In [90]:
```
# Pandas basics

# Python cannot handle data storage - it will use objects

# A data in python is a collection of various 1d arrays
    # 1. homogenous

# Panel Data Analysis
    # 1. 1 - D DS - Series
    # 2. 2 - D DS - DataFrame
```

In [91]:
```
l1 = [12,54,23,45,89,12,54,34,78,98,76,56,58,43,2,11,66,33]
```

In [92]:
```
import pandas as pd
ser1 = pd.Series(l1)
```

In [93]:
```
ser1 # index and values
```

Out[93]:
```
0     12
1     54
2     23
3     45
4     89
5     12
6     54
7     34
8     78
9     98
10    76
11    56
12    58
13    43
14     2
15    11
16    66
17    33
dtype: int64
```

In [94]:
```
# series index can be the default indexes 0 to n-1 or
# customized

#ser1.iloc[]
#ser1.loc[]

# only default indexes
# both iloc and loc can be used
# iloc[0:10] -> 0,1,2,3,...9
# loc[0:10] -> 0,1,2,3,4,5,...10

# if both default and user defined index exists
# iloc for default
# loc for user defined
```

```
# Importing Data

pd.read_XXXXXX

1. file type
2. file path, name and ext
    URL/name.ext
3. for excel = sheet
4. for text  = the separator (, or | or \t)
```

In [95]:
```
# Working Directory
# a folder is set as the default directory
# directly import files from this WD or export to WD
import os
os.chdir("C:/Users/admin/pandas/DataSets")
```

In [96]:
```
stores = pd.read_csv("stores.csv")
```

In [97]:
```
stores
```

Out[97]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	Total_Custo
0	STR101	Electronics Zone	Electronincs	Delhi	21.0	60	160.0	
1	STR102	Apparel Zone	Apparel	Delhi	21.0	60	160.0	
2	STR103	Super Bazar	Super Market	Delhi	22.8	40	108.0	
3	STR104	Super Market	Super Market	Delhi	21.4	60	258.0	
4	STR105	Central Store	Super Market	Delhi	18.7	80	360.0	
5	STR106	Apparel Zone	Apparel	Delhi	18.1	60	225.0	
6	STR107	Fashion Bazar	Apparel	Delhi	14.3	80	360.0	
7	STR108	Digital Bazar	Electronincs	Delhi	24.4	40	146.7	
8	STR109	Electronics Zone	Electronincs	Chennai	22.8	40	140.8	
9	STR110	Apparel Zone	Apparel	Chennai	19.2	60	167.6	
10	STR111	Super Bazar	Super Market	Chennai	17.8	60	167.6	
11	STR112	Super Market	Super Market	Chennai	16.4	80	275.8	
12	STR113	Central Store	Super Market	Chennai	17.3	80	275.8	
13	STR114	Apparel Zone	Apparel	Chennai	15.2	80	275.8	
14	STR115	Fashion Bazar	Apparel	Chennai	10.4	80	472.0	
15	STR116	Digital Bazar	Electronincs	Chennai	10.4	80	460.0	
16	STR117	Electronics Zone	Electronincs	Mumbai	14.7	80	440.0	
17	STR118	Apparel Zone	Apparel	Mumbai	32.4	40	78.7	
18	STR119	Super Bazar	Super Market	Mumbai	30.4	40	75.7	
19	STR120	Super Market	Super Market	Mumbai	33.9	40	71.1	
20	STR121	Central Store	Super Market	Mumbai	21.5	40	120.1	
21	STR122	Apparel Zone	Apparel	Mumbai	15.5	80	318.0	

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	Total_Custo
22	STR123	Fashion Bazar	Apparel	Mumbai	15.2	80	304.0	
23	STR124	Digital Bazar	Electronincs	Mumbai	13.3	80	350.0	
24	STR125	Electronics Zone	Electronincs	Kolkata	19.2	80	400.0	
25	STR126	Apparel Zone	Apparel	Kolkata	27.3	40	79.0	
26	STR127	Super Bazar	Super Market	Kolkata	26.0	40	120.3	
27	STR128	Super Market	Super Market	Kolkata	30.4	40	95.1	
28	STR129	Central Store	Super Market	Kolkata	15.8	80	351.0	
29	STR130	Apparel Zone	Apparel	Kolkata	19.7	60	145.0	
30	STR131	Fashion Bazar	Apparel	Kolkata	15.0	80	301.0	
31	STR132	Digital Bazar	Electronincs	Kolkata	21.4	40	121.0	

Basic Sumary of the Data

In [ ]:
```
n = 3
stores.head(n) # first 6/n obs
stores.tail(n)
```

In [11]:
```
stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 15 columns):
StoreCode          32 non-null object
StoreName          32 non-null object
StoreType          32 non-null object
Location           32 non-null object
OperatingCost      32 non-null float64
Staff_Cnt          32 non-null int64
TotalSales         32 non-null float64
Total_Customers    32 non-null int64
AcqCostPercust     29 non-null float64
BasketSize         32 non-null float64
ProfitPercust      32 non-null float64
OwnStore           32 non-null int64
OnlinePresence     32 non-null int64
Tenure             32 non-null int64
StoreSegment       32 non-null int64
dtypes: float64(5), int64(6), object(4)
memory usage: 3.8+ KB
```

In [13]:
```
stores.shape # (nrow, ncol)

nrow = stores.shape[0]
ncol = stores.shape[1]
```

In [15]:
```
stores.dtypes
stores.get_dtype_counts()
```

Out[15]:
```
float64    5
int64      6
object     4
dtype: int64
```

In [16]:
```
stores.count()
```

Out[16]:
```
StoreCode          32
StoreName          32
StoreType          32
Location           32
OperatingCost      32
Staff_Cnt          32
TotalSales         32
Total_Customers    32
AcqCostPercust     29
BasketSize         32
ProfitPercust      32
OwnStore           32
OnlinePresence     32
Tenure             32
StoreSegment       32
dtype: int64
```

In [17]:
```
stores.describe()
```

Out[17]:

	OperatingCost	Staff_Cnt	TotalSales	Total_Customers	AcqCostPercust	BasketSize	ProfitF
count	32.000000	32.000000	32.000000	32.000000	29.000000	32.000000	32.
mean	20.090625	61.875000	230.721875	146.687500	3.651034	3.217250	17.
std	6.026948	17.859216	123.938694	68.562868	0.532664	0.978457	1.
min	10.400000	40.000000	71.100000	52.000000	2.760000	1.513000	14.
25%	15.425000	40.000000	120.825000	96.500000	3.150000	2.581250	16.
50%	19.200000	60.000000	196.300000	123.000000	3.730000	3.325000	17.
75%	22.800000	80.000000	326.000000	180.000000	3.920000	3.610000	18.
max	33.900000	80.000000	472.000000	335.000000	4.930000	5.424000	22.

Columns

In [ ]:
```
# 1.  Extacting columns
# 1.1 The . operator
stores.TotalSales

# 1.2 the []

stores["TotalSales"]
stores[["Location","OperatingCost","TotalSales"]]

# 1.3 By using .loc (or .iloc[])
stores.loc[:10,["Location","OperatingCost","TotalSales"]] # rows -> inddex 0 to n
# DF with first 11 records and three mentioned columns
```

In [ ]:
```
# Changing Datatypes
pd.to_numeric(,errors = "coerce")
pd.to_datetime(,format = )
-----
dataframe.astype(bool/str)
```

In [98]:
```
# adding columns

# TotalSales -> Income
# OperatingCost -> expense
# NetProfit = TotalSales - OperatingCost

stores["NetProfit1"] = stores["TotalSales"] - stores["OperatingCost"]
stores.head(2)
# stores["NetProfit1"] = stores.TotalSales - stores.OperatingCost

stores = stores.assign(NetProfit2 = stores.TotalSales - stores.OperatingCost,
              GrandTotalSales = stores.TotalSales * stores.Total_Customers)
```

In [99]:
```
stores.head(2)
```

Out[99]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	Total_Custor
0	STR101	Electronics Zone	Electronincs	Delhi	21.0	60	160.0	
1	STR102	Apparel Zone	Apparel	Delhi	21.0	60	160.0	

In [102]:
```
stores.iloc[:,[1,2,3,4]]
```

Out[102]:

	StoreName	StoreType	Location	OperatingCost
0	Electronics Zone	Electronincs	Delhi	21.0
1	Apparel Zone	Apparel	Delhi	21.0
2	Super Bazar	Super Market	Delhi	22.8
3	Super Market	Super Market	Delhi	21.4
4	Central Store	Super Market	Delhi	18.7
5	Apparel Zone	Apparel	Delhi	18.1
6	Fashion Bazar	Apparel	Delhi	14.3
7	Digital Bazar	Electronincs	Delhi	24.4
8	Electronics Zone	Electronincs	Chennai	22.8
9	Apparel Zone	Apparel	Chennai	19.2
10	Super Bazar	Super Market	Chennai	17.8
11	Super Market	Super Market	Chennai	16.4
12	Central Store	Super Market	Chennai	17.3
13	Apparel Zone	Apparel	Chennai	15.2
14	Fashion Bazar	Apparel	Chennai	10.4
15	Digital Bazar	Electronincs	Chennai	10.4
16	Electronics Zone	Electronincs	Mumbai	14.7
17	Apparel Zone	Apparel	Mumbai	32.4
18	Super Bazar	Super Market	Mumbai	30.4
19	Super Market	Super Market	Mumbai	33.9
20	Central Store	Super Market	Mumbai	21.5
21	Apparel Zone	Apparel	Mumbai	15.5
22	Fashion Bazar	Apparel	Mumbai	15.2
23	Digital Bazar	Electronincs	Mumbai	13.3
24	Electronics Zone	Electronincs	Kolkata	19.2
25	Apparel Zone	Apparel	Kolkata	27.3
26	Super Bazar	Super Market	Kolkata	26.0
27	Super Market	Super Market	Kolkata	30.4
28	Central Store	Super Market	Kolkata	15.8
29	Apparel Zone	Apparel	Kolkata	19.7
30	Fashion Bazar	Apparel	Kolkata	15.0
31	Digital Bazar	Electronincs	Kolkata	21.4

In [105]:
```
stores.head(2)
```

Out[105]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	Total_Custor
0	STR101	Electronics Zone	Electronincs	Delhi	21.0	60	160.0	
1	STR102	Apparel Zone	Apparel	Delhi	21.0	60	160.0	

In [ ]:
```
stores.iloc[:,[0,1,2,3,4,5,6,15,7,8,9,10,11,12,13,14,16,17]]
```

In [108]:
```
[1,2,3] + [5,6,7] + range(3,7)
```

Out[108]:
```
[1, 2, 3, 5, 6, 7, 3, 4, 5, 6]
```

In [110]:
```
range(0,7)+[15]+range(7,15)+[16,17]
```

Out[110]:
```
[0, 1, 2, 3, 4, 5, 6, 15, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17]
```

In [ ]:
```
stores.iloc[:,range(0,7)+[15]+range(7,15)+[16,17]]
```

In [ ]:

In [ ]:

In [ ]:

In [26]:
```
# Removing columns

del stores["NetProfit2"]

# stores.drop("NetProfit2")
```

In [27]:
```
stores.head(2)
```

Out[27]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	Total_Custor
0	STR101	Electronics Zone	Electronincs	Delhi	21.0	60	160.0	
1	STR102	Apparel Zone	Apparel	Delhi	21.0	60	160.0	

In [ ]:
```
# renaming columns
stores.rename(columns = {})

# take a dictionary as input
# "key" : "value
    # "ExisitngColumnName" : "NewColumnName"
```

In [31]:
```
stores = stores.rename(columns = {"NetProfit1" : "NetProfit"})
stores.head(2)
```

Out[31]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	Total_Custor
0	STR101	Electronics Zone	Electronincs	Delhi	21.0	60	160.0	
1	STR102	Apparel Zone	Apparel	Delhi	21.0	60	160.0	

In [34]:
```
# ser1 > 30 and < 80
ser1.loc[(ser1 > 30) & (ser1 < 80)]
```

Out[34]:
```
1     54
3     45
6     54
7     34
8     78
10    76
11    56
12    58
13    43
16    66
17    33
dtype: int64
```

In [36]:
```
# Filtering - applying conditions to the data
# Get records from stores where Location is Delhi
SubSet1 = stores.loc[(stores.Location == "Delhi"),:]
```

In [39]:
```
# Records from Kolkata where TotalSales > 100 and < 300
stores.loc[(stores.Location == "Kolkata") & (stores.TotalSales >= 100) & (stores.
```

Out[39]:

	StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales	Total_Custo
26	STR127	Super Bazar	Super Market	Kolkata	26.0	40	120.3	
29	STR130	Apparel Zone	Apparel	Kolkata	19.7	60	145.0	
31	STR132	Digital Bazar	Electronincs	Kolkata	21.4	40	121.0	

In [ ]:
```
# All store types from Chennai where OperatingCost > 15
```

In [42]:
```
stores.loc[(stores.Location == "Chennai") & (stores.OperatingCost > 15),["StoreCo
```

Out[42]:

	StoreCode	StoreType
8	STR109	Electronincs
9	STR110	Apparel
10	STR111	Super Market
11	STR112	Super Market
12	STR113	Super Market
13	STR114	Apparel

In [43]:
```
# sorting data
stores.sort_values(["Location"],ascending=True)
```

In [ ]:
```
# sort data acc to one column - Location asc
stores.sort_values("Location",ascending=True)
```

In [ ]:
```
# sort data acc to one column - TotalSales desc
stores.sort_values("TotalSales",ascending=False)
```

In [ ]:
```
# sort data acc to two columns - Location, OperatingCost asc
stores.sort_values(["Location","OperatingCost"],ascending=True)
```

In [44]:
```
# sort data acc to two columns - Location, TotalSales desc
stores.sort_values(["Location","TotalSales"],ascending=False)
```

In [ ]:
```
# sort data acc to two columns - Location in asc, TotalSales in desc
stores.sort_values(["Location","TotalSales"],ascending=[True,False])
```

In [49]:
```
# Removing Duplicates
# Employees
Score = pd.read_csv("Score.csv")
Score
```

Out[49]:

	Student	Section	Test1	Test2	Final
0	Capalleti	1	94	91	87
1	Dubose	2	51	65	91
2	Engles	1	95	97	97
3	Grant	2	63	75	80
4	Krupski	2	80	76	71
5	Lundsford	1	92	40	86
6	Mcbane	1	75	78	72
7	Capalleti	1	94	65	87
8	Dubose	2	51	65	91
9	Engles	1	95	97	97
10	Grant	2	63	75	80
11	Krupski	2	80	76	71
12	Lundsford	1	92	40	86
13	Mcbane	1	75	78	72

In [52]:
```
Score.duplicated()
# 12,2,3,4,3,4,5,12,1,5

# F,F,F,F,T,T,F,T,F,T
```

Out[52]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8      True
9      True
10     True
11     True
12     True
13     True
dtype: bool
```

In [53]:
```
Score.loc[Score.duplicated(),:] # a subset of the data where I have all duplicate
```

Out[53]:

	Student	Section	Test1	Test2	Final
8	Dubose	2	51	65	91
9	Engles	1	95	97	97
10	Grant	2	63	75	80
11	Krupski	2	80	76	71
12	Lundsford	1	92	40	86
13	Mcbane	1	75	78	72

In [54]:
```
Score.loc[-Score.duplicated(),:]
```

Out[54]:

	Student	Section	Test1	Test2	Final
0	Capalleti	1	94	91	87
1	Dubose	2	51	65	91
2	Engles	1	95	97	97
3	Grant	2	63	75	80
4	Krupski	2	80	76	71
5	Lundsford	1	92	40	86
6	Mcbane	1	75	78	72
7	Capalleti	1	94	65	87

In [57]:
```
# WRT a column
Score.loc[-Score.Student.duplicated(),:]
```

Out[57]:

	Student	Section	Test1	Test2	Final
0	Capalleti	1	94	91	87
1	Dubose	2	51	65	91
2	Engles	1	95	97	97
3	Grant	2	63	75	80
4	Krupski	2	80	76	71
5	Lundsford	1	92	40	86
6	Mcbane	1	75	78	72

```
# Check duplicates wrt to two columns
EmpId     Email    Phone    Name    Designation    Team        Sal
-----     -----    -----    ----    -----------    -----       ----
1144    abc@s.com  5464   John D    Manager        Analytics   $1900
1144    abc@s.com  5464   John D    Manager        Analytics   $1900
1234    der@s.com  1234   John D    Manager        Analytics   $1900
1144    pqr@s.com  6767   John D    Manager        Analytics   $1900

Emp.loc[Emp.EmpID.duplicated(),:]
```

```
Emp.loc[Emp.EmpID.duplicated() & Emp.Email.duplicated(),:]
```

```
Score.loc[(Score.Student.duplicated()) & (Score.Test1.duplicated()),:]
```

In [ ]:
```
# AcqCostPercust

# 1. With a zero or an empty string

# 2. With the mean or median

# 3. Simply remove the rows where there are missing

#-----------------------------------------------------------------
# 4. Use any predictive mod tech to guess the missing value    #
#-----------------------------------------------------------------
```

In [67]:
```
# Percentage of missing values
# Acq

# Step 1 : Get total obs in the column
nrow = stores.shape[0]

# Step 2 : Get no of non missing values
nonmiss = stores.AcqCostPercust.count()
nmiss = nrow - nonmiss

# Step 3 : % of missing
permiss = float(nmiss)/nrow * 100 # why float? float(100)/3
print permiss
```

```
9.375
```

In [118]:
```
ACQ = stores.AcqCostPercust
# Imputing
# 1. filling missing with zeros
ACQ.isna()
```

Out[118]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11     True
12     True
13     True
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
Name: AcqCostPercust, dtype: bool
```

In [119]:
```
["#N/A","NULL","#VALUE",np.NaN]
```

Out[119]:
```
['#N/A', 'NULL', '#VALUE', nan]
```

In [71]:
```
# Filling NaN with mean/median
AvgOfACQ = ACQ.mean() # sum/no of elements
MedianOfACQ = ACQ.median()

R1 = ACQ.fillna(AvgOfACQ)
R2 = ACQ.fillna(MedianOfACQ)
```

In [ ]:
```
# dropping all rows with NaN
stores.dropna()
```

In [ ]:
```
# Binning

# TotalSales -> PerformanceCategory

# TS < 100 - "Low"
# TS > 100 and < 200 - "Agerage"
# TS > 200 and < 300 - "High"
# TS > 300 - "Very High"
```

In [79]:
```
import numpy as np
x = -10
print np.where(x > 0, "positive","negative")
```

```
negative
```

In [82]:
```
ser1
pd.Series(np.where(ser1 > 50,"GT50","LT50"))
```

Out[82]:
```
0     LT50
1     GT50
2     LT50
3     LT50
4     GT50
5     LT50
6     GT50
7     LT50
8     GT50
9     GT50
10    GT50
11    GT50
12    GT50
13    LT50
14    LT50
15    LT50
16    GT50
17    LT50
dtype: object
```

