TASK- 3

Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioral data. Use a dataset such as the Bank Marketing dataset from the UCI Machine Learning Repository.

In [2]:

```
#Load the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
#Load the data
df = pd.read_csv('D:\\mlworld\\bank+marketing\\kaggle\\bank.csv')
#View the data
df.head()
df
```

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact	day	m
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	
1	56	admin.	married	secondary	no	45	no	no	unknown	5	
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	
3	55	services	married	secondary	no	2476	yes	no	unknown	5	
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	
...	...	...	...	...	...	...	...	...	...	...	
11157	33	blue-collar	single	primary	no	1	yes	no	cellular	20	
11158	39	services	married	secondary	no	733	no	no	unknown	16	
11159	32	technician	single	secondary	no	29	no	no	cellular	19	
11160	43	technician	married	secondary	no	0	no	yes	cellular	8	
11161	34	technician	married	secondary	no	0	no	no	cellular	9	

11162 rows × 17 columns

In [5]:

```
# Load the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Load the data
df = pd.read_csv('D:\\mlworld\\bank+marketing\\kaggle\\bank.csv')

# Preprocessing
# Drop unnecessary columns
df = df.drop(['day', 'month'], axis=1)

# Encode categorical variables
label_encoder = LabelEncoder()
categorical_cols = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contac
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])

# Define features and target variable
X = df.drop('deposit', axis=1)
y = df['deposit']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Build the Decision Tree model
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_rep)
```

```
Accuracy: 0.7429467084639498
Confusion Matrix:
 [[883 283]
 [291 776]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.76      0.75      1166
           1       0.73      0.73      0.73      1067

    accuracy                           0.74      2233
   macro avg       0.74      0.74      0.74      2233
weighted avg       0.74      0.74      0.74      2233
```

In [ ]: