Pandas

What is Pandas?

Pandas is a data analysis & manipulation library.

Series - A Series is a one-dimensional labeled array, kind of like a single column in Excel.

DataFrame - A DataFrame is a two-dimensional table — like an Excel spreadsheet with rows and columns. Multiple Series make one DataFrame.

In [1]:
```
import numpy as np
import pandas as pd
l = [10,20,30,40,50]
for i in enumerate(l):
    print(i)

print()

d = {'a':1, 'b':2, 'c':3}
for i in enumerate(d):
    print(i)
```

```
(0, 10)
(1, 20)
(2, 30)
(3, 40)
(4, 50)

(0, 'a')
(1, 'b')
(2, 'c')
```

Series

In [7]:
```
print(pd.Series(l))
print()
#using list
l1 = ['hi', 45, 56.7, "hello", 0]
print(pd.Series(l1, index = ['h', 0, 1, 'he', '10']))

print()

#using tuple
t1 = ('hi', 45, 56.7, "hello", 0)
print(pd.Series(t1, index = ['h', 0, 1, 'he', '10']))
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64

h        hi
0        45
1      56.7
he    hello
10        0
dtype: object

h        hi
0        45
1      56.7
he    hello
10        0
dtype: object
```

In [9]:
```
s1 = pd.Series([3,4,5], index = ['a','b','c'])
s2 = pd.Series([6,7,8], index = ['a','b','c'])
s3 = pd.Series([6,0,1], index = ['a','b','d'])
print(s1+s2)
print()
print(s1+s3)
```

```
a     9
b    11
c    13
dtype: int64

a    9.0
b    4.0
c    NaN
d    NaN
dtype: float64
```

In [11]:
```
#Position based indexing
print(s1[1])
#Label based indexing
print(s1['c'])
```

```
4
5
```

In [17]:
```
lis = ['hi', 45, 56.7, "hello", 0]
s = pd.Series(lis, index = ['h', 0, 1, 'he', '10'])
print(s,"\n")
print(s[0:4], "\n") # last index is excluded in Position based indexing
print(s['h':'10'], "\n") # last index is included in Label based indexing
print(s[['h', 1, 'he']]) # Acessing random labels instead of a sequence
```

```
h        hi
0        45
1      56.7
he    hello
10        0
dtype: object

h        hi
0        45
1      56.7
he    hello
dtype: object

h        hi
0        45
1      56.7
he    hello
10        0
dtype: object

h        hi
1      56.7
he    hello
dtype: object
```

In [20]:
```
veg = pd.Series({"tomato":70, "potato": 87, "carrot":67, "brinjal":45, "cucumber":56, "chillies":32})
print(veg, "\n")
print(veg.values)
print(veg.index)
```

```
tomato      70
potato      87
carrot      67
brinjal     45
cucumber    56
chillies    32
dtype: int64

[70 87 67 45 56 32]
Index(['tomato', 'potato', 'carrot', 'brinjal', 'cucumber', 'chillies'], dtype='object')
```

In [28]:
```
print(veg[0], "\n")
print(veg["potato"], "\n")
print(veg["tomato":"brinjal"], "\n")
print(veg[0:3], "\n")
print(veg[["tomato","cucumber"]], "\n")
```

```
70

87

tomato     70
potato     87
carrot     67
brinjal    45
dtype: int64

tomato    70
potato    87
carrot    67
dtype: int64

tomato      70
cucumber    56
dtype: int64
```

In [29]:
```
print(veg[:"brinjal"], "\n")
print(veg["brinjal":], "\n")
print(veg[:], "\n")
print(veg[:-1], "\n")
```

```
tomato     70
potato     87
carrot     67
brinjal    45
dtype: int64

brinjal     45
cucumber    56
chillies    32
dtype: int64

tomato      70
potato      87
carrot      67
brinjal     45
cucumber    56
chillies    32
dtype: int64

tomato      70
potato      87
carrot      67
brinjal     45
cucumber    56
dtype: int64
```

In [30]:
```
print(veg[1::2]) # 2 step accessing
```

```
potato      87
brinjal     45
chillies    32
dtype: int64
```

In [32]:
```
veg.loc["brinjal":"chillies"]#Label based accessing
```

Out[32]:
```
brinjal     45
cucumber    56
chillies    32
dtype: int64
```

In [34]:
```
veg.iloc[0:3]#index based accessing
```

Out[34]:
```
tomato    70
potato    87
carrot    67
dtype: int64
```

In [35]:
```
veg > 60 #comparing and resulting boolean value
```

Out[35]:
```
tomato       True
potato       True
carrot       True
brinjal     False
cucumber    False
chillies    False
dtype: bool
```

In [36]:
```
veg[veg > 60] # Filtering values based on some condition
```

Out[36]:
```
tomato    70
potato    87
carrot    67
dtype: int64
```

In [37]:
```
veg.head() # results top 5 values
```

```
veg.head() # results top 5 values
```

Out[37]:
```
tomato      70
potato      87
carrot      67
brinjal     45
cucumber    56
dtype: int64
```

In [38]:
```
veg.tail() # results last 5 values
```

Out[38]:
```
potato      87
carrot      67
brinjal     45
cucumber    56
chillies    32
dtype: int64
```

In [39]:
```
veg.head(2)
```

Out[39]:
```
tomato    70
potato    87
dtype: int64
```

In [40]:
```
veg.tail(1)
```

Out[40]:
```
chillies    32
dtype: int64
```

In [47]:
```
veg[veg.isin([2,4])]
```

Out[47]:
```
Series([], dtype: int64)
```

In [48]:
```
list('abcde')
```

Out[48]:
```
['a', 'b', 'c', 'd', 'e']
```

In [50]:
```
new = pd.Series(list('abcd'*4))
new
```

Out[50]:
```
0     a
1     b
2     c
3     d
4     a
5     b
6     c
7     d
8     a
9     b
10    c
11    d
12    a
13    b
14    c
15    d
dtype: object
```

In [53]:
```
# unique() : name of all unique records
# nunique() : count of unique records
# value_counts() : name of unique records with their occurrences
print(new.unique())
print(new.nunique())
print(new.value_counts())
```

```
['a' 'b' 'c' 'd']
4
a    4
b    4
c    4
d    4
dtype: int64
```

In [54]:
```
# Dealing with duplicate values
new[new.duplicated()]
```

Out[54]:
```
4     a
5     b
6     c
7     d
8     a
9     b
10    c
11    d
12    a
13    b
14    c
15    d
dtype: object
```

In [56]:
```
new.drop_duplicates()
```

In [56]:
```
new.drop_duplicates()
```

Out[56]:
```
0    a
1    b
2    c
3    d
dtype: object
```

In [58]:
```
# Sorting
new.sort_values() # sorting based on values
```

Out[58]:
```
0     a
4     a
8     a
12    a
1     b
5     b
9     b
13    b
2     c
6     c
10    c
14    c
3     d
7     d
11    d
15    d
dtype: object
```

In [59]:
```
new.sort_index() # sorting based on index
```

Out[59]:
```
0     a
1     b
2     c
3     d
4     a
5     b
6     c
7     d
8     a
9     b
10    c
11    d
12    a
13    b
14    c
15    d
dtype: object
```

In [61]:
```
# Dealing with Missing records
x = pd.Series([1,3,4,5,np.nan,5,np.nan,np.nan])
x
```

Out[61]:
```
0    1.0
1    3.0
2    4.0
3    5.0
4    NaN
5    5.0
6    NaN
7    NaN
dtype: float64
```

In [62]:
```
x.isnull()
```

Out[62]:
```
0    False
1    False
2    False
3    False
4     True
5    False
6     True
7     True
dtype: bool
```

In [63]:
```
x.isnull().sum() # gives count of missing records in x
```

Out[63]:
```
3
```

In [65]:
```
x[x.notnull()] # only accessing non null values
```

Out[65]:
```
0    1.0
1    3.0
2    4.0
3    5.0
5    5.0
dtype: float64
```

In [67]:
```
x.fillna(1000) #filling missing records
```

Out[67]:
```
0       1.0
1       3.0
2       4.0
3       5.0
4    1000.0
5       5.0
6    1000.0
7    1000.0
dtype: float64
```

In [119…
```
x.fillna(x.mean())
```

Out[119]:
```
0    1.0
1    3.0
2    4.0
3    5.0
4    3.6
5    5.0
6    3.6
7    3.6
dtype: float64
```

In [120…
```
x.ffill() #forward fill
```

Out[120]:
```
0    1.0
1    3.0
2    4.0
3    5.0
4    5.0
5    5.0
6    5.0
7    5.0
dtype: float64
```

In [121…
```
x.bfill() #backward fill
```

Out[121]:
```
0    1.0
1    3.0
2    4.0
3    5.0
4    5.0
5    5.0
6    NaN
7    NaN
dtype: float64
```

In [122…
```
x.dropna() #dropping null values
```

Out[122]:
```
0    1.0
1    3.0
2    4.0
3    5.0
5    5.0
dtype: float64
```

In [123…
```
ch = pd.Series(["jasmin", "lav", "zaru", "ammu"])
ch.map(lambda x:len(x))
```

Out[123]:
```
0    6
1    3
2    4
3    4
dtype: int64
```

DataFrame

In [124…
```
dt = pd.DataFrame()
dt # Empty dataframe
```

Out[124]: —

In [125…
```
lis = [30,50,60]
dt = pd.DataFrame(lis)
dt
```

Out[125]:

	0
0	30
1	50
2	60

In [126…
```
lis = [["jasmin",23], ["zaru",24], ["lav",20], ["devi",25]] # using list
df = pd.DataFrame(lis, columns = ["name", "age"])
df
```

Out[126]:

	name	age
0	jasmin	23
1	zaru	24
2	lav	20
3	devi	25

In [127…
```
dic = {"name":["jasmin", "zaru", "lav", "devi","chandu","deepu","susmi"], "age":[23,20,45,12,89,67,90]} # using
df1 = pd.DataFrame(dic, index = ["rank1", "rank2", "rank3", "rank4","rank5","rank6","rank7"])
df1
```

Out[127]:

	name	age
rank1	jasmin	23
rank2	zaru	20
rank3	lav	45
rank4	devi	12
rank5	chandu	89
rank6	deepu	67
rank7	susmi	90

In [128…
```
df1.index
```

Out[128]:
```
Index(['rank1', 'rank2', 'rank3', 'rank4', 'rank5', 'rank6', 'rank7'], dtype='object')
```

In [129…
```
df1.columns
```

Out[129]:
```
Index(['name', 'age'], dtype='object')
```

In [130…
```
df1.dtypes
```

Out[130]:
```
name    object
age      int64
dtype: object
```

In [131…
```
df1.shape
```

Out[131]:
```
(7, 2)
```

In [132…
```
df1.head()
```

Out[132]:

	name	age
rank1	jasmin	23
rank2	zaru	20
rank3	lav	45
rank4	devi	12
rank5	chandu	89

