Pandas :

explore data , manipulate data and get it ready for Data Analysis

Why Pandas ?

> Simple to use
> Integrated with many other data science & MI python tools
> Helps you get data ready for machine learning

What are we going to cover in this section :

> Most useful functions
> Pandas Datatypes
> Importing & exporting data
> Describing data ## statically
> Viewing & selecting data
> Manipulating data

In [1]:
```
import pandas as pd ##importing Pandas in jupyter notebook
```

In [2]:
```
# 2 main data Types
series = pd.Series(["BMW","HONDA","AUDI"]) ## here after pd.(Series "S" should be capital)
```

In [3]:
```
series ## series is a 1D <column>
```

Out[3]:
```
0      BMW
1    HONDA
2     AUDI
dtype: object
```

In [4]:
```
colours = pd.Series(["RED","Yellow","BLUE"])
colours
```

Out[4]:
```
0       RED
1    Yellow
2      BLUE
dtype: object
```

In [5]:
```
#DataFrame is 2 dimesional
##More oftern used
## It takes a Python Dictionary
car_data = pd.DataFrame({"car make "   : series,"Colour":colours})
car_data
```

Out[5]:

	car make	Colour
0	BMW	RED
1	HONDA	Yellow
2	AUDI	BLUE

In [6]:
```
## Import Data
car_sales = pd.read_csv ("car-sales.csv")
car_sales
#reading car-sales.csv to the variable car_sales
#csv = comma seperated values
#row is reffered to axis  = 0
#column is referred to axis  =1
#the contents of this tables are called Data
```

Out[6]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [7]:
```
#Exporting a DataFrame
car_sales.to_csv("exported-car-sales.csv")
#can also export it to excel by  .to_excel
```

In [8]:
```
ls
```

```
 Volume in drive C is Windows
 Volume Serial Number is 36A3-5FA6

 Directory of C:\Users\test\Desktop\ML_Sample_UdemyCourse\Pandas

11/20/2020  12:41 PM    <DIR>          .
11/20/2020  12:41 PM    <DIR>          ..
11/19/2020  11:32 PM    <DIR>          .ipynb_checkpoints
11/07/2020  04:16 PM           427,861 .jpg
11/03/2020  05:49 PM           332,025 6-step-ml-framework.png
11/19/2020  11:21 PM                56 car_colour.csv
11/04/2020  02:51 PM               369 car-sales.csv
11/18/2020  08:28 PM               392 car-sales-exported.csv
11/06/2020  07:09 PM               287 car-sales-missing-data.csv
```

```
11/04/2020  03:58 PM                 0 cd
11/19/2020  03:42 PM               348 debug.log
11/20/2020  12:41 PM               392 exported-car-sales.csv
11/20/2020  12:33 PM               371 exported-indexFalsed-car-sales.csv
11/20/2020  12:33 PM               392 exported-indexTrue-car-sales.csv
11/03/2020  05:49 PM            11,328 heart-disease.csv
11/05/2020  12:10 PM            19,590 JupyterWalkthrough1.ipynb
11/20/2020  12:41 PM           231,973 Pandas1.ipynb
11/07/2020  05:02 PM           341,236 pandas-anatomy-of-a-dataframe.png
11/19/2020  11:35 PM           106,061 pandas-exercises.ipynb
11/19/2020  11:35 PM           130,908 pandas-exercises-solutions.ipynb
              17 File(s)      1,603,589 bytes
               3 Dir(s)  867,627,593,728 bytes free
```

In [9]:
```
car_sales.to_csv("exported-indexFalsed-car-sales.csv",index = False)
car_sales.to_csv("exported-indexTrue-car-sales.csv",index = True)##adding an index column
# here we are telling python that no need to add an
#extra column for indexing <by default index = False>
```

In [10]:
```
sales_car = pd.read_csv("exported-indexFalsed-car-sales.csv")
sales_car_True = pd.read_csv("exported-indexTrue-car-sales.csv")
```

In [11]:
```
sales_car
```

Out[11]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [12]:
```
sales_car_True
```

