```
import pandas as pd
```

```
df=pd.read_csv('sample_data/california_housing_train.csv')
```

```
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0
2	-114.56	33.69	17.0	720.0	174.0	333.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0

```
df.shape
```

```
(17000, 9)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           17000 non-null  float64
 1   latitude            17000 non-null  float64
 2   housing_median_age  17000 non-null  float64
 3   total_rooms         17000 non-null  float64
 4   total_bedrooms      17000 non-null  float64
 5   population          17000 non-null  float64
 6   households          17000 non-null  float64
 7   median_income       17000 non-null  float64
 8   median_house_value  17000 non-null  float64
dtypes: float64(9)
memory usage: 1.2 MB
```

```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000
mean	-119.562108	35.625225	28.589353	2643.664412	539.410824
std	2.005166	2.137340	12.586937	2179.947071	421.499452

```
df[['total_rooms','total_bedrooms']]
```

	total_rooms	total_bedrooms
0	5612.0	1283.0
1	7650.0	1901.0
2	720.0	174.0
3	1501.0	337.0
4	1454.0	326.0
...	...	...
16995	2217.0	394.0
16996	2349.0	528.0
16997	2677.0	531.0
16998	2672.0	552.0
16999	1820.0	300.0

17000 rows × 2 columns

```
dataframe = {
    'name':['shubham','b','c','c','d'],
     'surname':['q','w','e','r','r'],
    'phone':[1,2,3,4,4]
}
```

```
df1=pd.DataFrame(dataframe)
```

```
df1
```

	name	surname	phone
0	shubham	q	1
1	b	w	2
2	c	e	3
3	c	r	4
4	d	r	4

```
df1['name'].unique()
```

```
array(['shubham', 'b', 'c', 'd'], dtype=object)
```

```
df1['name'].nunique()
```

```
4
```

```
df1['name'].value_counts()
```

```
c          2
shubham    1
b          1
d          1
Name: name, dtype: int64
```

```
df.head(2)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0

```
for i in df.columns:
  if i.startswith('total'):
    print(i)
```

```
total_rooms
total_bedrooms
```

```
df['city']=df['population'].apply(lambda x:'Pune' if x > 0 and x<500 else 'Hydra' if x >=
```

```
df['city'].value_counts()
```

```
Mum      10311
Hydra     5088
Pune      1601
Name: city, dtype: int64
```

```
df['Country']=df['median_income'].apply(lambda x:'China' if x > 0 and x<1.5 else 'USA' if
```

```
df['Country'].value_counts()
```

```
IND      10954
USA       5388
China      658
Name: Country, dtype: int64
```

```
df.groupby('Country').agg({'total_rooms':['mean','sum'],'median_income':['min','median','m
```

	Country	total_rooms		median_income		
		mean	sum	min	median	max
0	China	1468.363222	966183.0	0.4999	1.27305	1.4952
1	IND	2941.993884	32226601.0	3.0000	4.32690	15.0001
2	USA	2180.681329	11749511.0	1.5000	2.34190	2.9982

```
df2=df.groupby('Country')['population'].sum().reset_index()
```

```
df2['rank']=df2['population'].rank(method='dense',ascending=False)
```

```
df2
```

	Country	population	rank
0	China	728874.0	3.0
1	IND	15865061.0	1.0
2	USA	7708822.0	2.0

```
comp= {
    'empid': [1,2,3,4,5,6],
    'name':['a','b','c','d','e','f'],
    'dept':['aa','bb','cc','dd','cc','dd'],
    'salary':[1000,3200,800,1400,500,600]
}
emp_df=pd.DataFrame(comp)
```

```
emp_df
```

	empid	name	dept	salary
0	1	a	aa	1000
1	2	b	bb	3200
2	3	c	cc	800
3	4	d	dd	1400
4	5	e	cc	500
5	6	f	dd	600

```
emp_df['rank_sal']=emp_df['salary'].rank(method='dense',ascending=False)
```

```
emp_df.sort_values(by=['rank_sal'])
```

	empid	name	dept	salary	rank_sal
1	2	b	bb	3200	1.0
3	4	d	dd	1400	2.0
0	1	a	aa	1000	3.0
2	3	c	cc	800	4.0
5	6	f	dd	600	5.0
4	5	e	cc	500	6.0

```
emp_df.replace('dd','shubham')
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_gro
0	1	a	aa	1000	3.0	1.0	1.0	
1	2	b	bb	3200	1.0	1.0	1.0	
2	3	c	cc	800	4.0	1.0	1.0	
3	4	d	shubham	1400	2.0	1.0	1.0	
4	5	e	cc	500	6.0	2.0	2.0	
5	6	f	shubham	600	5.0	2.0	2.0	

```
emp_df[emp_df['rank_sal']==4.0]['empid'].loc[2]
```

```
3
```

```
emp_df['rank_dept']=emp_df.groupby('dept')['salary'].rank(method='dense',ascending=False)
```

```
emp_df.sort_values(by=['dept','rank_dept'],ascending=[True,False])
```

	empid	name	dept	salary	rank_sal	rank_dept
0	1	a	aa	1000	3.0	1.0
1	2	b	bb	3200	1.0	1.0
4	5	e	cc	500	6.0	2.0
2	3	c	cc	800	4.0	1.0
5	6	f	dd	600	5.0	2.0
3	4	d	dd	1400	2.0	1.0

```
emp_df['rank_avg']=emp_df.groupby(['dept'])['salary'].rank(method='average',ascending=Fals
```

```
emp_df
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg
0	1	a	aa	1000	3.0	1.0	1.0
1	2	b	bb	3200	1.0	1.0	1.0
2	3	c	cc	800	4.0	1.0	1.0
3	4	d	dd	1400	2.0	1.0	1.0
4	5	e	cc	500	6.0	2.0	2.0
5	6	f	dd	600	5.0	2.0	2.0

```
emp_df['rank_avg_without_groupby']=emp_df['salary'].rank(method='average',ascending=False)
```

```
emp_df
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_groupby
0	1	a	aa	1000	3.0	1.0	1.0	3.(
1	2	b	bb	3200	1.0	1.0	1.0	1.(
2	3	c	cc	800	4.0	1.0	1.0	4.(
3	4	d	dd	1400	2.0	1.0	1.0	2.(
4	5	e	cc	500	6.0	2.0	2.0	6.(
5	6	f	dd	600	5.0	2.0	2.0	5.(

```
emp_df['empid_lag']=emp_df['empid'].shift(1)
```

```
emp_df
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_groupby
0	1	a	aa	1000	3.0	1.0	1.0	3.(
1	2	b	bb	3200	1.0	1.0	1.0	1.(
2	3	c	cc	800	4.0	1.0	1.0	4.(
3	4	d	dd	1400	2.0	1.0	1.0	2.(
4	5	e	cc	500	6.0	2.0	2.0	6.(
5	6	f	dd	600	5.0	2.0	2.0	5.(

```
emp_df['empid_lead']=emp_df['empid'].shift(-1)
```

```
emp_df
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_groupby
0	1	a	aa	1000	3.0	1.0	1.0	3.(
1	2	b	bb	3200	1.0	1.0	1.0	1.(
2	3	c	cc	800	4.0	1.0	1.0	4.(
3	4	d	dd	1400	2.0	1.0	1.0	2.(
4	5	e	cc	500	6.0	2.0	2.0	6.(

```
emp_df['rank_first']=emp_df.groupby(['dept'])['salary'].rank(method='first',ascending=Fals
```

```
emp_df
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_groupby
0	1	a	aa	1000	3.0	1.0	1.0	3.(
1	2	b	bb	3200	1.0	1.0	1.0	1.(
2	3	c	cc	800	4.0	1.0	1.0	4.(
3	4	d	dd	1400	2.0	1.0	1.0	2.(
4	5	e	cc	500	6.0	2.0	2.0	6.(
5	6	f	dd	600	5.0	2.0	2.0	5.(

```
person={
    'empid':[1,2,3,4,5],
    'email':['a@g.com','b@g.com','c@g.com','b@g.com','a@g.com']
}
```

```
person_df=pd.DataFrame(person)
```

```
person_df
```

	empid	email
0	1	a@g.com
1	2	b@g.com
2	3	c@g.com
3	4	b@g.com
4	5	a@g.com

```
person_df.drop_duplicates(subset=['email'],keep="last")
```

	empid	email
2	3	c@g.com
3	4	b@g.com

```
emp_df
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_groupby
0	1	a	aa	1000	3.0	1.0	1.0	3.(
1	2	b	bb	3200	1.0	1.0	1.0	1.(
2	3	c	cc	800	4.0	1.0	1.0	4.(
3	4	d	dd	1400	2.0	1.0	1.0	2.(
4	5	e	cc	500	6.0	2.0	2.0	6.(
5	6	f	dd	600	5.0	2.0	2.0	5.(

```
emp_df['cum_sum']=emp_df['salary'].cumsum()
```

```
emp_df
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_groupby
0	1	a	aa	1000	3.0	1.0	1.0	3.(
1	2	b	bb	3200	1.0	1.0	1.0	1.(
2	3	c	cc	800	4.0	1.0	1.0	4.(
3	4	d	dd	1400	2.0	1.0	1.0	2.(
4	5	e	cc	500	6.0	2.0	2.0	6.(
5	6	f	dd	600	5.0	2.0	2.0	5.(

```
emp_df['dept_wise_cumsum']=emp_df.sort_values(by=['salary']).groupby('dept')['salary'].cum
```

```
emp_df.sort_values(by=['dept','salary'])
```

	empid	name	dept	salary	rank_sal	rank_dept	rank_avg	rank_avg_without_groupby

```
emp_df.apply(min)
```

```
empid                         1
name                          a
dept                         aa
salary                      500
rank_sal                    1.0
rank_dept                   1.0
rank_avg                    1.0
rank_avg_without_groupby    1.0
empid_lag                   NaN
empid_lead                  2.0
rank_first                  1.0
cum_sum                    1000
dept_wise_cumsum            500
dtype: object
```

```
#apply function it will appply on particular column
person_df.apply(lambda x : len(x))
```

```
empid    5
email    5
dtype: int64
```

```
app = pd.DataFrame(
    [
        (1, 521, True, 10.1, 'Hello'),
        (2, 723, False, 54.2, 'Hey'),
        (3, 123, False, 33.2, 'Howdy'),
        (4, 641, True, 48.6, 'Hi'),
        (5, 467, False, 98.1, 'Hey'),
    ],
    columns=['colA', 'colB', 'colC', 'colD', 'colE']
)
```

```
app_df=pd.DataFrame(app)
```

```
app_df
```

	colA	colB	colC	colD	colE
0	1	521	True	10.1	Hello
1	2	723	False	54.2	Hey
2	3	123	False	33.2	Howdy
3	4	641	True	48.6	Hi
4	5	467	False	98.1	Hey

```
import numpy as np
```

```
app_df[['colA','colB']]=app_df[['colA','colB']].apply(np.sqrt)
```

```
app_df
```

	colA	colB	colC	colD	colE
0	1.000000	22.825424	True	10.1	Hello
1	1.414214	26.888659	False	54.2	Hey
2	1.732051	11.090537	False	33.2	Howdy
3	2.000000	25.317978	True	48.6	Hi
4	2.236068	21.610183	False	98.1	Hey

```
#map() method operates over one element at a time and missing values will be denoted as Na
```

```
app_df['colE']=app_df['colE'].map({'Hello':'Good bye','Hi':'Good Morning'})
```

```
app_df.fillna('Hello',inplace=True)
```

```
app_df
```

	colA	colB	colC	colD	colE
0	1.000000	22.825424	True	10.1	Good bye
1	1.414214	26.888659	False	54.2	Hello
2	1.732051	11.090537	False	33.2	Hello
3	2.000000	25.317978	True	48.6	Good Morning
4	2.236068	21.610183	False	98.1	Hello

```
# Use Pandas Series.apply() Function to Single Column
app_df["colB"]=app_df["colB"].apply(lambda x: x/10)
```

```
app_df
```

	colA	colB	colC	colD	colE
0	1.000000	2.282542	True	10.1	Good bye
1	1.414214	2.688866	False	54.2	Hello
2	1.732051	1.109054	False	33.2	Hello
3	2.000000	2.531798	True	48.6	Good Morning
4	2.236068	2.161018	False	98.1	Hello

```
# Use DataFrame.apply() method
app_df2 = app_df.apply(lambda x: len(x))
```

```
app_df2
```

```
colA    5
colB    5
colC    5
colD    5
colE    5
dtype: int64
```

```
# Use Pandas DataFrame.applymap() method
app_df3 = app_df.applymap(lambda a: a*10)
```

```
app_df3
```

	colA	colB	colC	colD	colE
0	10.000000	22.825424	10	101.0	Good byeGood byeGood byeGood byeGood byeGood b...
1	14.142136	26.888659	0	542.0	HelloHelloHelloHelloHelloHelloHelloHelloHelloH...
2	17.320508	11.090537	0	332.0	HelloHelloHelloHelloHelloHelloHelloHelloHelloH...
3	20.000000	25.317978	10	486.0	Good MorningGood MorningGood MorningGood Morni...
4	22.360680	21.610183	0	981.0	HelloHelloHelloHelloHelloHelloHelloHelloHelloH...

```
# Use DataFrame.applymap() method
app_df4 = app_df.applymap(lambda a: str(a)+".00")
```

```
app_df4
```

	colA	colB	colC	colD	colE
0	1.0.00	2.2825424421026654.00	True.00	10.1.00	Good bye.00
1	1.4142135623730951.00	2.68886593194975.00	False.00	54.2.00	Hello.00
2	1.7320508075688772.00	1.1090536506409419.00	False.00	33.2.00	Hello.00
3	2.0.00	2.5317977802344327.00	True.00	48.6.00	Good Morning.00
4	2.23606797749979.00	2.1610182784974308.00	False.00	98.1.00	Hello.00

```
a=[]
b=[]

for i in range(4):
  a.append(i)
  #print(a)
  b.append(a)
```

```
print(b)
#result --> [[0],[0,1],[0,1,2],[0,1,2,3],[0,1,2,3,4]]
```

```
[[0, 1, 2, 3], [0, 1, 2, 3], [0, 1, 2, 3], [0, 1, 2, 3]]
```

```
tup=(2,3,4,[5,6])
type(tup)
```

```
tuple
```

```
tup
```

```
(2, 3, 4, [5, 6])
```