- [Good relational database design]: Explain*
- [Modification anomalies]: Explain*? List
    - [Type 1]: Explain? EG
    - [Type 2]: Explain? EG
    - [Type 3]: Explain? EG

- [Normalisation]: List forms
    - [First normal form]: Explain
    - [Second normal form]: Explain
    - [Third normal form]: Explain

- [Process of normalisation]: Explain (Refer down there for the steps)

- [Partial functional dependencies and transitive functional dependencies]: Explain
- [Dependency preservation]: Explain
- [Lossless join property]: Explain

2 cards x 2 Min = MAX 4 MINS

Process of normalisation

Identify candidate key or more specifically the primary key-

- The primary key is a set of attributes that must be unique (no repeats) + determine every other attribute
- IGNORE candidate keys. Very confusing just think of primary key only

From there we can identify problems with current design-

- This is normally modification anomalies: Insertion, update, and deletion anomalies (do it in this order since simpler. Refer to assignment 1 for guidance)

Identify direct functional dependencies-

- The direct functional dependencies should be the most the most obvious ones (usually on the start or end of relation)
- At the end Include the primary key as a functional dependency but this FD determines the remaining attributes (the remaining attribute can be on the left or right of the arrow)
    - Must be done at and only left over

Work out which normal form it is currently in-

Start normalising-

- Often requires putting FD into new relations

Identify new design normal form-

EG:

StudentID	Name	Course	Semester	Year	Grade	UnitCode	Title
S1	Kyle	IT	1	2020	HD	ICT231	Systems Analysis
S1	Kyle	IT	1	2020	D	ICT208	BI T&T
S2	Helen	IT	1	2019	D	ICT231	Systems Analysis
S2	Helen	IT	1	2019	C	ICT218	Databases
S2	Helen	IT	2	2018	N	ICT218	Databases

Identify candidate key or more specifically the primary key-

StudentID, UnitCode, Semester, Year ← Because this is only combination of attribute where every record is unique

From there we can identify problems with current design-

Refer to assignment

In the current design, Insertion anomalies would arise when we want to add another Item in the relation. For example, the addition of a new Item:

…..

Another potential problem that may arise with the current design is an update anomaly. For example, an update

…

Identify direct functional dependencies- [Look at column with same rows. Check inconsistencies]

UnitCode → Title

StudentID → Name, Course (Yes course because S1→ IT AND S2 → IT always so no inconsistencies).
Therefore (not semester because S2 → 2019 and S2 → 2018)

StudentID, UnitCode, Semester, Year → Grade (LEFT OVERS)

Work out which normal form it is currently in-

Currently in 1NF meaning there is no repeating groups or nesting in the relation. But there is partial functional dependencies. A non-key attribute in FD Title is determined by a compound primary key attribute FD (UnitCode). Therefore, through the progress of normalisation we convert the first normal form design to second normal form design to remove some of the modification anomalies identified.

Start normalising-

Relation1 (Item Number, Item Description, Fee)

….

Relation3 (**Patient ID**, **Item Number**, **Provider Number**, Consult Date)

Identify new design normal form-

The current design is now in at least second normal form meaning there are no partial function dependencies. However, given there are no transitive functional dependencies as well, the design is really in third normal form. A transitive functional dependency is where a non-key attribute is determined by another non-key attribute. Since the new design is in third normal form the problems associated with the previous design are resolved.

Firstly, with the new design there is no possibility that insertion anomalies can arise.

…

Secondly, there is now no possibility that update anomalies can arise.

…

Thirdly, the new design also addresses the issue of deletion anomalies arising.

….

The set of relations for the new design also supports the lossless join property and dependency preserving.

….