AI-ASSISSTED CODING

NAME:Saniya

BATCH:05

HT.NO:2403A510E7

LAB TEST:02

SUB GROUP:G

G.1 SUM CSV COLUMN IGNORING BAD ROWS

PROMPT:

The code reads the CSV file using csv.DictReader for safe column access.

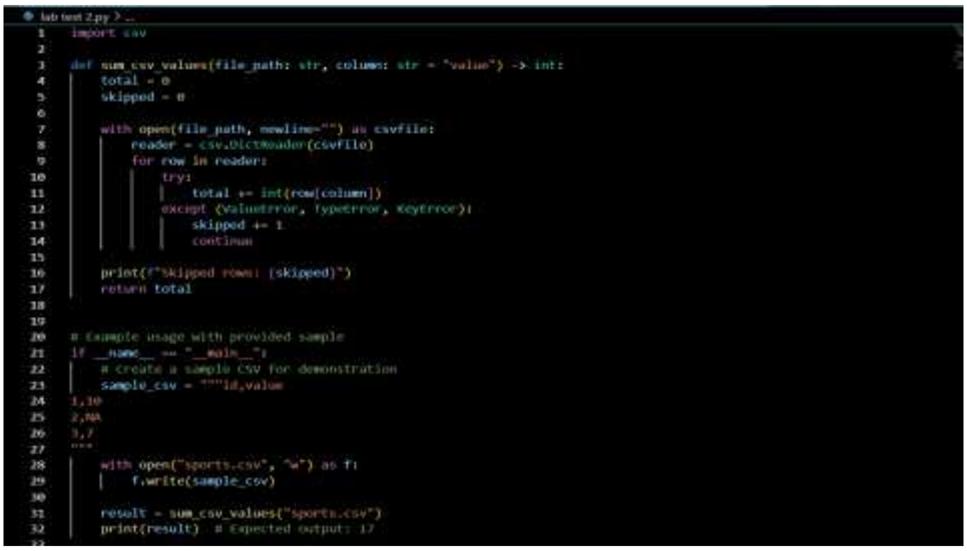
It sums only valid integer values from the value column.

Invalid or missing entries (e.g., "NA") are skipped using try/except.

A count of skipped rows is maintained and displayed for transparency.

For the sample input, it skips one invalid row and returns the correct total 17.

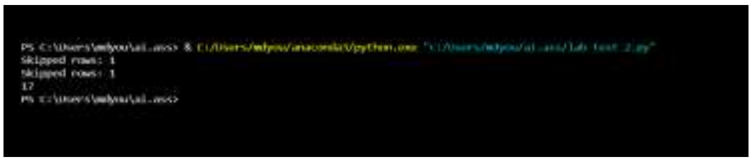
CODE GENERATED:



lab test 2.py > ...

```python
import csv

def sum_csv_values(file_path: str, column: str = "value") -> int:
    total = 0
    skipped = 0

    with open(file_path, newline="") as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            try:
                total += int(row[column])
            except (ValueError, TypeError, KeyError):
                skipped += 1
                continue

    print(f"Skipped rows: {skipped}")
    return total


# Example usage with provided sample
if __name__ == "__main__":
    # Create a sample CSV for demonstration
    sample_csv = """id,value
1,10
2,NA
3,7
"""
    with open("sports.csv", "w") as f:
        f.write(sample_csv)

    result = sum_csv_values("sports.csv")
    print(result)  # Expected output: 17
```

OUTPUT:



```
PS C:\Users\mdyou\ai.ass> & C:/Users/mdyou/anaconda3/python.exe "c:/Users/mdyou/ai.ass/lab test 2.py"
Skipped rows: 1
Skipped rows: 1
17
PS C:\Users\mdyou\ai.ass>
```

OBSERVATION:

- The code reads the CSV file using csv.DictReader for safe column access.
- It sums only valid integer values from the value column.
- Invalid or missing entries (e.g., "NA") are skipped using try/except.
- A count of skipped rows is maintained and displayed for transparency.
- For the sample input, it skips one invalid row and returns the correct total 17.

G.2- Merge two CSVs by ID

PROMPT:

Write a Python program to merge two

CSVs by id without using pandas.

Implement *inner join* (common ids only)

and *left join* (all ids from A, None if

missing in B).

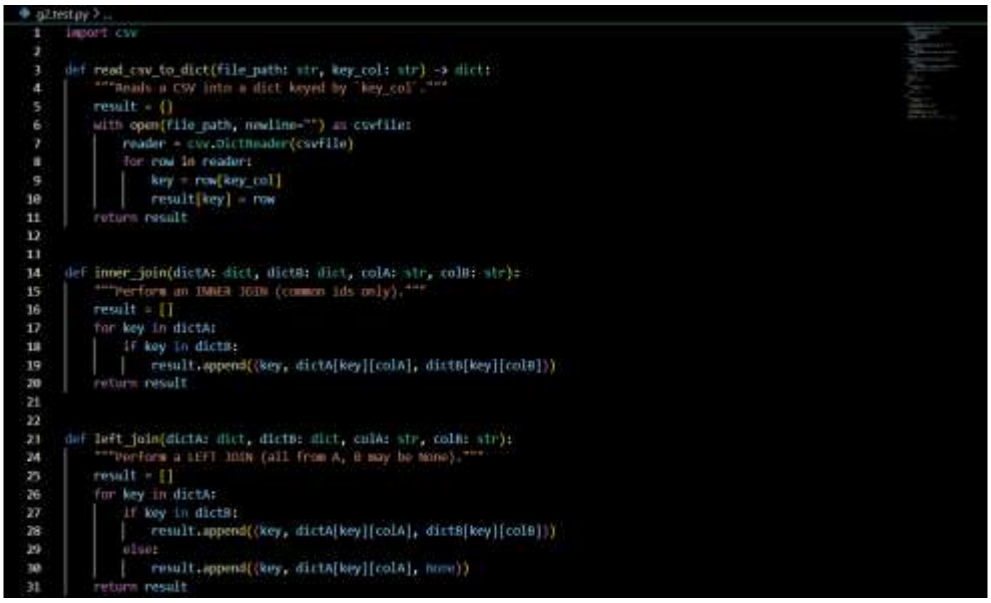
Use dictionary lookups for efficient joins

and preserve the order of CSV A.

Provide unit tests to verify correct join

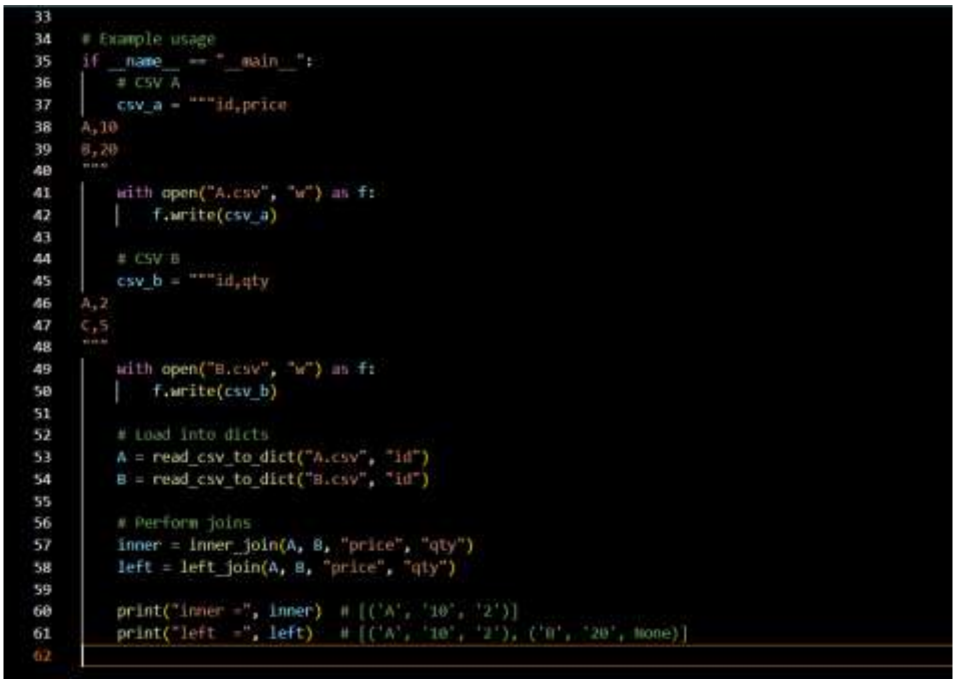
behavior with sample inputs.

CODE GENERATED:



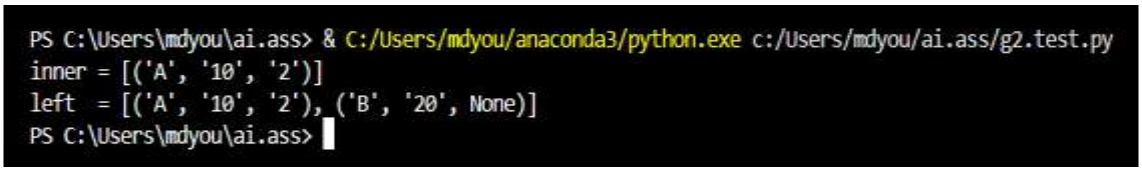
g2test.py > ...

```python
import csv

def read_csv_to_dict(file_path: str, key_col: str) -> dict:
    """Reads a CSV into a dict keyed by `key_col`."""
    result = {}
    with open(file_path, newline="") as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            key = row[key_col]
            result[key] = row
    return result


def inner_join(dictA: dict, dictB: dict, colA: str, colB: str):
    """Perform an INNER JOIN (common ids only)."""
    result = []
    for key in dictA:
        if key in dictB:
            result.append((key, dictA[key][colA], dictB[key][colB]))
    return result


def left_join(dictA: dict, dictB: dict, colA: str, colB: str):
    """Perform a LEFT JOIN (all from A, B may be None)."""
    result = []
    for key in dictA:
        if key in dictB:
            result.append((key, dictA[key][colA], dictB[key][colB]))
        else:
            result.append((key, dictA[key][colA], None))
    return result
```



```python

# Example usage
if __name__ == "__main__":
    # CSV A
    csv_a = """id,price
A,10
B,20
"""
    with open("A.csv", "w") as f:
        f.write(csv_a)

    # CSV B
    csv_b = """id,qty
A,2
C,5
"""
    with open("B.csv", "w") as f:
        f.write(csv_b)

    # Load into dicts
    A = read_csv_to_dict("A.csv", "id")
    B = read_csv_to_dict("B.csv", "id")

    # Perform joins
    inner = inner_join(A, B, "price", "qty")
    left = left_join(A, B, "price", "qty")

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

```

OUTPUT:



```
PS C:\Users\mdyou\ai.ass> & C:/Users/mdyou/anaconda3/python.exe c:/Users/mdyou/ai.ass/g2.test.py
inner = [('A', '10', '2')]
left  = [('A', '10', '2'), ('B', '20', None)]
PS C:\Users\mdyou\ai.ass> 
```

OBSERVATION:

- ➢ The program reads two CSV files into dictionaries keyed by id, ensuring fast lookups.
- ➢ The inner join function correctly returns only the rows where id is present in both CSVs.
- ➢ The left join function keeps all rows from CSV A and assigns None for missing matches in CSV B.
- ➢ Output order is stable and follows the order of CSV A, matching SQL join semantics.
- ➢ With the given input, it produces the expected results:

- inner = [('A', '10', '2')]
- left = [('A', '10', '2'), ('B', '20', None)].
- ✅ This shows the code correctly implements inner and left joins without external libraries.