

Paul Damein K

05-01-2025

trainity

PROJECT 3 OPERATIONAL ANALYTICS AND INVESTIGATING METRIC SPIKE

MySQL™

PROJECT DESCRIPTION:

- 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:

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:

- 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



MySQL™

1) CASE STUDY 1: JOB DATA ANALYSIS

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

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:

```

14
15 •   SELECT
16     *
17   FROM
18     job_data;
19
20 •   insert into job_data (ds, job_id, actor_id, event,language, time_spent, org)
21   values
22     ('2020-11-30',21,1001,'skip','English',15,'A'),
23     ('2020-11-30',22,1006,'transfer','Arabic',25,'B'),
24     ('2020-11-29',23,1003,'decision','Persian',20,'C'),
25     ('2020-11-28',23,1005,'transfer','Persian',22,'D'),
26     ('2020-11-28',25,1002,'decision','Hindi',11,'B'),
27     ('2020-11-27',11,1007,'decision','French',104,'D'),
28     ('2020-11-26',23,1004,'skip','Persian',56,'A'),
29     ('2020-11-25',20,1003,'transfer','italian',45,'C');

```

Administration Schemas Information Schema: core_project3

Output:

Result Grid | Filter Rows: [] Export: [] Wrap Cell Content: []

	ds	job_id	actor_id	event	language	time_spent	org
▶	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

Administration Schemas Information Schema: core_project3

Output

Action Output

#	Time	Action
1	03:59:26	SELECT * FROM core_project3.job_data

Message
8 row(s) returned

Object Info Session 06 May 2025 Tuesday 0 sec

CASE STUDY 1: 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.

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:

The screenshot shows a database interface with a query editor window. The query is as follows:

```
31  #Task-1 Calculate the number of jobs reviewed per hour for each day in November 2020.
32 • SELECT
33     ds AS date,
34     COUNT(job_id) AS tot_job_id,
35     ROUND((SUM(time_spent) / 3600), 2) AS tot_time_spent,
36     ROUND((COUNT(job_id) / (SUM(time_spent) / 3600))) AS jobs_review_perhr_perdy
37 FROM
38     job_data
39 WHERE
40     ds BETWEEN '2020-11-01' AND '2020-11-30'
41 GROUP BY ds
42 ORDER BY ds;
```

Output:

The screenshot shows the results of the query in a grid format. The columns are date, tot_job_id, tot_time_spent, and jobs_review_perhr_perdy. The data is as follows:

date	tot_job_id	tot_time_spent	jobs_review_perhr_perdy
2020-11-25	3	0.04	80
2020-11-26	3	0.05	64
2020-11-27	3	0.09	35
2020-11-28	6	0.03	218
2020-11-29	3	0.02	180
2020-11-30	6	0.03	180

Below the grid, the session details show the query was executed successfully.

Result 2 x Read Only

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
18	04:12:58	SELECT ds AS date, ROUND(COUNT(event) / SUM(time_spent), 2) AS weekly_throughput_avg FROM ...	6 row(s) returned	0.016 sec / 0.000 sec

Object Info Session

Query Completed

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:

The screenshot shows a database interface with two queries in the editor pane. The first query is a single-select statement from the 'job_data' table, resulting in a value of 0.03 for the column 'weekly_throughput_avg'. The second query is a grouped select statement from 'job_data' grouped by 'ds' (date), resulting in a grid of data where each row has a date and a corresponding throughput average. The schema browser on the left shows tables like 'core_project3.job_data', 'core_projects3_2.eventz', and 'email_eventz'. The results pane at the bottom shows the execution details.

```
44 #Task-2 Calculate the 7-day rolling average of throughput (number of events per second).
45 • SELECT
46     ROUND(COUNT(event) / SUM(time_spent), 2) AS weekly_throughput_avg
47     FROM
48         job_data; → 0.03
49
50 • SELECT
51     ds AS date,
52     ROUND(COUNT(event) / SUM(time_spent), 2) AS weekly_throughput_avg
53     FROM
54         job_data
55     GROUP BY ds
56     ORDER BY ds;
```

Output:

The screenshot shows the results grid from the previous query. It has two columns: 'date' and 'weekly_throughput_avg'. The data rows are: 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, and 2020-11-30 0.05. The bottom pane shows the execution details, including the query text, execution time (0.000 sec), and timestamp (06 May 2025 Tuesday).

date	weekly_throughput_avg
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
2020-11-30	0.05

Result 3 x Read Only

Output:

Action Output
Time Action Message Duration / Fetch 19 04:15:28 SELECT ds AS date, COUNT(job_id) AS tot_job_id, ROUND((SUM(time_spent) / 3600), 2) AS tot_time... 6 row(s) returned 0.000 sec / 0.000 sec 06 May 2025 Tuesday

Object Info Session Query Completed

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 data
SELECT    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:

Navigator: pj 3* x pj 3 task 2* email_eventz job_data

SCHEMAS

Filter objects

core_project3

- Tables
 - job_data
- Views
- Stored Procedures
- Functions

core_projects3_2

- Tables
 - email_eventz

59 #Task-3 Calculate the percentage share of each language in the last 30 days.

60 • 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;

66

Output:

Administration Schemas

Information

Schema: core_project3

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

language	percentage	total
English	12.50	24
Arabic	12.50	24
Persian	37.50	24
Hindi	12.50	24
French	12.50	24
Italian	12.50	24

Result 8 x Read Only

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
24	05.02.20	select actor_id, count(*) as duplicate from job_data group by actor_id having count(*)>1	7 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

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 data
SELECT    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:

```
▶ email_eventz  
▶ eventz  
▶ users  
Views  
Stored Procedures  
Functions  
▶ parks_and_recreation  
▶ sys  
  
66  
67      #Task-4 Identify duplicate rows in the data  
68 •  select actor_id,  
69      count(*) as duplicate from job_data  
70      group by actor_id  
71      having count(*)>1;  
72
```

Output:

Administration Schemas

Information

No object selected

Result Grid | Filter Rows: Export: Wrap Cell Content: ▾

actor_id	duplicate
1003	2

Result 2 x Read Only

Action Output

#	Time	Action	Message	Duration
7	05:19:21	insert into job_data (ds,job_id,actor_id,event.language,time_spent,org) values ('2020-11-30',21,1001,'skip','...')	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.015 sec

Object Info Session 06 May 2025 Tuesday

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

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:

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (core_project3, core_projects3_2, parks_and_recreation, sys).
- SQL Editor:** SQL File 7* containing the following SQL code:

```
4 #Table-1 Users
5 • CREATE TABLE users (
6     user_id INT,
7     created_at VARCHAR(50),
8     company_id INT,
9     language VARCHAR(50),
10    activated_at VARCHAR(50),
11    state VARCHAR(50)
12 );
13
14 • load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv'
15   into table users
16   fields terminated by ','
17   enclosed by '"'
18   lines terminated by '\n'
19   ignore 1 rows;
20
21 • alter table users add column temp_created_at datetime;
22 • UPDATE users
23   SET
24     temp_created_at = STR_TO_DATE(created_at, '%d-%m-%Y %H:%i');
25 • alter table users drop column created_at;
26 • alter table users change column temp_created_at created_at datetime;
27
28
29 #Table-2 events
```
- Output:** Action Output table showing two actions:

#	Time	Action	Message	Duration / Fetch
49	06:13:39	alter table users drop column created_at	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
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

Output:

Administration Schemas

Information

Schema: core_projects3_2

users 1 x Read Only

Action Output

#	Time	Action	Message	Duration / Fetch
50	06:13:43	altertable 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

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	01-01-01-01-2013 04:40	active	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

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:

The screenshot shows the MySQL Workbench interface with a SQL file open. The SQL code creates a table named 'eventz' with various columns like user_id, occurred_at, event_type, etc., and then imports data from a CSV file named 'events.csv' into this table. The code also includes some temporary settings and column modifications.

```
30 • CREATE TABLE eventz (
31     user_id INT,
32     occurred_at VARCHAR(50),
33     event_type VARCHAR(50),
34     event_name VARCHAR(50),
35     location VARCHAR(50),
36     device VARCHAR(100),
37     user_type INT
38 );
39 • show variables like 'secure_file_priv';
40
41 • load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/events.csv'
42 into table eventz
43 fields terminated by ','
44 enclosed by ""
45 lines terminated by '\n'
46 ignore 1 rows;
47
48 #set global sql_safe_updates = 0; Temp disable safe mode
49
50 • alter table eventz add column temp_occurred_at datetime;
51 • UPDATE eventz
52 SET
53     temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');
54 • alter table eventz drop column occurred_at;
55 • alter table eventz change column temp_occurred_at occurred_at datetime;
```

The 'Output' pane at the bottom shows the results of the last query, indicating 0 rows affected, 0 duplicates, and 0 warnings, with a duration of 0.031 sec.

#	Time	Action	Message	Duration / Fetch
56	06:20:21	alter table eventz drop column occurred_at	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec

Output:

MySQL Workbench Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas Tables Views Stored Procedures Functions Parks_and_recreation sys

Filter objects core_projects3 core_projects3_2 eventz

SQL File 7* eventz

1 • SELECT * FROM core_projects3_2.eventz;

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

user_id	event_type	event_name	location	device	user_type	occurred_at
10522	engagement	login	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	home_page	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	like_message	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10522	engagement	view_inbox	Japan	dell inspiron notebook	3	2014-05-02 11:04:00
10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10612	engagement	login	Netherlands	iphone 5	1	2014-05-01 09:59:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
10612	engagement	send_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:01:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:01:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:02:00
10612	engagement	view_inbox	Netherlands	iphone 5	1	2014-05-01 10:02:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:03:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:03:00
10612	engagement	send_message	Netherlands	iphone 5	1	2014-05-01 10:04:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:04:00

eventz 1 x Read Only

Action Output

#	Time	Action
57	06:20:25	alter table eventz change column temp_occurred_at occurred_at datetime

Object Info Session

Message 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 Duration / Fetch 0.015 sec

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_events.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_eventz
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:

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Shows the schema `core_projects3_2` with tables `eventz` and `users`.
- SQL Editor:** SQL File 7* containing the following code:

```
57  #Table-3 Emailevents
58  • CREATE TABLE email_eventz (
59      user_id INT,
60      occurred_at VARCHAR(50),
61      action VARCHAR(50),
62      user_type INT
63  );
64
65  • load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv'
66  into table email_eventz
67  fields terminated by ','
68  enclosed by "'"
69  lines terminated by '\n'
70  ignore 1 rows;
71
72  • alter table email_eventz add column temp_occurred_at datetime;
73  • UPDATE email_eventz
74  SET
75      temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');
76  • alter table email_eventz drop column occurred_at;
77  • alter table email_eventz change column temp_occurred_at occurred_at datetime;
```
- Output:** Action Output table showing the results of the operations:

#	Time	Action	Message	Duration / Fetch
60	06:23:09	load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv' into table email_eventz	90389 row(s) affected Records: 90389 Deleted: 0 Skipped: 0 Warnings: 0	1.000 sec
61	06:23:14	alter table email_eventz add column temp_occurred_at datetime	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
62	06:23:19	UPDATE email_eventz SET temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i')	90389 row(s) affected Rows matched: 90389 Changed: 90389 Warnings: 0	5.344 sec
63	06:23:29	alter table email_eventz drop column occurred_at	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
64	06:23:48	alter table email_eventz change column temp_occurred_at occurred_at datetime	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec

Output:

MySQL Workbench Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas core_projects3_2 Tables email_eventz

SQL File 7* email_eventz

1 • SELECT * FROM core_projects3_2.email_eventz;

Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows: Read Only

user_id	action	user_type	occurred_at
0	sent_weekly_digest	1	2014-05-06 09:30:00
0	sent_weekly_digest	1	2014-05-13 09:30:00
0	sent_weekly_digest	1	2014-05-20 09:30:00
0	sent_weekly_digest	1	2014-05-27 09:30:00
0	sent_weekly_digest	1	2014-06-03 09:30:00
0	email_open	1	2014-06-03 09:30:00
0	sent_weekly_digest	1	2014-06-10 09:30:00
0	email_open	1	2014-06-10 09:30:00
0	sent_weekly_digest	1	2014-06-17 09:30:00
0	email_open	1	2014-06-17 09:30:00
0	sent_weekly_digest	1	2014-06-24 09:30:00
0	sent_weekly_digest	1	2014-07-01 09:30:00
0	sent_weekly_digest	1	2014-07-08 09:30:00

