LONDON HOUSING DATASET

The data is primarrily created around the housing market of london. it contain a lot of additionally relevant data.

The data used here from year 1995 to 2019 each different area.

The datset is available as a csv file. Download from Kaggle

We will analyze the data using the pandas data frame.

Import Library

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

Uploading Csv fle

In [2]:
```
df = pd.read_csv("C:\Users\Syed Arif\Downloads\5. London Housing Data.csv")
```

```
  Input In [2]
    df = pd.read_csv("C:\Users\Syed Arif\Downloads\5. London Housing Data.cs
v")

^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in posi
tion 2-3: truncated \UXXXXXXXX escape
```

Remove Unicode error

We are Written Small r before Quatation

In [3]:
```
df = pd.read_csv(r"C:\Users\Syed Arif\Downloads\5. London Housing Data.csv")
```

In [4]:
```
df
```

Out[4]:

	date	area	average_price	code	houses_sold	no_of_crimes
0	1/1/1995	city of london	91449	E09000001	17.0	NaN
1	2/1/1995	city of london	82203	E09000001	7.0	NaN
2	3/1/1995	city of london	79121	E09000001	14.0	NaN
3	4/1/1995	city of london	77101	E09000001	7.0	NaN
4	5/1/1995	city of london	84409	E09000001	10.0	NaN
...	...	...	...	...	...	...
13544	9/1/2019	england	249942	E92000001	64605.0	NaN
13545	10/1/2019	england	249376	E92000001	68677.0	NaN
13546	11/1/2019	england	248515	E92000001	67814.0	NaN
13547	12/1/2019	england	250410	E92000001	NaN	NaN
13548	1/1/2020	england	247355	E92000001	NaN	NaN

13549 rows × 6 columns

Data Preprocessing

.head()

head is used show to the By default = 5 rows in the dataset

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

Out[5]:

	date	area	average_price	code	houses_sold	no_of_crimes
0	1/1/1995	city of london	91449	E09000001	17.0	NaN
1	2/1/1995	city of london	82203	E09000001	7.0	NaN
2	3/1/1995	city of london	79121	E09000001	14.0	NaN
3	4/1/1995	city of london	77101	E09000001	7.0	NaN
4	5/1/1995	city of london	84409	E09000001	10.0	NaN

.tail()

tail is used to show rows by Descending order

In [6]:
```
df.tail()
```

Out[6]:

	date	area	average_price	code	houses_sold	no_of_crimes
13544	9/1/2019	england	249942	E92000001	64605.0	NaN
13545	10/1/2019	england	249376	E92000001	68677.0	NaN
13546	11/1/2019	england	248515	E92000001	67814.0	NaN
13547	12/1/2019	england	250410	E92000001	NaN	NaN
13548	1/1/2020	england	247355	E92000001	NaN	NaN

.shape

It shoe the total no of rows & Column in the dataset

In [7]:
```
df.shape
```

Out[7]:
```
(13549, 6)
```

.Columns

It show the no of each Column

In [8]:
```
df.columns
```

Out[8]:
```
Index(['date', 'area', 'average_price', 'code', 'houses_sold', 'no_of_crime
s'], dtype='object')
```

.dtypes

This Attribute show the data type of each column

In [9]:
```
df.dtypes
```

Out[9]:
```
date              object
area              object
average_price      int64
code              object
houses_sold      float64
no_of_crimes     float64
dtype: object
```

.unique()

In a column, It show the unique value of specific column.

In [10]:
```
df["area"].unique()
```

Out[10]:
```
array(['city of london', 'barking and dagenham', 'barnet', 'bexley',
       'brent', 'bromley', 'camden', 'croydon', 'ealing', 'enfield',
       'tower hamlets', 'greenwich', 'hackney', 'south east',
       'hammersmith and fulham', 'haringey', 'harrow', 'havering',
       'hillingdon', 'hounslow', 'islington', 'kensington and chelsea',
       'kingston upon thames', 'lambeth', 'lewisham', 'merton', 'newham',
       'redbridge', 'richmond upon thames', 'southwark', 'sutton',
       'waltham forest', 'wandsworth', 'westminster', 'inner london',
       'outer london', 'north east', 'north west', 'yorks and the humber',
       'east midlands', 'west midlands', 'east of england', 'london',
       'south west', 'england'], dtype=object)
```

nuique()

It will show the total no of unque value from whole data frame

In [11]:
```
df.nunique()
```

Out[11]:
```
date              301
area               45
average_price    13343
code               45
houses_sold       3946
no_of_crimes      2669
dtype: int64
```

.describe()

It show the Count, mean , median etc

In [12]:
```
df.describe()
```

Out[12]:

	average_price	houses_sold	no_of_crimes
count	1.354900e+04	13455.000000	7439.000000
mean	2.635197e+05	3893.994129	2158.352063
std	1.876175e+05	12114.402476	902.087742
min	4.072200e+04	2.000000	0.000000
25%	1.323800e+05	247.000000	1623.000000
50%	2.229190e+05	371.000000	2132.000000
75%	3.368430e+05	3146.000000	2582.000000
max	1.463378e+06	132163.000000	7461.000000

.value_counts

It Shows all the unique values with their count

In [13]:
```
df["area"].value_counts()
```

Out[13]:
```
hackney                  302
south east               302
enfield                  302
tower hamlets            302
redbridge                301
richmond upon thames     301
southwark                301
sutton                   301
waltham forest           301
wandsworth               301
westminster              301
inner london             301
outer london             301
city of london           301
merton                   301
north east               301
north west               301
yorks and the humber     301
east midlands            301
west midlands            301
east of england          301
london                   301
south west               301
newham                   301
kingston upon thames     301
lewisham                 301
lambeth                  301
barnet                   301
bexley                   301
brent                    301
bromley                  301
camden                   301
croydon                  301
ealing                   301
greenwich                301
hammersmith and fulham   301
haringey                 301
harrow                   301
havering                 301
hillingdon               301
hounslow                 301
islington                301
kensington and chelsea   301
barking and dagenham     301
england                  301
Name: area, dtype: int64
```

In [14]:
```
df.isnull()
```

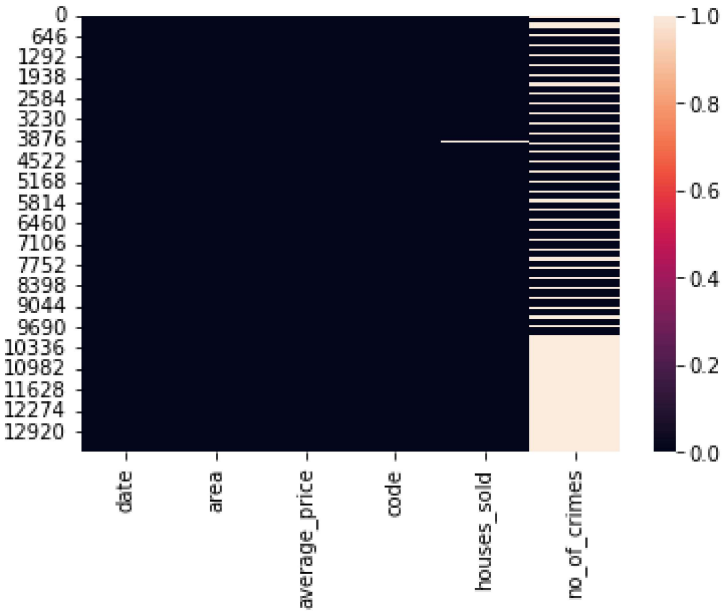
Out[14]:

	date	area	average_price	code	houses_sold	no_of_crimes
0	False	False	False	False	False	True
1	False	False	False	False	False	True
2	False	False	False	False	False	True
3	False	False	False	False	False	True
4	False	False	False	False	False	True
...	...	...	...	...	...	...
13544	False	False	False	False	False	True
13545	False	False	False	False	False	True
13546	False	False	False	False	False	True
13547	False	False	False	False	True	True
13548	False	False	False	False	True	True

13549 rows × 6 columns

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [16]:
```
sns.heatmap(df.isnull())
plt.show()
```



Convert the Datatype of "Date" Column to Date-time-format.

In [17]:
```
df.head()
```

Out[17]:

	date	area	average_price	code	houses_sold	no_of_crimes
0	1/1/1995	city of london	91449	E09000001	17.0	NaN
1	2/1/1995	city of london	82203	E09000001	7.0	NaN
2	3/1/1995	city of london	79121	E09000001	14.0	NaN
3	4/1/1995	city of london	77101	E09000001	7.0	NaN
4	5/1/1995	city of london	84409	E09000001	10.0	NaN

In [18]:
```
df.dtypes
```

Out[18]:
```
date              object
area              object
average_price      int64
code              object
houses_sold      float64
no_of_crimes     float64
dtype: object
```

In [19]:
```
df.date = pd.to_datetime(df.date)
```

In [20]:
```
df.dtypes
```

Out[20]:
```
date             datetime64[ns]
area                     object
average_price             int64
code                     object
houses_sold             float64
no_of_crimes            float64
dtype: object
```

Add a new column "year" in the dataframe , which contains years only.

In [21]:
```
df["Year"] = df.date.dt.year
```

In [22]:
```
df
```

Out[22]:

	date	area	average_price	code	houses_sold	no_of_crimes	Year
0	1995-01-01	city of london	91449	E09000001	17.0	NaN	1995
1	1995-02-01	city of london	82203	E09000001	7.0	NaN	1995
2	1995-03-01	city of london	79121	E09000001	14.0	NaN	1995
3	1995-04-01	city of london	77101	E09000001	7.0	NaN	1995
4	1995-05-01	city of london	84409	E09000001	10.0	NaN	1995
...	...	...	...	...	...	...	...
13544	2019-09-01	england	249942	E92000001	64605.0	NaN	2019
13545	2019-10-01	england	249376	E92000001	68677.0	NaN	2019
13546	2019-11-01	england	248515	E92000001	67814.0	NaN	2019
13547	2019-12-01	england	250410	E92000001	NaN	NaN	2019
13548	2020-01-01	england	247355	E92000001	NaN	NaN	2020

13549 rows × 7 columns

Add a new column "month" in the dataframe , which contains month only

In [23]:
```
df.insert(1, "month" , df.date.dt.month)
```

In [24]:
```
df
```

Out[24]:

	date	month	area	average_price	code	houses_sold	no_of_crimes	Year
0	1995-01-01	1	city of london	91449	E09000001	17.0	NaN	1995
1	1995-02-01	2	city of london	82203	E09000001	7.0	NaN	1995
2	1995-03-01	3	city of london	79121	E09000001	14.0	NaN	1995
3	1995-04-01	4	city of london	77101	E09000001	7.0	NaN	1995
4	1995-05-01	5	city of london	84409	E09000001	10.0	NaN	1995
...	...	...	...	...	...	...	...	...
13544	2019-09-01	9	england	249942	E92000001	64605.0	NaN	2019
13545	2019-10-01	10	england	249376	E92000001	68677.0	NaN	2019
13546	2019-11-01	11	england	248515	E92000001	67814.0	NaN	2019
13547	2019-12-01	12	england	250410	E92000001	NaN	NaN	2019
13548	2020-01-01	1	england	247355	E92000001	NaN	NaN	2020

13549 rows × 8 columns

Remove the column Month & year from the dataframe

In [25]:
```
df.drop(["month" , "Year"], axis=1 , inplace = True)
```

In [26]:
```
df
```

Out[26]:

	date	area	average_price	code	houses_sold	no_of_crimes
0	1995-01-01	city of london	91449	E09000001	17.0	NaN
1	1995-02-01	city of london	82203	E09000001	7.0	NaN
2	1995-03-01	city of london	79121	E09000001	14.0	NaN
3	1995-04-01	city of london	77101	E09000001	7.0	NaN
4	1995-05-01	city of london	84409	E09000001	10.0	NaN
...	...	...	...	...	...	...
13544	2019-09-01	england	249942	E92000001	64605.0	NaN
13545	2019-10-01	england	249376	E92000001	68677.0	NaN
13546	2019-11-01	england	248515	E92000001	67814.0	NaN
13547	2019-12-01	england	250410	E92000001	NaN	NaN
13548	2020-01-01	england	247355	E92000001	NaN	NaN

13549 rows × 6 columns

Show all the records where No of crime is 0 , and how man such records are there

In [27]:
```
len(df[df.no_of_crimes == 0])
```

Out[27]:
```
104
```

In [28]:
```
df[df.no_of_crimes == 0]
```

Out[28]:

	date	area	average_price	code	houses_sold	no_of_crimes
72	2001-01-01	city of london	284262	E09000001	24.0	0.0
73	2001-02-01	city of london	198137	E09000001	37.0	0.0
74	2001-03-01	city of london	189033	E09000001	44.0	0.0
75	2001-04-01	city of london	205494	E09000001	38.0	0.0
76	2001-05-01	city of london	223459	E09000001	30.0	0.0
...	...	...	...	...	...	...
178	2009-11-01	city of london	397909	E09000001	11.0	0.0
179	2009-12-01	city of london	411955	E09000001	16.0	0.0
180	2010-01-01	city of london	464436	E09000001	20.0	0.0
181	2010-02-01	city of london	490525	E09000001	9.0	0.0
182	2010-03-01	city of london	498241	E09000001	15.0	0.0

104 rows × 6 columns

What is maximum & minimum "average_price" per year in england

In [29]:
```
df = df[df.area == "england"]
```

In [38]:
```
df
```

Out[38]:

	date	area	average_price	code	houses_sold	no_of_crimes	Year
0	1995-01-01	city of london	91449	E09000001	17.0	NaN	1995
1	1995-02-01	city of london	82203	E09000001	7.0	NaN	1995
2	1995-03-01	city of london	79121	E09000001	14.0	NaN	1995
3	1995-04-01	city of london	77101	E09000001	7.0	NaN	1995
4	1995-05-01	city of london	84409	E09000001	10.0	NaN	1995
...	...	...	...	...	...	...	...
13544	2019-09-01	england	249942	E92000001	64605.0	NaN	2019
13545	2019-10-01	england	249376	E92000001	68677.0	NaN	2019
13546	2019-11-01	england	248515	E92000001	67814.0	NaN	2019
13547	2019-12-01	england	250410	E92000001	NaN	NaN	2019
13548	2020-01-01	england	247355	E92000001	NaN	NaN	2020

13549 rows × 7 columns

In [39]:
```
df["Year"] = df.date.dt.year
df
```

Out[39]:

	date	area	average_price	code	houses_sold	no_of_crimes	Year
0	1995-01-01	city of london	91449	E09000001	17.0	NaN	1995
1	1995-02-01	city of london	82203	E09000001	7.0	NaN	1995
2	1995-03-01	city of london	79121	E09000001	14.0	NaN	1995
3	1995-04-01	city of london	77101	E09000001	7.0	NaN	1995
4	1995-05-01	city of london	84409	E09000001	10.0	NaN	1995
...	...	...	...	...	...	...	...
13544	2019-09-01	england	249942	E92000001	64605.0	NaN	2019
13545	2019-10-01	england	249376	E92000001	68677.0	NaN	2019
13546	2019-11-01	england	248515	E92000001	67814.0	NaN	2019
13547	2019-12-01	england	250410	E92000001	NaN	NaN	2019
13548	2020-01-01	england	247355	E92000001	NaN	NaN	2020

13549 rows × 7 columns

In [40]:
```
df.groupby("Year").average_price.max()
```

Out[40]:
```
Year
1995     200722
1996     223197
1997     265112
1998     277600
1999     354241
2000     397353
2001     451028
2002     497538
2003     488704
2004     559286
2005     555847
2006     644541
2007     830950
2008     832753
2009     782459
2010     884674
2011     959520
2012    1077366
2013    1217729
2014    1365050
2015    1353679
2016    1357231
2017    1412255
2018    1463378
2019    1294113
2020    1178166
Name: average_price, dtype: int64
```

In [41]:
```
df.groupby("Year").average_price.min()
```

Out[41]:
```
Year
1995     41688
1996     40722
1997     42353
1998     43510
1999     43969
2000     47604
2001     49045
2002     54746
2003     67520
2004     88520
2005    110454
2006    121124
2007    131175
2008    120275
2009    117079
2010    119688
2011    115328
2012    113011
2013    112008
2014    114531
2015    117156
2016    121085
2017    121858
2018    124038
2019    124567
2020    126592
Name: average_price, dtype: int64
```

In [ ]:

In [ ]: