1. Create a Python application to connect to MongoDB.

```
import json
from bson import ObjectId
from pymongo import MongoClient
from bson.json_util import loads, dumps

client = MongoClient('localhost', 27017)
db = client.get_database('Mflix')
```

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

```
  collection = db.get_collection('movies')
data = []
for line in open("sample_mflix/{filename}"):
  if line:
     new_data = json.loads(line)
     json_str = dumps(new_data)
     record2 = loads(json_str)
     data.append(record2)

for line in data[:2]:
  collection.insert_one(line)
```

3. Create Python methods and MongoDB queries to insert new comments, movies, theatres, and users into respective MongoDB collections.
    a. comments
    b. movies
    c. theaters
    d. Users

```
def insert_users(new_data):
   print(new_data)
   user_collection = db.get_collection('users')
   print(user_collection)
   x = user_collection.insert_one(new_data)
   print("inserted_id ", x.inserted_id)


def insert_in_comments(new_data):
   comments_collection = db.get_collection('comments')
   x = comments_collection.insert_one(new_data)
   print("inserted_id ", x.inserted_id)


def insert_in_theaters(new_data):
   theater_collection = db.get_collection('theaters')
```

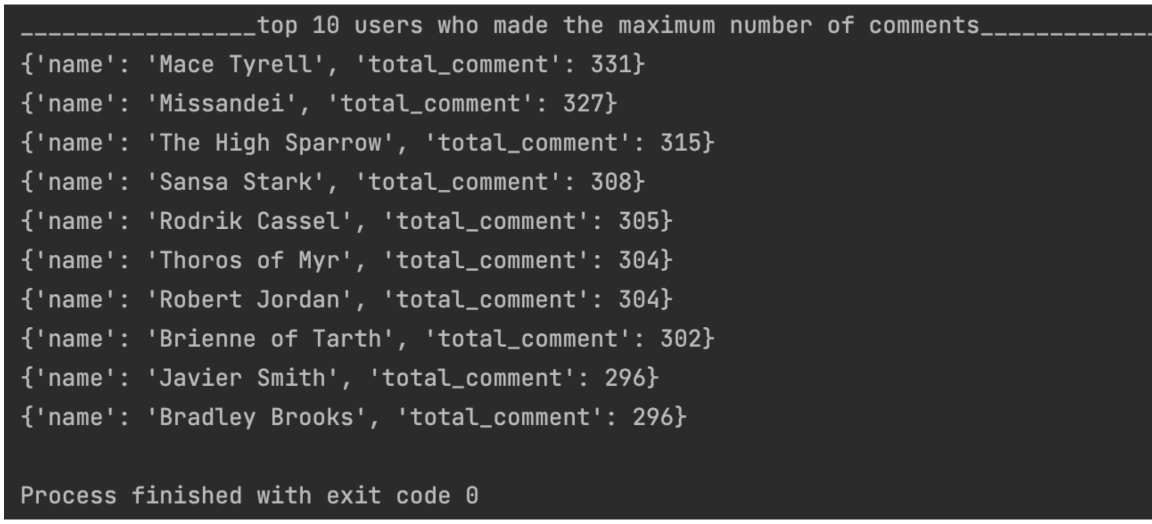
```
    x = theater_collection.insert_one(new_data)
    print("inserted_id ", x.inserted_id)


def insert_in_movies(new_data):
    movies_collection = db.get_collection('movies')
    x = movies_collection.insert_one(new_data)
    print("inserted_id ", x.inserted_id)


user_data = {
    "_id" : ObjectId("59b99db4cfa9a34dcd7885b6"),
    "name" : "Ned Stark",
    "email" : "sean_bean@gameofthron.es",
    "password" : "$2b$12$UREFwsRUoyF0CRqGNK0LzO0HM/jLhgUCNNIJ9RJAqMUQ74crlJ1Vu"
}
 insert_users(user_data)

comment_data = {
    "_id" : ObjectId("5a9427648b0beebeb69579cc"),
    "name" : "Andrea Le",
    "email" : "andrea_le@fakegmail.com",
    "movie_id" : ObjectId("573a1390f29313caabcd418c"),
    "text" : "Rem officiis eaque repellendus amet eos doloribus. Porro dolor voluptatum voluptates
neque culpa molestias. Voluptate unde nulla temporibus ullam.",
    "date" : ISODate("2012-03-26T23:20:16.000Z")
}
# insert_in_comments(comment_data)

theater_data = {
    "_id" : ObjectId("59a47286cfa9a3a73e51e72c"),
    "theaterId" : 1000,
    "location" : {
        "address" : {
            "street1" : "340 W Market",
            "city" : "Bloomington",
            "state" : "MN",
            "zipcode" : "55425"
        },
        "geo" : {
            "type" : "Point",
            "coordinates" : [
                -93.24565,
                44.85466
            ]
        }
    }
}
 insert_in_theaters(theater_data)
```

```
#
movie_data = {
   "_id" : ObjectId("573a1390f29313caabcd4135"),
   "plot" : "Three men hammer on an anvil and pass a bottle of beer around.",
   "genres" : [
      "Short"
   ],
   "runtime" : 1,
   "cast" : [
      "Charles Kayser",
      "John Ott"
   ],
   "num_mflix_comments" : 1,
   "title" : "Blacksmith Scene",
   "fullplot" : "A stationary camera looks at a large anvil with a blacksmith behind it and one on
either side. The smith in the middle draws a heated metal rod from the fire, places it on the anvil,
and all three begin a rhythmic hammering. After several blows, the metal goes back in the fire.
One smith pulls out a bottle of beer, and they each take a swig. Then, out comes the glowing
metal and the hammering resumes.",
   "countries" : [
      "USA"
   ],
   "released" : ISODate("1893-05-09T00:00:00.000Z"),
   "directors" : [
      "William K.L. Dickson"
   ],
   "rated" : "UNRATED",
   "awards" : {
      "wins" : 1,
      "nominations" : 0,
      "text" : "1 win."
   },
   "lastupdated" : "2015-08-26 00:03:50.133000000",
   "year" : 1893,
   "imdb" : {
      "rating" : 6.2,
      "votes" : 1189,
      "id" : 5
   },
   "type" : "movie",
   "tomatoes" : {
      "viewer" : {
         "rating" : 3,
         "numReviews" : 184,
         "meter" : 32
      },
      "lastUpdated" : ISODate("2015-06-28T18:34:09.000Z")
   }
}
```

insert_in_movies(movies_data)

4. Create Python methods and MongoDB queries to support the below operations -
   a. **comments** collection
      i. Find top 10 users who made the maximum number of comments

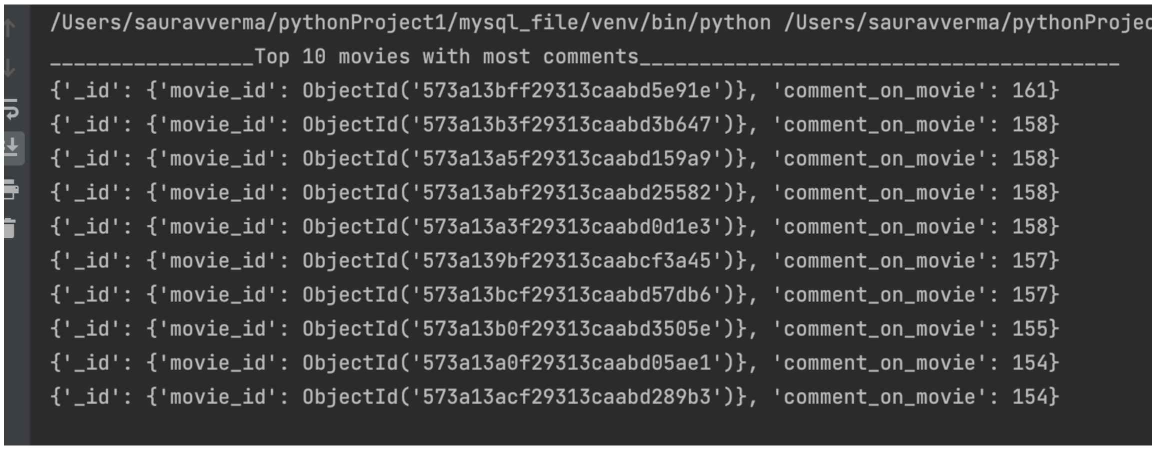
```
comment = db.get_collection('comments')
# print(comment)

print("_________________top 10 users who made the maximum number of comments___________________")
top_10_user = comment.aggregate([
  {"$project": {"_id": 0, "name": 1, 'email': 1}},
  {"$group": {"_id": {"email": "$email"}, "name": {"$first": "$name"},
"total_comment": {"$sum": 1}}},
  {"$sort": {"total_comment": -1}},
  {"$project": {"_id": 0, "name": 1, "total_comment": 1}},
  {"$limit": 10}
])
for item in list(top_10_user):
  print(item)
```



```
_________________top 10 users who made the maximum number of comments____________
{'name': 'Mace Tyrell', 'total_comment': 331}
{'name': 'Missandei', 'total_comment': 327}
{'name': 'The High Sparrow', 'total_comment': 315}
{'name': 'Sansa Stark', 'total_comment': 308}
{'name': 'Rodrik Cassel', 'total_comment': 305}
{'name': 'Thoros of Myr', 'total_comment': 304}
{'name': 'Robert Jordan', 'total_comment': 304}
{'name': 'Brienne of Tarth', 'total_comment': 302}
{'name': 'Javier Smith', 'total_comment': 296}
{'name': 'Bradley Brooks', 'total_comment': 296}

Process finished with exit code 0
```

      ii. Find top 10 movies with most comments

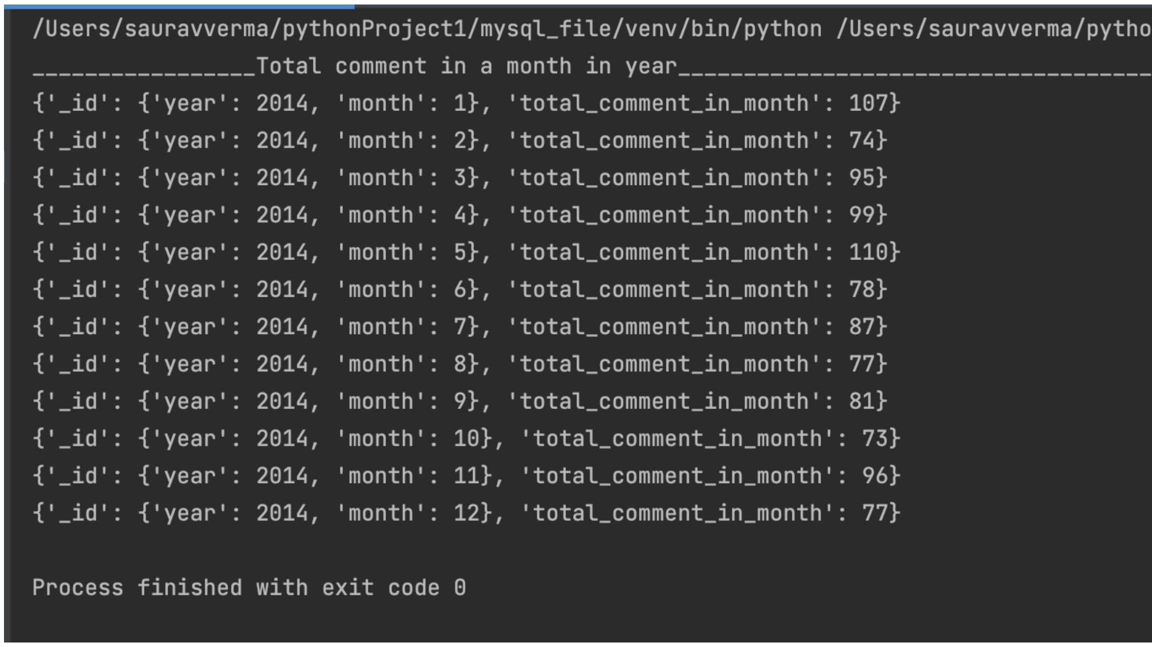
```
print("_________________Top 10 movies with most
comments__________________________________________")
most_commented_movie = comment.aggregate([
  {"$project": {"_id": 0, "movie_id": 1}},
  {"$group": {"_id": {"movie_id": "$movie_id"}, "comment_on_movie": {"$sum":
1}}},
  {"$sort": {"comment_on_movie": -1}},
  {"$project": {"comment_on_movie": 1}},
  {"$limit": 10}
])

for item in list(most_commented_movie):
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverma/pythonProjec
_________________Top 10 movies with most comments________________________________________
{'_id': {'movie_id': ObjectId('573a13bff29313caabd5e91e')}, 'comment_on_movie': 161}
{'_id': {'movie_id': ObjectId('573a13b3f29313caabd3b647')}, 'comment_on_movie': 158}
{'_id': {'movie_id': ObjectId('573a13a5f29313caabd159a9')}, 'comment_on_movie': 158}
{'_id': {'movie_id': ObjectId('573a13abf29313caabd25582')}, 'comment_on_movie': 158}
{'_id': {'movie_id': ObjectId('573a13a3f29313caabd0d1e3')}, 'comment_on_movie': 158}
{'_id': {'movie_id': ObjectId('573a139bf29313caabcf3a45')}, 'comment_on_movie': 157}
{'_id': {'movie_id': ObjectId('573a13bcf29313caabd57db6')}, 'comment_on_movie': 157}
{'_id': {'movie_id': ObjectId('573a13b0f29313caabd3505e')}, 'comment_on_movie': 155}
{'_id': {'movie_id': ObjectId('573a13a0f29313caabd05ae1')}, 'comment_on_movie': 154}
{'_id': {'movie_id': ObjectId('573a13acf29313caabd289b3')}, 'comment_on_movie': 154}
```

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

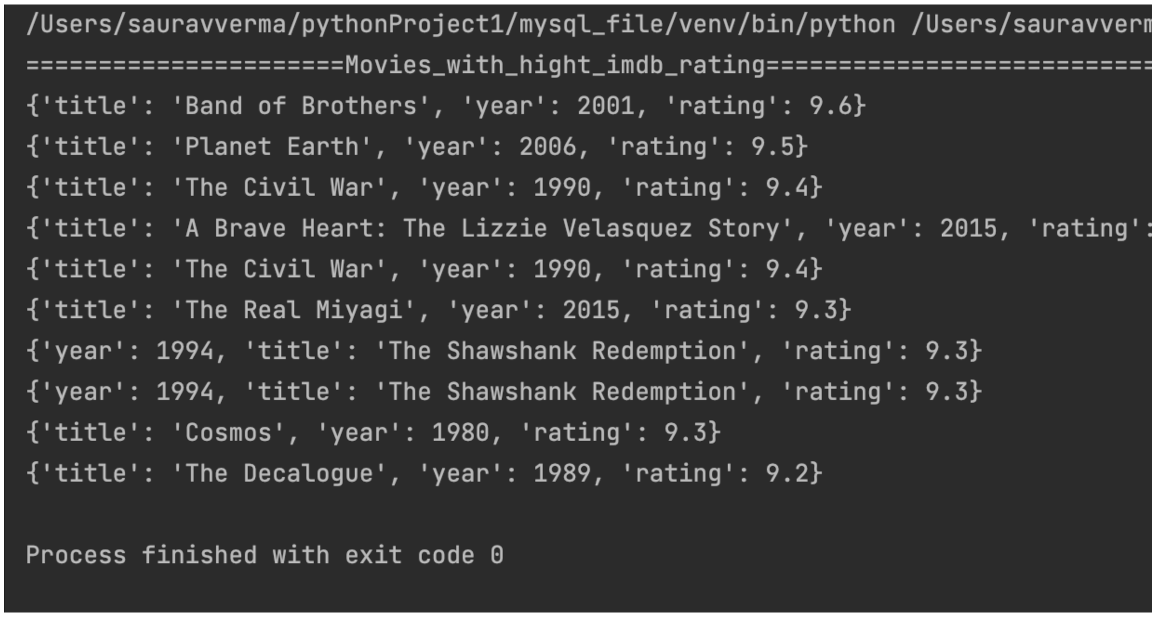
```
print("_________________Total comment in a month in
year__________________________________________")
total_comment_in_each_month_in_a_year=comment.aggregate([


{"$project":{"_id":0,"month":{"$month":"$date"},"year":{"$year":"$date"},"name":1,}
},
  {"$match":{"year":2014}},

{"$group":{"_id":{"year":"$year","month":"$month"},"total_comment_in_month":{"$
sum":1}}},
  {"$sort":{"_id":1}}
  ])
for item in list(total_comment_in_each_month_in_a_year):
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverma/pytho
_________________Total comment in a month in year______________________________________
{'_id': {'year': 2014, 'month': 1}, 'total_comment_in_month': 107}
{'_id': {'year': 2014, 'month': 2}, 'total_comment_in_month': 74}
{'_id': {'year': 2014, 'month': 3}, 'total_comment_in_month': 95}
{'_id': {'year': 2014, 'month': 4}, 'total_comment_in_month': 99}
{'_id': {'year': 2014, 'month': 5}, 'total_comment_in_month': 110}
{'_id': {'year': 2014, 'month': 6}, 'total_comment_in_month': 78}
{'_id': {'year': 2014, 'month': 7}, 'total_comment_in_month': 87}
{'_id': {'year': 2014, 'month': 8}, 'total_comment_in_month': 77}
{'_id': {'year': 2014, 'month': 9}, 'total_comment_in_month': 81}
{'_id': {'year': 2014, 'month': 10}, 'total_comment_in_month': 73}
{'_id': {'year': 2014, 'month': 11}, 'total_comment_in_month': 96}
{'_id': {'year': 2014, 'month': 12}, 'total_comment_in_month': 77}

Process finished with exit code 0
```

b. **movies** collection
   i. Find top `N` movies -
      1. with the highest IMDB rating

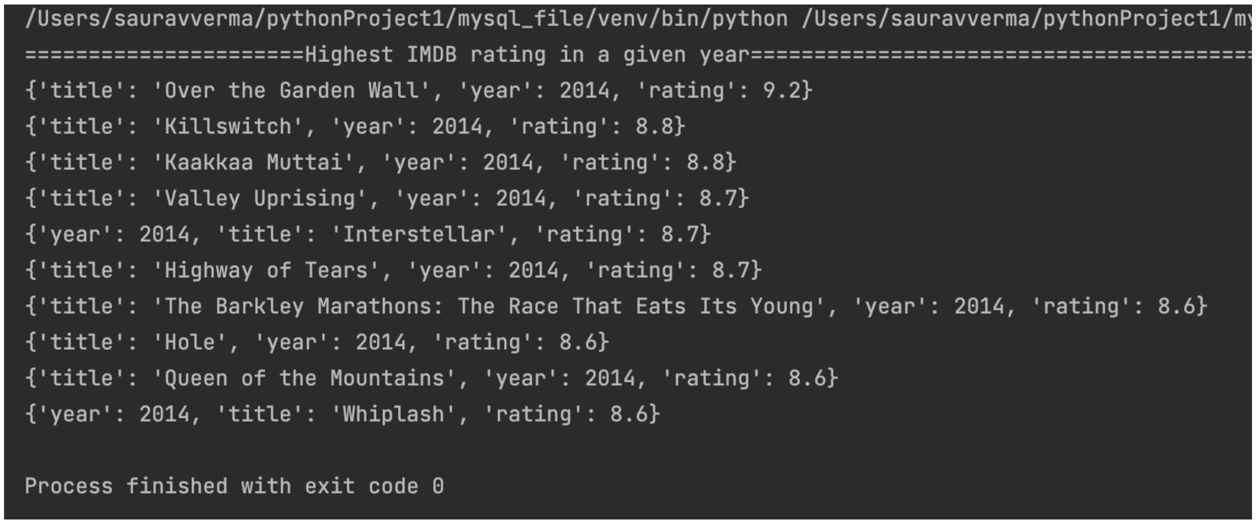
```
movies = db.get_collection('movies')
print("=====================Movies_with_hight_imdb_rating===========
==============================")
movies_with_hight_imdb_rating=movies.aggregate([

{"$project":{"_id":0,"title":1,"year":1,"rating":{"$convert":{"input":"$imdb.rating","to":
"double","onError":0.00}}}},
  {"$sort":{"rating":-1}},
  {"$limit":10}
 ])

for item in movies_with_hight_imdb_rating:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverm
=====================Movies_with_hight_imdb_rating=========================
{'title': 'Band of Brothers', 'year': 2001, 'rating': 9.6}
{'title': 'Planet Earth', 'year': 2006, 'rating': 9.5}
{'title': 'The Civil War', 'year': 1990, 'rating': 9.4}
{'title': 'A Brave Heart: The Lizzie Velasquez Story', 'year': 2015, 'rating':
{'title': 'The Civil War', 'year': 1990, 'rating': 9.4}
{'title': 'The Real Miyagi', 'year': 2015, 'rating': 9.3}
{'year': 1994, 'title': 'The Shawshank Redemption', 'rating': 9.3}
{'year': 1994, 'title': 'The Shawshank Redemption', 'rating': 9.3}
{'title': 'Cosmos', 'year': 1980, 'rating': 9.3}
{'title': 'The Decalogue', 'year': 1989, 'rating': 9.2}

Process finished with exit code 0
```

2. with the highest IMDB rating in a given year

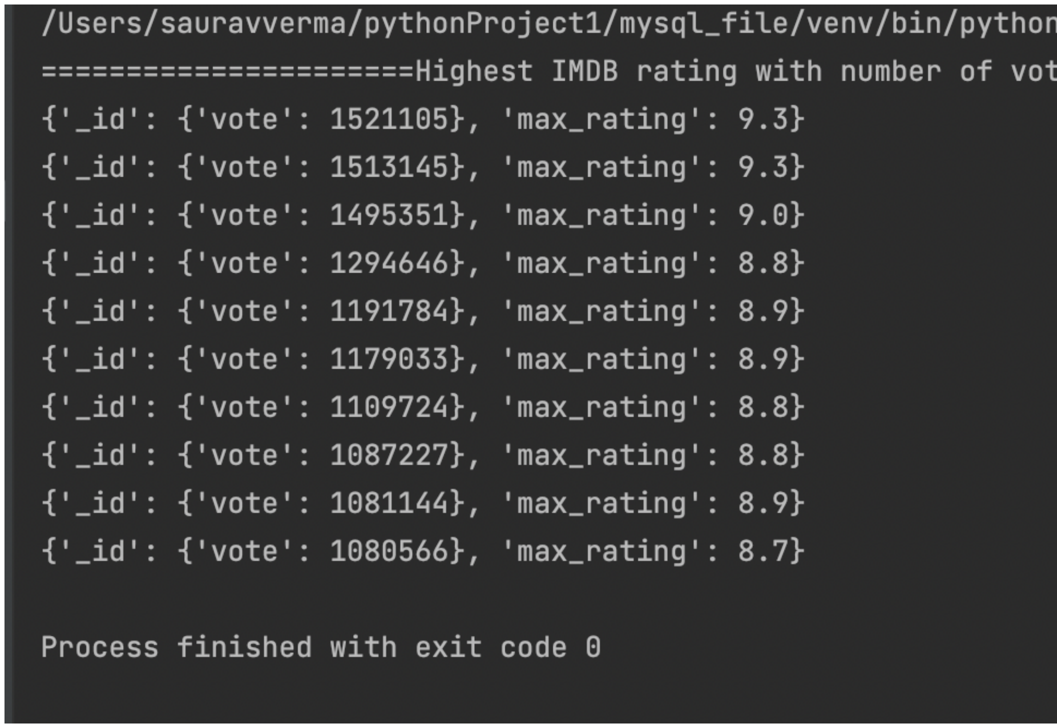
```
print("=====================Highest IMDB rating in a given
year==========================================")
highest_IMDB_rating_in_a_given_year=movies.aggregate([
 {"$match":{"year":2014}},

{"$project":{"_id":0,"title":1,"year":1,"rating":{"$convert":{"input":"$imdb.rati
ng","to":"double","onError":0.00}}}},
 {"$sort":{"rating":-1}},
 {"$limit":10}
 ])
for item in highest_IMDB_rating_in_a_given_year:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverma/pythonProject1/my
=====================Highest IMDB rating in a given year=====================================
{'title': 'Over the Garden Wall', 'year': 2014, 'rating': 9.2}
{'title': 'Killswitch', 'year': 2014, 'rating': 8.8}
{'title': 'Kaakkaa Muttai', 'year': 2014, 'rating': 8.8}
{'title': 'Valley Uprising', 'year': 2014, 'rating': 8.7}
{'year': 2014, 'title': 'Interstellar', 'rating': 8.7}
{'title': 'Highway of Tears', 'year': 2014, 'rating': 8.7}
{'title': 'The Barkley Marathons: The Race That Eats Its Young', 'year': 2014, 'rating': 8.6}
{'title': 'Hole', 'year': 2014, 'rating': 8.6}
{'title': 'Queen of the Mountains', 'year': 2014, 'rating': 8.6}
{'year': 2014, 'title': 'Whiplash', 'rating': 8.6}

Process finished with exit code 0
```

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

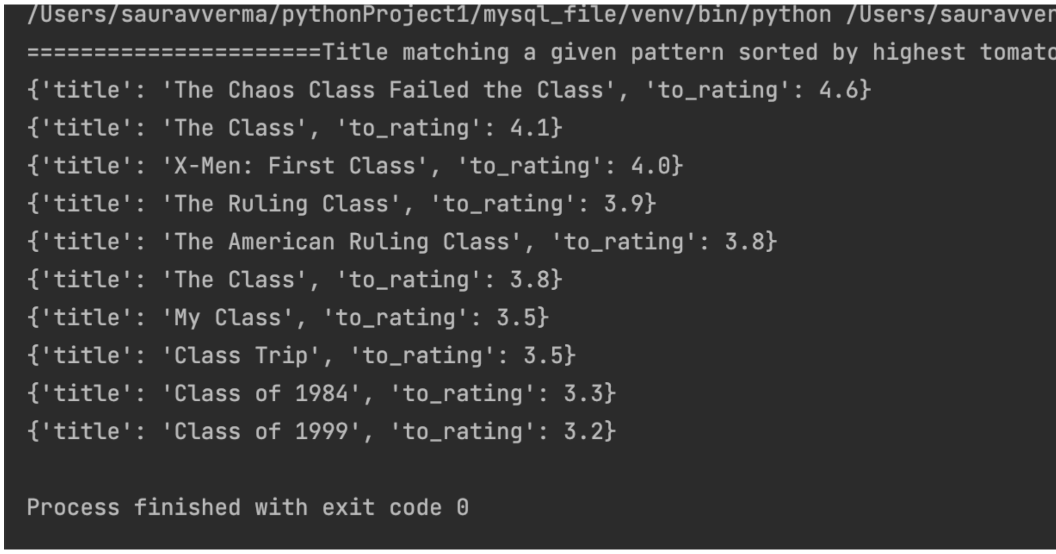
```
print("=====================Highest IMDB rating with number of votes>1000=================================")
highest_IMDB_rating_with_number_of_votes_gt_1000=movies.aggregate([

{"$project":{"_id":0,"voting":{"$convert":{"input":"$imdb.votes","to":"int","onError":0}},"rating":{"$convert":{"input":"$imdb.rating","to":"double","onError":0.00}}}},
 {"$sort":{"rating":-1}},
 {"$match":{"voting":{"$gt":1000}}},
 {"$group":{"_id":{"vote":"$voting"},"max_rating":{"$max":"$rating"}}},
 {"$sort":{"_id.vote":-1}},
 {"$limit":10},
 ])
for item in highest_IMDB_rating_with_number_of_votes_gt_1000:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python
=====================Highest IMDB rating with number of vot
{'_id': {'vote': 1521105}, 'max_rating': 9.3}
{'_id': {'vote': 1513145}, 'max_rating': 9.3}
{'_id': {'vote': 1495351}, 'max_rating': 9.0}
{'_id': {'vote': 1294646}, 'max_rating': 8.8}
{'_id': {'vote': 1191784}, 'max_rating': 8.9}
{'_id': {'vote': 1179033}, 'max_rating': 8.9}
{'_id': {'vote': 1109724}, 'max_rating': 8.8}
{'_id': {'vote': 1087227}, 'max_rating': 8.8}
{'_id': {'vote': 1081144}, 'max_rating': 8.9}
{'_id': {'vote': 1080566}, 'max_rating': 8.7}

Process finished with exit code 0
```

4. with title matching a given pattern sorted by highest tomatoes ratings

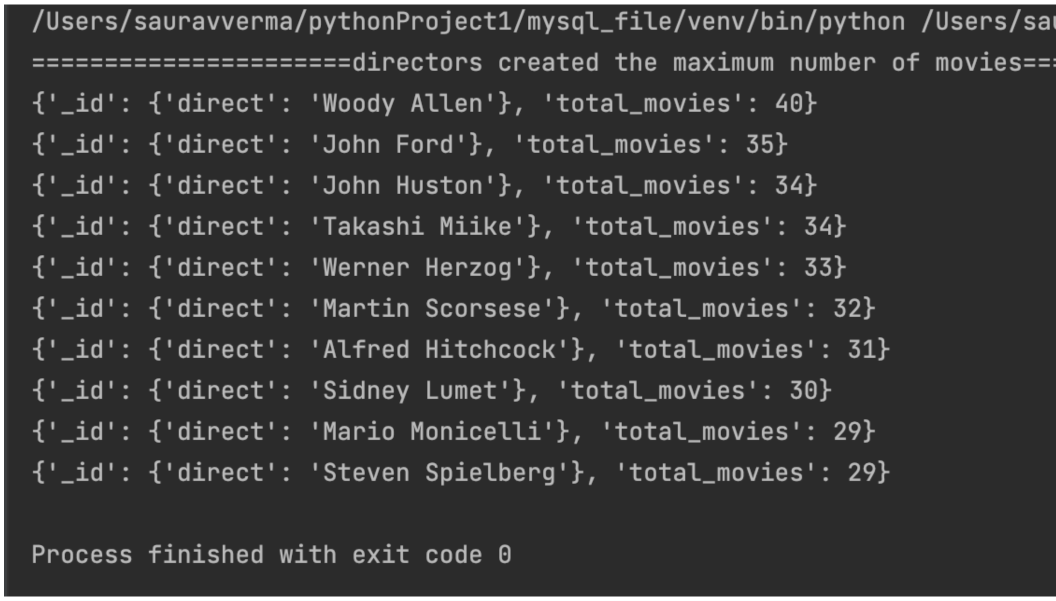
```
print("=====================Title matching a given pattern sorted by highest tomatoes ratings==============")
title_matching_pattern_sort_highest_to_ratings=movies.aggregate([
 {"$match":{"title":{"$regex":'Class'}}},
{"$project":{"_id":0,"title":1,"to_rating":{"$convert":{"input":"$tomatoes.vie
wer.rating","to":"double","onError":0.0,"onNull":0.0}}}},
 {"$sort":{"to_rating":-1}},
 {"$limit":10}
])
for item in title_matching_pattern_sort_highest_to_ratings:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravver
=====================Title matching a given pattern sorted by highest tomato
{'title': 'The Chaos Class Failed the Class', 'to_rating': 4.6}
{'title': 'The Class', 'to_rating': 4.1}
{'title': 'X-Men: First Class', 'to_rating': 4.0}
{'title': 'The Ruling Class', 'to_rating': 3.9}
{'title': 'The American Ruling Class', 'to_rating': 3.8}
{'title': 'The Class', 'to_rating': 3.8}
{'title': 'My Class', 'to_rating': 3.5}
{'title': 'Class Trip', 'to_rating': 3.5}
{'title': 'Class of 1984', 'to_rating': 3.3}
{'title': 'Class of 1999', 'to_rating': 3.2}

Process finished with exit code 0
```

ii. Find top `N` directors -
   1. who created the maximum number of movies

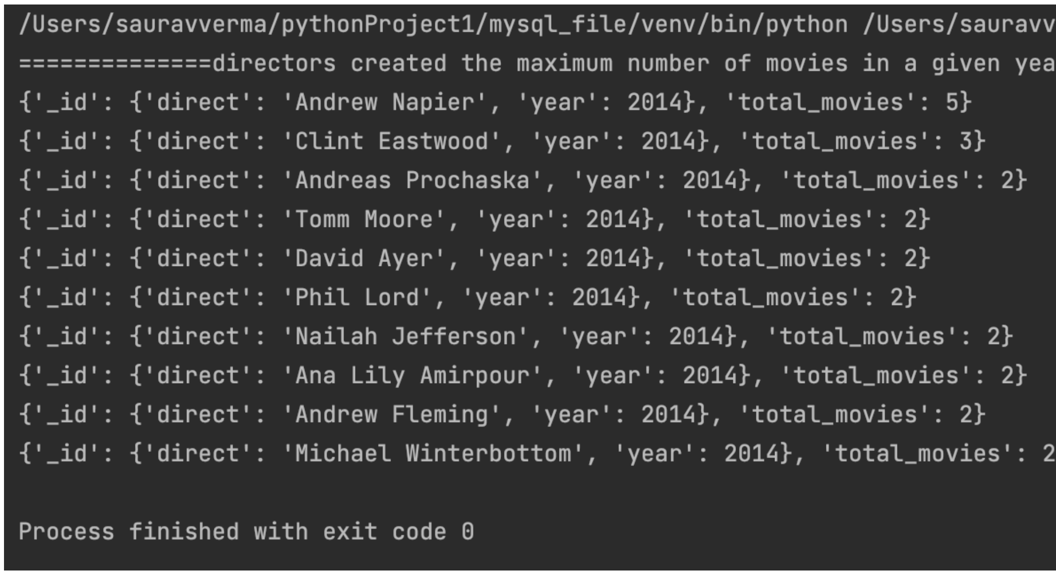
```
print("======================directors created the maximum
number of movies===============")
dr_created_the_maximum_number_of_movies=movies.aggregate([
    {"$unwind":"$directors"},

{"$project":{"_id":0,"movie_dir":{"$convert":{"input":"$directors","to":"string
","onError":"unkonwn","onNull":"unkonwn"}},"title":1}},
    {"$group":{"_id":{"direct":"$movie_dir"},"total_movies":{"$sum":1}}},
    {"$sort":{"total_movies":-1}},
    {"$limit":10}
   ])
for item in dr_created_the_maximum_number_of_movies:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sa
====================directors created the maximum number of movies===
{'_id': {'direct': 'Woody Allen'}, 'total_movies': 40}
{'_id': {'direct': 'John Ford'}, 'total_movies': 35}
{'_id': {'direct': 'John Huston'}, 'total_movies': 34}
{'_id': {'direct': 'Takashi Miike'}, 'total_movies': 34}
{'_id': {'direct': 'Werner Herzog'}, 'total_movies': 33}
{'_id': {'direct': 'Martin Scorsese'}, 'total_movies': 32}
{'_id': {'direct': 'Alfred Hitchcock'}, 'total_movies': 31}
{'_id': {'direct': 'Sidney Lumet'}, 'total_movies': 30}
{'_id': {'direct': 'Mario Monicelli'}, 'total_movies': 29}
{'_id': {'direct': 'Steven Spielberg'}, 'total_movies': 29}

Process finished with exit code 0
```

2. who created the maximum number of movies in a given year

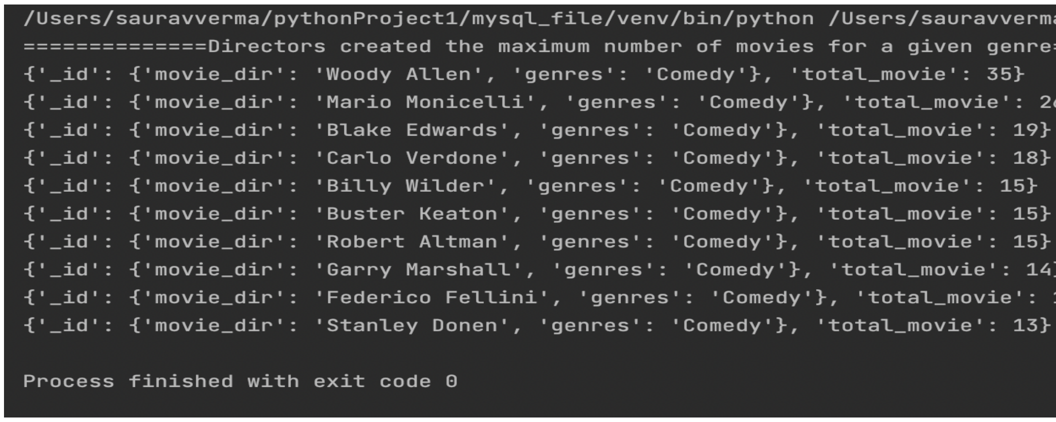
```
print("==============directors created the maximum number of movies in a given year==========")
dr_created_the_maximum_number_of_movies_in_a_year=movies.aggregate([
   {"$unwind":"$directors"},
  {"$match":{"year":2014}},

{"$project":{"_id":0,"year":1,"movie_dir":{"$convert":{"input":"$directors","to":"string","onError":"unkonwn","onNull":"unkonwn"}},"title":1}},

{"$group":{"_id":{"direct":"$movie_dir","year":"$year"},"total_movies":{"$sum":1}}},
   {"$sort":{"total_movies":-1}},
   {"$limit":10}
  ])
for item in dr_created_the_maximum_number_of_movies_in_a_year:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravv
=============directors created the maximum number of movies in a given yea
{'_id': {'direct': 'Andrew Napier', 'year': 2014}, 'total_movies': 5}
{'_id': {'direct': 'Clint Eastwood', 'year': 2014}, 'total_movies': 3}
{'_id': {'direct': 'Andreas Prochaska', 'year': 2014}, 'total_movies': 2}
{'_id': {'direct': 'Tomm Moore', 'year': 2014}, 'total_movies': 2}
{'_id': {'direct': 'David Ayer', 'year': 2014}, 'total_movies': 2}
{'_id': {'direct': 'Phil Lord', 'year': 2014}, 'total_movies': 2}
{'_id': {'direct': 'Nailah Jefferson', 'year': 2014}, 'total_movies': 2}
{'_id': {'direct': 'Ana Lily Amirpour', 'year': 2014}, 'total_movies': 2}
{'_id': {'direct': 'Andrew Fleming', 'year': 2014}, 'total_movies': 2}
{'_id': {'direct': 'Michael Winterbottom', 'year': 2014}, 'total_movies': 2

Process finished with exit code 0
```

3. who created the maximum number of movies for a given genre

```
int("==============Directors created the maximum number of movies for a given genre==========")
dr_created_the_maximum_number_of_movies_for_a_genre=movies.aggregate([
    {"$unwind":"$genres"},
    {"$unwind":"$directors"},
    {"$match":{"genres":"Comedy"}},
    {"$project":{"genres":1,"directors":1,"title":1}},

{"$group":{"_id":{"movie_dir":"$directors","genres":"$genres"},"total_movie":{"$sum":1}}},
    {"$sort":{"total_movie":-1}},
    {"$limit":10}
   ])
for item in dr_created_the_maximum_number_of_movies_for_a_genre:
  print(item)
```

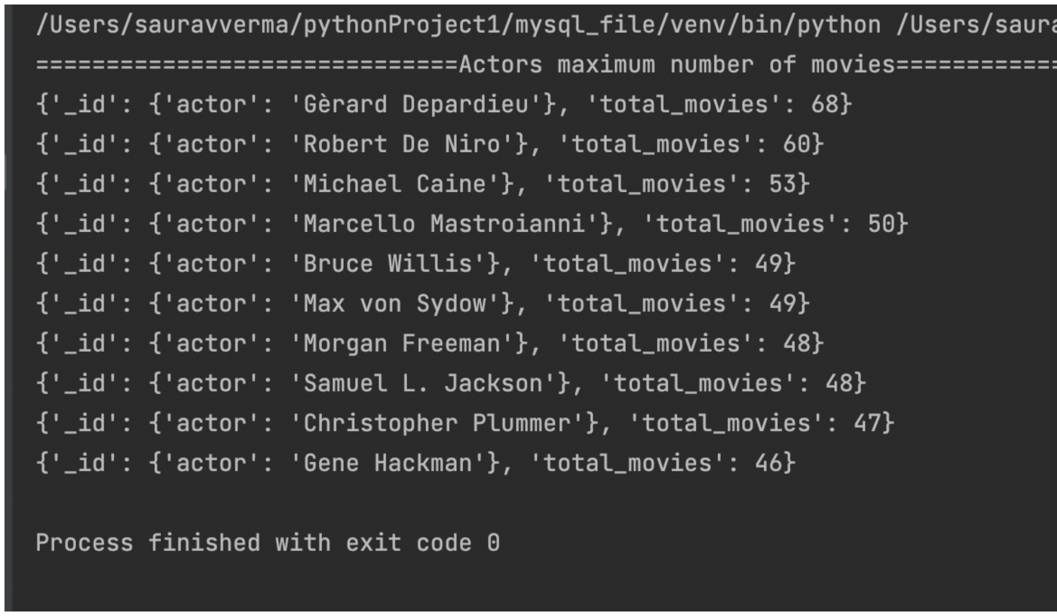


```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverma
==============Directors created the maximum number of movies for a given genre
{'_id': {'movie_dir': 'Woody Allen', 'genres': 'Comedy'}, 'total_movie': 35}
{'_id': {'movie_dir': 'Mario Monicelli', 'genres': 'Comedy'}, 'total_movie': 2
{'_id': {'movie_dir': 'Blake Edwards', 'genres': 'Comedy'}, 'total_movie': 19}
{'_id': {'movie_dir': 'Carlo Verdone', 'genres': 'Comedy'}, 'total_movie': 18}
{'_id': {'movie_dir': 'Billy Wilder', 'genres': 'Comedy'}, 'total_movie': 15}
{'_id': {'movie_dir': 'Buster Keaton', 'genres': 'Comedy'}, 'total_movie': 15}
{'_id': {'movie_dir': 'Robert Altman', 'genres': 'Comedy'}, 'total_movie': 15}
{'_id': {'movie_dir': 'Garry Marshall', 'genres': 'Comedy'}, 'total_movie': 14
{'_id': {'movie_dir': 'Federico Fellini', 'genres': 'Comedy'}, 'total_movie': 1
{'_id': {'movie_dir': 'Stanley Donen', 'genres': 'Comedy'}, 'total_movie': 13}

Process finished with exit code 0
```

iii. Find top `N` actors -

1. who starred in the maximum number of movies

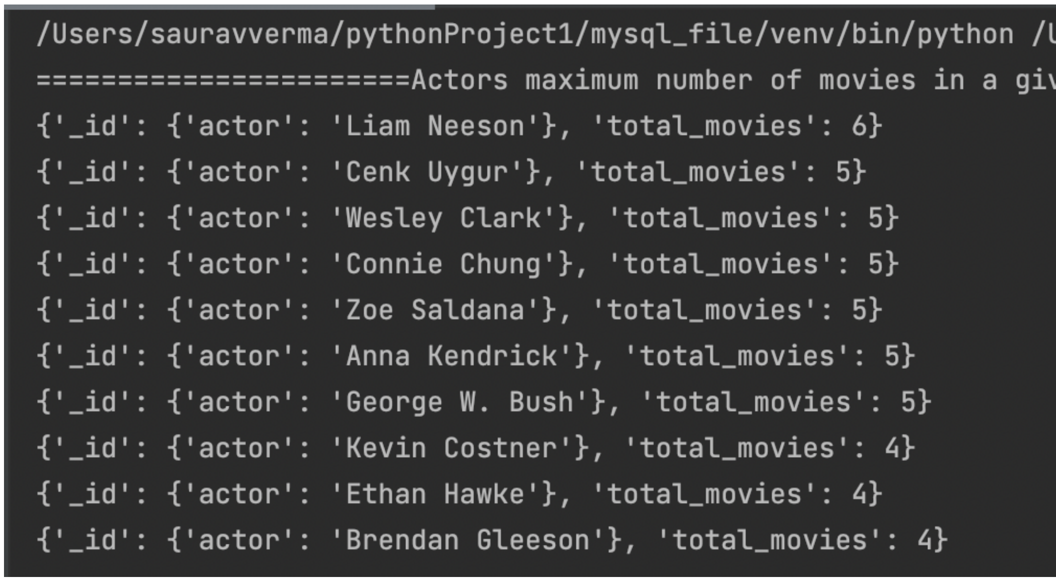
```
print("=============================Actors maximum number of
movies===========================")
ac_maximum_number_of_movies=movies.aggregate([
  {"$unwind":'$cast'},
  {"$project":{"_id":0,"title":1,"cast":1}},
  {"$group":{"_id":{"actor":"$cast"},"total_movies":{"$sum":1}}},
  {"$sort":{"total_movies":-1}},
  {"$limit":10},
])

for item in ac_maximum_number_of_movies:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/saura
=============================Actors maximum number of movies===========
{'_id': {'actor': 'Gèrard Depardieu'}, 'total_movies': 68}
{'_id': {'actor': 'Robert De Niro'}, 'total_movies': 60}
{'_id': {'actor': 'Michael Caine'}, 'total_movies': 53}
{'_id': {'actor': 'Marcello Mastroianni'}, 'total_movies': 50}
{'_id': {'actor': 'Bruce Willis'}, 'total_movies': 49}
{'_id': {'actor': 'Max von Sydow'}, 'total_movies': 49}
{'_id': {'actor': 'Morgan Freeman'}, 'total_movies': 48}
{'_id': {'actor': 'Samuel L. Jackson'}, 'total_movies': 48}
{'_id': {'actor': 'Christopher Plummer'}, 'total_movies': 47}
{'_id': {'actor': 'Gene Hackman'}, 'total_movies': 46}

Process finished with exit code 0
```

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

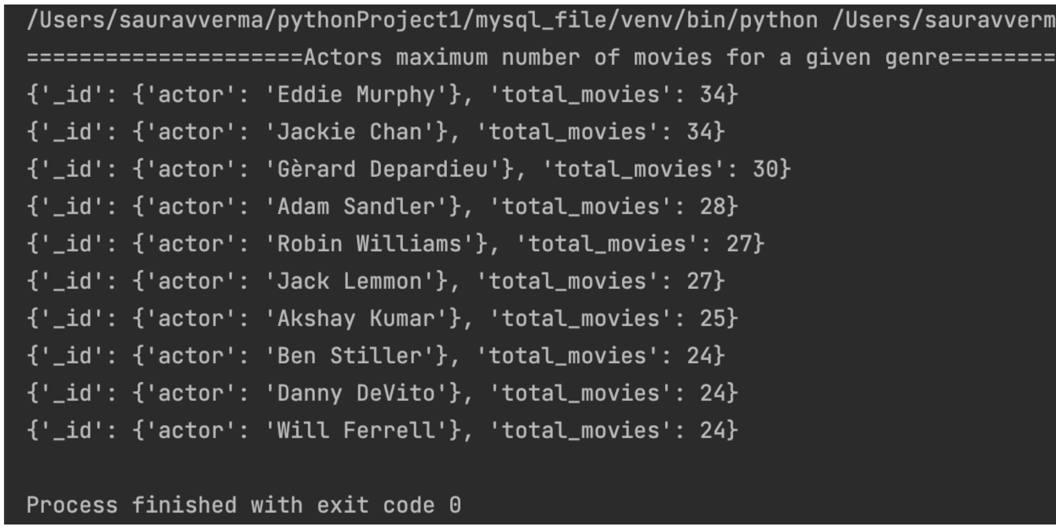
```
rint("======================Actors maximum number of movies in
a given year====================")
ac_maximum_number_of_movies_in_a_given_year=movies.aggregate([
   {"$unwind":'$cast'},
   {"$match":{"year":2014}},
   {"$project":{"_id":0,"year":1,"title":1,"cast":1}},
   {"$group":{"_id":{"actor":"$cast"},"total_movies":{"$sum":1}}},
   {"$sort":{"total_movies":-1}},
   {"$limit":10},

])

for item in ac_maximum_number_of_movies_in_a_given_year:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /U
=====================Actors maximum number of movies in a giv
{'_id': {'actor': 'Liam Neeson'}, 'total_movies': 6}
{'_id': {'actor': 'Cenk Uygur'}, 'total_movies': 5}
{'_id': {'actor': 'Wesley Clark'}, 'total_movies': 5}
{'_id': {'actor': 'Connie Chung'}, 'total_movies': 5}
{'_id': {'actor': 'Zoe Saldana'}, 'total_movies': 5}
{'_id': {'actor': 'Anna Kendrick'}, 'total_movies': 5}
{'_id': {'actor': 'George W. Bush'}, 'total_movies': 5}
{'_id': {'actor': 'Kevin Costner'}, 'total_movies': 4}
{'_id': {'actor': 'Ethan Hawke'}, 'total_movies': 4}
{'_id': {'actor': 'Brendan Gleeson'}, 'total_movies': 4}
```

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

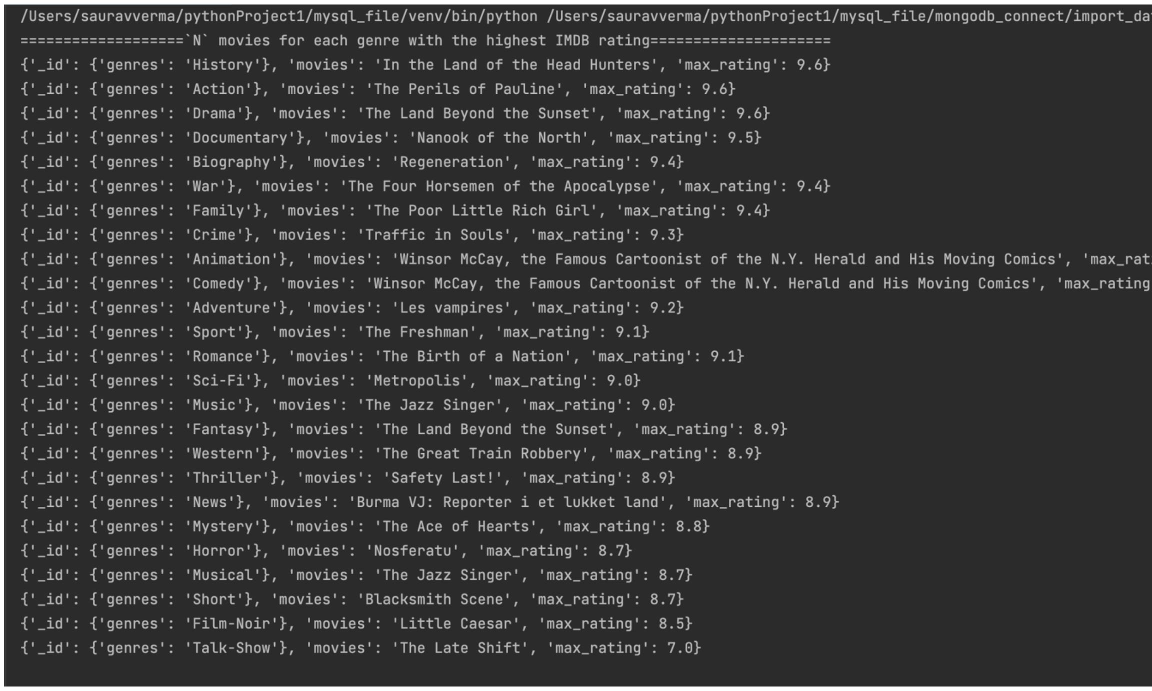
```
print("====================Actors maximum number of movies for a given genre====================")
ac_maximum_number_of_movies_for_a_given_genre=movies.aggregate([
   {"$unwind":'$cast'},
   {"$unwind":'$genres'},
   {"$match":{"genres":"Comedy"}},
   {"$project":{"_id":0,"genres":1,"title":1,"cast":1}},
   {"$group":{"_id":{"actor":"$cast"},"total_movies":{"$sum":1}}},
   {"$sort":{"total_movies":-1}},
   {"$limit":10},
])

for item in ac_maximum_number_of_movies_for_a_given_genre:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverm
====================Actors maximum number of movies for a given genre========
{'_id': {'actor': 'Eddie Murphy'}, 'total_movies': 34}
{'_id': {'actor': 'Jackie Chan'}, 'total_movies': 34}
{'_id': {'actor': 'Gèrard Depardieu'}, 'total_movies': 30}
{'_id': {'actor': 'Adam Sandler'}, 'total_movies': 28}
{'_id': {'actor': 'Robin Williams'}, 'total_movies': 27}
{'_id': {'actor': 'Jack Lemmon'}, 'total_movies': 27}
{'_id': {'actor': 'Akshay Kumar'}, 'total_movies': 25}
{'_id': {'actor': 'Ben Stiller'}, 'total_movies': 24}
{'_id': {'actor': 'Danny DeVito'}, 'total_movies': 24}
{'_id': {'actor': 'Will Ferrell'}, 'total_movies': 24}

Process finished with exit code 0
```

iv. Find top `N` movies for each genre with the highest IMDB rating

```
print("==================`N` movies for each genre with the highest IMDB
rating====================")
movies_for_each_genre_with_the_highest_IMDB_rating=movies.aggregate([
   {"$unwind":'$genres'},
   {"$project":{"_id":0,"genres":1,"title":1,
       "rating":{"$convert":{"input":"$imdb.rating","to":"double","onError":0.00}}}},

{"$group":{"_id":{"genres":"$genres"},"movies":{"$first":"$title"},"max_rating":{"$m
ax":"$rating"}}},
   {"$sort":{"max_rating":-1}},
])
for item in movies_for_each_genre_with_the_highest_IMDB_rating:
  print(item)
```

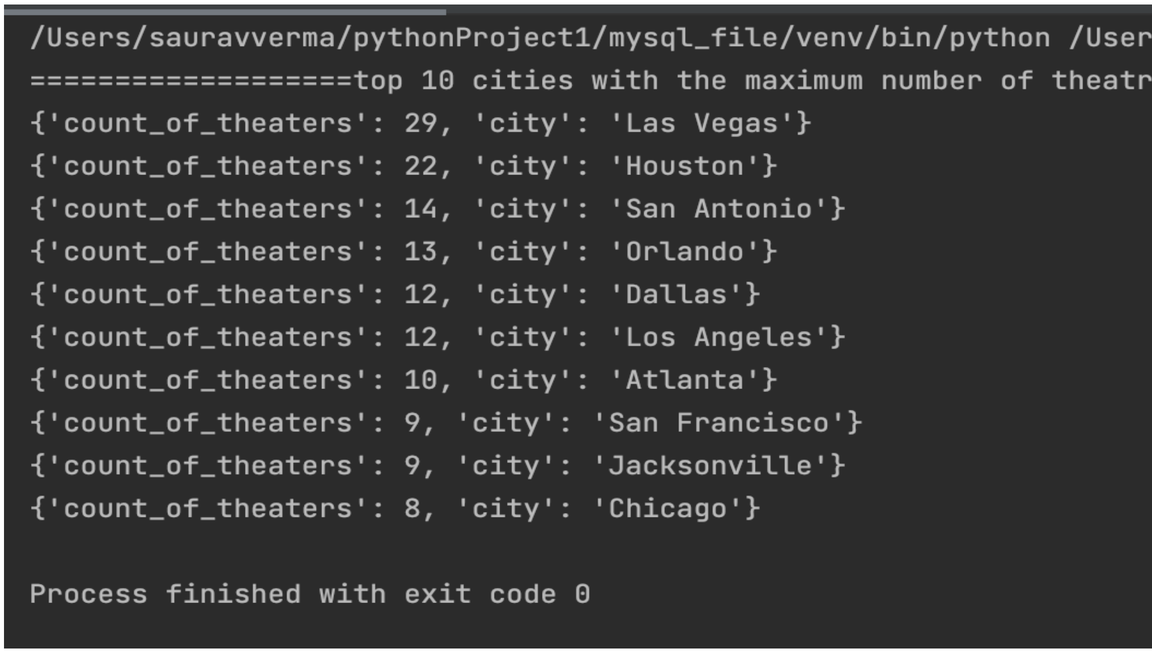


```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverma/pythonProject1/mysql_file/mongodb_connect/import_da
==================`N` movies for each genre with the highest IMDB rating====================
{'_id': {'genres': 'History'}, 'movies': 'In the Land of the Head Hunters', 'max_rating': 9.6}
{'_id': {'genres': 'Action'}, 'movies': 'The Perils of Pauline', 'max_rating': 9.6}
{'_id': {'genres': 'Drama'}, 'movies': 'The Land Beyond the Sunset', 'max_rating': 9.6}
{'_id': {'genres': 'Documentary'}, 'movies': 'Nanook of the North', 'max_rating': 9.5}
{'_id': {'genres': 'Biography'}, 'movies': 'Regeneration', 'max_rating': 9.4}
{'_id': {'genres': 'War'}, 'movies': 'The Four Horsemen of the Apocalypse', 'max_rating': 9.4}
{'_id': {'genres': 'Family'}, 'movies': 'The Poor Little Rich Girl', 'max_rating': 9.4}
{'_id': {'genres': 'Crime'}, 'movies': 'Traffic in Souls', 'max_rating': 9.3}
{'_id': {'genres': 'Animation'}, 'movies': 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comics', 'max_rat
{'_id': {'genres': 'Comedy'}, 'movies': 'Winsor McCay, the Famous Cartoonist of the N.Y. Herald and His Moving Comics', 'max_rating
{'_id': {'genres': 'Adventure'}, 'movies': 'Les vampires', 'max_rating': 9.2}
{'_id': {'genres': 'Sport'}, 'movies': 'The Freshman', 'max_rating': 9.1}
{'_id': {'genres': 'Romance'}, 'movies': 'The Birth of a Nation', 'max_rating': 9.1}
{'_id': {'genres': 'Sci-Fi'}, 'movies': 'Metropolis', 'max_rating': 9.0}
{'_id': {'genres': 'Music'}, 'movies': 'The Jazz Singer', 'max_rating': 9.0}
{'_id': {'genres': 'Fantasy'}, 'movies': 'The Land Beyond the Sunset', 'max_rating': 8.9}
{'_id': {'genres': 'Western'}, 'movies': 'The Great Train Robbery', 'max_rating': 8.9}
{'_id': {'genres': 'Thriller'}, 'movies': 'Safety Last!', 'max_rating': 8.9}
{'_id': {'genres': 'News'}, 'movies': 'Burma VJ: Reporter i et lukket land', 'max_rating': 8.9}
{'_id': {'genres': 'Mystery'}, 'movies': 'The Ace of Hearts', 'max_rating': 8.8}
{'_id': {'genres': 'Horror'}, 'movies': 'Nosferatu', 'max_rating': 8.7}
{'_id': {'genres': 'Musical'}, 'movies': 'The Jazz Singer', 'max_rating': 8.7}
{'_id': {'genres': 'Short'}, 'movies': 'Blacksmith Scene', 'max_rating': 8.7}
{'_id': {'genres': 'Film-Noir'}, 'movies': 'Little Caesar', 'max_rating': 8.5}
{'_id': {'genres': 'Talk-Show'}, 'movies': 'The Late Shift', 'max_rating': 7.0}
```

c. **theatre** collection

   i. Top 10 cities with the maximum number of theatres

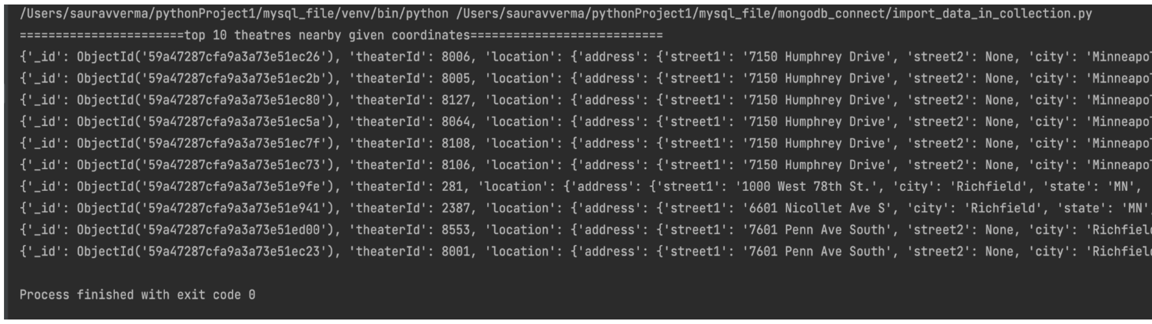
```
theaters=db.get_collection('theaters')
print("==================top 10 cities with the maximum number of
theatres====================")
1.
cities_with_the_maximum_number_of_theatres=theaters.aggregate([
  {"$project":{"_id":0,"city":"$location.address.city"}},
  {"$group":{"_id":{"city":"$city"},"count_of_theaters":{"$sum":1}}},
  {"$project":{"_id":0,"city":"$_id.city","count_of_theaters":1}},
  {"$sort":{"count_of_theaters":-1}},
  {"$limit":10}
])
for item in cities_with_the_maximum_number_of_theatres:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /User
===================top 10 cities with the maximum number of theatr
{'count_of_theaters': 29, 'city': 'Las Vegas'}
{'count_of_theaters': 22, 'city': 'Houston'}
{'count_of_theaters': 14, 'city': 'San Antonio'}
{'count_of_theaters': 13, 'city': 'Orlando'}
{'count_of_theaters': 12, 'city': 'Dallas'}
{'count_of_theaters': 12, 'city': 'Los Angeles'}
{'count_of_theaters': 10, 'city': 'Atlanta'}
{'count_of_theaters': 9, 'city': 'San Francisco'}
{'count_of_theaters': 9, 'city': 'Jacksonville'}
{'count_of_theaters': 8, 'city': 'Chicago'}

Process finished with exit code 0
```

ii. top 10 theatres nearby given coordinates

```
print("======================top 10 theatres nearby given coordinates==========================")
theatres_nearby_given_coordinates=theaters.find({
          "location.geo":{
                 "$near":{
                    "$geometry":{
                       "type":"Point",
                       "coordinates":[-93.24565,44.85466]
                       },
                       "$maxDistance":10000,
                       "$minDistance":100
                     }
                   }
}).limit(10)

for item in theatres_nearby_given_coordinates:
  print(item)
```



```
/Users/sauravverma/pythonProject1/mysql_file/venv/bin/python /Users/sauravverma/pythonProject1/mysql_file/mongodb_connect/import_data_in_collection.py
======================top 10 theatres nearby given coordinates=========================
{'_id': ObjectId('59a47287cfa9a3a73e51ec26'), 'theaterId': 8006, 'location': {'address': {'street1': '7150 Humphrey Drive', 'street2': None, 'city': 'Minneapo
{'_id': ObjectId('59a47287cfa9a3a73e51ec2b'), 'theaterId': 8005, 'location': {'address': {'street1': '7150 Humphrey Drive', 'street2': None, 'city': 'Minneapo
{'_id': ObjectId('59a47287cfa9a3a73e51ec80'), 'theaterId': 8127, 'location': {'address': {'street1': '7150 Humphrey Drive', 'street2': None, 'city': 'Minneapo
{'_id': ObjectId('59a47287cfa9a3a73e51ec5a'), 'theaterId': 8064, 'location': {'address': {'street1': '7150 Humphrey Drive', 'street2': None, 'city': 'Minneapo
{'_id': ObjectId('59a47287cfa9a3a73e51ec7f'), 'theaterId': 8108, 'location': {'address': {'street1': '7150 Humphrey Drive', 'street2': None, 'city': 'Minneapo
{'_id': ObjectId('59a47287cfa9a3a73e51ec73'), 'theaterId': 8106, 'location': {'address': {'street1': '7150 Humphrey Drive', 'street2': None, 'city': 'Minneapo
{'_id': ObjectId('59a47287cfa9a3a73e51e9fe'), 'theaterId': 281, 'location': {'address': {'street1': '1000 West 78th St.', 'city': 'Richfield', 'state': 'MN',
{'_id': ObjectId('59a47287cfa9a3a73e51e941'), 'theaterId': 2387, 'location': {'address': {'street1': '6601 Nicollet Ave S', 'city': 'Richfield', 'state': 'MN',
{'_id': ObjectId('59a47287cfa9a3a73e51ed00'), 'theaterId': 8553, 'location': {'address': {'street1': '7601 Penn Ave South', 'street2': None, 'city': 'Richfiel
{'_id': ObjectId('59a47287cfa9a3a73e51ec23'), 'theaterId': 8001, 'location': {'address': {'street1': '7601 Penn Ave South', 'street2': None, 'city': 'Richfiel

Process finished with exit code 0
```