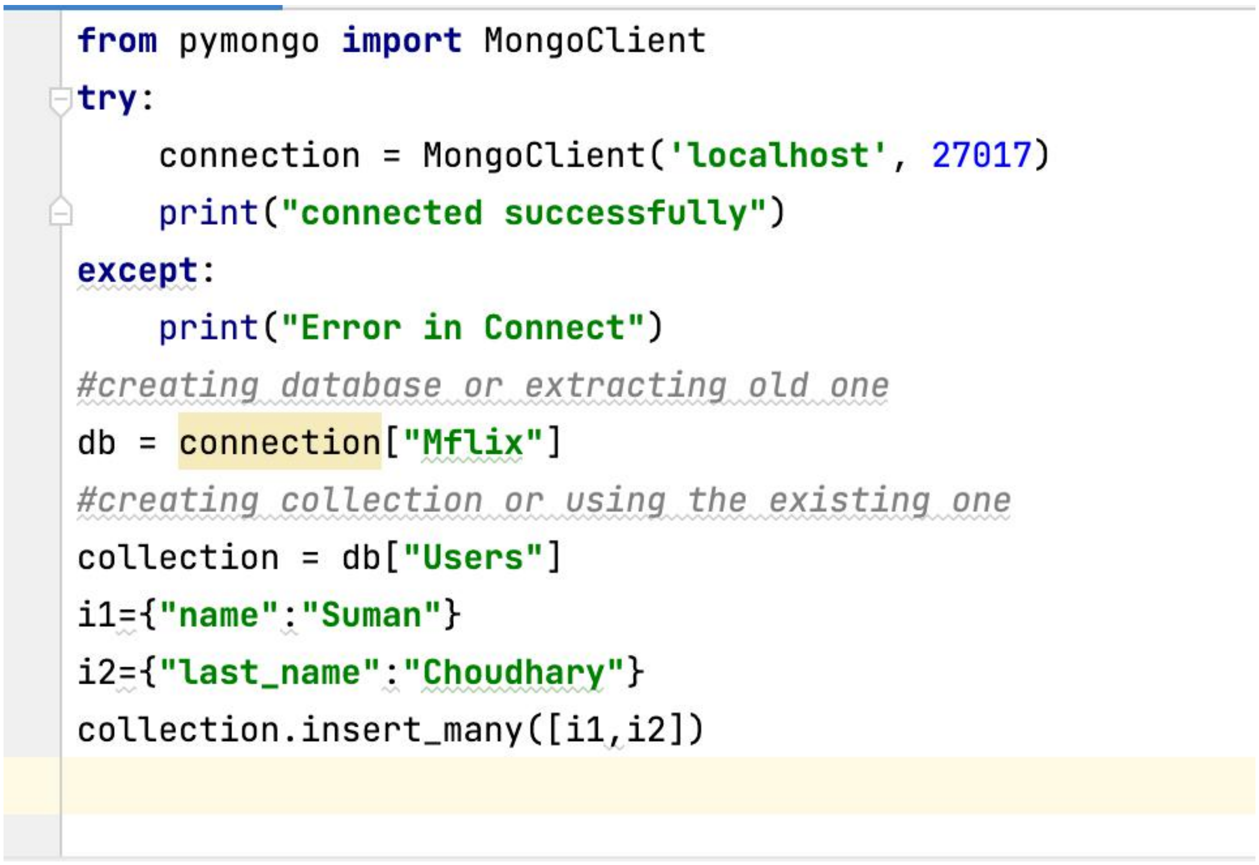
Tasks to be performed:

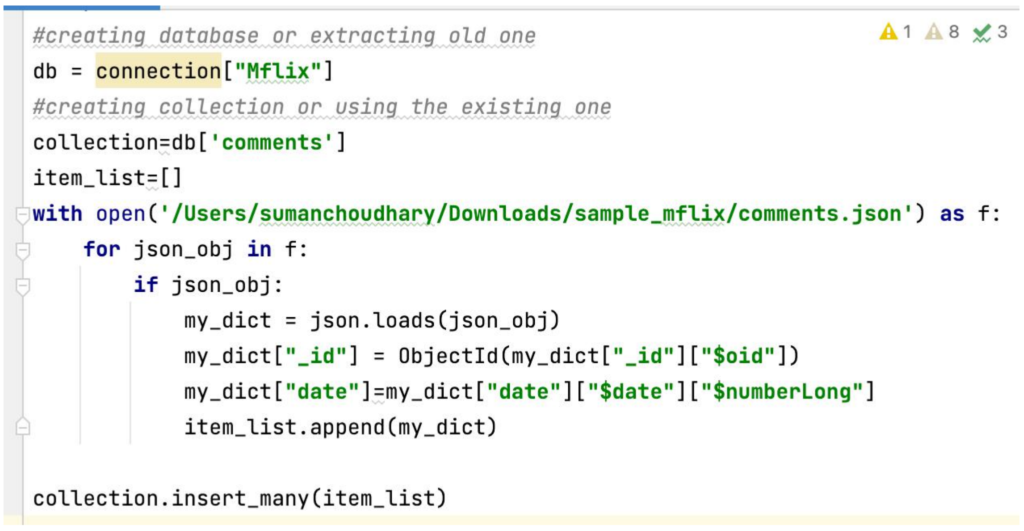
1. Create a Python application to connect to MongoDB.



```python
from pymongo import MongoClient
try:
    connection = MongoClient('localhost', 27017)
    print("connected successfully")
except:
    print("Error in Connect")
#creating database or extracting old one
db = connection["Mflix"]
#creating collection or using the existing one
collection = db["Users"]
i1={"name":"Suman"}
i2={"last_name":"Choudhary"}
collection.insert_many([i1,i2])
```

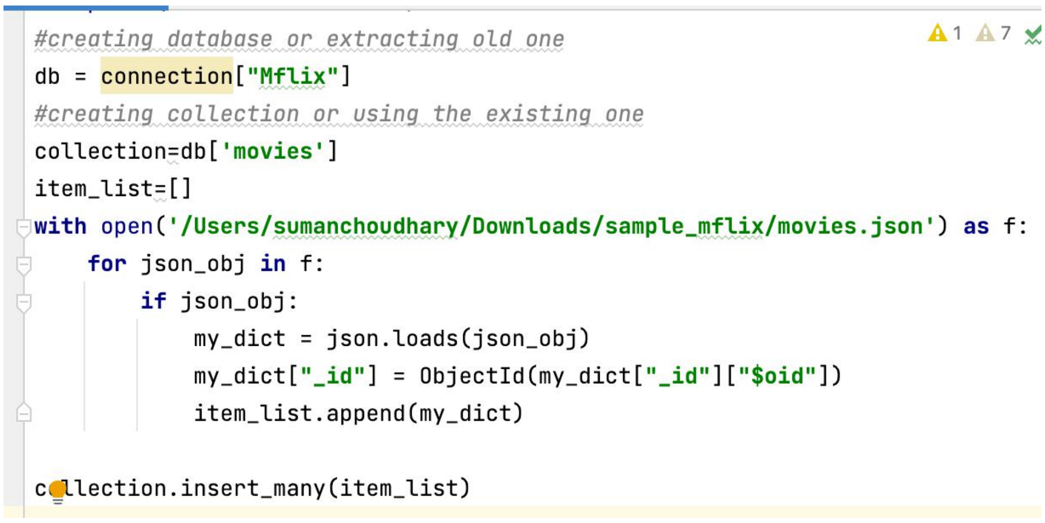
2. Bulk load the JSON files in the individual MongoDB collections using Python. MongoDB collections

A. Comments



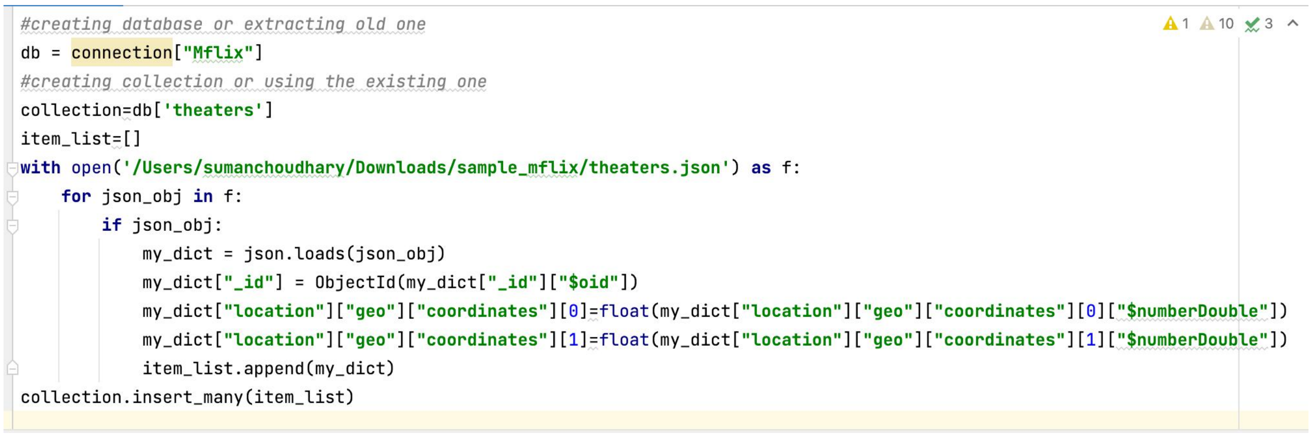
```python
#creating database or extracting old one
db = connection["Mflix"]
#creating collection or using the existing one
collection=db['comments']
item_list=[]
with open('/Users/sumanchoudhary/Downloads/sample_mflix/comments.json') as f:
    for json_obj in f:
        if json_obj:
            my_dict = json.loads(json_obj)
            my_dict["_id"] = ObjectId(my_dict["_id"]["$oid"])
            my_dict["date"]=my_dict["date"]["$date"]["$numberLong"]
            item_list.append(my_dict)

collection.insert_many(item_list)
```

B. Movies



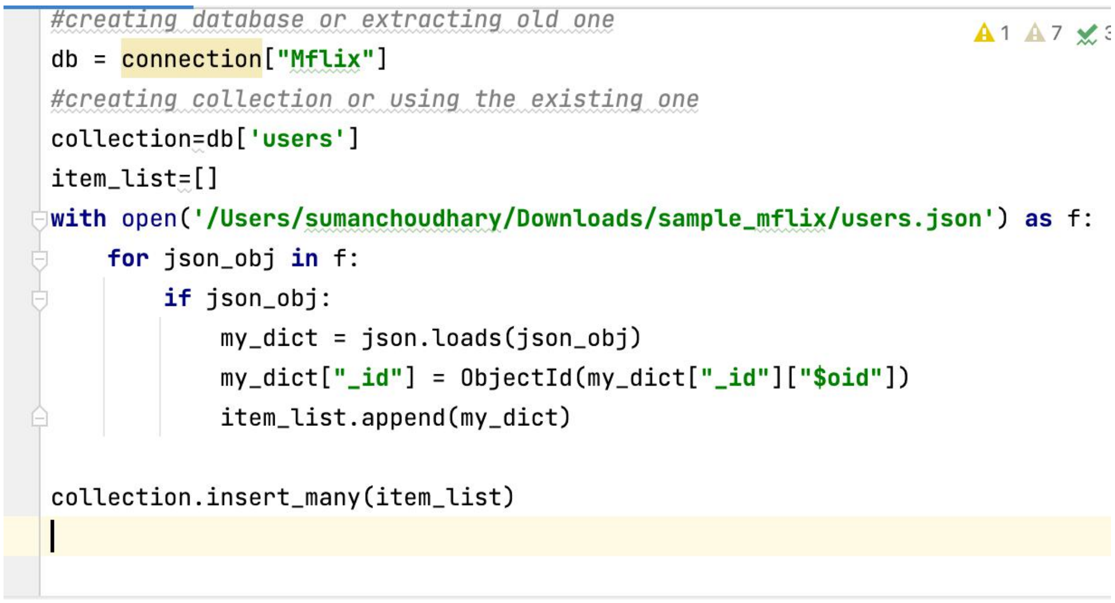
```python
#creating database or extracting old one
db = connection["Mflix"]
#creating collection or using the existing one
collection=db['movies']
item_list=[]
with open('/Users/sumanchoudhary/Downloads/sample_mflix/movies.json') as f:
    for json_obj in f:
        if json_obj:
            my_dict = json.loads(json_obj)
            my_dict["_id"] = ObjectId(my_dict["_id"]["$oid"])
            item_list.append(my_dict)

collection.insert_many(item_list)
```

C. Theaters



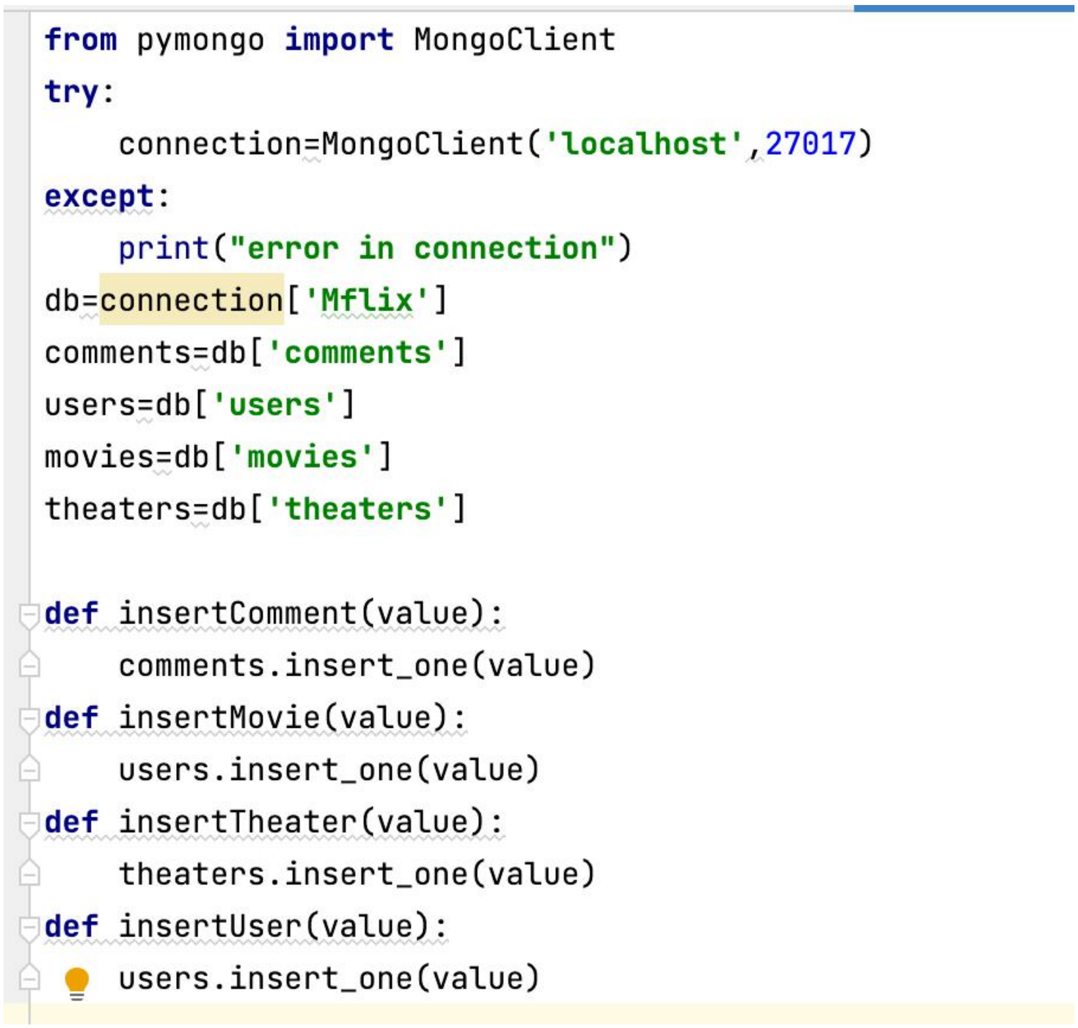
```python
#creating database or extracting old one
db = connection["Mflix"]
#creating collection or using the existing one
collection=db['theaters']
item_list=[]
with open('/Users/sumanchoudhary/Downloads/sample_mflix/theaters.json') as f:
    for json_obj in f:
        if json_obj:
            my_dict = json.loads(json_obj)
            my_dict["_id"] = ObjectId(my_dict["_id"]["$oid"])
            my_dict["location"]["geo"]["coordinates"][0]=float(my_dict["location"]["geo"]["coordinates"][0]["$numberDouble"])
            my_dict["location"]["geo"]["coordinates"][1]=float(my_dict["location"]["geo"]["coordinates"][1]["$numberDouble"])
            item_list.append(my_dict)
collection.insert_many(item_list)
```

D. Users



```python
#creating database or extracting old one
db = connection["Mflix"]
#creating collection or using the existing one
collection=db['users']
item_list=[]
with open('/Users/sumanchoudhary/Downloads/sample_mflix/users.json') as f:
    for json_obj in f:
        if json_obj:
            my_dict = json.loads(json_obj)
            my_dict["_id"] = ObjectId(my_dict["_id"]["$oid"])
            item_list.append(my_dict)

collection.insert_many(item_list)
```

3. Create Python methods and MongoDB queries to insert new comments, movies, theatres, and users into respective MongoDB collections.

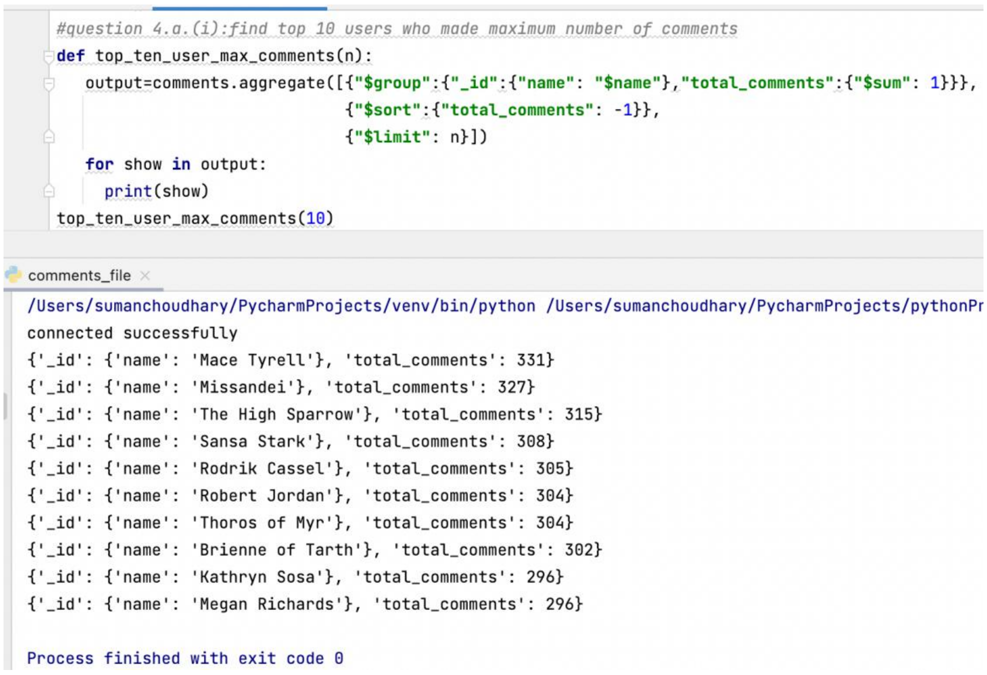


```
from pymongo import MongoClient
try:
    connection=MongoClient('localhost',27017)
except:
    print("error in connection")
db=connection['Mflix']
comments=db['comments']
users=db['users']
movies=db['movies']
theaters=db['theaters']

def insertComment(value):
    comments.insert_one(value)
def insertMovie(value):
    users.insert_one(value)
def insertTheater(value):
    theaters.insert_one(value)
def insertUser(value):
    users.insert_one(value)
```

4. Create Python methods and MongoDB queries to support the below operations -

a. <u>comments collection</u>

(i) Find top 10 users who made the maximum number of comments

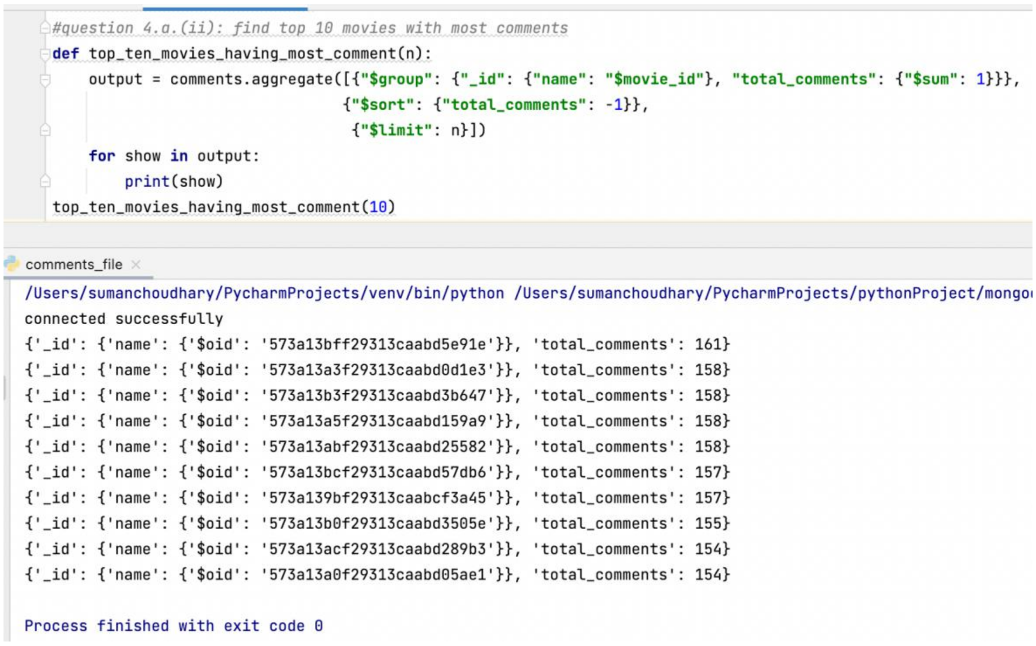


```
#question 4.a.(i):find top 10 users who made maximum number of comments
def top_ten_user_max_comments(n):
    output=comments.aggregate([{"$group":{"_id":{"name": "$name"},"total_comments":{"$sum": 1}}},
                               {"$sort":{"total_comments": -1}},
                               {"$limit": n}])
    for show in output:
      print(show)
top_ten_user_max_comments(10)
```

comments_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonPr
connected successfully
{'_id': {'name': 'Mace Tyrell'}, 'total_comments': 331}
{'_id': {'name': 'Missandei'}, 'total_comments': 327}
{'_id': {'name': 'The High Sparrow'}, 'total_comments': 315}
{'_id': {'name': 'Sansa Stark'}, 'total_comments': 308}
{'_id': {'name': 'Rodrik Cassel'}, 'total_comments': 305}
{'_id': {'name': 'Robert Jordan'}, 'total_comments': 304}
{'_id': {'name': 'Thoros of Myr'}, 'total_comments': 304}
{'_id': {'name': 'Brienne of Tarth'}, 'total_comments': 302}
{'_id': {'name': 'Kathryn Sosa'}, 'total_comments': 296}
{'_id': {'name': 'Megan Richards'}, 'total_comments': 296}

Process finished with exit code 0
```

(ii)Find top 10 movies with most comments

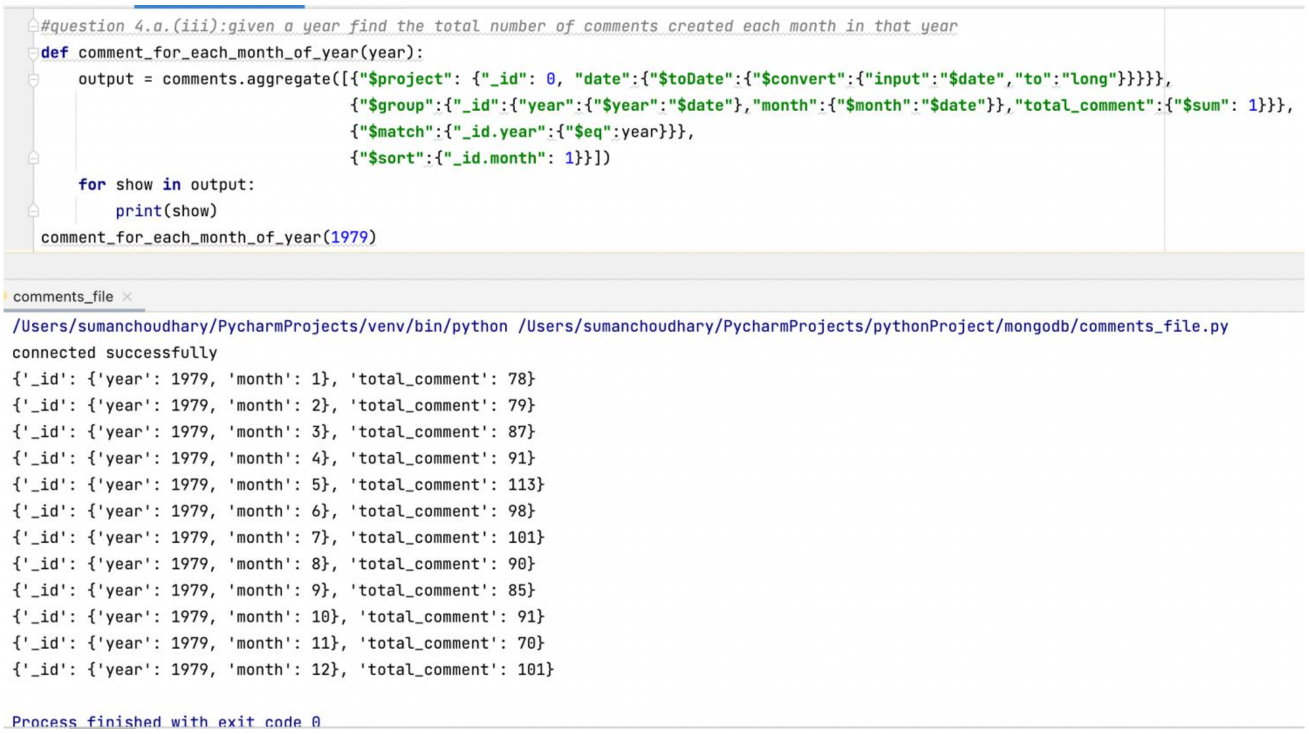


```
#question 4.a.(ii): find top 10 movies with most comments
def top_ten_movies_having_most_comment(n):
    output = comments.aggregate([{"$group": {"_id": {"name": "$movie_id"}, "total_comments": {"$sum": 1}}},
                                {"$sort": {"total_comments": -1}},
                                 {"$limit": n}])
    for show in output:
        print(show)
top_ten_movies_having_most_comment(10)
```

comments_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongo
connected successfully
{'_id': {'name': {'$oid': '573a13bff29313caabd5e91e'}}, 'total_comments': 161}
{'_id': {'name': {'$oid': '573a13a3f29313caabd0d1e3'}}, 'total_comments': 158}
{'_id': {'name': {'$oid': '573a13b3f29313caabd3b647'}}, 'total_comments': 158}
{'_id': {'name': {'$oid': '573a13a5f29313caabd159a9'}}, 'total_comments': 158}
{'_id': {'name': {'$oid': '573a13abf29313caabd25582'}}, 'total_comments': 158}
{'_id': {'name': {'$oid': '573a13bcf29313caabd57db6'}}, 'total_comments': 157}
{'_id': {'name': {'$oid': '573a139bf29313caabcf3a45'}}, 'total_comments': 157}
{'_id': {'name': {'$oid': '573a13b0f29313caabd3505e'}}, 'total_comments': 155}
{'_id': {'name': {'$oid': '573a13acf29313caabd289b3'}}, 'total_comments': 154}
{'_id': {'name': {'$oid': '573a13a0f29313caabd05ae1'}}, 'total_comments': 154}

Process finished with exit code 0
```

(iii)Given a year find the total number of comments created each month in that year



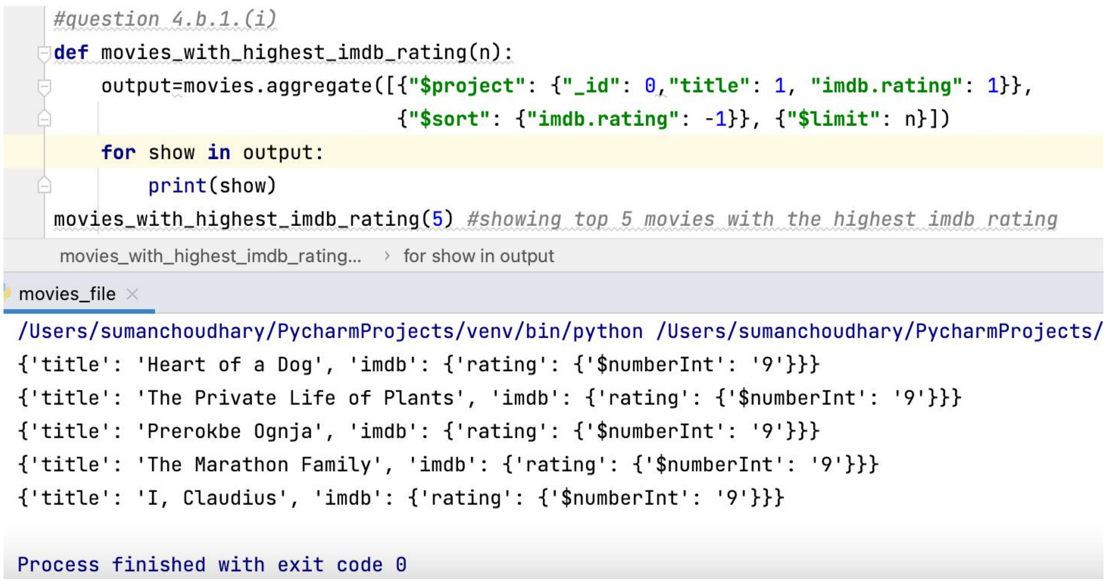
```
#question 4.a.(iii):given a year find the total number of comments created each month in that year
def comment_for_each_month_of_year(year):
    output = comments.aggregate([{"$project": {"_id": 0, "date":{"$toDate":{"$convert":{"input":"$date","to":"long"}}}}},
                                 {"$group":{"_id":{"year":{"$year":"$date"},"month":{"$month":"$date"}},"total_comment":{"$sum": 1}}},
                                 {"$match":{"_id.year":{"$eq":year}}},
                                 {"$sort":{"_id.month": 1}}])
    for show in output:
        print(show)
comment_for_each_month_of_year(1979)
```

