**IMPLEMENT PROGRAM FOR TIME SERIES DATA CLEANING, LOADING, HANDLING AND PREPROCESSING TECHNIQUES**

**Aim:**

To implement a program for time series data cleaning, loading, handling and preprocessing techniques.

**Procedure & Code:**

**Step 1: Importing the required libraries**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

**Step 2: Loading and visualizing the dataset (air passenger datset)**

url = "C:/Users/HDC0422272/Downloads/airline-passengers.csv"

data = pd.read\_csv(url, parse\_dates=['Month'], index\_col='Month')

print("Dataset Preview:")

print(data.head())

plt.figure(figsize=(12, 6))

plt.plot(data.index, data['Passengers'], marker='o', linestyle='-')

plt.title('Monthly Air Passengers (1949-1960)', fontsize=16)

plt.xlabel('Date', fontsize=12)

plt.ylabel('Number of Passengers (in thousands)', fontsize=12)

plt.grid()

plt.show()

Dataset Preview:

Passengers

Month

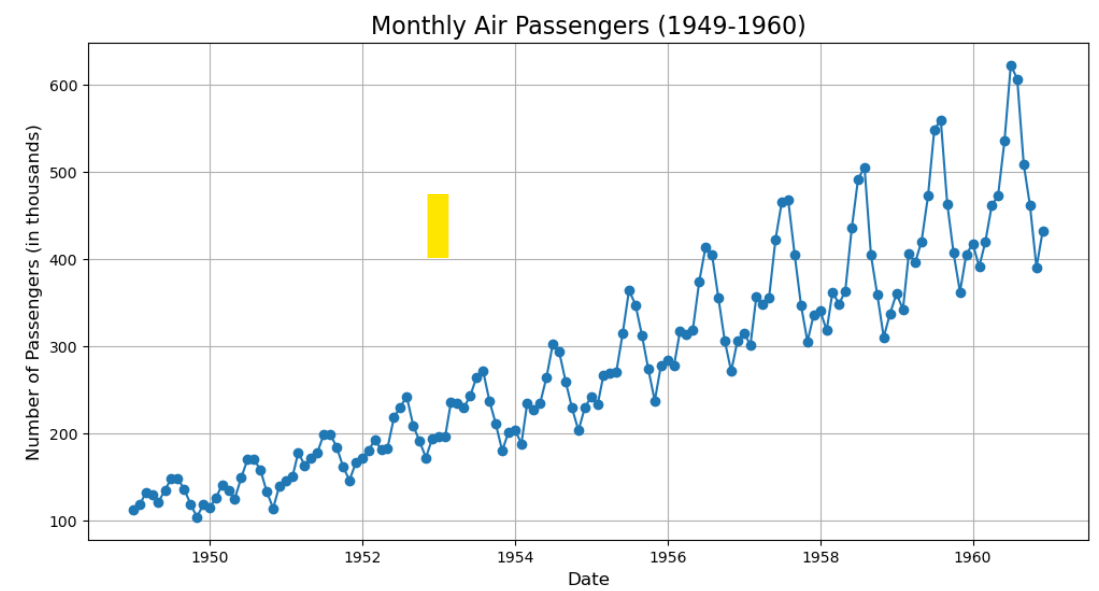
1949-01-01 112

1949-02-01 118

1949-03-01 132

1949-04-01 129

1949-05-01 121



**Step 3: Data pre-processing**

from sklearn.preprocessing import MinMaxScaler

print("Checking for missing values:")

print(data.isnull().sum())

data['Passengers'] = data['Passengers'].fillna(method='ffill')

scaler = MinMaxScaler(feature\_range=(0, 1))

data['Passengers\_Normalized'] = scaler.fit\_transform(data[['Passengers']])

print("\nPreprocessed Data (First 5 Rows):")

print(data.head())

plt.figure(figsize=(12, 6))

plt.plot(data.index, data['Passengers\_Normalized'], color='green', linestyle='-', marker='o')

plt.title('Normalized Air Passengers Data', fontsize=16)

plt.xlabel('Date', fontsize=12)

plt.ylabel('Normalized Passengers', fontsize=12)

plt.grid()

plt.show()

Checking for missing values:

Passengers 0

dtype: int64

Preprocessed Data (First 5 Rows):

Passengers Passengers\_Normalized

Month

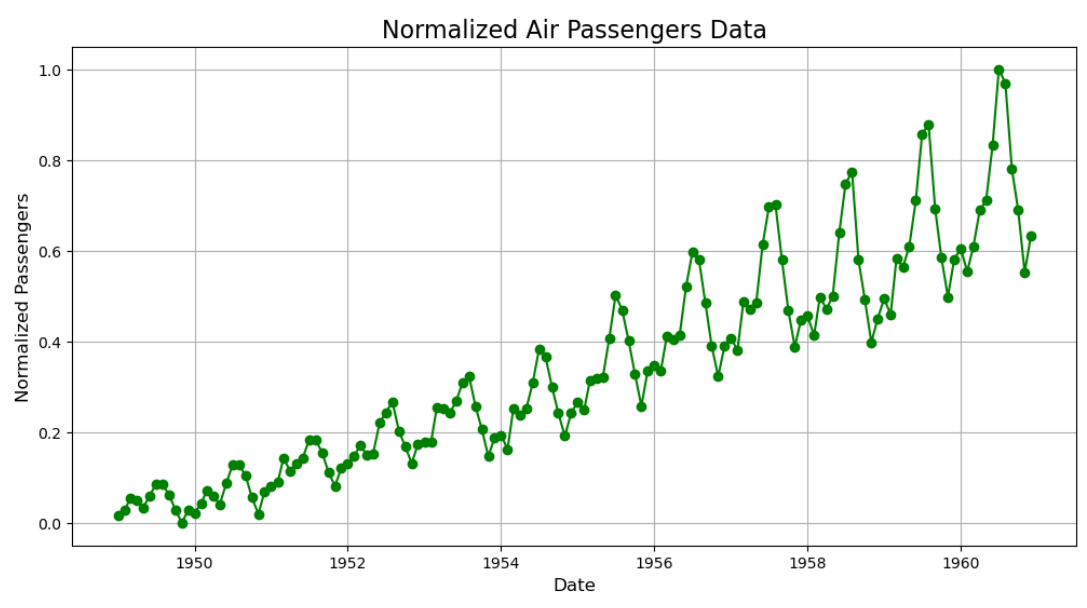
1949-01-01 112 0.015444

1949-02-01 118 0.027027

1949-03-01 132 0.054054

1949-04-01 129 0.048263

1949-05-01 121 0.032819

****

**Step 4: Splitting the data and Arima model implementation**

from statsmodels.tsa.arima.model import ARIMA

from sklearn.metrics import mean\_squared\_error

import numpy as np

train\_size = int(len(data) \* 0.8)

train, test = data['Passengers'][:train\_size], data['Passengers'][train\_size:]

model = ARIMA(train, order=(2, 1, 2))

arima\_model = model.fit()

forecast = arima\_model.forecast(steps=len(test))

forecast\_index = test.index

mse = mean\_squared\_error(test, forecast)

print(f"Mean Squared Error (MSE): {mse}")

print(f"Root Mean Squared Error (RMSE): {np.sqrt(mse)}")

Mean Squared Error (MSE): 6808.397036193119

Root Mean Squared Error (RMSE): 82.51301131453826

**Step 5: Model evaluation**

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import numpy as np

mae = mean\_absolute\_error(test, forecast)

mse = mean\_squared\_error(test, forecast)

rmse = np.sqrt(mse)

r2 = r2\_score(test, forecast)

plt.figure(figsize=(12, 6))

plt.plot(test.index, test, label='Actual', color='green')

plt.plot(forecast\_index, forecast, label='Forecast', color='red', linestyle='--')

plt.title('Actual vs Predicted - Model Evaluation', fontsize=16)

plt.xlabel('Date', fontsize=12)

plt.ylabel('Number of Passengers', fontsize=12)

plt.legend()

plt.grid()

plt.show()

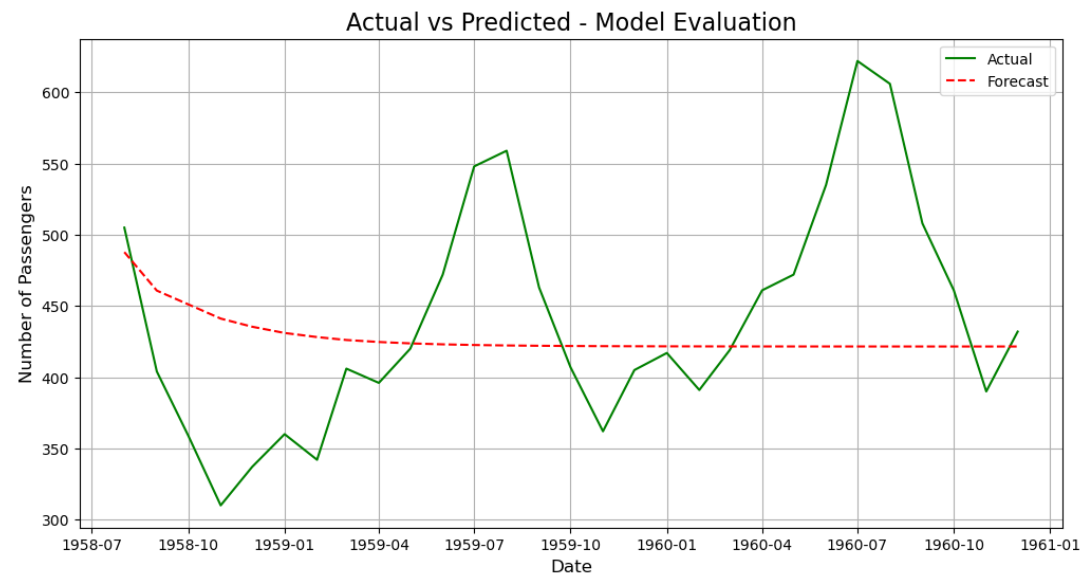
print("Model Performance Metrics:")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Squared Error (MSE): {mse}")

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"R-squared (R2): {r2}")



Model Performance Metrics:

Mean Absolute Error (MAE): 63.54531128529889

Mean Squared Error (MSE): 6808.397036193119

Root Mean Squared Error (RMSE): 82.51301131453826

R-squared (R2): -0.11530974013365625

**Result:**

Thus the implementation of a program for time series data cleaning, loading, handling and preprocessing techniques has been successfully written and executed successfully.