Coursework 01 - 2021 General Feedback

General Notes:

As you know we do not have to provide a detailed marking scheme for assessments (Marking Schemes). What we have provided with this coursework is the marking rubric which is available to you all in Blackboard. We deliberately do not provide you with a detailed marking scheme for the exercise, to do so would undermine the assessment process. Instead we provide some general guidance. Our aim is that you learn to take the material we have presented and learn yourself how to put this in to practice. As a reminder: a mark of 70%+ is an excellent (1st class) mark. In terms of degree classifications, a mark of 70% and a mark of 100% are equally 1st class and we do not distinguish between them. You should limit your expectations that you should be receiving 100% for everything you do. A key part of the learning process is to make mistakes and learn form these, there's nothing wrong with this. So, please do not question the mark awarded. We do not have to explicitly say how parts of the coursework should be done in the assessment script and any queries about this will be re-directed to this document and the link above.

Due to the large number of students taking the course unit, we are unable to provide additional feedback beyond that you have already received, so please do not contact us to request further feedback, or feedback to be explained in more detail. You are free to ask a GTA to go through your design and solutions in your lab drop-in session. If you contact us about your mark/feedback, then we will simply direct you to read this email. We will also not be providing the model answers to any part of the assessment beyond what is covered in this document and the marking rubric.

Part 1 - Monopolee ER Diagram:

As you already know there are usually many ways an ER Diagram can be made and therefore there is no model answer for this part. This is why the rubric was made the way it was. My approach would have been to go through all of the requirements and find all of the entities (including weak ones) first and create a table for them. From here adding all of the relevant attributes to the diagram would be a sensible course of action followed by distinguishing the PKs for each entity. If you included the FKs in your diagram then this is fine but not needed for this part.

Probably the hardest part of this part is deciding on suitable relationships between tables. Some of these were given to you in the requirements, see Requirement 3 for example which said "A player must choose one and only one token". It should be very easy to see from this that there must be a relationship between these two entities and one side of the connecting line must be a 1:1 notation. some relationships are not explicitly stated within the requirements so you need to use your own judgement and knowledge of ER design and apply the most suitable relationship. This is where the short report comes in. It was an opportunity for you to justify your design decisions such as why two tables have a certain relationship.

Overall the ER Diagram portion of the coursework was very well done with most students achieving high marks for this part, most common marks lost were due to missing entities or missing relationships

between entities.

Part 2.a. - Physical Design:

Overall, this was probably the part of the coursework which was done the best. Some students add the constraint of *NOT NULL* to a *PK*, this is unnecessary as by default if something is a PK then it cannot be *NULL*. The data types for the most part were also done well and as long as you used something sensible for this then the marks were awarded.

Part 2.b:

There was nothing to assess here.

Part 2.c. - CREATE Statements:

For this part you were tasked with writing all of the CREATE statements for the University Database. Something you had to be very mind full of was the order in which you created your tables. A method I sometimes use for doing this is to first write the *CREATE* statements for each table with their respective *PKs* in any order. That is the easy bit done.

Obviously you could now test these to see if they are created correctly and I recommend that you do. However, none of the tables will be connected ye as we have no considered the *FKs*, this is our next task.

By using the given ER Diagram and your Physical Design you should already know which tables need to be linked together. There will be a handful of tables which do not require a foreign key at all (e.g. *Department*) so it would make sense to create these first. In the ER Diagram if a table on has arrows going into it then that means that other tables rely on it but it does not rely on the existence of any other table. If we look at the *Advisor* table then this has two arrows going away from it (towards the *Instructor* & *Student* tables) this means that we cannot create the *FKs* for this table until the others have been created. From this we could determine that this would probably be one of the later tables we create. As all of the individual *CREATE* statements are written it is a simple task of copy & paste to move the tables around into a sensible order and add in the *FKs*.

If you follow this process through all of the tables you should then end up in a state where you can create all of the tables with the relevant *FKs*. Some students specified the storage engine when creating the tables, this was not needed but did not affect the marks. Finally, some students lost marks as they produced *FK* errors when creating their tables as the two referenced attributes did not have matching data types, make sure you double check the data types when creating a *FK*.

Part 2.d. - Inserting the Data:

Here you could test that you successfully created the tables by inserting the provided data. Nothing to submit for this part.

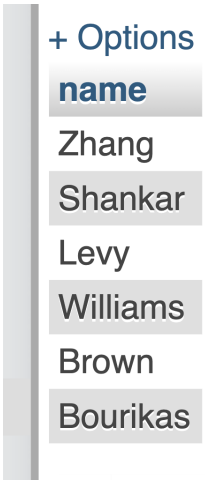
Part 2.e. - The 5 Queries:

Again, generally the queries that were created on the whole were very good but there were a few common mistakes which I will discuss here (I will discuss a good aproach for query building after this):

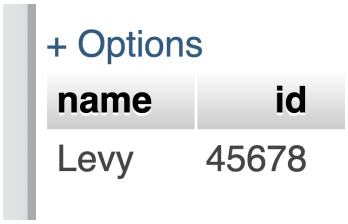
- Some students wondered why they were awarded 0 marks if their query did not produce the correct output. Simply put this is because your query was wrong. Lets look at it in terms of working in industry, if you produce queries which do not work or produce incorrect outputs then beyond getting a potential telling off from an employer this could lead to bigger issues and vulnerabilities later on.
- *NATURAL JOIN* Vs. *INNER JOIN* - Some students elected to use a *NATURAL JOIN* in their queries, whilst on the face of it they appear to perform the same job as an *INNER JOIN* there are a few key differences which mean that they are not considered to be appropriate. The *NATURAL JOIN* connect the two tables based on matching attributes and then simply displays that column once in the output where as with the *INNER JOIN* you have to explicitly state which the connecting columns are. This may seem like a more long winded way to do it but this method means that you or your colleagues always know which column is joining the two tables. When using a *NATURAL JOIN* the column linking the two tables together can change unexpectedly and therefore cause issues with your database and systems later on. More detail on the difference between the two joins can be found here: NATURAL JOIN Vs. INNER JOIN
- *Explicit* Vs. *Implicit* Joins - In some respects this can be seen in a similar way to my previous point. One of the major advantages of an explicit join is that it is far easier to see how tables are joined together and how they interact with each other. Whilst this may seem unnecessary for simply joining 2 tables together it is very important when joining multiple tables which have lots of filtering applied to them. Again, linking to industry, by running an implicit join it can be unclear to a colleague as to how you have linked tables together and potentially lead to errors in the future. implicit joins using the WHERE to specify the joining columns mean that maintaining the WHERE can be complicated and it is easy to introduce errors when modifying or extending the query in future, whereas explicit joins separate the join and filtering, making the WHERE easier to update and more logically contained. Implicit joins are often seen as an outdated method for queries. By using an implicit join you are essentially performing a *CROSS JOIN* with a larger set of filtering. For more information about the difference between an *INNER JOIN* & *CROSS JOIN* then please view this post (specifically the final section): INNER JOIN Vs. CROSS JOIN

In terms of finding the correct output for a query, I like to build my queries in a few different stages. Firstly (especially on an unfamiliar DB) one should become familiar with what data is actually contained within each table. Whilst it is technically more efficient to select each column individually (Why you shouldn't use *SELECT* $*$) performing a: `SELECT * FROM table_name;` would show you what is contained within each table. From there it can be quite useful to note the important information of each table especially when you know exactly what output you expect from your query. After this I start by selecting all of the relevant columns for my output and then to make sure these work join the relevant tables together. It can be useful at this point to run your query to get the full view of the output before applying any filtering. Once happy, apply the required filters to your query and then double check this against the ERD to check if this all makes logical sense.

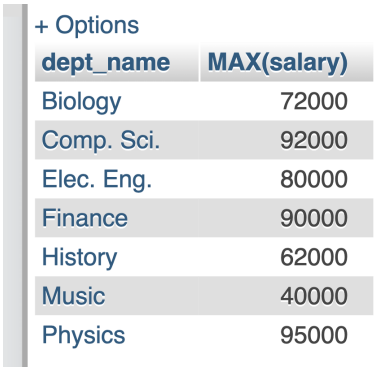
The Expected Outputs:



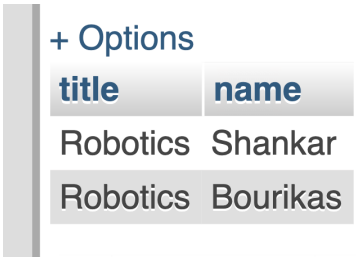
Query 1 - Find the names of all students who have taken at least one computer science course, making sure there are no duplicate names in the result



Query 2 - Find the IDs and names of all students with a fail grade



Query 3 - For each department, find the maximum salary of instructors in that department. Every department should have at least one instructor



+ Options

title	name
Robotics	Shankar
Robotics	Bourikas

Query 4 - Find the names of all course and the students enrolled on them which take place on a Friday afternoon and have at least 2 students enrolled on it

name	course_id	capacity
Srinivasan	CS-101	500
Katz	CS-101	500
Wu	FIN-201	500
Mozart	MU-199	500
Brandt	CS-190	70
Brandt	CS-319	70
Srinivasan	CS-347	70
Kim	EE-181	70

Query 5 - Find the names of all instructors and their course unit who teach on a course which uses a classroom with a capacity greater than 50