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/comments_file.py
connected successfully
{'_id': {'year': 1979, 'month': 1}, 'total_comment': 78}
{'_id': {'year': 1979, 'month': 2}, 'total_comment': 79}
{'_id': {'year': 1979, 'month': 3}, 'total_comment': 87}
{'_id': {'year': 1979, 'month': 4}, 'total_comment': 91}
{'_id': {'year': 1979, 'month': 5}, 'total_comment': 113}
{'_id': {'year': 1979, 'month': 6}, 'total_comment': 98}
{'_id': {'year': 1979, 'month': 7}, 'total_comment': 101}
{'_id': {'year': 1979, 'month': 8}, 'total_comment': 90}
{'_id': {'year': 1979, 'month': 9}, 'total_comment': 85}
{'_id': {'year': 1979, 'month': 10}, 'total_comment': 91}
{'_id': {'year': 1979, 'month': 11}, 'total_comment': 70}
{'_id': {'year': 1979, 'month': 12}, 'total_comment': 101}

Process finished with exit code 0
```

b. movies collection

1. Find top 'N' movies-

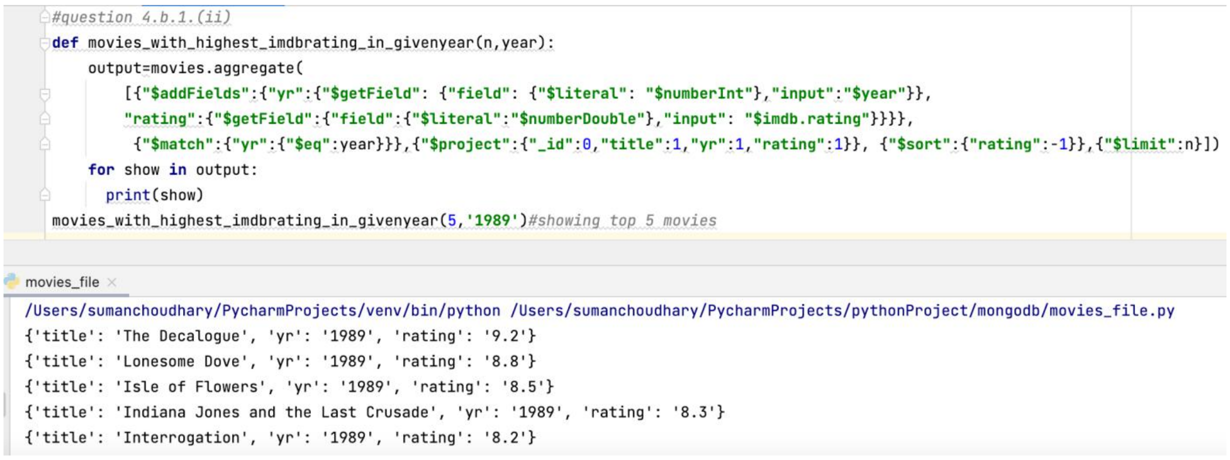
   i. with the highest IMDB rating



```
#question 4.b.1.(i)
def movies_with_highest_imdb_rating(n):
    output=movies.aggregate([{"$project": {"_id": 0,"title": 1, "imdb.rating": 1}},
                             {"$sort": {"imdb.rating": -1}}, {"$limit": n}])
    for show in output:
        print(show)
movies_with_highest_imdb_rating(5) #showing top 5 movies with the highest imdb rating
```

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/
{'title': 'Heart of a Dog', 'imdb': {'rating': {'$numberInt': '9'}}}
{'title': 'The Private Life of Plants', 'imdb': {'rating': {'$numberInt': '9'}}}
{'title': 'Prerokbe Ognja', 'imdb': {'rating': {'$numberInt': '9'}}}
{'title': 'The Marathon Family', 'imdb': {'rating': {'$numberInt': '9'}}}
{'title': 'I, Claudius', 'imdb': {'rating': {'$numberInt': '9'}}}

Process finished with exit code 0
```

ii. with the highest IMDB rating in a given year

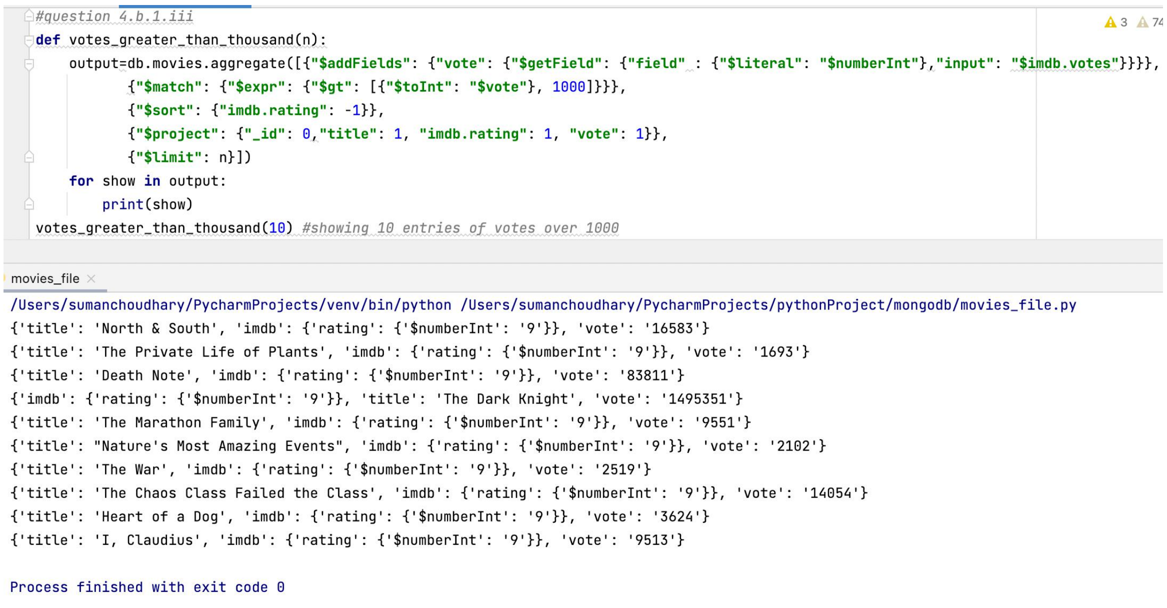


```
#question 4.b.1.(ii)
def movies_with_highest_imdbrating_in_givenyear(n,year):
    output=movies.aggregate(
        [{"$addFields":{"yr":{"$getField": {"field": {"$literal": "$numberInt"},"input":"$year"}},
        "rating":{"$getField":{"field":{"$literal":"$numberDouble"},"input": "$imdb.rating"}}}},
         {"$match":{"yr":{"$eq":year}}},{"$project":{"_id":0,"title":1,"yr":1,"rating":1}}, {"$sort":{"rating":-1}},{"$limit":n}])
    for show in output:
      print(show)
movies_with_highest_imdbrating_in_givenyear(5,'1989')#showing top 5 movies
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'title': 'The Decalogue', 'yr': '1989', 'rating': '9.2'}
{'title': 'Lonesome Dove', 'yr': '1989', 'rating': '8.8'}
{'title': 'Isle of Flowers', 'yr': '1989', 'rating': '8.5'}
{'title': 'Indiana Jones and the Last Crusade', 'yr': '1989', 'rating': '8.3'}
{'title': 'Interrogation', 'yr': '1989', 'rating': '8.2'}
```

iii.with highest IMDB rating with number of votes>1000

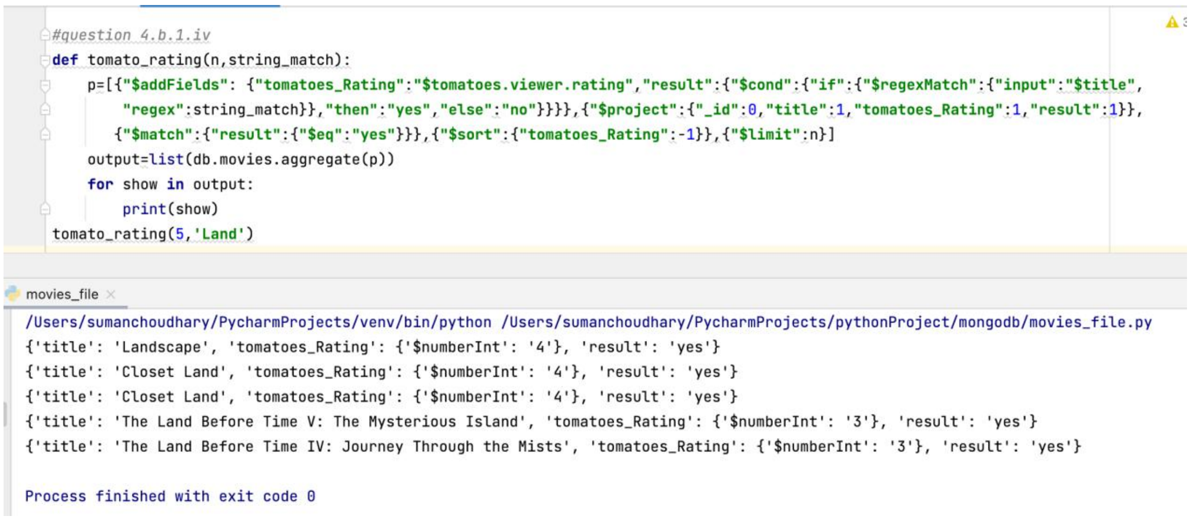


```
#question 4.b.1.iii
def votes_greater_than_thousand(n):
    output=db.movies.aggregate([{"$addFields": {"vote": {"$getField": {"field" : {"$literal": "$numberInt"},"input": "$imdb.votes"}}}},
           {"$match": {"$expr": {"$gt": [{"$toInt": "$vote"}, 1000]}}},
           {"$sort": {"imdb.rating": -1}},
           {"$project": {"_id": 0,"title": 1, "imdb.rating": 1, "vote": 1}},
           {"$limit": n}])
    for show in output:
        print(show)
votes_greater_than_thousand(10) #showing 10 entries of votes over 1000
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'title': 'North & South', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '16583'}
{'title': 'The Private Life of Plants', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '1693'}
{'title': 'Death Note', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '83811'}
{'imdb': {'rating': {'$numberInt': '9'}}, 'title': 'The Dark Knight', 'vote': '1495351'}
{'title': 'The Marathon Family', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '9551'}
{'title': "Nature's Most Amazing Events", 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '2102'}
{'title': 'The War', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '2519'}
{'title': 'The Chaos Class Failed the Class', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '14054'}
{'title': 'Heart of a Dog', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '3624'}
{'title': 'I, Claudius', 'imdb': {'rating': {'$numberInt': '9'}}, 'vote': '9513'}

Process finished with exit code 0
```

iv.with title matching a given pattern sorted by highest tomatoes



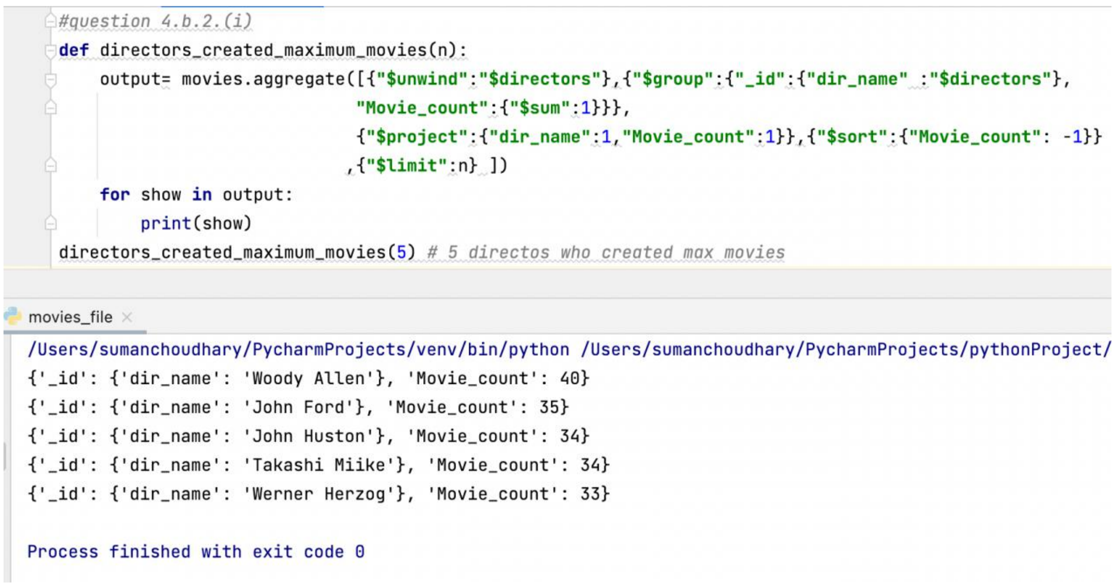
```
#question 4.b.1.iv
def tomato_rating(n,string_match):
    p=[{"$addFields": {"tomatoes_Rating":"$tomatoes.viewer.rating","result":{"$cond":{"if":{"$regexMatch":{"input":"$title",
        "regex":string_match}},"then":"yes","else":"no"}}}},{"$project":{"_id":0,"title":1,"tomatoes_Rating":1,"result":1}},
       {"$match":{"result":{"$eq":"yes"}}},{"$sort":{"tomatoes_Rating":-1}},{"$limit":n}]
    output=list(db.movies.aggregate(p))
    for show in output:
        print(show)
tomato_rating(5,'Land')
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'title': 'Landscape', 'tomatoes_Rating': {'$numberInt': '4'}, 'result': 'yes'}
{'title': 'Closet Land', 'tomatoes_Rating': {'$numberInt': '4'}, 'result': 'yes'}
{'title': 'Closet Land', 'tomatoes_Rating': {'$numberInt': '4'}, 'result': 'yes'}
{'title': 'The Land Before Time V: The Mysterious Island', 'tomatoes_Rating': {'$numberInt': '3'}, 'result': 'yes'}
{'title': 'The Land Before Time IV: Journey Through the Mists', 'tomatoes_Rating': {'$numberInt': '3'}, 'result': 'yes'}

Process finished with exit code 0
```

2. Find top “N” directors:

i.who created the maximum number of movies

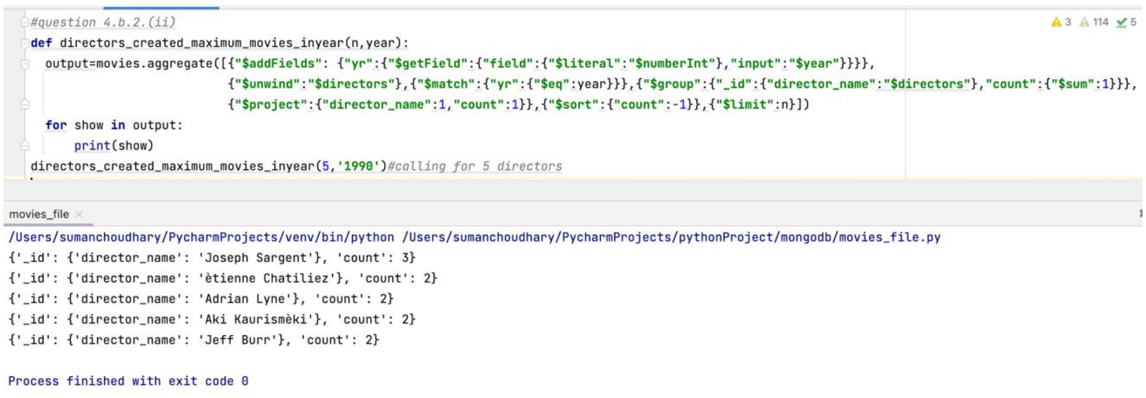


```
#question 4.b.2.(i)
def directors_created_maximum_movies(n):
    output= movies.aggregate([{"$unwind":"$directors"},{"$group":{"_id":{"dir_name" :"$directors"},
                              "Movie_count":{"$sum":1}}},
                             {"$project":{"dir_name":1,"Movie_count":1}},{"$sort":{"Movie_count": -1}}
                            ,{"$limit":n} ])
    for show in output:
        print(show)
directors_created_maximum_movies(5) # 5 directos who created max movies
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/
{'_id': {'dir_name': 'Woody Allen'}, 'Movie_count': 40}
{'_id': {'dir_name': 'John Ford'}, 'Movie_count': 35}
{'_id': {'dir_name': 'John Huston'}, 'Movie_count': 34}
{'_id': {'dir_name': 'Takashi Miike'}, 'Movie_count': 34}
{'_id': {'dir_name': 'Werner Herzog'}, 'Movie_count': 33}

Process finished with exit code 0
```

ii.who created maximum number of movies in a given year

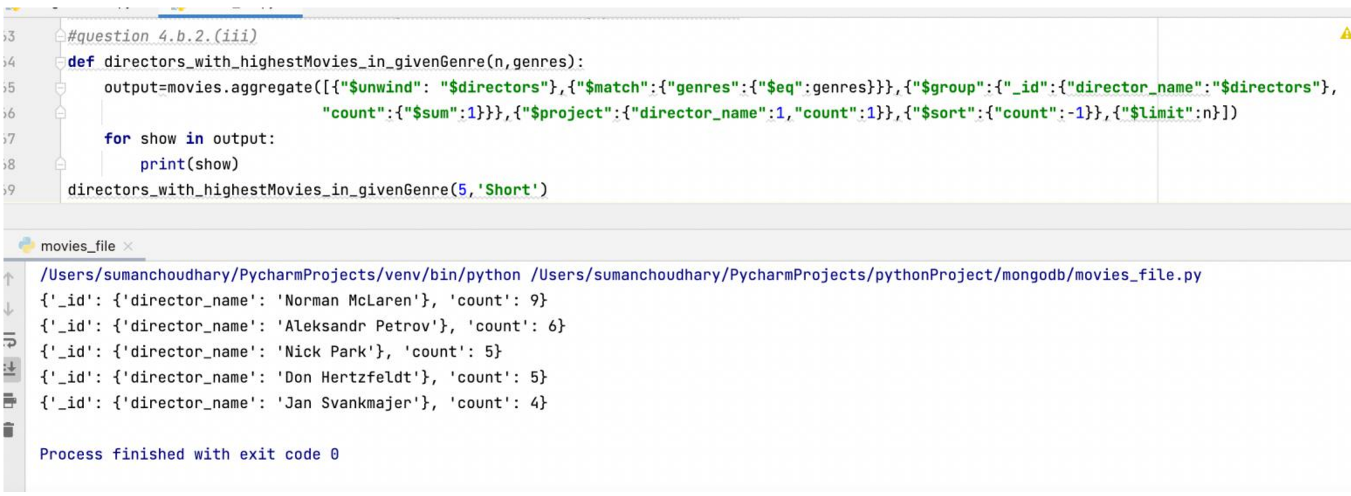


```
#question 4.b.2.(ii)
def directors_created_maximum_movies_inyear(n,year):
  output=movies.aggregate([{"$addFields": {"yr":{"$getField":{"field":{"$literal":"$numberInt"},"input":"$year"}}}},
                           {"$unwind":"$directors"},{"$match":{"yr":{"$eq":year}}},{"$group":{"_id":{"director_name":"$directors"},"count":{"$sum":1}}},
                           {"$project":{"director_name":1,"count":1}},{"$sort":{"count":-1}},{"$limit":n}])
  for show in output:
      print(show)
directors_created_maximum_movies_inyear(5,'1990')#calling for 5 directors
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'_id': {'director_name': 'Joseph Sargent'}, 'count': 3}
{'_id': {'director_name': 'ètienne Chatiliez'}, 'count': 2}
{'_id': {'director_name': 'Adrian Lyne'}, 'count': 2}
{'_id': {'director_name': 'Aki Kaurismèki'}, 'count': 2}
{'_id': {'director_name': 'Jeff Burr'}, 'count': 2}

Process finished with exit code 0
```

iii.who created maximum number of movies for a given genre



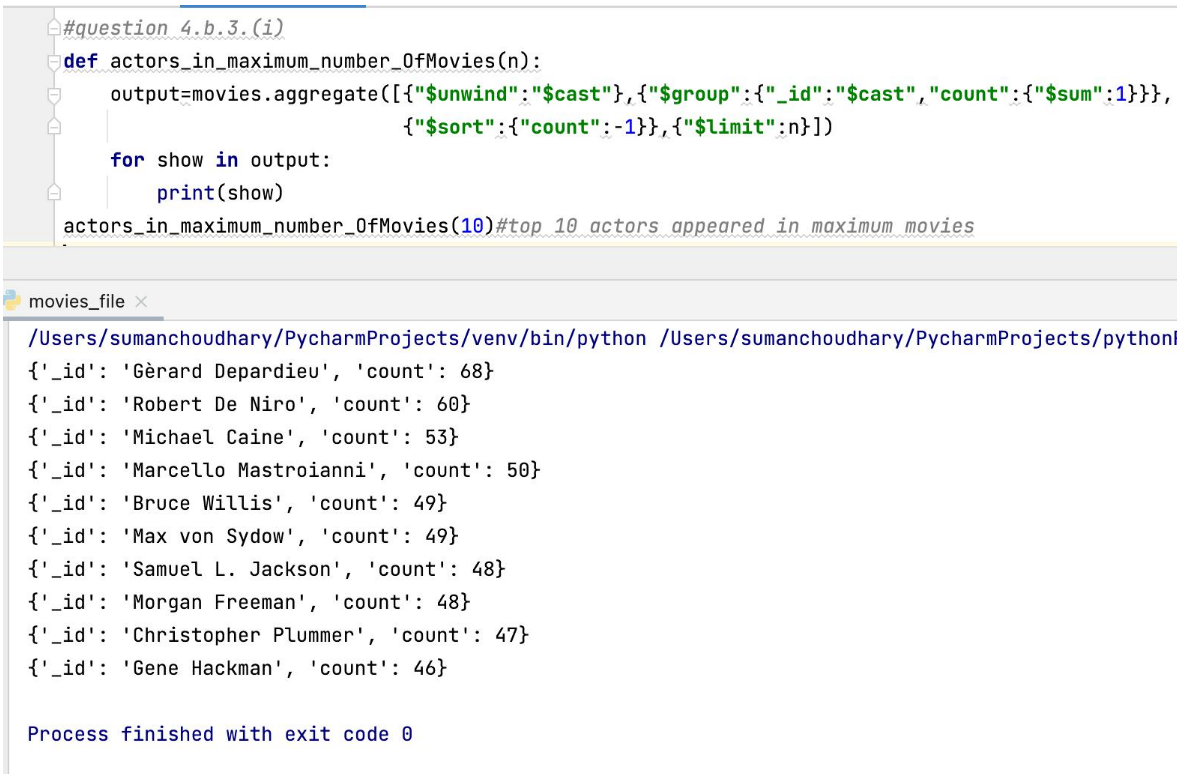
```
#question 4.b.2.(iii)
def directors_with_highestMovies_in_givenGenre(n,genres):
    output=movies.aggregate([{"$unwind": "$directors"},{"$match":{"genres":{"$eq":genres}}},{"$group":{"_id":{"director_name":"$directors"},
                         "count":{"$sum":1}}},{"$project":{"director_name":1,"count":1}},{"$sort":{"count":-1}},{"$limit":n}])
    for show in output:
        print(show)
directors_with_highestMovies_in_givenGenre(5,'Short')
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'_id': {'director_name': 'Norman McLaren'}, 'count': 9}
{'_id': {'director_name': 'Aleksandr Petrov'}, 'count': 6}
{'_id': {'director_name': 'Nick Park'}, 'count': 5}
{'_id': {'director_name': 'Don Hertzfeldt'}, 'count': 5}
{'_id': {'director_name': 'Jan Svankmajer'}, 'count': 4}

Process finished with exit code 0
```

3.Find top “N” actors

i.who starred in the maximum number of movies

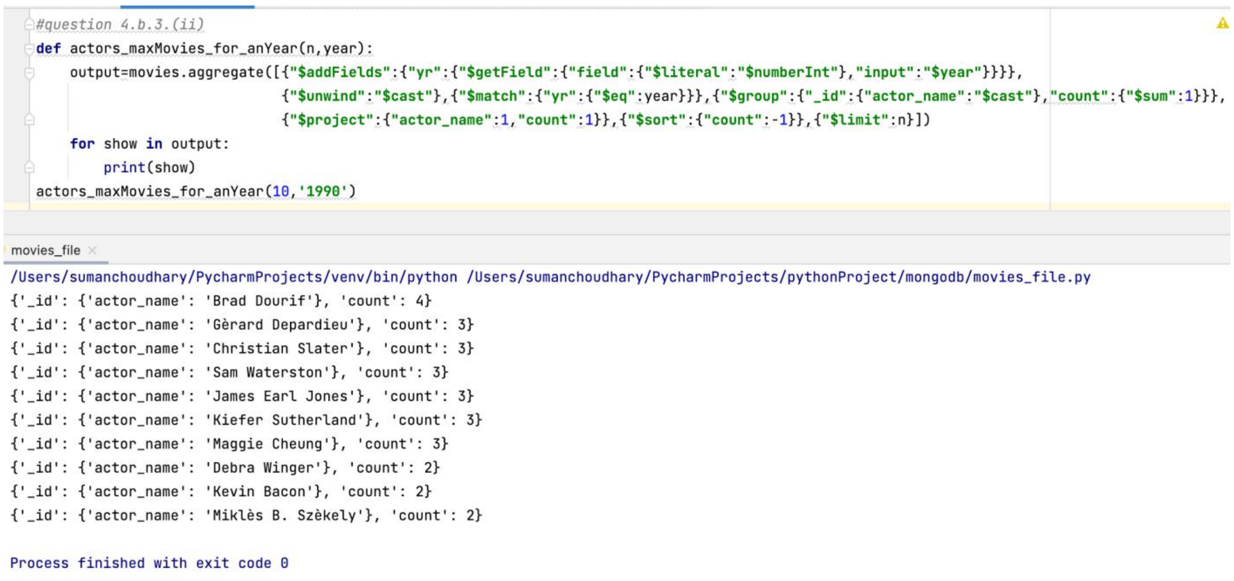


```
#question 4.b.3.(i)
def actors_in_maximum_number_OfMovies(n):
    output=movies.aggregate([{"$unwind":"$cast"},{"$group":{"_id":"$cast","count":{"$sum":1}}},
                            {"$sort":{"count":-1}},{"$limit":n}])
    for show in output:
        print(show)
actors_in_maximum_number_OfMovies(10)#top 10 actors appeared in maximum movies
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonP
{'_id': 'Gèrard Depardieu', 'count': 68}
{'_id': 'Robert De Niro', 'count': 60}
{'_id': 'Michael Caine', 'count': 53}
{'_id': 'Marcello Mastroianni', 'count': 50}
{'_id': 'Bruce Willis', 'count': 49}
{'_id': 'Max von Sydow', 'count': 49}
{'_id': 'Samuel L. Jackson', 'count': 48}
{'_id': 'Morgan Freeman', 'count': 48}
{'_id': 'Christopher Plummer', 'count': 47}
{'_id': 'Gene Hackman', 'count': 46}

Process finished with exit code 0
```

ii.who starred in the maximum number of movies in a given year

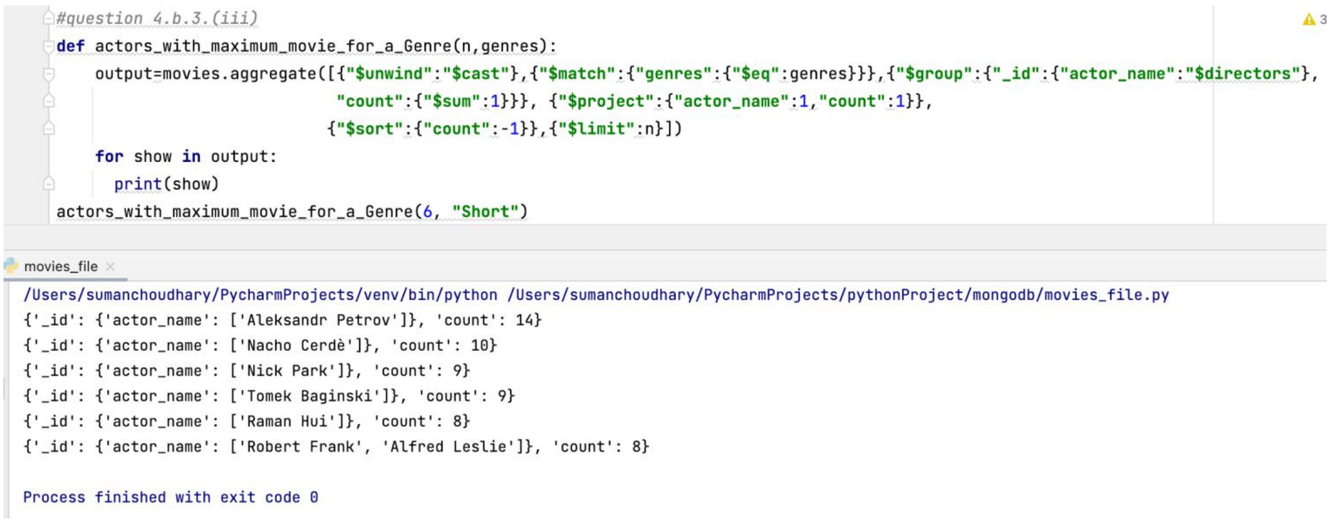


```
#question 4.b.3.(ii)
def actors_maxMovies_for_anYear(n,year):
    output=movies.aggregate([{"$addFields":{"yr":{"$getField":{"field":{"$literal":"$numberInt"},"input":"$year"}}}},
                            {"$unwind":"$cast"},{"$match":{"yr":{"$eq":year}}},{"$group":{"_id":{"actor_name":"$cast"},"count":{"$sum":1}}},
                            {"$project":{"actor_name":1,"count":1}},{"$sort":{"count":-1}},{"$limit":n}])
    for show in output:
        print(show)
actors_maxMovies_for_anYear(10,'1990')
```

movies_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'_id': {'actor_name': 'Brad Dourif'}, 'count': 4}
{'_id': {'actor_name': 'Gèrard Depardieu'}, 'count': 3}
{'_id': {'actor_name': 'Christian Slater'}, 'count': 3}
{'_id': {'actor_name': 'Sam Waterston'}, 'count': 3}
{'_id': {'actor_name': 'James Earl Jones'}, 'count': 3}
{'_id': {'actor_name': 'Kiefer Sutherland'}, 'count': 3}
{'_id': {'actor_name': 'Maggie Cheung'}, 'count': 3}
{'_id': {'actor_name': 'Debra Winger'}, 'count': 2}
{'_id': {'actor_name': 'Kevin Bacon'}, 'count': 2}
{'_id': {'actor_name': 'Miklès B. Szèkely'}, 'count': 2}

Process finished with exit code 0
```

iii.who starred in the maximum number of movies for a given genre

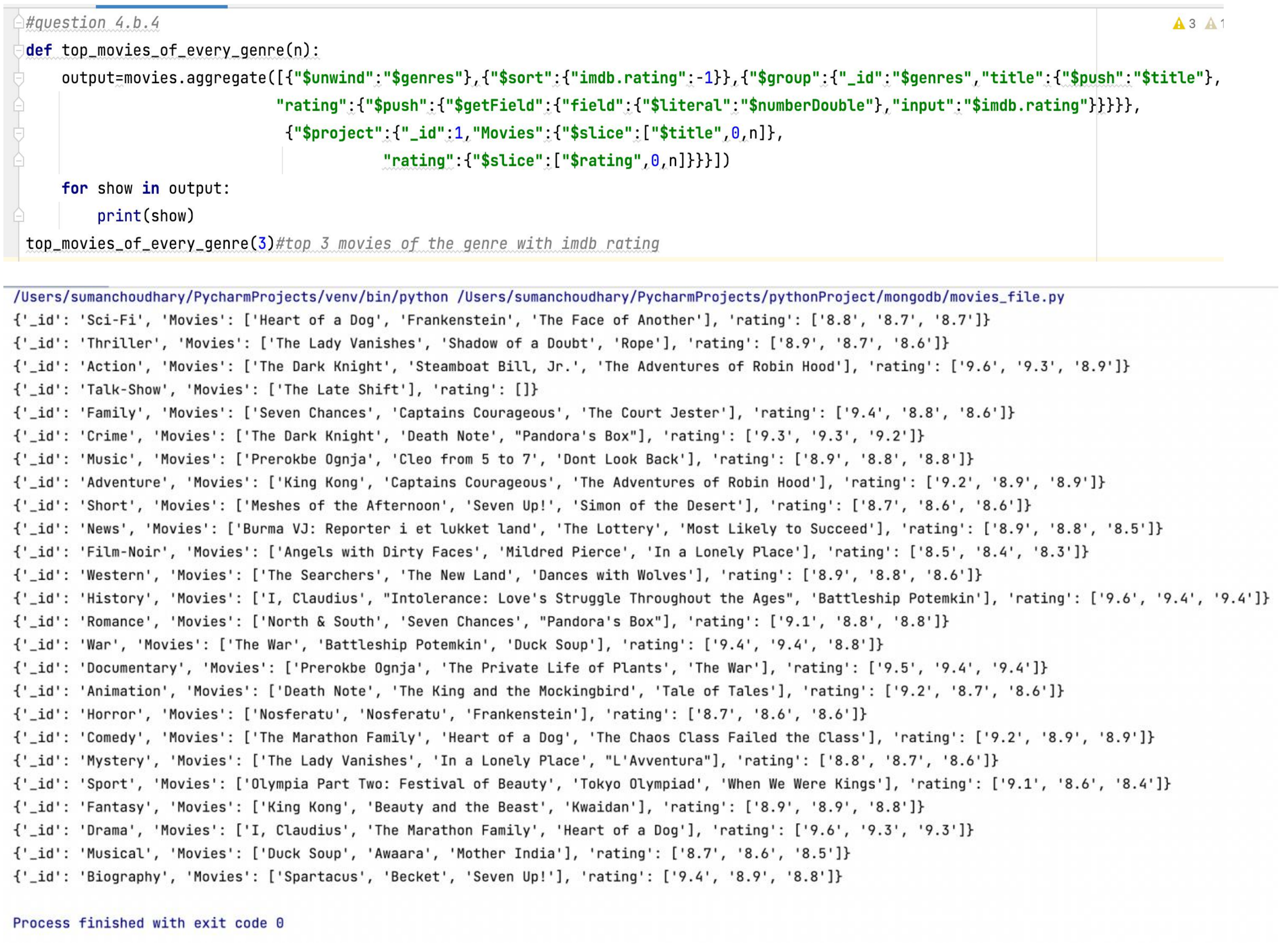


```
#question 4.b.3.(iii)
def actors_with_maximum_movie_for_a_Genre(n,genres):
    output=movies.aggregate([{"$unwind":"$cast"},{"$match":{"genres":{"$eq":genres}}},{"$group":{"_id":{"actor_name":"$directors"},
                            "count":{"$sum":1}}}, {"$project":{"actor_name":1,"count":1}},
                           {"$sort":{"count":-1}},{"$limit":n}])
    for show in output:
      print(show)
actors_with_maximum_movie_for_a_Genre(6, "Short")
```

movies_file ×

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'_id': {'actor_name': ['Aleksandr Petrov']}, 'count': 14}
{'_id': {'actor_name': ['Nacho Cerdè']}, 'count': 10}
{'_id': {'actor_name': ['Nick Park']}, 'count': 9}
{'_id': {'actor_name': ['Tomek Baginski']}, 'count': 9}
{'_id': {'actor_name': ['Raman Hui']}, 'count': 8}
{'_id': {'actor_name': ['Robert Frank', 'Alfred Leslie']}, 'count': 8}

Process finished with exit code 0
```

4.Find top 'N' movies for each genre with the highest IMDB rating

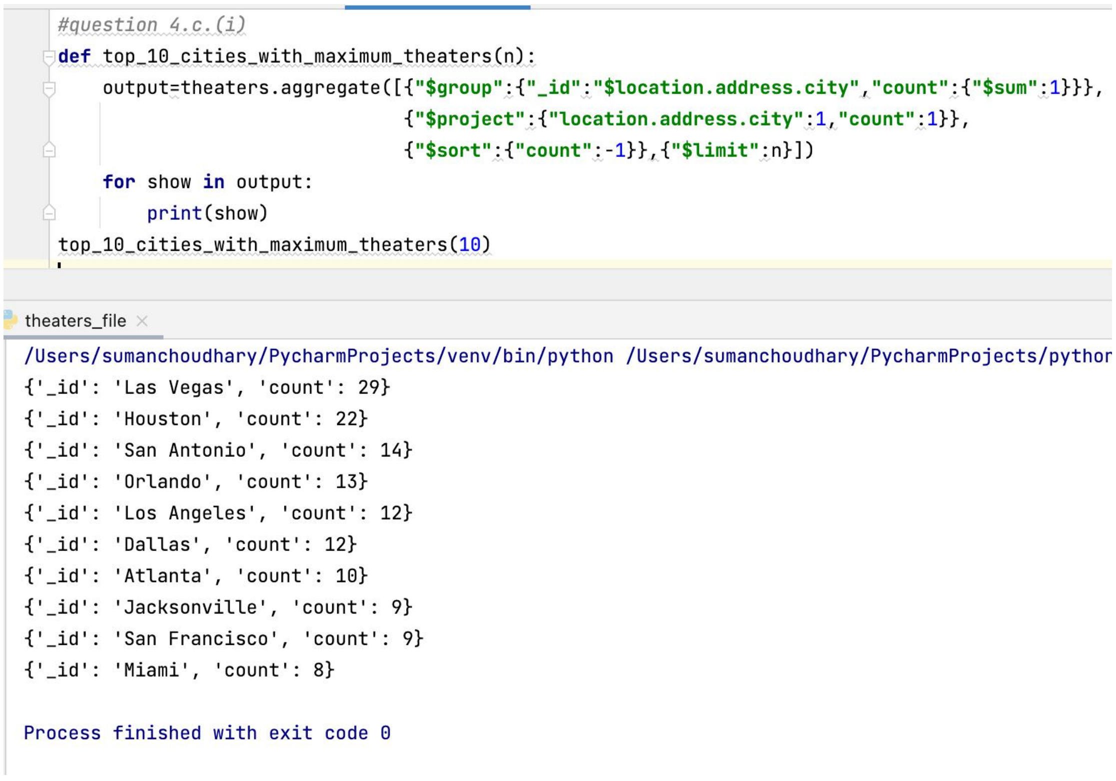


```
#question 4.b.4
def top_movies_of_every_genre(n):
    output=movies.aggregate([{"$unwind":"$genres"},{"$sort":{"imdb.rating":-1}},{"$group":{"_id":"$genres","title":{"$push":"$title"},
                            "rating":{"$push":{"$getField":{"field":{"$literal":"$numberDouble"},"input":"$imdb.rating"}}}}},
                             {"$project":{"_id":1,"Movies":{"$slice":["$title",0,n]},
                                       "rating":{"$slice":["$rating",0,n]}}}])
    for show in output:
        print(show)
top_movies_of_every_genre(3)#top 3 movies of the genre with imdb rating
```

