import data from a csv file and perform following task

* group them
* find minimum ,maximum and average value of any column

In [1]:
```
import pandas as pd
df=pd.read_csv("Citydata.csv")
```

In [ ]:
```
df
```

In [ ]:
```
ndf=df.groupby("city")
#it is dataframe group by object
#object has keys and values
#in this example 4 cities are there so 4 keys and corresponding values will be
created with fitem,price ,month
#this is how dataframe group by object internally
```

In [ ]:
```
ndf.get_group('delhi')
```

In [ ]:
```
ndf.get_group('noida')
```

In [ ]:
```
#ndf is a dataframe group by object
#loop will print sub dataframe
for df1 in ndf:
    print(df1)
```

In [ ]:
```
#i want to print "city" that is key and corresponding values
for city,df1 in ndf:
    print(city)
    print(df1)
```

In [ ]:
```
ndf.get_group('gurugram')
#it is similar to sql statement
#select* from data GROUP BY city
```

In [ ]:
```
#to find maximum and minimum price in each city
ndf.min()
#in groupby method , it has splitted the data
#to achieve min from each group it find min value from each group and combine
```

In [ ]:
```
ndf.max()
```

In [ ]:
```
df
```

In [ ]:
```
cdf=ndf.max()
```

In [ ]:
```
cdf
```

In [ ]:
```
type(cdf)
```

In [ ]:
```
cdf.index
```

In [ ]:
```
ndf.mean()
#mean or average
```

In [ ]:
```
ndf.describe()
#stats
```

In [ ]:
```
ndf
```

In [ ]:
```
ndf.median()
```

In [ ]:
```
#second method to get group data, but k is series
k=df['city']=='delhi'
df.loc[k]
#Pandas DataFrame.loc attribute access a group of rows and columns by label(s)
#or a boolean array in the given DataFrame
```

In [ ]:
```
type(k)
```

Concatenation

In [ ]:
```
import pandas as pd

df = pd.DataFrame({'Name':['Sam', 'Andrea', 'Alex', 'Robin', 'Kia'],
                   'Age':[14, 25, 55, 8, 21],
                  'Weight':[45, 88, 56, 15, 71]})
```

In [ ]:
```
df
```

In [ ]:
```
df1 = pd.DataFrame({"price":[12, 4, 5, 100, 1],
                    "item":['a','b','c','d','e']})
```

In [ ]:
```
df1
```

In [ ]:
```
ndf = pd.concat([df,df1])
#index added one after another
```

In [ ]:
```
ndf
```

In [ ]:
```
#concatenate one after another
ndf1 = pd.concat([df,df1],ignore_index=True)
#now index from 0 to n
```

In [ ]:
```
ndf1
```

In [ ]:
```
#you can associate key with each dataframe
ndf2 = pd.concat([df,ndf1], keys=["basic", "purchase"])
```

In [ ]:
```
ndf2
```

In [ ]:
```
type(ndf2)
```

In [ ]:
```
#search will start by key
ndf2.loc["basic"]
```

In [ ]:
```
temp = pd.DataFrame({
    "city": ["mumbai","delhi","banglore"],
    "temperature": [32,45,30],
}, index=[0,1,2])
```

In [ ]:
```
polu= pd.DataFrame({
    "city": ["delhi","banglore"],
    "polu": [337,132],
}, index=[1,2])
polu
```

In [ ]:
```
ndf = pd.concat([temp,polu],axis=1)
```

In [ ]:
```
ndf
```

In [ ]:
```
#if you want to add any series in concatenated datafram
s1=pd.Series(["sunny","Rain",'rain'],name="day")
df2 = pd.concat([temp,s1],axis=1)
df2
```

In [ ]:
```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                   'D': ['D0', 'D1', 'D2', 'D3']}, index=[0, 1, 2, 3])
```

In [ ]:
```
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7']},
                   index=[4, 5, 6, 7])
```

In [ ]:
```
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                    'B': ['B8', 'B9', 'B10', 'B11'],
                    'C': ['C8', 'C9', 'C10', 'C11'],
                    'D': ['D8', 'D9', 'D10', 'D11']},
                   index=[8, 9, 10, 11])
```

In [ ]:
```
output = pd.concat([df1,df2,df3])
output
```

In [ ]:
```
result = pd.concat([df1,df2,df3], keys=['x', 'y', 'z'])
result
```

In [ ]:
```
result.loc['x']
#resulting object index value
```

In [ ]:
```
#pd.concat(objs, axis=0, join='outer', ignore_index=False, keys=None,
  #        levels=None, names=None, verify_integrity=False, copy=True)
```

In [ ]:
```
result = pd.concat([df1, df2], axis=1, join='inner')
result
```

In [ ]:
```
result = pd.concat([df1, df2], axis=0, join='inner')
result
```

In [ ]:
```
#A useful shortcut to concat() are the append() instance methods on Series and
DataFrame
#In the case of DataFrame, the indexes must be disjoint but the columns do not
need to b
result = df1.append(df2)
result
```

In [ ]:
```
result = df1.append(df3, sort=False)
result
```

In [ ]:
```
result = df1.append(df3, sort=True)
result
```

In [ ]:
```
result = pd.concat([df1, df2], axis=1, join='outer')
result
```

Merge

In [ ]:
```
#pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None,
 #        left_index=False, right_index=False, sort=True,
  #       suffixes=('_x', '_y'), copy=True, indicator=False,
   #      validate=None)
```

The return type will be the same as left. If left is a DataFrame or named Series and right is a subclass of DataFrame, the return type will still be DataFrame.

merge is a function in the pandas namespace, and it is also available as a DataFrame instance method merge(), with the calling DataFrame being implicitly considered the left object in the join

Merge method SQL Join Name Description

left LEFT OUTER JOIN Use keys from left frame only

right RIGHT OUTER JOIN Use keys from right frame only

outer FULL OUTER JOIN Use union of keys from both frames

inner INNER JOIN Use intersection of keys from both frames

In [ ]:
```
import pandas as pd
mf1 = pd.DataFrame({
    "city": ["delhi","chicago","orlando"],
    "polu": [421,114,35],
})
mf1
```

In [ ]:
```
mf2 = pd.DataFrame({
    "city": ["delhi","chicago","mumbai"],
    "temp": [421,114,35],
})
mf2
```

In [ ]:
```
mf3 = pd.merge(mf1, mf2, on="city")
mf3
```

In [ ]:
```
#intersaction on key values
mf3 = pd.merge(mf1, mf2, on="city",how="inner")
mf3
```

In [ ]:
```
#outer join: all data and common values used once only
mf3 = pd.merge(mf1, mf2, on="city",how="outer")
mf3
```

In [ ]:
```
#left part and common
mf3 = pd.merge(mf1, mf2, on="city",how="left")
mf3
```

In [ ]:
```
#right part and common
mf3 = pd.merge(mf1, mf2, on="city",how="right")
mf3
```

In [ ]:
```
mf3 = pd.merge(mf1, mf2, on="city",how="right",indicator=True)
mf3
```

In [ ]:
```
mf1 = pd.DataFrame({
    "city": ["delhi","chicago","orlando"],
    "polu": [421,114,35],
    "temp" :[34,43,23]
})
mf2 = pd.DataFrame({
    "city": ["delhi","mumbai","noida"],
    "polu": [42,114,35],
    "temp" :[22,33,11]
})
```

In [ ]:
```
mf3 = pd.merge(mf1, mf2, on="city")
```

In [ ]:
```
mf3
```

In [ ]:
```
mf3 = pd.merge(mf1, mf2, on=["city","temp"])
mf3
```

In [ ]:
```
print(mf1)
print(mf2)
```

In [ ]:
```

```