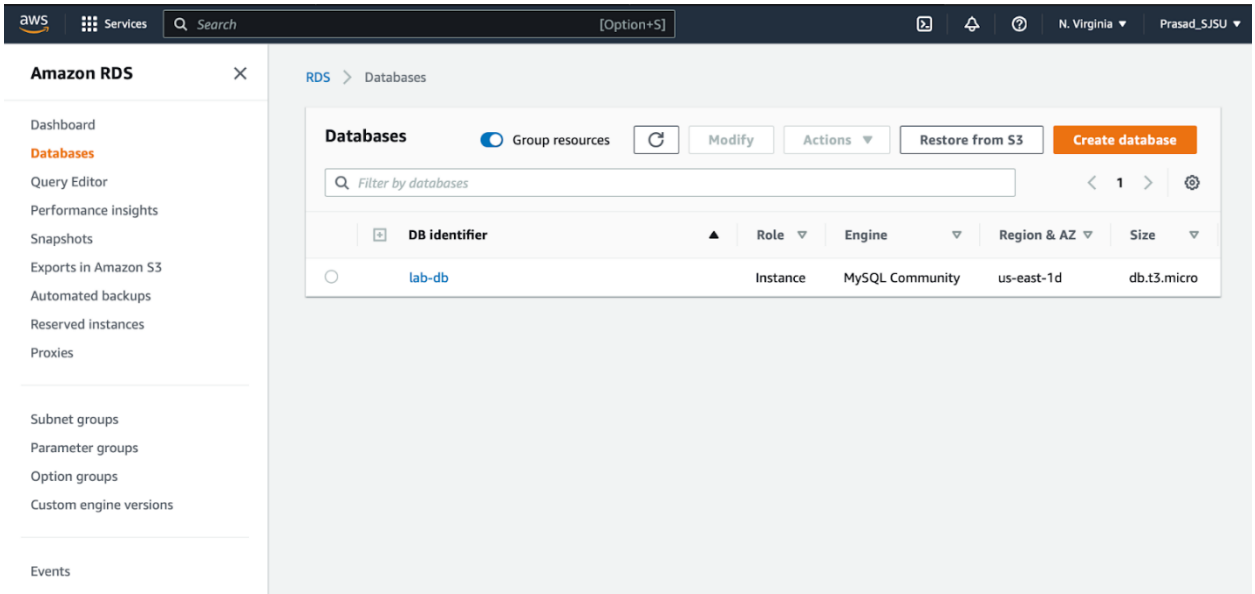
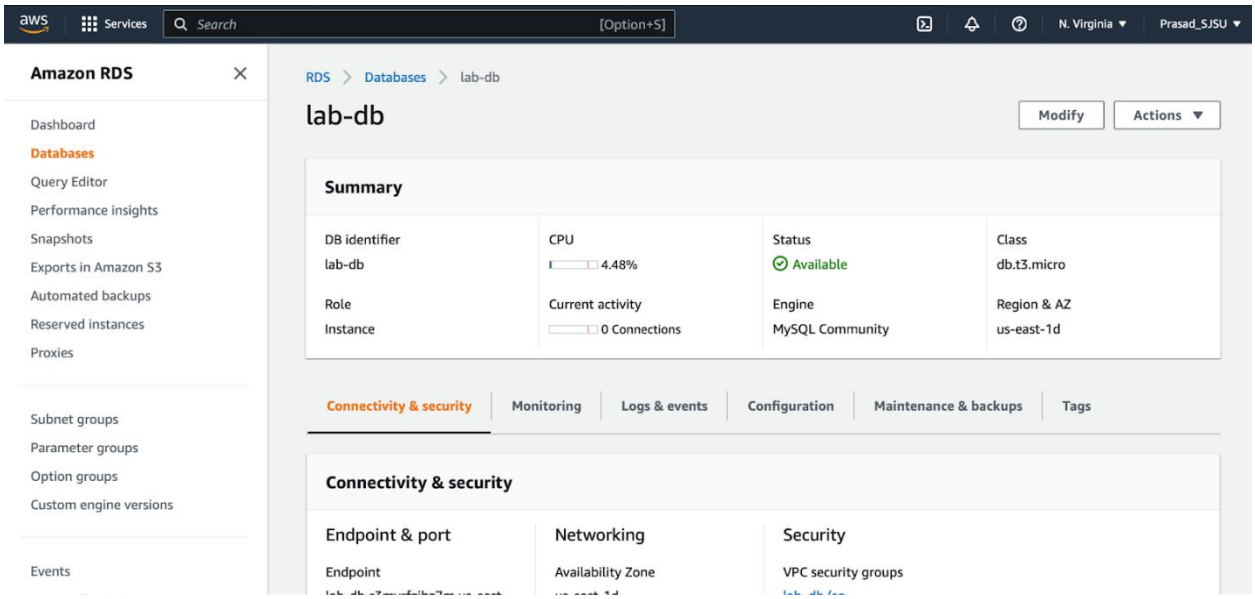
IV. RESULTS

A. AWS Instance DB Connection





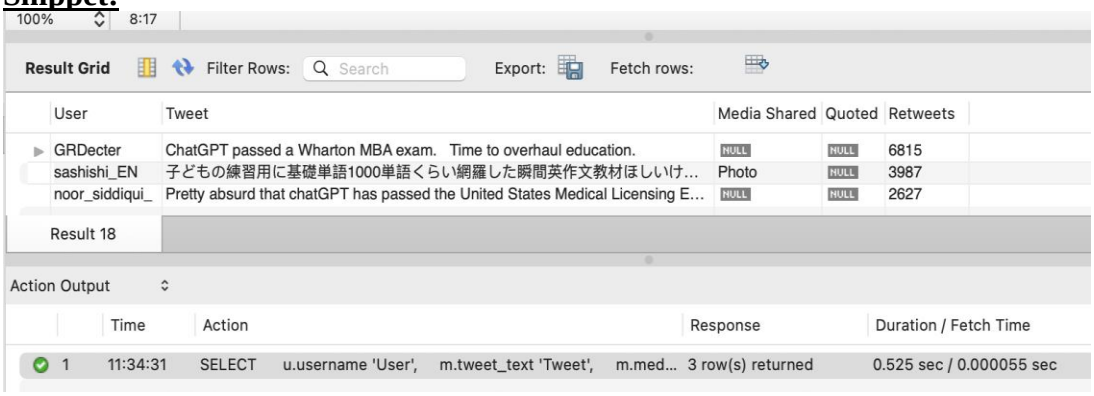
B. General SQL Queries

1. 1st Query: **Top 3 Viral tweets and its details based on retweet counts**.

Query:

```
SELECT
   u.username 'User',
   m.tweet_text 'Tweet',
   m.media_type 'Media Shared',
   m.quoted_tweet 'Quoted',
   c.retweet_count 'Retweets'
FROM
   tweet t
      JOIN
   user u USING (user_id)
      JOIN
   count c USING (tweet_id)
      JOIN
   media m USING (tweet_id)
GROUP BY tweet_id
ORDER BY retweets DESC
LIMIT 3
```

Snippet:



This SQL query retrieves the top three viral tweets based on the number of retweets. Using the 'tweet_id' attribute, it joins four tables: the 'tweet', the 'user', the 'count', and the 'media'. The 'user' table stores user information, while the 'tweet' table stores tweet text. Retweet counts are listed in our count table; media is listed in our media table.

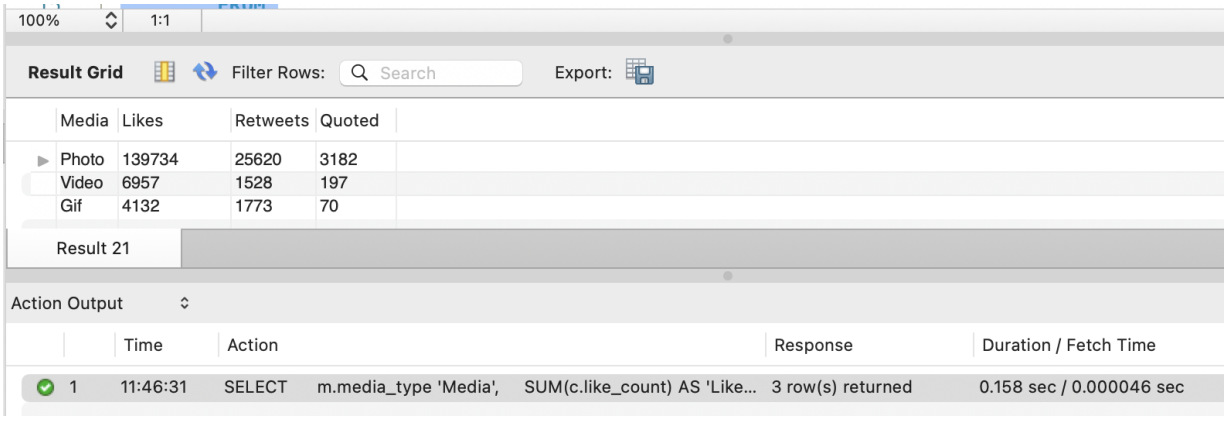
Among the results, the tweet by GRDecter with the text ‘ChatGPT passed a Wharton MBA exam. Time to overhaul education’ was the most viral tweet with the highest retweets of 6815.

2. **2nd Query: Most favorable media (photo/video/gif) used in chatgpt tweets based on likes, retweets and quotes.**

Query:

```
SELECT
   m.media_type 'Media',
   SUM(c.like_count) AS 'Likes',
   SUM(c.retweet_count) AS 'Retweets',
   SUM(c.quote_count) AS 'Quoted'
FROM
   tweet
      JOIN
   media m USING (tweet_id)
      JOIN
   count c USING (tweet_id)
WHERE
   m.media_type = ANY (SELECT DISTINCT
         media_type
      FROM
         media
      WHERE
         media_type IS NOT NULL)
GROUP BY m.media_type
ORDER BY Likes DESC , Retweets DESC , Quoted DESC
```

Snippet:



Media	Likes	Retweets	Quoted
Photo	139734	25620	3182
Video	6957	1528	197
Gif	4132	1773	70

Observation: Based on the results, we can see that 'Photo' media types received the highest number of likes, retweets, and quotes.

3. **3rd Query: Most shared material / website / platform discussed about chatgpt (using outlink):**

Query:

```
SELECT
   outlinks 'Content', COUNT(outlinks) AS 'Count'
FROM
   outlinks
WHERE
   outlinks NOT LIKE '%twitter.com%'
GROUP BY outlinks
ORDER BY count DESC
LIMIT 3
```

Snippet:

Content	Count
https://www.ft.com/content/7229ba86-142a-49f6…	156
https://fortune.com/2023/01/21/chatgpt-passed-…	88
http://twinybots.ch	87

Result 29

Action Output

	Time	Action	Response	Duration / Fetch Time
1	11:56:21	SELECT outlinks 'Content', COUNT(outlinks) AS 'Count' FRO…	3 row(s) returned	0.173 sec / 0.000036 sec

Observation: Based on the results, 'https://www.ft.com/content/7229ba86-142a-49f6-9821-f55c07536b7c' was the most shared article regarding chat GPT on the financial times website.

4. 4th Query: The top 5 languages based on tweets about chatgpt.

Query:

```
SELECT
   language 'Language', COUNT(language) AS '# of Tweets'
FROM
   tweet
GROUP BY language
ORDER BY COUNT(language) DESC
LIMIT 5
```

Snippet:

Language	# of Tweets
en	32076
ja	5046
es	3315
fr	2492
de	1207

Result 40

Action Output

	Time	Action	Response
1	12:15:57	SELECT language 'Language', COUNT(language) AS '# of Tweets' FROM tweet GROUP BY...	5 row(s) returned

Observation: Although English was the dominant language, Japanese and Spanish spoke more about chatgpt. The fourth and fifth languages discussed about chatgpt were French and German.

5. **5th Query: Top 10 hashtags used for chatgpt tweets and their counts from the dataset.**

Query:

```
SELECT
   hashtag 'Hashtag', COUNT(hashtag) AS 'Count'
FROM
   hashtag
GROUP BY hashtag
ORDER BY COUNT(hashtag) DESC
LIMIT 10
```

Snippet:

Hashtag	Count
#chatgpt	9612
#ai	2520
#openai	1074
#artificialintelligence	724
#microsoft	510
#technology	304
#ia	287
#google	284
#machinelearning	257
#chatgpt.	237

Result 50

Action Output

	Time	Action	Response	Duration / Fetch Time
1	12:30:18	SELECT hashtag 'Hashtag', COUNT(hashtag) AS 'Count' F...	10 row(s) returned	0.120 sec / 0.000023 sec

Observation: The query results reveal that, in addition to the hashtag '#chatgpt', the hashtags '#ai' and '#openai' are the next most frequently used hashtags in the list.

6. **6th Query: Information about the longest tweet text about chatgpt.**

Query:

```
SELECT
    u.username 'User',
    tweet_text 'Tweet',
    t.language 'Language',
    t.source 'Device',
    MAX(LENGTH(tweet_text)) 'Length'
FROM
    media m
        JOIN
    tweet t USING (tweet_id)
        JOIN
    user u USING (user_id)
```

Snippet:

User	Tweet	Language	Device	Length
mochico0123	ChatGPTで遊ぶの忘れてた！！ 書類作るコード書いてみてほしいのと、 どこまで思考整理付き合…	ja	Twitter for iPhone	737

Result 58

Action Output

	Time	Action	Response	Duration / Fetch Time
1	12:39:55	SELECT u.username 'User', tweet_text 'Tweet', t.langu…	1 row(s) returned	0.299 sec / 0.000047 sec

Observation: The query results indicate that the longest tweet, posted from an iPhone by the user 'mochico0123', contained 737 letters.

7. **7th Query: Tweet lengths of different languages showing the minimum, maximum and average text length for each language.**

Query:

```
SELECT
    t.language 'Language',
    MIN(LENGTH(m.tweet_text)) 'Minimum',
    MAX(LENGTH(m.tweet_text)) 'Maximum',
    AVG(LENGTH(m.tweet_text)) 'Average'
FROM
    media m
JOIN
    tweet t ON t.tweet_id = m.tweet_id
GROUP BY t.language
```

```
ORDER BY AVG(LENGTH(m.tweet_text)) DESC
```

Snippet:

Language	Minimum	Maximum	Average
ta	148	629	389.5000
mr	54	655	383.3750
te	261	458	359.5000
am	337	337	337.0000
gu	328	328	328.0000
th	45	737	301.8497
hy	294	294	294.0000
ml	95	631	293.6000
kn	159	521	261.6667
ar	41	466	256.7857
fa	35	459	251.4920
pa	176	350	251.3333

Result 64

Action Output

	Time	Action			Response	Duration / Fetch Time
1	12:53:05	SELECT	t.language 'Language',	MIN(LENGTH(m.tweet_text)) '...	61 row(s) returned	0.343 sec / 0.000081 sec

Observation: The query results provide information on the average, minimum, and maximum tweet lengths for 61 different languages used to tweet. Among these languages, it is noteworthy that the 'Tamil language' has the highest average tweet length compared to all other languages.

8. **8th Query: Total tweets about chatgpt on each day from the dataset.**

Query:

```
SELECT
    DATE_FORMAT(timestamp, '%d-%m-%Y') 'Date',
    COUNT(DATE_FORMAT(timestamp, '%d-%m-%Y')) '# of tweets'
FROM
    tweet
GROUP BY DATE_FORMAT(timestamp, '%d-%m-%Y')
ORDER BY DATE_FORMAT(timestamp, '%d-%m-%Y')
```

Snippet:

Result Grid Filter Rows: Search Export:

Date	# of tweets
22-01-2023	10068
23-01-2023	31700
24-01-2023	8233

Result 72

Action Output

	Time	Action	Response
1	13:03:40	SELECT DATE_FORMAT(timestamp, '%d-%m-%Y') 'Date', COU...	3 row(s) returned

<u>Observation:</u> According to the query results, the day of January 23, 2023, had the highest number of tweets. It is notable that this spike in activity may be due to the announcement that 'Microsoft plans' to invest $10 billion in OpenAI, the creator of ChatGPT.

9. **9th Query: The top 10 most mentioned twitter users in chatgpt related tweets.**

<u>Query:</u>

```
SELECT
    m.mentioned_user 'User',
    u.url 'Twitter URL',
COUNT(m.mentioned_user) 'Mentions'
    FROM
mentioned_users m
    LEFT JOIN
user u ON m.mentioned_user = u.username
GROUP BY mentioned_user
ORDER BY mentions DESC
    LIMIT 10;
```

Snippet:

User	Twitter URL	Mentions
openai	NULL	737
grdecter	https://twitter.com/GRDecter	495
elonmusk	NULL	355
youtube	NULL	308
chatgpt_issac	https://twitter.com/chatgpt_issac	195
microsoft	NULL	173
noor_siddiqui_	https://twitter.com/noor_siddiqui_	158
chatgpt	NULL	150
sama	NULL	139

Result 32

Action Output

	Time	Action	Response	Duration / Fetch Time
1	14:52:14	SELECT m.mentioned_user 'User', u.url 'Twitter URL', COUN...	10 row(s) returned	0.217 sec / 0.000024 sec

Observation: Based on the query results, it is clear that OpenAI is the most frequently mentioned user in the dataset, which may be due to their significant presence in the field of artificial intelligence. It is worth noting that the Twitter URL is only displayed for users who have tweeted within the given time period covered by the dataset.

10. **10th Query: Hashtag statistics - The minimum, maximum and average number of hashtags used in all the tweets from the dataset**

Query:

```
SELECT
    MIN(h.h_cnt) 'Min', MAX(h.h_cnt) 'Max', AVG(h.h_cnt) 'Avg'
FROM
    (SELECT COUNT(hashtag) h_cnt
FROM
    tweet t
LEFT JOIN hashtag h USING (tweet_id)
GROUP BY tweet_id) AS h
```

Snippet:

Result Grid Filter Rows: Search Export:

Min	Max	Avg
0	28	0.7765

Result 27

Action Output

	Time	Action	Response
1	16:28:09	SELECT MIN(h.h_cnt) 'Min', MAX(h.h_cnt) 'Max', AVG(h.h_cnt) 'Av...	1 row(s) returned

Observation: The query calculates the minimum, maximum, and average number of hashtags used in tweets from the dataset. The result shows that the minimum number of hashtags used is 0, the maximum is 28, and the average number is approximately 0.77.

11. 11th query: The top 5 most preferred sources of twitter to tweet during a time period from the dataset.

Query:

```
SELECT
    Source, COUNT(source) 'Count'
FROM
    tweet
GROUP BY source
ORDER BY COUNT(source) DESC
LIMIT 5;
```

Snippet:

Result Grid Filter Rows: Search Export: Fetch rows:

Source	Count
Twitter Web App	17814
Twitter for iPhone	12281
Twitter for Android	8972
IFTTT	1383
dlvr.it	959

Result 10

Action Output

	Time	Action		Response
1	16:36:47	SELECT	Source, COUNT(source) 'Count' FROM tweet GROUP...	5 row(s) returned

Observation: The query results reveal the top sources that were used to tweet, which include Web, iPhone, and Android. These results suggest that there are more users who are actively using web applications to tweet more than smartphone applications.

12. 12th Query: The top 5 most quoted tweets about chatgpt,by the user of the tweet and its text.

Query:

```
SELECT
      qt.qt_twt 'Quoted Tweet',
      qt.cnt 'Quoted Times',
       u.username 'User',
      m.tweet_text 'Tweet'
FROM
      (SELECT
      t.tweet_id AS id,
      t.user_id AS usr_id,
      m.quoted_tweet qt_twt,
      t.permalink perma,
      COUNT(quoted_tweet) cnt
 FROM
      media m
LEFT JOIN tweet t ON m.quoted_tweet = t.permalink
GROUP BY quoted_tweet
ORDER BY COUNT(quoted_tweet) DESC
LIMIT 10) AS qt
   LEFT JOIN
media m ON qt.id = m.tweet_id
   LEFT JOIN
user u ON qt.usr_id = u.user_id
LIMIT 5;
```

Snippet:

Result Grid Filter Rows: Search Export: Fetch rows:

Quoted Tweet	Quoted Times	User	Tweet
https://twitter.com/GRDecter/status/16171623...	209	GRDecter	ChatGPT passed a Wharton MBA exam. Time to overhaul education.
https://twitter.com/noor_siddiqui_/status/1617...	141	noor_siddiqui_	Pretty absurd that chatGPT has passed the United States Medical Licensing Examination (US...
https://twitter.com/WatcherGuru/status/16173...	99	WatcherGuru	ChatGPT, an artificial intelligence search tool, has passed the United States Medical Licensing...
https://twitter.com/GRDecter/status/16177157...	70	GRDecter	ChatGPT has passed: - United States medical license exam - The Bar Exam - MBA operatio...
https://twitter.com/nabeelqu/status/16168453...	69	NULL	NULL

Result 26

Action Output

	Time	Action	Response
1	17:13:03	SELECT qt.qt_twt 'Quoted Tweet', qt.cnt 'Quoted Times', u.u...	5 row(s) returned

Observation: The result shows that the tweet by GRDecter - ‘ChatGPT passed a Wharton MBA exam. Time to overhaul education.‘ has been the most quoted text within the dataset.

13. 13th Query: Peak hours of activity.

Query:

```
SELECT
   timestamp, COUNT(tweet_id) AS Tweets
FROM
   tweet
GROUP BY HOUR(timestamp)
ORDER BY COUNT(tweet_id) DESC
LIMIT 10
```

Snippet:

timestamp	Tweets
2023-01-22 15:00:01	3297
2023-01-22 16:00:00	3237
2023-01-22 14:00:00	3178
2023-01-22 17:00:00	2927
2023-01-22 18:00:01	2756
2023-01-22 19:00:00	2604
2023-01-22 20:00:00	2370
2023-01-22 21:00:01	2330
2023-01-22 22:00:01	2158
2023-01-23 06:00:00	2140

Result 4

Action Output

	Time	Action		Response	Duration / Fetch Time
1	19:20:59	SELECT	timestamp, COUNT(tweet_id) AS Tweets FROM tweet...	10 row(s) returned	0.240 sec / 0.000023 sec

Observation: The query results provide information on the peak hours of activity for tweets in the dataset. The results show the top 10 hours with the highest number of tweets, based on the count of tweet IDs. Highest is 3297 tweets at 15:00:01 on 22nd Jan of 2023.

C. Views:

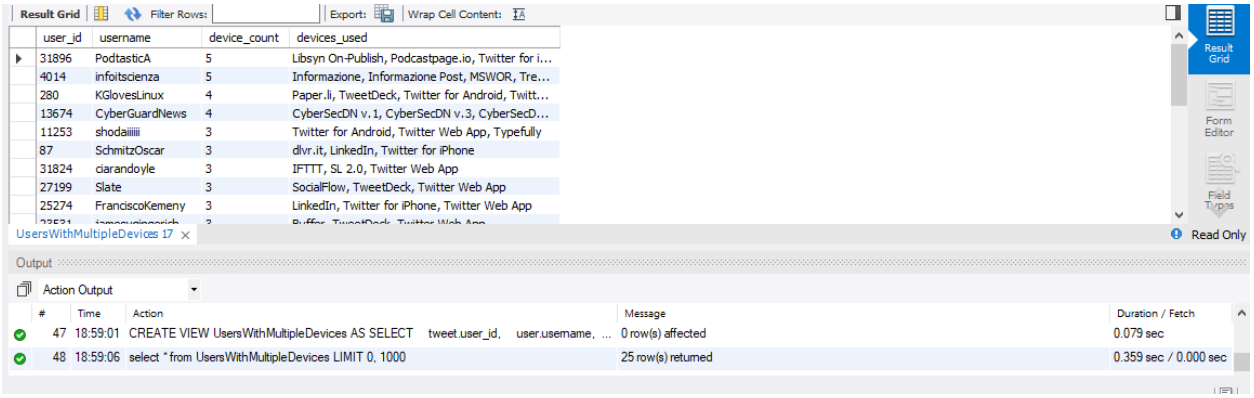
1. View that shows users who have atleast used 3 devices, along with the number of devices used and the names of the devices

Query:

```
CREATE VIEW UsersWithMultipleDevices AS
SELECT
   tweet.user_id,
   user.username,
   COUNT(DISTINCT tweet.source) AS device_count,
   GROUP_CONCAT(DISTINCT tweet.source ORDER BY tweet.source SEPARATOR ', ') AS
devices_used
FROM tweet
INNER JOIN user ON tweet.user_id = user.user_id
GROUP BY tweet.user_id
HAVING COUNT(DISTINCT tweet.source) >= 3
ORDER BY device_count DESC;
```

select * from UsersWithMultipleDevices

Snippet:



user_id	username	device_count	devices_used
31896	PodtasticA	5	Libsyn On-Publish, Podcastpage.io, Twitter for i...
4014	infoitscienza	5	Informazione, Informazione Post, MSWOR, Tre...
280	KGlovesLinux	4	Paper.li, TweetDeck, Twitter for Android, Twitt...
13674	CyberGuardNews	4	CyberSecDN v.1, CyberSecDN v.3, CyberSecD...
11253	shodaiiiiii	3	Twitter for Android, Twitter Web App, Typefully
87	SchmitzOscar	3	dlvr.it, LinkedIn, Twitter for iPhone
31824	ciarandoyle	3	IFTTT, SL 2.0, Twitter Web App
27199	Slate	3	SocialFlow, TweetDeck, Twitter Web App
25274	FranciscoKemeny	3	LinkedIn, Twitter for iPhone, Twitter Web App

#	Time	Action	Message	Duration / Fetch
47	18:59:01	CREATE VIEW UsersWithMultipleDevices AS SELECT tweet.user_id, user.username, ...	0 row(s) affected	0.079 sec
48	18:59:06	select * from UsersWithMultipleDevices LIMIT 0, 1000	25 row(s) returned	0.359 sec / 0.000 sec

Observation: The code creates a view called 'UsersWithMultipleDevices' that displays users who have used at least 3 devices, along with the number of devices used and the names of the devices. The query joins the 'tweet' and 'user' tables and groups the data by user ID to count the number of unique devices used by each user. The resulting view will only show users who meet the criteria and will be sorted based on the number of devices used.

