In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
import seaborn as sns
# Load the dataset
data = pd.read_csv(r"C:\Users\DANIEL\Downloads\bank+marketing\bank\bank-full
data.rename(columns={'y':'deposit'}, inplace=True)
data.head()
```

Out[1]:

	age	job	marital	education	default	balance	housing	loan	contact	day	mon
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	m
1	44	technician	single	secondary	no	29	yes	no	unknown	5	m
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	m
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	m
4	33	unknown	single	unknown	no	1	no	no	unknown	5	m

In [2]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  deposit    45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

In [3]:

```
data.shape
```

Out[3]:

```
(45211, 17)
```

In [4]:

```
data.columns
```

Out[4]:

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housin
g',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

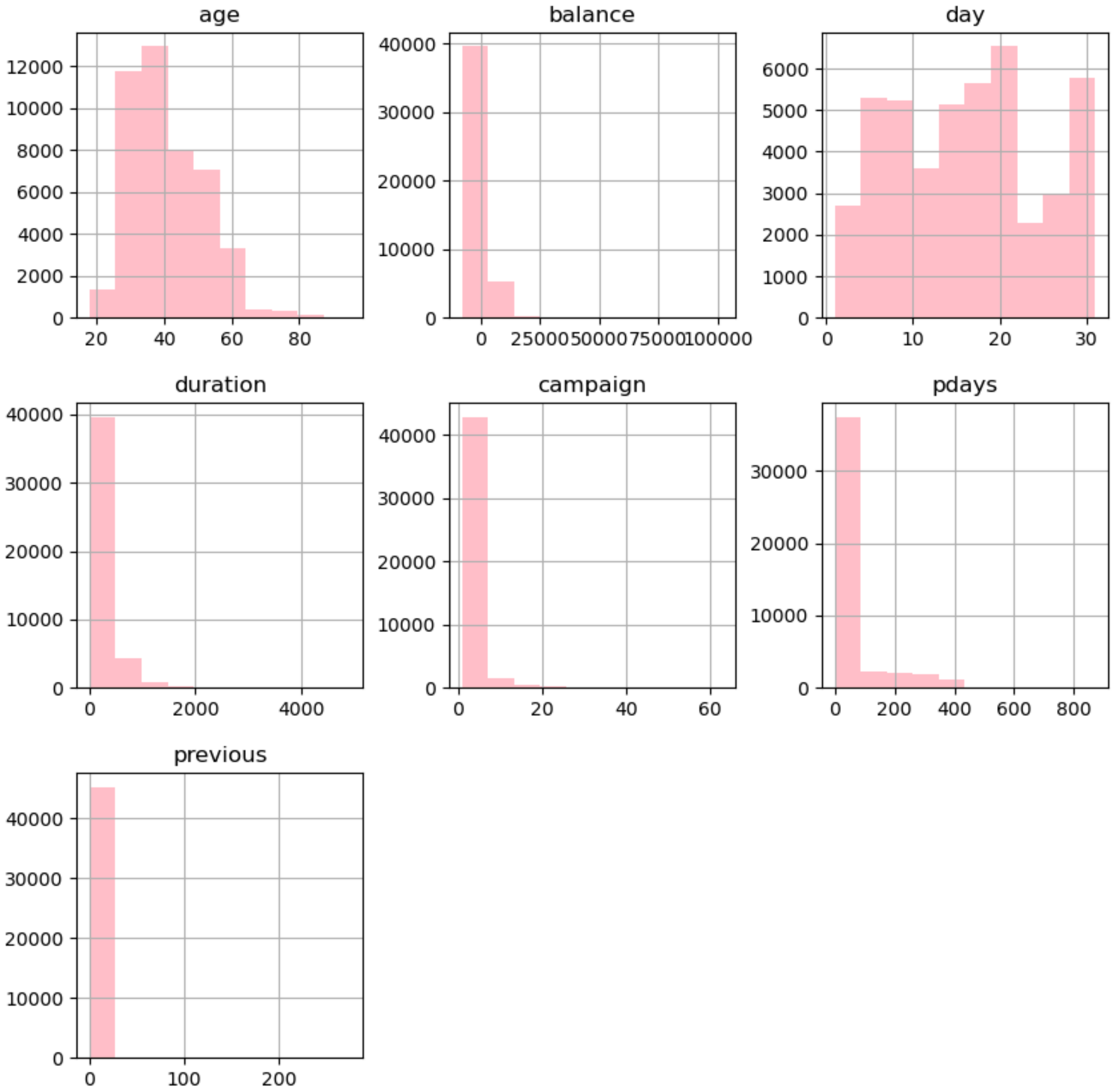
In [5]:

```
data.isna().sum()
```

Out[5]:

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
deposit      0
dtype: int64
```

In [6]:
```
data.hist(figsize=(10,10),color="pink")
plt.show()
```



In [7]:
```
cat_col = data.select_dtypes(include='object').columns
print(cat_col)

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

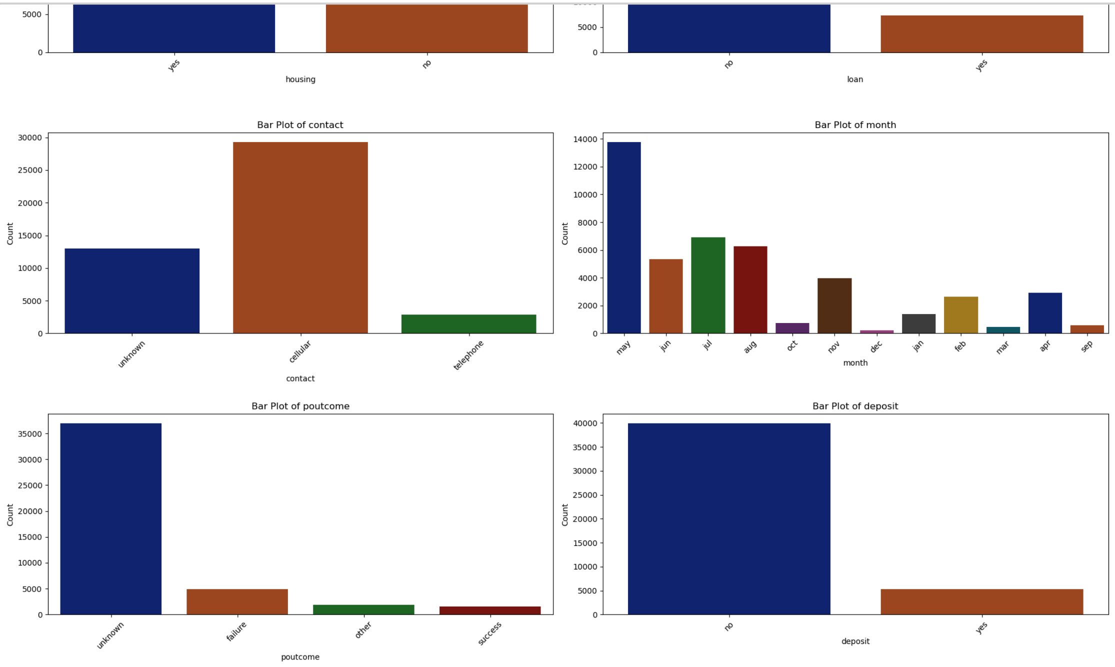
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'conta
ct',
       'month', 'poutcome', 'deposit'],
      dtype='object')
Index(['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previou
s'], dtype='object')
```

In [8]:
```
# Calculate the number of rows and columns for subplots
num_plots = len(cat_col)
num_rows = (num_plots + 1) // 2  # Add 1 and divide by 2 to round up for od
num_cols = 2
num_rows
```

Out[8]:
```
5
```

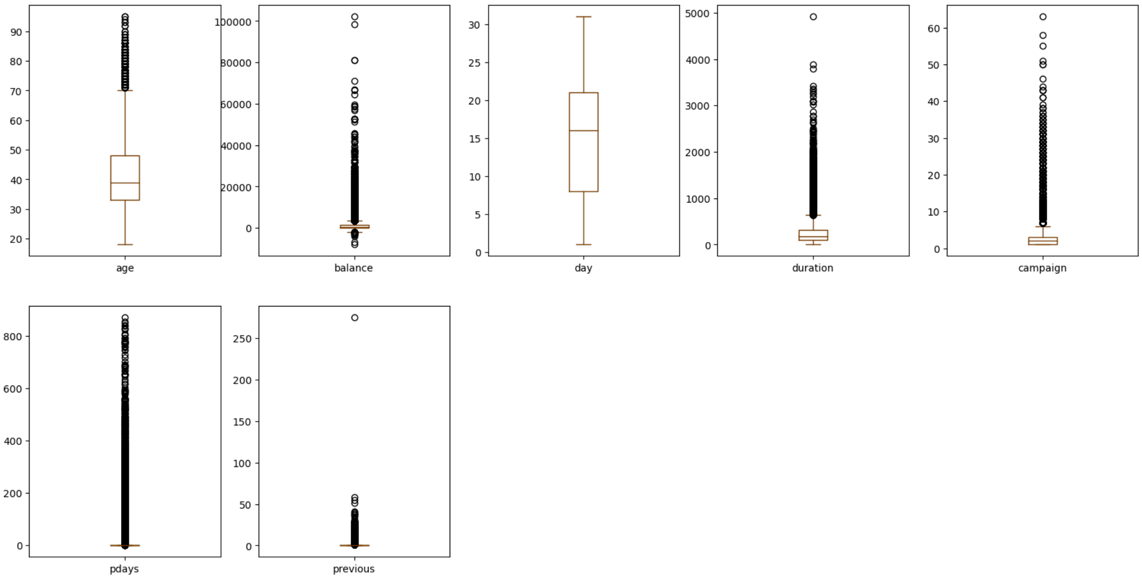
In [9]:

```python
plt.figure(figsize=(20, 25))  # Adjust the figure size as needed

# Loop through each feature and create a countplot
for i, feature in enumerate(cat_col, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.countplot(x=feature, data=data, palette='dark')
    plt.title(f'Bar Plot of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=45)

# Adjust layout to prevent overlap of subplots
plt.tight_layout()
plt.show()
```

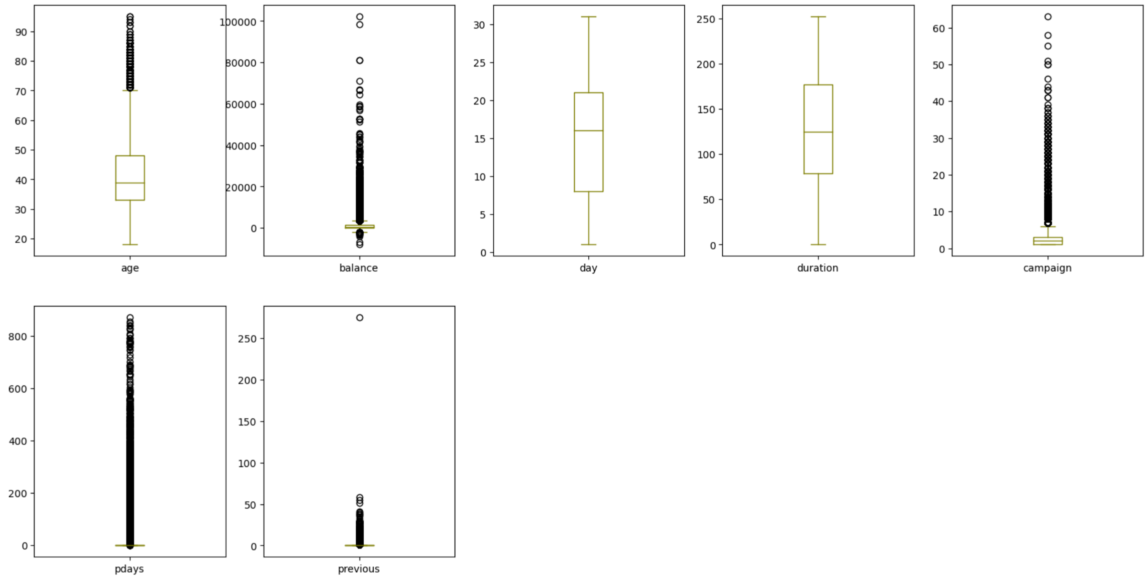


In [10]:

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



In [13]:
```
import numpy as np
column = data[['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
data[['age','campaign','duration']] = column[(column > lower_bound) & (colu
data.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='#8
plt.show()
```



In [ ]:
```
cat_col = data.select_dtypes(include='object').columns
print(cat_col)

# Exclude non-numeric columns
numeric_df = data.drop(columns=cat_col)

# Compute the correlation matrix
corr = numeric_df.corr()

# Filter correlations with absolute value >= 0.90
corr = corr[abs(corr) >= 0.90]

sns.heatmap(corr,annot=True,cmap='Set3',linewidths=0.2)
plt.show()
```

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

In [14]:
```
df1 = data.copy()
df1.columns
```

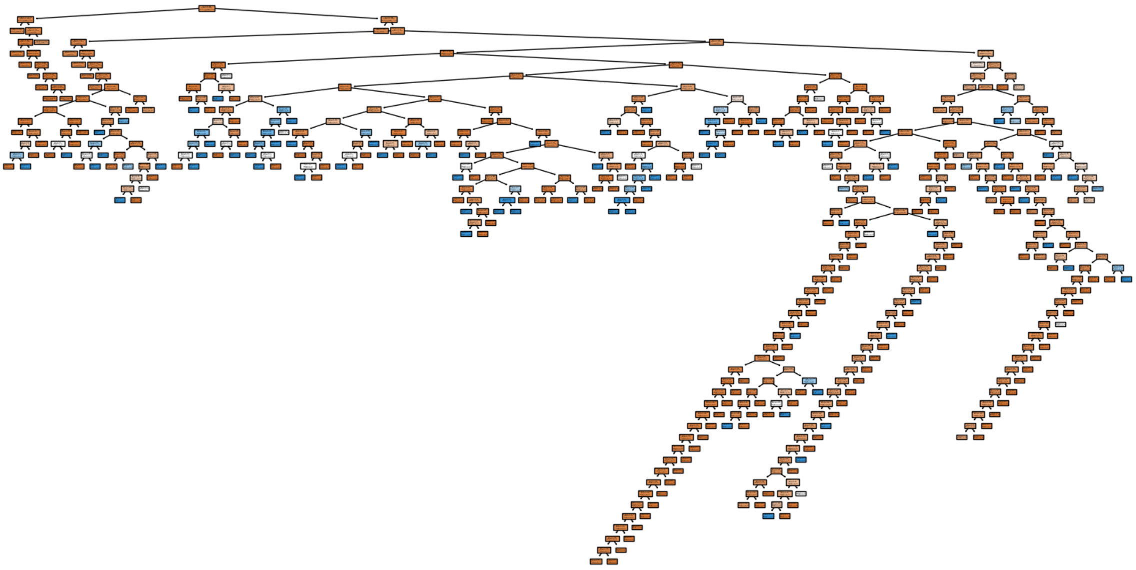
Out[14]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housin
g',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

In [15]:

```
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le
```

In [16]:

```

# Split the dataset into features and target variable
X = data.drop('deposit', axis=1)
y = data['deposit']

# Standardize numerical features
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran

# Initialize the decision tree classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Train the model
dt_classifier.fit(X_train, y_train)

# Feature names and class names
fn = X_train.columns.tolist()
cn = ['no', 'yes']

# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(dt_classifier, feature_names=fn, class_names=cn, filled=True)
plt.show()
```



In [ ]: