😊 Pivot Tables

What Are Pivot Tables in Pandas?

A **pivot table** in Pandas is a data processing tool that allows you to summarize, aggregate, and rearrange data in a tabular format. It is used to transform long-format data into a more organized structure for analysis. Specifically, pivot tables let you group data by one or more keys and perform aggregate functions such as calculating averages, sums, counts, or other statistics across these groupings.

Pivot tables are built using the pivot_table() function in Pandas, which offers a flexible way to manipulate data, similar to pivot tables in spreadsheet software like Microsoft Excel.

Key Concepts

1. **Index**: The column(s) to group by rows.
2. **Columns**: The column(s) to group by columns.
3. **Values**: The column(s) containing the values to aggregate.
4. **Aggregation Function (aggfunc)**: The function used to perform a summary operation such as mean, sum, count, etc.

Use Cases

1. **Summarizing Sales Data**: Pivot tables are often used in business analytics to summarize sales by product, region, or salesperson. For instance, a sales dataset can be pivoted to show total sales per region per month.
2. **Employee Performance or Salary Reports**: In HR analytics, pivot tables can show the average salary by department, or the number of employees per job role.
3. **Customer Behavior Analysis**: Marketers might use pivot tables to track customer purchases across different categories, time periods, or demographics.
4. **Inventory Management**: In supply chain management, pivot tables help analyze stock levels by product type and warehouse location.
5. **Web or App Usage Statistics**: Data analysts can use pivot tables to measure user engagement metrics by platform, region, or date.

Benefits of Using Pivot Tables in Pandas

1. **Efficient Data Summarization**: Pivot tables allow you to quickly summarize large datasets without needing to write complex code or perform manual calculations.
2. **Improved Readability**: By reshaping and organizing data, pivot tables help transform raw data into a format that is easier to read and understand.
3. **Flexibility and Customization**: You can define custom aggregation functions and combine multiple dimensions (rows and columns) to explore the data from various perspectives.

4. **Time-Saving for Exploratory Analysis**: Pivot tables are ideal for quick, preliminary analysis. They help analysts find trends, outliers, or patterns without needing to prepare formal reports.
5. **Integrates Seamlessly with Pandas Workflows**: Since pivot tables are a native Pandas feature, they work well with other data manipulation tools such as filtering, merging, and time series analysis.

When to Use Pivot Tables Instead of Other Methods

- Use pivot tables when you want to reshape data to show summary metrics across multiple dimensions.
- Use them instead of grouping and manually aggregating if you prefer a tabular output.
- Avoid pivot tables if your data is too sparse or if you need more complex transformations that go beyond two dimensions.

😄 To begin with the Lab

1. We start by importing NumPy and Pandas in our notebook, then create a DataFrame to read the CSV file and display a few of the entries from the CSV.

```
[2]: import numpy as np
     import pandas as pd
```

```
[4]: df = pd.read_csv('Sales_Funnel_CRM.csv')
```

```
[6]: df
```

[6]:

	Account Number	Company	Contact	Account Manager	Product	Licenses	Sale Price	Status
0	2123398	Google	Larry Pager	Edward Thorp	Analytics	150	2100000	Presented
1	2123398	Google	Larry Pager	Edward Thorp	Prediction	150	700000	Presented
2	2123398	Google	Larry Pager	Edward Thorp	Tracking	300	350000	Under Review
3	2192650	BOBO	Larry Pager	Edward Thorp	Analytics	150	2450000	Lost
4	420496	IKEA	Elon Tusk	Edward Thorp	Analytics	300	4550000	Won
5	636685	Tesla Inc.	Elon Tusk	Edward Thorp	Analytics	300	2800000	Under Review
6	636685	Tesla Inc.	Elon Tusk	Edward Thorp	Prediction	150	700000	Presented
7	1216870	Microsoft	Will Grates	Edward Thorp	Tracking	300	350000	Under Review
8	2200450	Walmart	Will Grates	Edward Thorp	Analytics	150	2450000	Lost

2. Using the help function, we can look into what Pivot is capable of in Pandas.

```
[9]: help(pd.pivot)
```

```
Help on function pivot in module pandas.core.reshape.pivot:

pivot(data: 'DataFrame', *, columns: 'IndexLabel', index: 'IndexLabel | lib.NoDefault' = typing.Literal[<no_default>], values: 'IndexLabel | lib.NoDefa
ult' = typing.Literal[<no_default>]) -> 'DataFrame'
    Return reshaped DataFrame organized by given index / column values.

    Reshape data (produce a "pivot" table) based on column values. Uses
    unique values from specified `index` / `columns` to form axes of the
    resulting DataFrame. This function does not support data
    aggregation, multiple values will result in a MultiIndex in the
    columns. See the :ref:`User Guide <reshaping>` for more on reshaping.

    Parameters
    ----------
    data : DataFrame
    columns : str or object or a list of str
        Column to use to make new frame's columns.
```

3. The code selects only the 'Company', 'Product', and 'Licenses' columns from the original DataFrame to avoid errors from duplicate rows, then creates a pivot table that rearranges the data so each row represents a company, each column represents a product, and the values show the number of licenses, effectively summarizing how many licenses each company holds for each product.

Here we have used two main methods or functions:

- `df[['Company','Product','Licenses']]` – This is **column selection**, a method of creating a new DataFrame by extracting specific columns from the original DataFrame.
- `pd.pivot()` – This is the **pivot function** from the Pandas library, used to reshape the data by specifying an index (rows), columns, and the values to populate the table.

[15]:
```
# Let's take a subset, otherwise we'll get an error due to duplicate rows and data
licenses = df[['Company','Product','Licenses']]
licenses
```

[15]:

	Company	Product	Licenses
0	Google	Analytics	150
1	Google	Prediction	150
2	Google	Tracking	300
3	BOBO	Analytics	150
4	IKEA	Analytics	300
5	Tesla Inc.	Analytics	300
6	Tesla Inc.	Prediction	150
7	Microsoft	Tracking	300
8	Walmart	Analytics	150

[17]:
```
pd.pivot(data=licenses,index='Company',columns='Product',values='Licenses')
```

[17]:

Product	Analytics	GPS Positioning	Prediction	Tracking
Company				
Google	150.0	NaN	150.0	300.0
ATT	NaN	NaN	150.0	150.0
Apple	300.0	NaN	NaN	NaN
BOBO	150.0	NaN	NaN	NaN
CVS Health	NaN	NaN	NaN	450.0
Cisco	300.0	300.0	NaN	NaN
Exxon Mobile	150.0	NaN	NaN	NaN
IKEA	300.0	NaN	NaN	NaN

4. The below cell creates a pivot table grouped by company and sums all numeric columns, though summing certain fields like account numbers may not make logical sense.

```
[22]: # Notice Account Number sum() doesn't make sense to keep/use
      pd.pivot_table(df,index="Company",aggfunc='sum')
```

[22]:

Company	Account Manager	Account Number	Contact	Licenses	Product	Sale Price	Status
Google	Edward ThorpEdward ThorpEdward Thorp	6370194	Larry PagerLarry PagerLarry Pager	600	AnalyticsPredictionTracking	3150000	PresentedPresentedUnder Review
ATT	Claude ShannonClaude Shannon	1396064	Cindy PhonerCindy Phoner	300	TrackingPrediction	1050000	Under ReviewPresented
Apple	Claude Shannon	405886	Cindy Phoner	300	Analytics	4550000	Won
BOBO	Edward Thorp	2192650	Larry Pager	150	Analytics	2450000	Lost
CVS Health	Claude Shannon	902797	Emma Gordian	450	Tracking	490000	Won
Cisco	Claude ShannonClaude Shannon	4338998	Emma GordianEmma Gordian	600	AnalyticsGPS Positioning	4900000	LostPresented
Exxon Mobile	Claude Shannon	470248	Cindy Phoner	150	Analytics	2100000	Presented
IKEA	Edward Thorp	420496	Elon Tusk	300	Analytics	4550000	Won
Microsoft	Edward Thorp	1216870	Will Grates	300	Tracking	350000	Under Review
Salesforce	Claude Shannon	2046943	Emma Gordian	750	Analytics	7000000	Won
Tesla Inc.	Edward ThorpEdward Thorp	1273370	Elon TuskElon Tusk	450	AnalyticsPrediction	3500000	Under ReviewPresented
Walmart	Edward Thorp	2200450	Will Grates	150	Analytics	2450000	Lost

5. It selects only the 'Licenses' and 'Sale Price' columns from the previous pivot table, removing unrelated or non-meaningful numeric columns like account numbers.

```
[24]: # Either grab the columns
      pd.pivot_table(df,index="Company",aggfunc='sum')[['Licenses','Sale Price']]
```

[24]:

Company	Licenses	Sale Price
Google	600	3150000
ATT	300	1050000
Apple	300	4550000
BOBO	150	2450000
CVS Health	450	490000
Cisco	600	4900000
Exxon Mobile	150	2100000
IKEA	300	4550000
Microsoft	300	350000
Salesforce	750	7000000
Tesla Inc.	450	3500000
Walmart	150	2450000

6. It does the same as the previous cell but selects the desired columns ('Licenses' and 'Sale Price') directly within the pivot function.

[26]:
```
# Or state them as wanted values
pd.pivot_table(df,index="Company",aggfunc='sum',values=['Licenses','Sale Price'])
```

[26]:

Company	Licenses	Sale Price
Google	600	3150000
ATT	300	1050000
Apple	300	4550000
BOBO	150	2450000
CVS Health	450	490000
Cisco	600	4900000
Exxon Mobile	150	2100000
IKEA	300	4550000
Microsoft	300	350000
Salesforce	750	7000000
Tesla Inc.	450	3500000
Walmart	150	2450000

7. The cell below groups data by company and sums values, similar to the pivot table but using groupby() instead, followed by selecting the relevant columns.

[28]:
```
df.groupby('Company').sum()[['Licenses','Sale Price']]
```

[28]:

Company	Licenses	Sale Price
Google	600	3150000
ATT	300	1050000
Apple	300	4550000
BOBO	150	2450000
CVS Health	450	490000
Cisco	600	4900000
Exxon Mobile	150	2100000
IKEA	300	4550000
Microsoft	300	350000
Salesforce	750	7000000
Tesla Inc.	450	3500000
Walmart	150	2450000

8. Creates a pivot table grouped by account manager and contact, showing total sale price per contact-person pair.

```
[30]: pd.pivot_table(df,index=["Account Manager","Contact"],values=['Sale Price'],aggfunc='sum')
```

[30]:

		Sale Price
Account Manager	**Contact**	
Claude Shannon	**Cindy Phoner**	7700000
	Emma Gordian	12390000
Edward Thorp	**Elon Tusk**	8050000
	Larry Pager	5600000
	Will Grates	2800000

9. Builds a pivot table grouped by account manager and contact, with products as columns and summed sale prices as the values.

```
[33]: pd.pivot_table(df,index=["Account Manager","Contact"],values=["Sale Price"],columns=["Product"],aggfunc=[np.sum])
```

[33]:

		sum			
		Sale Price			
	Product	**Analytics**	**GPS Positioning**	**Prediction**	**Tracking**
Account Manager	**Contact**				
Claude Shannon	**Cindy Phoner**	6650000.0	NaN	700000.0	350000.0
	Emma Gordian	11550000.0	350000.0	NaN	490000.0
Edward Thorp	**Elon Tusk**	7350000.0	NaN	700000.0	NaN
	Larry Pager	4550000.0	NaN	700000.0	350000.0
	Will Grates	2450000.0	NaN	NaN	350000.0

10. Same as above, but fills missing values (e.g., when a contact didn't buy a product) with zero instead of leaving them blank.

```
[35]: pd.pivot_table(df,index=["Account Manager","Contact"],values=["Sale Price"],columns=["Product"],aggfunc=[np.sum],fill_value=0)
```

[35]:

		sum			
		Sale Price			
	Product	**Analytics**	**GPS Positioning**	**Prediction**	**Tracking**
Account Manager	**Contact**				
Claude Shannon	**Cindy Phoner**	6650000	0	700000	350000
	Emma Gordian	11550000	350000	0	490000
Edward Thorp	**Elon Tusk**	7350000	0	700000	0
	Larry Pager	4550000	0	700000	350000
	Will Grates	2450000	0	0	350000

11. Adds both sum and average calculations for sale price per product, per contact, filling in any missing values with zero.

[37]:
```
# Can add multiple agg functions
pd.pivot_table(df,index=["Account Manager","Contact"],values=["Sale Price"],columns=["Product"],
               aggfunc=[np.sum,np.mean],fill_value=0)
```

[37]:

		sum				mean			
		Sale Price				Sale Price			
	Product	Analytics	GPS Positioning	Prediction	Tracking	Analytics	GPS Positioning	Prediction	Tracking
Account Manager	Contact								
Claude Shannon	Cindy Phoner	6650000	0	700000	350000	3325000	0	700000	350000
	Emma Gordian	11550000	350000	0	490000	5775000	350000	0	490000
Edward Thorp	Elon Tusk	7350000	0	700000	0	3675000	0	700000	0
	Larry Pager	4550000	0	700000	350000	2275000	0	700000	350000
	Will Grates	2450000	0	0	350000	2450000	0	0	350000

12. Adds license data along with sale price, showing total values per product per contact, grouped by account manager, filling blanks with zero.

[39]:
```
# Can add on multiple columns
pd.pivot_table(df,index=["Account Manager","Contact"],values=["Sale Price","Licenses"],columns=["Product"],
               aggfunc=[np.sum],fill_value=0)
```

[39]:

						sum			
						Sale Price			
	Product	Analytics	GPS Positioning	Prediction	Tracking	Analytics	GPS Positioning	Prediction	Tracking
Account Manager	Contact								
Claude Shannon	Cindy Phoner	450	0	150	150	6650000	0	700000	350000
	Emma Gordian	1050	300	0	450	11550000	350000	0	490000
Edward Thorp	Elon Tusk	600	0	150	0	7350000	0	700000	0
	Larry Pager	300	0	150	300	4550000	0	700000	350000
	Will Grates	150	0	0	300	2450000	0	0	350000

13. Groups by account manager, contact, and product, then calculates total sale price and licenses for each product-contact pair.

[41]:
```
# Can add on multiple columns
pd.pivot_table(df,index=["Account Manager","Contact","Product"],values=["Sale Price","Licenses"],
               aggfunc=[np.sum],fill_value=0)
```

[41]:

			sum	
			Licenses	Sale Price
Account Manager	Contact	Product		
Claude Shannon	Cindy Phoner	Analytics	450	6650000
		Prediction	150	700000
		Tracking	150	350000
	Emma Gordian	Analytics	1050	11550000
		GPS Positioning	300	350000
		Tracking	450	490000
Edward Thorp	Elon Tusk	Analytics	600	7350000
		Prediction	150	700000
	Larry Pager	Analytics	300	4550000
		Prediction	150	700000
		Tracking	300	350000
	Will Grates	Analytics	150	2450000
		Tracking	300	350000

14. Same as above, but adds a “Total” row and column for overall sums using `margins=True`.

[43]:
```
# get Final "ALL" with margins = True
# Can add on multiple columns
pd.pivot_table(df,index=["Account Manager","Contact","Product"],values=["Sale Price","Licenses"],
               aggfunc=[np.sum],fill_value=0,margins=True)
```

[43]:

			sum	
			Licenses	Sale Price
Account Manager	Contact	Product		
Claude Shannon	Cindy Phoner	Analytics	450	6650000
		Prediction	150	700000
		Tracking	150	350000
	Emma Gordian	Analytics	1050	11550000
		GPS Positioning	300	350000
		Tracking	450	490000
Edward Thorp	Elon Tusk	Analytics	600	7350000
		Prediction	150	700000
	Larry Pager	Analytics	300	4550000
		Prediction	150	700000
		Tracking	300	350000

15. Creates a pivot table grouped by account manager and status, summing sale prices and adding totals with `margins=True`.

[45]:
```
pd.pivot_table(df,index=["Account Manager","Status"],values=["Sale Price"],
               aggfunc=[np.sum],fill_value=0,margins=True)
```

[45]:

		sum
		Sale Price
Account Manager	Status	
Claude Shannon	Lost	4550000
	Presented	3150000
	Under Review	350000
	Won	12040000
Edward Thorp	Lost	4900000
	Presented	3500000
	Under Review	3500000
	Won	4550000
All		36540000