In [947…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind
from scipy.stats import chi2_contingency
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from warnings import filterwarnings
```

In [313…
```
filterwarnings("ignore")
```

Reading the data

In [314…
```
bank_df=pd.read_csv("Dataset-Data/bank.csv",sep=";")
bank_df.head()
```

Out[314…

	age	job	marital	education	default	balance	housing	loan	contact	day
0	30	unemployed	married	primary	no	1787	no	no	cellular	19
1	33	services	married	secondary	no	4789	yes	yes	cellular	11
2	35	management	single	tertiary	no	1350	yes	no	cellular	16
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5

Numerical and Categorical columnn analysis

In [315…
```
keys=bank_df.dtypes.keys()
dtype=bank_df.dtypes.values

cat_col=[]
num_col=[]

for i in zip(dtype,keys):
    if i[0] == 'object':
        cat_col.append(i[1])
    else:
        num_col.append(i[1])

print(f"Categorical columns are : {cat_col}")
print(f"Numerical columns are : {num_col}")
```

```
Categorical columns are : ['job', 'marital', 'education', 'default', 'housing', 'loa
n', 'contact', 'month', 'poutcome', 'y']
Numerical columns are : ['age', 'balance', 'day', 'duration', 'campaign', 'pdays',
'previous']
```

Data quick checking

In [316…
```
bank_df.shape
```

Out[316… (4521, 17)

In [317…
```
bank_df.size
```

Out[317… 76857

In [318…
```
len(bank_df)
```

Out[318… 4521

In [319…
```
bank_df.head(3)
```

Out[319…

	age	job	marital	education	default	balance	housing	loan	contact	day	r
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	

In [320…
```
bank_df.tail()
```

Out[320…

	age	job	marital	education	default	balance	housing	loan	contact	da
4516	33	services	married	secondary	no	-333	yes	no	cellular	3
4517	57	self-employed	married	tertiary	yes	-3313	yes	yes	unknown	
4518	57	technician	married	secondary	no	295	no	no	cellular	1
4519	28	blue-collar	married	secondary	no	1137	no	no	cellular	
4520	44	entrepreneur	single	tertiary	no	1136	yes	yes	cellular	

In [321…
```
bank_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        4521 non-null   int64 
 1   job        4521 non-null   object
 2   marital    4521 non-null   object
 3   education  4521 non-null   object
 4   default    4521 non-null   object
 5   balance    4521 non-null   int64 
 6   housing    4521 non-null   object
 7   loan       4521 non-null   object
 8   contact    4521 non-null   object
 9   day        4521 non-null   int64 
 10  month      4521 non-null   object
 11  duration   4521 non-null   int64 
 12  campaign   4521 non-null   int64 
 13  pdays      4521 non-null   int64 
 14  previous   4521 non-null   int64 
 15  poutcome   4521 non-null   object
 16  y          4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
```

In [322…
```
bank_df.dtypes
```

Out[322…
```
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 [323…
```
round(bank_df.describe(),2)
```

Out[323...

	age	balance	day	duration	campaign	pdays	previous
count	4521.00	4521.00	4521.00	4521.00	4521.00	4521.00	4521.00
mean	41.17	1422.66	15.92	263.96	2.79	39.77	0.54
std	10.58	3009.64	8.25	259.86	3.11	100.12	1.69
min	19.00	-3313.00	1.00	4.00	1.00	-1.00	0.00
25%	33.00	69.00	9.00	104.00	1.00	-1.00	0.00
50%	39.00	444.00	16.00	185.00	2.00	-1.00	0.00
75%	49.00	1480.00	21.00	329.00	3.00	-1.00	0.00
max	87.00	71188.00	31.00	3025.00	50.00	871.00	25.00

In [324...
```
bank_df.skew(numeric_only=True)
```

Out[324...
```
age          0.699501
balance      6.596431
day          0.094627
duration     2.772420
campaign     4.743914
pdays        2.717071
previous     5.875259
dtype: float64
```

In [325...
```
bank_df.kurt(numeric_only=True)
```

Out[325...
```
age          0.348775
balance     88.390332
day         -1.039531
duration    12.530050
campaign    37.168920
pdays        7.957128
previous    51.995212
dtype: float64
```

Data Cleaning

In [326...
```
bank_df.sample(5)
```

Out[326...

	age	job	marital	education	default	balance	housing	loan	contact	c
1823	35	technician	divorced	secondary	no	473	yes	yes	unknown	
1938	44	self-employed	divorced	secondary	no	80	yes	yes	unknown	
2615	45	blue-collar	married	secondary	no	776	yes	no	cellular	
1758	30	blue-collar	married	primary	no	-336	no	no	cellular	
2215	34	management	single	unknown	no	1534	yes	no	cellular	

i) Categorical Column Cleaning

In [327...
```
print(cat_col)
```

```
['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'p
outcome', 'y']
```

In [328...
```
for i in cat_col:
    cat_count=bank_df[i].unique()
    print(f"{i} : {cat_count}")
```

```
job : ['unemployed' 'services' 'management' 'blue-collar' 'self-employed'
 'technician' 'entrepreneur' 'admin.' 'student' 'housemaid' 'retired'
 'unknown']
marital : ['married' 'single' 'divorced']
education : ['primary' 'secondary' 'tertiary' 'unknown']
default : ['no' 'yes']
housing : ['no' 'yes']
loan : ['no' 'yes']
contact : ['cellular' 'unknown' 'telephone']
month : ['oct' 'may' 'apr' 'jun' 'feb' 'aug' 'jan' 'jul' 'nov' 'sep' 'mar' 'dec']
poutcome : ['unknown' 'failure' 'other' 'success']
y : ['no' 'yes']
```

In [329...
```
# In job column there are some special character like '.' and '-' we are going to r

bank_df['job'].replace('admin.','admin',inplace=True)
bank_df['job']=list(map(lambda x : "blue collar" if x=="blue-collar" else x,bank_df

def replace_data(x):
    if x=="self-employed":
        return "self employed"
    else:
        return x

bank_df['job']=bank_df['job'].apply(replace_data)

bank_df['job'].unique()
```

Out[329…
```
array(['unemployed', 'services', 'management', 'blue collar',
       'self employed', 'technician', 'entrepreneur', 'admin', 'student',
       'housemaid', 'retired', 'unknown'], dtype=object)
```

ii) Numerical Column Cleaning

In [330…
```
print(num_col)
```

```
['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']
```

In [331…
```
for i in num_col:
    mixed=[j for j in bank_df[i] if isinstance(j,str)]

    if len(mixed) != 0:
        bank_df[i].replace(mixed,np.nan)

    print(f"Mixed data in '{i}' column is {mixed}")
```

```
Mixed data in 'age' column is []
Mixed data in 'balance' column is []
Mixed data in 'day' column is []
Mixed data in 'duration' column is []
Mixed data in 'campaign' column is []
Mixed data in 'pdays' column is []
Mixed data in 'previous' column is []
```

iii) Duplicate checking and removing

In [332…
```
bank_df.duplicated(keep='first').sum()
```

Out[332…
```
0
```

iv) Y column converting into term_deposite

In [333…
```
bank_df.rename(columns={'y':'term_deposite'},inplace=True)
```

In [334…
```
cat_col=bank_df.select_dtypes(include='object').columns
cat_col
```

Out[334…
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'poutcome', 'term_deposite'],
      dtype='object')
```

Missing Value Analysis

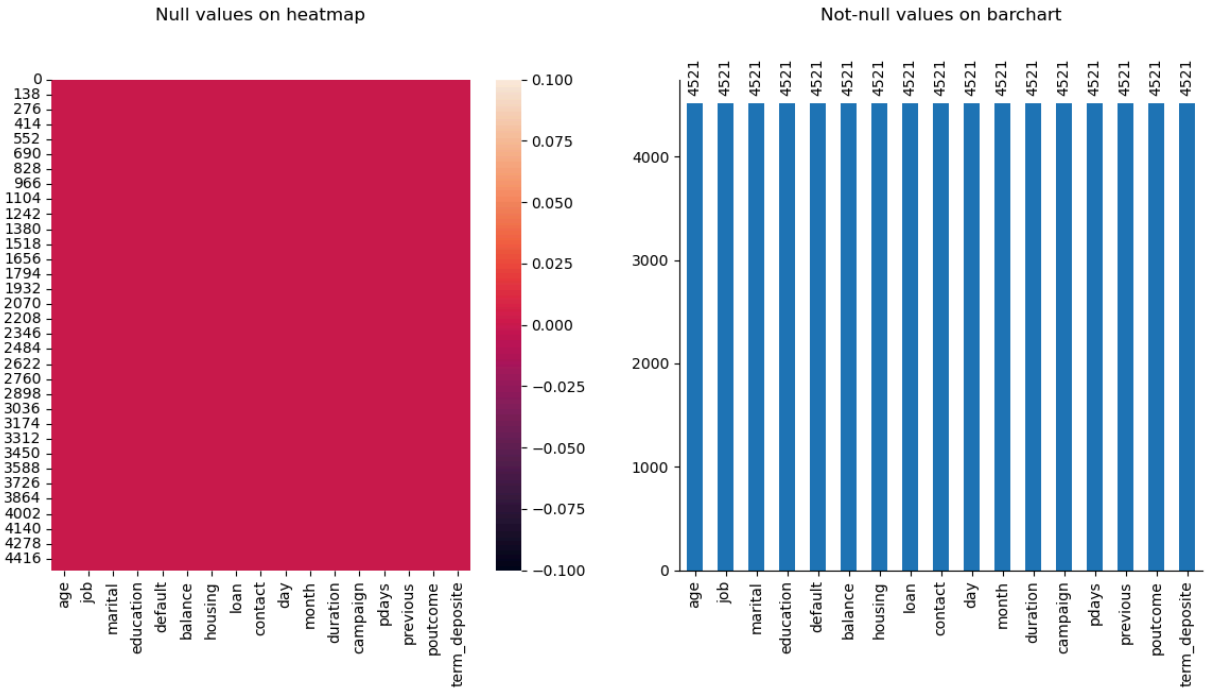
In [335…
```
# missing values count columnwise
bank_df.isnull().sum()
```

Out[335…

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

In [336…

```
# null value representation in graphical form

plt.figure(figsize=(14,6))
plt.subplot(1,2,1)

sns.heatmap(bank_df.isnull())
plt.title("Null values on heatmap",pad=40)
plt.subplot(1,2,2)

plt.title("Not-null values on barchart",pad=40)
bar=bank_df.notnull().sum().plot(kind='bar')
bar.spines['top'].set_visible(False)
bar.spines['right'].set_visible(False)
for i in bar.containers:
    bar.bar_label(i,rotation=90,padding=5)

plt.show()
```



In [337…
```
# null values count percentage columnwise
round((bank_df.isnull().sum())/len(bank_df),2)
```

Out[337…
```
age              0.0
job              0.0
marital          0.0
education        0.0
default          0.0
balance          0.0
housing          0.0
loan             0.0
contact          0.0
day              0.0
month            0.0
duration         0.0
campaign         0.0
pdays            0.0
previous         0.0
poutcome         0.0
term_deposite    0.0
dtype: float64
```

In [338…
```
# missing value overall percentage
round(bank_df.isnull().sum().sum()/bank_df.size)
```

Out[338…
```
0
```

Univariate Analysis

i) Categorical column analysis

In [339…
```
print(cat_col)
```

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

In [340…
```
# job column analysis

label=bank_df['job'].unique()
count=[]
for i in label:
    con=bank_df['job']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[340…

	Class-Name	Class-Frequency
2	management	969
3	blue collar	946
5	technician	768
7	admin	478
1	services	417
10	retired	230
4	self employed	183
6	entrepreneur	168
0	unemployed	128
9	housemaid	112
8	student	84
11	unknown	38

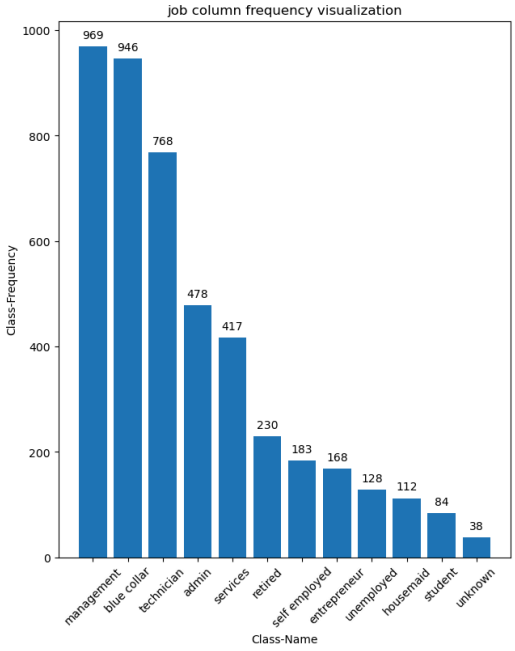
In [341…
```
label=bank_df['job'].unique()
count=[]
for i in label:
    con=bank_df['job']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

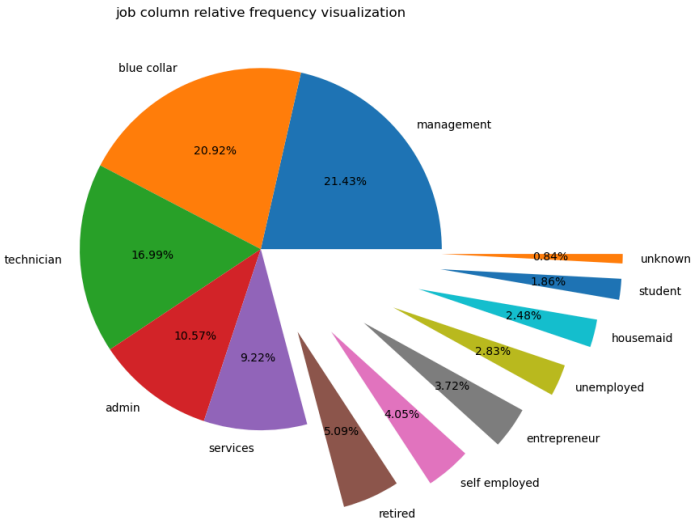
Out[341…

	Class-Name	Relative-Frequency
2	management	21.43
3	blue collar	20.92
5	technician	16.99
7	admin	10.57
1	services	9.22
10	retired	5.09
4	self employed	4.05
6	entrepreneur	3.72
0	unemployed	2.83
9	housemaid	2.48
8	student	1.86
11	unknown	0.84

In [342…

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar,padding=5)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].tick_params(axis='x', rotation=45)
ax[0].set_title("job column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("job column relative frequency visualization")

plt.show()
```





Conclucion

From Bar Chart

- Most common jobs are management, blue collar, technician. This shows, these jobs are most common in customers
- Least common jobs unknown, student, housemaid. This shows, these jobs are rare in customer
- There's a large gap between high-frequency and low-frequency job types, showing an imbalanced class distribution.

From Pie Chart

- management, blue collar, and technician together represent nearly 60% of the data, showing a dominance of these professions.
- unknown represents less than 1%, possibly missing or unclear data.
- There is a wide range of job types, indicating a diverse customer base.

In [343…

```
# marital column analysis

label=bank_df['marital'].unique()
count=[]
for i in label:
    con=bank_df['marital']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[343...

	Class-Name	Class-Frequency
0	married	2797
1	single	1196
2	divorced	528

In [344...

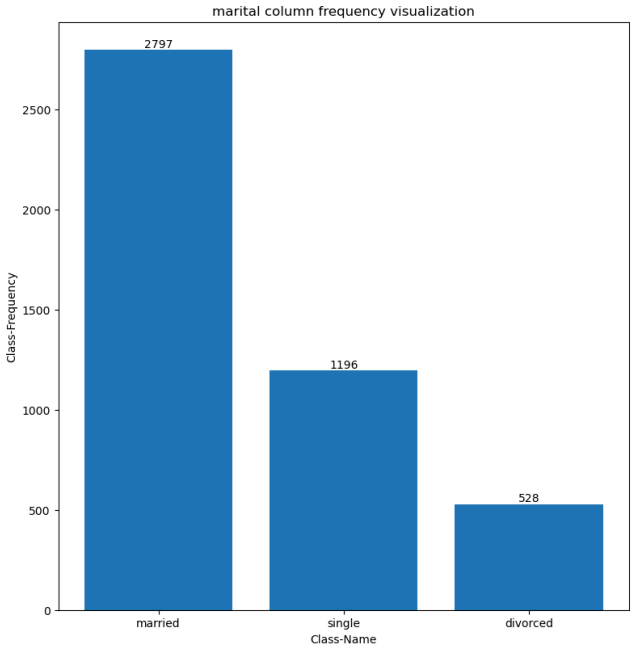
```
label=bank_df['marital'].unique()
count=[]
for i in label:
    con=bank_df['marital']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

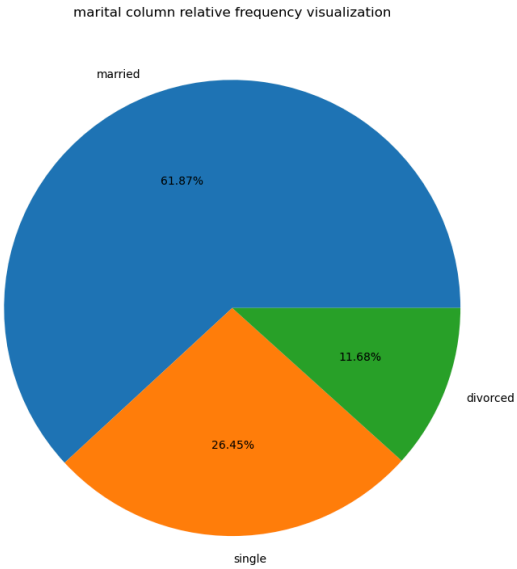
Out[344...

	Class-Name	Relative-Frequency
0	married	61.87
1	single	26.45
2	divorced	11.68

In [345...

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("marital column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("marital column relative frequency visualization")

plt.show()
```





Conclusion

From Bar Chart

- More customers are married. They are 2797 and dominating the dataset
- single is the second most frequency
- divorced is third frequency
- The column is imbalanced with married appearing more than twice of single, and over 5 times more than divorced.

From Pie Chart

- Married customer makes up 61.9% of the dataset.
- Single customer represents 26.5%, and Divorced customer is 11.7%.
- The data suggests that the majority of customers are married, which may influence financial product needs (e.g., joint accounts, loans, insurance).

In [346…

```
# education column analysis

label=bank_df['education'].unique()
count=[]
for i in label:
    con=bank_df['education']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[346...

	Class-Name	Class-Frequency
1	secondary	2306
2	tertiary	1350
0	primary	678
3	unknown	187

In [347...

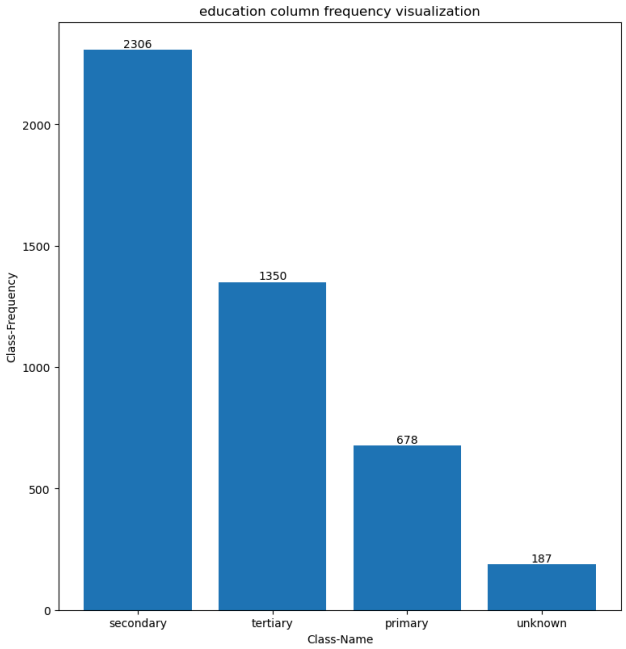
```
label=bank_df['education'].unique()
count=[]
for i in label:
    con=bank_df['education']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

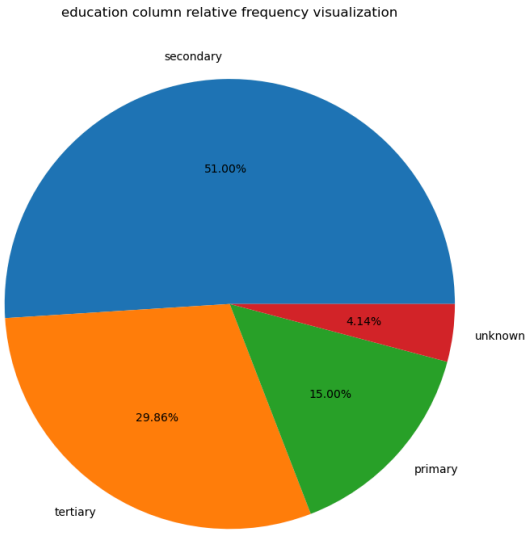
Out[347...

	Class-Name	Relative-Frequency
1	secondary	51.01
2	tertiary	29.86
0	primary	15.00
3	unknown	4.14

In [348...

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("education column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("education column relative frequency visualization")

plt.show()
```





Conclusion

From Bar Chart

- Majority cutomer education is secondary. It is highest frequency.
- Tertiary is second frequency and primary is third frequency.
- Secondary education customer is dominating dataset and it nearly two times of tertiary and three times of primary.

From Pie Chart

- Secondary education customer are 51% of dataset and they dominating the dataset.
- The unknown cutomers may be the missing data.

In [349…

```
# default column analysis

label=bank_df['default'].unique()
count=[]
for i in label:
    con=bank_df['default']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[349…

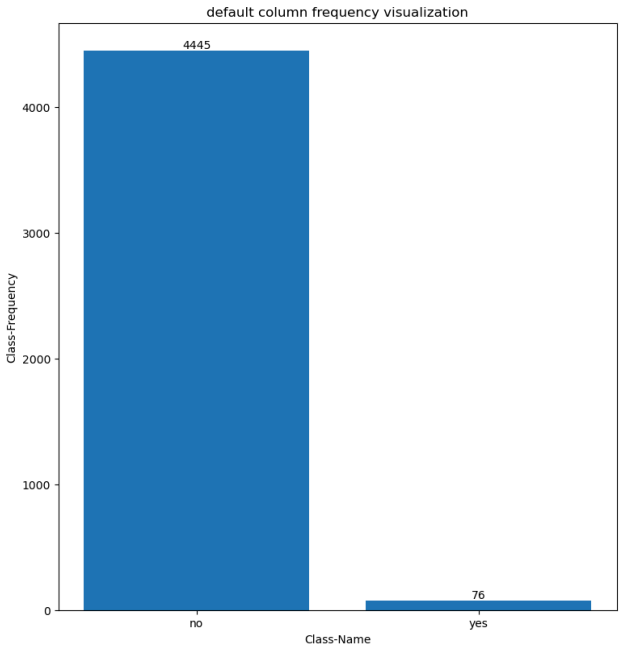
	Class-Name	Class-Frequency
0	no	4445
1	yes	76

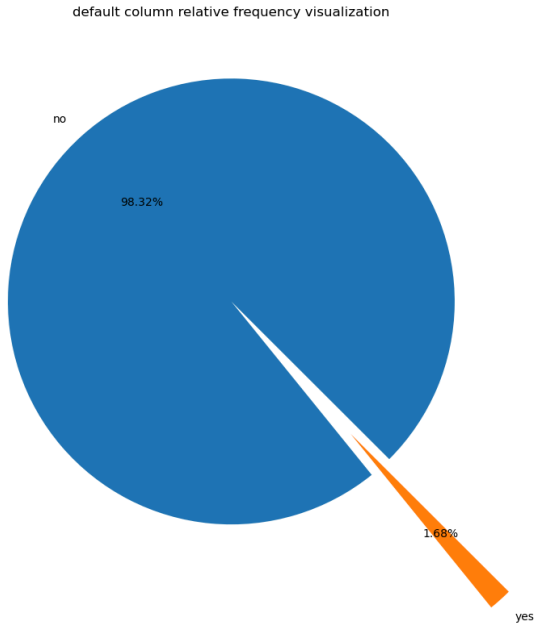
In [350…
```
label=bank_df['default'].unique()
count=[]
for i in label:
    con=bank_df['default']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

Out[350…

	Class-Name	Relative-Frequency
0	no	98.32
1	yes	1.68

In [351…
```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("default column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("default column relative frequency visualization")

plt.show()
```





conclusion

From Bar Chart

- "No" are 4,445 records. It is huge majority.

- "Yes" are Only 76 records. It is very small portion.
- This is a highly imbalanced categorical variable, with 98.3% of the data labeled as "no".

From Pie Chart

- 98.3% of customers do not have credit in default.
- Only 1.7% of customers do have credit in default.

In [352…
```
# housing column analysis

label=bank_df['housing'].unique()
count=[]
for i in label:
    con=bank_df['housing']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[352…

	Class-Name	Class-Frequency
1	yes	2559
0	no	1962

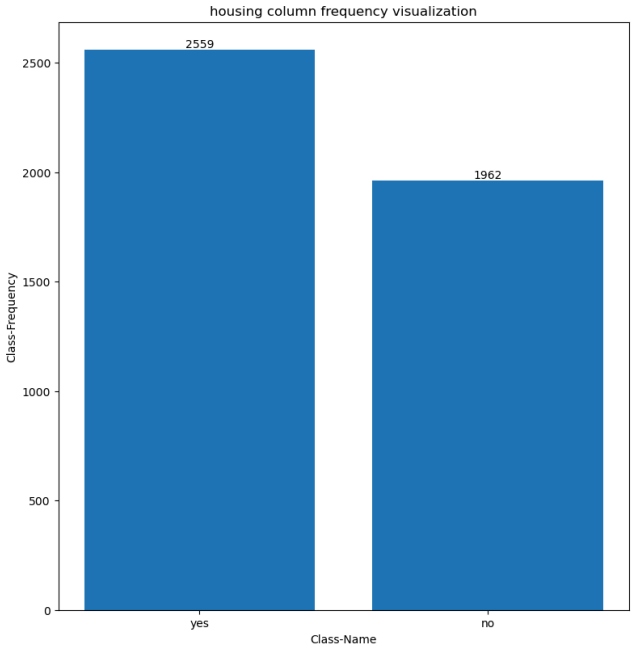
In [353…
```
label=bank_df['housing'].unique()
count=[]
for i in label:
    con=bank_df['housing']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

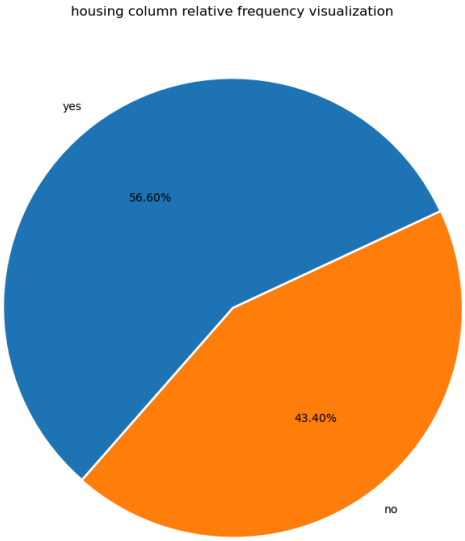
Out[353…

	Class-Name	Relative-Frequency
1	yes	56.6
0	no	43.4

In [354…
```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("housing column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("housing column relative frequency visualization")
```

```
plt.show()
```





Conclusion

From Bar Chart

- "yes": 2,559 individuals have a housing loan.
- "no": 1,962 individuals do not have a housing loan.
- The dataset shows a moderate class imbalance, with more people having housing loans than not.

From Pie Chart

- "yes": 56.6%
- "no": 43.4%
- The pie chart confirms a near-even split, but slightly more customers have a housing loan.

In [355…

```
# loan column analysis

label=bank_df['loan'].unique()
count=[]
for i in label:
    con=bank_df['loan']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[355...

	Class-Name	Class-Frequency
0	no	3830
1	yes	691

In [356...

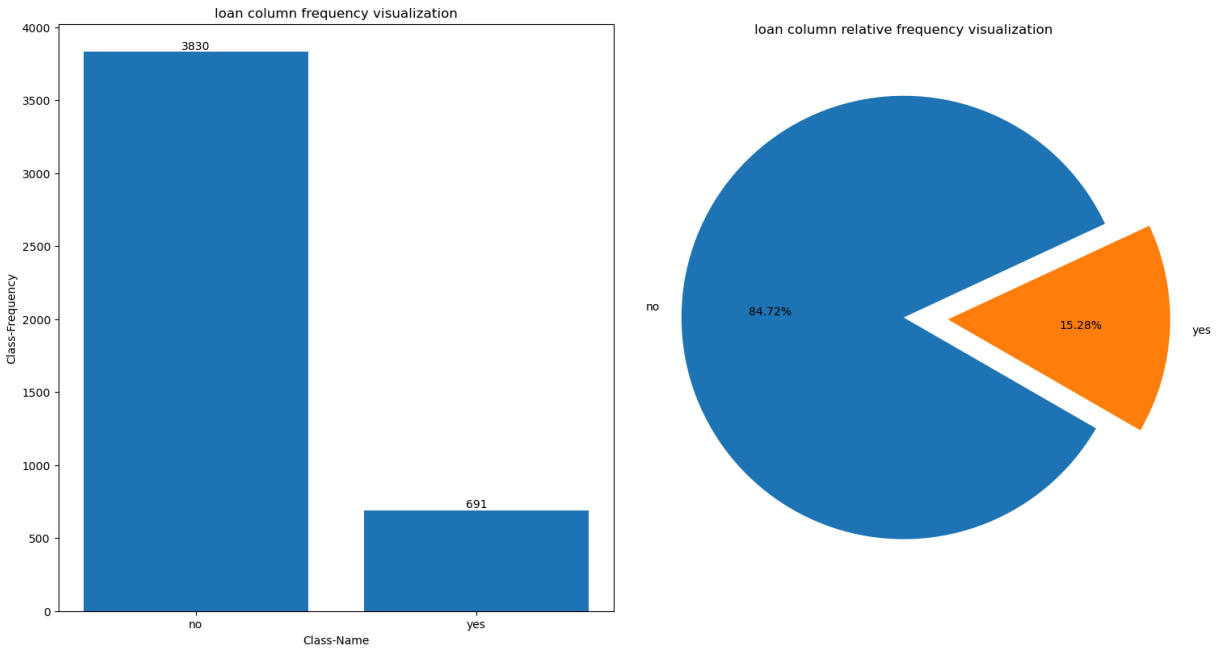
```
label=bank_df['loan'].unique()
count=[]
for i in label:
    con=bank_df['loan']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

Out[356...

	Class-Name	Relative-Frequency
0	no	84.72
1	yes	15.28

In [357...

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("loan column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("loan column relative frequency visualization")

plt.show()
```



Conclusion

From Bar Chart

- "no": 3,830 individuals do not have a personal loan.
- "yes": 691 individuals do have a personal loan.
- There is a strong class imbalance, with the majority of people not having a loan.

From Pie Chart

- "no": 84.7%
- "yes": 15.3%
- The pie chart showing few people have personal loans.

In [358…
```
# contact column analysis

label=bank_df['contact'].unique()
count=[]
for i in label:
    con=bank_df['contact']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[358…

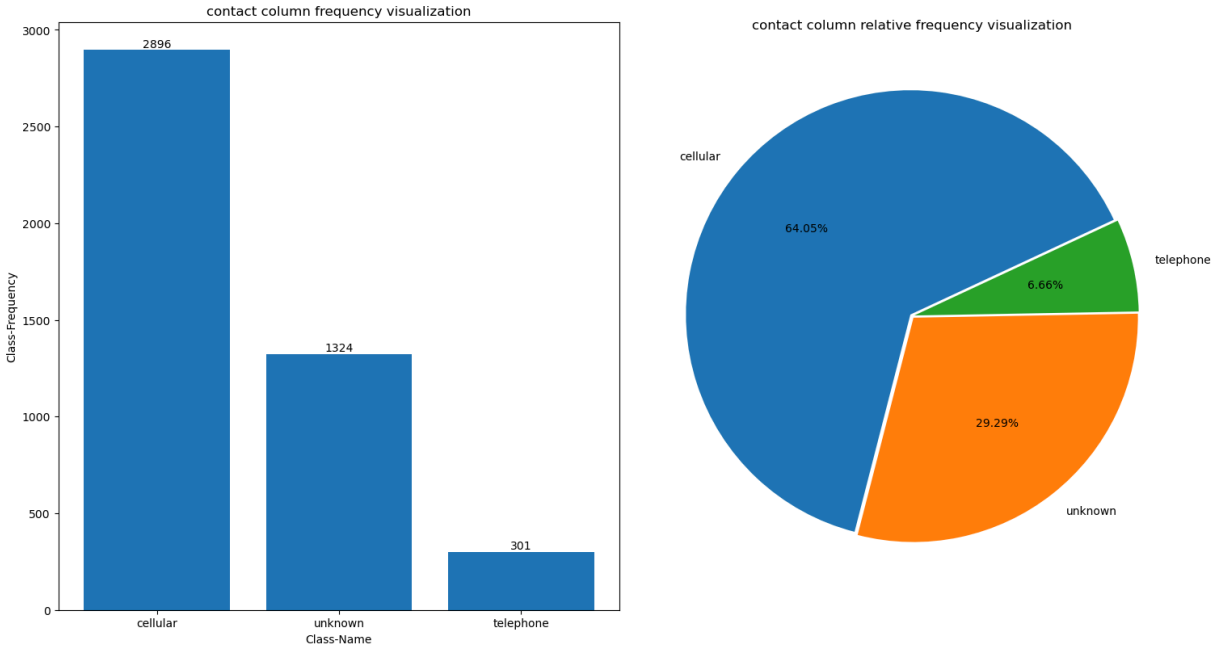
	Class-Name	Class-Frequency
0	cellular	2896
1	unknown	1324
2	telephone	301

In [359…
```
label=bank_df['contact'].unique()
count=[]
for i in label:
    con=bank_df['contact']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

Out[359…

	Class-Name	Relative-Frequency
0	cellular	64.06
1	unknown	29.29
2	telephone	6.66

In [360…
```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("contact column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("contact column relative frequency visualization")

plt.show()
```



Conclusion

From Bar Chart

- Most of the customer have cellular contact
- Telephone cutomer are very less
- There is imabalanced between cellular and telephone. cellular customers are almost ten times of telephone customer

From Pie Chart

- 64% of the dataset are cellular contact customer
- 8% peoples are telephone contact customer
- Here, unknown contact customer is more than telephone cutomer this may missing fields.

In [361…
```
# month column analysis

label=bank_df['month'].unique()
```

```
count=[]
for i in label:
    con=bank_df['month']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[361…

	Class-Name	Class-Frequency
1	may	1398
7	jul	706
5	aug	633
3	jun	531
8	nov	389
2	apr	293
4	feb	222
6	jan	148
0	oct	80
9	sep	52
10	mar	49
11	dec	20

In [362…

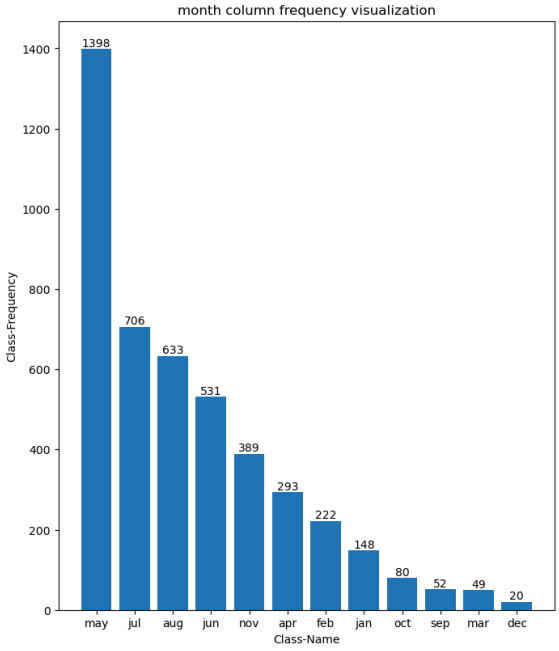
```
label=bank_df['month'].unique()
count=[]
for i in label:
    con=bank_df['month']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

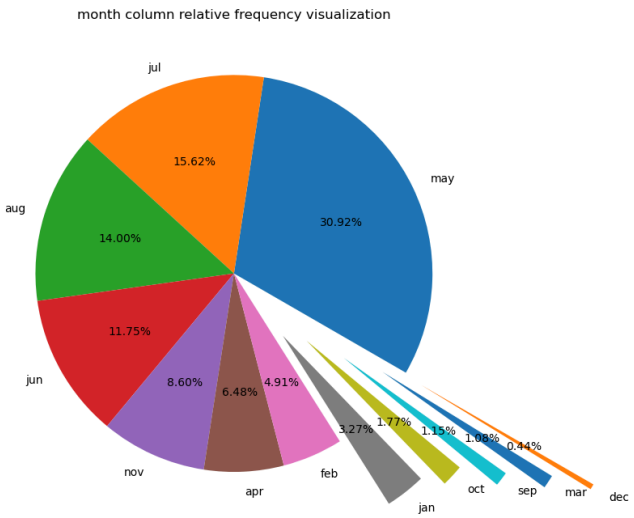
Out[362...

	Class-Name	Relative-Frequency
1	may	30.92
7	jul	15.62
5	aug	14.00
3	jun	11.75
8	nov	8.60
2	apr	6.48
4	feb	4.91
6	jan	3.27
0	oct	1.77
9	sep	1.15
10	mar	1.08
11	dec	0.44

In [363...

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("month column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("month column relative frequency visualization")

plt.show()
```





Conclusion

From Bar Chart

- Most of the term deposite happened in may month
- July month is second highest month for term deposite
- Aug, Jun, Nov are third, fourth and fifth frequent month for term deposite
- This dataset is highly imbalanced between May, Jul, Aug, Jun, Nov with other month
- May month term deposite is higher than two times of Jul and Aug month, three times of Jun, and four times of Nov.
- In december month term deposite happened very less. It may due to year end

From Pie Chart

- May, Jul, Aug and Jun months contains nearly about 70% of dataset

In [364…

```
# poutcome column analysis

label=bank_df['poutcome'].unique()
count=[]
for i in label:
    con=bank_df['poutcome']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[364...

	Class-Name	Class-Frequency
0	unknown	3705
1	failure	490
2	other	197
3	success	129

In [365...

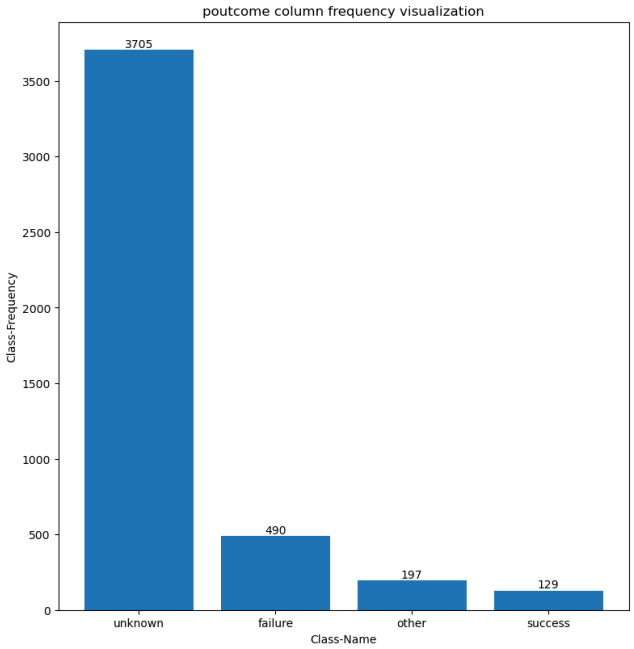
```
label=bank_df['poutcome'].unique()
count=[]
for i in label:
    con=bank_df['poutcome']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

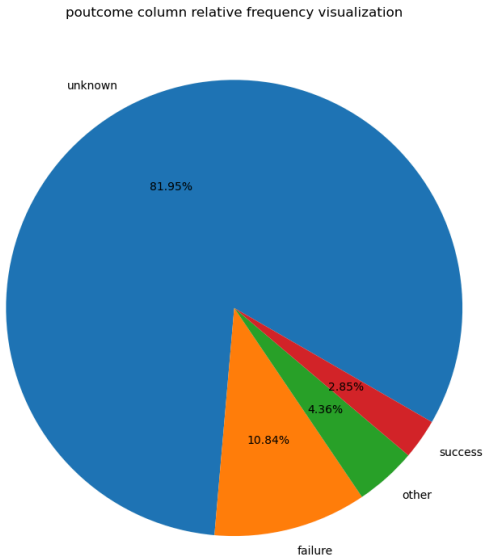
Out[365...

