Python

EDA with Data Visualization

Anita Mila Oktafani

DQLab Live Class Data Analyst with SQL & Python in Google Platform

TABLE OF CONTENT

The concept of data visualization and storytelling

Simplifying complex information into engaging story and presenting it visually enables decision-makers to make informed and effective decisions quickly and accurately.

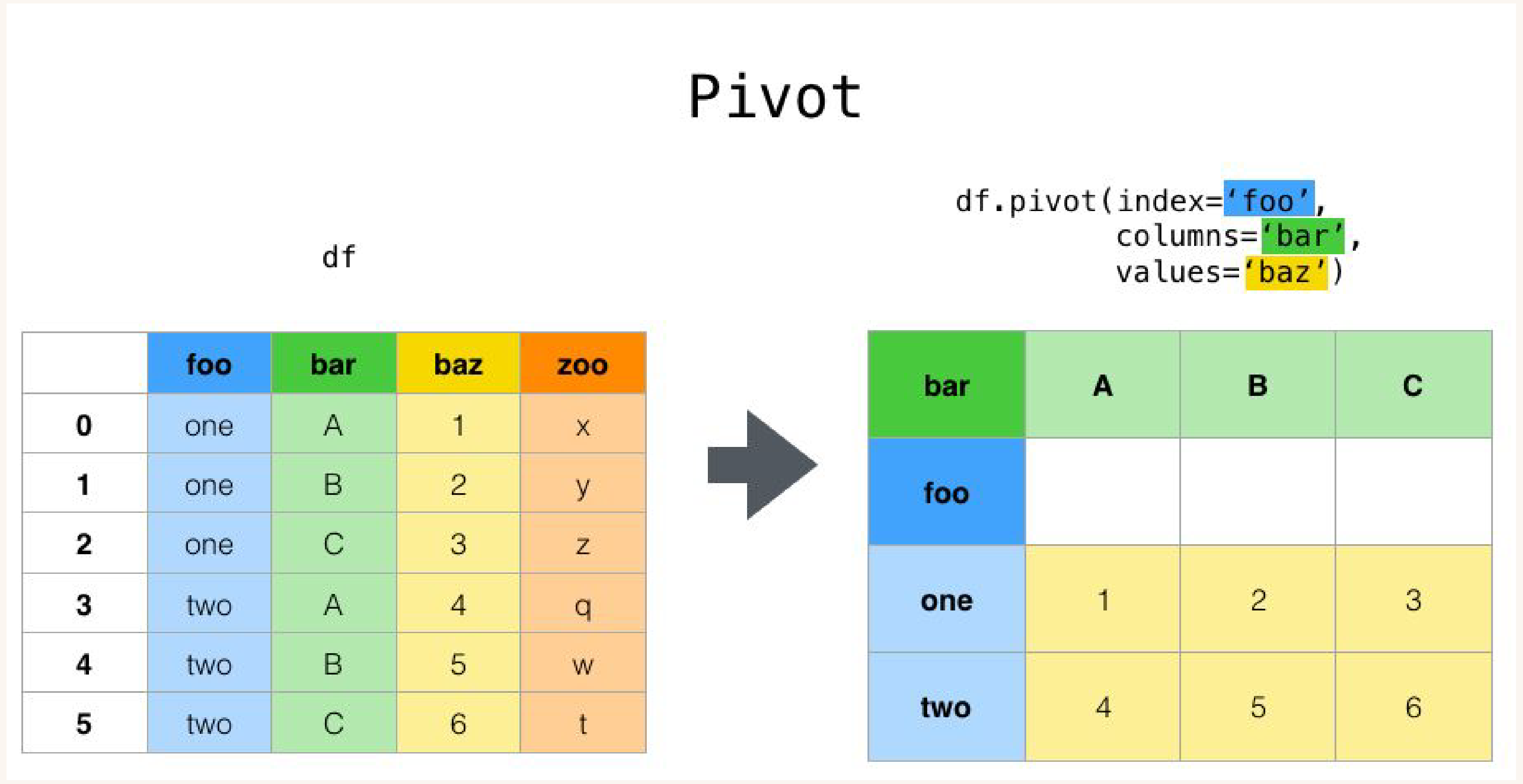
01. Simplify confusing data

02. Recognize recurring events (patterns) for use in forecasting

03. Get important information (insights)

Pivot Table

- Before creating visualizations, we must create several summaries of the data
- Summary data is used to find as much information as possible and test several hypotheses
- Summary data helps us sort out which information is important and answer problems
- A common method for creating summary data is a pivot table



Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

```
df.pivot(index='foo',
         columns='bar',
         values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

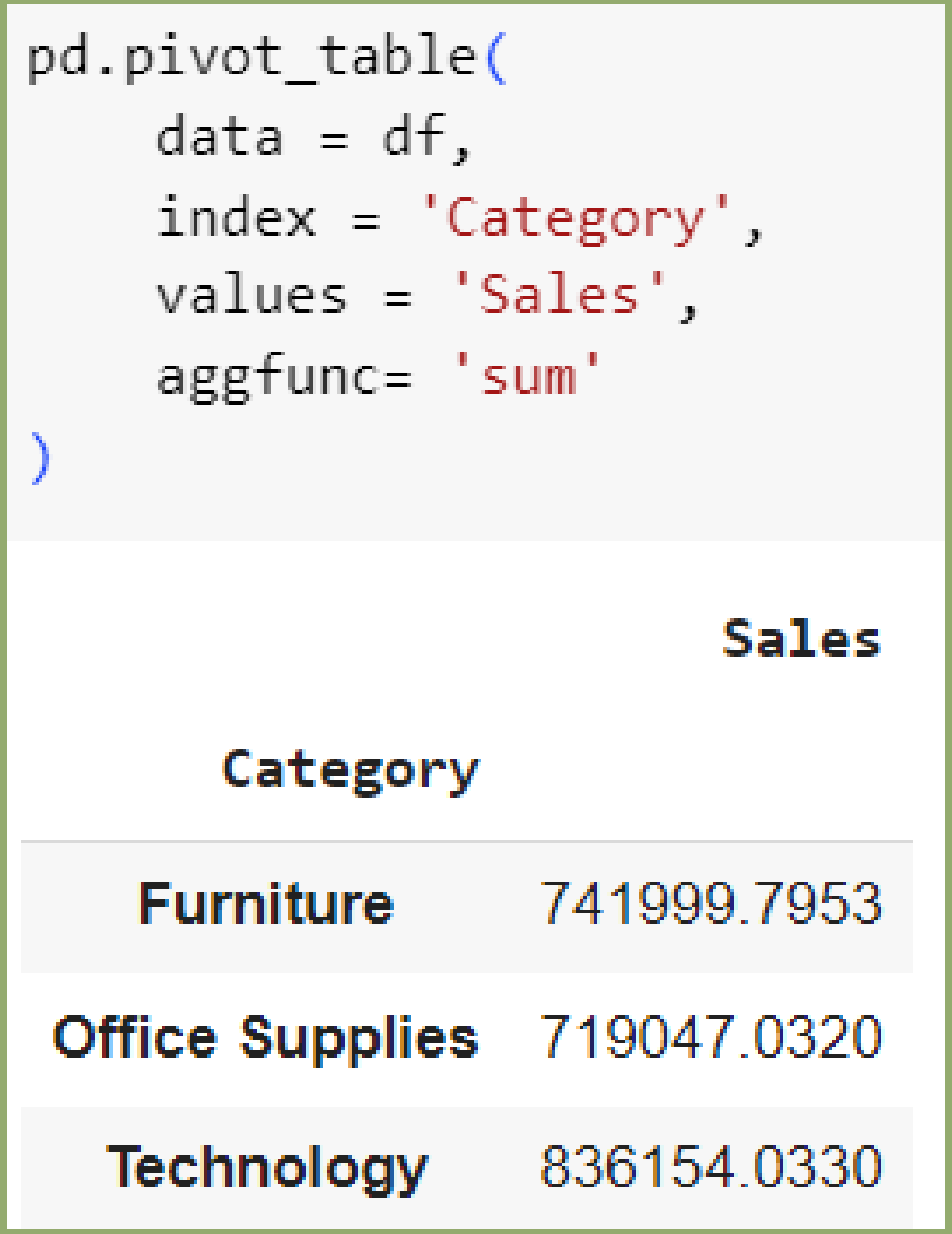
Component of Pivot Table

Component	Description
Data	Data that will be make a summary
Index/Row & Columns	Row & columns to determine how about data is displayed
Values	The values will be calculated
Aggregate Functions	Count function

