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 22nd Jan to 24th 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.

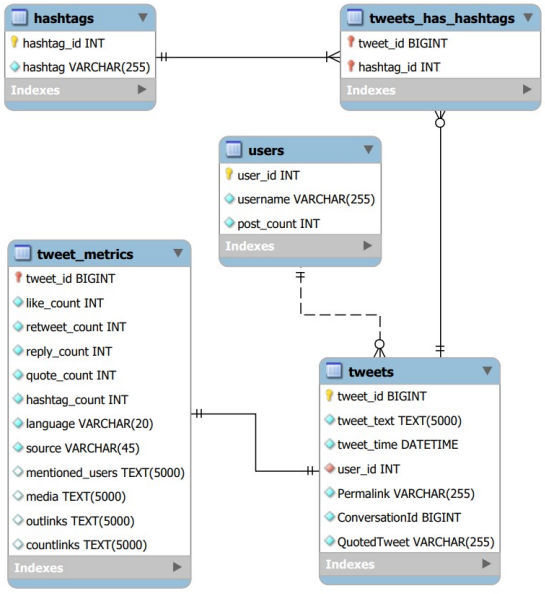


Fig.1 – Entity Relationship Diagram

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 cardinally.

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 and 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.
- Hangtags 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;
```

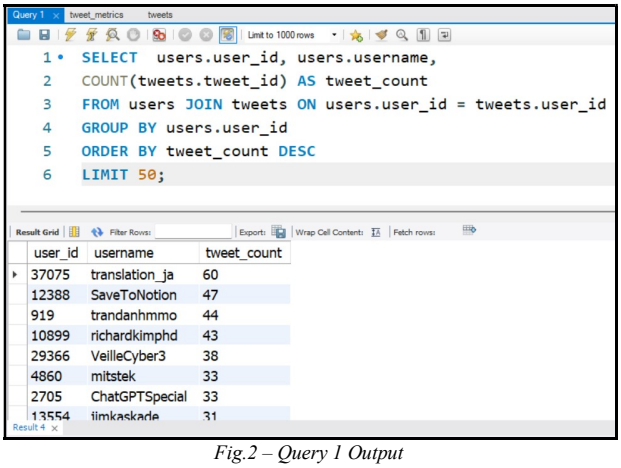


Fig.2 – Query 1 Output

[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;
```

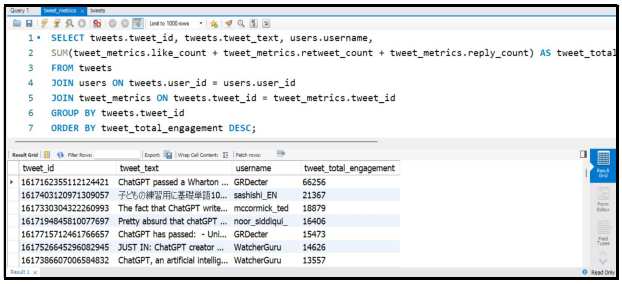


Fig.3 – Query 2 Output

[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;
```

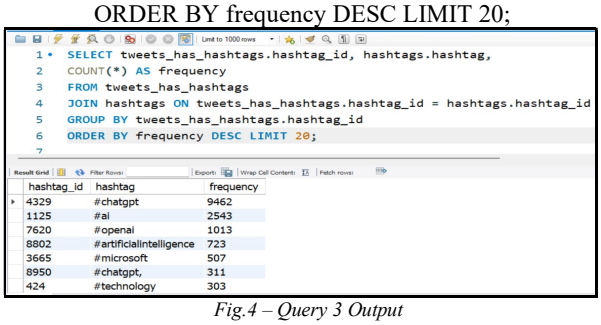


Fig.4 – Query 3 Output

[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;
```

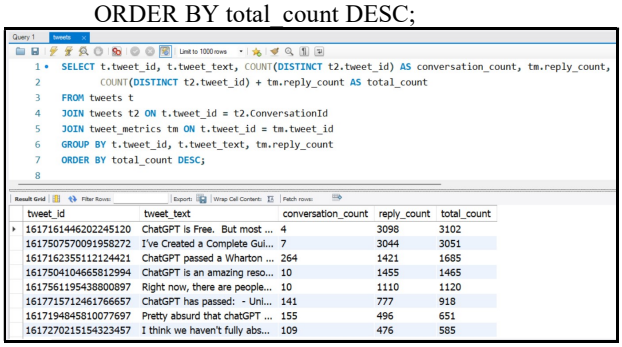
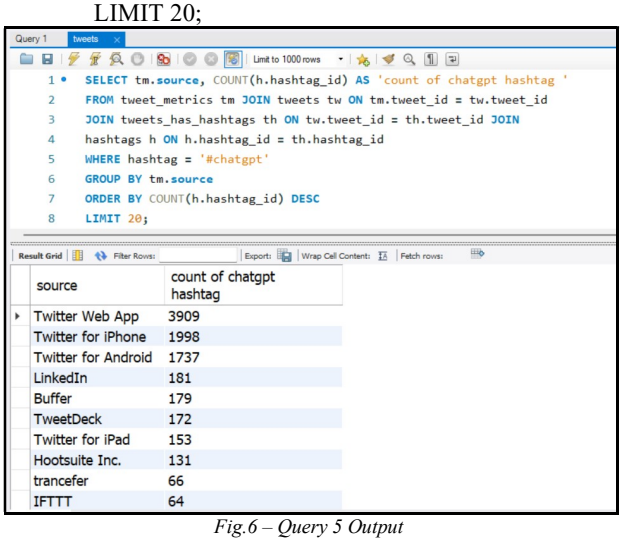


Fig.5 – Query 4 Output

[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 JOIN
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;
```

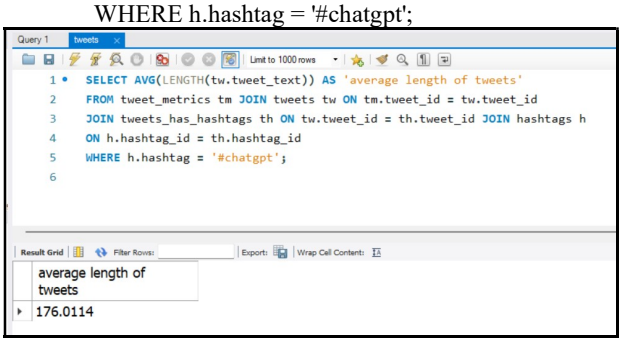


source	count of chatgpt hashtag
Twitter Web App	3909
Twitter for iPhone	1998
Twitter for Android	1737
LinkedIn	181
Buffer	179
TweetDeck	172
Twitter for iPad	153
Hootsuite Inc.	131
trancefer	66
IFTTT	64

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 JOIN hashtags h
ON h.hashtag_id = th.hashtag_id
WHERE h.hashtag = '#chatgpt';
```

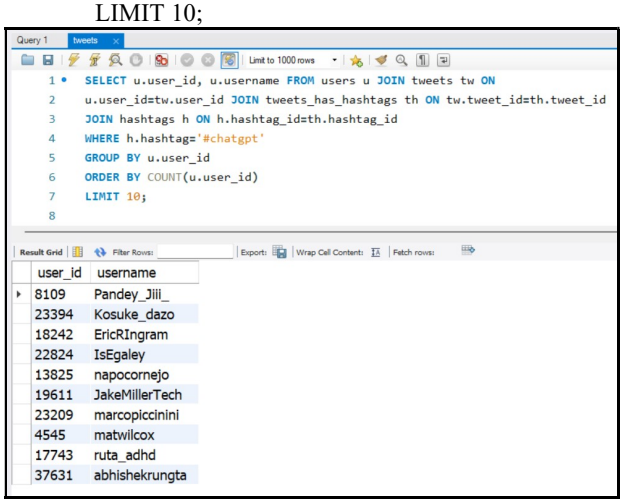


average length of tweets
176.0114

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;
```

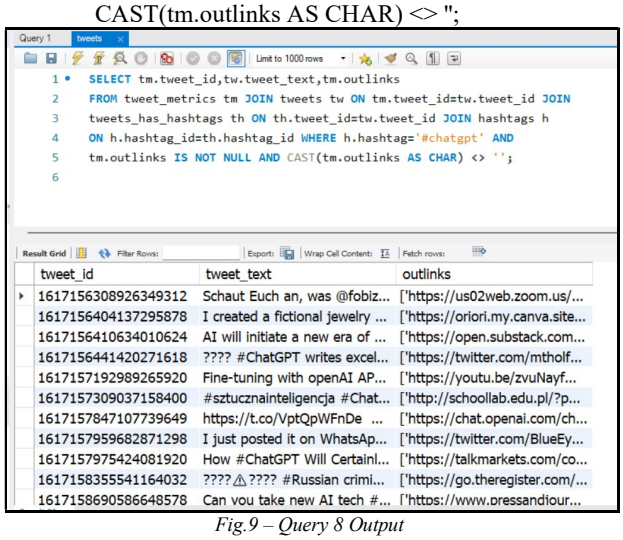


user_id	username
8109	Pandey_Jiii_
23394	Kosuke_dazo
18242	EricRIngram
22824	IsEgaley
13825	napocornejo
19611	JakeMillerTech
23209	marcopiccinini
4545	matwilcox
17743	ruta_adhd
37631	abhishekrungta

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
th.tweet_id=tw.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.substack.com...
1617156441420271618	???? #ChatGPT writes excel...	['https://twitter.com/mtholf...
1617157192989265920	Fine-tuning with openAI AP...	['https://youtu.be/zvuNayf...
1617157309037158400	#sztucznainteligencja #Chat...	['http://schoollab.edu.pl/?p...
1617157847107739649	https://t.co/VptQpWFnDe ...	['https://chat.openai.com/ch...
1617157959682871298	I just posted it on WhatsAp...	['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.pressandjour...

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;
```

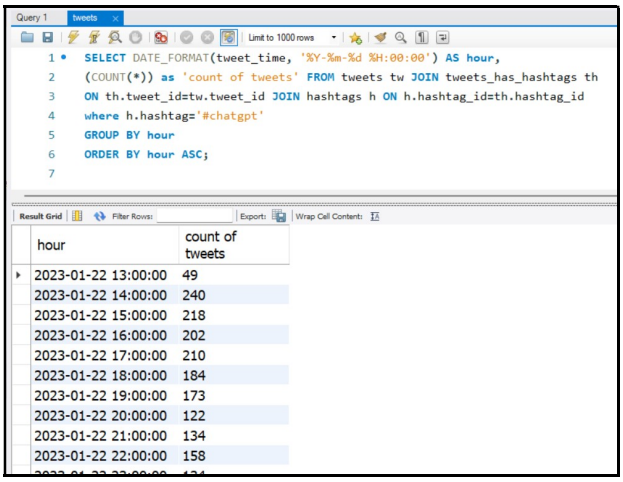


Fig.10 – Query 9 Output

[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
JOIN
hashtags ON tweets_has_hashtags.hashtag_id =
hashtags.hashtag_id
GROUP BY hashtags.hashtag
ORDER BY like_count DESC;
```

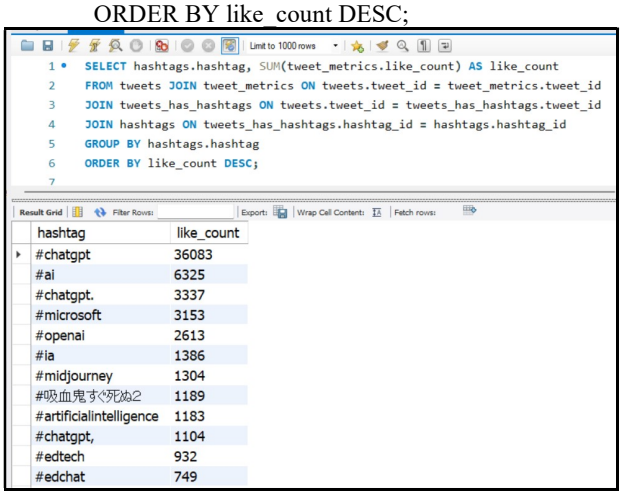


Fig.11 – Query 10 Output

[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;
```

