ASSIGNMENT #1. ANALYZING THE SHARING ECONOMY WITH AIRBNB DATA

Import Packages

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
import os
```

In [ ]:
```
```

Data Preprocessing

In [ ]:
```
drive.mount('/content/drive')
os.chdir("./drive/MyDrive/DSO 574 Assignment")
```

In [ ]:
```
ls
```

```
drive/  sample_data/
```

In [ ]:
```
df = pd.read_csv('Combined Listing Data [Summary].csv')
df.head()
```

Out[ ]:

	Unnamed: 0	id	name	host_id	host_name	neighbourho
0	0	3861476	venicelocalliving.com 341	16577861	John	
1	1	6431889	Turquoise Jewel at Venice Beach	18993265	Greg	
2	2	5754633	Terrane Resort - Oceanfront Suite	15241342	Monica	
3	3	6302541	Hollywood Hot Spot	32776680	Alix	
4	4	4469657	Main Street Retreat	10658835	Brandon	

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['Unnamed: 0', 'id', 'name', 'host_id', 'host_name',
       'neighbourhood_group', 'neighbourhood', 'latitude', 'longitude',
       'room_type', 'price', 'minimum_nights', 'number_of_reviews',
       'last_review', 'reviews_per_month', 'calculated_host_listings_coun
t',
       'availability_365', 'Scrape File'],
      dtype='object')
```

In [ ]:
```
df1 = pd.read_csv('Calendar Data [Temp].csv')
df1.head()
```

Out[ ]:

	Unnamed: 0	listing_id	date	available	file_date	DaysAhead
0	0	26082	2019-07-08	0	2019-07-01	7
1	1	109	2019-07-09	0	2019-07-01	8
2	2	109	2019-07-10	0	2019-07-01	9
3	3	109	2019-07-11	0	2019-07-01	10
4	4	109	2019-07-12	0	2019-07-01	11

In [ ]:
```
df1.columns
```

Out[ ]:
```
Index(['Unnamed: 0', 'listing_id', 'date', 'available', 'file_date',
       'DaysAhead'],
      dtype='object')
```

In [ ]:
```
#use dask to see all col-name
!pip install dask
import dask.dataframe as dd
detail = dd.read_csv('Combined Listing Data [Detailed].csv')
detail
```

```
Requirement already satisfied: dask in /usr/local/lib/python3.11/dist-packag
es (2024.10.0)
Requirement already satisfied: click>=8.1 in /usr/local/lib/python3.11/dist-
packages (from dask) (8.1.8)
Requirement already satisfied: cloudpickle>=3.0.0 in /usr/local/lib/python3.
11/dist-packages (from dask) (3.1.1)
Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.1
1/dist-packages (from dask) (2024.10.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/
dist-packages (from dask) (24.2)
Requirement already satisfied: partd>=1.4.0 in /usr/local/lib/python3.11/dis
t-packages (from dask) (1.4.2)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.11/di
st-packages (from dask) (6.0.2)
Requirement already satisfied: toolz>=0.10.0 in /usr/local/lib/python3.11/di
st-packages (from dask) (0.12.1)
Requirement already satisfied: importlib-metadata>=4.13.0 in /usr/local/lib/
python3.11/dist-packages (from dask) (8.6.1)
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.11/dist-
packages (from importlib-metadata>=4.13.0->dask) (3.21.0)
Requirement already satisfied: locket in /usr/local/lib/python3.11/dist-pack
ages (from partd>=1.4.0->dask) (1.0.0)
```

Out[ ]:

Dask DataFrame Structure:

	Unnamed: 0	id	listing_url	scrape_id	last_scraped	name	s
npartitions=99							
	int64	int64	string	int64	string	string	
	...	...	...	...	...	...	
...	...	...	...	...	...	...	
	...	...	...	...	...	...	
	...	...	...	...	...	...	

Dask Name: to_pyarrow_string, 2 graph layers

In [ ]:

```
#seperate the dataset into smaller part to display
chunk_size = 10000
chunks = pd.read_csv('Combined Listing Data [Detailed].csv', chunksize=chunk
df = next(chunks)

# Drop columns with more than 50% missing values
threshold = len(df) * 0.5
df_cleaned = df.dropna(thresh=threshold, axis=1)
df_cleaned.head()
```

Out[ ]:

	Unnamed: 0	id	listing_url	scrape_id	la
0	0	986942	https://www.airbnb.com/rooms/986942	20150902161235	
1	1	3249753	https://www.airbnb.com/rooms/3249753	20150902161235	
2	2	3250095	https://www.airbnb.com/rooms/3250095	20150902161235	
3	3	3250595	https://www.airbnb.com/rooms/3250595	20150902161235	
4	4	1941493	https://www.airbnb.com/rooms/1941493	20150902161235	

5 rows × 88 columns

In [ ]:
```
df_cleaned.columns
```

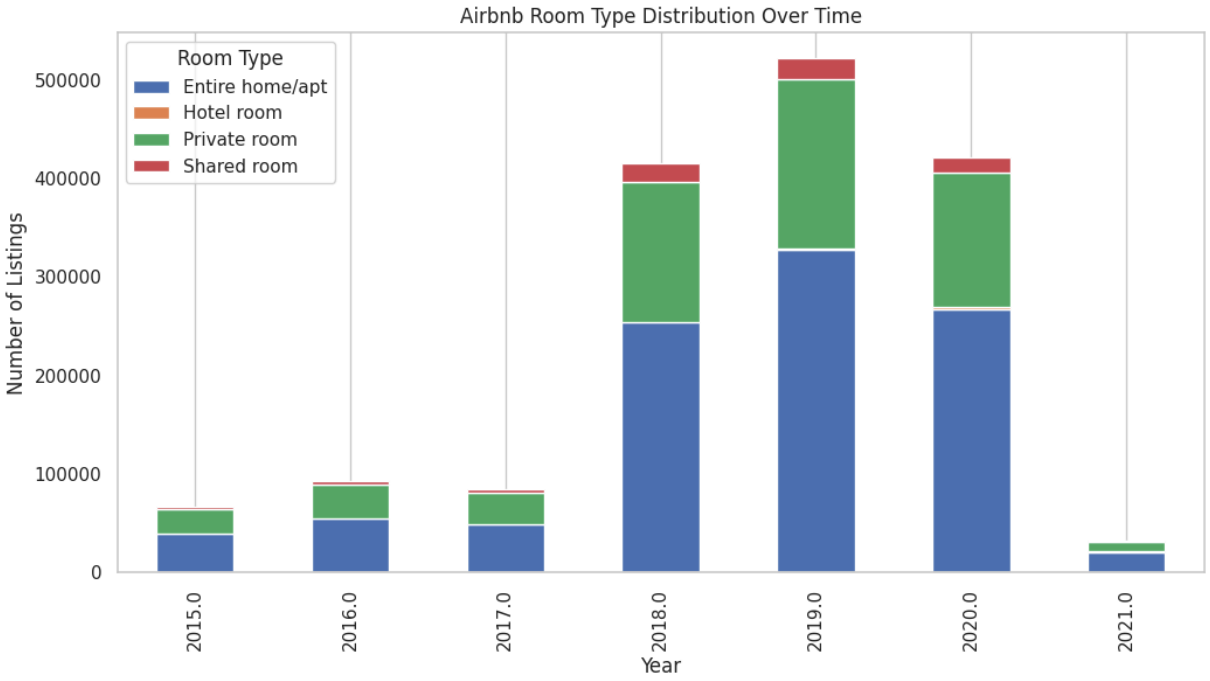
Out[ ]:
```
Index(['Unnamed: 0', 'id', 'listing_url', 'scrape_id', 'last_scraped', 'nam
e',
       'summary', 'space', 'description', 'experiences_offered',
       'neighborhood_overview', 'transit', 'thumbnail_url', 'medium_url',
       'picture_url', 'xl_picture_url', 'host_id', 'host_url', 'host_name',
       'host_since', 'host_location', 'host_about', 'host_response_time',
       'host_response_rate', 'host_acceptance_rate', 'host_is_superhost',
       'host_thumbnail_url', 'host_picture_url', 'host_neighbourhood',
       'host_listings_count', 'host_total_listings_count',
       'host_verifications', 'host_has_profile_pic', 'host_identity_verifie
d',
       'street', 'neighbourhood', 'neighbourhood_cleansed', 'city', 'stat
e',
       'zipcode', 'market', 'smart_location', 'country_code', 'country',
       'latitude', 'longitude', 'is_location_exact', 'property_type',
       'room_type', 'accommodates', 'bathrooms', 'bedrooms', 'beds',
       'bed_type', 'amenities', 'price', 'weekly_price', 'monthly_price',
       'security_deposit', 'cleaning_fee', 'guests_included', 'extra_peopl
e',
       'minimum_nights', 'maximum_nights', 'calendar_updated',
       'has_availability', 'availability_30', 'availability_60',
       'availability_90', 'availability_365', 'calendar_last_scraped',
       'number_of_reviews', 'first_review', 'last_review',
       'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'requires_license', 'instant_bookable',
       'cancellation_policy', 'require_guest_profile_picture',
       'require_guest_phone_verification', 'calculated_host_listings_coun
t',
       'reviews_per_month'],
      dtype='object')
```

In [ ]:
```
# Ensure 'Scrape File' column exists and extract the year
df['year'] = df['Scrape File'].str.extract(r'(\d{4})').astype(float)

# Exclude hotel listings and keep only home-sharing categories
home_sharing_types = ["Entire home/apt", "Private room", "Shared room"]
df_home_sharing = df[df['room_type'].isin(home_sharing_types)]

# Count listings per year (for future use)
listings_per_year = df_home_sharing['year'].value_counts().sort_index()

# Set seaborn style for uniformity
sns.set_theme(style="whitegrid")
```

Exploratory Data Analysis

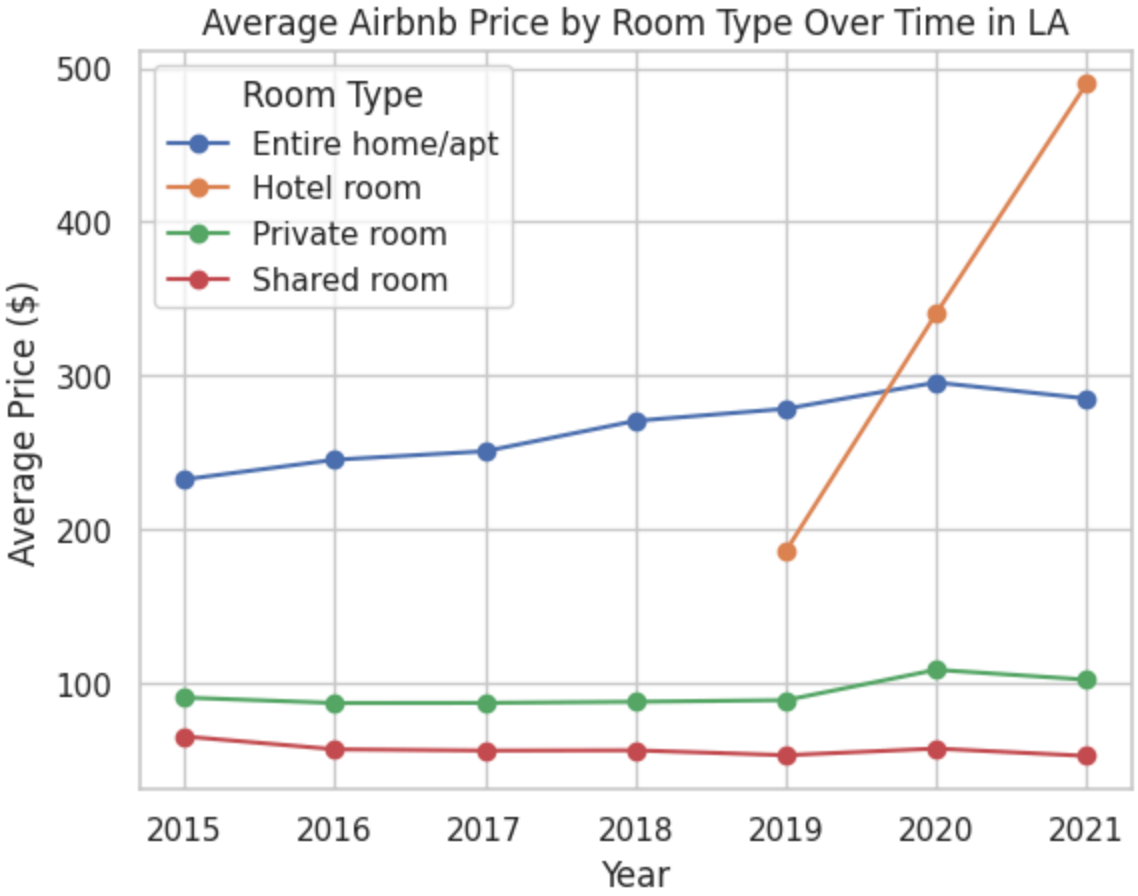
In [ ]:
```
# Pivot Table for Room Type Trends
room_type_trends = df.pivot_table(index='year', columns='room_type', values=

# Plot Room Type Trends Over Time
room_type_trends.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Airbnb Room Type Distribution Over Time")
plt.xlabel("Year")
```

```
plt.ylabel("Number of Listings")
plt.legend(title="Room Type")
plt.grid(axis='y')
plt.show()
```



In [ ]:
```
# Group by year and room_type, then compute average price
price_trends_by_room = df.groupby(['year', 'room_type'])['price'].mean().uns

# Plot the trend
plt.figure(figsize=(12, 6))
price_trends_by_room.plot(marker='o')
plt.title("Average Airbnb Price by Room Type Over Time in LA")
plt.xlabel("Year")
plt.ylabel("Average Price ($)")
plt.legend(title="Room Type")
plt.grid(True)
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



Since hotel rooms represent a commercial use of Airbnb rather than traditional home sharing, we focus only on peer-to-peer room types:

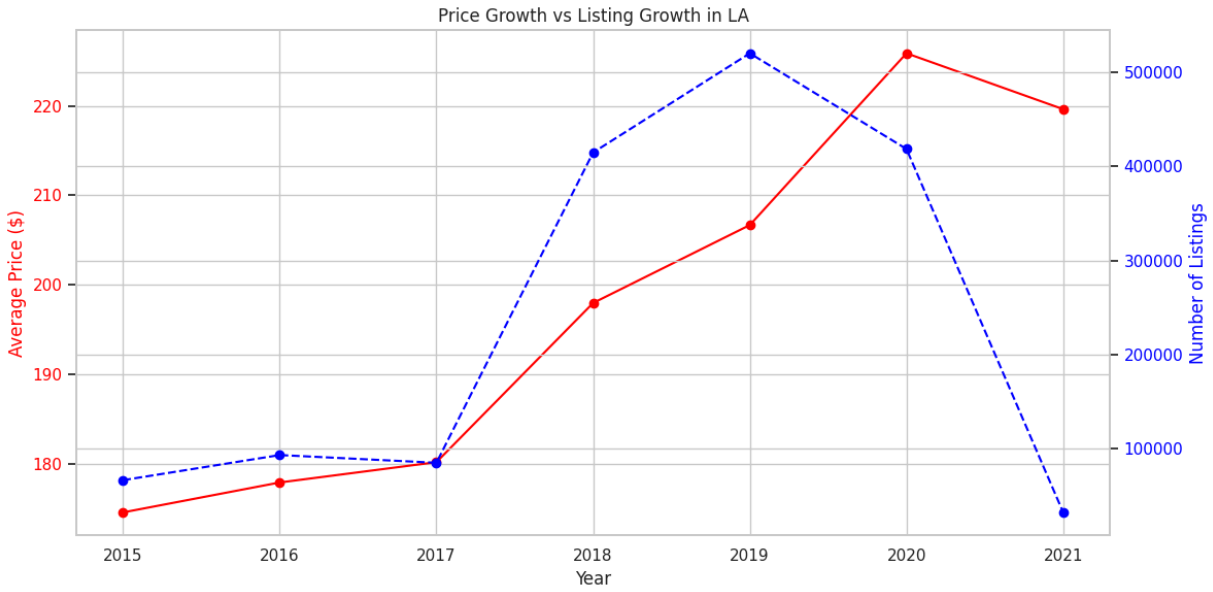
- Entire home/apt
- Private room
- Shared room.

A. Historical Trends in Home Sharing: Insights for Future Predictions

Overview of Home-Sharing Market Growth in LA: Prices vs Listings

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(12, 6))

# First y-axis (Price)
ax1.set_xlabel("Year")
ax1.set_ylabel("Average Price ($)", color="red")
ax1.plot(avg_price_per_year.index, avg_price_per_year.values, marker='o', cc
ax1.tick_params(axis='y', labelcolor="red")

# Second y-axis (Listings)
ax2 = ax1.twinx()
ax2.set_ylabel("Number of Listings", color="blue")
```

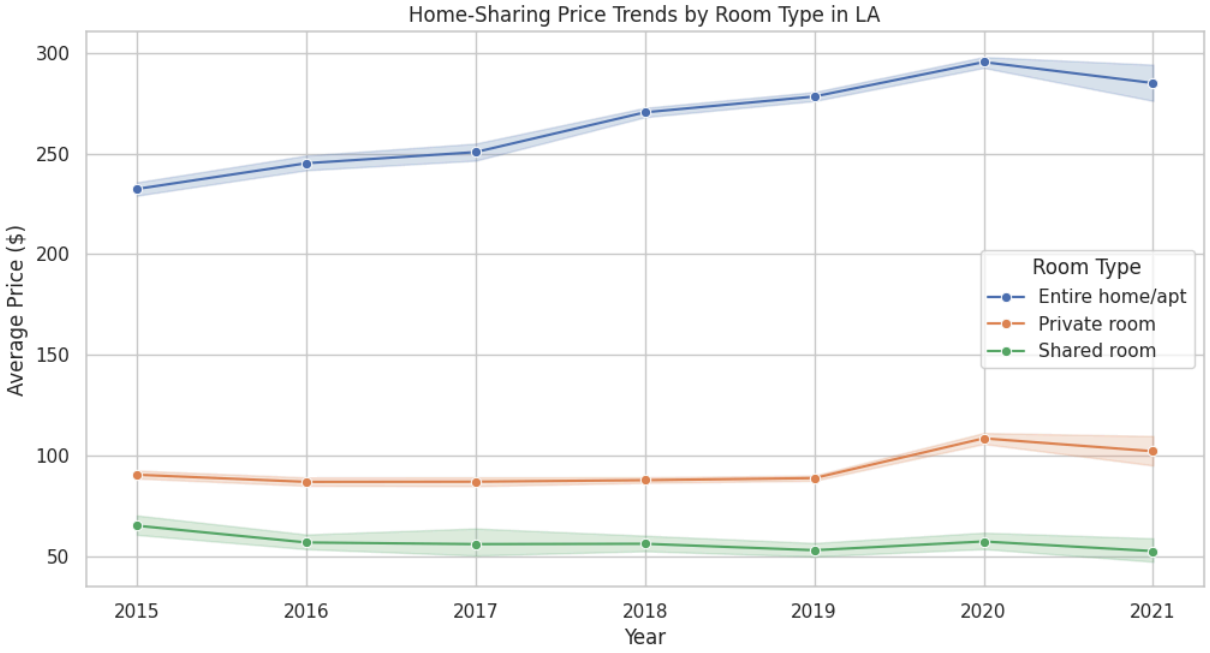
```
ax2.plot(listings_per_year.index, listings_per_year.values, marker='o', colo
ax2.tick_params(axis='y', labelcolor="blue")

plt.title("Price Growth vs Listing Growth in LA")
fig.tight_layout()
plt.show()
```



- Listings Boom & Drop: bold text Surge from 2017-2019, sharp decline in 2021 due to regulations and pandemic.
- Price Resilience: Fewer listings, but prices kept rising, indicating strong demand.
- Future Outlook: Stricter laws may limit supply, keeping prices high, while trends like extended stays reshape the market.

Price Trends Over Time for Home-Sharing

In [ ]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df_home_sharing, x='year', y='price', hue='room_type', mar
plt.title("Home-Sharing Price Trends by Room Type in LA")
plt.xlabel("Year")
plt.ylabel("Average Price ($)")
plt.legend(title="Room Type")
plt.grid(True)
plt.show()
```

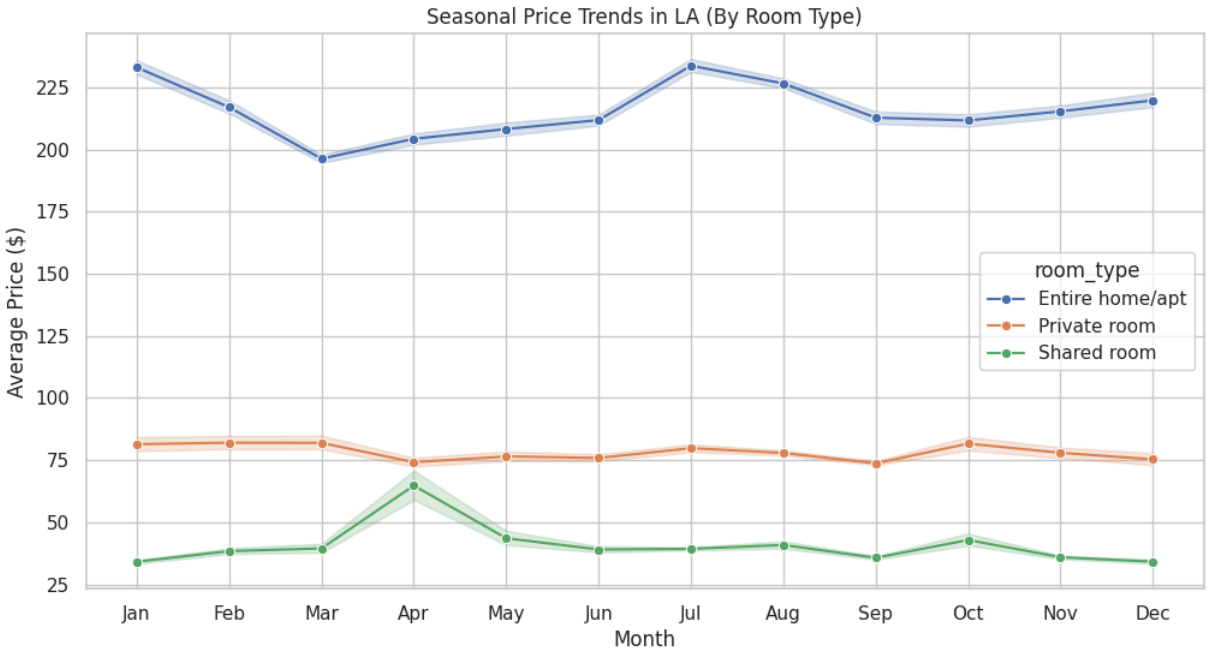


- Entire home/apt prices increased steadily, peaking in 2020 before a slight dip in 2021.
- Private room prices remained stable but spiked in 2020, possibly due to supply constraints.
- Shared room prices fluctuated but stayed low, indicating weaker demand.
- The price surge in 2020 suggests external factors like policy changes or shifts in traveler behavior.

Seasonal Price Trends in LA

In [ ]:
```
df_home_sharing['month'] = pd.to_datetime(df_home_sharing['last_review']).dt

plt.figure(figsize=(12, 6))
sns.lineplot(data=df_home_sharing, x='month', y='price', hue='room_type', ma
plt.title("Seasonal Price Trends in LA (By Room Type)")
plt.xlabel("Month")
plt.ylabel("Average Price ($)")
plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "
plt.grid(True)
plt.show()
```

```
<ipython-input-25-34b8e4c580f0>:1: 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
  df_home_sharing['month'] = pd.to_datetime(df_home_sharing['last_review']).
dt.month
```



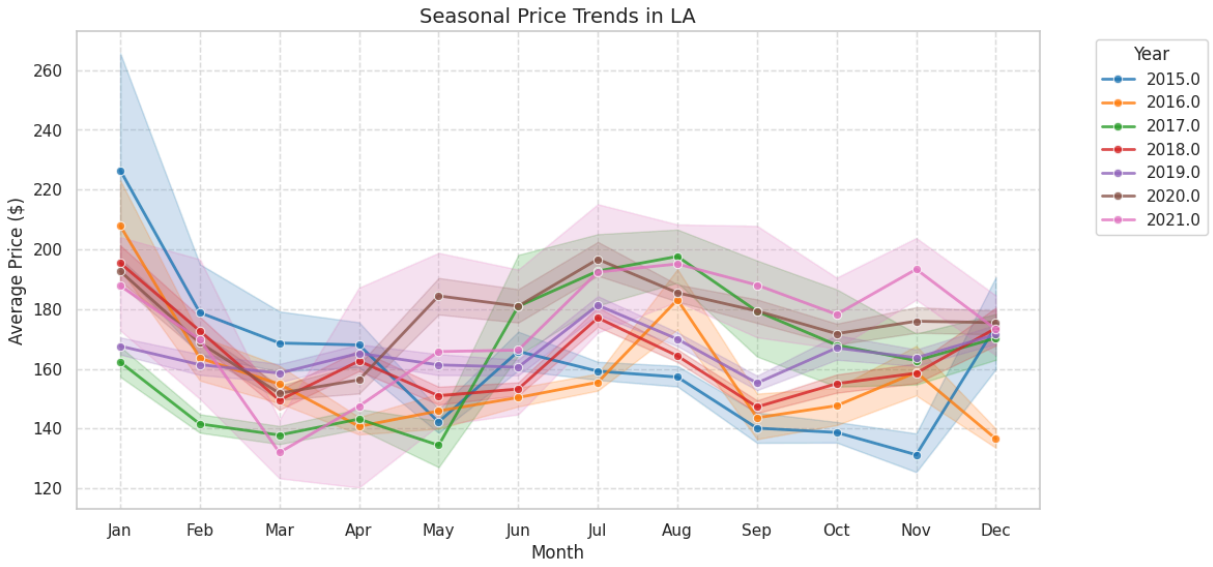
- Prices peak in summer and winter, dip in spring.
- Entire homes show the most seasonal fluctuation.
- Shared rooms spike in April, likely event-driven.
- Trends align with travel demand cycles.

In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Convert 'last_review' to datetime and extract month
df_home_sharing['month'] = pd.to_datetime(df_home_sharing['last_review']).dt

# Set figure size
plt.figure(figsize=(12, 6))

# Line plot with distinct colors
sns.lineplot(
    data=df_home_sharing,
    x='month',
    y='price',
    hue='year',
    marker='o',
    palette='tab10',  # Improved color contrast
    linewidth=2,  # Thicker lines for visibility
    alpha=0.8  # Reduces noise in overlapping lines
)

# Titles and labels
plt.title("Seasonal Price Trends in LA", fontsize=14)
plt.xlabel("Month", fontsize=12)
plt.ylabel("Average Price ($)", fontsize=12)

# Custom x-axis labels for months
```

```
plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
                          "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

# Grid and legend
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title="Year", bbox_to_anchor=(1.05, 1), loc='upper left')  # Move

# Show plot
plt.show()
```

```
<ipython-input-31-2dca7a5bd0e4>:6: 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
  df_home_sharing['month'] = pd.to_datetime(df_home_sharing['last_review']).
dt.month
```



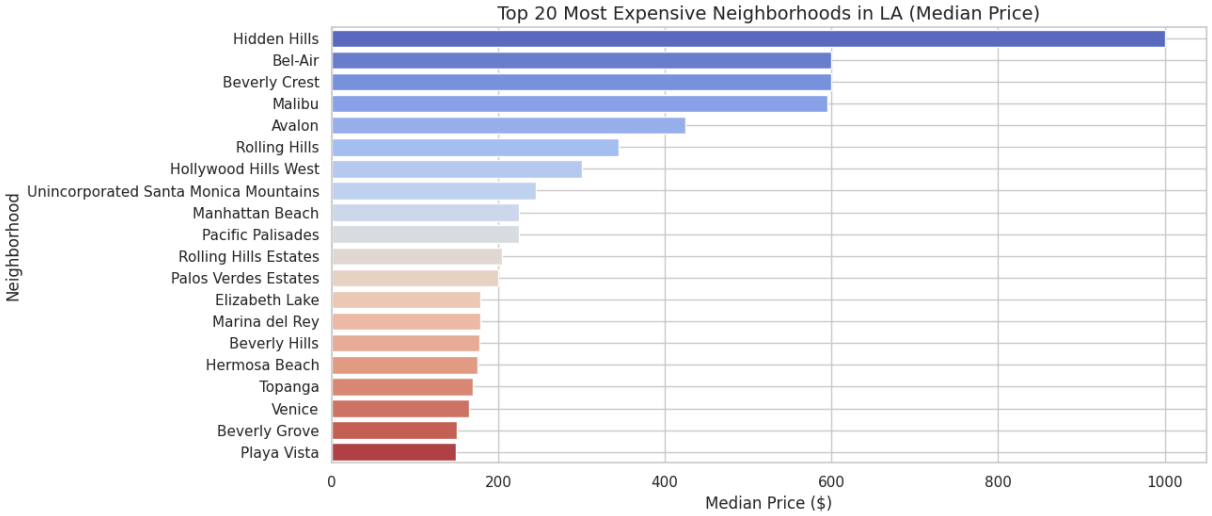
- Prices show a consistent seasonal pattern across years.
- January peaks, March-April dips, summer and holiday seasons rise.
- 2020-2021 fluctuations likely reflect COVID-19 impacts.
- Trends suggest price sensitivity to demand cycles and external factors.

Price Distribution by Neighborhood

In [ ]:
```
plt.figure(figsize=(12, 6))
top_neighborhoods = df_home_sharing.groupby('neighbourhood')['price'].median
sns.barplot(
    x=top_neighborhoods.values,
    y=top_neighborhoods.index,
    palette='coolwarm'
)
plt.title("Top 20 Most Expensive Neighborhoods in LA (Median Price)", fontsi
plt.xlabel("Median Price ($)", fontsize=12)
plt.ylabel("Neighborhood", fontsize=12)
```

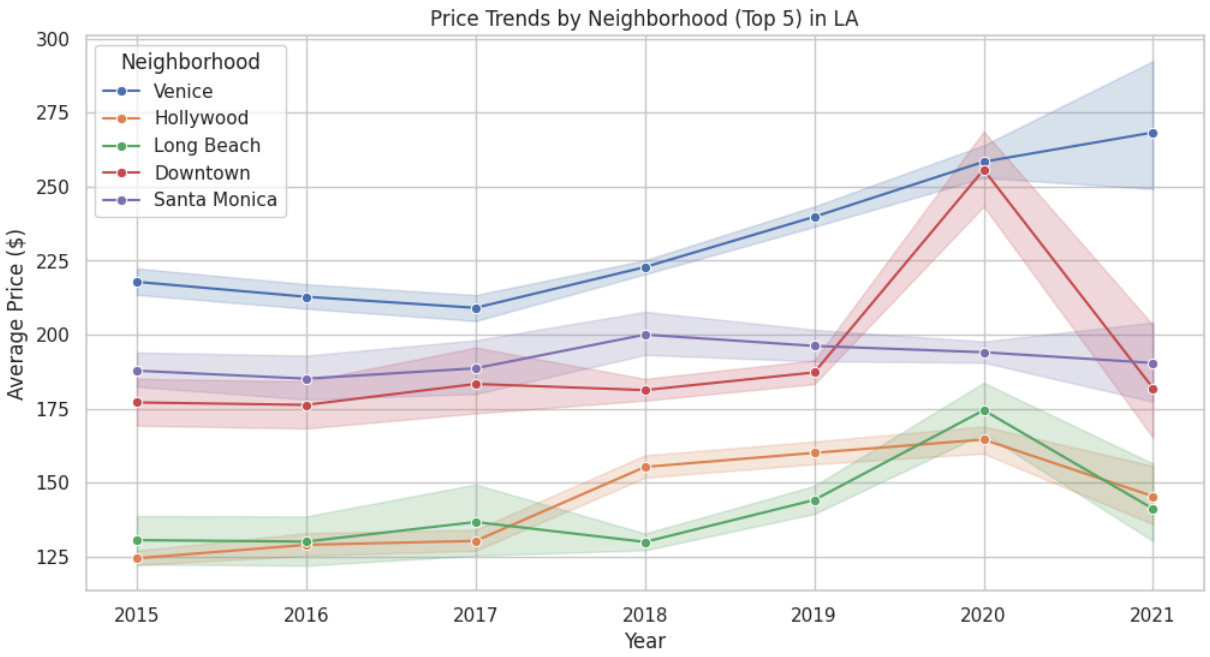
```
plt.grid(True)
plt.show()
```

```
<ipython-input-32-c8cf5065a35b>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the
same effect.

  sns.barplot(
```



- Hidden Hills, Bel-Air, and Beverly Crest have the highest median prices.
- Coastal and luxury areas dominate the top rankings.
- Price variations reflect demand, exclusivity, and location desirability.
- High-end areas may shape future pricing trends in LA’s home-sharing market.

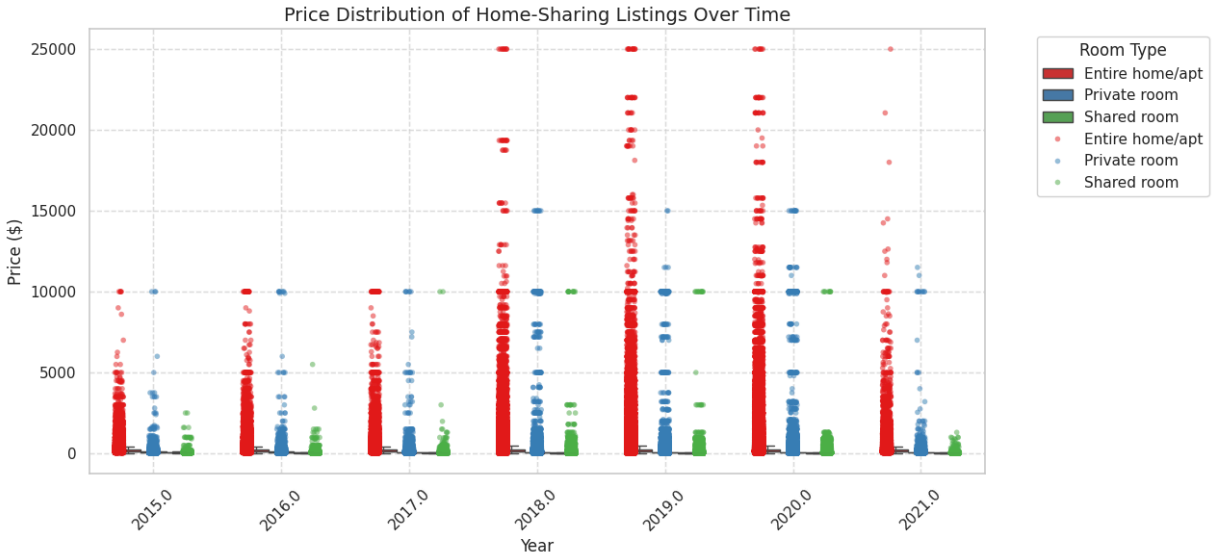
In [ ]:
```
top_neighborhoods = df_home_sharing['neighbourhood'].value_counts().nlargest
df_top_neighborhoods = df_home_sharing[df_home_sharing['neighbourhood'].isir

plt.figure(figsize=(12, 6))
sns.lineplot(data=df_top_neighborhoods, x='year', y='price', hue='neighbourh
plt.title("Price Trends by Neighborhood (Top 5) in LA")
plt.xlabel("Year")
plt.ylabel("Average Price ($)")
plt.legend(title="Neighborhood")
plt.grid(True)
plt.show()
```



- Venice and Santa Monica remained the most expensive neighborhoods.
- Price spikes in 2020 align with overall home-sharing trends during the pandemic.
- Downtown experienced volatility, showing a steep rise followed by a decline.
- Hollywood and Long Beach exhibited steady growth before dipping in 2021.

In [ ]:
```
plt.figure(figsize=(12, 6))

# Boxplot (without inner points)
sns.boxplot(
    data=df_home_sharing,
    x='year',
    y='price',
    hue='room_type',
    palette='Set1',  # Ensure color differences
    dodge=True,
    width=0.6,
    linewidth=1,
    fliersize=0  # Hide default boxplot outliers
)

# Stripplot (overlay colored dots for outliers)
sns.stripplot(
    data=df_home_sharing,
    x='year',
    y='price',
    hue='room_type',
    palette='Set1',  # Ensure it matches the boxplot
    dodge=True,
    jitter=True,  # Spread out points
    alpha=0.5,  # Transparency for better visibility
    size=4  # Adjust dot size
)
```

```
# Titles and labels
plt.title("Price Distribution of Home-Sharing Listings Over Time", fontsize=
plt.xlabel("Year", fontsize=12)
plt.ylabel("Price ($)", fontsize=12)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Grid and legend adjustments
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(title="Room Type", bbox_to_anchor=(1.05, 1), loc='upper left')

# Show plot
plt.show()
```



- Entire homes dominate, with rising high-end listings.
- Outliers ($25K+) suggest luxury rentals.
- Private/shared rooms stable with lower price variance.
- 2021 drop reflects market contraction.

B. Future Price Trends in LA's Home-Sharing Market **

Prediction model (SARIMA & XGBoost)

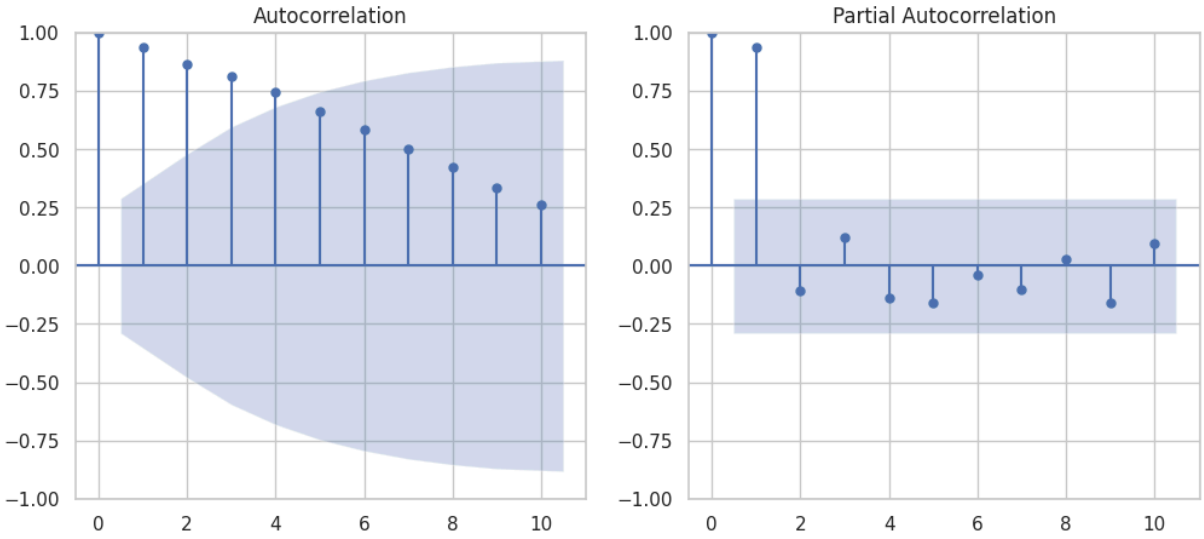
In [ ]:
```
#SARIMA
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
```

```
import warnings

warnings.filterwarnings("ignore")

df = pd.read_csv('Combined Listing Data [Summary].csv')

# Sample DataFrame (Replace with your actual DataFrame)
df['year'] = df['Scrape File'].str.extract(r'(\d{4})').astype(int)  # Extrac
df['month'] = df['Scrape File'].str.extract(r'_(\d{2})_').astype(int)  # Ext

# Create a proper date column
df['date'] = pd.to_datetime(df[['year', 'month']].assign(day=1))  # Set day

df_monthly = df.groupby('date').agg(
    avg_price=('price', 'mean'),         # Average price per month
).reset_index()

# Set 'date' as index
df_monthly = df_monthly.set_index('date')
```

In [ ]:
```
# Check for Stationarity
def adf_test(series):
    result = adfuller(series.dropna())
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {result[1]}')
    print("Stationary" if result[1] < 0.05 else "Non-Stationary")

adf_test(df_monthly['avg_price'])
```

```
ADF Statistic: -1.1696998137849097
p-value: 0.6864924287745809
Non-Stationary
```

In [ ]:
```
# determine para
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

max_lags = min(10, len(df_monthly) // 2)  # Adjust to ensure valid lag count

plt.figure(figsize=(12,5))
plt.subplot(121)
plot_acf(df_monthly, lags=max_lags, ax=plt.gca())  # ACF to determine 'q'
plt.subplot(122)
plot_pacf(df_monthly, lags=max_lags, ax=plt.gca())  # PACF to determine 'p'
plt.show()
```

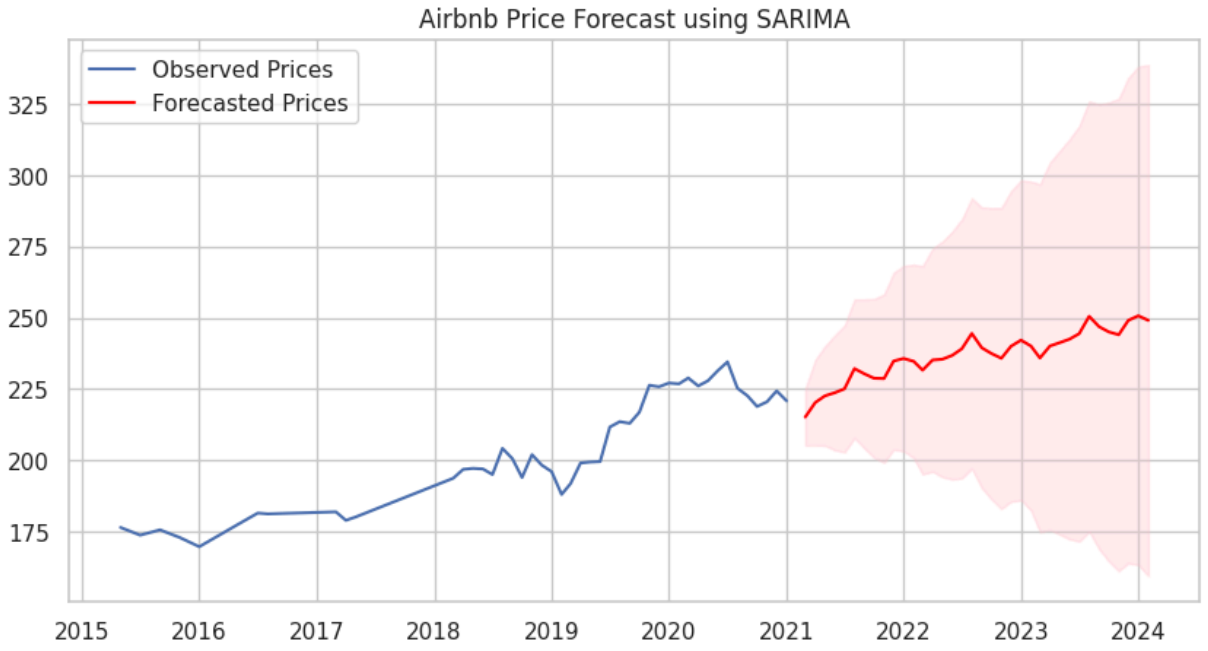


In [ ]:
```
# Define SARIMA model (adjust parameters as needed)
model = SARIMAX(df_monthly,
                order=(1, 1, 2),  # (p, d, q)
                seasonal_order=(1, 1, 1, 12),  # (P, D, Q, s)
                enforce_stationarity=False,
                enforce_invertibility=False)

# Fit the model
sarima_result = model.fit()
print(sarima_result.summary())
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                           avg_price   No. Observations:
46
Model:             SARIMAX(1, 1, 2)x(1, 1, [1], 12)   Log Likelihood
-54.933
Date:                              Fri, 21 Feb 2025   AIC
121.865
Time:                                      08:56:56   BIC
127.208
Sample:                                           0   HQIC
122.602
                                                - 46
Covariance Type:                                opg
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
ar.L1         -0.4698      1.075     -0.437      0.662      -2.577       1.6
38
ma.L1          0.6009      1.313      0.458      0.647      -1.972       3.1
74
ma.L2         -0.1672      0.576     -0.291      0.771      -1.295       0.9
61
ar.S.L12      -0.4287      0.364     -1.176      0.240      -1.143       0.2
86
ma.S.L12      -0.1827      0.784     -0.233      0.816      -1.719       1.3
53
sigma2        25.6132     11.891      2.154      0.031       2.306      48.9
20
==========================================================================================
=======
Ljung-Box (L1) (Q):                   0.07   Jarque-Bera (JB):
3.34
Prob(Q):                              0.80   Prob(JB):
0.19
Heteroskedasticity (H):               3.29   Skew:
-0.99
Prob(H) (two-sided):                  0.17   Kurtosis:
3.72
==========================================================================================
=======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (compl
ex-step).
```

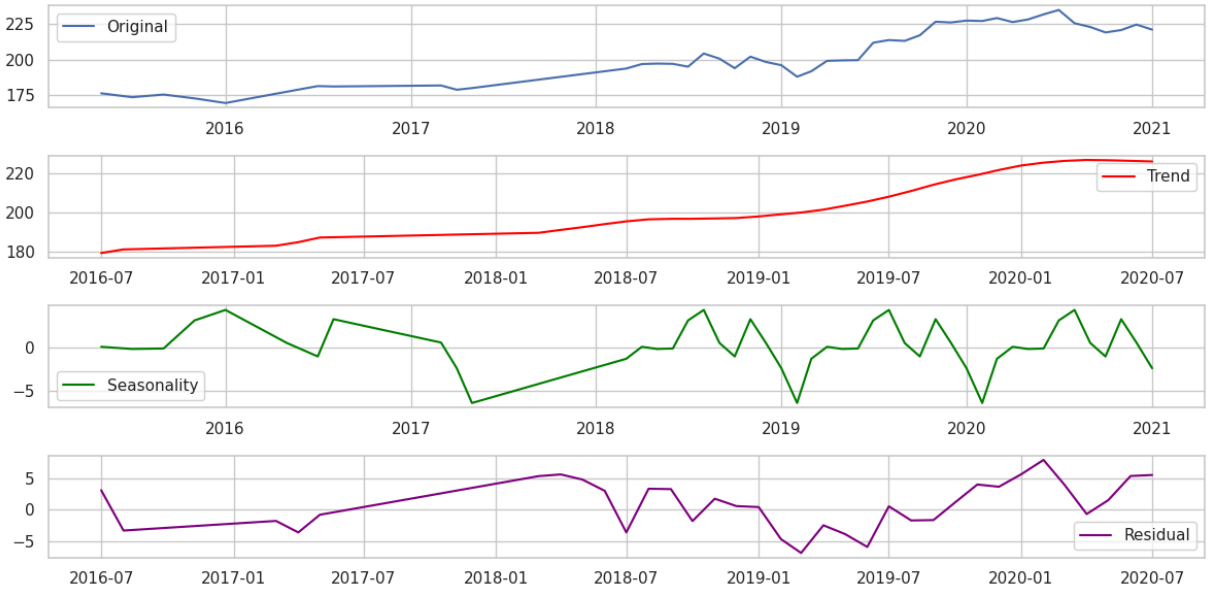
In [ ]:
```
# Forecast next 36 months
forecast_period = 36
forecast = sarima_result.get_forecast(steps=forecast_period)
forecast_index = pd.date_range(start=df.index[-1], periods=forecast_period,

# Correct forecast index
```

```
forecast_index = pd.date_range(start=df_monthly.index[-1] + pd.DateOffset(mc
                               periods=forecast_period,
                               freq='M')
# Confidence intervals
forecast_ci = forecast.conf_int()

# Plot results
plt.figure(figsize=(10,5))
plt.plot(df_monthly.index, df_monthly['avg_price'], label='Observed Prices')
plt.plot(forecast_index, forecast.predicted_mean, color='red', label='Foreca
plt.fill_between(forecast_index, forecast_ci.iloc[:,0], forecast_ci.iloc[:,1
plt.title("Airbnb Price Forecast using SARIMA")
plt.legend()
plt.show()
```



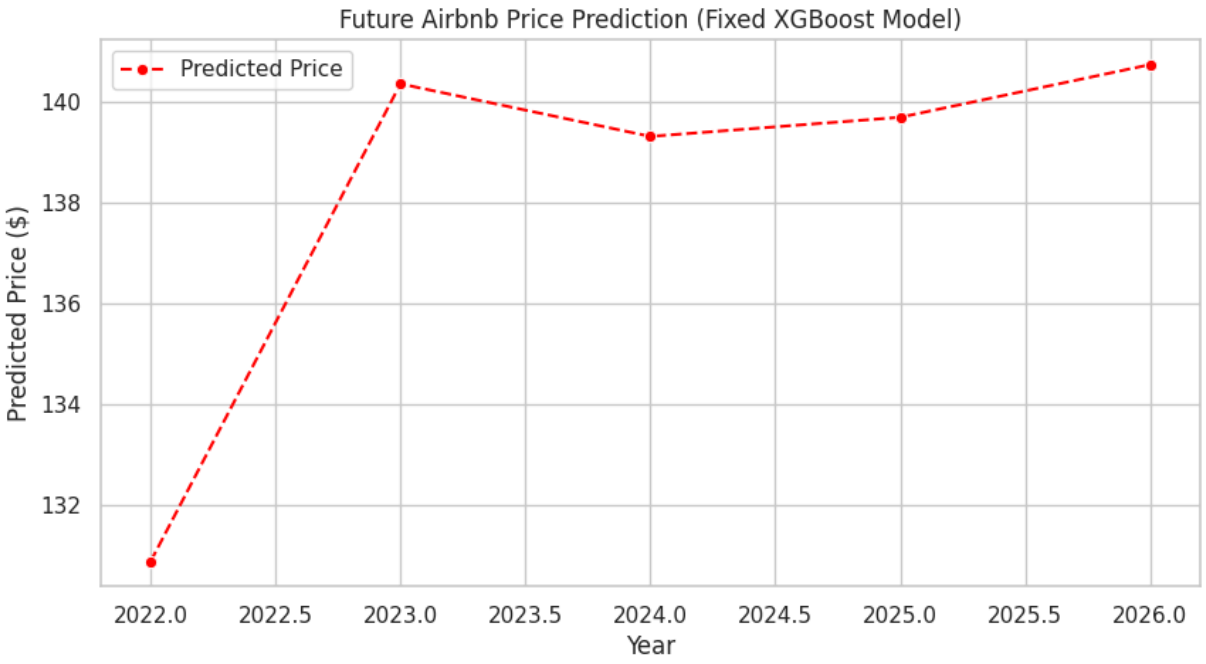
In [ ]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

decomposition = seasonal_decompose(df_monthly, model='additive', period=12)

#original data
plt.figure(figsize=(12,6))
plt.subplot(411)
plt.plot(decomposition.observed, label='Original')
plt.legend()

#trend
plt.subplot(412)
plt.plot(decomposition.trend, label='Trend', color='red')
plt.legend()

#seasonality
plt.subplot(413)
plt.plot(decomposition.seasonal, label='Seasonality', color='green')
plt.legend()

#residual
```

```
plt.subplot(414)
plt.plot(decomposition.resid, label='Residual', color='purple')
plt.legend()

plt.tight_layout()
plt.show()
```



In [ ]:
```
#XGBoost
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from xgboost import XGBRegressor
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error

df = pd.read_csv('Combined Listing Data [Summary].csv')

# Convert 'year' column to numeric
df['year'] = pd.to_datetime(df['Scrape File'].str.extract(r'(\d{4})')[0], fo

# Remove outliers (5th-95th percentile)
lower_bound = np.percentile(df['price'], 5)
upper_bound = np.percentile(df['price'], 95)
df_filtered = df[(df['price'] >= lower_bound) & (df['price'] <= upper_bound)

# Select Features (Adding 'year' to capture trends)
features = ['year', 'availability_365', 'number_of_reviews', 'calculated_hos
target = 'price'

# Remove missing values
df_filtered = df_filtered.dropna(subset=[target] + features)

# Define X (input) and y (target)
X = df_filtered[features]
y = df_filtered[target]
```

```
# Use Time-Based Train-Test Split
tscv = TimeSeriesSplit(n_splits=5)
for train_index, test_index in tscv.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

# Train XGBoost Model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=5, r
xgb_model.fit(X_train, y_train)

# Predict Future Prices
future_years = np.arange(df['year'].max() + 1, df['year'].max() + 6)
future_data = pd.DataFrame({
    'year': future_years,
    'availability_365': np.linspace(df['availability_365'].mean(), df['avail
    'number_of_reviews': np.linspace(df['number_of_reviews'].mean(), df['num
    'calculated_host_listings_count': np.linspace(df['calculated_host_listir
})

future_prices = xgb_model.predict(future_data)

# Plot Fixed XGBoost Price Predictions
plt.figure(figsize=(10, 5))
sns.lineplot(x=future_years, y=future_prices, linestyle='dashed', marker='o'
plt.xlabel("Year")
plt.ylabel("Predicted Price ($)")
plt.title("Future Airbnb Price Prediction (Fixed XGBoost Model)")
plt.grid(True)
plt.legend()
plt.show()
```



In [ ]:
```
# Price change
for i in range(1,len(future_prices)):
  print(f'Price change {(future_prices[i]-future_prices[i-1])/future_prices[
```

```
Price change 7.23700150847435%
Price change -0.7964931428432465%
Price change 0.28991049621254206%
Price change 0.79781049862504%
```