In [3]:

```
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 [4]:

```
df = pd.read_csv('C:\\Users\\jeeva\\OneDrive\\Documents\\bank-additional.csv',delimiter=';')
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_weel
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fr
1	39	services	single	high.school	no	no	no	telephone	may	fr
2	25	services	married	high.school	no	yes	no	telephone	jun	we
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fr
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mo

5 rows × 21 columns

In [5]:

```
df.tail()
```

Out[5]:

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

5 rows × 21 columns

In [6]:

```
df.shape
```

Out[6]:

```
(4119, 21)
```

In [7]:

```
df.columns
```

Out[7]:

```
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 [8]:

```
df.dtypes
```

```
Out[8]: 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 [9]: df.dtypes.value_counts()
```

```
Out[9]: object     11
        int64       5
        float64     5
        Name: count, dtype: int64
```

```
In [10]: 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 [11]: df.duplicated().sum()
```

```
Out[11]: 0
```

In [12]:
```
df.isna().sum()
```

Out[12]:
```
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 [13]:
```
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 [14]:
```
df.describe()
```

Out[14]:

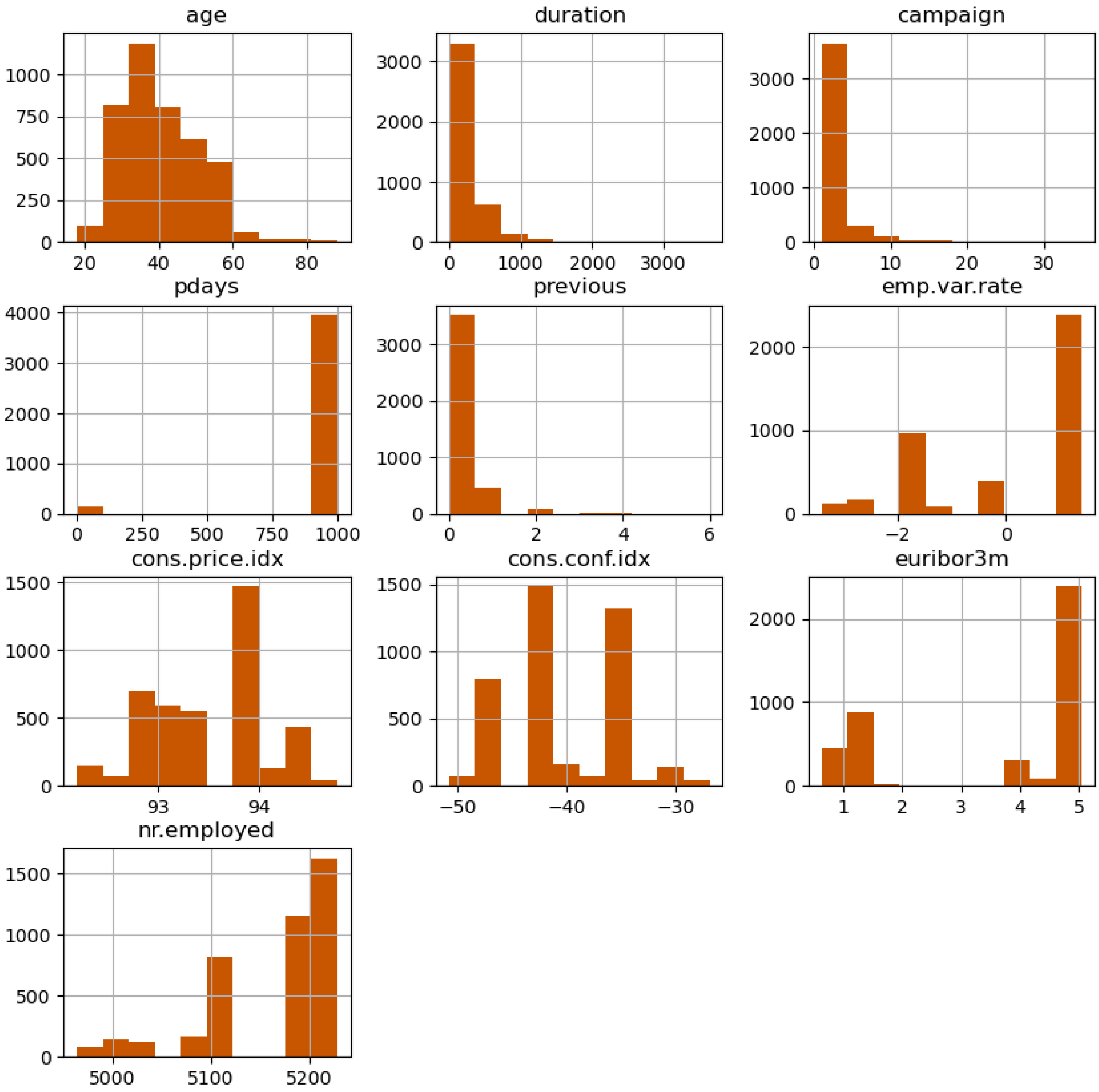
	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000

In [15]:
```
df.describe(include='object')
```

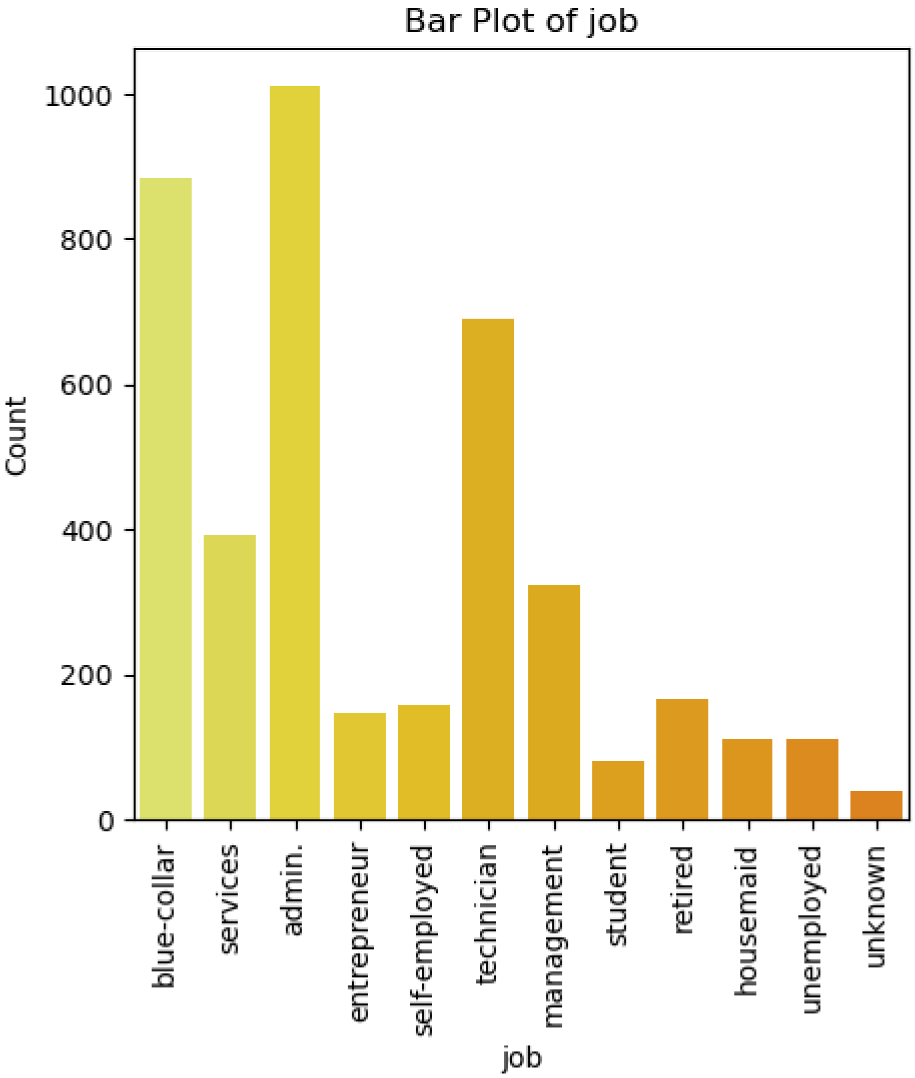
Out[15]:

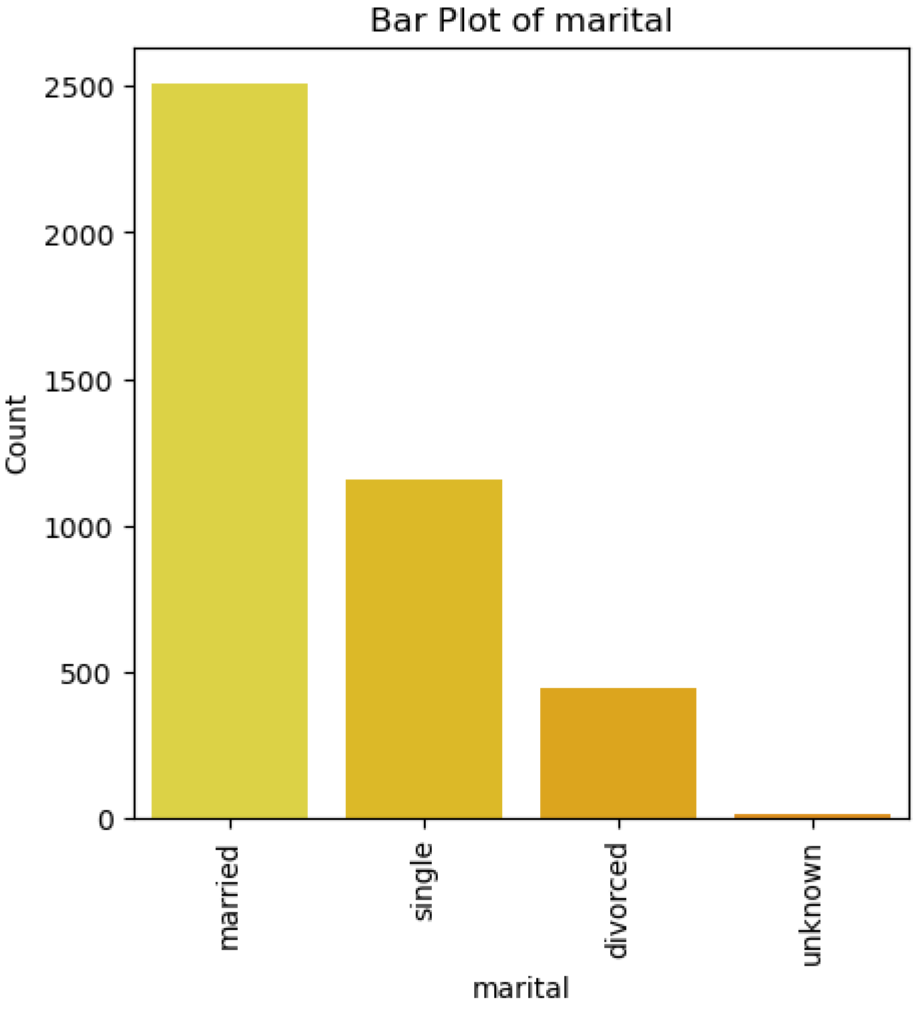
	job	marital	education	default	housing	loan	contact	month	day_of_week	pou
count	4119	4119	4119	4119	4119	4119	4119	4119	4119	
unique	12	4	8	3	3	3	2	10	5	
top	admin.	married	university.degree	no	yes	no	cellular	may	thu	none
freq	1012	2509	1264	3315	2175	3349	2652	1378	860	

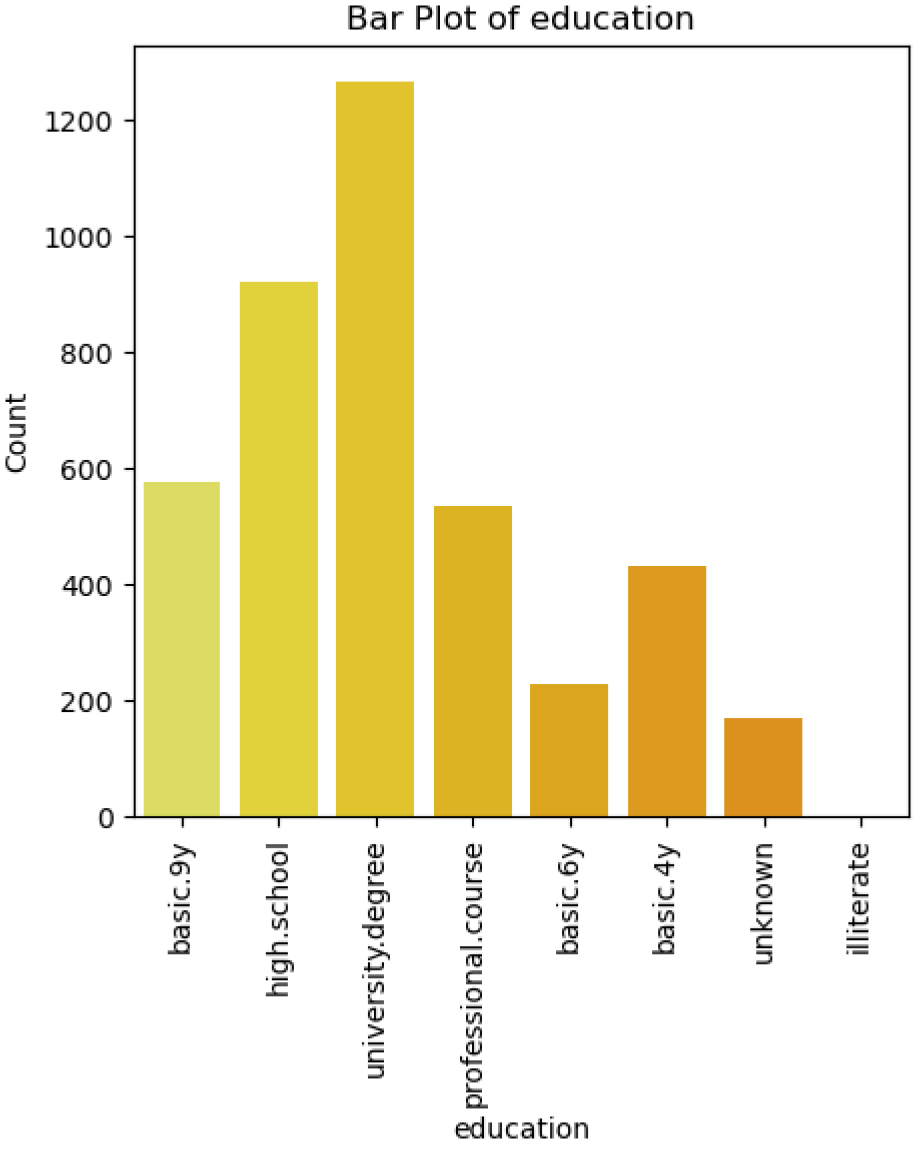
In [16]:
```
df.hist(figsize=(10,10),color='#cc5500')
plt.show()
```

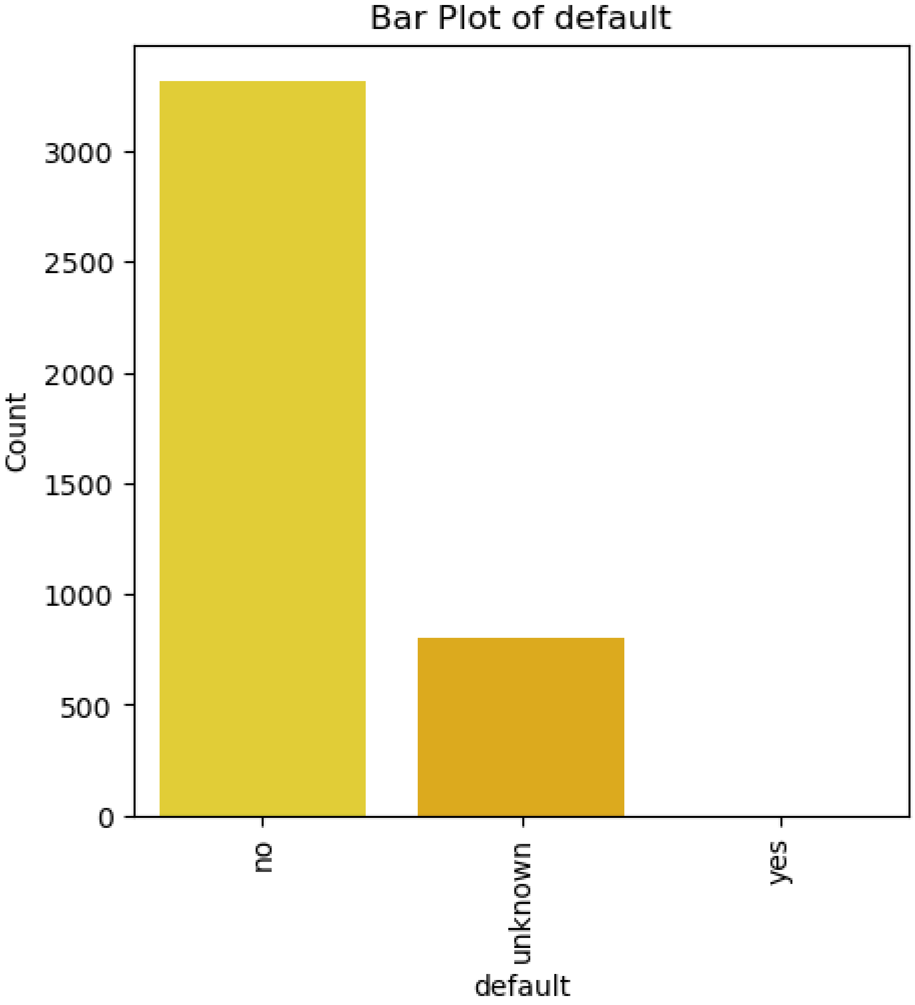


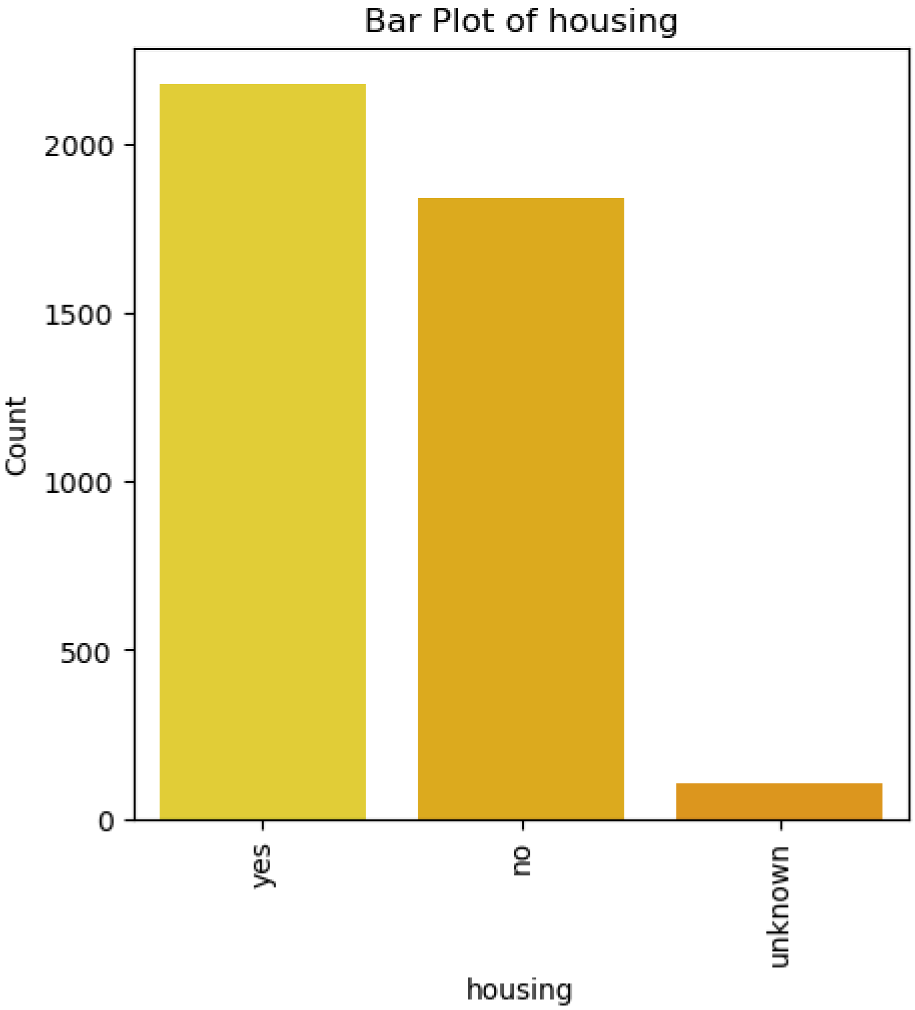
In [17]:
```
for feature in cat_cols:
    plt.figure(figsize=(5,5))  # Adjust the figure size as needed
    sns.countplot(x=feature, data=df, palette='Wistia')
    plt.title(f'Bar Plot of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=90)
    plt.show()
```

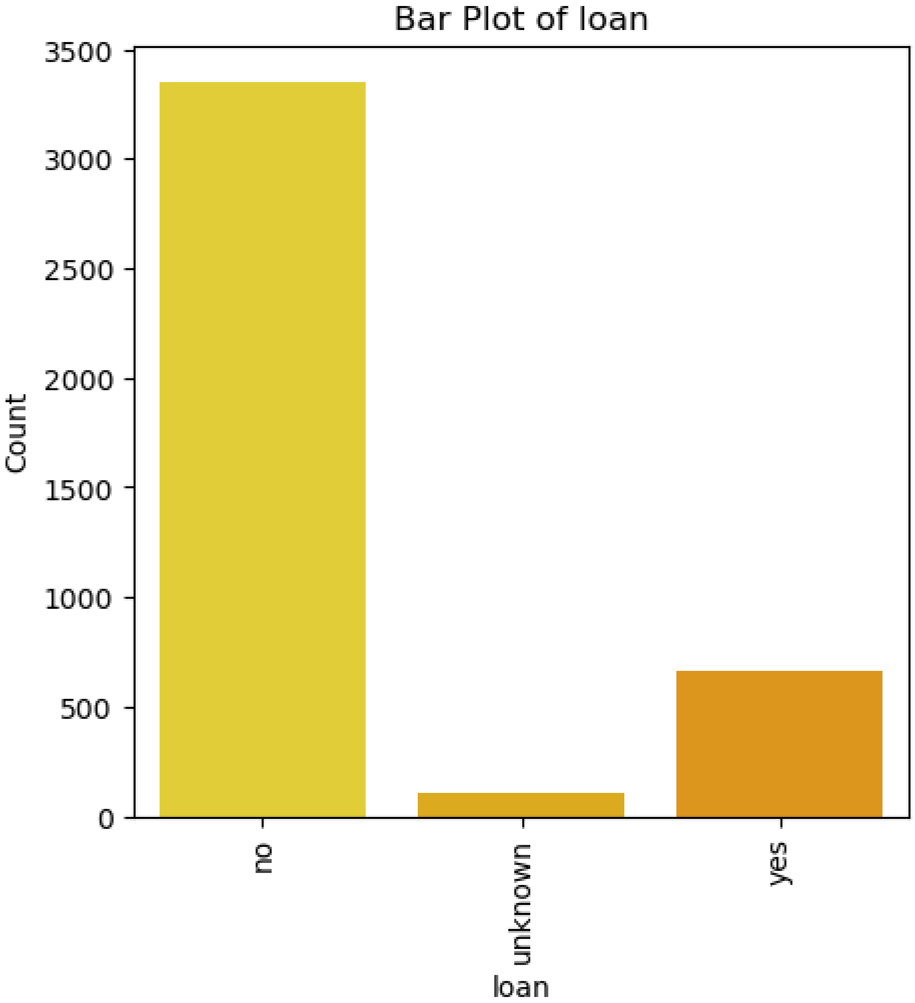

Bar Plot of job
Count
0
200
400
600
800
1000
blue-collar
services
admin.
entrepreneur
self-employed
technician
management
student
retired
housemaid
unemployed
unknown
job

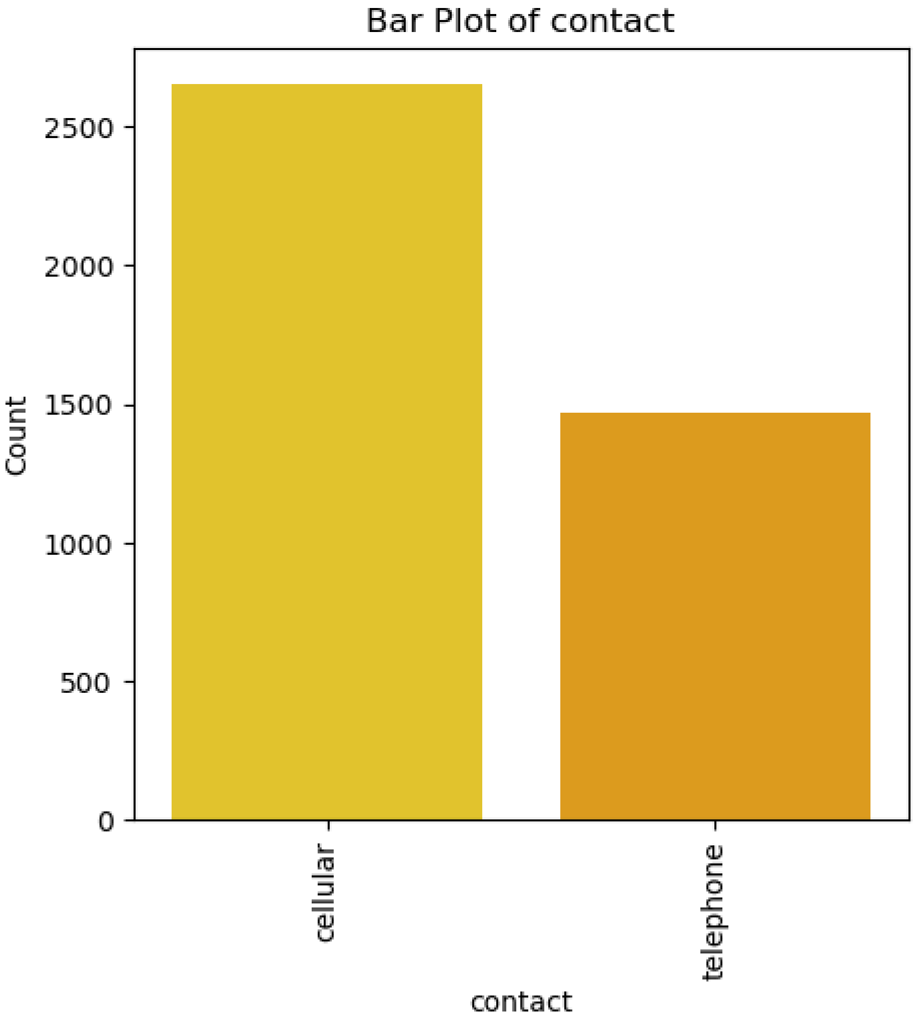

Bar Plot of marital
Count
0
500
1000
1500
2000
2500
married
single
divorced
unknown
marital

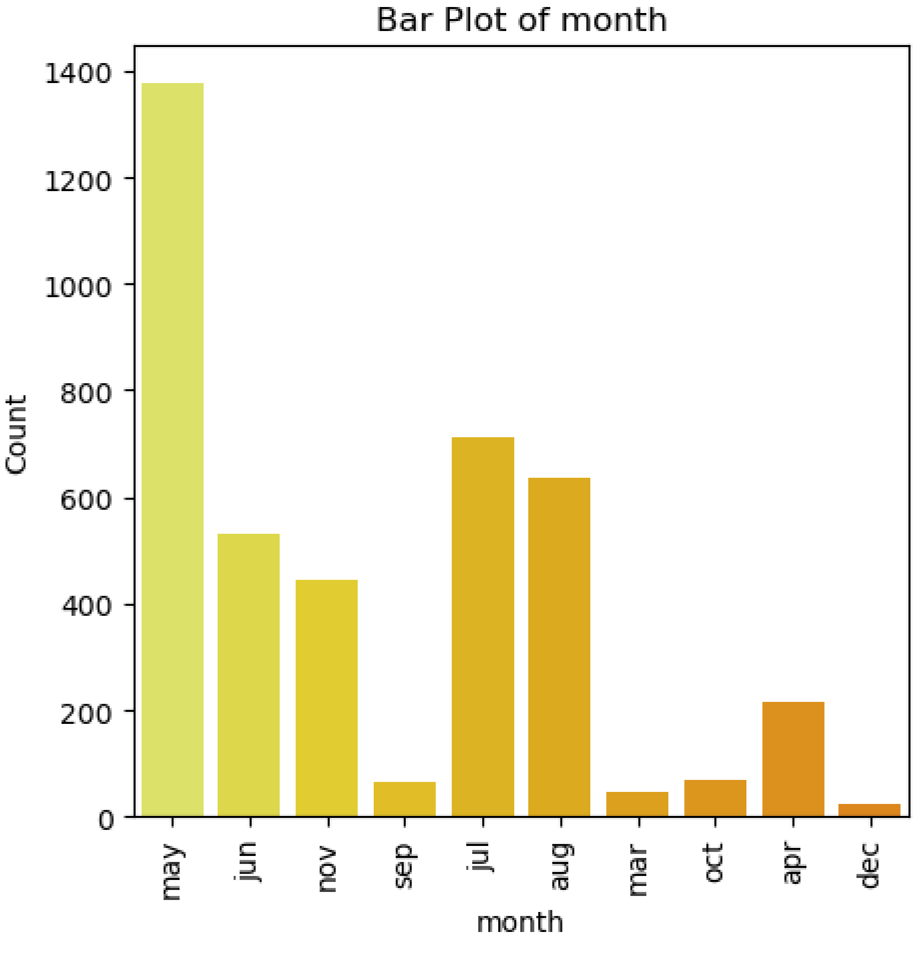

Bar Plot of education
Count
1200
1000
800
600
400
200
0
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate
education

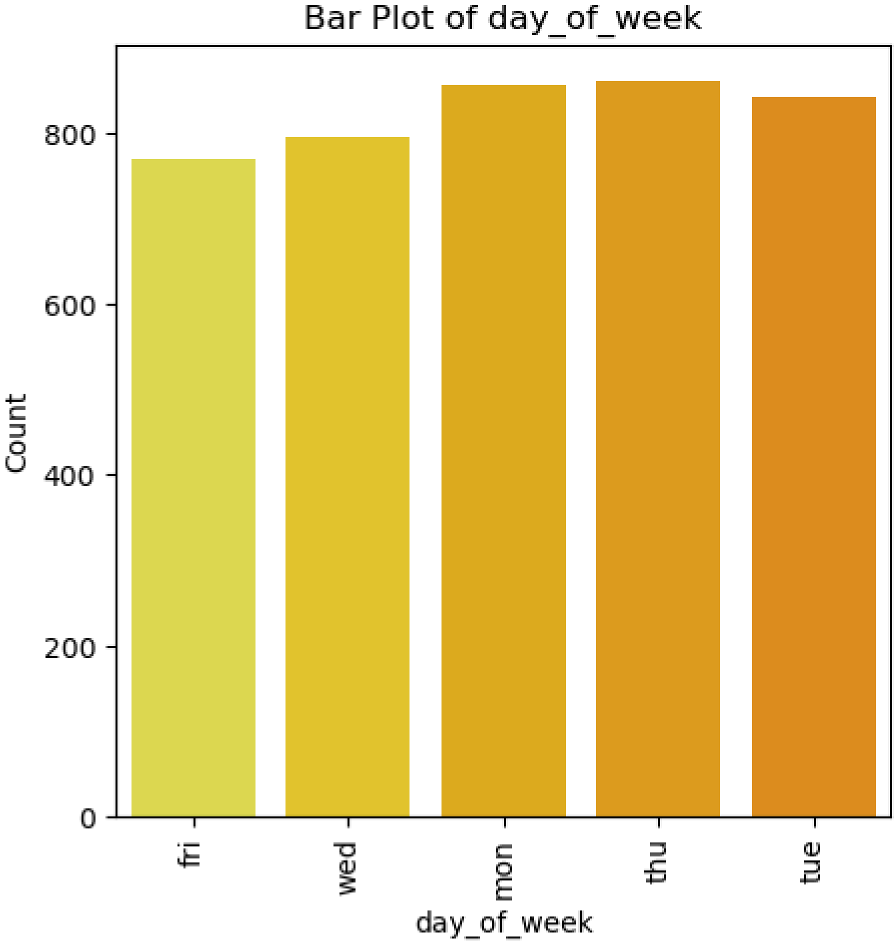

Bar Plot of default
Count
3000
2500
2000
1500
1000
500
0
no
unknown
yes
default

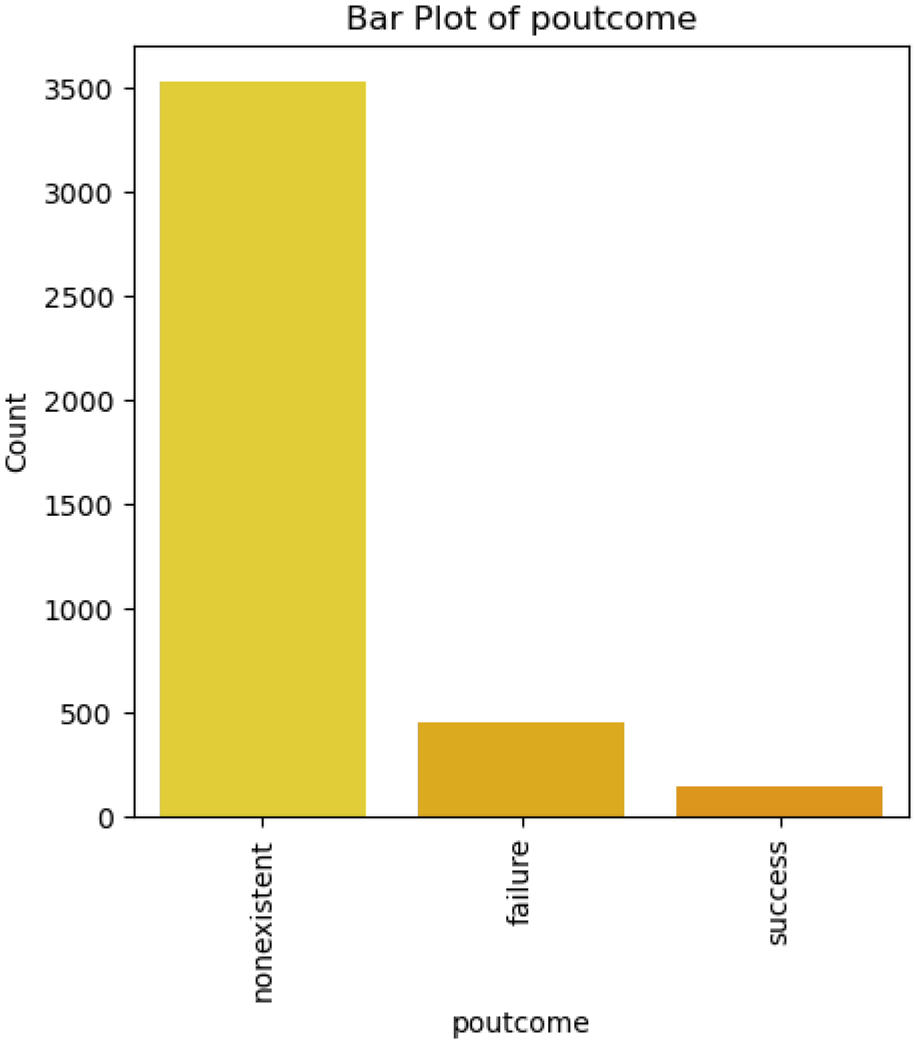

Bar Plot of housing
2000
1500
1000
500
0
Count
yes
no
unknown
housing

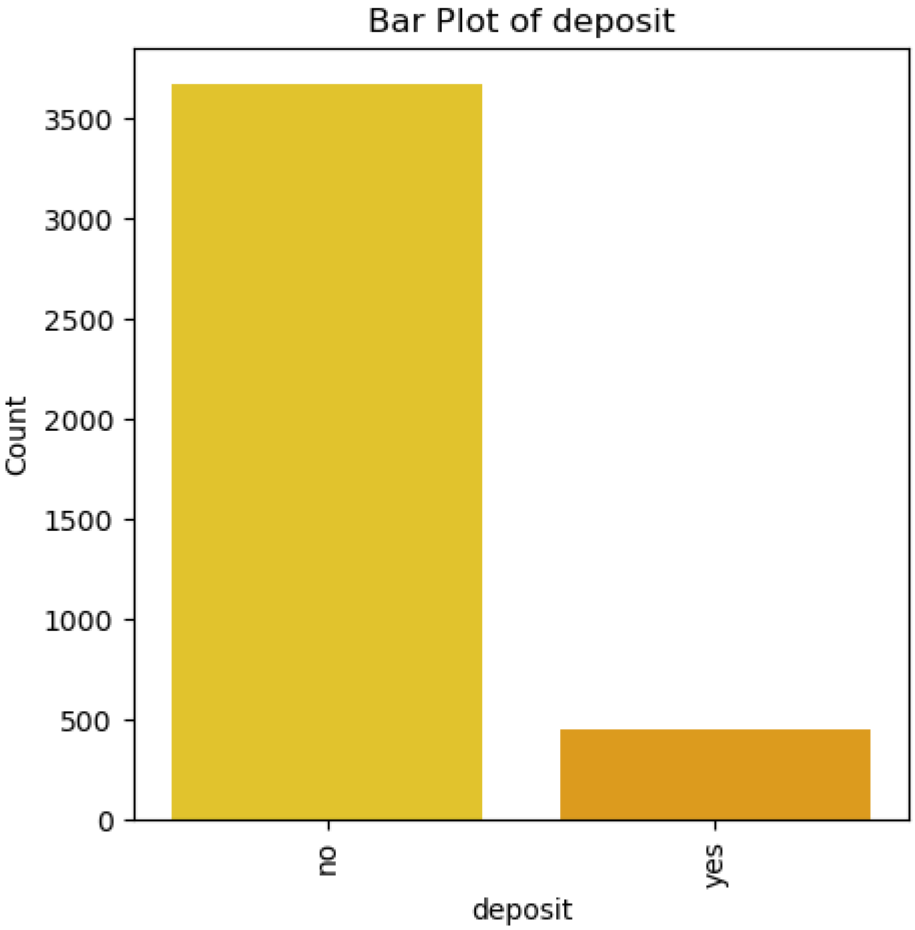

Bar Plot of loan
3500
3000
2500
2000
1500
1000
500
0
Count
no
unknown
yes
loan


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

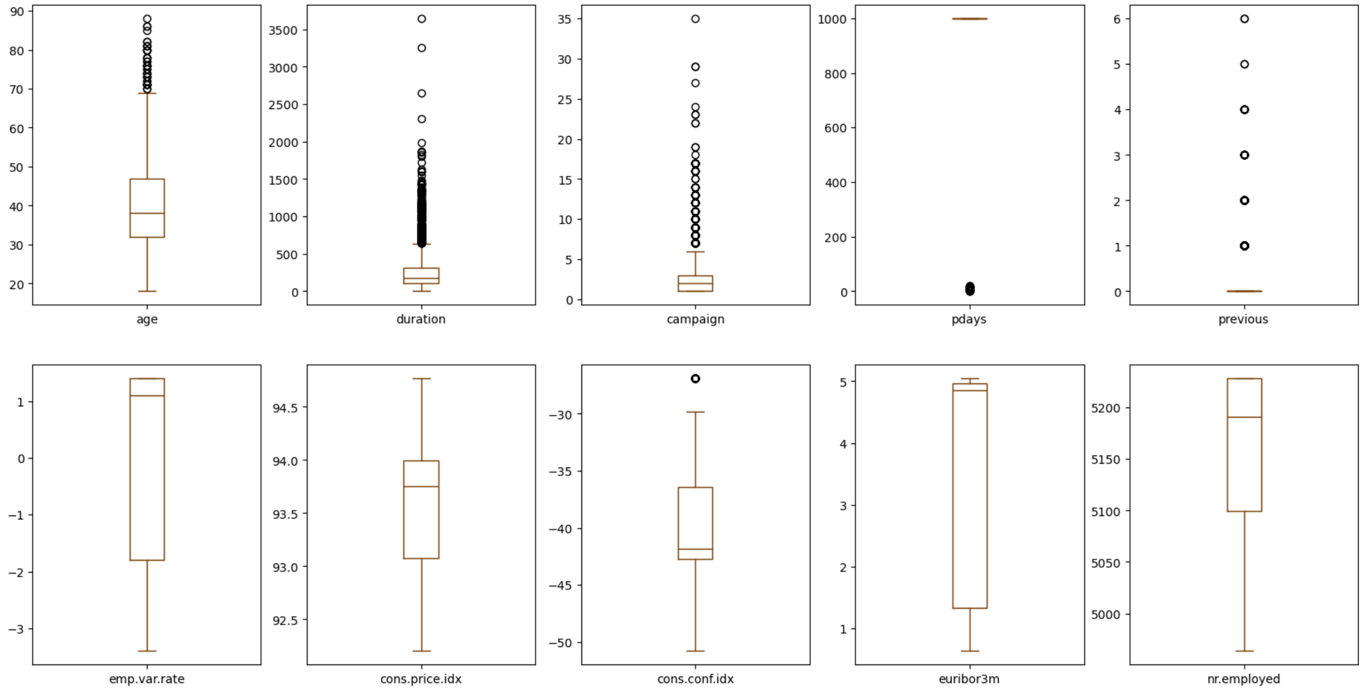

Bar Plot of month
Count
1400
1200
1000
800
600
400
200
0
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
month


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
0
500
1000
1500
2000
2500
3000
3500
nonexistent
failure
success
poutcome

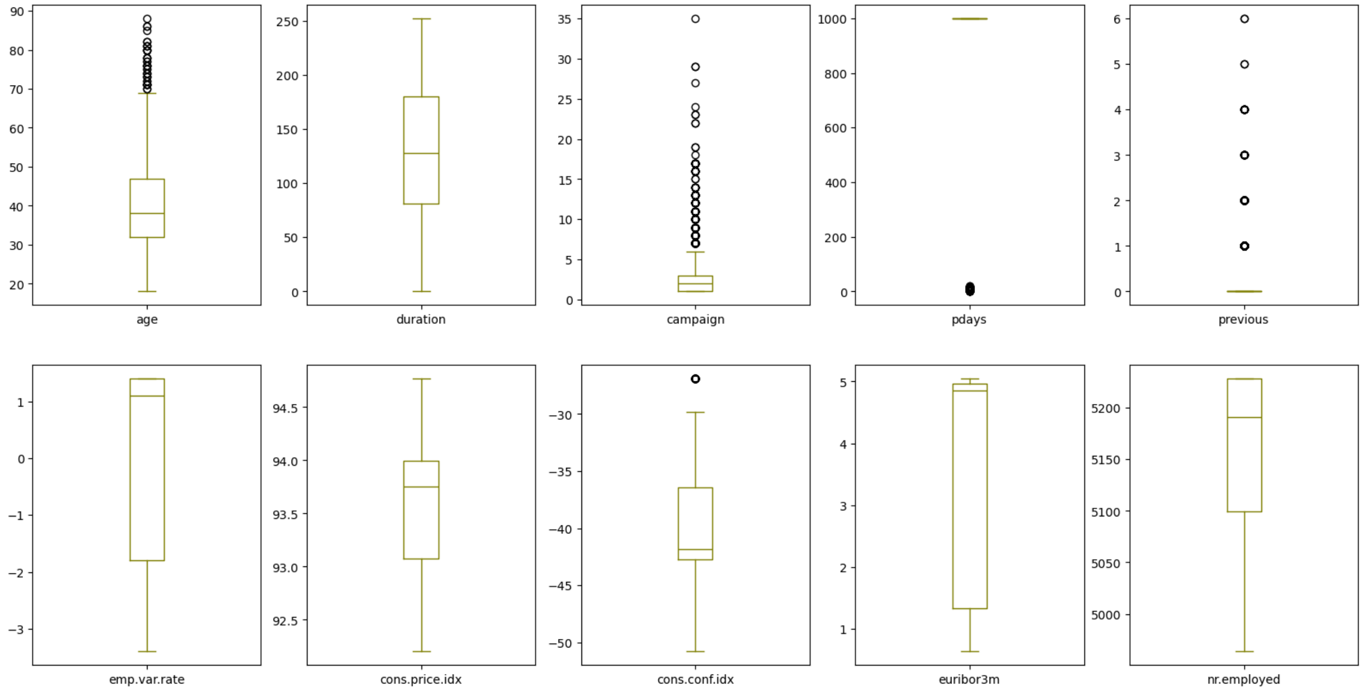


In [18]:
```python
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00')
plt.show()
```



In [19]:
```python
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 < upper_bound)]
```

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



```
In [22]: high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

```
In [23]: df1 = df.copy()
         df1.columns
```

```
Out[23]: 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 [24]: df1.drop(high_corr_cols,inplace=True,axis=1)  # axis=1 indicates columns
         df1.columns
```

```
Out[24]: Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
                'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
                'previous', 'poutcome', 'cons.price.idx', 'cons.conf.idx', 'deposit'],
               dtype='object')
```

```
In [25]: df1.shape
```

```
Out[25]: (4119, 18)
```

```
In [26]: from sklearn.preprocessing import LabelEncoder
         lb = LabelEncoder()
         df_encoded = df1.apply(lb.fit_transform)
         df_encoded
```

Out[26]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration
0	12	1	1	2	0	2	0	0	6	0	250
1	21	7	2	3	0	0	0	1	6	0	250
2	7	7	1	3	0	2	0	1	4	4	224
3	20	7	1	2	0	1	1	1	4	0	14
4	29	0	1	6	0	2	0	0	7	1	55
...	...	...	...	...	...	...	...	...	...	...	...
4114	12	0	1	1	0	2	2	0	3	2	50
4115	21	0	1	3	0	2	0	1	3	0	216
4116	9	8	2	3	0	0	0	0	6	1	61
4117	40	0	1	3	0	0	0	0	1	0	250
4118	16	4	2	3	0	2	0	0	7	4	172

4119 rows × 18 columns

In [27]:
```
df_encoded['deposit'].value_counts()
```

Out[27]:
```
deposit
0    3668
1     451
Name: count, dtype: int64
```

In [28]:
```
x = df_encoded.drop('deposit',axis=1)  # independent variable
y = df_encoded['deposit']              # 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 [29]:
```
from sklearn.model_selection import train_test_split

print(4119*0.25)
```

```
1029.75
```

In [30]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=1)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [31]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

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 [32]:
```
from sklearn.tree import DecisionTreeClassifier

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

Out[32]:
```
▾                 DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

In [33]:
```
mscore(dt)
```

```
Training Score 0.9148591777274199
Testing Score 0.8990291262135922
```

In [34]:
```
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

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

In [35]:
```
eval_model(y_test,ypred_dt)
```

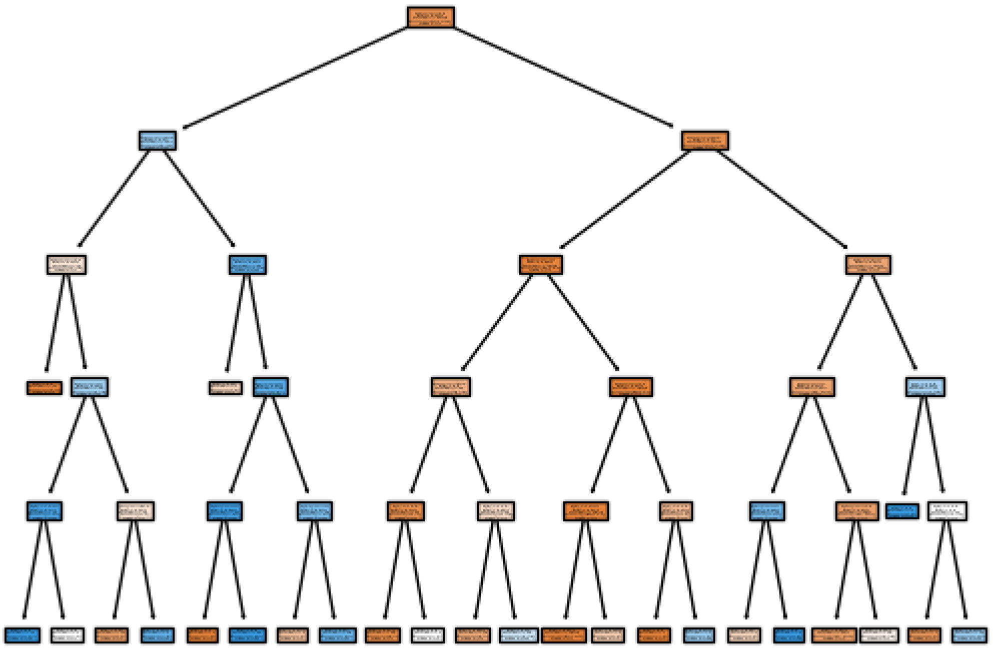
```
Accuracy_Score 0.8990291262135922
Confusion Matrix
 [[905  25]
 [ 79  21]]
Classification Report
               precision    recall  f1-score   support

           0       0.92      0.97      0.95       930
           1       0.46      0.21      0.29       100

    accuracy                           0.90      1030
   macro avg       0.69      0.59      0.62      1030
weighted avg       0.87      0.90      0.88      1030
```

In [36]:
```
from sklearn.tree import plot_tree
```

In [37]:
```
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 [38]:
```
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



In [39]:
```
dt1 = DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=15)
dt1.fit(x_train,y_train)
```

Out[39]:
```
▾                     DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=15)
```

In [40]:
```
mscore(dt1)
```

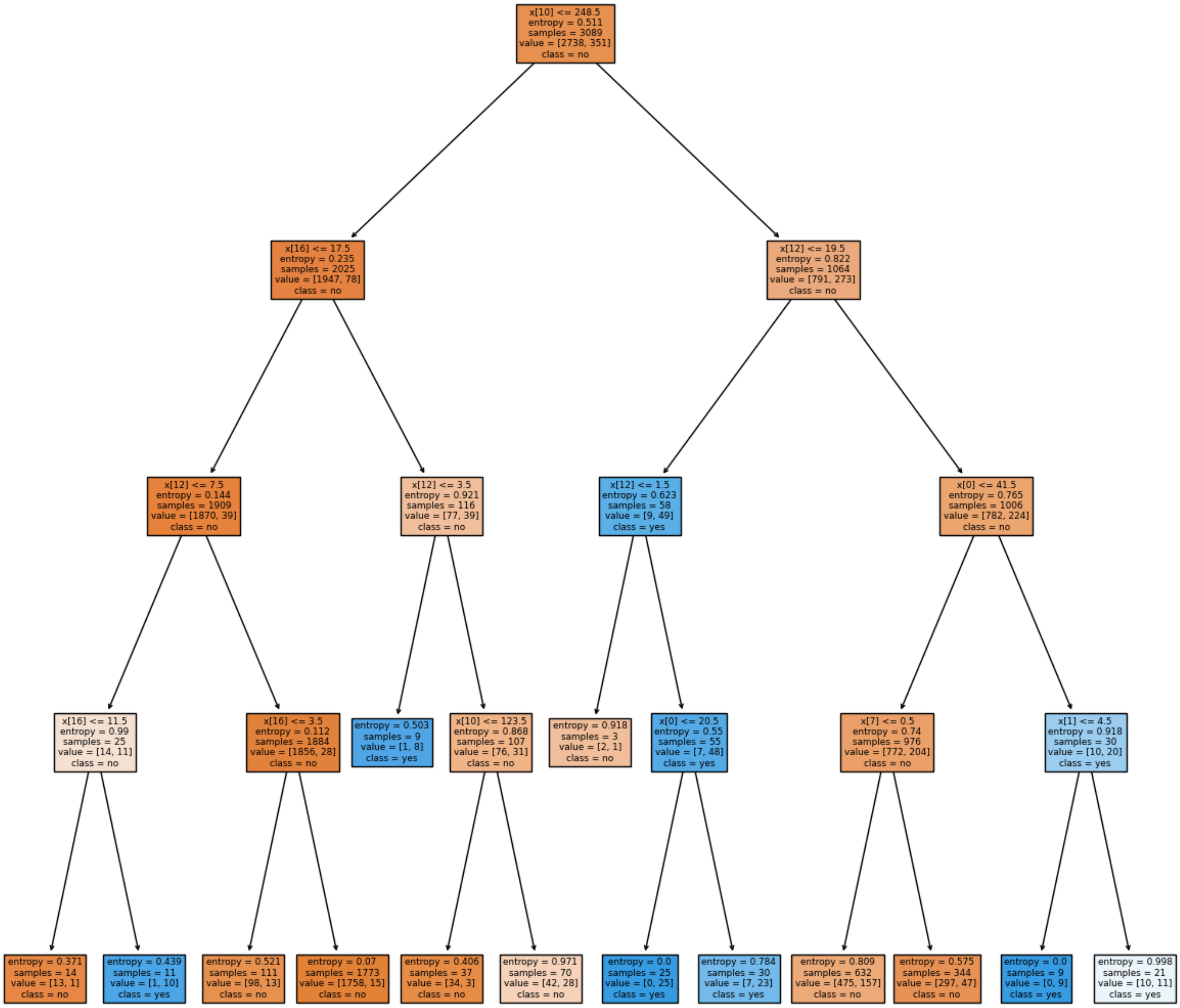
```
Training Score 0.9080608611201036
Testing Score 0.9048543689320389
```

In [41]:
```
ypred_dt1 = dt1.predict(x_test)
```

In [42]:
```
eval_model(y_test,ypred_dt1)
```

```
Accuracy_Score 0.9048543689320389
Confusion Matrix
 [[915  15]
 [ 83  17]]
Classification Report
               precision    recall  f1-score   support

           0       0.92      0.98      0.95       930
           1       0.53      0.17      0.26       100

    accuracy                           0.90      1030
   macro avg       0.72      0.58      0.60      1030
weighted avg       0.88      0.90      0.88      1030
```

In [43]:
```
plt.figure(figsize=(15,15))
plot_tree(dt1,class_names=cn,filled=True)
plt.show()
```


x[10] <= 248.5
entropy = 0.511
samples = 3089
value = [2738, 351]
class = no
x[16] <= 17.5
entropy = 0.235
samples = 2025
value = [1947, 78]
class = no
x[12] <= 19.5
entropy = 0.822
samples = 1064
value = [791, 273]
class = no
x[12] <= 7.5
entropy = 0.144
samples = 1909
value = [1870, 39]
class = no
x[12] <= 3.5
entropy = 0.921
samples = 116
value = [77, 39]
class = no
x[12] <= 1.5
entropy = 0.623
samples = 58
value = [9, 49]
class = yes
x[0] <= 41.5
entropy = 0.765
samples = 1006
value = [782, 224]
class = no
x[16] <= 11.5
entropy = 0.99
samples = 25
value = [14, 11]
class = no
x[16] <= 3.5
entropy = 0.112
samples = 1884
value = [1856, 28]
class = no
entropy = 0.503
samples = 9
value = [1, 8]
class = yes
x[10] <= 123.5
entropy = 0.868
samples = 107
value = [76, 31]
class = no
entropy = 0.918
samples = 3
value = [2, 1]
class = no
x[0] <= 20.5
entropy = 0.55
samples = 55
value = [7, 48]
class = yes
x[7] <= 0.5
entropy = 0.74
samples = 976
value = [772, 204]
class = no
x[1] <= 4.5
entropy = 0.918
samples = 30
value = [10, 20]
class = yes
entropy = 0.371
samples = 14
value = [13, 1]
class = no
entropy = 0.439
samples = 11
value = [1, 10]
class = yes
entropy = 0.521
samples = 111
value = [98, 13]
class = no
entropy = 0.07
samples = 1773
value = [1758, 15]
class = no
entropy = 0.406
samples = 37
value = [34, 3]
class = no
entropy = 0.971
samples = 70
value = [42, 28]
class = no
entropy = 0.0
samples = 25
value = [0, 25]
class = yes
entropy = 0.784
samples = 30
value = [7, 23]
class = yes
entropy = 0.809
samples = 632
value = [475, 157]
class = no
entropy = 0.575
samples = 344
value = [297, 47]
class = no
entropy = 0.0
samples = 9
value = [0, 9]
class = yes
entropy = 0.998
samples = 21
value = [10, 11]
class = yes

```
In [ ]:
```