Preparing Mexico Data

In [1]:
```
import pandas as pd
from IPython.display import VimeoVideo
```

Import

The first part of any data science project is preparing your data, which means making sure its in the right place and format for you to conduct your analysis. The first step of any data preparation is importing your raw data and cleaning it.

If you look in the `small-data` directory on your machine, you'll see that the data for this project comes in three CSV files: `mexico-real-estate-1.csv`, `mexico-real-estate-2.csv`, and `mexico-real-estate-3.csv`.

In [2]:
```
VimeoVideo("656321516", h="e85e3bf248", width=600)
```

Out[2]:

Task 1.2.1: Read these three files into three separate DataFrames named `df1`, `df2`, and `df3`, respectively.

- What's a DataFrame?
- What's a CSV file?
- Read a CSV file into a DataFrame using pandas.

In [3]:
```
df1 = pd.read_csv("data/mexico-real-estate-1.csv")
df2 = pd.read_csv("data/mexico-real-estate-2.csv")
df3 = pd.read_csv("data/mexico-real-estate-3.csv")
```

Clean `df1`

Now that you have your three DataFrames, it's time to inspect them to see if they need any cleaning. Let's look at them one-by-one.

In [4]:
```
VimeoVideo("656320563", h="a6841fed28", width=600)
```

Out[4]:

Task 1.2.2: Inspect `df1` by looking at its `shape` attribute. Then use the `info` method to see the data types and number of missing values for each column. Finally, use the `head` method to determine to look at the first five rows of your dataset.

- Inspect a DataFrame using the `shape`, `info`, and `head` in pandas.

In [5]:
```
df1.shape
```

Out[5]:
```
(700, 6)
```

In [6]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   property_type  700 non-null    object 
 1   state          700 non-null    object 
 2   lat            583 non-null    float64
 3   lon            583 non-null    float64
 4   area_m2        700 non-null    float64
 5   price_usd      700 non-null    object 
dtypes: float64(3), object(3)
memory usage: 32.9+ KB
```

It looks like there are a couple of problems in this DataFrame that you need to solve. First, there are many rows with `NaN` values in the `"lat"` and `"lon"` columns. Second, the data type for the

"price_usd" column is object when it should be float .

In [7]:
```
df1.head()
```

Out[7]:

	property_type	state	lat	lon	area_m2	price_usd
0	house	Estado de México	19.560181	-99.233528	150.0	$67,965.56
1	house	Nuevo León	25.688436	-100.198807	186.0	$63,223.78
2	apartment	Guerrero	16.767704	-99.764383	82.0	$84,298.37
3	apartment	Guerrero	16.829782	-99.911012	150.0	$94,308.80
4	house	Veracruz de Ignacio de la Llave	NaN	NaN	175.0	$94,835.67

In [8]:
```
VimeoVideo("656316512", h="33eb5cb26e", width=600)
```

Out[8]:

In [9]:
```
#Remove 'NAN' values

df1.dropna(inplace = True)
#if don't use ""inplace"" that time generate a new data frame.
## "" inplace"" means i want to change in the dataframe itself.
```

In [10]:
```
df1["price_usd"] = (df1["price_usd"]
                    .str.replace("$", "",regex=True)
                   .str.replace(",","")
                   .astype("float"))
```

In [11]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 583 entries, 0 to 699
Data columns (total 6 columns):
```

```
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   property_type  583 non-null    object
 1   state          583 non-null    object
 2   lat            583 non-null    float64
 3   lon            583 non-null    float64
 4   area_m2        583 non-null    float64
 5   price_usd      583 non-null    float64
dtypes: float64(4), object(2)
memory usage: 31.9+ KB
```

Task 1.2.3: Clean `df1` by dropping rows with `NaN` values. Then remove the `"$"` and `","` characters from `"price_usd"` and recast the values in the column as floats.

- What's a data type?
- Drop rows with missing values from a DataFrame using pandas.
- Replace string characters in a column using pandas.
- Recast a column as a different data type in pandas.

In [12]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 583 entries, 0 to 699
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   property_type  583 non-null    object
 1   state          583 non-null    object
 2   lat            583 non-null    float64
 3   lon            583 non-null    float64
 4   area_m2        583 non-null    float64
 5   price_usd      583 non-null    float64
dtypes: float64(4), object(2)
memory usage: 31.9+ KB
```

Clean `df2`

Now it's time to tackle `df2`. Take a moment to inspect it using the same commands you used before. You'll notice that it has the same issue of `NaN` values, but there's a new problem, too: The home prices are in Mexican pesos (`"price_mxn"`), not US dollars (`"price_usd"`). If we want to compare all the home prices in this dataset, they all need to be in the same currency.

In [13]:
```
VimeoVideo("656315668", h="c9bd116aca", width=600)
```

Out[13]:

Task 1.2.4: First, drop rows with `NaN` values in `df2`. Next, use the `"price_mxn"` column to create a new column named `"price_usd"`. (Keep in mind that, when this data was collected in 2014, a dollar cost 19 pesos.) Finally, drop the `"price_mxn"` from the DataFrame.

- Drop rows with missing values from a DataFrame using pandas.
- Create new columns derived from existing columns in a DataFrame using pandas.
- Drop a column from a DataFrame using pandas.

In [14]:
```
#Remove 'NAN' values
df2.dropna(inplace = True)
df2["price_usd"] = (df2["price_mxn"] / 19)
#Drop "price_mxn" columns
df2.drop(columns = ["price_mxn"], inplace = True)
df2.head()
```

Out[14]:

	property_type	state	lat	lon	area_m2	price_usd
0	apartment	Nuevo León	25.721081	-100.345581	72.0	68421.052632
2	house	Morelos	23.634501	-102.552788	360.0	278947.368421
6	apartment	Estado de México	19.272040	-99.572013	85.0	65789.473684
7	house	San Luis Potosí	22.138882	-100.996510	158.0	111578.947368
8	apartment	Distrito Federal	19.394558	-99.129707	65.0	39904.736842

Clean `df3`

Great work! We're now on the final DataFrame. Use the same `shape`, `info` and `head` commands to inspect the `df3`. Do you see any familiar issues?

You'll notice that we still have `NaN` values, but there are two new problems:

1. Instead of separate `"lat"` and `"lon"` columns, there's a single `"lat-lon"` column.

2. Instead of a `"state"` column, there's a `"place_with_parent_names"` column.

We need the resolve these problems so that `df3` has the same columns in the same format as `df1` and `df2`.

In [15]:
```
VimeoVideo("656314718", h="8d1127a93f", width=600)
```

Out[15]:

In [16]:
```
df3["lat-lon"].str.split(",")
```

Out[16]:
```
0                [19.52589, -99.151703]
1             [19.2640539, -99.5727534]
2               [19.268629, -99.671722]
3                                   NaN
4               [19.511938, -96.871956]
                     ...
695            [20.532264, -103.484418]
696           [18.9289862, -99.1802147]
697    [21.0284038368, -89.6530058049]
698     [22.11830417, -101.0321938992]
699             [19.233201, -99.558519]
Name: lat-lon, Length: 700, dtype: object
```

In [17]:
```
df3["lat-lon"].str.split(",", expand = True) #"expand" separete the list
```

Out[17]:

	0	1
0	19.52589	-99.151703
1	19.2640539	-99.5727534
2	19.268629	-99.671722
3	NaN	NaN
4	19.511938	-96.871956
...	...	...

	0	1
695	20.532264	-103.484418
696	18.9289862	-99.1802147
697	21.0284038368	-89.6530058049
698	22.11830417	-101.0321938992
699	19.233201	-99.558519

700 rows × 2 columns

Task 1.2.5: Drop rows with `NaN` values in `df3`. Then use the `split` method to create two new columns from `"lat-lon"` named `"lat"` and `"lon"`, respectively.

- Drop rows with missing values from a DataFrame using pandas.
- Split the strings in one column to create another using pandas.

In [19]:
```
#Remove '"NAN"' values

df3.dropna(inplace = True)

# Split "lat-lon" between two columns

df3[['lat', 'lon']] = df3["lat-lon"].str.split(",", expand = True) #"expand" separete t
df3.head()
```

Out[19]:

	property_type	place_with_parent_names	lat-lon	area_m2	price_usd	lat	
0	apartment	|México|Distrito Federal|Gustavo A. Madero|Acu...	19.52589,-99.151703	71.0	48550.59	19.52589	-9
1	house	|México|Estado de México|Toluca|Metepec|	19.2640539,-99.5727534	233.0	168636.73	19.2640539	-99
2	house	|México|Estado de México|Toluca|Toluca de Lerd...	19.268629,-99.671722	300.0	86932.69	19.268629	-9
4	apartment	|México|Veracruz de Ignacio de la Llave|Veracruz|	19.511938,-96.871956	84.0	68508.67	19.511938	-9
5	house	|México|Jalisco|Guadalajara|	20.689157,-103.366728	175.0	102763.00	20.689157	-10

In [20]:
```
VimeoVideo("656314050", h="13f6a677fd", width=600)
```

Out[20]:

Task 1.2.6: Use the `split` method again, this time to extract the state for every house. (Note that the state name always appears after `"México|"` in each string.) Use this information to create a `"state"` column. Finally, drop the `"place_with_parent_names"` and `"lat-lon"` columns from the DataFrame.

- Split the strings in one column to create another using pandas.
- Drop a column from a DataFrame using pandas.

In [21]:
```
df3["place_with_parent_names"].head()
```

Out[21]:
```
0    |México|Distrito Federal|Gustavo A. Madero|Acu...
1             |México|Estado de México|Toluca|Metepec|
2    |México|Estado de México|Toluca|Toluca de Lerd...
4    |México|Veracruz de Ignacio de la Llave|Veracruz|
5                         |México|Jalisco|Guadalajara|
Name: place_with_parent_names, dtype: object
```

In [22]:
```
df3["place_with_parent_names"].str.split("|", expand = True)
```

Out[22]:

	0	1	2	3	4	5	6
0	México	Distrito Federal	Gustavo A. Madero	Acueducto de Guadalupe			None
1	México	Estado de México	Toluca	Metepec			None
2	México	Estado de México	Toluca	Toluca de Lerdo		Ocho Cedros	
4	México	Veracruz de Ignacio de la Llave	Veracruz			None	None
5	México	Jalisco	Guadalajara			None	None
...	...	...	...	...	...	...	...
695	México		Jalisco	Tlajomulco de Zúñiga	Tlajomulco de Zúñiga		None

	0	1	2	3	4	5	6
696	México	Morelos	Jiutepec			None	None
697	México	Yucatán	Mérida			None	None
698	México	San Luis Potosí	San Luis Potosí			None	None
699	México	Estado de México	Toluca		Metepec		None

582 rows × 7 columns

In [23]:
```
df3["place_with_parent_names"].str.split("|", expand = True)[2]
```

Out[23]:
```
0                    Distrito Federal
1                    Estado de México
2                    Estado de México
4      Veracruz de Ignacio de la Llave
5                             Jalisco
                    ...
695                           Jalisco
696                           Morelos
697                           Yucatán
698                    San Luis Potosí
699                  Estado de México
Name: 2, Length: 582, dtype: object
```

In [24]:
```
df3["state"] = df3["place_with_parent_names"].str.split("|", expand = True)[2]
df3.head()
```

Out[24]:

	property_type	place_with_parent_names	lat-lon	area_m2	price_usd	lat	
0	apartment	|México|Distrito Federal|Gustavo A. Madero|Acu...	19.52589,-99.151703	71.0	48550.59	19.52589	-9
1	house	|México|Estado de México|Toluca|Metepec|	19.2640539,-99.5727534	233.0	168636.73	19.2640539	-99
2	house	|México|Estado de México|Toluca|Toluca de Lerd...	19.268629,-99.671722	300.0	86932.69	19.268629	-9
4	apartment	|México|Veracruz de Ignacio de la Llave|Veracruz|	19.511938,-96.871956	84.0	68508.67	19.511938	-9
5	house	|México|Jalisco|Guadalajara|	20.689157,-103.366728	175.0	102763.00	20.689157	-10

In [25]:
```
# Drop both "place_with_parent_names" & "lat-lon" columns

df3.drop(columns = ["place_with_parent_names", "lat-lon"], inplace  = True)
df3.head()
```

Out[25]:

	property_type	area_m2	price_usd	lat	lon	state
0	apartment	71.0	48550.59	19.52589	-99.151703	Distrito Federal
1	house	233.0	168636.73	19.2640539	-99.5727534	Estado de México
2	house	300.0	86932.69	19.268629	-99.671722	Estado de México
4	apartment	84.0	68508.67	19.511938	-96.871956	Veracruz de Ignacio de la Llave
5	house	175.0	102763.00	20.689157	-103.366728	Jalisco

Concatenate DataFrames

Great work! You have three clean DataFrames, and now it's time to combine them into a single DataFrame so that you can conduct your analysis.

In [26]:
```
VimeoVideo("656313395", h="ccadbc2689", width=600)
```

Out[26]:

Task 1.2.7: Use `pd.concat` to concatenate `df1`, `df2`, `df3` as new DataFrame named `df`. Your new DataFrame should have 1,736 rows and 6 columns: `"property_type"`, `"state"`, `"lat"`, `"lon"`, `"area_m2"`, `"price_usd"`, and `"price_per_m2"`.

- Concatenate two or more DataFrames using pandas.

In [27]:
```
df = pd.concat([df1,df2,df3])
print(df.shape)
df.head()
```

```
(1736, 6)
```

Out[27]:

	property_type	state	lat	lon	area_m2	price_usd
0	house	Estado de México	19.560181	-99.233528	150.0	67965.56
1	house	Nuevo León	25.688436	-100.198807	186.0	63223.78

	property_type	state	lat	lon	area_m2	price_usd
2	apartment	Guerrero	16.767704	-99.764383	82.0	84298.37
3	apartment	Guerrero	16.829782	-99.911012	150.0	94308.80
5	house	Yucatán	21.052583	-89.538639	205.0	105191.37

Save `df`

The data is clean and in a single DataFrame, and now you need to save it as a CSV file so that you can examine it in your exploratory data analysis.

In [29]:
```
VimeoVideo("656312464", h="81ee04de15", width=600)
```

Out[29]:

Task 1.2.8: Save `df` as a CSV file using the `to_csv` method. The file path should be `"./data/mexico-real-estate-clean.csv"`. Be sure to set the `index` argument to `False`.

- What's a CSV file?
- Save a DataFrame as a CSV file using pandas.

In [33]:
```
df.to_csv("data/Bikas-1st-concatenate-dataset.csv", index = False) ## "index" False mea
```