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/movies_file.py
{'_id': 'Sci-Fi', 'Movies': ['Heart of a Dog', 'Frankenstein', 'The Face of Another'], 'rating': ['8.8', '8.7', '8.7']}
{'_id': 'Thriller', 'Movies': ['The Lady Vanishes', 'Shadow of a Doubt', 'Rope'], 'rating': ['8.9', '8.7', '8.6']}
{'_id': 'Action', 'Movies': ['The Dark Knight', 'Steamboat Bill, Jr.', 'The Adventures of Robin Hood'], 'rating': ['9.6', '9.3', '8.9']}
{'_id': 'Talk-Show', 'Movies': ['The Late Shift'], 'rating': []}
{'_id': 'Family', 'Movies': ['Seven Chances', 'Captains Courageous', 'The Court Jester'], 'rating': ['9.4', '8.8', '8.6']}
{'_id': 'Crime', 'Movies': ['The Dark Knight', 'Death Note', "Pandora's Box"], 'rating': ['9.3', '9.3', '9.2']}
{'_id': 'Music', 'Movies': ['Prerokbe Ognja', 'Cleo from 5 to 7', 'Dont Look Back'], 'rating': ['8.9', '8.8', '8.8']}
{'_id': 'Adventure', 'Movies': ['King Kong', 'Captains Courageous', 'The Adventures of Robin Hood'], 'rating': ['9.2', '8.9', '8.9']}
{'_id': 'Short', 'Movies': ['Meshes of the Afternoon', 'Seven Up!', 'Simon of the Desert'], 'rating': ['8.7', '8.6', '8.6']}
{'_id': 'News', 'Movies': ['Burma VJ: Reporter i et lukket land', 'The Lottery', 'Most Likely to Succeed'], 'rating': ['8.9', '8.8', '8.5']}
{'_id': 'Film-Noir', 'Movies': ['Angels with Dirty Faces', 'Mildred Pierce', 'In a Lonely Place'], 'rating': ['8.5', '8.4', '8.3']}
{'_id': 'Western', 'Movies': ['The Searchers', 'The New Land', 'Dances with Wolves'], 'rating': ['8.9', '8.8', '8.6']}
{'_id': 'History', 'Movies': ['I, Claudius', "Intolerance: Love's Struggle Throughout the Ages", 'Battleship Potemkin'], 'rating': ['9.6', '9.4', '9.4']}
{'_id': 'Romance', 'Movies': ['North & South', 'Seven Chances', "Pandora's Box"], 'rating': ['9.1', '8.8', '8.8']}
{'_id': 'War', 'Movies': ['The War', 'Battleship Potemkin', 'Duck Soup'], 'rating': ['9.4', '9.4', '8.8']}
{'_id': 'Documentary', 'Movies': ['Prerokbe Ognja', 'The Private Life of Plants', 'The War'], 'rating': ['9.5', '9.4', '9.4']}
{'_id': 'Animation', 'Movies': ['Death Note', 'The King and the Mockingbird', 'Tale of Tales'], 'rating': ['9.2', '8.7', '8.6']}
{'_id': 'Horror', 'Movies': ['Nosferatu', 'Nosferatu', 'Frankenstein'], 'rating': ['8.7', '8.6', '8.6']}
{'_id': 'Comedy', 'Movies': ['The Marathon Family', 'Heart of a Dog', 'The Chaos Class Failed the Class'], 'rating': ['9.2', '8.9', '8.9']}
{'_id': 'Mystery', 'Movies': ['The Lady Vanishes', 'In a Lonely Place', "L'Avventura"], 'rating': ['8.8', '8.7', '8.6']}
{'_id': 'Sport', 'Movies': ['Olympia Part Two: Festival of Beauty', 'Tokyo Olympiad', 'When We Were Kings'], 'rating': ['9.1', '8.6', '8.4']}
{'_id': 'Fantasy', 'Movies': ['King Kong', 'Beauty and the Beast', 'Kwaidan'], 'rating': ['8.9', '8.9', '8.8']}
{'_id': 'Drama', 'Movies': ['I, Claudius', 'The Marathon Family', 'Heart of a Dog'], 'rating': ['9.6', '9.3', '9.3']}
{'_id': 'Musical', 'Movies': ['Duck Soup', 'Awaara', 'Mother India'], 'rating': ['8.7', '8.6', '8.5']}
{'_id': 'Biography', 'Movies': ['Spartacus', 'Becket', 'Seven Up!'], 'rating': ['9.4', '8.9', '8.8']}

Process finished with exit code 0
```

C. Theaters collection

i.Top 10 cities with the maximum number of theaters

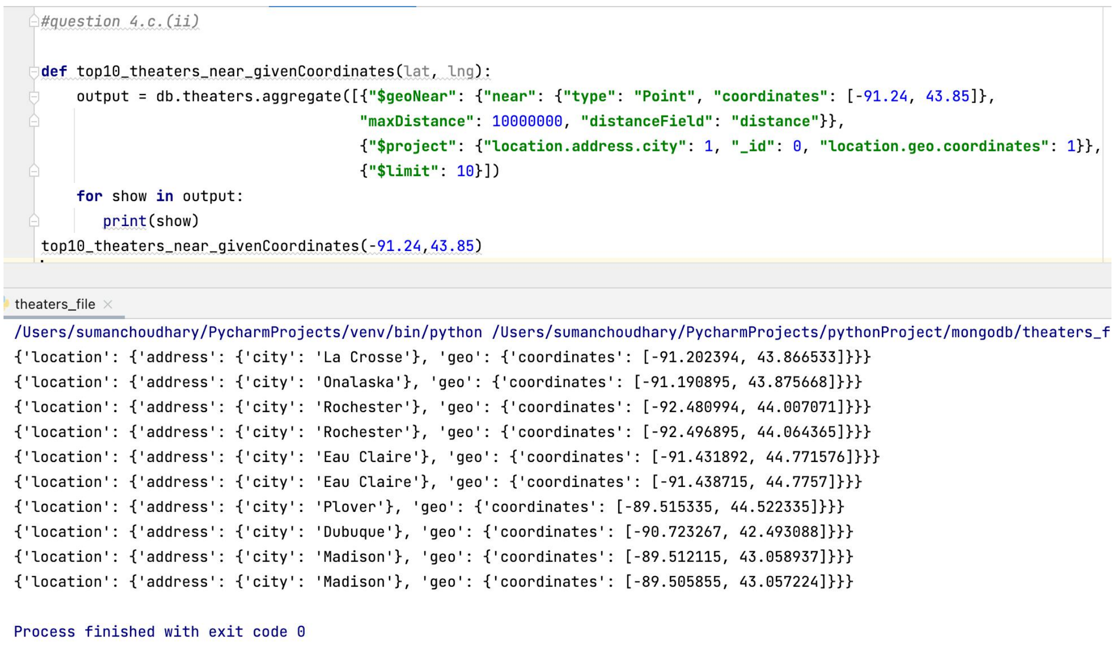


```
#question 4.c.(i)
def top_10_cities_with_maximum_theaters(n):
    output=theaters.aggregate([{"$group":{"_id":"$location.address.city","count":{"$sum":1}}},
                                {"$project":{"location.address.city":1,"count":1}},
                                {"$sort":{"count":-1}},{"$limit":n}])
    for show in output:
        print(show)
top_10_cities_with_maximum_theaters(10)
```

theaters_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythor
{'_id': 'Las Vegas', 'count': 29}
{'_id': 'Houston', 'count': 22}
{'_id': 'San Antonio', 'count': 14}
{'_id': 'Orlando', 'count': 13}
{'_id': 'Los Angeles', 'count': 12}
{'_id': 'Dallas', 'count': 12}
{'_id': 'Atlanta', 'count': 10}
{'_id': 'Jacksonville', 'count': 9}
{'_id': 'San Francisco', 'count': 9}
{'_id': 'Miami', 'count': 8}

Process finished with exit code 0
```

ii.Top 10 theaters nearby given coordinates



```
#question 4.c.(ii)

def top10_theaters_near_givenCoordinates(lat, lng):
    output = db.theaters.aggregate([{"$geoNear": {"near": {"type": "Point", "coordinates": [-91.24, 43.85]},
                                    "maxDistance": 10000000, "distanceField": "distance"}},
                                    {"$project": {"location.address.city": 1, "_id": 0, "location.geo.coordinates": 1}},
                                    {"$limit": 10}])
    for show in output:
        print(show)
top10_theaters_near_givenCoordinates(-91.24,43.85)
```

theaters_file

```
/Users/sumanchoudhary/PycharmProjects/venv/bin/python /Users/sumanchoudhary/PycharmProjects/pythonProject/mongodb/theaters_f
{'location': {'address': {'city': 'La Crosse'}, 'geo': {'coordinates': [-91.202394, 43.866533]}}}
{'location': {'address': {'city': 'Onalaska'}, 'geo': {'coordinates': [-91.190895, 43.875668]}}}
{'location': {'address': {'city': 'Rochester'}, 'geo': {'coordinates': [-92.480994, 44.007071]}}}
{'location': {'address': {'city': 'Rochester'}, 'geo': {'coordinates': [-92.496895, 44.064365]}}}
{'location': {'address': {'city': 'Eau Claire'}, 'geo': {'coordinates': [-91.431892, 44.771576]}}}
{'location': {'address': {'city': 'Eau Claire'}, 'geo': {'coordinates': [-91.438715, 44.7757]}}}
{'location': {'address': {'city': 'Plover'}, 'geo': {'coordinates': [-89.515335, 44.522335]}}}
{'location': {'address': {'city': 'Dubuque'}, 'geo': {'coordinates': [-90.723267, 42.493088]}}}
{'location': {'address': {'city': 'Madison'}, 'geo': {'coordinates': [-89.512115, 43.058937]}}}
{'location': {'address': {'city': 'Madison'}, 'geo': {'coordinates': [-89.505855, 43.057224]}}}

Process finished with exit code 0
```