CCT College Dublin Continuous Assessment

Programme Title	Higher Diploma in Science in Computing (FT Sept 21)	**Assignment Type**	Integrated
Module Titles	Databases, Soft. Dev. Fund., Web Dev.	**Lecturers**	{ aldana | ken | mikhail }@cct.ie
Assignment Title	Data Manipulation and Validation	**Weightings**	20% – DB, 15% – SDF, 20% – WD
Issue Date	11/11/2021	**Last Revision**	28/11/2021
Submission Deadline	05/12/2021 @23:59	**Submitted to**	Moodle
Feedback Method	Moodle gradebook	**Feedback by**	22/12/2021

Submission Requirements
All submissions must meet the minimum requirements or the mark awarded may be affected

- For **Software Development** – submit a complete NetBeans project file in a **ZIP** format
- For **Databases & WebDev** – submit your SQL queries on one single **PDF** document with a GitHub repo link. Other file formats will not be accepted.
- Use Harvard Referencing when citing third party material
- Be the student's own work

- Late submissions will be accepted up to **5 calendar days** after the deadline.
- All late submissions are subject to a penalty of **10%** of the mark awarded.
- Submissions received **more than 5 calendar days** after the deadline above **will not** be accepted and a mark of **0%** will be awarded.

Learning Outcomes
This is not the assessment task (detailed on the next page). This CA will assess student attainment of the following Minimum Intended Module Learning Outcomes (MIMLOs):

Databases	Software Dev. Fund.	Web Dev.
1. Outline the rationale for DBMS and identify and appraise desirable properties of a DBMS. 4. Make use of SQL, DDL and DML to translate a database model into a live database.	1. Understand & employ fundamental concepts and principles of programming (variables, control structures, etc.) 2. Demonstrate a structured approach to algorithmic design and problem solving 3. Differentiate, select and utilise suitable APIs in the construction of software	3. Survey the short term and long term data storage needs of a client and implement robust data storage solutions with thorough backup and recovery plans to prevent downtime during disaster and easy maintenance. 5. Implement multi-tiered website architectures utilising existing frameworks and libraries to maximise development time, utilising the latest user interface widgets and components to develop a responsive and intuitive interface design

QQI Description of Attainment
Attainment of the learning outcomes is the minimum requirement to achieve a Pass mark (40%). Higher marks are awarded where there is evidence of achievement beyond this, in accordance with QQI Assessment and Standards, Rev. 2013, and summarised in the following table:

% Range	Description	Level 6, 7 & 8 awards
90 – 100% 80 – 89% 70 – 79%	Exceptional Outstanding Excellent	Achievement includes that required for a Pass and in **most** respects is significantly and consistently beyond this
60 – 69%	Very Good	Achievement includes that required for a Pass and in **many** respects is significantly beyond this
50 – 59%	Good	Achievement includes that required for a Pass and in **some** respects is significantly beyond this
40 – 49%	Acceptable	Attains all the minimum intended programme learning outcomes
35 – 39%	Fail	Nearly (but not quite) attains the relevant minimum intended learning outcomes
0 – 34%	Fail	Does not attain some or all of the minimum intended learning outcomes

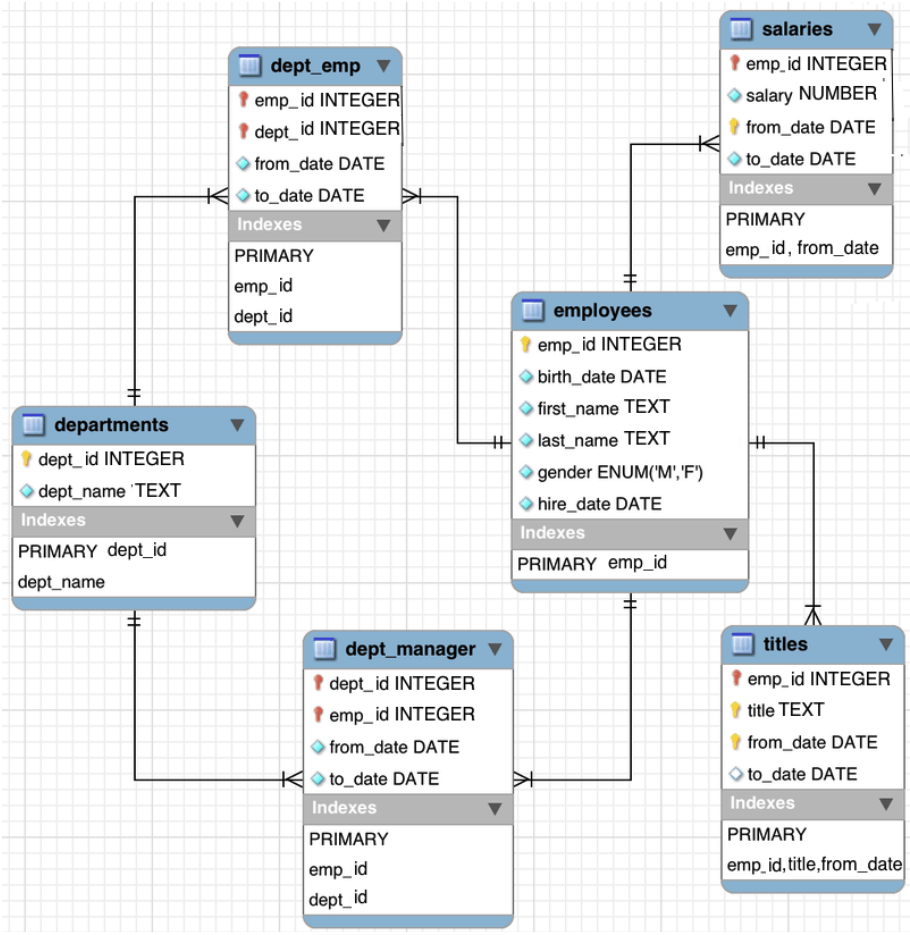
Please review the CCT Grade Descriptor available on the module Moodle page for a detailed description of the standard of work required for each grade band. The grading system in CCT is the QQI percentage grading system and is in common use in higher education institutions in Ireland. The pass mark and thresholds for different grade bands may be different from what you have experienced in the higher education system in other countries. CCT grades must be considered in the context of the grading system in Irish higher education and not assumed to represent the same standard the percentage grade reflects when awarded in an international context.

Additional Information

- Lecturers are not required to review draft assessment submissions. This may be offered at the lecturer's discretion.
- In accordance with CCT policy, feedback to learners may be provided in written, audio or video format and can be provided as individual learner feedback, small group feedback or whole class feedback.
- Results and feedback will only be issued when assessments have been marked and moderated / reviewed by a second examiner.
- Additional feedback may be requested by *emailing by a stated deadline.* Additional feedback may be provided as individual, small group or whole class feedback. Lecturers are not obliged to respond to email requests for additional feedback where this is not the specified process or to respond to further requests for feedback following the additional feedback.
- Following receipt of feedback, where a student believes there has been an error in the marks or feedback received, they should avail of the recheck and review process and should not attempt to get a revised mark / feedback by directly approaching the lecturer. Lecturers are not authorised to amend published marks outside of the recheck and review process or the Board of Examiners process.
- Students are advised that disagreement with an academic judgement is not grounds for review.
- For additional support with academic writing and referencing students are advised to contact the CCT Library Service or access the CCT Learning Space.
- For additional support with subject matter content students are advised to contact the CCT Student Mentoring Academy
- For additional support with IT subject content, students are advised to access the CCT Support Hub

Assignment Introduction

In this assignment, the **employees sample database** (created by Fusheng Wang and Carlo Zaniolo at Siemens Corporate Research), a large base of data spread over six separate tables and consisting of 4 million records in total that was created for system testing purposes. The following diagram provides an overview of the structure of the **employees.db**:



Employees.db Schema

- For the **Databases** module, you are required to write **SQL queries** that manipulate the data (see **Clause 1**).

- For the **Software Development Fundamentals** module, you are required to implement a **Java application** that can work with data and apply data validation and verification techniques (see **Clause 2**).

- For the **Web Development** module, you are required to use a **JavaScript library** in an **HTML file with CSS** styling in order to output the results of each **SQL query** (see **Clause 3**).

1. Databases CA Instructions

- This exercise requires composing **SQL** queries.
- In order to query the database with **SQL**, please use an online IDE – **https://sqliteonline.com:**
- Download **employees.db.bz2** archive from:
  - **https://github.com/siara-cc/employee_db/raw/master/employees.db.bz2**
- You must **unarchive** the **.bz2** file (with **7-Zip** [Win] or **Keka** [Mac] or similar) to get **employees.db** file
- Import your **employees.db** (not **employees.db.bz2** – please read above) into **https://sqliteonline.com**:
  - **File** → **Open DB** → locate **employees.db** on your computer → **Open**

For each of the questions below (Part 1-3), write the **SQL query in text** from **https://sqliteonline.com** and copy it into the final **PDF document** required for the submission.

1.1 Databases CA Part 1 (20%)

1. List all attributes present in the **departments** relation. [4 marks]

2. List all **employee IDs** of all past/current employees, their **first** and **last names**. [4 marks]

3. List all **job titles** held by any past/current employees. [4 marks]

4. List all **unique job titles** found in the database. [4 marks]

5. List all past/current **employees' names ordered alphabetically** in ascending order, i.e. first name and last name in alphabetical order. [4 marks]

1.2 Database CA Part 2 (40%)

1. List all department **titles** present in the database. [4 marks]

2. List the **total number** of all employees (past/current). [4 marks]

3. List **female employees** (past/current) together with all other relation attributes. [4 marks]

4. List past/current **employees** hired prior to **1986-01-01** with the surname **Simmel** [4 marks]

5. How many past/current **employees' last name** begins with the capital letter **B**?
   Use a column alias **total with B** to output your results. [4 marks]

6. Create a new table called **emp_training** with 3 columns:
   - **trainer_no**: this should be the primary key and is of type integer and is an auto-increment.
   - **first_name**: this data type is **varchar(30)** and should not be **NULL**
   - **last_name**: this data type is **varchar(30)** and should not be **NULL**
   - **t_module**: this data type is **varchar(20)** [4 marks]

7. Insert 2 new rows into the **emp_training** table:

   Row 1: fname: **Joe**
   lname: **Bloggs**
   module: **Google Docs**

   Row 2: fname: **Fred**
   lname: **Bloggs**
   module: **Google Sheets**

   [4 marks]

8. The organisation no longer wishes to record the employees training within the database. Therefore, delete the newly created **emp_training** table. [4 marks]

9. Alter the **employees** table to include an **email_address** field of type **varchar(20)**. [4 marks]

10. Update the email address of **Georgi Facello** to **gfacello@gmail.com**, where **emp_no** equals to **10001**. [4 marks]

1.3 Database CA Part 3 (40%)

1. The number of all employees that started on **1991-05-01**. [4 marks]

2. List all **emp_no** who have had strictly **more than 2 titles** and display **the total number of the titles** they have had. [4 marks]

3. The number of employees that have a current salary (i.e., **to_date** equals to **9999-01-01**) between **90000** and **90040**. [4 marks]

4. List all unique employees' last and first names (using **GROUP BY** method) that have a current salary (i.e., **to_date** equals to **9999-01-01**) greater than **90000**, outputting both names in descending order (sort by the last name first and then the first name) and also displaying their current salaries (using the **INNER JOIN** method). [4 marks]

5. First name, last name, all salary dates and related amounts for the employee with employee number **10012**. [6 marks]

6. In relation to the table named salaries in **Figure 1** above. Answer in text:
   a) What is the **degree** of this table?
   b) What column(s), if any, make(s) up the **primary key**?
   c) What column(s), if any, make(s) up the **foreign key**? [6 marks]

7. In the given schema, the tables **dept_emp**, **dept_manager**, **salaries**, **titles** have **composite** keys. Explain for each relation why this is the case? Answer in text. [12 marks]

2. Software Development CA Instructions

2.1 Software Development CA Part 1 (25%)

Create a **Java program** that will connect to the **Database** that you have downloaded/imported into **MySQL**. If there is an error in the connection. make sure that your program outputs a suitable error message for the user.

You will now create a simple console menu system to allow the user to carry out a selection of the **SQL** queries that you have been asked to complete. The last option on the menu selection should be 'Exit' or 'Quit'. If the user selects this option then the program should end. If the user selects an invalid option (at any time) then you need to display a suitable error message and re-display the menu.

If the user selects a valid option, then your program should carry out the appropriate task – see **Part 2** below.

2.2 Software Development CA Part 2 (75%)

For the menu system you must implement a **MINIMUM** of **THREE** queries that can be adjusted by the user input. For higher marks you should implement up to **SEVEN** of the queries.

In **EACH** case, you must validate the user's input so that the data used by the query is valid. So (for example) if the query requires certain text, then you have to check that the user's input will work. Similarly if it should be a number, or a number within a certain range, etc. If the user input is invalid then output a suitable message and re-prompt them.

You can then run the **SQL** query through your programme and output the results on the console. If there are no results then output a suitable message for the user.

Example 1 – see **Databases** part **2**, query **3** ("List all female…"). This could be implemented in your programme menu so that the user can decide whether to list all **female** OR **male** employees by making a choice:

```
**** MENU SYSTEM ****
  1) List all employees by gender
  2) etc…
  3) QUIT
**********************
```

<user selects option 1> – prompt user to enter the gender they are searching for…

Example 2 – see **Databases** part **3**, query **3** ("The number of employees that have a current salary…"). This could be implemented in your menu so that the user can decide the salary range:

```
**** MENU SYSTEM ****
  1) List all employees by gender
  2) List the number of current employees within a salary range
  3) etc…
  4) QUIT
**********************
```

<user selects option 4> – prompt user to enter the salary range they are seeking…

2.3 Software Development Marking Guide

PART 1

Program is able to connect to appropriate Database	[5 marks]
Menu system is clear / neatly presented, and functions as required (i.e. user selection validated / error messages are useful, etc.)	[10 marks]
Code is commented clearly	[10 marks]

PART 2

For <u>each</u> query implemented in the menu system (<u>where user can vary the query through keyboard input</u>)	[5 marks (max 35 marks)]
Code is commented clearly to explain your work	[10 marks]
Program is well-structured and modularised (i.e. you have written your own methods)	[20 marks]
Query results are displayed on screen in a clear, easy-to-understand manner	[10 marks]

3. WebDev CA Instructions (20 marks)

Produce a set of HTML files with the shared CSS styling that can read the database file and execute each query and answer questions from the **Databases CA** (Clause 1):

1. Use a SQL.js library and build **HTML** files, which will render the results of **SQL queries** run against an **SQLite database** in a browser:
   - **part1.1.html** for Part 1.1
   - **part1.2.html** for Part 1.2
   - **part1.3.html** for Part 1.3

   [1 mark]

2. You must assume that the original **employees.db** file is placed in the same folder where the **HTML** file is, but you **must not include it**, since the file exceeds the file size allowed by **GitHub**, when unarchived [1 mark]

3. Please create ***.gitignore*** file (like we did in class) in your repository and include **employees.db** there, so this file will be ignored and not be uploaded to **GitHub** [1 mark]

4. **CSS** file (**style.css**) should be in the same folder – you are free to use **Bootstrap**, if desired [1 mark]

5. Each question should have its **question number** and any **sub-part numbers** included in the button; Make sure you link all **HTML files** together **in the upper right menu** and the current page is shown in bold [1 mark]

6. Each query should have a green **Execute** button that executes each query and renders the results in the destination ** ** [1 mark]

7. Each **successful SQL query output** will count as **0.5 marks** in the **WebDev CA** (use **must use LIMIT 100 or 1000** in your SQL code as some of the queries with 1M records in the database can't be processed by a browser) [10 marks]

8. Make sure you use **HTML Escape Characters** when putting **SQL commands** inside ***data-sql*** attribute [1 mark]

9. Your **HTML page** should be **well-formed** and **pass validation** – https://validator.w3.org [1 mark]

10. The **CSS file** should be **external**, be **shared among all HTML pages** and **should pass validation** – https://jigsaw.w3.org/css-validator/ [1 mark]

11. All the files should be presented in a public **GitHub repository** with **at least 5 commits history** and the full link to this repository (of the following format: **https://github.com/<your username>/<repo title>**) must be included on the cover page of the **submitted PDF** – this aspect is also critical and is being marked (all final files must also be included in the zip with your submission) [1 mark]

You might get 0 for WebDev if the proper GitHub link is not present, it is not of the following format or it cannot be easily identified:
`https://github.com/<username>/<repo>`