National Housing Value Distribution

Utilizing Zillow Data

By: Gustavo Carmona, Kamila Czosnyka,
Sam Duffy, Caila Hanus

Project Overview

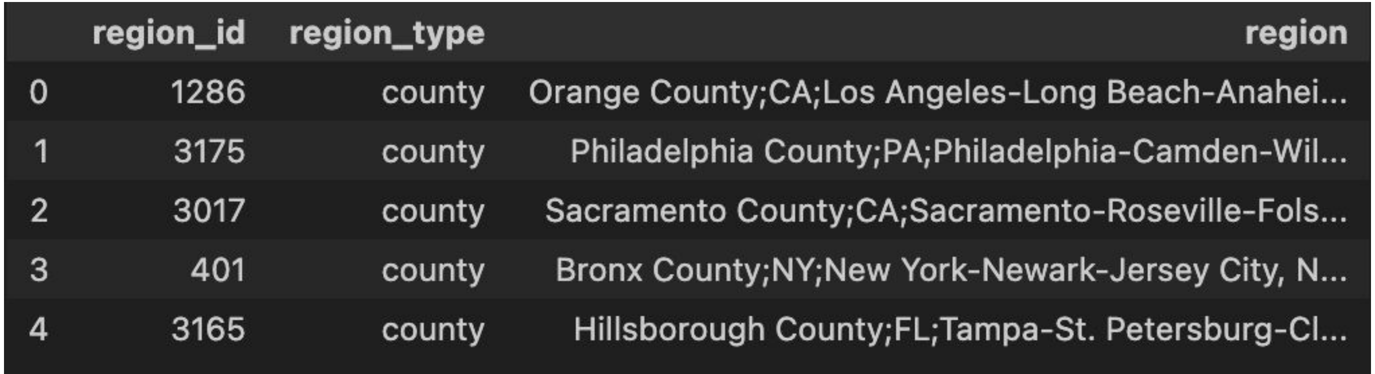
- Engineered a datafile with over 1 million lines of data! (1,048,576 lines to be exact!)
- Used Zillow CSV files and API calls, we used ETL processes to clean, edit, and ultimately upload the data to SQL
- Used data to compare Chicago-specific housing values to major US cities
- Used data to compare Chicago-area housing data to itself
- This data could be used to look at historical changes in a certain area or to compare different regions

API Resources / Data Extraction

- https://data.nasdaq.com/databases/ZILLOW
- This was used to both call data directly from the API as well as download pre-loaded CSV files
- The CSV files were needed as they provided information on the different Indicator and Region IDs which were needed in the URL of the API call.

Challenges

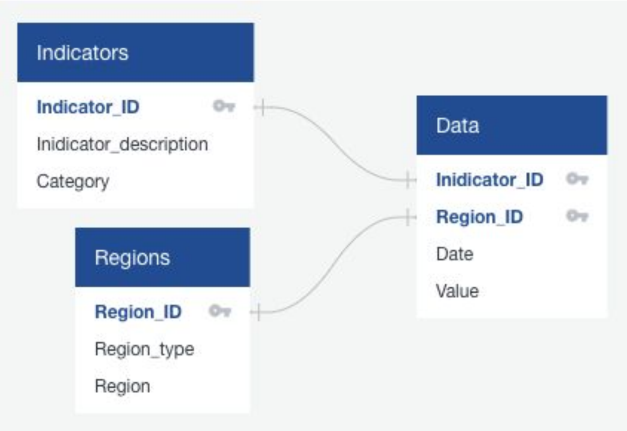
- While the data was geographically based, there was no geographical information provided other than region names and zip codes, so it proved challenging when mapping the data in a visualization
- The region CSV file required some cleaning so that all data lined up, as well as understanding how each region (zip, metro, neigh, city, county) was organized within the “region” column
- Could only call one indicator and one region at a time with API



	region_id	region_type	region
0	1286	county	Orange County;CA;Los Angeles-Long Beach-Anahei...
1	3175	county	Philadelphia County;PA;Philadelphia-Camden-Wil...
2	3017	county	Sacramento County;CA;Sacramento-Roseville-Fols...
3	401	county	Bronx County;NY;New York-Newark-Jersey City, N...
4	3165	county	Hillsborough County;FL;Tampa-St. Petersburg-Cl...

Transformation(Exploring Data)

- 3 data files extracted from API
    - Indicator
        - Summary of what each indicator_ID represented
            - Property size (Bedroom)
            - Property type (Condo, single family)
        - All ID’s tied to category in Home Values and Sales
    - Region
        - Summary of Region ID data based on types
            - City, County, Neighborhood, Metro Area, and Zip-code
    - Data
        - Holds the Home values and dates based on region and indicator ID’s



Indicators

Indicator_ID
Inidicator_description
Category

Regions

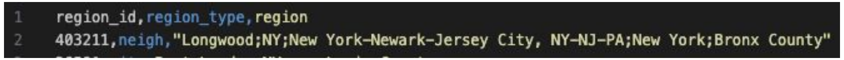
Region_ID
Region_type
Region

Data

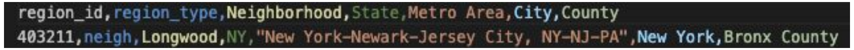
Inidicator_ID
Region_ID
Date
Value

Transformation(Cleaning)

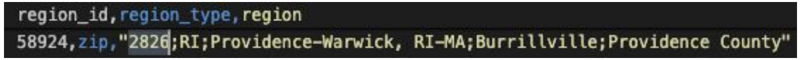
- Cleaning (PANDAS & PYDANTIC)
  - Compare (MERGE) & drop unnecessary data from Indicator and Region files
  - Utilized PYDANTIC to validate the data types on our extracted files (INT, OBJ)
  - Restructure for analysis: Region file data filtered & split by type(City, County, Metro, Neighborhood, and Zip-Code)
    - Splitting columns & renaming
      - Replacing Null data with 'NA'
      - Editing numerical data (Zip-Codes missing a number - invalid foreign codes, and converting dates)
- Results
  - 28/ 56 Indicators_id's
  - 27,879/ 89,306 Region_id's
  - 1,048,576 Data values w/ Dates



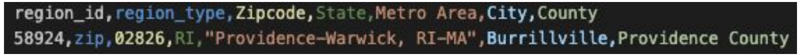
```
region_id,region_type,region
403211,neigh,"Longwood;NY;New York-Newark-Jersey City, NY-NJ-PA;New York;Bronx County"
```



```
region_id,region_type,Neighborhood,State,Metro Area,City,County
403211,neigh,Longwood,NY,"New York-Newark-Jersey City, NY-NJ-PA",New York,Bronx County
```

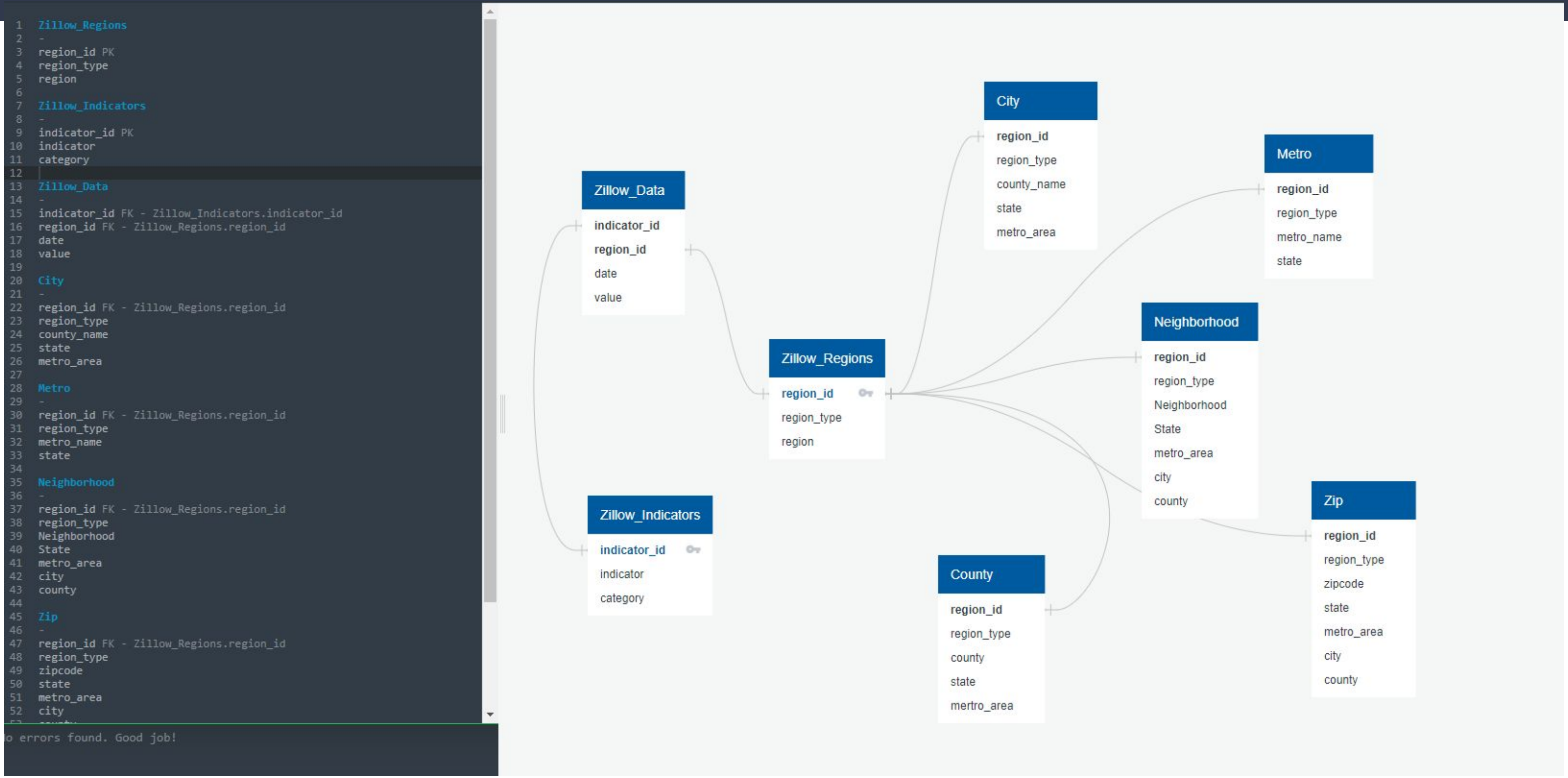


```
region_id,region_type,region
58924,zip,"2826;RI;Providence-Warwick, RI-MA;Burrillville;Providence County"
```



```
region_id,region_type,Zipcode,State,Metro Area,City,County
58924,zip,02826,RI,"Providence-Warwick, RI-MA",Burrillville,Providence County
```

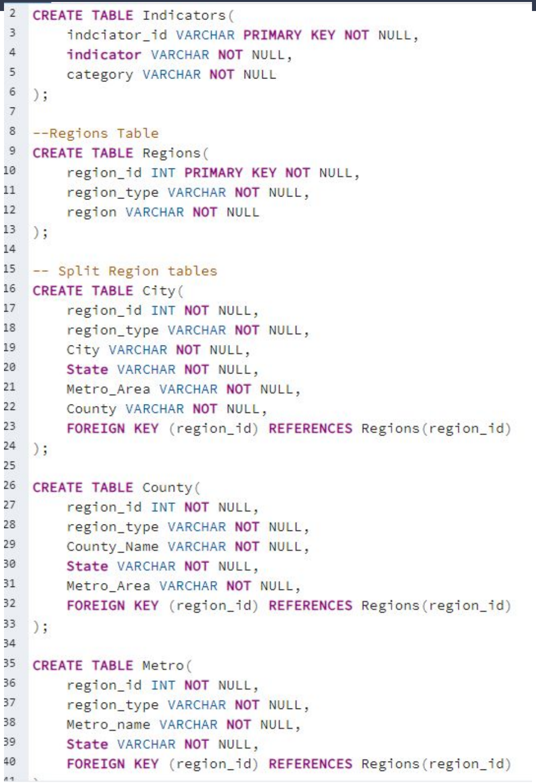
ERD Diagram



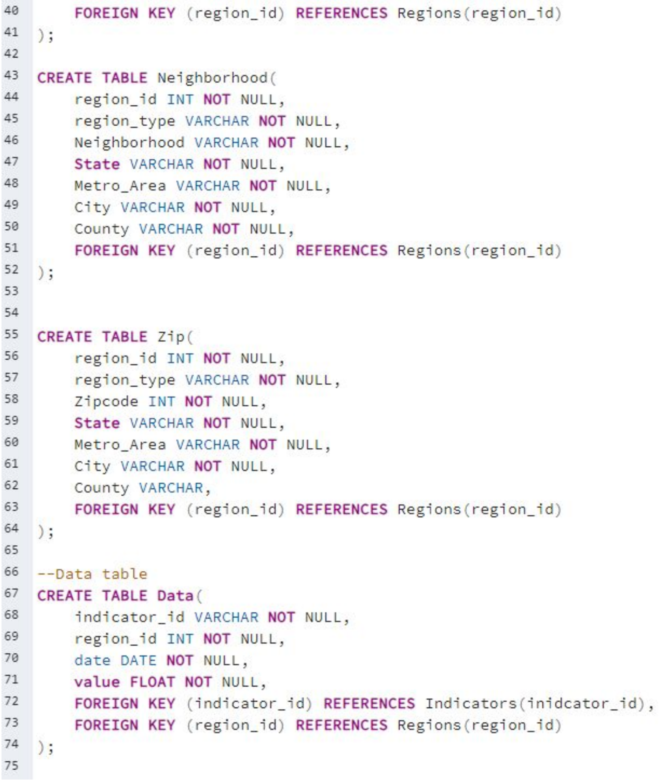
```
Zillow_Regions
-
region_id PK
region_type
region

Zillow_Indicators
-
indicator_id PK
indicator
category

Zillow_Data
-
indicator_id FK - Zillow_Indicators.indicator_id
region_id FK - Zillow_Regions.region_id
date
value

City
-
region_id FK - Zillow_Regions.region_id
region_type
county_name
state
metro_area

Metro
-
region_id FK - Zillow_Regions.region_id
region_type
metro_name
state

Neighborhood
-
region_id FK - Zillow_Regions.region_id
region_type
Neighborhood
State
metro_area
city
county

Zip
-
region_id FK - Zillow_Regions.region_id
region_type
zipcode
state
metro_area
city
```

```
o errors found. Good job!
```

SQL Database Storage



```sql
CREATE TABLE Indicators(
    indciator_id VARCHAR PRIMARY KEY NOT NULL,
    indicator VARCHAR NOT NULL,
    category VARCHAR NOT NULL
);

--Regions Table
CREATE TABLE Regions(
    region_id INT PRIMARY KEY NOT NULL,
    region_type VARCHAR NOT NULL,
    region VARCHAR NOT NULL
);

-- Split Region tables
CREATE TABLE City(
    region_id INT NOT NULL,
    region_type VARCHAR NOT NULL,
    City VARCHAR NOT NULL,
    State VARCHAR NOT NULL,
    Metro_Area VARCHAR NOT NULL,
    County VARCHAR NOT NULL,
    FOREIGN KEY (region_id) REFERENCES Regions(region_id)
);

CREATE TABLE County(
    region_id INT NOT NULL,
    region_type VARCHAR NOT NULL,
    County_Name VARCHAR NOT NULL,
    State VARCHAR NOT NULL,
    Metro_Area VARCHAR NOT NULL,
    FOREIGN KEY (region_id) REFERENCES Regions(region_id)
);

CREATE TABLE Metro(
    region_id INT NOT NULL,
    region_type VARCHAR NOT NULL,
    Metro_name VARCHAR NOT NULL,
    State VARCHAR NOT NULL,
    FOREIGN KEY (region_id) REFERENCES Regions(region_id)
```



```sql
    FOREIGN KEY (region_id) REFERENCES Regions(region_id)
);

CREATE TABLE Neighborhood(
    region_id INT NOT NULL,
    region_type VARCHAR NOT NULL,
    Neighborhood VARCHAR NOT NULL,
    State VARCHAR NOT NULL,
    Metro_Area VARCHAR NOT NULL,
    City VARCHAR NOT NULL,
    County VARCHAR NOT NULL,
    FOREIGN KEY (region_id) REFERENCES Regions(region_id)
);


CREATE TABLE Zip(
    region_id INT NOT NULL,
    region_type VARCHAR NOT NULL,
    Zipcode INT NOT NULL,
    State VARCHAR NOT NULL,
    Metro_Area VARCHAR NOT NULL,
    City VARCHAR NOT NULL,
    County VARCHAR,
    FOREIGN KEY (region_id) REFERENCES Regions(region_id)
);

--Data table
CREATE TABLE Data(
    indicator_id VARCHAR NOT NULL,
    region_id INT NOT NULL,
    date DATE NOT NULL,
    value FLOAT NOT NULL,
    FOREIGN KEY (indicator_id) REFERENCES Indicators(inidcator_id),
    FOREIGN KEY (region_id) REFERENCES Regions(region_id)
);
```

API Calls

Indicator ID Used:

- ZSFH: Single Family

Other Indicator Examples

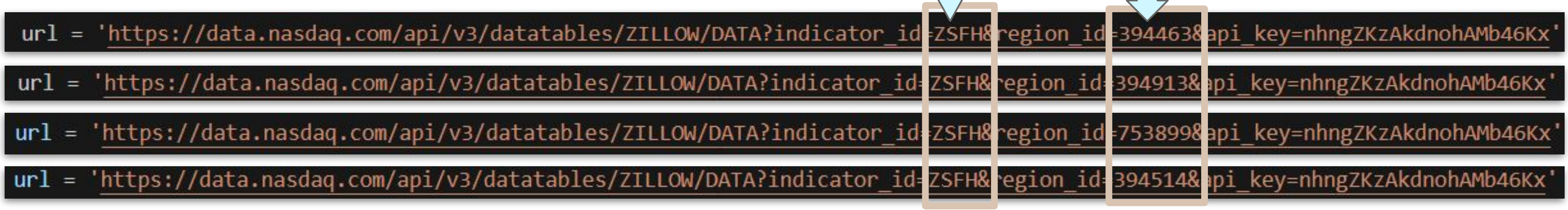
- ZATT: All Homes- Top Tier
- ZALL: All Homes
- Z1BR: 1- Bedroom Home
- Z2BR: 2- Bedroom Home

(Please reference Zillow Indicators.csv for more options)

Region ID

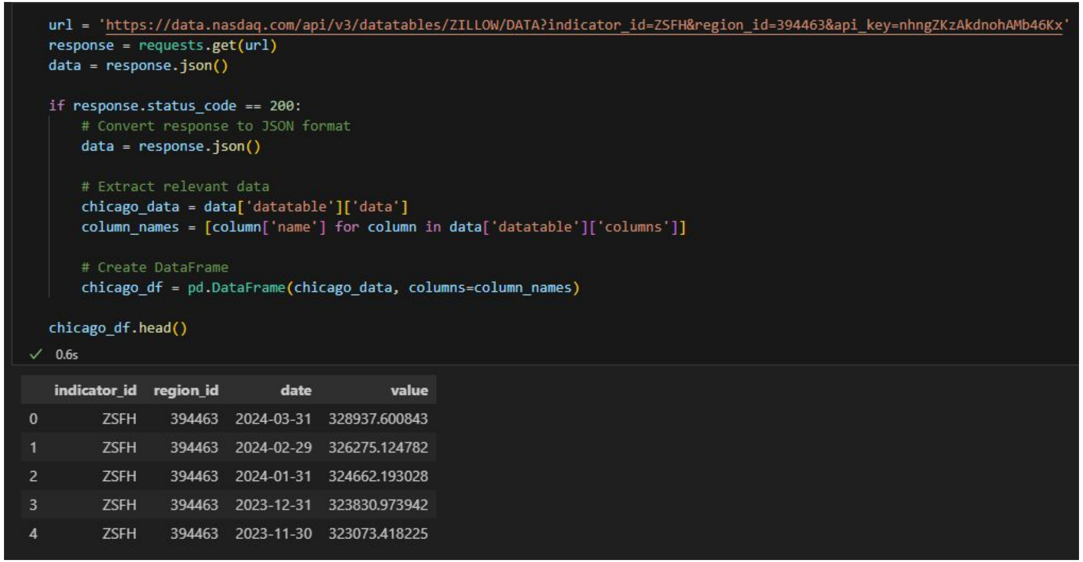
- 394463 = Chicago Metro Area
- 394913 = New York City Metro Area
- 753899 = Los Angeles Metro Area
- 394514 = Dallas Metro Area

(Reference Zillow Regions.csv for more options)



```
url = 'https://data.nasdaq.com/api/v3/datatables/ZILLOW/DATA?indicator_id=ZSFH&region_id=394463&api_key=nhngZKzAkdnohAMb46Kx'
url = 'https://data.nasdaq.com/api/v3/datatables/ZILLOW/DATA?indicator_id=ZSFH&region_id=394913&api_key=nhngZKzAkdnohAMb46Kx'
url = 'https://data.nasdaq.com/api/v3/datatables/ZILLOW/DATA?indicator_id=ZSFH&region_id=753899&api_key=nhngZKzAkdnohAMb46Kx'
url = 'https://data.nasdaq.com/api/v3/datatables/ZILLOW/DATA?indicator_id=ZSFH&region_id=394514&api_key=nhngZKzAkdnohAMb46Kx'
```

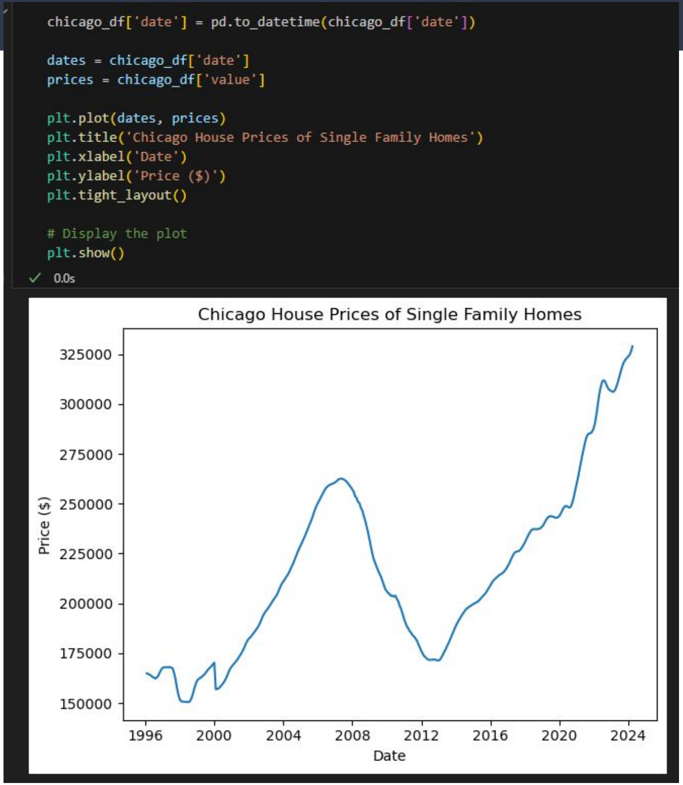
Sample API Call to Create Pandas DataFrame



```python
url = 'https://data.nasdaq.com/api/v3/datatables/ZILLOW/DATA?indicator_id=ZSFH&region_id=394463&api_key=nhngZKzAkdnohAMb46Kx'
response = requests.get(url)
data = response.json()

if response.status_code == 200:
    # Convert response to JSON format
    data = response.json()

    # Extract relevant data
    chicago_data = data['datatable']['data']
    column_names = [column['name'] for column in data['datatable']['columns']]

    # Create DataFrame
    chicago_df = pd.DataFrame(chicago_data, columns=column_names)

chicago_df.head()
```

✓ 0.6s

	indicator_id	region_id	date	value
0	ZSFH	394463	2024-03-31	328937.600843
1	ZSFH	394463	2024-02-29	326275.124782
2	ZSFH	394463	2024-01-31	324662.193028
3	ZSFH	394463	2023-12-31	323830.973942
4	ZSFH	394463	2023-11-30	323073.418225

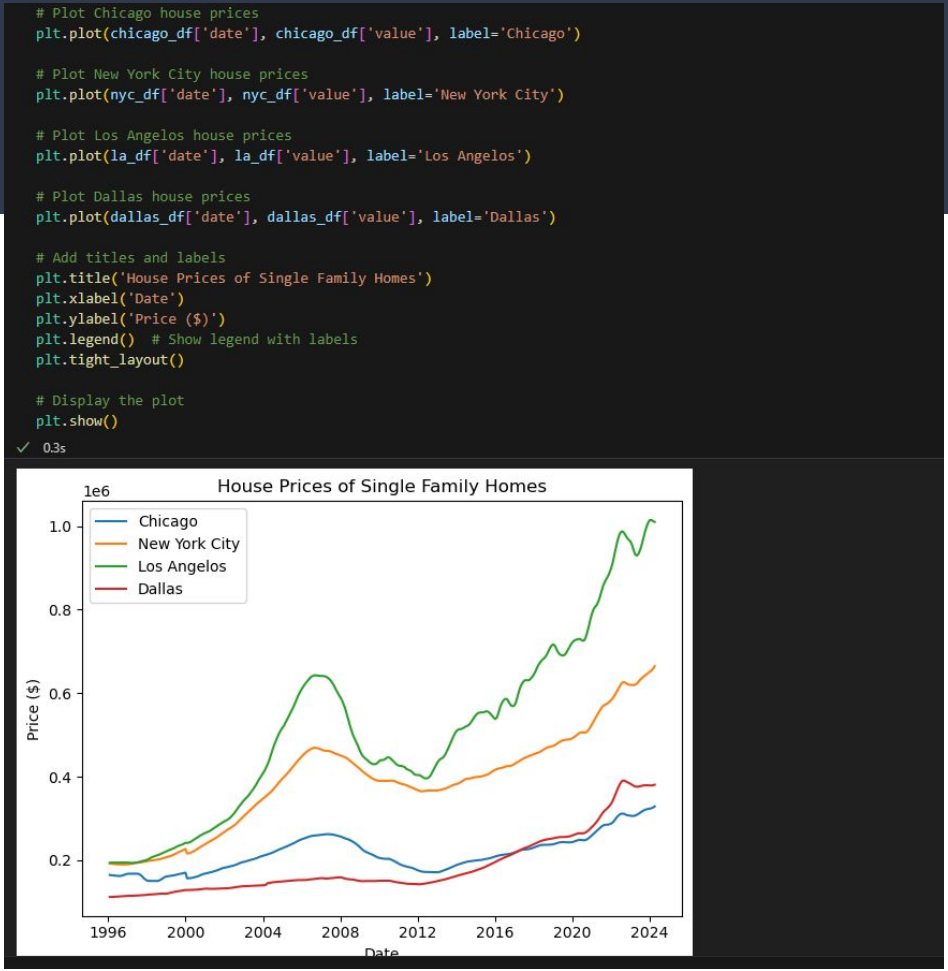


```python
chicago_df['date'] = pd.to_datetime(chicago_df['date'])

dates = chicago_df['date']
prices = chicago_df['value']

plt.plot(dates, prices)
plt.title('Chicago House Prices of Single Family Homes')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.tight_layout()

# Display the plot
plt.show()
```

✓ 0.0s

API Call with Pandas

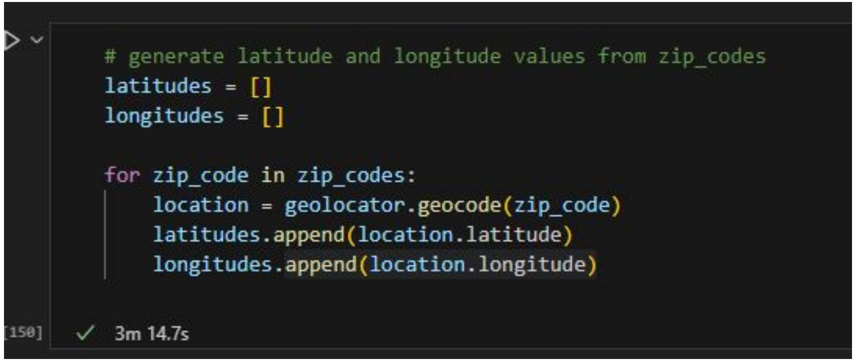
By making multiple API calls, each using a distinct region code, we gathered house value data for single-family homes across four major cities. This data was then plotted on a single chart, resulting in four scatterplots illustrating the trend of home values over the years



```
# Plot Chicago house prices
plt.plot(chicago_df['date'], chicago_df['value'], label='Chicago')

# Plot New York City house prices
plt.plot(nyc_df['date'], nyc_df['value'], label='New York City')

# Plot Los Angelos house prices
plt.plot(la_df['date'], la_df['value'], label='Los Angelos')

# Plot Dallas house prices
plt.plot(dallas_df['date'], dallas_df['value'], label='Dallas')

# Add titles and labels
plt.title('House Prices of Single Family Homes')
plt.xlabel('Date')
plt.ylabel('Price ($)')
plt.legend()  # Show legend with labels
plt.tight_layout()

# Display the plot
plt.show()
```

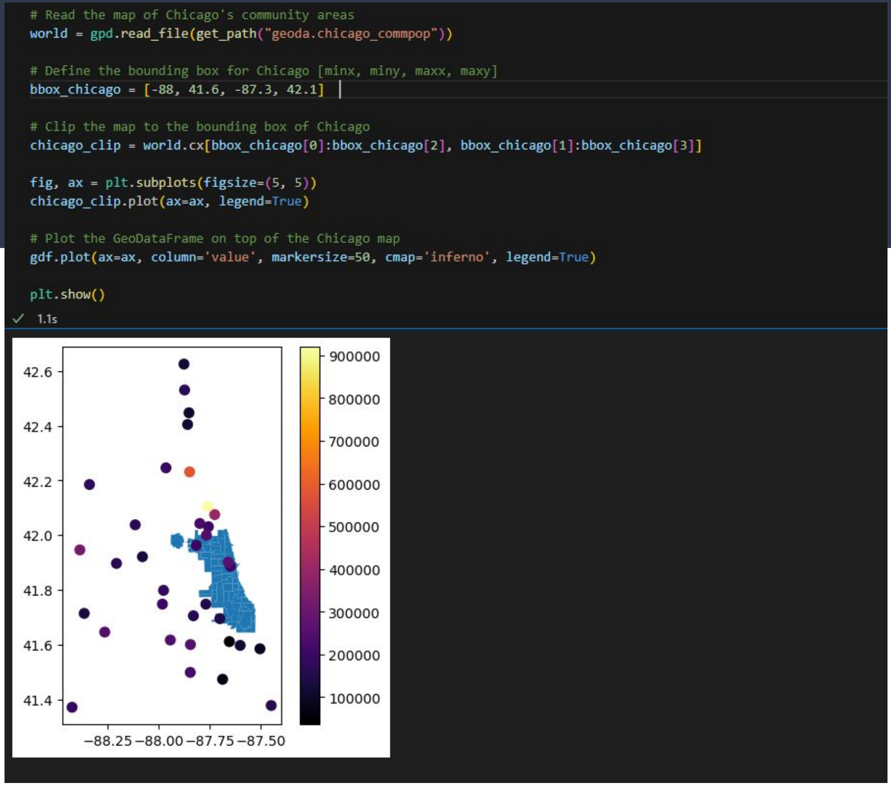
✓ 0.3s

GeoPandas



```
# generate latitude and longitude values from zip_codes
latitudes = []
longitudes = []

for zip_code in zip_codes:
    location = geolocator.geocode(zip_code)
    latitudes.append(location.latitude)
    longitudes.append(location.longitude)
```

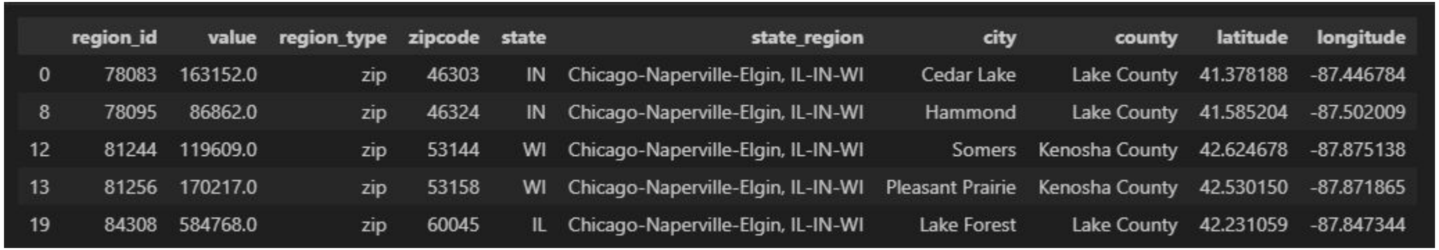
[150] ✓ 3m 14.7s



```
# Read the map of Chicago's community areas
world = gpd.read_file(get_path("geoda.chicago_commpop"))

# Define the bounding box for Chicago [minx, miny, maxx, maxy]
bbox_chicago = [-88, 41.6, -87.3, 42.1]

# Clip the map to the bounding box of Chicago
chicago_clip = world.cx[bbox_chicago[0]:bbox_chicago[2], bbox_chicago[1]:bbox_chicago[3]]

fig, ax = plt.subplots(figsize=(5, 5))
chicago_clip.plot(ax=ax, legend=True)

# Plot the GeoDataFrame on top of the Chicago map
gdf.plot(ax=ax, column='value', markersize=50, cmap='inferno', legend=True)

plt.show()
```

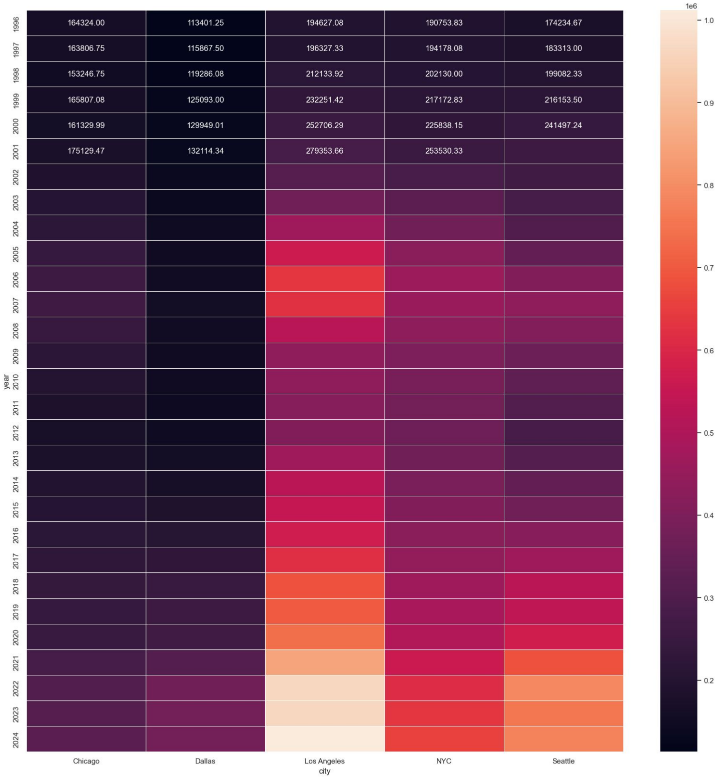
✓ 1.1s

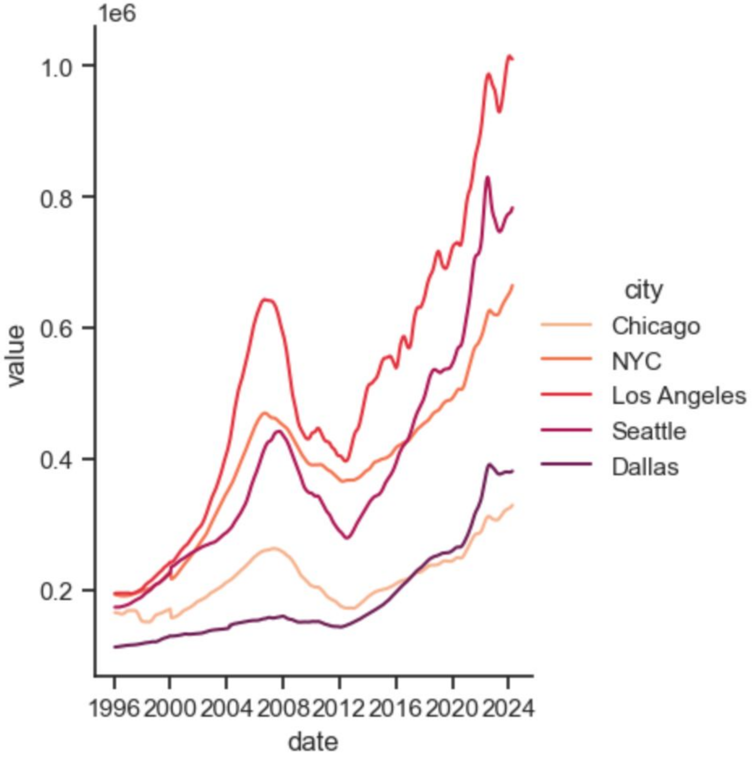
42.6
42.4
42.2
42.0
41.8
41.6
41.4
−88.25
−88.00
−87.75
−87.50
900000
800000
700000
600000
500000
400000
300000
200000
100000



	region_id	value	region_type	zipcode	state	state_region	city	county	latitude	longitude
0	78083	163152.0	zip	46303	IN	Chicago-Naperville-Elgin, IL-IN-WI	Cedar Lake	Lake County	41.378188	-87.446784
8	78095	86862.0	zip	46324	IN	Chicago-Naperville-Elgin, IL-IN-WI	Hammond	Lake County	41.585204	-87.502009
12	81244	119609.0	zip	53144	WI	Chicago-Naperville-Elgin, IL-IN-WI	Somers	Kenosha County	42.624678	-87.875138
13	81256	170217.0	zip	53158	WI	Chicago-Naperville-Elgin, IL-IN-WI	Pleasant Prairie	Kenosha County	42.530150	-87.871865
19	84308	584768.0	zip	60045	IL	Chicago-Naperville-Elgin, IL-IN-WI	Lake Forest	Lake County	42.231059	-87.847344

Seaborn


1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
year
164324.00
113401.25
194627.08
190753.83
174234.67
163806.75
115867.50
196327.33
194178.08
183313.00
153246.75
119286.08
212133.92
202130.00
199082.33
165807.08
125093.00
232251.42
217172.83
216153.50
161329.99
129949.01
252706.29
225838.15
241497.24
175129.47
132114.34
279353.66
253530.33
Chicago
Dallas
Los Angeles
NYC
Seattle
city
1e6
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2


1e6
1.0
0.8
0.6
0.4
0.2
value
1996
2000
2004
2008
2012
2016
2020
2024
date
city
Chicago
NYC
Los Angeles
Seattle
Dallas

Closing Remarks

- Future discovery could be playing with the different indicator IDs to expand search
- Diving into other area regions more in depth
- Researching other market factors to see why certain areas might be priced differently

Thank you for listening!

- Questions?