```
import pandas as pd

A Series is essentially a column, and a DataFrame is a
multi-dimensional table made up of a collection of Series.

data = {
'apples': [3, 2, 0, 1],
'oranges': [0, 3, 7, 2]
}

purchases = pd.DataFrame(data)

purchases

purchases = pd.DataFrame(data, index=['June', 'Robert', '
Lily', 'David'])

purchases

purchases.loc['June']

df = pd.read_csv('purchases.csv')

df

df = pd.read_csv('purchases.csv', index_col=0)

df

movies_df = pd.read_csv("IMDB-Movie-Data.csv", index_col="
Title")

movies_df.head()

movies_df.tail(2)
movies_df.info()
movies_df.shape

temp_df = movies_df.append(movies_df)

temp_df.shape
temp_df.drop_duplicates(inplace=True)
temp_df = movies_df.append(movies_df) # make a new copy

temp_df.drop_duplicates(inplace=True, keep=False)

temp_df.shape

#Column cleanup
#Many times datasets will have verbose column names with symbols, upper and lowercase words, spaces, and typos. To make selecting data by column name easier we can spend a little time cleaning up their names.

movies_df.columns
movies_df.rename(columns={
'Runtime (Minutes)': 'Runtime',
'Revenue (Millions)': 'Revenue_millions'
}, inplace=True)

movies_df.columns

movies_df.columns = ['rank', 'genre', 'description', 'director', 'actors', 'year', 'runtime',
'rating', 'votes', 'revenue_millions', 'metascore']

movies_df.columns

movies_df.columns = [col.lower() for col in movies_df]

movies_df.columns

movies_df.isnull()
movies_df.isnull().sum()
movies_df.dropna()
movies_df.dropna(axis=1)
#What's with this axis=1parameter?

#It's not immediately obvious where axis comes from and why you need it to be 1 for it to affect columns. To see why, just look at the .shape output:

#movies_df.shape

#Out: (1000, 11)

#As we learned above, this is a tuple that represents the shape of the DataFrame, i.e. 1000 rows and 11 columns. Note that the rows are at index zero of this tuple and columns are at index one of this tuple. This is why axis=1
```

```
 affects columns. This comes from NumPy, and is a great example of why learning NumPy is worth your time.

#Imputation
#Imputation is a conventional feature engineering technique used to keep valuable data that have null values.

#Let's look at imputing the missing values in the revenue_millions column. First we'll extract that column into its own variable:

revenue = movies_df['revenue_millions']
revenue.head()
revenue_mean = revenue.mean()

revenue_mean
revenue.fillna(revenue_mean, inplace=True)
movies_df.isnull().sum()

#.describe() can also be used on a categorical variable to get the count of rows, unique count of categories, top category, and freq of top category:

movies_df.describe()
movies_df['genre'].describe()
#.value_counts() can tell us the frequency of all values in a column:

movies_df['genre'].value_counts().head(10)

#Relationships between continuous variables
#By using the correlation method .corr() we can generate the relationship between each continuous variable:

#movies_df.corr()

genre_col = movies_df['genre']

type(genre_col)

genre_col = movies_df[['genre']] #2

type(genre_col)

```

```
subset = movies_df[['genre', 'rating']]

subset.head()

For rows, we have two options:

#.loc - locates by name
#.iloc- locates by numerical index
#Remember that we are still indexed by movie Title, so to
 use .loc we give it the Title of a movie:

prom = movies_df.loc["Prometheus"]

prom

prom = movies_df.iloc[1]

movie_subset = movies_df.loc['Prometheus':'Sing']

movie_subset = movies_df.iloc[1:4]

movie_subset

condition = (movies_df['director'] == "Ridley Scott")

condition.head()

We want to filter out all movies not directed by Ridley
Scott, in other words, we don't want the False films. To
return the rows where that condition is True we have to
pass this operation into the DataFrame:

movies_df[movies_df['director'] == "Ridley Scott"]
movies_df[movies_df['rating'] >= 8.6].head(3)
movies_df[(movies_df['director'] == 'Christopher Nolan')
| (movies_df['director'] == 'Ridley Scott')].head()

def rating_function(x):
if x >= 8.0:
return "good"
else:
return "bad"

movies_df["rating_category"] = movies_df["rating"].apply(
rating_function)
```

```

movies_df.head(2)

movies_df["rating_category"] = movies_df["rating"].apply(
lambda x: 'good' if x >= 8.0 else 'bad')

movies_df.head(2)

import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 20, 'figure.figsize': (
10, 8)}) # set font and plot size to be larger

movies_df.plot(kind='scatter', x='rating', y='
revenue_millions', title='Revenue (millions) vs Rating');

movies_df['rating'].plot(kind="box");

```