Table of Contents

[4. Functionality 2](#_Toc47451761)

[4.1 MySQL 2](#_Toc47451762)

[4.1.1 Alan’s travel details 7](#_Toc47451763)

[4.1.2 European countries with lower than average life expectancy 7](#_Toc47451764)

[4.1.3 Peoples stage of life 7](#_Toc47451765)

[4.1.4 Capitals and Official Languages of North America 8](#_Toc47451766)

[4.1.5 Length of Stays 10](#_Toc47451767)

[4.2 MongoDB 11](#_Toc47451768)

[4.2.1 Average Age of Students 12](#_Toc47451769)

[4.2.2 Honours Level 12](#_Toc47451770)

[4.2.3 Qualified Students 13](#_Toc47451771)

[4.2.4 Student and their Qualifications 13](#_Toc47451772)

[4.3 Normalisation 13](#_Toc47451773)

[4.4 Python 15](#_Toc47451774)

[4.4.1 Python program 15](#_Toc47451775)

[5. References 24](#_Toc47451776)

# 4. Functionality

## 4.1 MySQL

Import the **world** database from **world.sql** to MySQL

* **MySQL Workbench**

To import the *world* database from *world.sql* to MySQL I am using MySQL Workbench.

Open MySQL Workbench => Server => Data Import => Import from self-Contained File =>

Click and find the location where it is on my computer => start import => open new SQL query tab and type “SHOW DATABASES;” (When I run it I can see what databases are in world.sql).

* **Command prompt (cmder)**

Type commands in cmder prompt:

1. cd “\Program Files”
2. cd MySQL
3. cd MySQL Server 8.0
4. cd bin



1. Then type commad as follows:

mysql -u (here is my root name) -p(here is my password to my root) < "C:\Users\karolina\Desktop\AD\_project\_2020\world.sql" (enter)

I give the whole path to the file here

1. I can go back to MySQL Workbench and I can run “SHOW DATABASES;” and world database is there.



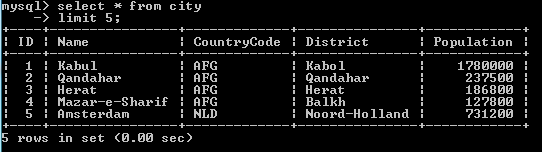
The ***world*** database is imported.

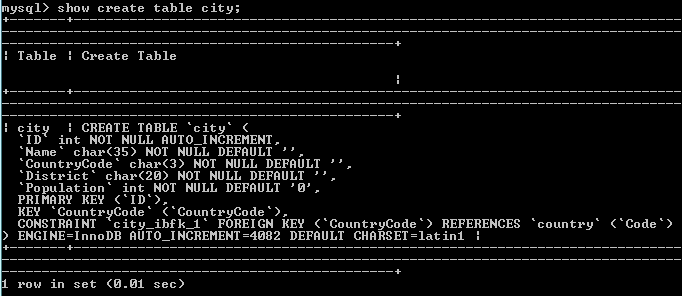
I can see what tables are in the ***world*** database by typing:

1. use world; (ctrl + enter)
2. show tables; (ctrl + enter)

The same commands I can type in MySQL 8.0 Command Line Client.

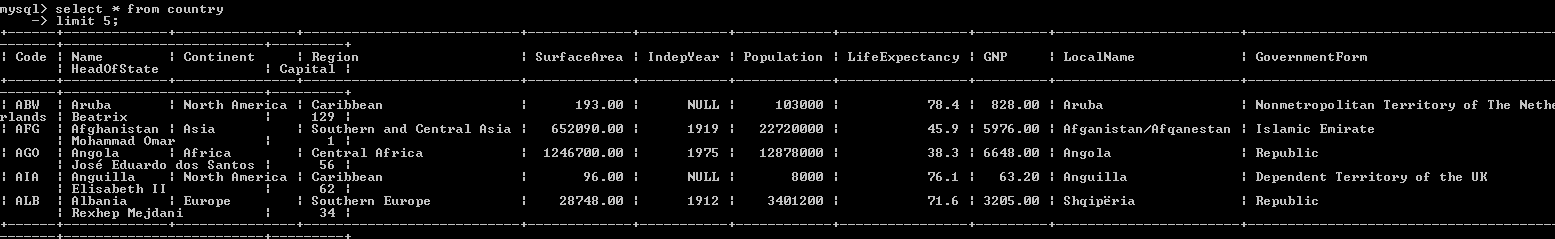


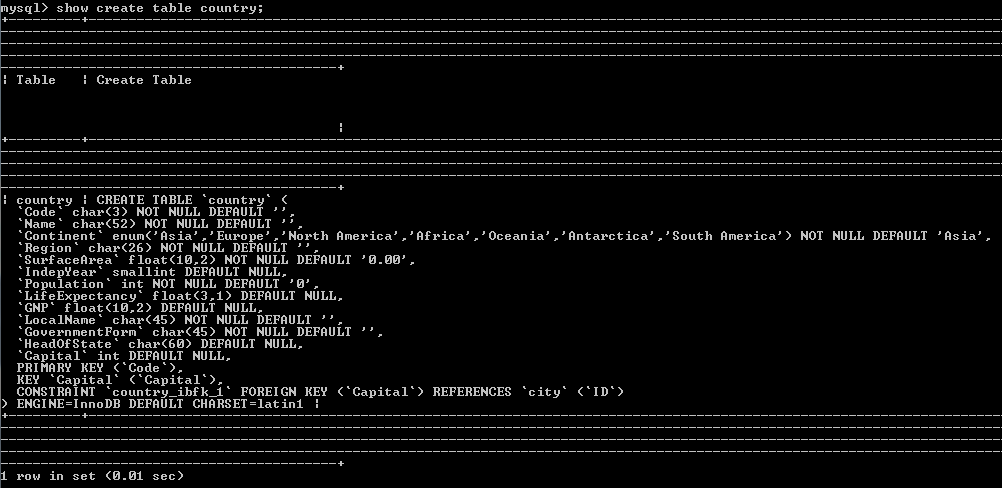




The primary key of the city table is the “ID” field.

There is a foreign key constraint in the city table, which is called “CountryCode “. This foreign key references the “Code” in the country table.





The primary key in the country table is “Code” field.

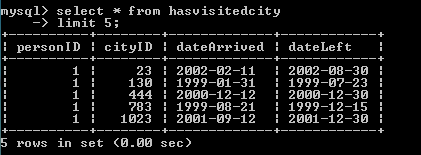
The country table has a foreign key constraint where the “Capital” field of the country table references the “ID” field in the city table.

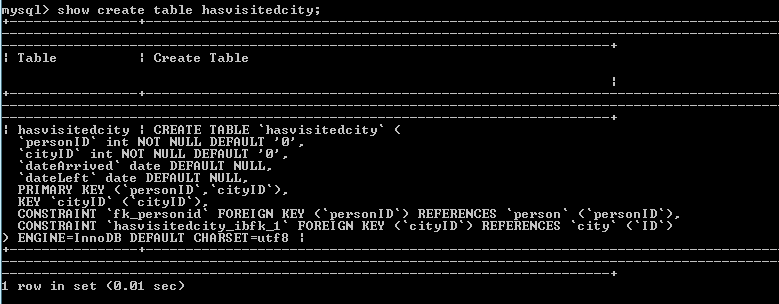




The primary key in the countrylanguage table is the “CountryCode” and the “Language” fields.

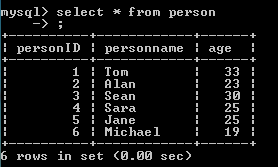
The countrylanguage table has a foreign key constraint where the “CountryCode” field references the “Code” field in the country table.

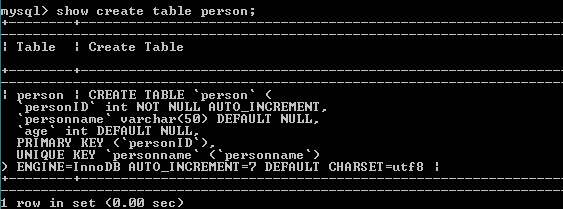




The primary key on the hasvisitiedcity table is the” personID” and “cityID” fields.

There are two foreign key constraints on the hasvisitedcity table. First “personID” field in the hasvisitedcity table references “personID” in the person table, and second “cityID in the hasvisitedcity table references the “ID” field in the city table.





The primary key of the person table is “personID”.

There is no foreign key in the person table, but there is a foreign key pointing in to it from the hasvisitedcity table.

### 4.1.1 Alan’s travel details

Give the MySQL command that shows;

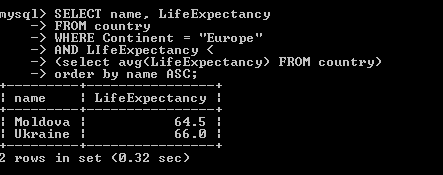
* the name of the cities,
* the arrival date in the cities,
* the name of the country the city is in.

For all cities and countries visited by Alan in alphabetical order by city name.



### 4.1.2 European countries with lower than average life expectancy

Give the MySQL command to show the country name the country's life expectancy for all countries in Europe whose life expectancy is lower than the average in alphabetical order by country name.

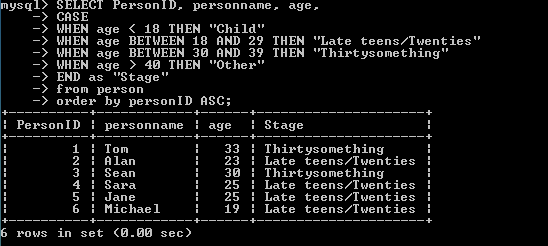


### 4.1.3 Peoples stage of life

Give the SQL command to show the following in ascending personID order:

* The person’s ID
* The person’s name
* The Person’s age
* A column called Stage that shows the following:

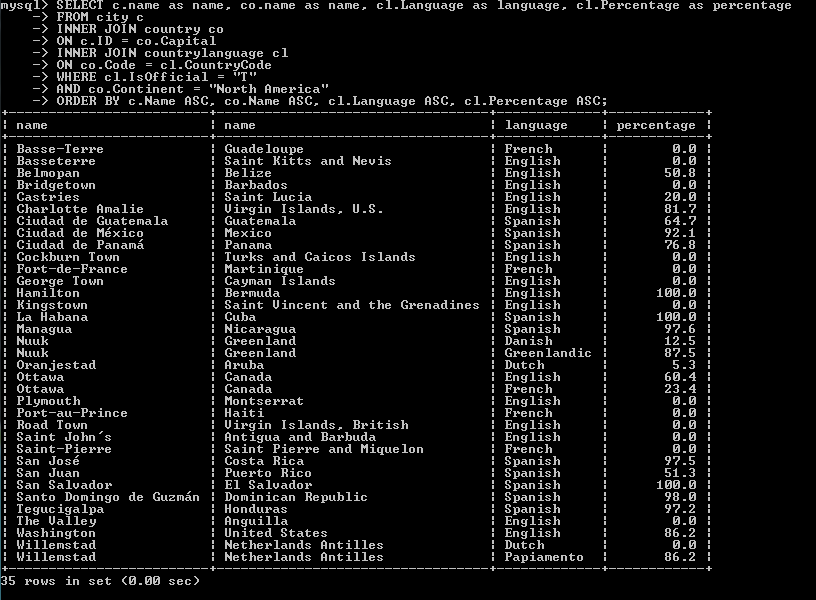
|  |  |
| --- | --- |
| **Person’s ID** | **Stage column output** |
| Under 18 | Child |
| Between 18 and 29 | Late teens/Twenties |
| Between 30 and 39 | Thirty something |
| 40 or older | Other |



### 4.1.4 Capitals and Official Languages of North America

Give the SQL command to show for each country in North America:

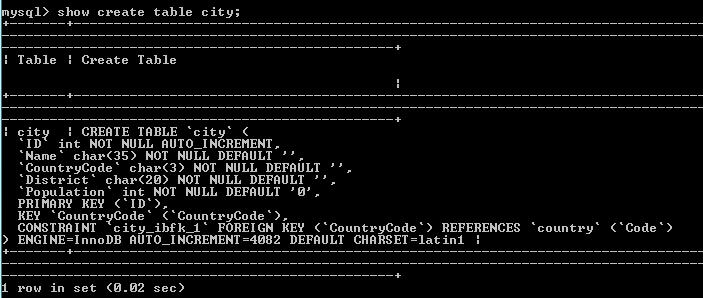
* The name of the capital city
* The name of the country
* The official language(s)
* The percentage of people who speak the official language(s) The results should be alphabetical city name order, and within that by country name order, and within that by language order, and within that by ascending percentage.



I had to work with 3 tables here to get all needed information.

1. the**city** table,
2. the **country** table
3. and the **countrylanguage** table.

* c.name is in the city table,
* co.name and co.Continent are in the country table,
* cl.language, cl.IsOfficial and cl.Percentage fields are in the countryLanguage table.







The country table has a foreign key called “Capital” field which references the “ID” field in the city table.

The city table has a foreign key called “CountryCode” which references the “Code” field in the country table.

The countrylanguage table has a foreign key constraint where the “CountryCode” field references the “Code” field in the country table.

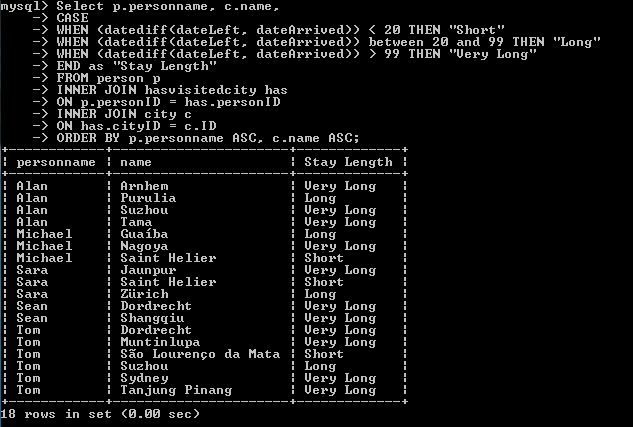
### 4.1.5 Length of Stays

Give the SQL command to show for each country person:

* The person’s name
* The name of the city the person visited
* A column called Stay Length that shows the following:

|  |  |
| --- | --- |
| Time the person stayed in city | Stay Length column output |
| Less than 20 days | Short |
| Between 20 and 99 days | Long |
| Over 99 days | Very long |

The results should be sorted alphabetically by personname, and within that by city name.



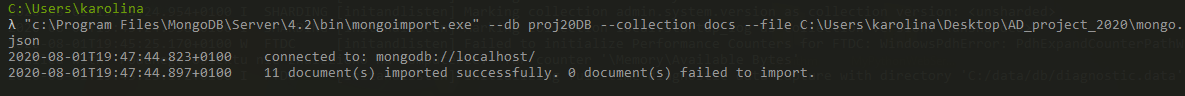
I used CASE statement which went through conditions and returned values (when the first condition was met). The DATEDIFF () function returns the difference between two date values.

## 4.2 MongoDB

Import the file mongo.json to a collection called docs In a database called proj20DB.

I opened cmder and I typed:

“c:\Program Files\MongoDB\Server\4.2\bin\mongoimport.exe” --db proj20DB – collection docs –file (drag and drop the whole path of the file to it) =>ENTER



11 documents were imported successfully.

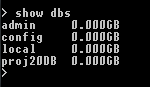
Then I had to type a command (showed below) to run mongod.



The next thing , I opened another command prompt (this time it was not cmder) to connect mongo there.



I typed show dbs to see if my database which I imported is there.





I typed db.docs.find() to see what is In the collection.



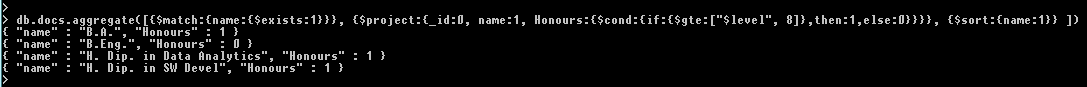
### 4.2.1 Average Age of Students

Give the MongoDB command to find the average age of students



### 4.2.2 Honours Level

Give the MongoDB command to show the name of each course and Honours which has the value true if the course level is 8 or higher, otherwise false. The output should be sorted by name.



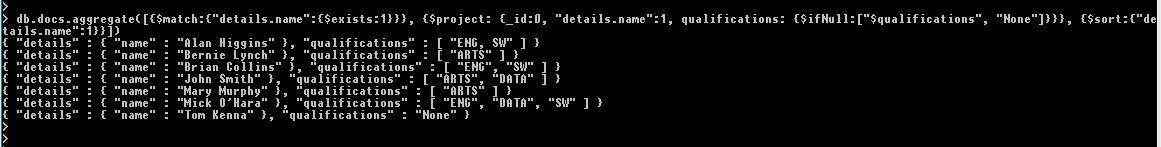
### 4.2.3 Qualified Students

Give the MongoDB command to show the number of Qualified Students i.e. those documents with a qualifications field.



### 4.2.4 Student and their Qualifications

Give the MongoDB command to show the name of each Student and his/her qualifications. The output should be in alphabetical name order. If the student has no qualifications the word “None” should appear.



## 4.3 Normalisation

Normalisation is the process of organizing the columns (attributes) and tables (relations) of a relational database to minimize data redundancy.

Being able to do data analysis more easily is reason enough for an organization to engage in data normalization. There are, however, many more reasons to perform this process, all of them highly beneficial. One of the most notable is the fact that data normalization means databases take up less space[1].

A primary key is a key in a relational database that is unique for each record. It is a unique identifier, such as a driver license number, telephone number or employee ID. It cannot contain null values. A relational database must always have one and only one primary key. Primary keys typically appear as columns in relational database tables.

A foreign key is a field or collection of fields in one table that uniquely identifies a row of another table. It is a column or group of columns in relational database table that provides a link between data in two tables. If there is more than one table, they are linked by foreign keys. The foreign key constraint is used to prevent actions that would destroy links between tables and prevent invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to[2][3].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EmployeeID\* | EmployeeName | Salary | DeptName | DeptLocation | DeptBudget |
| 100 | Sean | 35,000 | Sales | Dublin | 750,000 |
| 101 | Mary | 36,000 | Sales | Dublin | 750,000 |
| 102 | John | 40,000 | Sales | Dublin | 750,000 |
| 104 | Albert | 55,000 | R&D | Galway | 1,500,000 |
| 105 | Conor | 52,000 | R&D | Galway | 1,500,000 |
| 106 | Meave | 50,000 | R&D | Galway | 1,500,000 |
| 107 | Tom | 50,000 | R&D | Galway | 1,500,000 |
| 108 | Alice | 44,500 | HR | Limerick | 250,000 |

**Proposed Employee table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EmployeeID\* | EmployeeName | EmployeeSurname | BirthDate | Gender | HireDate | PhoneNumber | DeptNo |
| 100 | Sean | Moran | 20/05/1975 | M | 15/02/2019 | 52136987 | 100 |
| 101 | Mary | King | 15/02/1980 | F | 14/05/2018 | 58932145 | 100 |
| 102 | John | Long | 22/06/1990 | M | 22/05/2018 | 21587624 | 100 |
| 104 | Albert | Higgins | 30/11/1978 | M | 01/06/2017 | 15978963 | 200 |
| 105 | Conor | Arkins | 05/12/1988 | M | 09/08/2017 | 45698732 | 200 |
| 106 | Meave | O’Connor | 30/06/1981 | F | 25/08/2015 | 85469875 | 200 |
| 107 | Tom | Hanks | 07/01/1979 | M | 02/04/2016 | 12914569 | 200 |
| 108 | Alice | White | 19/07/1977 | F | 05/09/2017 | 45879365 | 300 |

Primary Key – EmpolyeeID. Each record needs to be unique and it is true in this database. This field should not be null and changed.

The “DeptNo” column in the Employee table is a FOREIGN KEY in the Employee table.

Thedata which is entered in the Employee table would be constrained by this foreign key reference. “DeptNo” that exist in the Department table may be referenced in the employee table.

**Proposed Department table**

|  |  |  |  |
| --- | --- | --- | --- |
| DeptName | DeptNo\* | DeptLocation | DeptBudget |
| Sales | 100 | Dublin | 750,000 |
| R&D | 200 | Galway | 1,500,000 |
| HR | 300 | Limerick | 250,000 |

The “DeptNo” column in the Depatrment table is the PRIMARY KEY in the Department table.

The Department table could be linked to Employee table to see what employees belong to what departments and Employees to Department to see what is the name, location and budget of the certain department.

**Proposed Salaries table**

|  |  |  |  |
| --- | --- | --- | --- |
| EmployeeID | Salary | From\_date | To\_date |
| 100 | 35,000 | 15/02/2019 | 02/08/2020 |
| 101 | 36,000 | 14/05/2018 | 02/08/2020 |
| 102 | 35,000 | 22/05/2018 | 15/09/2019 |
| 102 | 40,000 | 16/09/2019 | 08/08/2020 |
| 104 | 50,000 | 01/06/2017 | 31/05/2018 |
| 104 | 52,000 | 01/06/2018 | 31/05/2019 |
| 104 | 55,000 | 01/06/2019 | 02/08/2020 |
| 105 | 50,000 | 09/08/2017 | 31/08/2018 |
| 105 | 52,000 | 01/09/2018 | 02/08/2020 |
| 106 | 45,000 | 25/08/2015 | 31/08/2017 |
| 106 | 48,000 | 01/09/2017 | 31/08/2019 |
| 106 | 50,000 | 01/09/2019 | 02/08/2020 |
| 107 | 45,000 | 01/04/2016 | 31/03/2018 |
| 107 | 48,000 | 01/04/2018 | 31/03/2019 |
| 107 | 50,000 | 01/04/2019 | 02/08/2020 |
| 108 | 43,000 | 05/09/2017 | 30/09/2019 |
| 108 | 44,500 | 01/10/2019 | 02/08/2020 |

The Salary table was created to link Employee table and it would be done by “EmployeeID” as a foreign key. It could also be linked to Department table using “DeptNo” as a foreign key but I think salaries/wages are more attributed to the employees than departments. Salary table shows history of individual salaries of each employee and whoever analyzes the data from Salary table can easily get to dates of employments and increases of wages. Very useful data but should be confidential and not seen by everyone.

First Normal Form (1NF) rules in this database are met.

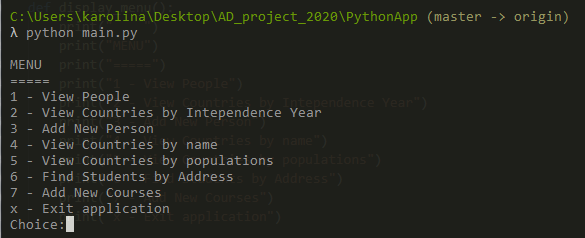
Second Normal Form (2NF) is also met as there is a primary key – EmployeeID.

3NF(Third Normal Form) rules not met as there are transitive functional dependencies. A transitive dependency exists when there is an intermediate functional dependency[4].

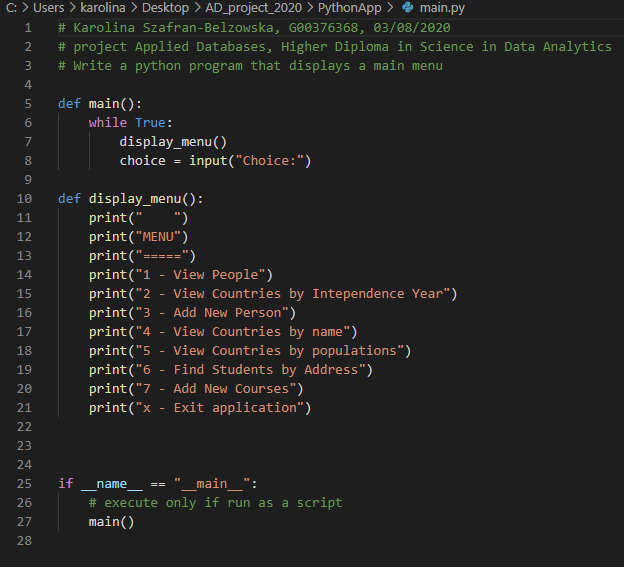
## 4.4 Python

### 4.4.1 Python program

Write a python program that displays a main menu as follows:

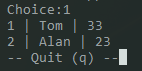


Below is shown a python program which I wrote in Visual Studio Code.

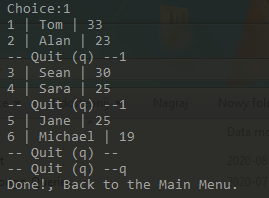


1. **View People**

The user is shown the list of People in the world database, in groups of 2:

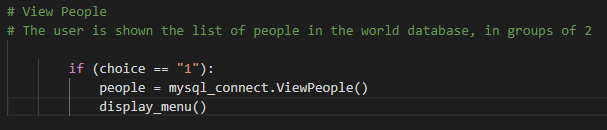


If the user presses any key except “q” the next 2 people in the database are shown:

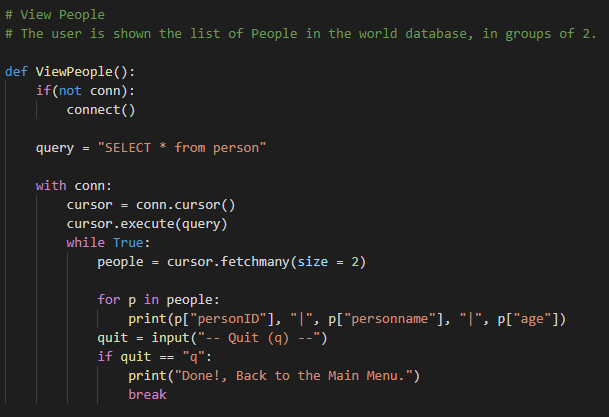


Whenever the user presses „q” he or she is brought back to the Main Menu.

main.py

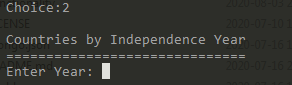


mysql\_connect.py



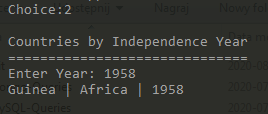
1. **View Countries by Independence Year**

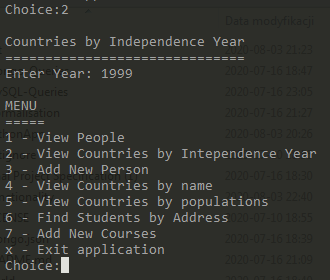
The user is asked to enter a year.



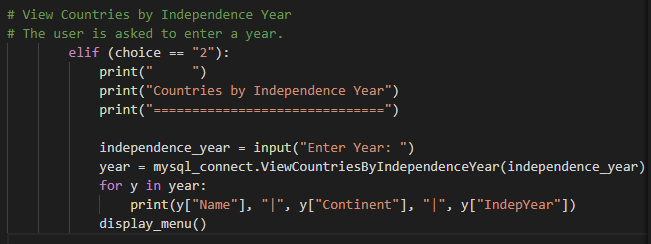
When a year is entered, the country’s Name, Continent and Independence Year is shown for each country whose Independence Year corresponds to what the user entered.

If no countries became independent in the year specified by the user nothing is shown and is brought back to the Main Menu.

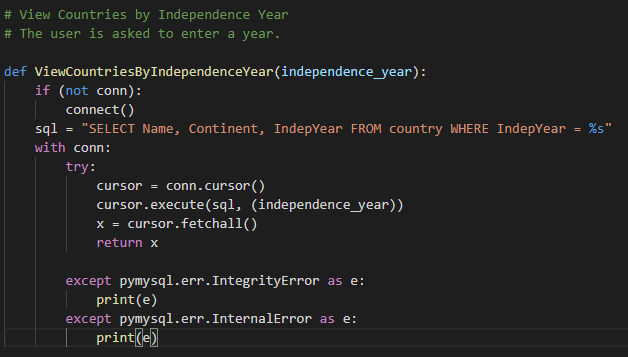




main.py

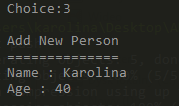


mysql\_connect.py

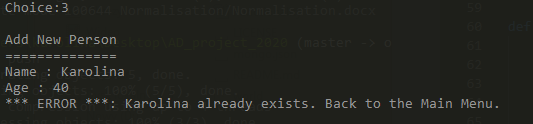


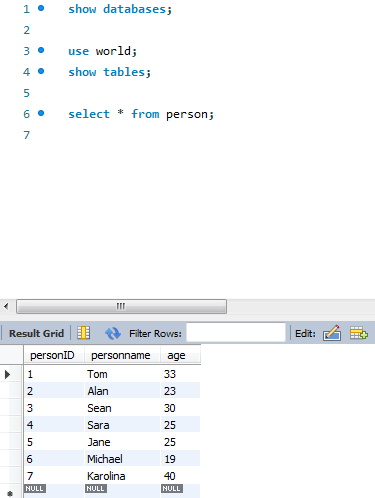
1. **Add New Person**

The user is asked to enter details of a new person as shown, the person is then added to the world database. (NOTE: The user should not be prompted to enter a personID).



If the user enters a name that already exists in the database, the person should not be added to the database, and an error message should be displayed.



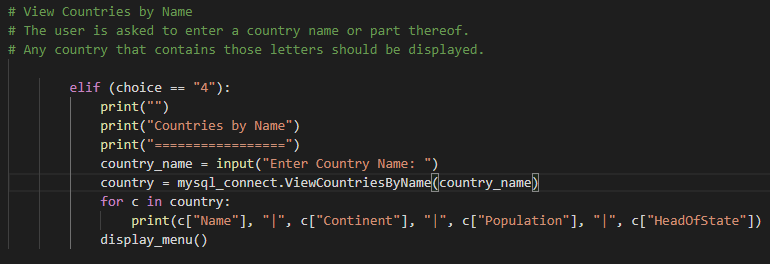


1. **View Countries by Name**

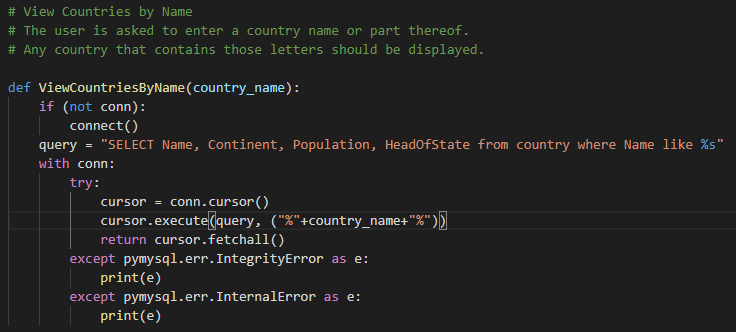
The user is asked to enter a country name or part thereof.

Any country that contains those letters should be displayed.

main.py



mysql\_connect.py



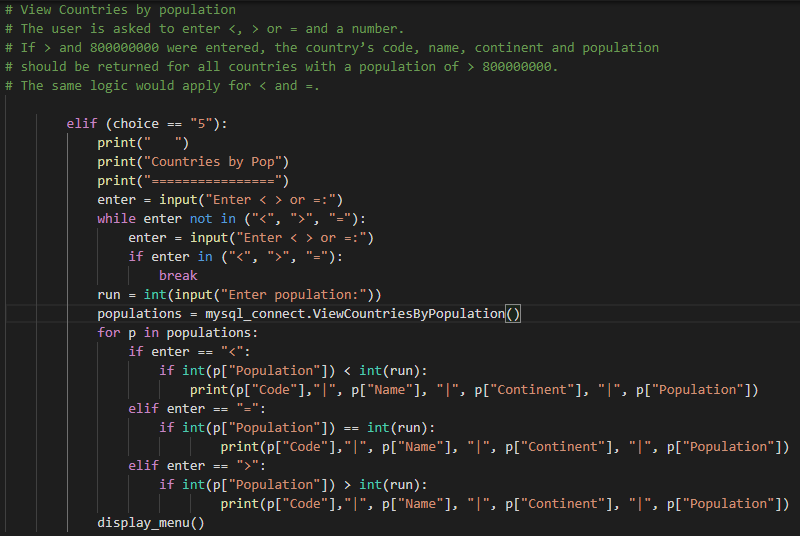


1. **View Countries by Population**

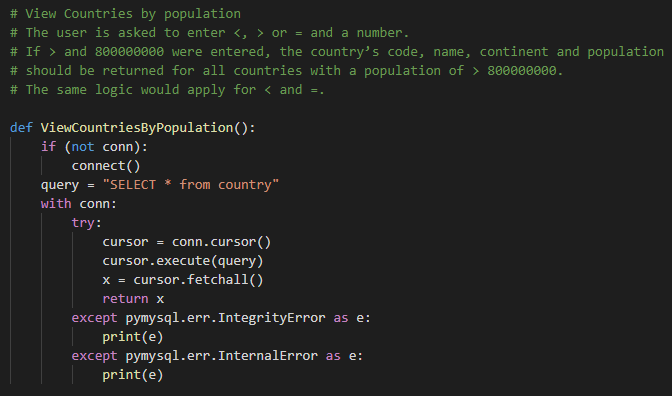
The user is asked to enter <, > or = and a number.

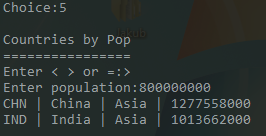
If > and 800000000 were entered, the country’s code, name, continent and population should be returned for all countries with a population of > 800000000. The same logic would apply for < and =.

main.py

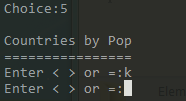


mysql\_connect.py





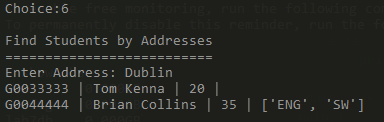
If the user does not enter < , > or = he/she is repeatedly asked until a valid choice is entered.



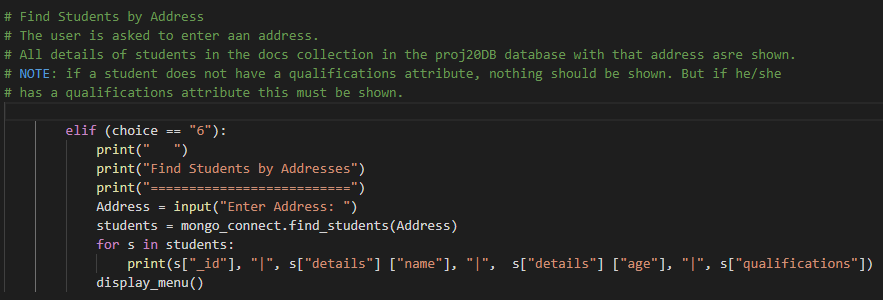
1. **Find Students by Address**

The user is asked to enter an address. All details of students in the “docs” collection in the “proj20DB”database with that address are shown.

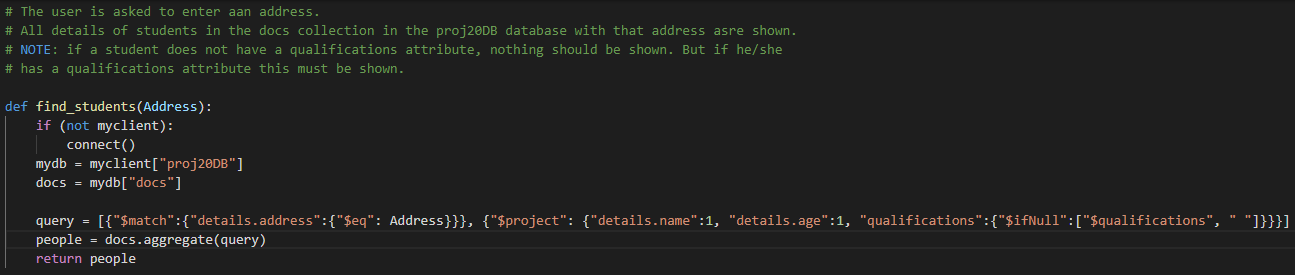
NOTE: If a student does not have a qualification attribute, nothing should be shown. But if he/she has a qualifications attribute this must be shown.



main.py



mongo\_connect.py



1. **Add New Course**

The user is asked to enter an \_id, Name and Level for a new course, which is then added to the “docs” collection in the “proj20DB” database.

# 5. References

[1] <https://www.import.io/post/what-is-data-normalization-and-why-is-it-important/> (02/08/2020)

[2] <https://www.geeksforgeeks.org/difference-between-primary-key-and-foreign-key/> (02/08/2020)

[3] <https://www.w3schools.com/sql/sql_foreignkey.asp> (02/08/2020)

[4] <https://en.wikipedia.org/wiki/Third_normal_form> (02/08/2020)

GitHub Repositories

The lecture slides presented in the course – Module: Applied Databases, Gerard Harrison – Higher Diploma in Data Analytics, 2020