EXPERIMENT 1: Implement programs for time series data cleaning, loading and handling times series data and preprocessing techniques.

Importing necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

1. **`import pandas as pd`**: Imports the `pandas` library, which is used for data manipulation and analysis.
2. **`import numpy as np`**: Imports the `numpy` library, used for numerical operations.
3. **`import matplotlib.pyplot as plt`**: Imports the `matplotlib.pyplot` module for creating visualizations.
4. **`%matplotlib inline`**: Enables inline plotting in Jupyter Notebook, so plots are displayed directly in the notebook.
5. **`import seaborn as sns`**: Imports `seaborn`, a library for advanced data visualization built on `matplotlib`.
6. **`import warnings; warnings.filterwarnings('ignore')`**: Suppresses warning messages in the output to improve readability.

Reading the data

```
data = pd.read_csv('108,110.csv', header=None)
data.columns = ['Month', 'Passengers']
data['Month'] = pd.to_datetime(data['Month'], format='%Y-%m')
data = data.set_index('Month')
data.head()
```

1. **`pd.read_csv('108,110.csv', header=None)`**: Reads a CSV file named `108,110.csv` with no headers (`header=None`).
2. **`data.columns = ['Month', 'Passengers']`**: Assigns column names to the dataset: 'Month' (date information) and 'Passengers' (number of passengers).
3. **`data['Month'] = pd.to_datetime(data['Month'], format='%Y-%m')`**: Converts the 'Month' column to a `datetime` object with the format `Year-Month`.
4. **`data = data.set_index('Month')`**: Sets the 'Month' column as the index for time-series analysis.
5. **`data.head()`**: Displays the first five rows of the dataset for verification.

Plotting the time-series data

```
data.plot(figsize=(12,4))
```

1. **`data.plot(figsize=(12,4))`**: Plots the 'Passengers' column against the index ('Month') as a time-series graph with a figure size of 12x4.

Handling missing values

```
data = data.assign(Passengers_Mean_imputation =
data['Passengers'].fillna(data['Passengers'].mean()))
data['Passengers_Mean_imputation'].plot(figsize=(12,4))
```

1. **`data.assign(Passengers_Mean_imputation = ...)`**: Creates a new column, `Passengers_Mean_imputation`, where missing values in the 'Passengers' column are replaced with the column's mean value using `fillna()`.
2. **`data['Passengers_Mean_imputation'].plot(figsize=(12,4))`**: Plots the 'Passengers_Mean_imputation' column as a time-series graph.

Handling missing values using linear interpolation

```
data = data.assign(Passengers_Linear_Interpolation =
data['Passengers'].interpolate(method='linear'))
data.head()

data['Passengers_Linear_Interpolation'].plot(figsize=(12,4))
```

1. **`data.assign(Passengers_Linear_Interpolation = ...)`**: Adds a new column, `Passengers_Linear_Interpolation`, where missing values in 'Passengers' are filled using linear interpolation. This method estimates missing values based on the values before and after the gap.
2. **`data.head()`**: Displays the first five rows of the dataset to verify the addition of the new column.
3. **`data['Passengers_Linear_Interpolation'].plot(figsize=(12,4))`**: Plots the 'Passengers_Linear_Interpolation' column as a time-series graph.

Outlier detection

```
fig = plt.subplots(figsize=(12,2))
ax = sns.boxplot(data['Passengers'], whis=1.5)
```

1. **`plt.subplots(figsize=(12,2))`**: Creates a figure for plotting with a size of 12x2.
2. **`sns.boxplot(data['Passengers'], whis=1.5)`**: Creates a boxplot using `seaborn` to detect outliers in the 'Passengers' column. The parameter `whis=1.5` determines the whisker length (1.5 times the interquartile range). Points outside the whiskers are considered outliers.

Histogram plot

```
data['Passengers'].hist(figsize=(12,4))
```

1. **`data['Passengers'].hist(figsize=(12,4))`**: Creates a histogram of the 'Passengers' column with a figure size of 12x4. This visualizes the distribution of passenger counts.