In [159]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Import the three datasets

In [96]:

```python
movies = pd.read_csv(r'movies.dat', sep = "::", names = ['MovieID', 'Title', 'Genres'], engine='python')
```

help(pd.read_csv) engine : {'c', 'python'}, optional Parser engine to use. The C engine is faster while the python engine is currently more feature-complete.

In [97]:

```python
movies.head()
```

Out[97]:

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

In [98]:

```
ratings = pd.read_csv(r'ratings.dat', sep = "::", names = ['UserID','MovieID', 'Rating', 'Timestamp'], engine='python')
```

In [99]:

```
ratings.head()
```

Out[99]:

UserID	MovieID	Rating	Timestamp

0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

In [100]:

```
users = pd.read_csv(r'users.dat', sep = "::", names = ['UserID', 'Gender', 'Age', 'Occupation', 'Zip-Code'],
engine='python')
users.head()
```

Out[100]:

	UserID	Gender	Age	Occupation	Zip-Code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

In [101]:

```
movies.shape, users.shape, ratings.shape
```

Out[101]:

```
((3883, 3), (6040, 5), (1000209, 4))
```

Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserId)

In [102]:

```
movie_ratings = pd.merge(movies, ratings, on = "MovieID")
display (movie_ratings.head())
display (movie_ratings.shape)
```

	MovieID	Title	Genres	UserID	Rating	Timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268

1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474

(1000209, 6)

In [103]:

```
users.head()
```

Out[103]:

	UserID	Gender	Age	Occupation	Zip-Code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

In [104]:

```
data = pd.merge(movie_ratings, users, on = "UserID")

display (data.head())
display (data.shape)
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-Code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067

(1000209, 10)

In [105]:

```
# choose 10 - 20 K rows for your analysis
```

Explore the datasets using visual representations (graphs or tables), also include your comments on the following:

1. User Age Distribution¶

In [106]:

```
import matplotlib.pyplot as plt
from matplotlib.style import use
%matplotlib inline

# Visualize age distribution of users
users.Age.plot.hist(bins=50)
plt.style.use('ggplot')
plt.title('User Age Distribution')
plt.xlabel('Age')
plt.show()
```

User rating of the movie “Toy Story” -

access the MovieID column & check where MovieID=1 mean of Ratio

---

In [107]:

```
#extract movie data for movie toy story
df_ts = data[data['MovieID'] == 1]

#View toy story first five records
df_ts.head()
#display (df_ts.shape)-(2077, 10)
```

Out[107]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-Code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008	F	50	9	55117
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496	M	25	12	11413
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952	M	25	17	61614
369	1	Toy Story (1995)	Animation|Children's|Co	10	5	978226474	F	35	1	95370

medy

In [108]:

```
df_ts['Rating'].mean()
```

Out[108]:

4.146846413095811

In [109]:

```
data.Rating[data['MovieID'] == 1].mean()
```

Out[109]:

4.146846413095811

Top 25 movies by viewership rating

table with Title & rating

In [110]:

```
#explore movie data for viewership by movie title
data_count = data['Title'].value_counts()
data_count[0:25]
```

Out[110]:

```
American Beauty (1999)                                   3428
Star Wars: Episode IV - A New Hope (1977)                2991
Star Wars: Episode V - The Empire Strikes Back (1980)    2990
Star Wars: Episode VI - Return of the Jedi (1983)        2883
Jurassic Park (1993)                                     2672
Saving Private Ryan (1998)                               2653
Terminator 2: Judgment Day (1991)                        2649
Matrix, The (1999)                                       2590
Back to the Future (1985)                                2583
Silence of the Lambs, The (1991)                         2578
Men in Black (1997)                                      2538
Raiders of the Lost Ark (1981)                           2514
Fargo (1996)                                             2513
Sixth Sense, The (1999)                                  2459
Braveheart (1995)                                        2443
Shakespeare in Love (1998)                               2369
Princess Bride, The (1987)                               2318
Schindler's List (1993)                                  2304
L.A. Confidential (1997)                                 2288
Groundhog Day (1993)                                     2278
E.T. the Extra-Terrestrial (1982)                        2269
Star Wars: Episode I - The Phantom Menace (1999)         2250
Being John Malkovich (1999)                              2241
Shawshank Redemption, The (1994)                         2227
Godfather, The (1972)                                    2223
Name: Title, dtype: int64
```

In [111]:

```
#explore movie data for viewership by movie title
titlewise_mean = pd.DataFrame(data.groupby('Title')['Rating'].mean())
display (titlewise_mean.head())
```

	Rating
Title	
$1,000,000 Duck (1971)	3.027027
'Night Mother (1986)	3.371429
'Til There Was You (1997)	2.692308
'burbs, The (1989)	2.910891
...And Justice for All (1979)	3.713568

In [112]:

```
titlewise_mean.sort_values('Rating',ascending=False).head(25)
top_25 = titlewise_mean.sort_values('Rating',ascending=False).head(25)
top_25
```

Out[112]:

	Rating
Title	
Ulysses (Ulisse) (1954)	5.000000
Lured (1947)	5.000000
Follow the Bitch (1998)	5.000000

Bittersweet Motel (2000)	5.000000
Song of Freedom (1936)	5.000000
One Little Indian (1973)	5.000000
Smashing Time (1967)	5.000000
Schlafes Bruder (Brother of Sleep) (1995)	5.000000
Gate of Heavenly Peace, The (1995)	5.000000
Baby, The (1973)	5.000000
I Am Cuba (Soy Cuba/Ya Kuba) (1964)	4.800000
Lamerica (1994)	4.750000
Apple, The (Sib) (1998)	4.666667
Sanjuro (1962)	4.608696
Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)	4.560510
Shawshank Redemption, The (1994)	4.554558
Godfather, The (1972)	4.524966
Close Shave, A (1995)	4.520548
Usual Suspects, The (1995)	4.517106
Schindler's List (1993)	4.510417
Wrong Trousers, The (1993)	4.507937
Dry Cleaning (Nettoyage � sec) (1997)	4.500000
Inheritors, The (Die Siebtelbauern) (1998)	4.500000
Mamma Roma (1962)	4.500000
Bells, The (1926)	4.500000

Find the ratings for all the movies reviewed by for a particular user of

user id = 2696

In [113]:

```
#View user records where UserID=2696
df_user = data[data['UserID'] == 2696]

df_user.head()
#display (df_ts.shape) #(2077, 10)
```

Out[113]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-Code
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	25	7	24210
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	25	7	24210
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	25	7	24210
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	25	7	24210
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M	25	7	24210

Feature Engineering: Use column genres:

1. Find out all the unique genres (Hint: split the data in column genre making a list and then process the data to find out only the unique categories of genres)

In [114]:

```
#df_genres = data['Genres']
#df_split = df_genres.split("|")
#df_genres
```

In [115]:

```
data.Genres.head()
```

Out[115]:

```
0              Animation|Children's|Comedy
1    Animation|Children's|Musical|Romance
2                                   Drama
3         Action|Adventure|Fantasy|Sci-Fi
4                               Drama|War
Name: Genres, dtype: object
```

In [116]:

```
data.Genres = data.Genres.str.split("|")
data.Genres[:3]
```

Out[116]:

```
0              [Animation, Children's, Comedy]
1    [Animation, Children's, Musical, Romance]
2                                      [Drama]
Name: Genres, dtype: object
```

In [117]:

```
data.shape
```

Out[117]:

```
(1000209, 10)
```

In [138]:

