CSCI 2141 - Course Assignment - Part 3

Maxwell Hayden
B00865110
Due: Dec 8, 2023

3A

Section 1: dataset introduction

The dataset I decided to construct looks at two major datasets off Kaggle: firstly, a dataset including every known programming language, and secondly a dataset including all top repos on git hub. The dataset I created combines those two elements, along with a table about country information and author information to derive statistics about what programming languages are used the most for GitHub repos, and to further explore that question.

The dataset I designed is comprised of 4 main tables and one linker table. The data for these tables were both comprised by a chat GPT prompt and as well from Kaggle. Below are all the references and links to the table sources:

T1: language
Source link: https://www.kaggle.com/datasets/sujaykapadnis/programming-languages
Licencing: Public Domain
Columns used: Language name, Language type, Language year.

T2: author
Source Link: https://www.kaggle.com/datasets/sujaykapadnis/programming-languages
Licencing: Public Domain
Columns Used: Author Name
Columns Generated: Birth Country
GPT Prompt: for each of the names in the list of software engineers I am about to give you, associate a country to each one. If no country is found, give a country of NULL

T3: git_repo
Source Link: https://www.kaggle.com/datasets/parulpandey/most-starred-github-repositories
Licencing: CC0 Public Domain
Columns Used: ID, Repo Name, stars, language Name, Last Commit

T4: country
Source Link: https://www.kaggle.com/datasets/nelgiriyewithana/countries-of-the-world-2023
Licencing: Attribution 4.0 International (CC BY 4.0)
Columns Used: Country, GDP, Minimum wage, Population

Script example:

```
import java.io.*;
import java.util.*;

public class App3 {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        String output[] = new String[5000];
        output[0] = in.nextLine();
        String xx[][] = new String[5000][50];

        int i = 1;
        String input = in.nextLine();

        while(!input.equals("exit")){
            output[i] = input;
            i++;
            input = in.nextLine();
        }

        for(int j = 0; j < i; j++){

            for(int k = 0; k < output[j].length(); k++){
                if (output[j].charAt(k) == '"'){
                    for(int s = k + 1; s < output[j].length(); s++){
                        if (output[j].charAt(s) == ','){
                            output[j] = output[j].substring(0, s) + output[j].substring(s + 1,
output[j].length());
                        } else if (output[j].charAt(s) == '"'){
                            k = s + 1;
                            break;
                        }
                    }
                }
            }

            xx[j] = output[j].split(",");
        }

        String zz[][] = new String[5000][2];
        int jj = 0;
        for(int j = 0; xx[j][0] != null; j++){
             if (xx[j][1].length() == 2){
                continue;
             }

            if (xx[j].length > 2){
                for(int a = 1; a<xx[j].length; a++){
                    zz[jj][0] = xx[j][0];
                    zz[jj][1] = xx[j][a];
                    jj++;
                }
            } else {
                zz[jj] = xx[j];
                jj++;
            }
        }

        printArrayy(zz);
    }

    public static void printArrayy(String[][] array) {
        try (FileWriter out = new FileWriter("lang_name.txt")) {
            for (int i = 0; array[i][0] != null; i++) {
                out.write("(");
                for (int j = 0; j < array[i].length - 1; j++) {
                    out.write(array[i][j] + ", ");
                }
                out.write(array[i][array[i].length - 1]);
                out.write("),\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In devising the aforementioned script, my primary aim was to seamlessly translate CSV file data into a SQL-compatible format. This involved parsing SQL code into an array, from which I selectively chose columns, refining the dataset to my specific requirements. Beyond mere translation, the script proved instrumental in reconciling data inconsistencies across Kaggle datasets through adept use of SQL SELECT statements and outer joins. Operating within a macOS environment, the script served as a workaround, addressing the lack of straightforward solutions for CSV file integration into the database. Furthermore, its utility extended to selectively importing only the essential columns, streamlining data and enhancing efficiency amidst the surplus information from Kaggle. In essence, the script emerged as a versatile tool, not only facilitating data translation but also adeptly handling the intricacies and challenges between the Kaggle datasets.

The major plan for the data set is to be able to combine the two main data sets, git hub repos and languages, together with joins to be able to draw conclusions based off the information from the new table. Some simple questions that will be answered includes:

- Where do most programming languages come from?
- What languages produce the most stars on GitHub?
- How does language year compare to last commit within git hub?
- What authors come from low-income countries vs high income countries?
- Does the income of a country or its population determine the number of languages from it?

The global country data set has been used for many projects including this research where the scientist dove in and looked at using the data to understand countries life expectancies, populations, and CO2 emissions and any co relations between such information.

Link: https://www.linkedin.com/pulse/analysis-global-country-information-dataset-2023-alexander-jimenez

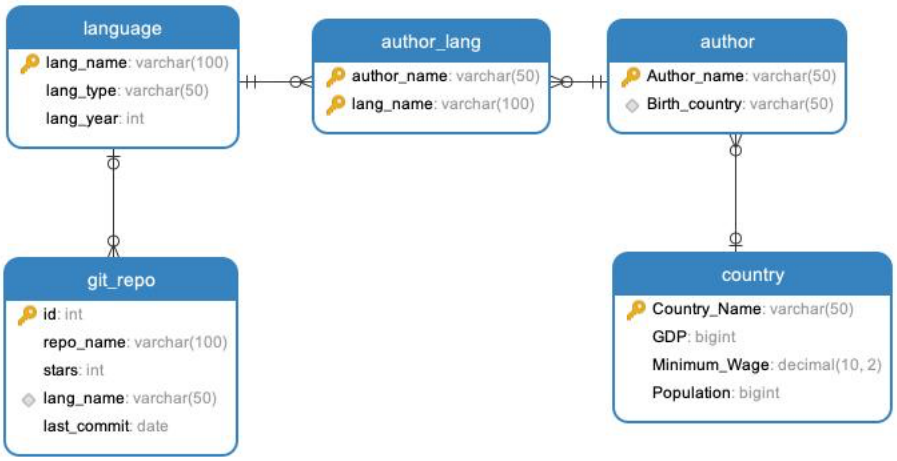
The programming language database is derived from a larger open-source git hub repo named Programming Language Database, or PLDB for short. PLDB has stirred up many conversations and a large community of people constantly updating and adding new developed languages to the database. The collective community works to keep this database up to date and active so that people have constant access to the information about the development, rank, and metadata about such new languages. Here is a long form post showing some active community discussion about the database!

Link: https://news.ycombinator.com/item?id=32619671

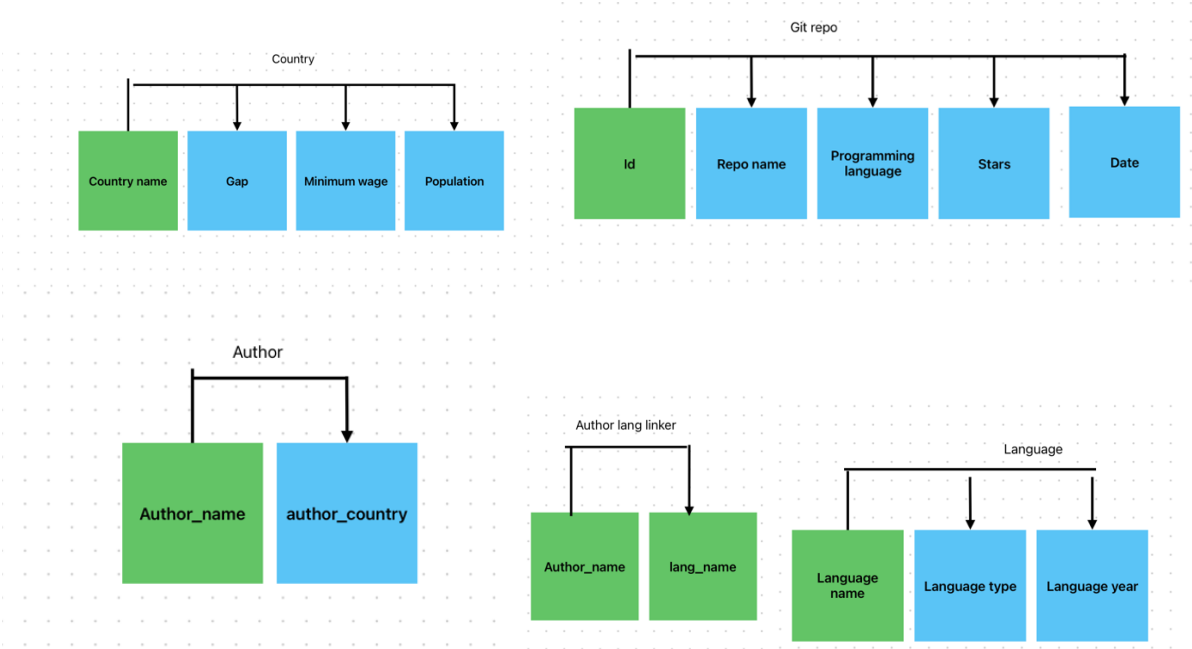
Section 2

	columns	Primary key	Foreign key	Col count	Row count
T1: language	**lang_name**: name of language **lang_type**: type of programming language **lang_year**: year of development	lang_name	N/A	3	4303
T2: author	**Author_name**: name of author **Birth_country**: birthplace of author, null if none known.	Author_name	Birth_country Relates to the country name in the country table	1210	2
T3: git_repo	**id**: auto incremented # to uniquely identify each git repo **repo_name**: name of git repository **Stars**: total # of stars on the repo **lang_name**: name of language used for the repo. **last_commit**: last time someone committed to the repo	id	lang_name Relates to lang_name in the language table. every language for a git repo should be present in the language table.	2800	5
T4: country	**Country_name** **GDP**: market value of goods and services provided by country **Minimum_wage** **Population**	Country_ name	N/A	195	4
T5: Author_lang	**Author_name**: name of author of a language **Lang_name:** name of language by the author	Author_name, Lang_name	Author_name - Connects author to author table Lang_name - Connects language to language table	1386	2

Section 3



Key = primary key
Diamond = foreign key



Every table in this set exhibits third normal form characteristics, devoid of both partial and transitive dependencies. The attributes within each table align comprehensively with their respective primary keys, minimizing complexity. The depicted arrows visually represent the relationships between these attributes, providing a clear illustration of their interconnectedness. Green represents a primary key.