**Logo, company name

Description automatically generated**

**[pandas - Python Data Analysis Library (pydata.org)](https://pandas.pydata.org/)**

***pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool,  
built on top of the [Python](https://www.python.org/) programming language.***

* Pandas is an open-source Python library designed to deal with data analysis and data manipulation.
* It is built on top of NumPy (a Python library for scientific computing) and it has several functions for cleaning, analyzing, and manipulating data, which can help you extract valuable insights about your data set.
* Pandas is great for working with tabular data, as in SQL tables or Excel spreadsheets.
* The main data structure in Pandas is a 2-dimensional table called DataFrame.
* In Google Colab, Pandas comes pre-installed.
* Pandas can handle [several file extensions.](https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html)
* Coding is **c**ase**S**ensitive.
* data frames are tabular data, containing rows and columns.
* In Pandas, **a single column** can be called **Series**.
* **loc** is used to access rows and columns by label/index or based on a boolean array.
* **iloc** is used to select data based on their integer location or a boolean array.
* .value\_counts() **won’t** consider/count null/nan values
* Histograms - to display the distribution of data.
* Scatter plots - to visualize the relationship between two variables.

**Start via:**

* [Anaconda](https://www.anaconda.com/)
* [Jupyter Notebook](https://jupyter.org/) in the Cloud
* [IBM Watson Studio](https://www.ibm.com/cloud/watson-studio)
* [Google Colab](https://colab.research.google.com/notebooks/intro.ipynb)

**To install Pandas (depends on package manager):**

pip install pandas **(or)** conda install pandas

**Importing pandas library**

**import pandas as pd**

**Reading files and creating a DataFrame +**

**To Display first 1000 rows +**

**To Check the type of the DataFrame +**

**PATH = "https://raw.githubusercontent.com/rmpbastos/data\_sets/main/kaggle\_housing/house\_df.csv"**

**df = pd.read\_csv(PATH)**

**display(df)**

**type(df)** 🡪 pandas.core.frame.DataFrame

**type(df[“LotArea”]) 🡪** pandas.core.series.Series

If we want to create a DataFrame from a Python **Dict**, **List**, **NumPy Array**, or even from another **DataFrame**, we can use the function below:

**df = pd.DataFrame(mydict)**

**Examining the DataFrame**

check the dimensions of our data using the shape attribute. It returns a tuple with the number of rows and columns.

**df.head(1)** 🡪 To check the first n entries on the DataFrame with the function .head(n)

**df.head()** 🡪 If n is not provided in function .head(n), we’ll see the first 5 rows as default

**df['Street'].head(2)** 🡪 To check the first n entries on the DataFrame column with the function .head(n)

**df.tail(1)** 🡪 To check the last n entries on the DataFrame with the function .tail(n)

**df.tail()** 🡪 If n is not provided in function .tail(n), we’ll see the last 5 rows as default

**df['Street'].tail(2)** 🡪 To check the last n entries on the DataFrame column with the function .tail(n)

**df.shape** 🡪 To check the dimensions of our data using the shape attribute

**df.info()** 🡪It provides us the type of the DataFrame with the number of rows and columns, dtypes with number of dtype based columns, memory usage and each columns Non-Null rowcounts with respective column’s Dtypes.

**df.describe()** 🡪 To view descriptive statistics about the dataset - use describe() function. This function displays the count, mean, median, standard deviation, upper and lower quartiles (25%,50%,75%), and minimum and maximum values for each column/feature. It only shows data about the numeric features (columns where the data type is int or float).

**df[“YearBuilt”].value\_counts()** 🡪 To return a Series containing the number of unique values for each column with its number of occurrences.

**DataFrame Index**

After checking our data, we noticed that the first column (**Id**) has a unique value for each row. We can take advantage of it and use this column as our index

**df.set\_index(“Id”, inplace=True)**

**df.index**

When setting the argument **inplace** as True the DataFrame will be updated in place. Otherwise, using **inplace = False, which is the default value**, **would return a copy of the DataFrame.**

If we know beforehand that we are going to use a column in our data set as the index, we can set it up when reading the file

**df = pd.read\_csv(PATH, index\_col=“Id”)**

**df.columns** 🡪 To check the columns of the dataset/DataFrame

**df.rename(columns={"BedroomAbvGr":"Bedroom", "GarageYrBlt":"Garage"}, inplace=True) 🡪** To rename columns.

**df\_copy = df.copy() 🡪** To create a copy of our DataFrame

**df\_copy[“Sold”] = 'N' 🡪** To add a column with default value

**Default Dtype of np.nan is float64**

**df\_copy['column\_new\_1'], df\_copy['column\_new\_2'], df\_copy['column\_new\_3'] = [np.nan, 'dogs', 3] 🡪** To add multiple columns with default values

**df\_to\_append = pd.DataFrame(data\_to\_append)**

**df\_copy = df\_copy.append(df\_to\_append, ignore\_index=True)**

**df\_copy.drop(labels=[1460,1461], axis=0, inplace=True) 🡪** To remove the rows, use .drop() function where axis=0 means row

**df\_copy.drop(labels=["Fence","PoolQC"], axis=1, inplace=True) 🡪** To remove the columns, use .drop() function where axis=1 means column

**index starts from 0 and ends with -1 in python**

**df\_copy.loc[1459]** 🡪 To Select data with .loc method, to bring all columns

**df\_copy.loc[1459,["Neighborhood","SalePrice"]] 🡪** To Select data with .loc method, to bring selected columns

**df\_copy.loc[df["SalePrice"] >= 600000, ["Neighborhood","SalePrice"] ] 🡪** To Select data with .loc method, to apply filter condition.

where ,["Neighborhood","SalePrice"] --> second parameter is optional. if not mentioned, then it will bring all columns.

**df.iloc[0,0]** 🡪 To select the data contained in the first row and the first column

**df.iloc[10,:]** 🡪 To select the data contained in an entire row

**df.iloc[:,-1]** 🡪 To select the data contained in the last column

**df.iloc[8:12, 2:5]** 🡪 To select multiple rows and columns

**Handling missing values**

**df.isnull()** 🡪 To Detect missing values.

This function will return an object with the same size as the original DataFrame, containing boolean values for each element in the set.

It will consider as **True** values such as **None** and **NumPy.NaN**.

**df.isnull().sum()** 🡪 To Detect the number of missing values under each column

**df.shape[0], df.shape[1]** 🡪 To get the number of rows or/and columns of the dataset/DataFrame

**(df.isnull().sum() / df.shape[0])\*100** 🡪 To check the proportion of missing values for each column

**To get only the columns with missing values and display the percentage of values that are missing**

**for column in df.columns:**

**if df[column].isnull().sum() > 0:**

**print(column, ': {:.2%}'.format(df[column].isnull().sum() /**

**df[column].shape[0]))**

**To Remove missing values**

**df\_toremove = df.copy()**

**df\_toremove.dropna(subset=[“GarageType“], axis=0, inplace=True)**

**df\_toremove.shape[0]**

**To Fill the missing values with default value**

**df\_tofill = df.copy()**

**df\_tofill[“Fence“].value\_counts()**

**df\_tofill[“Fence“].fillna(value=“NoFence“, inplace=True)**

**df\_tofill[“Fence“].value\_counts()**

**To Fill the missing values with median value**

**garage\_median = df\_tofill["Garage"].median()**

**df\_tofill.fillna({"GarageYrBlt": garage\_median}, inplace=True)**

**Visualizing data**

2 other Python libraries

* [Matplotlib](https://matplotlib.org/)
* [seaborn](https://seaborn.pydata.org/)

**df[“SalePrice“].plot(kind=“hist“)**

**df.plot(x="SalePrice", y="YearBuilt", kind="scatter")**

**Scatter plots** - is built using **cartesian coordinates** to display the values as a collection of points, each having the value of one variable determining the position on the x-axis and the value of the other variable determining the position on the y-axis.

The **.plot()** function also supports many other kinds of charts, such as line, bar, area, pie, etc.

**df.to\_csv('C:/pandas/My\_DataFrame.csv')** 🡪 To save our DataFrame as a file

**References:**

* [Boost your Data Analysis with Pandas | by Rafael Bastos | Aug, 2021 | Towards Data Science](https://towardsdatascience.com/boost-your-data-analysis-with-pandas-69c4be5d73bb)