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

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import LabelEncoder
```

```
data = pd.read_csv('bank.csv', sep=';')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        4521 non-null   int64 
 1   job        4521 non-null   object
 2   marital    4521 non-null   object
 3   education  4521 non-null   object
 4   default    4521 non-null   object
 5   balance    4521 non-null   int64 
 6   housing    4521 non-null   object
 7   loan       4521 non-null   object
 8   contact    4521 non-null   object
 9   day        4521 non-null   int64 
 10  month      4521 non-null   object
 11  duration   4521 non-null   int64 
 12  campaign   4521 non-null   int64 
 13  pdays      4521 non-null   int64 
 14  previous   4521 non-null   int64 
 15  poutcome   4521 non-null   object
 16  y          4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
```

```
data.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.542579
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.693562
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.000000
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.000000
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.000000
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.000000
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.000000

```
data.head(10)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	pc
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	u
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	u
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	u
5	35	management	single	tertiary	no	747	no	no	cellular	23	feb	141	2	176	3	
6	36	self-employed	married	tertiary	no	307	yes	no	cellular	14	may	341	1	330	2	
7	39	technician	married	secondary	no	147	yes	no	cellular	6	may	151	2	-1	0	u
8	41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may	57	2	-1	0	u

Next steps: Generate code with data View recommended plots New interactive sheet

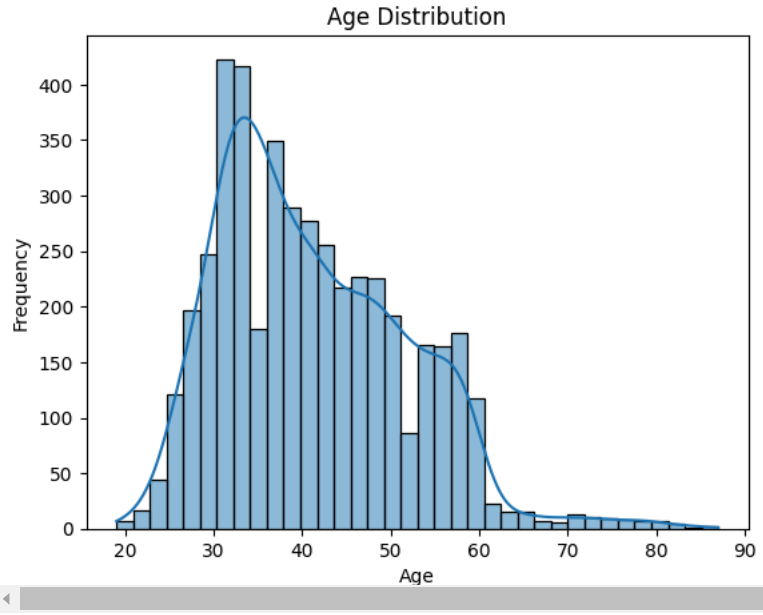
```
data.isnull().sum()
```

```
           0
      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
        y  0

dtype: int64
```

```
data.duplicated().sum()
```

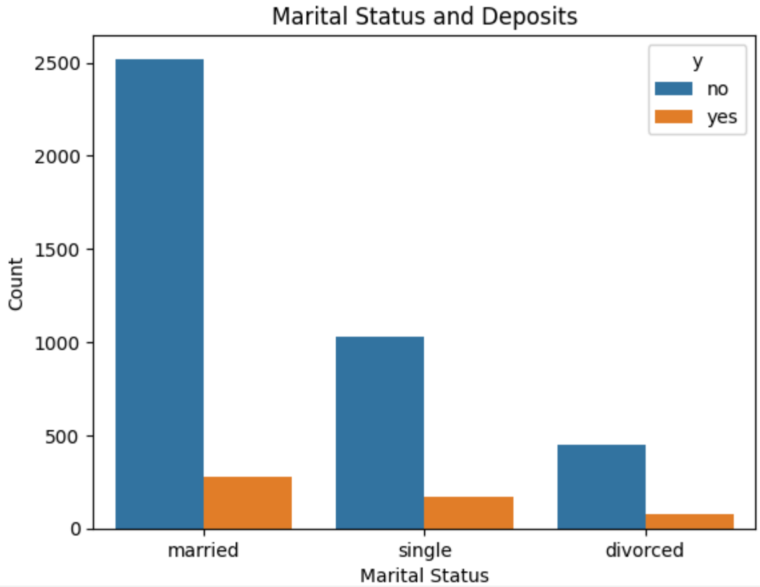
```
0
```

```
sns.histplot(data['age'],kde=True)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



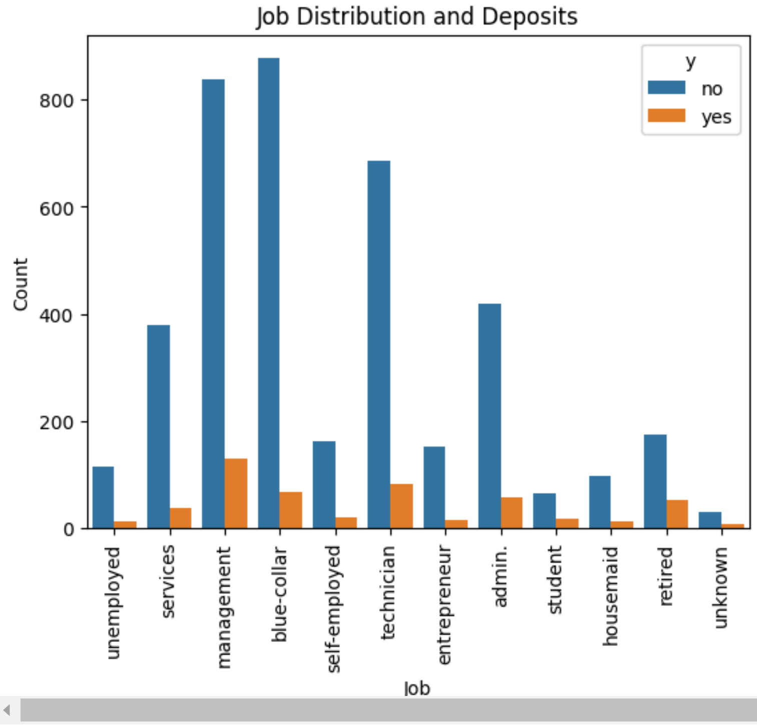
```
sns.countplot(x='marital', data=data, hue='y')
plt.title('Marital Status and Deposits')
plt.xlabel('Marital Status')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



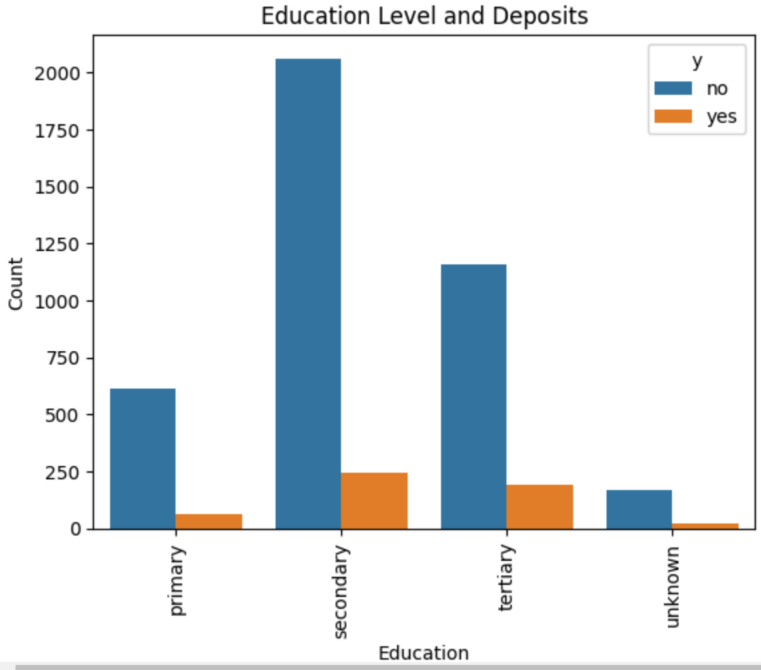
Marital Status and Deposits

```
sns.countplot(x='job', data=data, hue='y')
plt.title('Job Distribution and Deposits')
plt.xlabel('Job')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```



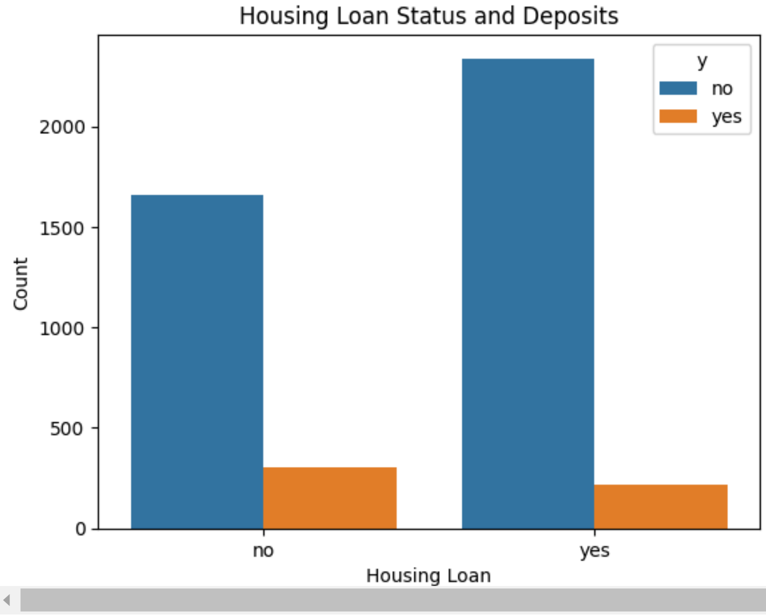
Job Distribution and Deposits

```
sns.countplot(x='education', data=data, hue='y')
plt.title('Education Level and Deposits')
plt.xlabel('Education')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```



```
sns.countplot(x='housing', data=data, hue='y')
plt.title('Housing Loan Status and Deposits')
plt.xlabel('Housing Loan')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



```
cols = data.select_dtypes("object").columns
cols
```

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

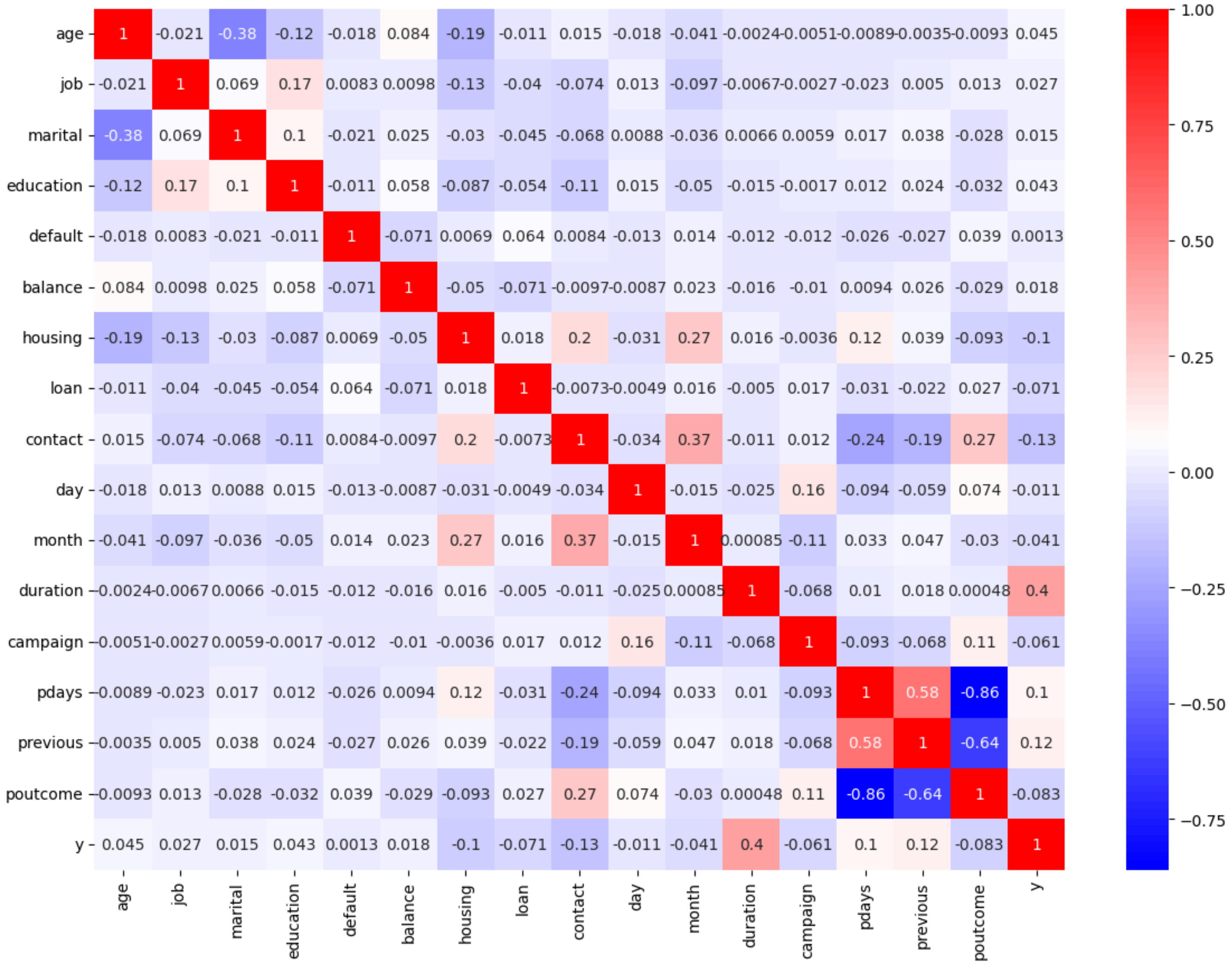
```
LaEn = LabelEncoder()
for col in cols:
    data[col] = LaEn.fit_transform(data[col])
```

```
data.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	30	10	1	0	0	1787	0	0	0	19	10	79	1	-1	0	3
1	33	7	1	1	0	4789	1	1	0	11	8	220	1	339	4	0
2	35	4	2	2	0	1350	1	0	0	16	0	185	1	330	1	0
3	30	4	1	2	0	1476	1	1	2	3	6	199	4	-1	0	3
4	59	1	1	1	0	0	1	0	2	5	8	226	1	-1	0	3

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
plt.figure(figsize=(14,10))
sns.heatmap(data.corr(), cmap='bwr', annot=True)
plt.show()
```



```
X = data.drop('y',axis = 1)
y = data['y']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

dt = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)
dt.fit(X_train, y_train)

y_pred = dt_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8983425414364641
```

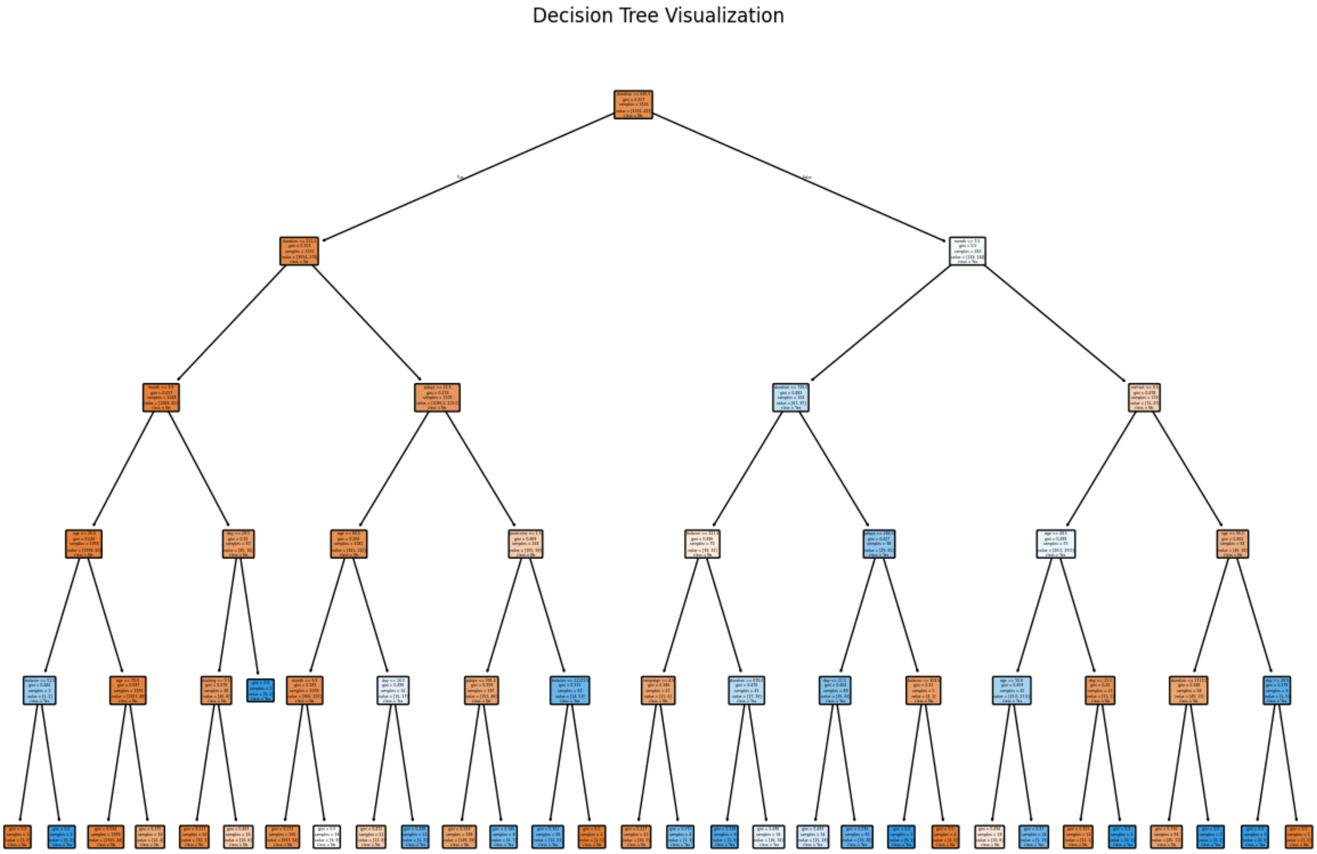
```
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.97      0.94       807
           1       0.55      0.33      0.41        98

    accuracy                           0.90       905
   macro avg       0.74      0.65      0.68       905
weighted avg       0.88      0.90      0.89       905
```

```
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[781  26]
 [ 66  32]]
```

```
plt.figure(figsize=(15, 10))
plot_tree(dt, feature_names=X.columns, class_names=['No', 'Yes'], filled=True, rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```


Decision Tree Visualization