In [84]:
```
# Location ->   Food
# Delhi    ->   "Chole Bhature"
# Mumbai   ->   "Vada Pav"
# Chennai  ->   "Pongal"
# Kolkata  ->   "Sandesh"

stores["Food"] = np.where((stores.Location == "Delhi"),"CB",
                          np.where((stores.Location == "Mumbai"),"VP",
                                   np.where((stores.Location == "Chennai"),"Ponga
                                            np.where((stores.Location == "Kolkata
```

In [88]:
```
# TS < 100 - "Low"
# TS > 100 and < 200 - "Agerage"
# TS > 200 and < 300 - "High"
# TS > 300 - "Very High"

TS = stores.TotalSales

stores["PerformaceCat"] = np.where((TS < 100),"Low",
         np.where((TS > 100) & (TS < 200),"Average",
                  np.where((TS > 200) & (TS < 300),"High","Very High")))
```

In [ ]:
```
# Gender -> M/F
# Gender1 -> F = 1 and M = 2
np.where(Gender == "F",1,np.where(Gender == "M",2,""))
```

In [122]:
```
def OddEven(x):
    if(x % 2 == 0):
        return(True)
    else:
        return(False)
```

Out[122]: False

In [ ]:
```
# Lambda functions
# mini/smaller unnamed or anonymous function
```

In [129]:
```
data = {'name': ['Dinkar', 'Vikalp', 'Sumeet', 'Shubham', 'Ramesh'],
        'year': [2012, 2012, 2013, 2014, 2014],
        'reports': [4, 24, 31, 2, 3],
        'coverage': [25, 94, 57, 62, 70]}
df = pd.DataFrame(data)
df = df.iloc[:,[1,0,2,3]]
```

In [127]:
```
def capitalizer(x):
    return x.upper()
```

In [128]:
```
capitalizer("xyz")
```

Out[128]: 'XYZ'

In [130]:
```
lambda x: x.upper()
df
```

Out[130]:

	name	coverage	reports	year
0	Dinkar	25	4	2012
1	Vikalp	94	24	2012
2	Sumeet	57	31	2013
3	Shubham	62	2	2014
4	Ramesh	70	3	2014

In [ ]:
```
capitalizer(df.name) # Syntax Error
```

In [133]:
```
df["name"].apply(capitalizer)
```

Out[133]:
```
0     DINKAR
1     VIKALP
2     SUMEET
3    SHUBHAM
4     RAMESH
Name: name, dtype: object
```

In [134]:
```
lambda x : x * 1000
```

Out[134]:
```
<function __main__.<lambda>>
```

In [135]:
```
df.coverage.apply(lambda x : x * 1000)
```

Out[135]:
```
0    25000
1    94000
2    57000
3    62000
4    70000
Name: coverage, dtype: int64
```

In [137]:
```
df[["coverage","reports","year"]].apply(lambda x : x * 1000)
```

Out[137]:

	coverage	reports	year
0	25000	4000	2012000
1	94000	24000	2012000
2	57000	31000	2013000
3	62000	2000	2014000
4	70000	3000	2014000

In [139]:
```
# Use a general function that returns multiple values
def var_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),
                      index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN
```

In [138]:
```
def summary(x):
    res = pd.Series([x.mean(),x.median(),x.sum()],index = ['mean','median','sum']

    return res
```

In [147]:
```
df.iloc[:,1:].apply(lambda x:summary(x))
```

Out[147]:

	coverage	reports	year
mean	61.6	12.8	2013.0
median	62.0	4.0	2013.0
sum	308.0	64.0	10065.0

In [ ]:
```
def summary(x):
        ....
        ....
```

In [148]:
```
# Group By
```

In [ ]:
```
# SELECT Location, sum(TotalSales) as 'SumOfTotalSales' FROM stores GROUP BY Loca
```

In [149]:
```
temp = stores.groupby(["Location"])
```

In [153]:
```
res = temp.TotalSales.agg({"SumOfTotalSales":"sum"})
res.reset_index()
```

```
C:\Users\admin\Anaconda2\lib\site-packages\ipykernel\__main__.py:1: FutureWarni
ng: using a dict on a Series for aggregation
is deprecated and will be removed in a future version
  if __name__ == '__main__':
```

Out[153]:

	Location	SumOfTotalSales
0	Chennai	2235.4
1	Delhi	1777.7
2	Kolkata	1612.4
3	Mumbai	1757.6

In [154]:
```
# select Location, StoreType, sum(TotalSales) from stores groupby Location, Store
```

In [156]:
```
temp = stores.groupby(["Location","StoreType"])
```

In [159]:
```
temp.TotalSales.agg({"SumOfTotalSale":"sum"}).reset_index()
# sum mean median var sd kurtosis skew len
```

```
C:\Users\admin\Anaconda2\lib\site-packages\ipykernel\__main__.py:1: FutureWarni
ng: using a dict on a Series for aggregation
is deprecated and will be removed in a future version
  if __name__ == '__main__':
```

Out[159]:

	Location	StoreType	SumOfTotalSale
0	Chennai	Apparel	915.4
1	Chennai	Electronincs	600.8
2	Chennai	Super Market	719.2
3	Delhi	Apparel	745.0
4	Delhi	Electronincs	306.7
5	Delhi	Super Market	726.0
6	Kolkata	Apparel	525.0
7	Kolkata	Electronincs	521.0
8	Kolkata	Super Market	566.4
9	Mumbai	Apparel	700.7
10	Mumbai	Electronincs	790.0
11	Mumbai	Super Market	266.9

In [160]:
```
temp = stores.groupby(["Location","StoreType"])
temp.TotalSales.agg({"SumOfTotalSales":"sum","MeanOfTotalSales":"mean"}).reset_in
```

```
C:\Users\admin\Anaconda2\lib\site-packages\ipykernel\__main__.py:2: FutureWarni
ng: using a dict on a Series for aggregation
is deprecated and will be removed in a future version
  from ipykernel import kernelapp as app
```

Out[160]:

	Location	StoreType	MeanOfTotalSales	SumOfTotalSales
0	Chennai	Apparel	305.133333	915.4
1	Chennai	Electronincs	300.400000	600.8
2	Chennai	Super Market	239.733333	719.2
3	Delhi	Apparel	248.333333	745.0
4	Delhi	Electronincs	153.350000	306.7
5	Delhi	Super Market	242.000000	726.0
6	Kolkata	Apparel	175.000000	525.0
7	Kolkata	Electronincs	260.500000	521.0
8	Kolkata	Super Market	188.800000	566.4
9	Mumbai	Apparel	233.566667	700.7
10	Mumbai	Electronincs	395.000000	790.0
11	Mumbai	Super Market	88.966667	266.9

In [168]:
```
temp = stores.groupby(["Location","StoreType"])
r = temp[["TotalSales","OperatingCost"]].agg({"Sum":"sum"}).reset_index()
r
```

Out[168]:

	Location	StoreType	Sum	
			TotalSales	**OperatingCost**
0	Chennai	Apparel	915.4	44.8
1	Chennai	Electronincs	600.8	33.2
2	Chennai	Super Market	719.2	51.5
3	Delhi	Apparel	745.0	53.4
4	Delhi	Electronincs	306.7	45.4
5	Delhi	Super Market	726.0	62.9
6	Kolkata	Apparel	525.0	62.0
7	Kolkata	Electronincs	521.0	40.6
8	Kolkata	Super Market	566.4	72.2
9	Mumbai	Apparel	700.7	63.1
10	Mumbai	Electronincs	790.0	28.0
11	Mumbai	Super Market	266.9	85.8

In [172]:
```
temp = stores.groupby(["Location","StoreType"])
res = temp[["TotalSales","OperatingCost"]].agg("sum").reset_index()
res = res.rename(columns = {"TotalSales":"SumOfTotalSales","OperatingCost":"SumOf
res
```

Out[172]:

	Location	StoreType	SumOfTotalSales	SumOfOperatingCost
0	Chennai	Apparel	915.4	44.8
1	Chennai	Electronincs	600.8	33.2
2	Chennai	Super Market	719.2	51.5
3	Delhi	Apparel	745.0	53.4
4	Delhi	Electronincs	306.7	45.4
5	Delhi	Super Market	726.0	62.9
6	Kolkata	Apparel	525.0	62.0
7	Kolkata	Electronincs	521.0	40.6
8	Kolkata	Super Market	566.4	72.2
9	Mumbai	Apparel	700.7	63.1
10	Mumbai	Electronincs	790.0	28.0
11	Mumbai	Super Market	266.9	85.8

In [174]:
```
temp = stores.groupby(["Location","StoreType"])
r = temp[["TotalSales","OperatingCost"]].agg({"sum","mean"}).reset_index()
r
```

Out[174]:

	Location	StoreType	TotalSales		OperatingCost	
			sum	**mean**	**sum**	**mean**
0	Chennai	Apparel	915.4	305.133333	44.8	14.933333
1	Chennai	Electronincs	600.8	300.400000	33.2	16.600000
2	Chennai	Super Market	719.2	239.733333	51.5	17.166667
3	Delhi	Apparel	745.0	248.333333	53.4	17.800000
4	Delhi	Electronincs	306.7	153.350000	45.4	22.700000
5	Delhi	Super Market	726.0	242.000000	62.9	20.966667
6	Kolkata	Apparel	525.0	175.000000	62.0	20.666667
7	Kolkata	Electronincs	521.0	260.500000	40.6	20.300000
8	Kolkata	Super Market	566.4	188.800000	72.2	24.066667
9	Mumbai	Apparel	700.7	233.566667	63.1	21.033333
10	Mumbai	Electronincs	790.0	395.000000	28.0	14.000000
11	Mumbai	Super Market	266.9	88.966667	85.8	28.600000

In [179]:
```
temp = stores.groupby(["Location","StoreType"])
r = temp[["TotalSales","OperatingCost"]].agg({"TotalSales":"sum","OperatingCost":
r = r.rename(columns = {"TotalSales":"SumOfToalSales","OperatingCost":"MeanOfOper
r
```

Out[179]:

	Location	StoreType	SumOfToalSales	MeanOfOperatingCost
0	Chennai	Apparel	915.4	14.933333
1	Chennai	Electronincs	600.8	16.600000
2	Chennai	Super Market	719.2	17.166667
3	Delhi	Apparel	745.0	17.800000
4	Delhi	Electronincs	306.7	22.700000
5	Delhi	Super Market	726.0	20.966667
6	Kolkata	Apparel	525.0	20.666667
7	Kolkata	Electronincs	521.0	20.300000
8	Kolkata	Super Market	566.4	24.066667
9	Mumbai	Apparel	700.7	21.033333
10	Mumbai	Electronincs	790.0	14.000000
11	Mumbai	Super Market	266.9	28.600000

In [187]:
```
#Finding top 5 customers with income by each category in SeriousDlqin2yrs

def topN(data, col, N):
    """
    Takes a dataframe, and returns N rows for given column (sorted by this column
    """
    #return data.sort_values(by=col, ascending=False).loc[:, col].head(N)
    return data.sort_values(by=col, ascending=False).head(N)
```

In [191]:
```
temp = stores.groupby("Location")
temp.apply(topN, col = "TotalSales", N = 2)
#          TotalSales
# Chennai  top1
#          top2
# Delhi    top1
#          top2
# Kolkata  top1
#          top2
# Mumbai   top1
#          top2
```

Out[191]:

		StoreCode	StoreName	StoreType	Location	OperatingCost	Staff_Cnt	TotalSales
Location								
Chennai	**14**	STR115	Fashion Bazar	Apparel	Chennai	10.4	80	472.0
	15	STR116	Digital Bazar	Electronincs	Chennai	10.4	80	460.0
Delhi	**4**	STR105	Central Store	Super Market	Delhi	18.7	80	360.0
	6	STR107	Fashion Bazar	Apparel	Delhi	14.3	80	360.0
Kolkata	**24**	STR125	Electronics Zone	Electronincs	Kolkata	19.2	80	400.0
	28	STR129	Central Store	Super Market	Kolkata	15.8	80	351.0
Mumbai	**16**	STR117	Electronics Zone	Electronincs	Mumbai	14.7	80	440.0
	23	STR124	Digital Bazar	Electronincs	Mumbai	13.3	80	350.0

In [192]:
```
# Merging
demographic = pd.read_csv("Demographic_Data.csv")
txn = pd.read_csv("Transaction_Summary.csv")
```

In [193]:
```
demographic
```

Out[193]:

	CustName	Gender	Age	Location	Salary	Education	Mobile
0	Alex	M	21	UK	19159	PhD	8834777722
1	Tom	M	34	USA	10461	X	8812237772
2	Michel	M	25	India	19961	XII	8834777232
3	Michael	M	28	Belgium	821	B.Com	8831234222
4	Patrik	M	29	Australia	13743	MBA	8823427722
5	Hans	M	34	Japan	25000	MS	8834772342
6	Biliana	F	26	Russia	3000	MS	8837427722
7	Raj	M	29	India	2000	MS	9892877722
8	Laila	F	26	India	4000	MBA	7867277722
9	Prabhas	F	35	India	20000	MBA	8834772321

In [194]:
```
txn
```

Out[194]:

	CustomerName	Total_Transaction_value	No_of_holding_prods	No_of_visits	No_of_channels
0	Alex	19159	3	3	1
1	Tom	10461	2	7	2
2	Michel	19961	4	8	3
3	Hans	821	1	2	2
4	Biliana	13743	6	9	3
5	Rajesh	25000	2	12	2
6	Laila	3000	1	6	1
7	Prabhas	2000	2	4	2
8	Ramu	30000	6	2	3

In [195]:
```
# Inner Join
pd.merge(left=demographic,right=txn,left_on="CustName",right_on="CustomerName",ho
```

Out[195]:

	CustName	Gender	Age	Location	Salary	Education	Mobile	CustomerName	Total_Transa
0	Alex	M	21	UK	19159	PhD	8834777722	Alex	
1	Tom	M	34	USA	10461	X	8812237772	Tom	
2	Michel	M	25	India	19961	XII	8834777232	Michel	
3	Hans	M	34	Japan	25000	MS	8834772342	Hans	
4	Biliana	F	26	Russia	3000	MS	8837427722	Biliana	
5	Laila	F	26	India	4000	MBA	7867277722	Laila	
6	Prabhas	F	35	India	20000	MBA	8834772321	Prabhas	

In [197]:
```
# Outer Join
pd.merge(left=demographic,right=txn,left_on="CustName",right_on="CustomerName",ho
```

Out[197]:

	CustName	Gender	Age	Location	Salary	Education	Mobile	CustomerName	Total_Tr
0	Alex	M	21.0	UK	19159.0	PhD	8.834778e+09	Alex	
1	Tom	M	34.0	USA	10461.0	X	8.812238e+09	Tom	
2	Michel	M	25.0	India	19961.0	XII	8.834777e+09	Michel	
3	Michael	M	28.0	Belgium	821.0	B.Com	8.831234e+09	NaN	
4	Patrik	M	29.0	Australia	13743.0	MBA	8.823428e+09	NaN	
5	Hans	M	34.0	Japan	25000.0	MS	8.834772e+09	Hans	
6	Biliana	F	26.0	Russia	3000.0	MS	8.837428e+09	Biliana	
7	Raj	M	29.0	India	2000.0	MS	9.892878e+09	NaN	
8	Laila	F	26.0	India	4000.0	MBA	7.867278e+09	Laila	
9	Prabhas	F	35.0	India	20000.0	MBA	8.834772e+09	Prabhas	
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Rajesh	
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Ramu	

In [198]:
```
# Left Join
pd.merge(left=demographic,right=txn,left_on="CustName",right_on="CustomerName",ho
```

Out[198]:

	CustName	Gender	Age	Location	Salary	Education	Mobile	CustomerName	Total_Transa
0	Alex	M	21	UK	19159	PhD	8834777722	Alex	
1	Tom	M	34	USA	10461	X	8812237772	Tom	
2	Michel	M	25	India	19961	XII	8834777232	Michel	
3	Michael	M	28	Belgium	821	B.Com	8831234222	NaN	
4	Patrik	M	29	Australia	13743	MBA	8823427722	NaN	
5	Hans	M	34	Japan	25000	MS	8834772342	Hans	
6	Biliana	F	26	Russia	3000	MS	8837427722	Biliana	
7	Raj	M	29	India	2000	MS	9892877722	NaN	
8	Laila	F	26	India	4000	MBA	7867277722	Laila	
9	Prabhas	F	35	India	20000	MBA	8834772321	Prabhas	

In [199]:
```
# Right Join
pd.merge(left=demographic,right=txn,left_on="CustName",right_on="CustomerName",ho
```

Out[199]:

	CustName	Gender	Age	Location	Salary	Education	Mobile	CustomerName	Total_Tra
0	Alex	M	21.0	UK	19159.0	PhD	8.834778e+09	Alex	
1	Tom	M	34.0	USA	10461.0	X	8.812238e+09	Tom	
2	Michel	M	25.0	India	19961.0	XII	8.834777e+09	Michel	
3	Hans	M	34.0	Japan	25000.0	MS	8.834772e+09	Hans	
4	Biliana	F	26.0	Russia	3000.0	MS	8.837428e+09	Biliana	
5	Laila	F	26.0	India	4000.0	MBA	7.867278e+09	Laila	
6	Prabhas	F	35.0	India	20000.0	MBA	8.834772e+09	Prabhas	
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Rajesh	
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Ramu	

In [200]:
```
print stores
```

```
   StoreCode         StoreName     StoreType Location  OperatingCost  \
0     STR101  Electronics Zone  Electronincs    Delhi           21.0
1     STR102      Apparel Zone       Apparel    Delhi           21.0
2     STR103       Super Bazar  Super Market    Delhi           22.8
3     STR104      Super Market  Super Market    Delhi           21.4
4     STR105     Central Store  Super Market    Delhi           18.7
5     STR106      Apparel Zone       Apparel    Delhi           18.1
6     STR107     Fashion Bazar       Apparel    Delhi           14.3
7     STR108     Digital Bazar  Electronincs    Delhi           24.4
8     STR109  Electronics Zone  Electronincs  Chennai           22.8
9     STR110      Apparel Zone       Apparel  Chennai           19.2
10    STR111       Super Bazar  Super Market  Chennai           17.8
11    STR112      Super Market  Super Market  Chennai           16.4
12    STR113     Central Store  Super Market  Chennai           17.3
13    STR114      Apparel Zone       Apparel  Chennai           15.2
14    STR115     Fashion Bazar       Apparel  Chennai           10.4
15    STR116     Digital Bazar  Electronincs  Chennai           10.4
16    STR117  Electronics Zone  Electronincs   Mumbai           14.7
17    STR118      Apparel Zone       Apparel   Mumbai           32.4
18    STR119       Super Bazar  Super Market   Mumbai           30.4
```

In [201]:
```
stores.to_excel("storesExport.xlsx")
```

In [206]:
```
from pandas import ExcelWriter
w = ExcelWriter("TestExport.xlsx")
```

In [207]:
```
stores.to_excel(w,sheet_name="stores first")
stores.to_excel(w,sheet_name="stores second")
stores.to_excel(w,sheet_name="stores third")
```

In [208]:
```
os.getcwd()
```

Out[208]: 'C:\\Users\\admin\\pandas\\DataSets'

In [209]:
```
w.close()
```

In [ ]:
```

```