Bank Marketing Predictive Model

> - **Data Collection: Gather customer data.**
> - **Data Preprocessing: Clean and prepare data.**
> - **Feature Engineering: Select relevant attributes.**
> - **Model Selection: Build a classification model.**
> - **Evaluation: Assess model performance.**
> - **Business Impact: Improve marketing campaign effectiveness and customer engagement.**
> - **Benefits: Targeted marketing, cost reduction, increased subscription rates.**

Importing Libraries:

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Loading Dataset:

In [2]:
```
df=pd.read_csv(r"C:\Users\jhonn\Downloads\sinsagar program\bank-additional-full.csv",sep
df
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...
...	...	...	...	...	...	...	...	...	...	...	...
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...

41188 rows × 21 columns

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

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1

1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1

5 rows × 21 columns

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

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	car
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	

5 rows × 21 columns

In [5]:
```
df.describe()
```

Out[5]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.co
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.0
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.5
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.6
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.8
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.7
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.8
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.4
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.9

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
```

```
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

Data Preprocessing:

In [7]:
```
df.duplicated().sum()
```

Out[7]:
```
12
```

In [8]:
```
df=df.drop_duplicates()
```

In [9]:
```
df=df.dropna()
```

In [10]:
```
df.shape
```

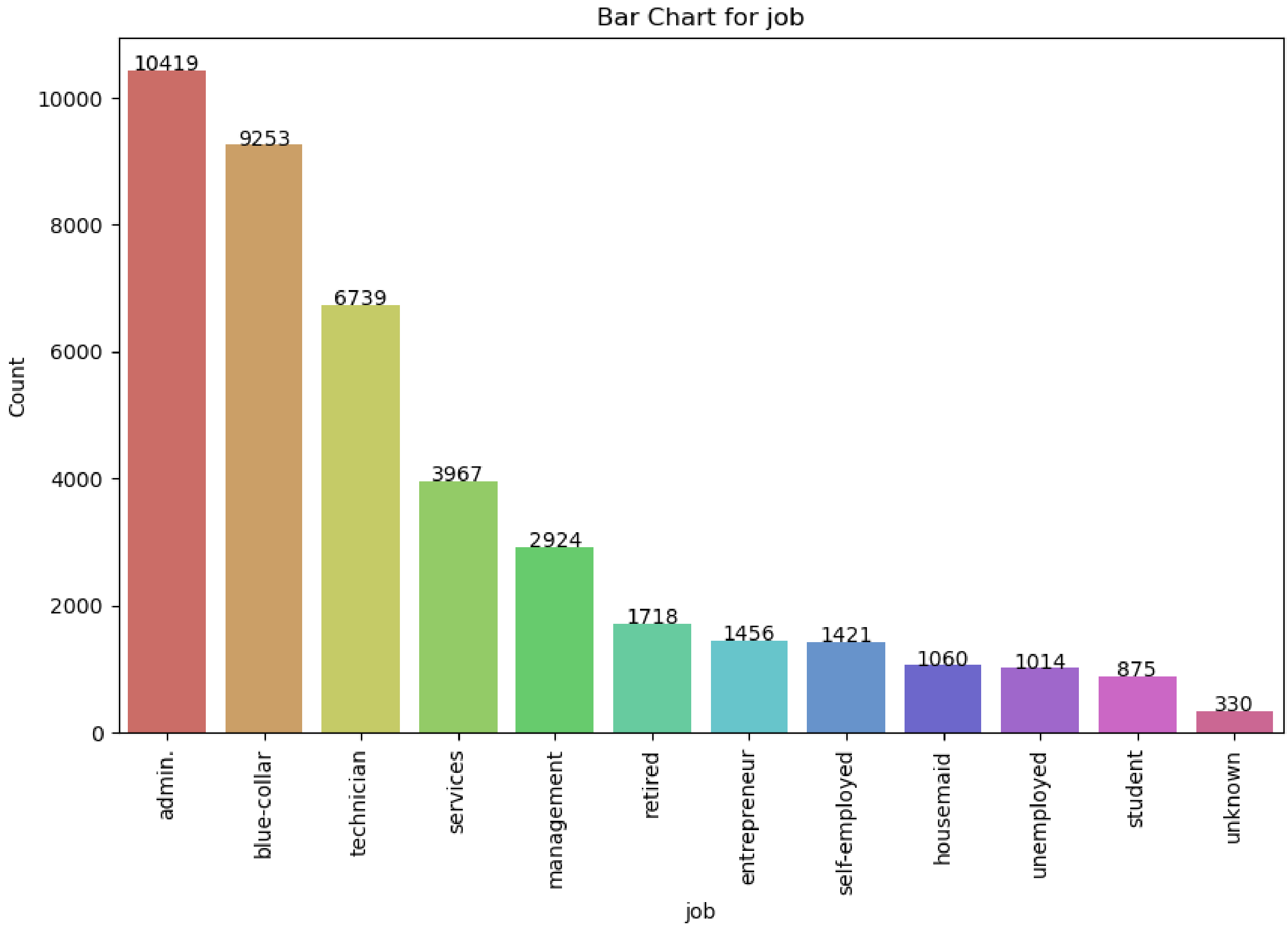
Out[10]:
```
(41176, 21)
```

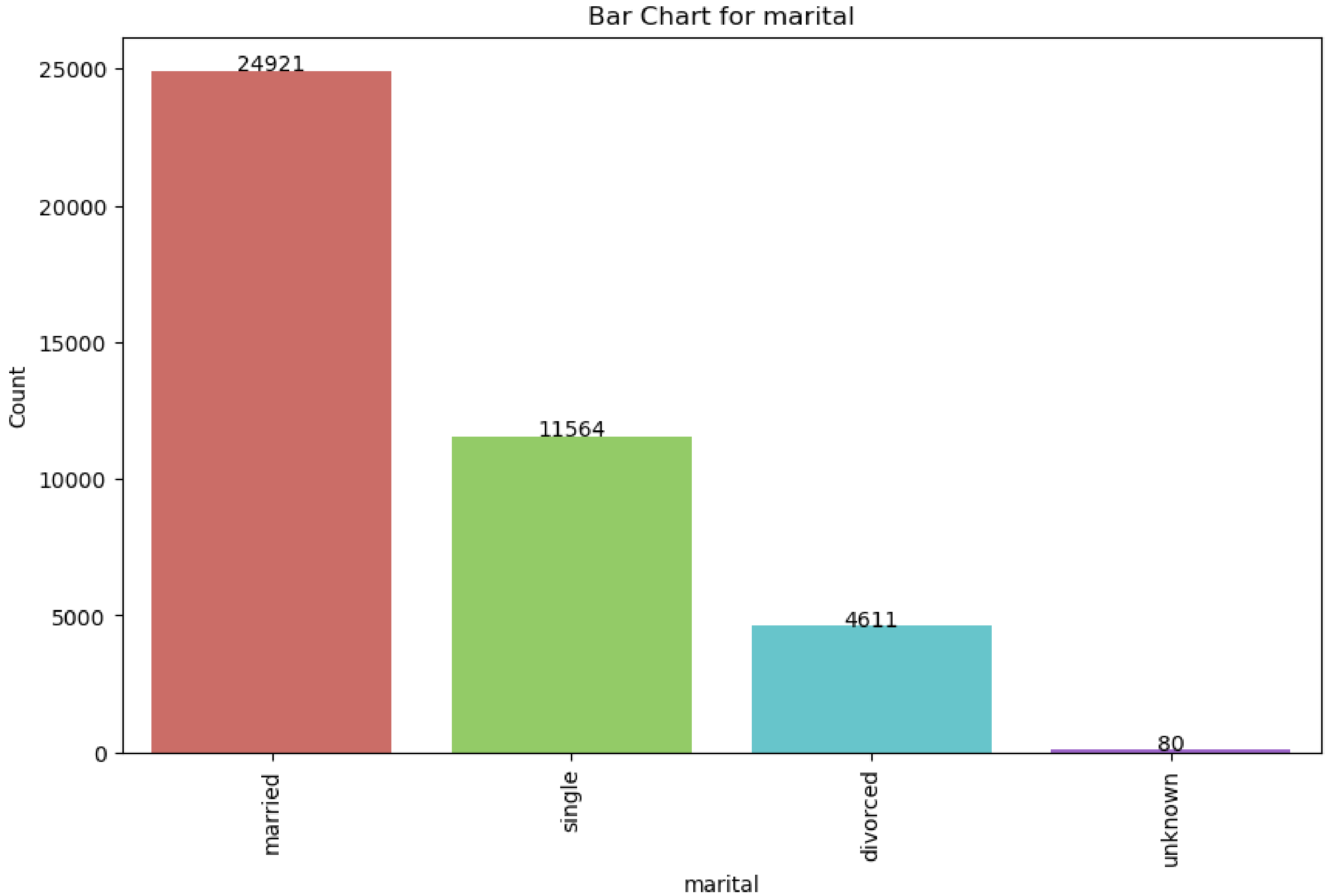
In [11]:
```
df.isna().sum()
```

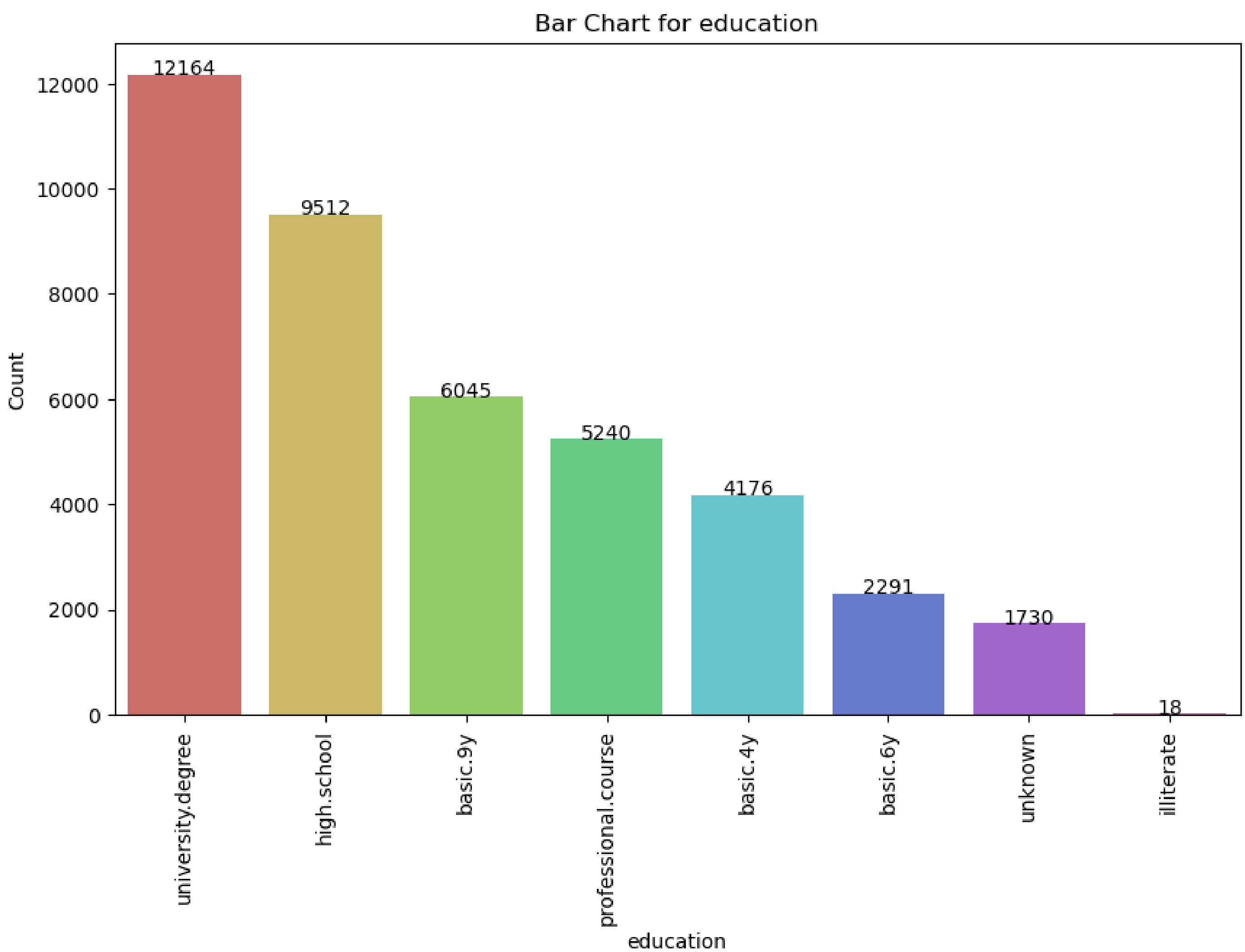
Out[11]:
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

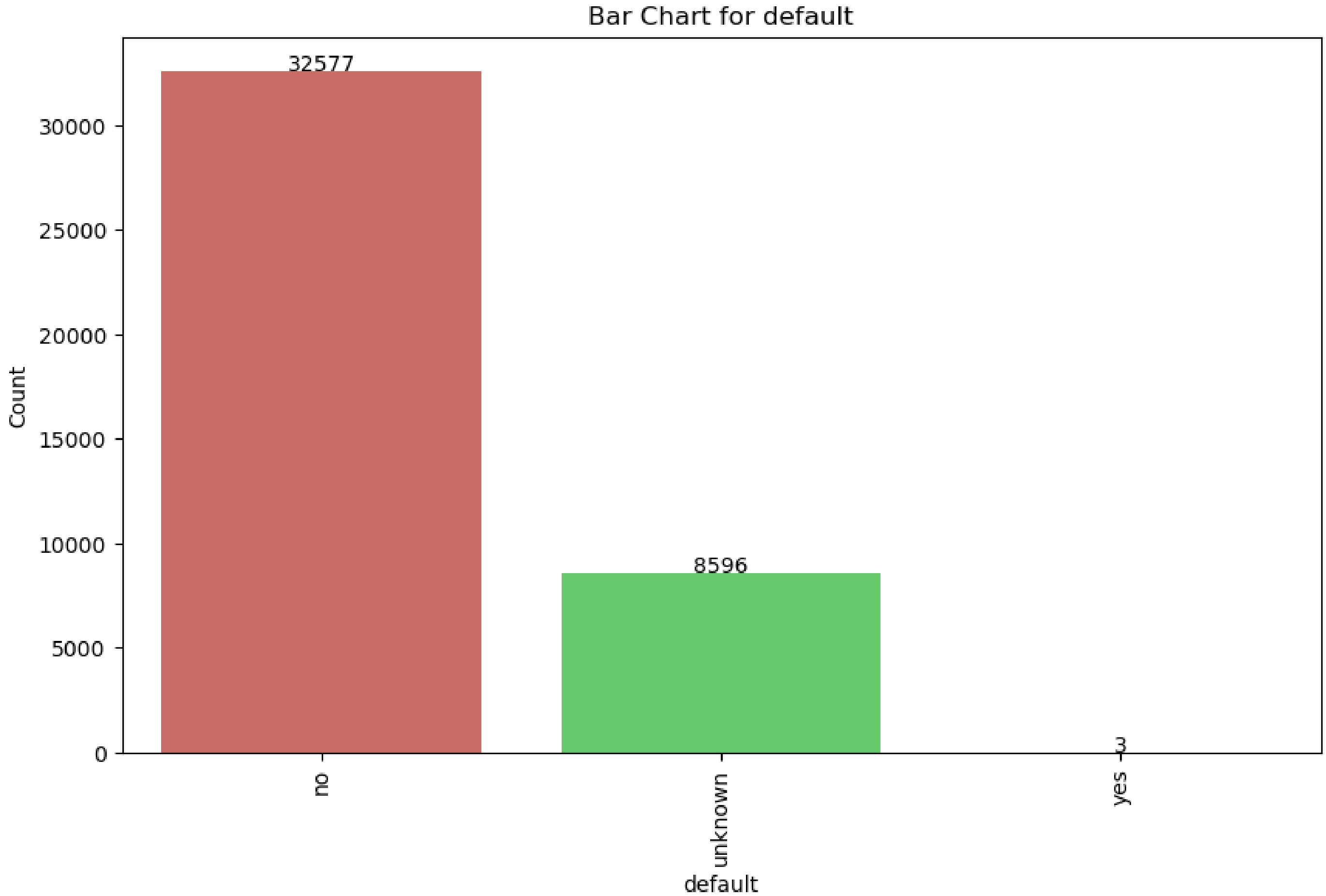
In [12]:
```
object_columns = df.select_dtypes(include=['object']).columns.tolist()
for i in object_columns:
    sorted_order = df[i].value_counts().index

    plt.figure(figsize=(10, 6))
    ax = sns.countplot(data=df, x=i, palette='hls', order=sorted_order)
    plt.xticks(rotation=90)
    plt.xlabel(i)
    plt.ylabel('Count')
    plt.title(f'Bar Chart for {i}')
    for p in ax.patches:
```

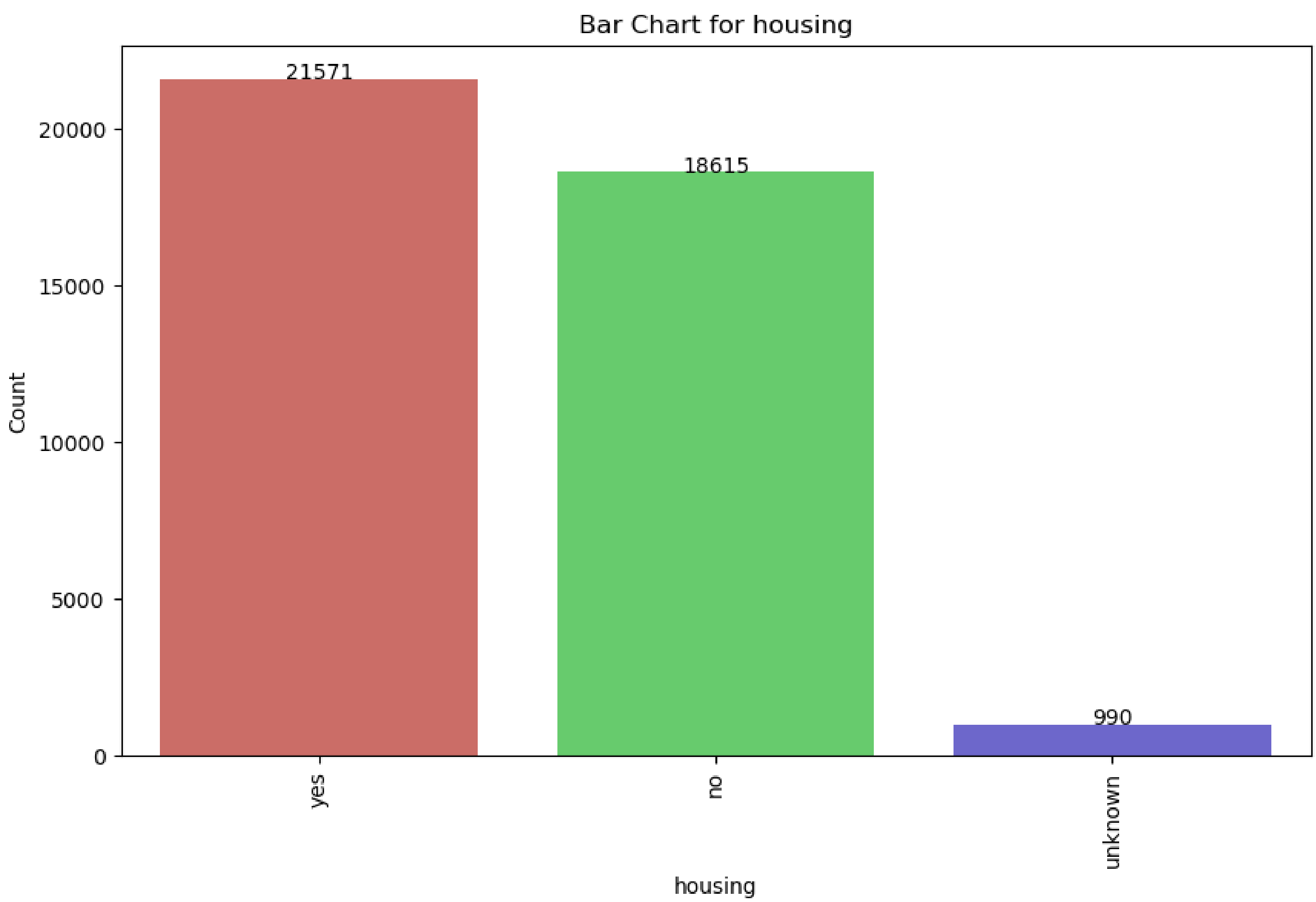
```
        ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_hei
    plt.show()
```

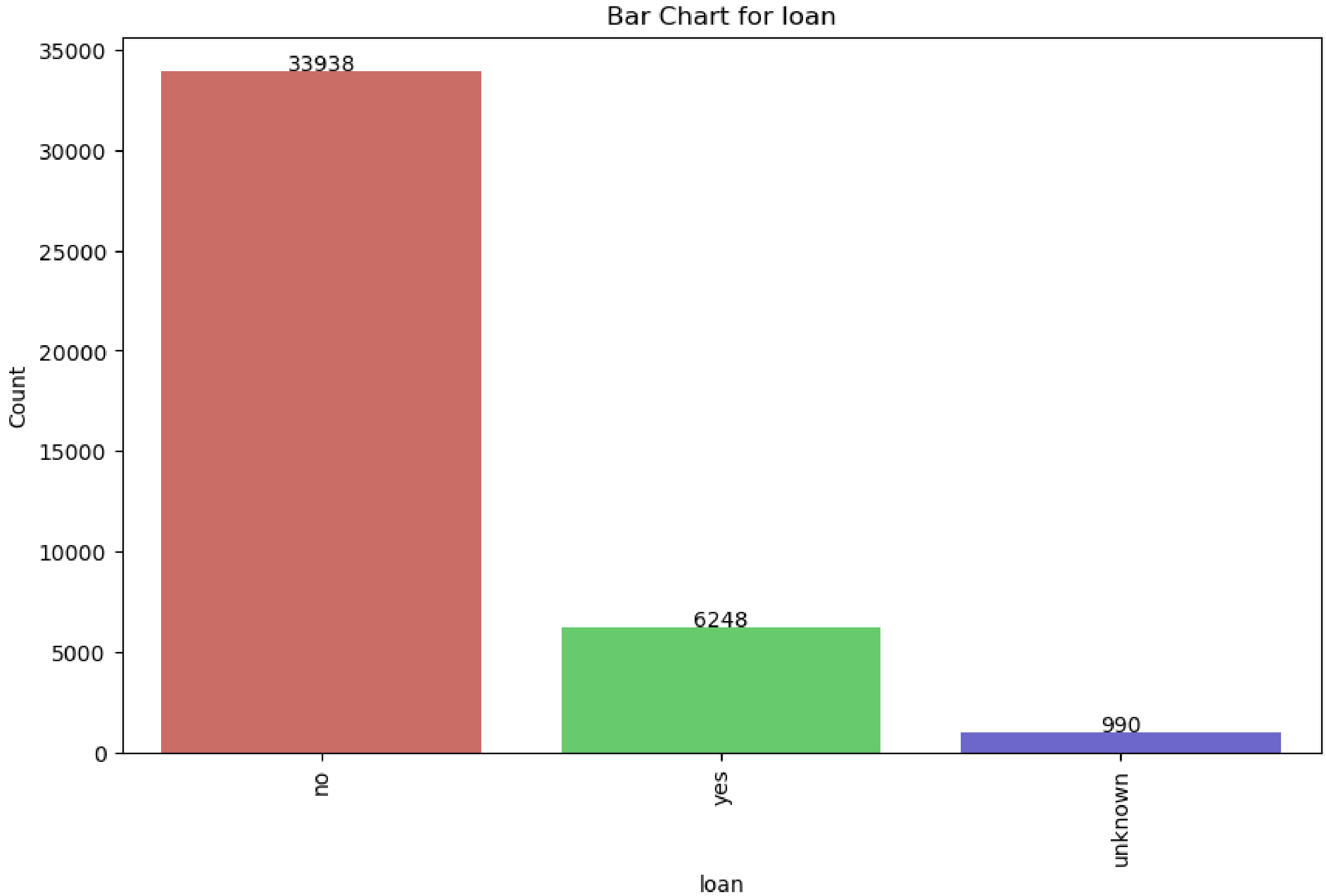


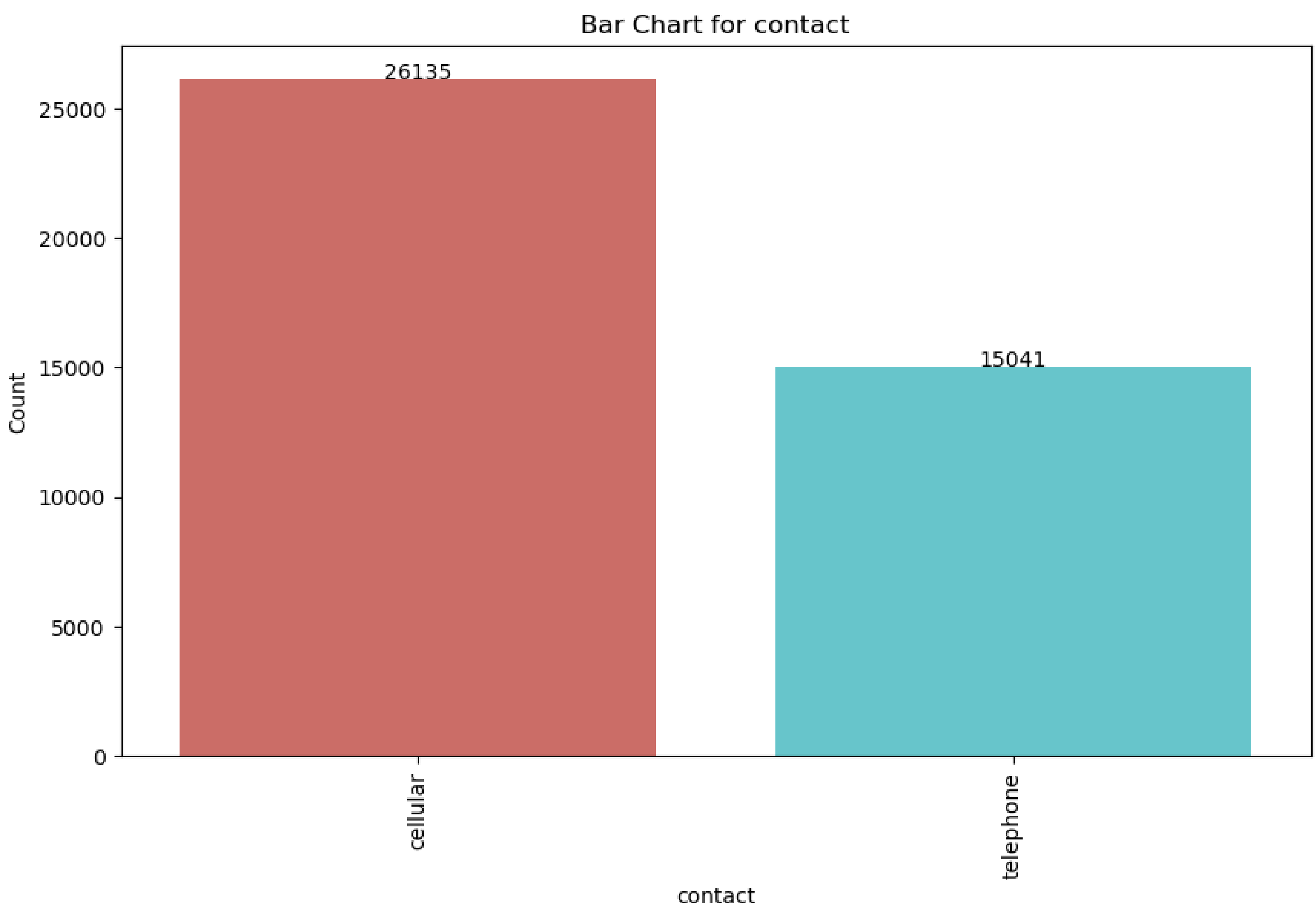

Bar Chart for marital
Count
25000
20000
15000
10000
5000
0
24921
11564
4611
80
married
single
divorced
unknown
marital

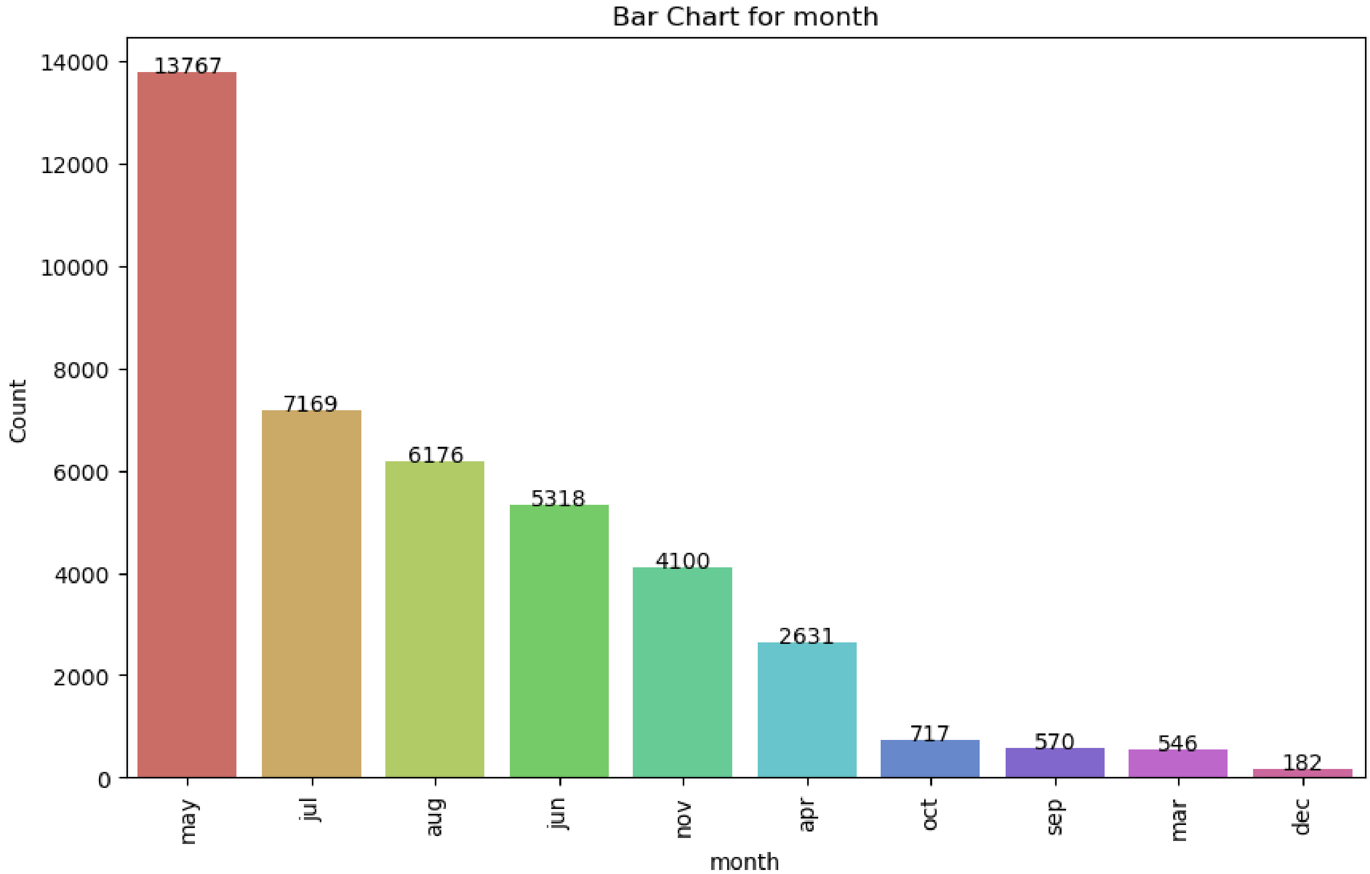

Bar Chart for education
Count
12000
10000
8000
6000
4000
2000
0
12164
9512
6045
5240
4176
2291
1730
18
university.degree
high.school
basic.9y
professional.course
basic.4y
basic.6y
unknown
illiterate
education

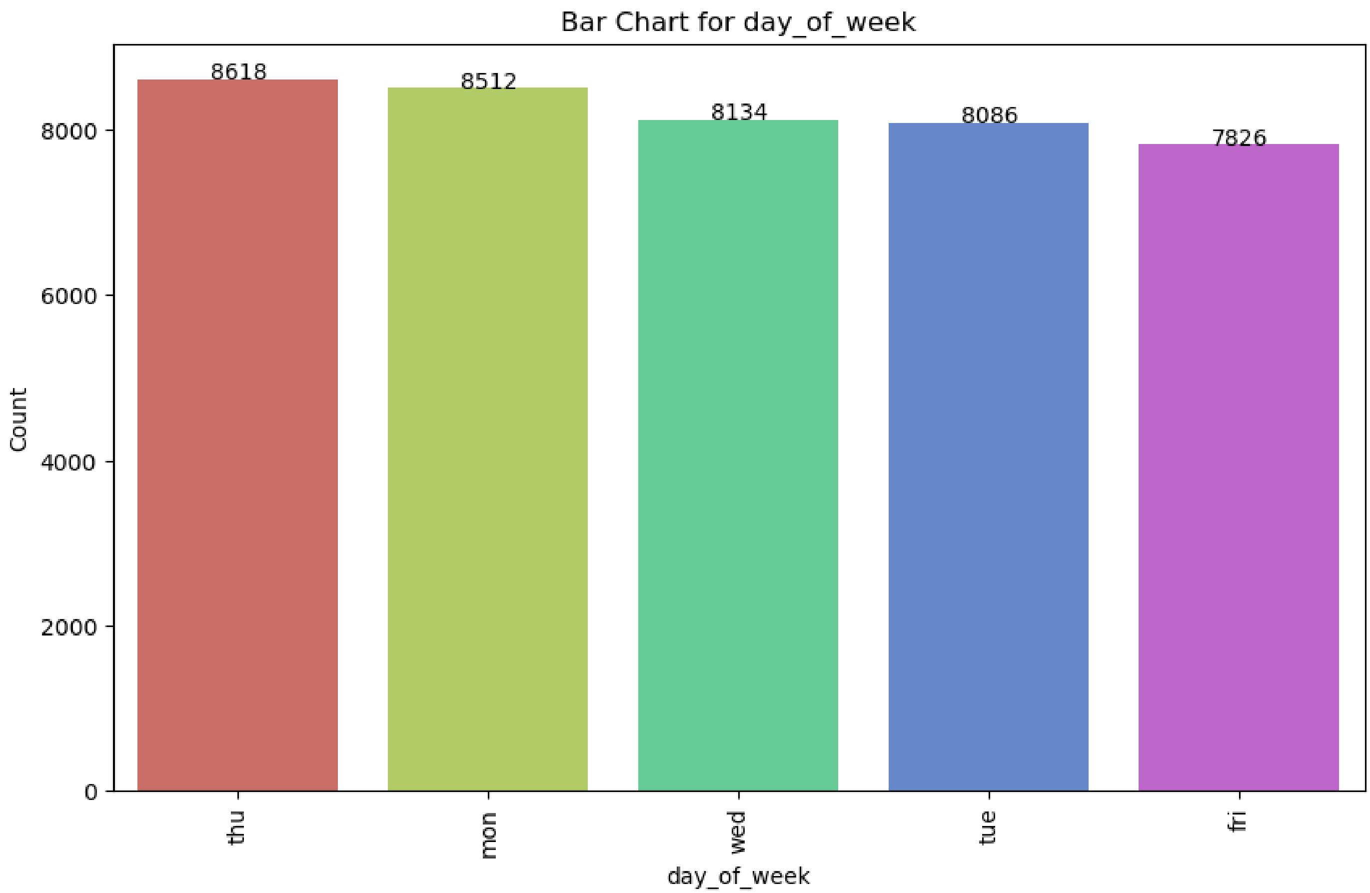

Bar Chart for default
Count
default
no
unknown
yes
32577
8596
3
0
5000
10000
15000
20000
25000
30000

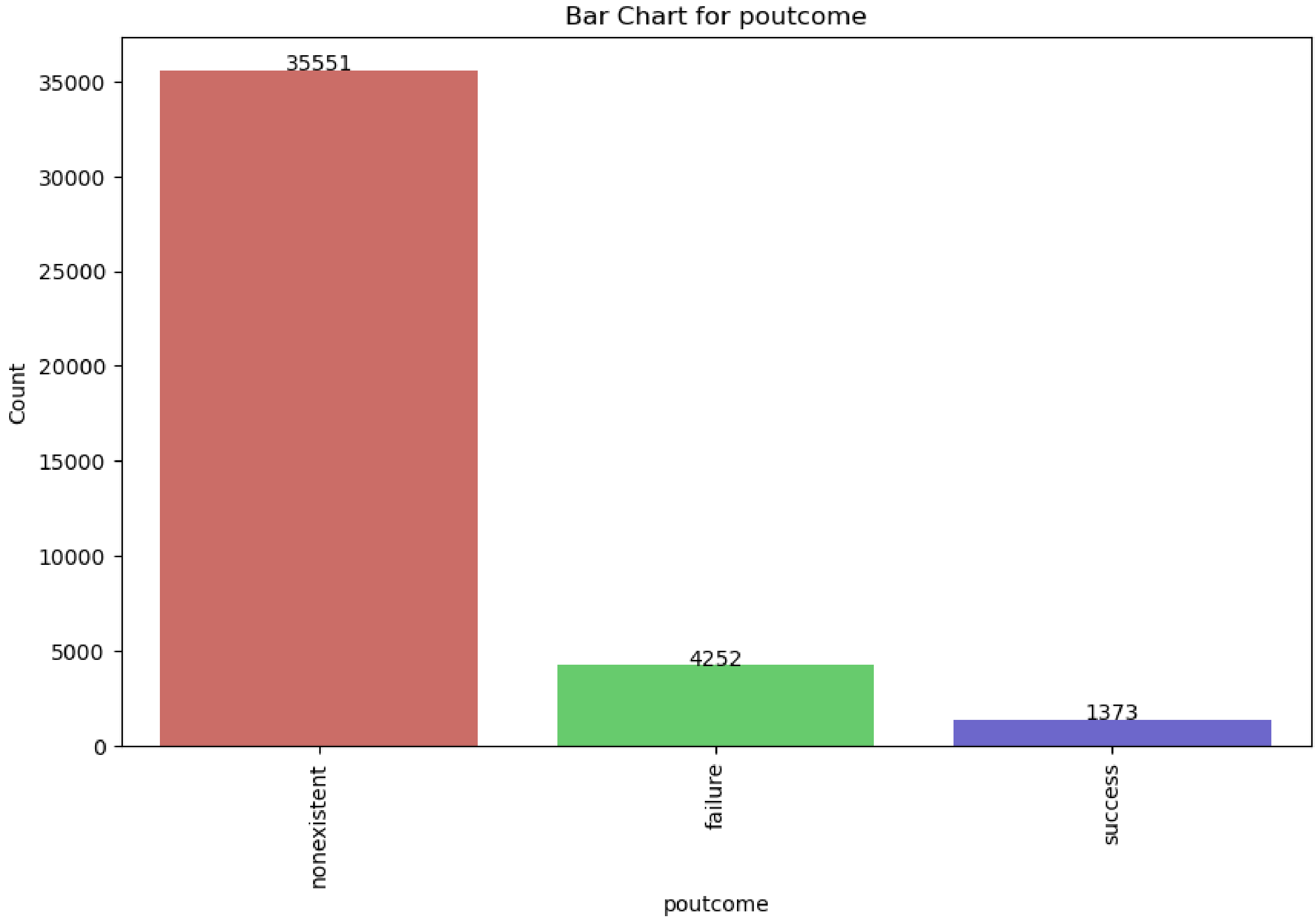

Bar Chart for housing
Count
housing
yes
no
unknown
21571
18615
990
0
5000
10000
15000
20000

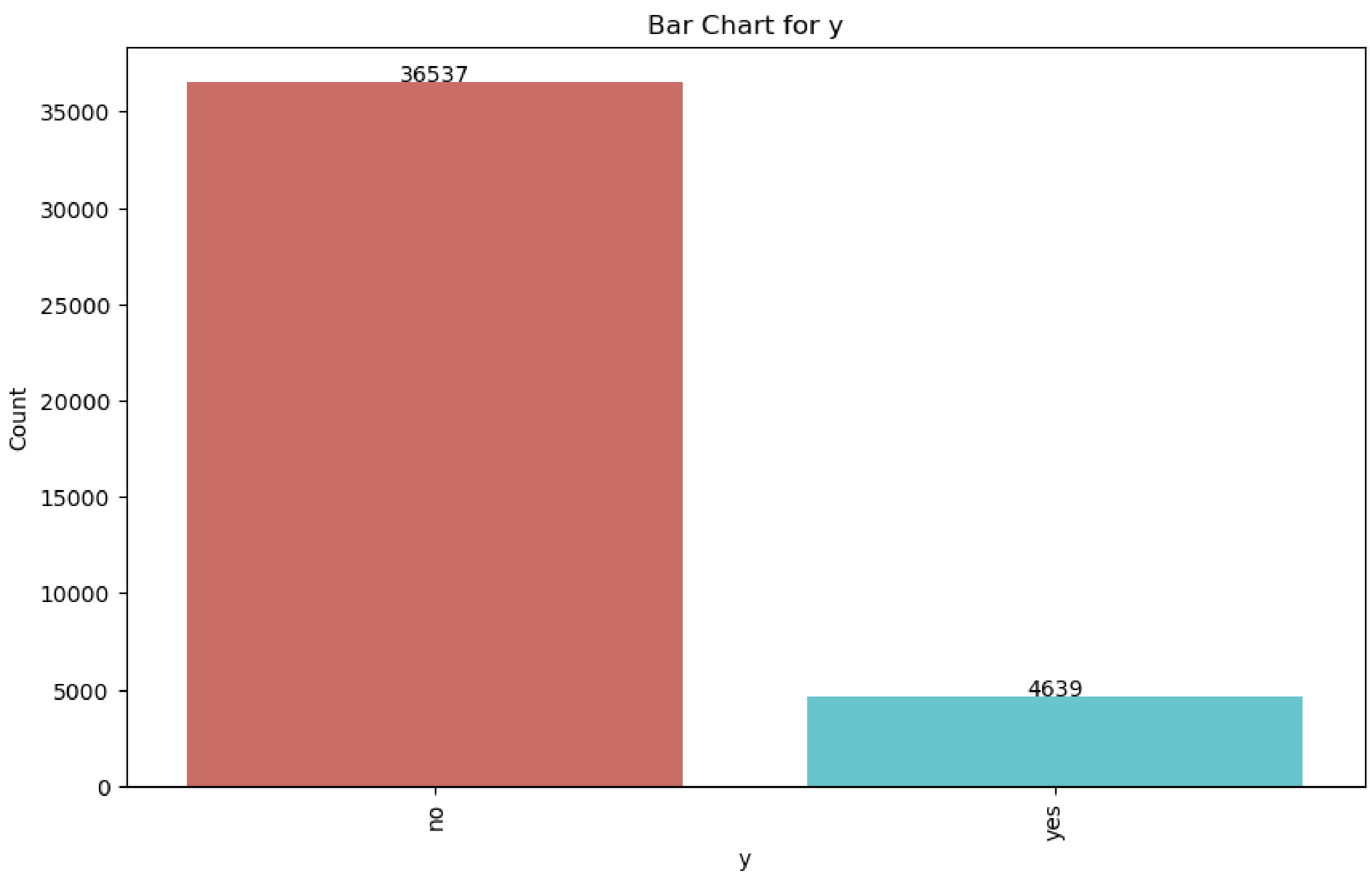

Bar Chart for loan
Count
loan
no
yes
unknown
33938
6248
990
35000
30000
25000
20000
15000
10000
5000
0


Bar Chart for contact
Count
contact
cellular
telephone
26135
15041
25000
20000
15000
10000
5000
0


Bar Chart for month
Count
month
13767
7169
6176
5318
4100
2631
717
570
546
182
may
jul
aug
jun
nov
apr
oct
sep
mar
dec
0
2000
4000
6000
8000
10000
12000
14000

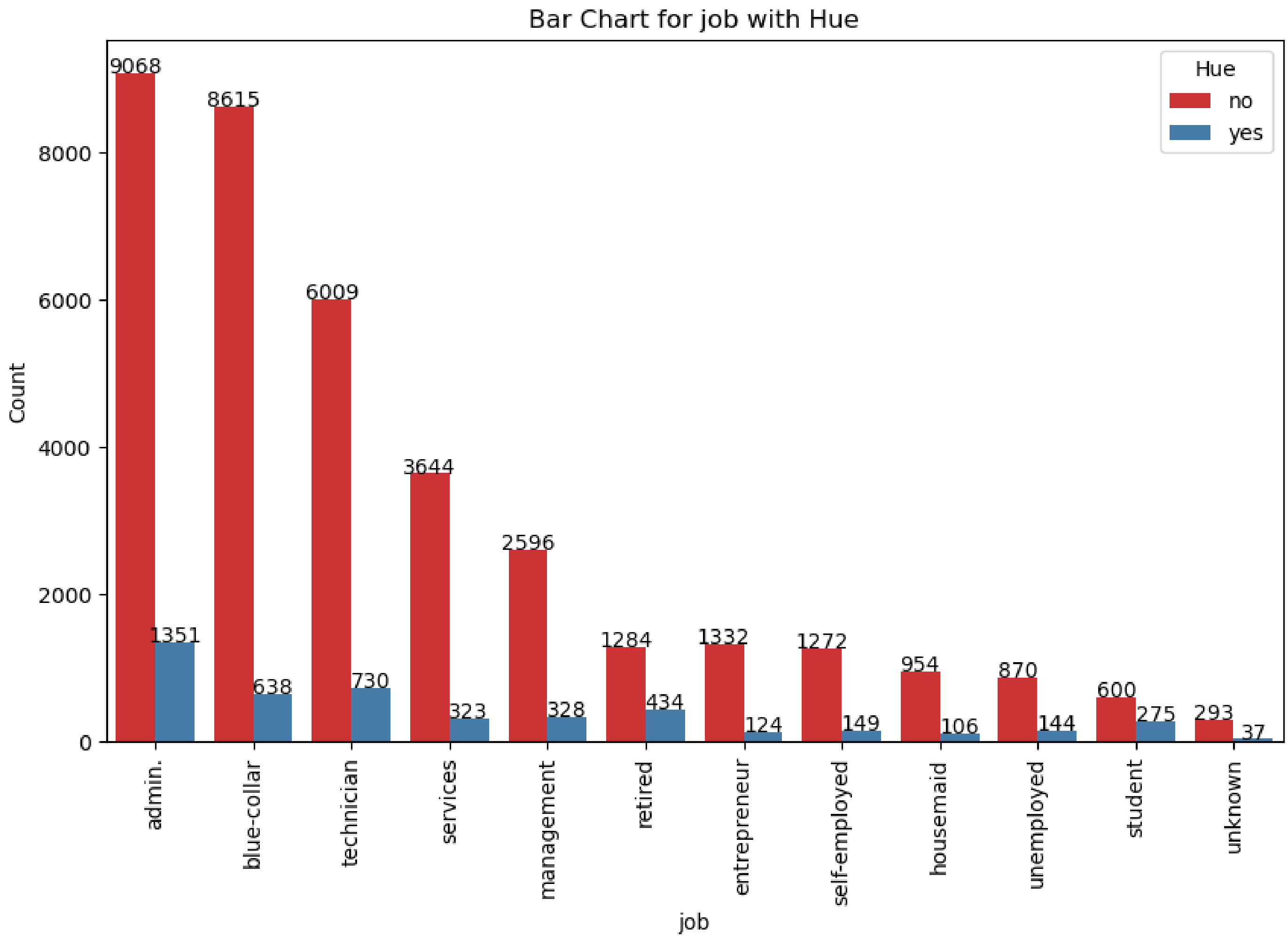

Bar Chart for day_of_week
Count
day_of_week
8618
8512
8134
8086
7826
thu
mon
wed
tue
fri
0
2000
4000
6000
8000

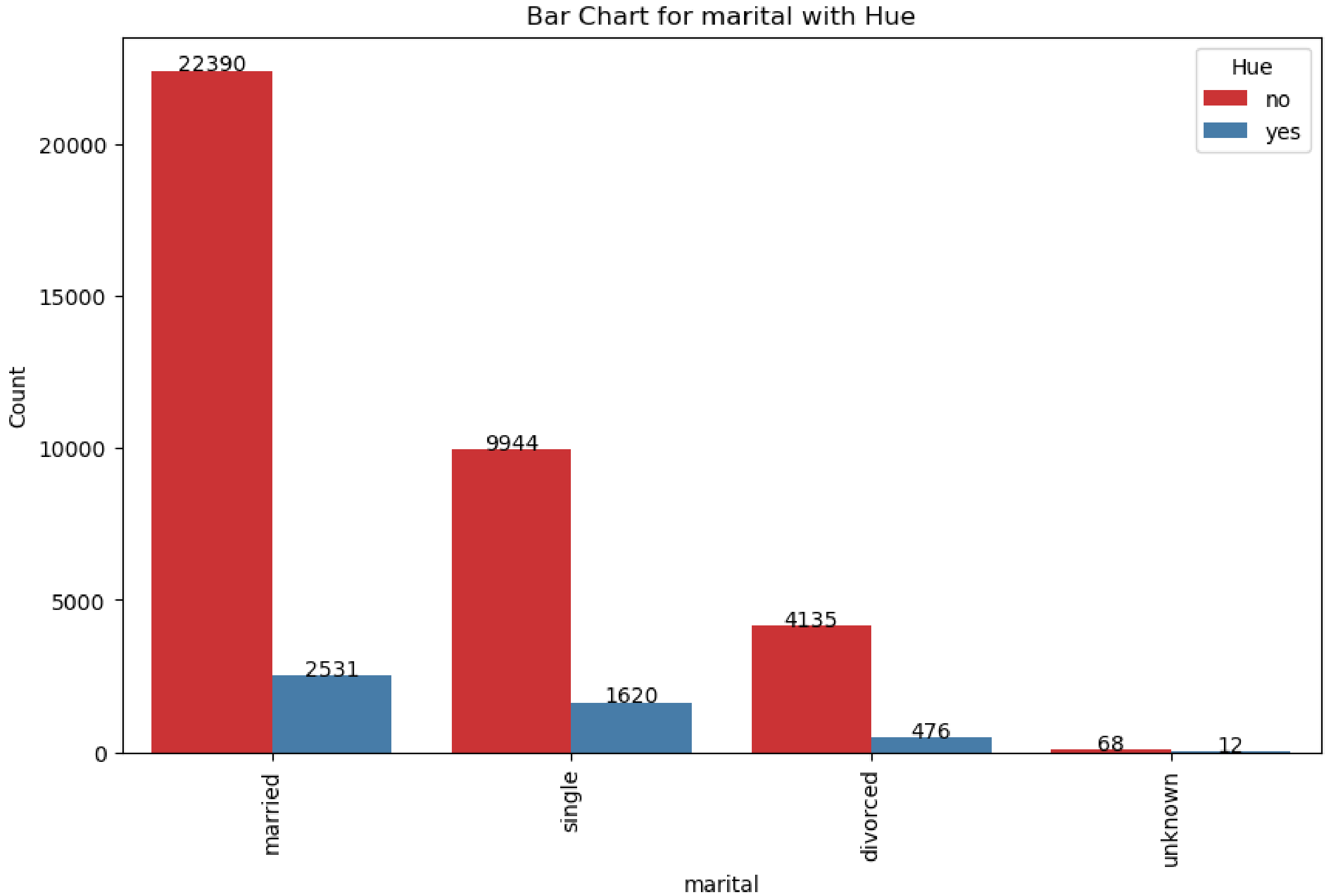




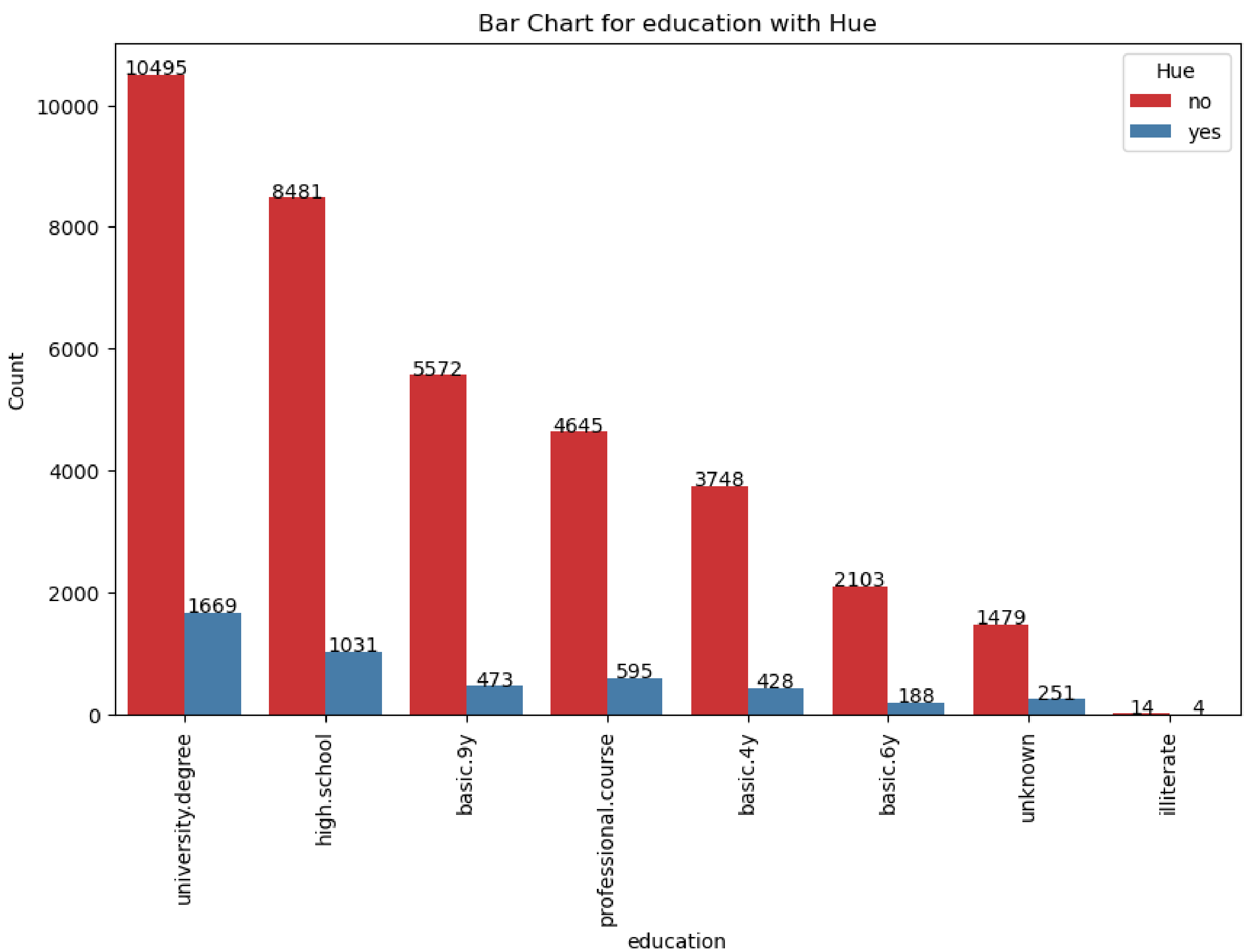
In [13]:
```
object_columns = df.select_dtypes(include=['object']).columns.tolist()
for i in object_columns:
    sorted_order = df[i].value_counts().index

    plt.figure(figsize=(10, 6))
    ax = sns.countplot(data=df, x=i, hue='y', palette='Set1', order=sorted_order)
```

```
    plt.xticks(rotation=90)
    plt.xlabel(i)
    plt.ylabel('Count')
    plt.title(f'Bar Chart for {i} with Hue')
    for p in ax.patches:
        ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_hei

    plt.legend(title='Hue', loc='upper right')  # Add legend
    plt.show()
```

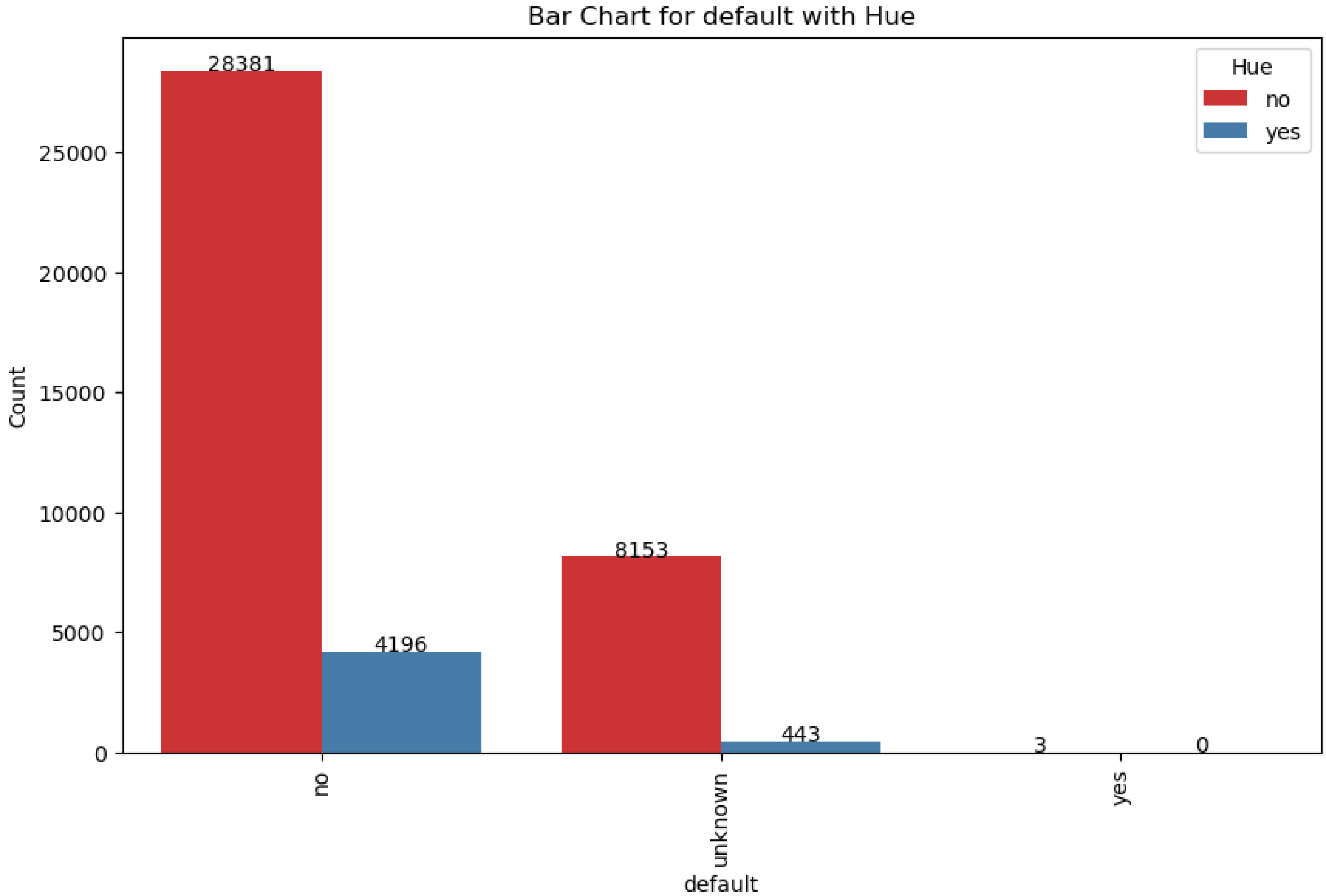




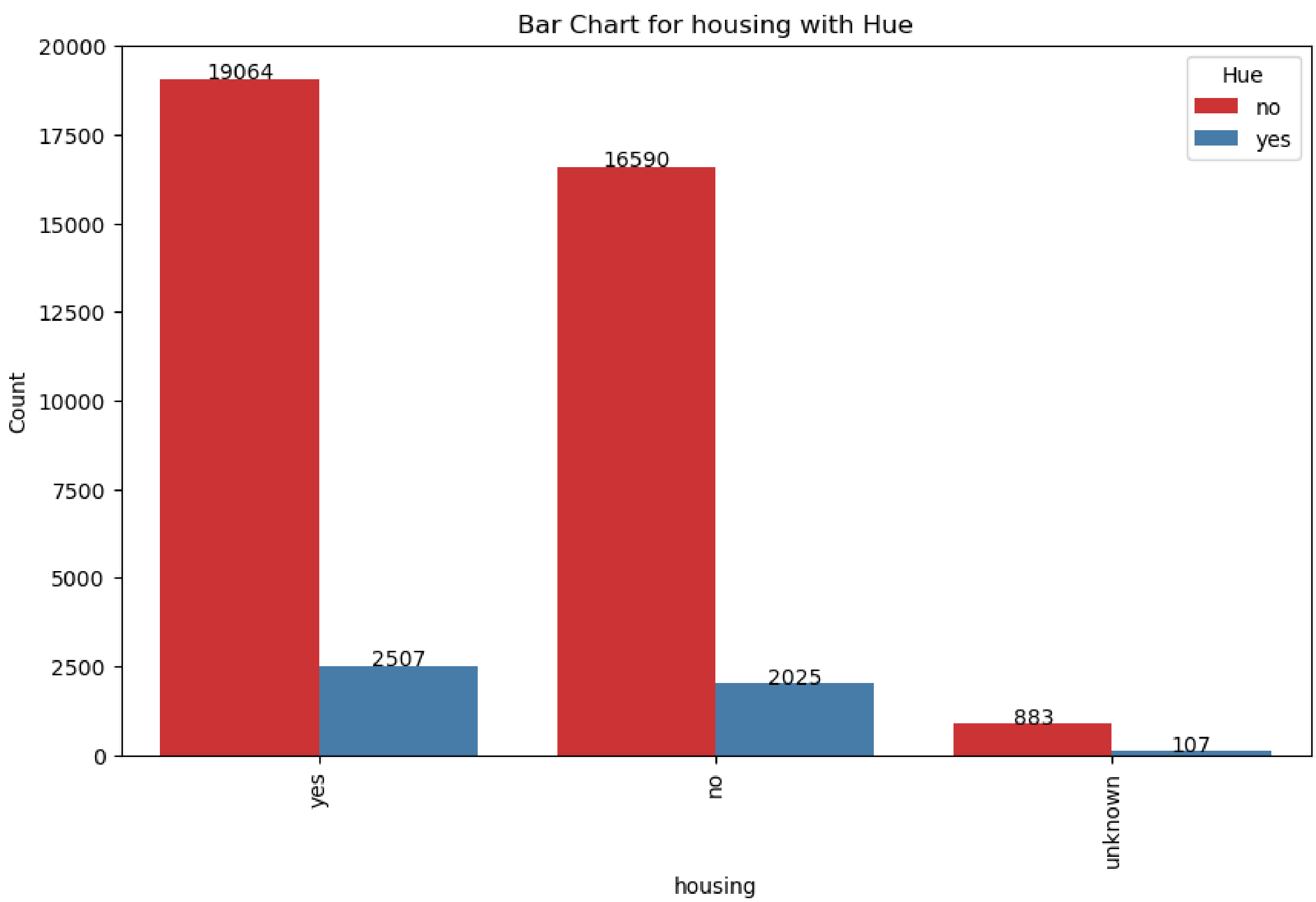
Bar Chart for marital with Hue
Hue
no
yes
22390
2531
9944
1620
4135
476
68
12
Count
20000
15000
10000
5000
0
married
single
divorced
unknown
marital



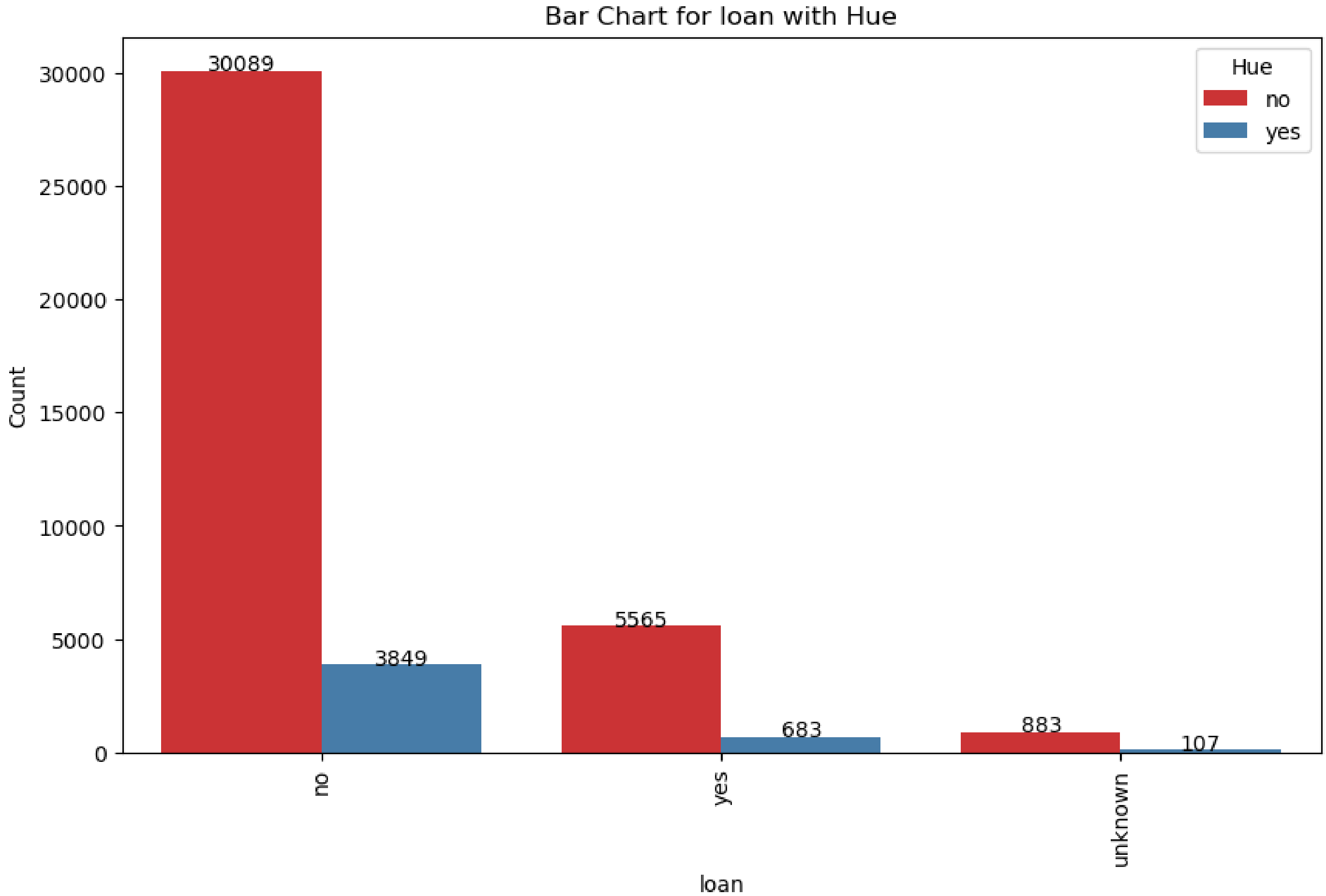
Bar Chart for education with Hue
Hue
no
yes
10495
1669
8481
1031
5572
473
4645
595
3748
428
2103
188
1479
251
14
4
Count
10000
8000
6000
4000
2000
0
university.degree
high.school
basic.9y
professional.course
basic.4y
basic.6y
unknown
illiterate
education

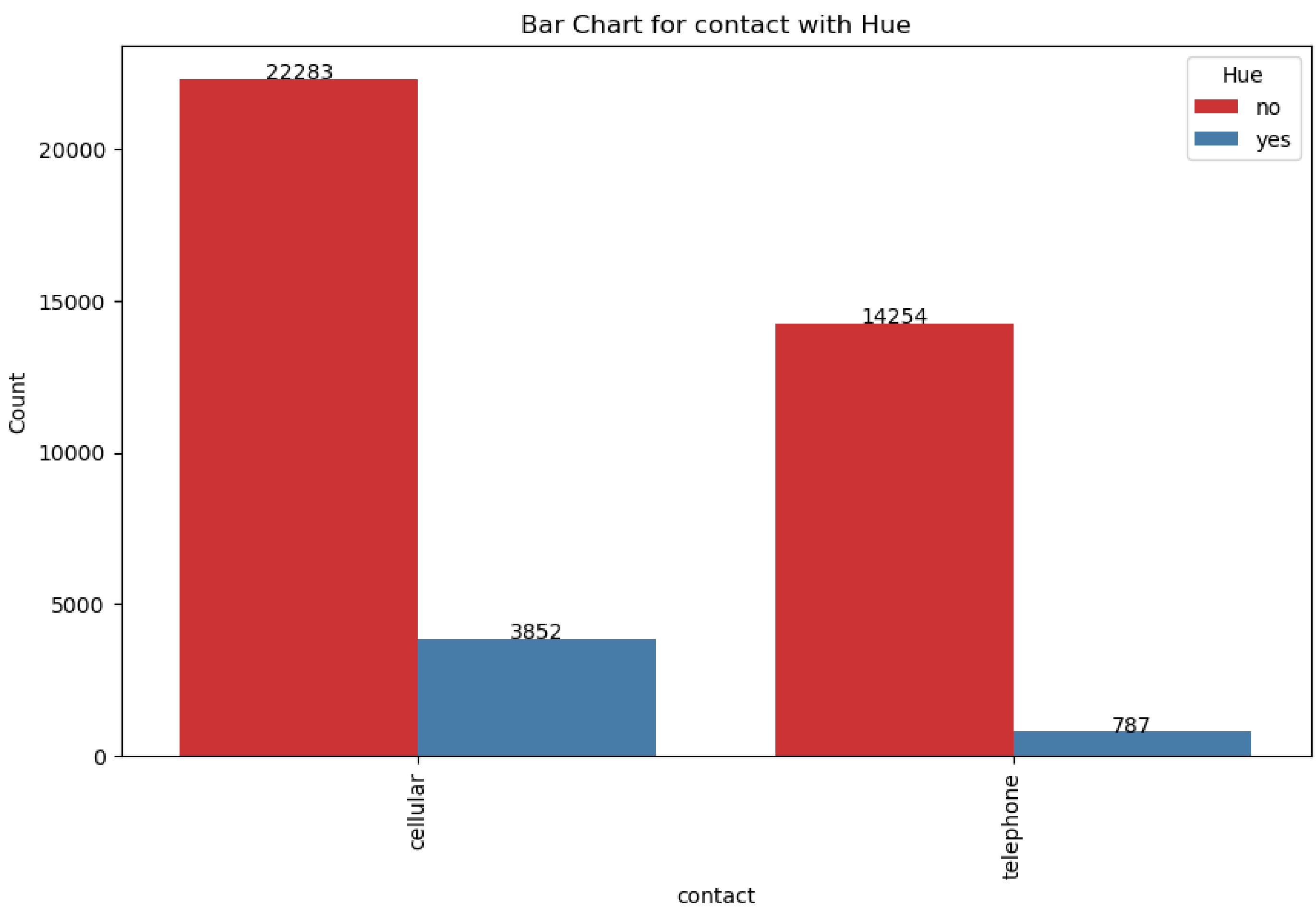


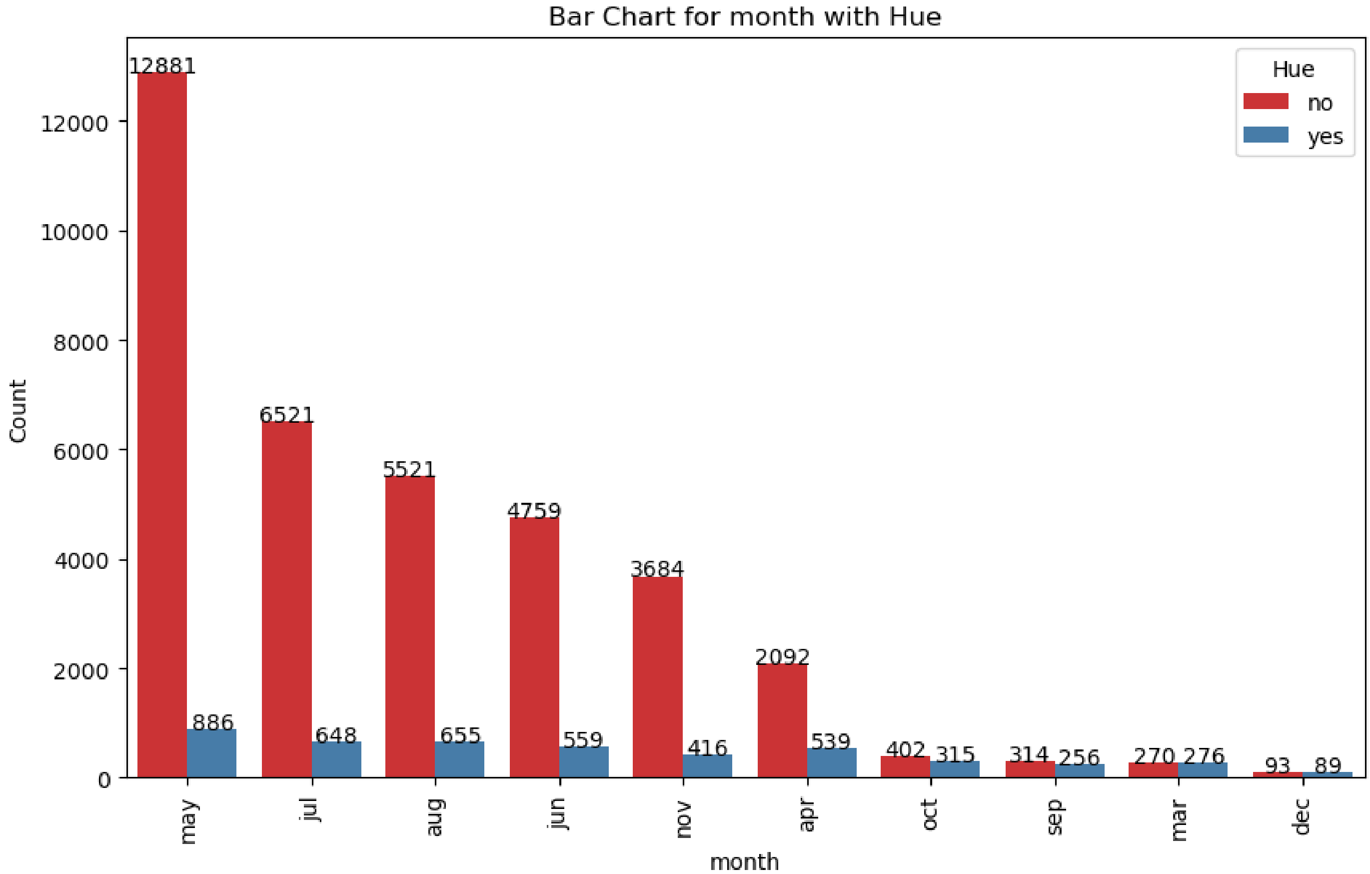
Bar Chart for default with Hue
Hue
no
yes
28381
4196
8153
443
3
0
Count
25000
20000
15000
10000
5000
0
no
unknown
yes
default

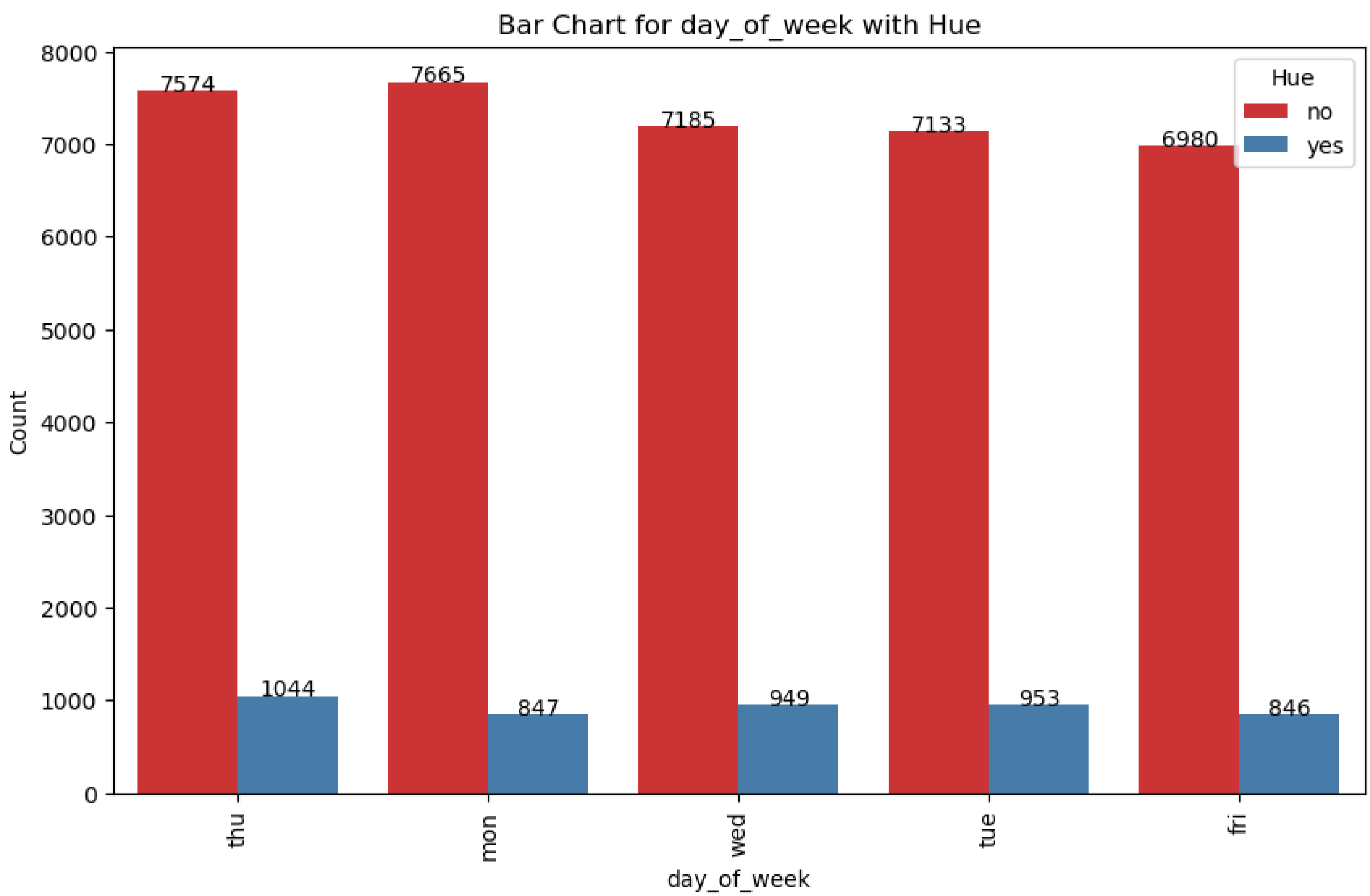


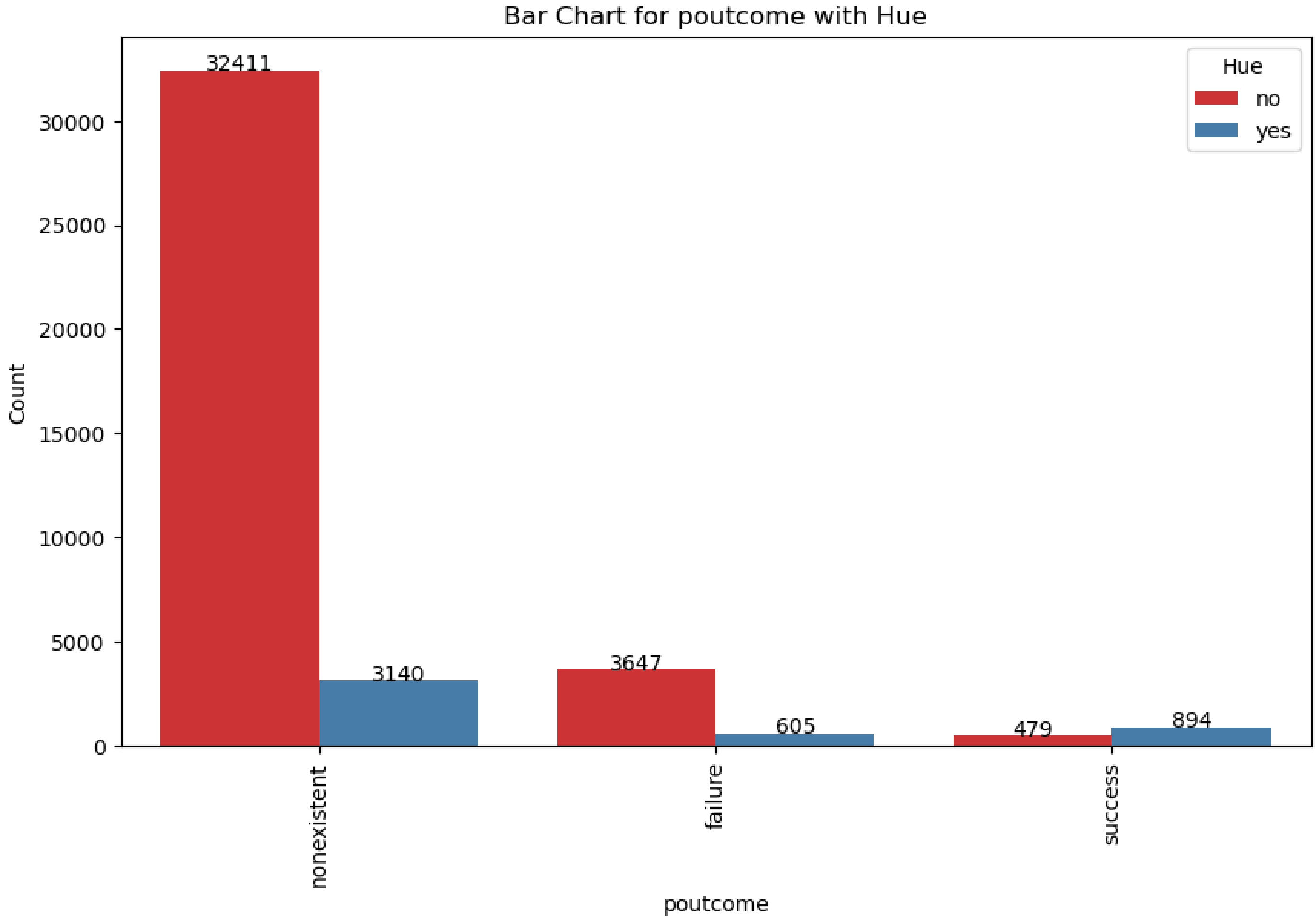
Bar Chart for housing with Hue
Hue
no
yes
19064
2507
16590
2025
883
107
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
yes
no
unknown
housing

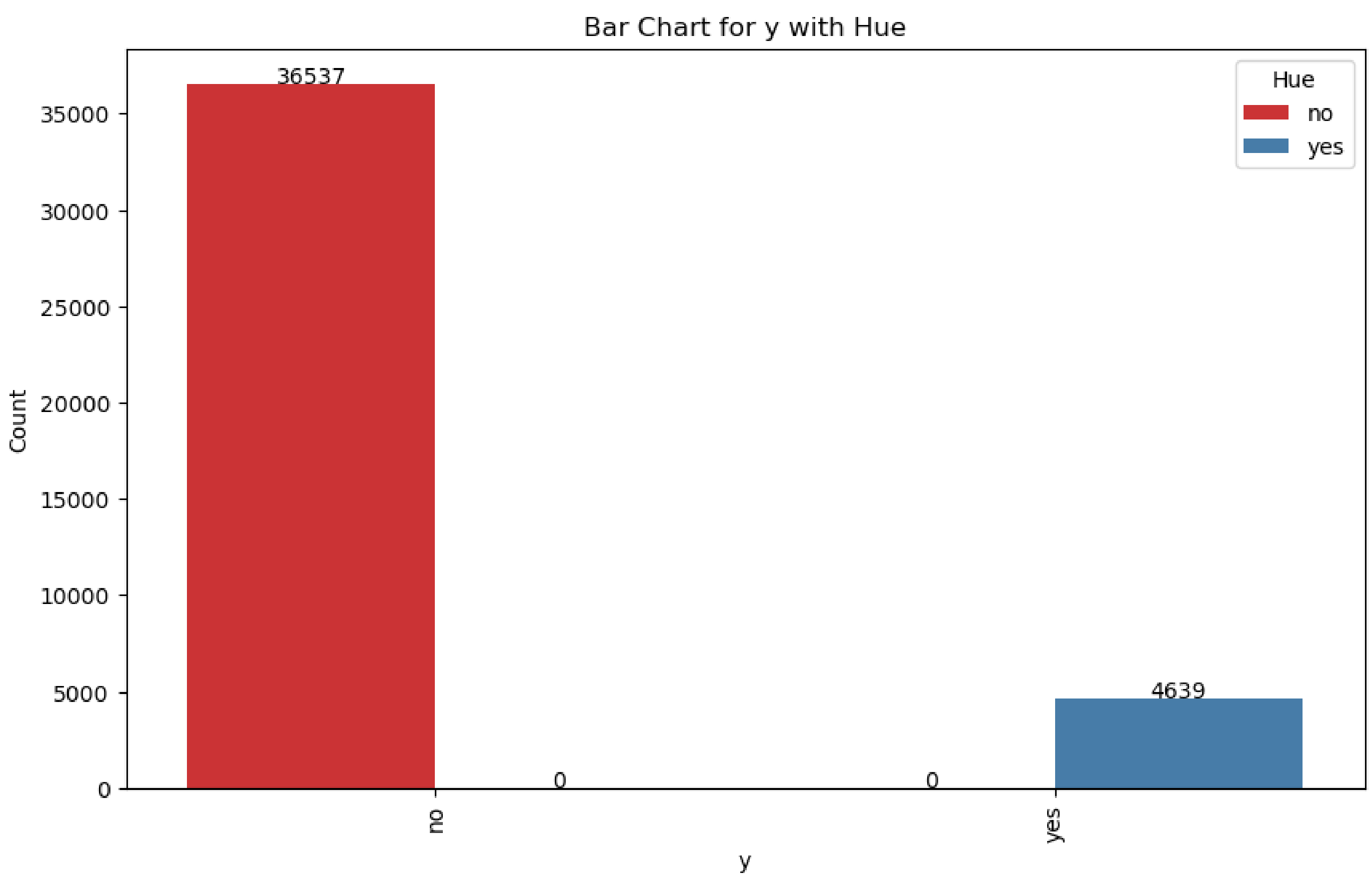

Bar Chart for loan with Hue
Count
loan
Hue
no
yes
30089
3849
5565
683
883
107
0
5000
10000
15000
20000
25000
30000
no
yes
unknown


Bar Chart for contact with Hue
Count
contact
Hue
no
yes
22283
3852
14254
787
0
5000
10000
15000
20000
cellular
telephone


Bar Chart for month with Hue
Hue
no
yes
Count
month
12881
6521
5521
4759
3684
2092
886
648
655
559
416
539
402
315
314
256
270
276
93
89
may
jul
aug
jun
nov
apr
oct
sep
mar
dec


Bar Chart for day_of_week with Hue
Hue
no
yes
Count
day_of_week
7574
7665
7185
7133
6980
1044
847
949
953
846
thu
mon
wed
tue
fri



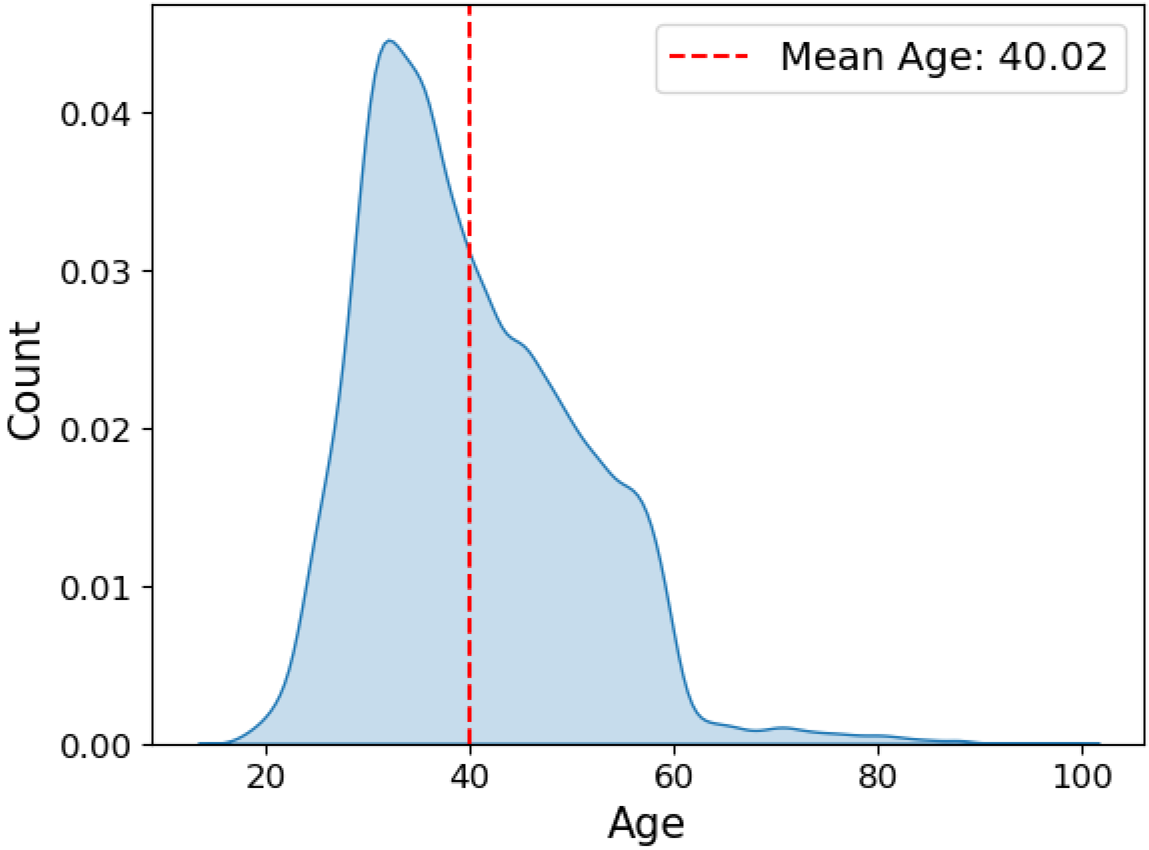


In [14]:

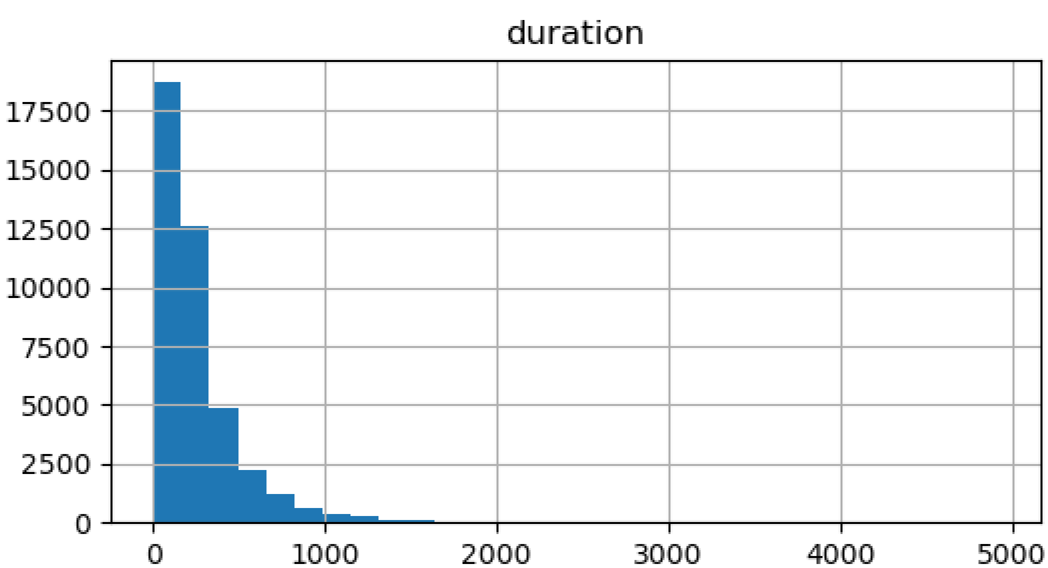
```
sns.kdeplot(x=df['age'],fill=True)
mean_age = df['age'].mean()
plt.axvline(mean_age, color='red', linestyle='--', label=f'Mean Age: {mean_age:.2f}')
plt.xlabel('Age',fontsize=15)
plt.ylabel('Count',fontsize=15)
plt.xticks(fontsize=12)
```

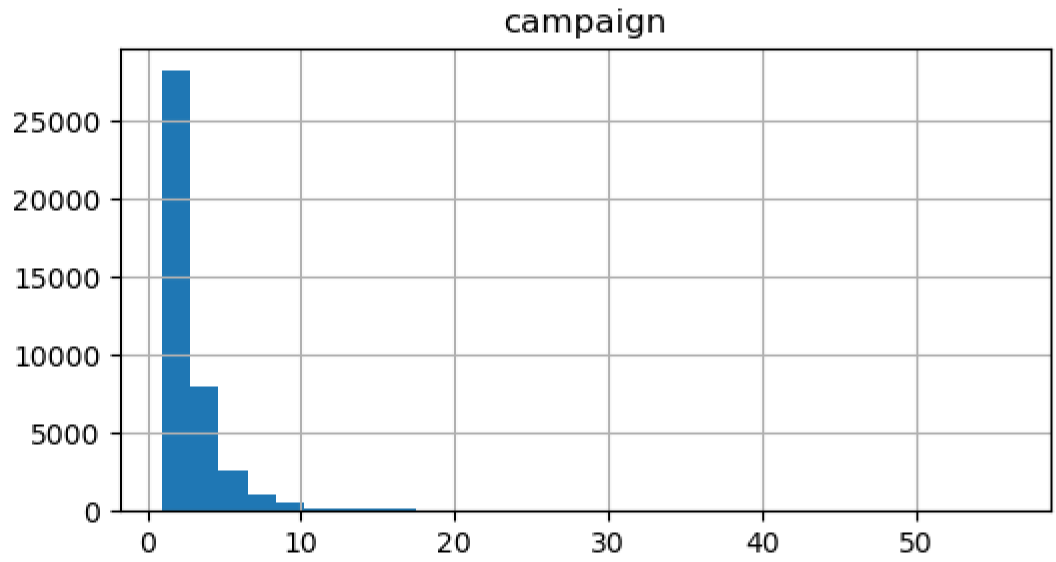
```
plt.yticks(fontsize=12)
plt.legend(fontsize=14, loc='upper right')
```

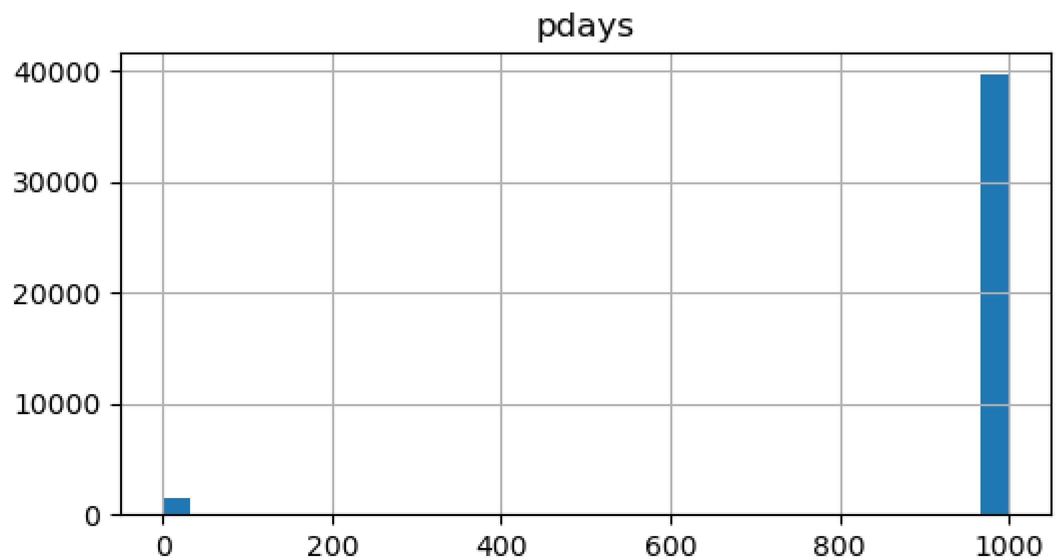
Out[14]: <matplotlib.legend.Legend at 0x2cb16ead350>

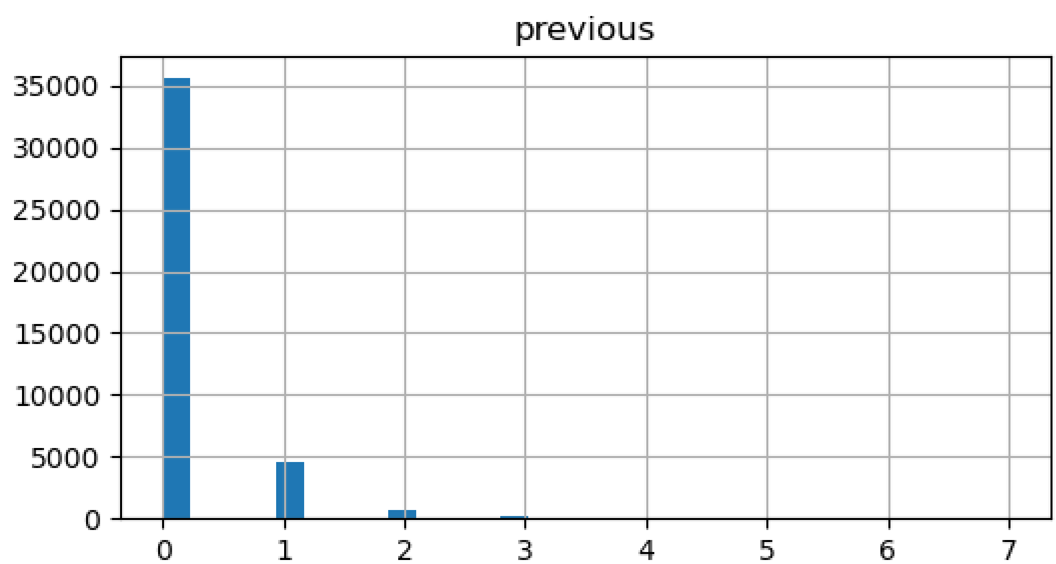


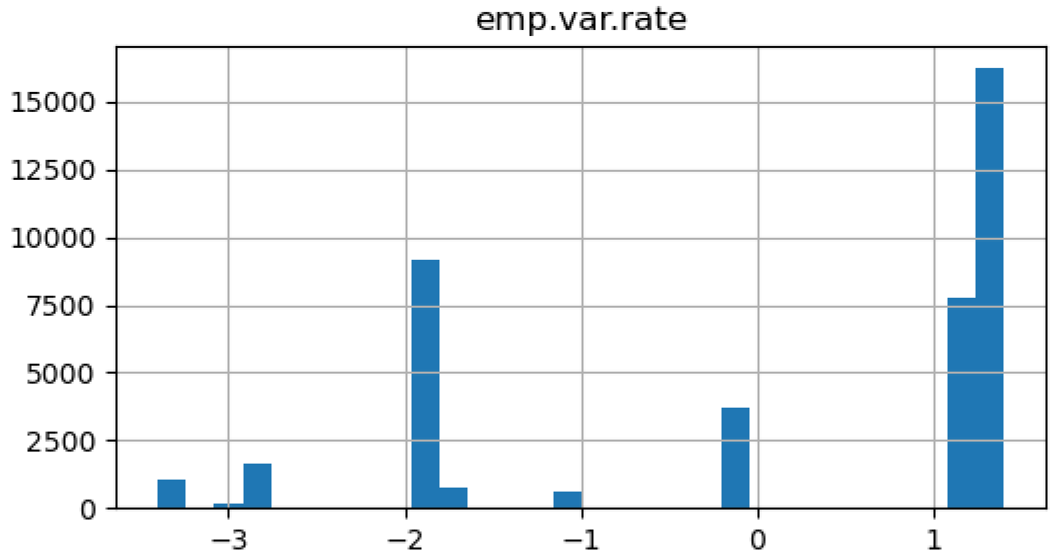
In [15]:
```
num_list = ['duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx
for i in num_list:
    plt.figure(figsize=(6, 3))
    df[i].hist(bins=30)
    plt.title(f'{i}')
```

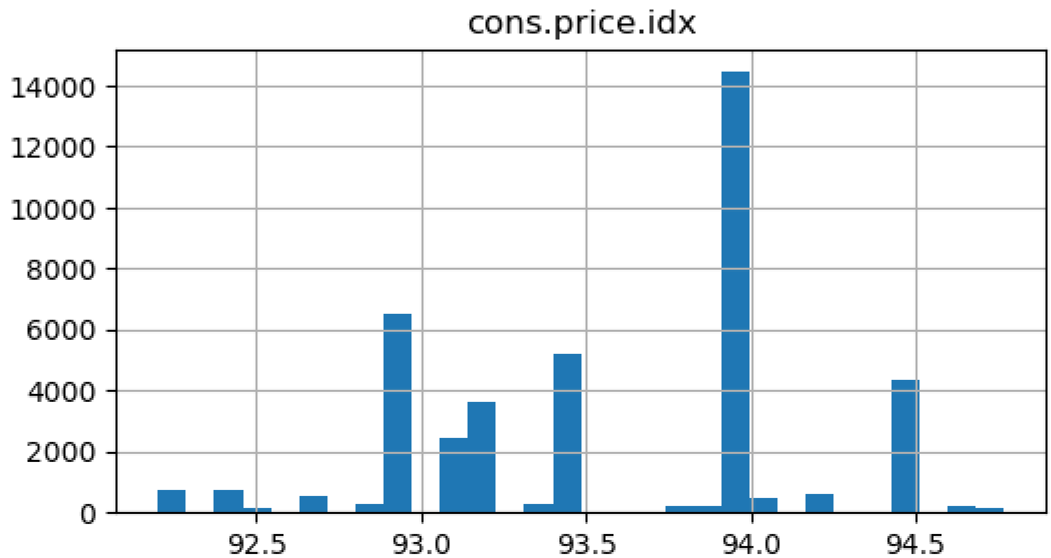


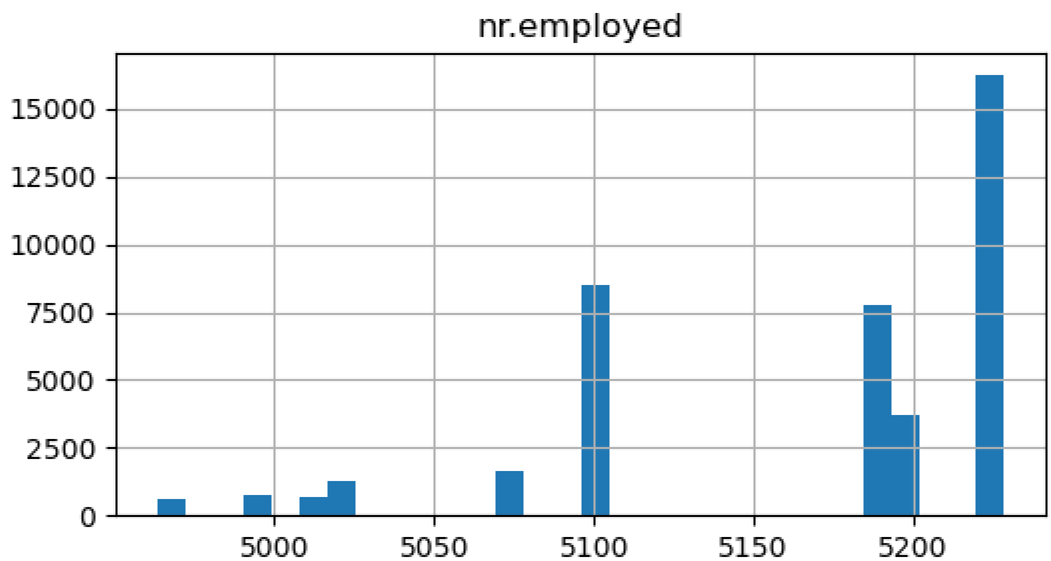

campaign
25000
20000
15000
10000
5000
0
0
10
20
30
40
50

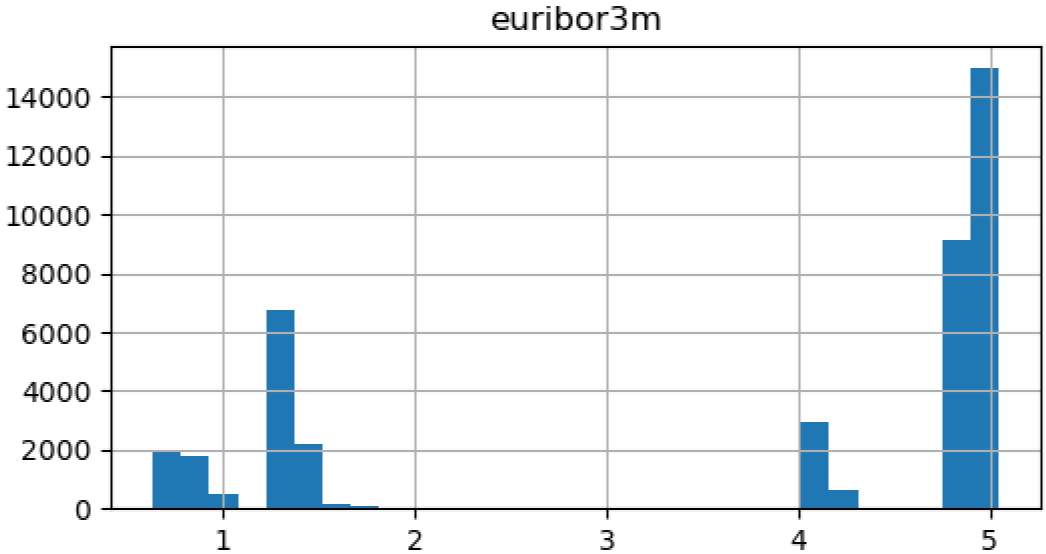

pdays
40000
30000
20000
10000
0
0
200
400
600
800
1000


previous
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
6
7


emp.var.rate
15000
12500
10000
7500
5000
2500
0
−3
−2
−1
0
1

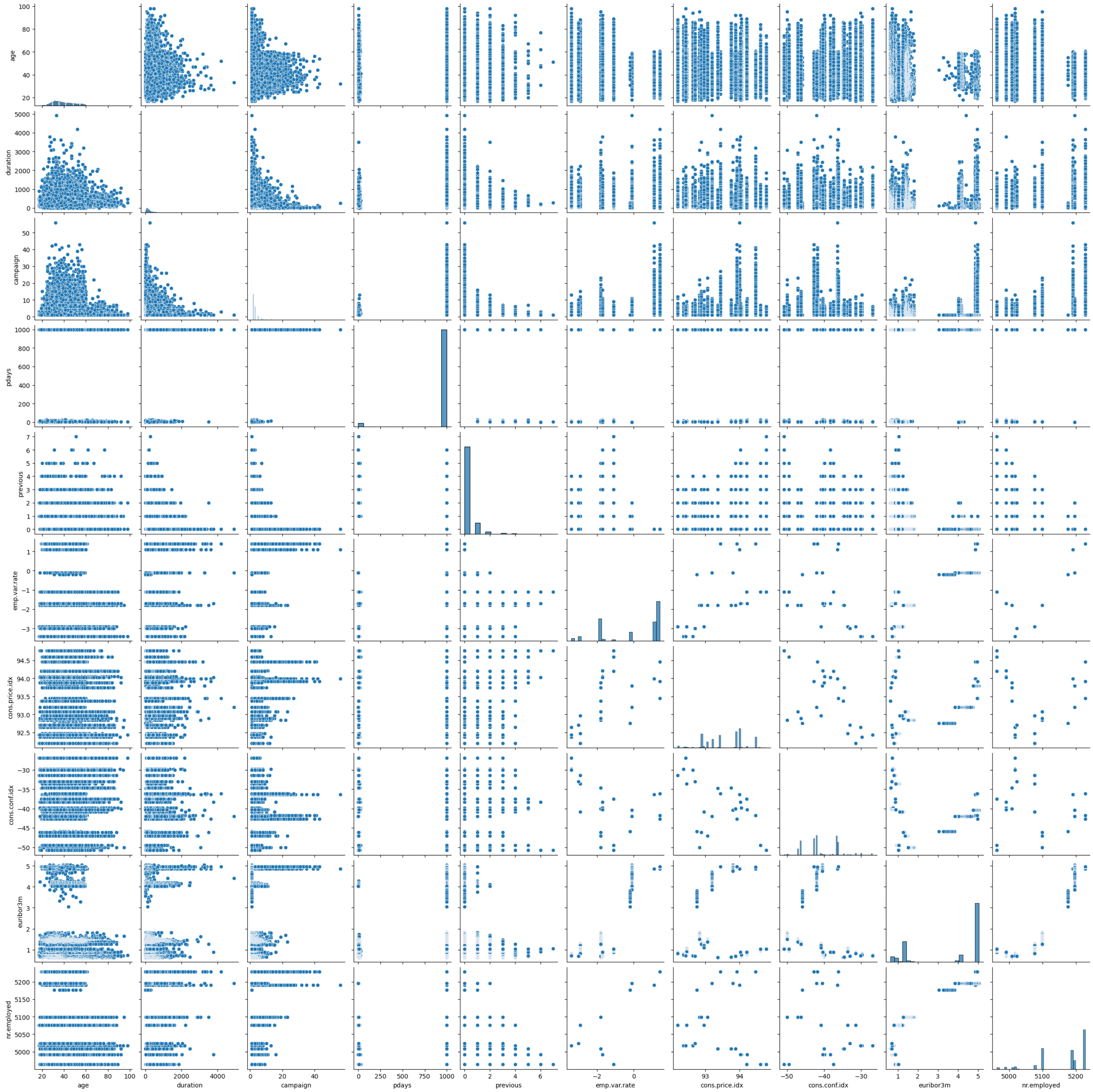

cons.price.idx
14000
12000
10000
8000
6000
4000
2000
0
92.5
93.0
93.5
94.0
94.5


nr.employed
15000
12500
10000
7500
5000
2500
0
5000
5050
5100
5150
5200



In [16]:
```
sns.pairplot(df)
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x2cb14e2d950>
```



In [17]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in ['job', 'marital', 'contact', 'education', 'month', 'poutcome', 'default',
    df[column] = label_encoder.fit_transform(df[column])
# df['job'] = label_encoder.fit_transform(df['job'])
df.drop(['day_of_week'],axis=1,inplace=True)
```

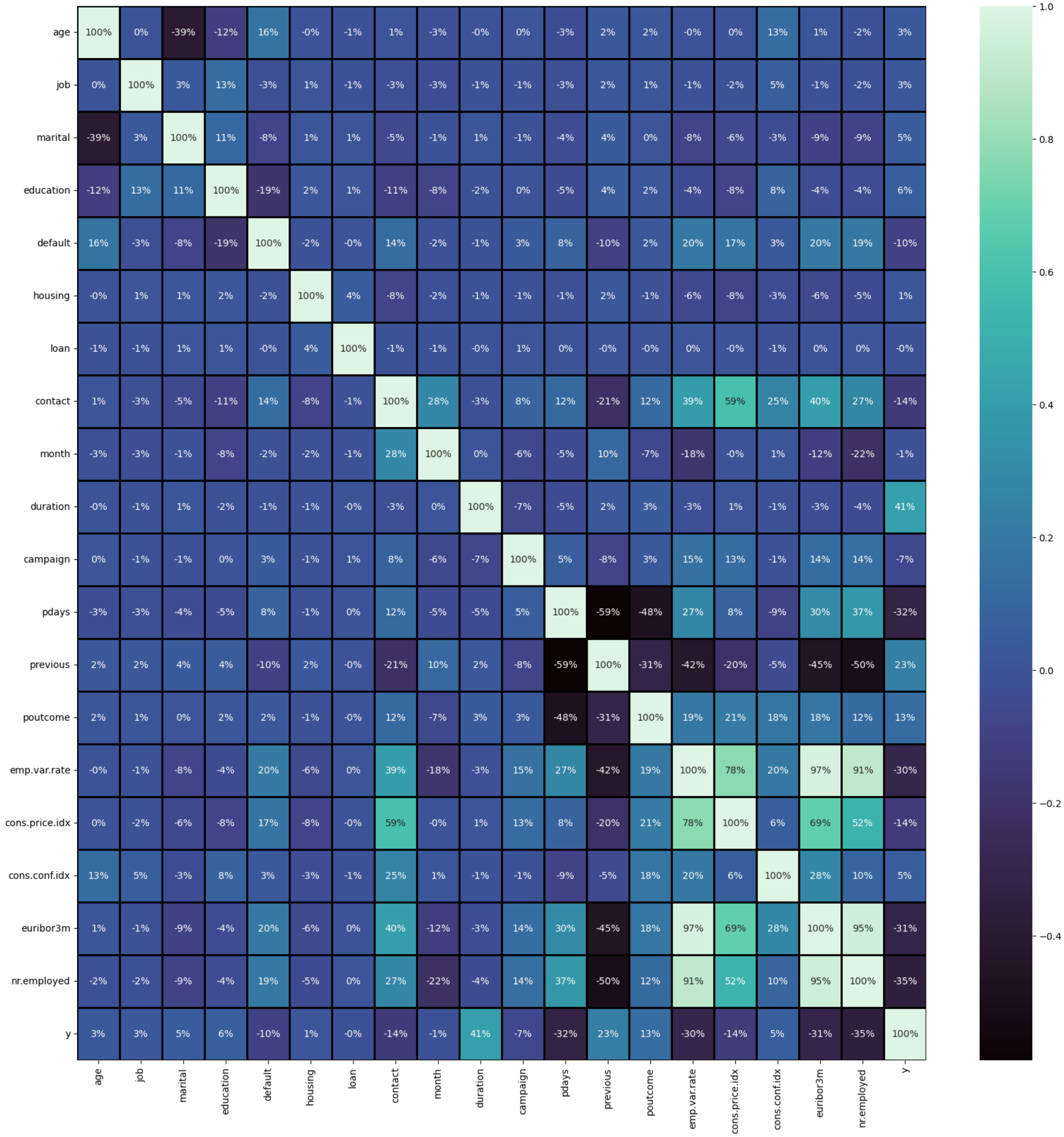
In [18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41176 entries, 0 to 41187
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41176 non-null  int64
 1   job             41176 non-null  int32
 2   marital         41176 non-null  int32
 3   education       41176 non-null  int32
 4   default         41176 non-null  int32
 5   housing         41176 non-null  int32
 6   loan            41176 non-null  int32
```

```
 7   contact          41176 non-null  int32
 8   month            41176 non-null  int32
 9   duration         41176 non-null  int64
 10  campaign         41176 non-null  int64
 11  pdays            41176 non-null  int64
 12  previous         41176 non-null  int64
 13  poutcome         41176 non-null  int32
 14  emp.var.rate     41176 non-null  float64
 15  cons.price.idx   41176 non-null  float64
 16  cons.conf.idx    41176 non-null  float64
 17  euribor3m        41176 non-null  float64
 18  nr.employed      41176 non-null  float64
 19  y                41176 non-null  int32
dtypes: float64(5), int32(10), int64(5)
memory usage: 5.0 MB
```

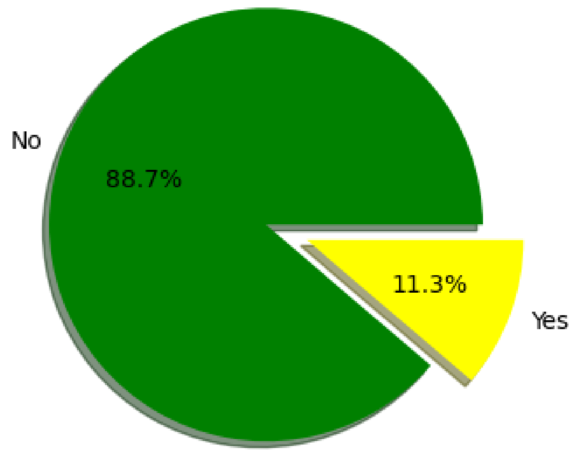
In [19]:

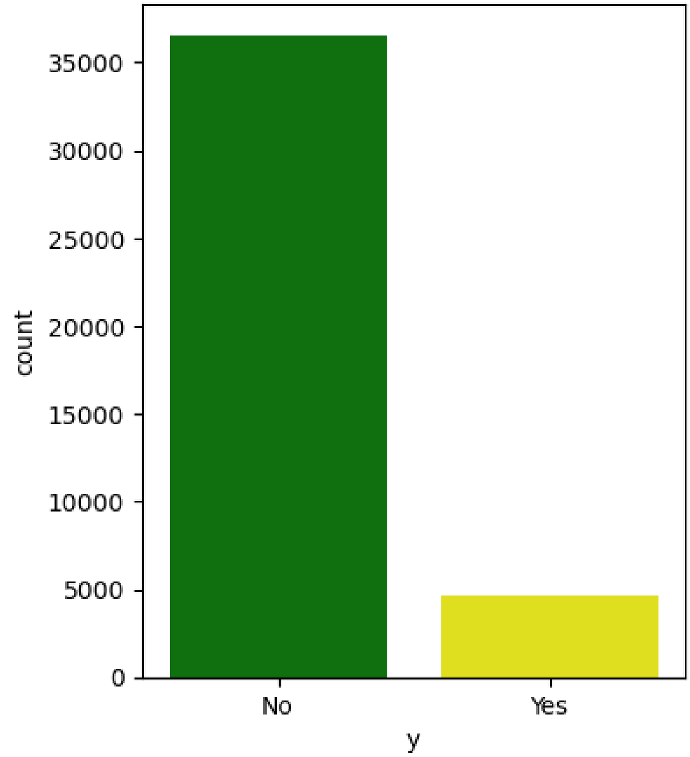
```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),annot=True,cmap='mako',linecolor='black',linewidth=1,fmt=".0%")
plt.show()
```



In [20]:

