MOVIE RATING PREDICTION WITH PYTHON

Recommendation systems are becoming increasingly important in today's extremely busy world. People are always short on time with the myriad tasks they need to accomplish in the limited 24 hours. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources.

The purpose of a recommendation system basically is to search for content that would be interesting to an individual. Moreover, it involves a number of factors to create personalised lists of useful and interesting content specific to each user/individual. Recommendation systems are Artificial Intelligence based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile, search/browsing history, what other people with similar traits/demographics are watching, and how likely are you to watch those movies. This is achieved through predictive modeling and heuristics with the data available.

Import Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings('ignore')
```

Load the data

In [2]:
```
movies = pd.read_csv(r"C:\Users\user\Downloads\movies.dat", sep='::', engine='python', e
movies
```

Out[2]:

	1	Toy Story (1995)	Animation|Children's|Comedy
0	2	Jumanji (1995)	Adventure|Children's|Fantasy
1	3	Grumpier Old Men (1995)	Comedy|Romance
2	4	Waiting to Exhale (1995)	Comedy|Drama
3	5	Father of the Bride Part II (1995)	Comedy
4	6	Heat (1995)	Action|Crime|Thriller
...	...	...	...
3877	3948	Meet the Parents (2000)	Comedy
3878	3949	Requiem for a Dream (2000)	Drama
3879	3950	Tigerland (2000)	Drama
3880	3951	Two Family House (2000)	Drama
3881	3952	Contender, The (2000)	Drama|Thriller

3882 rows × 3 columns

In [3]:
```
movies.columns =['MovieID', 'Title', 'Genres']
movies.dropna(inplace=True)
movies.head()
```

Out[3]:

	MovieID	Title	Genres
0	2	Jumanji (1995)	Adventure|Children's|Fantasy
1	3	Grumpier Old Men (1995)	Comedy|Romance
2	4	Waiting to Exhale (1995)	Comedy|Drama
3	5	Father of the Bride Part II (1995)	Comedy
4	6	Heat (1995)	Action|Crime|Thriller

In [4]:
```
movies.tail()
```

Out[4]:

	MovieID	Title	Genres
3877	3948	Meet the Parents (2000)	Comedy
3878	3949	Requiem for a Dream (2000)	Drama
3879	3950	Tigerland (2000)	Drama
3880	3951	Two Family House (2000)	Drama
3881	3952	Contender, The (2000)	Drama|Thriller

In [5]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3882 entries, 0 to 3881
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   MovieID  3882 non-null   int64 
 1   Title    3882 non-null   object
 2   Genres   3882 non-null   object
dtypes: int64(1), object(2)
memory usage: 91.1+ KB
```

In [6]:
```
movies.describe()
```

Out[6]:

	MovieID
count	3882.000000
mean	1986.560793
std	1146.483260
min	2.000000
25%	983.250000
50%	2010.500000
75%	2980.750000
max	3952.000000

In [7]:
```
movies.dtypes
```

Out[7]:
```
MovieID     int64
Title      object
Genres     object
dtype: object
```

In [8]:
```
movies.shape
```

Out[8]:
```
(3882, 3)
```

In [9]:
```
movies.isnull()
```

Out[9]:

	MovieID	Title	Genres
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
3877	False	False	False
3878	False	False	False
3879	False	False	False
3880	False	False	False
3881	False	False	False

3882 rows × 3 columns

In [10]:
```
ratings=pd.read_csv(r'C:\Users\user\Downloads\ratings.dat.zip',sep='::', engine='python'
ratings.columns =['UserID', 'MovieID', 'Rating', 'Timestamp']
ratings.dropna(inplace=True)

#Read the sample ratings dataset
ratings.head()
```

Out[10]:

	UserID	MovieID	Rating	Timestamp
0	1	661	3	978302109
1	1	914	3	978301968
2	1	3408	4	978300275
3	1	2355	5	978824291
4	1	1197	3	978302268

In [11]:
```
#input users dataset
users=pd.read_csv(r'C:\Users\user\Downloads\users.dat',sep='::',engine='python')
users.columns =['UserID', 'Gender', 'Age', 'Occupation', 'Zip-code']
users.dropna(inplace=True)

#Read the sample users dataset
users.head()
```

Out[11]:

	UserID	Gender	Age	Occupation	Zip-code
0	2	M	56	16	70072
1	3	M	25	15	55117
2	4	M	45	7	02460
3	5	M	25	20	55455
4	6	F	50	9	55117

In [12]:
```
#Merge the ratings and users with movieID and UserID
ratings_user = pd.merge(ratings,users, on=['UserID'])
ratings_movie = pd.merge(ratings,movies, on=['MovieID'])

master_data = pd.merge(ratings_user,ratings_movie,
                       on=['UserID', 'MovieID', 'Rating'])[['MovieID', 'Title', 'UserID'

master_data.head()
```

Out[12]:

	MovieID	Title	UserID	Age	Gender	Occupation	Rating
0	1357	Shine (1996)	2	56	M	16	5
1	3068	Verdict, The (1982)	2	56	M	16	4
2	1537	Shall We Dance? (Shall We Dansu?) (1996)	2	56	M	16	4
3	647	Courage Under Fire (1996)	2	56	M	16	3
4	2194	Untouchables, The (1987)	2	56	M	16	4

In [13]:
```
# all 5 rating movies list count = 225473
master_data[master_data['Rating'] == 5]
```

Out[13]:

	MovieID	Title	UserID	Age	Gender	Occupation	Rating
0	1357	Shine (1996)	2	56	M	16	5
6	2268	Few Good Men, A (1992)	2	56	M	16	5
10	3468	Hustler, The (1961)	2	56	M	16	5
15	3578	Gladiator (2000)	2	56	M	16	5
26	1610	Hunt for Red October, The (1990)	2	56	M	16	5
...	...	...	...	...	...	...	...
998065	1077	Sleeper (1973)	6040	25	M	6	5
998070	2022	Last Temptation of Christ, The (1988)	6040	25	M	6	5
998071	2028	Saving Private Ryan (1998)	6040	25	M	6	5
998076	1094	Crying Game, The (1992)	6040	25	M	6	5
998077	562	Welcome to the Dollhouse (1995)	6040	25	M	6	5

225473 rows × 7 columns

In [14]:
```
# all 5 rating movies list and Age Lass Then 25 count = 47163
master_data[(master_data['Rating'] == 5) & (master_data['Age'] < 25 ) ]
```

Out[14]:

	MovieID	Title	UserID	Age	Gender	Occupation	Rating
1883	2987	Who Framed Roger Rabbit? (1988)	18	18	F	3	5
1884	2989	For Your Eyes Only (1981)	18	18	F	3	5
1885	2622	Midsummer Night's Dream, A (1999)	18	18	F	3	5
1889	1683	Wings of the Dove, The (1997)	18	18	F	3	5
1893	3793	X-Men (2000)	18	18	F	3	5
...	...	...	...	...	...	...	...
996033	150	Apollo 13 (1995)	6031	18	F	0	5
996036	1010	Love Bug, The (1969)	6031	18	F	0	5
996038	1036	Die Hard (1988)	6031	18	F	0	5
996039	2001	Lethal Weapon 2 (1989)	6031	18	F	0	5
996043	1097	E.T. the Extra-Terrestrial (1982)	6031	18	F	0	5

47163 rows × 7 columns

In [15]:
```
# all 5 rating movies list and Age Lass Then 25 count = 47163
master_data[(master_data['Rating'] < 3) & (master_data['Age'] < 25 )]
```

