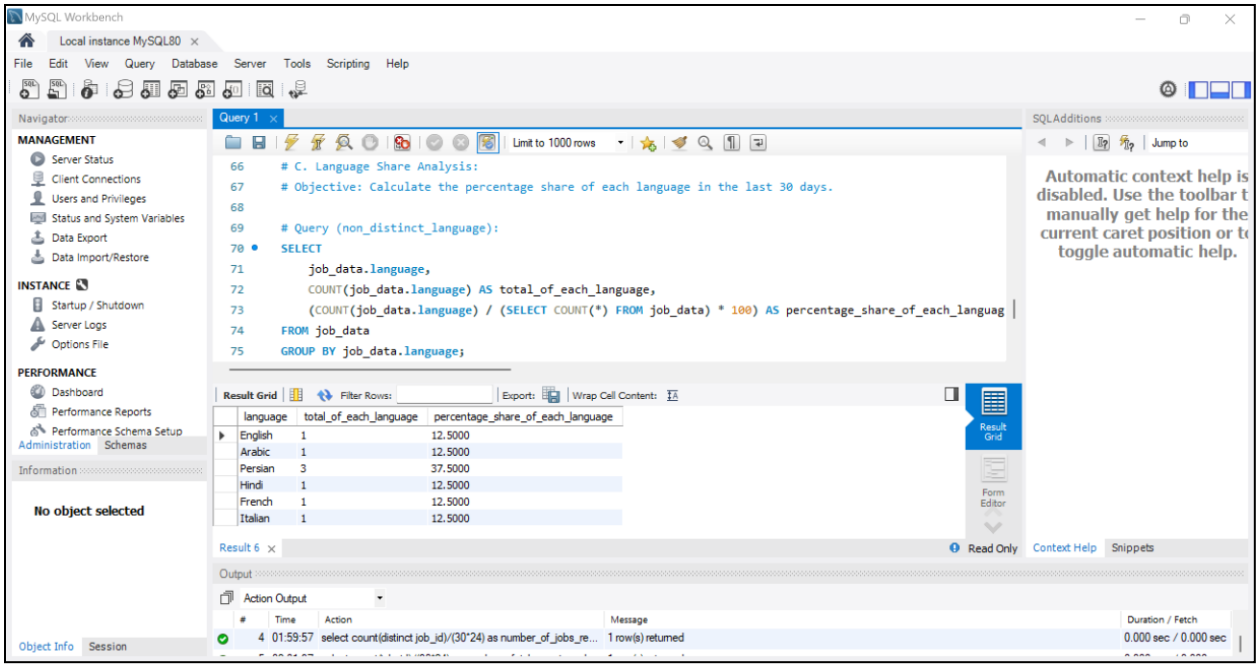
Query:

Query (non_distinct_language):

```
SELECT

    job_data.language,

    COUNT(job_data.language) AS total_of_each_language,

    (COUNT(job_data.language) / (SELECT COUNT(*) FROM job_data) * 100) AS
percentage_share_of_each_language

FROM job_data

GROUP BY job_data.language;
```

Result:

language	total_of_each_language	percentage_share_of_each_language
English	1	12.5
Arabic	1	12.5
Persian	3	37.5
Hindi	1	12.5
French	1	12.5
Italian	1	12.5



Query (distinct_language):

SELECT

job_data.language,

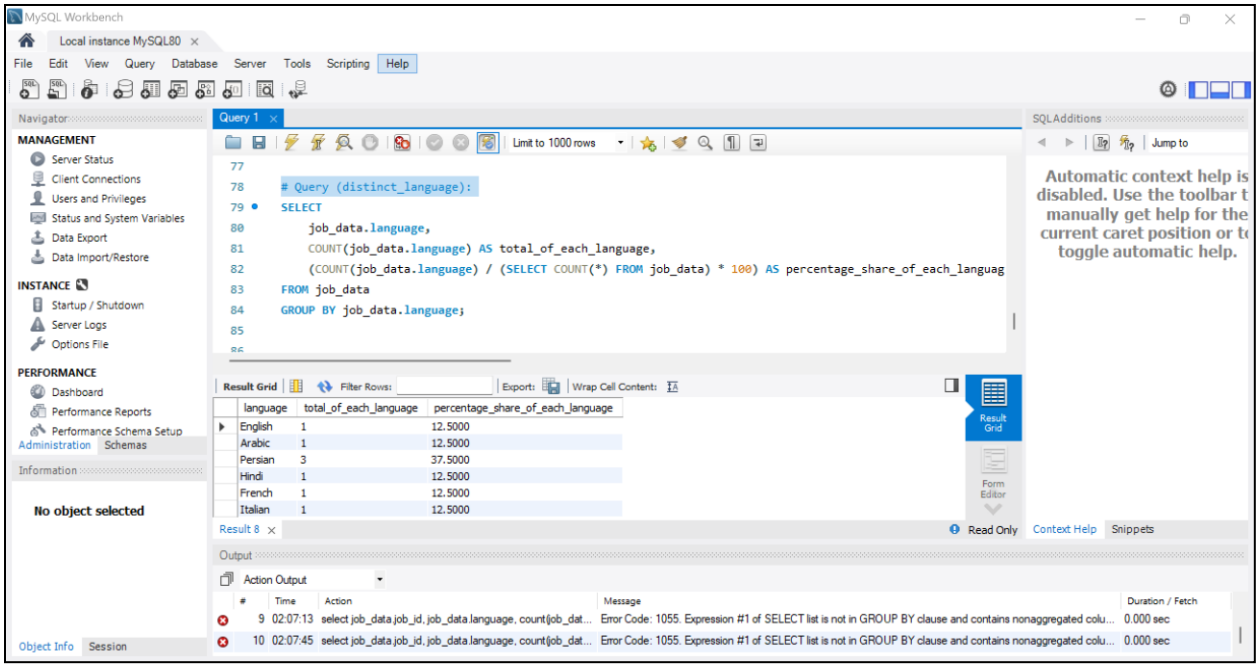
COUNT(job_data.language) AS total_of_each_language,

(COUNT(job_data.language) / (SELECT COUNT(*) FROM job_data) * 100) AS percentage_share_of_each_language

FROM job_data

GROUP BY job_data.language;

language	total_of_each_language	percentage_share_of_each_language
English	1	12.5
Arabic	1	12.5
Persian	3	37.5
Hindi	1	12.5
French	1	12.5
Italian	1	12.5



Insight: This query highlights the distribution of languages in job reviews, useful for understanding user preferences or regional trends in the data.