	Class-Name	Relative-Frequency
0	unknown	81.95
1	failure	10.84
2	other	4.36
3	success	2.85

In [366...

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("poutcome column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("poutcome column relative frequency visualization")

plt.show()
```





Conclusion

From Bar Chart

- unknown: 3705 clients (~82%) had no record of a previous campaign outcome.
- failure: 490 clients (~10.8%) were contacted in a previous campaign and did not subscribe.
- other: 197 clients (~4.4%) had a result that doesn't fall under "success" or "failure".
- success: 129 clients (~2.9%) were contacted before and did subscribe successfully.

From Pie Chart

- Visually reinforces that the majority of records (82%) have unknown outcomes, meaning those clients were either:
  - Not contacted in the previous campaign, or
  - The outcome was not recorded.
- Only a small fraction of clients (~13.7%) had a clear "failure" or "success" outcome from past efforts.

In [367…

```
# term_deposit column analysis

label=bank_df['term_deposite'].unique()
count=[]
for i in label:
    con=bank_df['term_deposite']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[367...

	Class-Name	Class-Frequency
0	no	4000
1	yes	521

In [368...

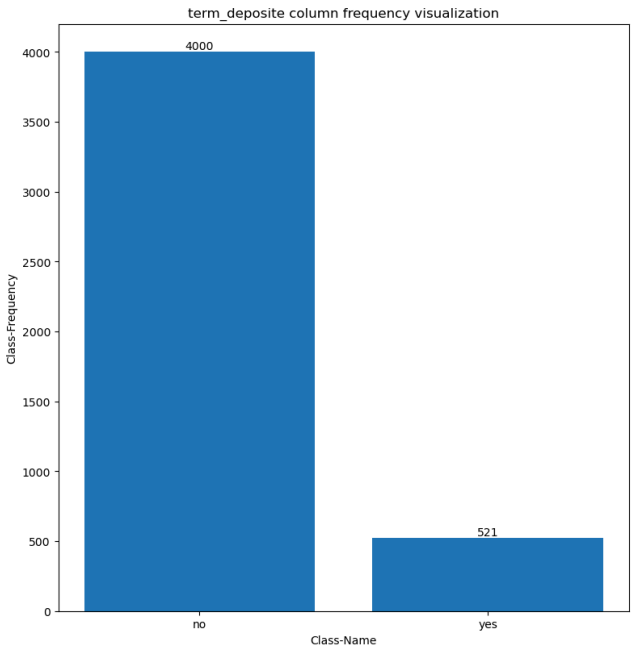
```
label=bank_df['term_deposite'].unique()
count=[]
for i in label:
    con=bank_df['term_deposite']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

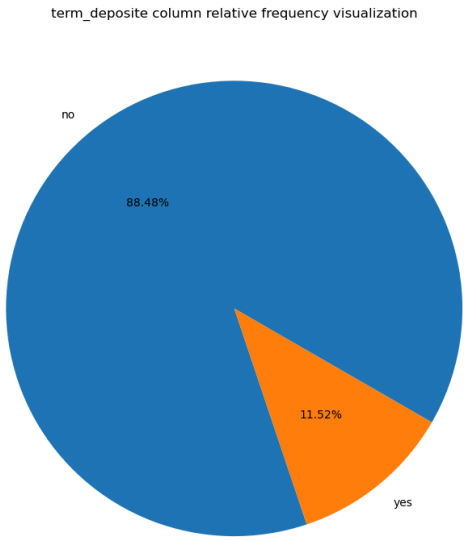
Out[368...

	Class-Name	Relative-Frequency
0	no	88.48
1	yes	11.52

In [369...

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("term_deposite column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("term_deposite column relative frequency visualization")

plt.show()
```





Conclusion

From Bar Chart

- No: 4000 customer did not invested in term deposit.
- Yes: 521 customer invested in term deposit.

From Pie Chart

- 88.5% of customer said "no" to term deposite.
- Only 11.5% of customer said "yes" to term deposite.

ii) Numerical column analysis

In [370…
```
num_col
```

Out[370…
```
['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']
```

In [371…
```
# Age column anlaysis

age=bank_df['age']
age_max=age.max()
age_min=age.min()
total_num=len(age)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((age_max-age_min)/count),2)
bins=list(np.arange(age_min,age_max+interval_gap,interval_gap))

fdt = pd.cut(age, bins=bins, right=False).value_counts().sort_index()

keys=[(i.left,i.right) for i in fdt.keys()]
value=fdt.values

df=pd.DataFrame(zip(keys,value),columns=["Class-Interval","Interval-Frequency"])
df
```

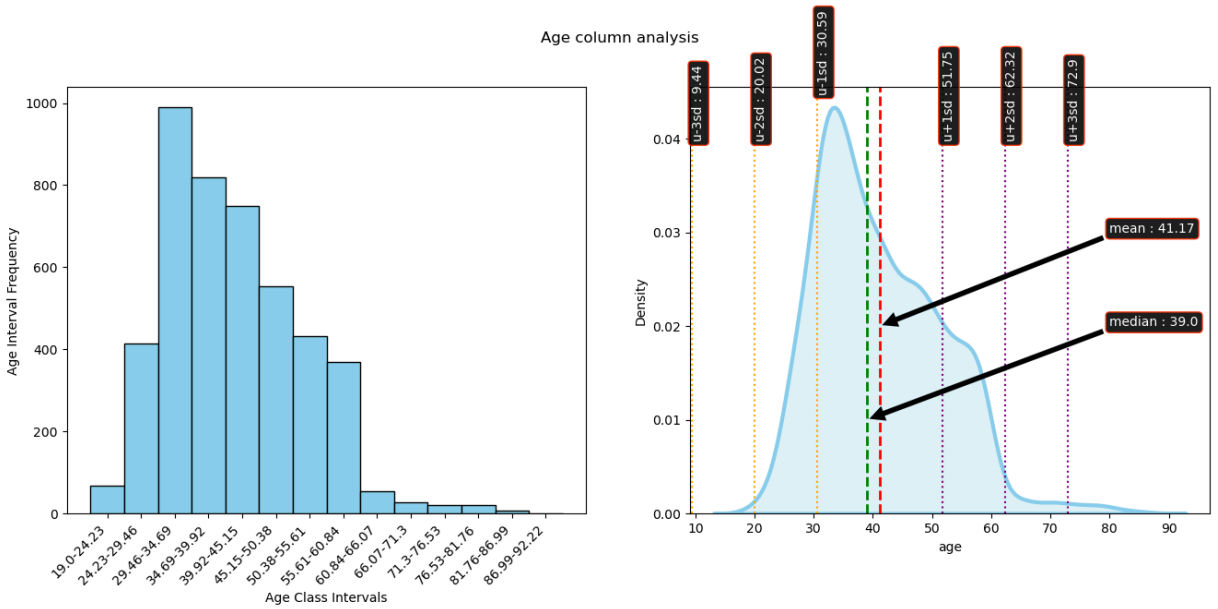
Out[371…

	Class-Interval	Interval-Frequency
0	(19.0, 24.23)	67
1	(24.23, 29.46)	415
2	(29.46, 34.69)	990
3	(34.69, 39.92)	818
4	(39.92, 45.15)	750
5	(45.15, 50.38)	554
6	(50.38, 55.61)	432
7	(55.61, 60.84)	368
8	(60.84, 66.07)	53
9	(66.07, 71.3)	26
10	(71.3, 76.53)	21
11	(76.53, 81.76)	20
12	(81.76, 86.99)	6
13	(86.99, 92.22)	1

In [372…

```
plt.figure(figsize=(16,6))
plt.suptitle("Age column analysis")

plt.subplot(1,2,1)
bin_inter=[f"{round(bins[i],2)}-{round(bins[i+1],2)}" for i in range(len(bins)-1)]
bin_centers = [(round(bins[i],2) + round(bins[i+1],2)) / 2 for i in range(len(bins)
plt.hist(age,bins=bins,color="skyblue",edgecolor='black')
plt.xticks(bin_centers,bin_inter,rotation=45,ha="right")
plt.xlabel("Age Class Intervals")
plt.ylabel("Age Interval Frequency")

plt.subplot(1,2,2)
sns.kdeplot(age,fill=True,color="skyblue",linewidth=3)

mean = age.mean()
median = age.median()
std = age.std()

plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean:.0f
plt.axvline(median, color='green', linestyle='--', linewidth=2, label=f'Median: {me

for i in range(1, 4):
    plt.axvline(mean+i*std, color='purple', linestyle=':', linewidth=1.5)
    plt.axvline(mean-i*std, color='orange', linestyle=':', linewidth=1.5)

plt.text(30.59,0.045,f"u-1sd : {round(mean-1*std,2)}",color='white',bbox=dict(alpha
plt.text(20.02,0.04,f"u-2sd : {round(mean-2*std,2)}",color='white',bbox=dict(alpha=
plt.text(9.44,0.04,f"u-3sd : {round(mean-3*std,2)}",color='white',bbox=dict(alpha=0
```

```
plt.text(51.75,0.04,f"u+1sd : {round(mean+1*std,2)}",color='white',bbox=dict(alpha=
plt.text(62.32,0.04,f"u+2sd : {round(mean+2*std,2)}",color='white',bbox=dict(alpha=
plt.text(72.9,0.04,f"u+3sd : {round(mean+3*std,2)}",color='white',bbox=dict(alpha=0

plt.annotate(f"mean : {round(mean,2)}", color='white', xy=(41.17, 0.02), xytext=(80
plt.annotate(f"median : {round(median,2)}", color='white', xy=(39, 0.01), xytext=(8

plt.show()

print(f"Mean of the data is {round(mean,2)}")
print(f"Median of the data is {round(median,2)}")
print(f"Mode of the data is {round(age.mode()[0],2)}")
print(f"68p data lies in {round(mean-1*std,2)} to {round(mean+1*std,2)}")
print(f"95p data lies in {round(mean-2*std,2)} to {round(mean+2*std,2)}")
print(f"99.7p data lies in {round(mean-3*std,2)} to {round(mean+3*std,2)}")
print(f"Minimum value is {age.min()}")
print(f"Maximum value is {age.max()}")
```




```
Mean of the data is 41.17
Median of the data is 39.0
Mode of the data is 34
68p data lies in 30.59 to 51.75
95p data lies in 20.02 to 62.32
99.7p data lies in 9.44 to 72.9
Minimum value is 19
Maximum value is 87
```

In [373…

```
print(f"Skewness is : {round(bank_df['age'].skew(),3)}")
print(f"Kurtosis is : {round(bank_df['age'].kurt(),2)}")
```

```
Skewness is : 0.7
Kurtosis is : 0.35
```

- Skewness 0.7 shows data is slightly right skewed
- Kurtosis 0.35 shows leptokutric distribution
  - high peak and more data at peak

- huge outliers at tails

Conclusion

From Histogram Graph

- Most of the customer lies in the range of 29-34 age
- Very less customers have in the range of 60-92 age
- 24-60 age cutomer contains almost 90% of the data
- Younger age customers are more interested to do the term deposit

From Distribution Graph

- The distribution of the age column is positive(Right) skewed.
- So, mean > median > mode
- mode is present at the peak.

In [374…

```
# balance column anlaysis

balance=bank_df['balance']
balance_max=balance.max()
balance_min=balance.min()
total_num=len(balance)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((balance_max-balance_min)/count),2)
bins=list(np.arange(balance_min,balance_max+interval_gap,interval_gap))

fdt = pd.cut(balance, bins=bins, right=False).value_counts().sort_index()

keys=[(i.left,i.right) for i in fdt.keys()]
value=fdt.values

df=pd.DataFrame(zip(keys,value),columns=["Class-Interval","Interval-Frequency"])
df
```

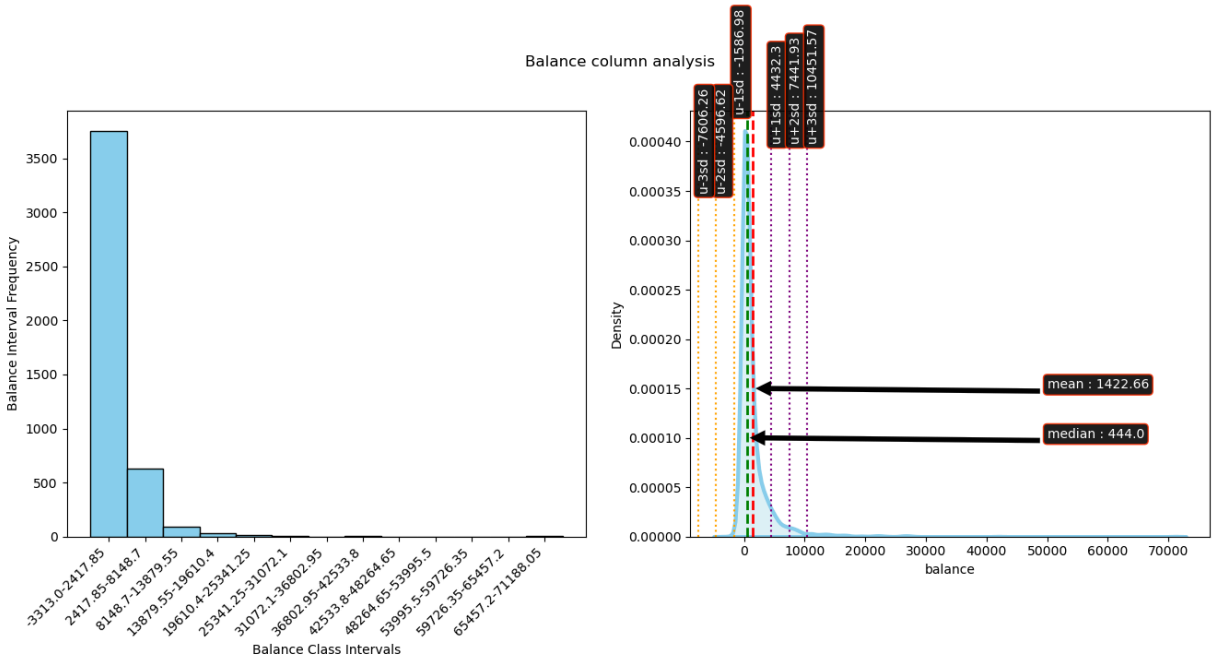
Out[374…

	Class-Interval	Interval-Frequency
0	(-3313.0, 2417.85)	3758
1	(2417.85, 8148.7)	625
2	(8148.7, 13879.55)	89
3	(13879.55, 19610.4)	29
4	(19610.4, 25341.25)	10
5	(25341.25, 31072.1)	8
6	(31072.1, 36802.95)	0
7	(36802.95, 42533.8)	1
8	(42533.8, 48264.65)	0
9	(48264.65, 53995.5)	0
10	(53995.5, 59726.35)	0
11	(59726.35, 65457.2)	0
12	(65457.2, 71188.05)	1

In [375…

```
plt.figure(figsize=(16,6))
plt.suptitle("Balance column analysis")

plt.subplot(1,2,1)
bin_inter=[f"{round(bins[i],2)}-{round(bins[i+1],2)}" for i in range(len(bins)-1)]
bin_centers = [(round(bins[i],2) + round(bins[i+1],2)) / 2 for i in range(len(bins)
plt.hist(balance,bins=bins,color="skyblue",edgecolor='black')
plt.xticks(bin_centers,bin_inter,rotation=45,ha="right")
plt.xlabel("Balance Class Intervals")
plt.ylabel("Balance Interval Frequency")

plt.subplot(1,2,2)
sns.kdeplot(balance,fill=True,color="skyblue",linewidth=3)

mean = balance.mean()
median = balance.median()
std = balance.std()

plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean:.0f
plt.axvline(median, color='green', linestyle='--', linewidth=2, label=f'Median: {me

for i in range(1, 4):
    plt.axvline(mean+i*std, color='purple', linestyle=':', linewidth=1.5)
    plt.axvline(mean-i*std, color='orange', linestyle=':', linewidth=1.5)

plt.text(-1586.98,0.00043,f"u-1sd : {round(mean-1*std,2)}",color='white',bbox=dict(
plt.text(-4596.62,0.00035,f"u-2sd : {round(mean-2*std,2)}",color='white',bbox=dict(
plt.text(-7606.26,0.00035,f"u-3sd : {round(mean-3*std,2)}",color='white',bbox=dict(

plt.text(4432.3,0.0004,f"u+1sd : {round(mean+1*std,2)}",color='white',bbox=dict(alp
```

```
plt.text(7441.93,0.0004,f"u+2sd : {round(mean+2*std,2)}",color='white',bbox=dict(al
plt.text(10451.57,0.0004,f"u+3sd : {round(mean+3*std,2)}",color='white',bbox=dict(a

plt.annotate(f"mean : {round(mean,2)}", color='white', xy=(1422.66, 0.00015), xytex
plt.annotate(f"median : {round(median,2)}", color='white', xy=(444, 0.00010), xytex


plt.show()

print(f"Mean of the data is {round(mean,2)}")
print(f"Median of the data is {round(median,2)}")
print(f"Mode of the data is {round(balance.mode()[0],2)}")
print(f"68p data lies in {round(mean-1*std,2)} to {round(mean+1*std,2)}")
print(f"95p data lies in {round(mean-2*std,2)} to {round(mean+2*std,2)}")
print(f"99.7p data lies in {round(mean-3*std,2)} to {round(mean+3*std,2)}")
print(f"Minimum value is {balance.min()}")
print(f"Maximum value is {balance.max()}")
```




```
Mean of the data is 1422.66
Median of the data is 444.0
Mode of the data is 0
68p data lies in -1586.98 to 4432.3
95p data lies in -4596.62 to 7441.93
99.7p data lies in -7606.26 to 10451.57
Minimum value is -3313
Maximum value is 71188
```

In [376…

```
print(f"Skewness is : {round(bank_df['balance'].skew(),3)}")
print(f"Kurtosis is : {round(bank_df['balance'].kurt(),2)}")
```

```
Skewness is : 6.596
Kurtosis is : 88.39
```

Conclusion

From Histogram Chart

- Most frequent balance range is -3313 to 2417 range
- This shows that lot of cutomer belongs to the middle class range

From Distribution Chart

- The distribution of the dataset is highly right(positive) skewed.
- Data density is very high at peak level and it contains the mode of the data.
- Here, Mean > Median > Mode
- Kurtosis is 88.39 so it is leptokutric distribution that shows:
  - most data is present at peak
  - tail contain more outliers
- Skewness is 6.596 that shows data is right skewed.

In [377…

```
# day column anlaysis

day=bank_df['day']
day_max=day.max()
day_min=day.min()
total_num=len(day)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((day_max-day_min)/count),2)
bins=list(np.arange(day_min,day_max+interval_gap,interval_gap))

fdt = pd.cut(day, bins=bins, right=False).value_counts().sort_index()

keys=[(i.left,i.right) for i in fdt.keys()]
value=fdt.values

df=pd.DataFrame(zip(keys,value),columns=["Class-Interval","Interval-Frequency"])
df
```

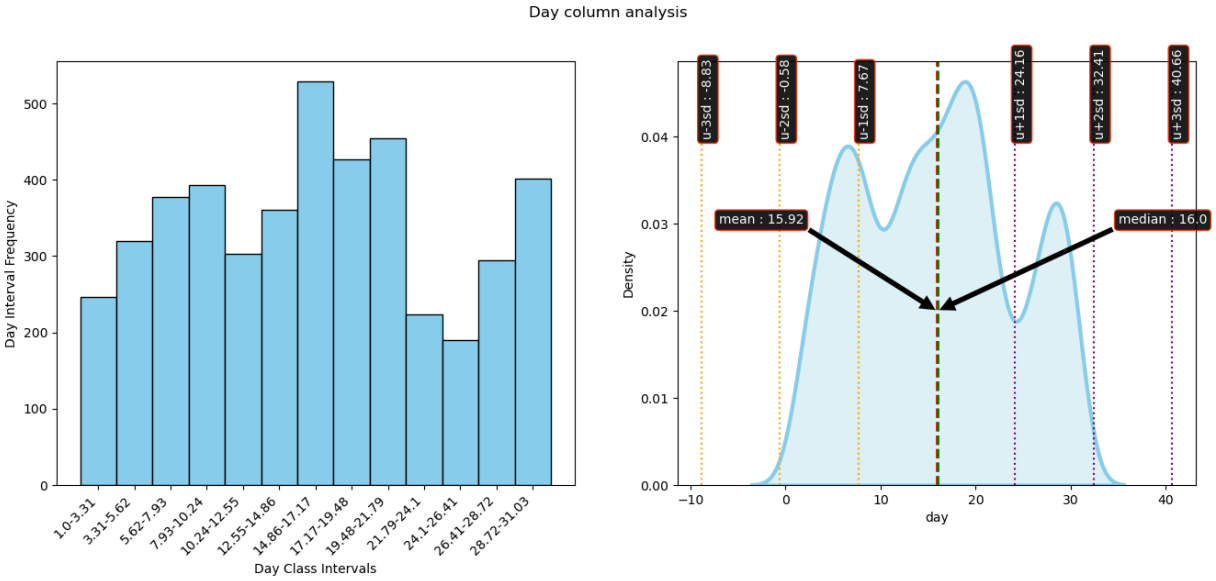
Out[377…

	Class-Interval	Interval-Frequency
0	(1.0, 3.31)	246
1	(3.31, 5.62)	320
2	(5.62, 7.93)	377
3	(7.93, 10.24)	393
4	(10.24, 12.55)	303
5	(12.55, 14.86)	361
6	(14.86, 17.17)	529
7	(17.17, 19.48)	427
8	(19.48, 21.79)	455
9	(21.79, 24.1)	224
10	(24.1, 26.41)	190
11	(26.41, 28.72)	294
12	(28.72, 31.03)	402

In [378…

```
plt.figure(figsize=(16,6))
plt.suptitle("Day column analysis")

plt.subplot(1,2,1)
bin_inter=[f"{round(bins[i],2)}-{round(bins[i+1],2)}" for i in range(len(bins)-1)]
bin_centers = [(round(bins[i],2) + round(bins[i+1],2)) / 2 for i in range(len(bins)
plt.hist(day,bins=bins,color="skyblue",edgecolor='black')
plt.xticks(bin_centers,bin_inter,rotation=45,ha='right')
plt.xlabel("Day Class Intervals")
plt.ylabel("Day Interval Frequency")

plt.subplot(1,2,2)
sns.kdeplot(day,fill=True,color="skyblue",linewidth=3)

mean = day.mean()
median = day.median()
std = day.std()

plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean:.0f
plt.axvline(median, color='green', linestyle='--', linewidth=2, label=f'Median: {me

for i in range(1, 4):
    plt.axvline(mean+i*std, color='purple', linestyle=':', linewidth=1.5)
    plt.axvline(mean-i*std, color='orange', linestyle=':', linewidth=1.5)

plt.text(7.67,0.04,f"u-1sd : {round(mean-1*std,2)}",color='white',bbox=dict(alpha=0
plt.text(-0.58,0.04,f"u-2sd : {round(mean-2*std,2)}",color='white',bbox=dict(alpha=
plt.text(-8.83,0.04,f"u-3sd : {round(mean-3*std,2)}",color='white',bbox=dict(alpha=

plt.text(24.16,0.04,f"u+1sd : {round(mean+1*std,2)}",color='white',bbox=dict(alpha=
```

```
plt.text(32.41,0.04,f"u+2sd : {round(mean+2*std,2)}",color='white',bbox=dict(alpha=
plt.text(40.66,0.04,f"u+3sd : {round(mean+3*std,2)}",color='white',bbox=dict(alpha=

plt.annotate(f"mean : {round(mean,2)}", color='white', xy=(15.92, 0.02), xytext=(-7
plt.annotate(f"median : {round(median,2)}", color='white', xy=(16.0, 0.02), xytext=

plt.show()

print(f"Mean of the data is {round(mean,2)}")
print(f"Median of the data is {round(median,2)}")
print(f"Mode of the data is {round(day.mode()[0],2)}")
print(f"68p data lies in {round(mean-1*std,2)} to {round(mean+1*std,2)}")
print(f"95p data lies in {round(mean-2*std,2)} to {round(mean+2*std,2)}")
print(f"99.7p data lies in {round(mean-3*std,2)} to {round(mean+3*std,2)}")
print(f"Minimum value is {day.min()}")
print(f"Maximum value is {day.max()}")
```




```
Mean of the data is 15.92
Median of the data is 16.0
Mode of the data is 20
68p data lies in 7.67 to 24.16
95p data lies in -0.58 to 32.41
99.7p data lies in -8.83 to 40.66
Minimum value is 1
Maximum value is 31
```

In [379…
```
round(bank_df['day'].skew(),3)
```

Out[379…
```
0.095
```

In [380…
```
round(bank_df['day'].kurt(),2)
```

Out[380…
```
-1.04
```

Conclusion

From Histogram Chart

- Between 14 to 17 day of the month bank calling to the maximum customer for term/fixed deposite.
- After every three days bank is calling more than 150 customers.
- Bank is calling to each and every customer at least one time in the month.
- Bank is calling to maximum customer in 14 to 21 dats of the month. These customer may

have financially strong.

- At the end of month bank again calling to maximum customers.

From Distribution Chart

- The data is not skewed to any side, we can say it is closely to bell curve But, it contains multiple peaks.
- A value of skewness is 0.095. It means the data is very slightly positively skewed, but it's almost symmetric.
- Negative kurtosis (-1.04) means the distribution is platykurtic:
  - Flatter peak than normal.
  - Lighter tails and fewer outliers.

In [381…

```
# Duration column anlaysis

duration=bank_df['duration']
duration_max=duration.max()
duration_min=duration.min()
total_num=len(duration)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((duration_max-duration_min)/count),2)
bins=list(np.arange(duration_min,duration_max+interval_gap,interval_gap))

fdt = pd.cut(duration, bins=bins, right=False).value_counts().sort_index()

keys=[(i.left,i.right) for i in fdt.keys()]
value=fdt.values

df=pd.DataFrame(zip(keys,value),columns=["Class-Interval","Interval-Frequency"])
df
```

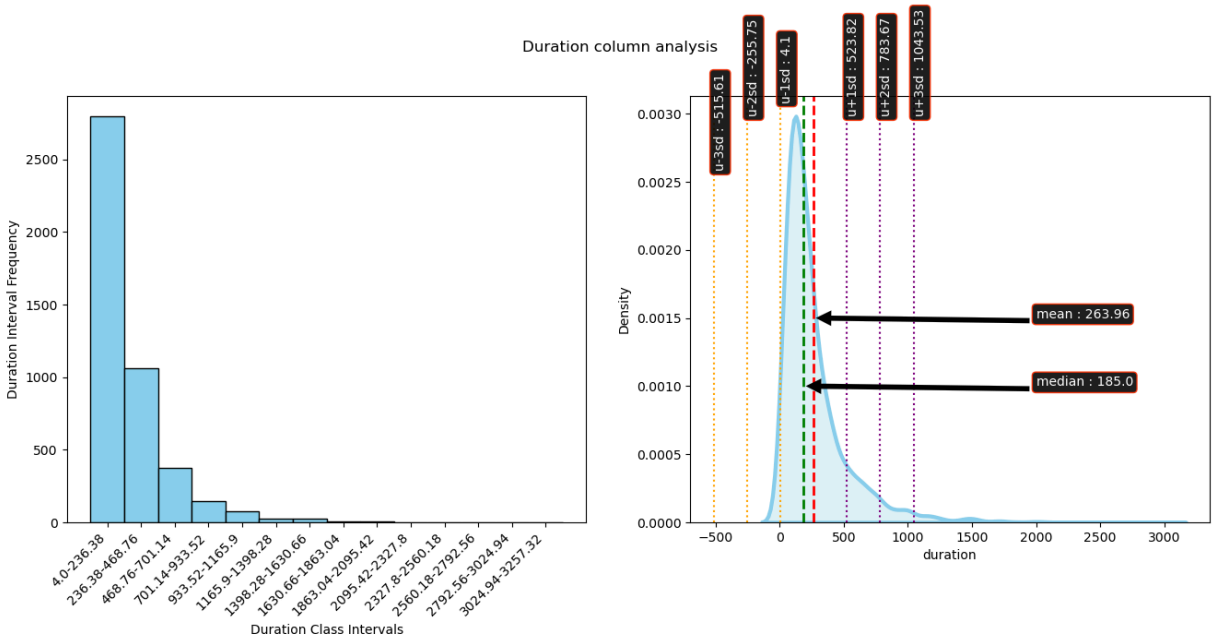
Out[381…

	Class-Interval	Interval-Frequency
0	(4.0, 236.38)	2798
1	(236.38, 468.76)	1062
2	(468.76, 701.14)	376
3	(701.14, 933.52)	148
4	(933.52, 1165.9)	74
5	(1165.9, 1398.28)	26
6	(1398.28, 1630.66)	22
7	(1630.66, 1863.04)	7
8	(1863.04, 2095.42)	5
9	(2095.42, 2327.8)	0
10	(2327.8, 2560.18)	1
11	(2560.18, 2792.56)	1
12	(2792.56, 3024.94)	0
13	(3024.94, 3257.32)	1

In [382…

```
plt.figure(figsize=(16,6))
plt.suptitle("Duration column analysis")

plt.subplot(1,2,1)
bin_inter=[f"{round(bins[i],2)}-{round(bins[i+1],2)}" for i in range(len(bins)-1)]
bin_centers = [(round(bins[i],2) + round(bins[i+1],2)) / 2 for i in range(len(bins)
plt.hist(duration,bins=bins,color="skyblue",edgecolor='black')
plt.xticks(bin_centers,bin_inter,rotation=45,ha="right")
plt.xlabel("Duration Class Intervals")
plt.ylabel("Duration Interval Frequency")

plt.subplot(1,2,2)
sns.kdeplot(duration,fill=True,color="skyblue",linewidth=3)

mean = duration.mean()
median = duration.median()
std = duration.std()

plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean:.0f
plt.axvline(median, color='green', linestyle='--', linewidth=2, label=f'Median: {me

for i in range(1, 4):
    plt.axvline(mean+i*std, color='purple', linestyle=':', linewidth=1.5)
    plt.axvline(mean-i*std, color='orange', linestyle=':', linewidth=1.5)

plt.text(4.1,0.0031,f"u-1sd : {round(mean-1*std,2)}",color='white',bbox=dict(alpha=
plt.text(-255.75,0.0030,f"u-2sd : {round(mean-2*std,2)}",color='white',bbox=dict(al
plt.text(-515.61,0.0026,f"u-3sd : {round(mean-3*std,2)}",color='white',bbox=dict(al
```

```
plt.text(523.82,0.0030,f"u+1sd : {round(mean+1*std,2)}",color='white',bbox=dict(alp
plt.text(783.67,0.0030,f"u+2sd : {round(mean+2*std,2)}",color='white',bbox=dict(alp
plt.text(1043.53,0.0030,f"u+3sd : {round(mean+3*std,2)}",color='white',bbox=dict(al

plt.annotate(f"mean : {round(mean,2)}", color='white', xy=(263.96, 0.0015), xytext=
plt.annotate(f"median : {round(median,2)}", color='white', xy=(185.0, 0.0010), xyte


plt.show()

print(f"Mean of the data is {round(mean,2)}")
print(f"Median of the data is {round(median,2)}")
print(f"Mode of the data is {round(duration.mode()[0],2)}")
print(f"68p data lies in {round(mean-1*std,2)} to {round(mean+1*std,2)}")
print(f"95p data lies in {round(mean-2*std,2)} to {round(mean+2*std,2)}")
print(f"99.7p data lies in {round(mean-3*std,2)} to {round(mean+3*std,2)}")
print(f"Minimum value is {duration.min()}")
print(f"Maximum value is {duration.max()}")
```




```
Mean of the data is 263.96
Median of the data is 185.0
Mode of the data is 123
68p data lies in 4.1 to 523.82
95p data lies in -255.75 to 783.67
99.7p data lies in -515.61 to 1043.53
Minimum value is 4
Maximum value is 3025
```

In [383…

```
print(f"Skewness is : {round(bank_df['duration'].skew(),3)}")
print(f"Kurtosis is : {round(bank_df['duration'].kurt(),2)}")
```

```
Skewness is : 2.772
Kurtosis is : 12.53
```

Above line shows

- Skewness is 2.772 shows positive(right) skewed distribution of the data.
- Kurtosis is 12.53 shows it is Leptokutric distribution.
  - So it have high pick than normal distribution long tails.
  - Dense outliers present in the tail.

conclusion

From Histogram Graph

- The most of the last calls with client happen 4 to 236 second range, It means talking happened with client is less than 5 minutes. It tells that most of the clients are not interested term deposite.
- from 468 to 3257 seconds call happens very less. These customers may show interest in term deposite

From Distribution Graph

- The distribution of the duration of calls in seconds is Right (Positive) skewed.
- It contain very huge right tail that shows very less call happens for longer duration
- So, Mean > Median > Mode

In [384…

```
# Campaign column anlaysis

campaign=bank_df['campaign']
campaign_max=campaign.max()
campaign_min=campaign.min()
total_num=len(campaign)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((campaign_max-campaign_min)/count),2)
bins=list(np.arange(campaign_min,campaign_max+interval_gap,interval_gap))

fdt = pd.cut(campaign, bins=bins, right=False).value_counts().sort_index()

keys=[(i.left,i.right) for i in fdt.keys()]
value=fdt.values

df=pd.DataFrame(zip(keys,value),columns=["Class-Interval","Interval-Frequency"])
df
```

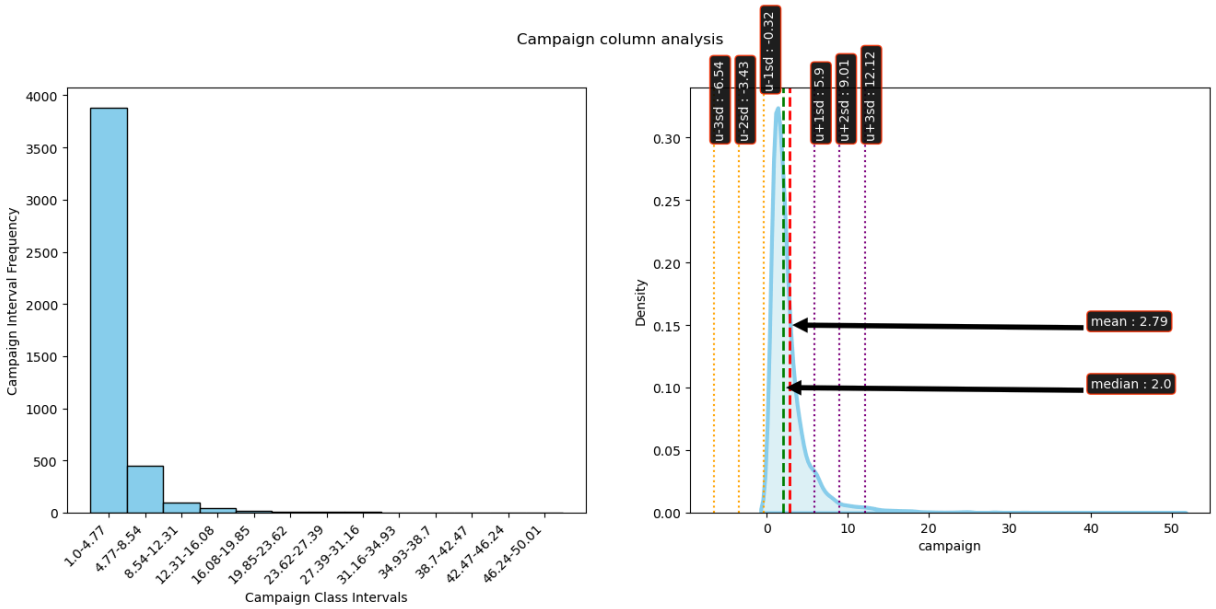
Out[384...

