Data Processing in Python - README

Overview

This project aims to generate synthetic e commerce order transaction data and create a Streamlit dashboard for visualizing and analyzing the data. The dashboard will allow users to inspect the CSV file, visualize daily profit/loss, view popular products, and filter data based on various fields. Additionally, a bonus dashboard will help identify potential fraudulent orders.

Table of Contents

Requirements

- Python 3.7 or higher
- Required Python libraries:
  - pandas
  - numpy
  - faker
  - streamlit
  - matplotlib
  - seaborn

Installation

- To set up the project, follow these steps:
- Clone the repository:
  git clone https://github.com/yourusername/ecommerce-data-processing.git
- cd ecommerce-data-processing
- Create a virtual environment (optional but recommended):
  python -m venv venv
- source venv/bin/activate # On Windows use `venv\Scripts\activate`
  Install the required dependencies:

  pip install -r requirements.txt

  This script will create a CSV file named **ecommerce_orders.csv** with at least 100,000 rows containing the following fields:
- order_id
- customer_name
- total_price
- total_discount
- product_name
- coupon_code
- cost_price (to calculate profit/loss)

Running the Streamlit App

- To run the Streamlit dashboard, execute the following command:
  streamlit run app.py

  This will open a new tab in your default web browser with the Streamlit application.

- Dashboards Overview
- The Streamlit application will contain the following features:
- CSV Inspection: Users can upload the generated CSV file to inspect its contents.
- Daily Profit/Loss Dashboard: Visualizes daily profits and losses based on the generated data.
- Popular Products Dashboard: Displays the most popular products or categories based on order frequency.
- Filtering Options: Users can filter the data based on any of the fields in the CSV.
- Graphs/Charts: Appropriate visualizations (e.g., bar charts, line graphs) will be included where useful.
- Fraudulent Orders Dashboard (Bonus): Identifies potential fraudulent orders based on criteria such as unusually high discounts or suspicious patterns in order history.
- Testing
- To ensure the reliability of the code, unit tests have been implemented. To run the tests, execute:
  pytest

Code Structure

ecommerce-data-processing/
- │
- ├── app.py # Main Streamlit application
- ├── generate_data.py # Script for generating synthetic data
- ├── requirements.txt # Python dependencies
- ├── tests/ # Directory for unit tests
- │ ├── test_generate_data.py # Tests for the data generation script
- │ └── test_app.py # Tests for the Streamlit app
- │
- └── .gitignore # Git ignore file

Contributing
- If you would like to contribute to this project, please fork the repository and submit a pull request with your changes.

By following this README, you should be able to install the necessary dependencies, generate synthetic data, and run the Streamlit application to visualize and analyze the ecommerce order transaction data.

Sure! Below is a structured approach to generate synthetic eCommerce order transaction data, create a Streamlit dashboard to visualize the data, and implement unit tests.

Step 1: Generate Synthetic eCommerce Data

1.1. Python Script to Generate Data

```
import pandas as pd
import numpy as np
import random
from faker import Faker

# Initialize Faker
fake = Faker()

# Constants
NUM_ROWS = 100000
PRODUCTS = ['Laptop', 'Smartphone', 'Tablet', 'Headphones', 'Smartwatch']
```

```
CATEGORIES = ['Electronics', 'Accessories', 'Gadgets']
COUPON_CODES = ['SAVE10', 'SUMMER20', 'FALL15', 'WINTER25', None]

# Generate synthetic data
def generate_data(num_rows):
    data = []
    for _ in range(num_rows):
        order_id = fake.uuid4()
        customer_name = fake.name()
        product_name = random.choice(PRODUCTS)
        category = random.choice(CATEGORIES)
        total_price = round(random.uniform(50, 1500), 2)
        total_discount = round(random.uniform(0, total_price * 0.3), 2)  # Up to 30%
discount
        cost_price = round(total_price - total_discount, 2) * random.uniform(0.5, 0.9)
# Cost is 50% to 90% of selling price
        coupon_code = random.choice(COUPON_CODES)

        data.append({
            'order_id': order_id,
            'customer_name': customer_name,
            'product_name': product_name,
            'category': category,
            'total_price': total_price,
            'total_discount': total_discount,
            'cost_price': cost_price,
            'coupon_code': coupon_code,
            'order_date': fake.date_time_this_year()
        })

    return pd.DataFrame(data)

# Create DataFrame and save to CSV
df = generate_data(NUM_ROWS)
df.to_csv('ecommerce_orders.csv', index=False)
print("Synthetic eCommerce data generated and saved to
'ecommerce_orders.csv'.")
```

Step 2: **Set Up Streamlit Dashboard**

2.1. **Streamlit App Structure**
Create a folder structure for the Streamlit app:

```
/ecommerce_dashboard
    ├─── app.py
    ├─── requirements.txt
    ├─── .gitignore
```

```
├── Dockerfile
├── tests
│   └── test_app.py
```

2.2. **Streamlit Application Code** (**app.py**)

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Load data
@st.cache
def load_data(file):
    data = pd.read_csv(file)
    data['profit'] = data['total_price'] - data['cost_price']
    data['order_date'] = pd.to_datetime(data['order_date'])
    return data

# Sidebar for file upload
st.sidebar.title("Upload CSV")
uploaded_file = st.sidebar.file_uploader("Choose a CSV file", type="csv")

if uploaded_file:
    df = load_data(uploaded_file)

    # Dashboard Title
    st.title("E-Commerce Dashboard")

    # Daily Profit/Loss
    st.subheader("Daily Profit/Loss")
    daily_profit = df.groupby(df['order_date'].dt.date)['profit'].sum()
    st.line_chart(daily_profit)

    # Most Popular Products
    st.subheader("Most Popular Products")
    popular_products = df['product_name'].value_counts()
    st.bar_chart(popular_products)

    # Filtering Options
    st.subheader("Filter Data")
    product_filter = st.selectbox("Select Product", df['product_name'].unique())
    filtered_data = df[df['product_name'] == product_filter]
    st.write(filtered_data)

    # Potential Fraud Detection (simple heuristic)
    st.subheader("Potential Fraudulent Orders")
    potential_fraud = df[(df['total_price'] > 1000) & (df['total_discount'] > 200)]
```

```
    st.write(potential_fraud)
```

2.3. **Requirements File** (**requirements.txt**)

```
streamlit
pandas
matplotlib
faker
```

2.4. **Dockerfile**

```
# Use the official Python image
FROM python:3.9

# Set the working directory
WORKDIR /app

# Copy the requirements file
COPY requirements.txt .

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Command to run the app
CMD ["streamlit", "run", "app.py", "--server.port=8501", "--
server.address=0.0.0.0"]
```