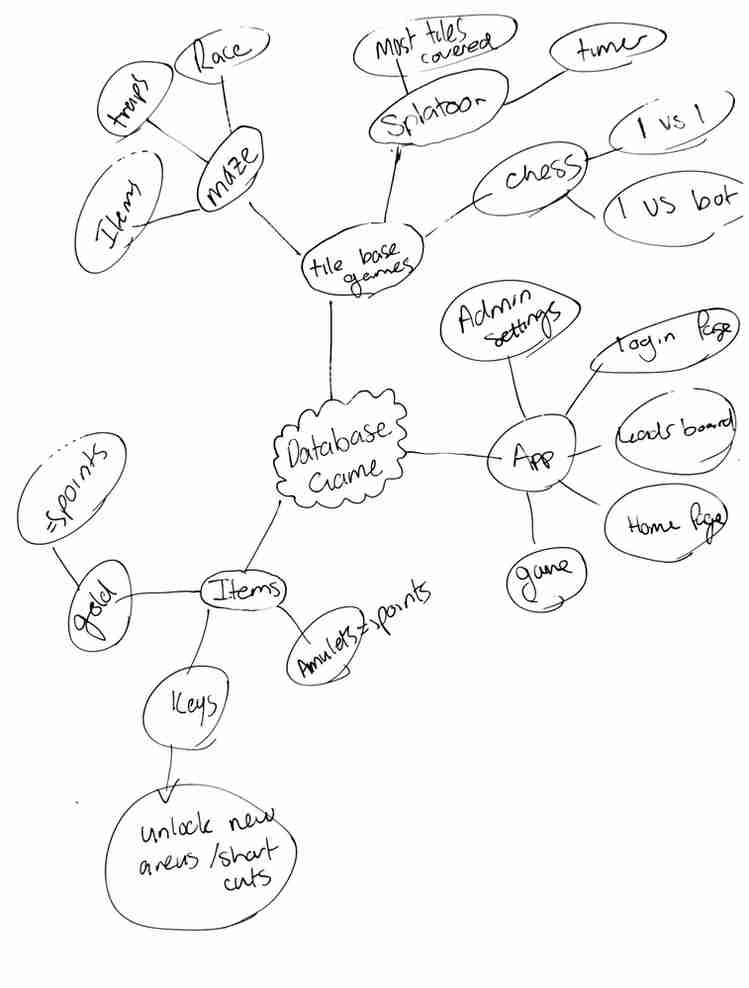
# Blog Post 1

## Sessions 1 – Introductions, what is the project?

In this lesson, we discussed ideas for our project for this semester. We were shown an example of what a previous student had done in other years. After a brief explanation of the requirements, I took a close look at what it was that I needed to do. I decided to plan out potential ideas for what I am going to develop. I created a brainstorm of possible games, features, app layouts, as shown below:

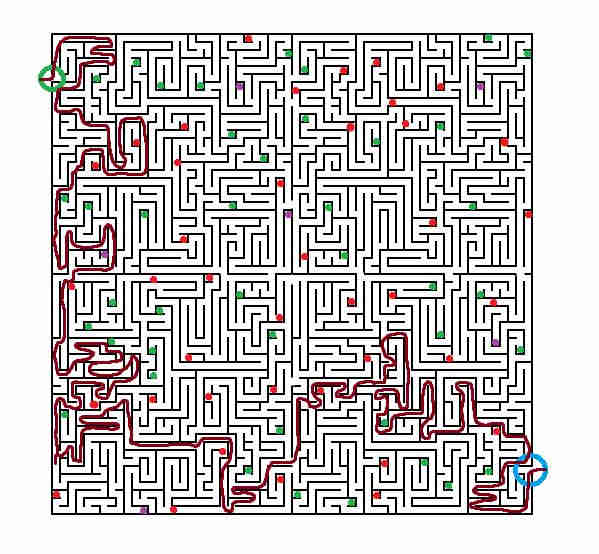


I liked the concept of a maze game. This would involve players racing through corridors tile by tile trying to reach the end. The maze would contain a variety of different elements such as dead ends, traps, locked doors, hidden treasures, and secret shortcuts.

The objective is to have the highest amount of points at the end of the race. This means that the fastest route isn’t necessarily the best route, although being first out may give a certain amount of points. Throughout the maze will be randomly placed items such as keys, gold, amulets/treasure, and possible boosts that allow you to move more than one tile at a time. The keys would be used to open doorways that lead to special items or shortcuts.

You can either try to race through and get the points for being the fastest which will add to the overall leaderboard or collect items along the way to get a higher score.

Items such as gold might be worth an additional 10 points, whereas an item such as a gold ring might grant you 40 points on your overall score. I could possibly add features that could cause a player to lose points from falling in a trap or getting attacked by something (maybe random chance events). These may be future additions to the game. I came up with a concept design of how the maze may look. The prototype for the maze would remain the same but place items in different locations each time. A procedurally generated map could be something to look at in the future. This design can be seen below:



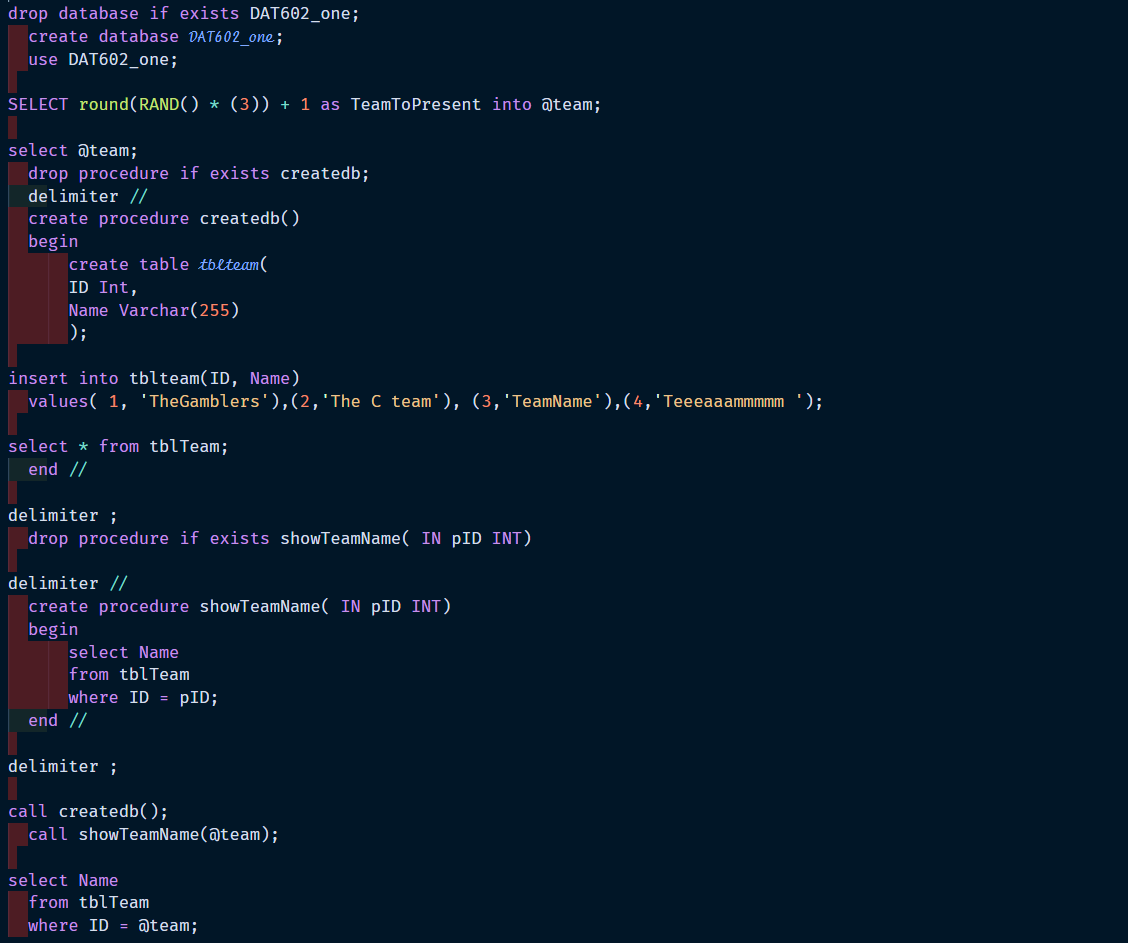
* As we can see there are several different elements within the maze:
* The green and blue circles represent the start (Home) and the finish
* The green dots represent the items that you can collect throughout the maze to gain extra points
* The purple dots are the super rare items that can be kept in your inventory across multiple games, these give you special abilities. These could possibly be permanent or be lost if another player steals them from you or if you land in a trap. (Concept idea)
* The red dots represent traps or monsters within the maze. If you land on these tiles, you will lose a random amount of points from the points you have gained throughout the maze.
* The blue circle will be the finish of the maze. When a player reaches this point, they will return to the main menu. The points they have gathered will add to their overall score on the leaderboard. The more games you play and the more points you finish with, the higher you will be. In the future there may be additional stats such as games played, items gathered, average score etc.
* Any special items that carry over games will be shown on the home page so you can see what you have. I could try and add a feature in future, maybe not for the assessment, that allows a user to choose what items they take in the maze since there is a chance to lose them. I don’t think this will be a feature in the prototype but is on the roadmap for future development.

We were also shown an example of the creation of a database within MySQL. This included the creation of a table that included a name and ID of a team. We inserted some dummy data into the table using the script shown below:



We then ran some scripts that selected the names from the table and returned the values.

The full code looks like this:

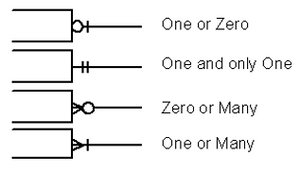


This was relatively straight forward as it had many similarities to the DDL scripts that we wrote in DAT502.

# Blog Post 2

## Sessions 2 – Entitiy Relationship diagrams

Today I did some learning about entity relation diagrams (ERD). ERD’s are a modelling type for database design. They display the relationships between entities within the systems scope using a variety of different symbols. We looked into crow’s foot notation (or IE notation) as the form that we are using for the assessment. Crows foot notation uses several different graphical symbols to represent the relationship types. Below we can see the different types of relationships.



This method was relatively easy to do as we had covered this in DAT502 last year. I did not have any problems remembering how to implement crows foot notation into an ERD. The only issue I faced was the planning of the entities within the database. I struggled to identify what was required in each table. To fix this issue, I went back to my planning and looked at what I could do to reduce the overall size of tables and remove redundancies. In doing this I was able to remove attributes from tables within the database to reduce the size of them. This was good for my learning as I was able to improve my understanding of how to create ERD’s

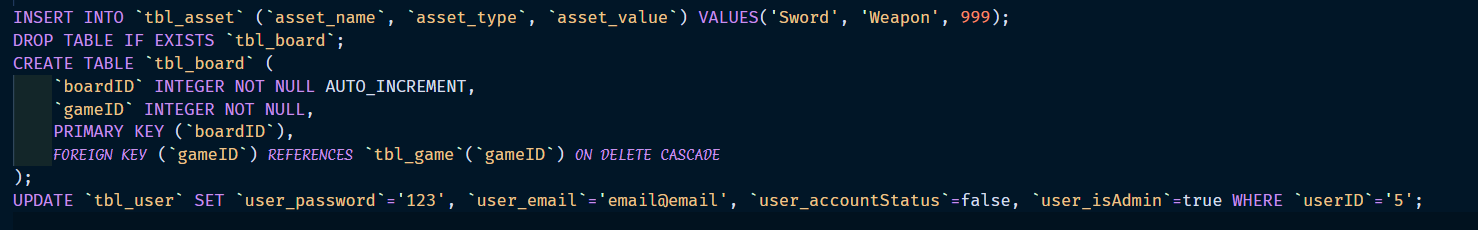
# Blog Post 3

## Sessions 3-4 – CRUD

What Have I Learned?

This week I learned about SQL procedures and how to implement them into my database. I learned about the basic procedures which are SELECT, FROM, WHERE, JOINS, FOREIGN KEY, PRIMARYKEY, and some different data types. I also learned about what storyboards and CRUD tables are and how we implement them.

Examples:

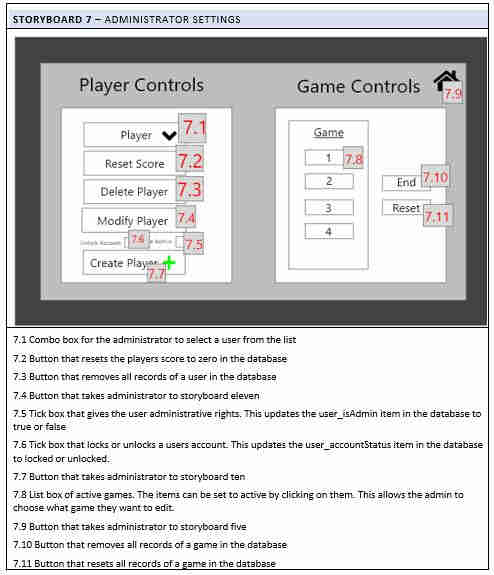


Why Have I learned This?

I believe that I have learned this as these procedures are key when create our databases and want to retrieve or manipulate data. These will come in very useful when It comes to coding our game and for our first milestone hand in. I also believe that these will b e very useful in future projects as well.

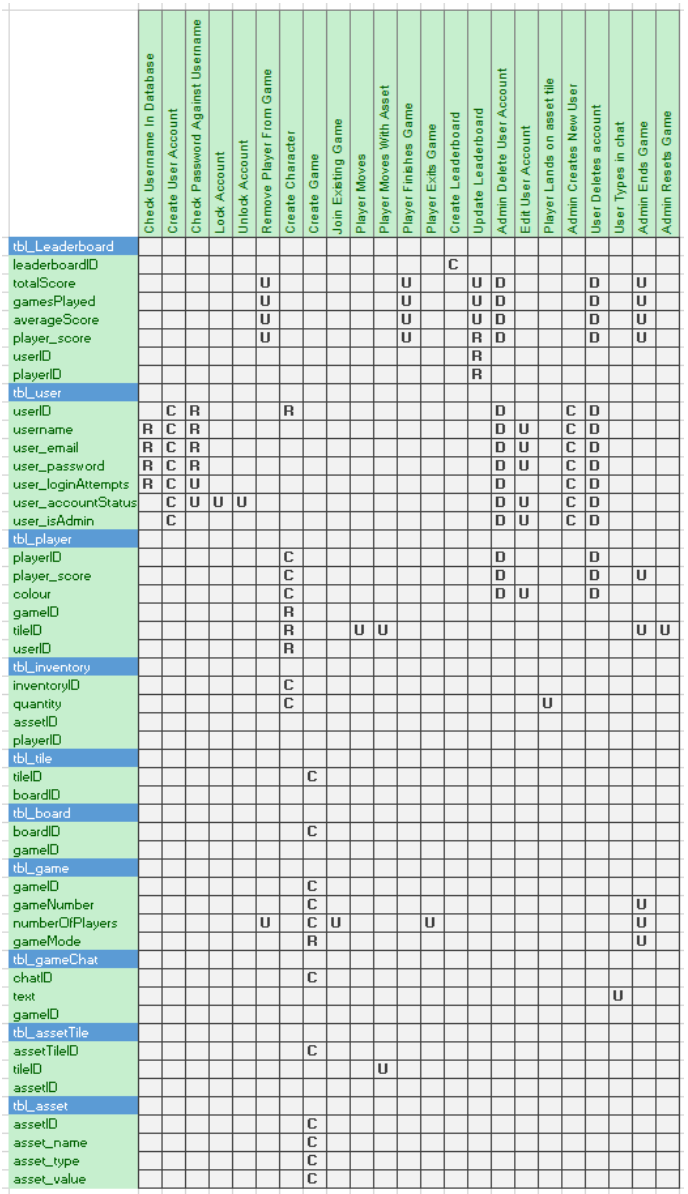
Storyboards give a good representation of how our system works and what actions happen when we do a certain task such as where a button takes us. These also give other users a walkthrough of the purpose of each of the screens we have designed.

Example:



CRUD tables are visual way of showing what tasks our system performs and how it interacts with our database. I believe that these are useful as it helps plan out what is required when it comes to writing queries to perform a certain task. When it comes to making the game, I can use it as a checklist to see if I have completed all the tasks that my system performs and does the required statements.

Example:



How Have I learned This?

I learned this by researching into how each of the procedures work and the purpose of each one. I looked at the default syntax of the procedures and how you would implement them into the system. I then ran through a few tests with some dummy data to see if the procedures work as expected.

I learned about how to create a storyboard by analyzing and exemplar and creating my version based on the idea of how it the exemplar was done. This did not require much learning as I had a reasonable understanding of how I needed to create one. This goes with the CRUD table as well. I’m not sure if I got a full understanding of the CRUD table, but I gave it a good shot and I believe that it gives a good idea of what actions my system will perform.

# Blog Post 4

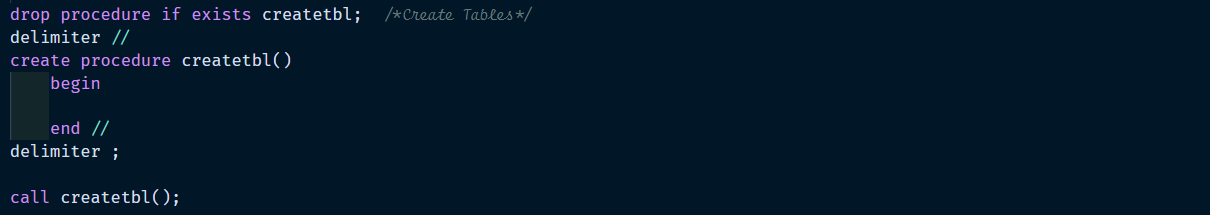
## Sessions 5-6 - Flow control in SQL and Update

What Have I Learned?

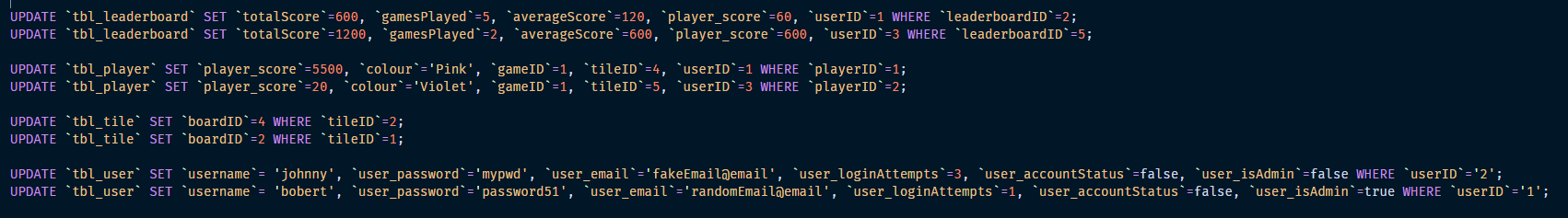
This week I learned quite a few key things as well as made a good start on my project. I learned about stored procedures and flow control, as well as UPDATE statements. A stored procedure is a section of SQL statements that are stored within our server. These can perform a variety of tasks but do not run unless we call it.

Example:

In this example, we have a procedure called "createtbl." We would place our SQL queries between the begin and end. This would store this procedure which we then call using "call createtbl();"

An update statement is used to update specific data within a table or multiple tables. This can be done on a certain row or multiple rows based on our WHERE clause. We can also specify what columns we want to update so that we don’t need to change all of the data of a row.

Examples:



Why Have I learned This?

Stored procedures are useful if we want to call a certain procedure later on, rather than all in one go. For example, creating a procedure that creates all of our tables before we call a procedure that inputs data into these tables. These procedures are cached for later use. I believe that the use of stored procedures will also come in very handy in future projects.

Update statements are very useful as we can manipulate data within our table after we have inserted our data. This will come in useful when we want to create the procedures for modifying existing users within our game. I believe that the use of update statements will also come in very handy in future projects.

How Have I learned This?

I learned this by researching what stored procedures and UPDATE queries are and how we implement them into our system. This was followed by doing a few test examples to see if I new what I was doing. This was not too much of a struggle as I picked up on it pretty quickly.

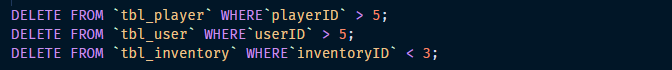
# Blog Post 5

## [Sessions 7-8](https://ecampus.nmit.ac.nz/moodle/mod/page/view.php?id=1054473) – Delete statements and database connection

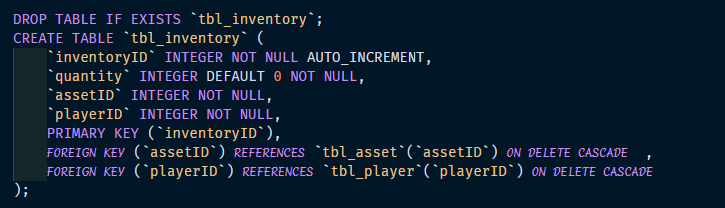
What Have I Learned?

This week I learned about DELETE statements, ON DELETE CASCADE, and connecting our databases with c#. We use delete statements to delete records within our database. This can be done or a single row, multiple rows, or all of the rows on a table.

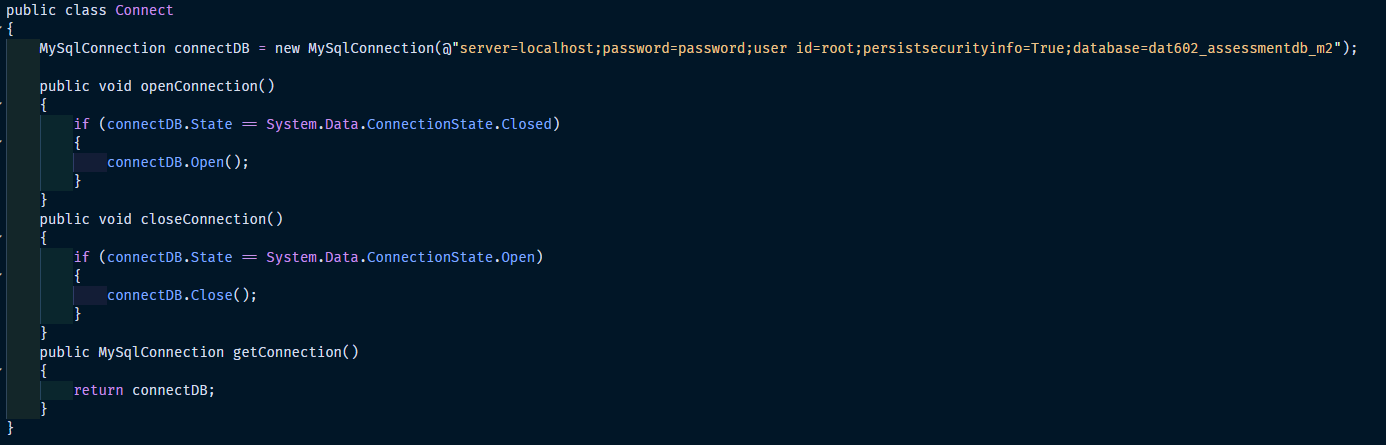
Here are some examples

We use the “on delete cascade” parameter on a foreign key so that when we delete the parent record, the child records also delete from the database. We can look at this as cutting a branch off of a tree. When we remove the branch, the leaves are removed as well since they are children of the branch.

Here is an example:

When we want to connect our application to the server and the database, we must first establish a connection so that we can perform actions such as insert statements and update statements. I decided I would make a class that I can call called “Connect” rather than establishing the connection on each form.

Here is what it looks like:



Why Have I learned This?

Delete statements are very useful if we want to remove records from tables within our database. This is especially useful for our project as we are required to delete users accounts from our game. We can use a WHERE statement to define what user we want to delete.

I believe that we learned about “on delete cascade” as it is a useful way of deleting related records across multiple tables. This will come in handy when we want to delete users from our games as they have a relationship with other tables.

I believe that we have learned about connecting our database with C# as it is a very useful skill to have. We need to make sure that our database can connect before we progress as there is no real point in trying to perform certain tasks if we cannot access our database.

How Have I learned This?

I have learned this by researching what a DELETE statement is and how I implement it, followed by practising it to ensure that I get the general structure of the statement is. I also looked into how we implement the cascade parameter and ran some tests to see if it worked as expected.

Learning how to connect my database was not too much of a struggle as there is not too much to it. All I needed to do was make sure that my connection string was correct and displayed the correct information. Once this was done, there was no need to go back and practice.

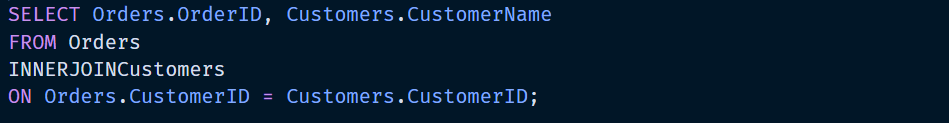
# Blog Post 6

## Sessions 9-10 – Group by Having

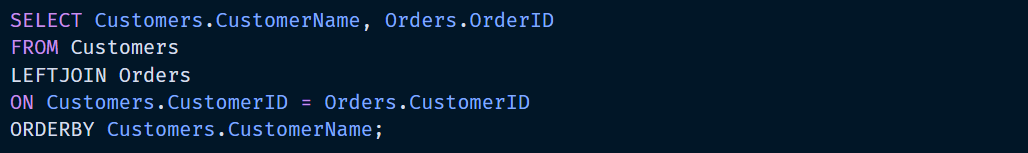
What Have I Learned?

This week I learned about different types of Join statements. These being:

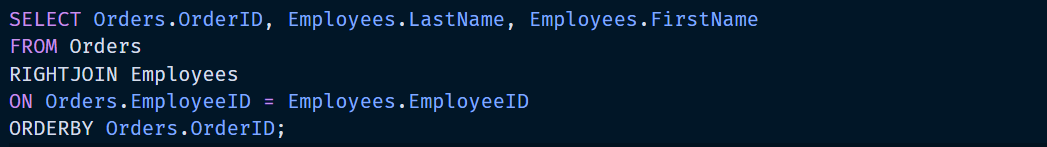
Inner Joins – Returns all records that have matching values in both tables



Left Joins – Is used to return all records from the left table and the matching records from the right table.



Right Joins – Is used to return all records from the right table and the matching records from the left table.



I also made a big start in coding the functionality side of my game. Through this, I have learned a lot about C# and how to run SQL procedures within C# to manipulate data.

Here is an example of how I would update some data:



Here is an example to see if the username already exists:



Why Have I learned This?

I believe that I have learned this as it is an important skill to learn when we want to select data from multiple tables. This will be useful when I want to create my leaderboard. I can select matching data from the user and the player tables to retrieve usernames, scores, etc.

How Have I learned This?

I have learned this by doing some research into what Join statements are and how I implement them into my code. I did some tests on the tables that I have created for my game to see if it works as expected.

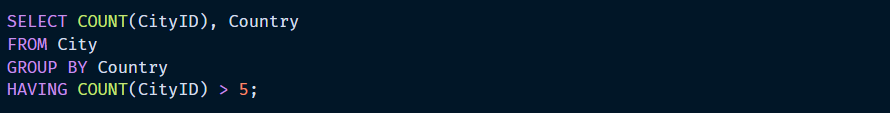
# Blog Post 7

## Sessions 11-12 – Group by Having

What Have I Learned?

This week I learned about ‘group by having’ statements. These statements are used instead of WHERE statements in the situations where we are using aggregate functions as WHERE statements cannot be used for them. In saying this, we can use a WHERE statement in the same query as a HAVING statement. A WHERE statement is applied to individual records whereas a HAVING statement applies to summarized group records. Only records that have met the HAVING criteria will be returned, like that of a WHERE statement. A GROUP BY clause must be present in order to employ a HAVING statement.

Here is an example of a ‘group by having’ statement:



This only selects countries with more than 5 cities.

Why Have I Learned This?

I believe that I have learned this skill as it can be useful in scenarios where I might want to summarize data from a database. Rather than calculating a table count application side, we can perform it database side and return the value that we want. For example, we may want to get the count of males and females having an age above 33. This can return a numerical value of the male count and a numerical value of the female count, as seen below:

|  |  |
| --- | --- |
| Gender | COUNT(‘Gender) |
| Female | 132 |
| Male | 222 |

How Have I Learned This?

I have learned this by doing some research into what ‘group by having’ statements are and how I implement them into my code. I did some tests on the tables that I have created for my game to see if it works as expected. I got the expected results and am satisfied that I understand how to use them for future tasks.

# Blog Post 8

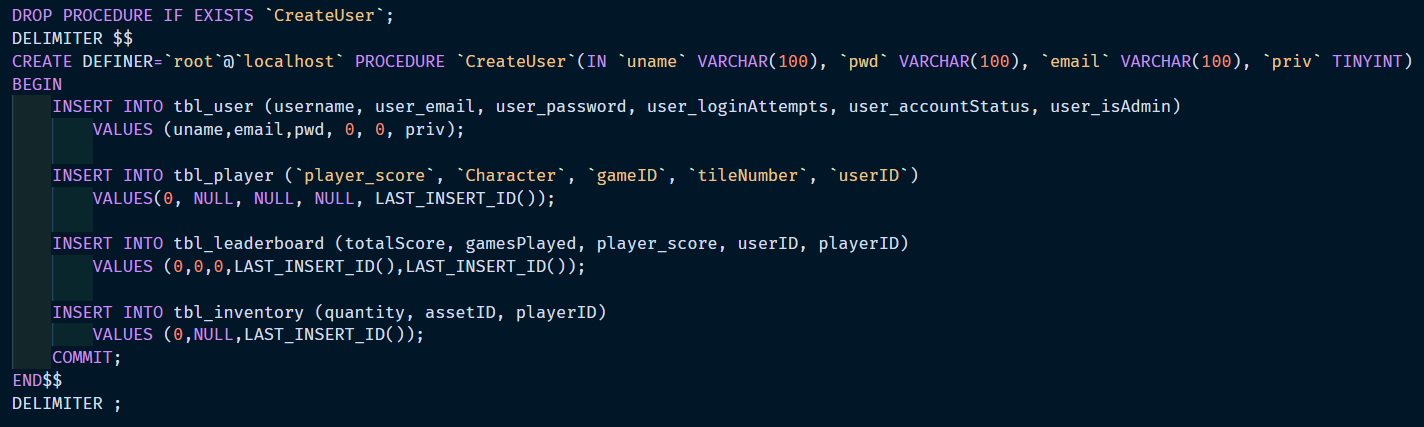
## Sessions 13-14 – Procedures and functions

What Have I Learned?

I learnt about stored procedures and functions this week. A stored procedure is a collection of declarative SQL statements that is kept on the server. Once a CALL statement has been used to construct these statements, they can be called at any time. The CREATE PROCEDURE statement contains these statements. Conditional statements such as IF and CASE statements are the only scripts that may be used inside a stored procedure. Stored procedures have the advantages of reducing network traffic, being simple to maintain, and being secure.

When we have several statements encapsulated within a stored procedure, we can call the stored procedure we send the name of the procedure and the values of the parameters rather than trying to execute multiple SQL statements. Stored procedures are also reusable and can be implemented in many areas of an application. This results in a more consistent database containing less code.

Below we can see an example of a stored procedure that I created for my game:



Why Have I Learned This?

As previously indicated, rather than attempting to run numerous SQL statements, we may call the stored procedure and pass the name of the procedure and the values of the parameters. Stored procedures are also reusable and can be used across several areas of a program. As a result, the database becomes more consistent and contains less code. This implies that understanding these approaches is beneficial to our learning since it ensures that our code is more polished, with less bloated and duplicated code.

How Have I Learned This?

I have learned this by doing some research into what procedures and functions are and how I implement them into my code. I did some tests on the tables that I have created for my game to see if it works as expected. I got the expected results and am satisfied that I understand how to use them for future tasks.

# Blog Post 9

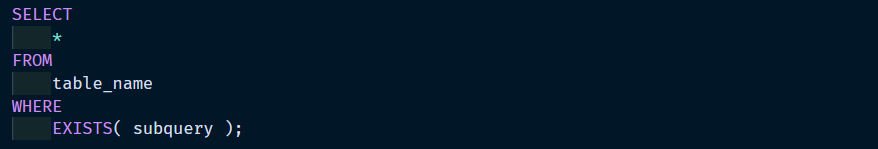
## Session 15 – Sub queries, referential integrity and cascades

What Have I Learned?

This week I learned about Sub queries, referential integrity, and cascades.

Sub queries: As stated by MySQLTutorial (2020), “A MySQL subquery is a query nested within another query such as [SELECT](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx), [INSERT](https://www.mysqltutorial.org/mysql-insert-statement.aspx), [UPDATE](https://www.mysqltutorial.org/mysql-update-data.aspx)or [DELETE](https://www.mysqltutorial.org/mysql-delete-statement.aspx). In addition, a subquery can be nested inside another subquery.” The query itself is referred to as an inner query whereas the statement that encapsulates it (i.e. SELECT) is referred to as the outer query.

Below we can see an example of how the query might be formatted:

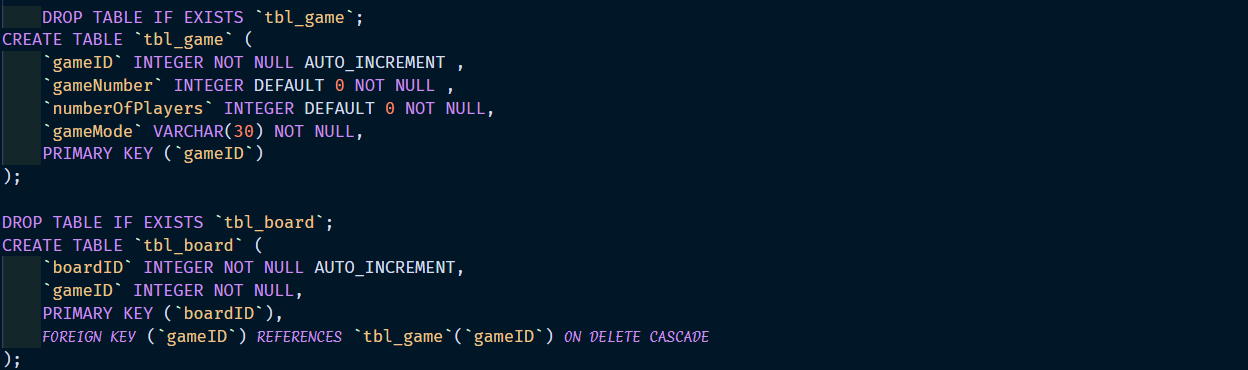


Referential integrity:

In database architecture, referential integrity is a crucial notion. The phrase refers to a condition in which all database references are legitimate, and no incorrect connections exist between the system's various tables. Any attempt to connect to a record that does not exist will fail if referential integrity is present. This helps to reduce user mistakes, resulting in a more accurate/reliable and usable database (Melonfire, 2006).

We generally implement referential integrity with the use of foreign keys. As stated by w3schools (n.d.), “The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables. A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another table.”

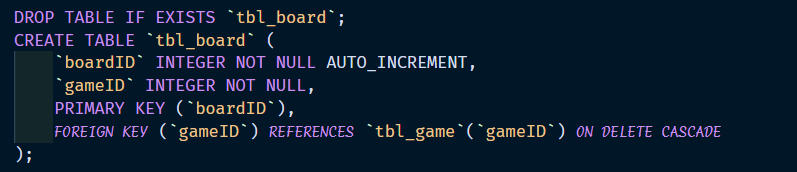
Below we can see an example of the use of constraints within a database:



Cascades:

Cascades are a method in which we can delete all relating records of child tables related to that of a deleted parent table. This relates back to the previous part on referential integrity. When we delete a record such as a user account, it is likely that we want to remove any other information about that user. If this information is spread across multiple tables, this data is not just removed when the parent is deleted. Therefor, we use on delete cascades to remove all child records.

Below we can see an example of the use of CASCADE within a database:



Why Have I Learned This?

I believe that I have learned this as it is a useful technique to ensure that data is not left in the database that is relevant to records that we have deleted. We do not want to use space with redundant data that could be filled with useful data.

How Have I Learned This?

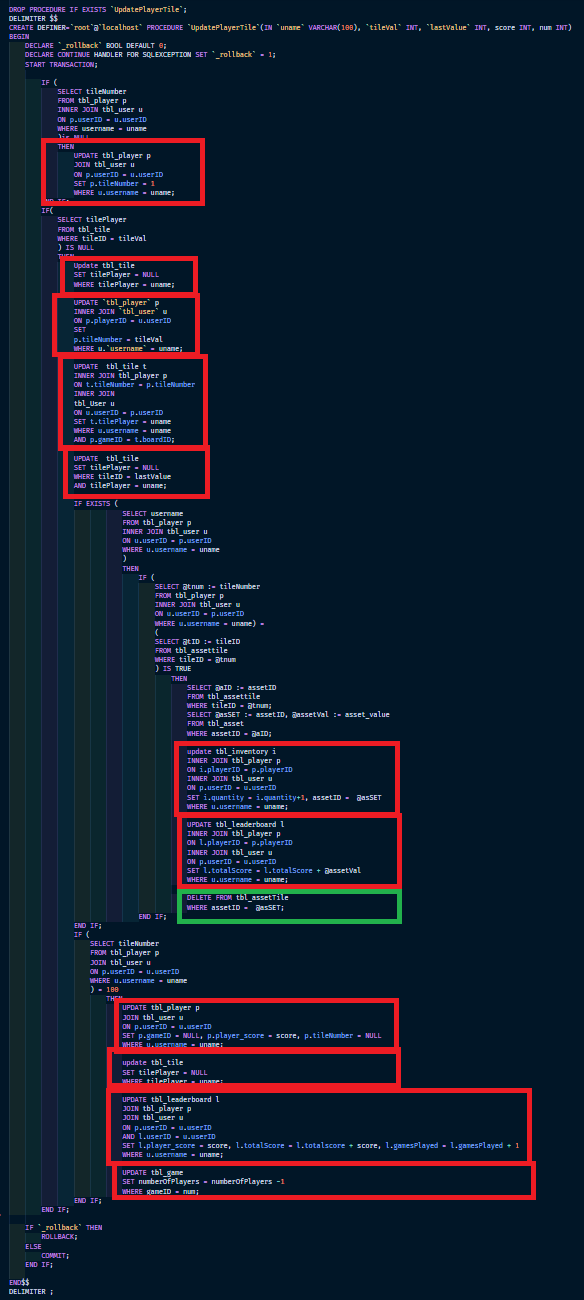
I have learned this by doing some research into what sub queries, referential integrity, and cascades are, and how I implement them into my code.

# Blog Post 10

## Sessions 11-12 – Transactions

What Have I Learned?

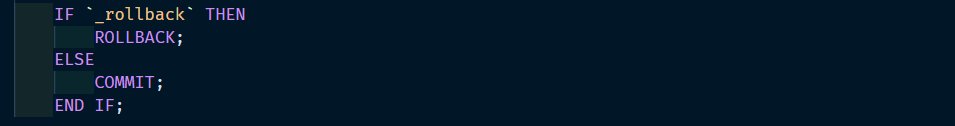
This week I learned about SQL Transactions. As stated by MySQLTutorial (2020), “MySQL transaction allows you to execute a set of MySQL operations to ensure that the database never contains the result of partial operations. In a set of operations, if one of them fails, the rollback occurs to restore the database to its original state. If no error occurs, the entire set of statements is committed to the database.” Transactions are very useful in many scenarios, especially when we are doing many statements in one procedure that we all to execute successfully, rather than the ones executing successfully up until the one that fails.

A good example of where we would want to use a transaction from my game is the procedure shown to the left.

I have highlighted all of the statements where we are either updating (red) or deleting (green) a record. If we were to run this without a transaction and it were to fail halfway through, any updates or deletes before the error point will still commit. This is very bad for the integrity of the data within our database. Therefore, we use transactions to create a rollback to ensure that the database never contains the result of partial operations.

By using a transaction in the stored procedure shown on the left, we can rollback the server to the state that it was in before the procedure was executed.

The method on the left has two declarations at the beginning of the procedure as shown below:

At the end of the procedure when we want to commit the changes, I initiate an if statement that checks the bool value of \_rollback. When true, a rollback is called, when false the changes commit. This is shown below:

Why Have I Learned This?

I believe that I have learned about transactions as they are very important when it comes to handling data in a database. If I am handling many insert and update at one time, I want to ensure that the data stays consistent across the records. It is very important that any relevant records are either manipulated fully or not manipulated at all (all or nothing).

How Have I Learned This?

I have learned this by doing some research into what transactions are and how I implement them into my code. I did some tests on the tables that I have created for my game to see if it works as expected. I got the expected results and am satisfied that I understand how to use them for future tasks.

# Blog Post 11

## Sessions 16-17 – LINQ

What Have I Learned?

This week we covered LINQ. I had already learned this prior to this class as I had used it in another class’s assignment. LINQ (Language Integrated Query) is a unified query syntax for retrieving data from many sources and formats in C# and VB.NET. It's integrated in C# and Visual Basic, removing the gap between programming languages and databases while also offering a single querying interface for many data sources (Tutorialsteacher, n.d.).

Why Have I Learned This?

LINQ is a sophisticated query language that may be used in a number of circumstances. When creating apps, there are numerous techniques we may employ to accomplish things, but some of them are patchwork, resulting in code that is longer than it has to be. It is preferable to employ a declarative high-level approach that explains what we want rather than elaborating on how we will achieve it. The great thing about LINQ is that it can be implemented in the same way regardless of the suitable data source you use.

LINQ provides us with a variety of benefits such as being a first-class language construct (like classes and methods) and is a consistent experience across objects (such as lists and arrays), relational databases (such as MYSQL), and XML.

Below is an example of LINQ that I have implemented into my code:



This changes the data source of a datagridview to a list of vehicle objects from a business's vehicle list and sorts the list in ascending order by car registration. This technique is considerably more readable and succinct than using a foreach loop to read over the list and add items to the datagridview. I noticed that, while a for loop appears to function, it really generates a duplicate of the data rather than directly referencing the objects in the list. This indicates that you're refining the data copies, which means that any changes you make to the data won't affect the parent list.

How Have I Learned This?

I have learned this by doing some research into what LINQ is and how I implement them into my code. I practised it many times and implemented it in as many scenarios as I could to ensure that my code was not bloated with unnecessary code. There were many situations where I found myself writing endless lines of code that was easily refined using LINQ. There are many tutorials and references to how to use LINQ online that have also helped me implement LINQ into my applications.

# Blog Post 12

## Sessions 18-19 – EXCEPTIONS

What Have I Learned?

This week I learned about exceptions/exception handling. As stated by TutorialsPoint (n.d.), “An exception is a problem that arises during the execution of a program. A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.”

Here is an example of implementing exception handling into my project:



As you can see, I have implemented a try and catch handler. This will try to perform the tasks that are encapsulated within the curly brackets. If it completes the task correctly, it will return the new data table. If it throws an error at any point, a dialog box will display with the error message string, then close the connection and return. In doing this, we can see what the error is that is occurring when the code runs. Rather than having the \_conn.Close() method being called, I can also implement a “finally” to close the connection.

Why Have I Learned This?

I believe that I have learned this as exceptions are very useful to help us trouble shoot our code. We can also use exceptions to perform other tasks in scenarios where we might want to do so. C# provides us with an easy to use structure for handling these exceptions, therefor I believe that it was a very useful and important skill to acquire.

How Have I Learned This?

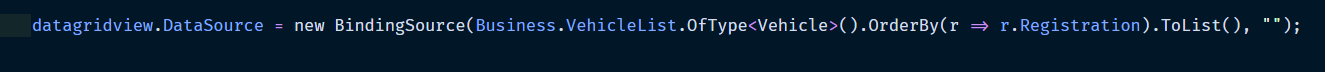
I have learned this by doing some research into what exceptions are and how I implement exception handling into my code. I performed a series of tests using stored procedures from my game and calling them from the Windows form application. I purposely made errors to see if it was picking up on them correctly. I got the results I expected and was satisfied.

# Blog Post 13

## Sessions 20-21 – GUI COMPONENTS

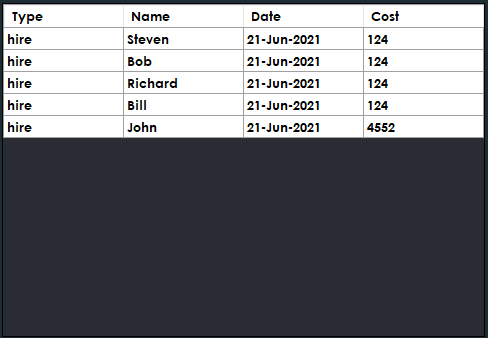
What Have I Learned?

This week I learned about GUI components, more specifically, datagridviews. DataGridViews are a very useful control for displaying data from a data source. There are a couple of good ways that I have learned through self-taught learning for connecting the datagridview to a data source. The two types of data sources that I have worked with are lists and a database. By using a data source property on a datagridview, we are able to bind that data to the control. Once bound, we can easily view and manipulate the data. Any changes to the data source can easy updated on the display just by setting the data source of the control to a new instance of the same data source, as shown below.



Every time the data is changed in the data source, we can rebind the data and update the display by calling the method above. The downside to using a data source is that you can’t perform the same methods for adding and sorting as you would with a datagridview that is not data bound. To my knowledge, when you bind a data source to a datagridview, by default it also automatically generates all of the columns, thus requiring you to hard code remove any that you do not want to display.

There are many cool features that datagridviews have that you can enable if you want. Such as making setting a column type to be either a button column, combo box column, check box column, image column, link column, or text box column. This opens a wide range of possible uses for our datagridviews.

Below is an example of a datagridview that I implemented into one of my other assessments:

This datagridviews data source is bound to a list of objects within a specific class. Each object has the four properties displayed across the column headers. There are two ways that I tried to achieve adding this data to the datagridview. One way was using a foreach loop that ran through the list, while the other used a binding source for the datagridviews data source. The issue I found with the foreach loop was that it was not directly referencing the list (almost like a clone), so any data manipulation would not change the data that was in source list. Whereas, using a binding source for the data source directly referenced each object in the list.

Another issue I ran into when using a binding source was that you cannot use the column headers to sort the rows by that columns data. When using a binding source, this throws an error saying that you can’t sort the datagridview as it is data bound. Therefor, you need to implement the sorting method when you set the data source of the datagridview. The method I used to fix this was to have a button that has a method that checks a bool value. If the value is true, then it sorts the source list by ascending, else descending.

Why Have I Learned This?

I believe that I have learned this as datagridviews are a very easy and useful tool for displaying data from a data source. There are other methods that we could use such as list boxes or multiline text boxes, but datagridviews seem to have a larger range of features while also having a lot more room for design manipulation. I have found that my experience with datagridviews has been far better for my learning than trying to use other controls like list boxes. There have been a few hurdles with learning how to use them as you have to make a few decisions along the way of how you wish to implement them.

How Have I Learned This?

I have learned this through a lot of trial and error as well as watching YouTube tutorials on different ways of using datagridviews. I have spent a lot of time practicing different techniques of how I can implement them as well as playing around with different configurations.

# References

Melonfire, C. (2006, February 6). *An introduction to foreign keys and referential integrity in MySQL*. TechRepublic. https://www.techrepublic.com/article/an-introduction-to-foreign-keys-and-referential-integrity-in-mysql/

MySQLTutorial. (2020a, April 11). *MySQL Subquery*. https://www.mysqltutorial.org/mysql-subquery/

MySQLTutorial. (2020b, April 11). *MySQL Transaction: START TRANSACTION, COMMIT & ROLLBACK by Examples*. https://www.mysqltutorial.org/mysql-transaction.aspx

Tutorialspoint. (n.d.). *C# - Exception Handling - Tutorialspoint*. Retrieved June 21, 2021, from https://www.tutorialspoint.com/csharp/csharp\_exception\_handling.htm

Tutorialsteacher. (n.d.). *What is LINQ*. Retrieved June 20, 2021, from https://www.tutorialsteacher.com/linq/what-is-linq

w3schools. (n.d.). *SQL FOREIGN KEY Constraint*. Retrieved June 20, 2021, from https://www.w3schools.com/sql/sql\_foreignkey.asp