Introduction to pandans

~ Panda Series

- Creating Series
- Accessing Series
- Flitering Series
- Arithmetic Series
- Ranking nad Sorting
- Null Values

Pandas Dataframe

- Reading Data Fram Different Sources
- DataFrame Manipulating
- Understanding Data
- Indexing Dataframe
- Slicing and ranking

What is Pandas

- Pandas is simple yet powerful and expressive tool
- It is a Open source Library
- It is useful in Data Manipulation(convert from one from to another from) and Analysis

In [ ]:
```
Pandas is high level datastructure and numpy is low level datastructu
data alignment merge or join two datasets or convert two datasets int
```

In [3]:
```
import pandas as pd
import numpy as np
```

Components of Pandas

1.series and Dataframes are the Major components here

2.one Dimensional - Series

3.Multi Dimensional - DataFrames and Panel Data

In [ ]:
```
`Create a Series

1.A Pandas has been created by using list and Arrays
```

In [4]:
```
# Creating with list

##pd.Series (S should be capital)

series_list = pd.Series([1,2,3,4,5])
print(series_list)
```

```
0    1
1    2
2    3
```

In [5]:
```
# Creating a series by using arrays

series_array = pd.Series(np.array([10,20,30,40]))
print(series_array)

# index is very very important to fetch the dataframes and datasets
```

```
0    10
1    20
2    30
3    40
dtype: int64
```

`index of Series

In [4]:
```
import pandas as pd
import numpy as np
#Setting the index by myself
series_index = pd.Series(
    np.array([11,12,13,14,15,16]),
    index = np.arange(0,12,2))
print(series_index)

#Setting the index by myself
series_index = pd.Series(
    np.array([11,12,13,14,15,16]),
    index = ['a','b','c','d','e','f']
)
print(series_index)
```

```
0     11
2     12
4     13
6     14
8     15
10    16
dtype: int64
a    11
b    12
c    13
d    14
e    15
f    16
dtype: int64
```

##Assignment Given an array of non-negative integers and an integers and an integer sum,find a subarray that adds to a given sum

In [1]:
```
def subarray(arr, n, givensum):
    sum1 = arr[0]
    start = 0
    i = 1
    while i <= n:
        while sum1 > givensum and start < i-1:
```

```
            sum1 = sum1 - arr[start]
            start += 1
        if sum1 == givensum:
            print ("Subarray with given sum is between indexes % d ar
            return 1
        if i < n:
            sum1 = sum1 + arr[i]
        i += 1
    print ("Subarray with given sum is NOT Found")
    return 0

arr = [2, 6, 5, 31, 11, 8]
n = len(arr)
givensum = 53

subarray(arr, n, givensum)
```

```
Subarray with given sum is between indexes  1 and  4
```

Out[1]: 1

In [8]:
```
#Create a series by dictionary:
alphabet_dict = {'a' : 1,
                 'b' : 2,
                 'c' : 3,
                 'd' : 4}
series_dict = pd.Series(alphabet_dict)
print(series_dict)
#series is the heterogenuous datatype
print('\n')
alphabet_dict = {'a' :[1,2,3],
                 'b' : [5,6,7],
                 'c' : [5,6],
                 'd' : 'Hello world'}
print(type(alphabet_dict))
series_dict = pd.Series(alphabet_dict)
print(series_dict)
```

```
a    1
b    2
c    3
d    4
dtype: int64


<class 'dict'>
a      [1, 2, 3]
b      [5, 6, 7]
c         [5, 6]
d    Hello world
dtype: object
```

In [19]:
```
print(series_dict.index)
print(series_dict.values)

alpha_array = np.array(['a','b','c','d','e'])
alpha_series = pd.Series(alpha_array)
# index
print(alpha_series[2])
# slicing
print(alpha_series[1:4])
print('\n')
```

```
series_index = pd.Series(
    np.array([11,12,13,14,15,16]),
    index = ['a','b','c','d','e','f']
)
print(series_index['c']) ## print(series_index['c','d','e']) is not v
print(series_index[['c','d','e']])# two dimensional
```

```
Index(['a', 'b', 'c', 'd'], dtype='object')
[list([1, 2, 3]) list([5, 6, 7]) list([5, 6]) 'Hello world']
c
1    b
2    c
3    d
dtype: object


13
c    13
d    14
e    15
dtype: int64
```

In [23]:
```
## filter the series

num_array = np.array([11,12,13,14,15,16,17])
num_series = pd.Series(num_array)
print(num_series)
# i want to filter out which is greater then 15
print(num_series > 15)
print(num_series[num_series > 15])
```

```
0    11
1    12
2    13
3    14
4    15
5    16
6    17
dtype: int64
0    False
1    False
2    False
3    False
4    False
5     True
6     True
dtype: bool
5    16
6    17
dtype: int64
```

In [29]:
```
# Arithmetic Operators
num_array = np.array([1,2,3])
num_series = pd.Series(num_array)
print(num_series*2)

odd_series = pd.Series([1,3,5,7])
even_series = pd.Series([2,4,6,8])
multi_odd_even = odd_series.multiply(even_series)
print(multi_odd_even)
```

```
add_odd_even = odd_series.add(even_series)
print(add_odd_even)
```

```
0    2
1    4
2    6
dtype: int64
0     2
1    12
2    30
3    56
dtype: int64
0     3
1     7
2    11
3    15
dtype: int64
```

In [ ]:
```
##Assignment
#create a series by using dictionary and filter out number which is g
#Create a series by using numpy array from the range 1 to 100 filter
#95 to 100
```

In [31]:
```
# 1.create a series by using dictionary and filter out number which i
alphabet_dict = {'a' : 1,
                 'b' : 2,
                 'c' : 3,
                 'd' : 4}
series_dict = pd.Series(alphabet_dict)
print(series_dict[series_dict > 3])
```

```
d    4
dtype: int64
```

In [75]:
```
import numpy as np
import pandas as pd
##2.Create a series by using numpy array from the range 1 to 100 filt
series_array = pd.Series(np.arange(100))
print(series_array[series_array > 95].reset_index(drop = True))
```

```
0    96
1    97
2    98
3    99
dtype: int64
```

In [70]:
```
num_array=np.array(range(1,101))
num_series=pd.Series(num_array)
num_series[(95 < num_series) & (num_series < 100)]
```

Out[70]:
```
95    96
96    97
97    98
98    99
dtype: int64
```

In [ ]:
```
import numpy as np
import pandas as pd
```

In [10]:
```
import numpy as np
import pandas as pd
# Sorting a Series by Values
```

```
#create a Series
num_series = pd.Series([123,445,np.nan,411,223,155,np.nan,314,210])
print(num_series)
print(num_series.sort_values())
print(num_series.sort_values(ascending = False,na_position = 'first'))
print(num_series.sort_values(ascending = False,na_position = 'last'))
num_series.sort_index(ascending = True)
# NaN is consider as null value
```

```
0    123.0
1    445.0
2      NaN
3    411.0
4    223.0
5    155.0
6      NaN
7    314.0
8    210.0
dtype: float64
0    123.0
5    155.0
8    210.0
4    223.0
7    314.0
3    411.0
1    445.0
2      NaN
6      NaN
dtype: float64
2      NaN
6      NaN
1    445.0
3    411.0
7    314.0
4    223.0
8    210.0
5    155.0
0    123.0
dtype: float64
1    445.0
3    411.0
7    314.0
4    223.0
8    210.0
5    155.0
0    123.0
2      NaN
6      NaN
dtype: float64
```

Out[10]:

```
0    123.0
1    445.0
2      NaN
3    411.0
4    223.0
5    155.0
6      NaN
7    314.0
8    210.0
dtype: float64
```

In [13]:
```
## Null Values:
num_series = pd.Series([123,445,np.nan,411,223,155,np.nan,np.nan,314,
print(num_series.isnull())
print(num_series.isnull().sum()) # how many null values are present
```

```
0    False
1    False
2     True
3    False
4    False
5    False
6     True
7     True
8    False
9    False
dtype: bool
3
```

Assignments

Write a python program to find palindromes in a given list of strings using lambda. Original list of strings: ['php','w3r','python','abcd','java','aaa'] List of pallindromes: ['php','aaa']

In [15]:
```
String = ['php','w3r','python','abcd','java','aaa']
a = list(filter(lambda x : (x == ''.join(reversed(x))),String))
print(a)
```

```
['php', 'aaa']
```

Pandas Dataframe:

- A DataFrame is a Two Dimensional data Structure
- Data are aligned in (rows and columns)

Features of DataFrame:

- Columns can be of different Types
- Size are Mutable
- Axes are labeled(rows and Colums)
- Arithmetic Operations on rows and Columns

In [2]:
```
# load the Library:
import pandas as pd
import numpy as np

word_list = ['Python','For','Data','Science']
df_words = pd.DataFrame(word_list)
df_words
```

Out[2]:

	0
0	Python
1	For

	0
2	Data

In [8]:
```
# Create a DataFrame by list of list:

salary_list = [['john',3000],['mia',4000],['Robert',6000]]
salary = pd.DataFrame(salary_list)
print(salary)
```

```
        0     1
0    john  3000
1     mia  4000
2  Robert  6000
```

In [25]:
```
# Create a DataFrame with Index

sales_list = {'Month' : ['jan','Feb','mar','apr'],
              'Sales': [50000,60000,70000,8000]}
df_sales = pd.DataFrame(sales_list,index = ['A','B','C','D'])
df_sales
```

Out[25]:

	Month	Sales
A	jan	50000
B	Feb	60000
C	mar	70000
D	apr	8000

In [4]:
```
data = {'Name' : ['Akshal','James','Mia','Emily','Robern','john','Jac
       'Score' : [12,19,15,10,17,8,17],
       'Attempts' : [3,2,1,4,5,2,1],
       'Qualify' : ['Yes','Yes','Yes','NO','Yes','No','Yes']}
student = pd.DataFrame(data)
student
```

Out[4]:

	Name	Score	Attempts	Qualify
0	Akshal	12	3	Yes
1	James	19	2	Yes
2	Mia	15	1	Yes
3	Emily	10	4	NO
4	Robern	17	5	Yes
5	john	8	2	No
6	Jacob	17	1	Yes

In [18]:
```
## Assignment
## Print only coloums
student.iloc[:,[0,2]]
```

Out[18]:

	Name	Attempts
0	Akshal	3
1	James	2
2	Mia	1
3	Emily	4

	Name	Attempts
4	Robern	5
5	john	2

In [ ]:
```
### LOc[]

- The loc[] selects the data by the label of rows and colums

- Retrieve the Rows and Cloums(location base)
```

In [11]:
```
print(student.loc[1]['Score'])
print(student.loc[[0,1,2],['Name','Qualify']])
```

```
19
     Name Qualify
0  Akshal     Yes
1   James     Yes
2     Mia     Yes
```

In [6]:
```
# retrieve all the information of students whose score is more than
student[student['Score']>12]
# or student[student.Score>12]
```

Out[6]:

	Name	Score	Attempts	Qualify
1	James	19	2	Yes
2	Mia	15	1	Yes
4	Robern	17	5	Yes
6	Jacob	17	1	Yes

In [12]:
```
# Retrieve the Student who either have more than two attempts or have

data = {'Name' : ['Akshal','James','Mia','Emily','Robern','john','Jac
        'Score' : [12,19,15,10,17,8,17],
        'Attempts' : [3,2,1,4,5,2,1],
        'Qualify' : ['Yes','Yes','Yes','NO','Yes','No','Yes']}
student = pd.DataFrame(data)
student[(student.Qualify == 'Yes') | (student.Attempts > 2)]
```

Out[12]:

	Name	Score	Attempts	Qualify
0	Akshal	12	3	Yes
1	James	19	2	Yes
2	Mia	15	1	Yes
3	Emily	10	4	NO
4	Robern	17	5	Yes
6	Jacob	17	1	Yes

In [20]:
```
# Read the Files

import pandas as pd
import numpy as np
df_market = pd.read_excel('Supermarketfile.xlsx',engine='openpyxl')
print(df_market)
```

	Day	Store	Percentage
0	Monday	A	79
1	Monday	B	81
2	Monday	C	74
3	Monday	D	77
4	Monday	E	66
5	Tuesday	A	78

In [11]:
```
import os
os.getcwd()
```

Out[11]: '/home/student/my_project_dir'

In [1]:
```
!pip list
```

```
Package               Version
--------------------- ---------
anyio                 3.6.2
argon2-cffi           21.3.0
argon2-cffi-bindings 21.2.0
async-generator       1.10
attrs                 22.2.0
Babel                 2.11.0
backcall              0.2.0
bleach                4.1.0
certifi               2022.12.7
cffi                  1.15.1
charset-normalizer    2.0.12
contextvars           2.4
cycler                0.11.0
dataclasses           0.8
decorator             5.1.1
defusedxml            0.7.1
entrypoints           0.4
et-xmlfile            1.1.0
idna                  3.4
immutables            0.19
importlib-metadata    4.8.3
install               1.3.5
ipykernel             5.5.6
ipython               7.16.3
ipython-genutils      0.2.0
jedi                  0.17.2
Jinja2                3.0.3
json5                 0.9.11
jsonschema            3.2.0
jupyter-client        7.1.2
jupyter-core          4.9.2
jupyter-server        1.13.1
jupyterlab            3.2.9
jupyterlab-pygments   0.1.2
jupyterlab-server     2.10.3
kiwisolver            1.3.1
MarkupSafe            2.0.1
```

In [21]:
```
# Sort the DataFrame :
print(df_market.sort_values('Percentage', ascending = False))
```

```
      Day  Store   Percentage
8  Tuesday      D          97
7  Tuesday      C          89
6  Tuesday      B          86
1  Monday       B          81
0  Monday       A          79
5  Tuesday      A          78
3  Monday       D          77
2  Monday       C          74
4  Monday       E          66
```

In [71]:
```
# Sort the DataFrame with the Condition of percentage is Greater thar
print(df_market[df_market.Percentage > 85].reset_index(drop = True))
```

```
      Day  Store   Percentage
```

In [23]:
```
### Pandas Advanced
- **Agenda**

- Concatenates Series and Dataframes
- Join the Dataframes
- Merge the Dataframe
- Stack/Unstack series and Dataframes
- Reshape
- Pivot Tables and Cross Tables
-
```

```
  File "<ipython-input-23-a44b7d63ee23>", line 2
    - **Agenda**
      ^
SyntaxError: invalid syntax
```

In [ ]:
```
#Concatenation link or join two or more strings.
#Append means to add an item to the exsiting list
```

In [23]:
```
# linspace in the form of random
#range function ending number is important
import pandas as pd
import numpy as np

even = np.linspace(start = 0, stop = 20, num = 11)
odd = np.linspace(start = 1,stop = 21,num = 11)
print(even)
print(odd)
even_Series = pd.Series(data = even)
odd_Series = pd.Series(data = odd)
print('Type of even numbers : ',even_Series)
print('Type of odd numbers : ',odd_Series)
```

```
[ 0.  2.  4.  6.  8. 10. 12. 14. 16. 18. 20.]
[ 1.  3.  5.  7.  9. 11. 13. 15. 17. 19. 21.]
Type of even numbers :  0      0.0
1      2.0
2      4.0
3      6.0
4      8.0
5     10.0
6     12.0
7     14.0
```

In [28]:
```
# Append Function
even_Series.append(odd_Series,ignore_index = True)
```

Out[28]:
```
0      0.0
1      2.0
2      4.0
3      6.0
4      8.0
5     10.0
6     12.0
7     14.0
8     16.0
9     18.0
10    20.0
11     1.0
12     3.0
13     5.0
14     7.0
15     9.0
16    11.0
17    13.0
18    15.0
19    17.0
20    19.0
21    21.0
dtype: float64
```

In [30]:
```
# Adding hierarchial index:

pd.concat([even_Series,odd_Series],keys = ['Even','Odd'],
          names = ['Category','Index'])
```

Out[30]:

```
Category  Index
Even      0        0.0
          1        2.0
          2        4.0
          3        6.0
          4        8.0
```

In [5]:
```
#Assignment
#Customer id,Age,Gender,Salary,City_Residence custermer_data_1
#Create an another sheet with same custermer_data_2
import pandas as pd
df_customer_1 = pd.read_excel('Untitled 2.xlsx',engine='openpyxl')
df_customer_1
```

Out[5]:

	Customer_id	Age	Gender	Salary	City_Residence
0	Nandini	21	Female	50000	chennai
1	Ammu	19	Female	30000	Anantapur
2	Banny	18	Male	40000	kadapa
3	Siree	23	Female	45000	kurnool
4	Sunny	18	Male	40000	Dharmavaram

In [12]:
```
import pandas as pd
df_customer_2 = pd.read_excel('customer_data.xlsx',engine='openpyxl')
df_customer_2
```

Out[12]:

	Customer_id	Age	Gender	Salary	City_Residence
0	Nandini	21	Female	50000	chennai
1	Ammu	19	Female	30000	Anantapur
2	Banny	18	Male	40000	kadapa
3	Siree	23	Female	45000	kurnool
4	Sunny	18	Male	40000	Dharmavaram

In [14]:
```
pd.concat([df_customer_1,df_customer_2],ignore_index = True)
```

Out[14]:

	Customer_id	Age	Gender	Salary	City_Residence
0	Nandini	21	Female	50000	chennai
1	Ammu	19	Female	30000	Anantapur
2	Banny	18	Male	40000	kadapa
3	Siree	23	Female	45000	kurnool
4	Sunny	18	Male	40000	Dharmavaram
5	Nandini	21	Female	50000	chennai
6	Ammu	19	Female	30000	Anantapur
7	Banny	18	Male	40000	kadapa
8	Siree	23	Female	45000	kurnool
9	Sunny	18	Male	40000	Dharmavaram

In [20]:
```
import pandas as pd
import numpy as np
```

```
even = np.linspace(start = 0, stop = 20, num = 11)
odd = np.linspace(start = 1,stop = 21,num = 11)
print(even)
print(odd)
even_Series = pd.Series(data = even)
odd_Series = pd.Series(data = odd)
print('Type of even numbers : ',even_Series)
print('Type of odd numbers : ',odd_Series)
```

```
[ 0.  2.  4.  6.  8. 10. 12. 14. 16. 18. 20.]
[ 1.  3.  5.  7.  9. 11. 13. 15. 17. 19. 21.]
Type of even numbers :  0      0.0
1      2.0
2      4.0
3      6.0
4      8.0
5     10.0
6     12.0
7     14.0
8     16.0
9     18.0
10    20.0
dtype: float64
Type of odd numbers :  0      1.0
1      3.0
2      5.0
3      7.0
4      9.0
5     11.0
6     13.0
7     15.0
8     17.0
9     19.0
10    21.0
dtype: float64
```

In [10]:
```
# Unstack a Series

odd_even_data = pd.concat([even_Series,odd_Series],
                          keys = ['Even','Odd'],
                          names = ['Category','Index'])
print(odd_even_data.unstack()) ## unstack is in tabular column
print('\n')
print(odd_even_data.unstack(level = 0))
```

```
Index      0    1    2    3    4     5     6     7     8     9
10
Category
Even      0.0  2.0  4.0  6.0  8.0  10.0  12.0  14.0  16.0  18.0  2
0.0
Odd       1.0  3.0  5.0  7.0  9.0  11.0  13.0  15.0  17.0  19.0  2
```

In [ ]:
```
# Unstacking is method to convert your hierarchial indexing format ir
# Unstacking to make hierarchial indexing into Tabular Format
# Stacking is inverse method of unstacking
# Stacking is directly proportional to hierarchial indexing
```

In [2]:
```
import pandas as pd
salary_list = [['john',3000],['mia',4000],['Robert',6000]]
salary = pd.DataFrame(salary_list)
print(salary)
```

```
        0     1
0    john  3000
1     mia  4000
2  Robert  6000
```

In [31]:
```
##df_sales_unstack()
# In DataFrame unstacking is meant of hiersrchial indexing
salary.unstack()
```

Out[31]:
```
0  0      john
   1       mia
   2    Robert
1  0      3000
   1      4000
   2      6000
dtype: object
```

In [32]:
```
salary.stack()
```

Out[32]:
```
0  0      john
   1      3000
1  0       mia
   1      4000
2  0    Robert
   1      6000
dtype: object
```

In [36]:
```
df_hr_employee = pd.read_excel('hr_data.xlsx',engine='openpyxl')
df_hr_employee
```

Out[36]:

	Age	Gender	Salary	City_residence
0	45	Male	40000	Mumbai
1	32	Male	50000	Banglore
2	26	Female	45000	Chennai
3	47	Female	55000	Delhi
4	44	Male	34000	Goa

Reshape()

- The Melt() method is used to Change the DataFrame format from wide to Long

In [39]:
```
df_melt = df_hr_employee.melt(id_vars = ['Gender','Age'])
print(df_melt)
```

```
   Gender  Age         variable     value
0    Male   45           Salary     40000
1    Male   32           Salary     50000
2  Female   26           Salary     45000
3  Female   47           Salary     55000
4    Male   44           Salary     34000
5    Male   45  City_residence    Mumbai
6    Male   32  City_residence  Banglore
7  Female   26  City_residence   Chennai
8  Female   47  City_residence     Delhi
9    Male   44  City_residence       Goa
```

In [42]:
```
##Assignment
# create excell sheet jan to dec yeilds season
import pandas as pd
weather_data = pd.read_excel('farming.xlsx',engine = 'openpyxl')
weather_data
```

Out[42]:

	Month	Yields	Season	Salary
0	Jan	5000	Winter	50000
1	Feb	4000	Winter	60000
2	Mar	3000	Summer	40000
3	June	2000	Summer	40000
4	July	6000	Rainy	50000
5	August	7000	Rainy	60000
6	September	4000	Monsoon	70000
7	October	3000	Autumn	70000
8	November	7000	Autumn	80000
9	December	6000	Winter	80000

In [43]:
```
weather_data.unstack()
```

Out[43]:

```
Month   0          Jan
        1          Feb
        2          Mar
        3         June
        4         July
        5       August
        6    September
        7      October
        8     November
        9     December
Yields  0         5000
        1         4000
        2         3000
        3         2000
        4         6000
        5         7000
        6         4000
        7         3000
        8         7000
        9         6000
Season  0       Winter
        1       Winter
        2       Summer
        3       Summer
        4        Rainy
        5        Rainy
        6      Monsoon
```

In [44]:
```
weather_data.stack()
```

Out[44]:

```
0  Month           Jan
   Yields         5000
   Season       Winter
   Salary        50000
1  Month           Feb
   Yields         4000
   Season       Winter
   Salary        60000
2  Month           Mar
   Yields         3000
   Season       Summer
```

In [45]:
```
# Pivot table
import pandas as pd
import numpy as np

df_yield = pd.read_excel('farming.xlsx',engine = 'openpyxl')
df_yield
```

Out[45]:

	Month	Yields	Season	Salary
0	Jan	5000	Winter	50000
1	Feb	4000	Winter	60000
2	Mar	3000	Summer	40000
3	June	2000	Summer	40000
4	July	6000	Rainy	50000
5	August	7000	Rainy	60000
6	September	4000	Monsoon	70000
7	October	3000	Autumn	70000
8	November	7000	Autumn	80000
9	December	6000	Winter	80000

Pivot Table

- It has a Dataframe like Structure
- It is used to dispaly the data for the specified column and index

In [47]:
```
# The Pivot() method generator a pivot table for the given index

pd.pivot_table(df_yield,index = ['Season'],
               values = ['Yields'])
# Fetch the average value of Yield in every year
```

Out[47]:

	Yields
Season	
Autumn	5000
Monsoon	4000
Rainy	6500
Summer	2500
Winter	5000

In [51]:
```
# Create a Pivot Table :
pd.pivot_table(df_yield,index = ['Season'],
               values = ['Yields'],
               aggfunc = 'sum')
#sum of all the  values in a season
```

Out[51]:

Season	Yields
Autumn	10000
Monsoon	4000
Rainy	13000
Summer	5000
Winter	15000

In [ ]:
```
# Cross table

- Cross Tables are very similar to pivot table
- It computes the cross tabulation of two or more factores
```

In [67]:
```
df_employee = pd.read_excel('employees.xlsx',engine = 'openpyxl')
df_employee
```

Out[67]:

	Age	Gender	Ciry_residence	Annual	Year of Experience	Designation
0	45	Male	Mumbai	21	5	Cloud engineer
1	67	Male	Mumbai	1	5	Daat analyst intern
2	33	Male	Mumbai	2	5	Sr. Data Analyst
3	56	Male	Mumbai	8	5	Big data Engineer
4	32	Male	Mumbai	4	5	Tutor
5	21	Male	Mumbai	9	7	Tutor
6	56	Male	Bangalore	3	7	Staff
7	43	Female	Bangalore	5	7	Staff
8	43	Female	Bangalore	6	4	Staff
9	42	Female	Pune	7	4	Java Developer
10	55	Female	Pune	8	4	Java Developer
11	44	Female	Pune	10	3	Java Developer
12	27	Male	Pune	4	3	Human Resource
13	27	Female	Pune	3	3	Human Resource
14	27	Female	Delhi	8	2	Sales Manager
15	45	Male	Delhi	9	2	Sales Manager
16	89	Female	Delhi	10	21	Marketing Manager
17	56	Male	Delhi	12	1	Marketing Manager
18	78	Female	Delhi	5	6	Cloud engineer

In [53]:
```
# Find the City-Wise Gender count using the crossTab() method:

pd.crosstab(df_employee.Gender,df_employee.Ciry_residence,
            rownames = ['Sex'],colnames = ['HomeTown'])
```

Out[53]:

HomeTown	Bangalore	Delhi	Mumbai	Pune
Sex				
Female	2	3	0	4
Male	1	2	6	1

In [ ]:
```
# Duplicate values
[1,1,1,1,1,1,1,2,1,2,2,2,1,2]
all other ones behind the 1st one are duplicate values
df_employee.duplicated(keeps = False)
```

In [68]:
```
# Assignment
##Find city-wise distibution of salary for different genders
pd.crosstab(df_employee.Gender,df_employee.Annual,
            rownames = ['Sex'],colnames = ['Salary'])
```

Out[68]:

Salary	1	2	3	4	5	6	7	8	9	10	12	21
Sex												
Female	0	0	1	0	2	1	1	2	0	2	0	0
Male	1	1	1	2	0	0	0	1	2	0	1	1

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [17]:
```

import seaborn as sns
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                 Name     Sex   A
ge  SibSp  \
0                               Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                                Heikkinen, Miss. Laina  female  2
6.0      0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                            Allen, Mr. William Henry    male  3
5.0      0
...                                                ...     ...
...    ...
1304                                 Spector, Mr. Woolf    male   N
aN      0
1305                       Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                       Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                                Ware, Mr. Frederick    male   N
aN      0
1308                           Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
1307      0              359309    8.0500  ...        S  1289.0
1308      1                2668   22.3583  ...        C   702.0

                                       Name_wiki Age_wiki  \
0                          Braund, Mr. Owen Harris     22.0
1     Cumings, Mrs. Florence Briggs (née Thayer)     35.0
2                            Heikkinen, Miss Laina     26.0
3              Futrelle, Mrs. Lily May (née Peel)     35.0
4                          Allen, Mr. William Henry     35.0
...                                             ...      ...
1304                         Spector, Mr. Woolf     23.0
1305        and maid, Doña Fermina Oliva y Ocana     39.0
```

```
1306                Sæther, Mr. Simon Sivertsen     43.0
1307                Ware, Mr. Frederick William     34.0
1308           Butrus-Youssef, Master Makhkhul       4.0

                                  Hometown      Boarded  \
0            Bridgerule, Devon, England  Southampton
1                New York, New York, US    Cherbourg
2                    Jyväskylä, Finland  Southampton
3          Scituate, Massachusetts, US  Southampton
```

In [18]:
```
df.titanic.head(10) # defalut we get 5 rows
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
5	6	0.0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583
6	7	0.0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625
7	8	0.0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750
8	9	1.0	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333
9	10	1.0	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708

10 rows × 21 columns

In [19]:
```
df_titanic.tail(30) # last rows
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1279	1280	NaN	3	Canavan, Mr. Patrick	male	21.0	0	0	364858
1280	1281	NaN	3	Palsson, Master. Paul Folke	male	6.0	3	1	349909
1281	1282	NaN	1	Payne, Mr. Vivian Ponsonby	male	23.0	0	0	12749
1282	1283	NaN	1	Lines, Mrs. Ernest H (Elizabeth Lindsey James)	female	51.0	0	1	PC 17592
1283	1284	NaN	3	Abbott, Master. Eugene	male	13.0	0	2	C.A. 2673

In [20]:
```
# Information about the Dataset:
df_titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
 12  WikiId       1304 non-null   float64
 13  Name_wiki    1304 non-null   object
 14  Age_wiki     1302 non-null   float64
 15  Hometown     1304 non-null   object
 16  Boarded      1304 non-null   object
 17  Destination  1304 non-null   object
 18  Lifeboat     502 non-null    object
 19  Body         130 non-null    object
 20  Class        1304 non-null   float64
dtypes: float64(6), int64(4), object(11)
memory usage: 214.9+ KB
```

In [5]:
```
df_titanic.isnull().sum()
df_titanic.isnull().sum().sum()
df_titanic.size #datatypes
df_titanic.shape# rows and columes
df_titanic.dtype
```

```
---------------------------------------------------------------------
--------
NameError                                 Traceback (most recent ca
ll last)
<ipython-input-5-11b0eeecc48a> in <module>
----> 1 df_titanic.isnull().sum()
      2 df_titanic.isnull().sum().sum()
      3 df_titanic.size #datatypes
      4 df_titanic.shape# rows and colums
      5 df_titanic.dtype

NameError: name 'df_titanic' is not defined
```

In [ ]:
```
# Create a DataFrame from(A-H)
#Location
#Sales Value as per user

# Create a Entire DataFrame
# Write a Programe to select stores located in hyderabad
# Write a Program to Select all columns except the location from the
# Write a Programe to Sort the columns except the location from the D
# Write a Programe to Sort the columns of sales and store
```

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = {'Store' : ['A','B','A','C','D','A','D','A'],
        'Location' : ['Mumbai','pune','hyderabad','Mumbai','pune','Del
        'Sales' : [40000,45000,50000,50000,89000,90000,40000,30000]
}
df = pd.DataFrame(data = data)

#Display the DataFrames
print(df)
print('\n')
print(df[df['Location'] == 'Hyderabad'])
print('\n')

df_sort = df.sort_value(by = )
```

```
  Store   Location  Sales
0     A     Mumbai  40000
1     B       pune  45000
2     A  hyderabad  50000
3     C     Mumbai  50000
4     D       pune  89000
5     A      Delhi  90000
6     D  hyderabad  40000
7     A       pune  30000


Empty DataFrame
Columns: [Store, Location, Sales]
Index: []
```

In [3]:
```
df_employee = pd.read_excel('Sales.xlsx',engine = 'openpyxl')
df_employee
```

Out[3]:

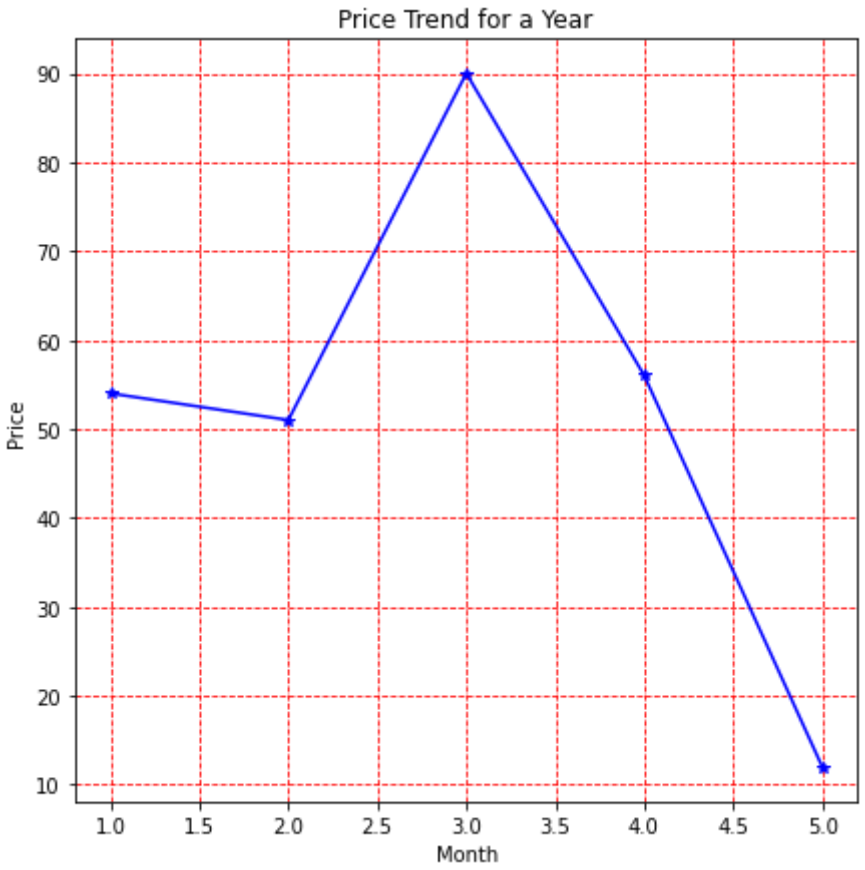
	Location	A	B	C	D	E	F	G	H
0	Bangalore	200	50	200	60	330	400	100	450
1	Hyderabad	100	220	180	80	430	500	150	550
2	Chennai	150	130	160	100	530	600	200	650
3	Goa	230	140	140	300	630	700	250	750
4	Delhi	180	150	120	280	730	800	300	850
5	Mumbai	120	190	100	260	830	900	350	950

In [ ]:
```
#Write a Programe to select stores located in hyderabad
```

Pandas Visualization

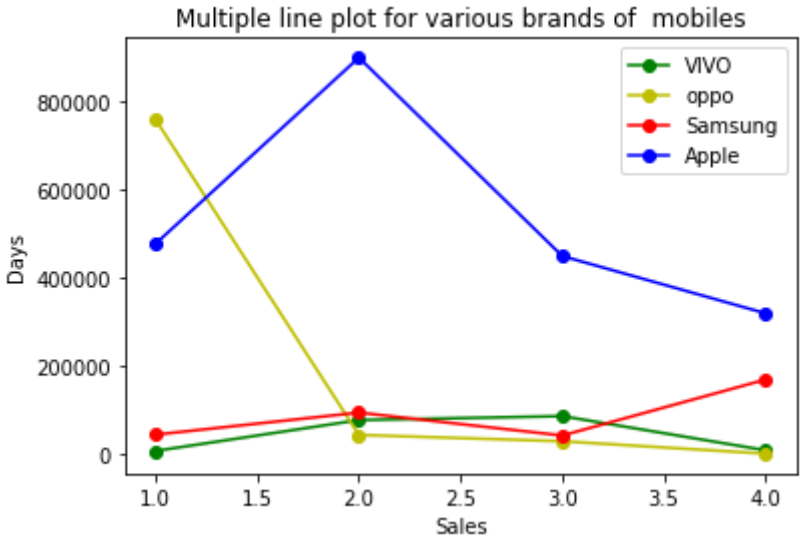
In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [11]:
```
# Line plot: for numerical datatypes sometimes for categerical datat

# Create Data:
month = [x for x in range(1,6)]
prices = [54,51,90,56,12]

# Set the Figure size:
plt.figure(figsize = (7,7))
# Create a Line Plot:
plt.plot(month,prices,color = 'b',marker = '*')

# Label the plot:
plt.title('Price Trend for a Year')

# Add axes Label:
plt.xlabel('Month')
plt.ylabel('Price')

# Add Grid line
plt.grid(color = 'r',ls = '--') # ls = line style

# Display the Plot
plt.show()
```



In [13]:
```
# Multiple Line Plots:

# Create a Data:
days = [1,2,3,4]
vivo_sales = [8000,78000,87000,9500]
oppo_sales = [759999,44440,30000,1900]
samsung_sales = [45000,95000,43000,170000]
apple_sales = [477777,900000,450000,320000]
```

```
# plot the Line Plot:
plt.plot(days,vivo_sales,color = 'g',label = 'VIVO',marker = 'o')
plt.plot(days,oppo_sales,color = 'y',label = 'oppo',marker = 'o')
plt.plot(days,samsung_sales,color = 'r',label = 'Samsung',marker = 'o
plt.plot(days,apple_sales,color = 'b',label = 'Apple',marker = 'o')

# Give Label:
plt.title('Multiple line plot for various brands of  mobiles')
plt.xlabel('Sales')
plt.ylabel('Days')

# Add Legend
plt.legend()
#display the plot
plt.show()
```



In [ ]:
```
## Assignment
Give employee id 10 and then salary and year of joining and age
do multiple line plots
```

In [28]:
```
id = [1001,1002,1003,1004,1005,1006,1007,1008,1009,1010]
salary = [45000,30000,35000,40000,50000,25000,30000,27000,34000,44000
year_joining = [2018,2019,2016,2018,2017,2020,2021,2022,2023,2016]
age = [23,24,25,26,27,22,24,28,29,30]
plt.grid()
plt.plot(id,salary,color = 'g',ls = '-',label = 'salary',marker = 'o'
plt.plot(id,year_joining,color = 'r',ls = '--',label = 'year_joining'
plt.plot(id,age,color = 'y',ls = '-',label = 'age',marker = 'o')
plt.title('Multiple line plots of details of employee')
plt.xlabel('id')
plt.ylabel('Salary')
plt.show()
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
#eye iris datase-sepel and petal length
df_iris = sns.load_dataset('iris')
df_iris
```

```
---------------------------------------------------------------------------
TimeoutError                              Traceback (most recent call last)
/usr/lib/python3.6/urllib/request.py in do_open(self, http_class, req, **http_conn_args)
   1324                 h.request(req.get_method(), req.selector, req.data, headers,
-> 1325                           encode_chunked=req.has_header('Transfer-encoding'))
   1326             except OSError as err: # timeout error

/usr/lib/python3.6/http/client.py in request(self, method, url, body, headers, encode_chunked)
   1284         """Send a complete request to the server."""
-> 1285         self._send_request(method, url, body, headers, encode_chunked)
   1286
```

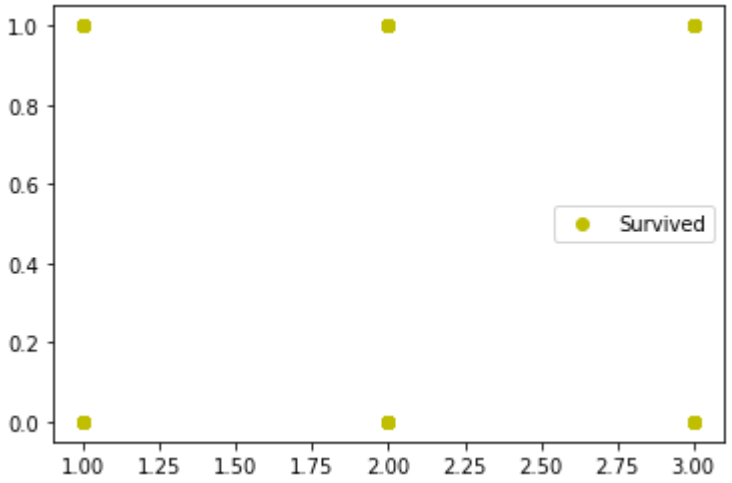
In [4]:
```
import seaborn as sns
```

```
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

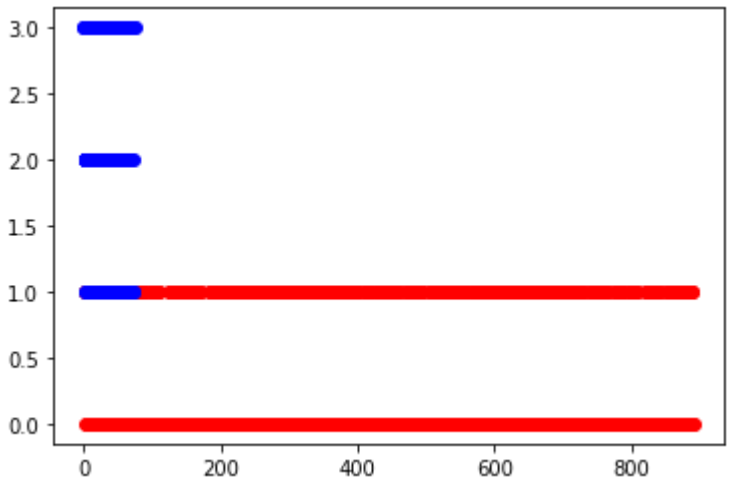
```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                 Name     Sex   A
ge  SibSp  \
0                             Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                              Heikkinen, Miss. Laina  female  2
6.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                            Allen, Mr. William Henry    male  3
5.0      0
...                                               ...     ...
...    ...
1304                               Spector, Mr. Woolf    male   N
aN      0
1305                     Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                     Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                              Ware, Mr. Frederick    male   N
aN      0
1308                          Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
1307      0              359309    8.0500  ...        S  1289.0
1308      1                2668   22.3583  ...        C   702.0

                                      Name_wiki Age_wiki  \
0                        Braund, Mr. Owen Harris     22.0
1     Cumings, Mrs. Florence Briggs (née Thayer)     35.0
2                          Heikkinen, Miss Laina     26.0
3              Futrelle, Mrs. Lily May (née Peel)     35.0
4                        Allen, Mr. William Henry     35.0
...                                           ...      ...
1304                         Spector, Mr. Woolf     23.0
```

```
1305        and maid, Doña Fermina Oliva y Ocana     39.0
1306                  Sæther, Mr. Simon Sivertsen     43.0
1307                  Ware, Mr. Frederick William     34.0
1308             Butrus-Youssef, Master Makhkhul       4.0

                                  Hometown      Boarded  \
0              Bridgerule, Devon, England  Southampton
1                  New York, New York, US    Cherbourg
2                      Jyväskylä, Finland  Southampton
3             Scituate, Massachusetts, US  Southampton
4     Birmingham, West Midlands, England  Southampton
...                                  ...          ...
1304                  London, England  Southampton
1305                    Madrid, Spain    Cherbourg
1306        Skaun, Sør-Trøndelag, Norway  Southampton
1307          Greenwich, London, England  Southampton
1308                   Sar'al[81], Syria    Cherbourg

                                 Destination Lifeboat  Body Class
0     Qu'Appelle Valley, Saskatchewan, Canada      NaN   NaN   3.0
1                     New York, New York, US        4   NaN   1.0
2                                New York City      14?   NaN   3.0
3                Scituate, Massachusetts, US        D   NaN   1.0
4                                New York City      NaN   NaN   3.0
...                                        ...      ...   ...   ...
1304                             New York City      NaN   NaN   3.0
1305                  New York, New York, US        8   NaN   1.0
1306                                      US      NaN  32MB   3.0
1307                             New York City      NaN   NaN   3.0
1308                      Detroit, Michigan, US        D   NaN   3.0

[1309 rows x 21 columns]
```

In [14]:
```
# the two colums should be numerical
plt.scatter(x = 'Pclass',y = 'Survived',data = df_titanic,color = 'y'
plt.legend()
plt.show()
```

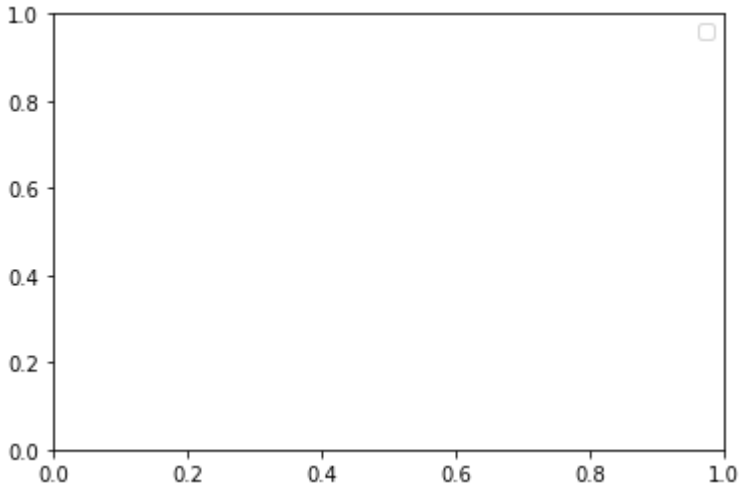


In [16]:
```
# Multiple scatter plot
plt.scatter(x = 'PassengerId',y = 'Survived',label = 'Survived',data
plt.scatter(x = 'Age_wiki',y = 'Class',label = 'Class',data = df_tita

plt.show()
plt.legend()
```

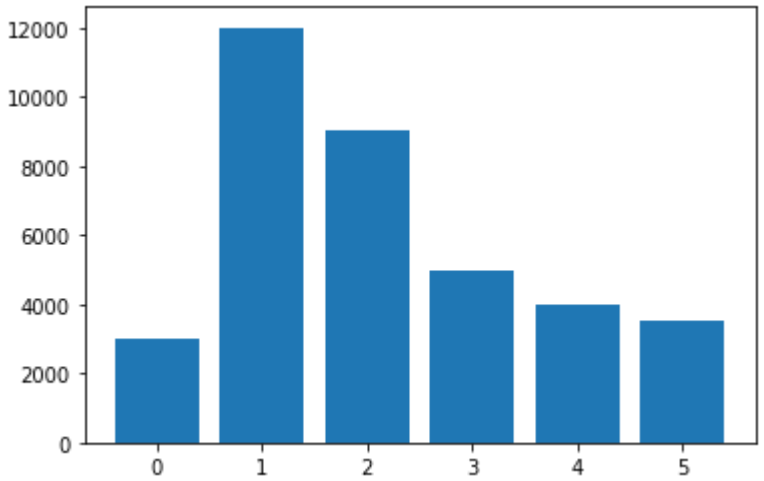



```
No handles with labels found to put in legend.
```

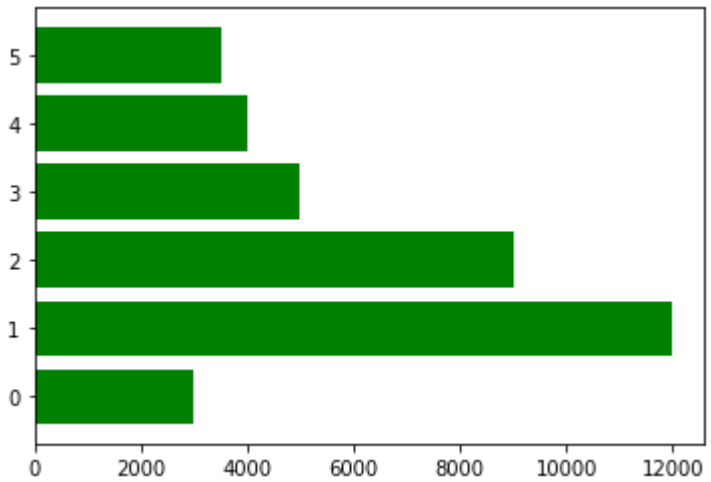
Out[16]:
```
<matplotlib.legend.Legend at 0x7f7a56077358>
```



In [10]:
```
# Bar plot without using dataset
amount = [3000,12000,9000,5000,4000,3500]
customer = ['john','nose','rick','Mia','Dinesh','nandu']
x = np.arange(len(customer))#compare categorical with numerical
#create a plot:
plt.bar(x = x ,height = amount)

#Display the plot:
plt.show()
```



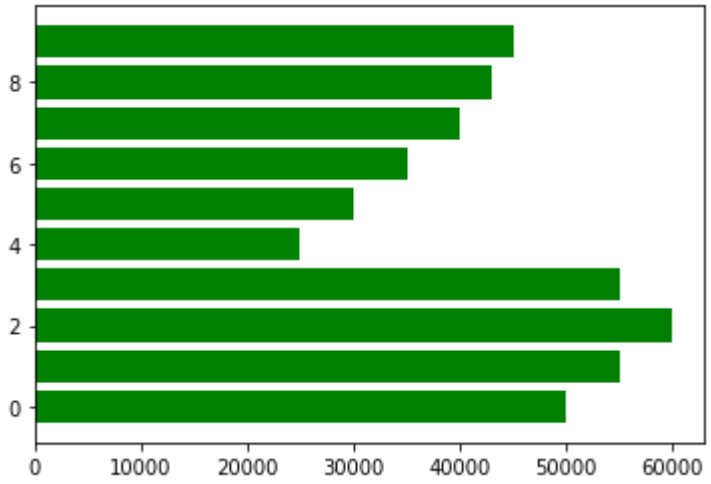
In [11]:
```
amount = [3000,12000,9000,5000,4000,3500]
customer = ['john','nose','rick','Mia','Dinesh','nandu']
x = np.arange(len(customer))#compare categorical with numerical
#create a plot:
plt.barh(y = x ,width = amount,color = 'g')

#Display the plot:
plt.show()
# for veritcal x and height
# for horizantal y and width
```



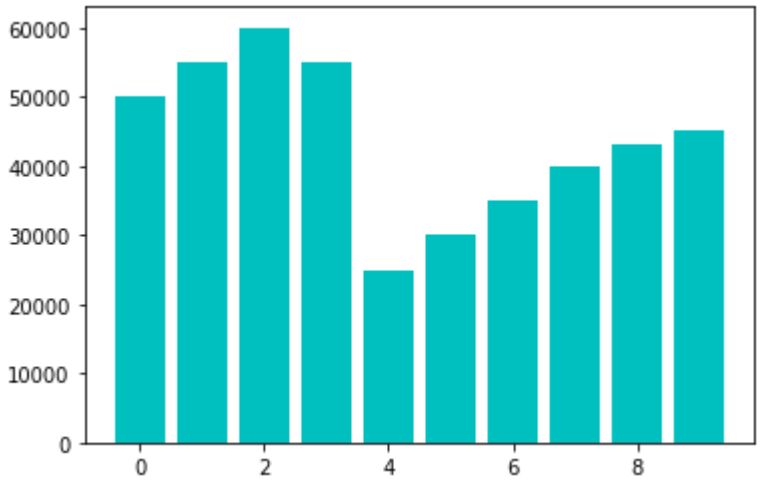
In [ ]:
```
name of employee names salary
```

In [12]:
```
employee_name = ['nandu','gowthu','banny','ammu','dsf','wefr','swapna
salary = [50000,55000,60000,55000,25000,30000,35000,40000,43000,45000
x = np.arange(len(employee_name))#compare categorical with numerical
#create a plot:
plt.barh(y = x ,width = salary,color = 'g')
```

```
#Display the plot:
plt.show()
```



In [13]:
```
x = np.arange(len(employee_name))#compare categorical with numerical
#create a plot:
plt.bar(x = x ,height = salary,color = 'c')

#Display the plot:
plt.show()
```



In [21]:
```
from sklearn.datasets import load_iris
iris = load_iris
```

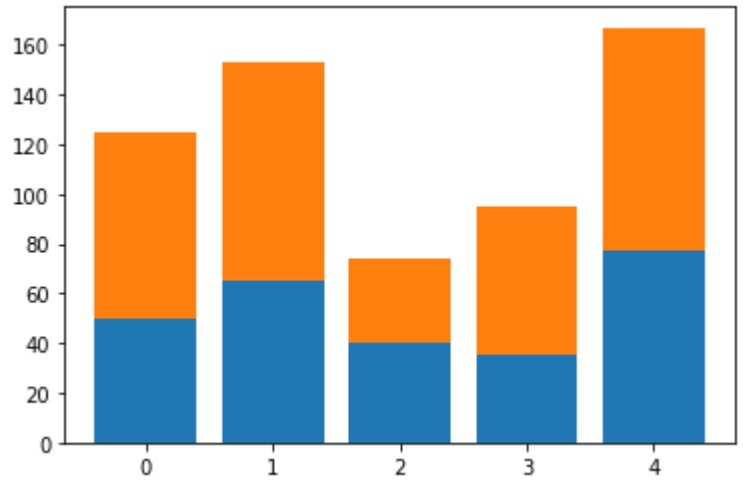
```
---------------------------------------------------------------------------
--------
ModuleNotFoundError                       Traceback (most recent ca
ll last)
<ipython-input-21-b88a4a0b9fa4> in <module>
----> 1 from sklearn.datasets import load_iris
      2 iris = load_iris

ModuleNotFoundError: No module named 'sklearn'
```

In [ ]:
```
#eda - Explotting data analysit
# used in datasets
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Stacked Bar plot
# Create a Data
python_marks = (50,65,40,35,77)
java_marks = (75,88,34,60,90)

#Set the Position of Bar Plot
index = np.arange(5)
plt.bar(x = index,height = python_marks,label='Python')
plt.bar(x = index,height = java_marks,bottom = python_marks,label = '

plt.xtics(tics = index,label = {'A','B','C','D','E'})
# Adding the Legend:
plt.legend()
plt.show()
```

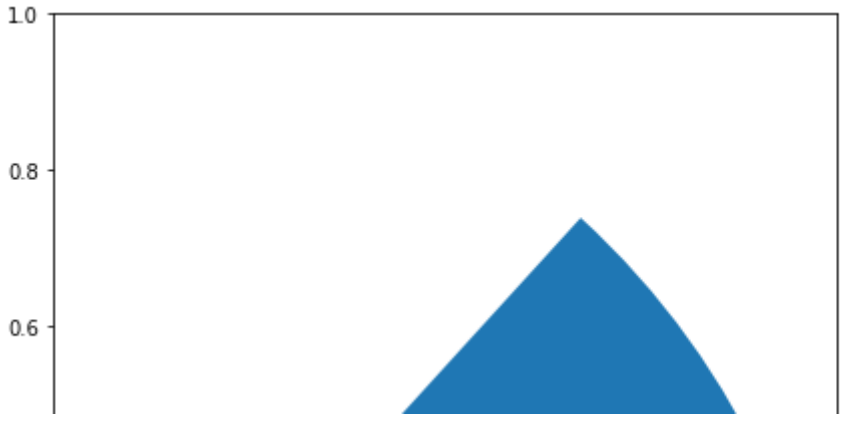
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-1-fdcc1f8baf52> in <module>
     12 plt.bar(x = index,height = java_marks,bottom = python_marks,label = 'Java')
     13
---> 14 plt.xtics(tics = index,label = {'A','B','C','D','E'})
     15 # Adding the Legend:
     16 plt.legend()

AttributeError: module 'matplotlib.pyplot' has no attribute 'xtics'
```



In [2]:
```
# Pie chart
#one column is chatagorical and another column is numerical values
plt.figure(figsize = (7,10))

countries = ('Germany','France','USA','Norway','Spain')
population =(8.26,6.7,37.22,5.33,4.90)
plt.pie(x = population,labels = countries , autopct = '%1.1f%')
circle = plt.Circle(xy = (0,0),radius = 0.5,color = 'b')

# Gca -A circular axis
plt.gcf()
# gca_get the circular figure:
plt.gca().add_artist(circle)

# Add tite of your Donut pie chart
```

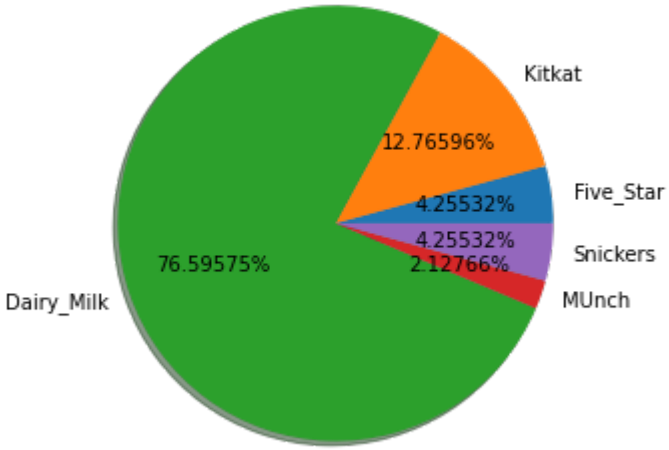
```
plt.title('Distribution of population')
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-2-3f2c5cff7ca8> in <module>
      5 countries = ('Germany','France','USA','Norway','Spain')
      6 population =(8.26,6.7,37.22,5.33,4.90)
----> 7 plt.pie(x = population,labels = countries , autopct = '%1.1f%')
      8 circle = plt.Circle(xy = (0,0),radius = 0.5,color = 'b')
      9

~/my_project_env/lib/python3.6/site-packages/matplotlib/pyplot.py in pie(x, explode, labels, colors, autopct, pctdistance, shadow, labeldistance, startangle, radius, counterclock, wedgeprops, textprops, center, frame, rotatelabels, normalize, data)
   2832         wedgeprops=wedgeprops, textprops=textprops, center=center,
   2833         frame=frame, rotatelabels=rotatelabels, normalize=normalize,
-> 2834         **({"data": data} if data is not None else {}))
   2835
   2836

~/my_project_env/lib/python3.6/site-packages/matplotlib/__init__.py in inner(ax, data, *args, **kwargs)
   1445     def inner(ax, *args, data=None, **kwargs):
   1446         if data is None:
-> 1447             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1448
   1449         bound = new_sig.bind(ax, *args, **kwargs)

~/my_project_env/lib/python3.6/site-packages/matplotlib/axes/_axes.py in pie(self, x, explode, labels, colors, autopct, pctdistance, shadow, labeldistance, startangle, radius, counterclock, wedgeprops, textprops, center, frame, rotatelabels, normalize)
   3103                 yt = y + pctdistance * radius * math.sin(thetam)
   3104                 if isinstance(autopct, str):
-> 3105                     s = autopct % (100. * frac)
   3106                 elif callable(autopct):
   3107                     s = autopct(100. * frac)

ValueError: incomplete format
```

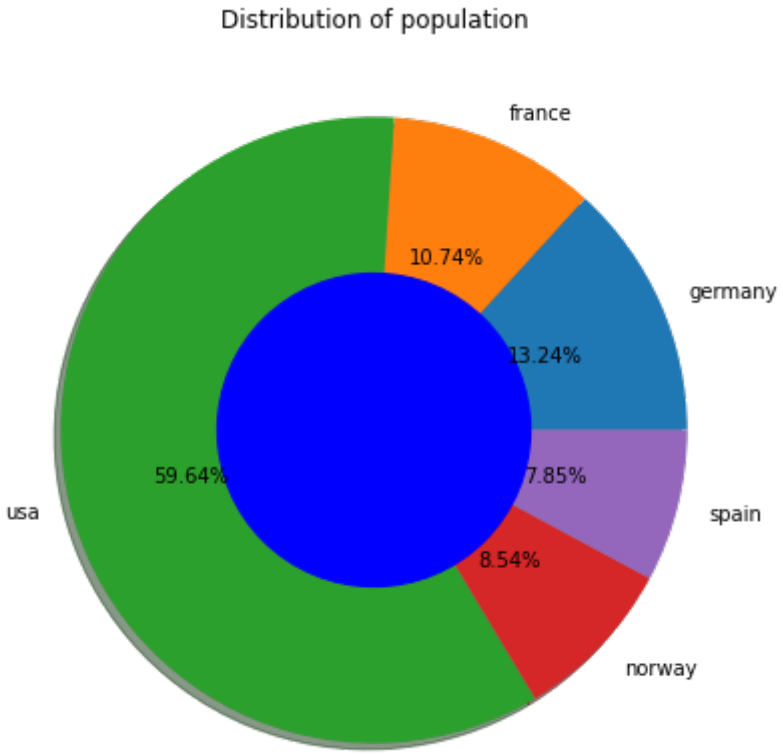


In [ ]:

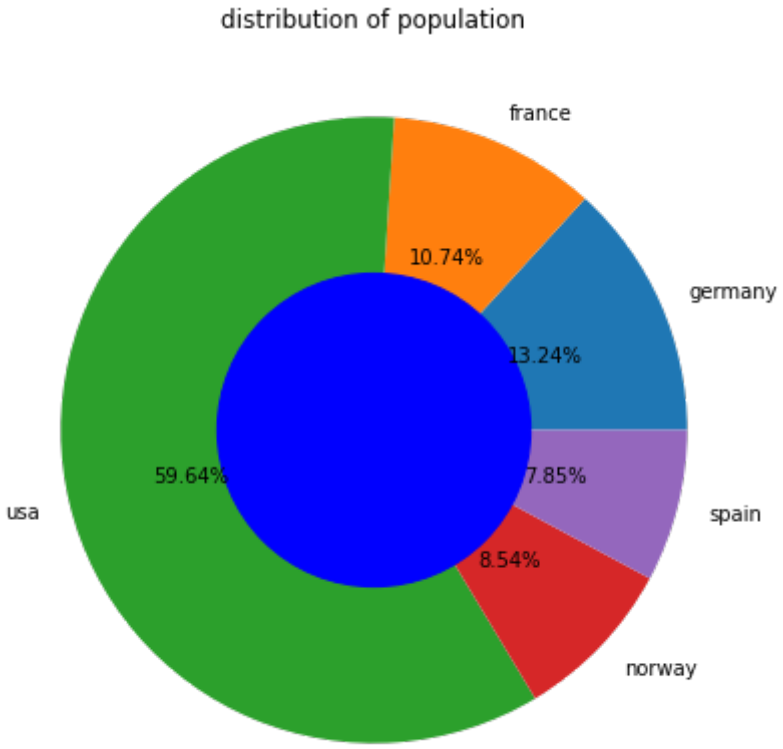
In [30]:
```
plt.figure(figsize = (5,5))
product_name = ('Five_Star','Kitkat','Dairy_Milk','MUnch','Snickers')
product_price = (10,30,180,5,10)
plt.pie(x = product_price , labels = product_name,autopct = '%1.5f%%')
plt.show()
```



In [27]:
```
plt.figure(figsize =(7,10))

countries = ('germany','france','usa','norway','spain')
population =(8.26,6.7,37.22,5.33,4.90)

plt.pie( x=population,labels=countries ,autopct = '%1.2f%%', shadow =
circle = plt.Circle(xy = (0,0),radius = 0.5,color = 'b')
# Gca -A circular axis
plt.gcf()
# gca_get the circular figure:
plt.gca().add_artist(circle)

# Add tite of your Donut pie chart
plt.title('Distribution of population')

plt.show()
```



In [29]:
```
plt.figure(figsize =(7,10))

countries = ('germany','france','usa','norway','spain')
population =(8.26,6.7,37.22,5.33,4.90)

plt.pie( x=population,labels=countries ,autopct = '%1.2f%%')

circle =plt.Circle(xy=(0,0), radius =0.5,color ='b')
# gca
plt.gcf()
# gca get the circular figure
plt.gca().add_artist(circle)
# add a title of your donut pie chart
plt.title('distribution of population')
plt.show()
```



In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                 Name     Sex   A
ge  SibSp  \
0                              Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                               Heikkinen, Miss. Laina  female  2
6.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                            Allen, Mr. William Henry    male  3
5.0      0
...                                                ...     ...
...    ...
1304                               Spector, Mr. Woolf    male   N
aN      0
1305                     Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                     Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                              Ware, Mr. Frederick    male   N
aN      0
1308                          Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
1307      0              359309    8.0500  ...        S  1289.0
1308      1                2668   22.3583  ...        C   702.0

                                      Name_wiki Age_wiki  \
0                         Braund, Mr. Owen Harris     22.0
1     Cumings, Mrs. Florence Briggs (née Thayer)     35.0
2                           Heikkinen, Miss Laina     26.0
3             Futrelle, Mrs. Lily May (née Peel)     35.0
4                         Allen, Mr. William Henry     35.0
...                                            ...      ...
1304                        Spector, Mr. Woolf     23.0
1305       and maid, Doña Fermina Oliva y Ocana     39.0
```

```
1306                 Sæther, Mr. Simon Sivertsen     43.0
1307                Ware, Mr. Frederick William     34.0
1308            Butrus-Youssef, Master Makhkhul      4.0

                                  Hometown      Boarded  \
0             Bridgerule, Devon, England  Southampton
1                 New York, New York, US    Cherbourg
2                     Jyväskylä, Finland  Southampton
3           Scituate, Massachusetts, US  Southampton
4     Birmingham, West Midlands, England  Southampton
...                                  ...          ...
1304                  London, England  Southampton
1305                    Madrid, Spain    Cherbourg
1306        Skaun, Sør-Trøndelag, Norway  Southampton
1307          Greenwich, London, England  Southampton
1308                   Sar'al[81], Syria    Cherbourg

                                Destination Lifeboat  Body Class
0     Qu'Appelle Valley, Saskatchewan, Canada      NaN   NaN   3.0
```

In [ ]:
```
Histogram
always contains with numerical columns
```

In [ ]:
```
```

In [ ]:
```
plt.title('Distribution of Sepal width')
plt.xlabel('sepal width')
plt.ylabel('Frequency')
plt.show()
```

In [ ]:
```
#Plot the Histogram with multiple bins  and add Grid:

plt.his(x = df['sepal_width'],color = 'r',bin = 10)
plt.title('Distribution of Sepal width')
plt.xlabel('sepal width')
plt.ylabel('Frequency')

#Plot the Grid
plt.grid()
plt.show()
```

In [ ]:
```
# Plot Multiple histogram for all the numerical col in my dataset
df.plot.his(subplot = True,layout = (2,2),
         figsize = (7,4))
plt.tight_layout()
plt.show()
```

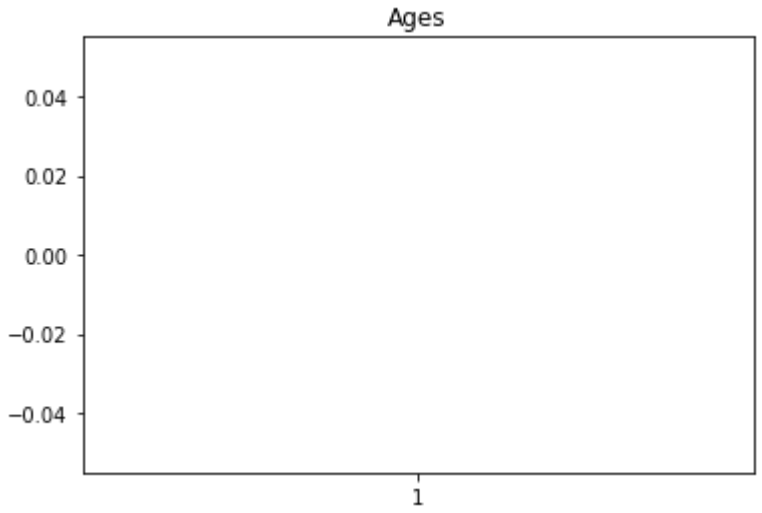
In [ ]:
```
#Vertical Box plot:
plt.boxplot(x = df['petal_length'])# it shows mean,median,max
#plt.boxplot(x = df['petal_length'],vert = False)
plt.title('Distribution of Sepal Width')
plt.show()
```

In [3]:
```
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

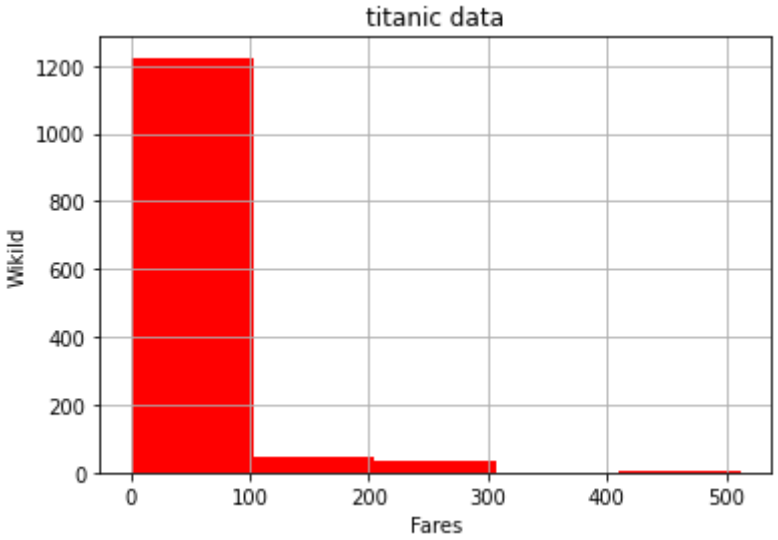
```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                 Name     Sex   A
ge  SibSp  \
0                               Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                                Heikkinen, Miss. Laina  female  2
6.0      0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                              Allen, Mr. William Henry    male  3
5.0      0
...                                                 ...     ...
...    ...
1304                                 Spector, Mr. Woolf    male   N
aN      0
1305                       Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                       Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                                Ware, Mr. Frederick    male   N
aN      0
1308                            Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
1307      0              359309    8.0500  ...        S  1289.0
1308      1                2668   22.3583  ...        C   702.0

                                      Name_wiki Age_wiki  \
0                         Braund, Mr. Owen Harris     22.0
1     Cumings, Mrs. Florence Briggs (née Thayer)     35.0
2                           Heikkinen, Miss Laina     26.0
3             Futrelle, Mrs. Lily May (née Peel)     35.0
4                         Allen, Mr. William Henry     35.0
...                                            ...      ...
1304                        Spector, Mr. Woolf     23.0
1305       and maid, Doña Fermina Oliva y Ocana     39.0
```

```
1306                 Sæther, Mr. Simon Sivertsen     43.0
1307                Ware, Mr. Frederick William     34.0
1308            Butrus-Youssef, Master Makhkhul      4.0

                                  Hometown      Boarded  \
0            Bridgerule, Devon, England  Southampton
1                New York, New York, US    Cherbourg
2                    Jyväskylä, Finland  Southampton
3           Scituate, Massachusetts, US  Southampton
4     Birmingham, West Midlands, England  Southampton
...                                  ...          ...
1304                  London, England  Southampton
1305                    Madrid, Spain    Cherbourg
1306       Skaun, Sør-Trøndelag, Norway  Southampton
1307         Greenwich, London, England  Southampton
1308                  Sar'al[81], Syria    Cherbourg

                                Destination Lifeboat  Body Class
0     Qu'Appelle Valley, Saskatchewan, Canada      NaN   NaN   3.0
1                     New York, New York, US        4   NaN   1.0
2                                New York City      14?   NaN   3.0
3                Scituate, Massachusetts, US        D   NaN   1.0
4                                New York City      NaN   NaN   3.0
...                                        ...      ...   ...   ...
1304                             New York City      NaN   NaN   3.0
```

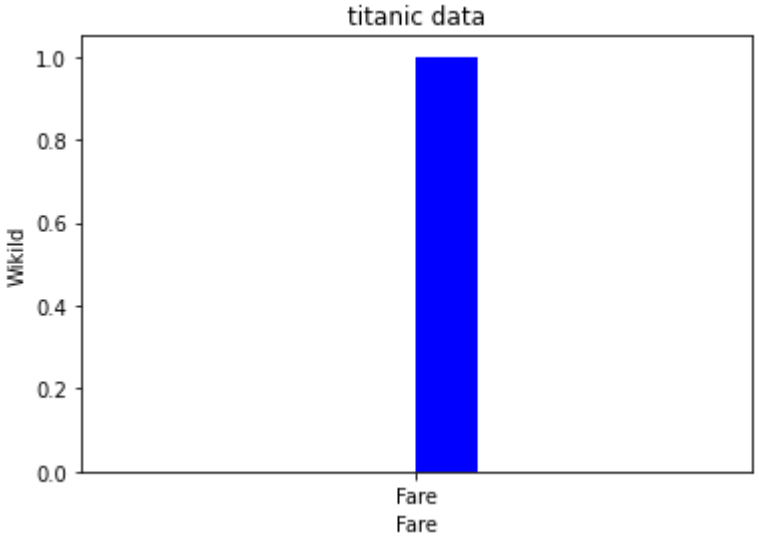
In [16]:
```
#Vertical Box plot:
plt.boxplot(x = df_titanic['WikiId'])# it shows mean,median,max
#plt.boxplot(x = df['petal_length'],vert = False)
plt.title('Ages')
plt.show()
```



In [17]:
```
# plot the histogram with multiple bins add grid :
plt.hist(x = df_titanic["Fare"],color ='r',bins=5)
plt.title("titanic data")
plt.xlabel("Fares")
plt.ylabel("WikiId")
# plot the grid:
plt.grid()
# display the plot:
plt.show()
```



In [14]:
```
plt.hist(x =['Fare'],color ='b')
# add title:
plt.title("titanic data")
# add laebels:
plt.xlabel("Fare")
plt.ylabel("WikiId")
# display the plot
plt.show()
```



Exploratory data Analysis

#what is data

1.it is a collection of information

2.Records of any instances

In [ ]:
```
## Variable Type
1.Numerical - Discrete,continuous
2.categorical-Binominal nominal ordinal
```

In [ ]:
```
### categorical Types
1.Nominal Data:
    - Nominal Data has no order
    - It has one or more than two categories
    - It represented in discrete units and are used for labeled varia
2.Binary Data:
    - Binary data has no order and strictly two categories
3.ordinal data:
    - Ordinal Data is Ordered Data
    - It represented as discrete and ordered units
    for an exmple - dataset from primary,secondaryschool
```

In [ ]:
```
### Obtain Trimmed Mean:

- Arithmetic obtained by ignoring lowest and highest alpha
- trimmed mean is also called as truncated mean
- trimmed mean is one of the methodo
```

Distribution of the Data:

- The distribution is a summary of the Frequency of values
- The Distribution of the data is the given information of the shape
- The Distribution given the spread of the data

Measure of Dispersion

In [ ]:
```
- The Measure of Dispersion is refer to variability of the data
- The Variability is the measure of how close or far the data lies fr
- calcuate by using range
```

In [ ]:
```
'Range'
1.Range is the difference between the smallest and largest observatio
2.The formula:
    range = Xn - X1
    -- Xn is the largest value
    -- X1 is the smallest value
```

In [2]:
```
import pandas as pd
import numpy as np

year = pd.Series((1999,1995,1955,2000))
range = year.max() - year.min()
range
```

Out[2]: 45

In [ ]:
```
# Variance:(vvm)
1.Variance is the Arithmetic mean of square of deviation taken from t
2.It show how far your data is spread out from the mean
```

In [4]:
```
prices = pd.Series([0,0,34,7,8,9,34,66,78,34,89,0,4,1])
```

In [7]:
```
# Find the variance:
print(prices.var())
# Obtain Standard Deviation
print(prices.std())
```

```
967.3846153846154
31.102807194602473
```

In [ ]:
```
# Coefficient of Variation:

1.The coefficient of the variation is the statistical measure of disp
```

In [13]:
```
from scipy.stats import variation
scipy.stats.variation(prices)
```

```
---------------------------------------------------------------------
--------
NameError                                 Traceback (most recent ca
ll last)
<ipython-input-13-726f935eae27> in <module>
      1 from scipy.stats import variation
----> 2 scipy.stats.variation(prices)

NameError: name 'scipy' is not defined
```

In [12]:
```
prices of watches range from 1 to 500
find range mean median mode trimmed mean coefficient
```

```
  File "<ipython-input-12-15419afdb1f9>", line 1
    prices of watches range from 1 to 500
               ^
SyntaxError: invalid syntax
```

In [16]:
```
from scipy.stats import variation
scipy.stats.variation(prices)

prices = pd.Series([10,100,200,300,400,500,350,440,480])
range = prices.max() - prices.min()
print(range)
# Find the variance:
print(prices.var())
# Obtain Standard Deviation
print(prices.std())
print(prices.mean())
print(prices.median())
print(prices.mode())
print(scipy.stats.trim_mean(prices,proportiontocut = 0.30))
```

```
---------------------------------------------------------------------------
--------
NameError                                 Traceback (most recent ca
ll last)
<ipython-input-16-cb07a5687ccd> in <module>
      1 from scipy.stats import variation
```

In [ ]:
```
import numpy as np
import pandas as pd
```

Handling Non - Numeric Data

In [ ]:
```
#encoding:convert catogerical column into numerical column
# there are many ways to encode a Categorical column:
# N-1 Dummy Encoding(2 outcomes)
# One - Hot Encoding(3 outcomes)
# Label Encoding
# Ordinal Encoding
# Frequency Encoding
# Target Encoding
```

In [24]:
```
df_market = pd.read_csv('auto-mpg.csv')
df_market
```

Out[24]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nam
0	18.0	8	307.0	130	3504	12.0	70	1	chevrole chevell malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buic skylar 32
2	18.0	8	318.0	150	3436	11.0	70	1	plymoutl satellit
3	16.0	8	304.0	150	3433	12.0	70	1	am rebel ss
4	17.0	8	302.0	140	3449	10.5	70	1	for torin
...	...	...	...	...	...	...	...	...	..
393	27.0	4	140.0	86	2790	15.6	82	1	for mustan g
394	44.0	4	97.0	52	2130	24.6	82	2	vw picku
395	32.0	4	135.0	84	2295	11.6	82	1	dodg rampag
396	28.0	4	120.0	79	2625	18.6	82	1	for range
397	31.0	4	119.0	82	2720	19.4	82	1	chev s-1

398 rows × 9 columns

In [10]:
```
# origin is a categorical  column
df_market['origin'].value_counts()
```

Out[10]:
```
1    249
3     79
2     70
Name: origin, dtype: int64
```

In [12]:
```
# Sort categorical column into numerical column machine could not und
## (N-1) Dummy Encoding
# Create a Dummy Variable for '\Origin column':
# Drop_first = 'we are just fropping row to get Dummy variable'
pd.get_dummies(df_market,columns = ['origin'],drop_first = True)
```

Out[12]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	origi
0	18.0	8	307.0	130	3504	12.0	70	chevrolet chevelle malibu	
1	15.0	8	350.0	165	3693	11.5	70	buick skylark 320	
2	18.0	8	318.0	150	3436	11.0	70	plymouth satellite	
3	16.0	8	304.0	150	3433	12.0	70	amc rebel sst	
4	17.0	8	302.0	140	3449	10.5	70	ford torino	
...	...	...	...	...	...	...	...	...	
393	27.0	4	140.0	86	2790	15.6	82	ford mustang gl	
394	44.0	4	97.0	52	2130	24.6	82	vw pickup	
395	32.0	4	135.0	84	2295	11.6	82	dodge rampage	
396	28.0	4	120.0	79	2625	18.6	82	ford ranger	
397	31.0	4	119.0	82	2720	19.4	82	chevy s-10	

398 rows × 10 columns

In [23]:
```
# One - Hot Encoding(3 outcomes)
one_hot_encoded_data = pd.get_dummies(data, columns = ['Remarks', 'Ge
print(one_hot_encoded_data)
```

```
---------------------------------------------------------------------
-------
```

In [13]:
```
df_market['origin','weight'].value_counts()
```

```
---------------------------------------------------------------------
--------
KeyError                                  Traceback (most recent ca
ll last)
~/my_project_env/lib/python3.6/site-packages/pandas/core/indexes/ba
se.py in get_loc(self, key, method, tolerance)
   2897             try:
-> 2898                 return self._engine.get_loc(casted_key)
   2899             except KeyError as err:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.P
yObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.P
yObjectHashTable.get_item()

KeyError: ('origin', 'weight')

The above exception was the direct cause of the following exceptio
n:

KeyError                                  Traceback (most recent ca
ll last)
<ipython-input-13-8ff262e75e56> in <module>
----> 1 df_market['origin','weight'].value_counts()

~/my_project_env/lib/python3.6/site-packages/pandas/core/frame.py i
n __getitem__(self, key)
   2904             if self.columns.nlevels > 1:
   2905                 return self._getitem_multilevel(key)
-> 2906             indexer = self.columns.get_loc(key)
   2907             if is_integer(indexer):
   2908                 indexer = [indexer]

~/my_project_env/lib/python3.6/site-packages/pandas/core/indexes/ba
se.py in get_loc(self, key, method, tolerance)
   2898                 return self._engine.get_loc(casted_key)
   2899             except KeyError as err:
-> 2900                 raise KeyError(key) from err
   2901
   2902         if tolerance is not None:

KeyError: ('origin', 'weight')
```

In [26]:
```
# One Hot Encoding
import sklearn
# Import Sklearn
from  sklearn.preprocessing import OneHotEncoder

# Create an Instance
```

```
encode = OneHotEncoder
df_encode = pd.DataFrame(encode.fit_transform(df_market[['origin']]).
                         columns = ['origin_europe','origin_japan','or
# Merge with main data frame:
# Axis = 1 : It stands for Column wise:
# Axis = 0 : Concatenation by means for Row wise:

df_encode = pd.concat([df_market,df_encode],axis = 1)
df_encode
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-26-2ad7892ab30f> in <module>
      6 # Create an Instance
      7 encode = OneHotEncoder
----> 8 df_encode = pd.DataFrame(encode.fit_transform(df_market[['o
rigin']]).toarray(),
      9                          columns = ['origin_europe','origin_
japan','origin_usa'])
     10 # Merge with main data frame:

TypeError: fit_transform() missing 1 required positional argument:
'X'
```

In [30]:
```
# Label Encoding

# The label Encoding consider a level in a Categorical variable by 'A

from sklearn.preprocessing import LabelEncoder

# Create an Instance
labelencoder = LabelEncoder()

#Fit the Encoder
df_market['Encoded_performance_of_a_car'] = labelencoder.fit_transfor

# Display the data
df_market
```

Out[30]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nam
0	18.0	8	307.0	130	3504	12.0	70	1	chevrole chevell malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buic skylar 32
2	18.0	8	318.0	150	3436	11.0	70	1	plymoutl satellit
3	16.0	8	304.0	150	3433	12.0	70	1	am rebel ss
4	17.0	8	302.0	140	3449	10.5	70	1	for torin
...	...	...	...	...	...	...	...	...	..

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nam
393	27.0	4	140.0	86	2790	15.6	82	1	for mustan g
394	44.0	4	97.0	52	2130	24.6	82	2	vv picku
395	32.0	4	135.0	84	2295	11.6	82	1	dodg rampag
396	28.0	4	120.0	79	2625	18.6	82	1	for range

In [32]:
```
df_market['Encoded_performance_of_a_car'].value_counts()
```

Out[32]:
```
17    21
19    18
72    18
70    17
61    17
      ..
26     1
54     1
22     1
43     1
0      1
Name: Encoded_performance_of_a_car, Length: 82, dtype: int64
```

In [ ]:
```
# Interpretation

# Low Performance ranges from 17-27
# Average Performance ranges from 37 - 43
# Super performance ranges from 61-72
```

In [ ]:
```
# Feature Scaling:
 - Feature Scaling is also called as Data Normalization
 - It is the Technique used to transform the data into a common scale
 - Since the Feature has Various ranges it becomes the Necessary step
```

import pandas as pd import numpy as np import matplotlib.pyplot as plt

df_car = pd.read_csv('auto-mpg.csv') df_car

In [ ]:
```
# Methods in Feature Scaling

1.Standardization or z-score Normalization
2 MinMax Normalization
```

Standardization or z-score Normalization

In [ ]:
```
 1.Standardization transfrom the data such that the data has mean and
2 The Procedure involves subsracting the mean from the Observation
```

In [10]:
```
from sklearn.preprocessing import StandardScaler
print('Minimum   value Before Transformation :',df_car.weight.min(),'
     'maximium value Before Transformation:',df_car.weight.max())
# Create an instance
standard_scale = StandardScaler()
```

```
# Fit the StandardScaler
# Transform the data

df_car['Scaled_weight'] = standard_scale.fit_transform(df_car[['weigh

print('Minimum   value After Transformation :',df_car['Scaled_weight'
     'maximium value After Transformation:',df_car['Scaled_weight'].m
```

```
Minimum   value Before Transformation : 1613
maximium value Before Transformation: 5140
Minimum   value After Transformation : -1.6049434405635041
 maximium value After Transformation: 2.565185359572092
```

Min-Max Normalization

In [ ]:
```
1.It perform the Linear transformation on the Original Data
2.The Min-Max Normalization is given as
3.[(xnorm) = X - Xmin/Xmax - Xmin]
4.Should come in range of 0 to 1
```

In [15]:
```
# Importing MinMaxNormalization from sklearn
from sklearn.preprocessing import MinMaxScaler

# Create An Instance

min_max = MinMaxScaler()

# Fit and transform of weight column:
df_car['min_max_scaled_weight'] = min_max.fit_transform(df_car[['weig

#minimum and maximum of Normalization of weight:
df_car['min_max_scaled_weight'].min(),df_car['min_max_scaled_weight']
```

Out[15]: (0.0, 1.0)

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [16]:
```
import seaborn as sns
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                     Name     Sex   A
ge  SibSp  \
0                                 Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                                  Heikkinen, Miss. Laina  female  2
6.0      0
3            Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                               Allen, Mr. William Henry    male  3
5.0      0
...                                                  ...     ...
...    ...
1304                                  Spector, Mr. Woolf    male   N
aN      0
1305                        Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                        Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                                 Ware, Mr. Frederick    male   N
aN      0
1308                             Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
1307      0              359309    8.0500  ...        S  1289.0
1308      1                2668   22.3583  ...        C   702.0

                                       Name_wiki Age_wiki  \
0                          Braund, Mr. Owen Harris     22.0
1     Cumings, Mrs. Florence Briggs (née Thayer)     35.0
2                            Heikkinen, Miss Laina     26.0
3              Futrelle, Mrs. Lily May (née Peel)     35.0
4                         Allen, Mr. William Henry     35.0
...                                            ...      ...
1304                         Spector, Mr. Woolf     23.0
1305        and maid, Doña Fermina Oliva y Ocana     39.0
```

```
1306                 Sæther, Mr. Simon Sivertsen     43.0
1307                Ware, Mr. Frederick William     34.0
1308            Butrus-Youssef, Master Makhkhul      4.0

                                  Hometown      Boarded  \
0             Bridgerule, Devon, England  Southampton
1                 New York, New York, US    Cherbourg
2                     Jyväskylä, Finland  Southampton
3           Scituate, Massachusetts, US  Southampton
4     Birmingham, West Midlands, England  Southampton
...                                  ...          ...
1304                  London, England  Southampton
1305                    Madrid, Spain    Cherbourg
1306        Skaun, Sør-Trøndelag, Norway  Southampton
1307          Greenwich, London, England  Southampton
1308                   Sar'al[81], Syria    Cherbourg
```

In [21]:
```
from sklearn.preprocessing import StandardScaler
print('Minimum   value Before Transformation :',df_titanic.Age_wiki.m
     'maximium value Before Transformation:',df_titanic.Age_wiki.max(
# Create an instance
standard_scale = StandardScaler()

# Fit the StandardScaler
# Transform the data

df_titanic['Scaled_weight'] = standard_scale.fit_transform(df_titanic

print('Minimum   value After Transformation :',df_titanic['Scaled_wei
     'maximium value After Transformation:',df_titanic['Scaled_weight
```

```
Minimum   value Before Transformation : 0.17
maximium value Before Transformation: 74.0
Minimum   value After Transformation : -2.1264019585284712
 maximium value After Transformation: 3.2416200575768737
```

In [20]:
```
# Importing MinMaxNormalization from sklearn
from sklearn.preprocessing import MinMaxScaler

# Create An Instance

min_max = MinMaxScaler()

# Fit and transform of weight column:
df_titanic['min_max_scaled_weight'] = min_max.fit_transform(df_titani

#minimum and maximum of Normalization of weight:
df_titanic['min_max_scaled_weight'].min(),df_titanic['min_max_scaled
```

Out[20]:
```
(0.0, 0.9999999999999999)
```

In [ ]:
```
Working with Missing Data in Pandas
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_csv('k_circle_sales.csv')
df
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	O
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.3	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.1	
4	NCD19	8.930	Low Fat	0.000000	Household	53.9	
...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.2	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.5	

8523 rows × 13 columns

In [5]:
```
# Summary stats for Categorical Column:
from warnings import filterwarnings
filterwarnings('ignore')
df.describe(include = [np.object])
```

Out[5]:

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Loc
count	8523	8523	8523	8523	6113	
unique	1559	5	16	10	3	
top	FDG33	Low Fat	Fruits and Vegetables	OUT027	Medium	
freq	10	5089	1232	935	2793	

In [11]:
```
# Summary stats for Numerical Column:
from warnings import filterwarnings
filterwarnings('ignore')
df.describe()
```

Out[11]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7774.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	11.676740	0.066132	140.998838	1997.831867	2181.288914
std	5.776851	0.051598	62.258099	8.371760	1706.499616
min	0.000000	0.000000	31.300000	1985.000000	33.290000
25%	7.720000	0.026989	93.800000	1987.000000	834.247400

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
50%	11.800000	0.053931	142.700000	1999.000000	1794.331000
75%	16.500000	0.094585	185.650000	2004.000000	3101.296400

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Item_Identifier               0
Item_Weight                 749
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type       2050
Outlet_Type                   0
Item_Outlet_Sales             0
Profit                        0
dtype: int64
```

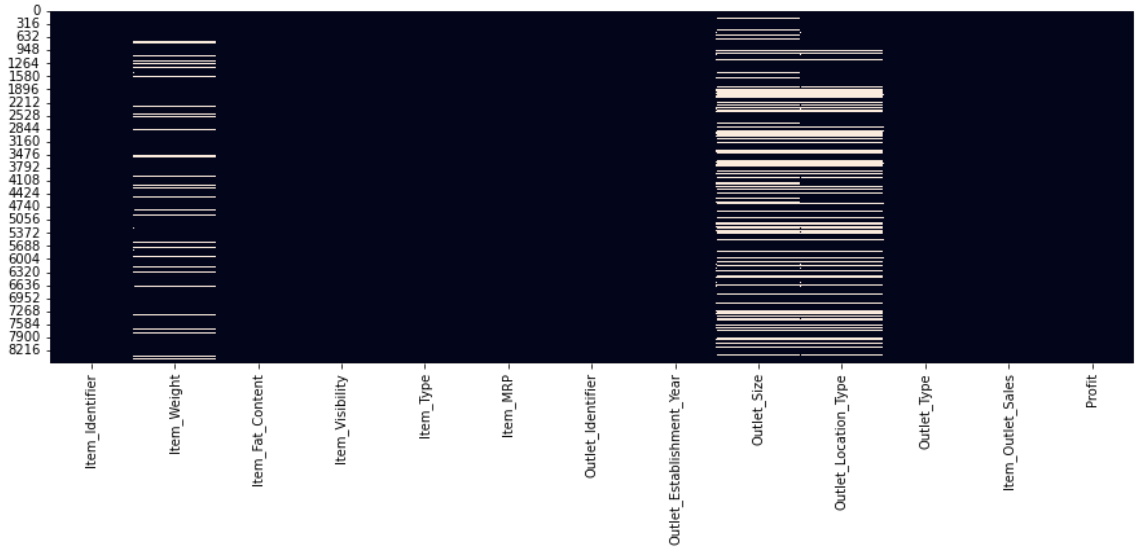
In [10]:
```
# Another method for null value
missing_values = df.isnull().sum()

# Check for missing values
total = df.isnull().sum().sort_values(ascending = False)

# Calculate the percentage of missing values:
percent = ((df.isnull().sum() / df.shape[0])*100)

# Sort the values in desecnding order
percent = percent.sort_values(ascending = False)

# Concatenate the Total Missing values:
missing_data = pd.concat([total,percent],axis = 1,
                         keys = ['Total missing values','ercentage of
# Add the Data types:
missing_data['Data(Dtype)'] = df[missing_data.index].dtypes
missing_data
```

Out[10]:

	Total missing values	ercentage of Missing Values	Data(Dtype)
Outlet_Size	2410	28.276428	object
Outlet_Location_Type	2050	24.052564	object
Item_Weight	749	8.787985	float64
Profit	0	0.000000	float64
Item_Outlet_Sales	0	0.000000	float64
Outlet_Type	0	0.000000	object
Outlet_Establishment_Year	0	0.000000	int64
Outlet_Identifier	0	0.000000	object
Item_MRP	0	0.000000	float64
Item_Type	0	0.000000	object
Item_Visibility	0	0.000000	float64
Item_Fat_Content	0	0.000000	object

Total missing values ercentage of Missing Values Data(Dtype)

In [25]:
```
# visualize the null values by means of heatmap:
# 1. set the figure size:
plt.rcParams['figure.figsize']  = [15,5]
# plot the heat map:
sns.heatmap(df.isnull(),cbar = False)

# display the heatmap:
plt.show()
```



In [14]:
```
# Drap the Rows which has Null Values
#1st Strategy
# To Eliminate the Null values from our Dataset

df_sales_drop = df.dropna()

# Display the shape after dropping null values from the data:
df_sales_drop.shape
```

Out[14]: (5364, 13)

In [16]:
```
# Sanity check for Null Values:
df.isnull().sum()
```

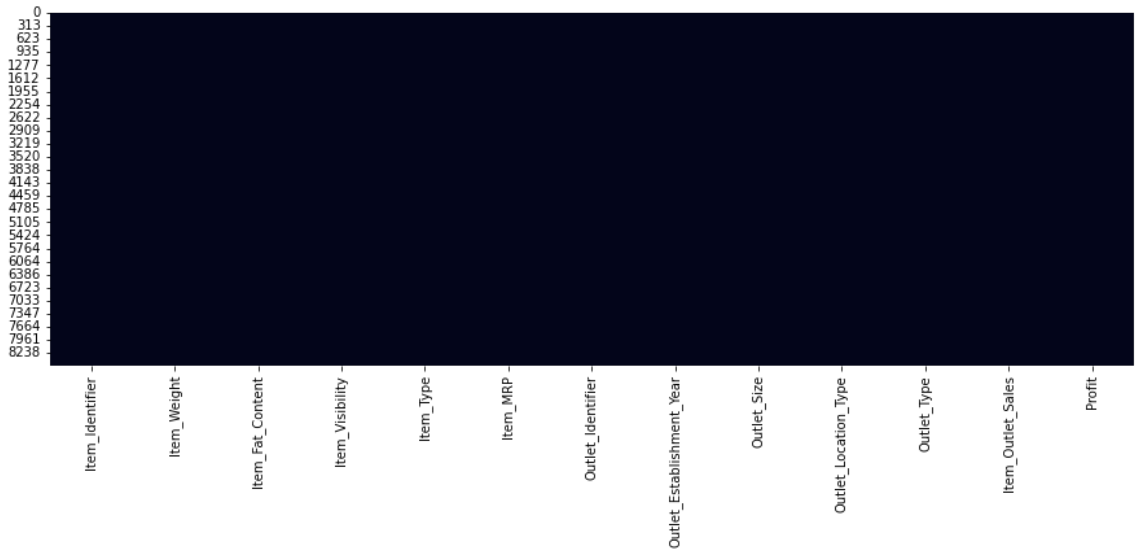
Out[16]:
```
Item_Identifier                 0
Item_Weight                   749
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  2410
Outlet_Location_Type         2050
Outlet_Type                     0
Item_Outlet_Sales               0
Profit                          0
dtype: int64
```

In [28]:
```
# To eliminate the null values
df_sales_drop = df.dropna(how='any',inplace = True)
```

In [29]:
```
# Sanity visualization check whether we do have null values in ds or
df.isnull().sum()
```

Out[29]:
```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
Profit                       0
dtype: int64
```

In [30]:
```
plt.rcParams['figure.figsize']  = [15,5]
# plot the heat map:
sns.heatmap(df.isnull(),cbar = False)

# display the heatmap:
plt.show()
```



In [4]:
```
import seaborn as sns
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

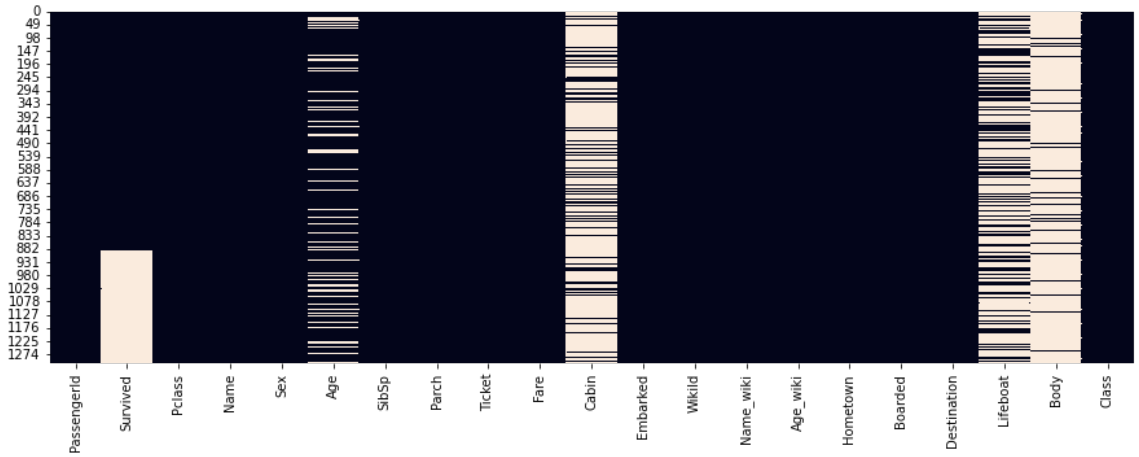
```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                     Name     Sex   A
ge  SibSp  \
0                                 Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                                  Heikkinen, Miss. Laina  female  2
6.0      0
3            Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                               Allen, Mr. William Henry    male  3
5.0      0
...                                                  ...     ...
...    ...
1304                                  Spector, Mr. Woolf    male   N
aN      0
1305                        Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                        Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                                 Ware, Mr. Frederick    male   N
aN      0
1308                             Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
1307      0              359309    8.0500  ...        S  1289.0
1308      1                2668   22.3583  ...        C   702.0

                                       Name_wiki Age_wiki  \
0                         Braund, Mr. Owen Harris     22.0
1     Cumings, Mrs. Florence Briggs (née Thayer)     35.0
2                           Heikkinen, Miss Laina     26.0
3              Futrelle, Mrs. Lily May (née Peel)     35.0
4                         Allen, Mr. William Henry     35.0
...                                            ...      ...
1304                          Spector, Mr. Woolf     23.0
1305        and maid, Doña Fermina Oliva y Ocana     39.0
```

```
1306                 Sæther, Mr. Simon Sivertsen     43.0
1307                Ware, Mr. Frederick William     34.0
1308            Butrus-Youssef, Master Makhkhul      4.0

                                    Hometown      Boarded  \
0             Bridgerule, Devon, England  Southampton
1                 New York, New York, US    Cherbourg
2                    Jyväskylä, Finland  Southampton
3           Scituate, Massachusetts, US  Southampton
4     Birmingham, West Midlands, England  Southampton
...                                  ...          ...
1304                  London, England  Southampton
1305                    Madrid, Spain    Cherbourg
1306        Skaun, Sør-Trøndelag, Norway  Southampton
1307          Greenwich, London, England  Southampton
1308                     Sar'al[81]  Syria    Cherbourg
```

In [5]:
```
# Drap the Rows which has Null Values
#1st Strategy
# To Eliminate the Null values from our Dataset

df_titanic_sales_drop = df_titanic.dropna()

# Display the shape after dropping null values from the data:
df_titanic_sales_drop.shape
```

Out[5]: (0, 21)

In [6]:
```
# visualize the null values by means of heatmap:
# 1. set the figure size:
plt.rcParams['figure.figsize']  = [15,5]
# plot the heat map:
sns.heatmap(df_titanic.isnull(),cbar = False)

# display the heatmap:
plt.show()
```



In [7]:
```
# Sanity check for Null Values
df_titanic.isnull().sum()
```

Out[7]:

```
PassengerId       0
Survived        418
Pclass            0
Name              0
Sex               0
Age             263
SibSp             0
Parch             0
Ticket            0
Fare              1
Cabin          1014
Embarked          2
WikiId            5
Name_wiki         5
```

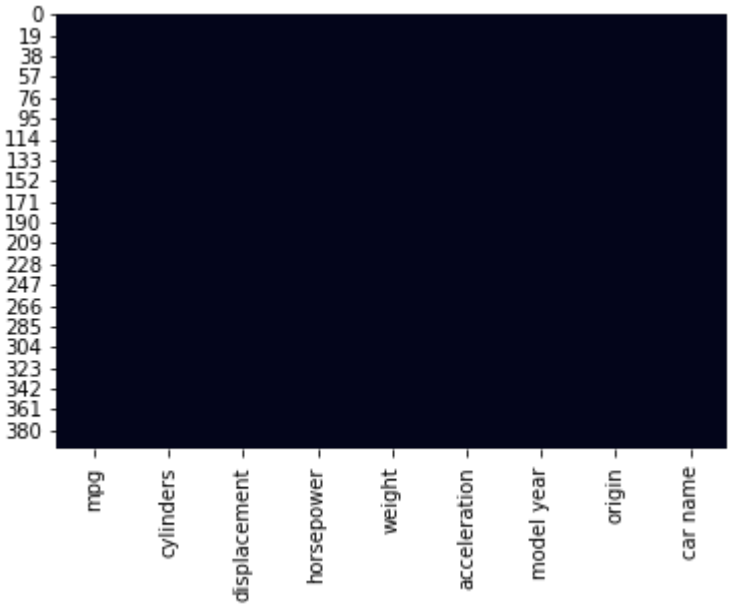
In [8]:
```
# To eliminate the null values
df_titanic_sales_drop = df_titanic.dropna(how='any',inplace = True)
```

In [9]:
```
df_titanic.isnull().sum()
```

Out[9]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
WikiId         0
Name_wiki      0
Age_wiki       0
Hometown       0
Boarded        0
Destination    0
Lifeboat       0
Body           0
Class          0
dtype: int64
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:
```
sns.heatmap(df.isnull(),cbar = False)
plt.show()
```



In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_excel('Items.xlsx',engine='openpyxl')
df
```

Out[3]:

	id	item	quantity	Prices	bought	forenoon	afternoon
0	1	milk	2.0	67.0	672.0	456.0	NaN
1	2	sugar	1.0	NaN	453.0	234.0	NaN
2	3	chips	NaN	45.0	456.0	322.0	NaN
3	4	coffee	2.0	45.0	672.0	564.0	NaN
4	5	meat	4.0	56.0	786.0	221.0	NaN
5	6	chocos	3.0	NaN	345.0	NaN	213.0
6	7	juice	1.0	78.0	765.0	NaN	344.0
7	8	jam	NaN	65.0	665.0	NaN	333.0
8	9	bread	3.0	NaN	NaN	NaN	567.0
9	10	butter	1.0	NaN	NaN	NaN	332.0

Filling Missing Values

1.filling with mean

2.filling with median

3.filling with standard deviation

4.filling with min values in our column

5.filling with max values in our column

In [4]:
```
df['quantity'].isnull().sum()
```

Out[4]: 2

In [5]:
```
# repalcing missing values to Quantity values
df['quantity'] = df['quantity'].fillna(df['quantity'].mean())
```

In [8]:
```
print(df['quantity'].mean())
print(df['quantity'].isnull().sum())
df
```

```
2.125
0
```

Out[8]:

	id	item	quantity	Prices	bought	forenoon	afternoon
0	1	milk	2.000	67.0	672.0	456.0	NaN
1	2	sugar	1.000	NaN	453.0	234.0	NaN
2	3	chips	2.125	45.0	456.0	322.0	NaN
3	4	coffee	2.000	45.0	672.0	564.0	NaN
4	5	meat	4.000	56.0	786.0	221.0	NaN
5	6	chocos	3.000	NaN	345.0	NaN	213.0
6	7	juice	1.000	78.0	765.0	NaN	344.0
7	8	jam	2.125	65.0	665.0	NaN	333.0
8	9	bread	3.000	NaN	NaN	NaN	567.0
9	10	butter	1.000	NaN	NaN	NaN	332.0

In [16]:
```
df['Prices'].isnull().sum()
```

Out[16]: 0

In [14]:
```
# repalcing missing values to Quantity values
print(df['Prices'].isnull().sum())
df['Prices'] = df['Prices'].fillna(df['Prices'].median())
print(df['Prices'].median())
print(df['Prices'].isnull().sum())
df
```

```
0
60.5
0
```

Out[14]:

	id	item	quantity	Prices	bought	forenoon	afternoon
0	1	milk	2.000	67.0	672.0	456.0	NaN
1	2	sugar	1.000	60.5	453.0	234.0	NaN
2	3	chips	2.125	45.0	456.0	322.0	NaN
3	4	coffee	2.000	45.0	672.0	564.0	NaN
4	5	meat	4.000	56.0	786.0	221.0	NaN
5	6	chocos	3.000	60.5	345.0	NaN	213.0
6	7	juice	1.000	78.0	765.0	NaN	344.0
7	8	jam	2.125	65.0	665.0	NaN	333.0

	id	item	quantity	Prices	bought	forenoon	afternoon
8	9	bread	3.000	60.5	NaN	NaN	567.0

In [21]:
```
df['bought'].isnull().sum()
```

Out[21]: 0

In [18]:
```
df['bought'] = df['bought'].fillna(df['bought'].std())
df
```

Out[18]:

	id	item	quantity	Prices	bought	forenoon	afternoon
0	1	milk	2.000	67.0	672.000000	456.0	NaN
1	2	sugar	1.000	60.5	453.000000	234.0	NaN
2	3	chips	2.125	45.0	456.000000	322.0	NaN
3	4	coffee	2.000	45.0	672.000000	564.0	NaN
4	5	meat	4.000	56.0	786.000000	221.0	NaN
5	6	chocos	3.000	60.5	345.000000	NaN	213.0
6	7	juice	1.000	78.0	765.000000	NaN	344.0
7	8	jam	2.125	65.0	665.000000	NaN	333.0
8	9	bread	3.000	60.5	162.022706	NaN	567.0
9	10	butter	1.000	60.5	162.022706	NaN	332.0

In [22]:
```
df['forenoon'].isnull().sum()
```

Out[22]: 5

In [23]:
```
df['forenoon'] = df['forenoon'].fillna(df['forenoon'].max())
df
```

Out[23]:

	id	item	quantity	Prices	bought	forenoon	afternoon
0	1	milk	2.000	67.0	672.000000	456.0	NaN
1	2	sugar	1.000	60.5	453.000000	234.0	NaN
2	3	chips	2.125	45.0	456.000000	322.0	NaN
3	4	coffee	2.000	45.0	672.000000	564.0	NaN
4	5	meat	4.000	56.0	786.000000	221.0	NaN
5	6	chocos	3.000	60.5	345.000000	564.0	213.0
6	7	juice	1.000	78.0	765.000000	564.0	344.0
7	8	jam	2.125	65.0	665.000000	564.0	333.0
8	9	bread	3.000	60.5	162.022706	564.0	567.0
9	10	butter	1.000	60.5	162.022706	564.0	332.0

In [24]:
```
df['afternoon'].isnull().sum()
```

Out[24]: 5

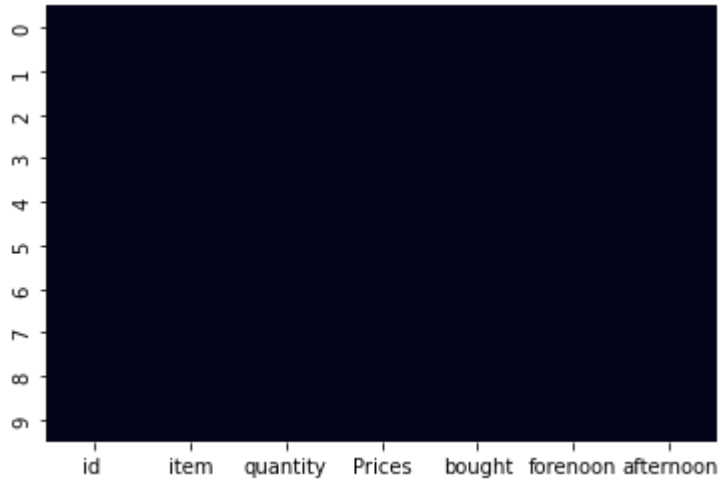
In [25]:
```
df['afternoon'] = df['afternoon'].fillna(df['afternoon'].min())
df
```

Out[25]:

	id	item	quantity	Prices	bought	forenoon	afternoon

	id	item	quantity	Prices	bought	forenoon	afternoon
0	1	milk	2.000	67.0	672.000000	456.0	213.0
1	2	sugar	1.000	60.5	453.000000	234.0	213.0
2	3	chips	2.125	45.0	456.000000	322.0	213.0
3	4	coffee	2.000	45.0	672.000000	564.0	213.0
4	5	meat	4.000	56.0	786.000000	221.0	213.0
5	6	chocos	3.000	60.5	345.000000	564.0	213.0
6	7	juice	1.000	78.0	765.000000	564.0	344.0
7	8	jam	2.125	65.0	665.000000	564.0	333.0
8	9	bread	3.000	60.5	162.022706	564.0	567.0

In [26]:
```
sns.heatmap(df.isnull(),cbar = False)
plt.show()
```



In [28]:
```
df = pd.read_csv('auto-mpg.csv')
df
```

Out[28]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nam
0	18.0	8	307.0	130	3504	12.0	70	1	chevrole chevell malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buic skylar 32
2	18.0	8	318.0	150	3436	11.0	70	1	plymoutl satellit
3	16.0	8	304.0	150	3433	12.0	70	1	am rebel ss
4	17.0	8	302.0	140	3449	10.5	70	1	for torin
...	...	...	...	...	...	...	...	...	..
393	27.0	4	140.0	86	2790	15.6	82	1	for mustan g
394	44.0	4	97.0	52	2130	24.6	82	2	vw picku

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nam
395	32.0	4	135.0	84	2295	11.6	82	1	dodg rampag
396	28.0	4	120.0	79	2625	18.6	82	1	for range

In [ ]:
```
# Data Transformation
skew
- output is positive number then positive skew
- output is negative number then negative skew
```

In [30]:
```
import seaborn as sns
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                     Name     Sex   A
ge  SibSp  \
0                                 Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                                  Heikkinen, Miss. Laina  female  2
6.0      0
3            Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                               Allen, Mr. William Henry    male  3
5.0      0
...                                                   ...     ...
...    ...
1304                                  Spector, Mr. Woolf    male   N
aN      0
1305                        Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                        Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                                 Ware, Mr. Frederick    male   N
aN      0
1308                             Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
```

In [69]:
```
df_titanic['Age'] = df_titanic['Age'].fillna(df_titanic['Age'].mean()
print(df_titanic['Age'].mean())
df_titanic['Fare'] = df_titanic['Fare'].fillna(df_titanic['Fare'].mir
print(df_titanic['Fare'].min())
df_titanic['Survived'] = df_titanic['Survived'].fillna(df_titanic['Su
print(df_titanic['Survived'].max())
df_titanic['Age_wiki'] = df_titanic['Age_wiki'].fillna(df_titanic['Ag
print(df_titanic['Age_wiki'].median())
df_titanic
```

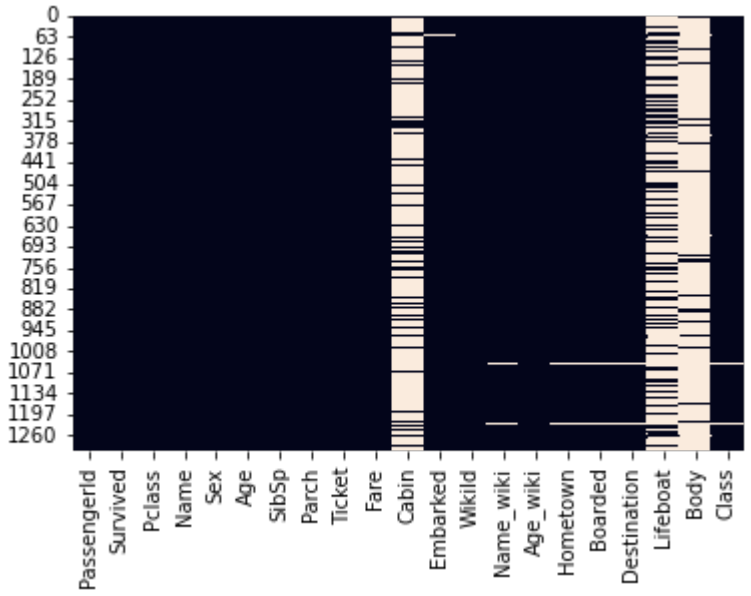
```
29.881137667304014
0 0
```

Out[69]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 211
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 175
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O 31012
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0.0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	
1304	1305	1.0	3	Spector, Mr. Woolf	male	29.881138	0	0	A.5. 32
1305	1306	1.0	1	Oliva y Ocana, Dona. Fermina	female	39.000000	0	0	PC 177
1306	1307	1.0	3	Saether, Mr. Simon Sivertsen	male	38.500000	0	0	SOTON/O. 31012
1307	1308	1.0	3	Ware, Mr. Frederick	male	29.881138	0	0	3593
1308	1309	1.0	3	Peter, Master. Michael J	male	29.881138	1	1	26

1309 rows × 21 columns

In [52]:
```
sns.heatmap(df_titanic.isnull(),cbar = False)
plt.show()
```



Categorical Data

In [56]:
```
# Handling Missing values in Categorical data
df_titanic[['Cabin','Embarked','Lifeboat','Body']].head()
```

Out[56]:

	Cabin	Embarked	Lifeboat	Body
0	NaN	S	NaN	NaN
1	C85	C	4	NaN
2	NaN	S	14?	NaN
3	C123	S	D	NaN
4	NaN	S	NaN	NaN

In [60]:
```
df_titanic.isnull().sum()
```

Out[60]:

```
PassengerId       0
Survived          0
```

In [66]:
```
# Replacing null values with Unknown Class
df_titanic['Cabin'] = df_titanic['Cabin'].fillna('Unknown')
df_titanic['Name_wiki'] = df_titanic['Name_wiki'].fillna('Unknown')
df_titanic['Hometown'] = df_titanic['Hometown'].fillna('Unknown')
df_titanic['Boarded'] = df_titanic['Boarded'].fillna('Unknown')
df_titanic['Destination'] = df_titanic['Destination'].fillna('Unknown
# Replacing the null values with the most frequent value
df_titanic['Embarked'] = df_titanic['Embarked'].fillna(df_titanic['Em
df_titanic['Lifeboat'] = df_titanic['Lifeboat'].fillna(df_titanic['Li
df_titanic['Body'] = df_titanic['Body'].fillna(df_titanic['Body'].val
df_titanic['Class'] = df_titanic['Class'].fillna(df_titanic['Class'].
df_titanic
```

Out[66]:

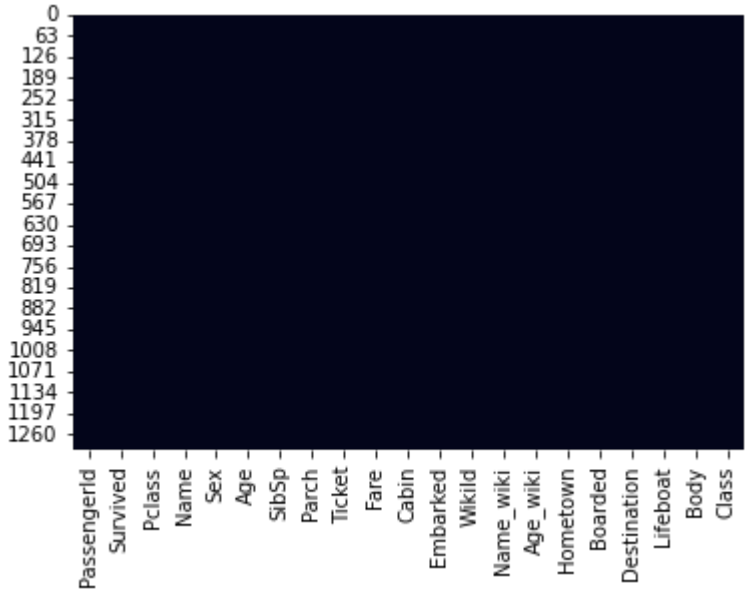
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 211
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 175
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O 31012
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	1138
4	5	0.0	3	Allen, Mr. William Henry	male	35.000000	0	0	3734
...	...	...	...	...	...	...	...	...	
1304	1305	1.0	3	Spector, Mr. Woolf	male	29.881138	0	0	A.5. 32
1305	1306	1.0	1	Oliva y Ocana, Dona. Fermina	female	39.000000	0	0	PC 177
1306	1307	1.0	3	Saether, Mr. Simon Sivertsen	male	38.500000	0	0	SOTON/O. 31012
1307	1308	1.0	3	Ware, Mr. Frederick	male	29.881138	0	0	3593
1308	1309	1.0	3	Peter, Master. Michael J	male	29.881138	1	1	26

1309 rows × 21 columns

In [67]:
```
df_titanic.isnull().sum()
```

Out[67]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
WikiId         0
Name_wiki      0
Age_wiki       0
Hometown       0
Boarded        0
Destination    0
Lifeboat       0
Body           0
Class          0
dtype: int64
```

In [71]:
```
sns.heatmap(df_titanic.isnull(),cbar = False)
plt.show()
```



In [72]:
```
## Another Code
df_titanic = pd.read_csv('titanic Manual Excel.csv')
print(df_titanic)
```

```
      PassengerId  Survived  Pclass  \
0               1       0.0       3
1               2       1.0       1
2               3       1.0       3
3               4       1.0       1
4               5       0.0       3
...           ...       ...     ...
1304         1305       NaN       3
1305         1306       NaN       1
1306         1307       NaN       3
1307         1308       NaN       3
1308         1309       NaN       3

                                                     Name     Sex   A
ge  SibSp  \
0                                 Braund, Mr. Owen Harris    male  2
2.0      1
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  3
8.0      1
2                                  Heikkinen, Miss. Laina  female  2
6.0      0
3            Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  3
5.0      1
4                               Allen, Mr. William Henry    male  3
5.0      0
...                                                  ...     ...
...    ...
1304                                  Spector, Mr. Woolf    male   N
aN      0
1305                        Oliva y Ocana, Dona. Fermina  female  3
9.0      0
1306                        Saether, Mr. Simon Sivertsen    male  3
8.5      0
1307                                 Ware, Mr. Frederick    male   N
aN      0
1308                             Peter, Master. Michael J    male   N
aN      1

      Parch              Ticket      Fare  ... Embarked  WikiId  \
0         0           A/5 21171    7.2500  ...        S   691.0
1         0            PC 17599   71.2833  ...        C    90.0
2         0    STON/O2. 3101282    7.9250  ...        S   865.0
3         0              113803   53.1000  ...        S   127.0
4         0              373450    8.0500  ...        S   627.0
...     ...                 ...       ...  ...      ...     ...
1304      0           A.5. 3236    8.0500  ...        S  1227.0
1305      0            PC 17758  108.9000  ...        C   229.0
1306      0  SOTON/O.Q. 3101262    7.2500  ...        S  1169.0
1307      0              359309    8.0500  ...        S  1289.0
1308      1                2668   22.3583  ...        C   702.0

                                       Name_wiki Age_wiki  \
0                         Braund, Mr. Owen Harris     22.0
1     Cumings, Mrs. Florence Briggs (née Thayer)     35.0
2                           Heikkinen, Miss Laina     26.0
3              Futrelle, Mrs. Lily May (née Peel)     35.0
4                         Allen, Mr. William Henry     35.0
...                                            ...      ...
1304                         Spector, Mr. Woolf     23.0
1305        and maid, Doña Fermina Oliva y Ocana     39.0
```

```
1306                 Sæther, Mr. Simon Sivertsen     43.0
1307                Ware, Mr. Frederick William     34.0
1308            Butrus-Youssef, Master Makhkhul      4.0

                                  Hometown      Boarded  \
0             Bridgerule, Devon, England  Southampton
1                 New York, New York, US    Cherbourg
2                     Jyväskylä, Finland  Southampton
3           Scituate, Massachusetts, US  Southampton
4     Birmingham, West Midlands, England  Southampton
...                                  ...          ...
1304                  London, England  Southampton
1305                    Madrid, Spain    Cherbourg
1306        Skaun, Sør-Trøndelag, Norway  Southampton
1307          Greenwich, London, England  Southampton
1308                   Sar'al[81], Syria    Cherbourg

                                Destination Lifeboat  Body Class
0     Qu'Appelle Valley, Saskatchewan, Canada      NaN   NaN   3.0
1                     New York, New York, US        4   NaN   1.0
2                               New York City      14?   NaN   3.0
```

In [75]:
```
# To eliminate the null values
df_titanic_sales_drop = df_titanic.dropna(how='any',inplace = True)
```

In [77]:
```
df_titanic.isnull().sum()
```

Out[77]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
WikiId         0
Name_wiki      0
Age_wiki       0
Hometown       0
Boarded        0
Destination    0
Lifeboat       0
Body           0
Class          0
dtype: int64
```

In [78]:
```
sns.heatmap(df_titanic.isnull(),cbar = False)
plt.show()
```

```
---------------------------------------------------------------------
--------
ValueError                                Traceback (most recent ca
ll last)
<ipython-input-78-ddf11c599519> in <module>
----> 1 sns.heatmap(df_titanic.isnull(),cbar = False)
      2 plt.show()

~/my_project_env/lib/python3.6/site-packages/seaborn/_decorators.py
in inner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.paramet
ers, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

~/my_project_env/lib/python3.6/site-packages/seaborn/matrix.py in h
eatmap(data, vmin, vmax, cmap, center, robust, annot, fmt, annot_kw
s, linewidths, linecolor, cbar, cbar_kws, cbar_ax, square, xticklab
els, yticklabels, mask, ax, **kwargs)
    540     plotter = _HeatMapper(data, vmin, vmax, cmap, center, r
obust, annot, fmt,
    541                           annot_kws, cbar, cbar_kws, xtickl
abels,
--> 542                           yticklabels, mask)
    543
    544     # Add the pcolormesh kwargs here

~/my_project_env/lib/python3.6/site-packages/seaborn/matrix.py in _
_init__(self, data, vmin, vmax, cmap, center, robust, annot, fmt, a
nnot_kws, cbar, cbar_kws, xticklabels, yticklabels, mask)
    158         # Determine good default values for the colormappin
g
    159         self._determine_cmap_params(plot_data, vmin, vmax,
--> 160                                     cmap, center, robust)
    161
    162         # Sort out the annotations

~/my_project_env/lib/python3.6/site-packages/seaborn/matrix.py in _
determine_cmap_params(self, plot_data, vmin, vmax, cmap, center, ro
bust)
    196                 vmin = np.nanpercentile(calc_data, 2)
    197             else:
--> 198                 vmin = np.nanmin(calc_data)
    199         if vmax is None:
    200             if robust:

<__array_function__ internals> in nanmin(*args, **kwargs)

~/my_project_env/lib/python3.6/site-packages/numpy/lib/nanfunction
s.py in nanmin(a, axis, out, keepdims)
    317         # Fast, but not safe for subclasses of ndarray, or
object arrays,
    318         # which do not implement isnan (gh-9009), or fmin c
orrectly (gh-8975)
--> 319         res = np.fmin.reduce(a, axis=axis, out=out, **kwarg
```

In [1]:
```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
## Another Code
df = pd.read_csv('k_circle_sales.csv')
df.head()
```

Out[1]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outle
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.3	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.1	
4	NCD19	8.93	Low Fat	0.000000	Household	53.9	

In [2]:
```
#Filling Out Numerical Columns from the Dataset

df_num = df.select_dtypes(include = [np.number])
df_num
```

Out[2]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales	I
0	9.300	0.016047	249.8	1999	3735.1380	
1	5.920	0.019278	48.3	2009	443.4228	
2	17.500	0.016760	141.6	1999	2097.2700	
3	19.200	0.000000	182.1	1998	732.3800	
4	8.930	0.000000	53.9	1987	994.7052	
...	...	...	...	...	...	
8518	6.865	0.056783	214.5	1987	2778.3834	
8519	8.380	0.046982	108.2	2002	549.2850	
8520	10.600	0.035186	85.1	2004	1193.1136	
8521	7.210	0.145221	103.1	2009	1845.5976	
8522	14.800	0.044878	75.5	1997	765.6700	

8523 rows × 6 columns

In [8]:
```
# Filling Out only Categorical column from your Dataset:
from warnings import filterwarnings
filterwarnings('ignore')
df_cat = df.select_dtypes(include = [np.object])
df_cat
```

Out[8]:

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Locat
0	FDA15	Low Fat	Dairy	OUT049	Medium	
1	DRC01	Regular	Soft Drinks	OUT018	Medium	
2	FDN15	Low Fat	Meat	OUT049	Medium	

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Locat
3	FDX07	Regular	Fruits and Vegetables	OUT010	NaN	
4	NCD19	Low Fat	Household	OUT013	High	
...	...	...	...	...	...	
8518	FDF22	Low Fat	Snack Foods	OUT013	High	
8519	FDS36	Regular	Baking Goods	OUT045	NaN	
8520	NCJ29	Low Fat	Health and Hygiene	OUT035	Small	
8521	FDN46	Regular	Snack Foods	OUT018	Medium	
8522	DRG01	Low Fat	Soft Drinks	OUT046	Small	

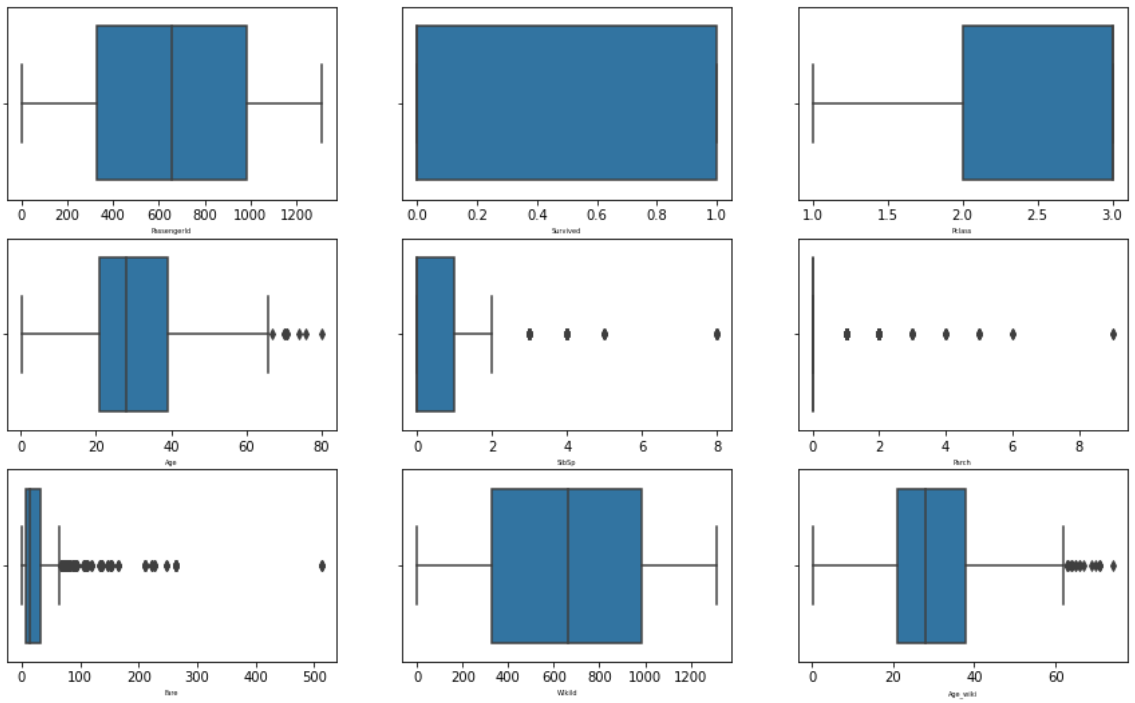
In [6]:
```
df_num_columns
```

Out[6]:
```
Index(['Item_Identifier', 'Item_Fat_Content', 'Item_Type', 'Outlet_
Identifier',
       'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type'],
      dtype='object')
```

In [9]:
```
df_cat_columns
```

Out[9]:
```
Index(['Item_Identifier', 'Item_Fat_Content', 'Item_Type', 'Outlet_
Identifier',
       'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type'],
      dtype='object')
```

In [20]:
```
# To identify the Outliers in Numerical Columns:
# Subplots()
fig, ax = plt.subplots(3,3,figsize =(15,9))


for variable,subplot in zip(df_num.columns,ax.flatten()):
    z = sns.boxplot(x = df_num[variable], orient = 'h',whis = 1.5,ax

    z.set_xlabel(variable,fontsize = 5)
```



In [15]:
```
df= pd.read_csv('titanic Manual Excel.csv')
df.head()
```

Out[15]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

5 rows × 21 columns

In [16]:
```
# filtering out numerical columns from the data set:
df_num = df.select_dtypes(include = 'number')
df_num
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	WikiId	Age_wiki	Class
0	1	0.0	3	22.0	1	0	7.2500	691.0	22.0	3.0
1	2	1.0	1	38.0	1	0	71.2833	90.0	35.0	1.0
2	3	1.0	3	26.0	0	0	7.9250	865.0	26.0	3.0
3	4	1.0	1	35.0	1	0	53.1000	127.0	35.0	1.0
4	5	0.0	3	35.0	0	0	8.0500	627.0	35.0	3.0
...	...	...	...	...	...	...	...	...	...	...
1304	1305	NaN	3	NaN	0	0	8.0500	1227.0	23.0	3.0
1305	1306	NaN	1	39.0	0	0	108.9000	229.0	39.0	1.0
1306	1307	NaN	3	38.5	0	0	7.2500	1169.0	43.0	3.0
1307	1308	NaN	3	NaN	0	0	8.0500	1289.0	34.0	3.0
1308	1309	NaN	3	NaN	1	1	22.3583	702.0	4.0	3.0

1309 rows × 10 columns

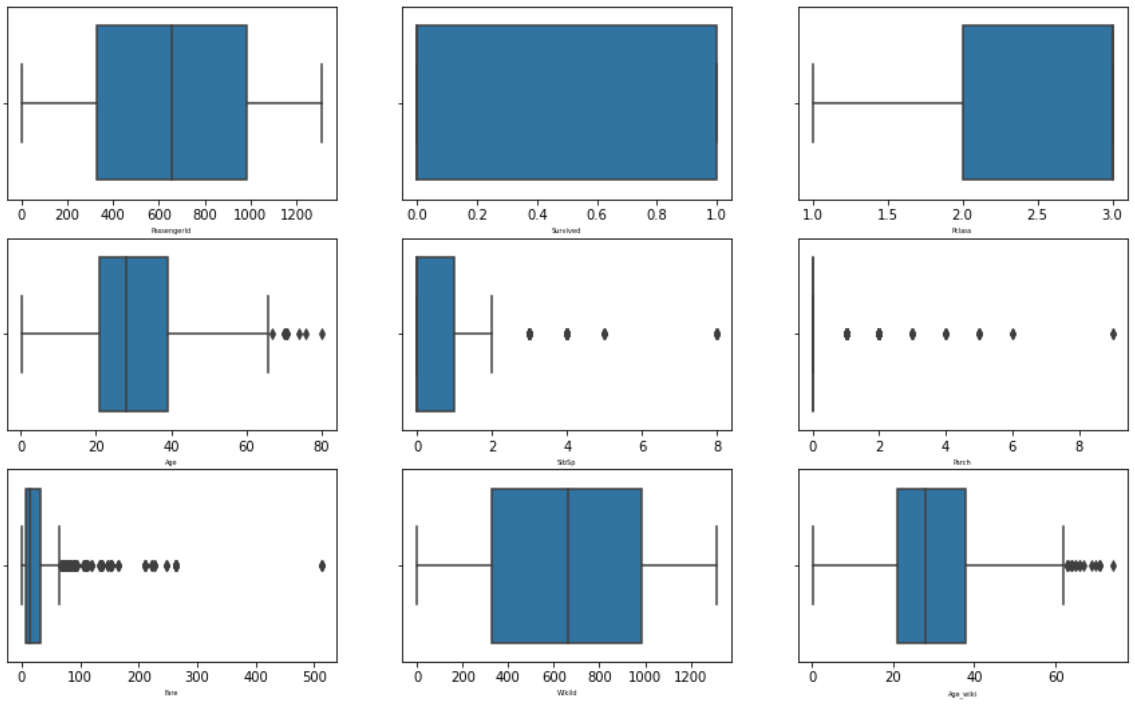
In [17]:
```
df_num.columns
```

Out[17]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch
', 'Fare',
       'WikiId', 'Age_wiki', 'Class'],
      dtype='object')
```

In [18]:
```
df_cat.columns
```

Out[18]:
```
Index(['Item_Identifier', 'Item_Fat_Content', 'Item_Type', 'Outlet_
Identifier',
       'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type'],
      dtype='object')
```

In [19]:
```
# To identify the outliers in numerical Columns:

# SUBPLOTS()
fig, ax = plt.subplots(3,3,figsize =(15,9))


for variable,subplot in zip(df_num.columns,ax.flatten()):
    z = sns.boxplot( x = df_num[variable], orient = 'h',whis = 1.5,ax

    z.set_xlabel(variable,fontsize = 5)
```



In [21]:
```
# 1. based on IQR method:
Q1 = df_num.quantile(0.25)


Q3 = df_num.quantile(0.75)

# obtain the type;
IQR =Q3-Q1
IQR
```

Out[21]:
```
PassengerId    654.0000
Survived         1.0000
Pclass           1.0000
Age             18.0000
SibSp            1.0000
Parch            0.0000
Fare            23.3792
WikiId         660.5000
Age_wiki        16.7500
Class            1.2500
dtype: float64
```

In [22]:
```
df_iqr = df[~((df_num<(Q1 - 1.5* IQR))|(df_num>(Q3+1.5*IQR))).any(axi

df_iqr.shape
```

Out[22]:
```
(883, 21)
```

In [23]:
```
df.shape
```

Out[23]:
```
(1309, 21)
```

In [ ]:
```
## Data Transformation
- To reduce the skewness in the distribution of the original data
- It makes the Data more Interpretable
- The Arithmetic mean of the log-Transformed is the Geometric mean of
- log transformation
- exponential transformation
- boxkak transformation-dataset having 10 00 000 lakhs above rows
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_csv("auto-mpg.csv")
df
```

Out[3]:

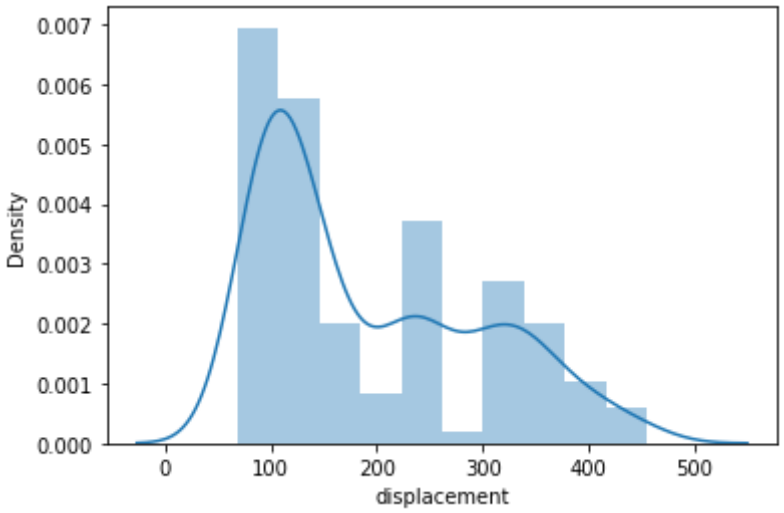
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nam
0	18.0	8	307.0	130	3504	12.0	70	1	chevrole chevell malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buic skylar 32
2	18.0	8	318.0	150	3436	11.0	70	1	plymoutl satellit
3	16.0	8	304.0	150	3433	12.0	70	1	am rebel ss
4	17.0	8	302.0	140	3449	10.5	70	1	for torin
...	...	...	...	...	...	...	...	...	..
393	27.0	4	140.0	86	2790	15.6	82	1	for mustan g
394	44.0	4	97.0	52	2130	24.6	82	2	vw picku
395	32.0	4	135.0	84	2295	11.6	82	1	dodg rampag
396	28.0	4	120.0	79	2625	18.6	82	1	for range
397	31.0	4	119.0	82	2720	19.4	82	1	chev s-1

398 rows × 9 columns

In [5]:
```
# Distribution of the Dispalcement column
sns.distplot(df['displacement'])
plt.ylabel('Density')
print('Skewness : ',df['displacement'].skew())
plt.show()
```

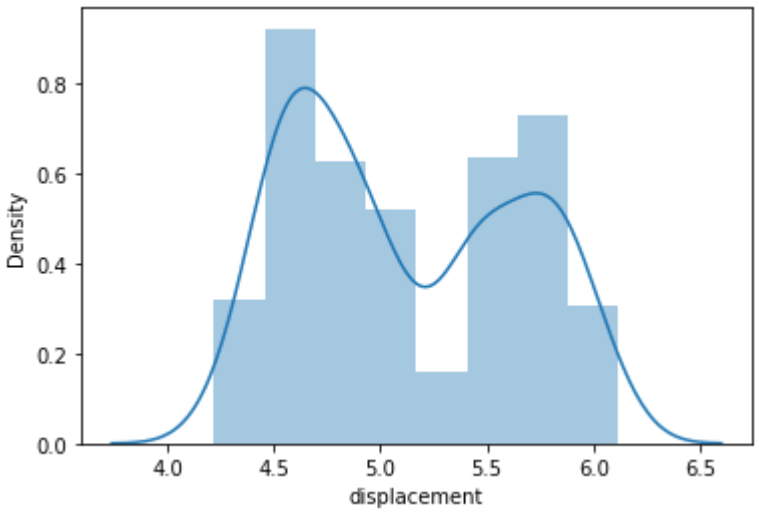
```
/home/student/my_project_env/lib/python3.6/site-packages/seaborn/di
stributions.py:2619: FutureWarning: `distplot` is a deprecated func
tion and will be removed in a future version. Please adapt your cod
e to use either `displot` (a figure-level function with similar fle
```

```
Skewness :  0.7196451643005952
```



In [10]:
```
# Apply natural log transformation for Displacement col:
log_displacement = np.log(df['displacement'])
print('Skewness after log Transformtion : ',log_displacement.skew())

sns.distplot(log_displacement)
plt.ylabel('Density')
plt.show()
```

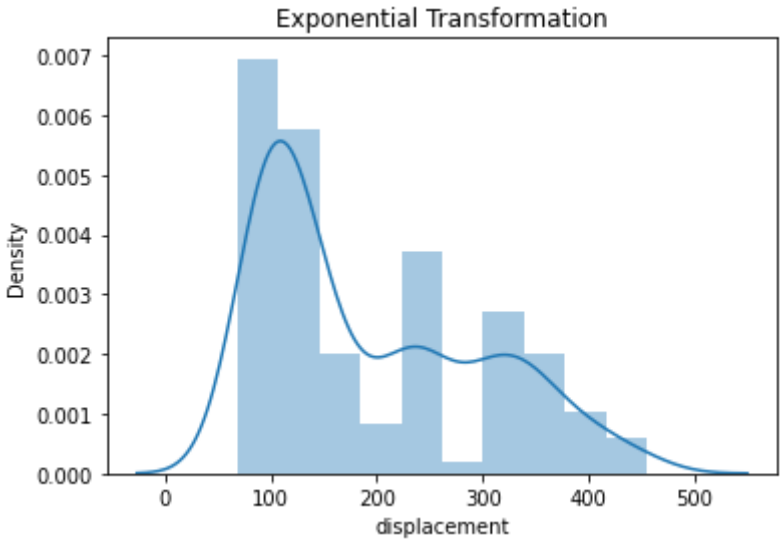
```
Skewness after log Transformtion :  0.22600298495225188

/home/student/my_project_env/lib/python3.6/site-packages/seaborn/di
stributions.py:2619: FutureWarning: `distplot` is a deprecated func
tion and will be removed in a future version. Please adapt your cod
e to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [9]:
```
# Anti - log or Exponential Transformation
displacement = np.exp(log_displacement)
# plot the Distribution
sns.distplot(displacement)
```

```
plt.ylabel('Density')
plt.title('Exponential Transformation')
plt.show()
```

```
/home/student/my_project_env/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [ ]: