In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import collections
import seaborn as sns
%matplotlib inline
```

In [2]:

```
ratings_data = pd.read_csv('ratings.csv')
movies_df = pd.read_csv("movies.csv")
tags = pd.read_csv('tags.csv')
scores = pd.read_csv('genome-scores.csv')
links = pd.read_csv('links.csv')
genome_tags = pd.read_csv('genome-tags.csv')
```

In [3]:

```
movies_df.head()
```

Out[3]:

	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

In [40]:

```
movies_df.shape
```

Out[40]:

```
(27278, 3)
```

In [42]:

```
movies_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27278 entries, 0 to 27277
Data columns (total 3 columns):
movieId    27278 non-null int64
title      27278 non-null object
genres     27278 non-null object
dtypes: int64(1), object(2)
memory usage: 639.5+ KB
```

In [44]:

```
movies_df.isnull().sum()
```

Out[44]:

```
movieId    0
title      0
genres     0
dtype: int64
```

In [41]:

```
movies_df.describe()
```

Out[41]:

	movieId
count	27278.000000
mean	59855.480570
std	44429.314697
min	1.000000
25%	6931.250000
50%	68068.000000
75%	100293.250000
max	131262.000000

In [4]:

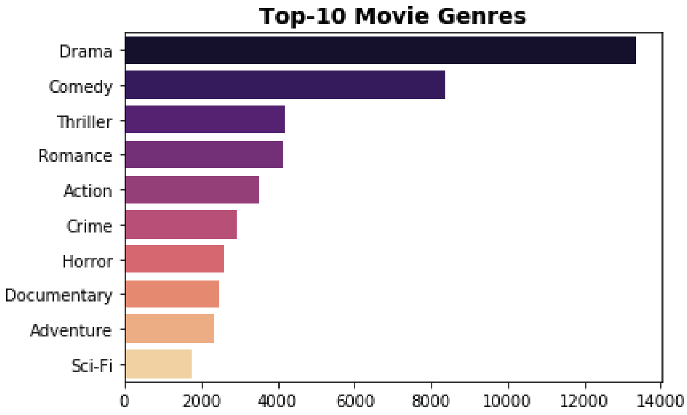
```
movie_genres = []

for genre in movies_df['genres']:
    for movie in genre.split('|'):
        movie_genres.append(movie)
```

In [5]:

```
genre_counts = pd.Series(movie_genres).value_counts()[:18]
```

In [6]:

```
top_genres = pd.Series(movie_genres).value_counts()[:10]

sns.barplot(y=top_genres.index, x=top_genres.values, palette='magma').set_title(
        'Top-10 Movie Genres', fontsize=14, weight='bold')

plt.show()
```



Drama is the most commonly occurring genre with almost half the movies identifying itself as a drama film. Comedy comes in at a distant second with 30% of the movies having adequate doses of humor. Other major genres represented in the top 10 are Thriller,Romance, Action, Crime, Horror, Documentry, Adventure, Scifi.

We will consider only those themes that appear in the top 10 most popular genres.

In [7]:

```
ratings_data.head()
```

Out[7]:

Out[7]:

	userId	movieId	rating	timestamp
0	1	2	3.5	1112486027
1	1	29	3.5	1112484676
2	1	32	3.5	1112484819
3	1	47	3.5	1112484727
4	1	50	3.5	1112484580

In [8]:

```
ratings_data['timestamp'] = pd.to_datetime(ratings_data['timestamp'], unit='s')
ratings_data.head()
```

Out[8]:

	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

In [9]:

```
ratings_data.shape
```

Out[9]:

```
(20000263, 4)
```

In [10]:

```
ratings_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000263 entries, 0 to 20000262
Data columns (total 4 columns):
userId       int64
movieId      int64
rating       float64
timestamp    datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(2)
memory usage: 610.4 MB
```

In [11]:

```
ratings_data.describe()
```

Out[11]:

	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

max	1.384930e+05	1.312620e+05	5.000000e+00
	userId	**movieId**	**rating**

In [12]:

```
ratings_data.corr()
```

Out[12]:

	userId	movieId	rating
userId	1.000000	-0.000638	0.001175
movieId	-0.000638	1.000000	0.001191
rating	0.001175	0.001191	1.000000

In [13]:

```
ratings_data['gave_rating_year'] = ratings_data['timestamp'].dt.year
ratings_data['gave_rating_month'] = ratings_data['timestamp'].dt.month_name().str[:3]

ratings_data.drop('timestamp', axis=1, inplace=True)
```

In [14]:

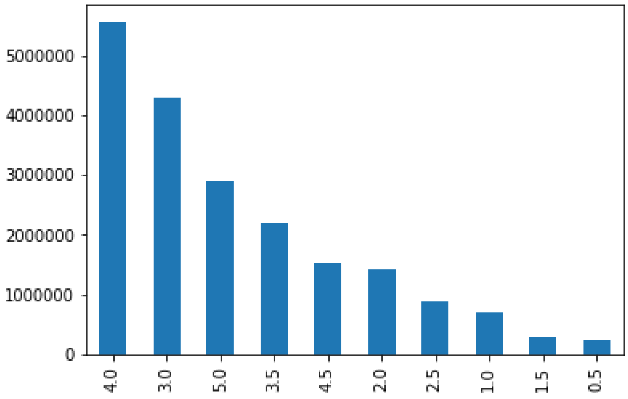
```
ratings_data.head()
```

Out[14]:

	userId	movieId	rating	gave_rating_year	gave_rating_month
0	1	2	3.5	2005	Apr
1	1	29	3.5	2005	Apr
2	1	32	3.5	2005	Apr
3	1	47	3.5	2005	Apr
4	1	50	3.5	2005	Apr

In [15]:

```
ratings_data['rating'].value_counts().plot(kind='bar');
```



In [16]:

```
ratings_data.rating.value_counts(normalize=True)
```

Out[16]:

```
4.0     0.278093
3.0     0.214557
5.0     0.144931
3.5     0.110006
4.5     0.076740
2.0     0.071540
```

```
2.0     0.071549
2.5     0.044169
1.0     0.034036
1.5     0.013962
0.5     0.011956
Name: rating, dtype: float64
```

In [45]:

```
ratings_data.isnull().sum()
```

Out[45]:

```
userId               0
movieId              0
rating               0
gave_rating_year     0
gave_rating_month    0
dtype: int64
```

In [17]:

```
movies_with_ratings = ratings_data.merge(movies_df, on='movieId', how='inner')
movies_with_ratings.head()
```

Out[17]:

	userId	movieId	rating	gave_rating_year	gave_rating_month	title	genres
0	1	2	3.5	2005	Apr	Jumanji (1995)	Adventure|Children|Fantasy
1	5	2	3.0	1996	Dec	Jumanji (1995)	Adventure|Children|Fantasy
2	13	2	3.0	1996	Nov	Jumanji (1995)	Adventure|Children|Fantasy
3	29	2	3.0	1996	Jun	Jumanji (1995)	Adventure|Children|Fantasy
4	34	2	3.0	1996	Oct	Jumanji (1995)	Adventure|Children|Fantasy

In [18]:

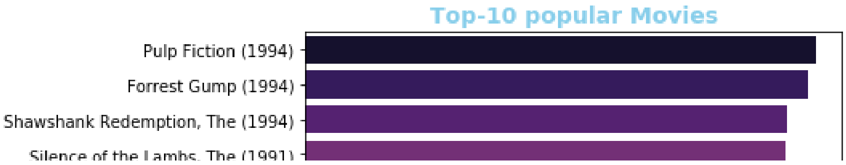
```
counts = movies_with_ratings['title'].value_counts()[:10]
counts
```

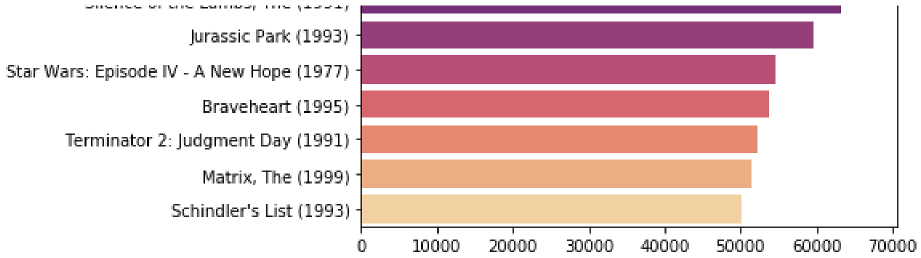
Out[18]:

```
Pulp Fiction (1994)                          67310
Forrest Gump (1994)                          66172
Shawshank Redemption, The (1994)             63366
Silence of the Lambs, The (1991)             63299
Jurassic Park (1993)                         59715
Star Wars: Episode IV - A New Hope (1977)    54502
Braveheart (1995)                            53769
Terminator 2: Judgment Day (1991)            52244
Matrix, The (1999)                           51334
Schindler's List (1993)                      50054
Name: title, dtype: int64
```

In [19]:

```
top_genres = pd.Series(counts)

sns.barplot(y=top_genres.index, x=top_genres.values, palette='magma').set_title(
        'Top-10 popular Movies', fontsize=14, weight='bold', color = 'skyblue')

plt.show()
```



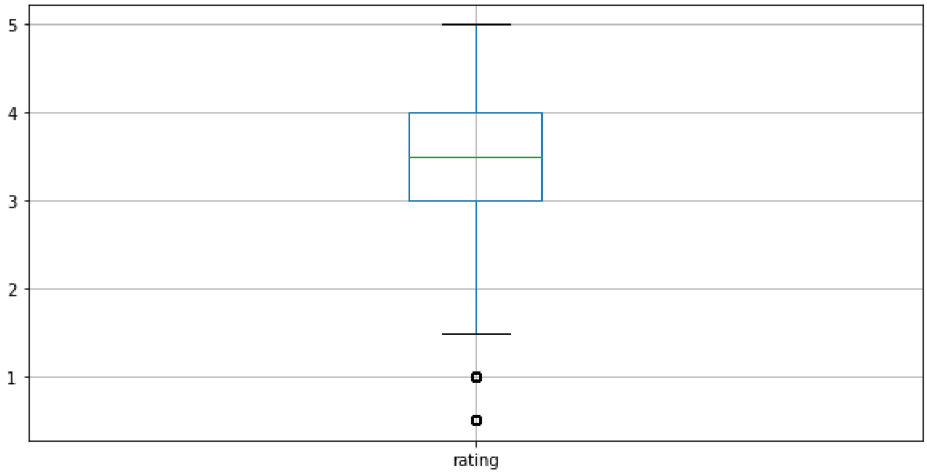


In [20]:

```
ratings_data.boxplot(column='rating', figsize=(10,5))
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1eeb92c1630>
```



Basic Statistics (#Ratings, #Users, and #Movies)

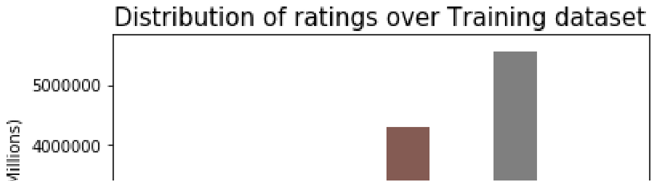
In [21]:

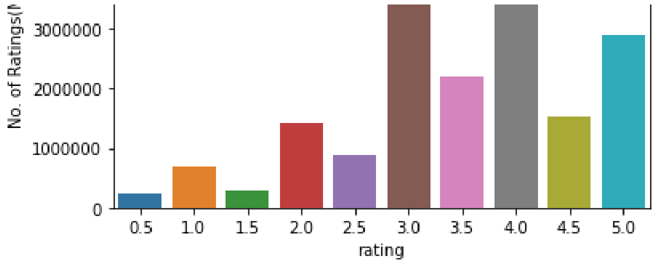
```
print("Total data ")
print("-"*50)
print("\nTotal no of ratings :", ratings_data.shape[0])
print("Total No of Users   :", len(np.unique(ratings_data.userId)))
print("Total No of movies  :", len(np.unique(ratings_data.movieId)))
```

```
Total data
--------------------------------------------------

Total no of ratings : 20000263
Total No of Users   : 138493
Total No of movies  : 26744
```

In [22]:

```
fig, ax = plt.subplots()
plt.title('Distribution of ratings over Training dataset', fontsize=15)
sns.countplot(ratings_data.rating)
#ax.set_yticklabels([human(item, 'M') for item in ax.get_yticks()])
ax.set_ylabel('No. of Ratings(Millions)')

plt.show()
```





In [23]:

```
no_of_rated_movies_per_user = ratings_data.groupby(by='userId')
['rating'].count().sort_values(ascending=False)

no_of_rated_movies_per_user.head()
```

Out[23]:

```
userId
118205    9254
8405      7515
82418     5646
121535    5520
125794    5491
Name: rating, dtype: int64
```

In [24]:

```
no_of_rated_movies_per_user.describe()
```

Out[24]:

```
count    138493.000000
mean        144.413530
std         230.267257
min          20.000000
25%          35.000000
50%          68.000000
75%         155.000000
max        9254.000000
Name: rating, dtype: float64
```

In [25]:

```
no_of_rated_movies_per_user.min()
```

Out[25]:

```
20
```

In [26]:

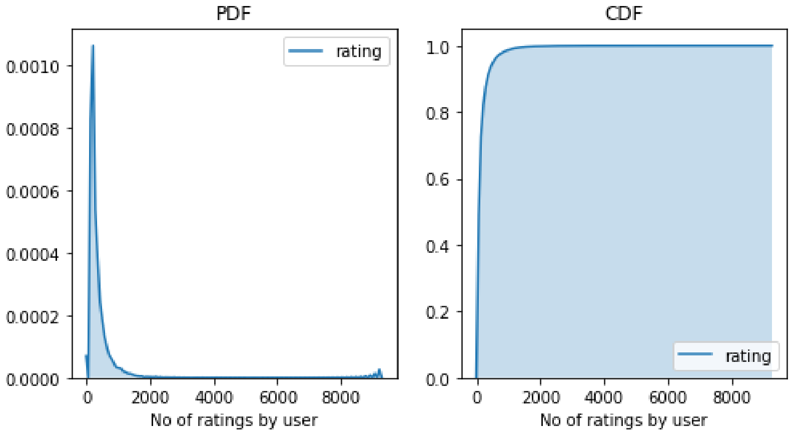
```
no_of_rated_movies_per_user.max()
```

Out[26]:

```
9254
```

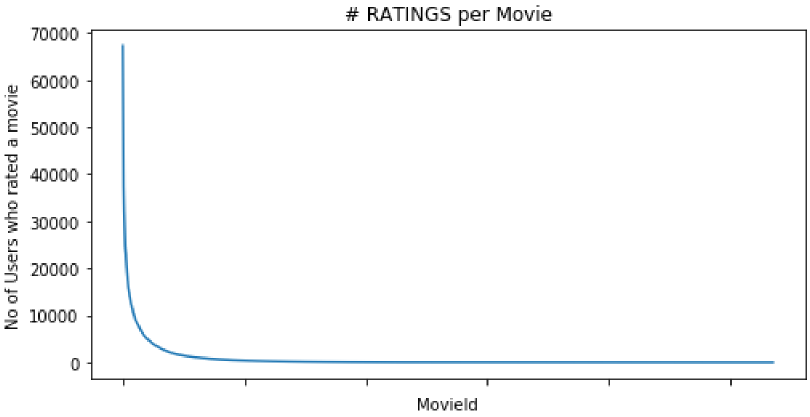
In [27]:

```
fig = plt.figure(figsize=plt.figaspect(.5))

ax1 = plt.subplot(121)
sns.kdeplot(no_of_rated_movies_per_user, shade=True, ax=ax1)
plt.xlabel('No of ratings by user')
plt.title("PDF")

ax2 = plt.subplot(122)
sns.kdeplot(no_of_rated_movies_per_user, shade=True, cumulative=True,ax=ax2)
plt.xlabel('No of ratings by user')
plt.xlabel('No of ratings by user')
plt.title('CDF')

plt.show()
```



In [28]:

```
no_of_ratings_per_movie = ratings_data.groupby(by='movieId')
['rating'].count().sort_values(ascending=False)

fig = plt.figure(figsize=plt.figaspect(.5))
ax = plt.gca()
plt.plot(no_of_ratings_per_movie.values)
plt.title('# RATINGS per Movie')
plt.xlabel('MovieId')
plt.ylabel('No of Users who rated a movie')
ax.set_xticklabels([])

plt.show()
```



It is very skewed.. just like nunmber of ratings given per user.

- There are some movies (which are very popular) which are rated by huge number of users.
- But most of the movies(like 90%) got some hundereds of ratings.

In [29]:

```
no_of_rated_movies_per_user.describe()
```

Out[29]:

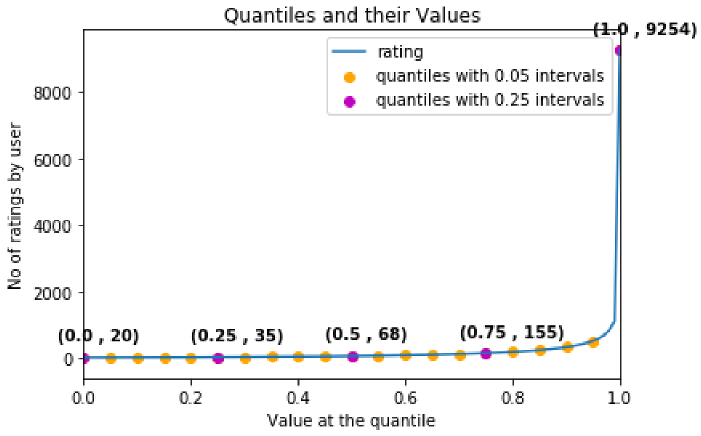
```
count    138493.000000
mean        144.413530
std         230.267257
min          20.000000
25%          35.000000
50%          68.000000
75%         155.000000
max        9254.000000
max         9254.000000
Name: rating, dtype: float64
```

In [30]:

```
quantiles = no_of_rated_movies_per_user.quantile(np.arange(0,1.01,0.01), interpolation='higher')
```

In [31]:

```
plt.title("Quantiles and their Values")
quantiles.plot()
# quantiles with 0.05 difference
plt.scatter(x=quantiles.index[::5], y=quantiles.values[::5], c='orange', label="quantiles with 0.05
intervals")
# quantiles with 0.25 difference
plt.scatter(x=quantiles.index[::25], y=quantiles.values[::25], c='m', label = "quantiles with 0.25
intervals")
plt.ylabel('No of ratings by user')
plt.xlabel('Value at the quantile')
plt.legend(loc='best')

# annotate the 25th, 50th, 75th and 100th percentile values....
for x,y in zip(quantiles.index[::25], quantiles[::25]):
    plt.annotate(s="({} , {})".format(x,y), xy=(x,y), xytext=(x-0.05, y+500)
                ,fontweight='bold')


plt.show()
```



In [32]:

```
quantiles[::5]
```

Out[32]:

```
0.00      20
0.05      21
0.10      24
0.15      27
0.20      30
0.25      35
0.30      39
0.35      45
0.40      51
0.45      59
0.50      68
0.55      79
0.60      93
0.65     108
0.70     127
0.75     155
0.80     193
0.85     246
0.90     334
0.95     520
1.00    9254
Name: rating, dtype: int64
```

```
Name: rating, dtype: int64
```

how many ratings at the last 5% of all ratings??

In [33]:

```
print('\n No of ratings at last 5 percentile : {}\n'.format(sum(no_of_rated_movies_per_user>= 520)
) )
```

```
 No of ratings at last 5 percentile : 6940
```

In [34]:

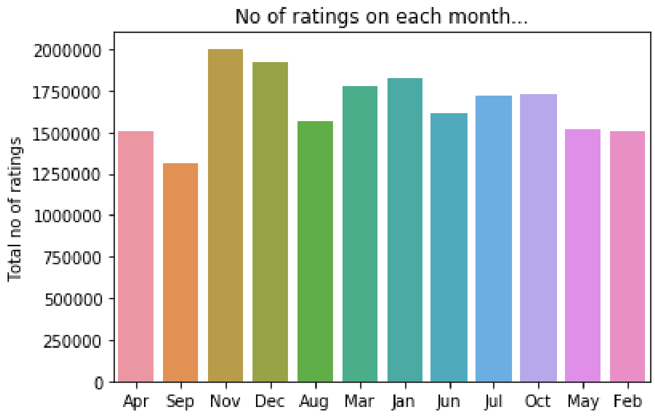
```
ratings_data.head(3)
```

Out[34]:

	userId	movieId	rating	gave_rating_year	gave_rating_month
0	1	2	3.5	2005	Apr
1	1	29	3.5	2005	Apr
2	1	32	3.5	2005	Apr

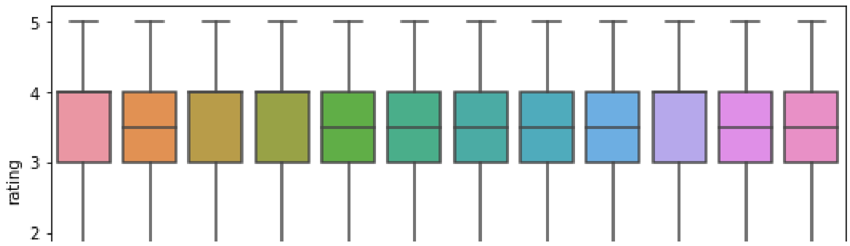
In [35]:

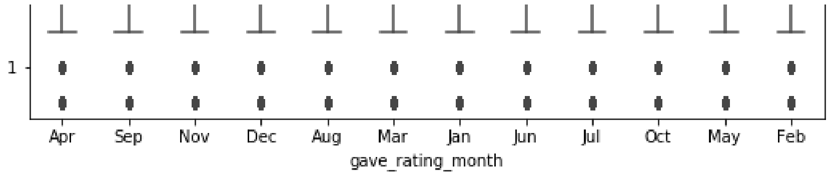
```
fig, ax = plt.subplots()
sns.countplot(x='gave_rating_month', data=ratings_data, ax=ax)
plt.title('No of ratings on each month...')
plt.ylabel('Total no of ratings')
plt.xlabel('')
#ax.set_yticklabels([human(item, 'M') for item in ax.get_yticks()])
plt.show()
```



In [36]:

```
from datetime import datetime
start = datetime.now()
fig = plt.figure(figsize=plt.figaspect(.45))
sns.boxplot(y='rating', x='gave_rating_month', data=ratings_data)
plt.show()
print(datetime.now() - start)
```





```
0:00:08.468619
```

In [37]:

```
avg_month_df = ratings_data.groupby(by=['gave_rating_month'])['rating'].mean()
print(" AVerage ratings")
print("-"*30)
print(avg_month_df)
print("\n")
```

```
 AVerage ratings
------------------------------
gave_rating_month
Apr    3.528203
Aug    3.505279
Dec    3.539704
Feb    3.524092
Jan    3.520437
Jul    3.517270
Jun    3.513506
Mar    3.496393
May    3.514545
Nov    3.546288
Oct    3.565256
Sep    3.528368
Name: rating, dtype: float64
```

In [ ]: