In [50]:

```
import numpy as np
import pandas as pd
```

In [51]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [52]:

```
data=r"C:\Users\rahee\Downloads\train.csv\train.csv"
df=pd.read_csv(data)
```

In [53]:

```
#Exploratory Data Analysis
df.head()
```

Out[53]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN

In [54]:

```
type(df)
```

Out[54]:

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

In [55]:

```
#checking the shape of the data
df.shape #(rows,coloumns)
```

Out[55]:

```
(550068, 12)
```

In [56]:

```
#viewing forst five rows of the dataframe
df.head()
```

Out[56]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN

In [57]:

```
#viewing consise summary of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

In [58]:
```
#handling missing values
#checking missing value
df.isnull().sum()
# we can see there are 2 category with missing values
```

Out[58]:
```
User_ID                            0
Product_ID                         0
Gender                             0
Age                                0
Occupation                         0
City_Category                      0
Stay_In_Current_City_Years         0
Marital_Status                     0
Product_Category_1                 0
Product_Category_2            173638
Product_Category_3            383247
Purchase                           0
dtype: int64
```

In [59]:
```
#detection NA values using isna() and notna()
df.isna().sum()
#we can se all the null values are encoded with NaN
```

```
Out[59]: User_ID                            0
         Product_ID                         0
         Gender                             0
         Age                                0
         Occupation                         0
         City_Category                      0
         Stay_In_Current_City_Years         0
         Marital_Status                     0
         Product_Category_1                 0
         Product_Category_2            173638
         Product_Category_3            383247
         Purchase                           0
         dtype: int64
```

Handling missing numerical values

- Drop missing value with dropna() method
- Fill missing values with zeros
- Fill missing value with test statistics
- Fill missing values backward or forward

> In this section we will use forward or backward method, which can be done using pad or fill and bfill or backfill options

```
In [61]: df=df.fillna(method='pad')
```

```
In [62]: df.isnull().sum()
```

Out[62]:

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            1
Product_Category_3            1
Purchase                      0
dtype: int64
```

In [63]:

```
#as we have only one missing value and we use forwar fill we can check this by using head method .
df.head()
```

Out[63]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	6.0	14.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	14.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	14.0	14.0

In [64]:

```
#as the elements Nan the pad method doesnt work, we should use bfill or back fill
df=df.fillna(method='backfill')
```

In [65]:

```
df.isnull().sum()
```

Out[65]:

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
dtype: int64
```

Chechking with ASSERT statement

- It checks the valeus programmatically, and returns if any 0 are there
- It also checks the programs running smoothly or not
- Assert statement will return nothing if the value being tested is true and will throw an AssertionError if the value is false

In [67]:

```python
#assert that there are no missing values in the dataframe
assert pd.notnull(df).all().all()
```

Indexing and Slicing in Pandas

In [69]:

```python
#make a copy of data frame
df1=df.copy()
```

In [70]:

```python
df1.loc[0]
```

Out[70]:

```
User_ID                       1000001
Product_ID                  P00069042
Gender                              F
Age                              0-17
Occupation                         10
City_Category                       A
Stay_In_Current_City_Years          2
Marital_Status                      0
Product_Category_1                  3
Product_Category_2                6.0
Product_Category_3               14.0
Purchase                         8370
Name: 0, dtype: object
```

In [71]:

```
#select the first 5 rows for specific columns
df1.loc[:,'Purchase'].head()
```

Out[71]:

```
0     8370
1    15200
2     1422
3     1057
4     7969
Name: Purchase, dtype: int64
```

In [72]:

```
# some examople of loc()
df1.loc[:,['Age','Occupation']].head()
```

Out[72]:

	Age	Occupation
0	0-17	10
1	0-17	10
2	0-17	10
3	0-17	10
4	55+	16

In [73]:

```
df1.loc[0:4]
```

Out[73]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	6.0	14.0
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	6.0	14.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	14.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	14.0	14.0

In [74]:

```
df1.head()
```

Out[74]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	6.0	14.0
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	6.0	14.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	14.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	14.0	14.0

In [75]:

```
# integer position based indexing using .iloc indexer
# rows selection using .iloc indexer
df1.iloc[0]
```

```
Out[75]: User_ID                       1000001
         Product_ID                  P00069042
         Gender                              F
         Age                              0-17
         Occupation                         10
         City_Category                       A
         Stay_In_Current_City_Years          2
         Marital_Status                      0
         Product_Category_1                  3
         Product_Category_2                6.0
         Product_Category_3               14.0
         Purchase                         8370
         Name: 0, dtype: object
```

```
In [76]: df1.iloc[1] #second row selection
```

```
Out[76]: User_ID                       1000001
         Product_ID                  P00248942
         Gender                              F
         Age                              0-17
         Occupation                         10
         City_Category                       A
         Stay_In_Current_City_Years          2
         Marital_Status                      0
         Product_Category_1                  1
         Product_Category_2                6.0
         Product_Category_3               14.0
         Purchase                        15200
         Name: 1, dtype: object
```

```
In [77]: df1.iloc[-1] # last row selection
```

Out[77]:

```
User_ID                                 1006039
Product_ID                            P00371644
Gender                                        F
Age                                       46-50
Occupation                                    0
City_Category                                 B
Stay_In_Current_City_Years                   4+
Marital_Status                                1
Product_Category_1                           20
Product_Category_2                          2.0
Product_Category_3                         11.0
Purchase                                    490
Name: 550067, dtype: object
```

In [78]:

```
df1.iloc[-2]# secondlast row
```

Out[78]:

```
User_ID                                 1006038
Product_ID                            P00375436
Gender                                        F
Age                                         55+
Occupation                                    1
City_Category                                 C
Stay_In_Current_City_Years                    2
Marital_Status                                0
Product_Category_1                           20
Product_Category_2                          2.0
Product_Category_3                         11.0
Purchase                                    365
Name: 550066, dtype: object
```

In [79]:

```
# select first row of dataframe
df1.iloc[0]
```

```
Out[79]: User_ID                          1000001
         Product_ID                     P00069042
         Gender                                 F
         Age                                 0-17
         Occupation                            10
         City_Category                          A
         Stay_In_Current_City_Years             2
         Marital_Status                         0
         Product_Category_1                     3
         Product_Category_2                   6.0
         Product_Category_3                  14.0
         Purchase                            8370
         Name: 0, dtype: object
```

```
In [80]: # column selection
         df1.iloc[:,0]
```

```
Out[80]: 0         1000001
         1         1000001
         2         1000001
         3         1000001
         4         1000002
                    ...
         550063    1006033
         550064    1006035
         550065    1006036
         550066    1006038
         550067    1006039
         Name: User_ID, Length: 550068, dtype: int64
```

```
In [81]: df1.iloc[:,1] # second column selection
```

Out[81]:

```
0         P00069042
1         P00248942
2         P00087842
3         P00085442
4         P00285442
            ...    
550063    P00372445
550064    P00375436
550065    P00375436
550066    P00375436
550067    P00371644
Name: Product_ID, Length: 550068, dtype: object
```

In [82]:

```
df1.iloc[:,-1] # last column selection
```

Out[82]:

```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...  
550063      368
550064      371
550065      137
550066      365
550067      490
Name: Purchase, Length: 550068, dtype: int64
```

In [83]:

```
#Multiple columns and row selection using .iloc[]
df1.iloc[0:5]
```

Out[83]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	6.0	14.0
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	6.0	14.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	14.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	14.0	14.0

In [84]:

```
df1.iloc[:,0:5] # selecting all rows and first five columns
```

Out[84]:

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 5 columns

In [85]:

```
df.iloc[[0,4,9],[0,3,6]] # custom selection of rows and columns
```

Out[85]:

	User_ID	Age	Stay_In_Current_City_Years
0	1000001	0-17	2
4	1000002	55+	4+
9	1000005	26-35	1

In [86]:

```
df1.iloc[0:5,5:8]
```

Out[86]:

	City_Category	Stay_In_Current_City_Years	Marital_Status
0	A	2	0
1	A	2	0
2	A	2	0
3	A	2	0
4	C	4+	0

In [87]:

```
# Indexing using idxmax() and idxmin() used to index the first occurence of max and min values

#geting index of first occurence of maximum Purchase value
df1['Purchase'].idxmax()
```

Out[87]:

```
87440
```

In [88]:

```
# max value with the row
df1.loc[df1['Purchase'].idxmax()]
```

Out[88]:

```
User_ID                          1001474
Product_ID                     P00052842
Gender                                 M
Age                                26-35
Occupation                             4
City_Category                          A
Stay_In_Current_City_Years             2
Marital_Status                         1
Product_Category_1                    10
Product_Category_2                  15.0
Product_Category_3                   8.0
Purchase                           23961
Name: 87440, dtype: object
```

In [89]:

```
#Indexing a single value with at() and iat()
#get values at 1st row and Purchase column pair
df1.at[1,'Purchase']
```

Out[89]:

```
15200
```

```
In [90]: # get value at 1st and 11th column pair
         df1.iat[1,11]
```

```
Out[90]: 15200
```

```
In [91]: #Boolean Indexing in Pandas
         df2=df.copy()
```

```
In [92]: df2.head()
```

Out[92]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	6.0	14.0
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	6.0	14.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	14.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	14.0	14.0

```
In [93]: # get the purchase amount with a given user_id and product_id
         df2.loc[((df2['User_ID'] == 1000001) & (df2['Product_ID'] == 'P00069042')), 'Purchase']
```

```
Out[93]: 0    8370
         Name: Purchase, dtype: int64
```

```
In [94]: #Indexing using isin()
         values=[1000001,'P00069042','F',0-17,10,'A',2,0,3,6,14,8370]
         df2_indexed=df2.isin(values)
         df2_indexed.head(10)
```

Out[94]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	True	True	True	False	True	True	False	True	True	True	True
1	True	False	True	False	True	True	False	True	False	True	True
2	True	False	True	False	True	True	False	True	False	True	True
3	True	False	True	False	True	True	False	True	False	True	True
4	False	False	False	False	False	False	False	True	False	True	True
5	False	False	False	False	False	True	False	True	False	True	True
6	False	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	True	False	False	False	False	False

In [95]:

```
# we can combine df isin() with all() and any() to quickly select and subset data to meet te give criteria
row_mask=df2.isin(values).any(1)

df[row_mask].head()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[95], line 2
      1 # we can combine df isin() with all() and any() to quickly select and subset data to meet te give criteria
----> 2 row_mask=df2.isin(values).any(1)
      4 df[row_mask].head()

TypeError: DataFrame.any() takes 1 positional argument but 2 were given
```

In [ ]:

```
# where() method and masking
```

In [102…

```
df2_where=df2.where(df2==0)
(df2_where).head(10)
```

Out[102...

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [104...

```
#Indexing with query() method
df2.query('(Product_Category_1 > Product_Category_2) & (Product_Category_2 > Product_Category_3)')
```

Out[104…

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Catego
165	1000033	P00111742	M	46-50	3	A	1	1	15	8.0	
304	1000053	P00117542	M	26-35	0	B	1	0	18	16.0	
351	1000058	P00288642	M	26-35	2	B	3	0	16	14.0	
387	1000062	P00087242	F	36-45	3	A	1	0	14	12.0	
724	1000137	P00124642	F	46-50	6	C	4+	1	16	14.0	
...	...	...	...	...	...	...	...	...	...	...	
545338	1005954	P00327342	M	46-50	11	A	2	1	16	11.0	
545339	1005954	P00087842	M	46-50	11	A	2	1	12	11.0	
545461	1005972	P00255842	F	26-35	20	B	0	0	16	11.0	
545747	1006016	P00058642	M	46-50	1	B	1	1	18	14.0	
545896	1006037	P00183142	F	46-50	1	C	4+	0	15	14.0	

2278 rows × 12 columns

Indexing and Reindexing in Pandas

In [106…

```
# let's create a new dataframe

food = pd.DataFrame({'Place':['Home', 'Home', 'Hotel', 'Hotel'],
                   'Time': ['Lunch', 'Dinner', 'Lunch', 'Dinner'],
                   'Food':['Soup', 'Rice', 'Soup', 'Chapati'],
                   'Price($)':[10, 20, 30, 40]})

food
```

Out[106…

	Place	Time	Food	Price($)
0	Home	Lunch	Soup	10
1	Home	Dinner	Rice	20
2	Hotel	Lunch	Soup	30
3	Hotel	Dinner	Chapati	40

In [108…

```
food_indexed1=food.set_index('Place')
food_indexed1
```

Out[108…

	Time	Food	Price($)
Place			
Home	Lunch	Soup	10
Home	Dinner	Rice	20
Hotel	Lunch	Soup	30
Hotel	Dinner	Chapati	40

In [109…

```
food_indexed2=food.set_index(['Place','Time'])
food_indexed2
```

Out[109...

		Food	Price($)
Place	**Time**		
Home	**Lunch**	Soup	10
	Dinner	Rice	20
Hotel	**Lunch**	Soup	30
	Dinner	Chapati	40

Reset the Index

In [111...

```
food_indexed2.reset_index()
```

Out[111...

	Place	Time	Food	Price($)
0	Home	Lunch	Soup	10
1	Home	Dinner	Rice	20
2	Hotel	Lunch	Soup	30
3	Hotel	Dinner	Chapati	40

Multiindex or Advanced Indexing

- Hierarchical Indexing or Multi-Index

In [113...

```
sales=pd.DataFrame([['books','online', 200, 50],['books','retail', 250, 75],
                    ['toys','online', 100, 20],['toys','retail', 140, 30],
                    ['watches','online', 500, 100],['watches','retail', 600, 150],
                    ['computers','online', 1000, 200],['computers','retail', 1200, 300],
                    ['laptops','online', 1100, 400],['laptops','retail', 1400, 500],
                    ['smartphones','online', 600, 200],['smartphones','retail', 800, 250]],
                    columns=['Items', 'Mode', 'Price', 'Profit'])
```

```
sales
```

Out[113…

	Items	Mode	Price	Profit
0	books	online	200	50
1	books	retail	250	75
2	toys	online	100	20
3	toys	retail	140	30
4	watches	online	500	100
5	watches	retail	600	150
6	computers	online	1000	200
7	computers	retail	1200	300
8	laptops	online	1100	400
9	laptops	retail	1400	500
10	smartphones	online	600	200
11	smartphones	retail	800	250

In [114…

```
# hierarchical index
sales1=sales.set_index(['Items','Mode'])
sales1
```

Out[114…

		Price	Profit
Items	**Mode**		
books	**online**	200	50
	retail	250	75
toys	**online**	100	20
	retail	140	30
watches	**online**	500	100
	retail	600	150
computers	**online**	1000	200
	retail	1200	300
laptops	**online**	1100	400
	retail	1400	500
smartphones	**online**	600	200
	retail	800	250

In [115…

```
#view index
sales1.index
```

Out[115…

```
MultiIndex([(      'books', 'online'),
            (      'books', 'retail'),
            (       'toys', 'online'),
            (       'toys', 'retail'),
            (    'watches', 'online'),
            (    'watches', 'retail'),
            (  'computers', 'online'),
            (  'computers', 'retail'),
            (    'laptops', 'online'),
            (    'laptops', 'retail'),
            ('smartphones', 'online'),
            ('smartphones', 'retail')],
           names=['Items', 'Mode'])
```

In [116…

```
# swapping the column in heirarchical index
sales2=sales1.swaplevel('Mode','Items')
sales2
```

Out[116…

		Price	Profit
Mode	**Items**		
online	**books**	200	50
retail	**books**	250	75
online	**toys**	100	20
retail	**toys**	140	30
online	**watches**	500	100
retail	**watches**	600	150
online	**computers**	1000	200
retail	**computers**	1200	300
online	**laptops**	1100	400
retail	**laptops**	1400	500
online	**smartphones**	600	200
retail	**smartphones**	800	250

Sorting in Pandas

Sorting by label

- we use sort_index() method to sort the object by labels
- DataFrame can be sorted by passing the axis arguments and the order of sorting.

In [119…

```
# sort the dataframe df2 by label
df2.sort_index()
```

```
Out[119…
        User_ID  Product_ID  Gender  Age    Occupation  City_Category  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  Product_Category_2  Product_Catego
0       1000001  P00069042   F       0-17   10          A              2                           0               3                   6.0
1       1000001  P00248942   F       0-17   10          A              2                           0               1                   6.0
2       1000001  P00087842   F       0-17   10          A              2                           0               12                  6.0
3       1000001  P00085442   F       0-17   10          A              2                           0               12                  14.0
4       1000002  P00285442   M       55+    16          C              4+                          0               8                   14.0
...     ...      ...         ...     ...    ...         ...            ...                         ...             ...                 ...
550063  1006033  P00372445   M       51-55  13          B              1                           1               20                  2.0
550064  1006035  P00375436   F       26-35  1           C              3                           0               20                  2.0
550065  1006036  P00375436   F       26-35  15          B              4+                          1               20                  2.0
550066  1006038  P00375436   F       55+    1           C              2                           0               20                  2.0
550067  1006039  P00371644   F       46-50  0           B              4+                          1               20                  2.0

550068 rows × 12 columns
```

```
In [120…
df2.sort_index(ascending=False)
```

Out[120…

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Catego
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	2.0	
550066	1006038	P00375436	F	55+	1	C	2	0	20	2.0	
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	2.0	
550064	1006035	P00375436	F	26-35	1	C	3	0	20	2.0	
550063	1006033	P00372445	M	51-55	13	B	1	1	20	2.0	
...	...	...	...	...	...	...	...	...	...	...	
4	1000002	P00285442	M	55+	16	C	4+	0	8	14.0	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	6.0	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
0	1000001	P00069042	F	0-17	10	A	2	0	3	6.0	

550068 rows × 12 columns

In [121…

```
# sorting by columns
df2.sort_index(axis=1)
```

Out[121…

	Age	City_Category	Gender	Marital_Status	Occupation	Product_Category_1	Product_Category_2	Product_Category_3	Product_ID	Purchase	Stay_In_Current_City_
0	0-17	A	F	0	10	3	6.0	14.0	P00069042	8370	
1	0-17	A	F	0	10	1	6.0	14.0	P00248942	15200	
2	0-17	A	F	0	10	12	6.0	14.0	P00087842	1422	
3	0-17	A	F	0	10	12	14.0	14.0	P00085442	1057	
4	55+	C	M	0	16	8	14.0	14.0	P00285442	7969	
...	...	...	...	...	...	...	...	...	...	...	
550063	51-55	B	M	1	13	20	2.0	11.0	P00372445	368	
550064	26-35	C	F	0	1	20	2.0	11.0	P00375436	371	
550065	26-35	B	F	1	15	20	2.0	11.0	P00375436	137	
550066	55+	C	F	0	1	20	2.0	11.0	P00375436	365	
550067	46-50	B	F	1	0	20	2.0	11.0	P00371644	490	

550068 rows × 12 columns

In [122…

```
#sorting by values
df2.sort_values(by=['Product_Category_1'])
```

Out[122...

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Catego
271814	1005880	P00016042	M	26-35	1	A	1	1	1	16.0	
208659	1002109	P00298942	M	26-35	16	B	2	0	1	5.0	
436707	1001231	P00334242	M	26-35	12	C	1	0	1	8.0	
108508	1004685	P00025442	M	36-45	1	B	2	1	1	2.0	
208658	1002109	P00062842	M	26-35	16	B	2	0	1	2.0	
...	...	...	...	...	...	...	...	...	...	...	
547638	1002549	P00375436	M	55+	13	C	3	1	20	2.0	
547640	1002553	P00375436	M	26-35	7	C	0	0	20	2.0	
547642	1002556	P00371644	M	26-35	4	C	2	0	20	2.0	
547644	1002558	P00375436	M	55+	17	C	3	1	20	2.0	
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	2.0	

550068 rows × 12 columns

In [123...

```
df2.sort_values(by=['Product_Category_1','Product_Category_2']) # sort by multiple columns
```

Out[123…

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Catego
5	1000003	P00193542	M	26-35	15	A	3	0	1	2.0	
13	1000005	P00145042	M	26-35	20	A	1	1	1	2.0	
39	1000010	P00221342	F	36-45	1	B	4+	1	1	2.0	
48	1000011	P00110842	F	26-35	1	C	1	0	1	2.0	
64	1000015	P00042142	M	26-35	7	A	1	0	1	2.0	
...	...	...	...	...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	1	1	20	2.0	
550064	1006035	P00375436	F	26-35	1	C	3	0	20	2.0	
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	2.0	
550066	1006038	P00375436	F	55+	1	C	2	0	20	2.0	
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	2.0	

550068 rows × 12 columns

In [124…

```
df2.sort_values(by='Product_Category_1',ascending=False) # sort in descending
```

Out[124…

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Catego
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	2.0	
547652	1002572	P00375436	M	36-45	14	C	1	0	20	2.0	
547670	1002593	P00375436	M	55+	16	C	2	0	20	2.0	
547668	1002590	P00371644	M	18-25	4	A	0	0	20	2.0	
547667	1002589	P00375436	M	26-35	0	C	3	0	20	2.0	
...	...	...	...	...	...	...	...	...	...	...	
35645	1005504	P00182342	M	46-50	7	B	1	1	1	5.0	
255039	1003384	P00111142	M	36-45	7	C	1	0	1	15.0	
438811	1001557	P00063342	F	18-25	4	C	1	0	1	2.0	
438812	1001558	P00110942	M	18-25	6	A	1	1	1	2.0	
56674	1002761	P00129342	M	36-45	6	B	0	0	1	5.0	

550068 rows × 12 columns

Categorical data in Pandas

In [126…

```
df3=df.copy()
df3.dtypes
```

Out[126…

```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category_1             int64
Product_Category_2           float64
Product_Category_3           float64
Purchase                       int64
dtype: object
```

In [127…

```
# description of categorical data
df3['Gender'].describe()
```

Out[127…

```
count     550068
unique         2
top            M
freq      414259
Name: Gender, dtype: object
```

In [128…

```
df3['Age'].describe()
```

Out[128…

```
count     550068
unique         7
top        26-35
freq      219587
Name: Age, dtype: object
```

In [129…

```
df3['City_Category'].describe()
```

Out[129…

```
count     550068
unique         3
top            B
freq      231173
Name: City_Category, dtype: object
```

In [130…

```
df3['Gender'].unique()
```

```
Out[130…  array(['F', 'M'], dtype=object)
```

```
In [131…  df3['Age'].unique()
```

```
Out[131…  array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
                dtype=object)
```

```
In [132…  df3['Gender'].value_counts()
```

```
Out[132…  Gender
          M    414259
          F    135809
          Name: count, dtype: int64
```

```
In [133…  df3['City_Category'].value_counts()
```

```
Out[133…  City_Category
          B    231173
          C    171175
          A    147720
          Name: count, dtype: int64
```

```
In [134…  df3['Gender'].value_counts(ascending=True)
```

```
Out[134…  Gender
          F    135809
          M    414259
          Name: count, dtype: int64
```

```
In [135…  df3['City_Category'].value_counts(ascending=True)
```

```
Out[135…  City_Category
          A    147720
          C    171175
          B    231173
          Name: count, dtype: int64
```

Descriptive Stats in Pandasive product

- count() - Number of non-null observations

- sum() - Sum of values
- mean() - Mean of values
- median() - Median of values
- mode() - Mode of values
- std() - Standard deviation of the values
- min() - Minimum value
- max() - Maximum value
- abs() - Absolute value
- prod() - Product of values
- cumsum() - Cumulative sum
- cumprod() - Cumulative product

In [138…

```
df4=df.copy()
```

In [139…

```
df4.max(0)
```

Out[139…

```
User_ID                        1006040
Product_ID                    P0099942
Gender                               M
Age                                55+
Occupation                          20
City_Category                        C
Stay_In_Current_City_Years          4+
Marital_Status                       1
Product_Category_1                  20
Product_Category_2                18.0
Product_Category_3                18.0
Purchase                         23961
dtype: object
```

summarizing data

- object – Summarizes string columns
- number – Summarizes numeric columns
- all – Summarizes all columns together

In [141…

```
df4.describe()
```

Out[141…

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9.863190	12.650723	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5.049456	4.115118	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	14.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	16.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	18.000000	23961.000000

Data Ranking

In [152…

```
df5=df.copy()

df5.head()
```

Out[152…

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	6.0	14.0
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	P00087842	F	0-17	10	A	2	0	12	6.0	14.0
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	14.0
4	1000002	P00285442	M	55+	16	C	4+	0	8	14.0	14.0

In [154…

```
df5.rank(method='min').head(25)
```

Out[154…

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Catego
0	1.0	100958.0	1.0	1.0	333113.0	1.0	268220.0	1.0	164243.0	148481.0	2225
1	1.0	384262.0	1.0	1.0	333113.0	1.0	268220.0	1.0	1.0	148481.0	2225
2	1.0	126991.0	1.0	1.0	333113.0	1.0	268220.0	1.0	515076.0	148481.0	2225
3	1.0	120944.0	1.0	1.0	333113.0	1.0	268220.0	1.0	515076.0	329172.0	2225
4	36.0	439307.0	135810.0	528565.0	436010.0	378894.0	465343.0	1.0	371329.0	329172.0	2225
5	113.0	307805.0	135810.0	114763.0	423845.0	1.0	370058.0	1.0	1.0	1.0	2225
6	142.0	293674.0	135810.0	444363.0	266143.0	147721.0	268220.0	324732.0	1.0	173407.0	4782
7	142.0	513077.0	135810.0	444363.0	266143.0	147721.0	268220.0	324732.0	1.0	411598.0	4782
8	142.0	546099.0	135810.0	444363.0	266143.0	147721.0	268220.0	324732.0	1.0	463267.0	4782
9	156.0	424058.0	135810.0	114763.0	516507.0	1.0	74399.0	324732.0	371329.0	463267.0	4782
10	156.0	390428.0	135810.0	114763.0	516507.0	1.0	74399.0	324732.0	196209.0	283521.0	4782
11	156.0	22349.0	135810.0	114763.0	516507.0	1.0	74399.0	324732.0	371329.0	283521.0	4782
12	156.0	47068.0	135810.0	114763.0	516507.0	1.0	74399.0	324732.0	371329.0	283521.0	4782
13	156.0	232254.0	135810.0	114763.0	516507.0	1.0	74399.0	324732.0	1.0	1.0	78
14	262.0	361984.0	1.0	490064.0	326822.0	1.0	74399.0	1.0	196209.0	173407.0	2225
15	262.0	301823.0	1.0	490064.0	326822.0	1.0	74399.0	1.0	184456.0	110095.0	2225
16	262.0	544723.0	1.0	490064.0	326822.0	1.0	74399.0	1.0	140379.0	69948.0	17
17	262.0	92472.0	1.0	490064.0	326822.0	1.0	74399.0	1.0	196209.0	329172.0	17
18	309.0	57746.0	135810.0	334350.0	69639.0	147721.0	74399.0	324732.0	1.0	329172.0	3712
19	326.0	385479.0	135810.0	114763.0	357629.0	378894.0	465343.0	324732.0	1.0	110095.0	2878
20	326.0	345339.0	135810.0	114763.0	357629.0	378894.0	465343.0	324732.0	196209.0	329172.0	2878
21	326.0	251903.0	135810.0	114763.0	357629.0	378894.0	465343.0	324732.0	371329.0	329172.0	2878

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Catego
22	326.0	333608.0	135810.0	114763.0	357629.0	378894.0	465343.0	324732.0	371329.0	329172.0	2878
23	326.0	334833.0	135810.0	114763.0	357629.0	378894.0	465343.0	324732.0	371329.0	329172.0	2878
24	326.0	464201.0	135810.0	114763.0	357629.0	378894.0	465343.0	324732.0	1.0	173407.0	2225

In [156...

```
df6=df.copy()
df6['Purchase'].aggregate(np.sum)
```

Out[156...

```
5095812742
```

In [158...

```
df6['Purchase'].aggregate([np.sum,np.mean])
```

Out[158...

```
sum     5.095813e+09
mean    9.263969e+03
Name: Purchase, dtype: float64
```

In [162...

```
#Applying aggregation multiple columns of a dataframe
df6[['Product_Category_1','Product_Category_2','Product_Category_3']].aggregate([np.sum,np.mean])
```

Out[162...

	Product_Category_1	Product_Category_2	Product_Category_3
sum	2.972716e+06	5.425425e+06	6.958758e+06
mean	5.404270e+00	9.863190e+00	1.265072e+01

In [166...

```
df6.aggregate({'Product_Category_1':np.sum,'Product_Category_2':np.mean})
```

Out[166...

```
Product_Category_1    2.972716e+06
Product_Category_2    9.863190e+00
dtype: float64
```

Pandas Merging and Joining

In [169...

```
# let's create two dataframes
```

```
batsmen = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Rohit', 'Dhawan', 'Virat', 'Dhoni', 'Kedar'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})

bowler = pd.DataFrame(
   {'id':[1,2,3,4,5],
   'Name': ['Kumar', 'Bumrah', 'Shami', 'Kuldeep', 'Chahal'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})


print(batsmen)


print(bowler)
```

```
   id    Name subject_id
0   1   Rohit       sub1
1   2  Dhawan       sub2
2   3   Virat       sub4
3   4   Dhoni       sub6
4   5   Kedar       sub5
   id     Name subject_id
0   1    Kumar       sub2
1   2   Bumrah       sub4
2   3    Shami       sub3
3   4  Kuldeep       sub6
4   5   Chahal       sub5
```

In [171…

```
# merge two dataframes on a key
pd.merge(batsmen,bowler,on='id')
```

Out[171...

	id	Name_x	subject_id_x	Name_y	subject_id_y
0	1	Rohit	sub1	Kumar	sub2
1	2	Dhawan	sub2	Bumrah	sub4
2	3	Virat	sub4	Shami	sub3
3	4	Dhoni	sub6	Kuldeep	sub6
4	5	Kedar	sub5	Chahal	sub5

In [173...

```
#merge two dataframes on multiple keys
pd.merge(batsmen,bowler,on=['id','subject_id'])
```

Out[173...

	id	Name_x	subject_id	Name_y
0	4	Dhoni	sub6	Kuldeep
1	5	Kedar	sub5	Chahal

In [175...

```
#Merge using How argument
#left join
pd.merge(batsmen,bowler,on='subject_id',how='left')
```

Out[175...

	id_x	Name_x	subject_id	id_y	Name_y
0	1	Rohit	sub1	NaN	NaN
1	2	Dhawan	sub2	1.0	Kumar
2	3	Virat	sub4	2.0	Bumrah
3	4	Dhoni	sub6	4.0	Kuldeep
4	5	Kedar	sub5	5.0	Chahal

In [177...

```
pd.merge(batsmen,bowler,on='subject_id',how='right')#right join
```

Out[177...

	id_x	Name_x	subject_id	id_y	Name_y
0	2.0	Dhawan	sub2	1	Kumar
1	3.0	Virat	sub4	2	Bumrah
2	NaN	NaN	sub3	3	Shami
3	4.0	Dhoni	sub6	4	Kuldeep
4	5.0	Kedar	sub5	5	Chahal

In [179...

```
pd.merge(batsmen,bowler,on='subject_id',how='outer')# outer join
```

Out[179...

	id_x	Name_x	subject_id	id_y	Name_y
0	1.0	Rohit	sub1	NaN	NaN
1	2.0	Dhawan	sub2	1.0	Kumar
2	NaN	NaN	sub3	3.0	Shami
3	3.0	Virat	sub4	2.0	Bumrah
4	5.0	Kedar	sub5	5.0	Chahal
5	4.0	Dhoni	sub6	4.0	Kuldeep

In [181...

```
pd.merge(batsmen,bowler,on='subject_id',how='inner')# inner join
```

Out[181...

	id_x	Name_x	subject_id	id_y	Name_y
0	2	Dhawan	sub2	1	Kumar
1	3	Virat	sub4	2	Bumrah
2	4	Dhoni	sub6	4	Kuldeep
3	5	Kedar	sub5	5	Chahal

In [183...

```
df5.describe().cov()
```

Out[183…

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
User_ID	1.347756e+11	-2.130859e+10	-2.130916e+10	-2.130857e+10	-2.130822e+10	-2.130775e+10	-2.034926e+10
Occupation	-2.130859e+10	3.782072e+10	3.782126e+10	3.782085e+10	3.782066e+10	3.782053e+10	3.719088e+10
Marital_Status	-2.130916e+10	3.782126e+10	3.782179e+10	3.782139e+10	3.782119e+10	3.782107e+10	3.719137e+10
Product_Category_1	-2.130857e+10	3.782085e+10	3.782139e+10	3.782098e+10	3.782079e+10	3.782066e+10	3.719101e+10
Product_Category_2	-2.130822e+10	3.782066e+10	3.782119e+10	3.782079e+10	3.782060e+10	3.782047e+10	3.719081e+10
Product_Category_3	-2.130775e+10	3.782053e+10	3.782107e+10	3.782066e+10	3.782047e+10	3.782034e+10	3.719068e+10
Purchase	-2.034926e+10	3.719088e+10	3.719137e+10	3.719101e+10	3.719081e+10	3.719068e+10	3.662011e+10

Pandas Concatenation

In [196…

```
batsmen = pd.DataFrame({
   'id':[1,2,3,4,5],
   'Name': ['Rohit', 'Dhawan', 'Virat', 'Dhoni', 'Kedar'],
   'subject_id':['sub1','sub2','sub4','sub6','sub5']})

bowler = pd.DataFrame(
   {'id':[1,2,3,4,5],
   'Name': ['Kumar', 'Bumrah', 'Shami', 'Kuldeep', 'Chahal'],
   'subject_id':['sub2','sub4','sub3','sub6','sub5']})


print(batsmen)


print(bowler)
```

```
   id    Name subject_id
0   1   Rohit       sub1
1   2  Dhawan       sub2
2   3   Virat       sub4
3   4   Dhoni       sub6
4   5   Kedar       sub5
   id     Name subject_id
0   1    Kumar       sub2
1   2   Bumrah       sub4
2   3    Shami       sub3
3   4  Kuldeep       sub6
4   5   Chahal       sub5
```

In [199…

```
team=[batsmen,bowler]
pd.concat(team)
```

Out[199…

	id	Name	subject_id
0	1	Rohit	sub1
1	2	Dhawan	sub2
2	3	Virat	sub4
3	4	Dhoni	sub6
4	5	Kedar	sub5
0	1	Kumar	sub2
1	2	Bumrah	sub4
2	3	Shami	sub3
3	4	Kuldeep	sub6
4	5	Chahal	sub5

In [201…

```
#associate keys with the dataframes
pd.concat(team,keys=['x','y'])
```

Out[201…

		id	Name	subject_id
x	**0**	1	Rohit	sub1
	1	2	Dhawan	sub2
	2	3	Virat	sub4
	3	4	Dhoni	sub6
	4	5	Kedar	sub5
y	**0**	1	Kumar	sub2
	1	2	Bumrah	sub4
	2	3	Shami	sub3
	3	4	Kuldeep	sub6
	4	5	Chahal	sub5

In [203…

```
pd.concat(team,keys=['x','y'],ignore_index=True)
```

Out[203…

	id	Name	subject_id
0	1	Rohit	sub1
1	2	Dhawan	sub2
2	3	Virat	sub4
3	4	Dhoni	sub6
4	5	Kedar	sub5
5	1	Kumar	sub2
6	2	Bumrah	sub4
7	3	Shami	sub3
8	4	Kuldeep	sub6
9	5	Chahal	sub5

In [205…

```
pd.concat(team,axis=1)
```

Out[205…

	id	Name	subject_id	id	Name	subject_id
0	1	Rohit	sub1	1	Kumar	sub2
1	2	Dhawan	sub2	2	Bumrah	sub4
2	3	Virat	sub4	3	Shami	sub3
3	4	Dhoni	sub6	4	Kuldeep	sub6
4	5	Kedar	sub5	5	Chahal	sub5

Reshapig by Melt and Pivot

In [211…

```
df11=df.copy()
df11.col
```

Out[211…

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

In [215…

```
df12=(pd.melt(frame=df11,id_vars=['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status','Purchase'],value_vars=['Product_Category_1',
       'Product_Category_2', 'Product_Category_3'],var_name='Product_Category',value_name='Amount'))
df12.head(10)
```

Out[215…

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Purchase	Product_Category	Amount
0	1000001	P00069042	F	0-17	10	A	2	0	8370	Product_Category_1	3.0
1	1000001	P00248942	F	0-17	10	A	2	0	15200	Product_Category_1	1.0
2	1000001	P00087842	F	0-17	10	A	2	0	1422	Product_Category_1	12.0
3	1000001	P00085442	F	0-17	10	A	2	0	1057	Product_Category_1	12.0
4	1000002	P00285442	M	55+	16	C	4+	0	7969	Product_Category_1	8.0
5	1000003	P00193542	M	26-35	15	A	3	0	15227	Product_Category_1	1.0
6	1000004	P00184942	M	46-50	7	B	2	1	19215	Product_Category_1	1.0
7	1000004	P00346142	M	46-50	7	B	2	1	15854	Product_Category_1	1.0
8	1000004	P0097242	M	46-50	7	B	2	1	15686	Product_Category_1	1.0
9	1000005	P00274942	M	26-35	20	A	1	1	7871	Product_Category_1	8.0

Options and Customization

- get_option()
- set_option()
- reset_option()
- describe_option()
- option_context()

```
In [231…  #display maximum rows
          pd.get_option("display.max_rows")

Out[231…  60

In [233…  pd.get_option("display.max_columns")

Out[233…  20

In [235…  pd.set_option("display.max_rows",80)
          pd.get_option("display.max_rows")

Out[235…  80

In [237…  pd.set_option("display.max_columns",30)
          pd.get_option("display.max_columns")

Out[237…  30

In [239…  #display maxium rows
          pd.reset_option("display.max_rows")
          pd.get_option("display.max_rows")

Out[239…  60

In [243…  # display maximum columns
          pd.reset_option("display.max_columns")
          pd.get_option("display.max_columns")

Out[243…  20

In [245…  #description of the displa maximum rows parameter
          pd.describe_option("display.max_rows")
```

```
display.max_rows : int
    If max_rows is exceeded, switch to truncate view. Depending on
    `large_repr`, objects are either centrally truncated or printed as
    a summary view. 'None' value means unlimited.

    In case python/IPython is running in a terminal and `large_repr`
    equals 'truncate' this can be set to 0 and pandas will auto-detect
    the height of the terminal and print a truncated object which fits
    the screen height. The IPython notebook, IPython qtconsole, or
    IDLE do not run in a terminal and hence it is not possible to do
    correct auto-detection.
    [default: 60] [currently: 60]
```

In [247…

```python
#set the parameter value with option_context
with pd.option_context("display.max_rows",10):
     print(pd.get_option("display.max_rows"))
     print(pd.get_option("display.max_rows"))
```

```
10
10
```

Summary and Conclusion

> In this kernel, I have explored pandas and important data analysis tools of pandas.

- I have used the Black Friday dataset and explore various functionalities offered by pandas.
- I have shed light on important functionalities of pandas like aggregations in pandas, iteration in pandas, Pandas GroupBy operations, Pandas merging and joining.
- I have also discussed Pandas concatenation operation, Reshaping by melt and pivot and Reshaping by stacking and unstacking.
- I have also discussed basic functionality in Pandas, descriptive statistics in Pandas and statistical functions in Pandas.
- Lastly, I have discussed options and customization options with Pandas.

In [ ]:

```

```