Action Output

#	Time	Action	Message	Duration / Fetch
61	06:23:14	alter table email_eventz add column temp_occurred_at datetime	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
62	06:23:19	UPDATE email_eventz SET temp_occurred_at = STR_TO_DATE(occurred_at, "%d-%m-%Y %H:%i")	90389 row(s) affected Rows matched: 90389 Changed: 90389 Warnings: 0	5.344 sec
63	06:23:29	alter table email_eventz drop column occurred_at	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
64	06:23:48	alter table email_eventz change column temp_occurred_at occurred_at datetime	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
65	06:24:23	SELECT * FROM core_projects3_2.email_eventz	90389 row(s) returned	0.000 sec / 0.141 sec

CASE STUDY 2: 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.

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:

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:

```
Filter objects
core_projects3_2
  Tables
    email_eventz
    eventz
    users
  Views
  Stored Procedures
  Functions
parks_and_recreation
sys

79  #Task-1 Write an SQL query to calculate the weekly user engagement.
80  ● SELECT
81      EXTRACT(WEEK FROM occurred_at) AS week_num,
82      COUNT(DISTINCT user_id) AS active_users
83  FROM
84      eventz
85  WHERE
86      event_type = 'engagement'
87  GROUP BY week_num
88  ORDER BY week_num;
89
```

Output:

Administration Schemas

Information

Schema: core_projects3_2

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Read Only

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	101

Result 2 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
66	06:29:01	SELECT YEAR(created_at) AS year, WEEK(created_at) AS week_num, COUNT(DISTINCT user_id) AS active_users	89 row(s) returned	0.047 sec / 0.000 sec
67	06:31:13	SELECT EXTRACT(WEEK FROM occurred_at) AS week_num, COUNT(DISTINCT user_id) AS active_users	19 row(s) returned	0.578 sec / 06 May 2025 Tuesday

Object Info Session

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:

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:

The screenshot shows a database interface with a query editor. On the left, there's a tree view of schemas and tables under 'core_projects3_2'. The 'Tables' section contains 'email_eventz', 'eventz', and 'users'. The 'Views' section is empty. The 'Stored Procedures' and 'Functions' sections also contain nothing. The 'parks_and_recreation' and 'sys' schemas are shown at the bottom. The main area displays a SQL query:

```
91  SELECT
92      YEAR(created_at) AS year,
93      WEEK(created_at) AS week_num,
94      COUNT(DISTINCT user_id) AS weekly_users,
95      SUM(COUNT(DISTINCT user_id)) OVER (ORDER BY YEAR(created_at), WEEK(created_at)) AS cumulative_users
96
97  FROM
98      users
99  GROUP BY
100      YEAR(created_at),
101      WEEK(created_at)
102  ORDER BY
103      year,
104      week_num;
```

Output:

The screenshot shows the results of the executed query in a grid format. The grid has four columns: 'year', 'week_num', 'weekly_users', and 'cumulative_users'. The data spans from week 0 to week 19 of 2013. The cumulative user count grows exponentially over time.

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

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_nums,
    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_nums,
        (EXTRACT(WEEK FROM occurred_at) - MIN(EXTRACT(WEEK FROM occurred_at))) OVER (PARTITION BY user_id) AS week_diff
    ) AS sub
GROUP BY
    week_nums
ORDER BY
    week_nums;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:

RESULT:

Query:

Navigator

SQL File 7* | email_eventz

SCHEMAS

core_projects3_2

Tables

- email_eventz
- eventz
- users
- Views
- Stored Procedures
- Functions
- parks_and_recreation
- sys

Administration Schemas

Information

Schema: core_projects3_2

```

108   SELECT
109     week_nums,
110     SUM(CASE WHEN week_diff = 0 THEN 1 ELSE 0 END) AS week_0,
111     SUM(CASE WHEN week_diff = 1 THEN 1 ELSE 0 END) AS week_1,
112     SUM(CASE WHEN week_diff = 2 THEN 1 ELSE 0 END) AS week_2,
113     SUM(CASE WHEN week_diff = 3 THEN 1 ELSE 0 END) AS week_3,
114     SUM(CASE WHEN week_diff = 4 THEN 1 ELSE 0 END) AS week_4,
115     SUM(CASE WHEN week_diff = 5 THEN 1 ELSE 0 END) AS week_5,
116     SUM(CASE WHEN week_diff = 6 THEN 1 ELSE 0 END) AS week_6,
117     SUM(CASE WHEN week_diff = 7 THEN 1 ELSE 0 END) AS week_7,
118     SUM(CASE WHEN week_diff = 8 THEN 1 ELSE 0 END) AS week_8,
119     SUM(CASE WHEN week_diff = 9 THEN 1 ELSE 0 END) AS week_9,
120     SUM(CASE WHEN week_diff = 10 THEN 1 ELSE 0 END) AS week_10,
121     SUM(CASE WHEN week_diff = 11 THEN 1 ELSE 0 END) AS week_11,
122     SUM(CASE WHEN week_diff = 12 THEN 1 ELSE 0 END) AS week_12,
123     SUM(CASE WHEN week_diff = 13 THEN 1 ELSE 0 END) AS week_13,
124     SUM(CASE WHEN week_diff = 14 THEN 1 ELSE 0 END) AS week_14,
125     SUM(CASE WHEN week_diff = 15 THEN 1 ELSE 0 END) AS week_15,
126     SUM(CASE WHEN week_diff = 16 THEN 1 ELSE 0 END) AS week_16,
127     SUM(CASE WHEN week_diff = 17 THEN 1 ELSE 0 END) AS week_17,
128     SUM(CASE WHEN week_diff = 18 THEN 1 ELSE 0 END) AS week_18
129
130   FROM (
131     SELECT
132       user_id,
133       EXTRACT(WEEK FROM occurred_at) AS login_week,
134       MIN(EXTRACT(WEEK FROM occurred_at)) OVER (PARTITION BY user_id) AS week_nums,
135       (EXTRACT(WEEK FROM occurred_at) - MIN(EXTRACT(WEEK FROM occurred_at)) OVER (PARTITION BY user_id)) AS week_diff
136
137     FROM
138       eventz
139     ) AS sub
140     GROUP BY
141       week_nums
142     ORDER BY
143       week_nums
144

```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
74	07:14:28	SELECT year_week, SUM(CASE WHEN device = 'dell inspiron notebook' THEN weekly_active_users E...	18 row(s) returned	0.859 sec / 0.000 sec

Object Info Session

RESULT:

Output:

SQL File 7* x email_eventz

```

128      SUM(CASE WHEN week_diff = 17 THEN 1 ELSE 0 END) AS week_17,
129      SUM(CASE WHEN week_diff = 18 THEN 1 ELSE 0 END) AS week_18
130
131  FROM (
132
133      SELECT
134          user_id,
135          EXTRACT(WEEK FROM occurred_at) AS login_week,
136          HOUR(FROM_UNIXTIME(UNIX_TIMESTAMP(occurred_at))) AS week_minute
137

```

	week_nums	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 10 x

Output:

Action Output

#	Time	Action	Message	Duration
74	07:14:28	SELECT year_week, SUM(CASE WHEN device = 'dell inspiron notebook' THEN weekly_active_users E...	18 row(s) returned	06 May 2025 0.859 se Tuesday

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 = 'samsung galaxy tablet' THEN weekly_active_users ELSE 0 END) AS 'samsung_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:

Navigator

SQL File 7* x email_eventz

Schemas

core_projects3_2

Tables

- email_eventz
- eventz
- users

Views

Stored Procedures

Functions

parks_and_recreation

sys

#Task 4 query that shows devices in columns

```

161  SELECT
162  year_week,
163  SUM(CASE WHEN device = 'dell inspiron notebook' THEN weekly_active_users ELSE 0 END) AS 'dell_inspiron_notebook',
164  SUM(CASE WHEN device = 'iphone 5' THEN weekly_active_users ELSE 0 END) AS 'iphone_5',
165  SUM(CASE WHEN device = 'iphone 4s' THEN weekly_active_users ELSE 0 END) AS 'iphone_4s',
166  SUM(CASE WHEN device = 'windows surface' THEN weekly_active_users ELSE 0 END) AS 'windows_surface',
167  SUM(CASE WHEN device = 'macbook air' THEN weekly_active_users ELSE 0 END) AS 'macbook_air',
168  SUM(CASE WHEN device = 'kindle fire' THEN weekly_active_users ELSE 0 END) AS 'kindle_fire',
169  SUM(CASE WHEN device = 'ipad mini' THEN weekly_active_users ELSE 0 END) AS 'ipad_mini',
170  SUM(CASE WHEN device = 'macbook pro' THEN weekly_active_users ELSE 0 END) AS 'macbook_pro',
171  SUM(CASE WHEN device = 'nexus 7' THEN weekly_active_users ELSE 0 END) AS 'nexus_7',
172  SUM(CASE WHEN device = 'nexus 5' THEN weekly_active_users ELSE 0 END) AS 'nexus_5',
173  SUM(CASE WHEN device = 'samsung galaxy s4' THEN weekly_active_users ELSE 0 END) AS 'samsung_galaxy_s4',
174  SUM(CASE WHEN device = 'lenovo thinkpad' THEN weekly_active_users ELSE 0 END) AS 'lenovo_thinkpad',
175  SUM(CASE WHEN device = 'samsung galaxy tablet' THEN weekly_active_users ELSE 0 END) AS 'samsung_galaxy_tablet',
176  SUM(CASE WHEN device = 'acer aspire notebook' THEN weekly_active_users ELSE 0 END) AS 'acer_aspire_notebook',
177  SUM(CASE WHEN device = 'asus chromebook' THEN weekly_active_users ELSE 0 END) AS 'asus_chromebook',
178  SUM(CASE WHEN device = 'dell inspiron desktop' THEN weekly_active_users ELSE 0 END) AS 'dell_inspiron_desktop',
179  SUM(CASE WHEN device = 'hp pavilion desktop' THEN weekly_active_users ELSE 0 END) AS 'hp_pavilion_desktop',
180  SUM(CASE WHEN device = 'htc one' THEN weekly_active_users ELSE 0 END) AS 'htc_one',
181  SUM(CASE WHEN device = 'iphon 5s' THEN weekly_active_users ELSE 0 END) AS 'iphone_5s',
182  SUM(CASE WHEN device = 'mac mini' THEN weekly_active_users ELSE 0 END) AS 'mac_mini',
183  SUM(CASE WHEN device = 'nexus 10' THEN weekly_active_users ELSE 0 END) AS 'nexus_10',
184  SUM(CASE WHEN device = 'nokia lumia 635' THEN weekly_active_users ELSE 0 END) AS 'nokia_lumia_635',
185  SUM(CASE WHEN device = 'samsung galaxy note' THEN weekly_active_users ELSE 0 END) AS 'samsung_galaxy_note'
186
187  FROM (
188    SELECT
189      DATE_FORMAT(occurred_at, 'NY-Nu') AS year_week, device,
190      COUNT(DISTINCT user_id) AS weekly_active_users
191    FROM events
192    GROUP BY year_week, device
193  ) AS sub
194  GROUP BY year_week
195  ORDER BY year_week;
  
```

Administration Schemas

Information

Schema: core_projects3_2

Output

RESULT:

Output:

SQL File 7* x email_eventz

```

185      SUM(CASE WHEN device = 'nokia lumia 635' THEN weekly_active_users ELSE 0 END) AS 'nokia_lumia_635',
186      SUM(CASE WHEN device = 'samsung galaxy note' THEN weekly_active_users ELSE 0 END) AS 'samsung_galaxy_note'
187  FROM (
188    SELECT
189      DATE_FORMAT(occurred_at, '%Y-%u') AS year_week, device,
190      COUNT(DISTINCT user_id) AS weekly_active_users
191    FROM events
192    GROUP BY year_week, device
193  ) AS sub
194  GROUP BY year_week
195  ORDER BY year_week;
196

```

Result Grid | Filter Rows: [] | Export: | Wrap Cell Content:

year_week	dell_inspiron_notebook	iphone_5	iphone_4s	windows_surface	macbook_air	kindle_fire	ipad_mini	macbook_pro	nexus_7	nexus_5	samsung_galaxy_s4	lenovo_thinkpad	samsung_galaxy_tablet	acer_aspire_notebo
2014-18	49	70	21	10	57	6	21	154	20	43	56	90	8	21
2014-19	78	114	47	10	119	26	29	248	29	73	80	155	11	34
2014-20	82	113	40	15	110	20	37	261	41	84	90	176	6	40
2014-21	84	128	56	19	119	22	32	256	31	99	92	177	9	40
2014-22	81	136	46	17	107	30	25	244	29	94	84	164	6	47
2014-23	91	122	41	15	145	21	32	254	44	95	103	170	11	39
2014-24	100	151	52	16	122	25	32	259	37	87	95	176	14	43
2014-25	102	143	52	21	149	25	38	251	47	85	102	164	11	42
2014-26	108	134	39	19	119	24	31	276	49	90	100	196	12	44
2014-27	90	150	49	22	134	26	41	259	45	86	114	188	12	35
2014-28	91	159	68	31	140	25	35	301	41	83	119	195	15	47
2014-29	100	148	58	33	145	29	34	295	39	83	120	220	9	50
2014-30	114	147	61	28	146	36	36	291	43	81	117	209	13	52
2014-31	125	151	63	18	156	24	34	317	60	81	104	208	9	62
2014-32	111	133	52	18	143	14	23	317	39	65	99	196	7	56
2014-33	101	119	35	10	124	12	31	307	24	66	85	177	6	56
2014-34	111	105	21	11	124	11	27	300	20	71	75	199	12	45

Result 9 x

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
73	06:59:27	SELECT week_nums, SUM(CASE WHEN week_diff = 0 THEN 1 ELSE 0 END) AS week_0, SUM(C... 19 row(s) returned		2.140 sec / 0.000 sec

Object Info Session

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 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- core_project3
- core_projects3_2
 - Tables
 - email_eventz
 - eventz
 - users
 - Views
 - Stored Procedures
 - Functions
- parks_and_recreation
- sys

SQL File 7* x email_eventz

```
196 #Task-5 Write an SQL query to calculate the email engagement metrics.
197 -- 1. Email Send Count:
198 • SELECT
199     DATE_FORMAT(occurred_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

Administration Schemas

Information

Schema:

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:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

core_projects3

core_projects3_2

Tables

- email_eventz
- eventz
- users
- Views
- Stored Procedures
- Functions

parks_and_recreation

sys

SQL File 7* email_eventz

```
210  #-- 2. Email Open Rate:
211 •  SELECT
212      DATE_FORMAT(occurred_at, '%Y-%m') AS month,
213      COUNT(DISTINCT CASE WHEN action IN ('sent_reengagement_email', 'sent_weekly_digest') THEN user_id END) AS users_sent,
214      COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) AS users_opened,
215      (COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) / COUNT(DISTINCT CASE WHEN action IN ('sent_reengagement_email', 'sent_weekly_digest'
216      FROM
217          email_eventz
218      WHERE action IN ('sent_reengagement_email', 'sent_weekly_digest', 'email_open')
219      GROUP BY
220          month
221      ORDER BY
222          month;
```

Result Grid | Filter Rows: Export: Wrap Cell Content: □

month	users_sent	users_opened	open_rate
2014-05	3289	2681	81.5141
2014-06	3736	3037	81.2901
2014-07	4195	3457	82.4076
2014-08	4766	3879	81.3890

Administration Schemas

Information

Schema: core_projects3_2

Result 13 x

Output

Action Output

#	Time	Action	Message	Duration /	Read Only
81	07:35:03	SELECT DATE_FORMAT(occurred_at, "%Y-%m") AS month, COUNT(DISTINCT CASE WHEN action IN (... 4 row(s) returned		0.375 sec	Tuesday

Object Info Session

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;

RESULT:

Output:

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS core_project3
core_projects3_2

Tables email_eventz eventz users Views Stored Procedures Functions parks_and_recreation sys

SQL File 7* email_eventz

224 #-- 3. Click-Through Rate (CTR):
225 • SELECT
226 DATE_FORMAT(occurred_at, '%Y-%m') AS month,
227 COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END) AS users_opened,
228 COUNT(DISTINCT CASE WHEN action = 'email_clickthrough' THEN user_id END) AS clicks,
229 (COUNT(DISTINCT CASE WHEN action = 'email_clickthrough' THEN user_id END) / COUNT(DISTINCT CASE WHEN action = 'email_open' THEN user_id END)) * 100 AS
230 FROM
231 email_eventz
232 WHERE action IN ('email_open', 'email_clickthrough')
233 GROUP BY
234 month
235 ORDER BY
236 month;

Result Grid | Filter Rows: Export: Wrap Cell Content: □

month	users_opened	clicks	ctr
2014-05	2681	1703	63.5211
2014-06	3037	1915	63.0556
2014-07	3457	2267	65.5771
2014-08	3879	1804	46.5068

Administration Schemas Information

Schema: core_projects3_2

Result 14 × Read Only

Action Output

#	Time	Action	Message	Duration
84	07:37:10	SELECT DATE_FORMAT(occurred_at, "%Y-%m") AS month, COUNT(DISTINCT CASE WHEN action = 'e... Error Code: 1054. Unknown column 'occurred_at' in field list'	Error Code: 1054. Unknown column 'occurred_at' in field list'	0.000 sec Tuesday

Object Info Session

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:

MySQL Workbench Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Schemas core_project3 core_projects3_2

Tables email_eventz eventz users Views Stored Procedures Functions parks_and_recreation sys

SQL File 7 × email_eventz

```
238  #-- 4. User Engagement:
239 •  SELECT
240    user_id,
241    COUNT(CASE WHEN action IN ('sent_reengagement_email', 'sent_weekly_digest') THEN user_id ELSE NULL END) AS sent,
242    COUNT(CASE WHEN action = 'email_open' THEN user_id ELSE NULL END) AS opens,
243    COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id ELSE NULL END) AS clicks
244  FROM
245    email_eventz
246  WHERE
247    action IN ('sent_reengagement_email', 'sent_weekly_digest', 'email_open', 'email_clickthrough')
248  GROUP BY
249    user_id
250  ORDER BY
251    user_id;
252 <
```

Administration Schemas

Information Schema: core_projects3_2

Result Grid | Filter Rows: Export: Wrap Cell Content: Fetch rows: Result 15 × Read Only

user_id	sent	opens	clicks
0	17	5	0
4	17	5	4
8	17	3	1
11	17	5	2
17	17	4	1
19	17	5	1
20	17	8	3
22	17	7	3

Action Output

#	Time	Action	Message	Duration /
85	07:37:16	SELECT DATE_FORMAT(occurred_at, "%Y-%m") AS month, COUNT(DISTINCT CASE WHEN action = 'e...	4 row(s) returned	0.171 sec / Tuesday

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:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

core_project3

core_projects3_2

Tables

email_eventz

eventz

users

Views

Stored Procedures

Functions

parks_and_recreation

sys

SQL File 7 x email_eventz

253 -- 5. Percentage of Each Action:

254 • SELECT

255 action,

256 COUNT(*) AS action_count,

257 (COUNT(*) / (SELECT COUNT(*) FROM email_eventz)) * 100 AS action_percentage

258 FROM

259 email_eventz

260 GROUP BY

261 action

262 ORDER BY

263 action_percentage DESC;

264

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

action	action_count	action_percentage
sent_weekly_digest	57267	63.3562
email_open	20459	22.6344
email_clickthrough	9010	9.9680
sent_reengagement_email	3653	4.0414

Administration Schemas

Information

Schema: core_projects3_2

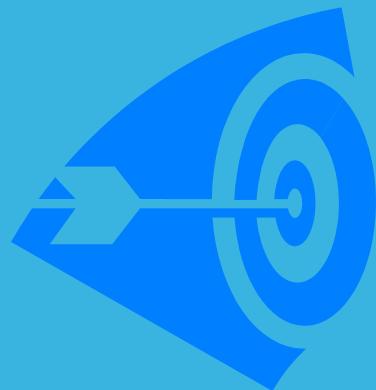
Result 16 x

Output

Action Output

#	Time	Action	Message	Duration
86	07:39:34	SELECT user_id, COUNT(CASE WHEN action IN ('sent_reengagement_email', 'sent_weekly_digest') THEN 1 ELSE 0 END) AS count FROM email_eventz WHERE user_id = 1	6179 row(s) returned	0.234 sec 06 May 2025 Tuesday

Object Info Session



**THANK
YOU**

