SJSU CHATGPT TWITTER ANALYSIS

Lab Report - 2 (Team - 8)

Shashank Reddy Kandimalla (016799523), Rohith Reddy Vangala (016762109), Sakshi Manish Mukkirwar (016794765), Swati (016702413), Yamini Muthyala (016766165)

ABSTRACT - This lab report describes the application for examining tweets related to ChatGPT, a language model built using the GPT-3.5 architecture. The tweets with the hashtag #chatgpt were collected from Twitter and used as the dataset[1] for this project. Here we have identified several interesting things about the dataset[1] and have generated the required outputs that are listed in Solution requirements (3). The collections are used to establish a conceptual design for the database, and functional analysis was carried out to determine the essential functional elements. The SJSU ChatGPT Tweets Analysis database was moved to Mongodb Atlas because the application system is cloud-native. The application helps users understand the attitudes, viewpoints, and actions of people discussing ChatGPT on Twitter by offering insights into the online discourse surrounding this language model.

KEYWORDS - Mongodb Atlas,Mongodb Compass, NoSQL, Cloud-Native, Data Processing, Data Visualization, Denormalization, Documents, Sharding.

1. INTRODUCTION

The creation of various language models that have transformed the field of natural language processing has been facilitated by the growth of artificial intelligence (AI) (NLP). Based on the GPT-3.5 architecture, ChatGPT is one such language model that has attracted a lot of attention recently. It is necessary to study this online discussion as people discuss ChatGPT and contribute their personal stories, viewpoints, and ideas in order to learn more about the attitudes and trends that surround this language model.

We propose an application that examines tweets concerning ChatGPT that have been scraped from Twitter and contain the hashtag #chatgpt in order to fill this demand. To help users understand the behaviors and patterns of those discussing ChatGPT, the application intends to provide insights into the online conversation surrounding this language model on twitter.

The development of this application is thoroughly described in this report. We start by identifying the functional requirements for the application, use the collections to conceptually build the database. Then, we conduct a functional analysis to pinpoint the essential functional elements needed for the application. Finally, we talk about how the SJSU ChatGPT Tweets Analysis database was moved to Mongodb Atlas Using a cloud-native strategy. Ultimately, the purpose of this study is to give readers a thorough understanding of the design and operation of the ChatgptAnalysisProgram.

2. PROBLEM STATEMENT

With the rise of language models, like ChatGPT, it is crucial to understand the attitudes, viewpoints, and actions of people discussing these models online. Twitter is a popular platform for discussing ChatGPT, and analyzing tweets related to this language model can provide valuable insights. Therefore, the problem statement is to develop an application that examines tweets containing the hashtag #chatgpt to understand the online discourse surrounding ChatGPT. The application should provide insights into the behaviors and patterns of those discussing ChatGPT on Twitter, by identifying the most active individuals, the most popular tweet, frequently used hashtags, and the tweet that received the most attention. Additionally, the application should provide a cloud-native solution for hosting the SJSU ChatGPT Tweets Analysis database.

To store the tweets and their associated metadata, as well as the results of our analysis, we will use a sophisticated database system that can handle large volumes of data and perform complex queries efficiently. The database will be designed to allow for easy updating and modification of the results of our analysis, as new data becomes available or as our research questions evolve.

SJSU CHATGPT TWITTER ANALYSIS

3. SOLUTION REQUIREMENTS

1. The proposed application system (fig-1) will extract Twitter data from the ChatGPT Twitter dataset and perform a comprehensive analysis of the dataset. The analyzed findings will be stored in a NoSQL database, providing insights into various aspects of ChatGPT tweets.
2. The system's capabilities include finding the most active Twitter users talking about ChatGPT and the viral tweet based on retweets and figuring out the most talked-about tweet with the most replies and conversations , as well as the most recent tweets and likes.
3. The application is designed to locate frequently used hashtags in ChatGPT tweets and recognize trends or conversational themes based on hashtag usage.
4. The system can identify the tweet count based on the different devices and can determine the connection between hyperlinks in tweets and the websites or pages that are most frequently shared on Twitter. Moreover ,it can identify the relationship between hyperlinks in tweets and engagement indicators such as retweets and likes.
5. The system can calculate the average engagement per like, retweet, and reply across the entire collection of tweets, based on the engagement indicators available on Twitter.
6. The system can locate tweets with the highest ratio of retweets to likes, as this can indicate particularly viral or shareable content.
7. The application can identify tweets with media (such as images or videos) that have the highest engagement based on reply count, retweet count, and like count. This can help identify the types of media that are most effective at engaging with users on Twitter.
8. Additionally, the system will recognize the most active hour of the day for tweeting. Regarding the individuals , the people with influence or expertise who are frequently referenced in tweets can also be recognized by the system.
9. The system will also determine the commonly used media by popular accounts or topics in their tweets.

3.1.LIMITATIONS

However, The system's limitations should also be taken into account. It does not have sentiment analysis or topic modeling for more extensive insights. The system may not be able to capture Twitter data outside the ChatGPT dataset, limiting its scope.
Also, The system may not always accurately determine the commonly used media by popular accounts or topics in their tweets, as this can be influenced by various factors such as user preferences, platform algorithms, and trending topics. The ChatGPT Twitter dataset may not be a representative sample of the overall population, and it may not reflect the opinions and experiences of all Twitter users. Thus, the results of the analysis may not be generalized to the entire population.The system can currently only analyze the ChatGPT dataset provided in .csv file format, json format, which may limit its ability to analyze other types of data, such as XML files. Additionally, the accuracy of the system's analysis may be affected by factors like Twitter API limitations or data quality issues. Thus, users of the system should be aware of these limitations and use the system's output accordingly. Overall, the system will be an essential tool for researchers and analysts seeking insights into ChatGPT tweets.

4. DATASET AND PRE-PROCESSING

There are 50,000 tweets altogether in the "ChatGPT Twitter Dataset"[1] dataset that is accessible on Kaggle. Tweets related to the ChatGPT language model are included in the dataset. Date time, tweet id, tweet text, Username, Permalink, User, Outlink, Countlinks, Reply Count, Retweet Count, Like Count, Quote Count, Conversation Count, Language, Source, Media, Quoted Tweet, Mentioned Users, hashtag, Hashtag Count are the attributes(fig-2) included in the collection. These characteristics offer insightful data on the users tweeting about ChatGPT and can be applied to various studies.
In NoSQL databases, such as document-oriented databases like MongoDB, Normalization is not as strictly applied in SQL databases as it is in relational databases. Denormalization is often used in NoSQL databases to improve performance and scalability. Multivalued attributes can be stored as arrays or nested documents within a single document in a document-oriented database i.e include countlinks, outlinks, mentioned users, and hashtags (fig-2).
Alternatively, multiple fields or properties can be created in a document to store each value separately. This allows for a more granular approach to querying or indexing individual values, and can accommodate specific performance or storage requirements.

SJSU CHATGPT TWITTER ANALYSIS

5. FLOW CHART

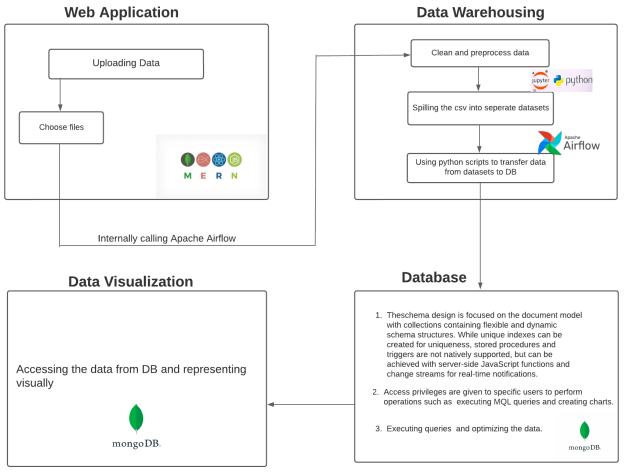

Fig-1 Flow chart of the proposed application

6. CONCEPTUAL DATABASE DESIGN

1. Identifying the collections: We analyzed the dataset and identified the main collections that need to be stored in the database. For the Twitter dataset, collections could include users, tweets, hashtags, mentions, and retweets. After identifying the collections, we determined the relationships between them. For example, a tweet is created by a user, a tweet can contain hashtags, and a tweet can be retweeted by other users.

