
BUSINESS
Economy of the European Union
25 great jobs for people who love to travel
8:30 AM - Marketing System meeting
2:45 PM - Review PPA user manual
* Call John again to arrange meeting
Book flight to Sydney
Sat, 10:00 AM
Flight: SD1675
AM - Office meeting
→ Organize new system
→ Develop social selling
→ Review background
Selling Pie Chart

Banking Project

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

warnings.filterwarnings('ignore')
bank_df = pd.read_csv('bank.csv')
bank_df.head()
```

```
   age         job  marital  education default  balance housing loan
contact  \
0   59      admin.  married  secondary      no     2343     yes   no
unknown
1   56      admin.  married  secondary      no       45      no   no
unknown
2   41  technician  married  secondary      no     1270     yes   no
unknown
3   55    services  married  secondary      no     2476     yes   no
unknown
4   54      admin.  married   tertiary      no      184      no   no
unknown

   day month  duration  campaign  pdays  previous poutcome deposit
0    5   may      1042         1     -1         0  unknown     yes
1    5   may      1467         1     -1         0  unknown     yes
2    5   may      1389         1     -1         0  unknown     yes
3    5   may       579         1     -1         0  unknown     yes
4    5   may       673         2     -1         0  unknown     yes
```

```
print(bank_df.columns,'\n',bank_df.shape)
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance',
'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign',
'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
 (11162, 17)
```

1. Make the data proper to make use of data for analysis

```
# A. Identify the Features data types before entering into the
analysis
# B. .Convert the datatypes which are wrongly identified according to
the business(domain).
# Kindly use the User Defined function and loop to convert the data
```

```
types once.
# C. Find and Remove missing if any. Use visualization to find the
missing values or Use
# general method to find the missing values.
# D. Find duplicates (if necessary)
```

```
bank_df.dtypes
```

```
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
deposit      object
dtype: object
```

```
bank_df = bank_df.rename(columns = {'contact':'Contact_type'}) #
renaming the column 'contact' to 'Contact_type'
bank_df[['Contact_type']].head(3).reset_index() # verifying the change
```

```
   index Contact_type
0      0      unknown
1      1      unknown
2      2      unknown
```

```
for i in bank_df.columns:
    if bank_df[i].nunique() < 10:
        print(i ,'\n', bank_df[i].unique()) # identifying the unique
values present in columns which has nunique below 10
```

```
marital
 ['married' 'single' 'divorced']
education
 ['secondary' 'tertiary' 'primary' 'unknown']
default
 ['no' 'yes']
housing
 ['yes' 'no']
loan
 ['no' 'yes']
```

```
Contact_type
 ['unknown' 'cellular' 'telephone']
poutcome
 ['unknown' 'other' 'failure' 'success']
deposit
 ['yes' 'no']
```

```
D = {}
for i in bank_df.columns:
    if bank_df[i].nunique() < 10:
        D[i] = bank_df[i].unique().tolist()

df = pd.DataFrame.from_dict(D, orient = 'index').transpose()
display(df)  # This code will create a DataFrame df where each column
contains the unique values from bank_df for columns with less than 10
unique values.
```

```
    marital  education default housing  loan Contact_type poutcome
deposit
0   married  secondary      no     yes    no      unknown  unknown
yes
1    single   tertiary     yes      no   yes     cellular    other
no
2  divorced    primary    None    None  None    telephone  failure
None
3      None    unknown    None    None  None         None  success
None
```

```
bank_df.isnull().sum() # this will give columnwise null values in
dataframe
```

```
age             0
job             0
marital         0
education       0
default         0
balance         0
housing         0
loan            0
Contact_type    0
day             0
month           0
duration        0
campaign        0
pdays           0
previous        0
poutcome        0
deposit         0
dtype: int64
```

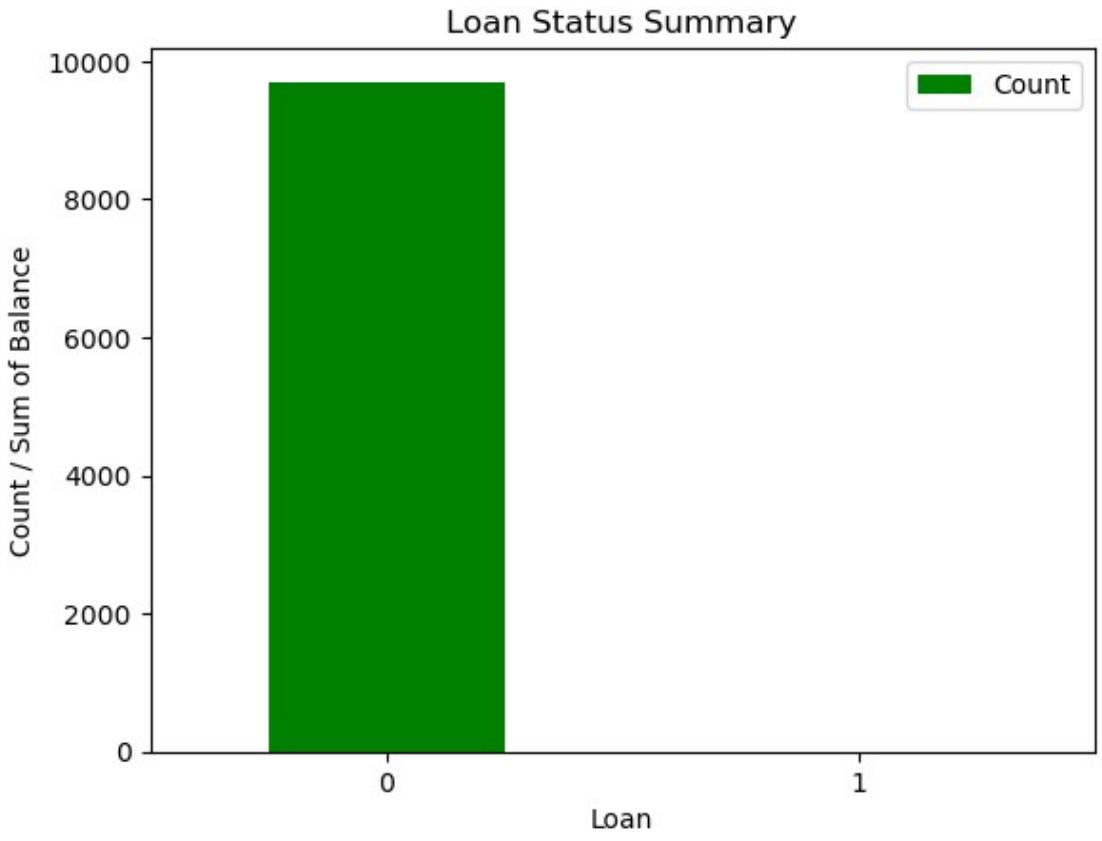
```
df.isnull().sum()
```

```
marital         1
education       0
default         2
housing         2
loan            2
Contact_type    1
poutcome        0
deposit         2
dtype: int64
```

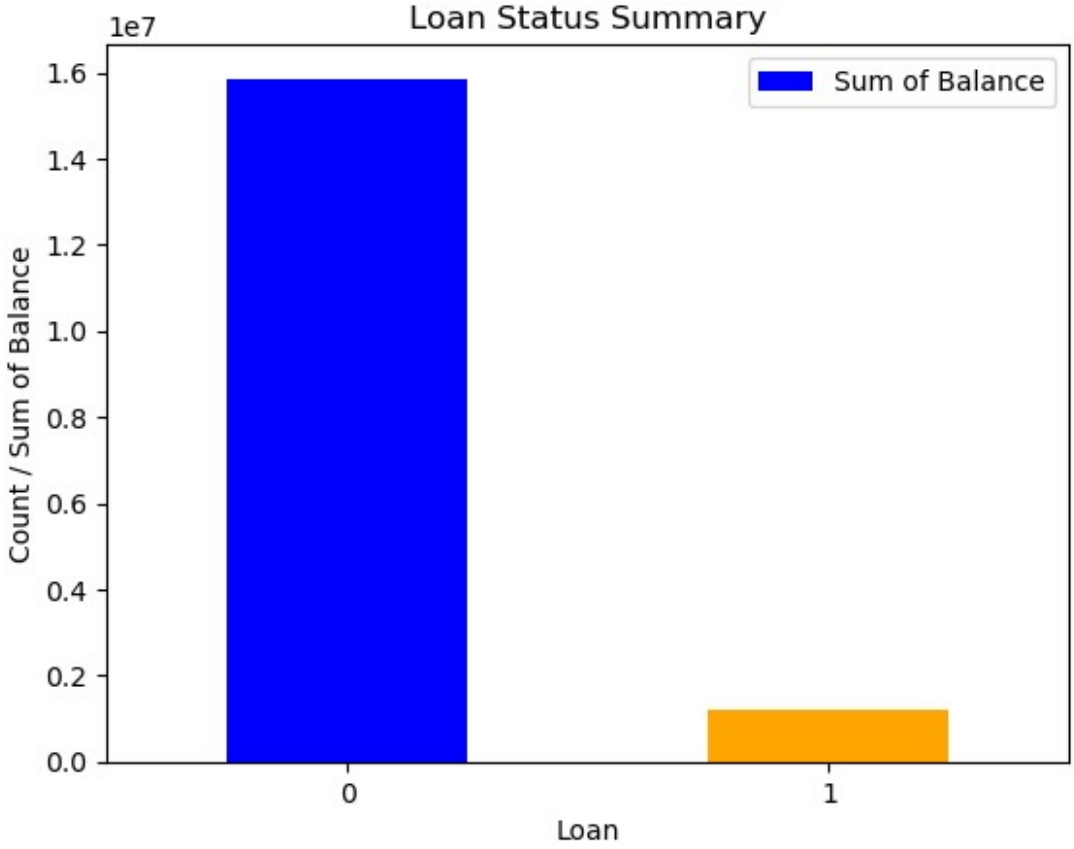
```
loan_summary = bank_df.groupby('loan').agg({'loan': 'count',
'balance': 'sum'})
loan_summary.columns = ['Count', 'Sum of Balance']
loan_summary = loan_summary.reset_index()
loan_summary
```

```
  loan  Count  Sum of Balance
0   no   9702        15856680
1  yes   1460         1204867
```

```
loan_summary.plot(kind='bar', stacked=True, rot=0, xlabel='Loan',y =
'Count', title='Loan Status Summary',color = ['Green','white'])
plt.ylabel('Count / Sum of Balance')
plt.legend()
plt.show()
```



```
loan_summary.plot(kind='bar', stacked=True, rot=0, xlabel='Loan',y =
'Sum of Balance', title='Loan Status Summary',color =
['Blue','orange'])
plt.ylabel('Count / Sum of Balance')
plt.legend()
plt.show()
```

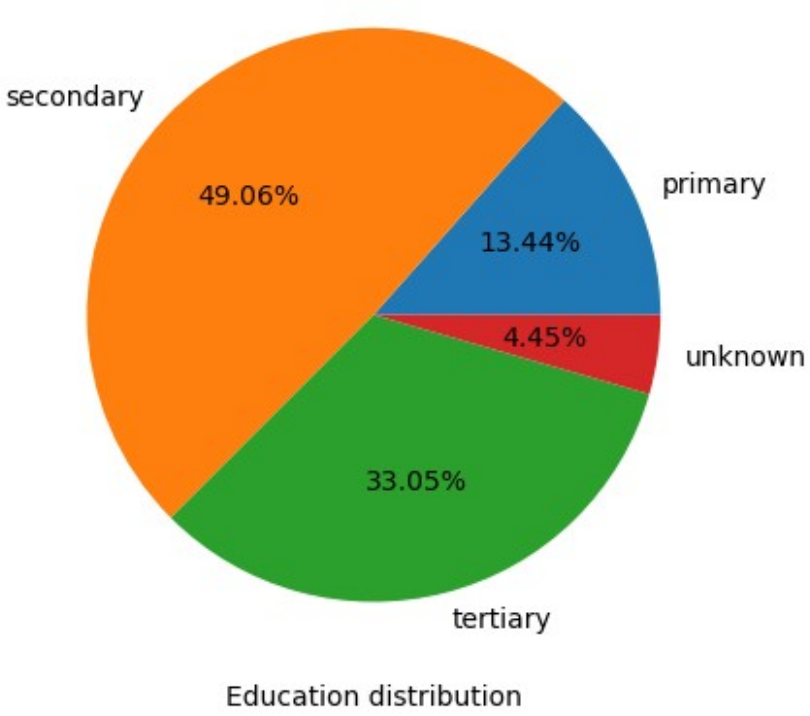



```
education_summary = bank_df.groupby('education').value_counts()
education_summary =education_summary.reset_index()
education_summary.head()
```

```
  education  age      job  marital default  balance housing loan
Contact_type  \
0   primary   18  student   single      no      608      no   no
cellular
1   primary   18  student   single      no      608      no   no
cellular
2   primary   86  retired  married      no     1255      no   no
telephone
3   primary   83  retired  married      no      425      no   no
telephone
4   primary   83  retired   single      no     1965      no   no
telephone
```

	day	month	duration	campaign	pdays	previous	poutcome	deposit	0
0	12	aug	267	1	-1	0	unknown	yes	1
1	13	nov	210	1	93	1	success	yes	1
2	14	jul	247	1	180	3	success	yes	1
3	22	sep	773	1	92	2	success	yes	1
4	13	oct	1003	3	-1	0	unknown	yes	1

```
bank_df.groupby('education')[['education']].value_counts().plot(kind =
'pie',autopct = '%.2f%%')
plt.xlabel('Education distribution')
plt.show()
```




```
bank_df['month'].sort_values().unique()
```

```
array(['apr', 'aug', 'dec', 'feb', 'jan', 'jul', 'jun', 'mar', 'may',
       'nov', 'oct', 'sep'], dtype=object)
```

```
#Assigning quarters based on month
def season(i):
    if i in ['jan','feb','mar']:
```

```
        return 'Q1'
    if i in ['apr','may','jun']:
        return 'Q2'
    if i in ['jul','aug','sep']:
        return 'Q3'
    if i in ['oct','nov','dec']:
        return 'Q4'
    return season
bank_df['Quarter'] = bank_df['month'].apply(season)
bank_df[['Quarter','month']] .head()
```

```
  Quarter month
0      Q2   may
1      Q2   may
2      Q2   may
3      Q2   may
4      Q2   may
```

```
bank_df.groupby('Quarter')[['Quarter']].value_counts().plot(kind =
'bar',title='Quarter
distribution',color=['Blue','green','red','brown'])
plt.xlabel('Quarter')
plt.ylabel('Count')
plt.show()
```