📌 Topic: Multivariate Analysis using Seaborn in Python

🎯 Introduction

- This session is part of a machine learning series.
- Previously covered: basic questions, univariate and bivariate analysis.
- Now moving on to **multivariate analysis**, where more than two variables are analyzed together.

🗂️ Agenda

- Multivariate visualization using **Seaborn**.
- Learn to use scatter plots and enrich them with multiple dimensions using hue, style, and size.
- Apply techniques on **different datasets** for better understanding.

🧰 Tools & Libraries Used

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

```
[14]: import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt
```

📊 Datasets Used

1. **tips** – restaurant billing dataset (from Seaborn)
    - **Columns: total_bill, tip, sex, smoker, day, time, size**

```
[3]: tips = sns.load_dataset('tips')
[6]: tips.head()
```

[6]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

2. **Titanic**

```
[7]: titanic = pd.read_csv('train.csv')
[8]: titanic.head()
```

[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

3. **Flights**

```
[9]: flights = sns.load_dataset("flights")
[10]: flights.head()
```

[10]:

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121

4. Iris

```
[11]: iris = sns.load_dataset('iris')
[12]: iris.head()
```

[12]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

📈 Scatter Plot: Numerical vs Numerical

Objective:

Understand relationship between two numerical variables (total_bill and tip).

sns.scatterplot(x=tips['total_bill'], y=tips['tip'])

- **Insight:** Clear linear relationship between bill and tip. As bill increases, tip also increases.

🧠 Multivariate Visualization (Adding Dimensions)

✅ Using hue: Differentiate by gender

sns.scatterplot(x=tips['total_bill'], y=tips['tip'], hue=tips['sex'])

- **Output:** Dots are colored based on gender.
- **Insight:** Gender-based trend visibility.

✅ Using style: Differentiate by smoker status

sns.scatterplot(x=tips['total_bill'], y=tips['tip'], hue=tips['sex'], style=tips['smoker'])

- **Output:** Shapes vary for smokers and non-smokers.

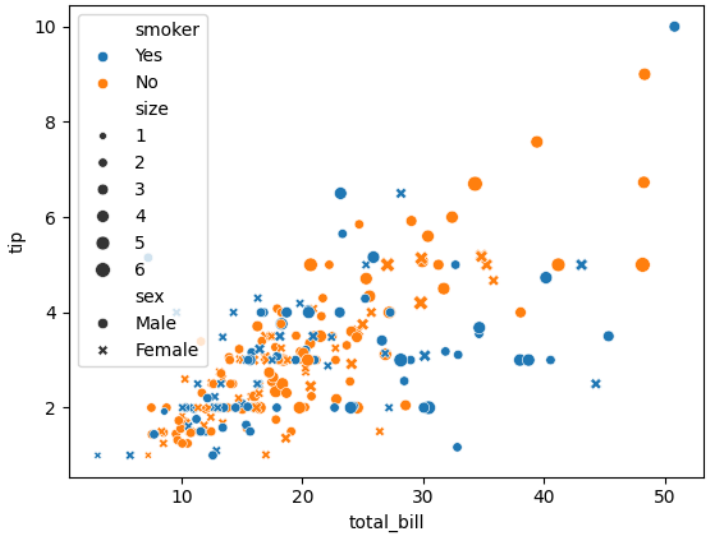
✅ Using size: Represent group size (people at table)

sns.scatterplot(x=tips['total_bill'], y=tips['tip'],hue=tips['sex'], style=tips['smoker'], size=tips['size'])

- **Output:** Dot size changes based on number of people.

```
[13]: sns.scatterplot(x=tips['total_bill'],y=tips['tip'],hue = tips['smoker'],style = tips['sex'],size = tips['size'])
```

```
[13]: <Axes: xlabel='total_bill', ylabel='tip'>
```



📌 Summary of Information Encoded in One Plot

A single scatter plot can simultaneously represent:

1. total_bill → x-axis
2. tip → y-axis
3. sex → color (hue)
4. smoker → shape (style)
5. size → point size

This is called a **Multivariate Plot**, capturing five dimensions of information.

⚠️ Common Errors and Fixes

- ❌ sns.scatterplot(tips['total_bill'], tips['tip'])
  ✅ Use keyword arguments:
  sns.scatterplot(x=tips['total_bill'], y=tips['tip'])
- ❌ Using undefined variable df
  ✅ Replace df with tips if tips is the loaded DataFrame.

🔚 Conclusion

- Seaborn allows detailed and powerful multivariate visualization.
- Scatter plots can go beyond two dimensions using hue, style, and size.
- Great tool for Exploratory Data Analysis (EDA).

📊 Bar Plot & Box Plot using Seaborn (categorical vs numerical)

🧭 Goal

To understand how to visualize **categorical vs numerical** data using **bar plots** and **box plots**. We will also learn how to introduce additional dimensions (multivariate analysis) using parameters like hue.

📌 1. Bar Plot (Numerical vs Categorical)

✅ Use Case:

"What is the average age of passengers in each class (1st, 2nd, 3rd) in the Titanic dataset?"

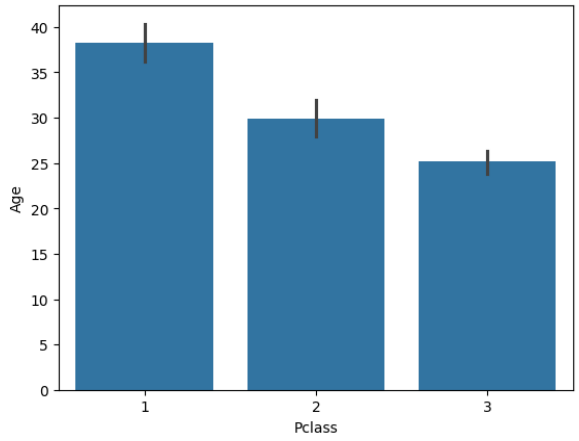
📦 Dataset:

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

titanic = sns.load_dataset('titanic')
```

📉 Basic Bar Plot Syntax

```
 sns.barplot(x='class', y='age', data=titanic)
plt.show()
```

```
[15]: sns.barplot(x=titanic['Pclass'],y=titanic['Age'])
[15]: <Axes: xlabel='Pclass', ylabel='Age'>
```



💡 Interpretation:

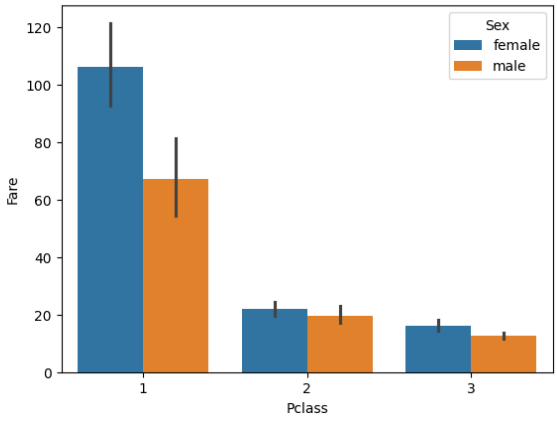
- This plot shows the **average age** of passengers for each travel class.
- E.g., First class may have average age ~38.3, third class slightly lower.

🔁 Multivariate Bar Plot using hue

“What is the average fare in each class, further categorized by gender?”

sns.barplot(x='class', y='fare', hue='sex', data=titanic)
plt.show()

```
[19]: sns.barplot(x=titanic['Pclass'],y=titanic['Fare'],hue = titanic['Sex'] )
[19]: <Axes: xlabel='Pclass', ylabel='Fare'>
```



🎯 Key Insights:

- This adds a second level of comparison: **class × sex**.
- You can visually compare:
  - Male vs Female fare in each class.
- For example, females in first class might have higher average fare than males.

🧠 Interpreting Confidence Intervals in Bar Plots

- Bars include **confidence intervals** by default.
- These can be ignored for visual insights unless specifically required for statistical interpretation.

📦 Example Use Cases of Bar Plot

1. Compare **average fare** by **class**.
2. Compare **average age** by **class**.
3. Compare **average fare** by **class and gender**.
4. Understand relationships using **categorical and numerical** combinations.

📌 2. Box Plot (Numerical vs Categorical)

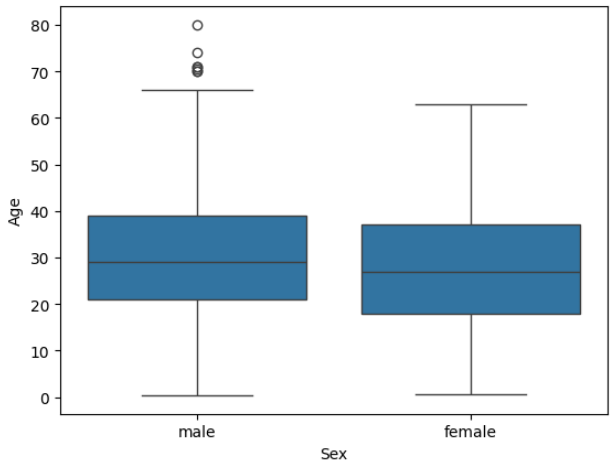
✅ Use Case:

“What is the distribution of age for males and females in the Titanic dataset?”

📉 Basic Box Plot Syntax

```
sns.boxplot(x='sex', y='age', data=titanic)
plt.show()
```

```
[25]: sns.boxplot(x=titanic['Sex'], y=titanic['Age'])
[25]: <Axes: xlabel='Sex', ylabel='Age'>
```



📌 Box Plot Elements:

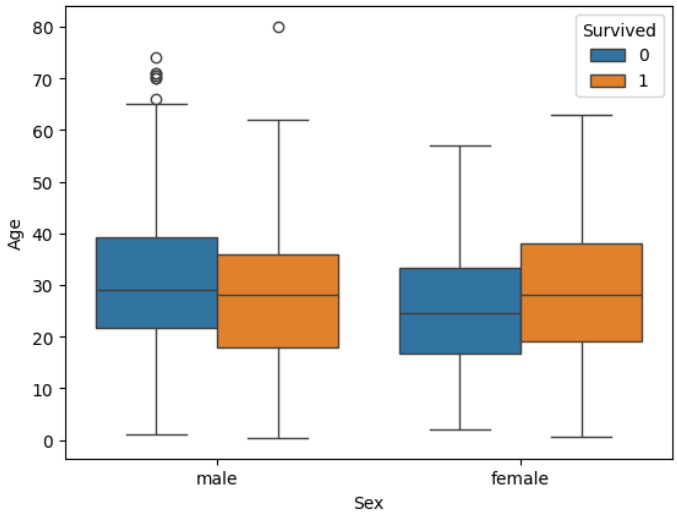
- **Median line** in the box
- **Interquartile range (IQR)** – the box
- **Whiskers** – range of normal data
- **Outliers** – shown as individual points

🔁 Box Plot with Additional Category using hue

“How does age distribution vary by class and sex?”

```
sns.boxplot(x='class', y='age', hue='sex', data=titanic)
plt.show()
```

```
[24]: sns.boxplot(x=titanic['Sex'], y=titanic['Age'],hue = titanic['Survived'])
[24]: <Axes: xlabel='Sex', ylabel='Age'>
```



🔍 Insights:

- View **age distribution** across classes separated by **gender**.
- You can compare:
    - Female vs Male age distribution in each class.
    - Female passengers in first class might be older on average compared to others.

🧠 When to Use Box Plot:

- To show **spread**, **central tendency**, and **outliers** of a numerical variable.
- Suitable when comparing distributions across **multiple categories**.

📌 Advanced Multivariate Example using hue

"Compare age across class and gender."

sns.boxplot(x='class', y='age', hue='sex', data=titanic)

- Use hue to **categorically split** box plots by gender.
- Helps identify **distribution pattern differences**.

🔍 Additional Tips

- You can use .groupby() with .mean() or .describe() for summary stats before plotting.
- Both bar plots and box plots are useful in EDA (Exploratory Data Analysis).

✅ Conclusion

Plot Type	Best For
Bar Plot	Comparing average (mean) of numerical variable across categories
Box Plot	Understanding distribution, spread, and outliers of a numerical variable across categories

📊 DistPlot (Numerical to Categorical Analysis)

Purpose:

To visualize the distribution (PDF - Probability Density Function) of a numerical variable (e.g., Age) based on a categorical variable (e.g., Survival Status).

Example Scenario:

You want to compare age distributions of:

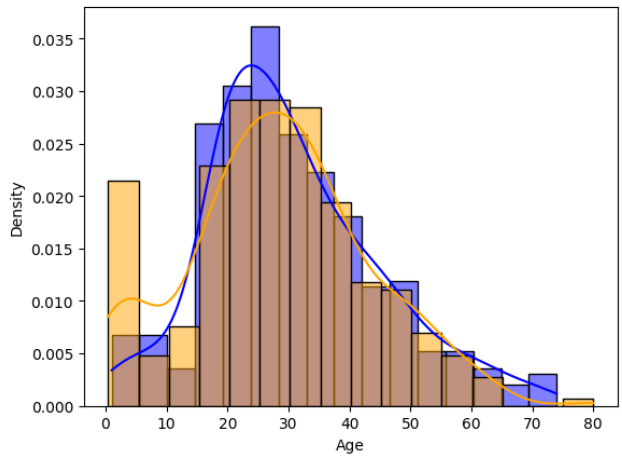
- People who **did not survive** vs.
- People who **survived**

Implementation Steps:

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

# Filter Titanic dataset into survived and not survived groups
not_survived = titanic[titanic['Survived'] == 0]
survived = titanic[titanic['Survived'] == 1]

# Plotting Distplot (PDFs of Age for both groups)
sns.distplot(not_survived['Age'], kde=True, color='blue', stat='density', label='Not Survived', bins=30)
sns.distplot(survived['Age'], kde=True, color='orange', stat='density', label='Survived', bins=30)

plt.legend()
plt.title('Age Distribution by Survival Status')
plt.xlabel('Age')
plt.ylabel('Density')
plt.show()
```

```
[31]: # Filter Titanic dataset into survived and not survived groups
not_survived = titanic[titanic['Survived'] == 0]
survived = titanic[titanic['Survived'] == 1]
# Plotting Distplot (PDFs of Age for both groups)
sns.histplot(not_survived['Age'], kde=True, color='blue', stat='density', label='Not Survived')
sns.histplot(survived['Age'], kde=True, color='orange', stat='density', label='Survived')
```

```
[31]: <Axes: xlabel='Age', ylabel='Density'>
```



Observations:

- **Children (Age < 15)**: Higher probability of survival.
- **Middle-aged (15-30)**: Lower survival probability compared to children.
- **Elderly (>50)**: Mortality probability is again high.

- The blue curve (not survived) dominates for middle age, and the orange curve (survived) dominates for children.
- **Inference**: Likely that during the Titanic accident, **priority was given to children**, hence they had higher chances of survival.

🗺️ HeatMap (Categorical to Categorical Analysis)

Purpose:

To study the relationship between two categorical variables, e.g., **Pclass** (Passenger Class) vs. **Survived**.

Step-by-Step Approach:

1. Create a Pivot Table:

```
import pandas as pd

# Pivot table of counts
pivot_table = pd.crosstab(titanic['Pclass'], titanic['Survived'])
print(pivot_table)
```

2. Plot the HeatMap:

```
sns.heatmap(pivot_table, annot=True, cmap='YlGnBu', fmt='d')
plt.title('Survival Count per Passenger Class')
plt.xlabel('Survived')
plt.ylabel('Passenger Class')
plt.show()
```

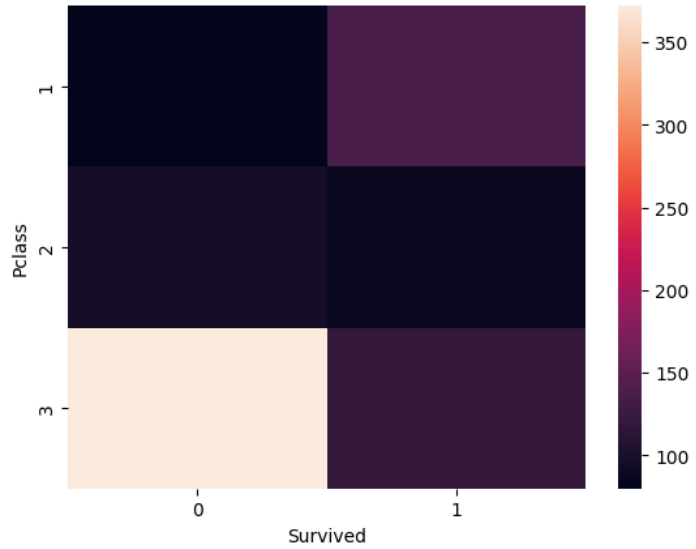
```
[15]: pd.crosstab(titanic['Pclass'],titanic['Survived'])
```

[15]:

Survived	0	1
Pclass		
1	80	136
2	97	87
3	372	119

```
[17]: sns.heatmap(pd.crosstab(titanic['Pclass'],titanic['Survived']))
```

```
[17]: <Axes: xlabel='Survived', ylabel='Pclass'>
```



Interpretation:

- Darker cells → Higher count

- Lighter cells → Lower count
- From the heatmap:
    - **Class 3 had the highest number of deaths.**
    - **Class 1 had a higher survival count.**

Problem with Raw Counts:

Raw numbers don't show the percentage of survivors in each class. For this, we calculate **survival rate per class**.

3. Calculate Survival Percentage per Class:

```
# Group by class and calculate mean survival rate
survival_rate = titanic.groupby('Pclass')['Survived'].mean() * 100
print(survival_rate)
```

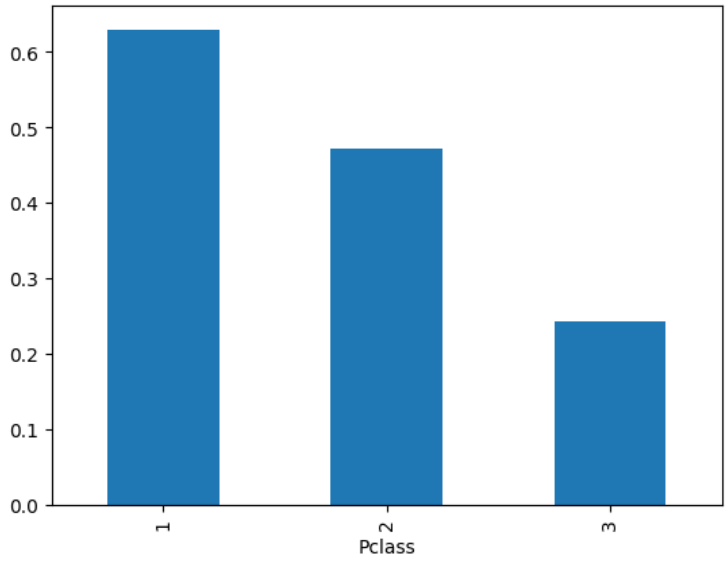
Sample Output:

- Pclass 1: 62%
- Pclass 2: 47%
- Pclass 3: 24%

Conclusion: Higher-class passengers had a better chance of survival.

```
[27]: titanic.groupby('Pclass')['Survived'].mean()

[27]: Pclass
      1    0.629630
      2    0.472826
      3    0.242363
      Name: Survived, dtype: float64
```

```
[26]: (titanic.groupby('Pclass')['Survived'].mean()).plot(kind='bar')

[26]: <Axes: xlabel='Pclass'>
```



👯 Further Analysis Examples:

A. Survival by Gender:

```
gender_survival = titanic.groupby('Sex')['Survived'].mean() * 100
print(gender_survival)
```

- Female: ~74%

- Male: ~18%
- **Conclusion**: Females were prioritized for rescue.

B. Survival by Embarkation Port:

```
embark_survival = titanic.groupby('Embarked')['Survived'].mean() * 100
print(embark_survival)
```

- Cherbourg (C): Higher survival rate.
- Queenstown (Q), Southampton (S): Lower survival rates.
- **Possible reasons**:
    - Socioeconomic status
    - Class distribution
    - More females embarked at Cherbourg, etc.

🧠 **Key Takeaways:**

- **DistPlots** are excellent for visualizing how a numerical feature is distributed across a categorical variable.
- **HeatMaps** reveal frequency or intensity of interaction between two categorical variables.
- **Survival rates** (percentage-based) offer more insight than raw counts.
- Visualization allows you to **extract stories** from data that mere raw inspection cannot reveal.

🔷 **Topic: Clustermap (Categorical vs Categorical Data)**

✅ **Key Concepts:**

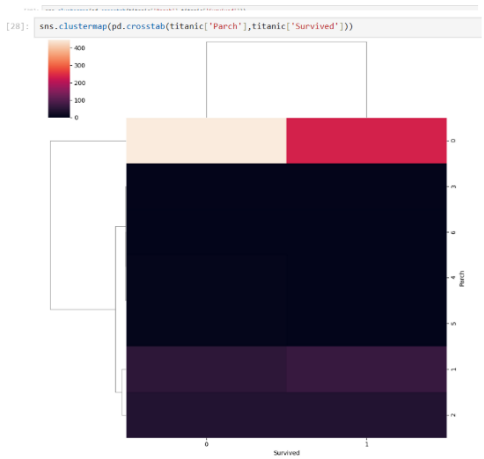
- **Clustermap** is used for visualizing **relationships between categorical columns**, such as Pclass and Survived from the Titanic dataset.
- It is a form of **heatmap** with additional clustering features that rearrange the rows and columns to group similar values together.

🧠 **Example:**

You're analyzing:

python
CopyEdit
titanic.pivot_table(index='Pclass', columns='Survived', aggfunc='size', fill_value=0)
Then plotting:

python
CopyEdit
import seaborn as sns
sns.clustermap(data)



📌 **Insights from Titanic:**

- Many passengers (398) who **traveled alone** died, and very few survived.
- Those who traveled with **1 child** had a relatively higher survival rate.
- People traveling with **6+ companions** showed similar survival patterns to those traveling alone.

💡 **Clustermap Usefulness:**

- Clustermap helps **identify groupings or similarities** between values.
- It can **highlight relationships** between categorical values (like Pclass, Survived, SibSp, etc.).
- For example, classes 4 and 3 might behave similarly in terms of survival probability.

🔷 Topic: PairPlot (Numerical vs Numerical Data)

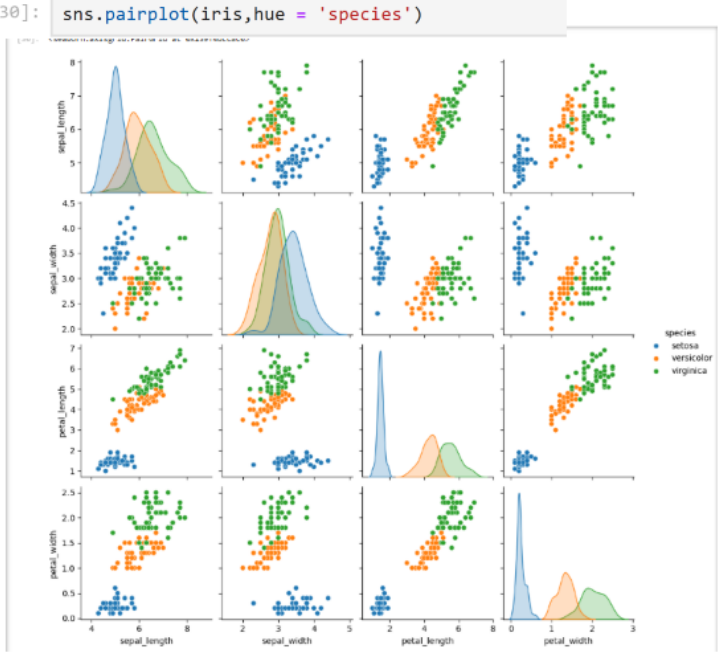
✅ Key Concepts:

- A **pair plot** (Seaborn's pairplot) shows scatter plots between **all pairs of numerical columns**.
- Also includes histograms along the diagonal (i.e., column vs itself).
- It's an **exploratory data analysis (EDA)** tool.

🧠 Example:

python
CopyEdit
import seaborn as sns
sns.pairplot(df)

```
[30]: sns.pairplot(iris,hue = 'species')
```



📌 Use Case:

- You have **multiple numerical columns**, and you want to explore **relationships** or patterns.
- Instead of plotting each scatter plot manually, pairplot() **automatically** detects numerical columns and plots all combinations.

🎯 Benefits:

- Helps identify **linear/non-linear relationships**, **clusters**, or **outliers** in the dataset.
- Can be customized using hue to add **categorical distinctions** (e.g., survival status).

🧪 Example with Hue:

python
CopyEdit
sns.pairplot(df, hue='Survived')

- hue='Survived' color-codes the plots based on the survival status (0 or 1).
- Very useful for **multi-class classification problems** and **multi-variate analysis**.

Comparison & Use Cases:

Feature	Clustermap	Pairplot
Input Type	Categorical-Categorical	Numerical-Numerical
Output	Heatmap with clustering	Grid of scatter plots + histograms
Goal	Visualize similarities/groups	Explore pairwise numerical relations
Use With	Titanic (e.g., Pclass vs Survived)	Titanic (e.g., Age, Fare, SibSp, etc.)
Extras	Dendrograms for grouping	hue to highlight categorical impact

Summary:

- Use **Clustermap** when you're dealing with **categorical vs categorical** relationships and want to understand **group-level similarity**.
- Use **PairPlot** for **quick EDA** on **numerical data**, especially when dataset has many numerical features.
- Both tools are **powerful visualizations** in Seaborn to uncover patterns, outliers, and correlations early in the analysis.

Line Plot (Numerical vs Numerical)

Definition:

- A **line plot** connects data points with lines, and is especially useful when the **x-axis variable is time-based** (e.g., Year, Month, Date).
- Helps visualize **trends, growth, or patterns** over time.

When to Use Line Plots:

Use a line plot when:

- You have **time series data**.
- You want to analyze the **trend or fluctuation over time**.
- Examples:
  - Passenger growth over years.
  - Monthly temperature variations.
  - Daily stock prices.

Example Dataset:

- Dataset contains:
  - Year
  - Month
  - Number of passengers

You want to **analyze how air travel increased over time**.

Data Preparation Steps:

1. **Group by Year**:

   python

CopyEdit
df.groupby('Year')['Passengers'].sum()

2. **Reset index to convert it into a flat DataFrame**:

   python
   CopyEdit
   df_grouped = df.groupby('Year')['Passengers'].sum().reset_index()

📈 **Creating a Line Plot (Yearly Trend):**

python

CopyEdit

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

sns.lineplot(data=df_grouped, x='Year', y='Passengers')
plt.title("Passenger Growth Over Years")
plt.show()
```

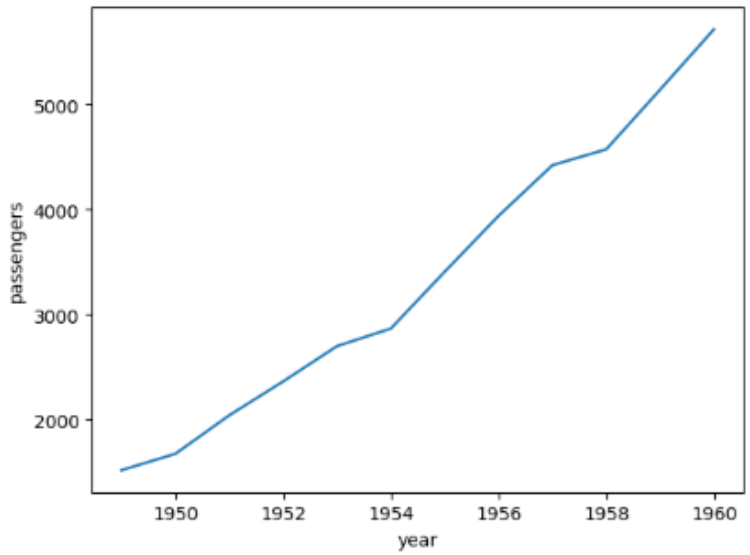
Lineplot

```
[31]: flights.head()
```

[31]:

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121

```
[35]: new = flights.groupby('year')['passengers'].sum().reset_index()
[38]: sns.lineplot(x=new['year'],y=new['passengers'])
[38]: <Axes: xlabel='year', ylabel='passengers'>
```



- This shows **how passenger counts change year by year**.
- Clear growth trend between 1948 to 1960 is observable.
- Minor dips could indicate **seasonal or economic factors**.

📊 Monthly Analysis (Seasonality):

Use a pivot table:

pivot_df = df.pivot_table(values='Passengers', index='Month', columns='Year')

- This gives a **monthly vs yearly passenger matrix**.
- Helps analyze **seasonal patterns**.

sns.heatmap(pivot_df, cmap='YlGnBu')

- January-February often behave similarly.
- July-August show **high travel volume** (summer vacation peak).
- November-December may reflect holiday patterns.

```
[42]: flights.pivot_table(values = 'passengers' , index = 'month' , columns = 'year')
```

```
C:\Users\anmol sarin\AppData\Local\Temp\ipykernel_23916\3832272452.py:1: FutureWarning: The de
ge to observed=True in a future version of pandas. Specify observed=False to silence this warn
  flights.pivot_table(values = 'passengers' , index = 'month' , columns = 'year')
```

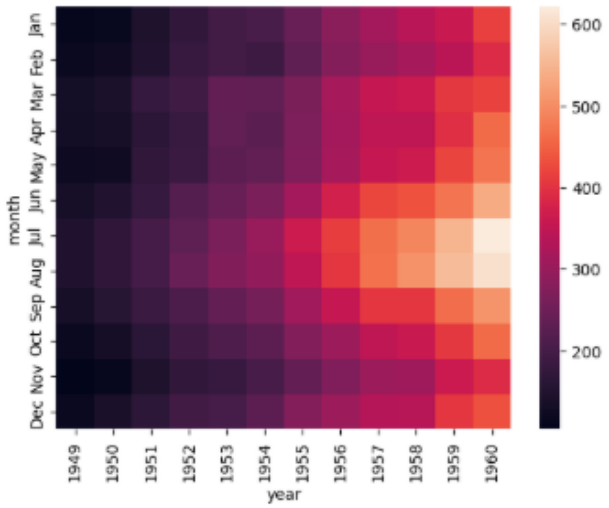
[42]:

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
Jan	112.0	115.0	145.0	171.0	196.0	204.0	242.0	284.0	315.0	340.0	360.0	417.0
Feb	118.0	126.0	150.0	180.0	196.0	188.0	233.0	277.0	301.0	318.0	342.0	391.0
Mar	132.0	141.0	178.0	193.0	236.0	235.0	267.0	317.0	356.0	362.0	406.0	419.0
Apr	129.0	135.0	163.0	181.0	235.0	227.0	269.0	313.0	348.0	348.0	396.0	461.0
May	121.0	125.0	172.0	183.0	229.0	234.0	270.0	318.0	355.0	363.0	420.0	472.0
Jun	135.0	149.0	178.0	218.0	243.0	264.0	315.0	374.0	422.0	435.0	472.0	535.0
Jul	148.0	170.0	199.0	230.0	264.0	302.0	364.0	413.0	465.0	491.0	548.0	622.0
Aug	148.0	170.0	199.0	242.0	272.0	293.0	347.0	405.0	467.0	505.0	559.0	606.0
Sep	136.0	158.0	184.0	209.0	237.0	259.0	312.0	355.0	404.0	404.0	463.0	508.0
Oct	119.0	133.0	162.0	191.0	211.0	229.0	274.0	306.0	347.0	359.0	407.0	461.0
Nov	104.0	114.0	146.0	172.0	180.0	203.0	237.0	271.0	305.0	310.0	362.0	390.0
Dec	118.0	140.0	166.0	194.0	201.0	229.0	278.0	306.0	336.0	337.0	405.0	432.0

```
[40]: sns.heatmap(flights.pivot_table(values = 'passengers' , index = 'month' , columns = 'year'))
```

```
C:\Users\anmol sarin\AppData\Local\Temp\ipykernel_23916\2636690497.py:1: FutureWarning: The de
ge to observed=True in a future version of pandas. Specify observed=False to silence this warn
  sns.heatmap(flights.pivot_table(values = 'passengers' , index = 'month' , columns = 'year'))
```

```
[40]: <Axes: xlabel='year', ylabel='month'>
```



🧪 Using Clustermap for Time Series Grouping:

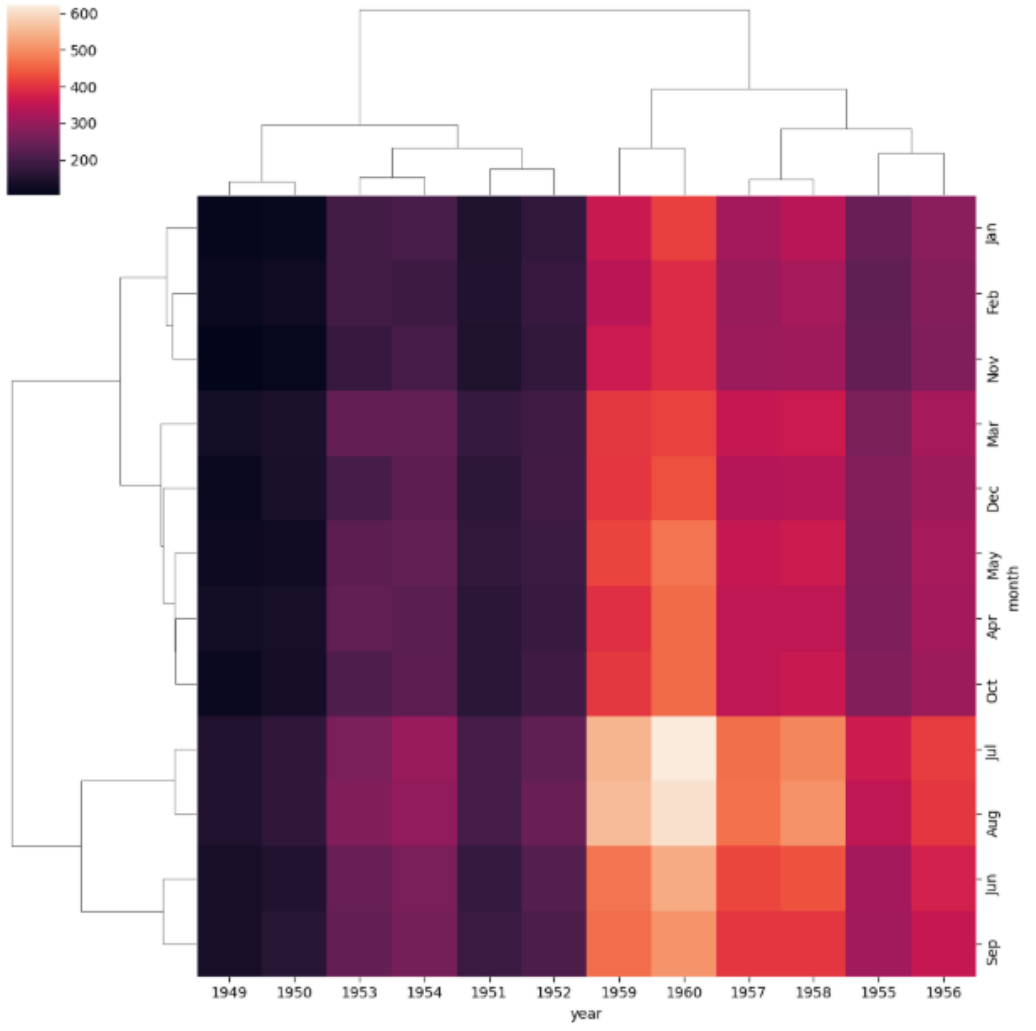
sns.clustermap(pivot_df)

- Clusters together months with similar travel patterns.
- Helps **identify correlated months or years**.
- July-August often cluster together (summer peak).
- January-February and November-December form another group (low or moderate traffic).

```
[43]: sns.clustermap(flights.pivot_table(values = 'passengers' , index = 'month' , columns = 'year'))
```

```
C:\Users\anmol sarin\AppData\Local\Temp\ipykernel_23916\3931183410.py:1: FutureWarning: The default value of observed=False is deprecated
ge to observed=True in a future version of pandas. Specify observed=False to silence this warning and retain the current behavior
  sns.clustermap(flights.pivot_table(values = 'passengers' , index = 'month' , columns = 'year'))
```

```
[43]: <seaborn.matrix.ClusterGrid at 0x139f3a1f460>
```



💡 Insights From This Analysis:

- **Trend:** Overall air travel increased significantly from 1948 to 1960.
- **Seasonality:** Summer months (Jul-Aug) show the highest number of travelers.
- **Yearly Variability:** Some years like 1960 show a clear growth peak.
- **Clustered Months:** Grouping helps understand marketing or capacity planning cycles.

🛠️ Tools & Techniques Mentioned:

Tool/Method	Purpose
groupby()	Summarize data year-wise
reset_index()	Flatten groupby output
lineplot()	Visualize trends over time
pivot_table()	Convert long data into matrix format
heatmap()	Show seasonal intensity
clustermap()	Auto-group similar months/years

🧭 Final Advice from Speaker:

- Don't just watch—**practice with real datasets**.
- Build your own **style of storytelling with data**.
- Use these visual tools to **draw actionable insights**.
- Practice turning analysis into **a to-do list** for business or decision-making.