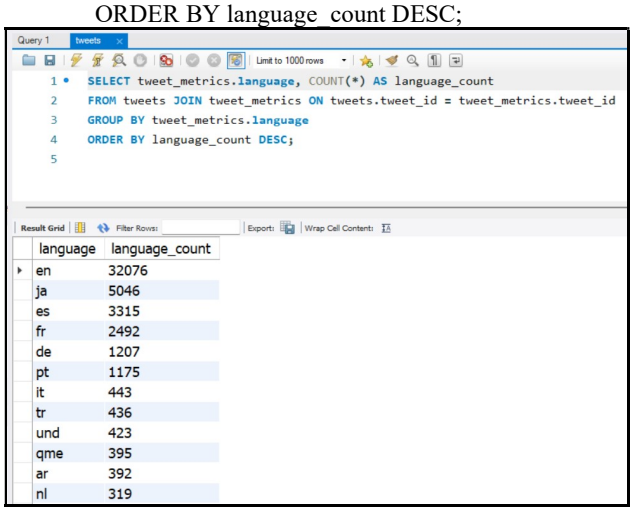


Fig.12 – Query 11 Output

[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;
```

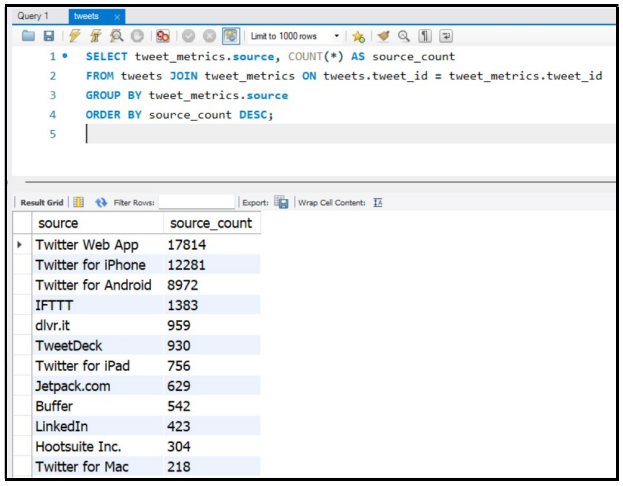


Fig.13 – Query 12 Output

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 ;

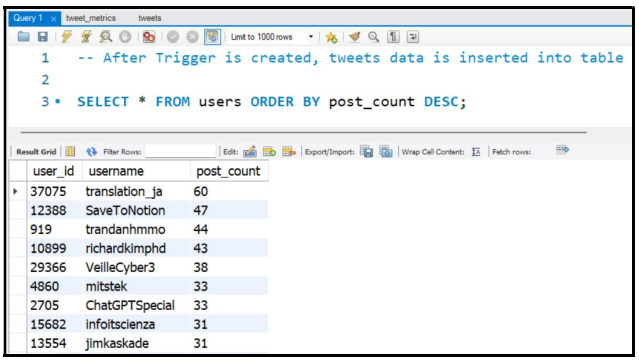


Fig.14 –Trigger 1 Output

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');
```

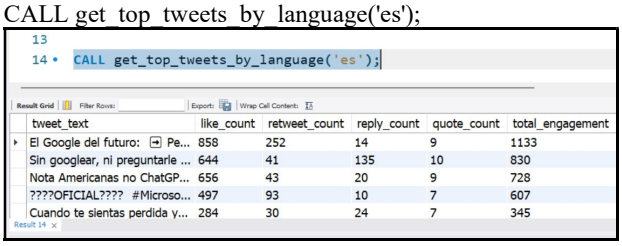


Fig.15 – S.P. 1 Output

[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');
```

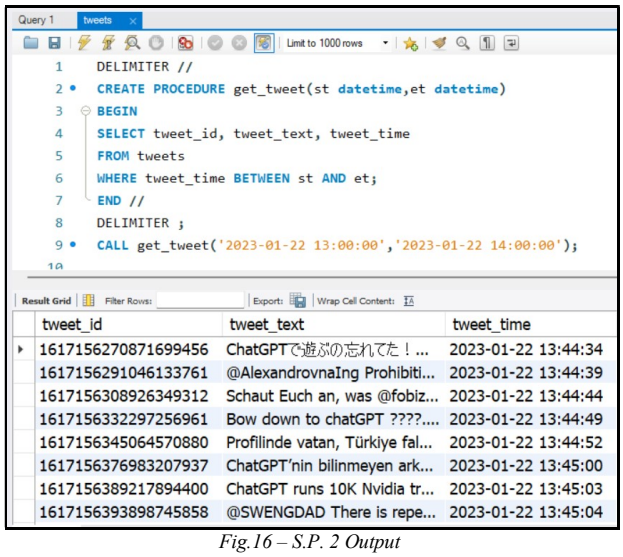


Fig.16 – S.P. 2 Output

[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')
```

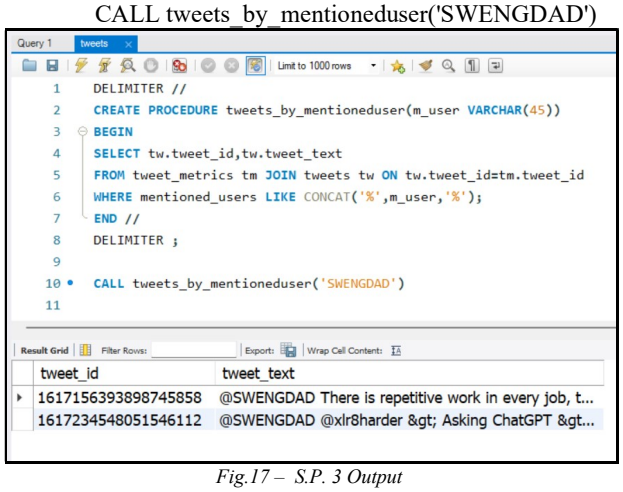


Fig.17 – S.P. 3 Output

[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_has_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');
```

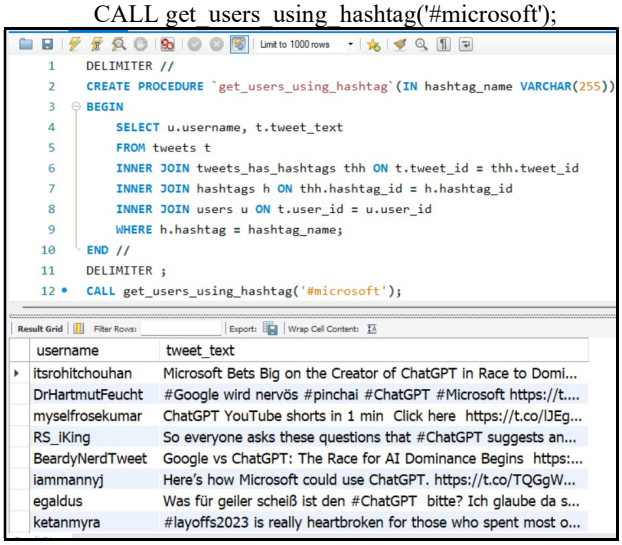


Fig. 18– S.P. 4 Output

[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');
```