D. Duplicate Rows Detection

Objective: Identify and display duplicate rows in the job_data table.

Approach:

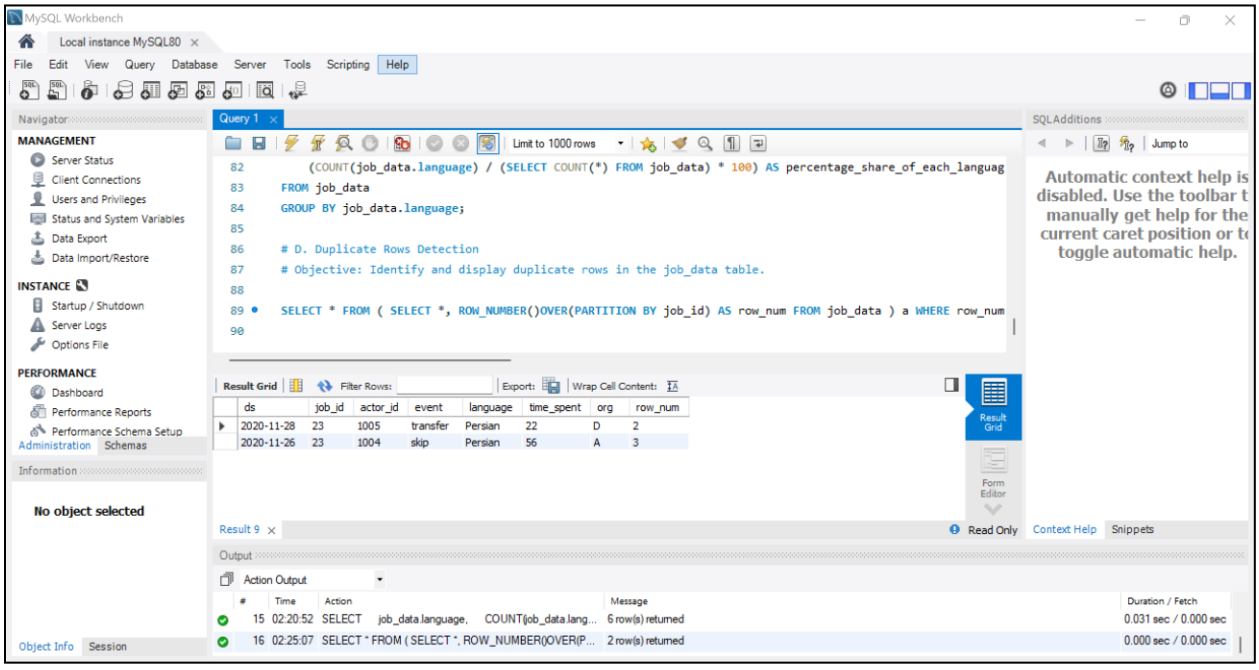
1. **Duplicate Identification**: Use ROW_NUMBER() to assign a unique row number to identical rows based on a specific column or set of columns.
2. **Filtering**: Select rows where the row number is greater than 1.

Query :

```
SELECT * FROM ( SELECT *, ROW_NUMBER()OVER(PARTITION BY job_id) AS row_num
FROM job_data ) a WHERE row_num>1;
```

Result:

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

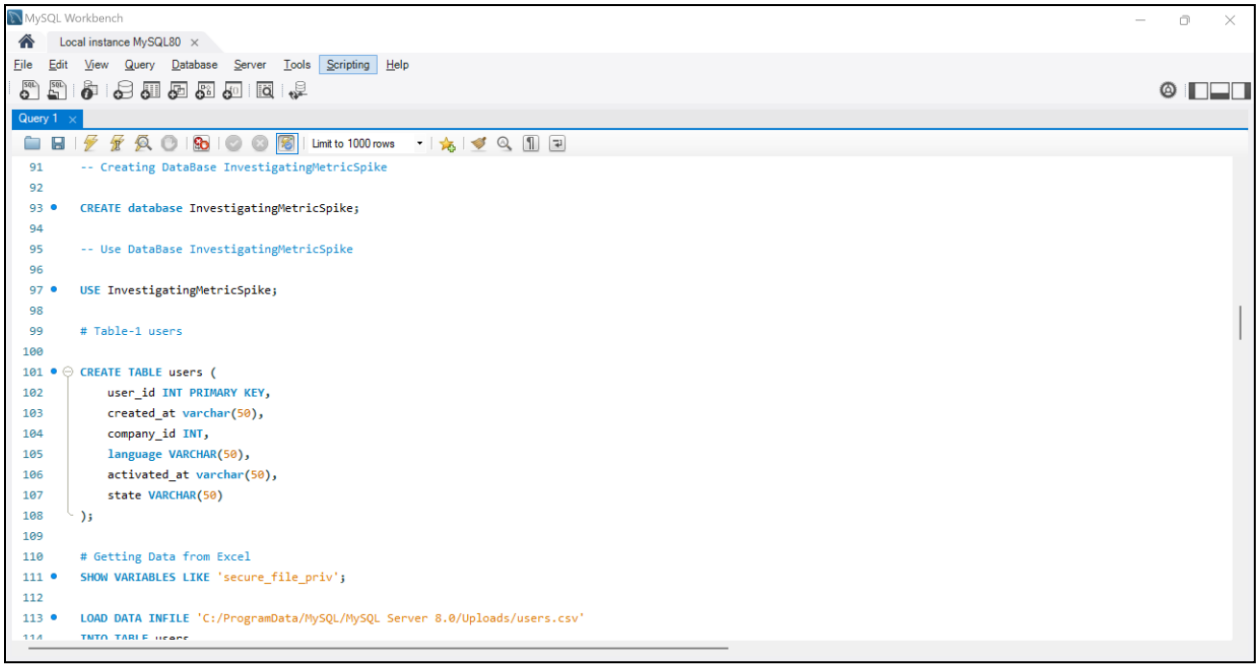


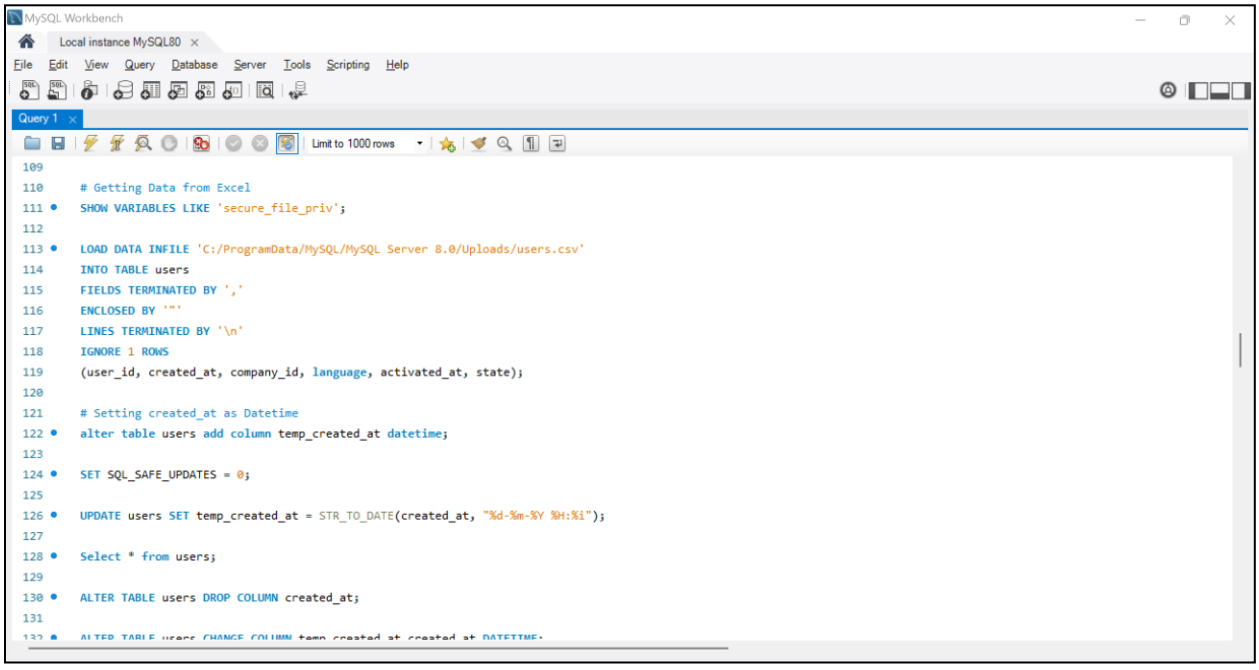
Insight: Detecting duplicate rows ensures data integrity, which is crucial for accurate reporting and analysis.

Case Study 2: Investigating Metric Spike

Creating Tables and importing Data from Excel

Table 1: users





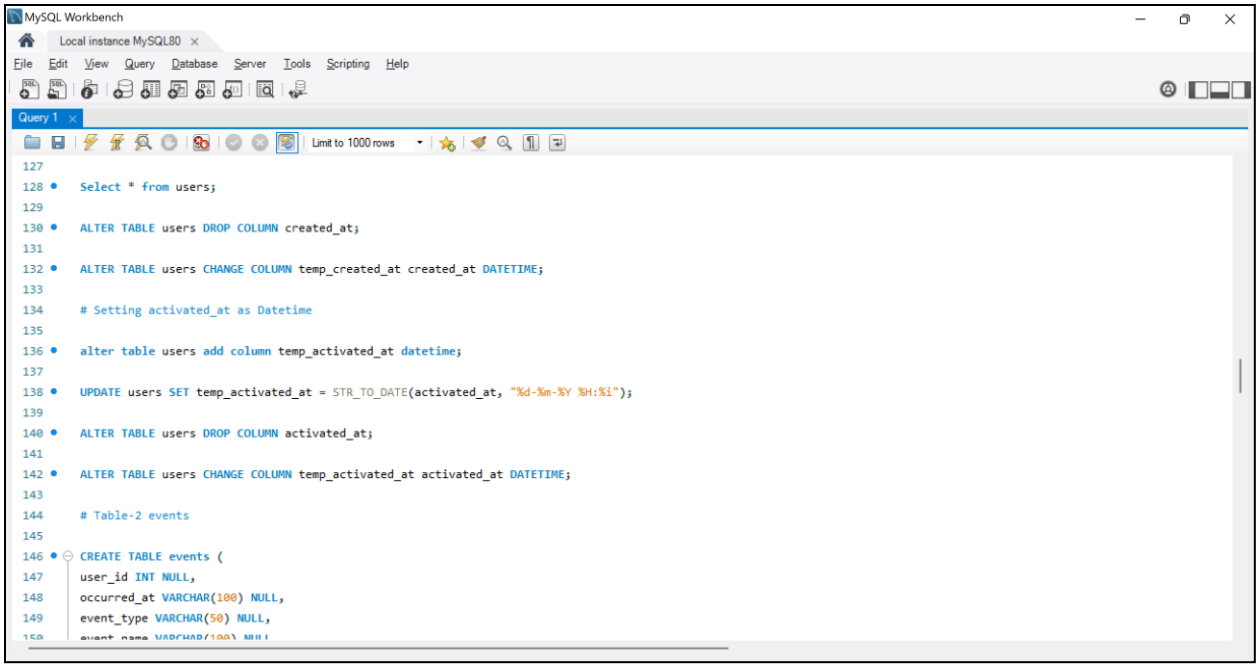
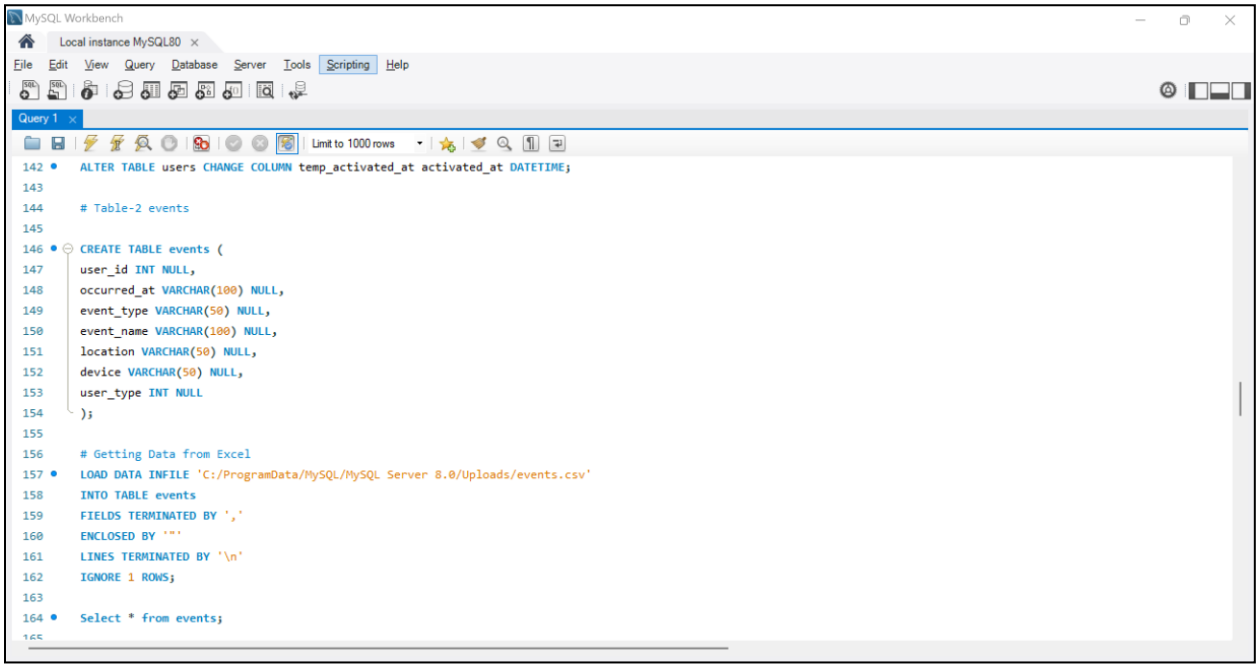
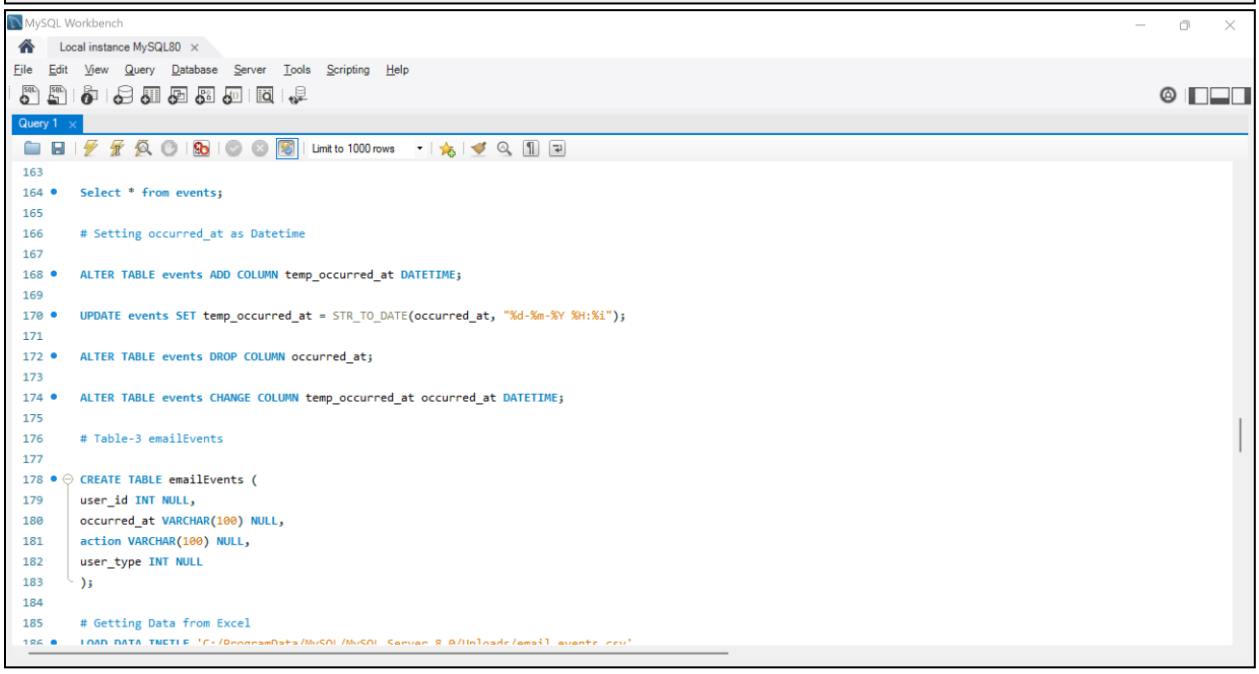

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Query 1
Limit to 1000 rows
127
128 Select * from users;
129
130 ALTER TABLE users DROP COLUMN created_at;
131
132 ALTER TABLE users CHANGE COLUMN temp_created_at created_at DATETIME;
133
134 # Setting activated_at as Datetime
135
136 alter table users add column temp_activated_at datetime;
137
138 UPDATE users SET temp_activated_at = STR_TO_DATE(activated_at, "%d-%m-%Y %H:%i");
139
140 ALTER TABLE users DROP COLUMN activated_at;
141
142 ALTER TABLE users CHANGE COLUMN temp_activated_at activated_at DATETIME;
143
144 # Table-2 events
145
146 CREATE TABLE events (
147 user_id INT NULL,
148 occurred_at VARCHAR(100) NULL,
149 event_type VARCHAR(50) NULL,

Table 2: events

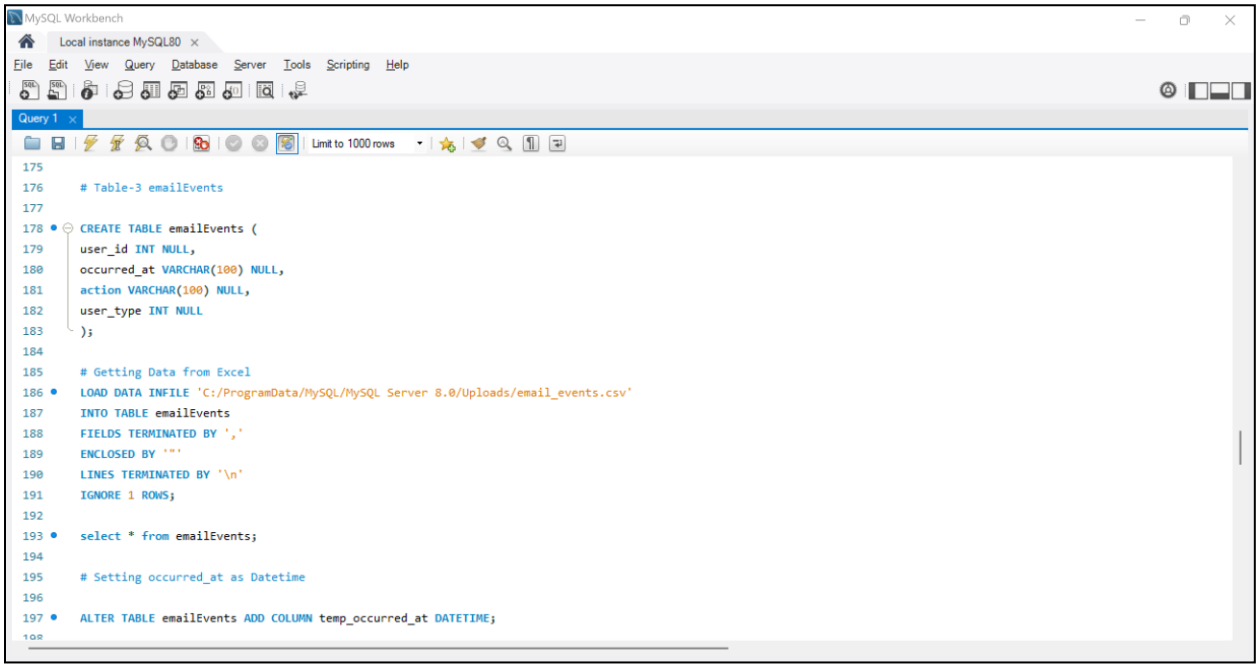


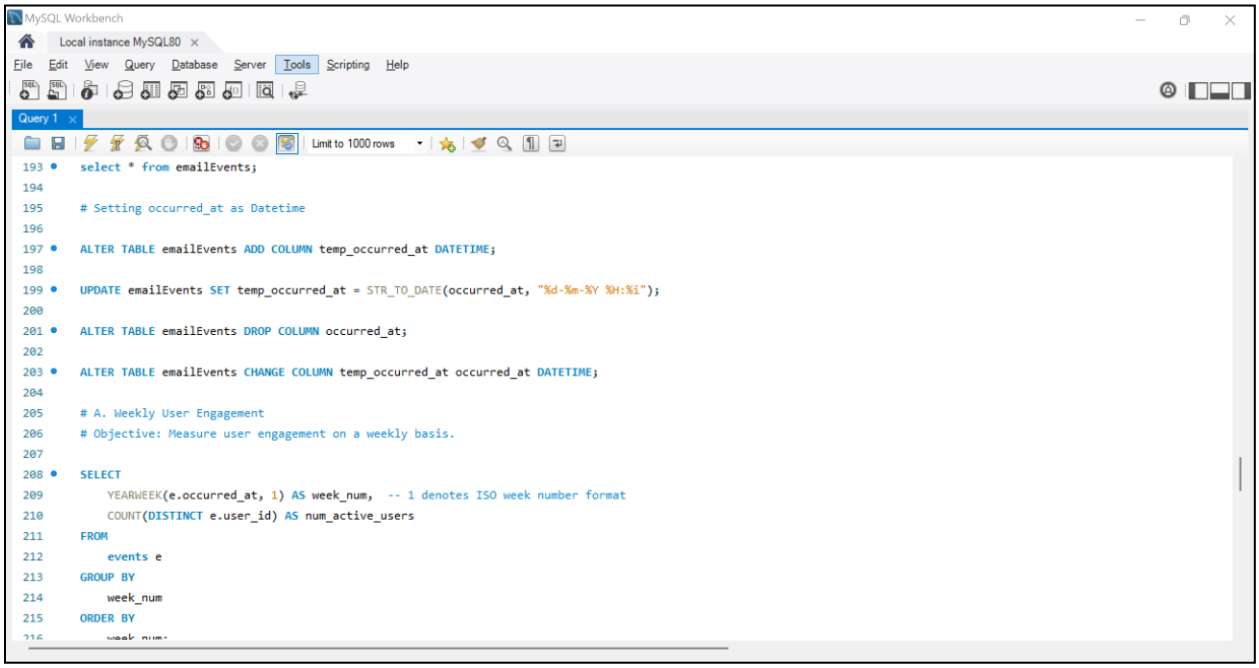
```
ALTER TABLE users CHANGE COLUMN temp_activated_at activated_at DATETIME;

# Table-2 events

CREATE TABLE events (
user_id INT NULL,
occurred_at VARCHAR(100) NULL,
event_type VARCHAR(50) NULL,
event_name VARCHAR(100) NULL,
location VARCHAR(50) NULL,
device VARCHAR(50) NULL,
user_type INT NULL
);

# Getting Data from Excel
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;
```



```
Select * from events;

# Setting occurred_at as Datetime

ALTER TABLE events ADD COLUMN temp_occurred_at DATETIME;

UPDATE events SET temp_occurred_at = STR_TO_DATE(occurred_at, "%d-%m-%Y %H:%i");

ALTER TABLE events DROP COLUMN occurred_at;

ALTER TABLE events CHANGE COLUMN temp_occurred_at occurred_at DATETIME;

# Table-3 emailEvents

CREATE TABLE emailEvents (
user_id INT NULL,
occurred_at VARCHAR(100) NULL,
action VARCHAR(100) NULL,
user_type INT NULL
);

# Getting Data from Excel
```

Table 3: emailEvents





A. Weekly User Engagement

Objective: Measure user engagement on a weekly basis.

Approach:

1. **Weekly Segmentation**: Extract the week number from the occurred_at timestamp.
2. **User Activity Count**: Count distinct user_id to measure engagement.

