```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from transformers import pipeline
import ipywidgets as widgets
from IPython.display import display, HTML


# Download NLTK data
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /
True
```

```
# Step 2: Data Upload and Extraction
from google.colab import files

# Upload CSV file
df = pd.read_csv('personal_transactions.csv')

# Display the first few rows of the dataframe
df.head()
```

	Date	Description	Amount	Transaction Type	Cate
0	01-01-2018	Amazon	11.11	debit	Shop
1	01-02-2018	Mortgage Payment	1247.44	debit	Mortga
2	01-02-2018	Thai Restaurant	24.22	debit	Restau
3	01-03-	Credit Card	2298.09	credit	Credit

Next steps: Generate code with df recommended

monthly_summary.csv

1 to 3 of 3 entries Filter

Month	Total_Amount	Total_Tra
2018-01	5875.860000000001	14
2018-02	4157.66	16
2018-03	5143.02	19

Show 25 per page

```
# Step 3: Data Preprocessing
# Handle missing values
df.fillna('', inplace=True)

# Convert date to datetime format
df['Date'] = pd.to_datetime(df['Date'], format='%m-%d-

# Display basic statistics
df.describe()
```

	Date	Amount
count	49	100.000000
mean	2018-02-08 04:24:29.387755008	271.298100
min	2018-01-01 00:00:00	2.000000
25%	2018-01-11 00:00:00	16.880000
50%	2018-02-09 00:00:00	35.475000
75%	2018-03-04 00:00:00	116.385000
max	2018-03-12 00:00:00	2298.090000
std	NaN	555.927068

```
# Step 4: NLP Analysis
from nltk.sentiment.vader import SentimentIntensityAna

# Initialize the sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Analyze sentiment of transaction descriptions
df['Sentiment'] = df['Description'].apply(lambda x: si

# Display the dataframe with sentiment scores
df.head()
```

	Date	Description	Amount	Transaction Type	Cat
0	2018-01-01	Amazon	11.11	debit	Sho
1	2018-01-02	Mortgage Payment	1247.44	debit	Mortg
2	2018-01-02	Thai Restaurant	24.22	debit	Restau
3	2018-01-03	Credit Card Payment	2298.09	credit	Credit Pay
4	2018-01-04	Netflix	11.76	debit	Mov

Next steps: Generate code with df recommended

```
# Step 5: Model Training
# Initialize a text generation pipeline
# Use a pipeline as a high-level helper
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("text-generation", model="TurkuNLP/gpt
```

```
/usr/local/lib/python3.10/dist-packages/huggingfac
The secret `HF_TOKEN` does not exist in your Colab
To authenticate with the Hugging Face Hub, create
You will be able to reuse this secret in all of yo
Please note that authentication is recommended but
  warnings.warn(
config.json: 100%  561/561 [00:00<00:00, 20.0kB/s]
pytorch_model.bin: 100%  743M/743M [00:07<00:00, 146MB/s
tokenizer_config.json: 100%  218/218 [00:00<00:00, 10.3kB/s]
tokenizer.json: 100%  6.23M/6.23M [00:00<00:00, 17.1MB/s]
```

```
# Example: Generate a report summary
example_summary = pipe("Generate a financial report su
                       max_length=200,
                       num_return_sequences=1,
                       truncation=True,
                       pad_token_id=pipe.tokenizer.eos

print(example_summary[0]['generated_text'])
```

```
Generate a financial report summary based on the f
```

```
# Step 6: Report Generation
def generate_financial_report(transactions):
    summary_prompt = "Generate a financial report summ
    for index, row in transactions.iterrows():
        summary_prompt += f"{row['Description']}: ${ro

    summary = pipe(summary_prompt, max_length=1000, nu
    return summary[0]['generated_text']

# Generate the report
financial_report = generate_financial_report(df)
print(financial_report)
```

```
Generate a financial report summary based on the f
Amazon: $11.11, Mortgage Payment: $1247.44, Thai F
Amazon: $10.99, Amazon: $10.99, Amazon: $10.99, An
```

```
import io  # Import the io module
import pandas as pd


##Monthly transactions
def create_monthly_summary(transactions):
    # Extract month and year from the Date column
    transactions['Month'] = transactions['Date'].dt.to

    # Group by Month and calculate the sum of Amount f
    monthly_summary = transactions.groupby('Month').ag
        Total_Amount=('Amount', 'sum'),
        Total_Transactions=('Amount', 'count')
    ).reset_index()

    # Save the monthly summary to a new CSV file
    monthly_summary.to_csv('monthly_summary.csv', inde

    return monthly_summary


monthly_summary = create_monthly_summary(df)
print(monthly_summary)
```

```
     Month  Total_Amount  Total_Transactions
0  2018-01       5875.86                  14
1  2018-02       4157.66                  16
2  2018-03       5143.02                  19
```

```
# Step 7: User Interface
# Upload CSV file widget
upload_widget = widgets.FileUpload(accept='.csv', multi
```

```
# Output widget for displaying the report
output_widget = widgets.Output()

# Button to generate the report
generate_button = widgets.Button(description="Generate

def on_generate_button_clicked(b):
    with output_widget:
        output_widget.clear_output()

        # Read the uploaded CSV file
        uploaded_file = upload_widget.value
        if uploaded_file:
            csv_file = list(uploaded_file.values())[0][
            df = pd.read_csv(io.StringIO(csv_file.decod


            # Preprocess and analyze the data
            df.fillna('', inplace=True)
            df['Date'] = pd.to_datetime(df['Date'], for
            df['Sentiment'] = df['Description'].apply(l

            # Generate the financial report
            report = generate_financial_report(df)
            print(report)

            # Create the monthly summary
            monthly_summary = create_monthly_summary(df
            print(monthly_summary)
            print("Monthly summary saved to 'monthly_su
        else:
            print("Please upload a CSV file.")

generate_button.on_click(on_generate_button_clicked)

# Display the widgets
display(HTML("<h2>Upload your transaction CSV file</h2>
display(upload_widget)
display(generate_button)
display(output_widget)
```

Upload your transaction CSV file

Upload (1)

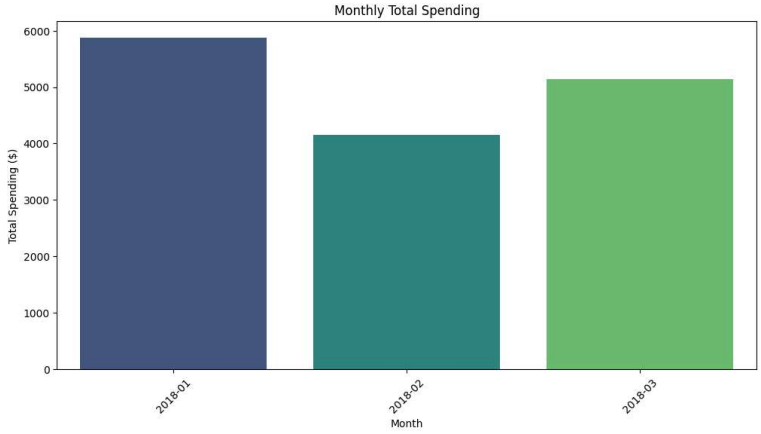
Generate Report

```
Generate a financial report summary based on the
following transactions:
Amazon: $11.11, Mortgage Payment: $1247.44, Thai
Restaurant: $24.22, Credit Card Payment:
$2298.09, Netflix: $11.76, American Tavern:
$25.85, Hardware Store: $18.45, Gas Company:
$45.0, Hardware Store: $15.38, Spotify: $10.69,
Phone Company: $89.46, Shell: $34.87, Grocery
Store: $43.54, Biweekly Paycheck: $2000.0, Pizza
Place: $32.91, Amazon: $39.05, Grocery Store:
$44.19, American Tavern: $64.11, City Water
Charges: $35.0, Power Company: $60.0, Biweekly
Paycheck: $2000.0, Amazon: $50.21, Credit Card
Payment: $554.99, Credit Card Payment: $309.81,
Credit Card Payment: $554.99, Hardware Store:
$17.38, Credit Card Payment: $309.81, Starbucks:
$3.0, Internet Service Provider: $69.99, Shell:
$30.42, Thai Restaurant: $25.0, Brunch
Restaurant: $17.62, Grocery Store: $27.79,
Amazon: $11.11, Mortgage Payment: $1247.44,
Biweekly Paycheck: $2000.0, Japanese Restaurant:
$57.02, Netflix: $11.76, Credit Card Payment:
$145.14, Credit Card Payment: $154.13, Credit
Card Payment: $154.13, Gas Company: $65.0,
Barbershop: $30.0, Spotify: $10.69, Bojangles:
$10.66, Fancy Restaurant: $106.8, Shell: $36.47,
Phone Company: $89.52, Brewing Company: $14.0,
American Tavern: $10.0, Power Company: $60.0,
Biweekly Paycheck: $2000.0, Brunch Restaurant:
$8.0, City Water Charges: $35.0, Grocery Store:
$35.95, Mexican Restaurant: $23.51, Starbucks:
$2.0, Starbucks: $4.0, Credit Card Payment:
$765.37, Credit Card Payment: $156.11, Credit
Card Payment: $765.37, Internet Service Provider:
$74.99, American Tavern: $85.52, Gas Station:
$32.21, Credit Card Payment: $156.11, Grocery
```

Data Visualization

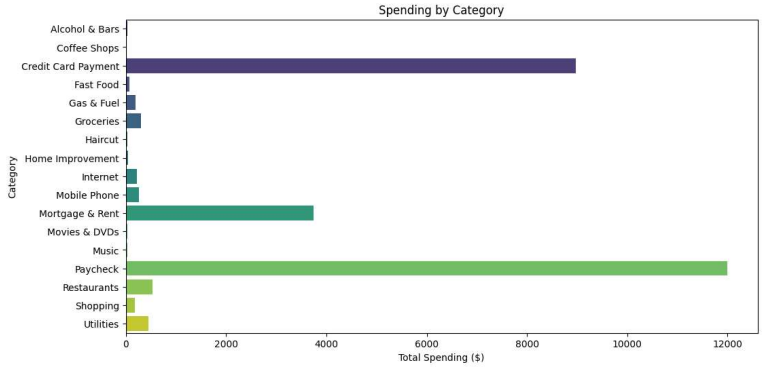
```
# Visualization 1: Monthly Total Spending
import matplotlib.pyplot as plt
import seaborn as sns

def plot_monthly_total_spending(transactions):
    transactions['Month'] = transactions['Date'].dt.to
    monthly_total = transactions.groupby('Month').agg(
    monthly_total['Month'] = monthly_total['Month'].as

    plt.figure(figsize=(12, 6))
    sns.barplot(x='Month', y='Total_Spending', data=mo
    plt.title('Monthly Total Spending')
    plt.xlabel('Month')
    plt.ylabel('Total Spending ($)')
    plt.xticks(rotation=45)
    plt.show()

plot_monthly_total_spending(df)
```

```
<ipython-input-13-acf0870b391d>:11: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x='Month', y='Total_Spending', data=
```

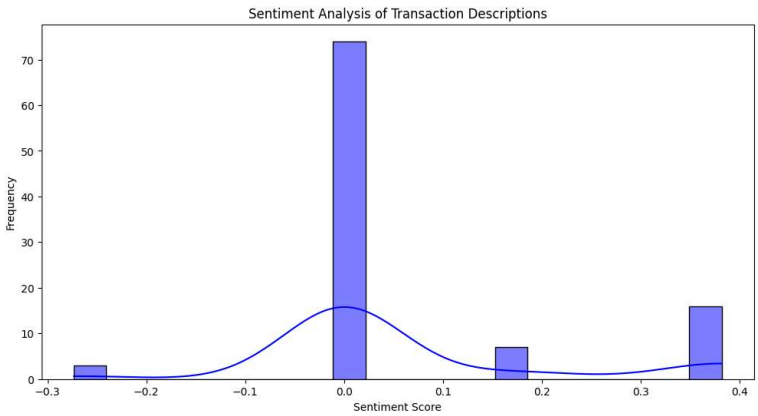


```
# Visualization 2: Spending by Category
def plot_spending_by_category(transactions):
    category_total = transactions.groupby('Category').

    plt.figure(figsize=(12, 6))
    sns.barplot(x='Total_Spending', y='Category', data
    plt.title('Spending by Category')
    plt.xlabel('Total Spending ($)')
    plt.ylabel('Category')
    plt.show()

plot_spending_by_category(df)
```

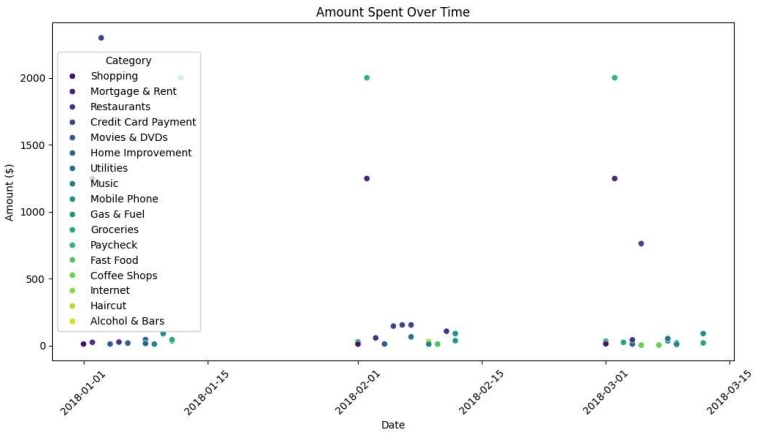
```
<ipython-input-16-6116550518c7>:6: FutureWarning:

Passing `palette` without assigning `hue` is depre

  sns.barplot(x='Total_Spending', y='Category', da
```




```
# Visualization 3: Sentiment Analysis of Transaction D
def plot_sentiment_analysis(transactions):
    plt.figure(figsize=(12, 6))
    sns.histplot(transactions['Sentiment'], bins=20, k
    plt.title('Sentiment Analysis of Transaction Descr
    plt.xlabel('Sentiment Score')
    plt.ylabel('Frequency')
    plt.show()

plot_sentiment_analysis(df)
```




```
# Visualization 4: Scatter Plot of Amount Spent Over T
def plot_amount_over_time(transactions):
    plt.figure(figsize=(12, 6))
    sns.scatterplot(x='Date', y='Amount', data=transac
    plt.title('Amount Spent Over Time')
    plt.xlabel('Date')
    plt.ylabel('Amount ($)')
    plt.xticks(rotation=45)
    plt.show()

plot_amount_over_time(df)
```



```
# Visualization 5: Pie Chart of Spending by Category
def plot_pie_chart_by_category(transactions):
```