CSV Query

This interview question asks you to write a small command-line for querying CSV files. You are expected to produce a working implementation of most of the listed features during the interview timeframe.

Introduction

You are provided 3 `.csv` files: `city.csv`, `country.csv`, and `language.csv`. The task is to write a small query command-line that a user can enter queries to answer questions.

This query finds the 7 cities in the dataset with the largest population:

```
> FROM city.csv ORDERBY CityPop TAKE 7 SELECT CityName,CityPop
CityName,CityPop
Mumbai_(Bombay),10500000
Seoul,9981619
Sâ€žo_Paulo,9968485
Shanghai,9696300
Jakarta,9604900
Karachi,9269265
Istanbul,8787958
```

Your task is to implement a program take reads commands from the user from the command-line, and performs these queries on the given `.csv` files. The precise behavior of these queries is specified in the following sections

Exercise notes:

- You are expected to demonstrate running, working code
- You may use any language you are comfortable with
- You may look up any online resources you want, but your code should be your own

For the purposes of this exercise, focus first on correctness with edge cases in mind and don't worry about performance or scalability -- all data will easily fit in memory.

Take-home note: The final submission is expected to contain comprehensive unit tests and to be of a quality "ready for code review". The quality of your code and testing is more important than finishing all parts of the spec.

Specification

Queries are space-separated lists of input tokens. For the purpose of this exercise, we will assume that none of the inputs contain any whitespace or other special characters that require escaping, and the input can be tokenized by a simple `String.split(" ")`.

The input `.csv` files' entries also do not contain any spaces, commas, or other special characters, and each line can be split into its entries by a simple `String.split(",")`

FROM

```
> FROM city.csv
CityID,CityName,CountryCode,CityPop
1,Kabul,AFG,1780000
2,Qandahar,AFG,237500
3,Herat,AFG,186800
4,Mazar-e-Sharif,AFG,127800
5,Amsterdam,NLD,731200
6,Rotterdam,NLD,593321
7,Haag,NLD,440900
...
```

```
> FROM country.csv
CountryCode,CountryName,Continent,CountryPop,Capital
ABW,Aruba,North_America,103000,129
AFG,Afghanistan,Asia,22720000,1
AGO,Angola,Africa,12878000,56
AIA,Anguilla,North_America,8000,62
ALB,Albania,Europe,3401200,34
AND,Andorra,Europe,78000,55
ANT,Netherlands_Antilles,North_America,217000,33
ARE,United_Arab_Emirates,Asia,2441000,65
```

```
...
```

```
> FROM language.csv
CountryCode,Language
ABW,Dutch
ABW,English
ABW,Papiamento
ABW,Spanish
AFG,Balochi
AFG,Dari
AFG,Pashto
AFG,Turkmenian
...
```

The `FROM` command loads a CSV dataset from disk and prints its contents to standard output, preserving the same column ordering as the file.

SELECT

```
> FROM city.csv SELECT CityName
CityName
Kabul
Qandahar
Herat
Mazar-e-Sharif
Amsterdam
Rotterdam
...
```

```
> FROM country.csv SELECT CountryCode,Continent,CountryPop
CountryCode,Continent,CountryPop
ABW,North_America,103000
AFG,Asia,22720000
AGO,Africa,12878000
AIA,North_America,8000
ALB,Europe,3401200
AND,Europe,78000
ANT,North_America,217000
...
```

The `SELECT` command lets you pick particular columns from the loaded dataset and display only those. The list of columns is provided as a single comma-separated list. The columns should be output in the same order provided by the user.

TAKE

```
> FROM city.csv TAKE 2
CityID,CityName,CountryCode,CityPop
1,Kabul,AFG,1780000
2,Qandahar,AFG,237500
```

```
> FROM city.csv TAKE 5
CityID,CityName,CountryCode,CityPop
1,Kabul,AFG,1780000
2,Qandahar,AFG,237500
3,Herat,AFG,186800
4,Mazar-e-Sharif,AFG,127800
5,Amsterdam,NLD,731200
```

```
> FROM city.csv TAKE 10
CityID,CityName,CountryCode,CityPop
1,Kabul,AFG,1780000
2,Qandahar,AFG,237500
3,Herat,AFG,186800
4,Mazar-e-Sharif,AFG,127800
5,Amsterdam,NLD,731200
6,Rotterdam,NLD,593321
7,Haag,NLD,440900
8,Utrecht,NLD,234323
9,Eindhoven,NLD,201843
10,Tilburg,NLD,193238
```

The `TAKE` command lets you limit the amount of output to display

ORDERBY

```
> FROM city.csv ORDERBY CityPop TAKE 10
CityID,CityName,CountryCode,CityPop
1024,Mumbai_(Bombay),IND,10500000
2331,Seoul,KOR,9981619
206,Sâ€žo_Paulo,BRA,9968485
1890,Shanghai,CHN,9696300
939,Jakarta,IDN,9604900
2822,Karachi,PAK,9269265
3357,Istanbul,TUR,8787958
2515,Ciudad_de_MÃˆxico,MEX,8591309
3580,Moscow,RUS,8389200
3793,New_York,USA,8008278
```

The `ORDERBY` command lets you sort the dataset by a single numeric column (and *only* numeric columns) in descending order.

JOIN

```
> FROM city.csv JOIN country.csv CountryCode
CityID,CityName,CountryCode,CityPop,CountryName,Continent,CountryPop,Capital
1,Kabul,AFG,1780000,Afghanistan,Asia,22720000,1
2,Qandahar,AFG,237500,Afghanistan,Asia,22720000,1
3,Herat,AFG,186800,Afghanistan,Asia,22720000,1
4,Mazar-e-Sharif,AFG,127800,Afghanistan,Asia,22720000,1
5,Amsterdam,NLD,731200,Netherlands,Europe,15864000,5
...
```

The `JOIN` command allows the user to combine two `.csv` datasets based on a common column. Multiple `JOIN` calls are also allowed:

```
> FROM city.csv JOIN country.csv CountryCode JOIN language.csv CountryCode
CityID,CityName,CountryCode,CityPop,CountryName,Continent,CountryPop,Capital,La
nguage
1,Kabul,AFG,1780000,Afghanistan,Asia,22720000,1,Balochi
2,Qandahar,AFG,237500,Afghanistan,Asia,22720000,1,Balochi
3,Herat,AFG,186800,Afghanistan,Asia,22720000,1,Balochi
4,Mazar-e-Sharif,AFG,127800,Afghanistan,Asia,22720000,1,Balochi
5,Amsterdam,NLD,731200,Netherlands,Europe,15864000,5,Arabic
...
```

You may assume that the joined column will always have the same name across both datasets, that there will be exactly one column in each of the datasets with that name, and that the two datasets will not have any other columns that share a name.

Column ordering should be preserved as if the first and second table had their columns concatenated, with the matching column name removed from the second column. This is the ordering shown in the examples above.

COUNTBY

```
> FROM language.csv COUNTBY Language ORDERBY count TAKE 7
Language,count
English,60
```

```
Arabic,33
Spanish,28
French,25
German,19
Chinese,19
Russian,17
```

The `COUNTBY` command takes a single column name, and reduces the dataset to only two columns: the specified column after `COUNTBY`, and a `count` column with the number of times that value of the specified column appears in the dataset

Preparing your code for review

Now that your query engine is complete, we would like it to be tested to ensure it keeps working in future. Please write automated tests to validate the behavior of your program, both in common cases and edge cases.

Additionally, ensure your code is readable and well-documented, as if you were submitting it as part of an actual code review.

[Bonus] Sort-merge Join

After having completed the other sections, ensuring your code is readable to a production quality, and has comprehensive test coverage, you may optimize the JOIN algorithm beyond a simple nested-loops join. For example, perhaps you would implement a sort-merge join.

It is not required to do this section to pass the interview - it's just for extra credit!