ABOUT THE DATASET

The dataset consists of direct marketing campaigns data of a "PORTUGUESE BANKING INSTITUTION. There were four variants of the datasets out of which we chose “ bank1.csv” which consists of 45211 data points with 17 independent variables out of which 10 are numeric features and 10 are categorical features. The list of features available to us are given below: bank client data:

age (numeric)

job : type of job (['management', 'technician', 'entrepreneur', 'blue-collar', 'unknown', 'retired', 'admin.', 'services', 'self-employed', 'unemployed', 'housemaid', 'student']

marital : marital status (categorical: ‘divorced’,’married’,’single’,’unknown’; note: ‘divorced’ means divorced or widowed)

education (categorical: ['tertiary', 'secondary', 'unknown', 'primary']

default: has credit in default? (categorical: ‘no’,’yes’,’unknown’) housing: has housing loan? (categorical: ‘no’,’yes’,’unknown’)

loan: has personal loan? (categorical: ‘no’,’yes’,’unknown’) Related with the last contact of the current campaign:

contact: contact communication type (categorical: ‘cellular’,’telephone’,unknown)

month: last contact month of year (categorical: ‘jan’, ‘feb’, ‘mar’, …, ‘nov’, ‘dec’)

day: (1st to 31 st of the month)

duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y=’no’). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model. other attributes:

campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

previous: number of contacts performed before this campaign and for this client (numeric)

poutcome: outcome of the previous marketing campaign (categorical: ['unknown', 'failure', 'other', 'success'])

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
df = pd.read_csv('bank1.csv')
```

```
df.shape
```

```
(45211, 17)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

There are 17 columns & 45211 rows in this dataset

```
df.head(10)
```

	age	job	marital	education	default	balance	housing	loan	contact	day
0	58	management	married	tertiary	no	2143	yes	no	unknown	5
1	44	technician	single	secondary	no	29	yes	no	unknown	5
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5
4	33	unknown	single	unknown	no	1	no	no	unknown	5
5	35	management	married	tertiary	no	231	yes	no	unknown	5
6	28	management	single	tertiary	no	447	yes	yes	unknown	5

```
df.columns
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

```
df
```

	age	job	marital	education	default	balance	housing	loan	contact
0	58	management	married	tertiary	no	2143	yes	no	unknown
1	44	technician	single	secondary	no	29	yes	no	unknown
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown
4	33	unknown	single	unknown	no	1	no	no	unknown
...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular
45207	71	retired	divorced	primary	no	1729	no	no	cellular
45208	72	retired	married	secondary	no	5715	no	no	cellular
45209	57	blue-collar	married	secondary	no	668	no	no	telephone

```
df.isnull().any()
```

```
age          False
job          False
marital      False
education    False
default      False
balance      False
housing      False
loan         False
contact      False
day          False
month        False
duration     False
campaign     False
pdays        False
previous     False
poutcome     False
y            False
dtype: bool
```

```
df.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	
age	45211	NaN	NaN	NaN	40.9362	10.6188	18	33	39	48	
job	45211	12	blue-collar	9732	NaN	NaN	NaN	NaN	NaN	NaN	
marital	45211	3	married	27214	NaN	NaN	NaN	NaN	NaN	NaN	
education	45211	4	secondary	23202	NaN	NaN	NaN	NaN	NaN	NaN	
default	45211	2	no	44396	NaN	NaN	NaN	NaN	NaN	NaN	
balance	45211	NaN	NaN	NaN	1362.27	3044.77	-8019	72	448	1428	10
housing	45211	2	yes	25130	NaN	NaN	NaN	NaN	NaN	NaN	
loan	45211	2	no	37967	NaN	NaN	NaN	NaN	NaN	NaN	
contact	45211	3	cellular	29285	NaN	NaN	NaN	NaN	NaN	NaN	
day	45211	NaN	NaN	NaN	15.8064	8.32248	1	8	16	21	
month	45211	12	may	13766	NaN	NaN	NaN	NaN	NaN	NaN	

Mean Age : 40 , Max Age : 95

Mean Balance : 1362 Max balance : 102127

Rest,we will do deep analysis on each entities

```
df.duplicated().any()  ## No duplicates value, Hence we hv all unique values
```

```
False
```

```
df.apply(lambda x: len(x.unique()))
```

```
age            77
job            12
marital         3
education       4
default         2
balance      7168
housing         2
loan            2
contact         3
day            31
month          12
duration     1573
campaign       48
pdays         559
previous       41
poutcome        4
y               2
dtype: int64
```

```
df.nunique()
```

```
age            77
job            12
marital         3
education       4
default         2
balance      7168
housing         2
loan            2
contact         3
day            31
month          12
duration     1573
campaign       48
pdays         559
previous       41
poutcome        4
y               2
dtype: int64
```

```
loans_counts=df['loan'].value_counts().to_frame()
loans_counts
```

	loan
no	37967
yes	7244

37967 People doesn't have any Loan while 7244 people already have Loan

```
7244/37967*100     ## 19.97 or 20 % People has Taken Loan
```

```
19.079727131456263
```

```
df.education.unique()
```

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

```
#Crosstab to display education stats with respect to Loan
pd.crosstab(index=df["education"],columns=df['loan'])
```

loan	no	yes
education		

Primary Education: 1024 Loans

Secondary Education: 18899 Loans

Tertiary Education: 1784 Loans

Unknown : 133

Now we will calculate how many % People from each Education group has taken Loan

```
1024/(1024+5827)*100    ## Primary Education : 15 % people has taken Loan
```

```
14.946723106115895
```

```
4303/(18899+4303)*100   ## 23.15 % people from Secondary Education has taken Loan
```

```
18.5458150159469
```

```
1784/(11517+ 1784)*100         ## 13 % people from Tertiary Education
```

```
13.412525374032027
```

```
133/(133+1784)*100      ## 7 % approx
```

```
6.9379238393322895
```

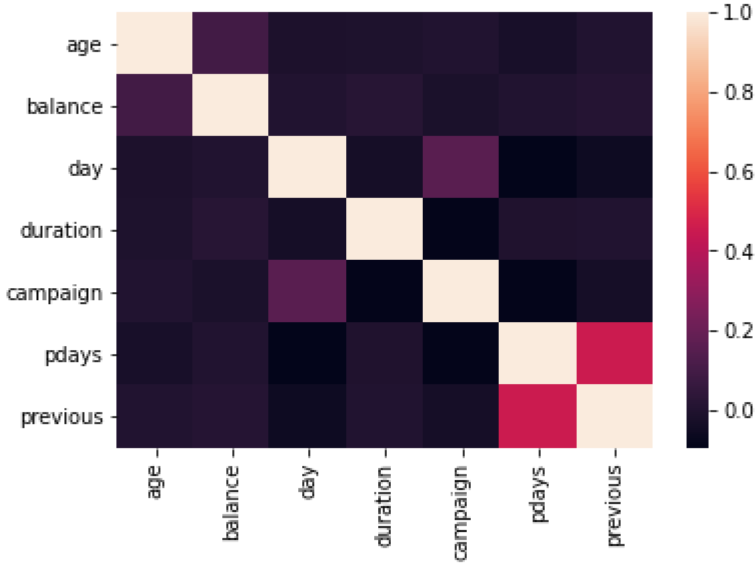
Maximum, Secondary Education has Taken Loan

```
df.corr()
```

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.097783	-0.009120	-0.004648	0.004760	-0.023758	0.001288

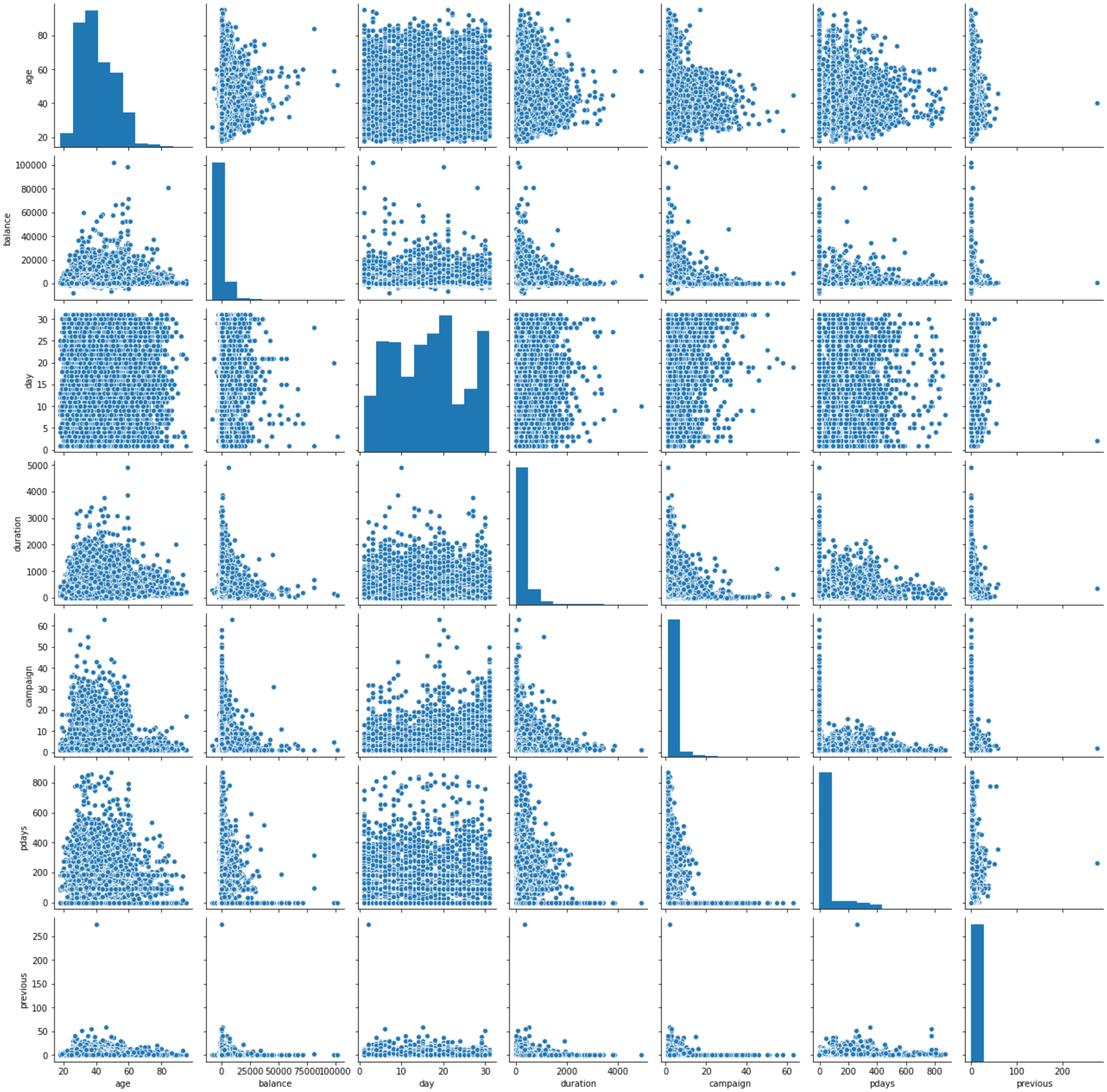
```
sns.heatmap(df.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd21451fa58>
```



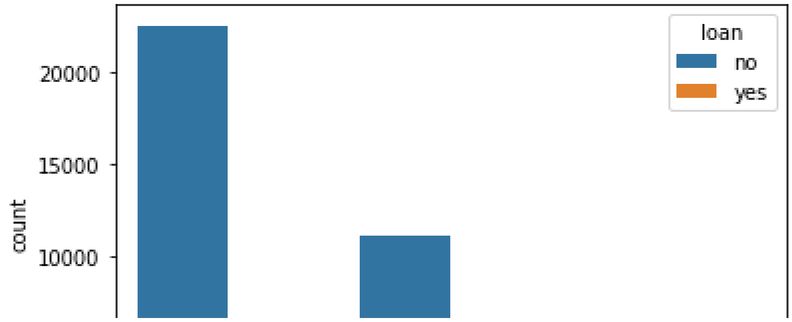

```
sns.pairplot(df)
```

<seaborn.axisgrid.PairGrid at 0x7fd2145620f0>




```
sns.countplot(x="marital", data=df,hue="loan")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd20f46c390>
```



"Married People" are more likely to take Loans , as compared to Single & divorced

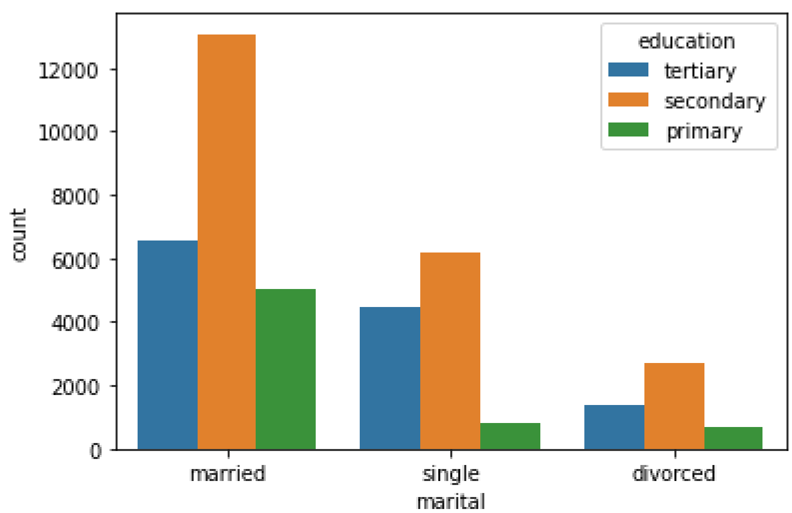
```
5000/20000*100      # Around 25% married people likely to take Loans
```

```
25.0
```

```
df.groupby(["loan"]).count()
```

```
sns.countplot(x="marital", data=df,hue="education")
```

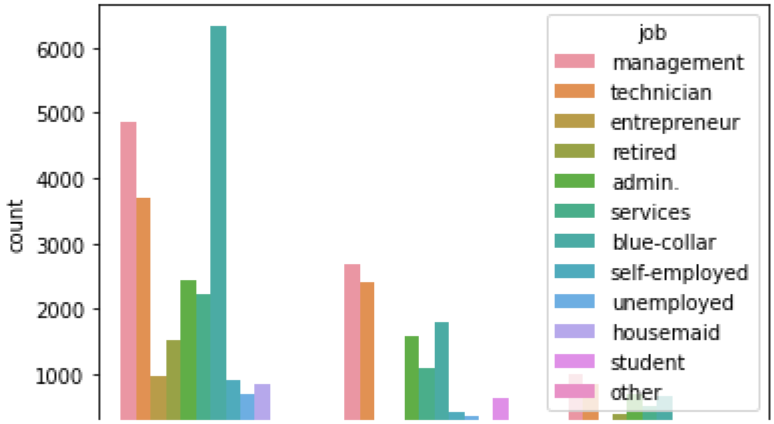
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc23e28c518>
```



Married People with "Secondary Education "are more likely to take loan

```
sns.countplot(x="marital", data=df,hue="job")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc23e54c780>
```



Most Married people have blue Collar jobs or in Management

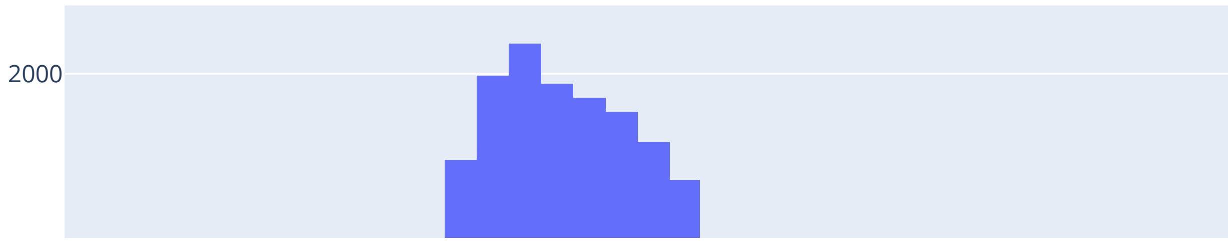
```
import plotly.express as px
```

```
df[['age']].describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	40841.0	40.790676	10.475473	18.0	33.0	39.0	48.0	95.0

Mean Age : 40

```
px.histogram(df , x = 'age')
```



We can see that age is **right skewed** and the average value of age from our targeted user is 40.

```
df[['job']].describe().T
```

	count	unique	top	freq
job	45211	12	blue-collar	9732

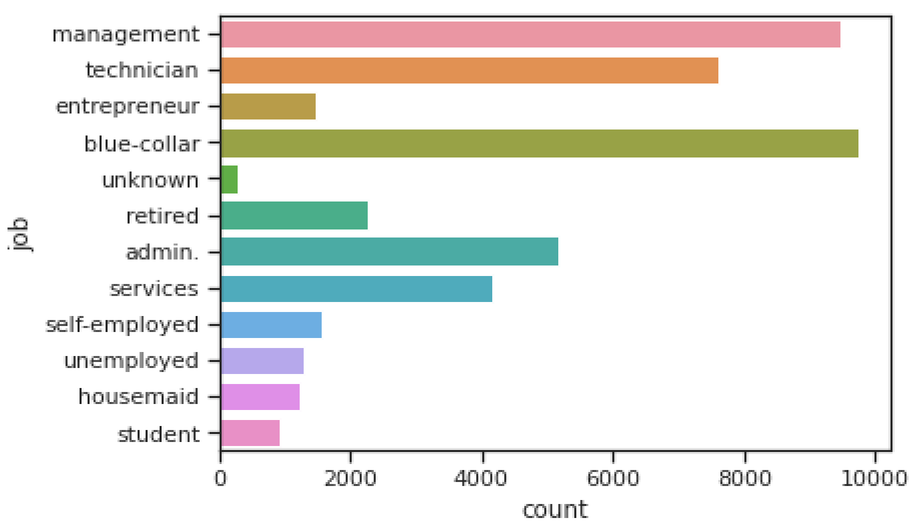
There are 12 types of job people are doing. Maximum people falls in "Blue-Collar" Category ie , 9732

```
9732/45211*100   ## Around 22 % people are Blue-Collar guy
```

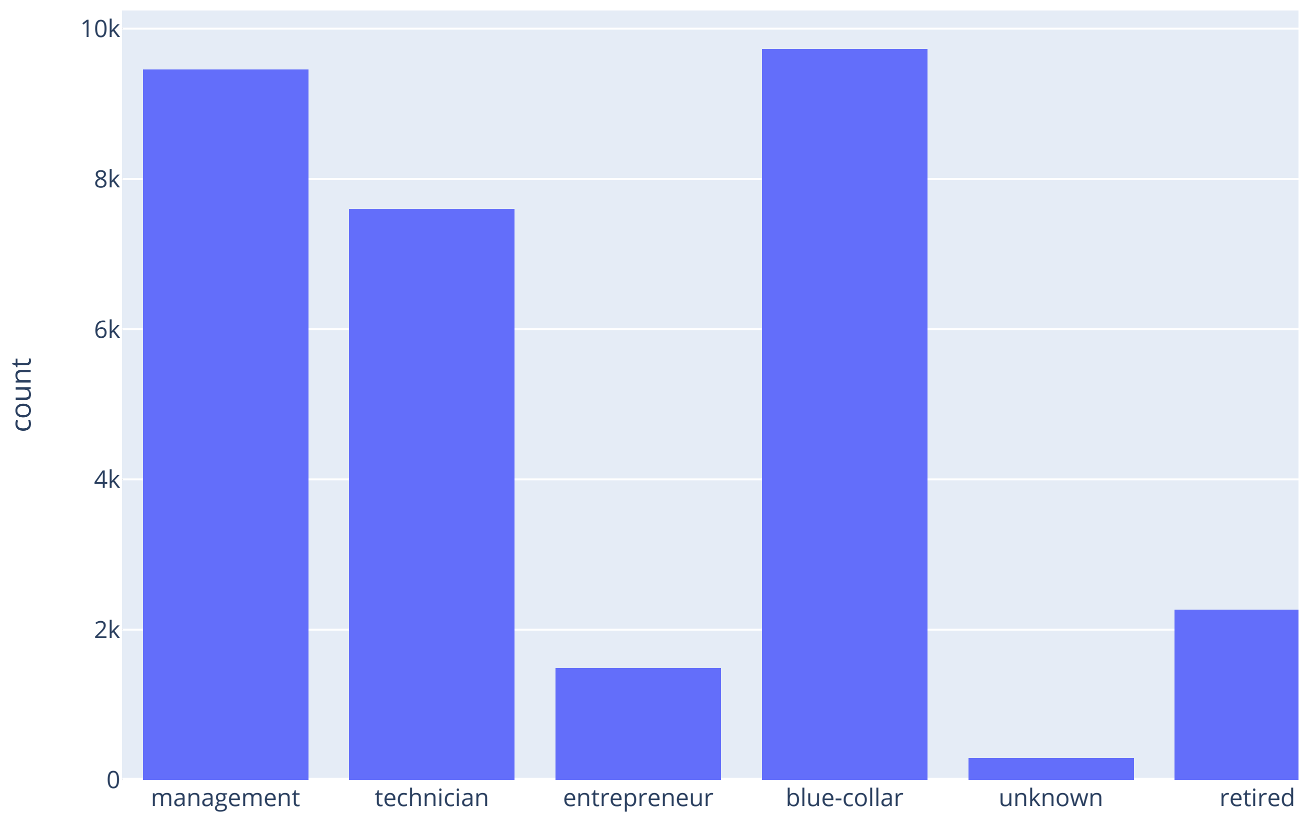
```
21.525734887527374
```

```
sns.set(style="ticks", color_codes=True)
sns.countplot(y='job', data=df)                    ##   THROUGH VISUALISATION
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd20ac9f748>
```




```
px.histogram(df , x = 'job')
```



We can observer that most of our targeted users are either Management or blue-collar workers.

We've also observed targeting retired , unemployed,housemaid and students , We would study the conversion rate.

Marital Status

We look at the distribution of Marital status

```
df.marital.unique()
```

```
array(['married', 'single', 'divorced'], dtype=object)
```

```
df.marital.value_counts()
```

```
married     23248
single      10508
divorced     4520
Name: marital, dtype: int64
```

'married', 'single', 'divorced'] - 3 types of Statuses

```
df[['marital']].describe().T
```

	count	unique	top	freq
marital	45211	3	married	27214

```
23248/45211*100
```

```
51.421114330583265
```

51 % People are Married

EDUCATION

We look at the distribution of education among our targeted users.

```
df[['education']].describe().T
```

	count	unique	top	freq
education	45211	4	secondary	23202

Most people are " Secondary educated"

```
df.education.value_counts()
```

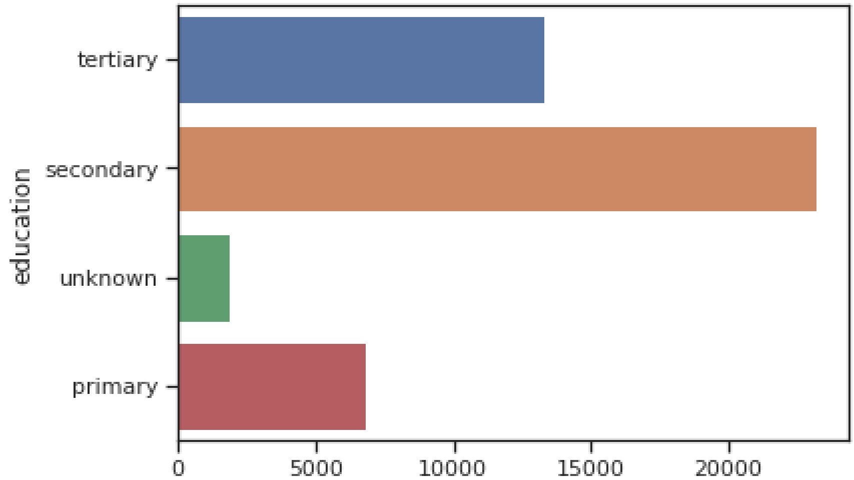
```
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: education, dtype: int64
```

```
23202/45211*100    ## Around 51 % are Secondary educated
```

```
51.3193691800668
```

```
sns.countplot(y='education', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd20ab9e198>
```



Default , Housing Loan , Personal Loans

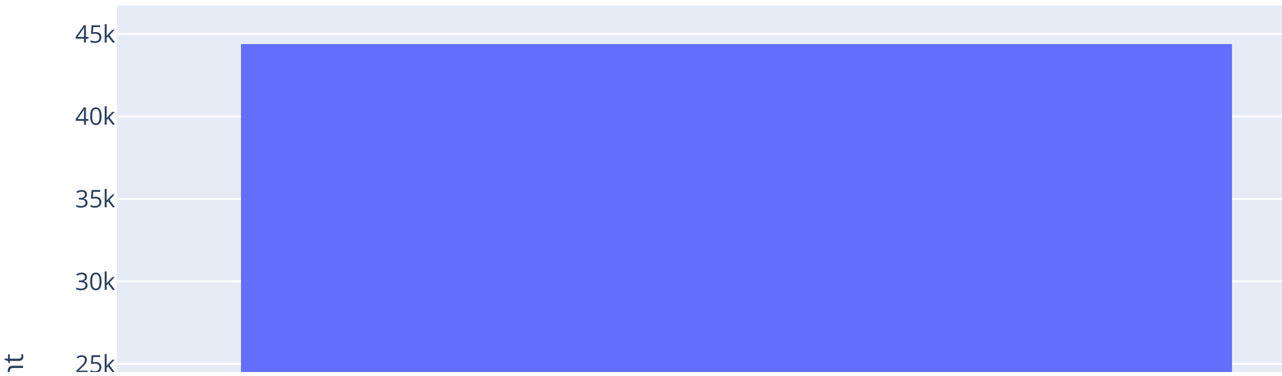
Here we look whether our targeted users are defaulters , or have a personal/housing loan.

```
df[['default','housing','loan']].describe().T
```

	count	unique	top	freq
default	45211	2	no	44396
housing	45211	2	yes	25130
loan	45211	2	no	37967

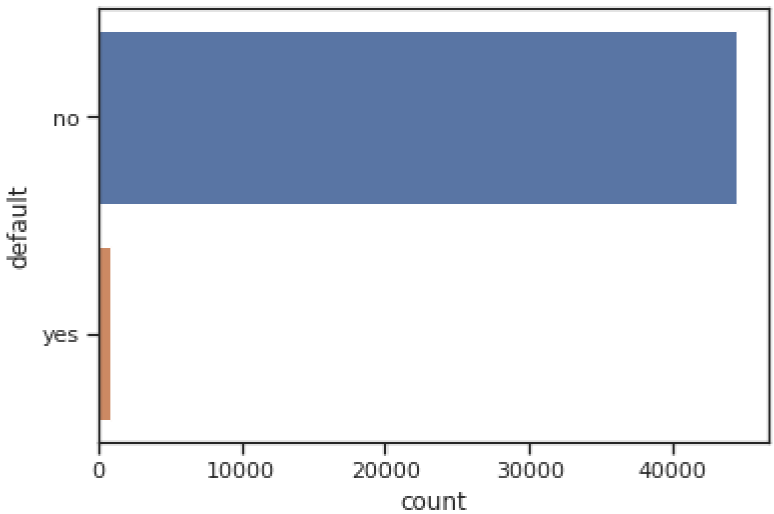
Default

```
px.histogram(df , x='default')
```




```
sns.countplot(y ='default',data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd20a135f28>
```




```
df.default.value_counts()     # 815 people are defaulters
```

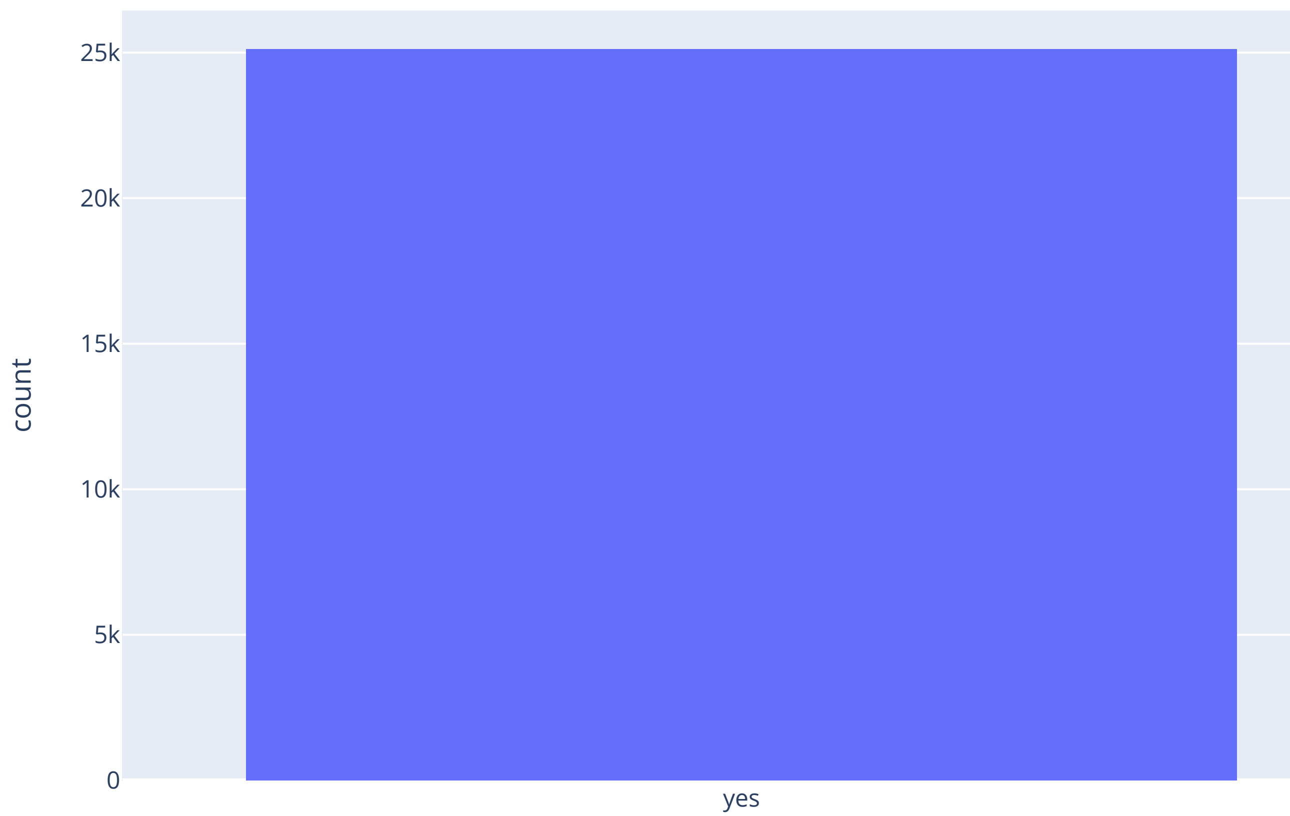
```
no     44396
yes      815
Name: default, dtype: int64
```

```
815/45211*100      #  2 %  people are "Defaulters"
```

```
1.8026586450200173
```

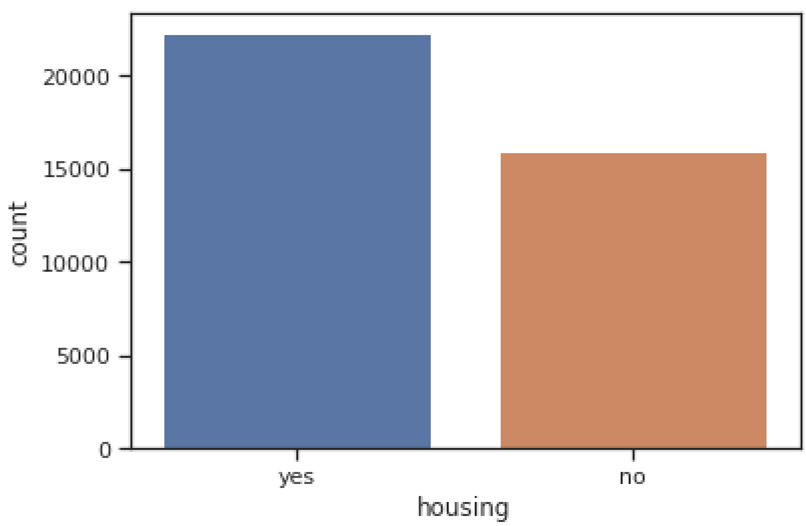
Housing Loan

```
px.histogram(df , x='housing')
```




```
sns.countplot(data = df1, x = 'housing')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd20a0f29e8>
```




```
df.housing.value_counts()
```

```
yes    25130
no     20081
Name: housing, dtype: int64
```

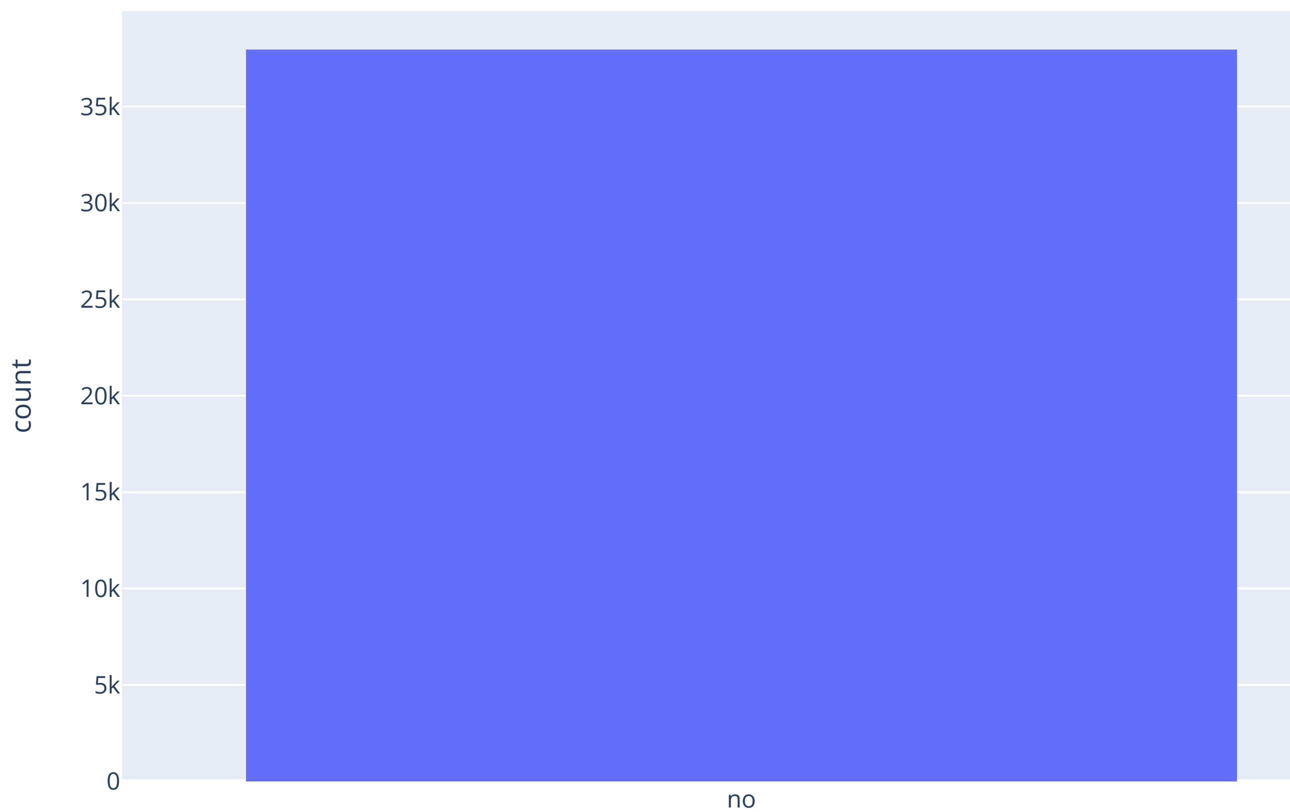
```
25130/45211*100
```

```
55.583818097365686
```

56 % People have housing Loans

Personal Loan

```
px.histogram(df , x='loan')
```




```
df.loan.value_counts()
```

```
no     37967
yes     7244
Name: loan, dtype: int64
```

```
7244/45211*100
```

```
16.022649355245406
```

Personal Loan Count = 16 %

We can see most of the people do not have a personal loan But have a "Housing loan" . We're also observing the most of our targeted customers are not defaulters

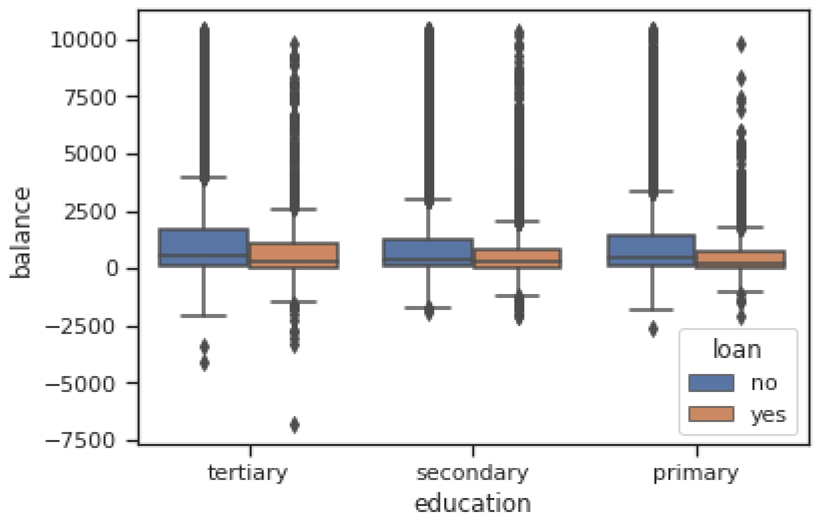
Hence, bank should be more focussed towards "Housing Loans" & Should put more schemas and Marketing Campaigns in the same.

```
px.box(df1,x='balance',y='age')
```

```
                    ]]
fig = plt.figure(figsize=(25, 25))
plt.suptitle('Pie Chart Distributions', fontsize=20)
for i in range(1, df2.shape[1] + 1):
    plt.subplot(6, 3, i)
    f = plt.gca()
    f.axes.get_yaxis().set_visible(False)
    f.set_title(df2.columns.values[i - 1])

    values = df2.iloc[:, i - 1].value_counts(normalize = True).values
    index = df2.iloc[:, i - 1].value_counts(normalize = True).index
    plt.pie(values, labels = index, autopct='%1.1f%%')
    plt.axis('equal')
fig.tight_layout(rect=[10, 10, 10, 10])
```

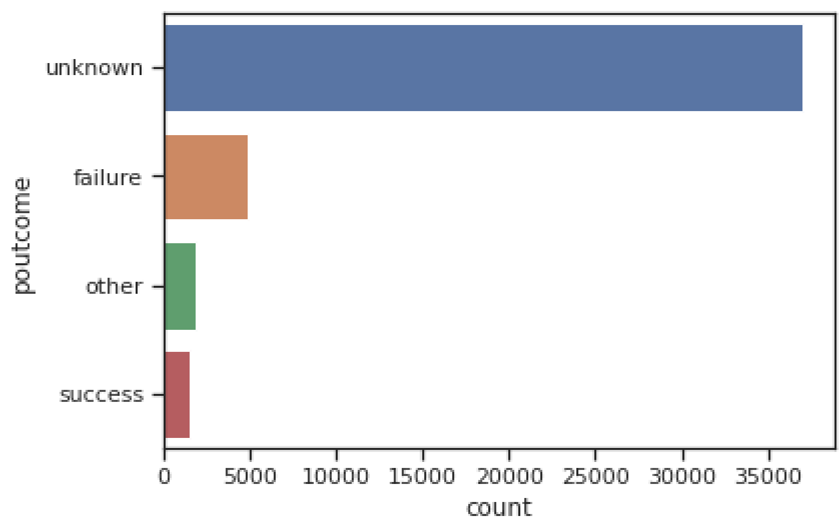
```
sns.boxplot(x=df['education'],y=df['balance'],hue=df['loan'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd209446e10>
```




```
sns.countplot(y = 'poutcome',data =df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd209446c18>
```

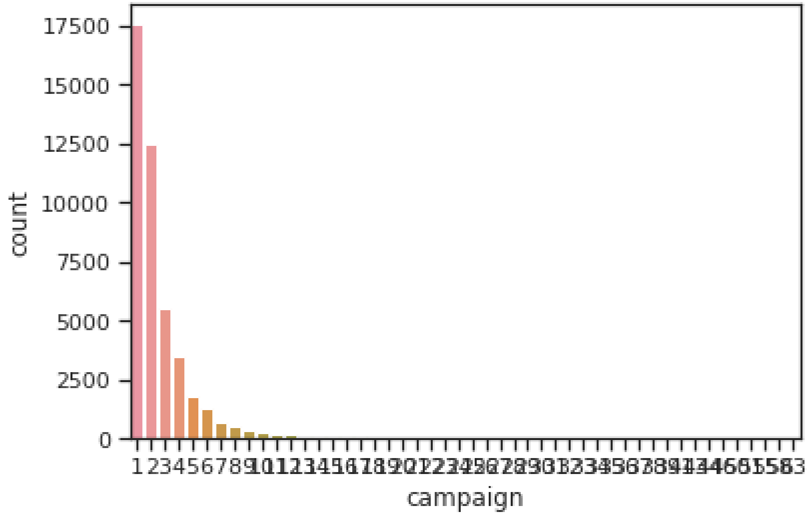


poutcome: This feature denotes the outcome of the previous marketing campaign.

Here ,We found most Campaigns to be failure.

```
sns.countplot(x = 'campaign',data =df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd209382ef0>
```

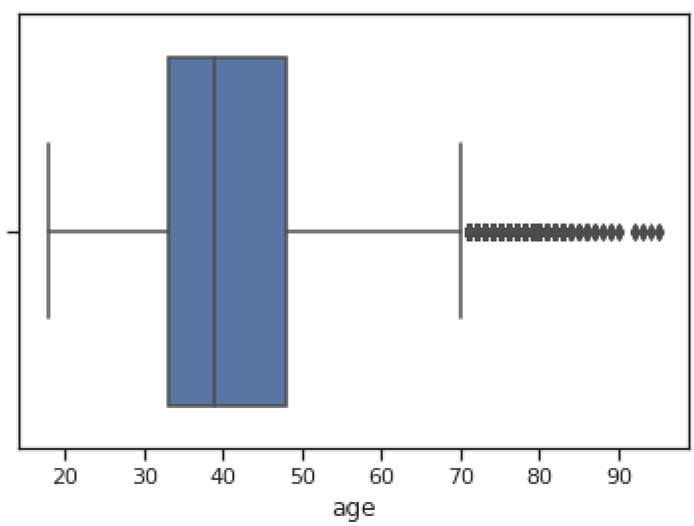


Hence,maximum number of Campaigns ,were in the first Year. Since, the Campaigns were failurre, the bank keep on decreasing the Marketing campaigns.

Therefore,we infer that Bank has Lost huge money due to Marketing Failure

```
sns.boxplot(x ='age', data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd209713dd8>
```

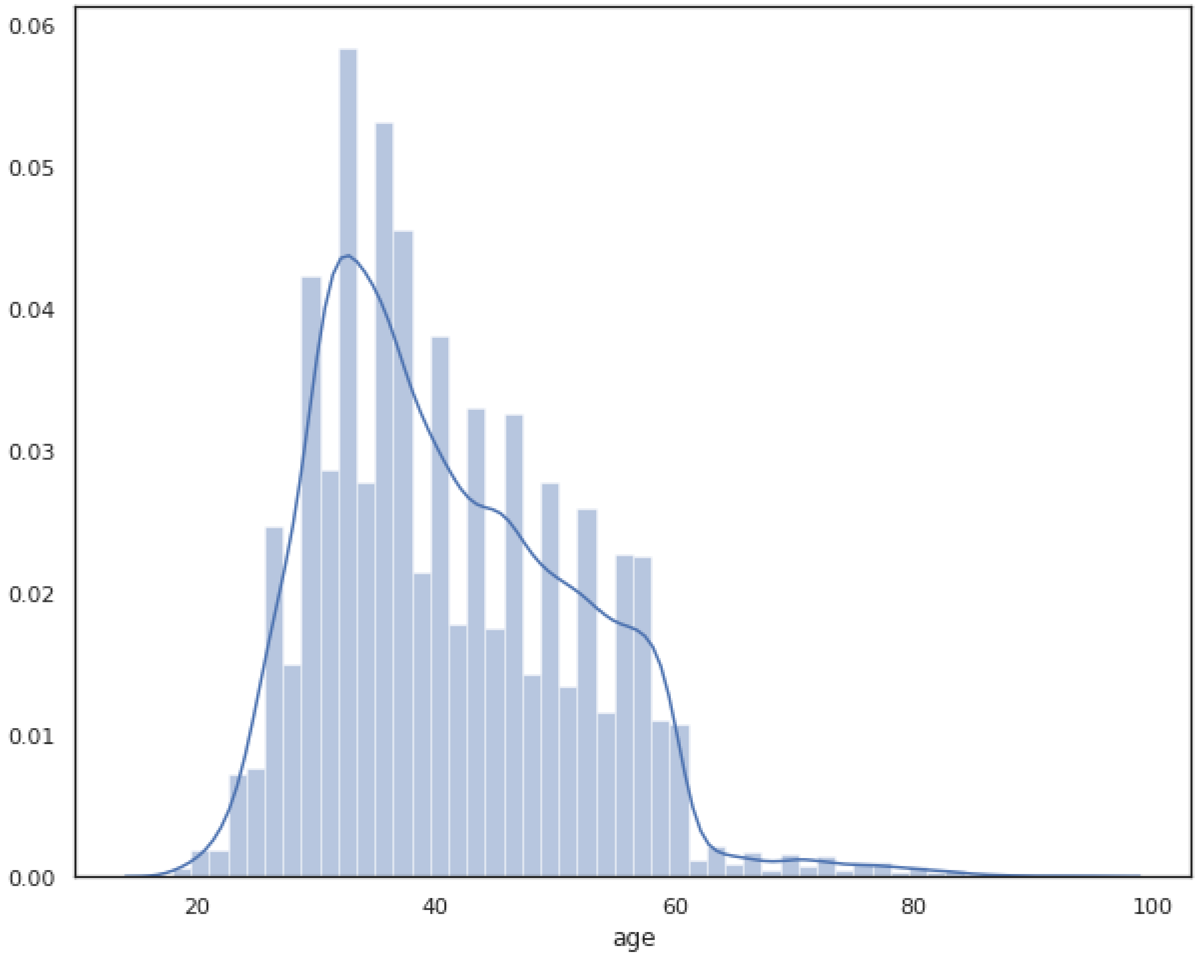


We find that maximum people falls in Age between 30 to 50

```
plt.figure(figsize=(10,8))
```

```
plt.figure(figsize=(10,8))
sns.distplot(df['age'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fd200996588>



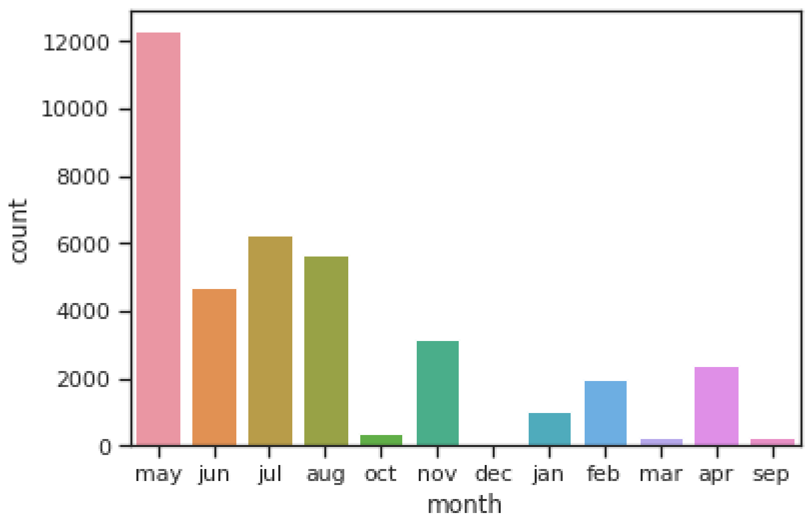

```
plt.figure(figsize=(10,8))
sns.distplot(df['duration'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1ff9c7fd0>
```

This gives us the insights on the (last contract duration),which we found Upto 20 max

```
sns.countplot(data = df1, x='month')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2098ca470>
```



Hence,we find that "May to August" is the Peak season ,for contracts & business.

Hence, we can also say that,maximum Marketing Campaigns should be done in these Months.

Maximum,schemas should be offered to "Secondary Education Class" & Target Age is : 30 to 45 years as most Customer falls in this Age which are in need of Loan.

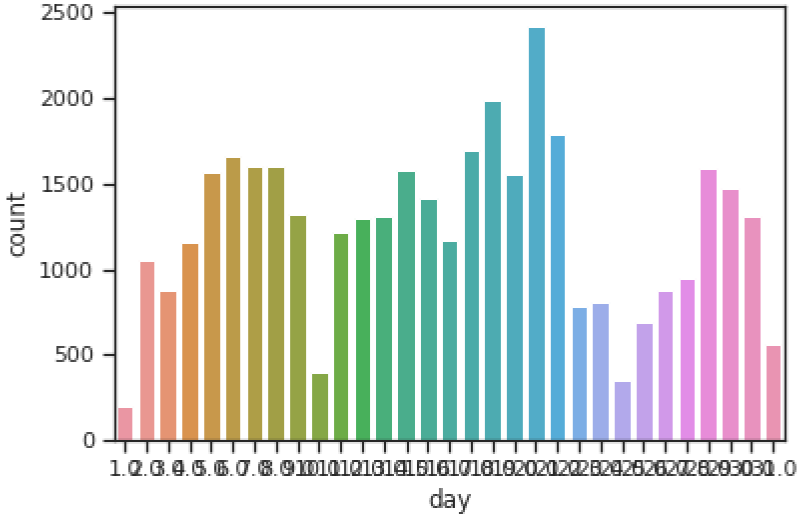
Also,People are more likely to take "Housing Loans as Stated Earlier",bank should put more schemas on housing loans.

Married People are more likely to take Loans,so bank should bring up the policy accordingly.

```
sns.countplot(data = df, x='day')
```

```
sns.countplot(data = df, x= 'day')
```

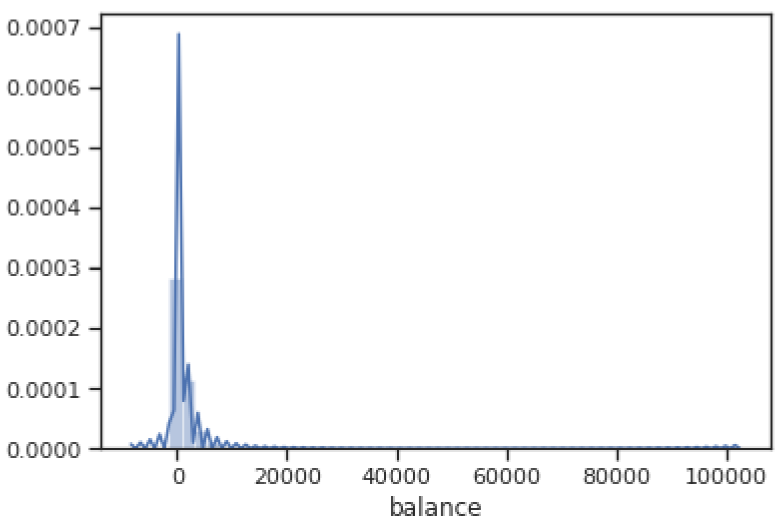
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd209cf2128>
```



Very Nice imformation we got, as we know the an Average Sales cycle for conversion is 20 days. Here, we see that in between 15th to 21st,there is maximum Conversion.

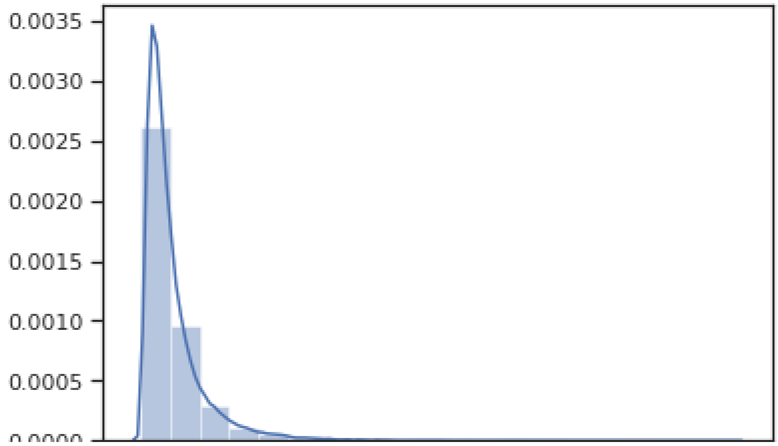
```
sns.distplot(df['balance'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd208f80b00>
```



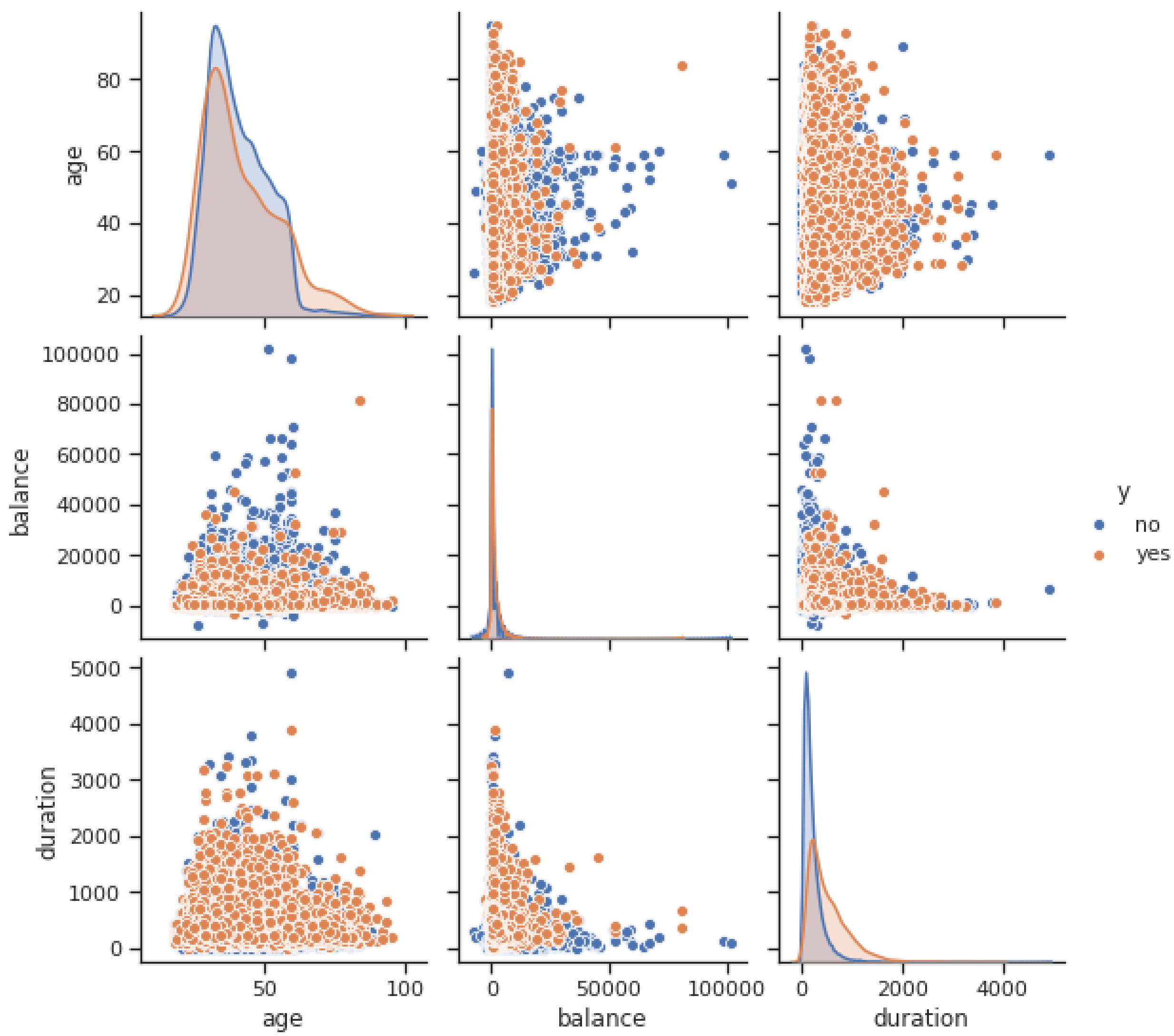

```
sns.distplot(df.duration, bins = 20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd20094f748>
```




```
sns.pairplot(data=df, hue='y', vars= ['age', 'balance', 'duration'])
```

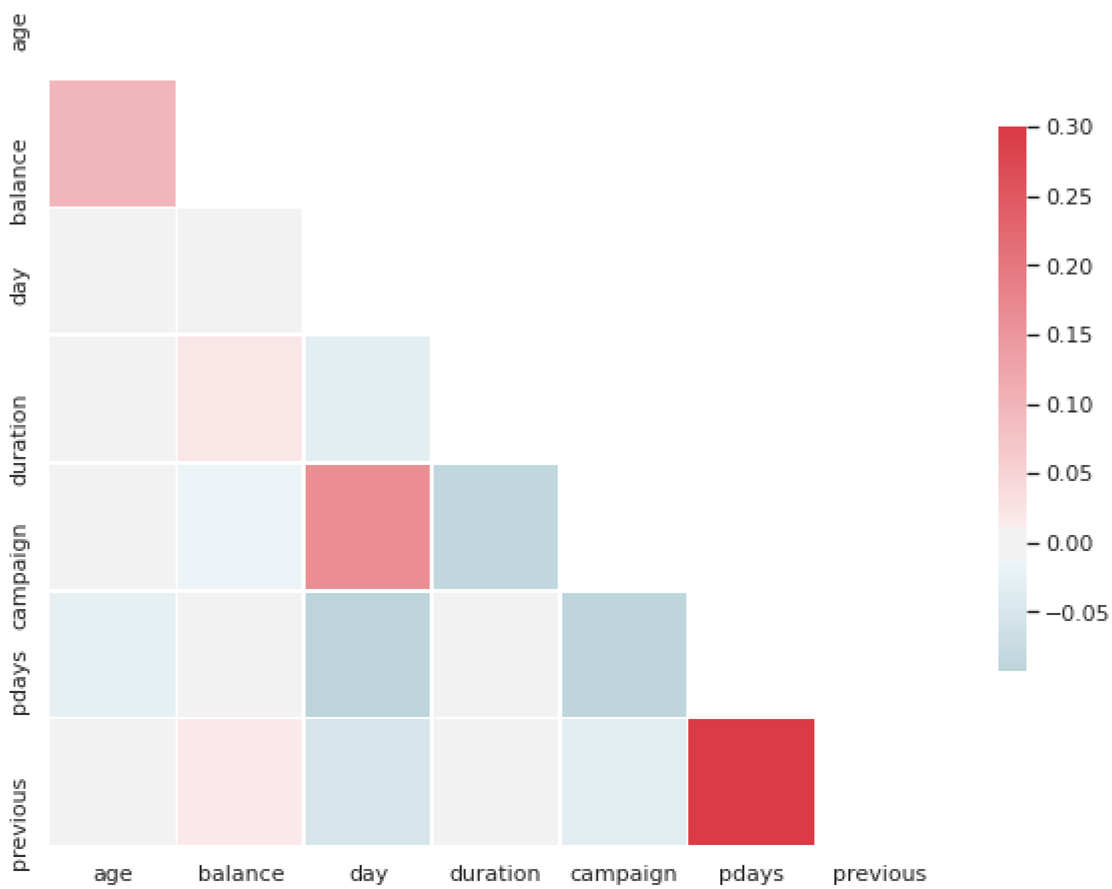
```
<seaborn.axisgrid.PairGrid at 0x7fd200d06d30>
```



Comparing we found that,that the age group between 30 to 40 have been contacted for more duration .

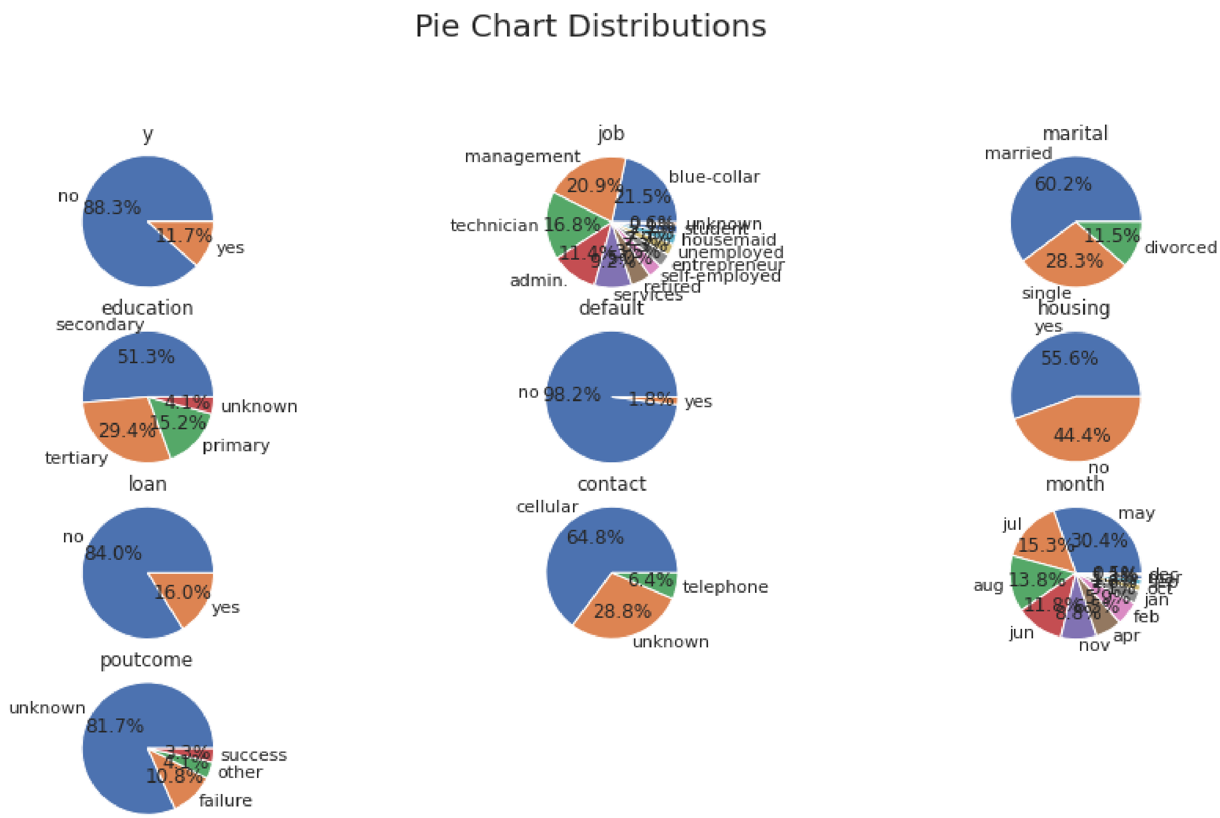
```
## Correlation Matrix
sns.set(style="white")

# Compute the correlation matrix
corr1= df.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr1, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10, 10))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr1, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd20085d550>
```




```
df.columns
df2 = df[['y','job','marital', 'education', 'default', 'housing','loan', 'contact',
             'month', 'poutcome'
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: UserWarning:

Tight layout not applied. The left and right margins cannot be made large enough to acc
```




```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: UserWarning:

Tight layout not applied. The left and right margins cannot be made large enough to acc
```