# Project Overview

Learning Goals

* Be able to import data from CSV files and JSON into a relational database
* Be able to write a range of SQL queries to extract data from a relational database
* Gain experience writing interactive command line programs that support a range of commands and options

You will write a program that creates a database to store information about gourmet chocolate bars. This data was originally retrieved from Kaggle (<https://www.kaggle.com/rtatman/chocolate-bar-ratings/data>), but you will be working with a cleaned-up version of the data. You will also be working with JSON data that was retrieved from <https://restcountries.eu/>. Both data files are provided to you in the starter files directory.

After loading the data into the database, you will add the ability for a user to issue several different types of queries to extract information from the database.

To get started, get the data sources and the starter files from [this Dropbox folder](https://www.dropbox.com/sh/nqlt0lrq3qvi1bn/AAAYzztncljFrOWQHfHaXwtua?dl=0).

When you are done, turn in your version of the proj3\_choc.py file to Canvas.

## Part 1: Populating the Database (60 points)

For part 1 you need to create a new database and add data to it from the files flavors\_of\_cacao\_cleaned.csv and countries.json that are included in the starter repo. To work with the unit tests we have provided, you will need to use the following table and column names.

|  |  |  |
| --- | --- | --- |
| Table: Bars |  |  |
| Id (primary key) | Integer | Primary key, assigned by DB |
| Company | Text | Name of the company who makes the bar |
| SpecificBeanBarName | Text | The name of the bar itself, or sometimes the name of the bean |
| REF | Text | Dunno what this is. |
| ReviewDate | Text | Date review was done |
| CocoaPercent | Real | % of cocoa in the bar |
| CompanyLocationId | Integer | Foreign key - points to Countries |
| Rating | Real | Rating given by chocolate experts |
| BeanType | Text | Category of the cocoa bean |
| BroadBeanOriginId | Integer | Foreign key - points to Countries |

|  |  |  |
| --- | --- | --- |
| Table: Countries |  |  |
| Id (primary key) | Integer | Primary key, assigned by DB |
| Alpha2 | Text | 2 letter country code |
| Alpha3 | Text | 3 letter country code |
| EnglishName | Text | English name for country |
| Region | Text | Broad region where country is located. |
| Subregion | Text | More specific subregion where country is located. |
| Population | Integer | Country’s population |
| Area | Real | Country’s area in km2 |

Note that the Bars table references the Countries table twice--with two Foreign Keys. You will need to make sure that all of the relations are correctly inserted into your database.

**Grading (all points include passing relevant tests):**

* **[20 points] Read all data from CSV into Bars table**
* **[20 points] Read all data from JSON into Countries table**
* **[20 points] Insert correct keys to model relationships**

## Part 2: Implement Query Interface (100 points)

To prepare for supporting interactive queries, in part 2 you will implement a function “process\_command” that takes a command string and returns a list of tuples representing records that match the query.

Your process\_command function must be able to support four main commands, along with a variety of parameters for each. The four commands are ‘bars’, ‘companies’, ‘countries’, and ‘regions.’ Each command supports parameters and provides results as detailed below.

* bars
  + Description: Lists chocolate bars, according the specified parameters.
  + Parameters:
    - sellcountry=<alpha2> | sourcecountry=<alpha2> | sellregion=<name> | sourceregion=<name> [default: all results, don’t limit by country]
      * Description: Specifies a country or region within which to limit the results, and also specifies whether to limit by the seller (or manufacturer) or by the bean origin source.
    - ratings | cocoa [default: ratings]
      * Description: Specifies whether to sort by rating or cocoa percentage
    - top=<limit> | bottom=<limit> [default: top=10]
      * Description: Specifies whether to list the top <limit> matches or the bottom <limit> matches.
* companies
  + Description: Lists chocolate bars sellers according to the specified parameters. Only companies that **sell more than 4** kinds of bars are listed in results.
  + Parameters:
    - country=<alpha2> | region=<name> [default: none]
      * Description: Specifies a country or region within which to limit the results.
    - ratings | cocoa | bars\_sold [default: ratings]
      * Description: Specifies whether to sort by rating, cocoa percentage, or the number of different types of bars sold
    - top=<limit> | bottom=<limit> [default: top=10]
      * Description: Specifies whether to list the top <limit> matches or the bottom <limit> matches.
* countries
  + Description: Lists countries according to specified parameters. Only countries that sell/source **more than 4** kinds of bars are listed in results.
  + Parameters:
    - region=<name> [default: none]
      * Description: Specifies a region within which to limit the results.
    - sellers | sources [default: sellers]
      * Description: Specifies whether to select countries based sellers or bean sources.
    - ratings | cocoa | bars\_sold [default: ratings]
      * Description: Specifies whether to sort by rating, cocoa percentage, or the number of different types of bars sold
    - top=<limit> | bottom=<limit> [default: top=10]
      * Description: Specifies whether to list the top <limit> matches or the bottom <limit> matches.
* regions
  + Description: Lists regions according to specified parameters. Only regions that sell/source **more than 4** kinds of bars are listed in results.
  + Parameters:
    - sellers | sources [default: sellers]
      * Description: Specifies whether to select countries based sellers or bean sources.
    - ratings | cocoa | bars\_sold [default: ratings]
      * Description: Specifies whether to sort by rating, cocoa percentage, or the number of different types of bars sold
    - top=<limit> | bottom=<limit> [default: top=10]
      * Description: Specifies whether to list the top <limit> matches or the bottom <limit> matches.

### Return Values

The return value for process\_command( ) varies depending on the command issued. Matching the return values as specified is essential for passing the unit tests.

The mapping of commands to outputs is as follows:

|  |  |  |
| --- | --- | --- |
| **Command** | **Return Value** | **Source Columns for tuple values** |
| bars | 6-tuple | 'SpecificBeanBarName','Company', 'CompanyLocation', 'Rating', 'CocoaPercent', 'BroadBeanOrigin' |
| companies | 3-tuple | 'Company', 'CompanyLocation', <agg>  *Where "agg" is the requested aggregation (i.e., average rating or cocoa percent, or number of bars sold)* |
| countries | 3-tuple | 'Country', 'Region', <agg>  *Where "agg" is the requested aggregation (i.e., average rating or cocoa percent, or number of bars sold)* |
| regions | 2-tuple | 'Region', <agg>  *Where "agg" is the requested aggregation (i.e., average rating or cocoa percent, or number of bars sold)* |

### **Grading**

* **[100 points] Implement commands with all parameters**
  + **~ 25 points each command, includes passing tests**

## Part 3: Interactive Capabilities [40 points]

Implement a command line interface to allow a user to specify queries using the language and syntax described in Part 2. The only things you’ll need to add in this part are

* prompting the user for input
* formatting the output “nicely”
* adding basic error handling (i.e., not crashing the program on invalid inputs)

Grading:

* [15 points] Graceful error handling
  + (This is only for handling **invalid** inputs. If your program crashes on valid inputs, you will get a 0 and need to submit again for regrade, which will be capped at 60% of the max grade. If you choose to implement different command, for example, “bars rating” instead of “bars ratings”, please update the instruction or it will be considered an error.)
* [20 points] Presentable output
* [10 points] Fixed width formatting matching (more or less) exactly the sample output below

Here is an example run:

spinoza$ python3 proj3\_choc\_solution.py

Enter a command: bars ratings

Chuao Amedei Italy 5.0 70% Venezuela (B...

Toscano Blac... Amedei Italy 5.0 70% Unknown

Pablino A. Morin France 4.0 70% Peru

Chuao A. Morin France 4.0 70% Venezuela (B...

Chanchamayo ... A. Morin France 4.0 63% Peru

Morobe Amano United State... 4.0 70% Papua New Gu...

Guayas Amano United State... 4.0 70% Ecuador

Porcelana Amedei Italy 4.0 70% Venezuela (B...

Nine Amedei Italy 4.0 75% Unknown

Madagascar Amedei Italy 4.0 70% Madagascar

Enter a command: bars sellcountry=US cocoa bottom=5

Peru, Madaga... Ethel's Arti... United State... 2.5 55% Unknown

Trinidad Ethel's Arti... United State... 2.5 55% Trinidad and...

O'ahu, N. Sh... Guittard United State... 3.0 55% United State...

O'ahu, N. Sh... Malie Kai (G... United State... 3.5 55% United State...

O'ahu, N. Sh... Malie Kai (G... United State... 2.8 55% United State...

Enter a command: companies region=Europe bars\_sold

Bonnat France 27

Pralus France 25

A. Morin France 23

Domori Italy 22

Valrhona France 21

Hotel Chocol... United Kingd... 19

Coppeneur Germany 18

Zotter Austria 17

Artisan du C... United Kingd... 16

Szanto Tibor Hungary 15

Enter a command: companies ratings top=8

Amedei Italy 3.8

Patric United State... 3.8

Idilio (Felc... Switzerland 3.8

Benoit Nihan... Belgium 3.7

Cacao Sampak... Spain 3.7

Bar Au Choco... United State... 3.6

Soma Canada 3.6

Brasstown ak... United State... 3.6

Enter a command: countries bars\_sold

United State... Americas 764

France Europe 156

Canada Americas 125

United Kingd... Europe 107

Italy Europe 63

Ecuador Americas 55

Australia Oceania 49

Belgium Europe 40

Switzerland Europe 38

Germany Europe 35

Enter a command: countries region=Asia ratings

Viet Nam Asia 3.4

Israel Asia 3.2

Korea (Repub... Asia 3.2

Japan Asia 3.1

Enter a command: regions bars\_sold

Americas 1085

Europe 568

Oceania 70

Asia 46

Africa 26

Enter a command: regions ratings

Oceania 3.3

Asia 3.2

Europe 3.2

Americas 3.2

Africa 3.0

Enter a command: bad command

Command not recognized: bad command

Enter a command:

Enter a command: bars nothing

Command not recognized: bars nothing

Enter a command: exit

bye