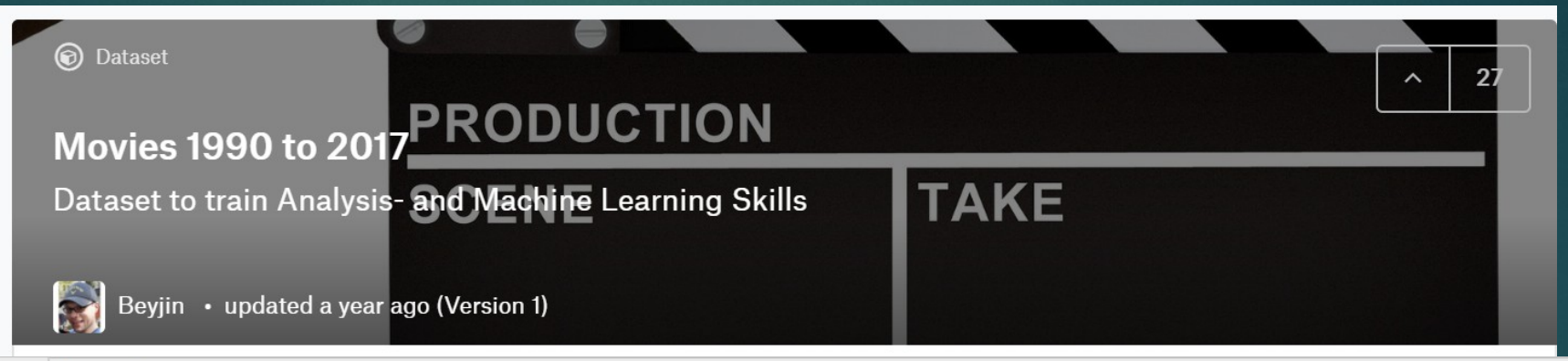
PREDICTING KEY METRICS FOR MOVIE PRODUCTION

BY NAVEEN ARUMANDLA

Key Questions Movie Producers Need Answers For

- Why this project?
- Why now?
- Why with this group of talent?
- With this investment of money?
- **What’s the genre?**
- **Who is the audience?**
- **Will this win an award?**
- **Will the audience like this?**
- **What is the probability of success**?

Dataset



In [1]:

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5  import datetime
6  movies = pd.read_csv('movies-1990-to-2017/Movie_Movies.csv') #Movies dataset with movies from 1900 to 2018
7  movies.head(5)
```

```
C:\Users\Jarvis\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3049: DtypeWarning: Columns (14) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[1]:

	Awards	Country	DVD	Director	Language	Plot	Poster	Production	Rated	Released	Runtime	Title	Type	Website	Year	imdbID	i
0	NaN	USA	NaN	Rose Cummings	English	Rachel constantly hears her baby cry from the ...	NaN	NaN	NaN	26 Apr 2012	20 min	Baby's Breath	movie	NaN	2012	tt2268369	
1	NaN	USA	NaN	James Byrne	NaN	The struggle against unfortunate circumstances...	NaN	NaN	NaN	NaN	9 min	Winter Trees	movie	NaN	2008	tt1560760	
2	NaN	USA	NaN	Dimitri Buchowetzki	NaN	NaN	NaN	NaN	NaN	27 Mar 1926	50 min	The Crown of Lies	movie	NaN	1926	tt0016750	

Data Cleaning

- This is mostly a string dataset with no numeric values. So these would have to be converted accordingly

```
In [2]: 1 movies.describe()
```

Out[2]:

	imdbRating
count	62073.000000
mean	6.416236
std	1.360472
min	1.000000
25%	5.600000
50%	6.500000
75%	7.300000
max	10.000000

```
In [3]: 1 movies.dtypes
```

```
Out[3]: Awards          object
        Country         object
        DVD             object
        Director        object
        Language        object
        Plot            object
        Poster          object
        Production      object
        Rated           object
        Released        object
        Runtime         object
```

- Filter on movies which were released from 1990

- New column to denote award winning movie with 1 and non-award winning movie with 0.

- Remove columns not required for analysis.