Out[12]:

	Unnamed: 0	Make	Colour	Odometer (KM)	Doors	Price
0	0	Toyota	White	150043	4	$4,000.00
1	1	Honda	Red	87899	4	$5,000.00
2	2	Toyota	Blue	32549	3	$7,000.00
3	3	BMW	Black	11179	5	$22,000.00
4	4	Nissan	White	213095	4	$3,500.00
5	5	Toyota	Green	99213	4	$4,500.00
6	6	Honda	Blue	45698	4	$7,500.00
7	7	Honda	Blue	54738	4	$7,000.00
8	8	Toyota	White	60000	4	$6,250.00
9	9	Nissan	White	31600	4	$9,700.00

Getting DataSet From GitHub Repository

heart_disease = pd.read_csv("https://raw.githubusercontent.com/mrdbourke/zero-to-mastery-ml/master/data/heart-disease.csv")
Note: If you're using a link from GitHub, make sure it's in the "raw" format, by clicking the raw button.

Describe Data

In [13]:
```
# Attribute - without parenthesis
# Atributes are just metaInfromation Stored about your Data
sales_car.dtypes
```

Out[13]:
```
Make             object
Colour           object
Odometer (KM)     int64
Doors             int64
Price            object
dtype: object
```

In [14]:
```
sales_car.columns # columns name returned as a list
```

Out[14]:
```
Index(['Make', 'Colour', 'Odometer (KM)', 'Doors', 'Price'], dtype='object')
```

In [15]:
```
sales_car.index #information abour index
```

Out[15]:
```
RangeIndex(start=0, stop=10, step=1)
```

In [16]:
```
sales_car
```

Out[16]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [17]:
```
sales_car.describe()
#This a function as this has parenthesis
#functions performs some task on data
#.describe fucntions returns some statistical information on Data<only on Integer Data Types> i. e it works only on numeric
```

Out[17]:

	Odometer (KM)	Doors
count	10.000000	10.000000
mean	78601.400000	4.000000
std	61983.471735	0.471405
min	11179.000000	3.000000
25%	35836.250000	4.000000
50%	57369.000000	4.000000
75%	96384.500000	4.000000
max	213095.000000	5.000000

In [18]:
```
sales_car.info()
#info is kind of like index combined with dtypes attributes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Make            10 non-null     object
 1   Colour          10 non-null     object
 2   Odometer (KM)   10 non-null     int64 
 3   Doors           10 non-null     int64 
 4   Price           10 non-null     object
dtypes: int64(2), object(3)
memory usage: 528.0+ bytes
```

In [19]:
```
sales_car.mean()
# gives average of numerical columns<integer type>
```

Out[19]:
```
Odometer (KM)    78601.4
Doors                4.0
dtype: float64
```

In [20]:
```
car_prices  = pd.Series([23232,232334,232323,232323]) ## creating a Series Data Type Just to Demonstrate What mean function
car_prices.mean()
```

Out[20]: 180053.0

In [21]:
```
sales_car.sum()
## combined all the columns
```

Out[21]:
```
Make             ToyotaHondaToyotaBMWNissanToyotaHondaHondaToyo...
Colour               WhiteRedBlueBlackWhiteGreenBlueBlueWhiteWhite
Odometer (KM)                                               786014
Doors                                                           40
Price            $4,000.00$5,000.00$7,000.00$22,000.00$3,500.00...
dtype: object
```

In [22]:
```
sales_car["Doors"].sum()
# running the sum function on a particular column
```

Out[22]: 40

In [23]:
```
len(sales_car) #returns the number of rows in your DataFrame
```

Out[23]: 10

Viewing and Selecting Data

In [24]:
```
sales_car
```

Out[24]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [25]:
```
sales_car.tail()#last 5 rows of DataFrame
```

Out[25]:

	Make	Colour	Odometer (KM)	Doors	Price
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [26]:
```
sales_car.head() #top 5 rows of dataFrame
```

Out[26]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00

In [27]:
```
animals = pd.Series(["cat", "parrot", "dog", "snake"],index=[0,3,1,2]) ##defining a Series and custom Index
```

In [27]:
```
animals = pd.Series([ 'cat' , 'pandas' , 'dog' , 'Snake' ],index=[0,2,4,2]) ##defining a series and custom index
```

In [28]:
```
animals
```

Out[28]:
```
0       cat
2    pandas
4       dog
2     Snake
dtype: object
```

In [29]:
```
animals.loc[2] ## loc refers to the index number specifies
# it reurns all the elements with the index specified
```

Out[29]:
```
2    pandas
2     Snake
dtype: object
```

In [30]:
```
sales_car.loc[3]
```

Out[30]:
```
Make                   BMW
Colour               Black
Odometer (KM)        11179
Doors                    5
Price          $22,000.00
Name: 3, dtype: object
```

In [31]:
```
sales_car
```

Out[31]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [32]:
```
animals
```

Out[32]:
```
0       cat
2    pandas
4       dog
2     Snake
dtype: object
```

In [33]:
```
#.iloc
# iloc refers to position or you can say original index or
#starting from 0 it returns the 2nd positioned item
animals.iloc[2]
```

Out[33]:
```
'dog'
```

In [34]:
```
animals.iloc[:2]##this is called slicing
#i.e,we specify the starting index : and the ending index and python returns all the index betwween them
# : 2 means starting from 0 i.e, from the first (called 0 during indexing) row  return till the second <upto position 2>
```

Out[34]:
```
0       cat
2    pandas
dtype: object
```

In [35]:
```
sales_car.loc[:2]
```

Out[35]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00

In [36]:
```
sales_car["Colour"]
#sales_car.Colour both do the same task i.e, selecting the column
```

Out[36]:
```
0    White
1      Red
2     Blue
3    Black
4    White
5    Green
6     Blue
7     Blue
8    White
9    White
Name: Colour, dtype: object
```

In [37]:
```
sales_car.Colour
## But if your column name has sapce it does not works you have to go with the[""] Syntax
```

Out[37]:
```
0    White
1      Red
2     Blue
3    Black
4    White
5    Green
6     Blue
7     Blue
8    White
9    White
Name: Colour, dtype: object
```

In [38]:
```
#sales_car.Odometer (KM) ##returns ERROR
```

In [39]:
```
sales_car["Odometer (KM)"] ##works perfectly Fine
```

Out[39]:
```
0    150043
1     87899
2     32549
```

```
3     11179
4    213095
5     99213
6     45698
7     54738
8     60000
9     31600
Name: Odometer (KM), dtype: int64
```

In [40]:
```
sales_car[sales_car["Make"]=="Toyota"] ##getting only specific data
#in the above code we are telling to the compiler that:
#show me only those rows which have Toyota under the Make Column
```

Out[40]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
2	Toyota	Blue	32549	3	$7,000.00
5	Toyota	Green	99213	4	$4,500.00
8	Toyota	White	60000	4	$6,250.00

In [41]:
```
sales_car
```

Out[41]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [42]:
```
[sales_car["Odometer (KM)"]>100000]
##in the Boolean Format
```

Out[42]:
```
[0     True
 1    False
 2    False
 3    False
 4     True
 5    False
 6    False
 7    False
 8    False
 9    False
 Name: Odometer (KM), dtype: bool]
```

In [43]:
```
sales_car[sales_car["Odometer (KM)"]>100000]
```

Out[43]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
4	Nissan	White	213095	4	$3,500.00

In [44]:
```
pd.crosstab(sales_car["Make"],sales_car["Doors"])
#comparing two Columns
#kind of aggregates the two columns
```

Out[44]:

Doors	3	4	5
Make			
BMW	0	0	1
Honda	0	3	0
Nissan	0	2	0
Toyota	1	3	0

In [45]:
```
sales_car
```

Out[45]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

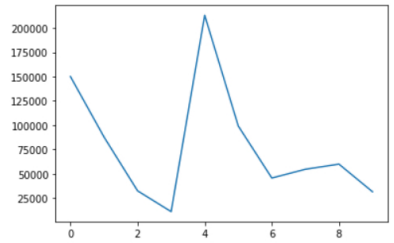
In [46]:
```
#Groupby
sales_car.groupby(["Make"]).mean()
#it groups the Data Frame on the Make column and gives the mean values of the all the Featues of the Data
# i.e, mean of the columns of Integer Data Types
#average
```

Out[46]:

	Odometer (KM)	Doors
Make		
BMW	11179.000000	5.00
Honda	62778.333333	4.00
Nissan	122347.500000	4.00
Toyota	85451.250000	3.75

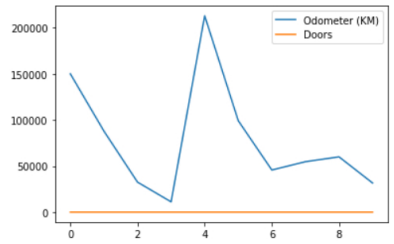
In [47]:
```
#Note :  the plot does not shows up, Run these codes before the plot fucntion and run the whole again
# %matplotlib inline
# import matplotlib.pyplot as plt
#this imports the library of plotting and inline function helps to show the plot in the Jupyter Notebook itself

#Note plot only the NUMERIC Data Types

sales_car["Odometer (KM)"].plot()
#visualization of the Specific Column
```

Out[47]: <AxesSubplot:>



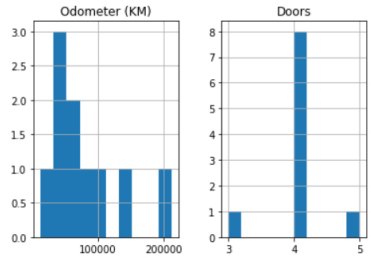
In [48]:
```
sales_car.plot()
#visualization of the entire DataFrame
```

Out[48]: <AxesSubplot:>



In [49]:
```
sales_car.hist()
# Histogram
```

Out[49]:
```
array([[<AxesSubplot:title={'center':'Odometer (KM)'}>,
        <AxesSubplot:title={'center':'Doors'}>]], dtype=object)
```



In [50]:
```
sales_car
```

Out[50]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [51]:
```
#sales_car["Price"].plot()
#This does not runs since the Price is Object and Not numeric Data Type
# therefore we have to figure out a way to convert this
```

In [52]:
```
sales_car["Price"] = sales_car["Price"].str.replace('[\$\,]|\.\d*', '').astype(int)
##converted the Data Types of Object to INteger
# Did this by searching the Problem statement into Google
#Then matching The problem in StackOverflow
#Copies the code , Tweaked it according to my Code Requirements/Prerequisites



#ALternative way:
#car_sales["Price"]=car_sales['Price'].str.replace(',', '').str.replace('$', '').astype(float)



#What this does is that it (.str.replace) replaces the $ sign with a " " space and replaces the type with an integer
#Note ; There are solutions that only replace the dollar sign and add extra zeros
#Look for the proper solutions
```

In [53]:
```
sales_car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
```

```
 0   Make            10 non-null     object
 1   Colour          10 non-null     object
 2   Odometer (KM)   10 non-null     int64
 3   Doors           10 non-null     int64
 4   Price           10 non-null     int32
dtypes: int32(1), int64(2), object(2)
memory usage: 488.0+ bytes
```

In [54]:
```
sales_car
```

Out[54]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000
1	Honda	Red	87899	4	5000
2	Toyota	Blue	32549	3	7000
3	BMW	Black	11179	5	22000
4	Nissan	White	213095	4	3500
5	Toyota	Green	99213	4	4500
6	Honda	Blue	45698	4	7500
7	Honda	Blue	54738	4	7000
8	Toyota	White	60000	4	6250
9	Nissan	White	31600	4	9700

In [55]:
```
sales_car
```

Out[55]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000
1	Honda	Red	87899	4	5000
2	Toyota	Blue	32549	3	7000
3	BMW	Black	11179	5	22000
4	Nissan	White	213095	4	3500
5	Toyota	Green	99213	4	4500
6	Honda	Blue	45698	4	7500
7	Honda	Blue	54738	4	7000
8	Toyota	White	60000	4	6250
9	Nissan	White	31600	4	9700

Manipulating Data

In [56]:
```
sales_car["Make"].str.lower() #lowercases
#here with the .str we are accesssing the string value of the column
#.lower() helps to lowercases alll the string accessed here
```

Out[56]:
```
0    toyota
1     honda
2    toyota
3       bmw
4    nissan
5    toyota
6     honda
7     honda
8    toyota
9    nissan
Name: Make, dtype: object
```

In [57]:
```
sales_car
#as we can see the loweered string has not been saved to the original data
#to save to the original data  we have to reassign the lowered to the original DataFrame
```

Out[57]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000
1	Honda	Red	87899	4	5000
2	Toyota	Blue	32549	3	7000
3	BMW	Black	11179	5	22000
4	Nissan	White	213095	4	3500
5	Toyota	Green	99213	4	4500
6	Honda	Blue	45698	4	7500
7	Honda	Blue	54738	4	7000
8	Toyota	White	60000	4	6250
9	Nissan	White	31600	4	9700

In [58]:
```
sales_car["Make"] = sales_car["Make"].str.lower()
#here we reassign the original Make column to the lowered column
```

In [59]:
```
sales_car
```

Out[59]:

	Make	Colour	Odometer (KM)	Doors	Price
0	toyota	White	150043	4	4000
1	honda	Red	87899	4	5000
2	toyota	Blue	32549	3	7000
3	bmw	Black	11179	5	22000
4	nissan	White	213095	4	3500
5	toyota	Green	99213	4	4500
6	honda	Blue	45698	4	7500
7	honda	Blue	54738	4	7000
8	toyota	White	60000	4	6250
9	nissan	White	31600	4	9700

In [60]:
```
#in real world we also get missing data
# to fix at we have :
```

In [61]:
```
sales_car_missing = pd.read_csv("car-sales-missing-data.csv")
```

In [62]:
```
sales_car_missing
#here the NaN are the missing values
```

Out[62]:

	Make	Colour	Odometer	Doors	Price
0	Toyota	White	150043.0	4.0	$4,000
1	Honda	Red	87899.0	4.0	$5,000
2	Toyota	Blue	NaN	3.0	$7,000
3	BMW	Black	11179.0	5.0	$22,000
4	Nissan	White	213095.0	4.0	$3,500
5	Toyota	Green	NaN	4.0	$4,500
6	Honda	NaN	NaN	4.0	$7,500
7	Honda	Blue	NaN	4.0	NaN
8	Toyota	White	60000.0	NaN	NaN
9	NaN	White	31600.0	4.0	$9,700

In [63]:
```
sales_car_missing["Odometer"]=sales_car_missing["Odometer"].fillna(sales_car_missing["Odometer"].mean())

# we can also use This line inplace = True will assign this to out DataFrame there itself
#sales_car_missing["Odometer"].fillna(sales_car_missing["Odometer"].mean(),inplace = True)

#what this does is that it takes all the missing values(NaN) and fills it with the mean vale of the Odometer
#At the same time we are assigning it to the original value as well
```

In [64]:
```
sales_car_missing.dropna()
#it drops the rows if one of the values in the rows is NaN
#by default INplace is Flase so this changes are not saved inthe original DataFrame
```

Out[64]:

	Make	Colour	Odometer	Doors	Price
0	Toyota	White	150043.000000	4.0	$4,000
1	Honda	Red	87899.000000	4.0	$5,000
2	Toyota	Blue	92302.666667	3.0	$7,000
3	BMW	Black	11179.000000	5.0	$22,000
4	Nissan	White	213095.000000	4.0	$3,500
5	Toyota	Green	92302.666667	4.0	$4,500

In [65]:
```
#creat data from existing data
# columns from series
seats_column = pd.Series([5,5,5,5,5])

#new column called seats
sales_car["Seats"] = seats_column
sales_car
#we created a new column called seats
```

Out[65]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats
0	toyota	White	150043	4	4000	5.0
1	honda	Red	87899	4	5000	5.0
2	toyota	Blue	32549	3	7000	5.0
3	bmw	Black	11179	5	22000	5.0
4	nissan	White	213095	4	3500	5.0
5	toyota	Green	99213	4	4500	NaN
6	honda	Blue	45698	4	7500	NaN
7	honda	Blue	54738	4	7000	NaN
8	toyota	White	60000	4	6250	NaN
9	nissan	White	31600	4	9700	NaN

In [66]:
```
sales_car["Seats"].fillna(5,inplace = True)
#filling in missing NaN values with 5
```

In [67]:
```
sales_car
```

Out[67]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats
0	toyota	White	150043	4	4000	5.0
1	honda	Red	87899	4	5000	5.0
2	toyota	Blue	32549	3	7000	5.0
3	bmw	Black	11179	5	22000	5.0
4	nissan	White	213095	4	3500	5.0
5	toyota	Green	99213	4	4500	5.0
6	honda	Blue	45698	4	7500	5.0
7	honda	Blue	54738	4	7000	5.0
8	toyota	White	60000	4	6250	5.0
9	nissan	White	31600	4	9700	5.0

In [68]:
```
#columns from python List
fuel_economy = [7,9,5,7,3,2,3,2,5,6]
sales_car["Fuel per 100 KM"] = fuel_economy
# here if you give less values (10 as of this data set ) we will get an error
```

In [69]:
```
sales_car
```

Out[69]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM
0	toyota	White	150043	4	4000	5.0	7
1	honda	Red	87899	4	5000	5.0	9
2	toyota	Blue	32549	3	7000	5.0	5
3	bmw	Black	11179	5	22000	5.0	7
4	nissan	White	213095	4	3500	5.0	3

5	toyota	Green	99213	4	4500	5.0	2
6	honda	Blue	45698	4	7500	5.0	3
7	honda	Blue	54738	4	7000	5.0	2
8	toyota	White	60000	4	6250	5.0	5
9	nissan	White	31600	4	9700	5.0	6

In [70]:
```
#creating column from another column
sales_car["Total fuel Used (Litres)"] = sales_car["Odometer (KM)"]/100 * sales_car["Fuel per 100 KM"]
sales_car
# created a new column named Total fuel used
# total fuel used = distance travelled by car("Odometer")/100 * fuel per 100 km
```

Out[70]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Total fuel Used (Litres)
0	toyota	White	150043	4	4000	5.0	7	10503.01
1	honda	Red	87899	4	5000	5.0	9	7910.91
2	toyota	Blue	32549	3	7000	5.0	5	1627.45
3	bmw	Black	11179	5	22000	5.0	7	782.53
4	nissan	White	213095	4	3500	5.0	3	6392.85
5	toyota	Green	99213	4	4500	5.0	2	1984.26
6	honda	Blue	45698	4	7500	5.0	3	1370.94
7	honda	Blue	54738	4	7000	5.0	2	1094.76
8	toyota	White	60000	4	6250	5.0	5	3000.00
9	nissan	White	31600	4	9700	5.0	6	1896.00

In [71]:
```
#create a column from a single value
sales_car["Number of Wheels"] = 4
sales_car
```

Out[71]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Total fuel Used (Litres)	Number of Wheels
0	toyota	White	150043	4	4000	5.0	7	10503.01	4
1	honda	Red	87899	4	5000	5.0	9	7910.91	4
2	toyota	Blue	32549	3	7000	5.0	5	1627.45	4
3	bmw	Black	11179	5	22000	5.0	7	782.53	4
4	nissan	White	213095	4	3500	5.0	3	6392.85	4
5	toyota	Green	99213	4	4500	5.0	2	1984.26	4
6	honda	Blue	45698	4	7500	5.0	3	1370.94	4
7	honda	Blue	54738	4	7000	5.0	2	1094.76	4
8	toyota	White	60000	4	6250	5.0	5	3000.00	4
9	nissan	White	31600	4	9700	5.0	6	1896.00	4

In [72]:
```
sales_car["Passed road safety"] = True
sales_car.dtypes
#bool  = Boolean
#pandas is versatile you can use variety of data tupes
```

Out[72]:
```
Make                       object
Colour                     object
Odometer (KM)               int64
Doors                       int64
Price                       int32
Seats                     float64
Fuel per 100 KM             int64
Total fuel Used (Litres)  float64
Number of Wheels            int64
Passed road safety           bool
dtype: object
```

In [73]:
```
sales_car
```

Out[73]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Total fuel Used (Litres)	Number of Wheels	Passed road safety
0	toyota	White	150043	4	4000	5.0	7	10503.01	4	True
1	honda	Red	87899	4	5000	5.0	9	7910.91	4	True
2	toyota	Blue	32549	3	7000	5.0	5	1627.45	4	True
3	bmw	Black	11179	5	22000	5.0	7	782.53	4	True
4	nissan	White	213095	4	3500	5.0	3	6392.85	4	True
5	toyota	Green	99213	4	4500	5.0	2	1984.26	4	True
6	honda	Blue	45698	4	7500	5.0	3	1370.94	4	True
7	honda	Blue	54738	4	7000	5.0	2	1094.76	4	True
8	toyota	White	60000	4	6250	5.0	5	3000.00	4	True
9	nissan	White	31600	4	9700	5.0	6	1896.00	4	True

In [74]:
```
#removing one of the columns
sales_car.drop("Total fuel Used (Litres)",axis = 1,inplace = True)
sales_car
#here axis = 1 means column and axis 0 means rows
```

Out[74]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Number of Wheels	Passed road safety
0	toyota	White	150043	4	4000	5.0	7	4	True
1	honda	Red	87899	4	5000	5.0	9	4	True
2	toyota	Blue	32549	3	7000	5.0	5	4	True
3	bmw	Black	11179	5	22000	5.0	7	4	True
4	nissan	White	213095	4	3500	5.0	3	4	True
5	toyota	Green	99213	4	4500	5.0	2	4	True
6	honda	Blue	45698	4	7500	5.0	3	4	True
7	honda	Blue	54738	4	7000	5.0	2	4	True
8	toyota	White	60000	4	6250	5.0	5	4	True
9	nissan	White	31600	4	9700	5.0	6	4	True

In [75]:
```
#shuffling rows in a database
shuffled_sales_car = sales_car.sample(frac=1)
#frac = what part of the data you want to randomize
# eg : 0.5 for 50% of the data 1 for 100% of data and same like that
```

In [76]:
```
shuffled_sales_car
```

Out[76]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Number of Wheels	Passed road safety
8	toyota	White	60000	4	6250	5.0	5	4	True
4	nissan	White	213095	4	3500	5.0	3	4	True
5	toyota	Green	99213	4	4500	5.0	2	4	True
6	honda	Blue	45698	4	7500	5.0	3	4	True
1	honda	Red	87899	4	5000	5.0	9	4	True
3	bmw	Black	11179	5	22000	5.0	7	4	True
2	toyota	Blue	32549	3	7000	5.0	5	4	True
7	honda	Blue	54738	4	7000	5.0	2	4	True
9	nissan	White	31600	4	9700	5.0	6	4	True
0	toyota	White	150043	4	4000	5.0	7	4	True

In [77]:
```
#only selecting 20%
#since in real world there are millions of rows
shuffled_sales_car.sample(frac = 0.2)
```

Out[77]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Number of Wheels	Passed road safety
2	toyota	Blue	32549	3	7000	5.0	5	4	True
0	toyota	White	150043	4	4000	5.0	7	4	True

In [78]:
```
#reseting the shuffled to order wise index numbers
shuffled_sales_car.reset_index(drop = True,inplace = True)
#drop is false by default and if it is false it does not resets ,
# it creates a new index column starting from zero
```

In [79]:
```
shuffled_sales_car
```

Out[79]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Number of Wheels	Passed road safety
0	toyota	White	60000	4	6250	5.0	5	4	True
1	nissan	White	213095	4	3500	5.0	3	4	True
2	toyota	Green	99213	4	4500	5.0	2	4	True
3	honda	Blue	45698	4	7500	5.0	3	4	True
4	honda	Red	87899	4	5000	5.0	9	4	True
5	bmw	Black	11179	5	22000	5.0	7	4	True
6	toyota	Blue	32549	3	7000	5.0	5	4	True
7	honda	Blue	54738	4	7000	5.0	2	4	True
8	nissan	White	31600	4	9700	5.0	6	4	True
9	toyota	White	150043	4	4000	5.0	7	4	True

In [80]:
```
sales_car
```

Out[80]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Number of Wheels	Passed road safety
0	toyota	White	150043	4	4000	5.0	7	4	True
1	honda	Red	87899	4	5000	5.0	9	4	True
2	toyota	Blue	32549	3	7000	5.0	5	4	True
3	bmw	Black	11179	5	22000	5.0	7	4	True
4	nissan	White	213095	4	3500	5.0	3	4	True
5	toyota	Green	99213	4	4500	5.0	2	4	True
6	honda	Blue	45698	4	7500	5.0	3	4	True
7	honda	Blue	54738	4	7000	5.0	2	4	True
8	toyota	White	60000	4	6250	5.0	5	4	True
9	nissan	White	31600	4	9700	5.0	6	4	True

In [81]:
```
sales_car["Odometer (KM)"] = sales_car["Odometer (KM)"].apply(lambda x : x/1.6)
#lambda is a fucntion in python
#here lambda function takes the x("Odometer values") and divides it by 1.6 and returns the divided number
#here we are converting KM to Miles
#apply is a way where we can assign a function to a column
sales_car
```

Out[81]:

	Make	Colour	Odometer (KM)	Doors	Price	Seats	Fuel per 100 KM	Number of Wheels	Passed road safety
0	toyota	White	93776.875	4	4000	5.0	7	4	True
1	honda	Red	54936.875	4	5000	5.0	9	4	True
2	toyota	Blue	20343.125	3	7000	5.0	5	4	True
3	bmw	Black	6986.875	5	22000	5.0	7	4	True
4	nissan	White	133184.375	4	3500	5.0	3	4	True
5	toyota	Green	62008.125	4	4500	5.0	2	4	True
6	honda	Blue	28561.250	4	7500	5.0	3	4	True
7	honda	Blue	34211.250	4	7000	5.0	2	4	True
8	toyota	White	37500.000	4	6250	5.0	5	4	True
9	nissan	White	19750.000	4	9700	5.0	6	4	True

In [82]:
```
#renaming a column
sales_car = sales_car.rename(columns={"Odometer (KM)": "Odometer (Miles)"})
sales_car
```

Out[82]:

	Make	Colour	Odometer (Miles)	Doors	Price	Seats	Fuel per 100 KM	Number of Wheels	Passed road safety
0	toyota	White	93776.875	4	4000	5.0	7	4	True
1	honda	Red	54936.875	4	5000	5.0	9	4	True
2	toyota	Blue	20343.125	3	7000	5.0	5	4	True

2	toyota	Blue	28543.125	5	7000	5.0	5	4	True
3	bmw	Black	6986.875	5	22000	5.0	7	4	True
4	nissan	White	133184.375	4	3500	5.0	3	4	True
5	toyota	Green	62008.125	4	4500	5.0	2	4	True
6	honda	Blue	28561.250	4	7500	5.0	3	4	True
7	honda	Blue	34211.250	4	7000	5.0	2	4	True
8	toyota	White	37500.000	4	6250	5.0	5	4	True
9	nissan	White	19750.000	4	9700	5.0	6	4	True