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://drew:123@localhost:27017/"
client = pymongo.MongoClient(uri)
mflixdb = client.mflix
```

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.

```
Start coding or generate with AI.
```

Question 1:

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

```
# Query
theaters_in_ma = mflixdb.theaters.find({"location.address.state": "MA"},
    {"street":"$location.address.street1", "city":"$location.address.city", "zip
)

# Output
print(dumps(theaters_in_ma, indent=3))
```

```
    "city": "Holyoke",
    "zipcode": "01040"
},
{
    "street": "700 Providence Hwy",
    "city": "Dedham",
    "zipcode": "02026"
},
{
    "street": "1337 S Washington St",
    "city": "North Attleboro",
    "zipcode": "02760"
},
{
    "street": "1 Worcester Rd",
    "city": "Framingham",
    "zipcode": "01701"
},
{
    "street": "100 Cambridgeside Pl",
    "city": "Cambridge",
    "zipcode": "02141"
},
{
    "street": "230 Independence Way",
    "city": "Danvers",
    "zipcode": "01923"
},
{
    "street": "550 Grossman Dr",
    "city": "Braintree",
    "zipcode": "02184"
},
{
    "street": "7 Neponset St",
    "city": "Worcester",
    "zipcode": "01606"
},
{
    "street": "793 Iyannough Rd",
    "city": "Hyannis",
    "zipcode": "02601"
},
{
    "street": "550 Arsenal St",
    "city": "Watertown",
    "zipcode": "02472"
},
{
```


Question 2:

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

```
# Query
theaters_by_state = mflixdb.theaters.aggregate([
    {"$group": {"_id": "$location.address.state", "count": {"$sum": 1}}}, {"$sor

# Output
print(dumps(theaters_by_state, indent=1))
```


```
[
 {
  "count": 4,
  "state": "AK"
 },
 {
  "count": 19,
  "state": "AL"
 },
 {
  "count": 16,
  "state": "AR"
 },
 {
  "count": 26,
  "state": "AZ"
 },
 {
  "count": 169,
  "state": "CA"
 },
 {
  "count": 26,
  "state": "CO"
 },
 {
  "count": 21,
  "state": "CT"
 },
 {
  "count": 3,
  "state": "DC"
 },
 {
  "count": 5,
  "state": "DE"
 },
 {
  "count": 111,
  "state": "FL"
 },
 {
  "count": 45,
  "state": "GA"
 },
 {
  "count": 8,
```

```
 "state": "HI"
},
{
 "count": 14,
 "state": "IA"
},
{
 "count": 7,
 "state": "ID"
},
{
```


Question 3:

How many movies are in the Comedy genre?

```
# Query
comedy_movies = mflixdb.movies.aggregate([{"$match": {"genres": "Comedy"}},
    {"$count": "comedy_movies"}])

# Output
print(f"Number of Comedies: {list(comedy_movies)[0]['comedy_movies']}")
```

```
Number of Comedies: 6532
```

Question 4:

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

```
# Query
longest_run_time = mflixdb.movies.find({ }, {"_id": 0, "title": 1, "genres": 1})

# Output
movie = list(longest_run_time)[0]
print(f"Longest Run Time: {movie['title']} \nGenres: {movie['genres']}")
```

```
Longest Run Time: Centennial
Genres: ['Action', 'Adventure', 'Drama']
```

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.

```
# Query
recent_highly_rated = mflixdb.movies.aggregate([{"$match": {"year": {"$gt": 2010
    {"$project": {"_id": 0, "viewer_rating": "$tomatoes.viewer.rating", "title":

# Output
print(dumps(recent_highly_rated, indent=4))
```


```
[
    {
        "title": "Good Ol' Boy",
        "viewer_rating": 5
    },
    {
        "title": "Winds",
        "viewer_rating": 5
    },
    {
        "title": "Beethoven's Christmas Adventure",
        "viewer_rating": 5
    },
    {
        "title": "Scattered Cloud",
        "viewer_rating": 5
    },
    {
        "title": "All Watched Over by Machines of Loving Grace",
        "viewer_rating": 5
    },
    {
        "title": "The Dancer",
        "viewer_rating": 5
    },
    {
        "title": "Wonder Women! The Untold Story of American Superheroines",
        "viewer_rating": 5
    },
    {
        "title": "So It Goes",
        "viewer_rating": 5
    },
    {
        "title": "The Man in the Orange Jacket",
        "viewer_rating": 5
    },
    {
        "title": "Souls of Zen",
        "viewer_rating": 5
    },
    {
        "title": "Seeking Asian Female",
        "viewer_rating": 5
    },
    {
        "title": "Uncanny",
        "viewer_rating": 5
```

```
    },
    {
        "title": "Finsterworld",
        "viewer_rating": 5
    },
    {
        "title": "Letters to Sofija",
        "viewer_rating": 5
    },
```


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.

```
# Query
police_plot = mflixdb.movies.aggregate([{"$match": {"plot": {"$regex": "police",
    {"$group": {"_id": "$year","count": {"$sum": 1}}}, {"$sort": {"_id": 1}},{"$

# Output
print(dumps(police_plot, indent=2))
```


```
[
  {
    "count": 1,
    "year": 1913
  },
  {
    "count": 1,
    "year": 1934
  },
  {
    "count": 1,
    "year": 1935
  },
  {
    "count": 1,
    "year": 1944
  },
  {
    "count": 1,
    "year": 1947
  },
  {
    "count": 2,
    "year": 1948
  },
  {
    "count": 1,
    "year": 1949
  },
  {
```

```
    "count": 2,
    "year": 1950
  },
  {
    "count": 2,
    "year": 1951
  },
  {
    "count": 1,
    "year": 1957
  },
  {
    "count": 1,
    "year": 1958
  },
  {
    "count": 2,
    "year": 1959
  },
  {
    "count": 1,
    "year": 1960
  },
  {
    "count": 1,
    "year": 1961
  },
```


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.

```
# Query
imdb_votes = mflixdb.movies.aggregate([
    {"$match": {"year": {"$gte": 1970, "$lte": 2000}}},
    {"$group": {"_id": "$year", "average_votes": {"$avg": "$imdb.votes"}}},
 {"$sort": {"_id": 1}},{"$project": {"_id": 0, "year":"$_id", "average_votes": 1

# Output
print(dumps(imdb_votes, indent=2))
```


```
[
  {
    "average_votes": 4786.925,
    "year": 1970
  },
  {
    "average_votes": 8528.462264150943,
    "year": 1971
  },
  {
```

```
  "average_votes": 13582.685950413223,
  "year": 1972
},
{
  "average_votes": 14478.785714285714,
  "year": 1973
},
{
  "average_votes": 17602.0,
  "year": 1974
},
{
  "average_votes": 19615.00934579439,
  "year": 1975
},
{
  "average_votes": 14259.76724137931,
  "year": 1976
},
{
  "average_votes": 14210.393442622952,
  "year": 1977
},
{
  "average_votes": 9992.5703125,
  "year": 1978
},
{
  "average_votes": 15729.419847328245,
  "year": 1979
},
{
  "average_votes": 16487.155688622755,
  "year": 1980
},
{
  "average_votes": 12193.797619047618,
  "year": 1981
},
{
  "average_votes": 15729.898305084746,
  "year": 1982
},
{
  "average_votes": 15521.664596273293,
  "year": 1983
},
```


Question 8:

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

```
# Query
distinct_languages = mflixdb.movies.distinct("languages")

# Output
print(distinct_languages)
```

```
[' Ancient (to 1453)', ' Old', 'Abkhazian', 'Aboriginal', 'Acholi', 'Afrikaans',
```