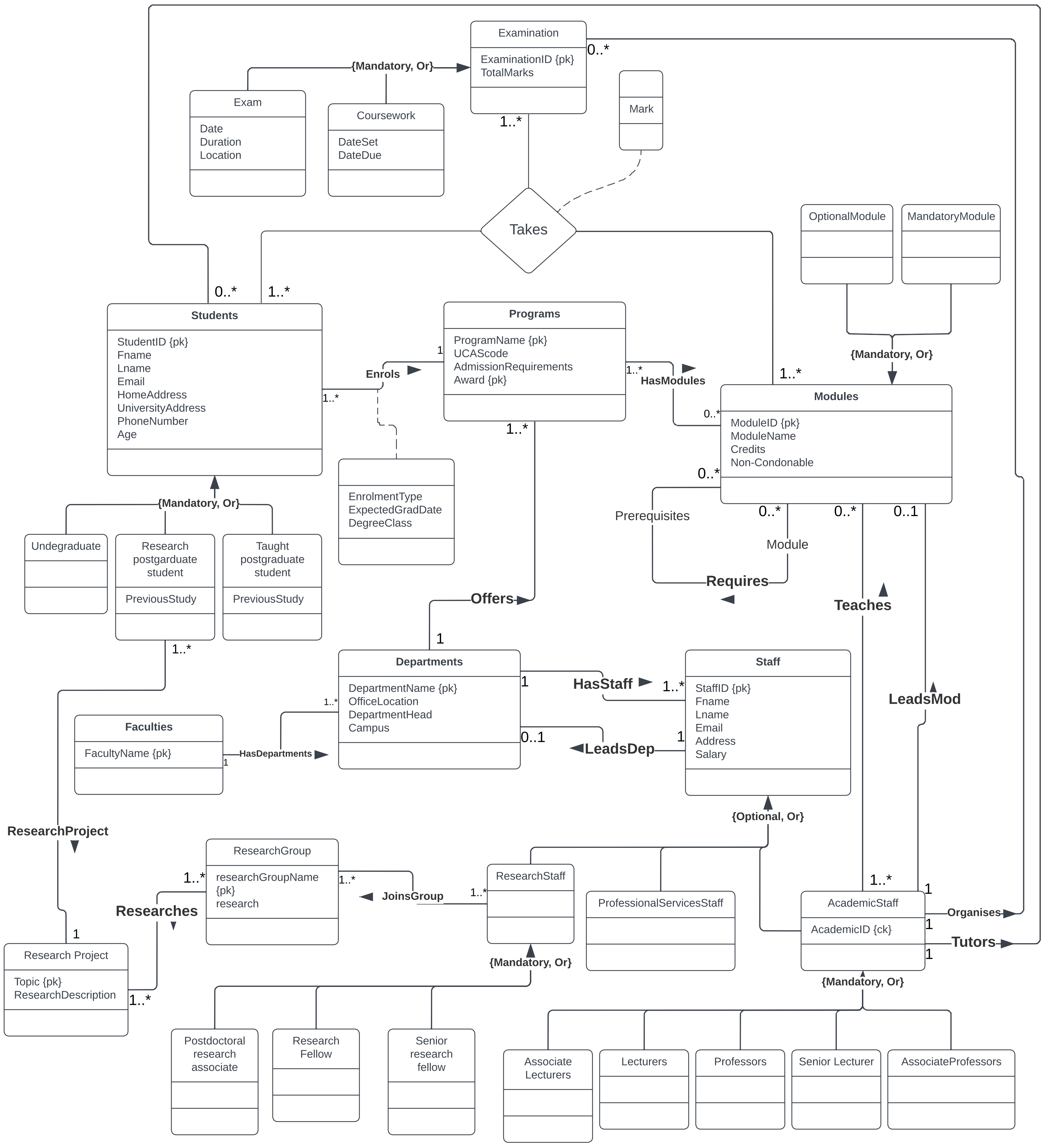
Part 1 | Entry Relationship Diagram with UML notation



