Mark Biggar – G00376334

Review of innovations that have been applied to the final assignment for the Applied Databases (module – 52553) of GMIT’s Higher Diploma in Science – Data Analytics.

Applied Databases

Innovations Implementation

Contents

[Table of Figures 1](#_Toc9005826)

[Introduction 2](#_Toc9005827)

[MongoDB 2](#_Toc9005828)

[Python 3](#_Toc9005829)

[Bibliography 4](#_Toc9005830)

# Table of Figures

[Figure 1: Code for bulkWrite() and updateMany() 2](#_Toc9005818)

[Figure 2: Function to query the SQL ‘world’ ‘city’ table and convert to pandas dataframe 3](#_Toc9005819)

[Figure 3: Code sample for printing dataframe by number or percentage of rows 4](#_Toc9005820)

# Introduction

A number of “innovations” have been added to the various scripts, codes, and queries that were used to questions asked as part the Applied Databases final assessment. While these are described as “innovations”, in the sense that they were not covered on the module syllabus, they are not normally innovations, in the true sense of the word. Instead, they are methods, or approaches, that are normally a standard part of the platforms being used. Often, they were used to provide some additional functionality, or improve speed, such as pandas, or as another method of implementing the ask in the question.

The main “innovations”, and added controls, are as follows.

MongoDB

In Q4.3.3 (“*Redefine County Populations*”), as part of the MongoDB assignment, the script used a combination of ***bulkWrite()*** and ***updateMany()***. The ***bulkWrite()*** command is described as taking “*an array of write operations and executes each … in order*” (MongoDB, 2019). Likewise, the ***updateMany()*** command is used to update “all matching documents … using the update criteria to apply modifications” (ibid). The combination of the two commands allows for the use of multiple updates (e.g. 2 or more) to be implemented as part of a single script.

In the question, this was implemented using the following code:



Figure : Code for bulkWrite() and updateMany()

In this code, the ***updateMany()*** goes through the documents in the database, and filters them based on the population criteria set (i.e. less than 100,000, or greater than or equal to 100,000), and replaces the population with the terms “Small county” or “Large county”. The ***bulkWrite()*** provides the functionality to process all the documents, and attach the correct ***updateMany()*** filters to the correct records.

# Python

A number of functionalities and controls were added to the python scripts for the SQL querying, that were beyond the requests of the assignment The main addition was the introduction of a pandas dataframe. These dataframes were initially used to store the results of the SQL queries. This was accomplished through the following code:



Figure : Function to query the SQL ‘world’ ‘city’ table and convert to pandas dataframe

A similar dataframe was also constructed for the ‘country’ table in the ‘world’ database.

There were a number of advantages to this approach, including:

* Once the dataframe was populated (it was created as an empty global variable i.e. ***dc\_ = None***) the query would not need to be rerun[[1]](#footnote-1). This satisfies the requirements of the assignment for options 6 and 7, whereby the SQL database can only be polled once.
* Additional functionality, especially for data sampling, and user inputs.
* Increased speed, as it removes the need to constantly query the database.

The additional functionality provided by the dataframes allows for additional options for users. For example, in option 2 of the script menu, a user is asked to select a population, and whether to filter by greater than (>), equal to (=), or less than (<) that population. It is possible, depending on the population chosen, and the sign used, that there could be more than 4,000 rows returned. This would obviously be impractical for a user to read. As such, pandas dataframes allow for returning a sample using either a percentage of the rows returned, or a fixed number of rows. In both cases, the rows returned are a sample of the overall dataframe.

In the python script used for this assignment, when the number of returns exceeds 30 rows, the user is given the option to limit the number of rows they would like to display, by either a percentage, or a fixed number of rows.



Figure : Code sample for printing dataframe by number or percentage of rows

# Bibliography

* MongoDB. (2019). *db.collection.bulkWrite()*. Retrieved May 3rd, 2019, from MongoDB.com: https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/

1. *Unless there was an addition to the database, such as provided for in option 3 of the script menu. In this case, the dataframe for cities was reset to an empty variable.* [↑](#footnote-ref-1)