

CISD41 Introduction to Data Science by Sohair Zaki

Mt San Antonio College

US Real Estate Market and Economics (YEAR 2020)

by Jack Chen and Alec Phong

Overview

1. Purpose & Questions
2. Importing/Clearning Data
 - Importing Libraries, Loading Data, Reading Data
 - Functions, Lambda, and Cleaning
 - Organizing Sub-Datasets
 - Merge Sub-Datasets and create House_df
3. Data Visualization
 - Pivot Tables and Groupby Table
 - Additional Cleaning and Organizing to create State_df
 - House_df
 - Quantitative Data Exploratory
 - Visualization
 - State_df
 - Quantitative Data Exploratory
 - Visualization
4. Testing Hypothesis, ANOVA
 - Pearson Correlation, Z-test, ANOVA, Chisquare
5. Conclusion
 - Questions and Answers
 - Interesting Findings
 - Main Takeaways
6. References

1. Purpose & Questions

Purpose

Real Estate is a huge investment throughout individual's life. This will enhance individual's knowledge of United States real estate market and related economic information.

Learn Python data analysis tools.

Questions

1. What is average and median house price across all USA? (Jack)

Loading [MathJax]/extensions/Safe.js **correlated with Price? If not what's most correlated with Price? (Jack)**

3. What's the primary type of houses people prefer in each state? (Jack)
4. What states have the largest and smallest avg size? (Jack)
5. What states have the highest and lowest \$ perSqFt? (Jack)
6. Find out each State's: Price, Size, perSqft, Annual Income, SavingsRate, Years of Savings to Buy a House(20% down). (Jack)
- 7. Since I live in California, what's California house avg like? (Jack)**
8. Is there Significant relationship between Regions and Price. (Jack)
- 9. Is there huge difference between House Type and Price? (Jack)**

2. Importing/Cleaning Data

Importing Libraries, Loading Data, Reading Data

In [2]:

```
# Import modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
import scipy.stats as st
from scipy.stats import norm
%matplotlib inline

# filter warnings
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

# Display plotly plots offline in notebook mode
init_notebook_mode(connected=True)
```

In [3]:

```
# Loading Data          # FOR YEAR 2020

# "main" data for housing
df = pd.read_csv('data/data_house.csv')

# subdata sets
spending_df = pd.read_csv('data/data_spending.csv')      # need 1 column of spending
gdp_df = pd.read_csv('data/data_gdp.csv')                # need 1 column of gdp per
population_df = pd.read_csv('data/data_population.csv')  # need 1 column of populati
income_df = pd.read_csv('data/data_income.csv')           # need 1 column of income p
```

In [4]:

```
# Read the shape of df
df.shape
```

Out[4]: (85509, 8)

In [5]:

```
# Reading the housing data
df.info()
```

Loading [MathJax]/extensions/Safe.js

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85509 entries, 0 to 85508
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Price        85509 non-null   object  
 1   Address      85509 non-null   object  
 2   Bedrooms     64999 non-null   object  
 3   Bathrooms    69439 non-null   object  
 4   Size          73698 non-null   object  
 5   Sale Status  69960 non-null   object  
 6   URL          85509 non-null   object  
 7   Raw Price    85509 non-null   float64 
dtypes: float64(1), object(7)
memory usage: 5.2+ MB

```

Out[5]:

		Price	Address	Bedrooms	Bathrooms	Size	Sale Status	
85504	\$79,000,000		2 Park Pl, New York, NY 10007	NaN	1 ba	9,680 sqft	Condo for sale	https://www.zillow.com/homedetails/Park-Park-P_35707778_zpid/
85505	\$90,000,000		432 Park Ave #82, New York, NY 10022	6 bds	8 ba	8,054 sqft	Condo for sale	https://www.zillow.com/homedetails/Park-Park_35707779_zpid/
85506	\$95,000,000		1441 Angelo Dr, Los Angeles, CA 90210	NaN	NaN	NaN	Lot / Land for sale	https://www.zillow.com/homedetails/1 Ang Park_35707780_zpid/
85507	\$99,000,000		908 Bel Air Rd, Los Angeles, CA 90077	9 bds	20 ba	34,000 sqft	House for sale	https://www.zillow.com/homedetails/Bel-Bel-Air_35707781_zpid/
85508	\$110,000,000		30 Beverly Park Ter, Beverly Hills, CA 90210	8 bds	12 ba	-- sqft	House for sale	https://www.zillow.com/homedetails/Beve-Beverly-Hills_35707782_zpid/



In [6]:

```
# read df head
df.head()
```

Out[6]:

	Price	Address	Bedrooms	Bathrooms	Size	Sale Status	URL	R Pr
--	-------	---------	----------	-----------	------	-------------	-----	---------

	Price	Address	Bedrooms	Bathrooms	Size	Sale Status	URL	R Pr
0	\$1	Airpark N, Loveland, CO 80538	NaN	NaN	NaN	Lot / Land for sale	https://www.zillow.com/homedetails/Airpark-N-L...	
1	\$1	2940 W Sunset Ave, Springdale, AR 72762	NaN	NaN	NaN	Lot / Land for sale	https://www.zillow.com/homedetails/2940-W-Suns...	
2	\$1	2392 SE Fruit Ave, Port Saint Lucie, FL 34952	3 bds	2 ba	1,649 sqft	NaN	https://www.zillow.com/homedetails/2392-SE-Fru...	
3	\$1	0 SW 38th Ter, Gainesville, FL 32605	NaN	NaN	NaN	Lot / Land for sale	https://www.zillow.com/homedetails/0-SW-38th-T...	
4	\$1	75th St NW, Rochester, MN 55901	NaN	NaN	NaN	NaN	https://www.zillow.com/homedetails/75th-St-NW-...	

Functions, Lambda, and Cleaning

In [7]:

```
# Function #1
# drop NaN, Price, URL, and assign to df1
def metric_deletion(x):
    x.dropna(axis='rows', inplace=True)                      # drop NA
    x = x[x.Bedrooms != '-- bds']                          # drop -- bds
    x = x[x.Bathrooms != '-- ba']                           # drop -- ba
    x = x[x.Size != '-- sqft']                            # drop -- sqft
    x.drop(['URL', 'Price'], axis=1, inplace=True)          # drop URL and Price column since they are not needed
    return x
df1 = metric_deletion(df)
df1.head()                                                 # applying the function to the entire DataFrame
```

Out[7]:

		Address	Bedrooms	Bathrooms	Size	Sale Status	Raw Price
5		3515 W Thompson Rd, Indianapolis, IN 46217	2 bds	1 ba	814 sqft	House for sale	1.0
53		3713 Hillside Ave, Indianapolis, IN 46218	2 bds	1 ba	1,728 sqft	House for sale	775.0
65		1337 W Livingston St APT 1, Allentown, PA 18102	3 bds	1 ba	1,000 sqft	House for sale	1050.0
70		1788 Westwood Dr, Troy, MI 48083	3 bds	2 ba	1,418 sqft	House for sale	1600.0

	Address	Bedrooms	Bathrooms	Size	Sale Status	Raw Price
72	390 Rosado Springs St, Henderson, NV 89014	2 bds	2 ba	1,060 sqft	Townhouse for sale	1700.0

In [8]:

```
# Converting Bathrooms into float
df1.Bathrooms = df1.Bathrooms.str.replace(' ba','').astype('float')      # remove the "
# Converting Bedrooms into float
df1.Bedrooms = df1.Bedrooms.str.replace(' bds','').astype('float')        # remove the "
```

In [9]:

```
# Function #2
# Converting Size to float
def filt_size(s):
    s= s.replace(',', '')
    s = s.replace(' sqft','')
    return float(s)
df1.Size = df1.Size.apply(filt_size)           # convert it to float
                                                # applying the function
```

In [10]:

```
# Splitting Address into Street, City, State, ZipCode, and drop the Address
df1.Address = df1.Address.astype('str')          # c
df1['Street']= df1.Address.apply(lambda x: x.split(', ')[0])      # s
df1['City']= df1.Address.apply(lambda x: x.split(', ')[1])        # s
df1['State']= df1.Address.apply(lambda x: (x.split(', ')[-1]).split(' ')[0])  # s
df1['ZipCode']= df1.Address.apply(lambda x: (x.split(', ')[-1]).split(' ')[1]) # s
# Assign to df2
df2 = df1.drop(['Address'],axis=1)               # s
# Reset the index
df2.reset_index(inplace=True,drop=True)           # r
df2.head()
```

Out[10]:

	Bedrooms	Bathrooms	Size	Sale Status	Raw Price	Street	City	State	ZipCode
0	2.0	1.0	814.0	House for sale	1.0	3515 W Thompson Rd	Indianapolis	IN	46217
1	2.0	1.0	1728.0	House for sale	775.0	3713 Hillside Ave	Indianapolis	IN	46218
2	3.0	1.0	1000.0	House for sale	1050.0	1337 W Livingston St APT 1	Allentown	PA	18102
3	3.0	2.0	1418.0	House for sale	1600.0	1788 Westwood Dr	Troy	MI	48083
4	2.0	2.0	1060.0	Townhouse for sale	1700.0	390 Rosado Springs St	Henderson	NV	89014

In [11]:

```
# Found two rows of abnormal values, so found exact address on google and replace with
df2.loc[28709:28711, 'State']= 'AZ'                      # the two rows are missing state an
df2.loc[28709:28711, 'ZipCode']= '85260'                  # df2.loc[ (index values) , column
```

```
In [12]: # Converting the columns as strings for further cleaning  
df2[['Street','City','State','ZipCode']].astype('str') # since they are object
```

```
Out[12]:
```

	Street	City	State	ZipCode
0	3515 W Thompson Rd	Indianapolis	IN	46217
1	3713 Hillside Ave	Indianapolis	IN	46218
2	1337 W Livingston St APT 1	Allentown	PA	18102
3	1788 Westwood Dr	Troy	MI	48083
4	390 Rosado Springs St	Henderson	NV	89014
...
45394	111 W 57th St PENTHOUSE 72	New York	NY	10019
45395	0 Del Valle Rd	Livermore	CA	94550
45396	1060 Brooklawn Dr	Los Angeles	CA	90077
45397	432 Park Ave #82	New York	NY	10022
45398	908 Bel Air Rd	Los Angeles	CA	90077

45399 rows × 4 columns

```
In [13]: # Finding the weird ZipCode, it is in Canada  
df2.loc[df2.ZipCode == 'N9V']
```

```
Out[13]:
```

	Bedrooms	Bathrooms	Size	Sale Status	Raw Price	Street	City	State	ZipCode
38379	4.0	4.0	2800.0	House for sale	865000.0	349 Benson Ct	Amherstburg	ON	N9V

```
In [14]: # Dropping the canada row  
df2.drop(df2.iloc[38379].name,inplace=True)
```

```
In [15]: # Now the ZipCode can be converted to Integer  
df2.ZipCode = df2.ZipCode.astype('int')
```

```
In [16]: # convert Sale Status into house Types  
# first we make a List  
house_status = list(df2['Sale Status'].unique())  
house_status
```

```
Out[16]: ['House for sale',  
         'Townhouse for sale',  
         'Multifamily home for sale',  
         'Condo for sale',  
         'Home for sale',  
         'sale']
```

Loading [MathJax]/extensions/Safe.js

```
In [17]: # Make a List of house types
house_type = ['House', 'Townhouse', 'Multifamily', 'Condo', 'Others', 'Apartment']

# Replacing house_status with house_type using .map
df2['Sale Status'] = df2['Sale Status'].map(dict(zip(house_status, house_type)))

# Rename column sale status to type
df2.rename(columns={"Sale Status": "Type"}, inplace=True)
```

```
In [18]: # Street column is not useful so we drop it and assign new df3
df3 = df2.drop(['Street'], axis=1)
df3.head()
```

Out[18]:

	Bedrooms	Bathrooms	Size	Type	Raw Price	City	State	ZipCode
0	2.0	1.0	814.0	House	1.0	Indianapolis	IN	46217
1	2.0	1.0	1728.0	House	775.0	Indianapolis	IN	46218
2	3.0	1.0	1000.0	House	1050.0	Allentown	PA	18102
3	3.0	2.0	1418.0	House	1600.0	Troy	MI	48083
4	2.0	2.0	1060.0	Townhouse	1700.0	Henderson	NV	89014

```
In [19]: # Change Raw Price column name to Price
df3.rename(columns={"Raw Price": "Price"}, inplace=True)

# Column - create perSqft column
df3['perSqFt'] = df3.Price / df3.Size
```

```
In [20]: # Visualize the last value counts per state
df3.State.value_counts().tail(5)

# HI NH is below 30 counts, so dropping HI and NH
df3 = df3[(df3['State'] != 'HI') & (df3['State'] != 'NH')]

# drop Type: Others
# because it is not the majority and has weird type of lots/houses/other asset types
df3 = df3[df3['Type'] != 'Others']
```

```
In [21]: # Due to previously dropping rows, reset index again
# West Virginia and Wyoming not in data
df3 = df3.reset_index(drop=True)

# Check how many states we have
len(df3.State.unique())
```

Out[21]: 46

```
In [22]: # Function #3 find_region()
# Assign Regions, also need to drop Wyoming and West Virginia since the dataset does not
Loading [MathJax]/extensions/Safe.js that certain State in the initial DataFrame
# this is created manually
```

```

west = ['CA', 'NV', 'AK', 'WA', 'OR', 'ID', 'MT', 'UT', 'CO', 'AZ', 'NM']
midwest = ['ND', 'WI', 'SD', 'NE', 'KS', 'MN', 'IA', 'MO', 'IL', 'IN', 'OH', 'MI']
north = ['PA', 'NY', 'MA', 'CT', 'ME', 'DC', 'NJ', 'RI']
south = ['TX', 'OK', 'AR', 'LA', 'MS', 'AL', 'TN', 'KY', 'VA', 'MD', 'DE', 'NC', 'SC', 

# List - full state list other than the df
full_state_list = west + midwest + north + south

# Creating function to assign regions
def find_region(state):
    if state in west:
        state = 'West'
    elif state in north:
        state = 'North'
    elif state in south:
        state = 'South'
    elif state in midwest:
        state = 'MidWest'
    return state

# Create Region column by using find_region function
df3['Region'] = df3.State.apply(find_region)

```

In [23]:

```

# Sort the full state list
full_state_list.sort()
full_state_list

# List - states from df
comparator = df3['State'].unique().tolist()
comparator.sort()
comparator

# compare full_state_list and our df-state-list,
compare_dict = dict(zip(full_state_list, comparator))
compare_dict

# Check our list of states and dataframe states are in sync
# df3 is fully cleaned, now we are making a "state_df"

```

Out[23]:

```
{'AK': 'AK',
 'AL': 'AL',
 'AR': 'AR',
 'AZ': 'AZ',
 'CA': 'CA',
 'CO': 'CO',
 'CT': 'CT',
 'DC': 'DC',
 'DE': 'DE',
 'FL': 'FL',
 'GA': 'GA',
 'IA': 'IA',
 'ID': 'ID',
 'IL': 'IL',
 'IN': 'IN',
 'KS': 'KS',
 'KY': 'KY',
 'LA': 'LA',
 'MA': 'MA'}
```

Loading [MathJax]/extensions/Safe.js

```
'ME': 'ME',
```

```
'MI': 'MI',
'MN': 'MN',
'MO': 'MO',
'MS': 'MS',
'MT': 'MT',
'NC': 'NC',
'ND': 'ND',
'NE': 'NE',
'NJ': 'NJ',
'NM': 'NM',
'NV': 'NV',
'NY': 'NY',
'OH': 'OH',
'OK': 'OK',
'OR': 'OR',
'PA': 'PA',
'RI': 'RI',
'SC': 'SC',
'SD': 'SD',
'TN': 'TN',
'TX': 'TX',
'UT': 'UT',
'VA': 'VA',
'WA': 'WA',
'WI': 'WI'}
```

Organizing Sub-Datasets

In [24]:

```
# Read gdp_df
gdp_df.head()
```

Out[24]:

	Unnamed: 0	GeoFips	GeoName	2020
0	0	0	United States	18384687.0
1	1	1000	Alabama	196906.1
2	2	2000	Alaska	50161.0
3	3	4000	Arizona	320550.6
4	4	5000	Arkansas	114943.5

In [25]:

```
# Checking the States in gdp_df
gdp_df.GeoName.unique()
```

Out[25]:

```
array(['United States', 'Alabama', 'Alaska', 'Arizona', 'Arkansas',
       'California', 'Colorado', 'Connecticut', 'Delaware',
       'District of Columbia', 'Florida', 'Georgia', 'Hawaii', 'Idaho',
       'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
       'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
       'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
       'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
       'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
       'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota',
       'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington',
       'West Virginia', 'Wisconsin', 'Wyoming', 'New England', 'Mideast',
       'Great Lakes', 'Plains', 'Southeast', 'Southwest',
       'Rocky Mountain', 'Far West'], dtype=object)
```

```
# Drop the non-used States/Regions and none used attributes
gdp_df = gdp_df[(gdp_df.GeoName != 'United States') & (gdp_df.GeoName != 'Southwest') &

# rename the columns
gdp_df.rename(columns={'GeoName':'State','2020':'GDP'},inplace=True)

# pick out the columns we need
gdp_df = gdp_df[['State', 'GDP']]

# reset index
gdp_df.reset_index(inplace=True,drop=True)

# reading last part to see if index match with ours
gdp_df.tail()
```

Out[26]:	State	GDP
42	Utah	171370.1
43	Vermont	28648.5
44	Virginia	473817.5
45	Washington	532861.9
46	Wisconsin	291715.8

```
In [27]: # Read population_df  
population_df.head()
```

Out[27]:	rank	State	Pop	Growth	Pop2018	Pop2010	growthSince2010	Percent	density
0	1	California	39613493	0.0038	39461588	37319502	0.0615	0.1184	254.2929
1	2	Texas	29730311	0.0385	28628666	25241971	0.1778	0.0889	113.8081
2	3	Florida	21944577	0.0330	21244317	18845537	0.1644	0.0656	409.2229
3	4	New York	19299981	-0.0118	19530351	19399878	-0.0051	0.0577	409.5400
4	5	Pennsylvania	12804123	0.0003	12800922	12711160	0.0073	0.0383	286.1704

```
In [28]: # Checking states in population_df  
population_df.State.unique()
```

```
Out[28]: array(['California', 'Texas', 'Florida', 'New York', 'Pennsylvania',
   'Illinois', 'Ohio', 'Georgia', 'North Carolina', 'Michigan',
   'New Jersey', 'Virginia', 'Washington', 'Arizona', 'Tennessee',
   'Massachusetts', 'Indiana', 'Missouri', 'Maryland', 'Colorado',
   'Wisconsin', 'Minnesota', 'South Carolina', 'Alabama', 'Louisiana',
   'Kentucky', 'Oregon', 'Oklahoma', 'Connecticut', 'Utah',
   'Puerto Rico', 'Nevada', 'Iowa', 'Arkansas', 'Mississippi',
   'Kansas', 'New Mexico', 'Nebraska', 'Idaho', 'West Virginia',
   'Hawaii', 'New Hampshire', 'Maine', 'Montana', 'Rhode Island',
   'Delaware', 'South Dakota', 'North Dakota', 'Alaska',
   'District of Columbia', 'Vermont', 'Wyoming'], dtype=object)
```

```

population_df = population_df[(population_df.State != 'Wyoming') & (population_df.State

# picking out the columns we want
population_df = population_df[['State', 'Pop']]

# sort by state
population_df.sort_values(by=['State'], inplace=True)

# replace index
population_df.reset_index(inplace=True, drop=True)

# read last part of df
population_df.tail()

```

Out[29]:

	State	Pop
42	Utah	3310774
43	Vermont	623251
44	Virginia	8603985
45	Washington	7796941
46	Wisconsin	5852490

In [30]:

```

# Check GeoName/States
income_df.GeoName.unique()

```

Out[30]:

```

array(['United States', 'Alabama', 'Alaska *', 'Arizona', 'Arkansas',
       'California', 'Colorado', 'Connecticut', 'Delaware',
       'District of Columbia', 'Florida', 'Georgia', 'Hawaii *', 'Idaho',
       'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
       'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
       'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
       'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
       'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
       'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota',
       'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington',
       'West Virginia', 'Wisconsin', 'Wyoming', 'New England', 'Mideast',
       'Great Lakes', 'Plains', 'Southeast', 'Southwest',
       'Rocky Mountain', 'Far West'], dtype=object)

```

In [31]:

```

# found astrious * in States, such as hawaii, so made a function to remove it
def conv_state(a):
    if '*' in a:
        a=a.replace(' *', '')
    return a

# rename the columns
income_df.rename(columns={'GeoName':'State', '2020':'income'}, inplace=True)

# pick the used columns
income_df = income_df[['State', 'income']]

# convert into string type
income_df.State.astype('str')

```

Loading [MathJax]/extensions/Safe.js
 # upcycling the conv_state function

```

income_df.State = income_df.State.apply(conv_state)

# Drop unused States/Region, rename GeoName to State and 2020 to income, filter out other
income_df = income_df[(income_df.State != 'United States') & (income_df.State != 'South

# reset index
income_df.reset_index(inplace=True, drop=True)

# read the tail of df
income_df.tail()

```

Out[31]:

	State	income
42	Utah	169656.3
43	Vermont	36894.2
44	Virginia	532256.2
45	Washington	516441.1
46	Wisconsin	324252.0

In [32]:

```
# Reading spending_df
spending_df.head(15)
```

Out[32]:

	Unnamed: 0	GeoFips	GeoName	LineCode	Description	2020
0	0	0	United States	1	Personal consumption expenditures	14047565.0
1	1	0	United States	2	Goods	4653822.0
2	2	0	United States	3	Durable goods	1616408.0
3	3	0	United States	4	Motor vehicles and parts	541265.0
4	4	0	United States	5	Furnishings and durable household equipment	390382.0
5	5	0	United States	6	Recreational goods and vehicles	476217.0
6	6	0	United States	7	Other durable goods	208545.0
7	7	0	United States	8	Nondurable goods	3037413.0
8	8	0	United States	9	Food and beverages purchased for off-pre...	1146676.0
9	9	0	United States	10	Clothing and footwear	362435.0
10	10	0	United States	11	Gasoline and other energy goods	246757.0

Unnamed: 0	GeoFips	GeoName	LineCode	Description	2020
11	11	0	United States	12	Other nondurable goods 1281545.0
12	12	0	United States	13	Services 9393744.0
13	13	0	United States	14	Household consumption expenditures (for se... 8872868.0
14	14	0	United States	15	Housing and utilities 2668144.0

In [33]:

```
# Personal consumption expenditure is the total of the sub categories, so we only want
spending_df = spending_df[spending_df['Description'] == 'Personal consumption expenditure']

# Dropping un-used region/State
spending_df = spending_df[(spending_df.GeoName != 'United States') & (spending_df.GeoNa

# Rename column names
spending_df.rename(columns={'GeoName': 'State', '2020': 'spending'}, inplace=True)

# Filter out needed columns
spending_df = spending_df[['State', 'spending']]

# Reset index
spending_df.reset_index(inplace=True, drop=True)

# reading last 5 rows of column
spending_df.tail()
```

Out[33]:

	State	spending
42	Utah	121445.4
43	Vermont	29544.8
44	Virginia	367302.7
45	Washington	354219.1
46	Wisconsin	238923.0

1. Merge the Sub-Datasets for future use
2. Create main - house_df

In [34]:

```
# Making a merge_df and merge all the data by state
merged_df = gdp_df.merge(spending_df, how = 'inner', on = 'State')
merged_df = merged_df.merge(population_df, how = 'inner', on = 'State')
merged_df = merged_df.merge(income_df, how = 'inner', on = 'State')

# Renaming columns, renaming to fullState is because we will insert abbreviations later
merged_df.rename(columns={'State': 'fullState', 'spending': 'Spending', 'Pop': 'Population'}
merged_df.tail()
```

Loading [MathJax]/extensions/Safe.js

Out[34]:

	fullState	GDP	Spending	Population	Income
42	Utah	171370.1	121445.4	3310774	169656.3
43	Vermont	28648.5	29544.8	623251	36894.2
44	Virginia	473817.5	367302.7	8603985	532256.2
45	Washington	532861.9	354219.1	7796941	516441.1
46	Wisconsin	291715.8	238923.0	5852490	324252.0

In [35]:

```
# sort df3 by state, reset index and set to our Final house_df
df3= df3.sort_values(by='State')

# reset index
df3.reset_index(inplace=True,drop=True)

# Drop - zipcode since we dont need it
df4=df3.drop('ZipCode', axis=1)

# assign a new df
df5 = df4
df5.head()
```

Out[35]:

	Bedrooms	Bathrooms	Size	Type	Price	City	State	perSqFt	Region
0	8.0	4.0	3264.0	Multifamily	459900.0	Anchorage	AK	140.900735	West
1	2.0	1.0	778.0	Condo	89900.0	Anchorage	AK	115.552699	West
2	8.0	4.0	3880.0	Multifamily	590000.0	Eagle River	AK	152.061856	West
3	6.0	5.0	4640.0	House	589900.0	Chugiak	AK	127.133621	West
4	2.0	2.0	1152.0	Condo	184000.0	Anchorage	AK	159.722222	West

In [36]:

```
# removing Vermont from df since we dont need that, now we have 46 instead of 47 in the
merged_df = merged_df[(merged_df['fullState'] != 'Vermont')]

# List - making a state list from our main df
list1 = list(df5.State.unique())

# List - making a state list from sub dataset
list2 = list(merged_df.fullState)

# Dict - make dict from list1 and list2
state_dict = dict(zip(list1,list2))

# read and see if they are correct
state_dict
```

Out[36]:

```
{'AK': 'Alabama',
'AL': 'Alaska',
'AR': 'Arizona',
'AZ': 'Arkansas',
'CA': 'California',
```

Loading [MathJax]/extensions/Safe.js

```
'CT': 'Connecticut',
'DC': 'Delaware',
'DE': 'District of Columbia',
'FL': 'Florida',
'GA': 'Georgia',
'IA': 'Idaho',
'ID': 'Illinois',
'IL': 'Indiana',
'IN': 'Iowa',
'KS': 'Kansas',
'KY': 'Kentucky',
'LA': 'Louisiana',
'MA': 'Maine',
'MD': 'Maryland',
'ME': 'Massachusetts',
'MI': 'Michigan',
'MN': 'Minnesota',
'MO': 'Mississippi',
'MS': 'Missouri',
'MT': 'Montana',
'NC': 'Nebraska',
'ND': 'Nevada',
'NE': 'New Jersey',
'NJ': 'New Mexico',
'NM': 'New York',
'NV': 'North Carolina',
'NY': 'North Dakota',
'OH': 'Ohio',
'OK': 'Oklahoma',
'OR': 'Oregon',
'PA': 'Pennsylvania',
'RI': 'Rhode Island',
'SC': 'South Carolina',
'SD': 'South Dakota',
'TN': 'Tennessee',
'TX': 'Texas',
'UT': 'Utah',
'VA': 'Virginia',
'WA': 'Washington',
'WI': 'Wisconsin'}
```

```
In [37]: # since many are not correct, we update them
# state_dict will be used later
state_dict.update({'AK': 'Alaska',
                   'AL': 'Alabama',
                   'AR': 'Arkansas',
                   'AZ': 'Arizona',
                   'CA': 'California',
                   'CO': 'Colorado',
                   'CT': 'Connecticut',
                   'DC': 'District of Columbia',
                   'DE': 'Delaware',
                   'FL': 'Florida',
                   'GA': 'Georgia',
                   'IA': 'Iowa',
                   'ID': 'Idaho',
                   'IL': 'Illinois',
                   'IN': 'Indiana',
                   'KS': 'Kansas',
                   'KY': 'Kentucky',
                   'LA': 'Louisiana',
                   'MA': 'Massachusetts',
                   'MD': 'Maryland',
                   'ME': 'Maine',
                   'MI': 'Michigan',
                   'MN': 'Minnesota',
                   'MO': 'Mississippi',
                   'MS': 'Missouri',
                   'MT': 'Montana',
                   'NC': 'North Carolina',
                   'ND': 'North Dakota',
                   'NE': 'New Hampshire',
                   'NJ': 'New Jersey',
                   'NM': 'New Mexico',
                   'NV': 'New York',
                   'NY': 'New York',
                   'OH': 'Ohio',
                   'OK': 'Oklahoma',
                   'OR': 'Oregon',
                   'PA': 'Pennsylvania',
                   'RI': 'Rhode Island',
                   'SC': 'South Carolina',
                   'SD': 'South Dakota',
                   'TN': 'Tennessee',
                   'TX': 'Texas',
                   'UT': 'Utah',
                   'VA': 'Virginia',
                   'WA': 'Washington',
                   'WI': 'Wisconsin'})
```

```
'MD': 'Maryland',
'ME': 'Maine',
'MI': 'Michigan',
'MN': 'Minnesota',
'MO': 'Missouri',
'MS': 'Mississippi',
'MT': 'Montana',
'NC': 'North Carolina',
'ND': 'North Dakota',
'NE': 'Nebraska',
'NJ': 'New Jersey',
'NM': 'New Mexico',
'NV': 'Nevada',
'NY': 'New York',
'OH': 'Ohio',
'OK': 'Oklahoma',
'OR': 'Oregon',
'PA': 'Pennsylvania',
'RI': 'Rhode Island',
'SC': 'South Carolina',
'SD': 'South Dakota',
'TN': 'Tennessee',
'TX': 'Texas',
'UT': 'Utah',
'VA': 'Virginia',
'WA': 'Washington',
'WI': 'Wisconsin'}, inplace=True)
```

In [38]:

```
# Creating a for Loop to remove outliers based on Price

# list of states in df
state_list = list(df5['State'].unique())

# create empty dataframe
df6 = pd.DataFrame(columns=df5.columns)

# Loop each state from state_list
for a in state_list:
    df = df5[df5['State'] == a]
    # calculating quantiles and IQR
    q1 = df['Price'].quantile(0.25)
    q3 = df['Price'].quantile(0.75)
    iqr = q3-q1
    fence_low = q1-1.5*iqr
    fence_high = q3+1.5*iqr
    # filtering out the outliers
    df_out = df.loc[(df['Price'] > fence_low) & (df['Price'] < fence_high)]
    # append to the empty df6
    df6 = df6.append(df_out)
```

In [39]:

```
# Creating a for Loop to remove outliers based on perSqFt
house_df = pd.DataFrame(columns=df6.columns)
for a in state_list:
    df = df6[df6['State'] == a]
    # calculating quantiles and IQR
    q1 = df['perSqFt'].quantile(0.25)
    q3 = df['perSqFt'].quantile(0.75)
```

Loading [MathJax]/extensions/Safe.js

```

iqr = q3-q1
fence_low = q1-1.5*iqr
fence_high = q3+1.5*iqr
# filtering out the outliers
df_out = df.loc[(df['perSqFt'] > fence_low) & (df['perSqFt'] < fence_high)]
# append to empty df house_df
house_df = house_df.append(df_out)

# Filter out size over 15,000 square feet
house_df = house_df[house_df.Size < 15000]

# Produce the main house_df and reset index
house_df = house_df.reset_index(drop=True)

```

In [40]:

```

# # Function to calculate z-score for each row.

# def create_val(a,b):
#     df = house_df[house_df['State'] == b]
#     avg = df['perSqFt'].mean()
#     std = df['perSqFt'].std()
#     z = (a - avg) / std
#     return z

# # creating z_score by applying function with Lambda
# house_df['z_score'] = house_df.apply(lambda x: create_val(x.perSqFt, x.State), axis =
# house_df = house_df[(house_df.z_score < 3) & (house_df.z_score > -3)]

```

In [41]:

```

# Function to assign Size into categories
def find_sizeCat(size):
    if size > 4500:
        size = "4500 and higher"
    elif 3501 < size <= 4500:
        size = "3501 to 4500"
    elif 2501 < size < 3500:
        size = "2501 to 3500"
    elif 1501 < size <= 2500:
        size = "1501 to 2500"
    elif 801 < size <= 1500:
        size = "801 to 1500"
    else:
        size = "800 and under"
    return size
house_df["SizeCat"] = house_df.Size.apply(find_sizeCat)

```

3. Data Visualization

Pivot Tables and Groupby Table

In [42]:

```

# Pivot Table #1 - by Region

# setting format display to float with 2 decimal places
pd.set_option('display.float_format', '{:.2f}'.format)

# by region (big picture/general overview), shows the region's average Bedrooms, Bathr
Loading [MathJax]/extensions/Safe.js house_df, values=['perSqFt','Size','Bedrooms','Bathrooms','Price'], inde

```

Out[42]:

	Bathrooms	Bedrooms	Price	Size	perSqFt
Region					
MidWest	2.11	3.16	232389.06	1851.80	126.30
North	2.26	3.50	458599.63	1793.98	281.09
South	2.51	3.26	313300.11	2026.79	156.33
West	2.59	3.46	624722.63	2052.39	326.17

West Region average Price, Size, and perSqFt are highest

In [43]:

```
# Pivot Tabel #2 - by State

# setting format display to float with 2 decimal places
pd.set_option('display.float_format', '{:.2f}'.format)

# by state, shows the region's average Bedrooms, Batherooms, Price, Size, and per SqFt
pivot_state = pd.pivot_table(house_df, values=['perSqFt','Size','Bedrooms','Bathrooms'],
pivot_state.head(15)
```

Out[43]:

	Bathrooms	Bedrooms	Price	Size	perSqFt	
Region	State					
MidWest	IA	1.95	3.13	149252.38	1686.97	90.80
	IL	2.24	2.99	268614.69	1756.19	159.48
	IN	2.13	3.09	216826.92	2076.28	98.88
	KS	2.55	3.41	259040.19	2287.53	108.62
	MI	1.81	3.13	142267.57	1587.69	86.42
	MN	2.18	3.11	369010.68	1850.69	205.52
	MO	2.23	2.97	238783.35	1944.31	123.43
	ND	2.22	3.58	259991.39	2286.61	112.95
	NE	2.45	3.38	276671.76	2150.49	127.82
	OH	1.84	3.35	134835.39	1721.55	79.80
North	SD	2.40	3.44	362883.80	2031.12	181.82
	WI	2.32	3.25	297997.15	1906.92	161.82
	CT	2.57	4.15	365254.44	2359.92	177.04
	DC	2.47	2.78	732301.21	1525.06	500.22
South	MA	2.37	3.44	800828.82	1826.08	501.47

In [44]:

```
# Made a groupby table so data can be viewed by Region then State
house_df.groupby(['Region','State']).mean()
```

Loading [MathJax]/extensions/Safe.js

Out[44]:

		Bedrooms	Bathrooms	Size	Price	perSqFt
Region	State					
MidWest	IA	3.13	1.95	1686.97	149252.38	90.80
	IL	2.99	2.24	1756.19	268614.69	159.48
	IN	3.09	2.13	2076.28	216826.92	98.88
	KS	3.41	2.55	2287.53	259040.19	108.62
	MI	3.13	1.81	1587.69	142267.57	86.42
	MN	3.11	2.18	1850.69	369010.68	205.52
	MO	2.97	2.23	1944.31	238783.35	123.43
	ND	3.58	2.22	2286.61	259991.39	112.95
	NE	3.38	2.45	2150.49	276671.76	127.82
	OH	3.35	1.84	1721.55	134835.39	79.80
North	SD	3.44	2.40	2031.12	362883.80	181.82
	WI	3.25	2.32	1906.92	297997.15	161.82
	CT	4.15	2.57	2359.92	365254.44	177.04
	DC	2.78	2.47	1525.06	732301.21	500.22
	MA	3.44	2.37	1826.08	800828.82	501.47
	ME	2.83	2.14	1673.86	713430.86	437.12
	NJ	3.86	2.28	1780.86	381615.45	240.69
	NY	3.56	2.19	1771.99	393834.38	232.72
	PA	3.22	2.13	1633.04	279879.74	177.84
	RI	3.14	1.98	1750.53	305050.04	191.57
South	AL	3.38	2.62	2319.30	305048.89	124.37
	AR	3.64	2.96	2804.81	337014.73	117.71
	DE	3.11	1.83	1537.86	192903.10	121.23
	FL	2.94	2.28	1596.70	322036.59	201.74
	GA	3.32	2.41	1969.07	253325.04	125.18
	KY	3.32	2.77	2387.65	394384.46	165.15
	LA	3.28	2.32	2040.95	281259.07	123.74
	MD	2.98	2.28	1577.91	240832.79	157.61
	MS	3.30	2.29	1907.25	181302.03	90.56
	NC	3.26	2.84	2235.15	478810.32	221.57
SC	OK	3.41	2.57	2511.43	354887.77	131.90
	SC	3.48	2.65	2039.93	378263.16	185.20

		Bedrooms	Bathrooms	Size	Price	perSqFt
Region	State					
	TN	3.26	2.58	2130.56	427594.91	203.61
	TX	3.42	2.62	2170.51	277502.48	128.98
	VA	3.23	2.52	1806.27	291610.56	161.60
West	AK	4.59	2.97	2392.84	363202.40	165.54
	AZ	3.34	2.51	2070.79	466962.46	224.66
	CA	3.44	2.56	1965.24	776273.37	424.09
	CO	3.26	2.64	2114.61	581882.04	289.85
	ID	3.91	2.75	2321.74	571593.22	253.87
	MT	3.45	2.48	2464.15	417416.81	168.74
	NM	3.37	2.46	2096.72	315167.86	152.91
	NV	3.32	2.71	1999.47	400673.05	202.44
	OR	3.44	2.83	2299.38	556123.87	250.94
	UT	4.15	2.73	2524.02	470316.15	200.74
	WA	3.60	2.49	2240.48	485569.83	230.23

Additional Cleaning and Organizing to create State_df

In [45]:

```
# count house data per state in house_df

# Create a column for house counts per State by making a valuecount_df
valuecount_df = pd.DataFrame(house_df['State'].value_counts())

# reset index
valuecount_df2 = valuecount_df.reset_index()

# rename column
valuecount_df2 = valuecount_df2.rename(columns={"State": "HouseCount", 'index':'State'})

# sort by state
valuecount_df2 = valuecount_df2.sort_values(by = "State")

# reset index
valuecount_df2 = valuecount_df2.reset_index(drop=True)
```

In [46]:

```
# Creating state_df, Step #1, from pivot table

# Converting pivot_state into DF and reset index
pivot_df =pd.DataFrame(pivot_state)
pivot_df = pivot_df.reset_index()

# create function to insert fullState name for merge
def insert_fullState(a):
    Loading [MathJax]/extensions/Safe.js ict[a]
    return a
```

```

# Merge with pivot df
pivot_df = pivot_df.merge(valuecount_df2, how = 'inner', on = 'State')

# insert both fullState and Region
pivot_df['fullState'] = pivot_df.State.apply(insert_fullState)
pivot_df['Region'] = pivot_df.State.apply(find_region)

# insert fullState into house_df
house_df['fullState'] = house_df.State.apply(insert_fullState)

# read pivot df
pivot_df.head(15)

```

Out[46]:

	Region	State	Bathrooms	Bedrooms	Price	Size	perSqFt	HouseCount	fullState
0	MidWest	IA	1.95	3.13	149252.38	1686.97	90.80	117	Iowa
1	MidWest	IL	2.24	2.99	268614.69	1756.19	159.48	1467	Illinois
2	MidWest	IN	2.13	3.09	216826.92	2076.28	98.88	990	Indiana
3	MidWest	KS	2.55	3.41	259040.19	2287.53	108.62	283	Kansas
4	MidWest	MI	1.81	3.13	142267.57	1587.69	86.42	871	Michigan
5	MidWest	MN	2.18	3.11	369010.68	1850.69	205.52	848	Minnesota
6	MidWest	MO	2.23	2.97	238783.35	1944.31	123.43	343	Missouri
7	MidWest	ND	2.22	3.58	259991.39	2286.61	112.95	151	North Dakota
8	MidWest	NE	2.45	3.38	276671.76	2150.49	127.82	266	Nebraska
9	MidWest	OH	1.84	3.35	134835.39	1721.55	79.80	1182	Ohio
10	MidWest	SD	2.40	3.44	362883.80	2031.12	181.82	25	South Dakota
11	MidWest	WI	2.32	3.25	297997.15	1906.92	161.82	320	Wisconsin
12	North	CT	2.57	4.15	365254.44	2359.92	177.04	406	Connecticut
13	North	DC	2.47	2.78	732301.21	1525.06	500.22	102	District of Columbia
14	North	MA	2.37	3.44	800828.82	1826.08	501.47	1068	Massachusetts

In [47]:

```

# Creating state_df, Step #2, now we have both state_df and house_df
state_df = merged_df.merge(pivot_df, how = 'inner', on = 'fullState')
state_df.head()

```

Out[47]:

	fullState	GDP	Spending	Population	Income	Region	State	Bathrooms	Bedrooms
0	Alabama	196906.10	176479.80	4934193	228748.80	South	AL	2.62	3.38 30%
1	Alaska	50161.00	35635.70	724357	46430.30	West	AK	2.97	4.59 36%
2	Arizona	320550.60	287090.10	7520103	368458.60	West	AZ	2.51	3.34 46%
3	Arkansas	114943.50	104488.80	3033946	143147.90	South	AR	2.96	3.64 33%

Loading [MathJax]/extensions/Safe.js 65.90 1835980.60 39613493 2763312.00 West CA 2.56 3.44 77%

In [48]:

```
# Column - calculate savings rate per state
state_df['SavingsRate'] = (state_df['Income']-state_df['Spending'])/state_df['Income']
state_df.head()
```

Out[48]:

	fullState	GDP	Spending	Population	Income	Region	State	Bathrooms	Bedrooms
0	Alabama	196906.10	176479.80	4934193	228748.80	South	AL	2.62	3.38 305
1	Alaska	50161.00	35635.70	724357	46430.30	West	AK	2.97	4.59 365
2	Arizona	320550.60	287090.10	7520103	368458.60	West	AZ	2.51	3.34 466
3	Arkansas	114943.50	104488.80	3033946	143147.90	South	AR	2.96	3.64 337
4	California	2663665.90	1835980.60	39613493	2763312.00	West	CA	2.56	3.44 776

In [49]:

```
# Calculate year of savings to buy a house with 20% down payment
state_df['Years_of_Savings'] = (state_df['Price'] * 0.2) / ((state_df['Income'] - state_df['Spending']))
state_df.sort_values(by='Years_of_Savings', ascending=True).head(5)
```

Out[49]:

	fullState	GDP	Spending	Population	Income	Region	State	Bathrooms	Bedrooms
14	Iowa	169420.30	118904.60	3167974	169181.60	MidWest	IA	1.95	3.13 1
33	Ohio	589897.70	466159.70	11714618	627231.30	MidWest	OH	1.84	3.35 1
19	Maryland	353052.50	268456.80	6065436	404520.70	South	MD	2.28	2.98 2
21	Michigan	445682.60	411364.20	9992427	530808.60	MidWest	MI	1.81	3.13 1
6	Connecticut	235888.60	179405.90	3552821	279612.40	North	CT	2.57	4.15 3

In [50]:

```
# Calculate per Annual Income
state_df['perAnnualIncome'] = (state_df['Income'] * 1000000 / state_df['Population'])
state_df.sort_values(by='perAnnualIncome', ascending=True).head(5)
```

Out[50]:

	fullState	GDP	Spending	Population	Income	Region	State	Bathrooms	Bedrooms
23	Mississippi	99667.50	95998.30	2966407	124988.20	South	MS	2.29	3.30 181
0	Alabama	196906.10	176479.80	4934193	228748.80	South	AL	2.62	3.38 305
29	New Mexico	92696.50	74276.40	2105005	97603.50	West	NM	2.46	3.37 315
3	Arkansas	114943.50	104488.80	3033946	143147.90	South	AR	2.96	3.64 337
16	Kentucky	185535.10	163749.90	4480713	211947.60	South	KY	2.77	3.32 394

In [51]:

```
#####
```

Loading [MathJax]/extensions/Safe.js

```
# Mortgage Amount - Calculating avg mortgage amount with avg price  
state_df['mortAmount'] = state_df['Price']*8
```

In [82]: # NOTE: We assume 20% down payment, 30yr Loan, 4% interest for whole project

```
# Variables  
i = 0.04          # interest rate  
yrs = 30          # years of mortgage  
mth = 12          # month per year  
  
divided_rate = (i/12)    #monthly rates  
thirty_total = yrs*mth  # total number of months  
  
# Column - Calculating mortgage payment per month for each state  
state_df['mortPayment'] = (state_df['mortAmount']*((i/12)*(1+(i/12))**yrs*mth)))
```

In [69]: # Desired Annual Income - income calculated using mortgage,
to live comfortably, paying off mortgage, have savings, and money to spend.
state_df['desiredAnnualIncome'] = (((state_df['mortPayment'] / 20) * 100) * 12) / 2

In [73]: # Column - Calculate Annual Income After Mortgage
state_df['annualIncomeAfterMortgage'] = state_df['perAnnualIncome'] - ((state_df['mortP']

House_Df

Descriptive Statistics

In [53]: house_df.head()

	Bedrooms	Bathrooms	Size	Type	Price	City	State	perSqFt	Region	SizeCat
0	8.00	4.00	3264.00	Multifamily	459900.00	Anchorage	AK	140.90	West	2501 to 3500
1	2.00	1.00	778.00	Condo	89900.00	Anchorage	AK	115.55	West	800 and under
2	8.00	4.00	3880.00	Multifamily	590000.00	Eagle River	AK	152.06	West	3501 to 4500
3	6.00	5.00	4640.00	House	589900.00	Chugiak	AK	127.13	West	4500 and higher
4	2.00	2.00	1152.00	Condo	184000.00	Anchorage	AK	159.72	West	801 to 1500

In [54]: house_df.describe()

	Bedrooms	Bathrooms	Size	Price	perSqFt
count	38479.00	38479.00	38479.00	38479.00	38479.00
mean	3.33	2.43	1973.83	407007.46	215.65
std	1.31	1.08	1036.53	326193.75	159.11
min	2.00	1.00	397.00	1.00	0.00
25%	3.00	2.00	1243.00	190000.00	118.15
50%	3.00	2.00	1701.00	324900.00	172.27
75%	4.00	3.00	2405.00	525000.00	256.68
max	84.00	16.00	13776.00	2500000.00	1293.53

Across US

Median are: 3 Bedrooms, 2 Bathrooms, 1702 SqFt, Price of 325,000

Average are: 3.3 Bedrooms, 2.5 Bathrooms, 2000 SqFt, Price of 407,000

In [55]:

```
# Correlation - of variables
house_df.corr()
```

Out[55]:

	Bedrooms	Bathrooms	Size	Price	perSqFt
Bedrooms	1.00	0.55	0.62	0.29	-0.10
Bathrooms	0.55	1.00	0.76	0.49	0.03
Size	0.62	0.76	1.00	0.48	-0.11
Price	0.29	0.49	0.48	1.00	0.73
perSqFt	-0.10	0.03	-0.11	0.73	1.00

Price and Bedrooms have correlation of 29%

Price and Size have correlation of 30%

Price and Bathrooms have correlation of 49%

Bedroom and Bathrooms have correlation of 56%

Price and Bathrooms have a higher correlation than Price and Bedrooms.

So the higher the price the number Bathrooms will increase faster than bedrooms you will have.
BAM!!!

In [85]:

```
#####
#filter the house_df table by creating a separate table where price is in the range of
houseFilter = house_df[house_df['Type'] == 'House']
houseMin = 55000 * 2.5
houseMax = 55000 * 3
houseRangeMin = houseFilter[houseFilter['Price'] >= houseMin]
houseRanges = houseRangeMin[houseRangeMin['Price'] <= houseMax]
houseRanges.head(2)
```

Out[85]:

	Bedrooms	Bathrooms	Size	Type	Price	City	State	perSqFt	Region	SizeCat	fullS
9	2.00	1.00	836.00	House	140000.00	Anchorage	AK	167.46	West	801 to 1500	All
12	3.00	2.00	830.00	House	165000.00	Anchorage	AK	198.80	West	801 to 1500	All

In [84]:

```
#using the monthly payment formula, grab the table price and apply the function.
twentyPercentDown = houseRanges['Price'] * .8
indMonthlyPayment = (twentyPercentDown * (divided_rate * (1 + divided_rate)**thirty_tot
indMonthlyPayment.head(2)
```

Out[84]:

9	534.71
12	630.19

Name: Price, dtype: float64

In [86]:

```
#add estimated montly morgage payment
houseRanges['accuredInterestPayMonth'] = indMonthlyPayment
houseRanges.head(2)
```

/var/folders/fy/_jcf8bdn3hbccf_yk638qvkw0000gn/T/ipykernel_16201/3317195810.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

Out[86]:

	Bedrooms	Bathrooms	Size	Type	Price	City	State	perSqFt	Region	SizeCat	fullS
9	2.00	1.00	836.00	House	140000.00	Anchorage	AK	167.46	West	801 to 1500	All
12	3.00	2.00	830.00	House	165000.00	Anchorage	AK	198.80	West	801 to 1500	All

House_df visualizations

In [56]:

```
# Setting Theme
sns.set_theme(style="whitegrid") # all charts will have a light grid

# Function to convert x to array, y to the count %
def ecdf(data):
    n = len(data)
    x = np.sort(data)
    y = np.arange(1,1+n)/n
    return x, y

# Figure size and axis
fig = plt.figure(figsize=(20,10))
Loading [MathJax]/extensions/Safe.js axes([0.1,0.1,0.8,0.8])
```

```

# Calculate x and y
x_1, y_1 = ecdf(house_df[ 'Bedrooms' ])
x_2, y_2 = ecdf(house_df[ 'Bathrooms' ])

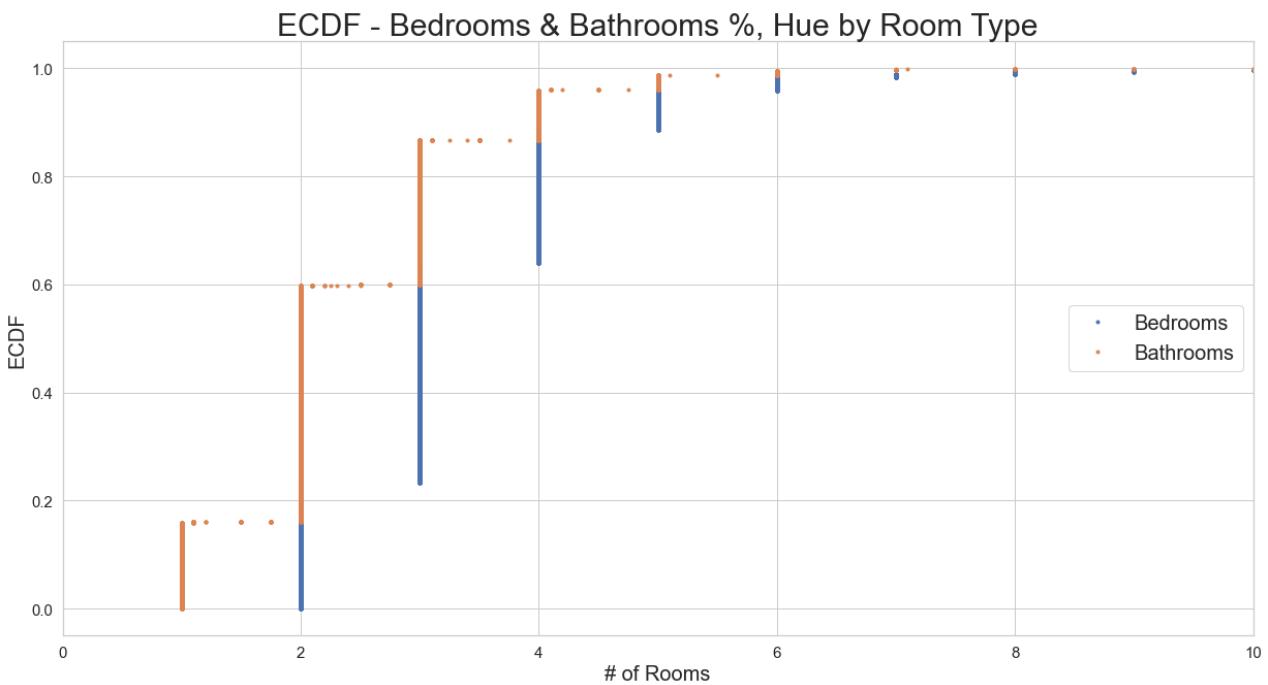
# Plot ECDFs
axes.plot(x_1,y_1,marker = '.',linestyle='none')
axes.plot(x_2,y_2,marker = '.',linestyle='none')

# Legend
plt.legend(( 'Bedrooms', 'Bathrooms'),loc=7, prop={'size': 20})

# Annotation, Label, Tick, Title
plt.xlabel('# of Rooms', fontsize=20)
plt.ylabel('ECDF', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('ECDF - Bedrooms & Bathrooms %, Hue by Room Type', fontsize=30)
# Set limit for x
axes.set_xlim(0,10)

# Show plot
plt.show()

```



According to ECDF, there are more Bedrooms than Bathrooms.

In [57]:

```

# Figure size and axis
fig = plt.figure(figsize=(20,10))
axes = fig.add_axes([0.1,0.1,0.8,0.8])

# Calculate x and y
x_1, y_1 = ecdf(house_df[house_df['Type'] == 'House'][ 'Price'])
x_2, y_2 = ecdf(house_df[house_df['Type'] == 'Townhouse'][ 'Price'])
x_3, y_3 = ecdf(house_df[house_df['Type'] == 'Condo'][ 'Price'])
x_4, y_4 = ecdf(house_df[house_df['Type'] == 'Apartment'][ 'Price'])
x_5, y_5 = ecdf(house_df[house_df['Type'] == 'Multifamily'][ 'Price'])

```

Loading [MathJax]/extensions/Safe.js

rcul LCDIS

```

axes.plot(x_1,y_1,marker = '.',linestyle='none')
axes.plot(x_2,y_2,marker = '.',linestyle='none')
axes.plot(x_3,y_3,marker = '.',linestyle='none')
axes.plot(x_4,y_4,marker = '.',linestyle='none')
axes.plot(x_5,y_5,marker = '.',linestyle='none')

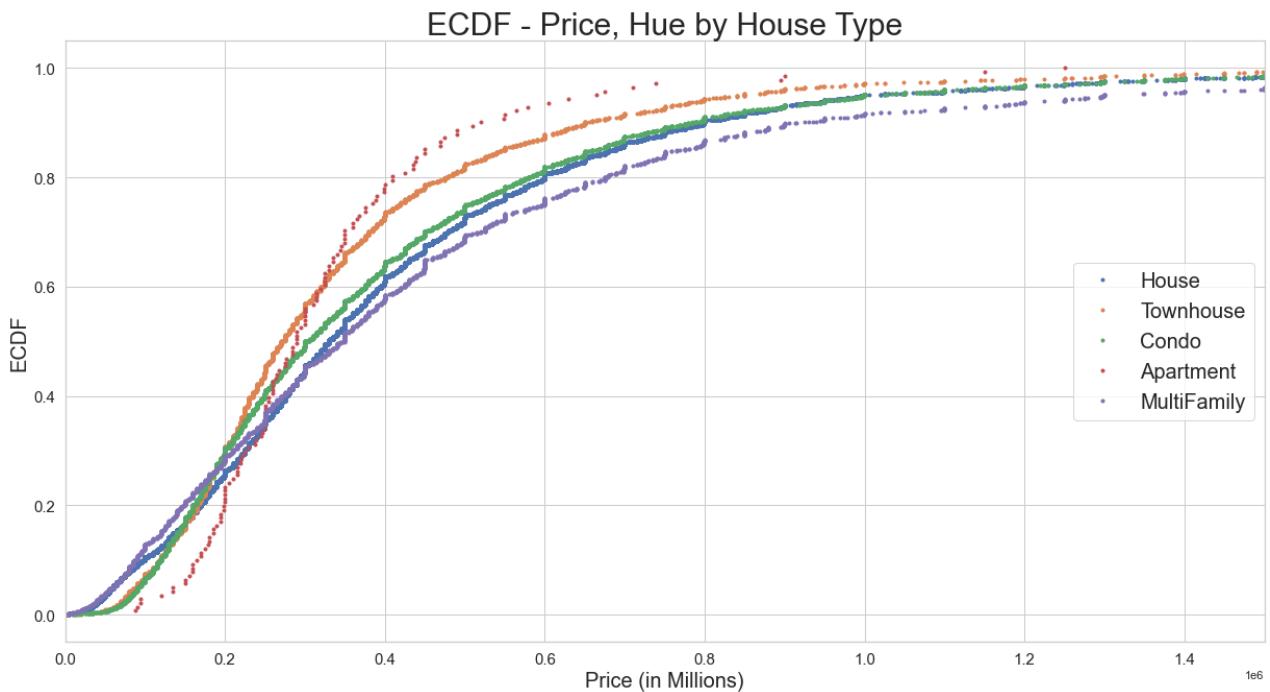
# Legend
plt.legend(['House', 'Townhouse', 'Condo', 'Apartment', 'MultiFamily'], loc=7, prop={'size': 16})

# Annotation, Label, Tick, Title
plt.xlabel('Price (in Millions)', fontsize=20)
plt.ylabel('ECDF', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('ECDF - Price, Hue by House Type', fontsize=30)

# Set Limit for x
axes.set_xlim(0,1500000)

# Show plot
plt.show()

```



Apartments near 200k range, have a clear premium/higher price than all other types of homes.

Appartments Prices drastically change near 350k to 400k and becomes a lot lower comparing to others

In [60]:

```

# DataFrame - sorting custom df for the plot
room_sorted_df = house_df[(house_df['Bedrooms'] <=20) & (house_df["Bathrooms"] <= 7)]

# PairGrid - with the sorted data frame with hue
g = sns.PairGrid(room_sorted_df, diag_sharey=False, hue="Type")
# Upper side of grid is scatter
g.map_upper(sns.scatterplot, s=15)

```

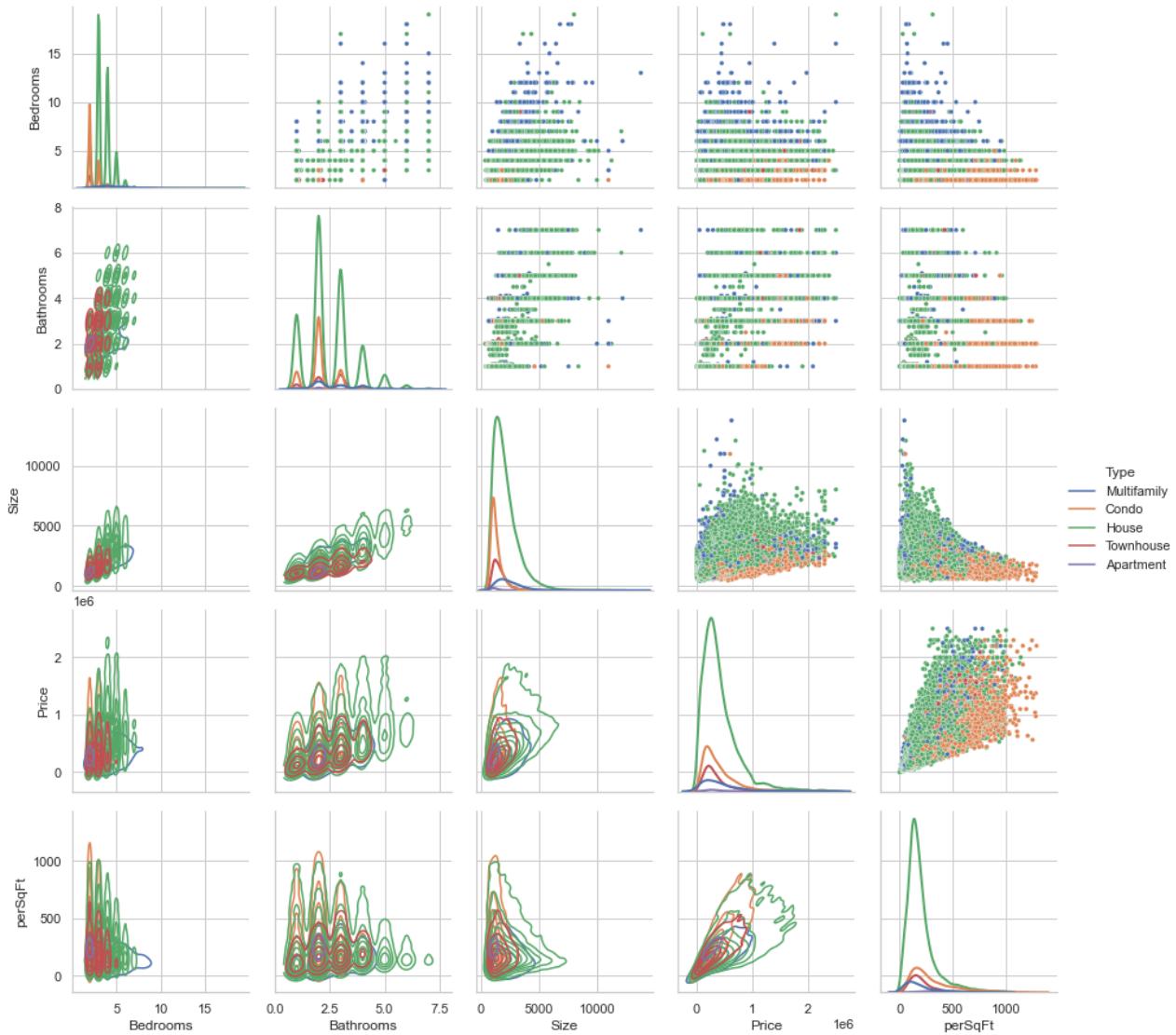
Loading [MathJax]/extensions/Safe.js KDE

```

g.map_lower(sns.kdeplot)
# Diagonal is also KDE
g.map_diag(sns.kdeplot, lw=2)
# Adding legend
g.add_legend()

```

Out[60]: <seaborn.axisgrid.PairGrid at 0x7fdcbdc438e0>



Top right side of graph, Orange - Condos has clearly smaller size

Bottom left side of graph, - Higher perSqFt has lessor Bedrooms

In [61]:

```

# Figure Size
plt.figure(figsize=(20,10))
# Countplot by state, how many data we have for each state
sns.barplot(data=house_df, x="Price", y="Type", palette="deep")

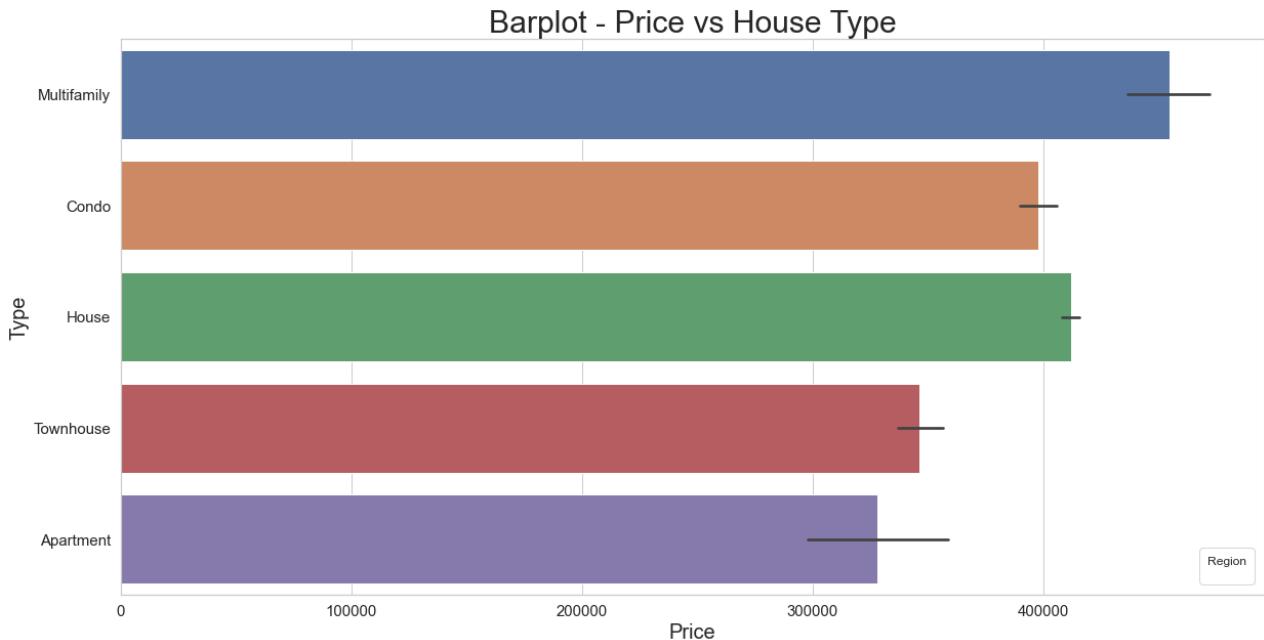
# Annotation, Label, Tick, Title
plt.xlabel('Price', fontsize=20)
plt.ylabel('Type', fontsize = 20)
plt.yticks(fontsize=15)

```

Loading [MathJax]/extensions/Safe.js size=15

```
plt.title('Barplot - Price vs House Type', fontsize=30)
plt.legend(title="Region", loc="lower right", fontsize=20)
```

No handles with labels found to put in legend.
Out[61]: <matplotlib.legend.Legend at 0x7fdcc07d0f10>



Multifamily - Most expensive

Condo - is close to House

House - is second highest for price

Townhouses - is a lot lower than house and condo, closer to apartment.

- This data is useful, because in theory, Townhouses should worth more than condos because they dont have common walls and have independent units. When shopping for homes, Townhouses might have higher chance to get a bargain or have higher values.

Apartments - lowest Price compare to others.

```
In [88]: # Figure Size
plt.figure(figsize=(20,10))

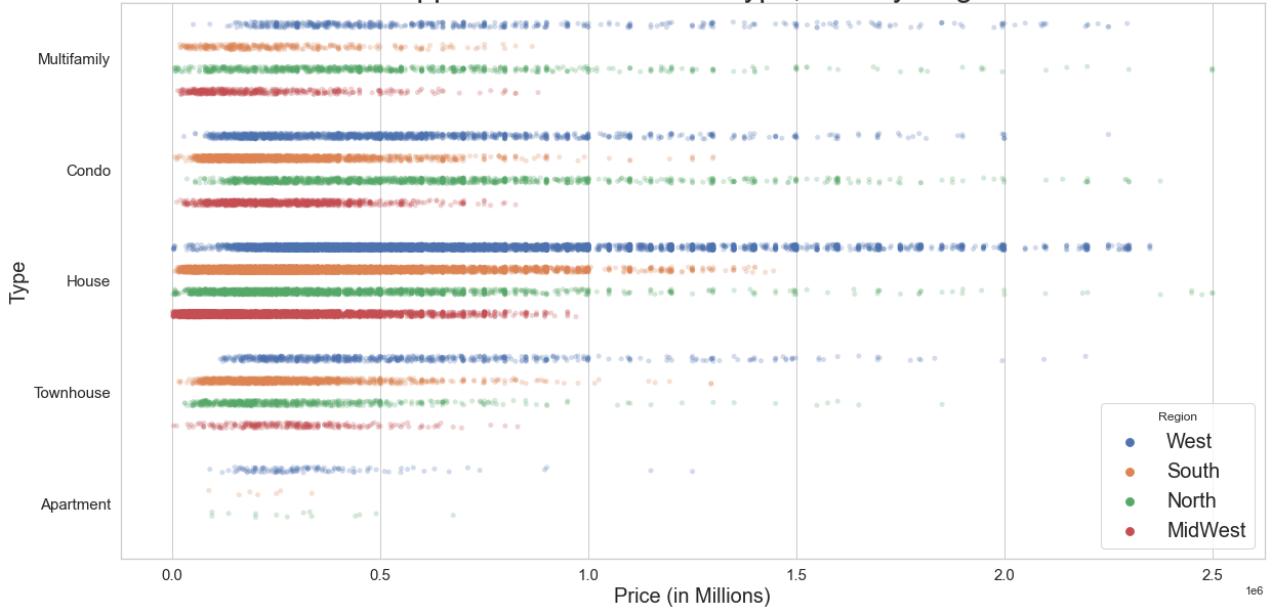
# Stripplot
# Dodge is not to mix the color together. (if False, all the dots will be mixed together)
sns.stripplot(x="Price", y="Type", hue="Region", data=house_df, dodge=True, alpha=.25)

# Annotation
plt.xlabel('Price (in Millions)', fontsize=20)
plt.ylabel('Type', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Stripplot - Price vs House Type, Hue by Region', fontsize=30)
plt.legend(title="Region", loc="lower right", fontsize=20)
```

Loading [MathJax]/extensions/Safe.js

Out[88]: <matplotlib.legend.Legend at 0x7fdcc08d8910>

Stripplot - Price vs House Type, Hue by Region



Stripplot can clearly see where most data is focused in for each Type of house via Region.

Multifamily - West Regions is clearly higher than all other regions

Condo - North and West Regions is clearly higher than all other regions

House - West Region is clearly higher than all other regions, lowest for MidWest

Townhouses - West Region is clearly higher than all other regions, the prices for other regions are extremely close

Apartments - All Region's price all relatively close to each other

MidWest has few/no Apartments

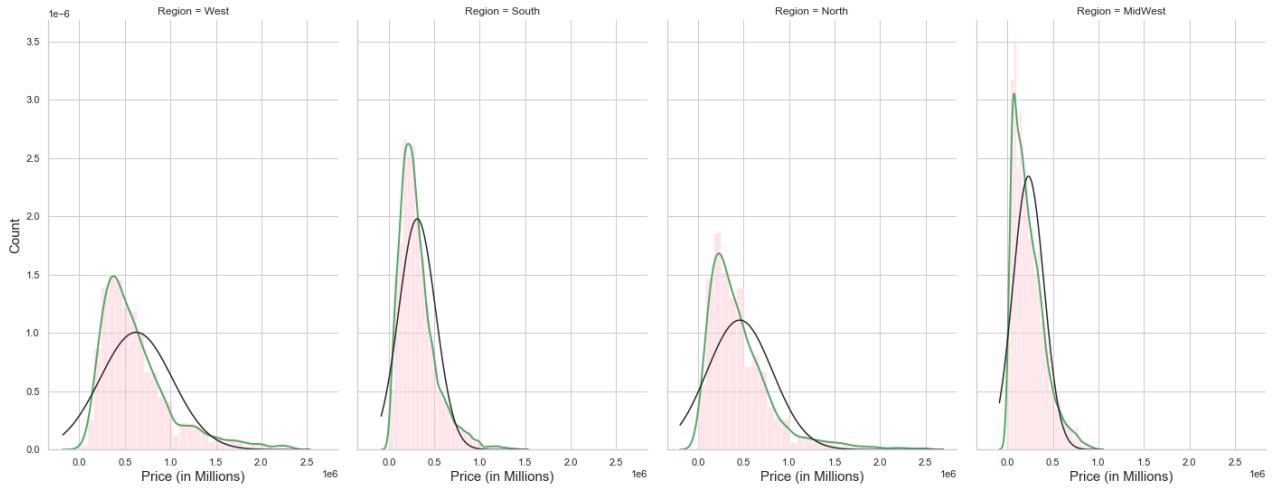
In []:

```
# FacetGrid - distribution by Price in Region
g = sns.FacetGrid(col='Region', data=house_df)
g.map(sns.distplot, 'Price', bins = 30, color='pink', kde=True, fit=stats.norm, kde_kws={"# Annotations
g.fig.set_figheight(10)
g.fig.set_figwidth(20)
g.set_xlabels("Price (in Millions)", fontsize=15)
g.set_ylabels("Count", fontsize=15)

# # Distribution plot of Prices divide into type of houses, hue of region.
# plt.figure(figsize=(20,10))
# sns.histplot(house_df,x='Price', hue='Region', edgecolor='0.5', bins=150,palette='bright')
# # Annotation
# plt.title('Histplot - Price, Hue by Region', fontsize=12)
# plt.xlabel('Price(in Millions)', fontsize=20)
# plt.ylabel('House Data Count', fontsize = 20)
# plt.yticks(fontsize=15)
# plt.xticks(fontsize=15)
```

Loading [MathJax]/extensions/Safe.js

Out[]: <seaborn.axisgrid.FacetGrid at 0x7ffc7abd03d0>



- West Region's most data is around 500k, comparing with other region, West have higher Price mean.
- South Region has most data points. Price is more focused around 250k, half of West Region.
- MidWest has lowest Price mean, where most data is focused on the left side of Price
- Most distribution are positively skewed towards to the right.

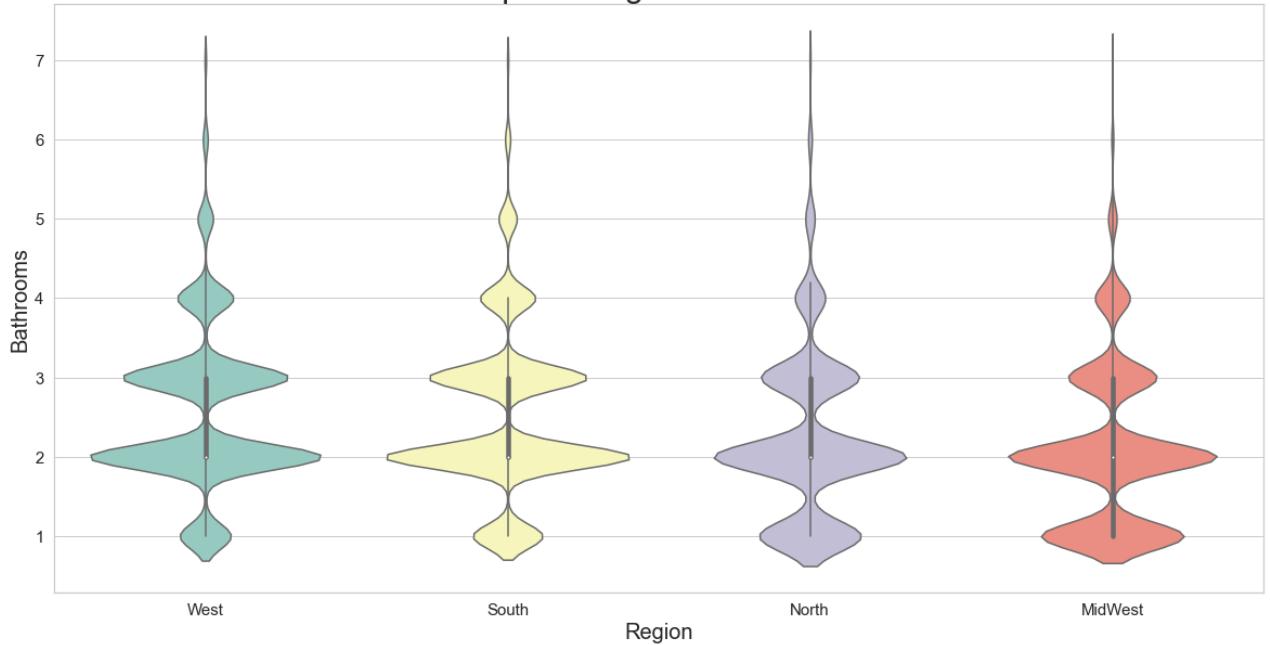
In []:

```
# Figure Size
plt.figure(figsize=(20,10))
bath_sorted_df = house_df[house_df["Bathrooms"] <= 7]
sns.violinplot(data=bath_sorted_df, x='Region', y='Bathrooms', palette="Set3")

# Annotation
plt.xlabel('Region', fontsize=20)
plt.ylabel('Bathrooms', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Violinplot - Region vs Bathrooms', fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Violinplot - Region vs Bathrooms')

Violinplot - Region vs Bathrooms



Violinplot to see each Region's number of bathrooms and how are data focused.

Overall, most data are around 2 and 3 Bathrooms.

MidWest also have many houses with 1 Bathroom.

In []:

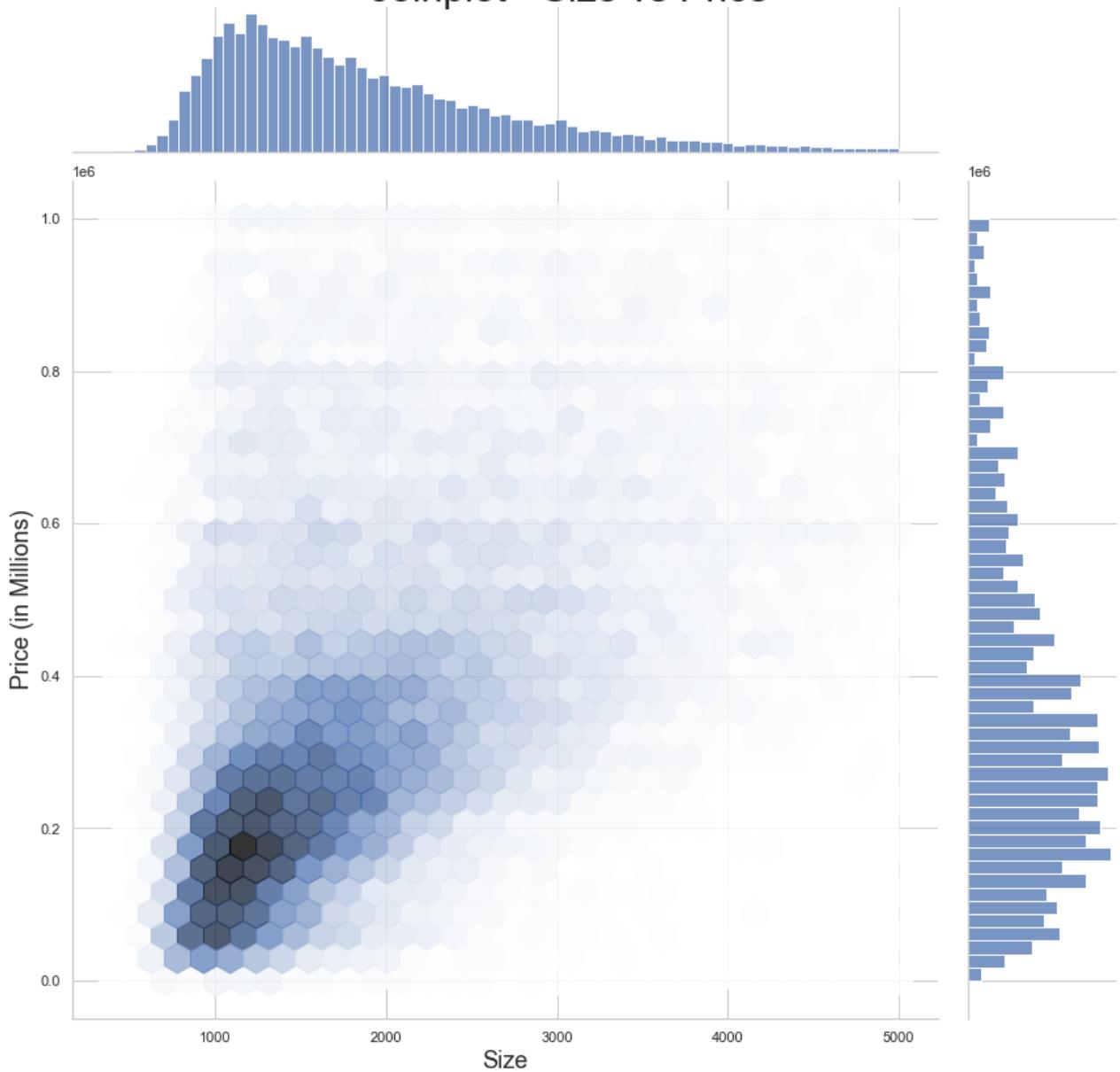
```
# DataFrame sorting custom df for graphing
price_sorted_df = house_df[(house_df["Price"] <= 1000000) & (house_df["Size"] <= 5000)]

# joint_kws is the hex size
_ = sns.jointplot(y='Price', x='Size', data=price_sorted_df, alpha=0.8, kind="hex", joint_kws={"hexbin": {"gridsize": 10}})

# Labels for jointplot
_.set_axis_labels("Size", "Price (in Millions)", fontsize = 18)

# Set title
_.fig.suptitle('Jointplot - Size vs Price', fontsize = 30)
# Adjusting jointplot size to 95%, to leave room for title
_.fig.subplots_adjust(top=0.95)
```

Joinplot - Size vs Price



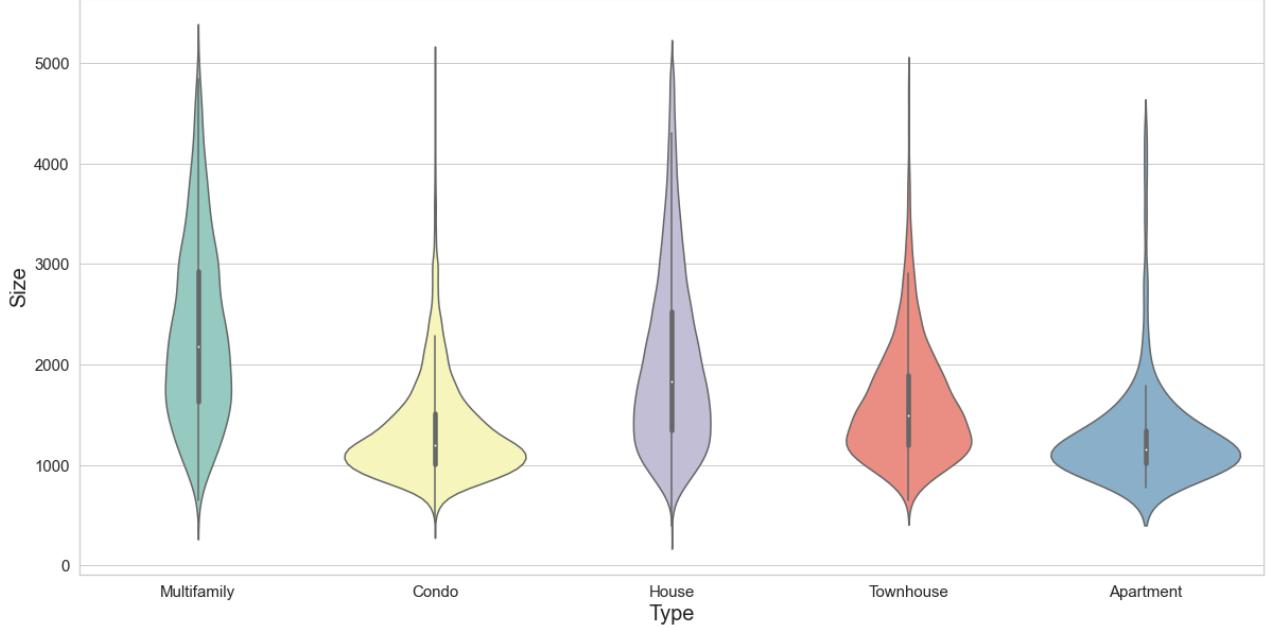
- Price: most data is around \$200,000
- Size: most data is between early 1,000 to 2,000 SqFt

In []:

```
# Figure Size
plt.figure(figsize=(20,10))
# Dataframe sort custom df for graph
bath_sorted_df = house_df[house_df["Size"] <= 5000]
sns.violinplot(data=bath_sorted_df, x='Type', y='Size', palette="Set3")

# Annotation
plt.xlabel('Type', fontsize=20)
plt.ylabel('Size', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Violinplot - Type vs Size', fontsize=30)
```

Violinplot - Type vs Size



Condo and Apartment are wider around 1000 SqFt.

Multifamily and House are slimmer where all data is spread between 1k and 4k, clearly multifamily have larger sizes.

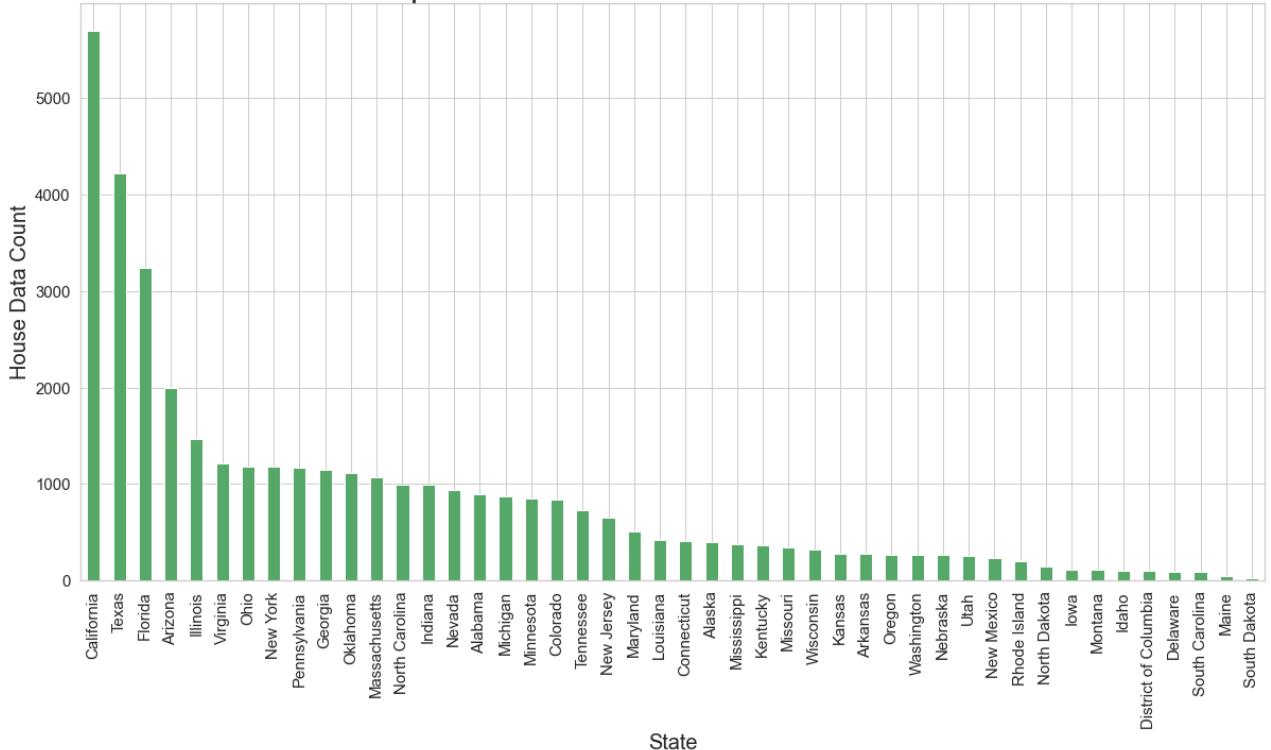
Townhouse is in between Condos vs Multifamilies

```
In [ ]: # Countplot by state, how many data we have for each state
plt.figure(figsize=(20,10))
house_df['fullState'].value_counts().plot(kind='bar',color='g')

# Annotation
plt.xlabel('State', fontsize=20)
plt.ylabel('House Data Count', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15,rotation=90)
plt.title('Countplot - Amount of House Data for each State', fontsize=30)
```

```
Out[ ]: Text(0.5, 1.0, 'Countplot - Amount of House Data for each State')
```

Countplot - Amount of House Data for each State



We have most data in California, Texas, and Florida. Partial indicator that these states have more real estate activities.

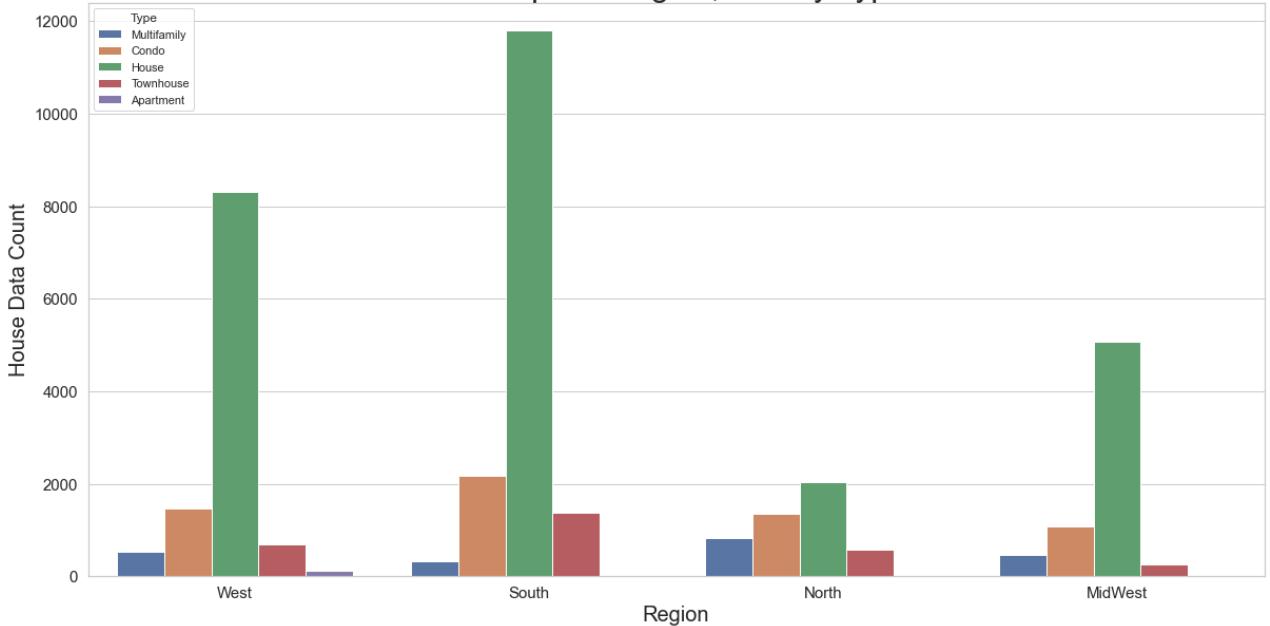
In []:

```
# Count plot x is region and hue is Type of houses
plt.figure(figsize=(20,10))
g=sns.countplot(x='Region',data=house_df,hue='Type')
g.set(ylim=(0, None))

# Annotation
plt.xlabel('Region', fontsize=20)
plt.ylabel('House Data Count', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Countplot - Region, Hue by Type' , fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Countplot - Region, Hue by Type')

Countplot - Region, Hue by Type



Countplot by Region and Hue by Property Types.

1. Single Family Residence are primary market for United States

2. Condos are 2nd, follow by Townhouses as 3rd

The North Region has a high percentage of Condos and Townhouses relative to the houses.

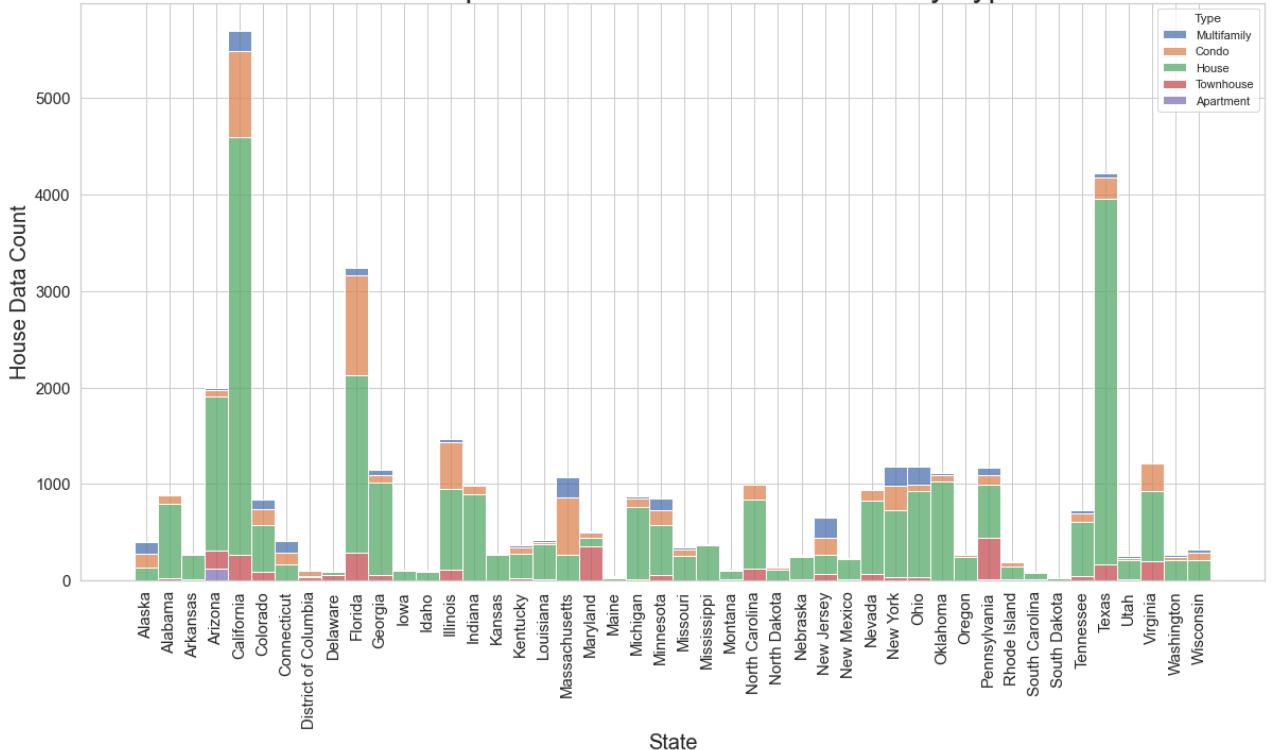
In []:

```
# Histplot stacked by type
plt.figure(figsize=(20,10))
sns.histplot(data=house_df,x='fullState',hue='Type',multiple='stack',edgecolor='white')

# Annotation
plt.xlabel('State', fontsize=20)
plt.ylabel('House Data Count', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15,rotation=90)
plt.title('Stack Histplot - State House Count Stack by Type', fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Stack Histplot - State House Count Stack by Type')

Stack Histplot - State House Count Stack by Type



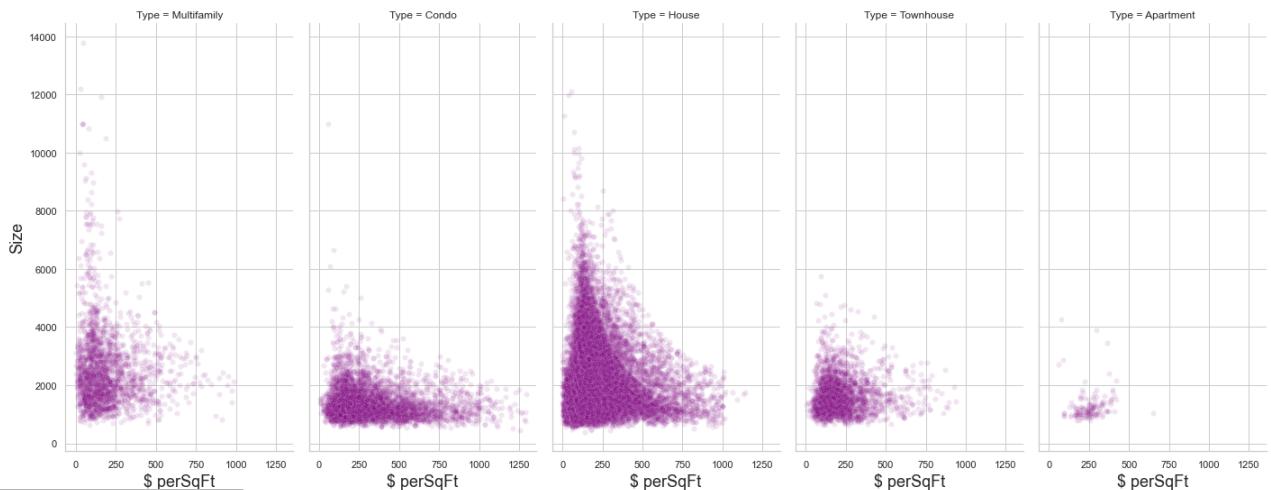
- Massachusetts, Alaska, District of Columbia are primarily Condos.
- Maryland, Pennsylvania, Delaware are primarily Townhouses or have huge % in Townhouses.

In []:

```
# FacetGrid
g = sns.FacetGrid(col='Type', data=house_df)
# Map out with scatter plot X perSqFt, y Size, Columns by Type
g.map(sns.scatterplot, "perSqFt", 'Size', alpha=0.1, color='purple')

# Figszie
g.fig.set_figheight(10)
g.fig.set_figwidth(20)
g.set_xlabels('$ perSqFt', fontsize = 18)
g.set_ylabels('Size', fontsize = 18)
```

Out[]: <seaborn.axisgrid.FacetGrid at 0x7ffc907e2100>



Loading [MathJax]/extensions/Safe.js

Multifamily perSqFt are mostly focused between \$0 - \$250.

Condo Size are clearly smaller. Most focused under 2000 SqFt
 House is everywhere, but there's clearly negative relationship between Size and perSqFt.
 Townhouse and Apartment is focused under 2000 SqFt and between \$0 - \$250 perSqFt

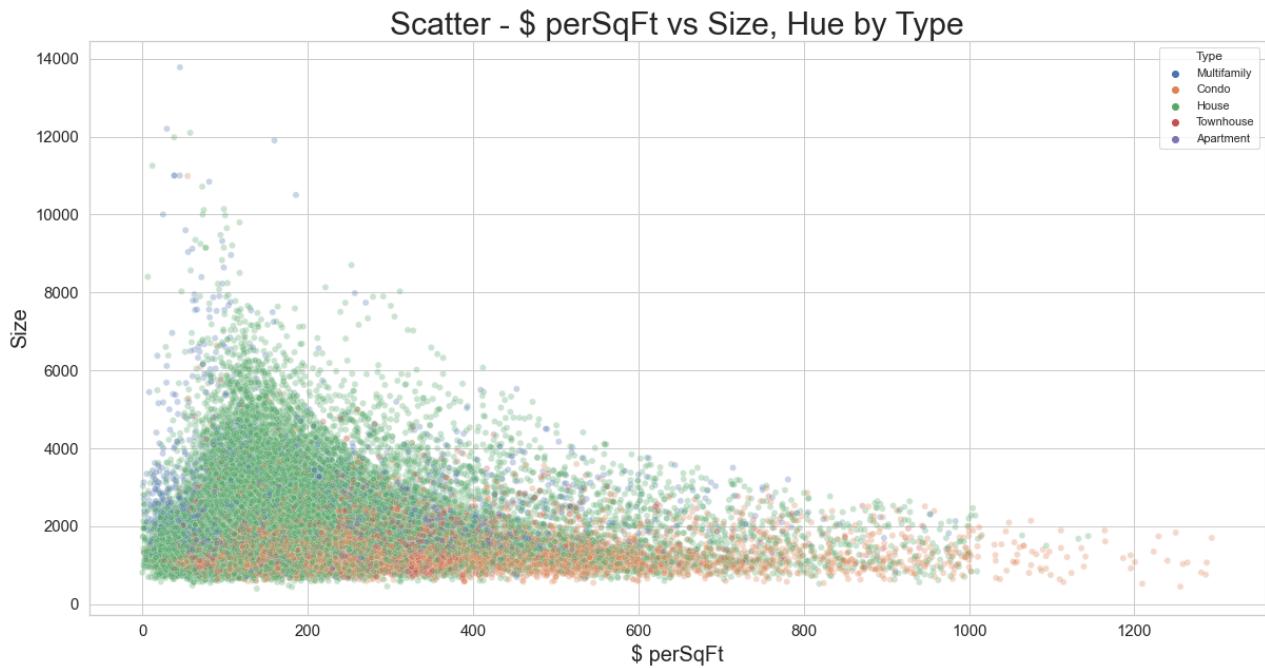
In []:

```
# Scatterplot perSqFt, Size, hue by Type
plt.figure(figsize=(20,10))

# Scatterplot
sns.scatterplot(data=house_df,x='perSqFt',y='Size',hue='Type', alpha=0.3)

# Annotation
plt.xlabel('$ perSqFt', fontsize=20)
plt.ylabel('Size', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Scatter - $ perSqFt vs Size, Hue by Type', fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Scatter - \$ perSqFt vs Size, Hue by Type')



- The higher the perSqFt the smaller the house Size.
- Most Condos have higher perSqFt prices.

In []:

```
# List - Making Size Category List, so we can sort it later
sizeCat_list = list(house_df['SizeCat'].unique())
sizeCat_list

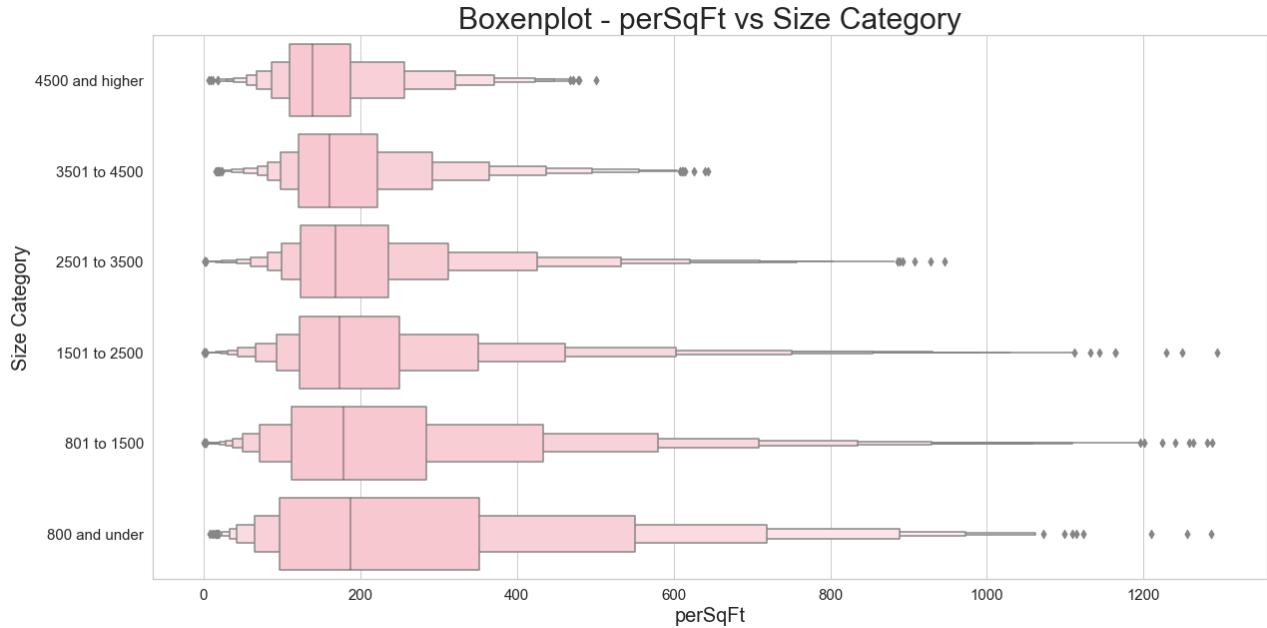
# Figure Size
plt.figure(figsize=(20,10))
# List - Sort the Size Category
sizeCat_list_sorted = ['800 and under','801 to 1500', '1501 to 2500','2501 to 3500','350
# Boxenplot
sns.boxenplot(y="SizeCat", x="perSqFt", color="pink", order=sizeCat_list_sorted, data=h
```

```

plt.xlabel('$ perSqFt', fontsize=20)
plt.ylabel('Size Category', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Boxenplot - $ perSqFt vs Size Category', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Boxenplot - perSqFt vs Size Category')



Steady increase of \$ perSqFt as Size decreases. (Negatively Correlated)

```

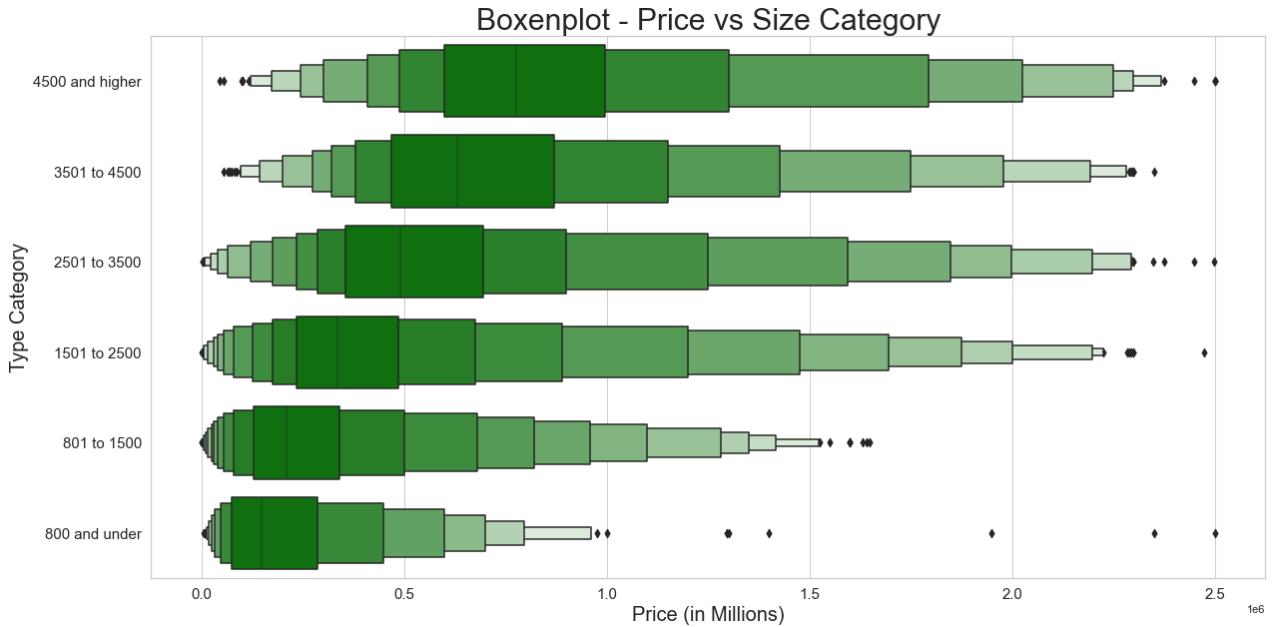
In [ ]:
# List - Making Size Category list, so we can sort it later
sizeCat_list = list(house_df['SizeCat'].unique())
sizeCat_list

# Figure Size
plt.figure(figsize=(20,10))
# List - Sort the Size Category
sizeCat_list_sorted = ['800 and under', '801 to 1500', '1501 to 2500', '2501 to 3500', '3501 to 4500', '4500 and higher']
# Boxenplot
sns.boxenplot(y="SizeCat", x="Price", color="green", scale='linear', order=sizeCat_list_sorted)

# Annotation
plt.xlabel('Price (in Millions)', fontsize=20)
plt.ylabel('Type Category', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Boxenplot - Price vs Size Category', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Boxenplot - Price vs Size Category')



Exact Opposite with Price, as Size get larger Price is larger. (Positively Correlated)

In []:

```
# Scatterplot perSqFt, Size, hue by Type
plt.figure(figsize=(20,10))
sns.scatterplot(data=house_df,x='Price',y='Size',hue='Type', alpha=0.4)

# Annotation
plt.xlabel('Price (in Millions)', fontsize=20)
plt.ylabel('Size', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Scatter - Price vs Size, Hue by Type', fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Scatter - Price vs Size, Hue by Type')



Loading [MathJax]/extensions/Safe.js

on between Price and Size. The larger the house the price will be slightly

increased.

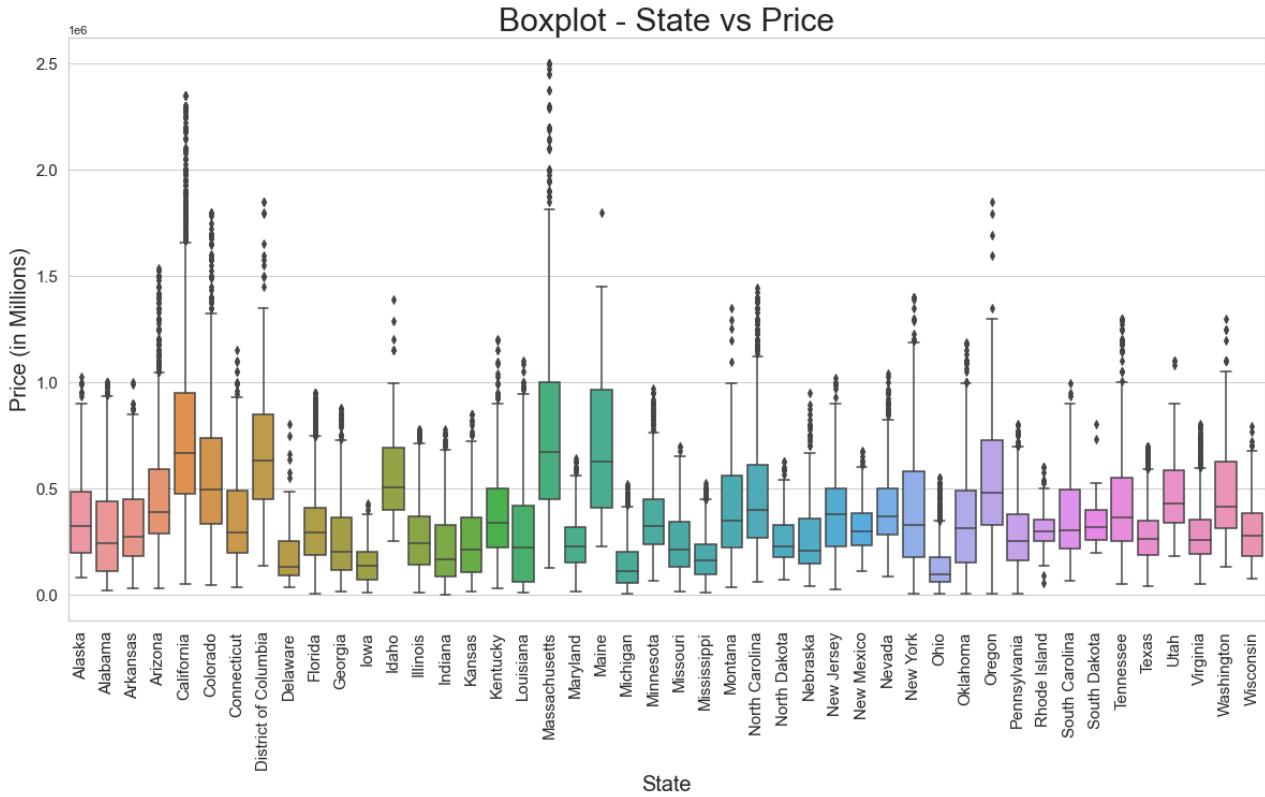
- Condo size is at the bottom of chart

In []:

```
# Boxplot of States by House price
plt.figure(figsize=(20,10))
ax = sns.boxplot(data = house_df, x = 'fullState' , y = 'Price')

# Annotation
plt.xlabel('State', fontsize=20)
plt.ylabel('Price (in Millions)', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15,rotation=90)
plt.title('Boxplot - State vs Price', fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Boxplot - State vs Price')



- The house price for California, District of Columbia, Maine, Massachusetts are fairly high. there are extreme outliers in many states.
- Rich people are everywhere!

Double BAM!!

In []:

```
# Boxplot of States by $ perSqFt
plt.figure(figsize=(20,10))
ax = sns.boxplot(data = house_df, x = 'fullState' , y = 'perSqFt')
ax.set_yticks([0, None])

# Annotation
plt.xlabel('State', fontsize=20)
plt.ylabel('perSqFt', fontsize = 20)
plt.yticks(size=15)
```

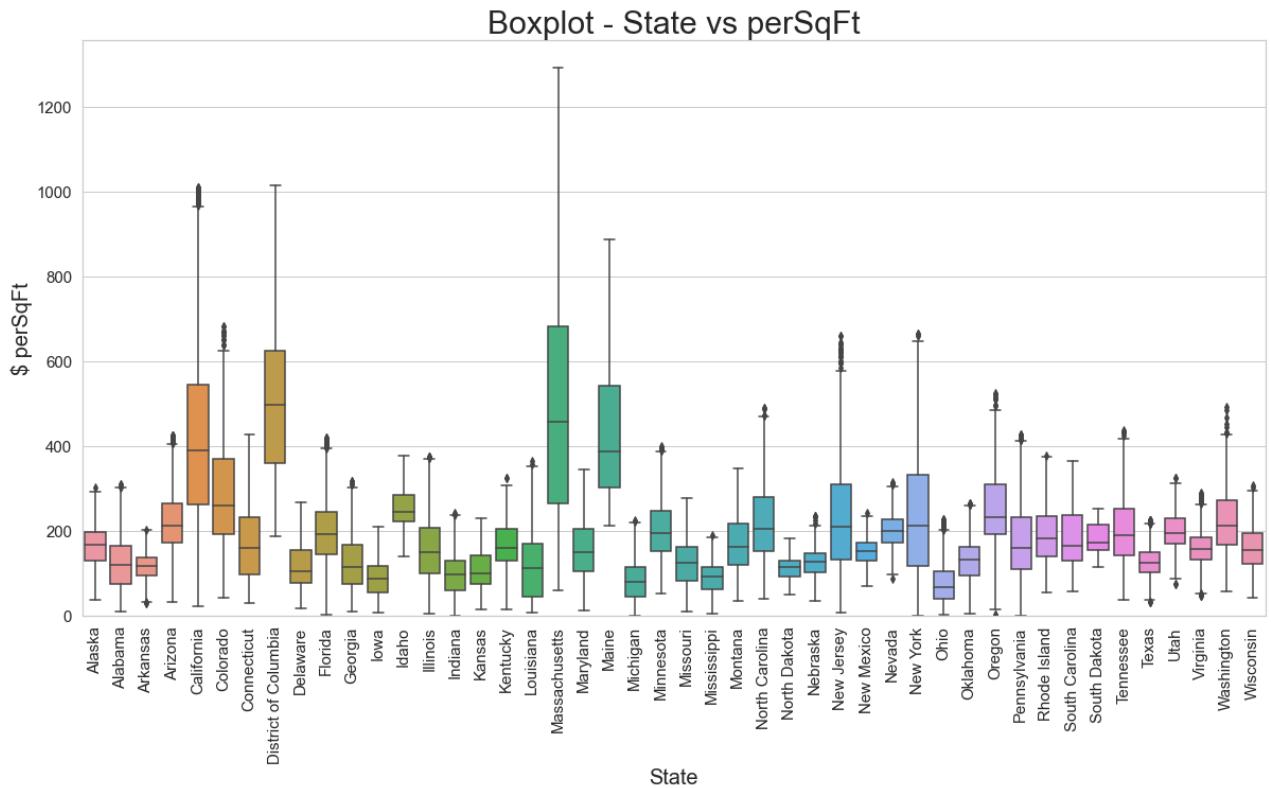
Loading [MathJax]/extensions/Safe.js

```

plt.xticks(fontsize=15, rotation=90)
plt.title('Boxplot - State vs $ perSqFt', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Boxplot - State vs perSqFt')



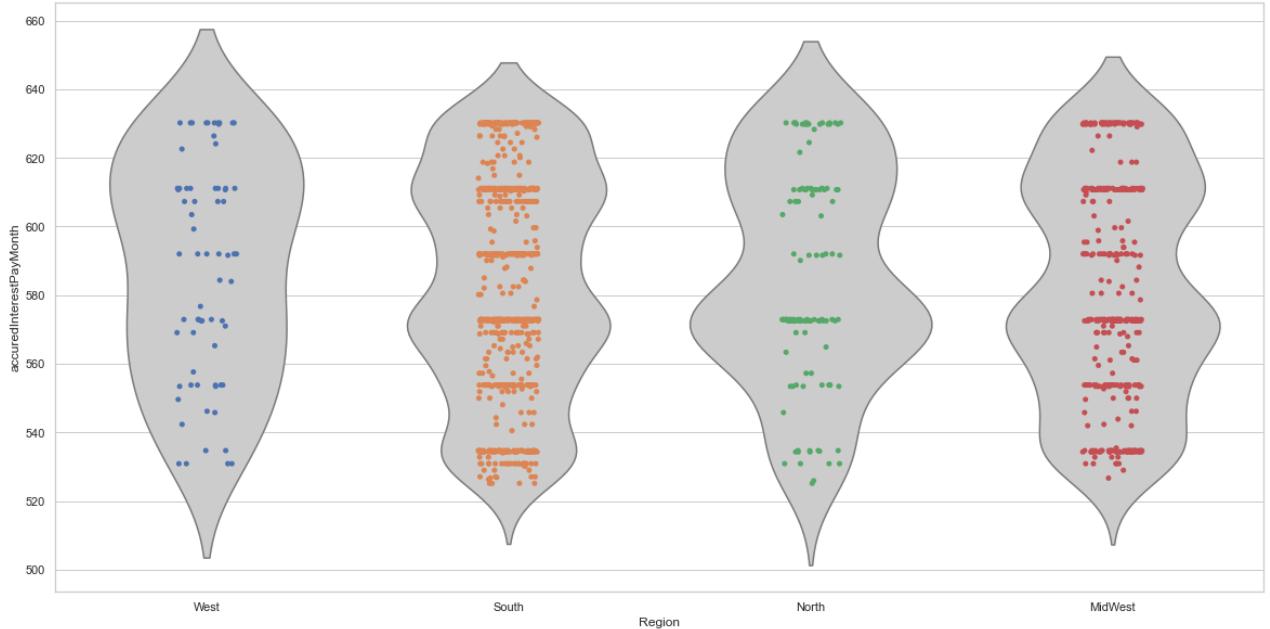
- The \$ perSqFt for the same states; California, District of Columbia, Massachusetts, Maine are clearly top 4 again.
- Comparing with the Price boxplot, Maine has higher Prices, lower perSqFt. So the house Size there should be proportionally larger.
This is exact opposite for District of Columbia lower Prices, higher perSqFt. So house Size should be proportionally smaller.

In [89]:

```

#####
#graph the distribution and see where the most of data points land using a violin plot
plt.figure(figsize=(20,10))
ax = sns.violinplot(x="Region", y="accuredInterestPayMonth", data=houseRanges, inner=None)
ax = sns.stripplot(x="Region", y="accuredInterestPayMonth", data=houseRanges)

```



Markdown: It seems that you are more likely to find a house in the South and Midwest, due to amount of distribution compared to the North and West Regions.

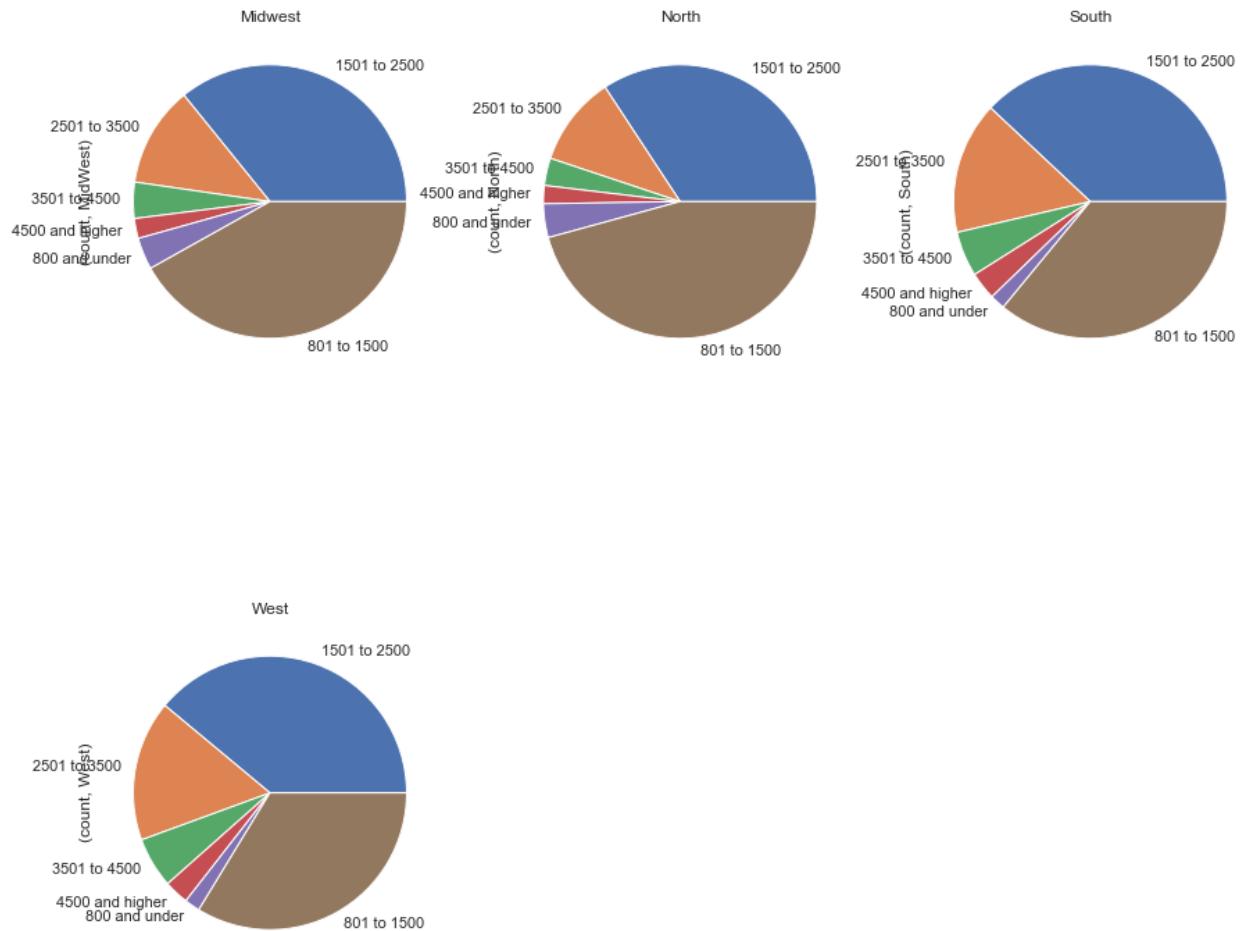
in the South you have plenty of options from ranges of \$560 to \$580 and \$600 and \$620;

In the Midwest, you also have plenty of options as well, from ranges of \$560 to \$580 and \$600 and \$620.

In [90]:

```
# Dataframe - create a multi-index dataframe for Pie Chart
regionSize_df = house_df.groupby(['Region', 'SizeCat']).size()
regionCounts= regionSize_df.to_frame(name = 'count')

#create the pie chart subplots
ax = regionCounts.unstack(level=0).plot(kind='pie', subplots=True, figsize=(15, 30), la
plt.show()
```



Interesting points is that the Midwest and North Regions have a larger distributions of properties that are 800 sqft and lower.

State_df

Descriptive for State_df

In []:

```
state df.head()
```

Out[]:

	fullState	GDP	Spending	Population	Income	Region	State	Bathrooms	Bedrooms	
0	Alabama	196906.10	176479.80	4934193	228748.80	South	AL	2.62	3.38	30%
1	Alaska	50161.00	35635.70	724357	46430.30	West	AK	2.97	4.59	36%
2	Arizona	320550.60	287090.10	7520103	368458.60	West	AZ	2.51	3.34	46%
3	Arkansas	114943.50	104488.80	3033946	143147.90	South	AR	2.96	3.64	33%
4	California	2663665.90	1835980.60	39613493	2763312.00	West	CA	2.56	3.44	77%

In []:

```
state df.describe()
```

Out[]:

	GDP	Spending	Population	Income	Bathrooms	Bedrooms	Price	Size	per
mean	390208.88	299606.37	7078103.24	419131.06	2.42	3.37	371094.07	2024.62	19
std	482294.66	340567.83	7600879.42	496796.28	0.29	0.34	160612.69	304.99	9
min	46158.10	33631.30	714153.00	46430.30	1.81	2.78	134835.39	1525.06	7
25%	118834.20	98120.93	2929519.75	129528.12	2.23	3.16	270628.96	1774.21	12
50%	229092.70	187682.30	5106011.50	265093.00	2.46	3.33	345951.25	2035.53	16
75%	492387.12	365095.10	8402224.00	531894.30	2.61	3.44	425050.39	2275.08	21
max	2663665.90	1835980.60	39613493.00	2763312.00	2.97	4.59	800828.82	2804.81	50

In []:

```

# Finding Highest/Lowest 3 States of per square footage and size
sqft_top = state_df.sort_values(by='perSqFt', ascending=False).head(3)[['State', 'perSqFt']]
sqft_bot = state_df.sort_values(by='perSqFt', ascending=True).head(3)[['State', 'perSqFt']]
size_top = state_df.sort_values(by='Size', ascending=False).head(3)[['State', 'Size']]
size_bot = state_df.sort_values(by='Size', ascending=True).head(3)[['State', 'Size']]
# For Loop to print the output for $ perSqFt
print('Top 3 Highest and Lowest $ perSqFt')
dflist = sqft_top, sqft_bot
for d in dflist:
    dict_d = dict(zip(d.iloc[:,0],d.iloc[:,1]))
    for k in dict_d:
        print('State {} have average of ${:.2f} per square feet'.format(k,dict_d[k]))

# For Loop to print the output for Size
print('\nTop 3 Largest and Smallest House Size')
dflist2 = size_top, size_bot
for d in dflist2:
    dict_d = dict(zip(d.iloc[:,0],d.iloc[:,1]))
    for k in dict_d:
        print('State {} have average of {:.0f} square feet.'.format(k,dict_d[k]))

```

Top 3 Highest and Lowest \$ per SqFt
State MA have average of \$501.47 per square feet
State DC have average of \$500.22 per square feet
State ME have average of \$437.12 per square feet
State OH have average of \$79.80 per square feet
State MI have average of \$86.42 per square feet
State MS have average of \$90.56 per square feet

Top 3 Largest and Smallest House Size
State AR have average of 2805 square feet.
State UT have average of 2524 square feet.
State OK have average of 2511 square feet.
State DC have average of 1525 square feet.
State DE have average of 1538 square feet.
State MD have average of 1578 square feet.

In []:

```
# Groupby table #2  
state df.groupby(['Region']).mean()
```

Out[]:

Loading [MathJax]/extensions/Safe.js
Region

Region	GDP	Spending	Population	Income	Bathrooms	Bedrooms	Price	Size	perS
MidWest	295767.66	233952.69	5709479.67	319849.12	2.19	3.24	248014.61	1940.53	128
North	450836.24	333676.30	6821733.50	487617.51	2.27	3.37	496524.37	1790.17	307
South	409233.81	327572.71	8341364.40	443628.85	2.50	3.29	314451.73	2069.02	150
West	423199.96	308314.50	7034969.00	444224.20	2.65	3.62	491380.10	2226.31	233

- Years of Saving to buy a house is highest in West Region than other regions and Lowest in MidWest Region.
- Individual's Income highest in North Region, lowest in South Region.
- perSqFt highest in North lowest in MidWest.

In [76]:

```
# Top income states after paying mortgage
state_df[['fullState','annualIncomeAfterMortgage']].sort_values(by='annualIncomeAfterMo
```

Out[76]:

	fullState	annualIncomeAfterMortgage
6	Connecticut	61961.22
30	New York	56563.87
28	New Jersey	56034.83
19	Maryland	55654.95
8	District of Columbia	52841.86

In [78]:

```
# compare that to the savings rate table
state_df[['fullState','SavingsRate']].sort_values(by='SavingsRate',ascending=False).hea
```

Out[78]:

	fullState	SavingsRate
6	Connecticut	0.36
20	Massachusetts	0.34
19	Maryland	0.34
4	California	0.34
30	New York	0.33

Because SavingsRate didnt account for different mortgage payments, there are some states that do not match. Connecticut, NY, and MD seem like the best states to live in for economic savings.

Visualization for state_df

In []:

```
# Figure Size
```

Loading [MathJax]/extensions/Safe.js size=(20,10)

```
# Heatmap plotting
```

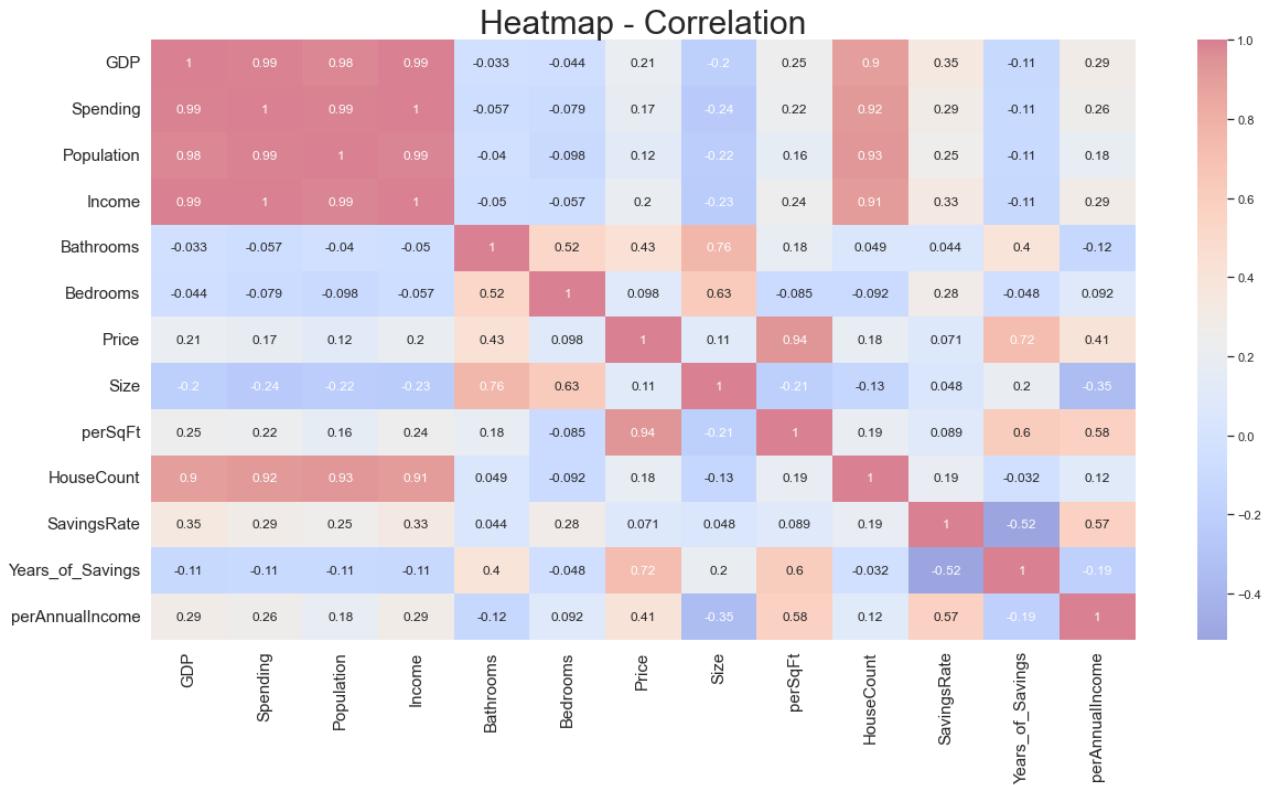
```

sns.heatmap(state_df.corr(), cmap='coolwarm' , annot=True, alpha=0.5)

# Annotations
plt.xticks(rotation=30)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15, rotation=90)
plt.title('Heatmap - Correlation', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Heatmap - Correlation')



In []:

```

# Heat Scatter
# Source: https://seaborn.pydata.org/examples/heat_scatter.html
# Note: this map is literally from the source, i really liked it thats why i want to pl

# Compute a correlation matrix and convert to Long-form
corr_mat = state_df.corr().stack().reset_index(name="correlation")

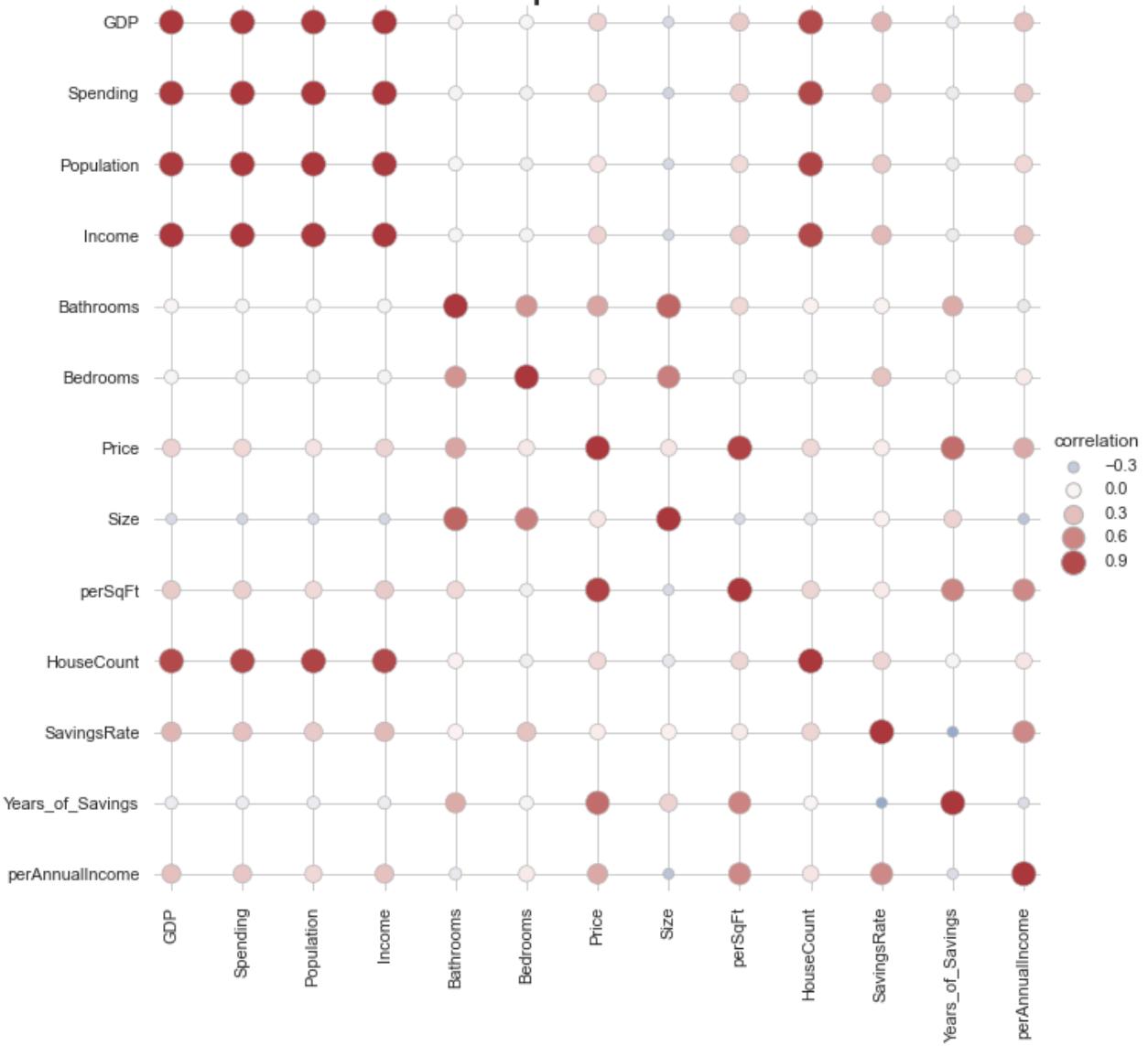
# Draw each cell as a scatter point with varying size and color
g = sns.relplot(data=corr_mat,x="level_0", y="level_1", hue="correlation", size="correl

# Tweak the figure to finalize
g.set(xlabel="", ylabel="", aspect="equal")
g.despine(left=True, bottom=True)
g.ax.margins(.02)
for label in g.ax.get_xticklabels():
    label.set_rotation(90)
for artist in g.legend.legendHandles:
    artist.set_edgecolor(".7")
plt.title('Heatmap - Correlation', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Heatmap - Correlation')

Heatmap - Correlation



Savings Rate is correlated highest with GDP and Income.

The house count is highly correlated with population. More population, more housing activity.

The \$ perSqFt has highest correlation with GDP

Size is highest correlated with Bedrooms, second correlated with Bathrooms.

It is negatively correlated with GDP, Population, Income, Spending, \$ perSqFt, and HouseCount.

Bedrooms almost same relationship correlations as Size

Unlike Bedrooms and Size, Bathrooms has a high correlation with Price.

Higher per Annual Income, smaller the Size

In []:

```
# Fig Size
plt.figure(figsize=(20,10))
```

Loading [MathJax]/extensions/Safe.js *kws is the reg Line*

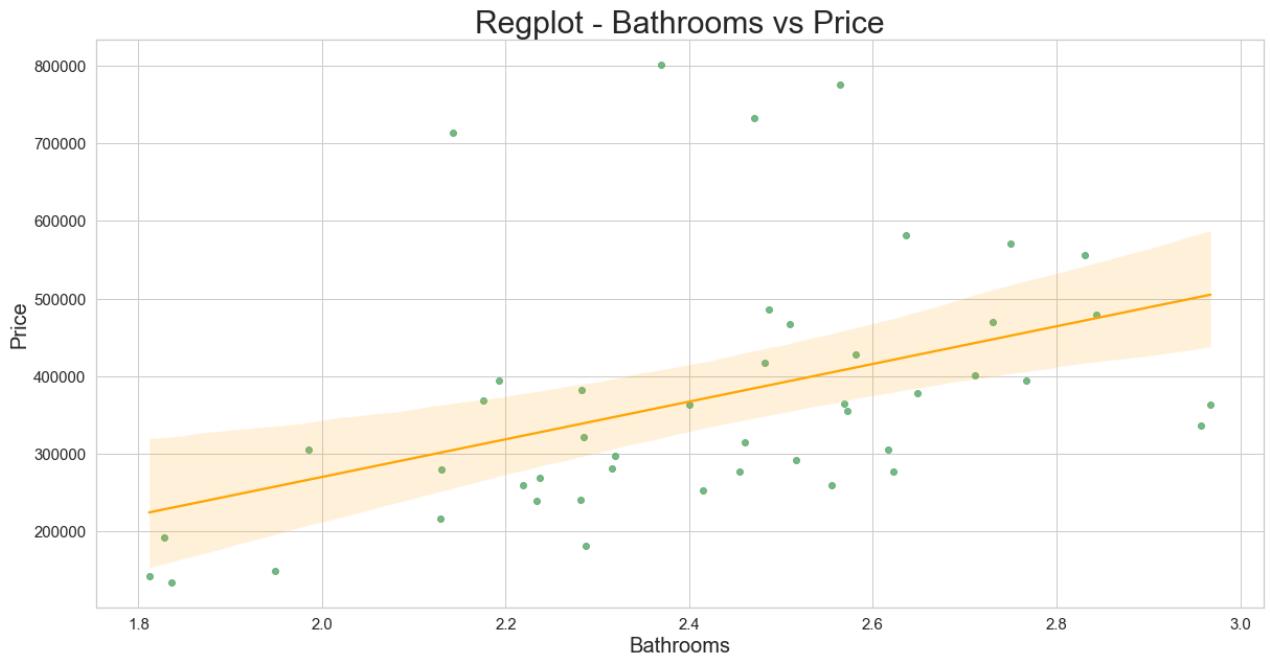
```

sns.regplot(data=state_df,y="Price", x="Bathrooms", color="g",line_kws={'color':'orange'})

# Annotation
plt.xlabel('Bathrooms', fontsize=20)
plt.ylabel('Price', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Regplot - Bathrooms vs Price', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Regplot - Bathrooms vs Price')



In []:

```

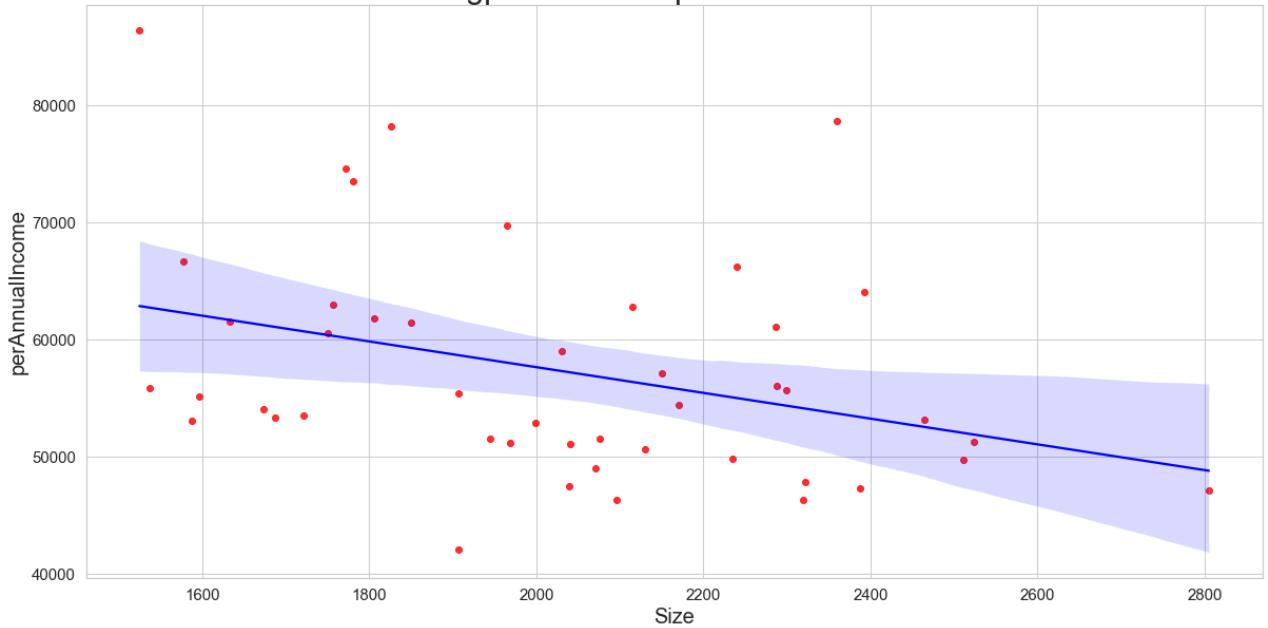
# Regression plot of SavingsRate vs perSqFt
plt.figure(figsize=(20,10))
g = sns.regplot(data = state_df, y = 'perAnnualIncome', x = 'Size', color='red', line_kws={'color':'red'})
plt.ticklabel_format(style='plain', axis='y')
plt.ticklabel_format(style='plain', axis='x')
# The States that have higher SavingRate will pay higher perSqFt for the houses. Note: I

# Annotation
plt.xlabel('Size', fontsize=20)
plt.ylabel('perAnnualIncome', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Regplot - Size vs perAnnualIncome', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Regplot - Size vs perAnnualIncome')

Regplot - Size vs perAnnuaIncome



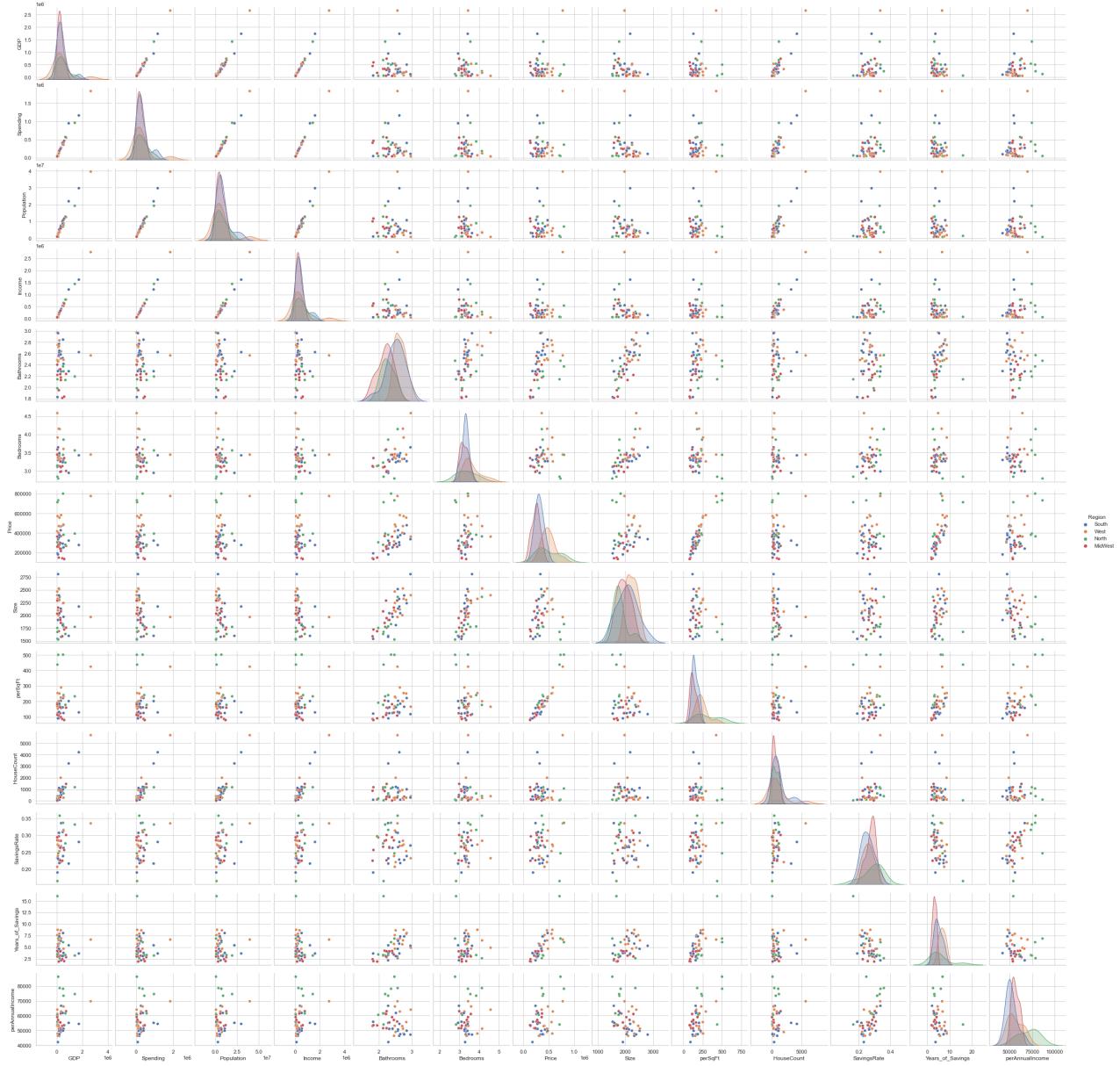
The more you make, the smaller place you get!! TRIPPLE BAMMM

In []:

```
# Print pairplot to visualize the correlations.  
sns.pairplot(data=state_df, hue="Region")  
# the more money you have the more bathrooms you will get.
```

Out[]:

```
<seaborn.axisgrid.PairGrid at 0x7ffc968da6a0>
```



GDP Income Spending and Population have a strong correlation.

Additionally, number of houses sold also have a strong correlation with these. So more real estate activity with higher GDP Income Spending and Population

SavingsRate and Bedrooms are slightly, the more you save the more bedrooms you are going to get.

Of course the bigger the house, the more Bedrooms and Bathrooms you will have.

SavingsRate and HouseCount has a slight correlation, the more you save, the more real estate activities there will be.

In []:

```
# Barplot
plt.figure(figsize=(20,10))
sorted_df = state_df.sort_values(by='Population', ascending=False)
sns.barplot(data=state_df, x='fullState',y='Population')

# Annotation
plt.xlabel('State', fontsize=20)
plt.ylabel('Population (in Ten Millions)', fontsize = 20)
```

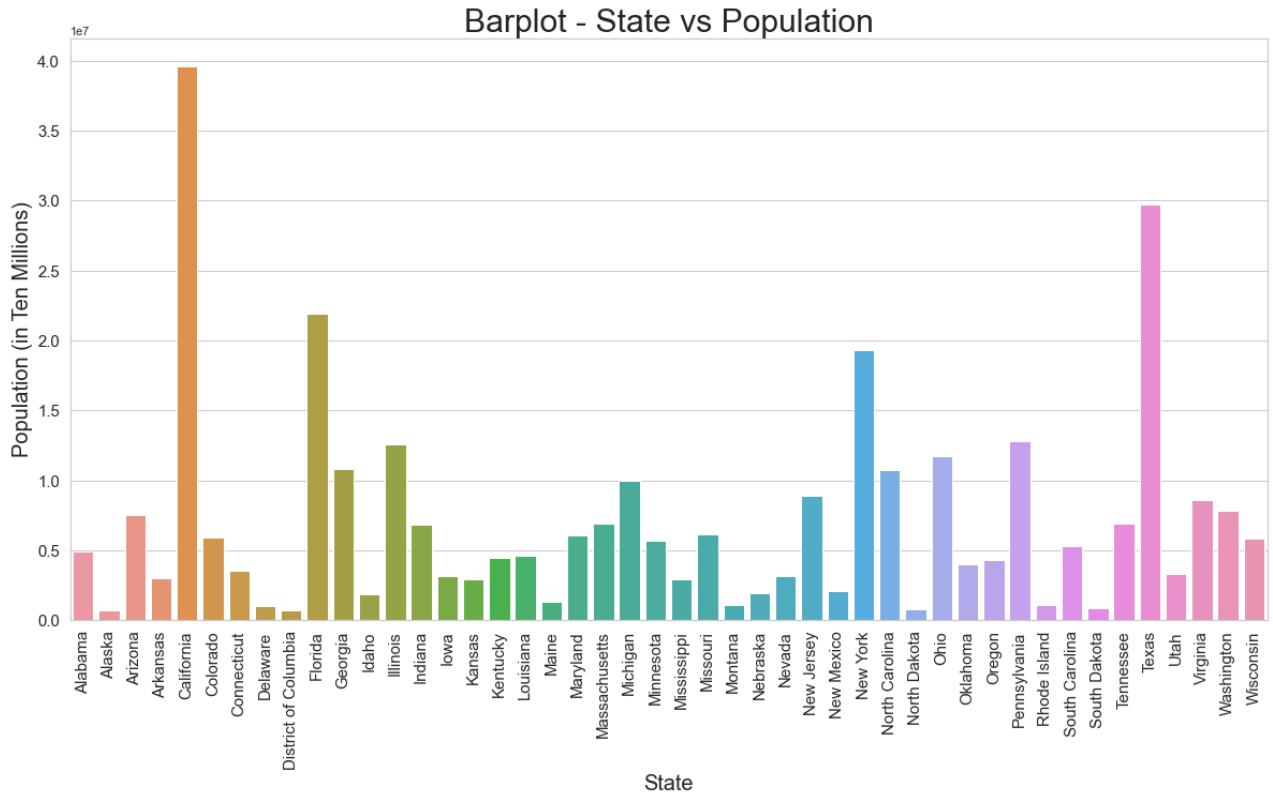
Loading [MathJax]/extensions/Safe.js size=15)

```

plt.xticks(fontsize=15, rotation=90)
plt.title('Barplot - State vs Population', fontsize=30)

```

Out[]: Text(0.5, 1.0, 'Barplot - State vs Population')



- Sorting states by population. Interesting enough that CA TX FL have most housing activities too.
- AZ has the 4th housing count, but the population is 14th. So AZ has a lot more real estate activities per person.

In []:

```

# Figure size
plt.figure(figsize=(20,10))
# Pointplot
# _ = sns.pointplot(data=state_df,x="fullState",y="GDP", color='orange', label='GDP')
# _ = sns.pointplot(data=state_df,x="fullState",y="Spending", color='red', label="Spendi
# _ = sns.pointplot(data=state_df,x="fullState",y="Income", color='Green', label="Incom
# https://stackoverflow.com/questions/69933566/multiple-seaborn-pointplots-not-showing-
# for pointplots legend doesnt show the right color, does not have label, so you have t

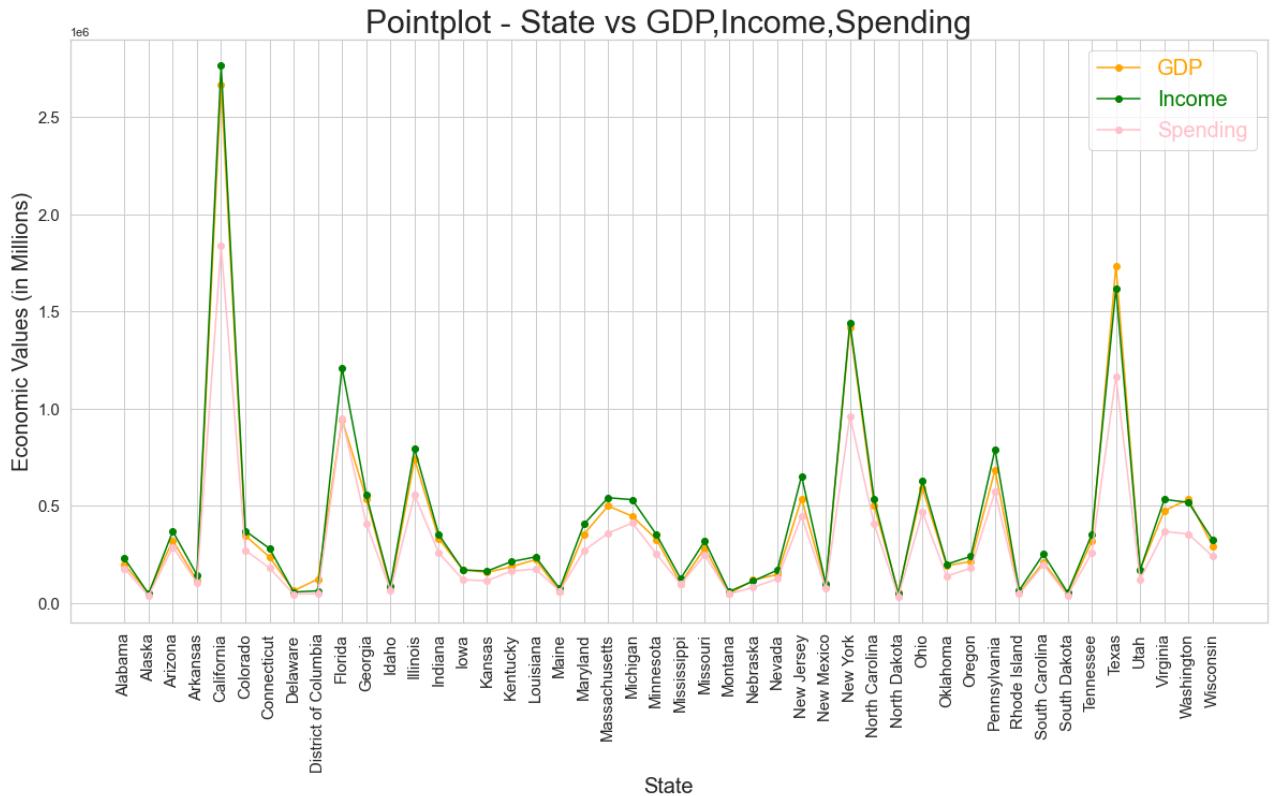
# using matplotlib to draw
plt.plot(state_df.fullState, state_df.GDP, 'o-', label="GDP", color='orange')
plt.plot(state_df.fullState, state_df.Income, 'o-', label="Income", color='green')
plt.plot(state_df.fullState, state_df.Spending, 'o-', label="Spending", color='pink')

# Annotation
plt.xlabel('State', fontsize=20)
plt.ylabel('Economic Values (in Millions)', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15, rotation=90)
plt.title('Pointplot - State vs GDP,Income,Spending', fontsize=30)

```

Loading [MathJax]/extensions/Safe.js ls=['GDP','Income','Spending'],labelcolor=['orange','green','pink'],loc=

```
Out[ ]: <matplotlib.legend.Legend at 0x7ffc74800b20>
```



California have more Income, less GDP. So produce less goods and earn more.
Texas have more GDP and less Income. So produce more goods and earn less.

```
In [ ]:
```

```
# Dataframe - top 10 GDP states and reset index
top_gdp_df = state_df.sort_values(by='GDP', ascending=False).head(10).reset_index(drop=True)
top_gdp_df['Size']=top_gdp_df['Size'].apply(round)
top_gdp_size_list=list(top_gdp_df.Size.unique())

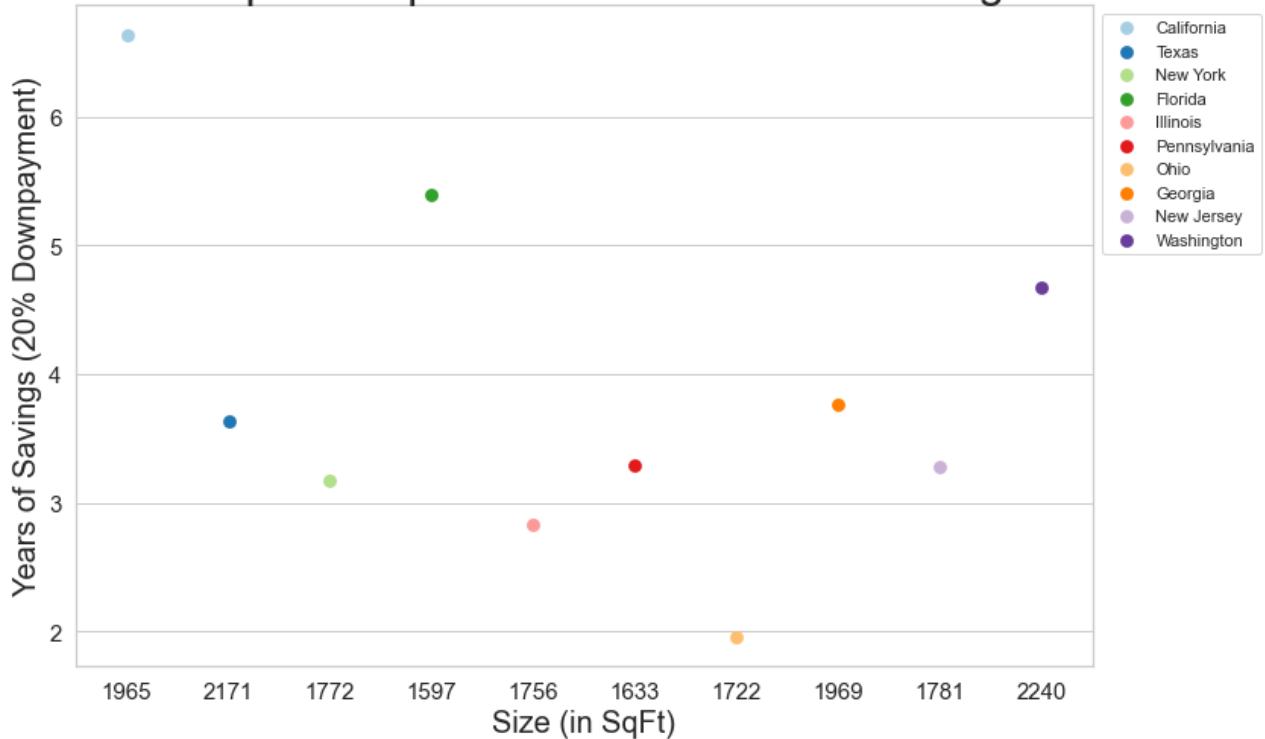
#Fig size
plt.figure(figsize=(12,8))

# Pointplot
a = sns.pointplot(data=top_gdp_df, x='Size', y="Years_of_Savings", hue="fullState", palette="Set1")

# Annotation
plt.xlabel('Size (in SqFt)', fontsize=20)
plt.ylabel('Years of Savings (20% Downpayment)', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Pointplot - Top 10 State vs Years of Savings', fontsize=30)
plt.legend(bbox_to_anchor=(1.175,1))
# for i in range(len(top_gdp_size_list)):
#     plt.text(x=top_gdp_df.Size[i], y = top_gdp_df.Years_of_Savings[i], s=top_gdp_df.fullState[i])
```

```
Out[ ]: <matplotlib.legend.Legend at 0x7ffc6da6f040>
```

Pointplot - Top 10 State vs Years of Savings



(Sorted by State GDP, Left to Right : #1 to #10) Years of Savings of top 10 States with highest GDP (Strong Economy)

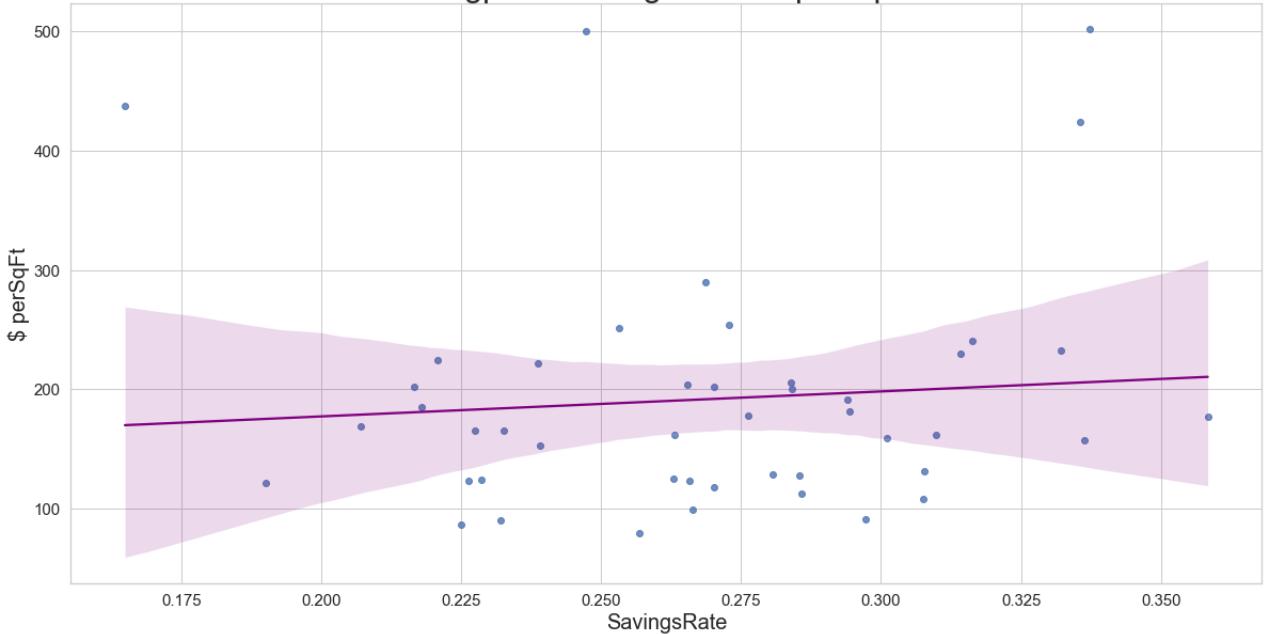
In []:

```
# Regression plot of SavingsRate vs perSqFt
plt.figure(figsize=(20,10))
g = sns.regplot(data = state_df, y = 'perSqFt', x = 'SavingsRate', line_kws={'color':'purple'})
plt.ticklabel_format(style='plain', axis='y')
plt.ticklabel_format(style='plain', axis='x')
# The States that have higher SavingRate will pay higher perSqFt for the houses. Note: This is a regression plot, so the order of the points does not matter.

# Annotation
plt.xlabel('SavingsRate', fontsize=20)
plt.ylabel('$ perSqFt', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Regplot - SavingsRate vs $ perSqFt', fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Regplot - SavingsRate vs perSqFt')

Regplot - SavingsRate vs perSqFt



- There is slight correlation between SavingsRate and perSqFt.
 - As people saves more money, they pay higher perSqFt for the house.
- Note:** This doesnt mean they will be better furnished houses, it may be geographic area.
We dont have enough data/evidence to prove this.

In []:

```
# Barplot
plt.figure(figsize=(20,10))

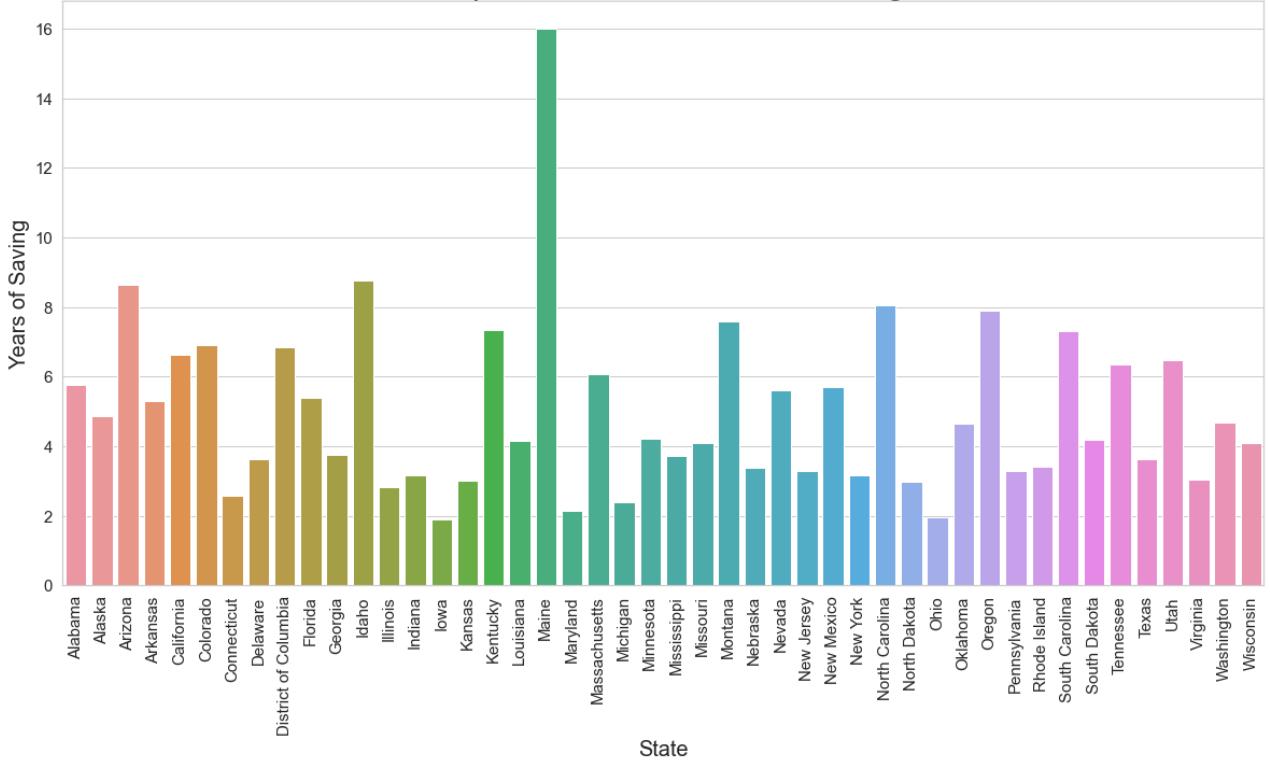
# sorted_df = state_df.sort_values(by='Years_of_Savings',ascending=True)
sns.barplot(data=state_df,x='fullState',y='Years_of_Savings')

# Annotation
plt.xlabel('State', fontsize=20)
plt.ylabel('Years of Saving', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15,rotation=90)
plt.title('Barplot - State vs Years of Saving', fontsize=30)

# value_array = state_df['Years_of_Savings'].to_list()
# for i in range(len(value_array)):
#     plt.text(x=state_df['fullState'][i],y=state_df["Years_of_Savings"][i]+0.1, s=stat
```

Out[]: Text(0.5, 1.0, 'Barplot - State vs Years of Saving')

Barplot - State vs Years of Saving



- Number of years to save in order to purchase a house with 20% down payment.

In []:

```
state_df.head()
```

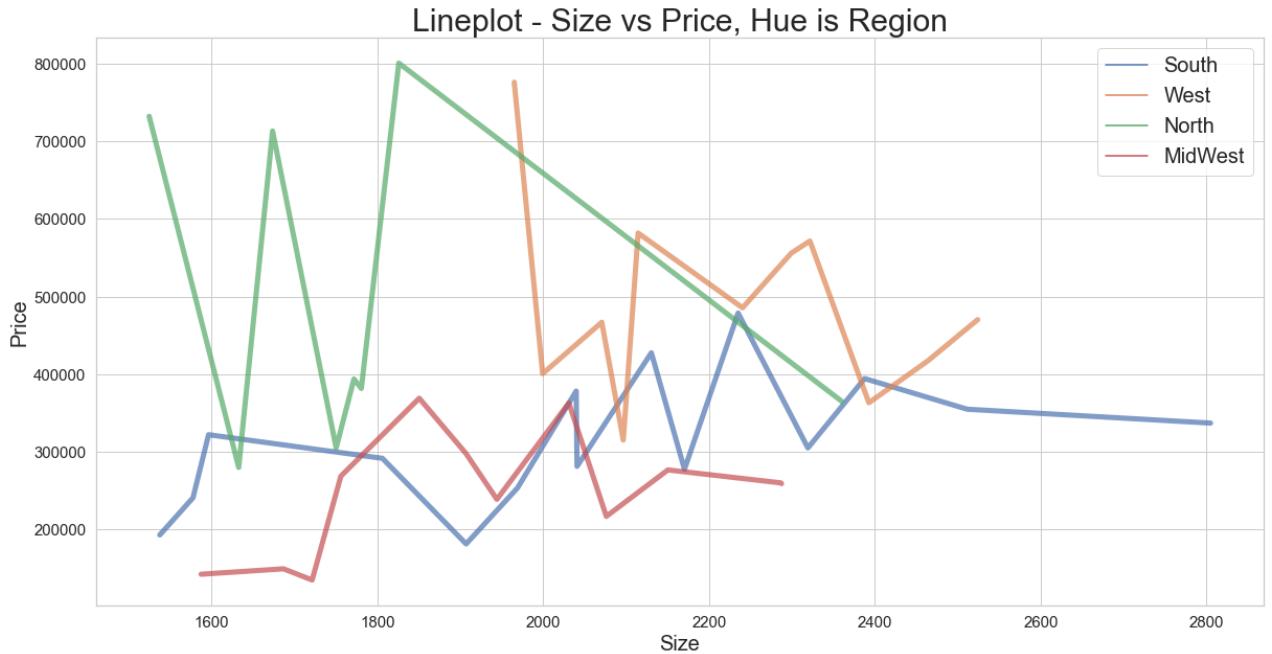
Out[]:

	fullState	GDP	Spending	Population	Income	Region	State	Bathrooms	Bedrooms
0	Alabama	196906.10	176479.80	4934193	228748.80	South	AL	2.62	3.38
1	Alaska	50161.00	35635.70	724357	46430.30	West	AK	2.97	4.59
2	Arizona	320550.60	287090.10	7520103	368458.60	West	AZ	2.51	3.34
3	Arkansas	114943.50	104488.80	3033946	143147.90	South	AR	2.96	3.64
4	California	2663665.90	1835980.60	39613493	2763312.00	West	CA	2.56	3.44

In []:

```
# Lineplot Size vs Price
plt.figure(figsize=(20,10))
sns.lineplot(x="Size", y='Price', hue='Region', data=state_df, lw=5, alpha=0.7)

# Annotation
plt.xlabel('Size', fontsize=20)
plt.ylabel('Price', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Lineplot - Size vs Price, Hue is Region', fontsize=30)
plt.legend(prop={'size': 20})
```



- Blue - South Region is kind of positive correlated, but Price does not exceed \$500k and have all kinds of Sizes
- Orange - West Region has higher prices and larger Sizes
- Green - North Region has high prices in smaller Sizes
- Red - MidWest has low prices and smaller Sizes

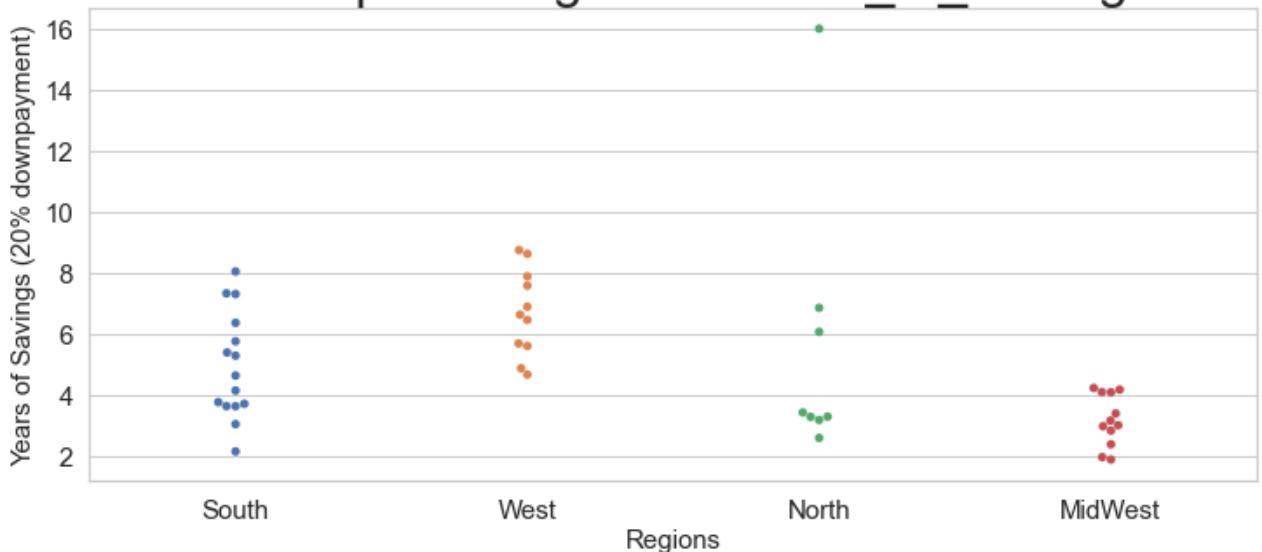
In []:

```
plt.figure(figsize=(12,5))
sns.swarmplot(x='Region',y='Years_of_Savings',data=state_df)
# sns.violinplot(data=state_df['Years_of_Savings'],orient='horizontal',palette='Set2')

# Annotation
plt.xlabel('Regions', fontsize=15)
plt.ylabel('Years of Savings (20% downpayment)', fontsize=15)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Swarmplot - Region vs Years_of_Savings', fontsize=30)
```

Out[]: Text(0.5, 1.0, 'Swarmplot - Region vs Years_of_Savings')

Swarmplot - Region vs Years_of_Savings



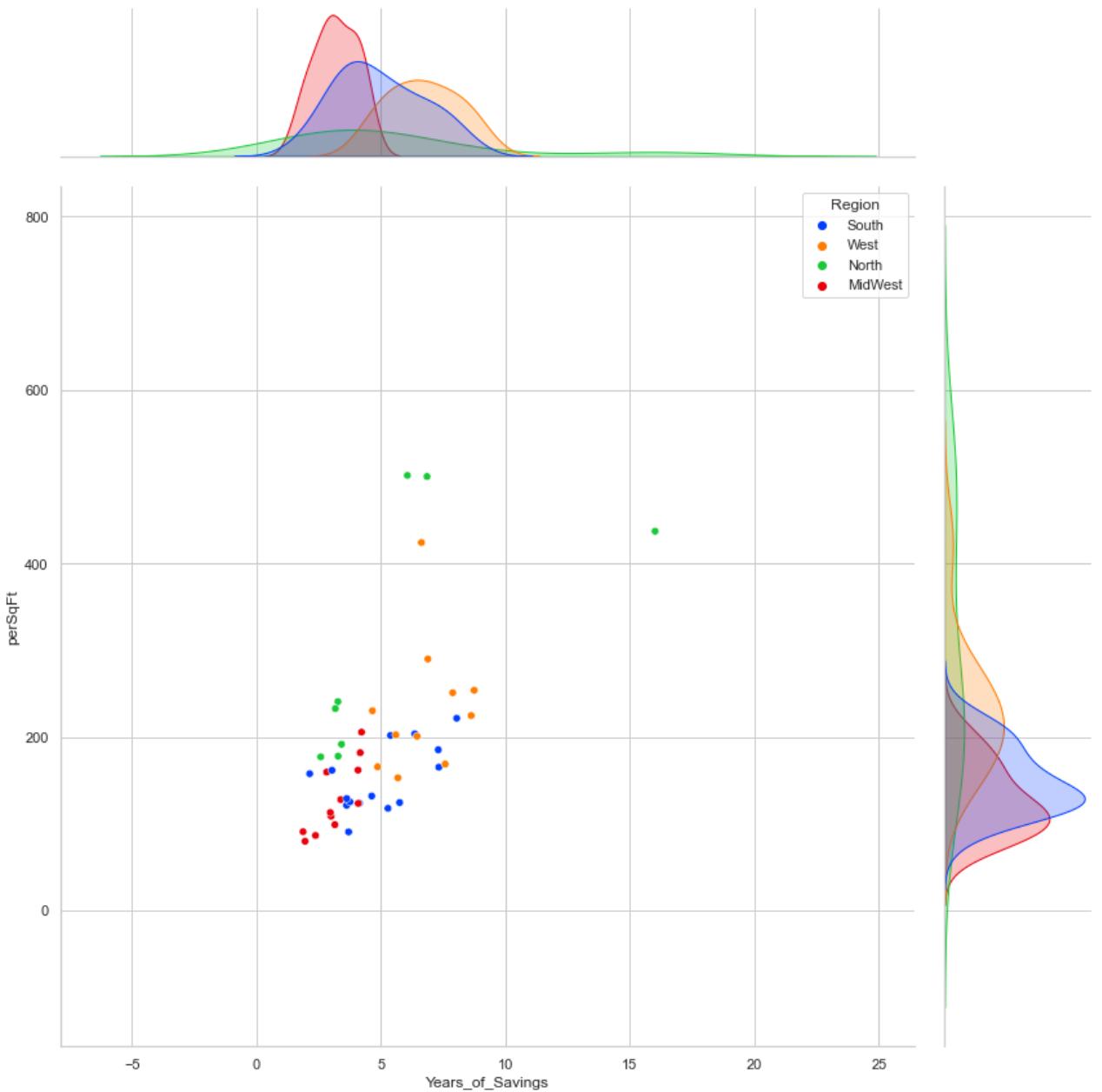
- **South Region is around 3 to 8 Years of Saving**
- **West Region have higher 4.5 to 9 Years of Saving**
- **North Region have outliers, but most are almost near 4 Years of Saving**
- **MidWest Region have overall lowest, and are around 1-4 Years of Saving**

In []:

```
# sns.jointplot(data=state_df,x='Years_of_Savings',y='Price',size='Population')
plt.figure(figsize=(20,10))
sns.jointplot(data=state_df,y='perSqFt',x='Years_of_Savings', hue='Region', palette='br
```

Out[]:

```
<seaborn.axisgrid.JointGrid at 0x7ffc82f943d0>
<Figure size 1440x720 with 0 Axes>
```



Year of Savings are mostly around 5 years.

Price is around 200k to 600k.

West has higher price range around 600k to 800k

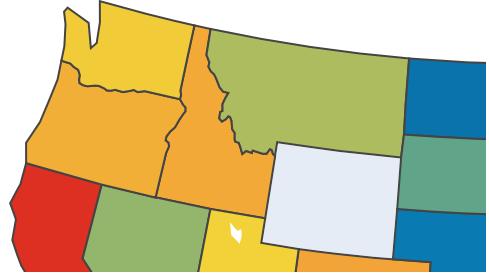
Preview the distribution of data and where they lie mostly.

In []:

```
# Choropleth graph
data = dict(type = 'choropleth', colorscale='Portland', locations=state_df['State'], locationmode='USA-states',
            choromap = go.Figure(data = [data], layout = dict(geo = {'scope':'usa'}, title_text='<b>House Price by State</b>'))
            iplot(choromap, validate=False)
```

House Price of each State (Hover for Avg Price)

Loading [MathJax]/extensions/Safe.js

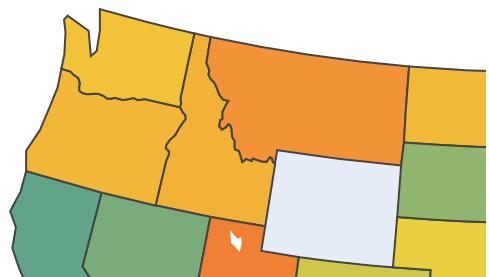


CA, MA have highest housing price across the United States. Additionally most western regions are pretty high.

```
In [ ]: # Choropleth graph  
data = dict(type = 'choropleth', colorscale='Portland', locations=state_df['State'], locationmode='USA-states',  
            choromap = go.Figure(data = [data], layout = dict(geo = {'scope':'usa'}, title_text='<b>Housing Price</b>'))  
iplot(choromap, validate=False)
```

House Size of each State

(Hover for Avg Size)



The Western Region is overall larger in size for living.

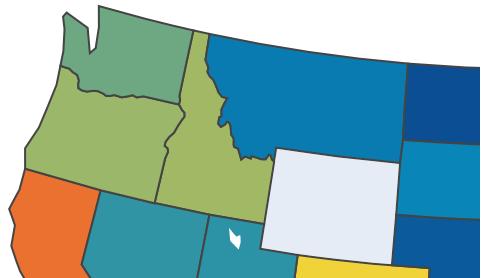
The Eastern Region is smaller in size for living.

In []:

```
# Choropleth graph
data = dict(type = 'choropleth', colorscale='Portland', locations=state_df['State'], location_labels=state_df['State'], z=state_df['per_sqft'], zmin=100, zmax=1000)
choromap = go.Figure(data = [data], layout = dict(geo_scope = 'usa'), title_text='<b>per_sqft</b> by State')
choromap.show()
```

perSqFt of each State

(Hover for Avg perSqFt)



From a \$ perSqFt point of view, the nation is actually pretty similar. Besides MA, ME, CA, and CO.

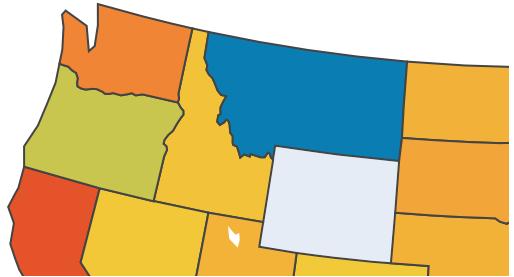
In []:

```
# Choropleth graph
Loading [MathJax]/extensions/Safe.js
data = dict(type = 'choropleth', colorscale='Portland', locations=state_df['State'], location_labels=state_df['State'], z=state_df['per_sqft'], zmin=100, zmax=1000)
choromap = go.Figure(data = [data], layout = dict(geo_scope = 'usa'), title_text='<b>per_sqft</b> by State')
choromap.show()
```

```
choromap = go.Figure(data = [data],layout = dict(geo = {'scope':'usa'},title_text='<b>S  
iplot(choromap,validate=False)
```

SavingsRate of each State

(Hover for Avg SavingsRate)



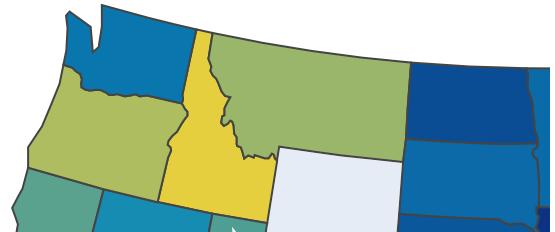
The whole country's SavingsRate is relatively similar, in the range of 25 to 35 %.

In []:

```
# Choropleth graph  
data = dict(type = 'choropleth',colorscale='Portland',locations=state_df['State'],locat  
choromap = go.Figure(data = [data],layout = dict(geo = {'scope':'usa'},title_text='<b>Y  
iplot(choromap,validate=False)
```

Years_of_Savings of each State

(Hover for Years_of_Savings)



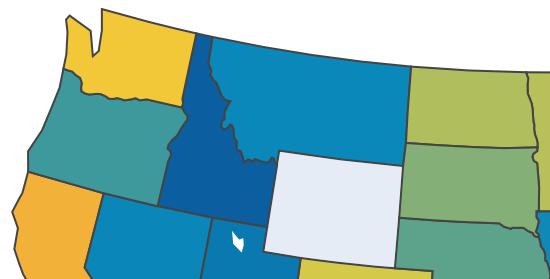


- West Region are mostly brighter/higher than other regions.

In []:

```
# Choropleth graph
data = dict(type = 'choropleth', colorscale='Portland', locations=state_df['State'], location_labels=state_df['State'], z=state_df['Avg_Income'], colorscale='Portland')
choromap = go.Figure(data = [data], layout = dict(geo = {'scope':'usa'}, title_text='<b>Avg Individual's Annual Income by State</b>'))
choromap.show()
iplot(choromap, validate=False)
```

Avg Individual's Annual Income of each State
(Hover for per Annual Income)



Colorado and Illinois are proportionally higher for rural areas.

In [95]:

```
# Regplot for GDP vs Price
plt.figure(figsize=(20,10))
g = sns.regplot(data = state_df, x = 'GDP', y = 'Price', color='orange', line_kws={'color': 'darkblue', 'dash': [4, 4]}, scatter_kws={'color': 'orange'})
plt.ticklabel_format(style='plain', axis='y')
plt.ticklabel_format(style='plain', axis='x')
# g.set(xlim = (0,1000000))

# Annotation
plt.xlabel('GDP', fontsize=20)
plt.ylabel('Price', fontsize = 20)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Regplot - GDP vs Price', fontsize=30)
```

Out[95]: Text(0.5, 1.0, 'Regplot - GDP vs Price')

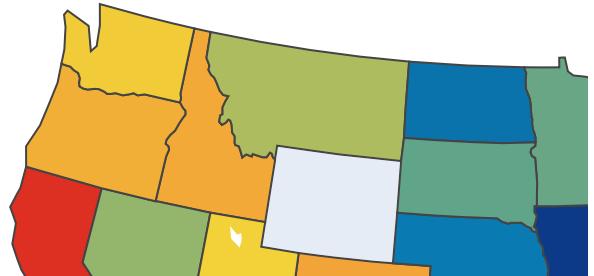


- There is slight correlation between GDP and House Price.

In [93]:

```
#####
#Choropleth Graph of Desired Annual Income per Individual
data = dict(type = 'choropleth', colorscale='Portland', locations=state_df['State'], locationmode='USA-states',
            choromap = go.Figure(data = [data], layout = dict(geo = {'scope':'usa'}, title_text='<b>Desired Annual Income</b>'),
            iplot(choromap, validate=False)
```

Desired Annual Income of each State (Hover for Desired Annual Income)

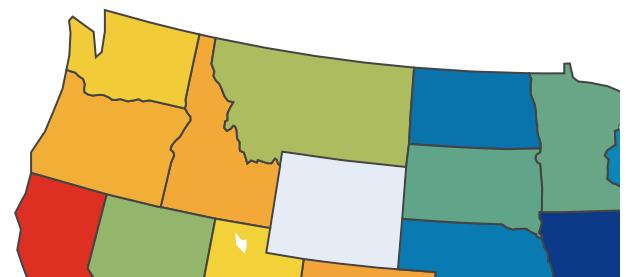


For highest desired income: California, New York, and Maine stand out on the top of this list.

In [94]:

```
#Choropleth Graph of Desired Annual Income per Individual
data = dict(type = 'choropleth', colorscale='Portland', locations=state_df['State'], location_labels=state_df['State'], hover_name=state_df['State'], hover_color=True, color_continuous_scale='Portland')
choromap = go.Figure(data = [data], layout = dict(geo = {'scope':'usa'}, title_text='<b>Desired Annual Income of each State</b>'))
choromap.show()
iplot(choromap, validate=False)
```

Desired Annual Income of each State (Hover for Monthly Mortgage Payment)



For highest average monthly payment: California, New York, and Maine stand out on the top of this list.

4. Testing Hypothesis, ANOVA

Importing Libraries

```
In [ ]: # Importing Libraries
from scipy.stats import normaltest
from statsmodels.stats.weightstats import ztest
import scipy.stats as stats
from scipy.stats import chi2_contingency
```

Pearson Coefficient vs P-Value

-- using correlation to test if variables are statistically significant

```
In [ ]: # Calculate pearson coefficient and pvalue
pearson_coef, p_value = stats.pearsonr(house_df['Bathrooms'], house_df['Price'])

# print out results
print(f'The Pearson Correlation Coefficient is {pearson_coef:.4f} with a P-value of {p_}
if p_value < 0.001:
    print('Since p-value is < 0.001, the correlation between Bathrooms and Price is sta
else:
    print('Since p-value is > 0.001, the correlation between Bathrooms and Price is not
```

The Pearson Correlation Coefficient is 0.4857 with a P-value of 0.0000
Since p-value is < 0.001, the correlation between Bathrooms and Price is statistically significant, although the linear relationship isn't extremely strong.

Z-Test Hypothesis

-- Hypothesis and statistical test that assumes normal distribution to determine whether two population means are different. Variances are known and sample size is large.

$$H_0 : \mu \leq \mu_0$$

California house prices is higher than 600k

$$H_1 : \mu > \mu_0$$

California house prices is lower than 600k

```
In [ ]: # Calculate test statistic and pvalue
(test_statistic, p_value) = ztest(house_df[house_df['State'] == 'CA']['Price'], value=6)

# print out results
print(f'The Test Statistic is {test_statistic:.4f} with a P-value of {p_value:.4f}')
if p_value < 0.005:
    print('Since p-value is < 0.01, we do not retain the null hypothesis.')
else:
    print('Since p-value is > 0.01, we retain the null hypothesis')
```

The Test Statistic is 30.5485 with a P-value of 1.0000
 Since p-value is > 0.01, we retain the null hypothesis

ANOVA

-- determine whether the differences between groups of data are statistically significant.

H_0 : Region and House Prices are not statistically different.

H_1 : Region and House Prices are statistically different.

```
In [ ]: anova_df = house_df[['Region', 'Price']]
groupby_anova = anova_df.groupby(['Region'])
F_oneway, p_value = stats.f_oneway(groupby_anova.get_group('West')['Price'], groupby_anova.get_group('North')['Price'])
print('ANOVA results: F-Oneway = {:.4f}, P-Value = {:.4f}'.format(F_oneway,p_value))

F_oneway, p_value = stats.f_oneway(groupby_anova.get_group('North')['Price'], groupby_anova.get_group('South')['Price'])
print('ANOVA results: F-Oneway = {:.4f}, P-Value = {:.4f}'.format(F_oneway,p_value))

F_oneway, p_value = stats.f_oneway(groupby_anova.get_group('South')['Price'], groupby_anova.get_group('West')['Price'])
print('ANOVA results: F-Oneway = {:.4f}, P-Value = {:.4f}'.format(F_oneway,p_value))

F_oneway, p_value = stats.f_oneway(groupby_anova.get_group('South')['Price'], groupby_anova.get_group('East')['Price'])
print('ANOVA results: F-Oneway = {:.4f}, P-Value = {:.4f}'.format(F_oneway,p_value))
```

ANOVA results: F-Oneway = 6028.2917, P-Value = 0.0000.
 ANOVA results: F-Oneway = 2060.2618, P-Value = 0.0000.
 ANOVA results: F-Oneway = 1265.6909, P-Value = 0.0000.
 ANOVA results: F-Oneway = 7083.4905, P-Value = 0.0000.
 ANOVA results: F-Oneway = 623.1008, P-Value = 0.0000.

As you can see all P-Value < 0.05, all region house prices are statistically different. Strong correlation between region variable and price.

Chi-Squared Test

-- determine whether there is a statistically significant and dependent on each other.

H_0 : SavingsRate and Bedrooms are not statistically different.

H_1 : SavingsRate and Bedrooms are statistically different.

```

# See if Savings Rate and Bedrooms are dependent
chi2_df = state_df[['SavingsRate', 'Bedrooms']]
stat, p, dof, expected = chi2_contingency(chi2_df)

# H0 = SavingsRate and Bedrooms are independent
# Ha = SavingsRate and Bedrooms are dependent

# print out results
print(f'The Test Statistic is {stat:.4f} with a P-value of {p:.4f}')
if p_value < 0.05:
    print('Since p-value is < 0.05, they are dependent.') # We Reject H0, statistically
else:
    print('Since p-value is > 0.05, they are independent')

```

The Test Statistic is 0.2829 with a P-value of 1.0000
 Since p-value is < 0.05, they are dependent.

5. Conclusion

Questions and Answers

1. What is average and median house price across all USA? (Jack)

- Median are: 3 Bedrooms, 2 Bathrooms, 1702 SqFt, Price of 325,000
- Average are: 3.3 Bedrooms, 2.5 Bathrooms, 2000 SqFt, Price of 407,000

2. Is Size highly correlated with Price? If not what's most correlated with Price. (Jack)

- Size and Price correlation is 48% - Highest correlation is Bathroom, 49%

3. What's the primary type of houses people prefer in each state? (Graph answer, house_df graph) (Jack)

- Primary Type is Single Family Residence

Massachusetts, Alaska, District of Columbia are primarily Condos.

Maryland, Pennsylvania, Delaware are primarily Townhouses or have huge % in Townhouses.

4. What states have the largest and smallest avg size? (Jack)

- States with Largest House Size

State AR have average of 2805 square feet.

State UT have average of 2524 square feet.

State OK have average of 2511 square feet.

- States with Smallest House Size

State DC have average of 1525 square feet.

State DE have average of 1538 square feet.

State MD have average of 1578 square feet.

5. What states have the highest and lowest \$ perSqFt? (Jack)

- Highest States \$ perSqFt

average of \$501.47 per square feet

State DC have average of \$500.22 per square feet

State ME have average of \$437.12 per square feet

- Lowest States \$ perSqFt

State OH have average of \$79.80 per square feet

State MI have average of \$86.42 per square feet

State MS have average of \$90.56 per square feet

6. Find out each State's: Price, Size, perSqft, Annual Income, SavingsRate, Years of Savings to

Buy a House(20% down). (Jack)

- See choropleth map

7. Since I live in California, what's California house avg like? (Jack)

6.6 years of saving to purchase a house with 20% down at 33.5% Savings Rate

70k perAnnualIncome

nearly 2000 SqFt at 421 perSqFt

770k Avg House Price

No.1 in GDP across USA

8. Relationship between Regions and Price. (Jack)

Yes, different Regions have clear Price difference and Size difference.

9. Is there huge difference between House Type and Price? (Jack)

Yes, most Townhouses across US may be under-valued.

10. Lets say you are a prospective buyer with a salary of \$55,000; which different regions are you more likely to buy in and how much are you paying (assuming you put 20% down and have a 4% interest rate)? (Alec)

General rule of thumb for calculating how much house you can buy with your salary is to multiply by the minimum of 2.5 and max of 3, where the house you can afford can fall into the range of these two houses

11. What does the distribution of the size of houses in each region? (Alec)

Knowing the Size categories, we can create a multi-index dataframe with regions and size categories.

Interesting Findings

- The \$ perSqFt and Savings rate are positively correlated. The more people save, the higher they will spend more money on \$ perSqFt in houses. (Jack)**
- There is a negative correlation between Individual's Annual Income and Size of House So the more you make the smaller place you live?! Make less money so you live in a big house! (Jack)**
- (Good to target ad) Country's savings rate is mostly 25-35%, since i know how much is in your wallet/savings, i can calculate what kind/price-range of items are appropriate for**

you to spend on. (Jack)

- For some states, you only need work 2-4 years to buy a house. Ex: Iowa (Jack)
- With higher \$ perSqFt, Size will be smaller, seems negatively correlated. (Jack)
- The West Region has higher real estate prices. (Jack)
- In AZ, there are a lot more house activities per person than other states. (Jack)
- Colorado has a high \$ perSqFt compared to average. From another student's project, it indicated that CO has a higher education level. (So it matches the findings) (Jack)
- Maine has high house Price vs Income difference. (Jack)
- Data can be used for many other exploratories

Main Takeaways

1. Townhouses should worth more than condos because they don't have common walls and have independent units. Since Townhouses price is cheaper than Condos, for home buyers, Townhouses might have higher chance to get a bargain/ buyers may obtain more value. (Jack)
2. Try to buy houses with more bathrooms, apparently it will have higher prices. For house flippers, it is good to find cheap houses with more bathrooms, then you can re-sell for more profit after flipping it. (Jack)
3. Top ten strong economies with savings rate and size. (Jack)
4. Given the average price of homes in each state, calculate the average monthly mortgage payments that one has to pay (Assuming that it is 4% interest rate and 20% down) (Alec) Since we know the average price for homes in each state, we can use the monthly mortgage formula $P = L[c(1 + c)n] / [(1+c)n - 1]$.
5. Given that the rule of thumb that you should not spend no more than 30% percent of your income in housing, calculate the desired average yearly income per individual. (Alec) Given the Average Monthly Payment of each state, you can use the Unitary Method in order to calculate the desired yearly income.
6. Given the average mortgage payments and average salary of each state, which state has the highest savings after mortgage payments? (Alec)
Using annual salary and mortgage payments per household we can create a bar plot visualization showing the two differences and we can use column subtraction to find the actual values.

6. References

Data of: Housing Price Data

<https://www.kaggle.com/dataranch/zillow-1>

Data of: Population

<https://worldpopulationreview.com/states>

Data of: 1-GDP 2-Income, 3-Spending

Loading [MathJax]/extensions/Safe.js

[https://apps.bea.gov/iTable/iTable.cfm?
reqid=70&step=1&isuri=1&acrdn=1#reqid=70&step=1&isuri=1&acrdn=1](https://apps.bea.gov/iTable/iTable.cfm?reqid=70&step=1&isuri=1&acrdn=1#reqid=70&step=1&isuri=1&acrdn=1)