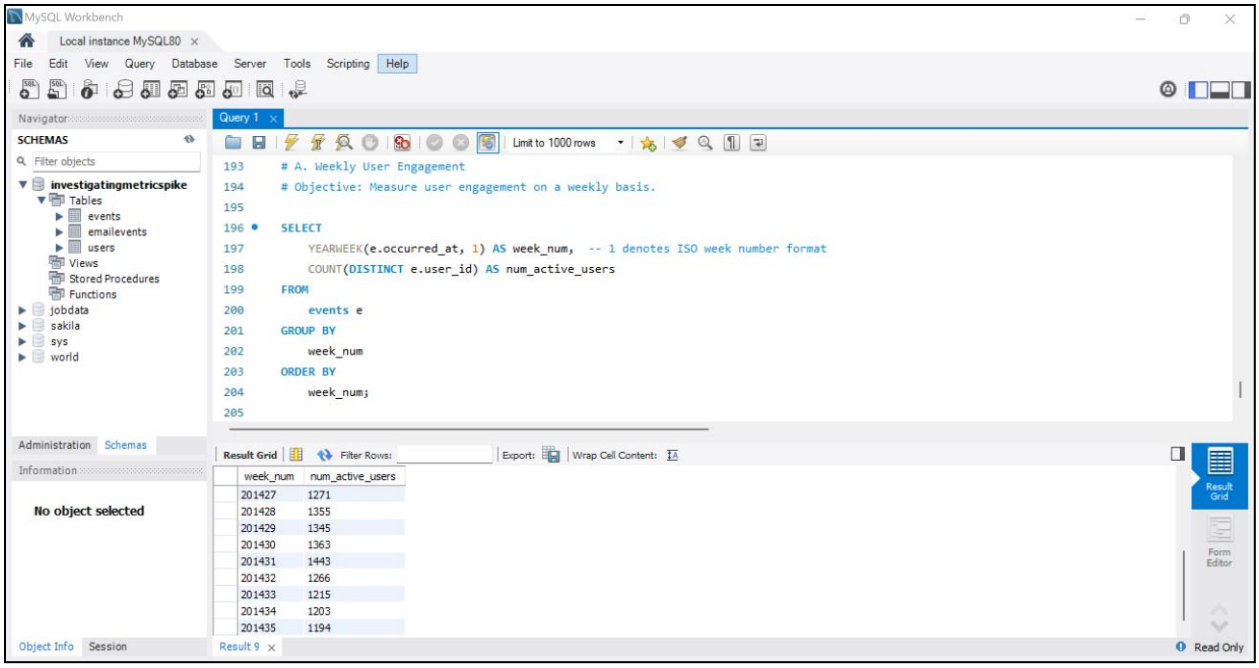
Query :

```
SELECT
    YEARWEEK(e.occurred_at, 1) AS week_num,
    COUNT(DISTINCT e.user_id) AS num_active_users
FROM
    events e
GROUP BY
    week_num
ORDER BY
    week_num;
```

Result:

week_num	num_active_users
201418	701
201419	1054
201420	1094
201421	1147
201422	1113
201423	1173

201424	1219
201425	1263
201426	1249
201427	1271
201428	1355
201429	1345
201430	1363
201431	1443
201432	1266
201433	1215
201434	1203
201435	1194



Insight: Weekly user engagement metrics help track trends in user activity and identify periods of high or low engagement.

B. User Growth Analysis

Objective: Analyze user growth over time.

Approach:

1. **Time Segmentation**: Extract the year and week from the activated_at timestamp.
2. **Growth Calculation**: Calculate cumulative user growth using a window function.

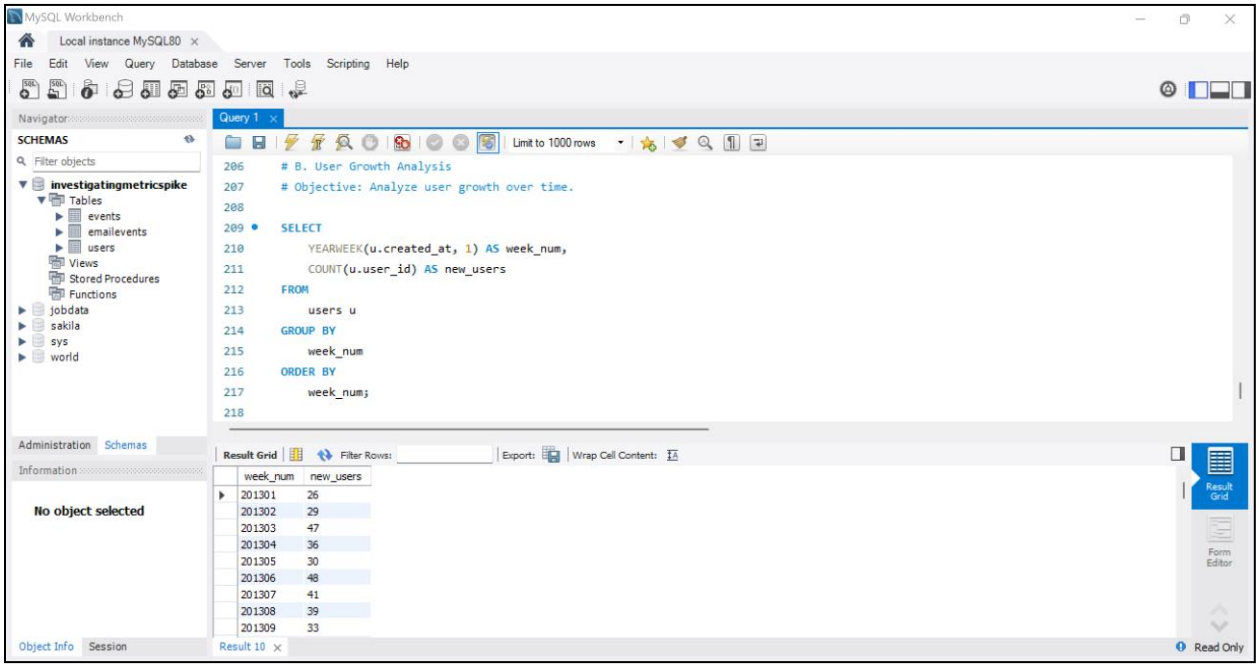
Query :

```
SELECT
    YEARWEEK(u.created_at, 1) AS week_num,
    COUNT(u.user_id) AS new_users
FROM
    users u
GROUP BY
    week_num
ORDER BY
    week_num;
```

Result:

week_num	num_active_users
201418	701
201419	1054
201420	1094
201421	1147
201422	1113
201423	1173
201424	1219

201425	1263
201426	1249
201427	1271
201428	1355
201429	1345
201430	1363
201431	1443
201432	1266
201433	1215
201434	1203
201435	1194



Insight: Monitoring cumulative user growth provides insights into the overall adoption rate and the success of user acquisition strategies.

C. Weekly Retention Analysis

Objective: Measure weekly user retention based on sign-up cohorts.

Approach:

1. **Cohort Definition**: Group users by the week of their sign-up.
2. **Retention Calculation**: Measure the percentage of users active in subsequent weeks.

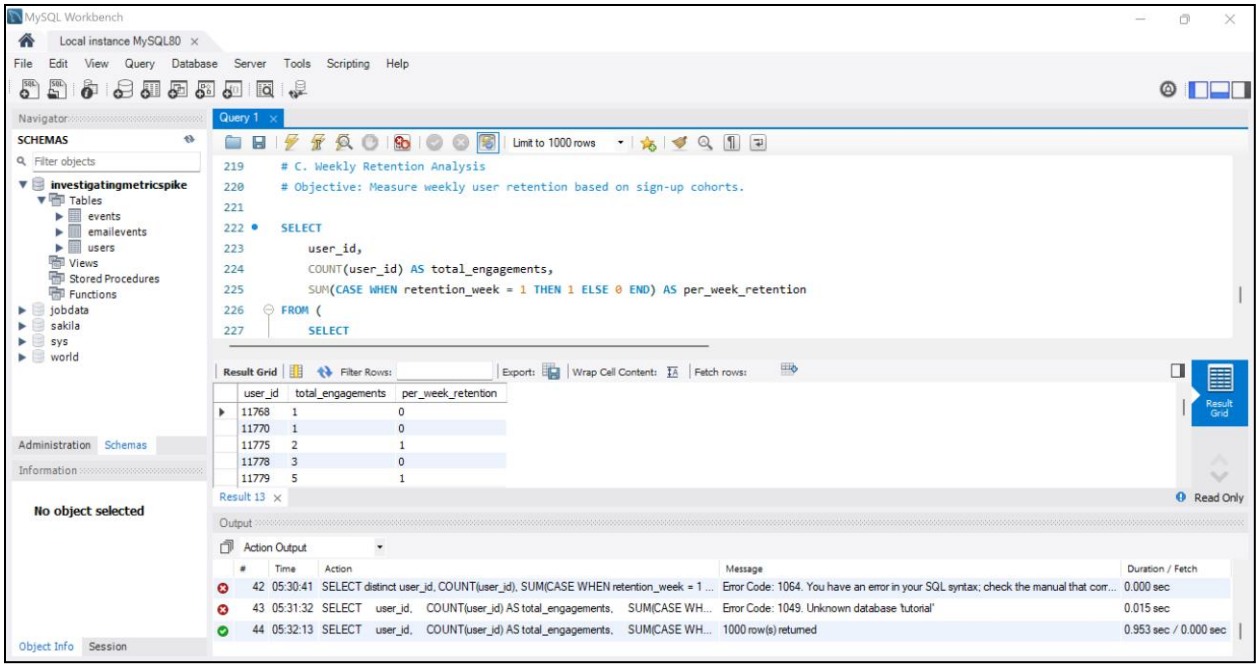
Query:

```
SELECT
    user_id,
    COUNT(user_id) AS total_engagements,
    SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS per_week_retention
FROM (
    SELECT
        a.user_id,
        a.signup_week,
        b.engagement_week,
        b.engagement_week - a.signup_week AS retention_week
    FROM (
        SELECT
            user_id,
            WEEK(occurred_at) AS signup_week
        FROM
            events
        WHERE
```

```
        event_type = 'signup_flow'
        AND event_name = 'complete_signup'
    GROUP BY
        user_id, signup_week
  ) a
  LEFT JOIN (
    SELECT
        user_id,
        WEEK(occurred_at) AS engagement_week
    FROM
        events
    WHERE
        event_type = 'engagement'
    GROUP BY
        user_id, engagement_week
  ) b ON a.user_id = b.user_id
) d
GROUP BY
  user_id
ORDER BY
  user_id;
```

Result:

https://drive.google.com/file/d/1UfiUGbVdgBgx50Fa66hdrn7xFu8X208Z/view?usp=sharing



Insight: Retention analysis helps understand user loyalty and the effectiveness of engagement strategies over time.

D. Weekly Engagement Per Device

Objective: Assess weekly user engagement by device type.

Approach:

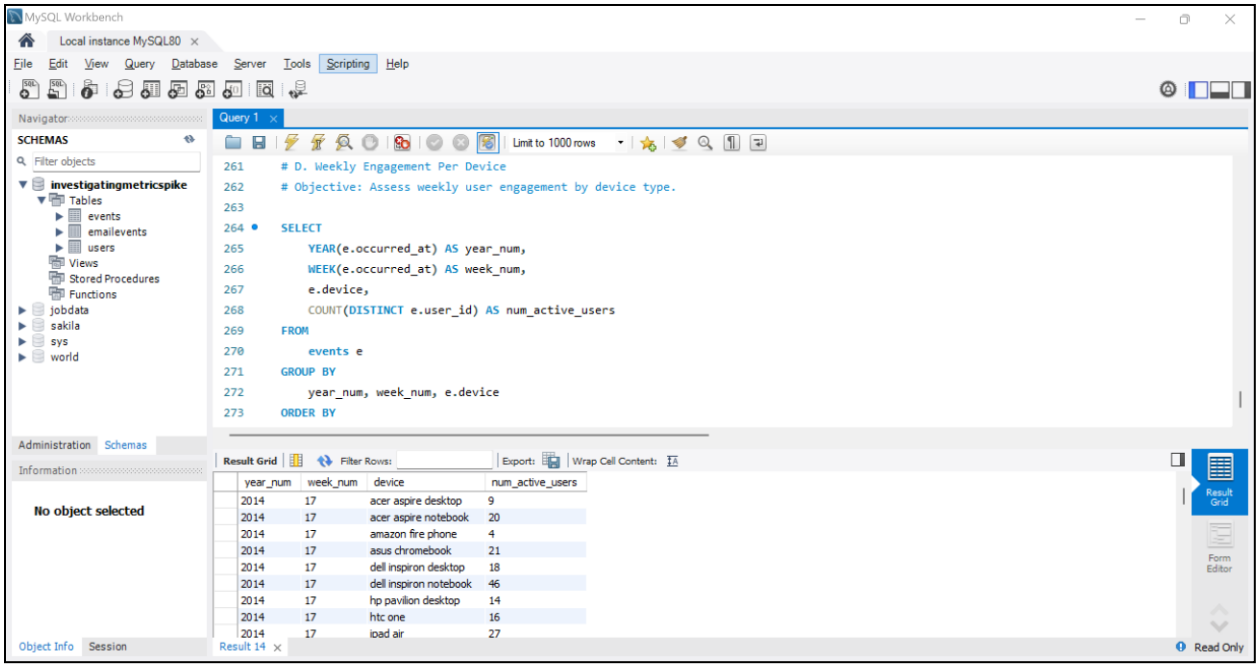
1. **Device Segmentation**: Group user events by the device used.
2. **Engagement Measurement**: Count distinct user_id to determine engagement levels per device.

Query:

```
SELECT
    YEAR(e.occurred_at) AS year_num,
    WEEK(e.occurred_at) AS week_num,
    e.device,
    COUNT(DISTINCT e.user_id) AS num_active_users
FROM
    events e
GROUP BY
    year_num, week_num, e.device
ORDER BY
    year_num, week_num, e.device;
```

Result:

https://drive.google.com/file/d/1WRJxtC08HKeaFLTsyYAoTZsEbSHBNHD9/view?usp=sharing



Insight: Understanding engagement across different devices can inform platform optimization and targeted marketing efforts.

E. Email Engagement Analysis

Objective: Analyze user interaction with email services.

Approach:

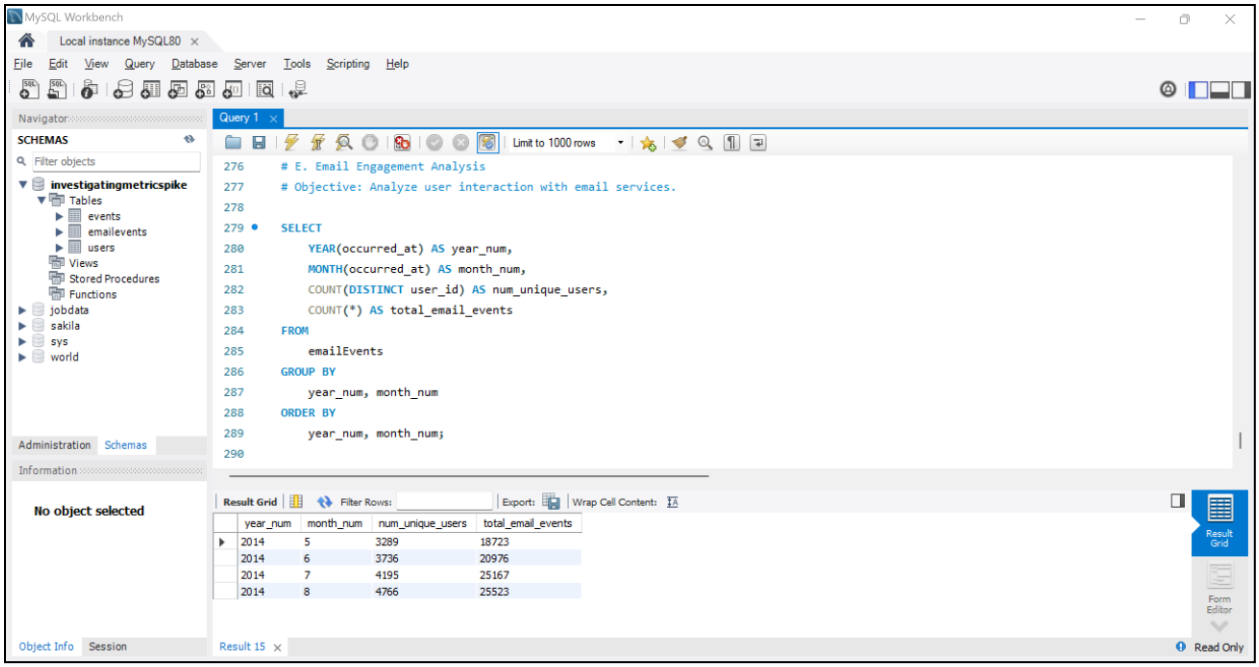
1. **Engagement Metrics**: Calculate open rates, click rates, and other key email metrics.
2. **Performance Evaluation**: Compare email engagement across different campaigns.

Query :

```
SELECT
    YEAR(occurred_at) AS year_num,
    MONTH(occurred_at) AS month_num,
    COUNT(DISTINCT user_id) AS num_unique_users,
    COUNT(*) AS total_email_events
FROM
    emailEvents
GROUP BY
    year_num, month_num
ORDER BY
    year_num, month_num;
```

Result:

year_num	month_num	num_unique_users	total_email_events
2014	5	3289	18723
2014	6	3736	20976
2014	7	4195	25167
2014	8	4766	25523



Insight: Analyzing email engagement metrics helps optimize future email campaigns by understanding what resonates with the audience.

Key Insights from Analysis:

1. **Job Reviews Per Hour:**
    - Identified peak hours for job reviews, highlighting periods of high activity.
    - Provided data to optimize review workflows and resource allocation.
2. **Throughput Analysis:**
    - 7-day rolling average smoothed out fluctuations, revealing consistent trends.
    - Helped in monitoring system performance and workload management.
3. **Language Share Analysis:**
    - Identified dominant languages used in job reviews, informing localization efforts.
    - Assisted in understanding regional trends and language preferences.
4. **Duplicate Rows Detection:**
    - Ensured data accuracy by identifying and eliminating duplicate records.
    - Improved the reliability of subsequent analysis and reporting.
5. **User Engagement and Retention:**
    - Revealed trends in user engagement, highlighting successful and underperforming weeks.
    - Provided insights into user growth and retention, informing future user acquisition strategies.

Conclusion:

The project successfully analyzed key metrics and trends within the job review data and user engagement records. By leveraging SQL and MySQL Workbench, we gained valuable insights that can directly influence strategic decisions and platform optimization.