Filmlytics Project

A Visual Exploration and Statical Analysis of Data: Feature Films

Authors: George Wang and Aryan Chaudhary

I. Problem

Our goal in this project is to better understand feature films (a.k.a movies) and the factual statistics in regards to a film's details which includes but is not limited to film genre, director, main star, runtime, release date, IMDB score, votes, country of production, budget, and gross. We constructed visual models of both quantitative and qualitative analytics to explore relationships and correlations between different attributes as well as discovering specific trends. Utilizing the public datasets "Movie Industry" and "Countries of the World" from Kaggle, we were able to collect sufficient data to conduct our qualitative and quantitative analysis (Grijalva, 2021; Marek, 2018). Our strategies are outlined in four specific data analysis tasks. In our first task, we explored the top n genres based on a score-vote criteria which is score multiplied with the number of votes. The user is able to decide the n value so long as n > 0 and n <= 6000 (which is relatively the max size of the data set) for an intuitive interactive application design. A visualization of a bar graph will pop up with the selected n value to give us the distribution of genres. In our second task, we qualitatively examined the top genres produced per country and utilized a unique global map-legend data visualization technique to illustrate and present each top genre with the country location. In our third task, we analyzed the relationship between GDP found in our "Countries of the World" dataset of a country and average budget-gross ratio computed metric of films in each country. Each country was represented as a point in a scatter plot for visual observation. Finally, in our fourth data analytics task, we explored the relationship between profit and a film's score for each of the films in our movies dataset, then, utilized a profit-score metric to determine the top N directors based on this profit multiplied score criteria and illustrated such via a bar graph. The user can intuitively input the N value via our application command line so long as n > 0 and n <= 10.

II. Software Design and Implementation

A. Software Design and NoSQL-Database and Tools Used

For designing the software, we utilized MongoDB, a NoSQL Database system, along with Python via Pymongo. Pymongo is a tool that allows for connectivity and interactivity between MongoDB and Python code. For outputting, visualizing, and performing our data analysis, we used prominent Python data science libraries. More specifically, we used Matplotlib, Scikit-learn, and NumPy. Additionally, we used a Python library known as pygal to visualize some of our data in the form of an international map with a legend. Finally, we also have an extra file, "countryCodes.py", which provides a hard-coded dictionary for country to country code mappings. This is necessary for using the pygal Python library.

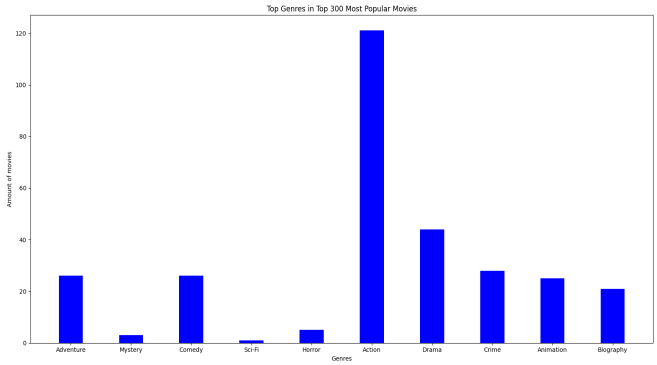
B. Parts Implemented

For the frontend/GUI of our project, we utilized a textual menu interface, allowing the user to view different facets of analysis based on a large amount of movie data. We also performed our desired querying through MongoDB's aggregation pipeline, which essentially allows for data in the document database to be retrieved through a series/sequence of operations analogous to relational algebra expressions. We analyzed the disparity of the different categories of international movies consumed through several pipeline aggregation operators like "group", "sort", and "limit". Similarly, we also limited this information to every country of origin listed in the database through complex nested grouping queries, which is a very tedious task to perform in relational databases. We also used MongoDB's "lookup" operator in order to join multiple datasets together on common attributes in order to analyze the relationship of distant attributes such as metrics for movie success and the movie's country of origin's economic success. In addition to these various data explorations, we used data science algorithms like curve fitting in order to further analyze the relationships found within our data, such as analyzing the relationship between the prodigy and the ratings of movies. The data science and analysis segment of our project is better understood below in our results and visualizations.

III. Project Outcome

A. First Data Analysis Task: Top Genres forTop N Popular Movies

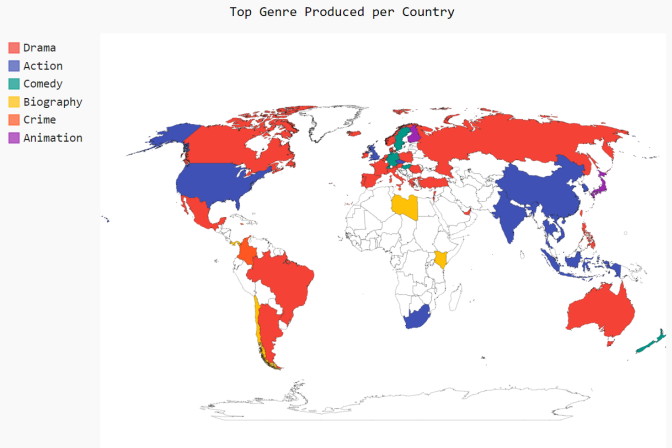
For our first data analytics task, we used the sample input of n set to 300 (n = 300) and we have the following genre distributions displayed for the top-popular genres based on our score multiplied votes metric for the top n (where n = 300 in this case) movies shown in Figure 1 below:



In Figure 1, the x axis represents the genres while the y axis is the total number of movies, so each bar would be the total number of movies in that genre based on our criterias of n and score multiplied by votes. The title of the chart would be "Top Genres in Top n Most Popular Movies" and would change depending on the user input of n in the application which is constrained by $0 < n <= 6000$. For our test input of n = 300, we have the following genres from left to right on the x-axis of the figure: Sci-Fi, Horror, Action, Drama, Crime, Mystery, Biography, Animation, Adventure, and Comedy. Shown in this distribution, Action is the genre that has the largest presence opposed to other genres when our n is set to 300. This is followed by Drama which is less than half of the amount Action represents. This chart shows that action represents more than a third of the top 300 most popular movies as the genre.

B. Second Data Analysis Task: Top Genre Produced per Country

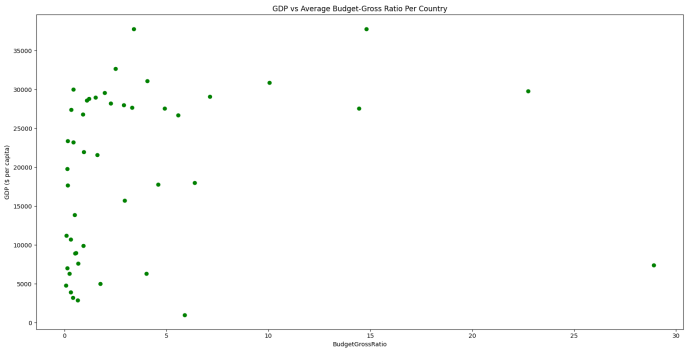
In our second Data Analysis task, we output via a .svg file in the application source repository that is a visualization of the top genre produced per country. Figure 2 below shows the output file illustration:



As shown in Figure 2, we observe that action (shown via the color blue) and drama (shown via the color red) tend to be the most prevalent genres throughout the globe. An example of some notable observations: The United States, India, and China have action as the top genre, while Canada, Russia, Australia, and Brazil have Drama as their top genre.

C. Third Data Analysis Task: GDP vs Budget-Gross Ratio

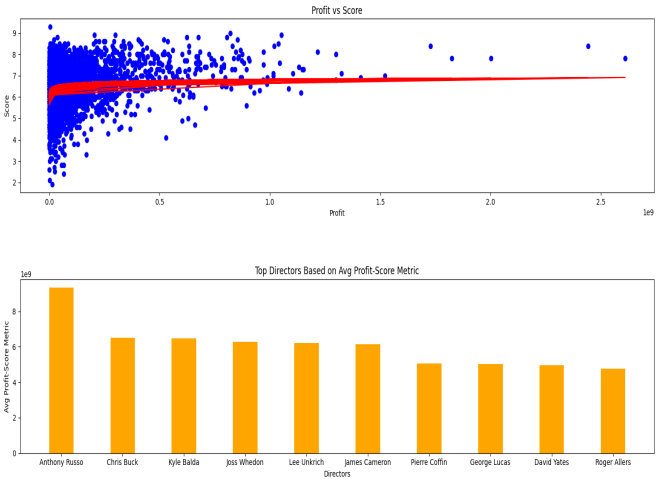
In our third data analysis task, we explore the relationship between GDP per capita of a country taken from the "Countries of the World" dataset to the calculated average Budget-Gross ratio of all the movies in that particular country. We illustrate this relationship in a scatter plot that is outputted by the application and shown in Figure 3:



In Figure 3, our title is "GDP vs Average Budget-Gross Ratio Per Country" and our x axis is the calculated average budget-gross ratio and our y axis is the GDP ($ per capita). In our observation, the majority of the countries fall within closer to the 0 budget-gross ratio regardless of the GDP. However, as the GDP There are some outliers where the average budget-gross ratio is extremely high, and this could possibly be due to the limited amount of data points (movies) for this particular country in the dataset.

D. Fourth Data Analysis Task: Profit vs. Score and Top N Directors Based on Average Profit-Score Metric

In our fourth data analysis task, we explore the relationship between profit and score of a movie and then utilize this profit-score metric which is profit multiplied store to find the top n directors. The user can interactively input the value of n in the application constrained by $0 < n <= 10$. For our test input of n = 10, we see the the visualization of our fourth data analysis outputted as one image in Figure 4 below:



The top half of the image displays the relationship between the profit and the score/rating of the movies within our dataset. According to the figure, it becomes apparent that these two variables seemingly have a logarithmic relationship in nature. The top half of the figure also suggests that if a movie does not perform well in terms of profit, it can still have an extremely variable rating, especially since most of the data is

clustered towards the center. As the profit increases, we see that the rating seems to get closer and closer asymptomatically to the max rating possible. In the bottom half of Figure 4, we have the title of the bar graph as “Top Directors Based on Avg Profit-Score Metric”where the x axis is the Directors and the y axis is the Avg Profit-Score metric. The directors listed with our test input n = 10 from left to right on the x axis are the following: Anthony Russo, Chris Buck, Kyle Balda, Joss Whedon, Lee Unkrich, James Cameron, Pierre Coffin, George Lucas, David Yates, and Roger Allers. The top director based on our profit multiplied score metric would be Anthony Russo followed by Chris buck and Kyle Balda.

App.py (main application)

References

[1] Grijalva, D. (2021, July 23). *Movie Industry*. Kaggle. Retrieved May 5, 2022, from https://www.kaggle.com/datasets/danielgrijalvas/movies

[2] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

[3] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95.

[4] Marek. (2018, October 12). *Predicting GDP: World countries*. Kaggle. Retrieved May 5, 2022, from https://www.kaggle.com/code/stieranka/predicting-gdp-world-countries/data

[5] Michael Ernst, How to Write a Technical Paper, https://homes.cs.washington.edu/~mernst/advice/write-technical-paper.html .

[6] Pygal. (n.d.). Retrieved May 5, 2022, from https://www.pygal.org/en/stable/

[7] *Pymongo 4.1.1 documentation¶*. PyMongo 4.1.1 Documentation - PyMongo 4.1.1 documentation. (n.d.). Retrieved May 5, 2022, from https://pymongo.readthedocs.io/en/stable/

[8] R. Elmasri and S. B. Navathe, Algorithms, Fundamentals of Database Systems, Seventh Edition, Pearson, 2017.

[9] *The Application Data Platform*. MongoDB. (n.d.). Retrieved May 5, 2022, from https://www.mongodb.com/

[10] *User guide: Contents*. scikit. (n.d.). Retrieved May 5, 2022, from https://scikit-learn.org/stable/user_guide.html

[11] Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

[12] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

```
import pymongo
import csv
import json
import matplotlib.pyplot as plt
import numpy as np
from pymongo import MongoClient
from sklearn.linear_model import LinearRegression
import pygal
from pygal.style import Style
from countryCodes import countryCodes
from scipy.optimize import curve_fit


def insert(collection):
    with open('movies.csv', mode='r', encoding='utf-8') as csv_file:
        dict_objs = csv.DictReader(csv_file)
        id = 1
        for d in dict_objs:
            d["_id"] = id
            # json_obj = json.dumps(d)
            collection.insert_one(d)
            id += 1


"""
Use MongoDB's aggregration pipeline to compute the most popular/acclaimed movies
based on genre.
Provide a data visualization indicating the disparity of genres among
popular/acclaimed films.
"""


def topPopularGenres(collection, n):
    collection.update_many({"$or": [{"score": ""}, {"votes": ""}]}, {"$set":
{"score": 0, "votes": 0}})
    doubleConversion = {
        "$addFields": {
            "convertedScore": {"$toDouble": "$score"},
            "convertedVotes": {"$toDouble": "$votes"},
        }
```

```
    }

    project = {
        "$project": {"_id": 0, "name": 1, "genre": 1, "metric": {"$multiply":
["$convertedScore", "$convertedVotes"]}},
    }

    sort = {
        "$sort": {"metric": -1}
    }

    limit = {"$limit": n}

    group = {"$group": {"_id": "$genre", "total_amount": {"$sum": 1}}}

    result = collection.aggregate(
        [
            doubleConversion,
            project,
            sort,
            limit,
            group
        ]
    )
    # cursor = collection.find().sort({"score": -1}).limit(15)

    genres = []
    totals = []

    for json in result:
        print(json)
        genres.append(json["_id"])
        totals.append(json["total_amount"])

    # Based on official documentation of matplotlib.
    figure = plt.figure(figsize=(20, 10))

    plt.bar(genres, totals, color='blue', width=0.4)

    plt.xlabel("Genres")
    plt.ylabel("Amount of movies")
```

```
    plt.title("Top Genres in Top 300 Most Popular Movies")
    plt.show()


"""
Utilize MongoDB querying to access various collections from the entire database of
movie information. Top genre produced per country map

"""


def topGenreCountry(collection1):
    genresCountryMappings = {}

    results = []
    for key, val in countryCodes.items():
        match = {"$match": {"country": key}}
        doubleConversion = {
            "$addFields": {
                "convertedScore": {"$toDouble": "$score"},
                "convertedVotes": {"$toDouble": "$votes"},
            }
        }
        project = {
            "$project": {"_id": 0, "country": 1, "genre": 1,
                         "metric": {"$multiply": ["$convertedScore",
"$convertedVotes"]}},
        }
        sort = {"$sort": {"metric": -1}}  # Descending order sort
        limit = {"$limit": 300}
        group = {"$group": {"_id": "$genre", "total_amount": {"$sum": 1}}}
        maximum1 = {"$sort": {"total_amount": -1}}
        maximum2 = {"$limit": 1}
        res = collection1.aggregate([match, doubleConversion, project, sort,
limit, group, maximum1, maximum2])
        for json in res:
            if json["_id"] in genresCountryMappings:
                (genresCountryMappings[json["_id"]]).append(val)
            else:
                genresCountryMappings[json["_id"]] = [val]
```

```
    print(genresCountryMappings)
    wmap = pygal.maps.world.World()
    wmap.title = 'Top Genre Produced per Country'

    for key, val in genresCountryMappings.items():
        wmap.add(key, val)
    wmap.render_to_file("worldmap.svg")
    print("Map created in directory.")


# {'_id': 'Kenya', 'total_amount': 1}
# {'_id': 'Yugoslavia', 'total_amount': 5}
# {'_id': 'Spain', 'total_amount': 47}
# {'_id': 'Ireland', 'total_amount': 43}
# {'_id': 'Sweden', 'total_amount': 25}
# {'_id': 'Japan', 'total_amount': 80}
# {'_id': 'Iran', 'total_amount': 10}
# {'_id': 'Czech Republic', 'total_amount': 8}
# {'_id': 'France', 'total_amount': 279}
# {'_id': 'Greece', 'total_amount': 2}
# {'_id': 'Federal Republic of Yugoslavia', 'total_amount': 2}
# {'_id': 'Poland', 'total_amount': 4}
# {'_id': 'Serbia', 'total_amount': 1}
# {'_id': 'Libya', 'total_amount': 1}
# {'_id': 'West Germany', 'total_amount': 12}
# {'_id': 'Canada', 'total_amount': 190}
# {'_id': 'Italy', 'total_amount': 61}
# {'_id': 'Denmark', 'total_amount': 32}
# {'_id': 'Vietnam', 'total_amount': 2}
# {'_id': 'Iceland', 'total_amount': 2}
# {'_id': 'United Kingdom', 'total_amount': 816}
# {'_id': 'Colombia', 'total_amount': 1}
# {'_id': 'Turkey', 'total_amount': 3}
# {'_id': 'Jamaica', 'total_amount': 1}
# {'_id': 'Taiwan', 'total_amount': 7}
# {'_id': 'Malta', 'total_amount': 1}
# {'_id': 'United Arab Emirates', 'total_amount': 2}
# {'_id': 'Netherlands', 'total_amount': 12}
# {'_id': 'New Zealand', 'total_amount': 25}
# {'_id': 'China', 'total_amount': 40}
# {'_id': 'Romania', 'total_amount': 1}
```

```
# {'_id': 'Indonesia', 'total_amount': 2}
# {'_id': 'Thailand', 'total_amount': 6}
# {'_id': 'Philippines', 'total_amount': 3}
# {'_id': '', 'total_amount': 3}
# {'_id': 'India', 'total_amount': 62}
# {'_id': 'South Africa', 'total_amount': 8}
# {'_id': 'Israel', 'total_amount': 5}
# {'_id': 'Germany', 'total_amount': 117}
# {'_id': 'Portugal', 'total_amount': 2}
# {'_id': 'Australia', 'total_amount': 92}
# {'_id': 'Soviet Union', 'total_amount': 2}


"""
Utilize MongoDB querying to access various collections from the entire database of
movie information.
Visualize the relationship between movie budget-revenue ratio and the country of
the movie's production GDP.
We also provide a model via simple linear regression.
"""


def budgetRevenueRelationship(collection1, collection2):
    # result = collection1.aggregate(
    #     [{"$group": {"_id": "$country", "total_amount": {"$sum": 1}}}]
    # )
    # for i in result:
    #     print(i)

    # string_conversion = {
    #     "$addFields": {
    #         "convertedBudget": {"$toString": "$budget"},
    #         "convertedGross": {"$toString": "gross"}
    #     }
    # }

    collection1.update_many({"$or": [{"score": 0}, {"gross": 0}, {"budget": 0}]},
                            {"$set": {"score": "", "budget": "", "gross": ""}})

    remove_whitespace = {
        "$addFields": {
```

```
            "country": {"$trim": {"input": "$Country"}}
        }
    }

    join = {
        "$lookup": {
            "from": "movies",
            "localField": "country",
            "foreignField": "country",
            "as": "MovieInfo"
        }
    }

    joined_res = collection2.aggregate([
        remove_whitespace,
        join,
    ])
    # Process data to be scattered/analyzed
    revGross = []
    gdps = []
    for x in joined_res:
        if x["MovieInfo"]:
            total = 0
            count = 0
            for y in x["MovieInfo"]:
                y["budget"] = str(y["budget"])
                y["gross"] = str(y["gross"])
                if (len(y["budget"]) != 0 and len(y["gross"]) != 0):
                    temp = float(y["budget"]) / float(y["gross"])
                    y["budgetGrossRatio"] = temp
                    total += temp
                    count += 1
                # print(y)
            if count != 0:
                avg = total / count
                revGross.append(avg)
                gdps.append(int(x["GDP ($ per capita)"]))

    figure = plt.figure(figsize=(15, 10))
    plt.scatter(revGross, gdps, color='green')
```

```
    plt.xlabel("BudgetGrossRatio")
    plt.ylabel("GDP ($ per capita)")
    plt.title("GDP vs Average Budget-Gross Ratio Per Country")

    # Best fit line for this data; However, not applicable for this specific data.
    #model = LinearRegression()
    #revGross = np.array(revGross)
    #gdps = np.array(gdps)
    #model.fit(revGross[:, np.newaxis], gdps)
    # print(model.coef_, model.intercept_)

    #plt.plot(revGross, model.predict(revGross[:, np.newaxis]), color='red')

    plt.show()


"""
Utilize MongoDB querying to access various collections from the entire database of
movie information.
profit (gross - budget) vs score. Top directors based on said metric.
Top 10/20 directors in a bargraph, and the average metric is (profit x score),
directors have to have atleast 2 films,
"""


def profitScoreMetricAnalysis(collection1, n):
    collection1.update_many({"$or": [{"score": ""}, {"gross": ""}, {"budget":
""}]},
                            {"$set": {"score": 0, "budget": 0, "gross": 0}})
    doubleConversion = {
        "$addFields": {
            "convertedGross": {"$toDouble": "$gross"},
            "convertedBudget": {"$toDouble": "$budget"},
            "convertedScore": {"$toDouble": "$score"}
        }
    }

    project1 = {
        "$project": {"_id": 0, "convertedScore": 1, "director": 1,
                     "profit": {"$subtract": ["$convertedGross",
"$convertedBudget"]}}
```

```
    }

    result = collection1.aggregate(
        [
            doubleConversion,
            project1
        ]
    )

    profit = []
    score = []

    for json in result:
        if json["profit"] > 0:
            profit.append(json["profit"])
            score.append(json["convertedScore"])

    def logfunc(x, a, b):
        return a * np.log(x) + b

    # Plot data analyzing profit and score; generate best fit curve for the data
    fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(15,15))
    axes[0].set_xlabel("Profit")
    axes[0].set_ylabel("Score")
    axes[0].set_title("Profit vs Score")
    score = np.array(score)
    popt, pcov = curve_fit(logfunc, profit, score)
    axes[0].scatter(profit, score, color='blue')
    axes[0].plot(profit, logfunc(profit, popt[0], popt[1]), color='red',
label='Curve')
    # model.fit(profit[:,np.newaxis], score)
    # plt.plot(profit, model.predict(profit[:,np.newaxis]), color='red')

    project2 = {
        "$project": {"_id": 0, "convertedScore": 1, "director": 1,
                     "profit": 1, "metric":
                         {
                             "$cond": {"if": {"$ne": ["$profit", 0]},
                                       "then": {"$multiply": ["$profit",
"$convertedScore"]}, "else": 0}
                         },
```

```
                    "count_movie":
                        {
                            "$cond": {"if": {"$eq": ["$profit", 0]},
                                      "then": 0, "else": 1}
                        },
                    }
    }
    group = {
        "$group": {"_id": "$director", "total_movies": {"$sum": "$count_movie"},
"total_metric": {"$sum": "$metric"}}}
    metric_avg = {"$project": {"_id": 1, "metric_avg":
        {
            "$cond": {"if": {"$lt": ["$total_movies", 2]}, "then":
-999999999999999,
                      "else": {"$divide": ["$total_metric", "$total_movies"]}}
        }}}
    sort = {"$sort": {"metric_avg": -1}}
    limit = {"$limit": n}
    result = collection1.aggregate([
        doubleConversion,
        project1,
        project2,
        group,
        metric_avg,
        sort,
        limit
    ])

    directors = []
    metric_avgs = []
    for json in result:
        directors.append(json["_id"])
        metric_avgs.append(json["metric_avg"])

    axes[1].set_xlabel("Directors")
    axes[1].set_ylabel("Avg Profit-Score Metric")
    axes[1].set_title("Top Directors Based on Avg Profit-Score Metric")
    axes[1].bar(directors, metric_avgs, color='orange', width=0.4)
    plt.subplots_adjust(hspace=0.5)
    plt.show()
```

```
if __name__ == "__main__":
    client = pymongo.MongoClient(

"mongodb+srv://george:sujoysikdar@cluster0.frmhm.mongodb.net/admin?retryWrites=tru
e&w=majority")
    db = client['movie_information']
    collection1 = db["movies"]
    collection2 = db["countries"]

    while (1):
        print(
            "Select the following data analysis functionalities for our movies
NOSQL database: \n 1: Compute the most popular/acclaimed movies based on genre
with visualization \n 2: Show the top genre of movies produced per country to
display on map \n 3: Analyze the relationship of GDP vs. the budget-gross ratio of
movies per country \n 4: Analyze the relationship between profit vs. the score of
a movie as well as this this profit-score metric to determine the top n
directors\n q: Quit program")
        inp = input("Enter 1, 2, 3, 4, or q: ")
        if (inp == "1"):
            n = int(input("Set n value for limit (Distribution of genres of
top-popular n movies): \n"))
            bool = n > 0 and n <= 6000
            while (not bool):
                n = int(input("Please input a valid input (n > 0 and n <= 60000):
\n"))
            topPopularGenres(collection1, n)
        elif (inp == "2"):
            topGenreCountry(collection1)
        elif (inp == "3"):
            budgetRevenueRelationship(collection1, collection2)
        elif (inp == "4"):
            n = int(input("Set n value for limit (Top n directors based on
profit-score metric): \n"))
            bool = n > 0 and n <= 10
            while (not bool):
                n = int(input("Please input a valid input (n > 0 and n <= 10):
\n"))
```

```
            profitScoreMetricAnalysis(collection1, n)
        elif (inp == "q"):
            break
        else:
            print("Invalid input! Please type in either 1, 2, 3, 4, or q.")
```

countryCodes.py (second file imported for ease of access to countryCodes)

```
countryList = {
    'AD': 'Andorra',
    'AE': 'United Arab Emirates',
    'AF': 'Afghanistan',
    'AG': 'Antigua & Barbuda',
    'AI': 'Anguilla',
    'AL': 'Albania',
    'AM': 'Armenia',
    'AN': 'Netherlands Antilles',
    'AO': 'Angola',
    'AQ': 'Antarctica',
    'AR': 'Argentina',
    'AS': 'American Samoa',
    'AT': 'Austria',
    'AU': 'Australia',
    'AW': 'Aruba',
    'AZ': 'Azerbaijan',
    'BA': 'Bosnia and Herzegovina',
    'BB': 'Barbados',
    'BD': 'Bangladesh',
    'BE': 'Belgium',
    'BF': 'Burkina Faso',
    'BG': 'Bulgaria',
    'BH': 'Bahrain',
    'BI': 'Burundi',
    'BJ': 'Benin',
    'BM': 'Bermuda',
    'BN': 'Brunei Darussalam',
    'BO': 'Bolivia',
    'BR': 'Brazil',
    'BS': 'Bahama',
    'BT': 'Bhutan',
    'BU': 'Burma (no longer exists)',
```

```
    'BV': 'Bouvet Island',
    'BW': 'Botswana',
    'BY': 'Belarus',
    'BZ': 'Belize',
    'CA': 'Canada',
    'CC': 'Cocos (Keeling) Islands',
    'CF': 'Central African Republic',
    'CG': 'Congo',
    'CH': 'Switzerland',
    'CI': 'Côte D\'ivoire (Ivory Coast)',
    'CK': 'Cook Iislands',
    'CL': 'Chile',
    'CM': 'Cameroon',
    'CN': 'China',
    'CO': 'Colombia',
    'CR': 'Costa Rica',
    'CS': 'Czechoslovakia (no longer exists)',
    'CU': 'Cuba',
    'CV': 'Cape Verde',
    'CX': 'Christmas Island',
    'CY': 'Cyprus',
    'CZ': 'Czech Republic',
    'DD': 'German Democratic Republic (no longer exists)',
    'DE': 'Germany',
    'DJ': 'Djibouti',
    'DK': 'Denmark',
    'DM': 'Dominica',
    'DO': 'Dominican Republic',
    'DZ': 'Algeria',
    'EC': 'Ecuador',
    'EE': 'Estonia',
    'EG': 'Egypt',
    'EH': 'Western Sahara',
    'ER': 'Eritrea',
    'ES': 'Spain',
    'ET': 'Ethiopia',
    'FI': 'Finland',
    'FJ': 'Fiji',
    'FK': 'Falkland Islands (Malvinas)',
    'FM': 'Micronesia',
    'FO': 'Faroe Islands',
```

```
    'FR': 'France',
    'FX': 'France, Metropolitan',
    'GA': 'Gabon',
    'GB': 'United Kingdom',
    'GD': 'Grenada',
    'GE': 'Georgia',
    'GF': 'French Guiana',
    'GH': 'Ghana',
    'GI': 'Gibraltar',
    'GL': 'Greenland',
    'GM': 'Gambia',
    'GN': 'Guinea',
    'GP': 'Guadeloupe',
    'GQ': 'Equatorial Guinea',
    'GR': 'Greece',
    'GS': 'South Georgia and the South Sandwich Islands',
    'GT': 'Guatemala',
    'GU': 'Guam',
    'GW': 'Guinea-Bissau',
    'GY': 'Guyana',
    'HK': 'Hong Kong',
    'HM': 'Heard & McDonald Islands',
    'HN': 'Honduras',
    'HR': 'Croatia',
    'HT': 'Haiti',
    'HU': 'Hungary',
    'ID': 'Indonesia',
    'IE': 'Ireland',
    'IL': 'Israel',
    'IN': 'India',
    'IO': 'British Indian Ocean Territory',
    'IQ': 'Iraq',
    'IR': 'Islamic Republic of Iran',
    'IS': 'Iceland',
    'IT': 'Italy',
    'JM': 'Jamaica',
    'JO': 'Jordan',
    'JP': 'Japan',
    'KE': 'Kenya',
    'KG': 'Kyrgyzstan',
    'KH': 'Cambodia',
```

```
    'KI': 'Kiribati',
    'KM': 'Comoros',
    'KN': 'St. Kitts and Nevis',
    'KP': 'Korea, Democratic People\'s Republic of',
    'KR': 'South Korea',
    'KW': 'Kuwait',
    'KY': 'Cayman Islands',
    'KZ': 'Kazakhstan',
    'LA': 'Lao People\'s Democratic Republic',
    'LB': 'Lebanon',
    'LC': 'Saint Lucia',
    'LI': 'Liechtenstein',
    'LK': 'Sri Lanka',
    'LR': 'Liberia',
    'LS': 'Lesotho',
    'LT': 'Lithuania',
    'LU': 'Luxembourg',
    'LV': 'Latvia',
    'LY': 'Libya',
    'MA': 'Morocco',
    'MC': 'Monaco',
    'MD': 'Moldova, Republic of',
    'MG': 'Madagascar',
    'MH': 'Marshall Islands',
    'ML': 'Mali',
    'MN': 'Mongolia',
    'MM': 'Myanmar',
    'MO': 'Macau',
    'MP': 'Northern Mariana Islands',
    'MQ': 'Martinique',
    'MR': 'Mauritania',
    'MS': 'Monserrat',
    'MT': 'Malta',
    'MU': 'Mauritius',
    'MV': 'Maldives',
    'MW': 'Malawi',
    'MX': 'Mexico',
    'MY': 'Malaysia',
    'MZ': 'Mozambique',
    'NA': 'Namibia',
    'NC': 'New Caledonia',
```

```
    'NE': 'Niger',
    'NF': 'Norfolk Island',
    'NG': 'Nigeria',
    'NI': 'Nicaragua',
    'NL': 'Netherlands',
    'NO': 'Norway',
    'NP': 'Nepal',
    'NR': 'Nauru',
    'NT': 'Neutral Zone (no longer exists)',
    'NU': 'Niue',
    'NZ': 'New Zealand',
    'OM': 'Oman',
    'PA': 'Panama',
    'PE': 'Peru',
    'PF': 'French Polynesia',
    'PG': 'Papua New Guinea',
    'PH': 'Philippines',
    'PK': 'Pakistan',
    'PL': 'Poland',
    'PM': 'St. Pierre & Miquelon',
    'PN': 'Pitcairn',
    'PR': 'Puerto Rico',
    'PT': 'Portugal',
    'PW': 'Palau',
    'PY': 'Paraguay',
    'QA': 'Qatar',
    'RE': 'Réunion',
    'RO': 'Romania',
    'RU': 'Russia',
    'RW': 'Rwanda',
    'SA': 'Saudi Arabia',
    'SB': 'Solomon Islands',
    'SC': 'Seychelles',
    'SD': 'Sudan',
    'SE': 'Sweden',
    'SG': 'Singapore',
    'SH': 'St. Helena',
    'SI': 'Slovenia',
    'SJ': 'Svalbard & Jan Mayen Islands',
    'SK': 'Slovakia',
    'SL': 'Sierra Leone',
```

```
    'SM': 'San Marino',
    'SN': 'Senegal',
    'SO': 'Somalia',
    'SR': 'Suriname',
    'ST': 'Sao Tome & Principe',
    'SU': 'Union of Soviet Socialist Republics (no longer exists)',
    'SV': 'El Salvador',
    'SY': 'Syrian Arab Republic',
    'SZ': 'Swaziland',
    'TC': 'Turks & Caicos Islands',
    'TD': 'Chad',
    'TF': 'French Southern Territories',
    'TG': 'Togo',
    'TH': 'Thailand',
    'TJ': 'Tajikistan',
    'TK': 'Tokelau',
    'TM': 'Turkmenistan',
    'TN': 'Tunisia',
    'TO': 'Tonga',
    'TP': 'East Timor',
    'TR': 'Turkey',
    'TT': 'Trinidad & Tobago',
    'TV': 'Tuvalu',
    'TW': 'Taiwan',
    'TZ': 'Tanzania, United Republic of',
    'UA': 'Ukraine',
    'UG': 'Uganda',
    'UM': 'United States Minor Outlying Islands',
    'US': 'United States',
    'UY': 'Uruguay',
    'UZ': 'Uzbekistan',
    'VA': 'Vatican City State (Holy See)',
    'VC': 'St. Vincent & the Grenadines',
    'VE': 'Venezuela',
    'VG': 'British Virgin Islands',
    'VI': 'United States Virgin Islands',
    'VN': 'Vietnam',
    'VU': 'Vanuatu',
    'WF': 'Wallis & Futuna Islands',
    'WS': 'Samoa',
    'YD': 'Democratic Yemen (no longer exists)',
```

```
    'YE': 'Yemen',
    'YT': 'Mayotte',
    'YU': 'Yugoslavia',
    'ZA': 'South Africa',
    'ZM': 'Zambia',
    'ZR': 'Zaire',
    'ZW': 'Zimbabwe',
    'ZZ': 'Unknown or unspecified country',
}
countryCodes = {val: key for key, val in countryList.items()}

for key, val in countryCodes.items():
    countryCodes[key] = val.lower()
```