```
import pandas as pd
df = pd.read_csv('/content/Loan.csv')
df.head()
```

	customer_id	customer_age	customer_income	home_ownership	employment_duration	loan_intent	loan_grade	loan_amnt	loan_int_rat
0	1.0	22	59222	RENT	3.0	PERSONAL	C	35222	16.2
1	2.0	21	9622	OWN	5.0	EDUCATION	A	1222	11.1
2	3.0	25	9622	MORTGAGE	1.0	MEDICAL	B	5522	12.8
3	4.0	23	65522	RENT	4.0	MEDICAL	B	35222	15.2
4	5.0	24	54422	RENT	8.0	MEDICAL	B	35222	14.2

Next steps: Generate code with df | View recommended plots

```
df.shape
```

```
(32586, 13)
```

```
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
```

```
# Separate features and target variable
X = df[['customer_age','loan_int_rate', 'home_ownership', 'loan_intent', 'customer_income','loan_amnt', 'loan_int_rate', 'term_years
y = df['status']
```

```
# Handle nominal variables
nominal_vars = ['home_ownership', 'loan_intent']
le = LabelEncoder()
for var in nominal_vars:
    X[var] = le.fit_transform(X[var])
```

Show hidden output

```
# 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)
```

```
# Create a decision tree classifier
clf = DecisionTreeClassifier()
```

```
# Handle missing values (example using SimpleImputer)
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean') # Or other strategies like 'median', 'most_frequent'
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Train the classifier
clf.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

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

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

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

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

# Calculate precision (specify average='micro' for multiclass)
precision = precision_score(y_test, y_pred, average='micro') # Handle multiclass
print("Precision:", precision)

```

```python
# Calculate recall (specify average='micro' for multiclass)
recall = recall_score(y_test, y_pred, average='micro') # Handle multiclass
print("Recall:", recall)

# Calculate F1-score (specify average='micro' for multiclass)
f1 = f1_score(y_test, y_pred, average='micro') # Handle multiclass
print("F1-score:", f1)
```

```
Accuracy: 0.855170297637312
Precision: 0.855170297637312
Recall: 0.855170297637312
F1-score: 0.855170297637312
```