**DATAFRAME IN PANDAS**

* Pandas is a powerfull python library for data manipulation,analysis and cleaning data.
* Pandas is a two-dimensional, tabular data structure with labeled axes (rows and columns).
* It is similar to a spreadsheet or SQL table.
* It has two index/axes - row index & column index.

**CREATING & DISPLAYING DATAFRAME**

A DataFrame can be created with the help of following:

1) 2-D Dictionary(having items as (key:value) pair)

2) 2-D ndarray (numpy array)

**DATAFRAME CREATION**

DataFrames can be created from various sources, including dictionaries, lists, NumPy arrays, and external files like CSV or Excel.

**INDEXING**

DataFrames can be accessed using various indexing methods, such as loc – label based indexing and iloc - integer-based indexing

**MISSING DATA**

DataFrames offer methods to handle missing values, such as dropna() to remove NaN values and fillna() to fill missing values with specified data.

**DATA INSPECTION AND EXPLORATION**

**head()**, **tail()**, and **info()** provide quick insights into the data, while **shape** and **dtypes** reveal its structure.

**DATA CLEANING AND HANDLING MISSING VALUES**

Methods like dropna() and fillna() help manage missing data, while techniques address the identification and handling of duplicate values.

**IMPORTING PANDAS**

Import Pandas as Pd

**CREATING A DATAFRAME**

**FROM LISTS**

**#take a 2d array as input to your dataframe**

two\_darray = py.array([[1,2,3],[4,5,6],[7,8,9]])

print(two\_darray)

print(pd.DataFrame(two\_darray))

**OUTPUT:**

[[1 2 3]

[4 5 6]

[7 8 9]]

0 1 2

0 1 2 3

1 4 5 6

2 7 8 9

**FROM DICTIONARY**

**#take a dictionary as input to your dataframe**

dict1 = {

1 : ['1','2'],

2: ['3','4'],

3:['5','6']

}

print(dict1)

print(pd.DataFrame(dict1))

**OUTPUT:**

{1: ['1', '2'], 2: ['3', '4'], 3: ['5', '6']}

1 2 3

0 1 3 5

1 2 4 6

**#take a dataframe as input to your dataframe**

my\_df = pd.DataFrame(data = [1,2,3,4],index = range(0,4),columns = ['A'])

print(my\_df)

**OUTPUT:**

A

0 1

1 2

2 3

3 4

**#take a series as input to your dataframe**

Series = pd.Series ({

'india':'new delhi',

'usa':'washington',

'united states':'america'

})

print((Series))

**OUTPUT:**

india new delhi

usa washington

united states america

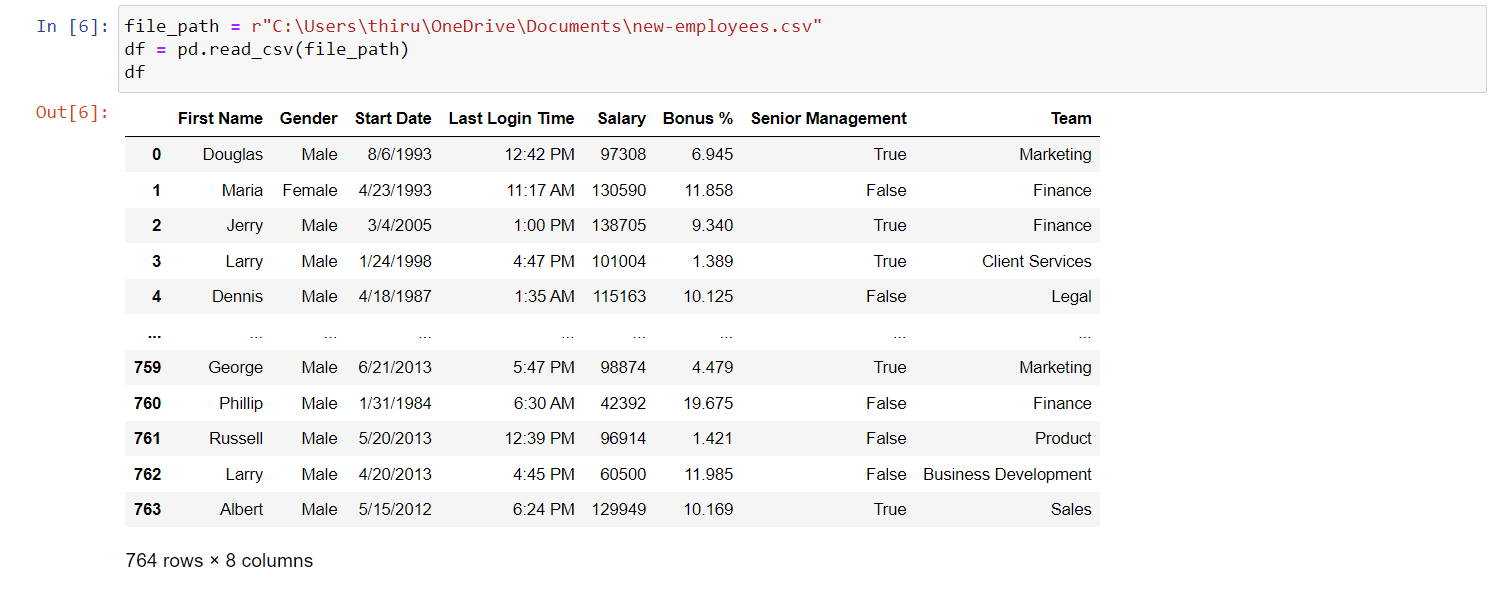
dtype: object

**FROM EXTERNAL FILE (CSV)**

file\_path = r"C:\Users\thiru\OneDrive\Documents\new-employees.csv"

df = pd.read\_csv(file\_path)

df



**VIEWING DATA**

df.head() # Returns the first 5 rows of the DataFrame.

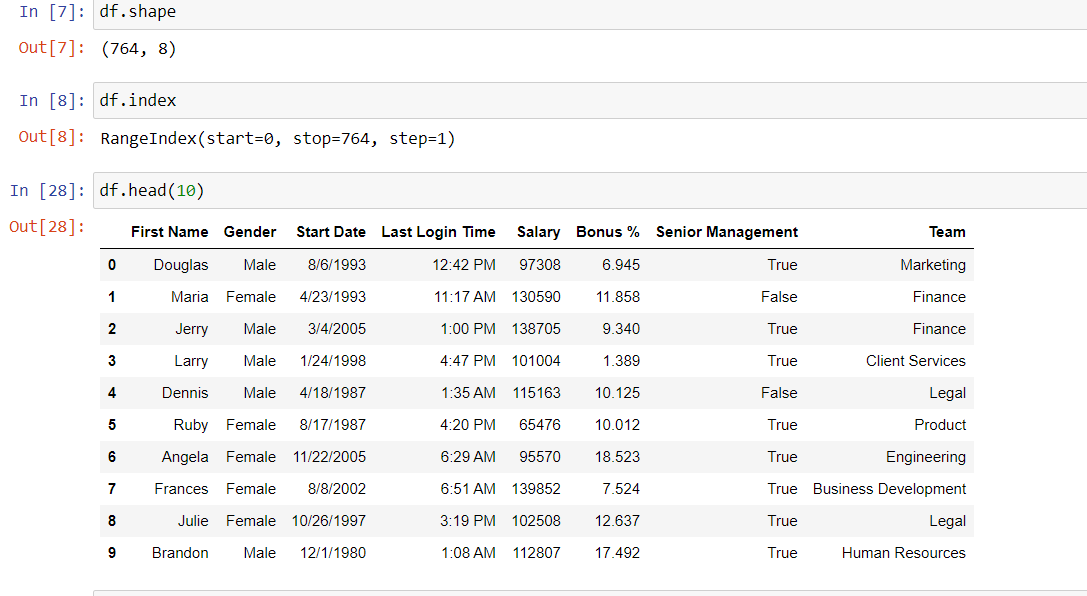
df.tail() # Returns the last 5 rows of the DataFrame.

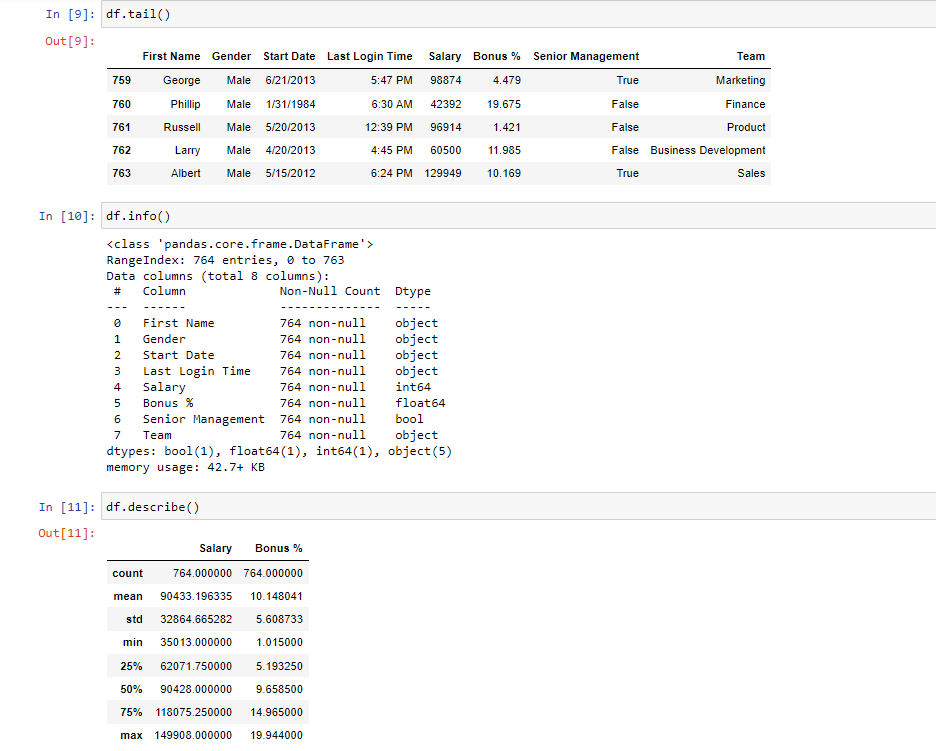
df.info() # Provides concise information about the DataFrame, including data types and non-null values.

df.describe() # Generates descriptive statistics (count, mean, std, min, 25%, 50%, 75%, max) for numerical columns.

df.shape # Returns a tuple representing the dimensions of the DataFrame (number of rows, number of columns).

df.index # Returns the index (row labels) of the DataFrame.





**INDEXING AND SELECTION**

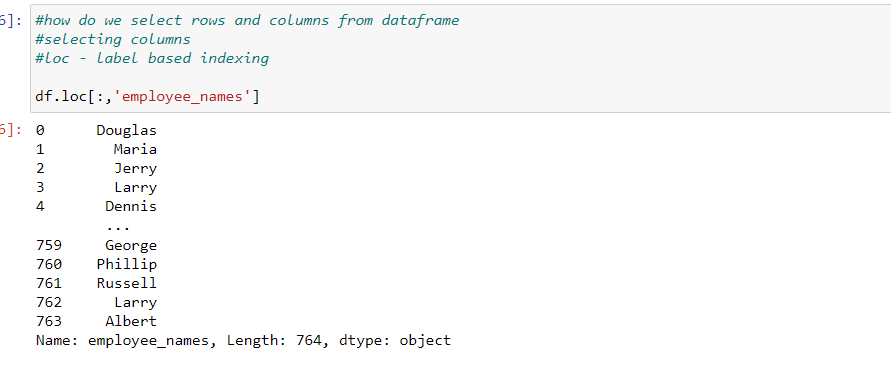
**#how do we select rows and columns from dataframe**

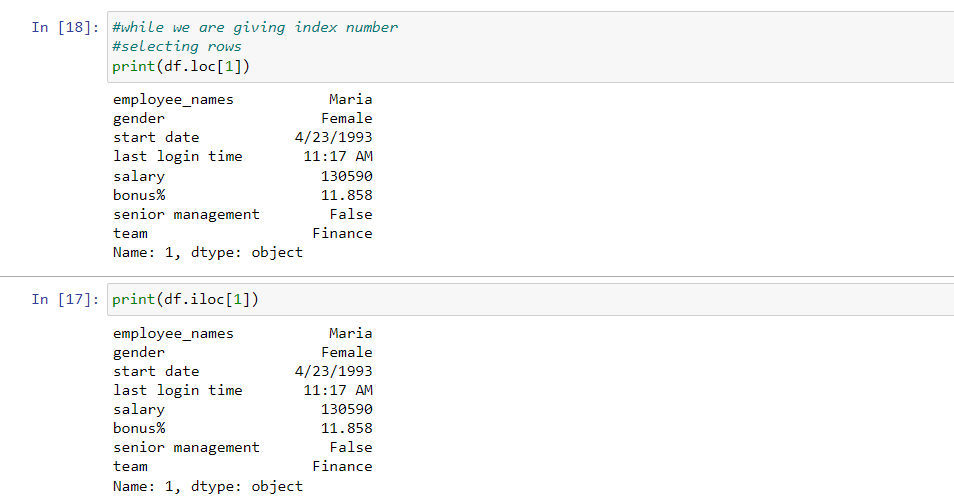
**#selecting columns**

**#loc - label based indexing**

df.loc[:,'employee\_names']

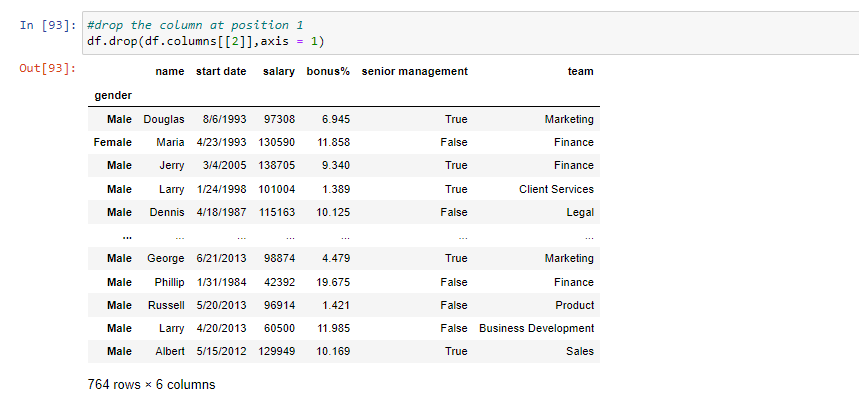
**Output**:

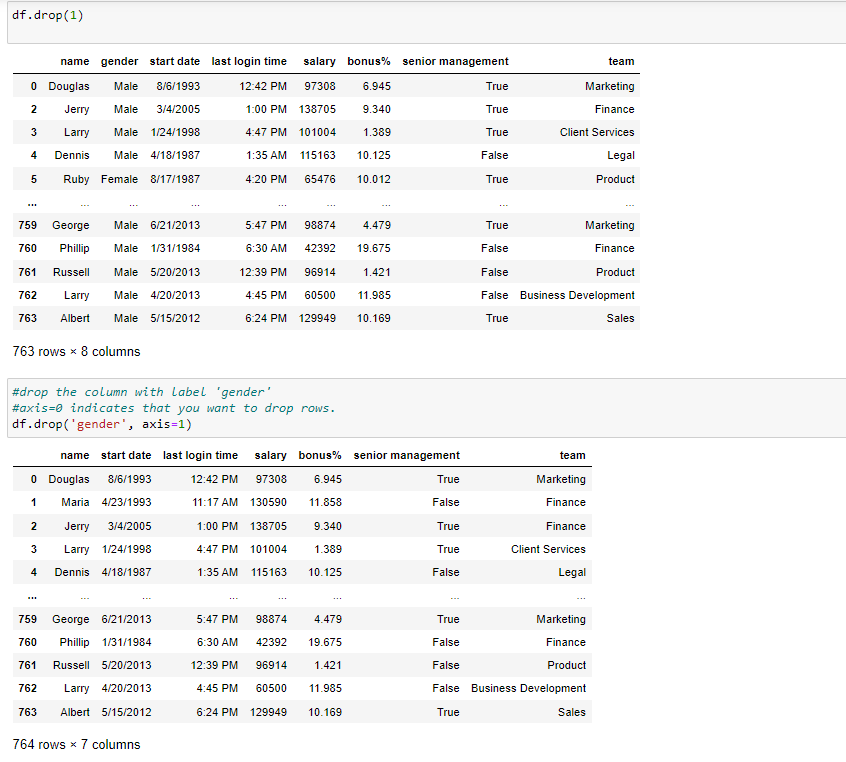




**ADDING AND REMOVING COLUMNS**

* labels: Rows or columns to be removed. It can be a single label or a list of labels.
* axis: Specify 0 to drop rows or 1 to drop columns.
* inplace: If set to True, it modifies the DataFrame directly. If False (default), it returns a new DataFrame with the specified labels removed.





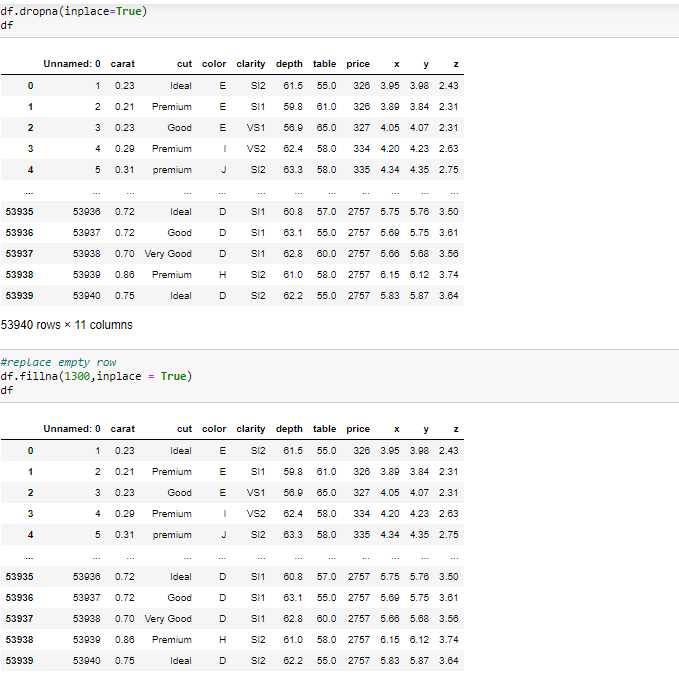
**HANDLING MISSING DATA**

df.dropna() # Drop rows with missing values

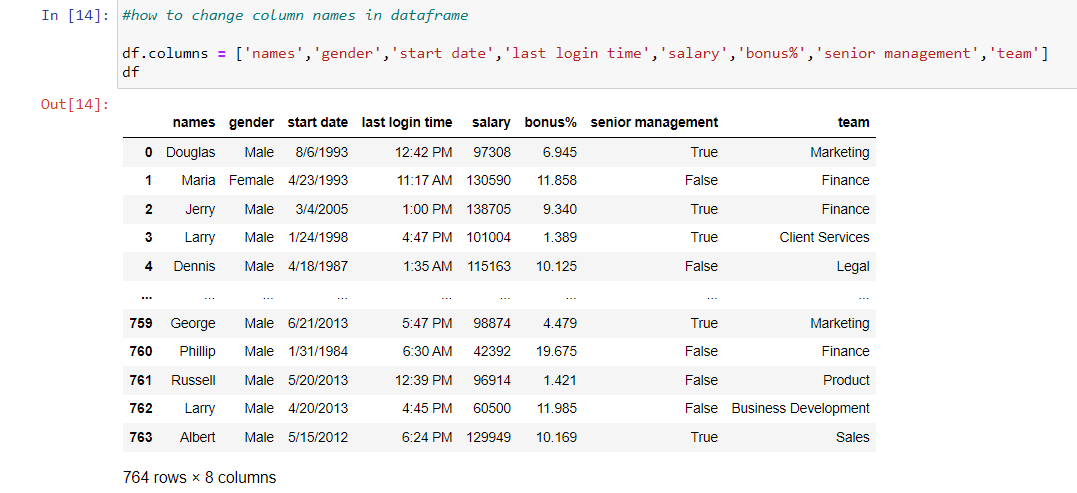
The **dropna** function in pandas is used to remove missing (NaN) values from a DataFrame.

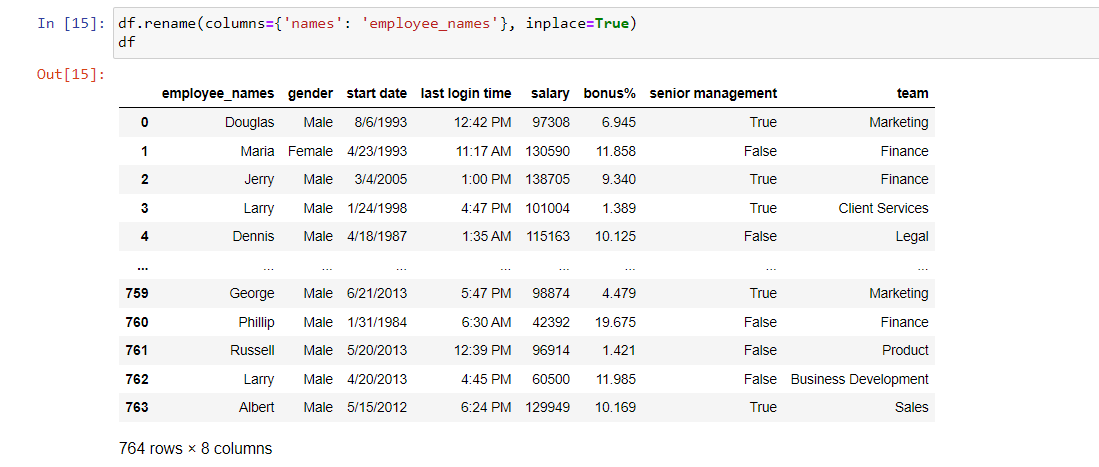
df.fillna(value) # Fill missing values with specified value

The **fillna** function in pandas is used to fill missing (NaN) values in a DataFrame with specified values.

****

**CHANGING COLUMNS**To change column names in a pandas DataFrame, you can use the rename method.

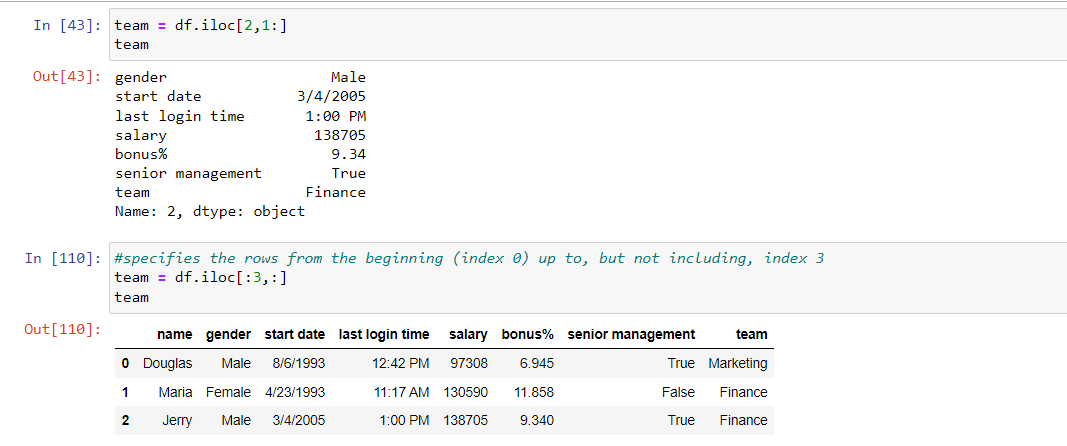




**SILICING A DATAFRAME**

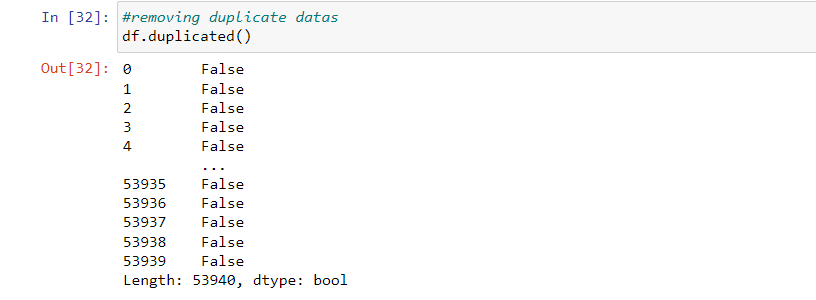
**Combining Row and Column Slicing:**

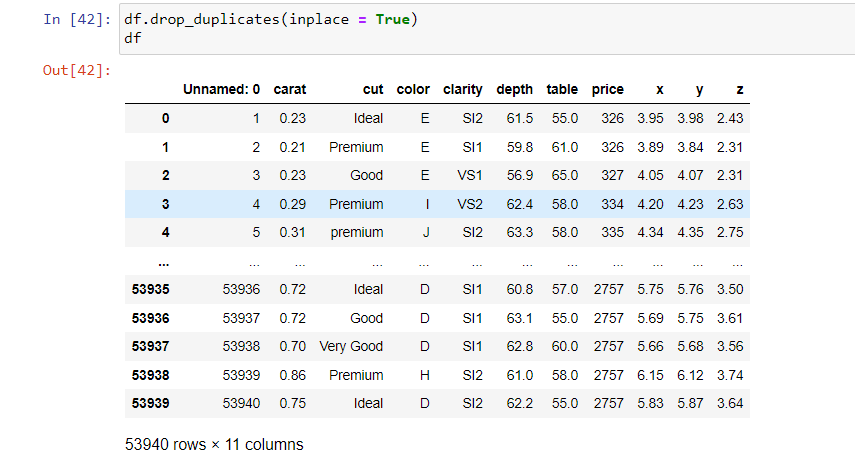
* You can combine both row and column slicing for more specific subsets.



**DROP DUPLICATES**

The **drop\_duplicates** method in pandas to remove duplicate rows from a DataFrame.





**CORRELATION**

The **corr()** function is used to compute the correlation between columns in a DataFrame.