	Class-Interval	Interval-Frequency
0	(1.0, 4.77)	3881
1	(4.77, 8.54)	453
2	(8.54, 12.31)	100
3	(12.31, 16.08)	44
4	(16.08, 19.85)	17
5	(19.85, 23.62)	9
6	(23.62, 27.39)	7
7	(27.39, 31.16)	6
8	(31.16, 34.93)	2
9	(34.93, 38.7)	0
10	(38.7, 42.47)	0
11	(42.47, 46.24)	1
12	(46.24, 50.01)	1

In [385...

```
plt.figure(figsize=(16,6))
plt.suptitle("Campaign column analysis")

plt.subplot(1,2,1)
bin_inter=[f"{round(bins[i],2)}-{round(bins[i+1],2)}" for i in range(len(bins)-1)]
bin_centers = [(round(bins[i],2) + round(bins[i+1],2)) / 2 for i in range(len(bins)
plt.hist(campaign,bins=bins,color="skyblue",edgecolor='black')
plt.xticks(bin_centers,bin_inter,rotation=45,ha="right")
plt.xlabel("Campaign Class Intervals")
plt.ylabel("Campaign Interval Frequency")

plt.subplot(1,2,2)
sns.kdeplot(campaign,fill=True,color="skyblue",linewidth=3)

mean = campaign.mean()
median = campaign.median()
std = campaign.std()

plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean:.0f
plt.axvline(median, color='green', linestyle='--', linewidth=2, label=f'Median: {me

for i in range(1, 4):
    plt.axvline(mean+i*std, color='purple', linestyle=':', linewidth=1.5)
    plt.axvline(mean-i*std, color='orange', linestyle=':', linewidth=1.5)

plt.text(-0.32,0.34,f"u-1sd : {round(mean-1*std,2)}",color='white',bbox=dict(alpha=
plt.text(-3.43,0.30,f"u-2sd : {round(mean-2*std,2)}",color='white',bbox=dict(alpha=
plt.text(-6.54,0.30,f"u-3sd : {round(mean-3*std,2)}",color='white',bbox=dict(alpha=

plt.text(5.9,0.30,f"u+1sd : {round(mean+1*std,2)}",color='white',bbox=dict(alpha=0.
```

```
plt.text(9.01,0.30,f"u+2sd : {round(mean+2*std,2)}",color='white',bbox=dict(alpha=0
plt.text(12.12,0.30,f"u+3sd : {round(mean+3*std,2)}",color='white',bbox=dict(alpha=

plt.annotate(f"mean : {round(mean,2)}", color='white', xy=(2.79, 0.15), xytext=(40,
plt.annotate(f"median : {round(median,2)}", color='white', xy=(2, 0.10), xytext=(40

plt.show()

print(f"Mean of the data is {round(mean,2)}")
print(f"Median of the data is {round(median,2)}")
print(f"Mode of the data is {round(campaign.mode()[0],2)}")
print(f"68p data lies in {round(mean-1*std,2)} to {round(mean+1*std,2)}")
print(f"95p data lies in {round(mean-2*std,2)} to {round(mean+2*std,2)}")
print(f"99.7p data lies in {round(mean-3*std,2)} to {round(mean+3*std,2)}")
print(f"Minimum value is {campaign.min()}")
print(f"Maximum value is {campaign.max()}")
```




```
Mean of the data is 2.79
Median of the data is 2.0
Mode of the data is 1
68p data lies in -0.32 to 5.9
95p data lies in -3.43 to 9.01
99.7p data lies in -6.54 to 12.12
Minimum value is 1
Maximum value is 50
```

In [386…

```
print(f"Skewness is : {round(bank_df['campaign'].skew(),2)}")
print(f"Kurtosis is : {round(bank_df['campaign'].kurt(),2)}")
```

```
Skewness is : 4.74
Kurtosis is : 37.17
```

Above line shows

- Skewness is 4.74 that shows the data is right(positive) skewed.
- Kurtosis is 12.53 shows it is Leptokutric distribution.
  - So it have high pick than normal distribution long tails.

- Dense outliers present in the tail.

Conclusion

From Histogram Graph

- Most people were contacted 1 to 4 times to the bank after campaign.
- Very few people were contacted more than 12 times to the bank for term deposite.

From Distribution Graph

- The distribution of campaign column is strong right(positive) skewed.
- so, Mean > Median > Mode
- Mode is present at the peak.
- The mean 2.79 shows on average each customer was contacted about 2.79 times to the bank.
- Median 2 shows half customers contacted to bank 2 or less than 2 times.

In [387…

```
# Pdays column anlaysis

pdays=bank_df['pdays']
pdays_max=pdays.max()
pdays_min=pdays.min()
total_num=len(pdays)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((pdays_max-pdays_min)/count),2)
bins=list(np.arange(pdays_min,pdays_max+interval_gap,interval_gap))

fdt = pd.cut(pdays, bins=bins, right=False).value_counts().sort_index()

keys=[(i.left,i.right) for i in fdt.keys()]
value=fdt.values

df=pd.DataFrame(zip(keys,value),columns=["Class-Interval","Interval-Frequency"])
df
```

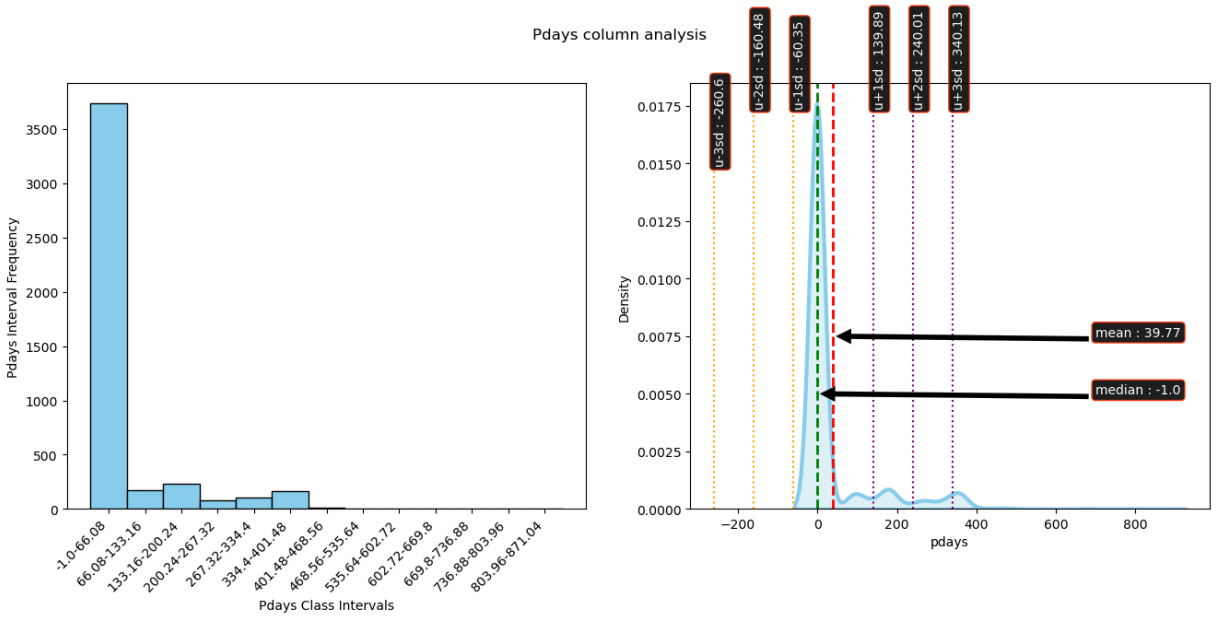
Out[387…

	Class-Interval	Interval-Frequency
0	(-1.0, 66.08)	3732
1	(66.08, 133.16)	174
2	(133.16, 200.24)	233
3	(200.24, 267.32)	82
4	(267.32, 334.4)	109
5	(334.4, 401.48)	165
6	(401.48, 468.56)	13
7	(468.56, 535.64)	5
8	(535.64, 602.72)	1
9	(602.72, 669.8)	0
10	(669.8, 736.88)	3
11	(736.88, 803.96)	1
12	(803.96, 871.04)	3

In [388…

```
plt.figure(figsize=(16,6))
plt.suptitle("Pdays column analysis")

plt.subplot(1,2,1)
bin_inter=[f"{round(bins[i],2)}-{round(bins[i+1],2)}" for i in range(len(bins)-1)]
bin_centers = [(round(bins[i],2) + round(bins[i+1],2)) / 2 for i in range(len(bins)
plt.hist(pdays,bins=bins,color="skyblue",edgecolor='black')
plt.xticks(bin_centers,bin_inter,rotation=45,ha='right')
plt.xlabel("Pdays Class Intervals")
plt.ylabel("Pdays Interval Frequency")

plt.subplot(1,2,2)
sns.kdeplot(pdays,fill=True,color="skyblue",linewidth=3)

mean = pdays.mean()
median = pdays.median()
std = pdays.std()

plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean:.0f
plt.axvline(median, color='green', linestyle='--', linewidth=2, label=f'Median: {me

for i in range(1, 4):
    plt.axvline(mean+i*std, color='purple', linestyle=':', linewidth=1.5)
    plt.axvline(mean-i*std, color='orange', linestyle=':', linewidth=1.5)

plt.text(-60.35,0.0175,f"u-1sd : {round(mean-1*std,2)}",color='white',bbox=dict(alp
plt.text(-160.48,0.0175,f"u-2sd : {round(mean-2*std,2)}",color='white',bbox=dict(al
plt.text(-260.6,0.0150,f"u-3sd : {round(mean-3*std,2)}",color='white',bbox=dict(alp

plt.text(139.89,0.0175,f"u+1sd : {round(mean+1*std,2)}",color='white',bbox=dict(alp
```

```
plt.text(240.01,0.0175,f"u+2sd : {round(mean+2*std,2)}",color='white',bbox=dict(alp
plt.text(340.13,0.0175,f"u+3sd : {round(mean+3*std,2)}",color='white',bbox=dict(alp

plt.annotate(f"mean : {round(mean,2)}", color='white', xy=(39.77, 0.0075), xytext=(
plt.annotate(f"median : {round(median,2)}", color='white', xy=(-1.0, 0.0050), xytex

plt.show()

print(f"Mean of the data is {round(mean,2)}")
print(f"Median of the data is {round(median,2)}")
print(f"Mode of the data is {round(pdays.mode()[0],2)}")
print(f"68p data lies in {round(mean-1*std,2)} to {round(mean+1*std,2)}")
print(f"95p data lies in {round(mean-2*std,2)} to {round(mean+2*std,2)}")
print(f"99.7p data lies in {round(mean-3*std,2)} to {round(mean+3*std,2)}")
print(f"Minimum value is {pdays.min()}")
print(f"Maximum value is {pdays.max()}")
```




```
Mean of the data is 39.77
Median of the data is -1.0
Mode of the data is -1
68p data lies in -60.35 to 139.89
95p data lies in -160.48 to 240.01
99.7p data lies in -260.6 to 340.13
Minimum value is -1
Maximum value is 871
```

In [389…

```
print(f"Skewness is : {round(bank_df['campaign'].skew(),2)}")
print(f"Kurtosis is : {round(bank_df['campaign'].kurt(),2)}")
```

```
Skewness is : 4.74
Kurtosis is : 37.17
```

From above lines

- Skewness is 4.74, shows the distribution of the data is highly right(positive) side.
- Kurtosis is 37.17 shows it is Leptokutric distribution.
  - So it have high pick than normal distribution long tails.

- Dense outliers present in the tail.

Conclusion

From Histogram Graph

- Majority of values are around -1, that indicates most clients were never contacted before to the bank before campaign
- Very few clients were contacted before, as the frequency sharply drops in the higher intervals.

From Distribution Graph

- The pdays column data distribution is right(positive) skewed.
- Median is -1 idicates half of the customers do not contacted to the bank before the campaign.
- Here, Mean > Median > Mode due to right skew.

In [390…

```
# Previous column anlaysis

previous=bank_df['previous']
previous_max=previous.max()
previous_min=previous.min()
total_num=len(previous)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((previous_max-previous_min)/count),2)
bins=list(np.arange(previous_min,previous_max+interval_gap,interval_gap))

fdt = pd.cut(previous, bins=bins, right=False).value_counts().sort_index()

keys=[(i.left,i.right) for i in fdt.keys()]
value=fdt.values

df=pd.DataFrame(zip(keys,value),columns=["Class-Interval","Interval-Frequency"])
df
```

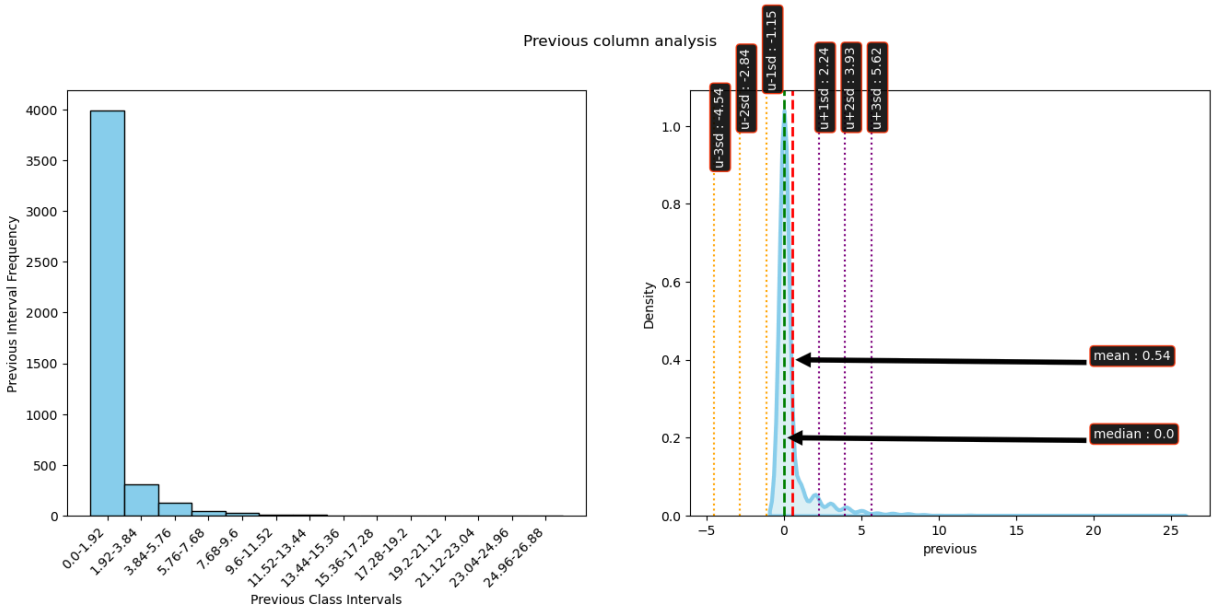
Out[390…

	Class-Interval	Interval-Frequency
0	(0.0, 1.92)	3991
1	(1.92, 3.84)	306
2	(3.84, 5.76)	125
3	(5.76, 7.68)	47
4	(7.68, 9.6)	28
5	(9.6, 11.52)	7
6	(11.52, 13.44)	6
7	(13.44, 15.36)	3
8	(15.36, 17.28)	1
9	(17.28, 19.2)	2
10	(19.2, 21.12)	1
11	(21.12, 23.04)	2
12	(23.04, 24.96)	1
13	(24.96, 26.88)	1

In [391…

```
plt.figure(figsize=(16,6))
plt.suptitle("Previous column analysis")

plt.subplot(1,2,1)
bin_inter=[f"{round(bins[i],2)}-{round(bins[i+1],2)}" for i in range(len(bins)-1)]
bin_centers = [(round(bins[i],2) + round(bins[i+1],2)) / 2 for i in range(len(bins)
plt.hist(previous,bins=bins,color="skyblue",edgecolor='black')
plt.xticks(bin_centers,bin_inter,rotation=45,ha='right')
plt.xlabel("Previous Class Intervals")
plt.ylabel("Previous Interval Frequency")

plt.subplot(1,2,2)
sns.kdeplot(previous,fill=True,color="skyblue",linewidth=3)

mean = previous.mean()
median = previous.median()
std = previous.std()

plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean:.0f
plt.axvline(median, color='green', linestyle='--', linewidth=2, label=f'Median: {me

for i in range(1, 4):
    plt.axvline(mean+i*std, color='purple', linestyle=':', linewidth=1.5)
    plt.axvline(mean-i*std, color='orange', linestyle=':', linewidth=1.5)

plt.text(-1.15,1.1,f"u-1sd : {round(mean-1*std,2)}",color='white',bbox=dict(alpha=0
plt.text(-2.84,1,f"u-2sd : {round(mean-2*std,2)}",color='white',bbox=dict(alpha=0.9
plt.text(-4.54,0.9,f"u-3sd : {round(mean-3*std,2)}",color='white',bbox=dict(alpha=0
```

```
plt.text(2.24,1,f"u+1sd : {round(mean+1*std,2)}",color='white',bbox=dict(alpha=0.9,
plt.text(3.93,1,f"u+2sd : {round(mean+2*std,2)}",color='white',bbox=dict(alpha=0.9,
plt.text(5.62,1,f"u+3sd : {round(mean+3*std,2)}",color='white',bbox=dict(alpha=0.9,

plt.annotate(f"mean : {round(mean,2)}", color='white', xy=(0.54, 0.4), xytext=(20,
plt.annotate(f"median : {round(median,2)}", color='white', xy=(0.0, 0.2), xytext=(2

plt.show()

print(f"Mean of the data is {round(mean,2)}")
print(f"Median of the data is {round(median,2)}")
print(f"Mode of the data is {round(previous.mode()[0],2)}")
print(f"68p data lies in {round(mean-1*std,2)} to {round(mean+1*std,2)}")
print(f"95p data lies in {round(mean-2*std,2)} to {round(mean+2*std,2)}")
print(f"99.7p data lies in {round(mean-3*std,2)} to {round(mean+3*std,2)}")
print(f"Minimum value is {previous.min()}")
print(f"Maximum value is {previous.max()}")
```




```
Mean of the data is 0.54
Median of the data is 0.0
Mode of the data is 0
68p data lies in -1.15 to 2.24
95p data lies in -2.84 to 3.93
99.7p data lies in -4.54 to 5.62
Minimum value is 0
Maximum value is 25
```

In [392…

```
print(f"Skewness is : {round(bank_df['previous'].skew(),2)}")
print(f"Kurtosis is : {round(bank_df['previous'].kurt(),2)}")
```

```
Skewness is : 5.88
Kurtosis is : 52.0
```

From above lines

- Skewness is 5.88 the data is stongly right(positive) skewed.
- Kurtosis is 52 shows it is Leptokutric distribution.

- So it have very high pick than normal distribution and have very long tails.
- Dense outliers present in the tail.

Conclusion

From Histogram Graph

- More than 90% customers are contacted to the bank at 0-2 time before the campaign
- Very few cutomers are contacted to the bank more than two time, that show very few peoples was interested to the term deposite before campaign.

From Distribution Graph

- Graph distribution is on right(positive) side.
- so, Mean > Median > Mode
- Median is 0 indicates 50P peoples never contacted to the bank before campaign.

Outlier Analysis

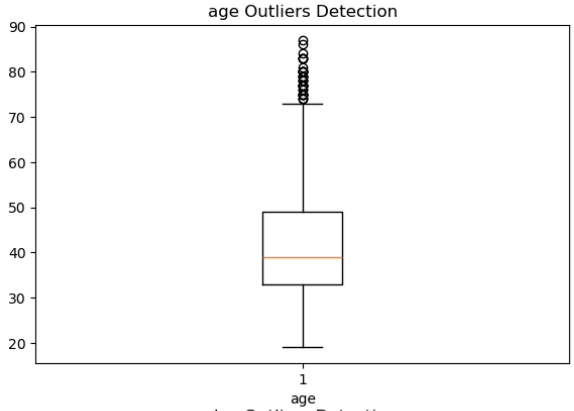
i) Outliers Detection

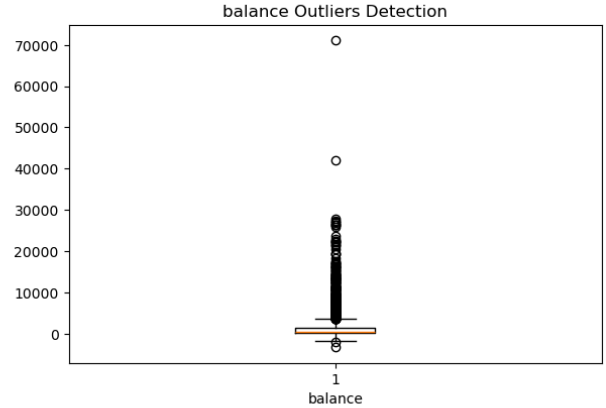
In [393…
```
num_col
```

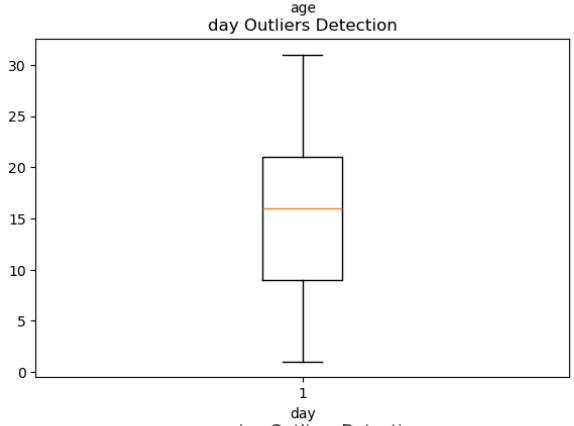
Out[393…
```
['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']
```

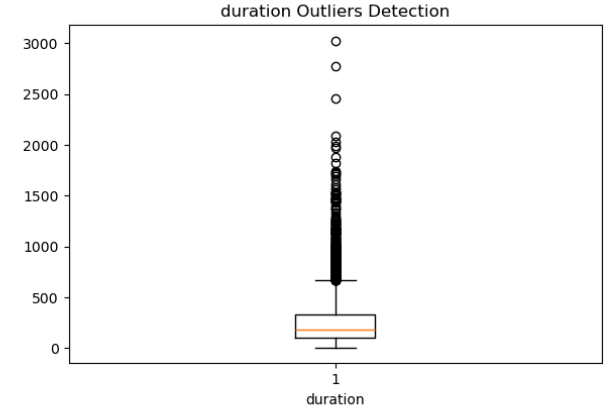
In [394…
```
plt.figure(figsize=(15,20))
plt.suptitle("Outlier Detection")
for i in range(0,len(num_col)):
    plt.subplot(4,2,i+1)
    plt.boxplot(bank_df[num_col[i]],vert=True)
    plt.title(f"{num_col[i]} Outliers Detection")
    plt.xlabel(f'{num_col[i]}')

plt.show()
```

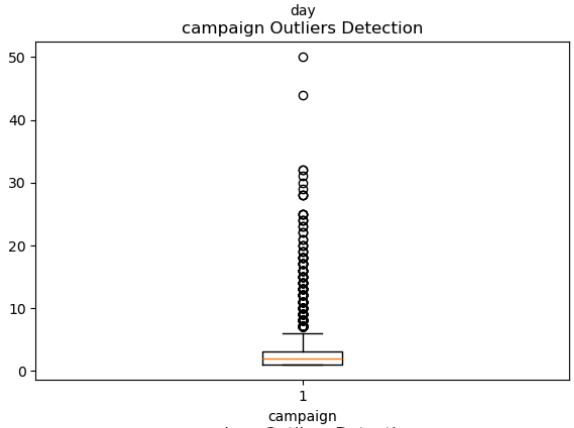
Outlier Detection

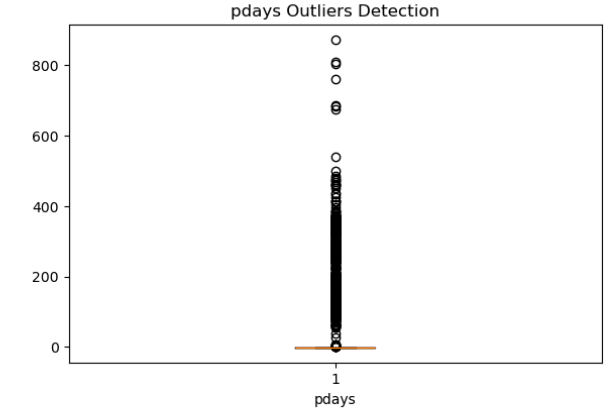


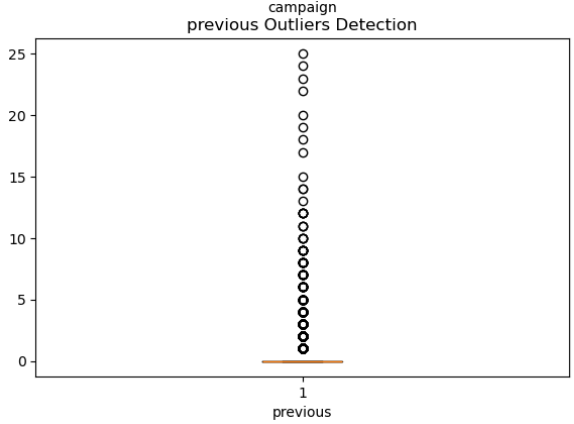












In [395…

```
pdays_unique=bank_df['pdays'].value_counts()
pdays_key=pdays_unique.keys()
pdays_value=pdays_unique.values
pdays_dict={}

for i in zip(pdays_key,pdays_value):
    pdays_dict[i[0]]=i[1]

print(f"Total unique values in pdays column is {len(pdays_unique)}")
print(f"variance is {round(bank_df['pdays'].var(),2)}")
print(f"Standard deviation is {round(bank_df['pdays'].std(),2)}")
pdays_dict
```

```
Total unique values in pdays column is 292
variance is 10024.24
Standard deviation is 100.12
```

Out[395…


```
{-1: 3705,
 182: 23,
 183: 20,
 363: 12,
 92: 12,
 91: 11,
 181: 10,
 169: 10,
 370: 9,
 364: 9,
 349: 8,
 99: 8,
 167: 8,
 94: 8,
 172: 7,
 184: 7,
 2: 7,
 176: 7,
 85: 6,
 175: 6,
 185: 6,
 351: 6,
 342: 6,
 345: 6,
 152: 6,
 87: 6,
 266: 6,
 95: 6,
 173: 6,
 368: 6,
 96: 6,
 150: 6,
 105: 5,
 272: 5,
 120: 5,
 360: 5,
 356: 5,
 170: 5,
 271: 5,
 357: 5,
 174: 5,
 367: 5,
 90: 5,
 350: 5,
 371: 5,
 330: 5,
 190: 5,
 98: 4,
 241: 4,
 358: 4,
 337: 4,
 327: 4,
 189: 4,
 273: 4,
 322: 4,
 147: 4,
```

```
84: 4,
344: 4,
246: 4,
112: 4,
247: 4,
340: 4,
259: 4,
107: 4,
78: 4,
195: 4,
93: 4,
335: 4,
188: 4,
127: 4,
347: 4,
97: 4,
196: 4,
366: 3,
146: 3,
339: 3,
186: 3,
253: 3,
204: 3,
114: 3,
140: 3,
124: 3,
298: 3,
168: 3,
261: 3,
161: 3,
154: 3,
180: 3,
148: 3,
256: 3,
309: 3,
365: 3,
343: 3,
122: 3,
336: 3,
353: 3,
104: 3,
346: 3,
192: 3,
111: 3,
275: 3,
297: 3,
89: 3,
334: 3,
295: 3,
138: 3,
64: 3,
321: 3,
7: 3,
102: 3,
249: 3,
178: 3,
```

```
317: 3,
355: 3,
332: 3,
315: 3,
352: 3,
88: 3,
211: 2,
362: 2,
359: 2,
274: 2,
280: 2,
300: 2,
227: 2,
299: 2,
316: 2,
286: 2,
361: 2,
224: 2,
166: 2,
106: 2,
201: 2,
153: 2,
474: 2,
369: 2,
326: 2,
109: 2,
197: 2,
338: 2,
244: 2,
415: 2,
291: 2,
179: 2,
258: 2,
221: 2,
177: 2,
264: 2,
193: 2,
287: 2,
145: 2,
137: 2,
116: 2,
325: 2,
323: 2,
308: 2,
281: 2,
278: 2,
135: 2,
373: 2,
133: 2,
136: 2,
164: 2,
260: 2,
303: 2,
207: 2,
126: 2,
80: 2,
```

```
288: 2,
191: 2,
293: 2,
333: 2,
198: 2,
209: 2,
292: 2,
113: 2,
1: 2,
100: 2,
130: 2,
187: 2,
270: 2,
149: 2,
461: 2,
141: 2,
110: 2,
101: 2,
62: 1,
250: 1,
231: 1,
397: 1,
541: 1,
165: 1,
348: 1,
82: 1,
79: 1,
119: 1,
242: 1,
414: 1,
265: 1,
484: 1,
319: 1,
61: 1,
28: 1,
139: 1,
761: 1,
123: 1,
144: 1,
804: 1,
331: 1,
219: 1,
57: 1,
77: 1,
76: 1,
74: 1,
5: 1,
56: 1,
687: 1,
159: 1,
311: 1,
436: 1,
500: 1,
450: 1,
160: 1,
460: 1,
```

```
86: 1,
283: 1,
312: 1,
223: 1,
75: 1,
467: 1,
267: 1,
385: 1,
388: 1,
83: 1,
158: 1,
115: 1,
206: 1,
199: 1,
248: 1,
210: 1,
208: 1,
375: 1,
143: 1,
232: 1,
73: 1,
313: 1,
162: 1,
117: 1,
131: 1,
462: 1,
268: 1,
205: 1,
60: 1,
69: 1,
3: 1,
225: 1,
262: 1,
479: 1,
674: 1,
808: 1,
103: 1,
378: 1,
58: 1,
382: 1,
305: 1,
171: 1,
151: 1,
59: 1,
329: 1,
683: 1,
38: 1,
235: 1,
255: 1,
254: 1,
239: 1,
426: 1,
435: 1,
222: 1,
294: 1,
238: 1,
```

```
 871: 1,
 341: 1,
 284: 1,
 212: 1,
 282: 1,
 374: 1,
 404: 1,
 118: 1,
 386: 1,
 63: 1,
 81: 1,
 234: 1}
```

In [396…

```
previous_unique=bank_df['previous'].value_counts()
previous_key=previous_unique.keys()
previous_value=previous_unique.values
previous_dict={}

for i in zip(previous_key,previous_value):
    previous_dict[i[0]]=i[1]

print(f"variance is {round(bank_df['previous'].var(),2)}")
print(f"Standard deviation is {round(bank_df['previous'].std(),2)}")
print(f"Total unique values in pdays column is {len(previous_unique)}")
previous_dict
```

```
variance is 2.87
Standard deviation is 1.69
Total unique values in pdays column is 24
```

Out[396…

```
{0: 3705,
 1: 286,
 2: 193,
 3: 113,
 4: 78,
 5: 47,
 6: 25,
 7: 22,
 8: 18,
 9: 10,
 12: 5,
 10: 4,
 11: 3,
 14: 2,
 24: 1,
 22: 1,
 23: 1,
 17: 1,
 18: 1,
 15: 1,
 13: 1,
 19: 1,
 20: 1,
 25: 1}
```

Conclusion

- From the above boxplots of numerical columns almost each and every column contain the outliers
- Almost all outliers contain the outliers on the positive side except balance column contain outliers on both positive and negative side.
- pdays And previous column boxplot is different from other columns boxplot this is because :
    1. In pdays column more than 95% values are belong to the -1 and In previous column more than 95% values are belongs to the 0.
    2. so, the pdays and previous column data distribution is highly right skewed
    3. maximum values are the are same so it there will be IQR 0, so it will consider every values inside Q1 and Q3 as the outlier(It will show all values present in these column as the outliers).
    4. If we fill all values with median then we lose the actual data of the columns and all data in the column will same so variance and standard deviation will zero basically the column will useless.
    5. Due to all of these IQR and ZScore methods fells on pdays and previous column.

ii) Outliers Handling

In [397…
```
num_col
```

Out[397…
```
['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']
```

In [398…
```
# outliers count and percentage

for i in num_col:
    Q1=bank_df[i].quantile(0.25)
    Q2=bank_df[i].quantile(0.50)
    Q3=bank_df[i].quantile(0.75)
    IQR=Q3-Q1

    lb=Q1-(1.5*IQR)
    ub=Q3+(1.5*IQR)

    con=(bank_df[i]<=lb) | (bank_df[i]>=ub)
    without_out_con=(bank_df[i]>lb) & (bank_df[i]<ub)

    per=round((len(bank_df[con])/len(bank_df))*100,2)
    outlier_free_per=round((len(bank_df[without_out_con])/len(bank_df))*100,2)

    keys=["Total-Count","25P","50P","75P","IQR","Lower-Bound","Upper-Bound","Outlie
    value=[len(bank_df[i]),Q1,Q2,Q3,IQR,lb,ub,len(bank_df[con]),per,len(bank_df[wit

    df=pd.DataFrame(zip(keys,value),columns=[f"{i}","Outlier-Data"])
    print(df)
    print("\n\n")
```

	age	Outlier-Data
0	Total-Count	4521.00
1	25P	33.00
2	50P	39.00
3	75P	49.00
4	IQR	16.00
5	Lower-Bound	9.00
6	Upper-Bound	73.00
7	Outlier-Count	44.00
8	Outlier-Percentage	0.97
9	outlier-free-count	4477.00
10	outlier-free-percentage	99.03

	balance	Outlier-Data
0	Total-Count	4521.00
1	25P	69.00
2	50P	444.00
3	75P	1480.00
4	IQR	1411.00
5	Lower-Bound	-2047.50
6	Upper-Bound	3596.50
7	Outlier-Count	506.00
8	Outlier-Percentage	11.19
9	outlier-free-count	4015.00
10	outlier-free-percentage	88.81

	day	Outlier-Data
0	Total-Count	4521.0
1	25P	9.0
2	50P	16.0
3	75P	21.0
4	IQR	12.0
5	Lower-Bound	-9.0
6	Upper-Bound	39.0
7	Outlier-Count	0.0
8	Outlier-Percentage	0.0
9	outlier-free-count	4521.0
10	outlier-free-percentage	100.0

	duration	Outlier-Data
0	Total-Count	4521.0
1	25P	104.0
2	50P	185.0
3	75P	329.0
4	IQR	225.0
5	Lower-Bound	-233.5
6	Upper-Bound	666.5
7	Outlier-Count	330.0
8	Outlier-Percentage	7.3
9	outlier-free-count	4191.0

```
10  outlier-free-percentage          92.7
```

```
                   campaign  Outlier-Data
0               Total-Count       4521.00
1                       25P          1.00
2                       50P          2.00
3                       75P          3.00
4                       IQR          2.00
5               Lower-Bound         -2.00
6               Upper-Bound          6.00
7             Outlier-Count        473.00
8        Outlier-Percentage         10.46
9        outlier-free-count       4048.00
10  outlier-free-percentage         89.54
```

```
                     pdays  Outlier-Data
0               Total-Count        4521.0
1                       25P          -1.0
2                       50P          -1.0
3                       75P          -1.0
4                       IQR           0.0
5               Lower-Bound          -1.0
6               Upper-Bound          -1.0
7             Outlier-Count        4521.0
8        Outlier-Percentage         100.0
9        outlier-free-count           0.0
10  outlier-free-percentage           0.0
```

```
                    previous  Outlier-Data
0               Total-Count        4521.0
1                       25P           0.0
2                       50P           0.0
3                       75P           0.0
4                       IQR           0.0
5               Lower-Bound           0.0
6               Upper-Bound           0.0
7             Outlier-Count        4521.0
8        Outlier-Percentage         100.0
9        outlier-free-count           0.0
10  outlier-free-percentage           0.0
```

Conclusion from the above code

- IQR outlier handling method is working on almost all column except pdays and previous column
- IQR fails on pdays and previous column because:

1. The pdays and previous column has a very skewed distribution
2. Maximum data belongs to the same value so the IQR becomes 0 and it fails

In [399…
```
# outliers filling with median using IQR

for i in num_col:
    cList=[]
    if i not in ['pdays', 'previous']:

        Q1=bank_df[i].quantile(0.25)
        Q2=bank_df[i].quantile(0.50)
        Q3=bank_df[i].quantile(0.75)
        IQR=Q3-Q1

        lb=Q1-(1.5*IQR)
        ub=Q3+(1.5*IQR)

        con=(bank_df[i]<=lb) | (bank_df[i]>=ub)

        for j in bank_df[i]:
            if j in bank_df[con][i].values:
                cList.append(Q2)
            else:
                cList.append(j)

        bank_df[i]=cList
```

Conclusion of the above code

- We handle the all column outliers using IQR method except pdays and previous column

In [400…
```
# handling pdays column data by converting numerical values into categorical column

def categorize_pdays(x):
    if x == -1:
        return 'not contacted'
    elif x <= 30:
        return 'recent'
    elif x <= 180:
        return 'intermediate'
    else:
        return 'old'

bank_df['pdays'] = bank_df['pdays'].apply(categorize_pdays)
```

In [401…
```
print(f"Data type of pdays column after conversion is : {bank_df['pdays'].dtype}")
bank_df['pdays'].value_counts()
```

```
Data type of pdays column after conversion is : object
```

Out[401…
```
pdays
not contacted    3705
old               486
intermediate      315
recent             15
Name: count, dtype: int64
```

In [402…
```
# handling previous column data by converting numerical values into categorical col

def categorize_previous(x):
    if x == 0:
        return 'not contacted'
    elif x <= 2:
        return 'Few contacts'
    else:
        return 'Many contacts'

bank_df['previous'] = bank_df['previous'].apply(categorize_previous)
```

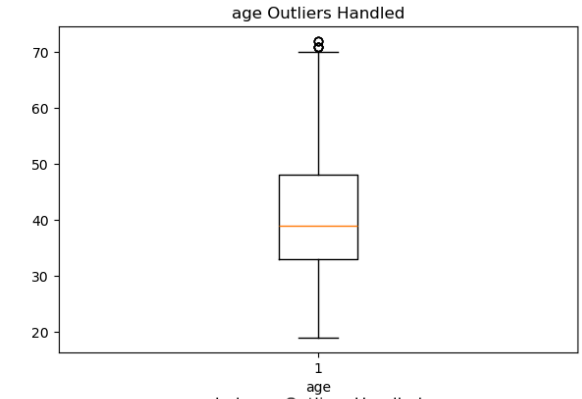
In [403…
```
print(f"Data type of previous column after conversion is : {bank_df['previous'].dty
bank_df['previous'].value_counts()
```

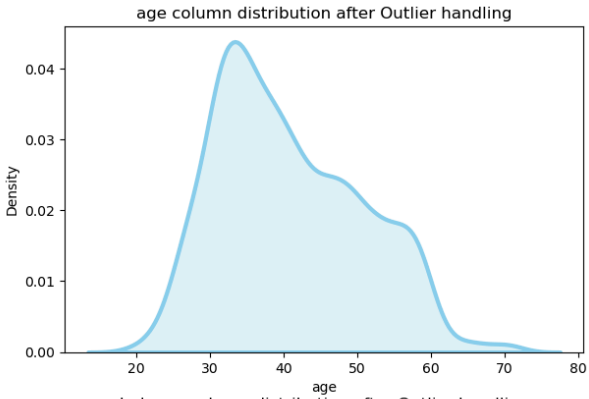
```
Data type of previous column after conversion is : object
```

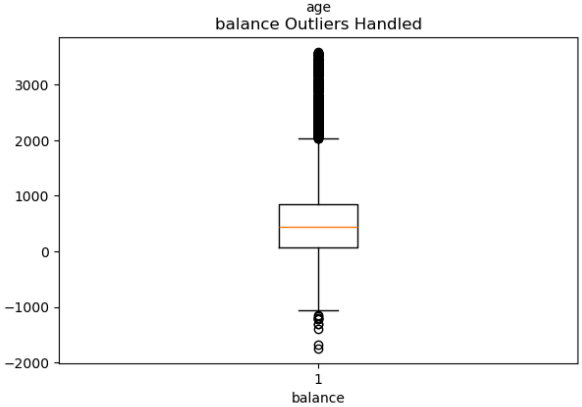
Out[403…
```
previous
not contacted    3705
Few contacts      479
Many contacts     337
Name: count, dtype: int64
```

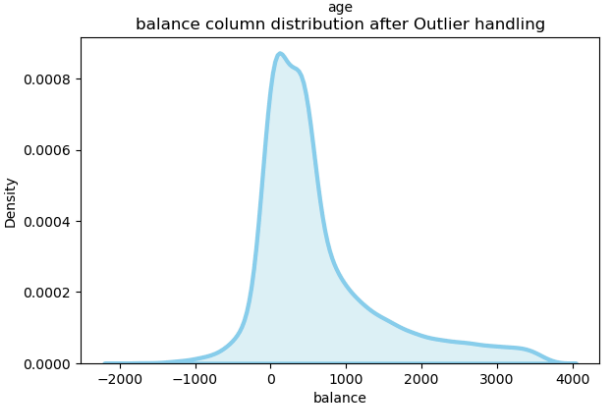
In [404…
```
# showing the boxplot and distribution plot of the column whose outliers are handle
# to check outliers are handled properly without affecting the distribution of the

count=1

plt.figure(figsize=(15,25))
plt.suptitle("Columns After Outlier Handling By Median")

for i in range(0,len(['age', 'balance', 'day', 'duration', 'campaign'])):
    plt.subplot(5,2,count)
    plt.boxplot(bank_df[num_col[i]],vert=True)
    plt.title(f"{num_col[i]} Outliers Handled")
    plt.xlabel(f'{num_col[i]}')

    plt.subplot(5,2,count+1)
    sns.kdeplot(bank_df[num_col[i]],fill=True,color="skyblue",linewidth=3)
    plt.title(f"{num_col[i]} column distribution after Outlier handling")
    plt.xlabel(f'{num_col[i]}')

    count+=2

plt.show()
```

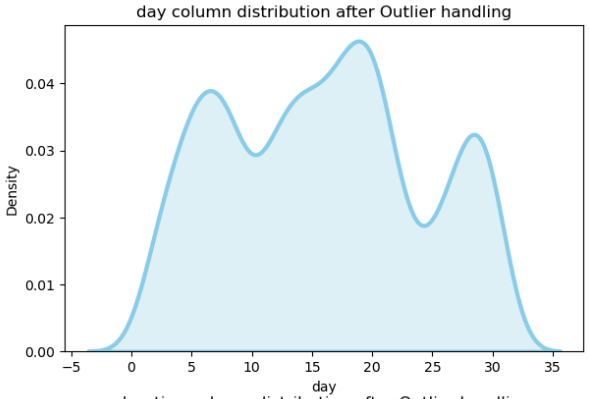
Columns After Outlier Handling By Median

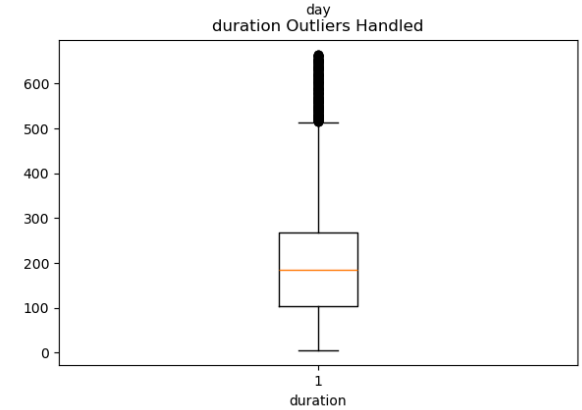


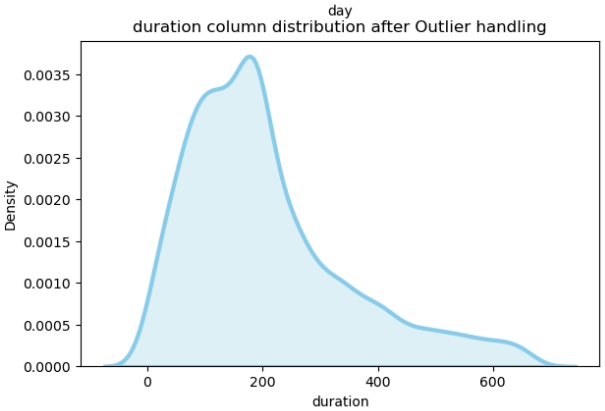


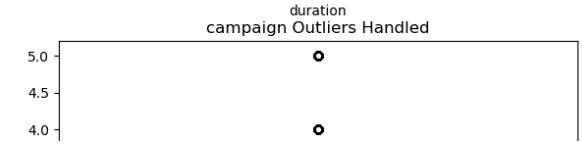


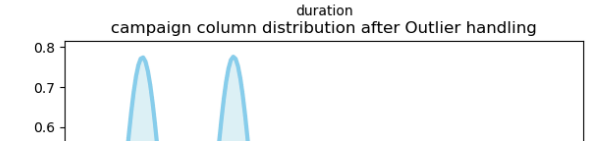


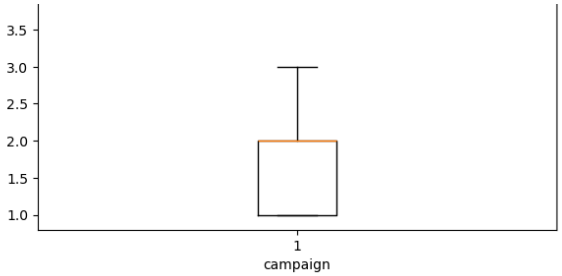


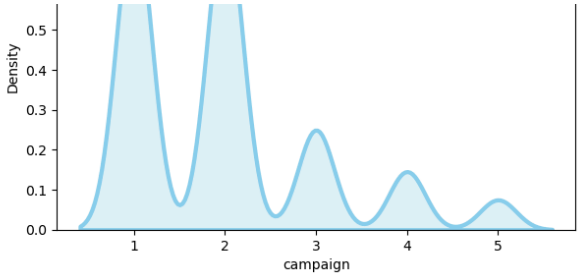












Conclusion

- Amost every column distribution shape is changed
- Extreme outliers are removed from almost all the column
- Some column have still fewer outliers but they are not affecting much to analysis
- Balance column do not contain any outlier so the distribution shape of the balance column remains same
- After handling outliers columns are getting more closer to bell curve than previous

pdays and previous column analysis after categorical conversion

In [405…

```
# pdays column analysis

label=bank_df['pdays'].unique()
count=[]
for i in label:
    con=bank_df['pdays']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[405…

	Class-Name	Class-Frequency
0	not contacted	3705
1	old	486
2	intermediate	315
3	recent	15

In [406…

```
label=bank_df['pdays'].unique()
count=[]
for i in label:
    con=bank_df['pdays']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)
```

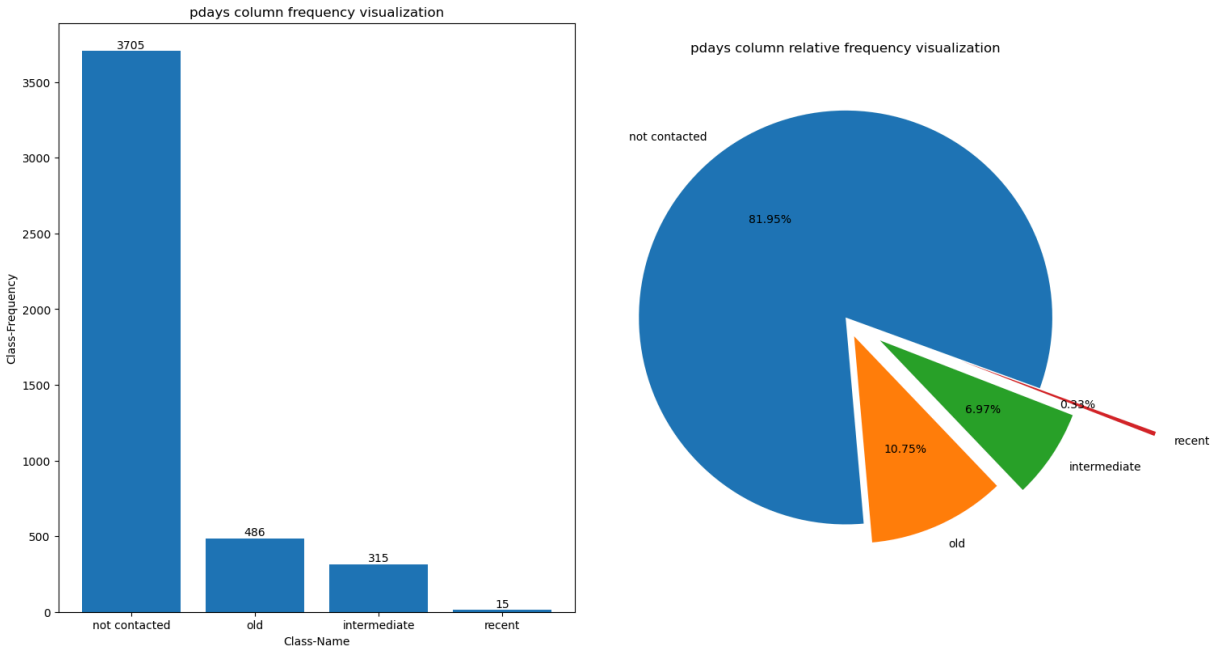
```
rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

Out[406…

	Class-Name	Relative-Frequency
0	not contacted	81.95
1	old	10.75
2	intermediate	6.97
3	recent	0.33

In [407…

```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")

bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("pdays column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("pdays column relative frequency visualization")

plt.show()
```



Conclusion

From Histogram Graph

- 'not contacted' category is the dominant category in the column having 3705 counts
- 'not contacted' category indicates that a large majority of customer were not contacted in previous campaign

- Other Categories Have Much Lower Counts "old": 486, "intermediate": 315, "recent": 15. These counts are significantly lower than "not contacted", showing high imbalance in this feature.

From Pie Chart

- 81.95% of the total data is belong to 'not contacted' category and dominating the distribution
- "old" category is 10.75% second big portion of the data
- "intermediate" category is 6.97% small portion of the data.
- "recent" category is 0.33%, making it extremely rare in the dataset.

In [408…
```
# previous column analysis

label=bank_df['previous'].unique()
count=[]
for i in label:
    con=bank_df['previous']==i
    total=len(bank_df[con])
    count.append(total)

freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Class-Frequency"]).
freq_table
```

Out[408…

	Class-Name	Class-Frequency
0	not contacted	3705
2	Few contacts	479
1	Many contacts	337

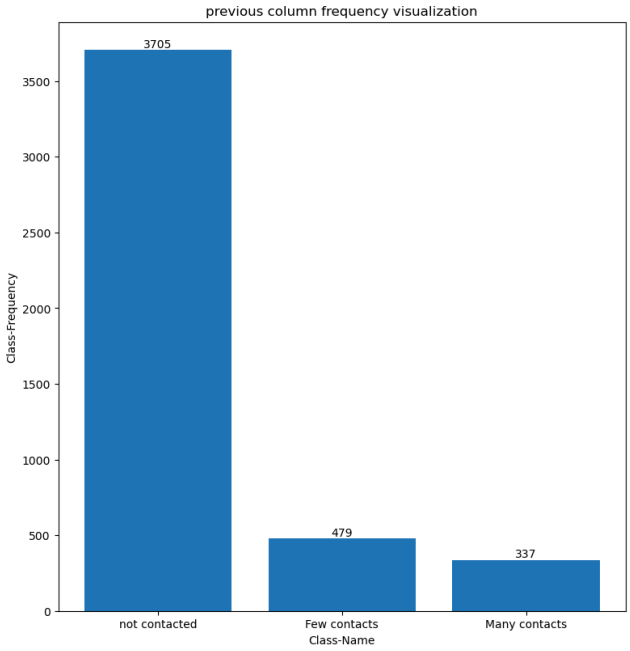
In [409…
```
label=bank_df['previous'].unique()
count=[]
for i in label:
    con=bank_df['previous']==i
    per=round((len(bank_df[con])/len(bank_df)*100),2)
    count.append(per)

rel_freq_table=pd.DataFrame(zip(label,count),columns=["Class-Name","Relative-Freque
rel_freq_table
```

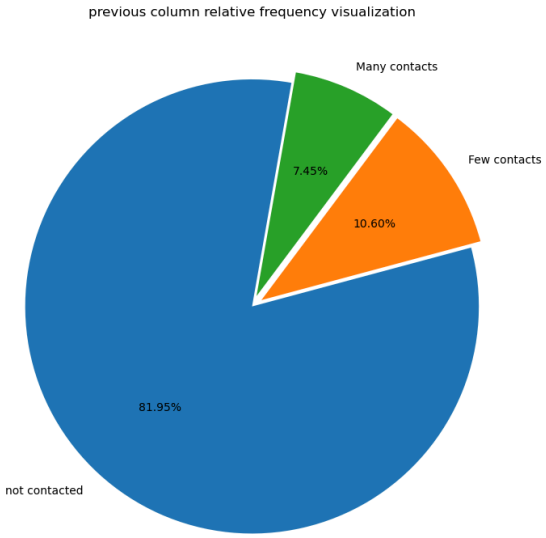
Out[409…

	Class-Name	Relative-Frequency
0	not contacted	81.95
2	Few contacts	10.60
1	Many contacts	7.45

In [410…
```
fig,ax=plt.subplots(1,2,figsize=(15,8),layout="constrained")
```

```
bar=ax[0].bar(freq_table['Class-Name'],freq_table['Class-Frequency'])
ax[0].bar_label(bar)
ax[0].set_xlabel("Class-Name")
ax[0].set_ylabel("Class-Frequency")
ax[0].set_title("previous column frequency visualization")

ax[1].pie(rel_freq_table['Relative-Frequency'],labels=rel_freq_table['Class-Name'],
ax[1].set_title("previous column relative frequency visualization")

plt.show()
```





Conclusion

From Histogram Graph

- "not contacted" has the highest count: 3705 entries.
- "Few contacts" has 479 entries.
- "Many contacts" has 337 entries.
- This shows a clear dominance of the "not contacted" class in the dataset.

From Pie Chart

- "not contacted" represents 81.95% of the total data.
- "Few contacts" accounts for 10.60%.
- "Many contacts" makes up 7.45%.
- This confirms a high class imbalance in the data.

Bivariate & Multivariate Analysis

In [411…

```
# Again seperating numrical and categorical column due to column type changed
```

```
cat_col=bank_df.select_dtypes(include='object').columns
num_col=bank_df.select_dtypes(exclude='object').columns

print(cat_col)
print(num_col)
```

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

i) Bivariate Analysis

In [412…

```
bank_df.head(3)
```

Out[412…

	age	job	marital	education	default	balance	housing	loan	contact	day
0	30.0	unemployed	married	primary	no	1787.0	no	no	cellular	19
1	33.0	services	married	secondary	no	444.0	yes	yes	cellular	11
2	35.0	management	single	tertiary	no	1350.0	yes	no	cellular	16

Checking the affect of categorical columns on the target variable

In [413…

```
cat_col
```

Out[413…

```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'pdays', 'previous', 'poutcome', 'term_deposite'],
      dtype='object')
```

In [414…

```
bank_df['term_deposite'].unique()
```

Out[414…

```
array(['no', 'yes'], dtype=object)
```

In [118…

```
# job vs term_deposite column analysis

job_col=bank_df['job']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=job_col.unique()

for i in labels:
    con1=job_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'
```

```
    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['job','td_ye
df
```

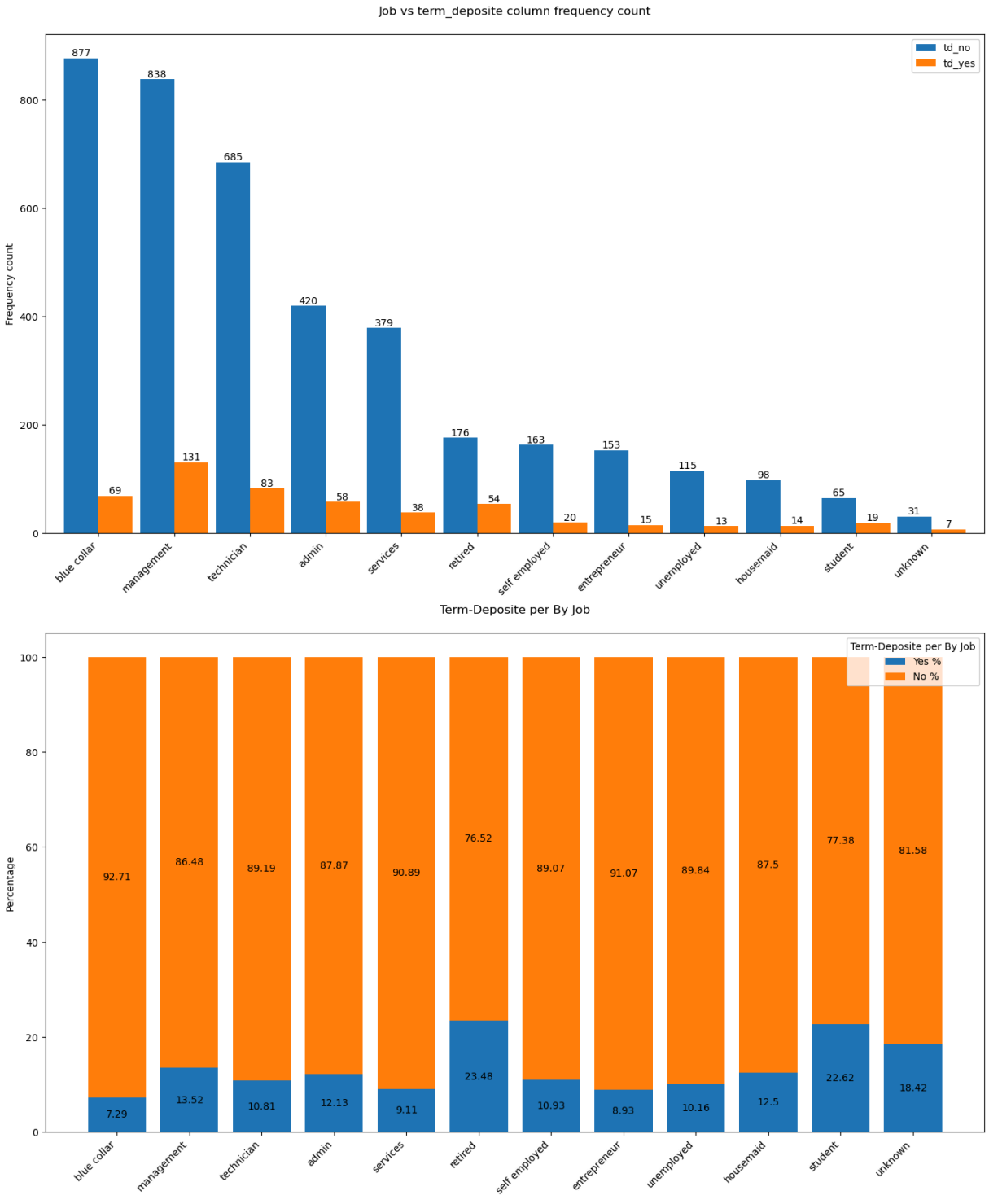
Out[118…

	job	td_yes	td_no	td_yes_per	td_no_per
3	blue collar	69	877	7.29	92.71
2	management	131	838	13.52	86.48
5	technician	83	685	10.81	89.19
7	admin	58	420	12.13	87.87
1	services	38	379	9.11	90.89
10	retired	54	176	23.48	76.52
4	self employed	20	163	10.93	89.07
6	entrepreneur	15	153	8.93	91.07
0	unemployed	13	115	10.16	89.84
9	housemaid	14	98	12.50	87.50
8	student	19	65	22.62	77.38
11	unknown	7	31	18.42	81.58

In [118…

```
fig,chart=plt.subplot_mosaic([['top','top'],['bottom','bottom']],figsize=(17,20))

bar_chart=df[['job','td_no','td_yes']].set_index('job').plot(kind='bar',ax=chart['t
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['top'].set_title("Job vs term_deposite column frequency count",pad=20)
chart['top'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['top'].set_ylabel('Frequency count')
chart['top'].set_xlabel('')

b1=chart['bottom'].bar(df['job'], df['td_yes_per'])
plt.bar_label(b1,label_type='center', label='Yes %')
b2=chart['bottom'].bar(df['job'], df['td_no_per'], bottom=df['td_yes_per'], label='
chart['bottom'].bar_label(b2,label_type='center')
```

```
chart['bottom'].set_title('Term-Deposite per By Job',pad=20)
chart['bottom'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Job"
chart['bottom'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right'
chart['bottom'].set_ylabel("Percentage")

plt.show()
```



Conclusion

From First Plot

- blue-collar, management, and technician have the highest total number of clients.
- retired and student jobs, despite having fewer total people, have a noticeably higher proportion of people who said "yes".

From Second Plot

- Highest conversion rate of term deposite yes is in Retired, Student, Unknown, Management and Admin. target them more in marketing.
- This tells Retired, student and higher post in finance peoples are more interested in the term_deposite
- Lowest conversion rate of term deposite yes is in Blue-Collar, Entrepreneur, Services. You can impove offers and messages
- Blue-Collar jobs client are more but it's conversion rate is very low and non conversion rate is highest

In [118…

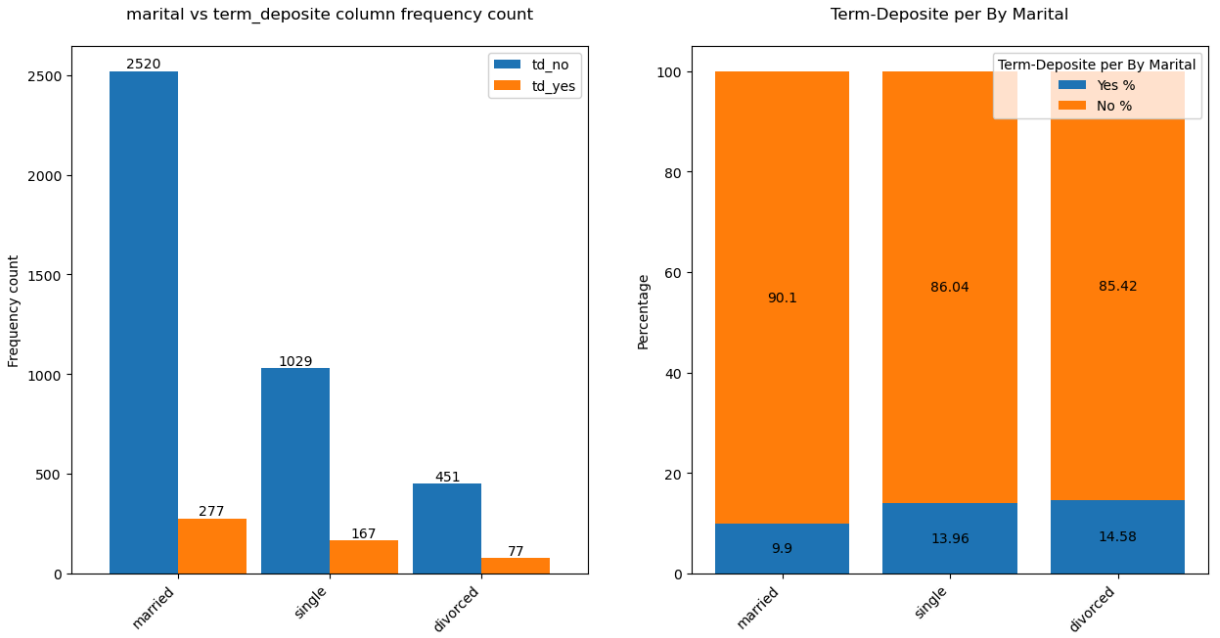
```
# marital vs term_deposite column analysis

marital_col=bank_df['marital']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=marital_col.unique()

for i in labels:
    con1=marital_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['marital','t
df
```

Out[118...

	marital	td_yes	td_no	td_yes_per	td_no_per
0	married	277	2520	9.90	90.10
1	single	167	1029	13.96	86.04
2	divorced	77	451	14.58	85.42

In [118...

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['marital','td_no','td_yes']].set_index('marital').plot(kind='bar',ax=
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("marital vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['marital'], df['td_yes_per'])
plt.bar_label(b1,label_type='center', label='Yes %')
b2=chart['right'].bar(df['marital'], df['td_no_per'], bottom=df['td_yes_per'], labe
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Marital',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Marit
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```



Conclusion

From Left Plot

- Married customers contribute highest count of the data. Out of 2797, 2520 says 'no' and 277 says 'yes' for term-deposite.
- Single clients shows moderate representation. Out of 1096, 1029 says 'no' and 67 says 'yes' for term-deposite
- Divorced clients contribute lowest count

From Right Plot

- married customers have 9.9% subscribed to term deposits. It is lowest conversion rate.
- single customers have 13.96% subscribed to term deposits. It has higher than married.
- divorced customers have 14.58% subscribed to term deposits. It has highest among all three categories.
- Divorced and Single clients are more likely to subscribe to term deposits than Married clients.
- Even though Married people are the majority, their conversion rate is the lowest.
- Consider targeting Single and Divorced clients in marketing campaigns for better conversion potential.

In [596…

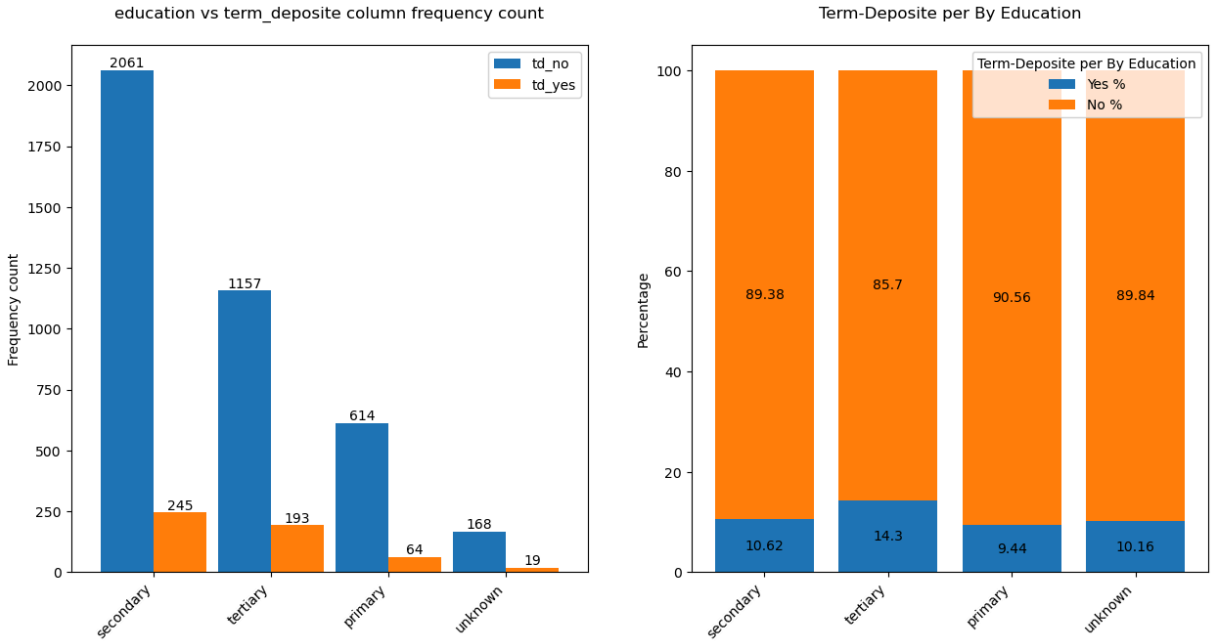
```
# education vs term_deposite column analysis

education_col=bank_df['education']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=education_col.unique()

for i in labels:
    con1=education_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['education',
df
```

Out[596...

	education	td_yes	td_no	td_yes_per	td_no_per
1	secondary	245	2061	10.62	89.38
2	tertiary	193	1157	14.30	85.70
0	primary	64	614	9.44	90.56
3	unknown	19	168	10.16	89.84

In [597...

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['education','td_no','td_yes']].set_index('education').plot(kind='bar'
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("education vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['education'], df['td_yes_per'])
plt.bar_label(b1,label_type='center', label='Yes %')
b2=chart['right'].bar(df['education'], df['td_no_per'], bottom=df['td_yes_per'], la
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Education',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Educa
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```



Conclusion

From Left Plot

- Secondary education group has the highest total count
- Unknown education group has the lowest total count
- Tertiary and Primary has moderate tatal count

From Right Plot

- Tertiary education group has the highest subscription rate 14.3%.
- Secondary and Unknown has almost equal subscription rate 10.62 and 10.16
- Secondary education client contains highest contribution of the data but subscription rate is very low here.
- unknown education client have very less but the subscription rate is very good as compare to others.
- primary education clients subscription rate is littel bit less than secondary and unknown education.

In [598…

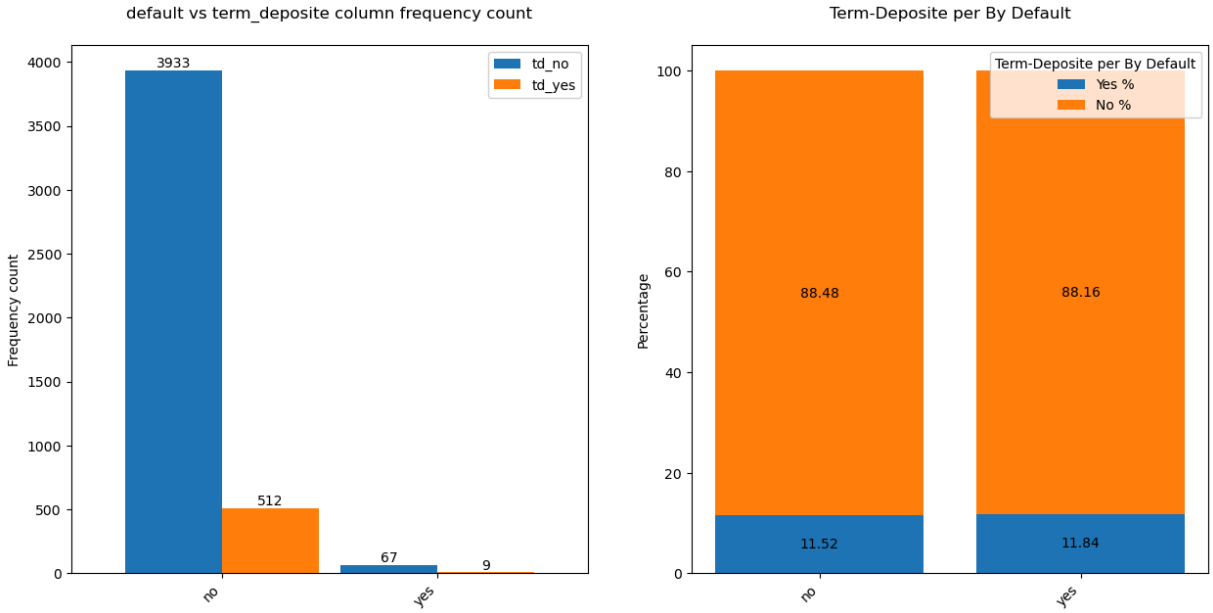
```
# default vs term_deposite column analysis

default_col=bank_df['default']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=default_col.unique()

for i in labels:
    con1=default_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['default','t
df
```

Out[598…

	default	td_yes	td_no	td_yes_per	td_no_per
0	no	512	3933	11.52	88.48
1	yes	9	67	11.84	88.16

In [600…

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['default','td_no','td_yes']].set_index('default').plot(kind='bar',ax=
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("default vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['default'], df['td_yes_per'])
plt.bar_label(b1,label_type='center', label='Yes %')
b2=chart['right'].bar(df['default'], df['td_no_per'], bottom=df['td_yes_per'], labe
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Default',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Defau
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```



Conclusion

- Customers who do not have a credit default history are far more in number and show a moderately higher likelihood of subscribing to a term deposit.
- Only a very small number of customers with a default history are present in the data, indicating they are either rare in the dataset.
- The subscription rate to term deposits is almost the same between defaulters and non-defaulters, with a negligible difference of only about 0.32%, showing no strong influence of default history on subscription decision.
- The majority of subscriptions come from non-defaulters

In [601…

```
# housing vs term_deposite column analysis

housing_col=bank_df['housing']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=housing_col.unique()

for i in labels:
    con1=housing_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['housing','t
df
```

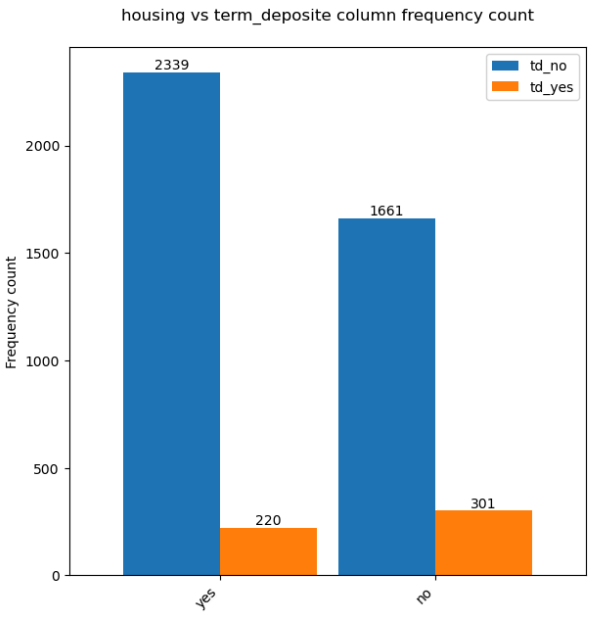
Out[601…

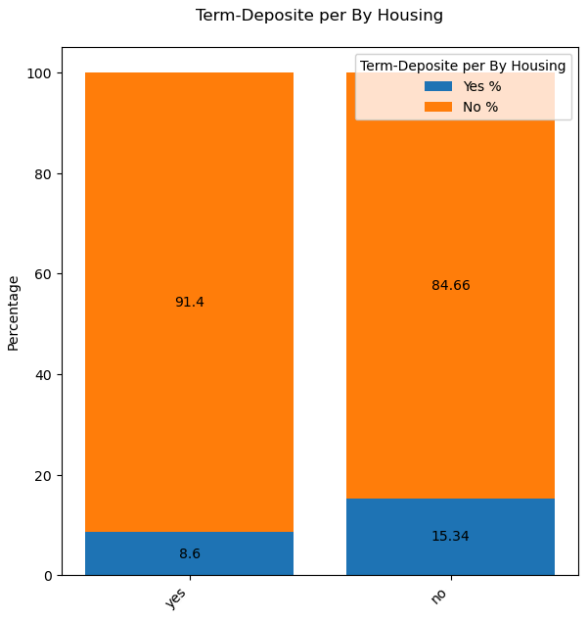
	housing	td_yes	td_no	td_yes_per	td_no_per
1	yes	220	2339	8.60	91.40
0	no	301	1661	15.34	84.66

In [602…

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['housing','td_no','td_yes']].set_index('housing').plot(kind='bar',ax=
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("housing vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['housing'], df['td_yes_per'])
plt.bar_label(b1,label_type='center', label='Yes %')
b2=chart['right'].bar(df['housing'], df['td_no_per'], bottom=df['td_yes_per'], labe
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Housing',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Housi
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")
```

```
plt.show()
```





Conclusion

- Clients without housing loans are almost twice as likely to subscribe to a term deposit compared to those with housing loans.
- Even though more clients have housing loans, their term deposit subscription rate is significantly lower.
- Housing loan is reducing the subscription rate to term deposite
- Marketing strategies could target clients without housing loans, as they show higher interest in term deposits.

In [609…

```
# loan vs term_deposite column analysis

loan_col=bank_df['loan']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=loan_col.unique()

for i in labels:
    con1=loan_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)
```

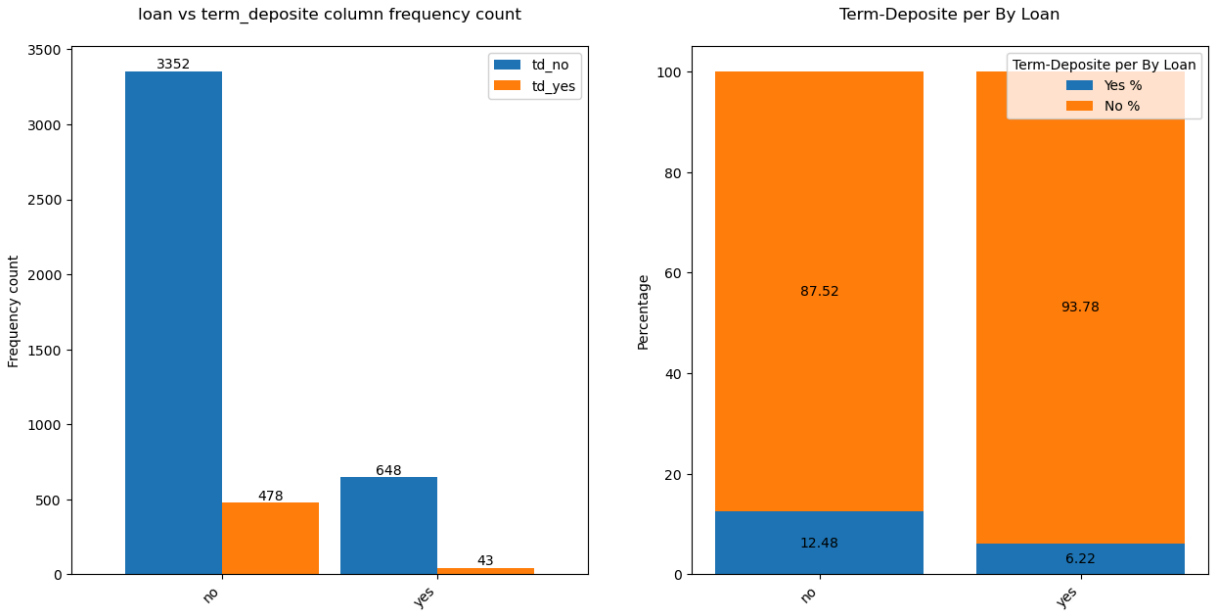
```
    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['loan','td_y
df
```

Out[609…

	loan	td_yes	td_no	td_yes_per	td_no_per
0	no	478	3352	12.48	87.52
1	yes	43	648	6.22	93.78

In [610…

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['loan','td_no','td_yes']].set_index('loan').plot(kind='bar',ax=chart[
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("loan vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['loan'], df['td_yes_per'])
plt.bar_label(b1,label_type='center')
b2=chart['right'].bar(df['loan'], df['td_no_per'], bottom=df['td_yes_per'])
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Loan',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Loan"
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```



Conclusion

- Clients without personal loans are twice as likely to subscribe to a term deposit compared to those with loans.
- Clients with active personal loans show low interest in term deposits, possibly due to financial burden.
- Financial stability may play a role in the decision to subscribe to a term deposit.
- Marketing strategies could target customers without existing loans for promoting term deposits.

In [612…

```
# contact vs term_deposite column analysis

contact_col=bank_df['contact']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=contact_col.unique()

for i in labels:
    con1=contact_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['contact','t
df
```

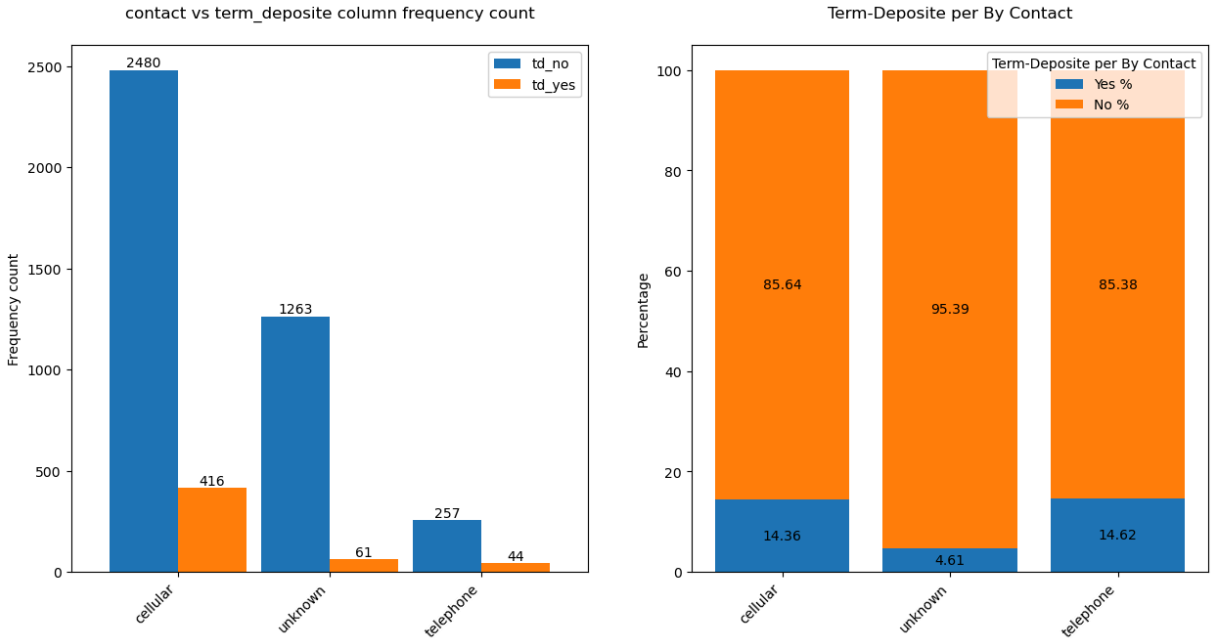
Out[612…

	contact	td_yes	td_no	td_yes_per	td_no_per
0	cellular	416	2480	14.36	85.64
1	unknown	61	1263	4.61	95.39
2	telephone	44	257	14.62	85.38

In [614…

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['contact','td_no','td_yes']].set_index('contact').plot(kind='bar',ax=
for i in bar_chart.containers:
```

```
    bar_chart.bar_label(i)

chart['left'].set_title("contact vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['contact'], df['td_yes_per'])
plt.bar_label(b1,label_type='center')
b2=chart['right'].bar(df['contact'], df['td_no_per'], bottom=df['td_yes_per'])
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Contact',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Conta
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```



Conclusion

- Telephone and Cellular contacts are significantly more effective in gaining term deposit subscriptions (14%+), compared to unknown contacts, which result in only 4.61% subscription.
- Customers with "unknown" contact methods have a much lower subscribing to a term deposit, likely due to lack of engagement or incomplete data.
- Targeting through telephone or cellular seems to be much more successful due to the personalized or immediate nature of the communication.

In [118…

```
# month vs term_deposite column analysis

month_col=bank_df['month']
term_deposite=bank_df['term_deposite']
```

```
yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=month_col.unique()

for i in labels:
    con1=month_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['month','td_
df
```

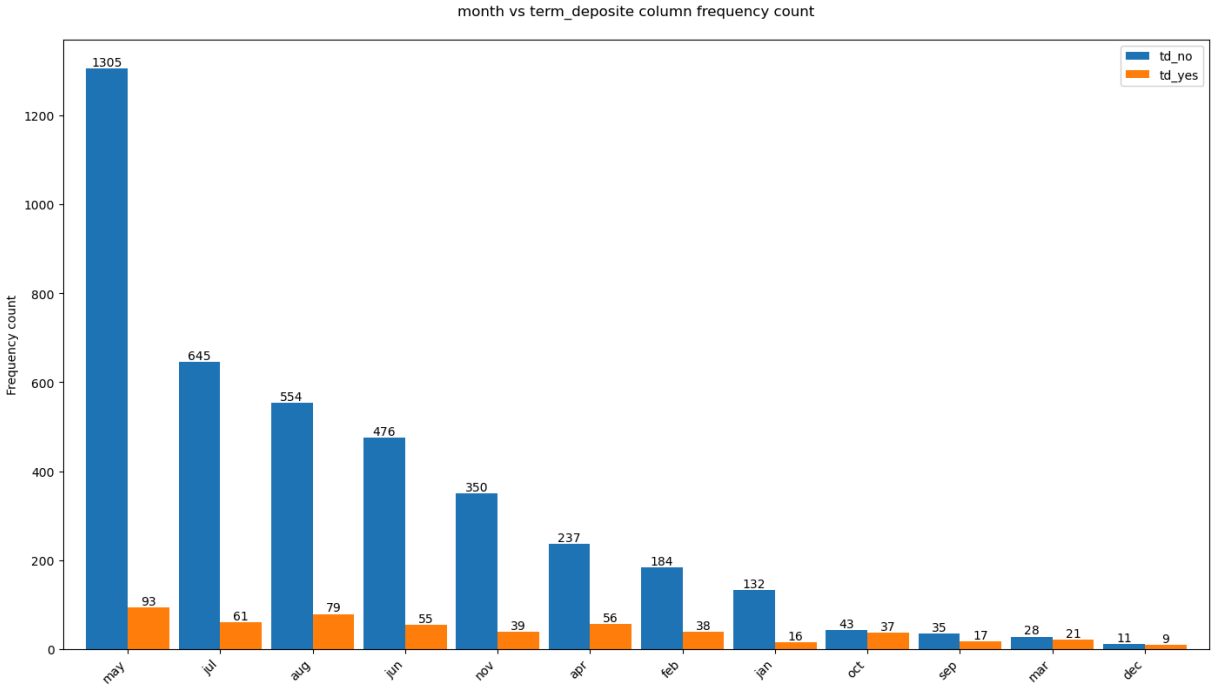
Out[118…

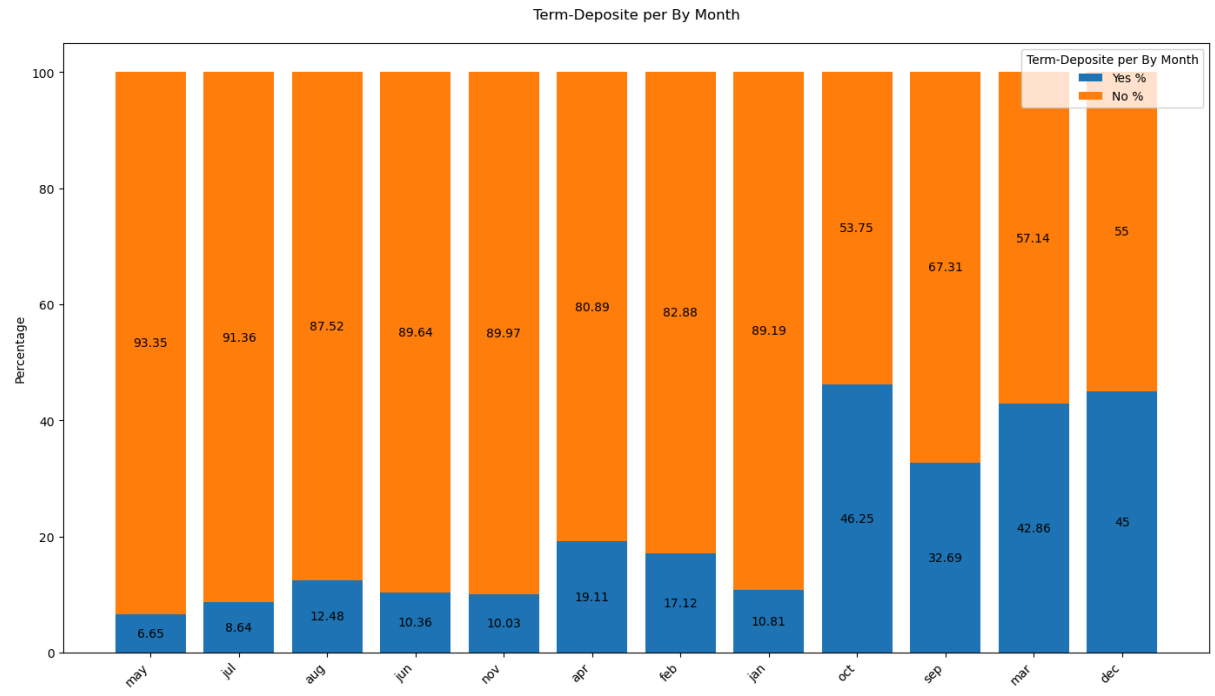
	month	td_yes	td_no	td_yes_per	td_no_per
1	may	93	1305	6.65	93.35
7	jul	61	645	8.64	91.36
5	aug	79	554	12.48	87.52
3	jun	55	476	10.36	89.64
8	nov	39	350	10.03	89.97
2	apr	56	237	19.11	80.89
4	feb	38	184	17.12	82.88
6	jan	16	132	10.81	89.19
0	oct	37	43	46.25	53.75
9	sep	17	35	32.69	67.31
10	mar	21	28	42.86	57.14
11	dec	9	11	45.00	55.00

In [118…

```
fig,chart=plt.subplot_mosaic([['top','top'],['bottom','bottom']],figsize=(17,20))

bar_chart=df[['month','td_no','td_yes']].set_index('month').plot(kind='bar',ax=char
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['top'].set_title("month vs term_deposite column frequency count",pad=20)
```

```
chart['top'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['top'].set_ylabel('Frequency count')
chart['top'].set_xlabel('')

b1=chart['bottom'].bar(df['month'], df['td_yes_per'])
plt.bar_label(b1,label_type='center', label='Yes %')
b2=chart['bottom'].bar(df['month'], df['td_no_per'], bottom=df['td_yes_per'], label
chart['bottom'].bar_label(b2,label_type='center')
chart['bottom'].set_title('Term-Deposite per By Month',pad=20)
chart['bottom'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Mont
chart['bottom'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right'
chart['bottom'].set_ylabel("Percentage")

plt.show()
```





Conclusion

- October, December, March, and September are the most effective months for subscription conversions, with subscription rates exceeding 30–50%.
- May, July, and August though having the most contacts but have very low conversion rates.
- This could indicate that quality of campaign strategies differ by month.
- spend more effort in October, December, March, and September to achieve better conversions.

In [617…

```
# pdays vs term_deposite column analysis

pdays_col=bank_df['pdays']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=pdays_col.unique()

for i in labels:
    con1=pdays_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['pdays','td_
df
```

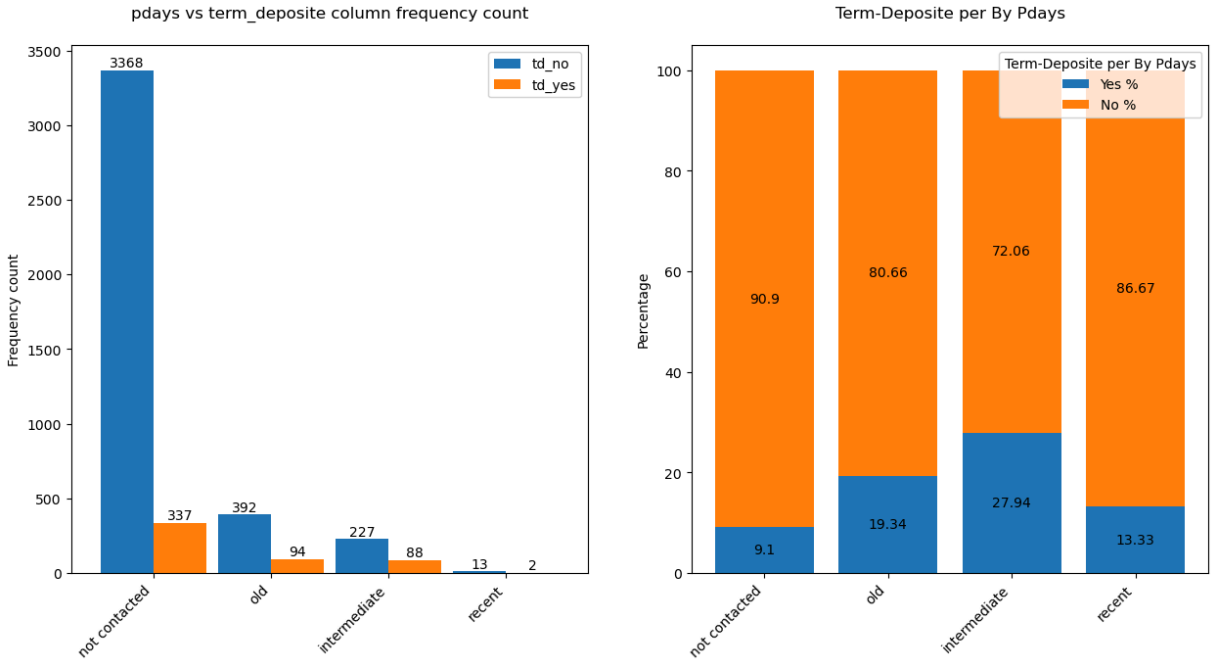
Out[617…

	pdays	td_yes	td_no	td_yes_per	td_no_per
0	not contacted	337	3368	9.10	90.90
1	old	94	392	19.34	80.66
2	intermediate	88	227	27.94	72.06
3	recent	2	13	13.33	86.67

In [618…

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['pdays','td_no','td_yes']].set_index('pdays').plot(kind='bar',ax=char
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("pdays vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['pdays'], df['td_yes_per'])
plt.bar_label(b1,label_type='center')
b2=chart['right'].bar(df['pdays'], df['td_no_per'], bottom=df['td_yes_per'])
chart['right'].bar_label(b2,label_type='center')
```

```
chart['right'].set_title('Term-Deposite per By Pdays',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Pdays
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```



Conclusion

- The "Intermediate" group (27.94%) shows the highest success rate for term deposit subscriptions. This suggests that people contacted a moderate time ago are more likely to convert for term-deposite.
- "Not contacted" is the largest group, but has a very low success rate (9.1%). This implies first-time contacts.
- "Old" and "Recent" contact history groups have moderate conversion, but still better than not contacted.
- Campaign strategies should focus more on those previously contacted, particularly intermediate and old contacts.

In [619…

```
# previous vs term_deposite column analysis

previous_col=bank_df['previous']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=previous_col.unique()

for i in labels:
    con1=previous_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'
```

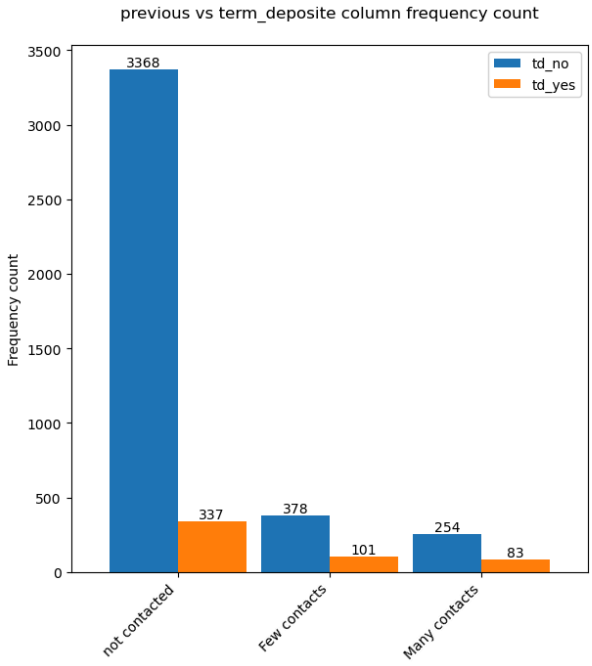
```
    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)

    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['previous','
df
```

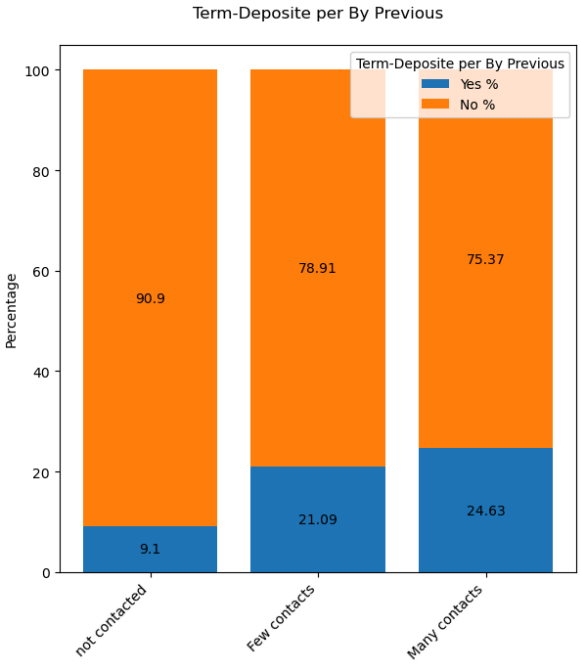
Out[619…

	previous	td_yes	td_no	td_yes_per	td_no_per
0	not contacted	337	3368	9.10	90.90
2	Few contacts	101	378	21.09	78.91
1	Many contacts	83	254	24.63	75.37

In [620…

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['previous','td_no','td_yes']].set_index('previous').plot(kind='bar',a
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("previous vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['previous'], df['td_yes_per'])
plt.bar_label(b1,label_type='center')
b2=chart['right'].bar(df['previous'], df['td_no_per'], bottom=df['td_yes_per'])
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Previous',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Previ
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```





Conclusion

- Previous interaction leads to higher subscription rates.
- Especially those with many past interactions have a conversion rate almost 3× higher than those not contacted.
- Focus on re-engaging customers who were contacted before.
- Avoid relying only on first-time. Less reach are much less effective.
- While the “not contacted” group is large, Try to reduce it and contact with client.
- “few” and “many” contacts are more valuable per contact.

In [621…

```
# poutcome vs term_deposite column analysis

poutcome_col=bank_df['poutcome']
term_deposite=bank_df['term_deposite']

yes_count,no_count,yes_per,no_per=[],[],[],[]
labels=poutcome_col.unique()

for i in labels:
    con1=poutcome_col==i
    con2=term_deposite=='yes'
    con3=term_deposite=='no'

    yes_con=con1 & con2
    no_con=con1 & con3
    total=len(bank_df[con1])

    yes_len=len(bank_df[yes_con])
    no_len=len(bank_df[no_con])
    y_per=round((yes_len/total)*100,2)
    n_per=round((no_len/total)*100,2)
```

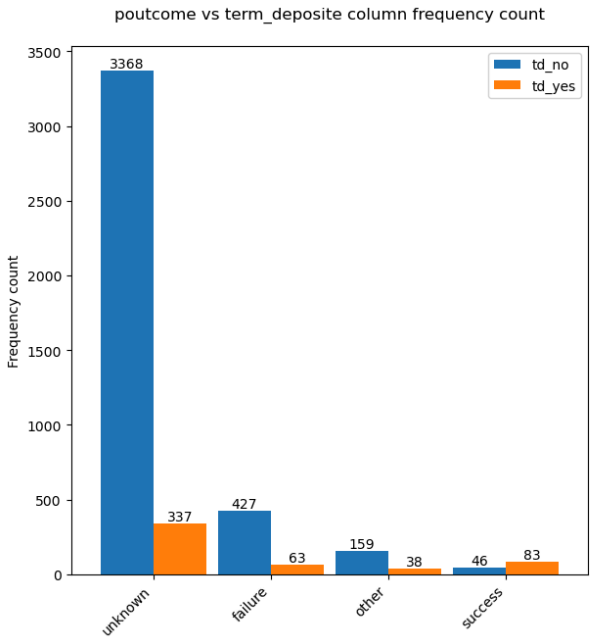
```
    yes_count.append(yes_len)
    no_count.append(no_len)
    yes_per.append(y_per)
    no_per.append(n_per)

df=pd.DataFrame(zip(labels,yes_count,no_count,yes_per,no_per),columns=['poutcome','
df
```

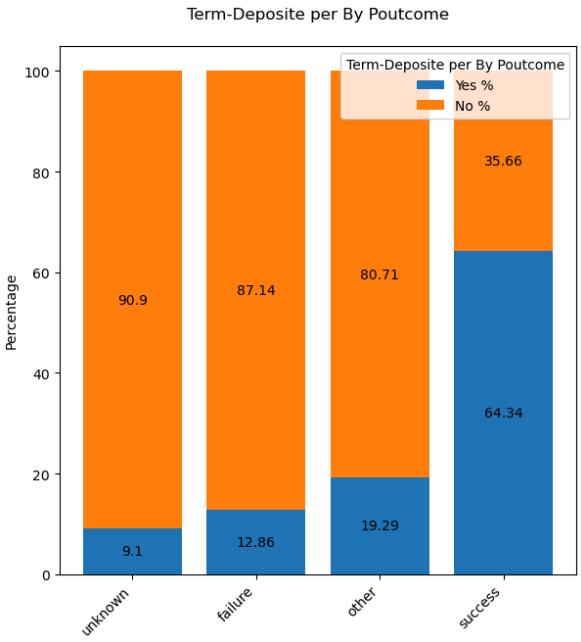
Out[621…

	poutcome	td_yes	td_no	td_yes_per	td_no_per
0	unknown	337	3368	9.10	90.90
1	failure	63	427	12.86	87.14
2	other	38	159	19.29	80.71
3	success	83	46	64.34	35.66

In [622…

```
fig,chart=plt.subplot_mosaic([['left','right'],['left','right']],figsize=(15,7))

bar_chart=df[['poutcome','td_no','td_yes']].set_index('poutcome').plot(kind='bar',a
for i in bar_chart.containers:
    bar_chart.bar_label(i)

chart['left'].set_title("poutcome vs term_deposite column frequency count",pad=20)
chart['left'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45, ha='right')
chart['left'].set_ylabel('Frequency count')
chart['left'].set_xlabel('')

b1=chart['right'].bar(df['poutcome'], df['td_yes_per'])
plt.bar_label(b1,label_type='center')
b2=chart['right'].bar(df['poutcome'], df['td_no_per'], bottom=df['td_yes_per'])
chart['right'].bar_label(b2,label_type='center')
chart['right'].set_title('Term-Deposite per By Poutcome',pad=20)
chart['right'].legend([b1, b2], ['Yes %', 'No %'],title="Term-Deposite per By Poutc
chart['right'].set_xticklabels(bar_chart.get_xticklabels(), rotation=45,ha='right')
chart['right'].set_ylabel("Percentage")

plt.show()
```





Conclusion

- Success in previous campaigns leads to a higher of term deposit subscription.
- Unknown and Failure outcomes correlate with low subscription rates.
- Despite the majority of data falling into the "unknown" category, its conversion rate is the lowest.
- Targeting people with previously successful outcomes can improve marketing efficiency.

Checking the affect of numerical columns on the target variable

In [446...
```
num_col
```

Out[446...
```
Index(['age', 'balance', 'day', 'duration', 'campaign'], dtype='object')
```

In [627...
```
bank_df.sample()
```

Out[627...

	age	job	marital	education	default	balance	housing	loan	contact	day	m
2617	41.0	services	married	secondary	no	171.0	no	no	cellular	21	

In [886...
```
for i in num_col:
    yes = bank_df[bank_df['term_deposite'] == 'yes'][i]
    no = bank_df[bank_df['term_deposite'] == 'no'][i]

    t_stat, p_val = ttest_ind(yes, no, equal_var=False)
    print(f"T-statistic of column {i} and term_deposite: {t_stat}, P-value of colum
```

```
T-statistic of column age and term_deposite: 0.4668753110526079, P-value of column a
ge and term_deposite: 0.6407502659395974
T-statistic of column balance and term_deposite: 4.499094830988442, P-value of colum
n balance and term_deposite: 8.120892800657694e-06
T-statistic of column day and term_deposite: -0.7569533024030284, P-value of column
day and term_deposite: 0.449346606766852
T-statistic of column duration and term_deposite: 14.036207417808788, P-value of col
umn duration and term_deposite: 3.128278260791641e-39
T-statistic of column campaign and term_deposite: -3.049733264633748, P-value of col
umn campaign and term_deposite: 0.002379863273070066
```

Conclusion from above code

duration

- Highly significant (p-value < 0.05)
- Strongest relationship with term_deposit
- Longer call durations are strongly associated with higher subscription rates
- Important for analysis and prediction

balance

- Statistically significant (p-value < 0.05)
- Customers with higher balances are more likely to subscribe
- Financial stability plays a role in decision-making
- Important for analysis

campaign

- Statistically significant (p-value < 0.05)
- Number of times a client was contacted matters
- Too many contacts may negatively affect subscription rates
- Useful for analysis

age

- Not significant (p-value >= 0.05)
- No meaningful difference in age between those who subscribed and those who didn't
- Less useful for predicting outcome

day

- Not significant (p-value >= 0.05)
- The day of the month the client was contacted does not impact subscription
- Not recommended for analysis

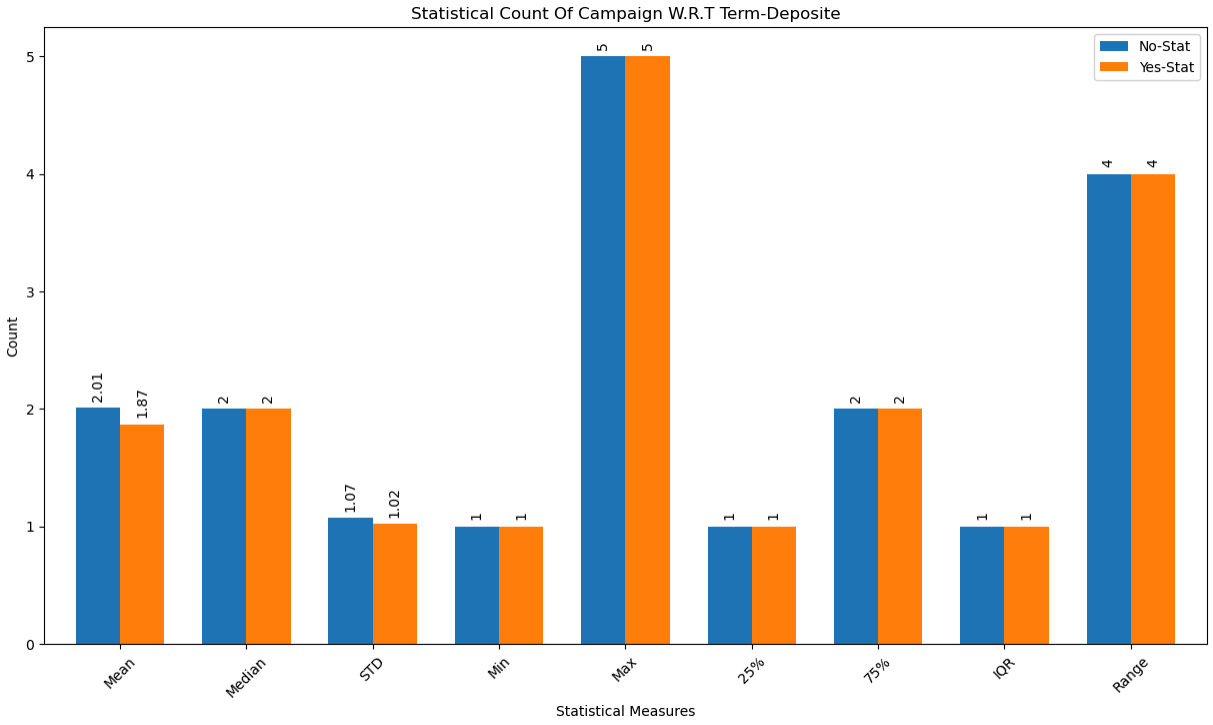
so we will use duration, campaign and balance with term_deposite column for bivariate analysis

In [119…
```
# Campaign vs Term_deposite column analysis

cam_stat=bank_df.groupby('term_deposite')['campaign'].describe()
count=cam_stat['count']
mean=cam_stat['mean']
median=cam_stat['50%']
std=round(cam_stat['std'],2)
MIN=cam_stat['min']
MAX=cam_stat['max']
p_25=cam_stat['25%']
p_75=cam_stat['75%']
index=cam_stat.index
a_range=MAX-MIN
IQR=p_75-p_25

col=["Count","Mean","Median","STD","Min","Max","25%","75%","IQR","Range"]
no=[count[0],round(mean[0],2),median[0],round(std[0],2),MIN[0],MAX[0],p_25[0],p_75[
yes=[count[1],round(mean[1],2),median[1],round(std[1],2),MIN[1],MAX[1],p_25[1],p_75

df=pd.DataFrame(zip(no,yes),columns=['No-Stat','Yes-Stat'],index=col)
df
```

Out[119…

	No-Stat	Yes-Stat
Count	4000.00	521.00
Mean	2.01	1.87
Median	2.00	2.00
STD	1.07	1.02
Min	1.00	1.00
Max	5.00	5.00
25%	1.00	1.00
75%	2.00	2.00
IQR	1.00	1.00
Range	4.00	4.00

In [895…
```
fig,ax=plt.subplots(1,1,figsize=(15,8))
bar=df[1:].plot(kind='bar',ax=ax,width=0.7)
for i in bar.containers:
    bar.bar_label(i,rotation=90,padding=5)
ax.set_xlabel("Statistical Measures")
ax.set_ylabel("Count")
ax.set_title("Statistical Count Of Campaign W.R.T Term-Deposite")
plt.tick_params(axis='x',rotation=45)
plt.show()
```



Conclusion

- The average of the cutomer contact who subsribed and unsubscribed is nearly about same.
- All other statistics are nearly identical, indicating that the distribution of campaign contacts is very similar between the two groups.

In [119…

```
cam=bank_df['campaign']
cam_max=cam.max()
cam_min=cam.min()
cam_num=len(cam)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((cam_max-cam_min)/count),2)
bins=list(np.arange(cam_min,cam_max+interval_gap,interval_gap))
camp_bins=bins

cam_group = pd.cut(cam, bins=bins, right=False)
keys=[f"{i.left}-{i.right}" for i in cam_group]

cam_df=pd.DataFrame(zip(keys,bank_df['term_deposite']),columns=["Campaign-Group","T

cam_table = cam_df.groupby('Campaign-Group')['Term-Deposite'].value_counts().unstac
cam_table['Total'] = cam_table.sum(axis=1)
cam_table['td_yes_per'] = round(cam_table['yes'] / cam_table['Total'] * 100, 2)
cam_table['td_no_per'] = round(cam_table['no'] / cam_table['Total'] * 100, 2)
```

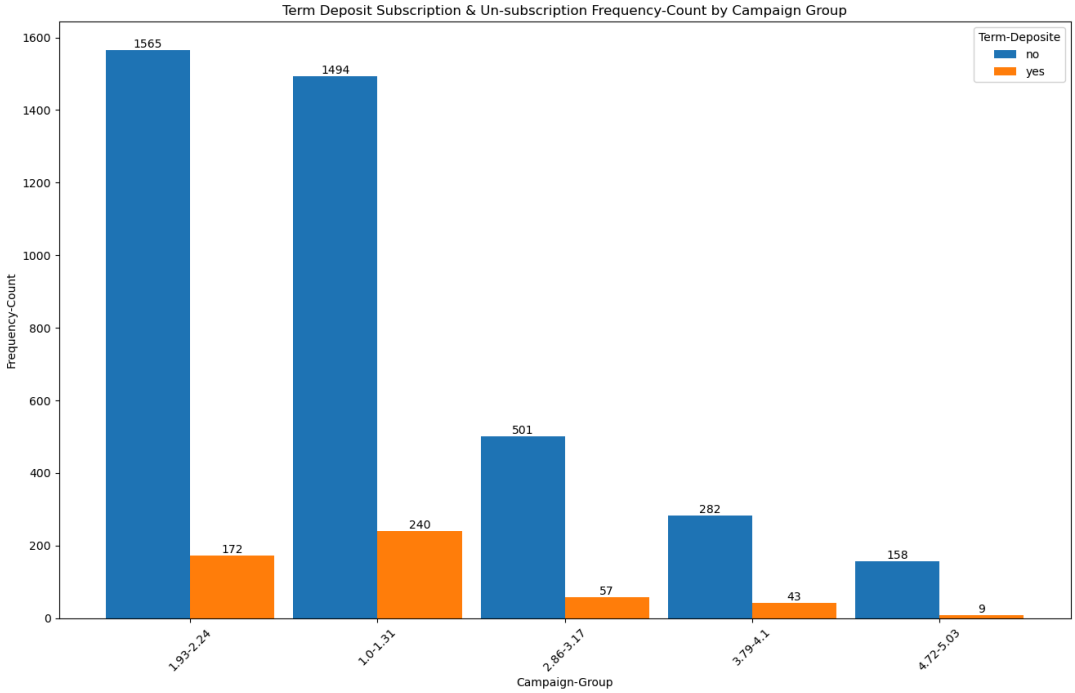
```
cam_table.reset_index(inplace=True)
cam_table=cam_table.sort_values(by=["Total"],ascending=False)
cam_table
```

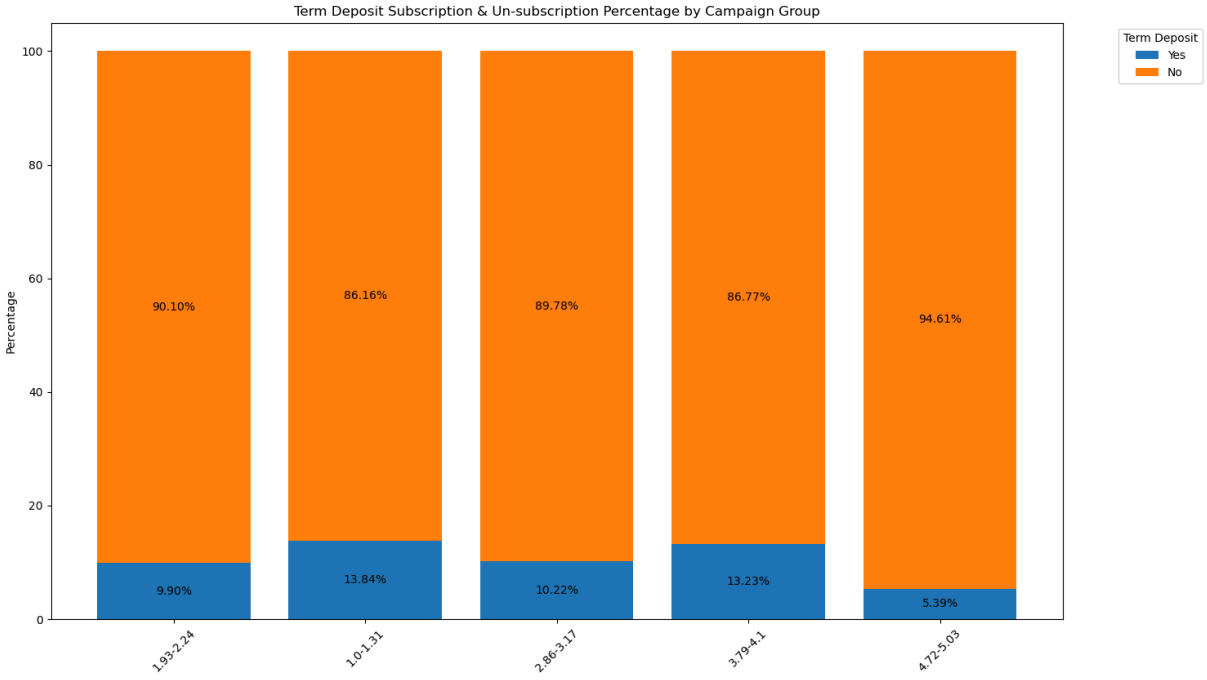
Out[119…

	Term-Deposite	Campaign-Group	no	yes	Total	td_yes_per	td_no_per
1		1.93-2.24	1565	172	1737	9.90	90.10
0		1.0-1.31	1494	240	1734	13.84	86.16
2		2.86-3.17	501	57	558	10.22	89.78
3		3.79-4.1	282	43	325	13.23	86.77
4		4.72-5.03	158	9	167	5.39	94.61

In [119…

```
fig,ax = plt.subplots(2,1,figsize=(15,17))
bar=cam_table[['Campaign-Group','no','yes']].set_index('Campaign-Group').plot(kind=
for i in bar.containers:
    bar.bar_label(i)
ax[0].set_ylabel('Frequency-Count')
ax[0].set_title('Term Deposit Subscription & Un-subscription Frequency-Count by Cam
ax[0].tick_params(axis='x', rotation=45)

bar1=ax[1].bar(cam_table['Campaign-Group'], cam_table['td_yes_per'], label='Yes')
bar2=ax[1].bar(cam_table['Campaign-Group'], cam_table['td_no_per'], bottom=cam_tabl
ax[1].set_ylabel('Percentage')
ax[1].set_title('Term Deposit Subscription & Un-subscription Percentage by Campaign
ax[1].bar_label(bar1, label_type='center', fmt='%.2f%%', rotation=0)
ax[1].bar_label(bar2, label_type='center', fmt='%.2f%%', rotation=0)
ax[1].tick_params(axis='x', rotation=45)
ax[1].legend(title="Term Deposit", bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
plt.show()
```





conclusion

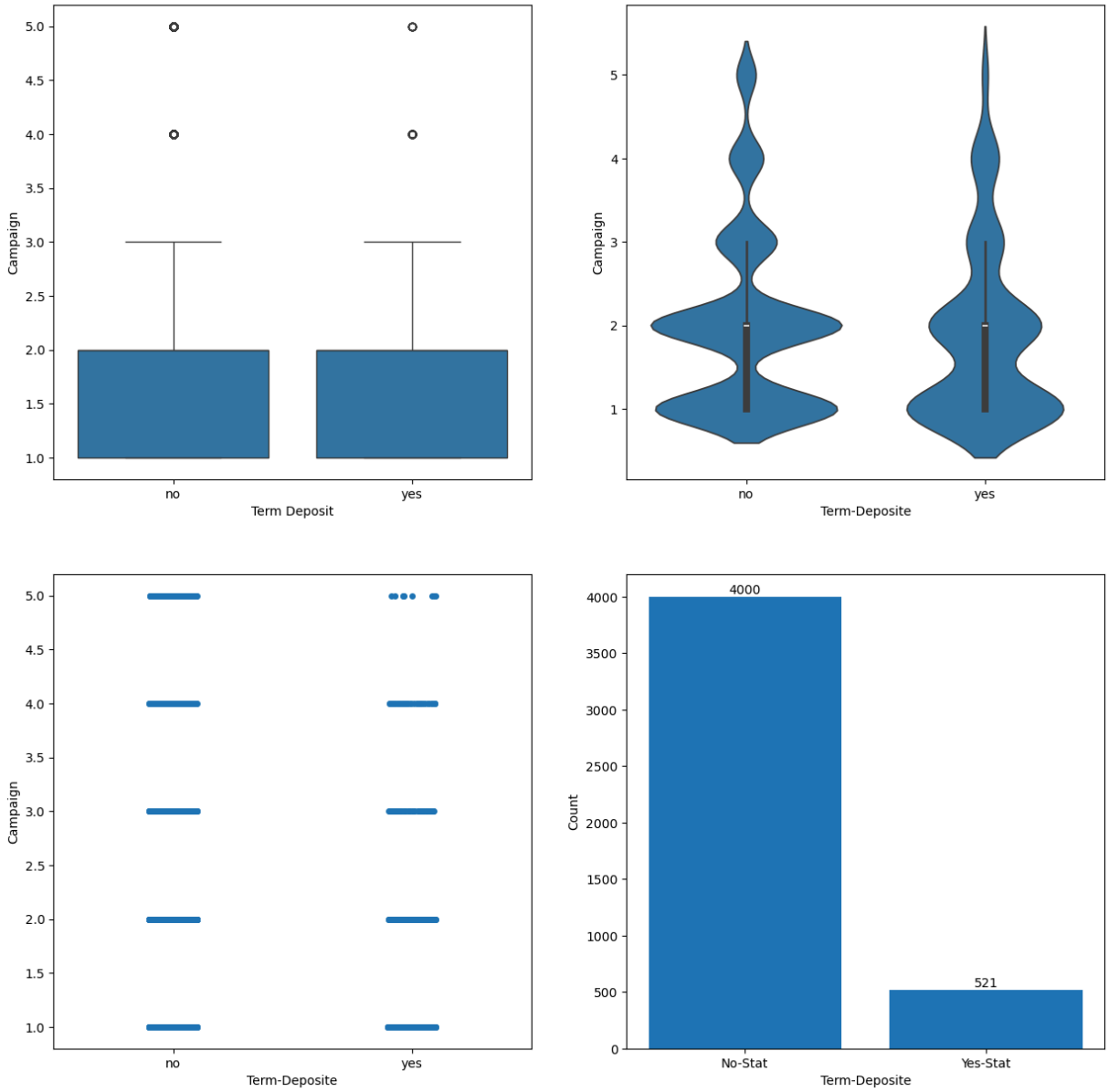
- Customers contacted 1.59–2.84 times or 1.0–1.51 times show the highest subscription counts.
- Beyond 3.79 contacts, subscription rates drop significantly.
- 1–3 contact attempts yield the highest success rate suggesting minimize over-contact.
- As the number of contacts increases, unsubscription rate rises sharply

In [119…

```
fig,ax=plt.subplots(2,2,figsize=(15,15))
sns.boxplot(x='term_deposite', y='campaign', data=bank_df, ax=ax[0][0])
fig.suptitle("Distribution of Campaign for Term Deposit")
ax[0][0].set_xlabel("Term Deposit")
```

```
ax[0][0].set_ylabel("Campaign")

sns.violinplot(x='term_deposite',y='campaign',data=bank_df,ax=ax[0][1])
ax[0][1].set_xlabel("Term-Deposite")
ax[0][1].set_ylabel("Campaign")

sns.stripplot(x='term_deposite', y='campaign', data=bank_df, ax=ax[1][0])
ax[1][0].set_xlabel("Term-Deposite")
ax[1][0].set_ylabel("Campaign")

bar_count=ax[1][1].bar(['No-Stat','Yes-Stat'],[df['No-Stat'][0],df['Yes-Stat'][0]])
plt.bar_label(bar_count)
plt.xlabel("Term-Deposite")
plt.ylabel("Count")

plt.show()
```

Distribution of Campaign for Term Deposit



Conclusion

From Box Plot

- The median number of contacts is almost the same (~1.5–2) for both groups.
- The interquartile range (IQR) for both is tight, showing that most people were contacted 1–3 times.
- Outliers exist in both cases, especially for those contacted more than 3 times, but this didn't significantly affect the outcome.

Violin Plot

- Both groups peak around 1–2 contact attempts.

- Slightly higher density near 2 contacts for yes group, reinforcing the sweet spot for conversion.
- The distribution is symmetrical, suggesting that the behavior of the campaign contacts is consistent across both yes and no responses.
- However, the density is thicker at lower campaign values for 'yes', showing shorter campaigns are more effective.

Strip Plot

- Most data points cluster around 1–3 contacts, with very few going beyond 4.
- The 'yes' class shows concentration at lower contact attempts, and less frequent higher contact points.
- contacting customers 1–2 times is most effective, and repeated efforts do not yield proportionate benefits

Bar Plot

- 4000 customers did not subscribe, while only 521 did.
- This imbalance could influence any ML models trained on this data, requiring techniques like resampling or class balancing.
- conversion rates remain low (~11.5%)—highlighting a need for different campaign strategies.

In [119…

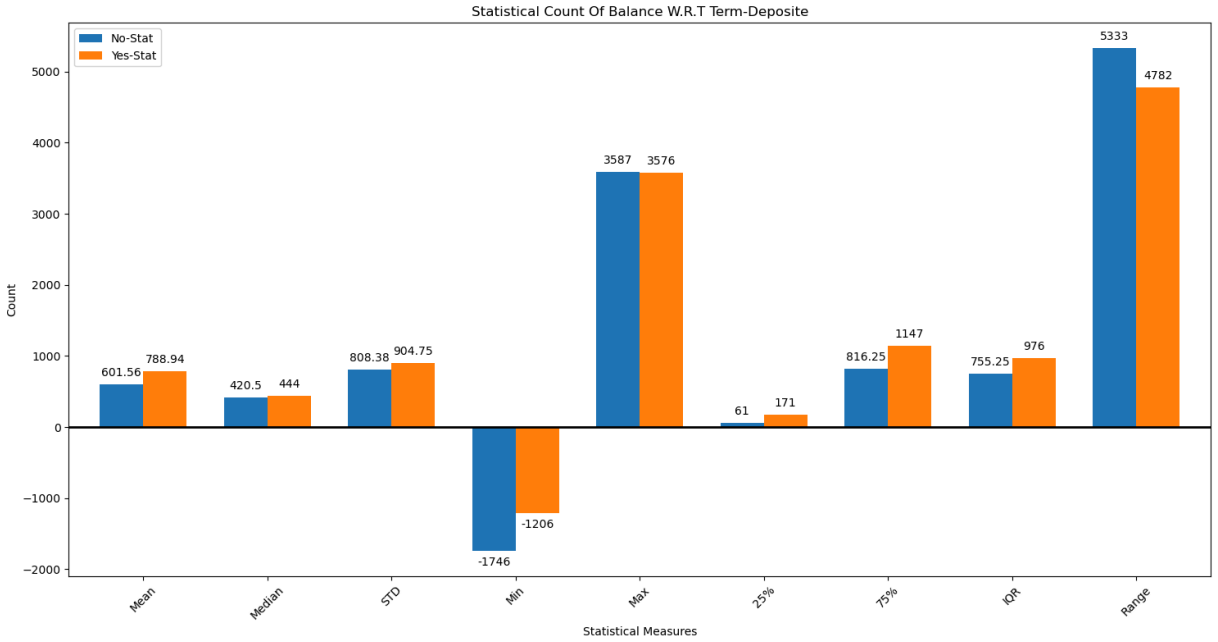
```
# Balance vs Term_deposite column analysis

bal_stat=bank_df.groupby('term_deposite')['balance'].describe()
count=bal_stat['count']
mean=bal_stat['mean']
median=bal_stat['50%']
std=round(bal_stat['std'],2)
MIN=bal_stat['min']
MAX=bal_stat['max']
p_25=bal_stat['25%']
p_75=bal_stat['75%']
index=bal_stat.index
a_range=MAX-MIN
IQR=p_75-p_25

col=["Count","Mean","Median","STD","Min","Max","25%","75%","IQR","Range"]
no=[count[0],round(mean[0],2),median[0],round(std[0],2),MIN[0],MAX[0],p_25[0],p_75[
yes=[count[1],round(mean[1],2),median[1],round(std[1],2),MIN[1],MAX[1],p_25[1],p_75

df=pd.DataFrame(zip(no,yes),columns=['No-Stat','Yes-Stat'],index=col)
df
```

Out[119...

	No-Stat	Yes-Stat
Count	4000.00	521.00
Mean	601.56	788.94
Median	420.50	444.00
STD	808.38	904.75
Min	-1746.00	-1206.00
Max	3587.00	3576.00
25%	61.00	171.00
75%	816.25	1147.00
IQR	755.25	976.00
Range	5333.00	4782.00

In [878...

```
fig,ax=plt.subplots(1,1,figsize=(15,8))
bar=df[1:].plot(kind='bar',ax=ax,width=0.7)
for i in bar.containers:
    bar.bar_label(i,rotation=0,padding=5)
ax.set_xlabel("Statistical Measures")
ax.set_ylabel("Count")
ax.set_title("Statistical Count Of Balance W.R.T Term-Deposite")
ax.tick_params(axis='x',rotation=45)

plt.axhline(0, color='black', linestyle='-', linewidth=2)
plt.tight_layout()
plt.show()
```



Conclusion

- People with higher average and median balances are more likely to subscribe.
- Suggests Yes-Stat customers are wealthier across all quartiles.
- Negative minimum balances exist in both groups, but more extreme for No-Stat
- Negative minimum balances exist in both groups, but more extreme for No-Stat

In [119…

```
bal=bank_df['balance']
bal_max=bal.max()
bal_min=bal.min()
bal_num=len(bal)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((bal_max-bal_min)/count),2)
bins=list(np.arange(bal_min,bal_max+interval_gap,interval_gap))
bal_bins=bins

bal_group = pd.cut(bal, bins=bins, right=False)
keys=[f"{i.left}-{i.right}" for i in bal_group]

bal_df=pd.DataFrame(zip(keys,bank_df['term_deposite']),columns=["Balance-Group","Te

bal_table = bal_df.groupby('Balance-Group')['Term-Deposite'].value_counts().unstack
bal_table['Total'] = bal_table.sum(axis=1)
bal_table['td_yes_per'] = round(bal_table['yes'] / bal_table['Total'] * 100, 2)
bal_table['td_no_per'] = round(bal_table['no'] / bal_table['Total'] * 100, 2)
bal_table.reset_index(inplace=True)
bal_table=bal_table.sort_values(by=["Total"],ascending=False)
bal_table
```

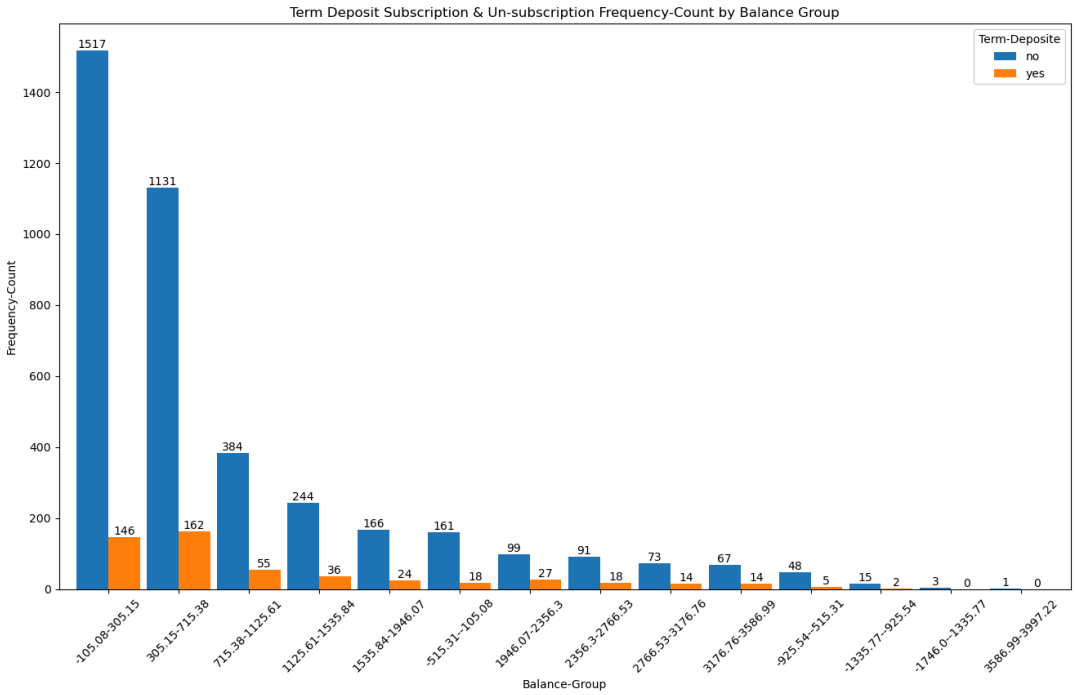
Out[119…

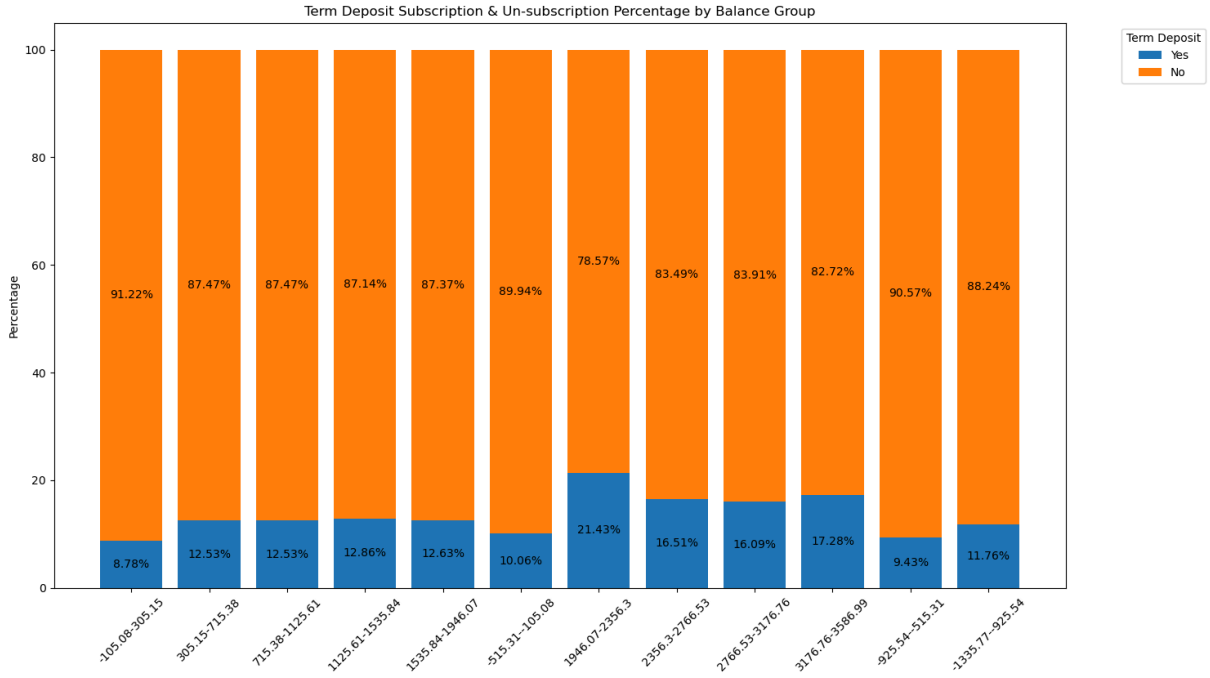
	Term-Deposite	Balance-Group	no	yes	Total	td_yes_per	td_no_per
0		-105.08-305.15	1517.0	146.0	1663.0	8.78	91.22
10		305.15-715.38	1131.0	162.0	1293.0	12.53	87.47
13		715.38-1125.61	384.0	55.0	439.0	12.53	87.47
5		1125.61-1535.84	244.0	36.0	280.0	12.86	87.14
6		1535.84-1946.07	166.0	24.0	190.0	12.63	87.37
3		-515.31--105.08	161.0	18.0	179.0	10.06	89.94
7		1946.07-2356.3	99.0	27.0	126.0	21.43	78.57
8		2356.3-2766.53	91.0	18.0	109.0	16.51	83.49
9		2766.53-3176.76	73.0	14.0	87.0	16.09	83.91
11		3176.76-3586.99	67.0	14.0	81.0	17.28	82.72
4		-925.54--515.31	48.0	5.0	53.0	9.43	90.57
1		-1335.77--925.54	15.0	2.0	17.0	11.76	88.24
2		-1746.0--1335.77	3.0	NaN	3.0	NaN	100.00
12		3586.99-3997.22	1.0	NaN	1.0	NaN	100.00

In [119…

```
fig,ax = plt.subplots(2,1,figsize=(15,17))
bar=bal_table[['Balance-Group','no','yes']].set_index('Balance-Group').plot(kind='b
for i in bar.containers:
    bar.bar_label(i)
ax[0].set_ylabel('Frequency-Count')
ax[0].set_title('Term Deposit Subscription & Un-subscription Frequency-Count by Bal
ax[0].tick_params(axis='x', rotation=45)

bar1=ax[1].bar(bal_table['Balance-Group'], bal_table['td_yes_per'], label='Yes')
bar2=ax[1].bar(bal_table['Balance-Group'], bal_table['td_no_per'], bottom=bal_table
ax[1].set_ylabel('Percentage')
ax[1].set_title('Term Deposit Subscription & Un-subscription Percentage by Balance
ax[1].bar_label(bar1, label_type='center', fmt='%.2f%%', rotation=0)
ax[1].bar_label(bar2, label_type='center', fmt='%.2f%%', rotation=0)
ax[1].tick_params(axis='x', rotation=45)
ax[1].legend(title="Term Deposit", bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
plt.show()
```

```
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
posx and posy should be finite values
```





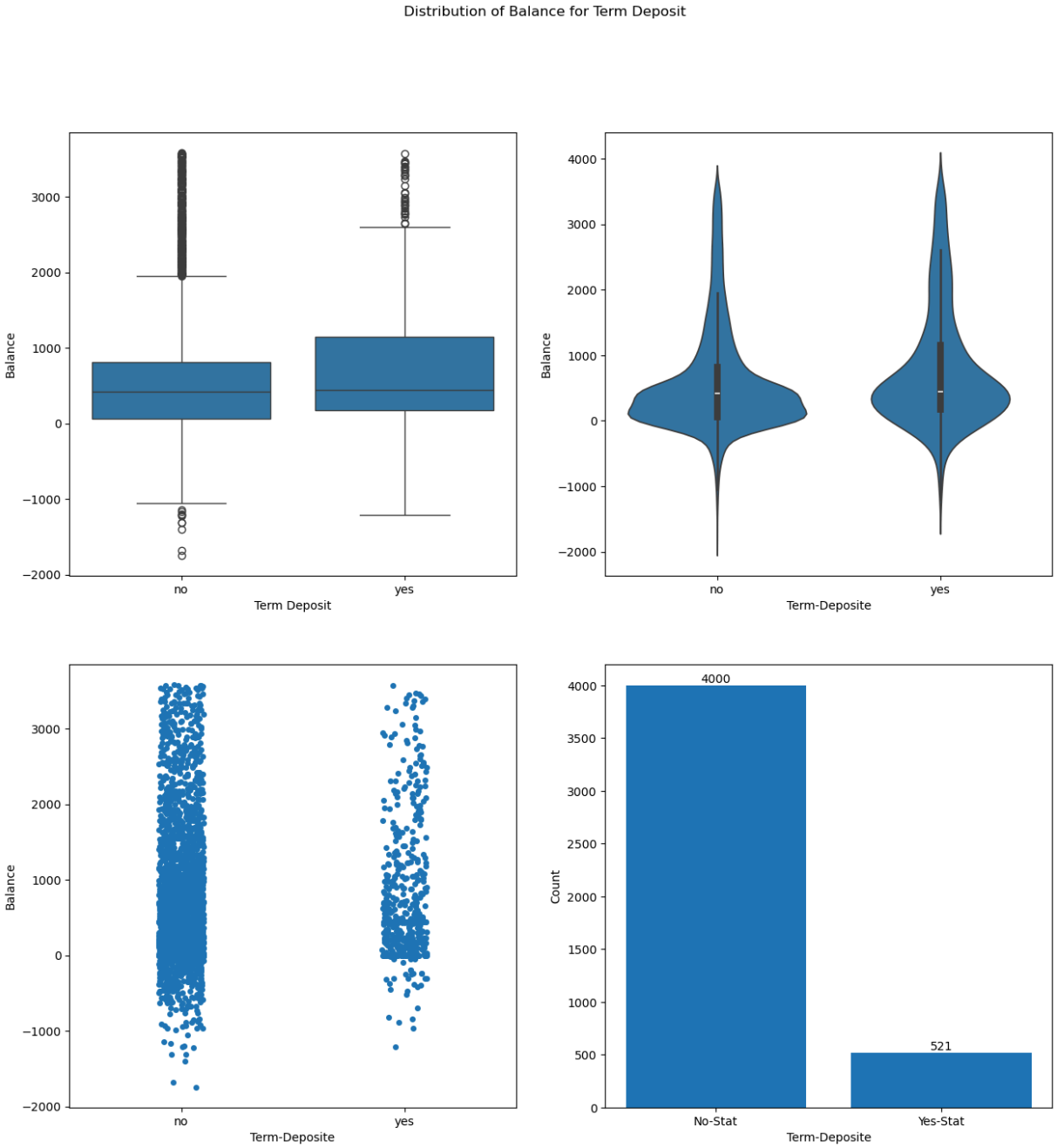
Conclusion

- Higher balance more increasing subscriber to term deposits.
- Most subscriptions come from moderate to high balance groups.
- Lower balance groups (especially negative or <₹500) show very low subscription rates (~8–13%).
- Although the majority of customers fall into low balance ranges, they rarely subscribe.
- Subscription percentage increases with balance up to a point, then fluctuates slightly due to fewer data points.

In [120…

```
fig,ax=plt.subplots(2,2,figsize=(15,15))
sns.boxplot(x='term_deposite', y='balance', data=bank_df, ax=ax[0][0])
```

```
fig.suptitle("Distribution of Balance for Term Deposit")
ax[0][0].set_xlabel("Term Deposit")
ax[0][0].set_ylabel("Balance")

sns.violinplot(x='term_deposite',y='balance',data=bank_df,ax=ax[0][1])
ax[0][1].set_xlabel("Term-Deposite")
ax[0][1].set_ylabel("Balance")

sns.stripplot(x='term_deposite', y='balance', data=bank_df, ax=ax[1][0])
ax[1][0].set_xlabel("Term-Deposite")
ax[1][0].set_ylabel("Balance")

bar_count=ax[1][1].bar(['No-Stat','Yes-Stat'],[df['No-Stat'][0],df['Yes-Stat'][0]])
plt.bar_label(bar_count)
plt.xlabel("Term-Deposite")
plt.ylabel("Count")

plt.show()
```



Conclusion

- Median balance is higher for customers who subscribed to term deposits ("yes") compared to non-subscribers ("no").
- Subscribers have fewer negative balances, non-subscribers show more extreme negative outliers.
- The violin plot shows that the balance distribution is more concentrated around 0 for non-subscribers.
- The swarm plot reinforces that higher balances are more common among subscribers.
- Far fewer people subscribe to term deposits (521) than those who don't (4000).

In [120…

```
# Duration vs Term_deposite column analysis
```

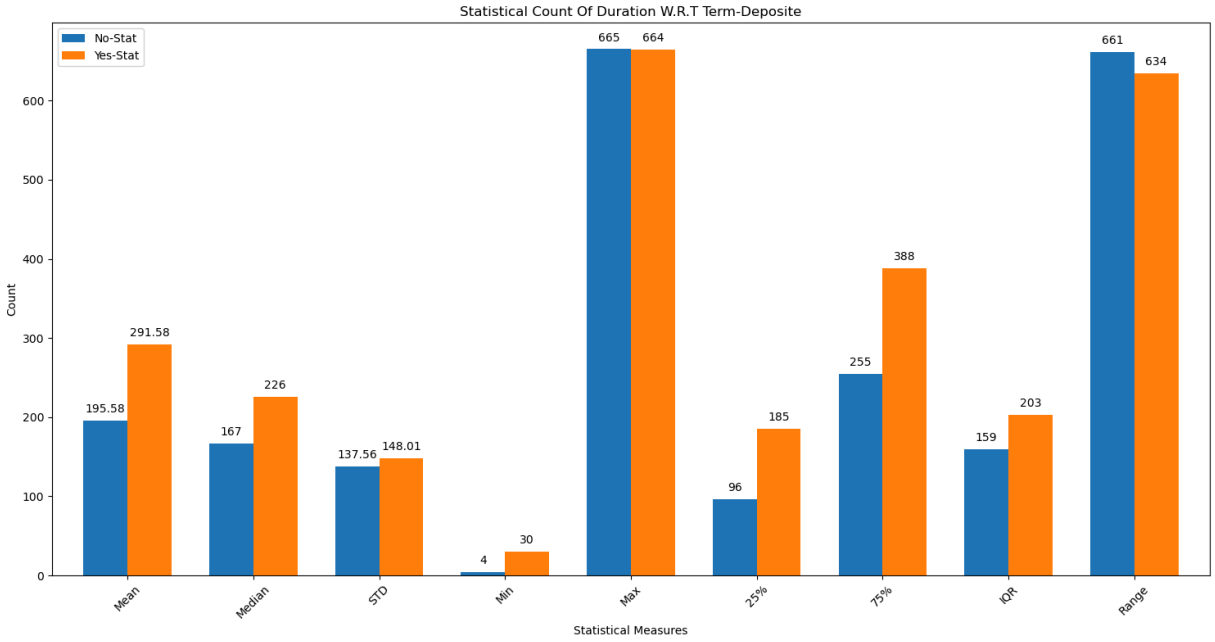
```
dur_stat=bank_df.groupby('term_deposite')['duration'].describe()
count=dur_stat['count']
mean=dur_stat['mean']
median=dur_stat['50%']
std=round(dur_stat['std'],2)
MIN=dur_stat['min']
MAX=dur_stat['max']
p_25=dur_stat['25%']
p_75=dur_stat['75%']
index=dur_stat.index
a_range=MAX-MIN
IQR=p_75-p_25

col=["Count","Mean","Median","STD","Min","Max","25%","75%","IQR","Range"]
no=[count[0],round(mean[0],2),median[0],round(std[0],2),MIN[0],MAX[0],p_25[0],p_75[
yes=[count[1],round(mean[1],2),median[1],round(std[1],2),MIN[1],MAX[1],p_25[1],p_75

df=pd.DataFrame(zip(no,yes),columns=['No-Stat','Yes-Stat'],index=col)
df
```

Out[120…

	No-Stat	Yes-Stat
Count	4000.00	521.00
Mean	195.58	291.58
Median	167.00	226.00
STD	137.56	148.01
Min	4.00	30.00
Max	665.00	664.00
25%	96.00	185.00
75%	255.00	388.00
IQR	159.00	203.00
Range	661.00	634.00

In [889…

```
fig,ax=plt.subplots(1,1,figsize=(15,8))
bar=df[1:].plot(kind='bar',ax=ax,width=0.7)
for i in bar.containers:
    bar.bar_label(i,rotation=0,padding=5)
ax.set_xlabel("Statistical Measures")
ax.set_ylabel("Count")
ax.set_title("Statistical Count Of Duration W.R.T Term-Deposite")
ax.tick_params(axis='x',rotation=45)

plt.tight_layout()
plt.show()
```



Conclusion

- Duration is higher on average (mean, median, 25%, 75%) for customers who subscribed ("Yes-Stat").
- Minimum duration is much lower for non-subscribers.
- Subscribers tend to have longer call durations, indicating stronger engagement.
- Higher IQR and mean show more consistent and longer durations for subscribers.

In [120…

```
dur=bank_df['duration']
dur_max=dur.max()
dur_min=dur.min()
dur_num=len(dur)

count=1
while True:
    intervals=2**count
    if intervals>total_num:
        break
    count+=1

interval_gap=round(((dur_max-dur_min)/count),2)
bins=list(np.arange(dur_min,dur_max+interval_gap,interval_gap))
dur_bins=bins

dur_group = pd.cut(dur, bins=bins, right=False)
keys=[f"{i.left}-{i.right}" for i in dur_group]

dur_df=pd.DataFrame(zip(keys,bank_df['term_deposite']),columns=["Duration-Group","T

dur_table = dur_df.groupby('Duration-Group')['Term-Deposite'].value_counts().unstac
dur_table['Total'] = dur_table.sum(axis=1)
dur_table['td_yes_per'] = round(dur_table['yes'] / dur_table['Total'] * 100, 2)
dur_table['td_no_per'] = round(dur_table['no'] / dur_table['Total'] * 100, 2)
dur_table.reset_index(inplace=True)
```

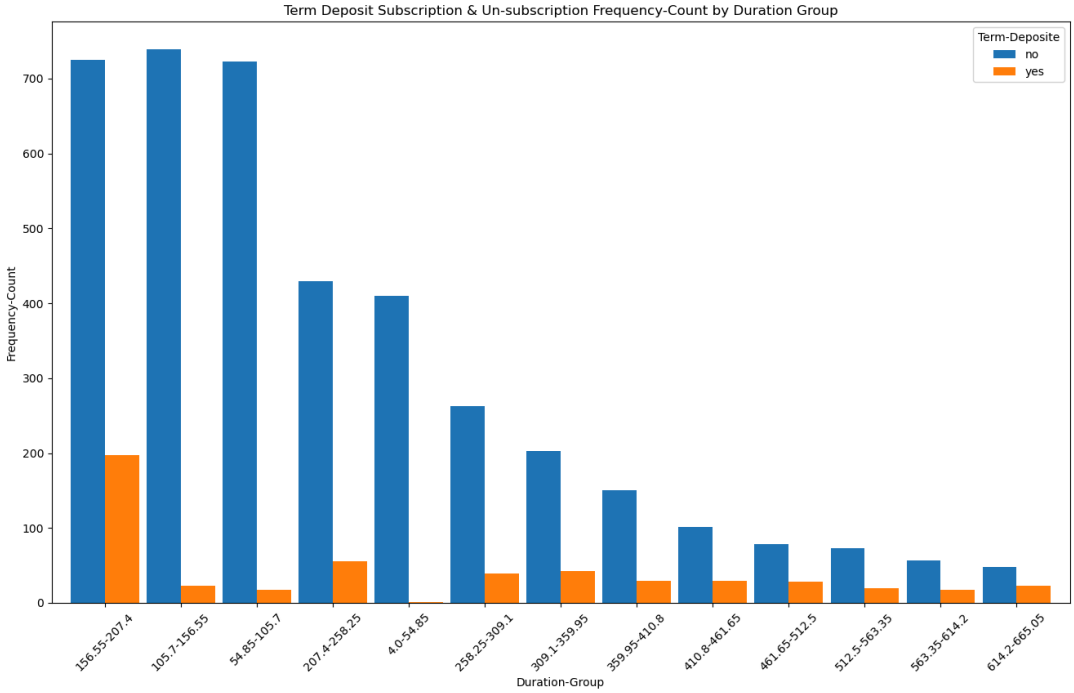
```
dur_table=dur_table.sort_values(by=["Total"],ascending=False)
dur_table
```

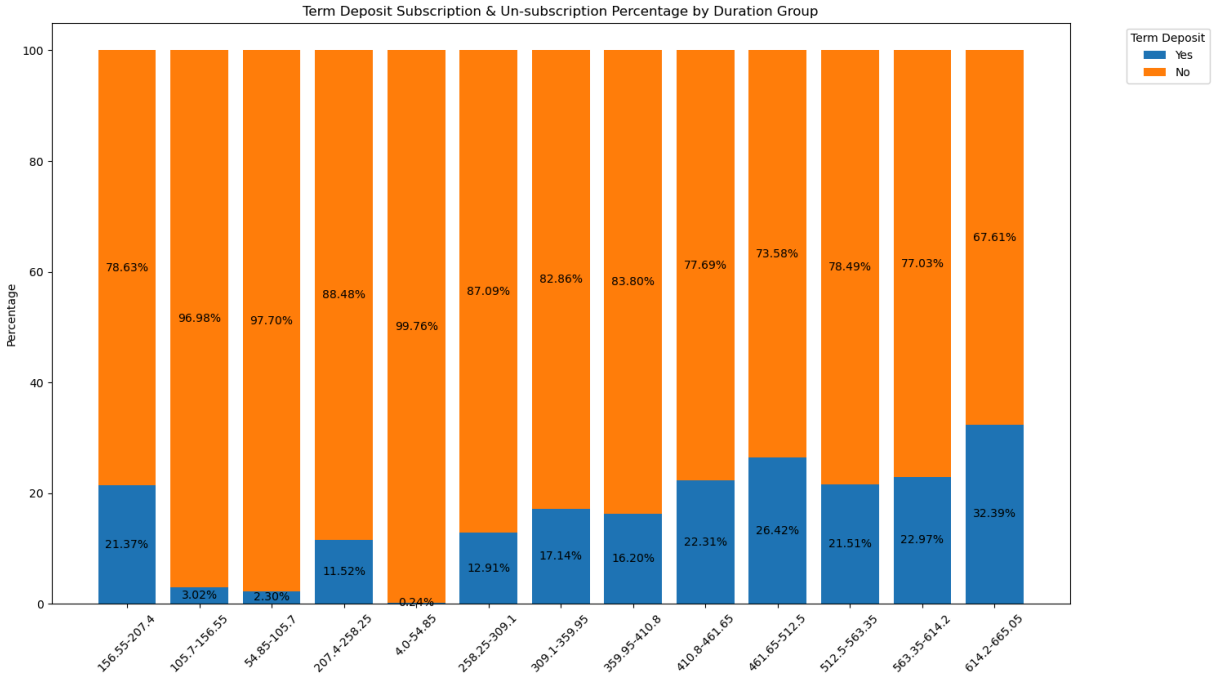
Out[120…

Term-Deposite	Duration-Group	no	yes	Total	td_yes_per	td_no_per
1	156.55-207.4	725	197	922	21.37	78.63
0	105.7-156.55	739	23	762	3.02	96.98
10	54.85-105.7	723	17	740	2.30	97.70
2	207.4-258.25	430	56	486	11.52	88.48
6	4.0-54.85	410	1	411	0.24	99.76
3	258.25-309.1	263	39	302	12.91	87.09
4	309.1-359.95	203	42	245	17.14	82.86
5	359.95-410.8	150	29	179	16.20	83.80
7	410.8-461.65	101	29	130	22.31	77.69
8	461.65-512.5	78	28	106	26.42	73.58
9	512.5-563.35	73	20	93	21.51	78.49
11	563.35-614.2	57	17	74	22.97	77.03
12	614.2-665.05	48	23	71	32.39	67.61

In [120…

```
fig,ax = plt.subplots(2,1,figsize=(15,17))
dur=dur_table[['Duration-Group','no','yes']].set_index('Duration-Group').plot(kind=
for i in bar.containers:
    bar.bar_label(i)
ax[0].set_ylabel('Frequency-Count')
ax[0].set_title('Term Deposit Subscription & Un-subscription Frequency-Count by Dur
ax[0].tick_params(axis='x', rotation=45)

bar1=ax[1].bar(dur_table['Duration-Group'], dur_table['td_yes_per'], label='Yes')
bar2=ax[1].bar(dur_table['Duration-Group'], dur_table['td_no_per'], bottom=dur_tabl
ax[1].set_ylabel('Percentage')
ax[1].set_title('Term Deposit Subscription & Un-subscription Percentage by Duration
ax[1].bar_label(bar1, label_type='center', fmt='%.2f%%', rotation=0)
ax[1].bar_label(bar2, label_type='center', fmt='%.2f%%', rotation=0)
ax[1].tick_params(axis='x', rotation=45)
ax[1].legend(title="Term Deposit", bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
plt.show()
```



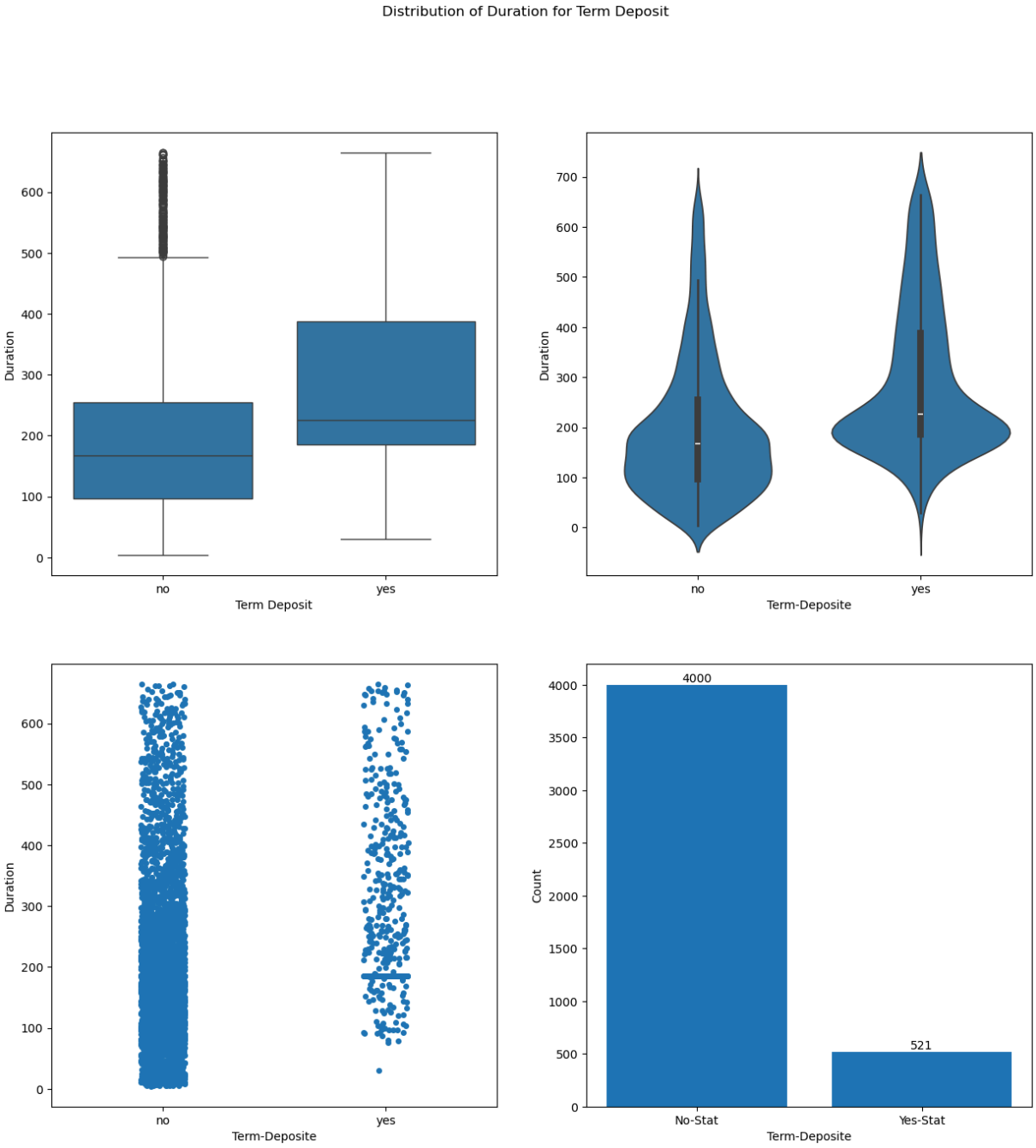


Conclusion

- Short durations (0–150s) have very low subscription rates (mostly below 5–10%).
- As call duration increases, the subscription percentage increases.
- Duration groups above ~400 seconds show notably higher subscription rates, peaking at 32.39% in the highest duration group (614–865s).
- Longer conversations strongly correlate with higher likelihood of term deposit subscription.
- Shows fewer overall calls in higher duration bins, their conversion rates are significantly better.

In [120…

```
fig,ax=plt.subplots(2,2,figsize=(15,15))
sns.boxplot(x='term_deposite', y='duration', data=bank_df, ax=ax[0][0])
fig.suptitle("Distribution of Duration for Term Deposit")
ax[0][0].set_xlabel("Term Deposit")
ax[0][0].set_ylabel("Duration")

sns.violinplot(x='term_deposite',y='duration',data=bank_df,ax=ax[0][1])
ax[0][1].set_xlabel("Term-Deposite")
ax[0][1].set_ylabel("Duration")

sns.stripplot(x='term_deposite', y='duration', data=bank_df, ax=ax[1][0])
ax[1][0].set_xlabel("Term-Deposite")
ax[1][0].set_ylabel("Duration")

bar_count=ax[1][1].bar(['No-Stat','Yes-Stat'],[df['No-Stat'][0],df['Yes-Stat'][0]])
plt.bar_label(bar_count)
plt.xlabel("Term-Deposite")
plt.ylabel("Count")

plt.show()
```



Conclusion

- Call duration is longer for customers who subscribed ('yes') to term deposits.
- Violin and box plots show higher median and spread for 'yes' group.
- Outliers are more frequent in the 'no' group.
- Strip plot confirms dense clustering at lower durations for 'no'.
- Bar chart shows far fewer total subscriptions (521 'yes' vs 4000 'no').

ii) Multivariate Analysis

In [901…

```
bank_df.head(2)
```

Out[901…

	age	job	marital	education	default	balance	housing	loan	contact	day	n
0	30.0	unemployed	married	primary	no	1787.0	no	no	cellular	19	
1	33.0	services	married	secondary	no	444.0	yes	yes	cellular	11	

In [902…

```
print(cat_col)
print(num_col)
```

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

In [909…

```
# using t-test

important_numerical = []

for i in num_col:
    group_yes = bank_df[bank_df['term_deposite'] == 'yes'][i]
    group_no = bank_df[bank_df['term_deposite'] == 'no'][i]
    stat, p = ttest_ind(group_yes, group_no, equal_var=False)
    if p < 0.05:
        important_numerical.append(i)

print(f"Important Numerical Features Are : {important_numerical}")
```

```
Important Numerical Features Are : ['balance', 'duration', 'campaign']
```

In [915…

```
# using chi-square test

important_categorical = []

for i in cat_col:
    contingency = pd.crosstab(bank_df[i], bank_df['term_deposite'])
    chi2, p, dof, ex = chi2_contingency(contingency)
    if p < 0.05:
        important_categorical.append(i)

print(f"Important Categorical Features Are : {important_categorical}")
```

```
Important Categorical Features Are : ['job', 'marital', 'education', 'housing', 'loa
n', 'contact', 'month', 'pdays', 'previous', 'poutcome', 'term_deposite']
```

In [917…

```
bank_df[important_numerical].corr()
```

Out[917…

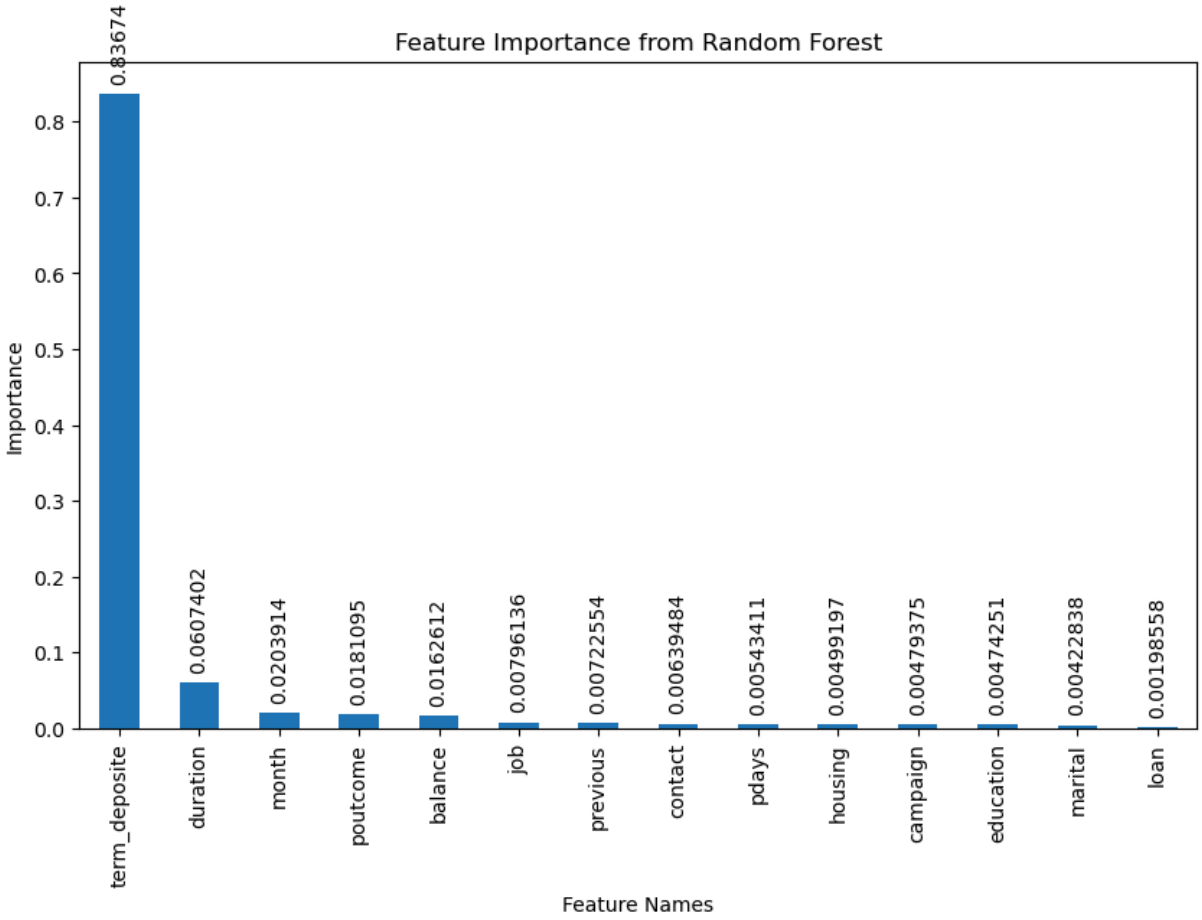
	balance	duration	campaign
balance	1.000000	0.029862	-0.017279
duration	0.029862	1.000000	-0.043310
campaign	-0.017279	-0.043310	1.000000

conclusion from above correlation

- Each and every important numerical variables are independent from each other
- There is no relation between numerical columns
- So, No feature contain similar type of information

In [120…

```
# Encode categorical columns for modeling
df_model = bank_df.copy()
le = LabelEncoder()
for col in important_categorical:
    df_model[col] = le.fit_transform(df_model[col])

# Combine selected features
X = df_model[selected_features]
y = df_model['term_deposite']

model = RandomForestClassifier()
model.fit(X, y)

# Feature importance
importances = pd.Series(model.feature_importances_, index=X.columns)
b=importances.sort_values(ascending=False).plot(kind='bar', figsize=(10,6))
for i in b.containers:
    b.bar_label(i,rotation=90,padding=5)
plt.xlabel("Feature Names")
plt.ylabel("Importance")
plt.title('Feature Importance from Random Forest')
plt.show()
```



Conclusion

- The above graph shows the importance of the column for decision making from higher importance to lower importance.
- The column that contain more value is more important and the column that contain less value is less important

In [950…
```
print(f"Important Numerical Column List : {important_numerical}")
print(f"Important Categorical Column List : {important_categorical}")
```

```
Important Numerical Column List : ['balance', 'duration', 'campaign']
Important Categorical Column List : ['job', 'marital', 'education', 'housing', 'loa
n', 'contact', 'month', 'pdays', 'previous', 'poutcome', 'term_deposite']
```

In [120…
```
# Analyse Job, Marital and Term-Deposite Column

job=bank_df['job']
marital=bank_df['marital']
term_dep=bank_df['term_deposite']

yes,no,all_count,yes_per,no_per,l1,l2=[],[],[],[],[],[],[]
labels=job_col.unique()
labels2=marital.unique()

for i in labels:
    con1=job==i
```

```
    for j in labels2:
        con2=marital==j
        con3=term_deposite=='yes'
        con4=term_deposite=='no'

        yes_con=con1 & con2 & con3
        no_con=con1 & con2 & con4
        total=len(bank_df[(con1 & con2)])

        yes_len=len(bank_df[yes_con])
        no_len=len(bank_df[no_con])
        y_per=(round((yes_len/total)*100,2) if total != 0 else 0)
        n_per=(round((no_len/total)*100,2) if total != 0 else 0)

        l1.append(i)
        l2.append(j)
        all_count.append(total)
        yes.append(yes_len)
        no.append(no_len)
        yes_per.append(y_per)
        no_per.append(n_per)

df=pd.DataFrame(zip(yes,no,all_count,yes_per,no_per),index=zip(l1,l2),columns=['td_
df
```

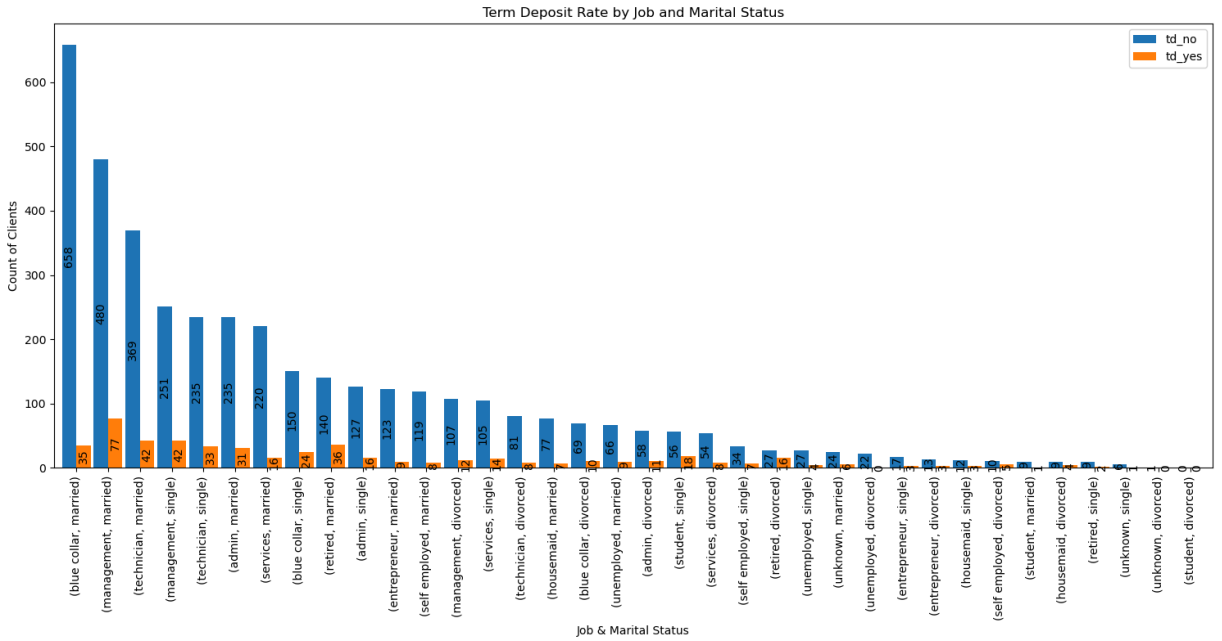
Out[120…

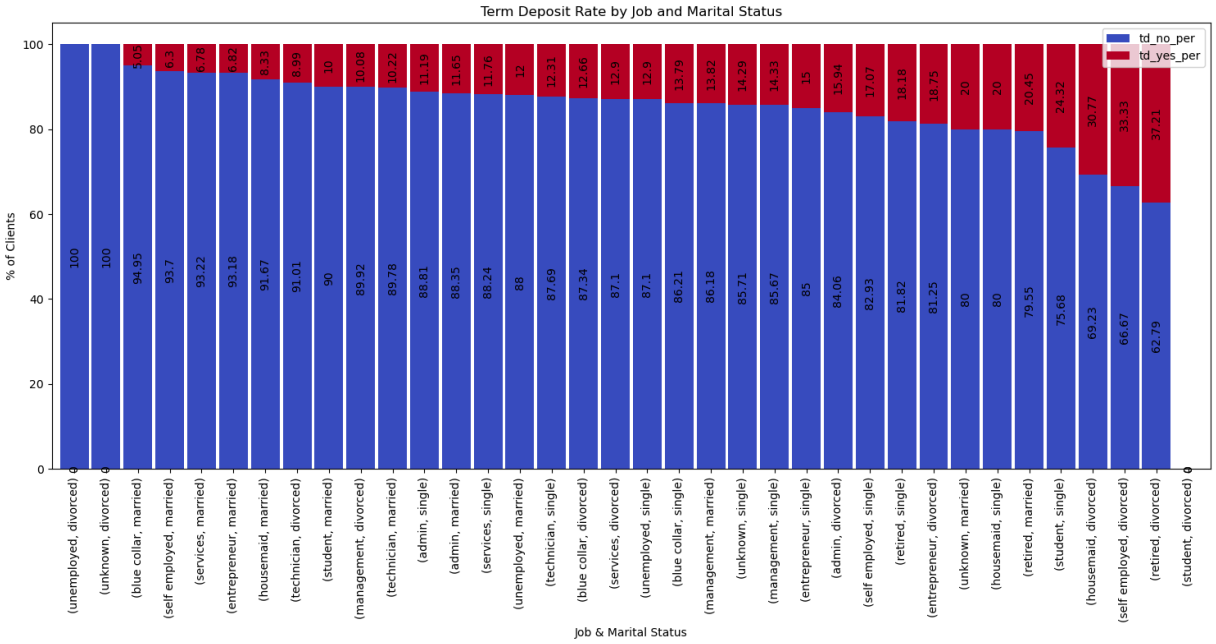
	td_yes	td_no	all	td_yes_per	td_no_per
(blue collar, married)	35	658	693	5.05	94.95
(management, married)	77	480	557	13.82	86.18
(technician, married)	42	369	411	10.22	89.78
(management, single)	42	251	293	14.33	85.67
(technician, single)	33	235	268	12.31	87.69
(admin, married)	31	235	266	11.65	88.35
(services, married)	16	220	236	6.78	93.22
(blue collar, single)	24	150	174	13.79	86.21
(retired, married)	36	140	176	20.45	79.55
(admin, single)	16	127	143	11.19	88.81
(entrepreneur, married)	9	123	132	6.82	93.18
(self employed, married)	8	119	127	6.30	93.70
(management, divorced)	12	107	119	10.08	89.92
(services, single)	14	105	119	11.76	88.24
(technician, divorced)	8	81	89	8.99	91.01
(housemaid, married)	7	77	84	8.33	91.67
(blue collar, divorced)	10	69	79	12.66	87.34
(unemployed, married)	9	66	75	12.00	88.00
(admin, divorced)	11	58	69	15.94	84.06
(student, single)	18	56	74	24.32	75.68
(services, divorced)	8	54	62	12.90	87.10
(self employed, single)	7	34	41	17.07	82.93
(retired, divorced)	16	27	43	37.21	62.79
(unemployed, single)	4	27	31	12.90	87.10
(unknown, married)	6	24	30	20.00	80.00
(unemployed, divorced)	0	22	22	0.00	100.00
(entrepreneur, single)	3	17	20	15.00	85.00
(entrepreneur, divorced)	3	13	16	18.75	81.25
(housemaid, single)	3	12	15	20.00	80.00
(self employed, divorced)	5	10	15	33.33	66.67

	td_yes	td_no	all	td_yes_per	td_no_per
(student, married)	1	9	10	10.00	90.00
(housemaid, divorced)	4	9	13	30.77	69.23
(retired, single)	2	9	11	18.18	81.82
(unknown, single)	1	6	7	14.29	85.71
(unknown, divorced)	0	1	1	0.00	100.00
(student, divorced)	0	0	0	0.00	0.00

In [120…

```
bar=df[['td_no','td_yes']].plot(kind='bar', figsize=(15,8),width=0.9)
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Job and Marital Status')
plt.ylabel('Count of Clients')
plt.xlabel('Job & Marital Status')
plt.tight_layout()

plt.show()

bar=df[['td_no_per','td_yes_per']].sort_values(by='td_no_per',ascending=False).plot
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Job and Marital Status')
plt.ylabel('% of Clients')
plt.xlabel('Job & Marital Status')
plt.tight_layout()

plt.show()
```





Conclusion

- (blue collar, married) and (management, married) dominate in count but have low to moderate term deposit rates.
- (retired, divorced), (housemaid, divorced), and (student, single) show the highest conversion rates, despite smaller group sizes.
- Divorced and single clients in certain jobs (e.g., students, retirees) are more likely to subscribe.
- (unemployed, divorced) and (unknown, divorced) have 0% subscription rates.
- Focus marketing on high-converting, smaller segments rather than just large groups. Prioritize quality over quantity.

In [120…

```
# Analyse Job, Balance and Term-Deposite Column

print(f"These are the balance bins : {bal_bins}\n\n")

b_group=pd.cut(bank_df['balance'],bins=bal_bins,right=False)
keys=[f"{i.left}-{i.right}" for i in b_group]

b_df=pd.DataFrame(zip(bank_df['job'],keys,bank_df['term_deposite']),columns=["Jobs"

b_table = b_df.groupby(['Jobs','Balance-Group'])['Term-Deposite'].value_counts().un
b_table['Total'] = b_table.sum(axis=1)
b_table['td_yes_per'] = round(b_table['yes'] / b_table['Total'] * 100, 2)
b_table['td_no_per'] = round(b_table['no'] / b_table['Total'] * 100, 2)
b_table.reset_index(inplace=True)
b_table=b_table.sort_values(by=["Total"],ascending=False)
b_table
table=pd.DataFrame(zip(b_table['no'],b_table['yes'],b_table['Total'],b_table['td_ye
pd.options.display.max_rows=133
table.fillna(0,inplace=True)
table
```

```
These are the balance bins : [-1746.0, -1335.77, -925.54, -515.31, -105.079999999999
93, 305.1500000000001, 715.3800000000001, 1125.6100000000001, 1535.8400000000001, 19
46.0700000000002, 2356.3, 2766.5300000000007, 3176.76, 3586.99, 3997.2200000000003]
```

Out[120…

	no	yes	Total	td_yes_per	td_no_per
(blue collar, -105.08-305.15)	321.0	17.0	338.0	5.03	94.97
(management, 305.15-715.38)	275.0	37.0	312.0	11.86	88.14
(management, -105.08-305.15)	279.0	30.0	309.0	9.71	90.29
(technician, -105.08-305.15)	264.0	26.0	290.0	8.97	91.03
(blue collar, 305.15-715.38)	241.0	17.0	258.0	6.59	93.41
(technician, 305.15-715.38)	190.0	24.0	214.0	11.21	88.79
(admin, -105.08-305.15)	168.0	21.0	189.0	11.11	88.89
(services, -105.08-305.15)	165.0	13.0	178.0	7.30	92.70
(admin, 305.15-715.38)	89.0	22.0	111.0	19.82	80.18
(services, 305.15-715.38)	97.0	11.0	108.0	10.19	89.81
(blue collar, 715.38-1125.61)	88.0	8.0	96.0	8.33	91.67
(retired, 305.15-715.38)	63.0	24.0	87.0	27.59	72.41
(management, 715.38-1125.61)	72.0	14.0	86.0	16.28	83.72
(technician, 715.38-1125.61)	74.0	5.0	79.0	6.33	93.67
(management, 1125.61-1535.84)	61.0	18.0	79.0	22.78	77.22
(entrepreneur, -105.08-305.15)	68.0	7.0	75.0	9.33	90.67
(retired, -105.08-305.15)	62.0	10.0	72.0	13.89	86.11
(self employed, -105.08-305.15)	64.0	5.0	69.0	7.25	92.75
(blue collar, 1125.61-1535.84)	62.0	3.0	65.0	4.62	95.38
(admin, 715.38-1125.61)	58.0	6.0	64.0	9.38	90.62
(self employed, 305.15-715.38)	52.0	8.0	60.0	13.33	86.67
(housemaid, -105.08-305.15)	49.0	5.0	54.0	9.26	90.74
(blue collar, -515.31--105.08)	45.0	6.0	51.0	11.76	88.24
(management, 1535.84-1946.07)	37.0	7.0	44.0	15.91	84.09
(entrepreneur, 305.15-715.38)	40.0	4.0	44.0	9.09	90.91
(technician, 1125.61-1535.84)	36.0	6.0	42.0	14.29	85.71
(unemployed, -105.08-305.15)	39.0	2.0	41.0	4.88	95.12
(blue collar, 1535.84-1946.07)	34.0	5.0	39.0	12.82	87.18
(housemaid, 305.15-715.38)	33.0	4.0	37.0	10.81	89.19
(student, -105.08-305.15)	26.0	9.0	35.0	25.71	74.29

	no	yes	Total	td_yes_per	td_no_per
(blue collar, 1946.07-2356.3)	23.0	9.0	32.0	28.12	71.88
(technician, -515.31--105.08)	26.0	5.0	31.0	16.13	83.87
(services, 715.38-1125.61)	27.0	3.0	30.0	10.00	90.00
(unemployed, 305.15-715.38)	26.0	3.0	29.0	10.34	89.66
(technician, 1535.84-1946.07)	24.0	4.0	28.0	14.29	85.71
(management, 2766.53-3176.76)	23.0	4.0	27.0	14.81	85.19
(management, -515.31--105.08)	24.0	2.0	26.0	7.69	92.31
(management, 2356.3-2766.53)	19.0	7.0	26.0	26.92	73.08
(management, 3176.76-3586.99)	19.0	6.0	25.0	24.00	76.00
(technician, 1946.07-2356.3)	22.0	3.0	25.0	12.00	88.00
(services, -515.31--105.08)	23.0	2.0	25.0	8.00	92.00
(admin, 1125.61-1535.84)	22.0	0.0	22.0	0.00	100.00
(retired, 715.38-1125.61)	13.0	9.0	22.0	40.91	59.09
(admin, -515.31--105.08)	18.0	3.0	21.0	14.29	85.71
(student, 305.15-715.38)	16.0	5.0	21.0	23.81	76.19
(technician, 2356.3-2766.53)	16.0	4.0	20.0	20.00	80.00
(services, 1125.61-1535.84)	18.0	2.0	20.0	10.00	90.00
(admin, 1535.84-1946.07)	17.0	1.0	18.0	5.56	94.44
(management, 1946.07-2356.3)	13.0	5.0	18.0	27.78	72.22
(blue collar, -925.54--515.31)	16.0	2.0	18.0	11.11	88.89
(blue collar, 2766.53-3176.76)	16.0	2.0	18.0	11.11	88.89
(services, 1535.84-1946.07)	15.0	3.0	18.0	16.67	83.33
(unemployed, 715.38-1125.61)	14.0	3.0	17.0	17.65	82.35
(self employed, 715.38-1125.61)	15.0	2.0	17.0	11.76	88.24
(blue collar, 2356.3-2766.53)	17.0	0.0	17.0	0.00	100.00
(admin, 2356.3-2766.53)	13.0	2.0	15.0	13.33	86.67
(admin, 1946.07-2356.3)	12.0	3.0	15.0	20.00	80.00
(entrepreneur, 715.38-1125.61)	13.0	0.0	13.0	0.00	100.00
(unknown, -105.08-305.15)	12.0	1.0	13.0	7.69	92.31
(retired, 1125.61-1535.84)	11.0	2.0	13.0	15.38	84.62

	no	yes	Total	td_yes_per	td_no_per
(self employed, 1125.61-1535.84)	10.0	2.0	12.0	16.67	83.33
(services, 1946.07-2356.3)	11.0	1.0	12.0	8.33	91.67
(unknown, 305.15-715.38)	9.0	3.0	12.0	25.00	75.00
(technician, 2766.53-3176.76)	11.0	1.0	12.0	8.33	91.67
(technician, 3176.76-3586.99)	9.0	3.0	12.0	25.00	75.00
(blue collar, 3176.76-3586.99)	11.0	0.0	11.0	0.00	100.00
(management, -925.54--515.31)	11.0	0.0	11.0	0.00	100.00
(admin, 2766.53-3176.76)	10.0	0.0	10.0	0.00	100.00
(retired, 1535.84-1946.07)	7.0	3.0	10.0	30.00	70.00
(unemployed, 1535.84-1946.07)	9.0	0.0	9.0	0.00	100.00
(entrepreneur, 1125.61-1535.84)	9.0	0.0	9.0	0.00	100.00
(unemployed, -515.31--105.08)	9.0	0.0	9.0	0.00	100.00
(admin, 3176.76-3586.99)	8.0	0.0	8.0	0.00	100.00
(unemployed, 3176.76-3586.99)	6.0	2.0	8.0	25.00	75.00
(services, 3176.76-3586.99)	7.0	1.0	8.0	12.50	87.50
(self employed, 1946.07-2356.3)	6.0	2.0	8.0	25.00	75.00
(technician, -925.54--515.31)	5.0	2.0	7.0	28.57	71.43
(services, -925.54--515.31)	7.0	0.0	7.0	0.00	100.00
(technician, -1335.77--925.54)	7.0	0.0	7.0	0.00	100.00
(entrepreneur, 1535.84-1946.07)	7.0	0.0	7.0	0.00	100.00
(student, 1125.61-1535.84)	7.0	0.0	7.0	0.00	100.00
(unemployed, 1125.61-1535.84)	5.0	1.0	6.0	16.67	83.33
(services, 2356.3-2766.53)	5.0	1.0	6.0	16.67	83.33
(student, 715.38-1125.61)	5.0	1.0	6.0	16.67	83.33
(retired, 1946.07-2356.3)	4.0	2.0	6.0	33.33	66.67
(housemaid, 715.38-1125.61)	4.0	2.0	6.0	33.33	66.67
(self employed, 1535.84-1946.07)	6.0	0.0	6.0	0.00	100.00
(entrepreneur, 2356.3-2766.53)	5.0	1.0	6.0	16.67	83.33
(retired, 2766.53-3176.76)	3.0	2.0	5.0	40.00	60.00
(retired, -515.31--105.08)	5.0	0.0	5.0	0.00	100.00

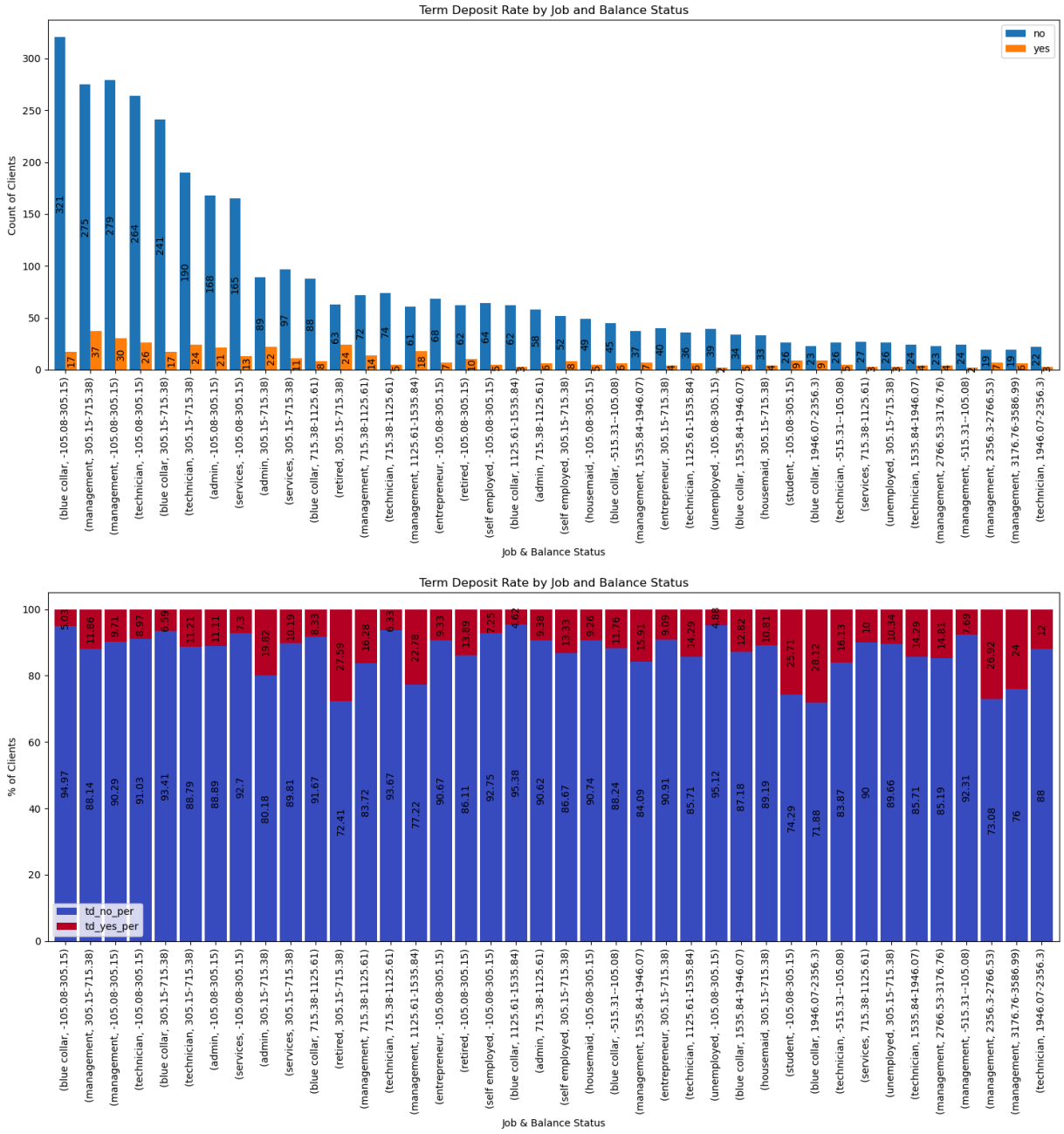
	no	yes	Total	td_yes_per	td_no_per
(student, 1535.84-1946.07)	4.0	1.0	5.0	20.00	80.00
(retired, 2356.3-2766.53)	5.0	0.0	5.0	0.00	100.00
(entrepreneur, 1946.07-2356.3)	5.0	0.0	5.0	0.00	100.00
(management, -1335.77--925.54)	4.0	1.0	5.0	20.00	80.00
(admin, -925.54--515.31)	4.0	0.0	4.0	0.00	100.00
(self employed, -515.31--105.08)	4.0	0.0	4.0	0.00	100.00
(services, 2766.53-3176.76)	3.0	1.0	4.0	25.00	75.00
(self employed, 2356.3-2766.53)	3.0	1.0	4.0	25.00	75.00
(entrepreneur, -515.31--105.08)	3.0	0.0	3.0	0.00	100.00
(self employed, 2766.53-3176.76)	3.0	0.0	3.0	0.00	100.00
(unemployed, -925.54--515.31)	3.0	0.0	3.0	0.00	100.00
(unknown, 1535.84-1946.07)	3.0	0.0	3.0	0.00	100.00
(housemaid, 1125.61-1535.84)	1.0	2.0	3.0	66.67	33.33
(housemaid, 1535.84-1946.07)	3.0	0.0	3.0	0.00	100.00
(unknown, 715.38-1125.61)	1.0	2.0	3.0	66.67	33.33
(unknown, 2356.3-2766.53)	3.0	0.0	3.0	0.00	100.00
(student, -515.31--105.08)	3.0	0.0	3.0	0.00	100.00
(student, 2356.3-2766.53)	1.0	2.0	3.0	66.67	33.33
(retired, 3176.76-3586.99)	2.0	1.0	3.0	33.33	66.67
(entrepreneur, 2766.53-3176.76)	0.0	2.0	2.0	100.00	0.00
(housemaid, 3176.76-3586.99)	2.0	0.0	2.0	0.00	100.00
(unemployed, 2356.3-2766.53)	2.0	0.0	2.0	0.00	100.00
(unemployed, 1946.07-2356.3)	1.0	1.0	2.0	50.00	50.00
(entrepreneur, -925.54--515.31)	1.0	1.0	2.0	50.00	50.00
(student, 2766.53-3176.76)	2.0	0.0	2.0	0.00	100.00
(entrepreneur, 3176.76-3586.99)	2.0	0.0	2.0	0.00	100.00
(unknown, 1125.61-1535.84)	2.0	0.0	2.0	0.00	100.00
(student, 3176.76-3586.99)	1.0	1.0	2.0	50.00	50.00
(retired, -1335.77--925.54)	1.0	1.0	2.0	50.00	50.00
(unknown, 1946.07-2356.3)	1.0	1.0	2.0	50.00	50.00

	no	yes	Total	td_yes_per	td_no_per
(housemaid, 2356.3-2766.53)	2.0	0.0	2.0	0.00	100.00
(housemaid, 2766.53-3176.76)	1.0	1.0	2.0	50.00	50.00
(unemployed, 2766.53-3176.76)	1.0	1.0	2.0	50.00	50.00
(blue collar, -1746.0--1335.77)	1.0	0.0	1.0	0.00	100.00
(blue collar, -1335.77--925.54)	1.0	0.0	1.0	0.00	100.00
(services, -1335.77--925.54)	1.0	0.0	1.0	0.00	100.00
(blue collar, 3586.99-3997.22)	1.0	0.0	1.0	0.00	100.00
(housemaid, -515.31--105.08)	1.0	0.0	1.0	0.00	100.00
(housemaid, -925.54--515.31)	1.0	0.0	1.0	0.00	100.00
(technician, -1746.0--1335.77)	1.0	0.0	1.0	0.00	100.00
(management, -1746.0--1335.77)	1.0	0.0	1.0	0.00	100.00
(admin, -1335.77--925.54)	1.0	0.0	1.0	0.00	100.00
(housemaid, 1946.07-2356.3)	1.0	0.0	1.0	0.00	100.00

In [116…
```
print("Total Data : ",len(bank_df))
top_40=table['Total'].head(40).values.sum()
print("Top 40 highest categories belonging data : ",top_40)
print(f"Top 40 rows contain data percentage for analysing : {(top_40/len(bank_df))*

# Top 40 rows of the table dataset contains 84.17% of the data. I think this data i
```

```
Total Data :  4521
Top 40 highest categories belonging data :  3805.0
Top 40 rows contain data percentage for analysing : 84.16279584162795
```

In [121…
```
bar=table[['no','yes']].head(40).plot(kind='bar', figsize=(15,8), width=0.9)
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Job and Balance Status')
plt.ylabel('Count of Clients')
plt.xlabel('Job & Balance Status')
plt.tight_layout()

plt.show()

bar=table[['td_no_per','td_yes_per']].head(40).plot(kind='bar', stacked=True, figsi
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Job and Balance Status')
plt.ylabel('% of Clients')
plt.xlabel('Job & Balance Status')
plt.tight_layout()

plt.show()
```



Conclusion

- Most clients are blue-collar, management, and technician in lower balance ranges (below 1050 or 1750). However, most of them did not subscribe to term deposits.
- The highest subscription rates (red bars) are observed in retired, student, and unemployed clients, especially those with moderate to high balances (e.g., 1535–1946, 1946+).
- Client type and balance range strongly influence deposit behavior. High volume does not equal high conversion. some smaller groups (like retired with moderate balances) show better targeting potential for deposits.

In [116…

```
print("Numerical Columns : ",important_numerical)
print("Categorical Columns : ",important_categorical)
```

```
Numerical Columns :  ['balance', 'duration', 'campaign']
Categorical Columns :  ['job', 'marital', 'education', 'housing', 'loan', 'contact',
'month', 'pdays', 'previous', 'poutcome', 'term_deposite']
```

In [117…

```
# Analyse Marital, Loan and Term-Deposite Column

print(bank_df['loan'].unique())
print(bank_df['marital'].unique())
```

```
['no' 'yes']
['married' 'single' 'divorced']
```

In [121…

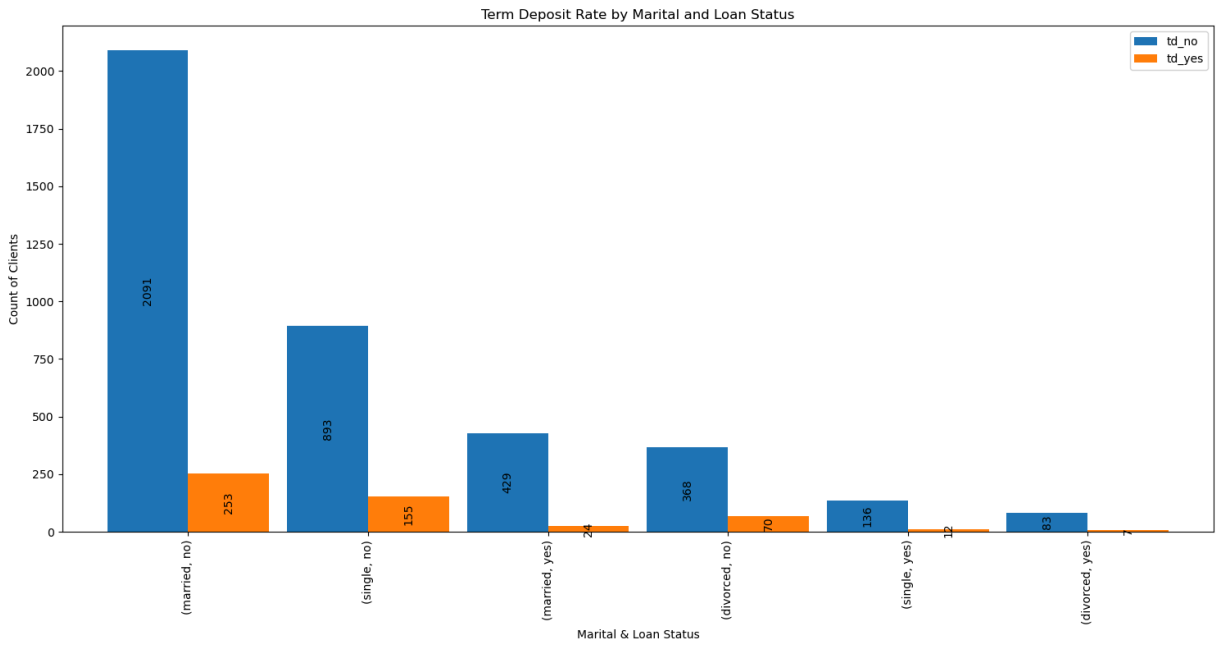
```
loan=bank_df['loan']
marital=bank_df['marital']
term_dep=bank_df['term_deposite']

yes,no,all_count,yes_per,no_per,l1,l2=[],[],[],[],[],[],[]
labels=marital.unique()
labels2=loan_col.unique()

for i in labels:
    con1=marital==i
    for j in labels2:
        con2=loan==j
        con3=term_deposite=='yes'
        con4=term_deposite=='no'

        yes_con=con1 & con2 & con3
        no_con=con1 & con2 & con4
        total=len(bank_df[(con1 & con2)])

        yes_len=len(bank_df[yes_con])
        no_len=len(bank_df[no_con])
        y_per=(round((yes_len/total)*100,2) if total != 0 else 0)
        n_per=(round((no_len/total)*100,2) if total != 0 else 0)

        l1.append(i)
        l2.append(j)
        all_count.append(total)
        yes.append(yes_len)
        no.append(no_len)
        yes_per.append(y_per)
        no_per.append(n_per)

df=pd.DataFrame(zip(yes,no,all_count,yes_per,no_per),index=zip(l1,l2),columns=['td_
df
```

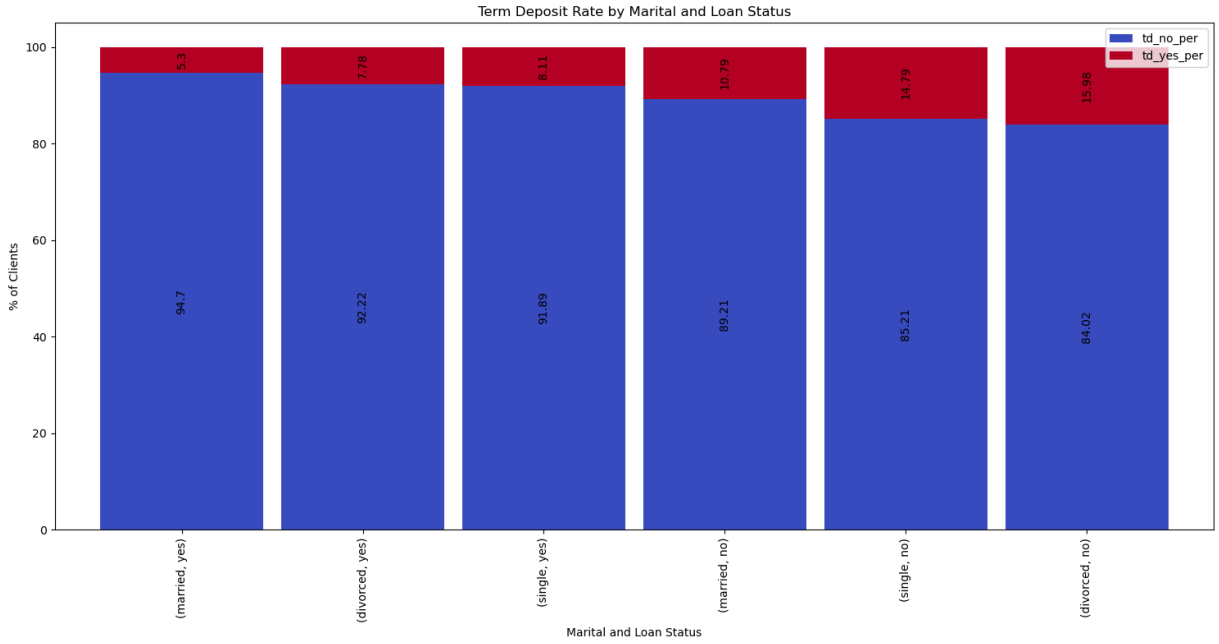
Out[121…

	td_yes	td_no	all	td_yes_per	td_no_per
(married, no)	253	2091	2344	10.79	89.21
(single, no)	155	893	1048	14.79	85.21
(married, yes)	24	429	453	5.30	94.70
(divorced, no)	70	368	438	15.98	84.02
(single, yes)	12	136	148	8.11	91.89
(divorced, yes)	7	83	90	7.78	92.22

In [121…

```
bar=df[['td_no','td_yes']].plot(kind='bar', figsize=(15,8),width=0.9)
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Marital and Loan Status')
plt.ylabel('Count of Clients')
plt.xlabel('Marital & Loan Status')
plt.tight_layout()

plt.show()

bar=df[['td_no_per','td_yes_per']].sort_values(by='td_no_per',ascending=False).plot
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Marital and Loan Status')
plt.ylabel('% of Clients')
plt.xlabel('Marital and Loan Status')
plt.tight_layout()

plt.show()
```





Conclusion

- Clients without loans are more likely to subscribe to term deposits.
- Divorced and single clients without loans have higher subscription percentages than married clients.
- no loans might contribute to willingness to invest in term deposits.

In [117…

```
# Analyse Job, Duration and Term-Deposite Column

print(f"No of unique values in job column : {bank_df['job'].unique()}")
print(f"No of bins that can forms in duration column : {len(dur_bins)}")
```

```
No of unique values in job column : ['unemployed' 'services' 'management' 'blue coll
ar' 'self employed'
 'technician' 'entrepreneur' 'admin' 'student' 'housemaid' 'retired'
 'unknown']
No of bins that can forms in duration column : 14
```

In [121…

```
print(f"These are the Duration bins : {dur_bins}\n\n")

d_group=pd.cut(bank_df['duration'],bins=bal_bins,right=False)
keys=[f"{i.left}-{i.right}" for i in d_group]

d_df=pd.DataFrame(zip(bank_df['job'],keys,bank_df['term_deposite']),columns=["Jobs"

d_table = d_df.groupby(['Jobs','Duration-Group'])['Term-Deposite'].value_counts().u
d_table['Total'] = d_table.sum(axis=1)
d_table['td_yes_per'] = round(d_table['yes'] / d_table['Total'] * 100, 2)
d_table['td_no_per'] = round(d_table['no'] / d_table['Total'] * 100, 2)
d_table.reset_index(inplace=True)
d_table=d_table.sort_values(by=["Total"],ascending=False)
d_table
table=pd.DataFrame(zip(d_table['no'],d_table['yes'],d_table['Total'],d_table['td_ye
pd.options.display.max_rows=133
```

```
table.fillna(0,inplace=True)
table
```

```
These are the Duration bins : [4.0, 54.85, 105.7, 156.55, 207.4, 258.25, 309.1, 359.
95, 410.8, 461.65000000000003, 512.5, 563.35, 614.2, 665.0500000000001]
```

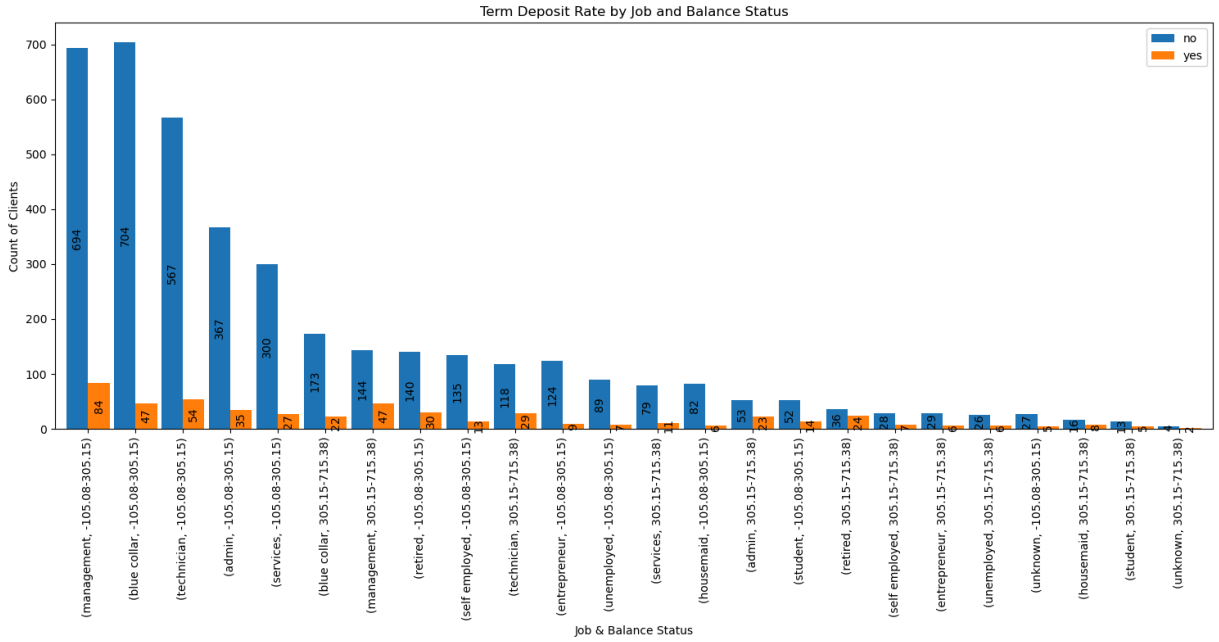
Out[121…

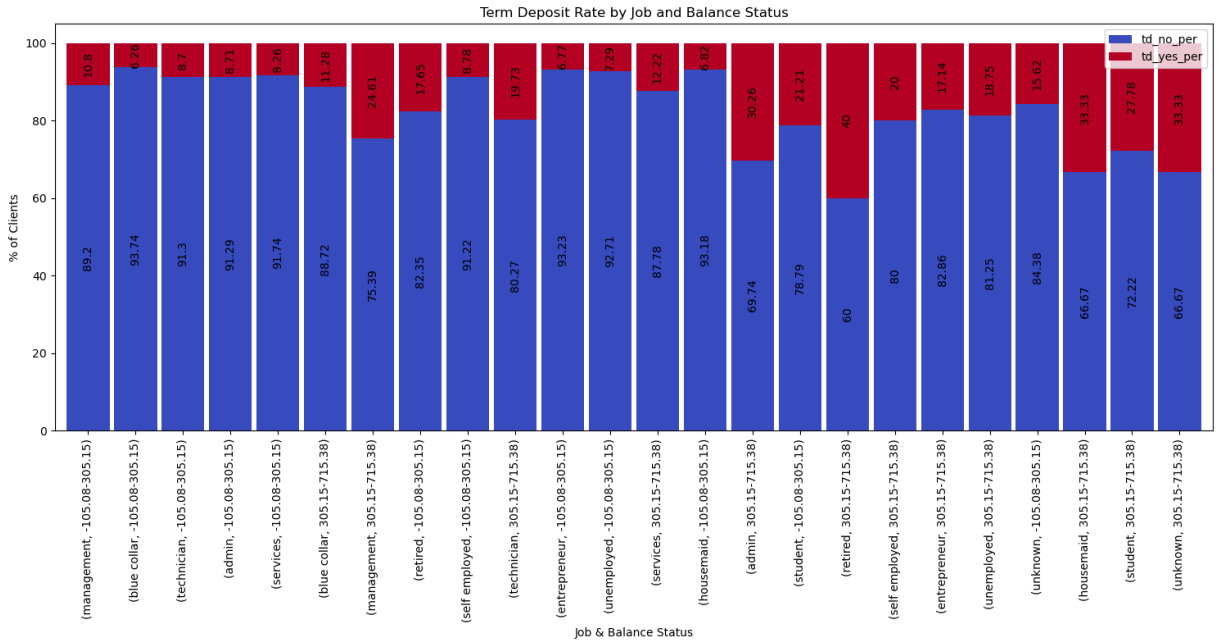
	no	yes	Total	td_yes_per	td_no_per
(management, -105.08-305.15)	694	84	778	10.80	89.20
(blue collar, -105.08-305.15)	704	47	751	6.26	93.74
(technician, -105.08-305.15)	567	54	621	8.70	91.30
(admin, -105.08-305.15)	367	35	402	8.71	91.29
(services, -105.08-305.15)	300	27	327	8.26	91.74
(blue collar, 305.15-715.38)	173	22	195	11.28	88.72
(management, 305.15-715.38)	144	47	191	24.61	75.39
(retired, -105.08-305.15)	140	30	170	17.65	82.35
(self employed, -105.08-305.15)	135	13	148	8.78	91.22
(technician, 305.15-715.38)	118	29	147	19.73	80.27
(entrepreneur, -105.08-305.15)	124	9	133	6.77	93.23
(unemployed, -105.08-305.15)	89	7	96	7.29	92.71
(services, 305.15-715.38)	79	11	90	12.22	87.78
(housemaid, -105.08-305.15)	82	6	88	6.82	93.18
(admin, 305.15-715.38)	53	23	76	30.26	69.74
(student, -105.08-305.15)	52	14	66	21.21	78.79
(retired, 305.15-715.38)	36	24	60	40.00	60.00
(self employed, 305.15-715.38)	28	7	35	20.00	80.00
(entrepreneur, 305.15-715.38)	29	6	35	17.14	82.86
(unemployed, 305.15-715.38)	26	6	32	18.75	81.25
(unknown, -105.08-305.15)	27	5	32	15.62	84.38
(housemaid, 305.15-715.38)	16	8	24	33.33	66.67
(student, 305.15-715.38)	13	5	18	27.78	72.22
(unknown, 305.15-715.38)	4	2	6	33.33	66.67

In [121…

```
bar=table[['no','yes']].head(40).plot(kind='bar', figsize=(15,8), width=0.9)
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
```

```
plt.title('Term Deposit Rate by Job and Balance Status')
plt.ylabel('Count of Clients')
plt.xlabel('Job & Balance Status')
plt.tight_layout()
plt.savefig("fig1.png")
plt.show()

bar=table[['td_no_per','td_yes_per']].head(40).plot(kind='bar', stacked=True, figsi
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Job and Balance Status')
plt.ylabel('% of Clients')
plt.xlabel('Job & Balance Status')
plt.tight_layout()
plt.savefig("fig2.png")
plt.show()
```





Conclusion

- Higher balance clients tend to subscribe more, even if they are fewer in number.
- Lower balance workers (especially blue collar and services) have very low subscription rates.
- Retired and student clients with higher balances are more likely to subscribe. They may financial secure and possibly more interest in term deposits.
- Target marketing could focus on:
  - Students and retirees with moderate to high balances.
  - Job segments with good balance and moderate conversion (e.g., management 305.15–715.38).
- Low-performing segments may need tailored communication or may not be ideal targets for term deposit offers.

In [121…

```
# Analyse Jobs, Housing and Term-Deposite Column

print(f"No of unique values in Jobs column : {bank_df['job'].unique()}")
print(f"No of unique values in Housing column : {bank_df['housing'].unique()}")
```

```
No of unique values in Jobs column : ['unemployed' 'services' 'management' 'blue col
lar' 'self employed'
 'technician' 'entrepreneur' 'admin' 'student' 'housemaid' 'retired'
 'unknown']
No of unique values in Housing column : ['no' 'yes']
```

In [122…

```
house=bank_df['housing']
job=bank_df['job']
term_dep=bank_df['term_deposite']

yes,no,all_count,yes_per,no_per,l1,l2=[],[],[],[],[],[],[]
labels=job.unique()
labels2=house.unique()

for i in labels:
    con1=job==i
    for j in labels2:
        con2=house==j
        con3=term_deposite=='yes'
        con4=term_deposite=='no'

        yes_con=con1 & con2 & con3
        no_con=con1 & con2 & con4
        total=len(bank_df[(con1 & con2)])

        yes_len=len(bank_df[yes_con])
        no_len=len(bank_df[no_con])
        y_per=(round((yes_len/total)*100,2) if total != 0 else 0)
        n_per=(round((no_len/total)*100,2) if total != 0 else 0)

        l1.append(i)
        l2.append(j)
        all_count.append(total)
```

```
        yes.append(yes_len)
        no.append(no_len)
        yes_per.append(y_per)
        no_per.append(n_per)

df=pd.DataFrame(zip(yes,no,all_count,yes_per,no_per),index=zip(l1,l2),columns=['td_
df
```

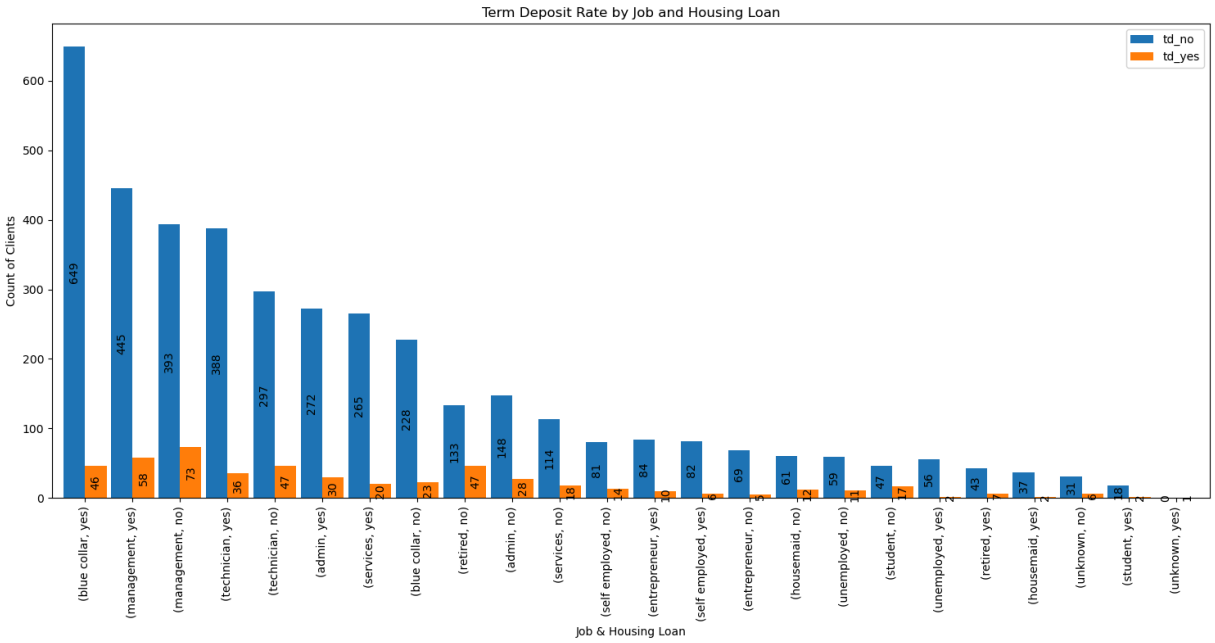
Out[122…

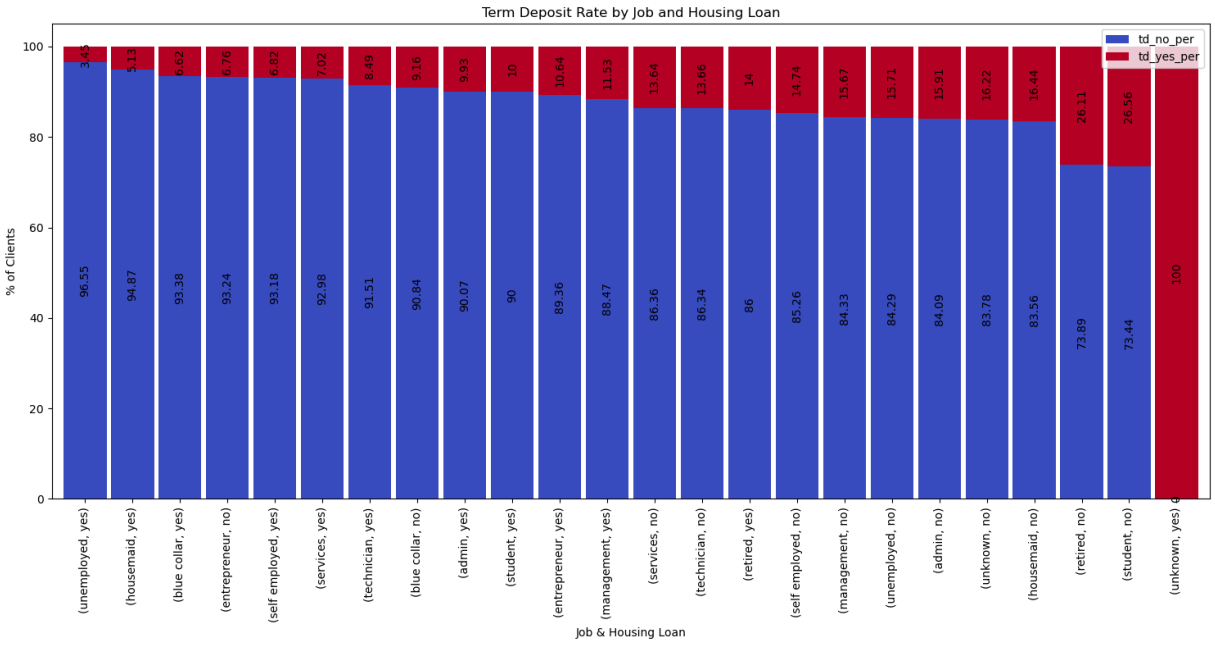
	td_yes	td_no	all	td_yes_per	td_no_per
(blue collar, yes)	46	649	695	6.62	93.38
(management, yes)	58	445	503	11.53	88.47
(management, no)	73	393	466	15.67	84.33
(technician, yes)	36	388	424	8.49	91.51
(technician, no)	47	297	344	13.66	86.34
(admin, yes)	30	272	302	9.93	90.07
(services, yes)	20	265	285	7.02	92.98
(blue collar, no)	23	228	251	9.16	90.84
(retired, no)	47	133	180	26.11	73.89
(admin, no)	28	148	176	15.91	84.09
(services, no)	18	114	132	13.64	86.36
(self employed, no)	14	81	95	14.74	85.26
(entrepreneur, yes)	10	84	94	10.64	89.36
(self employed, yes)	6	82	88	6.82	93.18
(entrepreneur, no)	5	69	74	6.76	93.24
(housemaid, no)	12	61	73	16.44	83.56
(unemployed, no)	11	59	70	15.71	84.29
(student, no)	17	47	64	26.56	73.44
(unemployed, yes)	2	56	58	3.45	96.55
(retired, yes)	7	43	50	14.00	86.00
(housemaid, yes)	2	37	39	5.13	94.87
(unknown, no)	6	31	37	16.22	83.78
(student, yes)	2	18	20	10.00	90.00
(unknown, yes)	1	0	1	100.00	0.00

In [122…

```
bar=df[['td_no','td_yes']].plot(kind='bar', figsize=(15,8),width=0.9)
for i in bar.containers:
```

```
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Job and Housing Loan')
plt.ylabel('Count of Clients')
plt.xlabel('Job & Housing Loan')
plt.tight_layout()

plt.show()

bar=df[['td_no_per','td_yes_per']].sort_values(by='td_no_per',ascending=False).plot
for i in bar.containers:
    bar.bar_label(i,label_type='center',rotation=90)
plt.title('Term Deposit Rate by Job and Housing Loan')
plt.ylabel('% of Clients')
plt.xlabel('Job & Housing Loan')
plt.tight_layout()

plt.show()
```





Conclusion

- Clients with housing loans are less likely to subscribe to term deposits.
- This trend likely reflects willingness to invest.
- Focus marketing efforts on the following high-conversion segments
  - Students (with no housing loan)
  - Retired individuals (with no housing loan)
  - Admin staff (with no housing loan)
  - Housemaids (with no housing loan)
- Avoid targeting these low-conversion groups
  - Unemployed clients with housing loans
  - Blue-collar workers with housing loans
  - Housemaids with housing loans
  - Entrepreneurs with housing loans

Overall Summary of Bank Term Deposit Analysis

Dataset Overview

- Total records: 4521
- Features: 17 (7 numerical, 10 categorical)
- Target variable: term_deposite (yes/no)
- No missing or duplicate data after cleaning.

Univariate Insights

- Most common jobs: management, blue collar, and technician (cover ~60% of customers).
- Majority marital status: married (61.87%).
- Education: Over half have secondary education.
- Default: 98.3% have no credit in default.
- Housing loan: 56.6% have a housing loan.
- Personal loan: Only 15.3% have a personal loan.
- Contact method: 64% contacted via cellular.
- Most active month: May has the highest number of contacts (~31%).

Numerical Insights

- Age: Mean = 41.17, most clients are between 29–45 years.
- Balance: Highly right-skewed (mean = 1422.66, median = 444), many customers have low or negative balances.

- Call duration: Most calls last under 5 minutes, and longer calls are associated with higher term deposit success.
- Campaign: Majority of customers contacted less than 5 times.

Target Variable (term_deposite)

- 11.5% of clients subscribed to term deposits, indicating a highly imbalanced dataset (88.5% said “no”).

Multivariate Analysis Highlights

1. $Job + MaritalStatus$:

- (retired, divorced) and (student, single) show the highest subscription rates.
- (married, blue collar) and (married, management) dominate in volume but have low conversion rates.

2. $Job + Balance$:

- Higher balance groups (e.g., retired, students) show higher subscription %.
- Low-balance workers (blue collar, services) have low success.

3. $Job + HousingLoan$:

- Clients without housing loans are more likely to subscribe.
- Best conversion segments: (retired, no loan), (student, no loan), (admin, no loan).

4. $Job + PersonalLoan$:

- Similar pattern: loan-free clients show higher interest in term deposits.

Feature Importance (Random Forest)

Most important features:

- term_deposit prediction is mostly influenced by duration, followed by poutcome, month, balance, and housing.
- Many features had negligible importance.

Key Points

- Retired, students, and single/divorced clients with no loans and moderate balances are high term_deposite conversion segments.
- Clients with loans (housing or personal), especially in blue-collar or unemployed categories are low-yield segments.
- Longer conversations correlate with higher chances of subscription.

- May, July, and August are the best months to campaign based on historical conversion volume.

In [ ]: