Name: Mark:

Database Assignment

Purpose and End User of my database

My database stores multiple movies, their names, their release year, length and rating. The end user is someone who wants information about movies to decide which one they want to spend their time watching. It can also be used by people who already know which kind of movie they want to watch and want to find more movies that they would like. Trusted people who know the password can also put new movies into the database.

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

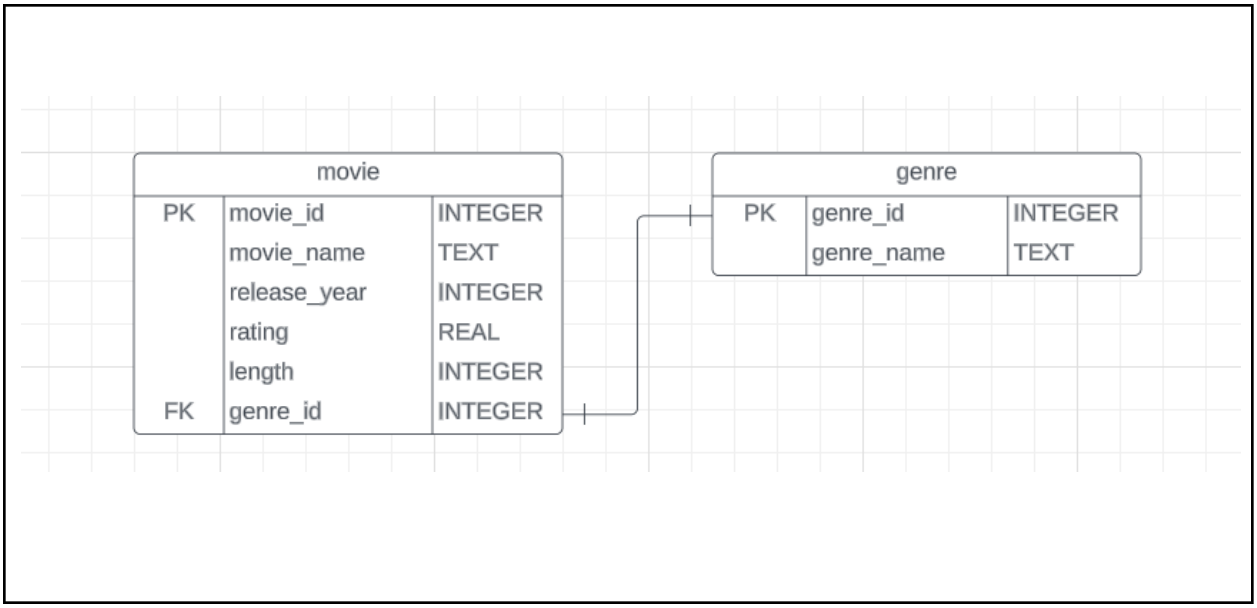
Functionality implications. When creating a movie database, we have to make sure that it is functional and does what it is supposed to do. For example, if the database shows the information for a different movie to the movie it was asked to show, then there could be people who watch a movie that they didn't want to instead. This is important as I want to give the users of my database accurate information about every movie.

Future proofing implications. When new movies come out, the database needs to be updated to make sure the information about the new movie is available in my database. For example, if the database only has information about old movies that not many people are interested in watching, the database will be less useful than if the information about new movies were added. I want my database to be useful for anyone who wants to watch any kind of movie and this will enable my database to do this.

Social implications. Some people like to watch movies with other people but one person may like a different genre than the person they are watching a movie with. For example someone may like horror movies and the person that they are watching a movie with likes action movies but both like adventure movies. I want to give people enough information about movies that people who have this problem can find a movie that they both want to watch to make sure both people enjoy the movie they watch.

Name: Mark:

Database Design- Your Entity Relationship Diagram.



Database Testing Table: SQL Statements

Purpose	**SQL Statement**	**Result Success?**
Select all everything from the table to show all everything in the table if the user wants to see everything.	SELECT * FROM movie	yes
Select everything except for the movie id because the id doesn't help the user in any way.	SELECT movie_name,release_year,length, rating FROM movie	yes
Select a specific item in the database. This is for when the user wants the information about one specific movie.	SELECT movie.movie_name,movie.release_year,movie.length,movie.rating,genre.genre_name FROM movie INNER JOIN genre ON movie.genre_id=genre.genre_id WHERE movie_id = ?	yes
Select the movie's id and movie's name to tell the user what the id for each movie is.	SELECT movie_id,movie_name FROM movie	yes

Name: Mark:

Select all the movies but show them in order from highest rated to lowest rated.	SELECT movie.movie_name,movie.release_year,movie.length,movie.rating,genre.genre_name FROM movie INNER JOIN genre ON movie.genre_id=genre.genre_id ORDER BY movie.rating DESC	yes
Select all of the movies but show them from the newest to oldest.	SELECT movie.movie_name,movie.release_year,movie.length,movie.rating,genre.genre_name FROM movie INNER JOIN genre ON movie.genre_id=genre.genre_id ORDER BY movie.release_year DESC	yes
Insert a new movie into the database.	INSERT INTO movie (movie_name,release_year,length,rating ,genre_id) VALUES (?,?,?,?,?)	yes
Delete a movie from the database.	DELETE FROM movie WHERE movie_name = 'Hypnotic';	yes
Join the movie table and the genre table so the movie table displays the genre name for each movie	SELECT movie.movie_id, movie.movie_name,movie.release_year,movie.length,movie.rating,genre.genre_name FROM movie INNER JOIN genre ON movie.genre_id=genre.genre_id	yes
Insert a new genre to the genre table	INSERT INTO genre (genre_name) VALUES (?)	yes
Print only the genre id and the genre name	SELECT * FROM genre	yes
Delete a genre from the genre table	DELETE FROM genre WHERE genre_id = (?)	yes
Print the movies that have a specific genre	SELECT movie.movie_name,movie.release_year,movie.length,movie.rating FROM movie INNER JOIN genre ON movie.genre_id=genre.genre_id WHERE movie.genre_id = (?)	yes
Select a specific genre and its id	SELECT * FROM genre WHERE genre_id = (?)	yes

Name: Mark:

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

I made a function that allows certain people who have the password to be able to add more movies. This addresses the future proofing implication as when new movies come out, someone can add the new movies.

I made a function that asks the user a genre that they like and then shows all of the movies that have that genre. This addresses the social implication as if two people want to watch a movie but there is only one genre that they mutually like, then they can type that genre in, and they will be shown movies that have that genre.

I thoroughly tested my program to make sure that everything works the way it is supposed to. This means that if the user tells the program to do something, it does it and not something else that the user did not tell it to do.Eg when the user asks to see the information about the movie The Matrix, the program shows the information about The Matrix and not about the movie Custody. Along with this, if the user inputs something that the program is not expecting, then the program does not break but rather tells the user that they did something wrong. These measures make sure the program addresses the functionality implication.

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.

https://drive.google.com/file/d/1CfncfogCnGfddIT1uOQB3Lbe0WAJDG8u/view?usp=drive_link

The code asks the user if they want to see all of the movies in the database, see all of the relevant information about a specific movie that they want to see, see all of the movies in order from highest rated to lowest rated, see all of the movies in order from newest to oldest, add a movie, delete a movie, delete a movie, add a genre, delete a genre, see all of the movie that have a genre that the user requested, or exit.

I only used to have one movie table but I added a genre table as well because one of my end user requirements was that the code had to help people who already know what kind of movie they want to watch. If I didn't add a genre table, and a function that shows all of the movies with a specific genre, then they would not get any suggestions on which movie to watch.

Name: Mark:

When I added the new genre table, I had to redo a lot of my sql queries as the previous queries did not join the two tables and the program was no longer able to do even the simple functions like printing all the information for the movies. Once I learnt how to use joins in sqlite and wrote the new queries, they worked as intended.

I tested the code by inputting things that were not asked to be inputted by the program to see if it broke the code. The first time I tried to do this it did break. This is no longer the case and whenever the code receives an input that it does not know what to do with, it prints out an error message for the user to read and then goes back to asking the user what they want to do. Eg when the program asks the user what the ID for the movie they want all of the information for, and the user inputs “Fish” instead of one of the IDs then the program prints “That is not a real ID!” instead of breaking like it used to. I did this by try and exepts, so it tries to find the movie with the ID of “Fish” but if it can’t then it prints the error message.

Only people who know the password can add movies and genres, and delete movies and genres. In the code that I sent in the zip file was told every user what the password was because if I hadn’t then the person marking would not be able to test everything but if the code was used only for the end user, then the password would not be printed. The reason I made those 4 functions password protected is so random people cannot add or delete anything from the database as that would not be very smart because someone could come along and delete everything in the database, making the database and the program useless.

Name: Mark:

Teacher Checklists:

AS91879- Develop a digital outcome to manage data

Credits: 4
NZQA: https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91879.pdf

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		✓
applying appropriate data integrity and testing procedures		✓
describing relevant implications.		✓
Merit- Develop an informed digital outcome to manage data		
using information from testing procedures to improve the quality and functionality of the outcome		✓
structuring, organising and querying the data logically		✓
addressing relevant implications.		✓
Excellence- Develop a refined digital outcome to manage data		
iterative improvement throughout the development and testing process		✓
presenting the data effectively for the purpose and to meet end-user requirements.		✓

Name: Mark:

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

Comments:

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