Cont. Pandas - II

🎯 Session Objectives:

- ✅ Understand the key attributes of a DataFrame
- ✅ Use various DataFrame methods to explore and manipulate data
- ✅ Perform row and column operations
- ✅ Use joining, merging, and concatenation techniques across DataFrames
- ✅ Understand and perform statistical operations
- ✅ Conduct frequency and aggregation analysis
- ✅ Learn about sorting and indexing
- ✅ Export DataFrames to CSV/Excel files

```
import numpy as np
import pandas as pd
# Importing the DataSet.....
try:
    customers = pd.read_excel('anaconda_projects/CN-Python-Weekend/customers.xlsx')
    products = pd.read_excel('anaconda_projects/CN-Python-Weekend/products.xlsx')
    purchases = pd.read_excel('anaconda_projects/CN-Python-Weekend/purchases.xlsx')
except FileNotFoundError:
    print("File Not Found!")
```

```
File Not Found!
```

```
import os
os.getcwd()
```

```
'C:\\Users\\krish\\anaconda_projects\\CN-Python-Weekend'
```

```
import numpy as np
import pandas as pd
# Importing the DataSet.....
try:
    customers = pd.read_excel('C:\\Users\\krish\\anaconda_projects\\CN-Python-Weekend\\customers.xlsx')
    products = pd.read_excel('C:\\Users\\krish\\anaconda_projects\\CN-Python-Weekend\\products.xlsx')
    purchases = pd.read_excel('C:\\Users\\krish\\anaconda_projects\\CN-Python-Weekend\\purchases.xlsx')
except FileNotFoundError:
    print("File Not Found!")
```

```
customers.head()
```

	id	first_name	last_name	email	gender	street_num	street_name	street_suffix	city	state	po
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California	9
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	NaN	El Paso	Texas	
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas	7
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee	3
4	5	Mikaela	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive	NaN	NaN	8

```
# DataFrame Attributes [.index] -> range
customers.index # range(Start=0,Stop=1000,Step=1)
```

```
RangeIndex(start=0, stop=1000, step=1)
```

```
products.index
```

```
RangeIndex(start=0, stop=60, step=1)
```

```
purchases.index
```

```
RangeIndex(start=0, stop=6000, step=1)
```

```
# Numpy 2D Array -> .shape(rows,cols)
# DataFrame.shape
customers.shape
```

```
(1000, 11)
```

```
products.shape
```

```
(60, 5)
```

```
purchases.shape
```

```
(6000, 7)
```

```
customers_rows = customers.shape[0]
customers_rows
```

```
1000
```

```
customers_cols = customers.shape[1]
customers_cols
```

```
11
```

```
# .columns [providing column headers details (label)]
customers.columns
```

```
Index(['id', 'first_name', 'last_name', 'email', 'gender', 'street_num',
       'street_name', 'street_suffix', 'city', 'state', 'postcode'],
      dtype='object')
```

```
products.columns
```

```
Index(['Unnamed: 0', 'id', 'product', 'cost', 'company'], dtype='object')
```

```
purchases.columns
```

```
Index(['Unnamed: 0', 'id', 'purch_date', 'customer_num', 'product_num',
       'amount', 'paid'],
      dtype='object')
```

```
# .dtypes [data types of each columns]
customers.dtypes
```

```
id              int64
first_name     object
last_name      object
email          object
gender         object
street_num    float64
street_name    object
street_suffix  object
city           object
state          object
postcode      float64
dtype: object
```

```
products.dtypes
```

```
Unnamed: 0     int64
id             int64
product       object
cost          object
company       object
dtype: object
```

```
purchases.dtypes
```

```
Unnamed: 0       int64
id               int64
purch_date      object
customer_num     int64
product_num      int64
amount           int64
paid            object
dtype: object
```

```
purchases.head()
```

	Unnamed: 0	id	purch_date	customer_num	product_num	amount	paid

```
purchases.head()
```

	Unnamed: 0	id	purch_date	customer_num	product_num	amount	paid
0	0	1	2019-01-03 00:00:00	823	27	12	$568.92
1	1	2	2019-01-03 00:00:00	606	28	14	$395.36
2	2	3	2019-01-03 00:00:00	955	9	17	$510.17
3	3	4	2019-01-03 00:00:00	577	19	3	$68.49
4	4	5	2019-01-03 00:00:00	429	8	18	$759.42

```
# .describe() -> Statistical Summary [Continuous Vs Categorical]
customers.describe() # Default cont.
```

	id	street_num	postcode
count	1000.000000	738.000000	843.000000
mean	500.500000	10536.439024	52669.548043
std	288.819436	23050.537603	28140.041026
min	1.000000	0.000000	214.000000
25%	250.750000	21.000000	29279.500000
50%	500.500000	445.500000	48232.000000
75%	750.250000	6918.500000	78337.500000
max	1000.000000	99918.000000	99812.000000

```
customers.describe(include = 'object')
```

	first_name	last_name	email	gender	street_name	street_suffix	city	state
count	1000	1000	878	957	963	963	921	920
unique	932	993	878	2	427	21	296	48
top	Berty	Sedworth	rsouthcott0@clickbank.net	Male	Arizona	Place	Washington	Texas
freq	4	3	1	485	6	57	29	107

```
customers.describe(include = 'O')
```

	first_name	last_name	email	gender	street_name	street_suffix	city	state
count	1000	1000	878	957	963	963	921	920
unique	932	993	878	2	427	21	296	48
top	Berty	Sedworth	rsouthcott0@clickbank.net	Male	Arizona	Place	Washington	Texas
freq	4	3	1	485	6	57	29	107

```
customers.describe(include = 'all')
```

