Pandas

- Series

```
In [15]: import pandas as pd
         pd.__version__      #version is a attribute

Out[15]: '0.24.2'
```

```
In [16]: import numpy as np
         np.array([1,2,3])

Out[16]: array([1, 2, 3])
```

```
In [17]: s=pd.Series([1,2,3])
         s

Out[17]: 0    1
         1    2
         2    3
         dtype: int64
```

numpy positional based index, pandas have both positional based and label based indexing

```
In [18]: s1=pd.Series([1,2,3],index=['I','II','III'])
         s1

Out[18]: I      1
         II     2
         III    3
         dtype: int64
```

```
In [19]: print(s[0])
         print(s1[0],s1['I'])   #position indexing and Label indexing

         1
         1 1
```

```
In [20]: print(s1[0:2])

         I     1
         II    2
         dtype: int64
```

In [21]:
```
s1['I':'III']   #stop value also included
```

Out[21]:
```
I      1
II     2
III    3
dtype: int64
```

In [22]:
```
s.index=['i1','i2','i3']
s
```

Out[22]:
```
i1    1
i2    2
i3    3
dtype: int64
```

In [23]:
```
marks={'maths':67,'science':78,'english':56}
marks
```

Out[23]:
```
{'maths': 67, 'science': 78, 'english': 56}
```

In [24]:
```
m=pd.Series(marks)   #passing dictionary as i/p, with keys as index
m
```

Out[24]:
```
maths      67
science    78
english    56
dtype: int64
```

In [25]:
```
pd.date_range('2019-11-23','2019-11-30')
```

Out[25]:
```
DatetimeIndex(['2019-11-23', '2019-11-24', '2019-11-25', '2019-11-26',
               '2019-11-27', '2019-11-28', '2019-11-29', '2019-11-30'],
              dtype='datetime64[ns]', freq='D')
```

In [26]:
```
pd.date_range('23-11-2019','30-11-2019')
```

Out[26]:
```
DatetimeIndex(['2019-11-23', '2019-11-24', '2019-11-25', '2019-11-26',
               '2019-11-27', '2019-11-28', '2019-11-29', '2019-11-30'],
              dtype='datetime64[ns]', freq='D')
```

In [27]:
```
pd.date_range('23-11-2019',periods=5)
```

Out[27]:
```
DatetimeIndex(['2019-11-23', '2019-11-24', '2019-11-25', '2019-11-26',
               '2019-11-27'],
              dtype='datetime64[ns]', freq='D')
```

In [28]:
```
temp=pd.Series([32,29,30,31],pd.date_range('23-11-2019',periods=4))
```

```
In [29]: temp
```

```
Out[29]: 2019-11-23    32
         2019-11-24    29
         2019-11-25    30
         2019-11-26    31
         Freq: D, dtype: int64
```

```
In [30]: temp.index
```

```
Out[30]: DatetimeIndex(['2019-11-23', '2019-11-24', '2019-11-25', '2019-11-26'], dtype
         ='datetime64[ns]', freq='D')
```

```
In [31]: s.index
```

```
Out[31]: Index(['i1', 'i2', 'i3'], dtype='object')
```

```
In [32]: s1.index
```

```
Out[32]: Index(['I', 'II', 'III'], dtype='object')
```

```
In [33]: pd.Series(np.arange(10))  #creating pandas series from numpy arrays
```

```
Out[33]: 0    0
         1    1
         2    2
         3    3
         4    4
         5    5
         6    6
         7    7
         8    8
         9    9
         dtype: int32
```

Data Frame (table)

- From a Dictionary
- From numpy 2d array

In [34]:

```
studentmarks={"name":['meena','sai','gayatri','lokesh','swamy'],
              "maths":[90,89,87,76,90],
              "science":[98,97,98,99,100],
              "english":[90,89,87,87]
             }
pd.DataFrame(studentmarks)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-34-3a017a858d9d> in <module>
      4               "english":[90,89,87,87]
      5              }
----> 6 pd.DataFrame(studentmarks)

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __init__(self, data, inde
x, columns, dtype, copy)
    390                                  dtype=dtype, copy=copy)
    391         elif isinstance(data, dict):
--> 392             mgr = init_dict(data, index, columns, dtype=dtype)
    393         elif isinstance(data, ma.MaskedArray):
    394             import numpy.ma.mrecords as mrecords

~\Anaconda3\lib\site-packages\pandas\core\internals\construction.py in init_dic
t(data, index, columns, dtype)
    210         arrays = [data[k] for k in keys]
    211 
--> 212     return arrays_to_mgr(arrays, data_names, index, columns, dtype=dtyp
e)
    213 
    214 

~\Anaconda3\lib\site-packages\pandas\core\internals\construction.py in arrays_t
o_mgr(arrays, arr_names, index, columns, dtype)
     49     # figure out the index, if necessary
     50     if index is None:
---> 51         index = extract_index(arrays)
     52     else:
     53         index = ensure_index(index)

~\Anaconda3\lib\site-packages\pandas\core\internals\construction.py in extract_
index(data)
    315             lengths = list(set(raw_lengths))
    316             if len(lengths) > 1:
--> 317                 raise ValueError('arrays must all be same length')
    318 
    319             if have_dicts:

ValueError: arrays must all be same length
```

In [35]:

```
studentmarks={"name":['meena','sai','gayatri','lokesh','swamy'],
              "maths":[90,89,87,76,90],
              "science":[98,97,98,99,100],
              "english":[90,89,87,86,87]
             }    #here all values must be equal length, 5 names, marks for 5 st
```

In [36]:
```
pd.DataFrame(studentmarks)  #here dictionary keys as taken as column names
```

Out[36]:

	name	maths	science	english
0	meena	90	98	90
1	sai	89	97	89
2	gayatri	87	98	87
3	lokesh	76	99	86
4	swamy	90	100	87

In [37]:
```
a2=np.array([['meena',90,98,90],['sai',89,97,89]])
pd.DataFrame(a2)
```

Out[37]:

	0	1	2	3
0	meena	90	98	90
1	sai	89	97	89

In [38]:
```
pd.DataFrame(a2,columns=['name','maths','science','english'])
```

Out[38]:

	name	maths	science	english
0	meena	90	98	90
1	sai	89	97	89

In [27]:
```
pd.DataFrame(a2,columns=['name','maths','science','english'],index=['std1','std2
```

Out[27]:

	name	maths	science	english
std1	meena	90	98	90
std2	sai	89	97	89

pd.read_csv() --> To read csv,tsv files

```
In [28]: pd.read_csv("marks.csv")   #bydefault it taking 1st row as headings
```

```
Out[28]:
          'meena'  90  98  90.1
0           'sai'  89  97  89.0
1       'gayatri'  87  98  87.0
2        'lokesh'  76  99  86.0
3         'swamy'  90  87   NaN
4   'ramalakshmi'  90  87  89.0
5         'bhanu'  90  87  76.0
6          'hima'  90  87  76.0
7     'chandrika'  89  76  85.0
8       'keerthi'  89  67  65.0
```

```
In [29]: pd.read_csv("marks.csv",header=None)
```

```
Out[29]:
                0   1   2     3
0         'meena'  90  98  90.0
1           'sai'  89  97  89.0
2       'gayatri'  87  98  87.0
3        'lokesh'  76  99  86.0
4         'swamy'  90  87   NaN
5   'ramalakshmi'  90  87  89.0
6         'bhanu'  90  87  76.0
7          'hima'  90  87  76.0
8     'chandrika'  89  76  85.0
9       'keerthi'  89  67  65.0
```

```
In [30]: pd.read_csv("marks.csv",header=1)    #taking row with with 1 index as heading
```

Out[30]:

	'sai'	89	97	89.1
0	'gayatri'	87	98	87.0
1	'lokesh'	76	99	86.0
2	'swamy'	90	87	NaN
3	'ramalakshmi'	90	87	89.0
4	'bhanu'	90	87	76.0
5	'hima'	90	87	76.0
6	'chandrika'	89	76	85.0
7	'keerthi'	89	67	65.0

```
In [31]: pd.read_csv("marks.csv",names=['name','science','maths','english'])    #names to
```

Out[31]:

	name	science	maths	english
0	'meena'	90	98	90.0
1	'sai'	89	97	89.0
2	'gayatri'	87	98	87.0
3	'lokesh'	76	99	86.0
4	'swamy'	90	87	NaN
5	'ramalakshmi'	90	87	89.0
6	'bhanu'	90	87	76.0
7	'hima'	90	87	76.0
8	'chandrika'	89	76	85.0
9	'keerthi'	89	67	65.0

In [32]:

```
help(pd.read_csv)
```

```
Help on function read_csv in module pandas.io.parsers:

read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer', names=N
one, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_co
ls=True, dtype=None, engine=None, converters=None, true_values=None, false_va
lues=None, skipinitialspace=False, skiprows=None, skipfooter=0, nrows=None, n
a_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blan
k_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=F
alse, date_parser=None, dayfirst=False, iterator=False, chunksize=None, compr
ession='infer', thousands=None, decimal=b'.', lineterminator=None, quotechar
='"', quoting=0, doublequote=True, escapechar=None, comment=None, encoding=No
ne, dialect=None, tupleize_cols=None, error_bad_lines=True, warn_bad_lines=Tr
ue, delim_whitespace=False, low_memory=True, memory_map=False, float_precisio
n=None)
    Read a comma-separated values (csv) file into DataFrame.

    Also supports optionally iterating or breaking of the file
    into chunks.
```

In [33]:

```
pd.read_csv("marks.csv",
            names=['name','science','maths','english'],
           usecols=['name','science'])  #usecols is to access only specified colu
```

Out[33]:

	name	science
0	'meena'	90
1	'sai'	89
2	'gayatri'	87
3	'lokesh'	76
4	'swamy'	90
5	'ramalakshmi'	90
6	'bhanu'	90
7	'hima'	90
8	'chandrika'	89
9	'keerthi'	89

In [41]:

```
df=pd.read_csv("marks.csv",
            names=['student name','science','maths','english'])
```

In [42]:
```
df  #it will take missing value as NaN (Not a Number)
```

Out[42]:

	student name	science	maths	english
0	'meena'	90	98	90.0
1	'sai'	89	97	89.0
2	'gayatri'	87	98	87.0
3	'lokesh'	76	99	86.0
4	'swamy'	90	87	NaN
5	'ramalakshmi'	90	87	89.0
6	'bhanu'	90	87	76.0
7	'hima'	90	87	76.0
8	'chandrika'	89	76	85.0
9	'keerthi'	89	67	65.0

In [43]:
```
df.student-name  # error while using dot space or - not acceptable
                 #student-name ,student  name not acceptable,   studentname is ac
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-43-2e03603c112d> in <module>
----> 1 df.student-name  # error while using dot space or - not acceptable
      2                  #student-name ,student  name not acceptable,   student
name is acceptable

~\Anaconda3\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5065             if self._info_axis._can_hold_identifiers_and_holds_name(nam
e):
   5066                 return self[name]
-> 5067             return object.__getattribute__(self, name)
   5068 
   5069     def __setattr__(self, name, value):

AttributeError: 'DataFrame' object has no attribute 'student'
```

In [ ]:
```
df['student name']  #to access a particular column name   #pandas series
```

In [ ]:
```
df.columns
```

In [44]:
```
df[['student name']]  ##data frame
```

Out[44]:

	student name
0	'meena'
1	'sai'
2	'gayatri'
3	'lokesh'
4	'swamy'
5	'ramalakshmi'
6	'bhanu'
7	'hima'
8	'chandrika'
9	'keerthi'

In [45]:
```
df[['student name','maths']]
```

Out[45]:

	student name	maths
0	'meena'	98
1	'sai'	97
2	'gayatri'	98
3	'lokesh'	99
4	'swamy'	87
5	'ramalakshmi'	87
6	'bhanu'	87
7	'hima'	87
8	'chandrika'	76
9	'keerthi'	67

In [46]:
```
df.values   #2D numpy array
```

Out[46]:
```
array([["'meena'", 90, 98, 90.0],
       ["'sai'", 89, 97, 89.0],
       ["'gayatri'", 87, 98, 87.0],
       ["'lokesh'", 76, 99, 86.0],
       ["'swamy'", 90, 87, nan],
       ["'ramalakshmi'", 90, 87, 89.0],
       ["'bhanu'", 90, 87, 76.0],
       ["'hima'", 90, 87, 76.0],
       ["'chandrika'", 89, 76, 85.0],
       ["'keerthi'", 89, 67, 65.0]], dtype=object)
```

In [47]:
```
df.dtypes        #to know datatypes  #english is float,we have missing value in
```

Out[47]:
```
student name     object
science           int64
maths             int64
english         float64
dtype: object
```

In [48]:
```
df.index
```

Out[48]:
```
RangeIndex(start=0, stop=10, step=1)
```

In [49]:
```
df.shape  #tuple of no of rows ,columns
```

Out[49]:
```
(10, 4)
```

In [50]:
```
#no of rows
print(df.shape[0])
#no of columns
print(df.shape[1])
```

```
10
4
```

