**ITD105 Laboratory Exercises #3**

**Exploratory Data Analysis**

**Name:** Allan Raymart C. Paraiso

**Datasets:**

**Regression:** GlobalTemperatures.csv

**Link**: https://www.kaggle.com/datasets/sachinsarkar/climate-change-global-temperature-data

**Video Link:**
https://drive.google.com/file/d/1xm2sIIZtD1YLkOyNIjCaBzoMTcIPhkrh/view?usp=sharing

**Classification**: Cirrhosis Patient Survival Prediction

**Link**: https://www.kaggle.com/datasets/joebeachcapital/cirrhosis-patient-survival-prediction

**Video Link:**
https://drive.google.com/file/d/1i51TkjZepZsR32DqUvgNTiIeQOju5GVF/view?usp=sharing

**Results:**

**Regression Dataset EDA**

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import swifter
import numpy as np  # Import NumPy module

# 1. Load the Data - import the necessary libraries (e.g., Pandas)
# to read your data from a file (e.g., CSV) or a database.

df = pd.read_csv('./GlobalLandTemperaturesByCountry.csv')
```

```python
# 2. Basic Data Exploration - Check the first few rows of the dataset
# to get an initial sense of the data's structure.

df.head()
```

| | dt | AverageTemperature | AverageTemperatureUncertainty | Country |
|---|---|---|---|---|
| 0 | 1838-06-01 | 23.950 | 2.510 | Afghanistan |
| 1 | 1838-07-01 | 26.877 | 2.883 | Afghanistan |
| 2 | 1838-08-01 | 24.938 | 2.992 | Afghanistan |
| 3 | 1838-09-01 | 18.981 | 2.538 | Afghanistan |
| 4 | 1838-10-01 | 13.428 | 3.039 | Afghanistan |

```python
# 3. Data Summary - Generate descriptive statistics for the data, including mean, median,
# standard deviation, and quartiles, to understand the central tendency and spread of the data.

df.describe()
```

| | AverageTemperature | AverageTemperatureUncertainty |
|---|---|---|
| count | 541658.000000 | 542396.000000 |
| mean | 17.262151 | 1.014977 |
| std | 10.935506 | 1.194665 |
| min | -37.711581 | 0.052000 |
| 25% | 10.156000 | 0.323000 |
| 50% | 20.997000 | 0.570000 |
| 75% | 25.829000 | 1.202000 |
| max | 38.842000 | 15.003000 |

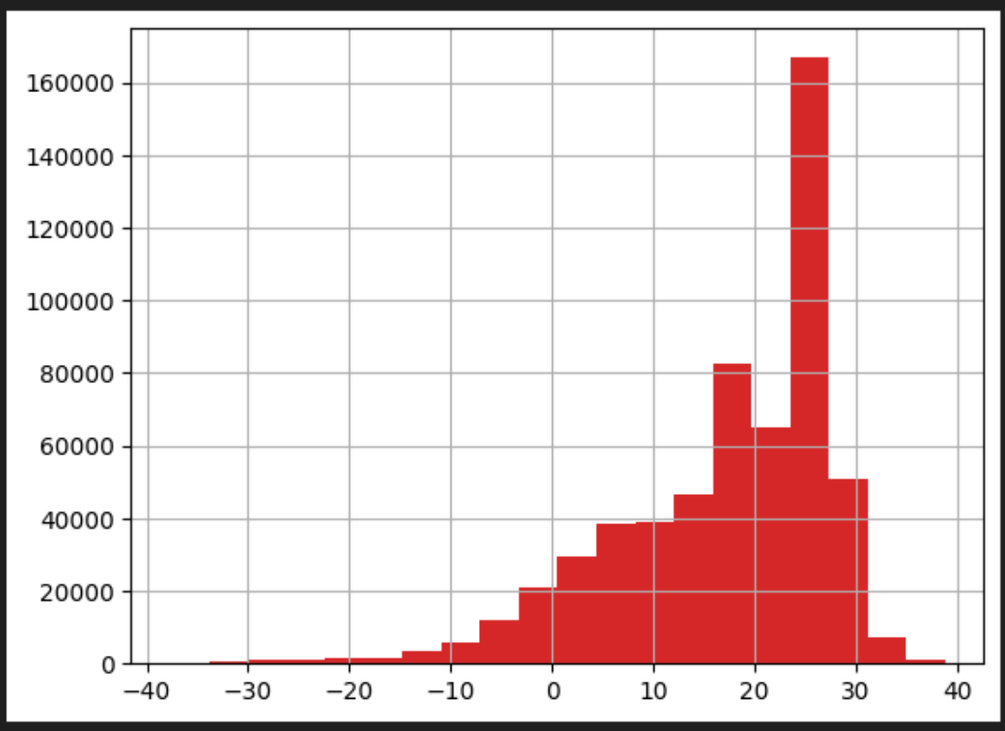
```python
#4. Data Information - check the data types of each column,
# the number of non-null values, and memory usage.