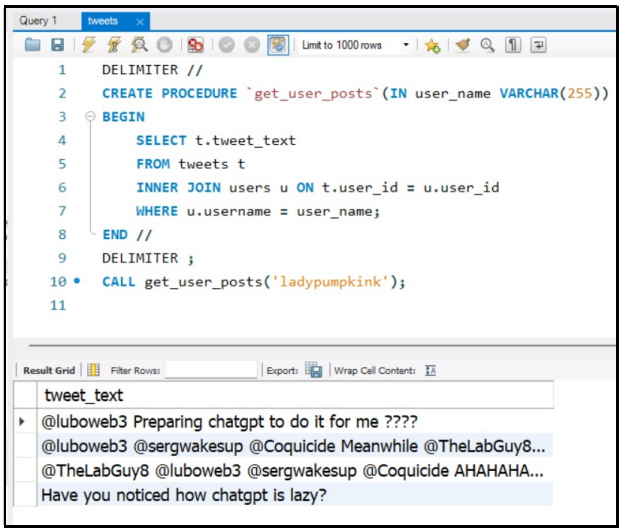


Fig.19 – S.P. 5 Output

[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');
```

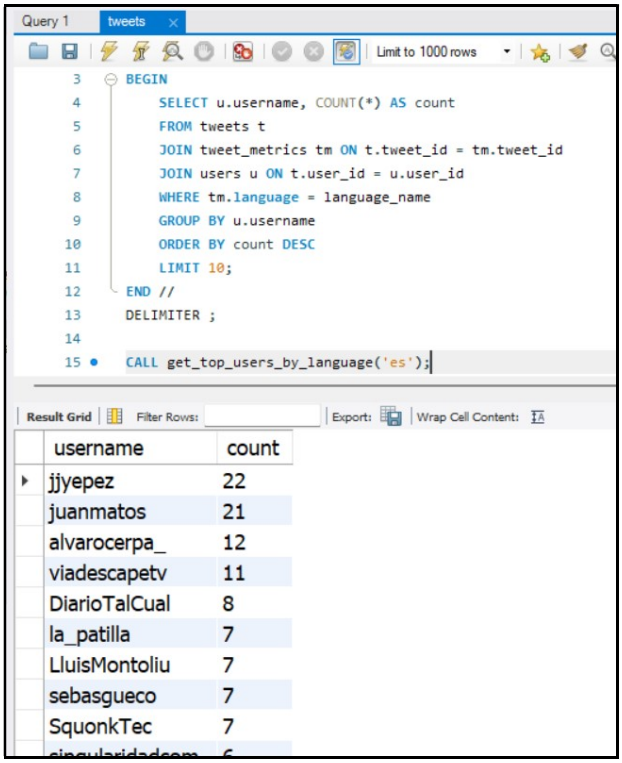


Fig.20 – S.P. 6 Output

[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");
```

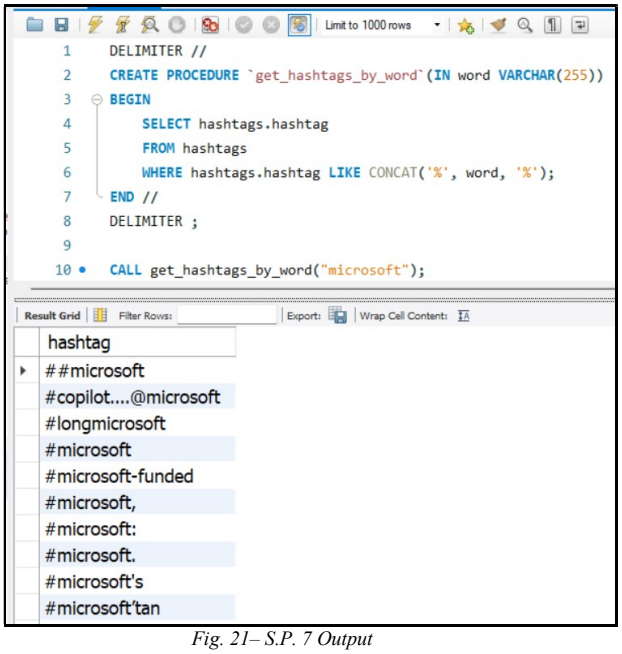


Fig. 21– S.P. 7 Output

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.