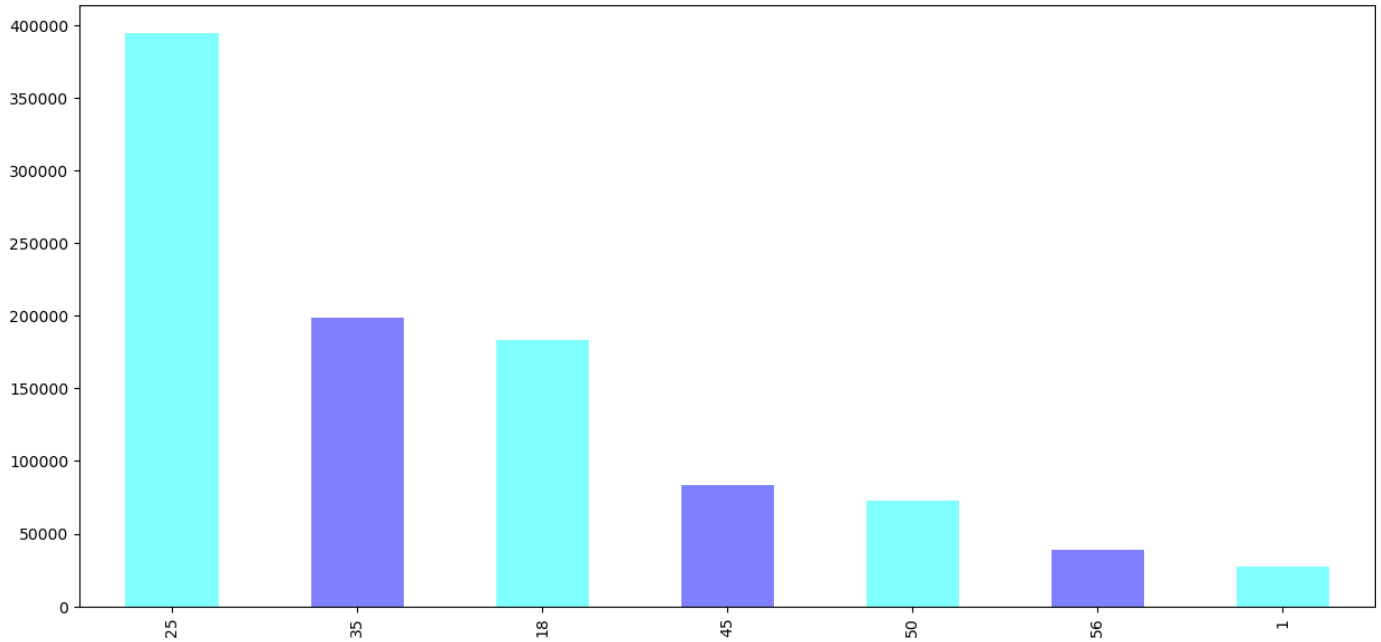
Out[15]:

	MovieID	Title	UserID	Age	Gender	Occupation	Rating
1898	1186	Sex, Lies, and Videotape (1989)	18	18	F	3	1
1902	3438	Teenage Mutant Ninja Turtles (1990)	18	18	F	3	2
1905	3439	Teenage Mutant Ninja Turtles II: The Secret of...	18	18	F	3	1
1907	1690	Alien: Resurrection (1997)	18	18	F	3	1
1909	2	Jumanji (1995)	18	18	F	3	2
...	...	...	...	...	...	...	...
996023	785	Kingpin (1996)	6031	18	F	0	2
996025	1648	House of Yes, The (1997)	6031	18	F	0	2
996030	1394	Raising Arizona (1987)	6031	18	F	0	2
996034	151	Rob Roy (1995)	6031	18	F	0	1
996041	553	Tombstone (1993)	6031	18	F	0	1

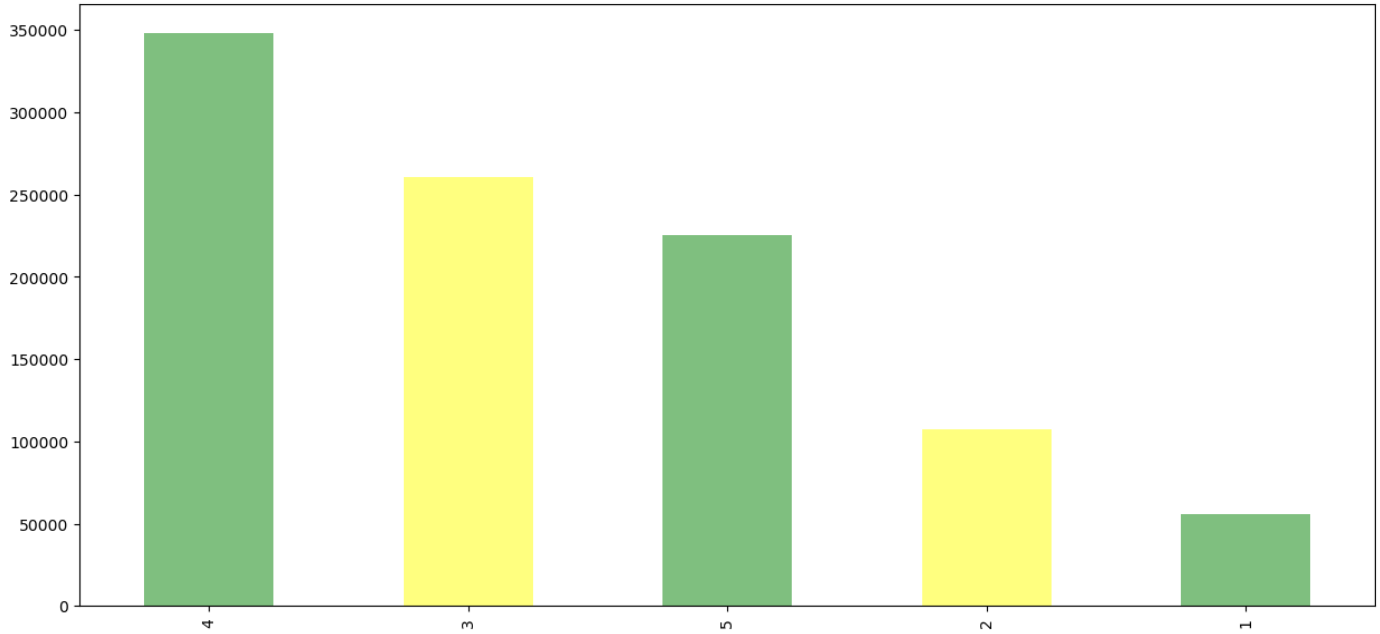
40329 rows × 7 columns

Data Visualization

In [16]:
```
master_data['Age'].value_counts().plot(kind='bar', color= ['cyan', 'blue'],alpha=0.5,fig
plt.show()
```



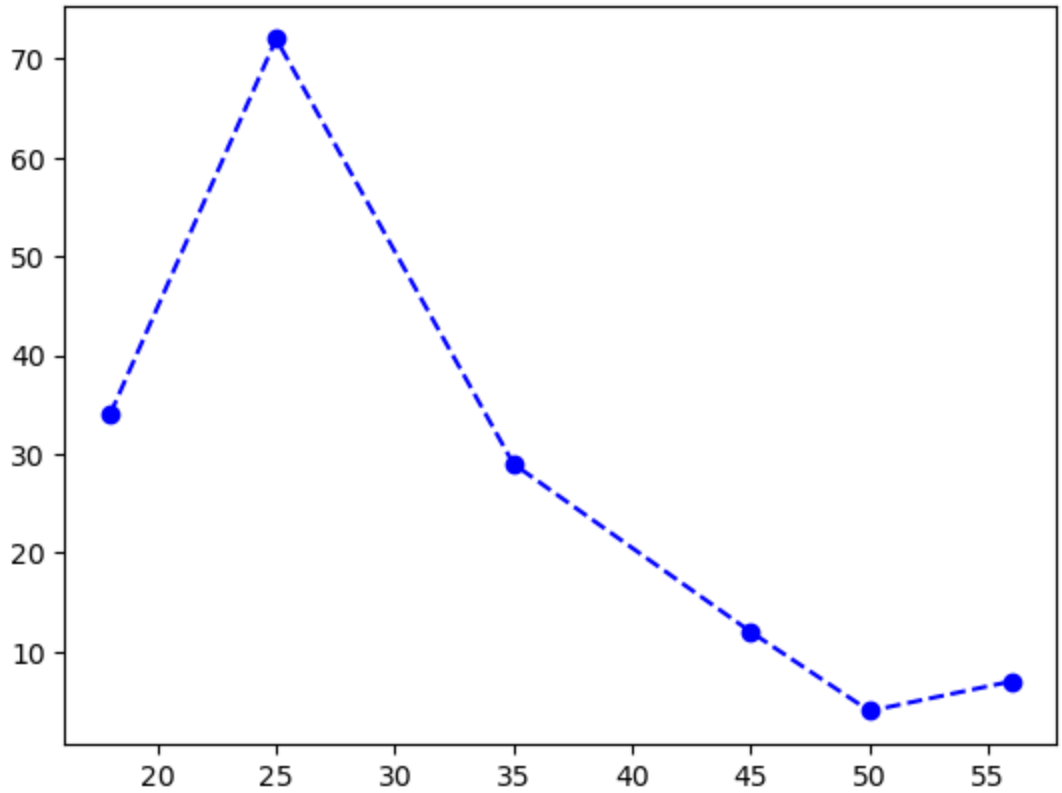
In [17]:
```
master_data['Rating'].value_counts().plot(kind='bar', color=['green', 'yellow'],alpha=0.
plt.show()
```



In [18]:
```
res = master_data[master_data.Title == "Only You (1994)"]

plt.plot(res.groupby("Age")["MovieID"].count(),'--bo')
res.groupby("Age")["MovieID"].count()
```

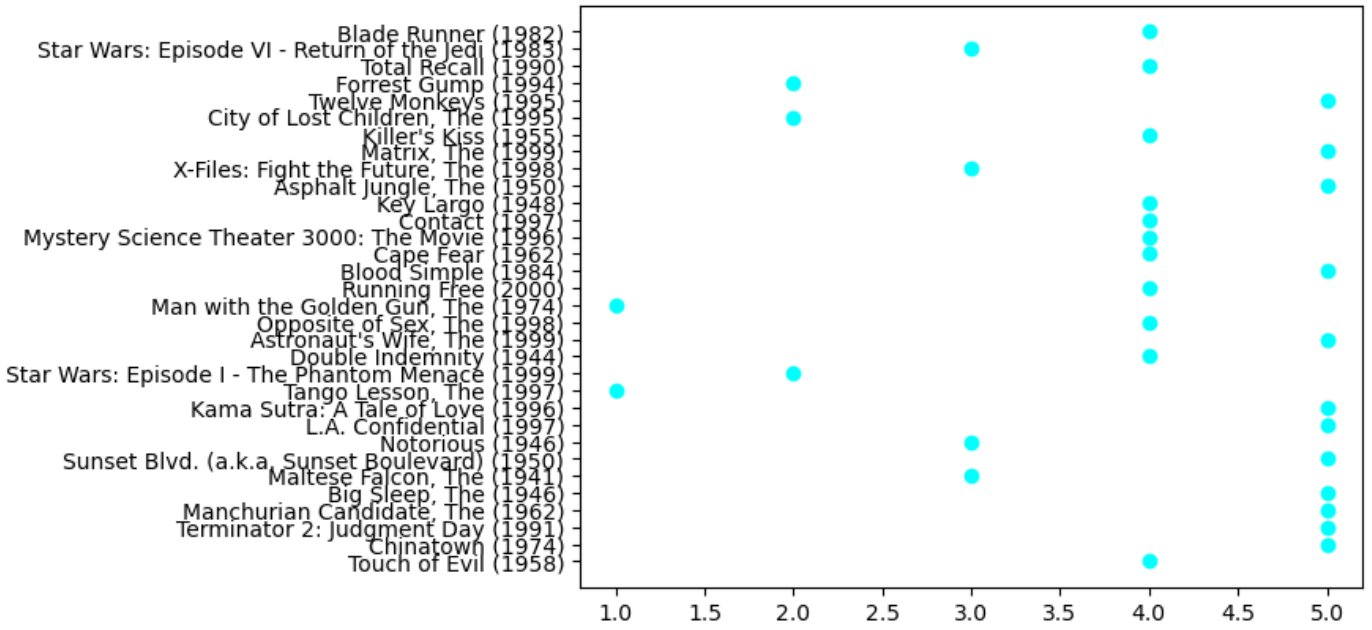
Out[18]:
```
Age
18    34
25    72
35    29
45    12
50     4
56     7
Name: MovieID, dtype: int64
```



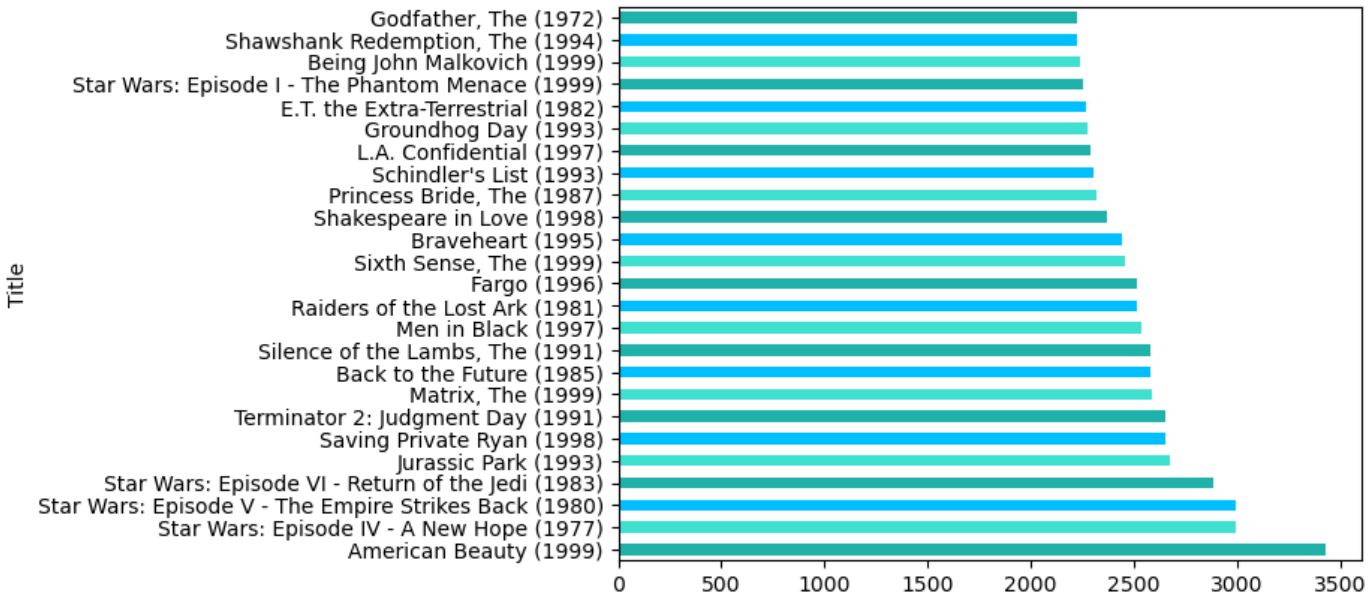
In [19]:
```
#Find the ratings for all the movies reviewed by for a particular user of user id = 700

res = master_data[master_data.UserID == 700]
```

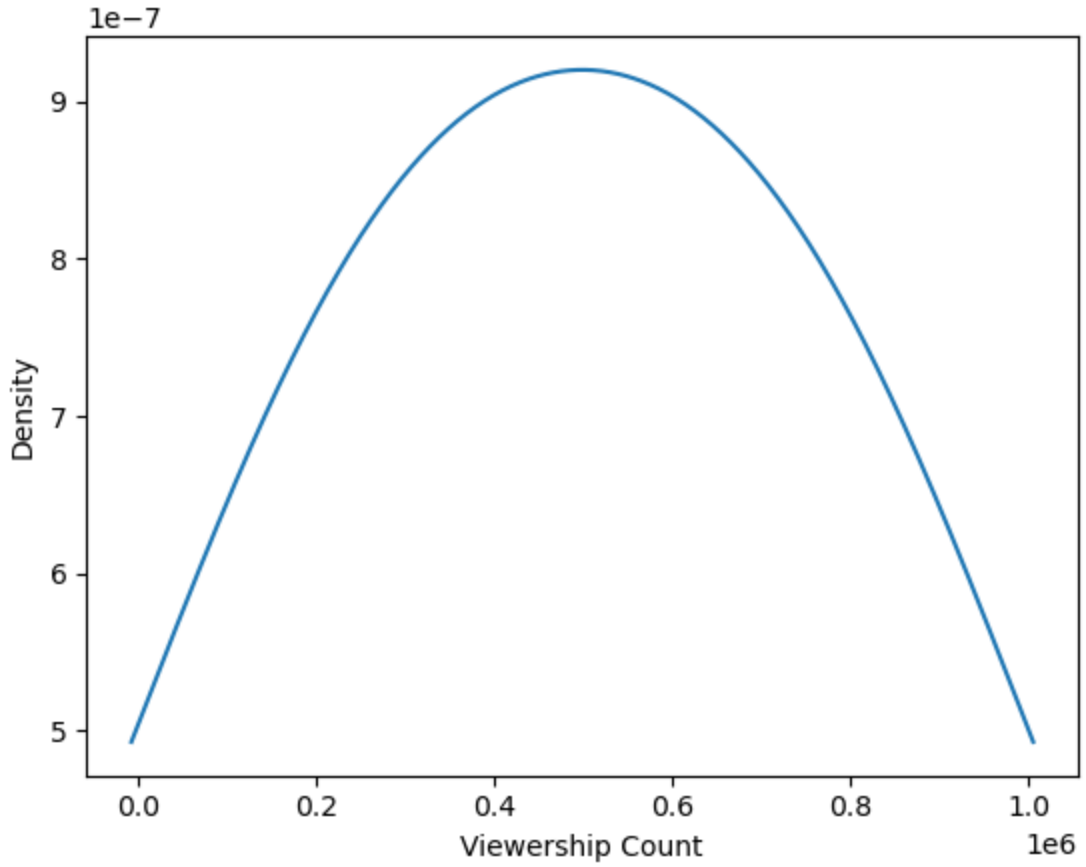
```
plt.scatter(y=res.Title, x=res.Rating , color = 'aqua')
plt.show()
```



In [20]:
```
res = master_data.groupby("Title").size().sort_values(ascending=False)[:25]
plt.ylabel("Title")
plt.xlabel("Viewership Count")
res.plot(kind="barh", color = ['lightseagreen', 'turquoise', 'deepskyblue'])
plt.show()
```

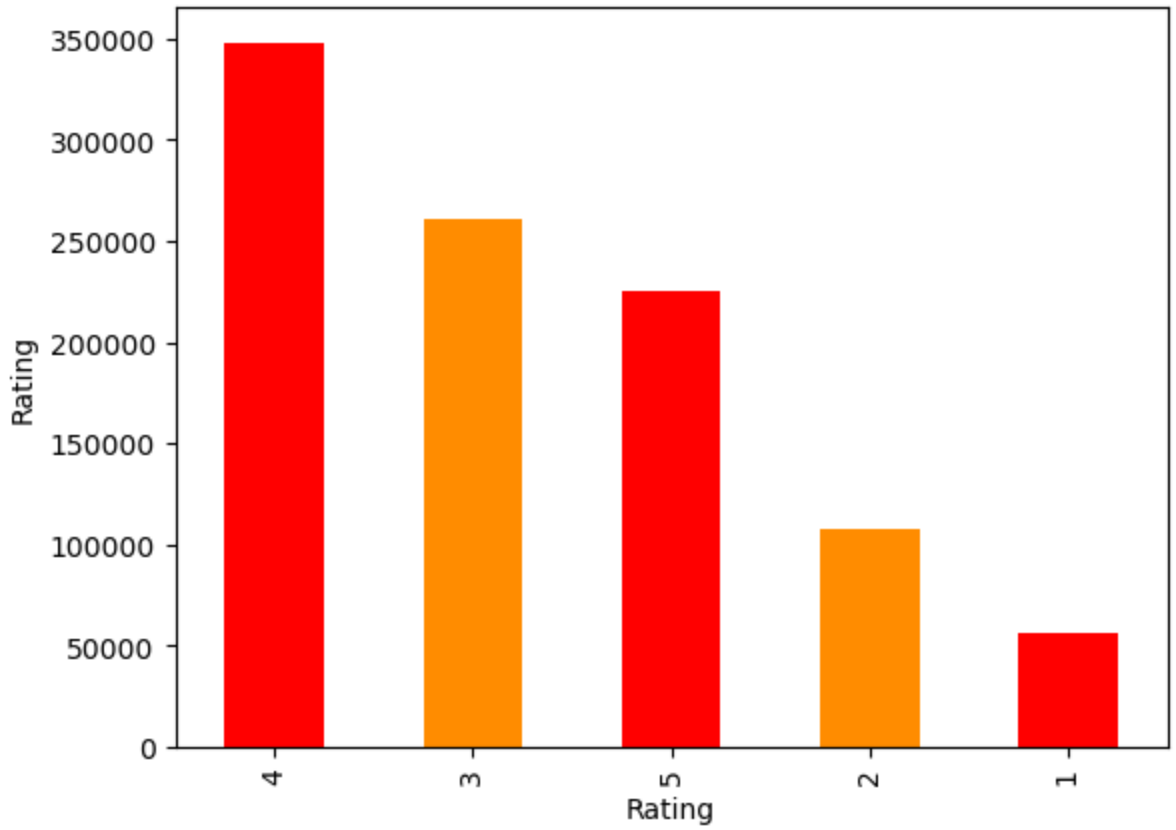


In [21]:
```
res = master_data.groupby("Gender").size().sort_values(ascending=False)[:25]
plt.ylabel("Gender")
plt.xlabel("Viewership Count")
res.plot(kind="kde")
plt.show()
```



In [22]:
```
res = master_data.groupby("Rating").size().sort_values(ascending=False)[:25]
plt.ylabel("Rating")
plt.xlabel("Viewership Count")
res.plot(kind='bar', color= ['red', 'darkorange'])
```

Out[22]:
```
<Axes: xlabel='Rating', ylabel='Rating'>
```



Model for Machine Learning

In [23]:
```
#First 500 extracted records
first_500 = master_data[500:]
first_500.dropna(inplace=True)
```

In [24]:
```
#Use the following features:movie id,age,occupation
features = first_500[['MovieID','Age','Occupation']].values
```

In [25]:
```
#Use rating as label
labels = first_500[['Rating']].values
```

In [26]:
```
#Create train and test data set
train, test, train_labels, test_labels = train_test_split(features,labels,test_size=0.33
```

Logistic Regression

In [27]:
```
logreg = LogisticRegression()
logreg.fit(train, train_labels)
Y_pred = logreg.predict(test)
acc_log = round(logreg.score(train, train_labels) * 100, 2)
acc_log
```

Out[27]:
```
34.86
```

K Nearest Neighbors Classifier

In [28]:
```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(train, train_labels)
Y_pred = knn.predict(test)
acc_knn = round(knn.score(train, train_labels) * 100, 2)
acc_knn
```

Out[28]:
```
44.86
```

Gaussian Naive Bayes

In [29]:
```
gaussian = GaussianNB()
gaussian.fit(train, train_labels)
Y_pred = gaussian.predict(test)
acc_gaussian = round(gaussian.score(train, train_labels) * 100, 2)
acc_gaussian
```

Out[29]:
```
34.88
```

Perceptron

In [30]:
```
perceptron = Perceptron()
perceptron.fit(train, train_labels)
Y_pred = perceptron.predict(test)
acc_perceptron = round(perceptron.score(train, train_labels) * 100, 2)
acc_perceptron
```

Out[30]: 33.05

Decision Tree

In [31]:
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(train, train_labels)
Y_pred = decision_tree.predict(test)
acc_decision_tree = round(decision_tree.score(train, train_labels) * 100, 2)
acc_decision_tree
```

Out[31]: 56.54

In [ ]:
```

```