iOS App Store Analysis

Submitted By :

Sithara Vanmerimeethal Paleri

Introduction:

The purpose of this project is about analyzing the data for the Apple iOS app store and creating a new wireframe for the web page to display the analytics matrix based on our analysis on the available dataset. We are performing analysis on the dataset using MongoDB, Neo4j and SAS. We have also analyzed the Apple iOS app store to visualize a pseudo schema for MongoDB and a graph model for Neo4j. Our project includes the design based on our understanding of the Apple iOS app store and from the dataset. We ran various queries on MongoDB and Neo4j in order to analyze the dataset in detail and used MongoDB and Neo4j to perform some analytical analysis over data to include various metrics on the webpage wireframe that we designed.

Analysis over our Apple IOS App Store Dataset:

Analysis over the data is been summarized below as findings from Part A, B, and C -

PART - A

Pseudo Schema for MongoDB

Listed below is the pseudo schema and we are considering two collections.

1. User Collection
2. Apps Collection

We considered these two collections because we don't want to complicate by just using one collection (App) and moreover, most of the queries are related to the App collection rather than the User. Therefore, by just referencing the User collection in Apps collection is much more convenient and reduces the data storage.

In the pseudo schema, User collections, we store the user details like appleID, Name etc.
In the Apps collections we are storing details of the apps like AppID, array of reviews document, Company details, Version details documents, Overall rating of the app and rankings etc. We have attribute purchasedBy in the Apps collection which is referencing to the User collection.

We have considered this schema because we are more inclined towards finding out the answers for the questions like finding out the reviewers with the most positive or negative reviews. Finding the ranking of the app based on the category or finding the latest version of the app which has been released recently.

Pseudo Schema – USERS:

```
{
"appleId": "userone01@gmail.com"
```

"name": "userone"

"country/region":"USA"

"phone": 4678972345

"unixUserTime":"132523538289"

},

{

"appleId": "usertwo02@gmail.com"

"name": "usertwo"

"country/region":"India"

"phone": 9988972345

"unixUserTime":132523538289

},

{

"appleId": "userthree03@gmail.com"

"name": "userthree"

"country/region":"India"

"phone": 9988972345

"unixUserTime":"132523538289"

}

APPS:

{

"appId": "5342318"

"storeAppId": "FER4526"

"description": "Canva makes design amazingly simple and fun"

"price":100

"overallRating":5.0

"reviews": [

{

```
                "reviewId": "1543"

                "title": "The app we all want"

                "reviewText": "This app is so simple to use"

                "reviewDate": "26/08/2019"

                "rating": 4.7

                "reviewBy": "usertwo"

                "developerResponse": "Hello, Thank you for your response"

                },

                {

                "reviewId": "6573"

                "title": "Pretty fun app"

                "reviewText": "This app is really fun, but I have few things that would be really great if
                you could add"

                "reviewDate": "02/08/2018"

                "rating": 4.0

                "reviewBy": "userone"

                "developerResponse": "Thank you for your suggestions"

                }

        ]

"purchasedBy": ["usertwo","userfour","userthree"]

"rankings": {

              "category": "photo $ video"

              "ranks": "#34"

               }

"company": {

               "sellerName": "Canva Pvt Ltd"

               "developerWebsite": "https://canva.com"

               "copyright": "@2019"

              }
```

```
"age": "+4"
"version": [
            {
             "versionNumber": "3.2.1"
             "releaseDate": "12/04/2018"
             "size": "128"
             "versionDescription": "This version is released to update the bugs and launch new feature"
             },
            {
            "versionNumber": "3.1.4"
             "releaseDate": "07/01/2017"
             "size": "128"
             "versionDescription": "This version is released to update the bugs and launch new feature"
             } ]
"unixAppTime": 4356724381
}
```

Graph Model for IOS App Review and Ranking System for Neo4j

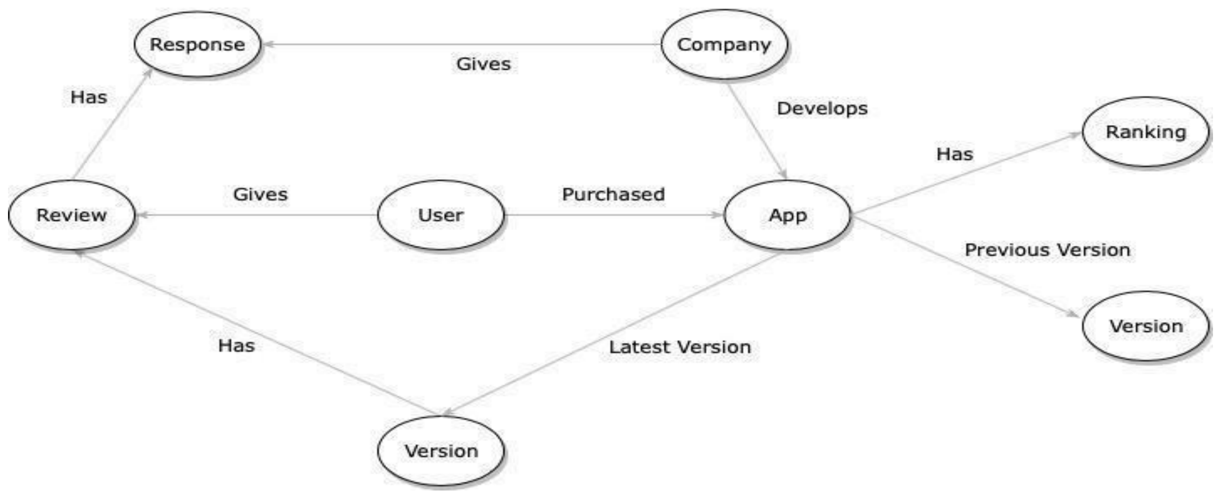
Graph to show Nodes and Relationships: This graph is to depict the nodes and their relationships for the iOS App Review and Rankings System. We have identified a total of 7 nodes as User, Apps, Version, Rankings, Company, Review and Response. The reason why we identified Version as a separate node instead of including it as App's attribute is because we need to track the version history with all the previous version and the latest one. Also, we have defined a relationship between Version (latest) with Review because as per the iOS App the reviews to latest version is only kept and as soon as the version becomes old(previous) then the Reviews attached are discarded. We have also created a separate node for Rankings so that we can easily query on the Ranks category wise and this node will also have frequent updates as and when the rankings are modified so keeping them as separate node will be easy to maintain. We have also identified Response as a different node so that it becomes easy to find the list of reviews for which the response by developer is not given. Our graph model has total 10 relationships as mentioned below- **Relationships:**

- Company DEVELOPS Apps
- Apps HAVE Rankings
- User PURCHASE Apps
- Apps HAD_PREVIOUS Version

This relationship has a property as "releaseDate" that has the value as the date when the older version was released. The reason why our model has this as the relationship property so that we can get the version history without traversing to all the version nodes which are now previous versions.

- Apps HAS_LATEST Version
- Version HAS Review
- User GIVES Review
- Company GIVES Response
- Review HAS Response

Graph To Show Nodes and Relationships



Nodes and Their Properties:

Properties of USER

- appleId
- name
- country/region
- unixUserTime

Properties of APPS

- appId
- storeAppId
- description
- price
- overallRating
- age
- unixAppTime

Properties of VERSION

- versionNumber
- releaseDate
- size
- versionDescription

Properties of REVIEW

- reviewId
- title
- reviewText
- reviewDate

- price

Properties of COMPANY

- sellerName
- developerWebsite
- copyright

Properties of RESPONSE

- developerResponse

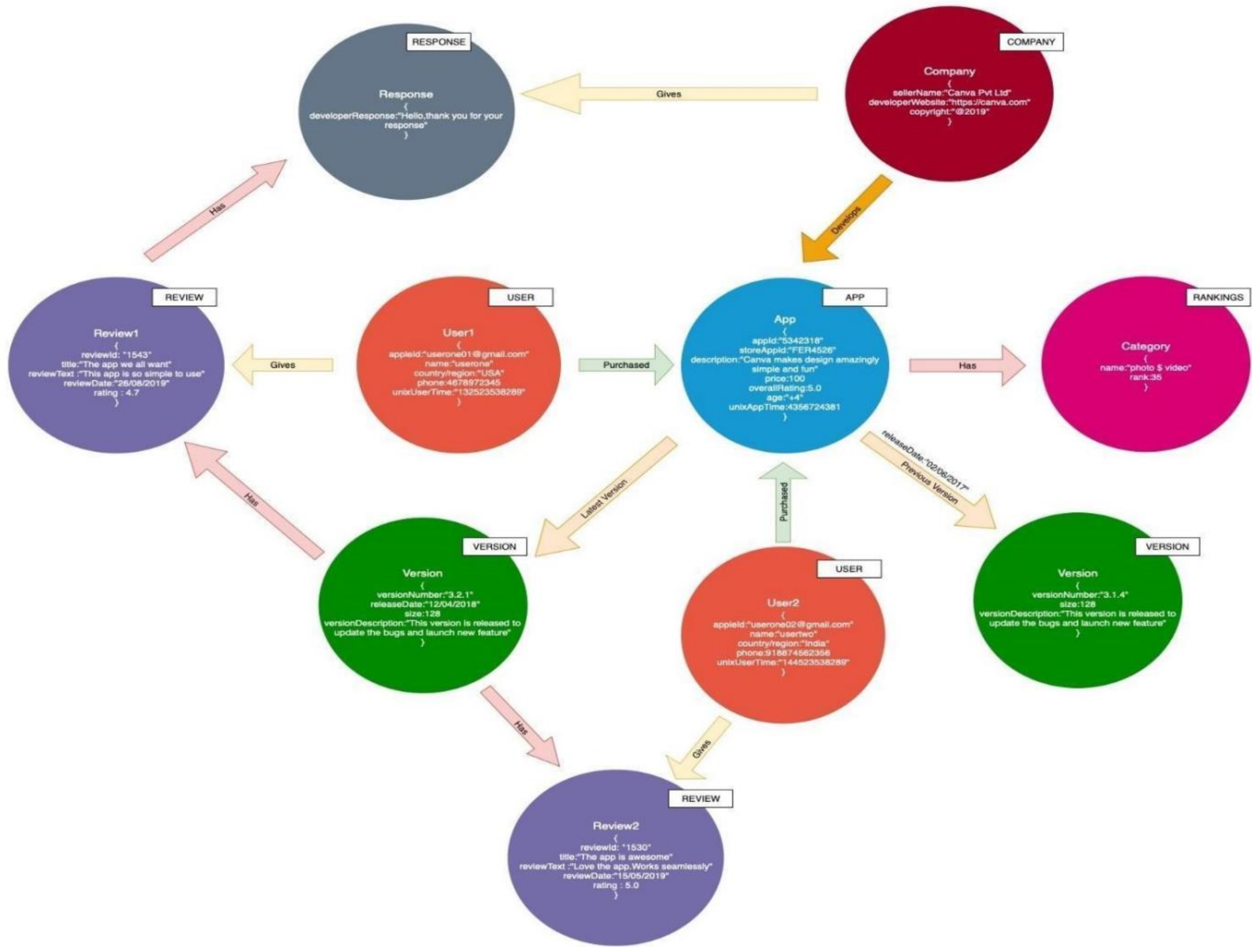
- rating

Properties of RANKINGS

- category
- rank

Graph to show iOS Apps Review and Rankings System: This graph depicts the sample nodes with their properties, its values and relationships between nodes. The below graph has one App which is developed by Company. This App is purchased by User1 and User2. This App has one old version and one latest version which has Reviews. Review1 is given by User1 and Review2 is given by User2. Review1 has Response given by Company. App has Rankings within the category they belong to.

Graph Model - IOS App System

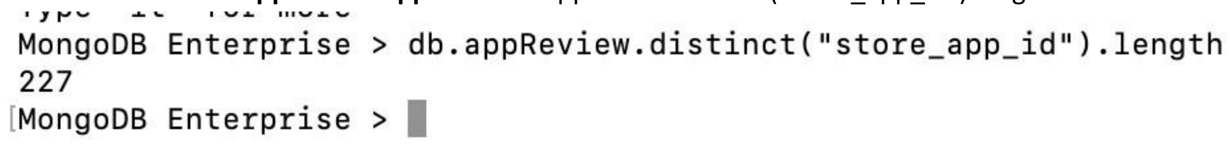


PART - B

Below mentioned queries are designed to have insights on the data that we have and also get detailed information based on which we can interpret business rules.

QUERIES:

1. **To find the number apps in the app store** db.appReview.distinct("store_app_id").length

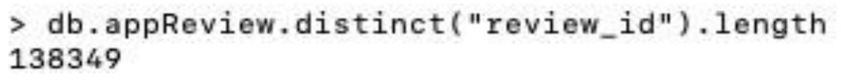


```
MongoDB Enterprise > db.appReview.distinct("store_app_id").length
227
MongoDB Enterprise >
```

Summary: This query is to retrieve the total number of apps we have in our app store dataset. Here we are finding out the total number of distinct apps using distinct on field “store_app_id” and then using length to get the count of total apps.

2. **To find the overall number of reviews**

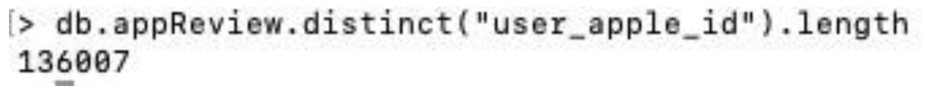
db.appReview.distinct("review_id").length



```
> db.appReview.distinct("review_id").length
138349
```

Summary: Here we are finding the total number of reviews in the system.

3. **To find the number of reviewers** db.appReview.distinct("user_apple_id").length



```
> db.appReview.distinct("user_apple_id").length
136007
```

Summary: Here we are finding the number of distinct reviewers who have reviewed in the system.

4. **To find the number of reviews per category wise**

We merged the Review and Rankings collection to get Reviews and Ranks in one collection:

a) db.appReview.aggregate([{$lookup: { from: "appRanking",localField: "store_app_id",foreignField: "store_app_id", as:"docs"}},{$replaceRoot: { newRoot: { $mergeObjects: [{ $arrayElemAt: ["$docs", 0] }, "$$ROOT"]}}},{$project:{docs: 0}}]).pretty()

b) db.iosAppData.aggregate(

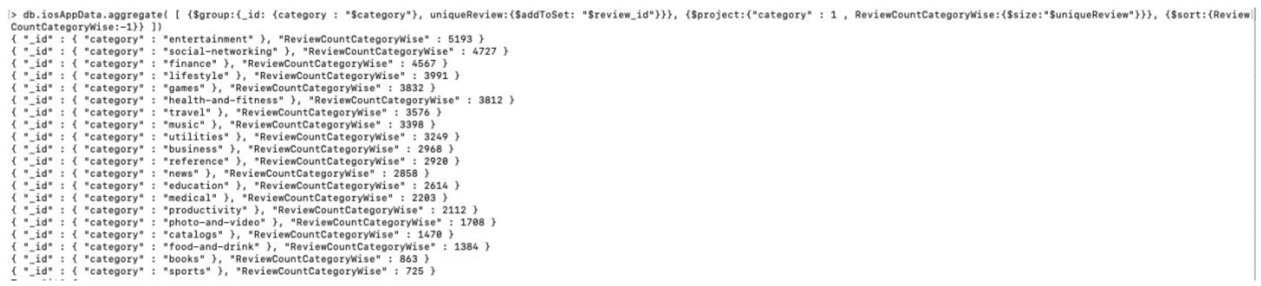
[

{$group:{_id: {category : "$category"}, uniqueReview:{$addToSet: "$review_id"}}},

{$project:{"category" : 1 , ReviewCountCategoryWise:{$size:"$uniqueReview"}}},

{$sort:{ReviewCountCategoryWise:-1}}

])

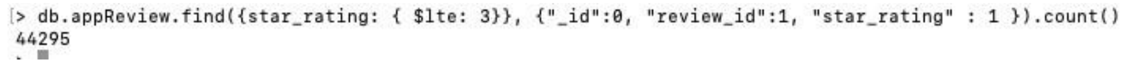


```
> db.iosAppData.aggregate( [ {$group:{_id: {category : "$category"}, uniqueReview:{$addToSet: "$review_id"}}}, {$project:{"category" : 1 , ReviewCountCategoryWise:{$size:"$uniqueReview"}}}, {$sort:{Review
CountCategoryWise:-1}} ])
{ "_id" : { "category" : "entertainment" }, "ReviewCountCategoryWise" : 5193 }
{ "_id" : { "category" : "social-networking" }, "ReviewCountCategoryWise" : 4727 }
{ "_id" : { "category" : "finance" }, "ReviewCountCategoryWise" : 4567 }
{ "_id" : { "category" : "lifestyle" }, "ReviewCountCategoryWise" : 3991 }
{ "_id" : { "category" : "games" }, "ReviewCountCategoryWise" : 3832 }
{ "_id" : { "category" : "health-and-fitness" }, "ReviewCountCategoryWise" : 3812 }
{ "_id" : { "category" : "travel" }, "ReviewCountCategoryWise" : 3576 }
{ "_id" : { "category" : "music" }, "ReviewCountCategoryWise" : 3398 }
{ "_id" : { "category" : "utilities" }, "ReviewCountCategoryWise" : 3249 }
{ "_id" : { "category" : "business" }, "ReviewCountCategoryWise" : 2968 }
{ "_id" : { "category" : "reference" }, "ReviewCountCategoryWise" : 2920 }
{ "_id" : { "category" : "news" }, "ReviewCountCategoryWise" : 2858 }
{ "_id" : { "category" : "education" }, "ReviewCountCategoryWise" : 2614 }
{ "_id" : { "category" : "medical" }, "ReviewCountCategoryWise" : 2203 }
{ "_id" : { "category" : "productivity" }, "ReviewCountCategoryWise" : 2112 }
{ "_id" : { "category" : "photo-and-video" }, "ReviewCountCategoryWise" : 1708 }
{ "_id" : { "category" : "catalogs" }, "ReviewCountCategoryWise" : 1470 }
{ "_id" : { "category" : "food-and-drink" }, "ReviewCountCategoryWise" : 1384 }
{ "_id" : { "category" : "books" }, "ReviewCountCategoryWise" : 863 }
{ "_id" : { "category" : "sports" }, "ReviewCountCategoryWise" : 725 }
```

Summary: We had two collections named appReview and appRanking.For the purpose of this query we have combined both the collection using a) query where lookup method is used with mergeObject method, matching the store_app_id from both the collection.

In query b) we are finding the number of reviews per category in an iOS app store. We are grouping the review's based on the category and calculating the size of the array in which we have pushed all the review_id based on the category, then we are sorting the list with the descending order of the ReviewCoutCategorywise.

5. **The overall number of reviews with rating less than 3** db.appReview.find({star_rating: { $lte: 3}}, {"_id":0, "review_id":1, "star_rating" : 1 }).count() output - 44295

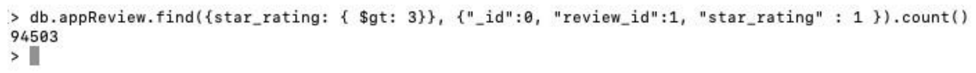


```
> db.appReview.find({star_rating: { $lte: 3}}, {"_id":0, "review_id":1, "star_rating" : 1 }).count()
44295
```

Summary : Here we are trying to get the total number of reviews that have ratings less than or equal to 3 so as to know how many reviews we have towards the bad side. As per the outcome, out of total 138349 we have total 44295 reviews which are more on the negative side.

6. **The overall number of reviews with rating more than 3**

db.appReview.find({star_rating: { $gt: 3}}, {"_id":0, "review_id":1, "star_rating" : 1 }).count()
94503



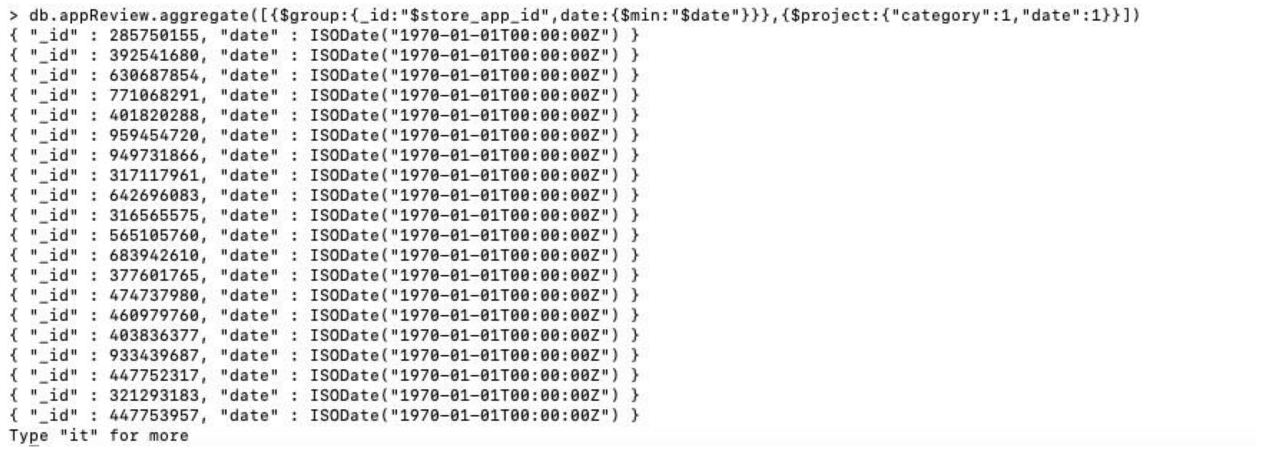
```
> db.appReview.find({star_rating: { $gt: 3}}, {"_id":0, "review_id":1, "star_rating" : 1 }).count()
94503
>
```

Summary : Here we are querying to retrieve the total number of reviews that have ratings more than 3, and to know how many reviews we have towards the good side. As per the outcome, out of total 138349 we have total 94503 reviews which are more on the positive side. Based on the above two queries we can see that we have more positive reviews than negative reviews.

7. **The date of the first review per app**

a) db.appReview.find().forEach(function(x){x.date=new Date(x.date);db.appReview.save(x)})

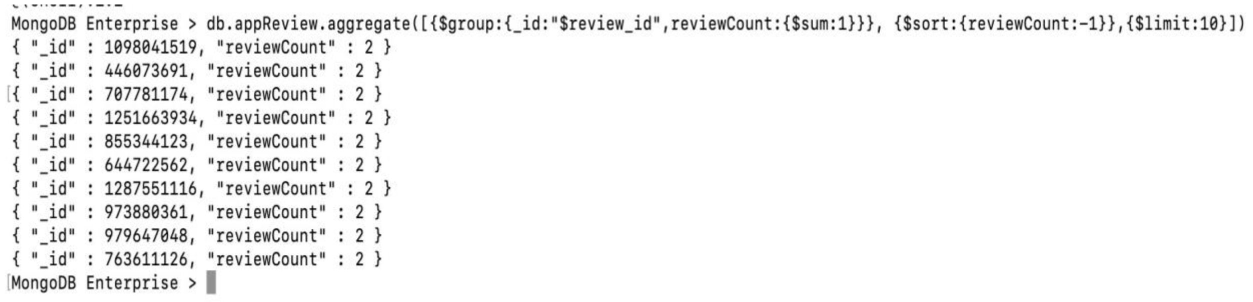
b) db.appReview.aggregate([{$group:{_id:"$store_app_id",date:{$min:"$date"}}},{$project:{"category":1,"date":1}}])



```
> db.appReview.aggregate([{$group:{_id:"$store_app_id",date:{$min:"$date"}}},{$project:{"category":1,"date":1}}])
{ "_id" : 285750155, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 392541680, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 630687854, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 771068291, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 401820288, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 959454720, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 949731866, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 317117961, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 642696083, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 316565575, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 565105760, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 683942610, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 377601765, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 474737980, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 460979760, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 403836377, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 933439687, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 447752317, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 321293183, "date" : ISODate("1970-01-01T00:00:00Z") }
{ "_id" : 447753957, "date" : ISODate("1970-01-01T00:00:00Z") }
Type "it" for more
```

Summary: In query a) we are formatting the data field, by changing the string data type to date time datatype.
In query b) we are grouping store_app_id and taking the first date on which an app was reviewed by applying the min method on the date field.

8. **The top 10 most prolific reviewers**
db.appReview.aggregate([{$group:{_id:"$review_id",reviewCount:{$sum:1}}},
{$sort:{reviewCount:-1}},{$limit:10}])

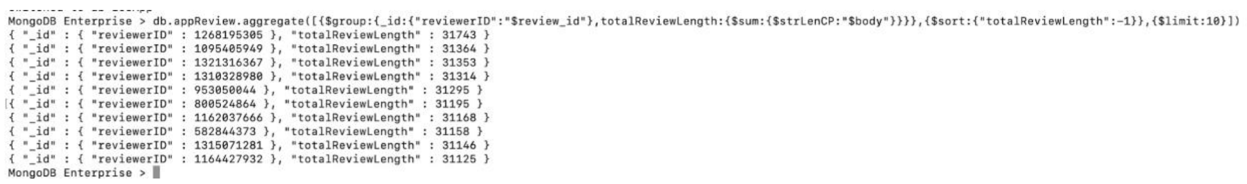


```
MongoDB Enterprise > db.appReview.aggregate([{$group:{_id:"$review_id",reviewCount:{$sum:1}}}, {$sort:{reviewCount:-1}},{$limit:10}])
{ "_id" : 1098041519, "reviewCount" : 2 }
{ "_id" : 446073691, "reviewCount" : 2 }
{ "_id" : 707781174, "reviewCount" : 2 }
{ "_id" : 1251663934, "reviewCount" : 2 }
{ "_id" : 855344123, "reviewCount" : 2 }
{ "_id" : 644722562, "reviewCount" : 2 }
{ "_id" : 1287551116, "reviewCount" : 2 }
{ "_id" : 973880361, "reviewCount" : 2 }
{ "_id" : 979647048, "reviewCount" : 2 }
{ "_id" : 763611126, "reviewCount" : 2 }
MongoDB Enterprise >
```

Summary : Here we are finding the list of reviewers who gives more number of reviews. In the query we are using aggregate to group on "review_id" and get the count of the number of reviews each reviewer has given and then we are sorting it on count of reviews and limiting it to 10 so as to get the list of top 10 most prolific reviewers.

9. **The top 10 most verbose reviewers**

To convert into string- db.appReview.find({body:{$exists:true}}).forEach(function(x){ db.appReview.update({_id:x._id},{$set:{body:x.body.toString()}})});
db.appReview.aggregate([{$group:{_id:{"reviewerID":"$review_id"},
totalReviewLength:{$sum:{$strLenCP:"$body"}}}},
{$sort:{"totalReviewLength":-1}},{$limit:10}])

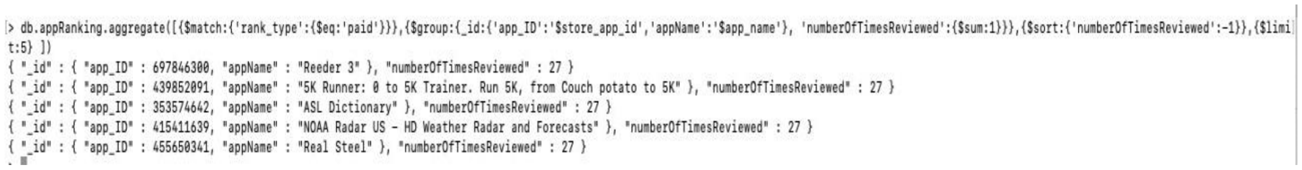


```
MongoDB Enterprise > db.appReview.aggregate([{$group:{_id:{"reviewerID":"$review_id"},totalReviewLength:{$sum:{$strLenCP:"$body"}}}},{$sort:{"totalReviewLength":-1}},{$limit:10}])
{ "_id" : { "reviewerID" : 1268195305 }, "totalReviewLength" : 31743 }
{ "_id" : { "reviewerID" : 1095405949 }, "totalReviewLength" : 31364 }
{ "_id" : { "reviewerID" : 1321316367 }, "totalReviewLength" : 31353 }
{ "_id" : { "reviewerID" : 1310328980 }, "totalReviewLength" : 31314 }
{ "_id" : { "reviewerID" : 953050044 }, "totalReviewLength" : 31295 }
{ "_id" : { "reviewerID" : 800524864 }, "totalReviewLength" : 31195 }
{ "_id" : { "reviewerID" : 1162037666 }, "totalReviewLength" : 31168 }
{ "_id" : { "reviewerID" : 582844373 }, "totalReviewLength" : 31158 }
{ "_id" : { "reviewerID" : 1315071281 }, "totalReviewLength" : 31146 }
{ "_id" : { "reviewerID" : 1164427932 }, "totalReviewLength" : 31125 }
MongoDB Enterprise >
```

Summary : In this query we are trying to find the top 10 most verbose reviewers. To get this we are using $strLenCP which will give us the length of the reviews given by reviewers. We used first query to convert the field body to string so that we can use $strLenCP. In the second query we are finding the length of reviews and listing out Reviewers with most length count.

10. **Top 5 paid apps**

db.appRanking.aggregate([{$match:{'rank_type':{$eq:'paid'}}},{$group:{_id:{'app_ID':'$store_app_id','appName':'$app_name'},
'numberOfTimesReviewed':{$sum:1}}},{$sort:{'numberOfTimesReviewed':-1}},{$limit:5}])

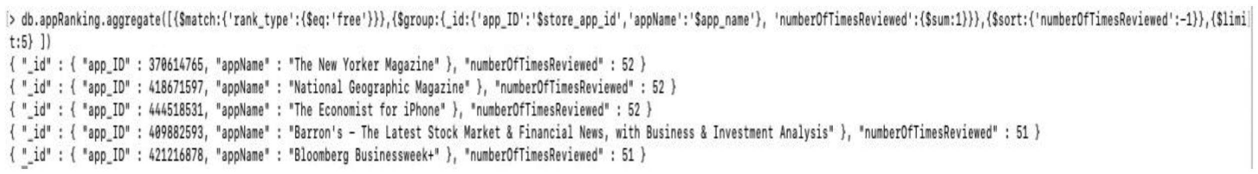


```
> db.appRanking.aggregate([{$match:{'rank_type':{$eq:'paid'}}},{$group:{_id:{'app_ID':'$store_app_id','appName':'$app_name'}, 'numberOfTimesReviewed':{$sum:1}}},{$sort:{'numberOfTimesReviewed':-1}},{$limi
t:5} ])
{ "_id" : { "app_ID" : 697846300, "appName" : "Reeder 3" }, "numberOfTimesReviewed" : 27 }
{ "_id" : { "app_ID" : 439852091, "appName" : "5K Runner: 0 to 5K Trainer. Run 5K, from Couch potato to 5K" }, "numberOfTimesReviewed" : 27 }
{ "_id" : { "app_ID" : 353574642, "appName" : "ASL Dictionary" }, "numberOfTimesReviewed" : 27 }
{ "_id" : { "app_ID" : 415411639, "appName" : "NOAA Radar US - HD Weather Radar and Forecasts" }, "numberOfTimesReviewed" : 27 }
{ "_id" : { "app_ID" : 455650341, "appName" : "Real Steel" }, "numberOfTimesReviewed" : 27 }
```

Summary: In this query we are finding out the top 5 apps which are paid and have been reviewed most. We are counting the number of reviews for an app to check the count of the download of the app. Here we are finding the match where rank_type is paid and grouped the document based on the store_app_id and app_name.Then counted the number of reviews on a particular app and sorted based on the count in descending order and displayed top 5.

11. **Top 5 free app**

db.appRanking.aggregate([{$match:{'rank_type':{$eq:'free'}}},{$group:{_id:{'app_ID':'$store_app_id','appName':'$app_name'},
'numberOfTimesReviewed':{$sum:1}}},{$sort:{'numberOfTimesReviewed':-1}},{$limit:5}])

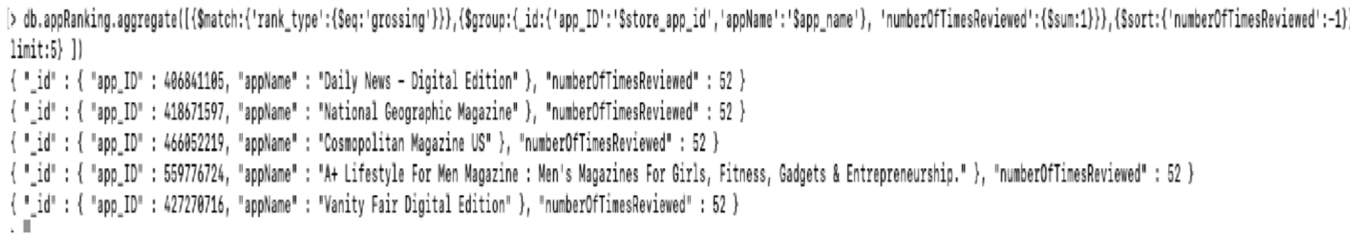


```
> db.appRanking.aggregate([{$match:{'rank_type':{$eq:'free'}}},{$group:{_id:{'app_ID':'$store_app_id','appName':'$app_name'}, 'numberOfTimesReviewed':{$sum:1}}},{$sort:{'numberOfTimesReviewed':-1}},{$limi
t:5} ])
{ "_id" : { "app_ID" : 370614765, "appName" : "The New Yorker Magazine" }, "numberOfTimesReviewed" : 52 }
{ "_id" : { "app_ID" : 418671597, "appName" : "National Geographic Magazine" }, "numberOfTimesReviewed" : 52 }
{ "_id" : { "app_ID" : 444518531, "appName" : "The Economist for iPhone" }, "numberOfTimesReviewed" : 52 }
{ "_id" : { "app_ID" : 409882593, "appName" : "Barron's - The Latest Stock Market & Financial News, with Business & Investment Analysis" }, "numberOfTimesReviewed" : 51 }
{ "_id" : { "app_ID" : 421216878, "appName" : "Bloomberg Businessweek+" }, "numberOfTimesReviewed" : 51 }
```

Summary: In this query we are finding out the top 5 apps which are free and have been reviewed most. We are counting the number of reviews for an app to check the count of the download of the app. Here we are finding the match where rank_type is free and grouped the document based on the store_app_id and app_name.Then counted the number of reviews on a particular app and sorted based on the count in descending order and displayed top 5.

12. Top 5 grossing app

db.appRanking.aggregate([{$match:{'rank_type':{$eq:'grossing'}}},{$group:{_id:{'app_ID':'$store_app_id','appName':'$app_name'}, 'numberOfTimesReviewed':{$sum:1}}},{$sort:{'numberOfTimesReviewed':-1}},{$limit:5}])



```
> db.appRanking.aggregate([{$match:{'rank_type':{$eq:'grossing'}}},{$group:{_id:{'app_ID':'$store_app_id','appName':'$app_name'}, 'numberOfTimesReviewed':{$sum:1}}},{$sort:{'numberOfTimesReviewed':-1}
limit:5} ])
{ "_id" : { "app_ID" : 406841105, "appName" : "Daily News - Digital Edition" }, "numberOfTimesReviewed" : 52 }
{ "_id" : { "app_ID" : 418671597, "appName" : "National Geographic Magazine" }, "numberOfTimesReviewed" : 52 }
{ "_id" : { "app_ID" : 466052219, "appName" : "Cosmopolitan Magazine US" }, "numberOfTimesReviewed" : 52 }
{ "_id" : { "app_ID" : 559776724, "appName" : "A+ Lifestyle For Men Magazine : Men's Magazines For Girls, Fitness, Gadgets & Entrepreneurship." }, "numberOfTimesReviewed" : 52 }
{ "_id" : { "app_ID" : 427270716, "appName" : "Vanity Fair Digital Edition" }, "numberOfTimesReviewed" : 52 }
```

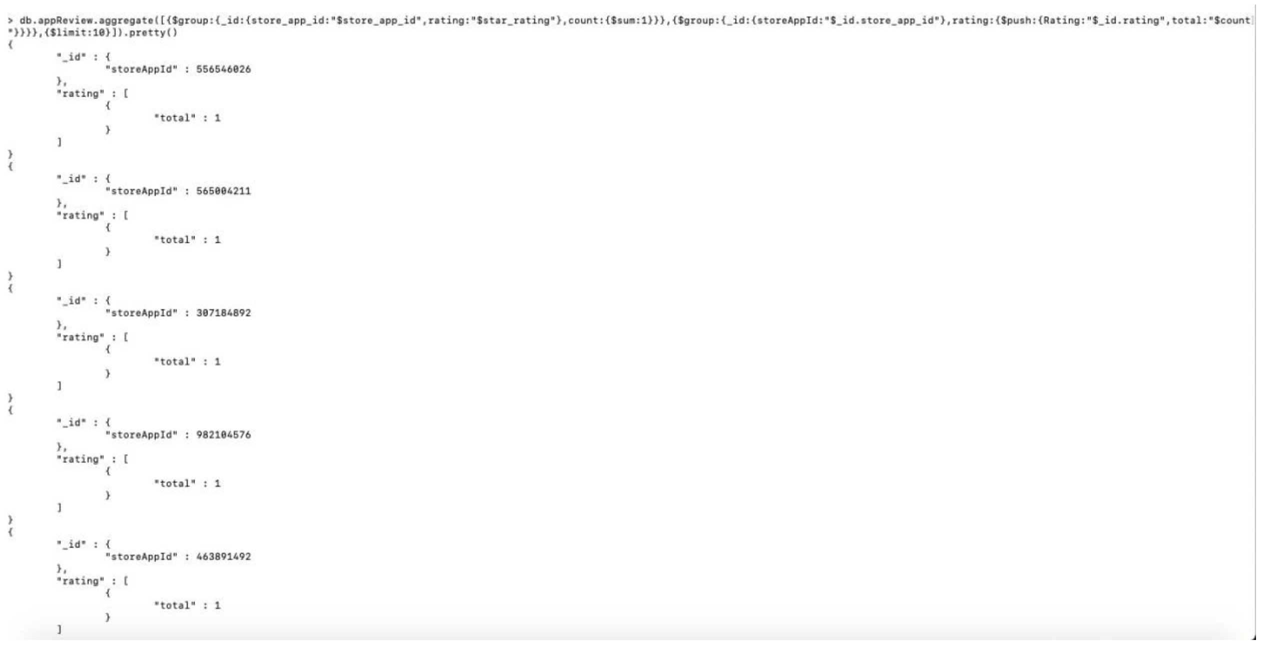
Summary: In this query we are finding out the top 5 apps which are grossing and have been reviewed most. We are counting the number of reviews for an app to check the count of the download of the app. Here we are finding the match where rank_type is grossing and grouped the document based on the store_app_id and app_name.Then counted the number of reviews on a particular app and sorted based on the count in descending order and displayed top 5.

PART C

Below are the MongoDB queries based on the useful analytical performance metrics. The queries retrieve valuable key information on the iOS apps.

MongoDB Analytics:

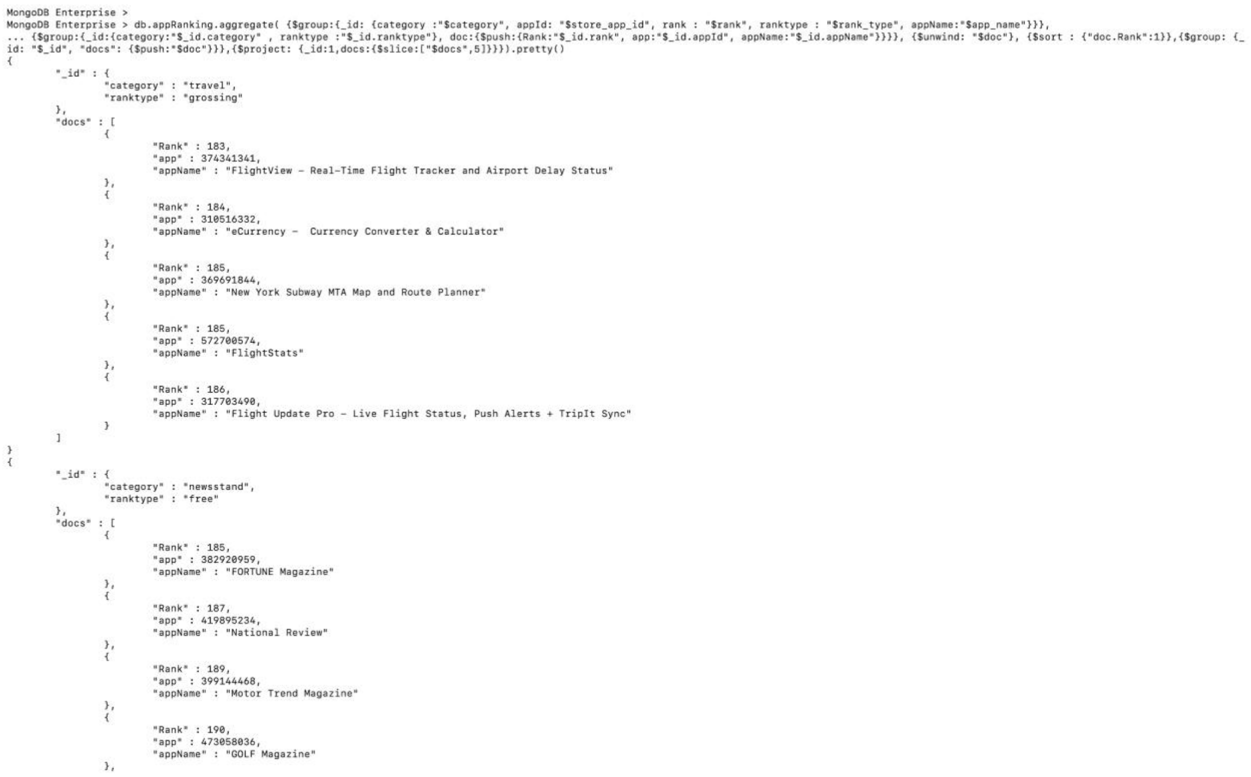
1. **Histogram of review ratings(5's, 4's, 3's, 2's and 1's)per app–display the results for 10 app only.**
db.appReview.aggregate([{$group:{_id:{store_app_id:"$store_app_id",rating:"$star_rating"},count:{$sum:1}}},{$group:{_id:{storeAppId:"$_id.store_app_id"},rating:{$push:{Rating:"$_id.rating",total:"$count"}}}},{$limit:10}]).pretty()



```
> db.appReview.aggregate([{$group:{_id:{store_app_id:"$store_app_id",rating:"$star_rating"},count:{$sum:1}}},{$group:{_id:{storeAppId:"$_id.store_app_id"},rating:{$push:{Rating:"$_id.rating",total:"$count
"}}}},{$limit:10}]).pretty()
{
        "_id" : {
                "storeAppId" : 556546026
        },
        "rating" : [
                {
                        "total" : 1
                }
        ]
}
{
        "_id" : {
                "storeAppId" : 565004211
        },
        "rating" : [
                {
                        "total" : 1
                }
        ]
}
{
        "_id" : {
                "storeAppId" : 307184892
        },
        "rating" : [
                {
                        "total" : 1
                }
        ]
}
{
        "_id" : {
                "storeAppId" : 982104576
        },
        "rating" : [
                {
                        "total" : 1
                }
        ]
}
{
        "_id" : {
                "storeAppId" : 463891492
        },
        "rating" : [
                {
                        "total" : 1
                }
        ]
```

Summary: We are considering this query for showing the count of each rating from 1-5 for a particular app. We are doing this because we want to give the user how many star reviews have been given by the other reviewers for the same app.
In this query we are grouping the store_app_id and star_rating and pushed the star_rating field and counted each of the rating separately. The visualization for this query is in the form of a histogram.

2. **Finding top trending apps within each category and rank type** db.appRanking.aggregate({$group:{_id: {category :"$category", appId: "$store_app_id", rank : "$rank", ranktype : "$rank_type", appName:"$app_name"}}}, {$group:{_id:{category:"$_id.category" , ranktype :"$_id.ranktype"}, doc:{$push:{Rank:"$_id.rank", app:"$_id.appId", appName:"$_id.appName"}}}}, {$unwind: "$doc"}, {$sort : {"doc.Rank":1}},{$group: {_id: "$_id", "docs": {$push:"$doc"}}},{$project: {_id:1,docs:{$slice:["$docs",5]}}}}).pretty()



```
MongoDB Enterprise >
MongoDB Enterprise > db.appRanking.aggregate( {$group:{_id: {category :"$category", appId: "$store_app_id", rank : "$rank", ranktype : "$rank_type", appName:"$app_name"}}},
... {$group:{_id:{category:"$_id.category" , ranktype :"$_id.ranktype"}, doc:{$push:{Rank:"$_id.rank", app:"$_id.appId", appName:"$_id.appName"}}}}, {$unwind: "$doc"}, {$sort : {"doc.Rank":1}},{$group: {_
id: "$_id", "docs": {$push:"$doc"}}},{$project: {_id:1,docs:{$slice:["$docs",5]}}}).pretty()
{
        "_id" : {
                "category" : "travel",
                "ranktype" : "grossing"
        },
        "docs" : [
                {
                        "Rank" : 183,
                        "app" : 374341341,
                        "appName" : "FlightView - Real-Time Flight Tracker and Airport Delay Status"
                },
                {
                        "Rank" : 184,
                        "app" : 310516332,
                        "appName" : "eCurrency -  Currency Converter & Calculator"
                },
                {
                        "Rank" : 185,
                        "app" : 369691844,
                        "appName" : "New York Subway MTA Map and Route Planner"
                },
                {
                        "Rank" : 185,
                        "app" : 572700574,
                        "appName" : "FlightStats"
                },
                {
                        "Rank" : 186,
                        "app" : 317703490,
                        "appName" : "Flight Update Pro - Live Flight Status, Push Alerts + TripIt Sync"
                }
        ]
}
{
        "_id" : {
                "category" : "newsstand",
                "ranktype" : "free"
        },
        "docs" : [
                {
                        "Rank" : 185,
                        "app" : 382920959,
                        "appName" : "FORTUNE Magazine"
                },
                {
                        "Rank" : 187,
                        "app" : 419895234,
                        "appName" : "National Review"
                },
                {
                        "Rank" : 189,
                        "app" : 399144468,
                        "appName" : "Motor Trend Magazine"
                },
                {
                        "Rank" : 190,
                        "app" : 473058036,
                        "appName" : "GOLF Magazine"
                },
```

Summary : We designed this query in order to provide users the list of top apps based on their category and rank type (free, grossing, paid). This query will help users to see the list of top free, top grossing and top paid apps for the category of the app they are on. This will help users to navigate to the best app within same category as per the rank type. In this query we are using aggregate and applying group on “category”, “store_app_id”, “rank”, “ranktype”, “app_name” then we pushed rank, appid and appName in a doc. We used unwind to sort on rank and the used group again on doc.

3. **Recommended Apps for Users - List of top ranked apps within the category for users who reviewed most apps for that category**

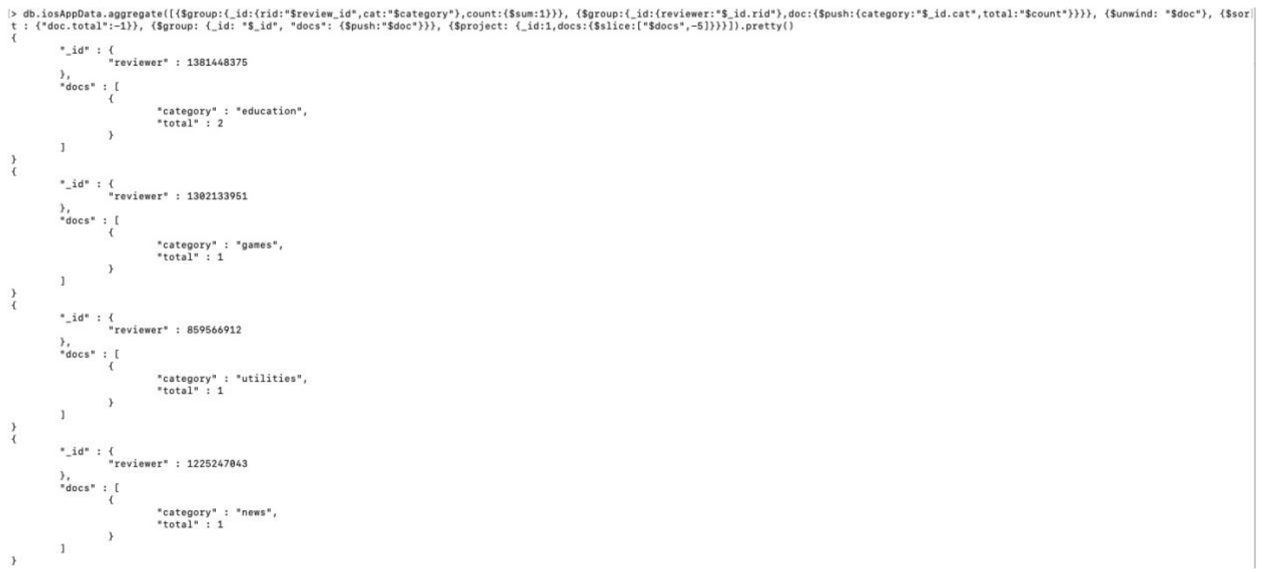
1. db.iosAppData.aggregate([{$group:{_id:{rid:"$review_id",cat:"$category"},count:{$sum: 1}}}, {$group:{_id:{reviewer:"$_id.rid"},doc:{$push:{category:"$_id.cat",total:"$count"}}}},

{$unwind: "$doc"}, {$sort : {"doc.total":-1}},

{$group: {_id: "$_id", "docs": {$push:"$doc"}}},

{$project: {_id:1,docs:{$slice:["$docs",5]}}}]).pretty()

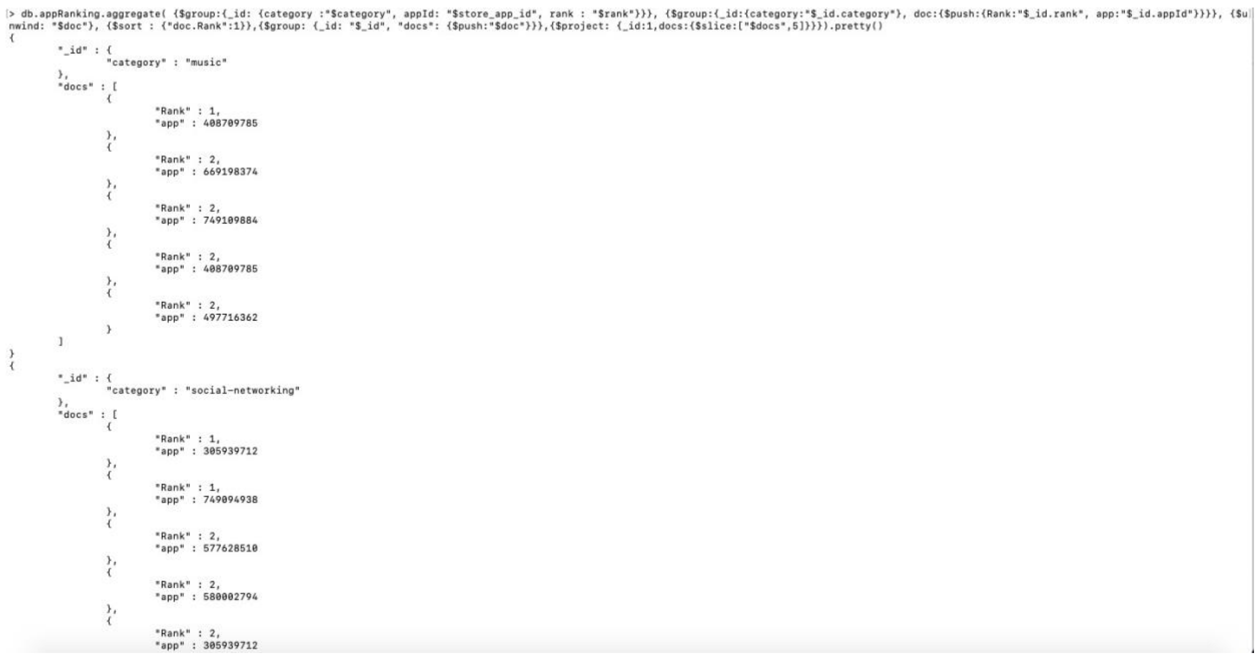
2. db.appRanking.aggregate({$group:{_id: {category :"$category", appId: "$store_app_id", rank : "$rank"}}}, {$group:{_id:{category:"$_id.category"}, doc:{$push:{Rank:"$_id.rank", app:"$_id.appId"}}}}, {$unwind: "$doc"}, {$sort : {"doc.Rank":1}},{$group: {_id: "$_id", "docs": {$push:"$doc"}}},{$project: {_id:1,docs:{$slice:["$docs",5]}}}).pretty()



```
> db.iosAppData.aggregate([{$group:{_id:{rid:"$review_id",cat:"$category"},count:{$sum:1}}}, {$group:{_id:{reviewer:"$_id.rid"},doc:{$push:{category:"$_id.cat",total:"$count"}}}}, {$unwind: "$doc"}, {$sor
t : {"doc.total":-1}}, {$group: {_id: "$_id", "docs": {$push:"$doc"}}}, {$project: {_id:1,docs:{$slice:["$docs",-5]}}}]).pretty()
{
        "_id" : {
                "reviewer" : 1381448375
        },
        "docs" : [
                {
                        "category" : "education",
                        "total" : 2
                }
        ]
}
{
        "_id" : {
                "reviewer" : 1302133951
        },
        "docs" : [
                {
                        "category" : "games",
                        "total" : 1
                }
        ]
}
{
        "_id" : {
                "reviewer" : 859566912
        },
        "docs" : [
                {
                        "category" : "utilities",
                        "total" : 1
                }
        ]
}
{
        "_id" : {
                "reviewer" : 1225247043
        },
        "docs" : [
                {
                        "category" : "news",
                        "total" : 1
                }
        ]
}
```

Summary : We designed the above two queries to display the apps for the users which we are recommending for them based on the apps that they have reviewed. As per our recommendation model we are taking the category for which the user has given the most number of reviews and based on that category we are recommending them the list of top ranked apps within the same category. As per our definition for recommended apps we are checking in which category the user is interested in and this count we are taking based on the categories that he had reviewed and the category which is most reviewed is taken up and we are using the second query mentioned above to display the list of top ranked apps for the same category. We are using this metrics to display the recommended apps for the user in wireframe of app. In the first query, we are trying to find out the category in which the user is interested by using group on “review_id” and “category” and then creating a document for calculating the total number of reviews given by user for that category. After getting the category output from the first query we will use the second query to generate the top ranked apps for that category so that we can recommend the top 5 apps for that category in the wireframe of the app that we designed.

4. **List of top ranking apps with same category** db.appRanking.aggregate({$group:{_id: {category :"$category", appId: "$store_app_id", rank : "$rank"}}}, {$group:{_id:{category:"$_id.category"}, doc:{$push:{Rank:"$_id.rank", app:"$_id.appId"}}}}, {$unwind: "$doc"}, {$sort : {"doc.Rank":1}},{$group: {_id: "$_id", "docs": {$push:"$doc"}}},{$project: {_id:1,docs:{$slice:["$docs",5]}}}}).pretty()



```
> db.appRanking.aggregate( {$group:{_id: {category :"$category", appId: "$store_app_id", rank : "$rank"}}}, {$group:{_id:{category:"$_id.category"}, doc:{$push:{Rank:"$_id.rank", app:"$_id.appId"}}}}, {$u
nwind: "$doc"}, {$sort : {"doc.Rank":1}},{$group: {_id: "$_id", "docs": {$push:"$doc"}}},{$project: {_id:1,docs:{$slice:["$docs",5]}}}).pretty()
{
        "_id" : {
                "category" : "music"
        },
        "docs" : [
                {
                        "Rank" : 1,
                        "app" : 408709785
                },
                {
                        "Rank" : 2,
                        "app" : 669198374
                },
                {
                        "Rank" : 2,
                        "app" : 749109884
                },
                {
                        "Rank" : 2,
                        "app" : 408709785
                },
                {
                        "Rank" : 2,
                        "app" : 497716362
                }
        ]
}
{
        "_id" : {
                "category" : "social-networking"
        },
        "docs" : [
                {
                        "Rank" : 1,
                        "app" : 305939712
                },
                {
                        "Rank" : 1,
                        "app" : 749094938
                },
                {
                        "Rank" : 2,
                        "app" : 577628510
                },
                {
                        "Rank" : 2,
                        "app" : 580002794
                },
                {
                        "Rank" : 2,
                        "app" : 305939712
```

Summary : We interpreted this query to display top ranking apps within a category when a user navigates to a particular app page. Here we are grouping the document using the category, app and it's ranking. We are pushing the documents which contain app name and corresponding ranking to an array to display it in the app page as shown in the wireframe design. At the end we are sorting on the ranking in ascending order and slicing it to show the first 5 ranked apps.

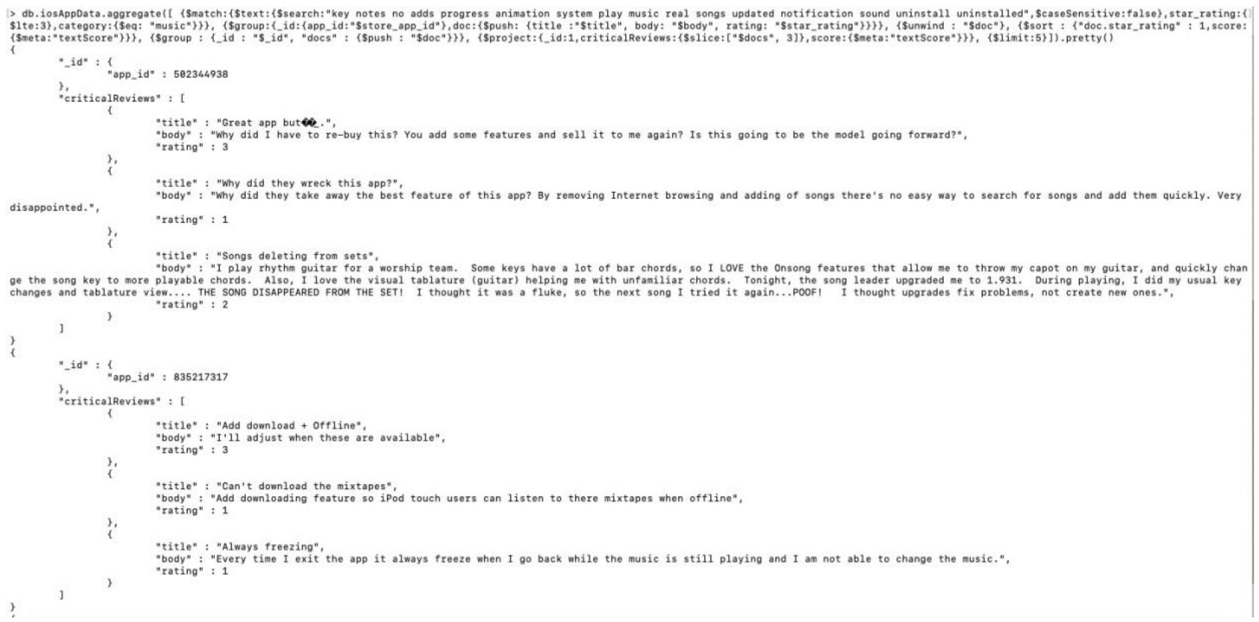
5. **Top Critical Reviews per app (reviews which technically critique the app)**

To find the critical reviews for an app we are taking into account the technical words in each category, as critical reviews usually technically critique the app, also critical reviewers do not generally give very high ratings to the app. So for our analysis we have chosen one category (music) for demo purpose and have created a technical library for it and also we have filtered the reviews which have rating equal to or less than 3.

```
db.iosAppData.createIndex({"body":'text',"title":'text'})

db.iosAppData.aggregate([

{$match:{$text:{$search:"key notes no adds progress animation system play music real songs updated notification sound uninstall uninstalled",$caseSensitive:false},star_rating:{$lte:3},category:{$eq: "music"}}},

{$group:{_id:{app_id:"$store_app_id"},doc:{$push: {title :"$title", body: "$body", rating: "$star_rating"}}}},

{$unwind : "$doc"},

{$sort : {"doc.star_rating" : 1,score:{$meta:"textScore"}}},

{$group : {_id : "$_id", "docs" : {$push : "$doc"}}},
```

{$project:{_id:1,criticalReviews:{$slice:["$docs", 3]},score:{$meta:"textScore"}}},

{$limit:5}]).pretty()



```
> db.iosAppData.aggregate([ {$match:{$text:{$search:"key notes no adds progress animation system play music real songs updated notification sound uninstall uninstalled",$caseSensitive:false},star_rating:{
$lte:3},category:{$eq: "music"}}}, {$group:{_id:{app_id:"$store_app_id"},doc:{$push: {title :"$title", body: "$body", rating: "$star_rating"}}}}, {$unwind : "$doc"}, {$sort : {"doc.star_rating" : 1,score:
{$meta:"textScore"}}}, {$group : {_id : "$_id", "docs" : {$push : "$doc"}}}, {$project:{_id:1,criticalReviews:{$slice:["$docs", 3]},score:{$meta:"textScore"}}}, {$limit:5}]).pretty()
{
        "_id" : {
                "app_id" : 502344938
        },
        "criticalReviews" : [
                {
                        "title" : "Great app but��.",
                        "body" : "Why did I have to re-buy this? You add some features and sell it to me again? Is this going to be the model going forward?",
                        "rating" : 3
                },
                {
                        "title" : "Why did they wreck this app?",
                        "body" : "Why did they take away the best feature of this app? By removing Internet browsing and adding of songs there's no easy way to search for songs and add them quickly. Very
disappointed.",
                        "rating" : 1
                },
                {
                        "title" : "Songs deleting from sets",
                        "body" : "I play rhythm guitar for a worship team.  Some keys have a lot of bar chords, so I LOVE the Onsong features that allow me to throw my capot on my guitar, and quickly chan
ge the song key to more playable chords.  Also, I love the visual tablature (guitar) helping me with unfamiliar chords.  Tonight, the song leader upgraded me to 1.931.  During playing, I did my usual key
changes and tablature view.... THE SONG DISAPPEARED FROM THE SET!  I thought it was a fluke, so the next song I tried it again...POOF!   I thought upgrades fix problems, not create new ones.",
                        "rating" : 2
                }
        ]
}
{
        "_id" : {
                "app_id" : 835217317
        },
        "criticalReviews" : [
                {
                        "title" : "Add download + Offline",
                        "body" : "I'll adjust when these are available",
                        "rating" : 3
                },
                {
                        "title" : "Can't download the mixtapes",
                        "body" : "Add downloading feature so iPod touch users can listen to there mixtapes when offline",
                        "rating" : 1
                },
                {
                        "title" : "Always freezing",
                        "body" : "Every time I exit the app it always freeze when I go back while the music is still playing and I am not able to change the music.",
                        "rating" : 1
                }
        ]
}
```

Summary: Here in this query we are first creating a text index on the 'title' and 'body' field. Then we are doing a text search on the technical library which we have created for the category 'music', then we are filtering the reviews and keeping only those which have a star rating of less than equal to 3. We are then grouping these reviews by store_app_id and then pushing all the reviews (including its detail) in it. Then we are sorting in descending order by the text search score and ascending order of the star rating (as we want the most critiqued review on the top). Displayed result shows the top 3 most critical reviews for 5 apps.

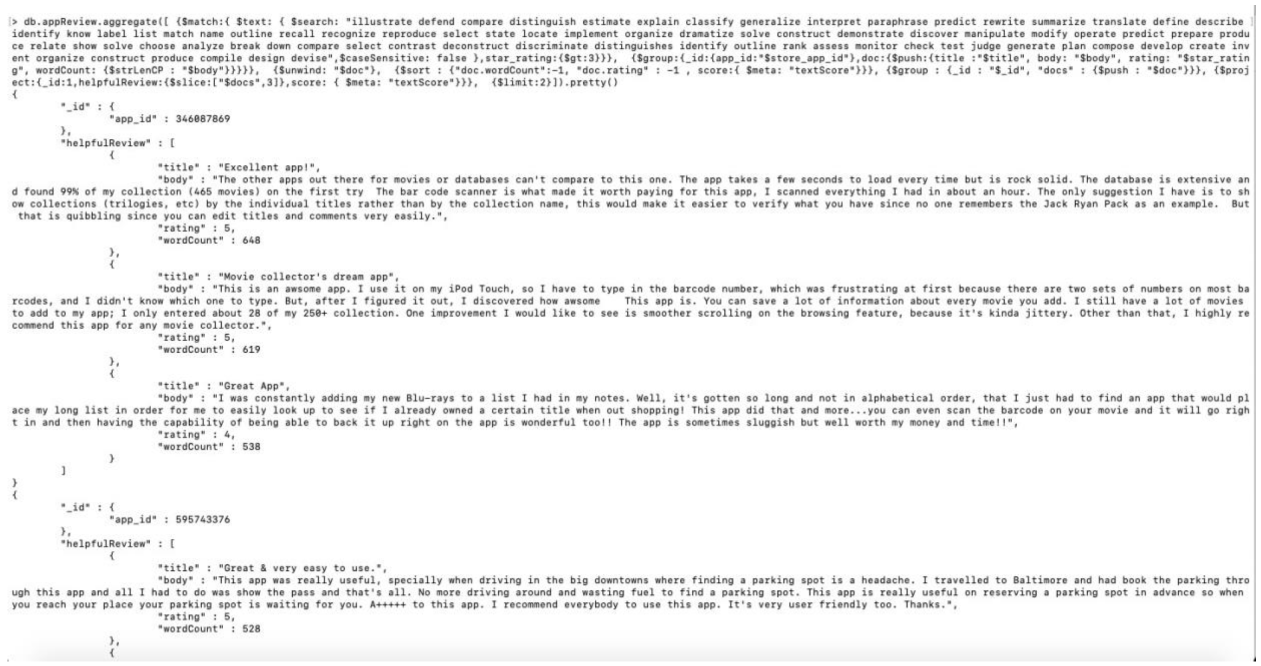
6. **Top 10 most helpful review per app:**

To identify the helpfulness of a review and to find top helpful reviews of an app, we are taking into account the presence of cognitive process words (words that describe attention, perception, higher reasoning), as it increases the possibility of a review being more helpful. People reading these reviews are influenced by the logical flow of the text. Furthermore, the verbosity of a review also affects its helpfulness (the more verbose a review, the more helpful it is perceived). Lastly, we are also considering that a reviewer giving a higher star rating to the app also has a positive effect on increasing the helpfulness of the review (as high star rating review is considered to be more helpful) [1].

db.appReview.createIndex({"body":'text',"title":'text'})

db.appReview.aggregate([
{$match:{ $text: { $search: "illustrate defend compare distinguish estimate explain classify generalize interpret paraphrase predict rewrite summarize translate define describe identify know label list match name outline recall recognize reproduce select state locate implement organize dramatize solve construct demonstrate discover manipulate modify operate predict

prepare produce relate show solve choose analyze break down compare select contrast deconstruct discriminate distinguishes identify outline rank assess monitor check test judge generate plan compose develop create invent organize construct produce compile design devise",$caseSensitive: false },star_rating:{$gt:3}}},
{$group:{_id:{app_id:"$store_app_id"},doc:{$push:{title :"$title", body: "$body", rating:
 "$star_rating", wordCount: {$strLenCP : "$body"}}}}},
{$unwind: "$doc"},
{$sort : {"doc.wordCount":-1, "doc.rating" : -1 , score:{ $meta: "textScore"}}},
{$group : {_id : "$_id", "docs" : {$push : "$doc"}}},
{$project:{_id:1,helpfulReview:{$slice:["$docs",10]},score: { $meta: "textScore"}}},
{$limit:2}]).pretty()



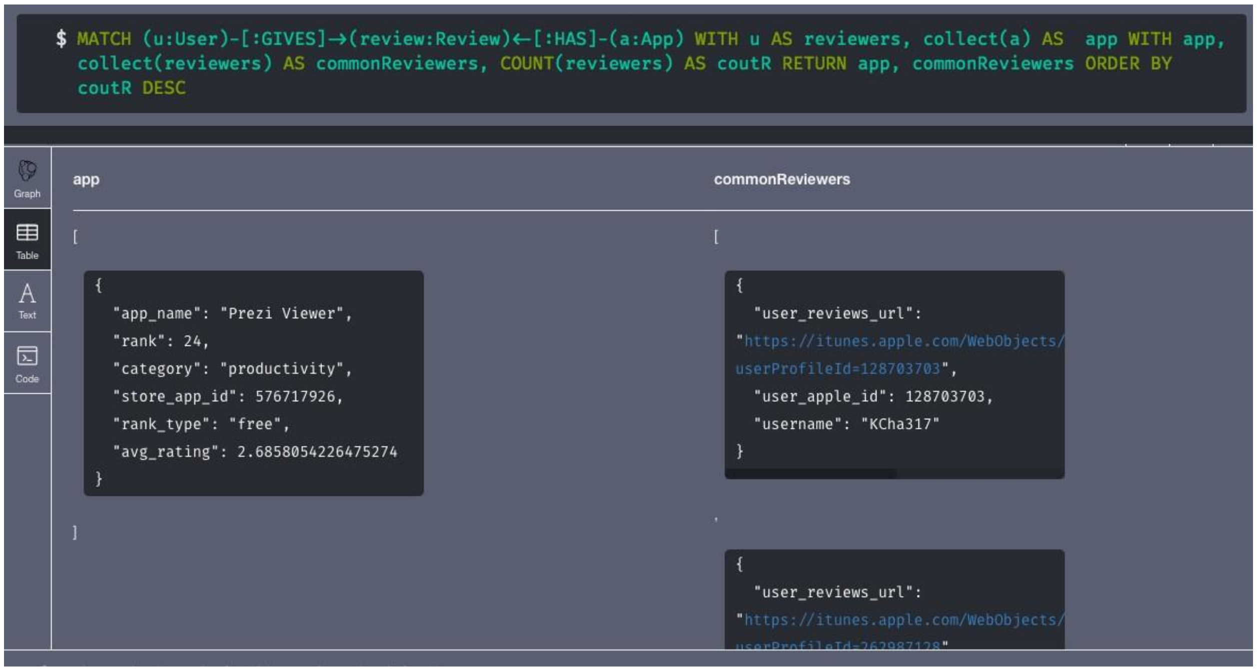
```
> db.appReview.aggregate([ {$match:{ $text: { $search: "illustrate defend compare distinguish estimate explain classify generalize interpret paraphrase predict rewrite summarize translate define describe
identify know label list match name outline recall recognize reproduce select state locate implement organize dramatize solve construct demonstrate discover manipulate modify operate predict prepare produ
ce relate show solve choose analyze break down compare select contrast deconstruct discriminate distinguishes identify outline rank assess monitor check test judge generate plan compose develop create inv
ent organize construct produce compile design devise",$caseSensitive: false },star_rating:{$gt:3}}},  {$group:{_id:{app_id:"$store_app_id"},doc:{$push:{title :"$title", body: "$body", rating: "$star_ratin
g", wordCount: {$strLenCP : "$body"}}}}},  {$unwind: "$doc"},  {$sort : {"doc.wordCount":-1, "doc.rating" : -1 , score:{ $meta: "textScore"}}}, {$group : {_id : "$_id", "docs" : {$push : "$doc"}}}, {$proj
ect:{_id:1,helpfulReview:{$slice:["$docs",3]},score: { $meta: "textScore"}}},  {$limit:2}]).pretty()
{
        "_id" : {
                "app_id" : 346087869
        },
        "helpfulReview" : [
                {
                        "title" : "Excellent app!",
                        "body" : "The other apps out there for movies or databases can't compare to this one. The app takes a few seconds to load every time but is rock solid. The database is extensive an
d found 99% of my collection (465 movies) on the first try  The bar code scanner is what made it worth paying for this app, I scanned everything I had in about an hour. The only suggestion I have is to sh
ow collections (trilogies, etc) by the individual titles rather than by the collection name, this would make it easier to verify what you have since no one remembers the Jack Ryan Pack as an example.  But
 that is quibbling since you can edit titles and comments very easily.",
                        "rating" : 5,
                        "wordCount" : 648
                },
                {
                        "title" : "Movie collector's dream app",
                        "body" : "This is an awsome app. I use it on my iPod Touch, so I have to type in the barcode number, which was frustrating at first because there are two sets of numbers on most ba
rcodes, and I didn't know which one to type. But, after I figured it out, I discovered how awsome    This app is. You can save a lot of information about every movie you add. I still have a lot of movies
to add to my app; I only entered about 28 of my 250+ collection. One improvement I would like to see is smoother scrolling on the browsing feature, because it's kinda jittery. Other than that, I highly re
commend this app for any movie collector.",
                        "rating" : 5,
                        "wordCount" : 619
                },
                {
                        "title" : "Great App",
                        "body" : "I was constantly adding my new Blu-rays to a list I had in my notes. Well, it's gotten so long and not in alphabetical order, that I just had to find an app that would pl
ace my long list in order for me to easily look up to see if I already owned a certain title when out shopping! This app did that and more...you can even scan the barcode on your movie and it will go righ
t in and then having the capability of being able to back it up right on the app is wonderful too!! The app is sometimes sluggish but well worth my money and time!!",
                        "rating" : 4,
                        "wordCount" : 538
                }
        ]
}
{
        "_id" : {
                "app_id" : 595743376
        },
        "helpfulReview" : [
                {
                        "title" : "Great & very easy to use.",
                        "body" : "This app was really useful, specially when driving in the big downtowns where finding a parking spot is a headache. I travelled to Baltimore and had book the parking thro
ugh this app and all I had to do was show the pass and that's all. No more driving around and wasting fuel to find a parking spot. This app is really useful on reserving a parking spot in advance so when
you reach your place your parking spot is waiting for you. A+++++ to this app. I recommend everybody to use this app. It's very user friendly too. Thanks.",
                        "rating" : 5,
                        "wordCount" : 528
                },
                {
```

Summary: In this query, first we are creating a text index on the field 'body' and 'title', then we have taken a sample of cognitive process words and applied text search on the reviews based on these words, then we are also counting the number of words in the review text (verbosity) and filtering only the reviews which have a star rating greater than 3 and then we are grouping by the category and store_app_id and sorting the results in descending order based on the text search score, word count and star rating and limiting the number of reviews to top ten per app.

Neo4j -

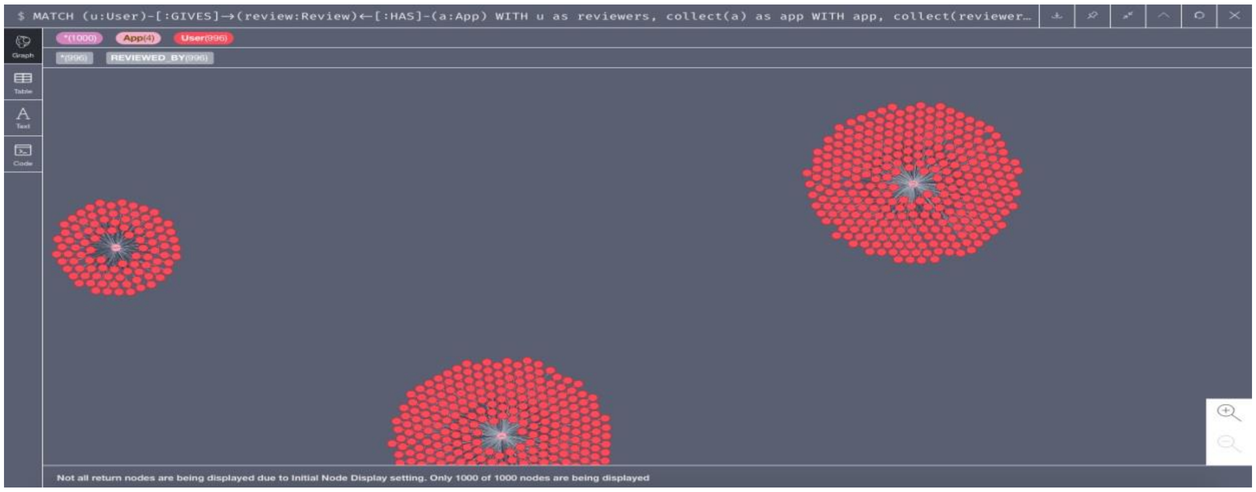
1. **Reviewers sometimes review as a mob (reviewing the common set of apps). Do you find this behavior in this dataset? Run the queries and derive the results.**

   MATCH (u:User)-[:GIVES]->(review:Review)<-[:HAS]-(a:App) WITH u AS reviewers, collect(a) AS app WITH app, collect(reviewers) AS commonReviewers, COUNT(reviewers) AS coutR RETURN app, commonReviewers ORDER BY coutR DESC



```
$ MATCH (u:User)-[:GIVES]→(review:Review)←[:HAS]-(a:App) WITH u AS reviewers, collect(a) AS  app WITH app,
  collect(reviewers) AS commonReviewers, COUNT(reviewers) AS coutR RETURN app, commonReviewers ORDER BY
  coutR DESC
```

```
app                                              commonReviewers
[                                                [
  {                                                {
    "app_name": "Prezi Viewer",                      "user_reviews_url":
    "rank": 24,                                    "https://itunes.apple.com/WebObjects/
    "category": "productivity",                    userProfileId=128703703",
    "store_app_id": 576717926,                       "user_apple_id": 128703703,
    "rank_type": "free",                             "username": "KCha317"
    "avg_rating": 2.6858054226475274               }
  }
                                                 ,
]
                                                   {
                                                     "user_reviews_url":
                                                   "https://itunes.apple.com/WebObjects/
                                                   userProfileId=262987128"
```



```
$ MATCH (u:User)-[:GIVES]→(review:Review)←[:HAS]-(a:App) WITH u as reviewers, collect(a) as app WITH app, collect(reviewer…
```

Not all return nodes are being displayed due to Initial Node Display setting. Only 1000 of 1000 nodes are being displayed

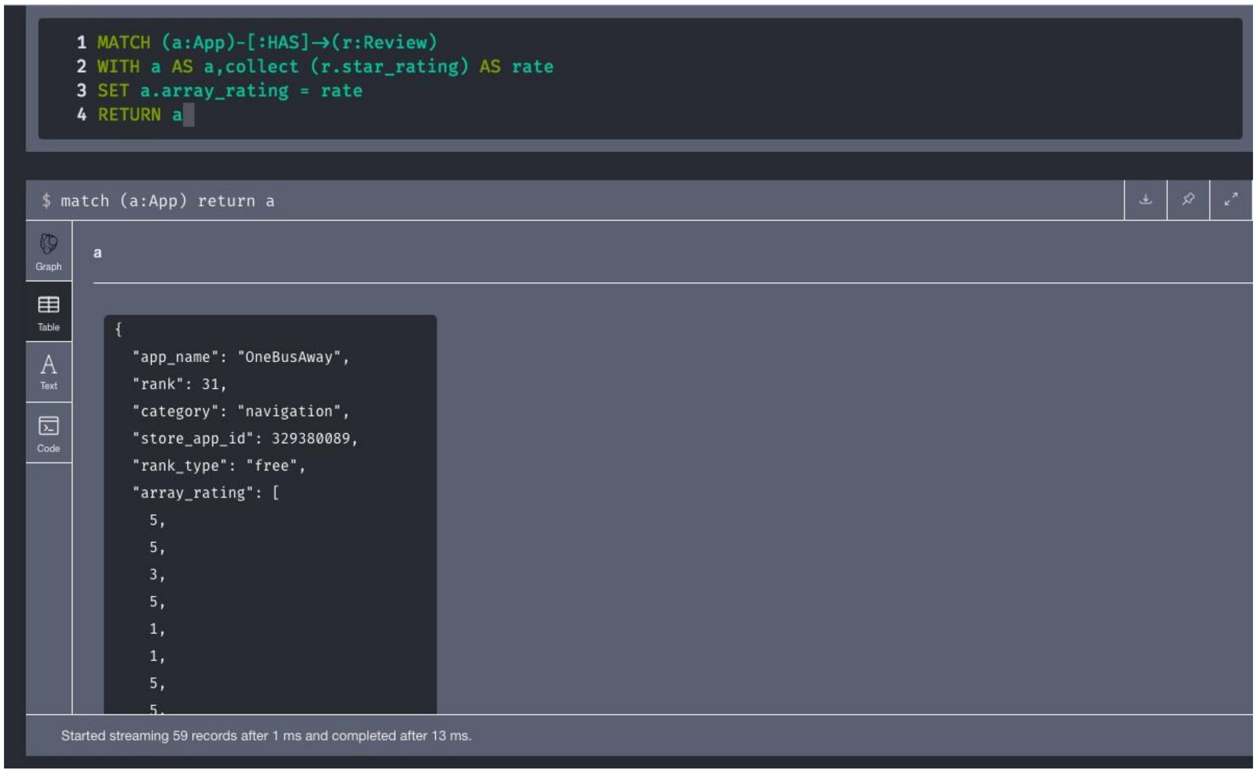
Summary: Since in the table format we are not able to capture the complete array of common apps and reviewers, we can visualize the result in graph form. From the above visualization, we can see that there are common set of reviewer's who are reviewing the same set of products. We are finding out this by finding the reviewers who have the relationship [:GIVES] between the User and the Review node and then again find the relationship [:HAS] with App and Review. By this kind of behavior we are interpreting that this mob reviewing tendency can be related to people deliberately giving fake reviews together for some apps to increase or decrease their popularity.

2. Does similarity in app functionality lead to uniformity in app ratings or divergence? Run the queries and derive the results. Explain your findings.

To check whether the similarity in app functionality is leading to uniformity in app ratings or divergence, we thought of finding the standard deviation of the star rating amongst the apps belonging to each category. Here we are considering that apps with similar app functionality belongs to the same category as we do not have an app description field in our data.

To find out the similarity in an app functionality below are the following steps

a) Set the field array_rating in the app node
MATCH (a:App)-[:HAS]->(r:Review) WITH a AS a,collect (r.star_rating) AS rate SET a.array_rating = rate RETURN a



```
MATCH (a:App)-[:HAS]→(r:Review)
WITH a AS a,collect (r.star_rating) AS rate
SET a.array_rating = rate
RETURN a
```

```
$ match (a:App) return a
```

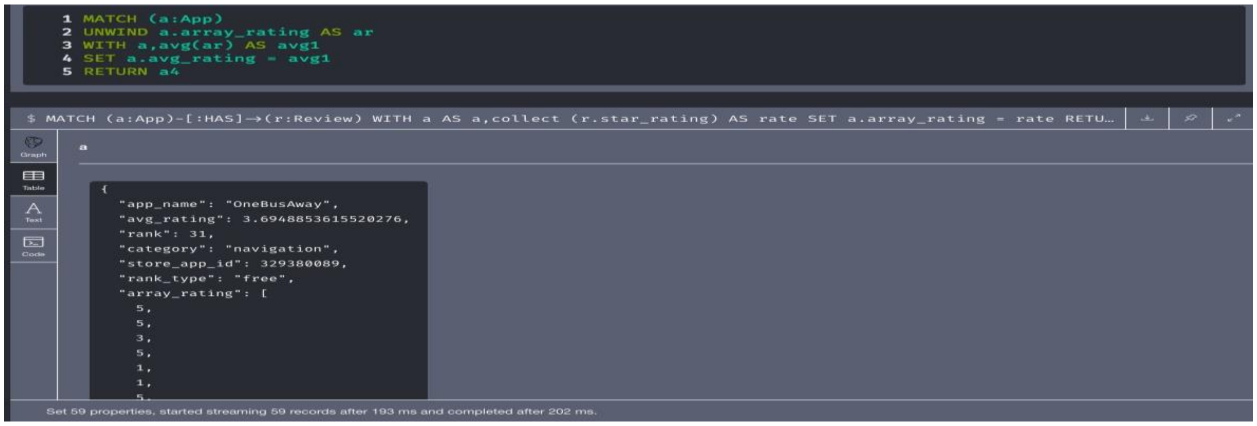
```
{
  "app_name": "OneBusAway",
  "rank": 31,
  "category": "navigation",
  "store_app_id": 329380089,
  "rank_type": "free",
  "array_rating": [
    5,
    5,
    3,
    5,
    1,
    1,
    5,
    5,
```

Started streaming 59 records after 1 ms and completed after 13 ms.

Summary: Here a new field is added for the ease of finding the average of the rating for a given app.

b) SET THE AVG RATING FROM THE ARRAY OF RATINGS

MATCH (a:App) UNWIND a.array_rating AS ar WITH a,avg(ar) AS avg1 SET a.avg_rating = avg1 RETURN a



```
MATCH (a:App)
UNWIND a.array_rating AS ar
WITH a,avg(ar) AS avg1
SET a.avg_rating = avg1
RETURN a4
```

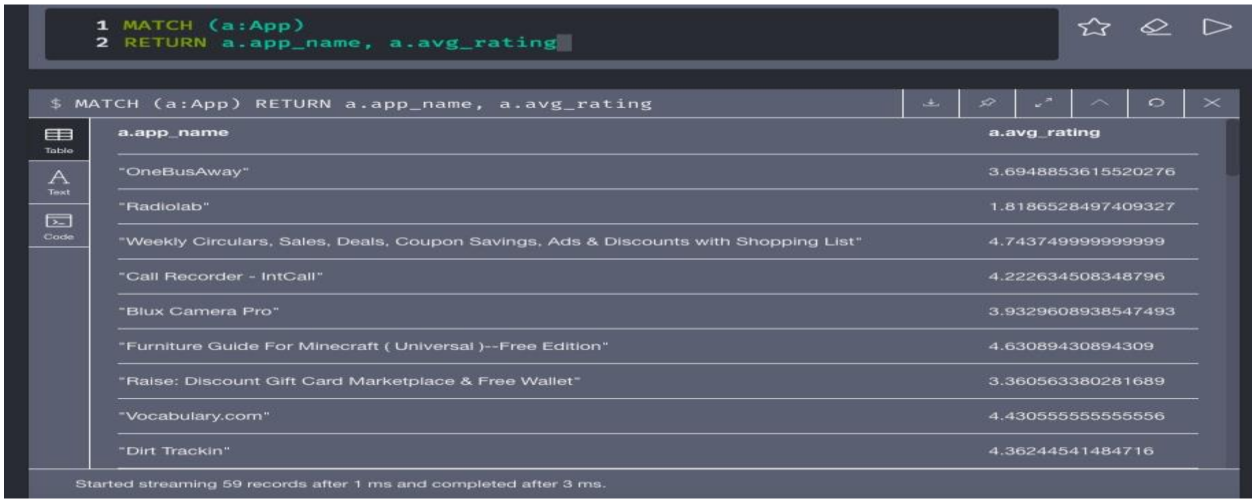
```
$ MATCH (a:App)-[:HAS]→(r:Review) WITH a AS a,collect (r.star_rating) AS rate SET a.array_rating = rate RETU…
```

```
a
{
  "app_name": "OneBusAway",
  "avg_rating": 3.6948853615520276,
  "rank": 31,
  "category": "navigation",
  "store_app_id": 329380089,
  "rank_type": "free",
  "array_rating": [
    5,
    5,
    3,
    5,
    1,
    1,
    5,
```

Set 59 properties, started streaming 59 records after 193 ms and completed after 202 ms.

Summary: Here we are finding the average within the array_rating and setting it as avr_rating.

Showing the different Apps with their average ratings



```
MATCH (a:App)
RETURN a.app_name, a.avg_rating
```

```
$ MATCH (a:App) RETURN a.app_name, a.avg_rating
```

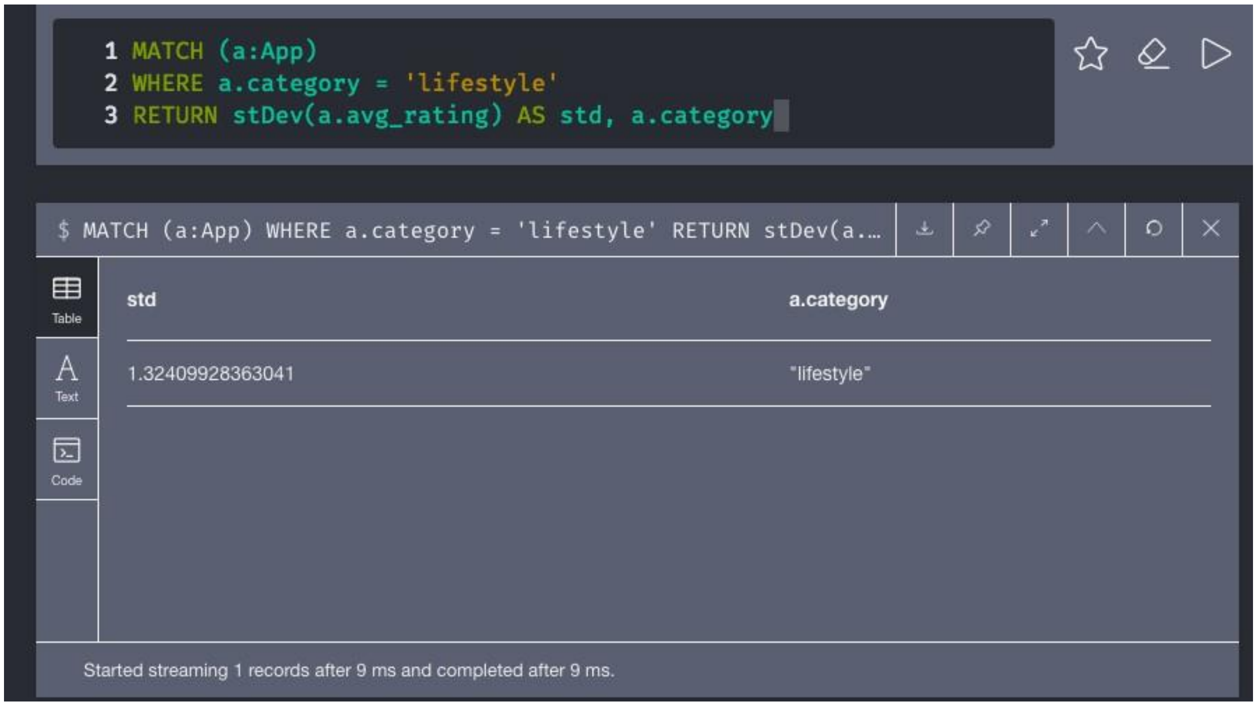
a.app_name	a.avg_rating
"OneBusAway"	3.6948853615520276
"Radiolab"	1.8186528497409327
"Weekly Circulars, Sales, Deals, Coupon Savings, Ads & Discounts with Shopping List"	4.743749999999999
"Call Recorder - IntCall"	4.222634508348796
"Blux Camera Pro"	3.9329608938547493
"Furniture Guide For Minecraft (Universal)--Free Edition"	4.63089430894309
"Raise: Discount Gift Card Marketplace & Free Wallet"	3.360563380281689
"Vocabulary.com"	4.430555555555556
"Dirt Trackin"	4.36244541484716

Started streaming 59 records after 1 ms and completed after 3 ms.

c) To find the Standard deviation of star rating for the lifestyle & business app categories (for demo purpose we are taking only 2 categories)

Standard deviation for the category lifestyle

MATCH (a:App) WHERE a.category = ‘lifestyle’ RETURN stDev(a.avg_rating) AS std, a.category

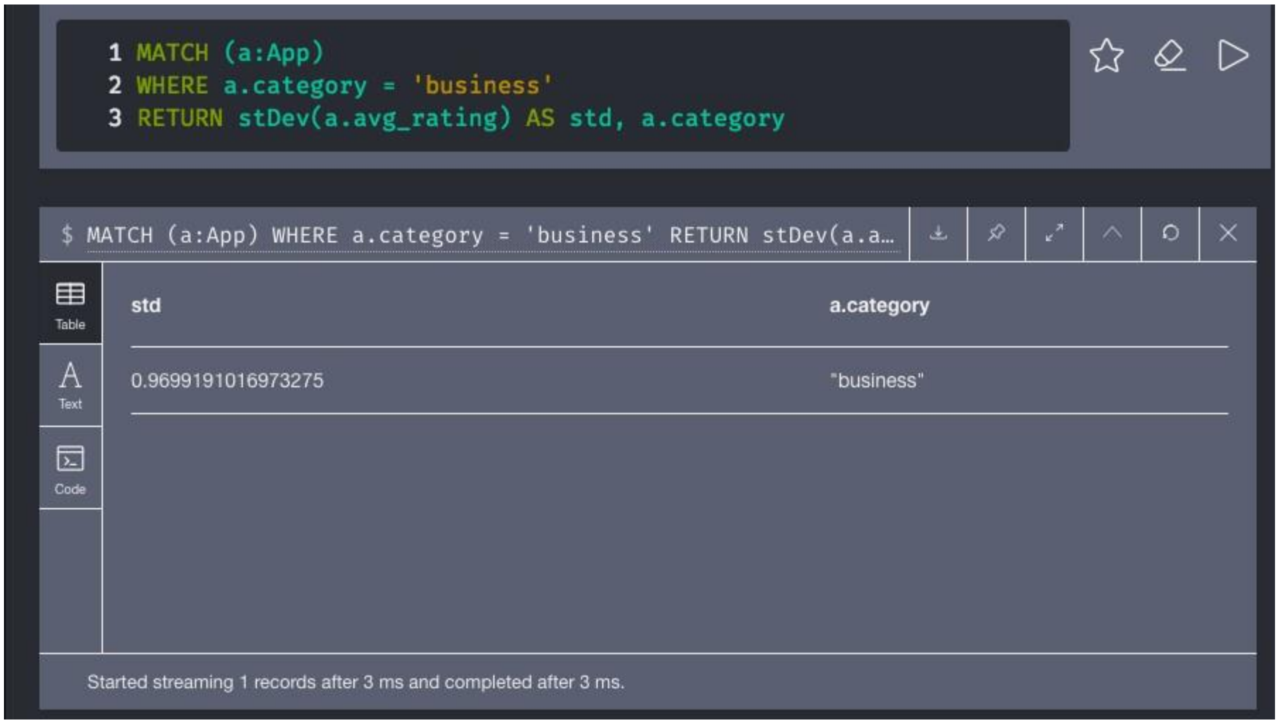

```
MATCH (a:App)
WHERE a.category = 'lifestyle'
RETURN stDev(a.avg_rating) AS std, a.category
```

```
$ MATCH (a:App) WHERE a.category = 'lifestyle' RETURN stDev(a.…
```

std	a.category
1.32409928363041	"lifestyle"

```
Started streaming 1 records after 9 ms and completed after 9 ms.
```

Standard deviation for the category business
MATCH (a:App) WHERE a.category = 'business' RETURN stDev(a.avg_rating) AS std, a.category


```
MATCH (a:App)
WHERE a.category = 'business'
RETURN stDev(a.avg_rating) AS std, a.category
```

```
$ MATCH (a:App) WHERE a.category = 'business' RETURN stDev(a.a…
```

std	a.category
0.9699191016973275	"business"

```
Started streaming 1 records after 3 ms and completed after 3 ms.
```

Summary: To find out the standard deviation, we have first collected all the star_rating for an app and added a field in the app node. Then we have calculated the average rating on the same array and then calculated the standard deviation on that average rating field of all the apps within a category.

Based on the results of the above query we found that apps with similar functionality have considerably divergent ratings (large standard deviation). Based on our findings we conclude that similar functioning apps impede each other's ratings, wherein a popular app in one category downgrade the ratings of other apps in the same category.

SAS -

The sales and popularity of an app are highly dependent on their ratings and reviews. People rely on these metrics for using any app. Since it's such an important feature for any app's success, there are people who try to give fake reviews so as to try and change the market trend towards a particular app or group of apps in lieu of incentives. Fake reviewers are either giving a positive review to certain apps to increase their overall rating or they are giving extremely negative reviews to some apps so as to downgrade their popularity and sales. In order to tackle this problem Opinion Mining in done on the reviews, so as to identify fake reviews. [2]

For our analysis to identify the fakeness of a review, we are proposing to text mine iOS app store data to look for the following features (which we have identified) which will help us finding fake reviews. They are:

1. Intensity of Positive and Negative Emotion words: We are assuming that highly positive or negative reviews have a high chance of being fake.
2. Use of Auxiliary Words (words such as being , be ,might, mine): We are assuming that review texts with a high percentage of auxiliary words can be fake.
3. Tense Used : Here we are assuming that fake reviewers most of the time uses first person words the most (such as I, me), as they want to make it sound more personal.
4. Less analytical text: Here we are taking into account that real reviews usually have a logical or hierarchical flow in their text body while fake reviews are mostly written without the consideration of this flow. A less analytical text translate to a higher chances of it being fake. We can find out the percentage of analytical text in the review body by calculating CDI (Categorical Dynamic Index) which gives a score from 0-100 where higher score infers more of logical text and a lower score infer more informal text [3]
5. The over use of similar words: Fake reviewers have a tendency to over use some terms in their reviews (such as treat and recommend) ,they do this as their main purpose of writing the review is to indirectly publicize the app and attract more customers **Text Mining for Fake Reviews -**

Process Flow

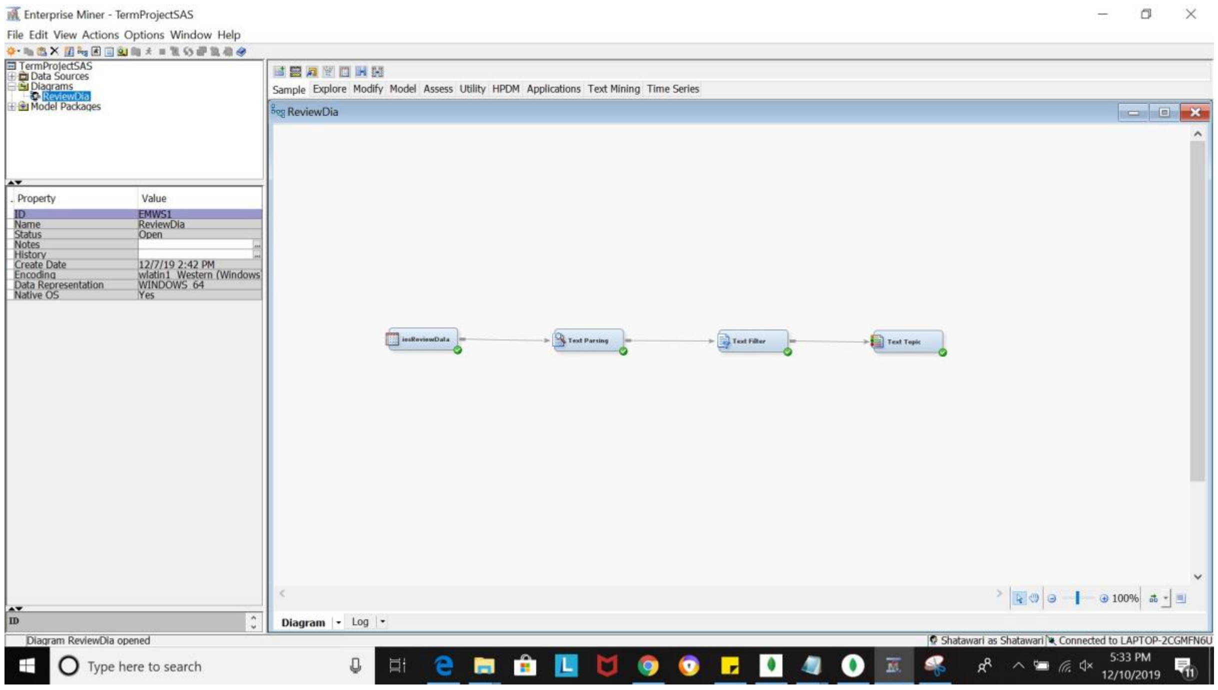
We first did the import of cleaned data in .csv format to SAS using Import Data wizard. The SAS table format data got created: Active Libraries -> work folder. In order to use the converted data in SA Enterprise Text Miner we used the below code to save the data in our machine.

```
"libname out 'D:\Term Project\SASLib\';

proc datasets library=work nodetails nolist;

copy in=work out=out; select New;

run;quit; "
```

We used the following workflow -

Data -> Text Parsing -> Text Filter -> Text Topic

Screenshot - Process flow diagram

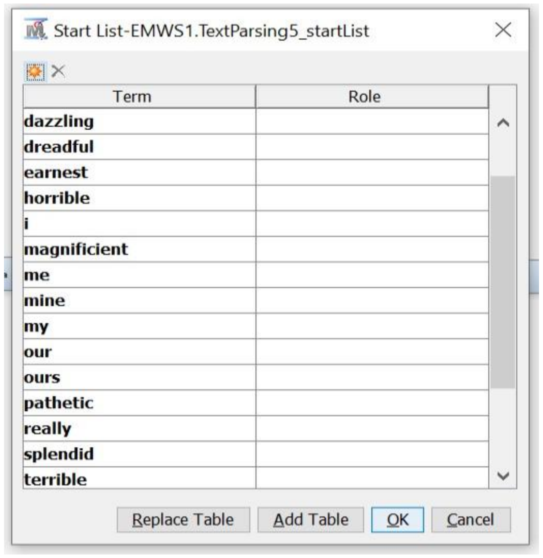


Property	Value
ID	EMWS1
Name	ReviewDia
Status	Open
Notes	
History	
Create Date	12/7/19 2:42 PM
Encoding	wlatin1 Western (Windows
Data Representation	WINDOWS 64
Native OS	Yes

Diagram ReviewDia opened

Shatawari as Shatawari | Connected to LAPTOP-2CGMFN6U

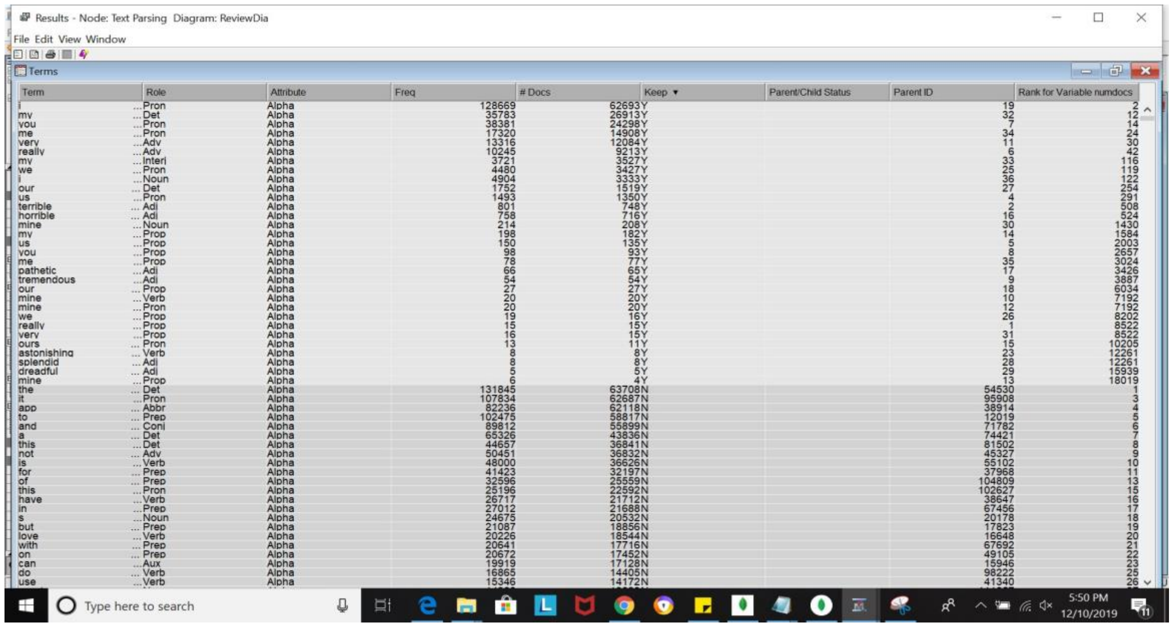
Screenshot - Start word list that we identified which can be categorized as part of Fake reviews



Start List-EMWS1.TextParsing5_startList

Term	Role
dazzling	
dreadful	
earnest	
horrible	
i	
magnificient	
me	
mine	
my	
our	
ours	
pathetic	
really	
splendid	
terrible	

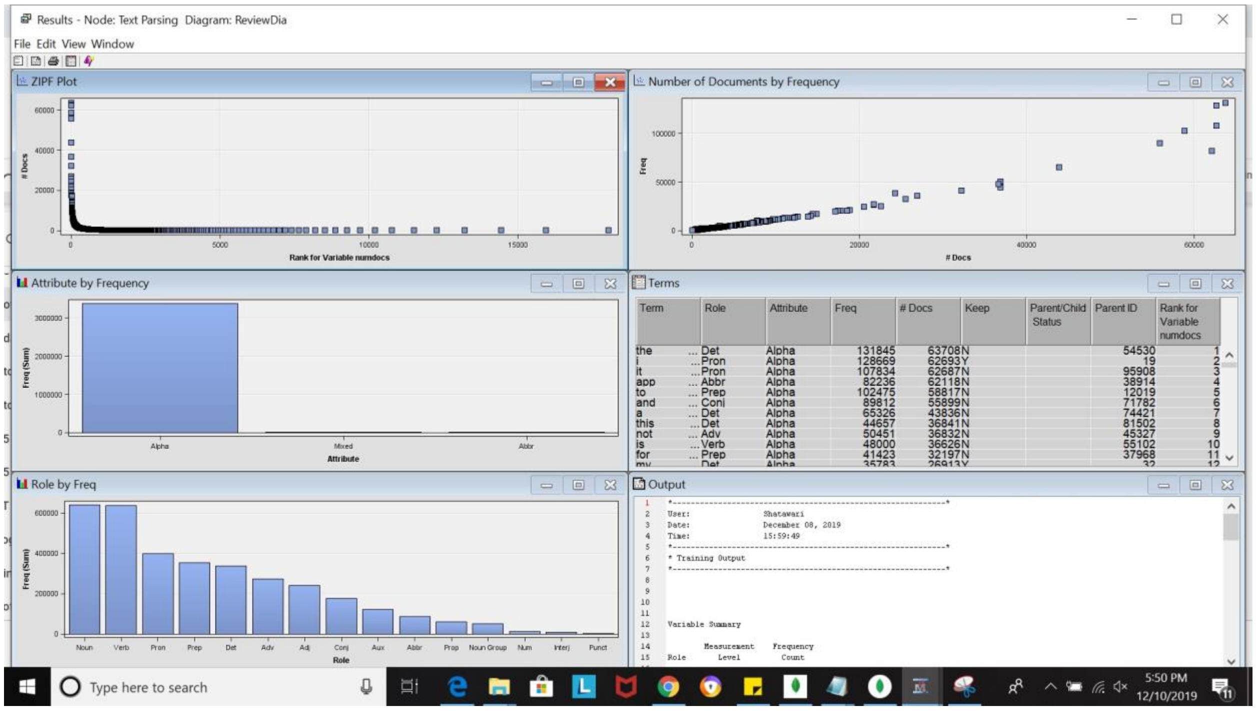
Screenshot - Result of Text Parsing node



Results - Node: Text Parsing Diagram: ReviewDia

Term	Role	Attribute	Freq	# Docs	Keep ▼	Parent/Child Status	Parent ID	Rank for Variable numdocs
i	...Pron	Alpha	128669	62693	Y		19	2
my	...Det	Alpha	35783	26913	Y		32	12
you	...Pron	Alpha	38381	24298	Y		7	14
me	...Pron	Alpha	17320	14908	Y		34	24
very	...Adv	Alpha	13316	12084	Y		11	30
really	...Adv	Alpha	10245	9213	Y		6	42
my	...Interj	Alpha	3721	3527	Y		33	116
we	...Pron	Alpha	4480	3427	Y		25	119
i	...Noun	Alpha	4904	3333	Y		36	122
our	...Det	Alpha	1752	1519	Y		27	254
us	...Pron	Alpha	1493	1350	Y		4	291
terrible	...Adj	Alpha	801	748	Y		2	508
horrible	...Adj	Alpha	758	716	Y		16	524
mine	...Noun	Alpha	214	208	Y		30	1430
my	...Prop	Alpha	198	182	Y		14	1584
us	...Prop	Alpha	150	135	Y		5	2003
you	...Prop	Alpha	98	93	Y		8	2657
me	...Prop	Alpha	78	77	Y		35	3024
pathetic	...Adj	Alpha	66	65	Y		17	3426
tremendous	...Adj	Alpha	54	54	Y		9	3887
our	...Prop	Alpha	27	27	Y		18	6034
mine	...Verb	Alpha	20	20	Y		10	7192
mine	...Pron	Alpha	20	20	Y		12	7192
we	...Prop	Alpha	19	16	Y		26	8202
really	...Prop	Alpha	15	15	Y		1	8522
very	...Prop	Alpha	16	15	Y		31	8522
ours	...Pron	Alpha	13	11	Y		15	10205
astonishing	...Verb	Alpha	8	8	Y		23	12261
splendid	...Adj	Alpha	8	8	Y		28	12261
dreadful	...Adj	Alpha	5	5	Y		29	15939
mine	...Prop	Alpha	6	4	Y		13	18019
the	...Det	Alpha	131845	63708	N		54530	1
it	...Pron	Alpha	107834	62687	N		95908	3
app	...Abbr	Alpha	82236	62118	N		38914	4
to	...Prep	Alpha	102475	58817	N		12019	5
and	...Conj	Alpha	89812	55899	N		71782	6
a	...Det	Alpha	65326	43836	N		74421	7
this	...Det	Alpha	44657	36841	N		81502	8
not	...Adv	Alpha	50451	36832	N		45327	9
is	...Verb	Alpha	48000	36626	N		55102	10
for	...Prep	Alpha	41423	32197	N		37968	11
of	...Prep	Alpha	32596	25559	N		104809	13
this	...Pron	Alpha	25196	22592	N		102627	15
have	...Verb	Alpha	26717	21712	N		38647	16
in	...Prep	Alpha	27012	21688	N		67456	17
s	...Noun	Alpha	24675	20532	N		20178	18
but	...Prep	Alpha	21087	18856	N		17823	19
love	...Verb	Alpha	20226	18544	N		16648	20
with	...Prep	Alpha	20641	17716	N		67692	21
on	...Prep	Alpha	20672	17452	N		49105	22
can	...Aux	Alpha	19919	17128	N		15946	23
do	...Verb	Alpha	16865	14405	N		98222	25
use	...Verb	Alpha	15346	14172	N		41340	26

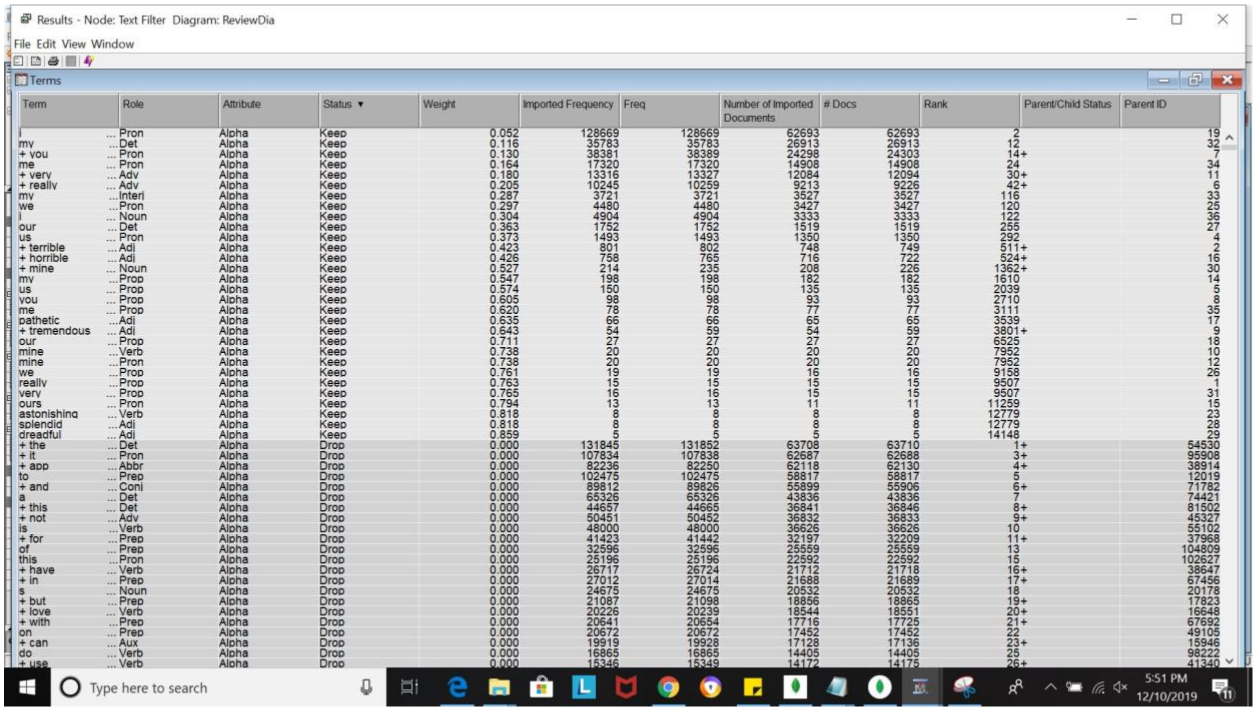
Screenshot – Text Parsing Node Result



Term	Role	Attribute	Freq	# Docs	Keep	Parent/Child Status	Parent ID	Rank for Variable numdocs
the	... Det	Alpha	131845	63708	N		54530	1
i	...Pron	Alpha	128669	62693	Y		19	2
it	...Pron	Alpha	107834	62687	N		95908	3
app	... Abbr	Alpha	82236	62118	N		38914	4
to	... Prep	Alpha	102475	58817	N		12019	5
and	... Conj	Alpha	89812	55899	N		71782	6
a	... Det	Alpha	65326	43836	N		74421	7
this	... Det	Alpha	44657	36841	N		81502	8
not	... Adv	Alpha	50451	36832	N		45327	9
is	... Verb	Alpha	48000	36626	N		55102	10
for	... Prep	Alpha	41423	32197	N		37968	11
my	Det	Alpha	35783	26913	Y		32	12

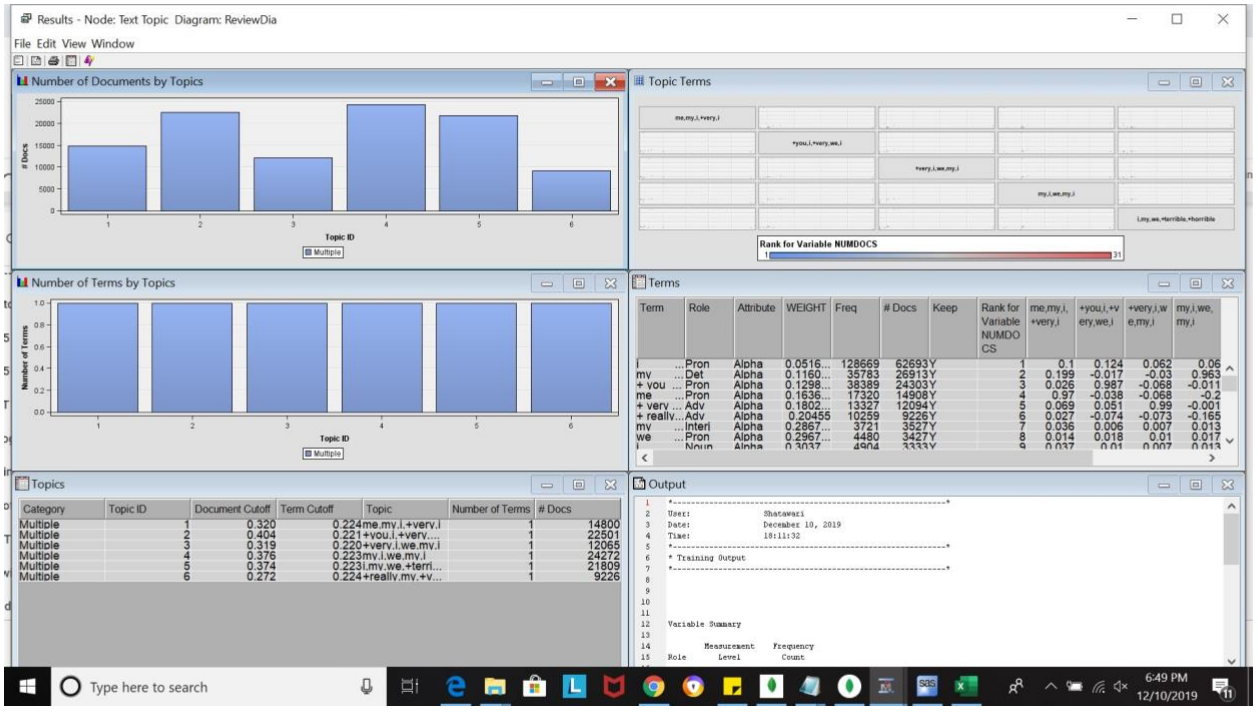
```
*------------------------------------------------------------*
User:               Shatawari
Date:               December 08, 2019
Time:               15:59:49
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*




Variable Summary

         Measurement    Frequency
Role        Level         Count
```

Screenshot : Text Filter Node



Term	Role	Attribute	Status ▼	Weight	Imported Frequency	Freq	Number of Imported Documents	# Docs	Rank	Parent/Child Status	Parent ID
i	... Pron	Alpha	Keep	0.052	128669	128669	62693	62693	2		19
my	...Det	Alpha	Keep	0.116	35783	35783	26913	26913	12		32
+ you	... Pron	Alpha	Keep	0.130	38381	38389	24298	24303	14+		7
me	... Pron	Alpha	Keep	0.164	17320	17320	14908	14908	24		34
+ very	... Adv	Alpha	Keep	0.180	13316	13327	12084	12094	30+		11
+ really	... Adv	Alpha	Keep	0.205	10245	10259	9213	9226	42+		6
my	...Interj	Alpha	Keep	0.287	3721	3721	3527	3527	116		33
we	...Pron	Alpha	Keep	0.297	4480	4480	3427	3427	120		25
i	... Noun	Alpha	Keep	0.304	4904	4904	3333	3333	122		36
our	... Det	Alpha	Keep	0.363	1752	1752	1519	1519	255		27
us	... Pron	Alpha	Keep	0.373	1493	1493	1350	1350	292		4
+ terrible	...Adj	Alpha	Keep	0.423	801	802	748	749	511+		2
+ horrible	...Adj	Alpha	Keep	0.426	758	765	716	722	524+		16
+ mine	... Noun	Alpha	Keep	0.527	214	235	208	226	1362+		30
my	...Prop	Alpha	Keep	0.547	198	198	182	182	1610		14
us	... Prop	Alpha	Keep	0.574	150	150	135	135	2039		5
you	... Prop	Alpha	Keep	0.605	98	98	93	93	2710		8
me	... Prop	Alpha	Keep	0.620	78	78	77	77	3111		35
pathetic	...Adj	Alpha	Keep	0.635	66	66	65	65	3539		17
+ tremendous	... Adj	Alpha	Keep	0.643	54	59	54	59	3801+		9
our	... Prop	Alpha	Keep	0.711	27	27	27	27	6525		18
mine	...Verb	Alpha	Keep	0.738	20	20	20	20	7952		10
mine	...Pron	Alpha	Keep	0.738	20	20	20	20	7952		12
we	...Prop	Alpha	Keep	0.761	19	19	16	16	9158		26
really	...Prop	Alpha	Keep	0.763	15	15	15	15	9507		1
very	... Prop	Alpha	Keep	0.765	16	16	15	15	9507		31
ours	... Pron	Alpha	Keep	0.794	13	13	11	11	11259		15
astonishing	... Verb	Alpha	Keep	0.818	8	8	8	8	12779		23
splendid	...Adj	Alpha	Keep	0.818	8	8	8	8	12779		28
dreadful	... Adj	Alpha	Keep	0.859	5	5	5	5	14148		29
+ the	... Det	Alpha	Drop	0.000	131845	131852	63708	63710	1+		54530
+ it	... Pron	Alpha	Drop	0.000	107834	107838	62687	62688	3+		95908
+ app	... Abbr	Alpha	Drop	0.000	82236	82250	62118	62130	4+		38914
to	... Prep	Alpha	Drop	0.000	102475	102475	58817	58817	5		12019
+ and	... Conj	Alpha	Drop	0.000	89812	89826	55899	55906	6+		71782
a	... Det	Alpha	Drop	0.000	65326	65326	43836	43836	7		74421
+ this	... Det	Alpha	Drop	0.000	44657	44665	36841	36846	8+		81502
+ not	... Adv	Alpha	Drop	0.000	50451	50452	36832	36833	9+		45327
is	...Verb	Alpha	Drop	0.000	48000	48000	36626	36626	10		55102
+ for	... Prep	Alpha	Drop	0.000	41423	41442	32197	32209	11+		37968
of	... Prep	Alpha	Drop	0.000	32596	32596	25559	25559	13		104809
this	...Pron	Alpha	Drop	0.000	25196	25196	22592	22592	15		102627
+ have	... Verb	Alpha	Drop	0.000	26717	26724	21712	21718	16+		38647
+ in	... Prep	Alpha	Drop	0.000	27012	27014	21688	21689	17+		67456
s	... Noun	Alpha	Drop	0.000	24675	24675	20532	20532	18		20178
+ but	... Prep	Alpha	Drop	0.000	21087	21098	18856	18865	19+		17823
+ love	... Verb	Alpha	Drop	0.000	20226	20239	18544	18551	20+		16648
+ with	...Prep	Alpha	Drop	0.000	20641	20654	17716	17725	21+		67692
on	... Prep	Alpha	Drop	0.000	20672	20672	17452	17452	22		49105
+ can	... Aux	Alpha	Drop	0.000	19919	19928	17128	17136	23+		15946
do	... Verb	Alpha	Drop	0.000	16865	16865	14405	14405	25		98222
+ use	... Verb	Alpha	Drop	0.000	15346	15349	14172	14175	26+		41340

Screenshot – Text Topic Node Result



Terms

Term	Role	Attribute	WEIGHT	Freq	# Docs	Keep	Rank for Variable NUMDOCS	me,my,i,+very,i	+you,i,+very,we,i	+very,i,we,my,i	my,i,we,my,i
i ...	Pron	Alpha	0.0516...	128669	62693	Y	1	0.1	0.124	0.062	0.06
my ...	Det	Alpha	0.1160...	35783	26913	Y	2	0.199	-0.017	-0.03	0.963
+ you ...	Pron	Alpha	0.1298...	38389	24303	Y	3	0.026	0.987	-0.068	-0.011
me ...	Pron	Alpha	0.1636...	17320	14908	Y	4	0.97	-0.038	-0.068	-0.2
+ very ...	Adv	Alpha	0.1802...	13327	12094	Y	5	0.069	0.051	0.99	-0.001
+ really...	Adv	Alpha	0.20455	10259	9226	Y	6	0.027	-0.074	-0.073	-0.165
my ...	Interj	Alpha	0.2867...	3721	3527	Y	7	0.036	0.006	0.007	0.013
we ...	Pron	Alpha	0.2967...	4480	3427	Y	8	0.014	0.018	0.01	0.017
i	Noun	Alpha	0.3037	4904	3333	Y	9	0.037	0.01	0.007	0.013

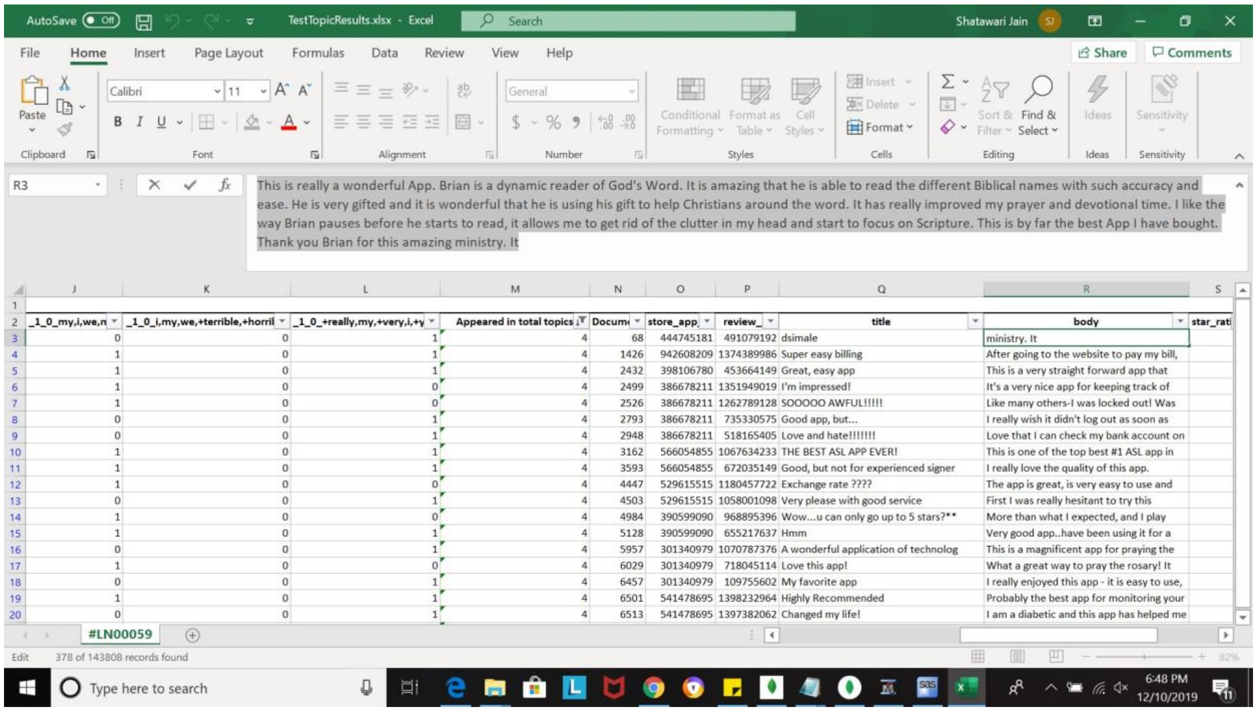
Topics

Category	Topic ID	Document Cutoff	Term Cutoff	Topic	Number of Terms	# Docs
Multiple	1	0.320	0.224	me,my,i,+very,i	1	14800
Multiple	2	0.404	0.221	+you,i,+very,...	1	22501
Multiple	3	0.319	0.220	+very,i,we,my,i	1	12065
Multiple	4	0.376	0.223	my,i,we,my,i	1	24272
Multiple	5	0.374	0.223	i,my,we,+terri...	1	21809
Multiple	6	0.272	0.224	+really,my,+v...	1	9226

Output

```
*------------------------------------------------------------*
User:                Shatawari
Date:                December 10, 2019
Time:                18:11:32
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary

         Measurement    Frequency
Role        Level         Count
```

Screenshot: Text_Topic Result converted in excel format



This is really a wonderful App. Brian is a dynamic reader of God's Word. It is amazing that he is able to read the different Biblical names with such accuracy and ease. He is very gifted and it is wonderful that he is using his gift to help Christians around the word. It has really improved my prayer and devotional time. I like the way Brian pauses before he starts to read, it allows me to get rid of the clutter in my head and start to focus on Scripture. This is by far the best App I have bought. Thank you Brian for this amazing ministry. It

	J	K	L	M	N	O	P	Q	R	S
1										
2	_1_0_my,i,we,n	_1_0_i,my,we,+terrible,+horril	_1_0_+really,my,+very,i,+y	Appeared in total topics	Docum	store_app_	review_	title	body	star_rati
3	0	0	1	4	68	444745181	491079192	dsimale	ministry. It	
4	1	0	1	4	1426	942608209	1374389986	Super easy billing	After going to the website to pay my bill,	
5	1	0	1	4	2432	398106780	453664149	Great, easy app	This is a very straight forward app that	
6	1	0	0	4	2499	386678211	1351949019	I'm impressed!	It's a very nice app for keeping track of	
7	1	0	0	4	2526	386678211	1262789128	SOOOOO AWFUL!!!!!	Like many others-I was locked out! Was	
8	0	0	1	4	2793	386678211	735330575	Good app, but...	I really wish it didn't log out as soon as	
9	0	0	1	4	2948	386678211	518165405	Love and hate!!!!!!!	Love that I can check my bank account on	
10	1	0	1	4	3162	566054855	1067634233	THE BEST ASL APP EVER!	This is one of the top best #1 ASL app in	
11	1	0	1	4	3593	566054855	672035149	Good, but not for experienced signer	I really love the quality of this app.	
12	1	0	0	4	4447	529615515	1180457722	Exchange rate ????	The app is great, is very easy to use and	
13	0	0	1	4	4503	529615515	1058001098	Very please with good service	First I was really hesitant to try this	
14	1	0	0	4	4984	390599090	968895396	Wow...u can only go up to 5 stars?**	More than what I expected, and I play	
15	1	0	1	4	5128	390599090	655217637	Hmm	Very good app..have been using it for a	
16	0	0	1	4	5957	301340979	1070787376	A wonderful application of technolog	This is a magnificent app for praying the	
17	1	0	0	4	6029	301340979	718045114	Love this app!	What a great way to pray the rosary! It	
18	0	0	1	4	6457	301340979	109755602	My favorite app	I really enjoyed this app - it is easy to use,	
19	1	0	1	4	6501	541478695	1398232964	Highly Recommended	Probably the best app for monitoring your	
20	0	0	1	4	6513	541478695	1397382062	Changed my life!	I am a diabetic and this app has helped me	

SAS Text Mining Outcome – For applying the SAS text mining on the review data we used four nodes Text Parsing, Text Filter, Text Topic.

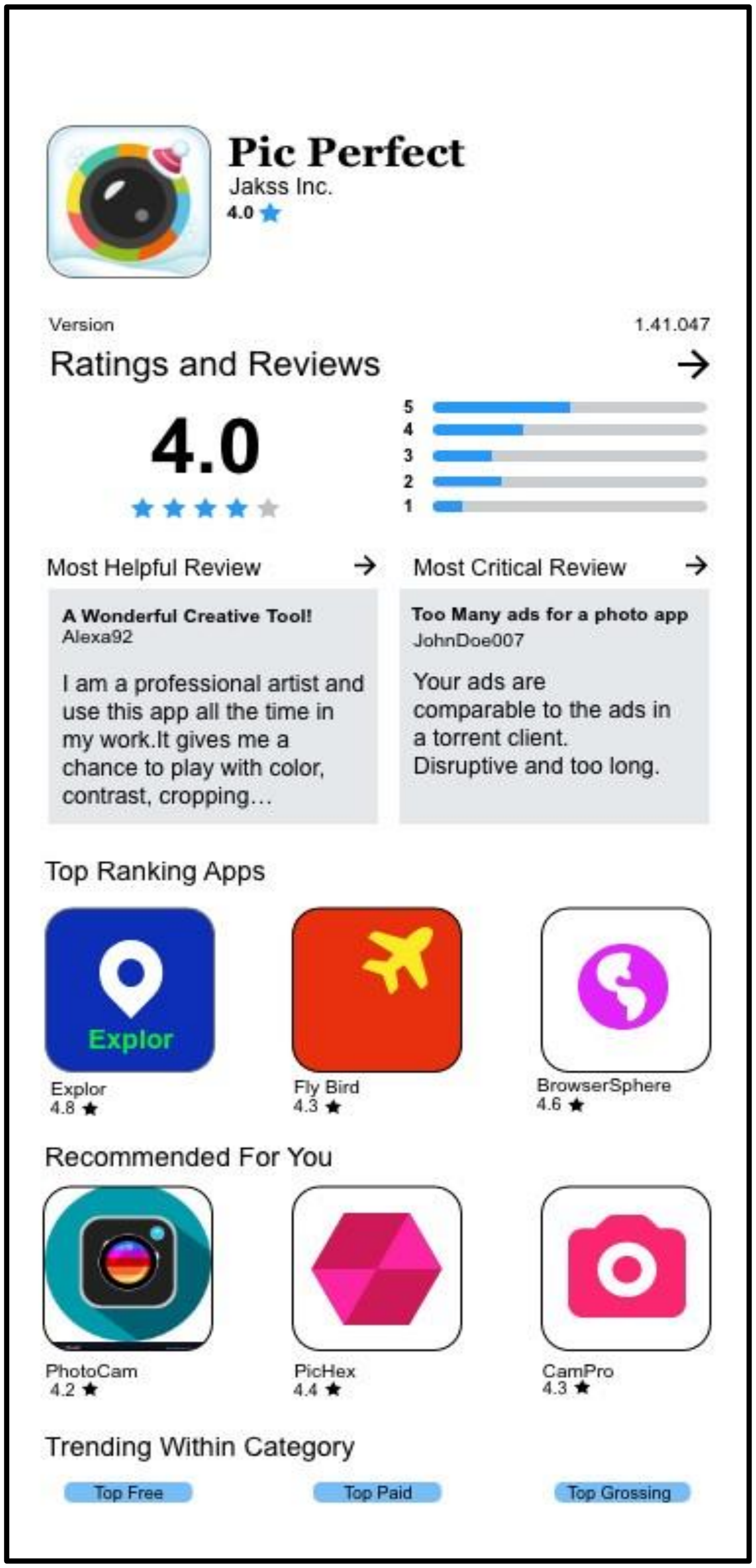
Text Parsing: In this we set start list by giving the list of words as per our recommendation for identifying fake reviews so that while text parsing these words can be kept for further analysis.

Text Filter: In this we set up the term weightings and spell check as yes to apply proper filter to the text parsing node result.

Text Topic: In this we got the topics as per the list of words that we provided and the list of documents in which that topic is present.

Result Analysis – We got the result which we converted in the excel format to analyze it further. As per analysis we can see that the reviews which are part of topic's list that we got post text mining are more towards the Fake side as per our recommendation methodology. We can also update the start list with more words and make the list more specific based on the score calculated by LIWC engine on the basis of Categorical Dynamic Index.

IOS Webpage Design :



Pic Perfect

Jakss Inc.

4.0 ★

Version 1.41.047

Ratings and Reviews →

4.0

★★★★★

Most Helpful Review →

A Wonderful Creative Tool!
Alexa92

I am a professional artist and use this app all the time in my work.It gives me a chance to play with color, contrast, cropping…

Most Critical Review →

Too Many ads for a photo app
JohnDoe007

Your ads are comparable to the ads in a torrent client. Disruptive and too long.

Top Ranking Apps

Explor
4.8 ★

Fly Bird
4.3 ★

BrowserSphere
4.6 ★

Recommended For You

PhotoCam
4.2 ★

PicHex
4.4 ★

CamPro
4.3 ★

Trending Within Category

Top Free | Top Paid | Top Grossing

Conclusion – Our research on iOS Apple App Review System includes understanding and knowledge on insights on how the current iOS App works and its history and evolution. As per our understanding of the iOS Apple app store we designed pseudo schema for MongoDB and graph model for Neo4j. Using various queries on MongoDB we were able to explore the dataset that we took for this project. Based on our understanding we created various analytical queries on MongoDB and Neo4j. We were able to implement our ideology on the wireframe of the webpage that we designed. Based on readings from research papers we constructed our own methodology to identify fake reviews and implemented the same using SAS Enterprise Text Miner.

REFERENCES

[1] https://www.mdpi.com/2071-1050/10/6/1735/pdf

[2] https://pdfs.semanticscholar.org/6005/58b138f32b211ffc71bf918c2005a9e4e0e5.pdf

[3] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0115844