In [11]:
```
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 [69]:
```
bank = pd.read_csv('C:/Users/DELL/Desktop/new_bank.csv', delimiter=';')
bank.rename(columns={'y':'deposit'}, inplace=True)
```

In [70]:
```
bank.head()
```

Out[70]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previ
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	

5 rows × 21 columns

In [71]:
```
# showing last 5 rows
bank.tail()
```

Out[71]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previou
4114	30	admin.	married	basic.6y	no	yes	yes	cellular	jul	thu	...	1	999	
4115	39	admin.	married	high.school	no	yes	no	telephone	jul	fri	...	1	999	
4116	27	student	single	high.school	no	no	no	cellular	may	mon	...	2	999	
4117	58	admin.	married	high.school	no	no	no	cellular	aug	fri	...	1	999	
4118	34	management	single	high.school	no	yes	no	cellular	nov	wed	...	1	999	

5 rows × 21 columns

In [72]:
```
# showing dimensions of the dataset
bank.shape
```

Out[72]:
```
(4119, 21)
```

In [73]:
```
bank.columns
```

Out[73]:
```
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 [74]:
```
# checking for data types
bank.dtypes
```

```
Out[74]: 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 [75]: # showing information about the dataset
         bank.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 [76]: # checking for duplicates
         bank.duplicated().sum()
```

```
Out[76]: 0
```

```
In [77]: #Handling null values
         bank.isna().sum()
```

Out[77]:
```
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 [78]:
```python
# Extracting Numerical and Categorical Columns
cat_cols = bank.select_dtypes(include='object').columns
print(cat_cols)
```

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

In [79]:
```python
num_cols = bank.select_dtypes(exclude='object').columns
print(num_cols)
```

```
Index(['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate',
       'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'],
      dtype='object')
```

In [80]:
```python
# For Numerical Columns
bank.describe()
```

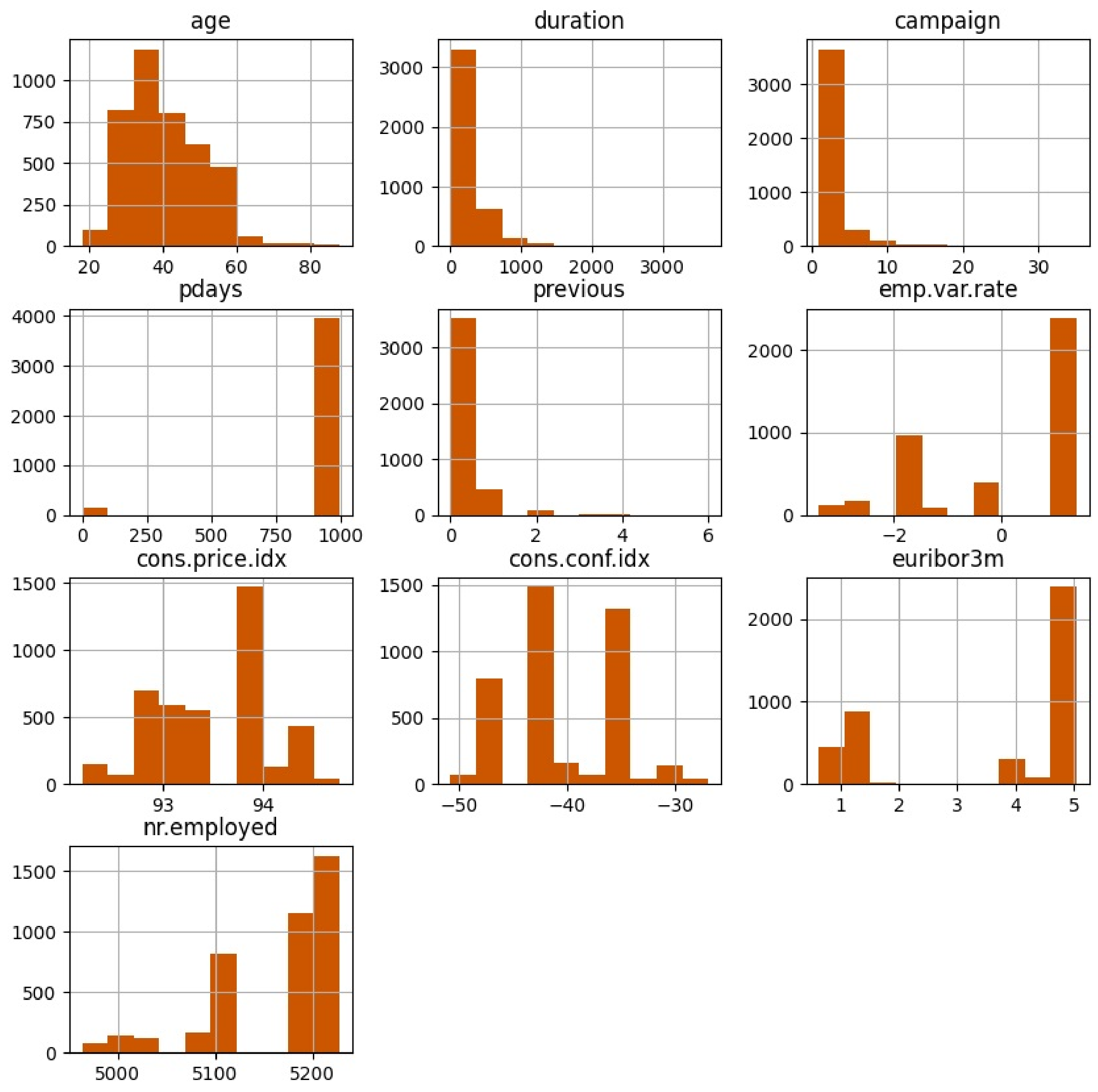
Out[80]:

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

In [81]:
```
Data Visualizations
```

```
  Cell In[81], line 1
    Data Visualizations
         ^
SyntaxError: invalid syntax
```

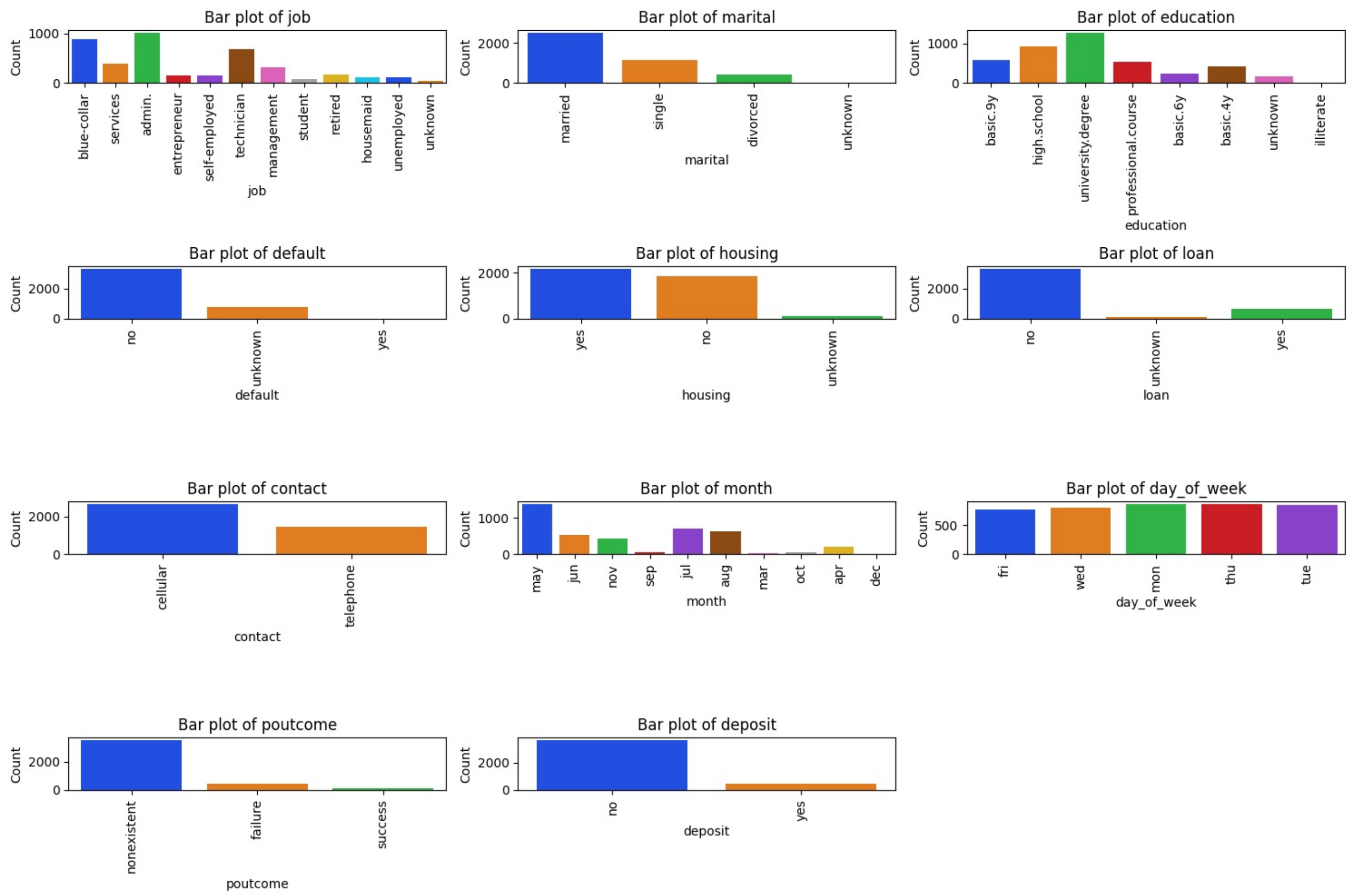
In [82]:
```python
# Visualizing Numerical columns using Histplot
bank.hist(figsize=(10,10),color='#cc5500')
plt.show()
```



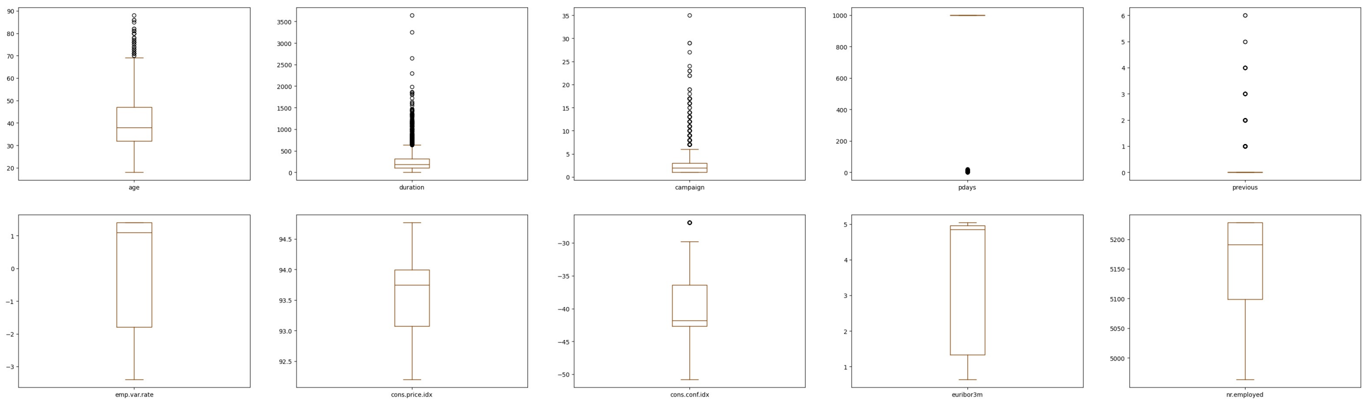
```
In [83]: Visualising categorial columns
```

```
  Cell In[83], line 1
    Visualising categorial columns
                ^
SyntaxError: invalid syntax
```

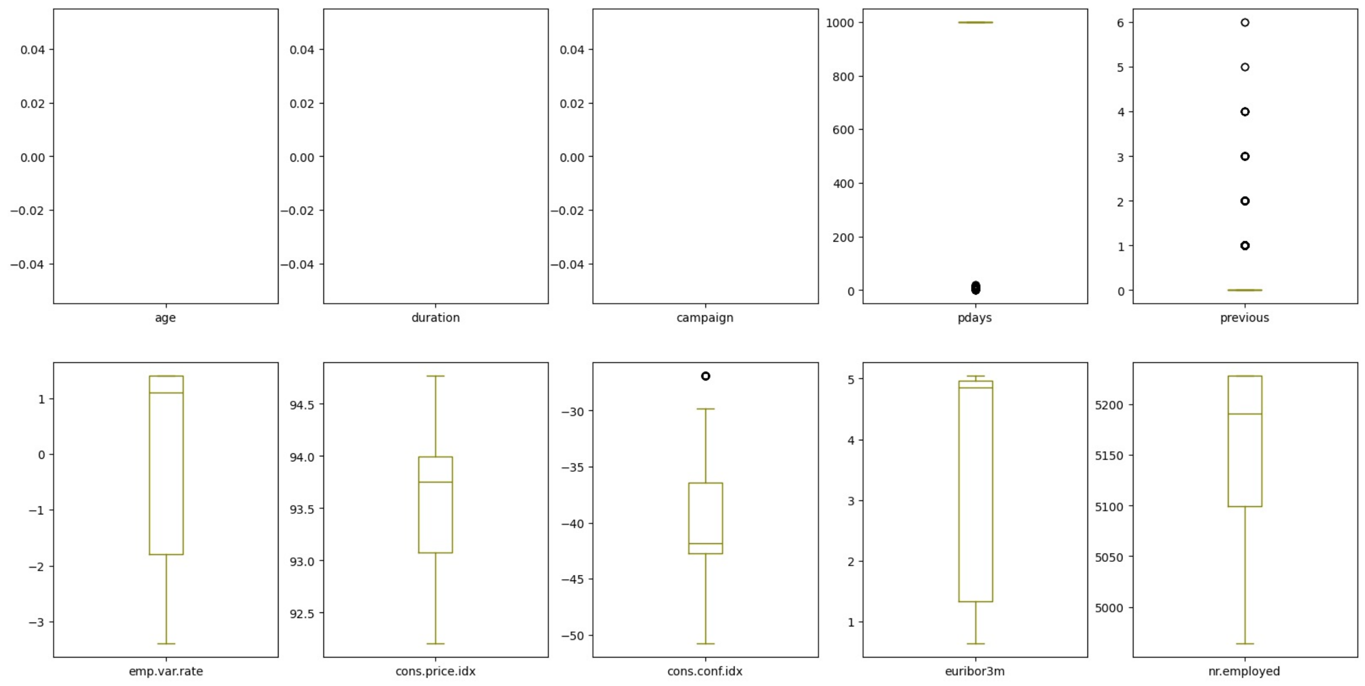
```
In [84]: num_rows = len(cat_cols)//2 - 1
         num_cols=2 if len(cat_cols)%2==0 else 3

         # Create subplots
         fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 10))

         # Flatten axes for easy iteration
         axes = axes.flatten()

         for i,feature in enumerate(cat_cols):
             sns.countplot(x=feature,data=bank,palette='bright',ax=axes[i])
             axes[i].set_title(f'Bar plot of {feature}')
             axes[i].set_xlabel(feature)
             axes[i].set_ylabel('Count')
             axes[i].tick_params(axis='x',rotation=90)
         # Remove any empty subplots if the number of columns is odd
         if len(cat_cols) % 2 != 0:
             fig.delaxes(axes[-1])

         # Adjust layout
         plt.tight_layout()
         plt.show()
```



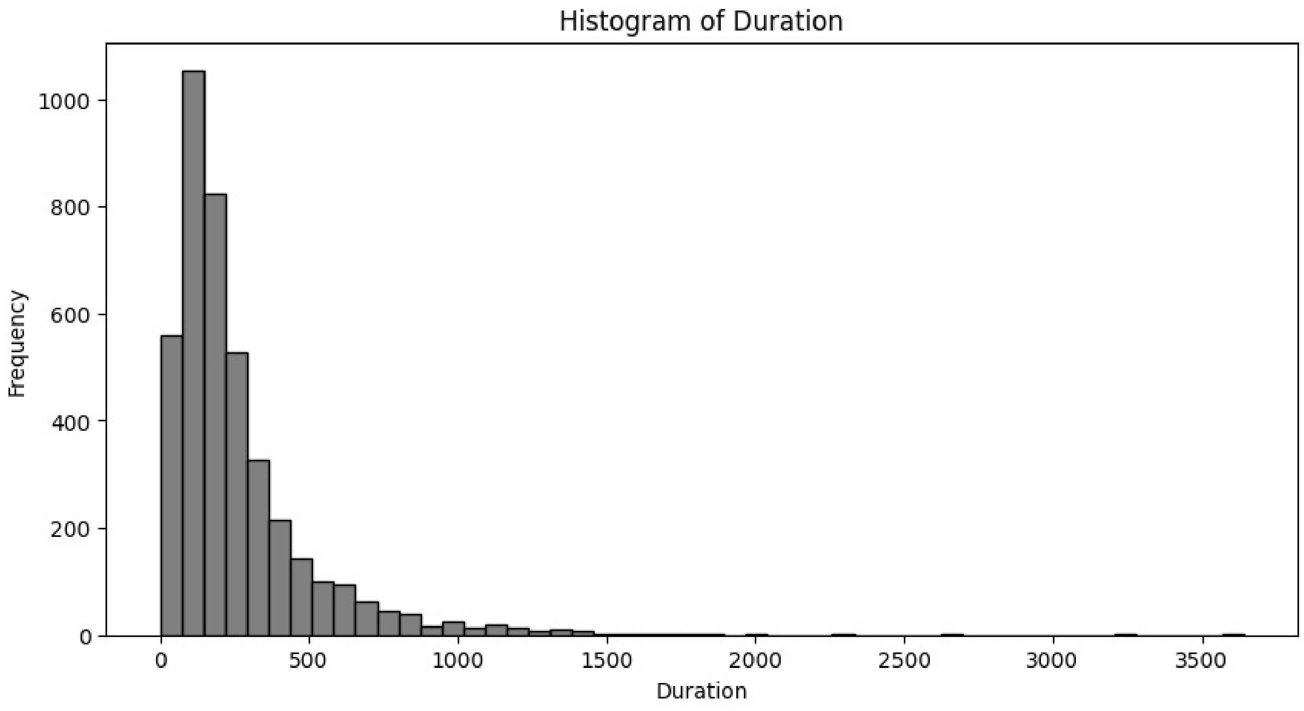
In [85]:
```
bank.plot(kind='box',subplots=True,layout=(5,5),figsize=(40,30),color='#7b3f03')
plt.show()
```

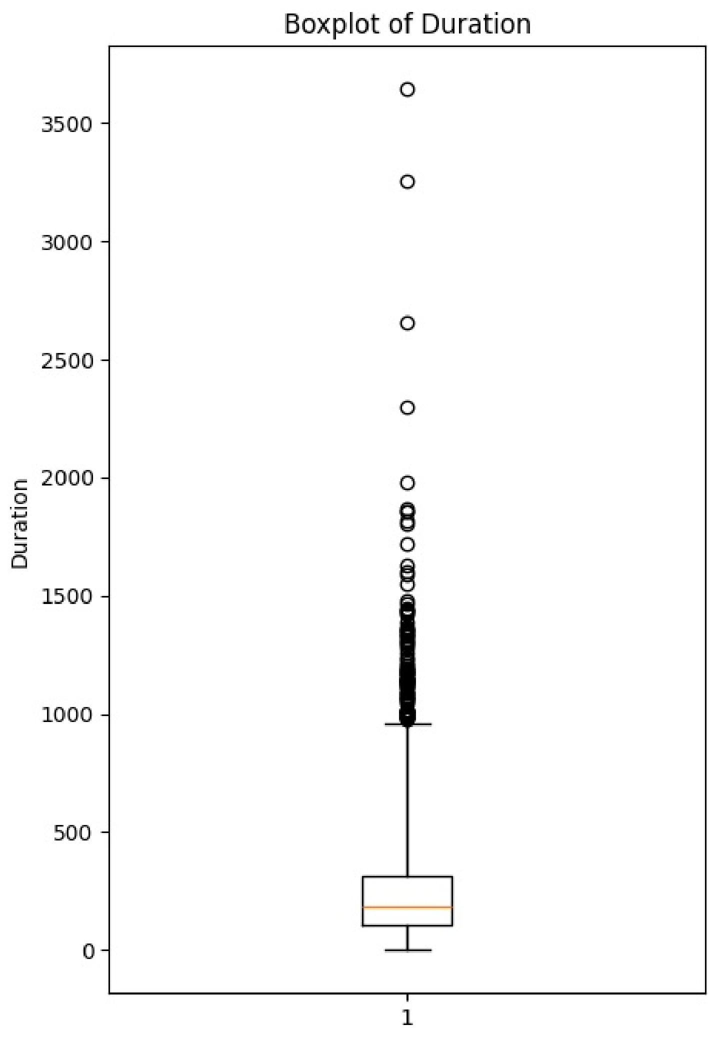


In [67]:
```
# Removing outliers

column = bank[['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
bank[['age','campaign','duration']] = column[(column > lower_bound) & (column < upper_bound)]


# Plotting boxplot after removing outliers
bank.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#808000')
plt.show()
```

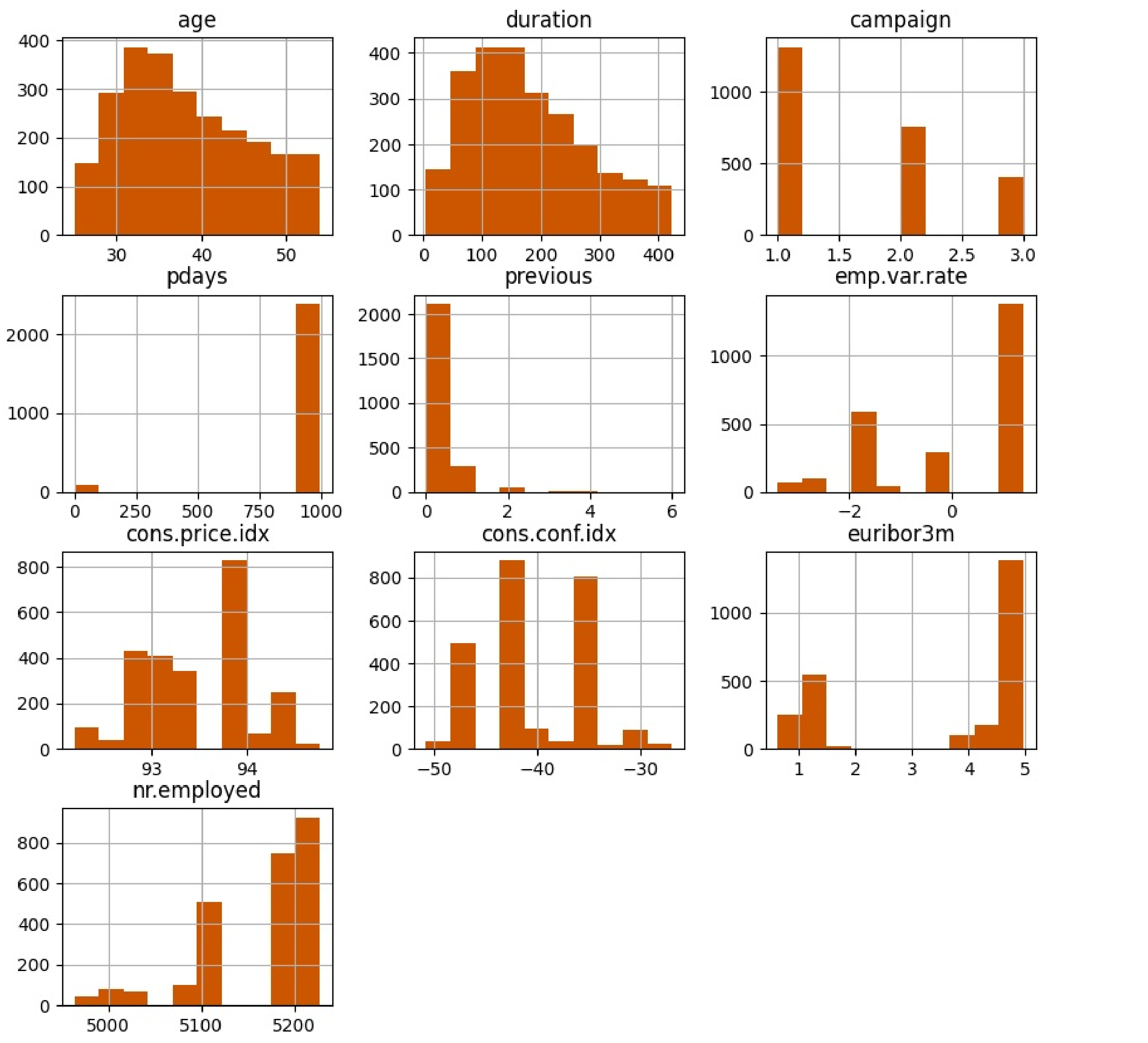


In [87]:
```python
# Plot histogram of 'duration'
plt.figure(figsize=(10, 5))
plt.hist(bank['duration'], bins=50, color='gray', edgecolor='black')
plt.title('Histogram of Duration')
plt.xlabel('Duration')
plt.ylabel('Frequency')
plt.show()

# Plotting a boxplot of 'duration' with potentially adjusted whiskers
plt.figure(figsize=(5, 8))
plt.boxplot(bank['duration'], whis=3)  # Increase the whisker multiplier to 3
plt.title('Boxplot of Duration')
plt.ylabel('Duration')
plt.show()
```

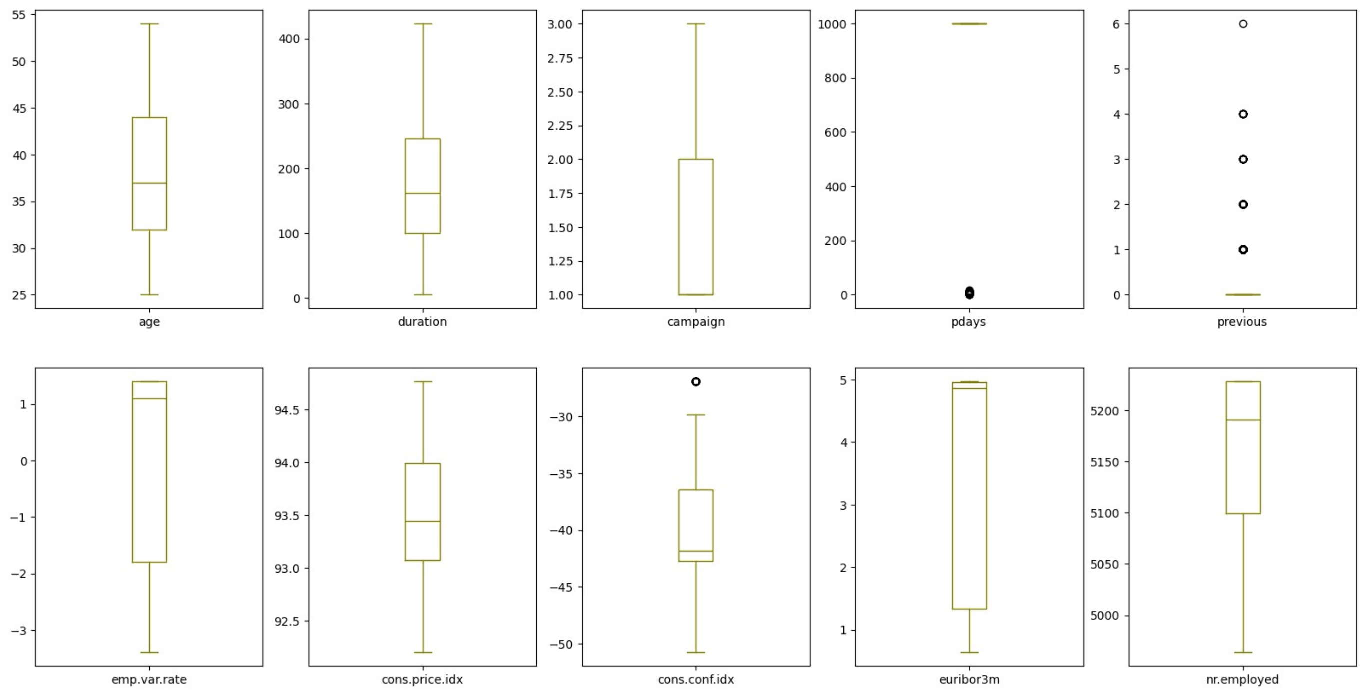




```
In [90]: # Visualizing Numerical columns using Histplot
         filtered_bank.hist(figsize=(10,10),color='#cc5500')
         plt.show()
```



In [89]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Assuming 'bank' is your dataframe

# Select the columns of interest
columns_of_interest = ['age', 'campaign', 'duration']

# Initialize an empty DataFrame to hold the filtered data
filtered_bank = bank.copy()

# Loop through each column to calculate and apply the IQR-based filter
for column in columns_of_interest:
    q1 = np.percentile(bank[column], 25)
    q3 = np.percentile(bank[column], 75)
    iqr = q3 - q1
    lower_bound = q1 - 0.5 * iqr
    upper_bound = q3 + 0.5 * iqr

    # Filter the data
    filtered_bank = filtered_bank[(bank[column] > lower_bound) & (bank[column] < upper_bound)]

# Plotting boxplot after removing outliers
filtered_bank.plot(kind='box', subplots=True, layout=(2,5), figsize=(20,10), color='#808000')
plt.show()
```

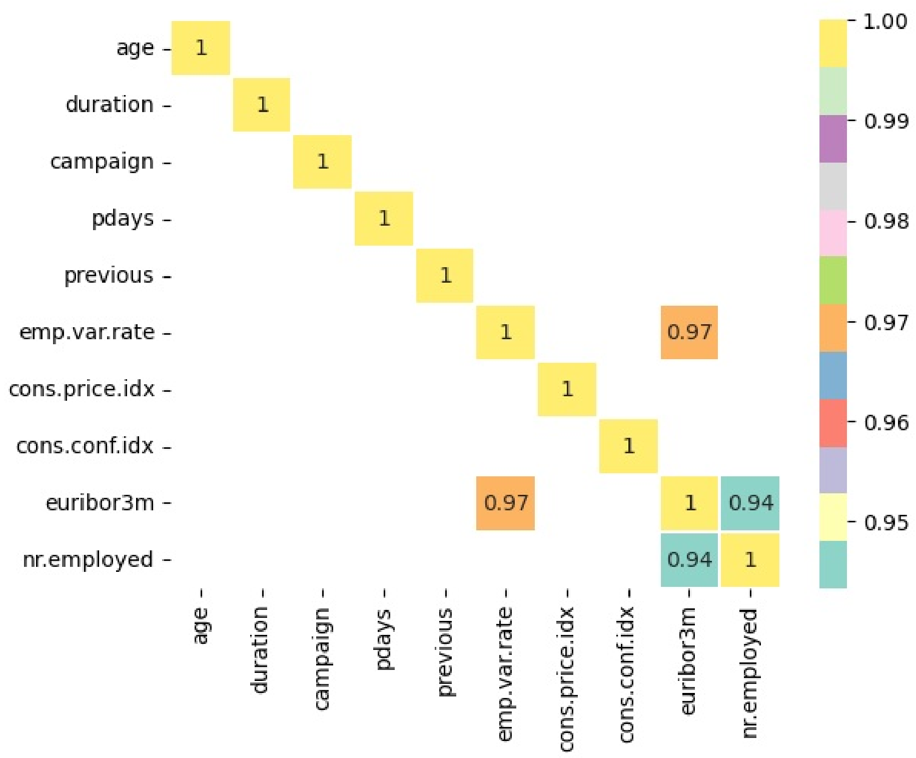


In [93]:
```
# Checking for correlation using Correlation Plot
numeric_cols = filtered_bank.select_dtypes(include=[np.number])

# Calculate the correlation matrix
corr = numeric_cols.corr()

# Print the correlation matrix
print(corr)
```

```
                     age  duration  campaign     pdays  previous  \
age             1.000000 -0.000105 -0.017044  0.028174 -0.018973
duration       -0.000105  1.000000  0.001270 -0.076364  0.057457
campaign       -0.017044  0.001270  1.000000  0.037584 -0.070025
pdays           0.028174 -0.076364  0.037584  1.000000 -0.574306
previous       -0.018973  0.057457 -0.070025 -0.574306  1.000000
emp.var.rate    0.067465 -0.057432  0.102236  0.272195 -0.425021
cons.price.idx  0.029300  0.014160  0.102541  0.087898 -0.197690
cons.conf.idx   0.051713 -0.007329  0.008159 -0.131763 -0.044772
euribor3m       0.081060 -0.067379  0.091352  0.293233 -0.460363
nr.employed     0.081648 -0.087543  0.084602  0.368059 -0.509903

                emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
age                 0.067465        0.029300       0.051713   0.081060
duration           -0.057432        0.014160      -0.007329  -0.067379
campaign            0.102236        0.102541       0.008159   0.091352
pdays               0.272195        0.087898      -0.131763   0.293233
previous           -0.425021       -0.197690      -0.044772  -0.460363
emp.var.rate        1.000000        0.763156       0.219754   0.969234
cons.price.idx      0.763156        1.000000       0.068188   0.665558
cons.conf.idx       0.219754        0.068188       1.000000   0.297615
euribor3m           0.969234        0.665558       0.297615   1.000000
nr.employed         0.898460        0.487764       0.125245   0.943299

                nr.employed
age                0.081648
duration          -0.087543
campaign           0.084602
pdays              0.368059
previous          -0.509903
emp.var.rate       0.898460
cons.price.idx     0.487764
cons.conf.idx      0.125245
euribor3m          0.943299
nr.employed        1.000000
```

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

In [94]:
```
corr = corr[abs(corr)>=0.90]
sns.heatmap(corr,annot=True,cmap='Set3',linewidths=0.2)
plt.show()
```



```
In [95]: # Feature Selection using Correlation
         high_corr_cols = ['emp.var.rate','euribor3m','nr.employed']
```

```
In [96]: # Removing high correlated columns from the dataset
         filtered_bank.drop(high_corr_cols,inplace=True,axis=1)  # axis=1 indicates columns
         filtered_bank.columns
```

```
Out[96]: 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 [98]: filtered_bank.shape
```

```
Out[98]: (2477, 18)
```

```
In [99]: # Conversion of categorical columns into numerical columns using label encoder.
         from sklearn.preprocessing import LabelEncoder
         lb = LabelEncoder()
         bank_df_encoded = filtered_bank.apply(lb.fit_transform)
         bank_df_encoded
```

Out[99]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	p
2	0	7	1	3	0	2	0	1	4	4	219	0	15	0	
3	13	7	1	2	0	1	1	1	4	0	12	2	15	0	
4	22	0	1	5	0	2	0	0	7	1	51	0	15	0	
5	7	7	2	5	0	0	0	0	9	2	121	2	15	2	
7	16	2	1	5	1	2	0	0	7	1	38	1	15	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4112	6	9	2	4	0	2	0	0	7	2	148	0	15	0	
4114	5	0	1	1	0	2	2	0	3	2	46	0	15	0	
4115	14	0	1	3	0	2	0	1	3	0	211	0	15	0	
4116	2	8	2	3	0	0	0	0	6	1	57	1	15	1	
4118	9	4	2	3	0	2	0	0	7	4	168	0	15	0	

2477 rows × 18 columns

```
In [102… bank_df_encoded['deposit'].value_counts()
```

```
Out[102… deposit
         0    2325
         1     152
         Name: count, dtype: int64
```

```
In [103…  x = bank_df_encoded.drop('deposit',axis=1)  # independent variable
          y = bank_df_encoded['deposit']              # dependent variable
          print(x.shape)
          print(y.shape)
          print(type(x))
          print(type(y))
```

```
(2477, 17)
(2477,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

```
In [104…  # Splitting the dataset into Train and Test datasets
          from sklearn.model_selection import train_test_split
```

```
In [105…  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)
```

```
(1857, 17)
(620, 17)
(1857,)
(620,)
```

```
In [106…  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)  # Training Accuracy
              print('Testing Score',test_score)    # Testing Accuracy
```

```
In [107…  # Importing  Decision Tree library
          from sklearn.tree import DecisionTreeClassifier
```

```
In [108…  # Building Decision Tree Classifier Model
          dt = DecisionTreeClassifier(criterion='gini',max_depth=5,min_samples_split=10)
          dt.fit(x_train,y_train)
```

```
Out[108…  ▾                    DecisionTreeClassifier

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

```
In [109…  # Evaluating training and testing accuracy
          mscore(dt)
```

```
Training Score 0.9633817985998923
Testing Score 0.9435483870967742
```

```
In [110…  # Generating prediction
          ypred_dt = dt.predict(x_test)
          print(ypred_dt)
```

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

In [111…

```
# # Evaluate the model - confusion matrix, classification Report, Accuaracy
eval_model(y_test,ypred_dt)
```

```
Accuracy_Score 0.9435483870967742
Confusion Matrix
 [[572   8]
 [ 27  13]]
Classification Report
               precision    recall  f1-score   support

           0       0.95      0.99      0.97       580
           1       0.62      0.33      0.43        40

    accuracy                           0.94       620
   macro avg       0.79      0.66      0.70       620
weighted avg       0.93      0.94      0.94       620
```

In [112…

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

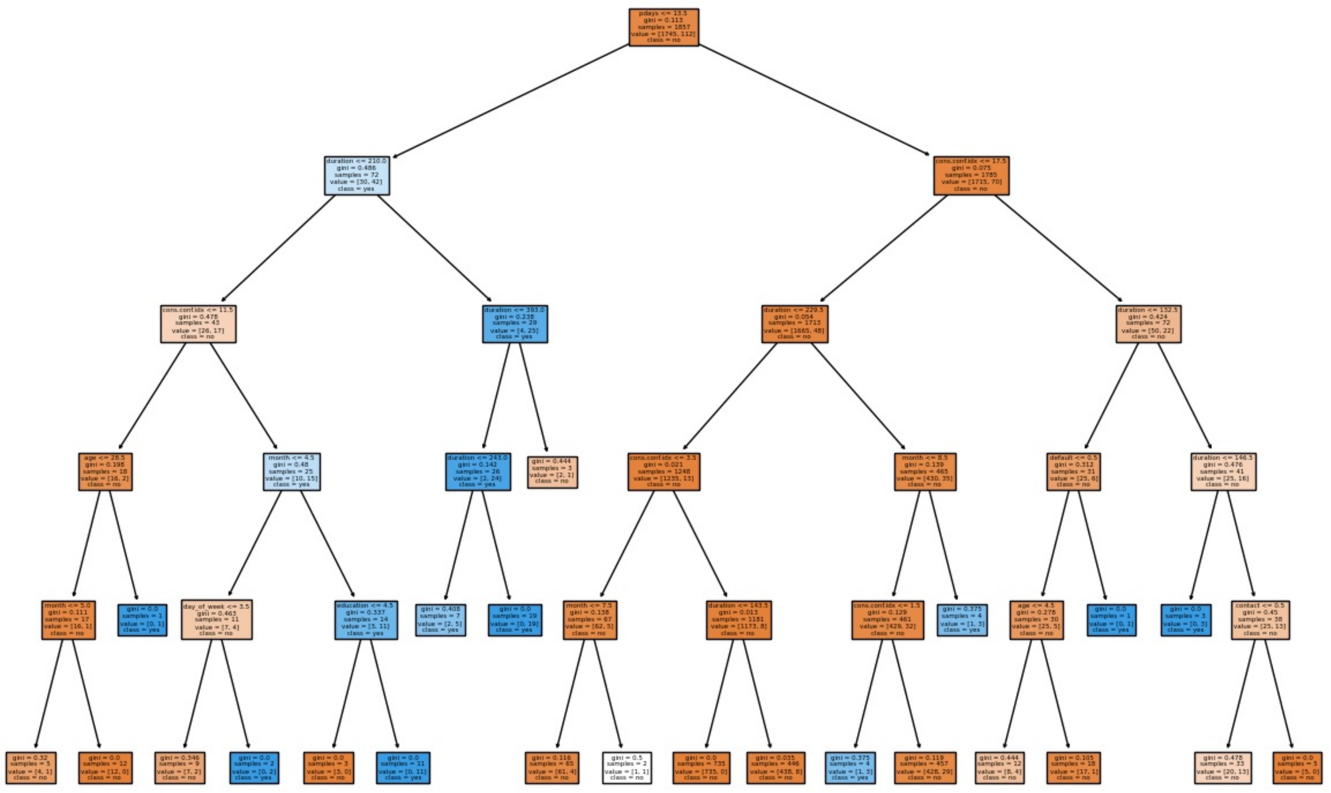
In [113…

```
# cn = class names, fn = feature_names
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 [115…

```
plt.figure(figsize=(15, 10))  # Set width to 15 inches and height to 10 inches
plot_tree(dt, feature_names=fn, class_names=cn, filled=True)
plt.show()
```



In [118…

```
# Generating prediction
ypred_dt1 = bank.predict(x_test)


# Evaluate the model - confusion matrix, classification Report, Accuaracy
eval_model(y_test,ypred_dt1)
```

```
Accuracy_Score 0.9451612903225807
Confusion Matrix
 [[578   2]
 [ 32   8]]
Classification Report
               precision    recall  f1-score   support

           0       0.95      1.00      0.97       580
           1       0.80      0.20      0.32        40

    accuracy                           0.95       620
   macro avg       0.87      0.60      0.65       620
weighted avg       0.94      0.95      0.93       620
```

In [116…

```
# Building Decision Tree Classifier Model
bank = DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=15)
bank.fit(x_train,y_train)
```

Out[116…

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

In [117…

```
mscore(bank)
```

```
Training Score 0.9569197630586969
Testing Score 0.9451612903225807
```

In [121…

```
ypred_dt1 = bank.predict(x_test)
print(ypred_dt)
```

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

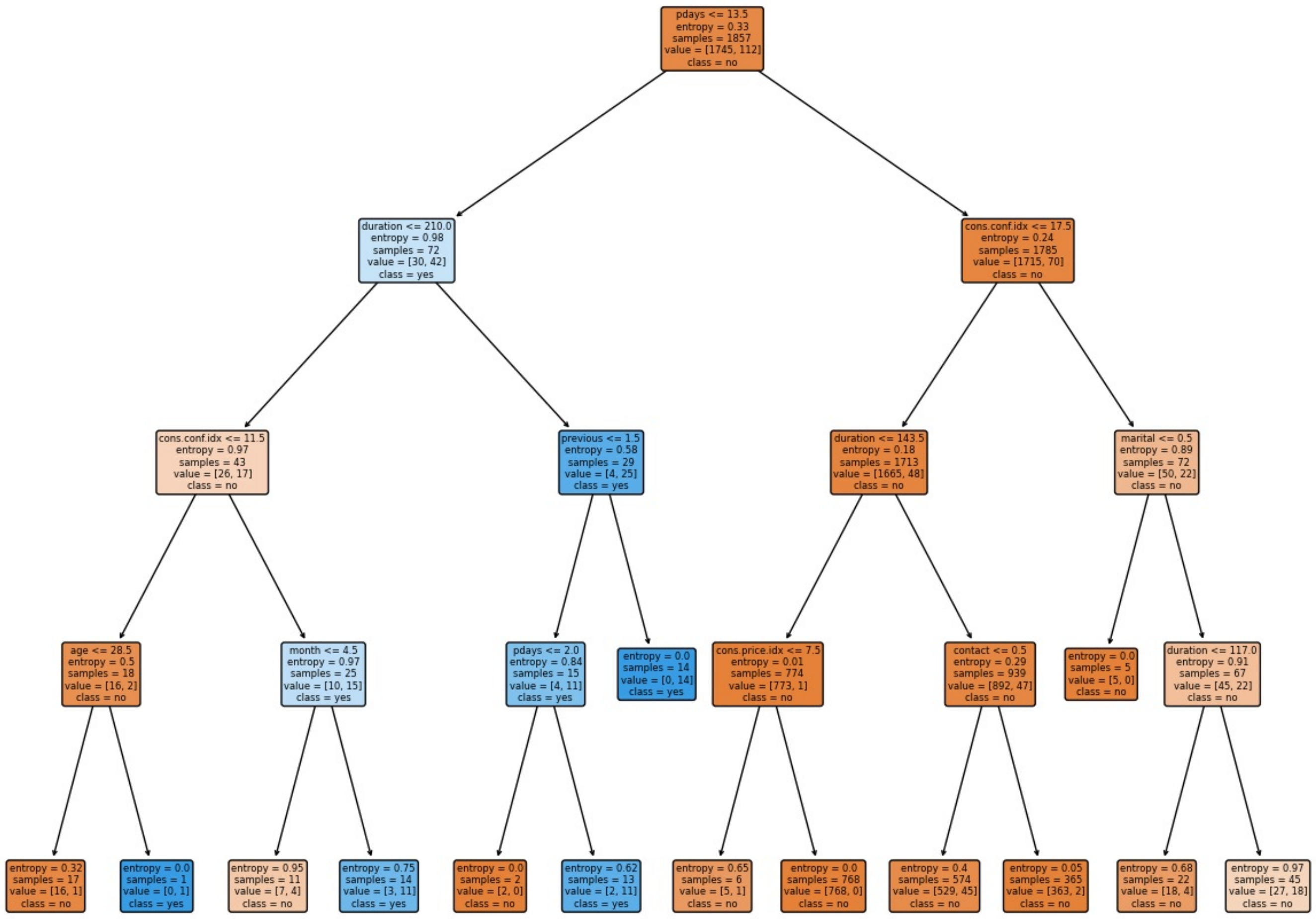
In [122…

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

```
Accuracy_Score 0.9451612903225807
Confusion Matrix
 [[578   2]
 [ 32   8]]
Classification Report
               precision    recall  f1-score   support

           0       0.95      1.00      0.97       580
           1       0.80      0.20      0.32        40

    accuracy                           0.95       620
   macro avg       0.87      0.60      0.65       620
weighted avg       0.94      0.95      0.93       620
```

In [123…

```
# Define the figure size for a square shape
plt.figure(figsize=(15, 12))

# Plot the decision tree
plot_tree(bank, feature_names=fn, class_names=cn, filled=True, proportion=False, rounded=True, precision=2)

# Show the plot
plt.show()
```


pdays <= 13.5
entropy = 0.33
samples = 1857
value = [1745, 112]
class = no
duration <= 210.0
entropy = 0.98
samples = 72
value = [30, 42]
class = yes
cons.conf.idx <= 17.5
entropy = 0.24
samples = 1785
value = [1715, 70]
class = no
cons.conf.idx <= 11.5
entropy = 0.97
samples = 43
value = [26, 17]
class = no
previous <= 1.5
entropy = 0.58
samples = 29
value = [4, 25]
class = yes
duration <= 143.5
entropy = 0.18
samples = 1713
value = [1665, 48]
class = no
marital <= 0.5
entropy = 0.89
samples = 72
value = [50, 22]
class = no
age <= 28.5
entropy = 0.5
samples = 18
value = [16, 2]
class = no
month <= 4.5
entropy = 0.97
samples = 25
value = [10, 15]
class = yes
pdays <= 2.0
entropy = 0.84
samples = 15
value = [4, 11]
class = yes
entropy = 0.0
samples = 14
value = [0, 14]
class = yes
cons.price.idx <= 7.5
entropy = 0.01
samples = 774
value = [773, 1]
class = no
contact <= 0.5
entropy = 0.29
samples = 939
value = [892, 47]
class = no
entropy = 0.0
samples = 5
value = [5, 0]
class = no
duration <= 117.0
entropy = 0.91
samples = 67
value = [45, 22]
class = no
entropy = 0.32
samples = 17
value = [16, 1]
class = no
entropy = 0.0
samples = 1
value = [0, 1]
class = yes
entropy = 0.95
samples = 11
value = [7, 4]
class = no
entropy = 0.75
samples = 14
value = [3, 11]
class = yes
entropy = 0.0
samples = 2
value = [2, 0]
class = no
entropy = 0.62
samples = 13
value = [2, 11]
class = yes
entropy = 0.65
samples = 6
value = [5, 1]
class = no
entropy = 0.0
samples = 768
value = [768, 0]
class = no
entropy = 0.4
samples = 574
value = [529, 45]
class = no
entropy = 0.05
samples = 365
value = [363, 2]
class = no
entropy = 0.68
samples = 22
value = [18, 4]
class = no
entropy = 0.97
samples = 45
value = [27, 18]
class = no

```
In [ ]:
```