```
import pandas as pd

#Manualy Extract
# Read CSV file
df = pd.read_csv('csv_3.csv')

# Display the dataset
df
```

	Transaction_Date	Description	Spend_Category	Amount
0	2020-08-17	MY SPICE HOUSE WINNIPEG MB	Retail and Grocery	11.00
1	2020-08-17	REAL CDN. SUPERSTORE # WINNIPEG MB	Retail and Grocery	22.37
2	2020-08-20	MPI BISON SERVICE CENTRE WINNIPEG MB	Professional and Financial Services	25.00
3	2020-08-20	SOBEYS #5037 WINNIPEG MB	Retail and Grocery	15.76
4	2020-08-22	TIM HORTONS #8152 WINNIPEG MB	Restaurants	1.98
...	...	...	...	...
126	2020-11-20	SHOPPERSDRUGMART0532 WINNIPEG MB	Health and Education	10.53
127	2020-11-23	SHOPPERSDRUGMART0532 WINNIPEG MB	Health and Education	41.43
128	2020-11-27	PIZZA PIZZA # 450 WINNIPEG MB	Restaurants	17.91
129	2020-11-27	SHOPPERSDRUGMART0532 WINNIPEG MB	Health and Education	2.84
130	2020-11-29	PIZZA PIZZA # 450 WINNIPEG MB	Restaurants	45.56

131 rows × 4 columns

Next steps: Generate code with df | New interactive sheet

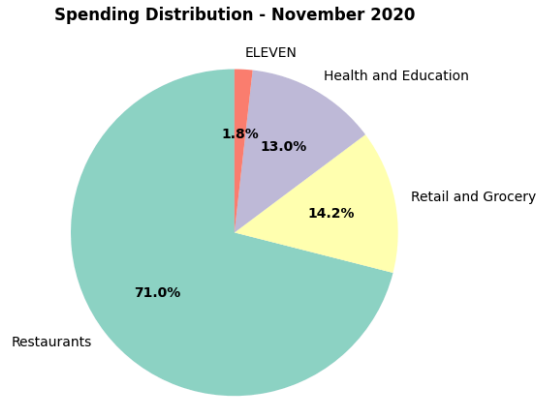
```
import pandas as pd
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display, clear_output

# Load and prepare data
df = pd.read_csv('csv_3.csv')  # Update with your actual filename
df['Transaction_Date'] = pd.to_datetime(df['Transaction_Date'])
df_2020 = df[(df['Transaction_Date'] >= '2020-08-01') & (df['Transaction_Date'] <= '2020-11-30')]

# Create dropdown for month selection
month_options = [('August 2020', '2020-08'),
                 ('September 2020', '2020-09'),
                 ('October 2020', '2020-10'),
                 ('November 2020', '2020-11')]

month_dropdown = widgets.Dropdown(
    options=month_options,
    value='2020-08',
    description='Select Month:',
    style={'description_width': 'initial'}
)

# Output widget for display
output = widgets.Output()

def update_report(change):
    with output:
        clear_output(wait=True)

        selected_month = change['new']
        selected_month_name = [name for name, value in month_options if value == selected_month][0]

        # Filter data
        month_data = df_2020[df_2020['Transaction_Date'].dt.strftime('%Y-%m') == selected_month]

        if month_data.empty:
            print("No data available for selected month")
            return

        # Calculate metrics
```

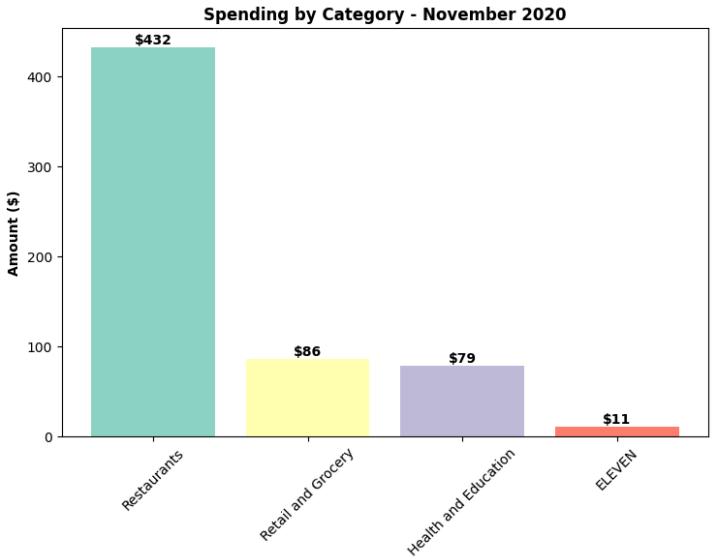
```
            total_spend = month_data['Amount'].sum()
            category_spend = month_data.groupby('Spend_Category')['Amount'].sum().sort_values(ascending=False)

            # Display header
            print(f"{'='*60}")
            print(f"📅 SPEND REPORT FOR {selected_month_name.upper()}")
            print(f"{'='*60}")
            print(f"💰 TOTAL SPEND: ${total_spend:,.2f}")
            print(f"📊 Total Transactions: {len(month_data)}")

            # Create visualization
            fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

            # Pie chart
            colors = plt.cm.Set3(range(len(category_spend)))
            wedges, texts, autotexts = ax1.pie(category_spend.values, labels=category_spend.index,
                                               autopct='%1.1f%%', colors=colors, startangle=90)
            ax1.set_title(f'Spending Distribution - {selected_month_name}', fontweight='bold')

            for autotext in autotexts:
                autotext.set_color('black')
                autotext.set_fontweight('bold')

            # Bar chart
            bars = ax2.bar(category_spend.index, category_spend.values, color=colors)
            ax2.set_title(f'Spending by Category - {selected_month_name}', fontweight='bold')
            ax2.set_ylabel('Amount ($)', fontweight='bold')
            ax2.tick_params(axis='x', rotation=45)

            for bar in bars:
                height = bar.get_height()
                ax2.text(bar.get_x() + bar.get_width()/2., height,
                        f'${height:,.0f}', ha='center', va='bottom', fontweight='bold')

            plt.tight_layout()
            plt.show()

            # Transaction details
            print(f"\n📋 TRANSACTION DETAILS")
            print("-" * 80)
            print(f"{'Date':<12} {'Category':<15} {'Description':<25} {'Amount':>10}")
            print("-" * 80)

            for _, transaction in month_data.sort_values('Transaction_Date').iterrows():
                date_str = transaction['Transaction_Date'].strftime('%m/%d/%Y')
                category = transaction['Spend_Category']
                description = (transaction['Description'][:23] + '...'
                               if len(transaction['Description']) > 25
                               else transaction['Description'])
                amount = transaction['Amount']

                print(f"{date_str:<12} {category:<15} {description:<25} ${amount:>9.2f}")

    # Connect the dropdown to the update function
    month_dropdown.observe(update_report, names='value')

    # Display the widgets
    print("📊 Interactive Monthly Spend Report")
    print("Select a month to view its spending analysis:")
    display(month_dropdown)
    display(output)

    # Trigger initial display
    month_dropdown.value = '2020-08'
```

```
📊 Interactive Monthly Spend Report
Select a month to view its spending analysis:
```

Select Month: November 2020

```
============================================================
📅 SPEND REPORT FOR NOVEMBER 2020
============================================================
💰 TOTAL SPEND: $608.39
📊 Total Transactions: 28
```






```
📋 TRANSACTION DETAILS
--------------------------------------------------------------------------------
```

Date	Category	Description	Amount
11/01/2020	Restaurants	SOUTHLAND RESTAURANT WI...	$ 29.21
11/02/2020	Restaurants	SOUTHLAND RESTAURANT WI...	$ 26.50
11/02/2020	Health and Education	SHOPPERSDRUGMART0532 WI...	$ 3.38
11/04/2020	Restaurants	SOUTHLAND RESTAURANT WI...	$ 25.43
11/04/2020	ELEVEN	7-ELEVEN 17060 D2811 WI...	$ 4.54
11/04/2020	Health and Education	SHOPPERSDRUGMART0532 WI...	$ 12.86
11/05/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 40.30
11/07/2020	Retail and Grocery	REAL CDN. SUPERSTORE # ...	$ 60.36
11/08/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 21.27
11/09/2020	Restaurants	SOUTHLAND RESTAURANT WI...	$ 27.10
11/10/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 22.39
11/10/2020	Health and Education	SHOPPERSDRUGMART0532 WI...	$ 4.57
11/14/2020	Health and Education	SHOPPERSDRUGMART0532 WI...	$ 3.42
11/14/2020	Retail and Grocery	REAL CDN. SUPERSTORE # ...	$ 26.03
11/14/2020	Restaurants	SOUTHLAND RESTAURANT WI...	$ 24.10
11/15/2020	Restaurants	Subway 11980 Winnipeg MB	$ 9.62
11/15/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 22.39
11/17/2020	Restaurants	Subway 11980 Winnipeg MB	$ 9.62
11/17/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 41.41
11/20/2020	Restaurants	SOUTHLAND RESTAURANT WI...	$ 39.70
11/20/2020	ELEVEN	7-ELEVEN 17060 D2811 WI...	$ 4.54
11/20/2020	ELEVEN	7-ELEVEN 17060 D2811 WI...	$ 1.73
11/20/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 29.65
11/20/2020	Health and Education	SHOPPERSDRUGMART0532 WI...	$ 10.53
11/23/2020	Health and Education	SHOPPERSDRUGMART0532 WI...	$ 41.43
11/27/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 17.91
11/27/2020	Health and Education	SHOPPERSDRUGMART0532 WI...	$ 2.84
11/29/2020	Restaurants	PIZZA PIZZA # 450 WINNI...	$ 45.56

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score

# Load the CSV file
df = pd.read_csv('/content/csv_3.csv')

print("📊 DATASET OVERVIEW")
```

```
print("=" * 50)
print(f"Dataset shape: {df.shape}")
print(f"Columns: {list(df.columns)}")
print("\nFirst 5 rows:")
print(df.head())
print("\n" + "=" * 50)

# Data Preprocessing
print("\n🔄 DATA PREPROCESSING...")

# Convert Transaction_Date to datetime and extract features
df['Transaction_Date'] = pd.to_datetime(df['Transaction_Date'])
df['Day'] = df['Transaction_Date'].dt.day
df['Month'] = df['Transaction_Date'].dt.month
df['DayOfWeek'] = df['Transaction_Date'].dt.dayofweek
df['DayOfYear'] = df['Transaction_Date'].dt.dayofyear

# Encode categorical variables
label_encoders = {}

# Encode Description
le_description = LabelEncoder()
df['Description_encoded'] = le_description.fit_transform(df['Description'])
label_encoders['Description'] = le_description

# Encode Spend_Category
le_category = LabelEncoder()
df['Spend_Category_encoded'] = le_category.fit_transform(df['Spend_Category'])
label_encoders['Spend_Category'] = le_category

print(f"Unique Descriptions: {len(df['Description'].unique())}")
print(f"Unique Categories: {len(df['Spend_Category'].unique())}")

# Prepare features and target
feature_columns = ['Day', 'Month', 'DayOfWeek', 'DayOfYear', 'Description_encoded', 'Spend_Category_encoded']
X = df[feature_columns]
y = df['Amount']

print(f"\nFeatures used: {feature_columns}")
print(f"Target variable: Amount")

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"\nTraining set: {X_train.shape[0]} samples")
print(f"Testing set: {X_test.shape[0]} samples")

# Train the model
print("\n🌲 TRAINING RANDOM FOREST REGRESSOR...")
model = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=10)
model.fit(X_train, y_train)

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

# Evaluate the model
print("\n📈 MODEL EVALUATION")
print("=" * 30)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error: ${mae:.2f}")
print(f"R² Score: {r2:.3f}")
print(f"Average prediction error: {mae/df['Amount'].mean()*100:.1f}% of average transaction amount")

# Generate 5 new sample rows for prediction
print("\n" + "=" * 60)
print("🎯 GENERATING PREDICTIONS FOR 5 NEW SAMPLES")
print("=" * 60)

# Create 5 new realistic sample transactions based on the data patterns
np.random.seed(42)

new_samples = []
for i in range(5):
    sample = {
        'Transaction_Date': f"2020-{np.random.randint(8, 12):02d}-{np.random.randint(1, 28):02d}",
        'Description': np.random.choice(df['Description'].unique())
```

```
            'Description': np.random.choice(df['Description'].unique()),
            'Spend_Category': np.random.choice(df['Spend_Category'].unique())
        }
        new_samples.append(sample)

    # Convert to DataFrame
    new_df = pd.DataFrame(new_samples)
    new_df['Transaction_Date'] = pd.to_datetime(new_df['Transaction_Date'])

    print("\n📋 GENERATED NEW SAMPLES:")
    print(new_df[['Transaction_Date', 'Description', 'Spend_Category']])

    # Preprocess new samples for prediction
    new_df['Day'] = new_df['Transaction_Date'].dt.day
    new_df['Month'] = new_df['Transaction_Date'].dt.month
    new_df['DayOfWeek'] = new_df['Transaction_Date'].dt.dayofweek
    new_df['DayOfYear'] = new_df['Transaction_Date'].dt.dayofyear

    # Encode categorical variables
    new_df['Description_encoded'] = new_df['Description'].apply(
        lambda x: le_description.transform([x])[0] if x in le_description.classes_ else -1
    )
    new_df['Spend_Category_encoded'] = new_df['Spend_Category'].apply(
        lambda x: le_category.transform([x])[0] if x in le_category.classes_ else -1
    )

    # Prepare features for prediction
    X_new = new_df[feature_columns]

    # Make predictions
    predictions = model.predict(X_new)

    # Display results
    print("\n" + "=" * 80)
    print("🎯 PREDICTION RESULTS")
    print("=" * 80)
    print(f"{'#':<2} {'Date':<12} {'Description':<30} {'Category':<25} {'Predicted Amount':<15}")
    print("-" * 80)

    for i, (idx, row) in enumerate(new_df.iterrows()):
        print(f"{i+1:<2} {row['Transaction_Date'].strftime('%Y-%m-%d'):<12} "
              f"{row['Description'][:28]:<30} "
              f"{row['Spend_Category']:<25} "
              f"${predictions[i]:.2f}")

    print("-" * 80)
    print(f"💰 Total Predicted Spend for 5 transactions: ${predictions.sum():.2f}")

    # Feature importance analysis
    print("\n🔍 FEATURE IMPORTANCE")
    print("=" * 30)
    feature_importance = pd.DataFrame({
        'Feature': feature_columns,
        'Importance': model.feature_importances_
    }).sort_values('Importance', ascending=False)

    print(feature_importance)

    # Show actual data statistics for comparison
    print(f"\n📊 ACTUAL DATA STATISTICS (for reference)")
    print("=" * 40)
    print(f"Total transactions in dataset: {len(df)}")
    print(f"Date range: {df['Transaction_Date'].min().strftime('%Y-%m-%d')} to {df['Transaction_Date'].max().strftime('%Y-%m-%
    print(f"Average transaction amount: ${df['Amount'].mean():.2f}")
    print(f"Minimum transaction amount: ${df['Amount'].min():.2f}")
    print(f"Maximum transaction amount: ${df['Amount'].max():.2f}")
    print(f"Most common category: {df['Spend_Category'].mode().values[0]}")
    print(f"Most frequent merchant: {df['Description'].mode().values[0]}")

    # Show some actual examples for comparison
    print(f"\n📝 ACTUAL TRANSACTION EXAMPLES")
    print("=" * 40)
    sample_actual = df.sample(3)
    for _, row in sample_actual.iterrows():
        print(f"Date: {row['Transaction_Date'].strftime('%Y-%m-%d')}, "
              f"Merchant: {row['Description'][:25]}, "
              f"Category: {row['Spend_Category']}, "
              f"Actual Amount: ${row['Amount']:.2f}")
```

```
============================================================
🎯 GENERATING PREDICTIONS FOR 5 NEW SAMPLES
============================================================

📋 GENERATED NEW SAMPLES:
  Transaction_Date                        Description  \
0       2020-10-20         DOLLARAMA #1286 WINNIPEG MB
1       2020-11-21       TIM HORTONS #2856 WINNIPEG MB
2       2020-10-23               BK #11895 WINNIPEG MB
3       2020-11-21            SOBEYS #5037 WINNIPEG MB
4       2020-09-21  REAL CDN. SUPERSTORE # WINNIPEG MB

                       Spend_Category
0                         Restaurants
1  Professional and Financial Services
2                         Restaurants
3                         Restaurants
4                 Health and Education

================================================================================
🎯 PREDICTION RESULTS
================================================================================
#  Date         Description                    Category                  Predicted Amount
--------------------------------------------------------------------------------
1  2020-10-20   DOLLARAMA #1286 WINNIPEG MB    Restaurants               $12.40
2  2020-11-21   TIM HORTONS #2856 WINNIPEG M   Professional and Financial Services $12.89
3  2020-10-23   BK #11895 WINNIPEG MB          Restaurants               $13.19
4  2020-11-21   SOBEYS #5037 WINNIPEG MB       Restaurants               $23.68
5  2020-09-21   REAL CDN. SUPERSTORE # WINNI   Health and Education      $13.09
--------------------------------------------------------------------------------
💰 Total Predicted Spend for 5 transactions: $75.24

🔍 FEATURE IMPORTANCE
==============================
                 Feature  Importance
3              DayOfYear    0.316359
4    Description_encoded    0.278969
5  Spend_Category_encoded    0.265727
0                    Day    0.088144
2              DayOfWeek    0.027663
1                  Month    0.023137

📊 ACTUAL DATA STATISTICS (for reference)
========================================
Total transactions in dataset: 131
Date range: 2020-08-17 to 2020-11-29
Average transaction amount: $14.32
Minimum transaction amount: $0.65
Maximum transaction amount: $298.00
Most common category: Restaurants
Most frequent merchant: TIM HORTONS #8152 WINNIPEG MB

📝 ACTUAL TRANSACTION EXAMPLES
========================================
Date: 2020-11-20, Merchant: PIZZA PIZZA # 450 WINNIPE, Category: Restaurants, Actual Amount: $29.65
Date: 2020-10-07, Merchant: TIM HORTONS #8152 WINNIPE, Category: Restaurants, Actual Amount: $6.49
Date: 2020-10-14, Merchant: BK #11895 WINNIPEG MB, Category: Restaurants, Actual Amount: $8.95
```

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# Load the CSV file
df = pd.read_csv('/content/csv_3.csv')

print("📊 DATASET OVERVIEW")
print("=" * 50)
print(f"Dataset shape: {df.shape}")
print(f"Columns: {list(df.columns)}")
print("\nFirst 5 rows:")
print(df.head())
print("\n" + "=" * 50)

# Data Preprocessing
print("\n🔄 DATA PREPROCESSING...")

# Convert Transaction_Date to datetime and extract features
```

```
df['Transaction_Date'] = pd.to_datetime(df['Transaction_Date'])
df['Day'] = df['Transaction_Date'].dt.day
df['Month'] = df['Transaction_Date'].dt.month
df['DayOfWeek'] = df['Transaction_Date'].dt.dayofweek
df['DayOfYear'] = df['Transaction_Date'].dt.dayofyear

# Encode categorical variables
label_encoders = {}

# Encode Description
le_description = LabelEncoder()
df['Description_encoded'] = le_description.fit_transform(df['Description'])
label_encoders['Description'] = le_description

# Prepare features and target
feature_columns = ['Day', 'Month', 'DayOfWeek', 'DayOfYear', 'Description_encoded', 'Amount']
X = df[feature_columns]
y = df['Spend_Category']  # This is our target for classification

print(f"Unique Categories: {df['Spend_Category'].unique()}")
print(f"Category distribution:")
print(df['Spend_Category'].value_counts())
print(f"\nFeatures used: {feature_columns}")
print(f"Target variable: Spend_Category")

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

print(f"\nTraining set: {X_train.shape[0]} samples")
print(f"Testing set: {X_test.shape[0]} samples")

# Train the model
print("\n🌲 TRAINING RANDOM FOREST CLASSIFIER...")
model = RandomForestClassifier(n_estimators=100, random_state=42, max_depth=10)
model.fit(X_train, y_train)

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

# Evaluate the model
print("\n📈 MODEL EVALUATION")
print("=" * 30)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.3f} ({accuracy*100:.1f}%)")
print(f"\nClassification Report:")
print(classification_report(y_test, y_pred))

# Generate 5 new sample rows for prediction
print("\n" + "=" * 60)
print("🎯 GENERATING PREDICTIONS FOR 5 NEW SAMPLES")
print("=" * 60)

# Create 5 new realistic sample transactions
np.random.seed(42)

new_samples = []
for i in range(5):
    sample = {
        'Transaction_Date': f"2020-{np.random.randint(8, 12):02d}-{np.random.randint(1, 28):02d}",
        'Description': np.random.choice(df['Description'].unique()),
        'Amount': np.random.uniform(1, 50)  # Random amount between $1 and $50
    }
    new_samples.append(sample)

# Convert to DataFrame
new_df = pd.DataFrame(new_samples)
new_df['Transaction_Date'] = pd.to_datetime(new_df['Transaction_Date'])

print("\n📋 GENERATED NEW SAMPLES:")
print(new_df[['Transaction_Date', 'Description', 'Amount']])

# Preprocess new samples for prediction
new_df['Day'] = new_df['Transaction_Date'].dt.day
new_df['Month'] = new_df['Transaction_Date'].dt.month
new_df['DayOfWeek'] = new_df['Transaction_Date'].dt.dayofweek
new_df['DayOfYear'] = new_df['Transaction_Date'].dt.dayofyear
```

```
# Encode Description
new_df['Description_encoded'] = new_df['Description'].apply(
    lambda x: le_description.transform([x])[0] if x in le_description.classes_ else -1
)

# Prepare features for prediction
X_new = new_df[feature_columns]

# Make predictions
category_predictions = model.predict(X_new)
prediction_probabilities = model.predict_proba(X_new)

# Display results
print("\n" + "=" * 80)
print("🎯 PREDICTION RESULTS")
print("=" * 80)
print(f"{'#':<2} {'Date':<12} {'Description':<30} {'Amount':<10} {'Predicted Category':<25} {'Confidence':<10}")
print("-" * 80)

for i, (idx, row) in enumerate(new_df.iterrows()):
    predicted_category = category_predictions[i]
    confidence = np.max(prediction_probabilities[i]) * 100

    print(f"{i+1:<2} {row['Transaction_Date'].strftime('%Y-%m-%d'):<12} "
          f"{row['Description'][:28]:<30} "
          f"${row['Amount']:<9.2f} "
          f"{predicted_category:<25} "
          f"{confidence:.1f}%")

print("-" * 80)

# Feature importance analysis
print("\n🔍 FEATURE IMPORTANCE")
print("=" * 30)
feature_importance = pd.DataFrame({
    'Feature': feature_columns,
    'Importance': model.feature_importances_
}).sort_values('Importance', ascending=False)

print(feature_importance)

# Show actual category distribution for comparison
print(f"\n📊 ACTUAL CATEGORY DISTRIBUTION")
print("=" * 40)
category_counts = df['Spend_Category'].value_counts()
for category, count in category_counts.items():
    percentage = (count / len(df)) * 100
    print(f"{category}: {count} transactions ({percentage:.1f}%)")

# Show some prediction probabilities for the first sample
print(f"\n🔮 DETAILED PROBABILITIES FOR SAMPLE 1")
print("=" * 40)
categories = model.classes_
probabilities = prediction_probabilities[0] * 100

for category, prob in zip(categories, probabilities):
    print(f"{category}: {prob:.1f}%")
```

```
📊 DATASET OVERVIEW
==================================================
Dataset shape: (131, 4)
Columns: ['Transaction_Date', 'Description', 'Spend_Category', 'Amount']

First 5 rows:
  Transaction_Date                          Description  \
0       2020-08-17           MY SPICE HOUSE WINNIPEG MB
1       2020-08-17    REAL CDN. SUPERSTORE # WINNIPEG MB
2       2020-08-20  MPI BISON SERVICE CENTRE WINNIPEG MB
3       2020-08-20              SOBEYS #5037 WINNIPEG MB
4       2020-08-22         TIM HORTONS #8152 WINNIPEG MB

                      Spend_Category  Amount
0                  Retail and Grocery   11.00
1                  Retail and Grocery   22.37
2  Professional and Financial Services   25.00
3                  Retail and Grocery   15.76
4                         Restaurants    1.98

==================================================
```

```
🔄 DATA PREPROCESSING...
Unique Categories: ['Retail and Grocery' 'Professional and Financial Services' 'Restaurants'
 'Health and Education' 'Personal and Household Expenses' 'ELEVEN']
Category distribution:
Spend_Category
Restaurants                          81
Health and Education                 18
Retail and Grocery                   16
ELEVEN                               12
Professional and Financial Services   2
Personal and Household Expenses       2
Name: count, dtype: int64

Features used: ['Day', 'Month', 'DayOfWeek', 'DayOfYear', 'Description_encoded', 'Amount']
Target variable: Spend_Category

Training set: 104 samples
Testing set: 27 samples

🌲 TRAINING RANDOM FOREST CLASSIFIER...

📈 MODEL EVALUATION
==============================
Accuracy: 1.000 (100.0%)

Classification Report:
                     precision    recall  f1-score   support

             ELEVEN       1.00      1.00      1.00         3
Health and Education       1.00      1.00      1.00         4
         Restaurants       1.00      1.00      1.00        17
  Retail and Grocery       1.00      1.00      1.00         3

           accuracy                           1.00        27
          macro avg       1.00      1.00      1.00        27
       weighted avg       1.00      1.00      1.00        27
```