	id	first_name	last_name	email	gender	street_num	street_name	street_suffix	ci
count	1000.000000	1000	1000	878	957	738.000000	963	963	9
unique	NaN	932	993	878	2	NaN	427	21	2
top	NaN	Berty	Sedworth	rsouthcott0@clickbank.net	Male	NaN	Arizona	Place	Washingt
freq	NaN	4	3	1	485	NaN	6	57	
mean	500.500000	NaN	NaN	NaN	NaN	10536.439024	NaN	NaN	Na
std	288.819436	NaN	NaN	NaN	NaN	23050.537603	NaN	NaN	Na
min	1.000000	NaN	NaN	NaN	NaN	0.000000	NaN	NaN	Na
25%	250.750000	NaN	NaN	NaN	NaN	21.000000	NaN	NaN	Na
50%	500.500000	NaN	NaN	NaN	NaN	445.500000	NaN	NaN	Na
75%	750.250000	NaN	NaN	NaN	NaN	6918.500000	NaN	NaN	Na
max	1000.000000	NaN	NaN	NaN	NaN	99918.000000	NaN	NaN	Na

```
# .values []
customers.values
```

```
array([[1, 'Romain', 'Southcott', ..., 'San Diego', 'California',
        92127.0],
       [2, 'Cosimo', 'Molyneaux', ..., 'El Paso', 'Texas', nan],
       [3, 'Bambi', 'Westrip', ..., 'San Antonio', 'Texas', 78220.0],
       ...,
       [998, 'Tobit', 'Birt', ..., 'Waterbury', 'Connecticut', 6721.0],
       [999, 'Issiah', 'Standbrooke', ..., 'Savannah', 'Georgia',
        31405.0],
       [1000, 'Elmore', 'Malpas', ..., 'Atlanta', 'Georgia', 30386.0]],
      dtype=object)
```

```
products.values
```

```
array([[0, 1, 'Liners - Baking Cups', '$6.36', 'Skipfire'],
       [1, 2, 'Nori Sea Weed - Gold Label', '$85.74', 'Dynazzy'],
       [2, 3, 'Bar Bran Honey Nut', '$65.40', 'Ntag'],
       [3, 4, 'Soup - Campbells Beef Stew', '$68.16', 'Photojam'],
       [4, 5, 'Wine - Shiraz Wolf Blass Premium', '$87.39', 'Eare'],
       [5, 6, 'Wine - White, Riesling, Semi - Dry', '$99.22', 'Livepath'],
       [6, 7, 'Brandy - Bar', '$13.83', 'Oloo'],
       [7, 8, 'Onions - White', '$42.19', 'Oozz'],
       [8, 9, 'Lettuce - Baby Salad Greens', '$30.01', 'Meevee'],
       [9, 10, 'Sambuca - Ramazzotti', '$88.99', 'Livepath'],
       [10, 11, 'Coffee - Decafenated', '$21.47', 'Meezzy'],
       [11, 12, 'Lamb Leg - Bone - In Nz', '$42.56', 'Photofeed'],
       [12, 13, 'Lettuce - Spring Mix', '$19.90', 'Twitterworks'],
       [13, 14, 'Cookies Oatmeal Raisin', '$99.40', 'Dynabox'],
       [14, 15, 'Cookies Oatmeal Raisin', '$14.13', 'Vitz'],
       [15, 16, 'Sausage - Chorizo', '$55.45', nan],
       [16, 17, 'Glaze - Apricot', '$33.78', 'Browsezoom'],
```

```
purchases.values
```

```
array([[0, 1, datetime.datetime(2019, 1, 3, 0, 0), ..., 27, 12,
        '$568.92'],
       [1, 2, datetime.datetime(2019, 1, 3, 0, 0), ..., 28, 14,
        '$395.36'],
       [2, 3, datetime.datetime(2019, 1, 3, 0, 0), ..., 9, 17, '$510.17'],
       ...,
       [5997, 5998, '06/20/2019', ..., 19, 9, '$205.47'],
       [5998, 5999, '06/20/2019', ..., 11, 20, '$429.40'],
       [5999, 6000, '06/20/2019', ..., 57, 4, '$274.52']], dtype=object)
```

```
# .size -> .shape(1000,11) -> 1000 * 11 = 11000
customers.size # -> Total Cell in Customer DataFrame
```

```
11000
```

```
products.size
```

```
300
```

```
purchases.size
```

```
42000
```

```
# .empty [Boolean Return][T/F] -> DataFrame is empty or not
customers.empty # False
```

```
False
```

```
products.empty # False
```

```
False
```

```
purchases.empty
```

```
False
```

```
empty_df = pd.DataFrame()
print(empty_df)
print(empty_df.empty) # True
```

```
Empty DataFrame
Columns: []
Index: []
True
```

```
# .ndim -> N-Dimension
customers.ndim # DataFrame -> 2 Dimension
```

```
2
```

```
products.ndim
```

```
2
```

```
purchases.ndim
```

```
2
```

```
# .head() & .tail()
customers.head(7) # default is 'First 5'
```

	id	first_name	last_name	email	gender	street_num	street_name	street_suffix	city	state	po
0	1	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California	9
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	NaN	El Paso	Texas	
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas	7
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee	3
4	5	Mikaela	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive	NaN	NaN	8
5	6	Magdalena	Cullip	mcullip5@tiny.cc	Female	9190.0	Packers	Drive	Baltimore	Maryland	2
6	7	Marietta	Heball	mheball6@blog.com	Female	12.0	Mallory	Center	Carol Stream	Illinois	6

```
products.tail(7) # default is 'Last 5'
```

	Unnamed: 0	id	product	cost	company
53	53	54	Plate - Foam, Bread And Butter	$37.79	Flipstorm
54	54	55	Beef - Bresaola	$8.55	Digitube
55	55	56	Peach - Fresh	$20.53	Zazio
56	56	57	Muffin Hinge - 211n	$68.63	Fanoodle
57	57	58	Bar Bran Honey Nut	$73.05	Yakijo
58	58	59	Nut - Almond, Blanched, Whole	$74.28	Eazzy
59	59	60	Table Cloth 90x90 Colour	$41.22	Quinu

```
# Case Study => Web Scrap data from .screener() -> Stock Analysis [.Transpose]
products.T
```

	0	1	2	3	4	5	6	7	8	9	...	50	51
Unnamed: 0	0	1	2	3	4	5	6	7	8	9	...	50	51
id	1	2	3	4	5	6	7	8	9	10	...	51	52
product	Liners - Baking Cups	Nori Sea Weed - Gold Label	Bar Bran Honey Nut	Soup - Campbells Beef Stew	Wine - Shiraz Wolf Blass Premium	Wine - White, Riesling, Semi - Dry	Brandy - Bar	Onions - White	Lettuce - Baby Salad Greens	Sambuca - Ramazzotti	...	Ecolab - Power Fusion	Soup - Knorr, Country Bean
cost	$6.36	$85.74	$65.40	$68.16	$87.39	$99.22	$13.83	$42.19	$30.01	$88.99	...	$75.32	$98.74
company	Skipfire	Dynazzy	Ntag	Photojam	Eare	Livepath	Oloo	Oozz	Meevee	Livepath	...	Fliptune	Eimbee

5 rows × 60 columns

```
products.transpose()
```

	0	1	2	3	4	5	6	7	8	9	...	50	51
Unnamed: 0	0	1	2	3	4	5	6	7	8	9	...	50	51
id	1	2	3	4	5	6	7	8	9	10	...	51	52
product	Liners - Baking Cups	Nori Sea Weed - Gold Label	Bar Bran Honey Nut	Soup - Campbells Beef Stew	Wine - Shiraz Wolf Blass Premium	Wine - White, Riesling, Semi - Dry	Brandy - Bar	Onions - White	Lettuce - Baby Salad Greens	Sambuca - Ramazzotti	...	Ecolab - Power Fusion	Soup - Knorr, Country Bean	Th
cost	$6.36	$85.74	$65.40	$68.16	$87.39	$99.22	$13.83	$42.19	$30.01	$88.99	...	$75.32	$98.74	
company	Skipfire	Dynazzy	Ntag	Photojam	Eare	Livepath	Oloo	Oozz	Meevee	Livepath	...	Fliptune	Eimbee	To

5 rows × 60 columns

```
# products.T.T
products.transpose().transpose()
```

	Unnamed: 0	id	product	cost	company
0	0	1	Liners - Baking Cups	$6.36	Skipfire
1	1	2	Nori Sea Weed - Gold Label	$85.74	Dynazzy
2	2	3	Bar Bran Honey Nut	$65.40	Ntag
3	3	4	Soup - Campbells Beef Stew	$68.16	Photojam
4	4	5	Wine - Shiraz Wolf Blass Premium	$87.39	Eare
5	5	6	Wine - White, Riesling, Semi - Dry	$99.22	Livepath
6	6	7	Brandy - Bar	$13.83	Oloo
7	7	8	Onions - White	$42.19	Oozz
8	8	9	Lettuce - Baby Salad Greens	$30.01	Meevee

```
# Operations on Rows And Columns in DataFrame [Feature Engineering]
# Create a new column name 'Country' -> 'United States'
customers['country'] = 'United States'
customers.head()
```

ne	last_name	email	gender	street_num	street_name	street_suffix	city	state	postcode	country
in	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California	92127.0	United States
no	Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	NaN	El Paso	Texas	NaN	United States
bi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas	78220.0	United States
ke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee	38150.0	United States
la	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive	NaN	NaN	83732.0	United States

```
# Membership_status - ['Diamond','Platinum','Gold','Silver','Bronze']
membership_status = ['Diamond','Platinum','Gold','Silver','Bronze']
customers['membership_status'] = membership_status
# ValueError: Length of values (5) does not match length of index (1000)
```

```
# .loc[rows , cols]
membership_status = ['Diamond','Platinum','Gold','Silver','Bronze']
customers.loc[:4,'membership_status'] = membership_status
# Rest 995 Records are NaN
customers
```

email	gender	street_num	street_name	street_suffix	city	state	postcode	country	membership_status
@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California	92127.0	United States	Diamond
ux1@wiley.com	Male	NaN	NaN	NaN	El Paso	Texas	NaN	United States	Platinum
)symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas	78220.0	United States	Gold
ın3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee	38150.0	United States	Silver
thorpe4@51.la	NaN	NaN	2nd	Drive	NaN	NaN	83732.0	United States	Bronze

```
# 1000 records // 5 count of membership_status -> 200
# np.tile(data , repetition)
# Construct an array by repeating A the number of times given by reps.
membership_status = ['Diamond','Platinum','Gold','Silver','Bronze']
customers['Membership_Status'] = np.tile(membership_status , len(customers) // len(membership_status))
# Rest 995 Records are NaN
customers
```

r	street_num	street_name	street_suffix	city	state	postcode	country	membership_status	Membership_Status
e	1.0	Trailsway	Road	San Diego	California	92127.0	United States	Diamond	Diamond
e	NaN	NaN	NaN	El Paso	Texas	NaN	United States	Platinum	Platinum
e	4057.0	Arkansas	Circle	San Antonio	Texas	78220.0	United States	Gold	Gold
e	74.0	Debs	Point	Memphis	Tennessee	38150.0	United States	Silver	Silver
N	NaN	2nd	Drive	NaN	NaN	83732.0	United States	Bronze	Bronze
...	...	...	...	...	...	...	...	...	...

```
customers['Membership_Status'].value_counts()
```

```
Membership_Status
Diamond     200
Platinum    200
Gold        200
Silver      200
Bronze      200
Name: count, dtype: int64
```

```
# Assigning Based on Conditions
customers['membershipStatus'] = customers['state'].apply(
    lambda x : 'Gold' if x == 'Texas' else 'Silver'
)
customers
```

t_name	street_suffix	city	state	postcode	country	membership_status	Membership_Status	membershipStatus
ailsway	Road	San Diego	California	92127.0	United States	Diamond	Diamond	Silver
NaN	NaN	El Paso	Texas	NaN	United States	Platinum	Platinum	Gold
rkansas	Circle	San Antonio	Texas	78220.0	United States	Gold	Gold	Gold
Debs	Point	Memphis	Tennessee	38150.0	United States	Silver	Silver	Silver
2nd	Drive	NaN	NaN	83732.0	United States	Bronze	Bronze	Silver
...	...	...	...	...	...	...	...	...

```
# Assigning Based on Conditions
customers['membershipStatus'] = customers['state'].apply(
    lambda x : 'Gold' if (x == 'Texas') or (x == 'California') or (x == 'Georgia') else 'Silver'
)
customers
```

t_name	street_suffix	city	state	postcode	country	membership_status	Membership_Status	membershipStatus
ailsway	Road	San Diego	California	92127.0	United States	Diamond	Diamond	Gold
NaN	NaN	El Paso	Texas	NaN	United States	Platinum	Platinum	Gold
rkansas	Circle	San Antonio	Texas	78220.0	United States	Gold	Gold	Gold
Debs	Point	Memphis	Tennessee	38150.0	United States	Silver	Silver	Silver
2nd	Drive	NaN	NaN	83732.0	United States	Bronze	Bronze	Silver
...	...	...	...	...	...	...	...	...

```
customers.columns
```

```
Index(['id', 'first_name', 'last_name', 'email', 'gender', 'street_num',
       'street_name', 'street_suffix', 'city', 'state', 'postcode', 'country',
       'membership_status', 'Membership_Status', 'membershipStatus'],
      dtype='object')
```

```
customers.dtypes
```

```
id                   int64
first_name          object
last_name           object
email               object
gender              object
street_num         float64
street_name         object
street_suffix       object
city                object
state               object
postcode           float64
country             object
membership_status   object
Membership_Status   object
membershipStatus    object
dtype: object
```

```
products.columns
```

```
Index(['Unnamed: 0', 'id', 'product', 'cost', 'company'], dtype='object')
```

```
products.dtypes
```

```
Unnamed: 0     int64
id             int64
product       object
cost          object
company       object
dtype: object
```

```
# Row Insertions = INSERT Command in SQL
products.loc['60'] = [60, 61, 'Iphone 17 Pro Max', '$1599', 'Apple']
products.loc['61'] = [61, 62, 'Nokia Lumia', '$400' , 'Nokia']
products.loc['62'] = [62, 63, 'Nothing Phone 2', '$1000', 'Nothing']
products.loc['63'] = [63, 64, 'Moto G','$700','Motorola']
products.loc['64'] = [64, 65, 'Redmi 15 Pro','$800', 'Redmi']
products
```

	Unnamed: 0	id	product	cost	company
0	0	1	Liners - Baking Cups	$6.36	Skipfire
1	1	2	Nori Sea Weed - Gold Label	$85.74	Dynazzy
2	2	3	Bar Bran Honey Nut	$65.40	Ntag
3	3	4	Soup - Campbells Beef Stew	$68.16	Photojam
4	4	5	Wine - Shiraz Wolf Blass Premium	$87.39	Eare
...	...	...	...	...	...
60	60	61	Iphone 17 Pro Max	$1599	Apple
61	61	62	Nokia Lumia	$400	Nokia
62	62	63	Nothing Phone 2	$1000	Nothing
63	63	64	Moto G	$700	Motorola
64	64	65	Redmi 15 Pro	$800	Redmi

65 rows × 5 columns

```
products.loc['65'] = None
products
```

	Unnamed: 0	id	product	cost	company
0	0.0	1.0	Liners - Baking Cups	$6.36	Skipfire
1	1.0	2.0	Nori Sea Weed - Gold Label	$85.74	Dynazzy
2	2.0	3.0	Bar Bran Honey Nut	$65.40	Ntag
3	3.0	4.0	Soup - Campbells Beef Stew	$68.16	Photojam
4	4.0	5.0	Wine - Shiraz Wolf Blass Premium	$87.39	Eare
...	...	...	...	...	...
61	61.0	62.0	Nokia Lumia	$400	Nokia
62	62.0	63.0	Nothing Phone 2	$1000	Nothing
63	63.0	64.0	Moto G	$700	Motorola
64	64.0	65.0	Redmi 15 Pro	$800	Redmi
65	NaN	NaN	NaN	NaN	NaN

66 rows × 5 columns

```
# Dropping a row -> axis = 0 [row dropping][Horizontal Axis]
products.drop(index = '65' , axis = 0 , inplace = True)
products
```

	Unnamed: 0	id	product	cost	company
0	0.0	1.0	Liners - Baking Cups	$6.36	Skipfire
1	1.0	2.0	Nori Sea Weed - Gold Label	$85.74	Dynazzy
2	2.0	3.0	Bar Bran Honey Nut	$65.40	Ntag
3	3.0	4.0	Soup - Campbells Beef Stew	$68.16	Photojam
4	4.0	5.0	Wine - Shiraz Wolf Blass Premium	$87.39	Eare
...	...	...	...	...	...
60	60.0	61.0	Iphone 17 Pro Max	$1599	Apple
61	61.0	62.0	Nokia Lumia	$400	Nokia
62	62.0	63.0	Nothing Phone 2	$1000	Nothing
63	63.0	64.0	Moto G	$700	Motorola
64	64.0	65.0	Redmi 15 Pro	$800	Redmi

65 rows × 5 columns

```
products.iloc[-5:]
```

	Unnamed: 0	id	product	cost	company
55	55.0	56.0	Peach - Fresh	$20.53	Zazio
56	56.0	57.0	Muffin Hinge - 211n	$68.63	Fanoodle
57	57.0	58.0	Bar Bran Honey Nut	$73.05	Yakijo
58	58.0	59.0	Nut - Almond, Blanched, Whole	$74.28	Eazzy
59	59.0	60.0	Table Cloth 90x90 Colour	$41.22	Quinu

```
drop_slicing = products.iloc[-5:].index
drop_slicing
```

```
Index(['60', '61', '62', '63', '64'], dtype='object')
```

```
# # Dropping a row -> axis = 0 [row dropping][Horizontal Axis]
# products.drop(index = ['63' , '64'] , axis = 0 , inplace = True)
drop_slicing = products.iloc[-5:].index
products.drop(index = drop_slicing , axis = 0 , inplace = True)
products
```

	Unnamed: 0	id	product	cost	company
0	0.0	1.0	Liners - Baking Cups	$6.36	Skipfire
1	1.0	2.0	Nori Sea Weed - Gold Label	$85.74	Dynazzy
2	2.0	3.0	Bar Bran Honey Nut	$65.40	Ntag
3	3.0	4.0	Soup - Campbells Beef Stew	$68.16	Photojam
4	4.0	5.0	Wine - Shiraz Wolf Blass Premium	$87.39	Eare
5	5.0	6.0	Wine - White, Riesling, Semi - Dry	$99.22	Livepath
6	6.0	7.0	Brandy - Bar	$13.83	Oloo
7	7.0	8.0	Onions - White	$42.19	Oozz
8	8.0	9.0	Lettuce - Baby Salad Greens	$30.01	Meevee

```
# Row Insertions = INSERT Command in SQL
products.loc[60] = [60, 61, 'Iphone 17 Pro Max', '$1599', 'Apple']
products.loc[61] = [61, 62, 'Nokia Lumia', '$400' , 'Nokia']
products.loc[62] = [62, 63, 'Nothing Phone 2', '$1000', 'Nothing']
products.loc[63] = [63, 64, 'Moto G','$700','Motorola']
products.loc[64] = [64, 65, 'Redmi 15 Pro','$800', 'Redmi']
products
```

	Unnamed: 0	id	product	cost	company
0	0.0	1.0	Liners - Baking Cups	$6.36	Skipfire
1	1.0	2.0	Nori Sea Weed - Gold Label	$85.74	Dynazzy
2	2.0	3.0	Bar Bran Honey Nut	$65.40	Ntag
3	3.0	4.0	Soup - Campbells Beef Stew	$68.16	Photojam
4	4.0	5.0	Wine - Shiraz Wolf Blass Premium	$87.39	Eare
...	...	...	...	...	...
60	60.0	61.0	Iphone 17 Pro Max	$1599	Apple
61	61.0	62.0	Nokia Lumia	$400	Nokia
62	62.0	63.0	Nothing Phone 2	$1000	Nothing
63	63.0	64.0	Moto G	$700	Motorola
64	64.0	65.0	Redmi 15 Pro	$800	Redmi

```
products.iloc[-5::2].index # This will drop alternative records from last 5 rows
```

```
Index([60, 62, 64], dtype='int64')
```

```
# .between(X,Y) X & Y Both are inclusive
products[products['id'].between(61,65)]
```

	Unnamed: 0	id	product	cost	company
60	60.0	61.0	Iphone 17 Pro Max	$1599	Apple
61	61.0	62.0	Nokia Lumia	$400	Nokia
62	62.0	63.0	Nothing Phone 2	$1000	Nothing
63	63.0	64.0	Moto G	$700	Motorola
64	64.0	65.0	Redmi 15 Pro	$800	Redmi

```
idx_range = products[products['id'].between(61,65)].index
idx_range
```

```
Index([60, 61, 62, 63, 64], dtype='int64')
```

```
# Impact Analysis -> Out of 1000 records , 157 are missing [<5%] ❌❌
customers.drop(index = postcode_idx , axis = 0 , inplace = True)
customers
```

```
# Imputing the missing Values
customers['postcode'] = customers['postcode'].fillna(0).astype(int)
customers.head()
```

first_name	last_name	email	gender	street_num	street_name	street_suffix	city	state	postcode
Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California	92127
Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	NaN	El Paso	Texas	0
Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas	78220
Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee	38150
Mikaela	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive	NaN	NaN	83732

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 1000 non-null   float64
 1   first_name         1000 non-null   object
 2   last_name          1000 non-null   object
 3   email              878 non-null    object
 4   gender             957 non-null    object
 5   street_num         738 non-null    float64
 6   street_name        963 non-null    object
 7   street_suffix      963 non-null    object
 8   city               921 non-null    object
 9   state              920 non-null    object
 10  postcode           1000 non-null   int32
 11  country            1000 non-null   object
 12  membership_status  1000 non-null   object
 13  Membership_Status  1000 non-null   object
 14  membershipStatus   1000 non-null   object
dtypes: float64(2), int32(1), object(12)
memory usage: 153.4+ KB
```

```
# Calculated Columns , Add Column [Column From Examples]
customers['customer_name'] = customers['first_name'] + ' ' + customers['last_name']
customers
```

ffix	city	state	postcode	country	membership_status	Membership_Status	membershipStatus	customer_name
oad	San Diego	California	92127	United States	Diamond	Diamond	Gold	Romain Southcott
laN	El Paso	Texas	0	United States	Platinum	Platinum	Gold	Cosimo Molyneaux
rcle	San Antonio	Texas	78220	United States	Gold	Gold	Gold	Bambi Westrip
oint	Memphis	Tennessee	38150	United States	Silver	Silver	Silver	Roarke Pankettman
rive	NaN	NaN	83732	United States	Bronze	Bronze	Silver	Mikaela Althorpe
...	...	...	...	...	...	...	...	...

```
products[['company','product','cost']]
```

	company	product	cost
0	Skipfire	Liners - Baking Cups	$6.36
1	Dynazzy	Nori Sea Weed - Gold Label	$85.74
2	Ntag	Bar Bran Honey Nut	$65.40
3	Photojam	Soup - Campbells Beef Stew	$68.16
4	Eare	Wine - Shiraz Wolf Blass Premium	$87.39
5	Livepath	Wine - White, Riesling, Semi - Dry	$99.22
6	Oloo	Brandy - Bar	$13.83
7	Oozz	Onions - White	$42.19

```
customers.columns
```

```
Index(['id', 'first_name', 'last_name', 'email', 'gender', 'street_num',
       'street_name', 'street_suffix', 'city', 'state', 'postcode', 'country',
       'membership_status', 'Membership_Status', 'membershipStatus',
       'customer_name'],
      dtype='object')
```

```
customer_col_drop = customers.iloc[:,-5:].columns
customer_col_drop
```

```
Index(['country', 'membership_status', 'Membership_Status', 'membershipStatus',
       'customer_name'],
      dtype='object')
```

```
# customers.drop(columns = ['country','membership_status','Membership_Status',
#                           'membershipStatus','customer_name'] , axis = 1 , inplace = True)
customers.drop(columns = customer_col_drop , axis = 1 , inplace = True)
customers
```

ast_name	email	gender	street_num	street_name	street_suffix	city	state	postcode
Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California	92127
Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	NaN	El Paso	Texas	0
Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas	78220
nkettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee	38150
Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive	NaN	NaN	83732
...	...	...	...	...	...	...	...	...
Alman	malmanrn@cornell.edu	Female	0.0	Thompson	Way	Reading	Pennsylvania	19610
Heckle	whecklero@fc2.com	Female	701.0	Rowland	Hill	Indianapolis	Indiana	46278
Birt	NaN	Male	2.0	Basil	Road	Waterbury	Connecticut	6721

```python
# Renaming the row index
index_rename = {
    0 : 'FirstCustomers',
    1 : 'SecondCustomers',
    2 : 'ThirdCustomers',
    3 : 'FourthCustomers',
    4 : 'FifthCustomers',
}
# customers.rename(index = index_rename , inplace = True)
customers = customers.rename(index = index_rename)
customers
```

	id	first_name	last_name	email	gender	street_num	street_name	street_suff
FirstCustomers	1.0	Romain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Ro
SecondCustomers	2.0	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	Na
ThirdCustomers	3.0	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Cir
FourthCustomers	4.0	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Po
FifthCustomers	5.0	Mikaela	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Dri
...	...	...	...	...	...	...	...	
995	996.0	Merrili	Alman	malmanrn@cornell.edu	Female	0.0	Thompson	W
996	997.0	Winonah	Heckle	whecklero@fc2.com	Female	701.0	Rowland	H
997	998.0	Tobit	Birt	NaN	Male	2.0	Basil	Ro
998	999.0	Issiah	Standbrooke	istandbrookerq@yellowpages.com	Male	NaN	Kenwood	Dri
999	1000.0	Elmore	Malpas	NaN	Male	205.0	Farwell	Pa

1000 rows × 11 columns

```python
# Renaming the column headers
col_rename = {
    'street_num' : 'street_number',
    'street_name' : 'street_address',
}
# customers.rename(columns = col_rename , inplace = True)
customers = customers.rename(columns = col_rename)
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
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	NaN	E
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	Ar
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Mei
4	5	Mikaela	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive	
...	...	...	...	...	...	...	...	...	
995	996	Merrili	Alman	malmanrn@cornell.edu	Female	0.0	Thompson	Way	Re
996	997	Winonah	Heckle	whecklero@fc2.com	Female	701.0	Rowland	Hill	Indian
997	998	Tobit	Birt	NaN	Male	2.0	Basil	Road	Wate
998	999	Issiah	Standbrooke	istandbrookerq@yellowpages.com	Male	NaN	Kenwood	Drive	Sava
999	1000	Elmore	Malpas	NaN	Male	205.0	Farwell	Park	A

1000 rows × 11 columns

```
# SELECT DISTINCT FROM Tables -> Unique Values
customers['state'].shape
```

```
(1000,)
```

```
customers['state'].ndim # Series
```

```
1
```

```
customers['state'].unique()
```

```
array(['California', 'Texas', 'Tennessee', nan, 'Maryland', 'Illinois',
       'Missouri', 'Alaska', 'Louisiana', 'Florida', 'New York',
       'District of Columbia', 'South Carolina', 'Washington', 'Nevada',
       'Indiana', 'Pennsylvania', 'Georgia', 'Virginia', 'Michigan',
       'Iowa', 'Kentucky', 'Oregon', 'North Carolina', 'Ohio', 'Kansas',
       'West Virginia', 'Oklahoma', 'North Dakota', 'Arizona', 'Utah',
       'Minnesota', 'Wisconsin', 'New Jersey', 'Montana', 'New Hampshire',
       'Colorado', 'Massachusetts', 'Alabama', 'Delaware', 'Mississippi',
       'Idaho', 'Arkansas', 'Hawaii', 'Nebraska', 'Connecticut',
       'Rhode Island', 'New Mexico', 'Maine'], dtype=object)
```

```
len(customers['state'].unique())
```

```
49
```

```
customers['state'].nunique() # Count of Unique States
```

```
48
```

```
customers['gender'].unique()
```

```
array(['Male', 'Female', nan], dtype=object)
```

```
customers['gender'].nunique() # 2  [Exclude 'NaN' values]
```

```
2
```

```
# Value_Counts() [Freq]
products['company'].value_counts().head(5)
```

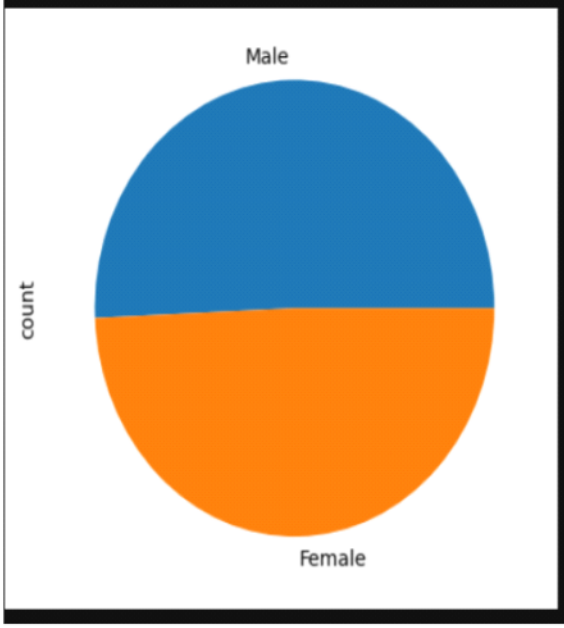
```
company
Vitz          2
Ntag          2
Livepath      2
Cogibox       2
Browsezoom    2
Name: count, dtype: int64
```

```
customers['gender'].value_counts()
```

```
gender
Male      485
Female    472
Name: count, dtype: int64
```

```
customers['gender'].value_counts().plot(kind = 'pie')
```

```
<Axes: ylabel='count'>
```




```
purchases.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    6000 non-null   int64 
 1   id            6000 non-null   int64 
 2   purch_date    6000 non-null   object
 3   customer_num  6000 non-null   int64 
 4   product_num   6000 non-null   int64 
 5   amount        6000 non-null   int64 
 6   paid          6000 non-null   object
dtypes: int64(5), object(2)
memory usage: 328.3+ KB
```

```
purchases['paid'] = purchases['paid'].astype('str').str.replace('[$,]' , '' , regex = True).astype('float')
purchases.head()
```

	Unnamed: 0	id	purch_date	customer_num	product_num	amount	paid
0	0	1	2019-01-03 00:00:00	823	27	12	568.92
1	1	2	2019-01-03 00:00:00	606	28	14	395.36
2	2	3	2019-01-03 00:00:00	955	9	17	510.17
3	3	4	2019-01-03 00:00:00	577	19	3	68.49
4	4	5	2019-01-03 00:00:00	429	8	18	759.42

```
purchases.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    6000 non-null   int64  
 1   id            6000 non-null   int64  
 2   purch_date    6000 non-null   object 
 3   customer_num  6000 non-null   int64  
 4   product_num   6000 non-null   int64  
 5   amount        6000 non-null   int64  
 6   paid          6000 non-null   float64
dtypes: float64(1), int64(5), object(1)
memory usage: 328.3+ KB
```

```
# Trend Axis [Time Intelligence] [Purchase_data] [Object - DateTime]
pd.to_datetime(purchases['purch_date']) # YYYY-MM-DD
```

```
0      2019-01-03
1      2019-01-03
2      2019-01-03
3      2019-01-03
4      2019-01-03
          ...
5995   2019-06-20
5996   2019-06-20
5997   2019-06-20
5998   2019-06-20
5999   2019-06-20
```

```
purchases['purch_date'] = pd.to_datetime(purchases['purch_date'])
purchases
```

	Unnamed: 0	id	purch_date	customer_num	product_num	amount	paid
0	0	1	2019-01-03	823	27	12	568.92
1	1	2	2019-01-03	606	28	14	395.36
2	2	3	2019-01-03	955	9	17	510.17
3	3	4	2019-01-03	577	19	3	68.49
4	4	5	2019-01-03	429	8	18	759.42
...	...	...	...	...	...	...	...
5995	5995	5996	2019-06-20	893	33	5	411.10
5996	5996	5997	2019-06-20	566	23	11	178.97
5997	5997	5998	2019-06-20	114	19	9	205.47
5998	5998	5999	2019-06-20	404	11	20	429.40
5999	5999	6000	2019-06-20	88	57	4	274.52

6000 rows × 7 columns

```
purchases.drop(columns = 'Unnamed: 0', axis = 1 , inplace = True)
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

```
purchases.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   id            6000 non-null   int64         
 1   purch_date    6000 non-null   datetime64[ns]
 2   customer_num  6000 non-null   int64         
 3   product_num   6000 non-null   int64         
 4   amount        6000 non-null   int64         
 5   paid          6000 non-null   float64       
dtypes: datetime64[ns](1), float64(1), int64(4)
memory usage: 281.4 KB
```

```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60 entries, 0 to 59
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  60 non-null     float64
 1   id          60 non-null     float64
 2   product     60 non-null     object 
 3   cost        60 non-null     object 
 4   company     55 non-null     object 
dtypes: float64(2), object(3)
memory usage: 2.8+ KB
```

```
products['cost'] = products['cost'].astype('str').str.replace('[$,]' , '' , regex = True).astype('float')
products.head()
```

	Unnamed: 0	id	product	cost	company
0	0	1	Liners - Baking Cups	6.36	Skipfire
1	1	2	Nori Sea Weed - Gold Label	85.74	Dynazzy
2	2	3	Bar Bran Honey Nut	65.40	Ntag
3	3	4	Soup - Campbells Beef Stew	68.16	Photojam
4	4	5	Wine - Shiraz Wolf Blass Premium	87.39	Eare

```
products.drop(columns = 'Unnamed: 0', axis = 1 , inplace = True)
products.head()
```

	id	product	cost	company
0	1	Liners - Baking Cups	6.36	Skipfire
1	2	Nori Sea Weed - Gold Label	85.74	Dynazzy
2	3	Bar Bran Honey Nut	65.40	Ntag
3	4	Soup - Campbells Beef Stew	68.16	Photojam
4	5	Wine - Shiraz Wolf Blass Premium	87.39	Eare

```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60 entries, 0 to 59
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   id       60 non-null     float64
 1   product  60 non-null     object 
 2   cost     60 non-null     float64
 3   company  55 non-null     object 
dtypes: float64(2), object(2)
memory usage: 2.3+ KB
```

```
products['company'].isna().sum()
```

```
5
```

```
products[products['company'].isna()]
```

	id	product	cost	company
15	16.0	Sausage - Chorizo	55.45	NaN
21	22.0	Scotch - Queen Anne	60.26	NaN
26	27.0	Spaghetti Squash	47.41	NaN
27	28.0	Wine - Niagara,vqa Reisling	28.24	NaN
45	46.0	Aromat Spice / Seasoning	61.34	NaN

```
# Replace NaN value with Unknown , Fill Down [.ffill()] or Fill Up[.bfill()]
# products.fillna({'company' : 'Unknown'} , inplace = True)
products['company'] = products['company'].ffill()
# products['company'] = products['company'].bfill()
products
```

	id	product	cost	company
0	1	Liners - Baking Cups	6.36	Skipfire
1	2	Nori Sea Weed - Gold Label	85.74	Dynazzy

0	1	Liners - Baking Cups	6.36	Skipfire
1	2	Nori Sea Weed - Gold Label	85.74	Dynazzy
2	3	Bar Bran Honey Nut	65.40	Ntag
3	4	Soup - Campbells Beef Stew	68.16	Photojam
4	5	Wine - Shiraz Wolf Blass Premium	87.39	Eare
5	6	Wine - White, Riesling, Semi - Dry	99.22	Livepath
6	7	Brandy - Bar	13.83	Oloo
7	8	Onions - White	42.19	Oozz
8	9	Lettuce - Baby Salad Greens	30.01	Meevee

```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60 entries, 0 to 59
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       60 non-null     float64
 1   product  60 non-null     object
 2   cost     60 non-null     float64
 3   company  60 non-null     object
dtypes: float64(2), object(2)
memory usage: 2.3+ KB
```

```
products.groupby('company')['cost'].sum().head()
```

```
company
Aibox          56.33
Babbleopia     63.98
Brainsphere    22.83
Browsezoom     82.10
Cogibox       102.27
Name: cost, dtype: float64
```

```
company_revenue = products.groupby('company')['cost'].sum().reset_index()
company_revenue
```

	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

```
# Top 7 Company based on Revenue
company_revenue.sort_values(by = 'cost' , ascending = False).head(7)
```

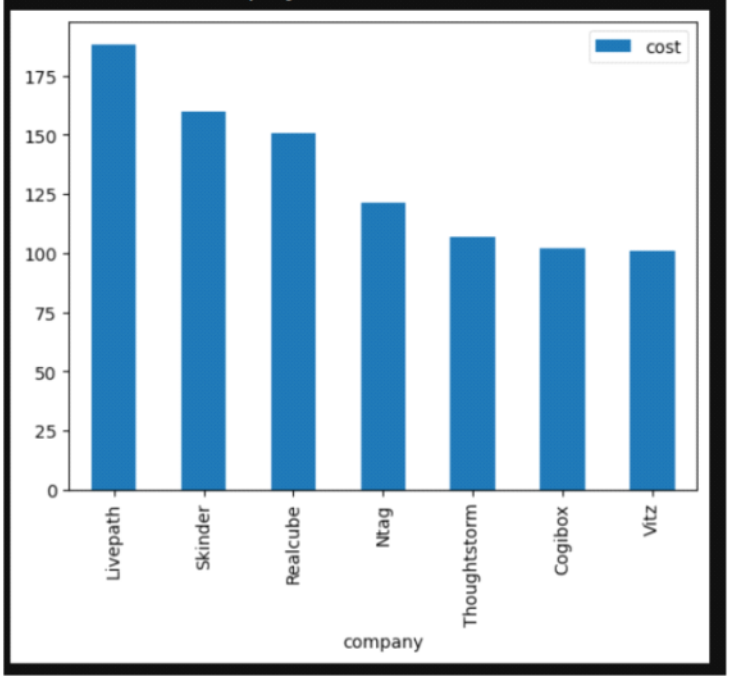
	company	cost
19	Livepath	188.21
37	Skinder	159.85
34	Realcube	150.74
25	Ntag	121.17
40	Thoughtstorm	107.11
4	Cogibox	102.27
44	Vitz	100.84

```
df_plot = company_revenue.sort_values(by = 'cost' , ascending = False).head(7).set_index('company') # Series
df_plot
```

	cost
company	
Livepath	188.21
Skinder	159.85
Realcube	150.74
Ntag	121.17
Thoughtstorm	107.11
Cogibox	102.27
Vitz	100.84

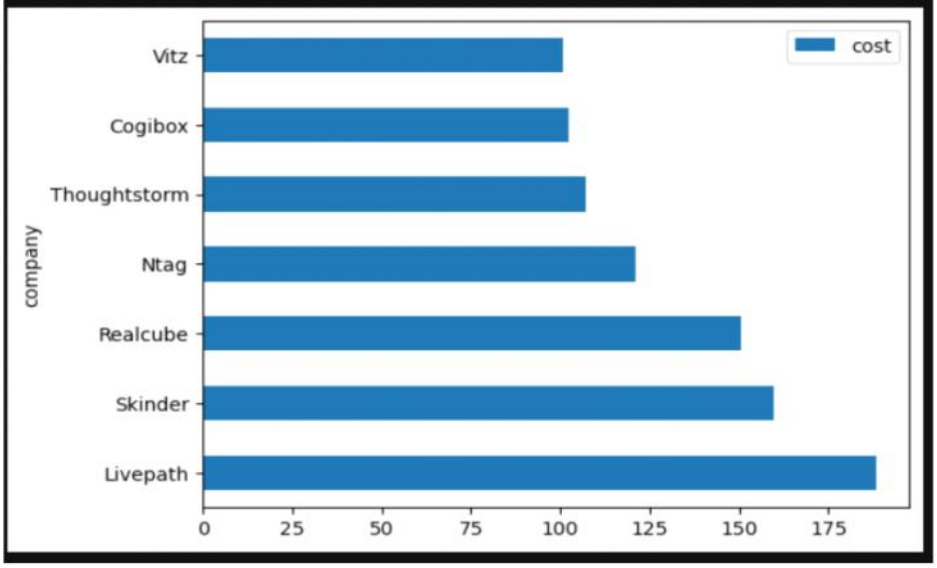
```
df_plot.plot(kind = 'bar')
```

```
<Axes: xlabel='company'>
```



```
df_plot.plot(kind = 'barh')
```

```
<Axes: ylabel='company'>
```




```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, FirstCustomers to 999
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              1000 non-null   float64
 1   first_name      1000 non-null   object
 2   last_name       1000 non-null   object
 3   email           878 non-null    object
 4   gender          957 non-null    object
 5   street_number   738 non-null    float64
 6   street_address  963 non-null    object
 7   street_suffix   963 non-null    object
 8   city            921 non-null    object
 9   state           920 non-null    object
 10  postcode        1000 non-null   int32
dtypes: float64(2), int32(1), object(8)
memory usage: 122.1+ KB
```

```
# 'email' -> Unique
customers.fillna({'email' : 'Unknown'} , inplace = True)
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	California
1	2	Cosimo	Molyneaux	cmolyneaux1@wiley.com	Male	NaN	NaN	NaN	El Paso	Texas
2	3	Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas
3	4	Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee
4	5	Mikaela	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive	NaN	NaN

```
customers['gender'].isnull().sum()
```

```
43
```

```
customers[customers['gender'].isnull()].head()
```

	id	first_name	last_name	email	gender	street_number	street_address	street_suffix
FifthCustomers	5.0	Mikaela	Althorpe	malthorpe4@51.la	NaN	NaN	2nd	Drive
7	8.0	Tine	McSperrin	tmcsperrin7@statcounter.com	NaN	530.0	Erie	Plaza
55	56.0	Shandy	Ditzel	sditzel1j@google.com	NaN	6051.0	Nova	Circle
64	65.0	Gamaliel	Cottel	gcottel1s@merriam-webster.com	NaN	28.0	Laurel	Place
87	88.0	Baxie	Bridger	bbridger2f@1und1.de	NaN	79016.0	Havey	Circle

```
customers['gender'].mode()
```

```
0    Male
Name: gender, dtype: object
```

```
customers['gender'].mode()[0]
```

```
'Male'
```

```
type(customers['gender'].mode()[0])
```

```
str
```

```
customers['gender'].value_counts()
```

```
gender
Male      485
Female    472
Name: count, dtype: int64
```

```
# customers.fillna({'gender' : 'Male'} , inplace = True)
customers.fillna({'gender' : customers['gender'].mode()[0]} , inplace = True)
# customers['gender'] = customers['gender'].ffill()
# customers['gender'] = customers['gender'].bfill()
customers.head()
```

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

```
customers['street_number'] = customers['street_number'].fillna(0)
customers['street_address'] = customers['street_address'].ffill()
customers['street_suffix'] = customers['street_suffix'].ffill()
customers['city'] = customers['city'].bfill()
customers['state'] = customers['state'].bfill()
customers.head()
```

name	last_name	email	gender	street_number	street_address	street_suffix	city	state	postcode
omain	Southcott	rsouthcott0@clickbank.net	Male	1.0	Trailsway	Road	San Diego	California	92127
osimo	Molyneaux	cmolyneaux1@wiley.com	Male	0.0	Trailsway	Road	El Paso	Texas	0
Bambi	Westrip	bwestrip2@symantec.com	Female	4057.0	Arkansas	Circle	San Antonio	Texas	78220
Roarke	Pankettman	rpankettman3@wiley.com	Male	74.0	Debs	Point	Memphis	Tennessee	38150
likaela	Althorpe	malthorpe4@51.la	Male	0.0	2nd	Drive	Baltimore	Maryland	83732

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              1000 non-null   int64
 1   first_name      1000 non-null   object
 2   last_name       1000 non-null   object
 3   email           1000 non-null   object
 4   gender          1000 non-null   object
 5   street_number   1000 non-null   float64
 6   street_address  1000 non-null   object
 7   street_suffix   1000 non-null   object
 8   city            1000 non-null   object
 9   state           1000 non-null   object
 10  postcode        1000 non-null   int32
dtypes: float64(1), int32(1), int64(1), object(8)
memory usage: 82.2+ KB
```

```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       60 non-null     int64
 1   product  60 non-null     object
 2   cost     60 non-null     float64
 3   company  60 non-null     object
dtypes: float64(1), int64(1), object(2)
memory usage: 2.0+ KB
```

```
purchases.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            6000 non-null   int64
 1   purch_date    6000 non-null   datetime64[ns]
 2   customer_num  6000 non-null   int64
 3   product_num   6000 non-null   int64
 4   amount        6000 non-null   int64
 5   paid          6000 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(4)
memory usage: 281.4 KB
```

```
-> Filling the Missing Values
-> Handling the Missing Values() -> 'NaN','NaT','None','<NA>'
1. Identify them , isnull(),isna(),notnull() [Boolean Return] -> Filter
customers[customers['gender'].isna()]
customers[customers['gender'].isnull()]

2. Strategies to handle the missing values
- fillna() - Object / Categorical Columns ['Unknown']
- fillna() - Numerical Columns / Continuous [0 or statistical_analysis]
- .bfill() - Backward Filling [Fill Up] [Bottom Up]
- .ffill() - Forward Filling [Fill Down] [Top To Bottom]
- .dropna() - Removing the NaN Values [axis =0(rows) | axis = 1(cols)] [Impact Analysis]
- .interpolate() - [Linear , Polynomial , degree 2] = [0 2 4 _ 8 10 _ 14 _ 18]
```