CSC4023- Assignment 1
Group 9 : MCHWIL006, CHTBLE001, BLNMAN002, MDZMAP001

1. Dataset

For this assignment we used a TVshows dataset. The dataset was harvested from https://github.com/jdorfman/awesome-json-datasets#tv-shows . The data contains TVShows and their lists of Episodes. Due to our database design, explained in question 2 below, we separated the TVShows and the Episodes to form two collections namely, TVShows collection and Episodes collection.

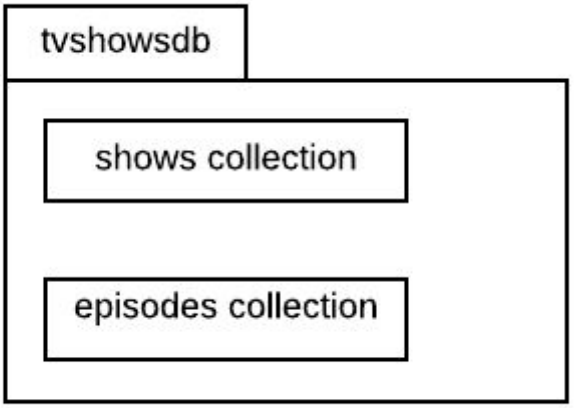
a) Shows Collection
This collection contains information about the TVShows. The attributes are as follow:
Id: Integer value of TVshow ID which is unique for each TVshow
url: String url where the tvshow can be found online
name: String value of the name of the TVshow
type: String value of type of the show whether it is an animation or scripted
language: String value of language which the show is acted in.
genres: An array containing genres of the show
status: String value of status of the show to indicate whether the show ended, or it is new or ongoing.
runtime: String value to indicate how long the show runs
premiered: String value of the date when the show was released.
officialSite: String url where the information about the show can be found
schedule: Document with:
time: String value of time when the show is shown
days: String Array of days the show is aired
rating: Document with
average: average rating out of 10 from viewers.
weight: An integer value to indicate
Network: Document with
-id: Integer value of the Network Id
-name: String value of the Network name
-country: Document with:
-name: String value of the country where the show is shown.
-code: String value of 2-Alpha code of the country
-timezone: String value of the timezone encoded with the show i Information
Episodes: An array of Episodes IDs listing the episodes of the show.

b) Episodes Collection
This collection contains information about the Episodes. The attributes are as follow:

Id: Integer value of Episode ID which is unique for each episode
Showname: Name of the TVShow the episode belong.
url: String url where the episode can be found online
name: String value of the name of the episode
season: Integer value of the season the episode belong
number: Integer value of the number of the episode.
airdate: String value of date when the episode was aired first
airstamp: String value of time when the episode went live first.
airtime: String value of time when the episode is run
runtime: String value of how long the episode is in terms of watching time.
image: Document with:
medium: String value of url where the medium size of logo of episode is found.
original: String value of url where the original size of logo of episode is found.
summary: String value of summary of the episode. Just a synopsis
links: Document with:
self: String value of link where data about the episode can be found.

2. Our MongoDB design

Our database design was affected by the context of the use cases and ensuring efficiency in our database operations.

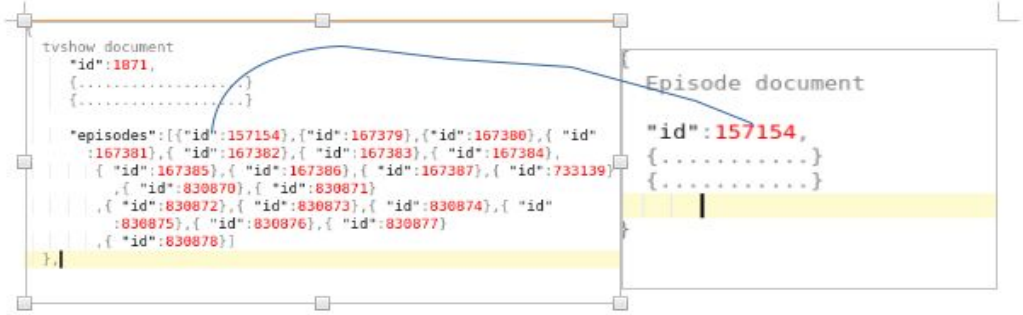


Context: TVshows have many episodes where the number of the episodes keeps growing. The episodes can be added anytime as it depends between the Tv Station programming and availability of content to create a new episode. Therefore this is a one to many relationship where one TVshow has many episodes.

Ensuring efficiency: The dataset we collected had tvshows embedded within it all the episodes that belonged to that tvshow. This affected the size of the tvshow document as more episodes could get added, the size of the tvshow document could grow really big and

become inefficient when retrieving information about a certain episode. For example, if a tvshow document is embedded with information of 200 episodes embedded together, to retrieve information about the 200th episode will take a long time. With this problem, we decided to use document references. To do this we first would create two collections namely the *shows* collections and the *episodes* collection. Each show document in the *shows* collection only stores metadata about the show and an array of episodes ids which belong to that tvshow and reference the episode documents stored in the *episodes* collection. The *episodes* collection stores the information about each and every episode. This will ensure that when a new episode is added, its id gets added to its respective show document (inside its episode array) in the *shows* collection it belongs to and its respective episode document is added to the episodes collection.

The design diagram illustrating our referencing approach is shown below:



3. Loading our MongoDB database

```
MongoDB Enterprise > use TvShows
switched to db TvShows
```

```
MongoDB Enterprise > show dbs
TvShows  0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
MongoDB Enterprise >
```

```
MongoDB Enterprise > db.createCollection("Shows")
{ "ok" : 1 }
```

```
MongoDB Enterprise > db.createCollection("Episodes")
{ "ok" : 1 }
MongoDB Enterprise >
```


```
Mandisas-MacBook-Pro:~ mandisabaleni$ mongoimport --db TvShows --collection Shows --drop --file /Users/mandisabaleni/Downloads/tvshows.json --jsonArray
2020-05-27T02:26:44.899+0200    connected to: mongodb://localhost/
2020-05-27T02:26:44.900+0200    dropping: TvShows.Shows
2020-05-27T02:26:44.933+0200    14 document(s) imported successfully. 0 document(s) failed to import.
Mandisas-MacBook-Pro:~ mandisabaleni$
```

```
Mandisas-MacBook-Pro:~ mandisabaleni$ mongoimport --db TvShows --collection Episodes --drop --file /Users/mandisabaleni/Downloads/Episodes.json --jsonArray
2020-05-27T02:28:13.818+0200    connected to: mongodb://localhost/
2020-05-27T02:28:13.818+0200    dropping: TvShows.Episodes
2020-05-27T02:28:13.863+0200    316 document(s) imported successfully. 0 document(s) failed to import.
Mandisas-MacBook-Pro:~ mandisabaleni$
```

```
MongoDB Enterprise > db.Shows.find().limit(1)
{ "_id" : ObjectId("5ecdb3c40bc7c93da9a81bb8"), "id" : 143, "url" : "http://www.tvmaze.com/shows/143/silicon-valley", "name" : "Silicon Valley", "type"
: "Scripted", "language" : "English", "genres" : [ "Comedy" ], "status" : "Ended", "runtime" : 30, "premiered" : "2014-04-06", "officialSite" : "http://
www.hbo.com/silicon-valley/", "schedule" : { "time" : "22:00", "days" : [ "Sunday" ] }, "rating" : { "average" : 8.5 }, "weight" : 96, "network" : { "id
" : 8, "name" : "HBO", "country" : { "name" : "United States", "code" : "US", "timezone" : "America/New_York" } }, "episodes" : [ { "id" : 10897 }, { "i
d" : 10898 }, { "id" : 10899 }, { "id" : 10900 }, { "id" : 10901 }, { "id" : 10902 }, { "id" : 10903 }, { "id" : 10904 }, { "id" : 117409 }, { "id" : 14
2992 }, { "id" : 142993 }, { "id" : 142994 }, { "id" : 153965 }, { "id" : 154580 }, { "id" : 155129 }, { "id" : 155130 }, { "id" : 155199 }, { "id" : 15
5200 } ] }
MongoDB Enterprise >
```

```
MongoDB Enterprise > db.Episodes.find().limit(1)
{ "_id" : ObjectId("5ecdb41da0de960ad884a504"), "id" : 167380, "url" : "http://www.tvmaze.com/episodes/167380/mr-robot-1x03-eps12d3bugmkv", "name" : "ep
s1.2_d3bug.mkv", "season" : 1, "number" : 3, "airdate" : "2015-07-08", "airtime" : "22:00", "airstamp" : "2015-07-09T02:00:00+00:00", "runtime" : 60, "i
mage" : { "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/106/266701.jpg", "original" : "http://static.tvmaze.com/uploads/images/or
iginal_untouched/106/266701.jpg" }, "summary" : "<p>After being shoved off the railing in Coney Island by Mr. Robot, Elliot wakes up in a hospital room,
 badly banged up. Krista, his psychiatrist, is inclined to send him to rehab unless he's willing to be regularly drug tested. Elliot complies, knowing t
hat he can easily change the hospital records for the tests since he hacked their system long ago.</p>", "_links" : { "self" : { "href" : "http://api.tv
maze.com/episodes/167380" } } }
MongoDB Enterprise >
```

4. Benefits and disadvantages of MongoDB against other databases

<u>i) Relational databases</u>

Relational databases model the data into rows and columns while MongoDB is a NoSQL type of database which models data into collections of JSON documents.

Benefits of MongoDB over Relational databases

- MongoDB has a flexible schema compared to relational databases which has rigid schema. Therefore, MongoDB handles complex data which is not structured while Relational only handles structured data. This makes MongoDB efficient for big data operations as data is complex and not well structured.
- MongoDB does not use Object Relational Mapping as relational databases use when used in developing applications. For software development, MongoDB maps documents directly to data structures unlike relational where you have to create an object first.
- MongoDB queries are faster as no join statements are used unlike in relational databases.

Disadvantages of MongoDB over Relational databases

- MongoDB uses “eventual consistency” unlike relational which uses “strict consistency”. This leads to data in some clusters (nodes) not being consistent with

others at some time but eventually the data is consistent. So if a query is done before eventual consistency happens, it may take time before being returned.
- MongoDB data is duplicated as it is designed for faster queries hence it can be larger than SQL with the same data.

The role of relational dbs if Polyglot Persistence if it was used:

For our use case, Tvshows and Episodes data, polyglot persistence of using both MongoDB and Relational database could not be used as this data is perfect when used with MongoDB as both Tvshows and Episodes data would well fit a non-rigid schema database. Eventual constituency will be okay for this data as its use doesn't have to be real time hence there is no need to store it in Relational data.

ii) Key-value databases

Benefits of MongoDB over Key-value databases:

- Able to get only relevant parts of a document (as a value) instead of the whole document because they have named fields whereas in key-value databases you can't query the values themselves because they're uninterpreted "blobs"
- To update a document (value), you may access it directly whereas with key-value, you need to re-create the key-value store with the same key to update its value
- MongoDB allows system to update a whole unit atomically, key-value dbs don't generally guarantee ACID properties

Disadvantages of MongoDB over Key-value databases:

- Key-value databases can perform operations faster
- Document sizes are limited to 16MB in MongoDB. For efficiency, you would want to limit document sizes and their nested objects. Key-value stores have no limit to how many key-value pairs exist in a store and also don't limit that value size.
- MongoDB has less flexibility to store unstructured data as it has a general document specification whereas key-value dbs have key-value pairs where the value can be stored arbitrarily and be of any type
- Key value stores have no schema

The role of key-value dbs if Polyglot Persistence if it was used:

If polyglot persistence was used with a key-value store included, pipelining several requests to the store could be done so that the store could process all operations without having to wait for one to fully complete before processing another like in Redis. We could make several queries about data whilst updating data concurrently.

Use case example: Since data pertaining to an overall show is less likely to change, store shows in key-value store where several read operations can be pipelined without having to wait for one operation to fully complete, reducing processing time and network overheads.

Store episodes in another kind of store that has ACID properties where it can be frequently updated safely since new episodes are being released all the time.

iii) Graph databases

Benefits of MongoDB over Graph databases:

- MongoDB handles incomplete data and read isolation
- Unlike the graph database which is hard to use on clusters, MongoDB is easy to use on a cluster
- The aggregate element in document supports key based access with BASE properties that allows Basically Available, Soft state, Eventual consistency

Disadvantages of MongoDB over Graph databases:

- no standard query syntax
- slow query performance
- It cannot be used to improve physical storage on nodes and their disk
- Graph Db uses the shortest path while MongoDB must check all documents to find the solution
- Graph dbs have connectedness amongst nodes (entities)

Use cases:

Graph db: social and other human systems for example of representatives, their insight, collaborators and ventures; recommended frameworks where you have to recognize what individuals are doing, purchasing, enjoying; conveyance, steering and other area based frameworks (e.g. coffee shop near you)

Document db: what you think about things fluctuates enormously from 1 thing to the following, and new information on things continue showing up (for example a db of TVShows, online journals; web investigation (recording site hits and guests); content administration frameworks that can store a wide range of sorts of substance

The role of graph dbs Polyglot Persistence if it was used

***To optimize for fast querying and lookup of TVShows' relationships with episodes a graph structure can be used with document db to achieve this.

iv) Column-family databases

Benefits of MongoDB over column-family databases

- Better support for query by value
- No enforcing of a schema which makes it better for unstructured data. Also, there is no need to design a schema before implementing the database.
- Better flexibility with data since documents can be changed on the fly
- Supports better replication and sharding since data which is most likely queried together is stored together in the same documents.

- Better with updates since individual documents are easier to access together with all their attributes
- Because it stores every data attribute as key value pairs, data accessing is very easy. Very easy to access any attribute of any document at once.

Disadvantages of MongoDB over column-family databases

- Column-family gives better performance with range based queries often used in data analytics
- Does not offer much space optimization like column family where rows can be compressed if they are homogeneous data.
- Does not support joins across shards
- Since it stores key value pairs for every attribute of data, it consumes more memory and there is lots of data redundancy
- Lower write scalability since only the master node takes writes
- Limited data size for each document (16MB maximum)

Polyglot persistence in our use case

There would be no need to adopt column-family databases for any part of our use case because our data is not data that grows as fast as to need much scaling. Also, it is not data on which analytics could need to be done.

Example use cases:

- Applications where high volumes of data are stored and are used for range based queries in analytics
- Applications where the ability to always write to the database is crucial

5. Testing MongoDB operations

Our queries have been divided into the four CRUD database operations which are Create, Read, Update and Delete. There are 8 Read operations, 4 Update operations, 2 create operations and 2 Delete operations.

1. **READ: Query that finds shows with a runtime in the range of 30 and 60 minutes and premiered on a 24 June in 2015: db.Shows.find({"runtime": {$gte: 30,$lte: 60},"premiered": "2015-06-24"}).pretty()**

```
MongoDB Enterprise > db.Shows.find({"runtime": {$gte: 30,$lte: 60},"premiered": "2015-06-24"}).pretty()
{
        "_id" : ObjectId("5ed171b85792dc079789c9d8"),
        "id" : 1871,
        "url" : "http://www.tvmaze.com/shows/1871/mr-robot",
        "name" : "Mr. Robot",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Drama",
                "Crime",
                "Thriller"
        ],
        "status" : "Ended",
        "runtime" : 60,
        "premiered" : "2015-06-24",
        "officialSite" : "http://www.usanetwork.com/mrrobot",
        "schedule" : {
                "time" : "22:00",
                "days" : [
                        "Sunday"
                ]
        },
        "rating" : {
                "average" : 8.4
        },
        "weight" : 98,
        "network" : {
                "id" : 30,
                "name" : "USA Network",
                "country" : {
                        "name" : "United States",
                        "code" : "US",
                        "timezone" : "America/New_York"
                }
        },
```

```
        "episodes" : [
                {
                        "id" : 157154
                },
                {
                        "id" : 167379
                },
                {
                        "id" : 167380
                },
                {
                        "id" : 167381
                },
                {
                        "id" : 167382
                },
                {
                        "id" : 167383
                },
                {
                        "id" : 167384
                },
                {
                        "id" : 167385
                },
                {
                        "id" : 167386
                },
                {
                        "id" : 167387
                },
                {
                        "id" : 733139
                },
                {
                        "id" : 830870
                },
                {
                        "id" : 830871
                },
                {
                        "id" : 830872
                },
```

2. **READ: Finding shows with episodes id greater than or equal to 500000 and less than 1000000 with a status to be determined and uses english as language: db.Shows.find({'episodes.id': {$gte: 500000,$lt: 1000000},"status": "To Be Determined","language": "English"})**

```
MongoDB Enterprise > db.Shows.find({'episodes.id': {$gte: 500000,$lt: 1000000},"status": "To Be Determined","language": "English"})
{ "_id" : ObjectId("5ed19e6c5792dc079789cc6a"), "id" : 305, "url" : "http://www.tvmaze.com/shows/305/black-mirror", "name" : "Black Mirror", "type" : "Scripted", "langu
age" : "English", "genres" : [ "Drama", "Science-Fiction", "Thriller" ], "status" : "To Be Determined", "runtime" : 60, "premiered" : "2011-12-04", "officialSite" : "ht
tps://www.netflix.com/title/70264888", "schedule" : { "time" : "", "days" : [ "Wednesday" ] }, "rating" : { "average" : 8.8 }, "weight" : 97, "network" : null, "episode
s" : [ { "id" : 30902 }, { "id" : 30903 }, { "id" : 30904 }, { "id" : 30905 }, { "id" : 30906 }, { "id" : 30907 }, { "id" : 878471 }, { "id" : 878474 }, { "id" : 878470
 }, { "id" : 878473 }, { "id" : 878469 }, { "id" : 878472 }, { "id" : 1281042 }, { "id" : 1281040 }, { "id" : 1281039 }, { "id" : 1281041 }, { "id" : 1281043 }, { "id"
: 1281044 }, { "id" : 1656203 }, { "id" : 1656204 }, { "id" : 1659382 } ] }
```

3. CREATE: Inserts one episode record with name Money Heist categorized as drama genre and have average rating of 8.9: db.Episodes.insertOne({"name": "Money Heist", genres: "Drama","rating.average": 8.9})

```
MongoDB Enterprise > db.Episodes.insertOne({"name": "Money Heist", genres: "Drama","rating.average": 8.9})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5ed197095792dc079789cb22")
}
```

4. UPDATE: Update all records in episodes collection with name eps1.0_hellofriend.mov, change the name to Black Widow: db.Episodes.update ({"name":"eps1.0_hellofriend.mov"}, {$set:{"name" :"Black Widow"}},{multi:true})

```
MongoDB Enterprise > db.Episodes.update({"name":"eps1.0_hellofriend.mov"},{$set:{"name" :"Black Widow"}},{multi:true})
WriteResult({ "nMatched" : 4, "nUpserted" : 0, "nModified" : 4 })
```

5. READ: Shows all shows that air on either the USA network or Showtime: db.Shows.find({$or: [{"network.name":"USA Network"},{"network.name":"Showtime"}]}, {episodes: 0, schedule:0}).pretty()


```
switched to db TvShows
MongoDB Enterprise > db.Shows.find({$or: [{"network.name":"USA Network"},{"network.name":"Showtime"}]}, {episodes: 0, schedule:0}).pretty()
{
        "_id" : ObjectId("5ecdb3c40bc7c93da9a81bb9"),
        "id" : 1871,
        "url" : "http://www.tvmaze.com/shows/1871/mr-robot",
        "name" : "Mr. Robot",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Drama",
                "Crime",
                "Thriller"
        ],
        "status" : "Ended",
        "runtime" : 60,
        "premiered" : "2015-06-24",
        "officialSite" : "http://www.usanetwork.com/mrrobot",
        "rating" : {
                "average" : 8.4
        },
        "weight" : 98,
        "network" : {
                "id" : 30,
                "name" : "USA Network",
                "country" : {
                        "name" : "United States",
                        "code" : "US",
                        "timezone" : "America/New_York"
                }
        }
}
{
        "_id" : ObjectId("5ecdb3c40bc7c93da9a81bba"),
        "id" : 7,
        "url" : "http://www.tvmaze.com/shows/7/homeland",
        "name" : "Homeland",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Drama",
                "Thriller",
                "Espionage"
        ],
        "status" : "Ended",
        "runtime" : 60,
        "premiered" : "2011-10-02",
        "officialSite" : "http://www.sho.com/sho/homeland/home",
        "rating" : {
                "average" : 8.3
        },
        "weight" : 99,
        "network" : {
                "id" : 9,
                "name" : "Showtime",
                "country" : {
                        "name" : "United States",
                        "code" : "US",
                        "timezone" : "America/New_York"
                }
        }
}
MongoDB Enterprise >
```


6. UPDATE:. Adds a field recommendation that says show is highly recommended if its average rating exceeds 7: db.Shows.updateMany({"rating.average":{$gte:7}}, {$set:{"recommendation": "highly recommended"}})
recommended")
Before update:

~ — mongo ~ — -bash

```
}
MongoDB Enterprise > db.Shows.find({"rating.average":{$gte:7}}, {episodes:0}).limit(2).pretty()
{
        "_id" : ObjectId("5ecdb3c40bc7c93da9a81bb8"),
        "id" : 143,
        "url" : "http://www.tvmaze.com/shows/143/silicon-valley",
        "name" : "Silicon Valley",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Comedy"
        ],
        "status" : "Ended",
        "runtime" : 30,
        "premiered" : "2014-04-06",
        "officialSite" : "http://www.hbo.com/silicon-valley/",
        "schedule" : {
                "time" : "22:00",
                "days" : [
                        "Sunday"
                ]
        },
        "rating" : {
                "average" : 8.5
        },
        "weight" : 96,
        "network" : {
                "id" : 8,
                "name" : "HBO",
                "country" : {
                        "name" : "United States",
                        "code" : "US",
                        "timezone" : "America/New_York"
                }
        }
}
{
        "_id" : ObjectId("5ecdb3c40bc7c93da9a81bb9"),
        "id" : 1871,
        "url" : "http://www.tvmaze.com/shows/1871/mr-robot",
        "name" : "Mr. Robot",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Drama",
                "Crime",
                "Thriller"
        ],
        "status" : "Ended",
        "runtime" : 60,
        "premiered" : "2015-06-24",
        "officialSite" : "http://www.usanetwork.com/mrrobot",
        "schedule" : {
                "time" : "22:00",
                "days" : [
                        "Sunday"
                ]
        },
        "rating" : {
                "average" : 8.4
        },
        "weight" : 98,
```

After Update:

~ — mongo ~ — -bash

```
MongoDB Enterprise > db.Shows.updateMany({"rating.average":{$gte:7}}, {$set:{"recommendation": "highly recommended"}})
{ "acknowledged" : true, "matchedCount" : 14, "modifiedCount" : 14 }
MongoDB Enterprise > db.Shows.find({"rating.average":{$gte:7}}, {episodes:0}).limit(2).pretty()
{
        "_id" : ObjectId("5ecdb3c40bc7c93da9a81bb8"),
        "id" : 143,
        "url" : "http://www.tvmaze.com/shows/143/silicon-valley",
        "name" : "Silicon Valley",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Comedy"
        ],
        "status" : "Ended",
        "runtime" : 30,
        "premiered" : "2014-04-06",
        "officialSite" : "http://www.hbo.com/silicon-valley/",
        "schedule" : {
                "time" : "22:00",
                "days" : [
                        "Sunday"
                ]
        },
        "rating" : {
                "average" : 8.5
        },
        "weight" : 96,
        "network" : {
                "id" : 8,
                "name" : "HBO",
                "country" : {
                        "name" : "United States",
                        "code" : "US",
                        "timezone" : "America/New_York"
                }
        },
        "recommendation" : "highly recommended"
}
{
        "_id" : ObjectId("5ecdb3c40bc7c93da9a81bb9"),
        "id" : 1871,
        "url" : "http://www.tvmaze.com/shows/1871/mr-robot",
        "name" : "Mr. Robot",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Drama",
                "Crime",
                "Thriller"
        ],
        "status" : "Ended",
        "runtime" : 60,
        "premiered" : "2015-06-24",
        "officialSite" : "http://www.usanetwork.com/mrrobot",
        "schedule" : {
                "time" : "22:00",
                "days" : [
                        "Sunday"
                ]
        },
        "rating" : {
                "average" : 8.4
```

```
        "rating" : {
                "average" : 8.4
        },
        "weight" : 98,
        "network" : {
                "id" : 30,
                "name" : "USA Network",
                "country" : {
                        "name" : "United States",
                        "code" : "US",
                        "timezone" : "America/New_York"
                }
        },
        "recommendation" : "highly recommended"
}
MongoDB Enterprise >
```

7. READ. Finds all episodes with a runtime between 60 and 70 mins and also air at 10pm but ignores the first 10: db.Episodes.find({$and: [{"runtime":{$gte:60,$lt:70}},{"airtime":"22:00"}]}).skip(10).pretty()


```
MongoDB Enterprise > db.Episodes.find({$and: [{"runtime":{$gte:60,$lt:70}},{"airtime":"22:00"}]}).skip(10).pretty()
{
        "_id" : ObjectId("5ecdb41da0de960ad884a514"),
        "id" : 830877,
        "url" : "http://www.tvmaze.com/episodes/830877/mr-robot-2x09-eps27init5fve",
        "name" : "eps2.7_init5.fve",
        "season" : 2,
        "number" : 9,
        "airdate" : "2016-08-31",
        "airtime" : "22:00",
        "airstamp" : "2016-09-01T02:00:00+00:00",
        "runtime" : 65,
        "image" : {
                "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/73/182686.jpg",
                "original" : "http://static.tvmaze.com/uploads/images/original_untouched/73/182686.jpg"
        },
        "summary" : "<p>Angela wants more from E Corp than they want to give; and Elliot and Darlene seek answers.</p>",
        "_links" : {
                "self" : {
                        "href" : "http://api.tvmaze.com/episodes/830877"
                }
        }
}
{
        "_id" : ObjectId("5ecdb41da0de960ad884a515"),
        "id" : 830878,
        "url" : "http://www.tvmaze.com/episodes/830878/mr-robot-2x10-eps28h1dden-pr0cessaxx",
        "name" : "eps2.8_h1dden-pr0cess.axx",
        "season" : 2,
        "number" : 10,
        "airdate" : "2016-09-07",
        "airtime" : "22:00",
        "airstamp" : "2016-09-08T02:00:00+00:00",
        "runtime" : 65,
        "image" : {
                "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/73/184601.jpg",
                "original" : "http://static.tvmaze.com/uploads/images/original_untouched/73/184601.jpg"
        },
        "summary" : "<p>Elliot wonders if Mr. Robot has been lying to him; Darlene attempts to do the right thing; Dom and the FBI get closer.</p>",
        "_links" : {
                "self" : {
                        "href" : "http://api.tvmaze.com/episodes/830878"
                }
        }
}
{
        "_id" : ObjectId("5ecdb41da0de960ad884a516"),
        "id" : 167387,
        "url" : "http://www.tvmaze.com/episodes/167387/mr-robot-1x10-eps19zer0-dayavi",
        "name" : "eps1.9_zer0-day.avi",
        "season" : 1,
        "number" : 10,
        "airdate" : "2015-09-02",
        "airtime" : "22:00",
        "airstamp" : "2015-09-03T02:00:00+00:00",
        "runtime" : 60,
        "image" : {
                "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/107/267554.jpg",
                "original" : "http://static.tvmaze.com/uploads/images/original_untouched/107/267554.jpg"
        },
        "summary" : "<p>In the Season 1 finale, Mr. Robot and Tyrell are MIA; and a past hack haunts Elliot.</p>",
        "_links" : {
                "self" : {
```

```
                "self" : {
                        "href" : "http://api.tvmaze.com/episodes/167387"
                }
        }
}
{
        "_id" : ObjectId("5ecdb41da0de960ad884a517"),
        "id" : 57361,
        "url" : "http://www.tvmaze.com/episodes/57361/better-call-saul-1x02-mijo",
        "name" : "Mijo",
        "season" : 1,
        "number" : 2,
        "airdate" : "2015-02-09",
        "airtime" : "22:00",
        "airstamp" : "2015-02-10T03:00:00+00:00",
        "runtime" : 60,
        "image" : {
                "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/8/21007.jpg",
                "original" : "http://static.tvmaze.com/uploads/images/original_untouched/8/21007.jpg"
        },
        "summary" : "<p>As his troubles escalate to a boiling point, Jimmy finds himself in dire straits. An act of carelessness puts Chuck at risk.</p>",
        "_links" : {
                "self" : {
                        "href" : "http://api.tvmaze.com/episodes/57361"
                }
        }
}
{
        "_id" : ObjectId("5ecdb41da0de960ad884a519"),
        "id" : 109590,
        "url" : "http://www.tvmaze.com/episodes/109590/better-call-saul-1x03-nacho",
        "name" : "Nacho",
        "season" : 1,
        "number" : 3,
        "airdate" : "2015-02-16",
        "airtime" : "22:00",
        "airstamp" : "2015-02-17T03:00:00+00:00",
        "runtime" : 60,
        "image" : {
                "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/8/21771.jpg",
                "original" : "http://static.tvmaze.com/uploads/images/original_untouched/8/21771.jpg"
        },
        "summary" : "<p>Jimmy pulls out all the stops to prove that a dangerous client is innocent, even though it causes some friction with Kim.<br></p>",
        "_links" : {
                "self" : {
                        "href" : "http://api.tvmaze.com/episodes/109590"
                }
        }
}
{
        "_id" : ObjectId("5ecdb41da0de960ad884a51a"),
        "id" : 109591,
        "url" : "http://www.tvmaze.com/episodes/109591/better-call-saul-1x04-hero",
        "name" : "Hero",
        "season" : 1,
        "number" : 4,
        "airdate" : "2015-02-23",
        "airtime" : "22:00",
        "airstamp" : "2015-02-24T03:00:00+00:00",
        "runtime" : 60,
        "image" : {
                "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/8/22220.jpg",
                "original" : "http://static.tvmaze.com/uploads/images/original_untouched/8/22220.jpg"
```


…

8. CREATE: Inserts two new aired episodes of the show Better Call me Saul with relevant information: db.Episodes.insertMany([{"id":109598,"showname":"Better Call Saul", "url":"http://www.tvmaze.com/episodes/109598/better-call-saul",

"season":1,"number":11, "airdate":"2015-04-16","runtime":60 }, {"id":109599, "showname":"Better Call Saul", "url":"http://www.tvmaze.com/episodes/109599/better-call-saul","season":1, "number":12, "airdate":"2015-04-23", "runtime":60}])

Before insert:

```
MongoDB Enterprise > db.Episodes.find({$or:[{id:109598}, {id:109599}]}).pretty()
MongoDB Enterprise >
```

After insert:

```
MongoDB Enterprise > db.Episodes.insertMany([{"id":109598,"showname":"Better Call Saul",  "url":"http://www.tvmaze.com/episodes/109598/better-call-saul", "season":1,"number":11, "airdate":"2015-04-16","runtime":60 }, {"id":109599, "showname":"Better Call Saul", "url":"http://www.tvmaze.com/episodes/109599/better-call-saul","season":1, "number":12, "airdate":"2015-04-23", "runtime":60}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5ed297a6ab46006c481ee4c1"),
                ObjectId("5ed297a6ab46006c481ee4c2")
        ]
}
MongoDB Enterprise > db.Episodes.find({$or:[{id:109598}, {id:109599}]}).pretty()
{
        "_id" : ObjectId("5ed297a6ab46006c481ee4c1"),
        "id" : 109598,
        "showname" : "Better Call Saul",
        "url" : "http://www.tvmaze.com/episodes/109598/better-call-saul",
        "season" : 1,
        "number" : 11,
        "airdate" : "2015-04-16",
        "runtime" : 60
}
{
        "_id" : ObjectId("5ed297a6ab46006c481ee4c2"),
        "id" : 109599,
        "showname" : "Better Call Saul",
        "url" : "http://www.tvmaze.com/episodes/109599/better-call-saul",
        "season" : 1,
        "number" : 12,
        "airdate" : "2015-04-23",
        "runtime" : 60
}
MongoDB Enterprise >
```

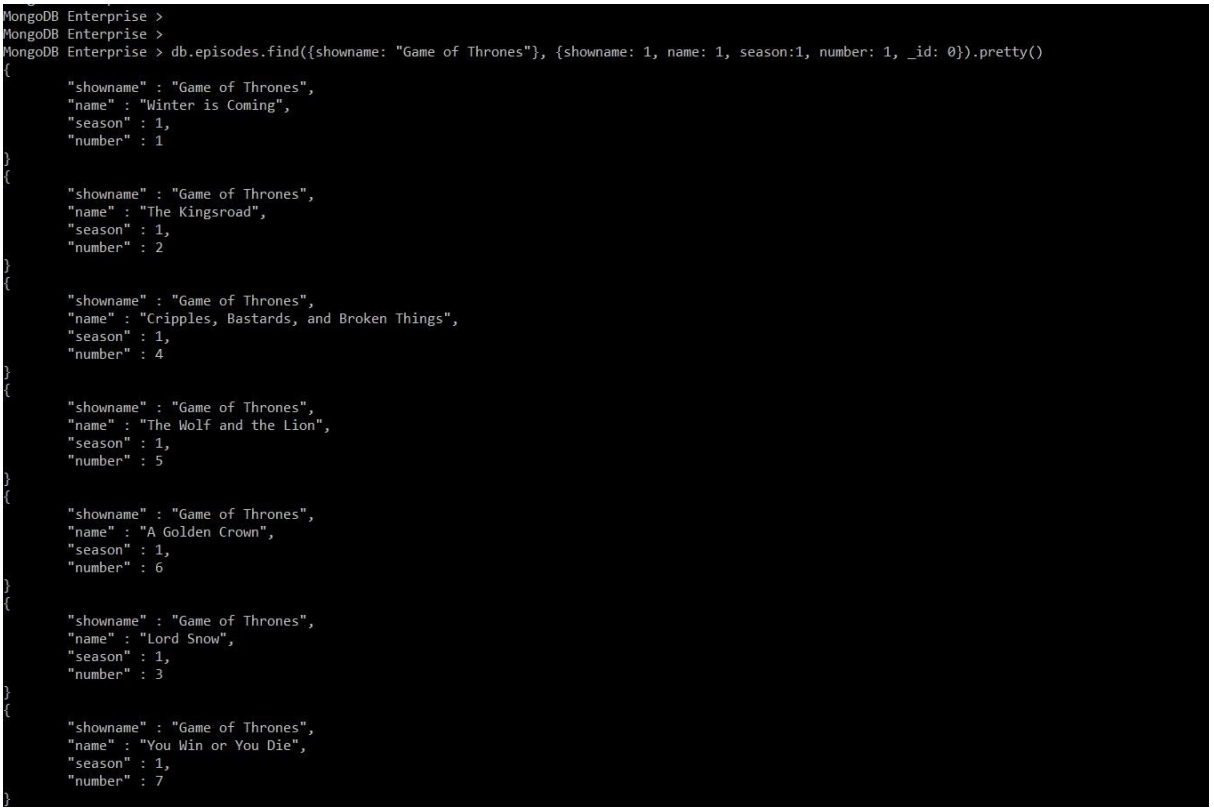
9. READ: Shows the name and the accumulation of runtime of all episodes that aired on 16 Feb 2015: db.Episodes.aggregate([{$match:{airdate:"2015-02-16"}} , {$group: {_id:"$name", totalscreening:{$sum:"$runtime"}}}]).pretty()

```
MongoDB Enterprise > db.Episodes.aggregate([ {$match:{airdate:"2015-02-16"}} , {$group: {_id:"$name", totalscreening:{$sum:"$runtime"}}}]).pretty()
{ "_id" : "Nacho", "totalscreening" : 60 }
MongoDB Enterprise >
```

10. READ: Display only names of shows that are comedy: db.shows.find({genres: "Comedy"}, {name: 1, _id:0}).pretty()

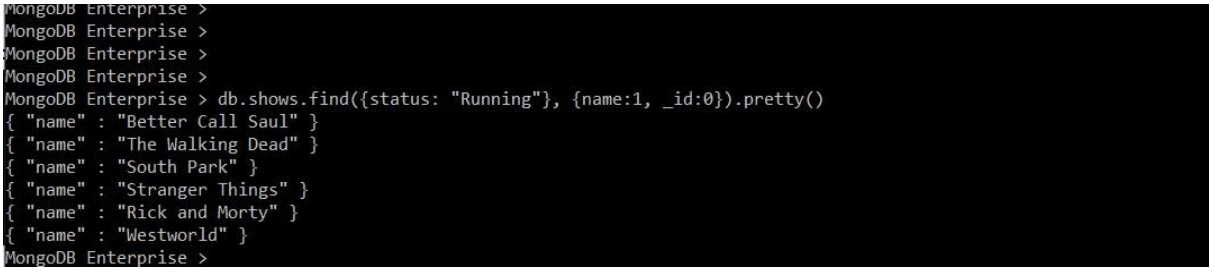
```
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.shows.find({genres: "Comedy"}, {episodes:0, network:0, _id:0}).pretty()
{
        "id" : 143,
        "url" : "http://www.tvmaze.com/shows/143/silicon-valley",
        "name" : "Silicon Valley",
        "type" : "Scripted",
        "language" : "English",
        "genres" : [
                "Comedy"
        ],
        "status" : "Ended",
        "runtime" : 30,
        "premiered" : "2014-04-06",
        "officialSite" : "http://www.hbo.com/silicon-valley/",
        "schedule" : {
                "time" : "22:00",
                "days" : [
                        "Sunday"
                ]
        },
        "rating" : {
                "average" : 8.5
        },
        "weight" : 96
}
{
        "id" : 112,
        "url" : "http://www.tvmaze.com/shows/112/south-park",
        "name" : "South Park",
        "type" : "Animation",
        "language" : "English",
        "genres" : [
                "Comedy"
        ],
        "status" : "Running",
        "runtime" : 30,
        "premiered" : "1997-08-13",
```

11. READ: Return the episodes of a particular show: db.episodes.find({showname: "Game of Thrones"}, {showname: 1, name: 1, season:1, number: 1, _id: 0}).pretty()



```
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.episodes.find({showname: "Game of Thrones"}, {showname: 1, name: 1, season:1, number: 1, _id: 0}).pretty()
{
        "showname" : "Game of Thrones",
        "name" : "Winter is Coming",
        "season" : 1,
        "number" : 1
}
{
        "showname" : "Game of Thrones",
        "name" : "The Kingsroad",
        "season" : 1,
        "number" : 2
}
{
        "showname" : "Game of Thrones",
        "name" : "Cripples, Bastards, and Broken Things",
        "season" : 1,
        "number" : 4
}
{
        "showname" : "Game of Thrones",
        "name" : "The Wolf and the Lion",
        "season" : 1,
        "number" : 5
}
{
        "showname" : "Game of Thrones",
        "name" : "A Golden Crown",
        "season" : 1,
        "number" : 6
}
{
        "showname" : "Game of Thrones",
        "name" : "Lord Snow",
        "season" : 1,
        "number" : 3
}
{
        "showname" : "Game of Thrones",
        "name" : "You Win or You Die",
        "season" : 1,
        "number" : 7
}
```

12. READ: Display all the shows that are running currently: db.shows.find({status: "Running"}, {name:1, _id:0}).pretty()U



```
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.shows.find({status: "Running"}, {name:1, _id:0}).pretty()
{ "name" : "Better Call Saul" }
{ "name" : "The Walking Dead" }
{ "name" : "South Park" }
{ "name" : "Stranger Things" }
{ "name" : "Rick and Morty" }
{ "name" : "Westworld" }
MongoDB Enterprise >
```

13. Update the airdate fields of all episodes from string to ISO data type: db.episodes.updateMany({}, [{$set: {airdate: {$dateFromString: {dateString: '$airdate'}}}}])

```
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.episodes.updateMany({}, [{$set: {airdate: {$dateFromString: {dateString: '$airdate'}}}} ])
{ "acknowledged" : true, "matchedCount" : 296, "modifiedCount" : 296 }
MongoDB Enterprise > db.episodes.findOne().pretty()
2020-05-31T19:01:29.955+0200 E  QUERY    [js] uncaught exception: TypeError: db.episodes.findOne(...).pretty is not a function :
@(shell):1:1
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.episodes.findOne()
{
        "_id" : ObjectId("5ed26e7f58bb5b7c2356fa5c"),
        "id" : 167381,
        "showname" : "Mr. Robot",
        "url" : "http://www.tvmaze.com/episodes/167381/mr-robot-1x04-eps13da3m0nsmp4",
        "name" : "eps1.3_da3m0ns.mp4",
        "season" : 1,
        "number" : 4,
        "airdate" : ISODate("2015-07-15T00:00:00Z"),
        "airtime" : "22:00",
        "airstamp" : "2015-07-16T02:00:00+00:00",
        "runtime" : 60,
        "image" : {
                "medium" : "http://static.tvmaze.com/uploads/images/medium_landscape/106/267080.jpg",
                "original" : "http://static.tvmaze.com/uploads/images/original_untouched/106/267080.jpg"
        },
        "summary" : "<p>At fsociety, Elliot lays out his plan to hack Steel Mountain's climate control system and raise the heat high enough to melt all the tapes that record the data. Romero points out the pla
's many flaws but when Elliot remembers that they only have a few days left before Evil Corp switches over to their new redundant security protocols based on what Tyrell Wellick told him, they reluctantly agree
it's now or never.</p>",
        "_links" : {
                "self" : {
                        "href" : "http://api.tvmaze.com/episodes/167381"
                }
        }
}
MongoDB Enterprise >
```

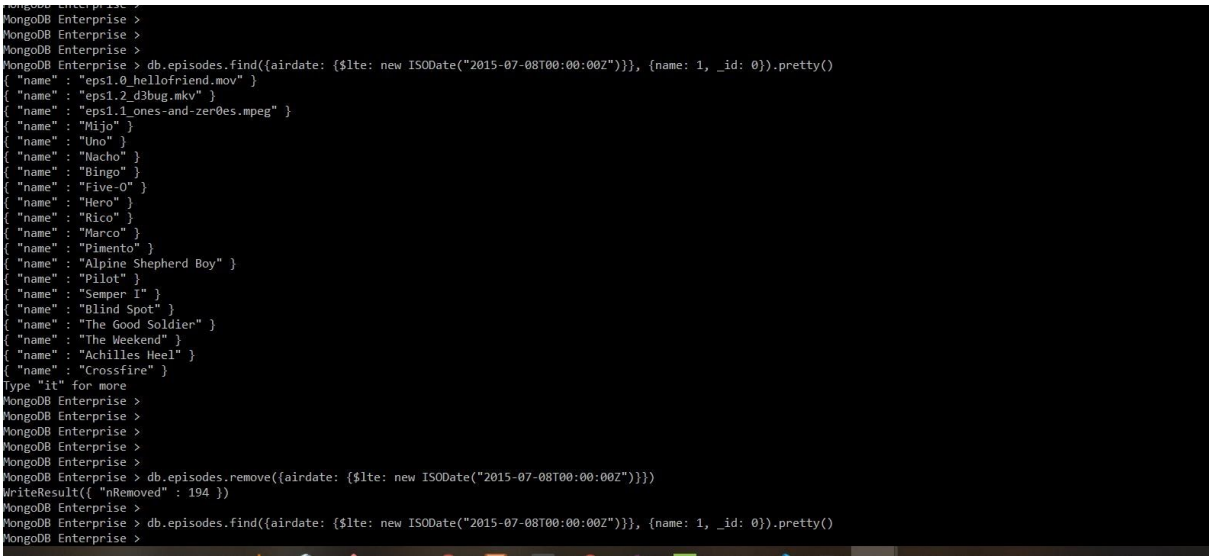
14. READ: Display all the shows that have more than one air day:

```
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.shows.find({$where: "this.schedule.days.length>1"}, {name: 1, _id:0}).pretty()
{ "name" : "Better Call Saul" }
{ "name" : "The Walking Dead" }
{ "name" : "Stranger Things" }
{ "name" : "Rick and Morty" }
{ "name" : "Westworld" }
MongoDB Enterprise >
```

15. DELETE: Delete Narcos from both episodes and shows (with two queries): db.shows.remove({name: "Narcos"}), db.episodes.remove({name: "Narcos"})

```
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.shows.remove({name: "Narcos"})
WriteResult({ "nRemoved" : 1 })
MongoDB Enterprise >
MongoDB Enterprise > db.episodes.remove({showname: "Narcos"})
WriteResult({ "nRemoved" : 20 })
MongoDB Enterprise >
MongoDB Enterprise >
MongoDB Enterprise > db.shows.find({name: "Narcos"}).pretty()
MongoDB Enterprise > db.episodes.find({name: "Narcos"}).pretty()
MongoDB Enterprise >
```

16. DELETE: Delete all episodes that aired before a certain date i.e old episodes: db.episodes.remove({airdate: {$lte: new ISODate("2015-07-08T00:00:00Z")}})



6. Java code is in the folder MongodbJavaApplication. The README File also illustrates how to run the code. The main code is in the **src folder** of the folder. In the **src open the main folder** and then open the **java folder to find the code**. The output text file is the **Mongodboutput.txt**

7. Division of work

Student Name	**Work**
Willie Macharia	Question 1, Question 2, Question 4, Question 6
Blessed Chitamba	Question 1, Question 4, Question 5,Question 6
Mandisa Baleni	Question 1, Question 3, Question 4, Question 5,Question 6
Mapule Madzena	Question 1, Question 4, Question 5,Question 6