CSCI 620

ASSIGNMENT 6

Abhishek Shah, as5553

Question 1

Extra-data.json

There are altogether roughly 232.k records in the extra-data.json file.

Each row in extra-data.json consists of the following keys: box_office_currencylabel, MPAA_film_ratingLabel, titlelabel, IMDb_ID, cost, distributorLabel and box_office.

Each key contains multiple fields and describes more about a movie.

Here consider record as a particular movie.

box_office_currencylabel contains,

1. xml:lang which just contains the language of the record.

Eg: "en" for English

2. type which is the type of data we have.

Eg: "literal" for literals

3. value represent the type of currency for that particular record.

Eg: "United State dollars", "Indian Rupee", "euro", etc.

There are altogether 14 different type of currency in the data.

MPAA_film_ratingLabel contains,

1. xml:lang which just contains the language of the record.

Eg: "en" for English

2. type which is the type of data we have.

Eg: "literal" for literals

3. value contains the rating label for that particular movie.

Eg: "PG-13", "NC-17", etc.

There are altogether 7 different type of ratings available.

titlelabel contains,

1. type which is the type of data we have.

Eg: "literal" for literals

2. value represents the name of the title for that particular record.

There are duplicate and null values over here.

IMDb_ID contains,

1. type which is the type of data we have.

Eg: "literal" for literals

2. value is nothing but IMDB id's of the movies.

cost contains,

1. Datatype which tells us the datatype of the values that cost contains.

Eg: "Decimal", "Int"

2. type which is the type of data we have.

Eg: "literal" for literals

3. value is the cost of the movie.

distributorLabel contains,

1. xml:lang which just contains the language of the record.

Eg: "en" for English

2. type which is the type of data we have.

Eg: "literal" for literals

3. value represents the name of the distributor label.

eg: "Intercom", "Lal Jose", "Barton Films", etc

It has 2285 distributors altogether.

box_office contains,

1. Datatype which tells us the datatype of the values that cost contains.

Eg: "Decimal", "Int"

2. type which is the type of data we have.

Eg: "literal" for literals

3. value represents the value the movie earned at the box office.

Looking at the data, we'll be only dealing with the "value" attribute from all the 7 keys that we have in the data.

Question 2

-- importing extra-data json to to mongoDb (Terminal)

mongoimport --db imdb620 --collection Extra --file /Users/abhishekshah/Downloads/extra-data.json

-- Removing unwanted fields (MongoDB)

```
db.Extra.updateMany({}, {$unset: {"box_office_currencyLabel.xml:lang": ""}})
db.Extra.updateMany({}, {$unset: {"box_office_currencyLabel.type": ""}})
db.Extra.updateMany({}, {$unset: {"MPAA_film_ratingLabel.xml:lang": ""}})
db.Extra.updateMany({}, {$unset: {"MPAA_film_ratingLabel.type": ""}})
db.Extra.updateMany({}, {$unset: {"titleLabel.type": ""}})
db.Extra.updateMany({}, {$unset: {"IMDb_ID.type": ""}})
db.Extra.updateMany({}, {$unset: {"cost.datatype": ""}})
db.Extra.updateMany({}, {$unset: {"cost.type": ""}})
db.Extra.updateMany({}, {$unset: {"distributorLabel.xml:lang": ""}})
db.Extra.updateMany({}, {$unset: {"distributorLabel.type": ""}})
db.Extra.updateMany({}, {$unset: {"box_office.datatype": ""}})
db.Extra.updateMany({}, {$unset: {"box_office.type": ""}})
```

-- Removing 'tt' (MongoDB)

```
db.Extra.updateMany(
 {"IMDb_ID.value":{$regex:/tt/}},
 [{
  $set:{"IMDb_ID.value":{
   $replaceOne:{input:"$IMDb_ID.value",find:"tt",replacement:""}
  }
 }}
 ])
```

```
-- Removing 'nm' (MongoDB)
db.Extra.updateMany(
  {"IMDb_ID.value":{$regex:/nm/}},
  [{
    $set:{"IMDb_ID.value":{
      $replaceOne:{input:"$IMDb_ID.value",find:"nm",replacement:""}
    }
  }}
  ])

-- Converting IMDb_ID.value to int (MongoDB)
db.Extra.updateMany({},[{ $set:{"IMDb_ID.value":{$toInt:"$IMDb_ID.value"}}}])

-- Converting cost.value to float (MongoDB)
db.Extra.updateMany({},
[{
  $set:{"cost.value":{$toDouble:"$cost.value"}}
}])

-- Converting box_office.value to float (MongoDB)
db.Extra.updateMany({},[{ $set:{"box_office.value":{$toDouble:"$box_office.value"}}}])

-- Removing null values (MongoDB)
db.Extra.updateMany({"box_office_currencyLabel.value":null},{$unset:
{"box_office_currencyLabel.value":1}})
db.Extra.updateMany({"MPAA_film_ratingLabel.value":null},{$unset:
{"MPAA_film_ratingLabel.value":1}})
db.Extra.updateMany({"titleLabel.value":null},{$unset:{"titleLabel.value":1}})
db.Extra.updateMany({"IMDb_ID.value":null},{$unset:{"IMDb_ID.value":1}})
```

```
db.Extra.updateMany({"cost.value":null},{$unset:{"cost.value":1}})

db.Extra.updateMany({"distributorLabel.value":null},{$unset:{"distributorLabel.value":1}})

db.Extra.updateMany({"box_office.value":null},{$unset:{"box_office.value":1}})


-- exporting Extra.json as csv to clean data (Terminal)

mongoexport --host localhost --db imdb620 --collection Extra --type=csv --out
/Users/abhishekshah/Downloads/test.csv --fields
IMDb_ID.value,box_office_currencyLabel.value,MPAA_film_ratingLabel.value,titleLabel.value,cost.v
alue,distributorLabel.value,box_office.value


-- renaming and removing duplicates (PostgresSQL)

create table extra(
        _id integer,
        box_office_currencyLabel varchar(255),
        MPAA_film_ratingLabel varchar(255),
        titleLabel text,
        cost float,
        distributorLabel varchar(255),
        box_office float
);

COPY extra FROM '/Users/abhishekshah/Downloads/test.csv' DELIMITER ',' CSV HEADER;


create table tempextra(
        _id integer,
        box_office_currencyLabel varchar(255),
        MPAA_film_ratingLabel varchar(255),
        titleLabel text,
        cost float,
        distributorLabel varchar(255),
        box_office float
```

```
);

insert into tempextra
select distinct _id, box_office_currencyLabel, MPAA_film_ratingLabel,
titleLabel, cost, distributorLabel, box_office
from extra
where box_office_currencyLabel = 'United States dollar';

-- removing extra " from some columns
update tempextra set titleLabel = replace(titleLabel, "", '');

COPY (
SELECT json_strip_nulls(row_to_json(res))
FROM(
SELECT _id, box_office_currencyLabel, MPAA_film_ratingLabel, titleLabel,
        cost, distributorLabel, box_office
FROM tempextra
)res)
TO '/Users/abhishekshah/Downloads/extra.json' with (FORMAT TEXT, HEADER false);

-- importing to json (Terminal)
mongoimport --db imdb620 --collection Extra1 --file /Users/abhishekshah/Downloads/extra.json

-- Merge two collections (MongoDB)
db.Extra1.aggregate({
  $merge: { into: "Movies", on: "_id"}
});
```

```
-- Check for updating count (MongoDB)
db.Movies.find({
  $and:[
    {"box_office_currencylabel":{$exists:true}},
    {"mpaa_film_ratinglabel":{$exists:true}},
    {"cost":{$exists:true}},
    {"distributorlabel":{$exists:true}},
    {"box_office":{$exists:true}},
    ]
}).count()
-- 181
```

Updating count: 181 records

That is 181 records were updated successfully in Movies collection.

Here, a record is considered updated if and only if that record now has additional four fields namely, box_office_currencylabel, mpaa_film_ratinglabel, cost, distributorlabel and box_office.

Question 3

If the documents in the file do not contain the IMDb ID's, I would perform the matching process using title(Movies) or TitleLabel.value(extra-data). We'll treat title as the key on which we will merge the data. That column is the only relevant column on which we'll be able to merge our two collections.

We will remove the already assigned/decided _id column from both the collections. Then, we'll make the title column in Movies collection and titleLabel.value in extra-data as _id for us to merge on.

title in Movies and TitleLabel.value just needs to be renamed to _id after we remove the original _id from both the collection.

I also made sure to delete the duplicates for them, as the data then needs to be imported to mongodb as well.

-- Using the csv(test.csv) of extra-data.json (Collection: Extra) to drop IMDb_value,
-- rename columns (Python/Pandas) and make titleLabel.value as _id

```
import pandas as pd

# reading csv files

df = pd.read_csv('/Users/abhishekshah/Downloads/test.csv')

# removing IMDb_ID.value

del df['IMDb_ID.value']

# renaming columns

df.rename(columns={'box_office_currencyLabel.value': 'box_office_currencyLabel',

                   'MPAA_film_ratingLabel.value': 'MPAA_film_ratingLabel',

                   'titleLabel.value': '_id',

                   'cost.value': 'cost',

                   'distributorLabel.value': 'distributorLabel',

                   'box_office.value': 'box_office'}, inplace=True)



# drop duplicates

df.drop_duplicates(subset=["_id"], inplace=True)


# exporting to json

df.to_json(r'/Users/abhishekshah/Downloads/testwithtitle.json', orient='records', lines=True)
```

-- importing converted file to mongodb as Extra2 (Terminal)

mongoimport --db imdb620 --collection Extra2 --file /Users/abhishekshah/Downloads/testwithtitle.json

-- Removing nulls (MongoDB)

db.Extra2.updateMany({"box_office_currencyLabel":null},{$unset:{"box_office_currencyLabel":1}})

db.Extra2.updateMany({"MPAA_film_ratingLabel":null},{$unset:{"MPAA_film_ratingLabel":1}})

db.Extra2.updateMany({"cost":null},{$unset:{"cost":1}})

db.Extra2.updateMany({"distributorLabel":null},{$unset:{"distributorLabel":1}})

db.Extra2.updateMany({"box_office":null},{$unset:{"box_office":1}})

-- Using the Movies.json(Assignment 4) to drop _id and make title as _id (Python/Pandas)

```
import pandas as pd

# reading json file
df = pd.read_json('/Users/abhishekshah/Downloads/Movies.json', lines=True)

# deleting column _id
del df['_id']

# renaming column
df.rename(columns={'title': '_id'}, inplace=True)

# dropping duplicates
df.drop_duplicates(subset=["_id"], inplace=True)

# export to json format
df.to_json(r'/Users/abhishekshah/Downloads/testwithmovies.json', orient='records', lines=True)
```

-- importing converted file to mongoldb as Movies2 (Terminal)

mongoimport --db imdb620 --collection Movies2 --file /Users/abhishekshah/Downloads/testwithmovies.json

```
-- Removing Nulls (MongoDB)
db.Movies2.updateMany({"box_office_currencyLabel":null},{$unset:
{"box_office_currencyLabel":1}})
db.Movies2.updateMany({"MPAA_film_ratingLabel":null},{$unset:{"MPAA_film_ratingLabel":1}})
db.Movies2.updateMany({"cost":null},{$unset:{"cost":1}})
db.Movies2.updateMany({"distributorLabel":null},{$unset:{"distributorLabel":1}})
db.Movies2.updateMany({"box_office":null},{$unset:{"box_office":1}})

-- Merging two collections (MongoDB)
-- Here _id is actually title
db.Extra2.aggregate({
  $merge: { into: "Movies2", on: "_id"}
});

-- Updating Count (MongoDB)
db.Movies2.find({
  $and:[
    {"box_office_currencyLabel":{$exists:true}},
    {"MPAA_film_ratingLabel":{$exists:true}},
    {"cost":{$exists:true}},
    {"distributorLabel":{$exists:true}},
    {"box_office":{$exists:true}},
    ]
}).count()
-- 187
```

Updating count: 187 records

That is 187 records existing documents in Movies2.

Here, a record is considered updated if and only if that record now has additional four fields namely, box_office_currencyLabel, MPAA_film_ratingLabel, cost, distributorLabel and box_office.

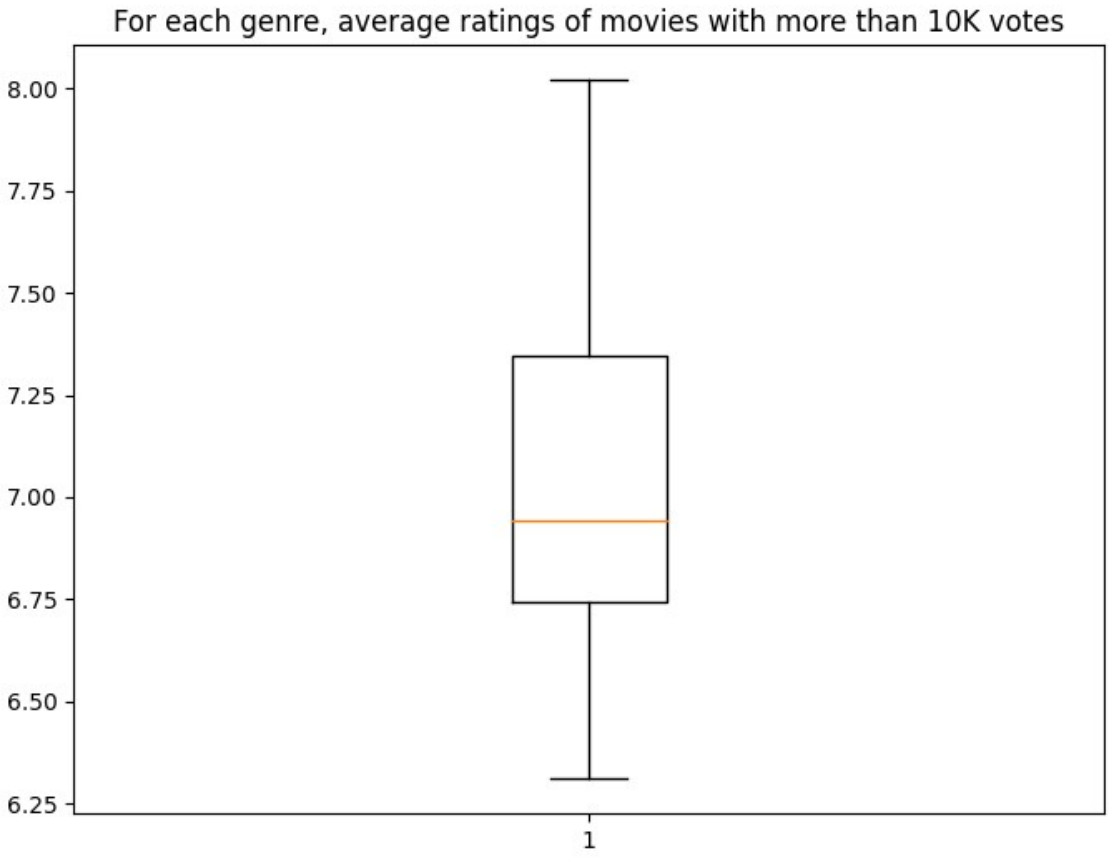
Question 4

```
import pymongo
import matplotlib.pyplot as plt
import numpy as np
# connecting to mongodb server

client = pymongo.MongoClient("mongodb://localhost:27017/")
database = client["imdb620"]
collection = database['Movies']

# code for query 1
def query_one(col):
    """
    There are altogether 28 unique genres for our database.
    So we'll actually need to plot 28 different graphs for each genre.
    However, for representation purpose, I am just displaying in a single graph.
    Also, we don't need to calculate five point summary for plotting.
    plt.boxplot() by default plots the graph by calculating
    the 5 point summary of the array provided to the function plt.boxplot().
    """
    query = col.aggregate([{"$unwind": "$genres"},
                           {"$match": {"numvotes": {"$gt": 10000}}},
                           {"$group": {"_id": "$genres", "avg": {"$avg":
"$avgrating"}}}
                           ])
    rating = []
    # aggregate gives result as a dictionary.
    # converting dictionary values to a list
    for result in query:
        rating.append(result["avg"])
    print("Five point summary:")
    print("Min: ", np.min(rating))
    print("Max: ", np.max(rating))
    print("First Quartile: ", np.std(rating))
    print("Third Quartile: ", np.mean(rating))
    print("Median: ", np.median(rating))
    plt.figure(figsize=(8, 6))
    plt.boxplot(rating)
    plt.title("For each genre, average ratings of movies with more than 10K votes")
    plt.savefig('query1.png')
    plt.show()

# code for query 2
def query_two(col):
    """
    Here, I am trying to find,
    (Total number of actors)/(Total number of movies) per genre.
    But I am unable to get anything.
    """
    query = col.aggregate([{"$match": {"actors": {"$ne":None}}},
```

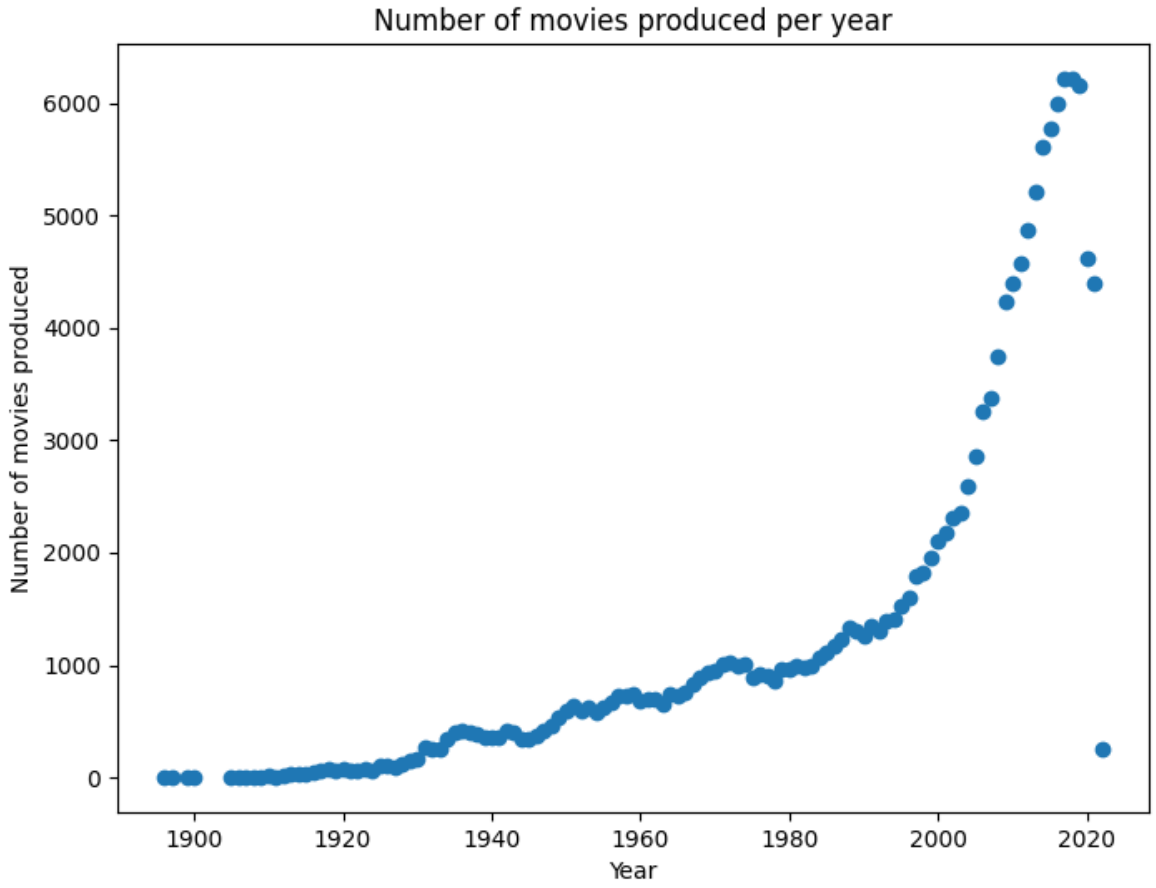
```
                            {"$unwind": "$actors"},
                            {"$unwind": "$genres"},
                            {"$group": {"_id": {"allmovies": "$_id", "allgenres":
"$genres"},
                                        "totalactors": {"$sum": 1}}},
                            {"$group": {"_id": "$_id.allgenres", "avg": {"$avg":
"$totalactors"}}}
                            ])
    x = []
    y = []
    # aggregate gives result as a dictionary.
    # converting dictionary values to a list
    for result in query:
        x.append(result["_id"])
        y.append(result["avg"])
    plt.figure(figsize=(8, 6))
    plt.bar(x, y)
    plt.xlabel("Genres")
    plt.ylabel("Average number of actors")
    plt.title("Average number of actors per movie by genre")
    plt.savefig('query2.png')
    plt.show()


# code for query 3
def query_three(col):
    query = col.aggregate([{"$match": {"producers": {"$ne": None},
                                       "startyear": {
                                           "$ne": None}, "type": "movie"}},
                            {"$group": {"_id": "$startyear", "no_of_movies": {"$sum":
1}}}
                            ])
    x = []
    y = []
    # aggregate gives result as a dictionary.
    # converting dictionary values to a list
    for result in query:
        x.append(result["_id"])
        y.append(result["no_of_movies"])
    plt.figure(figsize=(8, 6))
    plt.scatter(x, y)
    plt.xlabel("Year")
    plt.ylabel("Number of movies produced")
    plt.title("Number of movies produced per year")
    plt.savefig('query3.png')
    plt.show()


if __name__ == "__main__":
    query_one(collection)
    query_two(collection)
    query_three(collection)
```

Running the program above, produces following three graphs.

QUERY 1



QUERY 3


Number of movies produced per year
Number of movies produced
Year
0
1000
2000
3000
4000
5000
6000
1900
1920
1940
1960
1980
2000
2020