1. **Sensor Integration (Using Python and Raspberry Pi):**

Assuming you are using a sensor like the SDS011 for PM2.5 and PM10 measurements and an MQ-135 for CO2 and other gas measurements, here's an example of how to read data from these sensors using Python:from gpiozero import PWMLED

```
import serial
import time

# SDS011 sensor setup
ser = serial.Serial('/dev/ttyUSB0', 9600)
def read_sds011():
    while True:
        data = ser.read(10)
        if data[0] == b'\xaa' and data[1] == b'\xc0':
            pm25 = int.from_bytes(data[2:4], byteorder='little') / 10
            pm10 = int.from_bytes(data[4:6], byteorder='little') / 10
            return pm25, pm10
# MQ-135 sensor setup
MQ_PIN = 17
CO2_THRESHOLD = 500  # Define your CO2 threshold level
def read_mq135():
    adc_value = 0
    for _ in range(10):
        adc_value += PWMLED(MQ_PIN).value * 1023
    adc_value /= 10
    return adc_value
def is_air_quality_poor():
    co2_value = read_mq135()
    return co2_value > CO2_THRESHOLD
while True:
    pm25, pm10 = read_sds011()
    print(f'PM2.5: {pm25} µg/m³, PM10: {pm10} µg/m³')
    if is_air_quality_poor():
        print('Poor air quality! CO2 level is high.')
            time.sleep(2)  # Read data every 2 seconds
```

2. Cloud Integration (Using AWS IoT Core and AWS Lambda):

Set up AWS IoT Core to securely connect your Raspberry Pi to the cloud.Use AWS Lambda to process incoming data and store it in an Amazon DynamoDB database.Sample Lambda function (Python) to process incoming IoT data:import json

```
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('AirQualityData')  # Replace with your DynamoDB table name
def lambda_handler(event, context):
```

```
    try:
        for record in event['Records']:
            payload = json.loads(record['payload'])
            pm25 = payload['pm25']
            pm10 = payload['pm10']
            co2 = payload['co2']
            timestamp = payload['timestamp']
            # Store data in DynamoDB
            table.put_item(
                Item={
                    'Timestamp': timestamp,
                    'PM2.5': pm25,
                    'PM10': pm10,
                    'CO2': co2
                }
            )
            print('Data stored successfully in DynamoDB.')

        return {
            'statusCode': 200,
            'body': json.dumps('Data processed successfully.')
        }
    except Exception as e:
        print(f'Error: {e}')
        return {
            'statusCode': 500,
            'body': json.dumps('Error processing data.')
        }
```

3**. Web Interface (Using Flask for Python Backend and HTML/CSS/JS for Frontend):**

Create a Flask web application to display real-time and historical air quality data.Flask backend (app.py):from flask import Flask, render_template, jsonify
import boto3

```
app = Flask(__name__)
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('AirQualityData')  # Replace with your DynamoDB table name
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/api/data')
def get_data():
    # Retrieve data from DynamoDB
    # Implement logic to fetch real-time or historical data as needed
```

```
    # Example: data = table.scan()['Items']
    data = []  # Replace with fetched data
    return jsonify(data)
if __name__ == '__main__':
    app.run(debug=True)HTML template (index.html) for displaying data:<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Air Quality Monitoring System</title>
</head>
<body>
    <h1>Air Quality Monitoring System</h1>
    <div id="data-container"></div>
    <script>
        async function fetchData() {
            const response = await fetch('/api/data');
            const data = await response.json();
            const dataContainer = document.getElementById('data-container');
            dataContainer.innerHTML = JSON.stringify(data, null, 2);
        }

        fetchData();
        setInterval(fetchData, 5000);  // Refresh data every 5 seconds
    </script>
</body>

</html>
```