Applied Databases

Higher Diploma in Science in Data Analytics

1   Description

This document describes the final project specification for the Applied Databases module.

2   Marks

This project is worth 60% of the marks for the module.

2.1   Marking Scheme

~~85% of the marks will be awarded for implementing the functionality described in this document.~~

~~15% of the marks will be awarded for innovation and extra functionality.~~

~~Please describe your innovation (if any) in a document entitled *innovation.doc* which should be stored in the root folder of your project.~~

100% of the marks will be awarded for implementing the functionality described in this document.

2.1.1   Plagiarism

Plagiarism will be dealt with in accordance with the institute's Plagiarism policy.

3 Submission

Deadline for submissions is **Tuesday May 5th 2020 at 9:00am**.

- Firstly, download the file GXXXXXXXXXX.7z from Moodle
- Unzip it.
- Rename the unzipped folder from GXXXXXXXXXX to your Student Number e.g. G0012345678
- The folder contains 4 sub-folders:
    - **Innovation**
      ~~Write a Word/PDF document explaining any innovation/extra functionality you provided and place in this folder.~~
      ~~If none – just leave folder empty.~~

    - **Mongo-Queries**
      This folder contains 4 files, corresponding to each MongoDB question.
      Write only the exact MongoDB command for each question into the appropriate file.

    - **MySQL-Queries**
      This folder contains 5 files, corresponding to each MySQL question.
      Write only the exact MySQL command for each question into the appropriate file.

    - **Normalisation**
      Write a Word/PDF document with your answer to the Normalisation question and place in this folder.

    - **PythonApp**
      Write your Python App in this folder

- When you are finished, compress the folder – which is now called your Student number (using Winzip or 7zip) and upload to Moodle before the deadline.

4 Functionality

4.1 MySQL

Import the *world* database from *world.sql* to MySQL and write queries to satisfy the following.

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.

```
+----------+--------------+--------------+
| name     | dateArrived  | name         |
+----------+--------------+--------------+
| Arnhem   | 2002-04-14   | Netherlands  |
| Purulia  | 2002-06-20   | India        |
| Suzhou   | 2002-01-30   | China        |
| Tama     | 2009-02-13   | Japan        |
+----------+--------------+--------------+
4 rows in set (0.01 sec)
```

Figure 1 Example of output required for Q4.1.1

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.

```
+----------+----------------+
| name     | lifeexpectancy |
+----------+----------------+
| Moldova  |           64.5 |
| Ukraine  |           66.0 |
+----------+----------------+
2 rows in set (0.00 sec)
```

Figure 2 Example of output required for Q4.1.2

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 Age	*Stage* column output
Under 18	Child
Between 19 and 29	Late teens/Twenties
Between 30 and 39	Thirtysomething
40 or older	Other

```
+----------+------------+------+---------------------+
| personID | personname | age  | Stage               |
+----------+------------+------+---------------------+
|        1 | Tom        |   33 | Thirtysomething     |
|        2 | Alan       |   23 | Late teens/Twenties |
|        3 | Sean       |   30 | Thirtysomething     |
|        4 | Sara       |   25 | Late teens/Twenties |
|        5 | Jane       |   25 | Late teens/Twenties |
|        6 | Michael    |   19 | Late teens/Twenties |
+----------+------------+------+---------------------+
6 rows in set (0.00 sec)
```

Figure 3 Example output for Q4.1.3

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.

```
+----------------------+--------------------------------+------------+------------+
| name                 | name                           | language   | percentage |
+----------------------+--------------------------------+------------+------------+
| Basse-Terre          | Guadeloupe                     | French     |        0.0 |
| Basseterre           | Saint Kitts and Nevis          | English    |        0.0 |
| Belmopan             | Belize                         | English    |       50.8 |
| Bridgetown           | Barbados                       | English    |        0.0 |
| Castries             | Saint Lucia                    | English    |       20.0 |
| Charlotte Amalie     | Virgin Islands, U.S.           | English    |       81.7 |
| Ciudad de Guatemala  | Guatemala                      | Spanish    |       64.7 |
| Ciudad de México     | Mexico                         | Spanish    |       92.1 |
| Ciudad de Panamá     | Panama                         | Spanish    |       76.8 |
| Cockburn Town        | Turks and Caicos Islands       | English    |        0.0 |
| Fort-de-France       | Martinique                     | French     |        0.0 |
| George Town          | Cayman Islands                 | English    |        0.0 |
| Hamilton             | Bermuda                        | English    |      100.0 |
| Kingstown            | Saint Vincent and the Grenadines | English  |        0.0 |
| La Habana            | Cuba                           | Spanish    |      100.0 |
```

Figure 4 Example output for Q4.1.4

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.

```
+-------------+----------------+-------------+
| personname  | name           | Stay Length |
+-------------+----------------+-------------+
| Alan        | Arnhem         | Very Long   |
| Alan        | Purulia        | Long        |
| Alan        | Suzhou         | Very Long   |
| Alan        | Tama           | Very Long   |
| Michael     | Guaíba         | Long        |
| Michael     | Nagoya         | Very Long   |
| Michael     | Saint Helier   | Short       |
| Sara        | Jaunpur        | Very Long   |
| Sara        | Saint Helier   | Short       |
| Sara        | Zürich         | Long        |
| Sean        | Dordrecht      | Very Long   |
| Sean        | Shangqiu       | Very Long   |
```

Figure 5 Example output for Q4.1.5

4.2   MongoDB

Import the file *mongo.json* to a collection called *docs* in a database called *proj20DB* and write queries to satisfy the following.

4.2.1   Average Age of Students

Give the MongoDB command to find the average age of students.

```
{ "Average" : 33 }
```

Figure 6 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*.

```
{ "name" : "B.A.", "Honours" : true }
{ "name" : "B.Eng.", "Honours" : false }
{ "name" : "H. Dip. in Data Analytics", "Honours" : true }
{ "name" : "H. Dip. in SW Devel", "Honours" : true }
```

Figure 7 Honours Level

4.2.3   Qualified Students

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

```
{ "Qualified Students" : 6 }
```

Figure 8 Number of Qualified Students

4.2.4   Students 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:

```
{ "details" : { "name" : "Alan Higgins" }, "qualifications" : [ "ENG, SW" ] }
{ "details" : { "name" : "Bernie Lynch" }, "qualifications" : [ "ARTS" ] }
{ "details" : { "name" : "Brian Collins" }, "qualifications" : [ "ENG", "SW" ] }
{ "details" : { "name" : "John Smith" }, "qualifications" : [ "ARTS", "DATA" ] }
{ "details" : { "name" : "Mary Murphy" }, "qualifications" : [ "ARTS" ] }
{ "details" : { "name" : "Mick O'Hara" }, "qualifications" : [ "ENG", "DATA", "SW" ] }
{ "details" : { "name" : "Tom Kenna" }, "qualifications" : "None" }
```

Figure 9 Students and their Qualifications

4.3   Normalisation

4.3.1   Database Design

Examine the following database (consisting of one table, with the primary key = EmployeeID) that was designed to store the following information on a company's employees and departments:

- Employee ID
- Employee Name
- Employee Salary
- Department Name
- Department Location
- Department Budget

Give your opinion, **using examples from the data below**, on whether or not the current database is good or bad.

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	Maeve	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

Table 1 Proposed database table

4.4 Python

4.4.1 Python program

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

Figure 10 Main Menu

The choices are as follows:

- **1 (View People)**

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

  Figure 11 1st two people

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

  Figure 12 Next two people

  And so on until the user presses q:

  Figure 13 Next two people

  Whenever the user presses *q* he/she is brought back to the Main Menu.

- **2 (View Countries by Independence Year)**
  The user is asked to enter a year.

Figure 14 Enter Independence 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.

Immediately the user is brought back to the Main Menu.

Figure 15 Countries that became independent in 1993

- **3 (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*.

  Figure 16 New person added

  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:

  Figure 17 New city not added

- **4 (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:

  Figure 18 Countries matching search criteria

- **5 (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 =.

  Figure 19 Countries listed by population

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

  Figure 20 Invalid comparison operator

- **6 (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 *qualifications* attribute, nothing should be shown. But if he/she has a *qualifications* attribute this must be shown.

  Figure 21 Students at specified addrerss

- **7 (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:

  ```
  Choice: 7

  Add New Course
  --------------
  _id : BUS
  Name : Business Studies
  Level : 8
  ```

  Figure 22 New Course added

  These are then used to create a new document in mongodb as follows in the *docs* collection in the imported Mongo database. If the user enters an existing _id value an error message is shown.

  ```
  Choice: 7

  Add New Course
  --------------
  _id : DATA
  Name : Data Analysis
  Level : 8
  *** ERROR ***: _id DATA already exists
  ```

  Figure 23 New Course not added as _id already exists

- **x (Exit Application)**
  The program terminates

- **Anything Else**
  The menu is shown again.

NOTES

- For menu options 4 and 5 the information should be read from the database **only once.**
  E.g. If the user chooses 4 (View Countries by Name) or 5 (View Countries by Population) the countries are read from the database and stored in the program.

  If the user chooses 4 or 5 again, the information is **not** read from the database again. Instead, the information read the first time option 4 or 5 was chosen is used.

4.5 Testing

The MySQL and MongoDB sections are a pass/fail basis. Either you get all the marks or none.

You can test your answers as follows:

- Download the file **ExpectedQueryResults.7z** from Moodle and unzip it.
- It contains two folders:
    - MySQL
      This has 5 files which each of which has the correct output for the corresponding MySQL question.

    - MongoDB
      This has 4 files which each of which has the correct output for the corresponding MongoDB question.

4.5.1   How to test your MySQL queries

- Write your MySQL query in the MySQL console.
- When you think its correct copy the query to appropriate file in the MySQL-Queries folder of your answer folder.
- Run the following command from the Windows command line:

```
mysql.exe -u root -proot world < MySQL-Q411.txt > MySQL-Q411ANS.txt
```

mysql.exe is the location of mysql.exe e.g. "\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe"

-u root is the username, in this case root

-proot is the password, in this case root (no space between p and the password)

< The less than symbol means that the contents of the next file mentioned will be used as input to the mysql.exe command.

MySQL-Q411.txt is the location of the file with your MySQL query for this question e.g. "C:\Users\GHarrisson\Documents\New folder\GXXXXXXXX\MySQL-Queries\MySQL-Q411.txt"

> The greater than symbol means that the output from the mysql.exe command should be written to the file mentioned next

MySQL-Q411ANS.txt is the location of the file your query result will be written to e.g. "C:\Users\GHarrisson\Documents\New folder\MyAnswers\MySQL-Q411ANS.txt"

Figure 24 Creating MySQL result file

- Compare your answer with the correct answer:

  `fc` MySQL-Q411ANS.txt MySQL-411.txt

  fc the file tool compare program in windows

  MySQL-Q411ANS.txt is the location of the file your query result

  MySQL-411.txt is the location of the correct answer for this query.

Figure 25 Checking your MySQL result with actual result

- If the result of the fc command is not **FC: no differences encountered** no marks will be awarded for the question.

4.5.2   How to test your MongoDB queries
- Write your MongoDB query in the MongoDB shell.
- When you think its correct copy the query to appropriate file in the Mongo-Queries folder of your answer folder.
- Run the MongoDB command from the Windows command line as follows:

`mongo.exe proj20DB < Mongo-Q421.txt > Mongo-Q421ANS.txt`

`mongo.exe` is the location of mongo.exe e.g. "\Program Files\MongoDB\Server\4.0\bin\mongo.exe"

`proj20DB` is the name of the Mongo database

`<` The less than symbol means that the contents of the next file mentioned will be used as input to the mongo.exe command.

`Mongo-Q421.txt` is the location of the file with your Mongo query for this question e.g. "C:\Users\GHarrisson\Documents\New folder\GXXXXXXXX\Mongo-Queries\Mongo-Q421.txt"
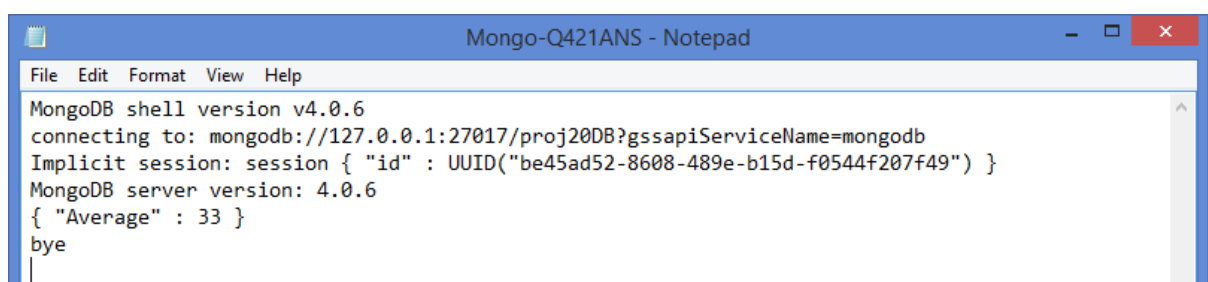
`>` The greater than symbol means that the output from the mongo.exe command should be written to the file mentioned next

`Mongo-Q421ANS.txt` is the location of the file your query result will be written to e.g. "C:\Users\GHarrisson\Documents\New folder\MyAnswers\Mongo-Q421ANS.txt"

```
C:\Users\GHarrisson>"\Program Files\MongoDB\Server\4.0\bin\mongo.exe" proj20DB < "C:\Users\GHarrisson\Documents\New fold
er\GXXXXXXXXX\Mongo-Queries\Mongo-Q421.txt" > "C:\Users\GHarrisson\Documents\New folder\MyAnswers\Mongo-Q421ANS.txt"
```

Figure 26 Creating Mongo Result file

- The first 4 lines of your query result file (**Mongo-Q421ANS.txt**) will have to be deleted so the fc command will only compare the actual output

```
Mongo-Q421ANS - Notepad
File  Edit  Format  View  Help
MongoDB shell version v4.0.6
connecting to: mongodb://127.0.0.1:27017/proj20DB?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("be45ad52-8608-489e-b15d-f0544f207f49") }
MongoDB server version: 4.0.6
{ "Average" : 33 }
bye
```

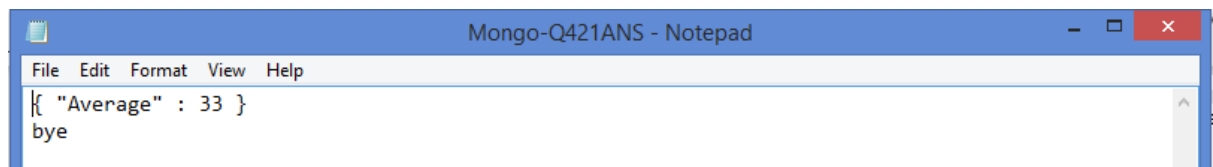*Figure 27 Mongo Query file before removal of first 4 lines*

Figure 28 Mongo Query file after removal of first 4 lines

- Compare your answer with the correct answer:

  `fc Mongo-Q421ANS.txt Mongo-Q421.txt`

  fc the file tool compare program in windows

  Mongo-Q421ANS.txt is the location of the file your query result

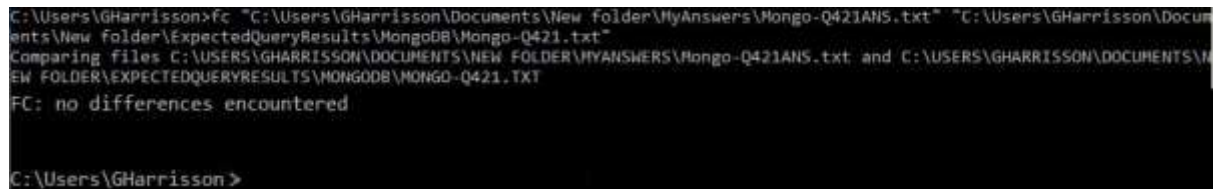 **Mongo-Q421.txt** is the location of the correct answer for this query.

Figure 29 Checking your Mongo result with actual result

- If the result of the fc command is not **FC: no differences encountered** no marks will be awarded for the question.