Assignment-1

```
# Problem statement:
We are provided with a data set, which consists of
10 million ratings and 100,000 tag applications
applied to 10,000 movies by 72,000 users. Design
aggregate structures to represent the data.

#I have used MongoDB along With Python.

# Following is the code run on PyMongo to import
documents from three .dat files into one collection
on the Studio3T server. Studio3T is an interface to
use MongoDB.


import pymongo
import os
from pymongo import MongoClient


class MongoDB:
    def __init__(self):
        print("init")
        self.client = MongoClient("127.0.0.1",
27017)
#localhost port: 27017
        self.db = self.client["mydb"]
#database name = db
        self.post = self.db["movies"]
#collections name : movies
        print("done init")

    def import_data(self):
        # contents = self.read_file()
        # self.process_movies(contents)
        # self.update_movies(contents)
        print("imported")

    def read_file(self):
        # if
os.access("/Users/nidhiagarwal/Desktop/ml-
10M100K/ratings.dat", os.R_OK):
        #     with open("/Users/ nidhiagarwal
/Desktop/ml-10M100K/ratings.dat", "r") as fratings:
        #  return fratings.read()
        # if os.access("/Users/ nidhiagarwal
/Desktop/ml-10M100K/movies.dat", os.R_OK):
        #  with open("/Users/ nidhiagarwal
/Desktop/ml-10M100K/movies.dat", "r") as fmovies:
        #       return fmovies.read()
                if os.access("/Users/ nidhiagarwal
/Desktop/ml-10M100K/tags.dat", os.R_OK):
                    with open("/Users/ nidhiagarwal
/Desktop/ml-10M100K/tags.dat", "r") as ftags:
                        return ftags.read()

    def process_movies(self, contents):
        movies = contents.splitlines()
        keys = ["MovieID", "Title", "Genres"]
        i = 0
        list = []
        while i < len(movies):
            list.append(dict(zip(keys,
movies[i].split("::"))))
            # list[2] = float(list[2])
            list[i][keys[2]] =
list[i][keys[2]].split("|")
            self.post.insert_one(list[i])
            i += 1
        print(list[0])
        # self.update_ratings(list)
```

Queries

# 1) Write a query that finds average rating of each movie.

```
    def query_avg_ratings(self):
        pipeline = [{"$project": {"MovieID": "$MovieID", "AVG":
{"$avg": "$Ratings.Rating"}}}]
        return self.post.aggregate(pipeline)
```

# 2) Write a query that finds users who are similar to a given user (target user), the id of the target user is an input parameter. Users are similar to the target user if they rate the same movies.

```
    def query_similar_users(self, target):
        query = {}
        query["Ratings.UserID"] = target
        projection = {}
        projection["MovieID"] = u"$MovieID"
        projection["Ratings.UserID"] = u"$Ratings.UserID"
        return self.post.find(query,projection=projection)
```

# 3) Write a query that finds the number of movies in each genre.

```
def query_find_movie_by_genre(self):
        pipeline =
[{"$unwind":"$Genres"},{"$group":{"_id":"$Genres","Count":{"$
sum":1}}}]
        return self.post.aggregate(pipeline)
```

```
  def update_movies(self, contents):
        content = contents.splitlines()
        keys = ["UserID", "MovieID", "Tag", "Timestamp"]
        i = 0
        lst = []
        while i < len(content):
            lst.append(dict(zip(keys, content[i].split("::"))))
            lst[i][keys[2]] = float(lst[i][keys[2]])
            print(lst[i])
            self.post.update(
                {"MovieID": "5627"},
                {"$push": {"Tags":{"UserID": lst[i][keys[0]], "Rating": 
lst[i][keys[2]], "Timestamp": lst[i][keys[3]]}}},
            )
            i += 1

if __name__ == '__main__':
    mongodb = MongoDB()
    # mongodb.import_data()
    i = 0
    # for doc in mongodb.query_avg_ratings():
    #     print(i, doc)
    #     i += 1
    # for doc in mongodb.query_similar_users("64647"):
    #     print(i, doc["Ratings"])
    #     i += 1
    for doc in mongodb.query_find_movie_by_genre():
        print(i, doc)
        i += 1
```

# 4) Write 3 different queries of your choice to demonstrate that your data storage is working.

4.1) To find the title of a movie :
find({“MovieID”:”25”,”Title”:”$Title”})

4.2) To find the movies rated by a specific user :
find ({“Ratings.UserId”:”22”,”Movie”:”$MovieID”})

4.3) To find the tags associated with each movie:
find({"MovieID":"34","Tag":"$Tags.Tag"})