VISVESVARAYA TECHNOLOGICAL UNIVERSITY,
BELAGAVI 590018

Report on

"Movie Data Analysis"

By

Lavanya G (1BM17CS045) K Rilika Uthappa(1BM17CS038)
K Lakshmi Sai Priya (1BM17CS036)

Under the Guidance of

Prof. Pallavi G B
Assistant Professor,
Department of CSE
BMS College of Engineering

Work carried out at

Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2020-2021

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the BIG DATA and ANALYTICS mini Project titled "**Movie Data Analysis"** has been carried out by K Lakshmi Sai Priya(1BM17CS036), Lavanya G(1BM17CS045), K Rilika Uthappa(1BM17CS038) during the academic year 2020-2021.

Signature of the guide
Prof. Pallavi G B,
Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, K Lakshmi Sai Priya (1BM17CS036), Lavanya G(1BM17CS045), K Rilika Uthappa (1BM17CS038), students of 7th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this assignment work entitled "**Movie Data Analysis**" has been carried out by us under the guidance of **Prof. Pallavi G B, Assistant Professor,** Department of CSE, BMS College of Engineering, Bangalore during the academic semester Aug-Dec 2020. We also declare that to the best of our knowledge and belief, the assignment reported here is not from part of any other report by any other students.

Signature of the Candidates

K Lakshmi Sai Priys(1BM17CS036)

Lavanya G(1BM17CS045)

K Rilika Uthappa(1BM17CS038)

Objectives:

- To perform CRUD (Create, Read, Update and Delete) operations on Movie Data Analysis dataset via a user interface developed using Python programming language and MongoDB.
- To analyze the data using analytical queries and gain meaningful insights about the movie dataset using Python .

Introduction:

Problem Statement

The TMDB wants to know about what kind of movies are being appreciated by the audience for their reference in the future. They expect to know the genre, status of the movies, revenue, budget and popularity of all the successful movies till date to predict the reviews of the movie.

The TMDB wants to know:

- Movies belonging to which genre are being most appreciated by the audience and are gaining more popularity.

- Within what budget should the movie should be produced.

Based on various movie data analysis, the TMDB has gathered a large dataset to predict the reviews of the movies.

Business Goal

TMDB will be required to predict accurate reviews of the movies based on the graphs generated using Python.

Design Modules:

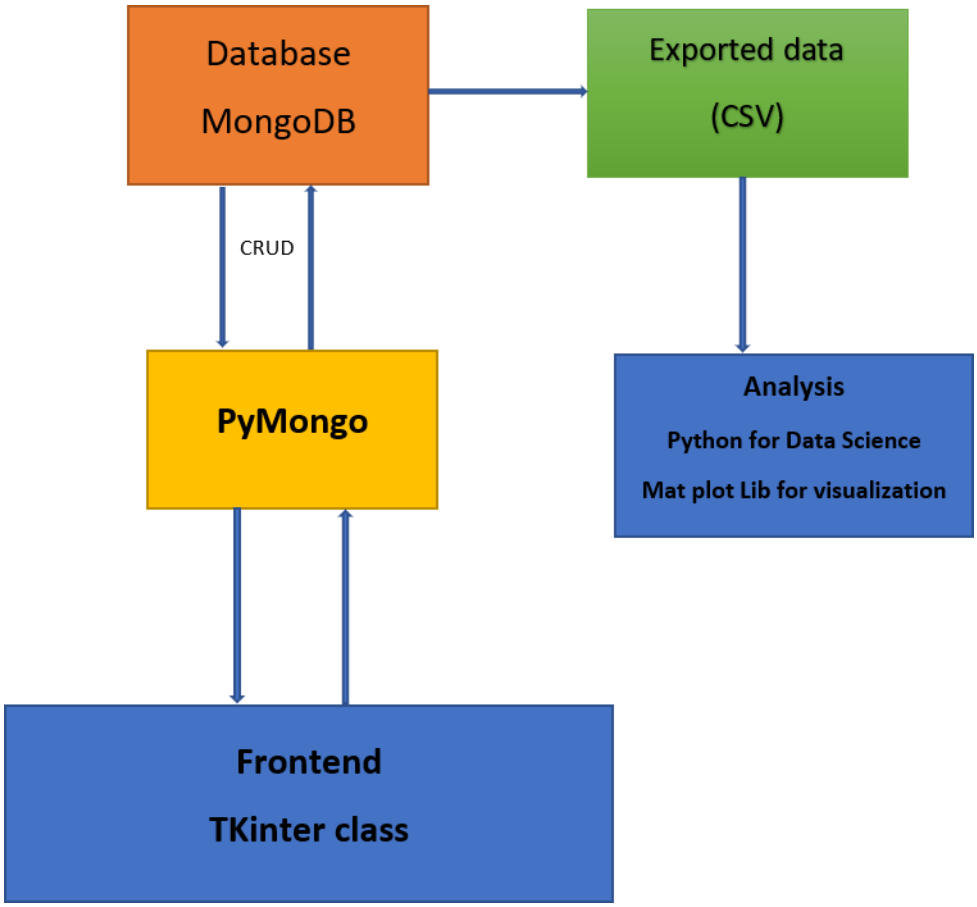


Figure 1 : Modules

Detailed Description of Modules:

- **Frontend – TKinter class:** Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

- **Database – MongoDB:** The data is exported from the csv file and imported into the MongoDB Database. The changes in the database including the CRUD operations are performed by the PyMongo using the MongoDB commands. The database is also queried for reviews and genre names.

- **PyMongo-** The PyMongo distribution contains tools for interacting with MongoDB database from Python. The bson package is an implementation of the BSON format for Python. The pymongo package is a native Python driver for MongoDB. The gridfs package is a gridfs implementation on top of pymongo.

- ## Code for INSERT:

  Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.
  We use the db.collection.insertOne() method to insert documents into a collection:

```
def Insert():

    def Ins():

        budget=Budget.get()

        genres=Genres.get()

        id=Id.get()

        original_language=Original_language.get()

        release_year=Release_year.get()

        popularity=Popularity.get()

        runtime=Runtime.get()

        title=Title.get()



        dic={ "budget":budget, "genres" : genres, "id" : id, "original_language" : original_language,
"release_year" : release_year, "popularity" : popularity, "Runtime" : runtime, "title": title }

        x = mycollection.insert_one(dic)
```

- ## Code for UPDATE:

  Update operations modify existing documents in a collection.

  We use the db.collection.updateOne() method to update documents of a collection:

```
def Update():

    def Upd():
```

```
#budget=Budget.get()
 #genres=Genres.get()
 revenue=Revenue.get()
 #original_language=Original_language.get()
 popularity=Popularity.get()
 title=Title.get()

 dic={"$set":{"revenue" :revenue,"popularity":popularity}}
 x = mycollection.update_many({"title":title},dic)
```

- **Code for DELETE:**

Delete operations remove documents from a collection.
We use the db.collection.deleteMany() method to delete documents of a collection.

```
def Delete():
    root3=tk.Tk()
    root3.geometry('1000x600')
    root3.configure(background='#bdbdbd')
    root3.title('Delete')
    def delete():
        title=Title.get()
        id=Id.get()
```

```
 g=mycollection.delete_many({"title":title,"id":id})

q=tk.StringVar(root3)

q.set('')

u=tk.IntVar(root3)

f1=tk.Frame(root3,bg='#bdbdbd')

at=tk.Label(f1,text="Title",font='Helvetica 18 bold',width=7,height=1)

at.pack(padx=5,pady=10,side='left')


Title=tk.Entry(f1,textvariable=q)

Title.pack(padx=5,pady=10,side='left')

at1=tk.Label(f1,text="ID",font='Helvetica 18 bold',width=7,height=1)

at1.pack(padx=5,pady=10,side='left')

Id=tk.Entry(f1,textvariable=u)

Id.pack(padx=5,pady=10,side='left')

btu=tk.Button(f1,text="Delete",fg='black',font="Helvetica bold",command=delete,width=8)

btu.pack(padx=5,pady=10,side='left')

f1.pack()

root3.mainloop()
```


- **Code for READ:**

Read operations retrieve documents from a collection; i.e. query a collection for documents.
We use the db.collection.find()method to read documents from a collection:

```
def Search():
```

```
def search():
title=Title.get()
d=mycollection.find({"title":title}).limit(1000)
eula.delete(0, 'end')#to clear listbox
for i in d:
    eula.insert(END,str(i))#to insert text into listbox
     q=tk.StringVar(root2)
     q.set('')
```

- **Code for Displaying Top Ten Movies**:

  Research Question 1 : Which are the most popular movies?

```
def hdrc():
    d=mycollection.find().sort([("popularity",pymongo.DESCENDING)]).limit(10)
  for i in d:
    eula.insert(END,str(i))#to insert text into listbox
```

- **Code for Displaying Top Movies under each Genre:**

  Research Question 2: Which are the most popular movies in each genre?

```
def hdr():
   def search():
    genres=Genres.get()

d=mycollection.find({"genres":genres}).sort([("popularity",pymongo.DESCENDING)]).l
imit(10)

    eula.delete(0, 'end')#to clear listbox
```

```
for i in d:

    eula.insert(END,str(i))#to insert text into listbox
```

Analysis Component (Python): This is used to create analytical queries to understand the dataset and gain meaningful insights from it. Python is a cross-functional, maximally interpreted language that has lots of advantages to offer. The object-oriented programming language is commonly used to streamline large complex data sets. Over and above, having a dynamic semantics plus unmeasured capacities of RAD(rapid application development), Python is heavily utilized to script as well. There is one more way to apply Python – as a coupling language.

A snapshot of the code is provided below:

- **Code for GRAPHS**:

Analysis Question 3: How is the Movie Production trend over the years?

```
def plot():

    movies_per_year= db['release_year'].value_counts().sort_index()

    plt.plot(movies_per_year)

    plt.title('Movie production trend over the years')

    plt.xlabel('Year')

    plt.ylabel('Number of movies released')

    plt.show()
```

Analysis Question 4: Which are the highest grossing movies?

```
def plot2():

    sorted_revenue = db['revenue'].sort_values(ascending=False)[:20]



    high_grossers=pd.DataFrame()

    titles=[]

    revenues=[]

    for i in sorted_revenue.index:
```

```
        titles.append(db.loc[i,'original_title'])
    revenues.append(sorted_revenue.loc[i])
    high_grossers['Titles']=titles
    high_grossers['Revenues']=revenues
    high_grossers.set_index('Titles',inplace=True)
    high_grossers.plot(kind ='bar',figsize=(5,5))
    plt.title('Top 20 highest grossing movies (1960 - 2015) ')
    plt.ylabel('Revenue in billions ($)')
    plt.show()
```

Analysis Question 5 : Which are the most expensive movies?

```
def plot3():
    sorted_budget = db['budget'].sort_values(ascending=False)[:20]
    high_budget=pd.DataFrame()
    titles_exp=[]
    budgets=[]
    for i in sorted_budget.index:
        titles_exp.append(db.loc[i,'original_title'])
        budgets.append(sorted_budget.loc[i])
    high_budget['Titles']=titles_exp
    high_budget['Budgets']=budgets
    high_budget.set_index('Titles',inplace=True)
    high_budget.plot(kind  ='bar',figsize=(8,8))
    plt.title('Top 20 most expensive movies (1960 - 2015) ')
    plt.ylabel('Budget in 100\'s of million ($)')
```

```
    plt.show()
```

Analysis Question 6: Which are the highest grossing movies?

```
def plot4():
    db['Profit']=db['revenue']-db['budget']
    db.plot(x='vote_average',y='Profit',kind='scatter',figsize=(8,8))
    plt.ylabel('Profit in billion ($)')
    plt.xlabel('Rating')
    plt.title('Rating vs Profit')
    plt.show()
```

Analysis Question 7: How many movies are produced in each genre?

```
def plot5():

    def count_genre(x):
        #concatenate all the rows of the genrs.
        data_plot = db[x].str.cat(sep = '|')
        data = pd.Series(data_plot.split('|'))
        #conts each of the genre and return.
        info = data.value_counts(ascending=False)
        return info

    #call the function for counting the movies of each genre.
    total_genre_movies = count_genre('genres')
#plot a 'barh' plot using plot function for 'genre vs number of movies'.
    total_genre_movies.plot(kind= 'barh',figsize = (8,8),fontsize=6,colormap='tab20c')
```

```
#setup the title and the labels of the plot.
    plt.title("Genre With Highest Release",fontsize=15)
    plt.xlabel('Number Of Movies',fontsize=13)
    plt.ylabel("Genres",fontsize= 13)
    plt.show()


def plot6():


    def count_genre(x):
        #concatenate all the rows of the genrs.
        data_plot = db[x].str.cat(sep = '|')
        data = pd.Series(data_plot.split('|'))
        #conts each of the genre and return.
        info = data.value_counts(ascending=False)
        return info
    i = 0
    genre_count = []
    total_genre_movies = count_genre('genres')
    for genre in total_genre_movies.index:
        genre_count.append([genre, total_genre_movies[i]])
        i = i+1


    plt.rc('font', weight='bold')
```

```
    f, ax = plt.subplots(figsize=(5, 5))
    genre_count.sort(key = lambda x:x[1], reverse = True)
    labels, sizes = zip(*genre_count)
    labels_selected = [n if v > sum(sizes) * 0.01 else '' for n, v in genre_count]
    ax.pie(sizes, labels=labels_selected,
      autopct = lambda x:'{:2.0f}%'.format(x) if x > 1 else '',
      shadow=False, startangle=0)
    ax.axis('equal')
    plt.tight_layout()
    plt.show()
```

Analysis Question 8 : How many movies are produced by each production company?

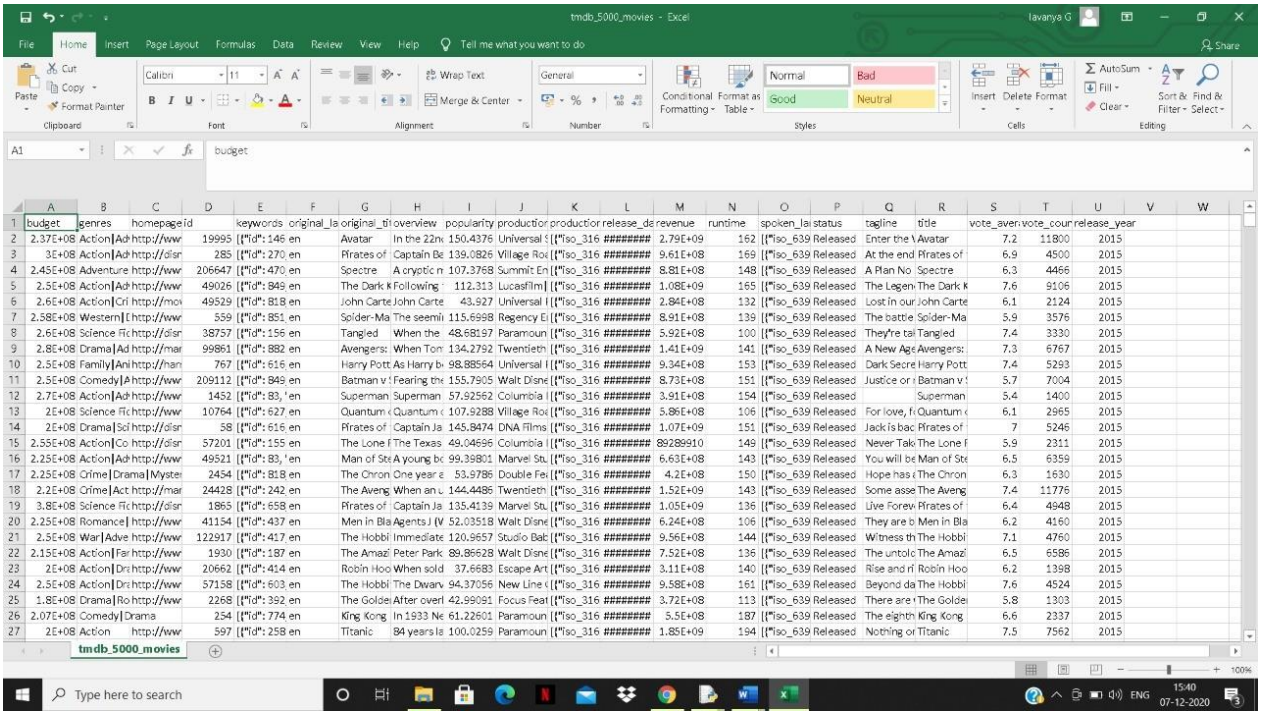
```
def plot7():
    def count_genre(x):
        #concatenate all the rows of the genrs.
        data_plot = db[x].str.cat(sep = '|')
        data = pd.Series(data_plot.split('|'))
        #conts each of the genre and return.
        info = data.value_counts(ascending=False)
        return info
    production_companies = count_genre('production_companies')

#plot he barh plot.
    production_companies.iloc[:20].plot(kind='barh',figsize=(16,8),fontsize=13)
```

```
plt.title("Production Companies Vs Number Of Movies",fontsize=15)

plt.xlabel('Number Of Movies',fontsize=14)


plt.show()
```

- **CSV File (Dataset):** The dataset consists of 21 variables which can be used to predict/analyse the review of the movie. Those are:
    - Budget: Amount spent in producing a movie.
    - Genres: Genre to which the movie belongs to.
    - Homepage: URL which leads to the information about the movie.
    - ID: ID of the movie.
    - Keywords: Keywords related to the movies.
    - Original Language: The language in which the movie was originally produced.
    - Original Title: Original name of the movie.
    - Overview: Summary of the movie.
    - Popularity: How popular the movie was
    - Production Companies: The company which produced the movies.
    - Production Countries: Countries in which the movie was produced.
    - Release Date: The date on which the movie was released.
    - Revenue: The amount which movie earned.
    - RunTime: Number of days for which the movie was running in the theatres.
    - Spoken Languages: Languages which were spoken in the movie.
    - Status: Status of the movie(eg: released, not released).
    - Tag Line: Slogan of the movie.
    - Title: Name of the movie.
    - Vote Average: Average rating of the movie.
    - Vote Count: Voting rate of the movie.

- Release Year: The year in which the movie was released.

A snapshot of the csv file is provided below:



budget	genres	homepage	id	keywords	original_la	original_ti	overview	popularity	production	production	release_da	revenue	runtime	spoken_la	status	tagline	title	vote_aver	vote_cour	release_year
2.37E+08	Action|Ad	http://ww	19995	[{"id": 146	en	Avatar	In the 22n	150.4376	Universal S	[{"iso_316	########	2.79E+09	162	[{"iso_639	Released	Enter the \	Avatar	7.2	11800	2015
3E+08	Action|Ad	http://disn	285	[{"id": 270	en	Pirates of	Captain Ba	139.0826	Village Roa	[{"iso_316	########	9.61E+08	169	[{"iso_639	Released	At the end	Pirates of	6.9	4500	2015
2.45E+08	Adventure	http://ww	206647	[{"id": 470	en	Spectre	A cryptic n	107.3768	Summit En	[{"iso_316	########	8.81E+08	148	[{"iso_639	Released	A Plan No	Spectre	6.3	4466	2015
2.5E+08	Action|Ad	http://ww	49026	[{"id": 849	en	The Dark K	Following	112.313	Lucasfilm|	[{"iso_316	########	1.08E+09	165	[{"iso_639	Released	The Legen	The Dark K	7.6	9106	2015
2.6E+08	Action|Cri	http://mov	49529	[{"id": 818	en	John Carte	John Carte	43.927	Universal I	[{"iso_316	########	2.84E+08	132	[{"iso_639	Released	Lost in our	John Carte	6.1	2124	2015
2.58E+08	Western|	http://ww	559	[{"id": 851	en	Spider-Ma	The seemi	115.6998	Regency Ei	[{"iso_316	########	8.91E+08	139	[{"iso_639	Released	The battle	Spider-Ma	5.9	3576	2015
2.6E+08	Science Fic	http://disn	38757	[{"id": 156	en	Tangled	When the	48.68197	Paramoun	[{"iso_316	########	5.92E+08	100	[{"iso_639	Released	They're tal	Tangled	7.4	3330	2015
2.8E+08	Drama|Ad	http://mar	99861	[{"id": 882	en	Avengers:	When Ton	134.2792	Twentieth	[{"iso_316	########	1.41E+09	141	[{"iso_639	Released	A New Age	Avengers:	7.3	6767	2015
2.5E+08	Family|Ani	http://har	767	[{"id": 616	en	Harry Pott	As Harry b	98.88564	Universal I	[{"iso_316	########	9.34E+08	153	[{"iso_639	Released	Dark Secre	Harry Pott	7.4	5293	2015
2.5E+08	Comedy|A	http://ww	209112	[{"id": 849	en	Batman v	Fearing the	155.7905	Walt Disne	[{"iso_316	########	8.73E+08	151	[{"iso_639	Released	Justice or	Batman v	5.7	7004	2015
2.7E+08	Action|Ad	http://ww	1452	[{"id": 83,	en	Superman	Superman	57.92562	Columbia I	[{"iso_316	########	3.91E+08	154	[{"iso_639	Released		Superman	5.4	1400	2015
2E+08	Science Fic	http://ww	10764	[{"id": 627	en	Quantum (	Quantum (	107.9288	Village Roa	[{"iso_316	########	5.86E+08	106	[{"iso_639	Released	For love, I	Quantum (	6.1	2965	2015
2E+08	Drama|Sci	http://disn	58	[{"id": 616	en	Pirates of	Captain Ja	145.8474	DNA Films	[{"iso_316	########	1.07E+09	151	[{"iso_639	Released	Jack is bac	Pirates of	7	5246	2015
2.55E+08	Action|Co	http://disn	57201	[{"id": 155	en	The Lone I	The Texas	49.04696	Columbia I	[{"iso_316	########	89289910	149	[{"iso_639	Released	Never Tak	The Lone I	5.9	2311	2015
2.25E+08	Action|Ad	http://ww	49521	[{"id": 83,	en	Man of Ste	A young bo	99.39801	Marvel Stu	[{"iso_316	########	6.63E+08	143	[{"iso_639	Released	You will be	Man of Ste	6.5	6359	2015
2.25E+08	Crime|Drama|Myste		2454	[{"id": 818	en	The Chron	One year a	53.9786	Double Fe	[{"iso_316	########	4.2E+08	150	[{"iso_639	Released	Hope has a	The Chron	6.3	1630	2015
2.2E+08	Crime|Act	http://mar	24428	[{"id": 242	en	The Aveng	When an u	144.4486	Twentieth	[{"iso_316	########	1.52E+09	143	[{"iso_639	Released	Some asse	The Aveng	7.4	11776	2015
3.8E+08	Science Fic	http://disn	1865	[{"id": 658	en	Pirates of	Captain Ja	135.4139	Marvel Stu	[{"iso_316	########	1.05E+09	136	[{"iso_639	Released	Live Forev	Pirates of	6.4	4948	2015
2.25E+08	Romance|	http://ww	41154	[{"id": 437	en	Men in Bla	Agents J (M	52.03518	Walt Disne	[{"iso_316	########	6.24E+08	106	[{"iso_639	Released	They are b	Men in Bla	6.2	4160	2015
2.5E+08	War|Adve	http://ww	122917	[{"id": 417	en	The Hobbi	Immediate	120.9657	Studio Bab	[{"iso_316	########	9.56E+08	144	[{"iso_639	Released	Witness th	The Hobbi	7.1	4760	2015
2.15E+08	Action|Far	http://ww	1930	[{"id": 187	en	The Amazi	Peter Park	89.86628	Walt Disne	[{"iso_316	########	7.52E+08	136	[{"iso_639	Released	The untold	The Amazi	6.5	6586	2015
2E+08	Action|Dra	http://ww	20662	[{"id": 414	en	Robin Hoo	When sold	37.6683	Escape Art|	[{"iso_316	########	3.11E+08	140	[{"iso_639	Released	Rise and ri	Robin Hoo	6.2	1398	2015
2.5E+08	Action|Dra	http://ww	57158	[{"id": 603	en	The Hobbi	The Dwarv	94.37056	New Line (	[{"iso_316	########	9.58E+08	161	[{"iso_639	Released	Beyond da	The Hobbi	7.6	4524	2015
1.8E+08	Drama|Ro	http://ww	2268	[{"id": 392	en	The Golde	After overl	42.99091	Focus Feat	[{"iso_316	########	3.72E+08	113	[{"iso_639	Released	There are	The Golde	5.8	1303	2015
2.07E+08	Comedy|Drama		254	[{"id": 774	en	King Kong	In 1933 Ne	61.22601	Paramoun	[{"iso_316	########	5.5E+08	187	[{"iso_639	Released	The eighth	King Kong	6.6	2337	2015
2E+08	Action	http://ww	597	[{"id": 258	en	Titanic	84 years la	100.0259	Paramoun	[{"iso_316	########	1.85E+09	194	[{"iso_639	Released	Nothing or	Titanic	7.5	7562	2015

Screenshots:

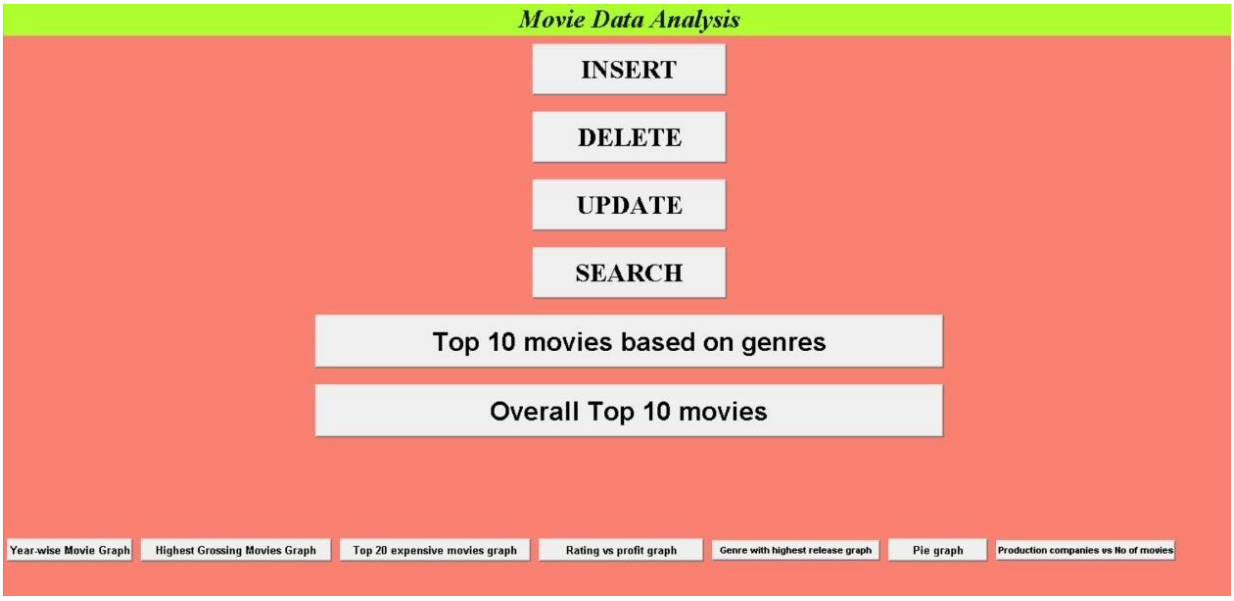


Figure 2 - Home page

Figure 3- To insert a new movie into the database

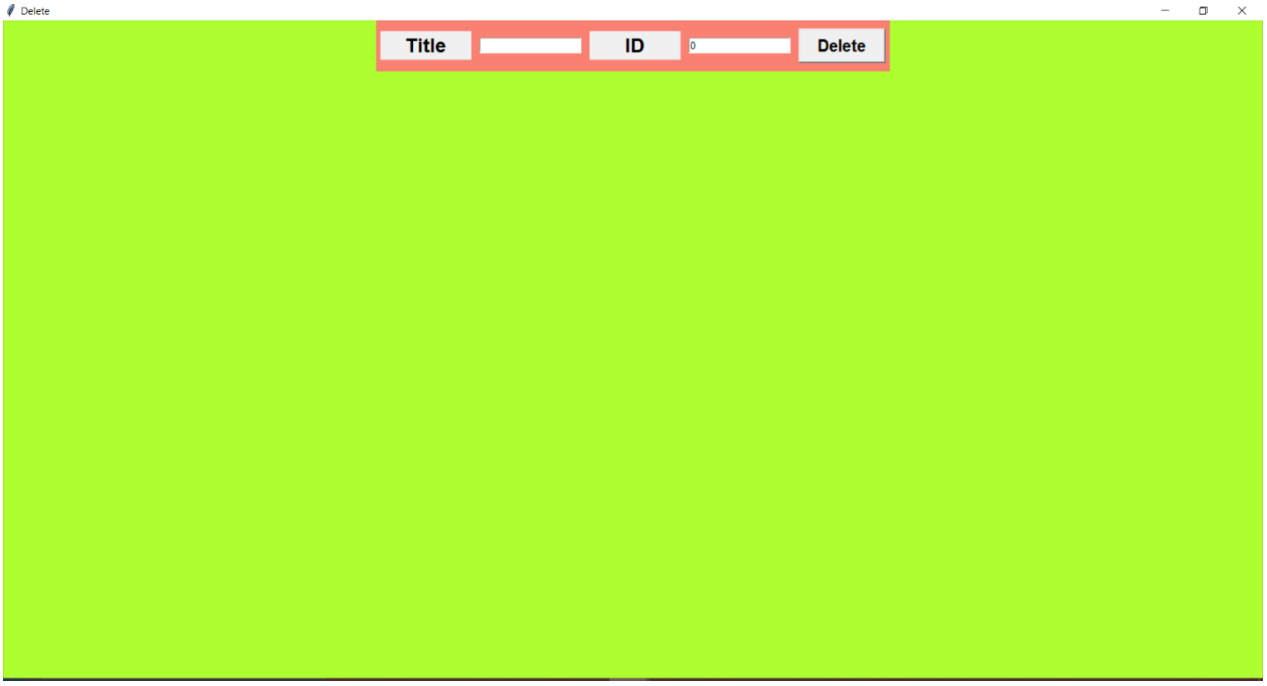


Figure 4 - Delete option provided for every movie

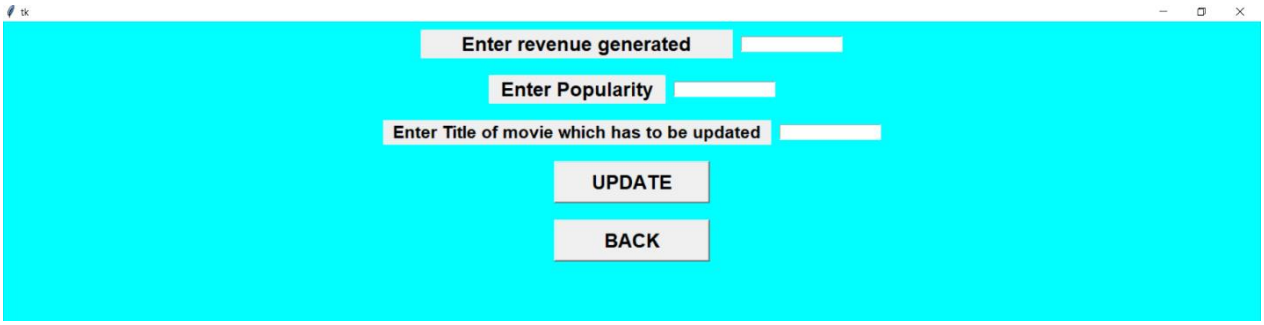


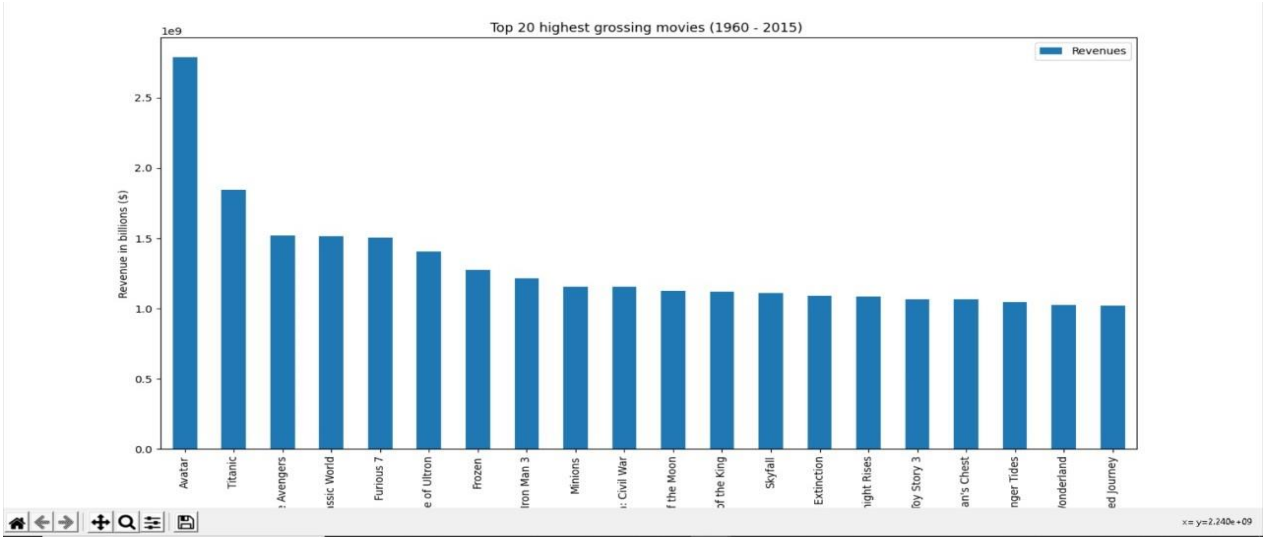
Figure 5 - Updating movie details

Figure 6 - Searching for a movie

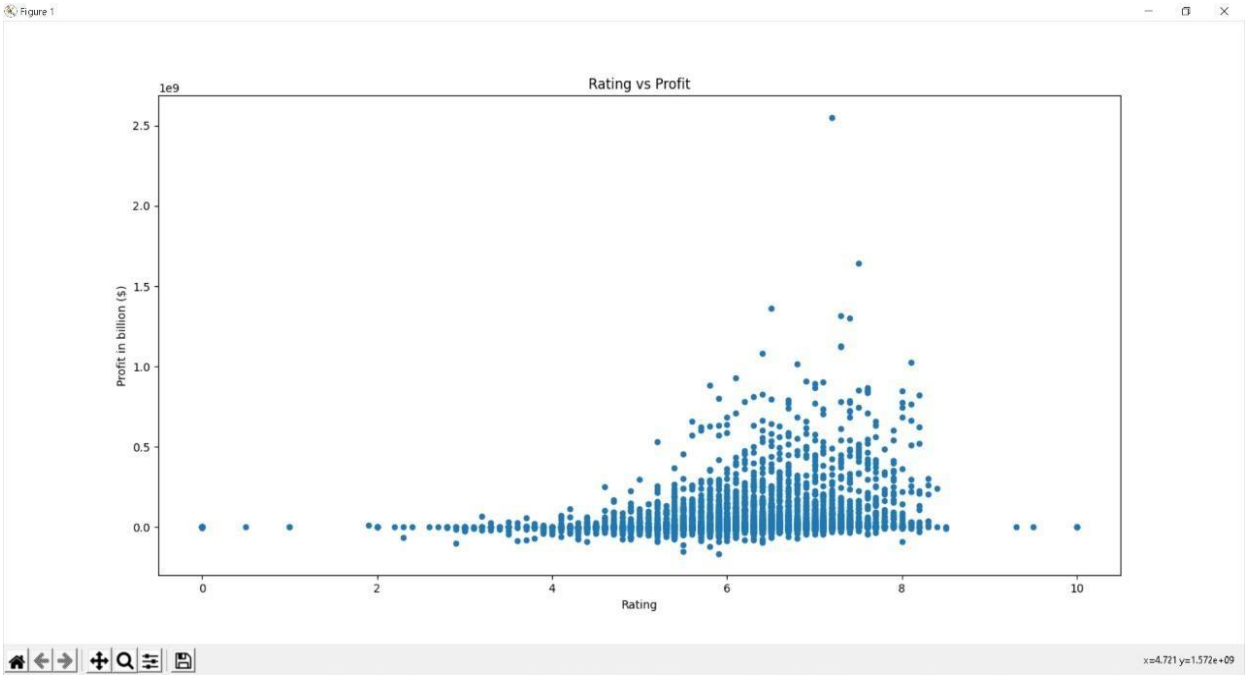
Figure 7 – Searching for Top 10 movies according to the Genre.

Figure 8 – Searching for overall Top 10 movies.

Screenshots of Graphs:

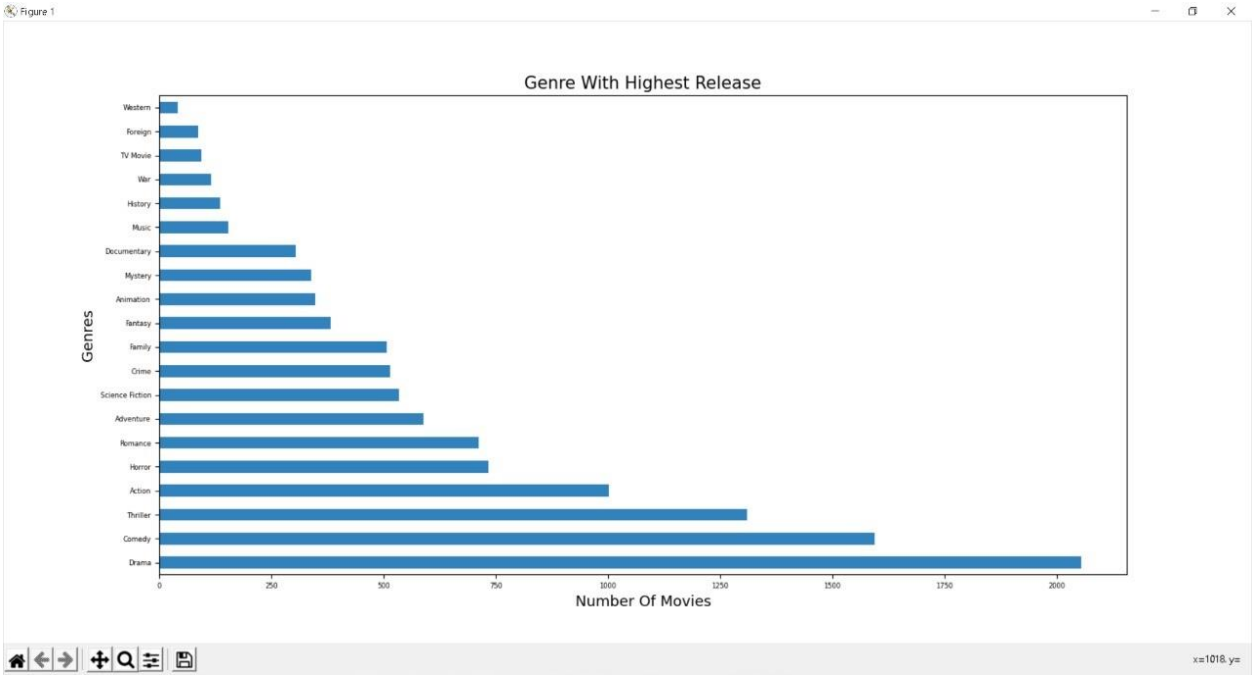


The above bar graph is a plot of the 'Top 20 highest grossing movies. 'Avatar' is the highest grossing movie with a revenue near to 3 billion.



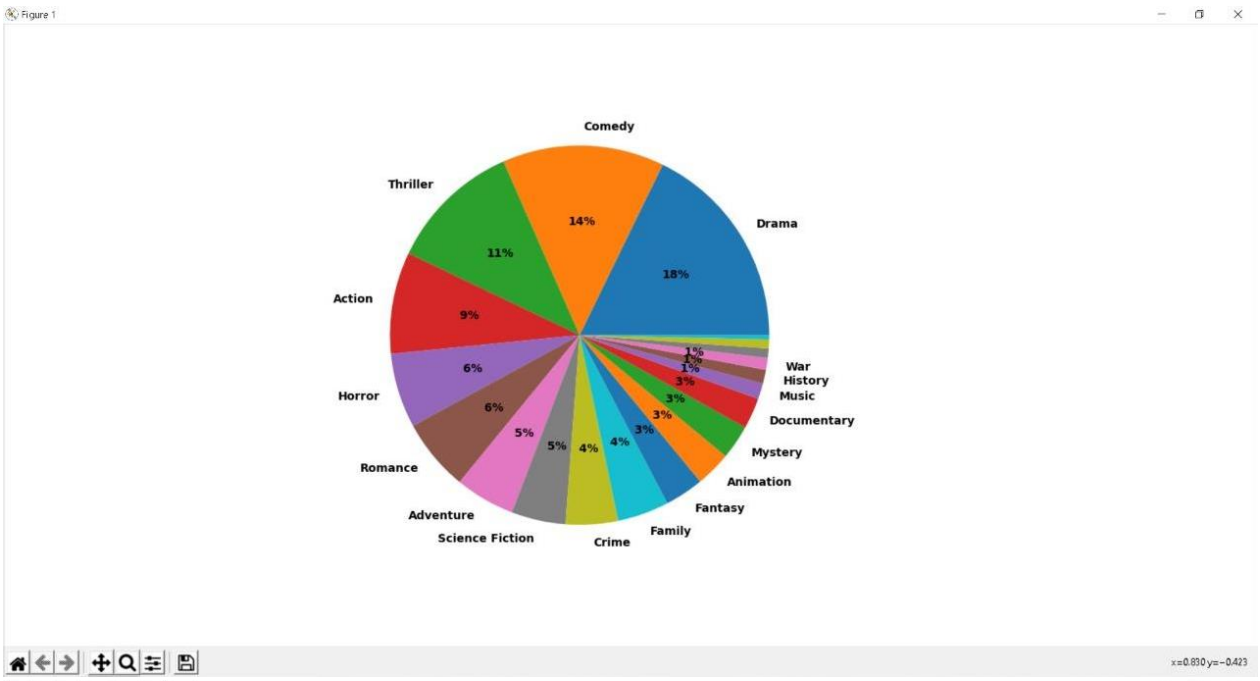
The above plot is a scatter plot of the ‘Rating vs ‘profit.

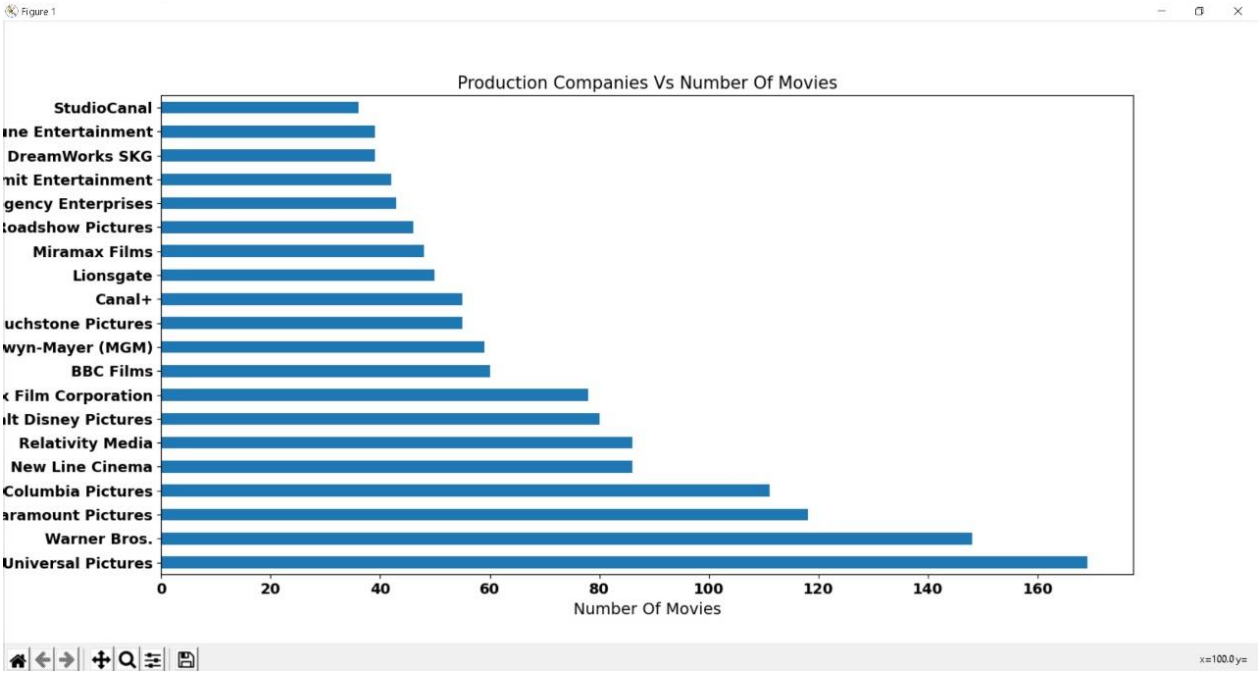
The highest profits are earned by the movies having an average rating between 6-8 with the maximum profit being 2.5 billion.



The above bar graph is a plot of the Genre with Highest Release

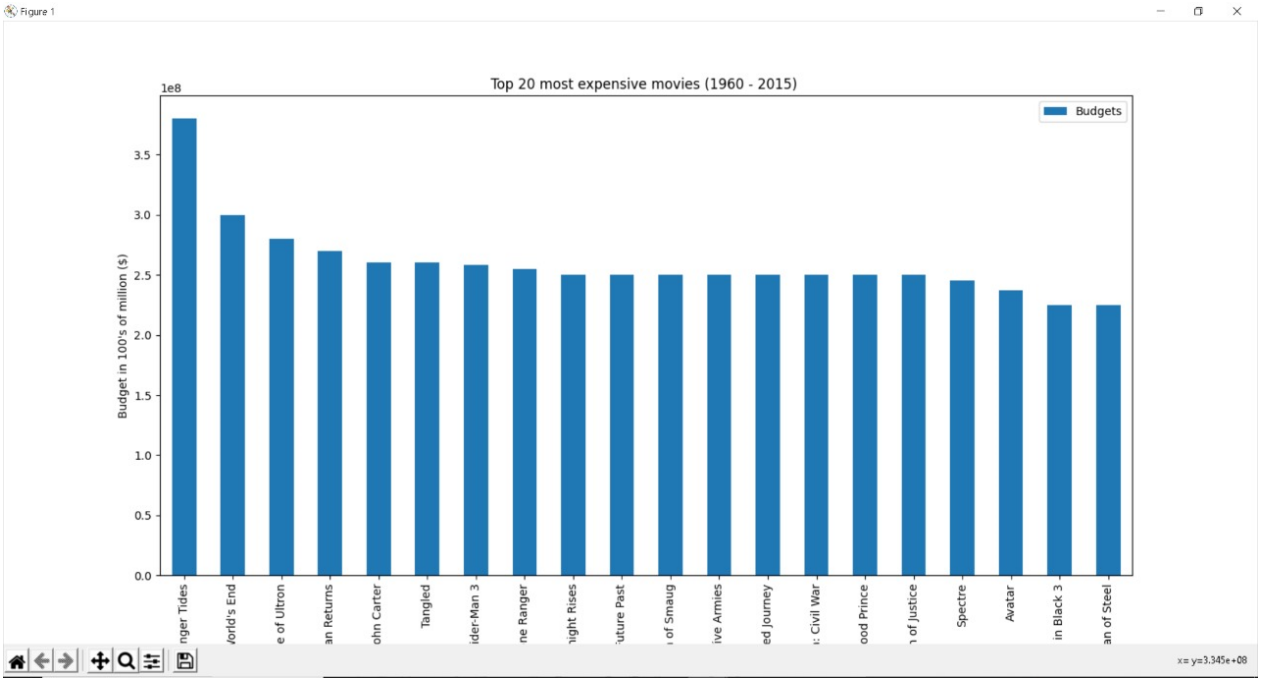
The maximum number of movies that are released are of the genre ‘Dramas’. There are 2000 movies from the ‘Drama’ genre.





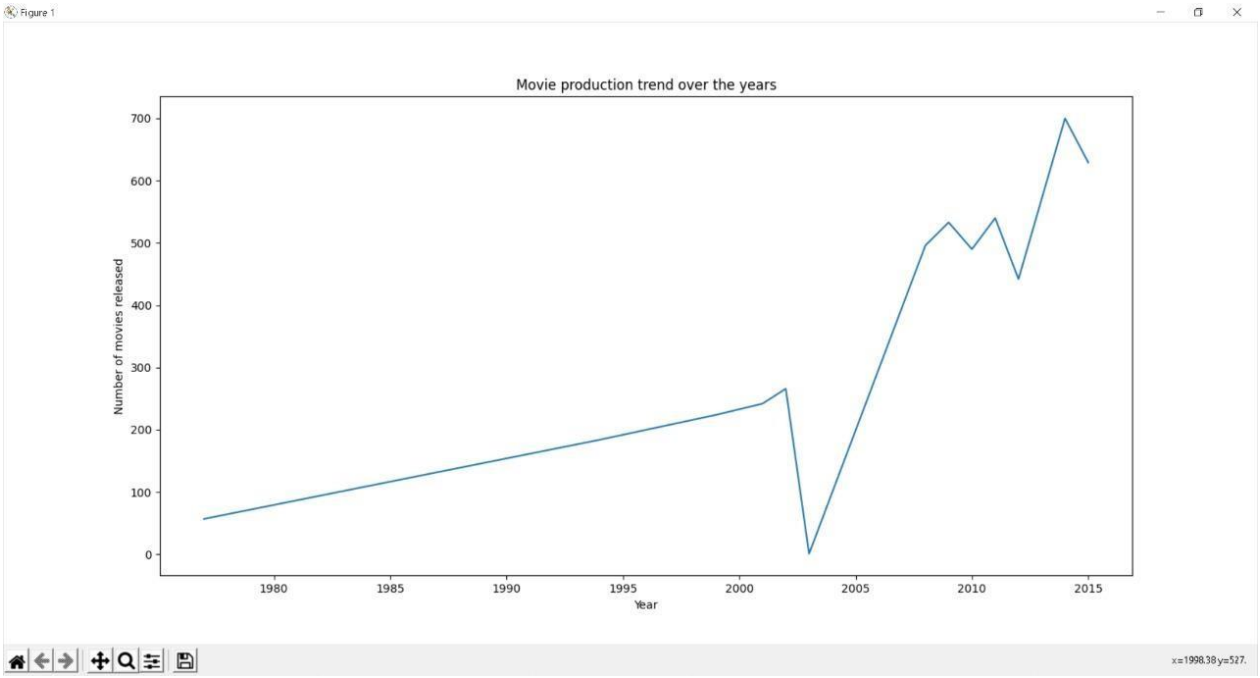
The above bar graph is a plot of Production Companies Vs Number of Movies.

Universal Pictures have produced the maximum number of movies of more than 160.



The above bar graph is a plot of the 'Top 20 most Expensive Movies'

Winger Tides was the most expensive movie with a budget of 4 million.



The above plot shows that the Movie production trend is gradually increasing over the years.

Conclusion:

From the analysis we can see that most movies classified themselves as dramas. Analysis will not be very effective if trying to figure out the single genre that will impact a movies success the most because it will be very unlikely that a movie will only be labeled as a drama.

So we identified the top genres so that it would be the most beneficial. For instance, based on one of the bar graph, the most popular genres that movies label themselves with, Action, Comedy, Drama, and Thriller.

New learnings from the programming assignment:

- To perform CRUD operations using MongoDB commands and Python user interface and backend was understood.
- Analysis of different kinds of data (Categorical, Numerical) using visualization techniques like bar plot, pie chart, line graph and scatter plot was understood.
- Techniques to clean the given data, map it to the appropriate NoSQL database and gain relevant information/insights by accurate analysis from the data given was understood.
- Hands-on experience of using tools such as PyMongo and MongoDB was obtained.

Future Enhancements:

- The current dataset does not include all the movies that are released . Hence , in future the dataset can be expanded further.
- We have only used about 8 variables to analyse the factors affecting the success rate of the movie. Hence , in future more variables can considered to give more accurate result.
- This analysis and prediction model can also be used to recommend movies according to their wish.