In [1]:

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

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [32]:

```
df=pd.read_csv('C:\\Users\\vidya\\Downloads\\bank+marketing\\bank-additional\\ba
```

In [13]:

```
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

Out[13]:

	age	job	marital	education	default	housing	loan	contact	mon
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	m
1	39	services	single	high.school	no	no	no	telephone	m
2	25	services	married	high.school	no	yes	no	telephone	ju
3	38	services	married	basic.9y	no	unknown	unknown	telephone	ju
4	47	admin.	married	university.degree	no	yes	no	cellular	n

5 rows × 21 columns

In [14]:

```
df.head()
```

Out[14]:

	age	job	marital	education	default	housing	loan	contact	mon
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	m
1	39	services	single	high.school	no	no	no	telephone	m
2	25	services	married	high.school	no	yes	no	telephone	ju
3	38	services	married	basic.9y	no	unknown	unknown	telephone	ju
4	47	admin.	married	university.degree	no	yes	no	cellular	n

5 rows × 21 columns

In [15]:

```
df.tail()
```

Out[15]:

	age	job	marital	education	default	housing	loan	contact	month
4114	30	admin.	married	basic.6y	no	yes	yes	cellular	ju
4115	39	admin.	married	high.school	no	yes	no	telephone	ju
4116	27	student	single	high.school	no	no	no	cellular	may
4117	58	admin.	married	high.school	no	no	no	cellular	aug
4118	34	management	single	high.school	no	yes	no	cellular	nov

5 rows × 21 columns

In [16]:
```
df.shape
```

Out[16]:
```
(4119, 21)
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'deposit'],
      dtype='object')
```

In [18]:
```
df.dtypes
```

Out[18]:
```
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
deposit            object
dtype: object
```

In [19]:
```
df.dtypes.value_counts()
```

Out[19]:
```
object     11
int64       5
float64     5
Name: count, dtype: int64
```

In [20]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4119 entries, 0 to 4118
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             4119 non-null   int64  
 1   job             4119 non-null   object 
 2   marital         4119 non-null   object 
 3   education       4119 non-null   object 
 4   default         4119 non-null   object 
 5   housing         4119 non-null   object 
 6   loan            4119 non-null   object 
 7   contact         4119 non-null   object 
 8   month           4119 non-null   object 
 9   day_of_week     4119 non-null   object 
 10  duration        4119 non-null   int64  
 11  campaign        4119 non-null   int64  
 12  pdays           4119 non-null   int64  
 13  previous        4119 non-null   int64  
 14  poutcome        4119 non-null   object 
 15  emp.var.rate    4119 non-null   float64
 16  cons.price.idx  4119 non-null   float64
 17  cons.conf.idx   4119 non-null   float64
 18  euribor3m       4119 non-null   float64
 19  nr.employed     4119 non-null   float64
 20  deposit         4119 non-null   object 
dtypes: float64(5), int64(5), object(11)
memory usage: 675.9+ KB
```

In [21]:

```
df.duplicated().sum()
```

Out[21]:

```
0
```

In [22]:

```
df.isna().sum()
```

Out[22]:

```
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
deposit           0
dtype: int64
```

In [23]:
```
cat_cols = df.select_dtypes(include='object').columns
print(cat_cols)

num_cols = df.select_dtypes(exclude='object').columns
print(num_cols)
```

```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'day_of_week', 'poutcome', 'deposit'],
      dtype='object')
Index(['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate',
       'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'],
      dtype='object')
```

In [24]:
```
df.describe()
```

Out[24]:

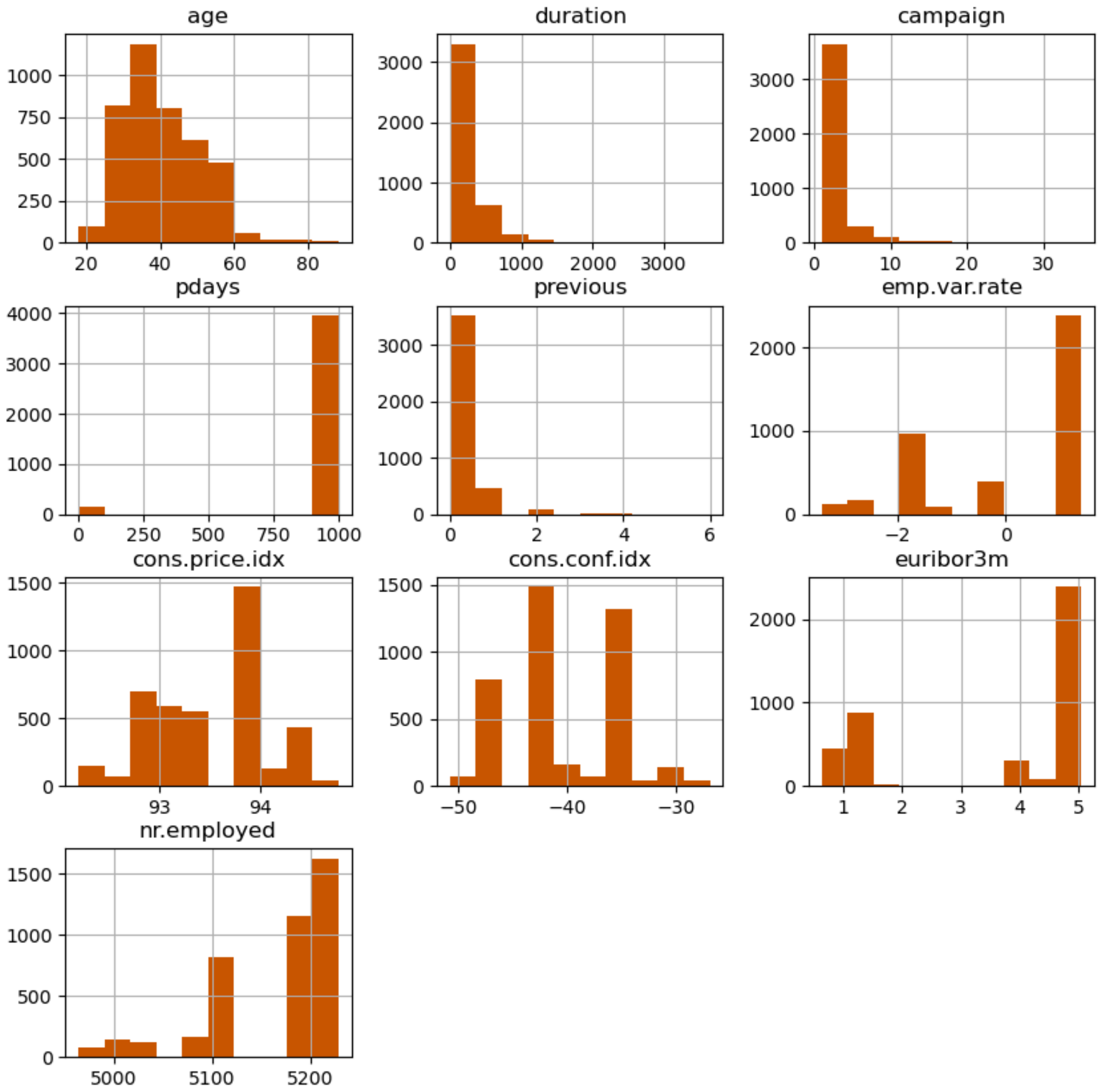
	age	duration	campaign	pdays	previous	emp.var.rate
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000

In [25]:
```
df.describe(include='object')
```

Out[25]:

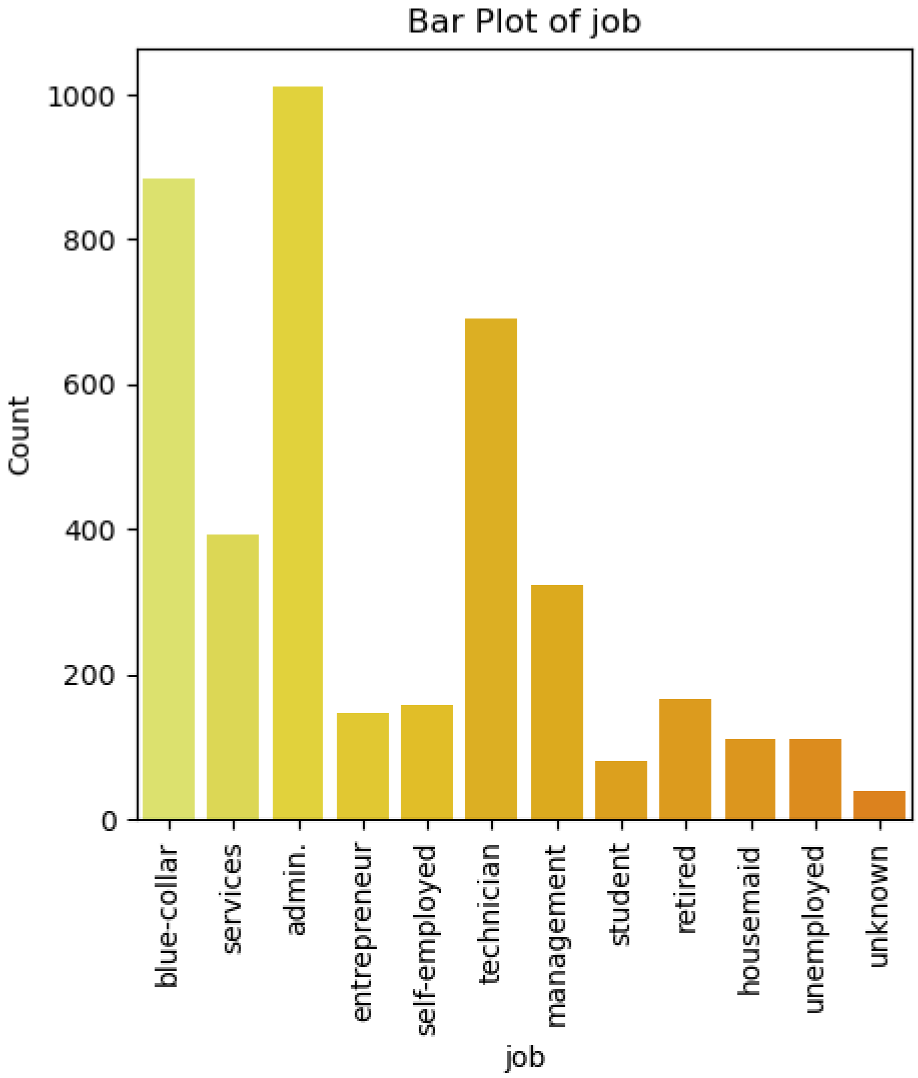
	job	marital	education	default	housing	loan	contact	month	day
count	4119	4119	4119	4119	4119	4119	4119	4119	
unique	12	4	8	3	3	3	2	10	
top	admin.	married	university.degree	no	yes	no	cellular	may	
freq	1012	2509	1264	3315	2175	3349	2652	1378	

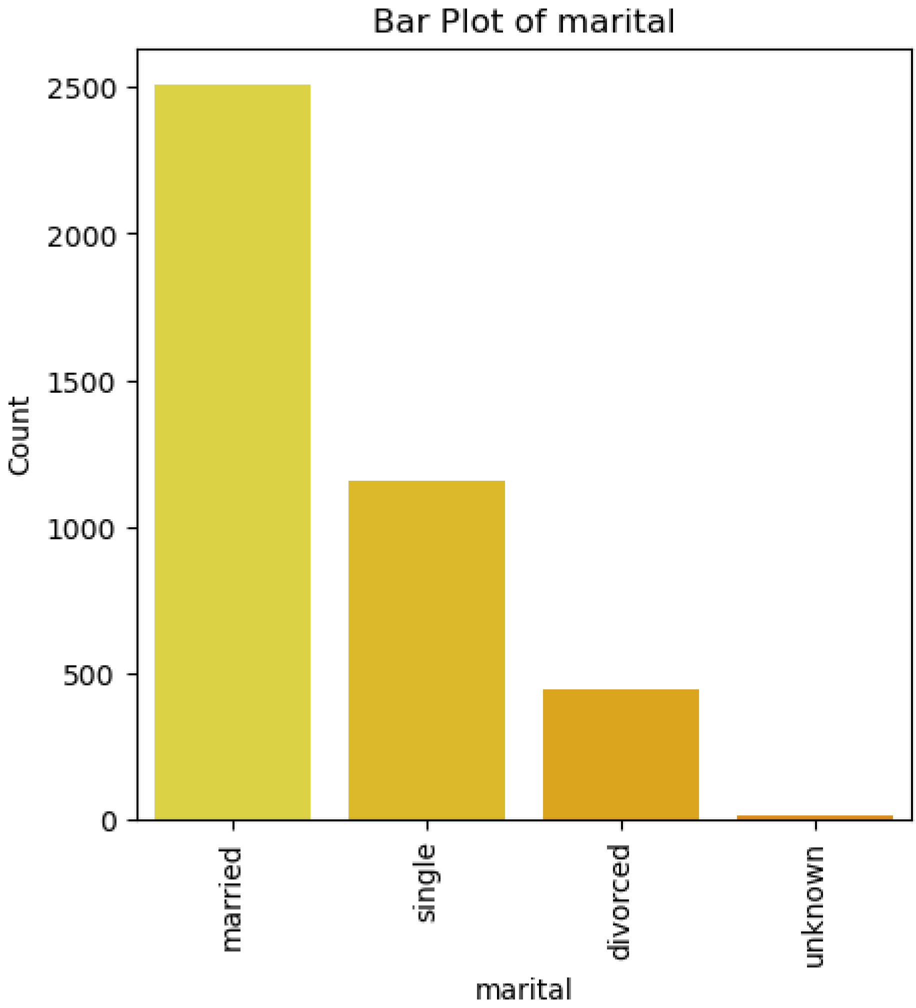
In [26]:
```
df.hist(figsize=(10,10),color='#cc5500')
plt.show()
```

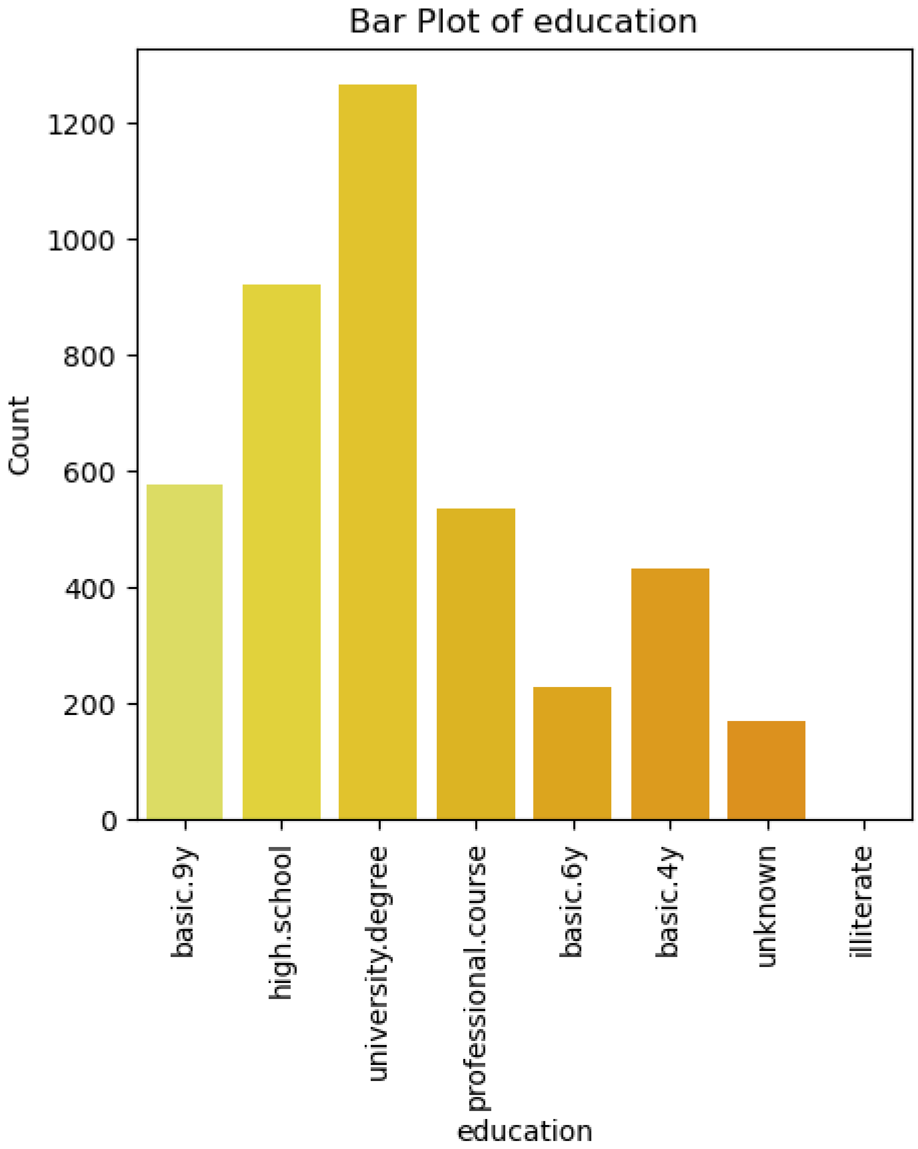


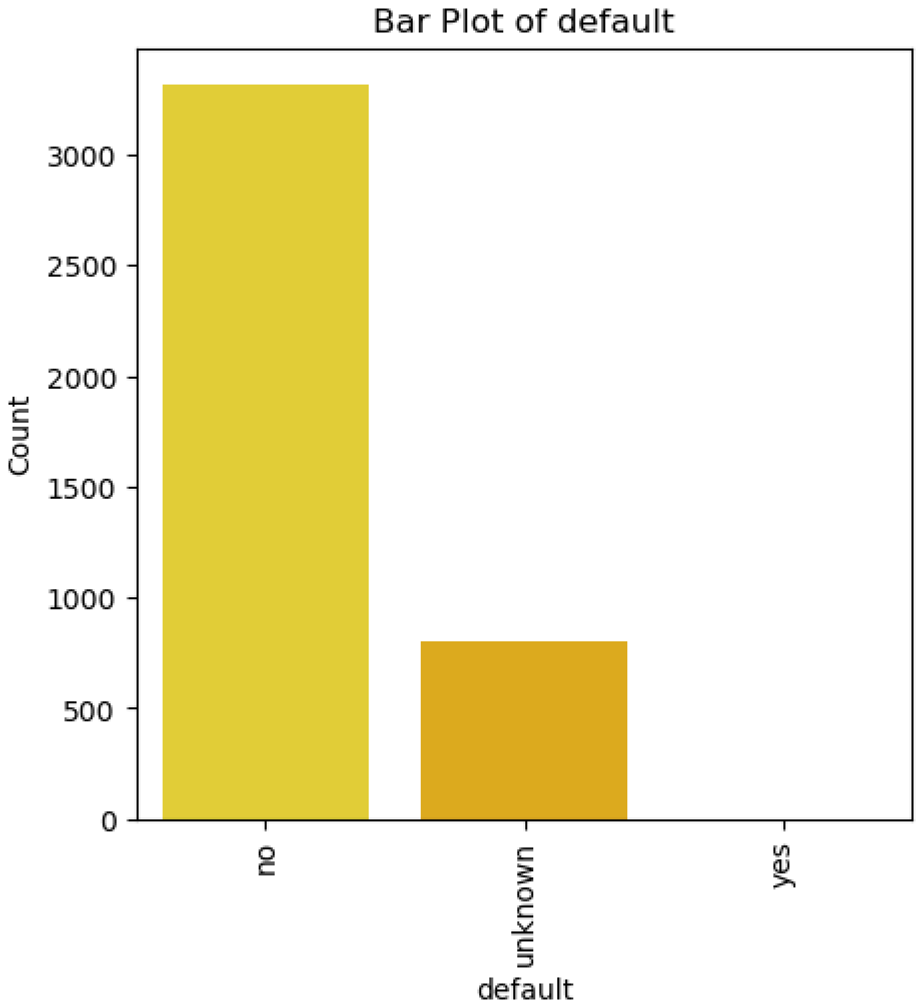
In [27]:

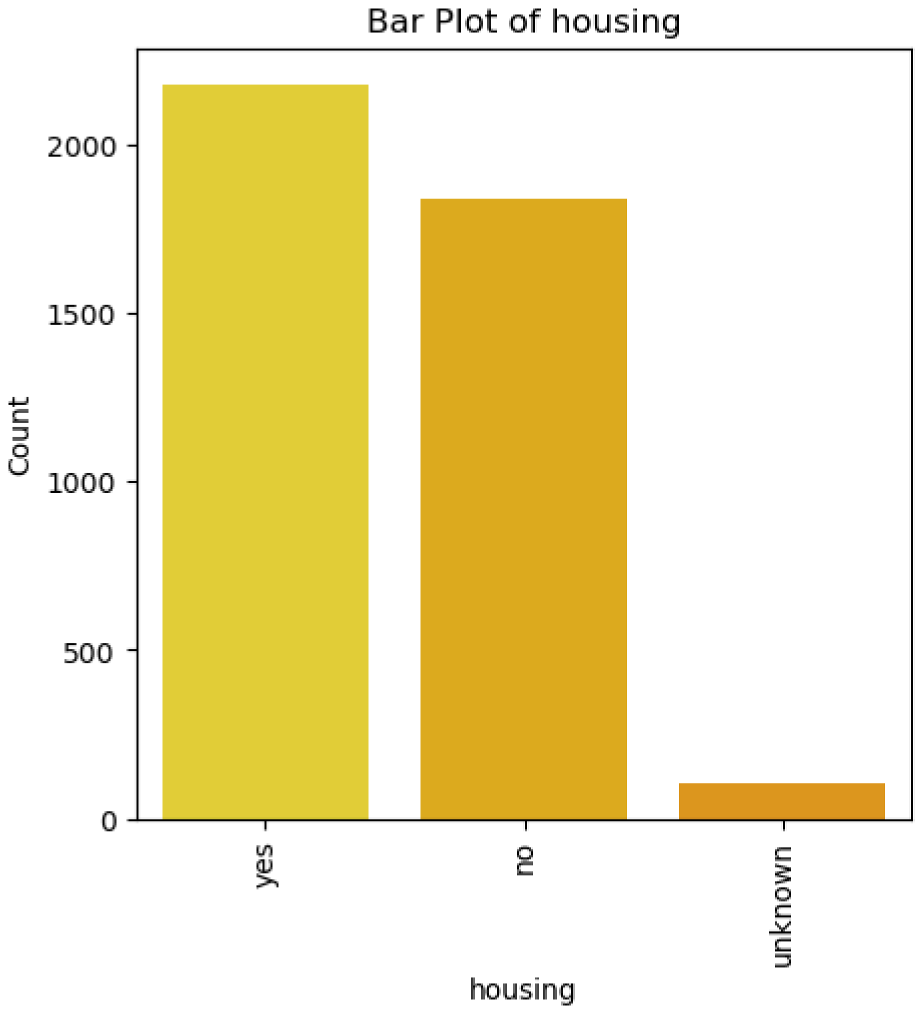
```
for feature in cat_cols:
    plt.figure(figsize=(5,5))  # Adjust the figure size as needed
    sns.countplot(x=feature, data=df, palette='Wistia')
    plt.title(f'Bar Plot of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=90)
    plt.show()
```

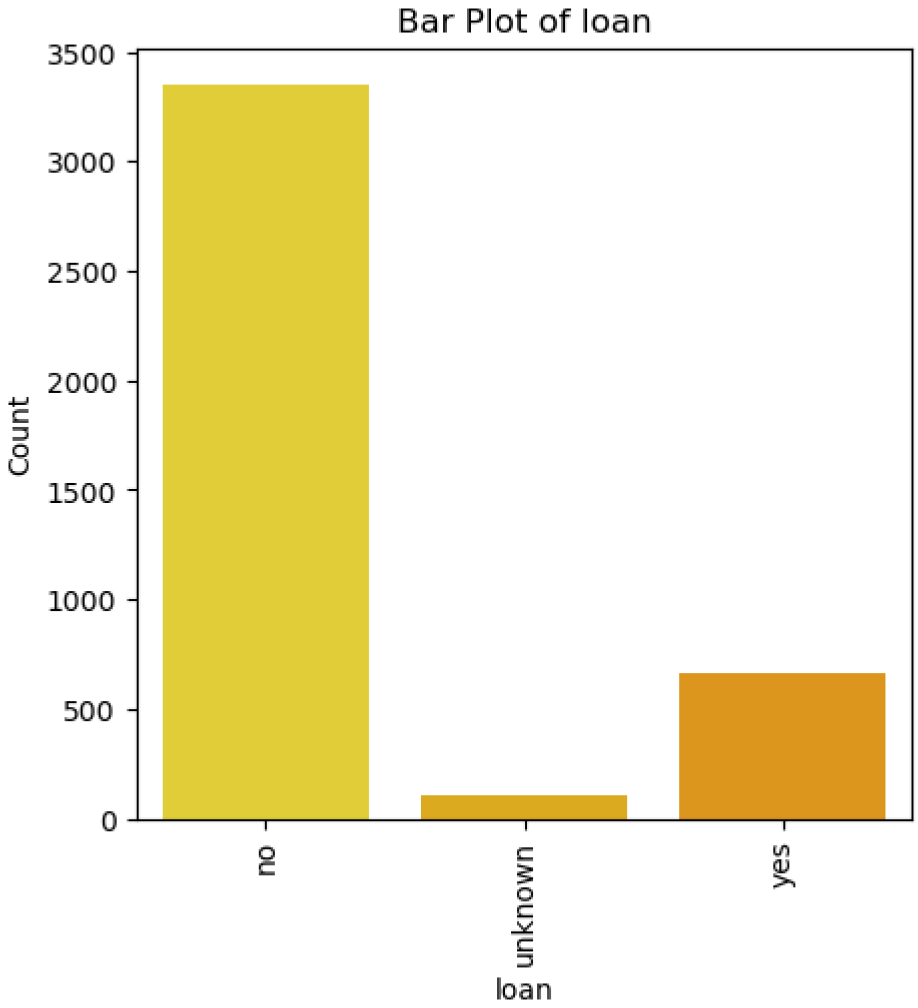

Bar Plot of job
Count
1000
800
600
400
200
0
blue-collar
services
admin.
entrepreneur
self-employed
technician
management
student
retired
housemaid
unemployed
unknown
job

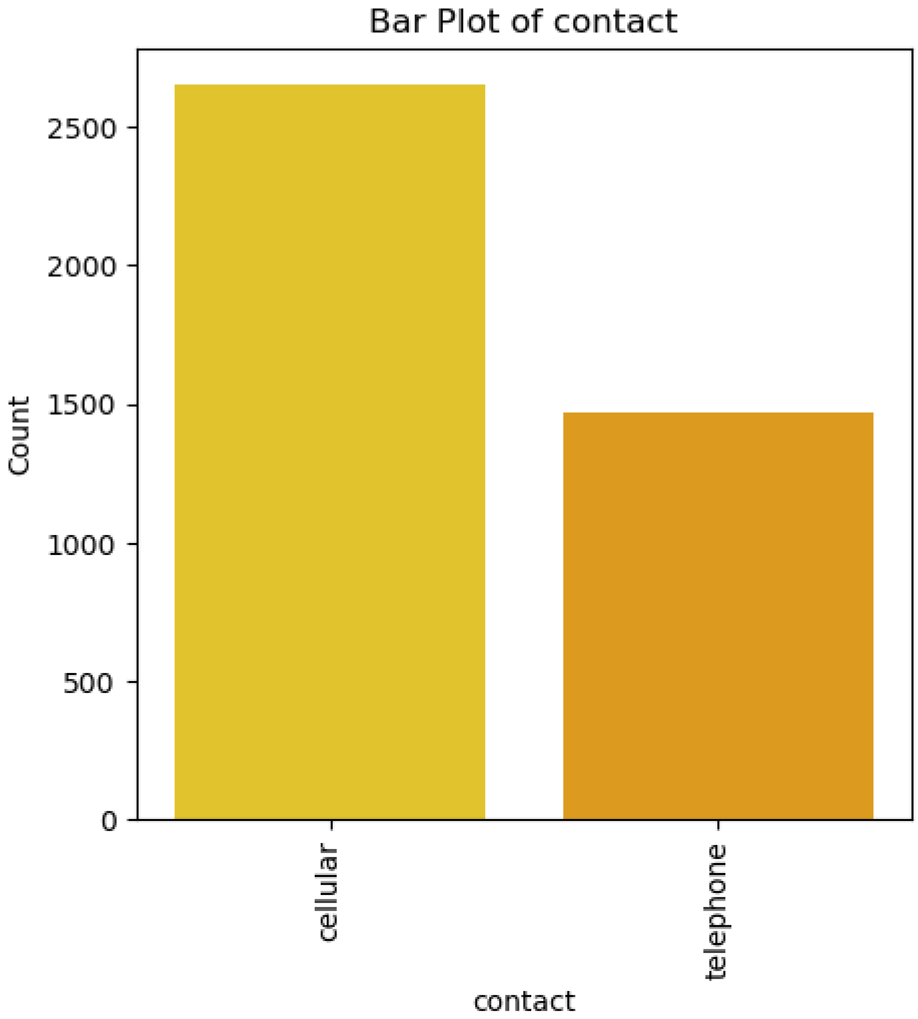

Bar Plot of marital
Count
2500
2000
1500
1000
500
0
married
single
divorced
unknown
marital

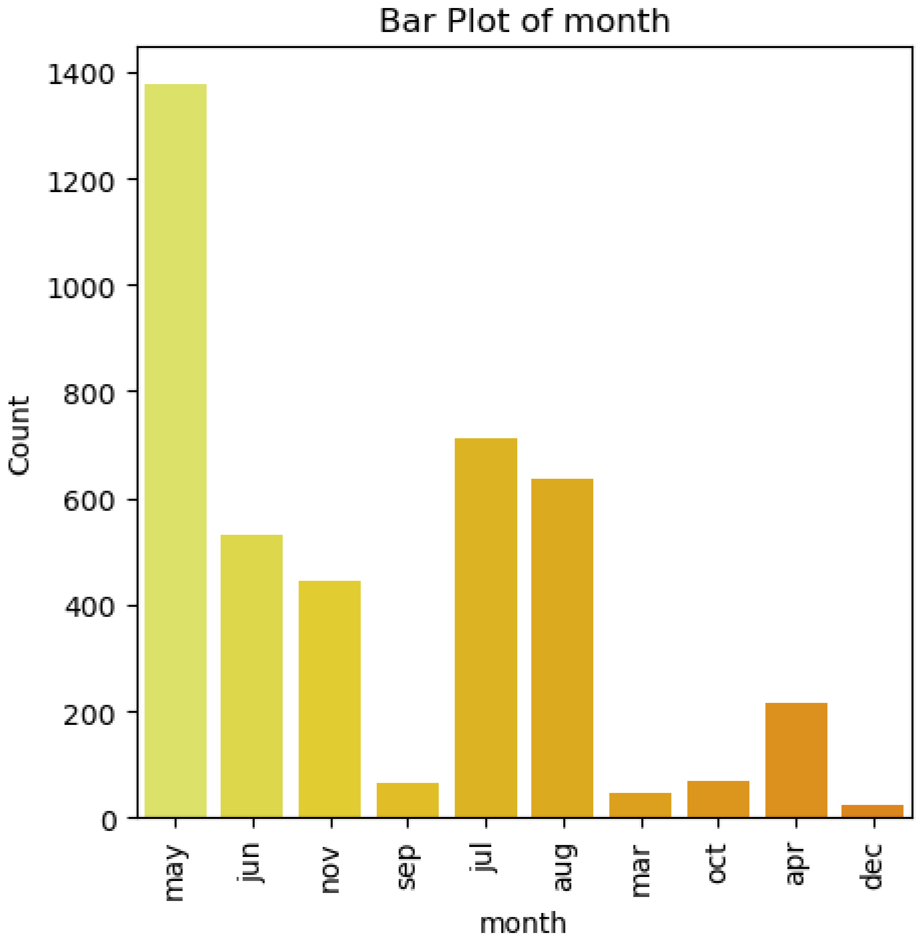

Bar Plot of education
Count
1200
1000
800
600
400
200
0
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate
education

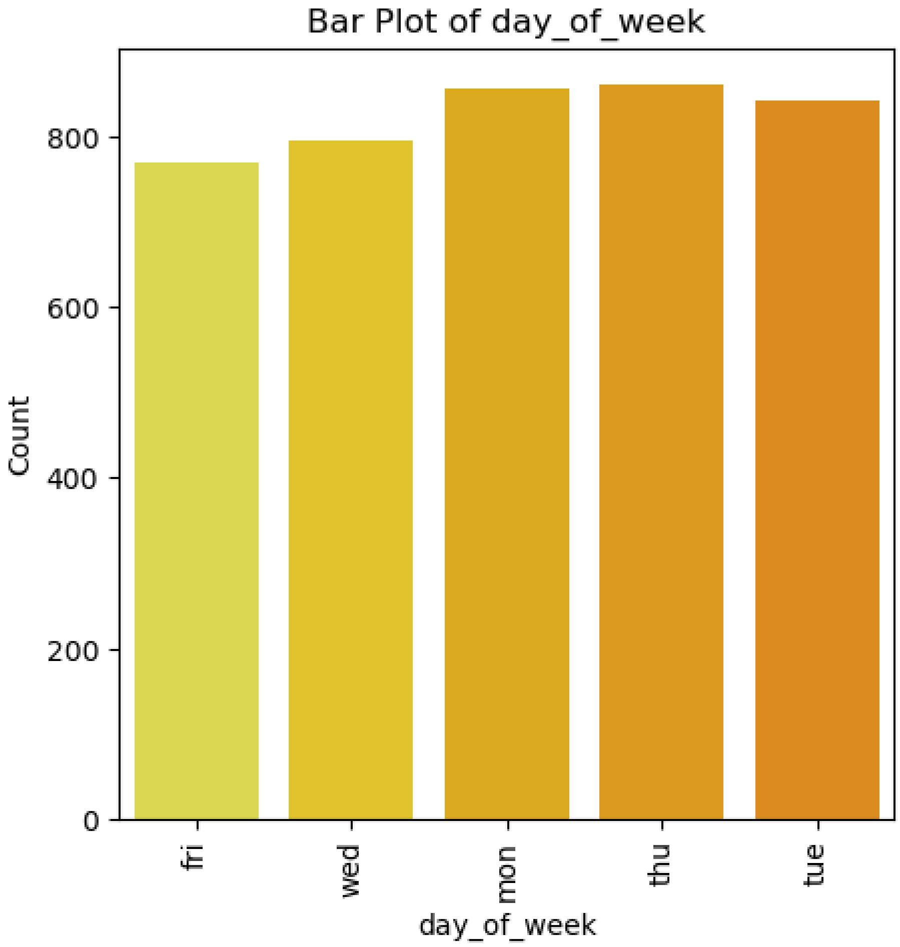

Bar Plot of default
Count
3000
2500
2000
1500
1000
500
0
no
unknown
yes
default

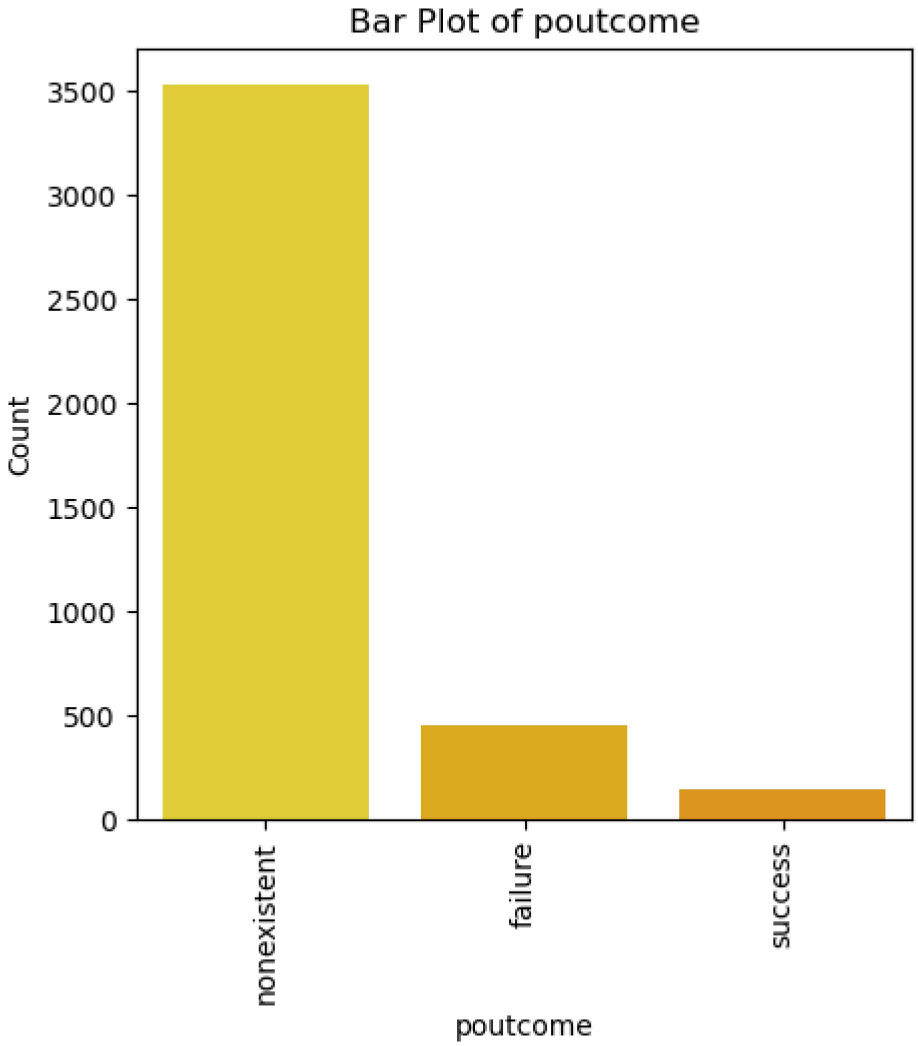

Bar Plot of housing
Count
0
500
1000
1500
2000
yes
no
unknown
housing

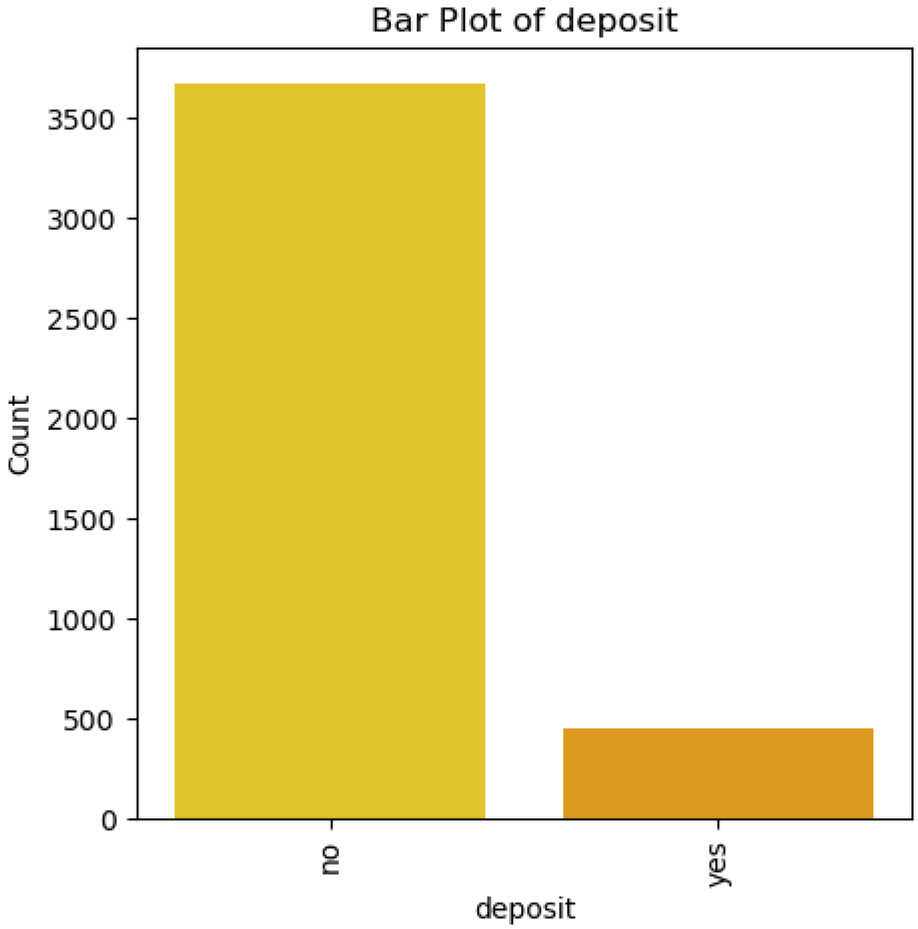

Bar Plot of loan
Count
3500
3000
2500
2000
1500
1000
500
0
no
unknown
yes
loan


Bar Plot of contact
Count
2500
2000
1500
1000
500
0
cellular
telephone
contact

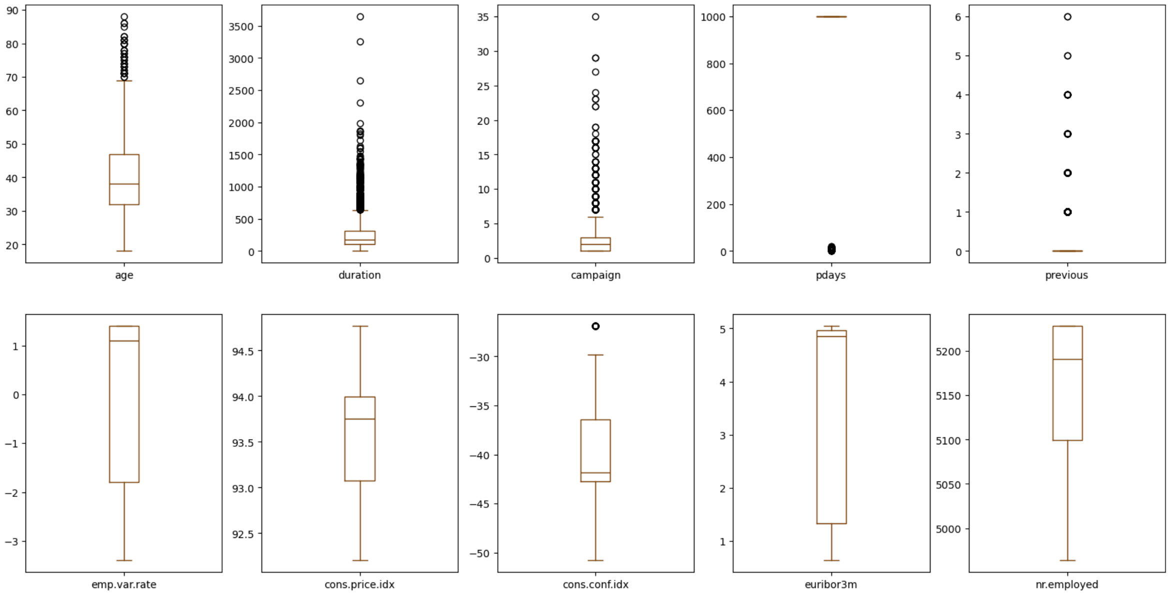

Bar Plot of month
Count
1400
1200
1000
800
600
400
200
0
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
month


Bar Plot of day_of_week
Count
0
200
400
600
800
fri
wed
mon
thu
tue
day_of_week


Bar Plot of poutcome
Count
3500
3000
2500
2000
1500
1000
500
0
nonexistent
failure
success
poutcome

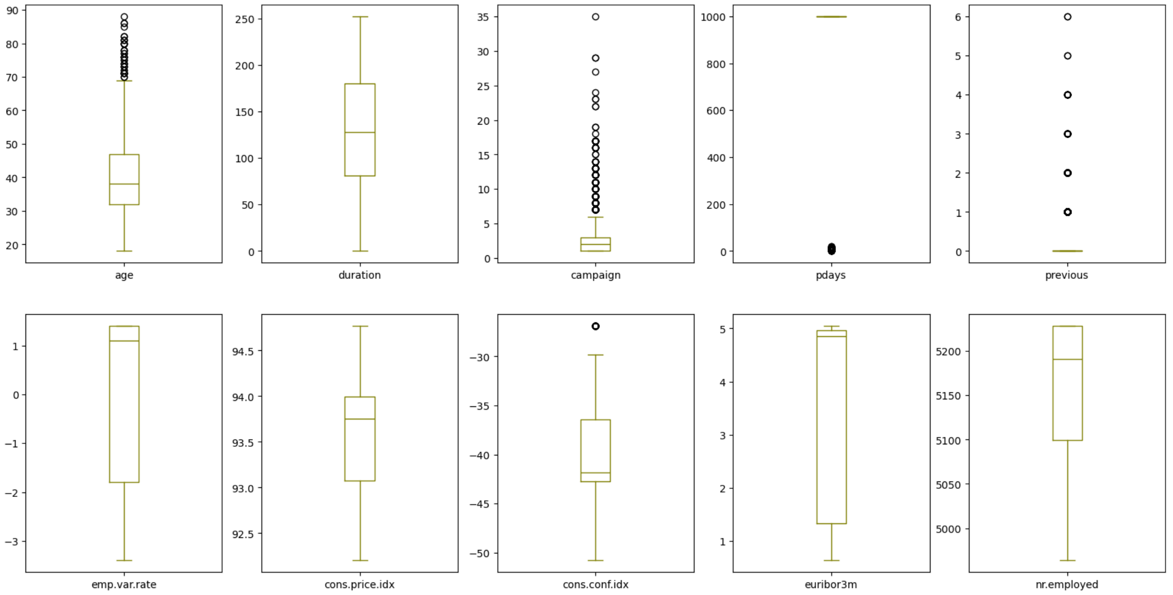

Bar Plot of deposit
Count
3500
3000
2500
2000
1500
1000
500
0
no
yes
deposit

In [28]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00')
plt.show()
```



In [29]:
```
column = df[['age','campaign','duration']]
q1 = np.percentile(column, 25)
q3 = np.percentile(column, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
df[['age','campaign','duration']] = column[(column > lower_bound) & (column < up
```

In [30]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#808000')
plt.show()
```



In [34]:
```
high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

In [35]:
```
df1 = df.copy()
df1.columns
```

Out[35]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

In [36]:
```
df1.drop(high_corr_cols,inplace=True,axis=1)  # axis=1 indicates columns
df1.columns
```

Out[36]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'cons.price.idx', 'cons.conf.idx', 'y'],
      dtype='object')
```

In [37]:
```
df1.shape
```

Out[37]:
```
(4119, 18)
```

In [38]:
```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

Out[38]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_we
0	12	1	1	2	0	2	0	0	6	
1	21	7	2	3	0	0	0	1	6	
2	7	7	1	3	0	2	0	1	4	
3	20	7	1	2	0	1	1	1	4	
4	29	0	1	6	0	2	0	0	7	
...	...	...	...	...	...	...	...	...	...	
4114	12	0	1	1	0	2	2	0	3	
4115	21	0	1	3	0	2	0	1	3	
4116	9	8	2	3	0	0	0	0	6	
4117	40	0	1	3	0	0	0	0	1	
4118	16	4	2	3	0	2	0	0	7	

4119 rows × 18 columns

In [40]:
```
df_encoded['y'].value_counts()
```

Out[40]:
```
y
0    3668
1     451
Name: count, dtype: int64
```

In [41]:
```
x = df_encoded.drop('y',axis=1)  # independent variable
y = df_encoded['y']              # dependent variable
print(x.shape)
print(y.shape)
print(type(x))
print(type(y))
```

```
(4119, 17)
(4119,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [42]:
```
from sklearn.model_selection import train_test_split

print(4119*0.25)
```

```
1029.75
```

In [43]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3089, 17)
(1030, 17)
(3089,)
(1030,)
```

In [44]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_scor

def eval_model(y_test,y_pred):
    acc = accuracy_score(y_test,y_pred)
    print('Accuracy_Score',acc)
    cm = confusion_matrix(y_test,y_pred)
    print('Confusion Matrix\n',cm)
    print('Classification Report\n',classification_report(y_test,y_pred))

def mscore(model):
    train_score = model.score(x_train,y_train)
    test_score = model.score(x_test,y_test)
    print('Training Score',train_score)
    print('Testing Score',test_score)
```

In [45]:
```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion='gini',max_depth=5,min_samples_split=10)
dt.fit(x_train,y_train)
```

Out[45]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

In [46]:
```
mscore(dt)
```

```
Training Score 0.923276141146002
Testing Score 0.9116504854368932
```

In [47]:
```
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

```
[0 0 1 ... 0 0 0]
```

In [48]:
```
eval_model(y_test,ypred_dt)
```

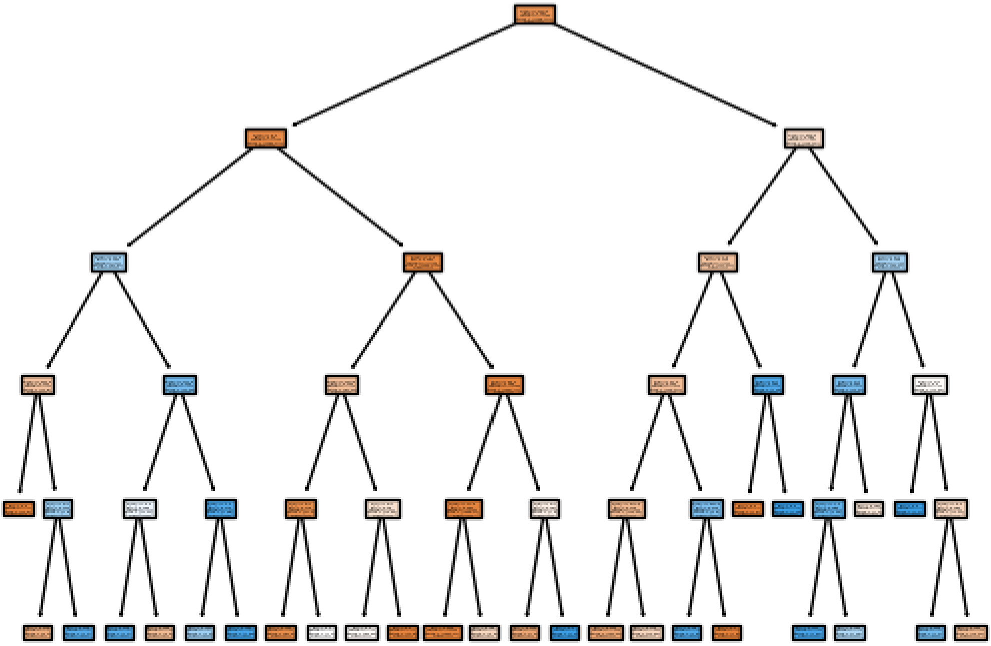
```
Accuracy_Score 0.9116504854368932
Confusion Matrix
 [[913  17]
 [ 74  26]]
Classification Report
               precision    recall  f1-score   support

           0       0.93      0.98      0.95       930
           1       0.60      0.26      0.36       100

    accuracy                           0.91      1030
   macro avg       0.76      0.62      0.66      1030
weighted avg       0.89      0.91      0.90      1030
```

In [49]:
```
from sklearn.tree import plot_tree
```

In [50]:
```
cn = ['no','yes']
fn = x_train.columns
print(fn)
print(cn)
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'cons.price.idx', 'cons.conf.idx'],
      dtype='object')
['no', 'yes']
```

In [51]:
```
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



In [52]:
```
dt1 = DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=1
dt1.fit(x_train,y_train)
```

Out[52]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_sp
lit=15)
```

In [53]:

```
mscore(dt1)
```

```
Training Score 0.9145354483651668
Testing Score 0.916504854368932
```

In [54]:

```
ypred_dt1 = dt1.predict(x_test)
```

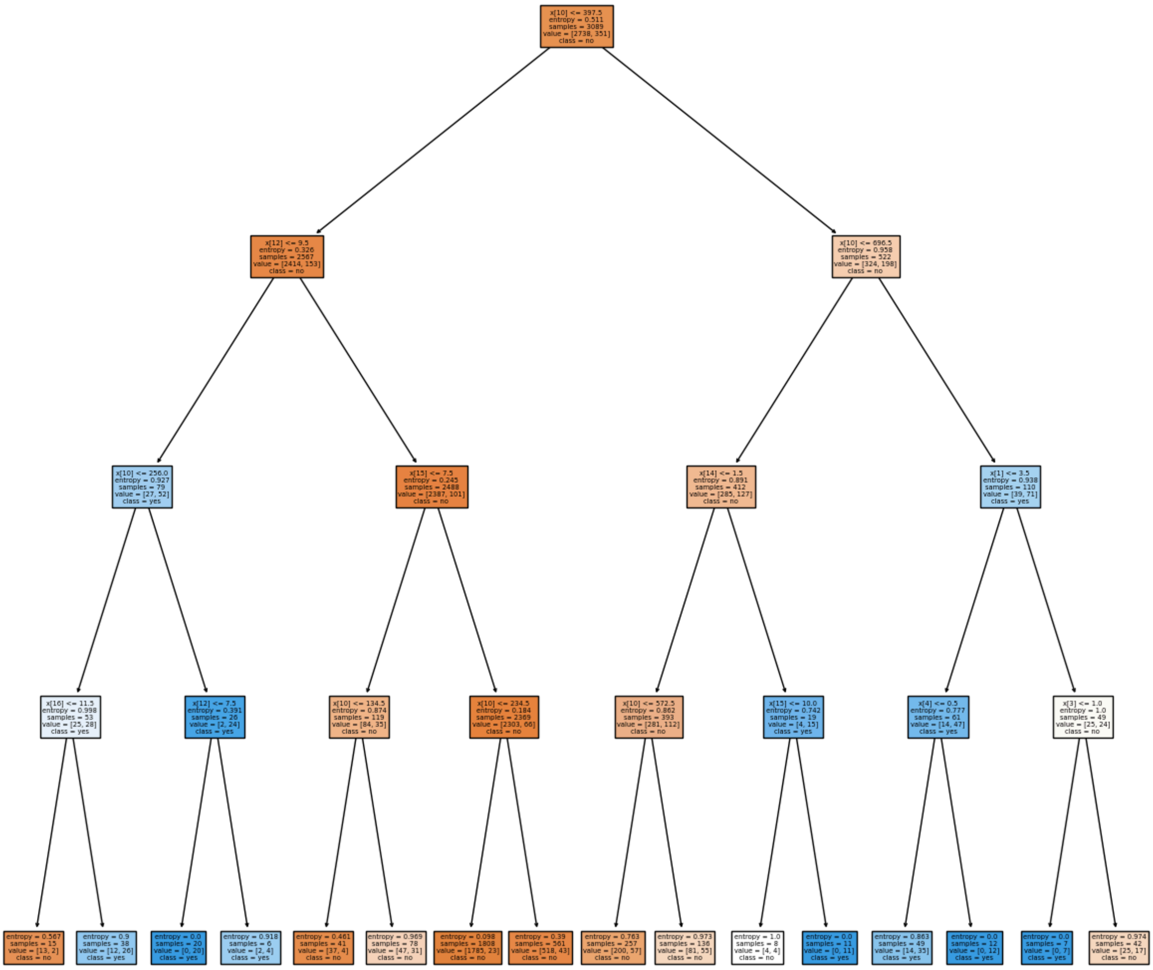
In [55]:

```
eval_model(y_test,ypred_dt1)
```

```
Accuracy_Score 0.916504854368932
Confusion Matrix
 [[912  18]
 [ 68  32]]
Classification Report
               precision    recall  f1-score   support

           0       0.93      0.98      0.95       930
           1       0.64      0.32      0.43       100

    accuracy                           0.92      1030
   macro avg       0.79      0.65      0.69      1030
weighted avg       0.90      0.92      0.90      1030
```

In [56]:

```
plt.figure(figsize=(15,15))
plot_tree(dt1,class_names=cn,filled=True)
plt.show()
```


x[10] <= 397.5 entropy = 0.511 samples = 3089 value = [2738, 351] class = no
x[12] <= 9.5 entropy = 0.326 samples = 2567 value = [2414, 153] class = no
x[10] <= 696.5 entropy = 0.958 samples = 522 value = [324, 198] class = no
x[10] <= 256.0 entropy = 0.927 samples = 79 value = [27, 52] class = yes
x[15] <= 7.5 entropy = 0.245 samples = 2488 value = [2387, 101] class = no
x[14] <= 1.5 entropy = 0.891 samples = 412 value = [285, 127] class = no
x[1] <= 3.5 entropy = 0.938 samples = 110 value = [39, 71] class = yes
x[16] <= 11.5 entropy = 0.998 samples = 53 value = [25, 28] class = yes
x[12] <= 7.5 entropy = 0.391 samples = 26 value = [2, 24] class = yes
x[10] <= 134.5 entropy = 0.874 samples = 119 value = [84, 35] class = no
x[10] <= 234.5 entropy = 0.184 samples = 2369 value = [2303, 66] class = no
x[10] <= 572.5 entropy = 0.862 samples = 393 value = [281, 112] class = no
x[15] <= 10.0 entropy = 0.742 samples = 19 value = [4, 15] class = yes
x[4] <= 0.5 entropy = 0.777 samples = 61 value = [14, 47] class = yes
x[3] <= 1.0 entropy = 1.0 samples = 49 value = [25, 24] class = no
entropy = 0.567 samples = 15 value = [13, 2] class = no
entropy = 0.9 samples = 38 value = [12, 26] class = yes
entropy = 0.0 samples = 20 value = [0, 20] class = yes
entropy = 0.918 samples = 6 value = [2, 4] class = yes
entropy = 0.461 samples = 41 value = [37, 4] class = no
entropy = 0.969 samples = 78 value = [47, 31] class = no
entropy = 0.098 samples = 1808 value = [1785, 23] class = no
entropy = 0.39 samples = 561 value = [518, 43] class = no
entropy = 0.763 samples = 257 value = [200, 57] class = no
entropy = 0.973 samples = 136 value = [81, 55] class = no
entropy = 1.0 samples = 8 value = [4, 4] class = no
entropy = 0.0 samples = 11 value = [0, 11] class = yes
entropy = 0.863 samples = 49 value = [14, 35] class = yes
entropy = 0.0 samples = 12 value = [0, 12] class = yes
entropy = 0.0 samples = 7 value = [0, 7] class = yes
entropy = 0.974 samples = 42 value = [25, 17] class = no