Matplotlib & Seaborn - 1

🎯 Session Objectives:

- ✅ Understand what data visualization is and why it matters
- ✅ Use Matplotlib to plot different types of charts
- ✅ Customize plots with markers, colors, linewidth, and line styles
- ✅ Integrate Matplotlib with NumPy and Pandas
- ✅ Understand why Seaborn is important in visualization
- ✅ Recognize common Seaborn plot types

```
# Concatenate : pd.concat() -> Append Queries [Union / Union All]
# Stack [Axis = 0]
combined_df = pd.concat([customers,products],axis = 0) # Prefer to use Identical Table [Sales2015/16/17]
combined_df
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city	s
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	Califo
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso	T
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	T
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tenne
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore	Mary
...	...	...	...	...	...	...	...	...	...	
55	56	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
# Merging [Merge Query] [Joins] 'on' constraints -> [P.K , F.K] [how = 'inner','left','right']
inner_merge_df = pd.merge(customers,products,on='id',how='inner')
inner_merge_df
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore

```
left_merge_df = pd.merge(customers,products,on='id',how='left')
left_merge_df
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	Sar
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	A
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	M

```
right_merge_df = pd.merge(customers,products,on='id',how='right') # Inner Join
right_merge_df
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore

```
right_merge_df = pd.merge(customers,purchases,on='id',how='right')
right_merge_df
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city	
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	Ca
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso	
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Ter
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore	M
...	...	...	...	...	...	...	...	...	...	
5995	5996	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5996	5997	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5997	5998	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5998	5999	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5999	6000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

6000 rows × 16 columns

```python
# Overlapping Columns Handles with Suffixes
Customer = pd.DataFrame(
    {
        'id' : [1,2,3],
        'name' : ['Shyam','Deepak','Rajat'],
        'product' : ['Book','Pen','NoteBook']
    }
)
Product = pd.DataFrame(
    {
        'id' : [2,3,4],
        'product' : ['Pen','NoteBook','Pencil'],
        'amount' : [10,25,15]
    }
)
Customer
```

	id	name	product
0	1	Shyam	Book
1	2	Deepak	Pen
2	3	Rajat	NoteBook

```python
Product
```

	id	product	amount
0	2	Pen	10
1	3	NoteBook	25
2	4	Pencil	15

```python
merge_df = pd.merge(Customer,Product,on='id',how='inner',suffixes=('_cust','_prod'))
merge_df
```

	id	name	product_cust	product_prod	amount
0	2	Deepak	Pen	Pen	10
1	3	Rajat	NoteBook	NoteBook	25

```python
merge_df = pd.merge(Customer,Product,on='id',how='left',suffixes=('_cust','_prod'))
merge_df
```

	id	name	product_cust	product_prod	amount
0	1	Shyam	Book	NaN	NaN
1	2	Deepak	Pen	Pen	10.0
2	3	Rajat	NoteBook	NoteBook	25.0

```
merge_df = pd.merge(Customer,Product,on='id',how='right',suffixes=('_cust','_prod'))
merge_df
```

	id	name	product_cust	product_prod	amount
0	2	Deepak	Pen	Pen	10
1	3	Rajat	NoteBook	NoteBook	25
2	4	NaN	NaN	Pencil	15

```
# suffixes Default _x , _y
merge_df = pd.merge(Customer,Product,on='id',how='inner')
merge_df
```

	id	name	product_x	product_y	amount
0	2	Deepak	Pen	Pen	10
1	3	Rajat	NoteBook	NoteBook	25

```
# set_index
customers = customers.set_index('id')
customers.head()
```

id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city	state
1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California
2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso	Texas
3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas
4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee
5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore	Maryland

```
purchases = purchases.set_index('id')
purchases.head()
```

id	purch_date	customer_num	product_num	amount	paid
1	2019-01-03	823	27	12	568.92
2	2019-01-03	606	28	14	395.36

```
# Joining with Index
joined_df = customers.join(purchases)
joined_df
```

id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city
1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego
2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso
3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio
4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis
5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore
...	...	...	...	...	...	...	...	...
996	Merrili	Alman	malmanrn@cornell.edu	Female	0.0	Thompson	Way	Reading
997	Winonah	Heckle	whecklero@fc2.com	Female	701.0	Rowland	Hill	Indianapolis
998	Tobit	Birt	Unknown	Male	2.0	Basil	Road	Waterbury
999	Issiah	Standbrooke	istandbrookerq@yellowpages.com	Male	0.0	Kenwood	Drive	Savannah
1000	Elmore	Malpas	Unknown	Male	205.0	Farwell	Park	Atlanta

1000 rows × 15 columns

```
customers = customers.reset_index()
customers.head()
```

	index	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city
0	0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego
1	1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso
2	2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio
3	3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis
4	4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore

```
customers = customers.drop('index',axis = 1)
customers
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San D
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	An
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Mer

```
purchases = purchases.reset_index()
purchases.head()
```

	id	purch_date	customer_num	product_num	amount	paid
0	1	2019-01-03	823	27	12	568.92
1	2	2019-01-03	606	28	14	395.36
2	3	2019-01-03	955	9	17	510.17
3	4	2019-01-03	577	19	3	68.49
4	5	2019-01-03	429	8	18	759.42

