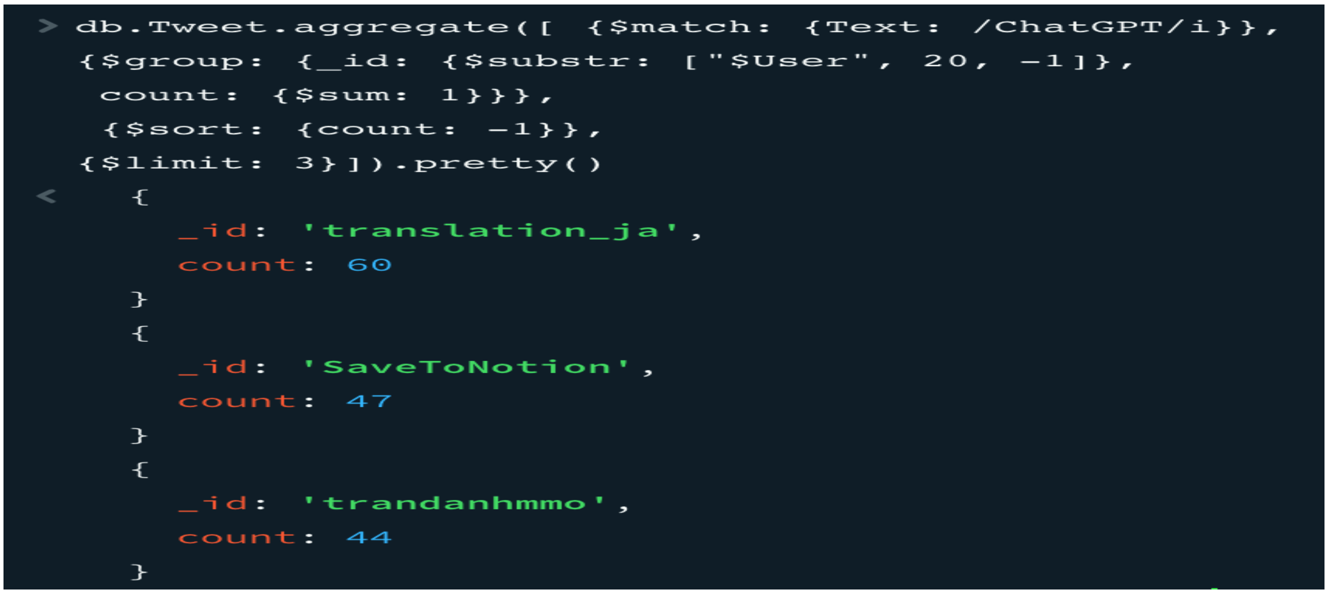
# SJSU CHATGPT TWITTER ANALYSIS

## Queries - Lab Report - 2 ( Team - 8 )

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

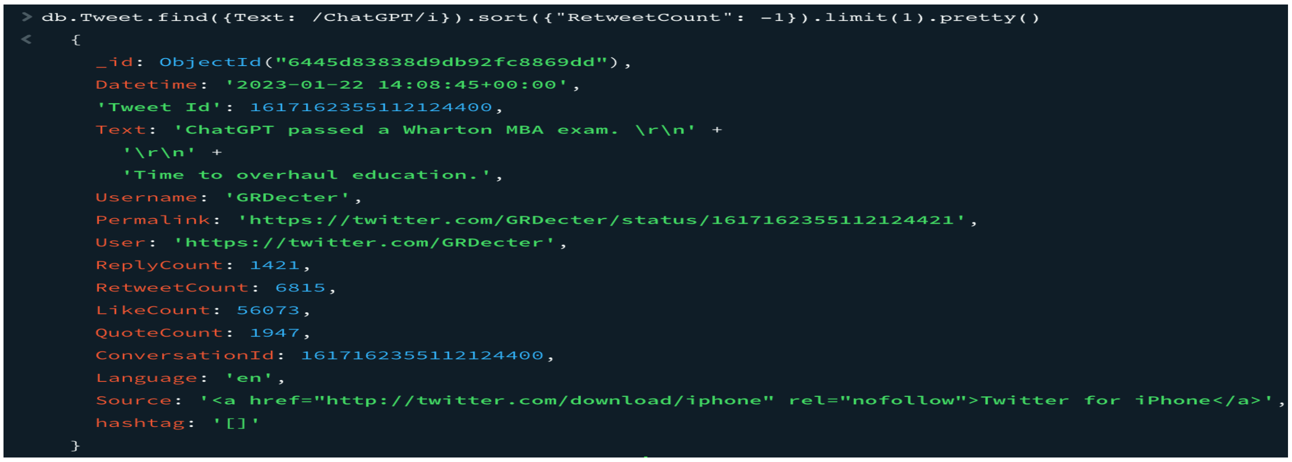
**1. Users most discussing about it on twitter(tweets):**
**Query:**
```
db.Tweet.aggregate([ {$match: {Text: /ChatGPT/i}},
{$group: {_id: {$substr: ["$User", 20, -1]},
 count: {$sum: 1}}},
 {$sort: {count: -1}},
{$limit: 3}]).pretty()
```

**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.


```
> db.Tweet.aggregate([ {$match: {Text: /ChatGPT/i}},
  {$group: {_id: {$substr: ["$User", 20, -1]},
   count: {$sum: 1}}},
   {$sort: {count: -1}},
  {$limit: 3}]).pretty()
<   {
      _id: 'translation_ja',
      count: 60
    }
    {
      _id: 'SaveToNotion',
      count: 47
    }
    {
      _id: 'trandanhmmo',
      count: 44
    }
```

**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.

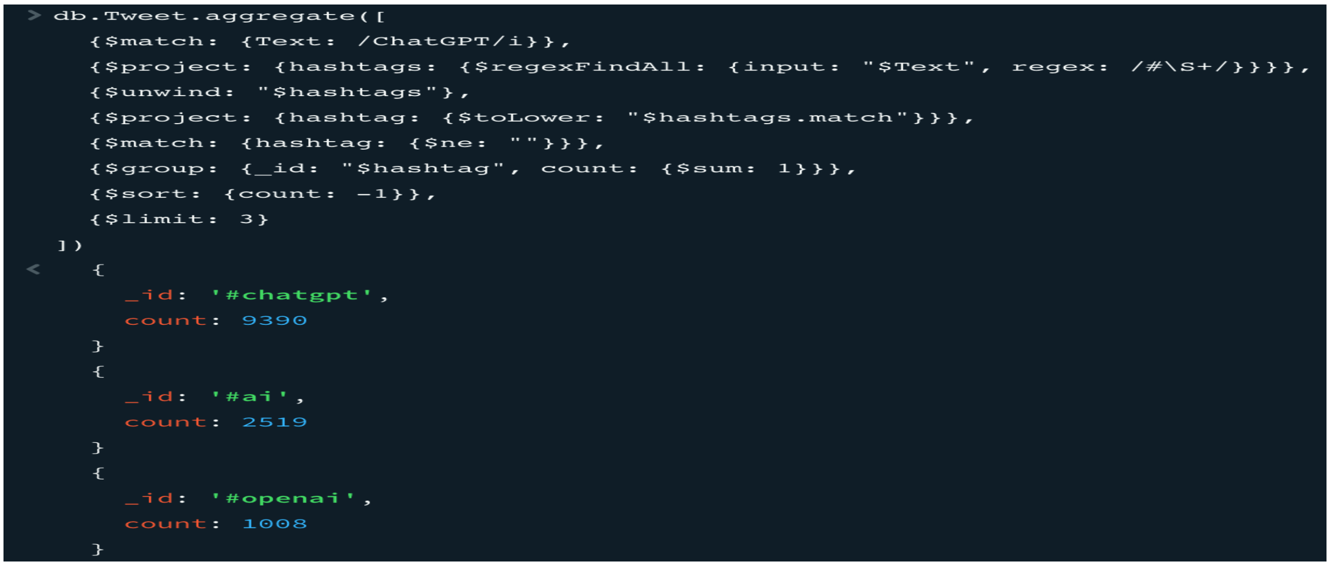


```
> db.Tweet.find({Text: /ChatGPT/i}).sort({"RetweetCount": -1}).limit(1).pretty()
<   {
      _id: ObjectId("6445d83838d9db92fc8869dd"),
      Datetime: '2023-01-22 14:08:45+00:00',
      'Tweet Id': 1617162355112124400,
      Text: 'ChatGPT passed a Wharton MBA exam. \r\n' +
        '\r\n' +
        'Time to overhaul education.',
      Username: 'GRDecter',
      Permalink: 'https://twitter.com/GRDecter/status/1617162355112124421',
      User: 'https://twitter.com/GRDecter',
      ReplyCount: 1421,
      RetweetCount: 6815,
      LikeCount: 56073,
      QuoteCount: 1947,
      ConversationId: 1617162355112124400,
      Language: 'en',
      Source: '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>',
      hashtag: '[]'
    }
```

**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}
])
```

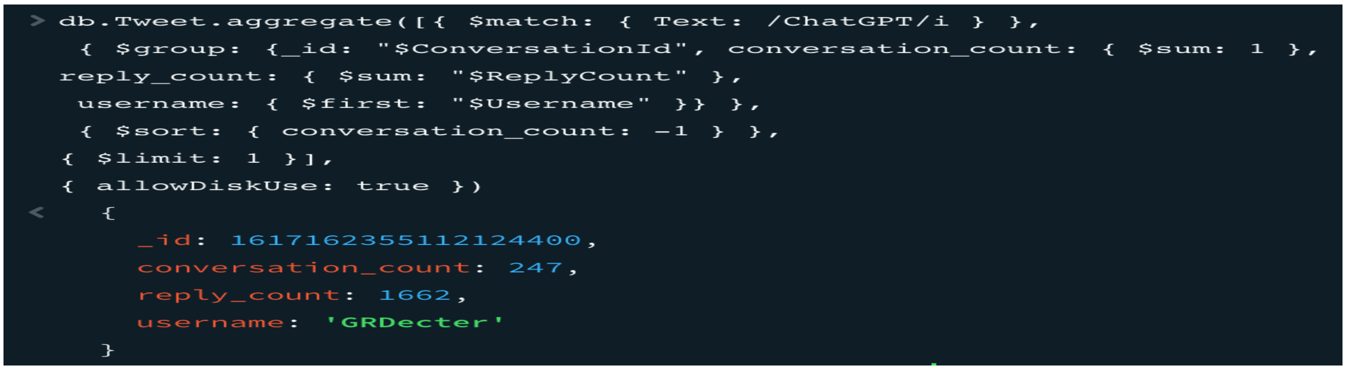
**Interaction with database:** The query searches for tweets in a MongoDB collection that contain the string "ChatGPT" in their Text field, extracts and counts the hashtags used in those tweets, removes any empty hashtags, and sorts the remaining hashtags in descending order based on the count of their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

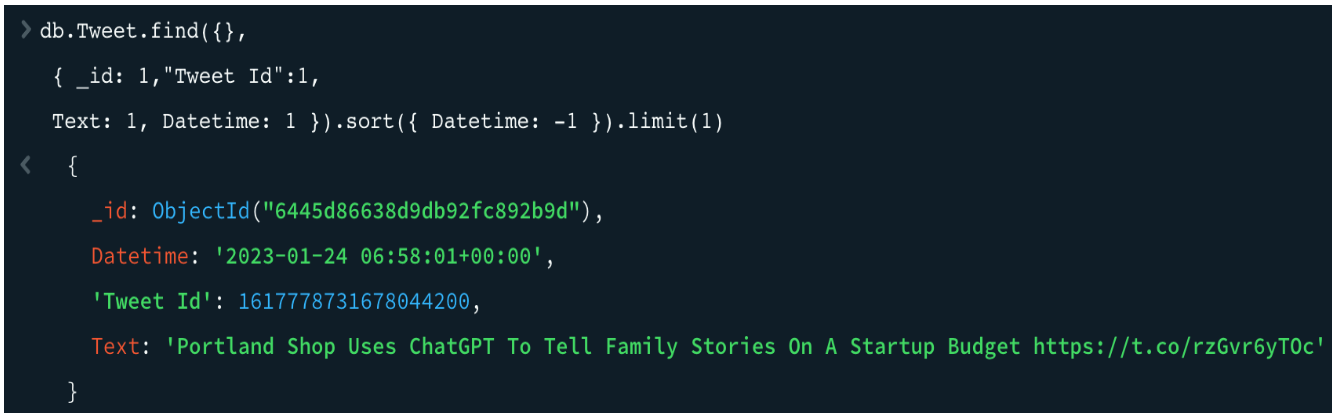
**4. Most discussed tweet (wrt replies and conversation) (tweet_counts, tweets):**
**Query:**
db.Tweet.aggregate([{ $match: { Text: /ChatGPT/i } },
 { $group: {_id: "$ConversationId", conversation_count: { $sum: 1 },
reply_count: { $sum: "$ReplyCount" },
 username: { $first: "$Username" }} },
 { $sort: { 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'
    }
```

**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.

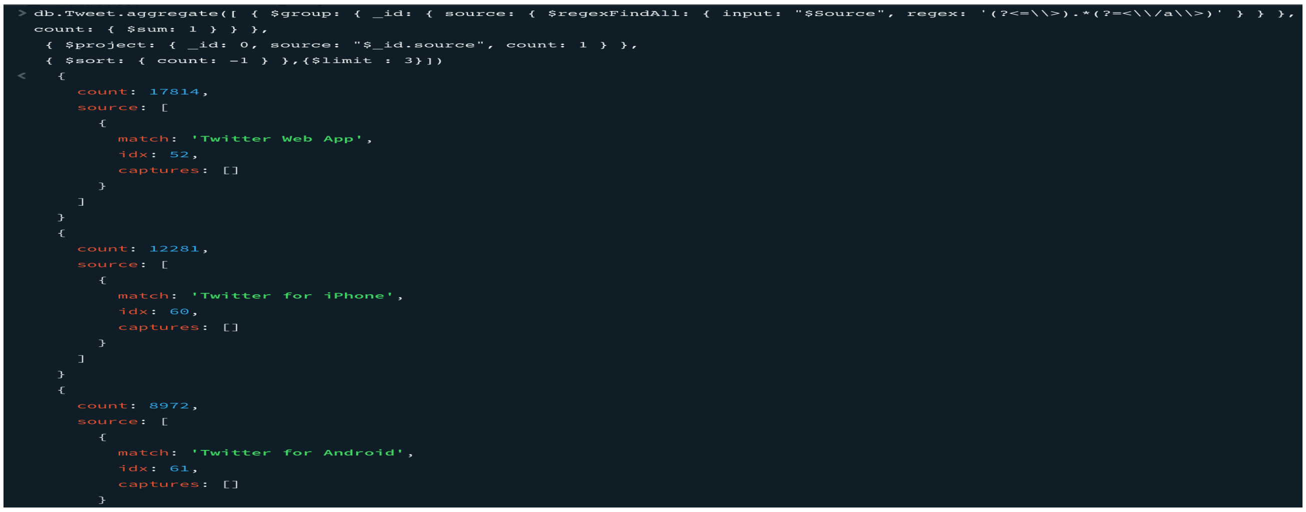

```
> db.Tweet.find({},
  { _id: 1,"Tweet Id":1,
  Text: 1, Datetime: 1 }).sort({ Datetime: -1 }).limit(1)
<  {
     _id: ObjectId("6445d86638d9db92fc892b9d"),
     Datetime: '2023-01-24 06:58:01+00:00',
     'Tweet Id': 1617778731678044200,
     Text: 'Portland Shop Uses ChatGPT To Tell Family Stories On A Startup Budget https://t.co/rzGvr6yTOc'
   }
```

**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.



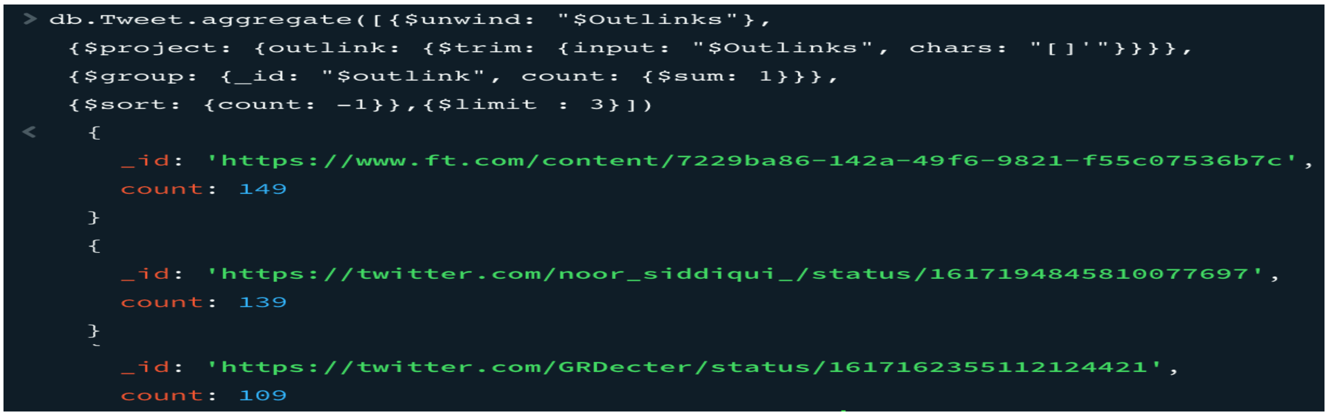
```
> db.Tweet.aggregate([ { $group: { _id: { source: { $regexFindAll: { input: "$Source", regex: '(?<=\\>).*(?=<\\/a\\>)' } } },
  count: { $sum: 1 } } },
   { $project: { _id: 0, source: "$_id.source", count: 1 } },
   { $sort: { count: -1 } },{$limit : 3}])
<   {
      count: 17814,
      source: [
        {
          match: 'Twitter Web App',
          idx: 52,
          captures: []
        }
      ]
    }
    {
      count: 12281,
      source: [
        {
          match: 'Twitter for iPhone',
          idx: 60,
          captures: []
        }
      ]
    }
    {
      count: 8972,
      source: [
        {
          match: 'Twitter for Android',
          idx: 61,
          captures: []
        }
```

**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.



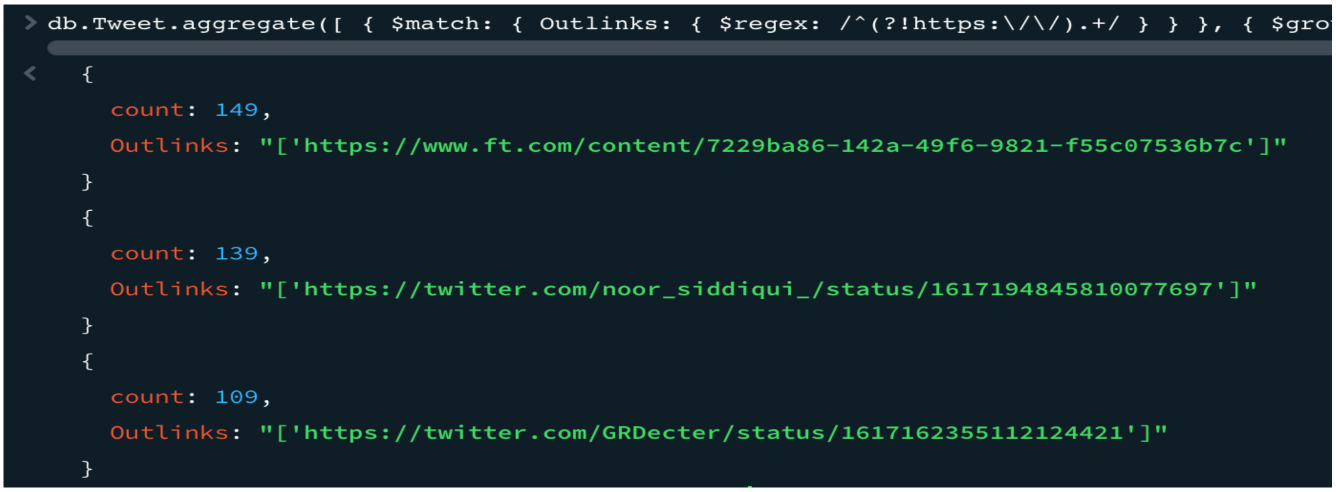
```
> db.Tweet.aggregate([{$unwind: "$Outlinks"},
   {$project: {outlink: {$trim: {input: "$Outlinks", chars: "[]'"}}}},
   {$group: {_id: "$outlink", count: {$sum: 1}}},
   {$sort: {count: -1}},{$limit : 3}])
<   {
      _id: 'https://www.ft.com/content/7229ba86-142a-49f6-9821-f55c07536b7c',
      count: 149
    }
    {
      _id: 'https://twitter.com/noor_siddiqui_/status/1617194845810077697',
      count: 139
    }
      _id: 'https://twitter.com/GRDecter/status/1617162355112124421',
      count: 109
```

**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.



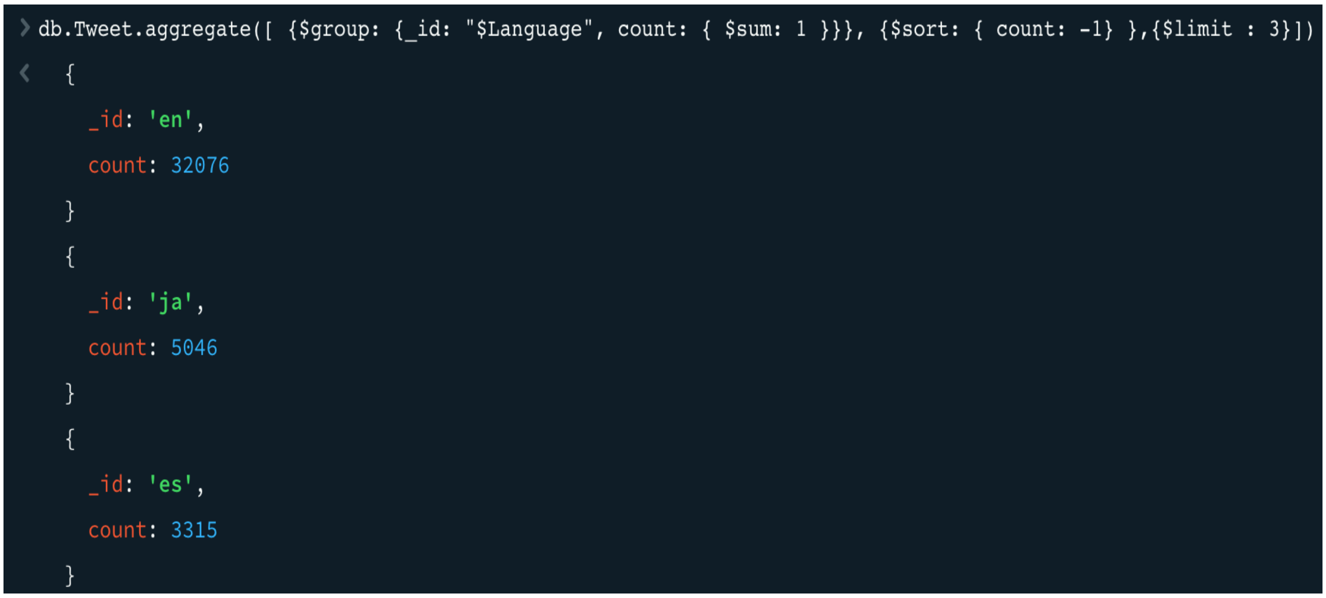
```
> db.Tweet.aggregate([ { $match: { Outlinks: { $regex: /^(?!https:\/\/).+/ } } }, { $gro
<   {
      count: 149,
      Outlinks: "['https://www.ft.com/content/7229ba86-142a-49f6-9821-f55c07536b7c']"
    }
    {
      count: 139,
      Outlinks: "['https://twitter.com/noor_siddiqui_/status/1617194845810077697']"
    }
    {
      count: 109,
      Outlinks: "['https://twitter.com/GRDecter/status/1617162355112124421']"
    }
```

**9. tweets by language:**

**Query:**

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

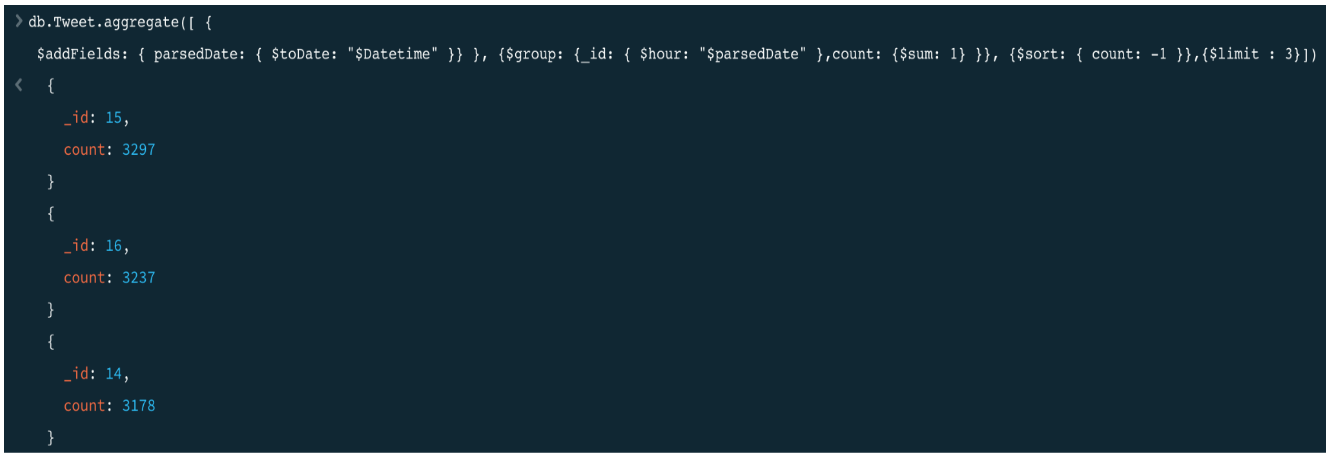
**Interaction with database:** The query uses the MongoDB aggregation pipeline to group tweets by language and count the number of tweets in each language. It then sorts the results in descending order based on the count. This query can be used to identify the number of tweets in each language present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



```
> db.Tweet.aggregate([ {$group: {_id: "$Language", count: { $sum: 1 }}}, {$sort: { count: -1} },{$limit : 3}])
<   {
      _id: 'en',
      count: 32076
    }
    {
      _id: 'ja',
      count: 5046
    }
    {
      _id: 'es',
      count: 3315
    }
```

**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}])

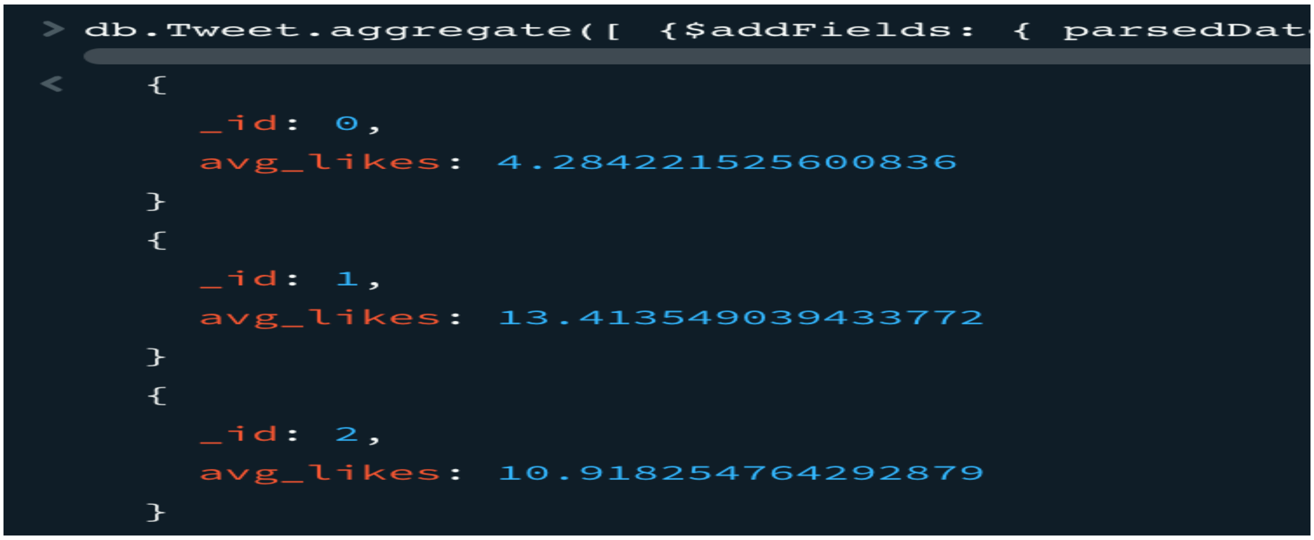
**Interaction with database:** This query adds a new field to the documents in the Tweet collection that represents the parsed date from the Datetime field. It then groups the tweets by the hour of the parsed date and counts the number of tweets in each group. Finally, the results are sorted in descending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



```
> db.Tweet.aggregate([ {
  $addFields: { parsedDate: { $toDate: "$Datetime" }} }, {$group: {_id: { $hour: "$parsedDate" },count: {$sum: 1} }}, {$sort: { count: -1 }},{$limit : 3}])
<  {
     _id: 15,
     count: 3297
   }
   {
     _id: 16,
     count: 3237
   }
   {
     _id: 14,
     count: 3178
   }
```

**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.

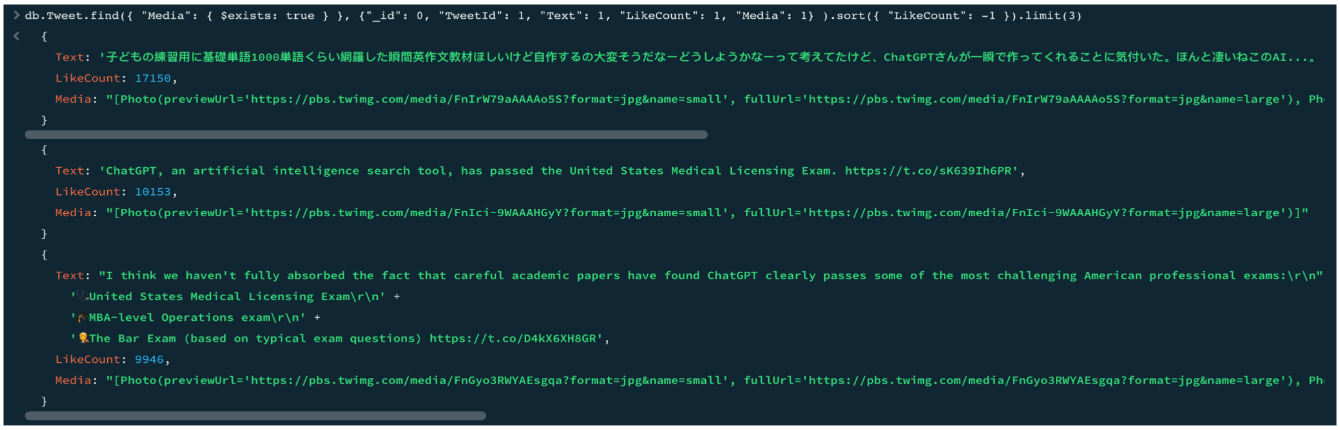


```
> db.Tweet.aggregate([ {$addFields: { parsedDat
<   {
      _id: 0,
      avg_likes: 4.284215256008 36
    }
    {
      _id: 1,
      avg_likes: 13.413549039433772
    }
    {
      _id: 2,
      avg_likes: 10.918254764292879
    }
```

## 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.

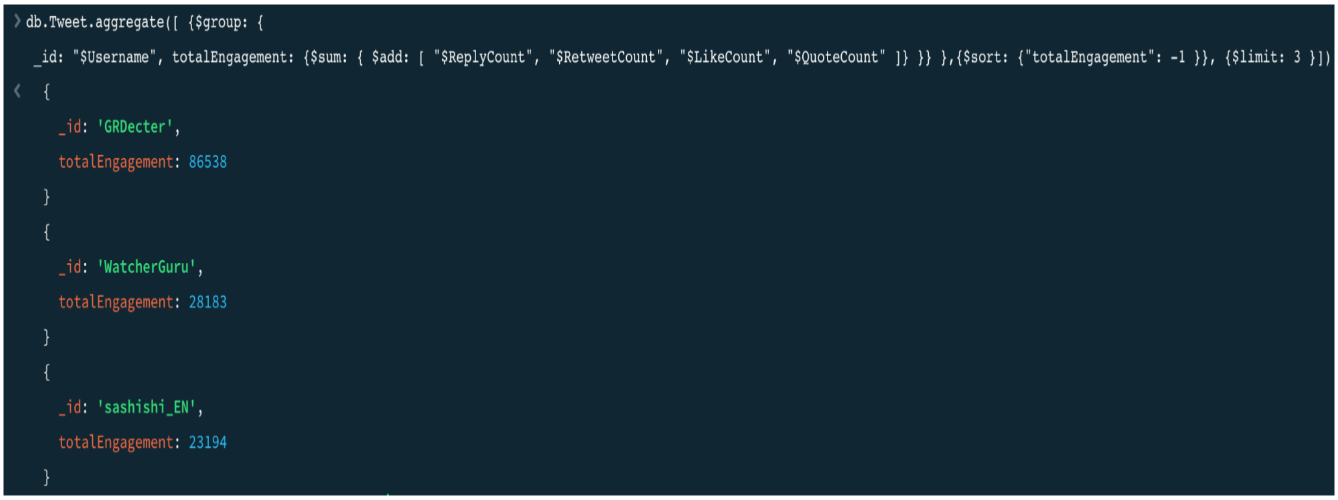


```
> db.Tweet.find({ "Media": { $exists: true } }, {"_id": 0, "TweetId": 1, "Text": 1, "LikeCount": 1, "Media": 1} ).sort({ "LikeCount": -1 }).limit(3)
<   {
      Text: '子どもの練習用に基礎単語1000単語くらい網羅した瞬間英作文教材ほしいけど自作するの大変そうだなーどうしようかなーって考えてたけど、ChatGPTさんが一瞬で作ってくれることに気付いた。ほんと凄いねこのAI...。
      LikeCount: 17150,
      Media: "[Photo(previewUrl='https://pbs.twimg.com/media/FnIrW79aAAAAo5S?format=jpg&name=small', fullUrl='https://pbs.twimg.com/media/FnIrW79aAAAAo5S?format=jpg&name=large'), Ph
    }
    {
      Text: 'ChatGPT, an artificial intelligence search tool, has passed the United States Medical Licensing Exam. https://t.co/sK639Ih6PR',
      LikeCount: 10153,
      Media: "[Photo(previewUrl='https://pbs.twimg.com/media/FnIci-9WAAAHGyY?format=jpg&name=small', fullUrl='https://pbs.twimg.com/media/FnIci-9WAAAHGyY?format=jpg&name=large')]"
    }
    {
      Text: "I think we haven't fully absorbed the fact that careful academic papers have found ChatGPT clearly passes some of the most challenging American professional exams:\r\n"
        '🩺United States Medical Licensing Exam\r\n' +
        '🎓MBA-level Operations exam\r\n' +
        '🧑‍⚖️The Bar Exam (based on typical exam questions) https://t.co/D4kX6XH8GR',
      LikeCount: 9946,
      Media: "[Photo(previewUrl='https://pbs.twimg.com/media/FnGyo3RWYAEsgqa?format=jpg&name=small', fullUrl='https://pbs.twimg.com/media/FnGyo3RWYAEsgqa?format=jpg&name=large'), Ph
    }
```

## 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.



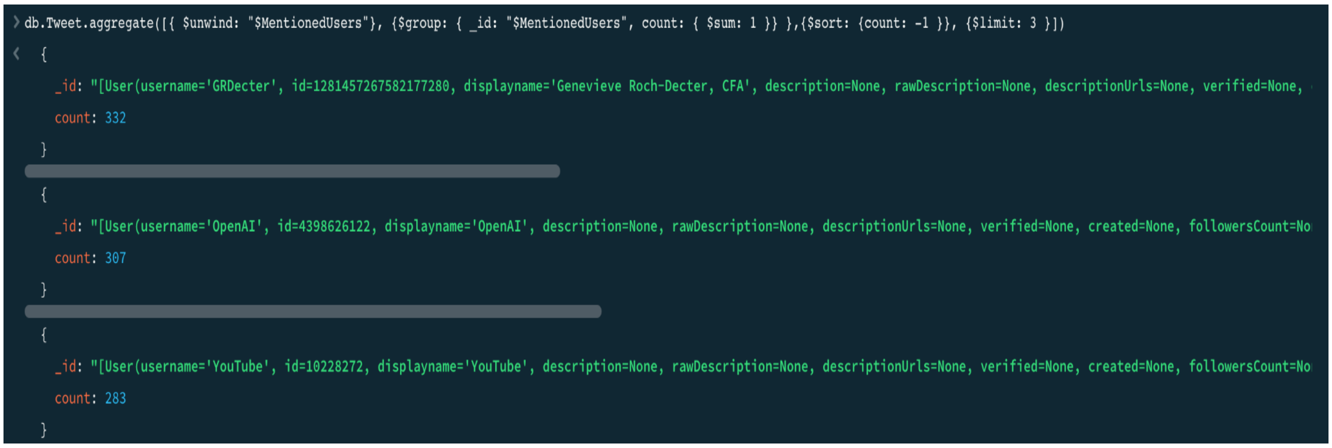
```
> db.Tweet.aggregate([ {$group: {
   _id: "$Username", totalEngagement: {$sum: { $add: [ "$ReplyCount", "$RetweetCount", "$LikeCount", "$QuoteCount" ]} }} },{$sort: {"totalEngagement": -1 }}, {$limit: 3 }])
<  {
     _id: 'GRDecter',
     totalEngagement: 86538
   }
   {
     _id: 'WatcherGuru',
     totalEngagement: 28183
   }
   {
     _id: 'sashishi_EN',
     totalEngagement: 23194
   }
```

**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.



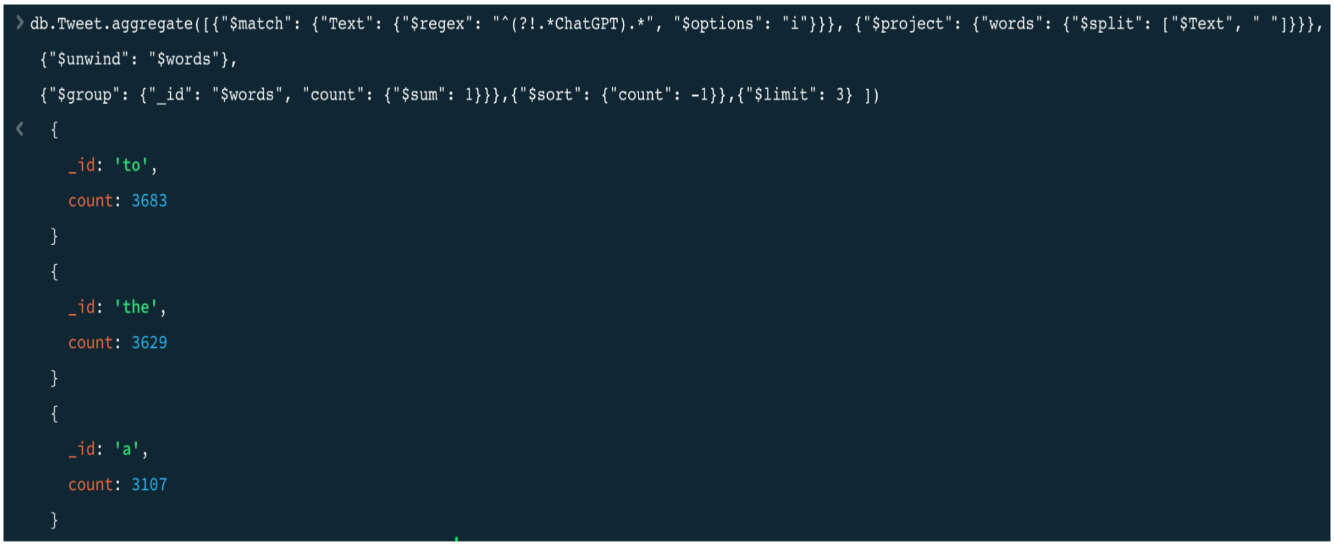
```
> db.Tweet.aggregate([{ $unwind: "$MentionedUsers"}, {$group: { _id: "$MentionedUsers", count: { $sum: 1 }} },{$sort: {count: -1 }}, {$limit: 3 }])
<  {
     _id: "[User(username='GRDecter', id=1281457267582177280, displayname='Genevieve Roch-Decter, CFA', description=None, rawDescription=None, descriptionUrls=None, verified=None,
     count: 332
   }
   {
     _id: "[User(username='OpenAI', id=4398626122, displayname='OpenAI', description=None, rawDescription=None, descriptionUrls=None, verified=None, created=None, followersCount=No
     count: 307
   }
   {
     _id: "[User(username='YouTube', id=10228272, displayname='YouTube', description=None, rawDescription=None, descriptionUrls=None, verified=None, created=None, followersCount=No
     count: 283
   }
```

**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} ])

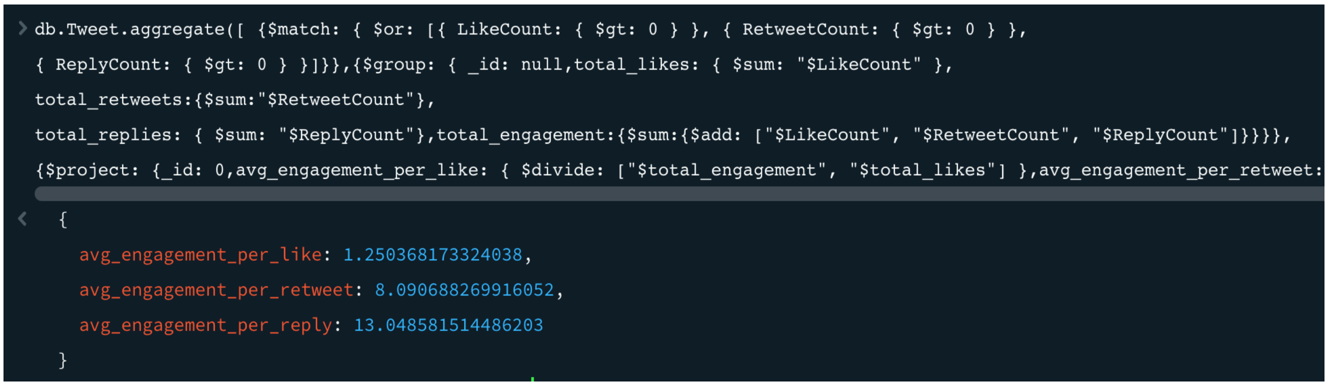
**Interaction with database:** This query matches tweets that don't contain the phrase "ChatGPT" in the text field, splits the text into words, unwinds the resulting array, groups by each unique word and counts their occurrences, sorts the result in descending order of count, and returns the top 10 most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



```
> 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} ])
<  {
     _id: 'to',
     count: 3683
   }
   {
     _id: 'the',
     count: 3629
   }
   {
     _id: 'a',
     count: 3107
   }
```

**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"] } }} ])

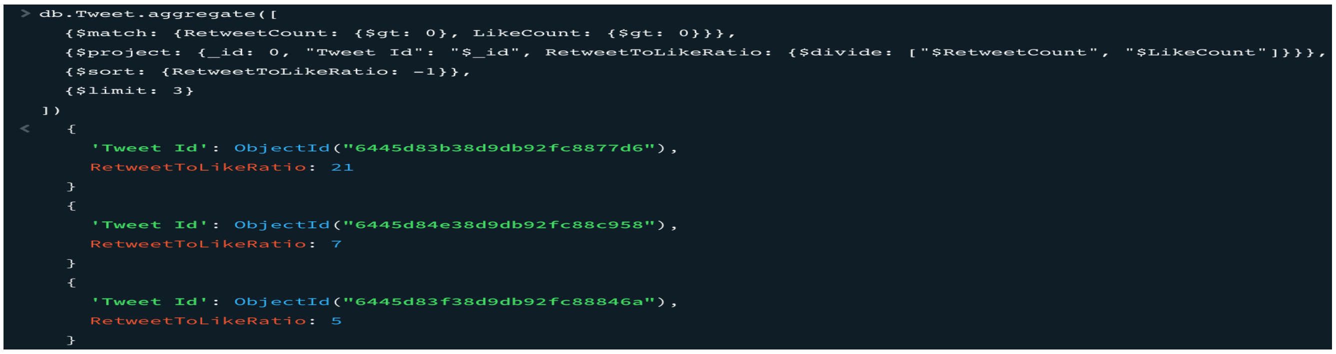
**Interaction with database:** This MongoDB query calculates the average engagement per like, retweet, and reply across the entire collection of tweets. It first matches tweets with at least one engagement (like, retweet, or reply), then groups all tweets to calculate the total number of likes, retweets, 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.



```
> 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:
<   {
      avg_engagement_per_like: 1.250368173324038,
      avg_engagement_per_retweet: 8.090688269916052,
      avg_engagement_per_reply: 13.048581514486203
    }
```

**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} ])

**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.

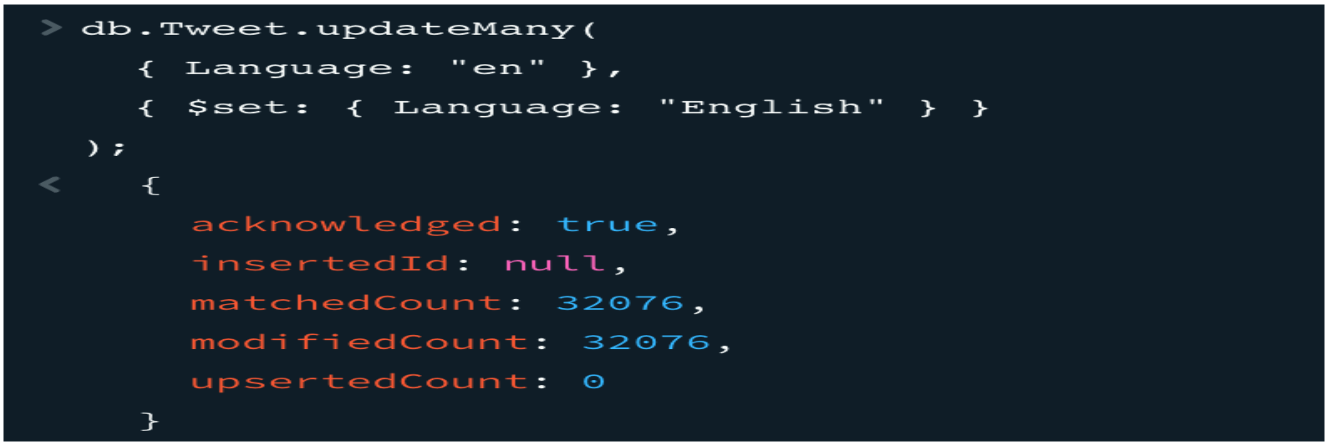


```
> db.Tweet.aggregate([
    {$match: {RetweetCount: {$gt: 0}, LikeCount: {$gt: 0}}},
    {$project: {_id: 0, "Tweet Id": "$_id", RetweetToLikeRatio: {$divide: ["$RetweetCount", "$LikeCount"]}}},
    {$sort: {RetweetToLikeRatio: -1}},
    {$limit: 3}
  ])
<   {
      'Tweet Id': ObjectId("6445d83b38d9db92fc8877d6"),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

**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".

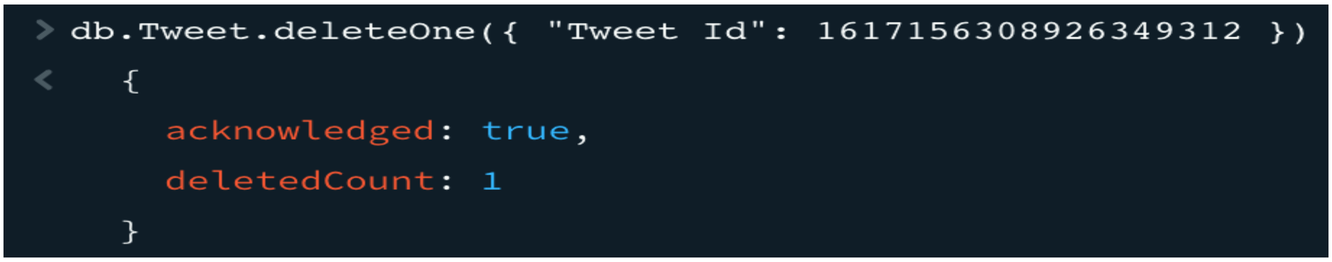


```
> db.Tweet.updateMany(
      { Language: "en" },
      { $set: { Language: "English" } }
  );
<     {
        acknowledged: true,
        insertedId: null,
        matchedCount: 32076,
        modifiedCount: 32076,
        upsertedCount: 0
      }
```

**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
    }
```

**THE END**