Name: Mark:

Database Assignment

Purpose and End User of my database

The Purpose of my Video Game Collection database is for a “storage” area of all the games I have played to track my progress on the games and how far I have to finish the game. The database can also be used for people to see how much I liked the game by what rating I gave it and how long I have played the game for, they can also see other statistics of the games such as platforms and genres if they are thinking about buying / playing a certain game on the database. The end user(s) of my database are people who are looking to buy or play a game but are uncertain and want to do some research, me so I can check how What genre the game is in and how many games I have of a specific genre, this way I can see what genre I like the most and find other games of that genre that I could potentially play. Find what games are on what platforms, this means I can see if I can play these games with my friends that don’t play the game on the same platform as me, I can also check if I have the platform to play certain games in case I want to buy and play said game.

Name: Mark:

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

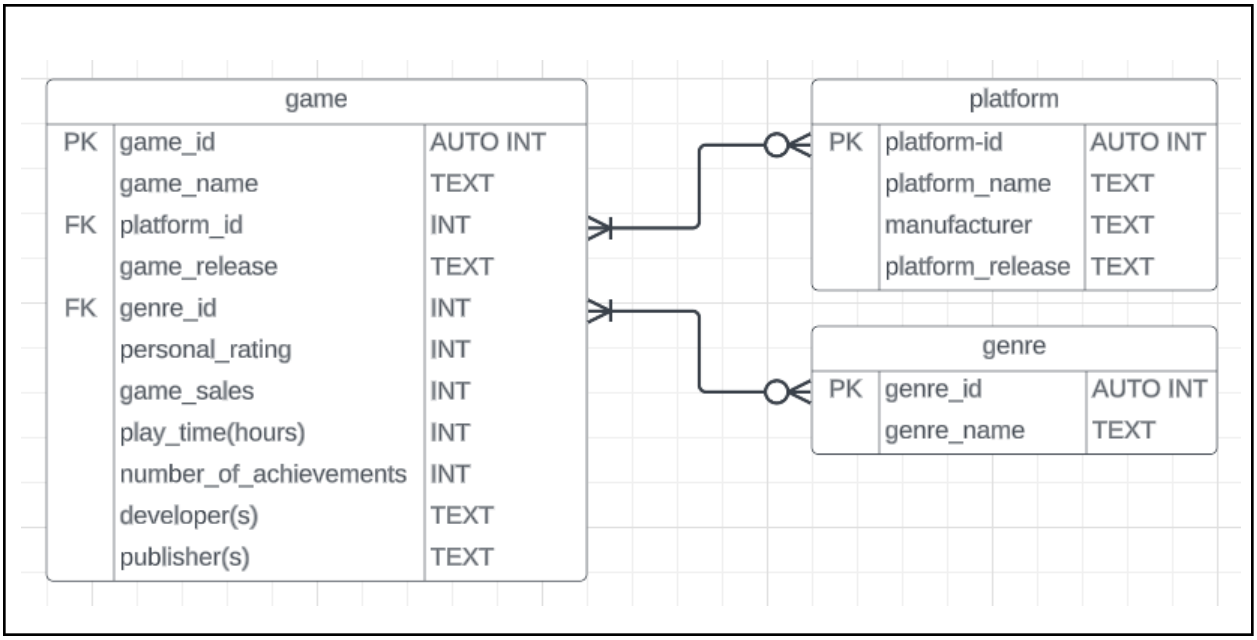
Functionality:
Functionality is one of if not the most important implication as when you create a database you need to make sure it actually works. In my video game collection database making my database to be able to function properly and work is important to the end users. This is because if the database does not function or work then there is no use for the database and it has no purpose which means that spending time to work on it was redundant.

Intellectual Property:
Intellectual Property is another very important relevant implication that should be put into consideration when creating a database that is going to be used. It's important as stealing someone else's data or stealing their idea without asking is an easy way to get yourself into trouble. If you stole data for your database from a company or person and they find out then you can get into some serious legal problems about copyright infringement which can get your database taken down. Ensuring that the data and ideas are yours and not copied / stolen from other sources without asking can show that the database is more professional and more reputable for the end users of the database.

Privacy:
Privacy in a database is a major relevant implication which you should think about during the creation of a database. Securing the contents and the data of its users is important as the database itself, this is because if the database and the users' data is breached then it could lead into some catastrophic problems such as identity theft, money being taken out of bank accounts, finding out where you live and more. Protecting your end users' privacy and ensuring the security of their data will show that the database is very reliable and safer to use than other databases created for a similar purpose, this can attract more end users who decide to choose your database over others.

Name: Mark:

Database Design- Your Entity Relationship Diagram.



Database Testing Table: SQL Statements

Purpose	**SQL Statement**	**Result Success?**
To see if the database actually works	SELECT * FROM game	Yes
Inner joins the genre and game tables to show the name of a genre and the names of the games in that genre	SELECT genre.genre_name, game.game_name FROM genre INNER JOIN game ON genre.genre_id = game.genre_id WHERE genre.genre_id = 7	Yes
See what games are RPG's	SELECT game_name FROM game WHERE genre_id = 7	Yes
Check if the genre table works	SELECT * FROM genre	Yes
Check if the platform table works	SELECT * FROM platform	Yes
Check if inserting new genres works	INSERT INTO genre (genre_name, genre_id) VALUES (	Yes

Name: Mark:

Count how many games are in the game table that are in the Adventure genre	SELECT COUNT(game_name) FROM game WHERE genre_id = 3	Yes
Check if deleting platforms works	DELETE FROM platform WHERE platform_name = 'Computer'	Yes

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

The first of the relevant implications that I identified was functionality, my database addressed this by doing what it was intended to do for its end users, me included. It is a quick and easy way to check through games that I own, what genre they are in, what platform(s) they're on and ways to add new games, genres or platforms.

Next was intellectual property, while I might not have created or helped in the creation of any of the games or platforms in the database, use of their names and data that is on the database is not violating their copyright or trademark on the game / platform. This means the database cannot get taken down for infringement on copyrights and shows its reliability to any end user.

Lastly it privacy, the database might not have any extra security measures or programs running on it that ensure nothing gets through and into the database or into an end user's data, the database is private and can only be accessed and edited by anyone that has the actual .db file, as long as someone doesn't download this to their computer behind your back while you aren't looking, your data is very much secured and away from any harm that hackers could cause you.

Name: Mark:

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.

```
        #Asks user what they want to do with their genre database
        if user_input2 == "2":
            while True:
                user_input3 = input("\nWhat do you want to do?\n1. Print All genres\n2. Add genre\n3. Delete genre\n4. Exit\n")
                if user_input3 == "1":
                    show_genres(connection)
                elif user_input3 == "2":
                    genre_name = input("genre Name: \n")
                    add_genre(connection,genre_name)
                    if genre_name == "back":
                        break
                elif user_input3 == "3":
                    genre_name = input("genre Name: \n")
                    if genre_name == 'back':
                        break
                    delete_genre(connection, genre_name)
                elif user_input3 == "4":
                    #go back to main menu
                    break
        #asks user what they want to do with their platform database
        if user_input2 == "3":
            while True:
                user_input4 = input("\nWhat do you want to do?\n1. Print All platforms\n2. Add platform\n3. Delete platform\n4. Exit\n")
                if user_input4 == "1":
                    show_platforms(connection)
                elif user_input4 == "2":
                    platform_name = input("platform Name: \n")
                    if platform_name == 'back':
                        break
                    add_platform(connection,platform_name)
```

```
with sqlite3.connect(DATABASE_FILE) as connection:
    while True:
        #asks user which table they want to edit
        user_input2 = input("\nWhich Table do you want to edit?\n1. game Database\n2. genre Database\n3. platform Database\n")
        #informs user how to cancel certain code actions
        print("to go back to menu when adding/deleting enter game/genre/platform name as 'back'")
        #asks user what they want to do with their game database
        if user_input2 == "1":
            while True:
                user_input = input("\nWhat do you want to do?\n1. Print All games\n2. Delete games\n3. Add games\n4. Show games by platform
                if user_input == "1":
                    show_games(connection)
                elif user_input == "3":
                    print("Make Sure You Input the correct Id for your platform/genre when adding a new game. Check platform/genre ID via p
                    game_name = input("game Name: \n")
                    if game_name == "back":
                        break
                    show_genres(connection)
                    genre_name = input("Which genre Produced?(via ID): \n")
                    show_platforms(connection)
                    platform_name = input("platform ID: \n")

                    add_game(connection,game_name,genre_name, platform_name)
                elif user_input == "2":
                    game_name = input("game Name: \n")
                    if game_name == "back":
                        break
                    delete_game(connection, game_name)
                elif user_input == "5":
                    break
                #goes back to main menu
```

Name: Mark:

```
        cursor.execute(sql)
        #executes SQL statement
        results = cursor.fetchall()
        #takes all data from specific database
        print(f"{'platform_name':<20}{'platform_id':<20}")
        for item in results:
            print(f"{item[1]:<20}{item[0]:<20}")
    except:
        print("Something went wrong with the connection")
#function to add a platform
def add_platform(connection, platform_name):
    try:
        cursor = connection.cursor()
        #Sql statement
        sql = "INSERT INTO platform (platform_name) VALUES (?)"
        cursor.execute(sql, (platform_name,))
        #executes SQL statement
        connection.commit()
        #saves any changes made to SQL database
        print(f"{platform_name} has been added")
    except:
        print("Item couldn't be added")
#function to delete platform
def delete_platform(connection, platform_name):
    try:
        cursor = connection.cursor()
        #Sql statement
        sql = "DELETE FROM platforms WHERE platform_name = (?)"
        cursor.execute(sql, (platform_name,))
        #executes SQL statement
        num_rows_affected = cursor.rowcount
        if num_rows_affected == 0:
```

Name: Mark:

```python
#function to print all genres
def show_genres(connection):
#    try:
        cursor = connection.cursor()
        #Sql statement
        sql = "SELECT * FROM genre"
        cursor.execute(sql)
        #executes SQL statement
        results = cursor.fetchall()
        #takes all the data from a specific database
        print(f"{'genre_name':<20}{'genre_id' :<30}")
        for item in results:
            print(f"{item[1]:<20}{item[0]:<20}")
#    except:
#        print("Something went wrong with the connection")
#function to add genre
def add_genre(connection, genre_name):
    try:
        cursor = connection.cursor()
        #Sql statement
        sql = "INSERT INTO genre (genre_name) VALUES (?)"
        cursor.execute(sql,(genre_name,))
        #executes SQL statement
        connection.commit()
        #saves changes to sql databse
        print(f"{genre_name} has been added")
    except:
        print("Item couldn't be added")
#function to delete genres
def delete_genre(connection, genre_name):
    try:
        cursor = connection.cursor()
```

Name: Mark:

```
    except:
        print("Something went wrong with the connection")
#function to add game
def add_game(connection, game_name, platform_id, game_release, genre_id, personal_rating, game_sales, hours_played, number_of_achievements,
    try:
        cursor = connection.cursor()
        #Sql statement
        sql = "INSERT INTO games (game_name, platform_id, game_release, genre_id, personal_rating, game_sales, hours_played, number_of_achi
        cursor.execute(sql,(game_name, platform_id, game_release, genre_id, personal_rating, game_sales, hours_played, number_of_achievemen
        #executes SQL statement
        connection.commit()
        #saves the changes to sql database
        print(f"{game_name} has been added")
    except:
        print("Item couldn't be added")
#function to delete game
def delete_game(connection, game_name):
    try:
        cursor = connection.cursor()
        #Sql statement
        sql = "DELETE FROM games WHERE game_name = ?"
        cursor.execute(sql,(game_name,))
        #executes SQL statement
        num_rows_affected = cursor.rowcount
        if num_rows_affected == 0:
            print("couldn't find item")
        else:
            connection.commit()
            #saves the changes to sql database
        print(f"{game_name} has been deleted")
    except:
```

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: Well written code. The next step of this would be to incorporate joins to better display the linked data in your tables. For instance I cant see the genres and platforms for Minecraft in any easy to recognise way.

Final grades will be decided using professional judgement based on a holistic examination of

Name: Mark:

the evidence provided against the criteria in the Achievement Standard.