In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
bankdata=pd.read_csv("/Users/shashankreddy/Desktop/Datafiles/bank-additional
```

In [3]:
```
bankdata.head()
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	mon
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	m
1	39	services	single	high.school	no	no	no	telephone	m
2	25	services	married	high.school	no	yes	no	telephone	j
3	38	services	married	basic.9y	no	unknown	unknown	telephone	j
4	47	admin.	married	university.degree	no	yes	no	cellular	n

5 rows × 21 columns

In [4]:
```
bankdata.dtypes
```

Out[4]:
```
age                int64
job               object
marital           object
education         object
default           object
housing           object
loan              object
contact           object
month             object
day_of_week       object
duration           int64
campaign           int64
pdays              int64
previous           int64
poutcome          object
emp.var.rate     float64
cons.price.idx   float64
cons.conf.idx    float64
euribor3m        float64
nr.employed      float64
y                 object
dtype: object
```

In [5]:
```
bankdata.size
```

Out[5]:
```
86499
```

In [6]:
```
bankdata.shape
```

Out[6]:
```
(4119, 21)
```

In [7]:
```
bankdata.columns
```

Out[7]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

In [8]:
```
bankdata.dtypes
```

Out[8]:
```
age                int64
job               object
marital           object
education         object
default           object
housing           object
loan              object
contact           object
month             object
day_of_week       object
duration           int64
campaign           int64
pdays              int64
previous           int64
poutcome          object
emp.var.rate     float64
cons.price.idx   float64
cons.conf.idx    float64
euribor3m        float64
nr.employed      float64
y                 object
dtype: object
```

In [ ]:
```
```

In [9]:
```
bankdata.select_dtypes(include="object")
```

Out[9]:

	job	marital	education	default	housing	loan	contact	m
0	blue-collar	married	basic.9y	no	yes	no	cellular	
1	services	single	high.school	no	no	no	telephone	
2	services	married	high.school	no	yes	no	telephone	
3	services	married	basic.9y	no	unknown	unknown	telephone	
4	admin.	married	university.degree	no	yes	no	cellular	
...	...	...	...	...	...	...	...	
4114	admin.	married	basic.6y	no	yes	yes	cellular	
4115	admin.	married	high.school	no	yes	no	telephone	
4116	student	single	high.school	no	no	no	cellular	
4117	admin.	married	high.school	no	no	no	cellular	
4118	management	single	high.school	no	yes	no	cellular	

