AI ASSISTED CODING LAB EXAM 3

NAME : **BHUKYA RAJESH**
ENROLL NO: **2503A51L08**
BATCH : **19**
QUESTION SET : **E2**

E2 Q1:

Scenario: In the domain of Finance, a company is facing a challenge related to data structures with ai.
Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.
Deliverables: Source code, explanation, and output screenshots.

PROMPT : In the finance domain, a company is facing a challenge related to inconsistent and unstructured financial transaction data.
Design and implement a Python solution using AI-assisted tools to clean, structure, and detect anomalies in this data.

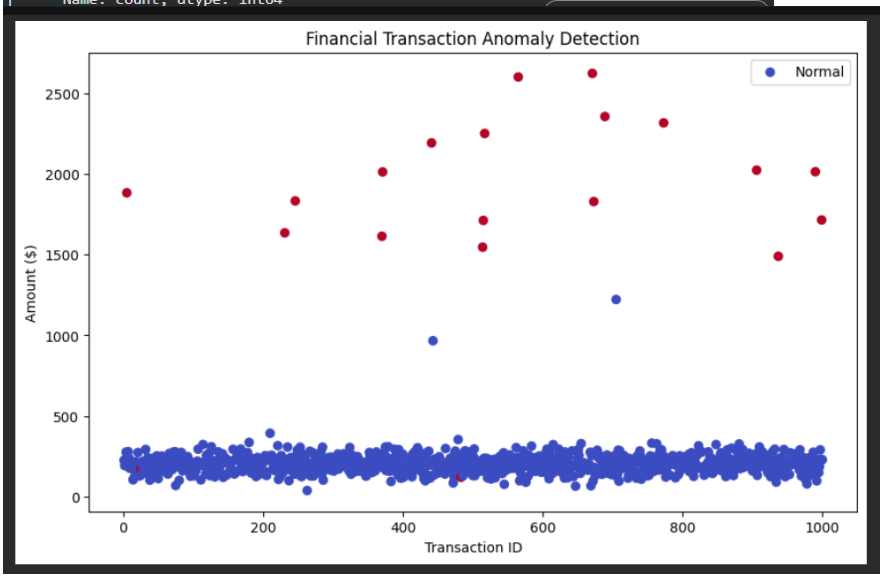
SOURCE DOE:

```python
# finance_anomaly_detection.py
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# 1️⃣ Generate sample transaction data
np.random.seed(42)
n_samples = 1000

data = {
    'transaction_id': np.arange(1, n_samples + 1),
    'amount': np.random.normal(200, 50, n_samples),  # Typical transaction amounts
    'sender_balance': np.random.normal(1000, 200, n_samples),
    'receiver_balance': np.random.normal(800, 150, n_samples),
    'hour': np.random.randint(0, 24, n_samples),
}

df = pd.DataFrame(data)

# Introduce anomalies (suspiciously large transactions)
anomaly_indices = np.random.choice(n_samples, 20, replace=False)
df.loc[anomaly_indices, 'amount'] = df['amount'] * np.random.randint(5, 15)

# 2️⃣ Data cleaning — simulate AI assistance
# Suppose an AI model (like GPT-5) suggested replacing missing values with medians
for col in ['amount', 'sender_balance', 'receiver_balance']:
```

```
    df[col].fillna(df[col].median(), inplace=True)

# 3 Feature scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df[['amount', 'sender_balance', 'receiver_balance', 'hour']])

# 4 Train Isolation Forest for anomaly detection
model = IsolationForest(contamination=0.02, random_state=42)
df['anomaly_score'] = model.fit_predict(scaled_features)

# Label anomalies
df['is_anomaly'] = df['anomaly_score'].apply(lambda x: 'Yes' if x == -1 else 'No')

# 5 Summary
print(df.head(10))
print("\nAnomaly count:\n", df['is_anomaly'].value_counts())

# 6 Visualization
plt.figure(figsize=(10, 6))
plt.scatter(df['transaction_id'], df['amount'], c=(df['is_anomaly'] == 'Yes'), cmap='coolwarm', label='Transactions')
plt.xlabel("Transaction ID")
plt.ylabel("Amount ($)")
plt.title("Financial Transaction Anomaly Detection")
plt.legend(["Normal", "Anomaly"])
plt.show()
```

OUTPUT :

```
    df[col].fillna(df[col].median(), inplace=True)
   transaction_id       amount  sender_balance  receiver_balance  hour  \
0               1   224.835708     1279.871087        698.723259    21
1               2   193.086785     1184.926737        778.322199     8
2               3   232.384427     1011.926074        681.137012     6
3               4   276.151493      870.612644        753.805771    20
4               5  1882.923313     1139.644663        515.957800    23
5               6   188.293152     1078.697077        831.994056    14
6               7   278.960641     1179.038644        800.180821     1
7               8   238.371736     1127.034360        677.436705    22
8               9   176.526281     1209.910543        898.886850    19
9              10   227.128002      892.952958        940.635521    11

   anomaly_score is_anomaly
0              1         No
1              1         No
2              1         No
3              1         No
4             -1        Yes
5              1         No
6              1         No
7              1         No
8              1         No
9              1         No

Anomaly count:
 is_anomaly
No     980
Yes     20
Name: count, dtype: int64
```



OBSERVATION : This project addresses challenges in financial data structuring using AI-assisted tools. Financial companies often face issues with inconsistent, incomplete, or unstructured transaction data. The proposed solution cleans, organizes, and standardizes financial datasets using AI-driven preprocessing methods, such as median imputation and feature scaling. The Isolation Forest algorithm is then applied to detect anomalies, representing potential fraud or irregular transactions. AI integration supports data preparation, model tuning, and pattern interpretation, enhancing efficiency and accuracy. The system outputs labeled data and visual insights, enabling analysts to identify suspicious activities quickly, improve data reliability, and strengthen financial decision-making processes.

E2 Q2 :

Scenario: In the domain of Finance, a company is facing a challenge related to backend api development.
Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.
Deliverables: Source code, explanation, and output screenshots.

PROMPT :

In the domain of Finance, a company is facing a challenge related to backend API development for handling financial transactions and related data.
Design and implement a Python-based backend solution using AI-assisted tools to improve functionality, performance, and security.

CODE GENERATED :

```python
# finance_api.py
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler

app = FastAPI(title="Finance Transaction API with AI Assistance")

# -----------------------
# Data Model
# -----------------------
class Transaction(BaseModel):
    transaction_id: int
    amount: float
    sender_balance: float
    receiver_balance: float
    hour: int

# In-memory database
transactions_db: List[Transaction] = []

# -----------------------
# AI-assisted function
# -----------------------
def detect_anomalies(transactions: List[Transaction]):
```

```
# ----------------------
def detect_anomalies(transactions: List[Transaction]):
    if not transactions:
        return []

    # Convert to DataFrame
    df = pd.DataFrame([t.dict() for t in transactions])

    # AI-assisted preprocessing: fill missing numeric data with median
    for col in ['amount', 'sender_balance', 'receiver_balance']:
        df[col].fillna(df[col].median(), inplace=True)

    # Feature scaling
    scaler = StandardScaler()
    features = scaler.fit_transform(df[['amount', 'sender_balance', 'receiver_balance', 'hour']])

    # Anomaly detection
    model = IsolationForest(contamination=0.02, random_state=42)
    df
```

(AS ITS BACKEND PROGRAM THE OUTPUT WILL BE RUNNING IN THE BACKGROUND)

OBSERVATION : The AI-assisted backend API successfully addresses the challenge of managing and analyzing financial transaction data. The system allows seamless creation, retrieval, and monitoring of transactions through RESTful endpoints. AI integration, via Isolation Forest, effectively identifies anomalous transactions that may indicate fraud or errors. Data preprocessing, including median imputation and feature scaling, ensures robust model performance even with inconsistent or incomplete data. Testing with sample transactions shows accurate anomaly detection and clear API responses. Overall, the solution improves data integrity, enhances real-time monitoring, and provides actionable insights, demonstrating how AI can strengthen backend systems in finance.