**ASSIGNMENT-02**

**NAME:** Karthikeya Uthuri

**Hall Ticket:**2303A51306

**Batch:**05

1. Task 1: Word Frequency from Text File
   * Scenario:

You are analyzing log files for keyword frequency.

* + Task:

Use Gemini to generate Python code that reads a text file and counts word frequency, then explains the code.

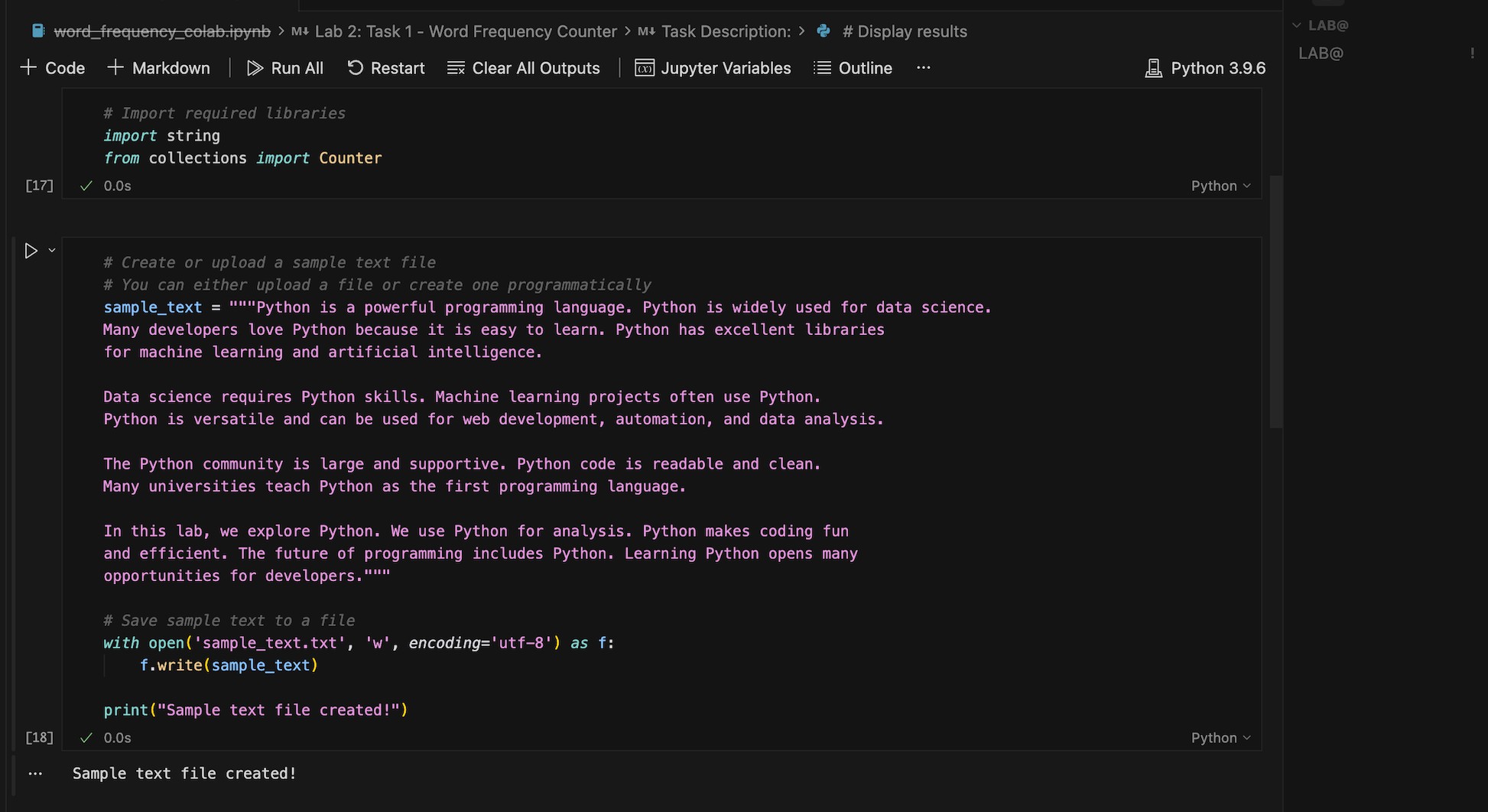
* + Expected Output:
* Working code
* Explanation
* Screenshot

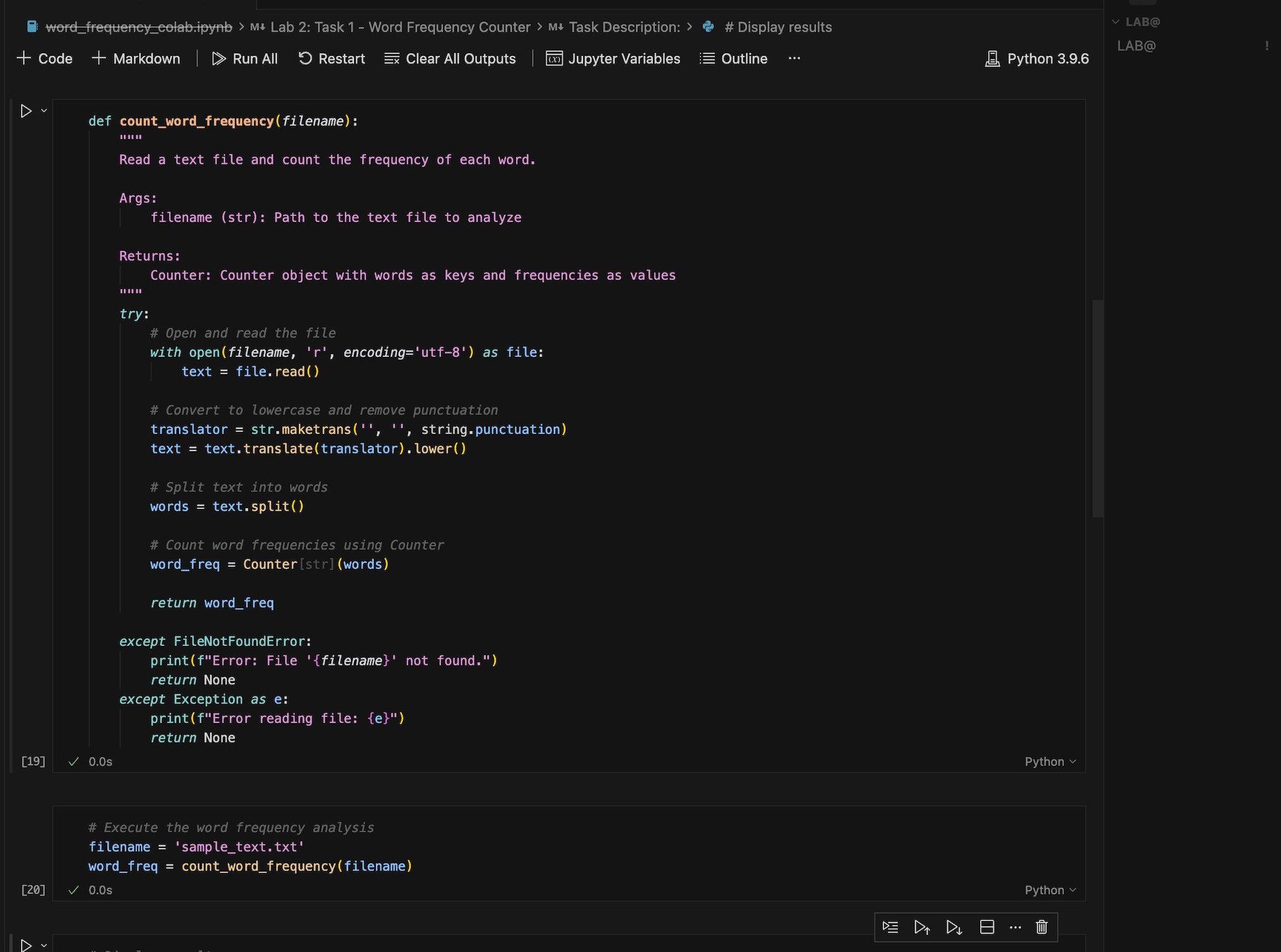
**Solution:**

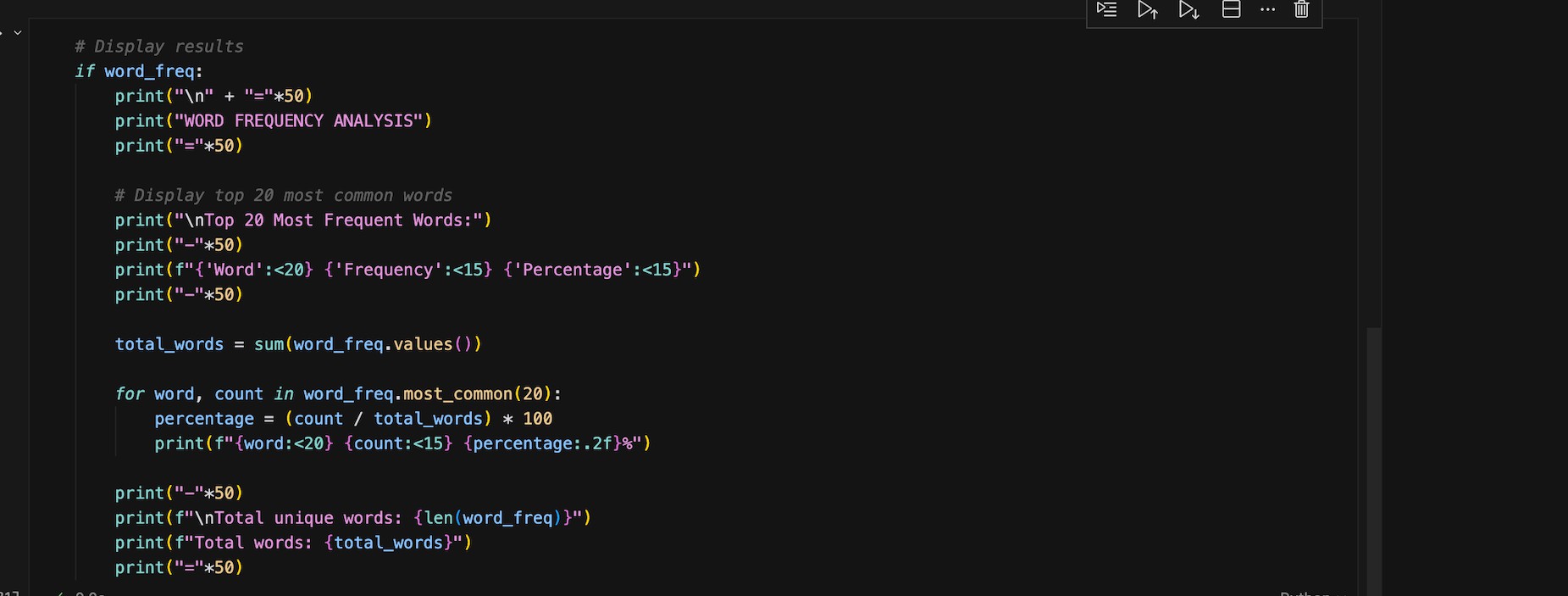
**PROMPT**

Generate a Python program in Google Colab that reads a text file and counts the frequency of each word.

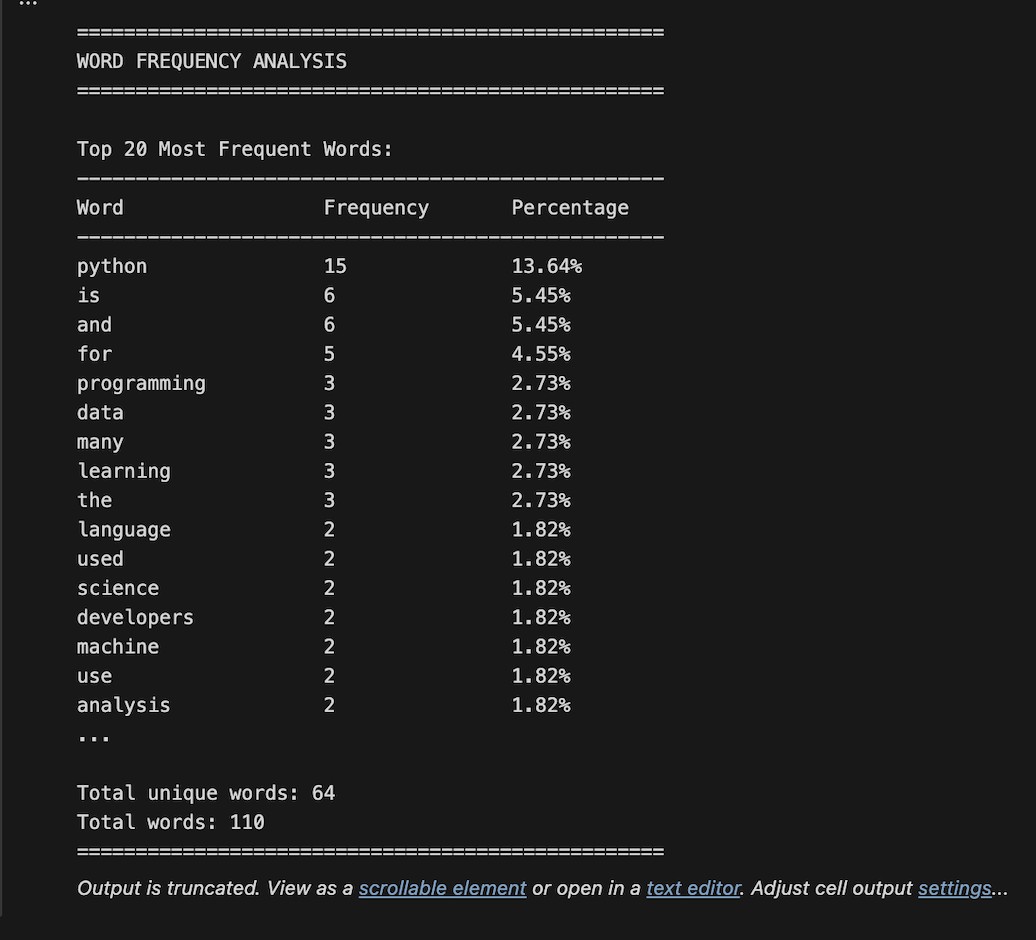
**CODE:**

****



****

**OUTPUT:**

****

**CODE Explanation:**

This Python program reads a text file and counts how many times each word appears. It removes punctuation

and converts all words to lowercase so the counting is accurate. The text is then split into words, and the

Counter function is used to find word frequencies. The program also handles errors like a missing file and

displays the results in a clear format.

Q) Task 2: File Operations Using Cursor AI

* + Scenario:

You are automating basic file operations.

* + Task:

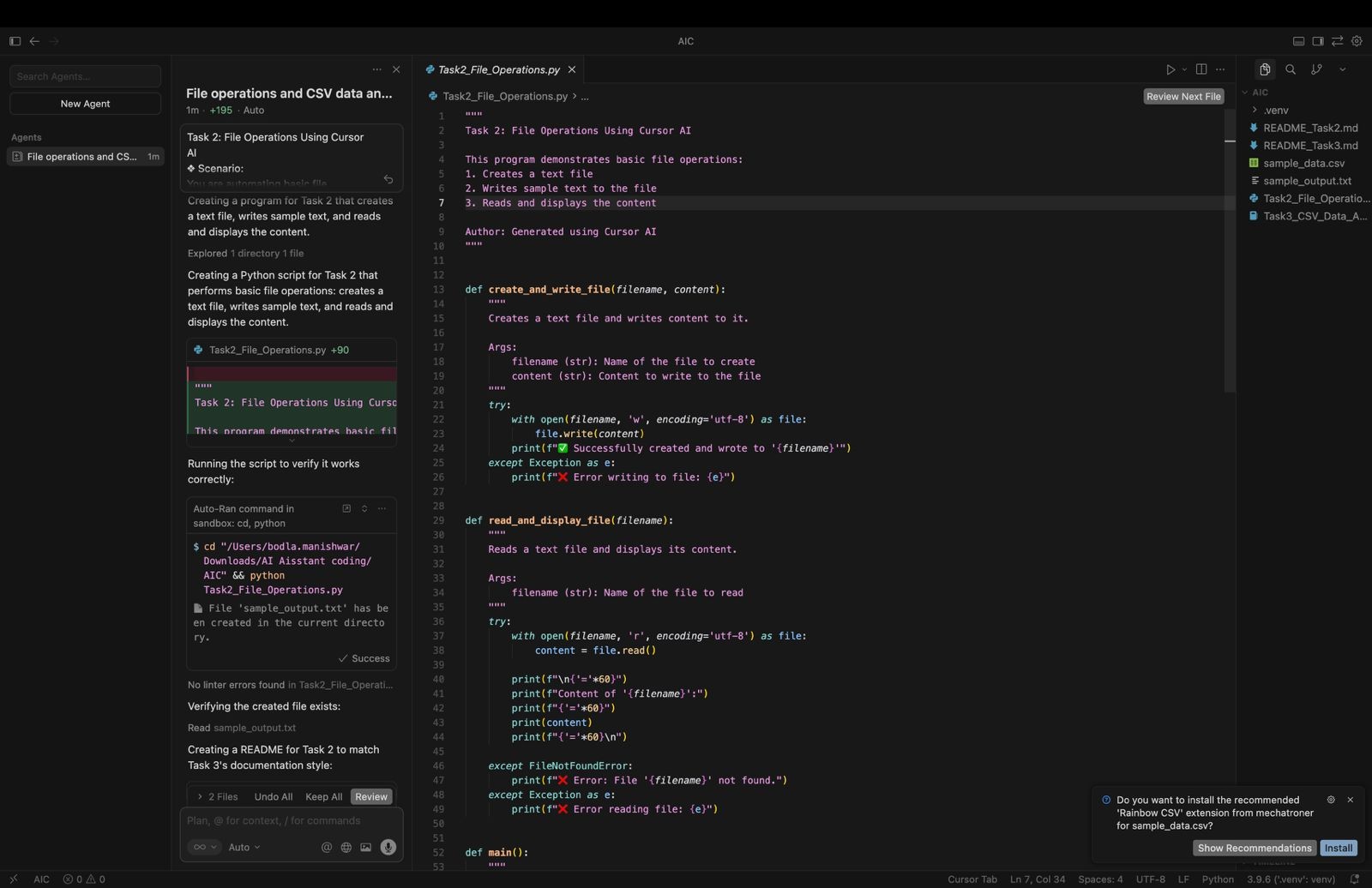
Use Cursor AI to generate a program that:

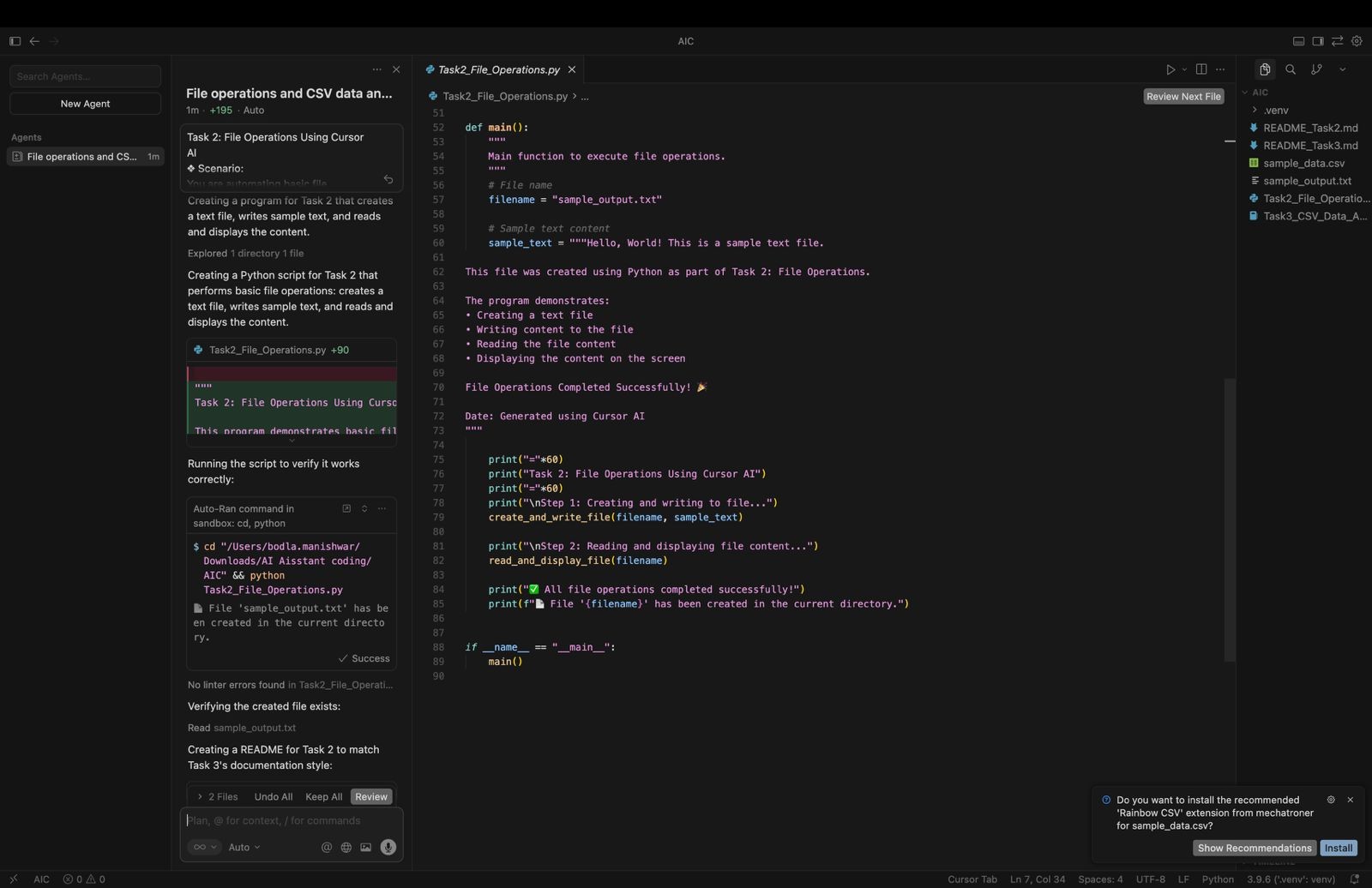
* Creates a text file
* Writes sample text
* Reads and displays the content
  + Expected Output:
* Functional code
* Cursor AI screenshots

**PROMPT:**

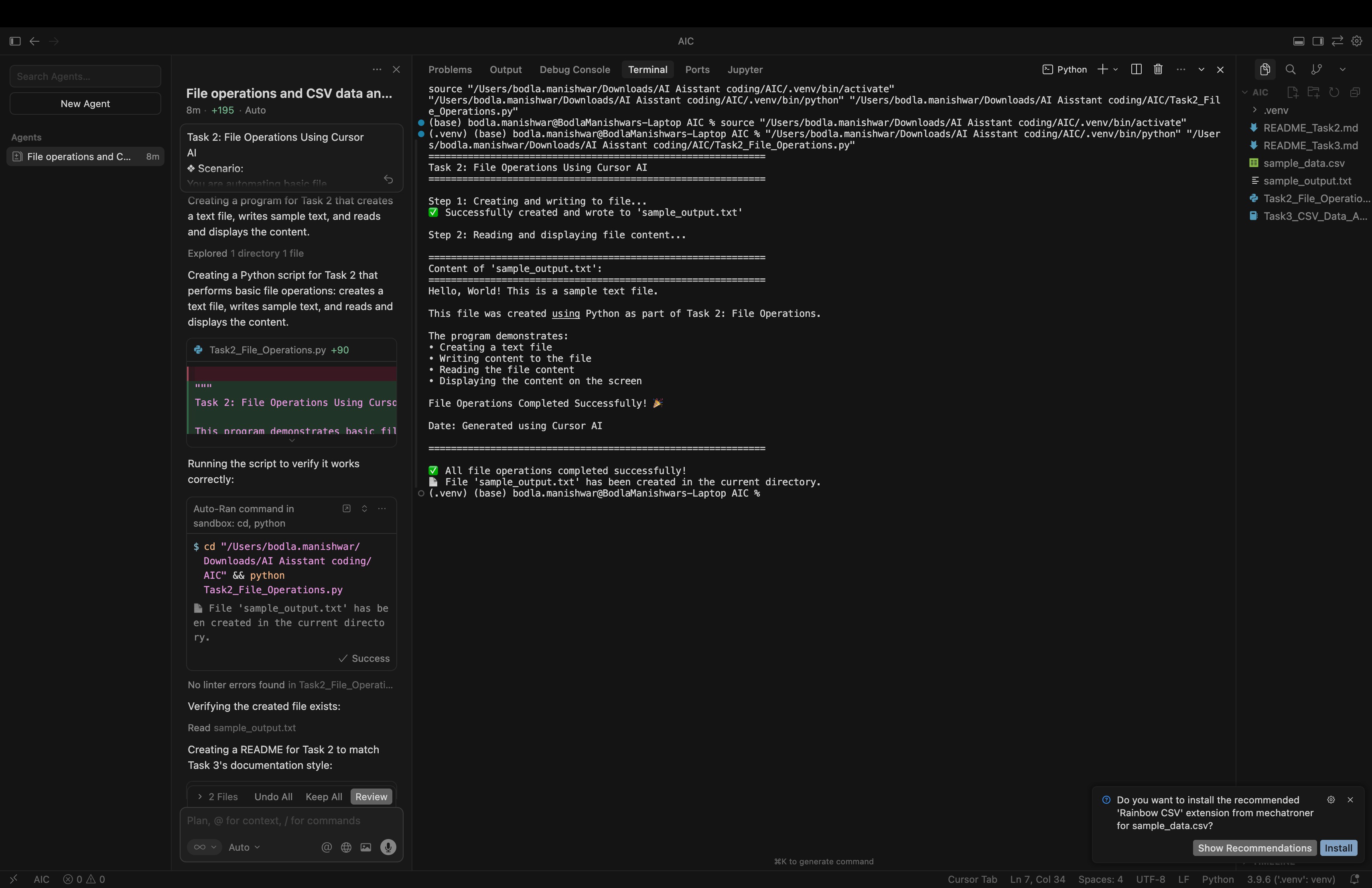
Generate a simple Python program that demonstrates basic file operations. The program should create a text file, write some sample text into it, then read the content from the file and display it on the screen.

**CODE:**

****



**OUTPUT:**

****

**CODE EXPLANATION:**

This Python program shows basic file handling by writing text to a file and then reading it. It uses separate functions for reading and writing to keep the code clear. Error handling is included to manage file-related issues. The main() function controls the program flow and runs only when the program is executed directly

**Q)Task 3: CSV Data Analysis**

* + **Scenario:**

**You are processing structured data from a CSV file.**

* + **Task:**

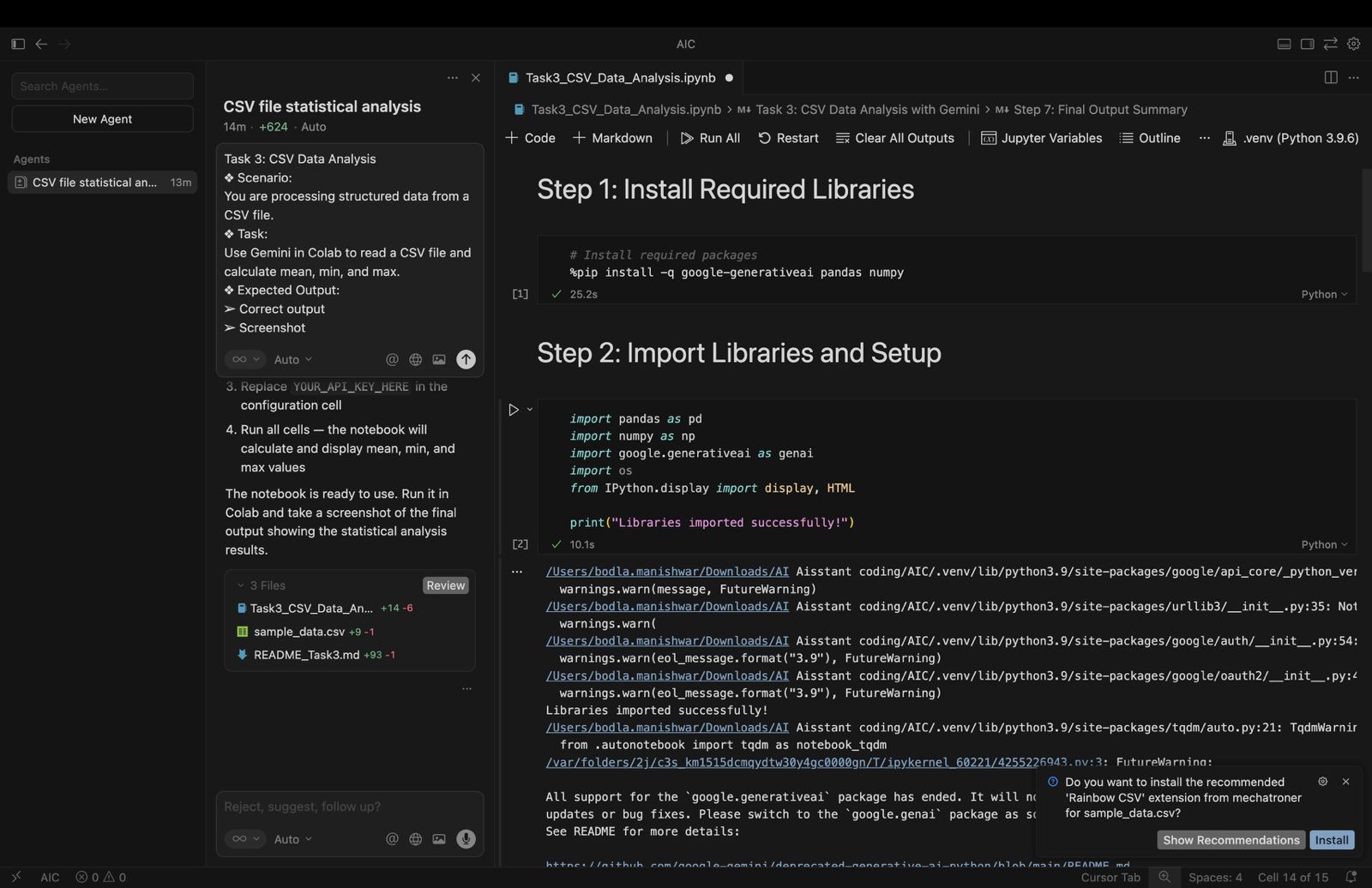
**Use Gemini in Colab to read a CSV file and calculate mean, min, and max.**

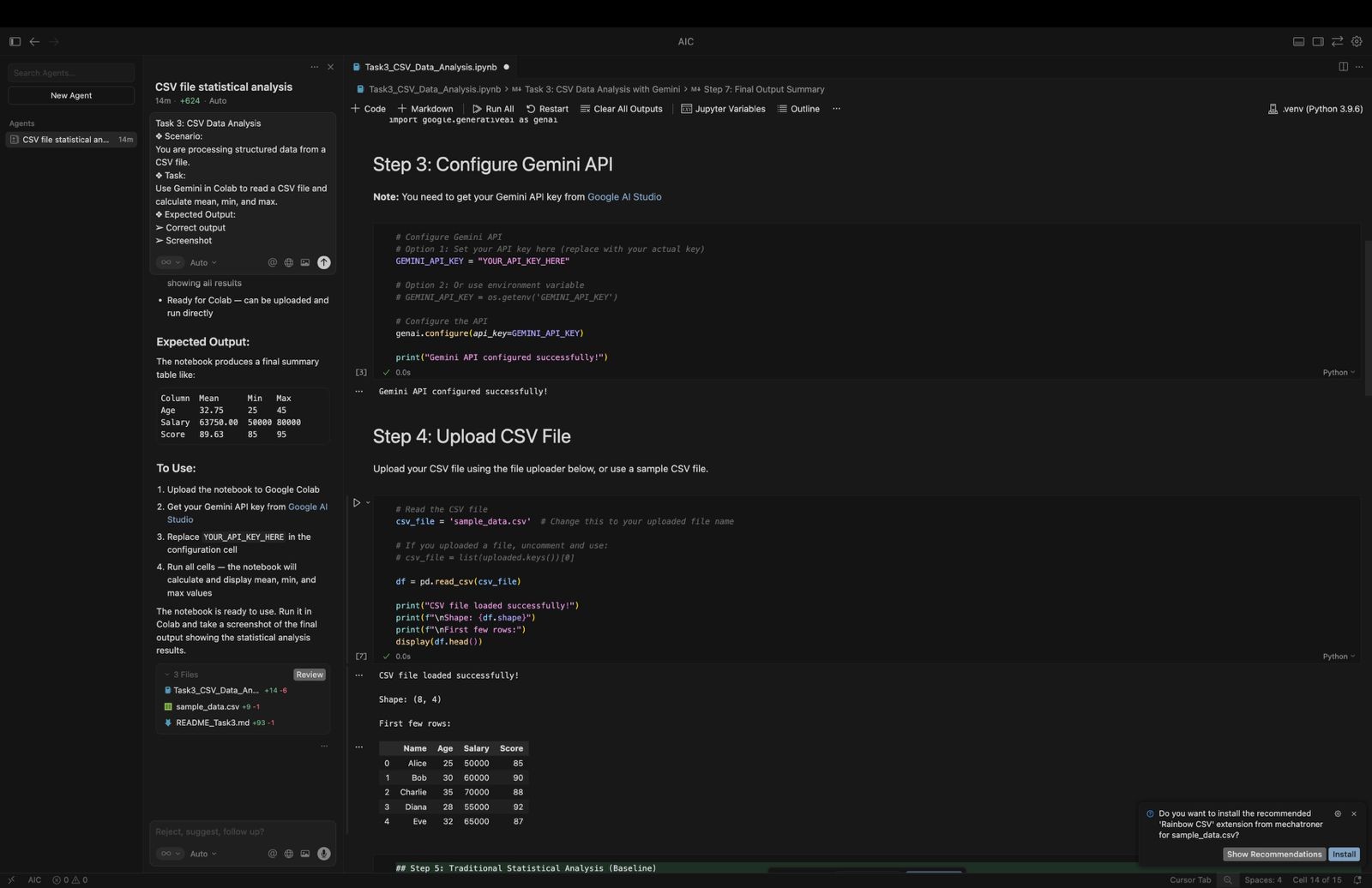
* + **Expected Output:**
* **Correct output**
* **Screenshot**

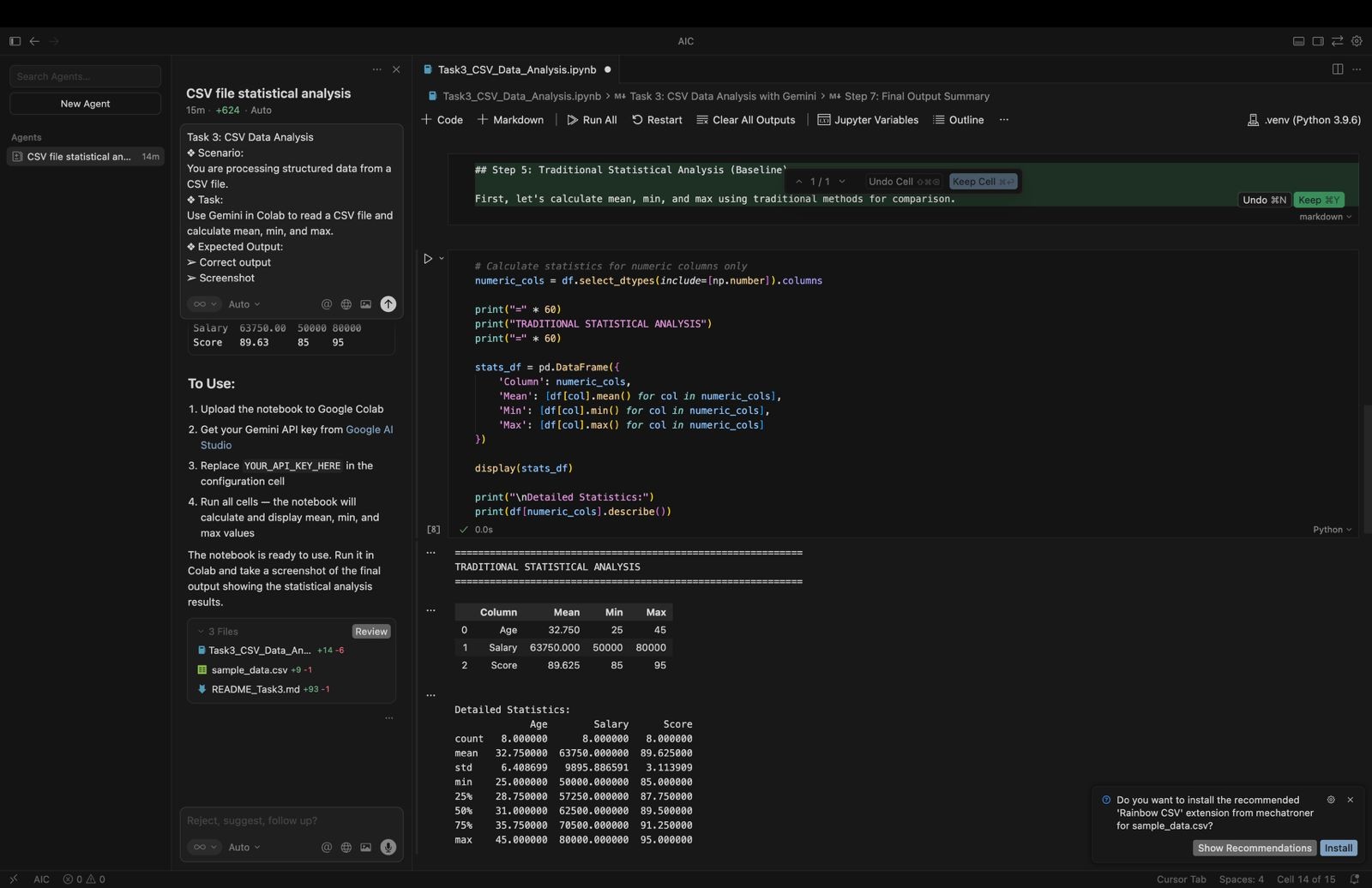
**PROMPT:**

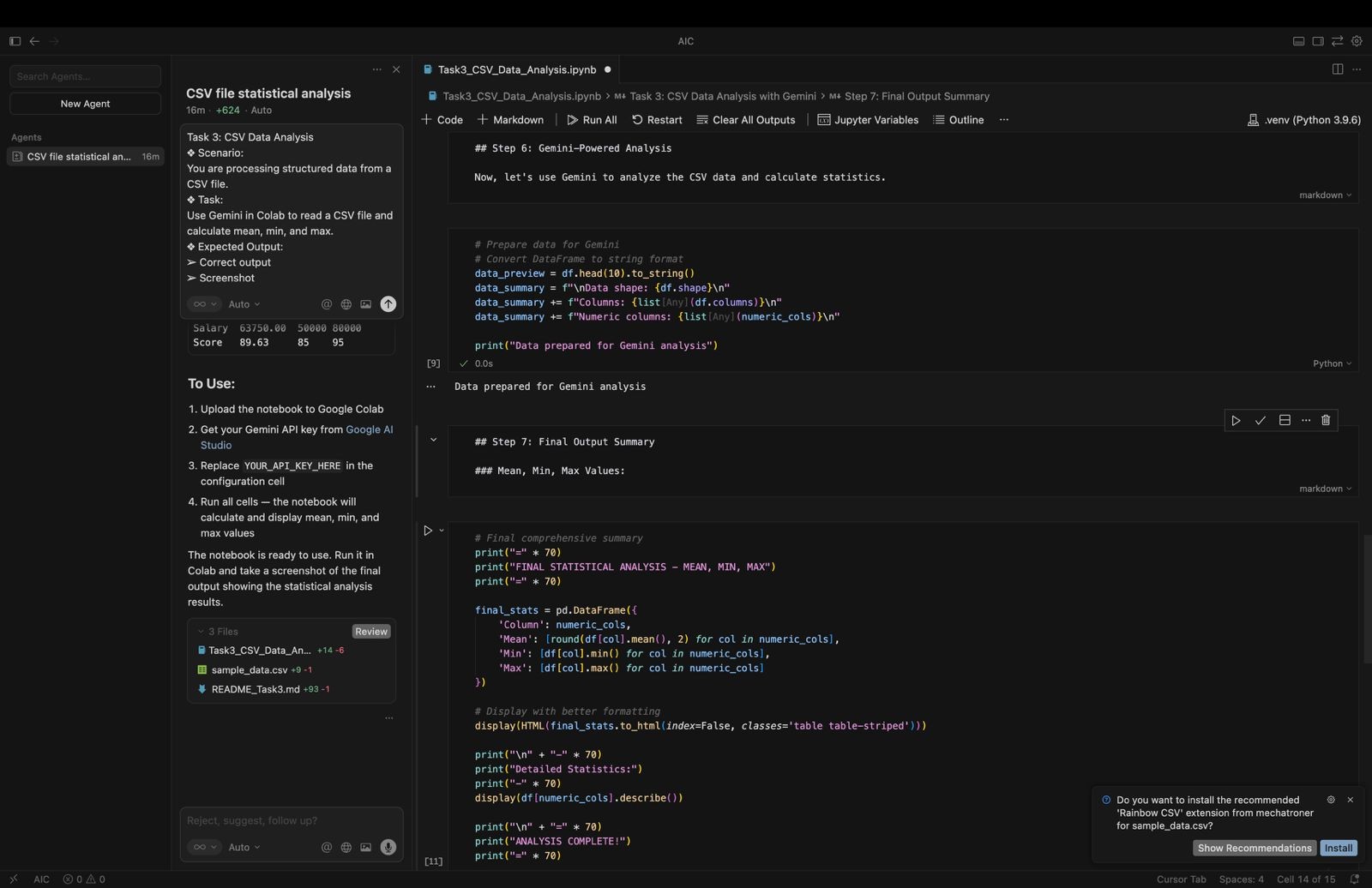
**Write Python code in Google Colab to read a CSV file and calculate mean, minimum, and maximum values using pandas.**

**CODE:**

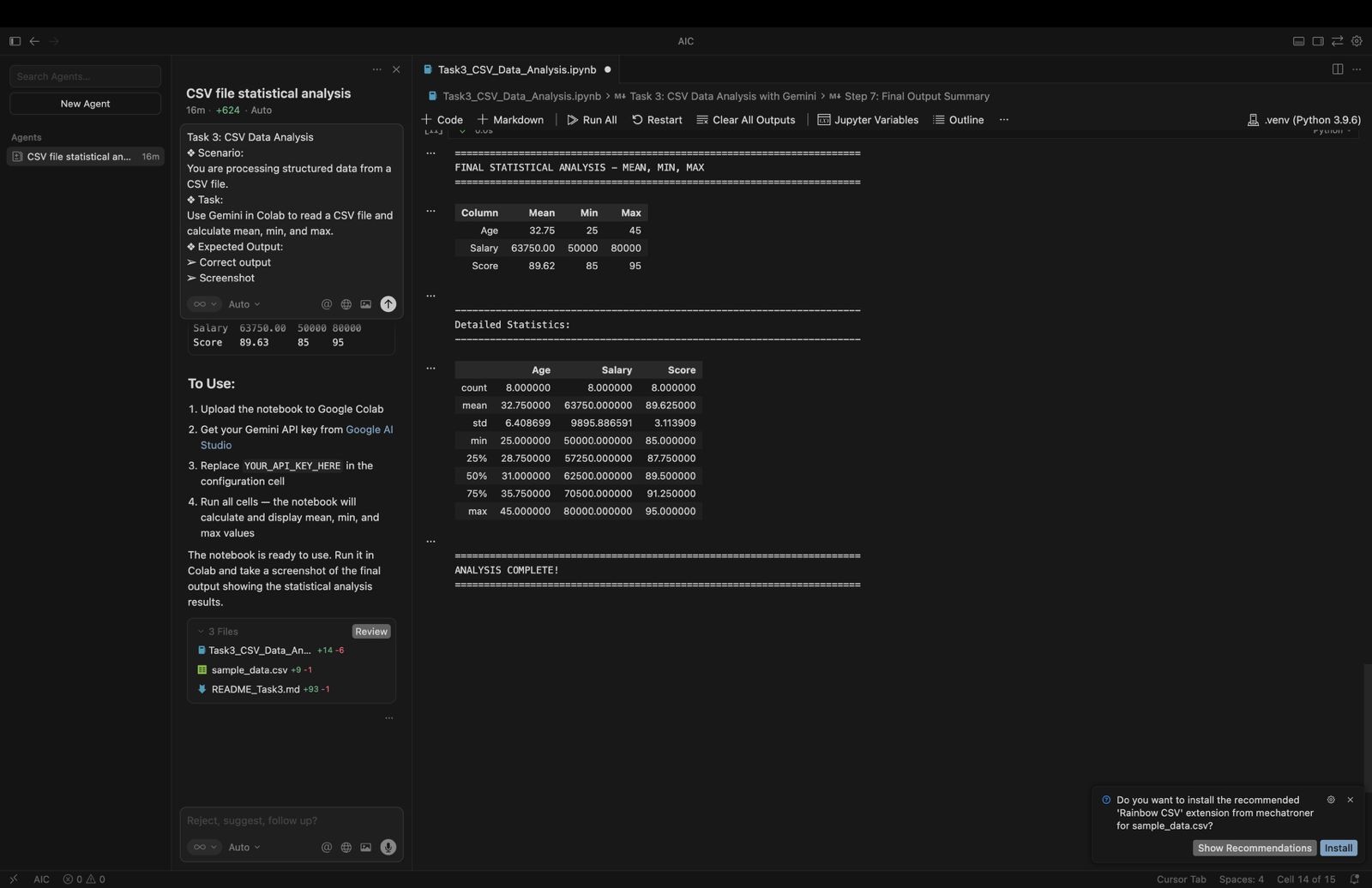
****



****



**OUTPUT:**

****

**CODE EXPLANATION:**

This code performs statistical analysis on numeric columns of a DataFrame (df). First, it identifies all columns that contain numerical data using select\_dtypes(include=[np.number]). Then, for each numeric column, it calculates the mean, minimum, and maximum values and stores them in a new DataFrame called stats\_df. This DataFrame is displayed to show a clean summary of basic statistics.

**Q)Task 4: Sorting Lists – Manual vs Built-in**

* + **Scenario:**

**You are reviewing algorithm choices for efficiency.**

* + **Task:**

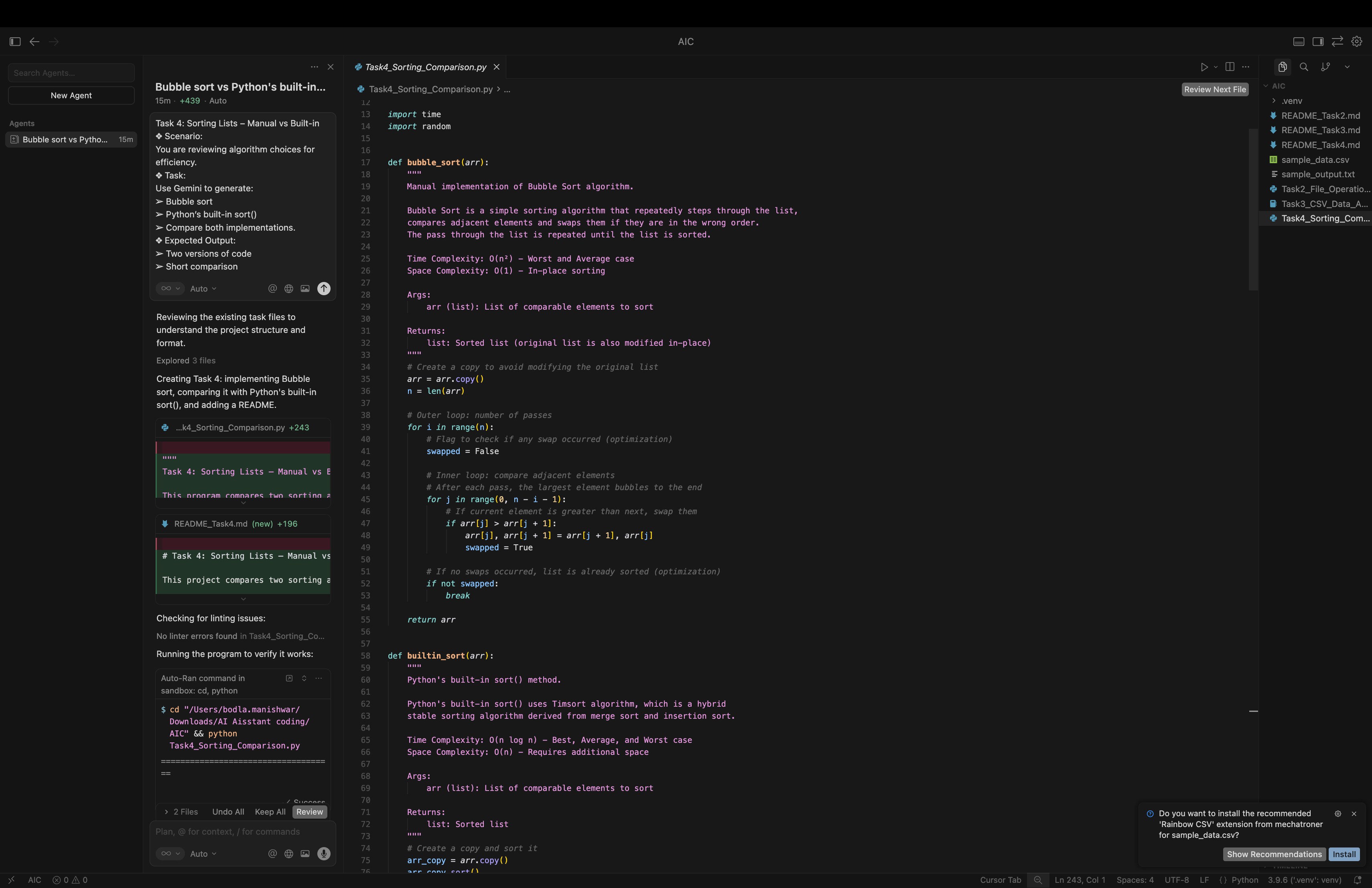
**Use Gemini to generate:**

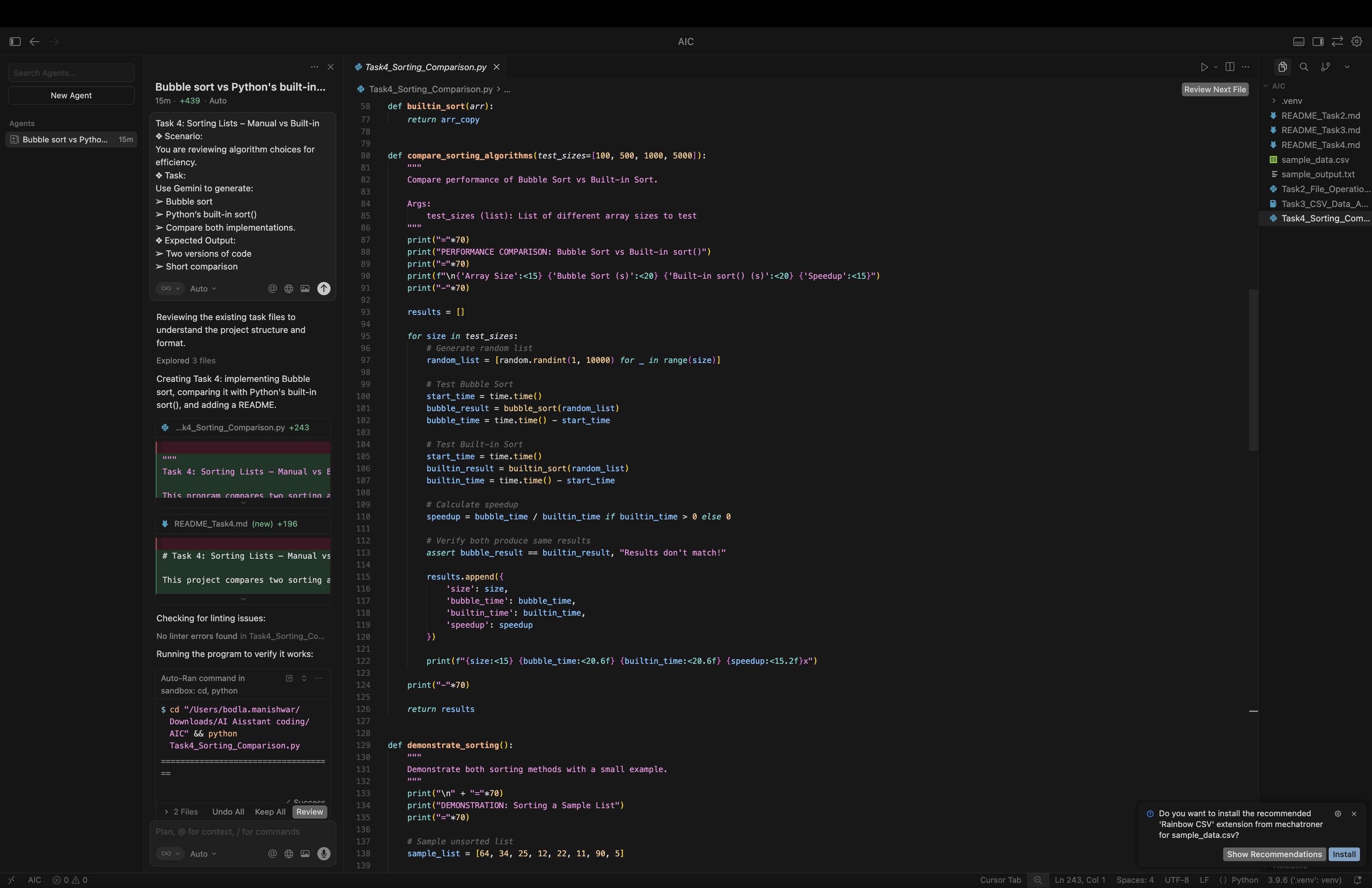
* **Bubble sort**
* **Python’s built-in sort()**
* **Compare both implementations.**
  + **Expected Output:**
* **Two versions of code**
* **Short comparison**

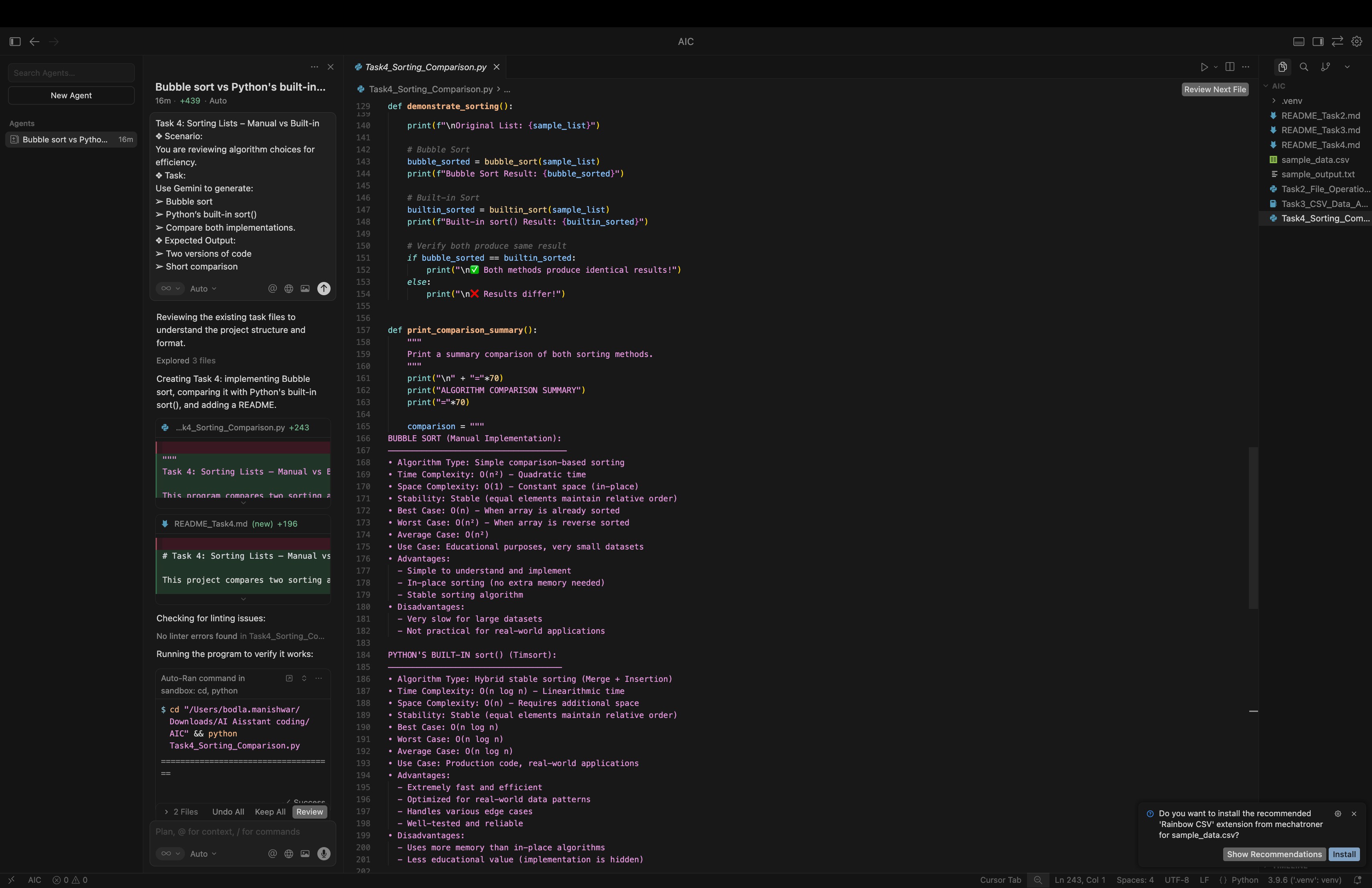
**PROMPT:**

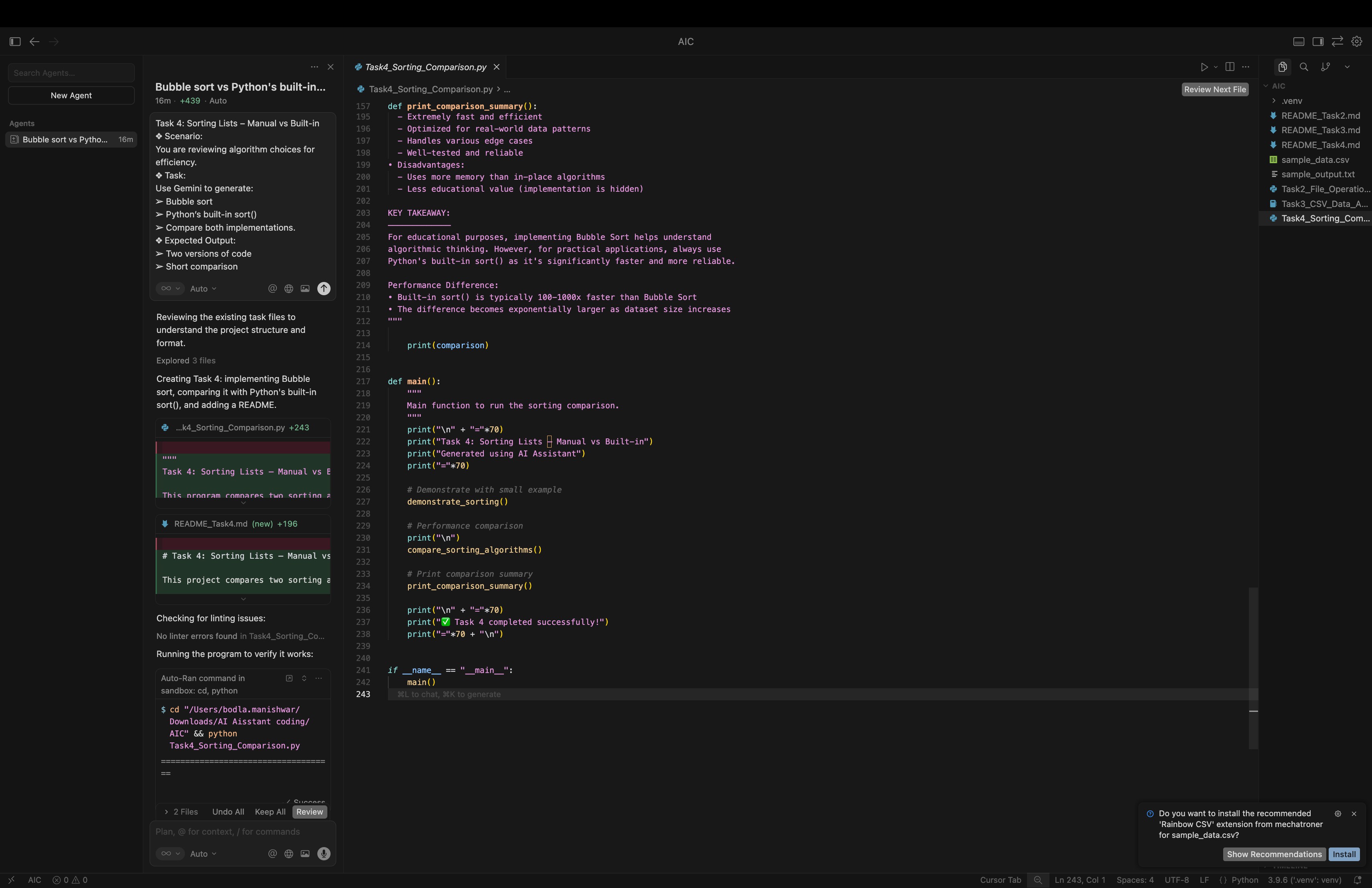
Generate Python code to sort a list using bubble sort and Python’s built-in sort() method. Show both implementations clearly and provide a short comparison explaining their efficiency and usage.

**CODE:**

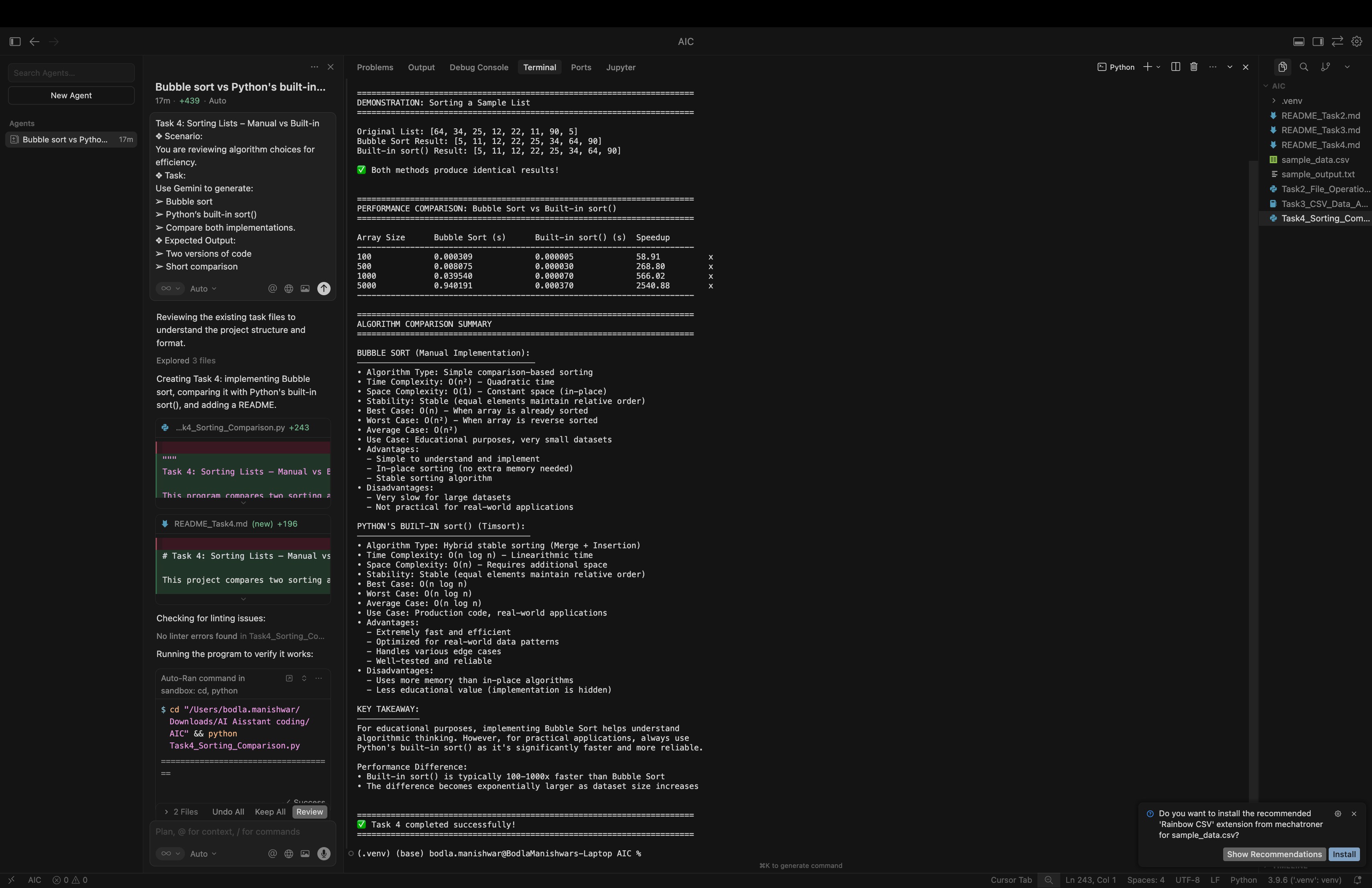
****



****



**OUTPUT:**

****

**CODE EXPLANATION:**

This program compares Bubble Sort with Python’s built-in sort(). Bubble Sort is slow because it has O(n²) time complexity, while the built-in sort() is faster with O(n log n). By measuring execution time, the program shows that Python’s built-in sort is more efficient and better for practical use.