

Sl. No.	Table of Contents
1.	Problem Statement
2.	Solution Requirements
3.	Limitations
4.	Conceptual Database Design
5.	Functional Analysis
6.	Table Structure & SQL <ul style="list-style-type: none"> <li>● Queries</li> <li>● Triggers</li> <li>● Stored Procedures</li> <li>● Access Privileges</li> </ul>
7.	AWS Migration <ul style="list-style-type: none"> <li>● Logging</li> </ul>
8.	SQL Performance Measurement

# SJSU ChatGPT Tweets Analysis

## I. PROBLEM STATEMENT

Since ChatGPT's launch, it has spread rapidly across multiple industries, particularly the technology sector, and throughout the entire world. Many companies, from small startups to multibillion-dollar corporations, are increasing their workforce in an effort to develop a competitive strategy that will replicate this product's functionality. However, the public has had a variety of reactions, with some seeing this move as a threat to many sectors of the workforce because it replaces people and causes massive global unemployment and others seeing it as a huge step in progress towards Artificial General Intelligence.

It is crucial for businesses to evaluate and develop strategies to compete with them in order to analyze and produce a thorough report on ChatGPT and how to estimate its public popularity. To accomplish all of this in a short amount of time while minimizing revenue loss is a challenging task. Using sentiment analysis of tweets about ChatGPT that were posted by members of the general public is one effective and unbiased way to accomplish this. We have built analysis application system called SJSUChatGPT Tweets analysis. Hashtags help group Tweets and conversations around a similar topic so people can easily find and follow what interests them. So, when someone clicks on or searches a specific hashtag, they will be able to find all the profiles and public posts that use that hashtag. Which is required to get information about users who use ChatGPT as a hashtag in their tweets. To achieve this, we must process the data, save the pertinent features, and produce insightful conclusions quickly. Businesses can make informed decisions and target their products more precisely and potentially with less time to market based on the results. We can search through the noise to find the posts that matter the most which is #chatgpt.

.

## II. SOLUTION REQUIREMENT

- This system is built for companies to help them monitor twitter, one of the largest and most active social media platforms, for competitor analysis.
- The company would be able to identify audience preferences, potential customers, current trends and threats.
- The system would allow them to know the tweet count based off of a particular hashtag.
- In addition, the number of tweets from a user about a particular topic, the time zone and country and language most engaged about a particular tweet subject, most popular users related to topic of interest (users with most like/retweet counts) can be known.
- The system would also provide opportunity to analyze how the company's competitors are doing, what their strategies are, and what would the company's counter strategies be.

- This would also branch into new age of marketing and creating impact in audience and growing revenue of businesses.
- Since the company is using language and country as filters, it can segregate strategies and alliances specific to the country/language.

## III. LIMITATIONS

- This application does not have access to user's personal information such as mobile number, email, location/geographical coordinates.
- The application doesn't enable users to edit or delete their previous tweets.
- The specified dataset has tweets data of only 3 days (from 22<sup>nd</sup> Jan to 24<sup>th</sup> Jan) however more data can be added as and when necessary.
- The user's credentials are not being stored right now. So, there is no way to retrieve lost account.
- The database is not scalable right now and multi region availability is not enabled.

## IV. CONCEPTUAL DATABASE DESIGN

We have a dataset containing data and metadata of tweets posted in various languages with wide variety of hashtags concentrated on a single topic ChatGPT. Based on the initial analysis and requirements, we have defined a relational schema for this application.

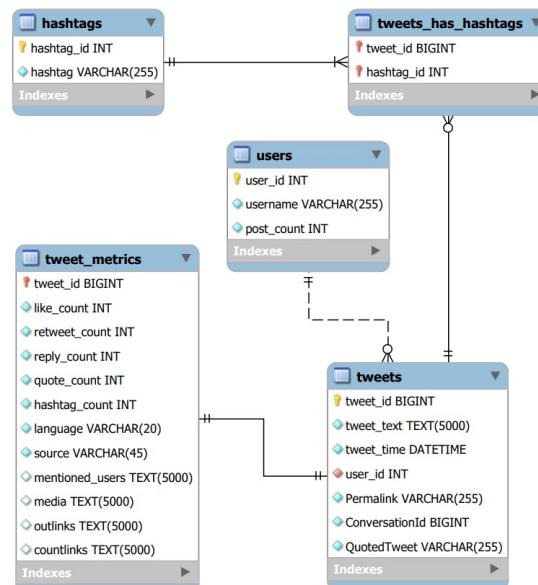


Fig.1 – Entity Relationship Diagram

Our chatgpt database has five entities.

1. **Users:** This entity contains details of all users signed up on twitter. It has following attributes.
  - **user\_id** (INT, PRIMARY KEY) – This attribute contains unique ID of each user.

- username (VARCHAR) – It is the username of each user.
  - post\_count (INT) – This is the number of tweets published by the user in twitter till date.
2. **Tweets:** This entity contains details of each tweet such as tweet content, published time, author, tweet URL etc.
- tweet\_id (BIGINT, PRIMARY KEY) – All tweets are assigned with a unique tweet ID.
  - tweet\_text (TEXT) – This field contains the tweet content
  - tweet\_time (DATETIME) – Time when the tweet was published.
  - user\_id (INT) – The user\_id of tweet's author
  - Permalink (VARCHAR) – This field contains the URL for the tweet.
  - ConversationId (BIGINT) – It contains the tweet\_id of the parent tweet if the given tweet is a reply to parent tweet.
  - QuotedTweet (VARCHAR) – It contains the quoted text after retweeting the original post.
3. **Hashtags:** All the hashtags used in this dataset are modeled in this entity.
- hashtag\_id (INT) - Contains a unique id corresponding to each hashtag that is used in a tweet.
  - hashtag (VARCHAR) – This attribute shows the hashtag used.
4. **Tweet\_Metrics:** This entity refers to a set of quantitative measures and statistics which are associated with each tweet. These metrics provide information about the engagement and reach of the tweet, as well as the level of interaction between users.
- tweet\_id (BIGINT, FK) – Each tweet is given a unique tweet\_id
  - like\_count (INT) – Shows the number of likes for each tweet
  - retweet\_count (INT) – Shows the number of retweets for each tweet
  - reply\_count (INT) – Shows the number of direct replies for a given tweet
  - quote\_count (INT) – Refers to the number of times a tweet has been quoted by other users.
  - hashtag\_count (INT) – Contains number of hashtags used in the tweet.
  - language (VARCHAR) – This attribute contains 2 letter ISO 639-2 language code based on the language used in that particular tweet.
  - source (VARCHAR) – It refers to the application or platform from where the tweet has been created.
  - mentioned\_users (TEXT) – Contains details of users mentioned in the tweet
  - media (TEXT) – This field describes any media content that is attached to a tweet such as Image, Video or GIFs.
  - outlinks (TEXT) – This contains list of all direct links attached in the tweet.
  - countlinks (TEXT) – This is same as outlinks but has shortened links for click analytics.
5. **Tweets\_Has\_Hashtags:** This is a bridge entity that is created between tweets and hashtags entities to resolve many-to-many relationship between those two. It has foreign keys from both tables.
- tweet\_id (BIGINT) – References Id of a tweet.
  - hashtag\_id (INT) – Contains ID of a hashtag used in tweet.
  - A user can post zero or many tweets but each tweet is posted by only one user.
  - A tweet can have zero or many hashtags and a hashtag can be used by zero or more tweets.
  - Tweet metrics exists only if the tweet exist. Hence both the tables are mapped one-to-one cardinality.

**Normalization:** For normalization of our database, we did data preprocessing and transformation using Python and divided all the entities and their respective attributes to normalize the database to 3NF by removing the partial as well as transitive dependencies.

## V. FUNCTIONAL ANALYSIS

The database is created for a Company, looking to create strategies based on tweets.

The database has the following functional components:

### DATA:

Data is the information that needs to be changed and processed in order to have any real meaning. To hold the names and attributes of the data elements that are being used in this, a database dictionary is required. A database maintains metadata, which is information that describes the data. This facilitates the management and storage of data inside a database.

The dataset which is being referred and used for analysis is the ChatGPT Twitter Dataset is an open Data from Kaggle. We have used this dataset to create our own database, created the table structure and updated the database with data.

When a twitter user tweets about something along with ‘ChatGPT’, it will be recorded in our system. We record the Date and time, Tweet Id assigned to the tweet, the text and user name. We even record any external links attached in the tweet, the replies received by the tweet, number of times it was retweeted, number of likes received by the tweet, number of times the tweet has been quoted by other users in a conversation, The language in which the tweet was written and from which kind of device it has been made (be it an iPhone or laptop or android), the links to all the quotes and original tweet. We also record any mentions of other people made by the user in that tweet along with the hashtags and their counts.

This would give us the essential data required to build our Entities.

Using this data, we build a relationship model, which would help the company access the following information  
**COMPANY:** We give the company a

- user engagement rate, which entails how much influence a user has over others in a discussion. (This would help the company find potential influencers to advertise their product).
- Retweet link, mentioning how many users have attached a retweet (external) link to their tweet. (To get information about other systems or competitors).
- The top ten people who have tweeted the most in a discussion.
- Which time of the day has most tweets posted.
- How many times is person (usually a celebrity) has been mentioned (usually to know which person's endorsement can help or hurt their product the most).
- Users who are most active on twitter regarding the ChatGPT topic.
- The tweet which has got most attention.
- Hashtags used prominently along with ChatGPT.
- The most famous tweet. (Can be used for analysis about the product's potential customer)

## VI. TABLE STRUCTURE AND SQL

### A. Queries

#### [1] Query to get users most discussing on twitter

```
SELECT users.user_id, users.username,
COUNT(tweets(tweet_id)) AS tweet_count
FROM users JOIN tweets ON users.user_id =
tweets.user_id
GROUP BY users.user_id
ORDER BY tweet_count DESC
LIMIT 50;
```

user_id	username	tweet_count
37075	translation_ja	60
12388	SaveToNotion	47
919	trandanhammo	44
10899	richardkimphd	43
29366	VeilleCyber3	38
4860	mitstek	33
2705	ChatGPTSpecial	33
13554	ilmaskarla	31

Fig.2 – Query 1 Output

#### [2] Query to get most viral tweets (wrt retweets)

```
SELECT tweets(tweet_id), tweets(tweet_text),
users.username,
SUM(tweet_metrics.like_count +
tweet_metrics.retweet_count +
tweet_metrics.reply_count) AS
tweet_total_engagement
FROM tweets
JOIN users ON tweets.user_id = users.user_id
JOIN tweet_metrics ON tweets(tweet_id) =
tweet_metrics(tweet_id)
```

```
GROUP BY tweets(tweet_id)
ORDER BY tweet_total_engagement DESC;
```

tweet_id	tweet_text	username	tweet_total_engagement
1617162355112124421	ChatGPT passed a Wharton ...	GRDector	66256
1617403597309057	子化の新規用語を単語10... sashih1_EN		21367
161750410465812994	Please don't ChatGPT ...	ChatGPT_b0ck_tad	18879
1617194864810077657	Please ignore ChatGPT ...	noor_siddiqui_1d	16406
1617152664929602945	ChatGPT has passed - Uni...	GRDector	1573
1617386607006584832	JUST IN: ChatGPT creator ...	WatcherGuru	14626
1617386607006584832	ChatGPT, an artificial intellige...	WatcherGuru	13557

Fig.3 – Query 2 Output

#### [3] Query to fetch top 20 hashtags used by most number of tweets in decreasing order

```
SELECT tweets_has_hashtags.hashtag_id,
hashtags.hashtag,
COUNT(*) AS frequency
FROM tweets_has_hashtags
JOIN hashtags ON tweets_has_hashtags.hashtag_id =
hashtags.hashtag_id
GROUP BY tweets_has_hashtags.hashtag_id
ORDER BY frequency DESC LIMIT 20;
```

hashtag_id	hashtag	frequency
1	#chatgpt	9462
1125	#ai	2543
7620	#openai	1013
8802	#artificialintelligence	723
3665	#microsoft	507
8950	#chatgpt	311
424	#technology	303

Fig.4 – Query 3 Output

#### [4] Query to get most discussed tweet with respect to replies and conversations

```
SELECT t(tweet_id), t(tweet_text),
COUNT(DISTINCT t2(tweet_id)) AS conversation_count,
tm.reply_count,
COUNT(DISTINCT t2(tweet_id)) +
tm.reply_count AS total_count
FROM tweets t
JOIN tweets t2 ON t(tweet_id) = t2.ConversationId
JOIN tweet_metrics tm ON t(tweet_id) =
tm(tweet_id)
GROUP BY t(tweet_id), t(tweet_text),
tm.reply_count
ORDER BY total_count DESC;
```

tweet_id	tweet_text	conversation_count	reply_count	total_count
1617161446202245120	ChatGPT Is Free. But most ...	4	3098	3102
1617507570091958272	I've Created a Complete Gui...	7	3044	3051
1617162355112124421	ChatGPT passed a Wharton ...	264	1421	1685
161750410465812994	ChatGPT is an amazing reso...	10	1455	1465
1617561195438800897	Right now, there are people...	10	1110	1120
1617715712461766567	ChatGPT has passed: - Uni...	141	777	918
1617194864810077657	Pretty absurd that chatGPT...	155	496	651
1617270215154323457	I think we haven't fully abs...	109	476	585

Fig.5 – Query 4 Output

**[5] Query to fetch top 20 platforms(android,iphone,etc) used for chatgpt discussions/tweets with #chatgpt**

```
SELECT tm.source, COUNT(h.hashtag_id) AS
'count of chatgpt hashtag'
FROM tweet_metrics tm JOIN tweets tw ON
tm(tweet_id = tw(tweet_id
JOIN tweets_has_hashtags th ON tw(tweet_id = th(tweet_id
hashtags h ON h.hashtag_id = th.hashtag_id
WHERE hashtag = '#chatgpt'
GROUP BY tm.source
ORDER BY COUNT(h.hashtag_id) DESC
LIMIT 20;
```

source	count of chatgpt hashtag
Twitter Web App	3909
Twitter for iPhone	1998
Twitter for Android	1737
LinkedIn	181
Buffer	179
TweetDeck	172
Twitter for iPad	153
Hootsuite Inc.	131
trancefer	66
IFTTT	64

Fig.6 – Query 5 Output

**[6] Query to find out the average length of tweets in a chatgpt conversion**

```
SELECT AVG(LENGTH(tw(tweet_text))) AS
'average length of tweets'
FROM tweet_metrics tm JOIN tweets tw ON
tm(tweet_id = tw(tweet_id
JOIN tweets_has_hashtags th ON tw(tweet_id = th(tweet_id
ON h.hashtag_id = th.hashtag_id
WHERE h.hashtag = '#chatgpt';
```

average length of tweets
176.0114

Fig.7 – Query 6 Output

**[7] Query to fetch top 10 users who tweeted most with #chatgpt hashtag**

```
SELECT u.user_id, u.username FROM users u
JOIN tweets tw ON
u.user_id=tw.user_id JOIN tweets_has_hashtags th
ON tw(tweet_id=th(tweet_id
JOIN hashtags h ON h.hashtag_id=th.hashtag_id
```

```
WHERE h.hashtag='#chatgpt'
GROUP BY u.user_id
ORDER BY COUNT(u.user_id)
LIMIT 10;
```

user_id	username
8109	Pandey_Jili_
23394	Kosuke_dazo
18242	EricIngram
22824	IsEgaley
13825	napocornejo
19611	JakeMillerTech
23209	marcopiccinini
4545	matwilcox
17743	ruta_adhd
37631	abhishekrungra

Fig.8 – Query 7 Output

**[8] Query to get the tweets that contain external links with chatgpt hashtag**

```
SELECT tm(tweet_id,tw(tweet_text,tm.outlinks
FROM tweet_metrics tm JOIN tweets tw ON tm(tweet_id=tw(tweet_id
JOIN tweets_has_hashtags th ON tw(tweet_id=th(tweet_id
JOIN hashtags h ON h.hashtag_id=th.hashtag_id WHERE
h.hashtag='#chatgpt' AND
tm.outlinks IS NOT NULL AND
CAST(tm.outlinks AS CHAR) <> '';
```

tweet_id	tweet_text	outlinks
1617156308926349312	Schaut Euch an, was @fobiz...	["https://us02web.zoom.us/..."]
1617156404137295878	I created a fictional jewelry ...	["https://oriori.my.canva.site..."]
1617156410634010624	AI will initiate a new era of ...	["https://open.stack.com..."]
1617156441420271618	???? #ChatGPT writes excel...	["https://twitter.com/mtholf..."]
1617157192989265920	Fine-tuning with openAI API...	["https://youtu.be/zvuNayf..."]
1617157309037158400	#sztucznainteligencja #Chat...	["http://schoolab.edu.pl/p..."]
1617157847107739649	https://t.co/vptQpWFnDe ...	["https://chat.openai.com/ch..."]
1617157959682871298	I just posted it on WhatsApp...	["https://twitter.com/BlueEy..."]
1617157975424081920	How #ChatGPT Will Certainl...	["https://talkmarkets.com/co..."]
1617158355541164032	????????? #Russian crimi...	["https://go.theregister.com/..."]
1617158690586648578	Can you take new AI tech #...	["https://www.pressandfour..."]

Fig.9 – Query 8 Output

**[9] Query to retrieve tweets by hour**

```
SELECT DATE_FORMAT(tweet_time, '%Y-%m-
%d %H:00:00') AS hour,
(COUNT(*)) as 'count of tweets' FROM tweets tw
JOIN tweets_has_hashtags th
ON th(tweet_id=tw(tweet_id
JOIN hashtags h ON h.hashtag_id=th.hashtag_id
where h.hashtag='#chatgpt'
GROUP BY hour
ORDER BY hour ASC;
```

```

1 • SELECT DATE_FORMAT(tweet_time, '%Y-%m-%d %H:00:00') AS hour,
2   (COUNT(*)) as 'count of tweets' FROM tweets tw JOIN tweets_has_hashtags th
3   ON th.tweet_id=tw(tweet_id) JOIN hashtags h ON h.hashtag_id=th.hashtag_id
4   where h.hashtag='#chatgpt'
5   GROUP BY hour
6   ORDER BY hour ASC;
7

```

hour	count of tweets
2023-01-22 13:00:00	49
2023-01-22 14:00:00	240
2023-01-22 15:00:00	218
2023-01-22 16:00:00	202
2023-01-22 17:00:00	210
2023-01-22 18:00:00	184
2023-01-22 19:00:00	173
2023-01-22 20:00:00	122
2023-01-22 21:00:00	134
2023-01-22 22:00:00	158
2023-01-22 23:00:00	124

Fig.10 – Query 9 Output

[10] Query is for counting the number of likes that were generated by the use of the hashtag in the tweet.

```

SELECT
    hashtags.hashtag,
    SUM(tweet_metrics.like_count) AS like_count
FROM
    tweets
JOIN
    tweet_metrics ON tweets(tweet_id) = tweet_metrics(tweet_id)
JOIN
    tweets_has_hashtags ON tweets(tweet_id) = tweets_has_hashtags(tweet_id)
GROUP BY hashtags.hashtag
ORDER BY like_count DESC;

```

```

1 • SELECT hashtags.hashtag, SUM(tweet_metrics.like_count) AS like_count
2   FROM tweets JOIN tweet_metrics ON tweets(tweet_id) = tweet_metrics(tweet_id)
3   JOIN tweets_has_hashtags ON tweets(tweet_id) = tweets_has_hashtags(tweet_id)
4   GROUP BY hashtags.hashtag
5   ORDER BY like_count DESC;
6

```

hashtag	like_count
#chatgpt	36083
#ai	6325
#chatgpt.	3337
#microsoft	3153
#openai	2613
#ia	1386
#midjourney	1304
#吸血鬼すぐ死ぬ2	1189
#artificialintelligence	1183
#chatgpt,	1104
#edtech	932
#edchat	749

Fig.11 – Query 10 Output

[11] Query to get the tweet count based on the language

```

SELECT tweet_metrics.language, COUNT(*) AS language_count
FROM tweets JOIN tweet_metrics ON
tweets(tweet_id) = tweet_metrics(tweet_id)
GROUP BY tweet_metrics.language

```

ORDER BY language\_count DESC;

```

1 • SELECT tweet_metrics.language, COUNT(*) AS language_count
2   FROM tweets JOIN tweet_metrics ON tweets(tweet_id) = tweet_metrics(tweet_id)
3   GROUP BY tweet_metrics.language
4   ORDER BY language_count DESC;
5

```

language	language_count
en	32076
ja	5046
es	3315
fr	2492
de	1207
pt	1175
it	443
tr	436
und	423
qme	395
ar	392
nl	319

Fig.12 – Query 11 Output

[12] Query is to get the sources that were used to create the highest number of tweets.

```

SELECT tweet_metrics.source, COUNT(*) AS source_count
FROM tweets JOIN tweet_metrics ON
tweets(tweet_id) = tweet_metrics(tweet_id)
GROUP BY tweet_metrics.source
ORDER BY source_count DESC;

```

```

1 • SELECT tweet_metrics.source, COUNT(*) AS source_count
2   FROM tweets JOIN tweet_metrics ON tweets(tweet_id) = tweet_metrics(tweet_id)
3   GROUP BY tweet_metrics.source
4   ORDER BY source_count DESC;
5

```

source	source_count
Twitter Web App	17814
Twitter for iPhone	12281
Twitter for Android	8972
IFTTT	1383
dlvr.it	959
TweetDeck	930
Twitter for iPad	756
Jetpack.com	629
Buffer	542
LinkedIn	423
Hootsuite Inc.	304
Twitter for Mac	218

Fig.13 – Query 12 Output

## B. Triggers

[1] Trigger to auto increment post count of user in users table for every insert in tweets table

```

DELIMITER $$
CREATE TRIGGER increment_post_count
AFTER INSERT ON tweets
FOR EACH ROW
BEGIN
    UPDATE users SET post_count = post_count+1
    WHERE user_id = new.user_id;
END $$
```

DELIMITER ;

```

Query 1 | tweet_metrics | tweets
1 -- After Trigger is created, tweets data is inserted into table
2
3 • SELECT * FROM users ORDER BY post_count DESC;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch Rows: |
user_id | username | post_count |
37075 | translation_ja | 60
12388 | SaveToNotion | 47
919 | trandanhammo | 44
10899 | richardkimphd | 43
29366 | VelleCyber3 | 38
4860 | miltsek | 33
2705 | ChatGPTSpecial | 33
15682 | infosciienza | 31
13554 | jimkaskade | 31

```

Fig.14 –Trigger 1 Output

### C. Stored Procedures

#### [1] Stored Procedure to fetch the top 20 tweets based on tweet's like, reply, retweet counts for a given language

DELIMITER \$\$

CREATE PROCEDURE get\_top\_tweets\_by\_language (IN language\_name VARCHAR(255))

BEGIN

```

SELECT tweets(tweet_text, tweet_metrics.like_count,
    tweet_metrics.retweet_count, tweet_metrics.reply_count,
    tweet_metrics.quote_count,
    (tweet_metrics.like_count +
    tweet_metrics.retweet_count + tweet_metrics.reply_count +
    tweet_metrics.quote_count) AS total_engagement
FROM tweets
JOIN tweet_metrics ON tweets(tweet_id =
    tweet_metrics(tweet_id)
WHERE tweet_metrics.language = language_name
ORDER BY total_engagement DESC
LIMIT 20;
END $$
```

DELIMITER ;

CALL get\_top\_tweets\_by\_language('es');

```

13
14 • CALL get_top_tweets_by_language('es');

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch Rows: |
tweet_text | like_count | retweet_count | reply_count | quote_count | total_engagement |
El Google del futuro: Pe... | 858 | 252 | 14 | 9 | 1133
Sin googlear, ni preguntarle ... | 644 | 41 | 135 | 10 | 830
Nota Americanas no ChatGP... | 656 | 43 | 20 | 9 | 728
????OFICIAL???? #Microso... | 497 | 93 | 10 | 7 | 607
Cuando te sientes perdida y... | 284 | 30 | 24 | 7 | 345

```

Fig.15 –S.P. 1 Output

#### [2] Stored Procedure to get all tweets tweeted within a specific period of time

DELIMITER //

CREATE PROCEDURE get\_tweet(st datetime,et datetime)

BEGIN

```

SELECT tweet_id, tweet_text, tweet_time
FROM tweets
WHERE tweet_time BETWEEN st AND et;
END //
```

DELIMITER ;

```

CALL get_tweet('2023-01-22 13:00:00','2023-01-22 14:00:00');
```

```

Query 1 | tweets
1 DELIMITER //
2 • CREATE PROCEDURE get_tweet(st datetime,et datetime)
3 BEGIN
4     SELECT tweet_id, tweet_text, tweet_time
5     FROM tweets
6     WHERE tweet_time BETWEEN st AND et;
7 END //
8 DELIMITER ;
9 • CALL get_tweet('2023-01-22 13:00:00','2023-01-22 14:00:00');

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch Rows: |
tweet_id | tweet_text | tweet_time |
1617156270871699456 | ChatGPTで遊びの忘れてた！... | 2023-01-22 13:44:34
1617156291046133761 | @AlexandrovnaIng Prohibiti... | 2023-01-22 13:44:39
1617156308926349312 | Schaut Euch an, was @fobiz... | 2023-01-22 13:44:44
1617156332297256961 | Bow down to chatGPT ?????... | 2023-01-22 13:44:49
1617156345064570880 | Profilinde vatan, Türkiye fal... | 2023-01-22 13:44:52
1617156376983207937 | ChatGPT'nin bilinmeyen ark... | 2023-01-22 13:45:00
1617156389217894400 | ChatGPT runs 10K Nvidia tr... | 2023-01-22 13:45:03
1617156393898745858 | @SWENGDAD There is repe... | 2023-01-22 13:45:04

```

Fig.16 –S.P. 2 Output

#### [3] Stored Procedure to retrieve all tweets that mention a specific user

```

DELIMITER //
CREATE PROCEDURE tweets_by_mentioneduser(m_user VARCHAR(45))
BEGIN
SELECT tw(tweet_id,tw(tweet_text
FROM tweet_metrics tm JOIN tweets tw ON
tw(tweet_id=tm(tweet_id
WHERE mentioned_users LIKE
CONCAT('%',m_user,'%'));
END //
DELIMITER ;
```

CALL tweets\_by\_mentioneduser('SWENGDAD')

```

Query 1 | tweets
1 DELIMITER //
2 • CREATE PROCEDURE tweets_by_mentioneduser(m_user VARCHAR(45))
3 BEGIN
4     SELECT tw(tweet_id,tw(tweet_text
5     FROM tweet_metrics tm JOIN tweets tw ON
6     WHERE mentioned_users LIKE CONCAT('%',m_user,'%');
7 END //
8 DELIMITER ;
9
10 • CALL tweets_by_mentioneduser('SWENGDAD')

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch Rows: |
tweet_id | tweet_text |
1617156393898745858 | @SWENGDAD There is repetitive work in every job, t...
1617234548051546112 | @SWENGDAD @xlr8harder &gt; Asking ChatGPT &gt;...
```

Fig.17 – S.P. 3 Output

#### [4] Stored procedure used to retrieve the users who have used a specific hashtag.

```

DELIMITER //
CREATE PROCEDURE `get_users_using_hashtag` (IN hashtag_name VARCHAR(255))
```

```

BEGIN
    SELECT u.username, t(tweet_text
    FROM tweets t
    INNER JOIN tweets_hashtags thh ON
    t(tweet_id = thh(tweet_id
    INNER JOIN hashtags h ON thh.hashtag_id =
    h.hashtag_id
    INNER JOIN users u ON t.user_id = u.user_id
    WHERE h.hashtag = hashtag_name;
END //
DELIMITER ;
CALL get_users_using_hashtag('#microsoft');

```

```

1  DELIMITER //
2  CREATE PROCEDURE `get_users_using_hashtag` (IN hashtag_name VARCHAR(255))
3  BEGIN
4      SELECT u.username, t(tweet_text
5      FROM tweets t
6      INNER JOIN tweets_hashtags thh ON t(tweet_id = thh(tweet_id
7      INNER JOIN hashtags h ON thh.hashtag_id = h.hashtag_id
8      INNER JOIN users u ON t.user_id = u.user_id
9      WHERE h.hashtag = hashtag_name;
10 END //
11 DELIMITER ;
12 • CALL get_users_using_hashtag('#microsoft');

```

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

username	tweet_text
Itsrohitchouhan	Microsoft Bets Big on the Creator of ChatGPT in Race to Domi...
DrHartmutFeucht	#Google wird nervös #pinchai #ChatGPT #Microsoft https://t...
myselfrosekumar	ChatGPT YouTube shorts in 1 min Click here https://t.co/JEg...
RS_IKing	So everyone asks these questions that #ChatGPT suggests an...
BeardyNerdTweet	Google vs ChatGPT: The Race for AI Dominance Begins https:...
iammannnyj	Here's how Microsoft could use ChatGPT. https://t.co/TQGgW...
egaldus	Was für geller scheiß ist den #ChatGPT bitte? Ich glaube da s...
ketanmyra	#layoffs2023 is really heartbroken for those who spent most o...

Fig. 18 – S.P. 4 Output

### [5] Stored procedure to retrieve a specific user's all posts in his/her account

```

DELIMITER //
CREATE PROCEDURE `get_user_posts` (IN
user_name VARCHAR(255))
BEGIN
    SELECT t(tweet_text
    FROM tweets t
    INNER JOIN users u ON t.user_id = u.user_id
    WHERE u.username = user_name;
END //
DELIMITER ;
CALL get_user_posts('ladypumpkink');

```

```

1  DELIMITER //
2  CREATE PROCEDURE `get_user_posts` (IN user_name VARCHAR(255))
3  BEGIN
4      SELECT t(tweet_text
5      FROM tweets t
6      INNER JOIN users u ON t.user_id = u.user_id
7      WHERE u.username = user_name;
8  END //
9  DELIMITER ;
10 • CALL get_user_posts('ladypumpkink');
11

```

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

tweet_text
@luboweb3 Preparing chatgpt to do it for me ????
@luboweb3 @sergwakesup @Coquicide Meanwhile @TheLabGuy8...
@TheLabGuy8 @luboweb3 @sergwakesup @Coquicide AHAHAHA...
Have you noticed how chatgpt is lazy?

Fig.19 – S.P. 5 Output

### [6] Stored procedure to locate top 10 users who used a specific language to tweet the most.

```

DELIMITER //
CREATE PROCEDURE `get_top_users_by_language` (IN language_name
VARCHAR(255))
BEGIN
    SELECT u.username, COUNT(*) AS count
    FROM tweets t
    JOIN tweet_metrics tm ON t(tweet_id =
tm(tweet_id
    JOIN users u ON t.user_id = u.user_id
    WHERE tm.language = language_name
    GROUP BY u.username
    ORDER BY count DESC
    LIMIT 10;
END //
DELIMITER ;
CALL get_top_users_by_language('es');

```

```

Query 1 tweets
1 BEGIN
2     SELECT u.username, COUNT(*) AS count
3     FROM tweets t
4     JOIN tweet_metrics tm ON t.tweet_id = tm.tweet_id
5     JOIN users u ON t.user_id = u.user_id
6     WHERE tm.language = language_name
7     GROUP BY u.username
8     ORDER BY count DESC
9     LIMIT 10;
10 END //
11 DELIMITER ;
12
13 • CALL get_top_users_by_language('es');

```

username	count
jjyeppez	22
juanmatos	21
alvarocerpa_	12
viadescapetv	11
DiarioTalCual	8
la_patilla	7
LluisMontoliu	7
sebasgueco	7
SquonkTec	7
circularideascom	6

Fig.20 – S.P. 6 Output

[7] Stored procedure to retrieve all hashtags include the given word or letter.

```

DELIMITER //
CREATE PROCEDURE `get_hashtags_by_word` (IN word
VARCHAR(255))
BEGIN
    SELECT hashtags.hashtag
    FROM hashtags
    WHERE hashtags.hashtag LIKE CONCAT('%', word, '%');
END //
DELIMITER ;
CALL get_hashtags_by_word("microsoft");

```

```

1 DELIMITER //
2 CREATE PROCEDURE `get_hashtags_by_word` (IN word VARCHAR(255))
3 BEGIN
4     SELECT hashtags.hashtag
5     FROM hashtags
6     WHERE hashtags.hashtag LIKE CONCAT('%', word, '%');
7 END //
8 DELIMITER ;
9
10 • CALL get_hashtags_by_word("microsoft");

```

hashtag
#microsoft
#copilot...@microsoft
#longmicrosoft
#microsoft
#microsoft-funded
#microsoft,
#microsoft:
#microsoft.
#microsoft's
#microsoft'tan

Fig. 21– S.P. 7 Output

#### D. Access Privileges

We have created 3 users with below mentioned access privileges on our database.

User	Privilege
professor	DB Admin privilege (all access) to our database along with grant permissions.
keerthana	Has limited privileges like querying, inserting new data and modifying existing data across all tables.
tiffany	User with only query privilege. No access to modify data or add data.

```

Query 1 tweet_metrics hashtags tweets tweet_metrics
1 • CREATE USER 'professor'@'%' IDENTIFIED BY 'strongpassword';
2 • GRANT ALL PRIVILEGES ON chatgpt.* TO 'professor'@'%' WITH GRANT OPTION;

Query 1 tweet_metrics hashtags tweets tweet_metrics
1 • CREATE USER 'keerthana'@'%' IDENTIFIED BY 'notsostrong';
2 • GRANT SELECT, INSERT, UPDATE ON chatgpt.* TO 'keerthana'@'%';

Query 1 tweet_metrics hashtags tweets tweet_metrics
1 • CREATE USER 'tiffany'@'%' IDENTIFIED BY 'notweakeither';
2 • GRANT SELECT ON chatgpt.* TO 'tiffany'@'%';

Query 1 tweet_metrics hashtags tweets tweet_metrics
1 • create table demo(id INT, name varchar(25));

```

Fig.22 – Figure showing users and access privileges

## VII. AWS CONNECTIVITY

As part of the Cloud Native implementation of Database, we have created a database instance in AWS RDS (Relational Database Service) with MySQL 8.0.31 version engine. The major reason behind preferring cloud native approach was due to its scalability and security offered. After the instance was created, we can establish its connection from any MySQL Clients like MySQL Workbench. We have used

Python with Embedded SQL to insert the data into the tables after rigorous preprocessing. Finally, we could get some insights from this db instance. We can check the database connection even from python with the below code.

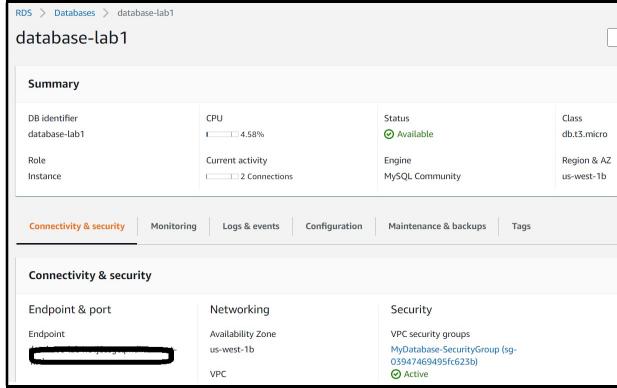


Fig. 23– RDS Instance in AWS

from mysql.connector import connect

```
endpoint = "database-lab1.c4j8ssgcqmdi.us-west-1.rds.amazonaws.com"
try:
    db_connection = connect(host=endpoint, username =
    input("Enter username: "), password=input("Enter
    password: "), database="chatgpt")
except Exception as e:
    print("Error Occurred - {}. Please try again".format(e))

cursor = db_connection.cursor()
query = "SELECT u.username, AVG(tm.like_count) FROM
tweet_metrics tm JOIN tweets t ON tm(tweet_id =
t(tweet_id JOIN users u ON t.user_id = u.user_id GROUP
BY u.username ORDER BY AVG(tm.like_count) DESC"

cursor.execute(query)
results = cursor.fetchall()
print("Top 5 users with highest average like counts..")
print(results[:5])

cursor.close()
db_connection.close()
```

```
Top 5 users with highest average like counts..
[('GRDector', Decimal('24061.0000')),
 ('WatcherGuru', Decimal('10836.5000')),
 ('Veskii ', Decimal('9125.0000')),
 ('mccormick_ted', Decimal('8468.5000')),
 ('sashishi_EN', Decimal('6230.6667'))]
```

Fig. 24 – Query Execution in Python

#### A. Logging:

We have multiple options for enabling logging in AWS RDS databases. We have created a new Parameter group with below features to enable audit, error and slow query logs to be sent to CloudWatch which is a monitoring service for realtime tracking of AWS resources. We have set **log\_output** to **file**, **slow\_query\_log** to 1 and modified our

instance with this parameter group and rebooted the instance. We can view and/or download these logs from CloudWatch console and analyze them.

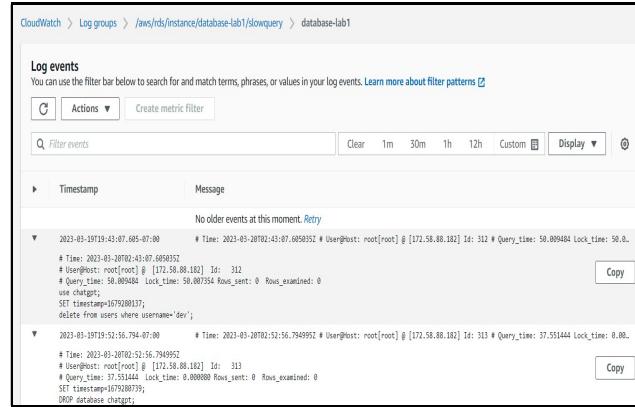


Fig. 25– Slow Query Log Screenshot

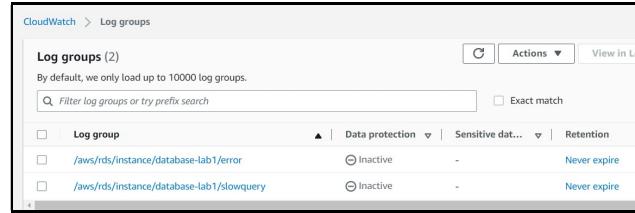


Fig. 26– Log Groups in CloudWatch

## VIII. SQL PERFORMANCE MEASUREMENT

The performance is one of the key factors to keep in mind while designing a database application. A lot of MySQL client applications has inbuilt tools for testing performance of a database. Similarly, MySQL Workbench has a lot to offer for measuring performance and stats of a database. For this, we must set **performance\_schema** to 1 in parameter group attached in RDS database instance and connect to the AWS RDS instance from MySQL interface. Under Performance Reports, we get stats on Memory Usage, Hotspots for I/O, InnoDB statistics, Database Statistics and lot more. It is evident that the top 5% costly operations are indexing, logs flush and other DDL operations. All other subqueries and join statements are relatively faster compared to DDL queries.

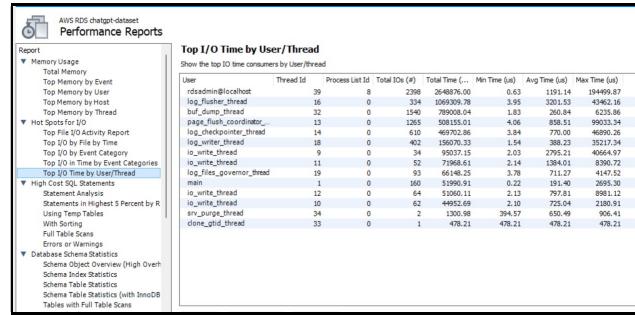


Fig. 27– Performance Report – I

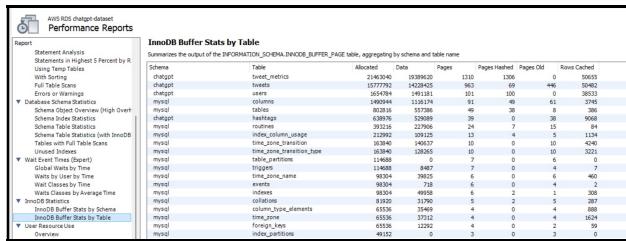


Fig.28 – Performance Report - 2

Also, for queries that are executed frequently, having Stored Procedures can increase the performance slightly. This can be interpreted by comparing the execution time of same queries, one directly and other one as a stored procedure.

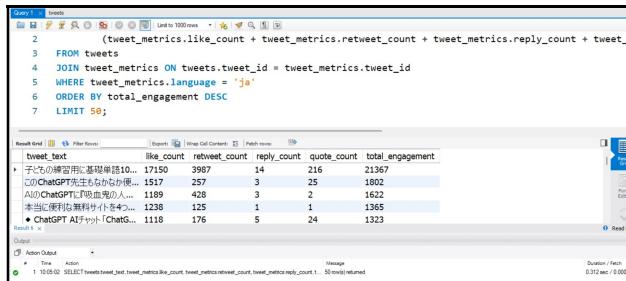


Fig.29 –Query executed directly – 0.312 sec

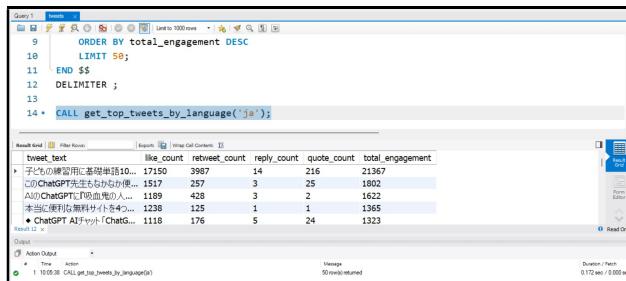


Fig.30– Query executed as a stored procedure – 0.172 sec

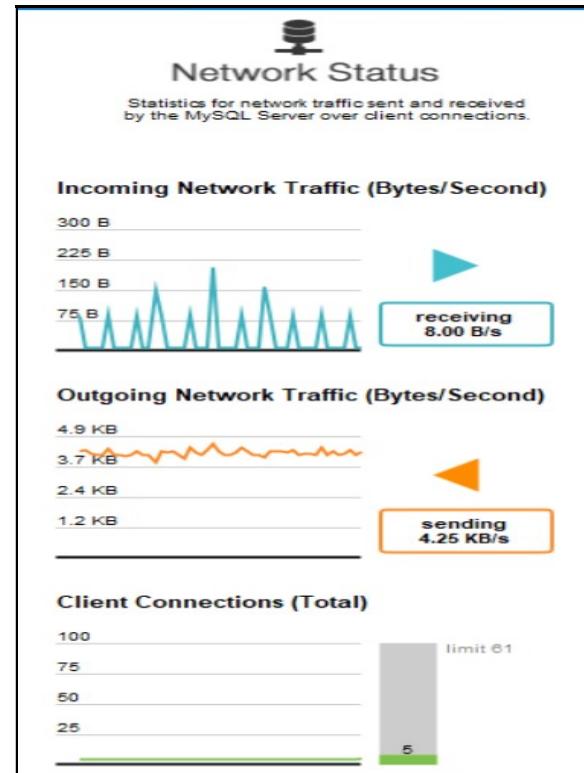


Fig. 32 – MySQL Workbench Performance Dashboard 2

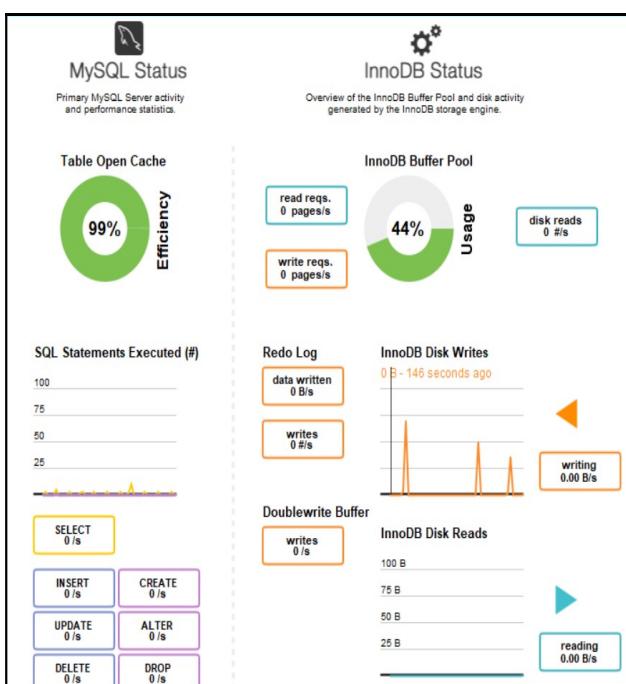


Fig. 31 – MySQL Workbench Performance Dashboard 1