DATA INSPECTION

```
import pandas as pd
housing_data = pd.read_csv('/Users/tonydao/Documents/housingMarketProject/global_housing_market.csv')
print(housing_data.head())
```

```
  Country  Year  House Price Index  Rent Index  Affordability Ratio  \
0     USA  2015         117.454012  116.550001             9.587945
1     USA  2016         150.807258   51.440915            11.729189
2     USA  2017         123.194502   70.386040             8.506676
3     USA  2018         131.423444   91.469020             3.418054
4     USA  2019         110.461377   56.837048             9.158097

   Mortgage Rate (%)  Inflation Rate (%)  GDP Growth (%)  \
0           4.493292            1.514121       -0.752044
1           5.662213            1.880204       -0.545400
2           2.197469            2.398940        0.930895
3           4.537724            1.608407       -1.479587
4           3.700762            1.293249        1.961415

   Population Growth (%)  Urbanization Rate (%)  Construction Index
0              -0.796707              85.985284          118.089201
1              -0.358084              69.127267          111.980515
2               0.596245              83.555279           85.973903
3               2.321099              88.968961          134.671788
4              -0.879640              87.279612           90.702399
```

- We first observe our columns labels and data types to see how we can relate cols information together.
- The next would be to see if our data contains any null value.

```
print(housing_data.isnull().sum())
```

```
Country                  0
Year                     0
House Price Index        0
Rent Index               0
Affordability Ratio      0
Mortgage Rate (%)        0
Inflation Rate (%)       0
GDP Growth (%)           0
Population Growth (%)    0
Urbanization Rate (%)    0
Construction Index       0
dtype: int64
```

```
dupIfX = housing_data.duplicated()
print(dupIfX.sum())
```

```
0
```

  - There are no duplicated informations in our data.

```
print(housing_data.dtypes)
```

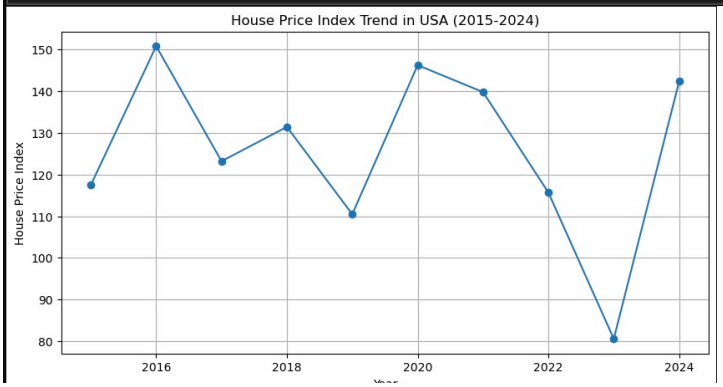
```
Country                   object
Year                       int64
House Price Index        float64
Rent Index               float64
Affordability Ratio      float64
Mortgage Rate (%)        float64
Inflation Rate (%)       float64
GDP Growth (%)           float64
Population Growth (%)    float64
Urbanization Rate (%)    float64
Construction Index       float64
dtype: object
```

  - The col data types correctly represent the cols.

EXPLORATORY ANALYSIS

- What does the house price index look like vs time in years for the USA?

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load your dataset
df = pd.read_csv('/Users/tonydao/Documents/housingMarketProject/global_housing_market.csv')

# Select a country to analyze
country = 'USA'
country_df = df[df['Country'] == country]

plt.figure(figsize=(10,5))
plt.plot(country_df['Year'], country_df['House Price Index'], marker='o')
plt.title(f'House Price Index Trend in {country} (2015-2024)')
plt.xlabel('Year')
plt.ylabel('House Price Index')
plt.grid(True)
plt.show()
```



  - There is a gradually decrease in the the house price index after 2020 until 2023 where a sharp increase occurred.
  - We can apply the Augmented Dickey-Fuller Test to see if our data is stationary or not. This is based on the p-value result.

```python
from statsmodels.tsa.stattools import adfuller
result = adfuller(country_df['House Price Index'])
print(f"ADF Statistic: {result[0]:.2f}")
print(f"p-value: {result[1]:.4f}")
```

```
ADF Statistic: -1.95
p-value: 0.3082
```

5tB28RF3

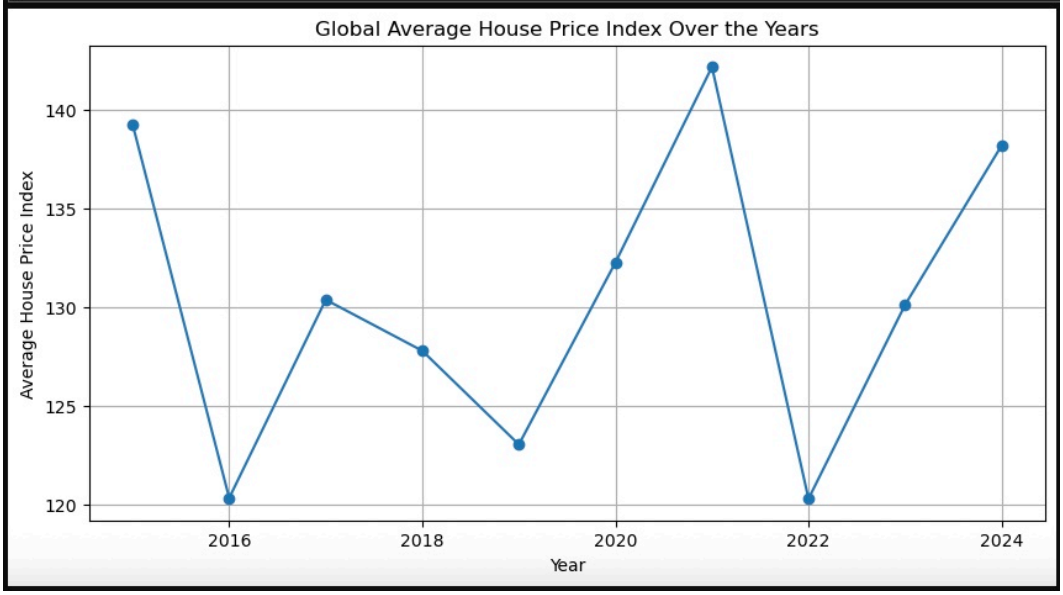
  - Based on a p-value = 0.3, we FAIL TO reject the null hypothesis meaning there is a root in our data. This mean our data is not stationary.
- Since this data is not stationary we apply a differencing technique to remove it.
  - This would entail using the Chow test to ID significant trend shifts

```python
from statsmodels.stats.diagnostic import breaks_cusumolsresid
import numpy as np

# Get residuals (replace with actual residuals if modeling)
residuals = country_df['House Price Index'].diff().dropna()  # Example: use price changes

# Run structural break test
sup_b, pval, crit = breaks_cusumolsresid(residuals, ddof=0)

# Find the index of the maximum cumulative residual
break_idx = np.argmax(np.abs(residuals.cumsum()))  # Actual break point index

# Get the corresponding year
break_year = country_df['Year'].iloc[break_idx]
print(f"Structural break detected in: {break_year}")
```

```
Structural break detected in: 2022
```

  - The result of this test shows a break in 2022.

```
# Group by year and calculate the mean House Price Index
yearly_hpi = df.groupby('Year')['House Price Index'].mean().reset_index()

# Plot the general trend
plt.figure(figsize=(10,5))
plt.plot(yearly_hpi['Year'], yearly_hpi['House Price Index'], marker='o')
plt.title('Global Average House Price Index Over the Years')
plt.xlabel('Year')
plt.ylabel('Average House Price Index')
plt.grid(True)
plt.show()
```

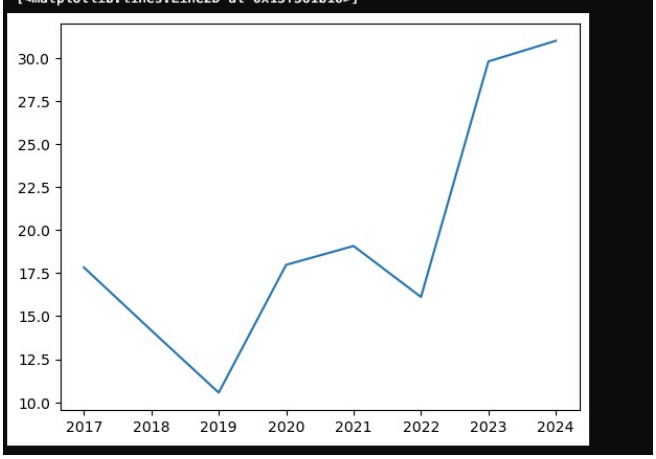


- We can confirm that break visually because there is a steep decrease to 2022 then increase after that.

- Volatility analysis
  - assessing the stability
  - Seeing if external factor (Ex: Covid) had any impact.
  - Conduct a 3-year rolling volatility
    - meaning for each year beginning 2017, it measure the fluctuation of HPI over the current year and the previous two years.

```
country_df['3Y_Volatility'] = country_df['House Price Index'].rolling(3).std()
plt.plot(country_df['Year'], country_df['3Y_Volatility'])
```

```
/var/folders/g7/k8tb4tqx737cssc6482hmgb40000gn/T/ipykernel_738/1737349007.py:1
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/st
  country_df['3Y_Volatility'] = country_df['House Price Index'].rolling(3).std
[<matplotlib.lines.Line2D at 0x15f561b10>]
```




```
country_df['3Y_Volatility']
```

```
0          NaN
1          NaN
2    17.831884
3    14.176930
4    10.561378
5    17.982798
6    19.074038
7    16.115335
8    29.788885
9    30.985391
```

  - Volatility was low from 2017 -2022

- Growth Rate Analysis

```
initial = country_df['House Price Index'].iloc[0]
final = country_df['House Price Index'].iloc[-1]
cagr = (final/initial)**(1/9) - 1
print(f"Annualized growth: {cagr*100:.2f}%")
```

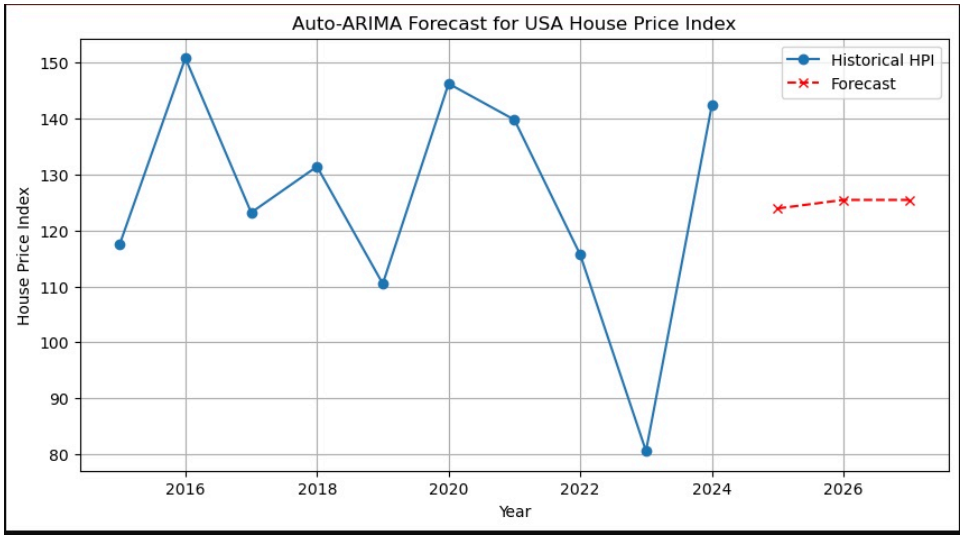
```
Annualized growth: 2.16%
```

- On average, prices increased by 2.16% each year.

- Auto-ARIMA modeling

```
from pmdarima import auto_arima
country = 'USA'
country_df = df[df['Country'] == country].sort_values('Year')
hpi_series = country_df['House Price Index'].values
model = auto_arima(
    hpi_series,
    seasonal=False,
    trace=True,  # Shows the parameter search process
    error_action='ignore',
    suppress_warnings=True
)
print(model.summary())
forecast = model.predict(n_periods=3)
print("Next 3 years forecast:", forecast)
plt.figure(figsize=(10,5))
plt.plot(country_df['Year'], hpi_series, marker='o', label='Historical HPI')
future_years = range(country_df['Year'].iloc[-1] + 1, country_df['Year'].iloc[-1] + 4)
plt.plot(future_years, forecast, marker='x', linestyle='--', color='red', label='Forecast')
plt.title(f'Auto-ARIMA Forecast for {country} House Price Index')
plt.xlabel('Year')
plt.ylabel('House Price Index')
plt.legend()
plt.grid(True)
plt.show()
```

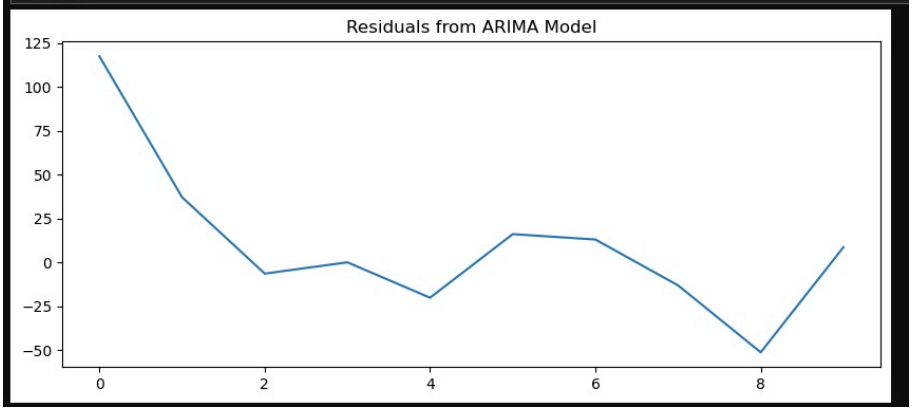
```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=104.091, Time=0.04 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=127.322, Time=0.00 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=104.476, Time=0.00 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=121.019, Time=0.00 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=103.044, Time=0.02 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=inf, Time=0.01 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=inf, Time=0.01 sec
 ARIMA(1,0,3)(0,0,0)[0]             : AIC=inf, Time=0.02 sec
 ARIMA(0,0,3)(0,0,0)[0]             : AIC=inf, Time=0.01 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=103.116, Time=0.02 sec
 ARIMA(2,0,3)(0,0,0)[0]             : AIC=inf, Time=0.03 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.03 sec

Best model:  ARIMA(1,0,2)(0,0,0)[0]
Total fit time: 0.210 seconds
                               SARIMAX Results
==========================================================================
Dep. Variable:                      y   No. Observations:                   10
Model:               SARIMAX(1, 0, 2)   Log Likelihood                -47.522
Date:                Fri, 09 May 2025   AIC                           103.044
Time:                        10:31:30   BIC                           104.254
Sample:                             0   HQIC                          101.716
                                  - 10
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1          1.0000      0.000   4805.726      0.000       1.000       1.000
ma.L1         -1.1795      1.308     -0.902      0.367      -3.743       1.384
ma.L2          0.1949      0.891      0.219      0.827      -1.551       1.941
sigma2       423.4857      0.003   1.31e+05      0.000     423.479     423.492
==========================================================================
Ljung-Box (L1) (Q):                   0.06   Jarque-Bera (JB):                 1.80
Prob(Q):                              0.81   Prob(JB):                         0.41
Heteroskedasticity (H):               2.68   Skew:                            -1.02
Prob(H) (two-sided):                  0.44   Kurtosis:                         3.37
==========================================================================
```



- This is the next 3 year forecast
  - It will decrease then stabilize.

- Residual Analysis

```
residuals = model.resid()
plt.figure(figsize=(10,4))
plt.plot(residuals)
plt.title('Residuals from ARIMA Model')
plt.show()
```



- good small fluctuation near zero.