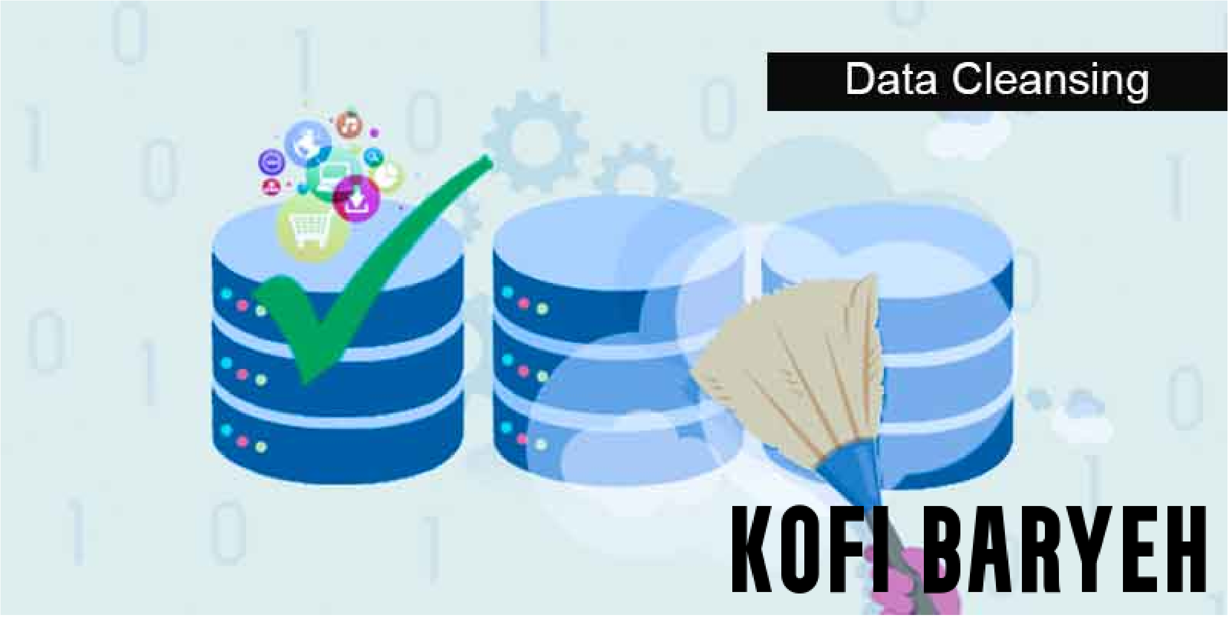
In [85]:
```
import pandas as pd
import numpy as np
from PIL import Image
```

In [86]:
```
img = Image.open('data-cleansing.png')    # Open image as PIL image object
img
```

Out[86]:



ABOUT THE DATASET

The dataset was downloaded form a random github account. The aim of this project it to use various methods available through python Pandas to clean up the data to a state where it is fit for Analysis and Visualization. The visualization aspect will be a separate project. This project is dedicated soley to Data Cleaning

READING HOUSING CSV DATASET INTO JUPYTER

In [87]:
```
df = pd.read_csv('nash_housing_data.csv')
```

DISPLAYING FIRST FIVE ROWS

In [88]:
```
df.head(5)
```

Out[88]:

	UniqueID	ParcelID	LandUse	PropertyAddress	SaleDate	SalePrice	LegalReference	SoldAsVa
0	2045	007 00 0 125.00	SINGLE FAMILY	1808 FOX CHASE DR, GOODLETTSVILLE	09-Apr-13	240000	20130412-0036474	
1	16918	007 00 0 130.00	SINGLE FAMILY	1832 FOX CHASE DR, GOODLETTSVILLE	10-Jun-14	366000	20140619-0053768	
2	54582	007 00 0 138.00	SINGLE FAMILY	1864 FOX CHASE DR, GOODLETTSVILLE	26-Sep-16	435000	20160927-0101718	
3	43070	007 00 0 143.00	SINGLE FAMILY	1853 FOX CHASE DR, GOODLETTSVILLE	29-Jan-16	255000	20160129-0008913	
4	22714	007 00 0 149.00	SINGLE FAMILY	1829 FOX CHASE DR, GOODLETTSVILLE	10-Oct-14	278000	20141015-0095255	

UNDERSTANDING FEATURES OF THE DATASET

In [89]:
```
df.shape
```

Out[89]:
```
(56477, 19)
```

In [90]:
```
df.describe()
```

Out[90]:

	UniqueID	Acreage	LandValue	BuildingValue	TotalValue	YearBuilt	Be
count	56477.000000	26015.000000	2.601500e+04	2.601500e+04	2.601500e+04	24163.000000	24157
mean	28334.001133	0.498923	6.906856e+04	1.607847e+05	2.323754e+05	1963.744899	3
std	16352.590651	1.570454	1.060401e+05	2.067999e+05	2.810643e+05	26.542982	0
min	0.000000	0.010000	1.000000e+02	0.000000e+00	1.000000e+02	1799.000000	0
25%	14186.000000	0.180000	2.100000e+04	7.590000e+04	1.028000e+05	1948.000000	3
50%	28313.000000	0.270000	2.880000e+04	1.114000e+05	1.485000e+05	1960.000000	3
75%	42513.000000	0.450000	6.000000e+04	1.807000e+05	2.683500e+05	1983.000000	3
max	56635.000000	160.060000	2.772000e+06	1.297180e+07	1.394040e+07	2017.000000	1

In [91]:
```
# From the counts we can see that some columns have empty cells
```

In [92]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56477 entries, 0 to 56476
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   UniqueID        56477 non-null  int64  
 1   ParcelID        56477 non-null  object 
 2   LandUse         56477 non-null  object 
 3   PropertyAddress  56448 non-null  object 
 4   SaleDate        56477 non-null  object 
 5   SalePrice       56477 non-null  object 
 6   LegalReference  56477 non-null  object 
 7   SoldAsVacant    56477 non-null  object 
 8   OwnerName       25261 non-null  object 
 9   OwnerAddress    26015 non-null  object 
 10  Acreage         26015 non-null  float64
 11  TaxDistrict     26015 non-null  object 
 12  LandValue       26015 non-null  float64
 13  BuildingValue   26015 non-null  float64
 14  TotalValue      26015 non-null  float64
 15  YearBuilt       24163 non-null  float64
 16  Bedrooms        24157 non-null  float64
 17  FullBath        24275 non-null  float64
 18  HalfBath        24144 non-null  float64
dtypes: float64(8), int64(1), object(10)
memory usage: 8.2+ MB
```

In [93]:
```
df.isnull().sum()
```

Out[93]:
```
UniqueID              0
ParcelID              0
LandUse               0
PropertyAddress      29
SaleDate              0
SalePrice             0
LegalReference        0
SoldAsVacant          0
OwnerName         31216
OwnerAddress      30462
Acreage           30462
TaxDistrict       30462
LandValue         30462
BuildingValue     30462
TotalValue        30462
YearBuilt         32314
Bedrooms          32320
FullBath          32202
HalfBath          32333
dtype: int64
```

In [94]:
```
# We can delee rows with null values with this code <df_drop=df.dropna()> but fr
# all nulls in place. I prefer leaving nulls in the dataset and ommitting them wh
# where the nulls will affect the outcome of the results
```

DROPPING DUPLICATE ROWS IF THEY EXIST

In [95]:

```
# let's drop duplicate rows
df=df.drop_duplicates(keep='last')
```

CHECKING IF THERE WERE DUPLICATES We do this by re-checking the shape

In [96]:

```
df.shape
```

Out[96]:

```
(56477, 19)
```

In [97]:

```
# The shape is still the same so there are no duplicates
```

CHANGING DATE SaleDate COLUMN IN TO DATETIME SO PANDAS RECOGNIZES IT AS A DATE COLUMN

In [98]:

```
df['SaleDate']=pd.to_datetime(df['SaleDate'])
```

CHECKING

In [99]:

```
df.head(1)
```

Out[99]:

	UniqueID	ParcelID	LandUse	PropertyAddress	SaleDate	SalePrice	LegalReference	SoldAsVa
0	2045	007 00 0 125.00	SINGLE FAMILY	1808 FOX CHASE DR, GOODLETTSVILLE	2013-04-09	240000	20130412-0036474	

In [100]:

```
#Splitting up the SaleDate colmun by the comma in the string
new=df["PropertyAddress"].str.split(",",n=1,expand=True)
```

ASSIGNING NAMES TO SPLIT PARTS

In [101]:

```
df["Property St. Name"]=new[0]
```

In [102]:

```
df["Property City"]=new[1]
```

NOW LET'S DROPP THE SaleDate COLUMN

In [103]:

```
df.drop(columns=["PropertyAddress"],inplace=True)
```

CHECKING NEW COLUMNS

```
In [104]: df.columns
```

```
Out[104]: Index(['UniqueID ', 'ParcelID', 'LandUse', 'SaleDate', 'SalePrice',
                 'LegalReference', 'SoldAsVacant', 'OwnerName', 'OwnerAddress',
                 'Acreage', 'TaxDistrict', 'LandValue', 'BuildingValue', 'TotalValue',
                 'YearBuilt', 'Bedrooms', 'FullBath', 'HalfBath', 'Property St. Name',
                 'Property City'],
                dtype='object')
```

PRINTING SOME ROWS TO CHECK IF THE COMMA THAT WAS IN THE ADDRESS HAS BEEN REMOVED

```
In [105]: df.head(5)
```

Out[105]:

	UniqueID	ParcelID	LandUse	SaleDate	SalePrice	LegalReference	SoldAsVacant	OwnerNam
0	2045	007 00 0 125.00	SINGLE FAMILY	2013-04-09	240000	20130412-0036474	No	FRAZIEF CYRENTH. LYNETT
1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	BONEF CHARLES LESLI
2	54582	007 00 0 138.00	SINGLE FAMILY	2016-09-26	435000	20160927-0101718	No	WILSON JAMES E. JOANN
3	43070	007 00 0 143.00	SINGLE FAMILY	2016-01-29	255000	20160129-0008913	No	BAKER, JAY K & SUSAN E
4	22714	007 00 0 149.00	SINGLE FAMILY	2014-10-10	278000	20141015-0095255	No	POST CHRISTOPHEI M. SAMANTHA C

```
In [106]: # It has been removed
```

LET'S USE SAME METHOD TO SPLIT UP THE OwnerAddress

```
In [107]: new1=df["OwnerAddress"].str.split(",",n=1,expand=True)
```

```
In [108]: df["Qwner_Street Name"]=new1[0]
```

```
In [109]: df["Owner_City"]=new1[1]
```

In [110]:
```
df["Owner_State"]=new1[2]
```

```
    356                 except ValueError as err:

ValueError: 2 is not in range

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

KeyError                                  Traceback (most recent call last)
<ipython-input-110-63183eb32529> in <module>
----> 1 df["Owner_State"]=new1[2]

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2900             if self.columns.nlevels > 1:
   2901                 return self._getitem_multilevel(key)
-> 2902             indexer = self.columns.get_loc(key)
   2903             if is_integer(indexer):
   2904                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\range.py in get_loc(self, k
ey, method, tolerance)
    355                     return self._range.index(new_key)
```

In [111]:
```
# I am getting an error because I have to split the second part. The syntax I use
```

In [112]:
```
df.head(5)
```

BONER, CHARLES & LESLIE	1832 FOX CHASE DR, GOODLETTSVILLE, TN	3.5	...	264100.0	319000.0	1998.0	3.0	3.0	2.0
WILSON, JAMES E. & JOANNE	1864 FOX CHASE DR, GOODLETTSVILLE, TN	2.9	...	216200.0	298000.0	1987.0	4.0	3.0	0.0
KER, JAY K. & SUSAN E.	1853 FOX CHASE DR, GOODLETTSVILLE, TN	2.6	...	147300.0	197300.0	1985.0	3.0	3.0	0.0
POST, RISTOPHER M. & MANTHA C.	1829 FOX CHASE DR, GOODLETTSVILLE, TN	2.0	...	152300.0	202300.0	1984.0	4.0	3.0	0.0

SPLITTING Owner_City Column

In [113]:
```
new2=df["Owner_City"].str.split(",",n=1,expand=True)
```

In [114]:
```
df["City"]=new2[0]
```

In [115]:

```
df["State"]=new2[1]
```

In [116]:

```
# Now let's drop Owner_City column and OwnerAddress
```

In [117]:

```
df.drop(columns=["Owner_City"],inplace=True)
```

In [118]:

```
df.drop(columns=["OwnerAddress"],inplace=True)
```

In [119]:

```
df.head(2)
```

Out[119]:

	UniqueID	ParcelID	LandUse	SaleDate	SalePrice	LegalReference	SoldAsVacant	OwnerName
0	2045	007 00 0 125.00	SINGLE FAMILY	2013-04-09	240000	20130412-0036474	No	FRAZIER, CYRENTHA LYNETTE
1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	BONER, CHARLES & LESLIE

2 rows × 22 columns

In [120]:

```
# Lets rename City and State columns to 'Owner_City' and 'Owner_State'
```

In [121]:

```
# Rename columns and assign to a dataframe
df2 = df.rename(columns={'City':'Owner_City', 'State':'Owner_State'})
```

SPLITTING UP THE OwnerName Column

In [122]:

```
new3=df2["OwnerName"].str.split(",",n=1,expand=True)
```

In [123]:

```
df2["Owner's L Name"]=new3[0]
```

In [124]:

```
df2["Owner's F&M Name"]=new3[1]
```

In [125]:

```
# Dropping 'OwnerName'
df2.drop(columns=["OwnerName"],inplace=True)
```

In [126]:
```
df2.head(5)
```

Out[126]:

TaxDistrict	LandValue	...	Bedrooms	FullBath	HalfBath	Property St. Name	Property City	Qwner_Stre Nam
GENERAL SERVICES DISTRICT	50000.0	...	3.0	3.0	0.0	1808 FOX CHASE DR	GOODLETTSVILLE	1808 FC CHASE D
GENERAL SERVICES DISTRICT	50000.0	...	3.0	3.0	2.0	1832 FOX CHASE DR	GOODLETTSVILLE	1832 FC CHASE D
GENERAL SERVICES DISTRICT	50000.0	...	4.0	3.0	0.0	1864 FOX CHASE DR	GOODLETTSVILLE	1864 FC CHASE D
GENERAL SERVICES DISTRICT	50000.0	...	3.0	3.0	0.0	1853 FOX CHASE DR	GOODLETTSVILLE	1853 FC CHASE D
GENERAL SERVICES DISTRICT	50000.0	...	4.0	3.0	0.0	1829 FOX CHASE DR	GOODLETTSVILLE	1829 FC CHASE D

In [127]:
```
# Listing unique values in SoldAsVacant column
print(df2['SoldAsVacant'].unique())
```

```
['No' 'N' 'Yes' 'Y']
```

In [73]:
```
# So let's convert 'N' to 'No' and 'Y' to 'Yes'
```

In [128]:
```
df2[['SoldAsVacant']]=df2[['SoldAsVacant']].replace('N','No')
```

In [129]:
```
df2[['SoldAsVacant']]=df2[['SoldAsVacant']].replace('Y','Yes')
```

LET'S RE CHECK UNIQUE VALUES IN SoldAsVacant

In [130]:
```
print(df2['SoldAsVacant'].unique())
```

```
['No' 'Yes']
```

YAY....IT WORKED!!!!

There are a lot of things we can do to the data but for the purpose of this project we will end here since what we have done is to introduce some basic methods that can be employed to clean up data. SEE YOU IN MY NEXT PROJECT

Now let us save the final file as a new csv file

```
In [131]: df2.to_csv('nash_housing_data_cleaned.csv',index=False)
```

THE END THE END THE END THE END THE END THE END THE END THE END THE END THE END THE END THE END THE END THE END THE END

READING CLEANED FILE

```
In [132]: df2.head(5)
```

Out[132]:

	UniqueID	ParcelID	LandUse	SaleDate	SalePrice	LegalReference	SoldAsVacant	Acreage	Tax
0	2045	007 00 0 125.00	SINGLE FAMILY	2013-04-09	240000	20130412-0036474	No	2.3	GE SE DI
1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	3.5	GE SE DI
2	54582	007 00 0 138.00	SINGLE FAMILY	2016-09-26	435000	20160927-0101718	No	2.9	GE SE DI
3	43070	007 00 0 143.00	SINGLE FAMILY	2016-01-29	255000	20160129-0008913	No	2.6	GE SE DI
4	22714	007 00 0 149.00	SINGLE FAMILY	2014-10-10	278000	20141015-0095255	No	2.0	GE SE DI

5 rows × 23 columns

```
In [ ]:
```