**University Tycoon World Championships 2025**

*University Tycoon* is a popular boardgame amongst the academic community and alumni of many universities. The basic premise of the game is that players move around the board, with the aim of collecting as many credits as possible. They do this by buying university buildings (with their credits), and charging tuition fees to other students who visit said buildings. Throughout the game, players can receive extra credits from extra curricular activites, they can be forced to transfer credits to another player, and they can even be placed on suspension!

Given the game's popularity, universities often sell their own version of the game through their merchandise stores. Another side effect of the game's popularity is that an annual world championship is held, where competing institutions enter their best player to see who can be crowned winner.

This academic year, Manchester will play host to the world championships, and you have been employed by the organisers to help create an important part of the games' information systems.

**Introduction**

Your task is to model the gameplay of *University Tycoon* using a relational database and SQL queries. Your database and the queries must be compatible with SQLite.

There is no single RIGHT way to do this assignment; there is no perfect solution. If you gave this exercise to 10 database professionals you would get 10 answers that differ in their detail, and I suspect that at least 5 answers would differ in their overall approach; that’s to be expected.

Requirements analysis and database design are to some extent formulaic, yet they are also arts, relying on experience, intuition and creativity. This assignment is as much about the problem-solving process itself, as it is about actually solving the problem. So the assessment will take your creativity into account. Your approach must satisfy the requirements, but exactly how you go about it, is up to you. Be creative and have fun!

Remember that there is more to SQL than we have looked at in the course, so you are encouraged to research and use SQL commands that we have not covered, although this is not an assessed requirement. Please note: SQLite does not support procedures, and this needs to be taken into account when creating your database.

**Scenario**

The list below gives the data definition and manipulation requirements for the game. This is deliberately not written with super-precise wording. When working on real projects, initial drafts of requirements are rarely complete and unambiguous. If you find any ambiguities, omissions, or imprecision here, make your own decision about what to do, **and mention this decision in your video**.

1. There are up to six players.
2. There are six tokens the players can choose from. Each token has a unique name (Mortarboard, Book, Certificate, Gown, Laptop, Pen).
3. A player must choose one and only one token.
4. Each space on the board is called a location. A location is one of: a Corner space, a Hearing space, a RAG space, or a Building space.
5. Corners, Hearings, and RAGs can be grouped together as “specials”. Every special has a unique id, its name (e.g., “RAG 1”) and a textual description of what it does.
6. You should ensure that you store data about players, buildings, and specials.
7. A building has either one owner or no owner.
8. A building has a unique name, and a tuition fee. Buildings can be purchased from the university by a player, in the case where the player has landed on the building and it does not already have an owner. The building can be purchased from the University for twice the amount shown as the tuition fee. The tuition fee is the amount that a player must pay to the owner (another player) when they land on that building.
9. Each player has a unique id, name, their chosen token, credit balance, and their current location. All this data should have suitable default values and constraints.
10. A special can be used by many players. A player has at most one special at any time in the gameplay.
11. There must be an audit trail of the gameplay. For each turn taken by a player, the audit trail should store the player’s id, location landed on, current credit balance, and number of the game round.

**Rules Of The Game**

* R0: Play moves in a clockwise direction. The player rolls a 6-sided fair dice once; they move their token the number of spaces clockwise, as shown on the dice.
* R1: If a player lands on a building without an owner, they must buy it from the university (at a price that is twice the amount of the tuition fee).
* R2: If player P lands on a building owned by player Q, then P pays Q the tuititon fee associated with that building. If Q owns all the buildings of a particular colour, P pays double the tuition fee.
* R3: If a player is "suspended", then they are in the location "Suspension"; they must roll a 6 to get out. They immediately roll again.
* R4: If a player lands on or passes "Welcome Week", they receive 100 credits.
* R5: If a player rolls a 6, they move 6 squares; whatever location they land on has no effect. They then get another roll immediately.
* R6: If a player lands on “You're Suspended!”, they move to the "Suspension" location, without passing Welcome Week or collecting the complimentary credits.
* R7: If a player lands on a "RAG" or "Hearing" location, the action described by the card description happens.
* R8: If a player lands on "Suspension" (and they are not suspended), then they are classed as "Visiting", and no action is taken (there is a visiting space for this purpose).

**Initial State of The Game**

You can see the game board, player token assignment, credit balances, and Special cards below. This gives all required information about the game.

You can tell a players location, by their token being on the ***inside edge*** of a location square on the board.

You can tell that a player owns a building by their token appearing in the ***top-right***of the location square on the board.

Each building has a colour associated with it, shown by the colour of the building's square. To aid accessability, there is also a high-contrast shape (triangle, square, circle, diamond, cross, ring) present in the top-left of each building square, which may act as a proxy for the colour.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Task 1**  
Create an entity relationship diagram that models this game as a relational database. The database must also contain an audit log, which records each move of the players. It should record the game round (shown in Task 4), players name, location, and current balance of credits. This diagram must use crows-foot notation. You should ensure that as the design of your database changes during implementation, that it is reflected in the ER diagram.

**Task 2**

Create a schema from your ER Diagram, and then implement your design in SQL. Your database must remain a relational database, and must be compatible with SQLite3. Your database must be populated with data that accurately represents the initial state of the game.

**Task 3**

Create an SQL view that will act as a datasource to external clients. Given that it is accessed by external clients, it MUST follow the format described here:

* is called "leaderboard"
* Column 1, called "name" gives the players name, exactly as shown in the brief
* Column 2, called "location" gives the current location, given in snake case (LINK)
  + eg: Ali G becomes "ali\_g", Welcome Week becomes "welcome\_week", IT becomes "it", Your'e Suspended becomes "youre\_suspended", Visitor becomes "visitor".
* Column 3, called "credits" gives the players credit balance as an integer value.
  + eg: If a player has 200 credits, then this will be displayed as 200.
* Column 4, called "buildings" displays a list of the buildings owned by the player (in snake case, separated by a comma and a following space), in order as they appear on the board, going clockwise and starting from Welcome Week.
  + eg: If the player owns Kilburn, IT, Sugden, and AMBS, then this will be displayed as "kilburn, it, ambs, sugden"

For a standard version, this view must display the players in order of current credit balance (highest at the top, lowest at the bottom).  
As an advanced task, display players in the order of their net worth. This is calculated as the summation of their current credit balance and the purchase price of all properties owned.

**Task 4:**  
Write SQL queries that model the following rounds of gameplay (a round consists of one turn by each player). The gameplay must follow on from the given initial state, which you must accept as accurate and complete.

Round 1:  
1: Gareth rolls a 4  
2: Uli rolls a 5  
3: Pradyumn rolls a 6, then a 4  
4: Ruth rolls a 5

Round 2:  
1: Gareth rolls a 4  
2: Uli rollss a 4  
3: Pradyumn rolls a 2  
4: Ruth rolls a 6, then a 1

These queries could consist of update statements to tables that make the required changes to the database. Or choose to exploit other features of SQL to make this more advanced.

In any case, the database must be brought into a state that accurately reflects the new state of the game.

**Task 5:**  
Create a video that explains your database design. This video should contain the following:

* Your face and voice. This could be in the corner of the screen using a webcam (you can optionally include subtitles or a transcript)
* Your ER Diagram, displayed clearly for at least 5 seconds
* Explanation of your database design, including assumptions made, and design descisions, along with well detailed defence/reasoning of these assumptions and descisions.
* Explanation of your SQL code. Be selective on the code you explain (ie: if two queries are very similar, there is no need to explain both), and also do not explain line-by-line all your code. Defend your choice of query and statements used. There is no need to explain the code that creates tables or populates the database.
* The leaderboard view at the end of each round of game play. Ensure this is shown clearly for 5 secodns.

Through your video, you should aim to convey your knowledge and understanding in a concise way. You should assume that your viewer has some knowledge of SQL, but that they are not an expert. Your voice should be clear, and the volume of your voice should be auidible, but not so loud that the audio distorts. Your visuals should be clear and should be relevant, ensuring they aid your explanation. 

You should not show your database "executing".

You should include the leaderboard view at the end of each round of game play. Ensure this is shown clearly for 5 secodns.

Your video should be no longer than 10 mins, and should be in MP4 format.

**File Submission:**

* Submit files using the submission point below this brief
* Your video should be named <student number>.mp4.
  + eg: If my student number is 12345678, then I would submit 12345678.mp4
* Use the following file names for your SQL code. Do not put anything other than what is requested in each file:
  + create.sql : This file must contain the statements required to create all the structure of your database, aswell as any triggers you may choose to create (including those in subsequent tasks).\
  + populate.sql : This file must contain the statements required to populate your database, such that it is an accurate representation of the initial game state as shown above.
  + view.sql : This file must contain the statemet that creates the view
  + q1.sql : Round 1, move 1
  + q2.sql : Round 1, move 2
  + q3.sql : Round 1, move 3
  + q4.sql : Round 1, move 4
  + q5.sql : Round 2, move 1
  + q6.sql : Round 2, move 2
  + q7.sql : Round 2, move 3
  + q8.sql : Round 2, move 4

**Regarding Your Assumptions:**

You can make reasonable assumptions, but you MUST state them in your video.

However, you must assume that the initial state is  **complete** and that no further actions or manipulations need to happen - the only changes you should make are those which are directly coming from the game play moves given in the brief.

**Useful Tools:**

You may wish to use Lucid Chart to make your ER Diagram (this is not an endorsement):  <https://www.lucidchart.com/pages/er-diagrams>

For making your video, you can use OBS for screen recording ( <https://obsproject.com/download>) and you can use Shotcut for editing (this is a very basic video editor with a low barrier of entry)   <https://www.shotcut.org/download/>

You may wish to check that your SQL files are named correctly, and that they are accessible to the automarker using this script (place it in the same directory as your files, and then run via the terminal):    
 [UniversityTycoonChecker.py](https://online.manchester.ac.uk/bbcswebdav/pid-16479026-dt-content-rid-189733829_1/xid-189733829_1)  
   
**You are not obliged to use this, and use of this script does not constitute any formal marking process.**

SQLite, as you are aware, has some unique aspects compared to MySQL or other DBMS. You should always look at the Documentation:  <https://www.sqlite.org/docs.html>

**Deadline**

The deadline is Sunday 10th November, 23:59