```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
df = pd.read_csv('/content/bank-additional.csv',delimiter=';')
df.rename(columns={'y':'deposit'}, inplace=True)
df.head()
```

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

5 rows × 21 columns

```
df.head()
```

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

5 rows × 21 columns

```
df.tail()
```

	age	job	marital	education	default	housing	loan	contact	month	day_
4114	30	admin.	married	basic.6y	no	yes	yes	cellular	jul	
4115	39	admin.	married	high.school	no	yes	no	telephone	jul	
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

```
df.shape
```

```
(4119, 21)
```

```
df.columns
```

```
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')
```

```
df.dtypes
```

```
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
```

```
df.dtypes.value_counts()
```

```
object     11
int64       5
float64     5
dtype: int64
```

```
df.info
```

```
<bound method DataFrame.info of       age          job  marital          education default  housing     loan  \
0      30  blue-collar  married           basic.9y      no      yes       no
1      39     services   single        high.school      no       no       no
2      25     services  married        high.school      no      yes       no
3      38     services  married           basic.9y      no  unknown  unknown
4      47       admin.  married  university.degree      no      yes       no
...   ...          ...      ...                ...     ...      ...      ...
4114   30       admin.  married           basic.6y      no      yes      yes
4115   39       admin.  married        high.school      no      yes       no
4116   27      student   single        high.school      no       no       no
4117   58       admin.  married        high.school      no       no       no
4118   34   management   single        high.school      no      yes       no

        contact month day_of_week  ...  campaign  pdays  previous  \
0      cellular   may         fri  ...         2    999         0
1     telephone   may         fri  ...         4    999         0
2     telephone   jun         wed  ...         1    999         0
3     telephone   jun         fri  ...         3    999         0
4      cellular   nov         mon  ...         1    999         0
...         ...   ...         ...  ...       ...    ...       ...
4114   cellular   jul         thu  ...         1    999         0
4115  telephone   jul         fri  ...         1    999         0
4116   cellular   may         mon  ...         2    999         1
4117   cellular   aug         fri  ...         1    999         0
4118   cellular   nov         wed  ...         1    999         0

         poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0     nonexistent         -1.8          92.893          -46.2      1.313
1     nonexistent          1.1          93.994          -36.4      4.855
2     nonexistent          1.4          94.465          -41.8      4.962
3     nonexistent          1.4          94.465          -41.8      4.959
4     nonexistent         -0.1          93.200          -42.0      4.191
...           ...          ...             ...            ...        ...
4114  nonexistent          1.4          93.918          -42.7      4.958
4115  nonexistent          1.4          93.918          -42.7      4.959
4116      failure         -1.8          92.893          -46.2      1.354
4117  nonexistent          1.4          93.444          -36.1      4.966
4118  nonexistent         -0.1          93.200          -42.0      4.120

      nr.employed  deposit
0          5099.1       no
1          5191.0       no
2          5228.1       no
3          5228.1       no
4          5195.8       no
...           ...      ...
4114       5228.1       no
```

```
4115       5228.1       no
4116       5099.1       no
4117       5228.1       no
4118       5195.8       no

[4119 rows x 21 columns]>
```

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

```
0
```

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

```
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
```

```
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')
```

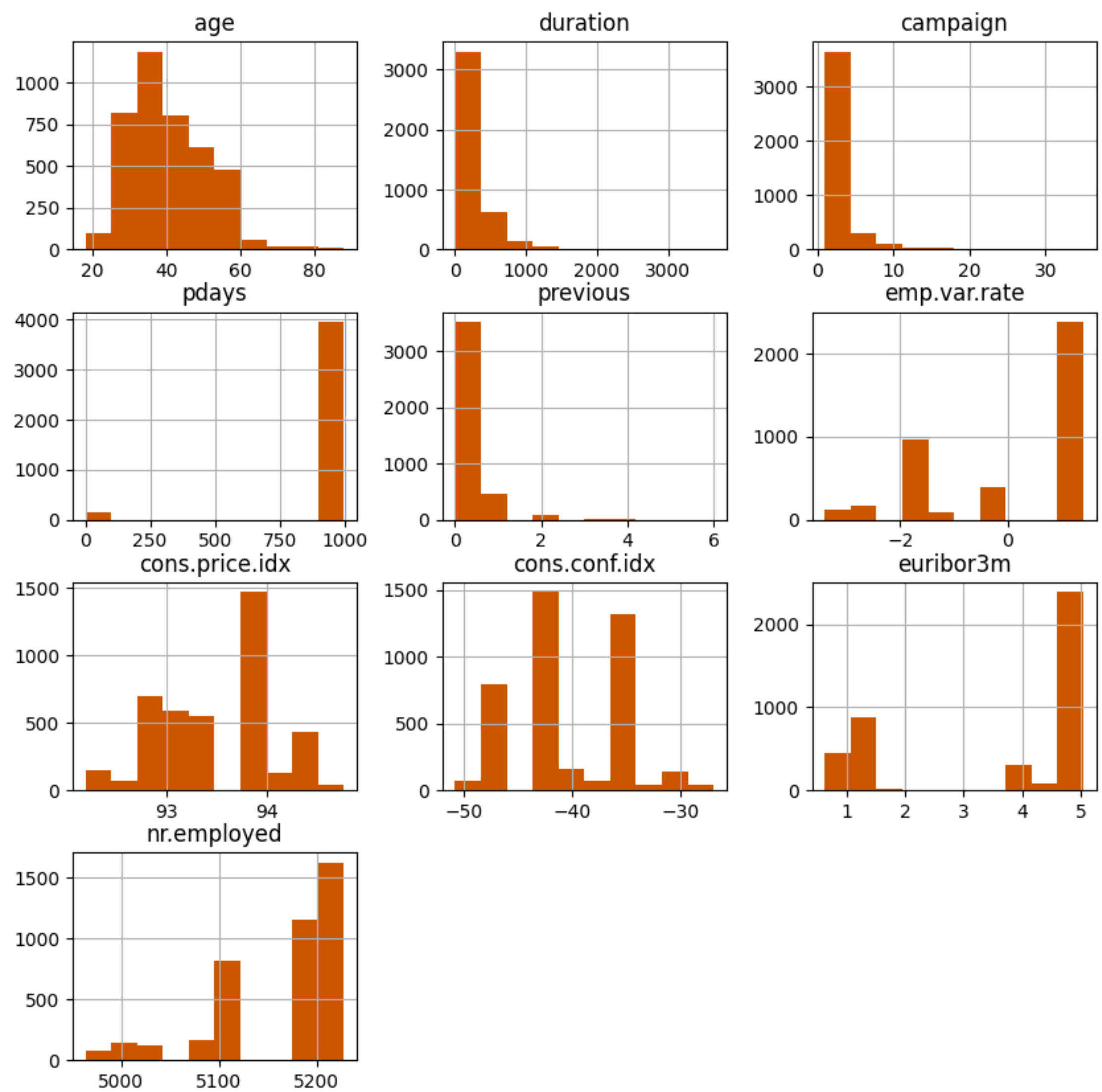
```
df.describe()
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employe
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.0000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704	-40.499102	3.621356	5166.4816
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349	4.594578	1.733591	73.6679
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.635000	4963.6000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.334000	5099.1000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.0000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.1000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000	-26.900000	5.045000	5228.1000

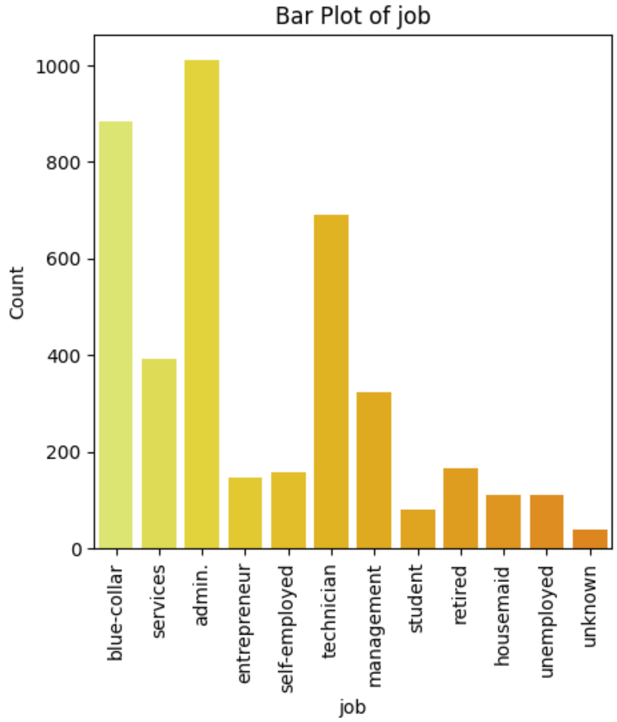
```
df.describe(include='object')
```

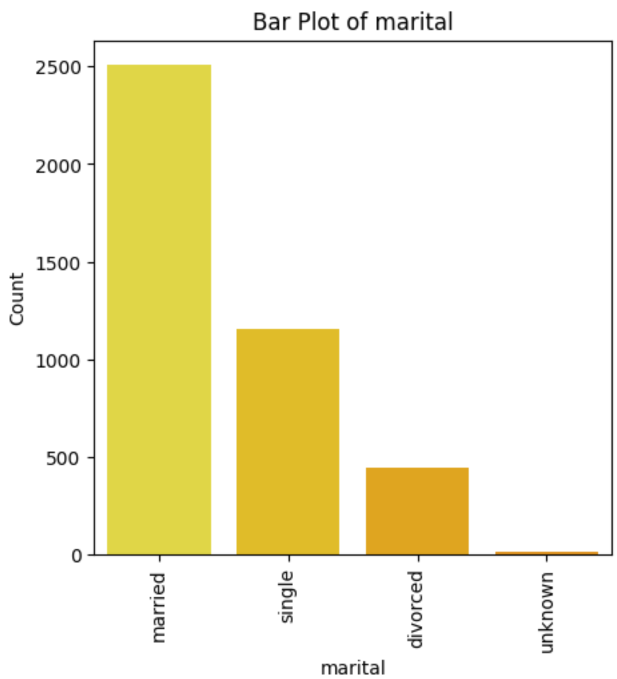
	job	marital	education	default	housing	loan	contact	month	day_of_week	poutcome	deposit
count	4119	4119	4119	4119	4119	4119	4119	4119	4119	4119	4119
unique	12	4	8	3	3	3	2	10	5	3	2
top	admin.	married	university.degree	no	yes	no	cellular	may	thu	nonexistent	no
freq	1012	2509	1264	3315	2175	3349	2652	1378	860	3523	3668

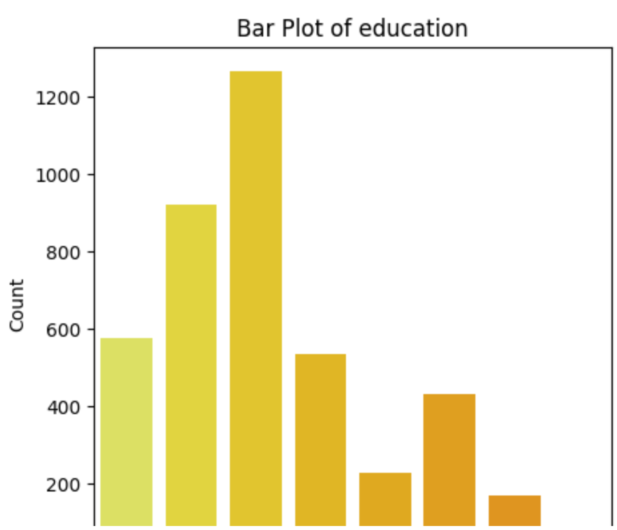
```
df.hist(figsize=(10,10),color='#cc5500')
plt.show()
```

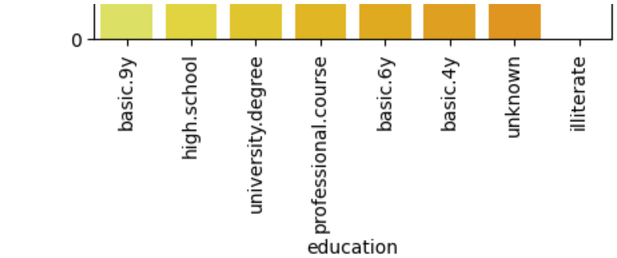


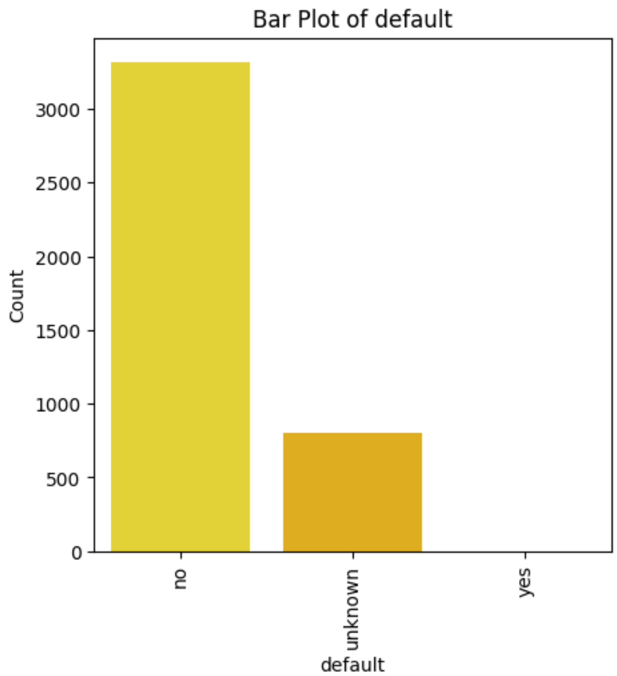

```
for feature in cat_cols:
 plt.figure(figsize=(5,5)) # Adjust the figure size as needed
 sns.countplot(x=feature, data=df, palette='Wistia')
 plt.title(f'Bar Plot of {feature}')
 plt.xlabel(feature)
 plt.ylabel('Count')
 plt.xticks(rotation=90)
 plt.show()
```

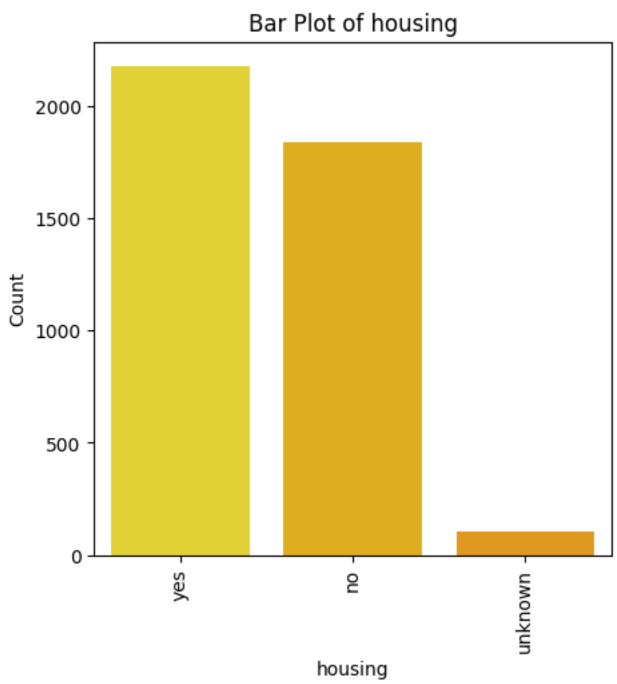

Bar Plot of job
Count
0
200
400
600
800
1000
blue-collar
services
admin.
entrepreneur
self-employed
technician
management
student
retired
housemaid
unemployed
unknown
job

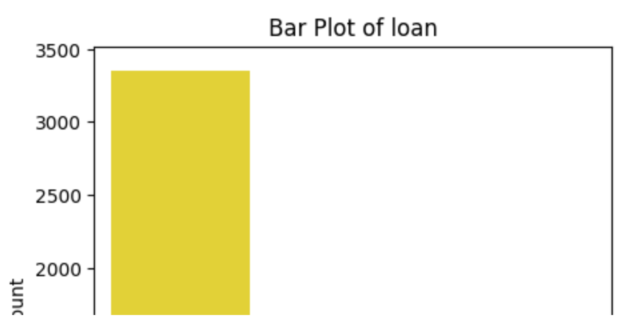

Bar Plot of marital
Count
0
500
1000
1500
2000
2500
married
single
divorced
unknown
marital

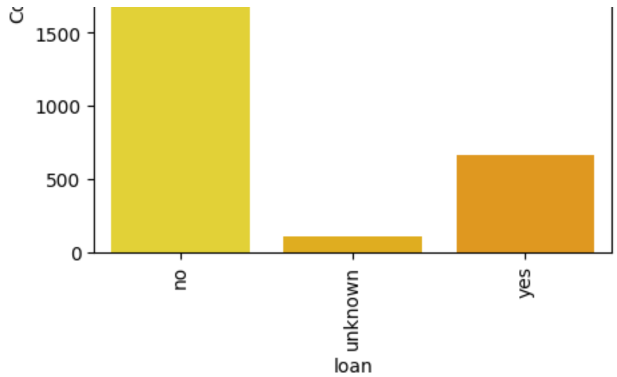

Bar Plot of education
Count
200
400
600
800
1000
1200


0
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate
education

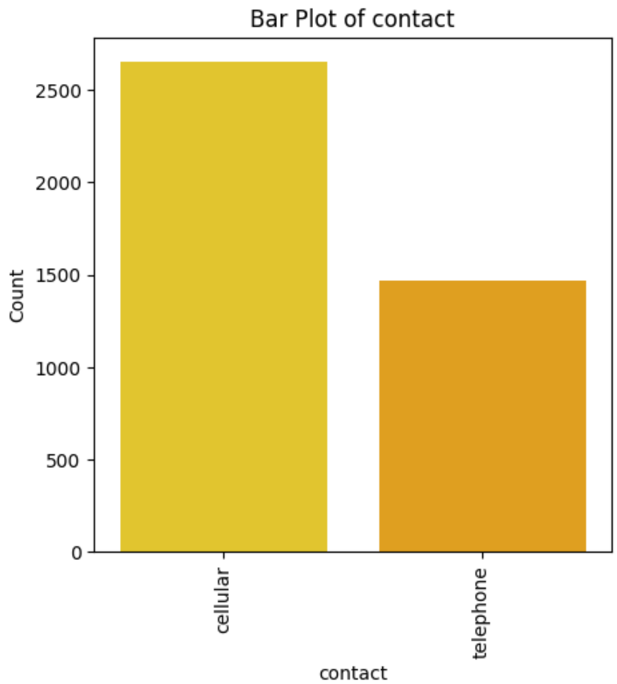

Bar Plot of default
Count
3000
2500
2000
1500
1000
500
0
no
unknown
yes
default


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

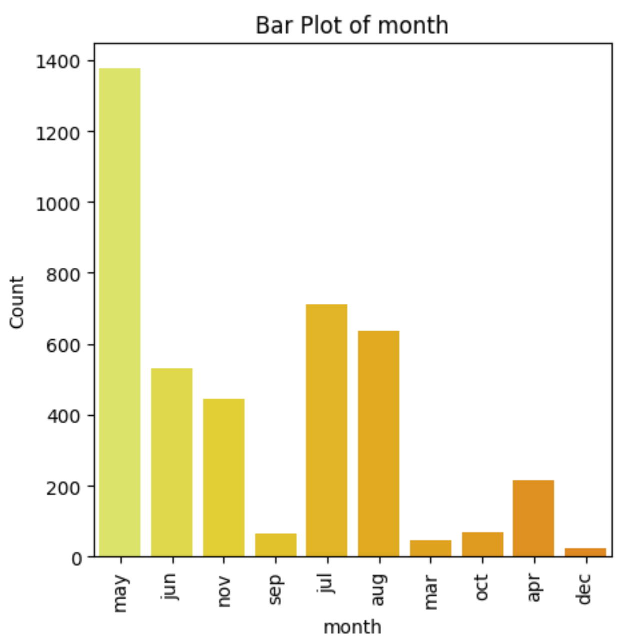

Bar Plot of loan
3500
3000
2500
2000



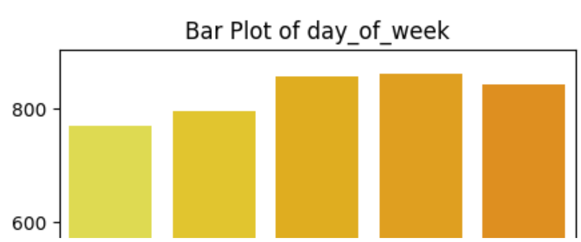
Co
1500
1000
500
0
no
unknown
yes
loan



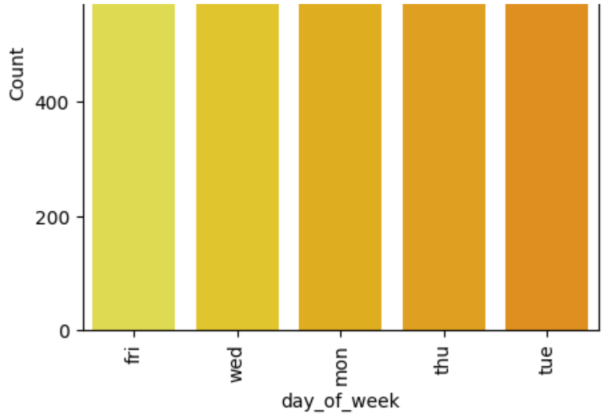
Bar Plot of contact
Count
2500
2000
1500
1000
500
0
cellular
telephone
contact

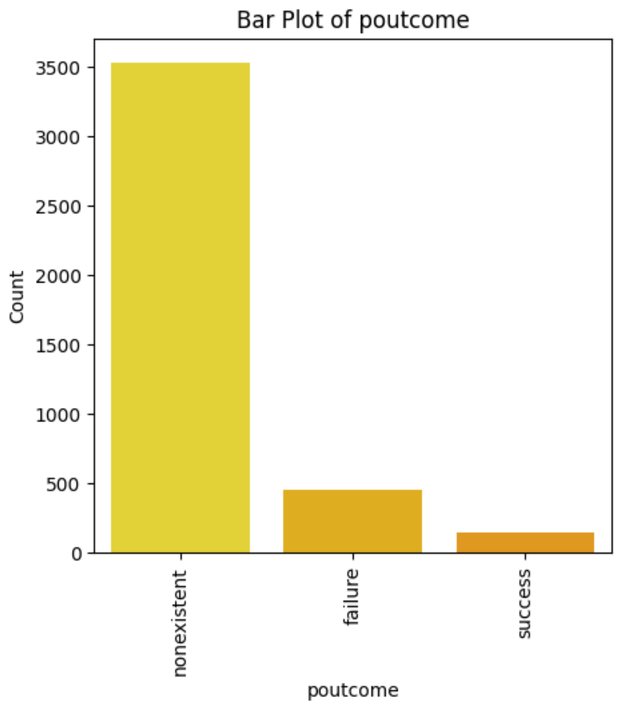


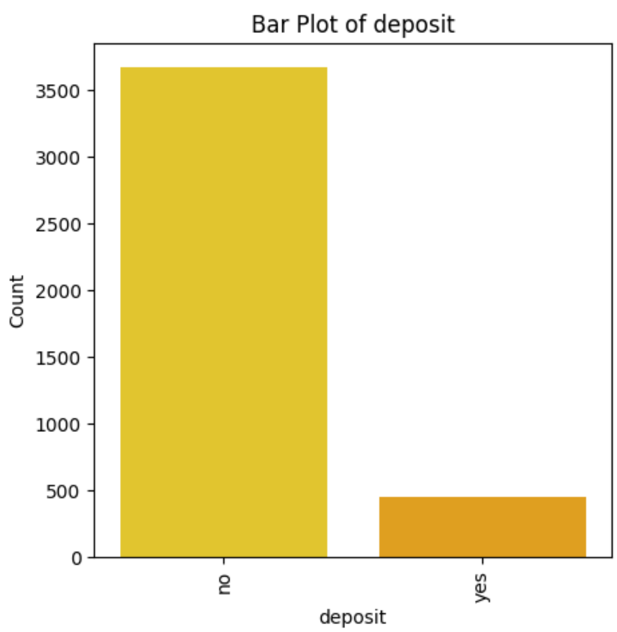
Bar Plot of month
Count
1400
1200
1000
800
600
400
200
0
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
month



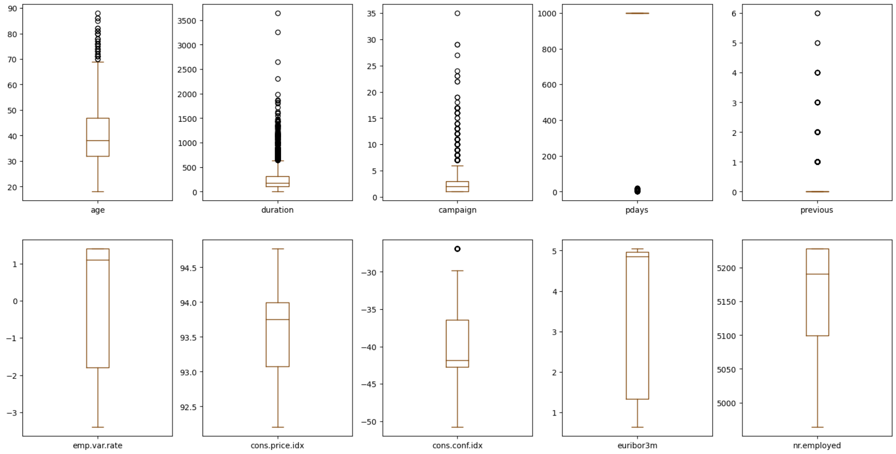
Bar Plot of day_of_week
800
600





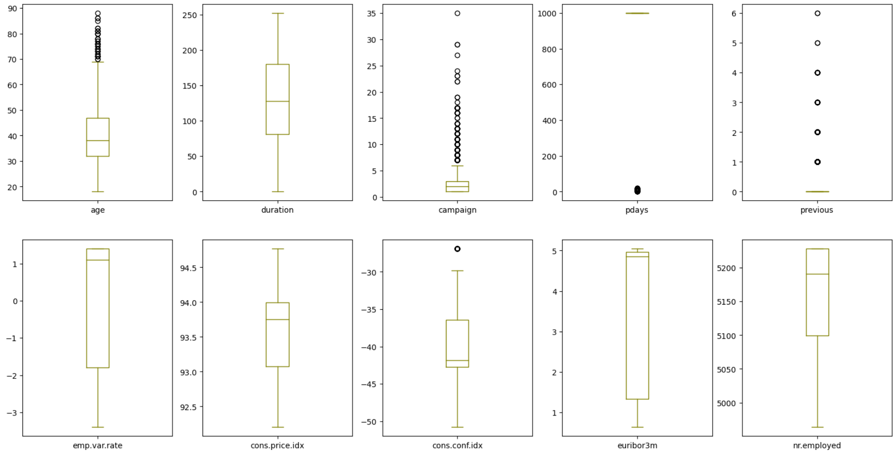



```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#7b3f00')
plt.show()
```



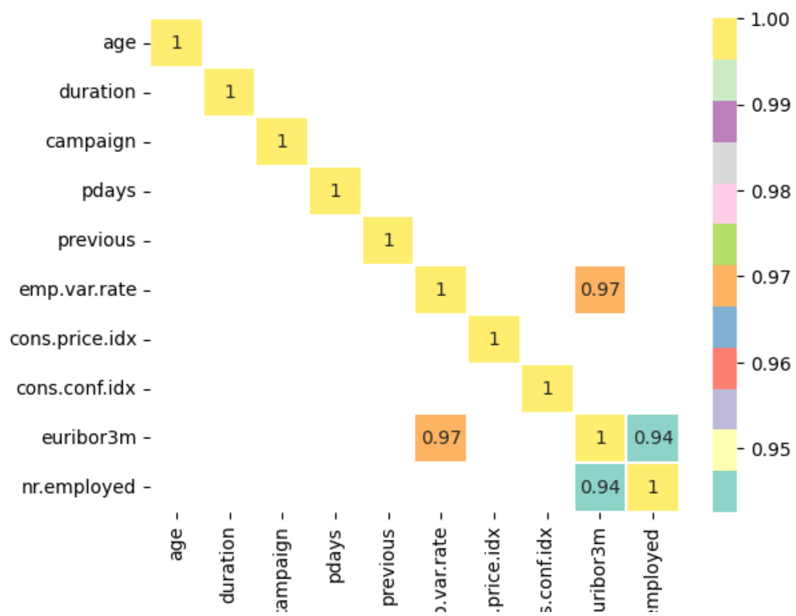

```
column = df[['age','campaign','duration']]
q1 = np.percentile(column, 25)
q3 = np.percentile(column, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
df[['age','campaign','duration']] = column[(column > lower_bound) & (column < upper_bound)]
```

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




```
corr = df.corr()
print(corr)
corr = corr[abs(corr)>=0.90]
sns.heatmap(corr,annot=True,cmap='Set3',linewidths=0.2)
plt.show()
```

```
                    age  duration  campaign     pdays  previous  \
age            1.000000  0.014048 -0.014169 -0.043425  0.050931
duration       0.014048  1.000000 -0.218111 -0.093694  0.094206
campaign      -0.014169 -0.218111  1.000000  0.058742 -0.091490
pdays         -0.043425 -0.093694  0.058742  1.000000 -0.587941
previous       0.050931  0.094206 -0.091490 -0.587941  1.000000
emp.var.rate  -0.019192 -0.063870  0.176079  0.270684 -0.415238
cons.price.idx -0.000482 -0.013338  0.145021  0.058472 -0.164922
cons.conf.idx  0.098135  0.045889  0.007882 -0.092090 -0.051420
euribor3m     -0.015033 -0.067815  0.159435  0.301478 -0.458851
nr.employed   -0.041936 -0.097339  0.161037  0.381983 -0.514853

               emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
age               -0.019192       -0.000482       0.098135  -0.015033
duration          -0.063870       -0.013338       0.045889  -0.067815
campaign           0.176079        0.145021       0.007882   0.159435
pdays              0.270684        0.058472      -0.092090   0.301478
previous          -0.415238       -0.164922      -0.051420  -0.458851
emp.var.rate       1.000000        0.755155       0.195022   0.970308
cons.price.idx     0.755155        1.000000       0.045835   0.657159
cons.conf.idx      0.195022        0.045835       1.000000   0.276595
euribor3m          0.970308        0.657159       0.276595   1.000000
nr.employed        0.897173        0.472560       0.107054   0.942589

               nr.employed
age              -0.041936
duration         -0.097339
campaign          0.161037
pdays             0.381983
previous         -0.514853
emp.var.rate      0.897173
cons.price.idx    0.472560
cons.conf.idx     0.107054
euribor3m         0.942589
nr.employed       1.000000
```




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

```
df1 = df.copy()
df1.columns
```

```
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')
```

```
df1.drop(high_corr_cols,inplace=True,axis=1) # axis=1 indicates columns
df1.columns
```

```
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')
```

```
df1.shape
```

```
(4119, 18)
```

```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df_encoded = df1.apply(lb.fit_transform)
df_encoded
```

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

4119 rows × 18 columns

```
df_encoded['deposit'].value_counts()
```

```
0    3668
1     451
Name: deposit, dtype: int64
```

```
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'>
```

```
from sklearn.model_selection import train_test_split
print(4119*0.25)
```

```
1029.75
```

```
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,)
```

```
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)
```

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

```
▾                 DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, min_samples_split=10)
```

```
mscore(dt)
```

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

```
ypred_dt = dt.predict(x_test)
print(ypred_dt)
```

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

```
eval_model(y_test,ypred_dt)
```

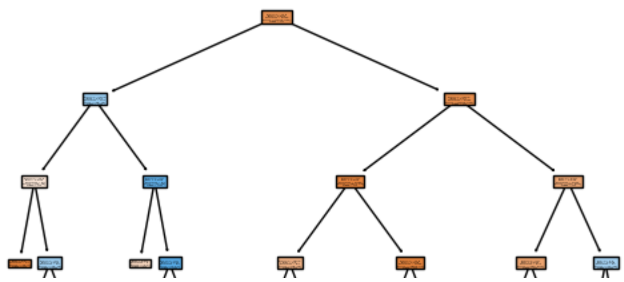
```
Accuracy_Score 0.8990291262135922
Confusion Matrix
 [[905  25]
 [ 79  21]]
Classification Report
               precision    recall  f1-score   support

           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
```

```
from sklearn.tree import plot_tree
```

```
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']
```

```
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



```
dt1 = DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=15)
dt1.fit(x_train,y_train)
```

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

```
mscore(dt1)
```

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

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

```
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
```

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