In [1]:
```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [2]:
```
# Load dataset
df = pd.read_csv("bank-full.csv", delimiter=";")  # UCI dataset uses ";" as separator
```

In [3]:
```
# Display dataset info
print("Dataset Overview:\n")
print(df.info())  # Column details
print("\nFirst 5 rows:\n", df.head())  # Preview data
```

```
Dataset Overview:

<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  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
None

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

   contact  day month  duration  campaign  pdays  previous poutcome   y  
0  unknown    5   may       261         1     -1         0  unknown  no  
1  unknown    5   may       151         1     -1         0  unknown  no  
2  unknown    5   may        76         1     -1         0  unknown  no  
3  unknown    5   may        92         1     -1         0  unknown  no  
4  unknown    5   may       198         1     -1         0  unknown  no  
```

In [4]:
```
# ---- EDA (Exploratory Data Analysis) ----
# Check missing values
print("\nMissing Values:\n", df.isnull().sum())
```

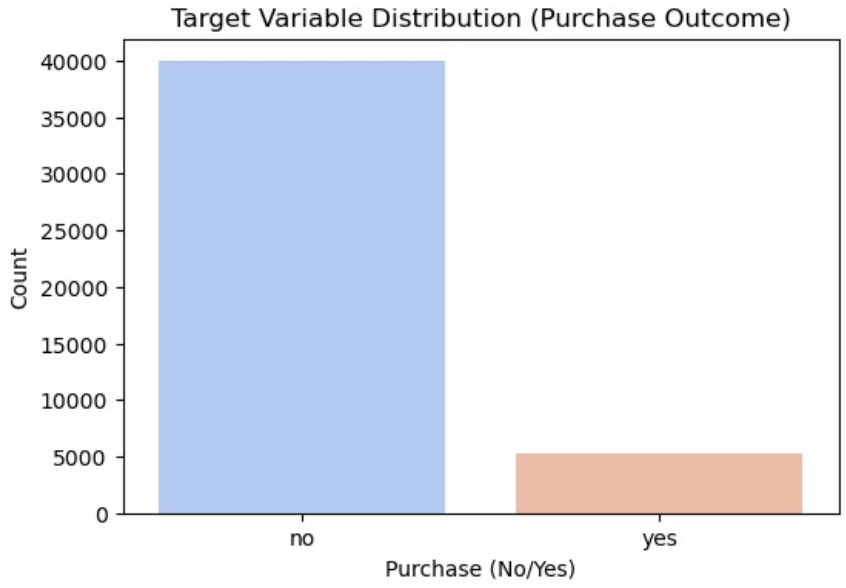
```
Missing Values:
 age          0
job           0
marital       0
education     0
default       0
balance       0
housing       0
loan          0
contact       0
day           0
month         0
duration      0
campaign      0
pdays         0
previous      0
poutcome      0
y             0
dtype: int64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

In [5]:
```
# Summary statistics
print("\nSummary Statistics:\n", df.describe())
```

```
Summary Statistics:
                 age        balance           day      duration      campaign  \
count  45211.000000   45211.000000  45211.000000  45211.000000  45211.000000
mean      40.936210    1362.272058     15.806419    258.163080      2.763841
std       10.618762    3044.765829      8.322476    257.527812      3.098021
min       18.000000   -8019.000000      1.000000      0.000000      1.000000
25%       33.000000      72.000000      8.000000    103.000000      1.000000
50%       39.000000     448.000000     16.000000    180.000000      2.000000
75%       48.000000    1428.000000     21.000000    319.000000      3.000000
max       95.000000  102127.000000     31.000000   4918.000000     63.000000

              pdays      previous
count  45211.000000  45211.000000
mean      40.197828      0.580323
std      100.128746      2.303441
min       -1.000000      0.000000
25%       -1.000000      0.000000
50%       -1.000000      0.000000
75%       -1.000000      0.000000
max      871.000000    275.000000
```

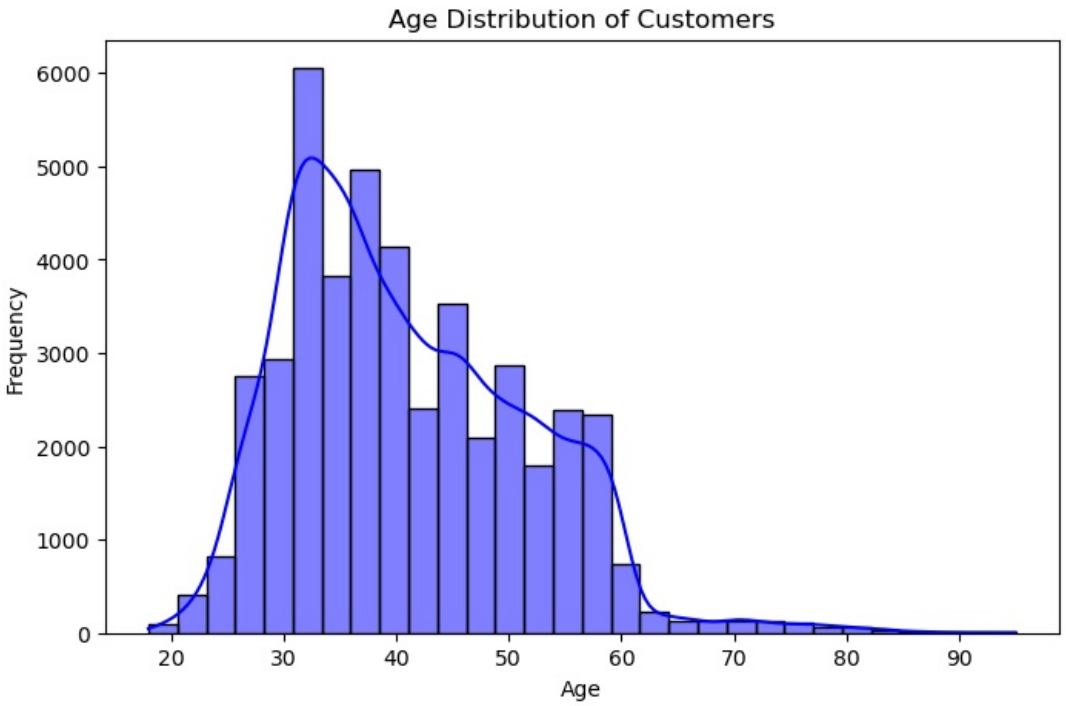
In [6]:
```
# Plot distribution of target variable
plt.figure(figsize=(6, 4))
sns.countplot(x="y", data=df, palette="coolwarm")
plt.title("Target Variable Distribution (Purchase Outcome)")
plt.xlabel("Purchase (No/Yes)")
plt.ylabel("Count")
plt.show()
```

```
/var/folders/r3/trn6xwcx7js3_2j79fkcxz880000gn/T/ipykernel_31171/2933036219.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.countplot(x="y", data=df, palette="coolwarm")
```



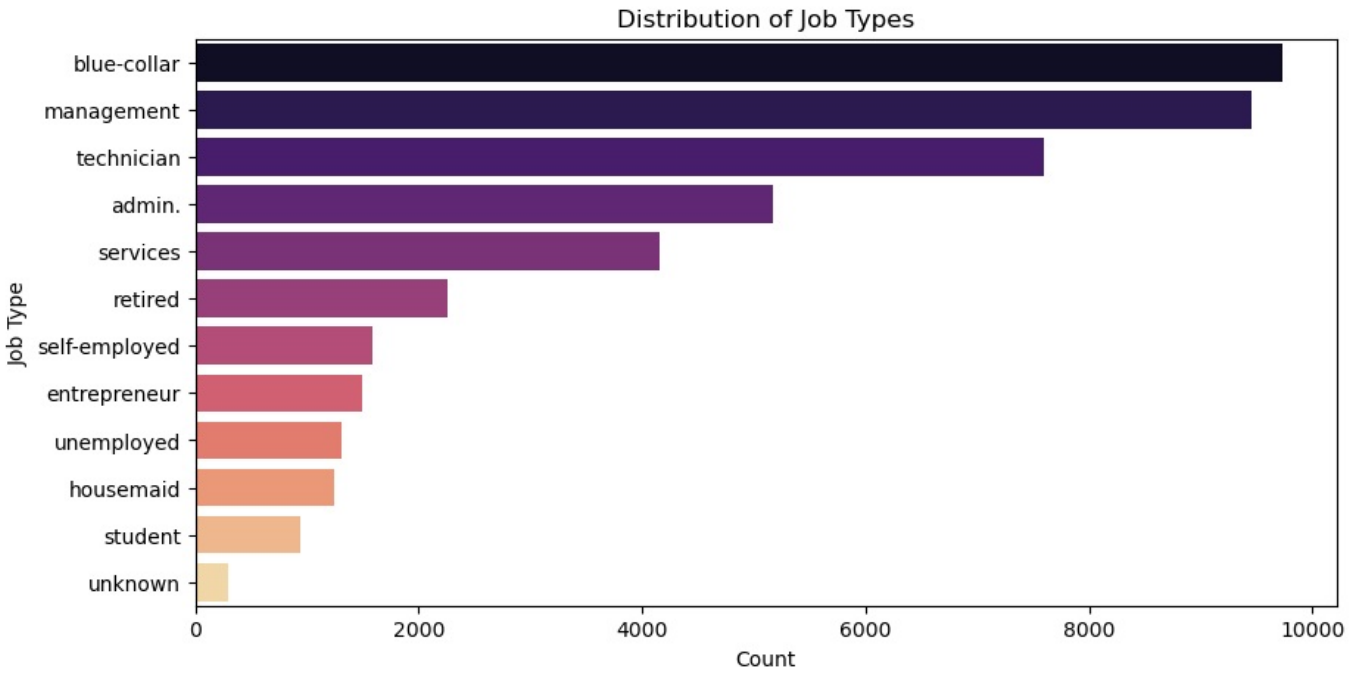
In [7]:
```
# Plot age distribution
plt.figure(figsize=(8, 5))
```

```
sns.histplot(df["age"], bins=30, kde=True, color="blue")
plt.title("Age Distribution of Customers")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```



In [8]:
```
# Plot job type distribution
plt.figure(figsize=(10, 5))
sns.countplot(y="job", data=df, palette="magma", order=df["job"].value_counts().index)
plt.title("Distribution of Job Types")
plt.xlabel("Count")
plt.ylabel("Job Type")
plt.show()
```

```
/var/folders/r3/trn6xwcx7js3_2j79fkcxz880000gn/T/ipykernel_31171/2305566282.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable
to `hue` and set `legend=False` for the same effect.

  sns.countplot(y="job", data=df, palette="magma", order=df["job"].value_counts().index)
```



In [9]:
```
# ---- Data Preprocessing ----

# Encode categorical variables
df_encoded = df.copy()
label_encoders = {}
categorical_columns = df.select_dtypes(include=["object"]).columns
```

In [10]:
```
for col in categorical_columns:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Store encoder for reference
```

In [11]:
```
# Split dataset into features (X) and target (y)
X = df_encoded.drop(columns=["y"])  # Features
y = df_encoded["y"]  # Target variable
```

In [12]:
```
# Split into training (80%) & testing (20%) data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

In [13]:
```
# ---- Train Decision Tree Model ----
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

Out[13]:
```
DecisionTreeClassifier(random_state=42)
```

In [14]:
```
# Predict on test set
y_pred = clf.predict(X_test)
```

In [15]:
```
# ---- Model Evaluation ----

# Print accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.4f}")
```

```
Model Accuracy: 0.8784
```

In [16]:
```
# Print classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

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

           0       0.93      0.93      0.93      7985
           1       0.48      0.49      0.48      1058

    accuracy                           0.88      9043
   macro avg       0.71      0.71      0.71      9043
weighted avg       0.88      0.88      0.88      9043
```

In [17]:
```
# Plot Confusion Matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues", xticklabels=["No", "Yes"], yti
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```