Assignment 1 Data Wrangling I

Importing pandas and numpy libs

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

Reading the dataset and loading into pandas dataframe

In [2]:
```
df=pd.read_csv('melb_data.csv')
```

Displaying the first 5 rows using `head()` function

In [3]:
```
df.head()
```

Out[3]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Po
0	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	
3	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	
4	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	

5 rows × 21 columns

Displaying the sum of null values in each column using `isnull().sum()` function and sorting it decending order

In [4]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[4]:
```
BuildingArea    6450
YearBuilt       5375
CouncilArea     1369
Car               62
Suburb             0
Bathroom           0
Regionname         0
Longtitude         0
Lattitude          0
Landsize           0
Bedroom2           0
Address            0
Postcode           0
Distance           0
Date               0
```

```
SellerG             0
Method              0
Price               0
Type                0
Rooms               0
Propertycount       0
dtype: int64
```

Displaying the statistical parameter related to dataset using `describe()` function

In [5]:
```
df.describe()
```

Out[5]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroon
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	13580.000000
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	1.534242
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	0.691712
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	0.000000
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	1.000000
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	1.000000
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	2.000000
max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	8.000000

Displaying the data type of each column in the data set using `dtypes`

In [6]:
```
df.dtypes
```

Out[6]:
```
Suburb             object
Address            object
Rooms               int64
Type               object
Price             float64
Method             object
SellerG            object
Date               object
Distance          float64
Postcode          float64
Bedroom2          float64
Bathroom          float64
Car               float64
Landsize          float64
BuildingArea      float64
YearBuilt         float64
CouncilArea        object
Lattitude         float64
Longtitude        float64
Regionname         object
Propertycount     float64
dtype: object
```

Displaying the number of rows and columns in the dataset using `shape`

In [7]:
```

```

```
print('Our data set contains {} rows and {} columns'.format(df.shape[0],df.sh
```

```
Our data set contains 13580 rows and 21 columns
```

Displaying the basic info related to all columns in the dataset using `info()` function

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Suburb         13580 non-null  object 
 1   Address        13580 non-null  object 
 2   Rooms          13580 non-null  int64  
 3   Type           13580 non-null  object 
 4   Price          13580 non-null  float64
 5   Method         13580 non-null  object 
 6   SellerG        13580 non-null  object 
 7   Date           13580 non-null  object 
 8   Distance       13580 non-null  float64
 9   Postcode       13580 non-null  float64
 10  Bedroom2       13580 non-null  float64
 11  Bathroom       13580 non-null  float64
 12  Car            13518 non-null  float64
 13  Landsize       13580 non-null  float64
 14  BuildingArea   7130 non-null   float64
 15  YearBuilt      8205 non-null   float64
 16  CouncilArea    12211 non-null  object 
 17  Lattitude      13580 non-null  float64
 18  Longtitude     13580 non-null  float64
 19  Regionname     13580 non-null  object 
 20  Propertycount  13580 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
```

Displaying the Types with value count for each category using `value_counts()` function

In [9]:
```
df.Type.value_counts()
```

Out[9]:
```
h    9449
u    3017
t    1114
Name: Type, dtype: int64
```

Filling all the null values in Car column with mean of Car column using `fillna()` function

In [10]:
```
df['Car'].fillna((df['Car'].mean()),inplace=True)
```

Changing the data type of Price, Postcode, bedroom2, Bathroom, Car, Propertycount columns to int using `astype()` function

In [11]:
```
df['Price']=df['Price'].astype(int)
```

In [12]:
```
df['Postcode']=df['Postcode'].astype(int)
```

In [13]:
```
df['Bedroom2']=df['Bedroom2'].astype(int)
```

In [14]:
```
df['Bathroom']=df['Bathroom'].astype(int)
```

In [15]:
```
df['Car']=df['Car'].astype(int)
```

In [16]:
```
df['Propertycount']=df['Propertycount'].astype(int)
```

In [17]:
```
df.dtypes
```

Out[17]:
```
Suburb           object
Address          object
Rooms             int64
Type             object
Price             int64
Method           object
SellerG          object
Date             object
Distance        float64
Postcode          int64
Bedroom2          int64
Bathroom          int64
Car               int64
Landsize        float64
BuildingArea    float64
YearBuilt       float64
CouncilArea      object
Lattitude       float64
Longtitude      float64
Regionname       object
Propertycount     int64
dtype: object
```

Converting the categorical varialbles in columns Type, regionname, Method to quantitative variable `replace()` function

In [18]:
```
df.Type.unique()
```

Out[18]:
```
array(['h', 'u', 't'], dtype=object)
```

In [19]:
```
df['Type'].replace(['h','u','t'],[1,2,3],inplace=True)
```

In [20]:
```
df.Type.unique()
```

Out[20]:
```
array([1, 2, 3])
```

In [21]:
```
df.Regionname.unique()
```

Out[21]:
```
array(['Northern Metropolitan', 'Western Metropolitan',
```

```
       'Southern Metropolitan', 'Eastern Metropolitan',
       'South-Eastern Metropolitan', 'Eastern Victoria',
       'Northern Victoria', 'Western Victoria'], dtype=object)
```

In [22]:
```
df['Regionname'].replace(['Northern Metropolitan','Western Metropolitan','Sou
```

In [23]:
```
df.Method.unique()
```

Out[23]:
```
array(['S', 'SP', 'PI', 'VB', 'SA'], dtype=object)
```

In [24]:
```
df['Method'].replace(['S','SP','PI','VB','SA'],[1,2,3,4,5],inplace=True)
```

In [25]:
```
df.Method.unique()
```

Out[25]:
```
array([1, 2, 3, 4, 5])
```

In [26]:
```
df.head()
```

Out[26]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Post
0	Abbotsford	85 Turner St	2	1	1480000	1	Biggin	3/12/2016	2.5	
1	Abbotsford	25 Bloomburg St	2	1	1035000	1	Biggin	4/02/2016	2.5	
2	Abbotsford	5 Charles St	3	1	1465000	2	Biggin	4/03/2017	2.5	
3	Abbotsford	40 Federation La	3	1	850000	3	Biggin	4/03/2017	2.5	
4	Abbotsford	55a Park St	4	1	1600000	4	Nelson	4/06/2016	2.5	

5 rows × 21 columns

In [ ]:
```

```