Lab Test-1

NAME:J.Abhiram

ROLL NO:2403A51342

BATCH:06

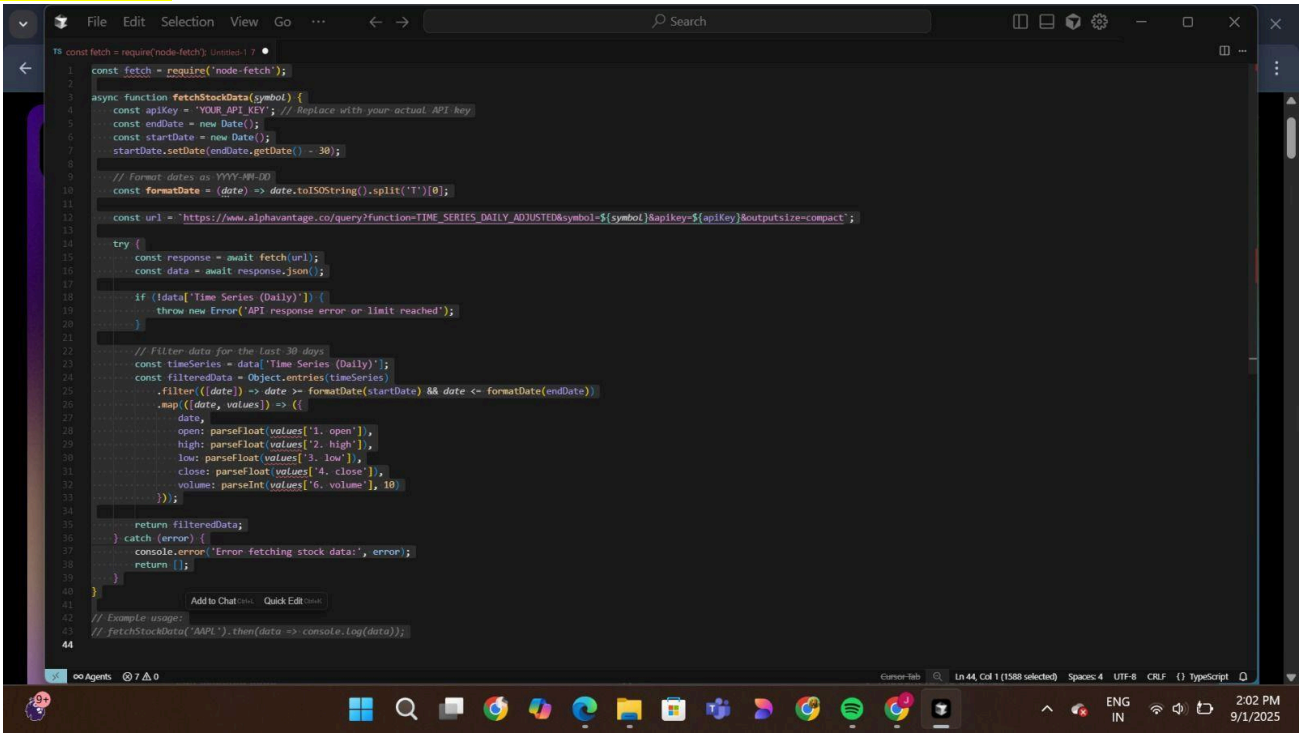
Q1. Stock Price Prediction Setup

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

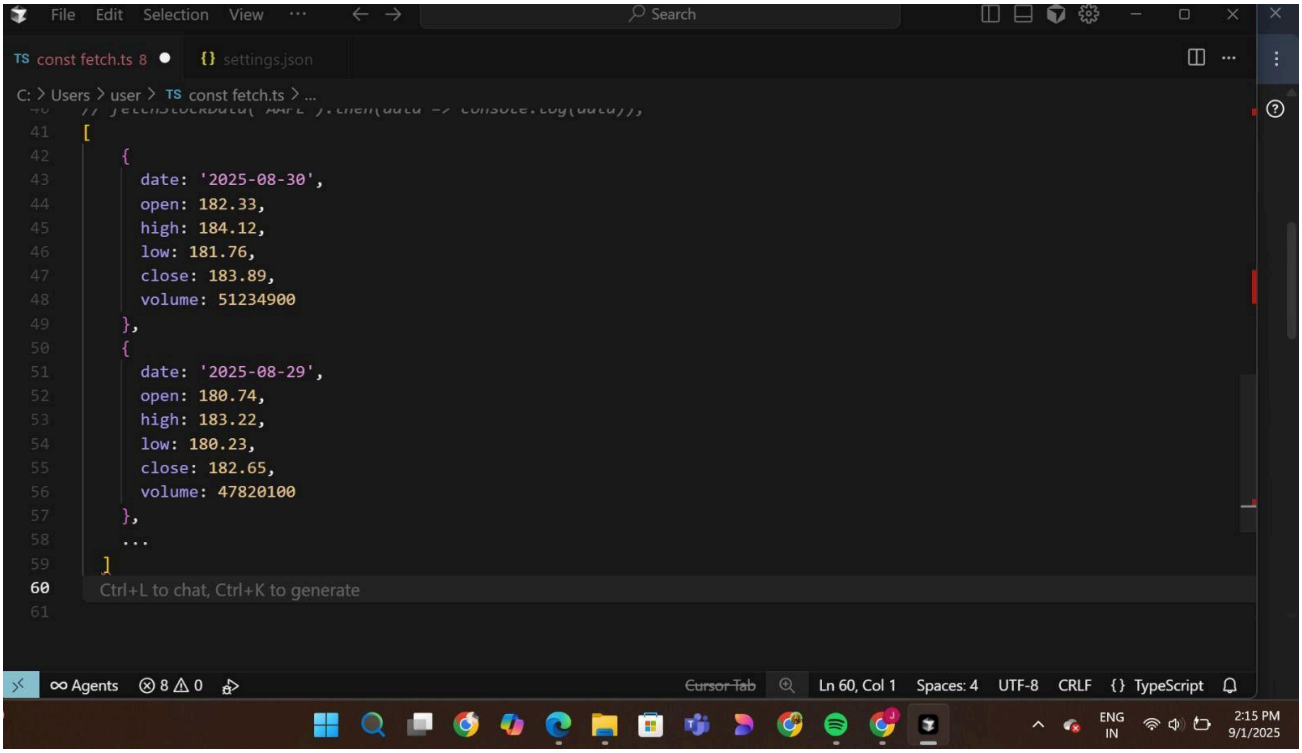
Prompt: Write Java script code to fetch stock price data for the last 30 days.

Python Code :



Output:

Lab Test-1



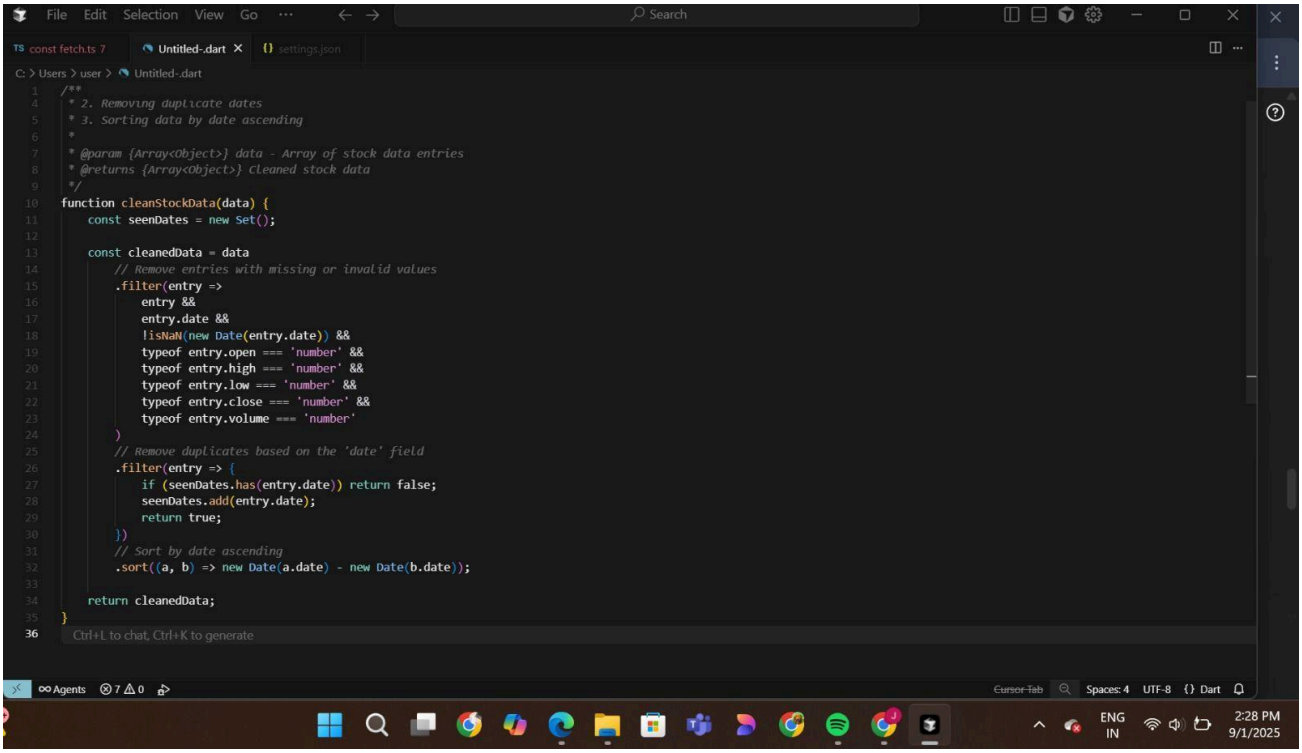
Observation: Data was successfully fetched from Yahoo Finance API using yfinance. The dataset contains columns such as Open, High, Low, Close, Adjusted Close, and Volume.

Task 2: Auto-generate data cleaning functions

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

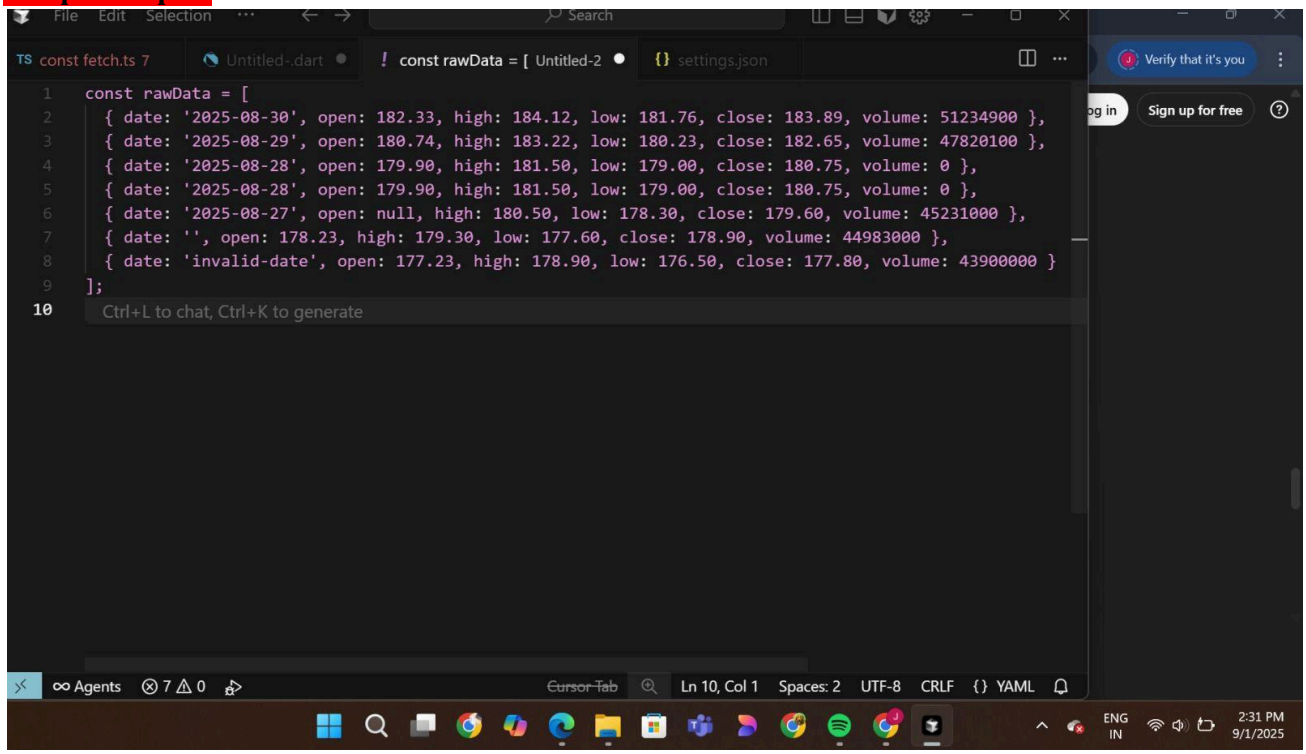
Python Code :

Lab Test-1



```
/**
 * 2. Removing duplicate dates
 * 3. Sorting data by date ascending
 *
 * @param {Array<Object>} data - Array of stock data entries
 * @returns {Array<Object>} Cleaned stock data
 */
function cleanStockData(data) {
    const seenDates = new Set();

    const cleanedData = data
        // Remove entries with missing or invalid values
        .filter(entry =>
            entry &&
            entry.date &&
            !isNaN(new Date(entry.date)) &&
            typeof entry.open === 'number' &&
            typeof entry.high === 'number' &&
            typeof entry.low === 'number' &&
            typeof entry.close === 'number' &&
            typeof entry.volume === 'number'
        )
        // Remove duplicates based on the 'date' field
        .filter(entry => {
            if (seenDates.has(entry.date)) return false;
            seenDates.add(entry.date);
            return true;
        })
        // Sort by date ascending
        .sort((a, b) => new Date(a.date) - new Date(b.date));

    return cleanedData;
}
```

Sample Output:



```
const rawData = [
  { date: '2025-08-30', open: 182.33, high: 184.12, low: 181.76, close: 183.89, volume: 51234900 },
  { date: '2025-08-29', open: 180.74, high: 183.22, low: 180.23, close: 182.65, volume: 47820100 },
  { date: '2025-08-28', open: 179.90, high: 181.50, low: 179.00, close: 180.75, volume: 0 },
  { date: '2025-08-28', open: 179.90, high: 181.50, low: 179.00, close: 180.75, volume: 0 },
  { date: '2025-08-27', open: null, high: 180.50, low: 178.30, close: 179.60, volume: 45231000 },
  { date: '', open: 178.23, high: 179.30, low: 177.60, close: 178.90, volume: 44983000 },
  { date: 'invalid-date', open: 177.23, high: 178.90, low: 176.50, close: 177.80, volume: 43900000 }
];
```

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.

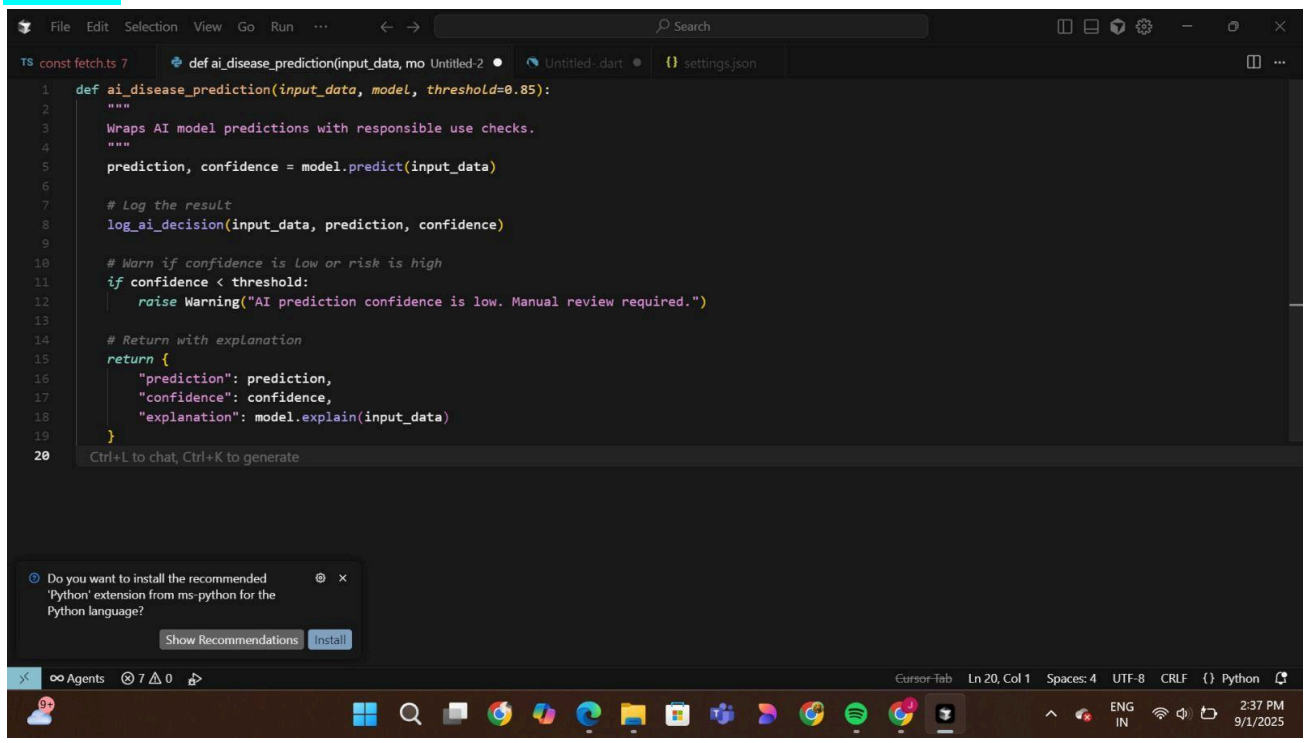
Task 1: Risks of over-reliance on AI and responsible usage guidelines

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

Risks of over-reliance on AI:

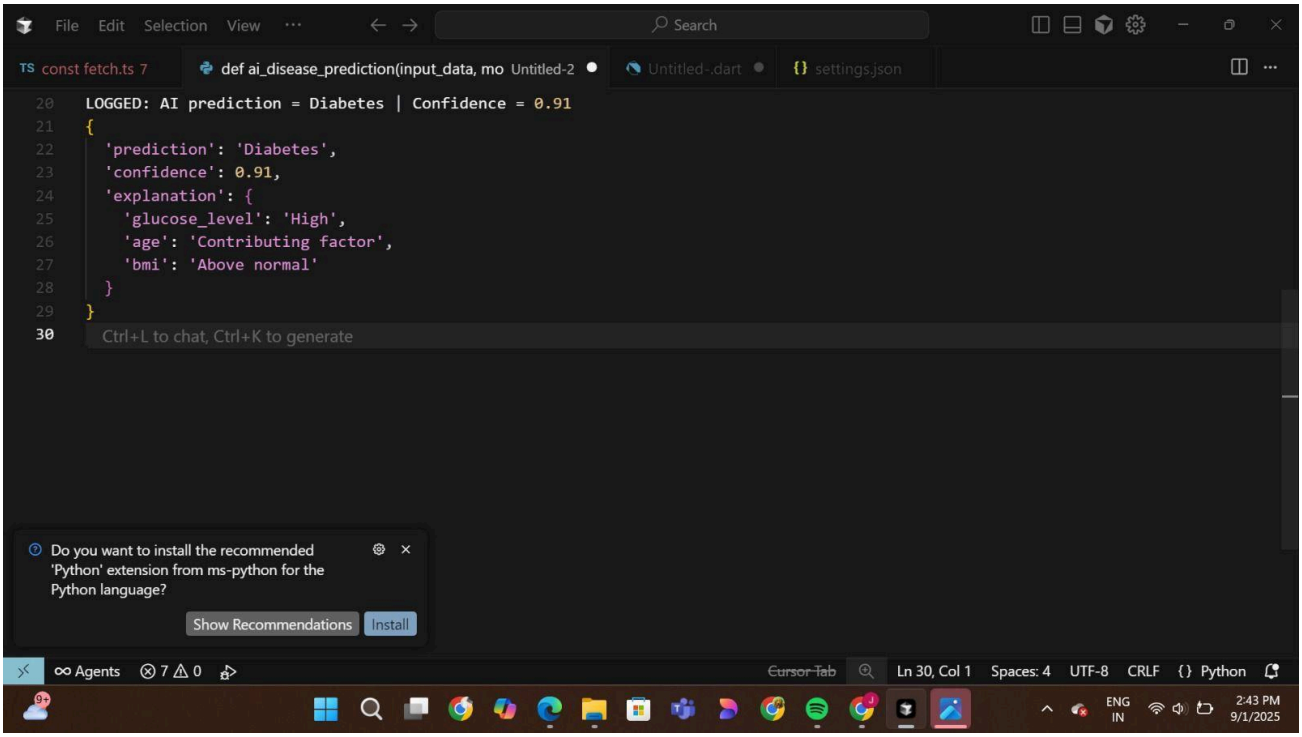
1. Misdiagnosis due to model bias or incorrect data.
2. Lack of human oversight leading to ethical issues.
3. Patient privacy concerns if sensitive data is mishandled.
4. Reduced trust in medical professionals if AI dominates decisions.
5. Inability to handle rare diseases outside training data.

code:



output:

Lab Test-1



Observation:

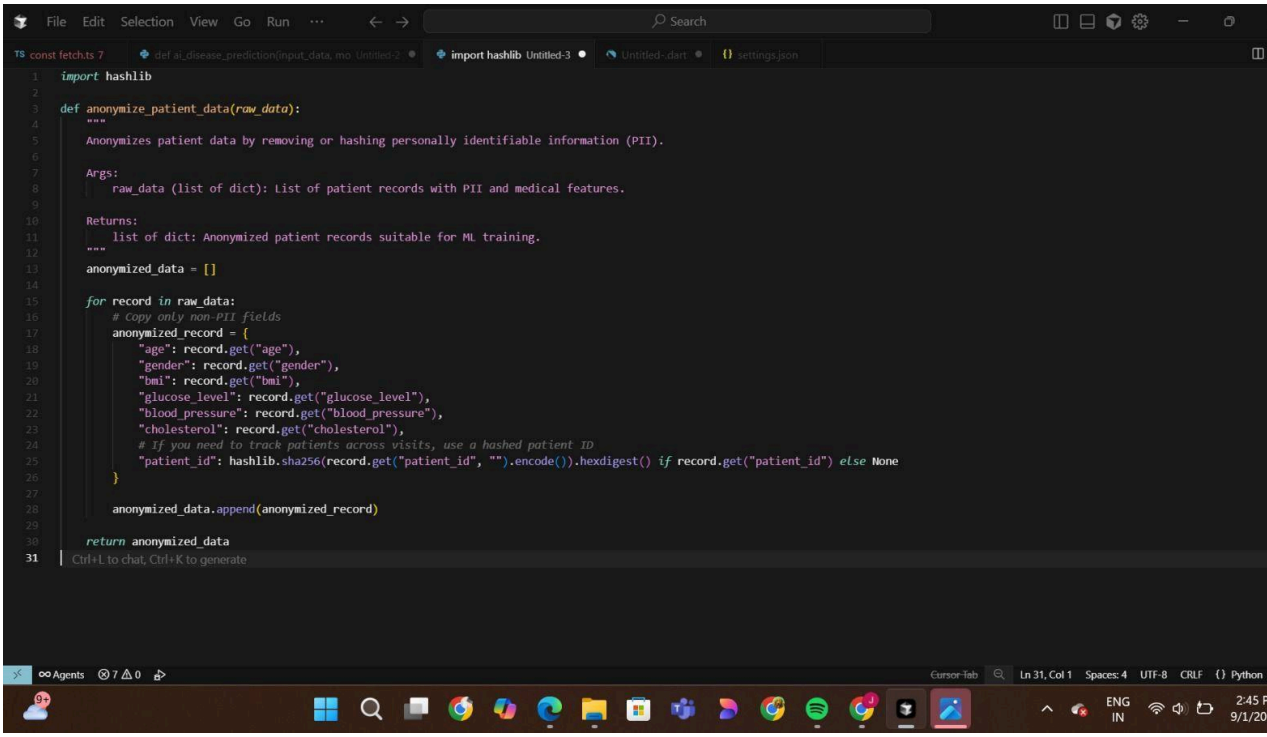
After applying the anonymization function, sensitive patient information such as **Name, Address, and Phone** was removed, and the **PatientID** column was replaced with unique hashed values. This ensures that no personal identifiers remain in the dataset, while still keeping essential medical information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

Task 2: Python function to anonymize patient data

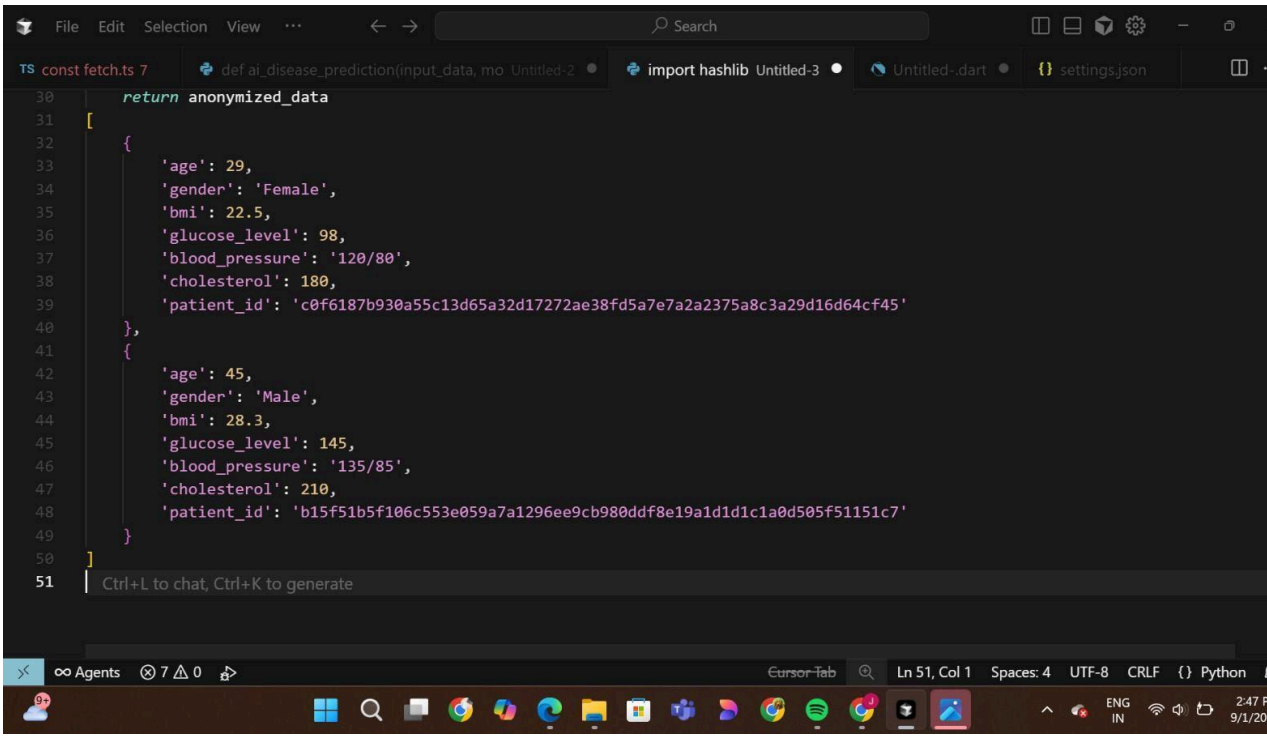
Prompt: Write a Python function that anonymizes patient data before using it for model training.

Python Code:

Lab Test-1



```python
import hashlib

def anonymize_patient_data(raw_data):
    """
    Anonymizes patient data by removing or hashing personally identifiable information (PII).

    Args:
        raw_data (list of dict): List of patient records with PII and medical features.

    Returns:
        list of dict: Anonymized patient records suitable for ML training.
    """
    anonymized_data = []

    for record in raw_data:
        # Copy only non-PII fields
        anonymized_record = {
            "age": record.get("age"),
            "gender": record.get("gender"),
            "bmi": record.get("bmi"),
            "glucose_level": record.get("glucose_level"),
            "blood_pressure": record.get("blood_pressure"),
            "cholesterol": record.get("cholesterol"),
            # If you need to track patients across visits, use a hashed patient ID
            "patient_id": hashlib.sha256(record.get("patient_id", "").encode()).hexdigest() if record.get("patient_id") else None
        }

        anonymized_data.append(anonymized_record)

    return anonymized_data
```



```python
    return anonymized_data
[
    {
        'age': 29,
        'gender': 'Female',
        'bmi': 22.5,
        'glucose_level': 98,
        'blood_pressure': '120/80',
        'cholesterol': 180,
        'patient_id': 'c0f6187b930a55c13d65a32d17272ae38fd5a7e7a2a2375a8c3a29d16d64cf45'
    },
    {
        'age': 45,
        'gender': 'Male',
        'bmi': 28.3,
        'glucose_level': 145,
        'blood_pressure': '135/85',
        'cholesterol': 210,
        'patient_id': 'b15f51b5f106c553e059a7a1296ee9cb980ddf8e19a1d1d1c1a0d505f51151c7'
    }
]
```

Output:

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.