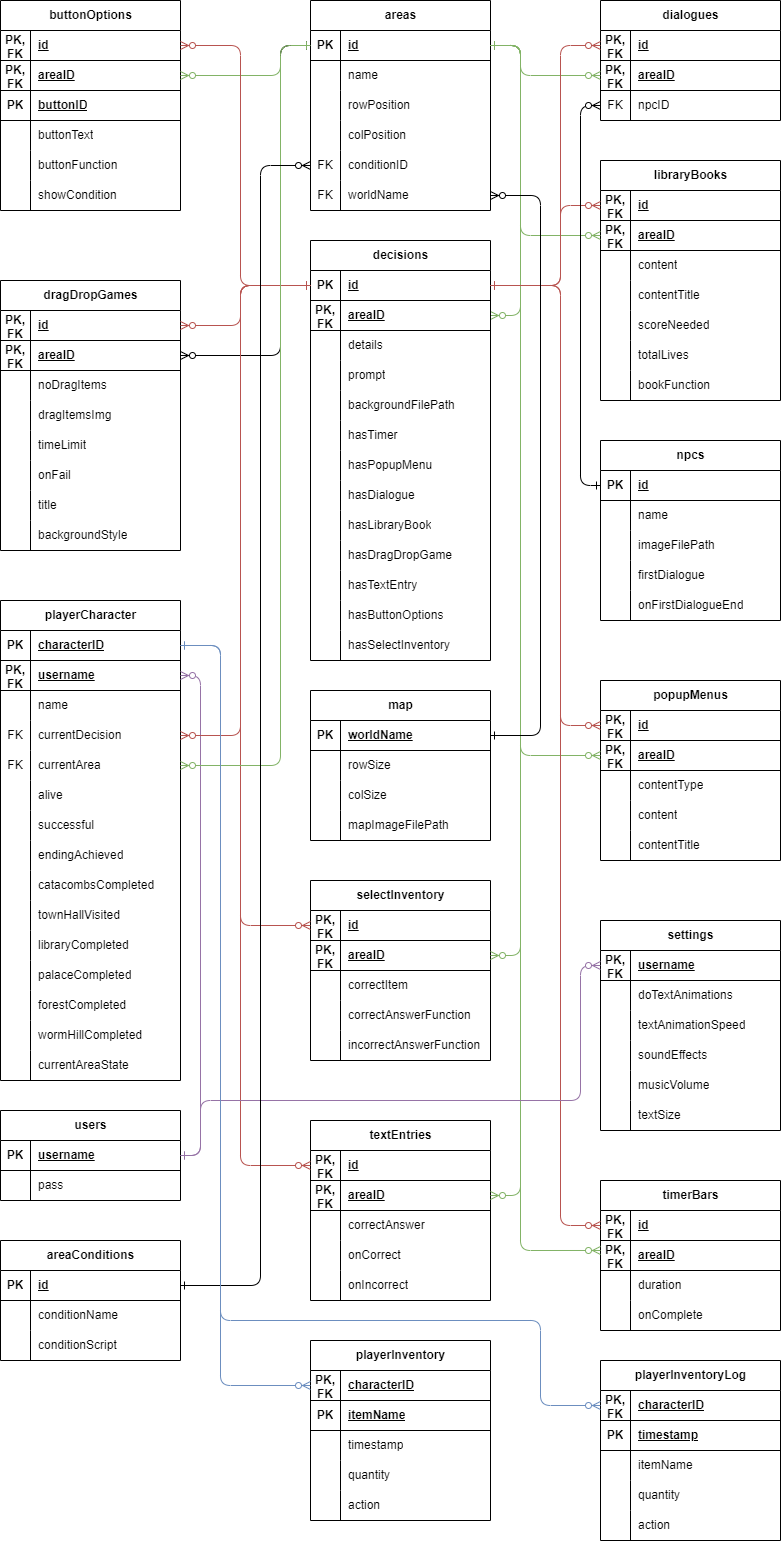
**Life After Death** -

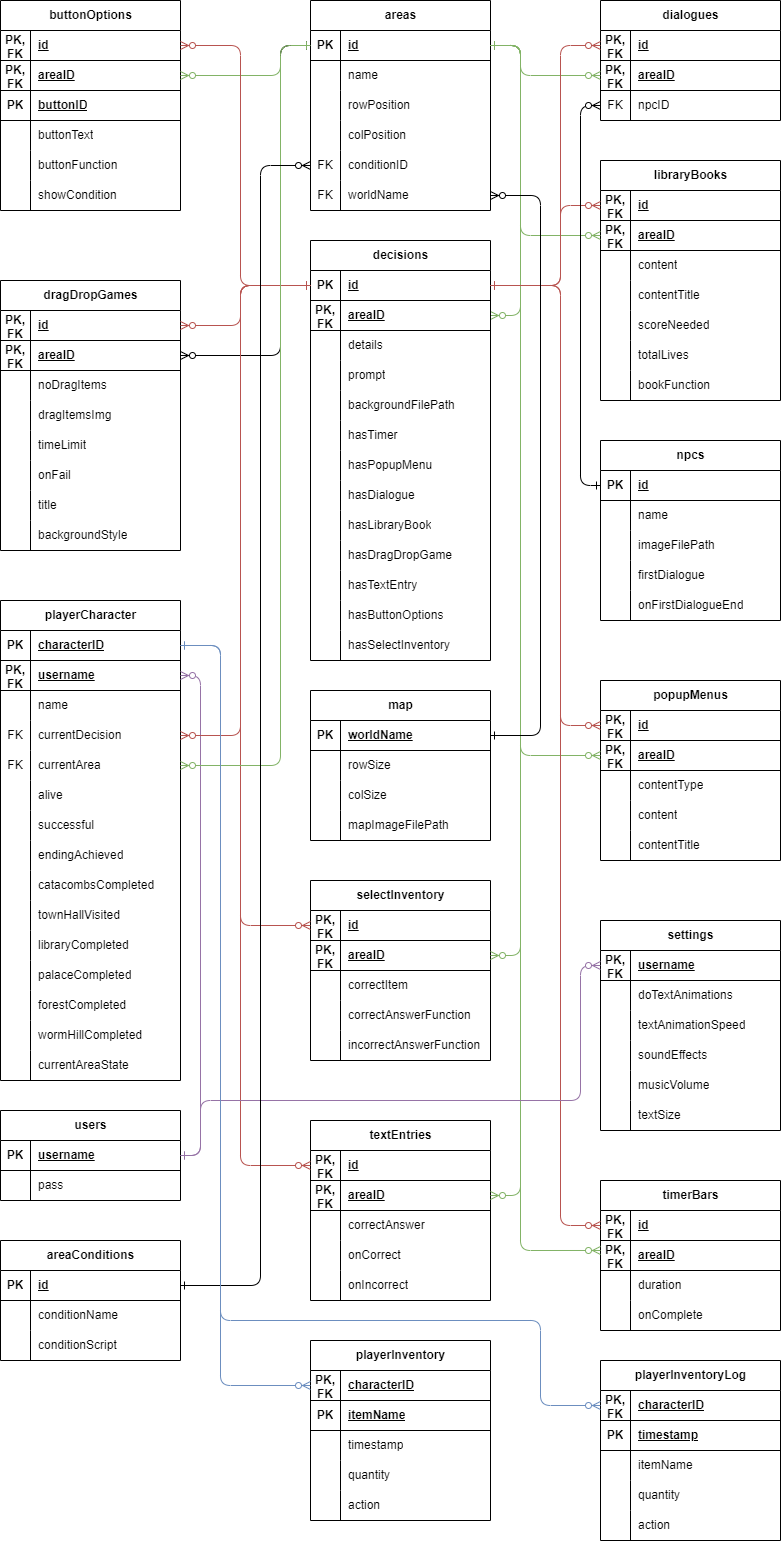
Database Design & Implementation Summary

# Group Members

* 40434771 - Daniel McClean
* 40446028 - Genna Gilpin
* 40405647 - Jack McCabe
* 40438408 - Jack Duke
* 40438752 - Joshua Brown
* 40444015 - Matthew Wallace

# Database Entity-Relationship Diagram





# Database Data Dictionary

| **DB NAME** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CSC1034\_CW\_54 | | | | | | | |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **areas** | | | | | | | |
| Contains information relating to the areas accessible within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| areaConditions | | | | conditionID | | | |
| map | | | | worldName | | | |
| Data Item | R | I | Data Type | Constraint | Size/  Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Unique identifier for each area | 3 |
| worldName | Y | N | char | FOREIGN KEY, NOT NULL | 20 | Name of the world map this area is contained within | “Potentia” |
| name | Y | N | char | NOT NULL | 20 | Name of the area | “Library” |
| rowPosition | Y | N | int | NOT NULL | 11 | Position horizontally on the map of this area | 5 |
| colPosition | Y | N | int | NOT NULL | 11 | Position vertically on the map of this area | 2 |
| conditionID | N | Y | int | FOREIGN KEY | 11 | References the condition which allows this area to be accessed | 2 |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **areaConditions** | | | | | | | |
| Contains information relating to when the areas in the game are accessible | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| - | | | | - | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY, NOT NULL,  AUTO INCREMENT | 11 | Unique identifier for each condition | 1 |
| conditionName | Y | N | varChar | NOT NULL | 50 | Text describing what the condition does | “Forest not complete” |
| conditionScript | Y | N | text | NOT NULL | - | Javascript method which checks if the necessary area has been completed for this area to be accessed | “(getState(“catacombsCompleted”) == false” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **buttonOptions** | | | | | | | |
| Contains information relating to the areas accessible within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| decisions | | | | id | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  FOREIGN KEY  NOT NULL | 11 | Unique identifier of each option | 1 |
| areaID | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Unique identifier for the area this button is in | 2 |
| buttonID | Y | Y | char | PRIMARY KEY,  NOT NULL | 100 | Identifier for each button | “option\_1” |
| buttonText | Y | N | char | NOT NULL | 50 | The text which will appear on the button | “Enter Library” |
| buttonFunction | Y | N | text | NOT NULL | 1000 | What code (arrow method) will be run upon clicking the button | “(() => {loadNextDecision(1,3)})” |
| showCondition | Y | N | text | NOT NULL | 1000 | Dictates whether or not this button displays on this screen.  If it is a “1”, it will always display”, otherwise, it can be replaced with a method to variable show the button | “1” or “(getState(“townHallVisited” == true)” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **decisions** | | | | | | | |
| Contains the information for the decisions to be made in the game which is used in building the page where the game takes place. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| areas | | | | areaID | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Unique identifier of this decision | 1 |
| areaID | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Identifier of the area this decision is contained in | 1 |
| details | N | N | text |  | 100 | Text which appears in the details box | “As you rise from the ground, you find yourself in…” |
| prompt | N | N | text |  | 100 | Text which appears at the top of the box containing the buttons | “Enter the Library?” |
| backgroundFilePath | Y | N | char | NOT NULL | 50 | File path for the image which will be displayed on the background of this decision | “images/backgrounds/library-outside.png” |
| hasTimer | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the timerBars table, if false, it will not build one. | 0 |
| hasPopupMenu | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the popupMenu table, if false, it will not build one. | 0 |
| hasDialogue | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the dialogues table, if false, it will not build one. | 0 |
| hasLibraryBook | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the libraryBooks table, if false, it will not build one. | 0 |
| hasDragDropGame | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the dragDropGames table, if false, it will not build one. | 0 |
| hasTextEntry | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the textEntries table, if false, it will not build one. | 0 |
| hasButtonOptions | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the buttonOptions table, if false, it will not build one. | 1 |
| hasSelectInventory | Y | N | tinyint | NOT NULL | 1 | If true, a timer will be built for this decision on screen, given that one exists in the selectionInventory table, if false, it will not build one. | 1 |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **dialogues** | | | | | | | |
| Connects the ids of the npcs and links them to a specific decision in the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| decisions | | | | id | | | |
| npcs | | | | npcID | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  FOREIGN KEY  NOT NULL | 11 | Unique identifier of this decision | 30 |
| areaID | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Identifier for each area, pulled from the areas table | 5 |
| npcID | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Identifier for each npc instance, pulled from the npcs table | 8 |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **dragDropGames** | | | | | | | |
| Contains the information for the dialogue boxes contained within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| decisions | | | | id | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  FOREIGN KEY  NOT NULL | 11 | Unique identifier of this decision | 16 |
| areaID | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Identifier for each area, pulled from the areas table | 2 |
| noDragItems | Y | N | int | NOT NULL | 11 | Number of items used in the drag drop game | 5 |
| dragItemsImg | Y | N | char | NOT NULL | 50 | The item which is displayed on each of the draggable objects | “images/items/fire.png” |
| timeLimit | Y | N | int | NOT NULL | 11 | How long the user has to complete the game | 20 |
| onFail | Y | N | text | NOT NULL | 1000 | What happens if the user fails the drag drop game | “(() => {goToDeathScreen(“As you attempt to set fire…”)})” |
| title | Y | N | char | NOT NULL | 50 | Title displayed above the game | “Set the rope on fire” |
| backgroundStyle | Y | N | text | NOT NULL |  | Styling for the background of the game | “background-color: rgba(200, 175, 100, 1);” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **libraryBooks** | | | | | | | |
| Contains the information for the system to build the library book games within the game | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| decisions | | | | id | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Unique identifier for the id of decision this libraryBook is contained within | 8 |
| areaID | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Identifier for each area, pulled from the areas table | 3 |
| content | Y | N | text | NOT NULL |  | What text is displayed within the book | “Peace is quiet, soft, and bright|  A gentle glow in darkest night.|  No war, no hate, just calm and free,|  A world of love for you and me.” |
| contentTitle | Y | N | char | NOT NULL | 50 | The title of the book | “The Gift of Peace” |
| scoreNeeded | Y | N | int | NOT NULL | 11 | Score needed for the player to progress | 10 |
| totalLives | Y | N | int | NOT NULL | 11 | Number of times the player can fail | 3 |
| bookFunction | Y | N | text | NOT NULL |  | What happens when the user succeeds | “(()=> {loadNextDecision(3, 9)})” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **map** | | | | | | | |
| Contains the information for the map, including the size and the image. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| - | | | | - | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| worldName | Y | Y | char | PRIMARY KEY,  NOT NULL | 20 | Name of the map | “Potentia” |
| rowSize | Y | N | int | NOT NULL | 11 | Number of rows on the map | 6 |
| colSize | Y | N | int | NOT NULL | 11 | Number of columns on the map | 7 |
| mapImagesFilePath | Y | N | char | NOT NULL | 50 | File path of the map image | “images/maps/potentia.png” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **npcs** | | | | | | | |
| Contains the information for the map, including the size and the image. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| - | | | | - | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  NOT NULL,  AUTO\_INCREMENT | 11 | Unique identifier for each npc interaction | 1 |
| name | Y | N | char | NOT NULL | 20 | Name which will appear in the name box when dialogue is shown | Ankou |
| imageFilePath | Y | N | char | NOT NULL | 50 | Contains the file path for the character image on this page | “images/npc/ankou.png” |
| firstDialogue | Y | N | text | NOT NULL | 1000 | DIalogue shown on the screen | “["Hrrrggghhh…",  "Death... Take... Thee... PlayerNameHere...",  "Ghhhuuuaaahh…"]” |
| onFirstDialogueEnd | Y | N | text | NOT NULL | 1000 | Javascript which is run | “(() => { goToDeathScreen("Calligraphous killed you"); })” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **playerCharacter** | | | | | | | |
| Contains the information for the map, including the size and the image. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| users | | | | username | | | |
| decisions | | | | currentDecision | | | |
| areas | | | | currentArea | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| characterID | Y | Y | int | PRIMARY KEY,  AUTO\_INCREMENT,  NOT NULL | 11 | Unique identifier for each character | 3 |
| username | Y | Y | varchar | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 50 | Username of the user which created this character | “JoshuaB” |
| name | Y | N | char | NOT NULL | 20 | String name of the character | “Hearth” |
| currentDecision | Y | Y | int | FOREIGN KEY,  NOT NULL | 11 | Contains the id of the current decision | 15 |
| currentArea | Y | Y | int | FOREIGN KEY,  NOT NULL | 11 | Contains the id of the current Area the user is in. | 3 |
| alive | Y | N | tinyint | NOT NULL | 1 | Dictates if the character is alive of dead | 1 or 0 |
| successful | N | N | tinyint |  | 1 | Dictates if the character is successful in beating the game. | 1 or 0 |
| endingAchieved | N | N | int |  | 11 | Contains which ending this character has achieved | 3 |
| catacombsCompleted | N | N | tinyint |  | 1 | Dictates if the user has cleared the catacombs or not | 1 or 0 |
| townHallVisited | N | N | tinyint |  | 1 | Dictates if the user has visited the town hall or not | 1 or 0 |
| libraryCompleted | N | N | tinyint |  | 1 | Dictates if the user has cleared the library or not | 1 or 0 |
| palaceCompleted | N | N | tinyint |  | 1 | Dictates if the user has cleared the palace or not | 1 or 0 |
| forestCompleted | N | N | tinyint |  | 1 | Dictates if the user has cleared the forest or not | 1 or 0 |
| wormHillCompleted | N | N | tinyint |  | 1 | Dictates if the user has cleared the worm hill or not | 1 or 0 |
| currentAreaState | N | N | json |  | 100 | Contains the state of the area, such as the information related to what has or has not been completed within the area the user is currently in. | “{"CHEST\_OPENED":true,"SARCOPHAGUS\_OPENED":true}” |
| currentPlayTime | Y | N | int | NOT NULL | 11 | The amount of time the user has spent within the game. Stored in milliseconds. | 87780 |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **playerInventory** | | | | | | | |
| Holds the items that are displayed in the inventory of the game, uniquely identified by the character id | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| playerCharacter | | | | characterID | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| characterID | Y | Y | int | PRIMARY KEY,  AUTO\_INCREMENT,  NOT NULL | 11 | Unique identifier for each character | 3 |
| itemName | Y | N | varchar | PRIMARY KEY,  NOT NULL | 50 | Contains the name of the item | OIL\_BOTTLE |
| quantity | Y | N | int | NOT NULL | 11 | Contains the quantity of the item in the character's inventory | 1 |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **playerInventoryLog** | | | | | | | |
| Holds a log of each item that gets added and removed into a player’s inventory | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| playerCharacter | | | | characterID | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| characterID | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  AUTO\_INCREMENT,  NOT NULL | 11 | Unique identifier for each character | 3 |
| timestamp | Y | Y | timestamp | PRIMARY KEY,  NOT NULL | 1000 | Contains the time this record was made into this table | “2025-03-31 12:18:21” |
| quantity | Y | N | int | NOT NULL | 11 | Contains quantity of item in this log instance | 1 |
| itemName | Y | N | varchar | NOT NULL | 50 | Contains item name in this log instance | RING OF STRENGTH |
| action | Y | N | varchar | NOT NULL | 50 | Whether the item has been added or removed | ADD |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **popupMenus** | | | | | | | |
| Contains the information for the system to build the popup menus within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| decisions | | | | id | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Unique identifier for the decision this is contained within | 10 |
| areaID | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Contains the id of the area this is related to | 2 |
| contentType | Y | N | char | NOT NULL | 20 | If it is text, display content as a paragraph, if it is list, display content as a bullet point list. | “text” or “list” |
| content | Y | N | text | NOT NULL | 1000 | Contains the content to be displayed on the page. | “Gold x100, Jar of Oil x1” |
| contentTitle | Y | N | char | NOT NULL | 50 | Contains the title to be displayed on the popup menu | “Old Tome” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **selectInventory** | | | | | | | |
| Contains the information for the system to build select inventory sections within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| id | | | | decisions | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Unique identifier for the decision this is contained within | 3 |
| areaID | Y | Y | int | NOT NULL | 11 |  | 5 |
| correctItem | Y | N | char | NOT NULL | 50 |  | “FEATHER” |
| correctAnswerFunction | Y | N | text | NOT NULL | 1000 |  | “(()=> {increaseStateVariable("RIDDLES\_CORRECT", 1); loadNextDecision(5, 4); changeState("RIDDLE1\_COMPLETE", true);})” |
| incorrectAnswerFunction | Y | N | text | NOT NULL | 1000 |  | “(()=> {changeState("RIDDLE1\_COMPLETE", true); loadNextDecision(5, 4);})” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **settings** | | | | | | | |
| Contains the information for the system to build select inventory sections within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| playerCharacter | | | | username | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| username | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Unique identifier for the decision this is contained within | 3 |
| doTextAnimations | Y | N | tinyint | NOT NULL | 1 | Dictates if the text animation will happen | 1 |
| textAnimationSpeed | Y | N | int | NOT NULL | 11 | Speed at which the text animation happens | 50 |
| textSize | Y | N | text | NOT NULL | 50 | Size of the text within the game (Small, medium or large) | “Small” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **textEntries** | | | | | | | |
| Contains the information for the system to build text entry sections within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| decisions | | | | id | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Unique identifier for the decision this appears on | 3 |
| areaID | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Unique identifier for the area which this appears in, references the areas table | 2 |
| correctAnswer | Y | N | char | NOT NULL | 50 | Text string of what the correct answer is for the text | fire |
| onCorrect | Y | N | text | NOT NULL | 1000 | What code is executed when the user inputs the correct answer | “(()=> {loadNextDecision(2, 26)})” |
| onIncorrect | Y | N | text | NOT NULL | 1000 | What code is executed when the user inputs an incorrect answer. | “(()=> {changePrompt("Incorrect. Try again.")})” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **timerBars** | | | | | | | |
| Contains the information for the system to build timer bars within the game. | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| decisions | | | | id | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| id | Y | Y | int | PRIMARY KEY,  NOT NULL | 11 | Unique identifier for the decision this timer bar appears on | 3 |
| areaID | Y | Y | int | PRIMARY KEY,  FOREIGN KEY,  NOT NULL | 11 | Unique identifier for the area which this appears in, references the areas table | 2 |
| duration | Y | N | int | NOT NULL | 11 | Gives the duration of the timer bar | 30 |
| onComplete | Y | N | text | NOT NULL | 1000 | When the timer bar completes it’s time, this code is executed. | “(()=>{goToDeathScreen("Indescision allowed Death to renew it&#39;s claim on You")})” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |
| **users** | | | | | | | |
| Contains the information for the users and their logins in the system | | | | | | | |
| **FK Table Name** | | | | **FK Field** | | | |
| - | | | | - | | | |
| Data Item | R | I | Data Type | Constraint | Size/Length | Description | Typical Data |
| username | Y | Y | varchar | PRIMARY KEY,  NOT NULL | 50 | Unique identifier for the decision this timer bar appears on | “ThisIsMyUsername” |
| pass | Y | N | varchar | NOT NULL | 255 | Contains the password of this user | “ThisIsMyPassword” |
| **- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -** | | | | | | | |

# Database Nested SQL

## Query Areas

Code

Found in queryAreas.js

| `SELECT  areas.id,  areas.name,  areas.rowPosition,  areas.colPosition,  areaConditions.conditionScript AS visitCondition FROM areas LEFT JOIN areaConditions ON areas.conditionID = areaConditions.id WHERE areas.${whereConditionAttribute} = '${value}';` |
| --- |

Associated HTML Pages

* map.html

Purpose

The purpose of the SQL is to select all of the areas inside the areas table as well as their associated coditionScript from the areaConditions table where the attribute defined in the **whereConditionAttribute** Variable is equal to the variable **value**. This query is used on the **map.html** page in order to fetch the location of each area on the map as well as to construct the button on the map, whether it is usable based on the evaluated value of the **visitCondition** and what to set the **currentArea** state variable to when the button is clicked to load this area.

## Query Decision

Code

Found in queryDecision.js

| `SELECT  decisions.id AS decisionID,  areas.name AS areaName,  decisions.details,  decisions.prompt,  decisions.backgroundFilePath,  decisions.hasTimer,  decisions.hasPopupMenu,  decisions.hasDialogue,  decisions.hasLibraryBook,  decisions.hasDragDropGame,  decisions.hasTextEntry,  decisions.hasButtonOptions,  decisions.hasSelectInventory,  -- Conditionally fetch Timer Bar Info  CASE WHEN decisions.hasTimer THEN timerBars.duration ELSE NULL END AS timerDuration,  CASE WHEN decisions.hasTimer THEN timerBars.onComplete ELSE NULL END AS timerOnComplete,  -- Conditionally fetch Popup Menu Info  CASE WHEN decisions.hasPopupMenu THEN popupMenus.contentType ELSE NULL END AS popupContentType,  CASE WHEN decisions.hasPopupMenu THEN popupMenus.content ELSE NULL END AS popupContent,  CASE WHEN decisions.hasPopupMenu THEN popupMenus.contentTitle ELSE NULL END AS popupContentTitle,  -- Conditionally fetch Text Entry Info  CASE WHEN decisions.hasTextEntry THEN textEntries.correctAnswer ELSE NULL END AS textEntryCorrectAnswer,  CASE WHEN decisions.hasTextEntry THEN textEntries.onCorrect ELSE NULL END AS textEntryOnCorrect,  CASE WHEN decisions.hasTextEntry THEN textEntries.onIncorrect ELSE NULL END AS textEntryOnIncorrect,  -- Conditionally fetch Library Book Info  CASE WHEN decisions.hasLibraryBook THEN libraryBooks.content ELSE NULL END AS bookContent,  CASE WHEN decisions.hasLibraryBook THEN libraryBooks.contentTitle ELSE NULL END AS bookTitle,  CASE WHEN decisions.hasLibraryBook THEN libraryBooks.scoreNeeded ELSE NULL END AS bookScoreNeeded,  CASE WHEN decisions.hasLibraryBook THEN libraryBooks.totalLives ELSE NULL END AS bookTotalLives,  CASE WHEN decisions.hasLibraryBook THEN libraryBooks.bookFunction ELSE NULL END AS bookFunction,  -- Conditionally fetch Drag Drop Game Info  CASE WHEN decisions.hasDragDropGame THEN dragDropGames.noDragItems ELSE NULL END AS noDragItems,  CASE WHEN decisions.hasDragDropGame THEN dragDropGames.dragItemsImg ELSE NULL END AS dragItemsImg,  CASE WHEN decisions.hasDragDropGame THEN dragDropGames.timeLimit ELSE NULL END AS dragDropTimeLimit,  CASE WHEN decisions.hasDragDropGame THEN dragDropGames.onFail ELSE NULL END AS dragDropOnFail,  CASE WHEN decisions.hasDragDropGame THEN dragDropGames.title ELSE NULL END AS dragDropTitle,  CASE WHEN decisions.hasDragDropGame THEN dragDropGames.backgroundStyle ELSE NULL END AS dragDropBackground,  -- Conditionally fetch Dialogue Info  CASE WHEN decisions.hasDialogue THEN dialogues.npcID ELSE NULL END AS npcID,  -- Conditionally fetch Button Options  CASE WHEN decisions.hasButtonOptions THEN JSON\_ARRAYAGG(buttonOptions.buttonID ORDER BY buttonOptions.buttonID) ELSE NULL END AS buttonIDs,  CASE WHEN decisions.hasButtonOptions THEN JSON\_ARRAYAGG(buttonOptions.buttonText ORDER BY buttonOptions.buttonID) ELSE NULL END AS buttonTexts,  CASE WHEN decisions.hasButtonOptions THEN JSON\_ARRAYAGG(buttonOptions.buttonFunction ORDER BY buttonOptions.buttonID) ELSE NULL END AS buttonFunctions,  CASE WHEN decisions.hasButtonOptions THEN JSON\_ARRAYAGG(buttonOptions.showCondition ORDER BY buttonOptions.buttonID) ELSE NULL END AS buttonConditions,  -- Conditionally fetch Select Inventory Info  CASE WHEN decisions.hasSelectInventory THEN selectInventory.correctItem ELSE NULL END AS selectInventoryCorrectItem,  CASE WHEN decisions.hasSelectInventory THEN selectInventory.correctAnswerFunction ELSE NULL END AS selectInventoryCorrectAnswerFunction,  CASE WHEN decisions.hasSelectInventory THEN selectInventory.incorrectAnswerFunction ELSE NULL END AS selectInventoryIncorrectAnswerFunction  FROM decisions JOIN areas ON decisions.areaID = areas.id  LEFT JOIN timerBars ON decisions.hasTimer = TRUE AND decisions.id = timerBars.id AND decisions.areaID = timerBars.areaID LEFT JOIN popupMenus ON decisions.hasPopupMenu = TRUE AND decisions.id = popupMenus.id AND decisions.areaID = popupMenus.areaID LEFT JOIN textEntries ON decisions.hasTextEntry = TRUE AND decisions.id = textEntries.id AND decisions.areaID = textEntries.areaID LEFT JOIN libraryBooks ON decisions.hasLibraryBook = TRUE AND decisions.id = libraryBooks.id AND decisions.areaID = libraryBooks.areaID LEFT JOIN dragDropGames ON decisions.hasDragDropGame = TRUE AND decisions.id = dragDropGames.id AND decisions.areaID = dragDropGames.areaID LEFT JOIN buttonOptions ON decisions.hasButtonOptions = TRUE AND decisions.id = buttonOptions.id AND decisions.areaID = buttonOptions.areaID LEFT JOIN dialogues ON decisions.hasDialogue = TRUE AND decisions.id = dialogues.id AND decisions.areaID = dialogues.areaID LEFT JOIN selectInventory ON decisions.hasSelectInventory = TRUE AND decisions.id = selectInventory.id AND decisions.areaID = selectInventory.areaID  WHERE decisions.id = ${decisionID} AND areas.id = ${areaID}  GROUP BY decisions.id, areas.id;` |
| --- |

Associated HTML Pages

* dungeon.html

Purpose

This query is used to fetch all of the data associated with a decision and returns it so that it can be used by the other JavaScript functions connected to dungeon.html inorder to render the decision on screen. If the property in decision table is set as **FALSE** for one of our web components then the properties of that component are set to **NULL** this is to avoid errors so we can guarantee that if the hasX property is **TRUE** then the other properties for the component must all be valid and it can be constructed by the JavaScript code. Data is fetched based on a given **decisionID** and **areaID**

## Query Map

Code

Found in queryMap.js

| `SELECT \*  FROM map  WHERE map.worldName = '${mapID}';` |
| --- |

Associated HTML Pages

* map.html

Purpose

This selects all the data for the map with a specified **worldName**. This is used to build the map on screen on the map.html page.

## Query NPC

Code

Found in queryNPC.js

| `SELECT  npcs.name,  npcs.imageFilePath,  npcs.firstDialogue,  npcs.onFirstDialogueEnd  FROM npcs  WHERE npcs.id = ${npcID}  GROUP BY npcs.id, npcs.name, npcs.imageFilePath, npcs.firstDialogue;` |
| --- |

Associated HTML Pages

* dungeon.html

Purpose

This query is used to fetch the required properties of an NPC in order to populate the dialogue component on a decision page in dungeon.html

## Query Settings

Code

Found in querySettings.js

| `SELECT \* FROM settings WHERE username = '${username}'` |
| --- |

Associated HTML Pages

* All HTML pages

Purpose

This query is used to load all of the settings stored for a particular user.

## Update Character Inventory - Add Item

Code

Found in inventoryFunctions.js -> function updateDatabaseInventory(inventory)

| `INSERT INTO playerInventory (characterID, itemName, quantity)  VALUES (${characterID}, '${itemName}', ${quantity})  ON DUPLICATE KEY UPDATE quantity = ${quantity}; ` |
| --- |

Associated HTML Pages

* dungeons.html

Purpose

This query is used to add new items to the playerInventory table and to update the quantity of existing items in the playerInventory table for a specific characterID. This function is called every time the loadNextDecision function is called and facilitates the auto saving functionality of the game.

## Update Character Inventory - Delete Item

Code

Found in inventoryFunctions.js -> function updateDatabaseInventory(inventory)

| `DELETE FROM playerInventory  WHERE characterID = ${characterID}  AND itemName NOT IN (${Object.keys(inventory).map((item) => `'${item}'`).join(',')}); ` |
| --- |

Associated HTML Pages

* dungeons.html

Purpose

This query is used to remove items from the playerInventory for a specific characterID if they have been removed from the inventory object. This is called at the same points as the above query.

## Update Character Inventory - Empty Inventory

Code

Found in inventoryFunctions.js -> function updateDatabaseInventory(inventory)

| `DELETE FROM playerInventory WHERE characterID = ${characterID};` |
| --- |

Associated HTML Pages

* dungeons.html

Purpose

This query is used to handle an edge case if the inventory object is completely empty to remove all inventory items in playerInventory for a characterID.

## Insert Into playerInventoryLog

Code

Found in inventoryFunctions.js -> function updateDatabaseInventory(inventory)

| `INSERT INTO playerInventoryLog (characterID, timestamp, itemName, quantity, action)  VALUES ${inventoryLog.map((record) => { return `(${record.characterID}, FROM\_UNIXTIME(${record.timestamp}), '${record.item}', ${record.quantity}, '${record.action}')` }).join(',')}; ` |
| --- |

Associated HTML Pages

* dungeons.html

Purpose

This query is used to update the playerInventoryLog with all the actions that have occurred to the playerInventory since the last decision. This records the characterID and the timestamp and allows for a complete history of the inventory to be constructed.

## Update State

Code

Found in stateFunctions.js -> function updateStateDatabase(gameState)

| `UPDATE playerCharacter  SET  ${attributeNames  .map(attribute => {  const value = gameState.globalState[attribute];  // If value is null or undefined, set it to NULL in the query  if (value === null || value === undefined) {  return `${attribute} = NULL`;  }  return `${attribute} = '${value}'`;  })  .join(", ")}  ,currentAreaState = '${JSON.stringify(gameState.localState)}'  WHERE username = '${gameState.globalState.username}' AND characterID = '${gameState.globalState.characterID}';` |
| --- |

Associated HTML Pages

* dungeons.html

Purpose

This query updates the state attributes of the playerCharacter table for a specific characterID. The properties in the gameState.globalState object are all individual properties as they are of relevance multiple times in different areas of the application while gameState.localState is mapped as JSON to the currentAreaState attribute as they are temporary state variables that only have meaning from within the current area. This is called every time the loadNextDecision function is called to facilitate autosaving.

## Query playerCharacter

Code

Found in loadLastGame.js -> function getAllCharacters()

| `SELECT \* FROM playerCharacter WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* loadGame.html

Purpose

This gets all the attributes for every character belonging to a user

## Query playerInventory

Code

Found in loadLastGame.js -> function getAllCharacters()

| `SELECT \* FROM playerInventory WHERE characterID = '${character.characterID}'` |
| --- |

Associated HTML Pages

* loadGame.html

Purpose

This gets all the inventory items and their quantity for a character

## Insert New playerCharacter

Code

Found in newCharaceter.js

| `INSERT INTO playerCharacter (  username,  name,  currentDecision,  currentArea,  alive,  successful,  endingAchieved,  catacombsCompleted,  townHallVisited,  libraryCompleted,  palaceCompleted,  forestCompleted,  wormHillCompleted,  currentAreaState,  currentPlayTime)  VALUES ('${user}', '${characterName}', 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, '{}', 0)` |
| --- |

Associated HTML Pages

* newGame.html

Purpose

This creates an new playerCharacter for a user with a name given by the user

## Query User

Code

Found in login.js

| `SELECT username, pass FROM users WHERE username = '${username}' AND pass = '${password}' ` |
| --- |

Associated HTML Pages

* login.html

Purpose

This checks that there is a user in the database with a username and password that matches those given

## Insert User

Code

Found in login.js

| `INSERT INTO users (username, pass) VALUES ('${username}', '${password}')` |
| --- |

Associated HTML Pages

* login.html

Purpose

This creates a new user with a username and password that matches those given by the user.

## Update User

Code

Found in edit.js

| `UPDATE users SET username = '${newUsername}', pass = '${password}' WHERE username = '${oldUsername}'; ` |
| --- |

Associated HTML Pages

* editAccount.html

Purpose

This updates a user with a new username and password that matches those given by the user.

## Delete User

Code

Found in delete.js

| `DELETE FROM users WHERE username = '${username}' AND pass = '${password}' |
| --- |

Associated HTML Pages

* deleteAccount.html

Purpose

This deletes a user with a username and password that matches those given by the user.

## Insert Settings

Code

Found in login.js

| `INSERT INTO settings (username, doTextAnimations, textAnimationSpeed, textSize) VALUES ('${username}', TRUE, 30, 'Medium') ' |
| --- |

Associated HTML Pages

* login.html

Purpose

This inserts the default settings for a user into the settings table

## Update Settings

Code

Found in settings.js

| `UPDATE settings SET doTextAnimations = ${settings.doTextAnimations ? 1 : 0}, textAnimationSpeed = ${settings.textAnimationSpeed}, textSize = '${settings.textSize}' WHERE username = '${username}'` |
| --- |

Associated HTML Pages

* settings.html

Purpose

This updates the settings for a user in the settings table with the new values they have selected.

## Users Who Completed Catacombs

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_completed\_catacombs  FROM playerCharacter  WHERE catacombsCompleted = TRUE;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has completed the catacombs. Used for getting percentage of users who have this achievement

## This User Completed Catacombs

Code

Found in achievements.js

| `SELECT catacombsCompleted FROM playerCharacter WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user completed the catacombs? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Visited The Town Hall

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_town\_hall\_visited FROM playerCharacter WHERE townHallVisited = TRUE;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has visited the Town Hall. Used for getting percentage of users who have this achievement

## This User Visited The Town Hall

Code

Found in achievements.js

| `SELECT townHallVisited FROM playerCharacter WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user visited the Town Hall? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Completed Library

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_library\_completed FROM playerCharacter WHERE libraryCompleted = TRUE;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has completed the Library. Used for getting percentage of users who have this achievement

## This User Completed Library

Code

Found in achievements.js

| `SELECT libraryCompleted FROM playerCharacter WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user completed the Library? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## 

## Users Who Completed Palace

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_palace\_completed FROM playerCharacter WHERE palaceCompleted = TRUE;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has completed the Palace. Used for getting percentage of users who have this achievement

## This User Completed Palace

Code

Found in achievements.js

| `SELECT palaceCompleted FROM playerCharacter WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user completed the Palace? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## 

## Users Who Completed Forbidden Forest

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_forest\_completed FROM playerCharacter WHERE forestCompleted = TRUE;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has completed the Forbidden Forest. Used for getting percentage of users who have this achievement

## This User Completed Forbidden Forest

Code

Found in achievements.js

| `SELECT forestCompleted FROM playerCharacter WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user completed the Forbidden Forest? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## 

## Users Who Completed Worm Hill

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_worm\_hill\_completed  FROM playerCharacter  WHERE wormHillCompleted = TRUE;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has completed the Worm Hill. Used for getting percentage of users who have this achievement

## This User Completed Worm Hill

Code

Found in achievements.js

| `SELECT wormHillCompleted  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user completed the Worm Hill? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Completed The Game

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_game\_completed  FROM playerCharacter  WHERE successful = TRUE;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has completed the Game. Used for getting percentage of users who have this achievement

## This User Completed The Game

Code

Found in achievements.js

| `SELECT successful  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user completed the Game? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## 

## Users Who Achieved Midas Reborn

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_midas\_reborn  FROM playerCharacter  WHERE endingAchieved = 1;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved Midas Reborn, ending 1. Used for getting percentage of users who have this achievement

## This User Achieved Midas Reborn

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved Midas Reborn, ending 1? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Achieved Black Hole

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_midas\_reborn  FROM playerCharacter  WHERE endingAchieved = 1;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved Black Hole, ending 3. Used for getting percentage of users who have this achievement

## This User Achieved Black Hole

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved Black Hole, ending 3? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Achieved Hero’s Death

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_hero\_death  FROM playerCharacter  WHERE endingAchieved = 2;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved Hero’s Death, ending 2. Used for getting percentage of users who have this achievement

## This User Achieved Black Hole

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved Hero’s Death, ending 2? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Achieved The True Ending

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_true\_ending  FROM playerCharacter  WHERE endingAchieved = 4;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved The True Ending, ending 4. Used for getting percentage of users who have this achievement

## This User Achieved Black Hole

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved The True Ending, ending 4? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## 

## Users Who Achieved Live By The Sword, Die By The Sword

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_die\_by\_sword  FROM playerCharacter  WHERE endingAchieved = 6;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved Live By The Sword, Die By The Sword, ending 6. Used for getting percentage of users who have this achievement

## This User Achieved Live By The Sword, Die By The Sword

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved Live By The Sword, Die By The Sword, ending 6? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Achieved Buried Secrets

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_buried\_secrets  FROM playerCharacter  WHERE endingAchieved = 7;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved Buried Secrets, ending 7. Used for getting percentage of users who have this achievement

## This User Achieved Buried Secrets

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved Buried Secrets, ending 7? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Achieved Should've Died A Hero

Code

Found in achievements.js

| `SELECT COUNT(DISTINCT username) AS users\_with\_died\_hero  FROM playerCharacter  WHERE endingAchieved = 5;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved Should've Died A Hero, ending 5. Used for getting percentage of users who have this achievement

## This User Achieved Should've Died A Hero

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved Should've Died A Hero, ending 5? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Users Who Achieved All Endings

Code

Found in achievements.js

| `SELECT COUNT(username) AS users\_with\_all\_endings  FROM (  SELECT username  FROM playerCharacter  WHERE endingAchieved IN (1,2,3,4,5,6,7) -- Only consider valid endings  GROUP BY username  HAVING COUNT(DISTINCT endingAchieved) = 7 -- Ensure they have all 7 endings  ) AS subquery;` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Gets the number of users who have a character that has Achieved All Endings. Used for getting percentage of users who have this achievement

## This User Achieved All Endings

Code

Found in achievements.js

| `SELECT endingAchieved  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}'` |
| --- |

Associated HTML Pages

* achievements.html

Purpose

Has the current user Achieved All Endings? Used to decide whether to show the achievement as complete or not. The boolean is evaluated in JavaScript.

## Query Character Name

Code

Found in gameSummary.js

| `SELECT name FROM playerCharacter WHERE characterID = ${sessionStorage.getItem('characterID')};` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets the name of the current character for the game summary

## Query Character Play Time

Code

Found in gameSummary.js

| `SELECT currentPlayTime FROM playerCharacter WHERE characterID = ${sessionStorage.getItem('characterID')};` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets the currentPlayTime of the current character for the game summary

## Query Character Current State

Code

Found in gameSummary.js

| `SELECT alive, successful FROM playerCharacter WHERE characterID = ${sessionStorage.getItem('characterID')};` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets whether this character has won, lost or is currently in progress for the Game Summary

## Query Character Current Progress

Code

Found in gameSummary.js

| `SELECT catacombsCompleted, townHallVisited, forestCompleted, libraryCompleted, palaceCompleted, wormHillCompleted, successful FROM playerCharacter WHERE characterID = ${sessionStorage.getItem('characterID')};` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets global state variables so percentage completion of the game can be calculated

## Query Character Current Area

Code

Found in gameSummary.js

| `SELECT areas.name, playerCharacter.successful FROM playerCharacter JOIN areas ON playerCharacter.currentArea = areas.id WHERE playerCharacter.characterID = '${sessionStorage.getItem('characterID')}';` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets the name of the current area and whether the character was successful or not to display on the game summary. If successful, the area displayed as - - -.

## Query Character Ending Achieved

Code

Found in gameSummary.js

| `SELECT endingAchieved, successful, alive FROM playerCharacter WHERE playerCharacter.characterID = '${sessionStorage.getItem('characterID')}';` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets the ending achieved as well as if they are successful and if they are alive. Not alive and not successful shows ‘Death & Defeat’, alive shows - - -.

## Query Character Currently Held Items

Code

Found in gameSummary.js

| `SELECT DISTINCT itemName FROM playerInventory WHERE characterID = ${sessionStorage.getItem('characterID')};` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets all the items currently in playerInventory belonging to this character

## Query Character All Collected Items

Code

Found in gameSummary.js

| `SELECT DISTINCT itemName FROM playerInventoryLog WHERE characterID = ${sessionStorage.getItem('characterID')} AND action='ADD';` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets all items ever added to the inventory of this character

## Query Character All Used Items

Code

Found in gameSummary.js

| `SELECT DISTINCT itemName FROM playerInventoryLog WHERE characterID = ${sessionStorage.getItem('characterID')} AND action=REMOVE;` |
| --- |

Associated HTML Pages

* gameSummary.html

Purpose

Gets all items ever removed from the inventory of this character

## Query Total Users

Code

Found in globalAnalytics.js

| 'SELECT COUNT(\*) FROM users' |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Gets the number of users in the database

## Query Average Characters Per User

Code

Found in globalAnalytics.js

| `SELECT AVG(COALESCE(character\_count, 0)) AS avg\_characters\_per\_user FROM (  SELECT u.username, COUNT(pc.characterID) AS character\_count  FROM users u  LEFT JOIN playerCharacter pc ON u.username = pc.username  GROUP BY u.username ) subquery;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Gets the average number of characters per user

## Query Average Deaths Per User

Code

Found in globalAnalytics.js

| `SELECT AVG(COALESCE(character\_count, 0)) AS avg\_deaths\_per\_user FROM (  SELECT u.username, COUNT(pc.characterID) AS character\_count  FROM users u  LEFT JOIN playerCharacter pc ON u.username = pc.username AND pc.alive = FALSE AND pc.successful = FALSE  GROUP BY u.username ) subquery;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Gets the average number of deaths per user

## Query Average Completions Per User

Code

Found in globalAnalytics.js

| `SELECT AVG(COALESCE(character\_count, 0)) AS avg\_deaths\_per\_user FROM (  SELECT u.username, COUNT(pc.characterID) AS character\_count  FROM users u  LEFT JOIN playerCharacter pc ON u.username = pc.username AND pc.successful = TRUE  GROUP BY u.username ) subquery;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Gets the average number of successful characters per user

## Query Average Play Time

Code

Found in globalAnalytics.js

| `SELECT AVG(COALESCE(user\_total\_playtime, 0)) AS avg\_playtime\_per\_user FROM (  SELECT users.username, SUM(playerCharacter.currentPlayTime) AS user\_total\_playtime  FROM users  LEFT JOIN playerCharacter ON users.username = playerCharacter.username  GROUP BY users.username ) AS user\_playtimes;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Gets the average playtime across all users

## Query Average Play Time For Successful Character

Code

Found in globalAnalytics.js

| `SELECT AVG(COALESCE(success\_total\_playtime, 0)) AS avg\_playtime\_per\_success FROM (  SELECT SUM(currentPlayTime) AS success\_total\_playtime  FROM playerCharacter  WHERE successful = TRUE  GROUP BY characterID ) AS successful\_playtimes;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Gets the average playtime for a successful character across all users

## Query Average Play Time For Death

Code

Found in globalAnalytics.js

| `SELECT AVG(COALESCE(success\_total\_playtime, 0)) AS avg\_playtime\_per\_success FROM (  SELECT SUM(currentPlayTime) AS success\_total\_playtime  FROM playerCharacter  WHERE successful = FALSE AND alive = FALSE  GROUP BY characterID ) AS successful\_playtimes;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Gets the average playtime to die across all users

## Query Most Collected Item

Code

Found in globalAnalytics.js

| `SELECT itemName, COUNT(\*) AS occurrences  FROM playerInventoryLog  WHERE action = 'ADD'  GROUP BY itemName  ORDER BY occurrences DESC  LIMIT 1;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Selects the item that was added the most to all characters inventory

## Query Most Used Item

Code

Found in globalAnalytics.js

| `SELECT itemName, COUNT(\*) AS occurrences  FROM playerInventoryLog  WHERE action = 'REMOVE'  GROUP BY itemName  ORDER BY occurrences DESC  LIMIT 1;` |
| --- |

Associated HTML Pages

* globalAnalytics.html

Purpose

Selects the item that was removed the most to all characters inventory

## 

## 

## Query Number of Characters for a User

Code

Found in globalAnalytics.js

| `SELECT COUNT(\*)  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}';  ` |
| --- |

Associated HTML Pages

* userAnalytics.html

Purpose

Gets the number of characters owned by this user

## Query Number of Successful Characters for a User

Code

Found in userAnalytics.js

| `SELECT COUNT(\*)  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}' AND successful = TRUE;  ` |
| --- |

Associated HTML Pages

* userAnalytics.html

Purpose

Gets the number of successful characters owned by this user

## Query Number of Dead Characters for a User

Code

Found in userAnalytics.js

| `SELECT COUNT(\*)  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}' AND successful = FALSE AND alive = FALSE;  ` |
| --- |

Associated HTML Pages

* userAnalytics.html

Purpose

Gets the number of dead characters owned by this user

## Query Number of In Progress Characters for a User

Code

Found in userAnalytics.js

| `SELECT COUNT(\*)  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}' alive = TRUE;  ` |
| --- |

Associated HTML Pages

* userAnalytics.html

Purpose

Gets the number of in progress characters owned by this user

## 

## 

## Query All Collected Items

Code

Found in userAnalytics.js

| `SELECT DISTINCT pil.itemName  FROM playerInventoryLog pil  JOIN playerCharacter pc ON pil.characterID = pc.characterID  WHERE pc.username = '${sessionStorage.getItem("username")}';` |
| --- |

Associated HTML Pages

* userAnalytics.html

Purpose

Gets the all the items that have been added to the inventory of any character belonging to this user

## Query All Collected Items

Code

Found in userAnalytics.js

| `SELECT DISTINCT pil.itemName  FROM playerInventoryLog pil  JOIN playerCharacter pc ON pil.characterID = pc.characterID  WHERE pc.username = '${sessionStorage.getItem("username")}' AND pil.action = ‘REMOVE’;` |
| --- |

Associated HTML Pages

* userAnalytics.html

Purpose

Gets the all the items that have been removed to the inventory of any character belonging to this user

## Query Fastest Completion Time

Code

Found in userAnalytics.js

| `SELECT MIN(currentPlayTime) AS min\_play\_time  FROM playerCharacter  WHERE username = '${sessionStorage.getItem("username")}' AND successful = TRUE;` |
| --- |

Associated HTML Pages

* userAnalytics.html

Purpose

Gets the fastest play time for a successful character