Contents

[4 Functionality 1](#_Toc8996432)

[4.1 MySQL 1](#_Toc8996433)

[4.1.6 Country Independence 1](#_Toc8996434)

[4.2 Normalisation 1](#_Toc8996435)

[4.4 Python 2](#_Toc8996436)

[4.4.1 Python 2](#_Toc8996437)

Innovation.doc G00376315 Elizabeth Daly H. Dip. Data Analytics

# 4 Functionality

## 4.1 MySQL

### 4.1.6 Country Independence

I will claim innovation here for reducing the number of WHEN cases in the query through use of IF statements. Instead of two lines to catch combined IndepYear and Population conditions:

WHEN YEAR(CURDATE()) - IndepYear < 10 AND Population > 100000000 THEN CONCAT("New Large ", GovernmentForm)

WHEN YEAR(CURDATE()) - IndepYear < 10 AND Population <= 100000000 THEN CONCAT("New ", GovernmentForm)

I incorporate an IF statement in a single WHEN:

WHEN YEAR(CURDATE()) - IndepYear < 10 THEN IF( Population > 100000000, CONCAT("New Large ", GovernmentForm), CONCAT("New ", GovernmentForm))

I also reduce the amount of code required to catch the “Modern” and “Early” independences through use of BETWEEN, so, instead of:

WHEN YEAR(CURDATE()) - IndepYear > 9 AND YEAR(CURDATE()) - IndepYear < 50 THEN IF( Population > 100000000, CONCAT("Modern Large ", GovernmentForm), CONCAT("Modern ", GovernmentForm))

I use:

WHEN IndepYear BETWEEN 1970 AND 2018 THEN IF( Population > 100000000, CONCAT("Modern Large ", GovernmentForm), CONCAT("Modern ", GovernmentForm))

## 4.2 Normalisation

It would be better to separate personal information about students (studentID, studentName, dob) from the details about modules (moduleID and moduleName). A **students** table could have columns (studentID, studentName, dob, and modulesTaken) PK =composite as in example. A separate **modules** table could have columns (moduleID and moduleName) with PK = moduleID. The **students** table could have a foreign key (FK)= modulesTaken which references the moduleID column of **modules** table. The modulesTaken column can only contain values that exist in moduleID column of modules table. This would avoid the possibility of a student enrolling on a module that doesn’t exist.

The **modules** table could also contain a “dummy” row to be referenced when a student has enrolled in college before making module decisions or has dropped out. For example, it could have moduleID=0 and moduleName=No modules chosen. That would avoid the modulesTaken column (part of the PK) of **students** table containing a NULL value.

modules table

|  |  |
| --- | --- |
| moduleID\* | moduleName |
| 0 | No modules chosen |
| 100 | Applied Databases |
| 101 | Java Programming |

students table

|  |  |  |  |
| --- | --- | --- | --- |
| studentID\* | studentName | dob | modulesTaken\* (FK) |
| 1 | Sean | xxx | 100 |
| 2 | Bill | yyy | 100 |
| 3 | Tom | zzz | 101 |
| 3 | Tom | zzz | 104 |
| 4 | Mary | www | 101 |
| 4 | Mary | www | 102 |
| 7 | John | aaa | 0 |

* Queries to the tables could be handled by joins.

## 4.4 Python

### 4.4.1 Python

PythonProj.py is the main script which displays the menu options and calls functions depending on which choice is made. The specific code to interact with the databases is kept separate in two scripts: script1.py for the MySQL commands and script2.py for the mongo commands.

#### Question 2

The user is asked to enter an operator and a population. A try except block checks that the population entered is a number, another checks that it has value >= 0, and a third checks that the user-entered operator is valid. Running the code:

Choice: 2

Q2 Cities by Population

-----------------------

Enter < > or = : gt

Enter population : 700

\*\*\* ERROR \*\*\*: Invalid operator, please use < > or =

Choice: 2

Q2 Cities by Population

-----------------------

Enter < > or = : =

Enter population : three

\*\*\* ERROR \*\*\*: Population must be a number, please choose again.

Choice: 2

Q2 Cities by Population

-----------------------

Enter < > or = : =

Enter population : -700

\*\*\* ERROR \*\*\*: Population must be >= 0, please choose again

Choice: 2

Q2 Cities by Population

-----------------------

Enter < > or = : =

Enter population : 700

Cities with a population = 700 are:

1791 | Flying Fish Cove | CXR | – | 700

#### Question 3

When checking to see if a CountryCode exists in the country table, I use limit to speed up the query:

mysql> select exists(select \* from city where CountryCode = "IRL");

+------------------------------------------------------+

| exists(select \* from city where CountryCode = "IRL") |

+------------------------------------------------------+

| 1 |

+------------------------------------------------------+

1 row in set (0.05 sec)

mysql> select exists(select \* from city where CountryCode = "IRL" limit 1);

+--------------------------------------------------------------+

| exists(select \* from city where CountryCode = "IRL" limit 1) |

+--------------------------------------------------------------+

| 1 |

+--------------------------------------------------------------+

1 row in set (0.00 sec)

I use try except in script1.py to catch invalid data types for field values.

Choice: 3

Q3 Add New City

---------------

Enter city name : Limerick

Country Code : IRL

District : Leinster

Population : hhhhh

countryCode IRL exists so city can be added

(1366, "Incorrect integer value: 'hhhhh' for column 'Population' at row 1")

#### Question 4

A try except block is used to ensure that engine size is a floating point number.

#### Question 5

A try except block is used to ensure that \_id is an integer and engine size is a floating point number.

Choice: 5

Q5 Add New Car

--------------

Enter \_id : jj

Enter Reg : 12-G-12

Enter Engine Size : 1.4

\_id must be an integer, Engine Size must be a number like 1.5. Please choose again.

Choice: 5

Q5 Add New Car

--------------

Enter \_id : 20

Enter Reg : 12-G-12

Enter Engine Size : hello

\_id must be an integer, Engine Size must be a number like 1.5. Please choose again.

In script2.py, a try except block is used to catch a pymongo DuplicateKeyError if the user-entered \_id already exists in the table.

Choice: 5

Q5 Add New Car

--------------

Enter \_id : 1

Enter Reg : 12-G-12

Enter Engine Size : 1

Connection successful - MongoClient(host=['localhost:27017'], document\_class=dict, tz\_aware=False, connect=True)

Error inserting document - \_id already exists.

#### Question 6

A global variable (num) is set up and incremented each time choice = 6 or choice = 7. The first time choice 6 or 7 is made, the full country table is stored in a variable called allcou. A message is printed to screen to indicate that the database is being read and tell the user how many rows were returned. All queries are then performed on that variable. The user is informed how many times 6 or 7 has been chosen.

I make sure that the user-entered country name is case insensitive via Python .lower() to catch Ireland and Kiribati for example.

Choice: 6

Q6 Countries by Name

--------------------

Option 6 or 7 has been chosen 1 time(s).

Enter Country Name : ir

Reading the database to retrieve country table. Rows returned: 239

Check if any Name contains ir

United Arab Emirates | Asia | 2441000 | Zayid bin Sultan al-Nahayan

Côte d’Ivoire | Africa | 14786000 | Laurent Gbagbo

Etc

#### Question 7

Same try except blocks are used as in Question 2 to ensure entered Popluation is a number and >=0.

If, elif, else is used to select which query to run and to ensure that the operator is a valid one.

Choice: 7

Q7 Countries by Population

--------------------------

Option 6 or 7 has been chosen 1 time(s).

Enter < > or = : >

Enter population : ggg

Reading the database to retrieve country table. Rows returned: 239

\*\*\* ERROR \*\*\*: Population must be a number, please choose again.

Choice: 7

Q7 Countries by Population

--------------------------

Option 6 or 7 has been chosen 2 time(s).

Enter < > or = : >

Enter population : -10

\*\*\* ERROR \*\*\*: Population must be >= 0, please choose again.

Choice: 7

Q7 Countries by Population

--------------------------

Option 6 or 7 has been chosen 3 time(s).

Enter < > or = : hhh

Enter population : 800000000

\*\*\* ERROR \*\*\*: Invalid operator, please use < > or =