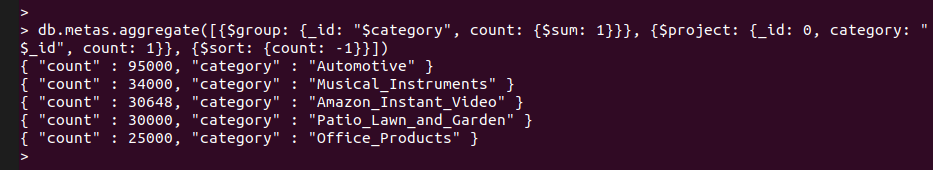
Mini Project 3

By: Anirudh Chaudhary, Quan Le, Bob LeBow, Richard More, Hao Nguyen

# Purpose 1 - MongoDB queries

**a-Overall total number of products per category.**

db.metas.aggregate([{$group: {\_id: "$category", count: {$sum: 1}}}, {$project: {\_id: 0, category: "$\_id", count: 1}}, {$sort: {count: -1}}])

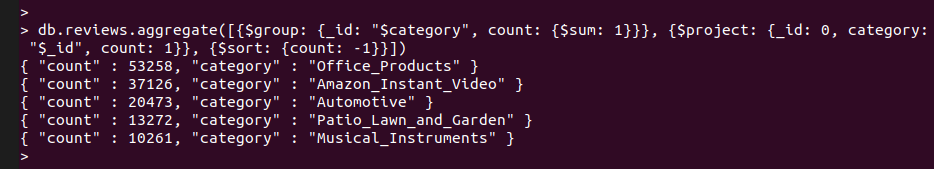


**Summary**: There are a total of 95000 products of Automotive category, 34000 products of Musical Instruments category, 30648 products of Amazon Instant videos category, 30000 products of Patio, lawn & Garden category and 25000 products of Office Products category.

**b-Overall total number of reviews per category, then per product.**

Reviews per category

db.reviews.aggregate([{$group: {\_id: "$category", count: {$sum: 1}}}, {$project: {\_id: 0, category: "$\_id", count: 1}}, {$sort: {count: -1}}])



Reviews per product

db.reviews.aggregate([{$group: {\_id: "$asin", count: {$sum: 1}}}, {$project: {\_id: 0, asin: "$\_id", count: 1}}, {$sort: {count: -1}}])



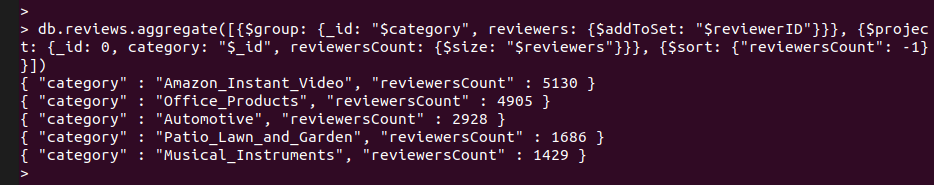
**Summary**: From category perspective, there are 53258, 37126, 20473, 13272 and 10261 number of reviews for Office Products, Amazon Instant videos, Automotive, Patio, lawn & Garden and Musical Instruments respectively.

The second part depicts the numbers of reviews for various products irrespective of its category with ‘count’ as number of reviews and ‘asin’ as a unique ID for each product.

**c-Overall total number of reviewers per category, then per product.**

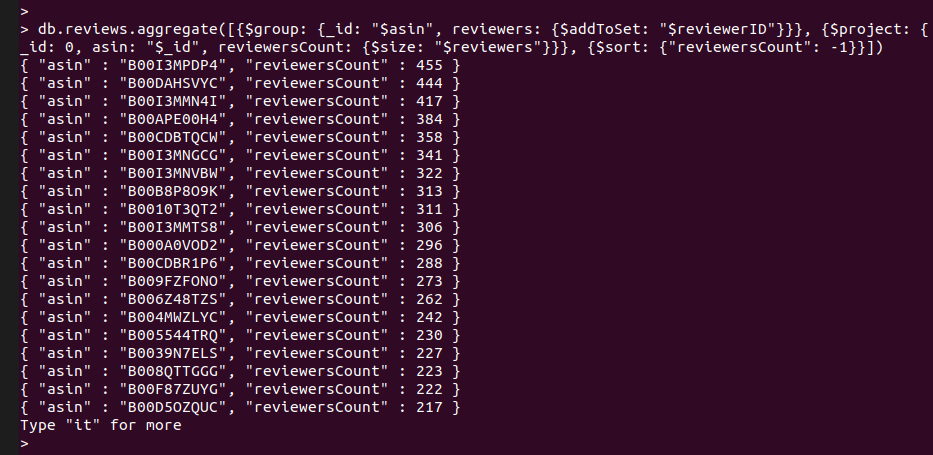
Reviewers per category

db.reviews.aggregate([{$group: {\_id: "$category", reviewers: {$addToSet: "$reviewerID"}}}, {$project: {\_id: 0, category: "$\_id", reviewersCount: {$size: "$reviewers"}}}, {$sort: {"reviewersCount": -1}}])



Reviewers per product

db.reviews.aggregate([{$group: {\_id: "$asin", reviewers: {$addToSet: "$reviewerID"}}}, {$project: {\_id: 0, asin: "$\_id", reviewersCount: {$size: "$reviewers"}}}, {$sort: {"reviewersCount": -1}}])



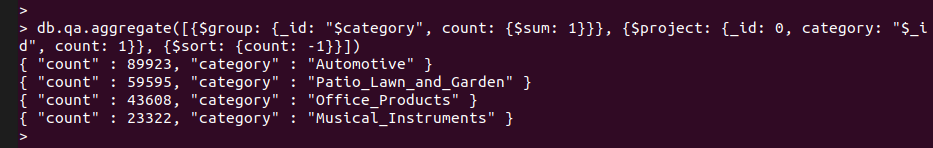
**Summary**: From category perspective, there are 4905, 5130, 2928, 1686 and 1429 number of reviewers for Office Products, Amazon Instant videos, Automotive, Patio, lawn & Garden and Musical Instruments respectively.

The second part depicts the numbers of reviewers for various products irrespective of its category with ‘reviewersCount’ as number of reviewers and ‘asin’ as a unique ID for each product.

**d-Overall total number of Q&A per category, then per product.**

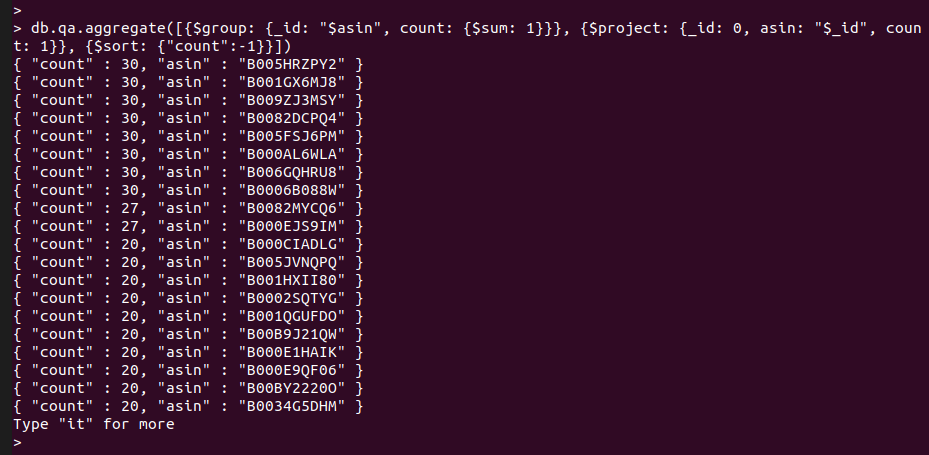
Per category

db.qa.aggregate([{$group: {\_id: "$category", count: {$sum: 1}}}, {$project: {\_id: 0, category: "$\_id", count: 1}}, {$sort: {count: -1}}])



Per product

db.qa.aggregate([{$group: {\_id: "$asin", count: {$sum: 1}}}, {$project: {\_id: 0, asin: "$\_id", count: 1}}, {$sort: {"count":-1}}])

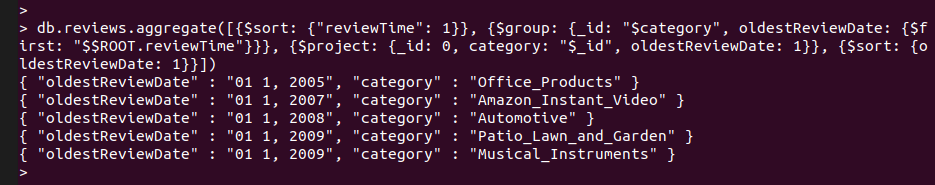


**Summary**: From category perspective, there are 43608, 89923, 59595 and 23322 number of Q&A for Office Products, Automotive, Patio, lawn & Garden and Musical Instruments respectively.

The second part depicts the numbers of Q&A for various products irrespective of its category with ‘count’ as number of Q&A and ‘asin’ as a unique ID for each product.

**e-Date of the oldest review per category**

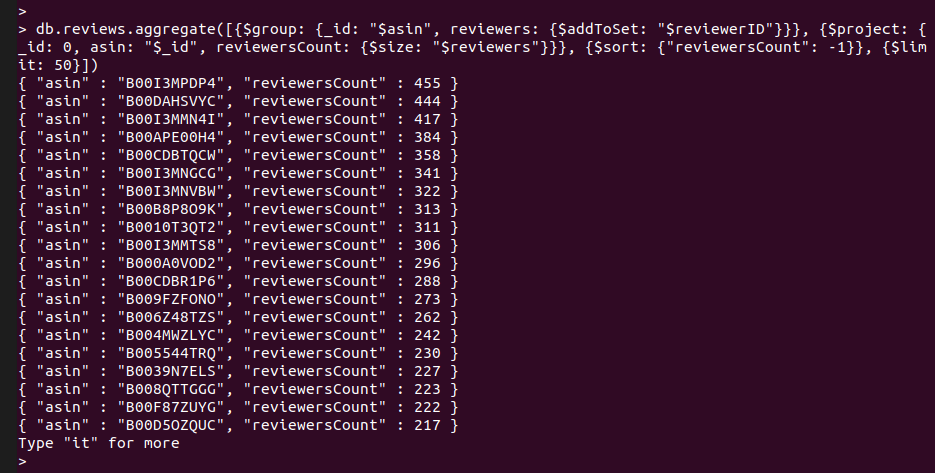
db.reviews.aggregate([{$sort: {"reviewTime": 1}}, {$group: {\_id: "$category", oldestReviewDate: {$first: "$$ROOT.reviewTime"}}}, {$project: {\_id: 0, category: "$\_id", oldestReviewDate: 1}}, {$sort: {oldestReviewDate: 1}}])



**Summary**: From category perspective, the oldest review for Office Products was put on 01-01-2005, for Amazon Instant videos it was put on 01-01-2007, for Automotive it was put on 01-01-2008, for Patio, Lawn & Garden it was put on 01-01-2009 and for Musical Instruments too it was put on 01-01-2009.

**f-Count of reviewers per product –display the results for 50 products only.**

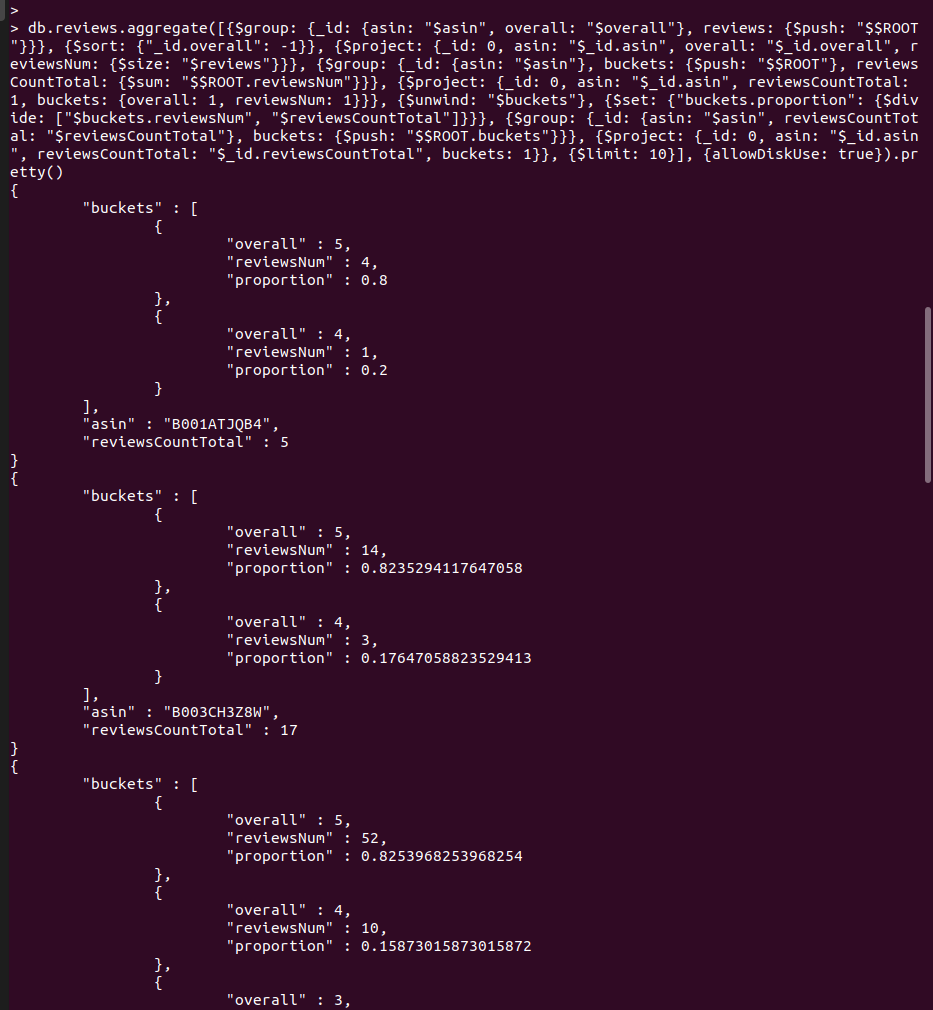
db.reviews.aggregate([{$group: {\_id: "$asin", reviewers: {$addToSet: "$reviewerID"}}}, {$project: {\_id: 0, asin: "$\_id", reviewersCount: {$size: "$reviewers"}}}, {$sort: {"reviewersCount": -1}}, {$limit: 50}])

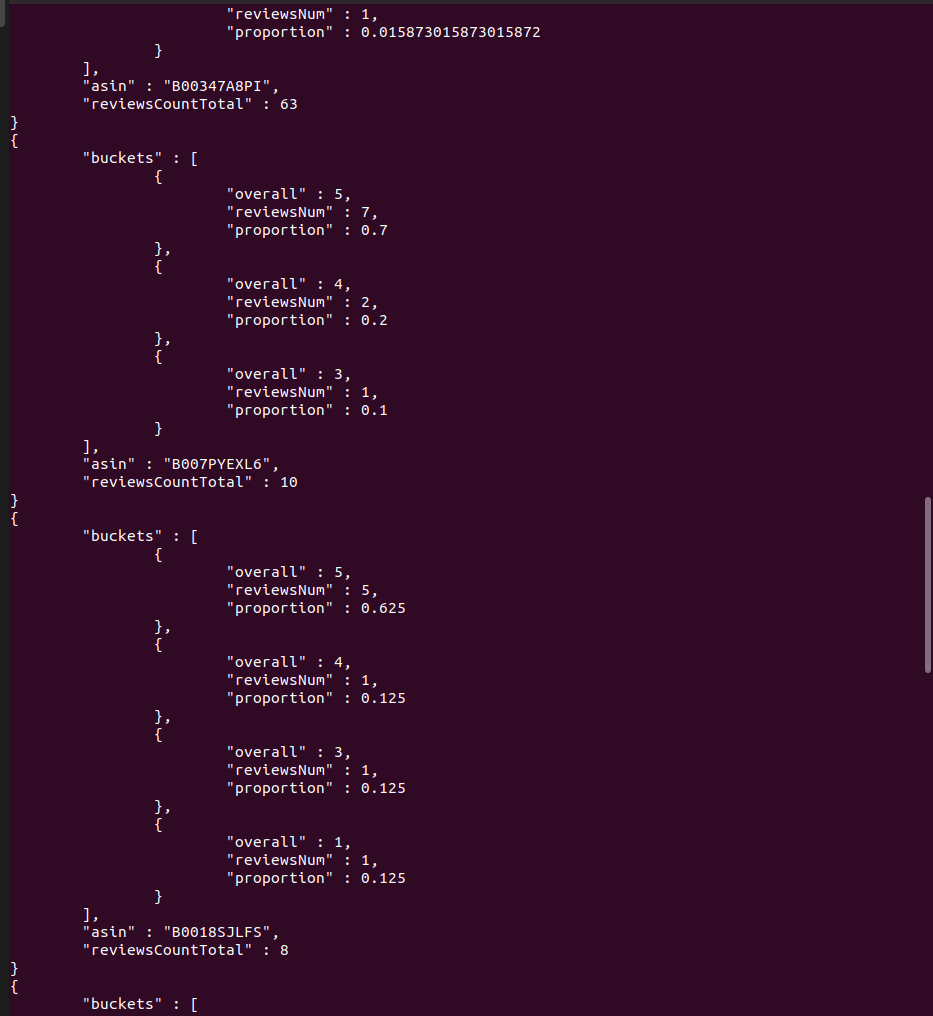


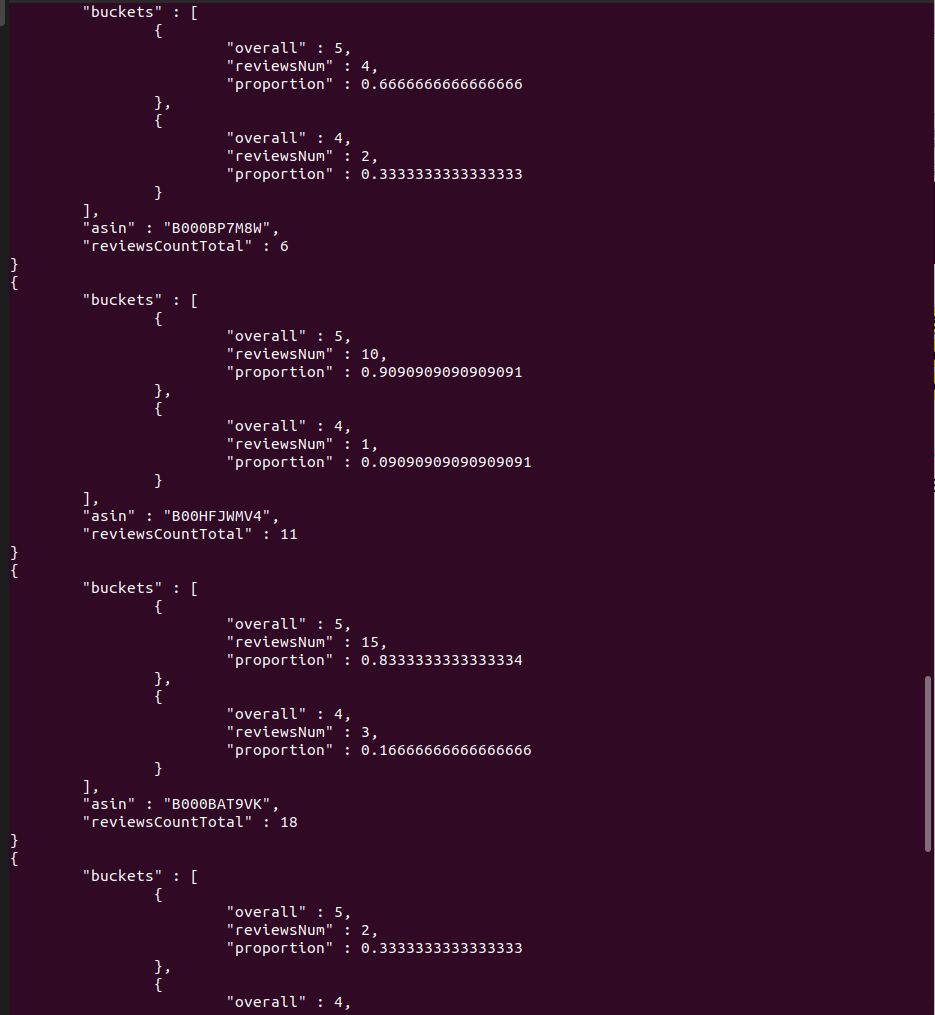
**Summary**: The result shows the count of the reviewers for top 50 products in descending order with ‘reviewersCount’ as number of reviewers and ‘asin’ as a unique ID for each product.

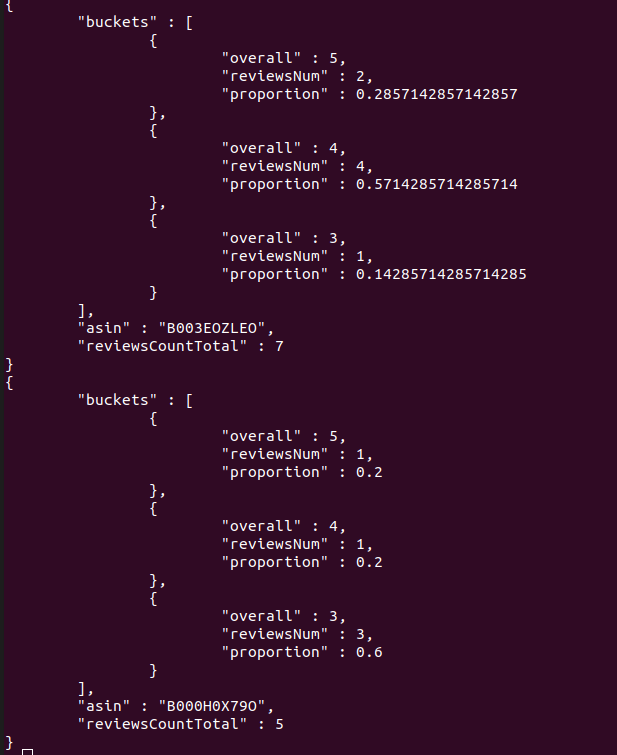
**g-Histogram of review ratings(5’s, 4’s, 3’s, 2’s and 1’s) per product–display the results for 10 products only. Note that there is no requirement for a graphical histogram(like the one on Amazon).**

db.reviews.aggregate([{$group: {\_id: {asin: "$asin", overall: "$overall"}, reviews: {$push: "$$ROOT"}}}, {$sort: {"\_id.overall": -1}}, {$project: {\_id: 0, asin: "$\_id.asin", overall: "$\_id.overall", reviewsNum: {$size: "$reviews"}}}, {$group: {\_id: {asin: "$asin"}, buckets: {$push: "$$ROOT"}, reviewsCountTotal: {$sum: "$$ROOT.reviewsNum"}}}, {$project: {\_id: 0, asin: "$\_id.asin", reviewsCountTotal: 1, buckets: {overall: 1, reviewsNum: 1}}}, {$unwind: "$buckets"}, {$set: {"buckets.proportion": {$divide: ["$buckets.reviewsNum", "$reviewsCountTotal"]}}}, {$group: {\_id: {asin: "$asin", reviewsCountTotal: "$reviewsCountTotal"}, buckets: {$push: "$$ROOT.buckets"}}}, {$project: {\_id: 0, asin: "$\_id.asin", reviewsCountTotal: "$\_id.reviewsCountTotal", buckets: 1}}, {$limit: 10}], {allowDiskUse: true}).pretty()







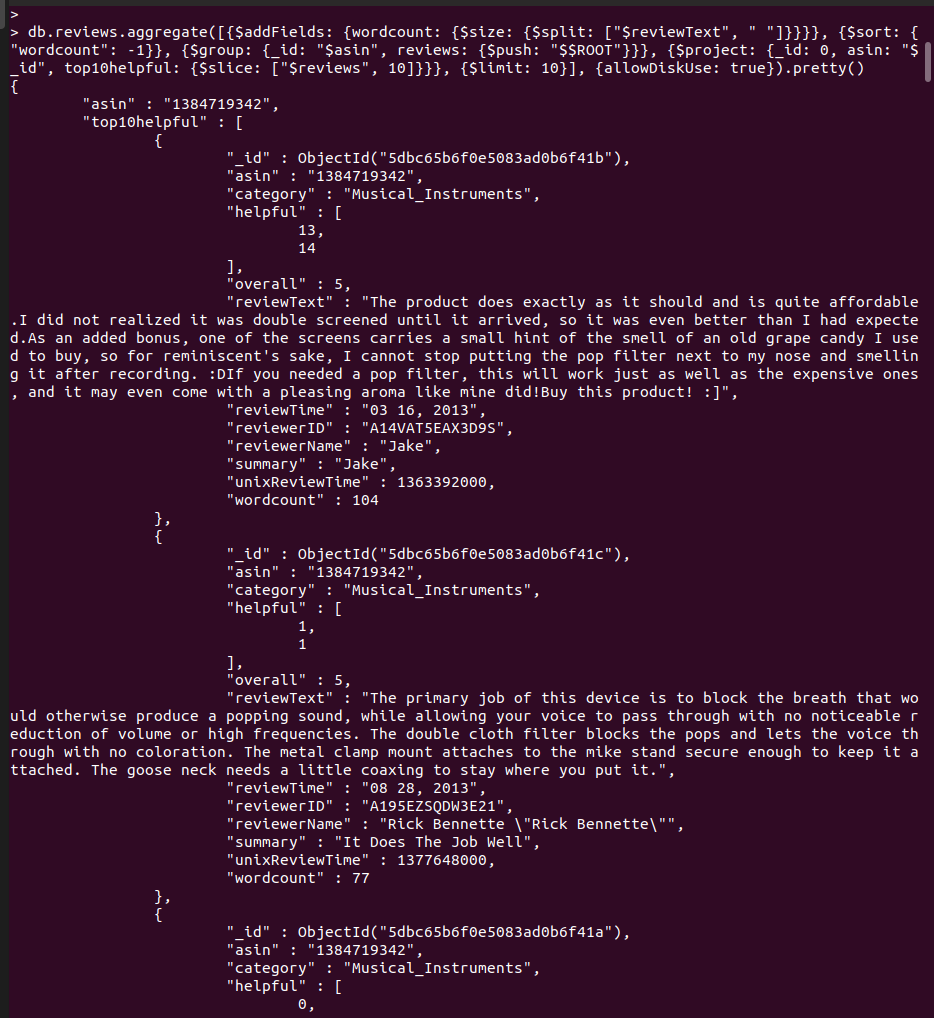


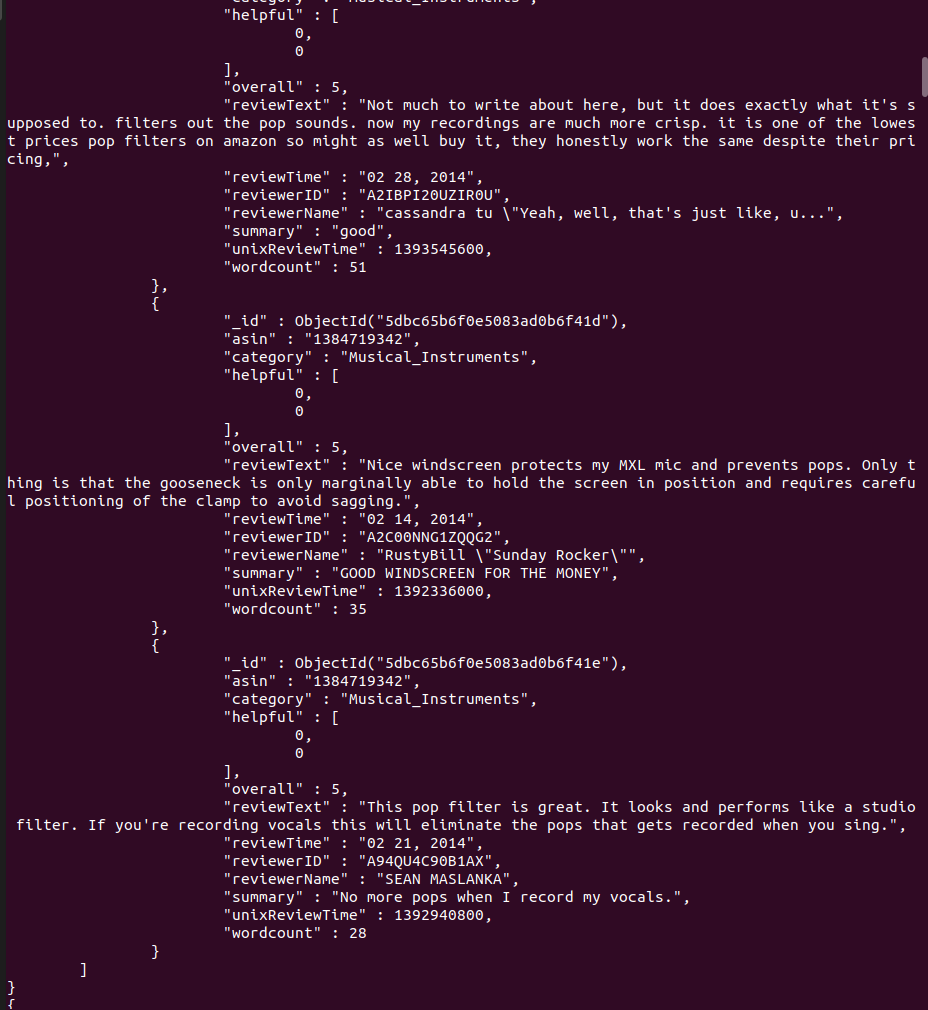
**Summary**: The result displays the distribution of review rating ranging from 1 to 5 for 10 products. It displays the product ID as ‘asin’, total number of reviews for that product as ‘reviewsCountTotal’ and the distribution of the ratings in the form of an array under ‘buckets’ showcasing how many reviews (‘reviewsNum’) received which rating (‘overall’) ranging from 5 to 1 (if any).

**h-Top 10 most helpful reviews per product –display the results for 10 products only (you will have to first provide and justify your own text-based definition of what helpful means and thus not use the Helpful field)**

Most helpful reviews are the longest reviews, so sorting can be based on the number of words in the review.

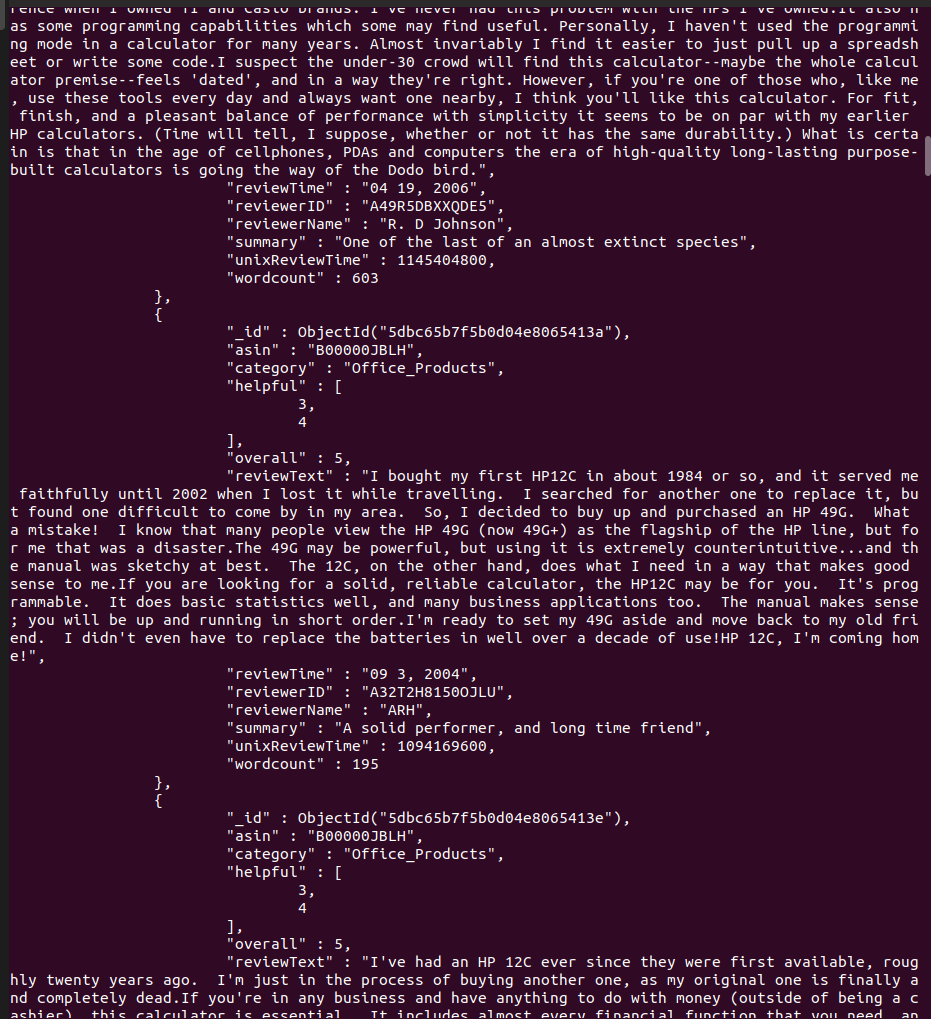
db.reviews.aggregate([{$addFields: {wordcount: {$size: {$split: ["$reviewText", " "]}}}}, {$sort: {"wordcount": -1}}, {$group: {\_id: "$asin", reviews: {$push: "$$ROOT"}}}, {$project: {\_id: 0, asin: "$\_id", top10helpful: {$slice: ["$reviews", 10]}}}, {$limit: 10}], {allowDiskUse: true}).pretty()









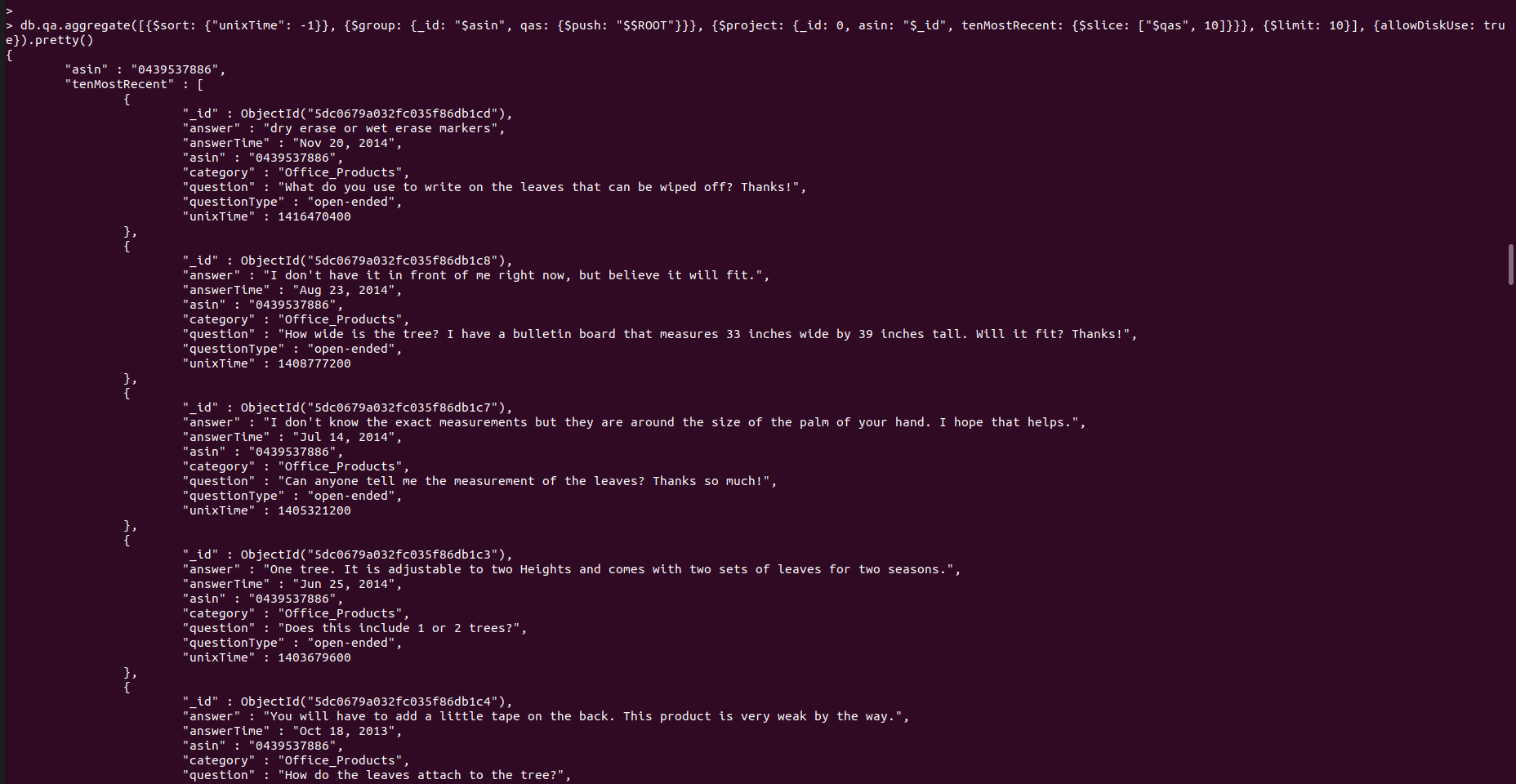


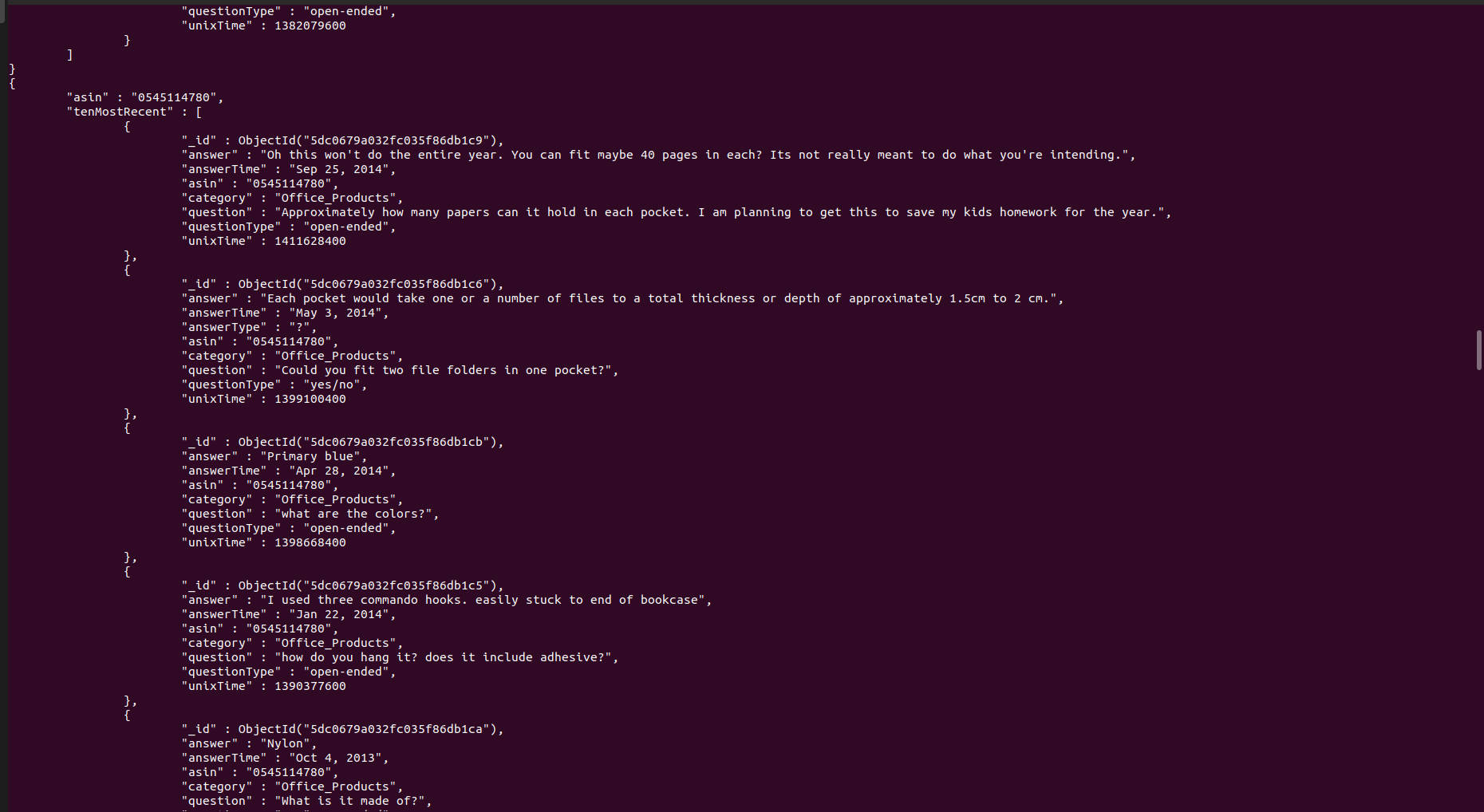
**Summary**: The result displays the top 10 most helpful reviews - the length of the review is the helpfulness - for 10 products with product id as ‘asin’ and the information of the top 10 reviews in the form of an array under ‘top10helpful’. Each subdocument in the array has all the info related to a review.

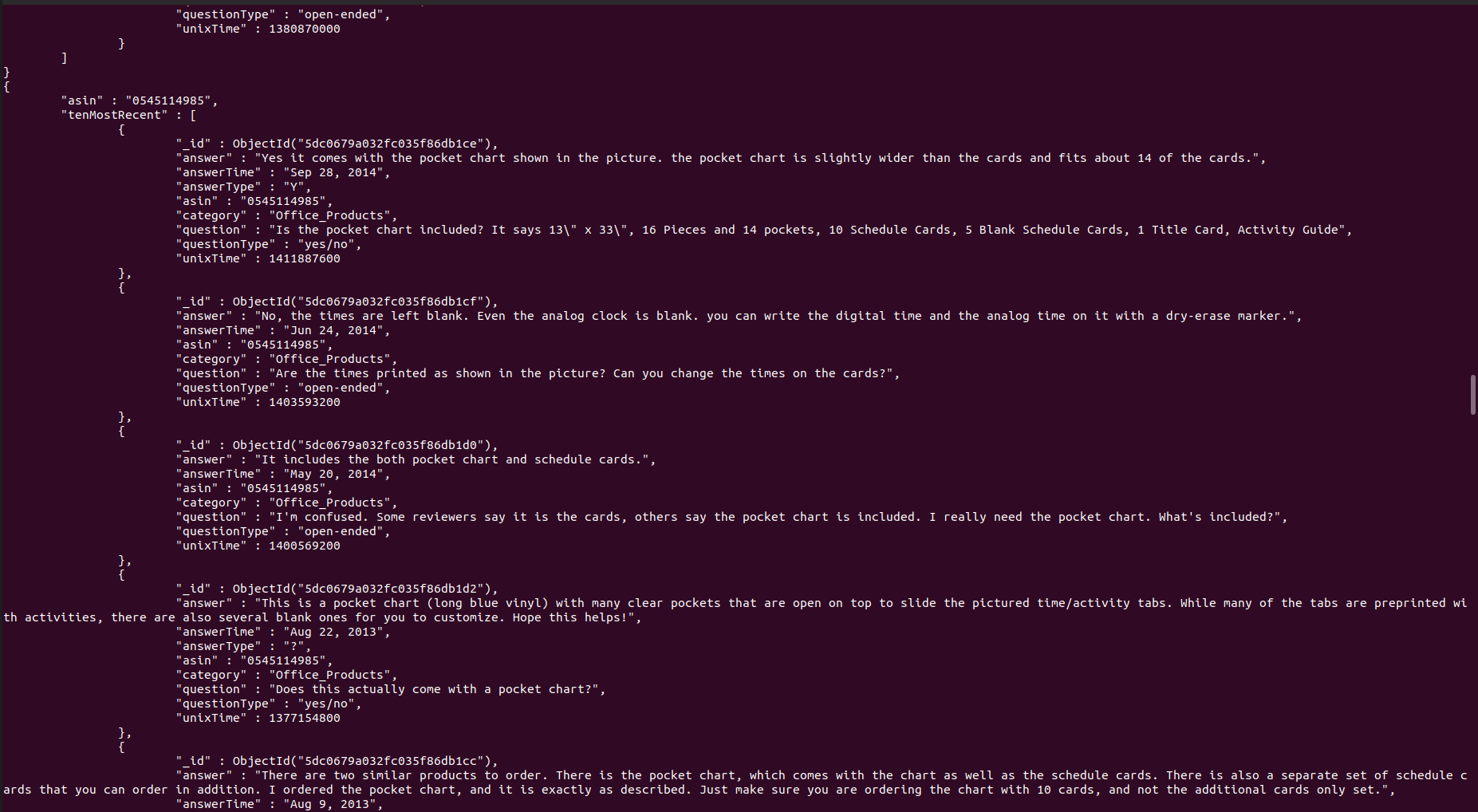
**i-Top 10 most recent Q&A per product –display the results for 10 products only**

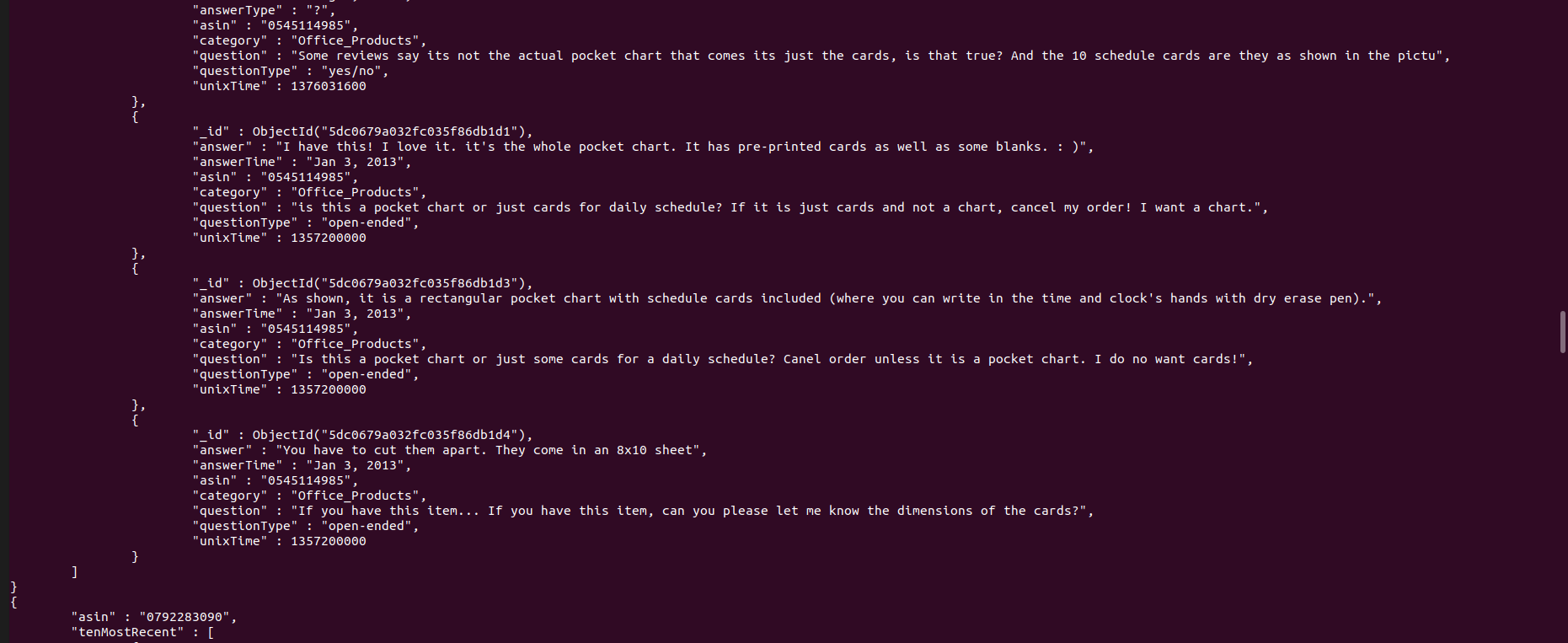
db.qa.aggregate([{$sort: {"unixTime": -1}}, {$group: {\_id: "$asin", qas: {$push: "$$ROOT"}}}, {$project: {\_id: 0, asin: "$\_id", tenMostRecent: {$slice: ["$qas", 10]}}}, {$limit: 10}], {allowDiskUse: true}).pretty()

Snapshot of the first 3 products and their top 10 Q&A - although these have less than 10 Q&As so only displaying the number of Q&As available





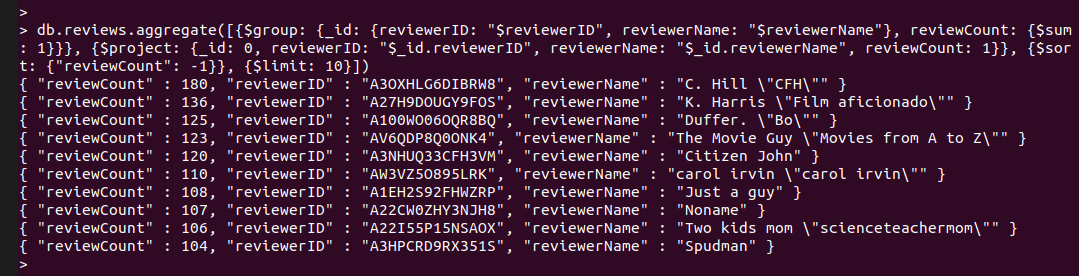




**Summary**: The result shows the top 10 most recent Q&A for 10 products in the form of an array under the notation ‘tenMostRecent’ where each sub document contains the information attributes about the Q&A like id, answer, answer time, product id, category, question, question type and question time.

**j-Top 10 most prolific reviewers (i.e. with most number of reviews)**

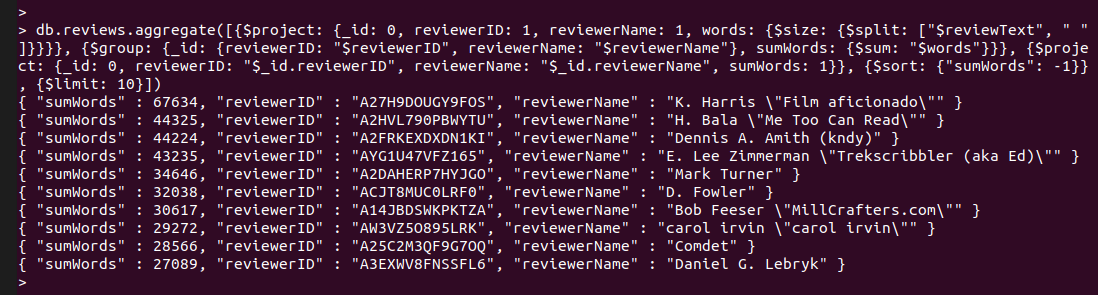
db.reviews.aggregate([{$group: {\_id: {reviewerID: "$reviewerID", reviewerName: "$reviewerName"}, reviewCount: {$sum: 1}}}, {$project: {\_id: 0, reviewerID: "$\_id.reviewerID", reviewerName: "$\_id.reviewerName", reviewCount: 1}}, {$sort: {"reviewCount": -1}}, {$limit: 10}])



**Summary**: The result displays the name, id and count of reviews of the top ten reviewers with most number of reviews ranging from 180 as maximum to 104 as the minimum number of reviews in the top 10 list.

**k-Top 10 most verbose reviewers(i.e. that write the most text)**

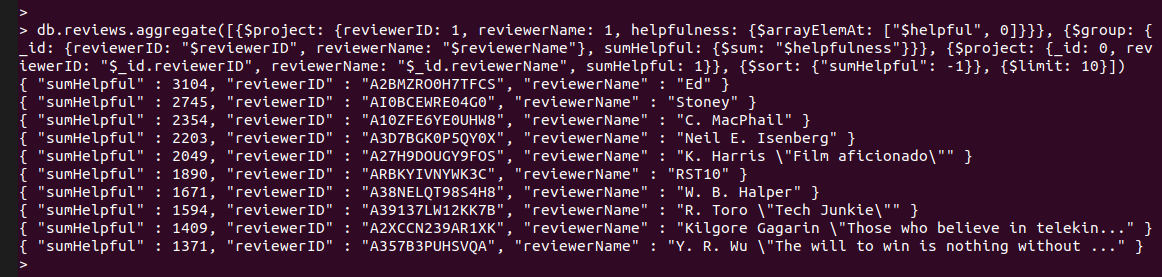
db.reviews.aggregate([{$project: {\_id: 0, reviewerID: 1, reviewerName: 1, words: {$size: {$split: ["$reviewText", " "]}}}}, {$group: {\_id: {reviewerID: "$reviewerID", reviewerName: "$reviewerName"}, sumWords: {$sum: "$words"}}}, {$project: {\_id: 0, reviewerID: "$\_id.reviewerID", reviewerName: "$\_id.reviewerName", sumWords: 1}}, {$sort: {"sumWords": -1}}, {$limit: 10}])



**Summary**: The result displays the name, id and number of words used (‘sumWords’) to write all the reviews of the top ten reviewers who used the most text while writing the reviews. With most words used ranging from 67634 as maximum to 27098 as the minimum number of words used in the top 10 list of the most verbose reviewers.

**l-Top 10 most positive reviewers based on the ratings of their reviews**

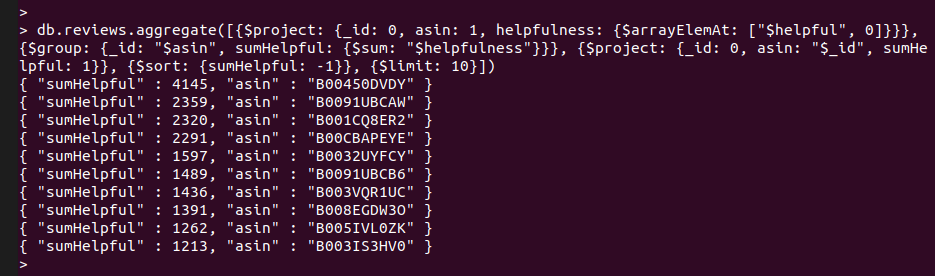
db.reviews.aggregate([{$project: {reviewerID: 1, reviewerName: 1, helpfulness: {$arrayElemAt: ["$helpful", 0]}}}, {$group: {\_id: {reviewerID: "$reviewerID", reviewerName: "$reviewerName"}, sumHelpful: {$sum: "$helpfulness"}}}, {$project: {\_id: 0, reviewerID: "$\_id.reviewerID", reviewerName: "$\_id.reviewerName", sumHelpful: 1}}, {$sort: {"sumHelpful": -1}}, {$limit: 10}])



**Summary**: The result displays the name, id and the cumulative helpfulness rating (‘sumHelpful’) a reviewer got from all their reviews of the top 10 most positive reviewers in the descending order of cumulative helpfulness rating.

**m-The total number of helpful votes received for each product (across all reviews) –display the results for 10 products only**

db.reviews.aggregate([{$project: {\_id: 0, asin: 1, helpfulness: {$arrayElemAt: ["$helpful", 0]}}}, {$group: {\_id: "$asin", sumHelpful: {$sum: "$helpfulness"}}}, {$project: {\_id: 0, asin: "$\_id", sumHelpful: 1}}, {$sort: {sumHelpful: -1}}, {$limit: 10}])



**Summary**: The result shows the top 10 products with the total number of helpful votes (‘sumHelpful’) received along with their id in form of ‘asin’.

# Purpose 2 - Technical Review

## Pros of MongoDB security features

A recent attack by ransomware caused about ten thousand MongoDB servers being compromised. This incident raised a concern among the community about security when storing important data on MongoDB. In fact, like many other databases, Mongo also has security features to protect data from attack, but users were not using these features properly. According to Murphy D. (2017), these are the common mistakes that MongoDB users:

* Using the default ports
* Not enabling authentication immediately (the biggest issue!)
* When using authentication, giving everyone broad access
* Not using LDAP to force password rotations
* Not forcing SSL usage on the database
* Not limiting database access to known network devices (application hosts, load balancers, and so on)
* Not limiting which network is listening (however this no longer affects any supported versions)

All these common causes mentioned above could be prevented if users utilized all security features of MongoDB. There are four main features in MongoDB which are Access Control and Authentication, Encryption, Auditing System, Network Limitation. If these features are implemented properly, MongoDB can secure data from most types of data compromising attempts.

### Access Control and Authentication

MongoDB requires username and password and authentication database associated with that user to allow someone access to its database. Before providing the username and password and authentication database, users must be able to connect to MongoDB instance first. This separation will assure that data will be secured even when MongoDB instance is accessed by an unauthorized person. Besides, MongoDB Enterprise provides LDAP proxy authentication and Kerberos authentication for a large system/network system.

Like other databases, MongoDB can enable access control to users to limit what they can query to the database. Specific roles will be granted to specific users to make sure enough data is available for them to perform their jobs.

### 2. Encryption

MongoDB provides TLS/SSL, Encryption at Rest and Client-Side Level Encryption option to secure Data at multiple locations. While transferring data within a network, MongoDB encrypts data to make sure only intended clients can read. On the other hand, when data is staying in MongoDB, Storage Engine will be encrypted with AES256 via OpenSSL. With AES256, a symmetric key is required to decrypt data which makes the MongoDB engine unable to be compromised by external people. Finally, when data is in an application instead of MongoDB engine, Client-Side field Level Encryption will be activated to protect data once it is pulled from the system. With all these encryption options enabled, data is in guard regardless of where it is located in MongoDB. Therefore, compromising data encrypted by MongoDB from outside is impossible unless the attack comes from inside.

### 3. Auditing System

MongoDB facilitates audited operation by allowing users to write audit events to the console, [Syslog](https://docs.mongodb.com/manual/reference/glossary/#term-syslog), a JSON file, or a BSON file in Enterprise version. MongoDB can log schema, replica set and sharded cluster, authentication and authorization, and CRUD operations. One advantage of Audit System is that it will write all audit events to memory first, then if an event of a connection was collected, all prior events of that connection will also be available for users to review. This feature guarantees users having information about prior events when they conduct an audit to a specific event and want to refer to its prior related events.

### 4. Network Limitation

MongoDB implements a firewall to allow administrator control over the network communications. Only IP addresses in a white list can access MongoDB successfully, or only IP addresses in black will not be able to access MongoDB. Either approach will limit potential attacks through the network. Also, Mongo strictly uses the IP binding which by default binds the Mongodb binaries to localhost. Administrators can change the IP for Mongodb binaries later by declaring specific IP addresses or net.bindIPAll. Knowing a specific IP address of MongoDB binaries will let administrators implement strictly limitations of access for that IP.

Compared to regular relational databases in terms of security, a NoSQL database as MongoDB is much weaker, so it is likely to be the target of hacking. However, MongoDB is still better than RDBMS in preventing SQL injection attacks. Since all data are not being able to access by SQL queries, MongoDB is immune to SQL injection attack, and that is the only advantage of MongoDB over RDBMS in terms of security.

## 

## Cons of MongoDB Security Features

### 5. Readily Available and Open Source

The 2017 ransomware attacks exposed many large flaws in the system, due to the nature of it being open source and readily available.Older versions of MongoDB, 2.6.0 and below, did not come with default authentication features. Hackers took advantage of this oversight. The attacks stemmed from a search engine for the internet of things called Shodan that allowed hackers to search for installations of MongoDB and check if authentication is required to access the NoSQL database. (Henry-Stocker) Once the hackers had access to the databases they made a copy for themselves and deleted the originals with the objective to hold this data for ransom. In just a few days more than 10,000 databases were deleted and held for ransom (Constantin). This security flaw is mostly due to human error as these databases were made publicly available. The creator of Shodan, John Matherly, found more than 30,000 or 684.8 terabytes of unauthorized databases in July 2015. He warned users then of a potential security breach and called for them to make sure they followed the security checkpoints, outlined by MongoDB themselves. Unfortunately not many users heeded his call (Constantin). Still today Hackers are taking advantage of users who have not updated their systems, evidenced by a 2018 hack that exposed 445 million records (Moore-Colyer).

### 6. Authentication and system access

Many cons of the systems come with from authentication. In 2013, As stated above MongoDB installed with no password credentials, leaving it up to the developers or the user has to make sure they are running in a trusted environment. Another related weakness, any created user is granted read access to the database in its entirety by default. This feature has great potential to cause a data breach, users should be very careful about who they allow to access their system. Once a user is granted admin access they can read/write everything. There is not a system in place by default to let users access only portions of the data (Kirpatrick).

### 7. Clear text data transformation

Data is sent as clear text throughout MongoDB. This present serious security concerns, leaving the system vulnerable to a man-in-the-middle attack. The option for cryptographic protection is left in the hands of application developers (Kirpatrick). Since MongoDB is open source is does allow for many adjustments to be made, but the fact that so many were left up to the developers leaves something to be desired.

### 8. System in early stages

This is a flaw that the developers of MongoDB cannot fix. MongoDB is still young when compared to traditional RDMS. Due to MongoDB’s age there are many flaws that are being patched constantly with updates, which is a good thing for the users. Unfortunately it is up to the users to keep their systems up to date. Which not many of them do evidenced by a 2019 hack where 12,000 databases were affected (TrendMicro). Currently, any competent hacker can look up vulnerabilities for older generations of MongoDB then find users still using operating on these out of date systems and attack them.

**Sources:**

Bob Rudis (2017). Rapid7. *The Ransomware Chronicles A DevOps Survival Guide*. Available At:<https://blog.rapid7.com/2017/01/30/the-ransomware-chronicles-a-devops-survival-guide/>

David Murphy (2017). InfoWorld. *The essential guide to MongoDB security*. Available At:<https://www.infoworld.com/article/3164504/the-essential-guide-to-mongodb-security.html>

MongoDB (n.d). *Security Checklist*. Available at:<https://docs.mongodb.com/manual/administration/security-checklist/>

SoftwareTestingHelp (2019). *SQL Vs NoSQL Exact Differences And Know When To Use NoSQL And SQL*. Available At:<https://www.softwaretestinghelp.com/sql-vs-nosql/>

[Lucian Constantin](https://www.computerworld.com/author/Lucian-Constantin/) (2017). Computer World. *Ransomware groups have deleted over 10,000 MongoDB databases.* Available at:

<https://www.computerworld.com/article/3155260/ransomware-groups-have-deleted-over-10000-mongodb-databases.html>

Roland Moore-Colyer (2018). CloudPro. *Unsecured MongoDB database exposed 200GB of Veeam customer data.* Available at:

<https://www.cloudpro.co.uk/it-infrastructure/security/7674/unsecured-mongodb-database-exposed-200gb-of-veeam-customer-data>

Sandra Henry-Stocker (2017). NetworkWorld. *MongoDB ransomware attacks and lessons learned*. Available at:

<https://www.networkworld.com/article/3157766/mongodb-ransomware-attacks-and-lessons-learned.html>

David Kirpatrick (2013). TrustWave. *Mongodb - Security Weaknesses in a typical NoSQL database.* Available at:

<https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/mongodb-security-weaknesses-in-a-typical-nosql-database/>