```
data5k = data[:5000]
data5k
```

Out[138]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-Code
0	1	Toy Story (1995)	[Animation, Children's, Comedy]	1	5	978824268	F	1	10	48067
1	48	Pocahontas (1995)	[Animation, Children's, Musical, Romance]	1	5	978824351	F	1	10	48067
2	150	Apollo 13 (1995)	[Drama]	1	5	978301777	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)	[Action, Adventure, Fantasy, Sci-Fi]	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	[Drama, War]	1	5	978824195	F	1	10	48067
5	531	Secret Garden, The (1993)	[Children's, Drama]	1	4	978302149	F	1	10	48067
6	588	Aladdin (1992)	[Animation,	1	4	978824268	F	1	10	48067

			Children's, Comedy, Musical]							
7	594	Snow White and the Seven Dwarfs (1937)	[Animation, Children's, Musical]	1	4	978302268	F	1	10	48067
8	595	Beauty and the Beast (1991)	[Animation, Children's, Musical]	1	5	978824268	F	1	10	48067
9	608	Fargo (1996)	[Crime, Drama, Thriller]	1	4	978301398	F	1	10	48067
10	661	James and the Giant Peach (1996)	[Animation, Children's, Musical]	1	3	978302109	F	1	10	48067
11	720	Wallace & Gromit: The Best of Aardman Animatio...	[Animation]	1	3	978300760	F	1	10	48067
12	745	Close Shave, A (1995)	[Animation, Comedy, Thriller]	1	3	978824268	F	1	10	48067
13	783	Hunchback of Notre Dame, The (1996)	[Animation, Children's, Musical]	1	4	978824291	F	1	10	48067
14	914	My Fair Lady (1964)	[Musical, Romance]	1	3	978301968	F	1	10	48067
15	919	Wizard of Oz, The (1939)	[Adventure, Children's, Drama, Musical]	1	4	978301368	F	1	10	48067

16	938	Gigi (1958)	[Musical]	1	4	978301752	F	1	10	48067
17	1022	Cinderella (1950)	[Animation, Children's, Musical]	1	5	978300055	F	1	10	48067
18	1028	Mary Poppins (1964)	[Children's, Comedy, Musical]	1	5	978301777	F	1	10	48067
19	1029	Dumbo (1941)	[Animation, Children's, Musical]	1	5	978302205	F	1	10	48067
20	1035	Sound of Music, The (1965)	[Musical]	1	5	978301753	F	1	10	48067
21	1097	E.T. the Extra-Terrestrial (1982)	[Children's, Drama, Fantasy, Sci-Fi]	1	4	978301953	F	1	10	48067
22	1193	One Flew Over the Cuckoo's Nest (1975)	[Drama]	1	5	978300760	F	1	10	48067
23	1197	Princess Bride, The (1987)	[Action, Adventure, Comedy, Romance]	1	3	978302268	F	1	10	48067
24	1207	To Kill a Mockingbird (1962)	[Drama]	1	4	978300719	F	1	10	48067
25	1246	Dead Poets Society (1989)	[Drama]	1	4	978302091	F	1	10	48067
26	1270	Back to the Future	[Comedy, Sci-	1	5	978300055	F	1	10	48067

		(1985)	Fi]							
27	1287	Ben-Hur (1959)	[Action, Adventure, Drama]	1	5	978302039	F	1	10	48067
28	1545	Ponette (1996)	[Drama]	1	4	978824139	F	1	10	48067
29	1566	Hercules (1997)	[Adventure, Animation, Children's, Comedy, Mus...	1	4	978824330	F	1	10	48067
...	...	...	...	...	...	...	...	...	...	...
4970	2499	God Said 'Ha!' (1998)	[Comedy]	78	5	978571083	F	45	1	98029
4971	2539	Analyze This (1999)	[Comedy]	78	1	978571371	F	45	1	98029
4972	2596	SLC Punk! (1998)	[Comedy, Drama]	78	4	978570873	F	45	1	98029
4973	2599	Election (1999)	[Comedy]	78	5	978570648	F	45	1	98029
4974	2622	Midsummer Night's Dream, A (1999)	[Comedy, Fantasy]	78	3	978571371	F	45	1	98029
4975	2671	Notting Hill (1999)	[Comedy, Romance]	78	3	978571281	F	45	1	98029
4976	2690	Ideal Husband, An (1999)	[Comedy]	78	3	978570974	F	45	1	98029
4977	2759	Dick (1999)	[Comedy]	78	3	978571281	F	45	1	98029
4978	2858	American Beauty	[Comedy, Drama]	78	5	978570648	F	45	1	98029

		(1999)								
4979	2971	All That Jazz (1979)	[Musical]	78	5	977811665	F	45	1	98029
4980	2997	Being John Malkovich (1999)	[Comedy]	78	2	978570648	F	45	1	98029
4981	3052	Dogma (1999)	[Comedy]	78	4	978570767	F	45	1	98029
4982	3060	Commitments, The (1991)	[Comedy, Drama]	78	4	978570974	F	45	1	98029
4983	3072	Moonstruck (1987)	[Comedy]	78	3	977811162	F	45	1	98029
4984	3114	Toy Story 2 (1999)	[Animation, Children's, Comedy]	78	4	978570648	F	45	1	98029
4985	3159	Fantasia 2000 (1999)	[Animation, Children's, Musical]	78	4	978570374	F	45	1	98029
4986	3174	Man on the Moon (1999)	[Comedy, Drama]	78	3	978570974	F	45	1	98029
4987	3175	Galaxy Quest (1999)	[Adventure, Comedy, Sci-Fi]	78	4	978571083	F	45	1	98029
4988	3178	Hurricane, The (1999)	[Drama]	78	4	978570873	F	45	1	98029
4989	3247	Sister Act (1992)	[Comedy, Crime]	78	4	978571371	F	45	1	98029
4990	3253	Wayne's World (1992)	[Comedy]	78	3	978571175	F	45	1	98029
4991	3255	League	[Comed	78	5	9785708	F	45	1	98029

		of Their Own, A (1992)	y, Drama]			73				
4992	3282	Different for Girls (1996)	[Comedy]	78	4	978571175	F	45	1	98029
4993	3358	Defending Your Life (1991)	[Comedy, Romance]	78	4	978570873	F	45	1	98029
4994	3545	Cabaret (1972)	[Musical, War]	78	5	978570374	F	45	1	98029
4995	3549	Guys and Dolls (1955)	[Musical]	78	4	977811666	F	45	1	98029
4996	3599	Anchors Aweigh (1945)	[Comedy, Musical]	78	3	977811666	F	45	1	98029
4997	3600	Blue Hawaii (1961)	[Comedy, Musical]	78	3	977811982	F	45	1	98029
4998	3606	On the Town (1949)	[Musical]	78	5	977811666	F	45	1	98029
4999	3614	Honeymoon in Vegas (1992)	[Comedy, Romance]	78	3	978571281	F	45	1	98029

5000 rows × 10 columns

In [139]:

```
x = []
for rn in range(len(data5k)):
  x = x + data5k.Genres[rn]

data5k.Genres
```

Out[139]:

```
0                    [Animation, Children's, Comedy]
1           [Animation, Children's, Musical, Romance]
2                                            [Drama]
3               [Action, Adventure, Fantasy, Sci-Fi]
4                                       [Drama, War]
5                                 [Children's, Drama]
6           [Animation, Children's, Comedy, Musical]
7                   [Animation, Children's, Musical]
8                   [Animation, Children's, Musical]
9                           [Crime, Drama, Thriller]
10                  [Animation, Children's, Musical]
11                                       [Animation]
12                    [Animation, Comedy, Thriller]
13                  [Animation, Children's, Musical]
14                               [Musical, Romance]
15          [Adventure, Children's, Drama, Musical]
16                                         [Musical]
17                  [Animation, Children's, Musical]
18                    [Children's, Comedy, Musical]
19                  [Animation, Children's, Musical]
20                                         [Musical]
21             [Children's, Drama, Fantasy, Sci-Fi]
22                                           [Drama]
23             [Action, Adventure, Comedy, Romance]
24                                           [Drama]
25                                           [Drama]
26                                  [Comedy, Sci-Fi]
27                       [Action, Adventure, Drama]
28                                           [Drama]
29     [Adventure, Animation, Children's, Comedy, Mus...
                          ...
4970                                        [Comedy]
4971                                        [Comedy]
4972                                 [Comedy, Drama]
4973                                        [Comedy]
4974                               [Comedy, Fantasy]
4975                               [Comedy, Romance]
4976                                        [Comedy]
4977                                        [Comedy]
4978                                 [Comedy, Drama]
4979                                       [Musical]
4980                                        [Comedy]
4981                                        [Comedy]
4982                                 [Comedy, Drama]
4983                                        [Comedy]
4984                 [Animation, Children's, Comedy]
```

```
4985                   [Animation, Children's, Musical]
4986                                    [Comedy, Drama]
4987                        [Adventure, Comedy, Sci-Fi]
4988                                            [Drama]
4989                                    [Comedy, Crime]
4990                                           [Comedy]
4991                                    [Comedy, Drama]
4992                                           [Comedy]
4993                                  [Comedy, Romance]
4994                                     [Musical, War]
4995                                          [Musical]
4996                                  [Comedy, Musical]
4997                                  [Comedy, Musical]
4998                                          [Musical]
4999                                  [Comedy, Romance]
Name: Genres, Length: 5000, dtype: object
```

In [140]:

```
unique_genres = list(set(x))
print (unique_genres)
```

```
['Crime', 'Drama', 'Adventure', 'Horror', 'Mystery', 'Documentary', 'Western', 'Animation', 'Comedy', 'Sci-Fi',
"Children's", 'Fantasy', 'Romance', 'Action', 'War', 'Film-Noir', 'Musical', 'Thriller']
```

1. Create a separate column for each genre category with a one-hot encoding (1 and 0) whether or not the movie belongs to that genre.

In [141]:

```
unique_genres = pd.Series(unique_genres)
```

In [142]:

```
df = pd.DataFrame()
for row in data5k.Genres:
    a = unique_genres.isin(row)
    df = df.append(a,ignore_index = True)
df[:5]
```

Out[142]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

In [143]:

```
df.columns = unique_genres
df.head()
```

Out[143]:

	Crime	Drama	Adventure	Horror	Mystery	Documentary	Western	Animation	Comedy	Sci-Fi	Children's	Fantasy	Romance	Action	War	Film-Noir	Musical	Thriller
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

In [144]:

```
data5k.head()
```

Out[144]:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-Code
0	1	Toy Story (1995)	[Animation, Children's, Comedy]	1	5	978824268	F	1	10	48067
1	48	Pocahontas (1995)	[Animation, Children's, Musical, Romance]	1	5	978824351	F	1	10	48067

2	150	Apollo 13 (1995)	[Drama]	1	5	978301777	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)	[Action, Adventure, Fantasy, Sci-Fi]	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	[Drama, War]	1	5	978824195	F	1	10	48067

In [147]:

```
#data5k = pd.concat((data5k, df), axis = 1)
data5k = pd.concat((data5k.Title, df), axis = 1)
data5k.head()
```

Out[147]:

	Title	Crime	Drama	Adventure	Horror	Mystery	Documentary	Western	Animation	Comedy	Sci-Fi	Children's	Fantasy	Romance	Action	War	Film-Noir	Musical	Thriller
0	Toy Story (1995)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	Pocahontas (1995)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
2	Apoll	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	o 13 (1995)																		
3	Star Wars: Episode IV - A New Hope (1977)	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
4	Schindler's List (1993)	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

1. Determine the features affecting the ratings of any particular movie.

In [160]:

```
#correlation - best
#hypothesis testing
import seaborn as sns
%matplotlib inline
```

In [161]:

```
plt.figure(figsize = (20,8))
```

```
corr = data5k.corr()
sns.heatmap(corr, xticklabels=corr.columns.values, yticklabels=corr.columns.values, annot = True,
annot_kws={'size':10})
```

Out[161]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff48622b6d8>
```

In [175]:

```
# create a Python list of feature names
feature_cols=['Age','Occupation']
# use the list to select a subset of the original DataFrame
X=data[feature_cols]
# select a Series from the DataFrame
y=data.Rating
```

1. Develop an appropriate model to predict the movie ratings

In [180]:

```
from sklearn.model_selection import train_test_split
# split into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1)
# import model
from sklearn.linear_model import LinearRegression
# instantiate
linreg=LinearRegression()
# fit the model to the training data (learn the coefficients)
linreg.fit(X_train,y_train)
```

```python
# make predictions on the testing set
y_pred=linreg.predict(X_test)
from sklearn.metrics import mean_squared_error
# compute the RMSE of our predictions
print(np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
1.1153284258531615
```

In [ ]: