Introduction to Pandas

In [4]:

```
import pandas as pd
import os
```

Creating filepath for csv file reading and saving

In [6]:

```
os.chdir('/Users/tomisin/Dropbox/My Mac (Tomisins-MacBook-Pro.local)/Documents/D
```

Importing Dataset

In [9]:

```
df1 = pd.read_csv('Housing_w_headers.csv')
```

In [10]:

```
df1.head()
```

Out[10]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhea
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

In [ ]:

```
!pip install pandas # run this if the import statement for pandas returned an er
```

In [15]:

```
df2 = pd.read_csv('diabetes.csv')
```

In [16]:

```
df3 = pd.concat([df1,df2], axis = 1)
```

In [17]:

```
df3
```

Out[17]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwa
0	13300000.0	7420.0	4.0	2.0	3.0	yes	no	no	
1	12250000.0	8960.0	4.0	4.0	4.0	yes	no	no	
2	12250000.0	9960.0	3.0	2.0	2.0	yes	no	yes	
3	12215000.0	7500.0	4.0	2.0	2.0	yes	no	yes	
4	11410000.0	7420.0	4.0	1.0	2.0	yes	yes	yes	
...	...	...	...	...	...	...	...	...	
763	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
764	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
765	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
766	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
767	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

768 rows × 22 columns

Outputting column names

In [18]:

```
df2.columns
```

Out[18]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'In
sulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

Creating a DataFrame

In [19]:

```
list1 = ['agatha', 'jerry','james','jacob','rita','Abdul']
list2 = [34,45,67,54,23,45,]

list3 = ['US', 'Nigeria', 'Zambia', 'Uganda', 'Ukrain','UAE']
list4  =[97,89,134,65,85,120]

```

In [20]:

```
#dfx = pd.DataFrame([list1,list2], columns = ['name', 'age'])

dfx = pd.DataFrame()
dfx['name'] = list1
dfx['age'] = list2
```

In [21]:

```
dfx
```

Out[21]:

	name	age
0	agatha	34
1	jerry	45
2	james	67
3	jacob	54
4	rita	23
5	Abdul	45

In [22]:

```
dfy = pd.DataFrame()
dfy['country'] = list3
dfy['weight'] = list4
dfy
```

Out[22]:

	country	weight
0	US	97
1	Nigeria	89
2	Zambia	134
3	Uganda	65
4	Ukrain	85
5	UAE	120

In [23]:

```
newDf = pd.concat([dfx,dfy], axis= 1)
newDf
```

Out[23]:

	name	age	country	weight
0	agatha	34	US	97
1	jerry	45	Nigeria	89
2	james	67	Zambia	134
3	jacob	54	Uganda	65
4	rita	23	Ukrain	85
5	Abdul	45	UAE	120

In [24]:

```
newDf = pd.concat([dfx,dfy], axis= 0)
newDf
```

Out[24]:

	name	age	country	weight
0	agatha	34.0	NaN	NaN
1	jerry	45.0	NaN	NaN
2	james	67.0	NaN	NaN
3	jacob	54.0	NaN	NaN
4	rita	23.0	NaN	NaN
5	Abdul	45.0	NaN	NaN
0	NaN	NaN	US	97.0
1	NaN	NaN	Nigeria	89.0
2	NaN	NaN	Zambia	134.0
3	NaN	NaN	Uganda	65.0
4	NaN	NaN	Ukrain	85.0
5	NaN	NaN	UAE	120.0

In [25]:

```
1 df1
```

Out[25]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterl
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	
...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	yes	no	yes	
541	1767150	2400	3	1	1	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	
543	1750000	2910	3	1	1	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	

545 rows × 13 columns

In [26]:

```
1 df1 = pd.read_csv('Housing_w_headers.csv')
```

Printing only first and last columns

In [27]:

```
1 df1.head(7)
```

Out[27]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterhea
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	
5	10850000	7500	3	3	1	yes	no	yes	
6	10150000	8580	4	3	4	yes	no	no	

In [28]:

```
1 df1.tail(7)
```

Out[28]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterh
538	1890000	3649	2	1	1	yes	no	no	
539	1855000	2990	2	1	1	no	no	no	
540	1820000	3000	2	1	1	yes	no	yes	
541	1767150	2400	3	1	1	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	
543	1750000	2910	3	1	1	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	

Deriving information from your data

In [29]:

```
1 df1.shape # gives you the total rows and columns in the DataFrame
```

Out[29]:

```
(545, 13)
```

In [30]:

```
1 df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

In [31]:

```
df1.columns.tolist()
```

Out[31]:

```
['price',
 'area',
 'bedrooms',
 'bathrooms',
 'stories',
 'mainroad',
 'guestroom',
 'basement',
 'hotwaterheating',
 'airconditioning',
 'parking',
 'prefarea',
 'furnishingstatus']
```

In [59]:

```
df1.describe() # Describes the distribution pattern in your data
```

Out[59]:

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

In [ ]:

```
# you cannot carry out any statistical analysis on categorical variables
# convert categoricals to numerics
```

Printing all column names with their index positions

In [35]:

```
collist =df1.columns.tolist()

for col in collist:
    idx = collist.index(col)
    print(col, ': ', idx)
```

```
price :  0
area :  1
bedrooms :  2
bathrooms :  3
stories :  4
mainroad :  5
guestroom :  6
basement :  7
hotwaterheating :  8
airconditioning :  9
parking :  10
prefarea :  11
furnishingstatus :  12
```

Slicing a dataframe and extracting some columns using loc and iloc

In [37]:

```
#loc
subsetDf = df1.loc[:,['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'park
```

In [39]:

```
subsetDf.head()
```

Out[39]:

	price	area	bedrooms	bathrooms	stories	parking
0	13300000	7420	4	2	3	2
1	12250000	8960	4	4	4	3
2	12250000	9960	3	2	2	2
3	12215000	7500	4	2	2	3
4	11410000	7420	4	1	2	2

In [42]:

```
#iloc
subsetDFidx = df.iloc[0:4,[0,1,2,3,4,10]]
```

In [43]:

```
1 subsetDFidx
```

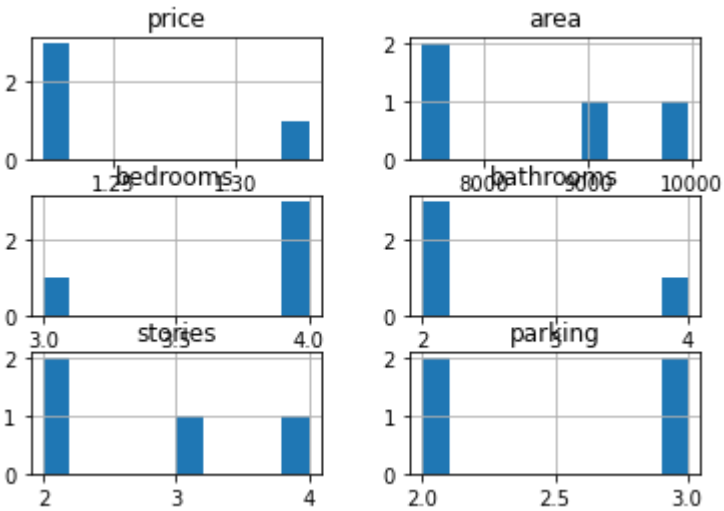
Out[43]:

	price	area	bedrooms	bathrooms	stories	parking
0	13300000	7420	4	2	3	2
1	12250000	8960	4	4	4	3
2	12250000	9960	3	2	2	2
3	12215000	7500	4	2	2	3

Making plots

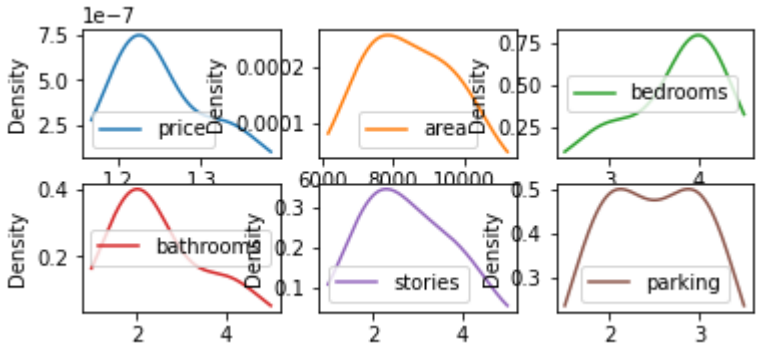
In [44]:

```
1 import matplotlib.pyplot as plt
2
3 subsetDFidx.hist()
4 plt.show()
5
```



In [45]:

```
subsetDFidx.plot(kind='density', subplots=True, layout=(3,3), sharex=False)
plt.show()
```



In [47]:

```
# extract one column and convert to a listabs
# using the price columnabs

priceList = df1.price
priceList1 = df1['price']
```

In [51]:

```
print(priceList1)
print(priceList)
```

```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
```

In [52]:

```
pList = priceList.values.tolist()
```

Checking minimum and maximum values

In [53]:

```
minPrice = min(pList)
```

In [54]:

```
minPrice
```

Out[54]:

```
1750000
```

In [55]:

```
maxPrice = max(pList)
maxPrice
```

Out[55]:

```
13300000
```

Combining two columns to generate a new column

In [57]:

```
area = df1['area']
bathrooms = df1['bathrooms']

# these two columns are still in dataframe format

# converting to a list

areaList = area.values.tolist()
bathroomsList = bathrooms.values.tolist()
```

In [58]:

```
type(area)
```

Out[58]:

```
pandas.core.series.Series
```

In [59]:

```
type(areaList)
```

Out[59]:

```
list
```

In [60]:

```
spaceList = []

for a,b in zip(areaList,bathroomsList):
    space = a*b
    spaceList.append(space)
```

In [61]:

```
# creating a new column in df called totalSpacea

df1['totalSpace'] = spaceList
```

In [62]:

```
df1.head()
```

Out[62]:

rice	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
000	7420	4	2	3	yes	no	no	no	
000	8960	4	4	4	yes	no	no	no	
000	9960	3	2	2	yes	no	yes	no	
000	7500	4	2	2	yes	no	yes	no	
000	7420	4	1	2	yes	yes	yes	no	

Extracting numerical variables from categorical ones

In [63]:

```
dataExtract = df1.iloc[:,[0,1,2,3,4,10,13]]
```

In [64]:

```
dataExtract
```

Out[64]:

	price	area	bedrooms	bathrooms	stories	parking	totalSpace
0	13300000	7420	4	2	3	2	14840
1	12250000	8960	4	4	4	3	35840
2	12250000	9960	3	2	2	2	19920
3	12215000	7500	4	2	2	3	15000
4	11410000	7420	4	1	2	2	7420
...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	2	3000
541	1767150	2400	3	1	1	0	2400
542	1750000	3620	2	1	1	0	3620
543	1750000	2910	3	1	1	0	2910
544	1750000	3850	3	1	2	0	3850

545 rows × 7 columns

Saving dataExtract back to disk/Persisting to file

In [67]:

```
savepath = '/Users/tomisin/Dropbox/My Mac (Tomisins-MacBook-Pro.local)/Documents/Data
savename = 'dataExtract.csv'

# saving dataExtract to file

dataExtract.to_csv(savepath+savename)
```

In [68]:

```
areaVec = df1.area.values
bathsVec = df1.bathrooms.values
totalSpace = areaVec*bathsVec
```

In [69]:

```
1 totalSpace
```

Out[69]:

```
array([14840, 35840, 19920, 15000,  7420, 22500, 25740, 48600,  8100,
       11500, 13200, 18000, 13100,  7000, 15600,  6000, 13200, 17000,
        9200, 12840,  4320, 14310,  8050,  9120, 17600, 13080, 12000,
        8875, 15900, 11000, 14950,  7000,  9760, 17880,  6840, 14000,
       14964, 18000,  6000, 12000,  6550, 12720, 12960, 12000, 12000,
       12000, 12000,  6600,  8600, 14880, 14880,  6325, 12000, 10300,
       12000,  6000, 11440, 18000, 15360, 12000, 12000,  8880, 12480,
       12720, 11175, 17760, 13200, 15400,  6000, 24180,  8000, 12000,
        5020, 13200,  4040,  8520, 12840, 13000,  5700, 12000,  6000,
        8000, 21000, 12000,  3760, 16500,  6670,  3960,  7410, 25740,
        5000,  6750,  9600, 14400, 12000,  8200,  9000,  6400, 13200,
        6000, 13200,  5500, 11000, 12700, 11000,  4500, 10900,  6420,
        3240, 13230,  6600,  8372,  8600,  9620,  6800,  8000, 13800,
        3700,  6420,  7020,  6540,  7231, 12508, 14640, 13050, 15600,
        7160, 13000,  5500, 11460,  4800,  5828,  5200,  4800,  7000,
       12000, 10800,  4640,  5000,  6360, 11600, 13320, 21000,  9600,
        4700,  5000, 10500, 11000,  6360, 13200,  5136,  4400,  5400,
        9900,  7300, 12200,  6900,  5634,  7980,  6300,  6210,  6100,
       13200,  6825, 13420, 12900,  7800,  9200,  4260, 13080, 11000,
       10269,  8400, 10600,  3800, 19600,  8520,  6050,  7085,  6360,
        9000,  7200,  3410,  7980,  6000,  3000, 11410,  6100,  5720,
        3540,  7600, 10700,  6600,  4800, 16300, 13230,  7686,  5600,
        5948,  4200,  4520,  4095,  4120,  5400,  4770,  6300,  5800,
        3000,  2970,  6720,  4646, 12900,  6840,  9990,  4350,  4160,
        6040,  6862,  4815,  7000,  8100,  6840,  9166,  6321, 10240,
        6440,  5170,  6000,  3630, 19334,  5400,  4320,  3745,  4160,
        7760,  5680,  2870,  5010,  9020,  4000,  3840,  3760,  3640,
        2550,  5320,  5360,  3520,  8400,  8200,  9980,  3510,  3450,
        9860,  3520,  4510,  5885,  4000,  8250,  4040,  6360,  3162,
        3510,  3750,  3968,  4900,  2880,  4880,  4920,  4950,  3900,
        9000,  1905,  4075,  3500,  6450,  4032,  4400, 10360,  3400,
        6360,  6360,  4500,  2175,  4360,  7770,  6650,  2787,  5500,
        5040,  5850,  7830,  2953,  5494,  4410,  8000,  2325,  9200,
        7280,  5800,  7000,  4079,  3520,  2145,  4500,  8250,  3450,
        4840,  4080,  4046,  4632,  5985,  6060,  3600,  7360,  4040,
        5600, 11800,  9984,  4340,  3000,  4320,  7260,  6920,  5400,
        4500,  3460,  4100,  6480,  9000,  3960,  4050, 14520,  5500,
        3000,  3290,  3816,  8080,  4290,  3780,  6360, 10600,  6360,
        7152,  4080,  3850,  2015,  2176,  3350,  6300,  4820,  3420,
        3600,  5830,  2856,  8400,  8250,  5040,  6930,  3480,  3600,
        4040,  6020,  4050,  3584,  3120,  5450,  3630,  3630,  5640,
        3600,  4280,  3570,  3180,  3000,  7040,  5960,  8260,  5700,
        2275,  3520,  4500,  4000,  3150,  9000,  4500,  3640,  3850,
        4240,  3650,  4600,  4270,  3036,  3990,  7424,  3480,  3600,
        3640,  5900,  3120,  7350,  3512,  9500,  5880, 12944,  4900,
        3060,  5320,  2145,  4000,  3185,  3850,  2145,  2610,  3900,
        4040,  4785,  3450,  3640,  3500,  4960,  4120,  4750,  3720,
        3750,  3100,  3185,  2700,  2145,  4040,  4775,  2500,  3180,
        6060,  3480,  3792,  4040,  2145,  5880,  4500,  3930,  3640,
        4370,  2684,  4320,  3120,  3450,  7972,  3500,  4095,  1650,
        3450,  6750,  9000,  3069,  4500,  5495,  2398,  3000,  3850,
        3500,  8100,  4960,  2160,  3090,  4500,  3800,  3090,  3240,
        2835,  4600,  5076,  3750,  3630,  8050,  4352,  3000,  5850,
        4960,  3600,  3660,  3480,  2700,  3150,  6615,  3040,  3630,
        6000,  5400,  5200,  3300,  4350,  2640,  2650,  3960,  6800,
```

```
4000,  4000,  3934,  2000, 10890,  2800,  2430,  3480,  4000,
3185,  4000,  2910,  3600,  4400,  7200,  2880,  3180,  3000,
4400,  3000,  3210,  3240,  3000,  3500,  4840,  7700,  3635,
2475,  5574,  3264,  3640,  3180,  1836,  3970,  3970,  1950,
5300,  3000,  2400,  3000,  3360,  3420,  1700,  3649,  2990,
3000,  2400,  3620,  2910,  3850])
```

In [70]:

```
1 type(areaVec)
```

Out[70]:

```
numpy.ndarray
```

In [ ]:

```
1
```