In Memory Database

Problem

Implement basic SQL-like operations, such as select, where, join, etc., of an in-memory database.

Instructions

Assume the following data is stored in the in-memory database

departments

id	name
0	engineering
1	finance

users

id	department_id	name
0	0	Ian
1	0	John
2	1	Eddie
3	1	Mark

salaries

id	user_id	amount
0	0	100
1	1	150
2	1	200
3	3	200
4	3	300
5	4	400

Building the binary

For Linux and Mac users, you can use the included Makefile to compile and run the project.

```
# Compile only
make
# Compile and run
make run
```

For Windows Visual Studio users, a vcproj template is included. Alternatively you can compile and run with the included Makefile.vs in Developer Command Prompt:

```
# cd to project directory
# Compile only
nmake -f Makefile.vs
# Compile and run
nmake -f Makefile.vs run
```

First step

Please implement `Select` and `Where` in `Table` class. For the sake of simplicity, you can assume all operations will be passed valid parameters, and all values are of type `string`. You do not need to concern yourself with error handling for occurrences like selecting an arbitrary column from a non-existent table.

For example,

```
// should print
// id, department_id, name
// 1, 0, John
Table* filtered_table = db->GetTable("users")->Where("id", "1");
Table* projected_table = filtered_table->Select(
    ToVector((string[]) { "id", "department_id", "name" }));
projected_table->Print();
```

Second step

Please implement `InnerJoin` in `Database` class.

```
// should print
// users.name, departments.name
// Ian, engineering
// John, engineering
Table* table = db->InnerJoin(
    db->GetTable("users"),
    "department_id",
    db->GetTable("departments"),
```

```
    "id");
Table* filtered_table = table->Where("departments.name", "engineering");
Table* projected_table = filtered_table->Select(
    ToVector((string[]) { "users.name", "departmetns.name" }));
projected_table->Print();
```

Third step

Please implement `LeftJoin`, `RightJoin` and `OuterJoin` in `Database` class, and use empty string "" for the missing values.

```
// should print
// users.name, salaries.amount
// Ian, 100
// John, 150
// John, 200
// Mark, 200
// Mark, 300
// Eddie,
Table* table = db->LeftJoin(
    db->GetTable("users"),
    "id",
    db->GetTable("salaries"),
    "user_id");
Table* projected_table = table->Select(
    ToVector((string[]) { "users.name", "salaries.amount" }));
projected_table->Print();

// should print
// users.name, salaries.amount
// Ian, 100
// John, 150
// John, 200
// Mark, 200
// Mark, 300
//     , 400
Table* table = db->RightJoin(
    db->GetTable("users"),
    "id",
    db->GetTable("salaries"),
    "user_id");
Table* projected_table = table->Select(
    ToVector((string[]) { "users.name", "salaries.amount" }));
projected_table->Print();
```

```
// should print
// users.name, salaries.amount
// Ian, 100
// John, 150
// John, 200
// Mark, 200
// Mark, 300
// Eddie,
// , 400
Table* table = db->OuterJoin(
    db->GetTable("users"),
    "id",
    db->GetTable("salaries"),
    "user_id");
Table* projected_table = table->Select(
    ToVector((string[]) { "users.name", "salaries.amount" }));
projected_table->Print();
```

Your solution will be scored in the descending priority of correctness, memory & time complexity, and comprehensibility.

Submission

Upon completion, please follow the instructions described in the website (where you found the instructions to download the project) to submit your solution. You can submit as many times as you prefer. Your last submission will be used for evaluation as well as marking the end of your coding assessment.

Lastly, do not be concerned if you are running a little bit over time (0-10 minutes). We do not penalize moderately tardy submissions.