```
# Joining with Index
joined_df = customers.set_index('id').join(purchases.set_index('id'))
joined_df
```

id	first_name	last_name	email	gender	street_number	street_address	street_suffix	ci
1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Dieg
2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Pa
3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	S Anton
4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memph
5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimo
...	...	...	...	...	...	...	...	
996	Merrili	Alman	malmanrn@cornell.edu	Female	0.0	Thompson	Way	Readi

```
customers.head()
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city	stat
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	Californi
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso	Texa
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texa
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennesse
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore	Marylan

```
# Drop_duplicates
unique_customers = customers.drop_duplicates(subset = 'first_name')
unique_customers
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	city
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore
...	...	...	...	...	...	...	...	...	...
994	995	Brana	Dixon	bdixonrm@myspace.com	Female	97.0	Truax	Avenue	Maple Plain
995	996	Merrili	Alman	malmanrn@cornell.edu	Female	0.0	Thompson	Way	Reading P
996	997	Winonah	Heckle	whecklero@fc2.com	Female	701.0	Rowland	Hill	Indianapolis
997	998	Tobit	Birt	Unknown	Male	2.0	Basil	Road	Waterbury
999	1000	Elmore	Malpas	Unknown	Male	205.0	Farwell	Park	Atlanta

932 rows × 11 columns

```
# Drop_duplicates
unique_customers = customers.drop_duplicates(subset = 'last_name')
unique_customers
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San D
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	An
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Mer
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Balti
...	...	...	...	...	...	...	...	...	
995	996	Merrili	Alman	malmanrn@cornell.edu	Female	0.0	Thompson	Way	Rea
996	997	Winonah	Heckle	whecklero@fc2.com	Female	701.0	Rowland	Hill	Indiana
997	998	Tobit	Birt	Unknown	Male	2.0	Basil	Road	Wate
998	999	Issiah	Standbrooke	istandbrookerq@yellowpages.com	Male	0.0	Kenwood	Drive	Sava
999	1000	Elmore	Malpas	Unknown	Male	205.0	Farwell	Park	At

993 rows × 11 columns

```
check_dup_cols = customers.iloc[:,1:].columns
check_dup_cols
```

```
Index(['first_name', 'last_name', 'email', 'gender', 'street_number',
       'street_address', 'street_suffix', 'city', 'state', 'postcode'],
      dtype='object')
```

```
# Drop_duplicates
unique_customers = customers.drop_duplicates(subset = check_dup_cols)
unique_customers
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix	
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	Sa
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	M
4	5	Mikaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Ba
...	...	...	...	...	...	...	...	...	
995	996	Merrili	Alman	malmanrn@cornell.edu	Female	0.0	Thompson	Way	
996	997	Winonah	Heckle	whecklero@fc2.com	Female	701.0	Rowland	Hill	India

```
customers['first_name'].nunique()
```

```
932
```

```
customers['first_name'].value_counts().head(7)
```

```
first_name
Berty      4
Leland     3
Abel       3
Gunther    2
Silvain    2
Corry      2
Obediah    2
Name: count, dtype: int64
```

```
dropna(axis = 0) -> If at any cell a NaN Value exist -> Removes the complete row
dropna(axis = 1) -> If at any cell a NaN Value exist -> Removes the complete column
```

```
# .interpolate() - [Linear , Polynomial , degree 2] = [0 2 4 _ 8 10 _ 14 _ 18]
data = {
    'index' : range(1,11), # [1,2,3....10]
    'cost' : [100,200,150,None,250,None,350,None,450,600]
}
df = pd.DataFrame(data)
df
```

	index	cost
0	1	100.0
1	2	200.0
2	3	150.0
3	4	NaN
4	5	250.0
5	6	NaN
6	7	350.0
7	8	NaN
8	9	450.0
9	10	600.0

```
df.dropna(axis = 0)
```

	index	cost
0	1	100.0
1	2	200.0
2	3	150.0
4	5	250.0
6	7	350.0
8	9	450.0
9	10	600.0

```
df.dropna(axis = 1)
```

	index
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10

```
df['cost'] = df['cost'].interpolate(method = 'linear')
df
```

	index	cost
0	1	100.0
1	2	200.0
2	3	150.0
3	4	200.0
4	5	250.0
5	6	300.0
6	7	350.0
7	8	400.0
8	9	450.0
9	10	600.0

```
# .interpolate() - [Linear , Polynomial , degree 2] = [0 2 4 _ 8 10 _ 14 _ 18]
data = {
    'index' : range(1,11), # [1,2,3....10]
    'cost' : [100,200,150,None,250,None,350,None,450,600]
}
df = pd.DataFrame(data)
df['cost'] = df['cost'].interpolate(method = 'polynomial' , order = 2)
df
```

	index	cost
0	1	100.000000
1	2	200.000000
2	3	150.000000
3	4	175.893566
4	5	250.000000
5	6	306.989674
6	7	350.000000
7	8	382.168388
8	9	450.000000
9	10	600.000000

```
# Data Aggregations + GroupBy [SQL]
products['cost'].aggregate('max')
```

```
99.54
```

```
products['cost'].aggregate('mean')
```

```
55.156166666666664
```

```
products['cost'].aggregate('median')
```

```
57.78
```

```
products['cost'].aggregate('sum')
```

```
3309.37
```

```
products['cost'].aggregate(['max','min','sum','count','mean','median','std','var'])
```

```
max         99.540000
min          6.360000
sum       3309.370000
count       60.000000
mean        55.156167
median      57.780000
std         27.552204
var        759.123963
Name: cost, dtype: float64
```

```
purchases['amount'].aggregate(['max','min','sum','count','mean','median','std','var']).round(2)
```

```
max          20.00
min           1.00
sum       63457.00
count      6000.00
mean         10.58
median       11.00
std           5.77
var          33.28
Name: amount, dtype: float64
```

```
purchases['paid'].aggregate(['max','min','sum','count','mean','median','std','var'])
```

```
max       1.990800e+03
min       3.630000e+00
sum       3.293228e+06
count     6.000000e+03
mean      5.488714e+02
median    4.227000e+02
std       4.442372e+02
var       1.973467e+05
Name: paid, dtype: float64
```

```
# GroupBY
company_cost_analysis = products.groupby('company')['cost'].sum().reset_index() # Series -> DataFrame
company_cost_analysis
```

	company	cost
0	Aibox	56.33
1	Babbleopia	63.98
2	Brainsphere	22.83
3	Browsezoom	82.10
4	Cogibox	102.27
5	Digitube	8.55
6	Dynabox	99.40
7	Dynazzy	85.74
8	Eare	87.39

```
type(company_cost_analysis)
```

```
pandas.core.frame.DataFrame
```

```
company_cost_analysis.sort_values(by = 'cost' , ascending = False).head(7)
```

	company	cost
19	Livepath	188.21
37	Skinder	159.85
34	Realcube	150.74
25	Ntag	121.17
40	Thoughtstorm	107.11
4	Cogibox	102.27
44	Vitz	100.84

```
products.groupby('company')['cost'].sum().nlargest(7).reset_index()
```

	company	cost
0	Livepath	188.21
1	Skinder	159.85
2	Realcube	150.74
3	Ntag	121.17
4	Thoughtstorm	107.11
5	Cogibox	102.27
6	Vitz	100.84

```
products.groupby('company')['cost'].sum().nsmallest(7).reset_index()
```

	company	cost
0	Skipfire	6.36
1	Digitube	8.55
2	Oloo	13.83
3	Quatz	13.83
4	Zoombox	16.27
5	Twitterworks	19.90
6	Zazio	20.53

```
# sort_values(by = ['col1','col2'] , ascending = [True,False] )
customers.sort_values(by = 'state' , ascending = False).head()
```

me	last_name	email	gender	street_number	street_address	street_suffix	city	state	postcode
ick	Gowthrop	Unknown	Male	0.0	Almo	Avenue	Springfield	Wisconsin	45505
tta	Leftwich	aleftwichdb@cafepress.com	Female	25.0	Vermont	Road	Milwaukee	Wisconsin	53234
rty	Meert	bmeertjr@hatena.ne.jp	Male	0.0	Lakeland	Drive	Milwaukee	Wisconsin	53215
Elle	Coultish	ecoultish8n@techcrunch.com	Female	88.0	Westerfield	Pass	Milwaukee	Wisconsin	53234
ulie	Gadault	pgadaultql@posterous.com	Female	2061.0	Aberg	Street	Milwaukee	Wisconsin	53285

```
products.sort_values(by='cost',ascending = False) # High to Low
```

	id	product	cost	company
33	34	Muffin Hinge Container 6	99.54	Skyble
13	14	Cookies Oatmeal Raisin	99.40	Dynabox
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath
51	52	Soup - Knorr, Country Bean	98.74	Eimbee
34	35	Pie Box - Cello Window 2.5	95.68	Linkbuzz
31	32	Container Clear 8 Oz	92.33	Muxo
20	21	Bagel - Whole White Sesame	90.48	Realcube
9	10	Sambuca - Ramazzotti	88.99	Livepath
41	42	Sauce - Demi Glace	87.51	Wordware

```
purchases.sort_values(by = ['amount','paid'] , ascending = False).head(10) # Both Columns [High to Low]
```

	id	purch_date	customer_num	product_num	amount	paid
211	212	2019-01-06	918	34	20	1990.8
2279	2280	2019-03-13	685	34	20	1990.8
4846	4847	2019-05-13	993	34	20	1990.8
2924	2925	2019-03-24	619	14	20	1988.0
3222	3223	2019-03-27	747	14	20	1988.0
4148	4149	2019-04-21	139	14	20	1988.0
697	698	2019-04-04	969	6	20	1984.4

```
purchases.sort_values(by = ['amount','paid'] , ascending = [False,True])
# amount -> Descending , paid - Ascending
```

	id	purch_date	customer_num	product_num	amount	paid
2422	2423	2019-03-16	690	1	20	63.60
5254	5255	2019-05-22	652	1	20	101.80
3388	3389	2019-03-29	155	1	20	127.20
3849	3850	2019-04-16	759	1	20	127.20
5849	5850	2019-06-17	113	1	20	127.20
...	...	...	...	...	...	...
5061	5062	2019-05-18	540	52	1	98.74
331	332	2019-02-05	418	6	1	99.22
5771	5772	2019-06-15	138	14	1	99.40
1301	1302	2019-07-06	450	34	1	99.54
5070	5071	2019-05-18	317	34	1	99.54

6000 rows × 6 columns

```
purchases.sort_values(by = ['amount','paid'] , ascending = [True,False])
# amount -> Ascending , paid - Descending
```

	id	purch_date	customer_num	product_num	amount	paid
1301	1302	2019-07-06	450	34	1	99.54
5070	5071	2019-05-18	317	34	1	99.54
5771	5772	2019-06-15	138	14	1	99.40
331	332	2019-02-05	418	6	1	99.22
641	642	2019-03-06	519	52	1	98.74
...	...	...	...	...	...	...
3388	3389	2019-03-29	155	1	20	127.20
3849	3850	2019-04-16	759	1	20	127.20
5849	5850	2019-06-17	113	1	20	127.20
5254	5255	2019-05-22	652	1	20	101.80
2422	2423	2019-03-16	690	1	20	63.60

6000 rows × 6 columns

```
# Calculated Columns [DAX] (row context , filtering)
# Add Column [Power Query Editor] [Date column -> Extract [Year,month,days]]
purchases['year'] = purchases['purch_date'].dt.year
purchases['month'] = purchases['purch_date'].dt.month
purchases['item_price'] = purchases['paid'] / purchases['amount']
purchases
```

	id	purch_date	customer_num	product_num	amount	paid	year	month	item_price
0	1	2019-01-03	823	27	12	568.92	2019	1	47.41
1	2	2019-01-03	606	28	14	395.36	2019	1	28.24
2	3	2019-01-03	955	9	17	510.17	2019	1	30.01
3	4	2019-01-03	577	19	3	68.49	2019	1	22.83
4	5	2019-01-03	429	8	18	759.42	2019	1	42.19
...	...	...	...	...	...	...	...	...	...
5995	5996	2019-06-20	893	33	5	411.10	2019	6	82.22
5996	5997	2019-06-20	566	23	11	178.97	2019	6	16.27
5997	5998	2019-06-20	114	19	9	205.47	2019	6	22.83
5998	5999	2019-06-20	404	11	20	429.40	2019	6	21.47
5999	6000	2019-06-20	88	57	4	274.52	2019	6	68.63

6000 rows × 9 columns

```
# argsort() in numpy -> idxmax()
products.idxmax()
```

```
id         59
product     5
cost       33
company    22
dtype: int64
```

```
products.idxmax()['product'] # 5
```

```
5
```

```
products.loc[products.idxmax()['product']]
```

```
id                                          6
product    Wine - White, Riesling, Semi - Dry
cost                                    99.22
company                              Livepath
Name: 5, dtype: object
```

```
products.iloc[products.idxmax()['product']]
```

```
id                                          6
product    Wine - White, Riesling, Semi - Dry
cost                                    99.22
company                              Livepath
Name: 5, dtype: object
```

```
products.loc[[products.idxmax()['product']]] # DataFrame
```

	id	product	cost	company
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath

```
products.loc[[5]]
```

	id	product	cost	company
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath

```
products.loc[[products.idxmax()['cost']]] # DataFrame
```

	id	product	cost	company
33	34	Muffin Hinge Container 6	99.54	Skyble

```
products.loc[[products.idxmax()['company']]] # DataFrame
```

	id	product	cost	company
22	23	Puree - Blackcurrant	16.27	Zoombox

```
products.loc[5] # Series
```

```
id                                          6
product    Wine - White, Riesling, Semi - Dry
cost                                    99.22
company                              Livepath
Name: 5, dtype: object
```

```
products.loc[[5]] # DataFrame
```

	id	product	cost	company
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath

```
# Group BY + Aggregations
sorted_product = products.sort_values(by = ['cost'] , ascending = False)
sorted_product
```

	id	product	cost	company
33	34	Muffin Hinge Container 6	99.54	Skyble
13	14	Cookies Oatmeal Raisin	99.40	Dynabox
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath
51	52	Soup - Knorr, Country Bean	98.74	Eimbee
34	35	Pie Box - Cello Window 2.5	95.68	Linkbuzz
31	32	Container Clear 8 Oz	92.33	Muxo
20	21	Bagel - Whole White Sesame	90.48	Realcube
9	10	Sambuca - Ramazzotti	88.99	Livepath
41	42	Sauce - Demi Glace	87.51	Wordware

```
sorted_product.groupby(['company'])['cost'].mean() # Series
```

```
company
Aibox           56.330000
Babbleopia      63.980000
Brainsphere     22.830000
Browsezoom      41.050000
Cogibox         51.135000
```

```
sorted_product.groupby(['company'])['cost'].mean().reset_index() # dataFrame
```

	company	cost
0	Aibox	56.330000
1	Babbleopia	63.980000
2	Brainsphere	22.830000
3	Browsezoom	41.050000
4	Cogibox	51.135000
5	Digitube	8.550000
6	Dynabox	99.400000
7	Dynazzy	85.740000
8	Eare	87.390000

```
sorted_product.groupby(['company'])['cost'].mean().reset_index().sort_values(by = 'cost' , ascending = False)
```

	company	cost
39	Skyble	99.540000
6	Dynabox	99.400000
11	Eimbee	98.740000
18	Linkbuzz	95.680000

```
sorted_product.groupby(['company'])['cost'].mean().nlargest(10)
```

```
company
Skyble      99.540
Dynabox     99.400
Eimbee      98.740
Linkbuzz    95.680
Livepath    94.105
Muxo        92.330
Wordware    87.510
Eare        87.390
Trunyx      87.050
Dynazzy     85.740
Name: cost, dtype: float64
```

```
# Trend-Axis
purchases.set_index('purch_date')
```

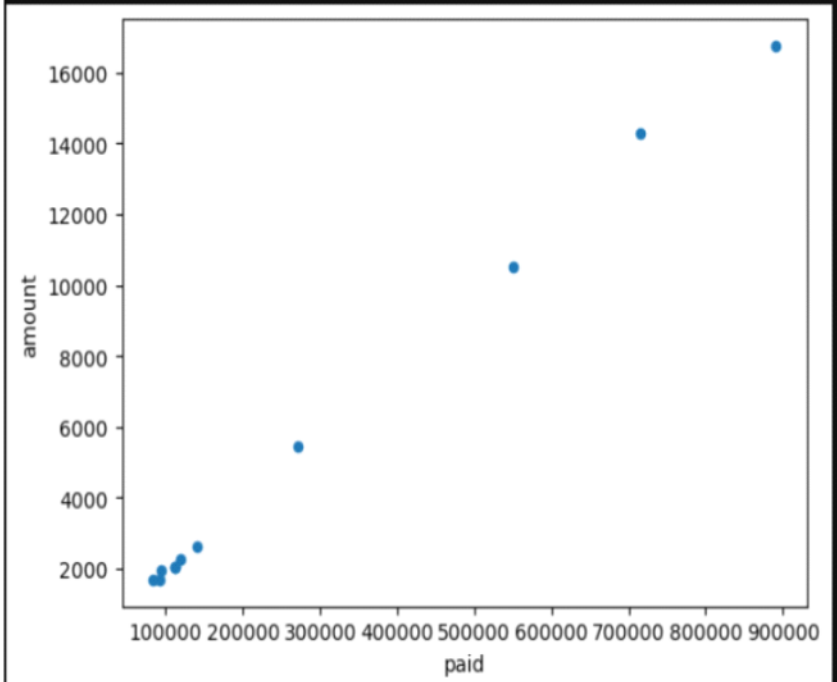
purch_date	id	customer_num	product_num	amount	paid	year	month	item_price
2019-01-03	1	823	27	12	568.92	2019	1	47.41
2019-01-03	2	606	28	14	395.36	2019	1	28.24
2019-01-03	3	955	9	17	510.17	2019	1	30.01
2019-01-03	4	577	19	3	68.49	2019	1	22.83
2019-01-03	5	429	8	18	759.42	2019	1	42.19
...	...	...	...	...	...	...	...	...
2019-06-20	5996	893	33	5	411.10	2019	6	82.22
2019-06-20	5997	566	23	11	178.97	2019	6	16.27

```
purchases.groupby('month')[['paid','amount']].sum()
```

month	paid	amount
1	139986.42	2643
2	83532.80	1713
3	890751.05	16752
4	715885.30	14268
5	550416.29	10537
6	271329.22	5457
7	112608.31	2063
8	118668.27	2291
9	94420.38	1962
10	92266.35	1695
11	111315.89	2044
12	112047.84	2032

```
purchases.groupby('month')[['paid','amount']].sum().plot.scatter('paid','amount')
```

```
<Axes: xlabel='paid', ylabel='amount'>
```




```
# .isin() -> In 'Membership Operators' [Boolean Results]
company_list = ['Ntag','Skipfire','Vitz','Realcube','Zoombox']
products[products['company'].isin(company_list)]
```

	id	product	cost	company
0	1	Liners - Baking Cups	6.36	Skipfire
2	3	Bar Bran Honey Nut	65.40	Ntag
14	15	Cookies Oatmeal Raisin	14.13	Vitz
15	16	Sausage - Chorizo	55.45	Vitz
20	21	Bagel - Whole White Sesame	90.48	Realcube
21	22	Scotch - Queen Anne	60.26	Realcube
22	23	Puree - Blackcurrant	16.27	Zoombox
23	24	Bread - Bagels, Plain	31.26	Vitz
30	31	Wine - Carmenere Casillero Del	55.77	Ntag

```
# .between() [Inclusive][Continous Column]
purchases[purchases['amount'].between(11,20)]
```

	id	purch_date	customer_num	product_num	amount	paid	year	month	item_price
0	1	2019-01-03	823	27	12	568.92	2019	1	47.41
1	2	2019-01-03	606	28	14	395.36	2019	1	28.24
2	3	2019-01-03	955	9	17	510.17	2019	1	30.01
4	5	2019-01-03	429	8	18	759.42	2019	1	42.19
5	6	2019-01-03	275	36	18	176.76	2019	1	9.82
...	...	...	...	...	...	...	...	...	...
5990	5991	2019-06-20	632	2	15	1286.10	2019	6	85.74
5992	5993	2019-06-20	159	18	17	957.61	2019	6	56.33
5994	5995	2019-06-20	840	23	18	292.86	2019	6	16.27
5996	5997	2019-06-20	566	23	11	178.97	2019	6	16.27
5998	5999	2019-06-20	404	11	20	429.40	2019	6	21.47

3023 rows × 9 columns

```
# .between() [Inclusive][Continous Column] # DatesBetween
purchases[purchases['purch_date'].between('01-01-2019','31-03-2019')]
```

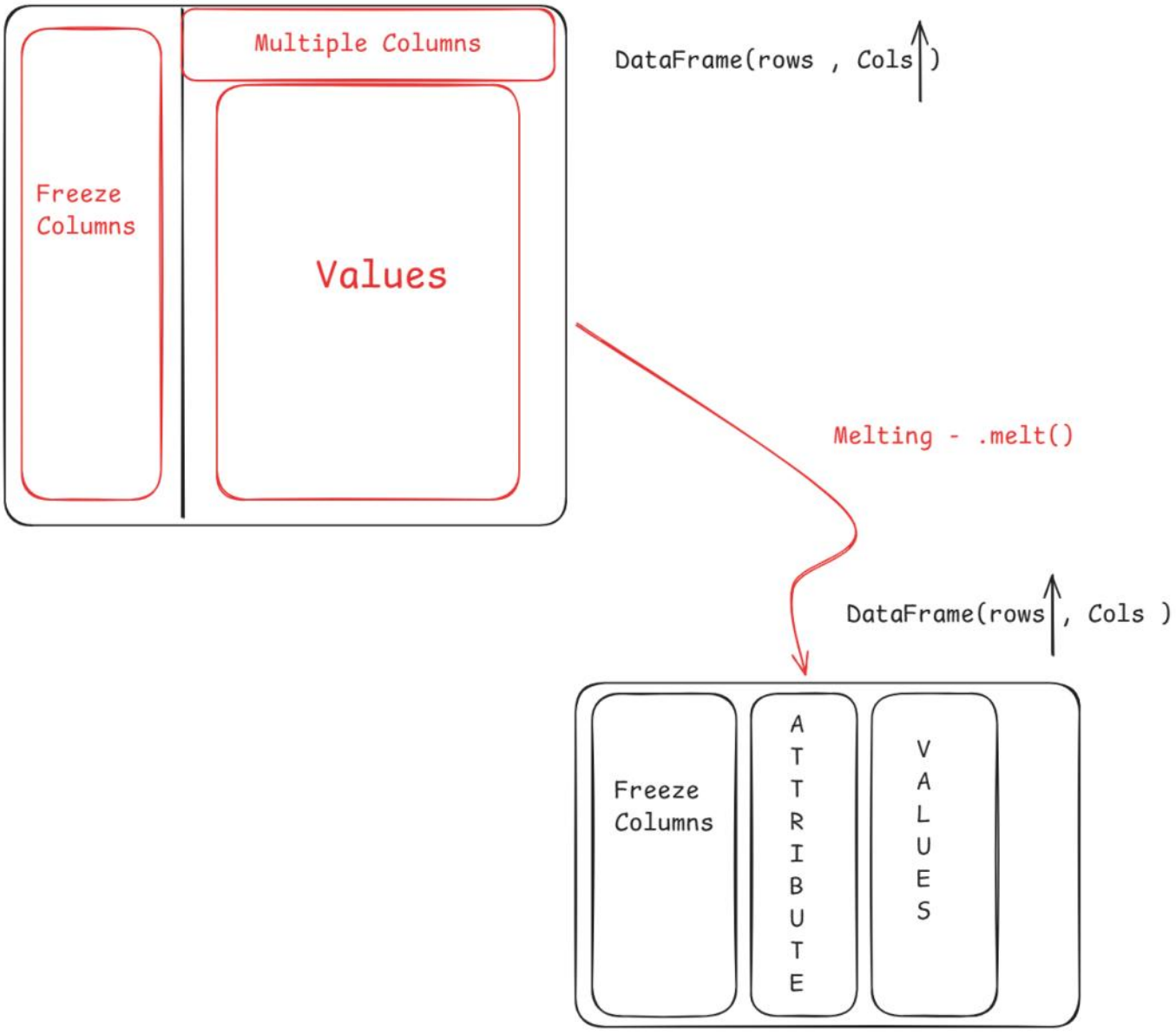
	id	purch_date	customer_num	product_num	amount	paid	year	month	item_price
0	1	2019-01-03	823	27	12	568.92	2019	1	47.41
1	2	2019-01-03	606	28	14	395.36	2019	1	28.24
2	3	2019-01-03	955	9	17	510.17	2019	1	30.01
3	4	2019-01-03	577	19	3	68.49	2019	1	22.83
4	5	2019-01-03	429	8	18	759.42	2019	1	42.19
...	...	...	...	...	...	...	...	...	...
3633	3634	2019-03-31	128	56	17	279.14	2019	3	16.42
3634	3635	2019-03-31	145	3	6	392.40	2019	3	65.40
3635	3636	2019-03-31	224	25	8	589.60	2019	3	73.70
3636	3637	2019-03-31	260	28	4	108.44	2019	3	27.11
3637	3638	2019-03-31	846	42	16	1400.16	2019	3	87.51

2010 rows × 9 columns

pandas.pivot_table

```
pandas.pivot_table(data, values=None, index=None, columns=None,
aggfunc='mean', fill_value=None, margins=False, dropna=True,
margins_name='All', observed=True, sort=True,
**kwargs)
```

[source]



pandas.melt

```
pandas.melt(frame, id_vars=None, value_vars=None, var_name=None,
value_name='value', col_level=None, ignore_index=True)
```

[source]

Unpivot a DataFrame from wide to long format, optionally leaving identifiers set.

```
# Pivot Table
data = {
    'Region' : ['North','East','East','West','West','North','East','East','West','North'],
    'Products' : ['Bikes','Clothings','Accessories','Components','Bikes','Clothings',
                  'Accessories','Components','Clothings','Accessories'],
    'Revenue' : [2000,1000,1500,1000,2500,5000,1500,2000,1000,7000]
}
df = pd.DataFrame(data)
df
```

	Region	Products	Revenue
0	North	Bikes	2000
1	East	Clothings	1000
2	East	Accessories	1500
3	West	Components	1000
4	West	Bikes	2500
5	North	Clothings	5000
6	East	Accessories	1500
7	East	Components	2000
8	West	Clothings	1000
9	North	Accessories	7000

```
pivot_table = df.pivot_table(
    values='Revenue',
    index='Region',
    columns='Products',
    aggfunc = 'sum'
)
pivot_table
```

Products	Accessories	Bikes	Clothings	Components
Region				
East	3000.0	NaN	1000.0	2000.0
North	7000.0	2000.0	5000.0	NaN
West	NaN	2500.0	1000.0	1000.0

```
pivot_table = df.pivot_table(
    values='Revenue',
    index='Region',
    columns='Products',
    aggfunc = 'sum',
    fill_value = 0
)
pivot_table
```

Products	Accessories	Bikes	Clothings	Components
Region				
East	3000	0	1000	2000
North	7000	2000	5000	0
West	0	2500	1000	1000

```
pivot_table = df.pivot_table(
    values='Revenue',
    index= 'Products',
    columns='Region',
    aggfunc = 'sum',
    fill_value = 0
)
pivot_table
```

Region	East	North	West
Products			
Accessories	3000	7000	0
Bikes	0	2000	2500
Clothings	1000	5000	1000
Components	2000	0	1000

```
filter_pivot = pivot_table[pivot_table.sum(axis=1) >= 7000]
filter_pivot
```

Region	East	North	West
Products			
Accessories	3000	7000	0
Clothings	1000	5000	1000

```
# Observe the caller()
pivot_table = pd.pivot_table(
    data = df,
    values='Revenue',
    index='Region',
    columns='Products',
    aggfunc = 'sum',
    fill_value = 0
)
pivot_table
```

Products	Accessories	Bikes	Clothings	Components
Region				
East	3000	0	1000	2000
North	7000	2000	5000	0
West	0	2500	1000	1000

```python
# Observe the caller()
pivot_table = pd.pivot_table(
    data = df,
    values='Revenue',
    index=['Region','Products'],
    aggfunc = 'sum',
)
pivot_table
```

		Revenue
Region	**Products**	
East	**Accessories**	3000
	Clothings	1000
	Components	2000
North	**Accessories**	7000
	Bikes	2000
	Clothings	5000
West	**Bikes**	2500
	Clothings	1000
	Components	1000

```python
# Performing Pivot Table with Multiple Aggregations
pivot_table = pd.pivot_table(
    data = df,
    values='Revenue',
    index=['Region','Products'],
    aggfunc = ['sum','mean','median','max','min'],
)
pivot_table
```

		sum	mean	median	max	min
		Revenue	Revenue	Revenue	Revenue	Revenue
Region	**Products**					
East	**Accessories**	3000	1500.0	1500.0	1500	1500
	Clothings	1000	1000.0	1000.0	1000	1000
	Components	2000	2000.0	2000.0	2000	2000
North	**Accessories**	7000	7000.0	7000.0	7000	7000
	Bikes	2000	2000.0	2000.0	2000	2000
	Clothings	5000	5000.0	5000.0	5000	5000
West	**Bikes**	2500	2500.0	2500.0	2500	2500
	Clothings	1000	1000.0	1000.0	1000	1000
	Components	1000	1000.0	1000.0	1000	1000

```
products_pivot = pd.pivot_table(
    data = products,
    values=['cost','id'],
    index=['company'],
    aggfunc = {'cost':'sum','id':'count'}
)
products_pivot
```

company	cost	id
Aibox	56.33	1
Babbleopia	63.98	1
Brainsphere	22.83	1
Browsezoom	82.10	2
Cogibox	102.27	2
Digitube	8.55	1
Dynabox	99.40	1
Dynazzy	85.74	1

```
type(products_pivot)
```

```
pandas.core.frame.DataFrame
```

```
products_pivot.sort_values(by = 'id' , ascending = False).head(7)
```

company	cost	id
Skinder	159.85	3
Vitz	100.84	3
Ntag	121.17	2
Browsezoom	82.10	2
Cogibox	102.27	2
Livepath	188.21	2
Realcube	150.74	2

```
# CrossTab
# Regions VS Products [Frequecy]
data = {
    'Region' : ['North','East','East','West','West','North','East','East','West','North'],
    'Products' : ['Bikes','Clothings','Accessories','Components','Bikes','Clothings',
                  'Accessories','Components','Clothings','Accessories'],
    'Revenue' : [2000,1000,1500,1000,2500,5000,1500,2000,1000,7000]
}
df = pd.DataFrame(data)
df
```

	Region	Products	Revenue
0	North	Bikes	2000
1	East	Clothings	1000
2	East	Accessories	1500
3	West	Components	1000
4	West	Bikes	2500
5	North	Clothings	5000
6	East	Accessories	1500
7	East	Components	2000
8	West	Clothings	1000
9	North	Accessories	7000

```
# CrossTab
cross_tab = pd.crosstab(df['Region'],df['Products'])
cross_tab
```

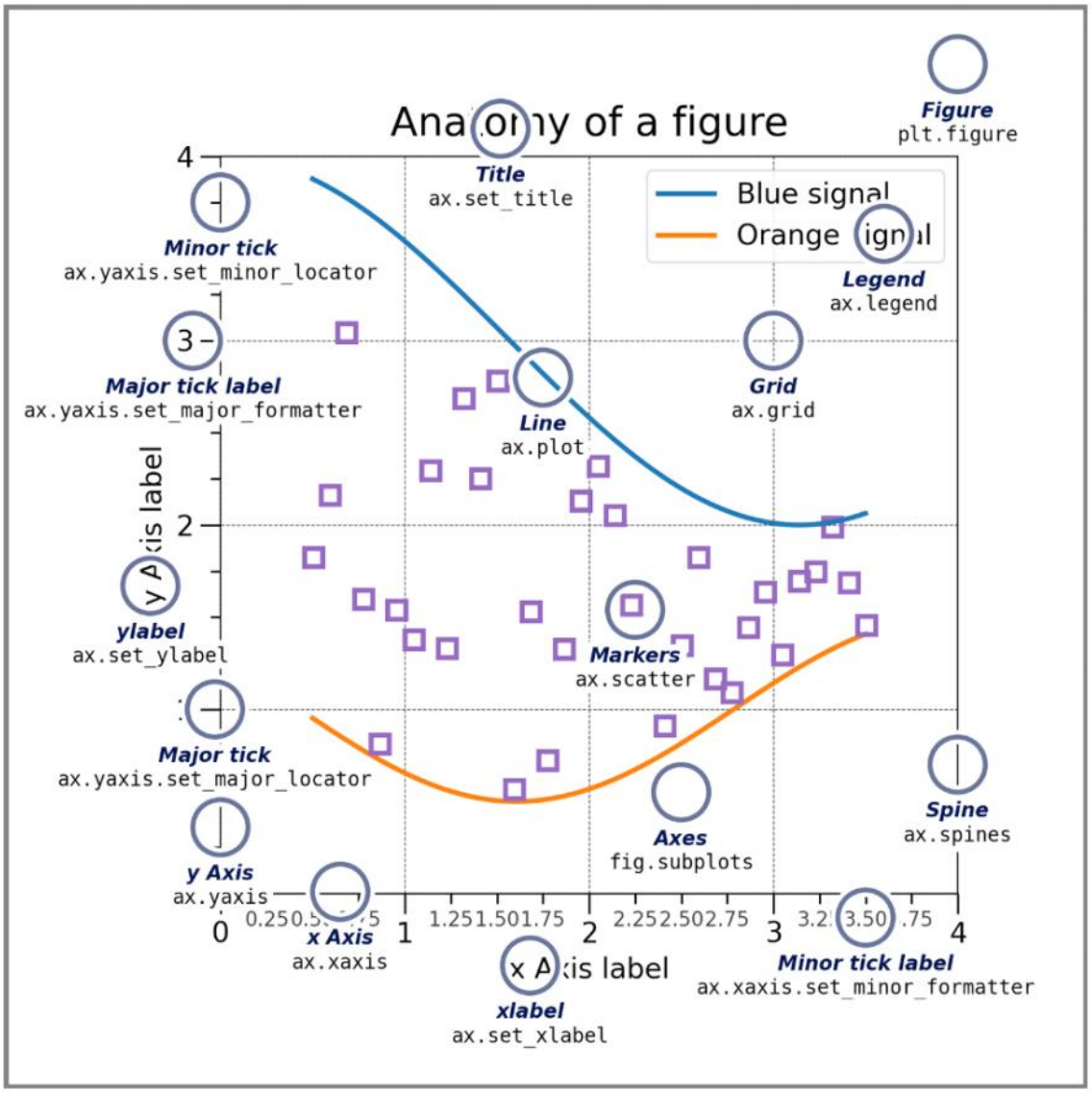
Products	Accessories	Bikes	Clothings	Components
Region				
East	2	0	1	1
North	1	1	1	0
West	0	1	1	1

```
# Pivot Table -> 'Unpivot' Table
data = {
    'Region' : ['North','East','West'],
    'Bikes' : [2000,1500,1000],
    'Components' : [200,500,700],
    'Accessories' : [250,300,100],
    'Clothings' : [100,700,900]
}
df = pd.DataFrame(data)
df
```

	Region	Bikes	Components	Accessories	Clothings
0	North	2000	200	250	100
1	East	1500	500	300	700
2	West	1000	700	100	900

```
# Melting the .df -> to have more extended rows
# Unpivot Table
melted_df = pd.melt(
    df,
    id_vars='Region',
    value_vars=['Bikes','Components','Accessories','Clothings'],
    var_name='Products',
    value_name= 'Total Sales',
)
melted_df
```

	Region	Products	Total Sales
0	North	Bikes	2000
1	East	Bikes	1500
2	West	Bikes	1000
3	North	Components	200
4	East	Components	500
5	West	Components	700
6	North	Accessories	250
7	East	Accessories	300
8	West	Accessories	100
9	North	Clothings	100
10	East	Clothings	700
11	West	Clothings	900


Anatomy of a figure
Figure
plt.figure
Title
ax.set_title
Minor tick
ax.yaxis.set_minor_locator
Blue signal
Orange signal
Legend
ax.legend
Major tick label
ax.yaxis.set_major_formatter
Grid
ax.grid
Line
ax.plot
y Axis label
ylabel
ax.set_ylabel
Markers
ax.scatter
Major tick
ax.yaxis.set_major_locator
Spine
ax.spines
Axes
fig.subplots
y Axis
ax.yaxis
x Axis
ax.xaxis
x Axis label
xlabel
ax.set_xlabel
Minor tick label
ax.xaxis.set_minor_formatter