Introduction

You have been approached by a University for the design and implementation of a relational database system that will provide information on the courses it offers, the academic departments that run the courses, the academic staff and the enrolled students. The system will be used mainly by the students and the academic staff.

The requirement collection and analysis phase of the database design process provided the following data requirements for the University Database system.

Requirements

Each department runs a number of courses. The university provides a set of modules used in different courses. Each course uses a number of modules but not every module is used. A course is assigned a unique course code and a module is identified by a unique module code. A module can be used in one course only, but can be studied by many students. In addition to the module code each module unique title, start date, end date, texts (books), and assessment scheme (i.e. coursework and exam marks percentages) are also stored.

Each course is managed by a member of academic staff, and each module is coordinated by a member of academic staff also. The database should also store each course unique title, and duration (in years).

A student can enrol in one course at a time. Once enrolled a student is assigned a unique matriculation number. To complete a course, each student must undertake and pass all the required modules in his/her course. This requires that the database store the performance (pass or fail) of each student in every module.

Additional data stored on each student includes student name (first and last), address (town, street, and post code), date-of-birth, sex, and financial loan. For emergency purposes the database stores the name (not composite), address (not composite), phone, and relationship of each student next-of-kin. None of the next-of-kin's attributes is unique. Assume that every next-of-kin is a next-of-kin of one student only.

Each department is managed by a member of academic staff. The database should record the date he/she started managing the department. Each department has a name, phone number, fax number, and location (e.g. E Block). Each department employs many members of academic staff.

A member of academic staff can be the leader (i.e. manager) of at most one course, but can be the coordinator of more than one module. A member of academic staff may not be assigned any of the above mentioned roles (coordinator, course leader, department manager). All members of academic staff teach modules. Every member of academic staff teaches one or more modules, and a module may be taught by more than one member of academic staff. The database should record the number of hours per week a member of academic staff spend teaching each module. Each member of academic staff is identified by a unique staff number. All members of staff and students have unique computer network user ID numbers. Additional data stored on each member of academic staff includes name (first and last), phone extension number, office number, sex, salary, post (lecturer, or senior lecturer, or Professor, etc.), qualifications, and address (not composite). A member of academic staff work for one department only.

Part 1 – Design the Database

1. Create an Entity–Relationship (ER) model of the data requirements for the University Database case study using the Chen notation.
2. Note: State any assumptions necessary to support your design. Derive relational schema from your ER model that represents the entities and relationships. Identify primary, alternate and foreign keys. Note: use the following notation to describe your relational schema, as shown in the example of a Staff relation given below.

Staff (staffNo, fName, lName, address, NIN, sex, DOB, deptNo)

i. **Primary Key** staffNo

ii. **Alternate Key** lName, DOB

iii. **Alternate Key NIN**

Foreign Key deptNo **references** Department(deptNo) **On Delete No Action On Update Cascade**

3. Use the technique of normalization to validate the structure of your relational schema. Demonstrate that each of your relations is in third normal form (3NF) by displaying the functional dependencies between attributes in each relation. Note, if any of your relations are not in 3NF, this may indicate that your ER model is structurally incorrect or that you have introduced errors in the process of deriving relations from your model.

Part 2 – Implement the Database

1. Create the tables for the University Database. Where appropriate set field and table properties. Ensure that referential integrity is established between related tables.
2. ~~Create customised forms for data entry.~~
3. Enter some test data (approximately 5 – 10 rows) into each table.

Part 3 – Query the Database

Before starting this section, please ensure that your tables contain sufficient data to enable you to test the query transactions described in the University Database case study.

1. Create and save the following query transactions:
   - **a.** List details of all departments located in E Block.
   - **b.** List title, start and end dates of all modules run in the PgDIT course.
   - **c.** List name, address, and salary for each female member of academic staff who manages a department.
   - **d.** List name, sex, and salary for each lecturer with a PhD degree.
   - **e.** List last name, post, and qualifications of all members of academic staff who are employed by CIS department.
   - **f.** List matriculation number, last name, and sex of all students who are studying 'multi-media' module. Order result alphabetically by last name.
   - **g.** List staff number, last name, sex, and post of all academic staff whose salary is greater than the average salary of all academic staff.
   - **h.** For each course with more than 10 students, list course title and the number of students (under an appropriate header).
   - **i.** List the number of female members of academic staff and the number of male members of academic staff employed by CIS department.

j. For each member of academic staff who spends more than 6 hours teaching any module list the member of academic staff last name, the module title and the number of hours.

k. For each department list the department name, and the number of female members of academic staff, and the number of male members of academic staff under appropriate headers (use a crosstab query).

2. Provide 5 additional examples of queries, which retrieve useful data from the University Database database. State the purpose of each query and attempt to use each example to demonstrate the breadth of your knowledge of SQL. Create stored procedures with these 5 queries.

Chen's ERD – Entity Representation Diagram Process
University Database System

1. Create an Entity–Relationship (ER) model of the data requirements for the University Database case study using the Chen notation.

Step 1* - *Identify Entities (List all potential entity types)

1. University
2. Department
3. Courses
4. Modules
5. Students
6. Enrolled Students
7. Teaching Hours
8. Books
9. Assessment
10. Coursework
11. Exams
12. Academic Staff
13. Performance
14. Next Of Kin
15. Job Roles
16. Computer User ID
17. Qualification

Step 2* - *Remove duplicate entities and entities you won't be using.

1. Department
2. Course
3. Module
4. Book
5. Assessment
6. Academic Staff
7. Qualification
8. Students
9. Next of Kin

Step 3* - *List the attributes of each entity

1. **Department**
    - Department ID
    - Department name
    - Department Manager
    - Phone Number
    - Fax Number
    - Location
2. **Course**
    - Course ID
    - Department ID
    - Course Manager
    - Course Title
    - Start Date
    - End Date

3. **Module**
    - Module ID
    - Course ID
    - Staff Coordinator
    - Module Title
    - Duration

4. **Book**
    - Book ID
    - Book Title
    - Author
    - Publication
    - Year

5. **Assessment**
    - Module ID
    - Assessment ID
    - Coursework result
    - Exam result

6. **Academic Staff**
    - Staff ID
    - Department ID
    - Start Date
    - First Name
    - Last Name
    - Gender
    - Address
    - Phone Number
    - Office Number
    - Salary

7. **Qualification**
    - Qualification ID
    - Staff ID
    - Educational Level
    - College Name
    - Location
    - Title

8. **Student**
    - Student ID
    - Course ID
    - First Name
    - Last Name
    - Gender
    - Address
    - Date of Birth
    - Phone Number
    - Email Address
    - GPA
    - Financial Loan

9. **Next of Kin**
    - Student ID
    - Name
    - Relationship
    - Phone Number

Step 4* - *Mark the primary keys and foreign key.

1. **Department**
    - Department ID **(Primary Key)**
    - Department name
    - Department Manager **(Foreign Key)**
    - Phone Number
    - Fax Number
    - Location

2. **Course**
    - Course ID **(Primary Key)**
    - Department ID **(Foreign Key)**
    - Course Manager **(Foreign Key)**
    - Course Title
    - Start Date
    - End Date

3. **Module**
    - Module ID **(Primary Key)**
    - Course ID **(Foreign Key)**
    - Staff Coordinator **(Foreign Key)**
    - Module Title
    - Duration

4. **Book**
    - Book ID **(Primary Key)**
    - Book Title
    - Author
    - Publication
    - Year

5. **Assessment**
    - Module ID **(Composite Primary Key)**
    - Assessment ID **(Composite Primary Key)**
    - Coursework result
    - Exams result

6. **Academic Staff**
    - Staff ID **(Primary Key)**
    - Department ID **(Foreign Key)**
    - Start Date
    - First Name
    - Last Name
    - Gender
    - Address
    - Phone Number
    - Office Number
    - Salary

7. **Qualification**
    - Qualification ID **(Primary Key)**
    - Staff ID **(Foreign Key)**
    - Title
    - Educational Level
    - College Name
    - Location

8. **Student**
    - Student ID **(Primary Key)**
    - Course ID **(Foreign Key)**
    - First Name
    - Last Name
    - Gender
    - Address
    - Date of Birth
    - Phone Number
    - Email Address
    - GPA
    - Financial Loan

9. **Next of Kin**
    - Student ID **(Composite Primary Key)**
    - Name **(Composite Primary Key)**
    - Relationship
    - Phone Number

***Step 5* Define relationships of Entities (Strong and Weak entities)**

Define entities of University Database system.

A **strong entity** always has the primary key.
A **weak entity** is an entity that cannot be uniquely identified by its attributes alone.

Strong entity

1. Department
2. Course
3. Module
4. Book
5. Academic Staff
6. Qualification
7. Student

Weak entity

1. Assessment
2. Next of Kin

Relationship assumptions of University Database system.

- Department runs a number of courses.
- Departments are managed by one academic staff.
- Department employs many academic staff.
- Courses contains a number of modules.
- Courses provides many books.
- Courses are managed by one academic staff.
- Courses are enrolled by many students.
- Modules can only be used in one course.
- Modules are studied by many students.
- Modules provides many books.
- Modules coordinated by academic staff
- Modules can be taught by more than one Academic Staff
- Module have one assessment.
- Books are involved with modules.
- Books are read by students.
- Academic Staff works in one department only.
- Academic Staff manages one course.
- Academic Staff works within a course.
- Academic Staffs coordinates modules
- Academic Staffs teaches many students.

- Academic Staffs can have many Qualifications.
- Students are involved with one department.
- Students are enrolled one course at a time.
- Students studies many modules.
- Students have to complete many assessments.
- Students have one next of kin.
- Students have many academic staff lecturers.
- Students read many books

Step 6 Describe the cardinality of the relationships.

(M: N) - Many to Many.
(M: 1) - Many to One.
(1: M) - One to Many.
(1: 1) - One to One.

- Department runs a number of courses **(1: M)**
- Department are managed by one academic staff. **(1: 1)**
- Department employs many academic staff. **(1: M)**
- Course contains many modules. **(1: M)**
- Courses provides books. **(M: N)**
- Course are managed by one academic staff.**(1: 1)**
- Courses are enrolled by many students. **(1: M)**
- Courses contains a number of modules. **(1: M)**
- Modules are studied by many students. **(M: N)**
- Modules can only be used in one course **(M: 1)**
- Modules provides many books. **(1: M)**
- Modules coordinated by academic staff.**(M: N)**
- Modules can be taught by more than one Academic Staff. **(M: N)**
- Module have one assessment. **(1: 1)**
- Books are involved with modules. **(M: N)**
- Books are read by students. **(M: N)**
- Academic Staff works in one department only.**(M: 1)**
- Academic Staff manages one course.**(1: 1)**
- Academic Staff works within a course.**(M: 1)**
- Academic Staffs coordinates modules**(M: M)**
- Academic Staffs teaches many students. **(M: M)**
- Academic Staffs can have many Qualifications. **(1: M)**
- Students are involved with one department. **(M: 1)**
- Students are enrolled one course at a time. **(M: 1)**
- Students studies many modules. **(M: M)**
- Students have to complete many assessments. **(M: M)**
- Students have one next of kin. **(1: 1)**
- Students have many academic staff lecturers. **(M: M)**
- Students read many books. **(M: M)**

Step 7 ***Remove redundant relationships if necessary (Define relationships again)***

- **[Department]----------[Runs]----------[Courses]** **(1: M)**
- **[Department]----------[Employs]---------- [Academic Staff]** **(1: M)**
- **[Course]----------[Contains]----------[Modules]** **(1: M)**
- **[Module]----------[Provides]----------[Book]** **(M: N)**
- **[Module]----------[Taught]----------[Academic Staff]** **(M: N)**
- **[Module]----------[Have]----------[Assessments]** **(1: 1)**
- **[Academic Staff]------[Works]----------[Course]** **(M: 1)**
- **[Academic Staff]------[Have]----------[Qualifications]** **(1: M)**
- **[Student]----------[Enrolled]----------[Course]** **(M: 1)**
- **[Student]----------[Studies]----------[Module]** **(M: N)**
- **[Student]----------[Have]----------[Next of Kin]** **(1: 1)**

Step 8 ***Turn many to many relationships into 1-M and M-1.***

[Module]----------[Taught]----------[Academic Staff] (M: N)

- **[Module]----------[Contains]----------[Staff_Module]** **(1: M)**
- **[Academic Staff]------[Contains]----------[Staff_Module]** **(1: M)**

[Module]----------[Provides]----------[Book] **(M: N)**

- **[Module]---------- [Contains]----------[Book_Module]** **(1: M)**
- **[Book]----------[Contains]----------[Book_Module]** **(1: M)**

[Student]----------[Studies]----------[Module] (M: M)

- **[Student]----------[Contains]----------[Student_Module]** **(1: M)**
- **[Module]----------[Contains]----------[Student_Module]** **(1: M)**

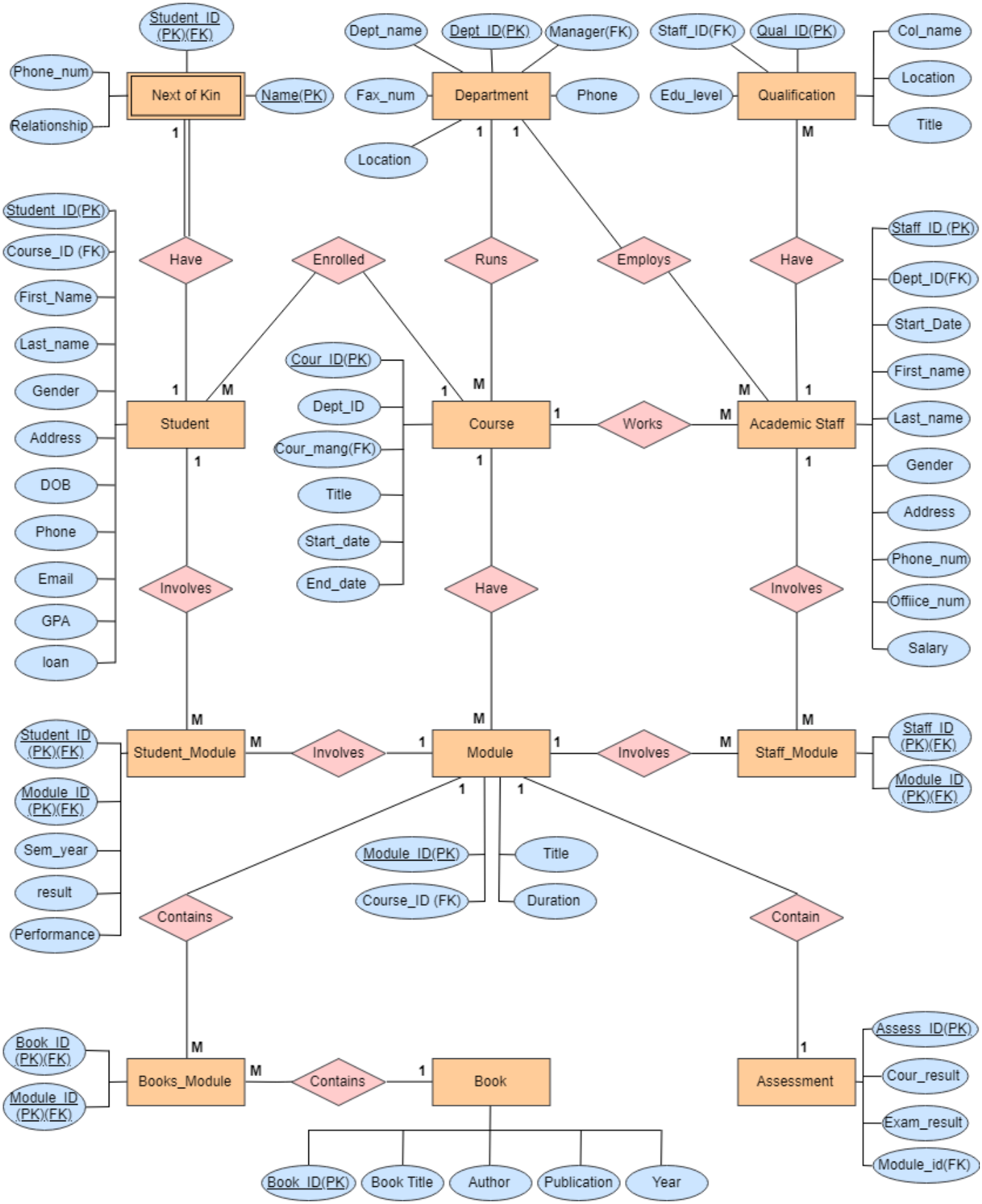
1. **Module**
    - Module ID **(Primary Key)**
    - Course ID **(Foreign Key)**
    - Module Title
    - ~~Staff Coordinator **(Foreign Key)**~~
    - ~~Books **(Foreign Key)**~~
    - Duration

3. **Staff_Module**
    - Staff_ID **(Foreign Key)**
    - Module_ID **(Foreign Key)**

5. **Book_module**(book_id,mod_id)
    - book_id **(Foreign Key)**
    - module_id **(Foreign Key)**

6. **Student_Module**
    - Student_ID **(Composite Primary Key)**
    - Module_ID **(Composite Primary Key)**
    - Semester
    - Result
    - Performance

***Step 9* Combine into single diagram. (All entities and the relationships between them should be combine.**



2. Derive relational schema from your ER model that represents the entities and relationships. Identify primary, alternate and foreign keys.

1. **department**(dept_id, dept_name, phone, fax_num, dept_location, dept_manager)
    i. **Primary Key** dept_id
    ii. **Alternate Key** dept_name, dept_phone, dept_fax, dept_location
    iii. **Foreign Key** dept_manager

   Foreign Key dept_manager **references** academic_staff(staff_id) **On Delete Set Null**
   On Update Cascade

2. **course**(course_id, title, start_date, end_date, course_manager, dept_id)
    i. **Primary Key** course_id
    ii. **Alternate Key** course_manager, title, start_date, end_date
    iii. **Foreign Key** course_manager dept_id

   Foreign Key course_manager **references** academic_staff(staff_id)**On Delete Set Null**
   On Update Cascade
   Foreign Key dept_id **references** department (dept_id) **On Delete Set Null**
   On Update Cascade

3. **module**(mod_id, title ,duration, course_id)
    i. **Primary Key** mod_id
    ii. **Alternate Key** title ,duration
    iii. **Foreign Key** course_id

   Foreign Key course_id **references** course (course_id) **On Delete Set Null**
   On Update Cascade

4. **book**(book_id, book_title, author, publication, year)
    i. **Primary Key** book_id
    ii. **Alternate Key** book_title, author, publication, year

5. **academic_staff**(staff_id ,first_name ,last_name ,gender ,address ,phone_num ,office_num , start_date, salary department)
    i. **Primary Key** staff_id
    ii. **Alternate Key** first_name ,last_name ,gender ,address ,phone_num ,office_num , start_num, salary
    iii. **Foreign Key** department

   Foreign Key department **references** department (dept_id) **On Delete Set Null**
   On Update Cascade.

6. **qualification**(qual_id, title, edu_level, college_name, location, staff_id)
    i. **Primary Key** qual_id
    ii. **Alternate Key** title ,edu_level, college_name, location
    iii. **Foreign Key** staff_id

   Foreign Key staff_id **references** academic_staff(staff_id) **On Delete Set Null**
   On Update Cascade

7. **student**(student_id, first_name, last_name, gender, address, dob, phone, email, gpa, loan, course_id)

    i. **Primary Key** student_id
    ii. **Alternate Key** first_name, last_name, gender, address, dob, phone, email, gpa, loan
    iii. **Foreign Key** course_id

    Foreign Key course_id **references** course(course_id) **On Delete Set Null**
    On Update Cascade

8. **assessment**(assess_id,coursework_result ,exams_result, module_id)
    i. **Primary Key** assess_id, module_id
    ii. **Alternate Key** coursework_result ,exams_result
    iii. **Foreign Key** module_id

    Foreign Key module_id **references** module(module_id) **On Delete Cascade**
    On Update Cascade

9. **next_of_kin**(student_id, name, relationship, phone_num)
    i. **Primary Key** student_id, name
    ii. **Alternate Key** relationship, phone_num
    iii. **Foreign Key** student_id

    Foreign Key student_id **references** student(student_id) **On Delete Cascade**
    On Update Cascade

10. **staff_module**(staff_id, module_id)
    i. **Primary Key** staff_id, module_id,
    ii. **Alternate Key**
    iii. **Foreign Key** staff_id, module_id

    Foreign Key staff_id **references** academic_staff ,(staff_id)**On Delete Cascade**
    On Update Cascade
    Foreign Key module_id **references** module(mod_id)**On Delete Cascade**
    On Update Cascade

11. **book_module**(book_id, module_id)
    iv. **Primary Key** book_id, module_id
    v. **Foreign Key** book_id, module_id

    Foreign Key book_id **references** book(book_id) **On Delete Cascade**
    On Update Cascade
    Foreign Key module_id **references** module(mod_id) **On Delete Cascade**
    On Update Cascade

12. **student_module**(student_id, module_id, semester, result, perform)
    vi. **Primary Key** student_id, module_id
    vii. **Alternate Key** semester_year, result, perform
    viii. **Foreign Key** student_id, module_id

    Foreign Key student_id **references** student(student_id) **On Delete Cascade**
    On Update Cascade
    Foreign Key module_id **references** module(mod_id)**On Delete Cascade**
    On Update Cascade

3. Use the technique of normalization to validate the structure of your relational schema. Demonstrate that each of your relations is in third normal form (3NF) by displaying the functional dependencies between attributes in each relation. Note, if any of your relations are not in 3NF, this may indicate that your ER model is structurally incorrect or that you have introduced errors in the process of deriving relations from your model.

Below are the relations that are already in Third Normal Form.

The text that are in **Orange** is the name of the relation.
The text that are in **Bold Black** is the primary key of the relation.
The text that are in **Bold Red** is the foreign key of the relation.

- **department**(**dept_id**, dept_name, phone, fax_num, dept_location, **dept_manager**)
- **module**(**mod_id**, title ,duration, **course_id**)
- **book**(**book_id**, book_title, author, publication, year)
- **student**(**student_id**, first_name, last_name, gender, address, dob, phone, email, gpa, loan, **course_id**)
- **next_of_kin**(**student_id**, **name**, relationship, phone_num)
  Primary Key student_id, name
- **staff_module**(**staff_id, module_id**, name)
  Primary Key staff_id, module_id,
- **book_module**(**book_id, module_id**)
  Primary Key book_id, module_id
- **student_module**(**student_id, module_id**, semester, result, perform)
  Primary Key student_id, module_id

Below are the relations that are already in Third Normal Form.

The text that are in **Orange** is the name of the relation.
The text that are in **Bold Black** is the primary key of the relation.
The text that are in **Bold Red** is the foreign key of the relation.

- **course**(**course_id**, title, start_date, end_date, **course_manager, dept_id**)
- **qualification**(**qual_id**, edu_level, college_name, location, title, **staff_id**)
- **assessment**(**assess_id**,coursework_result ,exams_result, **module_id**)
  Primary Key assess_id, module_id

Course (Normalization)

- UNF: **course**(**course_id**, title, start_date, end_date, **course_manager, dept_id**)
- 1NF: **course**(**course_id**, title, start_date, end_date, **course_manager, dept_id**)
- 2NF: **course**(**course_id**, title, **course_manager, dept_id**)
  course_date(**course_id**, start_date, end_date)
- 3NF: **course**(**course_id**, title, **course_manager, dept_id**)
  course_date(**course_id**, start_date, end_date)

Qualification *(Normalization)*

- UNF: **qualification**(**qual_id**, title, edu_level, college_name, location, **staff_id**)
- 1NF: **qualification**(**qual_id**, title, edu_level, college_name, location, **staff_id**)
- 2NF: **qualification**(**qual_id**, title, edu_level, college_name, location)
  staff_qualification (**qual_id**, **staff_id**)
- 3NF: **qualification**(**qual_id**, title, edu_level, college_name, location)
  staff_qualification (**qual_id**, **staff_id**)

Assessment (Normalization)

- UNF: **assessment**(**assess_id**,coursework_result ,exams_result, **module_id**)
  Primary Key assess_id, module_id
- 1NF: **assessment**(**assess_id**,coursework_result ,exams_result, **module_id**)
- 2NF: **assessment**(**assess_id**,coursework_result ,exams_result)
  module_assess(**module_id, assess_id**)
- 3NF: **assessment**(**assess_id**,coursework_result ,exams_result)
  module_assess(**module_id, assess_id**)

Academic_Staff (Normalization)

- UNF:**academic_staff**(**staff_id** ,first_name ,last_name ,gender ,address ,phone_num ,office_num , start_date, salary, **department**)
- 1NF:**academic_staff**(**staff_id** ,first_name ,last_name ,gender ,address ,phone_num ,office_num , start_date, salary, **department**)
- 2NF:**academic_staff**(**staff_id** ,first_name ,last_name ,gender ,address ,phone_num ,office_num , start_date, salary)
  department_staff(**dep_id, staff_id**)
- 3NF:**academic_staff**(**staff_id** ,first_name ,last_name ,gender ,address ,phone_num ,office_num , start_date, salary)
  department_staff(**dep_id, staff_id**)

Part 2 – Implement the Database

1. Create the tables for the University Database. Where appropriate set field and table properties. Ensure that referential integrity is established between related tables.

- **Create and Use University Database**

```
mysql> CREATE DATABASE university;
Query OK, 1 row affected (0.00 sec)

mysql> USE  university;
Database changed
```

- **department**(dept_id, dept_name, phone, fax_num, dept_location, dept_manager)

```
mysql> CREATE TABLE department(
    -> dept_id VARCHAR(25) NOT NULL,
    -> dept_name VARCHAR(35),
    -> phone VARCHAR(25),
    -> fax_num VARCHAR(25),
    -> dept_location VARCHAR(25),
    -> dept_manager INT,
    -> CONSTRAINT dept_id PRIMARY KEY(dept_id),
    -> CONSTRAINT dept_manager FOREIGN KEY(dept_manager)
    -> REFERENCES academic_staff(staff_id)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **course**(course_id, title, course_manager, dept_id)

```
mysql> CREATE TABLE course(
    -> course_id VARCHAR(25) NOT NULL,
    -> title VARCHAR(35),
    -> course_manager INT,
    -> dept_id VARCHAR(25),
    -> CONSTRAINT course_id PRIMARY KEY(course_id),
    -> CONSTRAINT course_manager FOREIGN KEY(course_manager)
    -> REFERENCES academic_staff(staff_id)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE,
    -> CONSTRAINT dept_id FOREIGN KEY(dept_id)
    -> REFERENCES department(dept_id)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **course_date**(course_id, start_date, end_date)

```
mysql> CREATE TABLE course_date(
    -> course_id VARCHAR(25) NOT NULL,
    -> start_date DATE,
    -> end_date DATE,
    -> CONSTRAINT course_id FOREIGN KEY(course_id)
    -> REFERENCES course(course_id)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **module**(mod_id, title ,duration, course_id)

```
mysql> CREATE TABLE module(
    -> mod_id VARCHAR(25) NOT NULL,
    -> title VARCHAR(35),
    -> duration INT,
    -> course_id VARCHAR(25) NOT NULL,
    -> CONSTRAINT mod_id PRIMARY KEY(mod_id),
    -> CONSTRAINT course FOREIGN KEY(course_id)
    -> REFERENCES course(course_id)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.02 sec)
```

- **book**(book_id, book_title, author, publication, year)

```
mysql> CREATE TABLE book(
    -> book_id INT NOT NULL,
    -> book_title VARCHAR(35),
    -> author VARCHAR(35),
    -> publication VARCHAR(25),
    -> year INT,
    -> CONSTRAINT mod_id PRIMARY KEY(book_id)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **academic_staff**(staff_id ,first_name ,last_name ,gender ,address ,phone_num ,office_num , start_date, salary)

```
mysql> CREATE TABLE academic_staff (
    -> staff_id INT NOT NULL,
    -> first_name VARCHAR(25),
    -> last_name VARCHAR(25),
    -> gender ENUM('M', 'F'),
    -> address VARCHAR(255),
    -> phone_num VARCHAR(25),
    -> office_num VARCHAR(25),
    -> start_date DATE,
    -> salary INT,
    -> CONSTRAINT staff_id PRIMARY KEY(staff_id)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

- **qualification**(qual_id, title, edu_level, college_name, location)

```
mysql> CREATE TABLE qualification(
    -> qual_id INT NOT NULL,
    -> edu_level VARCHAR(25),
    -> college_name VARCHAR(25),
    -> location VARCHAR(255),
    -> title VARCHAR(25),
    -> CONSTRAINT qual_id PRIMARY KEY(qual_id)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

- **staff_qualification** (staff_id, qual_id)

```
mysql> CREATE TABLE staff_qualification(
    -> staff_id INT NOT NULL,
    -> qual_id INT NOT NULL,
    -> CONSTRAINT staff FOREIGN KEY(staff_id)
    -> REFERENCES academic_staff(staff_id)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE,
    -> CONSTRAINT qual FOREIGN KEY(qual_id)
    -> REFERENCES qualification(qual_id)
    -> ON DELETE CASCADE
    -> ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **student**(student_id, first_name, last_name, gender, address, dob, phone, email, gpa, loan, course_id)

```
mysql> CREATE TABLE student(
    -> student_id INT NOT NULL,
    -> first_name VARCHAR(25),
    -> last_name VARCHAR(25),
    -> gender ENUM('M', 'F'),
    -> address VARCHAR(255),
    -> dob DATE,
    -> phone VARCHAR(25),
    -> email VARCHAR(50),
    -> gpa DECIMAL (21),
    -> loan INT,
    -> course_id VARCHAR(25),
    -> CONSTRAINT student_id PRIMARY KEY(student_id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE student
    ->   ADD CONSTRAINT
    ->   FOREIGN KEY (course_id)
    ->   REFERENCES course(course_id)
    ->   ON DELETE SET NULL
    ->   ON UPDATE CASCADE;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **assessment**(assess_id,coursework_result ,exams_result)

```
mysql> CREATE TABLE assessment(
    -> assess_id INT NOT NULL,
    -> coursework_result INT ,
    -> exams_result INT,
    -> PRIMARY KEY(assess_id)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **module_assess**(module_id, assess_id)

```
mysql> CREATE TABLE  module_assess(
    -> mod_id VARCHAR(25) NOT NULL,
    -> assess_id INT NOT NULL,
    -> CONSTRAINT module FOREIGN KEY(mod_id)
    -> REFERENCES module(mod_id)
    -> On Delete CASCADE
    -> ON UPDATE CASCADE,
    -> CONSTRAINT assess FOREIGN KEY(assess_id)
    -> REFERENCES assessment(assess_id)
    -> On Delete CASCADE
    -> ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **next_of_kin**(student_id, name, relationship, phone_num)

```
mysql> CREATE TABLE next_of_kin(
    -> fullname VARCHAR(50) NOT NULL,
    -> relationship VARCHAR(25),
    -> phone_num VARCHAR(25),
    -> student_id INT NOT NULL,
    -> PRIMARY KEY(fullname, student_id)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE next_of_kin
    ->   ADD CONSTRAINT
    ->   FOREIGN KEY(student_id)
    ->   REFERENCES student(student_id)
    ->   ON DELETE CASCADE
    ->   ON UPDATE CASCADE;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **staff_module**(staff_id, module_id)

```
mysql> CREATE TABLE staff_module(
    -> staff_id INT NOT NULL,
    -> module_id VARCHAR(25) NOT NULL,
    -> PRIMARY KEY(staff_id, module_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE staff_module
    ->   ADD FOREIGN KEY(staff_id)
    ->   REFERENCES academic_staff(staff_id)
    ->   ON DELETE CASCADE
    ->   ON UPDATE CASCADE;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE staff_module
    ->   ADD FOREIGN KEY(module_id)
    ->   REFERENCES module(mod_id)
    ->   ON DELETE CASCADE
    ->   ON UPDATE CASCADE;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **book_module**(book_id, module_id)

```
mysql> CREATE TABLE book_module(
    -> book_id INT NOT NULL,
    -> module_id VARCHAR(25) NOT NULL,
    -> CONSTRAINT book_id FOREIGN KEY(book_id)
    -> REFERENCES book(book_id)
    -> On Delete CASCADE
    -> ON UPDATE CASCADE,
    -> CONSTRAINT module_id FOREIGN KEY(module_id)
    -> REFERENCES module(mod_id)
    -> On Delete CASCADE
    -> ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- **student_module**(student_id, module_id, semester, result, perform)

```
mysql> CREATE TABLE student_module(
    -> student_id INT NOT NULL,
    -> module_id VARCHAR(25) NOT NULL,
    -> semester ENUM('1','2'),
    -> year INT,
    -> result INT,
    -> perform ENUM ('PASS','FAIL'),
    -> PRIMARY KEY(student_id, module_id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE student_module
    ->   ADD FOREIGN KEY(student_id)
    ->   REFERENCES student(student_id)
    ->   ON DELETE CASCADE
    ->   ON UPDATE CASCADE;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE student_module
    ->   ADD FOREIGN KEY(module_id)
    ->   REFERENCES module(mod_id)
    ->   ON DELETE CASCADE
    ->   ON UPDATE CASCADE;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **department_staff**(dept_id,staff_id)

```
mysql> CREATE TABLE department_staff(
    -> dept_id VARCHAR(25) NOT NULL,
    -> staff_id INT NOT NULL,
    -> PRIMARY KEY(dept_id, staff_id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> ALTER TABLE department_staff
    -> ADD FOREIGN KEY(dept_id)
    -> REFERENCES department(dept_id)
    -> On Delete CASCADE
    -> ON UPDATE CASCADE;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE department_staff
    ->     ADD FOREIGN KEY(staff_id)
    -> REFERENCES academic_staff(staff_id)
    -> On Delete CASCADE
    -> ON UPDATE CASCADE;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **Show University Tables**

```
mysql> SHOW TABLES;
+----------------------+
| Tables_in_university |
+----------------------+
| academic_staff       |
| assessment           |
| book                 |
| book_module          |
| course               |
| course_date          |
| department           |
| department_staff     |
| module               |
| module_assess        |
| next_of_kin          |
| qualification        |
| staff_module         |
| staff_qualification  |
| student              |
| student_module       |
+----------------------+
16 rows in set (0.00 sec)
```

~~2. Create customised forms for data entry.~~

3. **Enter some test data (approximately 5 – 10 rows) into each table.**

- **department**(dept_id, dept_name, phone, fax_num, dept_location, dept_manager)

```
mysql> SELECT * FROM department;
+---------+-----------+--------------+--------------+---------------+--------------+
| dept_id | dept_name | phone        | fax_num      | dept_location | dept_manager |
+---------+-----------+--------------+--------------+---------------+--------------+
| 100001  | CIS       | 519 137 6809 | 140-690-9484 | E Block       |       300001 |
| 100002  | ENG       | 338 955 9399 | 140-150-0352 | E Block       |       300002 |
| 100003  | ARC       | 824 899 3930 | 140-139-0195 | E Block       |       300003 |
| 100004  | ART       | 541 245 4137 | 140-578-2394 | A Block       |       300004 |
| 100005  | BCH       | 663 329 9562 | 140-536-9694 | B Block       |       300005 |
| 100006  | GEO       | 130 394 6553 | 140-952-7802 | C Block       |       300006 |
| 100007  | MGM       | 851 975 6290 | 140-862-3769 | C Block       |       300007 |
| 100008  | MUS       | 248 716 2470 | 140-415-1981 | A Block       |       300008 |
+---------+-----------+--------------+--------------+---------------+--------------+
8 rows in set (0.00 sec)
```

- **course**(course_id, title, course_manager, dept_id)

```
mysql> SELECT * FROM course;
+-----------+------------------+----------------+---------+
| course_id | title            | course_manager | dept_id |
+-----------+------------------+----------------+---------+
| 200001    | Computer Science |         300009 | 100001  |
| 200002    | Engineering      |         300010 | 100002  |
| 200003    | Architecture     |         300011 | 100003  |
| 200004    | Art              |         300012 | 100004  |
| 200005    | Business         |         300013 | 100005  |
| 200006    | Geography        |         300014 | 100006  |
| 200007    | Management       |         300015 | 100007  |
| 200008    | Music            |         300016 | 100008  |
+-----------+------------------+----------------+---------+
8 rows in set (0.00 sec)
```

- **course_date**(course_id, start_date, end_date)

```
mysql> SELECT * FROM course;
+-----------+------------------+----------------+---------+
| course_id | title            | course_manager | dept_id |
+-----------+------------------+----------------+---------+
| 200001    | Computer Science |         300009 | 100001  |
| 200002    | Engineering      |         300010 | 100002  |
| 200003    | Architecture     |         300011 | 100003  |
| 200004    | Art              |         300012 | 100004  |
| 200005    | Business         |         300013 | 100005  |
| 200006    | Geography        |         300014 | 100006  |
| 200007    | Management       |         300015 | 100007  |
| 200008    | Music            |         300016 | 100008  |
+-----------+------------------+----------------+---------+
8 rows in set (0.00 sec)
```

- **module**(mod_id, title ,duration, course_id)

```
mysql> SELECT * FROM module;
+--------+----------------------+----------+-----------+
| mod_id | title                | duration | course_id |
+--------+----------------------+----------+-----------+
| 500001 | Relational Database  |       80 | 200001    |
| 500002 | Java                 |       80 | 200001    |
| 500003 | Computer Network     |       60 | 200001    |
| 500004 | Web development      |      100 | 200001    |
| 500005 | Operating Systems    |       60 | 200001    |
| 500006 | Architecture         |      100 | 200003    |
| 500007 | Arch History         |       60 | 200003    |
| 500008 | Visual Communication |       80 | 200003    |
+--------+----------------------+----------+-----------+
8 rows in set (0.00 sec)
```

- **book**(book_id, book_title, author, publication, year)

```
mysql> SELECT * FROM book;
+---------+-------------------------------------+-------------------+-----------------+------+
| book_id | book_title                          | author            | publication     | year |
+---------+-------------------------------------+-------------------+-----------------+------+
|  600001 | Algorithms to Live                  | Tom Griffiths     | Pearson         | 1993 |
|  600002 | Heroes of the Computer Revolution   | Brian Christian   | ThomsonReuters  | 1983 |
|  600003 | The Soul of a New Machine           | Tracy Kidder      | Bertelsmann     | 2003 |
|  600004 | Superintelligence: Paths, Dangers,  | Nick Bostrom      | Grupo Planeta   | 2010 |
|  600005 |  The Hidden Language of Computer Ha | Steven Levy       | Wiley           | 2005 |
|  600006 |  The Second Machine Age:            | Andrew McAfee     | Springer Nature | 1999 |
|  600007 |  Thinking in Systems: A Primer      | Donella H. Meadows | Wiley           | 1995 |
|  600008 |  The Search: How Google and Its Riv | John Battelle     | ThomsonReuters  | 2011 |
+---------+-------------------------------------+-------------------+-----------------+------+
8 rows in set (0.00 sec)
```

- **academic_staff**(staff_id ,first_name ,last_name ,gender ,address ,phone_num ,office_num , start_date, salary)

```
mysql> SELECT * FROM academic_staff;
+----------+------------+---------------+--------+--------------------------+--------------+---------------+------------+--------+
| staff_id | first_name | last_name     | gender | address                  | phone_num    | office_num    | start_date | salary |
+----------+------------+---------------+--------+--------------------------+--------------+---------------+------------+--------+
|   300001 | Wesley     | Mansell       | M      | 5539 Hoepker Junction    | 948-656-3383 | (691) 3788519 | 2019-05-20 |  49473 |
|   300002 | Clevie     | Greenaway     | M      | 16245 Dayton Road        | 433-778-1463 | (375) 8685449 | 2019-02-16 |  43797 |
|   300003 | Magdalen   | Benthall      | F      | 91 Scott Avenue          | 138-322-2089 | (227) 1339420 | 2019-02-04 |  40068 |
|   300004 | Moore      | Biaggetti     | M      | 9 Vidon Crossing         | 706-557-5079 | (303) 2887165 | 2019-10-26 |  36935 |
|   300005 | Valeda     | Christofe     | F      | 81464 Glendale Pass      | 674-419-6130 | (151) 1242271 | 2019-11-15 |  34654 |
|   300006 | Ode        | Polye         | M      | 1517 Algoma Alley        | 418-112-1145 | (895) 7816915 | 2019-04-10 |  35022 |
|   300007 | Cly        | Gravenor      | M      | 81440 Hallows Terrace    | 942-288-9746 | (501) 5206513 | 2019-08-29 |  34704 |
|   300008 | Beatrix    | Buchan        | F      | 988 Pearson Park         | 373-204-2084 | (786) 9878587 | 2019-01-30 |  47515 |
|   300009 | Stuart     | Fragino       | M      | 602 Burning Wood Crossing | 754-400-0604 | (863) 3720070 | 2019-08-30 |  41407 |
|   300010 | Carly      | Postans       | M      | 6633 Kenwood Place       | 993-911-9210 | (574) 1672382 | 2019-08-02 |  40777 |
|   300011 | Elnore     | Tschierasche  | F      | 56083 Nobel Court        | 949-694-8939 | (582) 3249314 | 2019-12-09 |  30822 |
|   300012 | Gillian    | Ipwell        | F      | 22441 Roth Road          | 929-394-6307 | (701) 2611141 | 2019-02-22 |  41309 |
|   300013 | Sascha     | Panichelli    | F      | 67437 Northport Alley    | 312-268-8317 | (318) 2854251 | 2019-07-13 |  36665 |
|   300014 | Nevile     | Mocher        | M      | 482 Springview Way       | 212-772-6226 | (498) 6628288 | 2019-06-30 |  33248 |
|   300015 | Merrel     | MacDonogh     | M      | 049 Wayridge Center      | 737-559-9224 | (144) 1479212 | 2019-08-07 |  48651 |
|   300016 | Gardener   | Espinel       | M      | 86 Clyde Gallagher Plaza | 540-837-1949 | (859) 6245370 | 2019-12-02 |  40224 |
|   300017 | Coraline   | Mowsley       | F      | 89748 Haas Hill          | 310-284-2541 | (758) 2481282 | 2019-05-11 |  43198 |
|   300018 | Kristos    | Grzes         | M      | 1107 West Center         | 641-748-3777 | (564) 8730952 | 2019-04-17 |  36629 |
|   300019 | Becca      | Aleksandrikin | F      | 38 Shoshone Drive        | 413-571-5786 | (912) 4514697 | 2019-11-12 |  33208 |
|   300020 | Laureen    | Hilbourne     | F      | 0580 Eastlawn Alley      | 924-224-9953 | (143) 6580048 | 2019-06-07 |  35623 |
+----------+------------+---------------+--------+--------------------------+--------------+---------------+------------+--------+
20 rows in set (0.00 sec)
```

- **qualification**(qual_id, title, edu_level, college_name, location)

```
mysql> SELECT * FROM qualification;
+---------+--------------------------+-----------+-----------------------------------+-------------+
| qual_id | title                    | edu_level | college_name                      | location    |
+---------+--------------------------+-----------+-----------------------------------+-------------+
|  700001 | Computer Sciences        | PHD       |  University of Asia Pacific       | Japan       |
|  700002 | Artificial Intelligence  | PHD       |  Universidad Academia de Humanismo| Brazil      |
|  700003 | Applied Mathematics      | PHD       |  Zarka Private University         | Spain       |
|  700004 | Environmental Management | MSc       |  Kazak American University        | USA         |
|  700005 | Performing Arts - Music  | MSc       | Barat College                     | USA         |
|  700006 | Medicine                 | PHD       |  Université du Maine              | France      |
|  700007 | Sociolinguistics         | MSc       |  Islamic University in Uganda     | Uganda      |
|  700008 | Physical Geography       | PHD       |  Universidad Santa Paula          | Santa Paula |
|  700010 | Literature               | MSc       |  University of Geneva             | Switzerland |
|  700011 | Philosophy               | PHD       |  University of Manchester         | UK          |
|  700012 | Business Finance         | MSc       |  Moscow State University of Commerce | Russia   |
|  700013 | Business Analysis        | PHD       |  Universitas Komputer Indonesia   | Indonesia   |
|  700014 | Linnaean Taxonomy        | PHD       |  Western Michigan University      | USA         |
|  700015 | Curriculum And Instructio| PHD       | Hokusei Gakuen University         | Japan       |
+---------+--------------------------+-----------+-----------------------------------+-------------+
14 rows in set (0.00 sec)
```

- **staff_qualification** (staff_id, qual_id)

```
mysql> SELECT * FROM staff_qualification;
+----------+---------+
| staff_id | qual_id |
+----------+---------+
|   300001 |  700001 |
|   300009 |  700002 |
|   300002 |  700003 |
|   300003 |  700004 |
|   300004 |  700005 |
|   300006 |  700007 |
|   300008 |  700012 |
|   300009 |  700010 |
|   300010 |  700011 |
+----------+---------+
9 rows in set (0.00 sec)
```

- **student**(student_id, first_name, last_name, gender, address, dob, phone, email, gpa, loan, course_id)

```
mysql> SELECT * FROM student;
+------------+------------+------------+--------+----------------------+------------+--------------+-----------------------------+------+------+-----------+
| student_id | first_name | last_name  | gender | address              | dob        | phone        | email                       | gpa  | loan | course_id |
+------------+------------+------------+--------+----------------------+------------+--------------+-----------------------------+------+------+-----------+
|     800001 | Jaquelyn   | Anstis     | F      | 75518 Old Shore Point | 1985-12-17 | 365-432-0026 | janstis0@w3.org             |    3 | 8182 | 200001    |
|     800002 | Benedetto  | Tinmouth   | M      | 8 La Follette Pass   | 1981-12-17 | 382-732-7621 | btinmouth1@e-recht24.de     |    4 | 7296 | 200003    |
|     800003 | Cathe      | Jellybrand | F      | 0934 Moland Terrace  | 1983-04-26 | 779-333-6761 | cjellybrand2@uiuc.edu       |    3 | 9117 | 200001    |
|     800004 | Hildagard  | Vauter     | F      | 92311 Carpenter Alley | 1982-05-04 | 896-967-9243 | hvauter3@myspace.com        |    4 | 7659 | 200003    |
|     800005 | Tasia      | Sprague    | F      | 148 Summit Point     | 1980-12-23 | 432-206-5070 | tsprague4@paypal.com        |    4 | 8070 | 200001    |
|     800006 | Rice       | McInally   | M      | 96768 Ilene Alley    | 1981-07-12 | 208-827-3930 | rmcinally5@ustream.tv       |    3 | 7053 | 200003    |
|     800007 | Hersh      | Mcwhinney  | M      | 63 4th Point         | 1985-01-02 | 117-986-7270 | hmcwhinney6@upenn.edu       |    3 | 7323 | 200001    |
|     800008 | Natty      | Sharland   | F      | 836 Claremont Street | 1990-08-09 | 177-174-5862 | nsharland7@walmart.com      |    3 | 8286 | 200003    |
|     800009 | Aile       | Stollmeyer | F      | 26773 Helena Way     | 1983-04-24 | 778-179-3405 | astollmeyer8@toplist.cz     |    3 | 8845 | 200001    |
|     800010 | Debra      | Dearnaley  | F      | 4 Kim Terrace        | 1985-01-31 | 862-261-6583 | ddearnaley9@squidoo.com     |    3 | 7519 | 200003    |
|     800011 | Christy    | Simes      | F      | 562 Bultman Place    | 1983-06-05 | 341-908-1010 | csimesa@multiply.com        |    3 | 7179 | 200001    |
|     800012 | Elmo       | Crosetto   | M      | 59 Oak Valley Drive  | 1984-05-17 | 660-334-4471 | ecrosettob@mysql.com        |    4 | 8694 | 200003    |
|     800013 | Ephrayim   | Luesley    | M      | 950 Eagan Pass       | 1983-05-02 | 565-200-1432 | eluesleyc@wikia.com         |    4 | 9209 | 200001    |
|     800014 | Kirsti     | Elleyne    | F      | 93375 Huxley Street  | 1985-11-01 | 650-847-0462 | kelleyned@freewebs.com      |    3 | 7866 | 200003    |
|     800015 | Karl       | Jobbins    | M      | 1 Bonner Court       | 1981-02-20 | 294-109-1165 | kjobbinse@nytimes.com       |    3 | 8277 | 200001    |
|     800016 | Gottfried  | Perazzo    | M      | 878 Drewry Pass      | 1989-06-27 | 253-784-5079 | gperazzof@businesswire.com  |    4 | 8587 | 200003    |
|     800017 | Ozzy       | Curcher    | M      | 215 Scott Court      | 1985-10-03 | 864-392-7204 | ocurcherg@seattletimes.com  |    4 | 7990 | 200001    |
|     800018 | Friedrich  | Cambden    | M      | 06322 Tomscot Alley  | 1984-02-11 | 351-529-8998 | fcambdenh@1688.com          |    4 | 8987 | 200003    |
|     800019 | Jennine    | Bastable   | F      | 15 Straubel Center   | 1988-04-27 | 825-275-1782 | jbastablei@mapy.cz          |    3 | 7031 | 200001    |
|     800020 | Shel       | Rolance    | F      | 5892 John Wall Place | 1989-09-19 | 494-135-3028 | srolancej@cdbaby.com        |    4 | 7046 | 200003    |
+------------+------------+------------+--------+----------------------+------------+--------------+-----------------------------+------+------+-----------+
20 rows in set (0.00 sec)
```

- **assessment**(assess_id,coursework_result ,exams_result)

```
mysql> SELECT * FROM assessment;
+-----------+------------------+--------------+
| assess_id | coursework_result | exams_result |
+-----------+------------------+--------------+
|    900001 |               10 |           90 |
|    900002 |               20 |           80 |
|    900003 |               30 |           70 |
|    900004 |               40 |           60 |
|    900005 |               50 |           50 |
|    900006 |               60 |           40 |
|    900007 |               70 |           30 |
+-----------+------------------+--------------+
7 rows in set (0.00 sec)
```

- **module_assess**(module_id, assess_id)

```
mysql> SELECT * FROM module_assess;
+--------+-----------+
| mod_id | assess_id |
+--------+-----------+
| 500001 |    900001 |
| 500002 |    900002 |
| 500003 |    900003 |
| 500004 |    900004 |
| 500005 |    900005 |
| 500006 |    900006 |
| 500007 |    900007 |
+--------+-----------+
7 rows in set (0.00 sec)
```

- **next_of_kin**(student_id, name, relationship, phone_num)

```
mysql> SELECT * FROM next_of_kin;
+----------------------+--------------+--------------+------------+
| fullname             | relationship | phone_num    | student_id |
+----------------------+--------------+--------------+------------+
| Aldin Silverstone    | Mother       | 128-915-7542 |     800005 |
| Aluino Pylkynyton    | Father       | 270-232-8508 |     800006 |
| Annora Beckford      | Father       | 144-284-0577 |     800001 |
| Carlene Whild        | Father       | 276-860-4099 |     800009 |
| Eimile Thomson       | Father       | 592-907-2869 |     800007 |
| Kevyn Rubury         | Mother       | 480-182-7767 |     800004 |
| Minnaminnie Mealiffe | Father       | 316-965-4019 |     800010 |
| Pail Hofler          | Mother       | 538-474-5233 |     800003 |
| Rodge Burrett        | Mother       | 345-662-0032 |     800008 |
| Terese Keast         | Father       | 459-592-9337 |     800002 |
+----------------------+--------------+--------------+------------+
10 rows in set (0.00 sec)
```

- **staff_module**(staff_id, module_id)

```
mysql> SELECT * FROM staff_module;
+----------+-----------+
| staff_id | module_id |
+----------+-----------+
|   300001 | 500001    |
|   300002 | 500002    |
|   300003 | 500003    |
|   300004 | 500004    |
|   300005 | 500005    |
|   300006 | 500006    |
|   300001 | 500007    |
|   300007 | 500008    |
+----------+-----------+
8 rows in set (0.00 sec)
```

- **book_module**(book_id, module_id)

```
mysql> SELECT * FROM book_module;
+---------+-----------+
| book_id | module_id |
+---------+-----------+
|  600001 | 500001    |
|  600002 | 500002    |
|  600003 | 500003    |
|  600004 | 500004    |
|  600005 | 500005    |
|  600006 | 500006    |
|  600001 | 500007    |
|  600007 | 500008    |
+---------+-----------+
8 rows in set (0.00 sec)
```

- **student_module**(student_id, module_id, semester, result, perform)

```
mysql> SELECT * FROM student_module;
+------------+-----------+----------+--------+------+---------+
| student_id | module_id | semester | result | year | perform |
+------------+-----------+----------+--------+------+---------+
|     800001 | 500001    | 2        |     80 | 2018 | PASS    |
|     800002 | 500002    | 2        |     75 | 2018 | PASS    |
|     800003 | 500003    | 2        |     70 | 2018 | PASS    |
|     800004 | 500004    | 2        |     65 | 2018 | PASS    |
|     800005 | 500005    | 2        |     60 | 2018 | PASS    |
|     800006 | 500001    | 2        |     55 | 2018 | PASS    |
|     800007 | 500002    | 2        |     50 | 2018 | PASS    |
|     800008 | 500003    | 2        |     70 | 2018 | PASS    |
|     800009 | 500004    | 2        |     63 | 2018 | PASS    |
|     800010 | 500005    | 2        |     72 | 2018 | PASS    |
|     800011 | 500001    | 2        |     82 | 2018 | PASS    |
|     800012 | 500003    | 2        |     90 | 2018 | PASS    |
+------------+-----------+----------+--------+------+---------+
12 rows in set (0.00 sec)
```

- **department_staff**(dept_id,staff_id)

```
mysql> SELECT * FROM department_staff;
+---------+----------+
| dept_id | staff_id |
+---------+----------+
| 100001  |   300001 |
| 100002  |   300002 |
| 100003  |   300003 |
| 100004  |   300004 |
| 100005  |   300005 |
| 100006  |   300006 |
| 100001  |   300007 |
| 100007  |   300008 |
+---------+----------+
8 rows in set (0.00 sec)
```

Part 3 – Query the Database

Before starting this section, please ensure that your tables contain sufficient data to enable you to test the query transactions described in the University Database case study.

3. Create and save the following query transactions:

a. List details of all departments located in E Block.

```
mysql> SELECT * FROM department WHERE dept_location = 'E Block';
+---------+-----------+--------------+--------------+---------------+--------------+
| dept_id | dept_name | phone        | fax_num      | dept_location | dept_manager |
+---------+-----------+--------------+--------------+---------------+--------------+
| 100001  | CIS       | 519 137 6809 | 140-690-9484 | E Block       |       300001 |
| 100002  | ENG       | 338 955 9399 | 140-150-0352 | E Block       |       300002 |
| 100003  | ARC       | 824 899 3930 | 140-139-0195 | E Block       |       300003 |
+---------+-----------+--------------+--------------+---------------+--------------+
3 rows in set (0.00 sec)
```

b. List title, start and end dates of all modules run in the PgDIT course.

*PgDIT course = computer science course (200001)

```
mysql> SELECT module.title,cdate.start_date,cdate.end_date FROM module
    -> INNER JOIN course_date cdate
    -> WHERE module.course_id = cdate.course_id
    -> AND
    -> module.course_id = 200001;
+--------------------+------------+------------+
| title              | start_date | end_date   |
+--------------------+------------+------------+
| Relational Database | 2018-09-05 | 2019-05-05 |
| Java               | 2018-09-05 | 2019-05-05 |
| Computer Network   | 2018-09-05 | 2019-05-05 |
| Web development    | 2018-09-05 | 2019-05-05 |
| Operating Systems  | 2018-09-05 | 2019-05-05 |
+--------------------+------------+------------+
5 rows in set (0.00 sec)
```

c. List name, address, and salary for each female member of academic staff who manages a department.

```
mysql> SELECT
    -> academic_staff.first_name,
    -> academic_staff.last_name,
    -> academic_staff.address,
    -> academic_staff.salary
    -> FROM academic_staff,department
    -> WHERE academic_staff.staff_id = department.dept_manager
    -> AND academic_staff.gender = 'F';
+------------+-----------+--------------------+--------+
| first_name | last_name | address            | salary |
+------------+-----------+--------------------+--------+
| Magdalen   | Benthall  | 91 Scott Avenue    |  40068 |
| Valeda     | Christofe | 81464 Glendale Pass |  34654 |
| Beatrix    | Buchan    | 988 Pearson Park   |  47515 |
+------------+-----------+--------------------+--------+
3 rows in set (0.00 sec)
```

d. List name, sex, and salary for each lecturer with a PhD degree.

```
mysql> SELECT DISTINCT
    -> academic_staff.first_name,
    -> academic_staff.last_name,
    -> academic_staff.address,
    -> academic_staff.salary,
    -> qual.edu_level
    -> FROM academic_staff
    -> INNER JOIN staff_qualification squal
    -> INNER JOIN qualification qual
    -> WHERE squal.qual_id = qual.qual_id
    -> AND qual.edu_level = 'PHD';
+------------+--------------+--------------------------+--------+-----------+
| first_name | last_name    | address                  | salary | edu_level |
+------------+--------------+--------------------------+--------+-----------+
| Wesley     | Mansell      | 5539 Hoepker Junction    |  49473 | PHD       |
| Clevie     | Greenaway    | 16245 Dayton Road        |  43797 | PHD       |
| Magdalen   | Benthall     | 91 Scott Avenue          |  40068 | PHD       |
| Moore      | Biaggetti    | 9 Vidon Crossing         |  36935 | PHD       |
| Valeda     | Christofe    | 81464 Glendale Pass      |  34654 | PHD       |
| Ode        | Polye        | 1517 Algoma Alley        |  35022 | PHD       |
| Cly        | Gravenor     | 81440 Hallows Terrace    |  34704 | PHD       |
| Beatrix    | Buchan       | 988 Pearson Park         |  47515 | PHD       |
| Stuart     | Fragino      | 602 Burning Wood Crossing |  41407 | PHD       |
| Carly      | Postans      | 6633 Kenwood Place       |  40777 | PHD       |
| Elnore     | Tschierasche | 56083 Nobel Court        |  30822 | PHD       |
| Gillian    | Ipwell       | 22441 Roth Road          |  41309 | PHD       |
| Sascha     | Panichelli   | 67437 Northport Alley    |  36665 | PHD       |
| Nevile     | Mocher       | 482 Springview Way       |  33248 | PHD       |
| Merrel     | MacDonogh    | 049 Wayridge Center      |  48651 | PHD       |
| Gardener   | Espinel      | 86 Clyde Gallagher Plaza |  40224 | PHD       |
| Coraline   | Mowsley      | 89748 Haas Hill          |  43198 | PHD       |
| Kristos    | Grzes        | 1107 West Center         |  36629 | PHD       |
| Becca      | Aleksandrikin | 38 Shoshone Drive        |  33208 | PHD       |
| Laureen    | Hilbourne    | 0580 Eastlawn Alley      |  35623 | PHD       |
+------------+--------------+--------------------------+--------+-----------+
20 rows in set (0.00 sec)
```

e. List last name, post, and qualifications of all members of academic staff who are employed by CIS department.

* Used gender instead of post. (Do not have post attribute.)
* CIS department = 100001.

```
mysql> SELECT DISTINCT
    -> academic_staff.first_name,
    -> academic_staff.last_name,
    -> academic_staff.gender,
    -> qual.title
    -> FROM academic_staff,department_staff
    -> INNER JOIN staff_qualification squal
    -> INNER JOIN qualification qual
    -> WHERE squal.staff_id = academic_staff.staff_id
    -> AND squal.qual_id = qual.qual_id
    -> AND academic_staff.staff_id = department_staff.staff_id
    -> AND department_staff.dept_id= '100001';
+------------+-----------+--------+-------------------+
| first_name | last_name | gender | title             |
+------------+-----------+--------+-------------------+
| Wesley     | Mansell   | M      | Computer Sciences |
+------------+-----------+--------+-------------------+
1 row in set (0.00 sec)
```

f. List matriculation number, last name, and sex of all students who are studying 'multi-media' module. Order result alphabetically by last name.

* multi-media = Relational database (500001) (Do not have multi-media module.)

```
mysql> SELECT DISTINCT
    -> student.student_id,
    -> student.last_name,
    -> student.gender
    -> FROM student,student_module
    -> WHERE student.student_id = student_module.student_id
    -> AND student_module.module_id = 500001
    -> ORDER BY student.last_name;
+------------+-----------+--------+
| student_id | last_name | gender |
+------------+-----------+--------+
|     800001 | Anstis    | F      |
|     800006 | McInally  | M      |
|     800011 | Simes     | F      |
+------------+-----------+--------+
3 rows in set (0.00 sec)
```

g. List staff number, last name, sex, and post of all academic staff whose salary is greater than the average salary of all academic staff.

*Did not add post attribute below

```
mysql> SELECT staff_id, last_name, gender
    -> FROM academic_staff
    -> WHERE salary > (SELECT AVG(salary)FROM academic_staff);
+----------+-----------+--------+
| staff_id | last_name | gender |
+----------+-----------+--------+
|   300001 | Mansell   | M      |
|   300002 | Greenaway | M      |
|   300003 | Benthall  | F      |
|   300008 | Buchan    | F      |
|   300009 | Fragino   | M      |
|   300010 | Postans   | M      |
|   300012 | Ipwell    | F      |
|   300015 | MacDonogh | M      |
|   300016 | Espinel   | M      |
|   300017 | Mowsley   | F      |
+----------+-----------+--------+
10 rows in set (0.00 sec)
```

h. For each course with more than 10 students, list course title and the number of students (under an appropriate header).

*Used more than 8 students as there was no module with more then 10 students

```
mysql> SELECT
    -> course.title,
    -> Count(*)
    -> FROM student,course
    -> WHERE student.course_id = course.course_id
    -> GROUP BY course.title
    -> HAVING COUNT(*) > 8 ;
+------------------+----------+
| title            | Count(*) |
+------------------+----------+
| Architecture     |       10 |
| Computer Science |       10 |
+------------------+----------+
2 rows in set (0.00 sec)
```

i. List the number of female members of academic staff and the number of male members of academic staff employed by CIS department.

* CIS department = 100001.

```
mysql> SELECT gender,COUNT(gender)
    -> FROM academic_staff
    -> INNER JOIN department_staff
    -> WHERE department_staff.dept_id= '100001'
    -> GROUP BY gender;
+--------+---------------+
| gender | COUNT(gender) |
+--------+---------------+
| M      |            22 |
| F      |            18 |
+--------+---------------+
2 rows in set (0.01 sec)
```

j. For each member of academic staff who spends more than 6 hours teaching any module list the member of academic staff last name, the module title and the number of hours.

```
mysql> SELECT academic_staff.last_name, module.title, module.duration
    -> FROM academic_staff
    -> INNER JOIN staff_module
    -> INNER JOIN module
    -> WHERE academic_staff.staff_id = staff_module.staff_id
    -> AND staff_module.module_id = module.mod_id
    -> AND module.duration > 6;
+-----------+----------------------+----------+
| last_name | title                | duration |
+-----------+----------------------+----------+
| Mansell   | Relational Database  |       80 |
| Greenaway | Java                 |       80 |
| Benthall  | Computer Network     |       60 |
| Biaggetti | Web development      |      100 |
| Christofe | Operating Systems    |       60 |
| Polye     | Architecture         |      100 |
| Mansell   | Arch History         |       60 |
| Gravenor  | Visual Communication |       80 |
+-----------+----------------------+----------+
8 rows in set (0.00 sec)
```

k. For each department list the department name, and the number of female members of academic staff, and the number of male members of academic staff under appropriate headers (use a crosstab query).

```
mysql> SELECT department.dept_name, academic_staff.gender,COUNT(academic_staff.gender)
    -> FROM academic_staff
    -> INNER JOIN department_staff
    -> INNER JOIN department
    -> WHERE academic_staff.staff_id = department_staff.staff_id
    -> AND department_staff.dept_id = department.dept_id
    -> GROUP BY academic_staff.gender;
+-----------+--------+-----------------------------+
| dept_name | gender | COUNT(academic_staff.gender) |
+-----------+--------+-----------------------------+
| CIS       | M      |                           5 |
| ARC       | F      |                           3 |
+-----------+--------+-----------------------------+
2 rows in set (0.00 sec)
```

4. Provide 5 additional examples of queries, which retrieve useful data from the University Database database. State the purpose of each query and attempt to use each example to demonstrate the breadth of your knowledge of SQL. Create stored procedures with these 5 queries.

a. Highest department manager salary in University with his/her details.

```
mysql> SELECT
    -> dept.dept_name,
    -> academic_staff.first_name,
    -> academic_staff.last_name,
    -> academic_staff.gender,
    -> MAX(academic_staff.salary)
    -> FROM academic_staff
    -> INNER JOIN department dept
    -> WHERE academic_staff.staff_id = dept.dept_manager;
+-----------+------------+-----------+--------+----------------------------+
| dept_name | first_name | last_name | gender | MAX(academic_staff.salary) |
+-----------+------------+-----------+--------+----------------------------+
| CIS       | Wesley     | Mansell   | M      |                      49473 |
+-----------+------------+-----------+--------+----------------------------+
1 row in set (0.00 sec)
```

b. Count on female and male students in Computer Science Course.

```
mysql> SELECT
    -> course.title,
    -> student.gender,
    -> COUNT(student.gender)
    -> FROM student,course
    -> WHERE student.course_id = course.course_id
    -> AND course.title = 'Computer Science'
    -> GROUP BY student.gender;
+------------------+--------+----------------------+
| title            | gender | COUNT(student.gender) |
+------------------+--------+----------------------+
| Computer Science | M      |                    4 |
| Computer Science | F      |                    6 |
+------------------+--------+----------------------+
2 rows in set (0.00 sec)
```

c. Books that are have been published before year 2000.

```
mysql> SELECT
    -> module.title,
    -> book.book_title,
    -> book.author,
    -> book.publication,
    -> book.year
    -> FROM book
    -> INNER JOIN book_module bmod
    -> INNER JOIN module
    -> WHERE book.book_id = bmod.book_id
    -> AND bmod.module_id = module.mod_id
    -> HAVING module.title = 'Relational Database';
+---------------------+--------------------+---------------+-------------+------+
| title               | book_title         | author        | publication | year |
+---------------------+--------------------+---------------+-------------+------+
| Relational Database | Algorithms to Live | Tom Griffiths | Pearson     | 1993 |
+---------------------+--------------------+---------------+-------------+------+
1 row in set (0.00 sec)
```

d. Student details and there next of Kin.

```
mysql> SELECT
    -> student.student_id,
    -> student.first_name,
    -> student.last_name,
    -> kin.fullname Next_of_Kin,
    -> kin.relationship,
    -> kin.phone_num
    -> FROM student
    -> INNER JOIN next_of_kin kin
    -> WHERE kin.student_id = student.student_id;
+------------+------------+------------+----------------------+--------------+--------------+
| student_id | first_name | last_name  | Next_of_Kin          | relationship | phone_num    |
+------------+------------+------------+----------------------+--------------+--------------+
|     800005 | Tasia      | Sprague    | Aldin Silverstone    | Mother       | 128-915-7542 |
|     800006 | Rice       | McInally   | Aluino Pylkynyton    | Father       | 270-232-8508 |
|     800001 | Jaquelyn   | Anstis     | Annora Beckford      | Father       | 144-284-0577 |
|     800009 | Aile       | Stollmeyer | Carlene Whild        | Father       | 276-860-4099 |
|     800007 | Hersh      | Mcwhinney  | Eimile Thomson       | Father       | 592-907-2869 |
|     800004 | Hildagard  | Vauter     | Kevyn Rubury         | Mother       | 480-182-7767 |
|     800010 | Debra      | Dearnaley  | Minnaminnie Mealiffe | Father       | 316-965-4019 |
|     800003 | Cathe      | Jellybrand | Pail Hofler          | Mother       | 538-474-5233 |
|     800008 | Natty      | Sharland   | Rodge Burrett        | Mother       | 345-662-0032 |
|     800002 | Benedetto  | Tinmouth   | Terese Keast         | Father       | 459-592-9337 |
+------------+------------+------------+----------------------+--------------+--------------+
10 rows in set (0.00 sec)
```

e. Students that have passed their Relational Database Module.

```
mysql> SELECT module.title module_title,
    -> student.student_id,
    -> student.first_name,
    -> student.last_name,
    -> smod.perform performance,
    -> smod.result
    -> FROM student
    -> INNER JOIN student_module smod
    -> INNER JOIN module
    -> WHERE student.student_id = smod.student_id
    -> AND module.mod_id = smod.module_id
    -> HAVING module.title = 'Relational Database';
+---------------------+------------+------------+-----------+-------------+--------+
| module_title        | student_id | first_name | last_name | performance | result |
+---------------------+------------+------------+-----------+-------------+--------+
| Relational Database |     800001 | Jaquelyn   | Anstis    | PASS        |     80 |
| Relational Database |     800006 | Rice       | McInally  | PASS        |     55 |
| Relational Database |     800011 | Christy    | Simes     | PASS        |     82 |
+---------------------+------------+------------+-----------+-------------+--------+
3 rows in set (0.01 sec)
```