Project Title: GPU-Accelerated Joins on CSV Files Using CUDA

To reiterate on the project, I am planning on leveraging GPU parallelism using CUDA to speed up the process of performing "SQL-like" joins on CSV files, focusing on efficiency for large-scale data.

(reiterating) Goals:

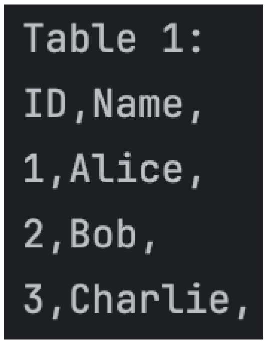
- 75% Goal: A basic INNER JOIN will be implemented using CUDA to join two CSV tables on a common key column.
- 100% Goal: This implementation will be extended to support both LEFT and RIGHT JOINS using CUDA, where the LEFT JOIN will include all rows from the left CSV file, with NULLs where there's no match in the right file (and vice versa for the RIGHT JOIN).
- 125% Goal: This implementation will support full OUTER JOIN operations or complex multi-join operations that support joining 3+ CSV tables at once.

Progress:

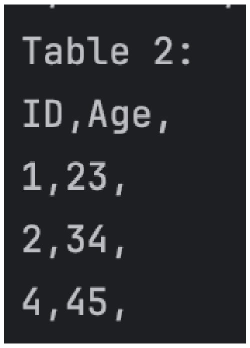
Completed subtasks:
[note that these are completed subtasks WITHOUT the use of CUDA]

- Able to Create CSV file based on vector data.
- Able to read CSV data from a file and store it into a vector.
- Able to take two tables (through unordered maps) and perform a simple inner join on them based on common table characteristics.

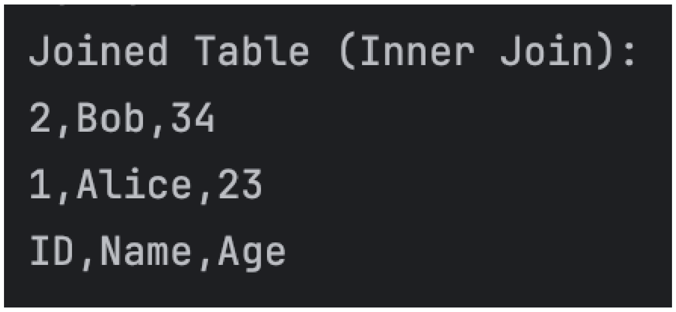
Example output:



```
Table 1:
ID,Name,
1,Alice,
2,Bob,
3,Charlie,
```

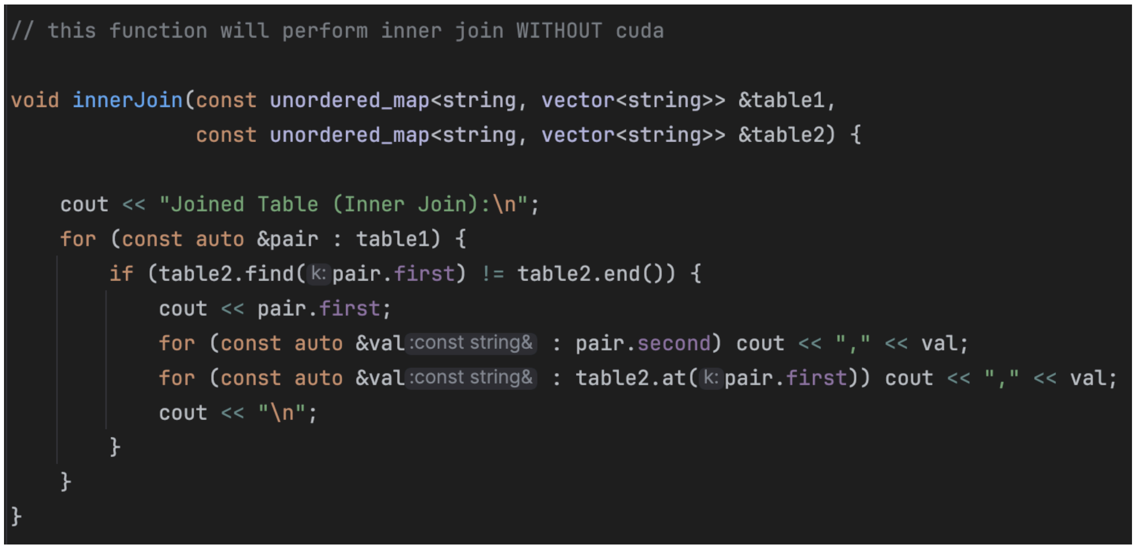


```
Table 2:
ID,Age,
1,23,
2,34,
4,45,
```



```
Joined Table (Inner Join):
2,Bob,34
1,Alice,23
ID,Name,Age
```

Example of current inner join function (without CUDA):



```cpp
// this function will perform inner join WITHOUT cuda

void innerJoin(const unordered_map<string, vector<string>> &table1,
               const unordered_map<string, vector<string>> &table2) {

    cout << "Joined Table (Inner Join):\n";
    for (const auto &pair : table1) {
        if (table2.find(k: pair.first) != table2.end()) {
            cout << pair.first;
            for (const auto &val :const string& : pair.second) cout << "," << val;
            for (const auto &val :const string& : table2.at(k: pair.first)) cout << "," << val;
            cout << "\n";
        }
    }
}
```

Remarks and Notes on Progress:

I would say that regarding goals, this is around **45% complete.** The basic functionality/framework is more or less implemented, allowing me to focus on the CUDA programming at this point. However, there are a few things I would like to check (the remaining 5% before 50%) to ensure that the testing will work on larger tables. For instance:

- Unsuccessful join on two tables with no common id
- Unsuccessful join on two tables with no common attribute
- [BUG] Inner join prints table in reverse order
- Successful join on large / very large table size
    - Large: 1000+ rows
    - Very large: 10000+ rows

The first two points just ensure consistency when using this for later applications especially when implementing the later join logic. Furthermore, they should be relatively quick checks that can be completed without too much effort.

The third point is simply a bug showing how the inner join prints the table in reverse order. This should also not be a complex fix.

The last point is essential for later measuring the effectiveness of CUDA. As we know, the CPU is generally superior to GPU computations regarding smaller computational sizes, because for smaller computations the CPU doesn't have to worry about the overhead of moving data to and from CPU and GPU, which causes the GPU to be slower for smaller numbers. For instance, during my GPU project where we sorted a large matrix, the matrix was a square matrix of size 10,000,000. I don't believe we need to test on this scale, since we are reading/writing to a file, but for later purposes in recording metrics, it will be important to support larger file sizes, since generally this is the merit of using GPU and CUDA-based programming anyways.

Conclusion / Next Steps:

As aforementioned, the focus now is to solidify the coding framework and ensure support for larger files. After that, the CUDA incorporation–the true essence of the project–can begin. Implementing inner join with CUDA will be the first large accomplishment, which will be achieving the 75% goal. I will also need to add some metric support for starting / stopping a timer to measure CPU and GPU difference. I had to do this often in my GPU projects, so I will draw inspiration from those frameworks.

Overall, there were no major changes I made from the scope of the project proposal.

I am confident, comfortable, and excited to continue working on the project and achieve the 125% goal.