```
plt.figure(figsize=(10,5))
label_count = df['y'].value_counts().values
label = df['y'].value_counts().index

plt.subplot(1, 2, 1)
explode=(0,0.2)
colors = ['green', 'yellow']
labels = ['No', 'Yes']
plt.pie(x=label_count, labels=labels, autopct='%1.1f%%', shadow=True, radius=1, colors=c

plt.subplot(1, 2, 2)
colors = [ 'green','yellow']
sns.countplot(x=df['y'],palette=colors)
plt.xticks([0, 1], ['No','Yes'])
plt.subplots_adjust(wspace=0.5)
plt.show()
```





In [21]:
```
x=df.iloc[:,:-1]
x
```

Out[21]:

	age	job	marital	education	default	housing	loan	contact	month	duration	campaign	pdays	previ
0	56	3	1	0	0	0	0	1	6	261	1	999	
1	57	7	1	3	1	0	0	1	6	149	1	999	
2	37	7	1	3	0	2	0	1	6	226	1	999	
3	40	0	1	1	0	0	0	1	6	151	1	999	
4	56	7	1	3	0	0	2	1	6	307	1	999	
...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	5	1	5	0	2	0	0	7	334	1	999	
41184	46	1	1	5	0	0	0	0	7	383	1	999	
41185	56	5	1	6	0	2	0	0	7	189	2	999	
41186	44	9	1	5	0	0	0	0	7	442	1	999	
41187	74	5	1	5	0	2	0	0	7	239	3	999	

41176 rows × 19 columns

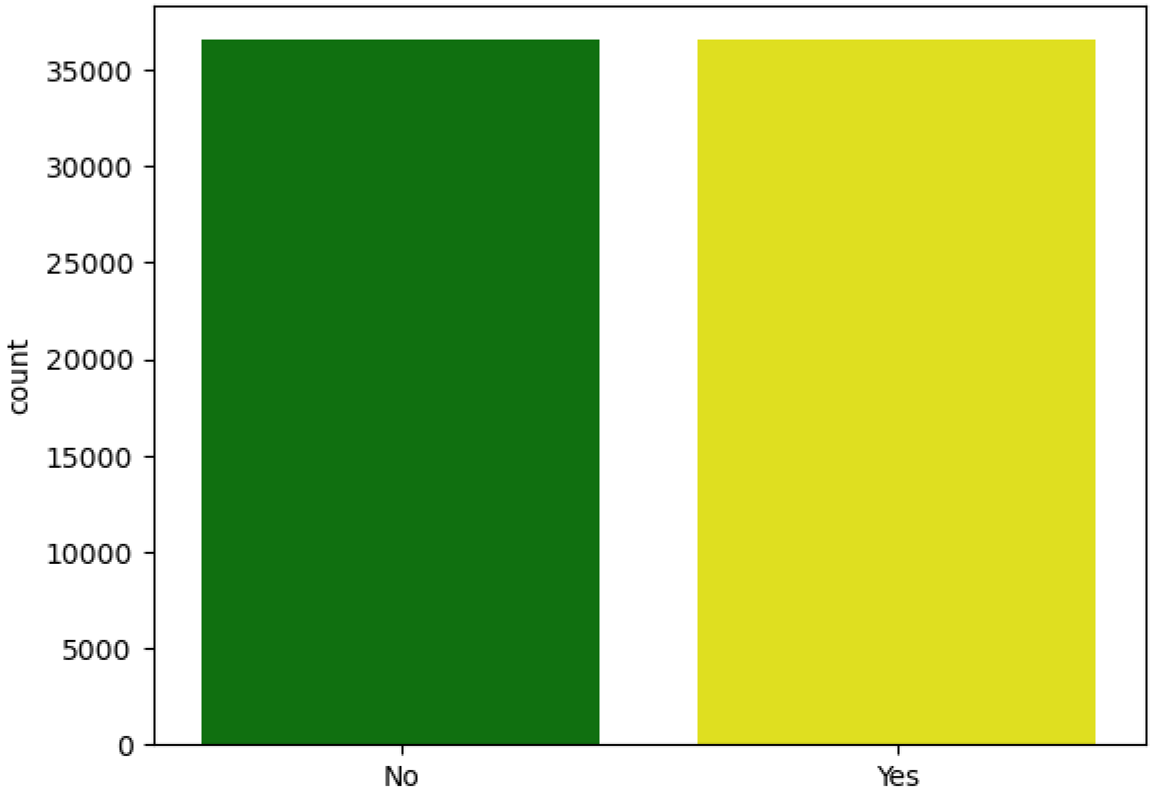
In [22]:
```
y=df.iloc[:,-1].values
y
```

Out[22]:
```
array([0, 0, 0, ..., 0, 1, 0])
```

In [23]:
```
from imblearn.over_sampling import SMOTE
x_resample, y_resample = SMOTE().fit_resample(x, y)
```

In [24]:
```
colors = ['green', 'yellow']
sns.countplot(x=y_resample,palette=colors)
```

```python
plt.xticks([0, 1],['No','Yes'])
plt.show()
```



In [25]:
```python
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x_resample,y_resample,test_size=0.30,random_s
```

In [26]:
```python
xtrain
```

Out[26]:

	age	job	marital	education	default	housing	loan	contact	month	duration	campaign	pdays	previ
33908	30	1	2	2	0	2	0	0	6	14	6	999	
65192	47	0	1	0	0	2	0	1	6	940	3	999	
42384	20	7	2	0	0	1	1	0	0	239	1	999	
28136	42	9	1	2	0	0	2	0	0	437	2	999	
63285	51	0	0	0	0	2	0	1	6	941	2	999	
...	...	...	...	...	...	...	...	...	...	...	...	...	
37194	44	0	1	6	0	2	0	0	1	254	2	999	
6265	34	1	1	2	0	0	2	1	6	100	2	999	
54886	30	4	1	5	0	1	1	0	6	1220	2	999	
860	40	4	1	6	0	2	0	1	6	295	2	999	
15795	19	8	2	2	1	2	0	0	3	87	4	999	

51151 rows × 19 columns

In [27]:
```python
xtest
```

Out[27]:

	age	job	marital	education	default	housing	loan	contact	month	duration	campaign	pdays	previ
4293	53	1	1	0	1	0	0	1	6	182	2	999	

28313	35	0	1	3	0	2	0	0	0	1190	1	999
42943	32	5	1	4	0	0	0	1	4	983	2	999
71581	40	1	0	4	0	1	1	0	3	430	1	999
67128	59	0	1	4	0	2	0	0	4	765	1	4
...	...	...	...	...	...	...	...	...	...	...	...	...
8370	35	9	1	5	0	2	0	1	4	589	6	999
3219	45	0	1	1	1	2	0	1	6	124	1	999
66571	36	1	0	3	0	2	0	0	3	812	2	999
69957	67	5	1	0	0	2	0	0	1	265	1	999
4029	32	0	1	3	0	0	0	1	6	70	1	999

21923 rows × 19 columns

In [28]:
```
ytrain
```
Out[28]:
```
array([0, 1, 1, ..., 1, 0, 0])
```

In [29]:
```
ytest
```
Out[29]:
```
array([1, 1, 1, ..., 1, 1, 0])
```

In [30]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

xtrain = scaler.fit_transform(xtrain)
xtest = scaler.transform(xtest)
```

In [31]:
```
xtrain
```
Out[31]:
```
array([[-0.87309781, -0.7714112 ,  1.48649342, ..., -1.14807156,
        -0.89746396, -0.42677908],
       [ 0.60266604, -1.06689382, -0.16839765, ...,  0.72769868,
         0.99704464,  0.63266387],
       [-1.74119419,  1.00148455,  1.48649342, ...,  1.06873402,
        -1.23523523, -1.46892972],
       ...,
       [-0.87309781,  0.11503668, -0.16839765, ...,  0.14827271,
         0.75719869,  0.66257685],
       [-0.00500143,  0.11503668, -0.16839765, ...,  0.72769868,
         0.99574982,  0.63266387],
       [-1.82800383,  1.29696717,  1.48649342, ..., -0.47815362,
         1.05082513,  1.06036065]])
```

In [32]:
```
xtest
```
Out[32]:
```
array([[ 1.12352387, -0.7714112 , -0.16839765, ...,  0.72769868,
         0.99574982,  0.63266387],
       [-0.43904962, -1.06689382, -0.16839765, ..., -1.32033617,
        -0.81061443, -0.42677908],
       [-0.69947853,  0.4105193 , -0.16839765, ..., -0.305889  ,
         1.00062839,  1.06036065],
       ...,
       [-0.35223998, -0.7714112 , -1.82328872, ..., -0.47815362,
         1.05489041,  1.06036065],
       [ 2.3388588 ,  0.4105193 , -0.16839765, ...,  1.40503395,
        -1.09150919, -0.69077521],
```

```
        [-0.69947853, -1.06689382, -0.16839765, ...,  0.72769868,
          0.99680896,  0.63266387]])
```

In [35]:
```python
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(xtrain, ytrain)
ypred=decision_tree.predict(xtest)
ypred
```

Out[35]:
```
array([0, 0, 1, ..., 1, 1, 0])
```

In [36]:
```python
from sklearn.metrics import accuracy_score,classification_report
accuracy = accuracy_score(ytest, ypred)
print("Decision Tree:", accuracy)

print(classification_report(ytest,ypred))
```

```
Decision Tree: 0.92345938055923
              precision    recall  f1-score   support

           0       0.93      0.92      0.92     11029
           1       0.92      0.93      0.92     10894

    accuracy                           0.92     21923
   macro avg       0.92      0.92      0.92     21923
weighted avg       0.92      0.92      0.92     21923
```

In [ ]: