**The Ultimate Guide to the Pandas Library for Data Science in Python**

The package is known for a very useful data structure called the pandas DataFrame. Pandas also allows Python developers to easily deal with tabular data (like spreadsheets) within a Python script.

*[pandas] is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. — Wikipedia*

### Article Resources:

## What's Pandas for?

This tool is essentially your data’s home. Through pandas, you get acquainted with your data by cleaning, transforming, and analyzing it

1-Load & read the data

2-Cleaning the data

3-Transforming the data to meaningful information.

4- Calculate statistics

5- Clean the data by doing things like removing missing values and filtering rows or columns by some criteria

6- Visualize the data with help from Matplotlib. Plot bars, lines, histograms, bubbles, and more.

7- Store the cleaned, transformed data back into a CSV, other file or database

Note: You should have solid understanding of data

**Shall start for pandas:**

## Pandas First Steps:

### Install and import

Pandas is an easy package to install. Open up your terminal program (for Mac users) or command line (for PC users) and install it using either of the following commands:

pip install pandas(Restart the kernel)

import pandas as pd

The primary two components of pandas are the Series and DataFrame.

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

### Creating DataFrames from scratch

There are *many* ways to create a DataFrame from scratch, but a great option is to just use a simple dict.

data = {

'seq\_1': [3, 2, 0, 1],

'seq\_2': [0, 3, 7, 2]

}

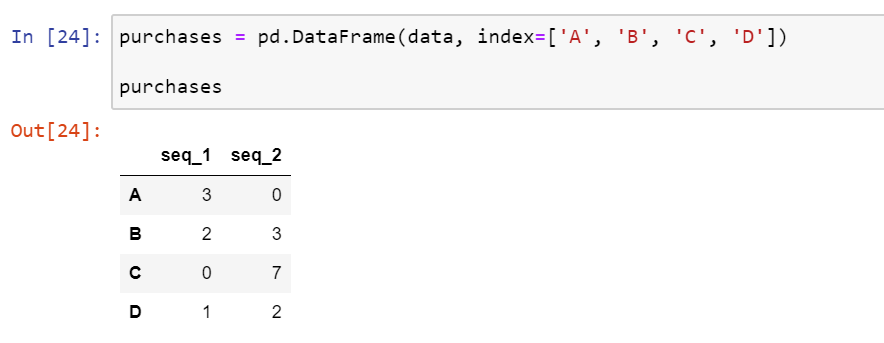
And then pass it to the pandas DataFrame constructor:

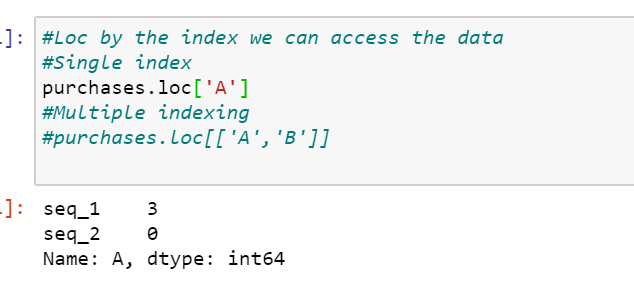
**Python** | **Pandas DataFrame**. **Pandas DataFrame** is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A **Data frame** is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

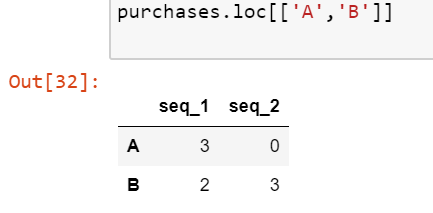
purchases = pd.DataFrame(data)



Changing the index to A B C D





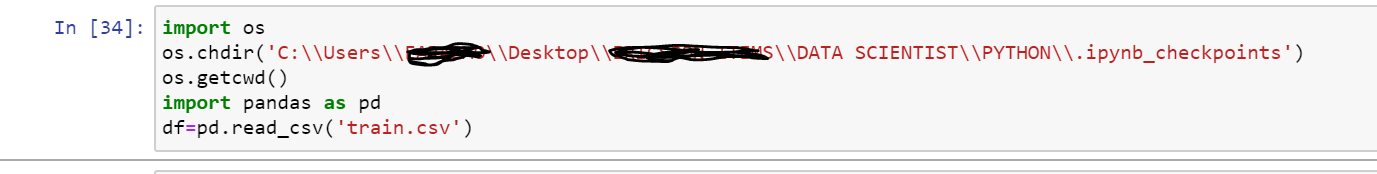


Reading different file format:

We have different kind of file format is there to read it.