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 574221 entries, 0 to 574220
Data columns (total 4 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   dt                             574221 non-null  object
 1   AverageTemperature             541658 non-null  float64
 2   AverageTemperatureUncertainty  542396 non-null  float64
 3   Country                        574221 non-null  object
dtypes: float64(2), object(2)
memory usage: 17.5+ MB
```

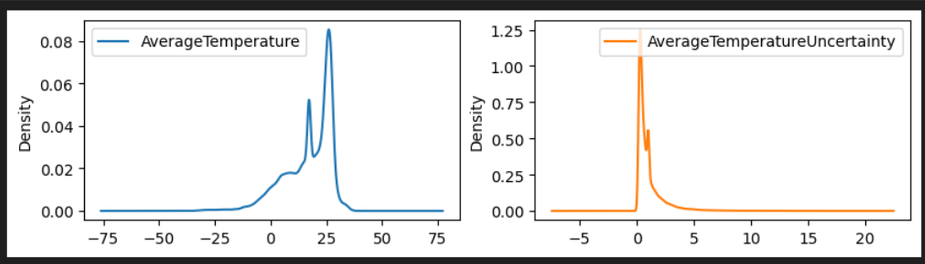
```python
# 5. Handling Missing Data - identify and handle missing values using techniques
# like imputation or removal.



# Exclude non-numeric columns
numeric_columns = df.select_dtypes(include=[np.number]).columns

# Calculate mean using swifter for parallel processing on numeric columns
mean_values = df[['AverageTemperature', 'AverageTemperatureUncertainty']].swifter.apply(lambda x: x.mean())

# Impute missing values with the calculated means
df[['AverageTemperature', 'AverageTemperatureUncertainty']] = df[['AverageTemperature', 'AverageTemperatureUncertainty']].fillna(mean_values)
```

```python
# Apply swifter to the histogram plotting
df.swifter.apply(lambda x: x.hist(bins=20) if x.name in ['AverageTemperature'] else
plt.show()
```

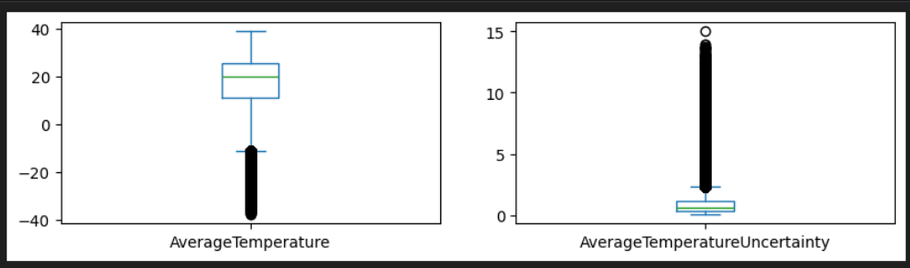
```
Pandas Apply: 100%|██████████| 4/4 [00:00<00:00, 80.54it/s]
```



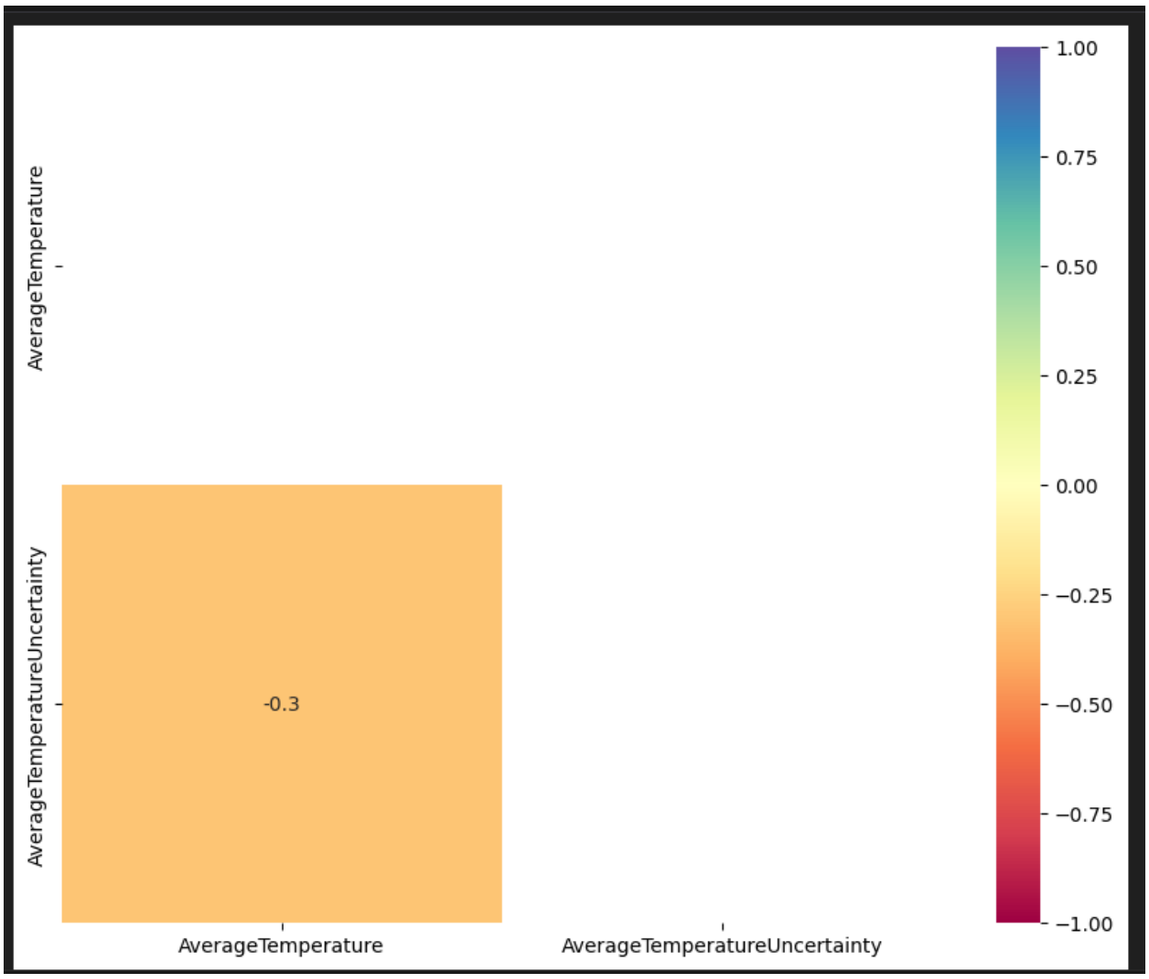
```python
# Density Plots
df.plot(kind='density', subplots=True, layout=(3,3), sharex=False, figsize=(15,8))
plt.show()
```



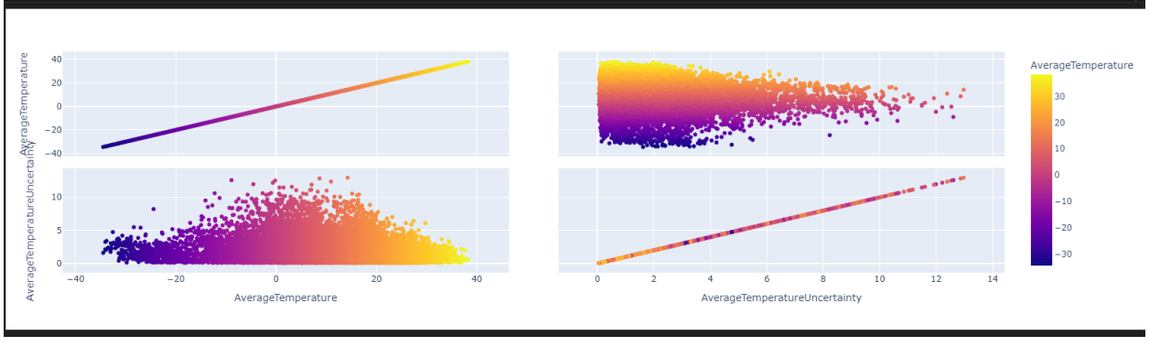

```python
# Box and Whisker Plots
df.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False, figsize=(15,8))
plt.show()
```



```python
# Exclude non-numeric columns from the correlation matrix
numeric_columns = df.select_dtypes(include=[np.number]).columns

# Calculate correlation matrix for all numeric variables
correlations = df[numeric_columns].corr(method='pearson')

# Correlation matrix plot
matrix = np.triu(correlations)
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.heatmap(correlations, annot=True, vmin=-1, vmax=1, center=0, cmap='Spectral', mask=matrix)
plt.show()
```




```python
# Using 10% of the data for visualization with Plotly

import plotly.express as px
df_sample = df.sample(frac=0.1)
fig = px.scatter_matrix(df_sample, dimensions=numeric_columns, color='AverageTemperature')
fig.show()
```



# Classification Dataset EDA

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 1. Load the Data - import the necessary libraries (e.g., Pandas)
# to read your data from a file (e.g., CSV) or a database.

df = pd.read_csv('./cirrhosis.csv')
```

```
# 2. Basic Data Exploration - Check the first few rows of the dataset
# to get an initial sense of the data's structure.

df.head()
```

| | ID | N_Days | Status | Drug | Age | Sex | Ascites | Hepatomegaly | Spiders | Edema | Bilirubin | Cholesterol | Albumin | Copper | Alk_Phos | SGOT | Tryglicerides | Platelets | Prothrombin | Stage |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 400 | D | D-penicillamine | 21464 | F | Y | Y | Y | Y | 14.5 | 261.0 | 2.60 | 156.0 | 1718.0 | 137.95 | 172.0 | 190.0 | 12.2 | 4.0 |
| 1 | 2 | 4500 | C | D-penicillamine | 20617 | F | N | Y | Y | N | 1.1 | 302.0 | 4.14 | 54.0 | 7394.8 | 113.52 | 88.0 | 221.0 | 10.6 | 3.0 |
| 2 | 3 | 1012 | D | D-penicillamine | 25594 | M | N | N | N | S | 1.4 | 176.0 | 3.48 | 210.0 | 516.0 | 96.10 | 55.0 | 151.0 | 12.0 | 4.0 |
| 3 | 4 | 1925 | D | D-penicillamine | 19994 | F | N | Y | Y | S | 1.8 | 244.0 | 2.54 | 64.0 | 6121.8 | 60.63 | 92.0 | 183.0 | 10.3 | 4.0 |
| 4 | 5 | 1504 | CL | Placebo | 13918 | F | N | Y | Y | N | 3.4 | 279.0 | 3.53 | 143.0 | 671.0 | 113.15 | 72.0 | 136.0 | 10.9 | 3.0 |

```
df.describe()
```

| | ID | N_Days | Age | Bilirubin | Cholesterol | Albumin | Copper | Alk_Phos | SGOT | Tryglicerides | Platelets | Prothrombin | Stage |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 418.000000 | 418.000000 | 418.000000 | 418.000000 | 284.000000 | 418.000000 | 310.000000 | 312.000000 | 312.000000 | 282.000000 | 407.000000 | 416.000000 | 412.000000 |
| mean | 209.500000 | 1917.782297 | 18533.351675 | 3.220813 | 369.510563 | 3.497440 | 97.648387 | 1982.655769 | 122.556346 | 124.702128 | 257.024570 | 10.731731 | 3.024272 |
| std | 120.810458 | 1104.672992 | 3815.845055 | 4.407506 | 231.944545 | 0.424972 | 85.613920 | 2140.388824 | 56.699525 | 65.148639 | 98.325585 | 1.022000 | 0.882042 |
| min | 1.000000 | 41.000000 | 9598.000000 | 0.300000 | 120.000000 | 1.960000 | 4.000000 | 289.000000 | 26.350000 | 33.000000 | 62.000000 | 9.000000 | 1.000000 |
| 25% | 105.250000 | 1092.750000 | 15644.500000 | 0.800000 | 249.500000 | 3.242500 | 41.250000 | 871.500000 | 80.600000 | 84.250000 | 188.500000 | 10.000000 | 2.000000 |
| 50% | 209.500000 | 1730.000000 | 18628.000000 | 1.400000 | 309.500000 | 3.530000 | 73.000000 | 1259.000000 | 114.700000 | 108.000000 | 251.000000 | 10.600000 | 3.000000 |
| 75% | 313.750000 | 2613.500000 | 21272.500000 | 3.400000 | 400.000000 | 3.770000 | 123.000000 | 1980.000000 | 151.900000 | 151.000000 | 318.000000 | 11.100000 | 4.000000 |
| max | 418.000000 | 4795.000000 | 28650.000000 | 28.000000 | 1775.000000 | 4.640000 | 588.000000 | 13862.400000 | 457.250000 | 598.000000 | 721.000000 | 18.000000 | 4.000000 |