2. Denormalizing the schema: We denormalized the schema to improve query performance and reduce the number of queries required to retrieve data. This involved combining related collections and embedding documents to eliminate the need for joins.

3. Determining data types: We determined the data types for each field in the collections. For example, the "Datetime" field in the tweets collection is declared with datetime data type.

4. Creating the collections, indexes : Using the denormalized schema and data types, we created the collections in the database. We also created indexes on frequently searched or sorted fields to improve query performance.

5. Populating the collections: We imported the data from the Twitter dataset into the database collections.

6. Testing the database: We tested the database to ensure that it is functioning correctly and that queries return the expected results.

7. Optimizing the database: We optimized the database by analyzing query performance, adjusting indexes, and optimizing queries as needed.

8. Monitoring and maintaining the database: We monitored the database to identify and fix any issues, and performed routine maintenance tasks such as backing up the database.

9. Defining user roles and access levels: We identified the different types of users who will be accessing the system and the data. We granted specific privileges, such as read and write access, based on the role and responsibilities of users.

10. Implementing authentication: We required users to authenticate themselves before accessing the system by implementing a username and password login system.

Overall, we followed these steps to design and implement our MongoDB database for the Twitter dataset, which allowed us to efficiently store and retrieve data, as well as perform complex queries and analysis.

6.1. DENORMALIZATION STEPS

The chatgpt dataset posed a challenge in terms of efficient data storage and retrieval. The initial storage mechanism employed was normalization of the data, which distributed the data into multiple collections. Although this approach has its benefits, such as reducing data redundancy, it can result in a slower database performance due to the need for JOIN operations between collections.

To address this performance issue, we opted to denormalize the data and consolidate it into a single collection, namely tweets, in our MongoDB database. This denormalization approach involved embedding arrays of documents from the other collections under a single document in the tweets collection, thereby creating a single document that contains all the relevant data for each entity. This approach eliminated the need for costly JOIN operations and allowed for faster and more efficient queries for related data.

The denormalization approach not only improved the performance of our database but also simplified the data retrieval process, as the need for complex queries and JOIN operations was eliminated. This resulted in a faster response time and a more streamlined user experience. Additionally, this approach can also help reduce the need for complex database schema designs, making it easier to maintain and scale the database in the future.

In summary, our denormalization approach was a successful strategy for improving the performance of the

SJSU CHATGPT TWITTER ANALYSIS

SJSU chat dataset. By consolidating the data into a single collection and embedding arrays of documents, we were able to simplify the data retrieval process, reduce the need for complex queries, and ultimately improve the user experience.

7. FUNCTIONAL ANALYSIS

1. Data Ingestion and storage: This component ingests data from a CSV file into MongoDB and stores it in a format that is suitable for analysis.

2. Data querying and filtering: This component allows users to query and filter the data based on various parameters, such as username, retweet count, and hashtag. It can do this by using a user interface for query input and filtering the data with the MongoDB query language. MongoDB's extensive query language supports complex filtering and aggregation operations, which allows us to handle large datasets more quickly and with less overhead than MySQL.

3. Data Aggregation and Analysis: This component analyzes the data that is gathered from the NoSQL database. It does this by using the aggregation pipeline in MongoDB to organize, sort, and summarize the data according to different criteria. It will offer the option to perform data analysis based on user activity, tweet virality, hashtag popularity, and debate trend.

a. User Analysis: This part will be in charge of figuring out which users are talking the most about ChatGPT on Twitter.

b. Viral Tweet Analysis: This section will be in charge of locating the ChatGPT-related tweet that has received the most retweets.

c. Hashtag Analysis: This section will be in charge of figuring out which hashtags were used frequently in tweets regarding ChatGPT.

d. Conversation Analysis: This section will be in charge of determining the tweet with the most ChatGPT- related conversations and replies.

5. Data Visualization Component: This component is in charge of displaying the studied data as graphs, tables, and charts. Various criteria, including user activity, tweet virality, hashtag popularity, and conversation trend, can be used to draw the insights and to illustrate the data in a user-friendly format.

8. MONGODB SCHEMA

ChatGPT dataset

Attribute	Datatype	Description
tweet_id	Long	Unique identifier for each tweet
text	String	The actual text content of the tweet
username	String	The username of the person who posted the tweet
permalink	String	The permanent link to the tweet on Twitter
user	String	Information about the user who posted the tweet
outlink	String	Any external links included in the tweet
countlinks	String	The number of external links included in the tweet
reply_count	Integer	The number of replies the tweet has received
retweet_count	Integer	The number of retweets the tweet has received
like_count	Integer	The number of likes the tweet has received
quote_count	Integer	The number of times the tweet has been quoted
conversation_count	Integer	The number of times the tweet has been replied to or quoted
language	String	The language the tweet is written in
source	String	The application or platform used to post the tweet
media	String	Any images, videos, or other media included in the tweet
quoted_tweet	String	If the tweet is a quote tweet, information about the original tweet
mentioned_users	String	Any Twitter users mentioned in the tweet
hashtags	String	Any hashtags included in the tweet
hashtag_count	Integer	The number of hashtags included in the tweet

Fig-2 Table that shows Attributes, Data Types of data.

```
{
 "_id": "object",
 "properties": {
  "date_time": {"type": "string"},
  "tweet_id": {"type": "integer"},
  "tweet_text": {"type": "string"},
  "username": {"type": "string"},
  "permalink": {"type": "string"},
  "user": {"type": "string"},
  "outlink": {"type": "string"},
  "countlinks": {"type": "integer"},
  "reply_count": {"type": "integer"},
  "retweet_count": {"type": "integer"},
  "like_count": {"type": "integer"},
  "quote_count": {"type": "integer"},
  "conversation_count": {"type": "integer"},
  "language": {"type": "string"},
  "source": {"type": "string"},
  "media": {"type": "string"},
  "quoted_tweet": {"type": "string"},
  "mentioned_users": {"type": "string"},
  "hashtag": {"type": "string"},
  "hashtag_count": {"type": "integer"}
 }
}
```

9. DOCUMENT STRUCTURE

One example :

```
{
 "_id": {
  "$oid": "6445e84a38d9db92fc895ea4"
 },
 "Datetime": {
```

"$date": "2023-01-22T13:48:13.000Z"
},
"Tweet Id": {
"$numberLong": "1617157186878115842"
},
"Text": "Just listened to @ChatGPTReport and it was amazing! Great guests and discussions on the ethical considerations of GPT. Highly recommend! #gptreport #AI note: ChatGPT wrote this tweet but, I concur with everything it says. https://t.co/cENncvheSQ",
"Username": "timetravelr2025",
"Permalink": "https://twitter.com/timetravelr2025/status/1617157186878115842",
"User": "https://twitter.com/timetravelr2025",
"Outlinks": "['https://podcasts.apple.com/us/podcast/the-chatgpt-report/id1660264772']",
"CountLinks": "['https://t.co/cENncvheSQ']",
"ReplyCount": 0,
"RetweetCount": 0,
"LikeCount": 0,
"QuoteCount": 0,
"ConversationId": {
"$numberLong": "1617157186878115842"
},
"Language": "en",
"Source": "Twitter for iPhone",
"MentionedUsers": "[User(username='ChatGPTReport', id=1601256235369734155, displayname='The ChatGPT Report', description=None, rawDescription=None, descriptionUrls=None, verified=None, created=None, followersCount=None, friendsCount=None, statusesCount=None, favouritesCount=None, listedCount=None, mediaCount=None, location=None, protected=None, linkUrl=None, linkTcourl=None, profileImageUrl=None, profileBannerUrl=None, label=None)]",
"hashtag": "['#gptreport', '#AI']"
}

10. MQL QUERIES

1. Users most discussing about it on twitter(tweets):
Query:

```
db.Tweet.aggregate([ {$match: {Text: /ChatGPT/i}},
{$group: {_id: {$substr: ["$User", 20, -1]},
 count: {$sum: 1}}},
 {$sort: {count: -1}},
{$limit: 3}]).pretty()
```

Interaction with database: The query uses the MongoDB aggregation pipeline to search for tweets in a collection that contain the string "ChatGPT" in their Text field. It then groups the matching tweets by user_name and counts the number of tweets posted by each user, sorts the results in descending order based on the total number of tweets, and returns only the top 3 users with the highest number of tweets related to "ChatGPT". Finally, the results are formatted in a human-readable format using the pretty function.

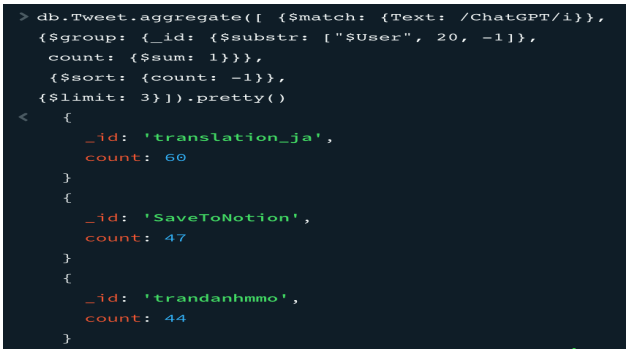

Fig-3 Output for Users most discussing about it on twitter(tweets)

2. Most viral tweet (wrt retweets) about the chatgpt(tweet_counts,tweets):
Query:

```
db.Tweet.find({Text: /ChatGPT/i}).sort({"RetweetCount": -1}).limit(1).pretty()
```

Interaction with database: The query searches for tweets in a MongoDB collection that contain the string "ChatGPT" in their Text field, sorts them in descending order based on their RetweetCount, returns the top result, and formats it in a human-readable format using the pretty function.

Fig-4 Output for Most viral tweet about the chatgpt

SJSU CHATGPT TWITTER ANALYSIS

3. Hashtags used prominently for the tweets about chatgpt(tweet_hashtags, tweet_hashtag_mappings):

Query:

```
db.Tweet.aggregate([
  {$match: {Text: /ChatGPT/i}},
    {$project: {hashtags: {$regexFindAll: {input: "$Text", regex: /#\S+/}}}},
  {$unwind: "$hashtags"},
  {$project: {hashtag: {$toLower: "$hashtags.match"}}},
  {$match: {hashtag: {$ne: ""}}},
  {$group: {_id: "$hashtag", count: {$sum: 1}}},
  {$sort: {count: -1}},
  {$limit: 3}
])
```

Interaction with database: The query searches for tweets in a MongoDB collection that contain the string "ChatGPT" in their Text field, extracts and counts the hashtags used in those tweets, removes any empty hashtags, and sorts the remaining hashtags in descending order based on the count of their occurrences.

```
> db.Tweet.aggregate([
    {$match: {Text: /ChatGPT/i}},
    {$project: {hashtags: {$regexFindAll: {input: "$Text", regex: /#\S+/}}}},
    {$unwind: "$hashtags"},
    {$project: {hashtag: {$toLower: "$hashtags.match"}}},
    {$match: {hashtag: {$ne: ""}}},
    {$group: {_id: "$hashtag", count: {$sum: 1}}},
    {$sort: {count: -1}},
    {$limit: 3}
  ])
<   {
      _id: '#chatgpt',
      count: 9390
    }
    {
      _id: '#ai',
      count: 2519
    }
    {
      _id: '#openai',
      count: 1008
    }
```

Fig-5 Output for hashtags used prominently for the tweets about chatgpt.

4. Most discussed tweet (wrt replies and conversation) (tweet_counts, tweets):

Query:

```
db.Tweet.aggregate([{ $match: { Text: /ChatGPT/i } },
 { $group: {_id: "$ConversationId", conversation_count: { $sum: 1 },
reply_count: { $sum: "$ReplyCount" },
 username: { $first: "$Username" }} },
 { $sort: { conversation_count: -1 } },
{ $limit: 1 }],
{ allowDiskUse: true })
```

Interaction with database: The query uses the MongoDB aggregation pipeline to search for tweets in a collection that contain the string "ChatGPT" in their Text field. It then groups the matching tweets by ConversationId, counts the number of conversations and replies, and selects the Username of the first tweet in each conversation. The results are sorted in descending order based on the conversation count and limited to the top result. Finally, the allowDiskUse option is used to allow MongoDB to use disk space to store temporary data if necessary.

```
> db.Tweet.aggregate([{ $match: { Text: /ChatGPT/i } },
   { $group: {_id: "$ConversationId", conversation_count: { $sum: 1 },
  reply_count: { $sum: "$ReplyCount" },
   username: { $first: "$Username" }} },
   { $sort: { conversation_count: -1 } },
  { $limit: 1 }],
  { allowDiskUse: true })
<   {
      _id: 1617162355112124400,
      conversation_count: 247,
      reply_count: 1662,
      username: 'GRDecter'
    }
```

Fig-6 Output for Most discussed tweet.

5. Latest tweet(tweet) :

Query:

```
db.Tweet.find({},
 { _id: 1,"Tweet Id":1,
 Text: 1, Datetime: 1 }).sort({ Datetime: -1 }).limit(1)
```

Interaction with database: The query searches for tweets in a MongoDB collection and retrieves the _id, Tweet Id, Text, and Datetime fields of each tweet. The results are sorted in descending order based on the Datetime field, and the top result is returned.

```
> db.Tweet.find({},
   { _id: 1,"Tweet Id":1,
   Text: 1, Datetime: 1 }).sort({ Datetime: -1 }).limit(1)
<   {
      _id: ObjectId("6445d86638d9db92fc892b9d"),
      Datetime: '2023-01-24 06:58:01+00:00',
      'Tweet Id': 1617778731678044200,
      Text: 'Portland Shop Uses ChatGPT To Tell Family Stories On A Startup Budget https://t.co/rz6vr6yTOc'
    }
```

Fig-7 Output for Latest tweet.

6. Count of tweets from different devices(tweets):

Query:

```
db.Tweet.aggregate([ { $group: { _id: { source: { $regexFindAll: { input: "$Source", regex: '(?<=\\>).*(?=<\\/a\\>)' } } },
count: { $sum: 1 } } },
 { $project: { _id: 0, source: "$_id.source", count: 1 } },
 { $sort: { count: -1 } ,{$limit : 3} ])
```

Interaction with database: The query uses the MongoDB aggregation pipeline to group tweets in a collection by the value of the Source field. It extracts the source name using a regular expression and counts the number of tweets associated with each source. The results are projected to

show only the source and count fields and sorted in descending order based on the count.

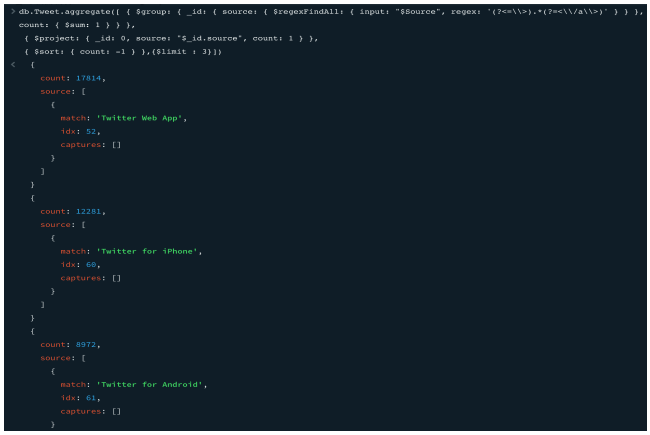

Fig-8 Output for count of tweets from different devices.

7. Analyzing which websites or pages are most commonly shared on Twitter(outlinks):

Query:

db.Tweet.aggregate([{$unwind: "$Outlinks"},
{$project: {outlink: {$trim: {input: "$Outlinks", chars: "[]'"}}}},
{$group: {_id: "$outlink", count: {$sum: 1}}},
{$sort: {count: -1}},{$limit : 3}])

Interaction with database: The query extracts outlinks from tweets in a MongoDB collection and counts the number of occurrences of each unique outlink. It uses the $unwind operator to create a separate document for each outlink, $project to trim the outlink, and $group to group by the outlink and count its occurrences. The results are sorted in descending order based on the count.

Fig-9 Output for websites or pages are most commonly shared on Twitter.

8. Websites which are not secure in outlinks:

Query:

db.Tweet.aggregate([{ $match: { Outlinks: { $regex: /^(?!https:\/\/).+/ } } }, { $group: { _id: "$Outlinks", count: { $sum: 1 } } }, { $sort: { count: -1 } }, { $project: { _id: 0, "Tweet Id":1, Outlinks: "$_id", count: 1 } }, { $limit: 3 }])

Interaction with database: The query uses the MongoDB aggregation pipeline to extract outlinks from tweets in a collection that do not start with "https://". It then counts the number of occurrences of each unique outlink, sorts the results in descending order based on the count, and limits the output to the top 3 results. The $project operator is used to shape the output to include only the tweet ID, outlink, and count of occurrences.

Fig-10 Output for Websites which are not secure in outlinks.

9. tweets by language:

Query:

db.Tweet.aggregate([{$group: {_id: "$Language", count: { $sum: 1 }}}, {$sort: { count: -1} },{$limit : 3}])

Interaction with database: The query uses the MongoDB aggregation pipeline to group tweets by language and count the number of tweets in each language. It then sorts the results in descending order based on the count. This query can be used to identify the number of tweets in each language present in the collection.

Fig-11 Output for tweets by language.

SJSU CHATGPT TWITTER ANALYSIS

10. Time where there are more tweets:

Query:

```
db.Tweet.aggregate([ {
 $addFields: { parsedDate: { $toDate: "$Datetime" }} },
{$group: {_id: { $hour: "$parsedDate" },count: {$sum: 1}
}}, {$sort: { count: -1 }}, {$limit : 3}])
```

Interaction with database: This query adds a new field to the documents in the Tweet collection that represents the parsed date from the Datetime field. It then groups the tweets by the hour of the parsed date and counts the number of tweets in each group. Finally, the results are sorted in descending order based on the tweet count.

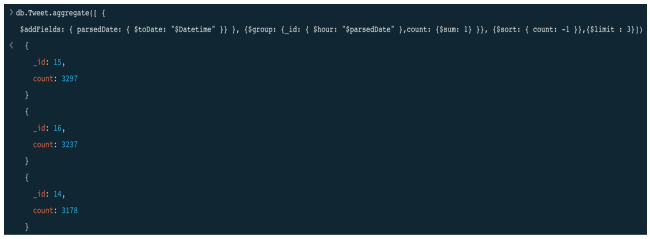

Fig-12 Output showing Time where there are more tweets.

11. Average tweets at particular time:

Query:

```
db.Tweet.aggregate([ {$addFields: { parsedDate: {$toDate:
"$Datetime" }} },{ $group: {_id: { $hour: "$parsedDate"
},avg_likes: { $avg: "$LikeCount" } }}, {$sort: {_id: 1 }}
,{$limit : 3}])
```

Interaction with database: The given MongoDB query calculates the average number of likes for tweets posted at each hour of the day. It first converts the Datetime field to a date type. It then groups the tweets based on the hour of the day and calculates the average number of likes for each group. Finally, it sorts the results by the hour of the day in ascending order.

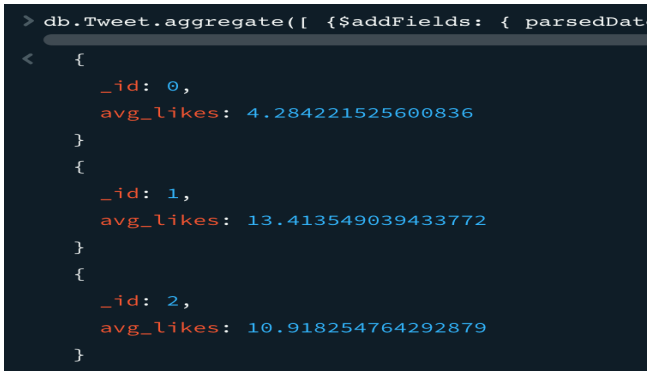

Fig-13 Output for Average tweets at particular time.

12. Top 3 media based on like count :

Query:

```
db.Tweet.find({ "Media": { $exists: true } }, {"_id": 0,
"TweetId": 1, "Text": 1, "LikeCount": 1, "Media": 1}
).sort({ "LikeCount": -1 }).limit(3)
```

Interaction with database: This MongoDB query retrieves the top 3 media objects (tweets that contain media) based on their LikeCount in descending order. The query first filters for tweets that have the "Media" field and selects the relevant fields (TweetId, Text, LikeCount, and Media) for those tweets. The results are sorted in descending order of LikeCount and limited to 3.

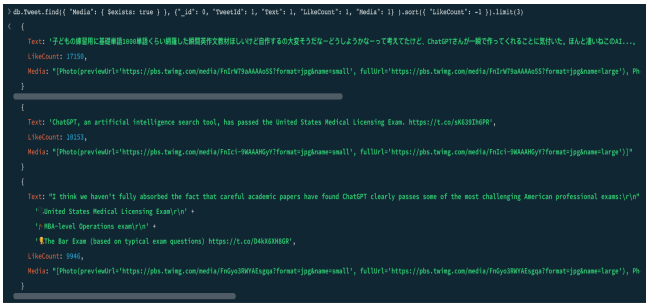

Fig-14 Output for top 3 media based on like count.

13. Finding the top 3 users with the highest engagement (total sum of reply count, retweet count, like count, and quote count):

Query:

```
db.Tweet.aggregate([ {$group: {
 _id: "$Username", totalEngagement: {$sum: { $add: [
"$ReplyCount", "$RetweetCount", "$LikeCount",
"$QuoteCount" ]} }} },{$sort: {"totalEngagement": -1 }},
{$limit: 3 }])
```

Interaction with database: This MongoDB query aggregates tweets by the Username field and calculates the total engagement for each user by summing up the ReplyCount, RetweetCount, LikeCount, and QuoteCount. The resulting documents are then sorted by the totalEngagement field in descending order and only the top 3 are returned.

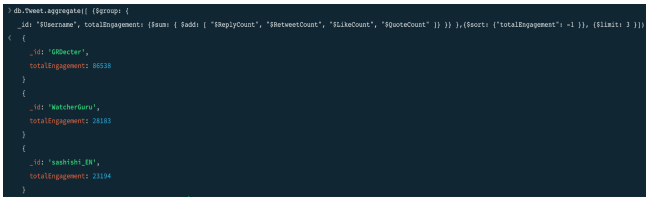

Fig-15 Output for top 3 users with the highest engagement.

SJSU CHATGPT TWITTER ANALYSIS

14. Most mentioned users :

Query:

```
db.Tweet.aggregate([{ $unwind: "$MentionedUsers"},
{$group: { _id: "$MentionedUsers", count: { $sum: 1 }}
},{$sort: {count: -1 }}, {$limit: 3 }])
```

Interaction with database: This MongoDB query starts by unwinding the MentionedUsers array, then grouping the tweets by the mentioned user and counting the number of occurrences. The result is then sorted in descending order by the count and limited to the top 3 mentioned users.

Fig-16 Output for Most mentioned users.

15. Most occurred word in the text field:

Query:

```
db.Tweet.aggregate([{"$match": {"Text": {"$regex":
"^(?!.*ChatGPT).*", "$options": "i"}}}, {"$project":
{"words": {"$split": ["$Text", " "]}}},
 {"$unwind": "$words"},
 {"$group": {"_id": "$words", "count": {"$sum":
1}}},{"$sort": {"count": -1}},{"$limit": 3} ])
```

Interaction with database: This query matches tweets that don't contain the phrase "ChatGPT" in the text field, splits the text into words, unwinds the resulting array, groups by each unique word and counts their occurrences, sorts the result in descending order of count, and returns the top 10 most occurred words.

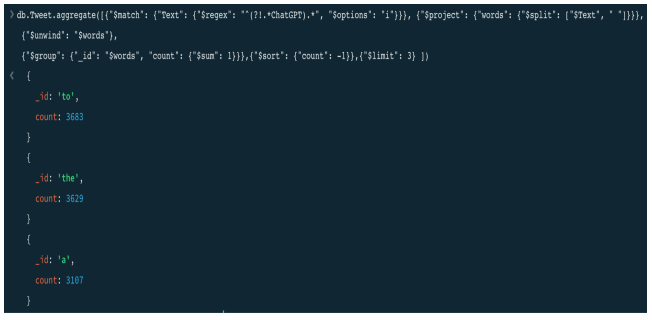

Fig-17 Output for Most occurred word in the text field.

16. Calculate the average engagement per like, retweet, and reply across the entire collection:

Query:

```
db.Tweet.aggregate([ {$match: { $or: [{ LikeCount: { $gt:
0 } }, { RetweetCount: { $gt: 0 } },
{ ReplyCount: { $gt: 0 } }]}},{$group: { _id:
null,total_likes: { $sum: "$LikeCount" },
total_retweets:{$sum:"$RetweetCount"},
total_replies: { $sum:
"$ReplyCount"},total_engagement:{$sum:{$add:
["$LikeCount", "$RetweetCount", "$ReplyCount"]}}}},
{$project: {_id: 0,avg_engagement_per_like: { $divide:
["$total_engagement", "$total_likes"]
},avg_engagement_per_retweet: { $divide:
["$total_engagement", "$total_retweets"]
},avg_engagement_per_reply: { $divide:
["$total_engagement", "$total_replies"] } }} ])
```

Interaction with database: This MongoDB query calculates the average engagement per like, retweet, and reply across the entire collection of tweets. It first matches tweets with at least one engagement (like, retweet, or reply), then groups all tweets to calculate the total number of likes, retweets, replies, and total engagement. Finally, it projects the average engagement per like, retweet, and reply by dividing the total engagement by the total number of likes, retweets, and replies, respectively. The result does not have an "_id" field and contains only the average engagement per like, retweet, and reply.

Fig-18 Output for average engagement per like, retweet, and reply across the entire collection.

17. Finding the tweets with the highest ratio of retweets to likes:

Query:

```
db.Tweet.aggregate([{$match: {RetweetCount: {$gt: 0},
LikeCount: {$gt: 0}}},
{$project: {_id: 0, “Tweet Id”: 1, RetweetToLikeRatio:
{$divide: ["$RetweetCount", "$LikeCount"]}}},
{$sort: {RetweetToLikeRatio: -1}},{$limit: 3} ])
```

SJSU CHATGPT TWITTER ANALYSIS

Interaction with database: This MongoDB query finds the tweets with the highest ratio of retweets to likes in the collection of tweets. It first filters for tweets with non-zero RetweetCount and LikeCount, then calculates the RetweetToLikeRatio by dividing the RetweetCount by the LikeCount for each tweet. The results are sorted in descending order by RetweetToLikeRatio and limited to the top 10 tweets. The output includes the Tweet Id and RetweetToLikeRatio fields for each of the 10 tweets.

```
> db.Tweet.aggregate([
    {$match: {RetweetCount: {$gt: 0}, LikeCount: {$gt: 0}}},
    {$project: {_id: 0, "Tweet Id": "$_id", RetweetToLikeRatio: {$divide: ["$RetweetCount", "$LikeCount"]}}},
    {$sort: {RetweetToLikeRatio: -1}},
    {$limit: 3}
  ])
<   {
      'Tweet Id': ObjectId("6445d83b38d9db92fc8877d6"),
      RetweetToLikeRatio: 21
    }
    {
      'Tweet Id': ObjectId("6445d84e38d9db92fc88c958"),
      RetweetToLikeRatio: 7
    }
    {
      'Tweet Id': ObjectId("6445d83f38d9db92fc88846a"),
      RetweetToLikeRatio: 5
    }
```

Fig-19 Output for tweets with the highest ratio of retweets to likes.

18. Updating the Languages name to their Full Name:
Query:

```
db.Tweet.updateMany(
  { Language: "en" },
  { $set: { Language: "English" } }
);
```

Interaction with database: This MongoDB query updates all documents in the "Tweet" collection where the "Language" field equals "en". It sets the "Language" field to "English".

```
> db.Tweet.updateMany(
    { Language: "en" },
    { $set: { Language: "English" } }
  );
<   {
      acknowledged: true,
      insertedId: null,
      matchedCount: 32076,
      modifiedCount: 32076,
      upsertedCount: 0
    }
```

Fig-20 Output for changing the Language name to full name.

19. Deleting a particular Tweet :
Query:

```
db.Tweet.deleteOne({ "Tweet Id": 1617156308926349312 })
```

Interaction with database: This MongoDB query deletes a single document from the "Tweet" collection where the value of the "Tweet Id" field is equal to 1617156308926349312.

```
> db.Tweet.deleteOne({ "Tweet Id": 1617156308926349312 })
<   {
      acknowledged: true,
      deletedCount: 1
    }
```

Fig-21 Output showing the deleted tweet.

11. VISUALIZATION OF MONGODB DATA

1. Top 10 Mentioned Users:
Below visualization shows us the top 10 users who were mentioned by other users. The bars represent the count of mentioned users. This visualization can help users to identify the most active and engaging users.

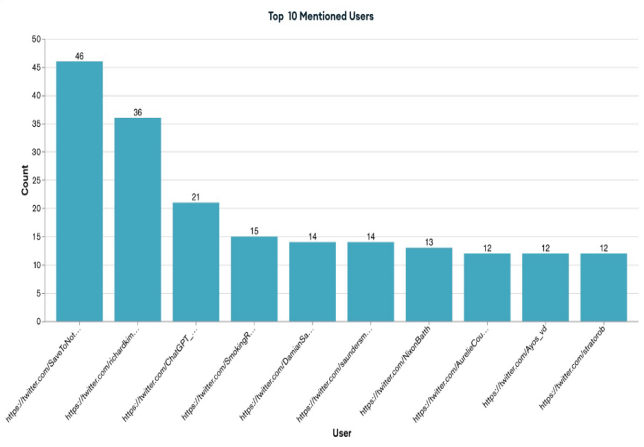

Fig-22 Top 10 Mentioned Users

2. Word Cloud of most used Hashtag:
This is a word cloud that shows us the top 100 trending '#' (hashtags) that have occurred in the dataset. A word cloud is a graphical representation of text data, where words are displayed in varying sizes based on their frequency or importance. In this case, the hashtags are the words that have been extracted from the dataset and arranged in the form of a word cloud.

SJSU CHATGPT TWITTER ANALYSIS

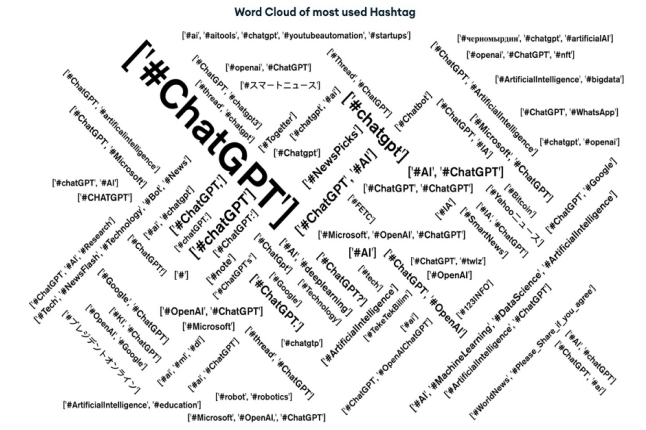

Fig-23 Word Cloud of most used hashtag

3. Language Proportions:

The below donut chart shows us the number of tweets in every language. Every segment of the chart shows us the percent of total that language was used. By analyzing the language distribution, users can gain insights into the geographic and demographic diversity of their audience.

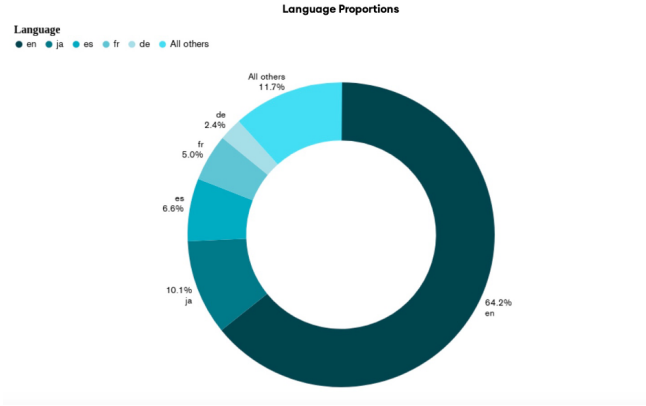

Fig-24 Language Proportions

4. Top 10 Most Used Hashtag:

The below visualization shows the top trending hashtags. The data has been filtered to exclude the hashtag #ChatGPT, which means that the remaining hashtags are the most popular and frequently used ones among the users. This visualization can be useful as it helps to understand the current trends and conversations among the users.

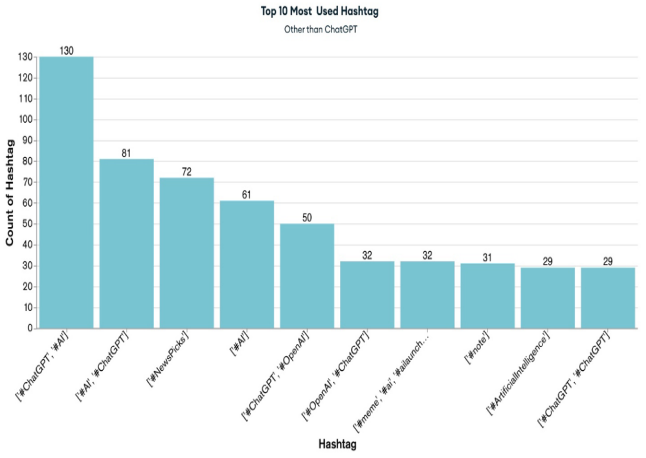

Fig-25 Top 10 most used hashtag

5. Source of Tweet:

This Visual gives the count of tweet w.r.t to their source. The visualization provides an overview of the count of tweets posted from different devices/sources, giving users a quick and easy way to understand the popularity of different platforms and devices among the users.

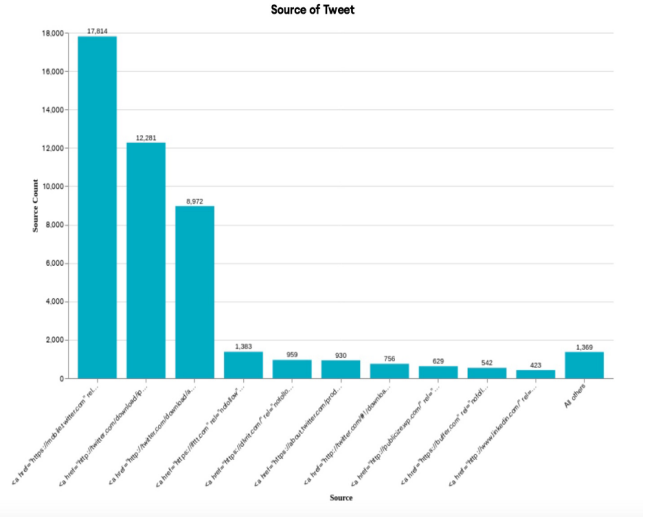

Fig-26 Source of tweet

6. Top 10 Most Quoted Tweet:

This visualization represents top tweets that have been quoted. Analyzing the top quoted tweets can provide valuable insights into the content and messaging that resonates with the audience.

SJSU CHATGPT TWITTER ANALYSIS

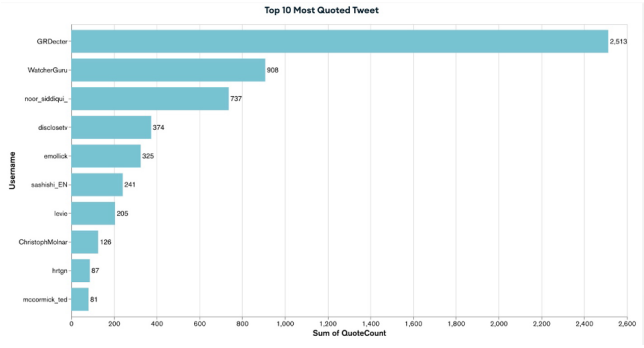

Fig-27 Top 10 Most Quoted tweet

7. Most Viral Tweet (wrt Retweet):

Here we are analyzing the most viral tweets wrt to their retweet counts. The most retweeted tweets are likely to have gained significant attention and engagement from the audience.

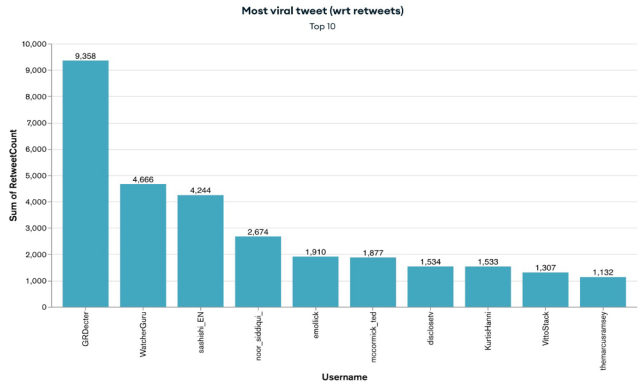

Fig-28 Most Viral Tweets

8. Time at which there are more Retweet Count, Reply Count, Like Count:

This visualization can provide valuable insights into the behavior of the audience and help to optimize the timing of tweets for maximum engagement.

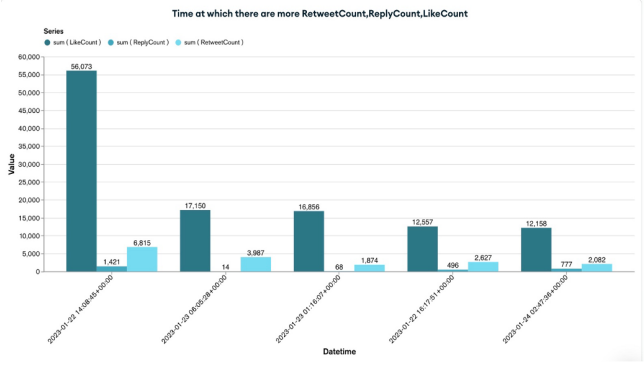

Fig-29 Time at which there are more Retweet Count, Reply Count, Like Count

9. Like Count Vs Retweet Count:

The below scatter plot shows us the distribution for like count and retweet count. This visualization is a useful tool for understanding the relationship between these two important engagement metrics on Twitter.

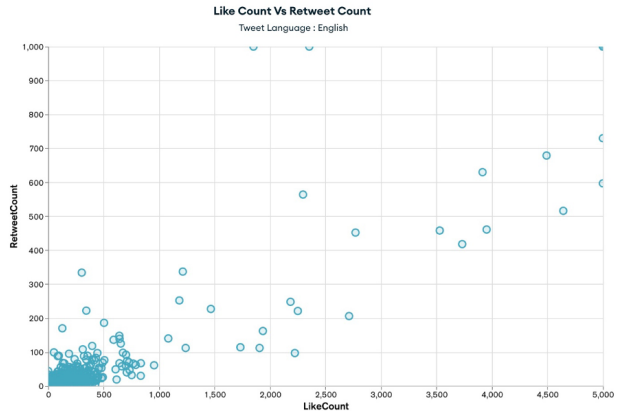

Fig-30 Like Count Vs Retweet Count

10. Media which has highest engagement rate w.r.t Like Count, Reply Count and Retweet Count:

The below visual shows engagement based on three important metrics. Analyzing the media that has the highest engagement rate in terms of Like Count, Reply Count, and Retweet Count can provide valuable insights into the type of content that resonates the most with the audience.

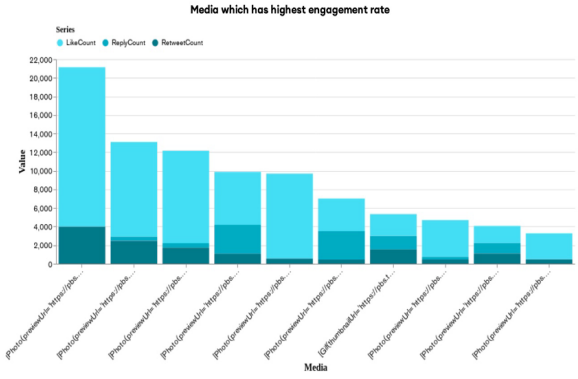

Fig-31 Media which has highest engagement rate w.r.t Like Count, Reply Count and Retweet Count.

11. Top Users with Maximum Tweet and Media Used:

The table below shows us username, tweet count and count of media. The data is sorted based on the Media Count in descending order. Conditional formatting is applied on the Number of tweets.

SJSU CHATGPT TWITTER ANALYSIS

Top Users with Maximum Tweet and Media Used

Username	Count of Media	Number of Tweets
VeilleCyber3	38	38
AaronMarcelineo	29	30
itsrohitchouhan	27	27
RubenKarvalho	27	27
genericgranola	26	26
MidJourneyAI_	25	25
Total	9,502	50,001

Fig-32 Top Users with Maximum Tweet and Media Used

12. Number of Tweets with Media:

Below gauge shows us the number of tweets that have used media in it. Analyzing the number of tweets with media can provide insights into how often users are incorporating media into their tweets.

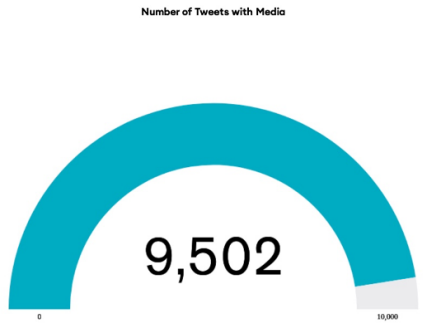

Fig-33 Number of tweets with media

13. Retweet and Quoted Tweet Engagement based on Media:

The Retweet and Quoted Tweet Engagement based on Media visualization provides valuable insights into the effectiveness of different media types in driving engagement and increasing the reach of tweets.

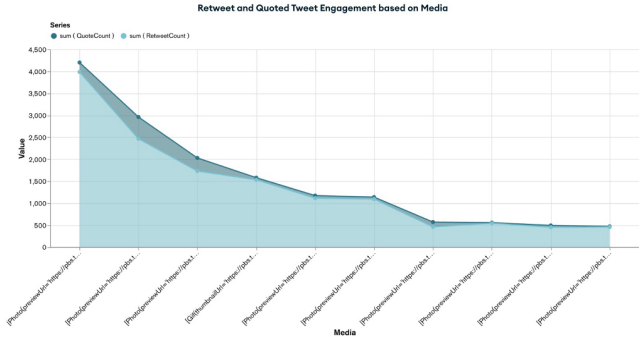

Fig-34 Retweet and Quoted Tweet Engagement based on Media

12. ACCESS PRIVILEGES

MongoDB uses a role-based access control system to manage access privileges. Users are given specific roles that determine what actions they can perform on a database. These roles can have different levels of access at different points in the database hierarchy and can be set up for specific databases or the entire cluster. There are built-in roles such as read, readWrite, and userAdmin that can be assigned to users, and custom roles can be created to provide more fine-grained access controls.

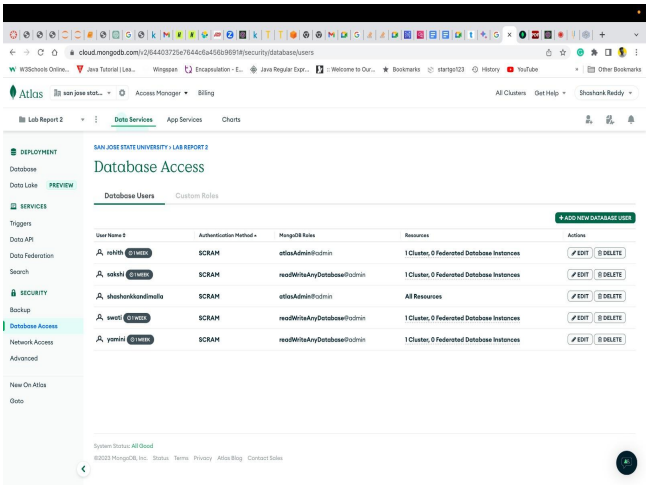

Fig-35.a Access privileges of the team mates

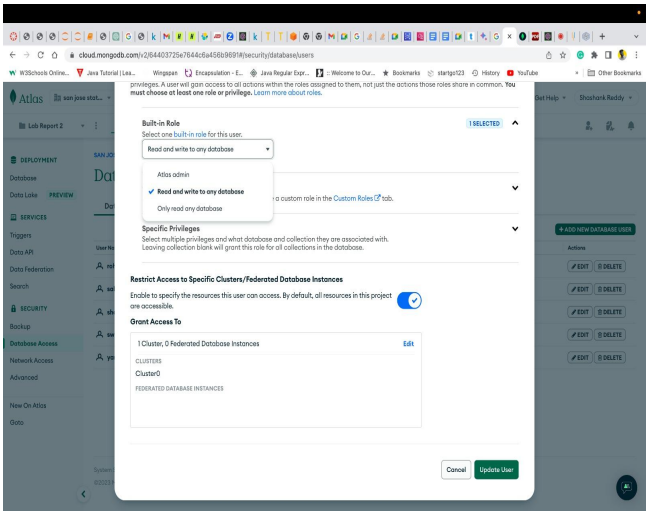

Fig-35.b Assigning privileges to team mates

SJSU CHATGPT TWITTER ANALYSIS

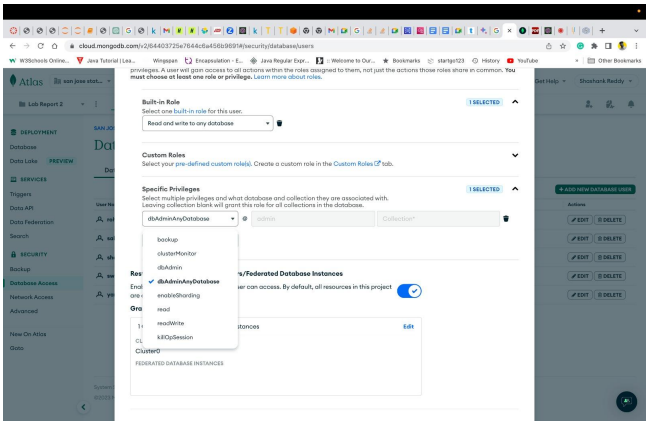

Fig-35.c Assigning special privileges to team mates

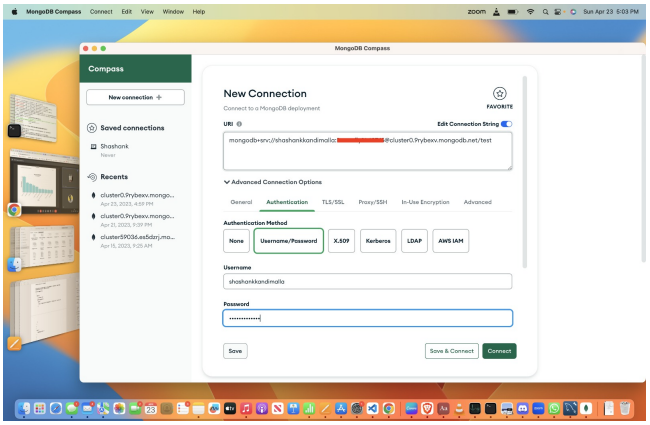

Fig-36.b connectivity to Mongodb atlas through mongodb compass.

13. CONNECTIVITY TO MONGODB IN CLOUD

In order to connect to MongoDB in the cloud, you must first register with a MongoDB service provider like MongoDB Atlas, then create a MongoDB instance with the desired configuration and create MongoDB users with the proper permissions. We then have to generate a connection string and connect using a MongoDB client or driver like MongoDB Compass, and finally test the connection to ensure that the instance is reachable and operating as intended.

The step by step connection is there in the mongodb connectivity pdf.

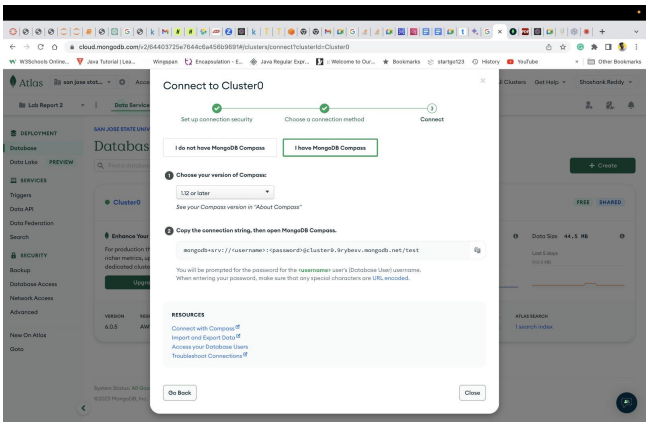

Fig-36.a connectivity to Mongodb atlas through mongodb compass.

14. LOGGING OF DB

MongoDB Atlas provides a secure way to log in to a database. Users can log in using their MongoDB Atlas account credentials, which are encrypted and transmitted securely over HTTPS. Once logged in, users can access a dashboard that displays important information about their databases, such as performance metrics and usage statistics. The dashboard also provides tools for managing and monitoring database activity, including the ability to create and manage users and roles, set up alerts, and view logs. MongoDB Atlas also offers multi-factor authentication and IP whitelisting to provide additional security measures for logging in.

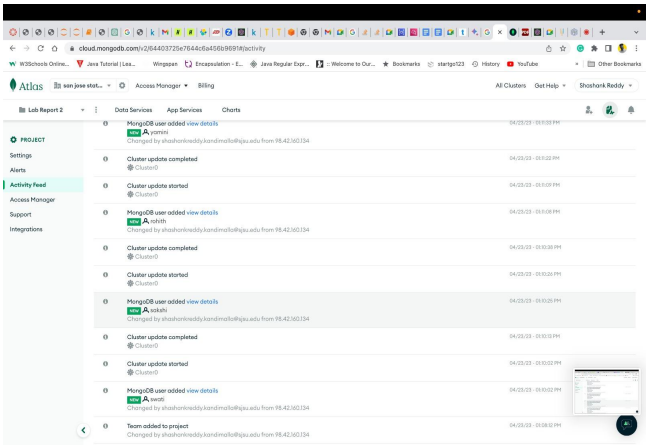

Fig-37.a Logging in mongodb

SJSU CHATGPT TWITTER ANALYSIS

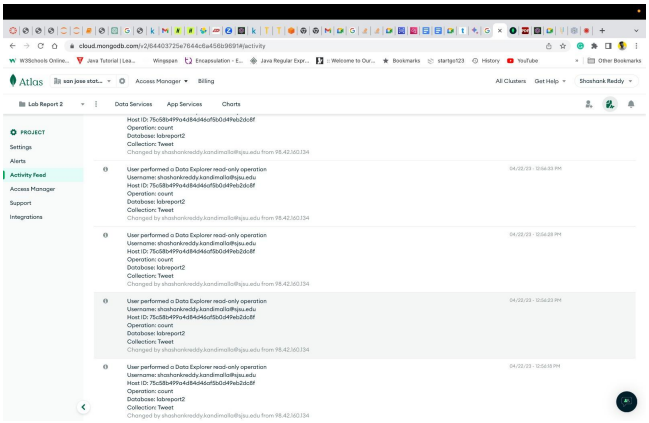

Fig-37.b Logging in mongodb

15. PERFORMANCE MEASUREMENT

Fig-38 Execution Status

We have used Explain function() to get information about the performance of a MongoDB query. When we called the explain() function on queries, MongoDB returned information about the query execution plan, including which indexes were used and its execution time.

Fig-39.a Metrics for mongodb server.

Metric	Description
Connections	Tracks the number of client connections to the MongoDB server and indicates high traffic and resource needs.
Logical Size	Measures the volume and complexity of data, including metadata and indexes, stored on the MongoDB server.
Network	Monitors network utilization, including incoming and outgoing traffic, and indicates high traffic and bandwidth needs.
Opcounters	Keeps track of the number of operations, categorized by type (inserts, queries, updates, deletes, etc.), performed by the MongoDB server to identify performance issues.

Fig-39.b Description of metrics for mongodb server.

Comparison of execution time in my sql and no sql :

	MY SQL (Execution Time (Sec))	NO SQL (Execution Time(Sec))
Users most discussing about it on Twitter(Tweets)	0.157	0.227
count of tweets from different devices(tweets)	0.406	0.286
Websites which are not secure in outlinks	0.312	0.128
Analyzing which websites or pages are most commonly shared on Twitter(outlines)	0.219	0.164
Most viral tweet (wrt retweets) about the ChatGPT(tweet_counts,tweets)	0.203	0.237
Most discussed tweet (wrt replies and conversation)(tweet_counts, tweets)	0.578	0.308
Latest tweet(tweet)	0.453	0.168
Hashtags used prominently for the tweets about ChatGPT(tweet_hashtags, tweet_hashtag_mappings)	0.203	0.397

Fig-40 Comparison of execution time in my sql and no sql

For instance, we repeatedly executed the query Most discussed tweet (wrt replies and conversation) (tweet_counts, tweets) to obtain its execution times in MySQL and NoSQL ,(1^{st}- 0.578 Sec, 2^{nd}- 0.532,3^{rd}- 0.703) in MySQL and 1^{st} - 0.308,2^{nd} - 0.297,3^{rd} - 0.251) in NoSQL respectively. The Average Execution time in My SQL is 0.604 whereas in NoSQL it is 0.285, making it very evident that NoSQL is quicker than MySQL.

From the above table we can conclude, NoSQL is approximately 24.29% faster than MySQL in executing queries.

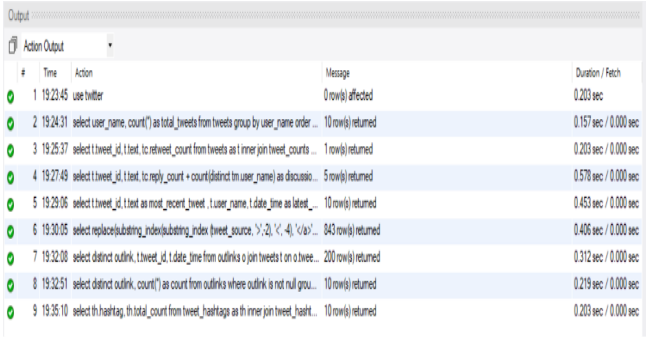

Fig-41 Execution time of queries in MySQL

SJSU CHATGPT TWITTER ANALYSIS

16. CONCLUSION AND FUTURE WORK

In order to gain insight into the online discussion surrounding this language model on Twitter, we have proposed an app to analyze tweets regarding ChatGPT. In this lab report, we have demonstrated the use of MongoDB, a NoSQL database, to analyze tweets related to the ChatGPT language model on Twitter. We have shown how MongoDB's document-oriented approach can be used to store and query tweets efficiently. The queries we have implemented provide valuable insights into the online discussion surrounding ChatGPT, including identifying the most popular tweets, users, and hashtags. We have also demonstrated how data can be moved to the cloud using a cloud-native strategy.

However, there is still room for future work to improve the capabilities of our application. Adding sentiment analysis could help understand the general sentiments regarding ChatGPT reflected in tweets, and expanding the analysis to other language models could provide a broader view of natural language processing. Additionally, machine learning models can be applied to enhance the precision of identifying popular tweets, users, and hashtags. The application could also be extended to analyze tweets from other social media platforms, providing a more comprehensive examination of online discussions.

Overall, this lab report shows how MongoDB can be used effectively to analyze social media data and provides a foundation for future research and development in social media analysis and natural language processing.

17. REFERENCES

[1] chatgpt twitter data analysis
https://www.kaggle.com/datasets/tariqsays/chatgpt-twitter-dataset.

[2] Denormalization
https://www.techtarget.com/searchdatamanagement/definition/denormalization#:~:text=Denormalization%20is%20the%20process%20of,of%20each%20piece%20of%20information.

[3] Mongodb
https://www.tutorialspoint.com/mongodb/index.htm

[4] Mongodb Atlas
https://www.mongodb.com/docs/atlas/#:~:text=MongoDB%20Atlas%20is%20a%20multi,cloud%20providers%20of%20your%20choice.

[5] Fundamentals of Database Systems, 7th edition, by Elmasri and Navathe, Addison Wesley Publishing Company, Menlo Park, CA.

[6] Jeffrey A. Hoffer, Ramesh Venkataraman, and Heikki Topi. Modern Database Management.Pearson. 13th Edition. ISBN-13: 978-1292263359