Python Data Analysis & Visualization Notes 🚀

1. Understanding `data.shape`

- `rows, cols = data.shape`
- Returns the number of **rows** and **columns** in the dataset.
- Example:
- `import pandas as pd`
- `df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})`
- `print(df.shape)  # Output: (3, 2)`

2. Descriptive Statistics: `data.describe()`

- Summarizes numerical columns, providing:
  - **Count** (number of non-null values)
  - **Mean** (average value)
  - **Standard Deviation** (spread of data)
  - **Min & Max** (minimum and maximum values)
  - **Quartiles** (25%, 50%, and 75%)
- Example:
- `df.describe()`

3. Counting Unique Values in a Column

- `df['Gender'].unique()` → Returns unique values in the column.
- `df['Smoking'].unique()` → Returns unique values in the Smoking column.
- `df['Gender'].value_counts()` → Returns count of each unique value.
- Example:
- `print(df['Gender'].unique())  # ['Male', 'Female']`
- `print(df['Smoking'].unique()) # ['Yes', 'No']`
- `df['Gender'].value_counts()   # Male: 120, Female: 130`

4. Using `np.unique()` with `return_counts=True`

- Finds unique values and their counts from an array.
- Example:
- `import numpy as np`
- `arrU = np.array([1, 2, 2, 3, 3, 3, 4])`
- `uniques, counts = np.unique(arrU, return_counts=True)`
- `print(uniques)  # [1 2 3 4]`
- `print(counts)   # [1 2 3 1]`

5. Filling Missing Values in a Dataset

```
  data["Age"] = data["Age"].fillna(data["Age"].median())
  data["Blood Pressure"] = data["Blood Pressure"].fillna(data["Blood Pressure"].mean())
  data["Cholesterol Level"] = data["Cholesterol Level"].fillna(data["Cholesterol
Level"].median())
```

- **`.fillna()`** replaces missing values with:
  - **Mean**: Average value of the column.
  - **Median**: Middle value when sorted.
- Used to handle missing data efficiently.

6. Creating a Histogram

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.hist(data['Age'], bins=10, color='skyblue', edgecolor='black')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

- `figsize=(8,6)`: Sets the figure size.
- `plt.hist()`: Creates a histogram.
- `bins=10`: Divides the data into 10 bins.
- `plt.show()`: Displays the plot.

7. Bar Plot for Heart Disease Count

```
import seaborn as sns
plt.figure(figsize=(10, 6))
disease = data["Heart Disease Status"].value_counts()
sns.barplot(x=disease.index, y=disease.values, color="#D8BFD8")  # Light purple
plt.xlabel("Disease")
plt.ylabel("Count")
plt.title("People with or without Heart Disease")
plt.xticks(rotation=45)
plt.show()
```

- `value_counts()`: Counts occurrences of "Yes" and "No".
- `sns.barplot()`: Creates a bar chart.
- `color="#D8BFD8"`: Sets the bars to light purple.
- `plt.xticks(rotation=45)`: Rotates x-axis labels for readability.

Summary Table

Concept	Description
`.shape`	Returns `(rows, columns)`
`.describe()`	Provides statistical summary
`.unique()`	Lists unique values in a column
`.value_counts()`	Counts occurrences of unique values
`np.unique(arr, return_counts=True)`	Returns unique values and their frequencies
`.fillna()`	Fills missing values with mean/median
`plt.hist()`	Creates a histogram
`sns.barplot()`	Creates a bar chart

Review Notes

- Use `.shape` to check dataset dimensions.
- Use `.describe()` to get a quick summary.
- Use `.value_counts()` for categorical data analysis.
- Use `.fillna()` to handle missing values.
- Visualize data using `matplotlib` and `seaborn`.