# Pandas -- Data Analysis in Python

iTech

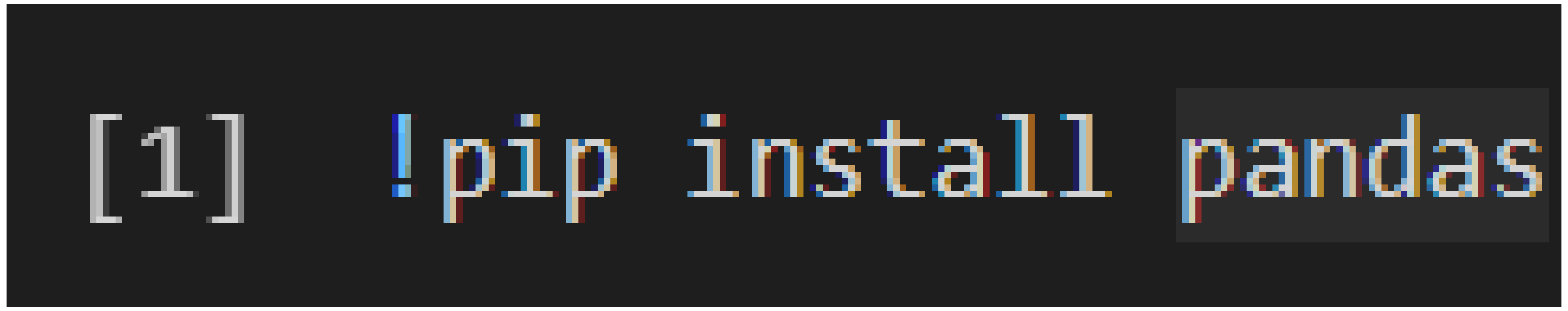
Miss Laiba Batool
Coding Instructor

## *Introduction to Pandas*

- **Pandas is a powerful tool for working with data.**
- **It helps you clean, transform, and analyze your data.**
- **You can load data from a CSV file into a table called a DataFrame.**
- **With Pandas, you can calculate statistics, clean data, and make plots.**
- **Pandas helps you understand your data before doing advanced analysis.**

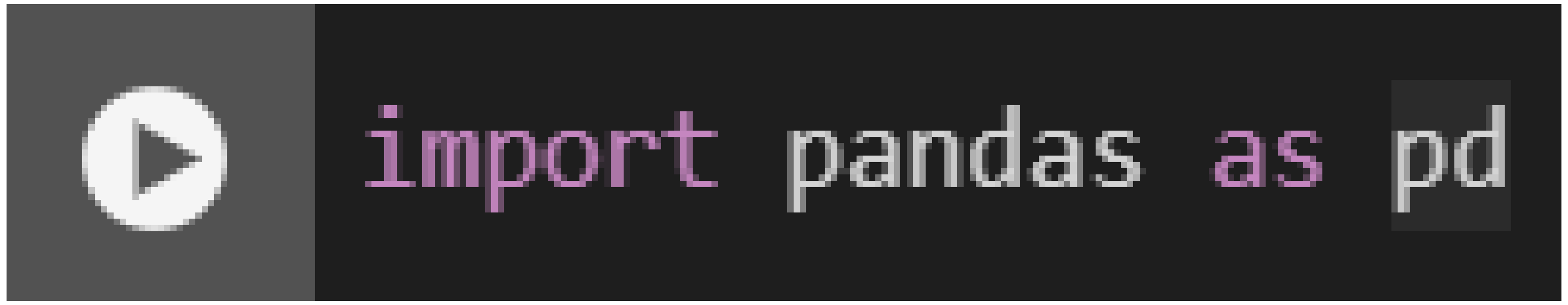
## *Installing and importing Pandas*

- You can use pip to install the pandas library.



```
[1]  !pip install pandas
```

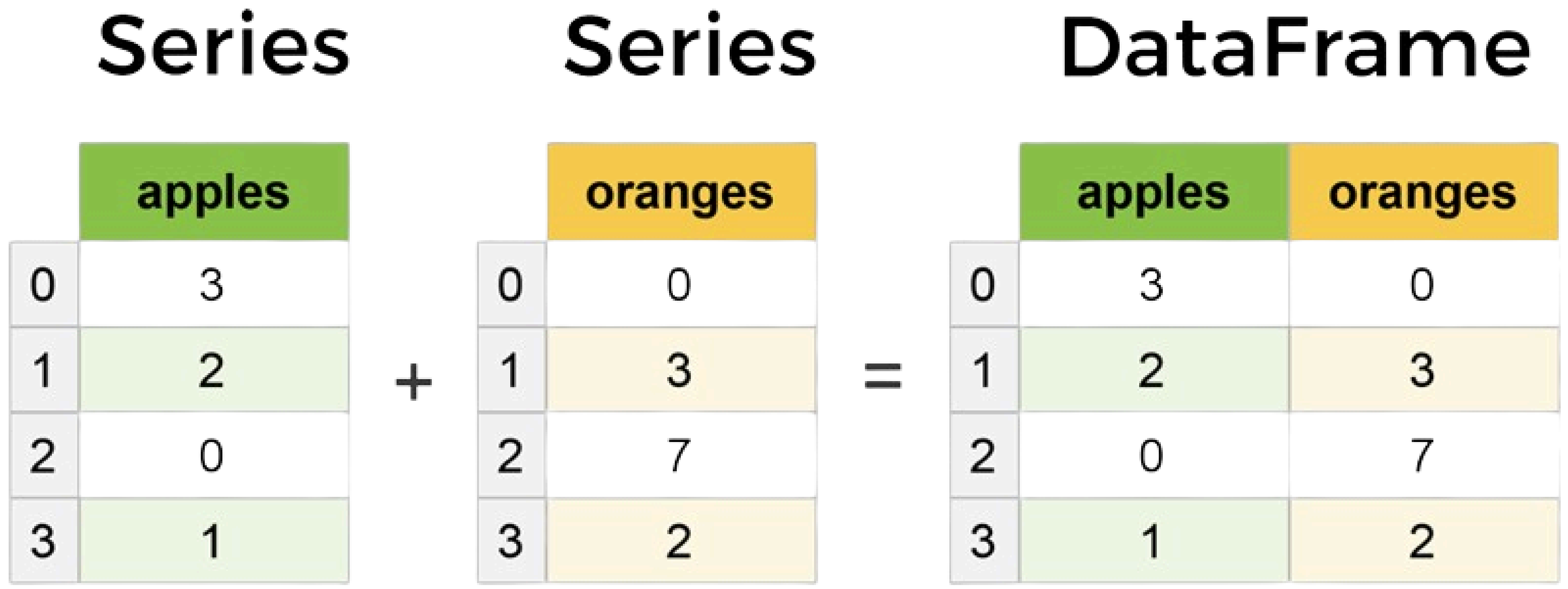
- You can simply use the import keyword to import the pandas library.



```
import pandas as pd
```

## *Main Components of Pandas: Series and DataFrames*

- Panda consists of two main components.
- A Series is like a single column of data.
- A DataFrame is like a big table made of many Series.
- You can do similar things with both, like fill in missing values and find the average.
- Data from CSV files is put into DataFrames.



| | apples |
|---|---|
| 0 | 3 |
| 1 | 2 |
| 2 | 0 |
| 3 | 1 |

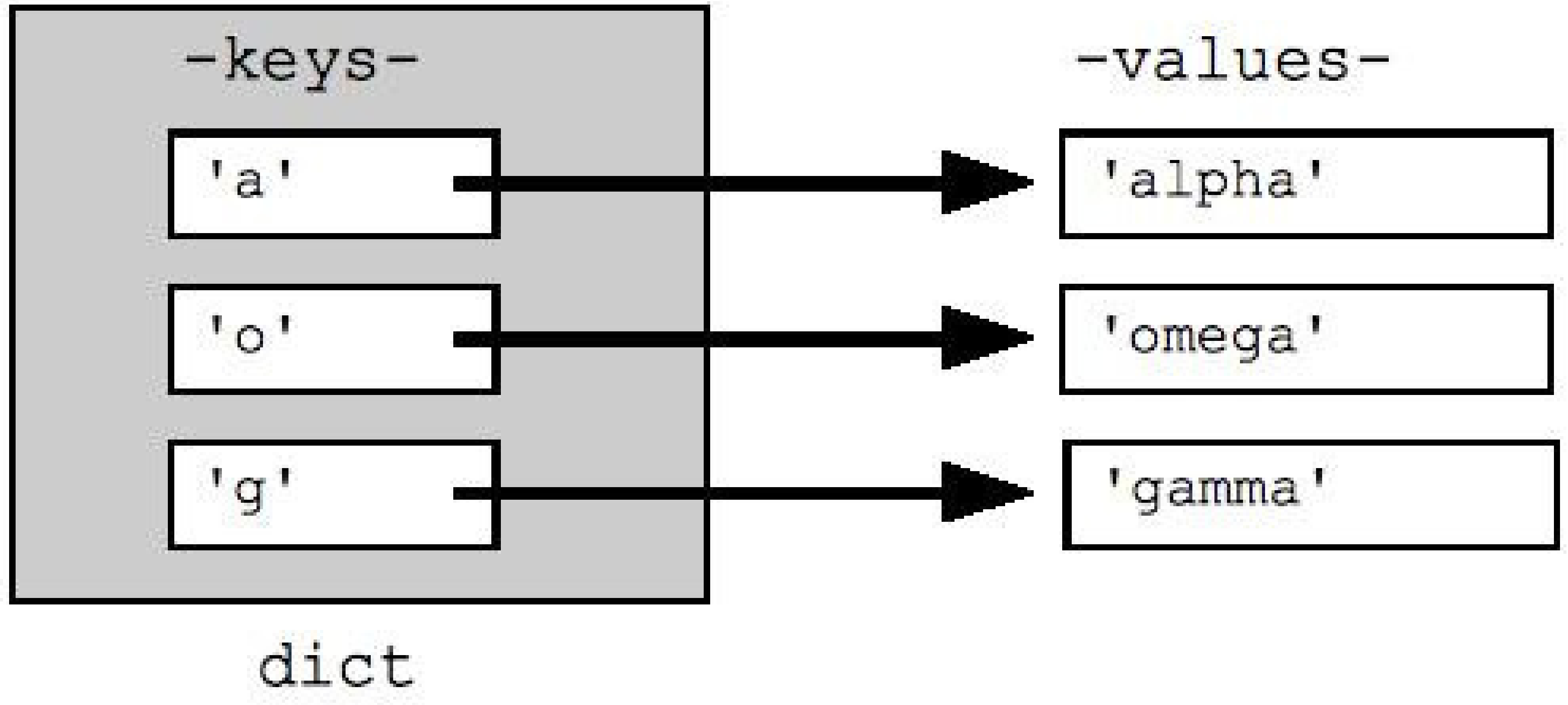
| | oranges |
|---|---|
| 0 | 0 |
| 1 | 3 |
| 2 | 7 |
| 3 | 2 |

| | apples | oranges |
|---|---|---|
| 0 | 3 | 0 |
| 1 | 2 | 3 |
| 2 | 0 | 7 |
| 3 | 1 | 2 |

## *Creating Data Frames*

- Creating DataFrames in Python is useful for testing new things.
- You can make a DataFrame from scratch using a simple dictionary.
- Each key-value pair in the dictionary becomes a column in the DataFrame.
- The DataFrame's index starts at 0, 1, 2, etc., but you can set your own index.

## Syntax:

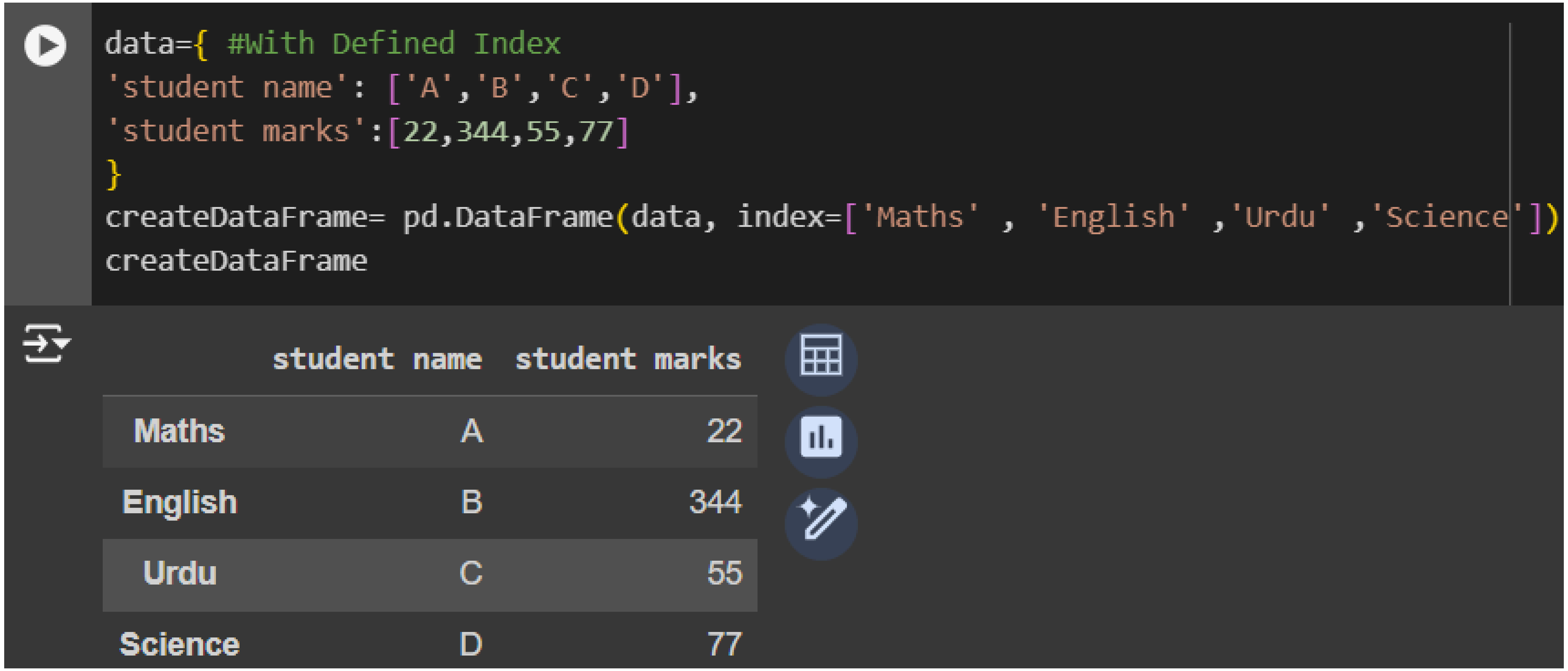


dictionary = {"key_1": "value_1",    "key_2": "value_2",    "key_3": "value_3"}

OR

dictionary = {"key_1": [List_1],    "key_2": [List_2],    "key_3": [List_3]  }

## Example:



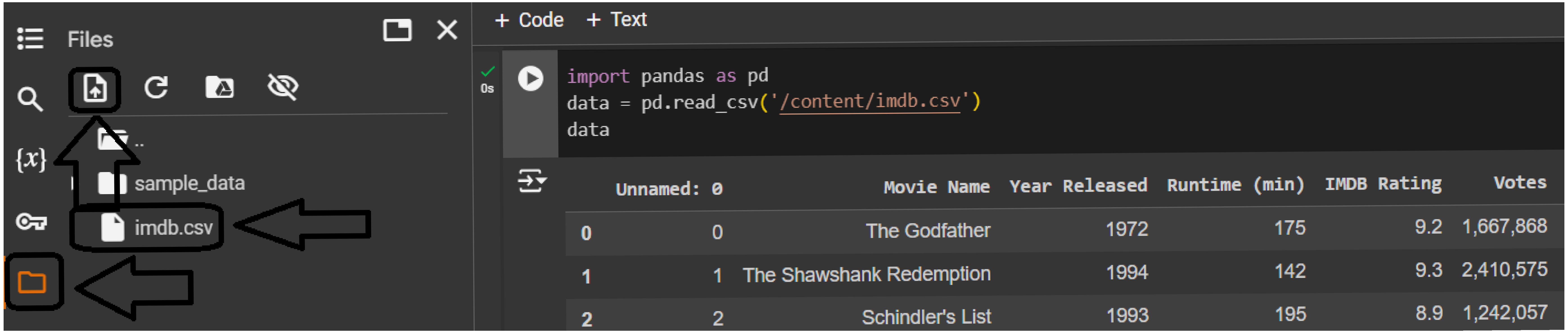
```python
data={ #With Defined Index
'student name': ['A','B','C','D'],
'student marks':[22,344,55,77]
}
createDataFrame= pd.DataFrame(data, index=['Maths' , 'English' ,'Urdu' ,'Science'])
createDataFrame
```

```
         student name  student marks
Maths               A             22
English             B            344
Urdu                C             55
Science             D             77
```

## Uploading Files into Colab:

- For uploading files to colab, click on the files icon on the left sidebar.

- Then click on the upload button and upload your file.

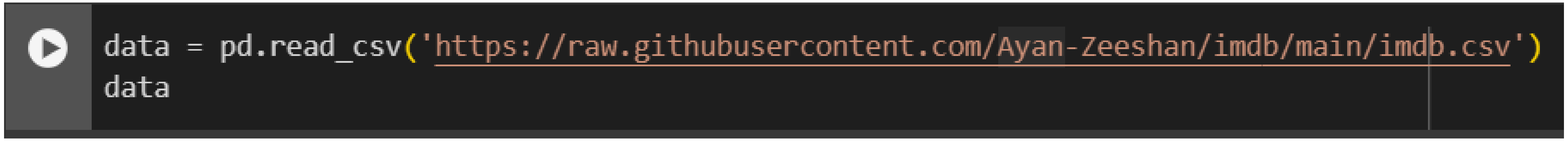


```
import pandas as pd
data = pd.read_csv('/content/imdb.csv')
data
```

```
   Unnamed: 0              Movie Name  Year Released  Runtime (min)  IMDB Rating      Votes
0           0           The Godfather           1972            175          9.2  1,667,868
1           1  The Shawshank Redemption         1994            142          9.3  2,410,575
2           2        Schindler's List           1993            195          8.9  1,242,057
```

## Accessing Data directly from Source

- It is also possible to directly access a file right from it's source using in url or link.



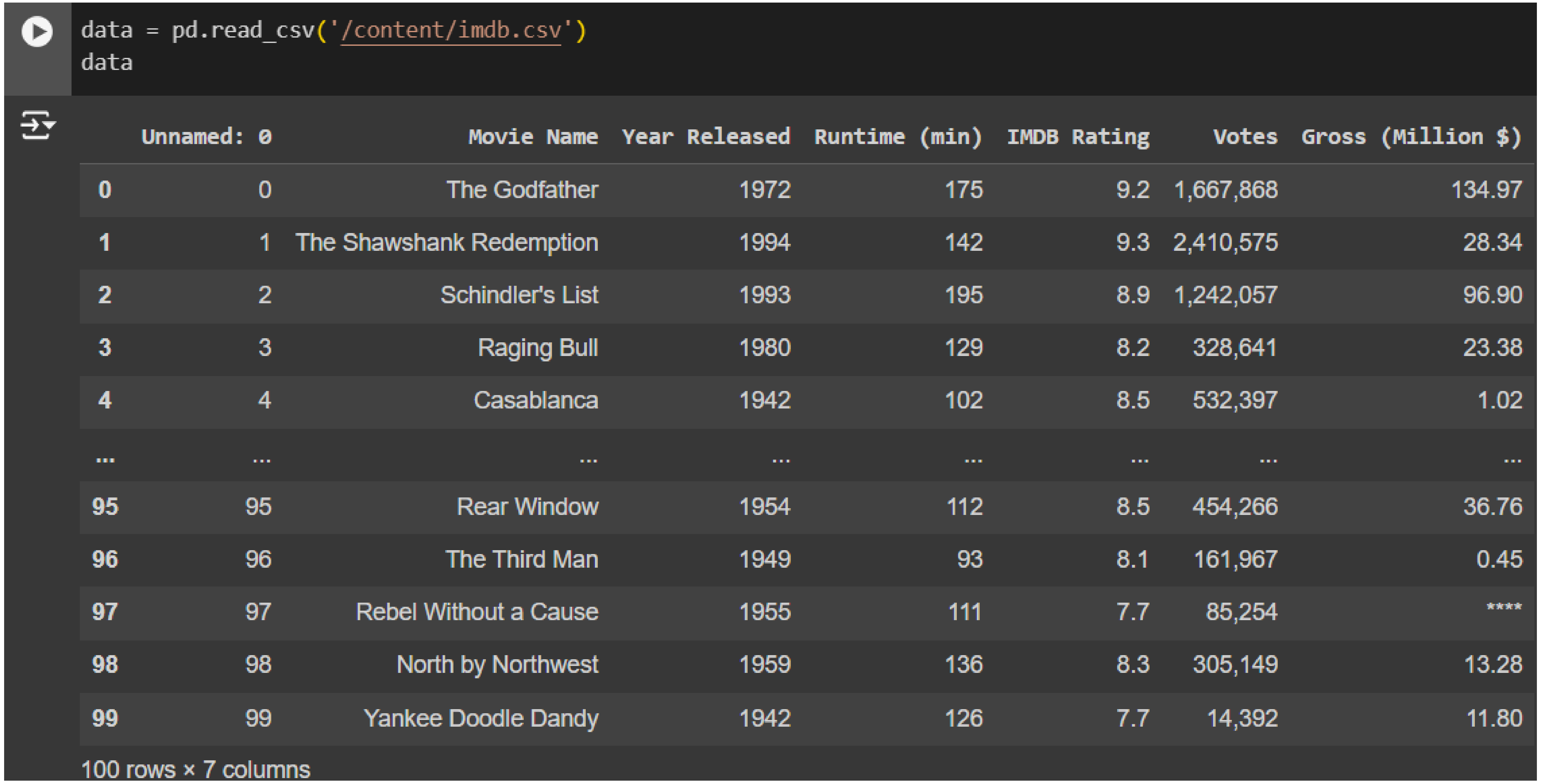
```
data = pd.read_csv('https://raw.githubusercontent.com/Ayan-Zeeshan/imdb/main/imdb.csv')
data
```

## Reading data from csv Files

- It's quite simple to load data from various file formats into a DataFrame.

- With CSV files all you need is a single line to load in the data using file path or simply file name sometimes.

## Examples:



```
data = pd.read_csv('/content/imdb.csv')
data
```

|  | Unnamed: 0 | Movie Name | Year Released | Runtime (min) | IMDB Rating | Votes | Gross (Million $) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | The Godfather | 1972 | 175 | 9.2 | 1,667,868 | 134.97 |
| 1 | 1 | The Shawshank Redemption | 1994 | 142 | 9.3 | 2,410,575 | 28.34 |
| 2 | 2 | Schindler's List | 1993 | 195 | 8.9 | 1,242,057 | 96.90 |
| 3 | 3 | Raging Bull | 1980 | 129 | 8.2 | 328,641 | 23.38 |
| 4 | 4 | Casablanca | 1942 | 102 | 8.5 | 532,397 | 1.02 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | 95 | Rear Window | 1954 | 112 | 8.5 | 454,266 | 36.76 |
| 96 | 96 | The Third Man | 1949 | 93 | 8.1 | 161,967 | 0.45 |
| 97 | 97 | Rebel Without a Cause | 1955 | 111 | 7.7 | 85,254 | **** |
| 98 | 98 | North by Northwest | 1959 | 136 | 8.3 | 305,149 | 13.28 |
| 99 | 99 | Yankee Doodle Dandy | 1942 | 126 | 7.7 | 14,392 | 11.80 |

100 rows × 7 columns

## Pandas Functions

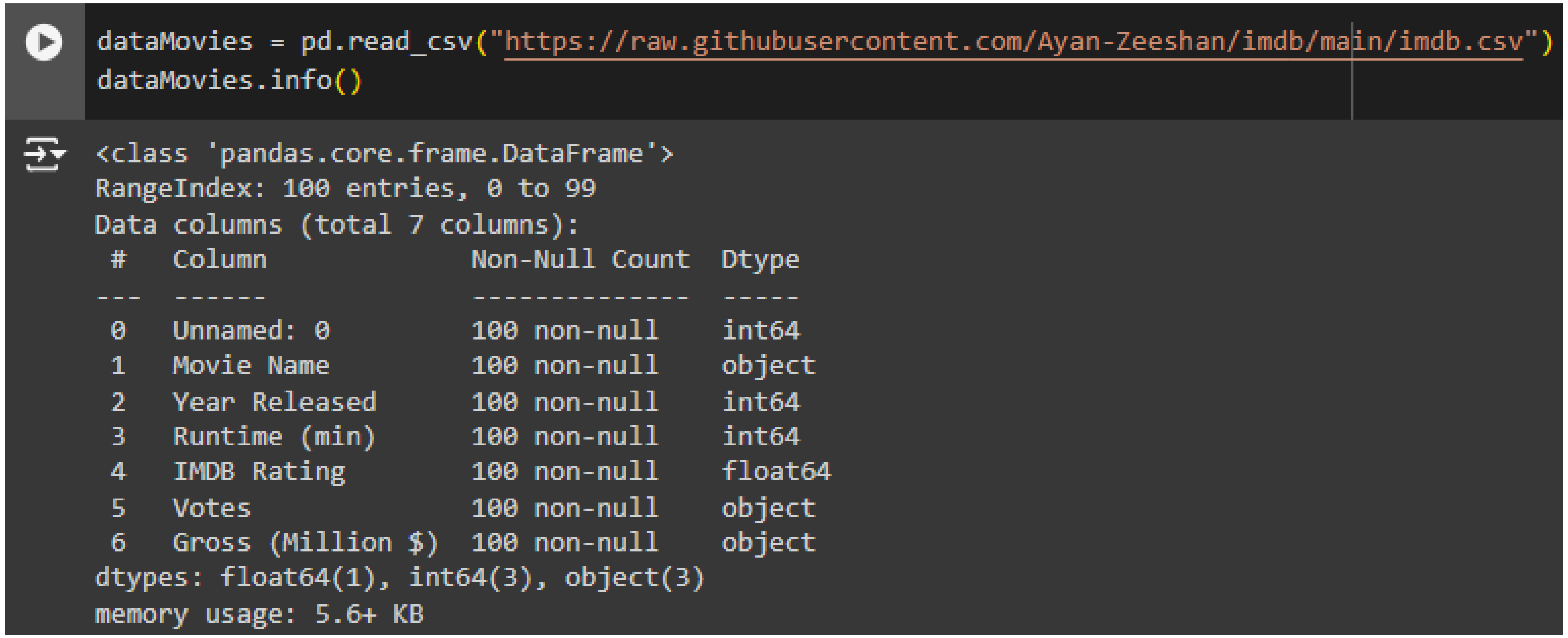
- Functions in pandas are used to manipulate, analyze, and visualize data efficiently in DataFrames.

The following functions are available in pandas:

- .info function
- .sum function
- .mean function
- .head function
- .tail function
- .rename function
- .columns function
- .describe function
- .shape function

## Info Function

- The info() function in pandas shows a summary of the DataFrame.
- It tells you about the columns, data types, and missing values in your data.



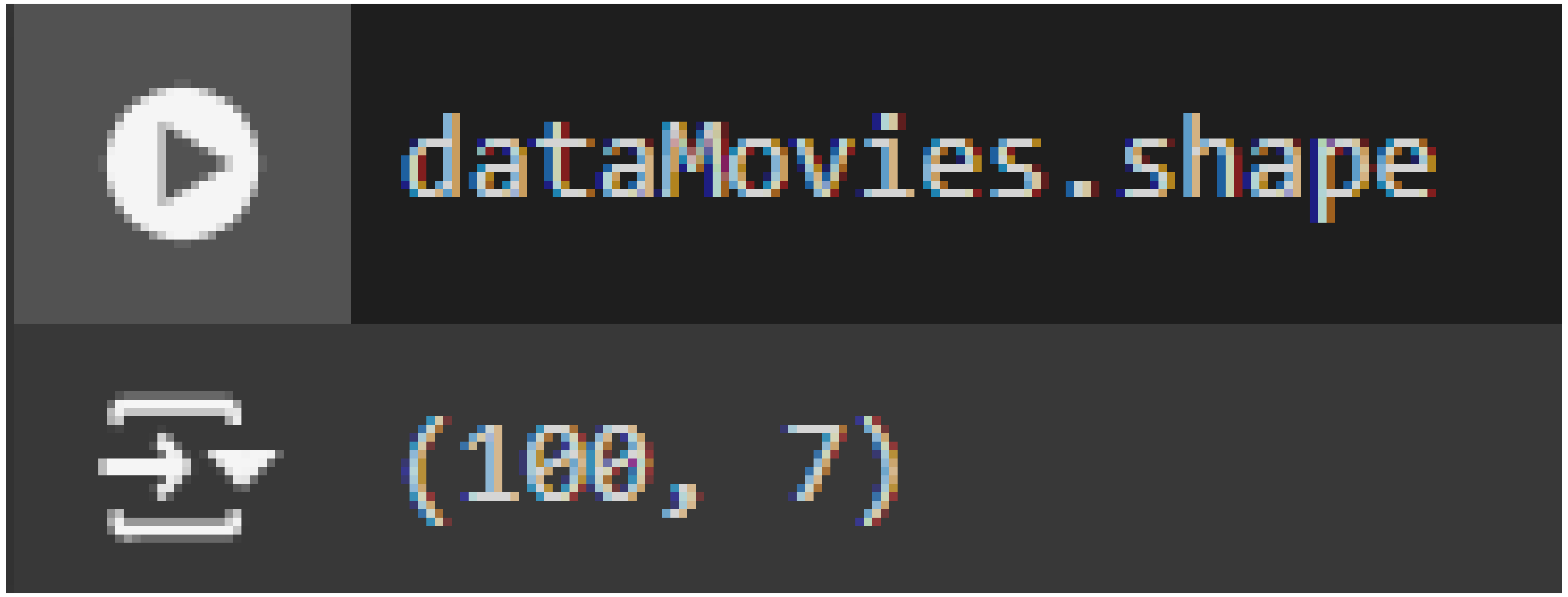
```
dataMovies = pd.read_csv("https://raw.githubusercontent.com/Ayan-Zeeshan/imdb/main/imdb.csv")
dataMovies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         100 non-null    int64  
 1   Movie Name         100 non-null    object 
 2   Year Released      100 non-null    int64  
 3   Runtime (min)      100 non-null    int64  
 4   IMDB Rating        100 non-null    float64
 5   Votes              100 non-null    object 
 6   Gross (Million $)  100 non-null    object 
dtypes: float64(1), int64(3), object(3)
memory usage: 5.6+ KB
```

## *Shape Function*

- The shape function in pandas tells you how many rows and columns are in your data.

- It helps you see the size of your dataset quickly by showing (rows, columns).

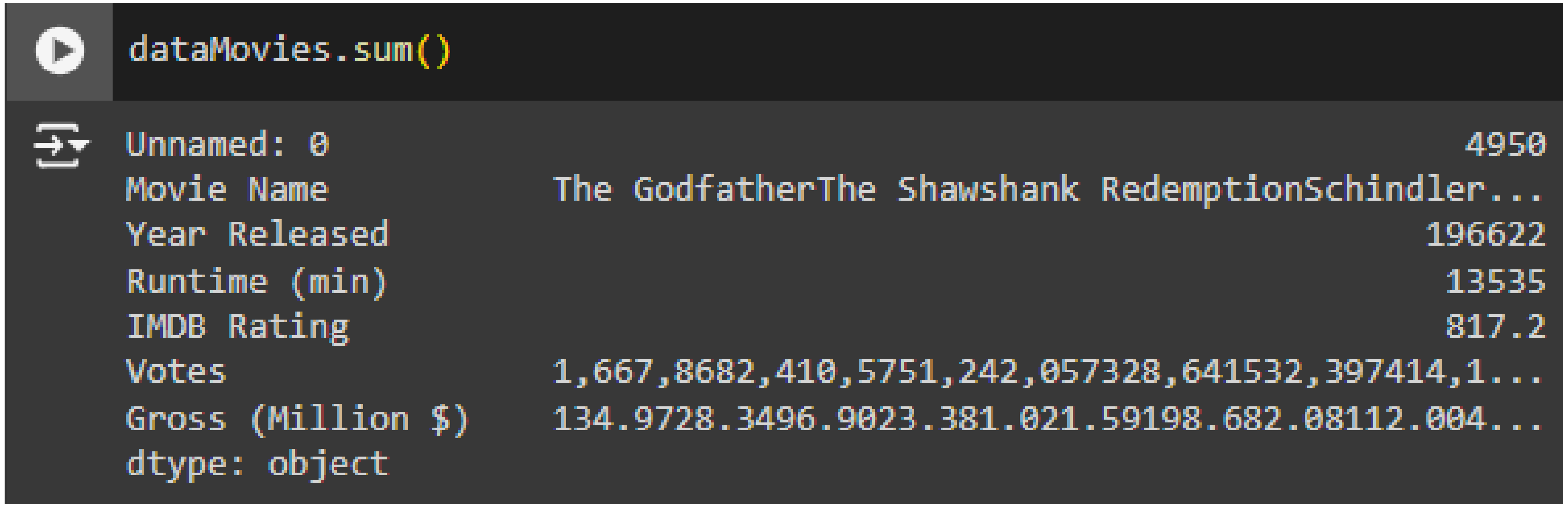


```
dataMovies.shape
```

```
(100, 7)
```

## *Sum Function*

- **The sum function in pandas adds up all the numbers in a column or row of a DataFrame.**
- **It helps you quickly find the total of values like scores or sales in your data.**

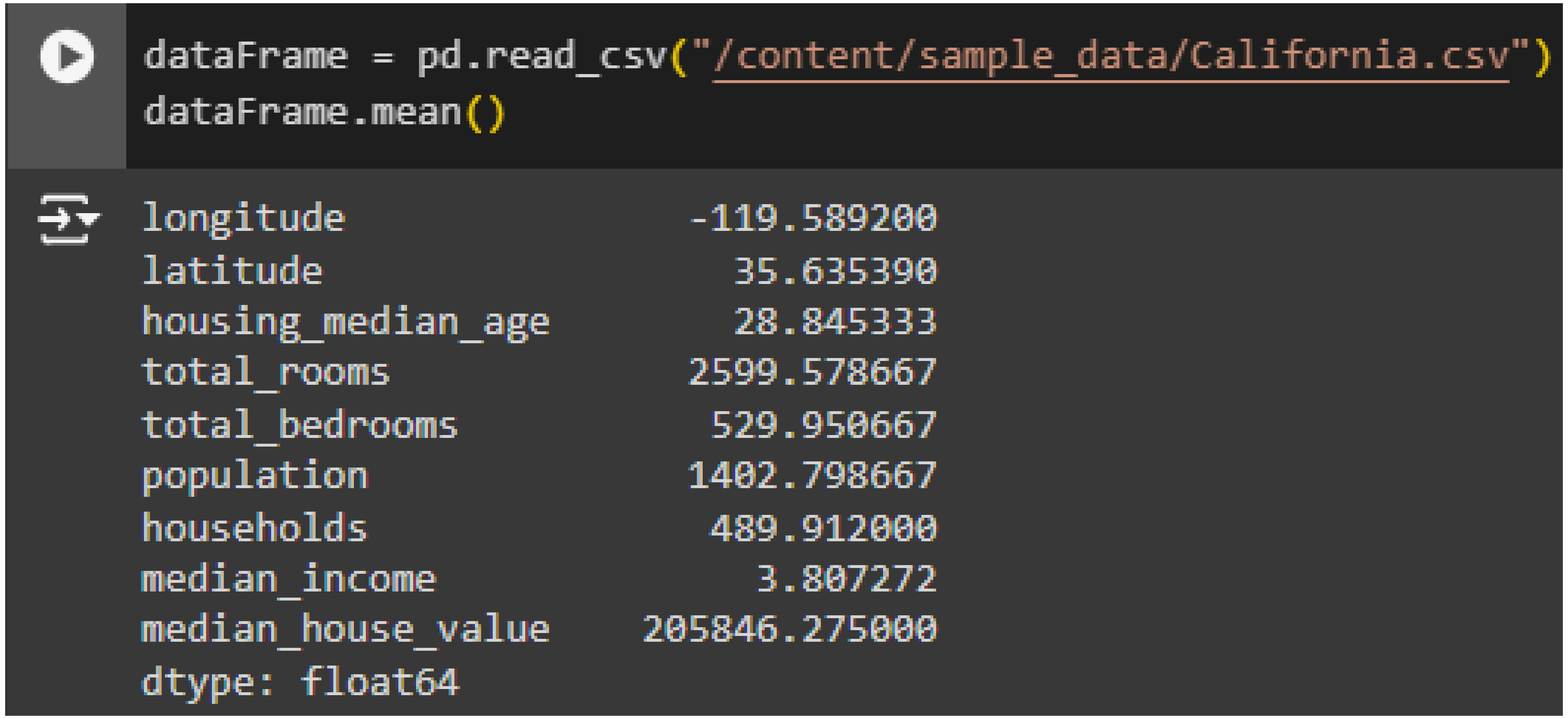


```
dataMovies.sum()
```

```
Unnamed: 0                                                      4950
Movie Name            The GodfatherThe Shawshank RedemptionSchindler...
Year Released                                                 196622
Runtime (min)                                                  13535
IMDB Rating                                                    817.2
Votes                 1,667,8682,410,5751,242,057328,641532,397414,1...
Gross (Million $)     134.9728.3496.9023.381.021.59198.682.08112.004...
dtype: object
```

## Mean Function

- The mean function in pandas calculates the average value of numbers in a column.
- It helps find the central value of data in a DataFrame.



```
dataFrame = pd.read_csv("/content/sample_data/California.csv")
dataFrame.mean()
```

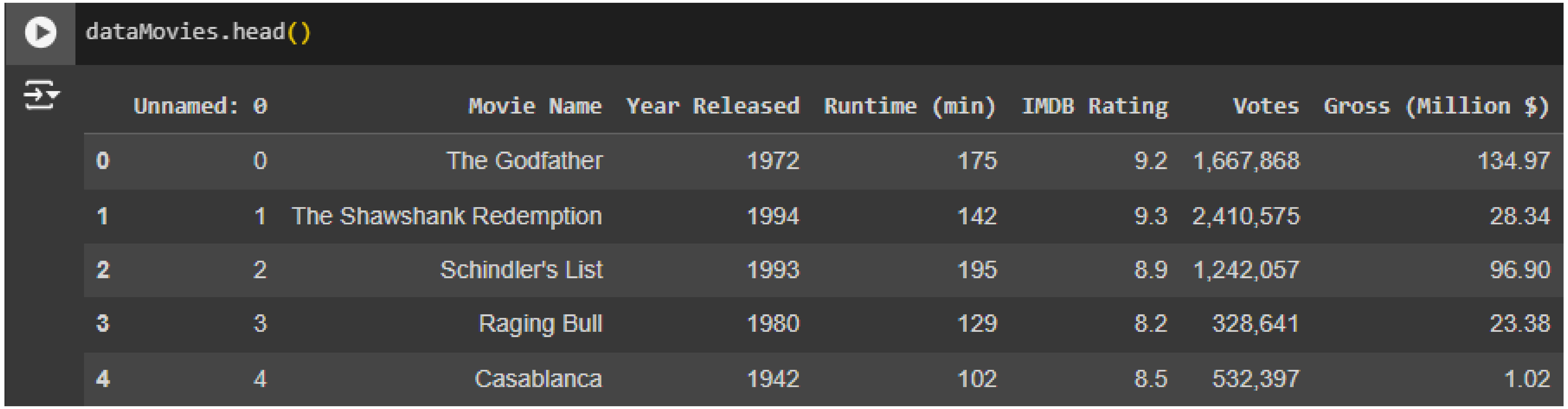
```
longitude                -119.589200
latitude                   35.635390
housing_median_age         28.845333
total_rooms              2599.578667
total_bedrooms            529.950667
population               1402.798667
households                489.912000
median_income               3.807272
median_house_value     205846.275000
dtype: float64
```

## *Head and Tail Functions*

***Head:*** **Shows the first 5 rows of a DataFrame to see the beginning of your data.**
**A number can also be passed.**

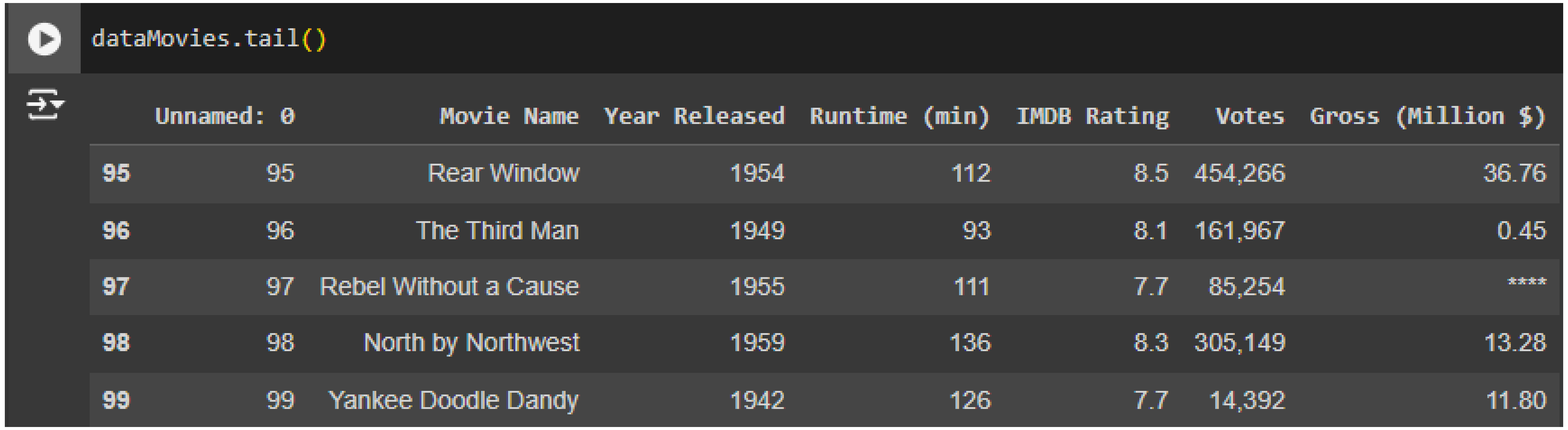
***Tail:*** **Shows the last 5 rows of a DataFrame to see the end of your data.**
**A number can also be passed.**

## Examples:



```
dataMovies.head()
```

```
   Unnamed: 0               Movie Name  Year Released  Runtime (min)  IMDB Rating      Votes  Gross (Million $)
0           0            The Godfather           1972            175          9.2  1,667,868             134.97
1           1  The Shawshank Redemption           1994            142          9.3  2,410,575              28.34
2           2          Schindler's List           1993            195          8.9  1,242,057              96.90
3           3              Raging Bull           1980            129          8.2    328,641              23.38
4           4               Casablanca           1942            102          8.5    532,397               1.02
```



```
dataMovies.tail()
```

```
    Unnamed: 0               Movie Name  Year Released  Runtime (min)  IMDB Rating    Votes  Gross (Million $)
95          95             Rear Window           1954            112          8.5  454,266              36.76
96          96           The Third Man           1949             93          8.1  161,967               0.45
97          97  Rebel Without a Cause           1955            111          7.7   85,254               ****
98          98      North by Northwest           1959            136          8.3  305,149              13.28
99          99      Yankee Doodle Dandy           1942            126          7.7   14,392              11.80
```

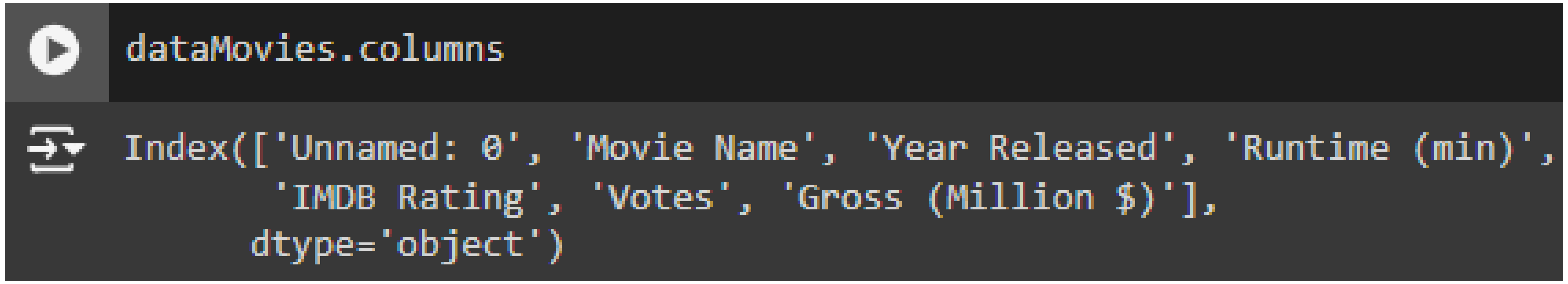
## Rename and Columns Functions

### Rename:

- Renames columns or indexes in a DataFrame.
- Allows changing the names of columns to make them more meaningful.

### Columns:

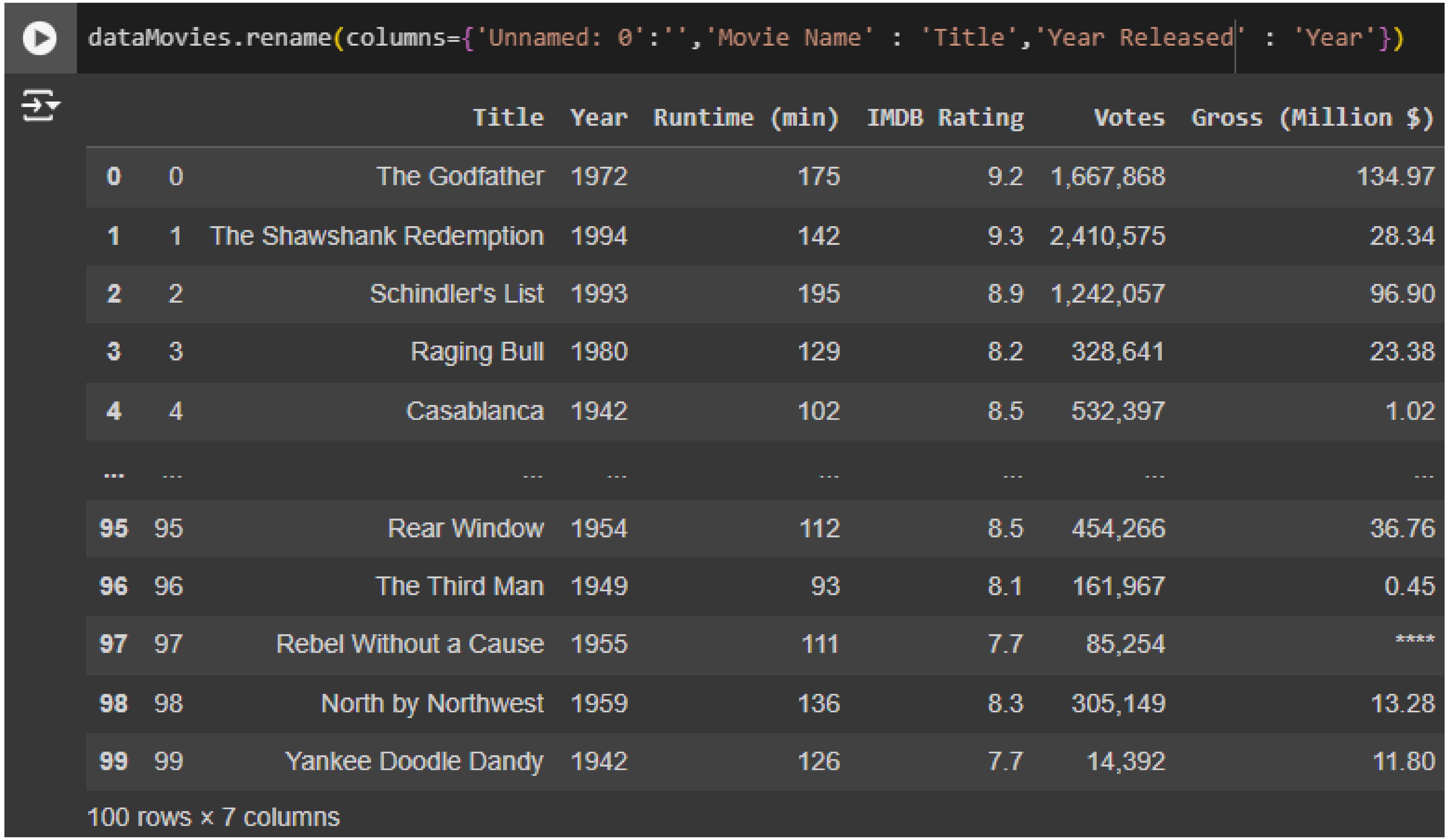
- Returns the names of all columns in a DataFrame.
- Helps to see what columns are available in the DataFrame.



```
dataMovies.columns
```

```
Index(['Unnamed: 0', 'Movie Name', 'Year Released', 'Runtime (min)',
       'IMDB Rating', 'Votes', 'Gross (Million $)'],
      dtype='object')
```

## Example:



```
dataMovies.rename(columns={'Unnamed: 0':'','Movie Name' : 'Title','Year Released' : 'Year'})
```

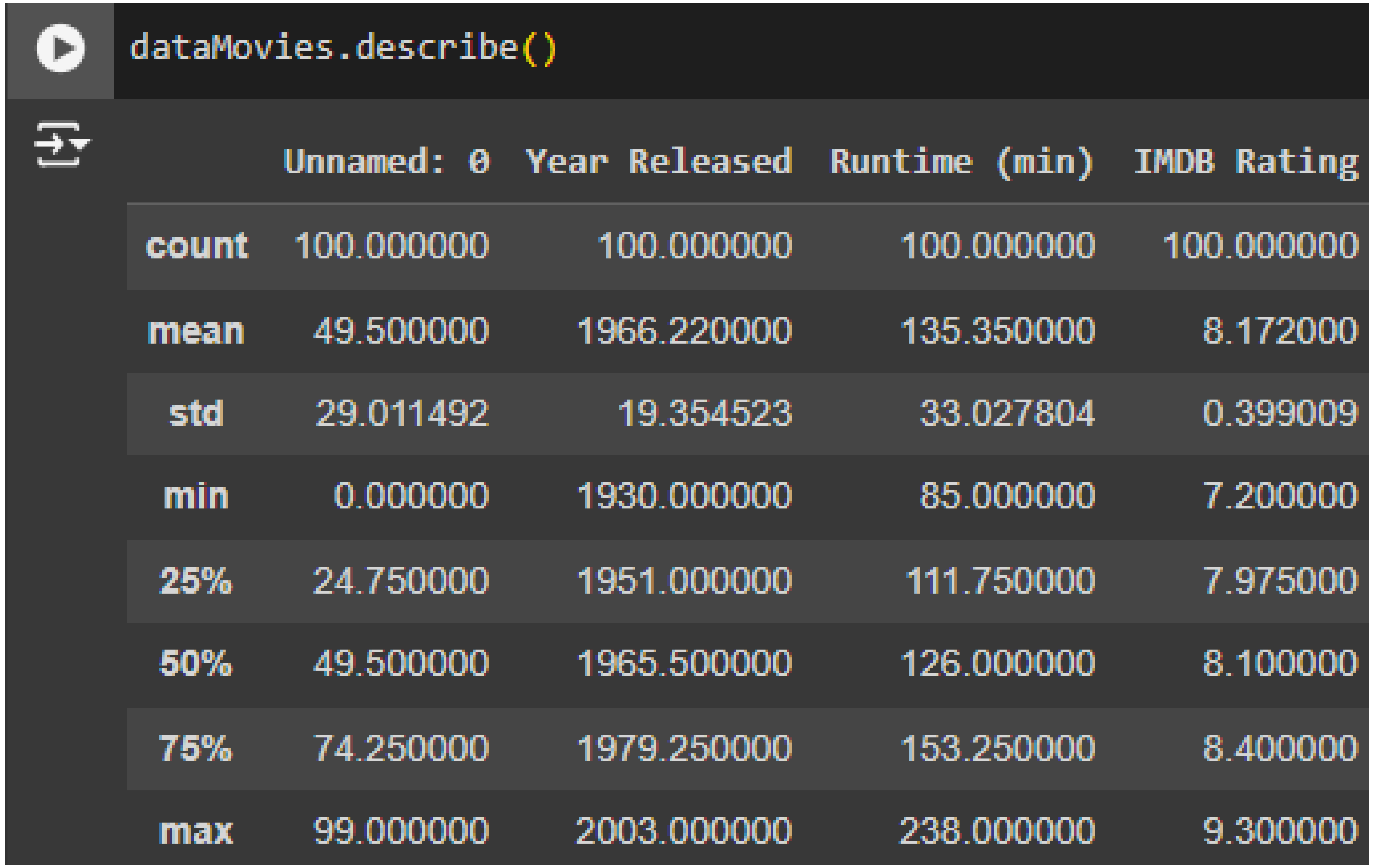
```
                                 Title  Year  Runtime (min)  IMDB Rating      Votes  Gross (Million $)
0    0                   The Godfather  1972            175          9.2  1,667,868             134.97
1    1        The Shawshank Redemption  1994            142          9.3  2,410,575              28.34
2    2                Schindler's List  1993            195          8.9  1,242,057              96.90
3    3                     Raging Bull  1980            129          8.2    328,641              23.38
4    4                      Casablanca  1942            102          8.5    532,397               1.02
...  ...                           ...   ...            ...          ...        ...                ...
95   95                    Rear Window  1954            112          8.5    454,266              36.76
96   96                  The Third Man  1949             93          8.1    161,967               0.45
97   97          Rebel Without a Cause  1955            111          7.7     85,254               ****
98   98             North by Northwest  1959            136          8.3    305,149              13.28
99   99             Yankee Doodle Dandy  1942            126          7.7     14,392              11.80
100 rows × 7 columns
```

## *Describe Function*

- **Describes the basic statistics like mean, median, and minium/maximum values of a DataFrame.**

- **Provides a quick overview of the data, including count, mean, std (standard deviation), min, 25th, 50th (median), and 75th percentiles, and max values.**

## Example:



```
dataMovies.describe()
```

```
       Unnamed: 0  Year Released  Runtime (min)  IMDB Rating
count  100.000000     100.000000     100.000000   100.000000
mean    49.500000    1966.220000     135.350000     8.172000
std     29.011492      19.354523      33.027804     0.399009
min      0.000000    1930.000000      85.000000     7.200000
25%     24.750000    1951.000000     111.750000     7.975000
50%     49.500000    1965.500000     126.000000     8.100000
75%     74.250000    1979.250000     153.250000     8.400000
max     99.000000    2003.000000     238.000000     9.300000
```

# *Task Time !!!*

- Create a DataFrame with information about ten friends: their names, ages, and favorite colors. [ Hint: first create a dictionary then convert it to data frame using pd.DataFrame()
- Use the .info() method to get a concise summary of the DataFrame.
- View the first 3 rows of the DataFrame.
- Find out the shape of the DataFrame.
- Slice and print the first 2 rows of the DataFrame.

***Make it work, make it right, make it fast. – Kent Beck***