

# DATA 230: Lab 1 ChatGPT Analyzer

SJSU ChatGPT Tweet Analysis

Anusmriti Sikdar  
Data Analytics  
SJSU  
San Jose, USA

anusmriti.sikdar@sjsu.edu

Priyanka Bhyregowda  
Data Analytics  
SJSU

San Jose, USA  
priyanka.bhyregowda@sjsu.edu

Akanksha Tyagi  
Data Analytics  
SJSU  
San Jose, USA  
akanksha.tyagi@sjsu.edu

Sahana Thoravalli Prabhuswamy  
Data Analytics  
SJSU  
San Jose, USA  
sahana.thoravalliprabhuswamy@sjsu.edu

Anjali Mishra  
Data Analytics  
SJSU  
San Jose, USA  
anjalihimanshu.ojha@sjsu.edu

**Abstract**—Recently, ChatGPT has been one of the most discussed topics. Twitter is a popular social media platform for people to connect and share their thoughts. Analyzing Twitter data provides deep insights into the way people think. By analyzing Twitter data on ChatGPT we can obtain valuable information. Hence, there is a need for a ChatGPT tweet analyzer application. A relational database provides many advantages like access, security, and integrity of the data compared to other earlier storage systems like files. Many users can concurrently access the database, the data is secured as authorization is required for data access. Any RDBMS promises data integrity with its ACID properties. MySQL is an open-source RDBMS that has all the required features for storing application-related data.

**Keywords**—ChatGPT, AWS RDS, RDBMS

## I. INTRODUCTION

Analysis of tweet data has been a major research trend. ChatGPT which is an AI-based chatbot is found to bring revolutionary changes to the field of Artificial Intelligence and Machine Learning technology. Developing a cloud-native application is the modern approach for every software solution. AWS RDS provides a cloud-based RDBMS environment for any data storage needs. Many useful metrics can be obtained by analyzing tweet data related to ChatGPT. A client-server model application for the extraction and visualization of these results is one of the solutions. The ChatGPT analyzer app can be run on any machine with a command line interface, python, and access to the internet, which will be available in most of the systems.

**A. Functional Requirements of our application from the given dataset:**

- 1) Most active users from the dataset
- 2) Most Discussed tweet with respect to replies based on conversation ID in the given data
- 3) Most used language to tweet on chatGPT
- 4) Most Liked Tweet about ChatGPT
- 5) Most Used Media Type
- 6) Users with more number of retweets
- 7) Number of Tweets on 'ChatGPT'
- 8) Most retweeted tweet with the comments
- 9) Conversations with the most number of media shares.
- 10) Most retweeted tweet in each language

- 11) Top 10 Users who have more outsource for reference on the tweets
- 12) Number of tweets each day on chatGPT
- 13) Most viral tweets
- 14) Most popular hashtags used
- 15) Most used sources to tweet about ChatGPT
- 16) Most mentioned users in the tweets about ChatGPT
- 17) Handling the deletion of records in the tweets table gracefully
- 18) Restricting tweet length to 280
- 19) Auditing the deleted tweets

**B. Limitations of our application from the given dataset:**

1. Provides a fixed number of metrics
2. The metrics are based on a small set of data
3. Static queries

**C. Deployment and Usage:**

The database of the application is deployed in the cloud and the users need to have the required files in their system to run the application. The user can run the application in the command line and obtain the required metrics. The user can load his tweet dataset into the AWS RDS DB instance using the scripts. Only a DB Admin connects to the AWS RDS DB instance and makes changes.

## II. FUNCTIONAL COMPONENT ANALYSIS

Functional Components of SJSU ChatGPT Analyzer

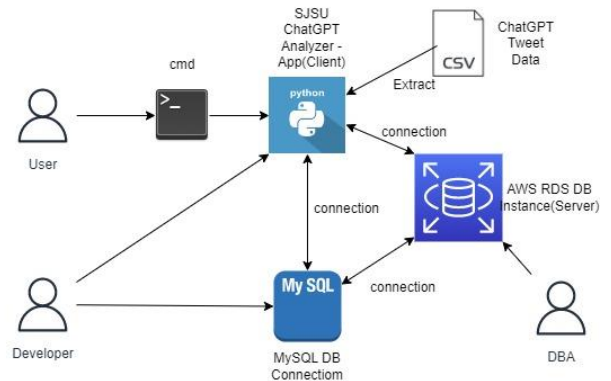
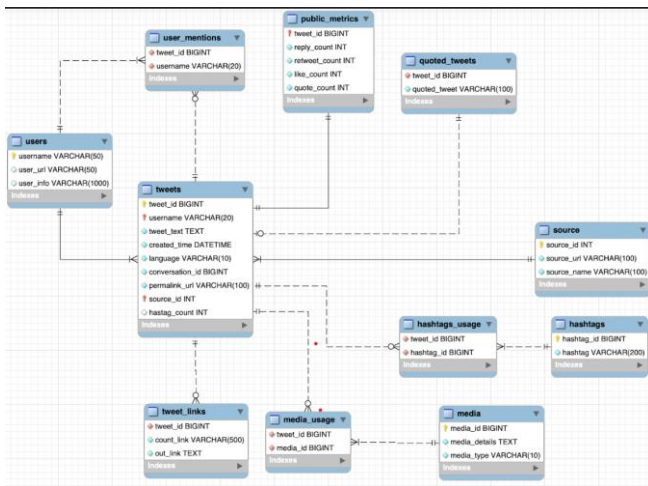


Fig. 1. Functional component analysis

SJSU ChatGPT Analyzer application has a client-server architecture. The Analyzer application acts as a client to the AWS RDS MySQL Instance which acts as a server. AWS RDS provides a cloud-based database environment that caters to the data need of our application. The raw data which is in the form of a .csv file is extracted using a python program and then cleaned and transformed into the required format and then loaded into the 'twitter' database in AWS RDS MySQL Instance. The application runs on a command line interface and the user of our application can select options to view a few of the interesting metrics that can be obtained from the application. The developer can connect to AWS RDS MySQL Instance endpoints either through a MySQL client or through the python application. Only a DB Admin can connect to the AWS RDS MySQL Instance and can do performance tuning and handles access and privileges of the DB.

### III. ER DIAGRAM



For the given data set we identified the entities like "User", "Tweet", "Media", "Hashtag", "Out\_Links" and "Source" and their corresponding relations. The chatgpt data has the above mentioned components. There are following relationship we captured between entities as "User\_Mentions", "Media\_Usage" and "Out\_Links". The main relation is "Tweets" relation which kinds of ties everything together. For the sources we see very high redundancy so keep it out in a different table which have only 843 different entries so keeping it out in a different table minimize the table size also.

While designing the database we keep the read and write operations in mind. Like "Public\_Metrics" which will be keep updated after every impression so keeping those things out from "tweets" table will minimize the unnecessary updates to the "tweet" table. Once a tweet is saved there are no updates regarding that tweet.

To capture all the "hashtags" in the system, we keep them in a separate table and there is relation "hashtag\_usage" will keep record which tweet uses which hashtags. If someone delete their tweet, it will not delete the hashtag, because they may be used in other tweets. We used

the similar design for the media also which is again reused many time.

"User\_mentions" table will keep which user mentioned in which tweet, since it captures the only users who are already in the system, so there is relationship between "users" table and "user\_mentions" table.

For the conversations, its associated with the parent tweet so we keep the conversationId within the "tweets" table and there are no other attributes associated with the conversation within the dataset which require it to be a separate entity.

Our database is 3NF normalized and we did not keep any partial and transitive dependencies. Our design keep in mind the kind of use case we are solving and make those queries as fast as possible with very minimal joins.

### IV. METHODS

Methods implemented in this Lab is summarized in this section.

#### A. Stored Procedures

- Stored Procedure 1: get\_most\_viral\_tweet()

The stored procedure get\_most\_viral\_tweet will return the 15 most viral tweets from the tweets table.

```

SQL Code:
DROP PROCEDURE IF EXISTS get_most_viral_tweet;
DELIMITER //
CREATE PROCEDURE get_most_viral_tweet()
BEGIN
    SELECT
        t.tweetId, t.userName, t.tweetText, t.createdAt, t.lang, t.conversationId, t.permalinkUrl,
        t.sourceId,
        COALESCE(pm.retweetCount, 0) + COALESCE(pm.quoteCount, 0) +
        COALESCE(pm.likeCount, 0) AS viralCount
    FROM
        tweets t LEFT JOIN public_metrics pm ON t.tweetId = pm.tweetId
    ORDER BY viralCount DESC
    LIMIT 15;
CALL get_most_viral_tweet();
  
```

Fig. 2. get\_most\_viral\_tweet()

Explanation- This stored procedure first joins the tweets table with the public\_metrics table using a left join, so that all tweets will be returned even if they have no associated public metrics yet. The COALESCE function is used to handle cases where the public metrics are NULL (i.e. if a tweet has no retweets, likes, or quotes). We then calculate the viral count by adding up the retweet, quote, and like counts. The results are sorted in descending order by viral count. LIMIT 15 will return 15 rows, which is the most viral tweets based on the number of retweets, quotes, and likes.

We can call the procedure with call statement and execute it like: CALL get\_most\_viral\_tweet();

- Stored Procedure 2: get\_hashtag\_count()

This procedure named GET\_HASHTAG\_COUNT processes data from hash\_tags table to count and return.

```

SQL Code:
DROP PROCEDURE IF EXISTS GET_HASHTAG_COUNT;
DELIMITER //
CREATE PROCEDURE GET_HASHTAG_COUNT()
BEGIN
    SELECT
        h.hashtag, count(tweetId) as
total_count
    FROM
        hashtags h join hashtags_usage hu
    using (hashtagId)
    group by
        h.hashtagId
    order by
        total_count desc;

```

Fig. 2. get\_hashtag\_count()

The stored procedure named GET\_HASHTAG\_COUNT processes data from hash\_tags table to count and return the most frequently used hashtags. It joins the hashtags table with the hashtag\_usage table using hashtag ID and retrieve the count of the Hashtags which is returned in DESCENDING order.

We can call the procedure with call statement and execute it like: CALL GET\_HASHTAG\_COUNT ();

- Stored Procedure 3: most\_used\_source()

This stored procedure most\_used\_source will return the top 10 most used sources for tweet.

```

SQL Code:

drop procedure if exists most_used_source;
DELIMITER //
CREATE PROCEDURE most_used_source()
BEGIN
    SELECT
        s.SourceName as Source_Name, count(t(tweetId) as Total_Count
    FROM
        tweets t JOIN `source` s using (sourceId)
    GROUP BY s.SourceName
    ORDER BY Total_Count desc
    LIMIT 10;

```

The stored procedure most\_used\_source selects and returns the top 10 most used sources for tweets and return the most used sources. The procedure starts by selecting the SourceName column from the source table and counting the number of tweets associated with each source using the COUNT() function. The tweets table is joined with the source table using the sourceId foreign key.

We can call the procedure with call statement and execute it like : CALL most\_used\_source ();

- Stored Procedure 4: most\_mentioned\_user()

The stored procedure most\_mentioned\_user aims to identify and return the top 10 most frequently mentioned Twitter users in the tweets table.

```

SQL Code:

drop procedure if exists most_mentioned_user ;
DELIMITER //
CREATE PROCEDURE most_mentioned_user()
BEGIN
    select
        username, count(tweetId) as Total_Mentions
    from
        user_mentions
    group by username
    order by Total_Mentions desc
    limit 10;

```

Fig. 4. most\_mentioned\_user()

The procedure selects the username column from the user\_mentions table and counting the number of tweets in which each user is mentioned using the COUNT() function. The results are grouped by the username using the GROUP BY clause and sorted in descending order by the total count of mentions for each user using the ORDER BY clause. Finally, the top 10 results are returned using the LIMIT clause. We can call the procedure with call statement and execute it like: most\_mentioned\_user ();

## B. Triggers

- Trigger 1: prepare\_before\_deleting\_tweet

The trigger prepare\_before\_deleting\_tweet is executed before a delete operation is performed on the tweets table for each row being deleted.

```

SQL Code:
DROP TRIGGER IF EXISTS prepare_before_deleting_tweet;
DELIMITER //
CREATE TRIGGER prepare_before_deleting_tweet
Before delete ON tweets
FOR EACH ROW
BEGIN
    -- Delete from user mentions table
    DELETE from user_mentions where tweetId = old(tweetId;
    -- Delte from hashtag usage table
    DELETE from hashtags_usage where tweetId = old(tweetId;
    -- Delete from media usage table
    DELETE from media_usage where tweetId = old(tweetId;
    -- Delete from media usage table
    DELETE from public_metrics where tweetId = old(tweetId;
    -- Delete from media usage table
    DELETE from tweet_links where tweetId = old(tweetId;
END;
DELIMITER ;;

delete from tweets where tweetId = '1617156404137295878';
select * from tweets where tweetId = 1617156404137295878;

```

This trigger will delete any rows from the user\_mentions table that reference the tweet being deleted using the DELETE statement with a WHERE clause that matches the tweetId column to the old(tweetId value. Then, it deletes any rows from the hashtags\_usage table, media\_usage table, public\_metrics table, and tweet\_links table that reference the tweet being deleted in the same way. This trigger ensures that all associated data is deleted before the actual tweet is deleted, so that there are no foreign key constraints that prevent the deletion.

- Trigger 2: trigger\_tweet\_length

The trigger trigger\_tweet\_length is created and is set to execute BEFORE INSERT (for each row) on the

[illegible]

- Trigger 3: log\_before\_delete

```
SQL Code:

CREATE TABLE log (
    delete_timestamp datetime primary key,
    db_user varchar(50),
    tweet_id bigint not null,
    username varchar(50) not null
);
DELIMITER //
CREATE TRIGGER log_before_delete
Before delete ON tweets
FOR EACH ROW
BEGIN
    -- Delete from user mentions table
    DELETE from user_mentions where tweetId = old.tweetId;
    -- Delete from hashtag usage table
    DELETE from hashtag_usage where tweetId = old.tweetId;
    -- Delete from media usage table
    DELETE from media_usage where tweetId = old.tweetId;
    INSERT INTO log values (now(), user(), old.tweetid,old.username);
END;;
DELIMITER //

delete from tweets where username='group7';
```

- Query 1 : Active Users tweeted in a given time in descending order

- Query 2: Most Discussed tweet with respect to replies based on conversation id in the given data

- Query 3: Tweets on ChatGPT based on Used Language:

- Query 4: Most Liked Tweet about ChatGP:

- Table: users



```

-----
-- Table `twitter`.`users`
-----

DROP TABLE IF EXISTS `twitter`.`users` ;

CREATE TABLE IF NOT EXISTS `twitter`.`users` (
  `username` VARCHAR(50) NOT NULL,
  `User_url` VARCHAR(50) NULL DEFAULT NULL,
  `userInfo` VARCHAR(1000) NULL DEFAULT NULL,
  PRIMARY KEY (`username`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE UNIQUE INDEX `username` ON `twitter`.`users` (`username` ASC) VISIBLE;

```

- Table: source

```

-----
-- Table `twitter`.`source`
-----

DROP TABLE IF EXISTS `twitter`.`source` ;

CREATE TABLE IF NOT EXISTS `twitter`.`source` (
  `sourceId` BIGINT NOT NULL,
  `Source_url` VARCHAR(4000) NOT NULL,
  `sourceName` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`sourceId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

- Table : tweets

```

DROP TABLE IF EXISTS `twitter`.`tweets` ;
CREATE TABLE IF NOT EXISTS `twitter`.`tweets` (
  `tweetId` BIGINT NOT NULL,
  `username` VARCHAR(20) NOT NULL,
  `tweetText` TEXT NOT NULL,
  `createdTime` DATETIME NOT NULL,
  `lang` VARCHAR(10) NOT NULL,
  `conversationId` BIGINT NOT NULL,
  `permalinkUrl` VARCHAR(100) NOT NULL,
  `sourceId` BIGINT NOT NULL,
  `hashtagCounts` INT NULL DEFAULT '0',
  PRIMARY KEY (`tweetId`, `username`, `sourceId`),
  CONSTRAINT `tweets_ibfk_1`
    FOREIGN KEY (`username`)
      REFERENCES `twitter`.`users` (`username`),
  CONSTRAINT `tweets_ibfk_2`
    FOREIGN KEY (`sourceId`)
      REFERENCES `twitter`.`source` (`sourceId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
CREATE INDEX `username` ON `twitter`.`tweets` (`username` ASC) VISIBLE;
CREATE INDEX `sourceId` ON `twitter`.`tweets` (`sourceId` ASC) VISIBLE;

```

- Table: hashtag\_usage

```

DROP TABLE IF EXISTS `twitter`.`hashtags_usage` ;

CREATE TABLE IF NOT EXISTS `twitter`.`hashtags_usage` (
  `tweetId` BIGINT NOT NULL,
  `hashtagId` BIGINT NOT NULL,
  CONSTRAINT `hashtags_usage_ibfk_1`
    FOREIGN KEY (`tweetId`)
      REFERENCES `twitter`.`tweets` (`tweetId`),
  CONSTRAINT `hashtags_usage_ibfk_2`
    FOREIGN KEY (`hashtagId`)
      REFERENCES `twitter`.`hashtags` (`hashtagId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE INDEX `tweetId` ON `twitter`.`hashtags_usage` (`tweetId` ASC) VISIBLE;

```

- Table: media

```

DROP TABLE IF EXISTS `twitter`.`media` ;

```

```

) CREATE TABLE IF NOT EXISTS `twitter`.`media` (
  `mediaId` BIGINT NOT NULL,
  `mediaDetails` TEXT NOT NULL,
  `mediaType` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`mediaId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

- Table: media\_usage

```

DROP TABLE IF EXISTS `twitter`.`media_usage` ;

CREATE TABLE IF NOT EXISTS `twitter`.`media_usage` (
  `tweetId` BIGINT NOT NULL,
  `mediaId` BIGINT NOT NULL,
  CONSTRAINT `media_usage_ibfk_1`
    FOREIGN KEY (`tweetId`)
      REFERENCES `twitter`.`tweets` (`tweetId`),
  CONSTRAINT `media_usage_ibfk_2`
    FOREIGN KEY (`mediaId`)
      REFERENCES `twitter`.`media` (`mediaId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE INDEX `tweetId` ON `twitter`.`media_usage` (`tweetId` ASC) VISIBLE;

CREATE INDEX `mediaId` ON `twitter`.`media_usage` (`mediaId` ASC) VISIBLE;

```

- Table: public\_metrics

```

DROP TABLE IF EXISTS `twitter`.`public_metrics` ;

```

```

) CREATE TABLE IF NOT EXISTS `twitter`.`public_metrics` (
  `tweetId` BIGINT NOT NULL,
  `replyCount` INT NOT NULL DEFAULT '0',
  `retweetCount` INT NOT NULL DEFAULT '0',
  `likeCount` INT NOT NULL DEFAULT '0',
  `quoteCount` INT NOT NULL DEFAULT '0',
  PRIMARY KEY (`tweetId`),
  CONSTRAINT `public_metrics_ibfk_1`
    FOREIGN KEY (`tweetId`)
      REFERENCES `twitter`.`tweets` (`tweetId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

- Table : quoted\_tweets

```

DROP TABLE IF EXISTS `twitter`.`quoted_tweets` ;

) CREATE TABLE IF NOT EXISTS `twitter`.`quoted_tweets` (
  `tweetId` BIGINT NOT NULL,
  `quotedTweet` VARCHAR(100) NOT NULL,
  CONSTRAINT `quotedTweets_ibfk_1`
    FOREIGN KEY (`tweetId`)
      REFERENCES `twitter`.`tweets` (`tweetId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE INDEX `tweetId` ON `twitter`.`quoted_tweets` (`tweetId` ASC) VISIBLE;

```

- Table : tweet\_link

```

DROP TABLE IF EXISTS `twitter`.`tweet_links` ;

CREATE TABLE IF NOT EXISTS `twitter`.`tweet_links` (
  `tweetId` BIGINT NOT NULL,
  `countLink` VARCHAR(500) NOT NULL,
  `outLink` TEXT NOT NULL,
  CONSTRAINT `tweet_links_ibfk_1`
    FOREIGN KEY (`tweetId`)
      REFERENCES `twitter`.`tweets` (`tweetId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE INDEX `tweetId` ON `twitter`.`tweet_links` (`tweetId` ASC) VISIBLE;

```

## E. Access Privilege

RDBMS systems provide concurrent access to data and data security. A DBA or database administrator can create users and grant access to various levels of access to the users. In the application, there are three levels of access privileges. We have totally three users for our DB:

- 1) Admin - Has all access to all DBs
- 2) Developer - Has all access to twitter db
- 3) Analyst1 - Has SELECT access on twitter db

```

sql = '''SELECT user FROM mysql. user'''
cursor.execute(sql)
cursor.fetchall()

[['Analyst1'],
 ['Developer'],
 ['admin'],
 ['mysql.infoschema'],
 ['mysql.session'],
 ['mysql.sys'],
 ['rdsadmin'],]]

15] sql = '''SHOW GRANTS FOR Developer;'''
cursor.execute(sql)
cursor.fetchall()

[['GRANT USAGE ON *.* TO `Developer`@`%`'],
 ['GRANT ALL PRIVILEGES ON `twitter`.`*` TO `Developer`@`%`'],]]

sql = '''SHOW GRANTS FOR Analyst1;'''
cursor.execute(sql)
cursor.fetchall()

[['GRANT USAGE ON *.* TO `Analyst1`@`%`'],
 ['GRANT SELECT ON `twitter`.`*` TO `Analyst1`@`%`'],]]

```

## F. Logging

Log files are stored in the form of tables, error logs are enabled by default and slow query logs and general logs are enabled for the general maintenance of the database.

### 1) Slow Query log-

```

493 Select * from mysql.slow_log
494 Select * from mysql.general_log
495
496

```

### 2) General Query Log-

```

493 Select * from mysql.slow_log
494 Select * from mysql.general_log
495
496
497
498
499
500
501
502

```

event_time	user_host	thread_id	server_id	command_type	argument
2023-03-20 01:37:15.521248	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:15.522574	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:15.523263	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:15.523686	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:16.396553	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:17.396665	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:18.396768	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:19.396852	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:20.396903	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:21.396978	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:22.397098	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:23.202854	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Query	SELECT
2023-03-20 01:37:23.397372	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:24.397240	rdsadmin[rdsadmin] @ localhost []	10	1548915789	Execute	SELECT
2023-03-20 01:37:24.600777	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:24.601420	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:24.602256	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:24.602741	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT
2023-03-20 01:37:24.604040	rdsadmin[rdsadmin] @ localhost [127.0.0.1]	8	1548915789	Query	SELECT

## V. RESULTS

### A. User Interface output:

User Interface will have a list of options that are discussed in the functional requirements. A few sample output is shown here:

#### 1) Most viral tweets

method_id	method_type
1	Get most viral tweet
2	Get most used hashtags
3	Get most used source
4	Get most mentioned user
5	Get most active users in descending order
6	Get most liked tweet about chatgpt
7	Get most used media type
8	Get users with more No of retweets
9	Get most discussed tweet with respect to replies based on conversation id
10	number of Tweets on Chat GPT
11	Get most retweeted tweet in each language
12	Get top 10 Users who have more outsource for reference on the tweet
13	Get conversations with most number of media shares
14	Get number of tweets in a day on Chatgpt
15	Get most retweeted tweet with the comments
exit	Exits program

method_id	method_type
1	Get most viral tweet
2	Get most used hashtags
3	Get most used source
4	Get most mentioned user
5	Get most active users in descending order
6	Get most liked tweet about chatgpt
7	Get most used media type
8	Get users with more No of retweets
9	Get most discussed tweet with respect to replies based on conversation id
10	number of Tweets on Chat GPT
11	Get most retweeted tweet in each language
12	Get top 10 Users who have more outsource for reference on the tweet
13	Get conversations with most number of media shares
14	Get number of tweets in a day on Chatgpt
15	Get most retweeted tweet with the comments
exit	Exits program

#### 2) Most used hashtags

Hashtag	Hashtag_count
#chatgpt	7723
#AI	1884
#chatgpt	1346
#OpenAI	741
#AI	614
#chatGPT	556
#ArtificialIntelligence	537
#Microsoft	448
#OpenAI	247
#AI	224

method_id	method_type
1	Get most viral tweet
2	Get most used hashtags
3	Get most used source
4	Get most mentioned user
5	Get most active users in descending order
6	Get most liked tweet about chatgpt
7	Get most used media type
8	Get users with more No of retweets
9	Get most discussed tweet with respect to replies based on conversation id
10	number of Tweets on Chat GPT
11	Get most retweeted tweet in each language
12	Get top 10 Users who have more outsource for reference on the tweet
13	Get conversations with most number of media shares
14	Get number of tweets in a day on Chatgpt
15	Get most retweeted tweet with the comments
exit	Exits program

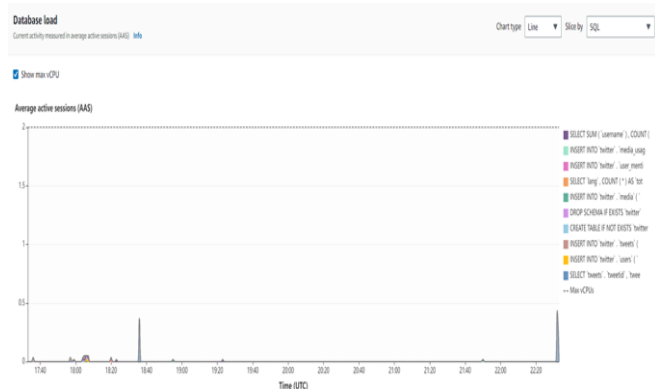
```

Enter the method name: 9
##### Get most Discussed tweet with respect to replies based on conversation id #####
method_id      tweetId      username      reply_to_tweet
1371823551312000000  ChatGPT passed a shartan RMA exam w/vLwLw vLwLw  @Gh0stcr  154
1371838544180100000  ChatGPT is free @vLwLw's x/y made this free...  chatgpt_isfree  328
1372293335452400000  Pretty much said that chatgpt has passed the test...  nro_aliddad  104
1371751742607400000  ChatGPT has passed w/vLwLw. United States medical...  @miller1  141
1372293335452400000  I think we know fully about the fact that ChatGPT...  @miller1  104
137121080123000000  Lehrer will not be hounded, said scholar here ...  @brn  87
1371806000000000000  ChatGPT is an artificial intelligence search tool  @Gh0stcr  41
1371838544180100000  ChatGPT just introduced a $42/month plan w/vLwLw  @Gh0stcr  41
1371838544180100000  Now the ChatGPT generated tweets sign ml...  wLwLwLwLw  54
1372977272207000000  I remain a little baffled by this. Every time ...  @mattw0_dgreen  48

##### Get most retweeted tweet in each language #####
method_id      method_type
1  Get most viral tweet
2  Get most used hashtags
3  Get most used source
4  Get most mentioned user
5  Get most active users in descending order
6  Get most liked tweets about chatgpt
7  Get most used media type
8  Get users with more fb of retweets
9  Get most Discussed tweet with respect to replies based on conversation id
10  number of tweets on Chat GPT
11  Get most retweeted tweet in each language
12  Get top 10 users who have more outcome for reference on the tweets
13  Get conversations with most number of media shares
14  Get number of tweets in a day on chatgpt
15  Get most retweeted tweet with the comments
exit  I exito program.

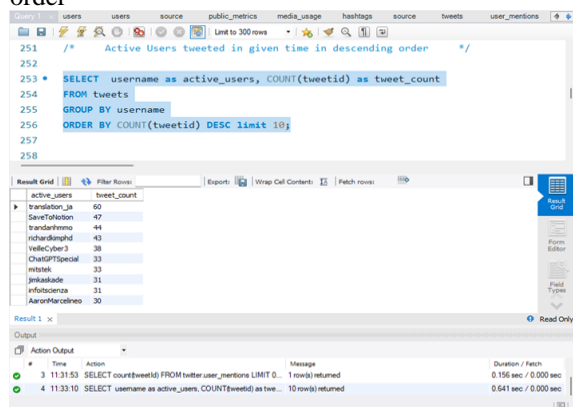
```

The screenshot displays the AWS CloudWatch console interface. At the top, there's a navigation bar with 'AWS' logo, 'CloudWatch' text, and a 'Last 5 hours' filter. Below this, the 'Counter metrics' section is active, showing three line graphs. The first graph, 'db.532.Com.selecting', shows a high, fluctuating value around 1.0. The second graph, 'db.532.Queries', shows a lower, fluctuating value around 0.5. The third graph, 'db.532.Slow.queries', shows a very low value with a single sharp spike at 18:00. The x-axis for all graphs is 'Time (UTC)' ranging from 17:00 to 23:00. The y-axis for the first graph is 'Queries' ranging from 0.0 to 1.0. The y-axis for the second graph is 'Queries' ranging from 0.0 to 1.0. The y-axis for the third graph is 'Queries' ranging from 0.0 to 1.0.

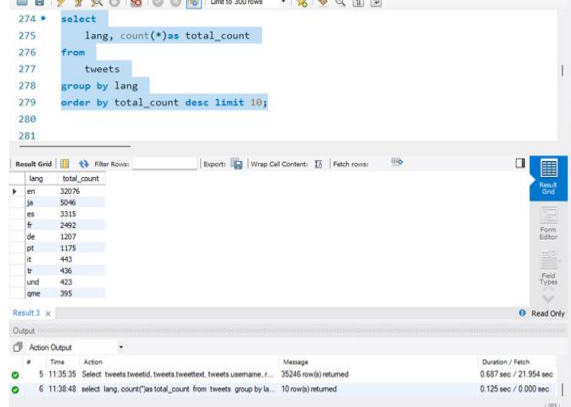


Load by sql (s/s)	SQL statements	Calls/sec	Avg latency (m/s)	Rows examined (all)
0.01	SELECT tweets `tweetid`, `tweets` `tweettext`, `tweets` `username` ..	0.00	-	-
<0.01	INSERT INTO `twitter`.`users` (`username`, `User_id`, `userbio` ) VALUES ..	0.00	70.76	0.00
<0.01	INSERT INTO twitter.users(username, `User_id`, `userbio` )VALUES (tweettext)..	-	-	-
<0.01	INSERT INTO twitter.users(username, `User_id`, `userbio` )VALUES (tweettext)..	-	-	-
<0.01	INSERT INTO twitter.users(username, `User_id`, `userbio` )VALUES (tweettext)..	-	-	-
<0.01	INSERT INTO `twitter`.`users` (`tweetid`, `username`, `tweettext`, `cont` ..	0.00	-	-
<0.01	CREATE TABLE IF NOT EXISTS `twitter`.`media_usage` (`tweetid` INT(4) NOT NULL ..	0.00	42.55	0.00
<0.01	DROP SCHEMA IF EXISTS `twitter`	0.00	216.41	0.00
<0.01	INSERT INTO `twitter`.`media` (`mediaid`, `mediaip`, `mediacreate` ) VAL ..	0.00	93.69	0.00
<0.01	SELECT lang, COUNT(*) AS total_count FROM `twitter` GROUP BY lang ORDER ..	-	-	-
<0.01	CALL `GET_HIGHTAG_COUNT`()	0.00	-	-
<0.01	INSERT INTO `twitter`.`user_mentions` (`tweetid`, `username` )VALUES (..) ..	-	-	-
<0.01	INSERT INTO `twitter`.`hashtags_usage` (`tweetid`, `hashtagid` )VALUES (..) ..	0.00	29.34	0.00

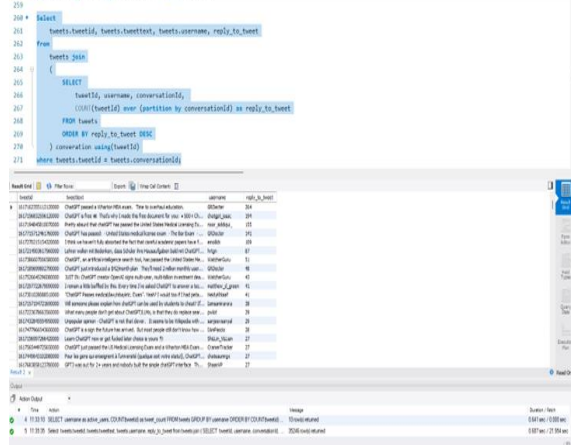
1. Active Users tweeted in given time in descending order



Query 1 x users users source public\_metrics media\_usage hashtags source tweets user\_mentions



users users source public\_metrics media\_stats heartings source loads user\_metrics user\_metrics



#### 4. Most Liked Tweet about ChatGPT

Query 1: users, source, public\_metrics, media\_usage, hashtags, source, tweets, user\_mentions

```

321 -- Most Liked Tweet about ChatGPT
322
323 * select tweetid, username, tweettext, retweetcount, likecount from tweets join public_metrics using (tweetid)
324 order by likecount desc;

```

tweetid	username	tweettext	retweetcount	likecount
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575

#### 5. Top 10 Users who have more outsource for reference on the tweet

Query 1: users, source, public\_metrics, media\_usage, hashtags, source, tweets, user\_mentions

```

322 -- top 10 Users who have more outlinks
323 * select username, count(outlink) as 'Number of links'
324 from tweets join tweet_links using (tweetid)
325 group by username
326 order by count(outlink) desc
327 limit 10;

```

username	Number of links
translation_ja	145
M157c_NewsRSS	52
shodasii	51
Artisist_j	50
mlayay_takashi	49
g_stakur	49
McJourneyAI	49
richardmoph	43
StdPro	40
YelleCyber3	38

#### 6. Most retweeted tweet in each language

Query 1: users, source, public\_metrics, media\_usage, hashtags, source, tweets, user\_mentions

```

321 -- Most retweeted tweet in each language
322
323 * select tweetid, lang, username, tweettext, retweetcount,
324 likecount over (partition by 'lang' order by retweetcount desc) as rnknum
325 from tweets join public_metrics using (tweetid)
326 select * from languages;
327 where rnknum <= 1;
328 order by retweetcount desc;
329
330 -- top 10 users who have more outlinks

```

tweetid	lang	username	tweettext	retweetcount	likecount
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575
1674232222222	en	g00000	ChatGPT is a game changer. This is a real revolution.	22412	94575

#### 7. Number of tweets each day on ChatGPT

Query 1: users, source, public\_metrics, media\_usage, hashtags, source, tweets, user\_mentions

```

335
336 -- Number of tweets each day on chatGPT
337 * select date(createdtime) as Tweeted_date, count(tweetid) as tweet_count
338 from tweets
339 group by Tweeted_date;

```

Tweeted_date	tweet_count
2023-01-22	10000
2023-01-23	31700
2023-01-24	8233

#### 8. Most Used Media Type

Query 1: users, source, public\_metrics, media\_usage, hashtags, source, tweets, user\_mentions

```

289
290 /* Most Used Media Type */
291
292 * select mediatype, count(*) as media_used from media
293 group by mediatype
294 order by media_used desc;
295
296 /* User who have used more Hashtags */
297
298

```

mediatype	media_used
Photo	35115
Video	518
Gf	405

#### 9. Conversations with Most Number of Media Shares

Query 1: users, source, public\_metrics, media\_usage, hashtags, source, tweets, user\_mentions

```

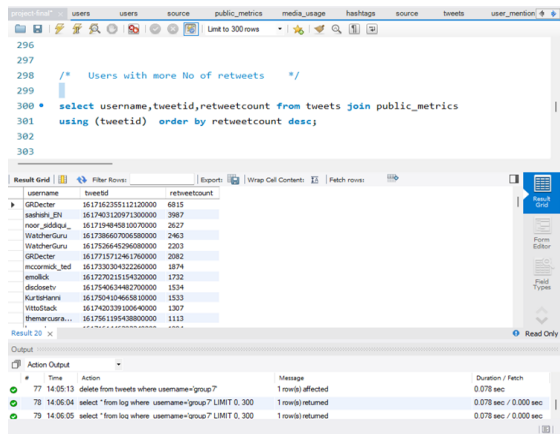
301
302 -- conversations with most number of media shares
303 * select conversationid, mediatype, count(mediaid) as 'number of media shared'
304 from tweets join media_usage using (tweetid) join media using (mediaid)
305 group by conversationid, mediatype
306 order by count(mediaid) desc
307 limit 10;
308

```

conversationid	mediatype	number of media shared
16177000002910000	Photo	21
16175402003860000	Photo	13
161728990389170000	Photo	12
16173706644600000	Photo	8
161796538975300000	Photo	9
161775736974300000	Photo	8
161763559351800000	Photo	8
161742384554950000	Photo	8
1617475796976930000	Photo	7
161728024928200000	Photo	7



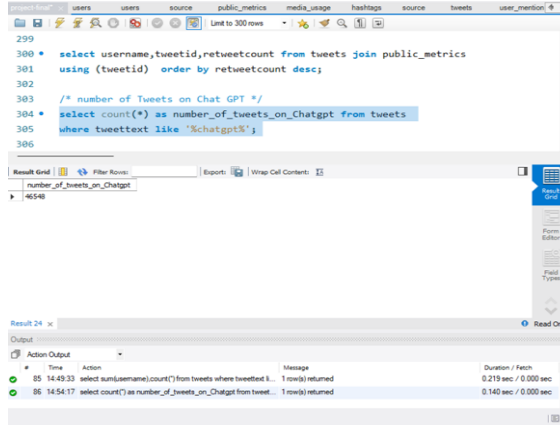
## 10. Users with higher retweets



```
296
297
298 /* Users with more No of retweets */
299
300 select username,tweetid,retweetcount from tweets join public_metrics
301 using (tweetid) order by retweetcount desc;
302
303
```

username	tweetid	retweetcount
GRDecker	1617462255112100000	6815
seaship_gi	161746212973100000	3987
noor_jidda_u	161734945810070000	2627
WahcherGuru	161738667096080000	2463
WahcherGuru	161732644529660000	2203
GRDecker	1617715712461760000	2082
recomend_ded	161733204222200000	1874
enashik	1617292215154520000	1732
dicodestv	1617540634482700000	1534
Kurhanova	161729404646810000	1533
Vitostack	1617420339100640000	1307
hemamusa...	1617561195438800000	1113

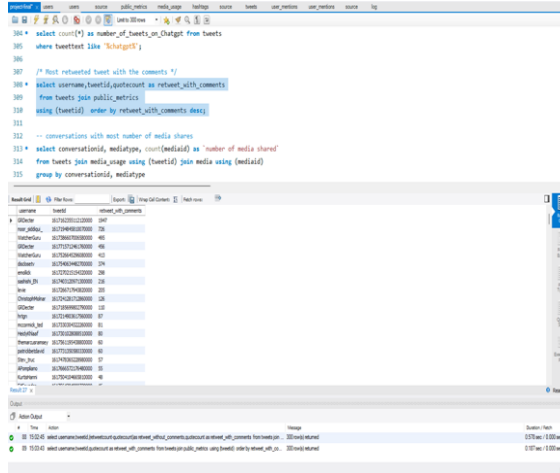
## 11. Number of Tweets on Chat GPT



```
299
300 select username,tweetid,retweetcount from tweets join public_metrics
301 using (tweetid) order by retweetcount desc;
302
303 /* number of Tweets on Chat GPT */
304 select count(*) as number_of_tweets_onChatgpt from tweets
305 where tweettext like '%chatgpt%';
306
```

number_of_tweets_onChatgpt
46548

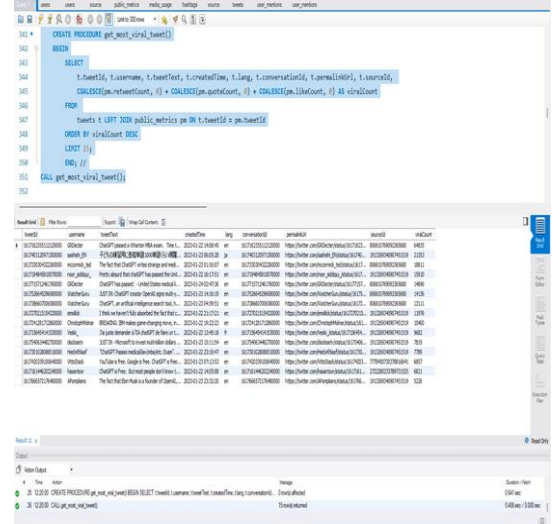
## 12. Most retweeted tweet with the comments



```
307 /* Most retweeted tweet with the comments */
308 select username,tweetid,retweetcount as retweet_with_comments
309 from tweets join public_metrics
310 using (tweetid) order by retweet_with_comments desc;
311
312 -- conversations with most number of media shares
313 select conversationid,mediatype, count(mediaid) as 'number of media share'
314 from tweets join media_usage using (tweetid) join media using (mediaid)
315 group by conversationid, mediatype
```

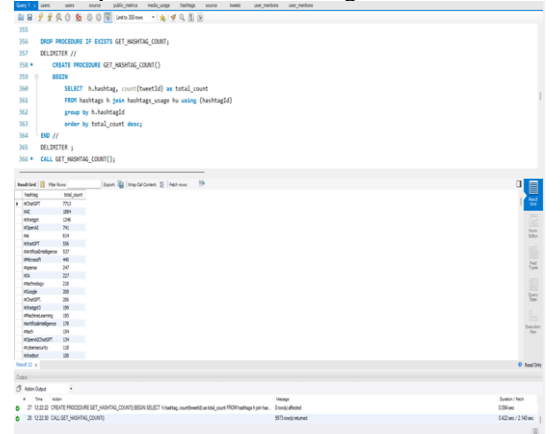
username	tweetid	retweetcount
GRDecker	1617462255112100000	6815
seaship_gi	161746212973100000	3987
noor_jidda_u	161734945810070000	2627
WahcherGuru	161738667096080000	2463
WahcherGuru	161732644529660000	2203
GRDecker	1617715712461760000	2082
recomend_ded	161733204222200000	1874
enashik	1617292215154520000	1732
dicodestv	1617540634482700000	1534
Kurhanova	161729404646810000	1533
Vitostack	1617420339100640000	1307
hemamusa...	1617561195438800000	1113

## 13. Stored Procedure Most Viral Tweet



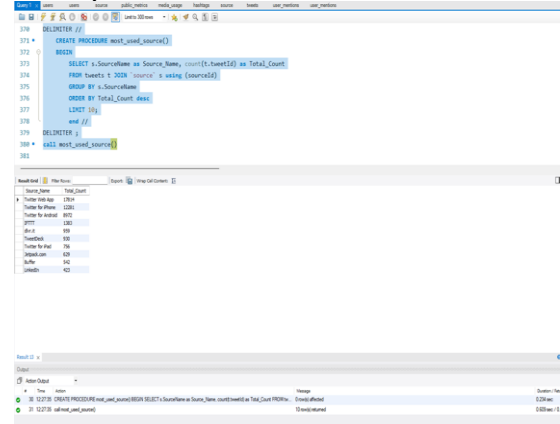
```
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

## 14. Stored procedure most hashtag used



```
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

## 15. Stored procedure most used source



```
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```

154 DELIMITER //
155 * CREATE PROCEDURE most_mentioned_user()
156 BEGIN
157     select username, count(userid) as Total_Mentions
158     from user_mentions
159     group by username
160     order by Total_Mentions desc
161     limit 1;
162 end //
163 DELIMITER //
164 call most_mentioned_user();
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```
907 SELECT * FROM user_mentions where tweetid = old_tweetid; -- Delete from user mentions table
908
909 DELETE FROM hashtag_usage where tweetid = old_tweetid; -- Delete from hashtag usage table
910
911 DELETE FROM media_links where tweetid = old_tweetid; -- Delete from media usage table
912
913 DELETE FROM public_metrics where tweetid = old_tweetid; -- Delete from public metric table
914
915 DELETE FROM link_lists where tweetid = old_tweetid; -- Delete from tweet links table
916 INSERT INTO logs (=CONJ(), user(), old_tweetid,old_tweetname);
917 END//
918
919 DELIMITER ;
920
921 insert into twitter.(username, User_ID) values ('group','https://twitter.com/group');
922
923 insert into twitter.tweets values
924 ('@GUSTAVOXXXXXXXXXXXXX group', 'text about a change!', '2020-01-21 00:00:00', '1617037527000000000',
925 'https://www.twitter.com/gustavoxxxxxx/1617037527000000000', '202001210000000000');
926
927 select * From tweets where tweetid=1617037527000000000;
```

The screenshot shows the SQL Workbench interface. The top pane displays the executed SQL queries, which are identical to those shown in the code block above. The bottom pane shows the results of the last query as a table:

ID	USER	TWEETID	TEXT	URLS	DATE_CREATED	RETWEETS	FOLLOWERS	LIKES
1	@GUSTAVOXXXXXXXXXXXXX group	1617037527000000000	text about a change!	https://www.twitter.com/gustavoxxxxxx/1617037527000000000	2020-01-21 00:00:00	0	0	0

```

--> submit_queries user_queries
Line to 200 rows
282 drop trigger if exists trigger_tweet_length;
283 DELIMITER //
284 * CREATE TRIGGER trigger_tweet_length
285 BEFORE Insert ON tweets
286 FOR EACH ROW
287 BEGIN
288     IF LENGTH(MEDIAtweetext) > 280 THEN
289         SIGNAL SQLSTATE '45000'
290             SET MESSAGE_TEXT = "tweet text cannot be longer than 280 characters";
291     END IF;
292 END//
293 Insert into tweets (tweetid,tweettext) values (1619508057289200000,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
294 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa);
295 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa);
296 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa);
297 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa);
298 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa);
299

```

**Output**

Time	Action	Message	Duration / Fetch
0   04:42:35	CREATE TRIGGER trigger_tweet_length BEFORE new ON tweets FOR EACH ROW BEGIN ...	Event(s) affected	0.172 sec
1   04:42:39	Insert into tweets (tweetid,tweettext) values (1619508057289200000, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...)	Error Code 1644: Tweet text cannot be longer than 280 characters	0.594 sec
2   04:42:39	Insert into tweets (tweetid,tweettext) values (1619508057289200000, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...)	Error Code 1644: Tweet text cannot be longer than 280 characters	0.594 sec
3   04:42:39	Insert into tweets (tweetid,tweettext) values (1619508057289200000, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...)	Error Code 1644: Tweet text cannot be longer than 280 characters	0.078 sec

[illegible]

```
4051 user       source      public_metrics      media_image      hashtag         source         tweets         user_mentions
4052 DROP TRIGGER IF EXISTS prepare_before_deleting_tweet;
4053 DELIMITER //
4054 * CREATE TRIGGER prepare_before_deleting_tweet
4055 BEFORE DELETE ON tweets
4056 FOR EACH ROW
4057 BEGIN
4058     DELETE FROM user_mentions WHERE tweetid = OLD.tweetid; -- Delete from user_mentions table
4059     DELETE FROM hashtag_image WHERE tweetid = OLD.tweetid; -- Delete from hashtag_image table
4060     DELETE FROM media_image WHERE tweetid = OLD.tweetid; -- Delete from media_image table
4061     DELETE FROM public_metrics WHERE tweetid = OLD.tweetid; -- Delete from public_metrics table
4062     DELETE FROM tweet_likes WHERE tweetid = OLD.tweetid; -- Delete from tweet_likes table
4063 END//
4064 DELIMITER ;;
4065
4066 delete from tweets where tweetid = 1517175688172709679;
4067 select * from tweets where tweetid = 151715688172709679;
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4
```

```
140 DELIMITER //
141
142 -- CREATE TRIGGER prepare_before_dropping_tweet
143 Before delete ON tweets FOR EACH ROW
144 BEGIN
145     DELETE from user_mentions where tweetid = old_tweetid; -- Delete from user_mentions table
146     DELETE from hashtags_usage where tweetid = old_tweetid; -- Delete from hashtags_usage table
147     DELETE from media_usage where tweetid = old_tweetid; -- Delete from media_usage table
148     DELETE from public_metrics where tweetid = old_tweetid; -- Delete from public_metrics table
149     DELETE from tweet_links where tweetid = old_tweetid; -- Delete from tweet_links table
150     INSERT INTO log values (now(), user(), old_tweetid,old_username);
151 END//
152 DELIMITER ;;
153
154 insert into users(username, user_id) values ('group7','https://twitter.com/group7');
155
156 -- (11311571200000000, 'group7', 'text data on chatgpt', '2023-06-26 16:45:16', '1', 11311571200000000,
157 -- https://www.twitter.com/jamba/status/1517512514000000000, 2000040347480100000, 0)
158 select * from tweets where tweetid=1517512514000000000;
159
160 delete from tweets where username='group7';
161
162 select * from log where username='group7';
```

Before delete

## VI. CONCLUSION

ChatGPT App is a cool app that provides many metrics, but a user can only access static queries, in the future it can be made into a dynamic query application. When more data is available the data loading part can be automated to execute at the scheduled time and push data into the cloud AWS RDS Database Instance on a regular basis. The next round of analysis can involve machine learning algorithms to make predictions and perform sentiment analysis on the tweet text.