In [133…
```
df1.head(6)
```

Out[133]:

	name	age
rank1	jasmin	23
rank2	zaru	20
rank3	lav	45
rank4	devi	12
rank5	chandu	89
rank6	deepu	67

In [134…
```
df1[2:4] # getting some range of rows
```

Out[134]:

	name	age
rank3	lav	45
rank4	devi	12

In [135…
```
df1["name"] #extracting specific column
```

Out[135]:
```
rank1    jasmin
rank2      zaru
rank3       lav
rank4      devi
rank5    chandu
rank6     deepu
rank7     susmi
Name: name, dtype: object
```

In [136…
```
df1.name #dataframe passing as object and extracting
```

Out[136]:
```
rank1    jasmin
rank2      zaru
rank3       lav
rank4      devi
rank5    chandu
rank6     deepu
rank7     susmi
Name: name, dtype: object
```

In [137…
```
df2 = pd.DataFrame(np.random.randint(0,60,30).reshape(6,5), index = list('abcdef'), columns = list('pqrst'))
df2
```

Out[137]:

	p	q	r	s	t
a	25	22	25	40	5
b	31	56	41	40	19
c	34	54	50	22	45
d	56	33	42	25	56
e	17	28	8	46	22
f	9	51	53	0	51

In [138…
```
df2['p']
df2.p
```

Out[138]:
```
a    25
b    31
c    34
d    56
e    17
f     9
Name: p, dtype: int32
```

In [139…
```
df2['p':'q']
```

Out[139]:

	p	q	r	s	t

In [140…
```
df2.loc['a':'d','p':'r'] # label based indexing
```

Out[140]:

	p	q	r
a	25	22	25
b	31	56	41
c	34	54	50
d	56	33	42

In [141…
```
df2.iloc[0:3, 1:4] # Position based indexing
```

Out[141]:

	q	r	s
a	22	25	40
b	56	41	40
c	54	50	22

Reading CSV data

In [143…
```
new_df = pd.read_csv(r"C:\Users\jasmi\Desktop\Datasets/Sucidedata.csv") # r is for to avoide unicode error.
#read_excel wuth .xlsx extension is for importing excel files
new_df
```

Out[143]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	genera
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,156,624,900	796	Genera
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,156,624,900	796	S
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,156,624,900	796	Genera
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,156,624,900	796	Genera
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,156,624,900	796	Boor
...	...	...	...	...	...	...	...	...	...	...	...	
27815	Uzbekistan	2014	female	35-54 years	107	3620833	2.96	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27816	Uzbekistan	2014	female	75+ years	9	348465	2.58	Uzbekistan2014	0.675	63,067,077,179	2309	S
27817	Uzbekistan	2014	male	5-14 years	60	2762158	2.17	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27818	Uzbekistan	2014	female	5-14 years	44	2631600	1.67	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27819	Uzbekistan	2014	female	55-74 years	21	1438935	1.46	Uzbekistan2014	0.675	63,067,077,179	2309	Boor

27820 rows × 12 columns

In [145…
```
new_df.head()
```

Out[145]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	generation
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,156,624,900	796	Generation X
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,156,624,900	796	Silent
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,156,624,900	796	Generation X
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,156,624,900	796	G.I. Generation
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,156,624,900	796	Boomers

In [146…
```
new_df[10:20]
```

Out[146]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	generation
10	Albania	1987	female	55-74 years	0	144600	0.00	Albania1987	NaN	2,156,624,900	796	G.I. Generation
11	Albania	1987	male	5-14 years	0	338200	0.00	Albania1987	NaN	2,156,624,900	796	Generation X
12	Albania	1988	female	75+ years	2	36400	5.49	Albania1988	NaN	2,126,000,000	769	G.I. Generation
13	Albania	1988	male	15-24 years	17	319200	5.33	Albania1988	NaN	2,126,000,000	769	Generation X
14	Albania	1988	male	75+ years	1	22300	4.48	Albania1988	NaN	2,126,000,000	769	G.I. Generation
15	Albania	1988	male	35-54 years	14	314100	4.46	Albania1988	NaN	2,126,000,000	769	Silent
16	Albania	1988	male	55-74 years	4	140200	2.85	Albania1988	NaN	2,126,000,000	769	G.I. Generation
17	Albania	1988	female	15-24 years	8	295600	2.71	Albania1988	NaN	2,126,000,000	769	Generation X
18	Albania	1988	female	55-74 years	3	147500	2.03	Albania1988	NaN	2,126,000,000	769	G.I. Generation
19	Albania	1988	female	25-34 years	5	262400	1.91	Albania1988	NaN	2,126,000,000	769	Boomers

In [147…
```
new_df.columns
```

Out[147]:
```
Index(['country', 'year', 'sex', 'age', 'suicides_no', 'population',
       'suicides/100k pop', 'country-year', 'HDI for year',
       ' gdp_for_year ($) ', 'gdp_per_capita ($)', 'generation'],
      dtype='object')
```

In [148…
```
new_df.index
```

Out[148]:
```
RangeIndex(start=0, stop=27820, step=1)
```

In [149…
```
new_df.tail()
```

Out[149]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	genera
27815	Uzbekistan	2014	female	35-54 years	107	3620833	2.96	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27816	Uzbekistan	2014	female	75+ years	9	348465	2.58	Uzbekistan2014	0.675	63,067,077,179	2309	S
27817	Uzbekistan	2014	male	5-14 years	60	2762158	2.17	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27818	Uzbekistan	2014	female	5-14 years	44	2631600	1.67	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27819	Uzbekistan	2014	female	55-74 years	21	1438935	1.46	Uzbekistan2014	0.675	63,067,077,179	2309	Boor

In [151…
```
# write final dataframe as csv or excel
new_df.to_csv(r"C:\Users\jasmi\Desktop\Datasets/save_newdf.csv")
```

In [163…
```
new_df['dataset']=' ' #empty column adding
new_df
```

Out[163]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	genera
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	2,156,624,900	796	Genera
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	2,156,624,900	796	S
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	2,156,624,900	796	Genera
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	2,156,624,900	796	Genera
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	2,156,624,900	796	Boor
...	...	...	...	...	...	...	...	...	...	...	...	
27815	Uzbekistan	2014	female	35-54 years	107	3620833	2.96	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27816	Uzbekistan	2014	female	75+ years	9	348465	2.58	Uzbekistan2014	0.675	63,067,077,179	2309	S
27817	Uzbekistan	2014	male	5-14 years	60	2762158	2.17	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27818	Uzbekistan	2014	female	5-14 years	44	2631600	1.67	Uzbekistan2014	0.675	63,067,077,179	2309	Genera
27819	Uzbekistan	2014	female	55-74 years	21	1438935	1.46	Uzbekistan2014	0.675	63,067,077,179	2309	Boor

27820 rows × 13 columns

In [164…

```
#adding new row to dataframe
new_df1 = new_df.append(pd.Series(), ignore_index = True)
# new_df.append(pd.Series(name="index_name")) -- for giving other name to index
new_df1
```

```
C:\Users\jasmi\AppData\Local\Temp\ipykernel_19668\3234479105.py:2: FutureWarning: The default dtype for empty S
eries will be 'object' instead of 'float64' in a future version. Specify a dtype explicitly to silence this war
ning.
  new_df1 = new_df.append(pd.Series(), ignore_index = True)
C:\Users\jasmi\AppData\Local\Temp\ipykernel_19668\3234479105.py:2: FutureWarning: The frame.append method is de
precated and will be removed from pandas in a future version. Use pandas.concat instead.
  new_df1 = new_df.append(pd.Series(), ignore_index = True)
```

Out[164]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	gene
0	Albania	1987.0	male	15-24 years	21.0	312900.0	6.71	Albania1987	NaN	2,156,624,900	796.0	Gene
1	Albania	1987.0	male	35-54 years	16.0	308000.0	5.19	Albania1987	NaN	2,156,624,900	796.0	
2	Albania	1987.0	female	15-24 years	14.0	289700.0	4.83	Albania1987	NaN	2,156,624,900	796.0	Gene
3	Albania	1987.0	male	75+ years	1.0	21800.0	4.59	Albania1987	NaN	2,156,624,900	796.0	Gene
4	Albania	1987.0	male	25-34 years	9.0	274300.0	3.28	Albania1987	NaN	2,156,624,900	796.0	Bo
...	...	...	...	...	...	...	...	...	...	...	...	
27816	Uzbekistan	2014.0	female	75+ years	9.0	348465.0	2.58	Uzbekistan2014	0.675	63,067,077,179	2309.0	
27817	Uzbekistan	2014.0	male	5-14 years	60.0	2762158.0	2.17	Uzbekistan2014	0.675	63,067,077,179	2309.0	Gene
27818	Uzbekistan	2014.0	female	5-14 years	44.0	2631600.0	1.67	Uzbekistan2014	0.675	63,067,077,179	2309.0	Gene
27819	Uzbekistan	2014.0	female	55-74 years	21.0	1438935.0	1.46	Uzbekistan2014	0.675	63,067,077,179	2309.0	Bo
27820	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

27821 rows × 13 columns

In [165…
```
# update values of entire column
new_df1["dataset"] = "Sucidedata"
new_df1
```

Out[165]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	gene
0	Albania	1987.0	male	15-24 years	21.0	312900.0	6.71	Albania1987	NaN	2,156,624,900	796.0	Gene
1	Albania	1987.0	male	35-54 years	16.0	308000.0	5.19	Albania1987	NaN	2,156,624,900	796.0	
2	Albania	1987.0	female	15-24 years	14.0	289700.0	4.83	Albania1987	NaN	2,156,624,900	796.0	Gene
3	Albania	1987.0	male	75+ years	1.0	21800.0	4.59	Albania1987	NaN	2,156,624,900	796.0	Gene
4	Albania	1987.0	male	25-34 years	9.0	274300.0	3.28	Albania1987	NaN	2,156,624,900	796.0	Bo
...	...	...	...	...	...	...	...	...	...	...	...	
27816	Uzbekistan	2014.0	female	75+ years	9.0	348465.0	2.58	Uzbekistan2014	0.675	63,067,077,179	2309.0	
27817	Uzbekistan	2014.0	male	5-14 years	60.0	2762158.0	2.17	Uzbekistan2014	0.675	63,067,077,179	2309.0	Gene
27818	Uzbekistan	2014.0	female	5-14 years	44.0	2631600.0	1.67	Uzbekistan2014	0.675	63,067,077,179	2309.0	Gene
27819	Uzbekistan	2014.0	female	55-74 years	21.0	1438935.0	1.46	Uzbekistan2014	0.675	63,067,077,179	2309.0	Bo
27820	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

27821 rows × 13 columns

In [170…
```
# replace particular value
new_df1["dataset"] = new_df1["dataset"].str.replace("Sucidedata", "Sucide", regex=False)
new_df1
```

Out[170]:

	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_for_year ($)	gdp_per_capita ($)	gene
0	Albania	1987.0	male	15-24 years	21.0	312900.0	6.71	Albania1987	NaN	2,156,624,900	796.0	Gene
1	Albania	1987.0	male	35-54 years	16.0	308000.0	5.19	Albania1987	NaN	2,156,624,900	796.0	
2	Albania	1987.0	female	15-24 years	14.0	289700.0	4.83	Albania1987	NaN	2,156,624,900	796.0	Gene
3	Albania	1987.0	male	75+ years	1.0	21800.0	4.59	Albania1987	NaN	2,156,624,900	796.0	Gene
4	Albania	1987.0	male	25-34 years	9.0	274300.0	3.28	Albania1987	NaN	2,156,624,900	796.0	Bo
...	...	...	...	...	...	...	...	...	...	...	...	
27816	Uzbekistan	2014.0	female	75+ years	9.0	348465.0	2.58	Uzbekistan2014	0.675	63,067,077,179	2309.0	
27817	Uzbekistan	2014.0	male	5-14 years	60.0	2762158.0	2.17	Uzbekistan2014	0.675	63,067,077,179	2309.0	Gene
27818	Uzbekistan	2014.0	female	5-14 years	44.0	2631600.0	1.67	Uzbekistan2014	0.675	63,067,077,179	2309.0	Gene
27819	Uzbekistan	2014.0	female	55-74 years	21.0	1438935.0	1.46	Uzbekistan2014	0.675	63,067,077,179	2309.0	Bo
27820	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

27821 rows × 13 columns

In [173…
```
# Changing Column names
# new_df1.columns = ['country1', .........] # give all columns in sequence
```

In [174…]:
```
new_df1.to_csv(r"C:\Users\jasmi\Desktop\Datasets/save_newdf.csv")
```

Dropping & Editing Values

In [24]:
```
lt = [("ponnur",2024),("guntur",2001),("nambur",2005),("mangalagiri",np.nan), ("kakani",2005)]
dt = pd.DataFrame(lt, columns = ["city","year"])
dt["country"] = "India"
dt["continent"] = "Asia"
dt
```

Out[24]:

	city	year	country	continent
0	ponnur	2024.0	India	Asia
1	guntur	2001.0	India	Asia
2	nambur	2005.0	India	Asia
3	mangalagiri	NaN	India	Asia
4	kakani	2005.0	India	Asia

In [25]:
```
# dropping a column
# axis 0 for rows
# axis 1 for columns
dt2 = dt.drop("country", axis = 1, inplace = False) # if inplace is True then it drops column from dt also
dt2
```

Out[25]:

	city	year	continent
0	ponnur	2024.0	Asia
1	guntur	2001.0	Asia
2	nambur	2005.0	Asia
3	mangalagiri	NaN	Asia
4	kakani	2005.0	Asia

In [26]:
```
# or use del() function
dt3 = pd.DataFrame(dt2)
del(dt3["continent"])
dt3
```

Out[26]:

	city	year
0	ponnur	2024.0
1	guntur	2001.0
2	nambur	2005.0
3	mangalagiri	NaN
4	kakani	2005.0

In [27]:
```
len(dt3) # returns no of rows
```

Out[27]:
```
5
```

In [28]:
```
dt3.count() # column of count returns excluding null values
```

Out[28]:
```
city    5
year    4
dtype: int64
```

In [31]:
```
dt3.shape
```

Out[31]:
```
(5, 2)
```

In [34]:
```
# gender = ['f','f','m','m','f']
# unique m,f
# nunique 2
# value_counts: f:3, m:2
dt3["year"].value_counts()
```

Out[34]:
```
2005.0    2
2024.0    1
2001.0    1
Name: year, dtype: int64
```

In [35]:
```
dt3.year.nunique()
```

Out[35]:
```
3
```

In [36]:
```
dt3.year.unique()
```

Out[36]:
```
array([2024., 2001., 2005.,   nan])
```

In [37]:
```
dt4 = dt3.drop_duplicates("year") #only gives first ocurring row of duplicate rows
dt4
```

Out[37]:

	city	year
0	ponnur	2024.0
1	guntur	2001.0
2	nambur	2005.0
3	mangalagiri	NaN

Lungs capacity Project

In [65]:
```
lc = pd.read_csv(r"C:\Users\jasmi\Desktop\Datasets/LungCap.csv")
lc
```

Out[65]:

	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean
0	6.475	6	62.1	no	male	no
1	10.125	18	74.7	yes	female	no
2	9.550	16	69.7	no	female	yes
3	11.125	14	71.0	no	male	no
4	4.800	5	56.9	no	male	no
...	...	...	...	...	...	...
720	5.725	9	56.0	no	female	no
721	9.050	18	72.0	yes	male	yes
722	3.850	11	60.5	yes	female	no
723	9.825	15	64.9	no	female	no
724	7.100	10	67.7	no	male	no

725 rows × 6 columns

Data Understanding

In [122…
```
lc.head()
```

Out[122]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean
755	6.475	6	62.1	no	male	no
816	10.125	18	74.7	yes	female	no
12	9.550	16	69.7	no	female	yes
690	11.125	14	71.0	no	male	no
752	4.800	5	56.9	no	male	no

In [123…
```
lc.tail()
```

Out[123]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean
869	5.725	9	56.0	no	female	no
240	9.050	18	72.0	yes	male	yes
328	3.850	11	60.5	yes	female	no
349	9.825	15	64.9	no	female	no
214	7.100	10	67.7	no	male	no

In [124…
```
lc[20:36]
```

Out[124]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean
525	3.975	6	57.3	no	male	no
282	5.325	8	59.7	no	female	no
33	10.025	16	72.4	no	male	no
741	8.725	11	68.0	no	male	yes
64	9.375	11	65.7	no	female	no
363	8.350	12	61.3	no	male	yes
353	6.750	12	60.7	no	female	no
783	9.025	9	65.6	no	male	no
87	1.125	4	48.7	no	female	no
20	10.475	18	72.0	yes	female	no
457	4.650	4	53.7	no	female	no
250	7.725	13	64.7	no	male	no
233	10.600	13	69.3	no	male	no
50	11.025	13	65.6	no	male	yes
309	8.650	12	67.8	no	male	no
365	8.825	10	65.5	no	male	no

In [125…
```
lc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 725 entries, 755 to 214
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   LungCap(cc)     725 non-null    float64
 1   Age( years)     725 non-null    int64  
 2   Height(inches)  725 non-null    float64
 3   Smoke           725 non-null    object 
 4   Gender          725 non-null    object 
 5   Caesarean       725 non-null    object 
dtypes: float64(2), int64(1), object(3)
memory usage: 55.8+ KB
```

In [126…
```
lc.describe() # gives statistical information for dataframe numeric columns
```

Out[126]:

	LungCap(cc)	Age(years)	Height(inches)
count	725.000000	725.000000	725.000000
mean	7.863148	12.326897	64.836276
std	2.662008	4.004750	7.202144
min	0.507000	3.000000	45.300000
25%	6.150000	9.000000	59.900000
50%	8.000000	13.000000	65.400000
75%	9.800000	15.000000	70.300000
max	14.675000	19.000000	81.800000

In [127…
```
list(lc.columns) # checking names of columns
```

Out[127]:
```
['LungCap(cc)',
 'Age( years)',
 'Height(inches)',
 'Smoke',
 'Gender',
 'Caesarean']
```

In [128…
```
for i in lc.columns:
    print(i)
```

```
LungCap(cc)
Age( years)
Height(inches)
Smoke
Gender
Caesarean
```

In [129…
```
lc.groupby(["Gender"]).size() #checking count of male and females in gender column
```

Out[129]:
```
Gender
female    358
male      367
dtype: int64
```

In [130…
```
lc.isnull().sum() # checking missing values
```

Out[130]:
```
LungCap(cc)       0
Age( years)       0
Height(inches)    0
Smoke             0
Gender            0
Caesarean         0
dtype: int64
```

In [131…
```
lc.isnull().values.any() # gives boolean output
```

Out[131]:
```
False
```

In [132…
```
lc["index1"] = np.random.randint(0,1000,725)
lc
```

Out[132]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
755	6.475	6	62.1	no	male	no	870
816	10.125	18	74.7	yes	female	no	940
12	9.550	16	69.7	no	female	yes	701
690	11.125	14	71.0	no	male	no	746
752	4.800	5	56.9	no	male	no	731
...	...	...	...	...	...	...	...
869	5.725	9	56.0	no	female	no	356
240	9.050	18	72.0	yes	male	yes	533
328	3.850	11	60.5	yes	female	no	100
349	9.825	15	64.9	no	female	no	346
214	7.100	10	67.7	no	male	no	17

725 rows × 7 columns

making a column as index lc.set_index("index1", inplace = True) #inplace is true because we are changing master data lc

Data Accessing

In [133…
```
# accessing only age and height columns with all rows
lc.loc[:,["Age( years)", "Height(inches)"]]
```

Out[133]:

index1	Age(years)	Height(inches)
755	6	62.1
816	18	74.7
12	16	69.7
690	14	71.0
752	5	56.9
...	...	...
869	9	56.0
240	18	72.0
328	11	60.5
349	15	64.9
214	10	67.7

725 rows × 2 columns

In [134…
```
lc[["Age( years)", "Height(inches)"]]
```

Out[134]:

index1	Age(years)	Height(inches)
755	6	62.1
816	18	74.7
12	16	69.7
690	14	71.0
752	5	56.9
...	...	...
869	9	56.0
240	18	72.0
328	11	60.5
349	15	64.9
214	10	67.7

725 rows × 2 columns

In [135…
```
# selecting specific columns from specific indexes
lc.loc[[755,240],["Age( years)", "Height(inches)"]]
```

Out[135]:

index1	Age(years)	Height(inches)
755	6	62.1
240	18	72.0

In [136…
```
# selecting rows and columns using loc i.e indexes
lc.iloc[1:8,1:4]
```

Out[136]:

index1	Age(years)	Height(inches)	Smoke
816	18	74.7	yes
12	16	69.7	no
690	14	71.0	no
752	5	56.9	no
461	11	58.7	no
395	8	63.3	no
371	11	70.4	no

In [137…
```
# select every 2nd row after 4th row with all columns
lc.iloc[4::2,:]
```

Out[137]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
752	4.800	5	56.9	no	male	no	731
395	4.950	8	63.3	no	male	yes	778
995	8.875	15	70.5	no	male	no	593
196	11.500	19	76.4	no	male	yes	532
616	6.525	12	57.5	no	male	no	937
...	...	...	...	...	...	...	...
743	7.075	11	66.7	no	male	yes	964
508	7.175	17	68.8	no	male	yes	906
869	5.725	9	56.0	no	female	no	356
328	3.850	11	60.5	yes	female	no	100
214	7.100	10	67.7	no	male	no	17

361 rows × 7 columns

In [138…
```
#select every row till 4th row and all columns
lc.iloc[:4,:]
```

Out[138]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
755	6.475	6	62.1	no	male	no	870
816	10.125	18	74.7	yes	female	no	940
12	9.550	16	69.7	no	female	yes	701
690	11.125	14	71.0	no	male	no	746

In [139…
```
# select every 2nd row and every 2nd column
lc.iloc[::2,::2]
```

Out[139]:

index1	LungCap(cc)	Height(inches)	Gender	index1
755	6.475	62.1	male	870
12	9.550	69.7	female	701
752	4.800	56.9	male	731
395	4.950	63.3	male	778
995	8.875	70.5	male	593
...	...	...	...	...
743	7.075	66.7	male	964
508	7.175	68.8	male	906
869	5.725	56.0	female	356
328	3.850	60.5	female	100
214	7.100	67.7	male	17

363 rows × 4 columns

In [140…
```
lc.loc[::2,::2] # just giving step not labels or position so loc also works here
```

Out[140]:

index1	LungCap(cc)	Height(inches)	Gender	index1
755	6.475	62.1	male	870
12	9.550	69.7	female	701
752	4.800	56.9	male	731
395	4.950	63.3	male	778
995	8.875	70.5	male	593
...	...	...	...	...
743	7.075	66.7	male	964
508	7.175	68.8	male	906
869	5.725	56.0	female	356
328	3.850	60.5	female	100
214	7.100	67.7	male	17

363 rows × 4 columns

In [141…
```
# select rows where age > 15
lc[lc["Age( years)"] > 15]
```

Out[141]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
816	10.125	18	74.7	yes	female	no	940
12	9.550	16	69.7	no	female	yes	701
196	11.500	19	76.4	no	male	yes	532
767	10.925	17	71.7	no	male	no	562
33	10.025	16	72.4	no	male	no	438
...	...	...	...	...	...	...	...
266	9.925	16	68.3	no	female	no	130
667	8.725	19	68.4	no	female	no	878
189	8.825	16	71.3	yes	female	no	617
508	7.175	17	68.8	no	male	yes	906
240	9.050	18	72.0	yes	male	yes	533

177 rows × 7 columns

In [142…
```
# select rows where age > 20 or age < 10
lc[(lc["Age( years)"] < 10) | (lc["Age( years)"] > 20)]
```

Out[142]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
755	6.475	6	62.1	no	male	no	870
752	4.800	5	56.9	no	male	no	731
395	4.950	8	63.3	no	male	yes	778
553	5.050	8	56.1	no	male	no	798
525	3.975	6	57.3	no	male	no	542
...	...	...	...	...	...	...	...
160	3.600	7	53.9	no	male	no	552
940	4.625	5	55.6	no	female	yes	229
197	3.425	3	51.0	no	male	yes	417
137	7.325	9	66.3	no	male	no	294
869	5.725	9	56.0	no	female	no	356

182 rows × 7 columns

In [143…
```
# select all data of not males
lc[lc["Gender"] != "male"]
```

Out[143]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
816	10.125	18	74.7	yes	female	no	940
12	9.550	16	69.7	no	female	yes	701
461	6.225	11	58.7	no	female	no	37
908	6.000	10	61.1	no	female	no	381
985	7.025	11	61.2	yes	female	no	216
...	...	...	...	...	...	...	...
667	8.725	19	68.4	no	female	no	878
189	8.825	16	71.3	yes	female	no	617
869	5.725	9	56.0	no	female	no	356
328	3.850	11	60.5	yes	female	no	100
349	9.825	15	64.9	no	female	no	346

358 rows × 7 columns

In [144…
```
# selecting all of males
lc[lc["Gender"] == "male"]
```

Out[144]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
755	6.475	6	62.1	no	male	no	870
690	11.125	14	71.0	no	male	no	746
752	4.800	5	56.9	no	male	no	731
395	4.950	8	63.3	no	male	yes	778
371	7.325	11	70.4	no	male	no	964
...	...	...	...	...	...	...	...
743	7.075	11	66.7	no	male	yes	964
508	7.175	17	68.8	no	male	yes	906
137	7.325	9	66.3	no	male	no	294
240	9.050	18	72.0	yes	male	yes	533
214	7.100	10	67.7	no	male	no	17

367 rows × 7 columns

In [145…
```
# selecting rows labeled 755 and 214
lc.loc[[755,214], :]
```

Out[145]:

index1	LungCap(cc)	Age(years)	Height(inches)	Smoke	Gender	Caesarean	index1
755	6.475	6	62.1	no	male	no	870
214	8.550	16	67.9	no	male	no	674
214	6.575	14	59.7	yes	female	yes	14
214	7.100	10	67.7	no	male	no	17

In [146…
```
# selecting rows labeled 755 and 214 and columns smoke and gender
lc.loc[[755,214], ["Smoke", "Gender"]]
```

Out[146]:

index1	Smoke	Gender
755	no	male
214	no	male
214	yes	female
214	no	male

Transforming Data

Group by is a function for grouping data objects into series (columns) or DataFrames (group of series) based on particular indicators.

In [147…
```
lc.groupby(["Gender"]).sum()
```

Out[147]:

Gender	LungCap(cc)	Age(years)	Height(inches)	index1
female	2651.257	4457	22842.4	174176
male	3049.525	4480	24163.9	181720

In [148…
```
# count of people from each gender
lc.groupby(["Gender"]).count()
```

Out[148]:

Gender	LungCap(cc)	Age(years)	Height(inches)	Smoke	Caesarean	index1
female	358	358	358	358	358	358
male	367	367	367	367	367	367

In [149…
```
lc.groupby("Gender")["Smoke"].count()
```

Out[149]:
```
Gender
female    358
male      367
Name: Smoke, dtype: int64
```

In [150…
```
lc.groupby("Gender")["Smoke"].count().reset_index()
```

Out[150]:

	Gender	Smoke
0	female	358
1	male	367

In [153…
```
#sort based on Smoke Count
lc.groupby("Gender")["Smoke"].count().reset_index().sort_values(by="Smoke", ascending=False)
```

Out[153]:

	Gender	Smoke
1	male	367
0	female	358

In [155…
```
lc.groupby(["Gender"]).mean().reset_index()
```

Out[155]:

	Gender	LungCap(cc)	Age(years)	Height(inches)	index1
0	female	7.405746	12.449721	63.805587	486.525140
1	male	8.309332	12.207084	65.841689	495.149864

Pivot Table in Pandas

In [10]:
```
import numpy as np
import pandas as pd
sf = pd.read_excel(r"C:\Users\jasmi\Desktop\Datasets/sales_funnel.xlsx")
sf
```

Out[10]:

	Account	Name	Rep	Manager	Product	Quantity	Price	Status
0	714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
1	714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
2	714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
3	737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
4	146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won
5	218895	Kulas Inc	Daniel Hilton	Debra Henley	CPU	2	40000	pending
6	218895	Kulas Inc	Daniel Hilton	Debra Henley	Software	1	10000	presented
7	412290	Jerde-Hilpert	John Smith	Debra Henley	Maintenance	2	5000	pending
8	740150	Barton LLC	John Smith	Debra Henley	CPU	1	35000	declined
9	141962	Herman LLC	Cedric Moss	Fred Anderson	CPU	2	65000	won
10	163416	Purdy-Kunde	Cedric Moss	Fred Anderson	CPU	1	30000	presented
11	239344	Stokes LLC	Cedric Moss	Fred Anderson	Maintenance	1	5000	pending
12	239344	Stokes LLC	Cedric Moss	Fred Anderson	Software	1	10000	presented
13	307599	Kassulke, Ondricka and Metz	Wendy Yule	Fred Anderson	Maintenance	3	7000	won
14	688981	Keeling LLC	Wendy Yule	Fred Anderson	CPU	5	100000	won
15	729833	Koepp Ltd	Wendy Yule	Fred Anderson	CPU	2	65000	declined
16	729833	Koepp Ltd	Wendy Yule	Fred Anderson	Monitor	2	5000	presented

In [12]:
```
pd.pivot_table(sf, index=["Name"])
```

Out[12]:

Name	Account	Price	Quantity
Barton LLC	740150	35000	1.000000
Fritsch, Russel and Anderson	737550	35000	1.000000
Herman LLC	141962	65000	2.000000
Jerde-Hilpert	412290	5000	2.000000
Kassulke, Ondricka and Metz	307599	7000	3.000000
Keeling LLC	688981	100000	5.000000
Kiehn-Spinka	146832	65000	2.000000
Koepp Ltd	729833	35000	2.000000
Kulas Inc	218895	25000	1.500000
Purdy-Kunde	163416	30000	1.000000
Stokes LLC	239344	7500	1.000000
Trantow-Barrows	714466	15000	1.333333

In [13]:
```
pd.pivot_table(sf, index=["Name", "Rep", "Manager"]) # passing multiple indexes
```

Out[13]:

Name	Rep	Manager	Account	Price	Quantity
Barton LLC	John Smith	Debra Henley	740150	35000	1.000000
Fritsch, Russel and Anderson	Craig Booker	Debra Henley	737550	35000	1.000000
Herman LLC	Cedric Moss	Fred Anderson	141962	65000	2.000000
Jerde-Hilpert	John Smith	Debra Henley	412290	5000	2.000000
Kassulke, Ondricka and Metz	Wendy Yule	Fred Anderson	307599	7000	3.000000
Keeling LLC	Wendy Yule	Fred Anderson	688981	100000	5.000000
Kiehn-Spinka	Daniel Hilton	Debra Henley	146832	65000	2.000000
Koepp Ltd	Wendy Yule	Fred Anderson	729833	35000	2.000000
Kulas Inc	Daniel Hilton	Debra Henley	218895	25000	1.500000
Purdy-Kunde	Cedric Moss	Fred Anderson	163416	30000	1.000000
Stokes LLC	Cedric Moss	Fred Anderson	239344	7500	1.000000
Trantow-Barrows	Craig Booker	Debra Henley	714466	15000	1.333333

In [14]:
```
pd.pivot_table(sf, index=["Manager", "Rep"])
```

Out[14]:

Manager	Rep	Account	Price	Quantity
Debra Henley	Craig Booker	720237.0	20000.000000	1.250000
	Daniel Hilton	194874.0	38333.333333	1.666667
	John Smith	576220.0	20000.000000	1.500000
Fred Anderson	Cedric Moss	196016.5	27500.000000	1.250000
	Wendy Yule	614061.5	44250.000000	3.000000

In [15]:
```
pd.pivot_table(sf, index=["Manager", "Rep"], values = ["Price"])
```

Out[15]:

Manager	Rep	Price
Debra Henley	Craig Booker	20000.000000
	Daniel Hilton	38333.333333
	John Smith	20000.000000
Fred Anderson	Cedric Moss	27500.000000
	Wendy Yule	44250.000000

In [17]:
```
#aggfun
pd.pivot_table(sf, index=["Manager", "Rep"], values = ["Price"],aggfunc=[len,np.mean])
```

Out[17]:

		len	mean
		Price	Price
Manager	Rep		
Debra Henley	Craig Booker	4	20000.000000
	Daniel Hilton	3	38333.333333
	John Smith	2	20000.000000
Fred Anderson	Cedric Moss	4	27500.000000
	Wendy Yule	4	44250.000000

In [18]:
```
#sum of price
pd.pivot_table(sf, index=["Manager", "Rep"], values = ["Price"],aggfunc=[np.sum])
```

Out[18]:

		sum
		Price
Manager	Rep	
Debra Henley	Craig Booker	80000
	Daniel Hilton	115000
	John Smith	40000
Fred Anderson	Cedric Moss	110000
	Wendy Yule	177000

In [19]:
```
pd.pivot_table(sf, index=["Manager", "Rep"], values = ["Price"], columns= ["Product"], aggfunc=[np.sum])
```

Out[19]:

		sum			
		Price			
	Product	CPU	Maintenance	Monitor	Software
Manager	Rep				
Debra Henley	Craig Booker	65000.0	5000.0	NaN	10000.0
	Daniel Hilton	105000.0	NaN	NaN	10000.0
	John Smith	35000.0	5000.0	NaN	NaN
Fred Anderson	Cedric Moss	95000.0	5000.0	NaN	10000.0
	Wendy Yule	165000.0	7000.0	5000.0	NaN

In [20]:
```
sf["Product"].unique()
```

Out[20]:
```
array(['CPU', 'Software', 'Maintenance', 'Monitor'], dtype=object)
```

In [21]:
```
pd.pivot_table(sf, index=["Manager", "Rep"], values = ["Price"], columns= ["Product"], aggfunc=[np.sum], fill_v
```

Out[21]:

		sum			
		Price			
	Product	CPU	Maintenance	Monitor	Software
Manager	Rep				
Debra Henley	Craig Booker	65000	5000	0	10000
	Daniel Hilton	105000	0	0	10000
	John Smith	35000	5000	0	0
Fred Anderson	Cedric Moss	95000	5000	0	10000
	Wendy Yule	165000	7000	5000	0

In [22]:
```
pd.pivot_table(sf, index=["Manager", "Rep"], values = ["Price", "Quantity"], columns= ["Product"], aggfunc=[np.
```

Out[22]:

		sum							
		Price				Quantity			
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software
Manager	Rep								
Debra Henley	Craig Booker	65000	5000	0	10000	2	2	0	1
	Daniel Hilton	105000	0	0	10000	4	0	0	1
	John Smith	35000	5000	0	0	1	2	0	0
Fred Anderson	Cedric Moss	95000	5000	0	10000	3	1	0	1
	Wendy Yule	165000	7000	5000	0	7	3	2	0

In [23]:
```
pd.pivot_table(sf, index=["Manager", "Rep", "Product"], values = ["Price", "Quantity"], aggfunc=[np.sum], fill_
```

Out[23]:

			sum	
			Price	Quantity
Manager	Rep	Product		
Debra Henley	Craig Booker	CPU	65000	2
		Maintenance	5000	2
		Software	10000	1
	Daniel Hilton	CPU	105000	4
		Software	10000	1
	John Smith	CPU	35000	1
		Maintenance	5000	2
Fred Anderson	Cedric Moss	CPU	95000	3
		Maintenance	5000	1
		Software	10000	1
	Wendy Yule	CPU	165000	7
		Maintenance	7000	3
		Monitor	5000	2

In [25]:
```
# for getting total added magins = True
pd.pivot_table(sf, index=["Manager", "Rep", "Product"], values = ["Price", "Quantity"], aggfunc=[np.sum], fill_
```

Out[25]:

			sum	
			Price	Quantity
Manager	Rep	Product		
Debra Henley	Craig Booker	CPU	65000	2
		Maintenance	5000	2
		Software	10000	1
	Daniel Hilton	CPU	105000	4
		Software	10000	1
	John Smith	CPU	35000	1
		Maintenance	5000	2
Fred Anderson	Cedric Moss	CPU	95000	3
		Maintenance	5000	1
		Software	10000	1
	Wendy Yule	CPU	165000	7
		Maintenance	7000	3
		Monitor	5000	2
All			522000	30

In [29]:
```
ft = pd.pivot_table(sf, index=["Manager", "Status"], columns = ["Product"],
                    values = ["Price", "Quantity"], aggfunc={"Quantity":len, "Price":[np.sum, np.mean]}, fill_value
ft
```

Out[29]:

		Price				Quantity							
		mean				sum				len			
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software
Manager	Status												
Debra Henley	declined	35000	0	0	0	70000	0	0	0	2	0	0	0
	pending	40000	5000	0	0	40000	10000	0	0	1	2	0	0
	presented	30000	0	0	10000	30000	0	0	20000	1	0	0	2
	won	65000	0	0	0	65000	0	0	0	1	0	0	0
Fred Anderson	declined	65000	0	0	0	65000	0	0	0	1	0	0	0
	pending	0	5000	0	0	0	5000	0	0	0	1	0	0
	presented	30000	0	5000	10000	30000	0	5000	10000	1	0	1	1
	won	82500	7000	0	0	165000	7000	0	0	2	1	0	0

In [33]:
```
# advance pivot table filtering
ft.query('Manager == ["Debra Henley"]') # single quotes only for whole query
```

Out[33]:

		Price				Quantity							
		mean				sum				len			
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software
Manager	Status												
Debra Henley	declined	35000	0	0	0	70000	0	0	0	2	0	0	0
	pending	40000	5000	0	0	40000	10000	0	0	1	2	0	0
	presented	30000	0	0	10000	30000	0	0	20000	1	0	0	2
	won	65000	0	0	0	65000	0	0	0	1	0	0	0

In [34]:
```
ft.query('Status == ["pending"]')
```

Out[34]:

		Price				Quantity							
		mean				sum				len			
	Product	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software	CPU	Maintenance	Monitor	Software
Manager	Status												
Debra Henley	pending	40000	5000	0	0	40000	10000	0	0	1	2	0	0
Fred Anderson	pending	0	5000	0	0	0	5000	0	0	0	1	0	0

Outlier in dataset

What is an outlier?

- An outlier is a data point in a data set that is distant from all other observations. A data point that lies outside the overall distribution of the dataset.
- An outliers causes significant problem to mean & standard deviation of dataset.

Varios ways to find outliers

- suing Scatter Plots
- box plot
- using Z Score
- using IQR (Inter Quantile Range)

In [41]:
```
ds = [1,16,19,1,16,7,6,0,16,11,20,3,7,15,10,10,18,8,13,12,5,19,15,0,13,6,8,1000,1200]
len(ds)
```

Out[41]:
```
29
```

In [42]:
```
np.mean(ds)
```

Out[42]:
```
85.34482758620689
```

In [46]:
```
np.median(ds) # best to use when we have outliers to take center of dataset
```

Out[46]:
```
11.0
```

In [47]:
```
np.std(ds)
```

Out[47]:
```
277.4621783311747
```

Detecting Outlier using Z-Score

Z score is also called standard score. This helps to know data value is greater than or smaller than mean and how far away from the mean. It specifically tells how many standard deviations away the data point is from the mean.

- Z score = (x - mean) / std.deviation

In a normal distribution it is estimated that

- 68% of the data points lie between +/- 1 standard deviation.
- 95% of the data points lie between +/- 2 standard deviation.
- 99.7% of the data points lie between +/- 3 standard deviation.

In [52]:
```
outliers = []
def detect_outliers(ds):
    threshold = 3
    mean = np.mean(ds)
    std = np.std(ds)

    for i in ds:
        zscore = (i - mean) / std
        if abs(zscore) > 3:
            outliers.append(i)
    return outliers
```

In [53]:
```
out_li = detect_outliers(ds)
out_li
```

Out[53]:
```
[1000, 1200]
```

InterQuantile Range

75%-25% values in a dataset

steps

1. Arrange the data in increasing order
2. calculate first(q1) and third quartile(q3)
3. Find the interquantile range (q3-q1) - IQR
4. Find lower bound q1 - (1.5 * IQR)
5. Find upper bound q3 + 1(.5 * IQR)

Anything that lies outside of lower and upper bound is an outlier

In [75]:
```
ds1 = [1,16,19,1,16,7,6,0,16,11,20,3,7,15,10,10,18,8,13,12,5,19,15,0,13,6,8,1000,1200]
q1, q3 = np.percentile(ds1,[25,75])
print(q1, q3)
```

```
6.0 16.0
```

In [76]:
```
iqr = q3 - q1
iqr
```

Out[76]:
```
10.0
```

In [77]:
```
lb = q1 - (1.5 * iqr)
ub = q3 + (1.5 * iqr)
print(lb, ub)
```

```
-9.0 31.0
```

In [78]:
```
out = []
for i in ds1:
    if (i < lb) or (i > ub):
        out.append(i)
print(out)
```

```
[1000, 1200]
```

Traffic Police Data case Study

In [35]:
```
import matplotlib.pyplot as plt
%matplotlib inline
ps = pd.read_csv(r"C:\Users\jasmi\Desktop\Datasets/police.csv")
ps
```

Out[35]:

	stop_date	stop_time	county_name	driver_gender	driver_age_raw	driver_age	driver_race	violation_raw	violation	search_co
0	02-01-2005	01:55	NaN	M	1985.0	20.0	White	Speeding	Speeding	
1	18-01-2005	08:15	NaN	M	1965.0	40.0	White	Speeding	Speeding	
2	23-01-2005	23:15	NaN	M	1972.0	33.0	White	Speeding	Speeding	
3	20-02-2005	17:15	NaN	M	1986.0	19.0	White	Call for Service	Other	
4	14-03-2005	10:00	NaN	F	1984.0	21.0	White	Speeding	Speeding	
...	...	...	...	...	...	...	...	...	...	
91736	31-12-2015	20:27	NaN	M	1986.0	29.0	White	Speeding	Speeding	
91737	31-12-2015	20:35	NaN	F	1982.0	33.0	White	Equipment/Inspection Violation	Equipment	
91738	31-12-2015	20:45	NaN	M	1992.0	23.0	White	Other Traffic Violation	Moving violation	
91739	31-12-2015	21:42	NaN	M	1993.0	22.0	White	Speeding	Speeding	
91740	31-12-2015	22:46	NaN	M	1959.0	56.0	Hispanic	Speeding	Speeding	

91741 rows × 15 columns

In [36]:
```
ps.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 91741 entries, 0 to 91740
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   stop_date           91741 non-null  object
 1   stop_time           91741 non-null  object
 2   county_name         0 non-null      float64
 3   driver_gender       86406 non-null  object
 4   driver_age_raw      86414 non-null  float64
 5   driver_age          86120 non-null  float64
 6   driver_race         86408 non-null  object
 7   violation_raw       86408 non-null  object
 8   violation           86408 non-null  object
 9   search_conducted    91741 non-null  bool
 10  search_type         3196 non-null   object
 11  stop_outcome        86408 non-null  object
 12  is_arrested         86408 non-null  object
 13  stop_duration       86408 non-null  object
 14  drugs_related_stop  91741 non-null  bool
dtypes: bool(2), float64(3), object(10)
memory usage: 9.3+ MB
```

In [37]:
```
ps.dtypes
```

Out[37]:
```
stop_date             object
stop_time             object
county_name          float64
driver_gender         object
driver_age_raw       float64
driver_age           float64
driver_race           object
violation_raw         object
violation             object
search_conducted        bool
search_type           object
stop_outcome          object
is_arrested           object
stop_duration         object
drugs_related_stop      bool
dtype: object
```

In [39]:
```
# converting stop_date and stop_time columns from object to date and timme
ps["stop_date"] = pd.to_datetime(ps.stop_date)
ps["stop_time"] = pd.to_datetime(ps.stop_time)
```

In [40]:
```
ps.dtypes
```

Out[40]:
```
stop_date             datetime64[ns]
stop_time             datetime64[ns]
county_name                  float64
driver_gender                 object
driver_age_raw               float64
driver_age                   float64
driver_race                   object
violation_raw                 object
violation                     object
search_conducted                bool
search_type                   object
stop_outcome                  object
is_arrested                   object
stop_duration                 object
drugs_related_stop              bool
dtype: object
```

In [41]:
```
# Search conducted for each age group for male & female
pd.pivot_table(ps, index = ["driver_gender", "violation"])
```

Out[41]:

		driver_age	driver_age_raw	drugs_related_stop	search_conducted
driver_gender	**violation**				
F	**Equipment**	31.521739	1978.642397	0.008042	0.042622
	Moving violation	33.954930	1972.106117	0.009363	0.036205
	Other	33.969343	1962.591304	0.010145	0.056522
	Registration/plates	32.850791	1976.183613	0.003949	0.066140
	Seat belt	30.124409	1983.960630	0.006299	0.012598
	Speeding	32.529023	1974.767666	0.002002	0.008720
M	**Equipment**	31.857210	1977.442517	0.022852	0.070081
	Moving violation	36.653404	1966.953994	0.016974	0.059831
	Other	40.620631	1912.469810	0.005514	0.047146
	Registration/plates	32.783023	1975.455974	0.016122	0.110376
	Seat belt	32.776867	1981.343116	0.017264	0.037980
	Speeding	34.000729	1972.413384	0.006186	0.024925

In [42]:
```
# min, max, average age of male and female for each violation and how many cases for each age on the basis of v
pd.pivot_table(ps, index = ["driver_gender", "violation"], values = ["driver_age"], aggfunc = [np.mean, np.max,
```

Out[42]:

		mean	amax	amin	len
		driver_age	driver_age	driver_age	driver_age
driver_gender	**violation**				
F	**Equipment**	31.521739	89.0	16.0	2487
	Moving violation	33.954930	99.0	15.0	3204
	Other	33.969343	74.0	16.0	690
	Registration/plates	32.850791	72.0	16.0	1013
	Seat belt	30.124409	71.0	17.0	635
	Speeding	32.529023	84.0	16.0	15482
M	**Equipment**	31.857210	85.0	16.0	8533
	Moving violation	36.653404	94.0	15.0	13020
	Other	40.620631	87.0	16.0	3627
	Registration/plates	32.783023	74.0	16.0	2419
	Seat belt	32.776867	77.0	17.0	2317
	Speeding	34.000729	90.0	15.0	32979

In [43]:
```
# How may people were arrested on each date
ps.groupby("stop_date").is_arrested.value_counts()
```

Out[43]:
```
stop_date   is_arrested
2005-01-04  False           1
2005-01-10  False          27
            True            2
2005-01-11  False          34
            True            3
                           ..
2015-12-28  False          20
2015-12-29  False          12
2015-12-30  False          21
            True            1
2015-12-31  False          24
Name: is_arrested, Length: 5681, dtype: int64
```

In [44]:
```
# On which time police is more active
ps["stop_time"].mean()
```

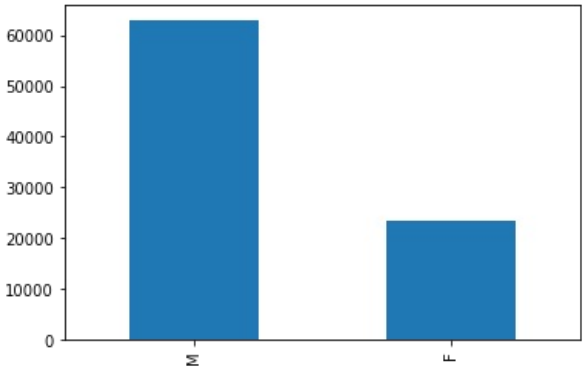
Out[44]:
```
Timestamp('2025-04-10 12:08:17.488363776')
```

In [45]:
```
#how many male & female drivers are there.
ps["driver_gender"].value_counts()
```

Out[45]:
```
M    62895
F    23511
Name: driver_gender, dtype: int64
```

In [46]:
```
ps.driver_gender.value_counts().plot.bar()
```

Out[46]:
```
<AxesSubplot:>
```



In [47]:
```
# find max, min, avg age of male and female
ps.groupby("driver_gender").driver_age.agg(["min", "max", "mean"])
```

Out[47]:

driver_gender	min	max	mean
F	15.0	99.0	32.607399
M	15.0	94.0	34.537886

In [50]:
```
# check drug activity for both male and female
ps.groupby("driver_gender" ).drugs_related_stop.value_counts()
```

Out[50]:
```
driver_gender  drugs_related_stop
F              False                23415
               True                    96
M              False                62176
               True                   719
Name: drugs_related_stop, dtype: int64
```

In [52]:
```
# extract true cases for male and female
ps.groupby("driver_gender" ).drugs_related_stop.sum()
```

Out[52]:
```
driver_gender
F     96
M    719
Name: drugs_related_stop, dtype: int64
```

In [53]:
```
# total cases for true and false for drugs_related_stop
ps["drugs_related_stop"].value_counts()
```

Out[53]:
```
False    90926
True       815
Name: drugs_related_stop, dtype: int64
```

In [54]:
```
# what type of action taken by police for each date
ps.groupby("stop_date").stop_outcome.value_counts()
```

Out[54]:
```
stop_date   stop_outcome
2005-01-04  Citation            1
2005-01-10  Citation           27
            Arrest Driver       2
2005-01-11  Citation           33
            Arrest Driver       3
                               ..
2015-12-30  N/D                 2
            Arrest Passenger    1
2015-12-31  Citation           16
            Warning             7
            No Action           1
Name: stop_outcome, Length: 8970, dtype: int64
```

In [64]:
```
# how many times police has given warning as stop outcome
warning_data = ps[ps["stop_outcome"]=="Warning"]
print(len(ps[ps["stop_outcome"]=="Warning"]))
```

```
warning_data
```

5294

Out[64]:

	stop_date	stop_time	county_name	driver_gender	driver_age_raw	driver_age	driver_race	violation_raw	violation	sea
99	2005-03-10	2025-04-10 08:37:00	NaN	M	1965.0	40.0	White	Equipment/Inspection Violation	Equipment	
108	2005-03-10	2025-04-10 17:00:00	NaN	M	1961.0	44.0	White	Equipment/Inspection Violation	Equipment	
168	2005-06-10	2025-04-10 06:30:00	NaN	M	1970.0	35.0	White	Speeding	Speeding	
260	2005-09-10	2025-04-10 16:00:00	NaN	F	1978.0	27.0	Black	Registration Violation	Registration/plates	
281	2005-10-10	2025-04-10 12:05:00	NaN	F	0.0	NaN	White	Other Traffic Violation	Moving violation	
...	...	...	...	...	...	...	...	...	...	
91730	2015-12-31	2025-04-10 15:36:00	NaN	F	1996.0	19.0	Hispanic	Equipment/Inspection Violation	Equipment	
91732	2015-12-31	2025-04-10 19:44:00	NaN	F	1969.0	46.0	White	Speeding	Speeding	
91736	2015-12-31	2025-04-10 20:27:00	NaN	M	1986.0	29.0	White	Speeding	Speeding	
91737	2015-12-31	2025-04-10 20:35:00	NaN	F	1982.0	33.0	White	Equipment/Inspection Violation	Equipment	
91738	2015-12-31	2025-04-10 20:45:00	NaN	M	1992.0	23.0	White	Other Traffic Violation	Moving violation	

5294 rows × 15 columns

In [65]:
```
# Warning cases for male and demale
warning_data.groupby("driver_gender").stop_outcome.value_counts()
```

Out[65]:
```
driver_gender  stop_outcome
F              Warning         1514
M              Warning         3779
Name: stop_outcome, dtype: int64
```

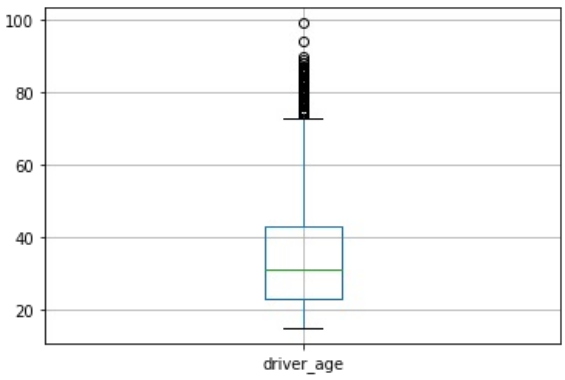
In [67]:
```
# How many male and female for each violation
pd.crosstab(ps["driver_gender"], ps["violation"], margins=True)
```

Out[67]:

violation	Equipment	Moving violation	Other	Registration/plates	Seat belt	Speeding	All
driver_gender							
F	2487	3204	690	1013	635	15482	23511
M	8533	13020	3627	2419	2317	32979	62895
All	11020	16224	4317	3432	2952	48461	86406

In [69]:
```
# data ddistribution visualization for driver_age
ps.boxplot("driver_age")
```

Out[69]:
```
<AxesSubplot:>
```



Weather Condition Analysis

In [74]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import calendar
import datetime

df = pd.read_csv(r"C:\Users\jasmi\Desktop\Datasets/weatherHistory.csv", parse_dates = True, index_col = "Format
df.head()
```

Out[74]:

Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
2006-04-01 00:00:00+02:00	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
2006-04-01 01:00:00+02:00	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2006-04-01 02:00:00+02:00	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
2006-04-01 03:00:00+02:00	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
2006-04-01 04:00:00+02:00	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

In [75]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 96453 entries, 2006-04-01 00:00:00+02:00 to 2016-09-09 23:00:00+02:00
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Summary                   96453 non-null  object 
 1   Precip Type               95936 non-null  object 
 2   Temperature (C)           96453 non-null  float64
 3   Apparent Temperature (C)  96453 non-null  float64
 4   Humidity                  96453 non-null  float64
 5   Wind Speed (km/h)         96453 non-null  float64
 6   Wind Bearing (degrees)    96453 non-null  float64
 7   Visibility (km)           96453 non-null  float64
 8   Loud Cover                96453 non-null  float64
 9   Pressure (millibars)      96453 non-null  float64
 10  Daily Summary             96453 non-null  object 
dtypes: float64(8), object(3)
memory usage: 8.8+ MB
```

In [80]:
```
df.head()
#df[:5]
```

Out[80]:

Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
2006-04-01 00:00:00+02:00	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
2006-04-01 01:00:00+02:00	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2006-04-01 02:00:00+02:00	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
2006-04-01 03:00:00+02:00	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
2006-04-01 04:00:00+02:00	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

In [81]:
```
df["Daily Summary"].unique()
```

Out[81]:
```
array(['Partly cloudy throughout the day.',
       'Mostly cloudy throughout the day.', 'Foggy in the evening.',
       'Foggy overnight and breezy in the morning.',
       'Overcast throughout the day.', 'Partly cloudy until night.',
```

```
        'Mostly cloudy until night.',
        'Foggy starting overnight continuing until morning.',
        'Foggy in the morning.', 'Partly cloudy until evening.',
        'Partly cloudy starting in the morning.',
        'Mostly cloudy starting overnight continuing until night.',
        'Mostly cloudy until evening.',
        'Partly cloudy starting in the morning continuing until evening.',
        'Partly cloudy starting in the afternoon.',
        'Partly cloudy starting overnight.',
        'Partly cloudy until morning.',
        'Partly cloudy starting overnight continuing until night.',
        'Partly cloudy starting in the afternoon continuing until night.',
        'Mostly cloudy starting overnight.',
        'Partly cloudy until afternoon.',
        'Mostly cloudy until night and breezy in the afternoon.',
        'Foggy starting in the evening.', 'Foggy throughout the day.',
        'Foggy starting in the evening continuing until night.',
        'Mostly cloudy until morning.',
        'Foggy starting in the morning continuing until evening.',
        'Foggy starting overnight continuing until afternoon.',
        'Partly cloudy starting in the morning continuing until afternoon.',
        'Foggy starting overnight.', 'Foggy until morning.',
        'Foggy starting overnight continuing until evening.',
        'Foggy starting in the afternoon.',
        'Partly cloudy starting overnight continuing until afternoon.',
        'Partly cloudy starting in the morning continuing until night.',
        'Overcast until night.',
        'Mostly cloudy starting overnight continuing until evening.',
        'Foggy overnight.', 'Partly cloudy in the morning.',
        'Mostly cloudy starting in the morning.',
        'Foggy starting in the afternoon continuing until evening.',
        'Mostly cloudy until afternoon.',
        'Foggy starting overnight continuing until night.',
        'Mostly cloudy throughout the day and breezy in the evening.',
        'Foggy starting in the morning continuing until afternoon.',
        'Partly cloudy in the afternoon.', 'Clear throughout the day.',
        'Partly cloudy starting in the afternoon continuing until evening.',
        'Partly cloudy overnight.', 'Overcast until evening.',
        'Foggy in the morning and breezy starting in the afternoon continuing until night.',
        'Breezy starting overnight continuing until afternoon and foggy starting in the morning continuing until
evening.',
        'Partly cloudy starting overnight continuing until morning.',
        'Mostly cloudy throughout the day and breezy in the afternoon.',
        'Mostly cloudy starting overnight and breezy in the afternoon.',
        'Partly cloudy throughout the day and breezy starting in the morning continuing until night.',
        'Mostly cloudy throughout the day and breezy in the morning.',
        'Partly cloudy starting in the evening continuing until night.',
        'Mostly cloudy until night and breezy starting in the morning continuing until afternoon.',
        'Partly cloudy starting in the morning continuing until evening and breezy starting in the morning conti
nuing until afternoon.',
        'Partly cloudy throughout the day and breezy starting in the morning continuing until afternoon.',
        'Partly cloudy throughout the day and breezy starting in the morning continuing until evening.',
        'Foggy until afternoon.',
        'Overcast until night and breezy overnight.',
        'Breezy until morning and mostly cloudy throughout the day.',
        'Mostly cloudy starting in the morning continuing until night.',
        'Breezy starting overnight continuing until morning and partly cloudy starting overnight continuing unti
l evening.',
        'Partly cloudy in the evening.',
        'Mostly cloudy starting overnight continuing until afternoon.',
        'Mostly cloudy starting in the morning continuing until afternoon.',
        'Mostly cloudy starting in the afternoon.',
        'Mostly cloudy starting in the morning continuing until evening.',
        'Partly cloudy starting overnight continuing until afternoon and breezy in the afternoon.',
        'Partly cloudy starting overnight and breezy in the afternoon.',
        'Mostly cloudy starting in the morning and breezy in the evening.',
        'Foggy starting in the afternoon continuing until night.',
        'Foggy until night.',
        'Foggy starting in the morning continuing until night.',
        'Foggy until evening.', 'Foggy starting in the morning.',
        'Partly cloudy starting overnight continuing until evening.',
        'Partly cloudy starting overnight continuing until evening and breezy starting in the morning continuing
until evening.',
        'Breezy starting overnight continuing until morning and foggy in the evening.',
        'Mostly cloudy throughout the day and breezy starting in the morning continuing until evening.',
        'Partly cloudy until evening and breezy starting in the morning continuing until afternoon.',
        'Mostly cloudy starting in the afternoon continuing until night.',
        'Breezy starting overnight continuing until afternoon and mostly cloudy starting overnight continuing un
til evening.',
        'Mostly cloudy throughout the day and windy starting in the morning continuing until evening.',
        'Breezy and partly cloudy in the afternoon.',
        'Mostly cloudy starting overnight and breezy starting in the morning continuing until afternoon.',
        'Partly cloudy until night and breezy starting in the morning continuing until afternoon.',
        'Breezy and mostly cloudy overnight.',
        'Mostly cloudy throughout the day and breezy overnight.',
        'Mostly cloudy throughout the day and breezy starting in the morning continuing until afternoon.',
        'Partly cloudy throughout the day and breezy in the morning.',
        'Partly cloudy starting in the morning continuing until evening and breezy starting in the afternoon con
tinuing until evening.',
```

```
       'Partly cloudy throughout the day and breezy starting in the afternoon continuing until evening.',
       'Mostly cloudy starting overnight and breezy in the morning.',
       'Partly cloudy starting in the afternoon and breezy in the afternoon.',
       'Partly cloudy starting in the morning and breezy in the evening.',
       'Partly cloudy until evening and breezy in the morning.',
       'Partly cloudy starting overnight continuing until evening and breezy starting in the morning continuing
until afternoon.',
       'Partly cloudy starting overnight continuing until evening and breezy in the evening.',
       'Mostly cloudy throughout the day and breezy starting in the evening.',
       'Mostly cloudy throughout the day and windy starting in the morning continuing until night.',
       'Breezy starting overnight continuing until morning and partly cloudy starting in the morning.',
       'Mostly cloudy starting in the morning and breezy overnight.',
       'Overcast throughout the day and breezy starting overnight continuing until morning.',
       'Partly cloudy throughout the day and breezy in the evening.',
       'Mostly cloudy until evening and breezy starting in the morning continuing until afternoon.',
       'Mostly cloudy until night and breezy in the evening.',
       'Partly cloudy starting in the evening.',
       'Overcast starting in the morning.',
       'Mostly cloudy starting overnight continuing until evening and breezy starting overnight continuing unti
l morning.',
       'Partly cloudy starting overnight continuing until morning and breezy starting in the morning continuing
until afternoon.',
       'Partly cloudy until evening and breezy starting in the morning continuing until evening.',
       'Breezy starting in the morning continuing until afternoon and partly cloudy starting in the morning.',
       'Partly cloudy starting in the morning and breezy starting in the afternoon continuing until evening.',
       'Mostly cloudy starting overnight continuing until morning.',
       'Mostly cloudy throughout the day and breezy starting overnight continuing until afternoon.',
       'Breezy starting overnight continuing until morning and foggy overnight.',
       'Mostly cloudy throughout the day and breezy starting overnight continuing until morning.',
       'Overcast throughout the day and breezy in the morning.',
       'Overcast throughout the day and breezy in the evening.',
       'Mostly cloudy starting in the morning continuing until night and breezy in the afternoon.',
       'Mostly cloudy until night and breezy starting in the evening continuing until night.',
       'Partly cloudy until night and breezy in the morning.',
       'Partly cloudy until evening and breezy overnight.',
       'Partly cloudy starting overnight continuing until night and windy starting in the morning continuing un
til afternoon.',
       'Breezy starting in the morning continuing until afternoon and mostly cloudy starting in the morning.',
       'Foggy starting overnight continuing until morning and breezy starting in the evening.',
       'Mostly cloudy until night and breezy starting in the afternoon.',
       'Foggy in the afternoon.',
       'Mostly cloudy until night and breezy starting in the afternoon continuing until night.',
       'Foggy starting overnight continuing until morning and breezy starting in the evening continuing until n
ight.',
       'Breezy until afternoon and mostly cloudy throughout the day.',
       'Mostly cloudy throughout the day and breezy starting in the morning continuing until night.',
       'Partly cloudy starting overnight continuing until evening and breezy in the morning.',
       'Mostly cloudy starting in the morning and breezy in the afternoon.',
       'Mostly cloudy starting overnight continuing until night and breezy starting in the morning continuing u
ntil evening.',
       'Foggy starting overnight continuing until morning and breezy starting in the morning continuing until a
fternoon.',
       'Mostly cloudy until evening and windy starting in the morning continuing until afternoon.',
       'Foggy starting overnight continuing until afternoon and breezy in the morning.',
       'Foggy starting in the morning continuing until afternoon and breezy starting in the evening.',
       'Partly cloudy starting overnight and breezy starting in the morning continuing until afternoon.',
       'Foggy starting overnight continuing until morning and breezy in the afternoon.',
       'Mostly cloudy starting overnight and breezy starting in the afternoon continuing until evening.',
       'Overcast throughout the day and breezy starting overnight continuing until afternoon.',
       'Partly cloudy starting in the morning continuing until evening and breezy in the afternoon.',
       'Partly cloudy starting in the morning continuing until night and breezy starting in the afternoon conti
nuing until evening.',
       'Mostly cloudy until night and breezy starting in the evening.',
       'Breezy in the morning and mostly cloudy starting in the morning.',
       'Mostly cloudy until night and breezy starting in the morning continuing until evening.',
       'Partly cloudy starting overnight continuing until evening and windy starting in the morning continuing
until evening.',
       'Breezy in the morning and partly cloudy starting in the evening continuing until night.',
       'Partly cloudy overnight and breezy starting in the morning continuing until afternoon.',
       'Light rain in the morning.', 'Light rain until morning.',
       'Light rain in the morning and afternoon.',
       'Partly cloudy starting in the morning continuing until night and breezy starting in the morning continu
ing until afternoon.',
       'Breezy starting in the afternoon continuing until night and mostly cloudy starting in the evening.',
       'Mostly cloudy throughout the day and breezy starting in the evening continuing until night.',
       'Foggy starting in the afternoon and breezy starting in the afternoon continuing until evening.',
       'Breezy and foggy until morning.',
       'Mostly cloudy until night and breezy starting overnight continuing until morning.',
       'Partly cloudy starting overnight continuing until night and breezy in the morning.',
       'Partly cloudy starting overnight continuing until night and breezy in the afternoon.',
       'Mostly cloudy starting in the morning and breezy starting in the afternoon continuing until evening.',
       'Partly cloudy starting overnight and breezy starting in the evening.',
       'Breezy overnight and overcast throughout the day.',
       'Partly cloudy until night and breezy in the afternoon.',
       'Mostly cloudy starting overnight and breezy starting in the evening.',
       'Breezy overnight and partly cloudy until evening.',
       'Mostly cloudy starting in the evening.',
       'Mostly cloudy throughout the day and breezy starting in the afternoon.',
       'Mostly cloudy throughout the day and breezy starting in the afternoon continuing until evening.',
```

```
'Mostly cloudy until night and windy starting in the morning continuing until afternoon.',
'Breezy and foggy starting in the evening.',
'Breezy overnight and partly cloudy throughout the day.',
'Overcast throughout the day and breezy starting in the evening.',
'Breezy until evening and foggy in the morning.',
'Breezy overnight and mostly cloudy throughout the day.',
'Partly cloudy until evening and breezy in the afternoon.',
'Partly cloudy starting in the morning and breezy starting in the morning continuing until afternoon.',
'Mostly cloudy until evening and breezy in the evening.',
'Windy in the afternoon.', 'Overcast until morning.',
'Mostly cloudy overnight.',
'Foggy starting in the morning continuing until evening and breezy in the evening.',
'Breezy starting overnight continuing until morning.',
'Breezy starting in the afternoon continuing until evening and foggy starting in the evening.',
'Mostly cloudy until night and breezy overnight.',
'Mostly cloudy starting in the morning and windy in the evening.',
'Partly cloudy throughout the day and windy starting in the morning continuing until afternoon.',
'Breezy until afternoon and overcast throughout the day.',
'Breezy in the morning and foggy in the evening.',
'Breezy starting in the afternoon continuing until evening and foggy in the evening.',
'Breezy starting in the morning continuing until night.',
'Breezy in the morning and mostly cloudy starting in the evening.',
'Mostly cloudy until evening and breezy in the afternoon.',
'Mostly cloudy until night and breezy starting in the afternoon continuing until evening.',
'Mostly cloudy until evening and breezy starting overnight continuing until morning.',
'Overcast throughout the day and breezy in the afternoon.',
'Overcast throughout the day and breezy starting in the morning continuing until evening.',
'Overcast throughout the day and breezy overnight.',
'Overcast starting in the afternoon.',
'Partly cloudy throughout the day and breezy in the afternoon.',
'Light rain starting overnight.',
'Drizzle starting in the evening.', 'Drizzle until morning.',
'Rain throughout the day.', 'Rain until morning.',
'Light rain overnight.', 'Rain until afternoon.'], dtype=object)
```

In [82]:
```
df["Daily Summary"].value_counts()
```

Out[82]:
```
Mostly cloudy throughout the day.                                                20085
Partly cloudy throughout the day.                                                 9981
Partly cloudy until night.                                                        6169
Partly cloudy starting in the morning.                                            5184
Foggy in the morning.                                                             4201
                                                                                 ...
Breezy starting overnight continuing until morning and foggy overnight.             24
Mostly cloudy throughout the day and breezy starting overnight continuing until afternoon.    24
Partly cloudy starting in the morning and breezy starting in the afternoon continuing until evening.    24
Rain until afternoon.                                                               24
Foggy starting overnight continuing until morning and breezy in the afternoon.      23
Name: Daily Summary, Length: 214, dtype: int64
```

In [87]:
```
len(df[df['Daily Summary'] == "Foggy in the morning."])
```

Out[87]:
```
4201
```

In [90]:
```
df[df['Daily Summary'] == "Foggy in the morning."].count()
```

Out[90]:
```
Summary                     4201
Precip Type                 4201
Temperature (C)             4201
Apparent Temperature (C)    4201
Humidity                    4201
Wind Speed (km/h)           4201
Wind Bearing (degrees)      4201
Visibility (km)             4201
Loud Cover                  4201
Pressure (millibars)        4201
Daily Summary               4201
dtype: int64
```

In [91]:
```
df["Wind Speed (km/h)"].unique()
```

Out[91]:
```
array([14.1197, 14.2646,  3.9284, ..., 37.0622, 35.5971, 30.751 ])
```

In [92]:
```
df["Wind Speed (km/h)"].value_counts()
```

Out[92]:
```
3.2200     2441
11.2700    1495
6.4400     1357
0.0000     1297
8.0500      920
           ...
34.3413       1
45.9333       1
31.5238       1
32.5864       1
41.6990       1
Name: Wind Speed (km/h), Length: 2484, dtype: int64
```

In [94]:
```
df["Wind Speed (km/h)"].value_counts()[31.5238]
```

In [94]:
```
df["Wind Speed (km/h)"].value_counts()[31.5238]
```

Out[94]:
```
1
```

In [96]:
```
len(df[(df["Wind Speed (km/h)"] == 31.5238)])
```

Out[96]:
```
1
```

In [97]:
```
df[["Daily Summary", "Temperature (C)"]].head()
```

Out[97]:

Formatted Date	Daily Summary	Temperature (C)
2006-04-01 00:00:00+02:00	Partly cloudy throughout the day.	9.472222
2006-04-01 01:00:00+02:00	Partly cloudy throughout the day.	9.355556
2006-04-01 02:00:00+02:00	Partly cloudy throughout the day.	9.377778
2006-04-01 03:00:00+02:00	Partly cloudy throughout the day.	8.288889
2006-04-01 04:00:00+02:00	Partly cloudy throughout the day.	8.755556

In [101…
```
df[:10] # first 20 rows oe df.head(10)
```

Out[101]:

Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
2006-04-01 00:00:00+02:00	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
2006-04-01 01:00:00+02:00	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2006-04-01 02:00:00+02:00	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
2006-04-01 03:00:00+02:00	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
2006-04-01 04:00:00+02:00	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
2006-04-01 05:00:00+02:00	Partly Cloudy	rain	9.222222	7.111111	0.85	13.9587	258.0	14.9569	0.0	1016.66	Partly cloudy throughout the day.
2006-04-01 06:00:00+02:00	Partly Cloudy	rain	7.733333	5.522222	0.95	12.3648	259.0	9.9820	0.0	1016.72	Partly cloudy throughout the day.
2006-04-01 07:00:00+02:00	Partly Cloudy	rain	8.772222	6.527778	0.89	14.1519	260.0	9.9820	0.0	1016.84	Partly cloudy throughout the day.
2006-04-01 08:00:00+02:00	Partly Cloudy	rain	10.822222	10.822222	0.82	11.3183	259.0	9.9820	0.0	1017.37	Partly cloudy throughout the day.
2006-04-01 09:00:00+02:00	Partly Cloudy	rain	13.772222	13.772222	0.72	12.5258	279.0	9.9820	0.0	1017.22	Partly cloudy throughout the day.

In [106…
```
# what were the first 5 pressure values recorded  on 2006-04-01
# df.loc["2006-04-01", "Pressure (millibars)"][:5]
```

In [114…
```
# Find all instances when wind speec is above 30 and visibility was 25
df[(df["Wind Speed (km/h)"] > 10) & (df["Visibility (km)"] > 15)]
```

Out[114]:

Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
2006-04-01 00:00:00+02:00	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
2006-04-01 01:00:00+02:00	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2006-04-01 03:00:00+02:00	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
2006-04-01 04:00:00+02:00	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
2006-04-10 00:00:00+02:00	Partly Cloudy	rain	10.422222	10.422222	0.62	16.9855	150.0	15.8263	0.0	1014.40	Mostly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...
2016-09-09 17:00:00+02:00	Partly Cloudy	rain	30.766667	29.311111	0.28	14.2163	24.0	15.5526	0.0	1013.83	Partly cloudy starting in the morning.
2016-09-09 18:00:00+02:00	Partly Cloudy	rain	28.838889	27.850000	0.32	12.2038	21.0	16.1000	0.0	1014.07	Partly cloudy starting in the morning.
2016-09-09 19:00:00+02:00	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
2016-09-09 20:00:00+02:00	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
2016-09-09 22:00:00+02:00	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.

10019 rows × 11 columns

In [115…
```
# which were the top 10 hottest temp values & their counts?
df["Temperature (C)"].value_counts().sort_values(ascending = False).head(10)
```

Out[115]:
```
7.222222     455
7.777778     408
12.777778    378
17.777778    373
6.111111     370
3.888889     369
2.777778     354
13.888889    342
5.000000     341
17.222222    330
Name: Temperature (C), dtype: int64
```

In [117…
```
# or
temp_df = df["Temperature (C)"].value_counts().sort_values(ascending = False)
temp_df.iloc[:10]
```

Out[117]:
```
7.222222     455
7.777778     408
12.777778    378
17.777778    373
6.111111     370
3.888889     369
2.777778     354
13.888889    342
5.000000     341
17.222222    330
Name: Temperature (C), dtype: int64
```

In [125…
```
#mean_temp_series = df.groupby(df['Formatted Date'].df.month)["Temperature (C)"].mean()
#print(mean_temp_series)
```

In [ ]: