**DESCRIPTION:**

Operation Analytics: Operation analytics involves analyzing operational metrics using SQL queries. By using real-world data, we bring out patterns and designs from the given data. These insights enable informed decisions, enhance user experiences, and guide strategic improvements.

Investigating Metric Spike: Investigating metric spikes is a daily task that helps uncover the reasons behind fluctuations in business performance. The ultimate objective is to address inquiries raised by various departments and extract meaningful insights from diverse datasets and tables provided.

Table created named job\_datas, includes columns named job\_id, actor\_id, event, language, time\_spent, org, ds.

Job\_datas TABLE

Input:

CREATE TABLE job\_datas (

ds DATE,

job\_id INT ,

actor\_id INT ,

event VARCHAR(255),

language VARCHAR(255),

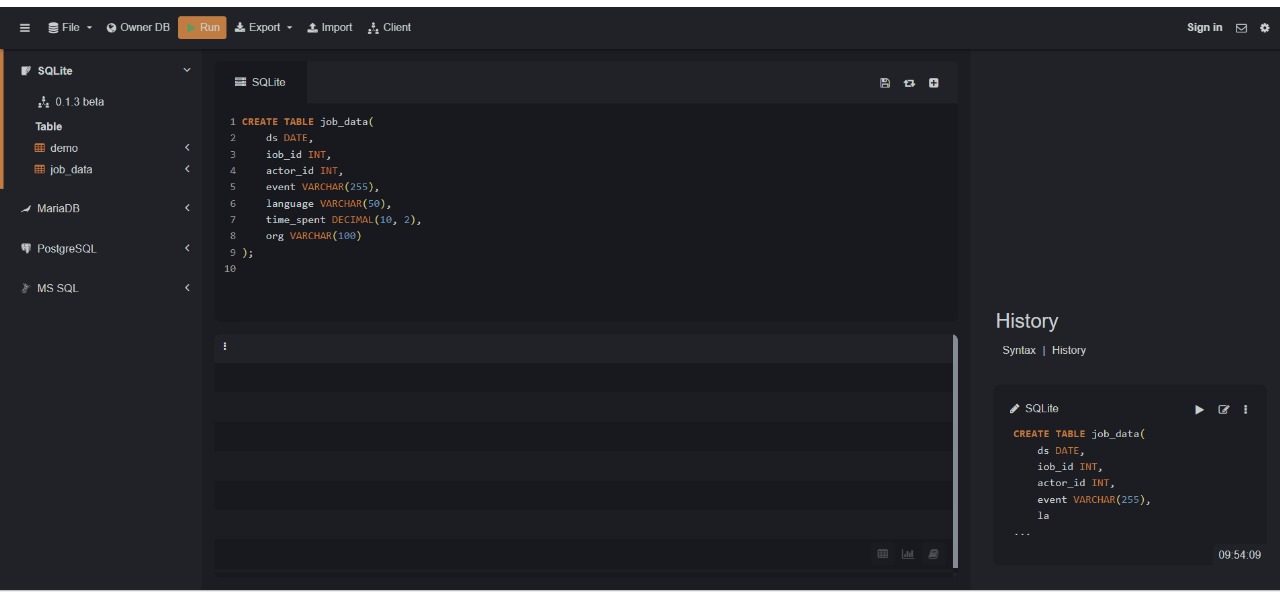
time\_spent DECIMAL(10, 2),

org VARCHAR(255)

);

Output:

Created table successfully.



Insert values:

INSERT INTO job\_datas VALUES ('2020-11-30', 21, 1001, 'skip', 'english', 15, 'A');

INSERT INTO job\_datas VALUES ('2020-11-30', 22, 1006, 'transfer', 'arabic', 25, 'B');

INSERT INTO job\_datas VALUES ('2020-11-29', 23, 1003, 'decision', 'persian', 20, 'c');

INSERT INTO job\_datas VALUES ('2020-11-28', 23, 1005, 'transfer', 'persian', 22, 'D');

INSERT INTO job\_datas VALUES ('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B');

INSERT INTO job\_datas VALUES ('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D');

INSERT INTO job\_datas VALUES ('2020-11-26', 23, 1004, 'skip', 'persian', 56, 'A');

INSERT INTO job\_datas VALUES ('2020-11-25', 20, 1003, 'transfer', 'italian', 45, 'C');

Output:

Successfully inserted values.

Display table:

Input:

select \* from job\_datas;

Output:

"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"

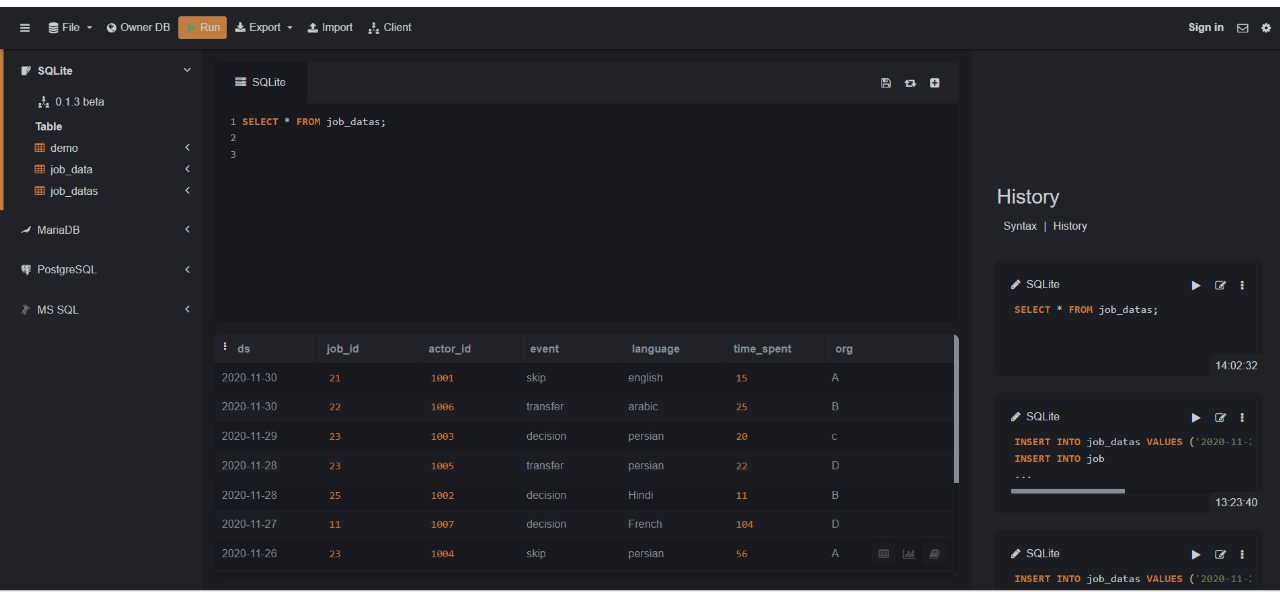
"2020-11-28" "23" "1005" "transfer" "persian" "22" "D"

"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"



CASE STYDY1: Job Data Analytics

(i)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.

Query:

SELECT

DATE(ds) AS review\_date,

CAST(strftime('%H', ds) AS INTEGER) AS review\_hour,

COUNT(\*) AS jobs\_reviewed

FROM job\_datas

WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'

GROUP BY review\_date, review\_hour

ORDER BY review\_date, review\_hour;

OUTPUT:

"2020-11-25" "0" "1"

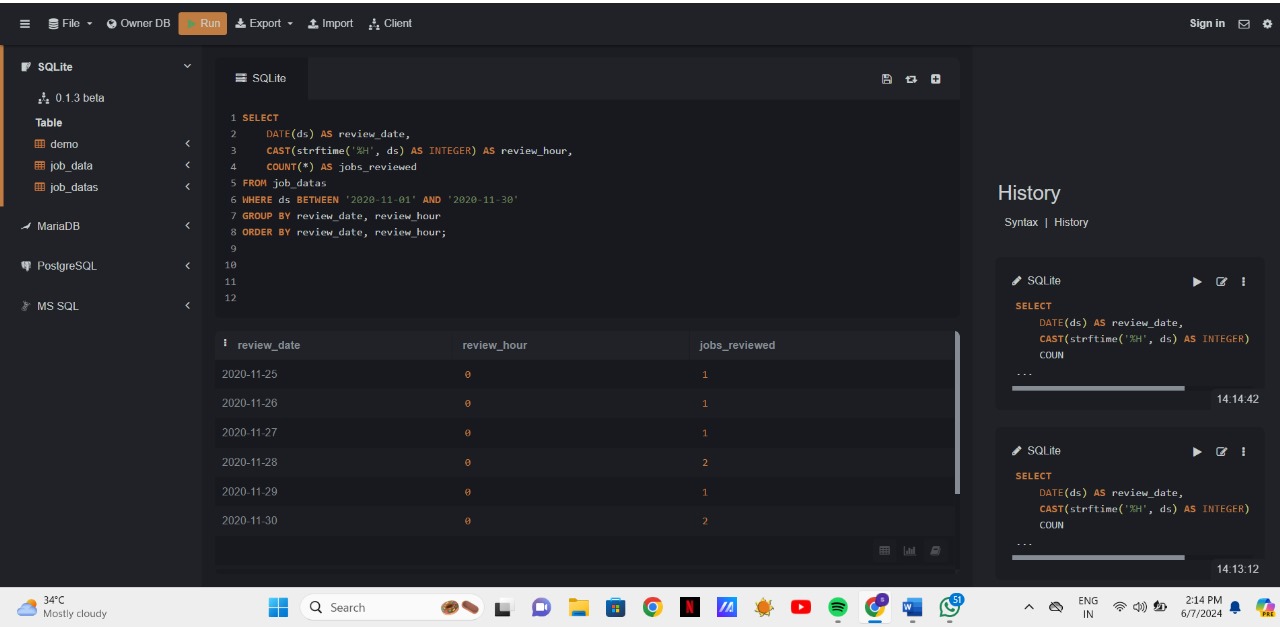
"2020-11-26" "0" "1"

"2020-11-27" "0" "1"

"2020-11-28" "0" "2"

"2020-11-29" "0" "1"

"2020-11-30" "0" "2"



(ii) 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.

Query:

SELECT

ds,

AVG(time\_spent) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS rolling\_avg\_throughput

FROM job\_datas;

Output:

ds rolling\_avg\_throughput

"2020-11-25" "45"

"2020-11-26" "50.5"

"2020-11-27" "68.33333333333333"

"2020-11-28" "56.75"

"2020-11-28" "47.6"

"2020-11-29" "43"

"2020-11-30" "39"

"2020-11-30" "36.142857142857146"

Question:

explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

Answer:

I prefer daily metrics for immediate insights and responsiveness, while the 7-day rolling average provides a more stable view of overall performance.

(iii) 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.

Query:

SELECT language,

ROUND((COUNT(\*) \* 100.0) / (SELECT COUNT(\*) FROM job\_datas WHERE ds BETWEEN '2020-11-25' AND '2020-11-30'), 2) AS percentage\_share

FROM job\_datas

WHERE ds BETWEEN '2020-11-25' AND '2020-11-30'

GROUP BY language

ORDER BY percentage\_share DESC;

Output:

Language percentage share

"persian" "37.5"

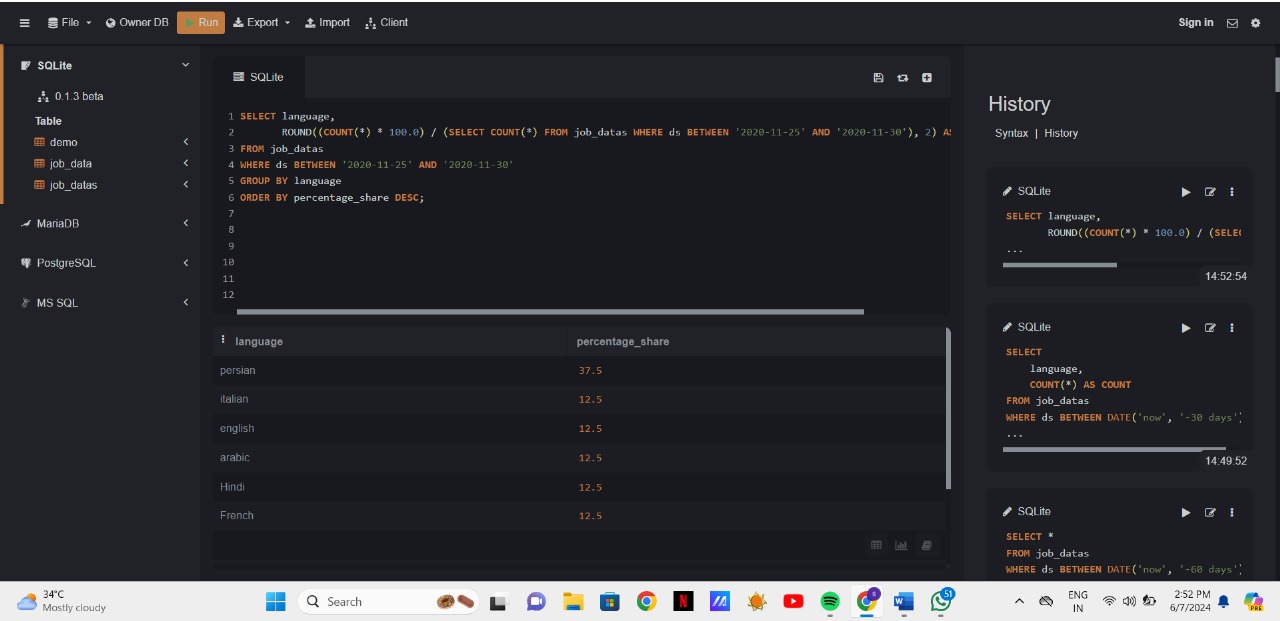
"italian" "12.5"

"english" "12.5"

"arabic" "12.5"

"Hindi" "12.5"

"French" "12.5"



(iv)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.

Query:

SELECT \*

FROM job\_datas

WHERE (job\_id, actor\_id, event, language, time\_spent, org, ds) IN (

SELECT job\_id, actor\_id, event, language, time\_spent, org, ds

FROM job\_data

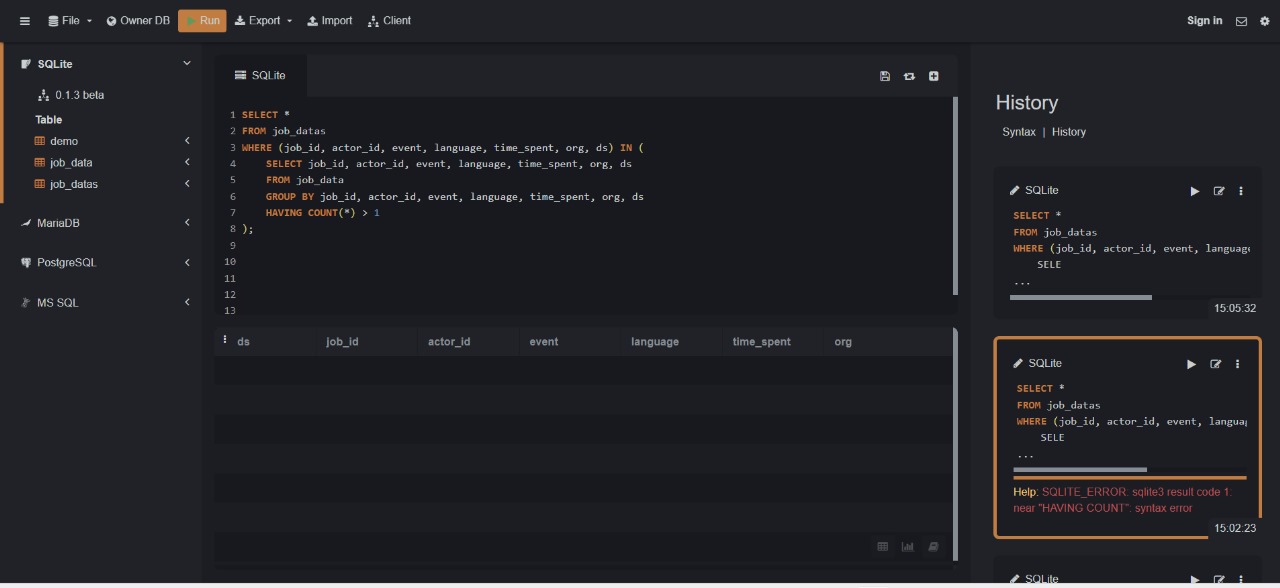
GROUP BY job\_id, actor\_id, event, language, time\_spent, org, ds

HAVING COUNT(\*) > 1

);

Output:

No duplicated rows found.



CASE STUDY2: Investigating Metric Spike

Create a table named users, including columns user\_id, created\_at, company\_id, language,activated\_at, state.

USERS TABLE:

Input:

CREATE TABLE users(

user\_id INT,

created\_at TIMESTAMP,

company\_id INT,

language VARCHAR(255),

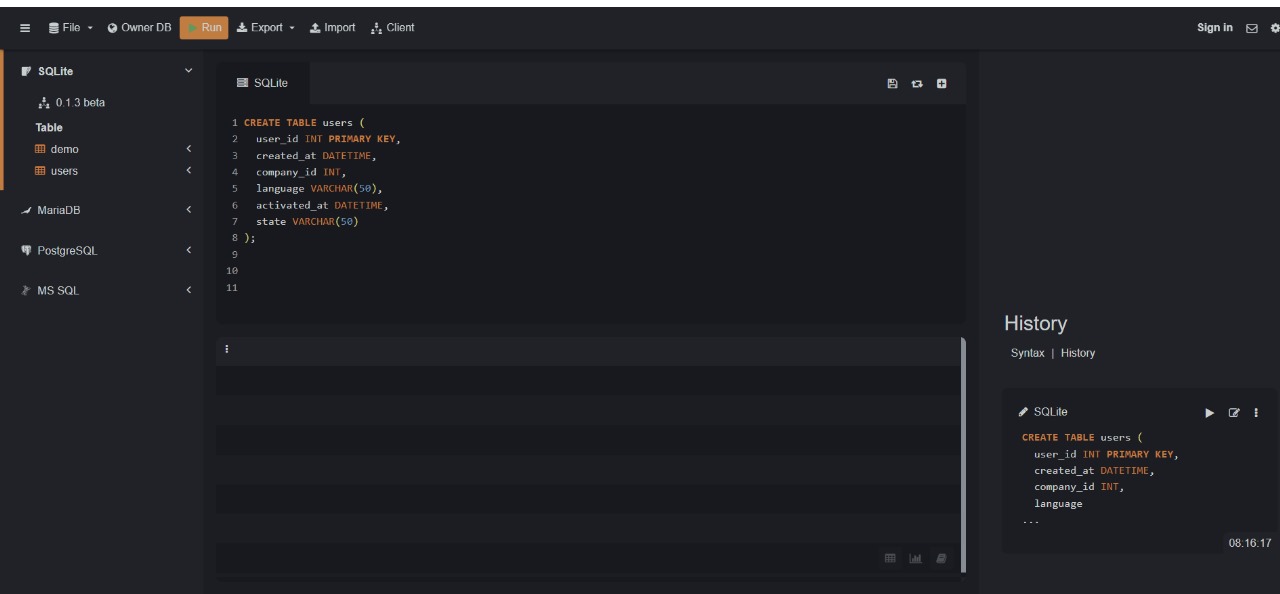
activated\_at TIMESTAMP,

state VARCHAR(50)

);

Output:

Successfully table created.



Insertion of values:

SELECT \* FROM "users (2)"

Output:

19039

31-08-2014 16:10

5815

english

31-08-2014 16:11

active

19040

31-08-2014 11:08

1444

spanish

31-08-2014 11:09

active

19042

31-08-2014 11:46

2121

arabic

31-08-2014 11:47

active

19043

31-08-2014 19:26

8950

english

31-08-2014 19:27

active

19044

31-08-2014 12:14

113

english

31-08-2014 12:15

active

19045

31-08-2014 17:06

4687

english

31-08-2014 17:07

active

19047

31-08-2014 07:35

3

english

31-08-2014 07:36

active

19048

31-08-2014 17:47

4381

portugese

31-08-2014 17:49

active

19049

31-08-2014 11:07

85

spanish

31-08-2014 11:09

active

19050

31-08-2014 10:22

15

english

31-08-2014 10:24

active

19053

31-08-2014 17:16

3225

japanese

31-08-2014 17:17

active

19055

31-08-2014 20:21

9073

french

31-08-2014 20:22

active

19056

31-08-2014 16:35

1234

english

31-08-2014 16:36

active

19059

31-08-2014 19:51

6817

indian

31-08-2014 19:53

active

19060

31-08-2014 12:19

1439

japanese

31-08-2014 12:20

active

19061

31-08-2014 13:21

2156

chinese

31-08-2014 13:22

active

19063

31-08-2014 07:10

72

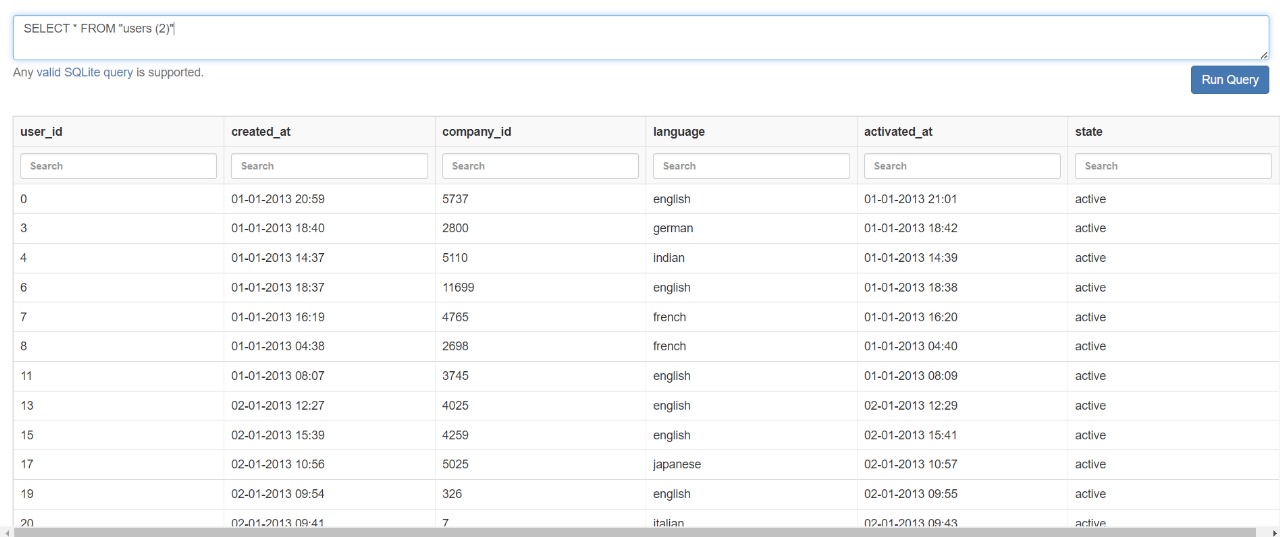
spanish

31-08-2014 07:12

Active

Output:

Successfully inserted values.



EVENTS TABLE:

Input:

CREATE TABLE events (

user\_id INT NOT NULL,

occurred\_at TIMESTAMP NOT NULL,

event\_type VARCHAR(255) NOT NULL,

event\_name VARCHAR(255) NOT NULL,

location VARCHAR(255) NOT NULL,

device VARCHAR(255) NOT NULL,

user\_type VARCHAR(255) NOT NULL,

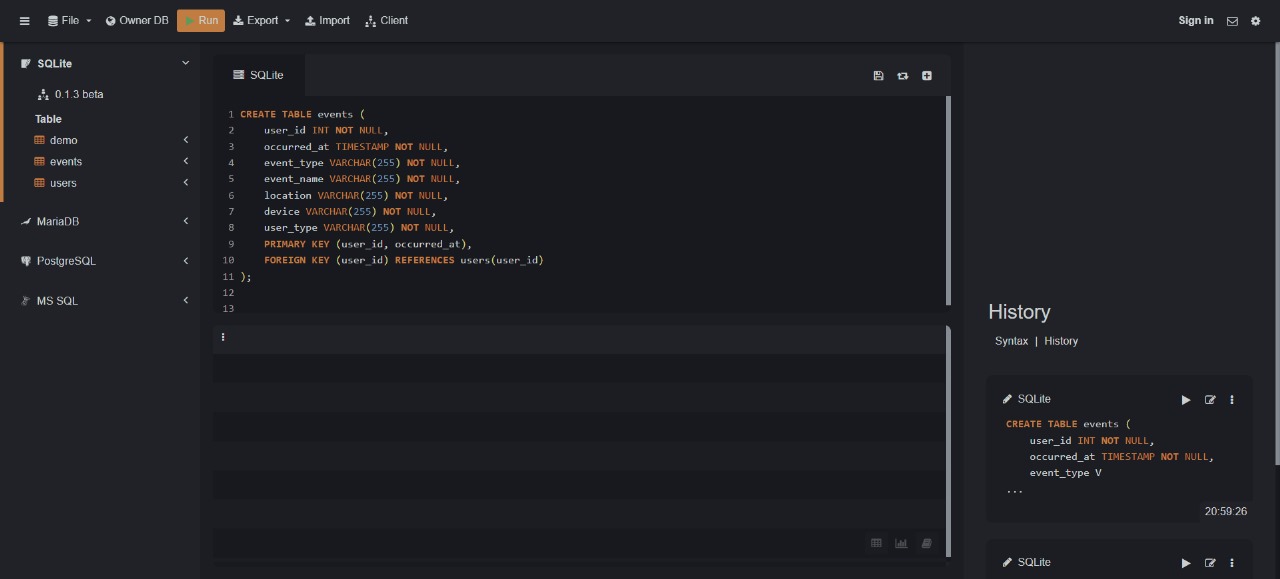
PRIMARY KEY (user\_id, occurred\_at),

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

Output:

Table created successfully



Insertion of values:

Import the dataset into SQL database

Input:

Import events.csv

Output:

Values inserted into table

Input:

Select \* from events;

Output:

16304

21-07-2014 01:54

engagement

view\_inbox

Saudi Arabia

asus chromebook

2

16304

21-07-2014 01:55

engagement

like\_message

Saudi Arabia

asus chromebook

2

16304

21-07-2014 01:55

engagement

home\_page

Saudi Arabia

asus chromebook

2

16304

21-07-2014 01:56

engagement

view\_inbox

Saudi Arabia

asus chromebook

2

16304

21-07-2014 01:56

engagement

send\_message

Saudi Arabia

asus chromebook

2

16304

21-07-2014 01:57

engagement

home\_page

Saudi Arabia

asus chromebook

2

16304

21-07-2014 01:57

engagement

view\_inbox

Saudi Arabia

asus chromebook

2

16304

21-07-2014 01:58

engagement

like\_message

Saudi Arabia

asus chromebook

2

16304

24-07-2014 13:22

engagement

login

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:22

engagement

home\_page

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:23

engagement

view\_inbox

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:23

engagement

home\_page

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:24

engagement

send\_message

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:24

engagement

home\_page

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:25

engagement

view\_inbox

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:25

engagement

home\_page

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:26

engagement

send\_message

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:26

engagement

home\_page

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:27

engagement

view\_inbox

Saudi Arabia

macbook pro

2

16304

24-07-2014 13:27

engagement

home\_page

Saudi Arabia

macbook pro

2

16306

21-07-2014 10:56

signup\_flow

complete\_signup

India

macbook air

3

16306

21-07-2014 10:56

engagement

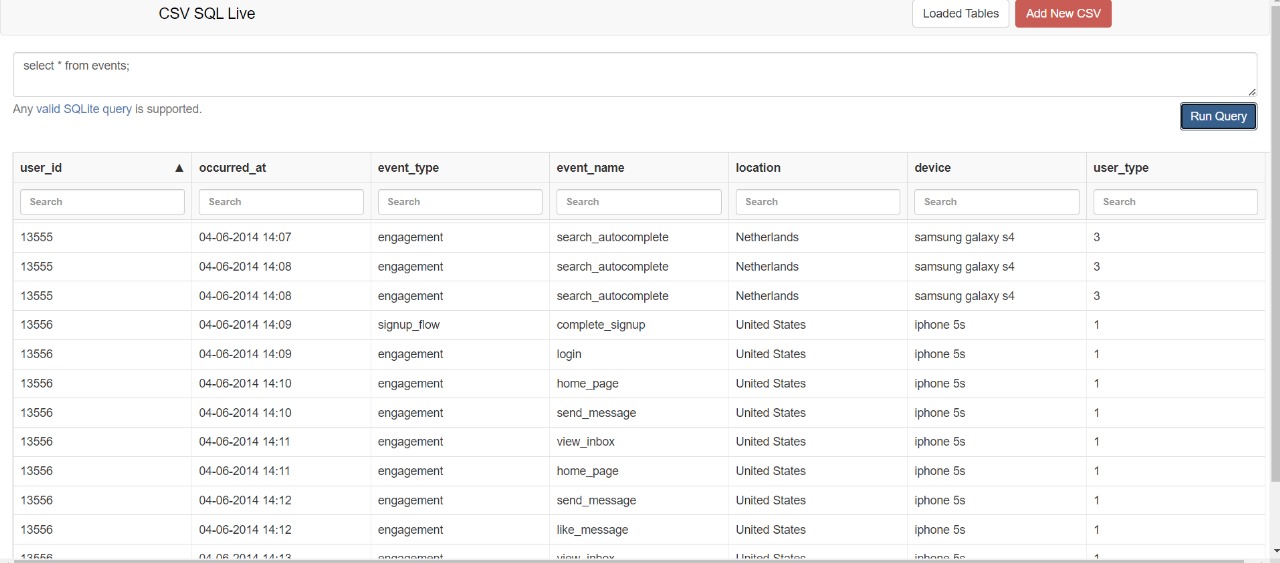
login

India

macbook air

3

Output: Values inserted into table



Email\_events TABLE:

Input:

CREATE TABLE Email\_events (

user\_id INT PRIMARY KEY,

occurred\_at TIMESTAMP NOT NULL,

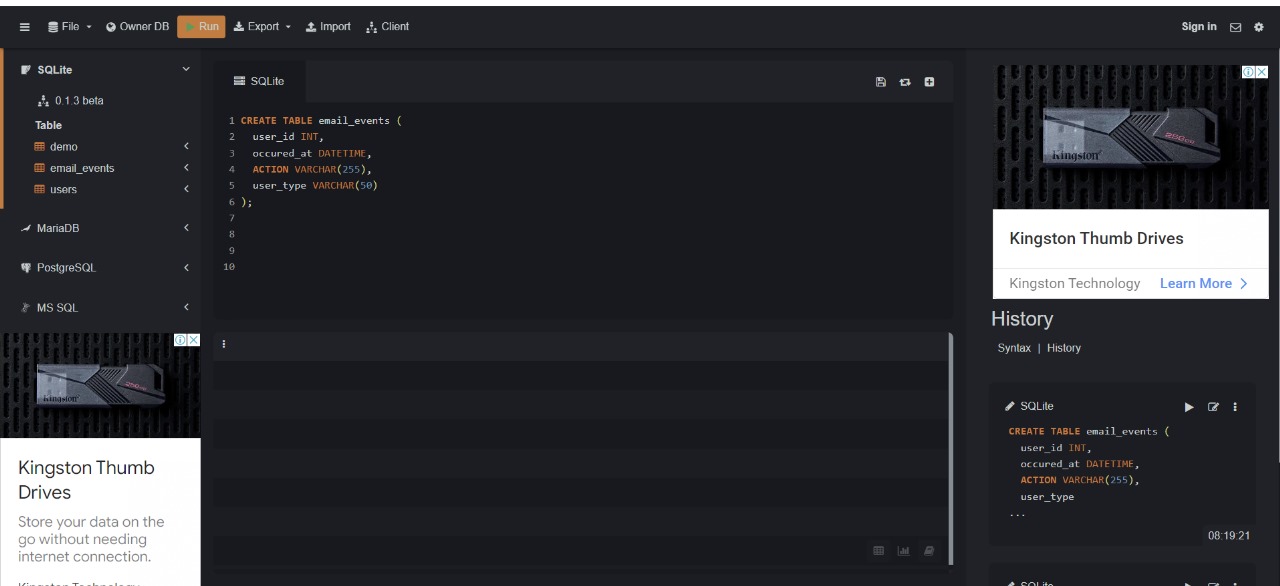
action VARCHAR(255) NOT NULL,

user\_type varchar(100)

);

Output:

Created table successfully.



Input:

Import csv file.

Output:

Values inserted into table.

Input:

Select \* from email\_events;

Output:

5352

15-05-2014 09:30

sent\_weekly\_digest

1

5352

15-05-2014 09:30

email\_open

1

5352

22-05-2014 09:30

sent\_weekly\_digest

1

5352

29-05-2014 09:30

sent\_weekly\_digest

1

5352

05-06-2014 09:30

sent\_weekly\_digest

1

5352

12-06-2014 09:30

sent\_weekly\_digest

1

5352

12-06-2014 09:30

email\_open

1

5352

19-06-2014 09:30

sent\_weekly\_digest

1

5352

26-06-2014 09:30

sent\_weekly\_digest

1

5352

03-07-2014 09:30

sent\_weekly\_digest

1

5352

10-07-2014 09:30

sent\_weekly\_digest

1

5352

17-07-2014 09:30

sent\_weekly\_digest

1

5352

24-07-2014 09:30

sent\_weekly\_digest

1

5352

31-07-2014 09:30

sent\_weekly\_digest

1

5352

31-07-2014 09:30

email\_open

1

5352

31-07-2014 09:30

email\_clickthrough

1

5352

07-08-2014 09:30

sent\_weekly\_digest

1

5352

07-08-2014 09:30

email\_open

1

5352

14-08-2014 09:30

sent\_weekly\_digest

1

5352

14-08-2014 09:30

email\_open

1

5352

14-08-2014 09:30

email\_clickthrough

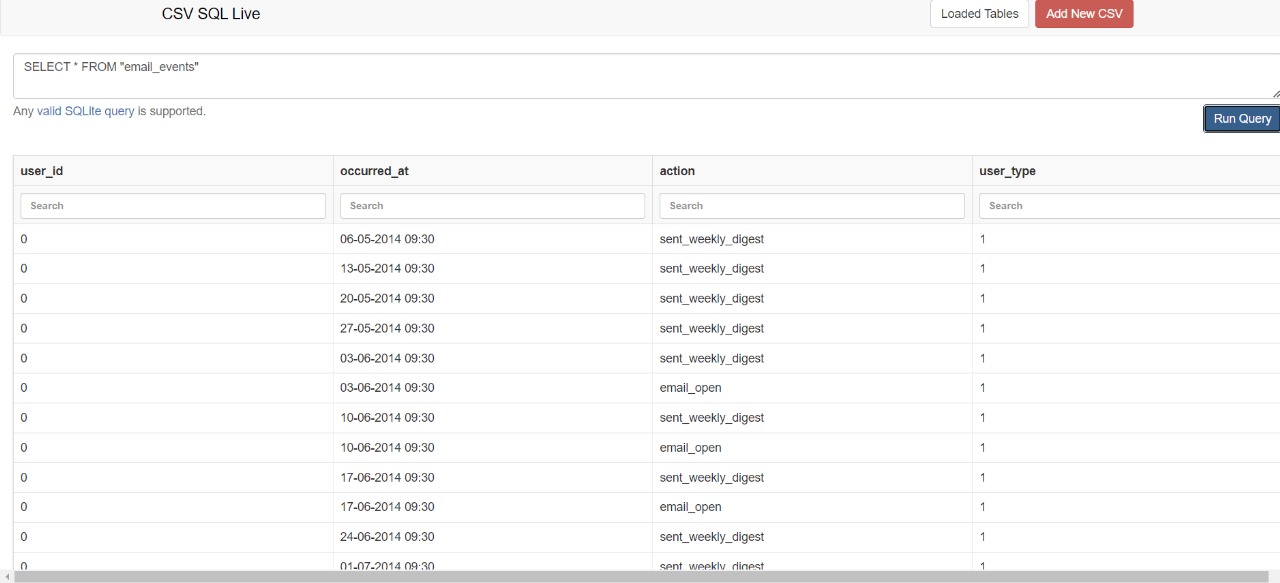
1

5352

21-08-2014 09:30

sent\_weekly\_digest

1



(i)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.

Query:

SELECT

strftime('%W', created\_at) AS weekly,

COUNT(user\_id) AS num\_users

FROM

users

WHERE

state = 'active'

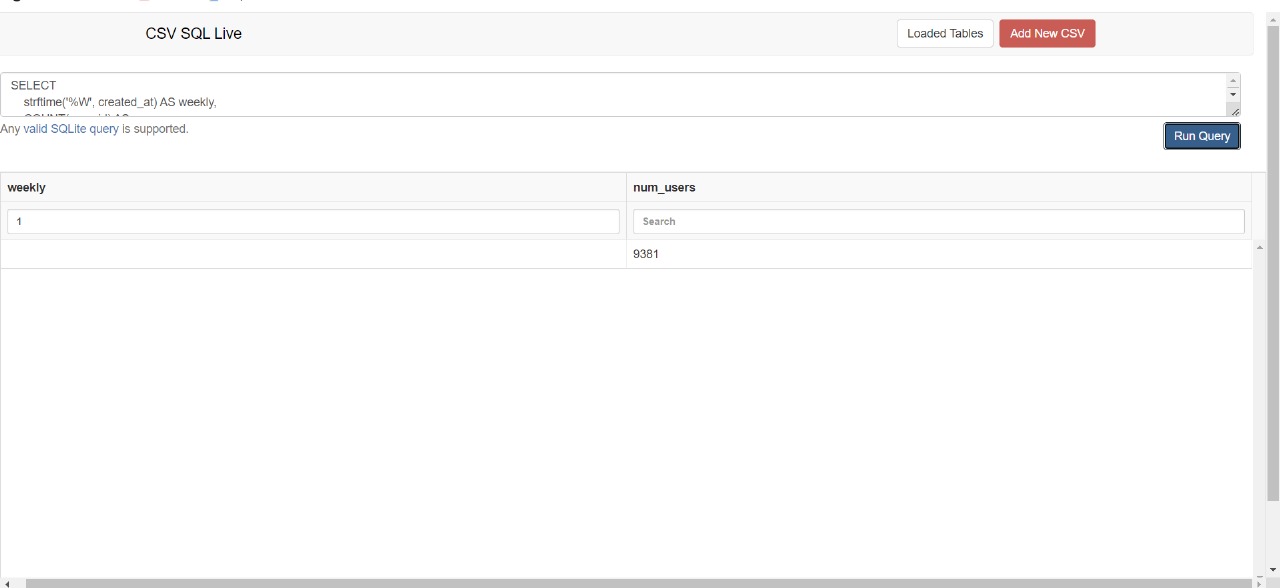
GROUP BY

weekly;

Output:

Num\_users:

9381



(ii)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.

Query:

SELECT

DATE(u.created\_at) AS date,

e.event\_name,

COUNT(DISTINCT u.user\_id) AS user\_count

FROM

users u

JOIN events e ON u.user\_id = e.user\_id

WHERE

u.state = 'active'

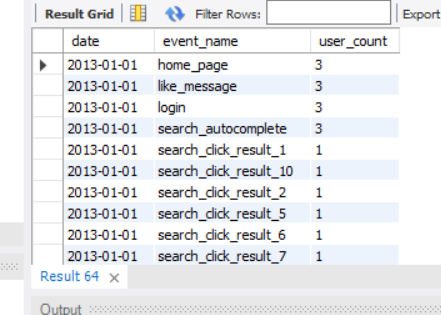
GROUP BY

DATE(u.created\_at), e.event\_name

ORDER BY

date ASC;

Output:



(iii)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.

Query:

select weekofyear(u.created\_at) as week , count(e.user\_id) as user\_retained

from users u

left join

events e

on

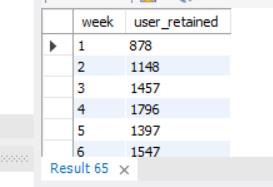
u.user\_id =e.user\_id

where u.state = 'active'

group by 1

order by 1;

Output:



(iv)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.

Input:

select weekofyear(u.created\_at) as weekly , e.device , count(u.user\_id) as users

from events e

right join

users u

on

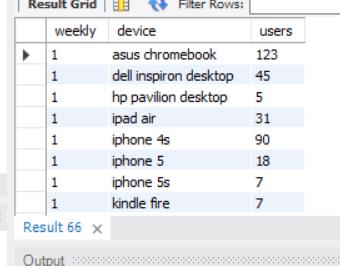
e.user\_id = u.user\_id

where e.event\_type = 'engagement'

group by 1,2

order by 1;

Output:



The number of rows are 572

(v)Email Engagement: Users engaging with the email service.

Your task: Calculate the email engagement metrics?

Input:

SELECT u.user\_id ,

count(case when ee.action = 'sent\_weekly\_digest' then 1 else null end) as

num\_sent\_weekly\_digest ,

count(case when ee.action = 'email\_open' then 1 else null end) as num\_email\_open ,

count(case when ee.action = 'email\_clickthrough' then 1 else null end) as

num\_email\_clickthrough ,

count(case when ee.action = 'sent\_reengagement\_email' then 1 else null end) as

num\_sent\_reengagement\_email

FROM users u

left join

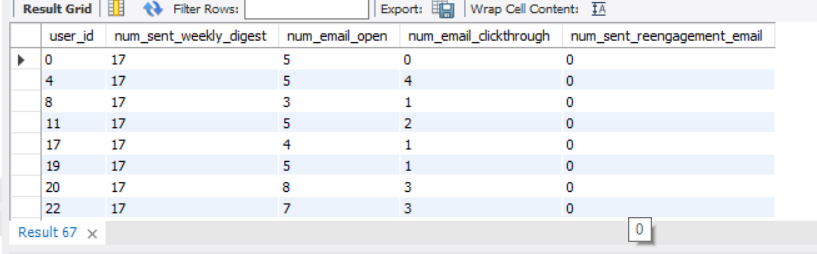
email\_events ee

on

u.user\_id = ee.user\_id

group by 1 ;

Output:



The total number of rows 19066

**INSIGHTS:**

Case Study 1: Job Data Analysis

Jobs Reviewed Over Time:

Insight: Peak review hours can vary by day, impacting workload distribution.

Actionable Implication: Allocate resources efficiently during high-review hours.

Throughput Analysis:

Insight: The 7-day rolling average smooths out daily variations.

Actionable Implication: prefer daily metrics for immediate insights and responsiveness, while the 7-day rolling average provides a more stable view of overall performance.

Language Share Analysis:

Insight: Language preferences impact user experience.

Actionable Implication: create content based on language distribution.

Duplicate Rows Detection:

Insight: Duplicate rows affect data accuracy.

Actionable Implication: Address duplicates to ensure reliable analysis and security .

Case Study 2: Investigating Metric Spike

Weekly User Engagement:

Insight: Weekly engagement trends reveal user behavior patterns.

Actionable Implication: Adapt strategies based on active user patterns.

User Growth Analysis:

Insight: User acquisition trends inform marketing efforts.

Actionable Implication: Optimize acquisition channels according to market trends.

Weekly Retention Analysis:

Insight: Post-signup retention impacts long-term success.

Actionable Implication: Focus on retaining users beyond sign-up.

Weekly Engagement Per Device:

Insight: Device-specific engagement guides feature enhancements.

Actionable Implication: Optimize platform features for each device.

Email Engagement Analysis:

Insight: Email metrics gauge user interest and campaign effectiveness.

Actionable Implication: Optimize email content and delivery strategies.

**TECH-STACK USED:**

I have used SQL lite to do the queries

SQLite is a self-contained, serverless, and file-based database system.

It doesn’t require a separate server process or installation, making it lightweight and easy to embed within applications.

SQLite databases are created as files on disk. No complex setup or configuration is needed.

SQLite performs well for read-heavy workloads (SELECT queries).