Data Frame Quick Checks

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

In [2]:
```
bank_df=pd.read_csv(r"C:\Users\Lenovo\Music\EDA Practice\bank.csv",sep=';')
bank_df
```

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact
0	30	unemployed	married	primary	no	1787	no	no	cellular
1	33	services	married	secondary	no	4789	yes	yes	cellular
2	35	management	single	tertiary	no	1350	yes	no	cellular
3	30	management	married	tertiary	no	1476	yes	yes	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown
...	...	...	...	...	...	...	...	...	...
4516	33	services	married	secondary	no	-333	yes	no	cellular
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown
4518	57	technician	married	secondary	no	295	no	no	cellular
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular

4521 rows × 17 columns

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

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	da
0	30	unemployed	married	primary	no	1787	no	no	cellular	
1	33	services	married	secondary	no	4789	yes	yes	cellular	
2	35	management	single	tertiary	no	1350	yes	no	cellular	
3	30	management	married	tertiary	no	1476	yes	yes	unknown	
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	

In [4]:
```
bank_df.tail()
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact
4516	33	services	married	secondary	no	-333	yes	no	cellular
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown
4518	57	technician	married	secondary	no	295	no	no	cellular
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular

In [5]:
```
bank_df.shape
```

Out[5]:
```
(4521, 17)
```

In [6]:
```
bank_df.size
```

Out[6]:
```
76857
```

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

Out[7]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [8]:
```
dtypes=bank_df.dtypes
dtypes
```

Out[8]:
```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

In [9]:
```
dtypes.keys()
```

Out[9]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [10]:
```
dtypes.values
```

Out[10]:
```
array([dtype('int64'), dtype('O'), dtype('O'), dtype('O'), dtype('O'),
       dtype('int64'), dtype('O'), dtype('O'), dtype('O'), dtype('int64'),
       dtype('O'), dtype('int64'), dtype('int64'), dtype('int64'),
       dtype('int64'), dtype('O'), dtype('O')], dtype=object)
```

In [11]:
```
for i,j in dtypes.items():
    if j=='object':
        print(i)
```

```
job
marital
education
default
housing
loan
contact
month
poutcome
y
```

Categorical Column Analysis

In [24]:
```
cat=bank_df.select_dtypes(include='object').columns
```

In [26]:
```
num=bank_df.select_dtypes(exclude='object').columns
```

In [28]:
```
cat
```

Out[28]:
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'poutcome', 'y'],
      dtype='object')
```

In [30]:
```
num
```

Out[30]:
```
Index(['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous'], d
type='object')
```

In [32]:
```
bank_df['education'].unique()
```

Out[32]:
```
array(['primary', 'secondary', 'tertiary', 'unknown'], dtype=object)
```

In [34]:
```
bank_df['education'].nunique()
```

Out[34]:
```
4
```

In [36]:
```
bank_df['education']
con=bank_df['education']=='primary'
bank_df[con]
```

Out[36]:

	age	job	marital	education	default	balance	housing	loan	contac
0	30	unemployed	married	primary	no	1787	no	no	cellula
9	43	services	married	primary	no	-88	yes	yes	cellula
18	25	blue-collar	single	primary	no	-221	yes	no	unknowı
26	55	blue-collar	married	primary	no	627	yes	no	unknowı
36	78	retired	divorced	primary	no	229	no	no	telephon
...	...	...	...	...	...	...	...	...	.
4480	23	blue-collar	married	primary	no	1158	yes	no	cellula
4485	53	blue-collar	married	primary	no	238	yes	no	cellula
4486	37	blue-collar	married	primary	no	378	yes	no	unknowı
4499	45	blue-collar	divorced	primary	no	942	no	no	cellula
4503	60	self-employed	married	primary	no	362	no	yes	cellula

678 rows × 17 columns

Now i want to find how many students from primary,secondary,teritory and unknown by using for loop

In [39]:
```
unique=bank_df['education'].unique()
for i in unique:
    bank_df['education']
    con=bank_df['education']==i
    count=len(bank_df[con])
    print(f"the number of students from {i} is : {count}")
```

```
the number of students from primary is : 678
the number of students from secondary is : 2306
the number of students from tertiary is : 1350
the number of students from unknown is : 187
```

In [44]:
```
import numpy as np
unique=bank_df['education'].unique()
count=[]
for i in unique:
    bank_df['education']
    con=bank_df['education']==i
    count.append(len(bank_df[con]))
count
```

Out[44]:
```
[678, 2306, 1350, 187]
```

frequency table

- Create a dataframe using count and unique
- create two columns 1)Education 2) No.of Students

In [47]:
```
cols=['education','no.of students']
pd.DataFrame(zip(unique,count), columns=cols)
```

Out[47]:

	education	no.of students
0	primary	678
1	secondary	2306
2	tertiary	1350
3	unknown	187

In [49]:
```
bank_df['education'].value_counts()
```

Out[49]:
```
education
secondary    2306
tertiary     1350
primary       678
unknown       187
Name: count, dtype: int64
```

In [51]:
```
keys=bank_df['education'].value_counts().keys()
values=bank_df['education'].value_counts().values
```

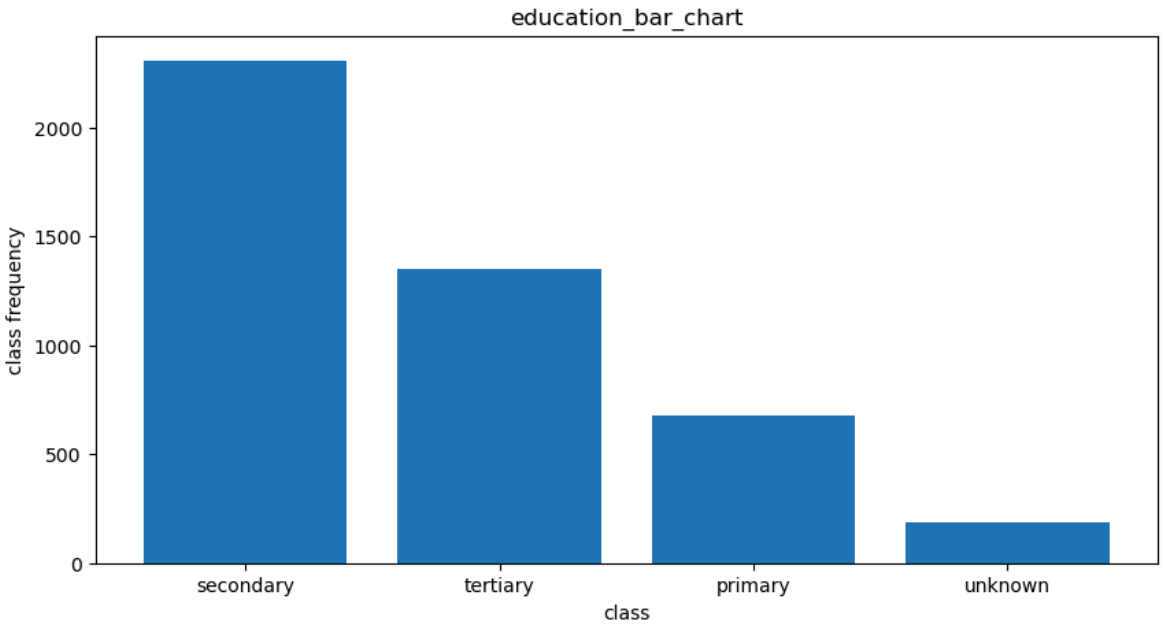
In [53]:
```
cols=['labels','counts']
df=pd.DataFrame(zip(keys,values),columns=cols)
df
```

Out[53]:

	labels	counts
0	secondary	2306
1	tertiary	1350
2	primary	678
3	unknown	187

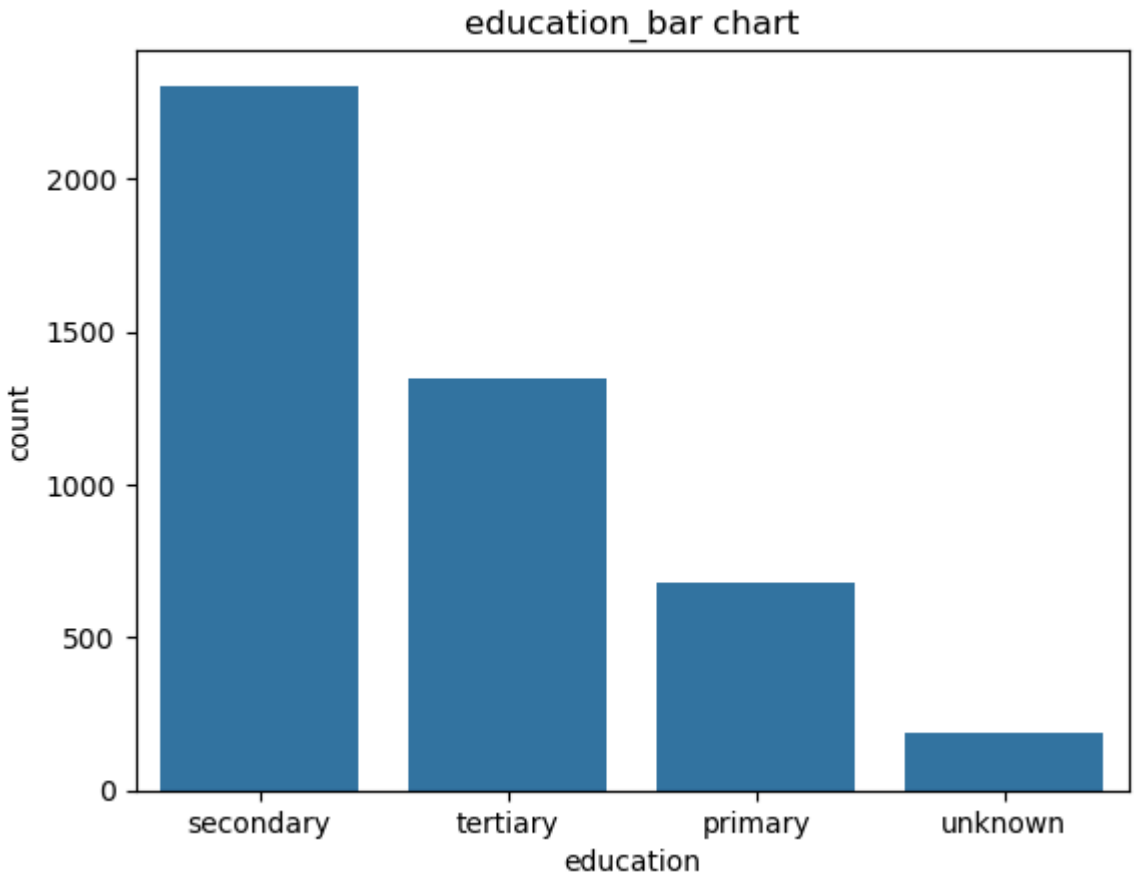
Bar Charts

In [60]:
```
keys=bank_df['education'].value_counts().keys()
values=bank_df['education'].value_counts().values
plt.figure(figsize=(10,5))
plt.bar(keys,values)
plt.xlabel('class')
plt.ylabel('class frequency')
plt.title('education_bar_chart')
plt.savefig('education_bar_chart.jpg')
plt.savefig('education_bar_chart.png')
plt.show()
```



we can draw the bar plot using seaborn

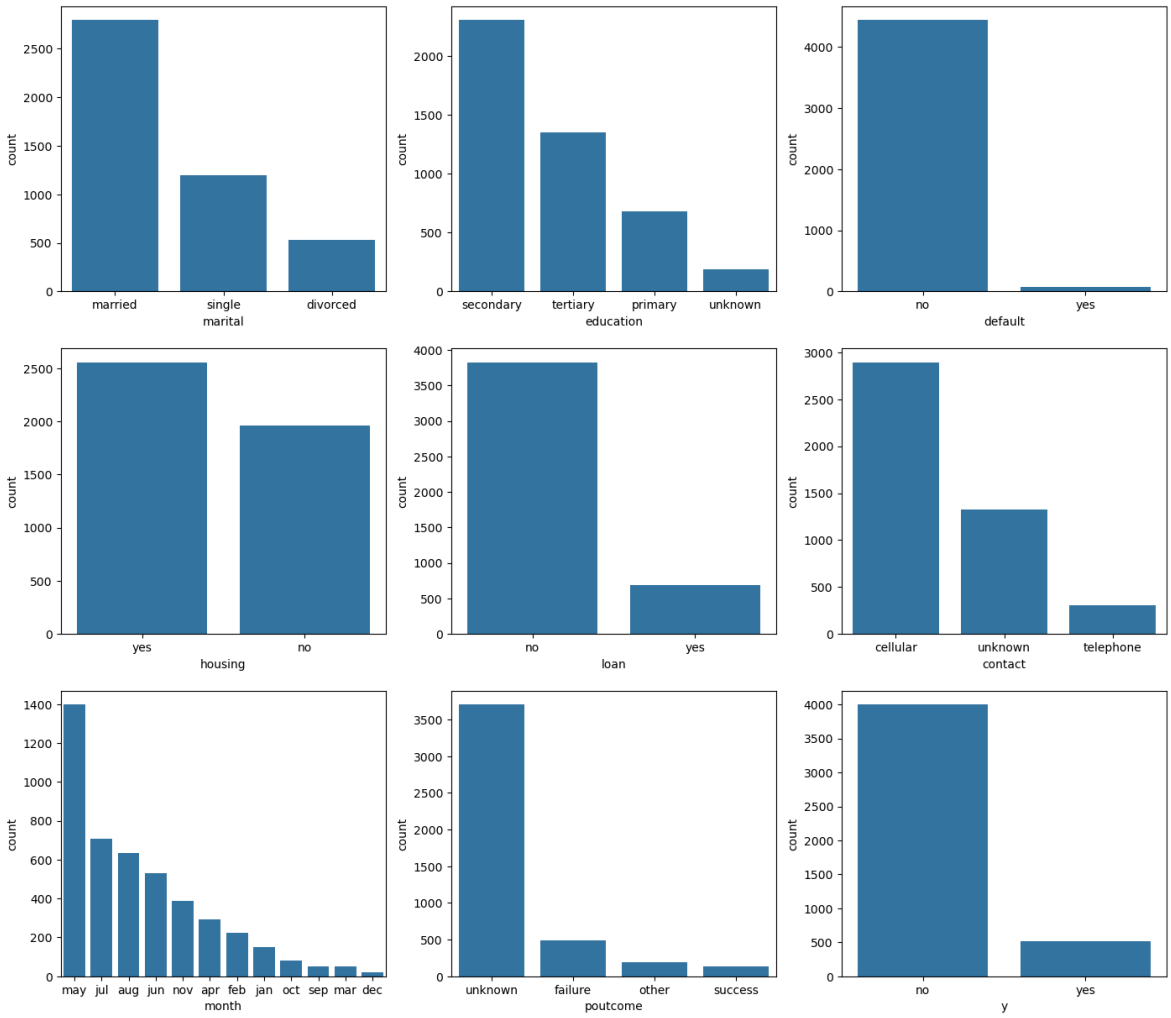
In [63]:
```
keys=bank_df['education'].value_counts().keys()
sns.countplot(data=bank_df,x='education',order=keys)
plt.title('education_bar chart')
plt.savefig('education_bar chart.jpg')
```



Subplots

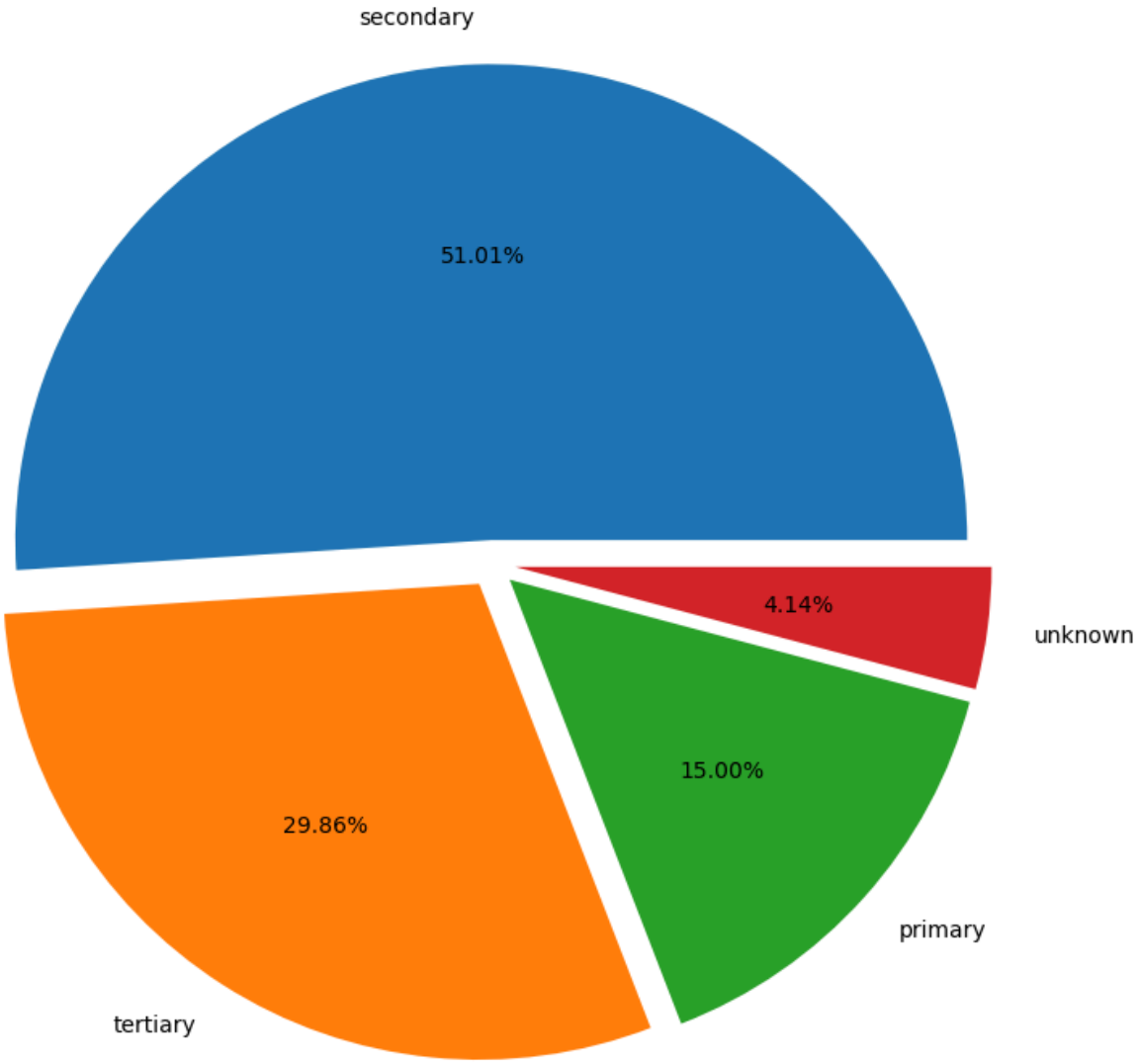
In [66]:
```
plt.figure(figsize=(17,15))
for i in range(1,10):
    keys=bank_df[cat[i]].value_counts().keys()
```

```
    plt.subplot(3,3,i)
    sns.countplot(data=bank_df,x=cat[i],order=keys)
```



Pie Charts

In [69]:
```
keys=bank_df['education'].value_counts().keys()
values=bank_df['education'].value_counts().values
plt.pie(values,labels=keys,autopct='%0.2f%%',radius=2,explode=[0.1,0.1,0.1,0.1])
plt.show()
```



Numerical Column Analysis

In [72]:
```
bal_data=bank_df['balance']
bal_count=len(bank_df['balance'])
bal_min=round(bank_df['balance'].min(),2)
bal_max=round(bank_df['balance'].max(),2)
bal_std=round(bank_df['balance'].std(),2)
bal_mean=round(bank_df['balance'].mean(),2)
bal_median=round(bank_df['balance'].median(),2)

print(f"the total count is :{bal_count}")
print(f"the min balance is :{bal_min}")
print(f"the max balance is : {bal_max}")
print(f"the std balance is : {bal_std}")
print(f"the mean balance is : {bal_mean}")
print(f"the median balance is : {bal_median}")
```

```
the total count is :4521
the min balance is :-3313
the max balance is : 71188
the std balance is : 3009.64
the mean balance is : 1422.66
the median balance is : 444.0
```

In [74]:
```
bal_data=bank_df['balance']
bal_count=len(bank_df['balance'])
bal_min=round(bank_df['balance'].min(),2)
```

```
bal_max=round(bank_df['balance'].max(),2)
bal_std=round(bank_df['balance'].std(),2)
bal_mean=round(bank_df['balance'].mean(),2)
bal_median=round(bank_df['balance'].median(),2)

idx=['count','min','max','std','mean','median']
data=[bal_count,bal_min,bal_max,bal_std,bal_mean,bal_median]
pd.DataFrame(data,index=idx,columns=['balance'])
```

Out[74]:

	balance
count	4521.00
min	-3313.00
max	71188.00
std	3009.64
mean	1422.66
median	444.00

Percentile & Quantile

- **we need to find -25% , 50%, 75% or by using percentile and quantile**

In [77]:
```
bal_data=bank_df['balance']
bal_25p=np.percentile(bal_data,25)
con=bal_data<bal_25p
len(bal_data[con])
```

Out[77]: 1129

In [79]:
```
bal_data=bank_df['balance']
bal_50p=np.percentile(bal_data,50)
con=bal_data<bal_50p
len(bal_data[con])
```

Out[79]: 2259

In [81]:
```
bal_data=bank_df['balance']
bal_75p=np.percentile(bal_data,75)
con=bal_data<bal_75p
len(bal_data[con])
```

Out[81]: 3390

In [83]:
```
bal_data=bank_df['balance']
bal_count=len(bank_df['balance'])
bal_min=round(bank_df['balance'].min(),2)
bal_max=round(bank_df['balance'].max(),2)
bal_std=round(bank_df['balance'].std(),2)
bal_mean=round(bank_df['balance'].mean(),2)
bal_median=round(bank_df['balance'].median(),2)
bal_25p=np.percentile(bal_data,25)
bal_50p=np.percentile(bal_data,50)
bal_75p=np.percentile(bal_data,75)
```

```
idx=['count','min','max','std','mean','median','25%','50%','75%']
cols=['balance']
data=[bal_count,bal_min,bal_max,bal_std,bal_mean,bal_median,bal_25p,bal_50p,bal_
pd.DataFrame(data,index=idx,columns=cols)
```

Out[83]:

	balance
count	4521.00
min	-3313.00
max	71188.00
std	3009.64
mean	1422.66
median	444.00
25%	69.00
50%	444.00
75%	1480.00

In [85]:
```
bank_df.describe()
```

Out[85]:

	age	balance	day	duration	campaign	pdays
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000

In [87]:
```
mean=bank_df['balance'].mean()
std=bank_df['balance'].std()
lb=mean-1*std

mean=bank_df['balance'].mean()
std=bank_df['balance'].std()
ub=mean+1*std

lb,ub

con1=bank_df['balance']>lb
con2=bank_df['balance']<ub
con3=con1&con2
len(bank_df[con3])
```

Out[87]: 4143

In [89]:
```
mean=bank_df['balance'].mean()
std=bank_df['balance'].std()
lb=mean-2*std

mean=bank_df['balance'].mean()
std=bank_df['balance'].std()
ub=mean+2*std

lb,ub

con1=bank_df['balance']>lb
con2=bank_df['balance']<ub
con3=con1&con2
len(bank_df[con3])
```

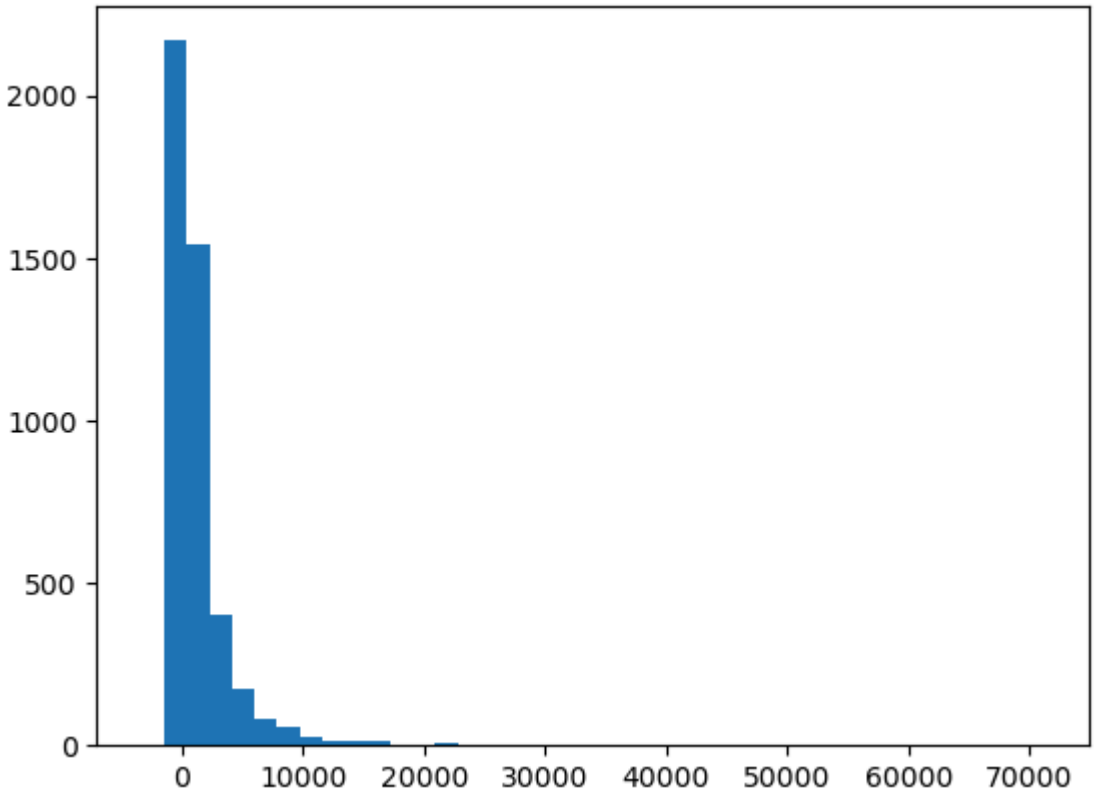
Out[89]: 4355

In [91]:
```
mean=bank_df['balance'].mean()
std=bank_df['balance'].std()
lb=mean-3*std

mean=bank_df['balance'].mean()
std=bank_df['balance'].std()
ub=mean+3*std

lb,ub

con1=bank_df['balance']>lb
con2=bank_df['balance']<ub
con3=con1&con2
len(bank_df[con3]), 99.7*4521/100
```

Out[91]: (4433, 4507.437)

Histogram

In [94]:
```
count,intervals,n=plt.hist(bank_df['balance'],bins=40)
```



In [98]:
```
count
```

Out[98]:
```
array([4.000e+00, 2.168e+03, 1.540e+03, 3.990e+02, 1.760e+02, 8.300e+01,
       5.600e+01, 2.500e+01, 1.600e+01, 1.600e+01, 1.200e+01, 3.000e+00,
       4.000e+00, 7.000e+00, 2.000e+00, 4.000e+00, 4.000e+00, 0.000e+00,
       0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
       1.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
       0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
       0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00])
```

In [100…
```
intervals
```

Out[100…
```
array([-3313.   , -1450.475,   412.05 ,  2274.575,  4137.1  ,  5999.625,
        7862.15 ,  9724.675, 11587.2  , 13449.725, 15312.25 , 17174.775,
       19037.3  , 20899.825, 22762.35 , 24624.875, 26487.4  , 28349.925,
       30212.45 , 32074.975, 33937.5  , 35800.025, 37662.55 , 39525.075,
       41387.6  , 43250.125, 45112.65 , 46975.175, 48837.7  , 50700.225,
       52562.75 , 54425.275, 56287.8  , 58150.325, 60012.85 , 61875.375,
       63737.9  , 65600.425, 67462.95 , 69325.475, 71188.   ])
```

In [102…
```
lb=4.000e+00
ub=2.168e+03
con1=bank_df['balance']>lb
con2=bank_df['balance']<ub
con3=con1&con
c=len(bank_df[con3])
print(f" we have {c} observations between {lb},{ub}")
```

```
we have 2607 observations between 4.0,2168.0
```

Box Plot

In [105…
```
bal_data=bank_df['balance']
q1=round(np.quantile(bal_data,0.25),2)
q3=round(np.quantile(bal_data,0.75),2)
```

```
IQR=q3-q1

lb=q1-1.5*IQR
ub=q3+1.5*IQR

con1=bank_df['balance']<lb
con2=bank_df['balance']>ub
con3=con1|con2
count=len(bank_df[con3])
count
```

Out[105… 506

In [107…
```
outliers_df=bank_df[con3]
outliers_df
```

Out[107…

	age	job	marital	education	default	balance	housing	loan	conta
1	33	services	married	secondary	no	4789	yes	yes	cellul
10	39	services	married	secondary	no	9374	yes	no	unknow
16	56	technician	married	secondary	no	4073	no	no	cellul
25	41	management	married	tertiary	no	5883	no	no	cellul
30	68	retired	divorced	secondary	no	4189	no	no	telephor
...	...	...	...	...	...	...	...	...	
4464	53	services	divorced	secondary	no	4554	no	no	cellul
4473	33	technician	married	secondary	no	4790	yes	no	cellul
4489	45	management	married	tertiary	no	6945	no	no	cellul
4500	38	admin.	married	secondary	no	4196	yes	no	cellul
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknow

506 rows × 17 columns

In [109…
```
bal_data=bank_df['balance']
q1=round(np.quantile(bal_data,0.25),2)
q3=round(np.quantile(bal_data,0.75),2)

IQR=q3-q1

lb=q1-1.5*IQR
ub=q3+1.5*IQR

con1=bank_df['balance']>lb
con2=bank_df['balance']<ub
con3=con1&con2
count=len(bank_df[con3])
count
```

Out[109… 4015

In [111…
```
non_outliers_data=bank_df[con3]
non_outliers_data
```

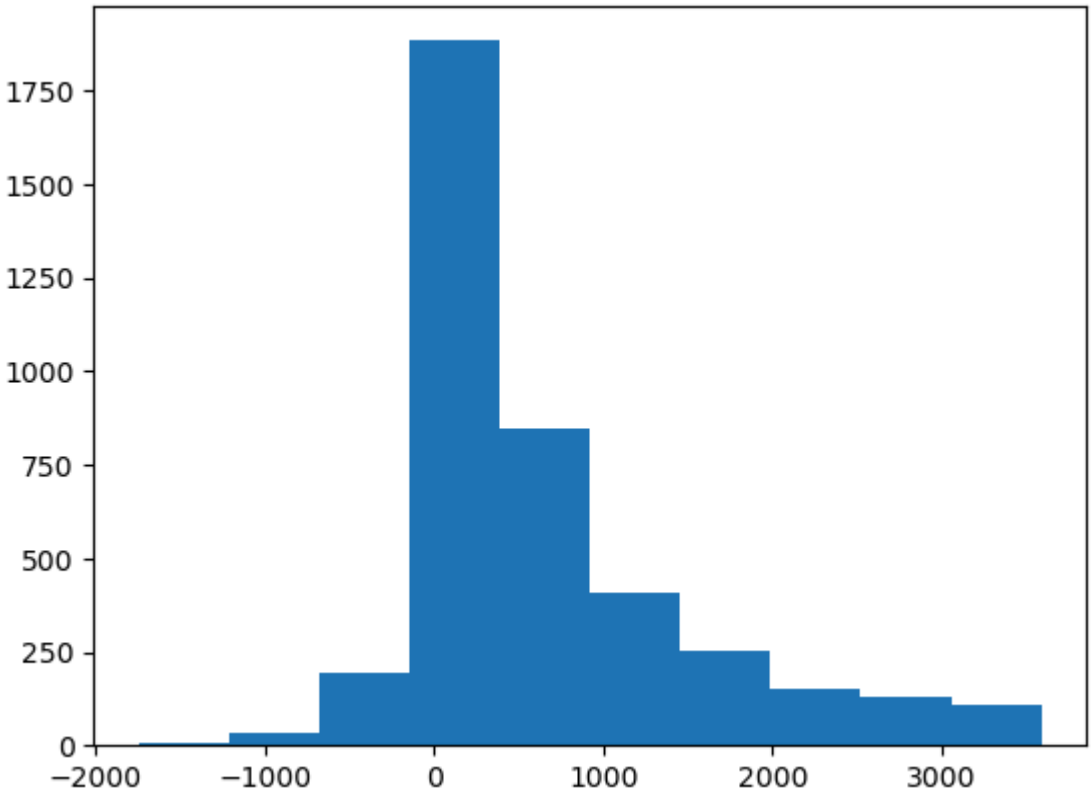
Out[111…

	age	job	marital	education	default	balance	housing	loan	contact
0	30	unemployed	married	primary	no	1787	no	no	cellular
2	35	management	single	tertiary	no	1350	yes	no	cellular
3	30	management	married	tertiary	no	1476	yes	yes	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown
5	35	management	single	tertiary	no	747	no	no	cellular
...	...	...	...	...	...	...	...	...	...
4515	32	services	single	secondary	no	473	yes	no	cellular
4516	33	services	married	secondary	no	-333	yes	no	cellular
4518	57	technician	married	secondary	no	295	no	no	cellular
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular

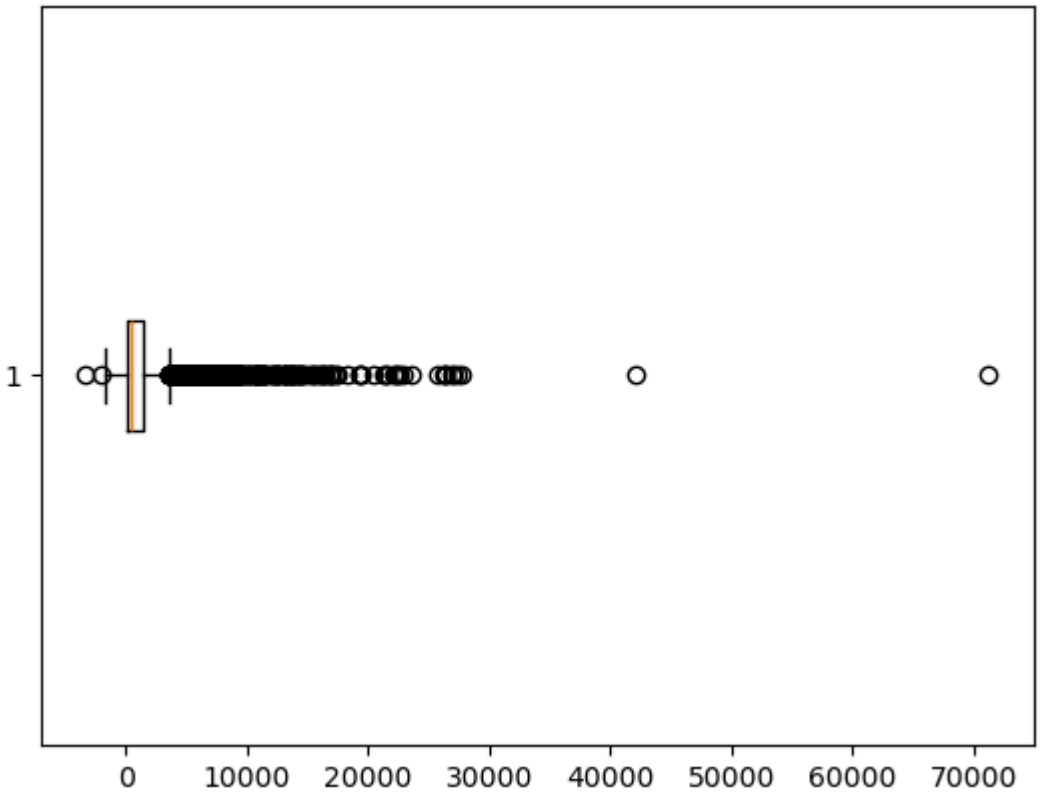
4015 rows × 17 columns

In [113…
```
plt.hist(non_outliers_data['balance'])
```

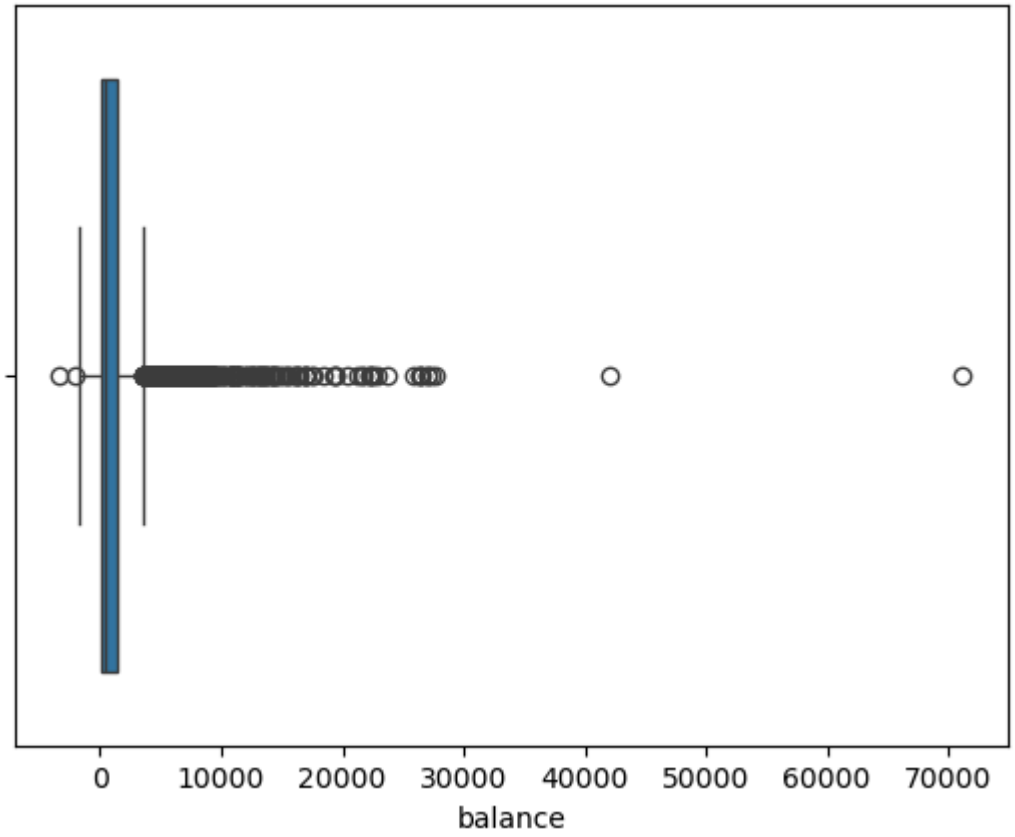
Out[113…
```
(array([   6.,   34.,  192., 1883.,  849.,  410.,  252.,  152.,  130.,
         107.]),
 array([-1746. , -1212.7,  -679.4,  -146.1,   387.2,   920.5,  1453.8,
        1987.1,  2520.4,  3053.7,  3587. ]),
 <BarContainer object of 10 artists>)
```



In [115…
```
plt.boxplot(bank_df['balance'],vert=False)
plt.show()
```

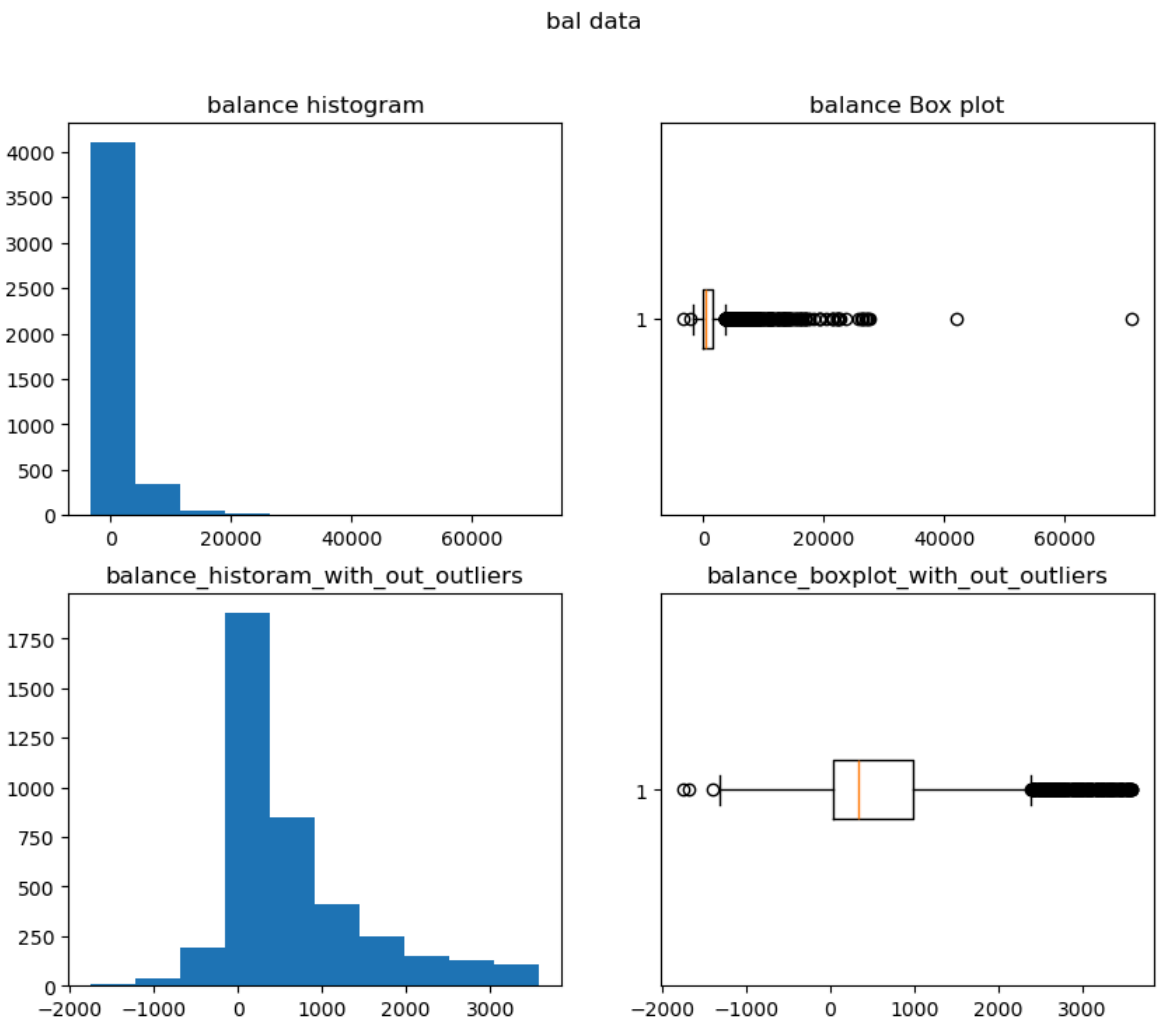


In [117…
```
sns.boxplot(bank_df['balance'],orient='h')
plt.show()
```



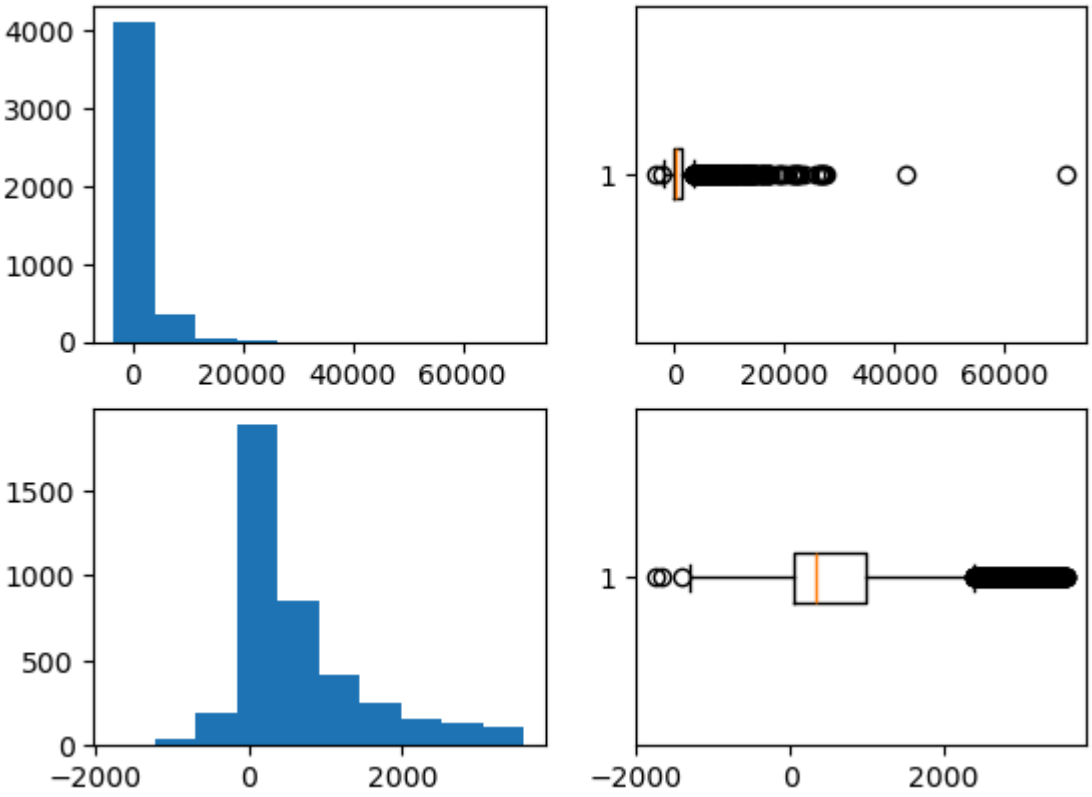
In [119…

```
plt.figure(figsize=(10,8))
plt.suptitle('bal data')
plt.subplot(2,2,1)
plt.hist(bank_df['balance'])
plt.title('balance histogram')
plt.subplot(2,2,2)
plt.boxplot(bank_df['balance'],vert=False)
plt.title('balance Box plot')
plt.subplot(2,2,3)
plt.hist(non_outliers_data['balance'])
plt.title('balance_historam_with_out_outliers')
plt.subplot(2,2,4)
plt.boxplot(non_outliers_data['balance'],vert=False)
plt.title('balance_boxplot_with_out_outliers')
plt.show()
```



In [121…

```
plt.subplot(2,2,1).hist(bank_df['balance'])
plt.subplot(2,2,2).boxplot(bank_df['balance'],vert=False)
plt.subplot(2,2,3).hist(non_outliers_data['balance'])
plt.subplot(2,2,4).boxplot(non_outliers_data['balance'],vert=False)
plt.show()
```



Outiler Analysis

In [124…
```
bal_data=bank_df['balance']
q1=np.quantile(bal_data,0.25)
q3=np.quantile(bal_data,0.75)
IQR=q3-q1
lb=q1-1.5*IQR
ub=q3+1.5*IQR
con1=bank_df['balance']>lb
con2=bank_df['balance']<ub
con3=con1&con2
len(bank_df[con3])
```

Out[124…
```
4015
```

In [126…
```
non_outlier_data=bank_df[con3]
non_outlier_data
```

Out[126...

	age	job	marital	education	default	balance	housing	loan	contact
0	30	unemployed	married	primary	no	1787	no	no	cellular
2	35	management	single	tertiary	no	1350	yes	no	cellular
3	30	management	married	tertiary	no	1476	yes	yes	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown
5	35	management	single	tertiary	no	747	no	no	cellular
...	...	...	...	...	...	...	...	...	...
4515	32	services	single	secondary	no	473	yes	no	cellular
4516	33	services	married	secondary	no	-333	yes	no	cellular
4518	57	technician	married	secondary	no	295	no	no	cellular
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular

4015 rows × 17 columns

how to treat outliers

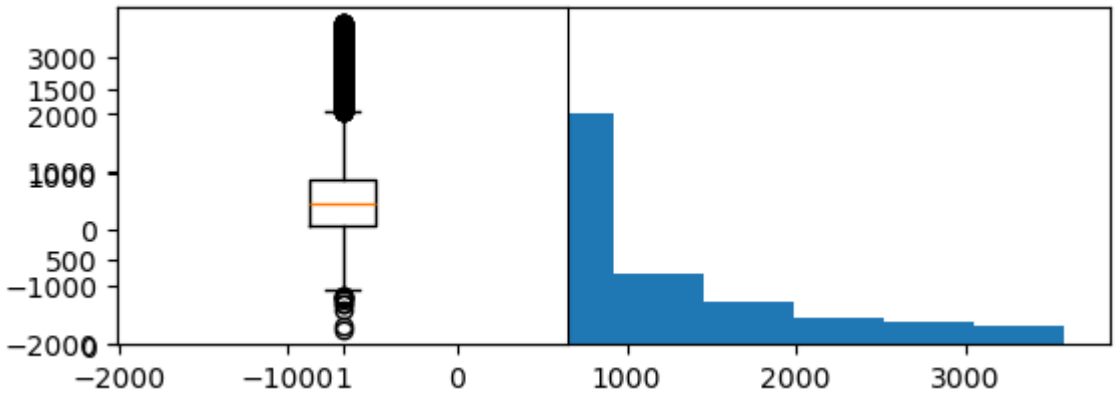
- drop the outliers
- fill with median
- fill with cap values(q1,q3)

In [129...

```
bal_data=bank_df['balance']
q1=round(np.quantile(bal_data,0.25),2)
q3=round(np.quantile(bal_data,0.75),2)
IQR=q3-q1
lb=q1-1.5*IQR
ub=q3+1.5*IQR
median=bal_data.median()
new_data=[]
for i in bal_data:
    if i<lb or i>ub:
        new_data.append(median)
    else:
        new_data.append(i)
bank_df['pbal']=new_data
```

In [131...

```
plt.subplot(2,1,1).hist(bank_df['pbal'])
plt.subplot(2,2,1).boxplot(bank_df['pbal'])
plt.show()
```



np.where

- np.where is a method used to do if-else task in single method
- np.where(con,true,false)

In [135…
```
dict1={'marks':[100,200,300,400],
       'sub':['DS','ML','DL','AI']}
df=pd.DataFrame(dict1)
```

In [137…
```
df
```

Out[137…

	marks	sub
0	100	DS
1	200	ML
2	300	DL
3	400	AI

In [139…
```
con=df['marks']>200
true=1
false=df['marks']
df['new']=np.where(con,true,false)
df
```

Out[139…

	marks	sub	new
0	100	DS	100
1	200	ML	200
2	300	DL	1
3	400	AI	1

Replacing the outliers using np.where

In [144…
```
bal_data=bank_df['balance']
q1=round(np.quantile(bal_data,0.25),2)
q3=round(np.quantile(bal_data,0.75),2)
IQR=q3-q1
```

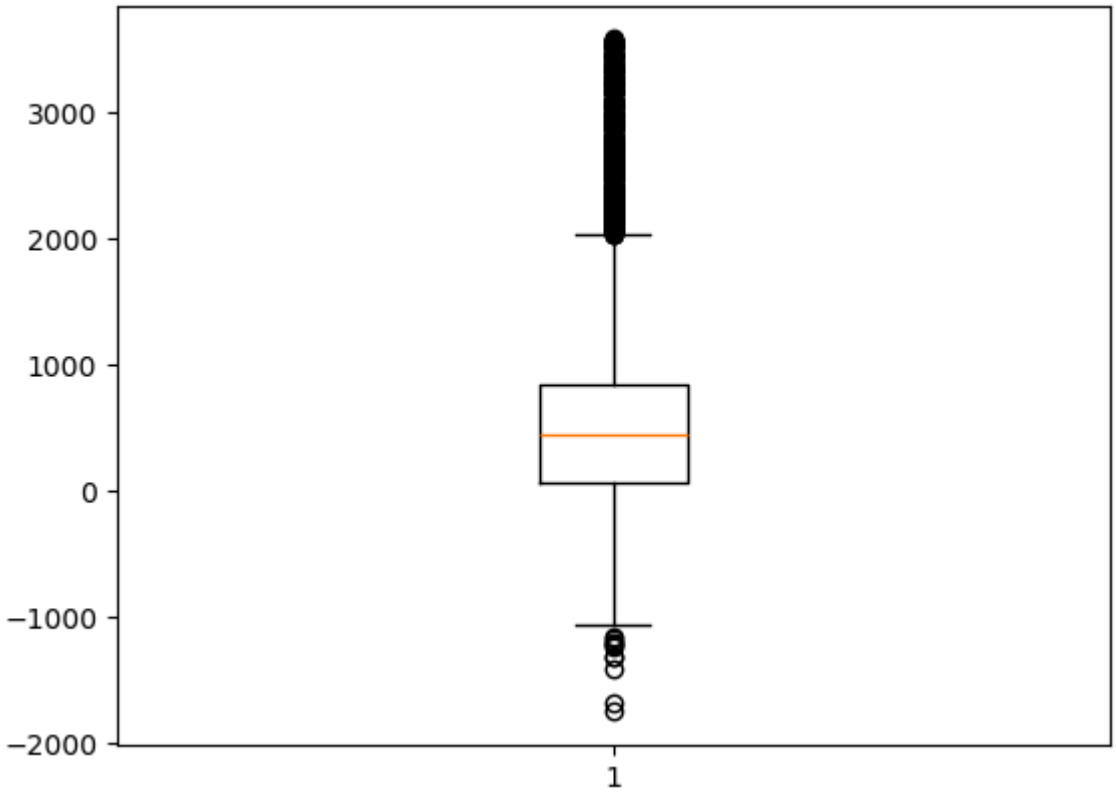
```
lb=q1-1.5*IQR
ub=q3+1.5*IQR
median=bal_data.median()
con=(bank_df['balance']<lb)|(bank_df['balance']>ub)
true=median
false=bank_df['balance']
bank_df['pbal_1']=np.where(con,true,false)
```

In [146…

```
plt.boxplot(bank_df['pbal_1'])
```

Out[146…

```
{'whiskers': [<matplotlib.lines.Line2D at 0x24d282bcf50>,
  <matplotlib.lines.Line2D at 0x24d282bed50>],
 'caps': [<matplotlib.lines.Line2D at 0x24d282bc320>,
  <matplotlib.lines.Line2D at 0x24d282bc650>],
 'boxes': [<matplotlib.lines.Line2D at 0x24d282bd760>],
 'medians': [<matplotlib.lines.Line2D at 0x24d282be000>],
 'fliers': [<matplotlib.lines.Line2D at 0x24d282beba0>],
 'means': []}
```



Bivariate Analysis

In [149…
```
labels=bank_df['education'].unique()
single,married,divorced=[],[],[]
for i in labels:
    con1=bank_df['education']==i
    con2=bank_df['marital']=='single'
    con3=bank_df['marital']=='married'
    con4=bank_df['marital']=='divorced'
    single_con=con1&con2
    marr_con=con1&con3
    div_con=con1&con4
    single.append(len(bank_df[single_con]))
    married.append(len(bank_df[marr_con]))
    divorced.append(len(bank_df[div_con]))
single,married,divorced
```

Out[149…
```
([73, 609, 468, 46], [526, 1427, 727, 117], [79, 270, 155, 24])
```

In [151…
```
pd.DataFrame(zip(single,married,divorced),index=labels,columns=['single','marrie
```

Out[151…

	single	married	divorced
primary	73	526	79
secondary	609	1427	270
tertiary	468	727	155
unknown	46	117	24

Cross Tab

In [154…
```
col1=bank_df['education']
col2=bank_df['marital']
pd.crosstab(col1,col2)
```

Out[154…

marital	divorced	married	single
education			
primary	79	526	73
secondary	270	1427	609
tertiary	155	727	468
unknown	24	117	46

In [156…
```
pd.crosstab(col2,col1)
```

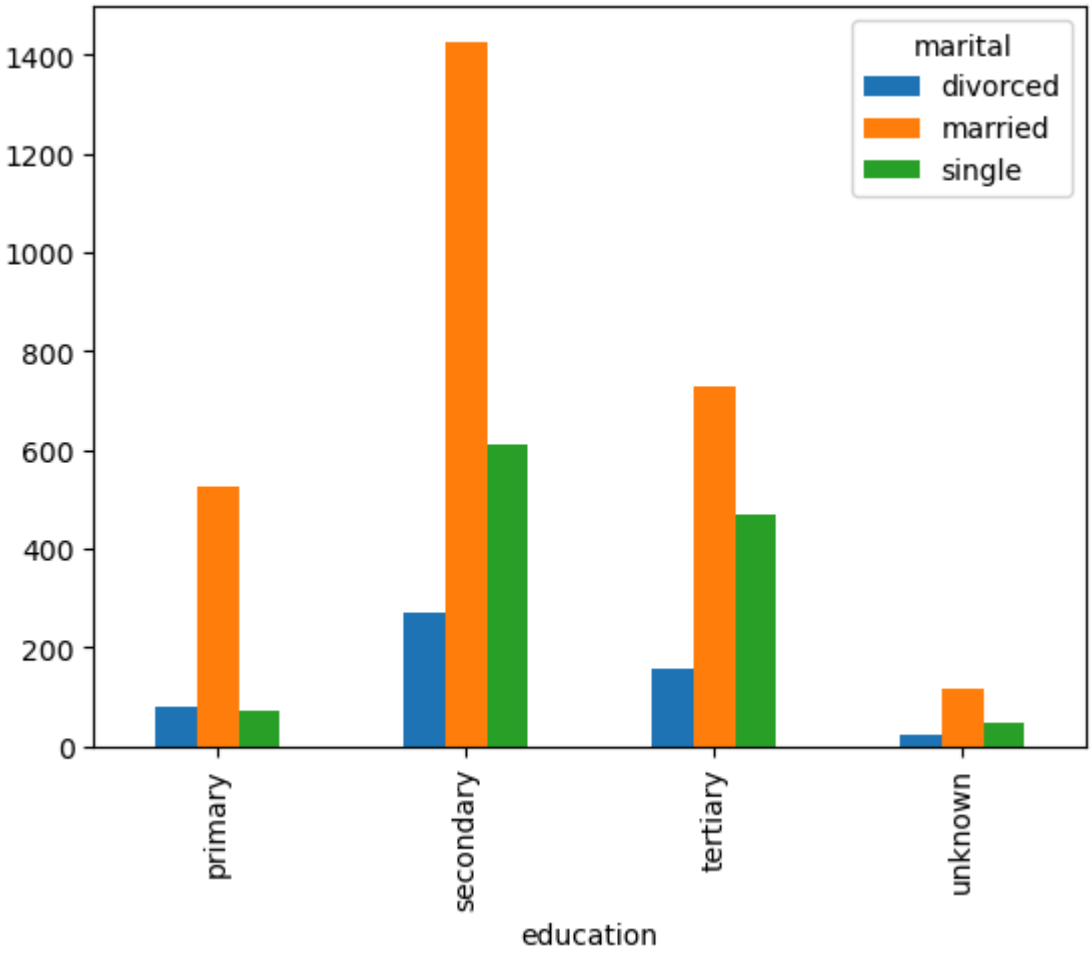
Out[156…

education	primary	secondary	tertiary	unknown
marital				
divorced	79	270	155	24
married	526	1427	727	117
single	73	609	468	46

In [158…
```
col1=bank_df['education']
col2=bank_df['marital']
r1=pd.crosstab(col1,col2)
r2=pd.crosstab(col2,col1)
```

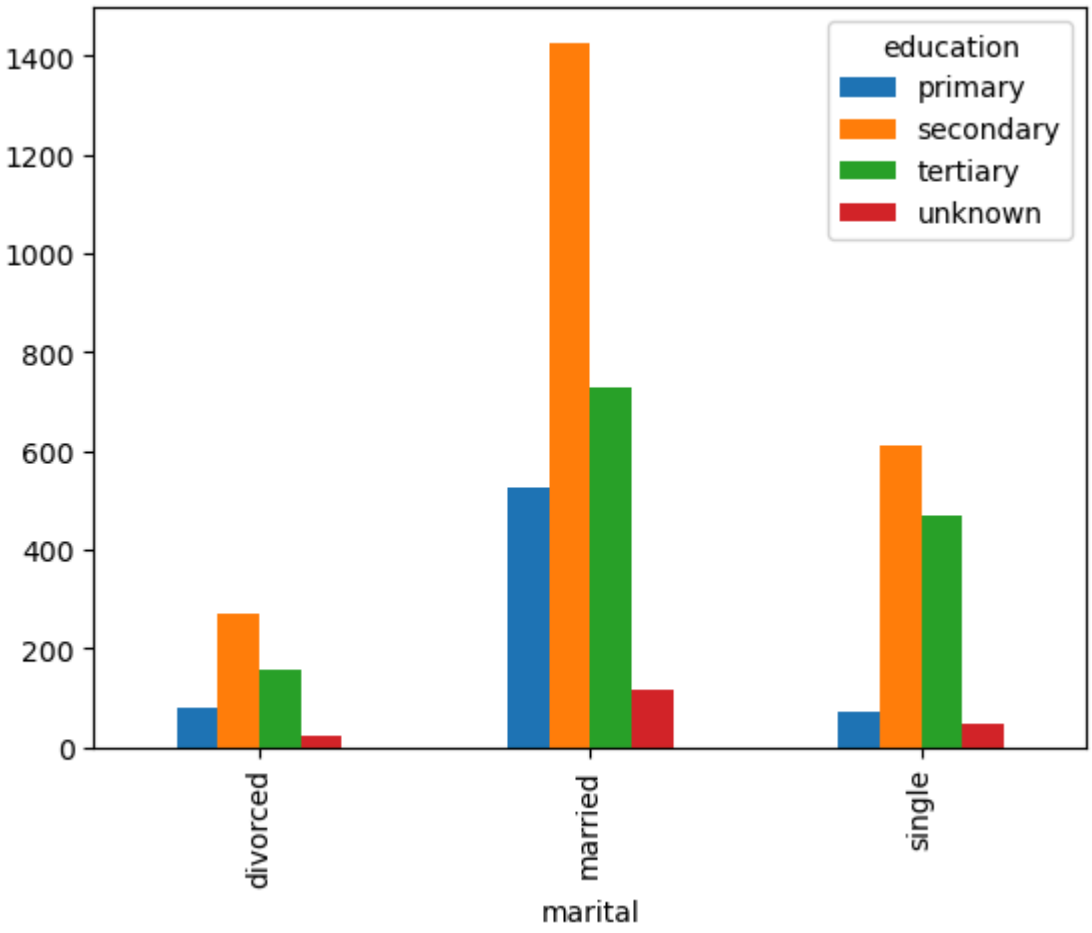
In [160…
```
r1.plot(kind='bar')
```

Out[160… <Axes: xlabel='education'>



In [162…
```
r2.plot(kind='bar')
```

Out[162… <Axes: xlabel='marital'>



In [164…
```
col1=bank_df['job']
col2=bank_df['marital']
col3=bank_df['education']
```

In [166…
```
r1=pd.crosstab(col1,[col2,col3])
r1
```

Out[166…

marital	divorced				r			
education	primary	secondary	tertiary	unknown	primary	secondary	tertiary	un
job								
admin.	3	58	3	5	12	218	27	
blue-collar	37	37	0	5	284	377	3	
entrepreneur	2	5	8	1	22	45	55	
housemaid	6	5	2	0	48	20	12	
management	2	11	103	3	33	73	434	
retired	13	17	9	4	64	83	20	
self-employed	2	4	9	0	12	59	53	
services	5	53	1	3	15	204	11	
student	0	0	0	0	0	4	3	
technician	2	67	17	3	12	299	88	
unemployed	7	13	2	0	17	38	19	
unknown	0	0	1	0	7	7	2	

In [168…

```
r2=pd.crosstab(col2,[col1,col3])
r2
```

Out[168…

job	admin.				blue-cc			
education	primary	secondary	tertiary	unknown	primary	secondary	tertiary	unkn
marital								
divorced	3	58	3	5	37	37	0	
married	12	218	27	9	284	377	3	
single	2	117	21	3	48	110	9	

3 rows × 48 columns

In [170…

```
r3=pd.crosstab(col3,[col1,col2])
r3
```

Out[170…

job	admin.			blue-collar			entrepren		
marital	divorced	married	single	divorced	married	single	divorced	married	sir
education									
primary	3	12	2	37	284	48	2	22	
secondary	58	218	117	37	377	110	5	45	
tertiary	3	27	21	0	3	9	8	55	
unknown	5	9	3	5	29	7	1	10	

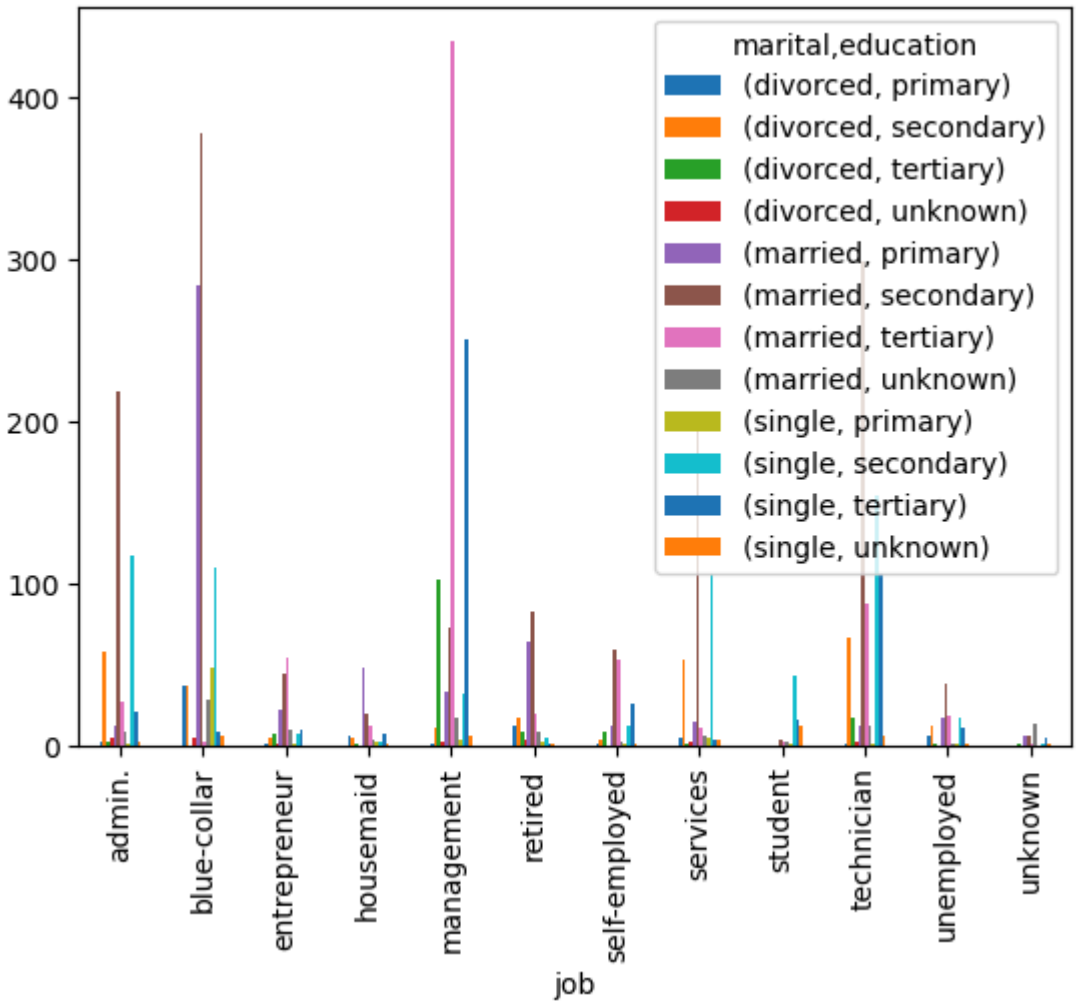
4 rows × 35 columns

In [172…

```
r1.plot(kind='bar')
```

Out[172…

```
<Axes: xlabel='job'>
```

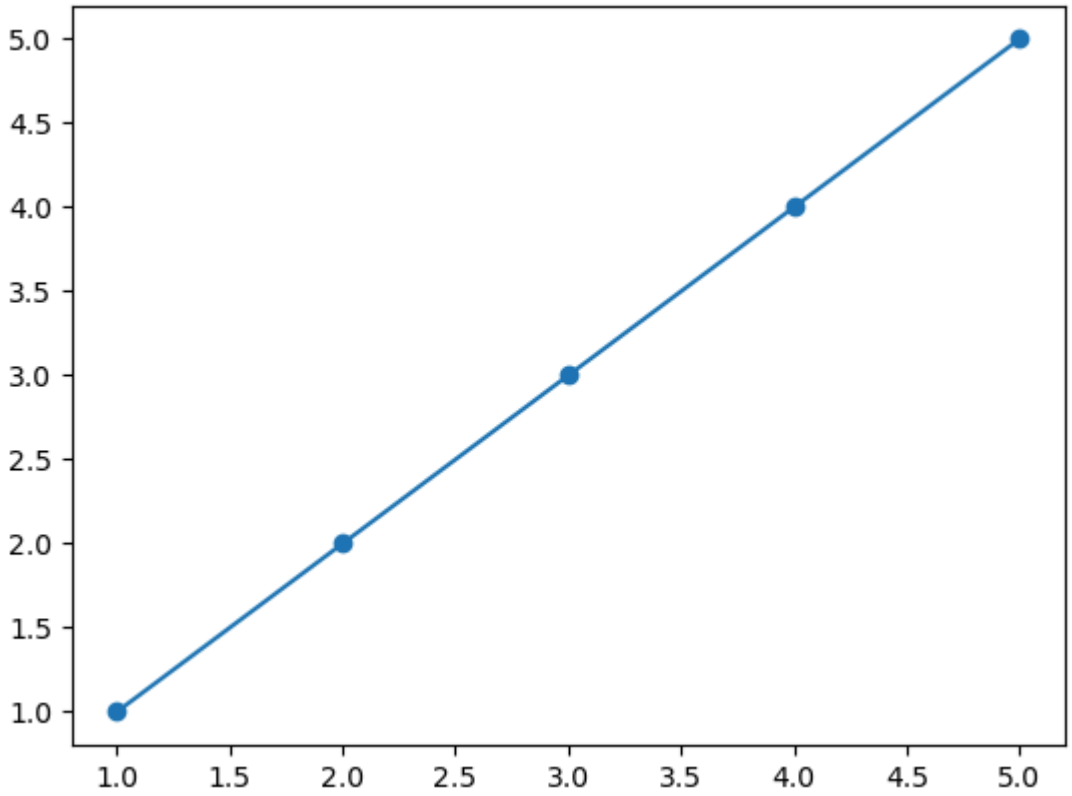


Numerical vs Numerical

- scatter plots are used to plot between numerical vs numerical
- it is under matplot
- plt.scatter()

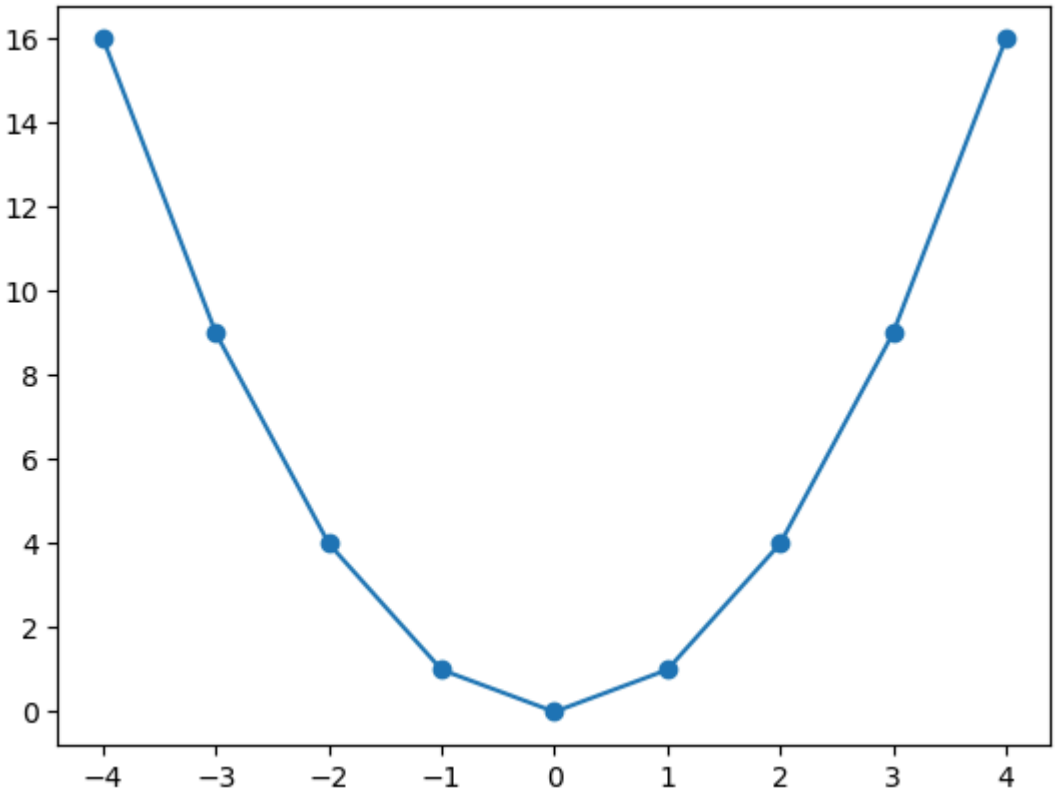
In [175…
```
x=[1,2,3,4,5]
y=[1,2,3,4,5]
plt.scatter(x,y)
plt.plot(x,y)
```

Out[175… [<matplotlib.lines.Line2D at 0x24d279f8a70>]



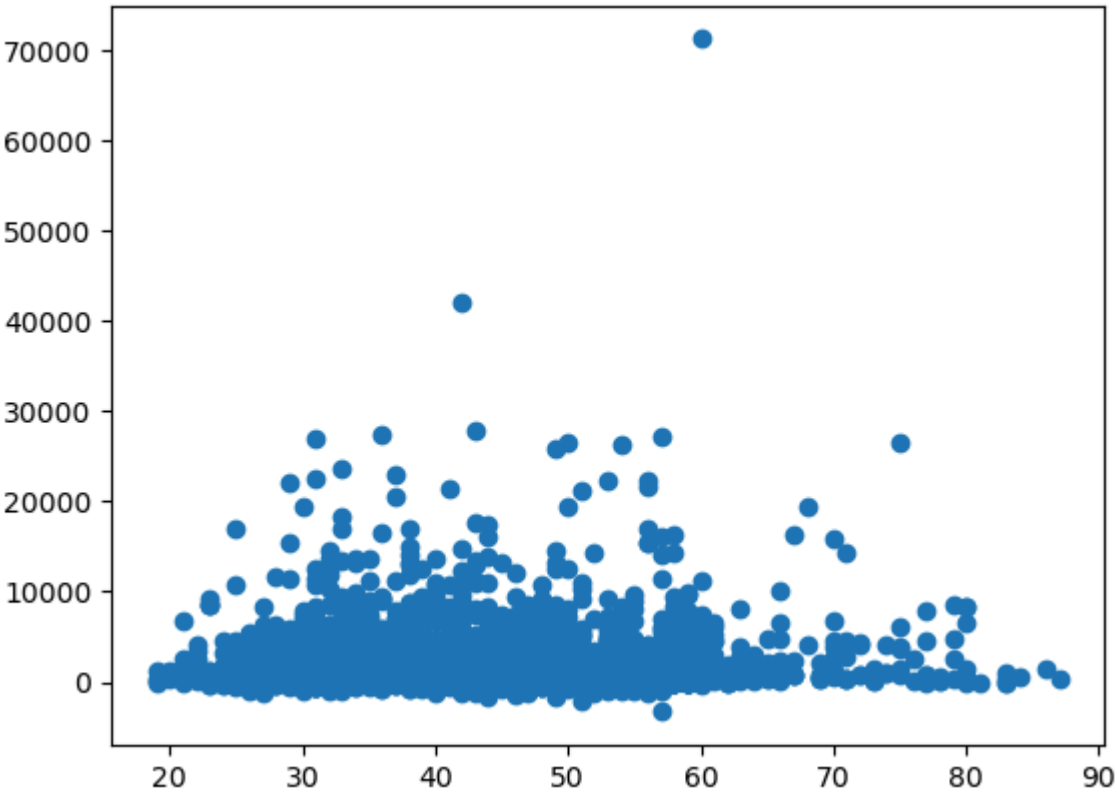
In [177…
```
x=[i for i in range(-4,5)]
y=[i*i for i in range(-4,5)]
plt.scatter(x,y)
plt.plot(x,y)
```

Out[177… [<matplotlib.lines.Line2D at 0x24d2a705010>]



In [179…
```
col1=bank_df['age']
col2=bank_df['balance']
col3=bank_df['day']
plt.scatter(col1,col2)
```

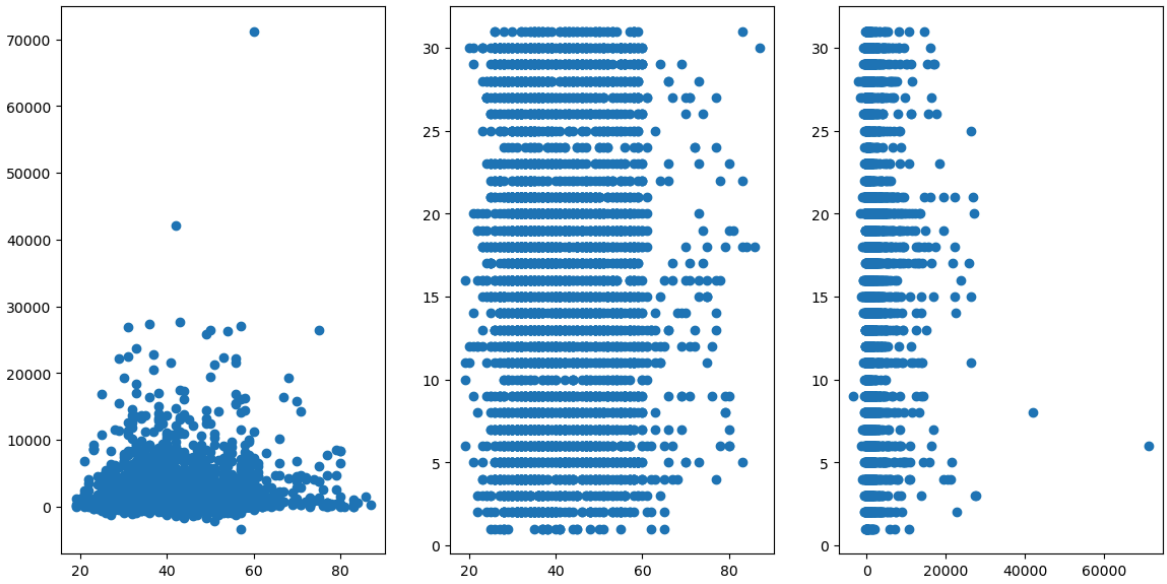
Out[179… `<matplotlib.collections.PathCollection at 0x24d282a8ef0>`



In [181…
```
plt.figure(figsize=(14,7))
plt.subplot(1,3,1).scatter(col1,col2)
```

```
plt.subplot(1,3,2).scatter(col1,col3)
plt.subplot(1,3,3).scatter(col2,col3)
```

Out[181… <matplotlib.collections.PathCollection at 0x24d283f7d70>



Correlation

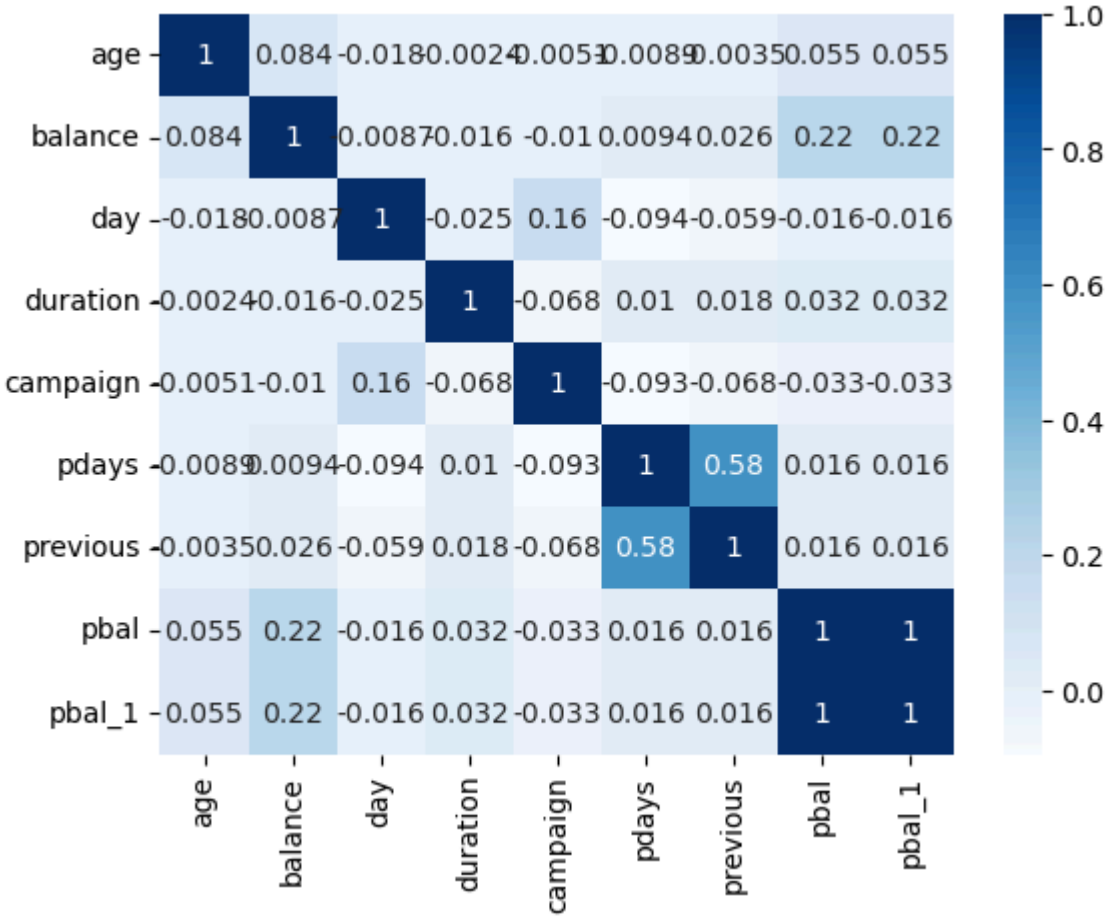
In [184…
```
bank_df.corr(numeric_only=True)
```

Out[184…

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.083820	-0.017853	-0.002367	-0.005148	-0.008894	-0.003511
balance	0.083820	1.000000	-0.008677	-0.015950	-0.009976	0.009437	0.026196
day	-0.017853	-0.008677	1.000000	-0.024629	0.160706	-0.094352	-0.059114
duration	-0.002367	-0.015950	-0.024629	1.000000	-0.068382	0.010380	0.018080
campaign	-0.005148	-0.009976	0.160706	-0.068382	1.000000	-0.093137	-0.067833
pdays	-0.008894	0.009437	-0.094352	0.010380	-0.093137	1.000000	0.577562
previous	-0.003511	0.026196	-0.059114	0.018080	-0.067833	0.577562	1.000000
pbal	0.055307	0.215264	-0.015530	0.031843	-0.033043	0.015697	0.015868
pbal_1	0.055307	0.215264	-0.015530	0.031843	-0.033043	0.015697	0.015868

In [186…
```
corr=bank_df.corr(numeric_only=True)
sns.heatmap(corr,annot=True,cmap='Blues')
```

Out[186… <Axes: >



Encoding

- The encoding is required to convert categorical columns to numerical columns
- we have some encoding techniques to convert categorical to numerical columns
- map
- label encoder
- np.where
- one hot encoder

map

In [190…
```
bank_df=pd.read_csv(r"C:\Users\Lenovo\Music\EDA Practice\bank.csv",sep=';')
for col in cat:
    d={}
    labels=bank_df[col].unique()
    for i in range (len(labels)):
        d[labels[i]]=i
    bank_df[col]=bank_df[col].map(d)
```

In [193…
```
bank_df
```

Out[193…

	age	job	marital	education	default	balance	housing	loan	contact	day	mo
0	30	0	0	0	0	1787	0	0	0	19	
1	33	1	0	1	0	4789	1	1	0	11	
2	35	2	1	2	0	1350	1	0	0	16	
3	30	2	0	2	0	1476	1	1	1	3	
4	59	3	0	1	0	0	1	0	1	5	
...	...	...	...	...	...	...	...	...	...	...	
4516	33	1	0	1	0	-333	1	0	0	30	
4517	57	4	0	2	1	-3313	1	1	1	9	
4518	57	5	0	1	0	295	0	0	0	19	
4519	28	3	0	1	0	1137	0	0	0	6	
4520	44	6	1	2	0	1136	1	1	0	3	

4521 rows × 17 columns

Label Encoder

- it is also used to convert the cat to num
- it is a sklearn package

In [196…
```
from sklearn.preprocessing import LabelEncoder
```

In [198…
```
le=LabelEncoder()
```

In [200…
```
bank_df['y']=le.fit_transform(bank_df['y'])
```

In [202…
```
bank_df
```

Out[202…

	age	job	marital	education	default	balance	housing	loan	contact	day	mo
0	30	0	0	0	0	1787	0	0	0	19	
1	33	1	0	1	0	4789	1	1	0	11	
2	35	2	1	2	0	1350	1	0	0	16	
3	30	2	0	2	0	1476	1	1	1	3	
4	59	3	0	1	0	0	1	0	1	5	
…	…	…	…	…	…	…	…	…	…	…	
4516	33	1	0	1	0	-333	1	0	0	30	
4517	57	4	0	2	1	-3313	1	1	1	9	
4518	57	5	0	1	0	295	0	0	0	19	
4519	28	3	0	1	0	1137	0	0	0	6	
4520	44	6	1	2	0	1136	1	1	0	3	

4521 rows × 17 columns

to get all the cat columns to num we use the for loop

In [205…

```
bank_df=pd.read_csv(r"C:\Users\Lenovo\Music\EDA Practice\bank.csv",sep=';')
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in cat:
    bank_df[i]=le.fit_transform(bank_df[i])

bank_df
```

Out[205…

	age	job	marital	education	default	balance	housing	loan	contact	day	mo
0	30	10	1	0	0	1787	0	0	0	19	
1	33	7	1	1	0	4789	1	1	0	11	
2	35	4	2	2	0	1350	1	0	0	16	
3	30	4	1	2	0	1476	1	1	2	3	
4	59	1	1	1	0	0	1	0	2	5	
…	…	…	…	…	…	…	…	…	…	…	
4516	33	7	1	1	0	-333	1	0	0	30	
4517	57	6	1	2	1	-3313	1	1	2	9	
4518	57	9	1	1	0	295	0	0	0	19	
4519	28	1	1	1	0	1137	0	0	0	6	
4520	44	2	2	2	0	1136	1	1	0	3	

4521 rows × 17 columns

np.where

- can change the data but using only 2 values

In [210…

```
bank_df=pd.read_csv(r"C:\Users\Lenovo\Music\EDA Practice\bank.csv",sep=';')
con=bank_df['y']=='yes'
true=0
false=1
np.where(con,true,false)
```

Out[210…

```
array([1, 1, 1, ..., 1, 1, 1])
```

One hot encoding

- one hot encoding means one will on and another will off
- on represents 1
- off represents 0

In [213…

```
bank_df=pd.read_csv(r"C:\Users\Lenovo\Music\EDA Practice\bank.csv",sep=';')
pd.get_dummies(bank_df['y'],prefix=['y'],dtype='int')
```

Out[213...

	['y']_no	['y']_yes
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
4516	1	0
4517	1	0
4518	1	0
4519	1	0
4520	1	0

4521 rows × 2 columns

In [215...

```
bank_df=pd.read_csv(r"C:\Users\Lenovo\Music\EDA Practice\bank.csv",sep=';')
pd.get_dummies(bank_df,dtype='int')
```

Out[215...

	age	balance	day	duration	campaign	pdays	previous	job_admin.	job_blue-collar
0	30	1787	19	79	1	-1	0	0	0
1	33	4789	11	220	1	339	4	0	0
2	35	1350	16	185	1	330	1	0	0
3	30	1476	3	199	4	-1	0	0	0
4	59	0	5	226	1	-1	0	0	1
...	...	...	...	...	...	...	...	...	...
4516	33	-333	30	329	5	-1	0	0	0
4517	57	-3313	9	153	1	-1	0	0	0
4518	57	295	19	151	11	-1	0	0	0
4519	28	1137	6	129	4	211	3	0	1
4520	44	1136	3	345	2	249	7	0	0

4521 rows × 53 columns

Scale The Data

Standard Scalar

- z=(x-mean)/sigma

In [219…
```
bal_data=bank_df['balance']
mean=bal_data.mean()
std=bal_data.std()
data=(bal_data-mean)/std
```

In [221…
```
data
```

Out[221…
```
0       0.121058
1       1.118521
2      -0.024142
3       0.017724
4      -0.472701
          ...
4516   -0.583345
4517   -1.573497
4518   -0.374682
4519   -0.094914
4520   -0.095247
Name: balance, Length: 4521, dtype: float64
```

In [223…
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
ss.fit_transform(bank_df[['balance']])
```

Out[223…
```
array([[ 0.12107186],
       [ 1.1186443 ],
       [-0.02414438],
       ...,
       [-0.37472364],
       [-0.09492484],
       [-0.09525714]])
```

In [225…
```
d=bank_df['balance'].values.reshape(-1,1)
ss.fit_transform(d)
```

Out[225…
```
array([[ 0.12107186],
       [ 1.1186443 ],
       [-0.02414438],
       ...,
       [-0.37472364],
       [-0.09492484],
       [-0.09525714]])
```

Missing Value Analysis

In [228…
```
#we fill the missing values using mean median mode
#for numerical columns we use mean and median
#for catergorical columns we use mode
```

In [242…
```
dict2={'Names':['Ramesh','Suresh','Mahesh',np.nan],
       'Age':[20,21,np.nan,22],
       'City':[np.nan,'Hyd','Pune','Berhampur']}

data=pd.DataFrame(dict2)
data
```

Out[242...

	Names	Age	City
0	Ramesh	20.0	NaN
1	Suresh	21.0	Hyd
2	Mahesh	NaN	Pune
3	NaN	22.0	Berhampur

In [244...
```
data.isnull()
```

Out[244...

	Names	Age	City
0	False	False	True
1	False	False	False
2	False	True	False
3	True	False	False

In [246...
```
data.isnull().sum()
```

Out[246...
```
Names    1
Age      1
City     1
dtype: int64
```

In [250...
```
data['Names'].fillna('ramu')
```

Out[250...
```
0    Ramesh
1    Suresh
2    Mahesh
3      ramu
Name: Names, dtype: object
```

In [252...
```
data
```

Out[252...

	Names	Age	City
0	Ramesh	20.0	NaN
1	Suresh	21.0	Hyd
2	Mahesh	NaN	Pune
3	NaN	22.0	Berhampur

In [256...
```
mean_age=data['Age'].mean()
data['Age'].fillna(mean_age)
```

Out[256...
```
0    20.0
1    21.0
2    21.0
3    22.0
Name: Age, dtype: float64
```

In [258…
```
med_age=data['Age'].median()
data['Age'].fillna(med_age)
```

Out[258…
```
0    20.0
1    21.0
2    21.0
3    22.0
Name: Age, dtype: float64
```

In [260…
```
mod_city=data['City'].mode()
data['City'].fillna(mod_city)
```

Out[260…
```
0    Berhampur
1          Hyd
2         Pune
3    Berhampur
Name: City, dtype: object
```

In [ ]: