Exp no: 2 **Implement programs for visualizing time series data**

Date: 31/1/25

Objectives:

The goal of this preprocessing pipeline is to prepare time series sales data for accurate predictive modeling. This involves handling missing values, removing anomalies, ensuring correct time order, and potentially transforming features. The aim is to create a clean and consistent dataset that improves the performance of time series forecasting models, enabling better predictions of future sales trends and the identification of seasonality patterns.

Background/Scope:

Time series sales data is a sequence of observations indexed by time. Preprocessing is crucial because missing values, outliers, incorrect time order, and other data quality issues can negatively impact forecasting models. This process addresses these challenges by filling gaps, removing outliers, ordering data chronologically, and potentially creating new features. Proper preprocessing is essential for time series models to accurately capture trends, predict future sales, and support informed business decisions.

Steps for Time Series Sales Data Preprocessing:

Step 1: Load the Dataset and libraries

1. **Load** the dataset from a CSV file into a Pandas DataFrame.

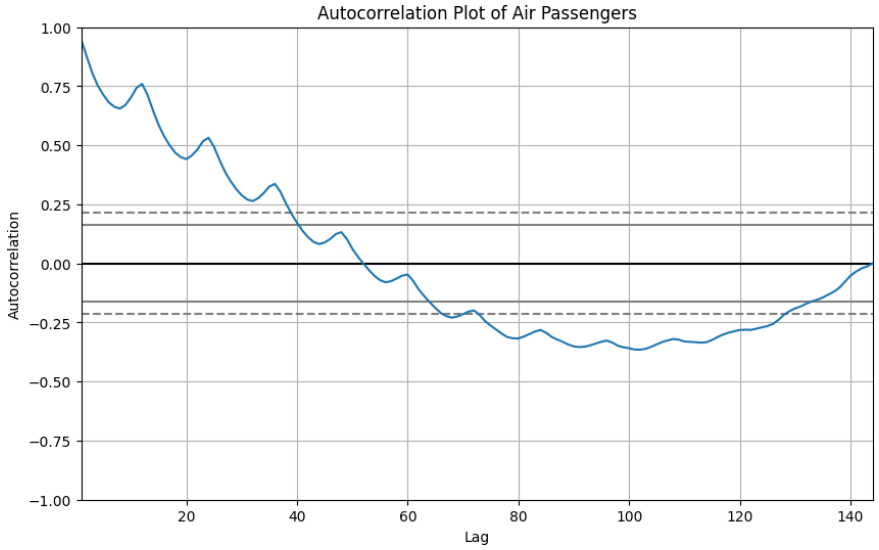
```
# Step 1: Load the Dataset

import pandas as pd import numpy
as np import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf from statsmodels.tsa.stattools
import adfuller df=pd.read_csv("/content/NFLX.csv")
```

Step 2: Visualizing techniques

Autocorrelation Plot: Create an autocorrelation plot to identify potential seasonality using the acf_plot

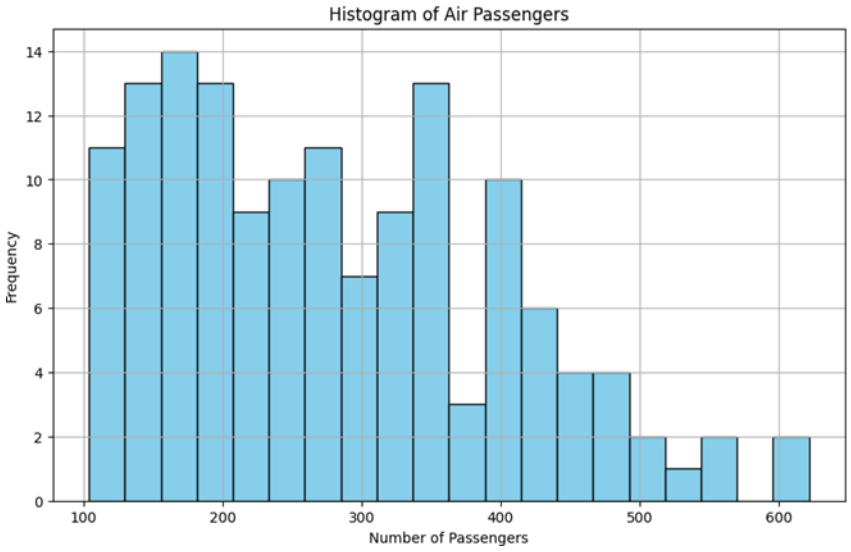
```
if 'Date' not in df.columns:
    print("'Date' is already the index or not present in the DataFrame.") else:
    df.set_index('Date', inplace=True)
```

Plot the ACF plt.figure(figsize=(12,6))
plot_acf(df['Volume'], lags=40) # You can adjust the number of lags as needed
plt.xlabel('Lag') plt.ylabel('Autocorrelation')
plt.title('Autocorrelation Function (ACF) Plot') plt.show()



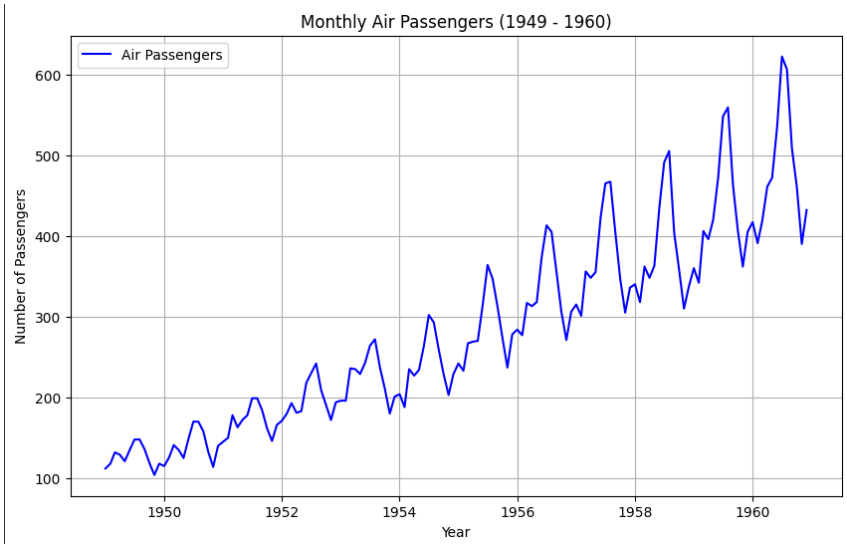
Histogram Plot: Create a histogram plot to visualize autocorrelation.

plt.figure(figsize=(10, 6))
plt.hist(df['#Passengers'], bins=20, color='skyblue', edgecolor='black') plt.title('Histogram of Air Passengers')
plt.xlabel('Number of Passengers')
plt.ylabel('Frequency')
plt.grid(True) plt.show()



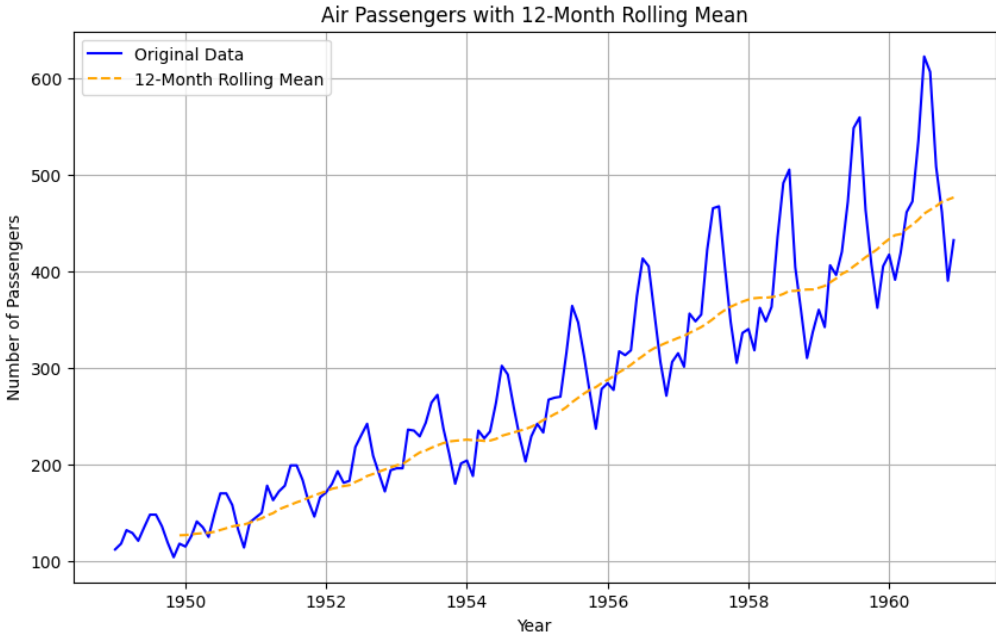
Line Plot (Basic Time Series Plot)

```
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['#Passengers'], label='Air Passengers', color='blue')
plt.title('Monthly Air Passengers (1949 - 1960)')
plt.xlabel('Year')
plt.ylabel('Number of Passengers')
plt.grid(True)
plt.legend()
plt.show()
```



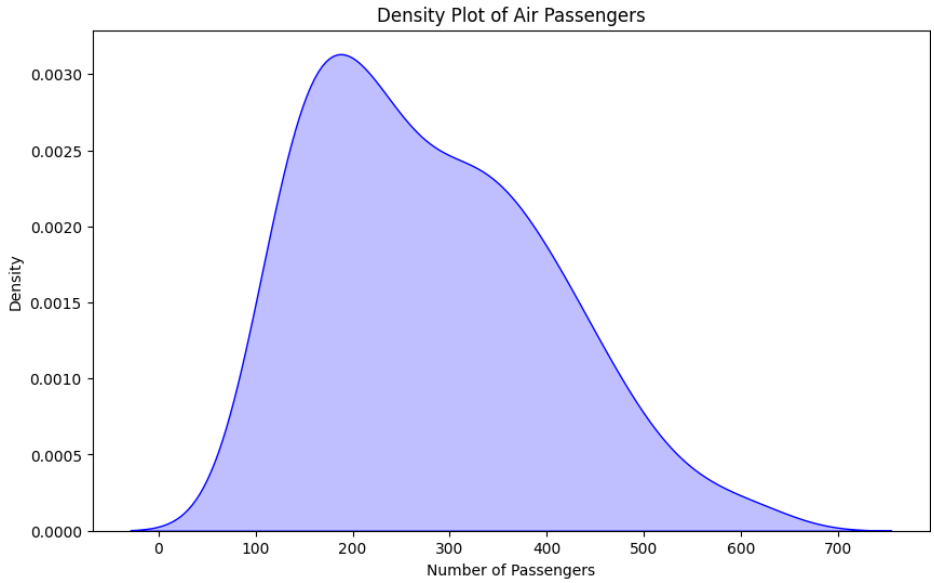
Rolling Mean (Moving Average) Plot

```
df['Rolling_Mean'] = df['#Passengers'].rolling(window=12).mean()

plt.figure(figsize=(10, 6))
plt.plot(df.index, df['#Passengers'], label='Original Data', color='blue')
plt.plot(df.index, df['Rolling_Mean'], label='12-Month Rolling Mean', color='orange', linestyle='--')
plt.title('Air Passengers with 12-Month Rolling Mean')
plt.xlabel('Year')
plt.ylabel('Number of Passengers')
plt.legend()
plt.grid(True)
plt.show()
```



Density Plot (Kernel Density Estimate)

```
plt.figure(figsize=(10, 6))
sns.kdeplot(df['#Passengers'], fill=True, color='blue')
plt.title('Density Plot of Air Passengers')
plt.xlabel('Number of Passengers')
plt.ylabel('Density')
plt.show()
```



Result:

Thus the time series dataset is visualized successfully