In [93]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
movies= pd.read_csv(r'S:\DOCS\4th sep  imdb\IMDB_dataset\movie.csv')
```

In [4]:
```
movies
```

Out[4]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
27273	131254	Kein Bund für's Leben (2007)	Comedy
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy
27275	131258	The Pirates (2014)	Adventure
27276	131260	Rentun Ruusu (2001)	(no genres listed)
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror

27278 rows × 3 columns

In [5]:
```
ratings=pd.read_csv(r'S:\DOCS\4th sep  imdb\IMDB_dataset\rating.csv')
```

In [7]:
```
ratings
```

Out[7]:

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47
1	1	29	3.5	2005-04-02 23:31:16
2	1	32	3.5	2005-04-02 23:33:39
3	1	47	3.5	2005-04-02 23:32:07
4	1	50	3.5	2005-04-02 23:29:40
...	...	...	...	...
20000258	138493	68954	4.5	2009-11-13 15:42:00
20000259	138493	69526	4.5	2009-12-03 18:31:48
20000260	138493	69644	3.0	2009-12-07 18:10:57
20000261	138493	70286	5.0	2009-11-13 15:42:24
20000262	138493	71619	2.5	2009-10-17 20:25:36

20000263 rows × 4 columns

In [10]:
```
tags=pd.read_csv(r'S:\DOCS\4th sep  imdb\IMDB_dataset\tag.csv')
```

In [11]:
```
tags
```

Out[11]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
1	65	208	dark hero	2013-05-10 01:41:18
2	65	353	dark hero	2013-05-10 01:41:19
3	65	521	noir thriller	2013-05-10 01:39:43
4	65	592	dark hero	2013-05-10 01:41:18
...	...	...	...	...
465559	138446	55999	dragged	2013-01-23 23:29:32
465560	138446	55999	Jason Bateman	2013-01-23 23:29:38
465561	138446	55999	quirky	2013-01-23 23:29:38
465562	138446	55999	sad	2013-01-23 23:29:32
465563	138472	923	rise to power	2007-11-02 21:12:47

465564 rows × 4 columns

In [12]:
```
movies
```

Out[12]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
27273	131254	Kein Bund für's Leben (2007)	Comedy
27274	131256	Feuer, Eis & Dosenbier (2002)	Comedy
27275	131258	The Pirates (2014)	Adventure
27276	131260	Rentun Ruusu (2001)	(no genres listed)
27277	131262	Innocence (2014)	Adventure|Fantasy|Horror

27278 rows × 3 columns

In [13]:
```
type(movies)
```

Out[13]:
```
pandas.core.frame.DataFrame
```

In [14]:
```
movies.head(20)
```

Out[14]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
10	11	American President, The (1995)	Comedy|Drama|Romance
11	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
12	13	Balto (1995)	Adventure|Animation|Children
13	14	Nixon (1995)	Drama
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
15	16	Casino (1995)	Crime|Drama
16	17	Sense and Sensibility (1995)	Drama|Romance
17	18	Four Rooms (1995)	Comedy
18	19	Ace Ventura: When Nature Calls (1995)	Comedy
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller

In [16]:
```
ratings
```

Out[16]:

	userId	movieId	rating	timestamp
0	1	2	3.5	2005-04-02 23:53:47
1	1	29	3.5	2005-04-02 23:31:16
2	1	32	3.5	2005-04-02 23:33:39
3	1	47	3.5	2005-04-02 23:32:07
4	1	50	3.5	2005-04-02 23:29:40
...	...	...	...	...
20000258	138493	68954	4.5	2009-11-13 15:42:00
20000259	138493	69526	4.5	2009-12-03 18:31:48
20000260	138493	69644	3.0	2009-12-07 18:10:57
20000261	138493	70286	5.0	2009-11-13 15:42:24
20000262	138493	71619	2.5	2009-10-17 20:25:36

20000263 rows × 4 columns

In [30]:
```
ratings
```

Out[30]:

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5
...	...	...	...
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

20000263 rows × 3 columns

In [ ]:
```
ratings= ratings.drop('timestamp', axis= 1)
```

In [31]:
```
tags
```

Out[31]:

	userId	movieId	tag	timestamp
0	18	4141	Mark Waters	2009-04-24 18:19:40
1	65	208	dark hero	2013-05-10 01:41:18
2	65	353	dark hero	2013-05-10 01:41:19
3	65	521	noir thriller	2013-05-10 01:39:43
4	65	592	dark hero	2013-05-10 01:41:18
...	...	...	...	...
465559	138446	55999	dragged	2013-01-23 23:29:32
465560	138446	55999	Jason Bateman	2013-01-23 23:29:38
465561	138446	55999	quirky	2013-01-23 23:29:38
465562	138446	55999	sad	2013-01-23 23:29:32
465563	138472	923	rise to power	2007-11-02 21:12:47

465564 rows × 4 columns

In [32]:
```
del tags['timestamp']
```

In [33]:
```
tags
```

Out[33]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero
2	65	353	dark hero
3	65	521	noir thriller
4	65	592	dark hero
...	...	...	...
465559	138446	55999	dragged
465560	138446	55999	Jason Bateman
465561	138446	55999	quirky
465562	138446	55999	sad
465563	138472	923	rise to power

465564 rows × 3 columns

In [34]:
```
row_0 = tags.iloc[0]
#will return the first row, iloc function is used to select rows and columns by
```

In [35]:
```
type(row_0)
```

Out[35]:
```
pandas.core.series.Series
```

In [36]:
```
row_0
```

Out[36]:
```
userId              18
movieId           4141
tag        Mark Waters
Name: 0, dtype: object
```

In [39]:
```
row_0['userId']
```

Out[39]:
```
18
```

In [40]:
```
'tag' in row_0
```

Out[40]:
```
True
```

In [41]:
```
tags.head()
```

Out[41]:

	userId	movieId	tag
0	18	4141	Mark Waters
1	65	208	dark hero
2	65	353	dark hero
3	65	521	noir thriller
4	65	592	dark hero

In [43]:
```
tags.index
```

Out[43]:
```
RangeIndex(start=0, stop=465564, step=1)
```

In [44]:
```
tags.columns
```

Out[44]:
```
Index(['userId', 'movieId', 'tag'], dtype='object')
```

In [46]:
```
tags.iloc[[0,11,500]] # we are selecting the rows at positions 0, 11, and 500
```

Out[46]:

	userId	movieId	tag
0	18	4141	Mark Waters
11	65	1783	noir thriller
500	342	55908	entirely dialogue

descriptive Statistics

In [48]:
```
ratings
```

Out[48]:

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5
...	...	...	...
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

20000263 rows × 3 columns

In [49]:
```
ratings['rating'].describe()
```

Out[49]:
```
count    2.000026e+07
mean     3.525529e+00
std      1.051989e+00
min      5.000000e-01
25%      3.000000e+00
50%      3.500000e+00
75%      4.000000e+00
max      5.000000e+00
Name: rating, dtype: float64
```

In [50]:
```
ratings.describe()
```

Out[50]:

	userId	movieId	rating
count	2.000026e+07	2.000026e+07	2.000026e+07
mean	6.904587e+04	9.041567e+03	3.525529e+00
std	4.003863e+04	1.978948e+04	1.051989e+00
min	1.000000e+00	1.000000e+00	5.000000e-01
25%	3.439500e+04	9.020000e+02	3.000000e+00
50%	6.914100e+04	2.167000e+03	3.500000e+00
75%	1.036370e+05	4.770000e+03	4.000000e+00
max	1.384930e+05	1.312620e+05	5.000000e+00

In [52]:
```
ratings['rating'].mean()
```

Out[52]:
```
3.5255285642993797
```

In [53]:
```
ratings.mean()
```

Out[53]:
```
userId     69045.872583
movieId     9041.567330
rating         3.525529
dtype: float64
```

In [55]:
```
ratings['rating'].min()
```

Out[55]:
```
0.5
```

In [56]:
```
ratings['rating'].max()
```

Out[56]:
```
5.0
```

In [57]:
```
ratings['rating'].std()  # The standard deviation of the ratings is 1.
```

Out[57]:
```
1.051988919275684
```

In [58]:
```
ratings['rating'].mode()  # This means that 4 is the most frequent rating
```

Out[58]:
```
0    4.0
Name: rating, dtype: float64
```

In [60]:
```
ratings.corr()  # correlation
```

Out[60]:

	userId	movieId	rating
userId	1.000000	-0.000850	0.001175
movieId	-0.000850	1.000000	0.002606
rating	0.001175	0.002606	1.000000

In [61]:
```
filter1=ratings['rating']>10
```

In [66]:
```
filter1
```

Out[66]:
```
0           False
1           False
2           False
3           False
4           False
            ...
20000258    False
20000259    False
20000260    False
20000261    False
20000262    False
Name: rating, Length: 20000263, dtype: bool
```

In [67]:
```
filter2= ratings['rating']>0
```

In [68]:
```
filter2
```

Out[68]:
```
0           True
1           True
2           True
3           True
4           True
            ...
20000258    True
20000259    True
20000260    True
20000261    True
20000262    True
Name: rating, Length: 20000263, dtype: bool
```

In [69]:
```
filter2.all()
```

Out[69]:
```
True
```

data cleaning

In [71]:
```
movies.shape
```

Out[71]:
```
(27278, 3)
```

In [73]:
```
movies.isnull().any().any()
```

Out[73]:
```
False
```

In [74]:
```
# no null value !
```

In [75]:
```
ratings.shape
```

Out[75]:
```
(20000263, 3)
```

In [77]:
```
ratings.isnull().any().any()
```

Out[77]:
```
False
```

In [78]:
```
# no null value in ratings !
```

In [79]:
```
tags.shape
```

Out[79]: (465564, 3)

In [81]:
```
tags.isnull().any().any()
```

Out[81]: True

In [82]:
```
# now we have null value in tags
```

In [83]:
```
tags=tags.dropna() #will drop all rows in the DataFrame, that have any missing v
```

In [84]:
```
tags.shape     # notice no. of rows have reduced !
```

Out[84]: (465548, 3)

In [85]:
```
tags.isnull().any().any()
```

Out[85]: False

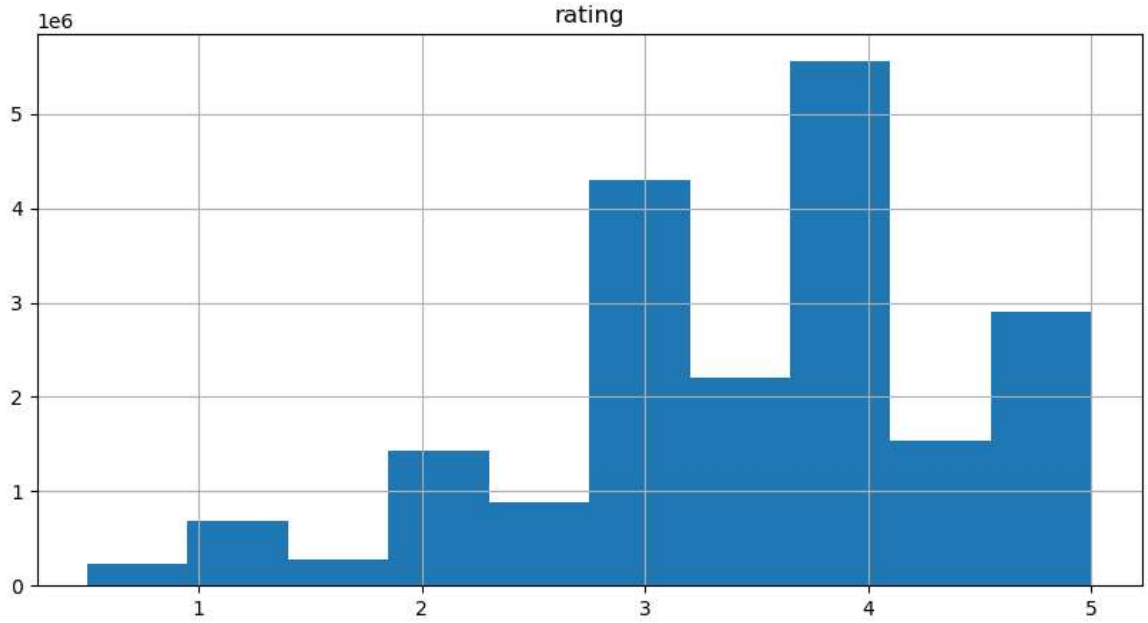
In [86]:
```
# now all the missing values is removed
```

data visualisation

In [90]:
```
%matplotlib inline
```

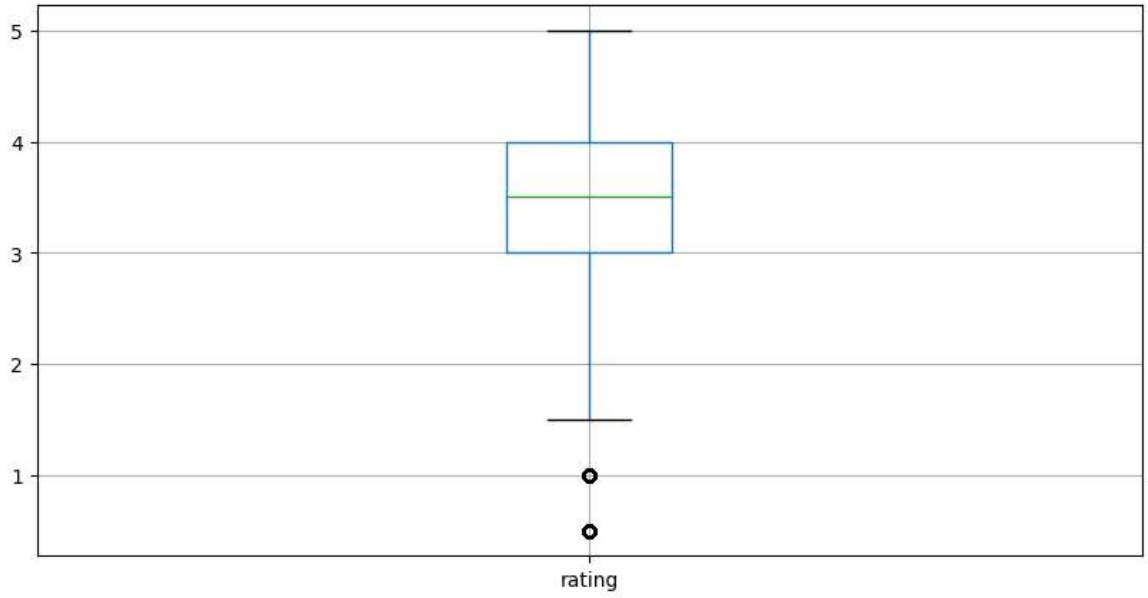
In [91]:
```
ratings.hist(column='rating', figsize=(10,5))
```

Out[91]: array([[<Axes: title={'center': 'rating'}>]], dtype=object)



In [92]:
```
ratings.boxplot(column='rating', figsize=(10,5))
```

Out[92]: <Axes: >



In [95]:
```
tags['tag'].head()
```

Out[95]:
```
0      Mark Waters
1        dark hero
2        dark hero
3    noir thriller
4        dark hero
Name: tag, dtype: object
```

In [98]:
```
movies[['title','genres']].head()
```

Out[98]:

	title	genres
0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	Jumanji (1995)	Adventure|Children|Fantasy
2	Grumpier Old Men (1995)	Comedy|Romance
3	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	Father of the Bride Part II (1995)	Comedy

In [100]:
```
ratings[-10:]
```

Out[100]:

	userId	movieId	rating
20000253	138493	60816	4.5
20000254	138493	61160	4.0
20000255	138493	65682	4.5
20000256	138493	66762	4.5
20000257	138493	68319	4.5
20000258	138493	68954	4.5
20000259	138493	69526	4.5
20000260	138493	69644	3.0
20000261	138493	70286	5.0
20000262	138493	71619	2.5

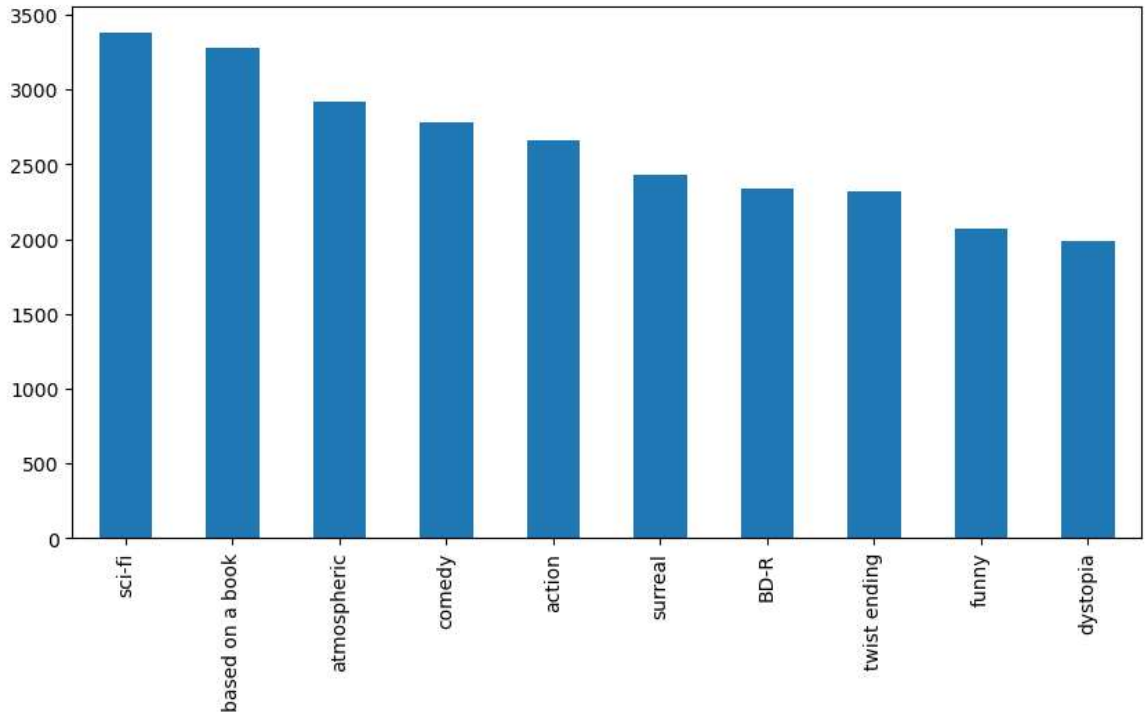
In [101]:
```
tags_counts=tags['tag'].value_counts()
```

In [104]:
```
tags_counts[10:]
```

Out[104]:
```
stylized                          1941
quirky                            1906
dark comedy                       1899
classic                           1769
psychology                        1754
                                  ...
Paul Adelstein                       1
the wig                              1
killer fish                          1
genetically modified monsters        1
topless scene                        1
Name: tag, Length: 38633, dtype: int64
```

In [109]:
```
tags_counts[:10].plot(kind='bar',figsize=(10,5))
```

Out[109]: <Axes: >



filters for slelecting rows

In [114]:
```
highly_rated=ratings['rating'] >= 5.0
```

In [116]:
```
ratings[highly_rated][30:50]
```

Out[116]:

	userId	movieId	rating
239	3	50	5.0
242	3	175	5.0
244	3	223	5.0
245	3	260	5.0
246	3	316	5.0
247	3	318	5.0
248	3	329	5.0
252	3	457	5.0
253	3	480	5.0
254	3	490	5.0
256	3	541	5.0
258	3	593	5.0
263	3	858	5.0
264	3	904	5.0
267	3	924	5.0
268	3	953	5.0
271	3	1060	5.0
272	3	1073	5.0
275	3	1084	5.0
276	3	1089	5.0

In [117]:
```
is_action=movies['genres'].str.contains("Action")
```

In [119]:
```
movies[is_action][5:20]
```

Out[119]:

	movieId	title	genres
22	23	Assassins (1995)	Action|Crime|Thriller
41	42	Dead Presidents (1995)	Action|Crime|Drama
43	44	Mortal Kombat (1995)	Action|Adventure|Fantasy
50	51	Guardian Angel (1994)	Action|Drama|Thriller
65	66	Lawnmower Man 2: Beyond Cyberspace (1996)	Action|Sci-Fi|Thriller
69	70	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
70	71	Fair Game (1995)	Action
75	76	Screamers (1995)	Action|Sci-Fi|Thriller
77	78	Crossing Guard, The (1995)	Action|Crime|Drama|Thriller
85	86	White Squall (1996)	Action|Adventure|Drama
88	89	Nick of Time (1995)	Action|Thriller
93	95	Broken Arrow (1996)	Action|Adventure|Thriller
96	98	Shopping (1994)	Action|Thriller
108	110	Braveheart (1995)	Action|Drama|War
110	112	Rumble in the Bronx (Hont faan kui) (1995)	Action|Adventure|Comedy|Crime

In [120]:
```
movies[is_action].head(15)
```

Out[120]:

	movieId	title	genres
5	6	Heat (1995)	Action|Crime|Thriller
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
19	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller
22	23	Assassins (1995)	Action|Crime|Thriller
41	42	Dead Presidents (1995)	Action|Crime|Drama
43	44	Mortal Kombat (1995)	Action|Adventure|Fantasy
50	51	Guardian Angel (1994)	Action|Drama|Thriller
65	66	Lawnmower Man 2: Beyond Cyberspace (1996)	Action|Sci-Fi|Thriller
69	70	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
70	71	Fair Game (1995)	Action
75	76	Screamers (1995)	Action|Sci-Fi|Thriller
77	78	Crossing Guard, The (1995)	Action|Crime|Drama|Thriller
85	86	White Squall (1996)	Action|Adventure|Drama

In [ ]:
```

```