D. Stored Procedures:

1. Stored Procedure to display the stat for the day

Query:

```
Delimiter **
CREATE PROCEDURE Stat(IN Input_date date)
BEGIN
SELECT
   DATE(tweet.timestamp),
   COUNT(DISTINCT (tweet.user_id)) AS 'Total Users for the day',
   COUNT(DISTINCT (tweet.tweet_id)) AS 'Total tweets for the day',
   COUNT(DISTINCT (hashtag.hashtag)) AS 'Total Hastags for the day'
FROM
   tweet
      JOIN
   hashtag ON hashtag.tweet_id = tweet.tweet_id
WHERE
   DATE(tweet.timestamp) = Input_date
GROUP BY DATE(tweet.timestamp)
ORDER BY COUNT(DISTINCT (tweet.tweet_id)) DESC , COUNT(DISTINCT (tweet.user_id))
, (hashtag.hashtag) DESC;
END **
Delimiter ;
```

call stat('2023-01-22')

Snippet:

Result Grid Filter Rows: Search Export:

DATE(tweet.timestamp)	Total Users for the day	Total tweets for the day	Total Hastags for the day
2023-01-23	7107	8991	6430

Result 9

Action Output

	Time	Action	Response	Duration / Fetch Time
1	18:09:58	call stat('2023-01-22')	1 row(s) returned	0.111 sec / 0.000042 sec
2	18:10:02	call stat('2023-01-23')	1 row(s) returned	0.254 sec / 0.000060 sec
3	18:10:07	call stat('2023-01-24')	1 row(s) returned	0.117 sec / 0.000062 sec
4	18:10:12	call stat('2023-01-25')	0 row(s) returned	0.107 sec / 0.000018 sec
5	18:10:17	call stat('2023-01-23')	1 row(s) returned	0.125 sec / 0.000053 sec

Observation: The result shows the statistics for the day '2023-01-22', including the total number of distinct users who tweeted, the total number of distinct tweets, and the total number of distinct hashtags used on that day. The results are sorted first by the total number of distinct tweets in descending order, then by the total number of distinct users in ascending order, and lastly by the hashtag in descending order.

2. Stored Procedures to have User Statistics.

Query:

```
Delimiter $$
Create Procedure user_stat(
IN uname VARCHAR(20)
)
BEGIN
SELECT
   u.username,
   COUNT(t.tweet_id) AS 'Tweets',
   SUM(c.reply_count) AS 'Replies',
   SUM(c.retweet_count) AS 'Retweets',
   SUM(c.like_count) AS 'Likes',
   SUM(c.quote_count) AS 'Times Quoted'
FROM
   user u
      JOIN
   tweet t USING (user_id)
      JOIN
   count c USING (tweet_id)
WHERE
   u.username = uname;
END $$
Delimiter ;
```

call user_stat('GRDecter')

Snippet:

Result Grid | Filter Rows: Search | Export:

username	Tweets	Replies	Retweets	Likes	Times Quoted
GRDecter	3	2484	9358	72183	2513

Result 16 | Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
1	18:27:57	call user_stat('GRDecter')	1 row(s) returned	0.086 sec / 0.000027 sec

Observation: The result shows the statistics for the user with the username 'GRDecter', including the total number of tweets made by the user, the total number of replies received by the user's tweets, the total number of retweets of the user's tweets, the total number of likes received by the user's tweets, and the total number of times the user's tweets were quoted.

3. Stored procedure to check whether a user was mentioned or not in a tweet.

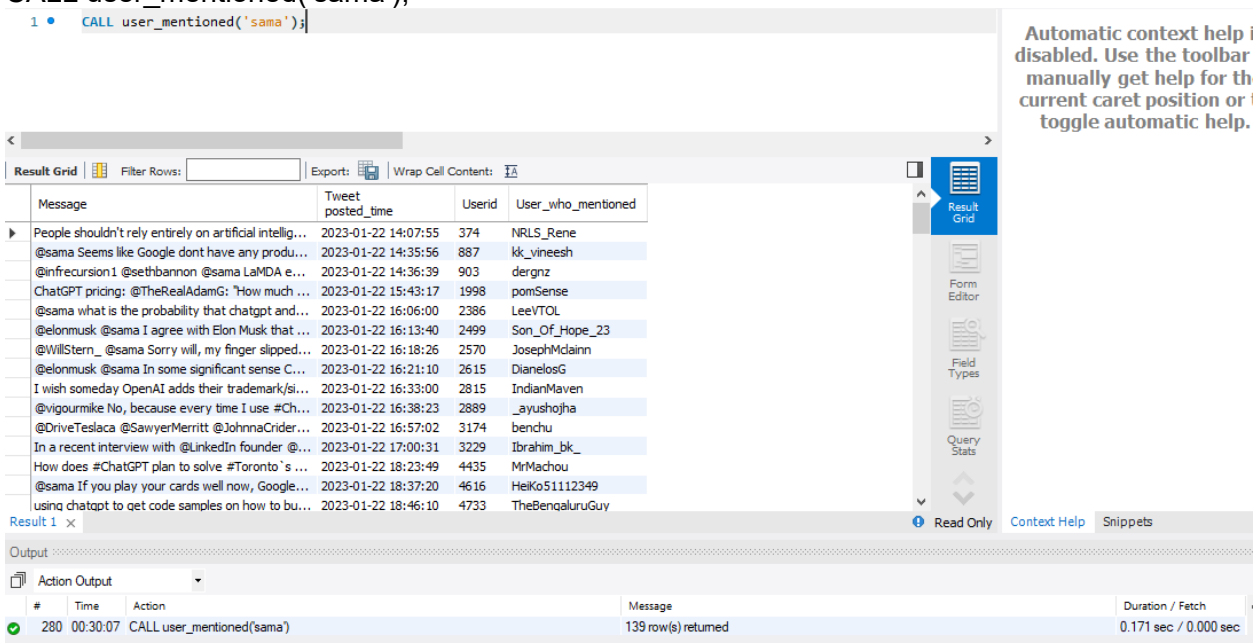
Query:

```
DROP PROCEDURE IF EXISTS user_mentioned;
DELIMITER //
CREATE PROCEDURE user_mentioned(IN p_username VARCHAR(20))
BEGIN
    SELECT tweet_text as 'Message',timestamp as 'Tweet posted_time',user_id as
'Userid',user_name as 'User_who_mentioned'
    FROM (
        SELECT media.tweet_text AS tweet_text,tweet.timestamp as timestamp,user.user_id as
user_id,user.username as user_name
        FROM media
        JOIN tweet ON tweet.tweet_id = media.tweet_id
        JOIN mentioned_users ON mentioned_users.tweet_id = tweet.tweet_id
        JOIN user ON user.user_id = tweet.user_id
        WHERE mentioned_users.mentioned_user = p_username
        UNION ALL
        SELECT CONCAT(p_username,' was not mentioned.') AS tweet_text,NULL as
timestamp,NULL as user_id ,NULL as user_name
        FROM dual
        WHERE NOT EXISTS(SELECT mentioned_user FROM mentioned_users WHERE
mentioned_user = p_username)
    ) t;
END //
DELIMITER ;
```

```
CALL user_mentioned(‘sama’);
CALL user_mentioned(‘gusthema’);
```

Snippet:

CALL user_mentioned('sama');



```
CALL user_mentioned('sama');
```

Message	Tweet posted_time	Userid	User_who_mentioned
People shouldn't rely entirely on artificial intellig...	2023-01-22 14:07:55	374	NRLS_Rene
@sama Seems like Google dont have any produ...	2023-01-22 14:35:56	887	kk_vineesh
@infrecursion1 @sethbannon @sama LaMDA e...	2023-01-22 14:36:39	903	dergnz
ChatGPT pricing: @TheRealAdamG: "How much ...	2023-01-22 15:43:17	1998	pomSense
@sama what is the probability that chatgpt and...	2023-01-22 16:06:00	2386	LeeVTOL
@elonmusk @sama I agree with Elon Musk that ...	2023-01-22 16:13:40	2499	Son_Of_Hope_23
@WillStern_ @sama Sorry will, my finger slipped...	2023-01-22 16:18:26	2570	JosephMclainn
@elonmusk @sama In some significant sense C...	2023-01-22 16:21:10	2615	DianelosG
I wish someday OpenAI adds their trademark/si...	2023-01-22 16:33:00	2815	IndianMaven
@vigourmike No, because every time I use #Ch...	2023-01-22 16:38:23	2889	_ayushojha
@DriveTeslaca @SawyerMerritt @JohnnaCrider...	2023-01-22 16:57:02	3174	benchu
In a recent interview with @LinkedIn founder @...	2023-01-22 17:00:31	3229	Ibrahim_bk_
How does #ChatGPT plan to solve #Toronto`s ...	2023-01-22 18:23:49	4435	MrMachou
@sama If you play your cards well now, Google...	2023-01-22 18:37:20	4616	HeiKo51112349
using chatgpt to get code samples on how to bu...	2023-01-22 18:46:10	4733	TheBengaluruGuy

Result 1 — Read Only

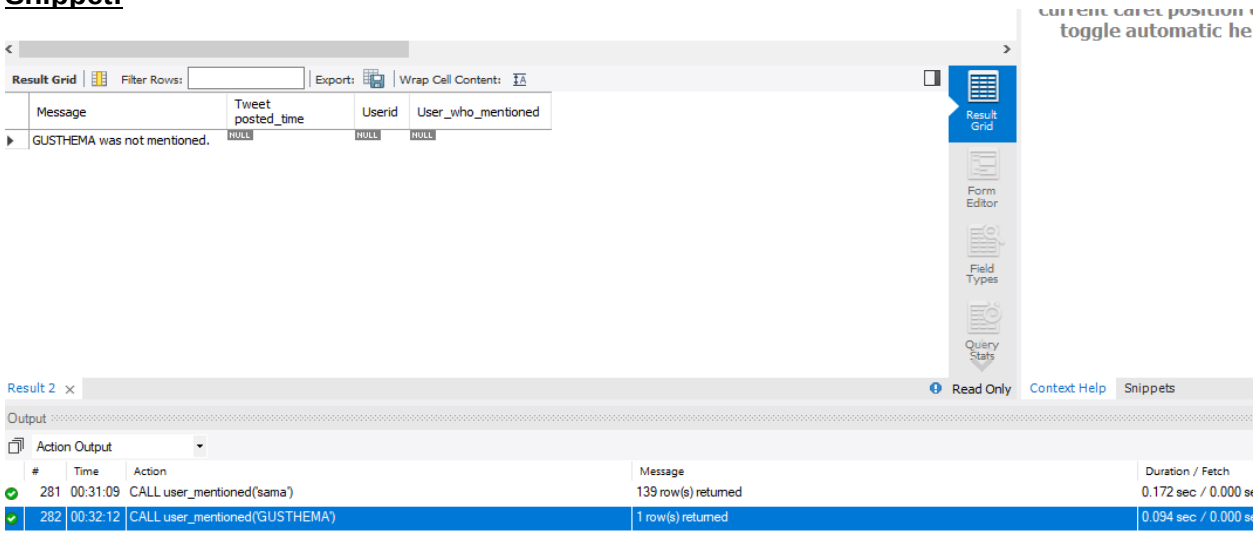
Output — Action Output

#	Time	Action	Message	Duration / Fetch
280	00:30:07	CALL user_mentioned('sama')	139 row(s) returned	0.171 sec / 0.000 sec

Observation: The above stored procedure shows the messages, timestamp of the tweet that is posted, the user_id and finally the user who mentioned the name 'sama'.

CALL user_mentioned('gusthema');

Snippet:



Message	Tweet posted_time	Userid	User_who_mentioned
GUSTHEMA was not mentioned.	NULL	NULL	NULL

Result 2 — Read Only

Output — Action Output

#	Time	Action	Message	Duration / Fetch
281	00:31:09	CALL user_mentioned('sama')	139 row(s) returned	0.172 sec / 0.000 se
282	00:32:12	CALL user_mentioned('GUSTHEMA')	1 row(s) returned	0.094 sec / 0.000 se

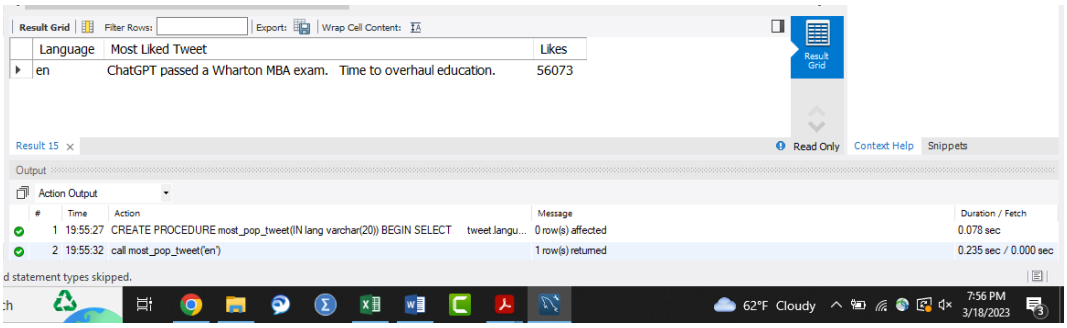
Observation: From the above call function, User 'gusthema' was not at all mentioned by any other users.

4. Stored Procedure to retrieve most popular/liked tweet for a given language.

Query:

```
DELIMITER //
CREATE PROCEDURE most_pop_tweet(IN lang varchar(20))
BEGIN
SELECT
   tweet.language as 'Language', media.tweet_text as 'Most Liked Tweet', count.like_count as
'Likes' FROM
   tweet
      JOIN
   media USING (tweet_id)
      JOIN
   count USING (tweet_id)
WHERE
   tweet.language = lang
ORDER BY like_count DESC
LIMIT 1;
END //

DELIMITER ;

call most_pop_tweet('en');
```

Snippet:



Observation: The expected result of the stored procedure 'most_pop_tweet' is to retrieve the most popular/liked tweet for a given language. When the stored procedure is called with the input parameter 'en' for the English language, it will return the language, the most liked tweet, and the number of likes for that tweet.

The observation from the result is that for the English language, the most liked tweet has been retrieved and displayed along with the number of likes it has received.

E. TRIGGERS:

DB Logging Using Triggers:

> Created triggers & displaying them for logging.

> Updating user and tweet tables and checking the record with log_details table.

Queries:

```
DELIMITER $$
CREATE TRIGGER update_like_count
AFTER UPDATE ON tweet
FOR EACH ROW
BEGIN
   IF NEW.like_count != OLD.like_count THEN
      UPDATE count SET like_count = NEW.like_count WHERE tweet_id = OLD.tweet_id;
   END IF;
END$$
DELIMITER ;
****************************************
DELIMITER $$
CREATE TRIGGER UpdateCountOnTweetUpdate
AFTER UPDATE ON tweet
FOR EACH ROW
BEGIN
   UPDATE count SET
      reply_count = NEW.reply_count,
      retweet_count = NEW.retweet_count,
      like_count = NEW.like_count,
      quote_count = NEW.quote_count
   WHERE tweet_id = NEW.tweet_id;
END $$
DELIMITER ;
****************************************
DELIMITER $$
CREATE TRIGGER Delete_Related_Rows
AFTER DELETE ON tweet
FOR EACH ROW
BEGIN
   DELETE FROM hashtag WHERE tweet_id = OLD.tweet_id;
   DELETE FROM media WHERE tweet_id = OLD.tweet_id;
   DELETE FROM mentioned_users WHERE tweet_id = OLD.tweet_id;
   DELETE FROM outlinks WHERE tweet_id = OLD.tweet_id;
   DELETE FROM count WHERE tweet_id = OLD.tweet_id;
   DELETE FROM user WHERE user_id = OLD.user_id
      AND NOT EXISTS (SELECT 1 FROM tweet WHERE user_id = OLD.user_id);
END $$
DELIMITER ;
****************************************
```

Creating table log_details to capture and have the track of changes made to user & Tweet Tables.

```
CREATE TABLE log_details (
        Log_id INT AUTO_INCREMENT PRIMARY KEY,
        table_name VARCHAR(255),
        column_name VARCHAR(255),
        old_value VARCHAR(255),
```

```
            new_value VARCHAR(255),
            changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            changed_by VARCHAR(255)
          );
```
****

Insert Trigger on USER table when we have a new row on the table.

```
DELIMITER $$
CREATE TRIGGER new_user_insert
AFTER INSERT ON user
FOR EACH ROW
BEGIN
  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('user', 'user_id', NULL, NEW.user_id, USER());

  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('user', 'username', NULL, NEW.username, USER());
END$$
DELIMITER ;
```
****

Update Trigger on USER table to update a new value replacing the old value.

```
DELIMITER $$
CREATE TRIGGER update_user AFTER UPDATE ON user
FOR EACH ROW
BEGIN
  IF OLD.user_id <> NEW.user_id THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('user', 'user_id', OLD.user_id, NEW.user_id, USER());
  END IF;

  IF OLD.username <> NEW.username THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('user', 'username', OLD.username, NEW.username, USER());
  END IF;
END$$
DELIMITER ;
```
****

Log the new value in the log_details table when we have a new value to get insert in the tweet table row.

```
DELIMITER $$
CREATE TRIGGER new_tweet_insert
AFTER INSERT ON tweet
FOR EACH ROW
BEGIN
  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('tweet', 'tweet_id', NULL, NEW.tweet_id, USER());

  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
```

```
  VALUES ('tweet', 'timestamp', NULL, NEW.timestamp, USER());

  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('tweet', 'conv_id', NULL, NEW.conv_id, USER());

  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('tweet', 'user_id', NULL, NEW.user_id, USER());

  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('tweet', 'language', NULL, NEW.language, USER());

  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('tweet', 'source', NULL, NEW.source, USER());

  INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
  VALUES ('tweet', 'permalink', NULL, NEW.permalink, USER());
END$$
DELIMITER ;
```

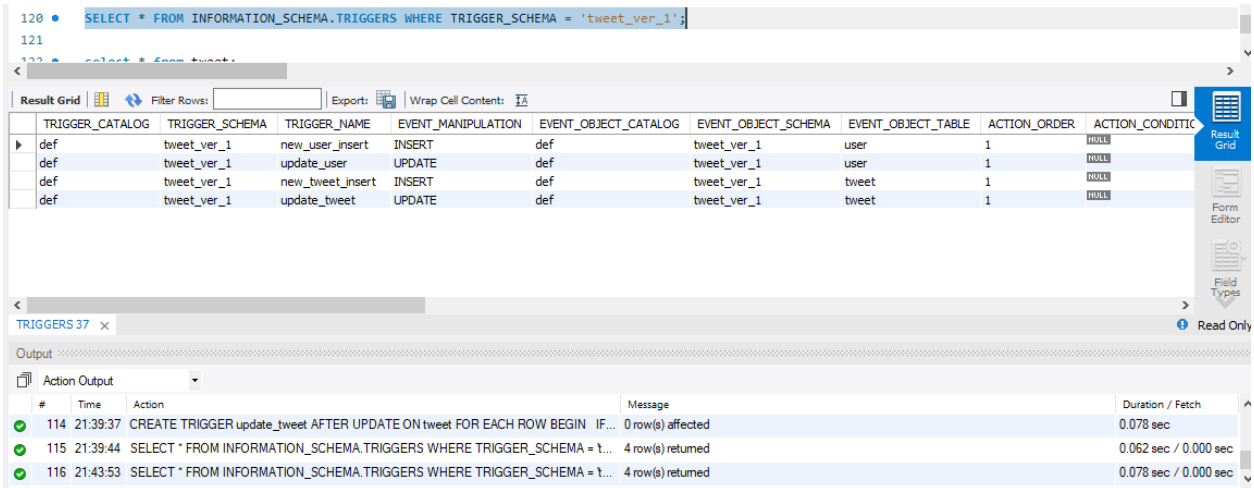
****

Update a old value with new value and having a log about it in the log_details table.

```
DELIMITER $$
CREATE TRIGGER update_tweet
AFTER UPDATE ON tweet
FOR EACH ROW
BEGIN
  IF OLD.tweet_id <> NEW.tweet_id THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('tweet', 'tweet_id',
          OLD.tweet_id, NEW.tweet_id, USER());
  END IF;

  IF OLD.timestamp <> NEW.timestamp THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('tweet', 'timestamp',
          OLD.timestamp, NEW.timestamp, USER());
  END IF;

  IF OLD.conv_id <> NEW.conv_id THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('tweet', 'conv_id',
          OLD.conv_id, NEW.conv_id, USER());
  END IF;

  IF OLD.user_id <> NEW.user_id THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('tweet', 'user_id',
          OLD.user_id, NEW.user_id, USER());
  END IF;

  IF OLD.language <> NEW.language THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
```

```
    VALUES ('tweet', 'language',
           OLD.language, NEW.language, USER());
  END IF;

  IF OLD.source <> NEW.source THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('tweet', 'source',
           OLD.source, NEW.source, USER());
  END IF;

  IF OLD.permalink <> NEW.permalink THEN
    INSERT INTO log_details (table_name, column_name, old_value, new_value, changed_by)
    VALUES ('tweet', 'permalink',
           OLD.permalink, NEW.permalink, USER());
  END IF;
END$$
DELIMITER ;
```

Snippet:



```
SELECT * FROM INFORMATION_SCHEMA.TRIGGERS WHERE TRIGGER_SCHEMA = 'tweet_ver_1';
```

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

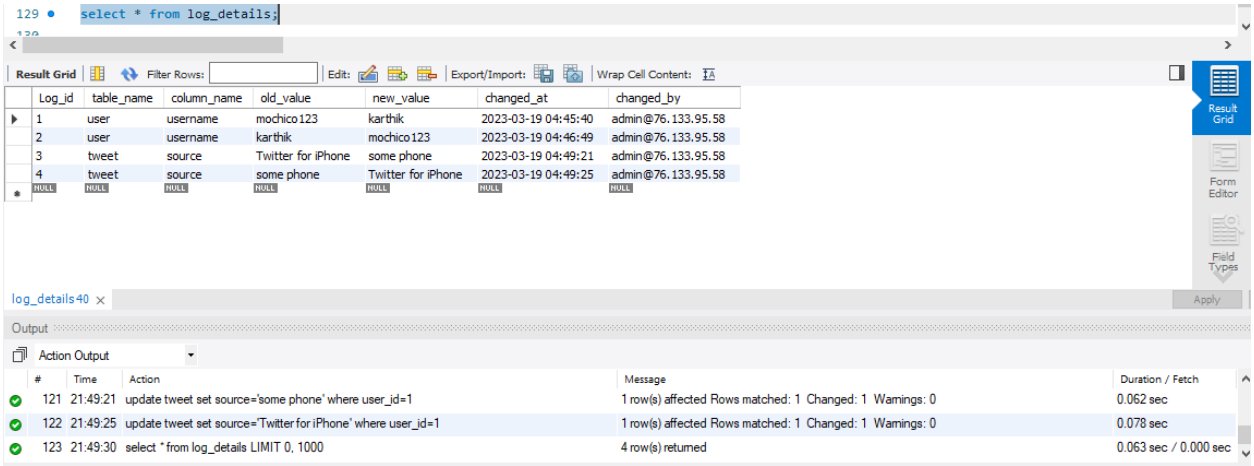
TRIGGER_CATALOG	TRIGGER_SCHEMA	TRIGGER_NAME	EVENT_MANIPULATION	EVENT_OBJECT_CATALOG	EVENT_OBJECT_SCHEMA	EVENT_OBJECT_TABLE	ACTION_ORDER	ACTION_CONDITIO
def	tweet_ver_1	new_user_insert	INSERT	def	tweet_ver_1	user	1	NULL
def	tweet_ver_1	update_user	UPDATE	def	tweet_ver_1	user	1	NULL
def	tweet_ver_1	new_tweet_insert	INSERT	def	tweet_ver_1	tweet	1	NULL
def	tweet_ver_1	update_tweet	UPDATE	def	tweet_ver_1	tweet	1	NULL

TRIGGERS 37 | Read Only

Output | Action Output

#	Time	Action	Message	Duration / Fetch
114	21:39:37	CREATE TRIGGER update_tweet AFTER UPDATE ON tweet FOR EACH ROW BEGIN IF...	0 row(s) affected	0.078 sec
115	21:39:44	SELECT * FROM INFORMATION_SCHEMA.TRIGGERS WHERE TRIGGER_SCHEMA = 't...	4 row(s) returned	0.062 sec / 0.000 sec
116	21:43:53	SELECT * FROM INFORMATION_SCHEMA.TRIGGERS WHERE TRIGGER_SCHEMA = 't...	4 row(s) returned	0.078 sec / 0.000 sec

Snippet:



```
129 •  select * from log_details;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Log_id	table_name	column_name	old_value	new_value	changed_at	changed_by
1	user	username	mochico123	karthik	2023-03-19 04:45:40	admin@76.133.95.58
2	user	username	karthik	mochico123	2023-03-19 04:46:49	admin@76.133.95.58
3	tweet	source	Twitter for iPhone	some phone	2023-03-19 04:49:21	admin@76.133.95.58
4	tweet	source	some phone	Twitter for iPhone	2023-03-19 04:49:25	admin@76.133.95.58
NULL	NULL	NULL	NULL	NULL	NULL	NULL

log_details 40

Output

Action Output

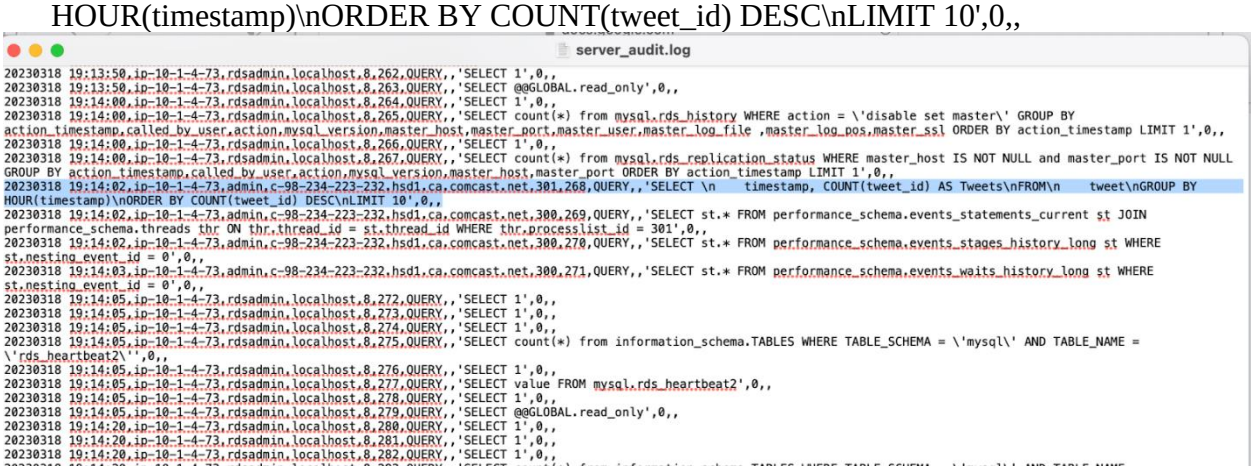
#	Time	Action	Message	Duration / Fetch
121	21:49:21	update tweet set source='some phone' where user_id=1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.062 sec
122	21:49:25	update tweet set source='Twitter for iPhone' where user_id=1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.078 sec
123	21:49:30	select * from log_details LIMIT 0, 1000	4 row(s) returned	0.063 sec / 0.000 sec

Observations: This code creates a table named "log_details" to track the changes made to the "user" and "tweet" tables. It also creates triggers to insert and update log records in the "log_details" table when the corresponding tables are updated. The update queries on the "user" and "tweet" tables demonstrate how the log details table tracks the changes made to the columns of the respective tables. The log_details table captures the details of the changes made to the user and tweet tables, including the name of the table, the name of the column, the old value, the new value, the time when the change was made, and who made the change. Overall, this code provides a useful mechanism for auditing changes made to the data in the "user" and "tweet" tables.

F. DB Logging

Example log of executing a query:

20230318 19:09:47,ip-10-1-4-73,admin,c-98-234-223-232.hsd1.ca.comcast.net,301,40,QUERY,,'SELECT \n timestamp, COUNT(tweet_id) AS Tweets\nFROM\n tweet\nGROUP BY HOUR(timestamp)\nORDER BY COUNT(tweet_id) DESC\nLIMIT 10',0,,



server_audit.log

```
20230318 19:13:50,ip-10-1-4-73,rdsadmin,localhost,8,262,QUERY,,'SELECT 1',0,,
20230318 19:13:50,ip-10-1-4-73,rdsadmin,localhost,8,263,QUERY,,'SELECT @@GLOBAL.read_only',0,,
20230318 19:14:00,ip-10-1-4-73,rdsadmin,localhost,8,264,QUERY,,'SELECT 1',0,,
20230318 19:14:00,ip-10-1-4-73,rdsadmin,localhost,8,265,QUERY,,'SELECT count(*) from mysql.rds_history WHERE action = \'disable set master\' GROUP BY
action_timestamp,called_by_user,action,mysql_version,master_host,master_port,master_user,master_log_file ,master_log_pos,master_ssl ORDER BY action_timestamp LIMIT 1',0,,
20230318 19:14:00,ip-10-1-4-73,rdsadmin,localhost,8,266,QUERY,,'SELECT 1',0,,
20230318 19:14:00,ip-10-1-4-73,rdsadmin,localhost,8,267,QUERY,,'SELECT count(*) from mysql.rds_replication_status WHERE master_host IS NOT NULL and master_port IS NOT NULL
GROUP BY action_timestamp,called_by_user,action,mysql_version,master_host,master_port ORDER BY action_timestamp LIMIT 1',0,,
20230318 19:14:02,ip-10-1-4-73,admin,c-98-234-223-232.hsd1.ca.comcast.net,301,268,QUERY,,'SELECT \n    timestamp, COUNT(tweet_id) AS Tweets\nFROM\n    tweet\nGROUP BY
HOUR(timestamp)\nORDER BY COUNT(tweet_id) DESC\nLIMIT 10',0,,
20230318 19:14:02,ip-10-1-4-73,admin,c-98-234-223-232.hsd1.ca.comcast.net,300,269,QUERY,,'SELECT st.* FROM performance_schema.events_statements_current st JOIN
performance_schema.threads thr ON thr.thread_id = st.thread_id WHERE thr.processlist_id = 301',0,,
20230318 19:14:02,ip-10-1-4-73,admin,c-98-234-223-232.hsd1.ca.comcast.net,300,270,QUERY,,'SELECT st.* FROM performance_schema.events_stages_history_long st WHERE
st.nesting_event_id = 0',0,,
20230318 19:14:03,ip-10-1-4-73,admin,c-98-234-223-232.hsd1.ca.comcast.net,300,271,QUERY,,'SELECT st.* FROM performance_schema.events_waits_history_long st WHERE
st.nesting_event_id = 0',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,272,QUERY,,'SELECT 1',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,273,QUERY,,'SELECT 1',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,274,QUERY,,'SELECT 1',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,275,QUERY,,'SELECT count(*) from information_schema.TABLES WHERE TABLE_SCHEMA = \'mysql\' AND TABLE_NAME =
\'rds_heartbeat2\'',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,276,QUERY,,'SELECT 1',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,277,QUERY,,'SELECT value FROM mysql.rds_heartbeat2',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,278,QUERY,,'SELECT 1',0,,
20230318 19:14:05,ip-10-1-4-73,rdsadmin,localhost,8,279,QUERY,,'SELECT @@GLOBAL.read_only',0,,
20230318 19:14:20,ip-10-1-4-73,rdsadmin,localhost,8,280,QUERY,,'SELECT 1',0,,
20230318 19:14:20,ip-10-1-4-73,rdsadmin,localhost,8,281,QUERY,,'SELECT 1',0,,
20230318 19:14:20,ip-10-1-4-73,rdsadmin,localhost,8,282,QUERY,,'SELECT 1',0,,
```

Reference - https://aws.amazon.com/blogs/database/configuring-an-audit-log-to-capture-database-activities-for-amazon-rds-for-mysql-and-amazon-aurora-with-mysql-compatibility/

Observation:
This is a log entry of a database query executed at a specific time (March 18th, 2023 at 7:09:47 PM) by an admin user from an IP address (10.1.4.73) associated with a host name (c-98-234-223-232.hsd1.ca.comcast.net). The query itself is selecting the timestamp and count of tweets from a table named "tweet", grouping the results by the hour of the timestamp, ordering the results by the count of tweet IDs in descending order, and limiting the results to the top 10.

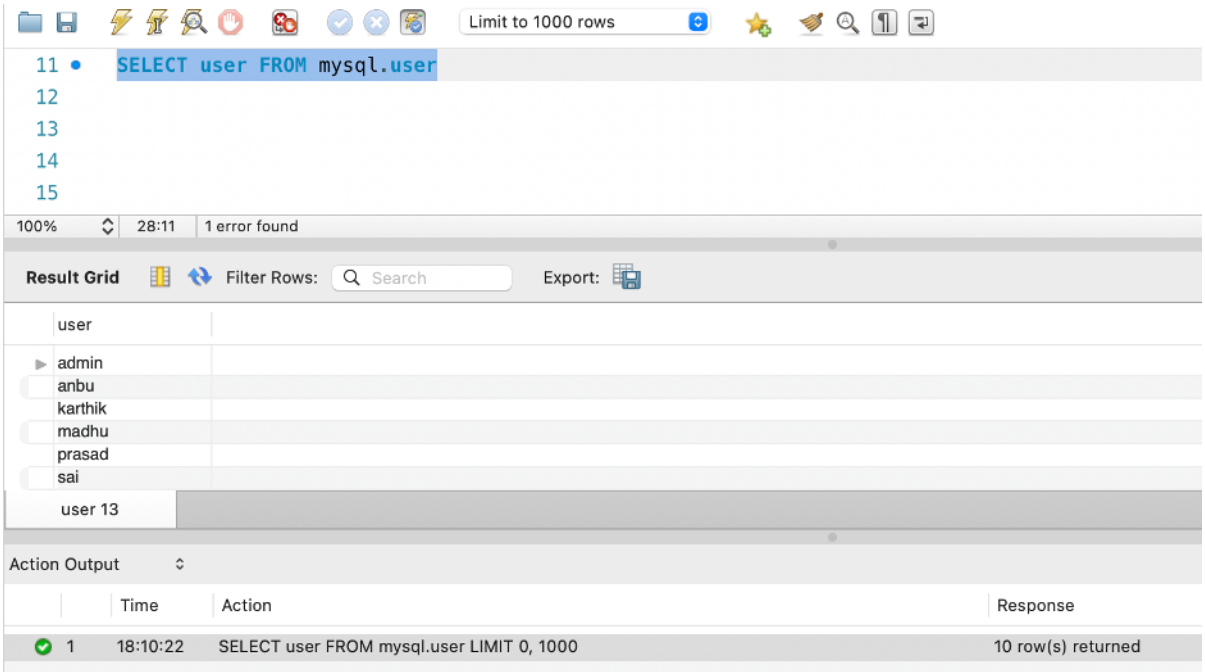
G. Access privileges:

MySQL provides Data Control Language (DCL) through GRANT and REVOKE statements for controlling access to specific roles or users. In our application, all group members have access to all Data Manipulation Language (DML) statements, while a restricted user is used to demonstrate access privileges.

1. We can create users

CREATE USER user@%
IDENTIFIED BY 'password';

Snippet:



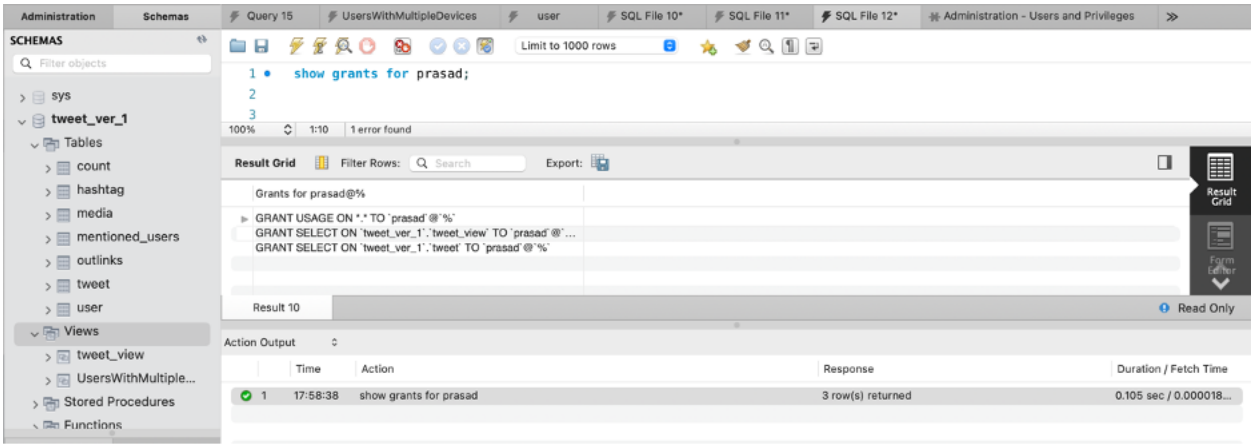
Observation: To create users, we can use the CREATE USER statement, and to list all users, we can use a SELECT statement on the mysql. user table. We can use GRANT

statements to provide restricted access to a specific table or view for a particular user, limiting their access to only the granted privileges.

2. **We grant restricted access to select particular table and view for specific user 'Prasad'**

GRANT SELECT ON tweet_ver_1.tweet_view TO 'prasad'@'%';
GRANT SELECT ON tweet_ver_1.tweet TO 'prasad'@'%';

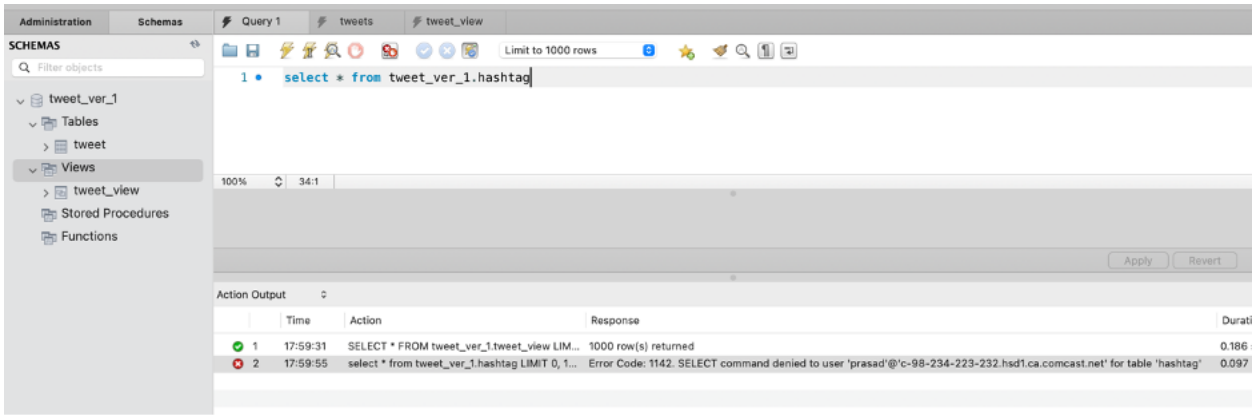
Snippet:



Observation: In these two statements and the result snippet, we are granting restricted access to a specific user named 'Prasad' to select from two tables: 'tweet_ver_1.tweet_view' and 'tweet_ver_1.tweet'. This means that the user 'Prasad' will only be able to perform SELECT queries on these two tables and won't be able to perform any other operations on them. The '@' symbol with '%' indicates that this user can access these tables from any host.

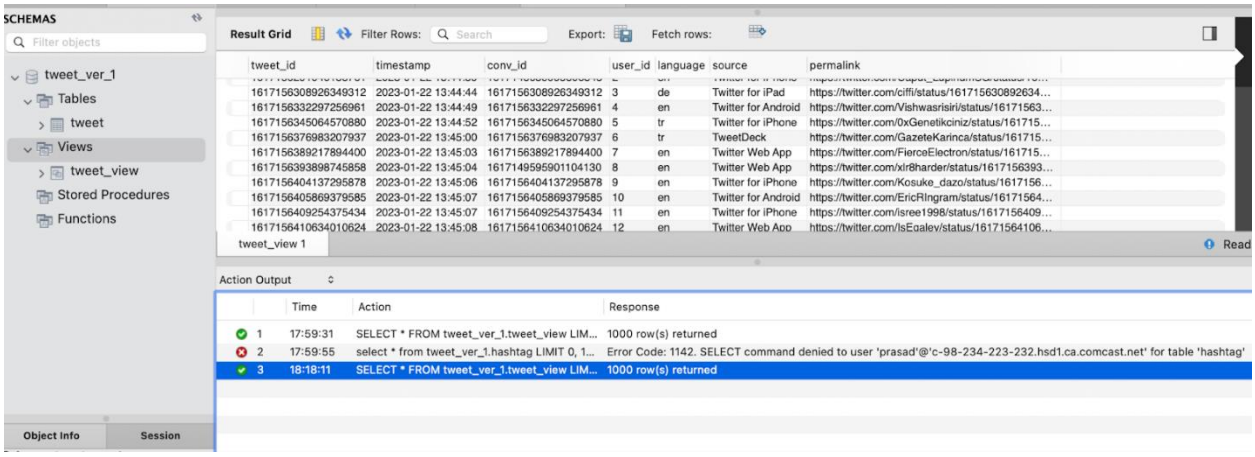
3. **The access privilege restricts the user to access anything other than the granted.**

Snippet:



Observation: This is an access privilege that restricted access to the user prasad as the command has been denied to the user.

Snippet:



Observation: This is an access privilege that has revoked the restriction and granted the permission to the user to have an access to the tables and the database.