>Make sure that Ur desired file should in that current directory otherwise file not found error will come.

Please follow the below steps to read a file without any error.

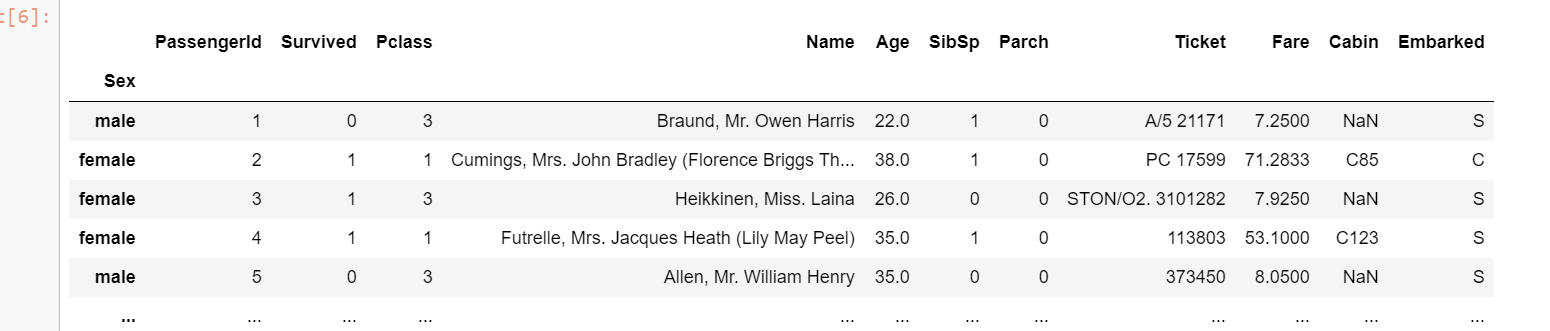


### Converting back to a CSV, JSON, or SQL

df.to\_csv('new\_purchases.csv')

df.to\_json('new\_purchases.json')





Rename the column name:

df.rename(columns={'Parch':"ZERO",'Cabin':"Cabin\_ID"})

df.columns#Dispaly all the columns in a list format

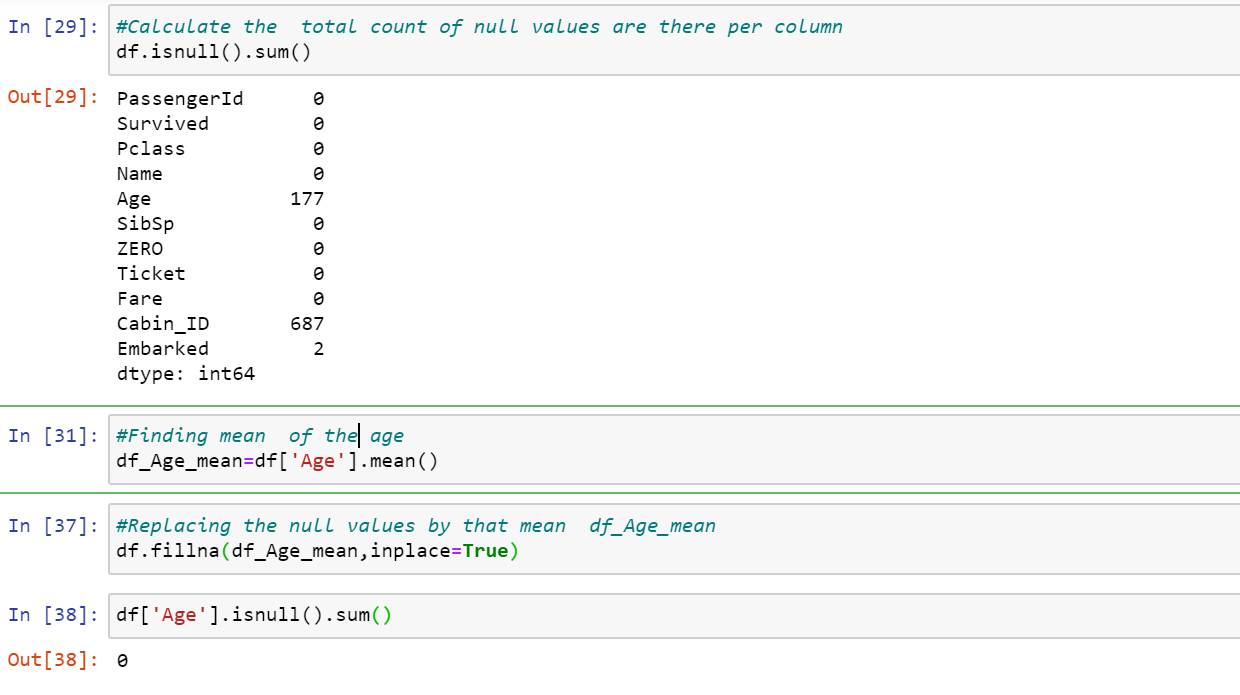
df.rename(columns={'Parch':"ZERO",'Cabin':"Cabin\_ID"},inplace=True) #Inplace will store the modified the changes and stored it in dataframe

### How to work with missing values

There are two options in dealing with nulls:

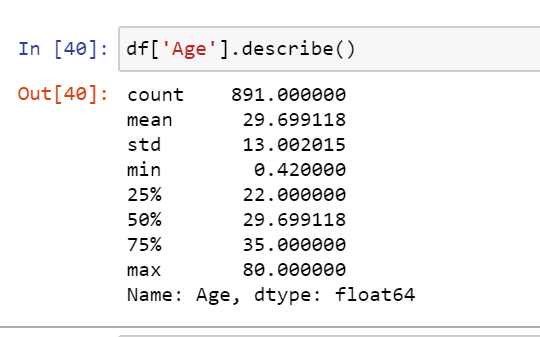
1. Get rid of rows or columns with nulls
2. Replace nulls with non-null values, a technique known as **imputation**

Let's calculate to total number of nulls in each column of our dataset. The first step is to check which cells in our DataFrame are nul



Using describe() on an entire DataFrame we can get a summary of the distribution of continuous variables:

.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:

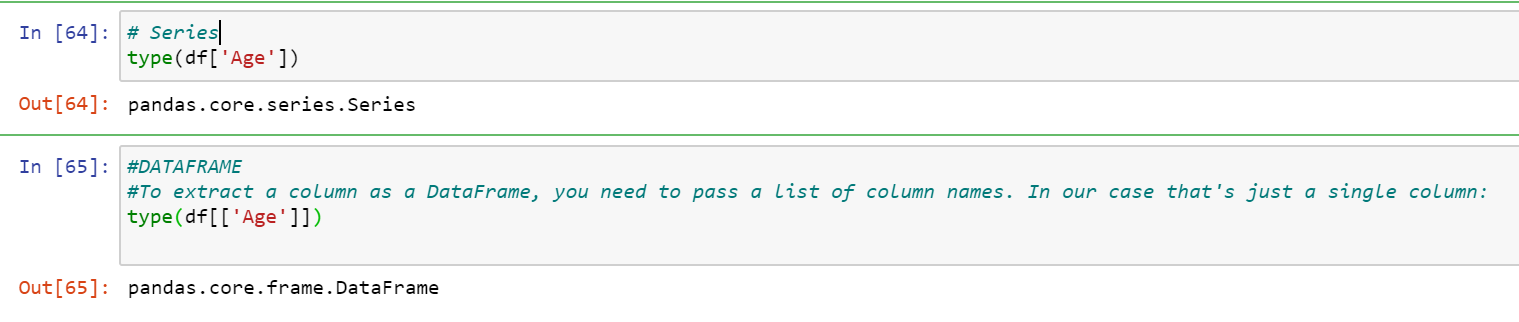


#### 

#### Relationships between continuous variables

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

Series vs DataFrame



### DataFrame slicing, selecting, extracting

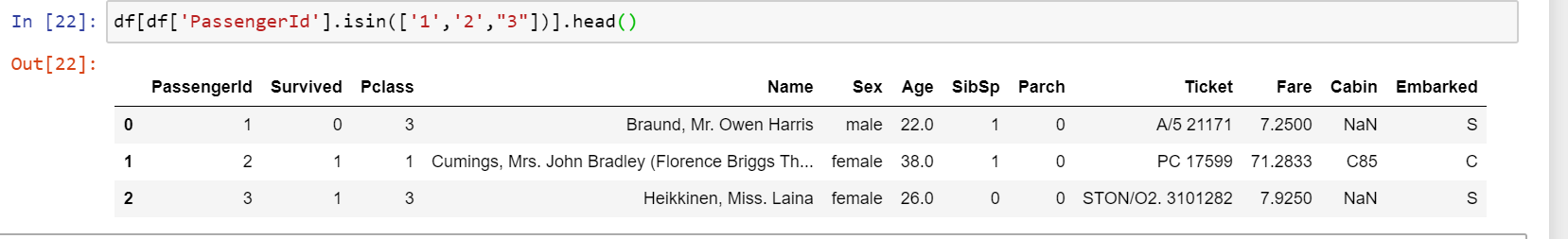


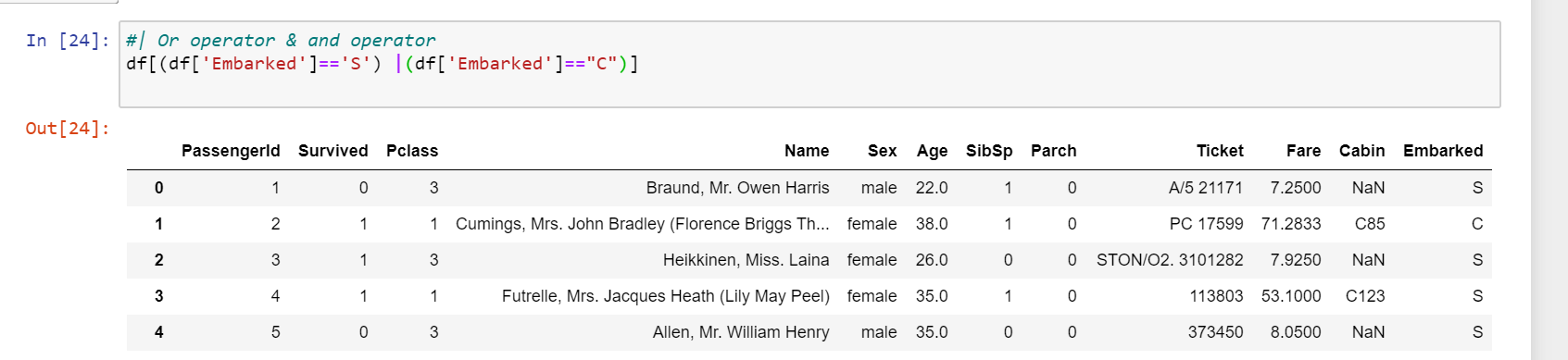
* loc gets rows (or columns) with particular *labels* from the index.
* iloc gets rows (or columns) at particular *positions* in the index (so it only takes integers).
* ix usually tries to behave like loc but falls back to behaving like iloc if a label is not present in the index.

It's important to note some subtleties that can make ix slightly tricky to use:

* if the index is of integer type, ix will only use label-based indexing and not fall back to position-based indexing. If the label is not in the index, an error is raised.
* if the index does not contain *only* integers, then given an integer, ix will immediately use position-based indexing rather than label-based indexing. If however ix is given another type (e.g. a string), it can use label-based indexing.

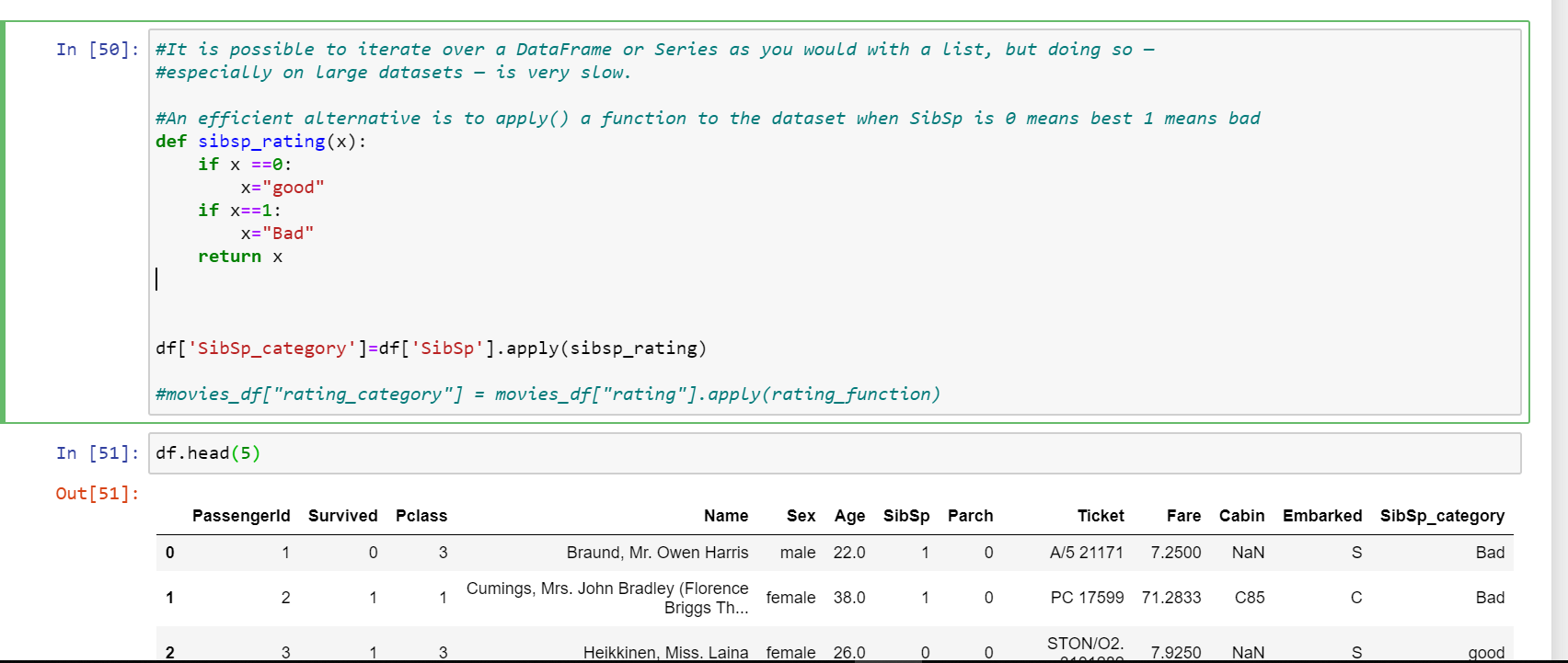
Using the isin() method we could make this more concise though







### Applying functions



using apply() will be much faster than iterating manually over rows because pandas is utilizing vectorization.

*Vectorization: a style of computer programming where operations are applied to whole arrays instead of individual elements —*[*Wikipedia*](https://en.wikipedia.org/wiki/Vectorization)

### Brief Plotting:

Another great thing about pandas is that it integrates with Matplotlib, so you get the ability to plot directly off DataFrames and Series. To get started we need to import Matplotlib (pip install matplotlib):

#### PLOTTING TIP

For categorical variables utilize Bar Charts\* and Boxplots.

For continuous variables utilize Histograms, Scatterplots, Line graphs, and Boxplots.