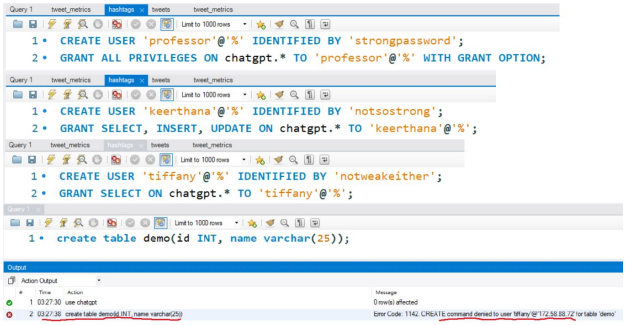


Fig.22 – Figure showing users and access privileges

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.

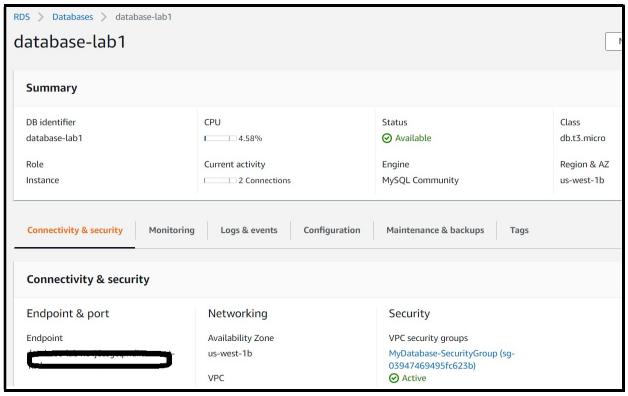


Fig. 23– RDS Instance in AWS

```
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..
[('GRDecter', 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

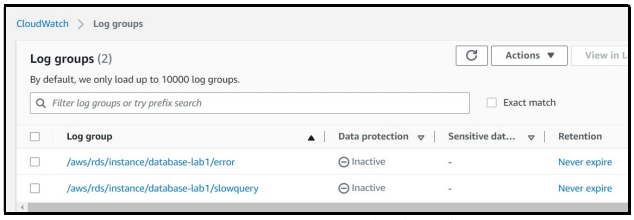


Fig. 26– Log Groups in CloudWatch

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.

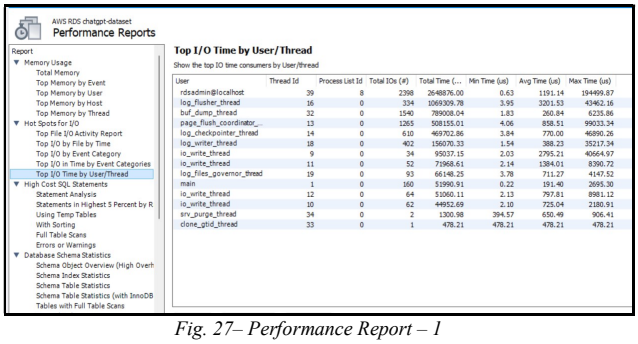


Fig. 27– Performance Report – 1

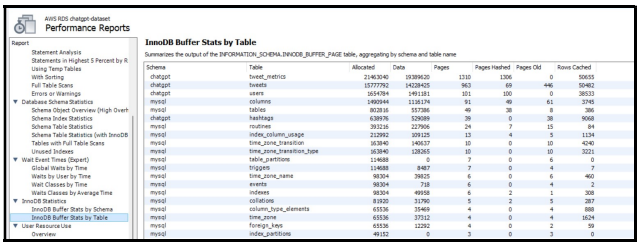


Fig.28 – Performance Report - 2

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.

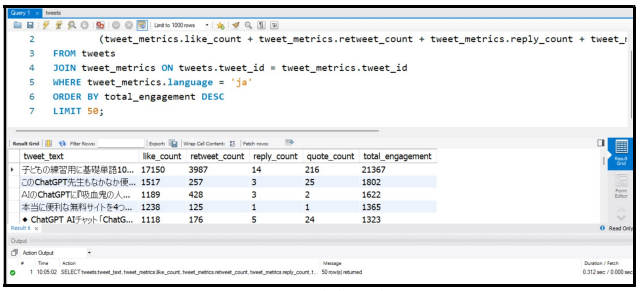


Fig.29 –Query executed directly – 0.312 sec

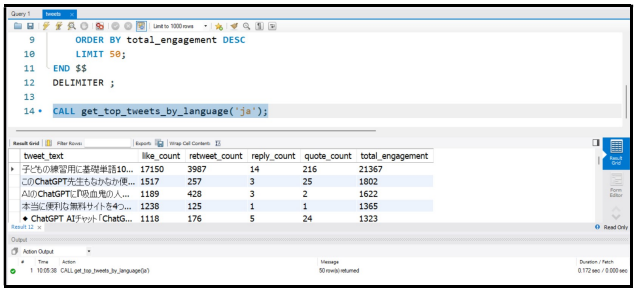


Fig. 30– Query executed as a stored procedure – 0.172 sec

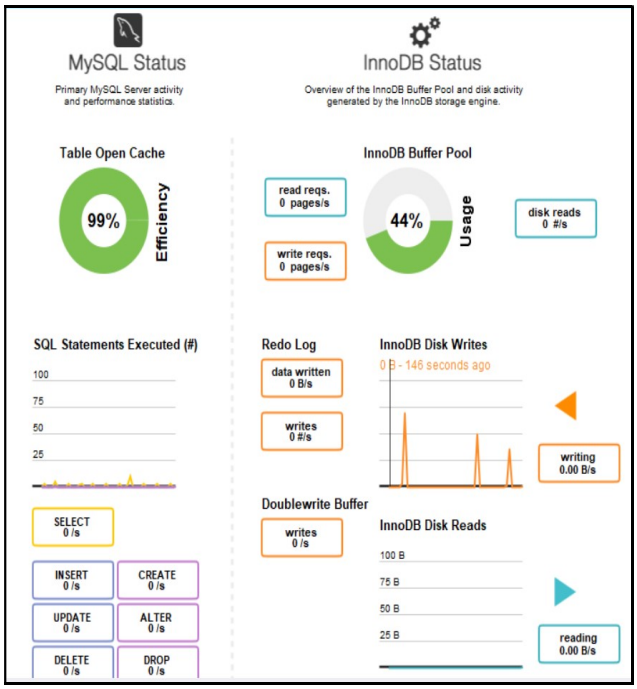


Fig. 31 – MySQL Workbench Performance Dashboard 1

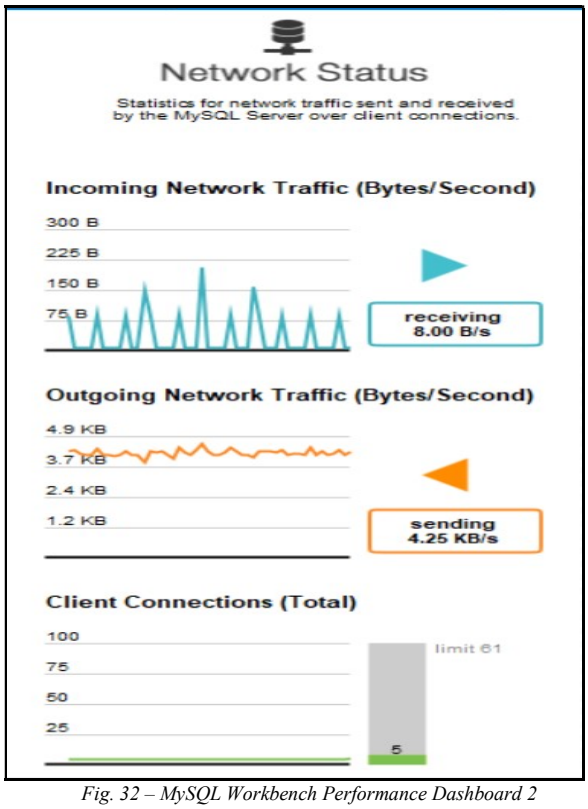


Fig. 32 – MySQL Workbench Performance Dashboard 2