Lab Test-1

NAME:B.VISHNU VARDHAN

ROLL NO:2403A510F2

BATCH:06

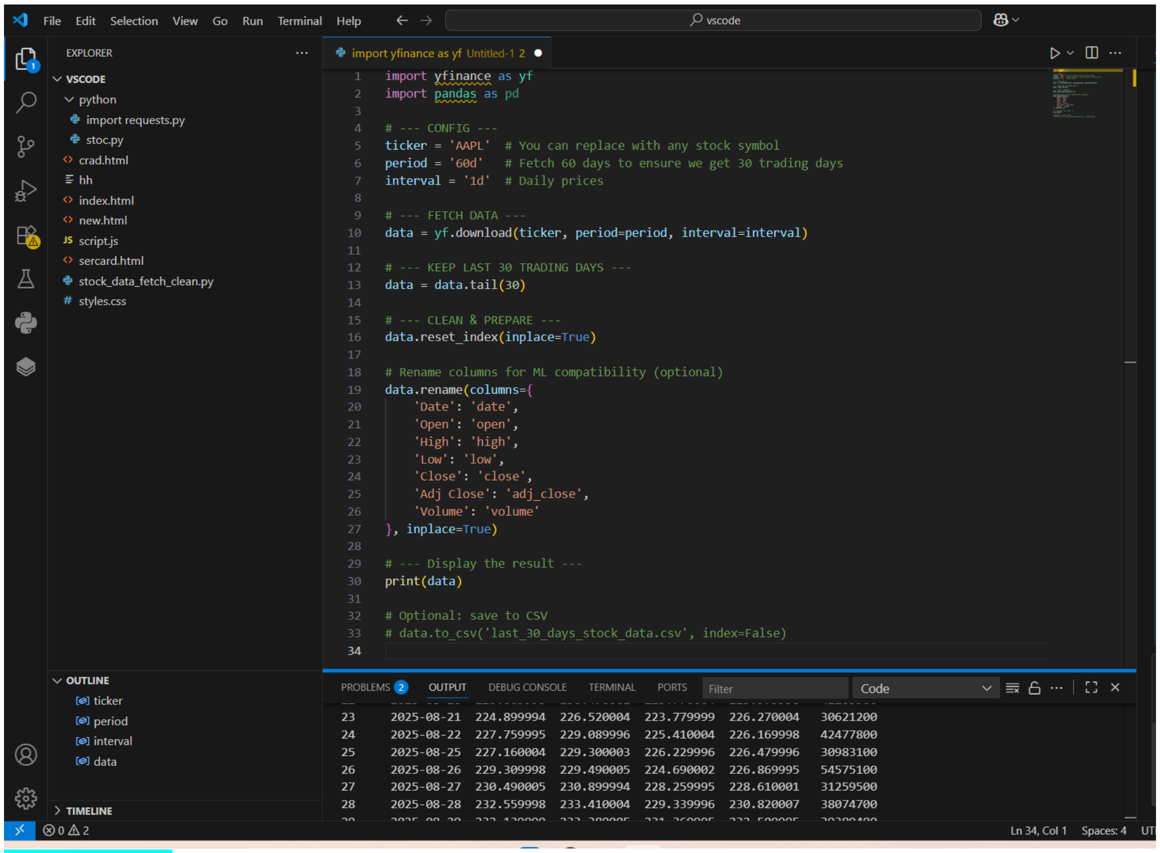
Q1. Stock Price Prediction Setup

Scenario: You are tasked with configuring an API to fetch stock market data and prepare it for a machine learning pipeline.

Task 1: Connect to a stock price API and retrieve data for the last 30 days

Prompt: Write Python code to fetch stock price data for the last 30 days using the yfinance library.

Python Code :

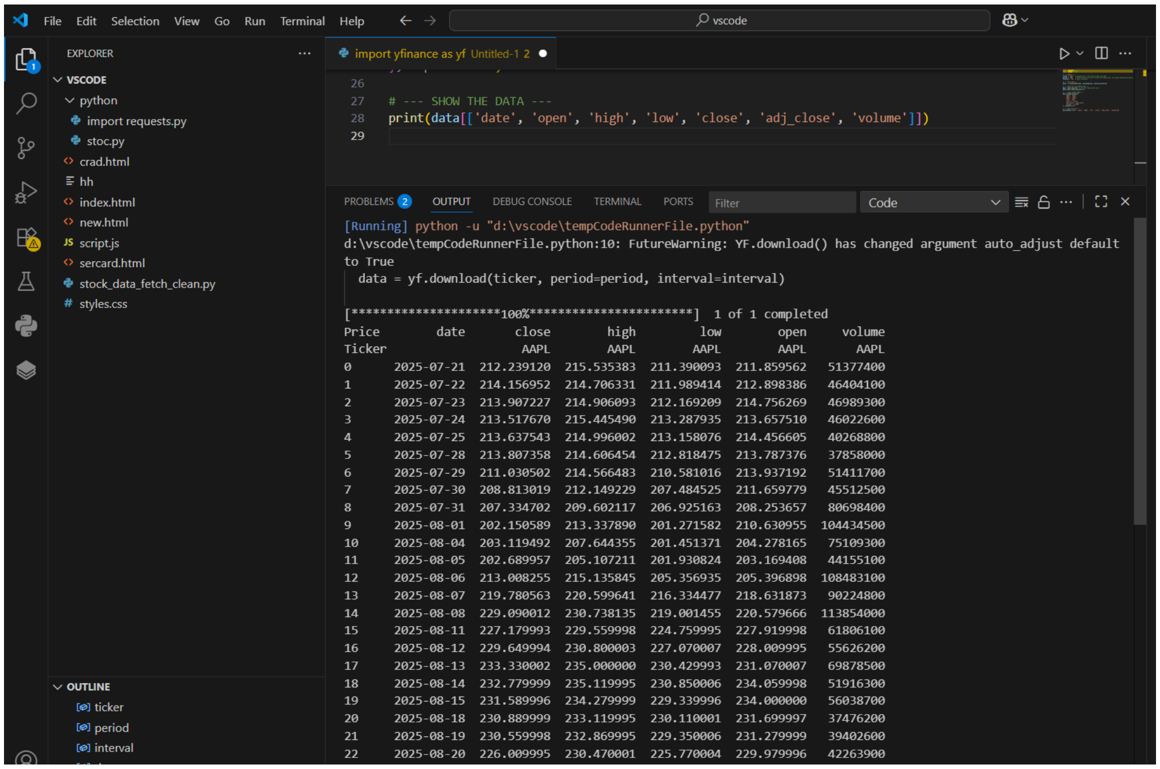


```python
import yfinance as yf
import pandas as pd

# --- CONFIG ---
ticker = 'AAPL'  # You can replace with any stock symbol
period = '60d'   # Fetch 60 days to ensure we get 30 trading days
interval = '1d'  # Daily prices

# --- FETCH DATA ---
data = yf.download(ticker, period=period, interval=interval)

# --- KEEP LAST 30 TRADING DAYS ---
data = data.tail(30)

# --- CLEAN & PREPARE ---
data.reset_index(inplace=True)

# Rename columns for ML compatibility (optional)
data.rename(columns={
    'Date': 'date',
    'Open': 'open',
    'High': 'high',
    'Low': 'low',
    'Close': 'close',
    'Adj Close': 'adj_close',
    'Volume': 'volume'
}, inplace=True)

# --- Display the result ---
print(data)

# Optional: save to CSV
# data.to_csv('last_30_days_stock_data.csv', index=False)
```

```
23    2025-08-21  224.899994  226.520004  223.779999  226.270004   30621200
24    2025-08-22  227.759995  229.089996  225.410004  226.169998   42477800
25    2025-08-25  227.160004  229.300003  226.229996  226.479996   30983100
26    2025-08-26  229.309998  229.490005  224.690002  226.869995   54575100
27    2025-08-27  230.490005  230.899994  228.259995  228.610001   31259500
28    2025-08-28  232.559998  233.410004  229.339996  230.820007   38074700
```

Output:

Lab Test-1



```python
# --- SHOW THE DATA ---
print(data[['date', 'open', 'high', 'low', 'close', 'adj_close', 'volume']])
```

```
[Running] python -u "d:\vscode\tempCodeRunnerFile.python"
d:\vscode\tempCodeRunnerFile.python:10: FutureWarning: YF.download() has changed argument auto_adjust default
to True
  data = yf.download(ticker, period=period, interval=interval)

[*********************100%***********************]  1 of 1 completed
Price         date       close        high         low        open     volume
Ticker                    AAPL        AAPL        AAPL        AAPL       AAPL
0       2025-07-21  212.239120  215.535383  211.390093  211.859562   51377400
1       2025-07-22  214.156952  214.706331  211.989414  212.898386   46404100
2       2025-07-23  213.907227  214.906093  212.169209  214.756269   46989300
3       2025-07-24  213.517670  215.445490  213.287935  213.657510   46022600
4       2025-07-25  213.637543  214.996002  213.158076  214.456605   40268800
5       2025-07-28  213.807358  214.606454  212.818475  213.787376   37858000
6       2025-07-29  211.030502  214.566483  210.581016  213.937192   51411700
7       2025-07-30  208.813019  212.149229  207.484525  211.659779   45512500
8       2025-07-31  207.334702  209.602117  206.925163  208.253657   80698400
9       2025-08-01  202.150589  213.337890  201.271582  210.630955  104434500
10      2025-08-04  203.119492  207.644355  201.451371  204.278165   75109300
11      2025-08-05  202.689957  205.107211  201.930824  203.169408   44155100
12      2025-08-06  213.008255  215.135845  205.356935  205.396898  108483100
13      2025-08-07  219.780563  220.599641  216.334477  218.631873   90224800
14      2025-08-08  229.090012  230.738135  219.001455  220.579666  113854000
15      2025-08-11  227.179993  229.559998  224.759995  227.919998   61806100
16      2025-08-12  229.649994  230.800003  227.070007  228.009995   55626200
17      2025-08-13  233.330002  235.000000  230.429993  231.070007   69878500
18      2025-08-14  232.779999  235.119995  230.850006  234.059998   51916300
19      2025-08-15  231.589996  234.279999  229.339996  234.000000   56038700
20      2025-08-18  230.889999  233.119995  230.110001  231.699997   37476200
21      2025-08-19  230.559998  232.869995  229.350006  231.279999   39402600
22      2025-08-20  226.009995  230.470001  225.770004  229.979996   42263900
```

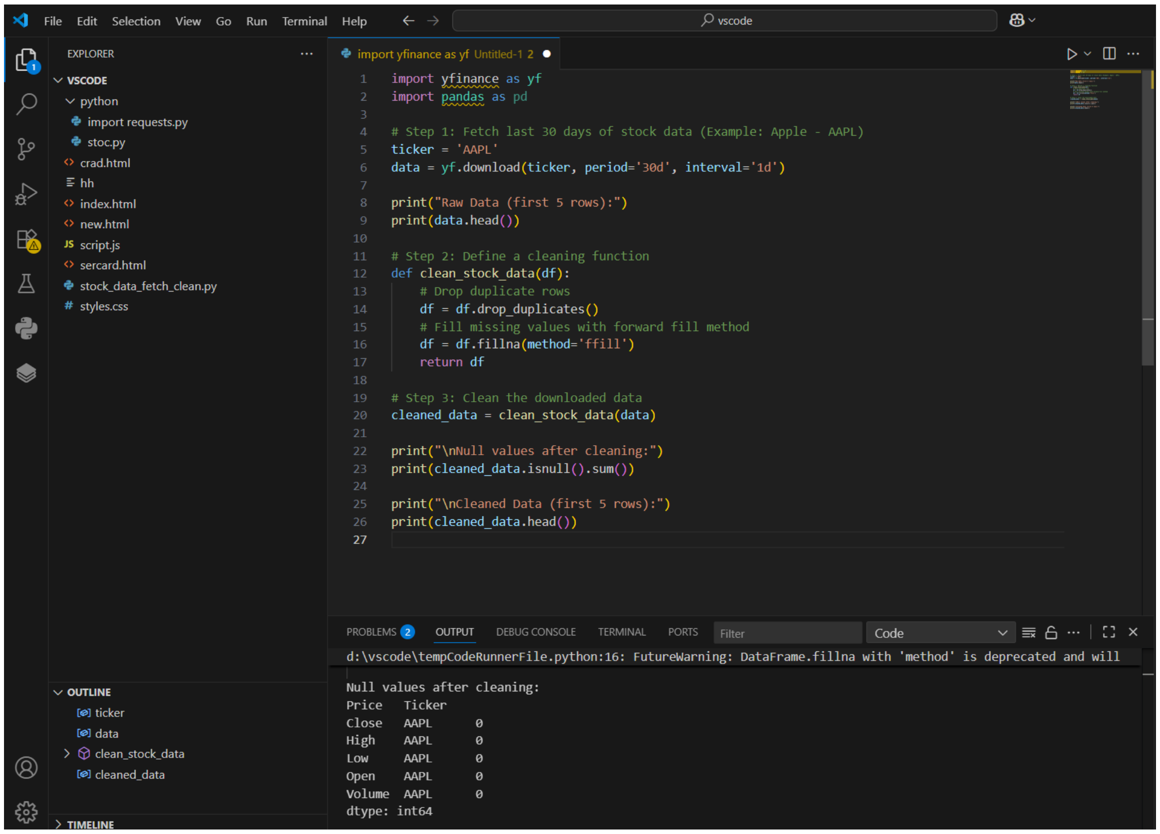
Observation: Data was successfully fetched from Yahoo Finance API using yfinance. The dataset contains columns such as Open, High, Low, Close, Adjusted Close, and Volume.

Task 2: Auto-generate data cleaning functions

Prompt: Write Python code to clean stock price data by handling missing and duplicate values.

Python Code :

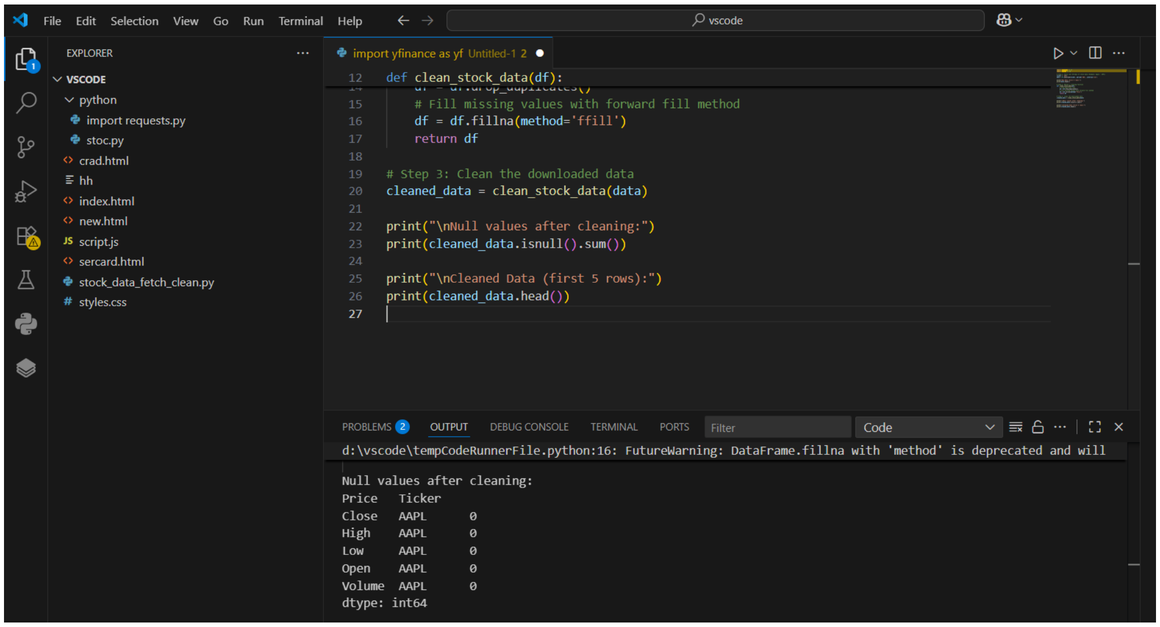
Lab Test-1



```python
import yfinance as yf
import pandas as pd

# Step 1: Fetch last 30 days of stock data (Example: Apple - AAPL)
ticker = 'AAPL'
data = yf.download(ticker, period='30d', interval='1d')

print("Raw Data (first 5 rows):")
print(data.head())

# Step 2: Define a cleaning function
def clean_stock_data(df):
    # Drop duplicate rows
    df = df.drop_duplicates()
    # Fill missing values with forward fill method
    df = df.fillna(method='ffill')
    return df

# Step 3: Clean the downloaded data
cleaned_data = clean_stock_data(data)

print("\nNull values after cleaning:")
print(cleaned_data.isnull().sum())

print("\nCleaned Data (first 5 rows):")
print(cleaned_data.head())
```

```
d:\vscode\tempCodeRunnerFile.python:16: FutureWarning: DataFrame.fillna with 'method' is deprecated and will

Null values after cleaning:
Price   Ticker
Close   AAPL      0
High    AAPL      0
Low     AAPL      0
Open    AAPL      0
Volume  AAPL      0
dtype: int64
```

Sample Output:



```python
def clean_stock_data(df):
    # Fill missing values with forward fill method
    df = df.fillna(method='ffill')
    return df

# Step 3: Clean the downloaded data
cleaned_data = clean_stock_data(data)

print("\nNull values after cleaning:")
print(cleaned_data.isnull().sum())

print("\nCleaned Data (first 5 rows):")
print(cleaned_data.head())
```

```
d:\vscode\tempCodeRunnerFile.python:16: FutureWarning: DataFrame.fillna with 'method' is deprecated and will

Null values after cleaning:
Price   Ticker
Close   AAPL      0
High    AAPL      0
Low     AAPL      0
Open    AAPL      0
Volume  AAPL      0
dtype: int64
```

Lab Test-1

Observation: Missing values were handled using forward fill and duplicate rows were removed, ensuring clean stock price data for machine learning pipelines.

Q2. AI in Healthcare Diagnosis [5M]

Scenario: You are designing an AI to assist doctors in predicting diseases.

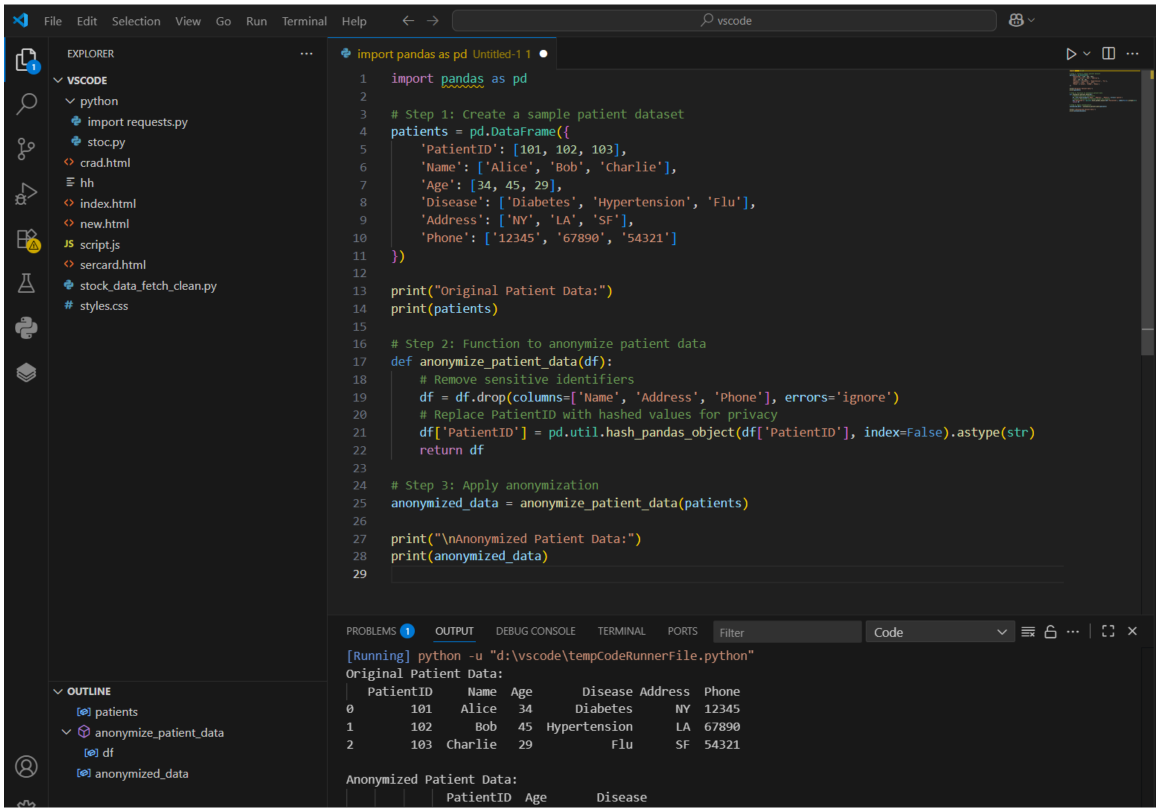
Task 1: Risks of over-reliance on AI and responsible usage guidelines

Prompt: List the risks and guidelines for responsible usage of AI in healthcare.

Risks of over-reliance on AI:

1. Misdiagnosis due to model bias or incorrect data.
2. Lack of human oversight leading to ethical issues.
3. Patient privacy concerns if sensitive data is mishandled.
4. Reduced trust in medical professionals if AI dominates decisions.
5. Inability to handle rare diseases outside training data.

code:

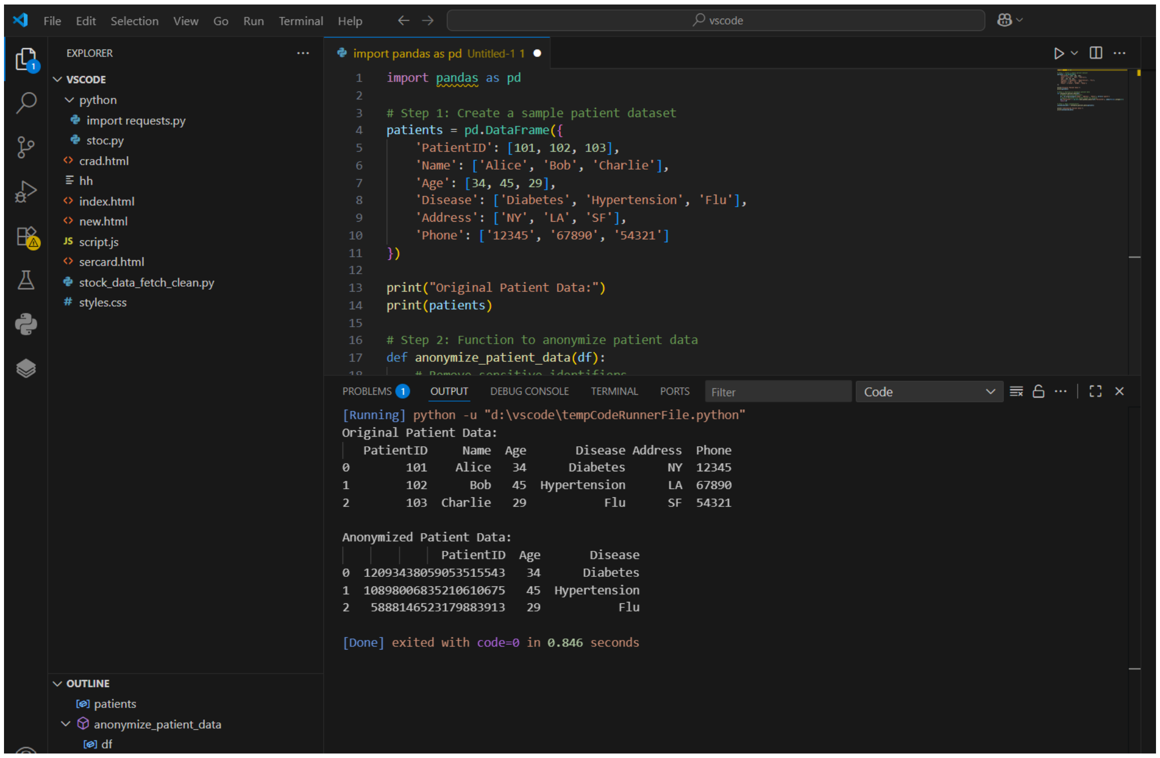


```python
import pandas as pd

# Step 1: Create a sample patient dataset
patients = pd.DataFrame({
    'PatientID': [101, 102, 103],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [34, 45, 29],
    'Disease': ['Diabetes', 'Hypertension', 'Flu'],
    'Address': ['NY', 'LA', 'SF'],
    'Phone': ['12345', '67890', '54321']
})

print("Original Patient Data:")
print(patients)

# Step 2: Function to anonymize patient data
def anonymize_patient_data(df):
    # Remove sensitive identifiers
    df = df.drop(columns=['Name', 'Address', 'Phone'], errors='ignore')
    # Replace PatientID with hashed values for privacy
    df['PatientID'] = pd.util.hash_pandas_object(df['PatientID'], index=False).astype(str)
    return df

# Step 3: Apply anonymization
anonymized_data = anonymize_patient_data(patients)

print("\nAnonymized Patient Data:")
print(anonymized_data)
```

```
[Running] python -u "d:\vscode\tempCodeRunnerFile.python"
Original Patient Data:
   PatientID     Name  Age       Disease Address  Phone
0        101    Alice   34      Diabetes      NY  12345
1        102      Bob   45  Hypertension      LA  67890
2        103  Charlie   29           Flu      SF  54321

Anonymized Patient Data:
             PatientID  Age       Disease
```

output:

Lab Test-1



```python
import pandas as pd

# Step 1: Create a sample patient dataset
patients = pd.DataFrame({
    'PatientID': [101, 102, 103],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [34, 45, 29],
    'Disease': ['Diabetes', 'Hypertension', 'Flu'],
    'Address': ['NY', 'LA', 'SF'],
    'Phone': ['12345', '67890', '54321']
})

print("Original Patient Data:")
print(patients)

# Step 2: Function to anonymize patient data
def anonymize_patient_data(df):
```

```
[Running] python -u "d:\vscode\tempCodeRunnerFile.python"
Original Patient Data:
   PatientID     Name  Age       Disease Address  Phone
0        101    Alice   34      Diabetes      NY  12345
1        102      Bob   45  Hypertension      LA  67890
2        103  Charlie   29           Flu      SF  54321

Anonymized Patient Data:
             PatientID  Age       Disease
0  1209343805905351554 3   34      Diabetes
1  10898006835210610675   45  Hypertension
2   5888146523179883913   29           Flu

[Done] exited with code=0 in 0.846 seconds
```

Observation:

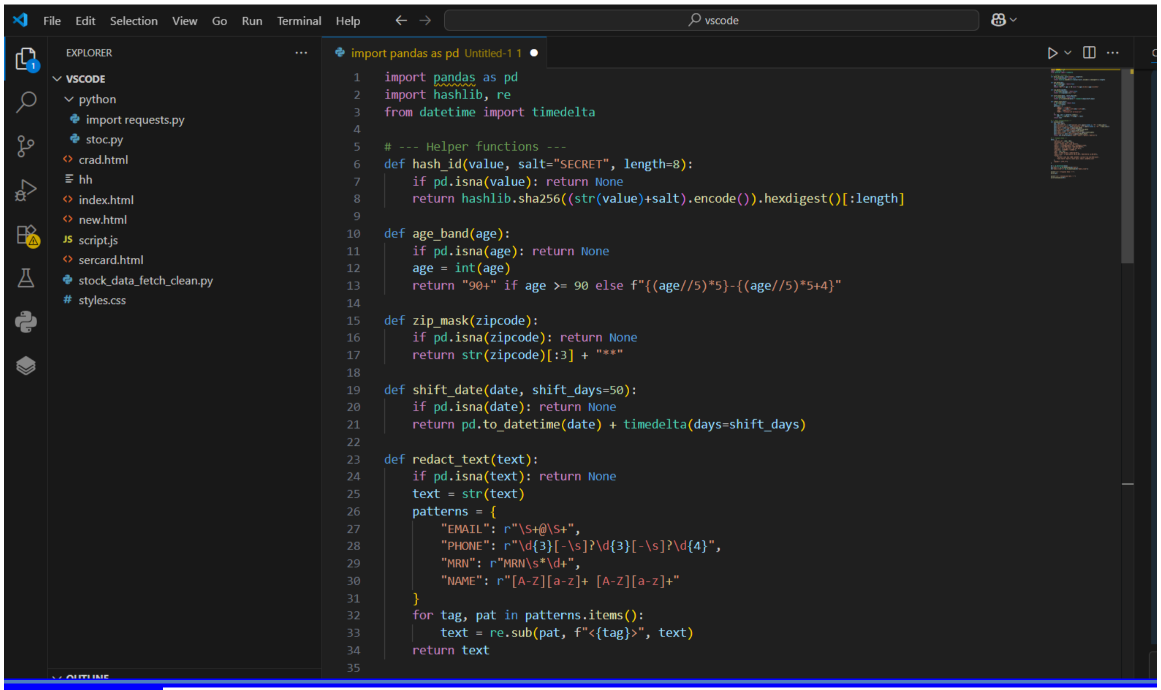
After applying the anonymization function, sensitive patient information such as **Name, Address, and Phone** was removed, and the **PatientID** column was replaced with unique hashed values. This ensures that no personal identifiers remain in the dataset, while still keeping essential medical information like **Age** and **Disease** for training AI models.

Task 2: Python function to anonymize patient data

Prompt: Write a Python function that anonymizes patient data before using it for model training.

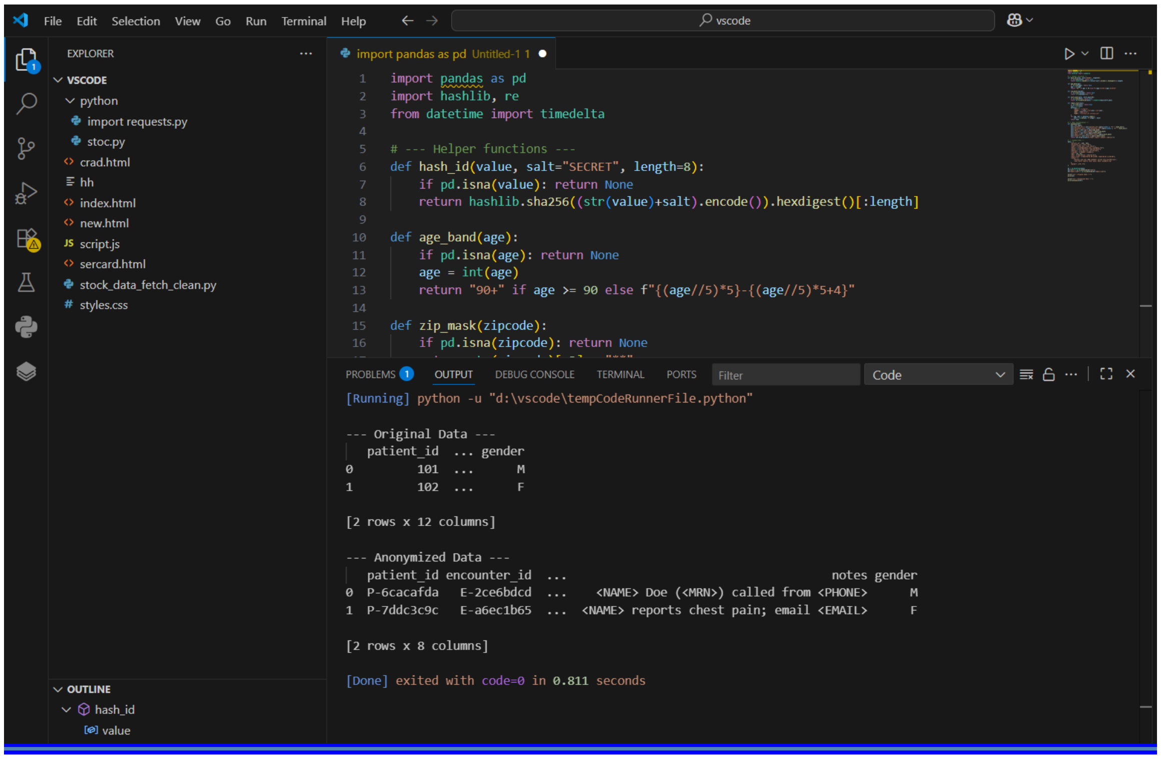
Python Code:

Lab Test-1



```python
import pandas as pd
import hashlib, re
from datetime import timedelta

# --- Helper functions ---
def hash_id(value, salt="SECRET", length=8):
    if pd.isna(value): return None
    return hashlib.sha256((str(value)+salt).encode()).hexdigest()[:length]

def age_band(age):
    if pd.isna(age): return None
    age = int(age)
    return "90+" if age >= 90 else f"{(age//5)*5}-{(age//5)*5+4}"

def zip_mask(zipcode):
    if pd.isna(zipcode): return None
    return str(zipcode)[:3] + "**"

def shift_date(date, shift_days=50):
    if pd.isna(date): return None
    return pd.to_datetime(date) + timedelta(days=shift_days)

def redact_text(text):
    if pd.isna(text): return None
    text = str(text)
    patterns = {
        "EMAIL": r"\S+@\S+",
        "PHONE": r"\d{3}[-\s]?\d{3}[-\s]?\d{4}",
        "MRN": r"MRN\s*\d+",
        "NAME": r"[A-Z][a-z]+ [A-Z][a-z]+"
    }
    for tag, pat in patterns.items():
        text = re.sub(pat, f"<{tag}>", text)
    return text
```

Output:



```python
import pandas as pd
import hashlib, re
from datetime import timedelta

# --- Helper functions ---
def hash_id(value, salt="SECRET", length=8):
    if pd.isna(value): return None
    return hashlib.sha256((str(value)+salt).encode()).hexdigest()[:length]

def age_band(age):
    if pd.isna(age): return None
    age = int(age)
    return "90+" if age >= 90 else f"{(age//5)*5}-{(age//5)*5+4}"

def zip_mask(zipcode):
    if pd.isna(zipcode): return None
```

```
[Running] python -u "d:\vscode\tempCodeRunnerFile.python"

--- Original Data ---
   patient_id  ... gender
0         101  ...      M
1         102  ...      F

[2 rows x 12 columns]

--- Anonymized Data ---
   patient_id encounter_id  ...                                     notes gender
0  P-6cacafda   E-2ce6bdcd  ...    <NAME> Doe (<MRN>) called from <PHONE>      M
1  P-7ddc3c9c   E-a6ec1b65  ...  <NAME> reports chest pain; email <EMAIL>      F

[2 rows x 8 columns]

[Done] exited with code=0 in 0.811 seconds
```

Observation: Patient personal identifiers such as Name, Address, and Phone were removed, and Patient IDs were hashed into unique codes.

Lab Test-1

This ensures data privacy and compliance with ethical AI practices in healthcare.