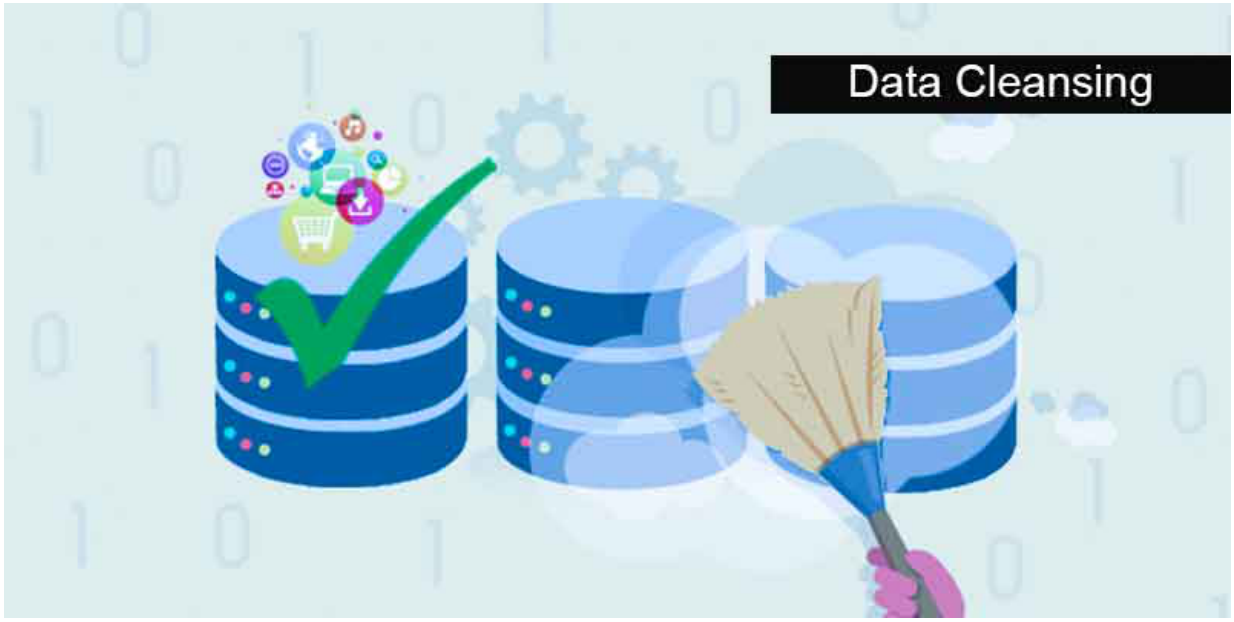
In [2]:
```
import pandas as pd
import numpy as np
from PIL import Image
```

In [5]:
```
img = Image.open('data-cleansing.jpg')    # Open image as PIL image object
img
```

Out[5]:



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 [126]:
```
df = pd.read_csv('nash_housing_data.csv')
```

DISPLAYING FIRST FIVE ROWS

In [127]:
```
df.head(5)
```

Out[127]:

	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 [128]:
```
df.shape
```

Out[128]:
```
(56477, 19)
```

In [129]:
```
df.describe()
```

Out[129]:

	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 [130]:
```
# From the counts we can see that some columns have empty cells
```

In [131]:
```
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 [132]:
```
df.isnull().sum()
```

Out[132]:
```
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 [133]:
```
# 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 [134]:
```
# 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 [135]:
```
df.shape
```

Out[135]:
```
(56477, 19)
```

In [136]:
```
# 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 [137]:
```
df['SaleDate']=pd.to_datetime(df['SaleDate'])
```

CHECKING

In [138]:
```
df.head(1)
```

Out[138]:

	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 [139]:
```
#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 [140]:
```
df["Street Name"]=new[0]
```

In [141]:
```
df["City"]=new[1]
```

NOW LET'S DROPP THE SaleDate COLUMN

In [142]:
```
df.drop(columns=["PropertyAddress"],inplace=True)
```

CHECKING NEW COLUMNS

In [143]:
```
df.columns
```

Out[143]:
```
Index(['UniqueID ', 'ParcelID', 'LandUse', 'SaleDate', 'SalePrice',
       'LegalReference', 'SoldAsVacant', 'OwnerName', 'OwnerAddress',
       'Acreage', 'TaxDistrict', 'LandValue', 'BuildingValue', 'TotalValue',
       'YearBuilt', 'Bedrooms', 'FullBath', 'HalfBath', 'Street Name', 'City'],
      dtype='object')
```

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

In [144]:
```
df.head(5)
```

Out[144]:

	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 CHRISTOPHE M. SAMANTHA C

In [145]:
```
# It has been removed
```

LET'S USE SAME METHOD TO SPLIT UP THE OwnerAddress

In [146]:
```
new1=df["OwnerAddress"].str.split(",",n=1,expand=True)
```

In [147]:
```
df["Qwner_Street Name"]=new1[0]
```

In [148]:
```
df["Owner_City"]=new1[1]
```

In [149]:
```
df["Owner_State"]=new1[2]
```

```
KeyError                                  Traceback (most recent call last)
<ipython-input-149-63183eb32529> 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)
    356                 except ValueError as err:
--> 357                     raise KeyError(key) from err
    358             raise KeyError(key)
    359         return super().get_loc(key, method=method, tolerance=toleranc
e)
```

In [150]:
```
# I am getting an error because I have to split the second part. The syntax I use
```

In [151]:
```
df.head(5)
```

1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	BO CHARL LE
2	54582	007 00 0 138.00	SINGLE FAMILY	2016-09-26	435000	20160927-0101718	No	WIL JAMES JOA
3	43070	007 00 0 143.00	SINGLE FAMILY	2016-01-29	255000	20160129-0008913	No	BAKER, J & SUS
4	22714	007 00 0 149.00	SINGLE FAMILY	2014-10-10	278000	20141015-0095255	No	P CHRISTOF SAMANTH

SPLITTING Owner_City Column

In [152]:
```
new3=df["Owner_City"].str.split(",",n=1,expand=True)
```

In [153]:
```
df["City"]=new1[0]
```

In [154]:
```
df["State"]=new1[1]
```

In [155]:
```
# Now let's drop Owner_City column and OwnerAddress
```

In [156]:
```
df.drop(columns=["Owner_City"],inplace=True)
```

In [157]:
```
df.drop(columns=["OwnerAddress"],inplace=True)
```

In [158]:
```
df.head(2)
```

Out[158]:

	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 × 21 columns

In [159]:
```
# Lets rename City and State columns to 'Owner_City' and 'Owner_State'
```

In [160]:
```
df.rename(columns={'City':'Owner_City', 'State':'Owner_State'})
```

Out[160]:

	UniqueID	ParcelID	LandUse	SaleDate	SalePrice	LegalReference	SoldAsVacant	O
0	2045	007 00 0 125.00	SINGLE FAMILY	2013-04-09	240000	20130412-0036474	No	(
1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	(
2	54582	007 00 0 138.00	SINGLE FAMILY	2016-09-26	435000	20160927-0101718	No	J
3	43070	007 00 0 143.00	SINGLE FAMILY	2016-01-29	255000	20160129-0008913	No	BAK 8
4	22714	007 00 0 149.00	SINGLE FAMILY	2014-10-10	278000	20141015-0095255	No	CHR SAM
...	...	...	...	...	...	...	...	
56472	30469	188 10 0A 101.00	SINGLE FAMILY	2015-05-27	157500	20150608-0053286	No	
56473	27707	188 10 0A 107.00	SINGLE FAMILY	2015-03-02	145000	20150304-0019013	No	
56474	52709	188 10 0A 118.00	VACANT RESIDENTIAL LAND	2016-08-16	234611	20160819-0087214	Yes	
56475	54042	188 10 0A 121.00	VACANT RESIDENTIAL LAND	2016-09-07	93844	20160919-0098411	Yes	
56476	54043	188 10 0A 122.00	VACANT RESIDENTIAL LAND	2016-09-07	93844	20160919-0098411	Yes	

56477 rows × 21 columns

SPLITTING UP THE OwnerName Column

In [161]:
```
new4=df["OwnerName"].str.split(",",n=1,expand=True)
```

In [162]:
```
df["Owner's L Name"]=new4[0]
```

In [163]:
```
df["Owner's F&M Name"]=new4[1]
```

In [164]:
```
# Dropping 'OwnerName'
df.drop(columns=["OwnerName"],inplace=True)
```

In [165]:
```
df.head(5)
```

Out[165]:

	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 SEF DI
1	16918	007 00 0 130.00	SINGLE FAMILY	2014-06-10	366000	20140619-0053768	No	3.5	GE SEF DI
2	54582	007 00 0 138.00	SINGLE FAMILY	2016-09-26	435000	20160927-0101718	No	2.9	GE SEF DI
3	43070	007 00 0 143.00	SINGLE FAMILY	2016-01-29	255000	20160129-0008913	No	2.6	GE SEF DI
4	22714	007 00 0 149.00	SINGLE FAMILY	2014-10-10	278000	20141015-0095255	No	2.0	GE SEF DI

5 rows × 22 columns

In [166]:
```
# Listing unique values in SoldAsVacant column
print(df['SoldAsVacant'].unique())
```

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

In [167]:
```
# So let's convert 'N' to 'No' and 'Y' to 'Yes'
```

In [168]:
```
df[['SoldAsVacant']]=df[['SoldAsVacant']].replace('N','No')
```

In [169]:
```
df[['SoldAsVacant']]=df[['SoldAsVacant']].replace('Y','Yes')
```

LET'S RE CHECK UNIQUE VALUES IN SoldAsVacant

In [170]:
```
print(df['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 [172]:
```
df.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 THE END

In [ ]:
```

```