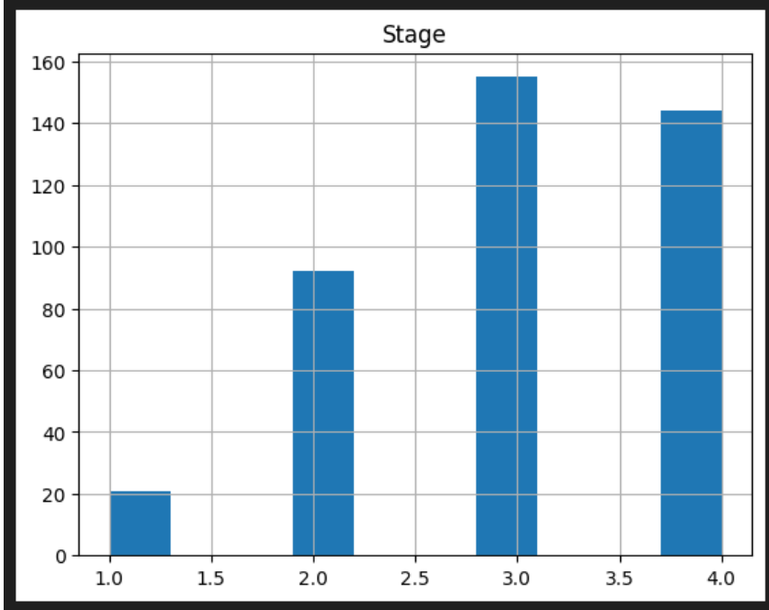
```
#4. Data Information - check the data types of each column,
# the number of non-null values, and memory usage.

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   ID             418 non-null    int64
 1   N_Days         418 non-null    int64
 2   Status         418 non-null    object
 3   Drug           312 non-null    object
 4   Age            418 non-null    int64
 5   Sex            418 non-null    object
 6   Ascites        312 non-null    object
```

```
# 5. Handling Missing Data - identify and handle missing values using techniques
# like imputation or removal.

missing_values = df.isnull().sum()
print("\nMissing Values:")
print(missing_values)

# Fill in missing values in the 'Bilirubin' column with the mean value of the column
df['Bilirubin'] = df['Bilirubin'].fillna(df['Bilirubin'].mean())
```

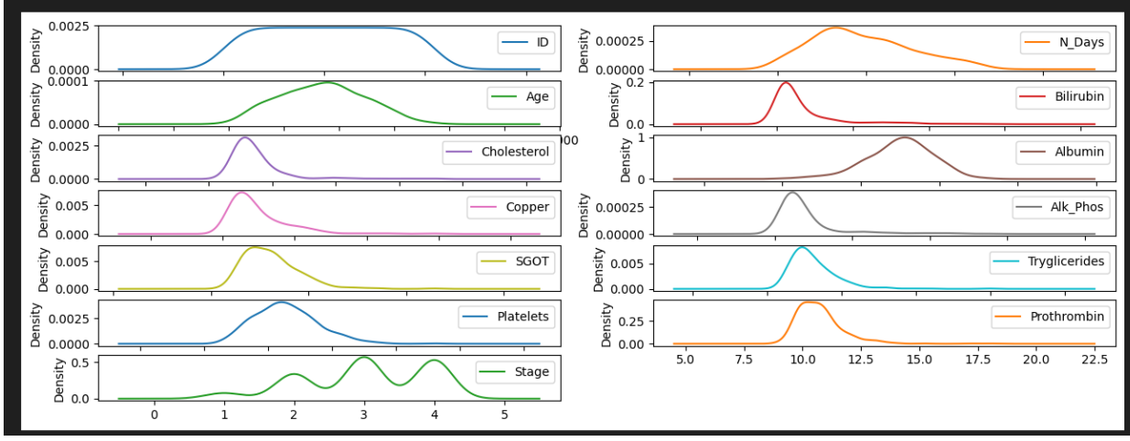
```
Missing Values:
ID                 0
N_Days             0
Status             0
Drug             106
Age                0
Sex                0
Ascites          106
Hepatomegaly     106
Spiders          106
Edema              0
Bilirubin          0
Cholesterol      134
Albumin            0
Copper           108
Alk_Phos         106
SGOT             106
Tryglicerides    136
Platelets         11
Prothrombin        2
Stage              6
dtype: int64
```

```python
#a. Univariate plots focus on a single variable or feature at a time.
#They are used to visualize the distribution and characteristics of individual variables in isolation.

#Histogram
df.hist('Stage')
plt.show()
```



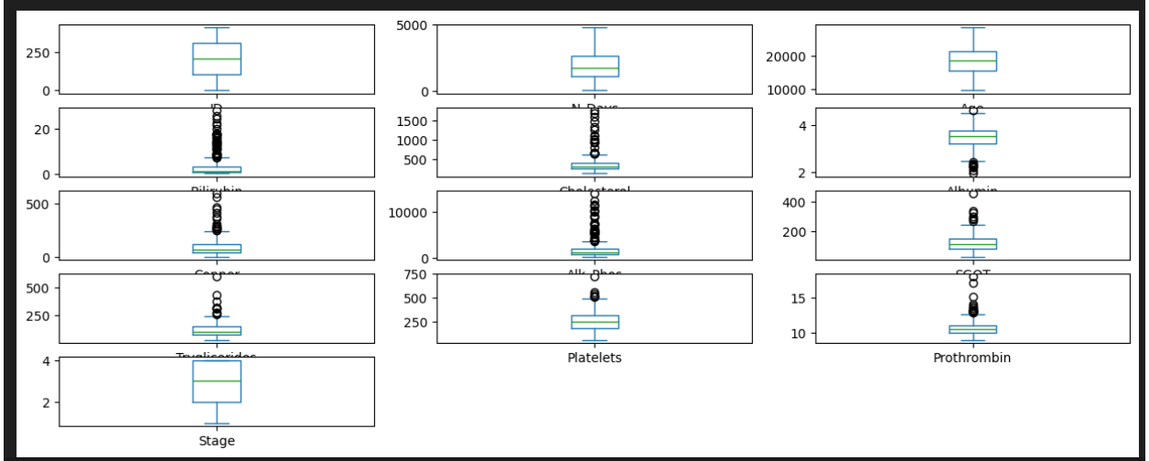

```python
# Check column names and data types
print(df.columns)

# Calculate the number of rows and columns for subplots
num_cols = len(df.columns)
num_rows = (num_cols + 1) // 2  # You can adjust this logic based on your preference

# Plot density plots with a dynamically determined layout
df.plot(kind='density', subplots=True, layout=(num_rows, 2), sharex=False, figsize=(15, 8))
plt.show()
```

```
Index(['ID', 'N_Days', 'Status', 'Drug', 'Age', 'Sex', 'Ascites',
       'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol',
       'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets',
       'Prothrombin', 'Stage'],
      dtype='object')
```

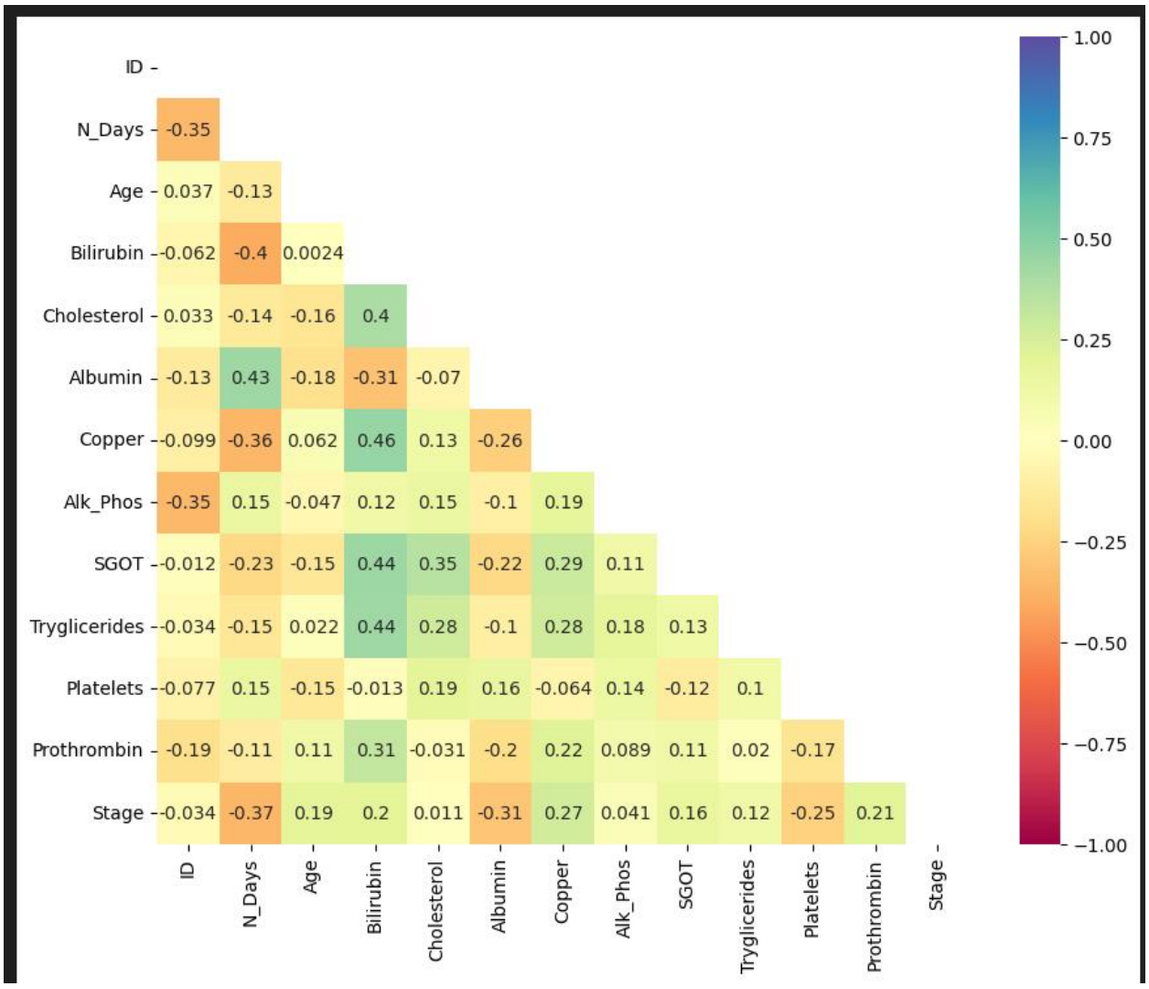


```python
# Check column names and data types
print(df.columns)

# Calculate the number of rows and columns for subplots
num_cols = len(df.columns)
num_rows = (num_cols + 2) // 3  # You can adjust this logic based on your preference

# Plot box and whisker plots with a dynamically determined layout
df.plot(kind='box', subplots=True, layout=(num_rows, 3), sharex=False, sharey=False, figsize=(15, 8))
plt.show()
```

```
Index(['ID', 'N_Days', 'Status', 'Drug', 'Age', 'Sex', 'Ascites',
       'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol',
       'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets',
       'Prothrombin', 'Stage'],
      dtype='object')
```




```python
#Multivariate plots involve the simultaneous visualization of two or more variables to explore relationships,
# correlations, and patterns among them.


# Exclude non-numeric columns from the correlation matrix
numeric_columns = df.select_dtypes(include=[np.number]).columns
correlations = df[numeric_columns].corr(method='pearson')

# Correlation matrix plot
matrix = np.triu(correlations)
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.heatmap(correlations, annot=True, vmin=-1, vmax=1, center=0, cmap='Spectral', mask=matrix)
plt.show()
```



```python
# Check column names
print(df.columns)

# Scatter Plot Matrix with 'Status' as hue
sns.set_theme()
sns.pairplot(df, corner=True, hue='Status')
plt.show()
```