```
# Set up your connection to Mongo DB here.

import pymongo
from bson.json_util import dumps

uri = "mongodb://sarahh:12345@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.

Question 1:

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

Answer Example 1:

```
locations = mflixdb.theaters.find({"location.address.state": "MA"}, {"location.address.street1": 1,
"location.address.city": 1, "location.address.zipcode": 1,
                                   "_id": 0}
)

print(dumps(locations, indent=2))
```

OR

```
c = mflixdb.theaters.find({'location.address.state' : 'MA'},
                          { "_id": 0, "location.address.street1": 1, "location.address.city": 1,
"location.address.zipcode": 1})
print(dumps(c, indent=2))
```

Question 2:

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

```
# query to count how many theaters are in each state, ordered by state
code
num_theaters_state = mflixdb.theaters.aggregate([
    {"$group": {"_id": {"State": "$location.address.state"}, "Count":
{"$sum": 1}}},
    {"$sort": {"_id": 1}}
    ])

print(dumps(num_theaters_state, indent = 2))

OR

c = mflixdb.theaters.aggregate([{"$group": { "_id":
"$location.address.state", "count": { "$sum": 1 }}},
                                { "$sort": { "_id": 1 }}
                                ])

# Print formatted JSON output
print(dumps(c, indent=2))
```

Question 3:

How many movies are in the Comedy genre?

```
# query to find how many movies are in the comedy genre
num_comedies = mflixdb.movies.aggregate([
    {"$unwind":"$genres"},
    {"$match": {"genres": "Comedy"}},
    {"$group": {"_id": None, "count": {"$sum": 1}}}
])
print(dumps(num_comedies, indent = 2))

OR

count = mflixdb.movies.count_documents({ "genres": "Comedy" })
print(f"Total Comedy Movies: {count}")
```

Question 4:

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

```
# query to get the title and genres of the movie with the longest runtime
longest_movie = mflixdb.movies.find({}, {"title": 1, "genres": 1,"_id": 0}
                ).sort("runtime", -1).limit(1)

print(dumps(longest_movie, indent = 3))
```

OR

```
c = mflixdb.movies.aggregate([
{ "$sort": { "runtime": -1 } },  # Sort movies by runtime in descending
order
    { "$limit": 1 },  # Get only the longest movie
    { "$project": { "_id": 0, "title": 1, "genres": 1, "runtime": 1 } }]
)
print(dumps(c, 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.

```
# query to get the title and roten tomatoes viewer rating score for movies
with a score of at least 3

highly_rated = mflixdb.movies.aggregate([

    {"$match": {"year": {"$gt": 2010}, "tomatoes.viewer.rating": {"$gte":
3}}},
```

```
    {"$sort": {"tomatoes.viewer.rating": -1}},
    {"$project": {"_id":0, "title":1, "viewer_rating":
"$tomatoes.viewer.rating"}},
])
print(dumps(highly_rated, indent=4))

OR
c = mflixdb.movies.aggregate([
    { "$match": {
        "year": { "$gte": 2010 },
        "tomatoes.viewer.rating": { "$gte": 3 }
    }},
    { "$project": {
        "_id": 0,
        "title": 1,
        "viewer_rating": "$tomatoes.viewer.rating"
    }},
    { "$sort": { "viewer_rating": -1 } }
])
print(dumps(c, 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.

```
# query to count the number of movies in each year containing "police" in
the plot string
police_by_year = mflixdb.movies.aggregate([
                    {"$match": {"plot": {"$regex": "police", "$options":
"i"}}},
```

```
                    {"$group": {"_id": {"Year": "$year"}, "Count":
{"$sum": 1}}},
                    {"$sort": {"_id": 1}}
])

print(dumps(police_by_year, indent=2))
```

OR

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 to find the average number of imdb votes per year for movies
between 1970 and 2000, ordered by year
avg_imdb = mflixdb.movies.aggregate([
            {"$match": {"year": {"$gte": 1970, "$lte": 2000}}},
            {"$group": {"_id": {"Year": "$year"}, "Avg Imdb Votes":
{"$avg": "$imdb.votes"}}},
            {"$sort": {"_id": 1}}
])

print(dumps(avg_imdb, indent = 2))
```

OR

Question 8:

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

```
c = mflixdb.movies.aggregate(
    [
    { "$unwind": "$languages" },  # Unwind language arrays if they exist
    { "$group": { "_id": "$languages" } },  # Group by unique language
    { "$sort": { "_id": 1 } }  # Sort alphabetically
]
)
print(dumps(c, indent=2))
```

```
# Set up your connection to Mongo DB here.
import pymongo
from bson.json_util import dumps
uri = "mongodb://melissa:abc216@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.

```
theaters = mflixdb.theaters.find({"location.address.state":
"MA"},
                                 {"location.address.street1":1,
```

```
                                        "location.address.city":1,
"location.address.zipcode":1})

print(dumps(theaters, indent=2))
```

Question 2:

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

```
theaters_by_state = mflixdb.theaters.aggregate([
   {"$group": {"_id": "$location.address.state", "Theater Count": {"$sum": 1}}},
   {"$sort": {"_id": 1}}
])


print(dumps(theaters_by_state, indent = 2))
```

Question 3:

How many movies are in the Comedy genre?

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


print('Number of Movies in Comedy Genre:', comedy_movies)
```

Question 4:

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

```
longest_run_time = mflixdb.movies.find({}, {"_id":0, "title":1, "genres":1}).sort("runtime", -1).limit(1)


print(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.

```
movies_after_2010 = mflixdb.movies.find({"year": {"$gt": 2010}, "tomatoes.viewer.rating":{"$gte":3}},
                                        {"_id": 0, "title":1, "Viewer Rating":
"$tomatoes.viewer.rating"}).sort("tomatoes.viewer.rating", -1)

print(dumps(movies_after_2010, 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.

```
plot_police = mflixdb.movies.aggregate([
    {"$match": {"plot": {"$regex": "police", "$options": "i"}}},
    {"$group": {"_id": "$year", "Movie Count": {"$sum":1}}},
    {"$sort": {"_id": 1}}
    ])

print(dumps(plot_police, indent = 2))
```

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.

```
avg_imdb = mflixdb.movies.aggregate([
    {"$match": {"year": {"$gte": 1970, "$lte": 2000}, "imdb.votes": {"$exists": True}}},  # Ensure votes exist
    {"$group": {"_id": "$year", "Avg Votes": {"$avg": "$imdb.votes"}}},
    {"$sort": {"_id": 1}}
])

print(dumps(avg_imdb, indent = 2))
```

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")

print(movie_languages)
```