AI ASSISTED CODING:

LAB-EXAM

M.KEERTHANA

2403A51259

BATCH-11

CSE-GEN

SET-G

F–G: TDD with AI:

Subgroup G:

G.1 — [S18G1] **Sum CSV column ignoring bad rows**

Context:

CSV exports in real estate listings platform have invalid numerics.

Your Task: Sum 'value' ints; skip invalid rows.

Data & Edge Cases: id,value

1,10

2,NA

3,7 -> 17.

AI Assistance Expectation:Use AI to draft csv.DictReader solution and tests.

Constraints & Notes: Optionally report skipped count.

Sample Input

id,value

1,10

2,NA

3,7

Sample Output

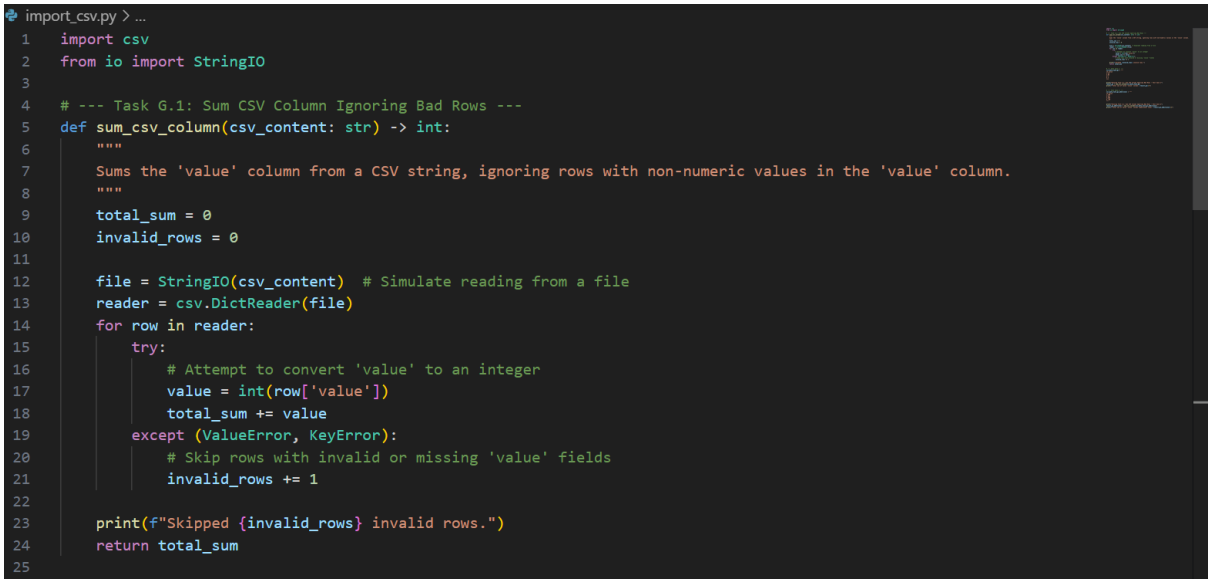
17

Acceptance Criteria: Skips invalid rows; correct total

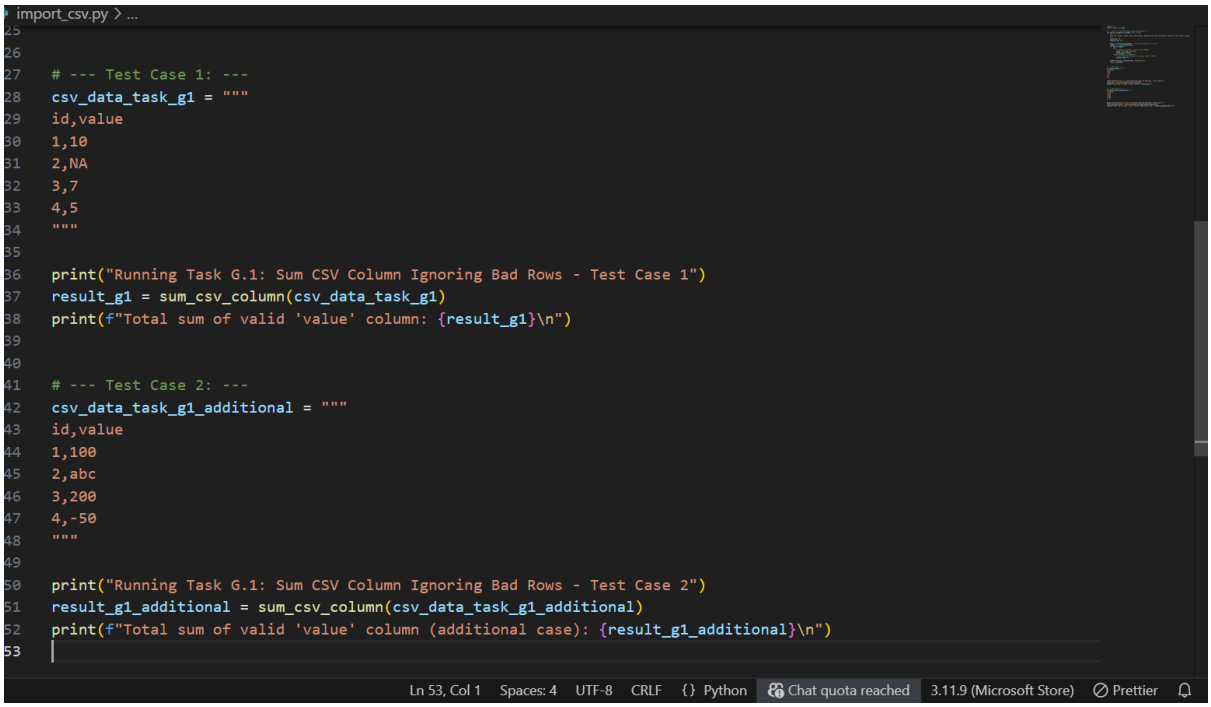
PROMPT:

"Write a Python function that reads a CSV file and sums the 'value' column, skipping any rows that contain non-numeric values or invalid data. Use csv.DictReader and report the total sum after processing the file."

CODE:



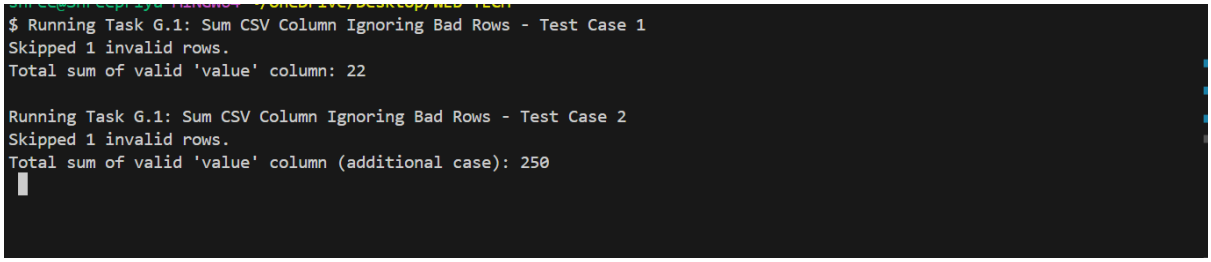
```python
import csv
from io import StringIO

# --- Task G.1: Sum CSV Column Ignoring Bad Rows ---
def sum_csv_column(csv_content: str) -> int:
    """
    Sums the 'value' column from a CSV string, ignoring rows with non-numeric values in the 'value' column.
    """
    total_sum = 0
    invalid_rows = 0

    file = StringIO(csv_content)  # Simulate reading from a file
    reader = csv.DictReader(file)
    for row in reader:
        try:
            # Attempt to convert 'value' to an integer
            value = int(row['value'])
            total_sum += value
        except (ValueError, KeyError):
            # Skip rows with invalid or missing 'value' fields
            invalid_rows += 1

    print(f"Skipped {invalid_rows} invalid rows.")
    return total_sum
```



```python
# --- Test Case 1: ---
csv_data_task_g1 = """
id,value
1,10
2,NA
3,7
4,5
"""

print("Running Task G.1: Sum CSV Column Ignoring Bad Rows - Test Case 1")
result_g1 = sum_csv_column(csv_data_task_g1)
print(f"Total sum of valid 'value' column: {result_g1}\n")


# --- Test Case 2: ---
csv_data_task_g1_additional = """
id,value
1,100
2,abc
3,200
4,-50
"""

print("Running Task G.1: Sum CSV Column Ignoring Bad Rows - Test Case 2")
result_g1_additional = sum_csv_column(csv_data_task_g1_additional)
print(f"Total sum of valid 'value' column (additional case): {result_g1_additional}\n")
```

Ln 53, Col 1 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached 3.11.9 (Microsoft Store) Prettier

OUTPUT:



G.2 — [S18G2] Merge two CSVs by id

Context:

Merge two CSVs in real estate listings platform by id.

Your Task:

Implement inner & left joins without pandas.

Data & Edge Cases:

A:id,price; B:id,qty.

AI Assistance Expectation:

Dict map + loops; add tests.

Constraints & Notes:

Stable order preferred.

Sample Input

id,price

A,10

B,20

id,qty

A,2

C,5

Sample Output

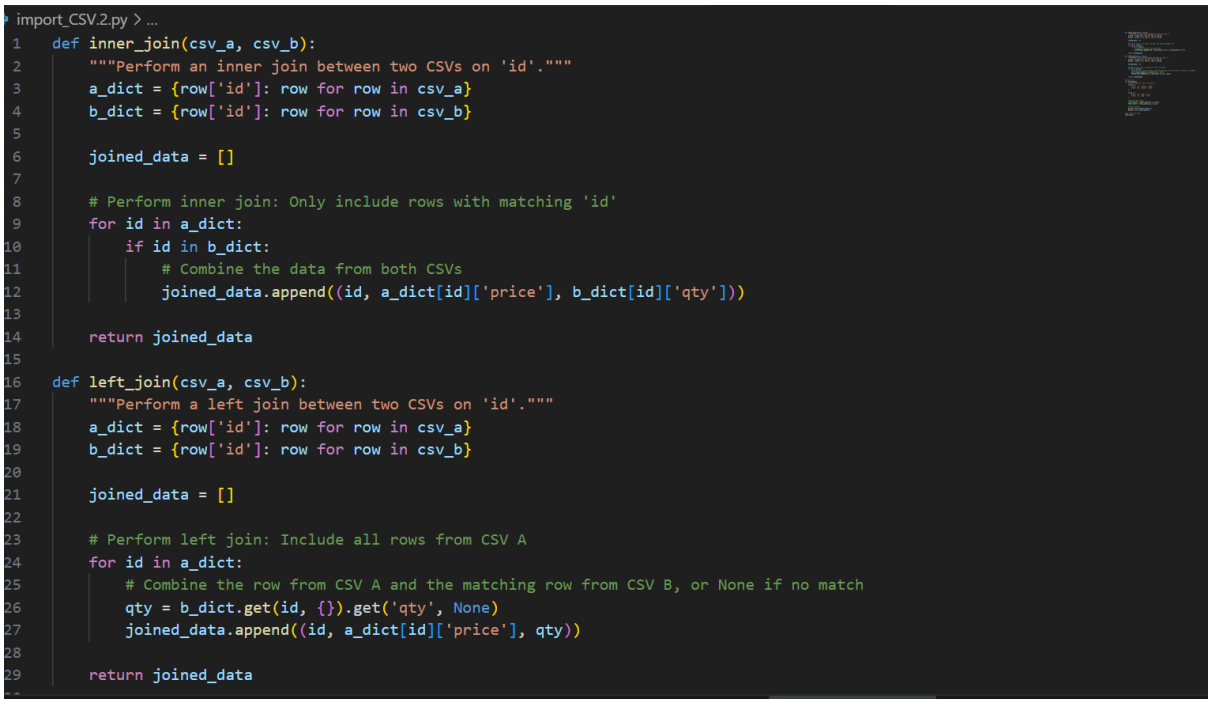
inner=[('A',10,2)], left=[('A',10,2),('B',20,None)]

Acceptance Criteria: Correct semantics

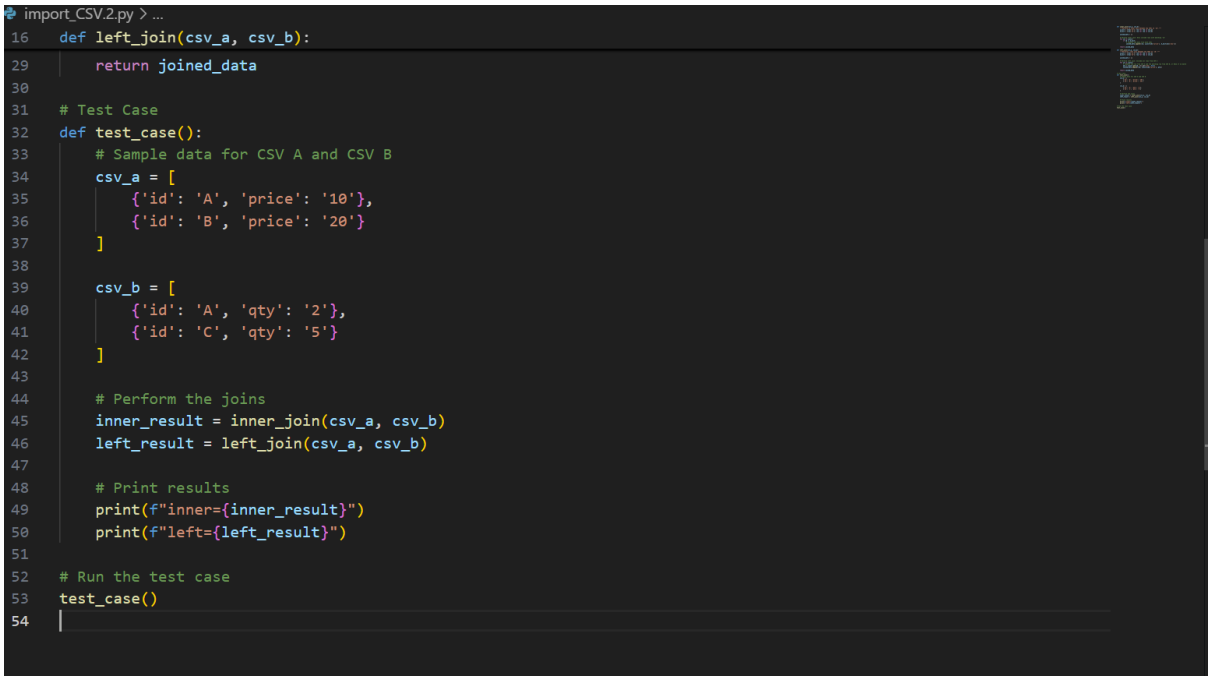
PROMPT:

"Write a Python function that reads a CSV file with columns id, name, and age. The function should return a list of dictionaries, skipping rows with missing fields or invalid age data. Ensure that missing or malformed age fields are handled gracefully, and assign a default value of 0 for missing age."

CODE:

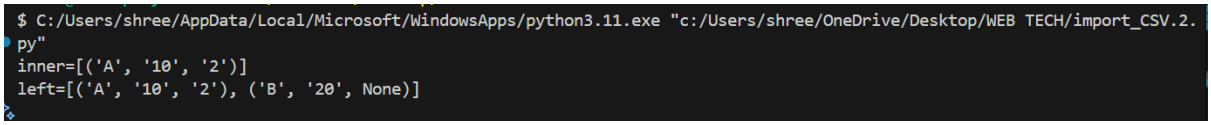


```python
def inner_join(csv_a, csv_b):
    """Perform an inner join between two CSVs on 'id'."""
    a_dict = {row['id']: row for row in csv_a}
    b_dict = {row['id']: row for row in csv_b}

    joined_data = []

    # Perform inner join: Only include rows with matching 'id'
    for id in a_dict:
        if id in b_dict:
            # Combine the data from both CSVs
            joined_data.append((id, a_dict[id]['price'], b_dict[id]['qty']))

    return joined_data

def left_join(csv_a, csv_b):
    """Perform a left join between two CSVs on 'id'."""
    a_dict = {row['id']: row for row in csv_a}
    b_dict = {row['id']: row for row in csv_b}

    joined_data = []

    # Perform left join: Include all rows from CSV A
    for id in a_dict:
        # Combine the row from CSV A and the matching row from CSV B, or None if no match
        qty = b_dict.get(id, {}).get('qty', None)
        joined_data.append((id, a_dict[id]['price'], qty))

    return joined_data
```



```python
def left_join(csv_a, csv_b):
    return joined_data

# Test Case
def test_case():
    # Sample data for CSV A and CSV B
    csv_a = [
        {'id': 'A', 'price': '10'},
        {'id': 'B', 'price': '20'}
    ]

    csv_b = [
        {'id': 'A', 'qty': '2'},
        {'id': 'C', 'qty': '5'}
    ]

    # Perform the joins
    inner_result = inner_join(csv_a, csv_b)
    left_result = left_join(csv_a, csv_b)

    # Print results
    print(f"inner={inner_result}")
    print(f"left={left_result}")

# Run the test case
test_case()
```

OUTPUT:


$ C:/Users/shree/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/shree/OneDrive/Desktop/WEB TECH/import_CSV.2.
py"
inner=[('A', '10', '2')]
left=[('A', '10', '2'), ('B', '20', None)]