EXPERIMENT 2: Implement programs for visualizing time series data.

Line Chart (Basic Time Series Visualization)

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

plt.plot(data['Passengers'], label='Monthly Air Passengers', color='blue')

plt.title('Air Passengers Over Time', fontsize=16)

plt.xlabel('Year', fontsize=14)

plt.ylabel('Number of Passengers', fontsize=14)

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.show()
```

Explanation:

1. **`plt.figure(figsize=(12, 6))`**: Sets the size of the figure.
2. **`plt.plot(data['Passengers'], ...)`**: Plots the number of passengers against the time (index of the dataframe).
3. **`plt.title()`**, **`plt.xlabel()`**, **`plt.ylabel()`**: Adds a title and labels to the chart.
4. **`plt.legend()`**: Adds a legend to describe the line.
5. **`plt.grid()`**: Adds gridlines for better readability.
6. **`plt.show()`**: Displays the plot.

Uses:

- Visualizes trends, patterns, and fluctuations in time series data.
- Helps in identifying long-term trends or seasonality.

Seasonal Decomposition Plot

```
result = seasonal_decompose(data['Passengers'],
model='multiplicative', period=12)

result.plot()

plt.show()
```

Explanation:

1. **`seasonal_decompose()`:** Decomposes the time series into three components:
   - **Trend:** Long-term progression in the data.
   - **Seasonality:** Repeated patterns at fixed intervals.
   - **Residuals:** Noise or random fluctuations.
2. **`model='multiplicative'`:** Assumes the components multiply together (used for time series with growth).
3. **`result.plot()`:** Plots the decomposition components.

Uses:

- Identifies how seasonality and trend contribute to the observed time series.
- Helps in modeling or forecasting.

Autocorrelation Plot (ACF)

```
plot_acf(data['Passengers'], lags=40)

plt.title('Autocorrelation Plot (ACF)', fontsize=16)

plt.show()
```

Explanation:

1. **`plot_acf()`:** Computes the correlation of the time series with lagged versions of itself.
2. **`lags=40`:** Displays up to 40 lags in the plot.

Uses:

- Identifies repeated patterns or seasonality in the data.
- Helps determine the lag values for time series modeling.

Partial Autocorrelation Plot (PACF)

```
plot_pacf(data['Passengers'], lags=40, method='ywm')

plt.title('Partial Autocorrelation Plot (PACF)', fontsize=16)

plt.show()
```

Explanation:

1. **`plot_pacf()`:** Displays correlation between a time series and its lags, excluding the influence of intermediate lags.
2. **`method='ywm'`:** Specifies the calculation method for PACF.
3. **`lags=40`:** Analyzes the first 40 lags.

Uses:

- Determines the direct relationship between a time series and its past values.
- Useful for deciding the order of AR terms in ARIMA models.

Histogram

```
plt.hist(data['Passengers'], bins=20, color='skyblue',
edgecolor='black')

plt.title('Histogram of Air Passengers', fontsize=16)

plt.xlabel('Number of Passengers', fontsize=14)

plt.ylabel('Frequency', fontsize=14)

plt.grid(True, linestyle='--', alpha=0.6)

plt.show()
```

Explanation:

1. **`plt.hist()`:** Creates a histogram to show the distribution of passenger counts.
2. **`bins=20`:** Divides the range of data into 20 intervals.
3. **`color`, `edgecolor`:** Adjusts the visual appearance.

Uses:

- Understands the distribution of data.
- Detects skewness or outliers.

Box Plot

```
sns.boxplot(x=data.index.month, y='Passengers',
data=data.reset_index(), palette='coolwarm')

plt.title('Monthly Trends of Air Passengers (Boxplot)', fontsize=16)

plt.xlabel('Month', fontsize=14)

plt.ylabel('Number of Passengers', fontsize=14)

plt.grid(True, linestyle='--', alpha=0.6)

plt.show()
```

Explanation:

1. **`sns.boxplot()`:** Creates a box plot for passenger counts across months.
2. **`x=data.index.month`:** Groups data by month.
3. **`palette='coolwarm'`:** Sets the color scheme.

Uses:

- Compares distributions across months.
- Detects outliers, variability, and seasonal patterns.

Heatmap

```
pivot_data = data.pivot_table(values='Passengers',
index=data.index.year, columns=data.index.month)

sns.heatmap(pivot_data, annot=True, fmt='.0f', cmap='coolwarm',
cbar=True)

plt.title('Heatmap of Monthly Passenger Counts', fontsize=16)

plt.xlabel('Month', fontsize=14)
```

```
plt.ylabel('Year', fontsize=14)

plt.show()
```

Explanation:

1. **`pivot_table()`:** Reshapes the data into a matrix format (years as rows, months as columns).
2. **`sns.heatmap()`:** Creates a heatmap where the intensity of colors represents the passenger count.

Uses:

- Visualizes patterns over months and years.
- Identifies high and low passenger counts.

Rolling Mean and Standard Deviation Plot

```
rolling_mean = data['Passengers'].rolling(window=12).mean()

rolling_std = data['Passengers'].rolling(window=12).std()

plt.plot(data['Passengers'], label='Original Data', color='blue')

plt.plot(rolling_mean, label='Rolling Mean (12 months)', color='red')

plt.plot(rolling_std, label='Rolling Std Dev (12 months)',
color='green')

plt.title('Rolling Mean and Standard Deviation', fontsize=16)

plt.xlabel('Year', fontsize=14)

plt.ylabel('Number of Passengers', fontsize=14)

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.show()
```

Explanation:

1. `rolling(window=12).mean()`: Computes the 12-month rolling average.
2. `rolling(window=12).std()`: Computes the 12-month rolling standard deviation.
3. `plt.plot()`: Plots the original data, rolling mean, and rolling standard deviation.

Uses:

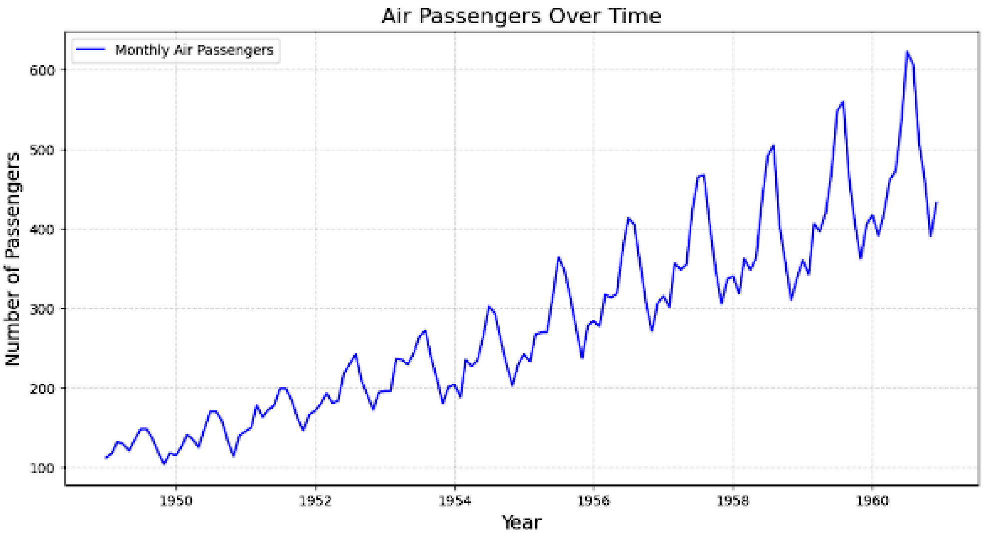
- Tracks changes in trends and variability over time.
- Identifies periods of high volatility or stability.

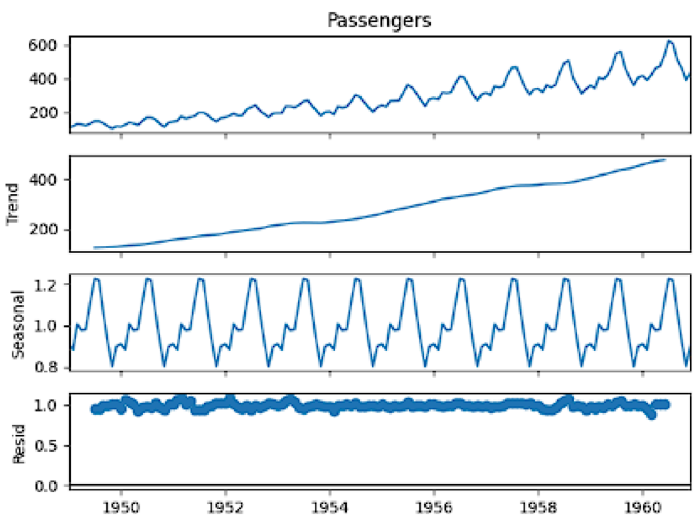
Each visualization serves a specific purpose in time series analysis:

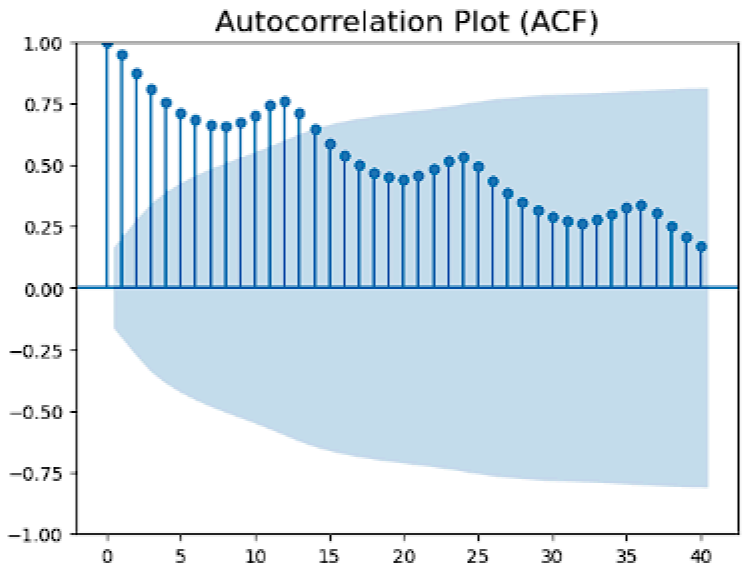
- **Line charts** show overall trends.
- **Seasonal decomposition** reveals underlying components.
- **ACF and PACF** identify lag relationships for model building.
- **Histograms and box plots** analyze data distribution and variability.
- **Heatmaps** uncover patterns over time.
- **Rolling statistics** highlight trends and volatility.

Output Screenshots:

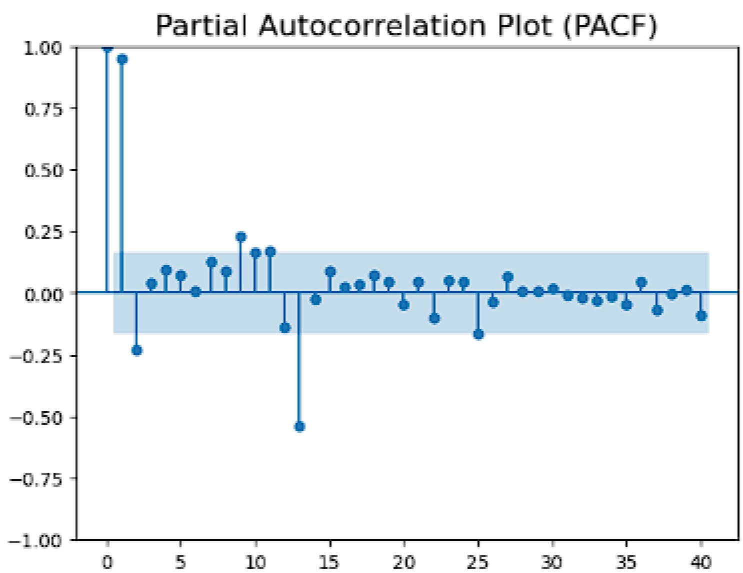
```
4   1949-05          121
```

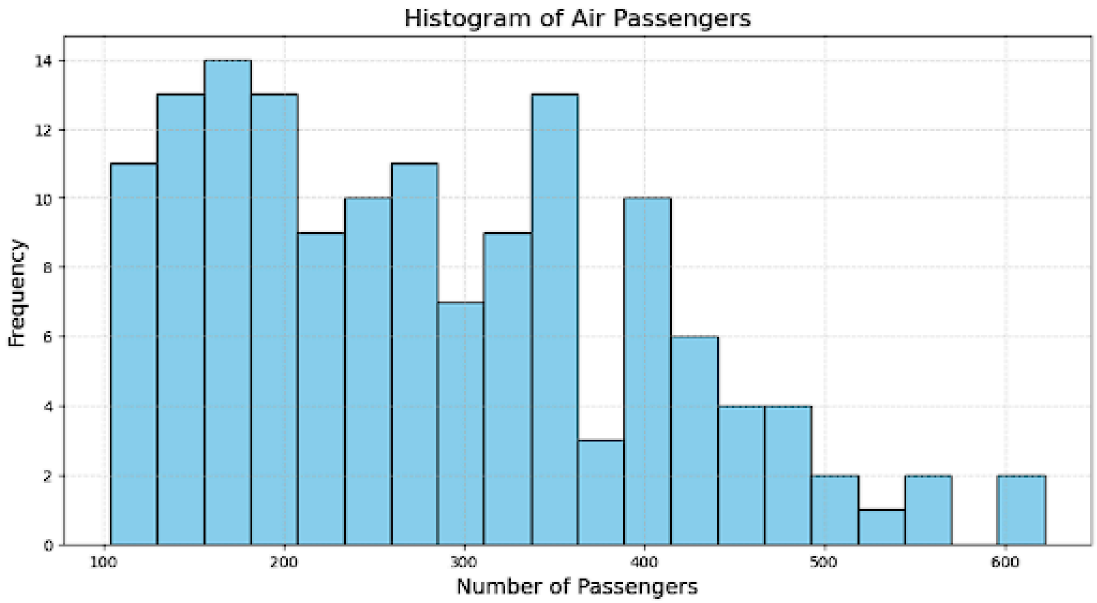






<Figure size 1200x600 with 0 Axes>

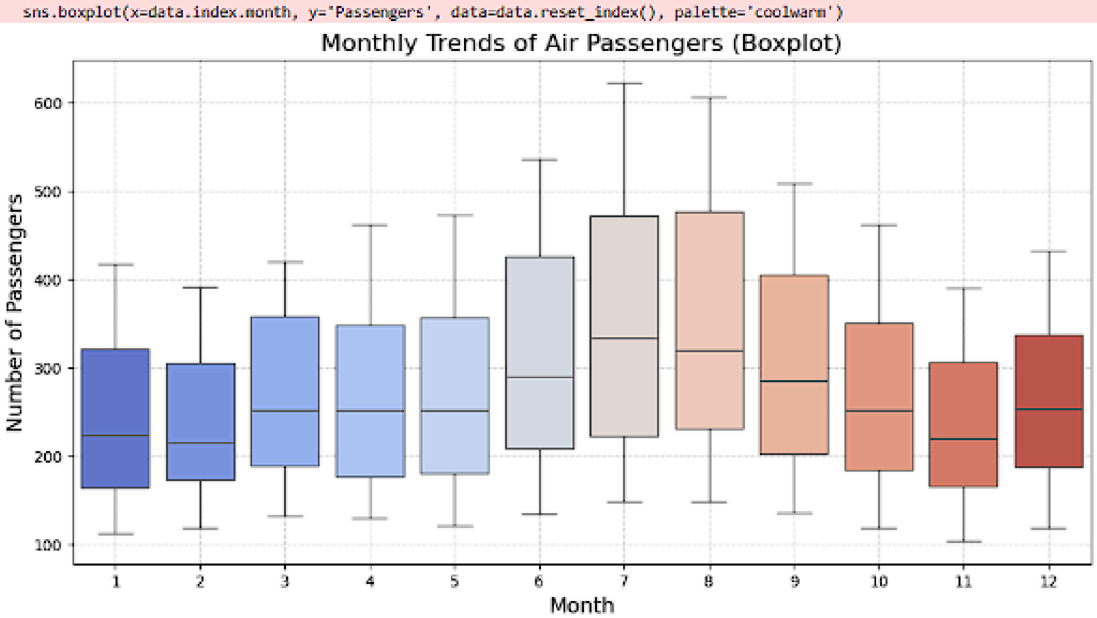


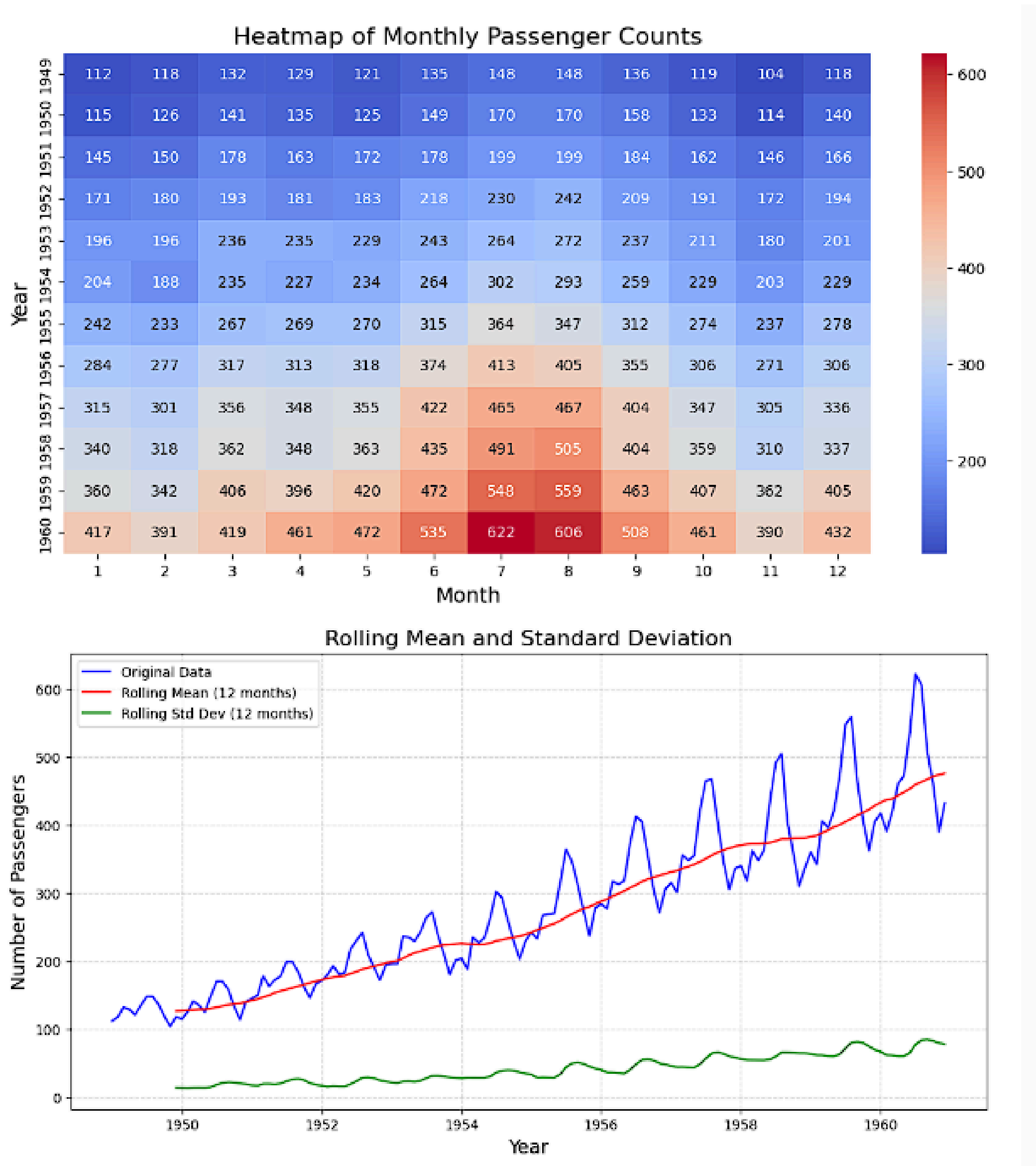



```
<ipython-input-7-822b10f5e192>:64: FutureWarning:

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

  sns.boxplot(x=data.index.month, y='Passengers', data=data.reset_index(), palette='coolwarm')
```





Result: Thus the program to `implement programs for visualizing time series data has been implemented successfully.`