Indexing

- Position based Indexing(iloc)
- Label based Indexing(loc)

In [51]:
```
df['student name']
```

Out[51]:
```
0          'meena'
1            'sai'
2        'gayatri'
3         'lokesh'
4          'swamy'
5    'ramalakshmi'
6          'bhanu'
7           'hima'
8      'chandrika'
9        'keerthi'
Name: student name, dtype: object
```

In [52]:
```
df.iloc[4,0:2]
```

Out[52]:
```
student name    'swamy'
science              90
Name: 4, dtype: object
```

In [53]:
```
df.iloc[4:6,:]  #row with index 4 & 5 and : for all columns
```

Out[53]:

	student name	science	maths	english
4	'swamy'	90	87	NaN
5	'ramalakshmi'	90	87	89.0

In [54]:
```
df.iloc[2,3]  #3rd row, 4th column
```

Out[54]:
```
87.0
```

In [55]:
```
df.iloc[2,:]  #df.iloc[2,]
```

Out[55]:
```
student name    'gayatri'
science                87
maths                  98
english                87
Name: 2, dtype: object
```

In [56]:
```
df.iloc[:,1]  #2nd column
```

Out[56]:
```
0    90
1    89
2    87
3    76
4    90
5    90
6    90
7    90
8    89
9    89
Name: science, dtype: int64
```

In [57]:
```
df
```

Out[57]:

	student name	science	maths	english
0	'meena'	90	98	90.0
1	'sai'	89	97	89.0
2	'gayatri'	87	98	87.0
3	'lokesh'	76	99	86.0
4	'swamy'	90	87	NaN
5	'ramalakshmi'	90	87	89.0
6	'bhanu'	90	87	76.0
7	'hima'	90	87	76.0
8	'chandrika'	89	76	85.0
9	'keerthi'	89	67	65.0

In [58]:
```
df.iloc[[1,3,6],[2,0,1]]
```

Out[58]:

	maths	student name	science
1	97	'sai'	89
3	99	'lokesh'	76
6	87	'bhanu'	90

In [59]:
```
df.loc[0,'student name']
```

Out[59]:
```
"'meena'"
```

In [54]:
```
df.loc[2:4,'student name':'maths']  #slicing in the same order in original data
```

Out[54]:

	student name	science	maths
2	'gayatri'	87	98
3	'lokesh'	76	99
4	'swamy'	90	87

In [55]:
```
df.loc[2:4,'science':'student name']  #reverse is not possible
```

Out[55]:

2
3
4

In [56]:
```
df.loc[2:4,['science','student name']]   #to access multiple cols use lists
```

Out[56]:

	science	student name
2	87	'gayatri'
3	76	'lokesh'
4	90	'swamy'

In [57]:
```
df[['student name','science','maths','english']]
```

Out[57]:

	student name	science	maths	english
0	'meena'	90	98	90.0
1	'sai'	89	97	89.0
2	'gayatri'	87	98	87.0
3	'lokesh'	76	99	86.0
4	'swamy'	90	87	NaN
5	'ramalakshmi'	90	87	89.0
6	'bhanu'	90	87	76.0
7	'hima'	90	87	76.0
8	'chandrika'	89	76	85.0
9	'keerthi'	89	67	65.0

In [58]:
```
df1=df.set_index('student name')  #setting index
df1
```

Out[58]:

	science	maths	english
student name			
'meena'	90	98	90.0
'sai'	89	97	89.0
'gayatri'	87	98	87.0
'lokesh'	76	99	86.0
'swamy'	90	87	NaN
'ramalakshmi'	90	87	89.0
'bhanu'	90	87	76.0
'hima'	90	87	76.0
'chandrika'	89	76	85.0
'keerthi'	89	67	65.0

In [59]:
```
df  #original data not modified
```

Out[59]:

	student name	science	maths	english
0	'meena'	90	98	90.0
1	'sai'	89	97	89.0
2	'gayatri'	87	98	87.0
3	'lokesh'	76	99	86.0
4	'swamy'	90	87	NaN
5	'ramalakshmi'	90	87	89.0
6	'bhanu'	90	87	76.0
7	'hima'	90	87	76.0
8	'chandrika'	89	76	85.0
9	'keerthi'	89	67	65.0

In [60]:
```
df1.loc["'meena'",'science']
```

Out[60]:
```
90
```

In [61]:
```
df.info()  #information about our data frame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
student name    10 non-null object
science         10 non-null int64
maths           10 non-null int64
english         9 non-null float64
dtypes: float64(1), int64(2), object(1)
memory usage: 400.0+ bytes
```

In [62]:

```
df['Total']=df['english']+df['science']+df['maths']
df
```

Out[62]:

	student name	science	maths	english	Total
0	'meena'	90	98	90.0	278.0
1	'sai'	89	97	89.0	275.0
2	'gayatri'	87	98	87.0	272.0
3	'lokesh'	76	99	86.0	261.0
4	'swamy'	90	87	NaN	NaN
5	'ramalakshmi'	90	87	89.0	266.0
6	'bhanu'	90	87	76.0	253.0
7	'hima'	90	87	76.0	253.0
8	'chandrika'	89	76	85.0	250.0
9	'keerthi'	89	67	65.0	221.0

In [60]:

```
df.sum() #column wise sum
```

Out[60]:

```
student name    'meena''sai''gayatri''lokesh''swamy''ramalaksh...
science                                                       880
maths                                                         883
english                                                       743
dtype: object
```

In [64]:

```
df.iloc[:,1:4].sum(axis=1)  #axis=1 means col wise
```

Out[64]:

```
0    278.0
1    275.0
2    272.0
3    261.0
4    177.0
5    266.0
6    253.0
7    253.0
8    250.0
9    221.0
dtype: float64
```

In [65]:
```
df
```

Out[65]:

	student name	science	maths	english	Total
0	'meena'	90	98	90.0	278.0
1	'sai'	89	97	89.0	275.0
2	'gayatri'	87	98	87.0	272.0
3	'lokesh'	76	99	86.0	261.0
4	'swamy'	90	87	NaN	NaN
5	'ramalakshmi'	90	87	89.0	266.0
6	'bhanu'	90	87	76.0	253.0
7	'hima'	90	87	76.0	253.0
8	'chandrika'	89	76	85.0	250.0
9	'keerthi'	89	67	65.0	221.0

In [66]:
```
df.describe()   #dataframe description
```

Out[66]:

	science	maths	english	Total
count	10.000000	10.000000	9.000000	9.000000
mean	88.000000	88.300000	82.555556	258.777778
std	4.320494	10.488618	8.442617	17.448336
min	76.000000	67.000000	65.000000	221.000000
25%	89.000000	87.000000	76.000000	253.000000
50%	89.500000	87.000000	86.000000	261.000000
75%	90.000000	97.750000	89.000000	272.000000
max	90.000000	99.000000	90.000000	278.000000

In [67]:
```
df.describe(include=object)
```

Out[67]:

	student name
count	10
unique	10
top	'sai'
freq	1

In [68]:
```
df.describe(include=np.int64)
```

Out[68]:

	science	maths
count	10.000000	10.000000
mean	88.000000	88.300000
std	4.320494	10.488618
min	76.000000	67.000000
25%	89.000000	87.000000
50%	89.500000	87.000000
75%	90.000000	97.750000
max	90.000000	99.000000

In [69]:
```
student_df=pd.read_csv("student.csv",names=['student name','college','course'])
student_df
```

Out[69]:

	student name	college	course
0	'meena'	VRSEC	ML
1	'sai'	LPU	AGBSC
2	'gayatri'	KBN	TESTING
3	'lokesh'	VRSEC	IOT
4	'swamy'	IIT	BIG DATA
5	'swathi'	VRSEC	SELENIUM
6	'sarayu'	MIC	IOT
7	'neelu'	VVIT	HADOOP

Check the

- head
- tail
- shape
- columns
- dtypes
- index
- info
- describe

In [70]:
```
student_df.head(1) #rows from top
```

Out[70]:

	student name	college	course
0	'meena'	VRSEC	ML

In [71]:
```
student_df.tail(1)  #rows from bottom
```

Out[71]:

	student name	college	course
7	'neelu'	VVIT	HADOOP

In [72]:
```
student_df.sample(4)   #randomly select no of rows mentioned
```

Out[72]:

	student name	college	course
7	'neelu'	VVIT	HADOOP
0	'meena'	VRSEC	ML
2	'gayatri'	KBN	TESTING
3	'lokesh'	VRSEC	IOT

In [73]:
```
student_df.shape  #no of rows,cols
```

Out[73]:
```
(8, 3)
```

In [74]:
```
student_df.columns
```

Out[74]:
```
Index(['student name', 'college', 'course'], dtype='object')
```

In [75]:
```
student_df.dtypes
```

Out[75]:
```
student name    object
college         object
course          object
dtype: object
```

In [76]:
```
student_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 3 columns):
student name    8 non-null object
college         8 non-null object
course          8 non-null object
dtypes: object(3)
memory usage: 272.0+ bytes
```

In [77]:
```
student_df.describe()   #unique-unique no of entries, top: most repeated entries
```

Out[77]:

	student name	college	course
count	8	8	8
unique	8	6	7
top	'sai'	VRSEC	IOT
freq	1	3	2

In [78]:
```
help(df.describe())
```

```
Help on DataFrame in module pandas.core.frame object:

class DataFrame(pandas.core.generic.NDFrame)
 |  DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
 |  
 |  Two-dimensional size-mutable, potentially heterogeneous tabular data
 |  structure with labeled axes (rows and columns). Arithmetic operations
 |  align on both row and column labels. Can be thought of as a dict-like
 |  container for Series objects. The primary pandas data structure.
 |  
 |  Parameters
 |  ----------
 |  data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
 |      Dict can contain Series, arrays, constants, or list-like objects
 |  
 |      .. versionchanged :: 0.23.0
 |         If data is a dict, argument order is maintained for Python 3.6
 |         and later.
 |  
 |  index : Index or array-like
```

In [79]:
```
student_df.nunique()
```

Out[79]:
```
student name    8
college         6
course          7
dtype: int64
```

In [80]:
```
student_df.college.value_counts()
```

Out[80]:
```
VRSEC    3
IIT      1
VVIT     1
KBN      1
MIC      1
LPU      1
Name: college, dtype: int64
```

pd.merge

In [81]:
```
pd.merge(df,student_df,on='student name')  #inner join , on=which column bases j
```

Out[81]:

	student name	science	maths	english	Total	college	course
0	'meena'	90	98	90.0	278.0	VRSEC	ML
1	'sai'	89	97	89.0	275.0	LPU	AGBSC
2	'gayatri'	87	98	87.0	272.0	KBN	TESTING
3	'lokesh'	76	99	86.0	261.0	VRSEC	IOT
4	'swamy'	90	87	NaN	NaN	IIT	BIG DATA

In [82]:
```
pd.merge(df,student_df,on='student name',how="outer")    #outer-->union, default
```

Out[82]:

	student name	science	maths	english	Total	college	course
0	'meena'	90.0	98.0	90.0	278.0	VRSEC	ML
1	'sai'	89.0	97.0	89.0	275.0	LPU	AGBSC
2	'gayatri'	87.0	98.0	87.0	272.0	KBN	TESTING
3	'lokesh'	76.0	99.0	86.0	261.0	VRSEC	IOT
4	'swamy'	90.0	87.0	NaN	NaN	IIT	BIG DATA
5	'ramalakshmi'	90.0	87.0	89.0	266.0	NaN	NaN
6	'bhanu'	90.0	87.0	76.0	253.0	NaN	NaN
7	'hima'	90.0	87.0	76.0	253.0	NaN	NaN
8	'chandrika'	89.0	76.0	85.0	250.0	NaN	NaN
9	'keerthi'	89.0	67.0	65.0	221.0	NaN	NaN
10	'swathi'	NaN	NaN	NaN	NaN	VRSEC	SELENIUM
11	'sarayu'	NaN	NaN	NaN	NaN	MIC	IOT
12	'neelu'	NaN	NaN	NaN	NaN	VVIT	HADOOP

In [83]:
```
pd.merge(df,student_df,on='student name',how="left")
```

Out[83]:

	student name	science	maths	english	Total	college	course
0	'meena'	90	98	90.0	278.0	VRSEC	ML
1	'sai'	89	97	89.0	275.0	LPU	AGBSC
2	'gayatri'	87	98	87.0	272.0	KBN	TESTING
3	'lokesh'	76	99	86.0	261.0	VRSEC	IOT
4	'swamy'	90	87	NaN	NaN	IIT	BIG DATA
5	'ramalakshmi'	90	87	89.0	266.0	NaN	NaN
6	'bhanu'	90	87	76.0	253.0	NaN	NaN
7	'hima'	90	87	76.0	253.0	NaN	NaN
8	'chandrika'	89	76	85.0	250.0	NaN	NaN
9	'keerthi'	89	67	65.0	221.0	NaN	NaN

In [84]:
```
pd.merge(df,student_df,on='student name',how="right")
```

Out[84]:

	student name	science	maths	english	Total	college	course
0	'meena'	90.0	98.0	90.0	278.0	VRSEC	ML
1	'sai'	89.0	97.0	89.0	275.0	LPU	AGBSC
2	'gayatri'	87.0	98.0	87.0	272.0	KBN	TESTING
3	'lokesh'	76.0	99.0	86.0	261.0	VRSEC	IOT
4	'swamy'	90.0	87.0	NaN	NaN	IIT	BIG DATA
5	'swathi'	NaN	NaN	NaN	NaN	VRSEC	SELENIUM
6	'sarayu'	NaN	NaN	NaN	NaN	MIC	IOT
7	'neelu'	NaN	NaN	NaN	NaN	VVIT	HADOOP

Boolean or Fancy Indexing

In [85]:
```
df['maths']>60
```

Out[85]:
```
0    True
1    True
2    True
3    True
4    True
5    True
6    True
7    True
8    True
9    True
Name: maths, dtype: bool
```

In [86]:
```
df[df['maths']>60]
```

Out[86]:

	student name	science	maths	english	Total
0	'meena'	90	98	90.0	278.0
1	'sai'	89	97	89.0	275.0
2	'gayatri'	87	98	87.0	272.0
3	'lokesh'	76	99	86.0	261.0
4	'swamy'	90	87	NaN	NaN
5	'ramalakshmi'	90	87	89.0	266.0
6	'bhanu'	90	87	76.0	253.0
7	'hima'	90	87	76.0	253.0
8	'chandrika'	89	76	85.0	250.0
9	'keerthi'	89	67	65.0	221.0

In [87]:
```
df[(df['maths']>60) & (df['maths']<80)]
```

Out[87]:

	student name	science	maths	english	Total
8	'chandrika'	89	76	85.0	250.0
9	'keerthi'	89	67	65.0	221.0

In [88]:
```
df[(df['maths']>60) & (df['english']>50)]['science']
```

Out[88]:
```
0    90
1    89
2    87
3    76
5    90
6    90
7    90
8    89
9    89
Name: science, dtype: int64
```

In [89]:
```
df['science'][(df['maths']>60) & (df['english']>50)]
```

Out[89]:
```
0    90
1    89
2    87
3    76
5    90
6    90
7    90
8    89
9    89
Name: science, dtype: int64
```

In [90]:
```
student_df['course']=='IOT'
```

Out[90]:
```
0    False
1    False
2    False
3     True
4    False
5    False
6     True
7    False
Name: course, dtype: bool
```

In [91]:
```
student_df[student_df['course']=='IOT']
```

Out[91]:

	student name	college	course
3	'lokesh'	VRSEC	IOT
6	'sarayu'	MIC	IOT

In [92]:
```
student_df[(student_df['college']=='VRSEC')&(student_df['course']=='IOT')]
```

Out[92]:

	student name	college	course
3	'lokesh'	VRSEC	IOT

In [93]:
```
student_df
```

Out[93]:

	student name	college	course
0	'meena'	VRSEC	ML
1	'sai'	LPU	AGBSC
2	'gayatri'	KBN	TESTING
3	'lokesh'	VRSEC	IOT
4	'swamy'	IIT	BIG DATA
5	'swathi'	VRSEC	SELENIUM
6	'sarayu'	MIC	IOT
7	'neelu'	VVIT	HADOOP

In [94]:

```
df[df['Total']>250]['student name']
```

Out[94]:

```
0           'meena'
1             'sai'
2         'gayatri'
3          'lokesh'
5     'ramalakshmi'
6           'bhanu'
7            'hima'
Name: student name, dtype: object
```

In [95]:

```
#To retrieve student name,course ,college based on total >260

data=pd.merge(df,student_df,on='student name',how='outer')
print(data)
data[data['Total']>260][['student name','course','college']]
```

```
     student name  science  maths  english  Total college     course
0         'meena'     90.0   98.0     90.0  278.0   VRSEC         ML
1           'sai'     89.0   97.0     89.0  275.0     LPU      AGBSC
2       'gayatri'     87.0   98.0     87.0  272.0     KBN    TESTING
3        'lokesh'     76.0   99.0     86.0  261.0   VRSEC        IOT
4         'swamy'     90.0   87.0      NaN    NaN     IIT   BIG DATA
5   'ramalakshmi'     90.0   87.0     89.0  266.0     NaN        NaN
6         'bhanu'     90.0   87.0     76.0  253.0     NaN        NaN
7          'hima'     90.0   87.0     76.0  253.0     NaN        NaN
8     'chandrika'     89.0   76.0     85.0  250.0     NaN        NaN
9       'keerthi'     89.0   67.0     65.0  221.0     NaN        NaN
10       'swathi'      NaN    NaN      NaN    NaN   VRSEC   SELENIUM
11       'sarayu'      NaN    NaN      NaN    NaN     MIC        IOT
12        'neelu'      NaN    NaN      NaN    NaN    VVIT     HADOOP
```

Out[95]:

	student name	course	college
0	'meena'	ML	VRSEC
1	'sai'	AGBSC	LPU
2	'gayatri'	TESTING	KBN
3	'lokesh'	IOT	VRSEC
5	'ramalakshmi'	NaN	NaN

Sorting

In [96]:
```
df.sort_values('maths')  #by default ascending is true, incremental order
```

Out[96]:

	student name	science	maths	english	Total
9	'keerthi'	89	67	65.0	221.0
8	'chandrika'	89	76	85.0	250.0
4	'swamy'	90	87	NaN	NaN
5	'ramalakshmi'	90	87	89.0	266.0
6	'bhanu'	90	87	76.0	253.0
7	'hima'	90	87	76.0	253.0
1	'sai'	89	97	89.0	275.0
0	'meena'	90	98	90.0	278.0
2	'gayatri'	87	98	87.0	272.0
3	'lokesh'	76	99	86.0	261.0

In [97]:
```
df.sort_values('maths',ascending=False)
```

Out[97]:

	student name	science	maths	english	Total
3	'lokesh'	76	99	86.0	261.0
0	'meena'	90	98	90.0	278.0
2	'gayatri'	87	98	87.0	272.0
1	'sai'	89	97	89.0	275.0
4	'swamy'	90	87	NaN	NaN
5	'ramalakshmi'	90	87	89.0	266.0
6	'bhanu'	90	87	76.0	253.0
7	'hima'	90	87	76.0	253.0
8	'chandrika'	89	76	85.0	250.0
9	'keerthi'	89	67	65.0	221.0

In [98]:

```
df.sort_values('student name')  #printing student names in alphabetical order
```

Out[98]:

	student name	science	maths	english	Total
6	'bhanu'	90	87	76.0	253.0
8	'chandrika'	89	76	85.0	250.0
2	'gayatri'	87	98	87.0	272.0
7	'hima'	90	87	76.0	253.0
9	'keerthi'	89	67	65.0	221.0
3	'lokesh'	76	99	86.0	261.0
0	'meena'	90	98	90.0	278.0
5	'ramalakshmi'	90	87	89.0	266.0
1	'sai'	89	97	89.0	275.0
4	'swamy'	90	87	NaN	NaN

In [99]:

```
df.sort_index()   #index based sorting
```

Out[99]:

	student name	science	maths	english	Total
0	'meena'	90	98	90.0	278.0
1	'sai'	89	97	89.0	275.0
2	'gayatri'	87	98	87.0	272.0
3	'lokesh'	76	99	86.0	261.0
4	'swamy'	90	87	NaN	NaN
5	'ramalakshmi'	90	87	89.0	266.0
6	'bhanu'	90	87	76.0	253.0
7	'hima'	90	87	76.0	253.0
8	'chandrika'	89	76	85.0	250.0
9	'keerthi'	89	67	65.0	221.0

Missing Values

In [100]:
```
df.isnull()  #to identify missing values
```

Out[100]:

	student name	science	maths	english	Total
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	True	True
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False

In [101]:
```
df.isna()  #to identify missing values
```

Out[101]:

	student name	science	maths	english	Total
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	True	True
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False

In [102]:
```
df.isnull().sum()  #missing values count
```

Out[102]:
```
student name    0
science         0
maths           0
english         1
Total           1
dtype: int64
```

In [103]:

```
print(df.science.min(),df.science.max())
print(df.science.mean())
print(df.science.median())
print(df.science.mode())
```

```
76 90
88.0
89.5
0    90
dtype: int64
```

In [104]:

```
df.science.value_counts()
```

Out[104]:

```
90    5
89    3
76    1
87    1
Name: science, dtype: int64
```

In [105]:

```
df['english'].fillna(df['english'].mean())
```

Out[105]:

```
0    90.000000
1    89.000000
2    87.000000
3    86.000000
4    82.555556
5    89.000000
6    76.000000
7    76.000000
8    85.000000
9    65.000000
Name: english, dtype: float64
```

In [106]:

```
df
```

Out[106]:

	student name	science	maths	english	Total
0	'meena'	90	98	90.0	278.0
1	'sai'	89	97	89.0	275.0
2	'gayatri'	87	98	87.0	272.0
3	'lokesh'	76	99	86.0	261.0
4	'swamy'	90	87	NaN	NaN
5	'ramalakshmi'	90	87	89.0	266.0
6	'bhanu'	90	87	76.0	253.0
7	'hima'	90	87	76.0	253.0
8	'chandrika'	89	76	85.0	250.0
9	'keerthi'	89	67	65.0	221.0

median--->Outlier

- Based on missing values
- delete row or columns
- replace with numerical=mean,median,catogorical-mode

In [107]:

```
df['english'].fillna(df['english'].mean(),inplace=True)  #to replace in original
df                                                       #save memory
```

Out[107]:

	student name	science	maths	english	Total
0	'meena'	90	98	90.000000	278.0
1	'sai'	89	97	89.000000	275.0
2	'gayatri'	87	98	87.000000	272.0
3	'lokesh'	76	99	86.000000	261.0
4	'swamy'	90	87	82.555556	NaN
5	'ramalakshmi'	90	87	89.000000	266.0
6	'bhanu'	90	87	76.000000	253.0
7	'hima'	90	87	76.000000	253.0
8	'chandrika'	89	76	85.000000	250.0
9	'keerthi'	89	67	65.000000	221.0

In [108]:

```
df2=pd.read_csv("marks.csv",                                              #accessing da
                names=['student name','science','maths','english'])
```

In [109]:

```
df2  #data with
```

Out[109]:

	student name	science	maths	english
0	'meena'	90	98	90.0
1	'sai'	89	97	89.0
2	'gayatri'	87	98	87.0
3	'lokesh'	76	99	86.0
4	'swamy'	90	87	NaN
5	'ramalakshmi'	90	87	89.0
6	'bhanu'	90	87	76.0
7	'hima'	90	87	76.0
8	'chandrika'	89	76	85.0
9	'keerthi'	89	67	65.0

In [110]:
```
df2.dropna()    #deleting missing value row
```

Out[110]:

	student name	science	maths	english
0	'meena'	90	98	90.0
1	'sai'	89	97	89.0
2	'gayatri'	87	98	87.0
3	'lokesh'	76	99	86.0
5	'ramalakshmi'	90	87	89.0
6	'bhanu'	90	87	76.0
7	'hima'	90	87	76.0
8	'chandrika'	89	76	85.0
9	'keerthi'	89	67	65.0

In [111]:
```
df2.dropna(axis=1)  #deleting missing value column
```

Out[111]:

	student name	science	maths
0	'meena'	90	98
1	'sai'	89	97
2	'gayatri'	87	98
3	'lokesh'	76	99
4	'swamy'	90	87
5	'ramalakshmi'	90	87
6	'bhanu'	90	87
7	'hima'	90	87
8	'chandrika'	89	76
9	'keerthi'	89	67

In [112]:
```
df2.drop('maths',axis=1)   #deleting particular column
```

Out[112]:

	student name	science	english
0	'meena'	90	90.0
1	'sai'	89	89.0
2	'gayatri'	87	87.0
3	'lokesh'	76	86.0
4	'swamy'	90	NaN
5	'ramalakshmi'	90	89.0
6	'bhanu'	90	76.0
7	'hima'	90	76.0
8	'chandrika'	89	85.0
9	'keerthi'	89	65.0

In [113]:
```
final_df=pd.merge(df,student_df,on='student name',how='outer')
```

In [114]:
```
final_df
```

Out[114]:

	student name	science	maths	english	Total	college	course
0	'meena'	90.0	98.0	90.000000	278.0	VRSEC	ML
1	'sai'	89.0	97.0	89.000000	275.0	LPU	AGBSC
2	'gayatri'	87.0	98.0	87.000000	272.0	KBN	TESTING
3	'lokesh'	76.0	99.0	86.000000	261.0	VRSEC	IOT
4	'swamy'	90.0	87.0	82.555556	NaN	IIT	BIG DATA
5	'ramalakshmi'	90.0	87.0	89.000000	266.0	NaN	NaN
6	'bhanu'	90.0	87.0	76.000000	253.0	NaN	NaN
7	'hima'	90.0	87.0	76.000000	253.0	NaN	NaN
8	'chandrika'	89.0	76.0	85.000000	250.0	NaN	NaN
9	'keerthi'	89.0	67.0	65.000000	221.0	NaN	NaN
10	'swathi'	NaN	NaN	NaN	NaN	VRSEC	SELENIUM
11	'sarayu'	NaN	NaN	NaN	NaN	MIC	IOT
12	'neelu'	NaN	NaN	NaN	NaN	VVIT	HADOOP

In [115]:

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13 entries, 0 to 12
Data columns (total 7 columns):
student name    13 non-null object
science         10 non-null float64
maths           10 non-null float64
english         10 non-null float64
Total           9 non-null float64
college         8 non-null object
course          8 non-null object
dtypes: float64(4), object(3)
memory usage: 832.0+ bytes
```

In [116]:

```
final_df.isnull().sum()   #to findout column wise missing values count
```

Out[116]:

```
student name    0
science         3
maths           3
english         3
Total           4
college         5
course          5
dtype: int64
```

In [117]:

```
final_df.columns
```

Out[117]:

```
Index(['student name', 'science', 'maths', 'english', 'Total', 'college',
       'course'],
      dtype='object')
```

In [118]:

```
final_df.dtypes
```

Out[118]:

```
student name     object
science         float64
maths           float64
english         float64
Total           float64
college          object
course           object
dtype: object
```

In [119]:

```
final_df.dtypes != object
```

Out[119]:

```
student name    False
science          True
maths            True
english          True
Total            True
college         False
course          False
dtype: bool
```

In [120]:

```
final_df.columns[final_df.dtypes != object]  #For numerical data
```

Out[120]:

```
Index(['science', 'maths', 'english', 'Total'], dtype='object')
```

In [121]:

```
final_df.describe()
```

Out[121]:

	science	maths	english	Total
count	10.000000	10.000000	10.000000	9.000000
mean	88.000000	88.300000	82.555556	258.777778
std	4.320494	10.488618	7.959775	17.448336
min	76.000000	67.000000	65.000000	221.000000
25%	89.000000	87.000000	77.638889	253.000000
50%	89.500000	87.000000	85.500000	261.000000
75%	90.000000	97.750000	88.500000	272.000000
max	90.000000	99.000000	90.000000	278.000000

In [122]:

```
print(final_df.describe().columns)      #numerical
print(final_df.describe(include=object).columns)  #Categorical
```

```
Index(['science', 'maths', 'english', 'Total'], dtype='object')
Index(['student name', 'college', 'course'], dtype='object')
```

In [123]:

```
final_df.columns[final_df.dtypes == object]   #for catogorical data
```

Out[123]:

```
Index(['student name', 'college', 'course'], dtype='object')
```

In [124]:

```
cat_cols=final_df.describe(include=object).columns
num_cols=final_df.columns[final_df.dtypes!=object]
```

In [125]:

```
cat2=final_df.columns[final_df.dtypes==object]
print(cat2)
```

```
Index(['student name', 'college', 'course'], dtype='object')
```

In [126]:

```
print(cat_cols)
```

```
Index(['student name', 'college', 'course'], dtype='object')
```

In [127]:

```
print(num_cols)
```

```
Index(['science', 'maths', 'english', 'Total'], dtype='object')
```

In [128]:

```
print(final_df.college.mode())
```

```
0    VRSEC
dtype: object
```

Numerical columns

In [205]:
```
for col in num_cols:
    final_df[col].fillna(final_df[col].mean(),inplace=True)
```

In [206]:
```
final_df
```

Out[206]:

	student name	science	maths	english	Total	college	course
0	'meena'	90.0	98.0	90.000000	278.000000	VRSEC	ML
1	'sai'	89.0	97.0	89.000000	275.000000	LPU	AGBSC
2	'gayatri'	87.0	98.0	87.000000	272.000000	KBN	TESTING
3	'lokesh'	76.0	99.0	86.000000	261.000000	VRSEC	IOT
4	'swamy'	90.0	87.0	82.555556	258.777778	IIT	BIG DATA
5	'ramalakshmi'	90.0	87.0	89.000000	266.000000	VRSEC	IOT
6	'bhanu'	90.0	87.0	76.000000	253.000000	VRSEC	IOT
7	'hima'	90.0	87.0	76.000000	253.000000	VRSEC	IOT
8	'chandrika'	89.0	76.0	85.000000	250.000000	VRSEC	IOT
9	'keerthi'	89.0	67.0	65.000000	221.000000	VRSEC	IOT
10	'swathi'	88.0	88.3	82.555556	258.777778	VRSEC	SELENIUM
11	'sarayu'	88.0	88.3	82.555556	258.777778	MIC	IOT
12	'neelu'	88.0	88.3	82.555556	258.777778	VVIT	HADOOP

In [129]:
```
for col in cat_cols:
    final_df[col].fillna(final_df[col].mode()[0],inplace=True)
```

In [130]:
```
final_df['college'][final_df['college'].isna()]
```

Out[130]:
```
Series([], Name: college, dtype: object)
```

In [131]:
```
final_df['college'].mode()[0]
```

Out[131]:
```
'VRSEC'
```

In [132]:

```
final_df
```

Out[132]:

	student name	science	maths	english	Total	college	course
0	'meena'	90.0	98.0	90.000000	278.0	VRSEC	ML
1	'sai'	89.0	97.0	89.000000	275.0	LPU	AGBSC
2	'gayatri'	87.0	98.0	87.000000	272.0	KBN	TESTING
3	'lokesh'	76.0	99.0	86.000000	261.0	VRSEC	IOT
4	'swamy'	90.0	87.0	82.555556	NaN	IIT	BIG DATA
5	'ramalakshmi'	90.0	87.0	89.000000	266.0	VRSEC	IOT
6	'bhanu'	90.0	87.0	76.000000	253.0	VRSEC	IOT
7	'hima'	90.0	87.0	76.000000	253.0	VRSEC	IOT
8	'chandrika'	89.0	76.0	85.000000	250.0	VRSEC	IOT
9	'keerthi'	89.0	67.0	65.000000	221.0	VRSEC	IOT
10	'swathi'	NaN	NaN	NaN	NaN	VRSEC	SELENIUM
11	'sarayu'	NaN	NaN	NaN	NaN	MIC	IOT
12	'neelu'	NaN	NaN	NaN	NaN	VVIT	HADOOP

In [133]:

```
final_df[['college','maths']]
```

Out[133]:

	college	maths
0	VRSEC	98.0
1	LPU	97.0
2	KBN	98.0
3	VRSEC	99.0
4	IIT	87.0
5	VRSEC	87.0
6	VRSEC	87.0
7	VRSEC	87.0
8	VRSEC	76.0
9	VRSEC	67.0
10	VRSEC	NaN
11	MIC	NaN
12	VVIT	NaN

Grouping

In [134]:

```
final_df.groupby('college')['maths'].max()
```

Out[134]:

```
college
IIT       87.0
KBN       98.0
LPU       97.0
MIC        NaN
VRSEC     99.0
VVIT       NaN
Name: maths, dtype: float64
```

In [135]:

```
final_df.groupby('college')['maths','science'].sum()
```

Out[135]:

	maths	science
college		
IIT	87.0	90.0
KBN	98.0	87.0
LPU	97.0	89.0
MIC	0.0	0.0
VRSEC	601.0	614.0
VVIT	0.0	0.0

In [136]:

```
final_df.groupby('college').sum()
```

Out[136]:

	science	maths	english	Total
college				
IIT	90.0	87.0	82.555556	0.0
KBN	87.0	98.0	87.000000	272.0
LPU	89.0	97.0	89.000000	275.0
MIC	0.0	0.0	0.000000	0.0
VRSEC	614.0	601.0	567.000000	1782.0
VVIT	0.0	0.0	0.000000	0.0

In [137]:

```
final_df.groupby('course').maths.max()
```

Out[137]:

```
course
AGBSC       97.0
BIG DATA    87.0
HADOOP       NaN
IOT         99.0
ML          98.0
SELENIUM     NaN
TESTING     98.0
Name: maths, dtype: float64
```

In [138]:

```
final_df.groupby(['course','college']).maths.max()
```

Out[138]:

```
course    college
AGBSC     LPU        97.0
BIG DATA  IIT        87.0
HADOOP    VVIT        NaN
IOT       MIC         NaN
          VRSEC      99.0
ML        VRSEC      98.0
SELENIUM  VRSEC       NaN
TESTING   KBN        98.0
Name: maths, dtype: float64
```

Global sales data

- Observe five csv files in global sales data
- Read the market_fact.csv data

In [139]:

```
#import the required packages

df=pd.read_csv("market_fact.csv")
df
```

Out[139]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.91
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84
9	Ord_4725	Prod_6	SHP_6593	Cust_1641	57.2200	0.07	8	-27.72
10	Ord_4743	Prod_2	SHP_6615	Cust_1641	4072.0100	0.01	43	1675.98
11	Ord_1925	Prod_6	SHP_2637	Cust_708	465.9000	0.05	38	79.34
12	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.0500	0.04	27	23.12
13	Ord_2207	Prod_11	SHP_3093	Cust_839	3364.2480	0.10	15	-693.23
14	Ord_2207	Prod_10	SHP_3006	Cust_839	1410.9300	0.08	10	-317.48
15	Ord_2280	Prod_5	SHP_3114	Cust_839	460.6900	0.06	48	-103.48
16	Ord_2282	Prod_9	SHP_3122	Cust_839	443.4600	0.06	30	193.12
17	Ord_4471	Prod_15	SHP_6228	Cust_1521	13255.9300	0.02	25	4089.27
18	Ord_4427	Prod_6	SHP_6171	Cust_1521	283.1300	0.08	45	-141.26
19	Ord_996	Prod_13	SHP_1378	Cust_371	41.9700	0.05	12	-37.03
20	Ord_996	Prod_13	SHP_1378	Cust_371	57.1700	0.08	18	-24.03
21	Ord_996	Prod_6	SHP_1378	Cust_371	81.2500	0.01	11	-44.54
22	Ord_996	Prod_5	SHP_1377	Cust_371	3202.2500	0.09	44	991.26
23	Ord_996	Prod_7	SHP_1378	Cust_371	35.6400	0.05	10	-0.71
24	Ord_2573	Prod_3	SHP_3525	Cust_931	197.6100	0.08	13	3.46
25	Ord_2335	Prod_13	SHP_3204	Cust_931	38.2600	0.03	22	-2.34
26	Ord_2456	Prod_5	SHP_3367	Cust_931	109.5800	0.00	13	31.32
27	Ord_2405	Prod_9	SHP_3300	Cust_931	1062.6900	0.01	28	401.80
28	Ord_2573	Prod_4	SHP_3527	Cust_931	3594.7435	0.05	38	1016.97
29	Ord_2478	Prod_12	SHP_3395	Cust_931	139.9800	0.07	33	-140.54
...	...	...	...	...	...	...	...	...
8369	Ord_3633	Prod_3	SHP_5031	Cust_1274	1169.2600	0.02	41	515.62

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Sh
8370	Ord_2696	Prod_13	SHP_3690	Cust_1006	62.7800	0.04	20	-17.75	
8371	Ord_2624	Prod_4	SHP_3591	Cust_1006	4924.1350	0.07	28	1049.54	
8372	Ord_2772	Prod_9	SHP_3806	Cust_1006	56.9000	0.03	7	12.64	
8373	Ord_2600	Prod_16	SHP_3560	Cust_1006	106.6400	0.10	30	-31.95	
8374	Ord_2658	Prod_5	SHP_3637	Cust_1006	1082.6600	0.08	14	-256.93	
8375	Ord_2772	Prod_3	SHP_3806	Cust_1006	1413.8200	0.10	47	226.53	
8376	Ord_2624	Prod_8	SHP_3590	Cust_1006	1211.0000	0.00	36	-27.99	
8377	Ord_2722	Prod_12	SHP_3729	Cust_1006	34.0100	0.00	12	10.58	
8378	Ord_2706	Prod_2	SHP_3705	Cust_1006	1361.9100	0.05	20	312.52	
8379	Ord_2722	Prod_5	SHP_3730	Cust_1006	1008.9500	0.04	41	69.31	
8380	Ord_2772	Prod_6	SHP_3807	Cust_1006	308.9200	0.04	45	-143.58	
8381	Ord_2696	Prod_4	SHP_3691	Cust_1006	2836.0505	0.01	25	561.13	
8382	Ord_2658	Prod_3	SHP_3636	Cust_1006	120.9800	0.00	28	-92.85	
8383	Ord_2722	Prod_1	SHP_3731	Cust_1006	3508.3300	0.04	21	-546.98	
8384	Ord_4620	Prod_3	SHP_6435	Cust_1577	59.6200	0.04	10	-56.30	
8385	Ord_1833	Prod_3	SHP_2527	Cust_637	611.1600	0.04	46	100.22	
8386	Ord_2324	Prod_7	SHP_3189	Cust_851	121.8700	0.07	39	11.32	
8387	Ord_2220	Prod_3	SHP_3019	Cust_851	41.0600	0.04	4	-16.39	
8388	Ord_4424	Prod_1	SHP_6165	Cust_1519	994.0400	0.03	10	-335.06	
8389	Ord_4444	Prod_13	SHP_6192	Cust_1519	159.4100	0.00	44	34.68	
8390	Ord_5435	Prod_16	SHP_7594	Cust_1798	316.9900	0.04	47	-276.54	
8391	Ord_5435	Prod_4	SHP_7594	Cust_1798	1991.8985	0.07	20	88.36	
8392	Ord_5384	Prod_9	SHP_7519	Cust_1798	181.5000	0.08	43	-6.24	
8393	Ord_5348	Prod_8	SHP_7470	Cust_1798	356.7200	0.07	9	12.61	
8394	Ord_5353	Prod_4	SHP_7479	Cust_1798	2841.4395	0.08	28	374.63	
8395	Ord_5411	Prod_6	SHP_7555	Cust_1798	127.1600	0.10	20	-74.03	
8396	Ord_5388	Prod_6	SHP_7524	Cust_1798	243.0500	0.02	39	-70.85	
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.8700	0.03	23	565.34	
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.6900	0.00	47	131.39	

8399 rows × 10 columns

In [140]:

```
#check the shape

df.shape
```

Out[140]:

```
(8399, 10)
```

In [141]:

```
# Observe 5 rows randomly

df.sample(5)
```

Out[141]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shi
666	Ord_662	Prod_4	SHP_906	Cust_212	5067.5725	0.09	50	1275.91	
450	Ord_1415	Prod_1	SHP_1954	Cust_510	396.6900	0.09	12	-18.45	
1693	Ord_1941	Prod_1	SHP_2658	Cust_696	2651.2300	0.09	27	-741.81	
2482	Ord_2508	Prod_1	SHP_3437	Cust_974	580.0400	0.09	36	-31.86	
6161	Ord_3358	Prod_8	SHP_4655	Cust_1228	5144.9400	0.09	22	729.06	

In [142]:

```
#Observe last 2 rows
df.tail(2)
```

Out[142]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shippi
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.87	0.03	23	565.34	
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.69	0.00	47	131.39	

In [143]:

```
#Observe the tp 4 rows
df.head(4)
```

Out[143]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	

In [144]:

```
#check the column names
df.columns
```

Out[144]:

```
Index(['Ord_id', 'Prod_id', 'Ship_id', 'Cust_id', 'Sales', 'Discount',
       'Order_Quantity', 'Profit', 'Shipping_Cost', 'Product_Base_Margin'],
      dtype='object')
```

In [145]:

```
#check the row indices
df.index
```

Out[145]:

```
RangeIndex(start=0, stop=8399, step=1)
```

In [146]:

```


#check for missing values if there are any print the count column wise

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

```
Ord_id                 0
Prod_id                0
Ship_id                0
Cust_id                0
Sales                  0
Discount               0
Order_Quantity         0
Profit                 0
Shipping_Cost          0
Product_Base_Margin   63
dtype: int64
```

In [147]:

```


#total missing value count from all columns

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

```
63
```

In [148]:

```
#check the datatypes of columns
df.dtypes
```

Out[148]:

```
Ord_id                  object
Prod_id                 object
Ship_id                 object
Cust_id                 object
Sales                  float64
Discount               float64
Order_Quantity           int64
Profit                 float64
Shipping_Cost          float64
Product_Base_Margin    float64
dtype: object
```

In [149]:

```
#calculate the no of columns for each type

#df.info()
#df.describe()
#print(df.columns[df.dtypes==object])
#print(df.columns[df.dtypes=='int64'])
#print(df.columns[df.dtypes=='float64'])
df.dtypes.value_counts()
```

Out[149]:

```
float64    5
object     4
int64      1
dtype: int64
```

In [150]:

```
#what are the max and min values in each column

numcol=df.columns[df.dtypes!=object]

numcol
```

Out[150]:

```
Index(['Sales', 'Discount', 'Order_Quantity', 'Profit', 'Shipping_Cost',
       'Product_Base_Margin'],
      dtype='object')
```

In [151]:

```
df['Sales'].max()
```

Out[151]:

```
89061.05
```

In [152]:

```
for col in numcol:
    print(col,df[col].max(),df[col].min())
    #pd.DataFrame(df[col].max(),df[col].min(),columns=['maximum','minimum'])
```

```
Sales 89061.05 2.24
Discount 0.25 0.0
Order_Quantity 50 1
Profit 27220.69 -14140.7
Shipping_Cost 164.73 0.49
Product_Base_Margin 0.85 0.35
```

In [153]:

```
df.describe()
```

Out[153]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Ma
count	8399.000000	8399.000000	8399.000000	8399.000000	8399.000000	8336.000
mean	1775.878179	0.049671	25.571735	181.184424	12.838557	0.512
std	3585.050525	0.031823	14.481071	1196.653371	17.264052	0.135
min	2.240000	0.000000	1.000000	-14140.700000	0.490000	0.350
25%	143.195000	0.020000	13.000000	-83.315000	3.300000	0.380
50%	449.420000	0.050000	26.000000	-1.500000	6.070000	0.520
75%	1709.320000	0.080000	38.000000	162.750000	13.990000	0.590
max	89061.050000	0.250000	50.000000	27220.690000	164.730000	0.850

In [154]:

```
df.describe().loc[['max','min']]
df.describe().iloc[[3,7]]
```

Out[154]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
min	2.24	0.00	1.0	-14140.70	0.49	0.35
max	89061.05	0.25	50.0	27220.69	164.73	0.85

In [155]:

```
s =pd.DataFrame()
s['hell']=[1,2,3]
s
```

Out[155]:

	hell
0	1
1	2
2	3

In [156]:

```
# Calculate count,mean,std,max,calculate 25%,50%,75% quartiles
import numpy as np

#print(df.count())
#print(df.mean())
#print(df.std())
#print(df.max())
#print(df.quantile(0.25))
#print(df.quantile(0.5))
#print(df.quantile(0.75))
df.describe()
```

Out[156]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Ma
count	8399.000000	8399.000000	8399.000000	8399.000000	8399.000000	8336.000
mean	1775.878179	0.049671	25.571735	181.184424	12.838557	0.512
std	3585.050525	0.031823	14.481071	1196.653371	17.264052	0.135
min	2.240000	0.000000	1.000000	-14140.700000	0.490000	0.350
25%	143.195000	0.020000	13.000000	-83.315000	3.300000	0.380
50%	449.420000	0.050000	26.000000	-1.500000	6.070000	0.520
75%	1709.320000	0.080000	38.000000	162.750000	13.990000	0.590
max	89061.050000	0.250000	50.000000	27220.690000	164.730000	0.850

In [157]:

```
# Calculate the no of unique values in each categorical column

cat_cols=df.columns[df.dtypes==object]
cat_cols

df[cat_cols].nunique()
```

Out[157]:

```
Ord_id      5506
Prod_id       17
Ship_id     7701
Cust_id     1832
dtype: int64
```

In [158]:

```
df.describe(include=object)
```

Out[158]:

	Ord_id	Prod_id	Ship_id	Cust_id
count	8399	8399	8399	8399
unique	5506	17	7701	1832
top	Ord_2506	Prod_6	SHP_1378	Cust_1140
freq	6	1225	4	30

In [159]:

```
# Calculate category frequecies for each value in the categorical column

for i in cat_cols:
    print(df[i].value_counts())
```

```
Ord_2506    6
Ord_542     6
Ord_1581    5
Ord_2370    5
Ord_1846    5
Ord_56      5
Ord_845     5
Ord_2970    5
Ord_4946    5
Ord_1791    5
Ord_1639    5
Ord_1931    5
Ord_2894    5
Ord_1234    5
Ord_996     5
Ord_1980    5
Ord_4025    5
Ord_5186    5
Ord_1664    5
```

In [160]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8399 entries, 0 to 8398
Data columns (total 10 columns):
Ord_id                 8399 non-null object
Prod_id                8399 non-null object
Ship_id                8399 non-null object
Cust_id                8399 non-null object
Sales                  8399 non-null float64
Discount               8399 non-null float64
Order_Quantity         8399 non-null int64
Profit                 8399 non-null float64
Shipping_Cost          8399 non-null float64
Product_Base_Margin    8336 non-null float64
dtypes: float64(5), int64(1), object(4)
memory usage: 656.2+ KB
```

Read the remaing four csv files.Store them in cust_df,prod_df,ship_df,order_df

In [161]:

```
cust_df =pd.read_csv("cust_dimen.csv")
prod_df =pd.read_csv("prod_dimen.csv")
ship_df =pd.read_csv("shipping_dimen.csv")
order_df=pd.read_csv("orders_dimen.csv")
```

In [162]:

```
cust_df
```

Out[162]:

	Customer_Name	Province	Region	Customer_Segment	Cust_id
0	MUHAMMED MACINTYRE	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_1
1	BARRY FRENCH	NUNAVUT	NUNAVUT	CONSUMER	Cust_2
2	CLAY ROZENDAL	NUNAVUT	NUNAVUT	CORPORATE	Cust_3
3	CARLOS SOLTERO	NUNAVUT	NUNAVUT	CONSUMER	Cust_4
4	CARL JACKSON	NUNAVUT	NUNAVUT	CORPORATE	Cust_5
5	MONICA FEDERLE	NUNAVUT	NUNAVUT	CORPORATE	Cust_6
6	DOROTHY BADDERS	NUNAVUT	NUNAVUT	HOME OFFICE	Cust_7
7	NEOLA SCHNEIDER	NUNAVUT	NUNAVUT	HOME OFFICE	Cust_8
8	CARLOS DALY	NUNAVUT	NUNAVUT	HOME OFFICE	Cust_9
9	CLAUDIA MINER	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_10
10	ALLEN ROSENBLATT	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_11
11	SYLVIA FOULSTON	NUNAVUT	NUNAVUT	HOME OFFICE	Cust_12
12	JIM RADFORD	NUNAVUT	NUNAVUT	CORPORATE	Cust_13
13	CARL LUDWIG	NUNAVUT	NUNAVUT	CORPORATE	Cust_14
14	DON MILLER	NUNAVUT	NUNAVUT	HOME OFFICE	Cust_15
15	ANNIE CYPRUS	NUNAVUT	NUNAVUT	HOME OFFICE	Cust_16
16	GRANT CARROLL	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_17
17	ALAN BARNES	NUNAVUT	NUNAVUT	CORPORATE	Cust_18
18	JACK GARZA	NUNAVUT	NUNAVUT	CORPORATE	Cust_19
19	JULIA WEST	NUNAVUT	NUNAVUT	CORPORATE	Cust_20
20	EUGENE BARCHAS	NUNAVUT	NUNAVUT	CORPORATE	Cust_21
21	EDWARD HOOKS	NUNAVUT	NUNAVUT	CONSUMER	Cust_22
22	BRAD EASON	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_23
23	NICOLE HANSEN	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_24
24	DOROTHY WARDLE	NUNAVUT	NUNAVUT	CORPORATE	Cust_25
25	AARON BERGMAN	NUNAVUT	NUNAVUT	CORPORATE	Cust_26
26	DON JONES	NUNAVUT	NUNAVUT	CORPORATE	Cust_27
27	BETH THOMPSON	NUNAVUT	NUNAVUT	CORPORATE	Cust_28
28	FRANK PRICE	NUNAVUT	NUNAVUT	CORPORATE	Cust_29
29	MICHELLE LONSDALE	NUNAVUT	NUNAVUT	HOME OFFICE	Cust_30
...	...	...	...	...	...
1802	TONJA TURNELL	ALBERTA	WEST	CONSUMER	Cust_1803
1803	BRENDAN SWEED	ALBERTA	WEST	CONSUMER	Cust_1804
1804	TONY SAYRE	ALBERTA	WEST	HOME OFFICE	Cust_1805

	Customer_Name	Province	Region	Customer_Segment	Cust_id
1805	JIM KARLSSON	ALBERTA	WEST	SMALL BUSINESS	Cust_1806
1806	ROY PHAN	ALBERTA	WEST	CORPORATE	Cust_1807
1807	STEVEN ROELLE	ALBERTA	WEST	SMALL BUSINESS	Cust_1808
1808	CHRISTOPHER CONANT	ALBERTA	WEST	CONSUMER	Cust_1809
1809	ANDREW ROBERTS	ALBERTA	WEST	HOME OFFICE	Cust_1810
1810	CYMA KINNEY	ALBERTA	WEST	SMALL BUSINESS	Cust_1811
1811	CHRISTINE ABELMAN	ALBERTA	WEST	CORPORATE	Cust_1812
1812	ERICA SMITH	ALBERTA	WEST	SMALL BUSINESS	Cust_1813
1813	CHRISTOPHER CONANT	ALBERTA	WEST	HOME OFFICE	Cust_1814
1814	SARAH BROWN	ALBERTA	WEST	CONSUMER	Cust_1815
1815	SHUI TOM	ALBERTA	WEST	HOME OFFICE	Cust_1816
1816	FRANK HAWLEY	ALBERTA	WEST	HOME OFFICE	Cust_1817
1817	AARON BERGMAN	ALBERTA	WEST	CORPORATE	Cust_1818
1818	VICTORIA BRENNAN	ALBERTA	WEST	SMALL BUSINESS	Cust_1819
1819	ADRIAN SHAMI	ALBERTA	WEST	CONSUMER	Cust_1820
1820	PHILLINA OBER	ALBERTA	WEST	SMALL BUSINESS	Cust_1821
1821	ANDREW ROBERTS	ALBERTA	WEST	SMALL BUSINESS	Cust_1822
1822	JEREMY LONSDALE	ALBERTA	WEST	CORPORATE	Cust_1823
1823	SHUI TOM	ALBERTA	WEST	CONSUMER	Cust_1824
1824	ANDY YOTOV	ALBERTA	WEST	CORPORATE	Cust_1825
1825	NICOLE BRENNAN	ALBERTA	WEST	HOME OFFICE	Cust_1826
1826	JESSICA MYRICK	ALBERTA	WEST	SMALL BUSINESS	Cust_1827
1827	NICOLE BRENNAN	ALBERTA	WEST	CONSUMER	Cust_1828
1828	JASON FORTUNE	ALBERTA	WEST	CORPORATE	Cust_1829
1829	HARRY GREENE	ALBERTA	WEST	CORPORATE	Cust_1830
1830	GRANT DONATELLI	ALBERTA	WEST	CONSUMER	Cust_1831
1831	MICK BROWN	ALBERTA	WEST	CONSUMER	Cust_1832

1832 rows × 5 columns

In [163]:

```
df  #market_df
```

Out[163]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Sl
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87	
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64	
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32	
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.91	
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84	
9	Ord_4725	Prod_6	SHP_6593	Cust_1641	57.2200	0.07	8	-27.72	
10	Ord_4743	Prod_2	SHP_6615	Cust_1641	4072.0100	0.01	43	1675.98	
11	Ord_1925	Prod_6	SHP_2637	Cust_708	465.9000	0.05	38	79.34	
12	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.0500	0.04	27	23.12	
13	Ord_2207	Prod_11	SHP_3093	Cust_839	3364.2480	0.10	15	-693.23	
14	Ord_2207	Prod_10	SHP_3006	Cust_839	1410.9300	0.08	10	-317.48	
15	Ord_2280	Prod_5	SHP_3114	Cust_839	460.6900	0.06	48	-103.48	
16	Ord_2282	Prod_9	SHP_3122	Cust_839	443.4600	0.06	30	193.12	
17	Ord_4471	Prod_15	SHP_6228	Cust_1521	13255.9300	0.02	25	4089.27	
18	Ord_4427	Prod_6	SHP_6171	Cust_1521	283.1300	0.08	45	-141.26	
19	Ord_996	Prod_13	SHP_1378	Cust_371	41.9700	0.05	12	-37.03	
20	Ord_996	Prod_13	SHP_1378	Cust_371	57.1700	0.08	18	-24.03	
21	Ord_996	Prod_6	SHP_1378	Cust_371	81.2500	0.01	11	-44.54	
22	Ord_996	Prod_5	SHP_1377	Cust_371	3202.2500	0.09	44	991.26	
23	Ord_996	Prod_7	SHP_1378	Cust_371	35.6400	0.05	10	-0.71	
24	Ord_2573	Prod_3	SHP_3525	Cust_931	197.6100	0.08	13	3.46	
25	Ord_2335	Prod_13	SHP_3204	Cust_931	38.2600	0.03	22	-2.34	
26	Ord_2456	Prod_5	SHP_3367	Cust_931	109.5800	0.00	13	31.32	
27	Ord_2405	Prod_9	SHP_3300	Cust_931	1062.6900	0.01	28	401.80	
28	Ord_2573	Prod_4	SHP_3527	Cust_931	3594.7435	0.05	38	1016.97	
29	Ord_2478	Prod_12	SHP_3395	Cust_931	139.9800	0.07	33	-140.54	
...	...	...	...	...	...	...	...	...	
8369	Ord_3633	Prod_3	SHP_5031	Cust_1274	1169.2600	0.02	41	515.62	
8370	Ord_2696	Prod_13	SHP_3690	Cust_1006	62.7800	0.04	20	-17.75	
8371	Ord_2624	Prod_4	SHP_3591	Cust_1006	4924.1350	0.07	28	1049.54	

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Sh
8372	Ord_2772	Prod_9	SHP_3806	Cust_1006	56.9000	0.03	7	12.64	
8373	Ord_2600	Prod_16	SHP_3560	Cust_1006	106.6400	0.10	30	-31.95	
8374	Ord_2658	Prod_5	SHP_3637	Cust_1006	1082.6600	0.08	14	-256.93	
8375	Ord_2772	Prod_3	SHP_3806	Cust_1006	1413.8200	0.10	47	226.53	
8376	Ord_2624	Prod_8	SHP_3590	Cust_1006	1211.0000	0.00	36	-27.99	
8377	Ord_2722	Prod_12	SHP_3729	Cust_1006	34.0100	0.00	12	10.58	
8378	Ord_2706	Prod_2	SHP_3705	Cust_1006	1361.9100	0.05	20	312.52	
8379	Ord_2722	Prod_5	SHP_3730	Cust_1006	1008.9500	0.04	41	69.31	
8380	Ord_2772	Prod_6	SHP_3807	Cust_1006	308.9200	0.04	45	-143.58	
8381	Ord_2696	Prod_4	SHP_3691	Cust_1006	2836.0505	0.01	25	561.13	
8382	Ord_2658	Prod_3	SHP_3636	Cust_1006	120.9800	0.00	28	-92.85	
8383	Ord_2722	Prod_1	SHP_3731	Cust_1006	3508.3300	0.04	21	-546.98	
8384	Ord_4620	Prod_3	SHP_6435	Cust_1577	59.6200	0.04	10	-56.30	
8385	Ord_1833	Prod_3	SHP_2527	Cust_637	611.1600	0.04	46	100.22	
8386	Ord_2324	Prod_7	SHP_3189	Cust_851	121.8700	0.07	39	11.32	
8387	Ord_2220	Prod_3	SHP_3019	Cust_851	41.0600	0.04	4	-16.39	
8388	Ord_4424	Prod_1	SHP_6165	Cust_1519	994.0400	0.03	10	-335.06	
8389	Ord_4444	Prod_13	SHP_6192	Cust_1519	159.4100	0.00	44	34.68	
8390	Ord_5435	Prod_16	SHP_7594	Cust_1798	316.9900	0.04	47	-276.54	
8391	Ord_5435	Prod_4	SHP_7594	Cust_1798	1991.8985	0.07	20	88.36	
8392	Ord_5384	Prod_9	SHP_7519	Cust_1798	181.5000	0.08	43	-6.24	
8393	Ord_5348	Prod_8	SHP_7470	Cust_1798	356.7200	0.07	9	12.61	
8394	Ord_5353	Prod_4	SHP_7479	Cust_1798	2841.4395	0.08	28	374.63	
8395	Ord_5411	Prod_6	SHP_7555	Cust_1798	127.1600	0.10	20	-74.03	
8396	Ord_5388	Prod_6	SHP_7524	Cust_1798	243.0500	0.02	39	-70.85	
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.8700	0.03	23	565.34	
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.6900	0.00	47	131.39	

8399 rows × 10 columns

In [164]:

```
print(cust_df.shape)
print(df.Cust_id.nunique())
```

```
(1832, 5)
1832
```

In [165]:

```
print(prod_df.shape)
print(df.Prod_id.nunique())
```

```
(17, 3)
17
```

In [166]:

```
print(order_df.shape)
print(df.Ord_id.nunique())
```

```
(5506, 4)
5506
```

In [167]:
```
ship_df
```

Out[167]:
```
      Order_ID        Ship_Mode   Ship_Date    Ship_id
   0         3      REGULAR AIR  20-10-2010      SHP_1
   1       293  DELIVERY TRUCK  02-10-2012      SHP_2
   2       293      REGULAR AIR  03-10-2012      SHP_3
   3       483      REGULAR AIR  12-07-2011      SHP_4
   4       515      REGULAR AIR  30-08-2010      SHP_5
   5       613      REGULAR AIR  17-06-2011      SHP_6
   6       613      REGULAR AIR  18-06-2011      SHP_7
   7       643      EXPRESS AIR  25-03-2011      SHP_8
   8       678      REGULAR AIR  26-02-2010      SHP_9
   9       807      REGULAR AIR  24-11-2010     SHP_10
  10       868      REGULAR AIR  09-06-2012     SHP_11
  11       868      REGULAR AIR  10-06-2012     SHP_12
  12       933      REGULAR AIR  04-08-2012     SHP_13
  13       995      REGULAR AIR  31-05-2011     SHP_14
  14       998      REGULAR AIR  26-11-2009     SHP_15
  15      1154  DELIVERY TRUCK  16-02-2012     SHP_16
  16      1154      REGULAR AIR  16-02-2012     SHP_17
  17      1344      REGULAR AIR  22-04-2012     SHP_18
  18      1344      REGULAR AIR  19-04-2012     SHP_19
  19      1412      EXPRESS AIR  14-03-2010     SHP_20
  20      1412      REGULAR AIR  14-03-2010     SHP_21
  21      1539      REGULAR AIR  11-03-2011     SHP_22
  22      1539      REGULAR AIR  14-03-2011     SHP_23
  23      1540      REGULAR AIR  06-08-2012     SHP_24
  24      1702      REGULAR AIR  07-05-2011     SHP_25
  25      1761  DELIVERY TRUCK  25-12-2010     SHP_26
  26      1792      REGULAR AIR  13-11-2010     SHP_27
  27      2275      REGULAR AIR  22-10-2012     SHP_28
  28      2277      REGULAR AIR  02-01-2011     SHP_29
  29      2277      REGULAR AIR  03-01-2011     SHP_30
 ...       ...              ...         ...        ...
7671     57125      REGULAR AIR  05-06-2010   SHP_7672
7672     57152      REGULAR AIR  03-09-2012   SHP_7673
7673     57152      REGULAR AIR  10-09-2012   SHP_7674
```

	Order_ID	Ship_Mode	Ship_Date	Ship_id
7674	57216	DELIVERY TRUCK	29-07-2010	SHP_7675
7675	57216	REGULAR AIR	31-07-2010	SHP_7676
7676	57281	REGULAR AIR	20-04-2010	SHP_7677
7677	57281	DELIVERY TRUCK	22-04-2010	SHP_7678
7678	57827	DELIVERY TRUCK	04-07-2009	SHP_7679
7679	57827	REGULAR AIR	03-07-2009	SHP_7680
7680	58949	REGULAR AIR	17-12-2012	SHP_7681
7681	1222	REGULAR AIR	04-02-2010	SHP_7682
7682	5767	EXPRESS AIR	29-04-2012	SHP_7683
7683	5767	REGULAR AIR	30-04-2012	SHP_7684
7684	8961	DELIVERY TRUCK	03-07-2011	SHP_7685
7685	11712	REGULAR AIR	28-04-2009	SHP_7686
7686	14883	DELIVERY TRUCK	11-05-2011	SHP_7687
7687	14883	REGULAR AIR	15-05-2011	SHP_7688
7688	20193	REGULAR AIR	09-11-2010	SHP_7689
7689	36772	DELIVERY TRUCK	17-05-2010	SHP_7690
7690	39492	REGULAR AIR	20-04-2011	SHP_7691
7691	46212	EXPRESS AIR	14-09-2012	SHP_7692
7692	46437	REGULAR AIR	17-09-2009	SHP_7693
7693	47360	DELIVERY TRUCK	10-10-2010	SHP_7694
7694	52706	EXPRESS AIR	16-07-2012	SHP_7695
7695	54279	DELIVERY TRUCK	31-07-2011	SHP_7696
7696	55558	DELIVERY TRUCK	09-08-2010	SHP_7697
7697	55558	REGULAR AIR	11-08-2010	SHP_7698
7698	56550	EXPRESS AIR	10-04-2011	SHP_7699
7699	56550	REGULAR AIR	09-04-2011	SHP_7700
7700	56581	EXPRESS AIR	11-02-2009	SHP_7701

7701 rows × 4 columns

Diplay the top five rows of the five datasets and understand the data. Check for the co

In [168]:

```
# Merge the market_fact and cust_df files and store the result
# in df1

df1=pd.merge(df,cust_df,on='Cust_id',how='outer')
df1
```

Out[168]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32

In [169]:

```
print(df.shape)
print(df1.shape)  #4 -columns added after merging
```

```
(8399, 10)
(8399, 14)
```

In [170]:

```
print(df1.head())
```

```
     Ord_id  Prod_id  Ship_id    Cust_id    Sales  Discount  Order_Quantity  \
0  Ord_5446  Prod_16  SHP_7609  Cust_1818   136.81      0.01              23
1  Ord_5406  Prod_13  SHP_7549  Cust_1818    42.27      0.01              13
2  Ord_5446   Prod_4  SHP_7610  Cust_1818  4701.69      0.00              26
3  Ord_5456   Prod_6  SHP_7625  Cust_1818  2337.89      0.09              43
4  Ord_5485  Prod_17  SHP_7664  Cust_1818  4233.15      0.08              35

    Profit  Shipping_Cost  Product_Base_Margin  Customer_Name Province Region
\
0   -30.51           3.60                 0.56  AARON BERGMAN  ALBERTA   WEST
1     4.56           0.93                 0.54  AARON BERGMAN  ALBERTA   WEST
2  1148.90           2.50                 0.59  AARON BERGMAN  ALBERTA   WEST
3   729.34          14.30                 0.37  AARON BERGMAN  ALBERTA   WEST
4  1219.87          26.30                 0.38  AARON BERGMAN  ALBERTA   WEST

  Customer_Segment
0        CORPORATE
1        CORPORATE
2        CORPORATE
3        CORPORATE
4        CORPORATE
```

In [171]:

```
# Merge the df1 and prod_df files and store the result  in df2

df2=pd.merge(df1,prod_df,on='Prod_id',how='outer')
df2.shape
```

Out[171]:

```
(8399, 16)
```

In [172]:

```
# Merge the df2 and ship_df files and store the result  in df3

df3=pd.merge(df2,ship_df,on='Ship_id',how='outer')
df3
```

Out[172]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51
1	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.12
2	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.58
3	Ord_3730	Prod_16	SHP_5175	Cust_1314	459.08	0.04	34	61.57
4	Ord_4143	Prod_16	SHP_5771	Cust_1417	207.21	0.06	24	-78.64
5	Ord_4796	Prod_16	SHP_6686	Cust_1659	95.09	0.09	9	-13.53
6	Ord_4796	Prod_6	SHP_6686	Cust_1659	122.09	0.04	6	-15.20

In [173]:

```
# Merge the df3 and order_df files and store the result  in
# master_df

master_df=pd.merge(df3,order_df,on='Ord_id',how='outer')
master_df.shape
```

Out[173]:

```
(8399, 22)
```

In [174]:

```
master_df.Product_Category.value_counts()
```

Out[174]:

```
OFFICE SUPPLIES    4610
TECHNOLOGY         2065
FURNITURE          1724
Name: Product_Category, dtype: int64
```

In [175]:

```
master_df.groupby('Product_Category').Sales.max()
```

Out[175]:

```
Product_Category
FURNITURE          29345.27
OFFICE SUPPLIES    25409.63
TECHNOLOGY         89061.05
Name: Sales, dtype: float64
```

In [176]:

```
master_df.groupby('Product_Category').Sales.max().max()
```

Out[176]:

```
89061.05
```

In [177]:

```
master_df.groupby('Customer_Segment').Sales.max()
```

Out[177]:

```
Customer_Segment
CONSUMER          89061.05
CORPORATE         41343.21
HOME OFFICE       45923.76
SMALL BUSINESS    33367.85
Name: Sales, dtype: float64
```

In [178]:

```
master_df.groupby(['Customer_Segment','Product_Category']).Sales.max()
```

Out[178]:

```
Customer_Segment  Product_Category
CONSUMER          FURNITURE           28389.14
                  OFFICE SUPPLIES     23516.31
                  TECHNOLOGY          89061.05
CORPORATE         FURNITURE           29345.27
                  OFFICE SUPPLIES     23106.46
                  TECHNOLOGY          41343.21
HOME OFFICE       FURNITURE           28180.08
                  OFFICE SUPPLIES     18697.24
                  TECHNOLOGY          45923.76
SMALL BUSINESS    FURNITURE           24639.80
                  OFFICE SUPPLIES     25409.63
                  TECHNOLOGY          33367.85
Name: Sales, dtype: float64
```

In [179]:

```
master_df[master_df.Product_Category=='FURNITURE'].Sales.max()
```

Out[179]:

```
29345.27
```

In [182]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [183]:

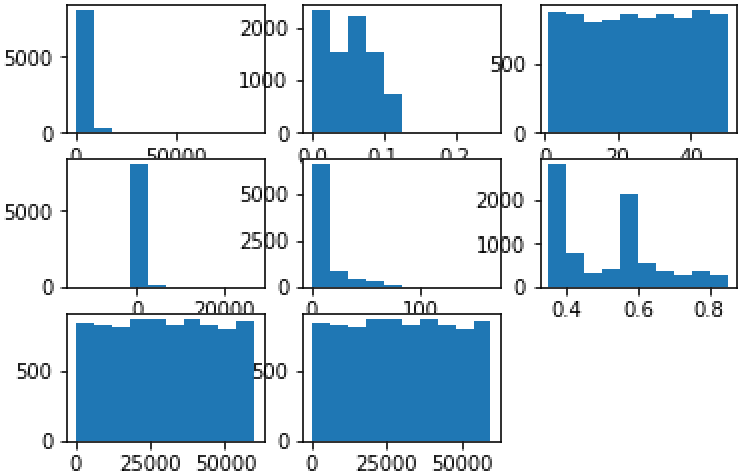
```
num_cols=master_df.columns[master_df.dtypes!=object]
num_cols
```

Out[183]:

```
Index(['Sales', 'Discount', 'Order_Quantity', 'Profit', 'Shipping_Cost',
       'Product_Base_Margin', 'Order_ID_x', 'Order_ID_y'],
      dtype='object')
```

In [206]:

```
for i in range(len(num_cols)):
    plt.subplot(3,3,i+1)
    plt.hist(master_df[num_cols[i]])
```



In [186]:

```
import matplotlib
matplotlib.__version__
```

Out[186]:

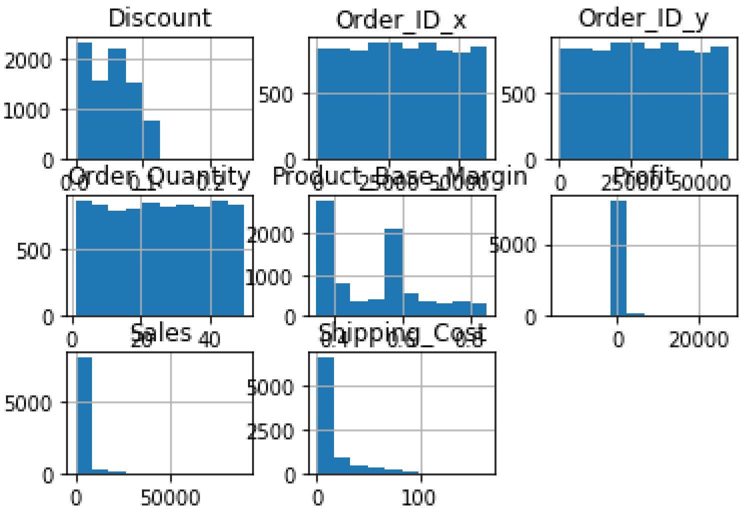
```
'3.0.3'
```

In [187]:

```
master_df.hist()   #pandas hist()
```

Out[187]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001A0612AC780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001A0619202E8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001A061944860>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001A06196DDD8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001A06199B390>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001A0619C0908>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001A0619EBE80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001A061A1A470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001A061A1A4A8>]],
      dtype=object)
```



In [190]:
```
cat_cols=master_df.columns[master_df.dtypes=='object']
cat_cols
```

Out[190]:
```
Index(['Ord_id', 'Prod_id', 'Ship_id', 'Cust_id', 'Customer_Name', 'Province',
       'Region', 'Customer_Segment', 'Product_Category',
       'Product_Sub_Category', 'Ship_Mode', 'Ship_Date', 'Order_Date',
       'Order_Priority'],
      dtype='object')
```

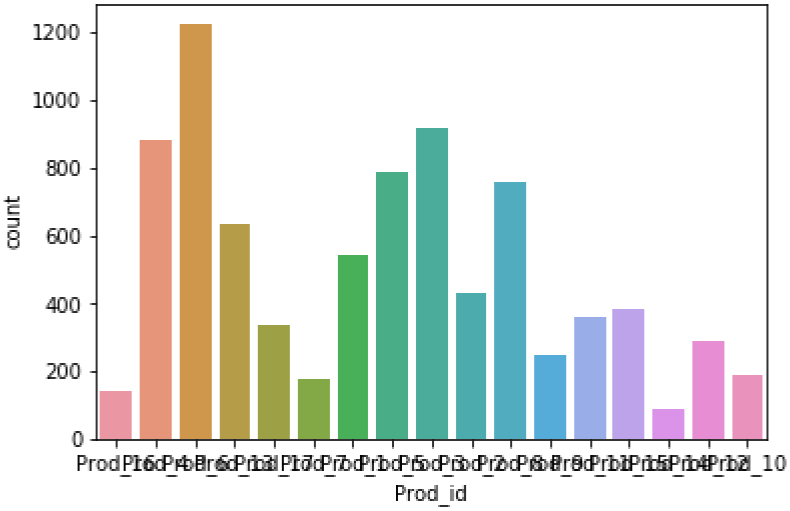
In [ ]:
```
for i in range(len(cat_cols)):
    plt.subplot(5,3,i+1)
    plt.hist(master_df[cat_cols[i]])
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-210-7a923d54f30c> in <module>
      1 for i in range(len(cat_cols)):
      2     plt.subplot(5,3,i+1)
----> 3     plt.hist(master_df[cat_cols[i]])

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in hist(x, bins, range, dens
ity, weights, cumulative, bottom, histtype, align, orientation, rwidth, log, co
lor, label, stacked, normed, data, **kwargs)
   2657         align=align, orientation=orientation, rwidth=rwidth, log=log,
   2658         color=color, label=label, stacked=stacked, normed=normed,
-> 2659         **({"data": data} if data is not None else {}), **kwargs)
   2660
   2661

~\Anaconda3\lib\site-packages\matplotlib\__init__.py in inner(ax, data, *args,
 **kwargs)
   1808                         "the Matplotlib list!)" % (label_namer, func.__
name__),
   1809                         RuntimeWarning, stacklevel=2)
-> 1810             return func(ax, *args, **kwargs)
   1811
   1812         inner.__doc__ = _add_data_doc(inner.__doc__,

~\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py in hist(self, x, bins, r
ange, density, weights, cumulative, bottom, histtype, align, orientation, rwidt
h, log, color, label, stacked, normed, **kwargs)
   6665                 patch = _barfunc(bins[:-1]+boffset, height, width,
   6666                                  align='center', log=log,
-> 6667                                  color=c, **{bottom_kwarg: bottom})
   6668                 patches.append(patch)
   6669                 if stacked:

~\Anaconda3\lib\site-packages\matplotlib\__init__.py in inner(ax, data, *args,
 **kwargs)
   1808                         "the Matplotlib list!)" % (label_namer, func.__
name__),
   1809                         RuntimeWarning, stacklevel=2)
-> 1810             return func(ax, *args, **kwargs)
   1811
   1812         inner.__doc__ = _add_data_doc(inner.__doc__,

~\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py in bar(self, x, height,
 width, bottom, align, **kwargs)
   2339             ymin = max(ymin * 0.9, 1e-100)
   2340             self.dataLim.intervaly = (ymin, ymax)
-> 2341         self.autoscale_view()
   2342
   2343         bar_container = BarContainer(patches, errorbar, label=label)

~\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in autoscale_view(self,
 tight, scalex, scaley)
```

```
   2427                x_stickies = sum([sticky.x for sticky in stickies], [])
   2428                y_stickies = sum([sticky.y for sticky in stickies], [])
-> 2429                if self.get_xscale().lower() == 'log':
   2430                    x_stickies = [xs for xs in x_stickies if xs > 0]
   2431                if self.get_yscale().lower() == 'log':

KeyboardInterrupt:
```

In [207]:
```
sns.countplot(master_df['Prod_id'])
```

Out[207]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a0630fbe48>
```



Boxplot (Univariant, numeric)

In [193]:
```
plt.boxplot([1,10,20,200,50])
```

...

In [194]:
```
q1=np.quantile([1,10,20,200,50],0.25)  #median of first half
q2=np.quantile([1,10,20,200,50],0.5)  #median
q3=np.quantile([1,10,20,200,50],0.75)  #second half median
```

In [195]:
```
q1,q2,q3
```

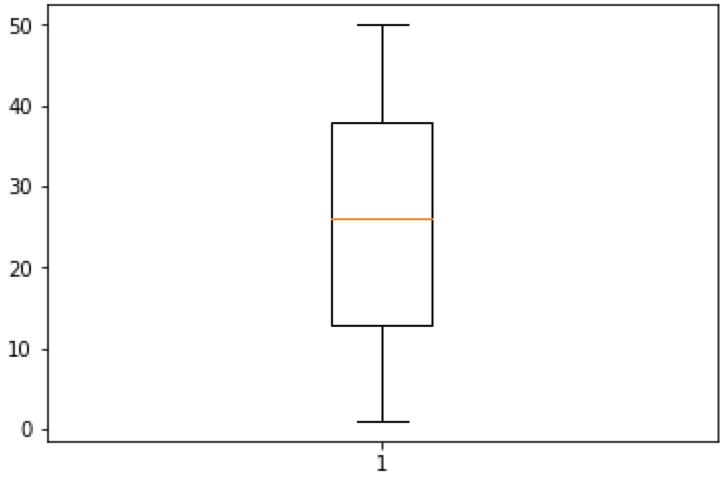
Out[195]:
```
(10.0, 20.0, 50.0)
```

In [199]:
```
IQR=1.5*(q3-q1)  #Inter Quartile range
print(q1-IQR,q3+IQR)
#values which are outside of this range are outliers
```

```
-50.0 110.0
```

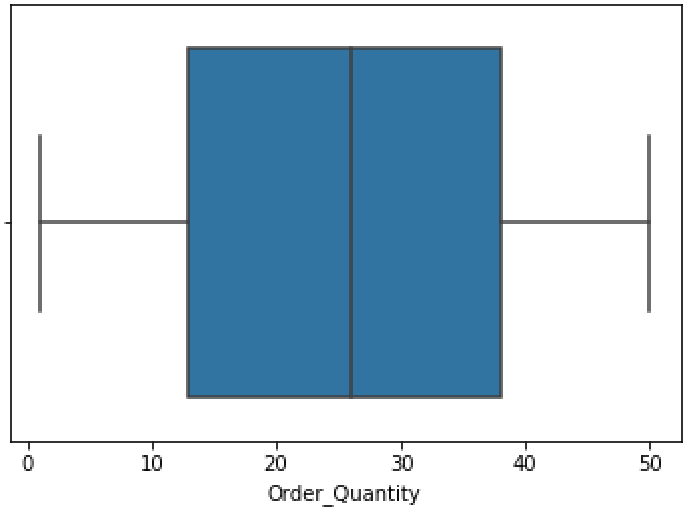
In [200]:
```
plt.boxplot(master_df['Order_Quantity'])
```

Out[200]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a0615ac358>,
  <matplotlib.lines.Line2D at 0x1a061625fd0>],
 'caps': [<matplotlib.lines.Line2D at 0x1a061625550>,
  <matplotlib.lines.Line2D at 0x1a0616252e8>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a06163ad30>],
 'medians': [<matplotlib.lines.Line2D at 0x1a061625860>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a061625898>],
 'means': []}
```



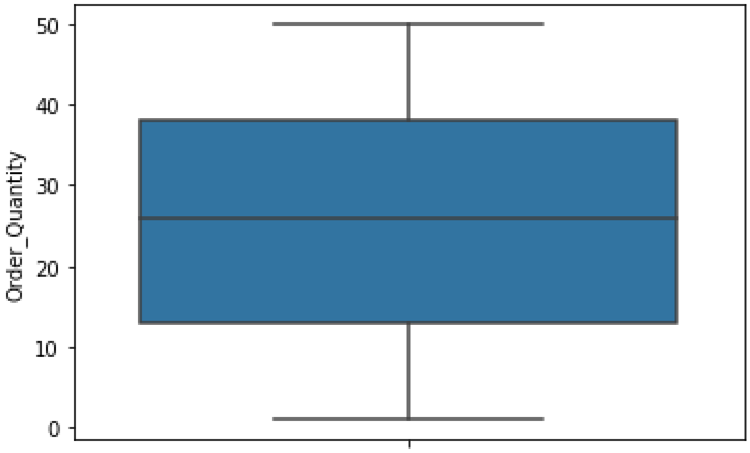
In [201]:
```
sns.boxplot(master_df['Order_Quantity'])
```

Out[201]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a061896400>
```



```
In [202]: sns.boxplot(y=master_df['Order_Quantity'])
```

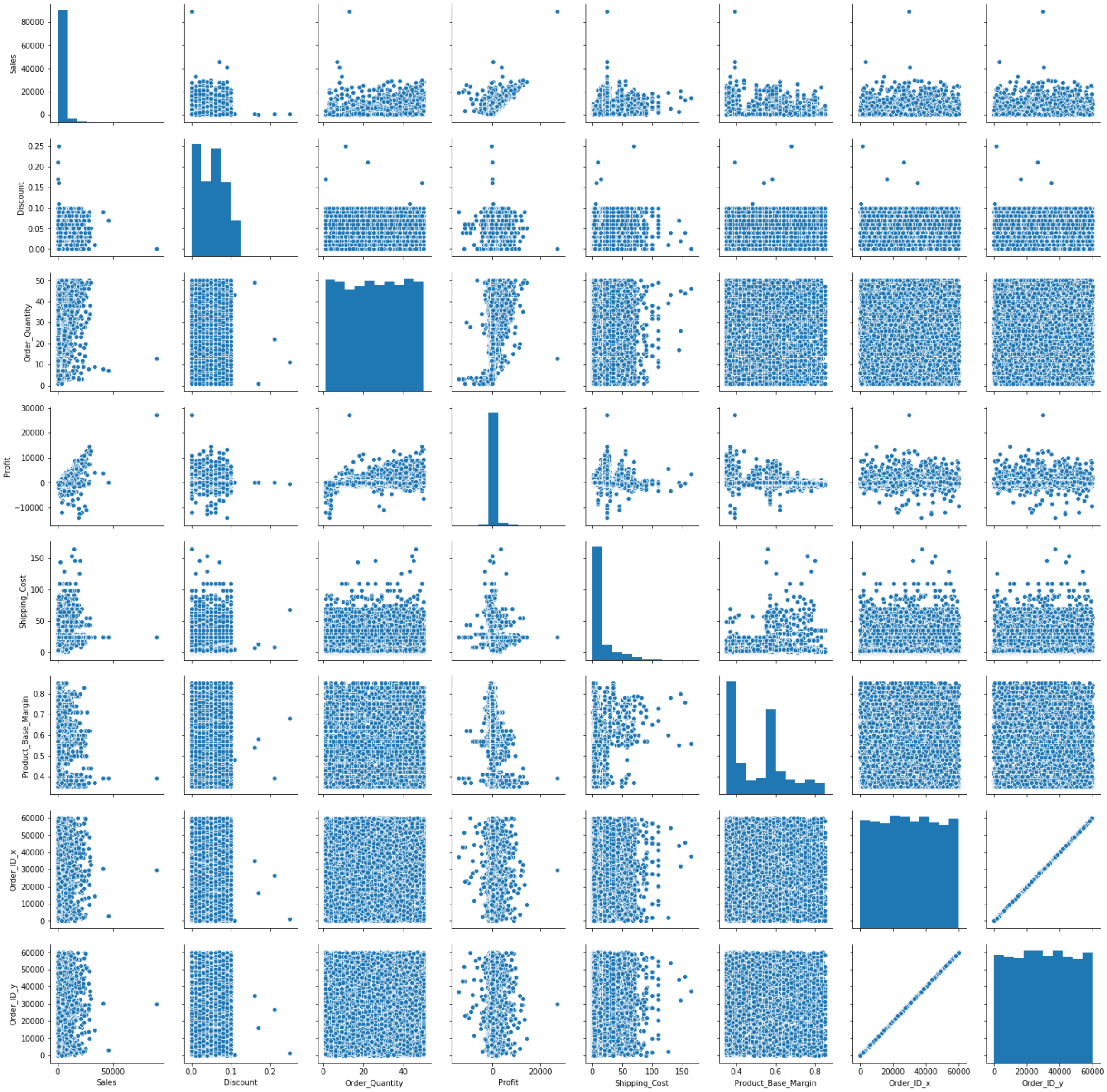
```
Out[202]: <matplotlib.axes._subplots.AxesSubplot at 0x1a061861ac8>
```



PairPlot

In [208]:
```
sns.pairplot(master_df)
```

Out[208]:
```
<seaborn.axisgrid.PairGrid at 0x1a06179f6d8>
```



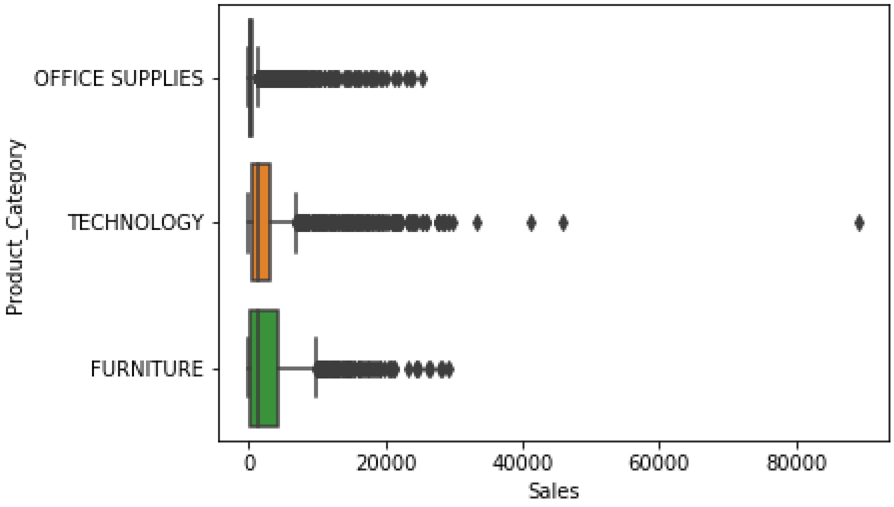
```
In [209]: master_df.corr()
```

Out[209]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Bas
Sales	1.000000	-0.019686	0.220582	0.581960	0.434578	
Discount	-0.019686	1.000000	-0.009649	-0.037128	-0.001956	
Order_Quantity	0.220582	-0.009649	1.000000	0.194655	-0.011457	
Profit	0.581960	-0.037128	0.194655	1.000000	-0.021362	-
Shipping_Cost	0.434578	-0.001956	-0.011457	-0.021362	1.000000	
Product_Base_Margin	0.156759	0.004079	0.007839	-0.112985	0.373826	
Order_ID_x	-0.007792	-0.003213	0.010953	-0.006820	-0.004582	-
Order_ID_y	-0.007792	-0.003213	0.010953	-0.006820	-0.004582	-

```
In [215]: sns.boxplot(y=master_df['Product_Category'],          #categorical category
                      x=master_df['Sales'])                     #numeric category
```

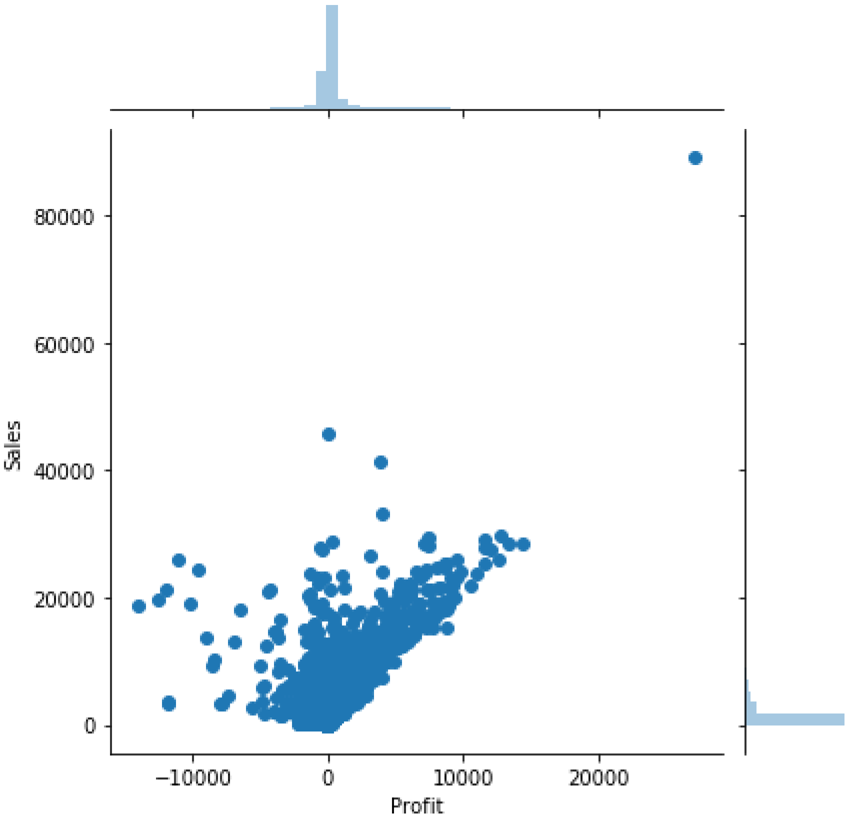
```
Out[215]: <matplotlib.axes._subplots.AxesSubplot at 0x1a0774ad7b8>
```



JointPlot (scatterplot with histogram)

```
In [216]: sns.jointplot(master_df['Profit'],master_df['Sales'])
```

```
Out[216]: <seaborn.axisgrid.JointGrid at 0x1a077430320>
```



```
In [ ]:
```