Database Assignment

### Purpose and End User of my database

| The purpose of this database will be to document significant Marvel movie characters and their powers, abilities and a brief description of their backstories. The end users will be casual or hardcore fans who need a reminder of a character’s name or backstory, or which ones share the same powers. The end users can also be people who don’t know the Marvel Cinematic Universe (MCU) well enough and want to hold a conversation with a fan. |
| --- |

### Describe at least 3 implications that are relevant to your database and its use by the end user and why they are important

| **Usability**  My first relevant implication is usability. Usability is how easy it is to use a product, with an organised interface and few redundancies. This will also include not showing unneeded information, and making it comprehensible. I will make the instructions and prompts easy to understand, and lay them out in an easy to understand and read way. Usability is important because a user doesn’t want a product that is difficult to use, and so I need to design a database that doesn’t cause any confusion.  **Functionality**  Functionality is the way in which a product works. Often, usability and functionality is a trade-off, because sometimes the more features something has, the less usable it becomes. All the features a product has, both mechanical and internal, are part of functionality. If I add too many columns or features, the database could become repetitive, but I also want enough features that the database can be accessed and changed in all the ways I need it to. If I don’t keep an eye on the functionality of my database, it could be unusable or annoying.  **Future proofing**  How updateable a product or program is is future proofing. If my data becomes outdated, then it would need to be updated, and future proofing includes how easy it is to do so. Since my database will be to show characters in an ever-changing universe and fandom, I will need it to be easy to update when new information comes out. This will include adding characters and updating existing powers and backstories. If data is not updated, it could create confusion or misinformation. |
| --- |

### Database Design- Your Entity Relationship Diagram.

|  |
| --- |

### Database Testing Table: SQL Statements

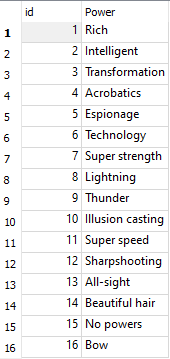
| **Purpose** | **SQL Statement** | **Result Success?** |
| --- | --- | --- |
| Join the character table on the bridge table | SELECT \*  FROM Characters  JOIN  Character\_powers ON Characters.id = Character\_powers.Character\_id; | Yes |
| Join all tables together | SELECT \*  FROM Characters  JOIN  Character\_powers ON Characters.id = Character\_powers.Character\_id  JOIN  Powers ON Powers.id = Character\_powers.Power\_id; | Yes |
| Show all text information | SELECT Characters.Name,  Characters.Alter\_ego,  Characters.Backstory,  Powers.Power  FROM Characters  JOIN  Character\_powers ON Characters.id = Character\_powers.Character\_id  JOIN  Powers ON Powers.id = Character\_powers.Power\_id; | Yes |
| Show all information regarding The Winter Soldier, not including the bridging table or IDs | SELECT Characters.Name,  Characters.Alter\_ego,  Characters.Backstory,  Powers.Power  FROM Characters  JOIN  Character\_powers ON Characters.id = Character\_powers.Character\_id  JOIN  Powers ON Powers.id = Character\_powers.Power\_id  WHERE Characters.id = 13; | Yes |
| Show just id, name and alter-ego of all characters | SELECT id,  Name,  Alter\_ego  FROM Characters; | Yes |
| Show data in Powers table | SELECT \*  FROM Powers; | Yes |

### Relevant Implications- Explain how your database addresses the relevant implications that you identified at the start.

| **Usability**  This database fulfils all the tasks I planned for it to do, while being pretty easy to use. It’s not easy to get lost in the menus, and you can cancel any action. I have made it easy to read and understand, so it is nice and painless to use. I have also commented my code sufficiently that if anyone would need to perform maintenance (although it shouldn’t be needed) they could understand what was going on. As the database increases in size, however, it may become difficult to find a specific character, as there is no way to search for them. As it is right now, with only a small list of under 20 characters, it is alright.  **Functionality**  My database does everything I wanted it to, and is also efficient. I have made it impossible to break, I have gotten multiple people to try and all were unsuccessful. By making a bridging table for the powers, I allowed multiple powers to be added to a single character, since there are heroes with more than one power. This did come up with other problems, such as not being able to show the powers in the “show all characters” action, but I got around this in a clean way that I think makes the result nicer anyway.  **Future proofing**  I ensured this database was easily updated. Since movie characters constantly change between movies, this was important. I made it easily possible to update a character, but also that you can skip data that you don’t want to change instead of having to re-write everything. |
| --- |

### Showcase:

Give evidence of your database and the Python code that interfaces with it. Use screenshots or a short video. Explain how it improved, how it functions, how it was tested etc.



| *Above: Structure table for powers table*  *Left: Data table for the powers*  *Above: Data table for characters*  *Below: Structure table for characters table*    *Left: Data table for the bridging. This allowed me to add multiple powers to each character and vice versa, using the JOIN statement. This isn’t the full table, there are 41 rows total.*  *Below: Structure table for the above bridging table.*    *Prints a table with all the characters Names, Alter-egos and IDs*  *Deletes a character*  *Continued on next page*  *Adding a character and their details*  *Printing only one character, if the character has multiple powers there will be multiple rows, so it separates the tuples before printing. The continued line is*  cursor.execute("SELECT Characters.Name, Characters.Alter\_ego, Powers.Power, Characters.Backstory FROM Characters JOIN Character\_powers ON Characters.id = Character\_powers.Character\_id JOIN Powers ON Powers.id = Character\_powers.Power\_id WHERE Characters.id = ?;", (character\_id,))  *The menu with options when the user is logged out. The logged-in menu has the same structure, except*  while account == False:  *becomes*  while account == True:  *And the options are*  print("What would you like to do? Input number for action.")  print("1. Create new character")  print("2. Show all characters")  print("3. Show all powers")  print("4. Show all information on one character")  print("5. Update a character")  print("6. Delete a character")  print("7. Logout")  print("8. End")  Not all the code is shown, these are examples of the code.  All of the above is in a file called “Functions”, and is 440 lines long. The main code is in a separate file called “Main Code”, which is 15 lines long and shown below:    I started the coding by building it up one function at a time. I would come up with a function I needed, then start writing it, and when I realised I needed a new function to put in, I would make the new function, then continue with the previous. Every two or three functions, I went over everything, bulletproofing anything that I thought could easily break, and also adding in comments if I felt I hadn’t added enough.  At the very end, I arranged everything to be in two menus, one when logged in and one when not, and tested everything. There were a lot more bugs than I thought, and it took me a solid week to fix everything. I then got my friends to test the code and try to break it, and did the same with the teacher. |
| --- |

**Teacher Checklist:**

**AS91879- Develop a digital outcome to manage data**

| Achieved- Develop a digital outcome to manage data | **Evidence** |  |
| --- | --- | --- |
| using appropriate tools and techniques to structure, organise, query and present data for a purpose and end user |  | Y |
| applying appropriate data integrity and testing procedures |  | Y |
| describing relevant implications. |  | Y |
| Merit- Develop an informed digital outcome to manage data |  |  |
| using information from testing procedures to improve the quality and functionality of the outcome |  | Y |
| structuring, organising and querying the data logically |  | Y |
| addressing relevant implications. |  | Y |
| Excellence- Develop a refined digital outcome to manage data |  |  |
| iterative improvement throughout the development and testing process | Great- love the added admin | Y |
| presenting the data effectively for the purpose and to meet end-user requirements. | Nice presentation | Y |

Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Achievement Standard.

**Develop a computer program**

**Credits:** 4 (Internal)

**NZQA:** <http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2018/as91883.pdf>

| **Achieved**  **Develop a computer program** | **Evidence** |  |
| --- | --- | --- |
| Wrote a program that performs a specific task using a suitable programming language |  | Y |
| Set out the program code clearly |  | Y |
| Documented the program with comments |  | Y |
| Tested and debugged to ensure that it works on a sample of expected cases |  | Y |
| **Merit**  **Develop an informed computer program** |  |  |
| Documented the program with variable names and comments that describe code function and behaviour |  | Y |
| Following conventions of the chosen programming language |  | Y |
| Tested and debugged the program in an organised way to ensure it works on expected and relevant boundary cases |  | Y |
| **Excellence**  **Develop a refined computer program** |  |  |
| Ensured the program is a well structured logical solution to the task |  | Y |
| Making the program flexible and robust |  | Y |
| Comprehensively tested and debugged the program |  | Y |

Comments:  
Really great program- unbreakable and very refined.