```
pd.pivot_table(
        data = df,
        index = ‘column_a’,
        columns = ‘column_b’
        values = ‘ column_c’
        aggfunc = <function_name>
)
```

Aggregate function that most frequently used is **sum, mean, min, max, count**.

Example: Pivot Table (1)

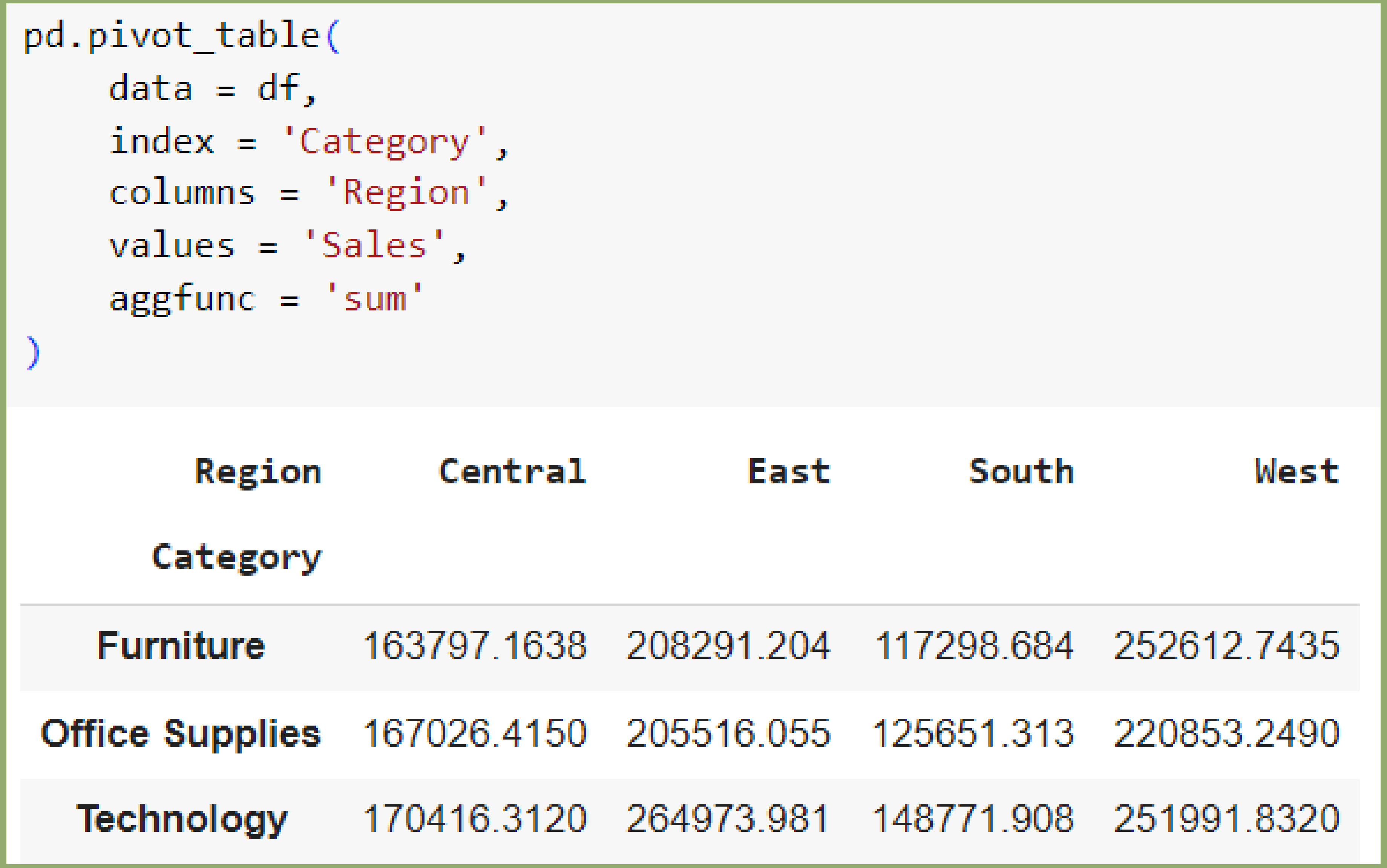


```
pd.pivot_table(
    data = df,
    index = 'Category',
    values = 'Sales',
    aggfunc= 'sum'
)
```

	Sales
Category	
Furniture	741999.7953
Office Supplies	719047.0320
Technology	836154.0330

Total income (sales) by category products sold

Example: Pivot Table (2)

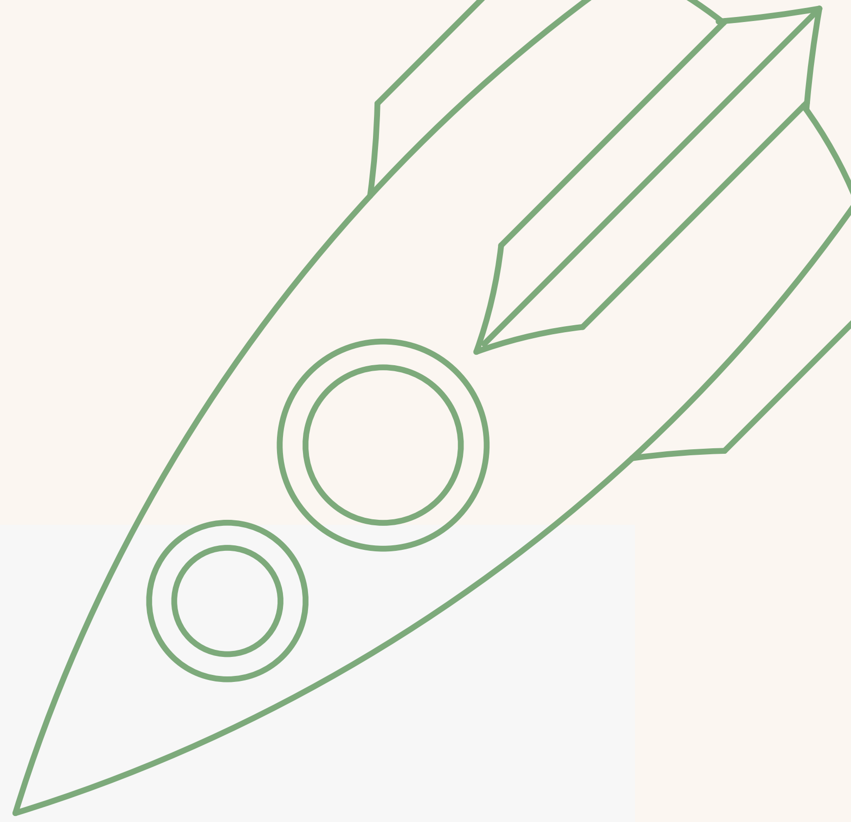


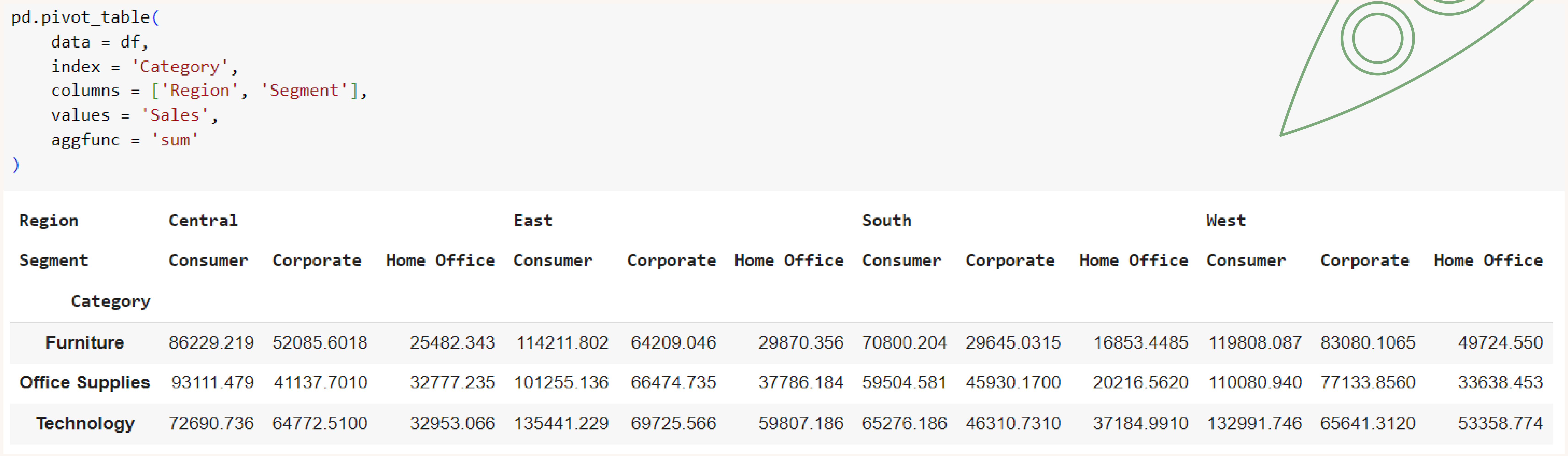
```
pd.pivot_table(
    data = df,
    index = 'Category',
    columns = 'Region',
    values = 'Sales',
    aggfunc = 'sum'
)
```

Region	Central	East	South	West
Category				
Furniture	163797.1638	208291.204	117298.684	252612.7435
Office Supplies	167026.4150	205516.055	125651.313	220853.2490
Technology	170416.3120	264973.981	148771.908	251991.8320

Total income (sales) based on product category and region

Example: Pivot Table (3)





```python
pd.pivot_table(
    data = df,
    index = 'Category',
    columns = ['Region', 'Segment'],
    values = 'Sales',
    aggfunc = 'sum'
)
```

Region	Central			East			South			West		
Segment	Consumer	Corporate	Home Office	Consumer	Corporate	Home Office	Consumer	Corporate	Home Office	Consumer	Corporate	Home Office
Category												
Furniture	86229.219	52085.6018	25482.343	114211.802	64209.046	29870.356	70800.204	29645.0315	16853.4485	119808.087	83080.1065	49724.550
Office Supplies	93111.479	41137.7010	32777.235	101255.136	66474.735	37786.184	59504.581	45930.1700	20216.5620	110080.940	77133.8560	33638.453
Technology	72690.736	64772.5100	32953.066	135441.229	69725.566	59807.186	65276.186	46310.7310	37184.9910	132991.746	65641.3120	53358.774

Total income (sales) based on product category, region and customers segmentation

Indexes or columns can be created in layers by entering variable names into the list

Data Visualization with SEABORN

- **Seaborn** is a python library that focuses on data visualization
- Seaborn is built on top of matplotlib while simplifying matplotlib syntax
- `import seaborn as sns` to use seaborn

Line Chart Single

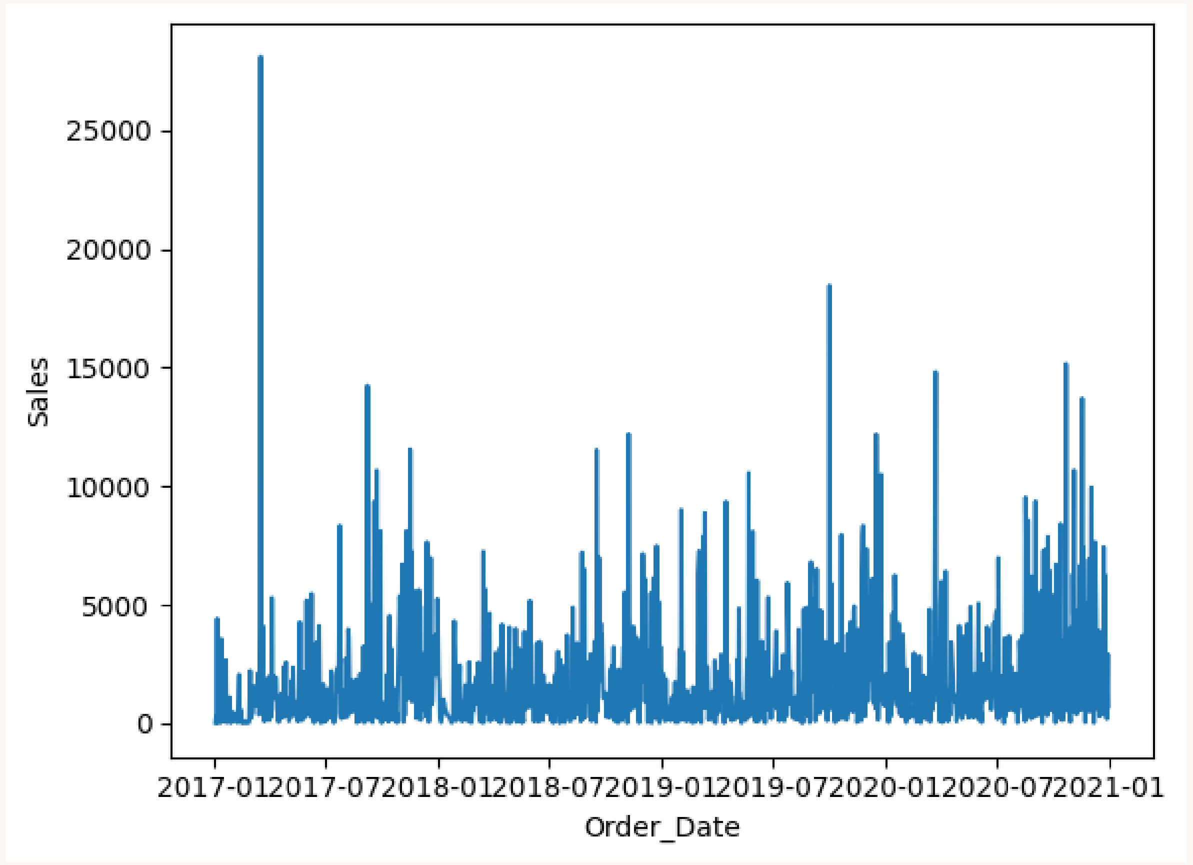
- Line charts are usually used to see trends or changes over time
- The x-axis on a line chart is usually a column with ordered data types, for example: date

```
data = pd.pivot_table(
    data = df,
    index = 'Order_Date',
    values = 'Sales',
    aggfunc = 'sum'
).reset_index()

data.head()
```

	Order_Date	Sales
0	2017-01-03	16.448
1	2017-01-04	288.060
2	2017-01-05	19.536
3	2017-01-06	4407.100
4	2017-01-07	87.158

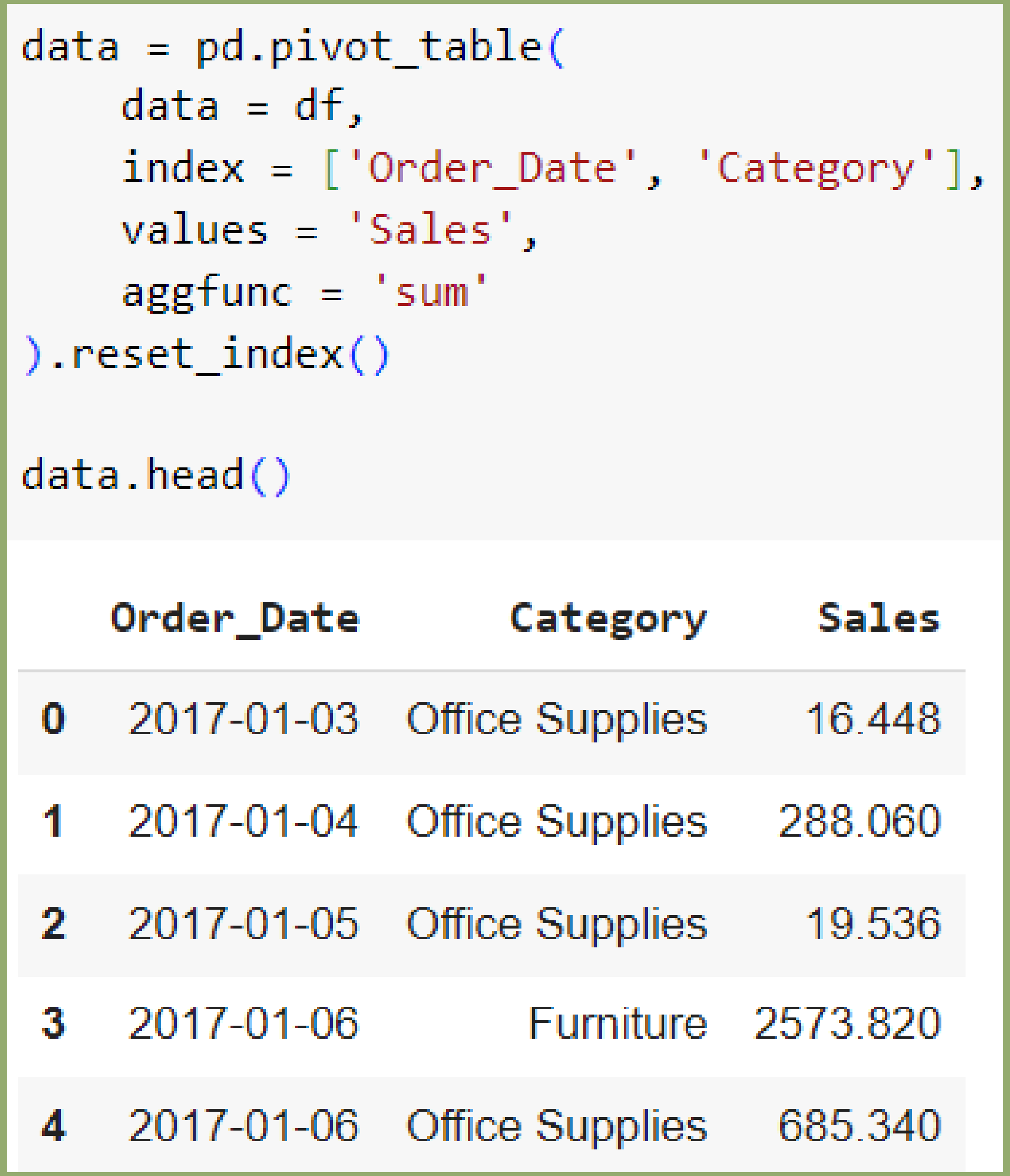
sns.lineplot(data, x_axis, y_axis)

```
sns.lineplot(
    data = data,
    x = 'Order_Date',
    y = 'Sales'
)
```



Multiple Line Chart

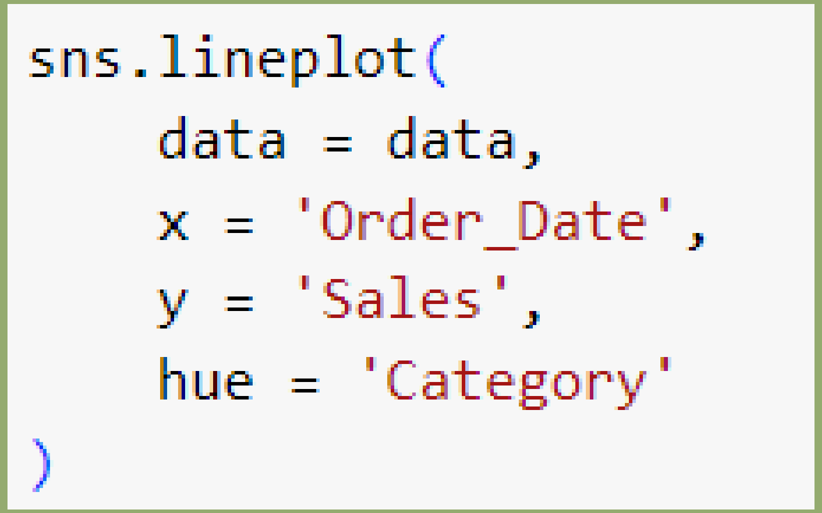
- To create multiple line charts, prepare a column that shows the division of the line chart
- Enter this column into the hue parameter in the sns.linechart function



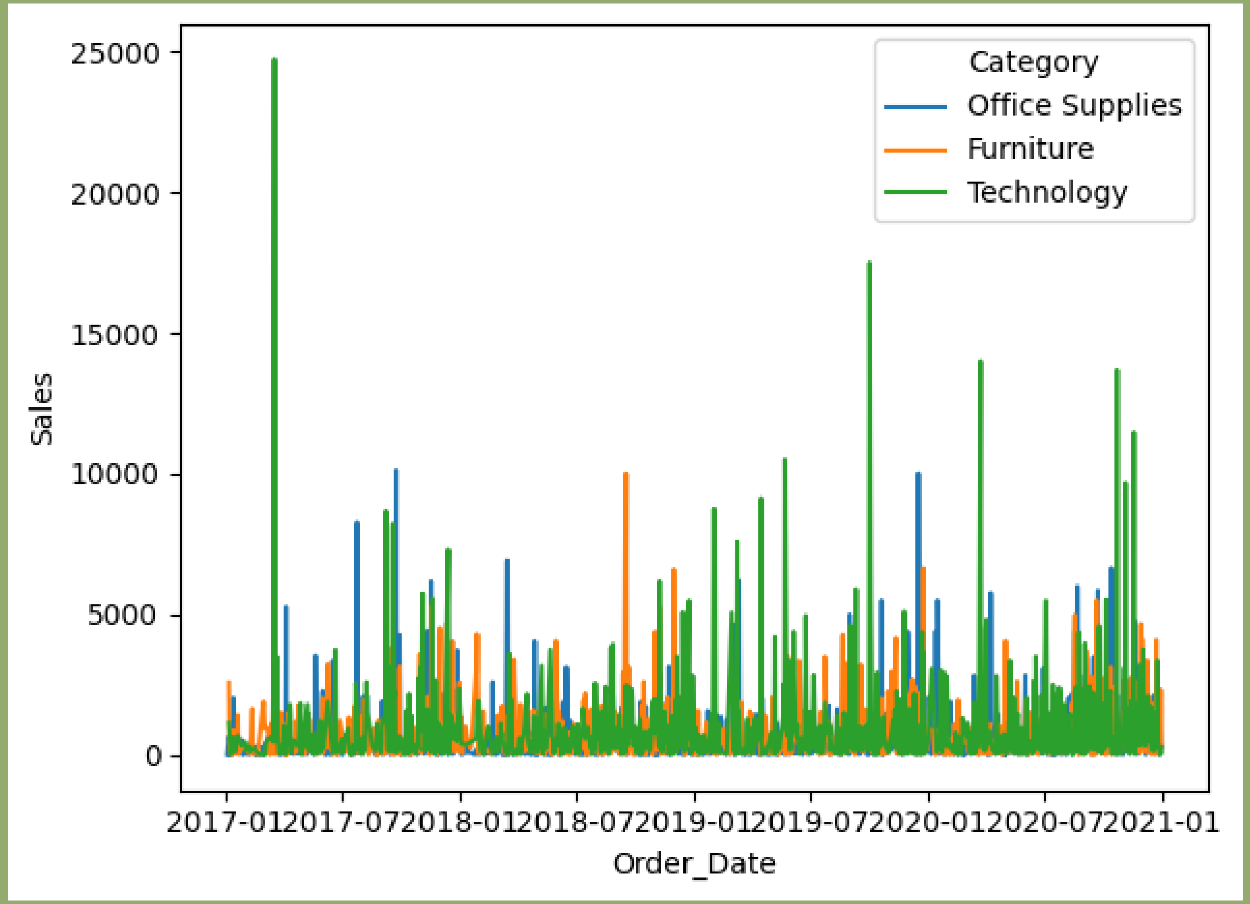
```
data = pd.pivot_table(
    data = df,
    index = ['Order_Date', 'Category'],
    values = 'Sales',
    aggfunc = 'sum'
).reset_index()

data.head()
```

	Order_Date	Category	Sales
0	2017-01-03	Office Supplies	16.448
1	2017-01-04	Office Supplies	288.060
2	2017-01-05	Office Supplies	19.536
3	2017-01-06	Furniture	2573.820
4	2017-01-06	Office Supplies	685.340

sns.lineplot(data, x, y, hue)



```
sns.lineplot(
    data = data,
    x = 'Order_Date',
    y = 'Sales',
    hue = 'Category'
)
```



Bar Chart

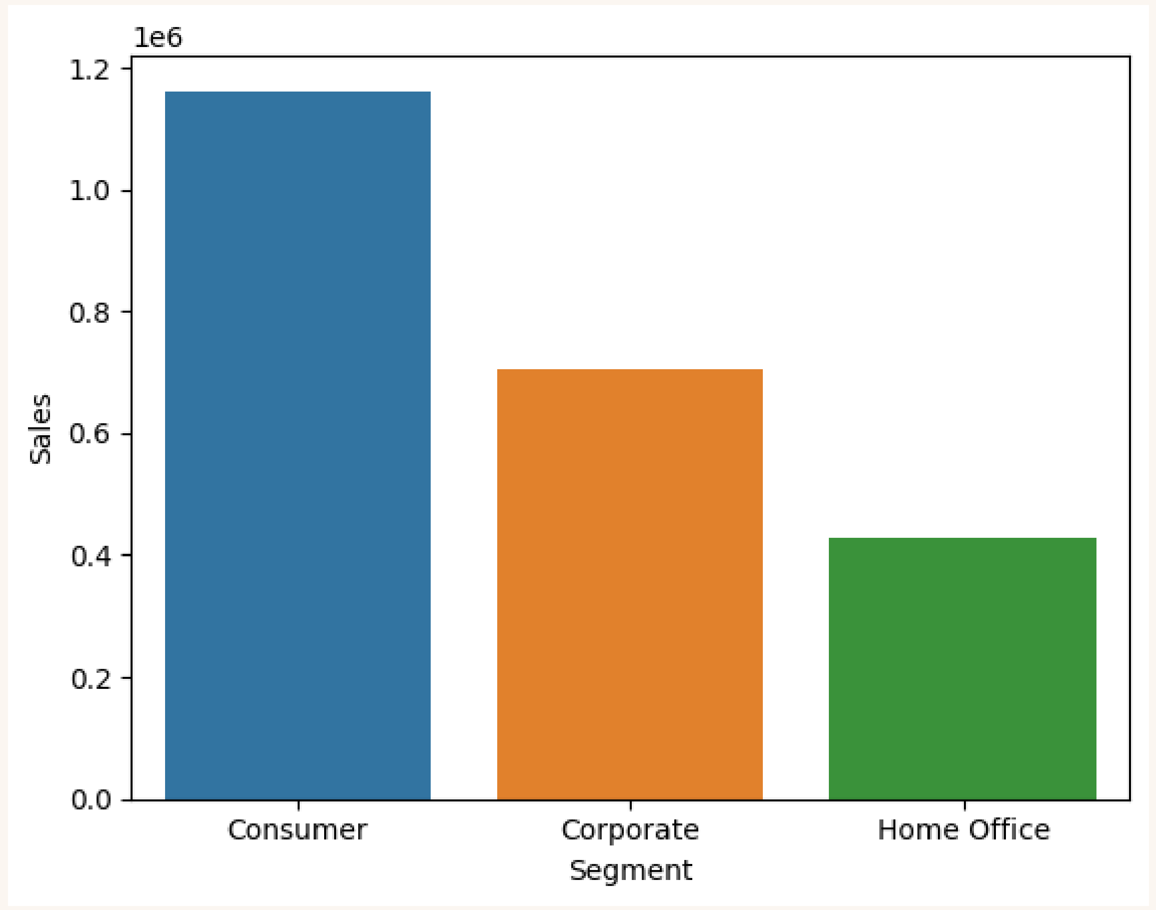
- Bar charts are used to compare values between variables
- The x-axis on a bar chart does not need to be an ordered variable

```
data = pd.pivot_table(
    data = df,
    index = 'Segment',
    values = 'Sales',
    aggfunc = 'sum'
).reset_index()

data.head()
```

	Segment	Sales
0	Consumer	1.161401e+06
1	Corporate	7.061464e+05
2	Home Office	4.296531e+05

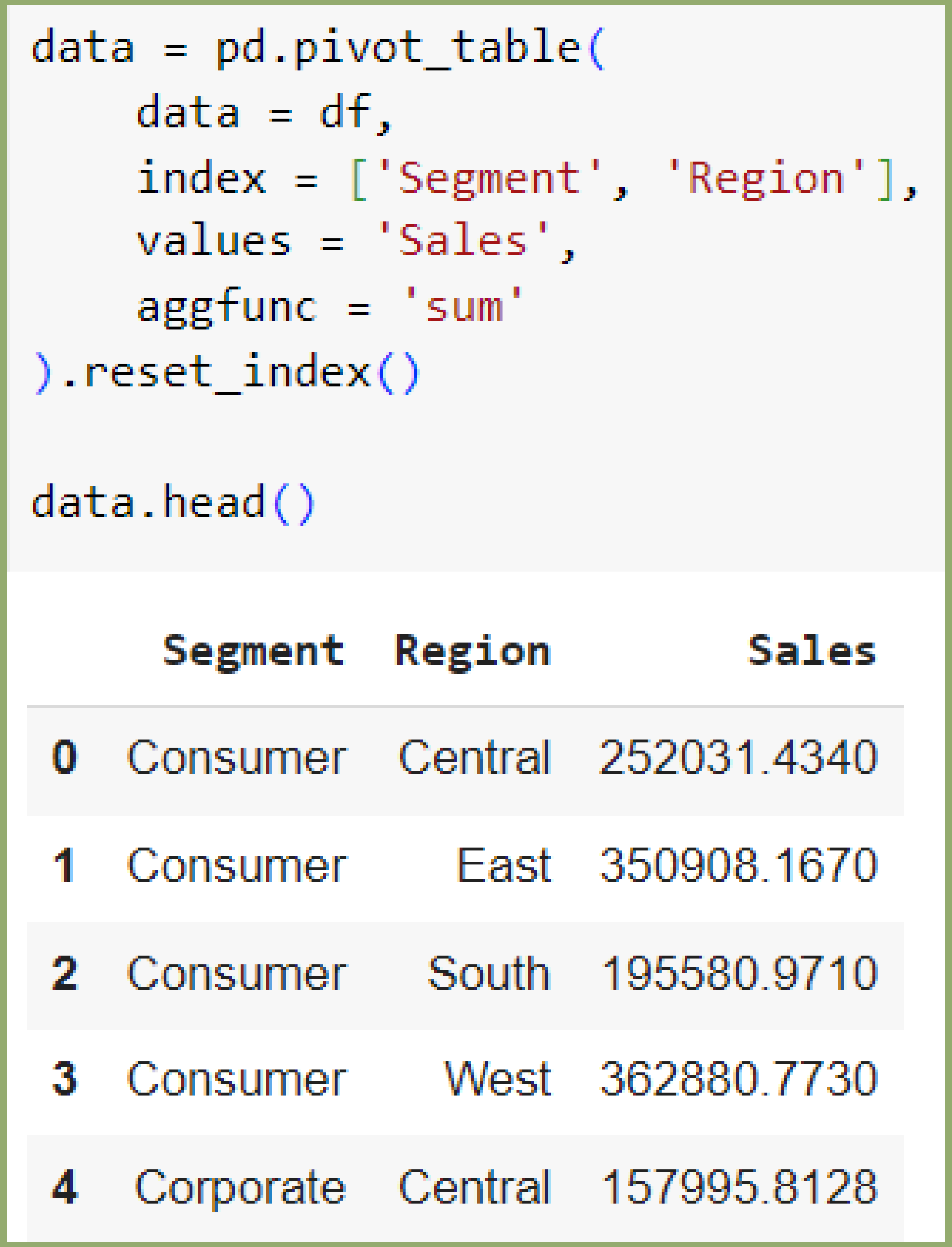
sns.barplot(data, x_axis, y_axis)

```
sns.barplot(data=data, x='Segment', y='Sales')
```



Cluster Bar Chart

- Apart from making a simple barchart, we can also make a breakdown of the barchart into its components
- Enter the variables that will be the components into the hue parameter

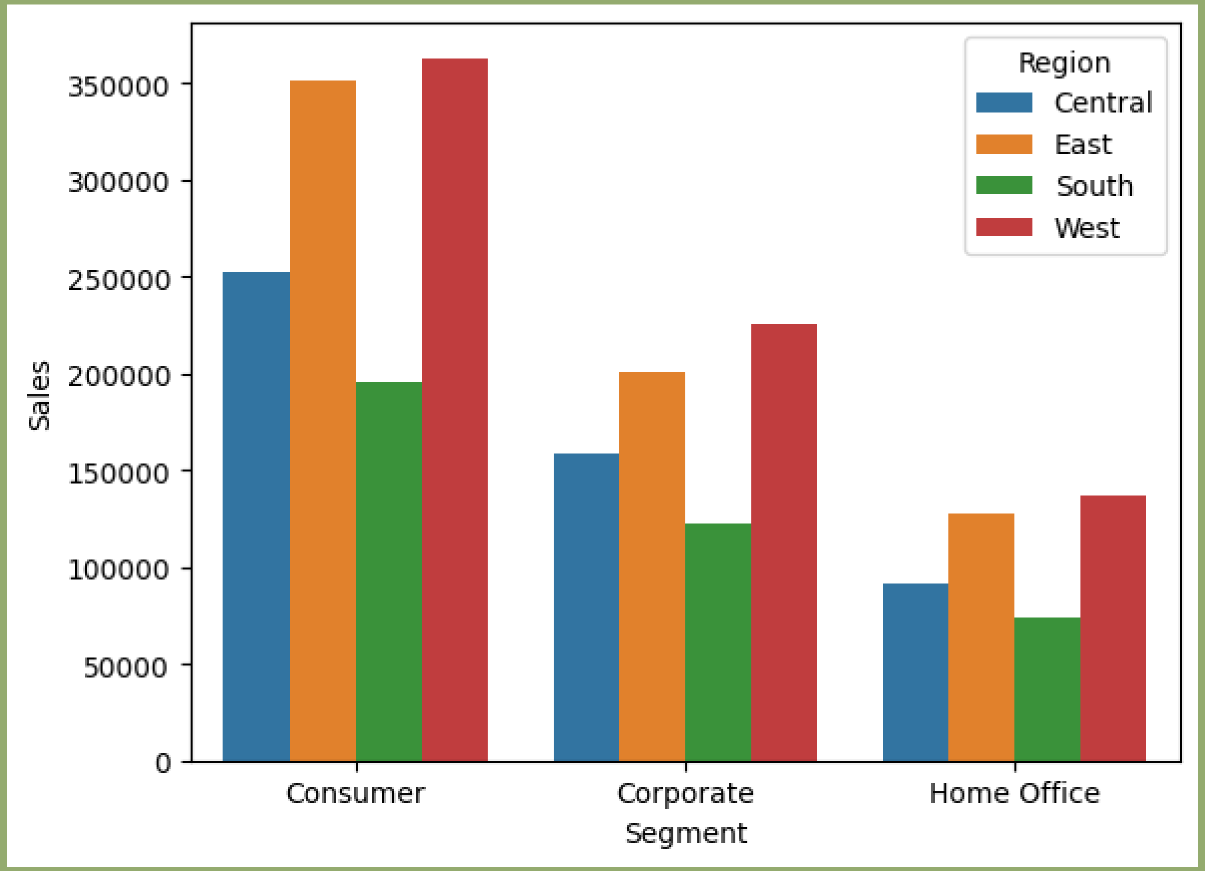


```
data = pd.pivot_table(
    data = df,
    index = ['Segment', 'Region'],
    values = 'Sales',
    aggfunc = 'sum'
).reset_index()

data.head()
```

	Segment	Region	Sales
0	Consumer	Central	252031.4340
1	Consumer	East	350908.1670
2	Consumer	South	195580.9710
3	Consumer	West	362880.7730
4	Corporate	Central	157995.8128

sns.lineplot(data, x, y, hue)

```
sns.barplot(data=data, x='Segment', y='Sales', hue='Region')
```

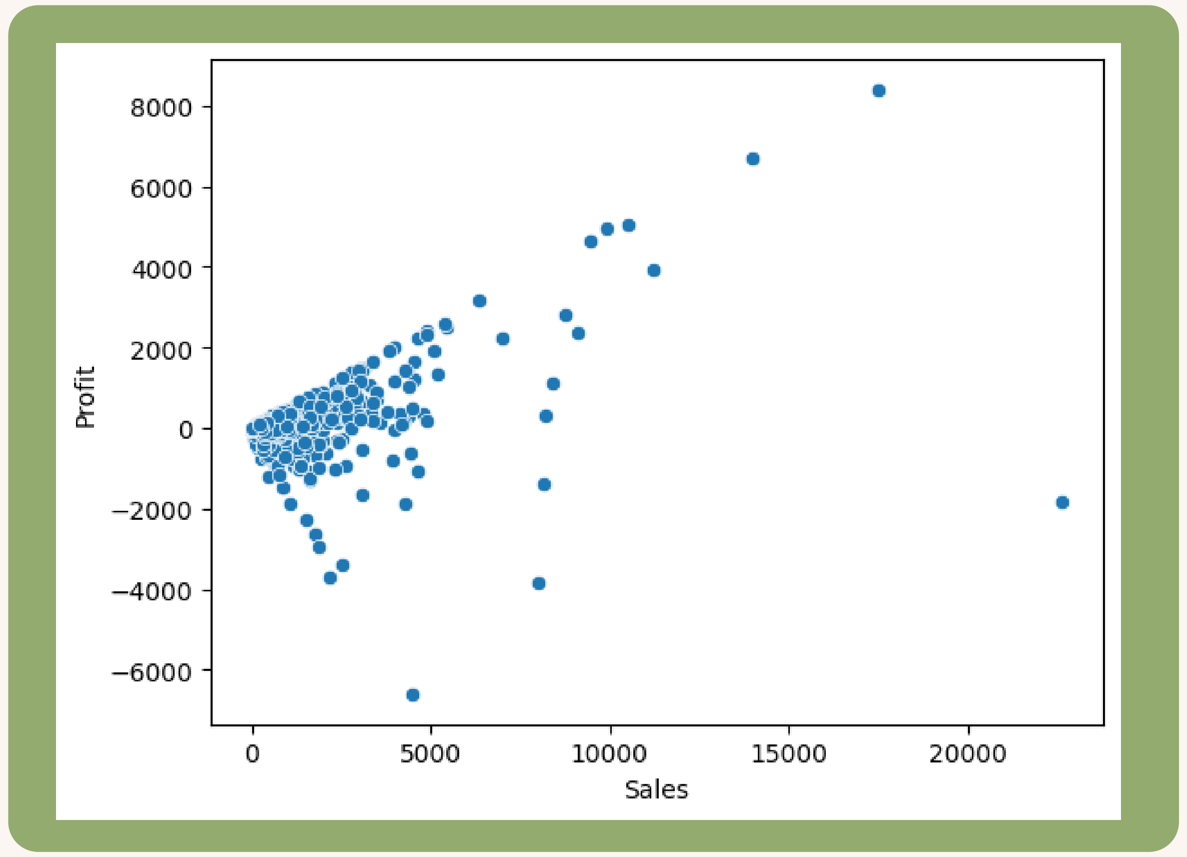


Scatterplot

Scatterplot is used to see the correlation or relationship between two numerical variables

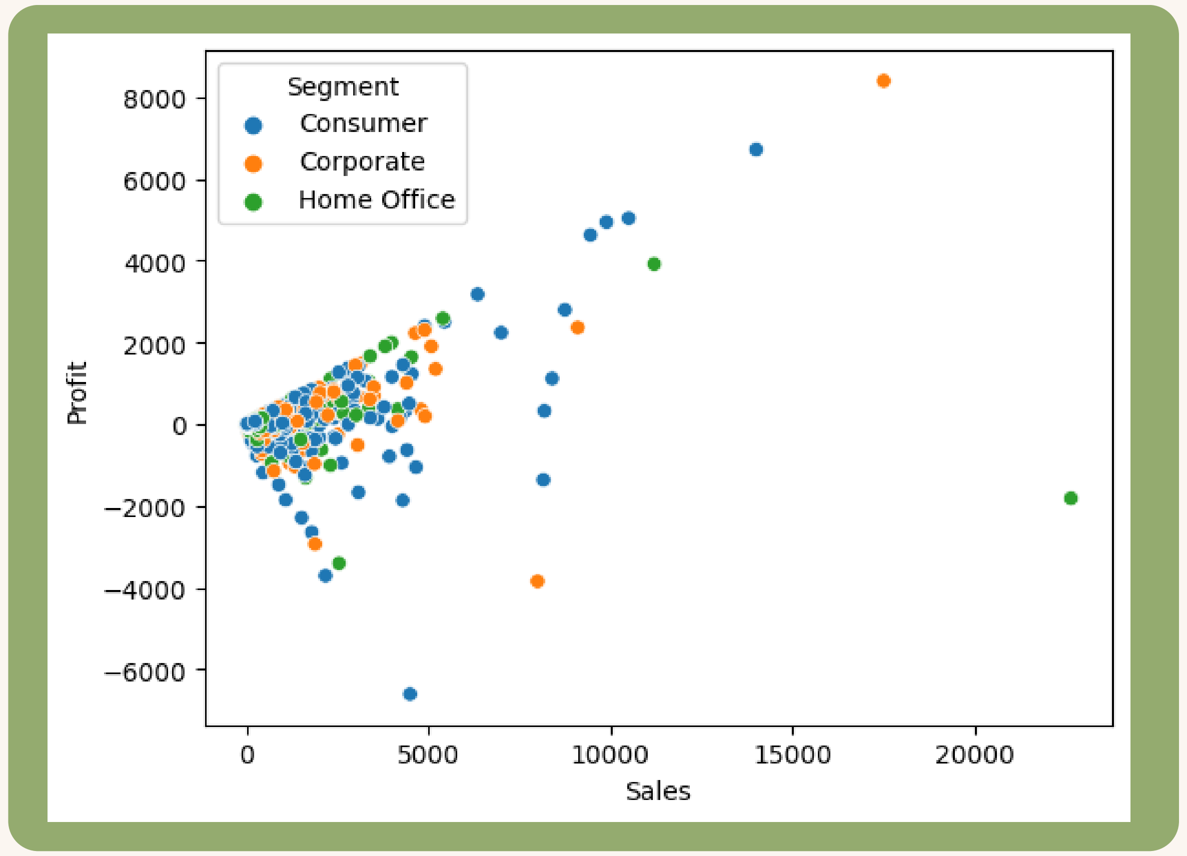
sns.scatterplot(data, x_axis, y_axis)

```
sns.scatterplot(data=df, x='Sales', y='Profit')
```



sns.scatterplot(data, x, y, hue)

```
sns.scatterplot(data=df, x='Sales', y='Profit', hue='Segment')
```

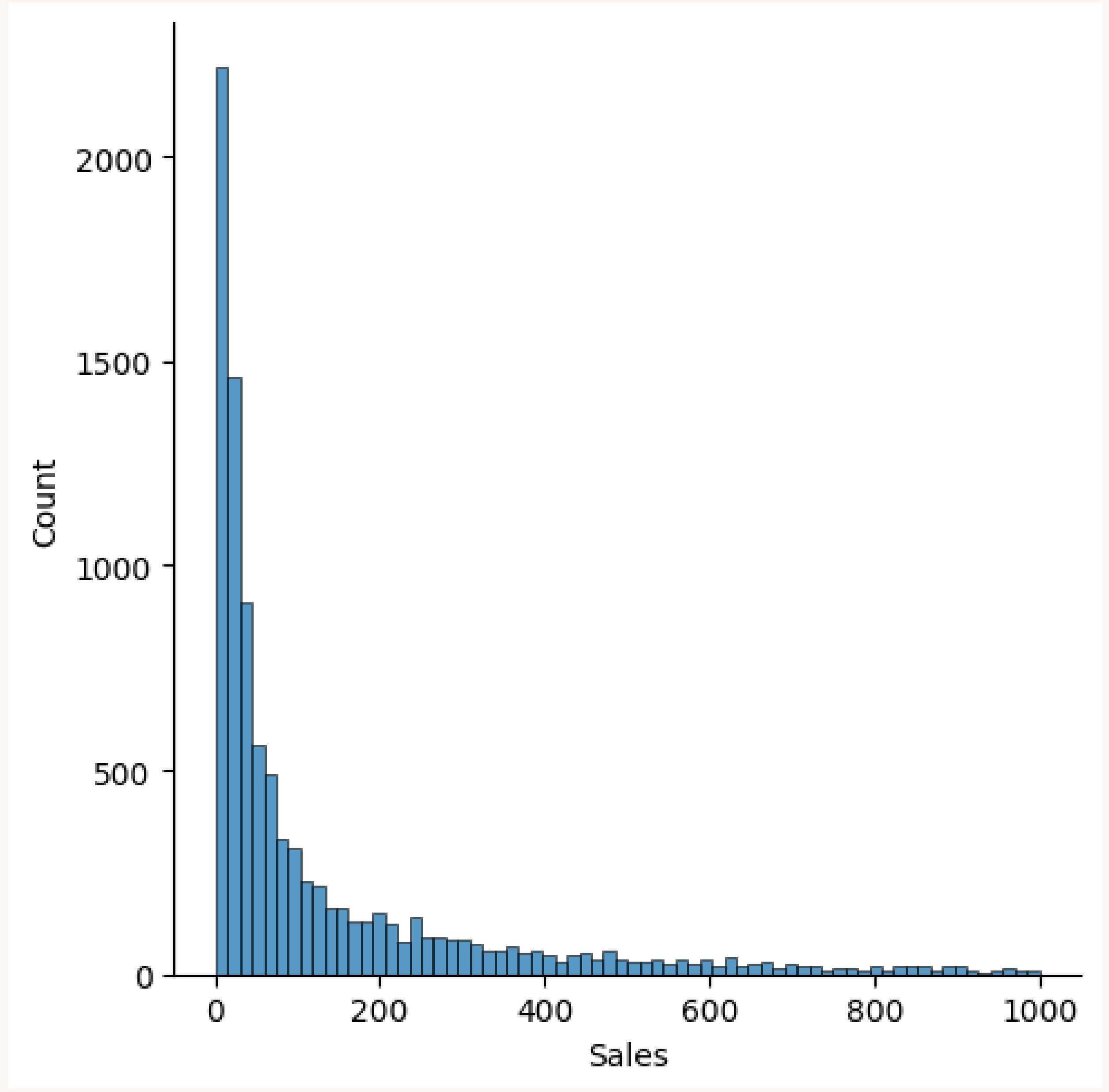


Displot

- Displot is used to display the distribution of a numerical series
- By default the plot will display a histogram

sns.displot(data [series])

```
sns.displot(df[df['Sales'] < 1000]['Sales'])
```



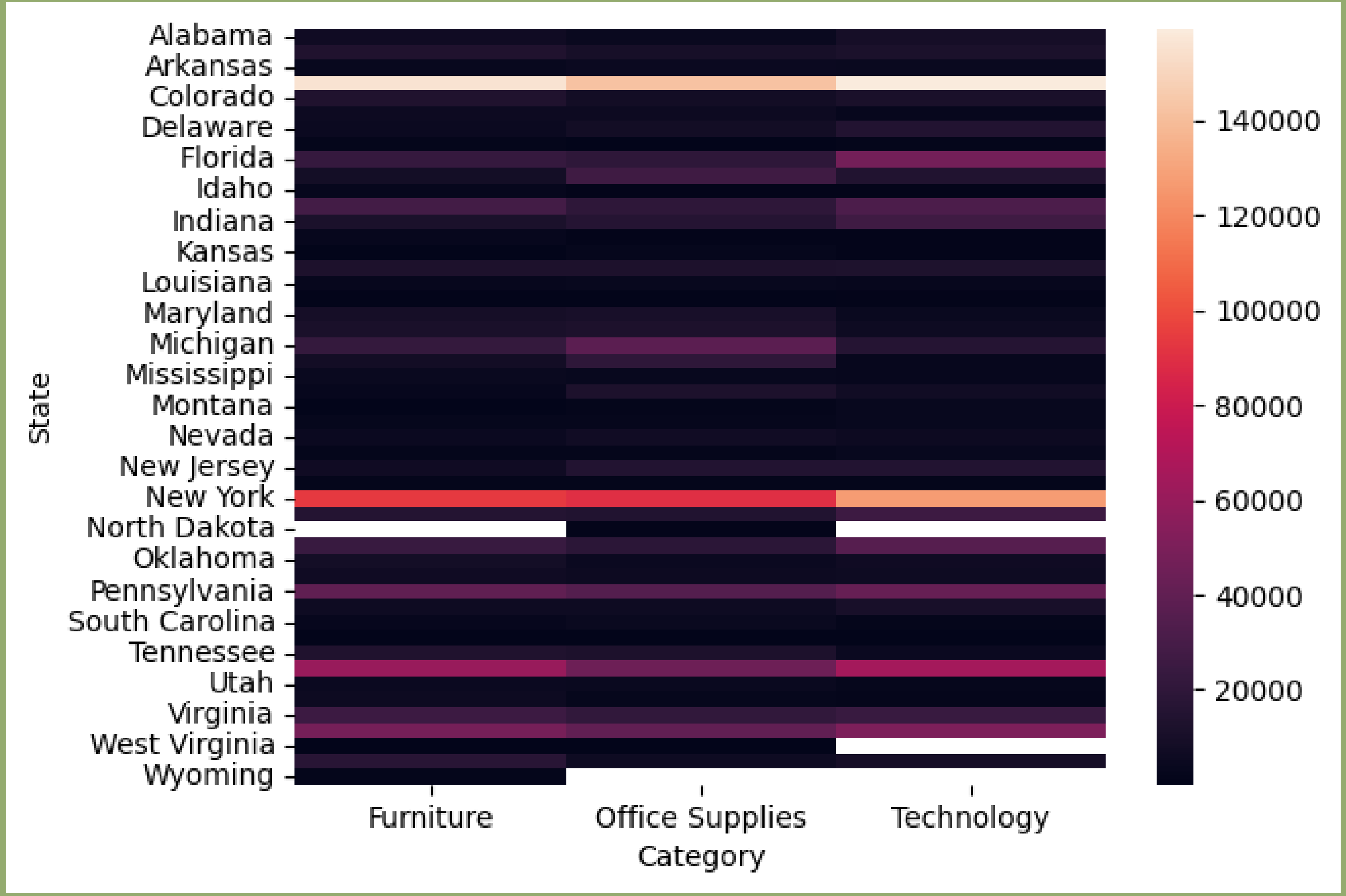
Heatmap

Heatmap makes it easier to read tables by assigning colors to cells based on the value in that cell

```
data = pd.pivot_table(
    data = df,
    index = 'State',
    columns = 'Category',
    values = 'Sales',
    aggfunc = 'sum'
)
```

```
sns.heatmap(data)
```

sns.heatmap(table)



Thank you!

Linkedin:
https://www.linkedin.com/in/anitamilaoktafani/