4119 rows × 11 columns

In [10]:
```
bankdata.select_dtypes(include="object").columns
```

Out[10]:
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contac
t',
       'month', 'day_of_week', 'poutcome', 'y'],
      dtype='object')
```

In [11]:
```
bankdata.select_dtypes(exclude="object")
```

Out[11]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.cc
0	30	487	2	999	0	-1.8	92.893	
1	39	346	4	999	0	1.1	93.994	
2	25	227	1	999	0	1.4	94.465	
3	38	17	3	999	0	1.4	94.465	
4	47	58	1	999	0	-0.1	93.200	
...	...	...	...	...	...	...	...	
4114	30	53	1	999	0	1.4	93.918	
4115	39	219	1	999	0	1.4	93.918	
4116	27	64	2	999	1	-1.8	92.893	
4117	58	528	1	999	0	1.4	93.444	
4118	34	175	1	999	0	-0.1	93.200	

4119 rows × 10 columns

In [12]:
```
bankdata.select_dtypes(exclude="object").columns
```

Out[12]:
```
Index(['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate',
       'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'],
      dtype='object')
```

In [13]:
```
bankdata.select_dtypes(exclude="object").columns.size
```

Out[13]:
```
10
```

In [14]:
```
bankdata.isnull()
```

Out[14]:

	age	job	marital	education	default	housing	loan	contact	month	day_o
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
4114	False	False	False	False	False	False	False	False	False	
4115	False	False	False	False	False	False	False	False	False	
4116	False	False	False	False	False	False	False	False	False	
4117	False	False	False	False	False	False	False	False	False	
4118	False	False	False	False	False	False	False	False	False	

4119 rows × 21 columns

In [15]:
```
bankdata.isnull().sum()
```

Out[15]:
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

In [16]:
```
bankdata.isna()
```

Out[16]:

	age	job	marital	education	default	housing	loan	contact	month	day_o
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
4114	False	False	False	False	False	False	False	False	False	
4115	False	False	False	False	False	False	False	False	False	
4116	False	False	False	False	False	False	False	False	False	
4117	False	False	False	False	False	False	False	False	False	
4118	False	False	False	False	False	False	False	False	False	

4119 rows × 21 columns

In [17]:
```
bankdata.isna().sum()
```

Out[17]:
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

drop Duplicates

In [18]:
```
bankdata.drop_duplicates()
```

Out[18]:

	age	job	marital	education	default	housing	loan	conta
0	30	blue-collar	married	basic.9y	no	yes	no	cellul
1	39	services	single	high.school	no	no	no	telephor
2	25	services	married	high.school	no	yes	no	telephor
3	38	services	married	basic.9y	no	unknown	unknown	telephor
4	47	admin.	married	university.degree	no	yes	no	cellul
...	...	...	...	...	...	...	...	
4114	30	admin.	married	basic.6y	no	yes	yes	cellul
4115	39	admin.	married	high.school	no	yes	no	telephor
4116	27	student	single	high.school	no	no	no	cellul
4117	58	admin.	married	high.school	no	no	no	cellul
4118	34	management	single	high.school	no	yes	no	cellul

4119 rows × 21 columns

In [19]:
```
bankdata.drop_duplicates().all()
```

Out[19]:
```
age               True
job               True
marital           True
education         True
default           True
housing           True
loan              True
contact           True
month             True
day_of_week       True
duration         False
campaign          True
pdays            False
previous         False
poutcome          True
emp.var.rate      True
cons.price.idx    True
cons.conf.idx     True
euribor3m         True
nr.employed       True
y                 True
dtype: bool
```

INfo

In [20]:
```
#Provide info about bankdata
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             4119 non-null   int64  
 1   job             4119 non-null   object 
 2   marital         4119 non-null   object 
 3   education       4119 non-null   object 
 4   default         4119 non-null   object 
 5   housing         4119 non-null   object 
 6   loan            4119 non-null   object 
 7   contact         4119 non-null   object 
 8   month           4119 non-null   object 
 9   day_of_week     4119 non-null   object 
 10  duration        4119 non-null   int64  
 11  campaign        4119 non-null   int64  
 12  pdays           4119 non-null   int64  
 13  previous        4119 non-null   int64  
 14  poutcome        4119 non-null   object 
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   object 
dtypes: float64(5), int64(5), object(11)
memory usage: 675.9+ KB
```

take

Take is used to extract the data with repsect to index.

it will get entire data.

it has one parameter axis

default axis=0

axis=0 means rows

In [21]:
```
bankdata.take([1,2])
```

Out[21]:

	age	job	marital	education	default	housing	loan	contact	month	day_o
1	39	services	single	high.school	no	no	no	telephone	may	
2	25	services	married	high.school	no	yes	no	telephone	jun	

2 rows × 21 columns

In [22]:
```
bankdata.take([2,3,4],axis=1)
```

Out[22]:

	marital	education	default
0	married	basic.9y	no
1	single	high.school	no
2	married	high.school	no
3	married	basic.9y	no
4	married	university.degree	no
...	...	...	...
4114	married	basic.6y	no
4115	married	high.school	no
4116	single	high.school	no
4117	married	high.school	no
4118	single	high.school	no

4119 rows × 3 columns

ILOC property

visadata.iolc[[rows],[columns]]

visadata.iloc[[start:end],[start:end]]

In [23]:
```
bankdata.iloc[[0]]
```

Out[23]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_we
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	

1 rows × 21 columns

In [24]:
```
bankdata.iloc[[],[0]]
```

Out[24]:

	age

In [25]:
```
bankdata.iloc[[0],[0]]
```

Out[25]:

	age
0	30

In [26]:
```
bankdata.iloc[:,:]
```

Out[26]:

	age	job	marital	education	default	housing	loan	conta
0	30	blue-collar	married	basic.9y	no	yes	no	cellul
1	39	services	single	high.school	no	no	no	telephor
2	25	services	married	high.school	no	yes	no	telephor
3	38	services	married	basic.9y	no	unknown	unknown	telephor
4	47	admin.	married	university.degree	no	yes	no	cellul
...	...	...	...	...	...	...	...	
4114	30	admin.	married	basic.6y	no	yes	yes	cellul
4115	39	admin.	married	high.school	no	yes	no	telephor
4116	27	student	single	high.school	no	no	no	cellul
4117	58	admin.	married	high.school	no	no	no	cellul
4118	34	management	single	high.school	no	yes	no	cellul

4119 rows × 21 columns

In [27]:
```
bankdata.iloc[[1,2,3],[1,2,3]]
```

Out[27]:

	job	marital	education
1	services	single	high.school
2	services	married	high.school
3	services	married	basic.9y

In [28]:
```
bankdata.loc[6]
```

Out[28]:
```
age                            32
job                         admin.
marital                     single
education         university.degree
default                         no
housing                        yes
loan                            no
contact                   cellular
month                          sep
day_of_week                    mon
duration                       290
campaign                         4
pdays                          999
previous                         0
poutcome                nonexistent
emp.var.rate                  -1.1
cons.price.idx               94.199
cons.conf.idx                 -37.5
euribor3m                     0.879
nr.employed                  4963.6
y                               no
Name: 6, dtype: object
```

loc method

it will directly take column name instaed of number in iloc

In [29]:
```
bankdata.loc[:,"age"]
```

Out[29]:
```
0       30
1       39
2       25
3       38
4       47
        ..
4114    30
4115    39
4116    27
4117    58
4118    34
Name: age, Length: 4119, dtype: int64
```

In [30]:
```
bankdata.columns
```

Out[30]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

In [31]:
```
bankdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             4119 non-null   int64  
 1   job             4119 non-null   object 
 2   marital         4119 non-null   object 
 3   education       4119 non-null   object 
 4   default         4119 non-null   object 
 5   housing         4119 non-null   object 
 6   loan            4119 non-null   object 
 7   contact         4119 non-null   object 
 8   month           4119 non-null   object 
 9   day_of_week     4119 non-null   object 
 10  duration        4119 non-null   int64  
 11  campaign        4119 non-null   int64  
 12  pdays           4119 non-null   int64  
 13  previous        4119 non-null   int64  
 14  poutcome        4119 non-null   object 
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  y               4119 non-null   object 
dtypes: float64(5), int64(5), object(11)
memory usage: 675.9+ KB
```

In [32]:
```
bankdata["age"].unique()
```

Out[32]:
```
array([30, 39, 25, 38, 47, 32, 41, 31, 35, 36, 29, 27, 44, 46, 45, 50, 55,
       40, 28, 34, 33, 51, 48, 20, 76, 56, 24, 58, 60, 37, 52, 42, 49, 54,
       59, 57, 43, 53, 75, 82, 71, 21, 22, 23, 26, 81, 61, 67, 73, 18, 64,
       74, 77, 86, 85, 63, 88, 78, 72, 68, 80, 66, 19, 62, 65, 69, 70])
```

In [33]:
```
bankdata["job"].unique()
```

Out[33]:
```
array(['blue-collar', 'services', 'admin.', 'entrepreneur',
       'self-employed', 'technician', 'management', 'student', 'retired',
       'housemaid', 'unemployed', 'unknown'], dtype=object)
```

Culmulative data frequency

categorical data Analysis

value Counts

It will give unique items and how many times they are repeating

In [34]:
```
cdf =bankdata["job"].value_counts()
```

In [35]:
```
cdf
```

Out[35]:
```
job
admin.           1012
blue-collar       884
technician        691
services          393
management        324
retired           166
self-employed     159
entrepreneur      148
unemployed        111
housemaid         110
student            82
unknown            39
Name: count, dtype: int64
```

In [36]:
```
type(cdf)
```

Out[36]:
```
pandas.core.series.Series
```

In [37]:
```
cdf.keys()
```

Out[37]:
```
Index(['admin.', 'blue-collar', 'technician', 'services', 'management',
       'retired', 'self-employed', 'entrepreneur', 'unemployed', 'housemai
d',
       'student', 'unknown'],
      dtype='object', name='job')
```

In [38]:
```
cdf.values
```

Out[38]:
```
array([1012,  884,  691,  393,  324,  166,  159,  148,  111,  110,   82,
         39])
```

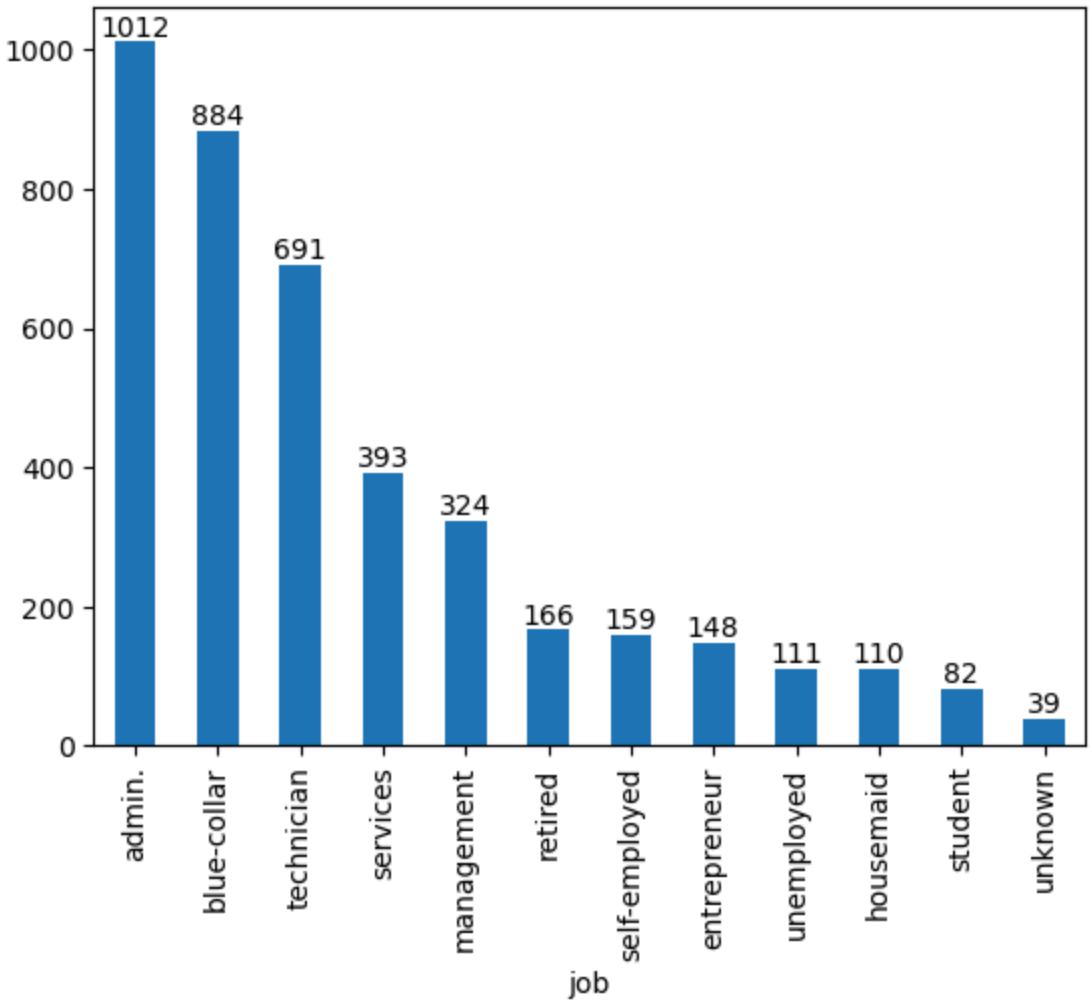
In [39]:
```
cdf.index
```

Out[39]:
```
Index(['admin.', 'blue-collar', 'technician', 'services', 'management',
       'retired', 'self-employed', 'entrepreneur', 'unemployed', 'housemai
d',
       'student', 'unknown'],
      dtype='object', name='job')
```

In [40]:
```
cdf.items()
```

Out[40]:
```
<zip at 0x134825380>
```

In [41]:
```
cd=bankdata['job'].value_counts()
ax=cd.plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.show()
```



In [42]:
```
bankdata.dtypes
```

Out[42]:
```
age                int64
job               object
marital           object
education         object
default           object
housing           object
loan              object
contact           object
month             object
day_of_week       object
duration           int64
campaign           int64
pdays              int64
previous           int64
poutcome          object
emp.var.rate     float64
cons.price.idx   float64
cons.conf.idx    float64
euribor3m        float64
nr.employed      float64
y                 object
```

In [43]:
```
bankdata.value_counts()
```

Out[43]:
```
age  job          marital   education            default  housing  loan
contact    month  day_of_week  duration  campaign  pdays  previous  poutcom
e     emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed
y
18   student      single    unknown              no       no       no
cellular   sep    thu          385       1         3      1         success
-3.4          92.379          -29.8          0.809      5017.5       yes
1
43   unemployed   married   university.degree    unknown  no       no
telephone  jun    mon          114       1         999    0         nonexis
tent   1.4          94.465          -41.8          4.865      5228.1
no     1
44   blue-collar  married   basic.4y             no       no       no
telephone  jun    thu          153       1         999    0         nonexis
tent   1.4          94.465          -41.8          4.866      5228.1
no     1
                                                 unknown  unknown
telephone  jun    thu          471       3         999    0         nonexis
tent   1.4          94.465          -41.8          4.866      5228.1
no     1
                                                 unknown  no       no
cellular   may    tue          659       1         999    1         failure
-1.8          92.893          -46.2          1.344      5099.1       no
1

..
34   blue-collar  married   basic.9y             unknown  yes      no
telephone  jun    thu          266       4         999    0         nonexis
tent   1.4          94.465          -41.8          4.961      5228.1
no     1
                             high.school          no       no       no
cellular   may    thu          144       3         999    1         failure
-1.8          92.893          -46.2          1.266      5099.1       no
1
                                                 yes      no
cellular   may    wed          133       1         999    1         failure
-1.8          92.893          -46.2          1.281      5099.1       no
1
                             professional.course  no       yes      no
cellular   apr    fri          377       1         999    0         nonexis
tent  -1.8          93.075          -47.1          1.405      5099.1
no     1
88   retired      divorced  basic.4y             no       yes      yes
cellular   mar    wed          82        2         999    0         nonexis
tent  -1.8          92.843          -50.0          1.663      5099.1
no     1
Name: count, Length: 4119, dtype: int64
```

Numerical Data Analysis

Out[44]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.cc
0	30	487	2	999	0	-1.8	92.893	
1	39	346	4	999	0	1.1	93.994	
2	25	227	1	999	0	1.4	94.465	
3	38	17	3	999	0	1.4	94.465	
4	47	58	1	999	0	-0.1	93.200	
...	...	...	...	...	...	...	...	
4114	30	53	1	999	0	1.4	93.918	
4115	39	219	1	999	0	1.4	93.918	
4116	27	64	2	999	1	-1.8	92.893	
4117	58	528	1	999	0	1.4	93.444	
4118	34	175	1	999	0	-0.1	93.200	

4119 rows × 10 columns

In [45]:
```
bankdata.select_dtypes(exclude="object").columns
```

Out[45]:
```
Index(['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate',
       'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'],
      dtype='object')
```

In [46]:
```
bankdata["age"].unique()
```

Out[46]:
```
array([30, 39, 25, 38, 47, 32, 41, 31, 35, 36, 29, 27, 44, 46, 45, 50, 55,
       40, 28, 34, 33, 51, 48, 20, 76, 56, 24, 58, 60, 37, 52, 42, 49, 54,
       59, 57, 43, 53, 75, 82, 71, 21, 22, 23, 26, 81, 61, 67, 73, 18, 64,
       74, 77, 86, 85, 63, 88, 78, 72, 68, 80, 66, 19, 62, 65, 69, 70])
```

work on pandas like dictionaries

len

min

max

mean

standarddevation

In [47]:
```
length=len(bankdata["age"])
```

In [48]:
```
minvalue=bankdata["age"].min()
```

```
18
```

In [50]:
```
min(bankdata["age"])
```

Out[50]: 18

In [51]:
```
maxvalue=bankdata["age"].max()
```

In [52]:
```
print(maxvalue)
```

```
88
```

In [53]:
```
max(bankdata["age"])
```

Out[53]: 88

In [54]:
```
mean=bankdata["age"].mean()
```

In [55]:
```
print(mean)
```

```
40.11361981063365
```

In [56]:
```
std=bankdata["age"].std()
```

In [57]:
```
std
```

Out[57]: 10.313361547199822

percentile

percentile and quartile are in Numpy module

np.percentile() takes two arguments

a,q

a=values or data

q=percentile it takes values from 0 to 100

if we want 50 percentile we need to give q=50

for quantile:

np.quantile() takes two arguments

a,q

a=values or data

q=percentile it takes values from 0 to 1

if we want 50 percentile we need to give q=0.5

In [58]:
```
p50=round(np.percentile(a=bankdata["age"],q=50),2)
print(p50)
```

```
38.0
```

In [59]:
```
p75=round(np.percentile(a=bankdata["age"],q=75),2)
```

In [60]:
```
print(p75)
```

```
47.0
```

In [61]:
```
p25=round(np.percentile(a=bankdata["age"],q=25),2)
```

Quantile

In [62]:
```
q25=round(np.quantile(a=bankdata["age"],q=.25),2)
```

In [63]:
```
print(q25)
```

```
32.0
```

In [64]:
```
q50=round(np.quantile(a=bankdata["age"],q=.50),2)
```

In [65]:
```
print(q50)
```

```
38.0
```

In [66]:
```
q75=round(np.quantile(a=bankdata["age"],q=.75),2)
```

In [67]:
```
print(q75)
```

```
47.0
```

In [68]:
```
l1=[]
length=len(bankdata['age'])
l1.append(length)
minvalue=round(bankdata['age'].min(),2)
l1.append(minvalue)
maxvalue=round(bankdata['age'].max(),2)
l1.append(maxvalue)
mean=round(bankdata['age'].mean(),2)
l1.append(mean)
median=round(bankdata['age'].median(),2)
l1.append(median)
```

```
l1.append(std)
p25=round(np.percentile(a=bankdata["age"],q=25),2)
p50=round(np.percentile(a=bankdata["age"],q=50),2)
p75=round(np.percentile(a=bankdata["age"],q=75),2)
l1.append(p25)
l1.append(p50)
l1.append(p75)
index1=["length","min","max","mean","median","std","p25","p50","p75"]
DataFrame=pd.DataFrame(l1,index1,columns=["age"])
```

In [69]:
```
DataFrame
```

Out[69]:

	age
length	4119.00
min	18.00
max	88.00
mean	40.11
median	38.00
std	10.31
p25	32.00
p50	38.00
p75	47.00

In [70]:
```
numcolumns=bankdata.select_dtypes(exclude="object").columns
```

In [71]:
```
numcolumns
```

Out[71]:
```
Index(['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate',
       'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'],
      dtype='object')
```

In [72]:
```
type(numcolumns)
```

Out[72]:
```
pandas.core.indexes.base.Index
```

In [73]:
```
list(numcolumns)
```

Out[73]:
```
['age',
 'duration',
 'campaign',
 'pdays',
 'previous',
 'emp.var.rate',
 'cons.price.idx',
 'cons.conf.idx',
 'euribor3m',
```

In [74]:
```
l2=[]
cols=list(numcolumns)
for i in cols:
    count=round(len(bankdata[i]),2)
    min=round(bankdata[i].min(),2)
    max=round(bankdata[i].max(),2)
    mean=round(bankdata[i].mean(),2)
    median=round(bankdata[i].median(),2)
    std=round(bankdata[i].std(),2)
    p25=round(np.quantile(bankdata[i],.25),2)
    p50=round(np.quantile(bankdata[i],.50),2)
    p75=round(np.quantile(bankdata[i],.75),2)
    l2.append([count,min,max,mean,median,std,p25,p50,p75])
Index_val=["count","min","max","mean","median","std","25%","50%","75%"]
df=pd.DataFrame(l2,columns=Index_val,index=cols)
df.T
```

Out[74]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx
count	4119.00	4119.00	4119.00	4119.00	4119.00	4119.00	4119.00
min	18.00	0.00	1.00	0.00	0.00	-3.40	92.20
max	88.00	3643.00	35.00	999.00	6.00	1.40	94.77
mean	40.11	256.79	2.54	960.42	0.19	0.08	93.58
median	38.00	181.00	2.00	999.00	0.00	1.10	93.75
std	10.31	254.70	2.57	191.92	0.54	1.56	0.58
25%	32.00	103.00	1.00	999.00	0.00	-1.80	93.08
50%	38.00	181.00	2.00	999.00	0.00	1.10	93.75
75%	47.00	317.00	3.00	999.00	0.00	1.40	93.99

we can achieve all the above with describe function only for numerical data

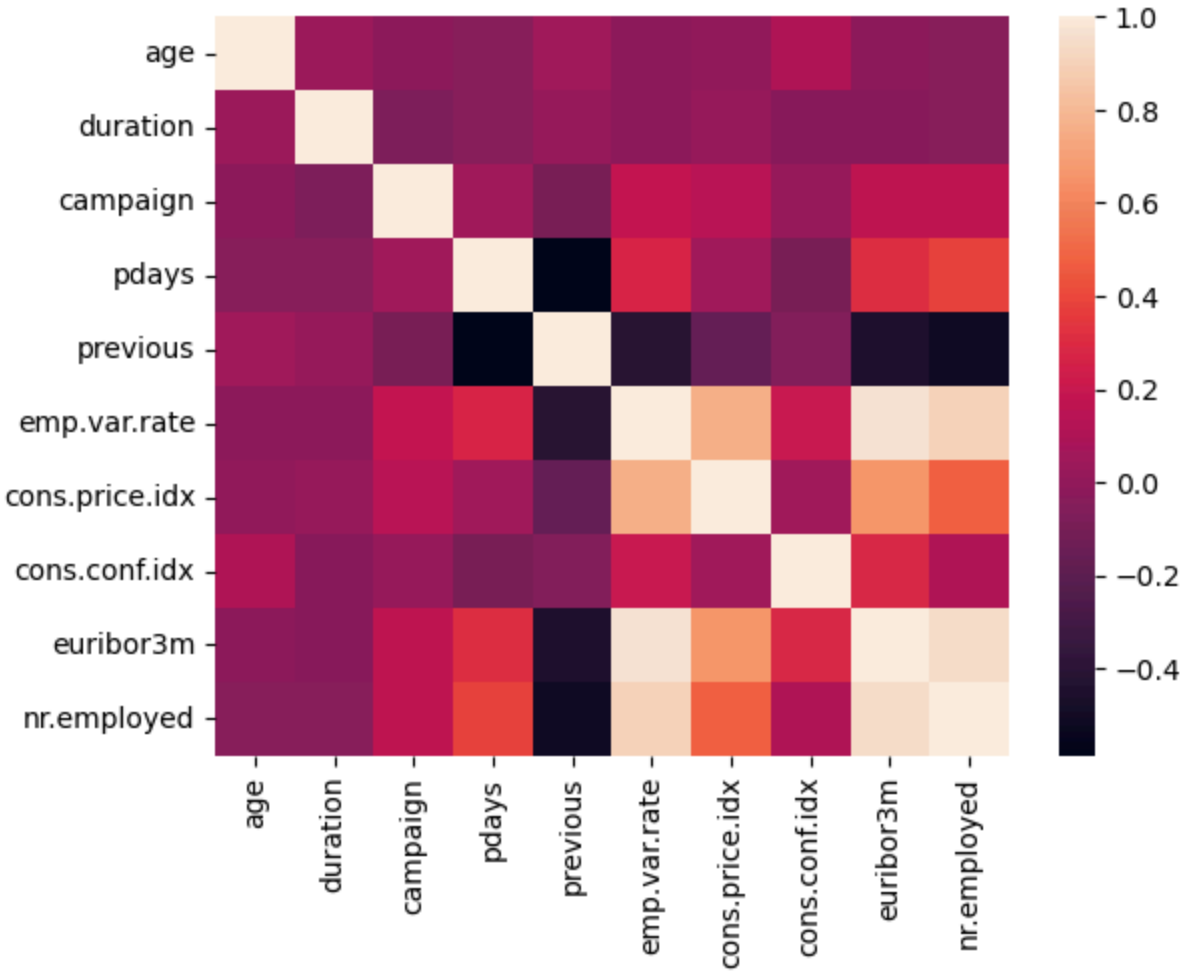
In [75]:
```
bankdata.describe()
```

Out[75]:

	age	duration	campaign	pdays	previous	emp.var.rat
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.00000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.08497
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.56311
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.40000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.80000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.10000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.40000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.40000

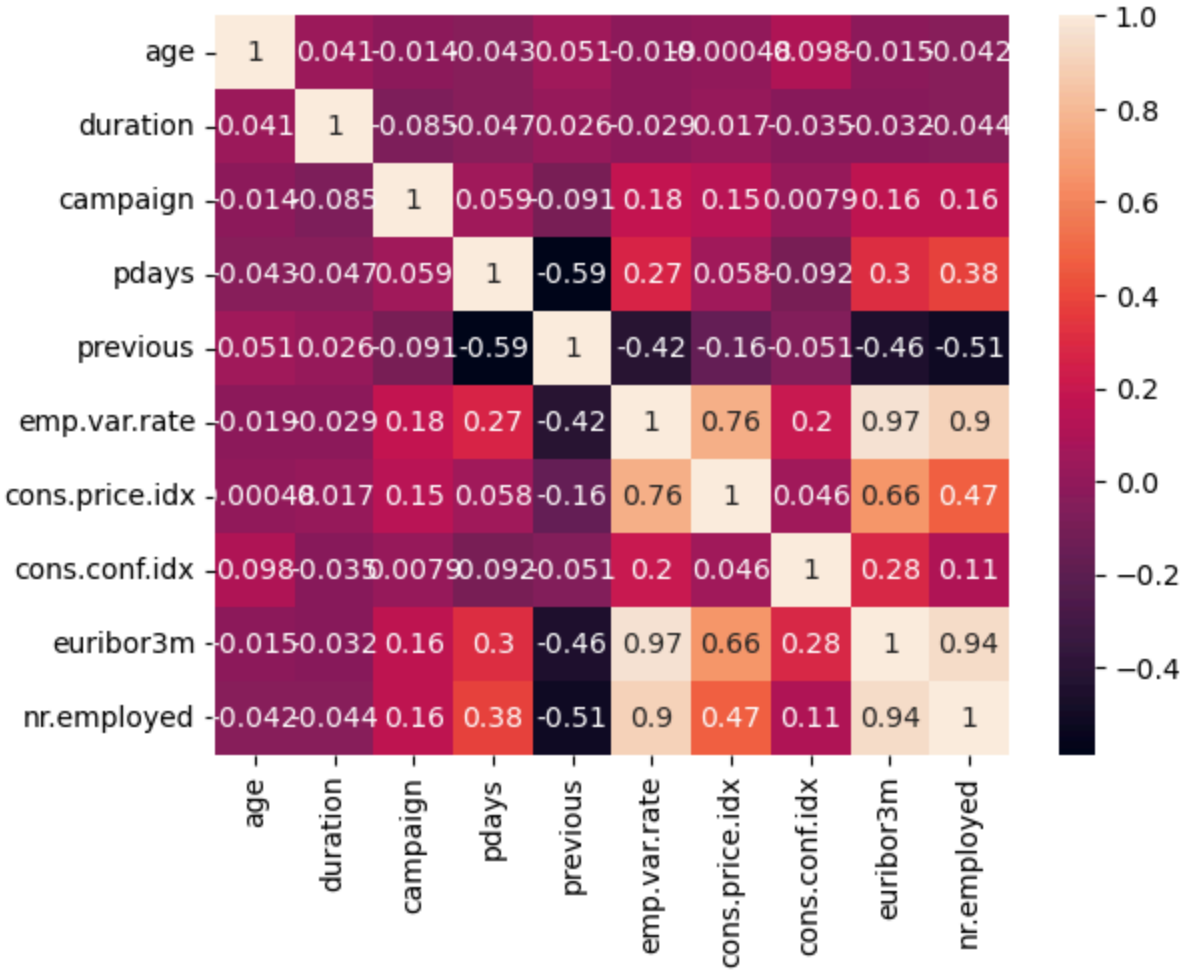
In [76]:
```
bank_corr=bankdata.corr(numeric_only=True)
sns.heatmap(bank_corr)
```

Out[76]: <Axes: >



In [77]:
```
bank_corr=bankdata.corr(numeric_only=True)
sns.heatmap(bank_corr,annot=True)
```

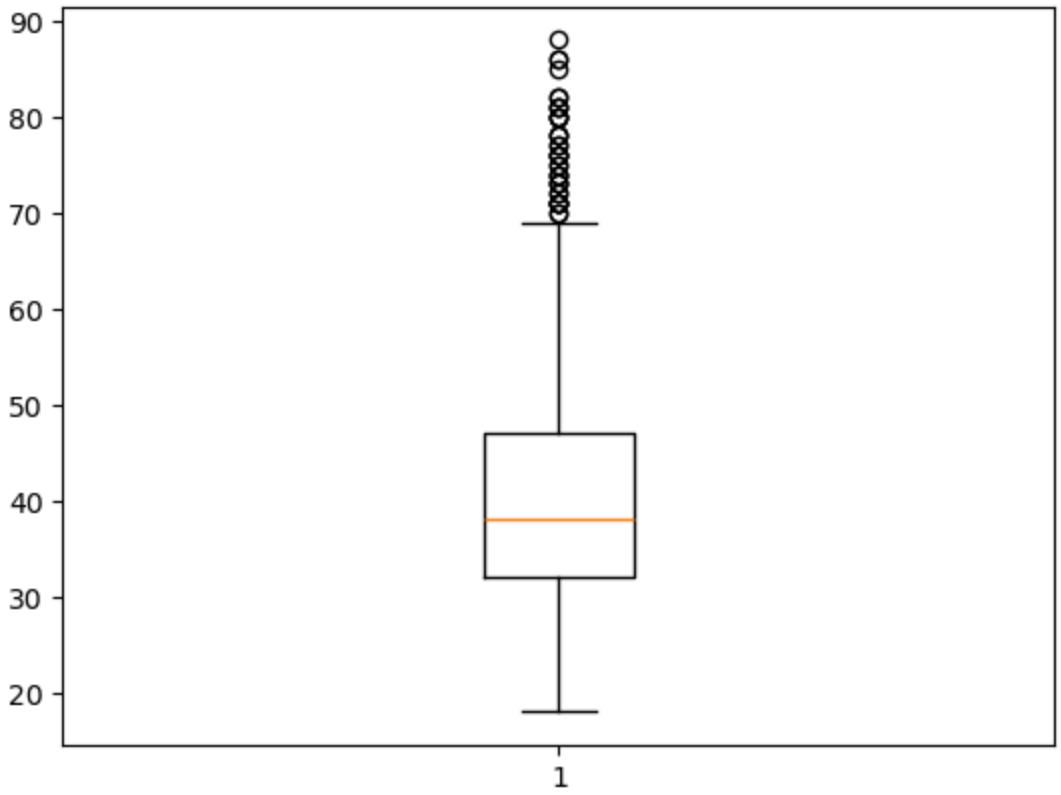

Out[77]: <Axes: >



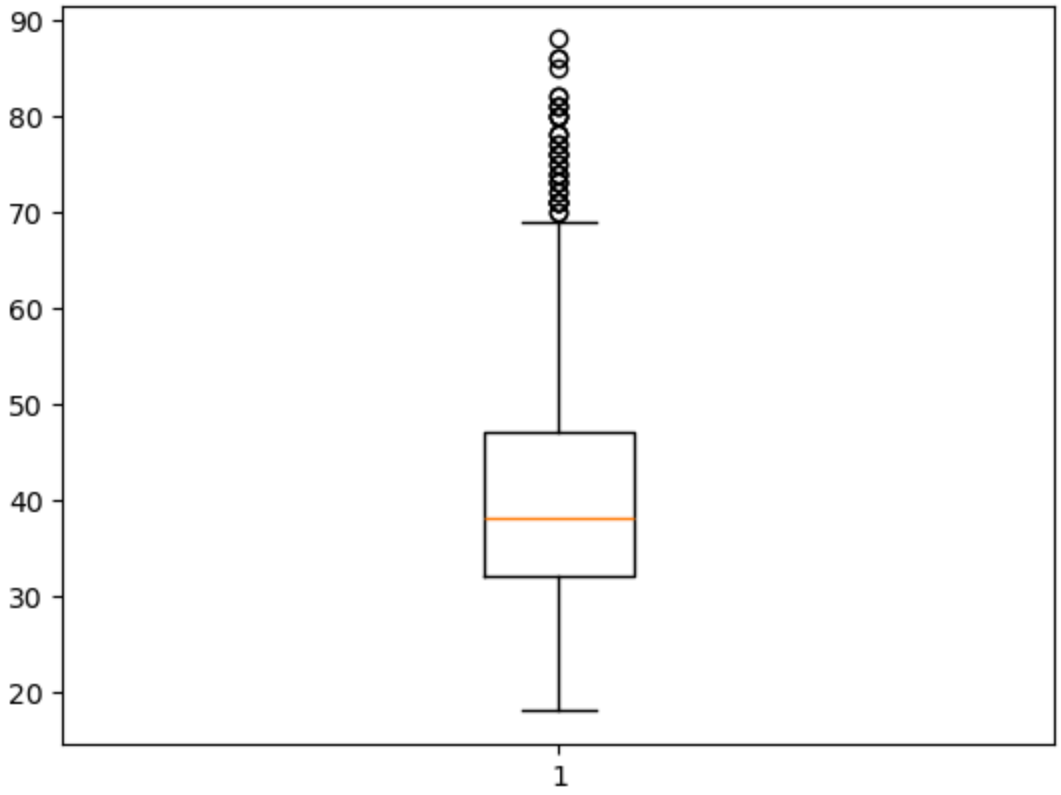
outlier Analysis

In [78]:
```
plt.boxplot(bankdata['age'])
```

Out[78]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x134ca8ad0>,
  <matplotlib.lines.Line2D at 0x134ca92d0>],
 'caps': [<matplotlib.lines.Line2D at 0x134ca9cd0>,
  <matplotlib.lines.Line2D at 0x134caa5d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x134a1f9d0>],
 'medians': [<matplotlib.lines.Line2D at 0x134caae10>],
 'fliers': [<matplotlib.lines.Line2D at 0x134cab710>],
 'means': []}
```

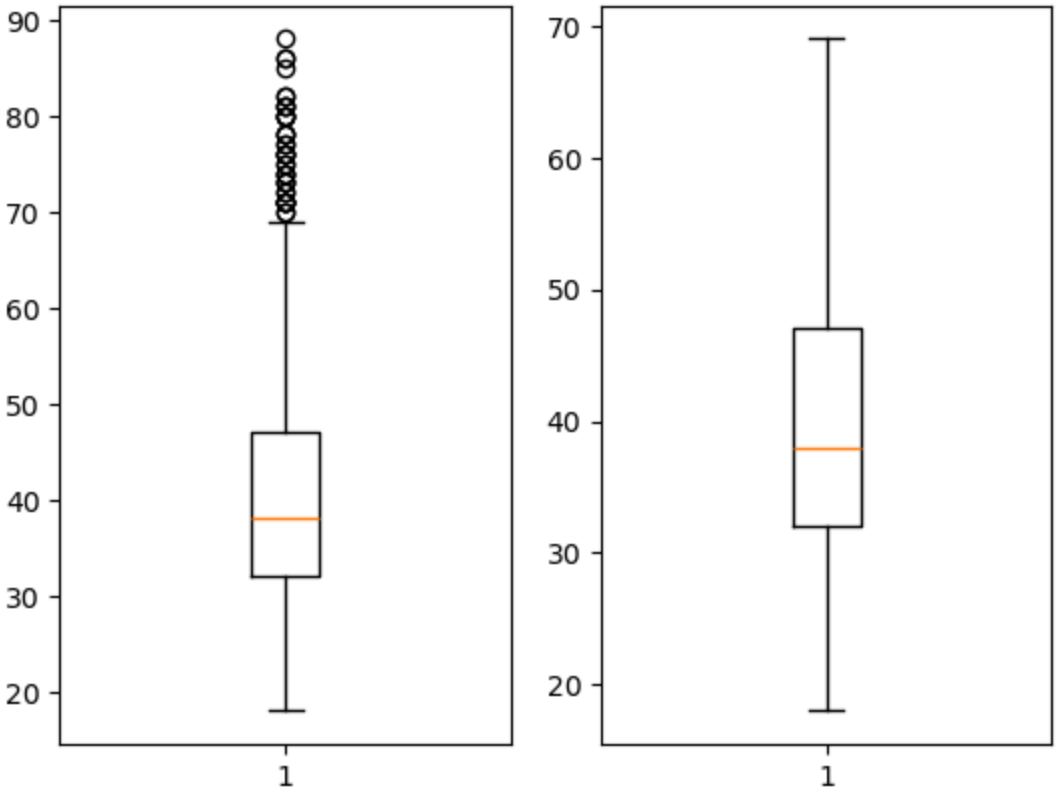


In [79]:
```
plt.boxplot(bankdata['age'])
plt.show()
```



In [80]:
```
l1=[]
q2=round(np.percentile(bankdata['age'],50),2)
```

```
q3=round(np.percentile(bankdata['age'],75),2)
iqr=q3-q1
lb=q1-1.5*(iqr)
ub=q3+1.5*(iqr)
medianvalue=bankdata['age'].median()
for i in bankdata['age'].values:
    if i >ub or i<lb:
        l1.append(medianvalue)
    else:
        l1.append(i)
bankdatacopy=bankdata.copy()
bankdatacopy["age"]=l1
plt.subplot(1,2,1).boxplot(bankdata["age"])
plt.subplot(1,2,2).boxplot(bankdatacopy["age"])
plt.show()
```



Np.where

In [81]:
```
q1=round(np.percentile(bankdata['age'],25),2)
q2=round(np.percentile(bankdata['age'],50),2)
q3=round(np.percentile(bankdata['age'],75),2)
iqr=q3-q1
lb=q1-1.5*(iqr)
ub=q3+1.5*(iqr)
con1=bankdata["age"]<lb
con2=bankdata["age"]>ub
con=con1 | con2
```

In [82]:
```
l4
```

Out[82]:
```
array([30., 39., 25., ..., 27., 58., 34.])
```

Bivariate Analysis

Categorical vs Categorical

In [83]:
```
bankdata.dtypes
```

Out[83]:
```
age                int64
job               object
marital           object
education         object
default           object
housing           object
loan              object
contact           object
month             object
day_of_week       object
duration           int64
campaign           int64
pdays              int64
previous           int64
poutcome          object
emp.var.rate     float64
cons.price.idx   float64
cons.conf.idx    float64
euribor3m        float64
nr.employed      float64
y                 object
dtype: object
```

In [84]:
```
bankdata.select_dtypes(include="object")
```

Out[84]:

	job	marital	education	default	housing	loan	contact	m
0	blue-collar	married	basic.9y	no	yes	no	cellular	
1	services	single	high.school	no	no	no	telephone	
2	services	married	high.school	no	yes	no	telephone	
3	services	married	basic.9y	no	unknown	unknown	telephone	
4	admin.	married	university.degree	no	yes	no	cellular	
...	...	...	...	...	...	...	...	
4114	admin.	married	basic.6y	no	yes	yes	cellular	
4115	admin.	married	high.school	no	yes	no	telephone	
4116	student	single	high.school	no	no	no	cellular	
4117	admin.	married	high.school	no	no	no	cellular	
4118	management	single	high.school	no	yes	no	cellular	

4119 rows × 11 columns

In [85]:
```
bankdata.select_dtypes(include="object").columns
```

Out[85]:
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contac
t',
       'month', 'day_of_week', 'poutcome', 'y'],
      dtype='object')
```

Bivariate Analysis (categorical)

JOB vs Education

In [86]:
```
bankdata["job"]
```

Out[86]:
```
0       blue-collar
1          services
2          services
3          services
4             admin.
           ...
4114          admin.
4115          admin.
4116         student
4117          admin.
4118     management
Name: job, Length: 4119, dtype: object
```

Out[87]:
```
0               basic.9y
1            high.school
2            high.school
3               basic.9y
4       university.degree
              ...
4114            basic.6y
4115         high.school
4116         high.school
4117         high.school
4118         high.school
Name: education, Length: 4119, dtype: object
```

In [88]:
```
bankdata["education"].sum
```

Out[88]:
```
<bound method Series.sum of 0               basic.9y
1            high.school
2            high.school
3               basic.9y
4       university.degree
              ...
4114            basic.6y
4115         high.school
4116         high.school
4117         high.school
4118         high.school
Name: education, Length: 4119, dtype: object>
```

In [89]:
```
bankdata["education"].describe()
```

Out[89]:
```
count                 4119
unique                   8
top       university.degree
freq                  1264
Name: education, dtype: object
```

In [90]:
```
bankdata["education"].value_counts()
```

Out[90]:
```
education
university.degree      1264
high.school             921
basic.9y                574
professional.course     535
basic.4y                429
basic.6y                228
unknown                 167
illiterate                1
Name: count, dtype: int64
```

In [91]:
```
bankdata["job"].value_counts()
```

Out[91]:
```
job
admin.           1012
blue-collar       884
technician        691
services          393
management        324
retired           166
self-employed     159
entrepreneur      148
unemployed        111
housemaid         110
student            82
unknown            39
Name: count, dtype: int64
```

In [92]:
```
# find number of people who are university.dergree and umemployed
con1=bankdata["education"]=="university.degree"
con2=bankdata["job"]=="unemployed"
con=con1 & con2
bankdata[con]
```

Out[92]:

	age	job	marital	education	default	housing	loan	con
86	37	unemployed	single	university.degree	no	yes	yes	cel
153	31	unemployed	single	university.degree	no	yes	no	cel
266	27	unemployed	single	university.degree	no	no	no	teleph
299	28	unemployed	single	university.degree	no	yes	no	cel
337	43	unemployed	married	university.degree	unknown	yes	no	teleph
561	39	unemployed	married	university.degree	no	yes	no	teleph
674	58	unemployed	married	university.degree	no	yes	no	cel
692	45	unemployed	married	university.degree	no	no	no	teleph
1044	31	unemployed	married	university.degree	no	no	no	cel
1049	37	unemployed	unknown	university.degree	no	no	no	cel
1070	30	unemployed	single	university.degree	no	yes	yes	cel
1166	43	unemployed	married	university.degree	unknown	no	no	teleph
1220	56	unemployed	divorced	university.degree	unknown	yes	no	teleph
1297	53	unemployed	married	university.degree	no	unknown	unknown	cel
1427	32	unemployed	married	university.degree	no	yes	no	cel
1666	36	unemployed	married	university.degree	no	no	no	cel
1684	31	unemployed	single	university.degree	no	yes	no	cel
1879	37	unemployed	married	university.degree	no	no	yes	teleph
1923	29	unemployed	single	university.degree	no	yes	no	cel
1932	39	unemployed	single	university.degree	no	yes	no	cel
2117	31	unemployed	single	university.degree	no	yes	no	cel
2220	31	unemployed	married	university.degree	no	no	no	cel
2267	39	unemployed	married	university.degree	unknown	yes	no	cel
2321	39	unemployed	married	university.degree	no	yes	no	teleph
2371	52	unemployed	married	university.degree	no	no	no	cel
2459	55	unemployed	married	university.degree	no	yes	no	cel
2774	34	unemployed	single	university.degree	no	no	no	teleph
2872	35	unemployed	married	university.degree	no	no	no	cel
3007	39	unemployed	married	university.degree	no	no	no	teleph
3010	27	unemployed	single	university.degree	no	no	no	teleph
3083	39	unemployed	married	university.degree	no	no	no	teleph
3112	26	unemployed	single	university.degree	no	yes	no	cel

	age	job	marital	education	default	housing	loan	con
3312	29	unemployed	single	university.degree	no	no	no	teleph
3400	43	unemployed	married	university.degree	unknown	unknown	unknown	teleph
3816	33	unemployed	single	university.degree	no	no	no	cel
3993	43	unemployed	married	university.degree	unknown	no	no	teleph
4062	31	unemployed	divorced	university.degree	no	yes	yes	cel

37 rows × 21 columns

In [93]:
```
telephonelist=[]
cellularlist=[]
for i in bankdata["job"].unique():
    con1=bankdata["job"] == i
    con2=bankdata["contact"]=="telephone"
    con3=bankdata["contact"]=="cellular"
    telephonecon=con1 & con2
    cellularcon=con1 & con3
    bankdata[telephonecon]
    telephonelist.append(len(bankdata[telephonecon]))
    bankdata[cellularcon]
    cellularlist.append(len(bankdata[cellularcon]))
```

In [94]:
```
telephoneandcellularcounts=pd.DataFrame(zip(telephonelist,cellularlist),colu
```

Pandas has the crosstab which is used to perform bi variate analysis

In [95]:
```
con1=bankdata["job"]
con2=bankdata["education"]
pd.crosstab(con1,con2)
```

Out[95]:

education job	basic.4y	basic.6y	basic.9y	high.school	illiterate	professional.course
admin.	8	20	44	311	0	38
blue-collar	222	152	324	89	0	49
entrepreneur	18	5	23	17	0	14
housemaid	52	9	5	11	0	8
management	13	8	20	41	0	7
retired	59	6	11	24	1	28
self-employed	11	2	28	15	0	12
services	16	12	56	254	0	25
student	2	0	5	35	0	8
technician	7	10	34	95	0	330
unemployed	13	3	18	23	0	14
unknown	8	1	6	6	0	2

In [96]:
```
con1=bankdata["job"]
con2=bankdata["education"]
con3=bankdata["marital"]
pd.crosstab(con1,con2,con3)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[96], line 4
      2 con2=bankdata["education"]
      3 con3=bankdata["marital"]
----> 4 pd.crosstab(con1,con2,con3)

File /Applications/Anaconda/anaconda3/lib/python3.11/site-packages/pandas/core/reshape/pivot.py:671, in crosstab(index, columns, values, rownames, colnames, aggfunc, margins, margins_name, dropna, normalize)
    668     raise ValueError("aggfunc cannot be used without values.")
    670 if values is not None and aggfunc is None:
--> 671     raise ValueError("values cannot be used without an aggfunc.")
    673 if not is_nested_list_like(index):
    674     index = [index]

ValueError: values cannot be used without an aggfunc.
```

we should do three or four columns like this

In [97]:
```
con1=bankdata["job"]
```

```
con3=bankdata["marital"]
con=[con1,con2]
pd.crosstab(con,con3)
```

Out[97]:

job	education	marital	divorced	married	single	unknown
admin.	basic.4y		1	5	2	0
	basic.6y		2	12	6	0
	basic.9y		7	27	10	0
	high.school		41	164	105	1
	professional.course		3	21	14	0
...	...		...	...	...	...
unknown	basic.9y		0	4	2	0
	high.school		2	2	2	0
	professional.course		0	2	0	0
	university.degree		0	1	2	0
	unknown		0	12	1	0

84 rows × 4 columns

In [98]:

```
con1=bankdata["job"]
con2=bankdata["education"]
con3=bankdata["marital"]
con4=bankdata["age"]
con=[con1,con2]
con5=[con3,con4]
pd.crosstab(con,con5)
```

Out[98]:

	marital								divorced	...	single
	age	24	25	28	29	30	31	32	33	34	35	...	76
job	education												
admin.	basic.4y	0	0	0	0	0	0	0	0	0	0	...	0
	basic.6y	0	0	0	0	0	0	0	0	0	0	...	0
	basic.9y	0	0	0	0	0	0	1	0	0	0	...	0
	high.school	0	0	0	0	1	0	0	0	0	2	...	0
	professional.course	0	0	0	0	0	0	0	0	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
unknown	basic.9y	0	0	0	0	0	0	0	0	0	0	...	0
	high.school	0	0	0	0	0	0	0	0	0	0	...	0
	professional.course	0	0	0	0	0	0	0	0	0	0	...	0
	university.degree	0	0	0	0	0	0	0	0	0	0	...	0
	unknown	0	0	0	0	0	0	0	0	0	0	...	0

84 rows × 165 columns

Inorder to find if there is any relation between them we need find the corelation coefficient

Corelation coefficient

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

- pearson co relation coefficient gives amount of relation between the variables
- denoted by r
- positive means r = o to 1
- negative relation r=-1 to 0

- no relation r ==0
- in python we have corr function under pandas library
- it will give the covariance matrix
- diagonal represets varianve
- upper triangle and lower trainge represets co-variance

In [99]:
```
bankdata.corr()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[99], line 1
----> 1 bankdata.corr()

File /Applications/Anaconda/anaconda3/lib/python3.11/site-packages/pandas/core/frame.py:10704, in DataFrame.corr(self, method, min_periods, numeric_only)
  10702 cols = data.columns
  10703 idx = cols.copy()
> 10704 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10706 if method == "pearson":
  10707     correl = libalgos.nancorr(mat, minp=min_periods)

File /Applications/Anaconda/anaconda3/lib/python3.11/site-packages/pandas/core/frame.py:1889, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1887 if dtype is not None:
   1888     dtype = np.dtype(dtype)
-> 1889 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1890 if result.dtype is not dtype:
   1891     result = np.array(result, dtype=dtype, copy=False)

File /Applications/Anaconda/anaconda3/lib/python3.11/site-packages/pandas/core/internals/managers.py:1656, in BlockManager.as_array(self, dtype, copy, na_value)
   1654         arr.flags.writeable = False
   1655 else:
-> 1656     arr = self._interleave(dtype=dtype, na_value=na_value)
   1657     # The underlying data was copied within _interleave, so no need
   1658     # to further copy if copy=True or setting na_value
   1660 if na_value is lib.no_default:

File /Applications/Anaconda/anaconda3/lib/python3.11/site-packages/pandas/core/internals/managers.py:1715, in BlockManager._interleave(self, dtype, na_value)
   1713     else:
   1714         arr = blk.get_values(dtype)
-> 1715     result[rl.indexer] = arr
   1716     itemmask[rl.indexer] = 1
   1718 if not itemmask.all():

ValueError: could not convert string to float: 'blue-collar'
```

In [100…
```
bankdata.corr(numeric_only=True)
```

Out[100…

	age	duration	campaign	pdays	previous	emp.var.rate
age	1.000000	0.041299	-0.014169	-0.043425	0.050931	-0.019192
duration	0.041299	1.000000	-0.085348	-0.046998	0.025724	-0.028848
campaign	-0.014169	-0.085348	1.000000	0.058742	-0.091490	0.176079
pdays	-0.043425	-0.046998	0.058742	1.000000	-0.587941	0.270684
previous	0.050931	0.025724	-0.091490	-0.587941	1.000000	-0.415238
emp.var.rate	-0.019192	-0.028848	0.176079	0.270684	-0.415238	1.000000
cons.price.idx	-0.000482	0.016672	0.145021	0.058472	-0.164922	0.755155
cons.conf.idx	0.098135	-0.034745	0.007882	-0.092090	-0.051420	0.195022
euribor3m	-0.015033	-0.032329	0.159435	0.301478	-0.458851	0.970308
nr.employed	-0.041936	-0.044218	0.161037	0.381983	-0.514853	0.897173

Categorical to numerical

- categorical to numerical
- it is very important to convert categorical to nurerical because understands only numbers
- we have different types of methods to convert categorical data to numerical data
  - 1)Map method
  - 2)np.where
  - 3)One hot Encoder
  - 4)Label Encoder

In [102…
```
bankdata["contact"].unique()
```

Out[102…
```
array(['cellular', 'telephone'], dtype=object)
```

In [103…
```
d={"cellular":1,"telephone":0}
```

In [104…
```
bankdata["contact"].map(d)
```

Out[104…
```
0       1
1       0
2       0
3       0
4       1
       ..
4114    1
4115    0
4116    1
4117    1
4118    1
Name: contact, Length: 4119, dtype: int64
```

In [105…
```
bankdata["contact"]=bankdata["contact"].map(d)
```

Another starting for categorical to Numerical

In [101…
```
bankdata
```

Out[101…

	age	job	marital	education	default	housing	loan	conta
0	30	blue-collar	married	basic.9y	no	yes	no	cellul
1	39	services	single	high.school	no	no	no	telephor
2	25	services	married	high.school	no	yes	no	telephor
3	38	services	married	basic.9y	no	unknown	unknown	telephor
4	47	admin.	married	university.degree	no	yes	no	cellul
...	...	...	...	...	...	...	...	
4114	30	admin.	married	basic.6y	no	yes	yes	cellul
4115	39	admin.	married	high.school	no	yes	no	telephor
4116	27	student	single	high.school	no	no	no	cellul
4117	58	admin.	married	high.school	no	no	no	cellul
4118	34	management	single	high.school	no	yes	no	cellul

4119 rows × 21 columns

In [106…
```
bankdata.select_dtypes(include=object)
```

Out[106…

	job	marital	education	default	housing	loan	month	day_
0	blue-collar	married	basic.9y	no	yes	no	may	
1	services	single	high.school	no	no	no	may	
2	services	married	high.school	no	yes	no	jun	
3	services	married	basic.9y	no	unknown	unknown	jun	
4	admin.	married	university.degree	no	yes	no	nov	
...	...	...	...	...	...	...	...	
4114	admin.	married	basic.6y	no	yes	yes	jul	
4115	admin.	married	high.school	no	yes	no	jul	
4116	student	single	high.school	no	no	no	may	
4117	admin.	married	high.school	no	no	no	aug	
4118	management	single	high.school	no	yes	no	nov	

4119 rows × 10 columns

categorical to numerical

it is very important to convert categorical to nurerical because understands only numbers

we have different types of methods to convert categorical data to numerical data

- 1)Map method
- 2)np.where
- 3)One hot Encoder
- 4)Label Encoder

In [107…
```
bankdata["poutcome"]
```

Out[107…
```
0       nonexistent
1       nonexistent
2       nonexistent
3       nonexistent
4       nonexistent
           ...
4114    nonexistent
4115    nonexistent
4116        failure
4117    nonexistent
4118    nonexistent
```

In [108…
```
bankdata["poutcome"].unique()
```

Out[108…
```
array(['nonexistent', 'failure', 'success'], dtype=object)
```

MAP Method

- First get the unique labels for each column
- create a dictionary by provididng values to each unique label
- In this example for bankdata["poutcome"] we have 3 values nonexistent,failure and success
- we need to assign alphabetically
- nonexistent=0
- failure=1`
- success=2

create dictionary as below

In [109…
```
d={"nonexistent":0,"failure":1,"success":2}
```

In [110…
```
type(d)
```

Out[110…
```
dict
```

In [112…
```
bankdata["poutcome"].map(d)
```

Out[112…
```
0       0
1       0
2       0
3       0
4       0
       ..
4114    0
4115    0
4116    1
4117    0
4118    0
Name: poutcome, Length: 4119, dtype: int64
```

In [113…
```
bankdata["poutcome"]=bankdata["poutcome"].map(d)
```

Out[114…

	age	job	marital	education	default	housing	loan	contact
0	30	blue-collar	married	basic.9y	no	yes	no	1
1	39	services	single	high.school	no	no	no	0
2	25	services	married	high.school	no	yes	no	0
3	38	services	married	basic.9y	no	unknown	unknown	0
4	47	admin.	married	university.degree	no	yes	no	1
...	...	...	...	...	...	...	...	...
4114	30	admin.	married	basic.6y	no	yes	yes	1
4115	39	admin.	married	high.school	no	yes	no	0
4116	27	student	single	high.school	no	no	no	1
4117	58	admin.	married	high.school	no	no	no	1
4118	34	management	single	high.school	no	yes	no	1

4119 rows × 21 columns

In [115…
```
bankdata.columns
```

Out[115…
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

In [116…
```
colcat=bankdata.select_dtypes(include="object").columns
```

In [117…
```
colcat
```

Out[117…
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'mont
h',
       'day_of_week', 'y'],
      dtype='object')
```

In [121…
```
colcat["education"]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[121], line 1
----> 1 colcat["education"]

File /Applications/Anaconda/anaconda3/lib/python3.11/site-packages/pandas/core/indexes/base.py:5383, in Index.__getitem__(self, key)
   5380     else:
   5381         key = np.asarray(key, dtype=bool)
-> 5383 result = getitem(key)
   5384 # Because we ruled out integer above, we always get an arraylike here
   5385 if result.ndim > 1:

IndexError: only integers, slices (`:`), ellipsis (`...`), numpy.newaxis (`None`) and integer or boolean arrays are valid indices
```

In [122…

```
for i in colcat:
    labels=list(bankdata[i].unique())
    values= [i for i in range(len(labels))]
    d=dict(zip(labels,values))
    bankdata[i]=bankdata[i].map(d)
bankdata
```

Out[122…

	age	job	marital	education	default	housing	loan	contact	month	day_of_w
0	30	0	0	0	0	0	0	1	0	
1	39	1	1	1	0	1	0	0	0	
2	25	1	0	1	0	0	0	0	1	
3	38	1	0	0	0	2	1	0	1	
4	47	2	0	2	0	0	0	1	2	
...	...	...	...	...	...	...	...	...	...	
4114	30	2	0	4	0	0	2	1	4	
4115	39	2	0	1	0	0	0	0	4	
4116	27	7	1	1	0	1	0	1	0	
4117	58	2	0	1	0	1	0	1	5	
4118	34	6	1	1	0	0	0	1	2	

4119 rows × 21 columns

Above step explanation:

First get the object elements

Then for each object element get the number of unique fields

assign values to each field

then create a dictionary

LAbel encoder

label encodeer is a package under scikit learn

Scikit learn package is heart of machine learning

In scikit learn package we have a method called PreProcessing

In preprocessing we have LableEncoder

any scikit package we have 3 steps

step 1 Read the package

step 2 Save the package

step 3 Apply Fit transform

In [123…
```
bankdata=pd.read_csv("/Users/shashankreddy/Desktop/Datafiles/bank-additional
```

In [124…
```
#read the package
from sklearn.preprocessing import LabelEncoder
```

In [125…
```
#save package
#create an objct for Label Encoder
le=LabelEncoder()
```

In [126…
```
#apply fit transform
bankdata["poutcome"]=le.fit_transform(bankdata["poutcome"])
```

In [127…
```
bankdata
```

Out [127…

	age	job	marital	education	default	housing	loan	conta
0	30	blue-collar	married	basic.9y	no	yes	no	cellul
1	39	services	single	high.school	no	no	no	telephor
2	25	services	married	high.school	no	yes	no	telephor
3	38	services	married	basic.9y	no	unknown	unknown	telephor
4	47	admin.	married	university.degree	no	yes	no	cellul
...	...	...	...	...	...	...	...	
4114	30	admin.	married	basic.6y	no	yes	yes	cellul
4115	39	admin.	married	high.school	no	yes	no	telephor
4116	27	student	single	high.school	no	no	no	cellul
4117	58	admin.	married	high.school	no	no	no	cellul
4118	34	management	single	high.school	no	yes	no	cellul

4119 rows × 21 columns

doing for all columns using Label encoder

In [128…

```
bankdata=pd.read_csv("/Users/shashankreddy/Desktop/Datafiles/bank-additional
catcols=bankdata.select_dtypes(include="object").columns
```

In [129…

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in catcols:
    bankdata[i]=le.fit_transform(bankdata[i])
bankdata
```

Out [129…

	age	job	marital	education	default	housing	loan	contact	month	day_of_w
0	30	1	1	2	0	2	0	0	6	
1	39	7	2	3	0	0	0	1	6	
2	25	7	1	3	0	2	0	1	4	
3	38	7	1	2	0	1	1	1	4	
4	47	0	1	6	0	2	0	0	7	
…	…	…	…	…	…	…	…	…	…	
4114	30	0	1	1	0	2	2	0	3	
4115	39	0	1	3	0	2	0	1	3	
4116	27	8	2	3	0	0	0	0	6	
4117	58	0	1	3	0	0	0	0	1	
4118	34	4	2	3	0	2	0	0	7	

4119 rows × 21 columns

np.where

- np where is applicable only for binary values
- it atkes three arguments
- condition,true value,false value

In [132…

```
bankdata=pd.read_csv("/Users/shashankreddy/Desktop/Datafiles/bank-additional
con=bankdata["contact"]=="cellular"
bankdata["contact"]=np.where(con,0,1)
```

In [133…

```
bankdata
```

Out[133...

	age	job	marital	education	default	housing	loan	contact
0	30	blue-collar	married	basic.9y	no	yes	no	0
1	39	services	single	high.school	no	no	no	1
2	25	services	married	high.school	no	yes	no	1
3	38	services	married	basic.9y	no	unknown	unknown	1
4	47	admin.	married	university.degree	no	yes	no	0
...	...	...	...	...	...	...	...	...
4114	30	admin.	married	basic.6y	no	yes	yes	0
4115	39	admin.	married	high.school	no	yes	no	1
4116	27	student	single	high.school	no	no	no	0
4117	58	admin.	married	high.school	no	no	no	0
4118	34	management	single	high.school	no	yes	no	0

4119 rows × 21 columns

one hot Encoder

- If one is on means another is off
- on represented with 1
- off represented with 0
- if we apply one got encoder on case-status it will create two extra columns called

  – case status denied

  –case status certified

We can perform one hot encoding using pd.get_dummies() method

- read the data
- choose one column
- apply pd.get_dummies

In [134…
```
bankdata=pd.read_csv("/Users/shashankreddy/Desktop/Datafiles/bank-additional
data=bankdata["marital"]
pd.get_dummies(data)
```

Out[134…

	divorced	married	single	unknown
0	False	True	False	False
1	False	False	True	False
2	False	True	False	False
3	False	True	False	False
4	False	True	False	False
…	…	…	…	…
4114	False	True	False	False
4115	False	True	False	False
4116	False	False	True	False
4117	False	True	False	False
4118	False	False	True	False

4119 rows × 4 columns

In [135…
```
bankdata=pd.read_csv("/Users/shashankreddy/Desktop/Datafiles/bank-additional
data=bankdata["marital"]
pd.get_dummies(data,dtype=int)
```

Out[135…

	divorced	married	single	unknown
0	0	1	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	0	0
4	0	1	0	0
…	…	…	…	…
4114	0	1	0	0
4115	0	1	0	0
4116	0	0	1	0
4117	0	1	0	0
4118	0	0	1	0

4119 rows × 4 columns

for all columns

In [136…

```
bankdata=pd.read_csv("/Users/shashankreddy/Desktop/Datafiles/bank-additional
catcols=bankdata.select_dtypes(include="object").columns
for i in catcols:
   print(pd.get_dummies(bankdata[i]))
```

```
      admin.  blue-collar  entrepreneur  housemaid  management  retired  \
0      False         True         False      False       False    False
1      False        False         False      False       False    False
2      False        False         False      False       False    False
3      False        False         False      False       False    False
4       True        False         False      False       False    False
...      ...          ...           ...        ...         ...      ...
4114    True        False         False      False       False    False
4115    True        False         False      False       False    False
4116   False        False         False      False       False    False
4117    True        False         False      False       False    False
4118   False        False         False      False        True    False

      self-employed  services  student  technician  unemployed  unknown
0             False     False    False       False       False    False
1             False      True    False       False       False    False
2             False      True    False       False       False    False
3             False      True    False       False       False    False
4             False     False    False       False       False    False
...             ...       ...      ...         ...         ...      ...
4114          False     False    False       False       False    False
4115          False     False    False       False       False    False
4116          False     False     True       False       False    False
4117          False     False    False       False       False    False
4118          False     False    False       False       False    False

[4119 rows x 12 columns]
      divorced  married  single  unknown
0        False     True   False    False
1        False    False    True    False
2        False     True   False    False
3        False     True   False    False
4        False     True   False    False
...        ...      ...     ...      ...
4114     False     True   False    False
4115     False     True   False    False
4116     False    False    True    False
4117     False     True   False    False
4118     False    False    True    False

[4119 rows x 4 columns]
      basic.4y  basic.6y  basic.9y  high.school  illiterate  \
0        False     False      True        False       False
1        False     False     False         True       False
2        False     False     False         True       False
3        False     False      True        False       False
4        False     False     False        False       False
...        ...       ...       ...          ...         ...
4114     False      True     False        False       False
4115     False     False     False         True       False
4116     False     False     False         True       False
4117     False     False     False         True       False
4118     False     False     False         True       False

      professional.course  university.degree  unknown
0                   False              False    False
```

```
1                  False             False    False
2                  False             False    False
3                  False             False    False
4                  False              True    False
...                  ...              ...      ...
4114               False             False    False
4115               False             False    False
4116               False             False    False
4117               False             False    False
4118               False             False    False

[4119 rows x 8 columns]
       no  unknown    yes
0     True    False  False
1     True    False  False
2     True    False  False
3     True    False  False
4     True    False  False
...    ...      ...    ...
4114  True    False  False
4115  True    False  False
4116  True    False  False
4117  True    False  False
4118  True    False  False

[4119 rows x 3 columns]
         no  unknown    yes
0     False    False   True
1      True    False  False
2     False    False   True
3     False     True  False
4     False    False   True
...     ...      ...    ...
4114  False    False   True
4115  False    False   True
4116   True    False  False
4117   True    False  False
4118  False    False   True

[4119 rows x 3 columns]
         no  unknown    yes
0      True    False  False
1      True    False  False
2      True    False  False
3     False     True  False
4      True    False  False
...     ...      ...    ...
4114  False    False   True
4115   True    False  False
4116   True    False  False
4117   True    False  False
4118   True    False  False

[4119 rows x 3 columns]
```

```
1        False       True
2        False       True
3        False       True
4         True      False
...        ...        ...
4114      True      False
4115     False       True
4116      True      False
4117      True      False
4118      True      False

[4119 rows x 2 columns]
        apr    aug    dec    jul    jun    mar    may    nov    oct    sep
0     False  False  False  False  False  False   True  False  False  False
1     False  False  False  False  False  False   True  False  False  False
2     False  False  False  False   True  False  False  False  False  False
3     False  False  False  False   True  False  False  False  False  False
4     False  False  False  False  False  False  False   True  False  False
...     ...    ...    ...    ...    ...    ...    ...    ...    ...    ...
4114  False  False  False   True  False  False  False  False  False  False
4115  False  False  False   True  False  False  False  False  False  False
4116  False  False  False  False  False  False   True  False  False  False
4117  False   True  False  False  False  False  False  False  False  False
4118  False  False  False  False  False  False  False   True  False  False

[4119 rows x 10 columns]
        fri    mon    thu    tue    wed
0      True  False  False  False  False
1      True  False  False  False  False
2     False  False  False  False   True
3      True  False  False  False  False
4     False   True  False  False  False
...     ...    ...    ...    ...    ...
4114  False  False   True  False  False
4115   True  False  False  False  False
4116  False   True  False  False  False
4117   True  False  False  False  False
4118  False  False  False  False   True

[4119 rows x 5 columns]
      failure  nonexistent  success
0       False         True    False
1       False         True    False
2       False         True    False
3       False         True    False
4       False         True    False
...       ...          ...      ...
4114    False         True    False
4115    False         True    False
4116     True        False    False
4117    False         True    False
4118    False         True    False

[4119 rows x 3 columns]
```

```
1     True  False
2     True  False
3     True  False
4     True  False
...    ...    ...
4114  True  False
4115  True  False
4116  True  False
4117  True  False
4118  True  False

[4119 rows x 2 columns]
```

In [138…
```
bankdata
```

Out[138…

	age	job	marital	education	default	housing	loan	conta
0	30	blue-collar	married	basic.9y	no	yes	no	cellul
1	39	services	single	high.school	no	no	no	telephor
2	25	services	married	high.school	no	yes	no	telephor
3	38	services	married	basic.9y	no	unknown	unknown	telephor
4	47	admin.	married	university.degree	no	yes	no	cellul
...	...	...	...	...	...	...	...	
4114	30	admin.	married	basic.6y	no	yes	yes	cellul
4115	39	admin.	married	high.school	no	yes	no	telephor
4116	27	student	single	high.school	no	no	no	cellul
4117	58	admin.	married	high.school	no	no	no	cellul
4118	34	management	single	high.school	no	yes	no	cellul

4119 rows × 21 columns

standardization

standardization

- standardization is a technique to get all data in single scale
- different columns has different data with different units
- some has large and some has small values
- It is important to keep all values under one scale

- we have two methods to overcome this issue
    - standardization
    - Normalization

standarization

- Z score
- values Ranges from -3 to +3

$$Z = \frac{x - \mu}{\sigma}$$

normalization

- Normalization
    - min max scalar
    - values ranges from 0 to 1

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

standarization for nr employed

In [139…
```
bankdata
```

Out[139…

	age	job	marital	education	default	housing	loan	conta
0	30	blue-collar	married	basic.9y	no	yes	no	cellul
1	39	services	single	high.school	no	no	no	telephor
2	25	services	married	high.school	no	yes	no	telephor
3	38	services	married	basic.9y	no	unknown	unknown	telephor
4	47	admin.	married	university.degree	no	yes	no	cellul
...	...	...	...	...	...	...	...	
4114	30	admin.	married	basic.6y	no	yes	yes	cellul
4115	39	admin.	married	high.school	no	yes	no	telephor
4116	27	student	single	high.school	no	no	no	cellul
4117	58	admin.	married	high.school	no	no	no	cellul
4118	34	management	single	high.school	no	yes	no	cellul

4119 rows × 21 columns

In [140…
```
mean=bankdata["nr.employed"].mean()
```

In [141…
```
std=bankdata["nr.employed"].std()
```

In [142…
```
nr=bankdata["nr.employed"]-mean
```

In [143…
```
bankdata["nr.employed_z"]=nr/std
```

In [144…
```
bankdata
```

Out[144…

	age	job	marital	education	default	housing	loan	conta
0	30	blue-collar	married	basic.9y	no	yes	no	cellul
1	39	services	single	high.school	no	no	no	telephor
2	25	services	married	high.school	no	yes	no	telephor
3	38	services	married	basic.9y	no	unknown	unknown	telephor
4	47	admin.	married	university.degree	no	yes	no	cellul
...	...	...	...	...	...	...	...	
4114	30	admin.	married	basic.6y	no	yes	yes	cellul
4115	39	admin.	married	high.school	no	yes	no	telephor
4116	27	student	single	high.school	no	no	no	cellul
4117	58	admin.	married	high.school	no	no	no	cellul
4118	34	management	single	high.school	no	yes	no	cellul

4119 rows × 22 columns

TASK 2

COMPARE TWO nr.employed COLUMNS

GET THE MIN AND MAX VALUES FROM EACH AND THEY SHOULD MATCH

IDXMAX,IDXMIN

In [146…
```
MAXX=bankdata["nr.employed"].idxmax(),bankdata["nr.employed_z"].idxmax()
MINN=bankdata["nr.employed"].idxmin(),bankdata["nr.employed_z"].idxmin()
```

In [147…
```
MAXX
```

Out[147… (2, 2)

In [148…
```
MINN
```

Out[148… (5, 5)

In [149…
```
MAXvalue=bankdata["nr.employed"].max(),bankdata["nr.employed_z"].max()
ta["nr.employed_z"].min()
```

In [150…
```
MAXvalue
```

Out[150… (5228.1, 0.8364335407764235)

In [151…
```
MINvalue
```

Out[151… (4963.6, -2.754003912008983)

standard scalar

- Standard scalar is same as z score by using package
- We use SCIKIT libarary preprocessing method

StandardScalar

- StandardScalar same as Z-score but by using pacakge
- It is under sklearn package
- In the sklearn we have preprocessing
- Read the package
- Save the package
- Apply fit transform
- Compare 3 coumns
  - One is original
  - Manually we did z-score
  - Column with package

In [153…
```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
bankdata['nr.employed_ss']=ss.fit_transform(bankdata[['nr.employed']])
```

- Single square bracket is series
- Double square bracket is Data frame
- Whenever you see the shape error apply double square bracket

Normalization

- Read the data agian
- step-1: take prevailing wage column : **visa_df['prevailing_wage']**
- step-2: calculate the min value prevailing wage:min= **visa_df['prevailing_wage'].min**
- step-3: calculate the min value prevailing wage:max= **visa_df['prevailing_wage'].max**
- step-4: Calculate the Nr : step-1 - step-2:Nr= **visa_df['prevailing_wage']-min**
- step-5: DR= step-3-step-2
- step-6: divide the step4/step5

In [154…
```
x_max = bankdata['nr.employed'].max()
x_min =  bankdata['nr.employed'].min()
Nr =  bankdata['nr.employed'] - x_min
bankdata['nr.employed_min_max'] = Nr/(x_max - x_min)
bankdata[['nr.employed', 'nr.employed_min_max']]
```

Out[154…

	nr.employed	nr.employed_min_max
0	5099.1	0.512287
1	5191.0	0.859735
2	5228.1	1.000000
3	5228.1	1.000000
4	5195.8	0.877883
…	…	…
4114	5228.1	1.000000
4115	5228.1	1.000000
4116	5099.1	0.512287
4117	5228.1	1.000000
4118	5195.8	0.877883

4119 rows × 2 columns

package name: MinMaxScalar

In [155…
```
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
bankdata['nr.employed_min_max_ss'] = mms.fit_transform(bankdata[['nr.employe
```

```
bankdata[['nr.employed','nr.employed_min_max','nr.employed_min_max_ss']]
```

Out[155…

	nr.employed	nr.employed_min_max	nr.employed_min_max_ss
0	5099.1	0.512287	0.512287
1	5191.0	0.859735	0.859735
2	5228.1	1.000000	1.000000
3	5228.1	1.000000	1.000000
4	5195.8	0.877883	0.877883
…	…	…	…
4114	5228.1	1.000000	1.000000
4115	5228.1	1.000000	1.000000
4116	5099.1	0.512287	0.512287
4117	5228.1	1.000000	1.000000
4118	5195.8	0.877883	0.877883

4119 rows × 3 columns

In [ ]: