A) Understanding Line Plot

- Line plot connects data points with straight lines.
- Best for showing trends over time (days, months, years) or over an ordered sequence.
- X-axis: Time/Sequence (ordered)
- Y-axis: Numeric Value

It helps us:

- Show growth/decline over time
- Compare multiple trends on the same chart
- Spot patterns, spikes, drops, seasonality

Key Points:

- Used for Univariate Analysis and Bivariate Analysis
- Line Plots are for Ordered Numerical (X) vs Numerical (Y)
- Line Plots are Not for → Categorical (X) vs Numerical (Y)
- One line = univariate trend; multiple lines = comparative trends
- For unordered X → use scatter plot instead

B) Line Plot — Example 1

1) Create a simple time-series DataFrame

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

mydata = {
    'Date': pd.date_range(start='2025-01-01', periods=10, freq='D'),
    'Sales': [120, 135, 150, 140, 160, 170, 165, 180, 140, 150]
}

mydf = pd.DataFrame(mydata)

mydf.head()
```

```
         Date  Sales
0  2025-01-01    120
1  2025-01-02    135
2  2025-01-03    150
3  2025-01-04    140
4  2025-01-05    160
```

Next steps: Generate code with mydf New interactive sheet

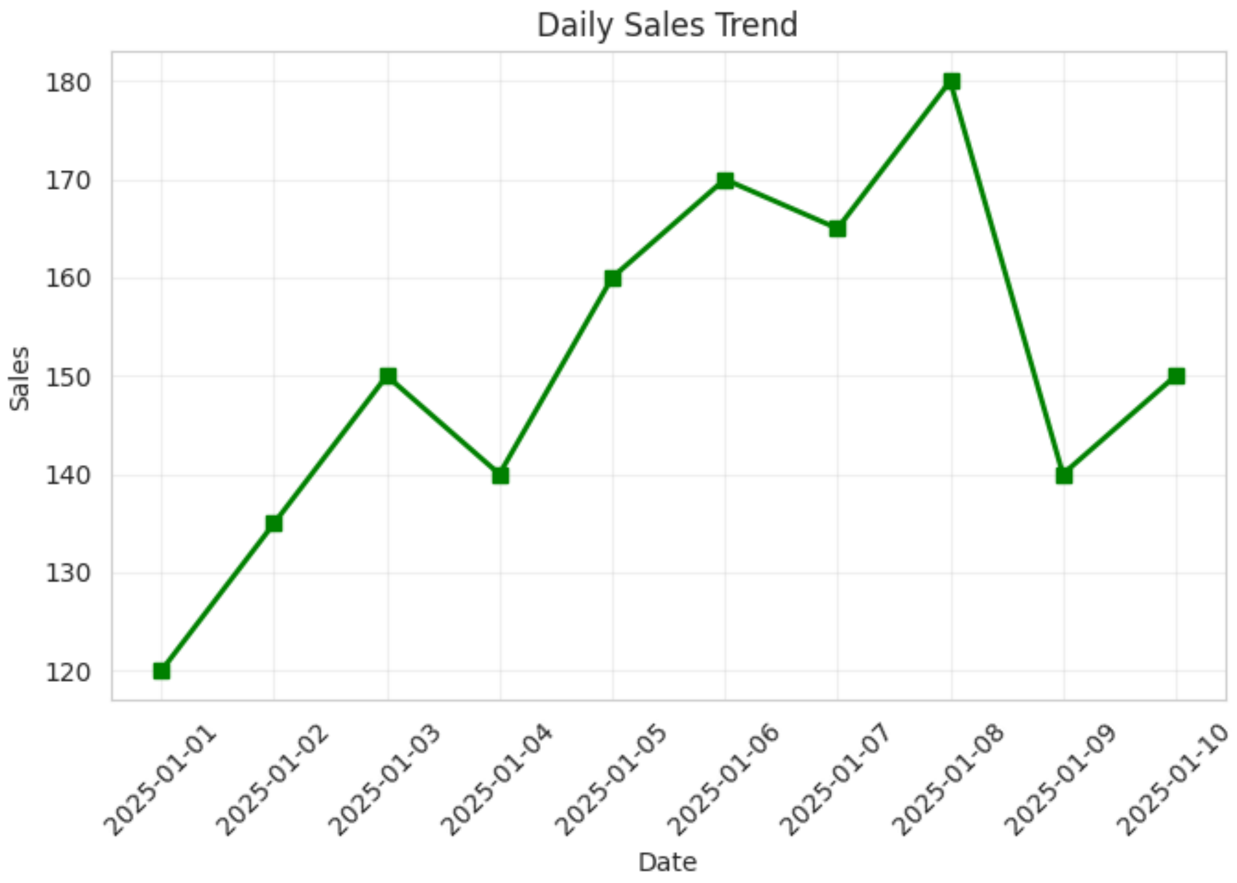
2) Lineplot with Matplotlib

```
plt.figure(figsize=(7, 5))

plt.plot(mydf['Date'], mydf['Sales'], marker='s', linewidth=2, color="g

plt.title("Daily Sales Trend")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.grid(True, alpha=0.3)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Date'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call
last)
                         2 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/indexes/base.py in
get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
```

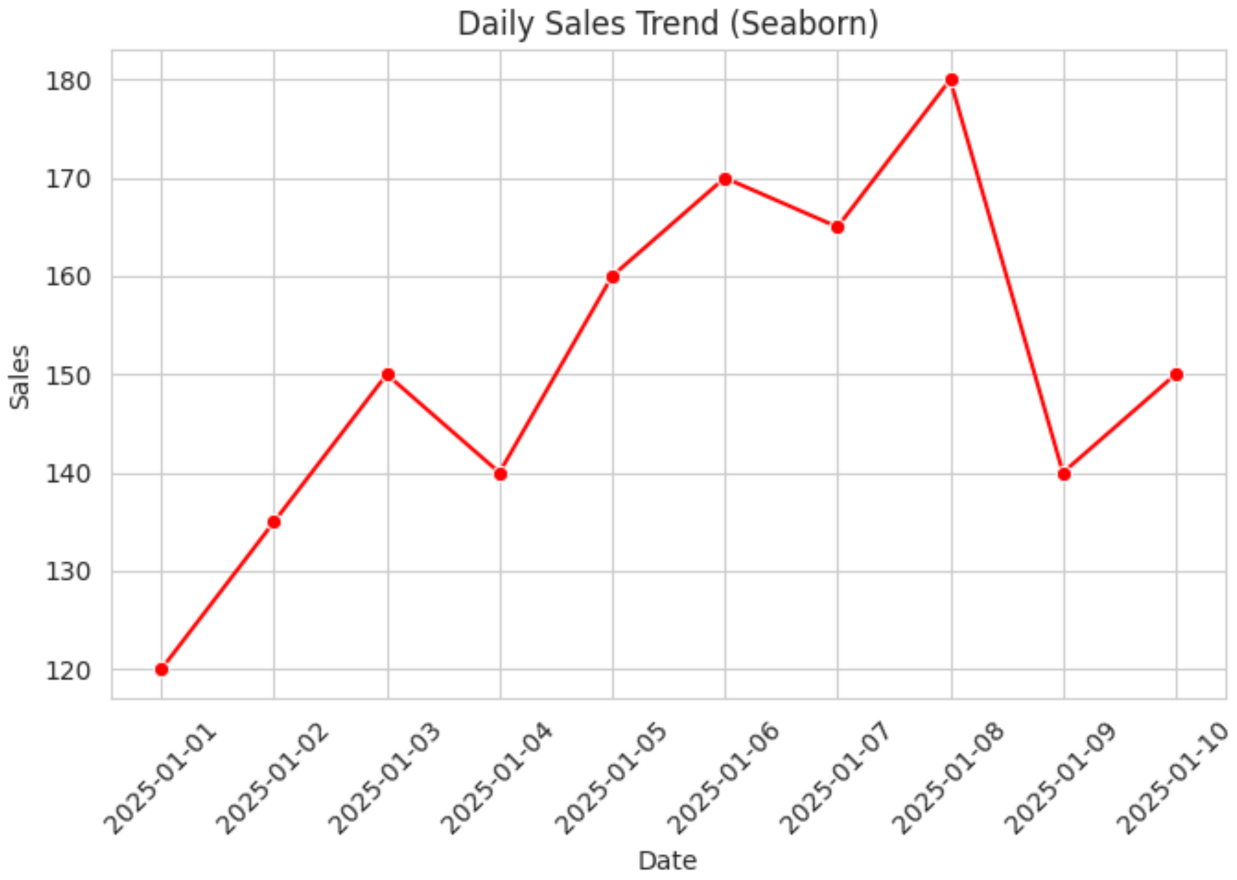
Next steps: Explain error

```
plt.figure(figsize=(7, 5))

plt.plot(mydf['Date'], mydf['Sales'], marker='s', linewidth=2, color='

plt.title("Daily Sales Trend")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.grid(True, alpha=0.3)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



3) Lineplot with Seaborn

```
plt.figure(figsize=(7, 5))

sns.lineplot(data=mydf, x='Date', y='Sales', marker='o',color="red")

plt.title("Daily Sales Trend (Seaborn)")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



C) Line Plot — Example 2

1) Create a DataFrame with two products across months

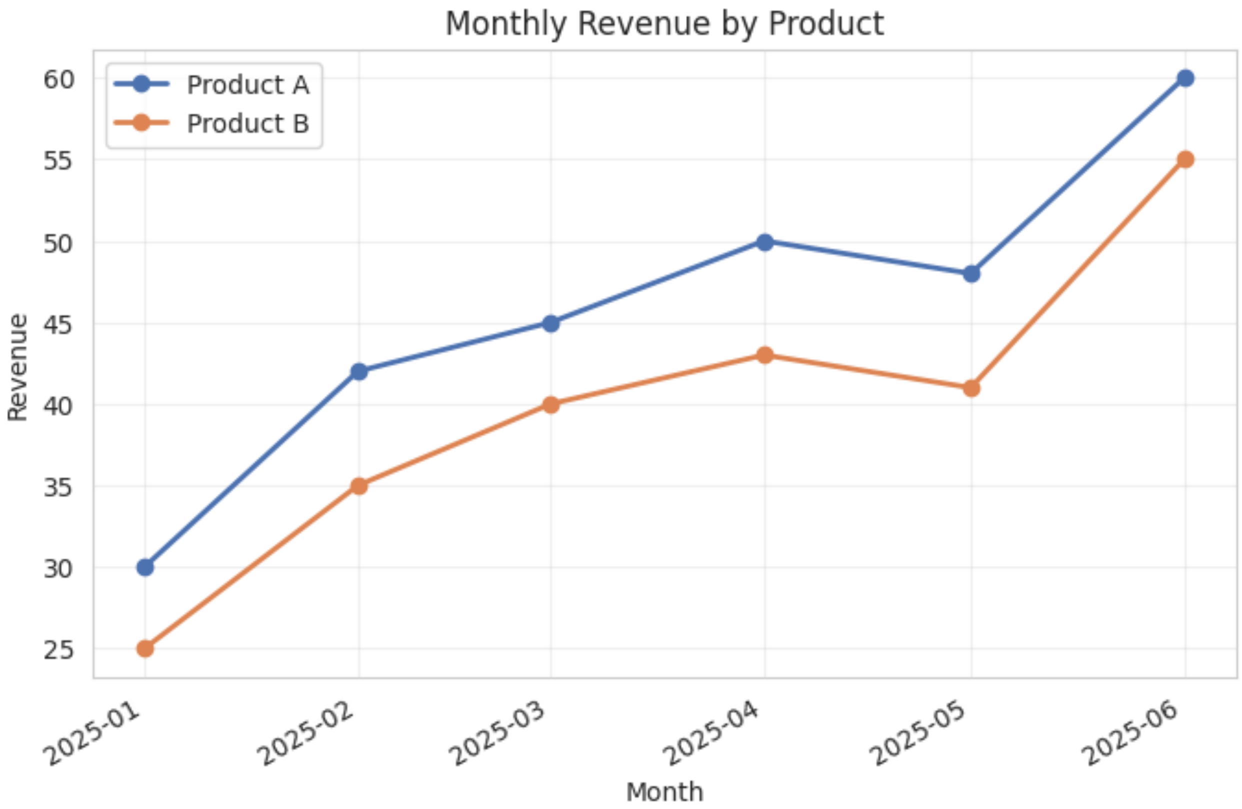
```
months = pd.date_range('2025-01-01', periods=6, freq='MS')

mydata = {
    'Month': list(months) * 2,                          # 6 months repeate
    'Product': ['A']*6 + ['B']*6,                       # 12 values
    'Revenue': [30, 42, 45, 50, 48, 60, 25, 35, 40, 43, 41, 55]  # 12
}

mydf = pd.DataFrame(mydata)

mydf.head()
```

```
        Month  Product  Revenue
0  2025-01-01        A       30
1  2025-02-01        A       42
2  2025-03-01        A       45
3  2025-04-01        A       50
4  2025-05-01        A       48
```

Next steps: Generate code with mydf New interactive sheet

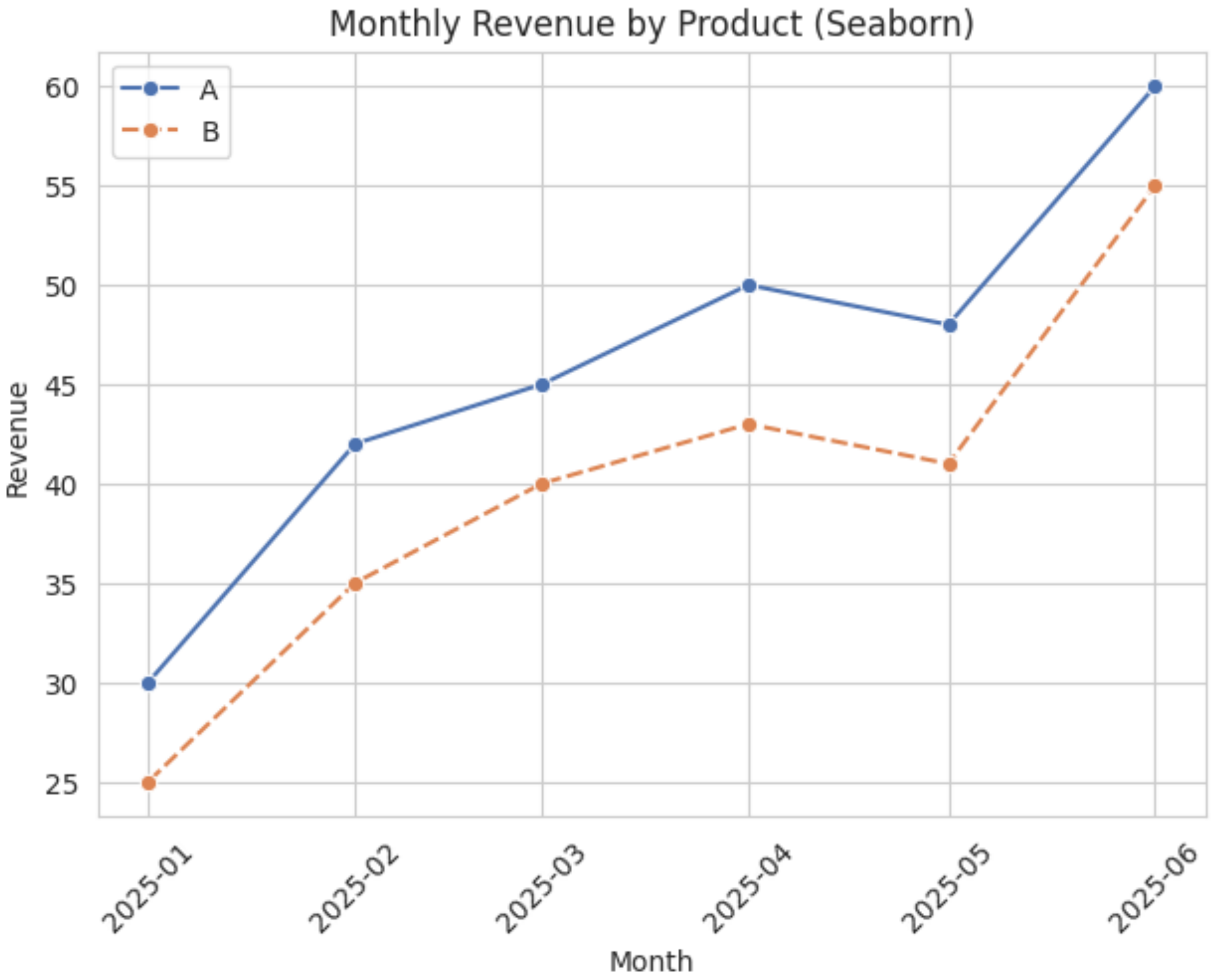
2) Matplotlib — two lines on same axes

```
fig, ax = plt.subplots(figsize=(8, 5))

for prod, sub_df in mydf.groupby('Product'):
    ax.plot(sub_df['Month'], sub_df['Revenue'], marker='o', linewidth=

ax.set_title("Monthly Revenue by Product")
ax.set_xlabel("Month")
ax.set_ylabel("Revenue")
ax.legend()
ax.grid(True, alpha=0.3)
fig.autofmt_xdate()  # Neat date labels
plt.show()
```



3) Seaborn — color by category (easier)

```
    sns.lineplot( data=mydf, x='Month', y='Revenue', hue='Product', style=

    plt.title("Monthly Revenue by Product (Seaborn)")
    plt.xlabel("Month")
    plt.ylabel("Revenue")
    plt.tight_layout()
    plt.xticks(rotation=45)
    plt.legend()

    plt.show()
```



D) Real-Time Use Cases of Line Plots

1) Time Series Analysis (Trends Over Time)

Goal:

- Show how a metric changes over time.

Examples:

- Daily sales over months (Retail)
- Website traffic by day (Digital marketing)
- Stock prices by date (Finance)

Why line plot:

- Best for showing trend, direction, and seasonality.

2) Comparing Multiple Trends

Goal:

- Compare how two or more series behave over the same timeline.

Examples:

- Product A vs Product B monthly revenue
- Male vs Female average scores across semesters
- Website visits vs conversions per week

Why line plot:

- Easy to overlay multiple lines to see relative growth.

3) Monitoring Performance Metrics

Goal:

- Track key performance indicators (KPIs) continuously.

Examples:

- Server CPU or memory usage over time (DevOps)
- Application response time vs load (System Design)
- Employee productivity trends (HR analytics)

Why line plot:

- Shows spikes, dips, anomalies quickly.

4) Forecasting & Predictive Analysis

Goal:

- Show past data and predicted future values together.

Examples:

- Sales forecast for next quarter
- Weather temperature forecast
- Demand prediction curves

7. Exploring Bar Plots

A) Understanding Bar Plot

- Bar chart shows comparisons between categories.
- X-axis: categories (discrete, like Product, City)
- Y-axis: numeric values (like Sales, Revenue)
- Each bar's height shows the value of that category.

It helps us:

- Compare category-wise totals or averages
- Quickly find highest or lowest category
- Visualize categorical distributions

Key Points:

- For Categorical vs Numeric data
- Each bar = one category
- Not for continuous numeric data (use Histogram instead)
- Works well for grouped/aggregated data

B) Simple Bar Plot— Example

1) Create a sample dataset

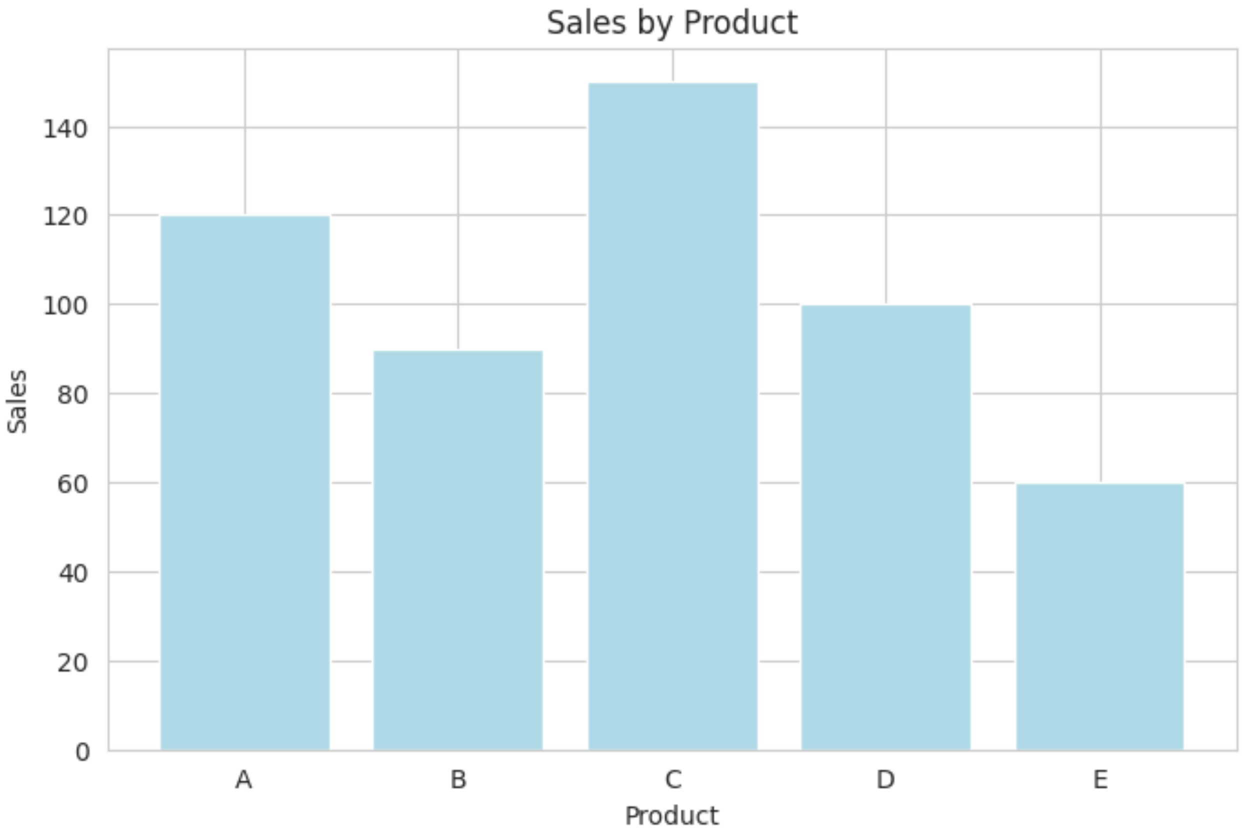
```
data = {
    'Product': ['A', 'B', 'C', 'D', 'E'],
    'Sales': [120, 90, 150, 100, 60]
}

mydf = pd.DataFrame(data)

mydf
```

	Product	Sales
0	A	120
1	B	90
2	C	150
3	D	100
4	E	60

Next steps: Generate code with mydf New interactive sheet

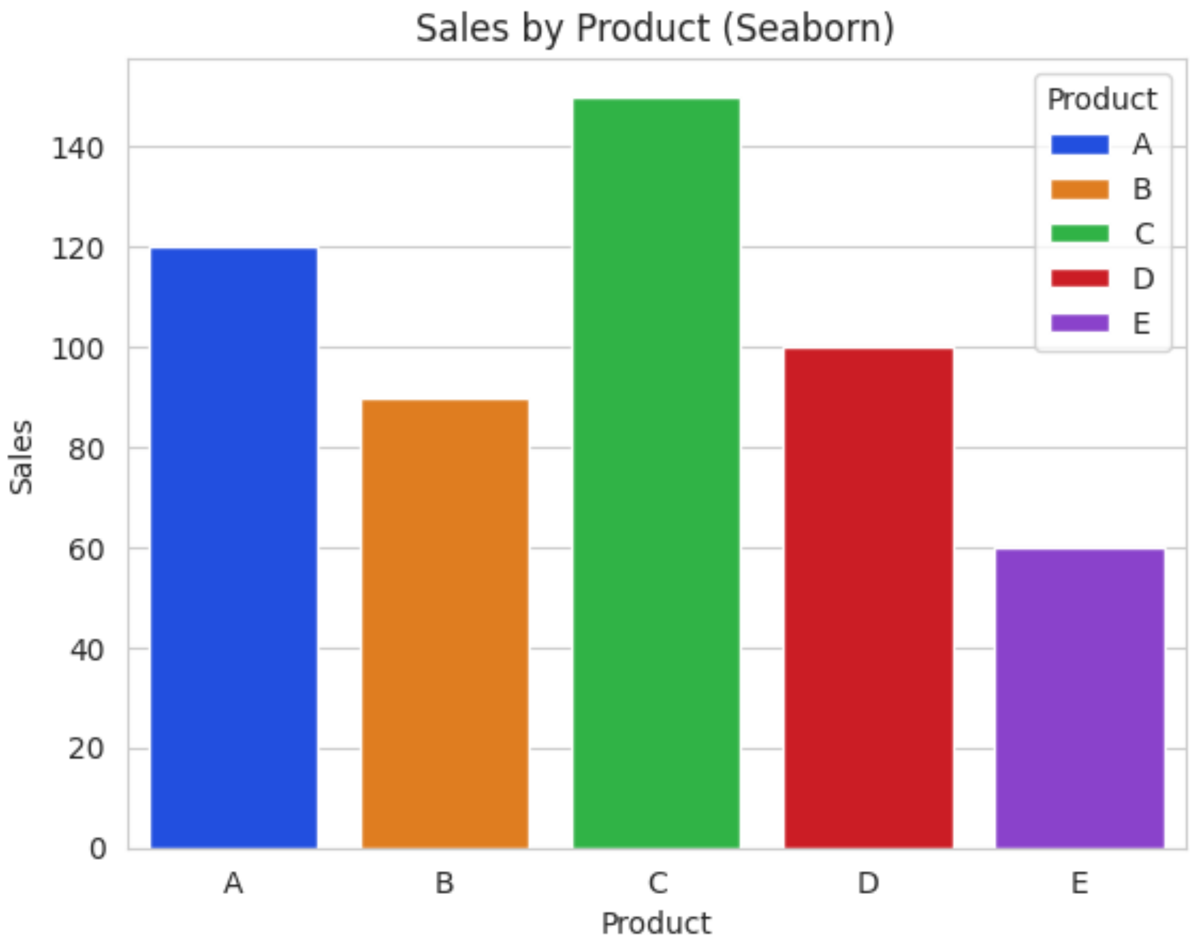
2) Bar Plot using Matplotlib

```
plt.figure(figsize=(8, 5))
plt.bar(mydf['Product'], mydf['Sales'], color='lightblue')

plt.title("Sales by Product")
plt.xlabel("Product")
plt.ylabel("Sales")
plt.show()
```



3) Bar Chart using Seaborn

```
sns.barplot(data=mydf, x='Product',  y='Sales', hue='Product',  palett

plt.title("Sales by Product (Seaborn)")
plt.xlabel("Product")
plt.ylabel("Sales")
plt.show()
```



C) Grouped Bar Plot

- Extension of a bar plot where each category on the X-axis contains multiple sub-categories.
- Bars are placed side by side within each main category.
- Useful for comparing groups across multiple categories at once.
- Best for showing category vs numeric values with sub-group comparisons.

Example:

- Sales by Quarter (main category) split into Region A and Region B (sub-categories).

Key Point:

- Grouped = Side by side bars for each sub-category inside a main category.

Grouped Bar Plot — Example

1) Create a grouped dataset

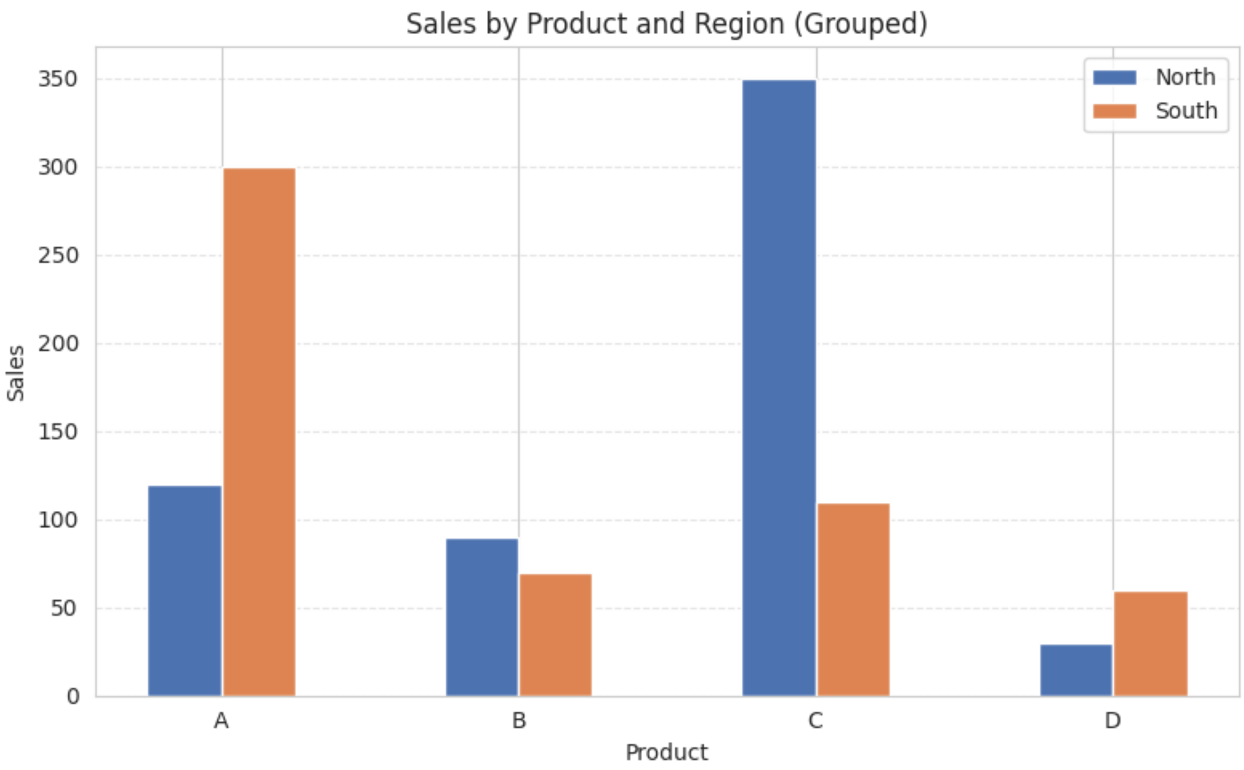
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

mydata = {
    'Product': ['A','A','B','B','C','C','D','D'],
    'Region': ['North','South','North','South','North','South','North
    'Sales': [120, 300, 90, 70, 350, 110,30,60]
}

mydf = pd.DataFrame(mydata)

mydf
```

	Product	Region	Sales
0	A	North	120
1	A	South	300
2	B	North	90
3	B	South	70
4	C	North	350
5	C	South	110
6	D	North	30
7	D	South	60

Next steps: Generate code with mydf New interactive sheet

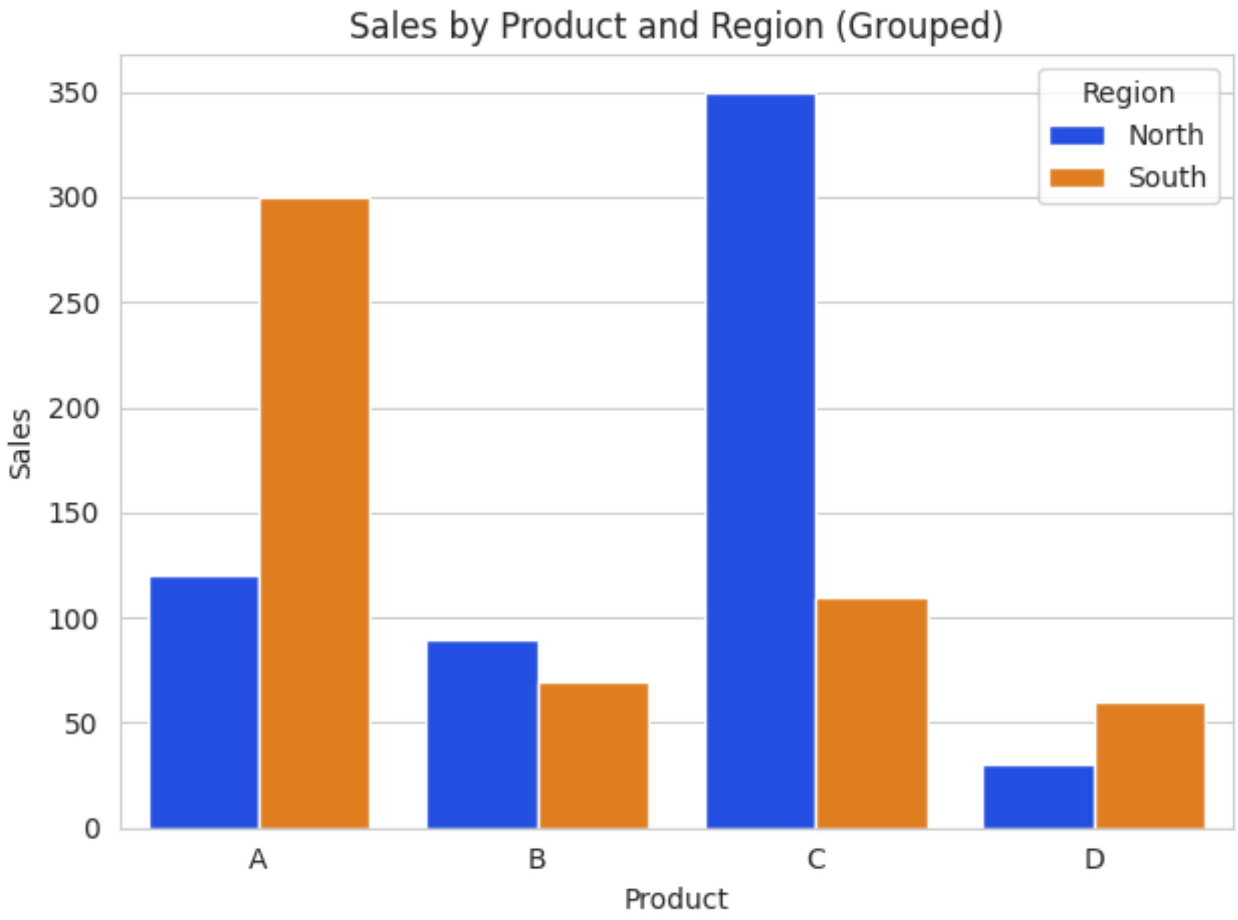
2) Grouped Bar Plot using Matplotlib

```
# Get unique products and regions
products = mydf['Product'].unique() #[A,B,C,D]
regions = mydf['Region'].unique() # [South, North]

x = np.arange(len(products))  # x positions [0,1,2,3]
width = 0.25                   # width of each bar

fig, ax = plt.subplots(figsize=(8, 5))

for i, region in enumerate(regions):
    sales = mydf[mydf['Region'] == region]['Sales']
    ax.bar(x + i*width, sales, width, label=region)

ax.set_xticks(x + width/2)
ax.set_xticklabels(products)
ax.set_title("Sales by Product and Region (Grouped)")
ax.set_xlabel("Product")
ax.set_ylabel("Sales")
ax.legend()
ax.grid(axis='y', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()
```



3) Grouped Bar Plot using Seaborn

```
sns.barplot( data=mydf, x='Product', y='Sales', hue='Region', palette=

plt.title("Sales by Product and Region (Grouped)")
plt.xlabel("Product")
plt.ylabel("Sales")
plt.tight_layout()
plt.show()
```



D) Stacked Bar Plot

- A variation of bar plots where sub-categories are stacked on top of each other in a single bar.
- Each bar's total height represents the sum of all sub-category values.
- Good for showing part-to-whole relationships.
- Often used to show composition of categories (e.g., sales split by regions, expenses split by type).

Example:

- Total Sales per Quarter, stacked by Region (A + B).

Key Point

- Stacked = One bar per category, divided into colored layers for sub-categories.

Stacked Bar Plot — Example

1) Prepare same data in pivot form

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

mydata = {
    'Product': ['A','A','B','B','C','C','D','D'],
    'Region': ['North','South','North','South','North','South','North
    'Sales': [120, 300, 90, 70, 350, 110,30,60]
}

mydf = pd.DataFrame(mydata)

mydf
```

	Product	Region	Sales
0	A	North	120
1	A	South	300
2	B	North	90
3	B	South	70
4	C	North	350
5	C	South	110
6	D	North	30
7	D	South	60

Next steps: Generate code with mydf New interactive sheet

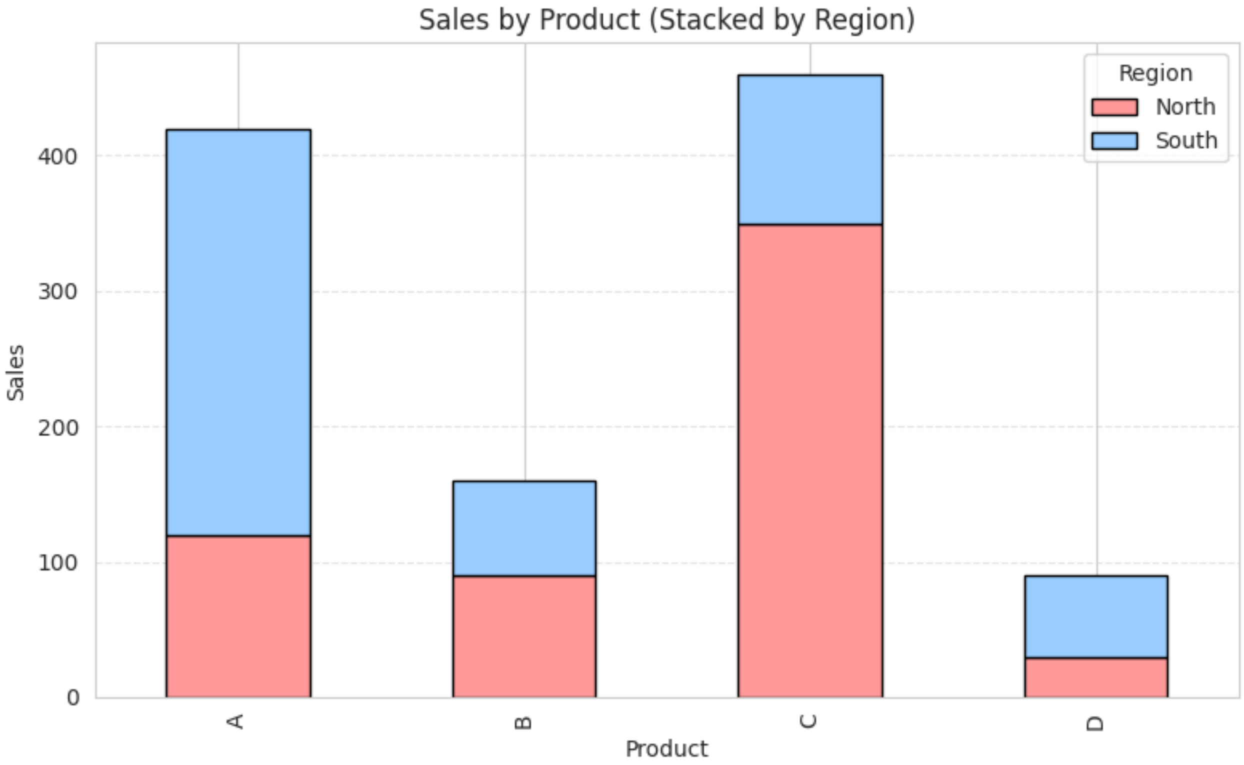
```
pivot = mydf.pivot(index='Product', columns='Region', values='Sales')
pivot
```

Region	North	South
Product		
A	120	300
B	90	70
C	350	110
D	30	60

Next steps: Generate code with `pivot` New interactive sheet

2) Stacked Bar Plot using Matplotlib

```
    pivot.plot(
        kind='bar',
        stacked=True,
        color=['#FF9999', '#99CCFF'],
        figsize=(8, 5),
        edgecolor='black'
    )

    plt.title("Sales by Product (Stacked by Region)")
    plt.xlabel("Product")
    plt.ylabel("Sales")
    plt.legend(title='Region')
    plt.grid(axis='y', linestyle='--', alpha=0.5)
    plt.tight_layout()
    plt.show()
```



Simple Bar Plot

- Shows the value of each category (one bar per category)
- Only one numeric value per category

Grouped Bar Plot

- Compares subcategories side by side within each category
- Want to clearly compare exact values of subcategories

Stacked Bar Plot

- Show subcategory contribution within total
- Want to emphasize part-to-whole relationship

E) Real-World Use Cases of Bar Plots

1) Product / Category Comparison

Goal:

- Compare sales or revenue across different categories.

Examples:

- Sales by product type
- Revenue by region
- Orders by delivery channel

Why bar plot:

- Shows which category performs best/worst at a glance.

2) Demographic Comparisons

Goal:

- Compare numeric metrics across demographic groups.

Examples:

- Average salary by department
- Exam scores by gender
- User counts by age group

Why bar plot:

- Clearly shows group-wise averages or totals.

3) Summarized Aggregated Data

Goal:

- Visualize results of groupby/aggregation operations.

Examples:

- Mean ratings by city
- Total sales by quarter
- Average order size by payment method

Why bar plot:

- Ideal for grouped/aggregated categorical data.

4) Survey or Poll Results

Goal:

- Show how responses are distributed across options.

Examples:

- Preferred brand in a survey
- Satisfaction level counts (Excellent / Good / Average / Poor)

Why bar plot:

- Makes categorical frequencies easy to compare.

8. Exploring Histograms

A) Understanding Histograms

- Histogram shows the distribution of a numeric variable by grouping values into intervals (called bins).
- X-axis → Numeric variable (divided into ranges/bins).
- Y-axis → frequency/Count (how many observations fall in each bin).

It helps us:

- Understand distribution shape (normal, skewed, uniform, etc.)
- Detect outliers or gaps in data
- Compare spreads and ranges
- Decide transformations (log scale, normalization, etc.)

Key Points:

- For Univariate Analysis (one numeric column at a time)
- Only works with numeric data
- Different from Bar Plot:
    - Bar Plot → categorical vs numeric
    - Histogram → numeric vs frequency (distribution)

Bin size matters:

- Too few bins → oversimplified
- Too many bins → too detailed/noisy

B) Histogram — Example 1

1) Create a sample dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Random data
np.random.seed(42)

data = {
    'Age': np.random.randint(18, 60, size=100)
}
df = pd.DataFrame(data)
print(df["Age"].min())
print(df["Age"].max())

df.head()
```

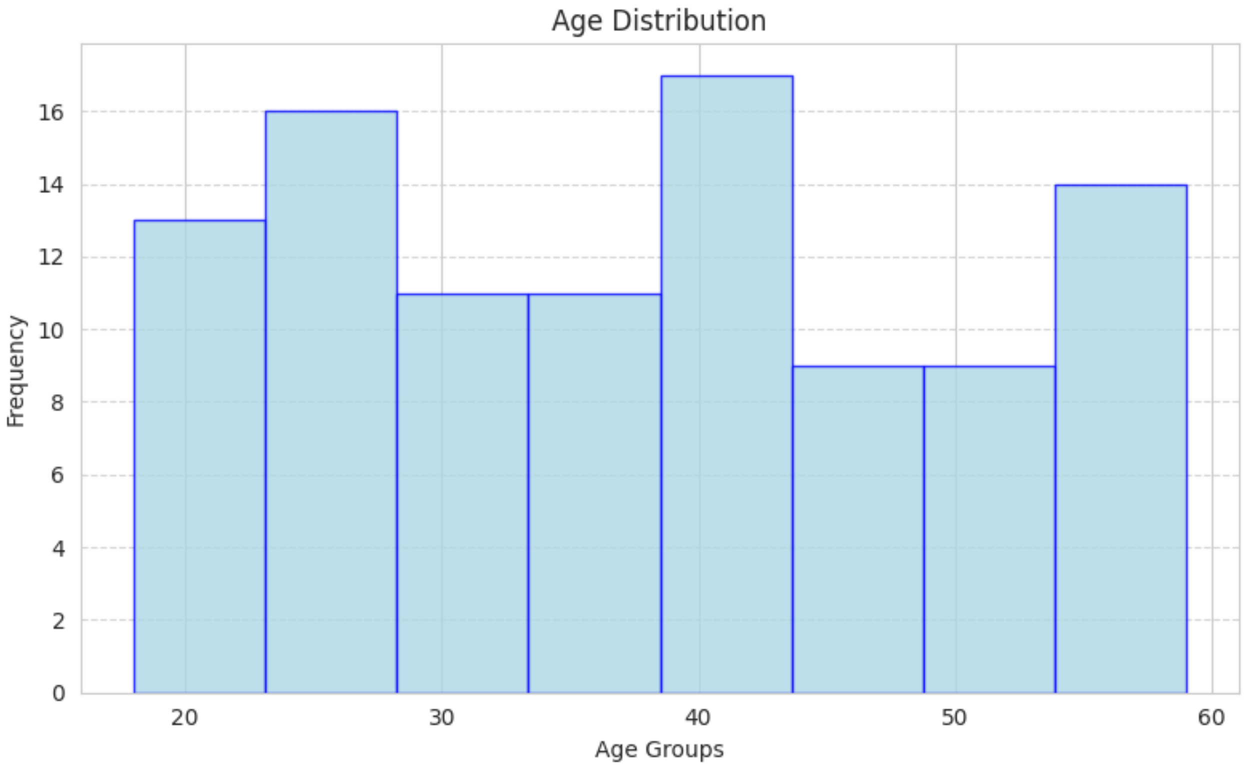
```
18
59
```

	Age
0	56
1	46
2	32
3	25
4	38

Next steps: Generate code with df New interactive sheet

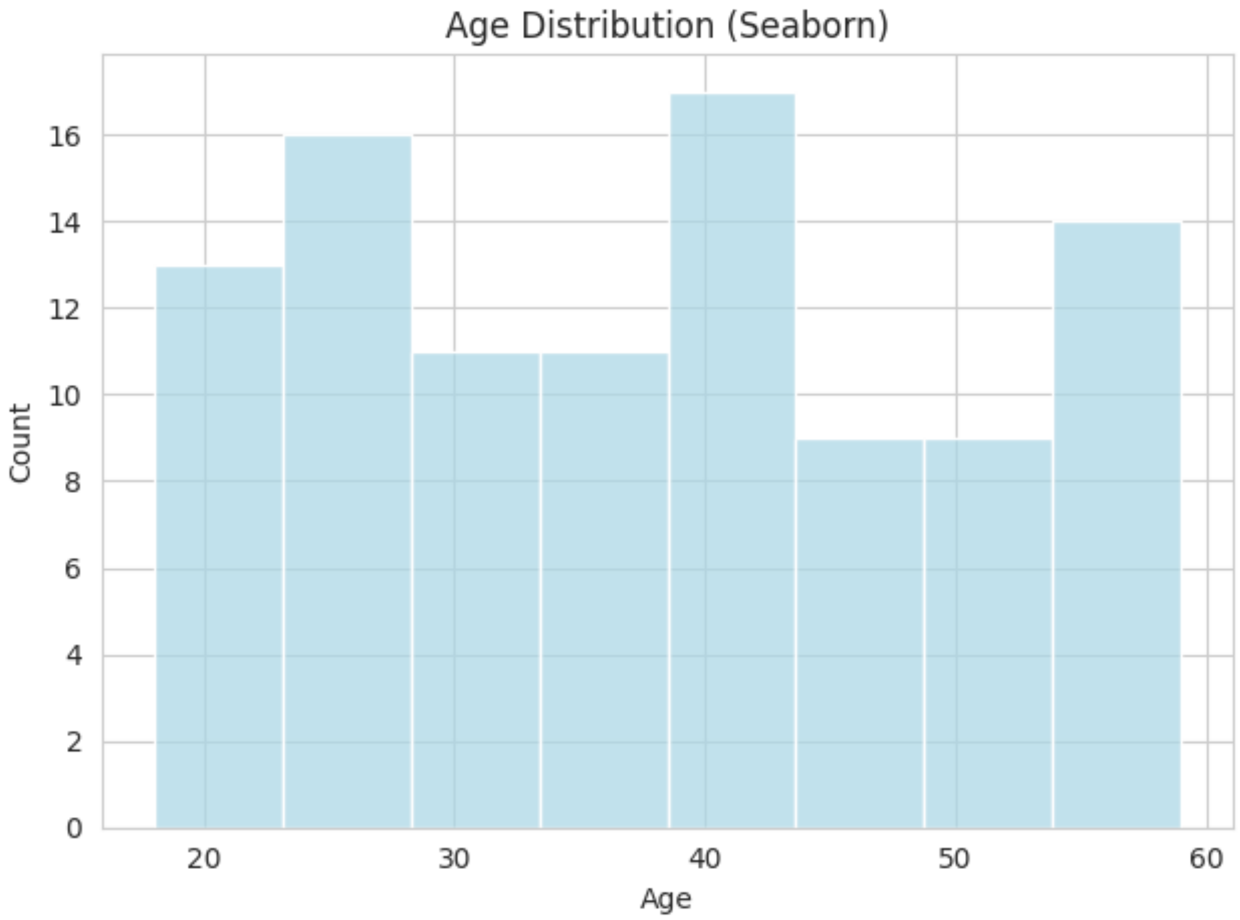
2) Histogram using Matplotlib

```
plt.figure(figsize=(8, 5))

plt.hist(df['Age'], bins=8, color='lightblue', edgecolor='blue', alpha

plt.title("Age Distribution")
plt.xlabel("Age Groups")
plt.ylabel("Frequency")
plt.grid(axis='y', linestyle='--', alpha=0.8)

plt.tight_layout()
plt.show()
```



3) Histogram using Seaborn

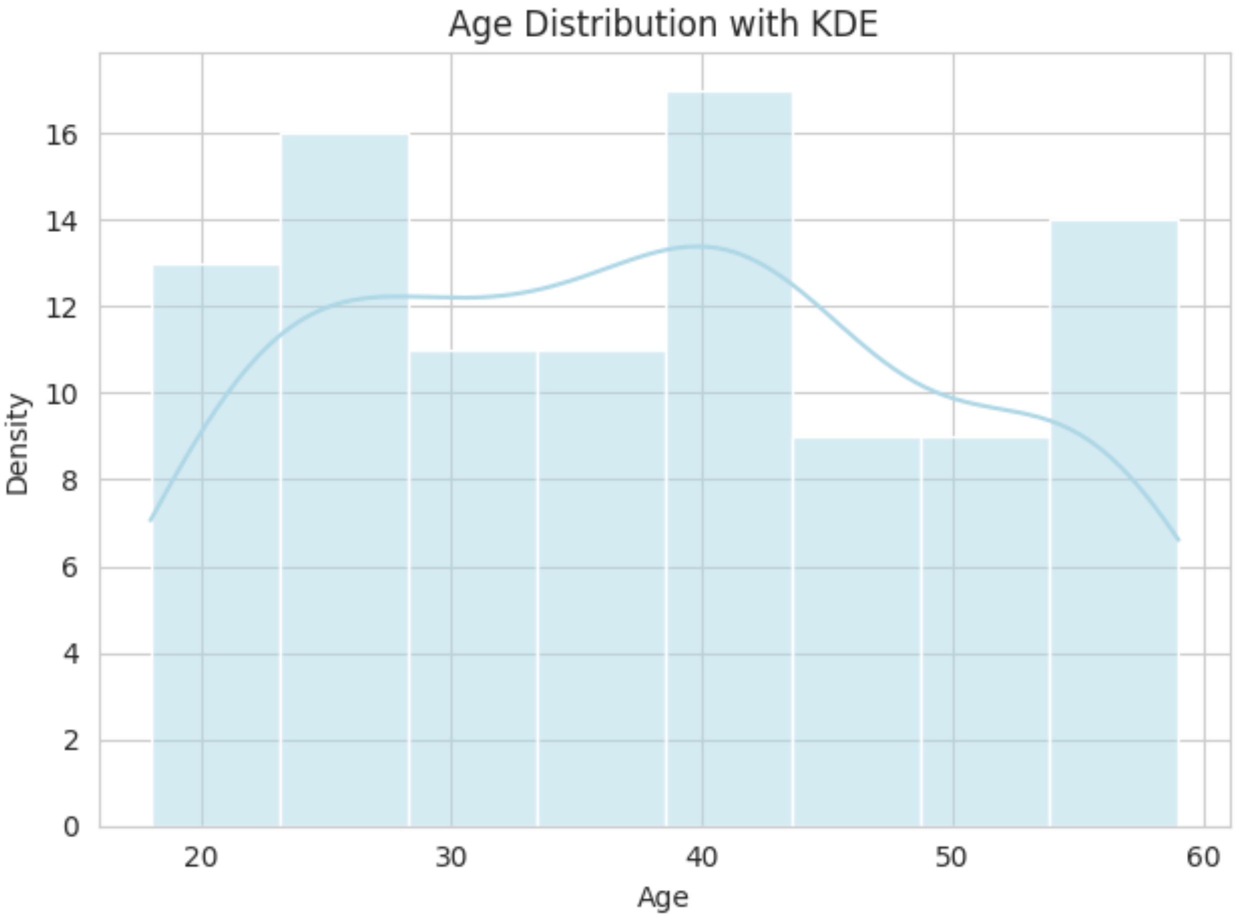
```
    sns.histplot(data=df, x='Age', bins=8, color='lightblue', kde=False)

    plt.title("Age Distribution (Seaborn)")
    plt.xlabel("Age")
    plt.ylabel("Count")

    plt.tight_layout()
    plt.show()
```



C) Histogram — Example 2 (KDE Overlay)

- Kernel Density Estimate (KDE) adds a smooth curve to show probability distribution.
- KDE shows the shape of distribution, not just counts.

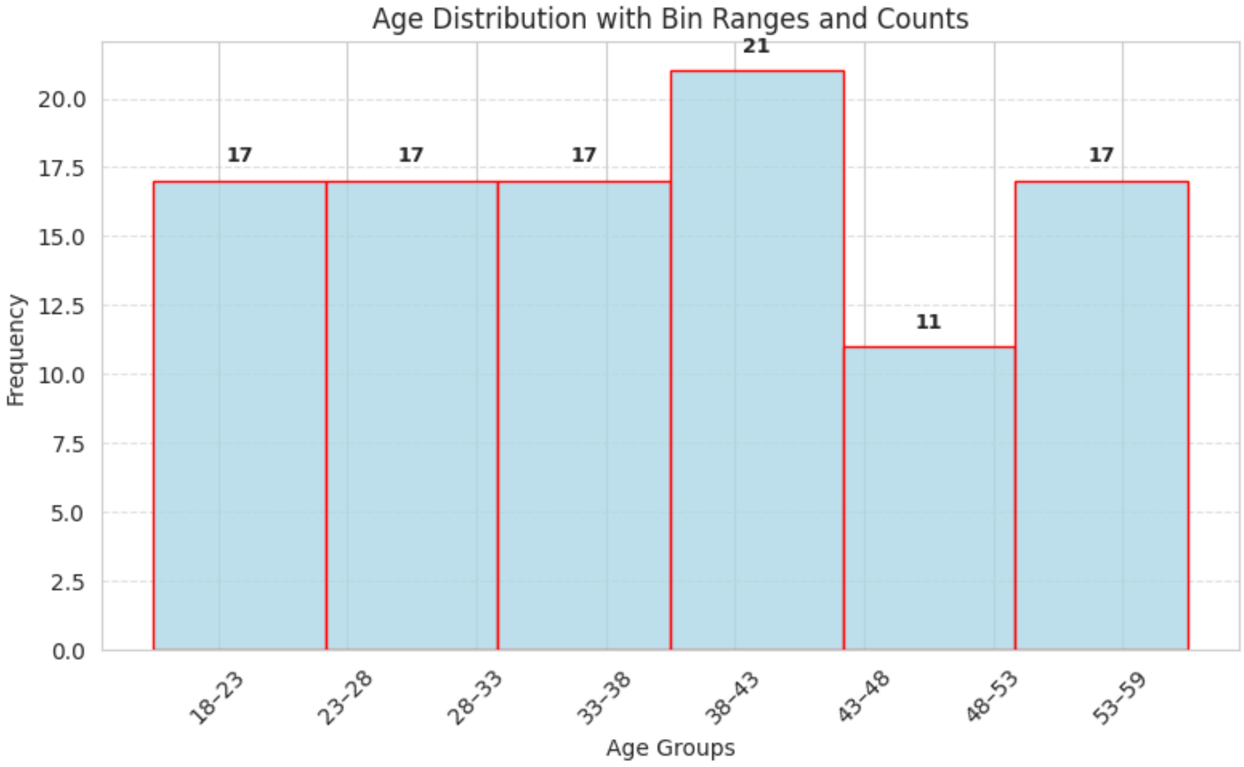
```
sns.histplot(data=df, x='Age', bins=8, color='lightblue', kde=True)

plt.title("Age Distribution with KDE")
plt.xlabel("Age")
plt.ylabel("Density")
plt.tight_layout()
plt.show()
```



Real-world Example

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(8,5))

# Plot histogram with seaborn
ax = sns.histplot(data=df, x="Age", bins=6, color='lightblue', edgeco

# Get bin edges from the patches
```

```
bin_edges = np.linspace(df['Age'].min(), df['Age'].max(), 9)  # 8 bins
bin_labels = [f"{int(bin_edges[i])}–{int(bin_edges[i+1])}" for i in ra

# Add counts on top of each bar
for p, label in zip(ax.patches, bin_labels):
    height = p.get_height()
    ax.text(p.get_x() + p.get_width()/2,
            height + 0.5,
            int(height),
            ha='center', va='bottom', fontsize=9, fontweight='bold')

# Replace X-ticks with bin range labels
ax.set_xticks([(bin_edges[i]+bin_edges[i+1])/2 for i in range(len(bin_
ax.set_xticklabels(bin_labels, rotation=45)

plt.title("Age Distribution with Bin Ranges and Counts")
plt.xlabel("Age Groups")
plt.ylabel("Frequency")
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



D) Real-Time Use-Cases of Histograms

1) Customer Demographics

Goal:

- Understand how people are distributed by age, income, or spending capacity.

Example:

- Age distribution of customers in a retail store
- Income distribution in a survey dataset
- Spending range of online shoppers

Why Histogram?

- Shows clusters of ages/income ranges and detects extremes.

2) Business & Sales Analysis

Goal:

- Analyze transaction amounts or order values.

Example:

- Order value distribution in an e-commerce site
- Frequency of daily sales amounts
- Basket size distribution in supermarkets

Why Histogram?

- Reveals common purchase ranges and detects anomalies.

3) Finance & Risk Management

Goal:

- Assess variability and risk in financial metrics.

Example:

- Distribution of credit scores of applicants
- Distribution of returns on investments
- Daily stock return variations

Why Histogram?

- Helps identify skewness, risk ranges, and tail risks.

4) Education & Exams

Goal:

- Check how students' scores are distributed in a test.

Example:

- Exam score distribution across a class
- Standardized test score ranges
- Assignment grades distribution

Why Histogram?

- Quickly shows performance spread and identifies toppers/outliers.

9. Distribution and Types

- Distribution describes how data values are spread across a range.
- Box Plot + Histogram/KDE help us visualize distributions.

Common Types of Distributions

1. Normal Distribution (Bell Curve)
2. Uniform Distribution
3. Right-Skewed Distribution
4. Left-Skewed Distribution
5. Bimodal Distribution

6. Multimodal Distribution

1) Normal Distribution (Bell Curve)

- Symmetric, unimodal
- Mean ≈ Median ≈ Mode

Example:

- Heights, Exam scores (large groups)

2) Uniform Distribution

- All values equally likely
- Flat shape in histogram

Example:

- Rolling a fair die

3) Right-Skewed Distribution

- Most values are low, few extreme high values

Example:

- Income, house prices

4) Left-Skewed Distribution

- Most values are high, few extreme low values

Example:

- Retirement age, exam pass marks (if almost everyone scores high)

5) Bimodal Distribution

- Two peaks (modes)

Example:

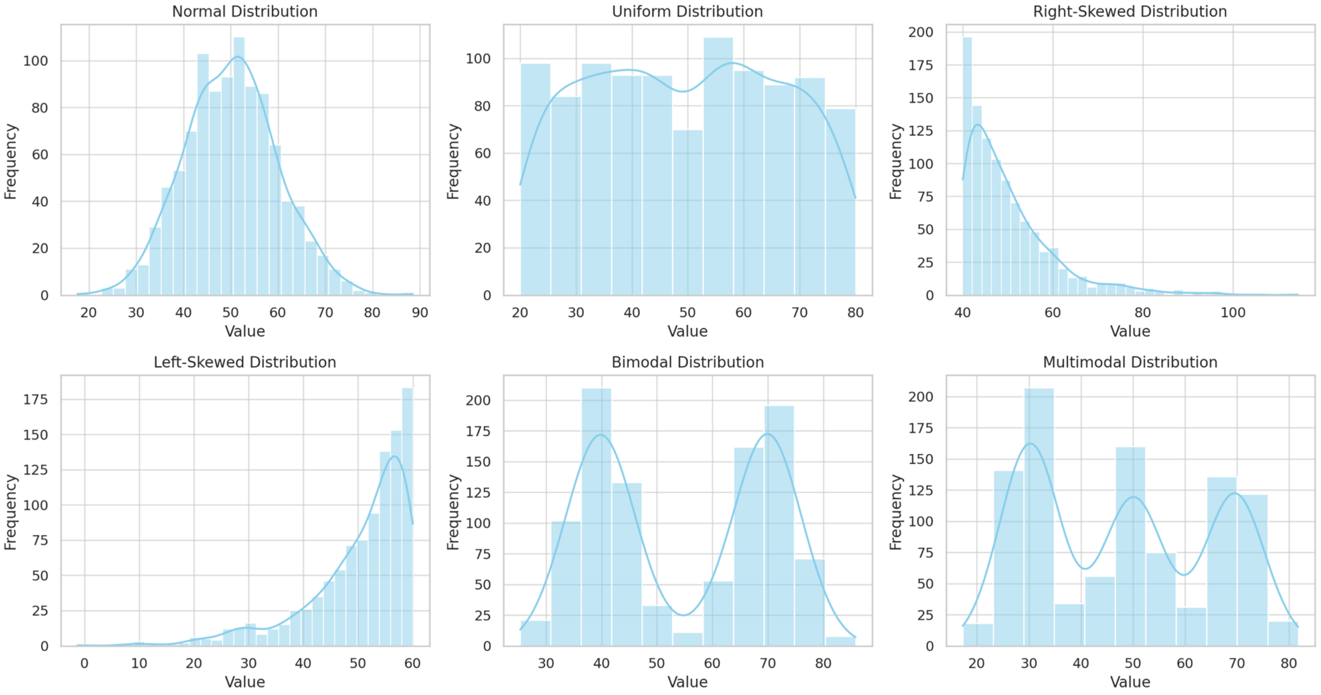
- Test scores with two groups (weak vs strong students)

6) Multimodal Distribution

- More than two peaks

Example:

- Website visits at different times of day (morning, afternoon, night peaks)



Normal Distribution

Uniform Distribution

Right-Skewed Distribution

Left-Skewed Distribution

Bimodal Distribution

Multimodal Distribution

10. Skewness and Box Plots

- Skewness = Tells if distribution is symmetric or leaning (tail direction).
- In a Box Plot, skewness is visible by comparing median's position inside the box and whisker lengths.

Types of Skewness

1. Symmetric Distribution
2. Right Skew (Positive Skew)
3. Left Skew (Negative Skew)

1) Symmetric Distribution

- Median is in the center of the box
- Whiskers are nearly equal length

Example:

- Normal distribution

2) Right Skew (Positive Skew)

- Tail extends to the right (higher values)
- Median is closer to bottom (Q1)

Example:

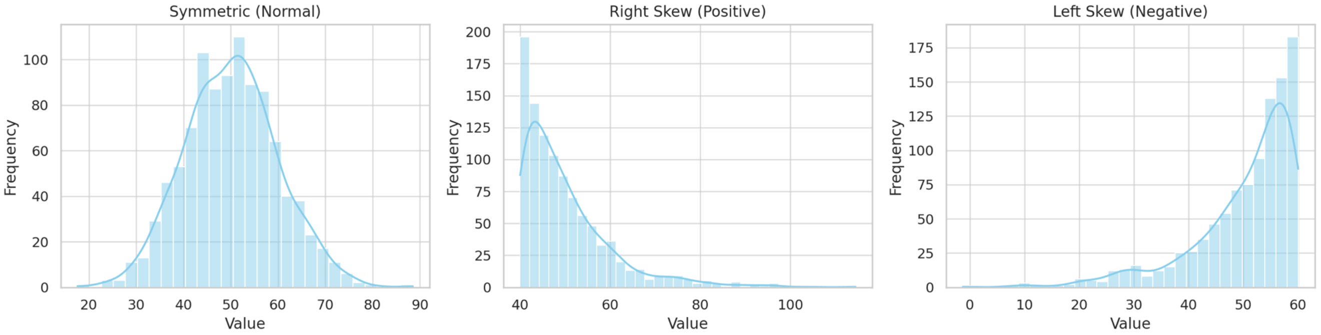
- Income distribution (a few very high earners)

3) Left Skew (Negative Skew)

- Tail extends to the left (lower values)
- Median is closer to top (Q3)

Example:

- Age at retirement (most older, few younger retirees)



11. Exploring Box Plots