**1.2. PREPARING MEXICO DATA**

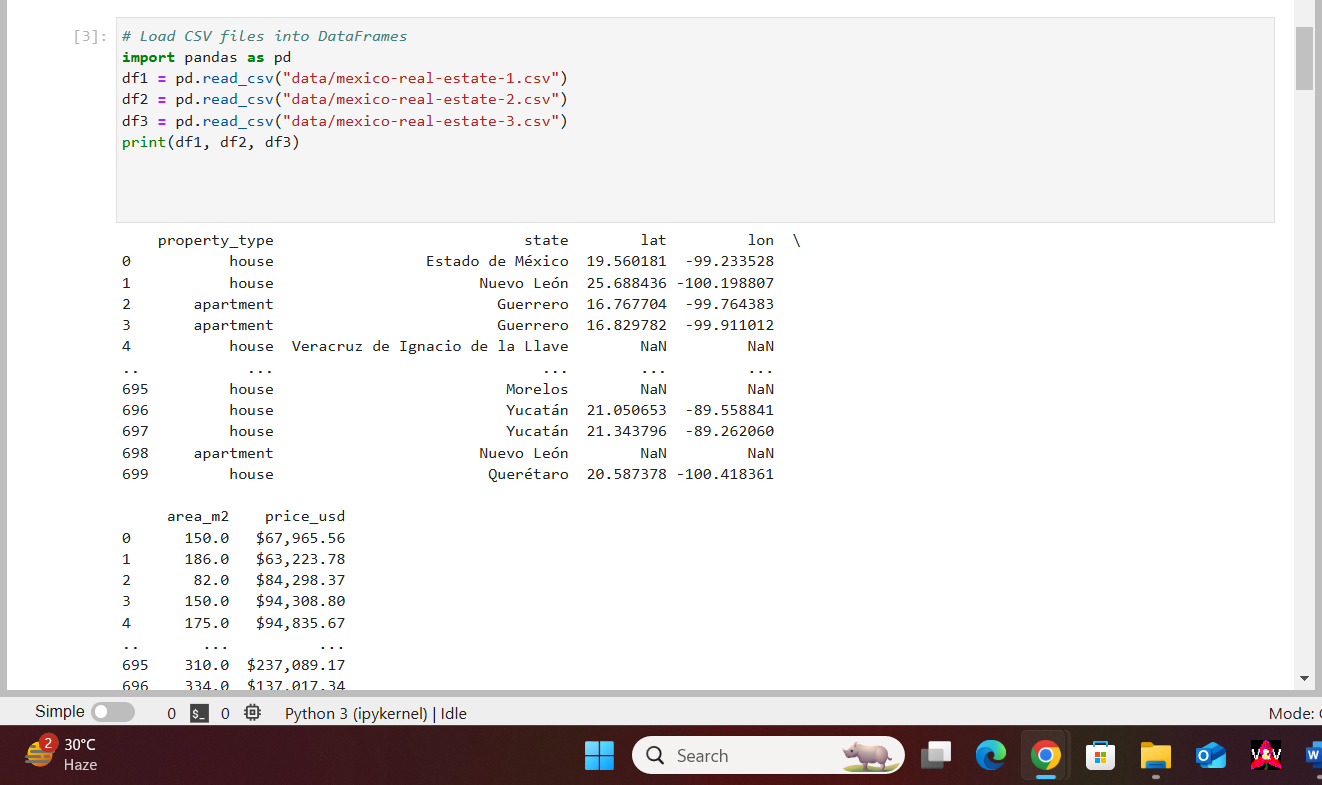
import pandas as pd

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.

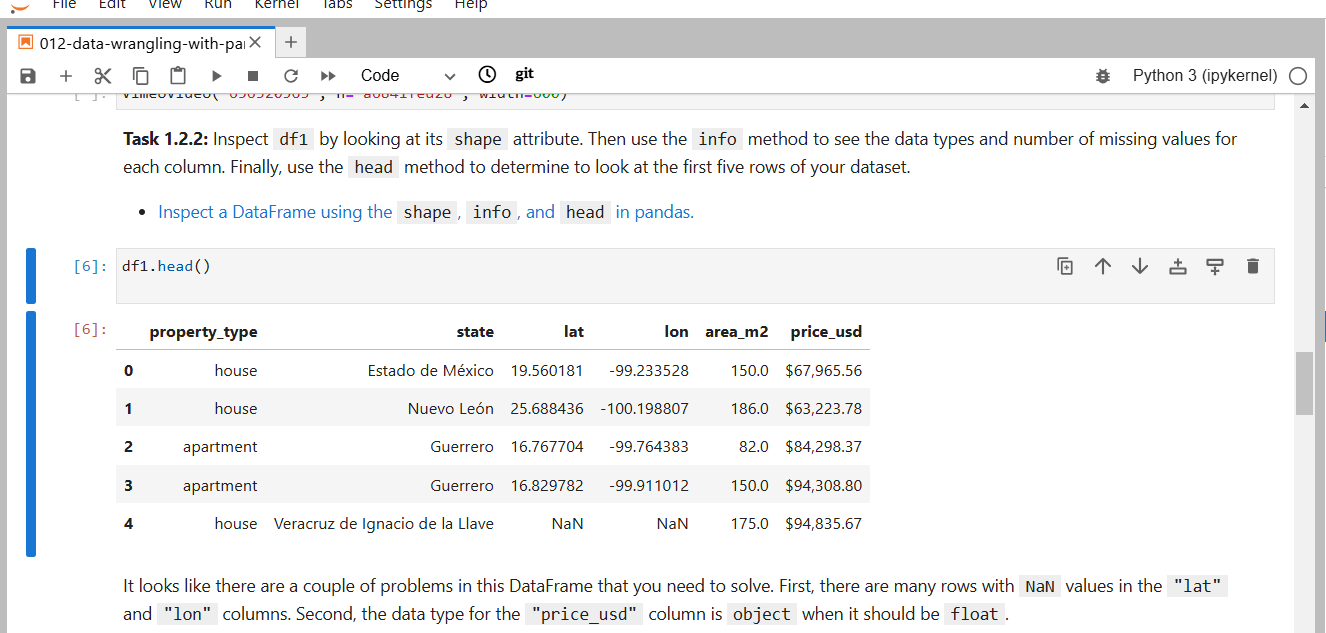
If you look in the small-data directory on your machine, you'll see that the data for this project comes in three CSV files: mexico-real-estate-1.csv, mexico-real-estate-2.csv, and mexico-real-estate-3.csv.



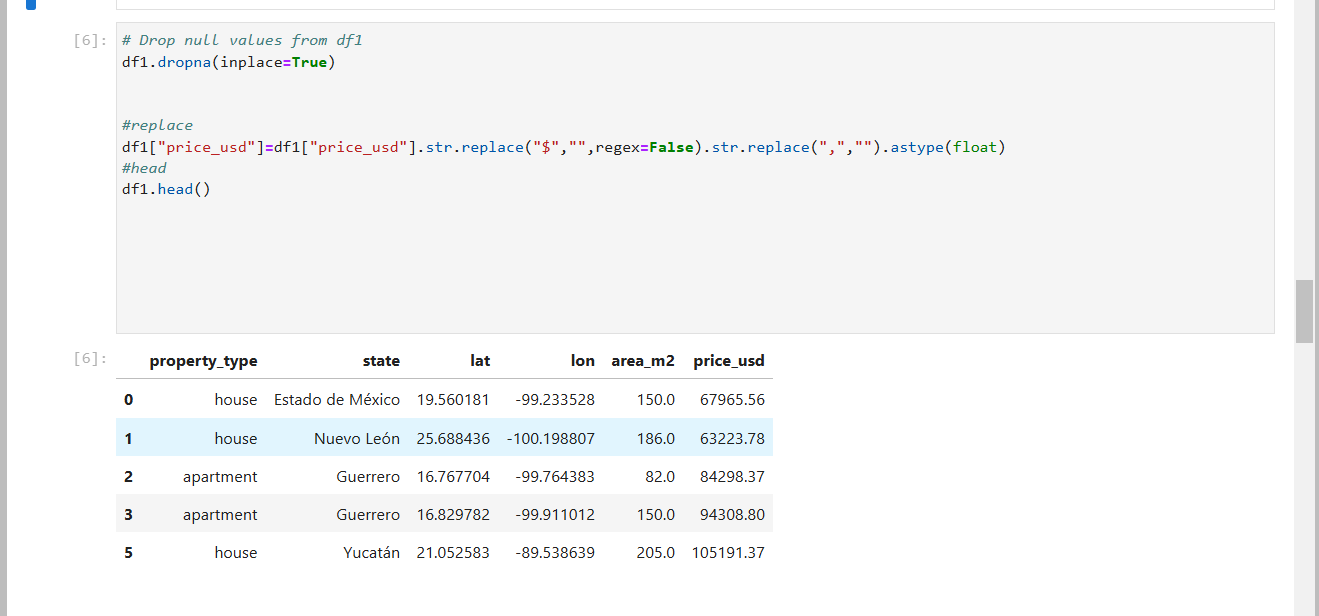
**Clean df1**

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.

**Task 1.2.2:** Inspect df1 by looking at its [shape](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.shape.html) attribute. Then use the [info](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.info.htm) method to see the data types and number of missing values for each column. Finally, use the [head](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html) method to determine to look at the first five rows of your dataset.



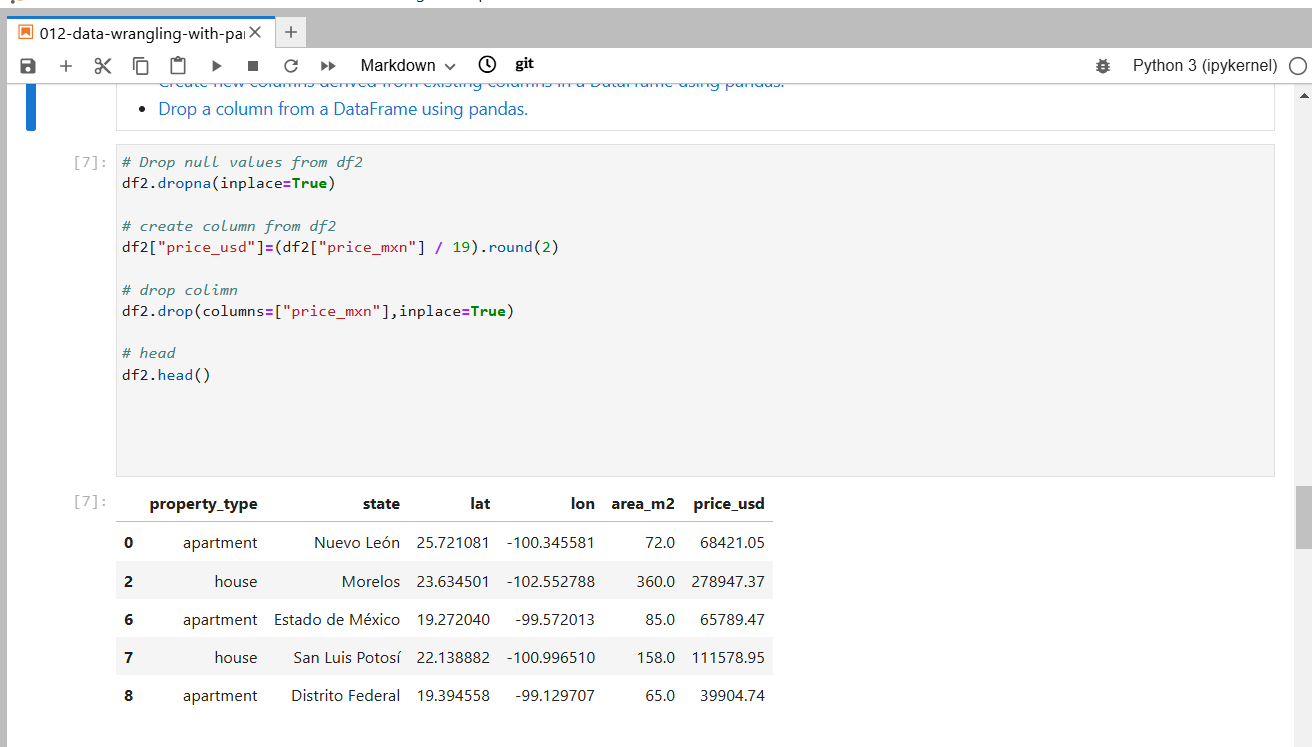
**Task 1.2.3:** Clean df1 by dropping rows with NaN values. Then remove the "$" and "," characters from "price\_usd" and recast the values in the column as floats.



**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.

**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.



**Clean df3**

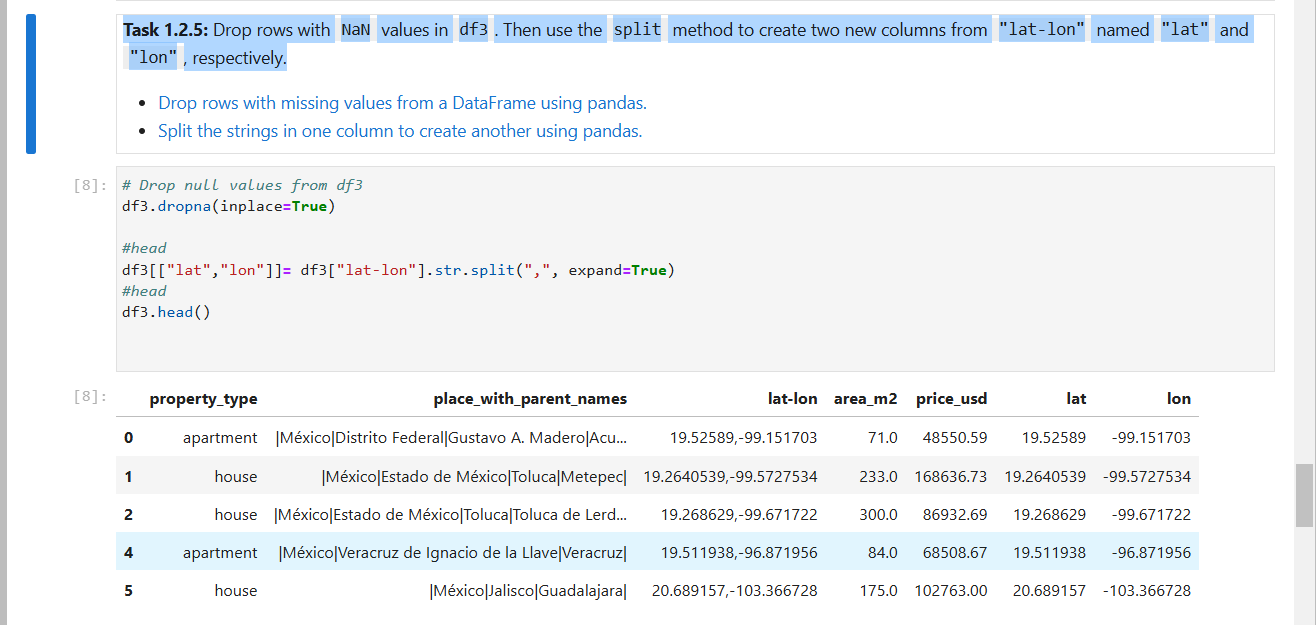
Great work! We're now on the final DataFrame. Use the same shape, info and head commands to inspect the df3. Do you see any familiar issues?

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.

**Task 1.2.5:** Drop rows with NaN values in df3. Then use the [split](https://pandas.pydata.org/docs/reference/api/pandas.Series.str.split.html) method to create two new columns from "lat-lon" named "lat" and "lon", respectively.

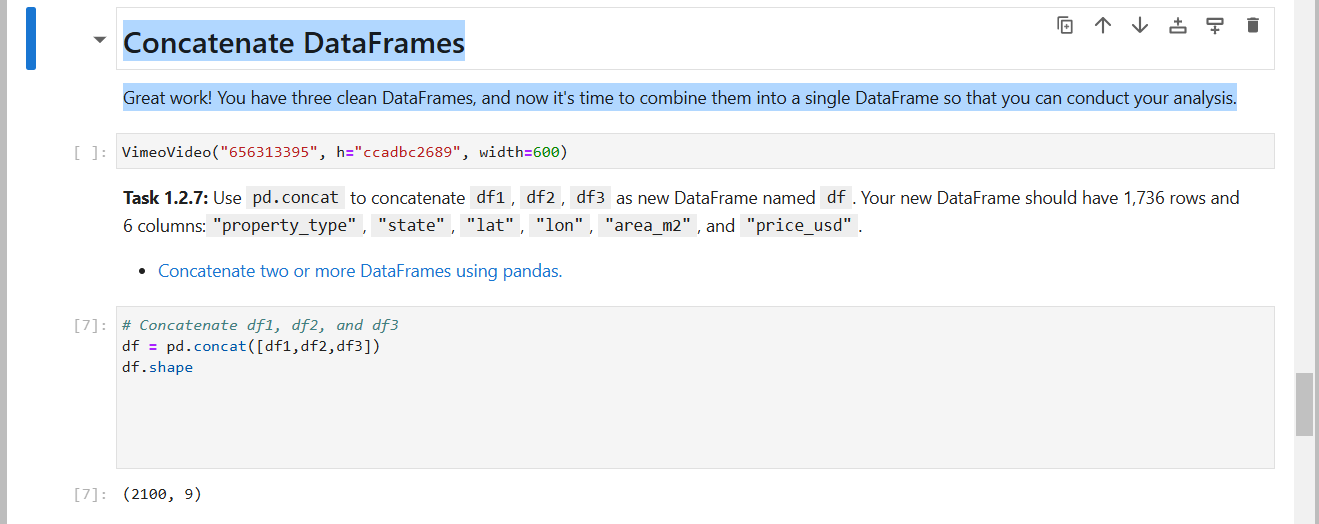


**Task 1.2.6:** Use the [split](https://pandas.pydata.org/docs/reference/api/pandas.Series.str.split.html) method again, this time to extract the state for every house. (Note that the state name always appears after "México|" in each string.) Use this information to create a "state" column. Finally, drop the "place\_with\_parent\_names" and "lat-lon" columns from the DataFrame.



**Concatenate DataFrames**

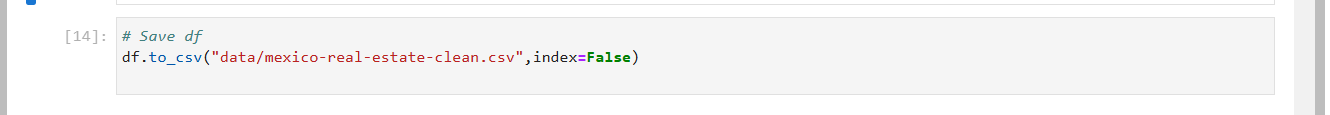
Great work! You have three clean DataFrames, and now it's time to combine them into a single DataFrame so that you can conduct your analysis.



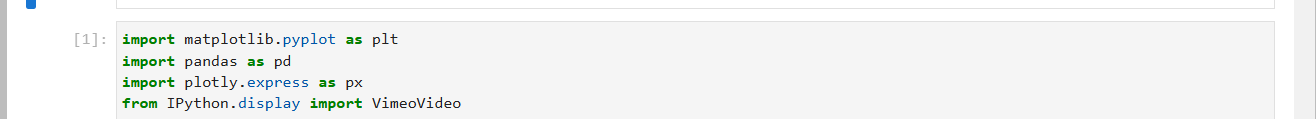
**Save df**

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.

**Task 1.2.8:** Save df as a CSV file using the [to\_csv](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html" \t "_blank) method. The file path should be "./data/mexico-real-estate-clean.csv". Be sure to set the index argument to False.



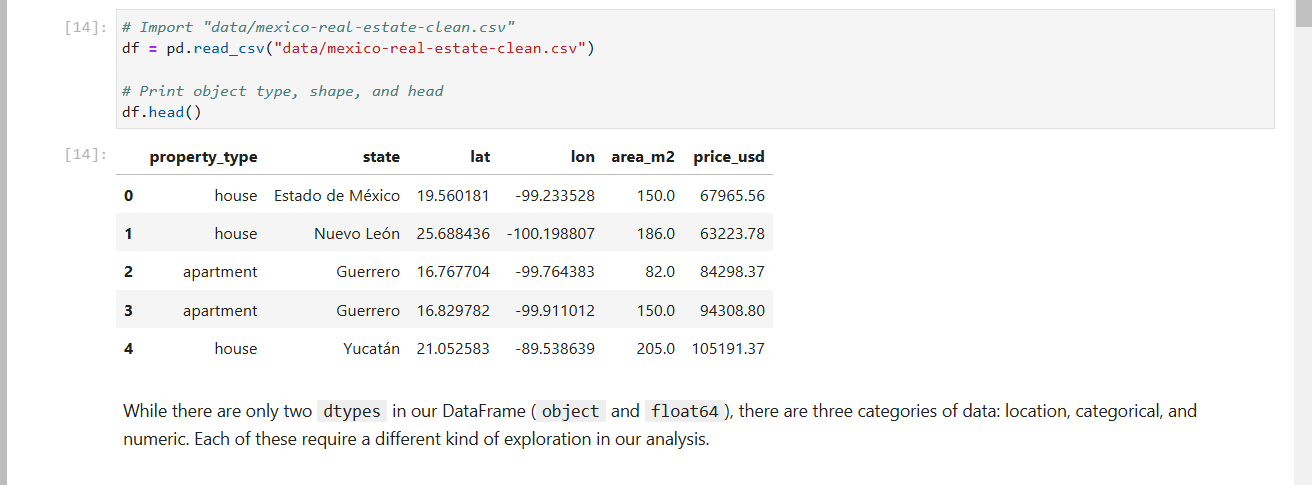
**1.3. EXPLORATORY DATA ANALYSIS**



After importing, the next step in many data science projects is exploratory data analysis (EDA), where you get a feel for your data by summarizing its main characteristics using descriptive statistics and data visualization. A good way to plan your EDA is by looking each column and asking yourself questions what it says about your dataset.

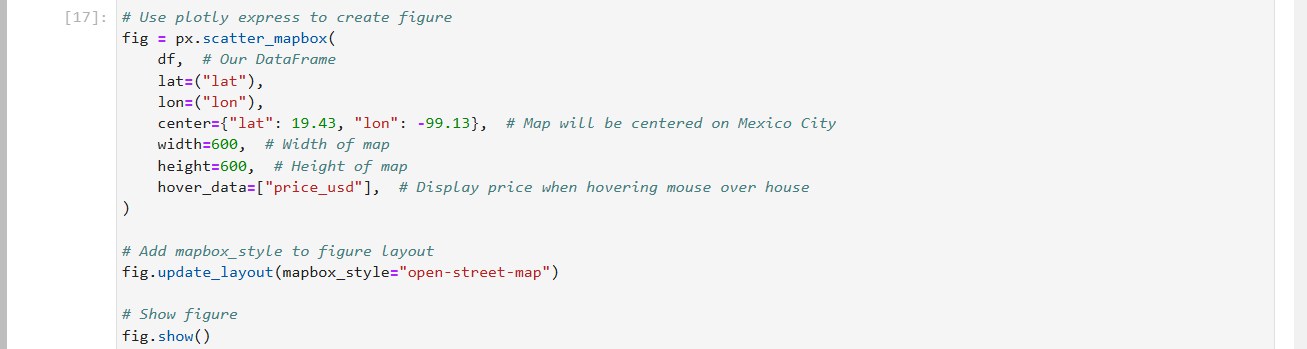
**Import Data**

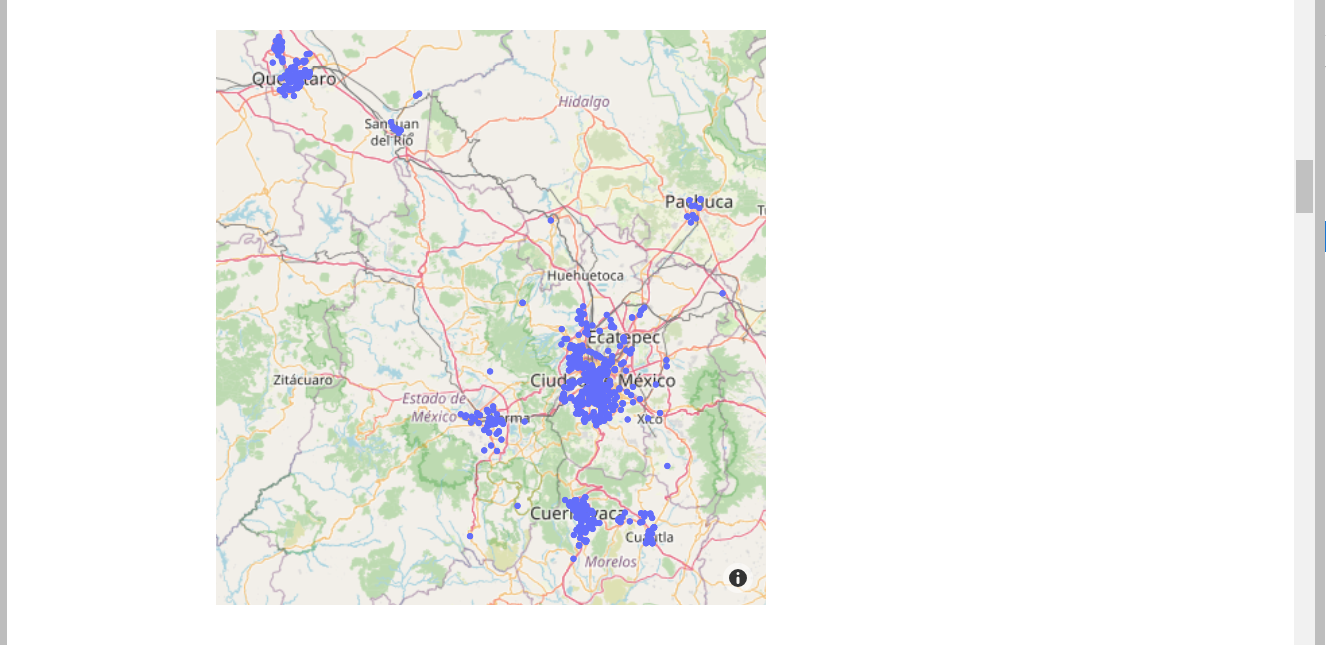
**Task 1.3.1:** Read the CSV file that you created in the last notebook ("../small-data/mexico-real-estate-clean.csv") into a DataFrame named df. Be sure to check that all your columns are the correct data type before you go to the next task.



**Location Data: "lat" and "lon"**

They say that the most important thing in real estate is location, and we can see where where in Mexico our houses are located by using the "lat" and "lon" columns. Since latitude and longitude are based on a coordinate system, a good way to visualize them is to create a scatter plot on top of a map. A great tool for this is the [scatter\_mapbox](https://plotly.github.io/plotly.py-docs/generated/plotly.express.scatter_mapbox.html" \t "_blank) from the plotly library.

**Task 1.3.2:** Add "lat" and "lon" to the code below, and run the code. You'll see a map that's centered on Mexico City, and you can use the "Zoom Out" button in the upper-right corner of the map so that you can see the whole country.

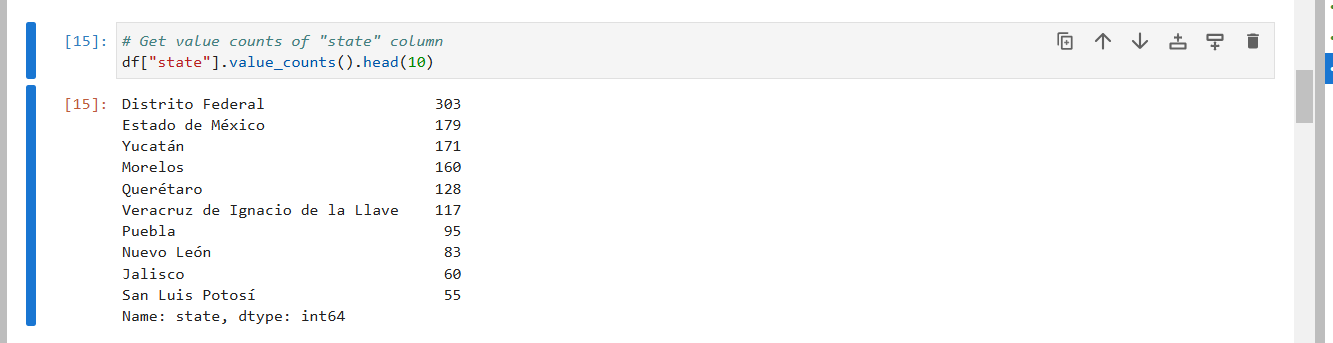


Looking at this map, are the houses in our dataset distributed evenly throughout the country, or are there states or regions that are more prevalent? Can you guess where Mexico's biggest cities are based on this distribution?

**Categorical Data: "state"**

Even though we can get a good idea of which states are most common in our dataset from looking at a map, we can also get the exact count by using the "state" column.

**Task 1.3.3:** Use the [value\_counts](https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html" \t "_blank) method on the "state" column to determine the 10 most prevalent states in our dataset.



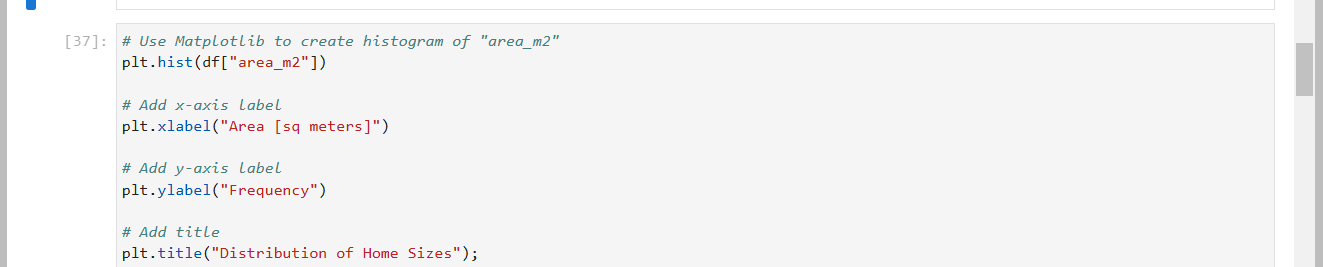
**Numerical Data: "area\_m2" and "price\_usd"**

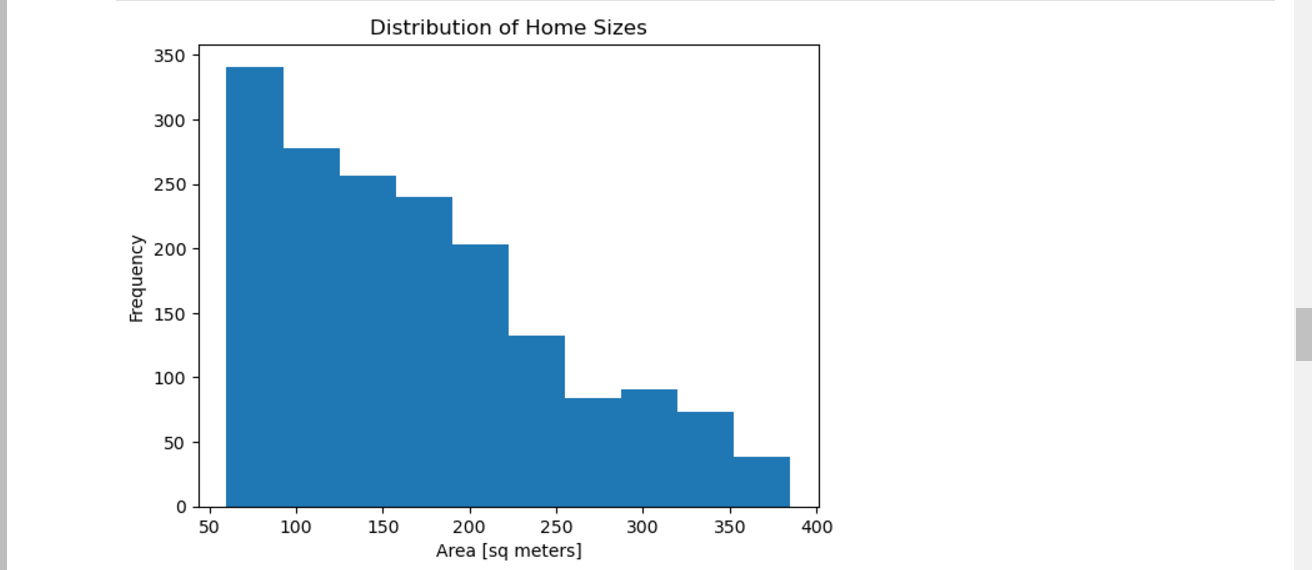
We have a sense for where the houses in our dataset are located, but how much do they cost? How big are they? The best way to answer those questions is looking at descriptive statistics.



Let's start by looking at "area\_m2". It's interesting that the mean is larger than the median (another name for the 50% quartile). Both of these statistics are supposed to give an idea of the "typical" value for the column, so why is there a difference of almost 15 m2 between them? To answer this question, we need to see how house sizes are distributed in our dataset. Let's look at two ways to visualize the distribution: a histogram and a boxplot.

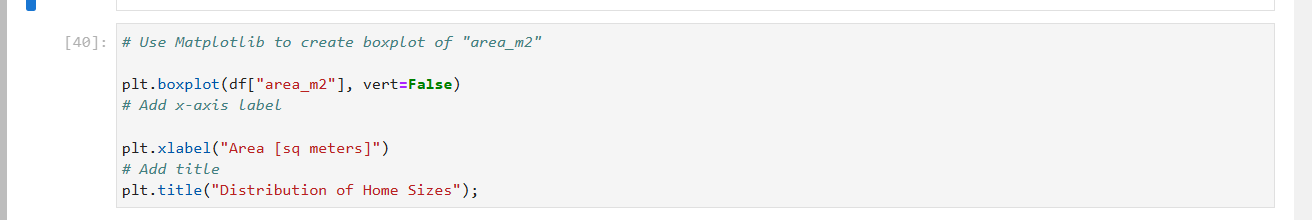
**Task 1.3.5:** Create a histogram of "area\_m2". Make sure that the x-axis has the label "Area [sq meters]", the y-axis has the label "Frequency", and the plot has the title "Distribution of Home Sizes".

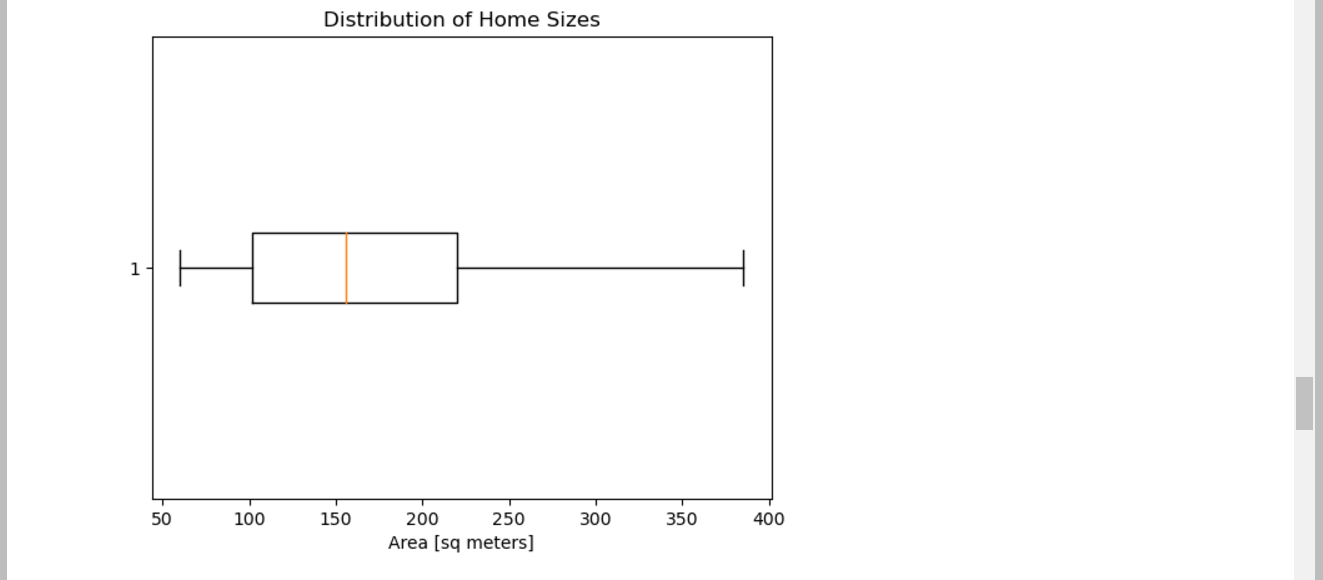




Looking at our histogram, we can see that "area\_m2" skews right. In other words, there are more houses at the lower end of the distribution (50–200m2) than at the higher end (250–400m2). That explains the difference between the mean and the median.

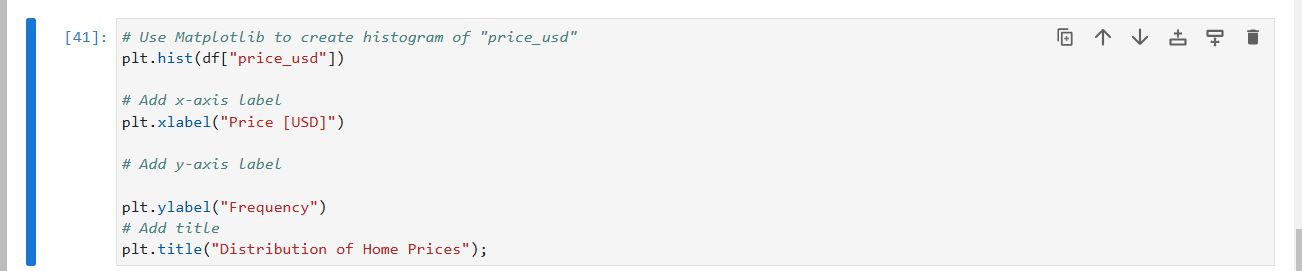
**Task 1.3.6:** Create a horizontal boxplot of "area\_m2". Make sure that the x-axis has the label "Area [sq meters]" and the plot has the title "Distribution of Home Sizes". How is the distribution and its left skew represented differently here than in your histogram?

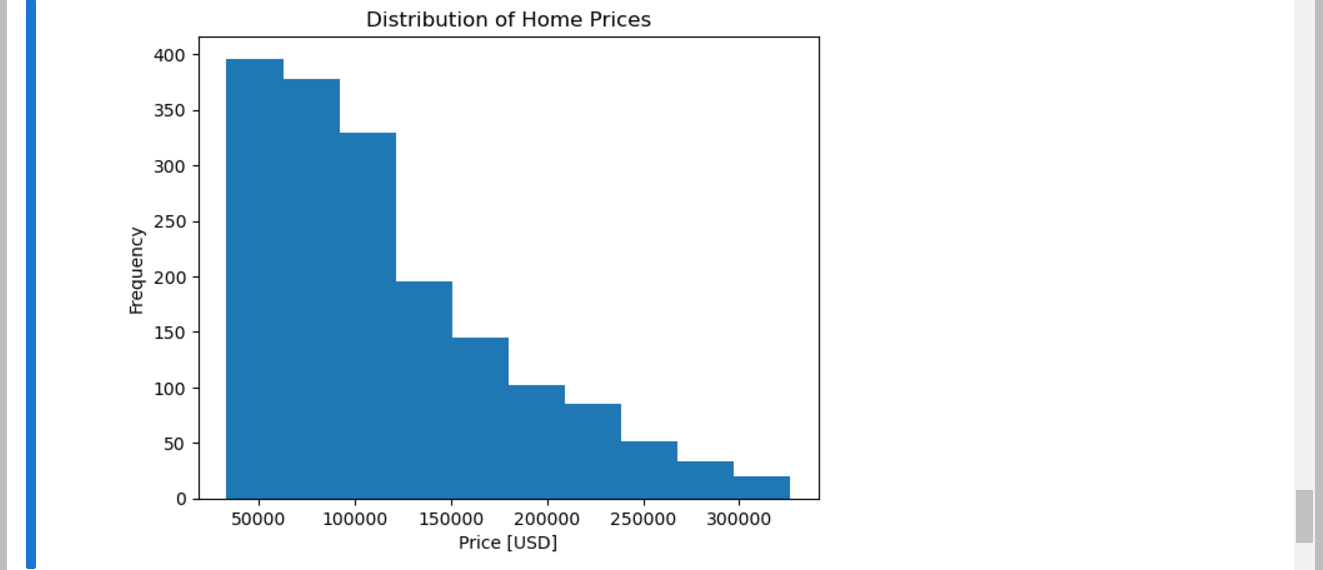




Does "price\_usd" have the same distribution as "price\_per\_m2"? Let's use the same two visualization tools to find out.

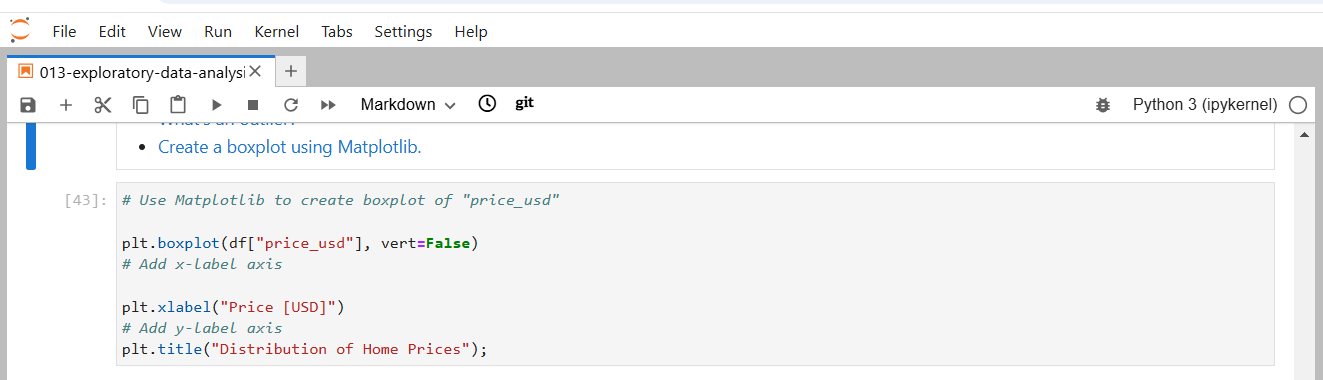
**Task 1.3.7:** Create a histogram of "price\_usd". Make sure that the x-axis has the label "Price [USD]", the y-axis has the label "Frequency", and the plot has the title "Distribution of Home Prices".

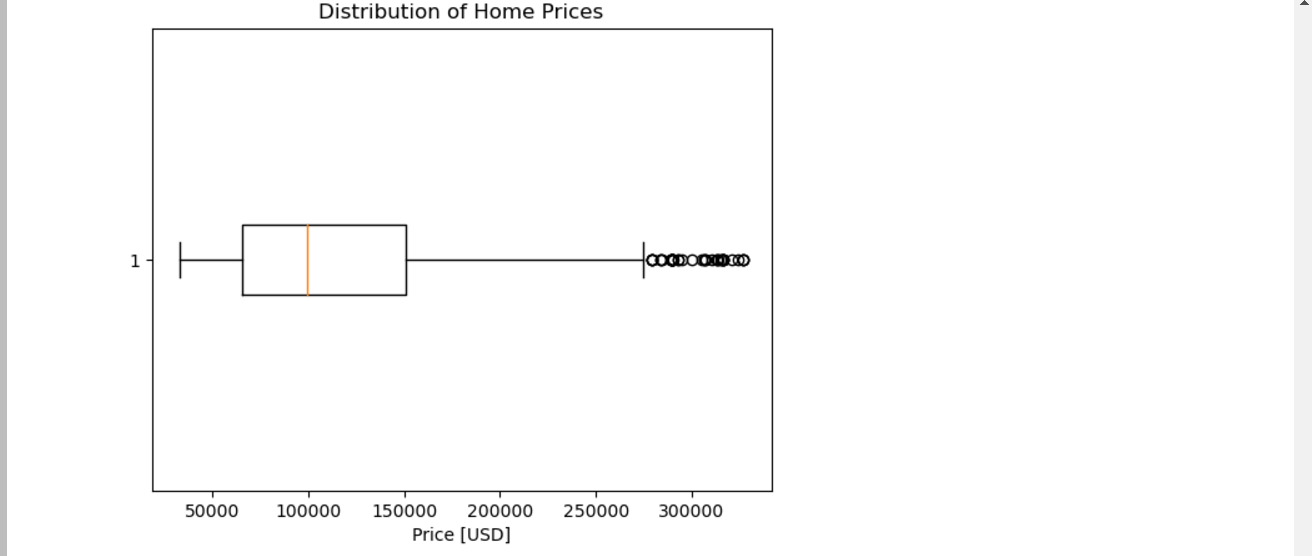




Looks like "price\_usd" is even more skewed than "area\_m2". What does this bigger skew look like in a boxplot?

**Task 1.3.8:** Create a horizontal boxplot of "price\_usd". Make sure that the x-axis has the label "Price [USD]" and the plot has the title "Distribution of Home Prices".





Great work