Homework Assignment 03

DS 4300 - Spring 2025

- **EC Due Date**: Feb 16, 2025 @ 11:59pm
- **Regular Due Date**: Feb 18, 2025 @ 11:59pm
- Upload to GradeScope (no question/solutions to Match)

Set up your connection to Mongo DB here.

```
import pymongo
from bson.json_util import dumps

uri = "mongodb://sawyerss:Northeastern1!!@localhost:27017/"
client = pymongo.MongoClient(uri)
```

Directions:

- Use the mflix sample database to prepare a pymongo query each of the following prompts.
- Be sure to print the results of your query using the `dumps` function.

```
mflixdb = client.mflix
```

Question 1:

Give the street, city, and zipcode of all theaters in Massachusetts.

```
ma_theaters = mflixdb.theaters.aggregate([
    {"$match": {"location.address.state": "MA"}},
    {"$group": {"_id": {"street": "$location.address.street1", "city": "$location.address.city",
"zipcode": "$location.address.zipcode"}}}, # group by location to get rid of duplicates
    {"$project": {"_id": 0, "street": "$_id.street", "city": "$_id.city", "zipcode": "$_id.zipcode"}} # use
project to make output more readable
])

print('Locations of all theaters in Massachusetts:\n', dumps(ma_theaters, indent = 2))
```

Question 2:

How many theaters are there in each state? Order the output in alphabetical order by 2-character state code.

```
state_counts = mflixdb.theaters.aggregate([
```

```
    {"$group": {"_id": "$location.address.state", "num_theaters": {"$sum": 1}}},
    {"$sort" : {"_id": 1}},
    {"$project": {"_id": 0, "state": "$_id", "num_theaters": "$num_theaters"}} # use project to make
output more readable
])

print('Number of theaters in each state:\n', dumps(state_counts, indent = 2))
```

Question 3:

How many movies are in the Comedy genre?

```
comedy_count = mflixdb.movies.count_documents({"genres": "Comedy"})

print(f'There are {comedy_count} movies in the Comedy genre')
```

Question 4:

What movie has the longest run time? Give the movie's title and genre(s).

```
longest_run_time = mflixdb.movies.aggregate([
    {"$group": {"_id": None, "max_runtime": {"$max": "$runtime"}}},
    {"$lookup": {
        "from": "movies",
        "localField": "max_runtime",
        "foreignField": "runtime",
        "as": "longest_run_time_movie"
    }},
    {"$unwind": "$longest_run_time_movie"}, # use unwind for simpler projection
    {"$project": {"_id": 0, "title": "$longest_run_time_movie.title", "genres":
"$longest_run_time_movie.genres"}}
])

print('Movie with the longest run time:\n', dumps(longest_run_time, indent = 2))
```

Question 5:

Which movies released after 2010 have a Rotten Tomatoes viewer rating of 3 or higher? Give the title of the movies along with their Rotten Tomatoes viewer rating score. The viewer rating score should become a top-level attribute of the returned documents. Return the matching movies in descending order by viewer rating.

```
rotten_tomatoes_movies = mflixdb.movies.aggregate([
    {"$match": {"year": {"$gt": 2010}, "tomatoes.viewer.rating": {"$gte": 3}}},
    {"$sort": {"tomatoes.viewer.rating": -1}},
    {"$project": {"_id": 0, "title": 1, "rotten_tomatoes_viewer_rating_score":
"$tomatoes.viewer.rating"}}
```

```
])

print('Movies released after 2010 with a Rotten Tomatoes viewer rating of 3 or higher:\n',
dumps(rotten_tomatoes_movies, indent = 2))
```

Question 6:

How many movies released each year have a plot that contains some type of police activity (i.e., plot contains the word "police")? The returned data should be in ascending order by year.

```
police_movies_by_year = mflixdb.movies.aggregate([
   {"$match": {
      "plot": {"$regex": "police", "$options": "i"},
      "year": {"$type": "int"}  # remove non-integer/misformatted years
   }},
   {"$group": {"_id": "$year", "police_activity_plot_count": {"$sum": 1}}},
   {"$sort": {"_id": 1}},
   {"$project": {"_id": 0, "year": "$_id", "police_activity_plot_count":
"$police_activity_plot_count"}} # use project to make output more readable
])

print('Number of movies released each year with a plot that contains some type of police
activity:\n',
     dumps(police_movies_by_year, indent = 2, ensure_ascii = True))
```

Question 7:

What is the average number of imdb votes per year for movies released between 1970 and 2000 (inclusive)? Make sure the results are order by year.

```
average_votes_per_year = mflixdb.movies.aggregate([
   {"$match": {
      "year": {"$gte": 1970, "$lte": 2000},
      "imdb.votes": {"$ne": ""} # filter out movies that do not have imdb.votes
   }},
   {"$group": {
      "_id": "$year", "avg_num_imdb_votes": {"$avg": "$imdb.votes"}
   }},
   {"$sort" : {"_id": 1}},
   {"$project": {"_id": 0, "release_year": "$_id", "avg_num_imdb_votes":
"$avg_num_imdb_votes"}} # use project to make output more readable
])

print('Average number of IMDb votes per year for movies released between 1970 and 2000
(inclusive):\n',
     dumps(average_votes_per_year, indent = 2, ensure_ascii = True))
```

Question 8:

What distinct movie languages are represented in the database? You only need to provide the list of languages.

```
movie_languages = mflixdb.movies.distinct("languages")
stripped_movie_languages = [language.strip() for language in movie_languages] # remove excess whitespace

print('Distinct movie languages represented in the database:\n', dumps(stripped_movie_languages, indent = 2))
```