Introduction to Pandas

Installation

Simply,

```
pip install pandas
```

Reading data from a CSV file

You can read data from a CSV file using the `read_csv` function. By default, it assumes that the fields are comma-separated.

In [60]:
```
# import pandas
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

> The `imdb.csv` dataset contains Highest Rated IMDb "Top 1000" Titles.

In [61]:
```
# load imdb dataset as pandas dataframe
df = pd.read_csv('imdb_1000.csv')
df
```

Out[61]:

	star_rating	title	content_rating	genre	duration	actors_lis
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tin Robbins' u'Morgai Freeman' u'Bob Gunt..

	star_rating	title	content_rating	genre	duration	actors_lis
1	9.2	The Godfather	R	Crime	175	[u'Marlo Brando', u'A Pacino' u'James Caan'
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino' u'Robert De Niro' u'Rober Duv..
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christia Bale' u'Heat Ledger' u'Aaron E..
4	8.9	Pulp Fiction	R	Crime	154	[u'Joh Travolta' u'Um Thurman' u'Samuel L...
...	...	...	...	...	...	..
974	7.4	Tootsie	PG	Comedy	116	[u'Dusti Hoffman' u'Jessic Lange', u'Te G..
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J Fox' u'Christophe Lloyd' u'Ma..
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russe Crowe' u'Pau Bettany' u'Billy Bo..
977	7.4	Poltergeist	PG	Horror	114	[u'JoBet Williams' u"Heathe O'Rourke" u'Cr..
978	7.4	Wall Street	R	Crime	126	[u'Charli Sheen' u'Michae Douglas' u'Tamar..

979 rows × 6 columns

```
In [62]: # show first 5 rows of imdb_df
```

```
df.head(5)
```

Out[62]:

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....

> The `bikes.csv` dataset contains information about the number of bicycles that used certain bicycle lanes in Montreal in the year 2012.

In [98]:
```
# load bikes dataset as pandas dataframe
df2 = pd.read_csv('bikes.csv',sep=';',encoding='latin1',parse_dates
=['Date'], dayfirst=True)
df2
```

Out[98]:

	Date	Unnamed: 1	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1	Br
0	2012-01-01	00:00	16	35	51	38	

	Date	Unnamed: 1	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1	Br
1	2012-01-02	00:00	43	83	153	68	
2	2012-01-03	00:00	58	135	248	104	
3	2012-01-04	00:00	61	144	318	116	
4	2012-01-05	00:00	95	197	330	124	
...	...	...	...	...	...	...	
361	2012-12-27	00:00	8	12	7	4	
362	2012-12-28	00:00	0	35	3	38	
363	2012-12-29	00:00	0	27	8	42	
364	2012-12-30	00:00	0	5	1	5	
365	2012-12-31	00:00	0	4	3	8	

366 rows × 11 columns

In [64]:
```
# show first 3 rows of bikes_df
df2.head(3)
```

Out[64]:

	Date	Unnamed: 1	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1
0	01/01/2012	00:00	16	35	51	38
1	02/01/2012	00:00	43	83	153	68
2	03/01/2012	00:00	58	135	248	104

Selecting columns

When you read a CSV, you get a kind of object called a DataFrame, which is made up of rows and columns. You get columns out of a DataFrame the same way you get elements out of a dictionary.

In [65]:
```python
# list columns of imdb_df
print(df.columns)
```

```
Index(['star_rating', 'title', 'content_rating', 'genre', 'dura
tion',
       'actors_list'],
      dtype='object')
```

In [66]:
```python
# what are the datatypes of values in columns
df.dtypes
```

Out[66]:
```
star_rating       float64
title              object
content_rating     object
genre              object
duration            int64
actors_list        object
dtype: object
```

In [67]:
```python
# list first 5 movie titles
df[['title']].head(5)
```

Out[67]:

	title
0	The Shawshank Redemption
1	The Godfather
2	The Godfather: Part II
3	The Dark Knight
4	Pulp Fiction

In [68]:
```python
# show only movie title and genre
df[['title', 'genre']]
```

Out[68]:

	title	genre
0	The Shawshank Redemption	Crime
1	The Godfather	Crime
2	The Godfather: Part II	Crime
3	The Dark Knight	Action
4	Pulp Fiction	Crime
...	...	...
974	Tootsie	Comedy
975	Back to the Future Part III	Adventure

	title	genre
976	Master and Commander: The Far Side of the World	Action
977	Poltergeist	Horror
978	Wall Street	Crime

979 rows × 2 columns

Understanding columns

On the inside, the type of a column is `pd.Series` and pandas Series are internally numpy arrays. If you add `.values` to the end of any Series, you'll get its internal **numpy array**.

In [69]:
```
# show the type of duration column
print(type(df['duration']))
```

```
<class 'pandas.core.series.Series'>
```

In [70]:
```
# show duration values of movies as numpy arrays
d=df[['duration']].to_numpy()
d
```

Out[70]:
```
array([[142],
       [175],
       [200],
       [152],
       [154],
       [ 96],
       [161],
       [201],
       [195],
       [139],
       [178],
       [148],
       [124],
       [142],
       [179],
       [169],
       [133],
       [207],
       [146],
       [121],
       [136]
```

```
[130],
[130],
[130],
[106],

[127],
[116],
[175],
[118],
[110],
[ 87],
[125],
[112],
[102],
[107],
[119],
[ 87],
[169],
[115],
[112],
[109],
[189],
[110],
[150],
[165],
[155],
[137],
[113],
[165],
[ 95],
[151],
[155],
[153],
[125],
[130],
[116],
[ 89],
[137],
[117],
[ 88],
[165],
[170],
[ 89],
[146],
[ 99],
[ 98],
[116],
[156],
[122],
[149],
[134],
[122],
[136],
[157],
[123],
[119],
[137],
[128],
[120],
[229],
[107],
[134],
[103],
[177],
[129],
[102],
[216],
[136],
[ 93],
[ 68],
[189],
[ 99],
[108],
[113],
[181],
[103],
[138],
[110],
[129],
[ 88],
[160],
[126],
[ 91],
[116],
[125],
[143],
[ 93],
[102],
[132],
[153],
[183],
[160],
[120],
[138],
[140],
[153],
[170],
[129],
[ 81],
[127],
[131],
[172],
[115],
[108],
[107],
[129],
[156],
[ 96],
[ 91],
[ 95],
[162],
[130],
[ 86],
[186],
[151],
[ 96],
[170],
[118],
[161],
[131],
[126],
[131],
[129],
[224],
[180],
[105],
[117],
[140],
[119],
[124],
[130],
[139],
[107],
[132],
[117],
[126],
[122],
[178],
[238],
[149],
[172],
[ 98],
[116],
[116],
[123],
[148],
[123],
[182],
[ 92],
[ 93],
[100],
[135],
[105],
[ 94],
[140],
[ 83],
[ 95],
[ 98],
[143],
[ 99],
[ 98],
[121],
[163],
[121],
[167],
[188],
[121],
[109],
[110],
[129],
[127],
[ 94],
[107],
[100],
[117],
[129],
[120],
[121],
[133],
[111],
[122],
[101],
[134],
[165],
[138],
[212],
[154],
[ 89],
[134],
[ 93],
[114],
[ 88],
[130],
[101],
[158],
[ 99],
[108],
[124],
[132],
[113],
[131],
[191],
[167],
[130],
[147],
[102],
[ 88],
[165],
[132],
[118],
[101],
[108],
[174],
[ 98],
[ 92],
[ 98],
[106],
[ 85],
[101],
[105],
[115],
[115],
[124],
[105],
[103],
[138],
[184],
[120],
[ 99],
[131],
[138],
[ 98],
[123],
[118],
[114],
[118],
[112],
[124],
[160],
[ 67],
[146],
[125],
[115],
[134],
[141],
[129],
[104],
[ 94],
[124],
[150],
[119],
[128],
[143],
[ 85],
[151],
[118],
[101],
[ 99],
[ 92],
[125],
[102],
[106],
[107],
[ 91],
[ 80],
[122],
[102],
[112],
[ 92],
[135],
[136],
[153],
[105],
[126],
[ 68],
[ 84],
[103],
[145],
[ 80],
[106],
[127],
[178],
[137],
[172],
[ 76],
[130],
[ 96],
[ 82],
[115],
[ 95],
[143],
[125],
[120],
[127],
[112],
[104],
[113],
[189],
[127],
[188],
[ 96],
[113],
[132],
[122],
[126],
[140],
[133],
[112],
[102],
[ 93],
[109],
[ 81],
[108],
[111],
[119],
[ 96],
[197],
[127],
[138],
[ 66],
[181],
[141],
[144],
[128],
[ 83],
[121],
[ 98],
[117],
[ 94],
[174],
[112],
[108],
[ 70],
[ 99],
[ 96],
[137],
[136],
[100],
[ 90],
[ 96],
[161],
[115],
[181],
[169],
[ 98],
[136],
[138],
[119],
[123],
[112],
[120],
[103],
[ 97],
[109],
[165],
[156],
[147],
[100],
[126],
[115],
[105],
[144],
[102],
[107],
[119],
[ 90],
[197],
[ 86],
[113],
[133],
[ 64],
[141],
[113],
[122],
[105],
[119],
[102],
[122],
[123],
[113],
[ 97],
[117],
[111],
[ 85],
[ 99],
[134],
[ 75],
[109],
[117],
[ 91],
[128],
[111],
[119],
[100],
[114],
[ 92],
[141],
[101],
[115],
[127],
[ 96],
[129],
[135],
[109],
[157],
[193],
[ 94],
[155],
[113],
[130],
[117],
[159],
[141],
[112],
[162],
[126],
[120],
[144],
[120],
[109],
[ 97],
[108],
[115],
[136],
[ 90],
[170],
[220],
[116],
[134],
[124],
[ 91],
[103],
[ 96],
[119],
[ 98],
[101],
[ 93],
[179],
[155],
[121],
[103],
[127],
[103],
[107],
[100],
[ 92],
[ 94],
[158],
[ 86],
[115],
[130],
[107],
[100],
[108],
[124],
[131],
[122],
[242],
[127],
[141],
[126],
[ 89],
[113],
[152],
[107],
[ 92],
[145],
[ 84],
[126],
[132],
[ 78],
[117],
[128],
[100],
[128],
[100],
[143],
[107],
[100],
[125],
[106],
[157],
[ 94],
[104],
[ 94],
[153],
[123],
[120],
[105],
[166],
[112],
[ 94],
[111],
[ 87],
[ 80],
[108],
[102],
[105],
[136],
[101],
[100],
[108],
[101],
[106],
[105],
[ 88],
[129],
[138],
[129],
[118],
[139],
[123],
[150],
[132],
[ 80],
[178],
[ 79],
[163],
[114],
[144],
[130],
[154],
[ 81],
[ 95],
[101],
[120],
[ 93],
[115],
[106],
[120],
[110],
[123],
[142],
[ 99],
[112],
[120],
[122],
[130],
[139],
[129],
[141],
[127],
[130],
[118],
[110],
[115],
[101],
[112],
[108],
[143],
[152],
[117],
[121],
[119],
[114],
[122],
[105],
[110],
[116],
[139],
[121],
[109],
[146],
[113],
[100],
[121],
[113],
[113],
[117],
[ 93],
[101],
[ 90],
[131],
[ 98],
[ 93],
[121],
[ 82],
[124],
[100],
[117],
[ 99],
[116],
[123],
[114],
[133],
[105],
[127],
[144],
[ 90],
[118],
[122],
[102],
[187],
[ 87],
[154],
[ 89],
[ 88],
[129],
[ 95],
[118],
[ 98],
[194],
[114],
[ 80],
[130],
[135],
[101],
[105],
[129],
[134],
[143],
[202],
[ 89],
[106],
[ 92],
[137],
[124],
[122],
[113],
[112],
[110],
[135],
[ 88],
[146],
[104],
[125],
[131],
[ 91],
[142],
[129],
[158],
[102],
[ 99],
[101],
[104],
[119],
[103],
[ 88],
[100],
[146],
[150],
[140],
[140],
[110],
[ 97],
[102],
[152],
[100],
[ 94],
[ 85],
[104],
[112],
[191],
[119],
[111],
[ 94],
[103],
[134],
[157],
[158],
[127],
[101],
[109],
[ 98],
[134],
[168],
[ 93],
[145],
[111],
[ 88],
[123],
[121],
[144],
[116],
[124],
[147],
[106],
[113],
[129],
[ 94],
[126],
[128],
[111],
[146],
[ 85],
[105],
[132],
[130],
[162],
[113],
[110],
[113],
[162],
[110],
[ 78],
[ 92],
[ 95],
[138],
[102],
[128],
[ 96],
[164],
[126],
[110],
[109],
[122],
[124],
[128],
[157],
[120],
[102],
[152],
[170],
[102],
[116],
[164],
[ 85],
[104],
[ 99],
[ 81],
[131],
[101],
[121],
[ 69],
[ 92],
[ 94],
[141],
[116],
[ 98],
[135],
[121],
[108],
[114],
[102],
[112],
[140],
[157],
[128],
[108],
[139],
[ 83],
[131],
[113],
[136],
[ 96],
[118],
[108],
[ 83],
[205],
[124],
[111],
[137],
[104],
[109],
[115],
[ 99],
[114],
[133],
[131],
[131],
[123],
[126],
[ 97],
[126],
[ 89],
[122],
[121],
[125],
[105],
[118],
[ 81],
[115],
[126],
[113],
[107],
[126],
[111],
[ 85],
[120],
[ 93],
[100],
[107],
[126],
[131],
[116],
[143],
[ 93],
[ 96],
[160],
[137],
[119],
[114],
[ 98],
[111],
[138],
[144],
[107],
[114],
[125],
[146],
[113],
[ 85],
[ 97],
[ 93],
[113],
[128],
[114],
[147],
[127],
[167],
[124],
[109],
[124],
[102],
[130],
[125],
[ 98],
[154],
[ 97],
[128],
[ 98],
[101],
[147],
[172],
[125],
[ 88],
[125],
[121],
[112],
[120],
[133],
[110],
[135],
[110],
[ 80],
[135],
[101],
[112],
[124],
[129],
[104],
[ 96],
[105],
[109],
[117],
[ 78],
[138],
[106],
[ 90],
[106],
[ 91],
[112],
[134],
[112],
[104],
[101],
[120],
[123],
[133],
[110],
[129],
[120],
[ 83],
[ 88],
[ 96],
[139],
[113],
[ 84],
[106],
[170],
[144],
[104],
[153],
[ 85],
[130],
[ 93],
[148],
[115],
[102],
[125],
[106],
[121],
[143],
[116],
[ 99],
[116],
[128],
[119],
[104],
[114],
[152],
[129],
[103],
[112],
[ 92],
[141],
[106],
[109],
[104],
[122],
[111],
[112],
[116],
[130],
[133],
[134],
[ 92],
[104],
[110],
[ 97],
[100],
[118],
[112],
[138],
[ 92],
[118],
[140],
[128],
[101],
[ 75],
[ 97],
[111],
[175],
[ 94],
[ 97],
[132],
[120],
[107],
[ 89],
[ 86],
[103],
[108],
[105],
[117],
[102],
[104],
[157],
[104],
[ 98],
[109],
[ 96],
[103],
[114],
[ 99],
[118],
[150],
[105],
[ 87],
[136],
[162],
[109],
[107],
[ 90],
[112],
[126],
[116],
[118],
[138],
[114],
[126]], dtype=int64)
```

Applying functions to columns

Use `.apply` function to apply any function to each element of a column.

In [16]:
```
# convert all the movie titles to uppercase
df['title']=df['title'].str.upper()
df['title']
```

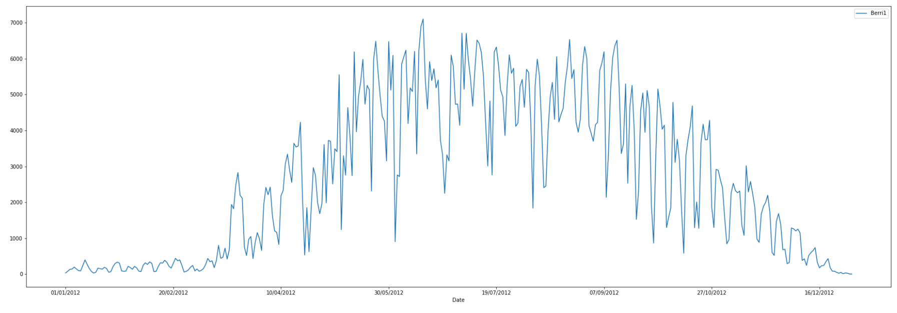
Out[16]:
```
0                                THE SHAWSHANK REDEMPTION
1                                           THE GODFATHER
2                                  THE GODFATHER: PART II
3                                         THE DARK KNIGHT
4                                            PULP FICTION
                              ...                        
974                                               TOOTSIE
975                           BACK TO THE FUTURE PART III
976    MASTER AND COMMANDER: THE FAR SIDE OF THE WORLD
977                                           POLTERGEIST
978                                           WALL STREET
Name: title, Length: 979, dtype: object
```

Plotting a column

Use `.plot()` function!

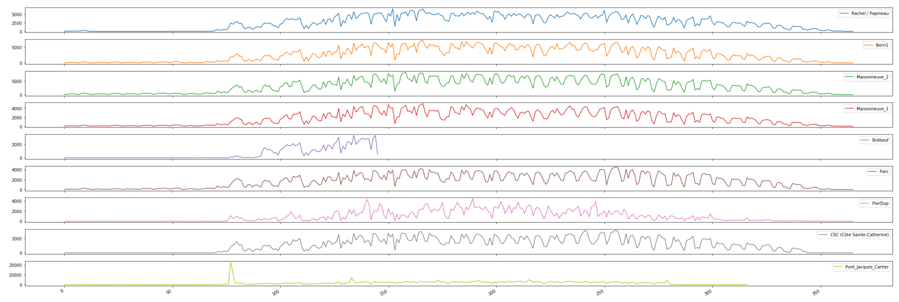
In [71]:
```
# plot the bikers travelling to Berri1 over the year
import matplotlib.pyplot as plt

df2.plot(x='Date',y='Berri1',kind='line',figsize=(30,10))
plt.show()
```



In [72]:
```
# plot all the columns of bikes_df
df2.plot(subplots=True,figsize=(30,10))
```

```
plt.tight_layout()
plt.show()
```



Value counts

Get count of unique values in a particular column/Series.

In [73]:

```
# what are the unique genre in imdb_df?
#s = pd.value_counts(df.genre)
#s1 = pd.Series({'nunique': len(s), 'unique values': s.index.tolist
()})
#s.append(s1)
print(df.genre.unique())
u = df.genre.nunique()
u
```

```
['Crime' 'Action' 'Drama' 'Western' 'Adventure' 'Biography' 'Co
medy'
 'Animation' 'Mystery' 'Horror' 'Film-Noir' 'Sci-Fi' 'History'
'Thriller'
 'Family' 'Fantasy']
```

Out[73]:

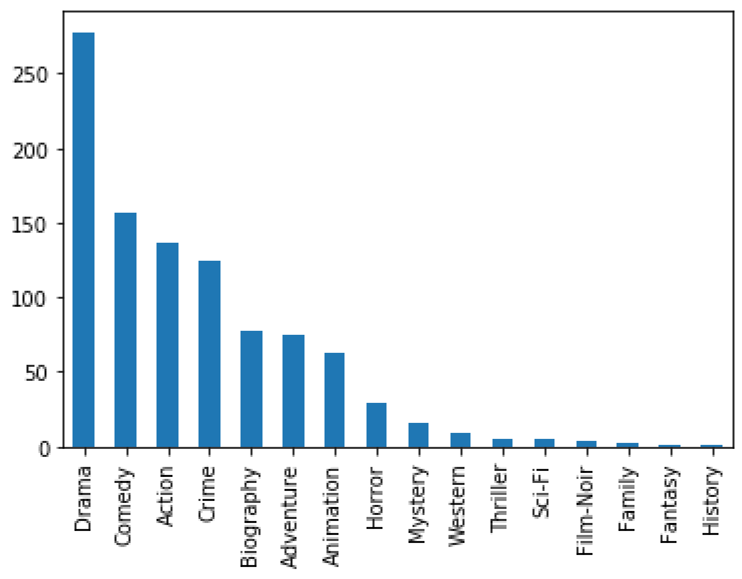
```
16
```

In [74]:

```
# plotting value counts of unique genres as a bar chart
df['genre'].value_counts().plot(kind='bar')
```

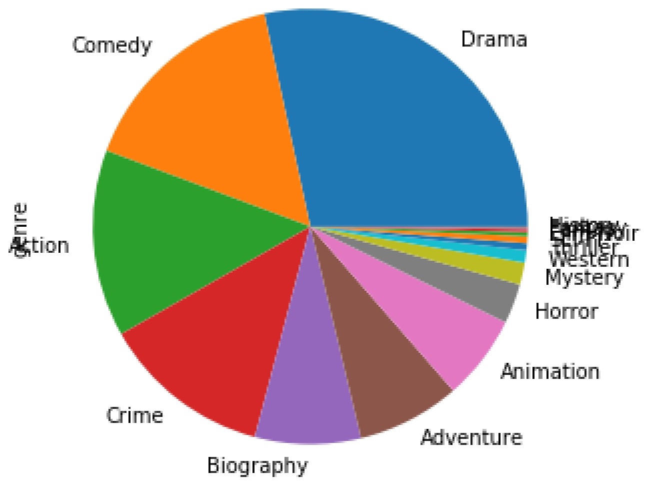
Out[74]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19a97f3b6c8>
```



In [75]:
```
# plotting value counts of unique genres as a pie chart
df['genre'].value_counts().plot(kind='pie',figsize=(10,5))
```

Out[75]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19a979957c8>
```



Index

DATAFRAME = COLUMNS + INDEX + ND DATA

SERIES = INDEX + 1-D DATA

Index or (**row labels**) is one of the fundamental data structure of pandas. It can be thought of as an **immutable array** and an **ordered set**.

> Every row is uniquely identified by its index value.

In [76]:
```
# show index of bikes_df
df2.index
```

Out[76]:
```
RangeIndex(start=0, stop=366, step=1)
```

In [77]:
```
# get row for date 2012-01-01
df2[df2['Date']=="2012-01-01"]
```

Out[77]:

Date	Unnamed: 1	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1	Brébeu

To get row by integer index:

Use `.iloc[]` for purely integer-location based indexing for selection by position.

In [78]:
```
# show 11th row of imdb_df using iloc
df2.iloc[11]
```

Out[78]:
```
Date                             12/01/2012
Unnamed: 1                            00:00
Rachel / Papineau                        63
Berri1                                  157
Maisonneuve_2                           261
Maisonneuve_1                           134
Brébeuf                                   3
Parc                                    137
PierDup                                   9
CSC (Côte Sainte-Catherine)               1
Pont_Jacques_Cartier                     15
Name: 11, dtype: object
```

```
Name: 11, dtype: object
```

Selecting rows where column has a particular value

In [79]:
```
# select only those movies where genre is adventure
df.loc[df['genre'] == 'Adventure']['title']
```

Out[79]:
```
7          The Lord of the Rings: The Return of the King
10     The Lord of the Rings: The Fellowship of the Ring
14                 The Lord of the Rings: The Two Towers
15                                          Interstellar
54                                    Back to the Future
                             ...
936                                            True Grit
937                                            Labyrinth
943                                      The Bucket List
953                                 The NeverEnding Story
975                              Back to the Future Part III
Name: title, Length: 75, dtype: object
```

In [80]:
```
# which genre has highest number of movies with star rating above 8
and duration more than 130 minutes?
df[(df['star_rating']>8) & (df['duration']>130)]['genre']
```

Out[80]:
```
0          Crime
1          Crime
2          Crime
3         Action
4          Crime
         ...
273    Biography
288        Drama
289        Drama
290        Crime
296       Action
Name: genre, Length: 115, dtype: object
```

Adding a new column to DataFrame

In [100]:
```
# add a weekday column to bikes_df
df2['weekday']=df2['Date'].dt.day_name()
df2
```

Out[100]:

	Date	Unnamed: 1	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1
0	2012-01-01	00:00	16	35	51	38
1	2012-01-02	00:00	43	83	153	68
2	2012-01-03	00:00	58	135	248	104
3	2012-01-04	00:00	61	144	318	116
4	2012-01-05	00:00	95	197	330	124
...	...	...	...	...	...	...
361	2012-12-27	00:00	8	12	7	4
362	2012-12-28	00:00	0	35	3	38
363	2012-12-29	00:00	0	27	8	42
364	2012-12-30	00:00	0	5	1	5
365	2012-12-31	00:00	0	4	3	8

366 rows × 12 columns

Deleting an existing column from DataFrame

In [84]:

```python
# remove column 'Unnamed: 1' from bikes_df
z=df2.drop('Unnamed: 1',axis=1)
z
```

Out[84]:

	Date	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1	Brébeuf
0	01/01/2012	16	35	51	38	5.0

	Date	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1	Brébeuf
1	02/01/2012	43	83	153	68	11.0
2	03/01/2012	58	135	248	104	2.0
3	04/01/2012	61	144	318	116	2.0
4	05/01/2012	95	197	330	124	6.0
...	...	...	...	...	...	...
361	27/12/2012	8	12	7	4	NaN
362	28/12/2012	0	35	3	38	NaN
363	29/12/2012	0	27	8	42	NaN
364	30/12/2012	0	5	1	5	NaN
365	31/12/2012	0	4	3	8	NaN

366 rows × 10 columns

Deleting a row in DataFrame

In [85]:

```
# remove row no. 1 from bikes_df
z.drop([1],axis=0)
```

Out[85]:

	Date	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1	Brébeuf
0	01/01/2012	16	35	51	38	5.0
2	03/01/2012	58	135	248	104	2.0
3	04/01/2012	61	144	318	116	2.0
4	05/01/2012	95	197	330	124	6.0
5	06/01/2012	75	146	244	98	4.0
...	...	...	...	...	...	...
361	27/12/2012	8	12	7	4	NaN
362	28/12/2012	0	35	3	38	NaN
363	29/12/2012	0	27	8	42	NaN
364	30/12/2012	0	5	1	5	NaN
365	31/12/2012	0	4	3	8	NaN

365 rows × 10 columns

Group By

Any groupby operation involves one of the following operations on the original object. They are −

- Splitting the Object
- Applying a function
- Combining the results

In many situations, we split the data into sets and we apply some functionality on each subset. In the apply functionality, we can perform the following operations −

- **Aggregation** − computing a summary statistic
- **Transformation** − perform some group-specific operation
- **Filtration** − discarding the data with some condition

In [23]:
```
# group imdb_df by movie genres
#print(df.groupby('genre').groups)
by_genres=df.groupby('genre')
by_genres
```

Out[23]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000
19A95072408>
```

In [81]:
```
# get crime movies group
cr = df.groupby('genre')
print(cr.get_group('Crime'))
```

```
     star_rating                          title content_rating  genr
e  duration  \
0            9.3  THE SHAWSHANK REDEMPTION              R  Crim
e       142
1            9.2             THE GODFATHER              R  Crim
e       175
2            9.1    THE GODFATHER: PART II              R  Crim
e       200
4            8.9              PULP FICTION              R  Crim
e       154
21           8.7               CITY OF GOD              R  Crim
e       130
..           ...                       ...            ...
...       ...
927          7.5                     BRICK              R  Crim
```

```
e       110
931          7.4              MEAN STREETS             R  Crim
e       112
950          7.4                     BOUND             R  Crim
e       108
969          7.4       LAW ABIDING CITIZEN             R  Crim
e       109
978          7.4               WALL STREET             R  Crim
e       126

                                               actors_list
0    [u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1      [u'Marlon Brando', u'Al Pacino', u'James Caan']
2    [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
4    [u'John Travolta', u'Uma Thurman', u'Samuel L....
21   [u'Alexandre Rodrigues', u'Matheus Nachtergael...
..                                                 ...
927  [u'Joseph Gordon-Levitt', u'Lukas Haas', u'Emi...
931  [u'Robert De Niro', u'Harvey Keitel', u'David ...
950  [u'Jennifer Tilly', u'Gina Gershon', u'Joe Pan...
969  [u'Gerard Butler', u'Jamie Foxx', u'Leslie Bibb']
978  [u'Charlie Sheen', u'Michael Douglas', u'Tamar...

[124 rows x 6 columns]
```

In [82]:

```
# get mean of movie durations for each group
print(cr['duration'].agg(np.mean))
```

```
genre
Action       126.485294
Adventure    134.840000
Animation     96.596774
Biography    131.844156
Comedy       107.602564
Crime        122.298387
Drama        126.539568
Family       107.500000
Fantasy      112.000000
Film-Noir     97.333333
History       66.000000
Horror       102.517241
Mystery      115.625000
Sci-Fi       109.000000
Thriller     114.200000
Western      136.666667
Name: duration, dtype: float64
```

In [24]:
```
# change duration of all movies in a particular genre to mean durat
ion of the group
crime=df.groupby('genre').get_group('Crime')
crime['duration']=crime['duration'].mean()
crime[['title','duration']]
```

```
C:\Users\user\anaconda3\lib\site-packages\ipykernel_launcher.p
y:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFra
me.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.or
g/pandas-docs/stable/user_guide/indexing.html#returning-a-view-
versus-a-copy
  This is separate from the ipykernel package so we can avoid d
oing imports until
```

Out[24]:

	title	duration
0	THE SHAWSHANK REDEMPTION	122.298387
1	THE GODFATHER	122.298387
2	THE GODFATHER: PART II	122.298387
4	PULP FICTION	122.298387
21	CITY OF GOD	122.298387
...	...	...
927	BRICK	122.298387
931	MEAN STREETS	122.298387
950	BOUND	122.298387
969	LAW ABIDING CITIZEN	122.298387
978	WALL STREET	122.298387

124 rows × 2 columns

In [86]:
```
# drop groups/genres that do not have average movie duration greate
r than 120.
y=df.drop(df[df.duration<120].index,inplace=True)
print(y)
df
```

```
None
```

Out[86]:

	star_rating	title	content_rating	genre	duration	actors_list

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...
4	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
967	7.4	The Rock	R	Action	136	[u'Sean Connery', u'Nicolas Cage', u'Ed Harris']
968	7.4	The English Patient	R	Drama	162	[u'Ralph Fiennes', u'Juliette Binoche', u'Will...
973	7.4	The Cider House Rules	PG-13	Drama	126	[u'Tobey Maguire', u'Charlize Theron', u'Micha...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...

	star_rating	title	content_rating	genre	duration	actors_list
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

454 rows × 6 columns

```
In [101]: # group weekday wise bikers count
          coun=df2.groupby('weekday').sum()
```

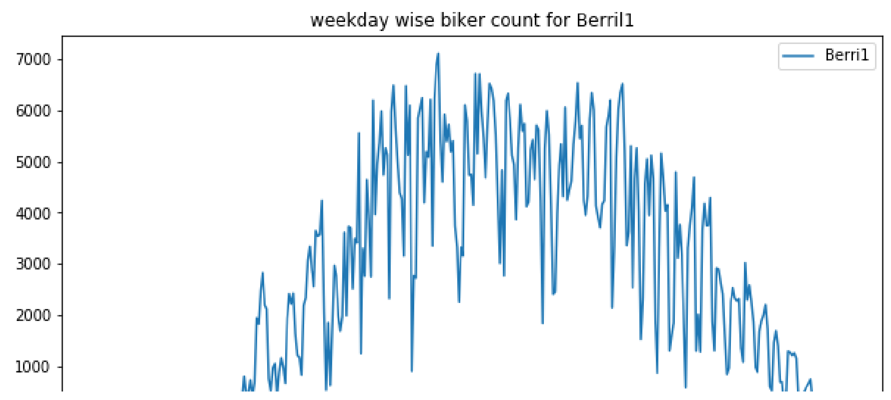
```
In [102]: # get weekday wise biker count
          coun
```

Out[102]:

	Rachel / Papineau	Berri1	Maisonneuve_2	Maisonneuve_1	BrÃ©beuf	
weekday						
Friday	146979	150493	183961	104584	12259.0	9
Monday	138881	142285	174610	95565	15436.0	9
Saturday	118560	105635	109902	64872	11170.0	5
Sunday	122115	102447	102272	57438	12032.0	5
Thursday	150971	169976	210039	118633	15679.0	11
Tuesday	131632	145795	179939	99421	10629.0	10
Wednesday	144531	163603	200273	112344	14876.0	11

```
In [103]: # plot weekday wise biker count for 'Berril1'
          df2.plot(x='weekday',y='Berri1',title='weekday wise biker count for
          Berril1',figsize=(10,5))
```

```
Out[103]: <matplotlib.axes._subplots.AxesSubplot at 0x19a963dab48>
```


weekday wise biker count for Berril1
Berri1