- Merge another database with information on ‘Genre’

- Resulting dataframe has multiple lines for same movie due to multiple genres for a movie

In [4]:
```
movies['Released'] = movies['Released'].astype('datetime64[ns]') #change the 'Released' column to date data type
movies = movies[movies['Released'] > datetime.date(1989,12,31)] #only movies released after January 1990 will appear
```

In [8]:
```
movies['Award?']=np.where(movies['Awards'].isna(),0, 1)
del movies['Poster']
del movies['DVD']
del movies['Rated']
del movies['Type']
del movies['Website']
movies.head(5)
```

Out[8]:

	Awards	Country	Director	Language	Plot	Production	Released	Runtime	Title	Year	imdbID	imdbRating	imdbVotes	Award?
0	NaN	USA	Rose Cummings	English	Rachel constantly hears her baby cry from the ...	NaN	2012-04-26	20 min	Baby's Breath	2012	tt2268369	NaN	NaN	0
3	NaN	USA	Julia Hechler	English	A Gift introduces Samuel Green, Washington Sta...	NaN	2013-05-27	2 min	A Gift	2013	tt3405286	NaN	NaN	0
4	NaN	Sri Lanka	Udara Siriruwan	Sinhalese	NaN	NaN	2014-03-20	23 min	Journey	2014	tt3816698	NaN	NaN	0

In [9]:
```
genres = pd.read_csv('movies-1990-to-2017/Movie_Genres.csv')
movies = pd.merge(movies,genres, on='imdbID') #Merge two CSV files
movies.head(5)
```

Out[9]:

Country	Director	Language	Plot	Production	Released	Runtime	Title	Year	imdbID	imdbRating	imdbVotes	Award?	Unnamed: 0	Genre
USA	Rose Cummings	English	Rachel constantly hears her baby cry from the ...	NaN	2012-04-26	20 min	Baby's Breath	2012	tt2268369	NaN	NaN	0	0	Short
USA	Rose Cummings	English	Rachel constantly hears her baby cry from the ...	NaN	2012-04-26	20 min	Baby's Breath	2012	tt2268369	NaN	NaN	0	1	Drama
USA	Julia Hechler	English	A Gift introduces Samuel Green, Washington Sta...	NaN	2013-05-27	2 min	A Gift	2013	tt3405286	NaN	NaN	0	6	Documentary
			A Gift introduces											

- Duplicate rows with same movie title removed

- Column ‘Runtime’ cannot be converted to string because it has values followed by strings “h” and “,”

- Slice function used to remove “min” string at the end of column value

- Two columns created to identify which rows have “h” and “,”. Identified rows were removed from Dataframe

In [10]:
```
movies = movies.drop_duplicates(subset = 'Title')
movies.head(5)
```

Out[10]:

Country	Director	Language	Plot	Production	Released	Runtime	Title	Year	imdbID	imdbRating	imdbVotes	Award?	Unnamed: 0	Genre
USA	Rose Cummings	English	Rachel constantly hears her baby cry from the ...	NaN	2012-04-26	20 min	Baby's Breath	2012	tt2268369	NaN	NaN	0	0	Short
USA	Julia Hechler	English	A Gift introduces Samuel Green, Washington Sta...	NaN	2013-05-27	2 min	A Gift	2013	tt3405286	NaN	NaN	0	6	Documentary
Sri Lanka	Udara Siriruwan	Sinhalese	NaN	NaN	2014-03-20	23 min	Journey	2014	tt3816698	NaN	NaN	0	8	Short

In [13]:
```
sub1 = 'h'
sub2 = ','
sub3 = '.'
movies['Runtime'] = movies['Runtime'].str.slice(0,-4) #remove last four characters from 'Runtime' column which has 'min' str
movies['Runtime_index1'] = movies['Runtime'].str.find(sub1)
movies['Runtime_index2'] = movies['Runtime'].str.find(sub2)
movies = movies.dropna(subset=['Runtime'])
movies.head(5)
```

Out[13]:

Plot	Production	Released	Runtime	Title	Year	imdbID	imdbRating	imdbVotes	Award?	Unnamed: 0	Genre	Runtime_index1	Runtime_index2
Rachel onstantly hears her baby cry om the ...	NaN	2012-04-26		Baby's Breath	2012	tt2268369	NaN	NaN	0	0	Short	-1.0	-1.0
A Gift ıtroduces Samuel Green, ashington Sta...	NaN	2013-05-27		A Gift	2013	tt3405286	NaN	NaN	0	6	Documentary	-1.0	-1.0
NaN	NaN	2014-03-20		Journey	2014	tt3816698	NaN	NaN	0	8	Short	-1.0	-1.0

- Dataset has been cleaned to contain necessary variables
1) Runtime (in float)
2) Movie Rating (in float)
3) Votes (in int)
4) Identifier for award winning movie

In [20]:
```
movies['Runtime'] = pd.to_numeric(movies['Runtime']).astype(float)
movies['imdbRating'] = pd.to_numeric(movies['imdbRating']).astype(float)
movies['imdbVotes'] = movies['imdbVotes'].fillna('0',inplace=True)
movies['Votes_index1'] = movies['imdbVotes'].str.replace(',','')
movies['imdbVotes'] = pd.to_numeric(movies['imdbVotes']).astype(float)
movies.dtypes
```

Out[20]:
```
Awards                    object
Country                   object
Director                  object
Language                  object
Plot                      object
Production                object
Released          datetime64[ns]
Runtime                  float64
Title                     object
Year                      object
imdbID                    object
imdbRating               float64
imdbVotes                float64
Award?                     int32
Unnamed: 0                 int64
Genre                     object
Runtime_index1             int32
Runtime_index2             int32
Votes_index1              object
dtype: object
```

Who is the audience? What Genre?

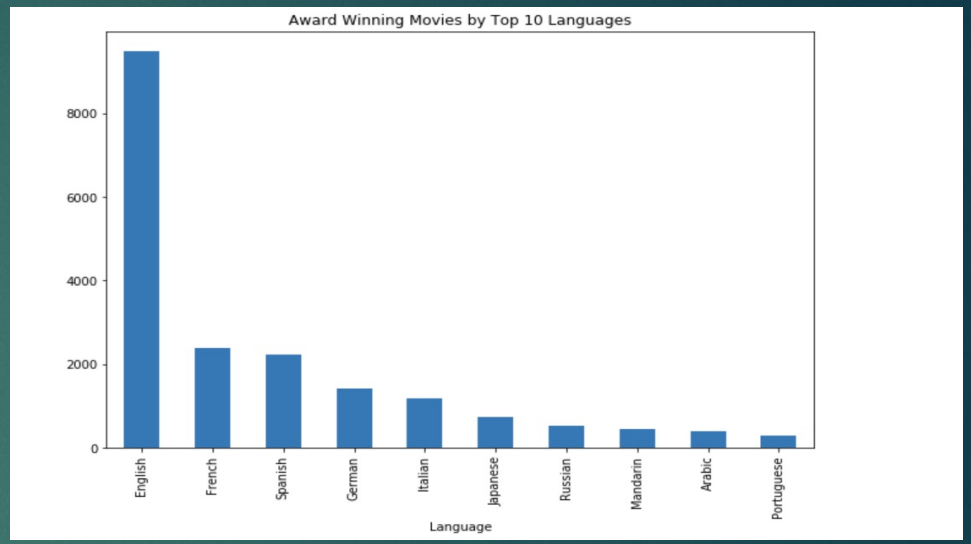
In [24]:
```
def new_language(df):
    new_df = \
    (df.set_index(df.columns.drop('Language',1).tolist())
     .Language.str.split(', ',expand=True)
     .stack()
     .reset_index()
     .rename(columns={0:'Language'})
     .loc[:,df.columns])
    return new_df
```

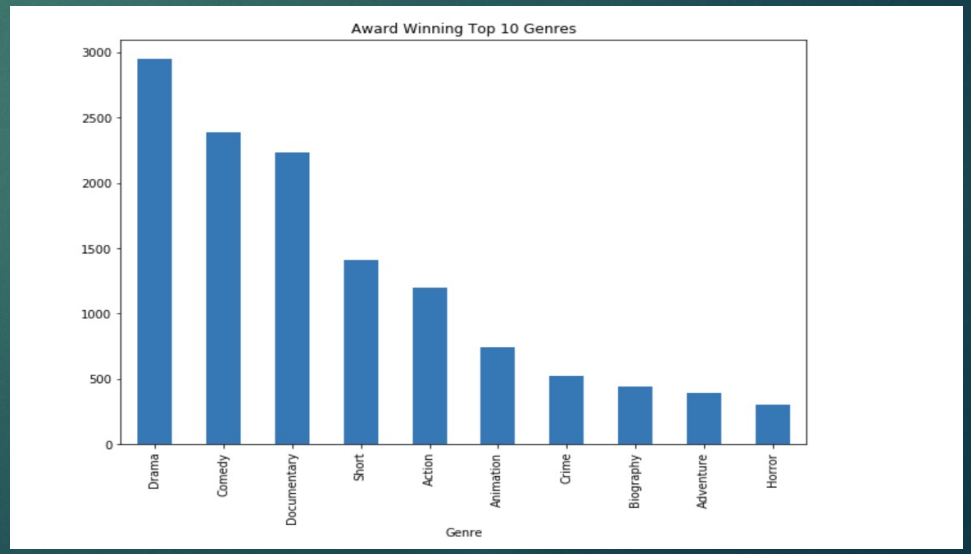
In [28]:
```
val1 = input("Enter preferred movie type to choose a genre and language:\n1. Award winning movie\n2. Highest rated movies (n

if val1 == '1':
    award_movies = movies[movies['Award?']==1]
    print(award_movies.groupby('Genre').size().sort_values(ascending=False).head(10).plot(kind='bar',figsize=(10,7),title="A
    reshaped_award = new_language(award_movies)
    x=reshaped_award.groupby(['Language']).size().sort_values(ascending=False).head(10)
    print("Award Winning Movies by Top 10 Languages")
    print(x)

elif val1 == '2':
    rating = movies[movies['Award?']==0]
    rating['imdbVotes'] = rating['imdbVotes']>10000
    rating['imdbRating'] = rating['imdbRating'] > 8
    print(rating.groupby('Genre').mean()['imdbRating'].sort_values(ascending=False).head(10).plot(kind='bar',figsize=(10,7),
    reshaped_rated = new_language(rating)
    x=reshaped_rated.groupby(['Language']).size().sort_values(ascending=False).head(10)
    print("High Rated Movies (>10000 votes and >8.0 rating) by Top 10 Languages")
    print(x)
```

```
Enter preferred movie type to choose a genre and language:
1. Award winning movie
2. Highest rated movies (no awards)
3. Award winning and high rating
```





- Typing 1, gives me a visual for the top 10 languages in which award winning movies were made and top 10 genres they were made in
- Similar graphs will be displayed for choices 2, 3

Probability of success?

```
In [*]:
val2 = input("Enter preferred genre:")

prob_award = movies[movies['Award?'] == 1]
prob_award['imdbRating'] = prob_award['imdbRating']<8
count_genre_award = prob_award.loc[prob_award['Genre']== val2].count()
probability1 = ((count_genre_award['Genre'])/(prob_award['Genre'].count()))*100
print("Probability of '%s' genre winning an award is %.0f%%"% (val2,probability1))

prob_rated = movies[movies['Award?'] == 0]
prob_rated['imdbVotes'] = prob_rated['imdbVotes']>10000
prob_rated['imdbRating'] = prob_rated['imdbRating']>8
count_genre_rated = prob_rated.loc[prob_rated['Genre']== val2].count()
prob_rated['Genre'].count()
probability2 = ((count_genre_rated['Genre'])/(prob_rated['Genre'].count()))*100
print("Probability of '%s' genre being rated above 8 without any award is %.0f%%"% (val2,round(probability2)))

prob_both = movies[movies['Award?'] == 0]
prob_both['imdbVotes'] = prob_both['imdbVotes']>10000
prob_both['imdbRating'] = prob_both['imdbRating']>8
count_genre_both = prob_both.loc[prob_both['Genre']== val2].count()
prob_both['Genre'].count()
probability3 = ((count_genre_both['Genre'])/(prob_both['Genre'].count()))*100
print("Probability of '%s' genre winning an award and being rated above 8 is %.0f%%"% (val2,round(probability3)))

Enter preferred genre:
```

```
Enter preferred genre:Drama
C:\Users\Jarvis\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#index
  after removing the cwd from sys.path.
C:\Users\Jarvis\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#index
  # Remove the CWD from sys.path while we load stuff.
C:\Users\Jarvis\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#index
  # This is added back by InteractiveShellApp.init_path()

Probability of 'Drama' genre winning an award is 23%
Probability of 'Drama' genre being rated above 8 without any award is 6%
Probability of 'Drama' genre winning an award and being rated above 8 is 6%
```

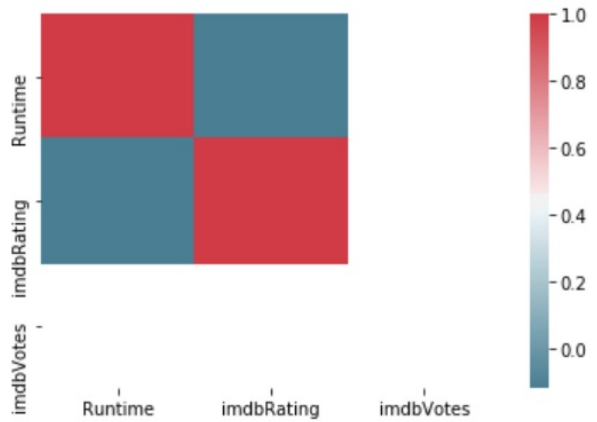
Inputting the preferred type of genre, will provide the probability of the genre:

1) Winning an award
2) Being rated 8.0 or more on IMDB
3) Both being awarded and being rated above 8.0

Relationship between Variables

```
In [21]:
1 corr = movies.loc[:,movies.dtypes == 'float64'].corr()
2 sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, cmap=sns.diverging_palette(220, 10, as_cmap=True))

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x160f79aa128>
```



“Number of votes” on IMDB and “IMDB rating” have a positive but weak correlation.

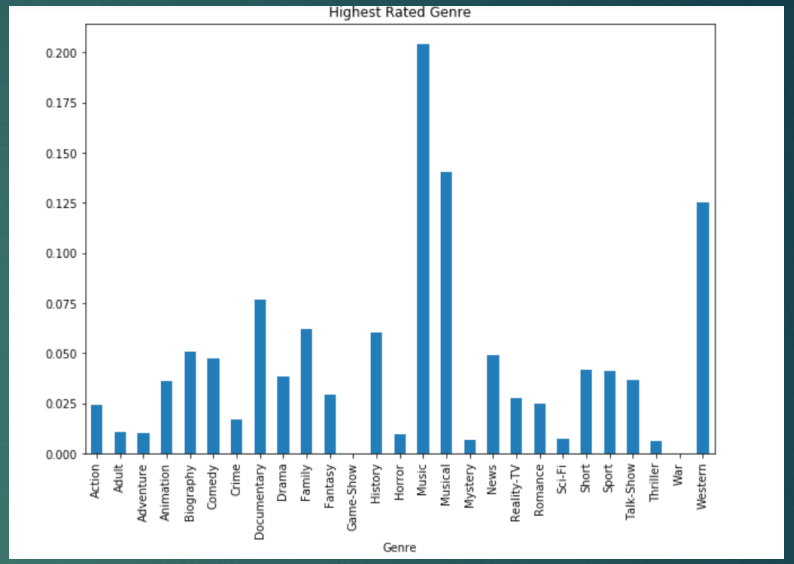
THEREFORE, more voting population means better movie ratings

Evidence

- DATA MODEL VALIDATED

Normalizing Data

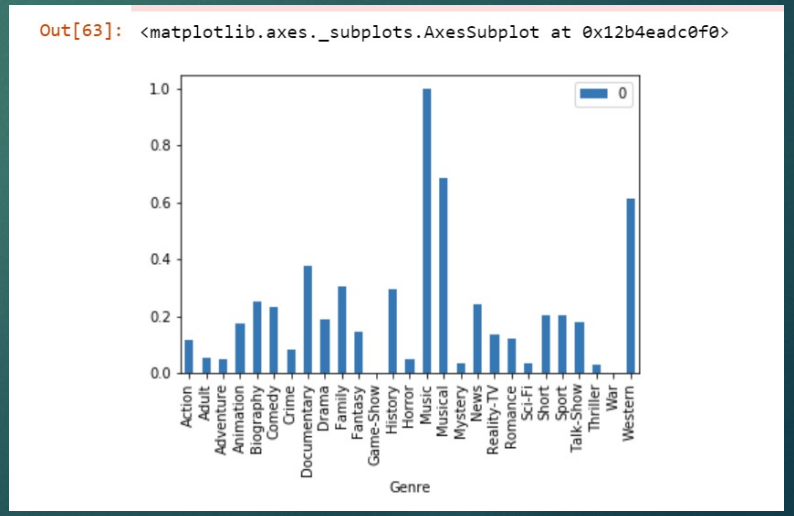
- While calculating average rating of genres, values had massive variance



- So values were normalized to test for accuracy of analysis

```
In [63]:
#normalize means to check if result remains same

norm_mean = movies[movies['Award?']==0]
norm_mean['imdbVotes'] = norm_mean['imdbVotes']>10000
norm_mean['imdbRating'] = norm_mean['imdbRating'] > 8
norm_mean = norm_mean.groupby('Genre').mean()['imdbRating']
norm_mean = pd.DataFrame(norm_mean)
scaler = preprocessing.MinMaxScaler()
norm_mean_scaled = scaler.fit_transform(norm_mean)
scaled_mean = pd.DataFrame(norm_mean_scaled)
scaled_mean.set_index(norm_mean.index).plot(kind='bar')
```



Additional Guidelines

```
In [*]:
1  val3 = input("Enter preferred language:")
2
3  run_award = movies[movies['Award?'] == 1]
4  avg_run_award = run_award.loc[run_award['Language']== val3].mean()
5  avg_run_award = avg_run_award['Runtime'].mean()
6  print("Average runtime of award winning movie in '%s' language is %.0f minutes" % (val3,avg_run_award))
7
8  run_rated = movies[movies['Award?'] == 0]
9  run_rated['imdbVotes'] = run_rated['imdbVotes']>10000
10 run_rated['imdbRating'] = run_rated['imdbRating']>8
11 avg_run_rated = run_rated.loc[run_rated['Language']== val3].mean()
12 avg_run_rated = avg_run_rated['Runtime'].mean()
13 print("Average runtime of high rated (>8.0) movie with no awards in '%s' language is %.0f minutes" % (val3,avg_run_rated))
14
15 run_both = movies[movies['Award?'] == 0]
16 run_both['imdbVotes'] = run_both['imdbVotes']>10000
17 run_both['imdbRating'] = run_both['imdbRating']>8
18 avg_run_both = run_both.loc[run_both['Language']== val3].mean()
19 avg_run_both = avg_run_both['Runtime'].mean()
20 print("Average runtime of award winning high rated (>8.0) movie in '%s' language is %.0f minutes" % (val3,avg_run_both))

Enter preferred language:
```

```
Enter preferred language:Japanese
Average runtime of award winning movie in 'Japanese' language is 104 minutes

C:\Users\Jarvis\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versu
  if __name__ == '__main__':
C:\Users\Jarvis\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versu
  # Remove the CWD from sys.path while we load stuff.

Average runtime of high rated (>8.0) movie with no awards in 'Japanese' language is 89 minutes
Average runtime of award winning high rated (>8.0) movie in 'Japanese' language is 89 minutes
```

Inputting the preferred type of language, will provide the average runtime of the three choices by language

Other improvements which can be made?

- Budget required
- Choosing a cast member
- Correlation between box office collection and budget