Entity	Description
Faculties	As shown in the specification departments have faculties. I have created an entity called Faculties which will contain all faculties in the university. Such as the faculty of Environment, Science and Economy as mentioned in the specification. **Attributes**: - FacultyName: this is the primary key that identifies the faculty. I have used its name as it is unique. **Relationships**: - HasDepartments
Departments	To represent the computer science department, I have created an entity to contain all information about a department in a faculty. This can contain the department of computer science and will be linked to all other data about the department, such as the office location as described in the specification. **Attributes**: - DepartmentName: this is the primary key of the department; it is a unique name to all other departments. - OfficeLocation: this is the location of the department's offices; I have included this as in the specification the computer science departments offices are included. - DepartmentHead: this is the head of the department - Campus: what campus the department is located at. **Relationships**: - Offers - HasDepartments - HasStaff - Leads
Students	As described in the specification students are enrolled onto program. I have created an entity to contain all information about students at the university. This entity will have relationships with other entities, which will describe the student's university achievements and progression within their chosen program. Generalisation/specialisation is used within my student's entity. As described in the CA specification a student can either be an Undergraduate, a Research Postgraduate student or a Taught Postgraduate student. From this, I created a Mandatory Or relation to the three possible student types. Within both Research Postgraduate student and Taught Postgraduate student, there is an attribute of PreviousStudy, to represent what studies they have completed prior in other universities. **Attributes**: - StudentID: A primary key, which is a unique number, this will be in all relation the student entity creates, to identify it. - Fname: The first name of the student. I have chosen separate name attributes as you may not want to collect their full name. - Lname: The last name of the student. - Email: This is the university email that they are assigned, so that the department and university can communicate with them. - HomeAddress: The students home address before university. - UniversityAddress: The students address while at university. So that the university can send them mail. - PhoneNumber: The students phone number, so that the university and department can have direct contact with the student if needed. - Age: The age of the student so that the department can identify what percentage of students are mature students. **Relationships**: - Enrols - Tutors - Takes - Research Postgraduate students have the relation ResearchProject.
Programs	This is an entity that contains all the details about programs that a department of computer science runs. It has relationships with other entities such as the modules it offers, and which students are enrolled on. Thus, allowing the department of computer science to manage each program and its associated data. I have used a composite key of ProgramName and Award to identify each program in the program's entity. This is because the program name alone is not unique. Upon researching programs run by the department of computer science, there are many programs that share the same name, the only attribute that differentiates them is their award. Thus, when these two attributes are combined, they create a unique key. **Attributes**: - ProgramName {PK}: this is the name of the program that the department offers. - UCAScode: this is the UCAS code that the university uses so that students can apply to the program. - AdmssionRequirments: This is the entry requirements for the program. - Award {PK}: what type of degree the program awards the student. **Relationships**: - Enrols - Offers - HasModules
Modules	For each program that a department offers there are modules that makeup them. This is an entity that contains all the details of each module along with the relations to other data that make up this module, such as the module lead and staff who teach the module. As detailed in the specification modules offered can be either optional or mandatory, I have used generalisation/specialisation to create two relations that are {Mandatory, Or}, OptionalModule and MandatoryModule. Therefore, a module relation tuple can be either optional or mandatory. **Attributes**: - ModuleID: This is the primary key of my module entity. I have used a unique code rather than the module name as the module name may not be unique. - ModuleName: This is the name of the module. - Credits: What credits the module award the student. - Non-condonable: Whether the module is condonable to not. I have used a Boolean datatype for this. **Relationships**: - HasModules - Requires - Teaches - Leads
Staff	Within a department there are staff. I have created a staff entity that will represent each member of staff in the department of computer science. Generalisation/specialisation has been used multiple times within the staff entity. In the specification, it states that "generally" a member of staff falls into either being Research Staff, ProfessionalServicesStaff or Academic staff. I have made this generalisation/specialisation

	an optional or, as it is a general description, so staff do not have to 100% fall into these categories. Within both Research staff and Academic staff, I have included more generalisation/specialisation that is both mandatory or types, for the other categories the staff types can fall into. **Attributes**: - StaffID: This is the primary key that identifies the staff member, it is a unique code as their name may not be unique in the department of computer science. - Fname: The first name of the staff member. - Lname: The last name of the staff member - Email: The university email of the staff member, so that the department of computer science can communicate with its staff members - Address: The address of the staff member so that the department can send them documents and manage where they are living. - Salary: The salary of the staff member so that the department knows how much they are paying this staff member and if this is the correct amount. **Relationships**: - HasStaff - LeadsDep
Academic Staff	As described in the staff entity there are categories of staff, one of which is academic staff. This entity is used to store the details of the academic staff in the department. This entity included more generalisation/specialisation to the 5 different types of academic staff that exist in the department, this is using a mandatory or relation. **Attributes**: - AcademicID: This is a candidate key that can also uniquely identify the member of staff. **Relationships**: - Teaches - LeadMod - Organises - Tutors
ResearchGroup	As shown in the specification the department has research groups. To track research within the department I have created a ResearchGroup entity that contains all information about each research instance and the research staff that are involved in it. The department can track and allocate funding to groups with this entity. **Attributes**: - researchGroupName: this is the primary key of the research group. I have chosen this as each research group is assumed to have a unique name - research: this is a descriptor attribute that describes what the research is that they are carrying out **Relationships**: - JoinsGroup - Researches
Examination	The department requires the tracking of education within its department. This entity is used to contain all details about the exams that are run in the department. It is joined to a complex relationship that links it to both a mark, student, and module. The department will be able to see all the exams that a module runs, as well as the exams that a student has taken and the marks that they achieved. I have decided to use specialisation in the Examination relation with two relations that have a mandatory or to Examination. These are the Exam and Coursework. As detailed in the specification, a module examination can be both either Exams or Coursework, therefore an examination relation can be specialized to both or either of the two. **Attributes**: - ExaminationID: The primary key of an examination. - TotalMarks: The total marks available for that examination - Exam: o Date: Date for the examination o Duration: duration time the exam will be for o Location: where the exam will take place - Coursework: o DateSet: The date the coursework was set o DateDue The date the coursework is due **Relationships**: - Takes - Organises
ResearchProject	The departments research groups and PHD students have research projects. Therefore, for the university to manage these projects I have created an entity called ResearchGroup, to contain and represent each research project that the department and its students undertake. **Attributes**: - Topic - the primary key for the research project - ResearchDescription - A description of the project and its research. **Relationships**: - ResearchProject - Researches

Relationship	Description
HasDepartments	A faculty has departments within it. This is a relationship between the faculty and its departments entities, associating departments with a faculty. A binary one-to-many relationship, where a faculty can have many departments, and a department has one faculty.
Enrols	This is a relationship between the student's entity and the program's entity. The relationship is from students to programs. Its cardinality is a binary many-to-one relationship; where one student can only enrol on 1 program, but a program can have multiple students enrolled. From the relationship there are attributes made, these are enrolment type, ExpectedGradDate and DegreeClass. I have included these attributes, as the department of computer science in managing its students will want to know if their students are full-time or part-time; as well as when a group of students will graduate so that student numbers can be managed.

Offers	The department of computer science offers programs as shown in the program list provided. Offers is a relationship between the Departments entity and the Program entity. The relationship is from department to program. Its cardinality is a binary one-to-many relationship. A department can offer 1 or more programs, but a program can only be offered by one department.
HasStaff	In the specification, a list of staff in the computer science department is provided. This is a relationship between the Staff entity and the Departments entity, so that staff can be associated with a department. The relationship is from departments to staff. Its cardinality is a binary one-to-many relationship; a department can have many staff members, but a staff member only has one department.
LeadsDep	Each department has a lead. The lead is a member of staff from the staff entity. This is a relationship between department and staff and is from staff to department. Its cardinality is a binary one-to-one relationship, there is only one staff member who leads a department. I have made the relationship optional as not all staff lead a department.
LeadsMod	As described in the specification each module has a leader who organises the module's teachings and assessment activities. This is a relationship between AcademicStaff and Modules entities, from AcademicStaff to Modules. Its cardinality is a binary one-to-one relationship, as there can only be one lead for a module and a module can only have one lead. Otherwise, staff will have too large of a workload. I have made the relationship optional as not all academic staff members are module leaders.
Requires	Upon researching modules in the computer science department, I found that some modules require other modules to have been completed for a student to take a module. The relationship is a recursive relationship from module entity to module entity. This is a binary many-to-many relationship as a module could require many modules to have been completed. But the relationships bound start from 0 as a module may not require any modules to have been taken.
HasModules	The specification describes how programs offered by the department contain many modules within them. Upon further research, some of these modules are also shared between several programs, such as Computer Science BSc and Computer Science with Mathematics BSc share many modules. This is a relationship between the Programs entity and the Modules entity. I have assumed some students enrolled on a program will be PhD students, and as such, they will have no modules to complete; I have come to this conclusion through my research on the department. Therefore, a program may have 0 modules. The relationship goes from programs to modules. Its cardinality is a binary many-to-many relationship that starts from 1; a program can have 0 to many modules and a module can be on 1 to many programs. I have taken the assumption that a module will always be a part of at least one program, otherwise, the module isn't able to be taught and taken by students.
Teaches	As described in the specification a module can be taught by one or more academic staff. To show the involvement of an academic staff member in a module, I have made a relationship Teaches between them. The relationship is between the module's entity and AcademicStaff entity. The relationship is going from Academic staff to modules. Its cardinality is a binary many-to-many relationship, but I have assumed an academic staff member may teach 0..* modules, as they may not be teaching that academic year. But a module will have 1..* teachers, as it will always need at least one teacher to be run.
Tutors	The specification describes that a student is assigned an academic staff member upon enrolment. To show this, I have made the relationship Tutors. This is a relationship between the AcademicStaff and Students entities, going from AcademicStaff to Students. Its cardinality is a binary zero-to-many relationship. A staff member can be the academic tutor of zero to many students, but a student can only have one academic tutor. I have assumed that all academic staff are also academic tutors. The department of computer science will be able to use this to track which students are with which Academic Staff member. They will also be able to manage tutor groups if an academic tutor leaves the department and there needs to be a reassigning.
JoinsGroup	The specification describes that there are research staff within the department. These staff members are part of research groups. This is a relation that links research staff to a research group so that the university can track what staff members are in what and the research that they are completing. This is a relation between the ResearchGroup entity and the ResearchStaff entity. The relationship is from ResearchStaff to ResearchGroup entities. Its cardinality is binary many-to-many. The bounds for each are 1..*, as a research staff member may be a part of 1 to many groups, and a research staff can have 1 to many research staff members.
Takes	For the department of computer science to track education in the department, it must be able to track the performance of students enrolled on its programs. I have used a ternary complex relation to join the student's entity, Examination Entity and Module Entity. The Takes relationship creates an attribute from it called Mark, this is the mark that the student achieves for an Examination; as well as the mark that the student gets for the module overall. The cardinality of students to modules is many-to-many as a student can take many modules and a module can be taken by many students. Examination has the cardinality of 1..* as there could be many examinations for both a student and a module. For example, a student will have exams for multiple modules, and a module could have many exams, such as an exam, resit exam and coursework. The department of computer science can use this relation to track the performance of each module and the student's academic performance in the course. I have assumed that students can take optional modules during their studies. As programs offer optional modules (shown in spec). These modules are unique to the student and may not be listed on the student's program. Therefore, the take relation between student and module, allows the department to manage and view each student's allocated modules, regardless of the module being included within a program's module list. The relation between students and modules also allows the department to view each student's module list without the need to refer to a program relation, allowing for a smoother management experience. An assumption that I have made is that a module has the potential for resit papers, as not all students pass examinations. Therefore, assuming this is true, a module will have a relationship to Examination of 1..*. As there could be 1 too many exams created for a module. Coupled with this, I have also assumed that past papers for a module are included in the relation to examination, as the department may want to keep and manage all past papers for teaching and student use. Thus, further supporting my decision for a one-to-many relationship.
Organises	As described in the specification each module has a module leader. It states how a module leader is responsible for organising both the module and the examinations for the module. I have assumed through the relation LeadsMod, that this encapsulates the organising and running of the module's teachings, however, this does not include the organising of exams. This relation has the cardinality of a binary one-to-many relationship from Academic Staff to Examination entities. An academic staff member can organise zero to many exams for one module, but an examination can only be organised by one module lead. Therefore, the department can now manage and view which academic staff members are organising which examinations.
ResearchProject	PhD students are research students, and therefore will need to be assigned to a research project for their qualification. Therefore, I have made a relationship between the relations Research Project and Research Postgraduate student, so that the department can manage and view all students and their research topics. This relationship has the cardinality, Binary many-to-one from Research Postgraduate student entity to Research Project entity. The student can only have one research project, and a Research Project could have one too many research students.
Researches	Research Groups as shown through my research into the department can have many research projects carried out. Therefore, I have created a relationship between Research Project and the Research Group, to represent the many projects a group undertake. The cardinality of this relationship is a binary many-to-many relationship, from the ResearchGroup entity to the ResearchProject entity. A project could have many groups researching this topic, and a research group as described above can have many research projects.

700037354 | ECM2419 Datebase Theory and Design Cousework | Part 2 Normalisation

Staff Number	Staff Name	Position	Year of Hire	Module Number	Research Group	Group Lead	Module Name	Term	Number Of Times
35	Mark	Lecturer	2018	ECM1400 ECM2418 ECM2433	ML&CV	Sarah	Programming Computer Languages The C Family	1 1 2	1 1 1
11	Jennifer	Lecturer	2019	ECM3423 ECM1400	CS	Peter	Computer Graphics Programming	1 1	2 1
23	Mat	Professor	2014	ECM3408 ECM3423 ECM1400	ML&CV	Sarah	Enterprise Computing Computer Graphics Programming	2 1 1	3 2 2
36	Bob	Associate Professor	2016	ECM3408 ECM2433 ECM3423 ECM2418	HPC	Jack	Enterprise Computing The C Family Computer Graphics Computer Languages	2 2 1 1	1 3 2 4

StaffTeachings(StaffNumber,Staff Name,Position,YearOfHire,ModuleNumber,Research Group,Group Lead,ModuleName,Term,NumberOfTimes)

1st Normal Form

1st Normal Form Schema:
StaffTeachings(StaffNumber,Staff Name,Position,YearOfHire,ModuleNumber,Research Group,Group Lead,ModuleName,Term,NumberOfTimes)

Staff Number	Staff Name	Position	Year of Hire	Module Number	Research Group	Group Lead	Module Name	Term	Number Of Times
35	Mark	Lecturer	2018	ECM1400	ML&CV	Sarah	Programming	1	1
35	Mark	Lecturer	2018	ECM2418	ML&CV	Sarah	Computer Languages	1	1
35	Mark	Lecturer	2018	ECM2433	ML&CV	Sarah	The C Family	2	1
11	Jennifer	Lecturer	2019	ECM3423	CS	Peter	Computer Graphics	1	2
11	Jennifer	Lecturer	2019	ECM1400	CS	Peter	Programming	1	1
23	Mat	Professor	2014	ECM3408	ML&CV	Sarah	Enterprise Computing	2	3
23	Mat	Professor	2014	ECM3423	ML&CV	Sarah	Computer Graphics	1	2
23	Mat	Professor	2014	ECM1400	ML&CV	Sarah	Programming	1	2
36	Bob	Associate Professor	2016	ECM3408	HPC	Jack	Enterprise Computing	2	1
36	Bob	Associate Professor	2016	ECM2433	HPC	Jack	The C Family	2	3
36	Bob	Associate Professor	2016	ECM3423	HPC	Jack	Computer Graphics	1	2
36	Bob	Associate Professor	2016	ECM2418	HPC	Jack	Computer Languages	1	4

2nd Normal Form

PD1 : Staff Number -> Staff Name, Positon, Year of Hire, Research Group, Group lead
PD2 : Module Number -> Module Name, Term
FD1: Staff Number, Module Number -> Number of Times

2nd Normal Form Schema:
StaffTeachings(*Staff Number*, *Module Number*, Number of Times), Staff Number and ModuleNumber are a composite primary key
Modules(Module Number, Module Name, Term)
StaffDetails(Staff Number, Staff Name, Position, Year of Hire, Research Group, Group Lead)

Staff Number	Staff Name	Position	Year of Hire	Research Group	Group Lead
35	Mark	Lecturer	2018	ML&CV	Sarah
11	Jennifer	Lecturer	2019	CS	Peter
23	Mat	Professor	2014	ML&CV	Sarah
36	Bob	Associate Professor	2016	HPC	Jack

Staff Number	Module Number	Number of Times
35	ECM1400	1
35	ECM2418	1

Module Number	Module Name	Term
ECM1400	rogrammir	1
ECM2418	Computer Languages	1

35	ECM2433	1
11	ECM3423	2
11	ECM1400	1
23	ECM3408	3
23	ECM3423	2
23	ECM1400	2
36	ECM3408	1
36	ECM2433	3
36	ECM3423	2
36	ECM2418	4

ECM2433	The C Family	2
ECM3423	Computer Graphics	1
ECM3408	Enterprise Computing	2

3rd Normal Form

TD1 : Staff Number -> Research Group -> Group Lead

3rd Normal Form Schema:
StaffTeachings(*Staff Number* , *Module Number* , Number of Times), Staff Number and ModuleNumber are a composite primary key
StaffDetails(Staff Number, Staff Name, Position, Year of Hire, *Research Group*) , Staff Number is the primary key
ResearchGroup(ResearchGroup, Group Lead)
Modules(Module Number, Module Name, Term)

Staff Number	Staff Name	Position	Year of Hire	Research Group
35	Mark	Lecturer	2018	ML&CV
11	Jennifer	Lecturer	2019	CS
23	Mat	Professor	2014	ML&CV
36	Bob	Associate Professor	2016	HPC

Research Group	Group Lead
ML&CV	Sarah
CS	Peter
HPC	Jack

Staff Number	Module Number	Number of Times
35	ECM1400	1
35	ECM2418	1
35	ECM2433	1
11	ECM3423	2
11	ECM1400	1
23	ECM3408	3
23	ECM3423	2
23	ECM1400	2
36	ECM3408	1
36	ECM2433	3
36	ECM3423	2
36	ECM2418	4

Module Number	Module Name	Term
ECM1400	Programmi	1
ECM2418	Computer Languages	1
ECM2433	The C Family	2
ECM3423	Computer Graphics	1
ECM3408	Enterprise Computing	2

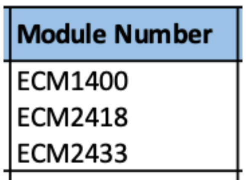
Normalisation - Is the process of producing a set of relations that have no redundancies and remove the potential for insertion, modification, and deletion anomalies.

Below is my reasoning for the decisions I have made in my relational model and schema section.

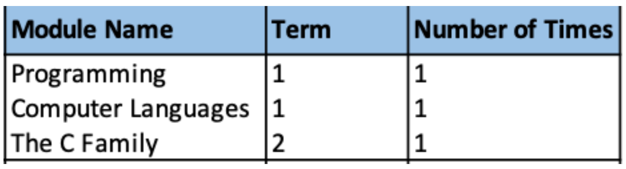
<u>Reasoning and Explanation | Normalisation</u>

1st Normal Form:

- First normal form requires there to be no repeating groups in attributes.
- In the initial relation the attributes module number, Module Name, Term, and Number of Times have repeating data in each tuple that uses that attribute. This is shown below.



Module Number
ECM1400 ECM2418 ECM2433



Module Name	Term	Number of Times
Programming Computer Languages The C Family	1 1 2	1 1 1

- To make the relation 1st normal form we must duplicate each tuple for each row in the repeated data group.
- This is shown in my answer for 1st normal form as there are multiple tuples for each staff member, each representing the different modules that they teach.
  - StaffTeachings (Staff Number, Staff Name, Position, Year of Hire, Module Number, Research Group, Group Lead, Module Name, Term, Number of Times)

2nd Normal Form:

- Second normal form requires there to be no partial dependencies in the relations.
- In my answer to 1st Normal form there are two partial dependencies:
  - Staff Number -> Staff Name, Position, Year of Hire, Research Group, Group lead
    - The primary key (attribute) Staff Number is causing partial dependencies for Staff Name, Position, Year of Hire, Research Group, and Group.
    - The staff number determines staff name, position, year of hire as these are all specific to the staff member in question. Also group name and group lead are dependent on staff number as, in the specification it describes that each academic staff member is part of a research group, and each group has a group lead. Thus, the staff number determines the research group as each member of staff has been allocated a research group.
  - Module Number -> Module Name, Term
    - The foreign key Module Number is causing a partial dependency on Module Name and Term. Each module number determines a module name and the term it is taught.
    - I have assumed that the attribute term is not unique, meaning a module is only taught during 1 term and not in multiple.
- There is also a fully functional dependency of Staff Number, Module Number -> Number of Times
  - This is not a partial dependency, as if you remove a value from A (A->B) then the dependency will no longer hold i.e., Staff Number alone cannot determine the Number of Times.
  - This dependency is to be contained within its own relation called StaffTeachings and will uniquely identify each staff and their teachings.
- Therefore, this requires the original relation to be split into three new schemas, StaffTeachings, Modules and StaffDetails, so that the details of staff members, staff teachings and modules can be contained in their own relation and protected from any anomalies.
  - StaffTeachings (Staff Number, Module Number, Number of Times)
    - This relation contains all teaching for staff for the academic year as well as the number of times they will be teaching a module
    - Staff Number and Module Number are a composite primary key in the relation, along with being foreign keys.
    - It uses the foreign key module number to link a module to a member of staff. Linking to the relation modules.

        - This relation contains all listings for staff and their teachings, each tuple is an instance of a staff member and a module. The relation may contain staff multiple times if they teach more than 1 module, but each instance will be contained within its own tuple.
    - Modules (Module Number, Module Name, Term)
        - This relation contains all details about modules that are taught in the department, including the name and term that they occur in. Module number is the primary key. These are attributes that are partially dependent on a module number
        - This relation allows for one central location for the information about modules ran by the department of computer science. Each module is defined once in this one location and in no other relation. Therefore, it can be managed and updated in one location removing all risk of data anomalies.
    - StaffDetails (Staff Number, Staff Name, Position, Year of Hire, Research Group, Group Lead)
        - The new relation StaffDetails contains all information about staff in the department, meaning all details for staff are contained within only one relation and can then be accessed for use elsewhere (StaffTeachings) using the foreign key staff number. They are not defined within any other relations other than for foreign key use.
        - This relation contains all attributes that are partially dependent on a staff member
        - This relation allows one singular relation where a member of staff can be updated and managed. Removing all risk of anomalies in staff details within the relations.
        - Staff Number is the primary key

3rd Normal Form:

- Third normal form requires all transitive dependencies from the relation.
- In my answer to 2nd Normal form there is one transitive dependency. This is in StaffDetails.
    - StaffDetails has the transitive dependency of Staff Number -> Research Group -> Group lead.
        - Staff number determines Research group as each academic staff member has one research group. Group lead is determined by the attribute Research Group as there is only 1 group lead to a research group. Thus, group lead is dependent on Staff Number.
- My solution to make the relation 3rd Normal form is, to convert the 3 relations into 4 relations.
    - StaffTeachings (Staff Number, Module Number, Number of Times)
        - Described in 2NF above, no changes have been made to this relation.
    - StaffDetails (Staff Number, Staff Name, Position, Year of Hire, Research Group)
        - This relation contains all data about staff, this is both personal data and research group data. This relation provides a singular location for all details about staff members, allowing a central place to update and maintain staff. To access the data, you must use its foreign key staff number.
        - It uses the foreign key Research group (I have assumed that each research group has a unique name), to link a staff member to a research group that is contained in the relation research group.
    - ResearchGroup (ResearchGroup, Group Lead)
        - This relation contains all details about research groups in the department and which staff lead the groups.
        - This relation allows one central location for information about the research groups to be stored and maintained. Therefore, there will be no anomalies of data consistency as it is only created once and updated in one place. The relation is accessed with its foreign key research group.
    - Modules (Module Number, Module Name, Term)
        - Described in 2NF above, no changes have been made to this relation.

Therefore, the relations are now in 3rd normal for and normalised as there is no repeating groups, redundant data, or possibilities for anomalies to occur with the use of these relations.