read/write CSV with python

possible data page

BURNSIDE HIGH SCHOOL
WAIMAIRI-IRI

Internal Assessment Resource

Digital Technologies | Hangarau Matihiko

Level 1

This resource supports assessment against Achievement Standard 91902 and 91906

Standard title: Use complex techniques to develop a database

Use complex programming techniques to develop a computer program

Credits: 10 Credits

Resource title: **My Data**

This resource:

- Clarifies the requirements of the achievement standard
- Supports good assessment practice
- Should be subjected to the school's usual assessment quality assurance process
- Should be modified to make the context relevant to students in their school/kura environment and ensure that submitted evidence is authentic

Authenticity of evidence	Teachers/Kaiako must manage authenticity for any assessment from a public source, because students may have access to the assessment schedule or student exemplar material. Using this assessment resource without modification may mean that students' work is not authentic. The teacher/kaiako may need to change figures, measurements or data sources or set a different context or topic to be investigated or a different text to read or perform. All code must be written by the students and a "docstring" at the top is recommended. Students must write their report in their own words and reference any quotes accordingly.

Internal Assessment Resource

This resource supports assessment against Achievement Standard 91902 and 91906

Standard title:	Use complex techniques to develop a database Use complex programming techniques to develop a computer program
Credits:	10 Credits
Resource title:	**My Data**

Student/Ākonga instructions

Introduction/Kupu Arataki

Databases are everywhere. For example, the massive stock databases like those owned and managed by Amazon and the massive amounts of user data gathered by Google every second. Applications have to be able to Create, Read, Update and Delete this data- that is commonly referred to as CRUD. This is usually done by web applications or other types of database applications managed and written by real (highly paid!) people and some elements of this have to be open to users. This brings with it obvious implications around security, privacy, functionality and usability to name just a few.

Task/Hei Mahi

You will:

1. Research and decide on a purpose and end user for a data project
2. Design the database structure with an ERD.
3. Create the database in SQLite Studio. This could include writing code to parse data from another source (eg. a CSV file)
4. Write a Python program to Create, Read, Update and Delete data in the database
5. Address any relevant implications

The code and supporting evidence is due in Term 1

Part 1- Research and Decide

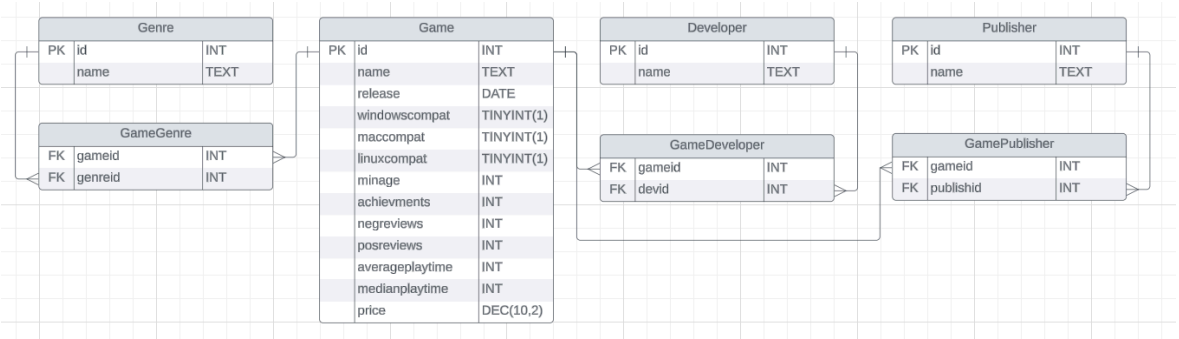
You must decide on a database purpose and end user. This should be kept fairly simple with between 1 and 3 tables. Examples might include:

- Data about crime, health or education for large datasets such as:
    - https://www.pewresearch.org/internet/datasets/

    - https://data.gov/
    - https://catalogue.data.govt.nz/dataset
    - https://www.stats.govt.nz/
- A simple food shop ordering system for customers to make orders (like Dominoes©!)
- A Video Game rating and review database to help game buyers make informed decisions
- A PC Parts database to help buyers create and cost a new PC Build
- A Character Database (eg Marvel Heroes or LoL) to help players find more information and statistics on their favourite characters
- A Quiz program to help students study for a subject at school and make their own questions to help their learning
- Any other similar database purpose and end user

Once you have researched and decided you need to create the **Entity Relationship Diagram**. This will include the tables, fields and data types as well as any Primary Keys or Foreign Keys. This MUST contain at least one many to many relationship.

Add the Entity Relationship Diagram here



CheckPoint #1- Get this design checked off by your teacher!

Part 2- Create the Database

You must use the design to create the database. You will create the database file, the tables, the fields and any primary keys or foreign keys you need. You may need to write code to parse data from large external datasets into your database.
Once this is done, the database should be tested to ensure it functions as expected with several sql queries that select, sort, filter, insert, update and delete entries. Decide on a number of queries that you will probably need for this data and write and test the SQL queries on your database in SQLite Studio. Add them to the table below.

SQL Query Testing Table		
Purpose	**Query**	**Pass/Fail**
Read games with id below 1000	SELECT * FROM Game WHERE id<1000;	Pass
See all game info including genres developers and publishers but not including the bridging tables for games with id below 1000	SELECT Game.name, Game.releasedate, Game.windowscompat, Game.maccompat, Game.linuxcompat, Game.minage, Game.achievments, Game.negreviews, Game.posreviews, Game.averageplaytime, Game.medianplaytime, Game.price, Genre.name, Developer.name, Publisher.name FROM Game JOIN GameGenre ON Game.id = GameGenre.gameid JOIN Genre ON Genre.id = GameGenre.genreid JOIN GameDeveloper ON Game.id = GameDeveloper.gameid JOIN Developer ON GameDeveloper.devid = Developer.id JOIN GamePublisher ON Game.id = GamePublisher.gameid JOIN Publisher ON Publisher.id = GamePublisher.publishid WHERE Game.id < 1000;	Pass
Add a game called “TEST GAME” with random input (i make it up) only in Game table	INSERT INTO Game VALUES (54321, 'TEST GAME', ‘2024-3-14’, 1, 1, 1, 0, 69, 420, 69420,	Pass

	25, 99999, 420.69)	
Update the game “TEST GAME” to have 5 more positive reviews	UPDATE Game SET posreviews = 69425 WHERE name = 'TEST GAME';	Pass
Delete the game “TEST GAME”	DELETE FROM Game WHERE name = 'TEST GAME';	Pass
Show all games with “count” somewhere in the name	SELECT * FROM GAME WHERE name LIKE "%count%"; for python it has to be SELECT * FROM GAME WHERE name LIKE ?; and ? is “%count%”	Pass

Part 3- Create a Program in Python

You now need to plan and **incrementally** develop a program for your end-user (and potentially for the database system administrator) to interface with this database.
This will most likely include the ability for the user to:

- Create entries in the database
- Read, filter, sort and display relevant data from the database
- Update selected data if required
- Delete selected data from the database if required

If you have not already done so, create a local folder for your code and **publish it to github.**

Start small with simple functionality. Test regularly and improve the program by adding in more features. Commit regularly with good commit messages to your github repo.
The github repo will be your final submission and your commit history will help show how much it improved.

Submit a link to your github repo below:
https://github.com/AlexCotiga07/Steam-Database

Evidence of Program Testing

Your final program must now be tested thoroughly before submission. Think of all the possible inputs that the program can be given and test them all. Write the tests down in the table below in a format that best suits your program. You may wish to test small parts of the program (specific functions) or the whole program.
Don't forget to include expected, boundary and invalid data inputs. Try to break your code!!

Program Testing Table			
What are you testing	**Input Value(s)**	**Expected Result(s)**	**Pass/Fail**
Ending program from menu (rest of commands are tested while testing everything else)	11	program ends	pass
unexpected value in menu	hello	"Invalid command" and asks again	pass
reading one game, boundary testing	0, -555, 999999999 (9 9 times), 9999999999 (9 10 times), ahhhhhhh, 11 (no game with this id)	"Not a valid id" OR "That game doesn't exist" and asks again	999999999 and 11 failed, rest passed
reading one game boundary testing retry	11, 999999999 (9 9 times)	"That game doesn't exist"	pass
showing all genres	N/A	prints all the genres in a list alongside their IDs, there are fewer than 40 genres so there should not be a page switching mechanic	pass
showing all developers	N/A	Shows list of first 40 (in alphabetical order) developers with options to switch pages	pass
switching pages	From first page showing devs: next, Next, NEXT, end, End, END, hello, ahhhh, 42, /	next - "Invalid command, try again." Next - "Invalid command, try again."	all pass

		NEXT - switches to next 40 developers end - “Invalid command, try again.” End - “Invalid command, try again.” END - returns to menu hello - “Invalid command, try again.” ahhhh - “Invalid command, try again.” 42 - “Invalid command, try again.” -42 - “Invalid command, try again.” / - “Invalid command, try again.”	
switching pages from second page	from second page showing devs:	next - “Invalid command, try again.” Next - “Invalid command, try again.” NEXT - switches to next 40 developers back - “Invalid command, try again.” Back - “Invalid command, try again.” BACK - switches to previous (first) 40 devs end - “Invalid command, try again.” End - “Invalid command, try again.” END - returns to menu hello - “Invalid command, try again.” ahhhh - “Invalid command, try again.” 42 - “Invalid command, try again.” -42 - “Invalid command, try again.” / - “Invalid command, try again.”	all pass
showing all publishers (same function used to show pages as showing developers)	N/A	shows first 40 (in alphabetical order) publishers just like developers	pass

show games in 1 genre (for those with a lot of games it will use pages like showing devs and publishers, those with not enough for multiple pages will test the part or)	1, 2, 0, -42, 42 (doesn't exist), 999999999 (9 9 times), 9999999999 (9 10 times), hello, /	1 - shows list of games 2 - shows list of games 0 - "Not a valid ID" -42 - "Not valid ID" 42 - "That genre doesn't exist" 999999999 - "That genre doesn't exist" 9999999999 - "Not a valid ID" hello - "Not a valid ID" / - "Not a valid ID"	42 and 9999999 99 failed, rest passed
show games in 1 genre retry	42, 999999999	42 - "That genre doesn't exist" 999999999 - "That genre doesn't exist"	pass
show all games by one developer	1, 2, 0, -42, 42, 999999999 (9 9 times), 9999999999 (9 10 times), hello, /	1 - shows by dev 1 2 - shows by dev 2 0 - "Not a valid ID" -42 - "Not a valid ID" 42 - shows by dev 42 999999999 - "That developer doesn't exist" 9999999999 - "Not a valid ID" hello - "Not a valid ID" / - "Not a valid ID"	all pass
show all games by 1 publisher	1, 2, 0, -42, 42, 999999999 (9 9 times), 9999999999 (9 10 times), hello, /	1 - shows by publisher 1 2 - shows by publisher 2 0 - "Not a valid ID" -42 - "Not a valid ID" 42 - shows by publisher 42 999999999 - "That publisher doesn't exist" 9999999999 - "Not a valid ID" hello - "Not a valid ID" / - "Not a valid ID"	all pass
search game by name	count, aiughfaishf	count - shows all with "count" somewhere in the name aiughfaishf - "No results"	pass
search dev by name	lar, aiughfaishf	lar - shows all with "lar" somewhere in the name aiughfaishf - "No results"	pass

search publisher by name	lar, aiughfaishf	lar - shows all with “lar” somewhere in the name aiughfaishf - “No results”	pass
Copy pasting a wikipedia article into all inputs	https://en.wikipedia.org/wiki/Cat everything on this page	Shouldn’t be accepted, output depends on where it is input	Because of the formatting the article isn’t very helpful
Copy-pasting lots of stuff from a wiki article into inputs	first couple of paragraphs of above article	Shouldn’t be accepted, output depends on where it is input	All pass, proofed against this scenario though just in case

Part 4- Relevant Implications

In your report explain how you addressed any relevant implications for your database and program. Take into account the purpose and end user(s) and the nature of data storage and distribution.

Implications	**How I addressed it (if relevant)**
social	
cultural	
legal	The data I used was a csv that was publicly available online and that said it was usable for any purpose
ethical	
intellectual property	The data I used was a csv that was publicly available online and that said it was usable for any purpose
privacy	
accessibility	When an incorrect value is input by the user, the program says what is wrong in words rather than saying an error in computer

	terminology. This makes it so anyone using this that can read english knows what went wrong.
usability	When the user makes an error, the program loops so they can try again rather than having to start over every time. There is also a way to search items by name, so that you don't need to know the ID of all items until you need it.
functionality	A user can read data generally or specifically, by seeing large chunks of data or searching within a specific field
aesthetics	
sustainability and future-proofing	
end-user considerations	Users won't be expected to know the ID of all games, genres, developers and publishers because they can search these by name and get the ID from that search. They also won't necessarily know all error codes so if something is input incorrectly the program says the error in regular words. The user is expected to be someone looking for data on games, so there is a lot of useful data available that the user may need (like rating, genres, achievements etc.)
health and safety.	

Teacher Checklists:

AS91906 - Use complex programming techniques to develop a computer program

Credits: 6 (Internal)
NZQA: https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91906.pdf

Achieved	**Evidence**	
writing code for a program that performs a specified task		
using complex techniques in a suitable programming language		
setting out the program code clearly and documenting the program with comments		
testing and debugging the program to ensure that it works on a sample of expected cases.		
Merit		
documenting the program with appropriate names and organised comments that describe code function and behaviour		
following conventions for the chosen programming language		
testing and debugging the program effectively to ensure that it works on a sample of both expected cases and relevant boundary cases.		
Excellence		
ensuring that the program is a well-structured, logical response to the specified task		
making the program flexible and robust		
comprehensively testing and debugging the program.		

AS91902- Use complex techniques to develop a database

Credits: 4 (internal)
NZQA: https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91902.pdf

Achieved- develop a database	**Evidence**	✔
designing the structure of the data		
using appropriate tools and advanced techniques to organise, query and present data for a purpose and end users		
applying appropriate data integrity and testing procedures		
addressing relevant implications.		
Merit- Develop and informed database		
using information from testing procedures to improve the quality of the outcome		
structuring, organising and querying the data logically		
Excellence- develop a refined database		
iterative improvement throughout the design, development and testing process		
using efficient tools and techniques in the outcome's production		
presenting the data effectively for the purpose and end users.		

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