UNIVESITY DATABASE

Contents

Introduction to Coursework

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*.

Coursework Requirements

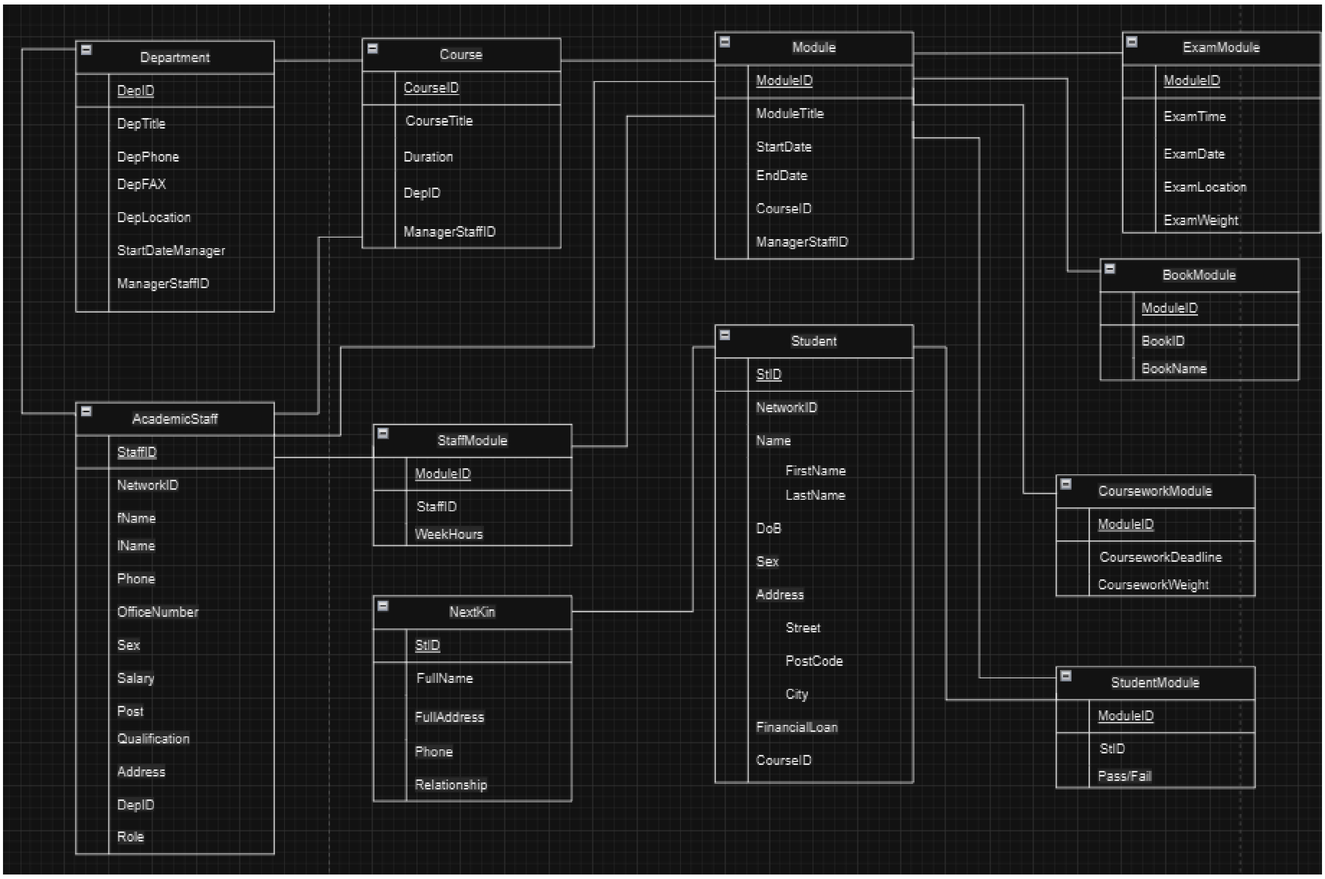
Each department runs a number of courses. The university provides a set of modules used in different courses. Each course uses a number of modules but not every module is used. A course is assigned a unique course code and a module is identified by a unique module code. A module can be used in one course 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 spends 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 works for one department only.

Part 1 – Design the Database

1. Diagrams

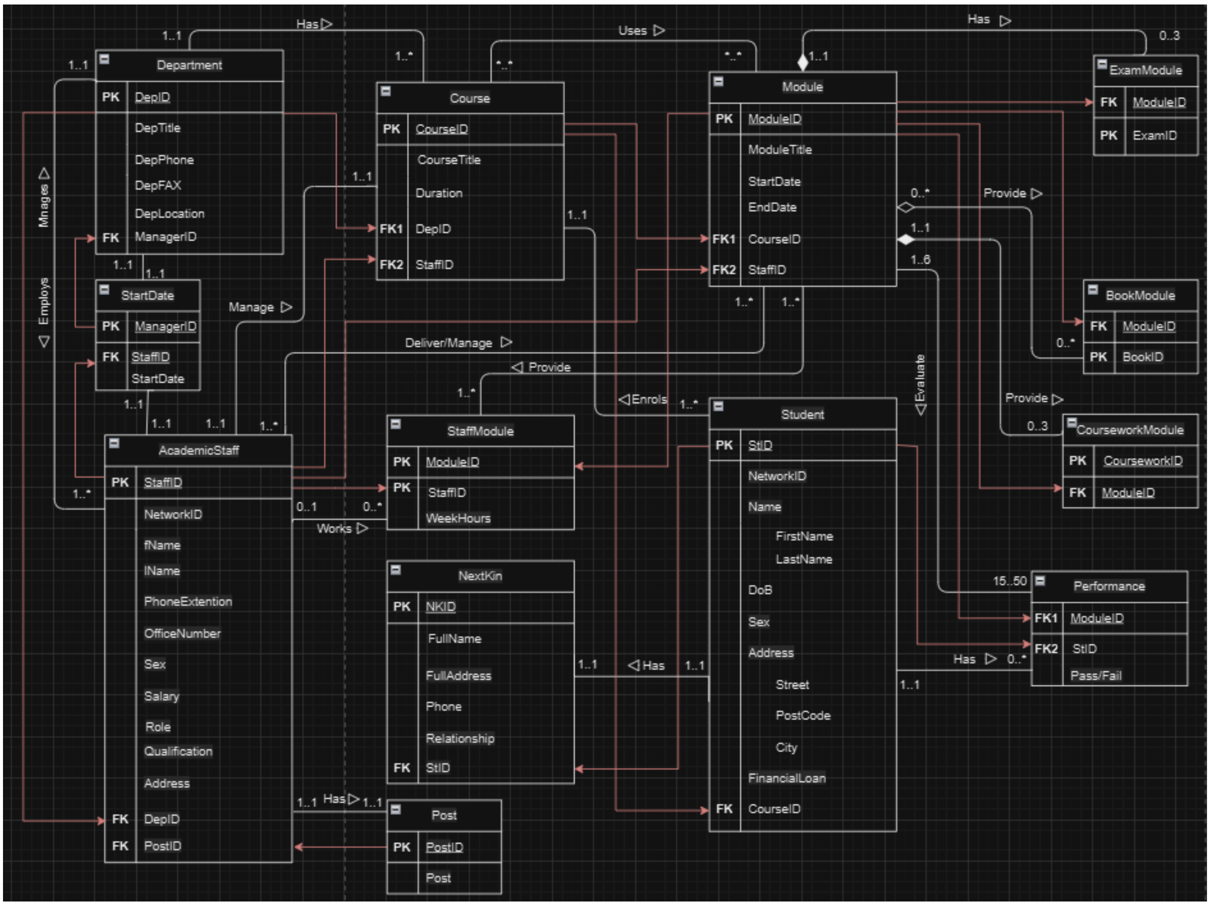
Create an Entity–Relationship (ER) model of the data requirements for the University Database case study using the UML notation. Use the additional concepts of the Enhanced Entity–Relationship (EER) model. State any assumptions necessary to support your design. **(5 marks)**

The Entity–Relationship (ER) model was created by retrieving values from the requrements in UNF form and dividing them into relations.



Entity-Relationship Diagram (ER)

The Enhanced Entity–Relationship (EER) model was developed after achieving 3NF.



Enhanced Entity–Relationship Diagram (EER)

2. Relational schema

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. **(5 marks)**

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

Primary Key staffNo Alternate Key lName, DOB

Alternate Key NIN

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

The *Relational schema* was created based on EER

1) **Department** (DeptId, DepTitle, DepPhone, DepFax, DepLocation, ManagerStaffId)
   Primary Key: DeptId
   Alternate Key: DepTitle, DepLocation
   Foreign Key: ManagerId references StartDate (ManagerId) On Delete Set Null On Update Cascade

2) **StartDate** (ManagertId , StaffId, StartDateManager)
   Primary Key: ManagertId
   Foreign Key: StaffId references AcademicStaff (StaffId) On Delete Set Null On Update Cascade

3) **Course** (CourseId, CourseTitle, CourseDuration, DeptId, ManagerStaffId)
   Primary Key: CourseId
   Alternate Key: CourseTitle, ManagerStaffId
   Foreign Key: ManagerStaffId references Staff (ManagerStaffId) On Delete No Action On Update Cascade
   Foreign Key: DepId references Department (DepId) On Delete No Action On Update Cascade

4) **Module** (ModuleId, ModuleTitle, startDate, endDate, CourseId, MansgerStaffId)
   Primary Key: ModuleId
   Foreign Key: MansgerStaffId references Staff (MansgerStaffId) On Delete No Action On Update Cascade
   Foreign Key: CourseId references Course (CourseId) On Delete No Action On Update Cascade

5) **Student** (StId, NetworkId, Name(FirstName, LastName), DoB, Sex, Address(Street,PostCode,City) FinancialLoan, CourseId)
   Primary Key: StId
   Alternate Key: NetworkId
   Foreign Key: CourseId references Course (CourseId) On Delete No Action On Update Cascade

6) **AcademicStaff** (StaffId, NetworkId, fName, lName, PhoneExtension, OfficeNumber, Sex, Salary, PostId, Qualification, Address, DepId, Role)
   Primary Key: StaffId
   Alternate Key: NetworkId
   Foreign Key: DepId references Department (DepId) On Delete No Action On Update Cascade
   Foreign Key: PostID references Post (PostID) On Delete No Action On Update Cascade

7) Post(PostID,Post)
Primary Key: PostID

8) **NextKin**(StId, FullName, FullAddress, Phone, Relationship, nextOfKinId)
Primary Key: nextOfKinId
Alternate Key: Phone
Foreign Key: StId references Student (StId) On Delete Cascade

9) **StaffModule** (ModuleId, StaffId, WeekHours)
Primary Key: ModuleId, StaffId
Foreign Key: ModuleId references Module (ModuleId) On Delete No Action On Update Cascade
Foreign Key: StaffId references AcademicStaff (StaffId) On Delete Set Null On Update Cascade

10) **Performance** (StId, ModuleId, Pass/Fail)
Primary Key: (StId, ModuleId)
Foreign Key: StId references Student (SId) On Delete Set Null On Update Cascade
Foreign Key: ModuleId references Module (ModuleId) On Delete No Action On Update Cascade

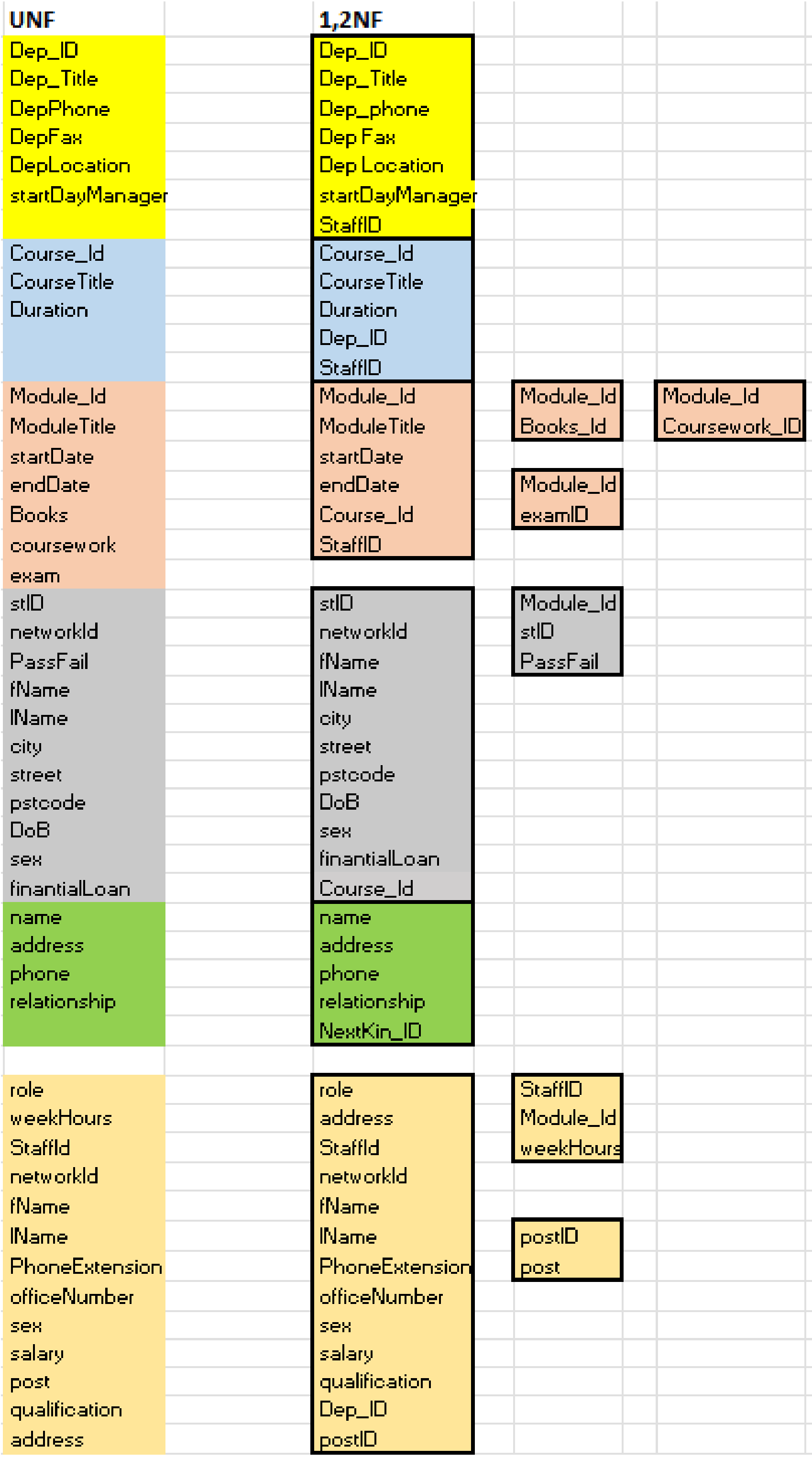
11) **ExamModule**(ModuleId, ExamID)
Primary Key: (ExamID)
Foreign Key: ModuleId references Module (ModuleId) On Delete Cascade

12) **BookModule** (ModuleId, BookId)
Primary Key: BookId
Foreign Key: ModuleId references Module (ModuleId) On Delete Cascade

13) **CourseworkModule**(ModuleId, CourseworkID)
Primary Key: CourseworkID
Foreign Key: ModuleId references Module (ModuleId) On Delete Cascade

3. Normalization

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. **(5 marks)**



UNF	1,2NF		
Dep_ID	Dep_ID		
Dep_Title	Dep_Title		
DepPhone	Dep_phone		
DepFax	Dep Fax		
DepLocation	Dep Location		
startDayManager	startDayManager		
	StaffID		
Course_Id	Course_Id		
CourseTitle	CourseTitle		
Duration	Duration		
	Dep_ID		
	StaffID		
Module_Id	Module_Id	Module_Id	Module_Id
ModuleTitle	ModuleTitle	Books_Id	Coursework_ID
startDate	startDate		
endDate	endDate	Module_Id	
Books	Course_Id	examID	
coursework	StaffID		
exam			
stID	stID	Module_Id	
networkId	networkId	stID	
PassFail	fName	PassFail	
fName	lName		
lName	city		
city	street		
street	pstcode		
pstcode	DoB		
DoB	sex		
sex	finantialLoan		
finantialLoan	Course_Id		
name	name		
address	address		
phone	phone		
relationship	relationship		
	NextKin_ID		
role	role	StaffID	
weekHours	address	Module_Id	
StaffId	StaffId	weekHours	
networkId	networkId		
fName	fName		
lName	lName	postID	
PhoneExtension	PhoneExtension	post	
officeNumber	officeNumber		
sex	sex		
salary	salary		
post	qualification		
qualification	Dep_ID		
address	postID		

UNF to 1,2NF

fd1:{Dep_ID, Dep_Title, DepPhone, DepFax, DepLocation, startDayManager, Staff_ID**}**

fd2:{CourseID, CourseTitle, Duration, Dep_ID, StafID}

Since the *Modules* table contains non-atomic values, I divided it into four separate relations.

fd3:{ModuleID, ModuleTitle, StartDate, EndDate, CourseID, StaffID}

fd4:{ ModuleID, Books_ID}

fd5:{ ModuleID, exam_ID}

fd6:{ ModuleID, coursework_ID}

Student performance is non-atomic and involves a transitive dependency, so I moved it to a separate table.

fd7:{ StID,NetworkID,}

fd8:{ ModuleID, StID, Pass/Fail}

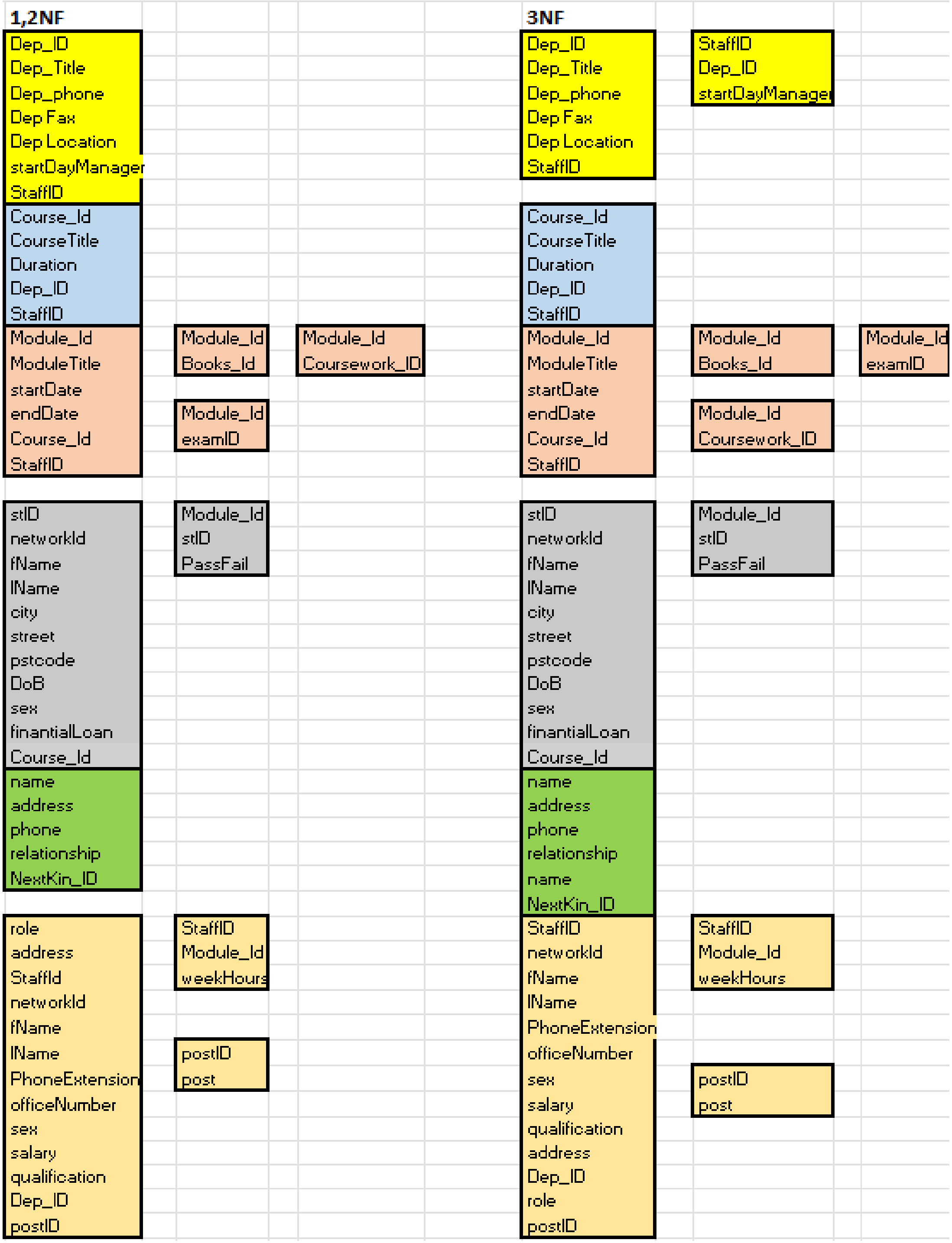
fd9:{NextKin_Id, Name, Address, Phone, Relation, StID}

In the *Staff* relation, there is a partial dependency involving *weekHours* and *post*. To address this, I will split it into three separate relations.

fd10:{StaffID, NetworkID fName, lName, PhoneExtnsion, OfficeNumber, sex, salary, qualification, Dep_ID}

fd11:{StaffID, ModuleID, weekHours}

fd11:{StaffID, post}



1,2NF

Dep_ID
Dep_Title
Dep_phone
Dep Fax
Dep Location
startDayManager
StaffID

Course_Id
CourseTitle
Duration
Dep_ID
StaffID

Module_Id
ModuleTitle
startDate
endDate
Course_Id
StaffID

Module_Id
Books_Id

Module_Id
Coursework_ID

Module_Id
examID

stID
networkId
fName
lName
city
street
pstcode
DoB
sex
finantialLoan
Course_Id

Module_Id
stID
PassFail

name
address
phone
relationship
NextKin_ID

role
address
StaffId
networkId
fName
lName
PhoneExtension
officeNumber
sex
salary
qualification
Dep_ID
postID

StaffID
Module_Id
weekHours

postID
post

3NF

Dep_ID
Dep_Title
Dep_phone
Dep Fax
Dep Location
StaffID

StaffID
Dep_ID
startDayManager

Course_Id
CourseTitle
Duration
Dep_ID
StaffID

Module_Id
ModuleTitle
startDate
endDate
Course_Id
StaffID

Module_Id
Books_Id

Module_Id
examID

Module_Id
Coursework_ID

stID
networkId
fName
lName
city
street
pstcode
DoB
sex
finantialLoan
Course_Id

Module_Id
stID
PassFail

name
address
phone
relationship
name
NextKin_ID

StaffID
networkId
fName
lName
PhoneExtension
officeNumber
sex
salary
qualification
address
Dep_ID
role
postID

StaffID
Module_Id
weekHours

postID
post

To move from 2NF to 3NF, I will ensure that all functional dependencies are preserved.

In the *Department* table, there is a transitive dependency involving *startDateManager*. To resolve this, I will split the table into two separate relations.

fd1:{Dep_ID, Dep_Title, DepPhone, DepFax, DepLocation, startDayManager, Staff_ID}

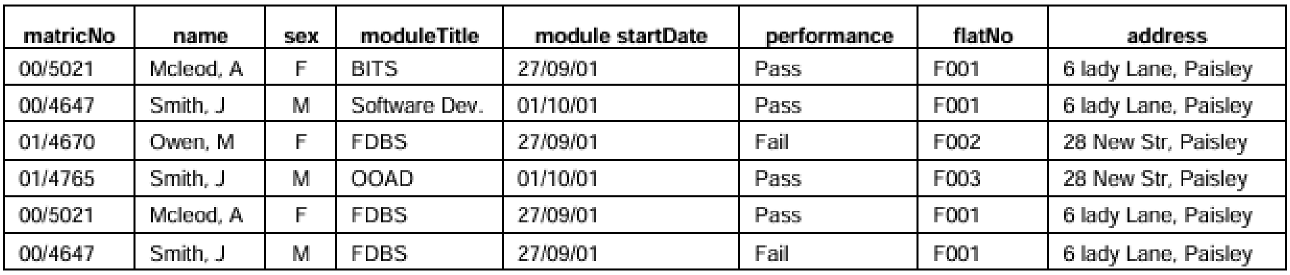
fd12:{ Dep_ID, Dep_Title, DepPhone, DepFax, DepLocation, Staff_ID}

fd13:{ Dep_ID, Staff_ID, startDayManager}

Each functional dependency from fd2 to fd13 will be transformed into separate relations.

4. Demonstration of Normalization

To further demonstrate your knowledge of normalization, assume that a proposed (badly structured) relation for the University Database database has the following structure.



matricNo	name	sex	moduleTitle	module startDate	performance	flatNo	address
00/5021	Mcleod, A	F	BITS	27/09/01	Pass	F001	6 lady Lane, Paisley
00/4647	Smith, J	M	Software Dev.	01/10/01	Pass	F001	6 lady Lane, Paisley
01/4670	Owen, M	F	FDBS	27/09/01	Fail	F002	28 New Str, Paisley
01/4765	Smith, J	M	OOAD	01/10/01	Pass	F003	28 New Str, Paisley
00/5021	Mcleod, A	F	FDBS	27/09/01	Pass	F001	6 lady Lane, Paisley
00/4647	Smith, J	M	FDBS	27/09/01	Fail	F001	6 lady Lane, Paisley

Identify the functional dependencies represented in this relation and demonstrate the process of normalizing this relation into 3NF relations. **(5 marks)**

fd1: {matricNo, name, sex, moduleTitle, flatNo}
fd2: {moduleTitle, moduleStartDate}
fd3: {flatNo, address}
fd4: {matricNo, moduleTitle, performance}

fd2, fd3 violates 2NF so I split them in different tables.

matricNo	name	sex	moduleTitle	flatNo	performance

moduleTitle	moduleStartDate

flatNo	address

2NF
fd1: matricNo, name, sex, moduleTitle, flatNo
fd2: moduleTitle, moduleStartDate,
fd3: flatNo, address
fd4: matricNo, moduleTitle, performance

fd4 violates 3NF.

3NF

matricNo (PK)	name	sex	moduleTitle (FK)	flatNo (FK)

moduleTitle (PK)	moduleStartDate

flatNo (PK)	address

matricoNo (PK)	moduleTitle (PK)	performance

Here we have a composite key (matricoNo, moduleTitle)

Part 2 – Implement the Database

1. Table Creation

Create the tables for the University Database database. Where appropriate set field and table properties, including any required indexes. Ensure that referential integrity is established between related tables. **(5 marks)**

```
 1) CREATE TABLE Department
(Dep_ID char(5)PRIMARY KEY,
Dep_Title varchar(35) NOT NULL,
Dep_Phone number (10) UNIQUE,
Dep_Fax number (10) UNIQUE,
Dep_Location varchar(15) NOT NULL,
Manager_ID char(5) NOT NULL);
```

```
 2) CREATE TABLE StartDate
(Manager_ID char(5)PRIMARY KEY ,
  Staff_ID char(5) NOT NULL,
  Start_Date_Manager date NOT NULL );
```

```
 3) CREATE TABLE Course
(Course_ID char(5)PRIMARY KEY,
Course_Title varchar(50) NOT NULL,
Duration varchar(9) NOT NULL,
Dep_ID char(5) NOT NULL,
Staff_ID char(5) NOT NULL);
```

```
 4) CREATE TABLE Modules
(Module_ID char(5)PRIMARY KEY,
Module_Title varchar(35) NOT NULL,
Start_Date date NOT NULL,
End_Date date NOT NULL,
Course_ID char(5) NOT NULL,
Staff_ID char(5)  Unique NOT NULL);
```

```
 5) CREATE TABLE AcademicStaff
(Staff_ID char(5)PRIMARY KEY,
Network_ID varchar(6) UNIQUE,
fName varchar(10) NOT NULL,
lname varchar(10) NOT NULL,
phoneExtension number(4) UNIQUE ,
officeNumber number(3) NOT NULL,
sex varchar(1) NOT NULL,
salary number(6) NOT NULL,
PostID varchar(17) NOT NULL,
qualification varchar(22) NOT NULL,
address varchar(40) NOT NULL,
Dep_ID char(5) NOT NULL,
Roles varchar(20));
```

```
  6) CREATE TABLE Post
(Post_ID char(2) PRIMARY KEY,
Post varchar(15));

  7) CREATE TABLE StaffModule
(Module_ID char(5),
Staff_ID varchar(5),
Week_Hours varchar(2));

  8) CREATE TABLE Student
(St_ID char(5)PRIMARY KEY,
Network_ID varchar(6) UNIQUE,
fName varchar(10) NOT NULL,
lname varchar(10) NOT NULL,
DoB date NOT NULL,
sex varchar(1) NOT NULL,
street varchar(35) NOT NULL,
post_code varchar(35) NOT NULL,
city varchar(12) NOT NULL,
financial_loan varchar(5),
Course_ID varchar(10) NOT NULL);

  9) CREATE TABLE Performance
(Module_ID char(5),
St_ID varchar(5),
Pass_Fail varchar(4));

  10) CREATE TABLE NextKin
(Next_Kin_ID char(5) UNIQUE,
full_Name varchar(20)NOT NULL,
Full_Address varchar(43)NOT NULL,
Phone varchar(15) NOT NULL UNIQUE,
Relationship varchar(10) NOT NULL,
St_ID char(5)NOT NULL UNIQUE);

  11) CREATE TABLE examModule
(Module_ID char(5) NOT NULL,
Exam_ID varchar(4) PRIMARY KEY);

  12) CREATE TABLE BookModule
(Book_ID varchar(8) PRIMARY KEY,
Module_ID char(5));

  13) CREATE TABLE CourseworkModule
(Module_ID char(5) PRIMARY KEY,
Coursework_ID varchar(4));
```

I am assuming that the tables examModule, BookModule, and CourseworkModule already exist.

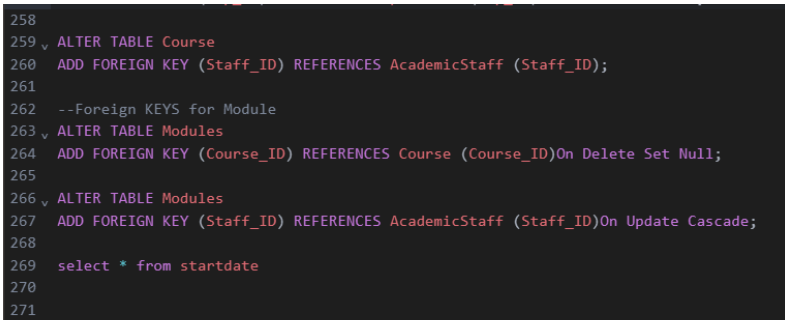
2. Customised forms

Create customised forms for data entry. **(2.5 marks)**

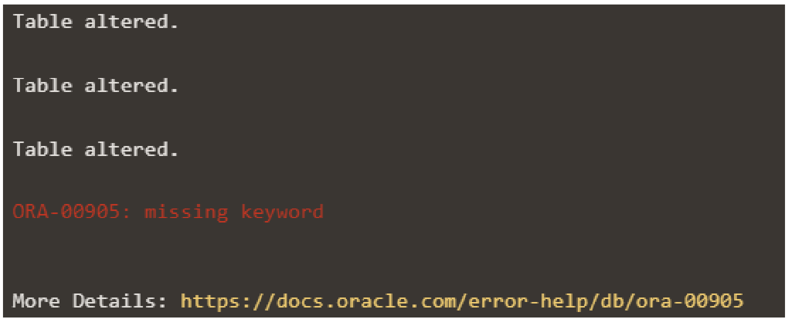
```
--Foreign KEYS for Department
ALTER TABLE Department
ADD FOREIGN KEY (Manager_ID) REFERENCES StartDate (Manager_ID) On Delete Set Null;

----Foreign KEYS for StartDate
ALTER TABLE StartDate
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID) On Delete Set Null;

--Foreign KEYS for Course
ALTER TABLE Course
ADD FOREIGN KEY (Dep_ID) REFERENCES Department (Dep_ID);

ALTER TABLE Course
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID);

--Foreign KEYS for Module
ALTER TABLE Modules
ADD FOREIGN KEY (Course_ID) REFERENCES Course (Course_ID);

ALTER TABLE Modules
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID);

--Foreign KEYS for Student
ALTER TABLE Student
ADD FOREIGN KEY (Course_ID) REFERENCES Course (Course_ID);

--Foreign KEYS for AcademicStaff
ALTER TABLE AcademicStaff
ADD FOREIGN KEY (Dep_ID) REFERENCES Department (Dep_ID);

ALTER TABLE AcademicStaff
ADD FOREIGN KEY (Post_ID) REFERENCES Post (Post_ID);

--Foreign KEYS for NextKin
ALTER TABLE NextKin
ADD FOREIGN KEY (St_ID) REFERENCES Student (St_ID) On Delete Cascade;

--Foreign KEYS for StaffModule
ALTER TABLE StaffModule
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID) On Delete Set Null;

ALTER TABLE StaffModule
ADD FOREIGN KEY (Module_ID) REFERENCES Modules (Module_ID);
```

```
--Foreign KEYS for Performance
ALTER TABLE Performance
ADD FOREIGN KEY (St_ID) REFERENCES Student (St_ID) On Delete Set Null;

ALTER TABLE Performance
ADD FOREIGN KEY (Module_ID) REFERENCES Modules (Module_ID);

--Foreign KEYS for ExamModule
ALTER TABLE examModule
ADD FOREIGN KEY (Module_ID) REFERENCES Modules (Module_ID) On
Delete Cascade;

--Foreign KEYS for BookModule
ALTER TABLE BookModule
ADD FOREIGN KEY (Module_ID) REFERENCES Modules (Module_ID) On
Delete Cascade;

--Foreign KEYS for CourseworkModule
ALTER TABLE CourseworkModule
ADD FOREIGN KEY (Module_ID) REFERENCES Modules (Module_ID) On
Delete Cascade;
```

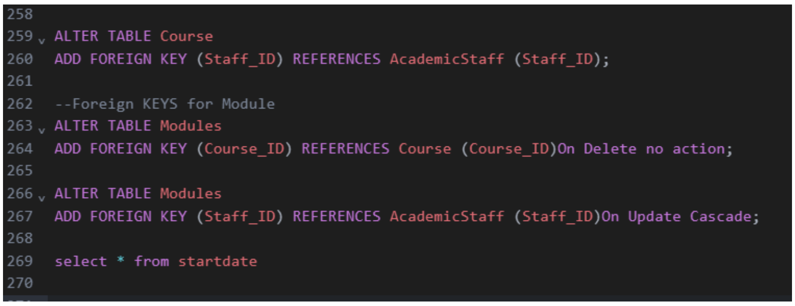
This code establishes the relationship between the primary key in one table and the foreign key in another table.



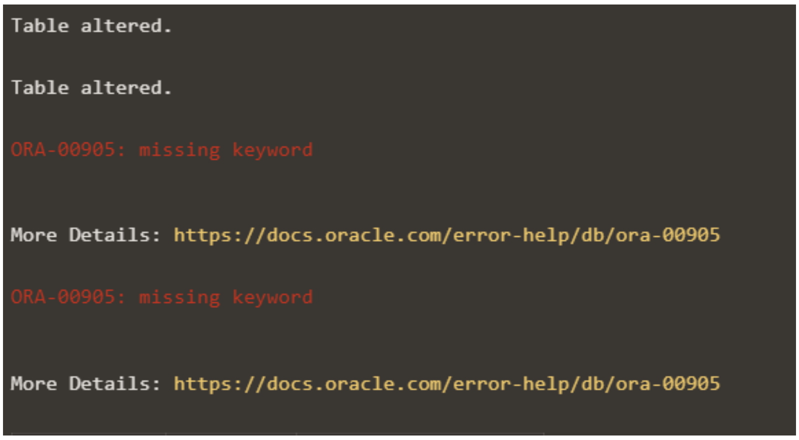
```

ALTER TABLE Course
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID);

--Foreign KEYS for Module
ALTER TABLE Modules
ADD FOREIGN KEY (Course_ID) REFERENCES Course (Course_ID)On Delete Set Null;

ALTER TABLE Modules
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID)On Update Cascade;

select * from startdate


```



```
Table altered.


Table altered.


Table altered.


ORA-00905: missing keyword


More Details: https://docs.oracle.com/error-help/db/ora-00905
```



```
ALTER TABLE Course
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID);

--Foreign KEYS for Module
ALTER TABLE Modules
ADD FOREIGN KEY (Course_ID) REFERENCES Course (Course_ID)On Delete no action;

ALTER TABLE Modules
ADD FOREIGN KEY (Staff_ID) REFERENCES AcademicStaff (Staff_ID)On Update Cascade;

select * from startdate

```



```
Table altered.

Table altered.

ORA-00905: missing keyword


More Details: https://docs.oracle.com/error-help/db/ora-00905

ORA-00905: missing keyword


More Details: https://docs.oracle.com/error-help/db/ora-00905
```

I didn’t include ON DELETE NO ACTION and ON UPDATE CASCADE because either SQL doesn’t recognize these commands, or they are the default behaviour.

3. Data insertion

Enter some test data (approximately 5 – 10 rows) into each table. **(2.5 marks)**

Values for **Department** table:

INSERT INTO Department VALUES ('CDM', 'Computing and Digital Media', '020568492', '020168492', 'Block A', '1234J');

INSERT INTO Department VALUES ('HS', 'Human Sciences', '020577492', '020268492', 'Block C', '1826R');

INSERT INTO Department VALUES ('AAD', 'Art, Architecture and Design', '020368492', '020827492', 'Block E', '1435J');

INSERT INTO Department VALUES ('CIS', 'Computer and Information Systems', '020832492', '020468492', 'Block E', '1235A');

INSERT INTO Department VALUES ('EE', 'Electrical Engineering', '020565492', '020568492', 'Block E', '2234D');

INSERT INTO Department VALUES ('BA', 'Business Administration', '020543292', '020668492', 'Block A', '1596E');

Values for **StartDate** table:

INSERT INTO StartDate VALUES ('CDM', '1234J', '18-AUG-2023');

INSERT INTO StartDate VALUES ('HS', '1826R', '12-SEP-2024');

INSERT INTO StartDate VALUES ('AAD', '1435J', '13-JAN-2022');

INSERT INTO StartDate VALUES ('CIS', '1235A', '28-JUL-2024');

INSERT INTO StartDate VALUES ('EE', '2234D', '18-SEP-2023');

INSERT INTO StartDate VALUES ('BA', '1596E', '1-OCT-2024');

Values for **Course** table:

INSERT INTO Course VALUES ('CS001', 'Computer Science', '4 Years', 'CDM', '12345');

INSERT INTO Course VALUES ('CS002', 'Cyber Security', '3 Years', 'CIS', '28345');

INSERT INTO Course VALUES ('GP001', 'Games Programming', '5 Years', 'AAD', '32345');

INSERT INTO Course VALUES ('PgDIT', 'Post Graduate D in Inform Tech', '2.5 Years', 'CIS', '42345');

INSERT INTO Course VALUES ('DS001', 'Data Science', '3 Years', 'BA', '52345');

Values for **Modules** table:

```sql
INSERT INTO Modules VALUES ('SE123', 'Software Engineering', '15-SEP-2024', '20-MAY-2024', 'CS001', '1434A');

INSERT INTO Modules VALUES ('MM001', 'Multi-Media', '18-SEP-2024', '20-DEC-2024', 'GP001', '2234T');

INSERT INTO Modules VALUES ('RAI01', 'Robotics and AI', '17-SEP-2024', '17-DEC-2024', 'PgDIT', '2214T');

INSERT INTO Modules VALUES ('DB001', 'Database', '18-SEP-2024', '18-MAY-2024', 'CS001', '2224C');

INSERT INTO Modules VALUES ('AE001', 'Aerospace Engineering', '20-SEP-2024', '18-DEC-2024', 'PgDIT', '2254B');

INSERT INTO Modules VALUES ('DSS01', 'Data Science', '18-SEP-2024', '16-DEC-2024', 'GP001', '2264A');

INSERT INTO Modules VALUES ('CSS01', 'Cyber Security', '16-SEP-2024', '22-MAY-2024', 'CS001', '2284G');
```

Values for **AcademicStaff** table:

```sql
INSERT INTO AcademicStaff VALUES ('12345', 'js123', 'John', 'Smith', '101', '25', 'M', '24000', 'L', 'BsC', '221B Baker Street, London NW1 6XE', 'EE', 'course leader');

INSERT INTO AcademicStaff VALUES ('1234J', 'js004', 'James', 'Smith', '103', '32', 'M', '28000', 'L', 'PhD', '10 Downing Street, London SW1A 2AA', 'CDM', 'department manager');

INSERT INTO AcademicStaff VALUES ('1435J', 'jW1001', 'Julia', 'White', '104', '15', 'F', '26000', 'P', 'PhD', '50 Oxford Street, London W1D 1AZ', 'AAD', 'department manager');

INSERT INTO AcademicStaff VALUES ('1235A', 'aw1002', 'Ava', 'Wood', '105', '15', 'F', '32000', 'P', 'BsC', '95 Oxford Street, London W1D 18d', 'CIS', 'department manager');

INSERT INTO AcademicStaff VALUES ('1434A', 'ad001', 'Adam', 'Grid', '106', '18', 'M', '28000', 'SL', 'PhD', '18 Down Street, London SP1A 2AB', 'CIS', 'Module coordinator');

INSERT INTO AcademicStaff VALUES ('2234T', 'ts100', 'Tim', 'Scam', '107', '18', 'M', '28000', 'L', 'Postdoctoral Research', '55 Downtown Street, London Srt 2AB', 'CIS', 'Module coordinator');

INSERT INTO AcademicStaff VALUES ('1596E', 'em100', 'Eva', 'Mind', '108', '16', 'F', '31050', 'SL', 'BsC', '88 Dock Street, London ER1A 8AB', 'BA', 'department manager');

INSERT INTO AcademicStaff VALUES ('1396D', 'dc100', 'Don', 'Cast', '102', '19', 'M', '33200', 'SL', 'PhD', '38 Far Street, London sR1A 4AB', 'EE', 'department manager');

INSERT INTO AcademicStaff VALUES ('1826R', 'rs100', 'Rima', 'Soul', '109', '28', 'F', '30500', 'SL', 'PhD', '58 Tree Street, London EP9A 85B', 'HS', 'department manager');

INSERT INTO AcademicStaff VALUES ('28345', 'od413', 'Olivia', 'Davis', '110', '55', 'F', '24000', 'L', 'Postdoctoral Research', '258B Bank Street, London NW1 6fE', 'CDM', 'course leader');
INSERT INTO AcademicStaff VALUES ('32345', 'ag453', 'Alfred', 'Greco', '111', '36', 'M',
'24800', 'L', 'MsC', '1 Trafalgar Square, London WC2N 5DN', 'HM', 'course leader');

INSERT INTO AcademicStaff VALUES ('42345', 'sw193', 'Sophia', 'Wilson', '112', '25', 'F',
'24080', 'SL', 'PhD', '60 Finchley Road, London NW3 5HN', 'CDM', 'course leader');

INSERT INTO AcademicStaff VALUES ('52345', 'st127', 'Sam', 'Thomas', '131', '22', 'M',
'25600', 'L', 'Postdoctoral Research', '75 Commercial Street, London E1 6BD', 'BA', 'course
leader');

INSERT INTO AcademicStaff VALUES ('2214T', 'td100', 'Tom', 'Din', '114', '18', 'M', '40000',
'P', 'PhD', '55 Downtown Street, London Srt 4XV', 'CIS', 'Module coordinator');

INSERT INTO AcademicStaff VALUES ('2224C', 'cm100', 'Carl', 'Marks', '115', '18', 'F',
'26300', 'L', 'MsC', '30 Hammersmith Broadway, London W6 9YD', 'EE', 'Module coordinator');

INSERT INTO AcademicStaff VALUES ('2254B', 'bs100', 'Bill', 'Scam', '116', '22', 'M', '42000',
'P', 'PhD', '120 Notting Hill Gate, London W11 3QE', 'AAD', 'Module coordinator');

INSERT INTO AcademicStaff VALUES ('2264A', 'ai100', 'Amanda', 'Icam', '117', '33', 'F',
'35000', 'L', 'MsC', '88 Denmark Hill,  London SE5 8EH', 'BA', 'Module coordinator');

INSERT INTO AcademicStaff VALUES ('2284G', 'gs100', 'George', 'Snow', '118', '25', 'M',
'38000', 'L', 'PhD', '12 Downtown Street, London Srt 2AB', 'AAD', 'Module coordinator');
```

Values for **Post** table:

```
INSERT INTO Post VALUES ('L', 'Lecturer');

INSERT INTO Post VALUES ('P', 'Professor');

INSERT INTO Post VALUES ('SL', 'Senior Lecturer');
```

Values for **StaffModule** table:

```
  INSERT INTO StaffModule VALUES ('SE123', '12345', '15');

  INSERT INTO StaffModule VALUES ('SE123', '2284G','11');

  INSERT INTO StaffModule VALUES ('MM001', '2264A', '4');

  INSERT INTO StaffModule VALUES ('RAI01', '2254B', '5');

  INSERT INTO StaffModule VALUES ('DB001', '2224C', '8');

  INSERT INTO StaffModule VALUES ('AE001', '42345', '12');

  INSERT INTO StaffModule VALUES ('DSS01', '1235A', '10');

  INSERT INTO StaffModule VALUES ('CSS01', '1234J', '8');

  INSERT INTO StaffModule VALUES ('RAI01', '32345', '3');

  INSERT INTO StaffModule VALUES ('DB001', '1596E', '7');

  INSERT INTO StaffModule VALUES ('AE001', '2224C', '8');
```

```
INSERT INTO StaffModule VALUES ('DSS01', '42345', '2');
INSERT INTO StaffModule VALUES ('CSS01', '2284G', '8');
```

Values for **Student** table:

```
INSERT INTO Student VALUES ('10051', 'bt001', 'Ben', 'Ten', '18-SEP-2001', 'M', '92
Kensington', 'SW1A 3X', 'Crawley', 'yes', 'CS001');

INSERT INTO Student VALUES ('20404', 'ar001', 'Adam', 'Rich', '22-JUNE-1998', 'M', '57
Kensington', 'SM4M 2S', 'Watford', 'yes', 'CS001');

INSERT INTO Student VALUES ('30466', 'am001', 'Adar', 'Mich', '22-JAN-2003', 'M', '10
Kensington', 'SM6A 2S', 'London', 'no', 'CS001');

INSERT INTO Student VALUES ('40466', 'lm001', 'Lily', 'Much', '22-MAY-1998', 'F', '10
London Road', 'ST6A 2S', 'Watford', 'yes', 'CS001');

INSERT INTO Student VALUES ('50466', 'mm001', 'Matt', 'Mich', '22-JUNE-2004', 'M', '33
London Road', 'SM6A 2S', 'London', 'no', 'CS001');

INSERT INTO Student VALUES ('60466', 'ad001', 'Asma', 'Derr', '22-MAY-1998', 'F', '61
London Road', 'BM6A 2S', 'Watford', 'yes', 'CS001');

INSERT INTO Student VALUES ('70465', 'mb001', 'Mary', 'Buch', '13-JUNE-2002', 'F', '3
Hackney Road', 'SM6A 2S', 'London', 'yes', 'PgDIT');

INSERT INTO Student VALUES ('80469', 'dl001', 'Dasy', 'Lee', '22-MAY-1998', 'F', '42
Hackney Road', 'JM6A 2V', 'London', 'no', 'DS001');

INSERT INTO Student VALUES ('90468', 'pa001', 'Poppy','Adams', '22-JUNE-1998', 'F', '80
Hackney Road', 'GG6A 2R', 'Crawley', 'yes', 'CS001');

INSERT INTO Student VALUES ('41467', 'vd001', 'Vivien', 'Dass', '2-MAY-2002', 'F', '19
Parlia Street', 'SM6A 2S', 'Luton', 'yes', 'CS001');

INSERT INTO Student VALUES ('42466', 'am002', 'Anna', 'Meg', '22-JAN-1998', 'F', '81 Parlia
Street', 'DM6A JS', 'London', 'no', 'PgDIT');

INSERT INTO Student VALUES ('43465', 'dg001', 'Dan', 'Gross', '22-JUNE-2002', 'M', '45
Alexandra Park Road', 'SM6A 2S', 'London', 'yes', 'PgDIT');

INSERT INTO Student VALUES ('44464', 'bh001', 'Ben', 'Hill', '28-JAN-1997', 'M', '75
Alexandra Park Road', 'PS6A 2S', 'Luton', 'no', 'DS001');

INSERT INTO Student VALUES ('45463', 'vs001', 'Victor', 'Sims', '22-JUNE-1998', 'M', '12
Alexandra Park Road', 'EE6A 2S', 'London', 'yes', 'CS001');

INSERT INTO Student VALUES ('46462', 'dd001', 'Dag', 'Dird', '12-MAY-2003', 'M', '1 Parlia
Street', 'SM6A 2S', 'Luton', 'yes', 'CS001');

INSERT INTO Student VALUES ('47461', 'bc001', 'Bill', 'Cook', '15-MAY-1996', 'M', '15 Prelia
Street', 'DM6A 2P', 'Crawley', 'no', 'CS001');
```

Values for **Performance** table:

```
INSERT INTO Performance VALUES ('CS001', '40051', 'PASS');
INSERT INTO Performance VALUES ('MM001', '40051', 'FAIL');
INSERT INTO Performance VALUES ('CS001', '70465', 'PASS');
INSERT INTO Performance VALUES ('MM001', '46462', 'FAIL');
INSERT INTO Performance VALUES ('MM001', '40404', 'PASS');
INSERT INTO Performance VALUES ('CS001', '40404', 'FAIL');
INSERT INTO Performance VALUES ('MM001', '70465', 'PASS');
INSERT INTO Performance VALUES ('DSS01', '47461', 'FAIL');
INSERT INTO Performance VALUES ('DSS01', '46462', 'PASS');
INSERT INTO Performance VALUES ('SE123', '40051', 'FAIL');
INSERT INTO Performance VALUES ('SE123', '46462', 'PASS');
INSERT INTO Performance VALUES ('SE123', '47461', 'PASS');
```

Values for **NextKin** table:

```
INSERT INTO NextKin VALUES ('nk001', 'Emma Wins', '10 Dow Road, Bow, Crawley E8 3AA', '+448512347891', 'Mother', '40051');
INSERT INTO NextKin VALUES ('nk002', 'Olivia Dins', '202 Brow Road, Bow, London E4 3AD', '+448512347892', 'Sister', '47461');
INSERT INTO NextKin VALUES ('nk003', 'Tom Cruise', '100 Bow Road, Bow, Luton E3 3SR', '+448512347893', 'Brother', '46462');
INSERT INTO NextKin VALUES ('nk004', 'Jenna Mils', '22 Bown Road, Bow, London E3 5EA', '+448512347894', 'Mother', '44464');
INSERT INTO NextKin VALUES ('nk005', 'Eva Sins', '85 Higate Hill, Highgate, London N19 5ND', '+448512347895', 'Sister', '43465');
INSERT INTO NextKin VALUES ('nk007', 'John Woos', '25 Highgate Hill, Highgate, London N19 5ND', '+448512347896', 'Father', '45463');
INSERT INTO NextKin VALUES ('nk006', 'Eca Brown', '15 Sighgate Hill, Highgate, Luton N19 5GD', '+448512347897', 'Sister', '42466');
INSERT INTO NextKin VALUES ('nk008', 'Sam Fast', '20 Farringdon Street, Crawley PC4A 4DR', '+448512347898', 'Brother', '41467');
INSERT INTO NextKin VALUES ('nk009', 'Mary Wins', '20 Larringdon Street, London EC4A 4AB', '+448512347899', 'Sister', '90468');
INSERT INTO NextKin VALUES ('nk010', 'Lisa Elins', '32 Farringdon Street, Manchester DC4A 4AL', '+448512747898', 'Mother', '80469');
```

Values for **examModule** table:

```
  INSERT INTO examModule VALUES ('CS001', 'cs99');
  INSERT INTO examModule VALUES ('CS001', 'cs98');
  INSERT INTO examModule VALUES ('DSS01', 'ds99');
  INSERT INTO examModule VALUES ('DSS01', 'ds98');
  INSERT INTO examModule VALUES ('MM001', 'mm99');
  INSERT INTO examModule VALUES ('MM001', 'mm98');
  INSERT INTO examModule VALUES ('MM001', 'mm97');
```

Values for **BookModule** table:

```
  INSERT INTO BookModule  VALUES ('40051448','CS001');
  INSERT INTO BookModule  VALUES ('40022448','DSS01');
  INSERT INTO BookModule  VALUES ('61051448','CS001');
  INSERT INTO BookModule  VALUES ('61022448','DSS01');
  INSERT INTO BookModule  VALUES ('50051448','CS001');
  INSERT INTO BookModule  VALUES ('50022448','MM001');
```

Values for **CourseworkModule** table:

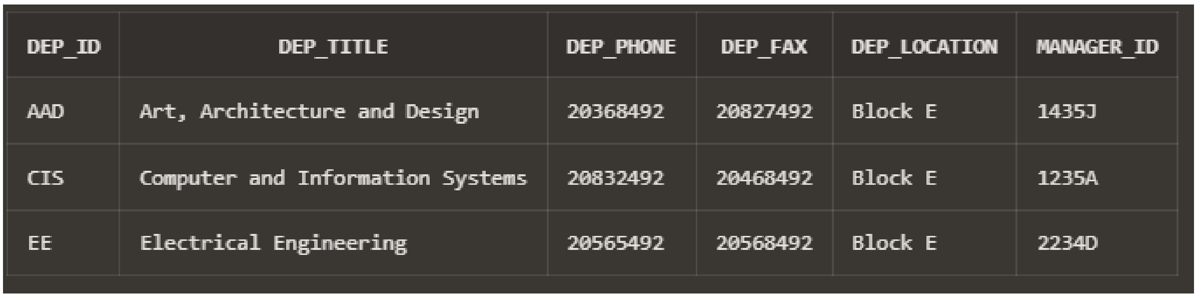
```
  INSERT INTO CourseworkModule VALUES ('CSS01', 'cs50');
  INSERT INTO CourseworkModule VALUES ('MM001', 'mm50');
  INSERT INTO CourseworkModule VALUES ('AE001', 'ae50');
  INSERT INTO CourseworkModule VALUES ('DSS01', 'dss50');
  INSERT INTO CourseworkModule VALUES ('AE001', 'ae51');
  INSERT INTO CourseworkModule VALUES ('RAI01', 'rai50');
```

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.

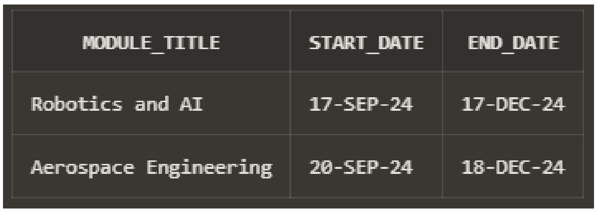
1. Create and save the following query transactions: **(0.5 mark each)**

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



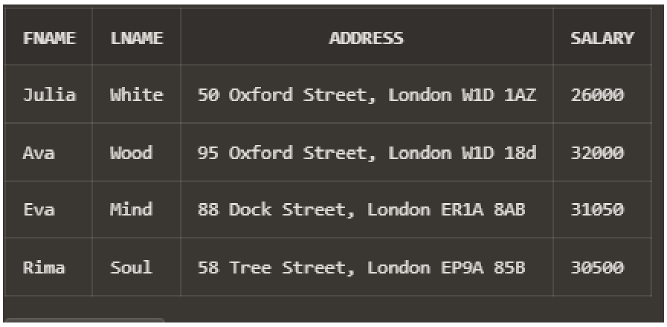
DEP_ID	DEP_TITLE	DEP_PHONE	DEP_FAX	DEP_LOCATION	MANAGER_ID
AAD	Art, Architecture and Design	20368492	20827492	Block E	1435J
CIS	Computer and Information Systems	20832492	20468492	Block E	1235A
EE	Electrical Engineering	20565492	20568492	Block E	2234D

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



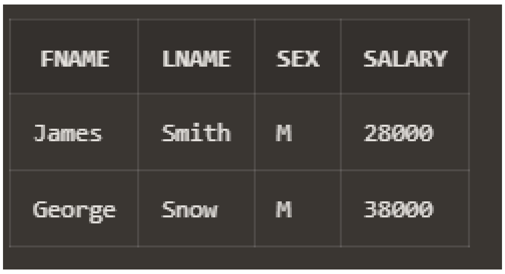
MODULE_TITLE	START_DATE	END_DATE
Robotics and AI	17-SEP-24	17-DEC-24
Aerospace Engineering	20-SEP-24	18-DEC-24

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



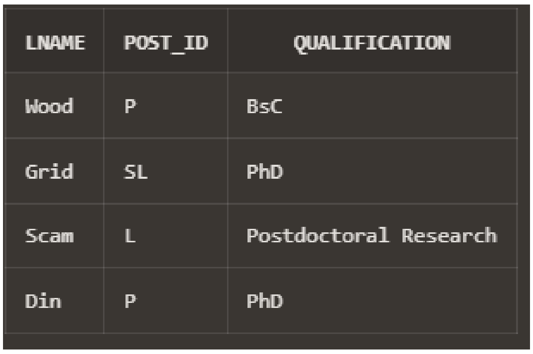
FNAME	LNAME	ADDRESS	SALARY
Julia	White	50 Oxford Street, London W1D 1AZ	26000
Ava	Wood	95 Oxford Street, London W1D 18d	32000
Eva	Mind	88 Dock Street, London ER1A 8AB	31050
Rima	Soul	58 Tree Street, London EP9A 85B	30500

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



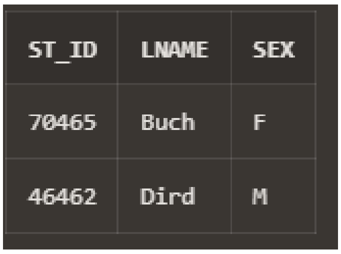
FNAME	LNAME	SEX	SALARY
James	Smith	M	28000
George	Snow	M	38000

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



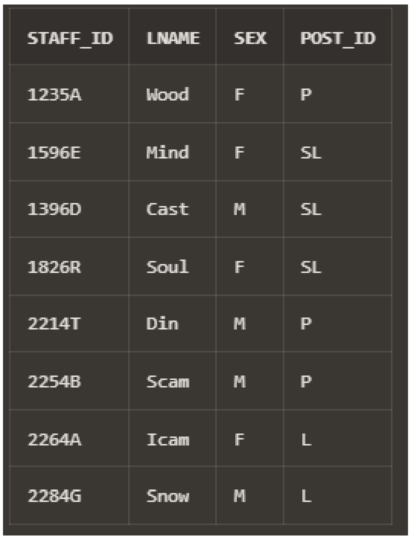
LNAME	POST_ID	QUALIFICATION
Wood	P	BsC
Grid	SL	PhD
Scam	L	Postdoctoral Research
Din	P	PhD

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



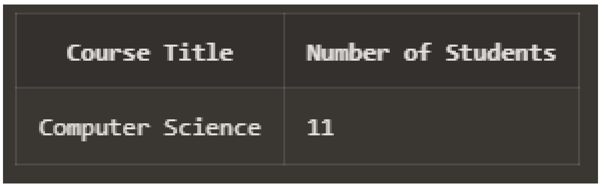
ST_ID	LNAME	SEX
70465	Buch	F
46462	Dird	M

(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.



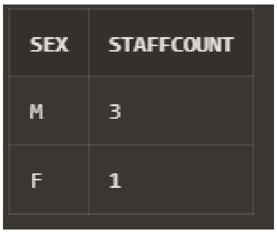
STAFF_ID	LNAME	SEX	POST_ID
1235A	Wood	F	P
1596E	Mind	F	SL
1396D	Cast	M	SL
1826R	Soul	F	SL
2214T	Din	M	P
2254B	Scam	M	P
2264A	Icam	F	L
2284G	Snow	M	L

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



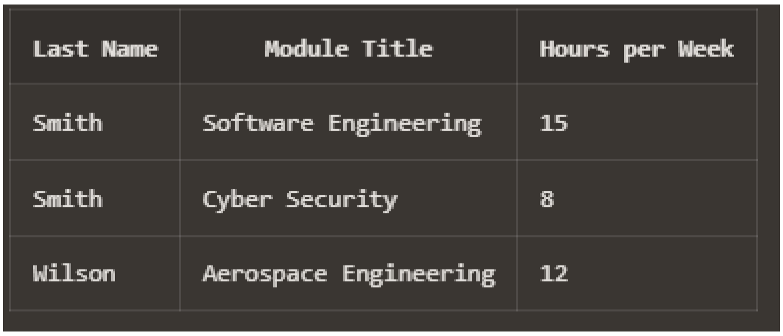
Course Title	Number of Students
Computer Science	11

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



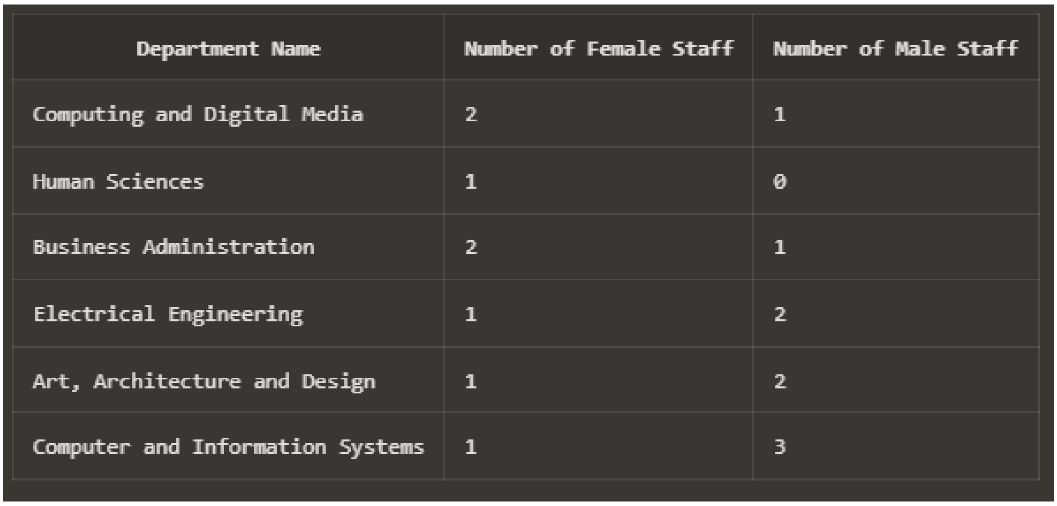
SEX	STAFFCOUNT
M	3
F	1

(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.



Last Name	Module Title	Hours per Week
Smith	Software Engineering	15
Smith	Cyber Security	8
Wilson	Aerospace Engineering	12

(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).



Department Name	Number of Female Staff	Number of Male Staff
Computing and Digital Media	2	1
Human Sciences	1	0
Business Administration	2	1
Electrical Engineering	1	2
Art, Architecture and Design	1	2
Computer and Information Systems	1	3

2. Create a customised form or a report for each saved query. **(4.5 marks)**

a) To list details of all departments located in E Block I used next code:

```
SELECT *

FROM department

WHERE dep_location = 'Block E'
```

b) To list title, start and end dates of all modules run in the PgDIT course I implemented this code:

```
SELECT  Module_Title , Start_Date,  End_Date

FROM  Modules

WHERE Course_ID = 'PgDIT';
```

c) To dysplay name, address, and salary for each female member of academic staff who manages a department I used tins code:

```
SELECT fname, lname,  address, salary

FROM academicstaff

WHERE roles = 'department manager' and sex = 'F'
```

d) To list name, sex, and salary for each lecturer with a PhD degree I used this code:

```
SELECT fname, lname, sex, salary

FROM AcademicStaff

WHERE post_Id = 'L' AND qualification = 'PhD';
```

e) I used the following code to list the last name, post, and qualifications of all academic staff members employed by the CIS department:

```
SELECT  lname, post_Id, qualification

FROM AcademicStaff

WHERE dep_id = 'CIS'
```

f) I used the following code to display the matriculation number, last name, and sex of all students studying the 'multi-media' module, ordered alphabetically by last name:

```
SELECT Student.st_id, Student.lName, Student.Sex

FROM Student

JOIN performance ON Student.st_id = performance.st_id

JOIN Modules ON performance.Module_ID = Modules.Module_ID

WHERE Modules.Module_Id = 'MM001'

ORDER BY Student.lName ASC;
```

g) To find staff number, last name, sex, and post of all academic staff whose salary is greater than the average salary of all academic staff. I used this code:

```
SELECT Staff_ID, lname, sex, post_id

FROM AcademicStaff

WHERE salary > (SELECT AVG(salary) FROM AcademicStaff);
```

h) To display each course with more than 10 students, list course title and the number of students (under an appropriate header). I implemented this code:

```
SELECT Course.Course_Title AS "Course Title", COUNT(Student.st_Id) AS "Number of Students"

FROM Course

JOIN Student ON Course.course_Id = Student.course_Id

GROUP BY Course.Course_Title

HAVING COUNT(Student.st_Id) > 10;
```

i) To print the number of female members of academic staff and the number of male members of academic staff employed by CIS department. I created this code:

```
SELECT Sex, COUNT(*) AS StaffCount

FROM  AcademicStaff

WHERE Dep_id = 'CIS'

GROUP BY Sex;
```

j) To display the last name of each academic staff member, along with the module title and the number of hours they teach, for those who spend more than 6 hours teaching any module, I used this:

```
SELECT  AcademicStaff.lname AS "Last Name",

        Modules.Module_Title AS "Module Title",

        StaffModule.Week_Hours AS "Hours per Week"

FROM AcademicStaff

JOIN StaffModule ON AcademicStaff.Staff_ID = StaffModule.staff_Id

JOIN Modules ON StaffModule.module_Id = Modules.module_Id

WHERE StaffModule.Week_Hours > 6;
```

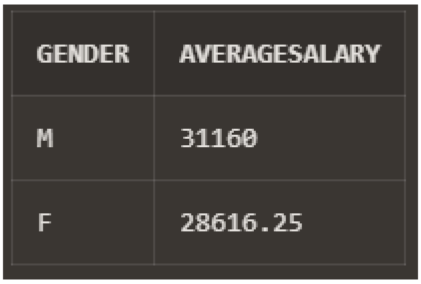
k)To display each department list the department name, and the number of female members of academic staff, and the number of male members of academic staff under appropriate headers (use a **crosstab** query). I made this code:

```
SELECT  Department.Dep_Title AS "Department Name",

SUM(CASE WHEN AcademicStaff.sex = 'F' THEN 1 ELSE 0 END) AS "Number of Female Staff",

SUM(CASE WHEN AcademicStaff.sex = 'M' THEN 1 ELSE 0 END) AS "Number of Male Staff"

FROM Department

JOIN AcademicStaff ON Department.Dep_ID = AcademicStaff.Dep_ID

GROUP BY  Department.Dep_Title;
```

3. Additional Examples

Provide 10 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 QBE/SQL. **(5 marks)**

1) Analyse and compare the average salaries of female and male academic staff.



GENDER	AVERAGESALARY
M	31160
F	28616.25

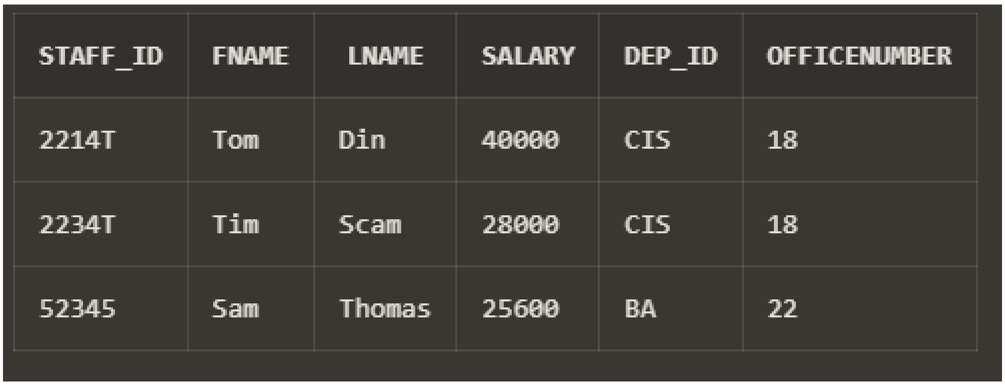
SELECT sex AS Gender,

AVG(salary) AS AverageSalary

FROM AcademicStaff

GROUP BY sex;

2) Retrieve the first name, last name , salary, department ID , and office number of all academic staff members whose first or last name starts with the letter "T".



STAFF_ID	FNAME	LNAME	SALARY	DEP_ID	OFFICENUMBER
2214T	Tom	Din	40000	CIS	18
2234T	Tim	Scam	28000	CIS	18
52345	Sam	Thomas	25600	BA	22

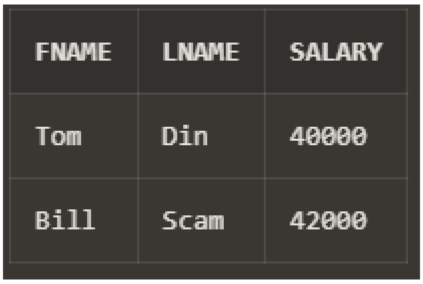
SELECT Staff_ID, fName, lName, salary, Dep_ID, officeNumber

FROM AcademicStaff

WHERE fName LIKE 'T%' OR lName LIKE 'T%'

ORDER BY salary DESC;

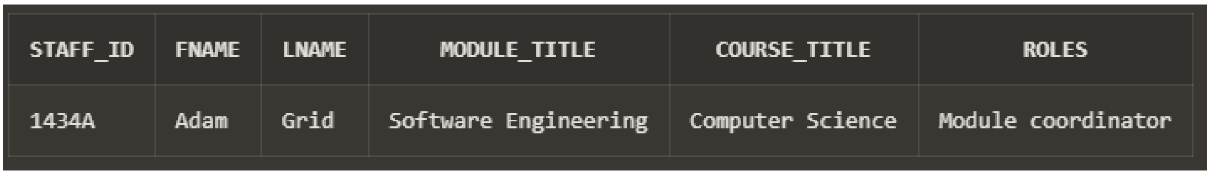
3) Retrieve the first name, last name, and salary of all academic staff members who have a PhD qualification, hold the position of a professor, and are assigned the role of a module coordinator.



FNAME	LNAME	SALARY
Tom	Din	40000
Bill	Scam	42000

```
SELECT fName, lName, salary
FROM AcademicStaff
WHERE qualification = 'PhD'
  AND Post_ID = 'P'
  AND Roles = 'Module coordinator';
```

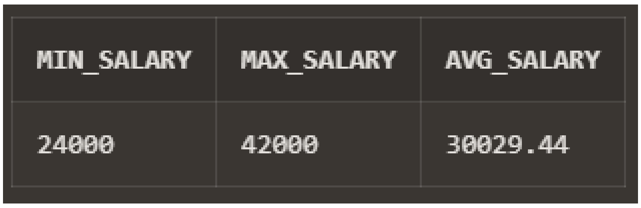
4) Retrieve Staff ID, First Name, Last Name, Module Title, Course Title and the Role of an academic staff members who are responsible for teaching the "Software Engineering" module in the "Computer Science" course.



STAFF_ID	FNAME	LNAME	MODULE_TITLE	COURSE_TITLE	ROLES
1434A	Adam	Grid	Software Engineering	Computer Science	Module coordinator

```
SELECT a.Staff_ID, a.fName, a.lName, m.Module_Title, c.Course_Title, a.Roles
FROM AcademicStaff a
JOIN Modules m ON a.Staff_ID = m.Staff_ID
JOIN Course c ON m.Course_ID = c.Course_ID
WHERE m.Module_Title = 'Software Engineering'
  AND c.Course_Title = 'Computer Science';
```

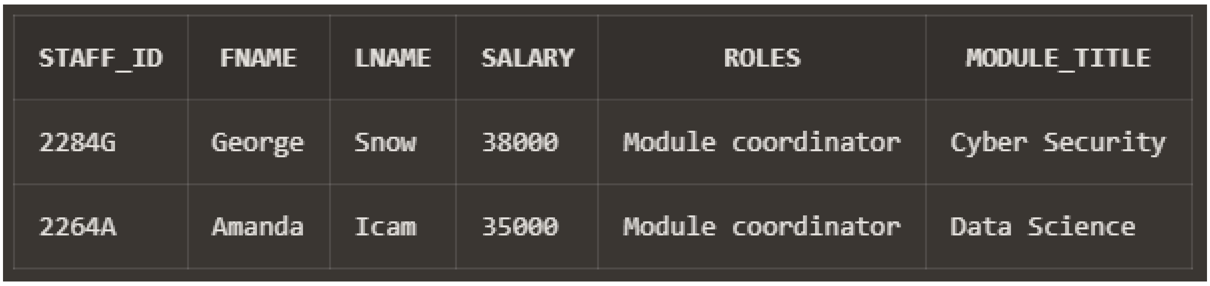
5) Retrieve the minimum, maximum, and average salary of all academic staff. Ensure the average salary is rounded to two decimal places.



MIN_SALARY	MAX_SALARY	AVG_SALARY
24000	42000	30029.44

```
SELECT
   MIN(salary) AS Min_Salary,
   MAX(salary) AS Max_Salary,
   ROUND(AVG(salary), 2) AS Avg_Salary
FROM AcademicStaff;
```

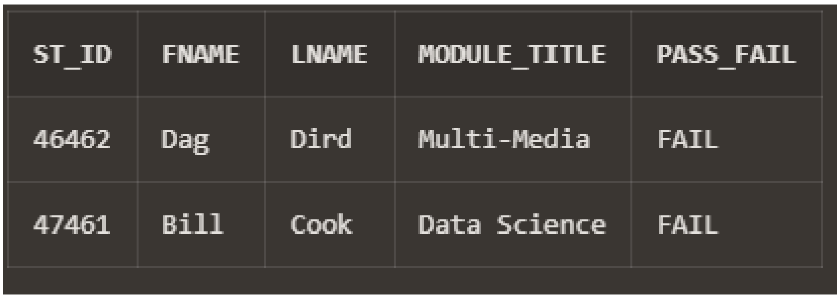
6) Retrieve Lecturer's Name, Salary, Role and Titles of the modules they are teaching whose salaries exceed the average salary of academic staff in descending order.



STAFF_ID	FNAME	LNAME	SALARY	ROLES	MODULE_TITLE
2284G	George	Snow	38000	Module coordinator	Cyber Security
2264A	Amanda	Icam	35000	Module coordinator	Data Science

```
SELECT a.Staff_ID, a.fName, a.lName, a.salary, a.Roles, m.Module_Title
FROM AcademicStaff a
JOIN Modules m ON a.Staff_ID = m.Staff_ID
WHERE a.Post_ID = 'L'
  AND a.salary > (SELECT AVG(salary) FROM AcademicStaff)
ORDER BY a.salary DESC;
```

7) Retrieve the details of all students who have failed a module, including their student number, first name, last name, the module title, and the grade they received.



ST_ID	FNAME	LNAME	MODULE_TITLE	PASS_FAIL
46462	Dag	Dird	Multi-Media	FAIL
47461	Bill	Cook	Data Science	FAIL

SELECT s.st_id, s.fName, s.lName, m.Module_Title, p.Pass_fail

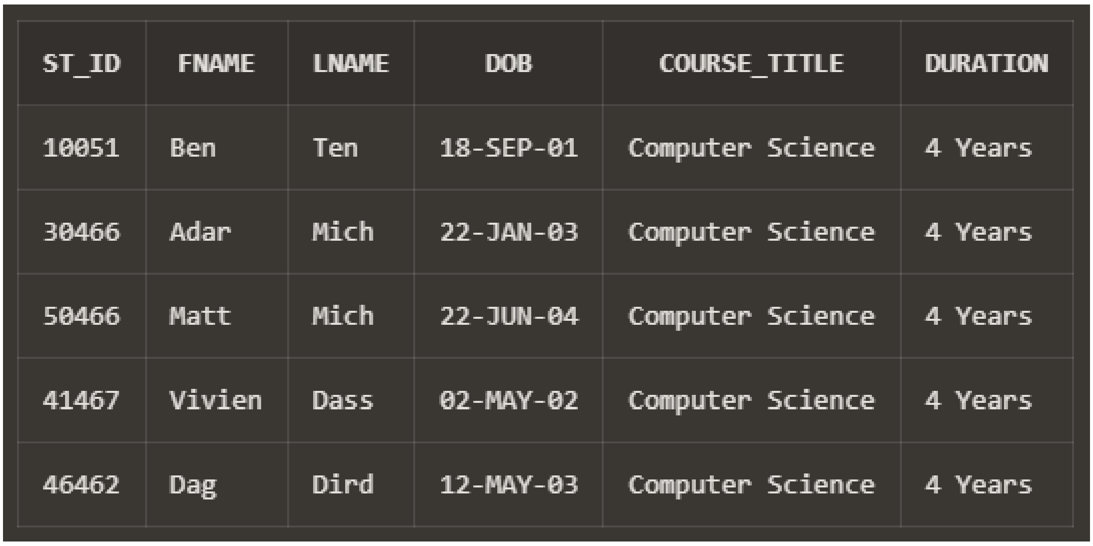
FROM Student s

JOIN Performance p ON s.st_id = p.st_id

JOIN Modules m ON p.Module_ID = m.Module_ID

WHERE p.Pass_fail = 'FAIL';

8) Retrieve the student Number, First Name, Last Name, Date of Birth, Course Title and duration of the course of all students who were born after 2000 and are enrolled in a course with a duration of more than 3 years.



ST_ID	FNAME	LNAME	DOB	COURSE_TITLE	DURATION
10051	Ben	Ten	18-SEP-01	Computer Science	4 Years
30466	Adar	Mich	22-JAN-03	Computer Science	4 Years
50466	Matt	Mich	22-JUN-04	Computer Science	4 Years
41467	Vivien	Dass	02-MAY-02	Computer Science	4 Years
46462	Dag	Dird	12-MAY-03	Computer Science	4 Years

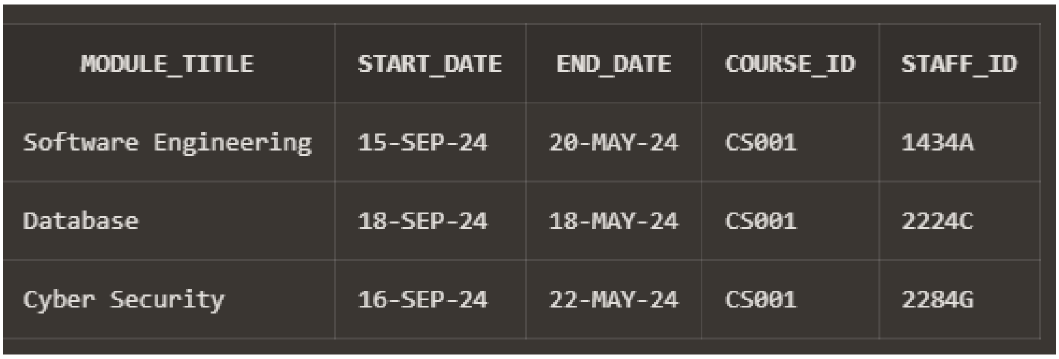
SELECT s.st_id, s.fName, s.lName, s.Dob, c.Course_Title, c.duration

FROM Student s

JOIN Course c ON s.Course_ID = c.Course_ID

WHERE s.Dob > '01-JAN-2000' AND c.Duration > '3 Years';

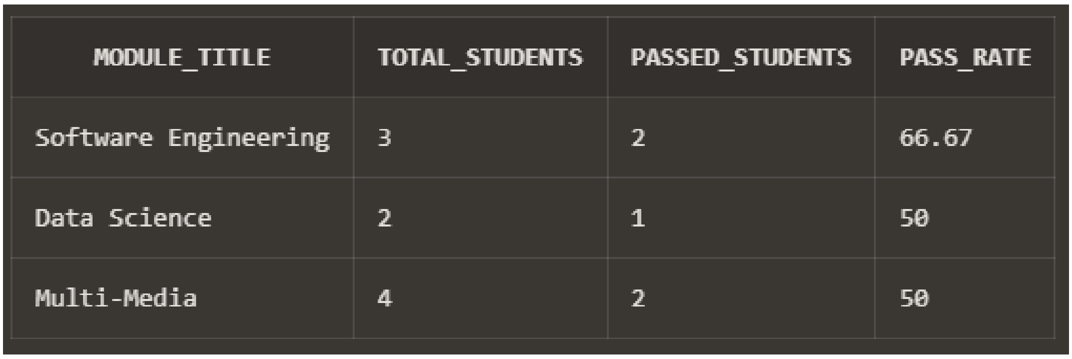
9) List all the modules related to the "Computing and Digital Media" department. The output should display module title, start date, end date, course ID, and staff ID.



MODULE_TITLE	START_DATE	END_DATE	COURSE_ID	STAFF_ID
Software Engineering	15-SEP-24	20-MAY-24	CS001	1434A
Database	18-SEP-24	18-MAY-24	CS001	2224C
Cyber Security	16-SEP-24	22-MAY-24	CS001	2284G

```
SELECT m.Module_Title, m.Start_Date, m.End_Date, m.Course_ID, m.Staff_ID

FROM Modules m

JOIN Course c ON m.Course_ID = c.Course_ID

WHERE c.Dep_ID = 'CDM'
```

10) Retrieve the pass rate for each module in the university. The pass rate should be calculated as the percentage of students who passed each module, based on the total number of students enrolled in the module. Display pass rate in descending order.



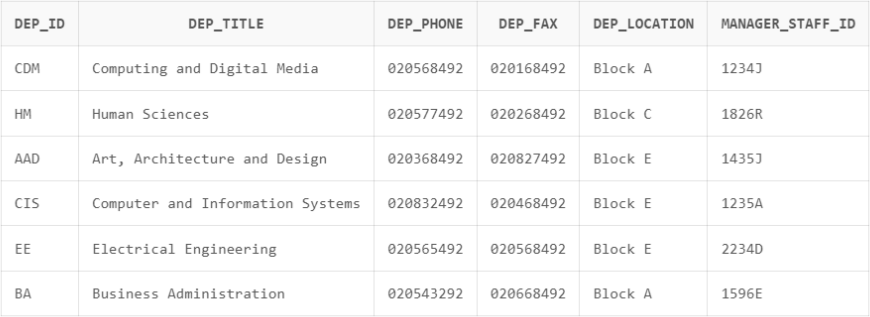
MODULE_TITLE	TOTAL_STUDENTS	PASSED_STUDENTS	PASS_RATE
Software Engineering	3	2	66.67
Data Science	2	1	50
Multi-Media	4	2	50

```
SELECT M.Module_Title,

    COUNT(P.St_ID) AS Total_Students,

    SUM(CASE WHEN P.Pass_fail = 'PASS' THEN 1 ELSE 0 END) AS Passed_Students,

    ROUND(SUM(CASE WHEN P.Pass_fail = 'PASS' THEN 1 ELSE 0 END) / COUNT(P.St_ID) * 100, 2) AS Pass_Rate

FROM Modules M

JOIN Performance P ON M.Module_ID = P.Module_ID

GROUP BY M.Module_Title

ORDER BY Pass_Rate DESC;
```

Part 4 – Implement Database Application

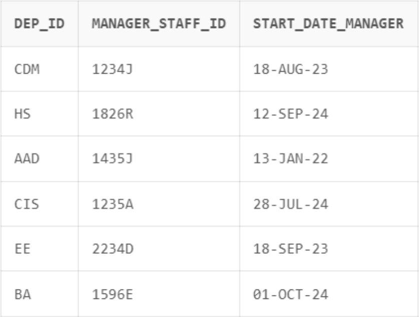
Implement a prototype database application for the University Database. The purpose of this prototype is to allow the Director to provide feedback on your proposed design. The prototype should facilitate the creation, maintenance and querying of records and where appropriate automate various tasks for the user. **(10 marks)**

Department



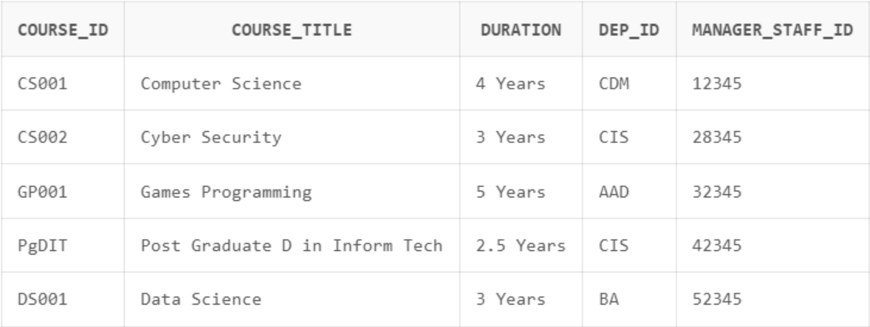
DEP_ID	DEP_TITLE	DEP_PHONE	DEP_FAX	DEP_LOCATION	MANAGER_STAFF_ID
CDM	Computing and Digital Media	020568492	020168492	Block A	1234J
HM	Human Sciences	020577492	020268492	Block C	1826R
AAD	Art, Architecture and Design	020368492	020827492	Block E	1435J
CIS	Computer and Information Systems	020832492	020468492	Block E	1235A
EE	Electrical Engineering	020565492	020568492	Block E	2234D
BA	Business Administration	020543292	020668492	Block A	1596E

StartDate



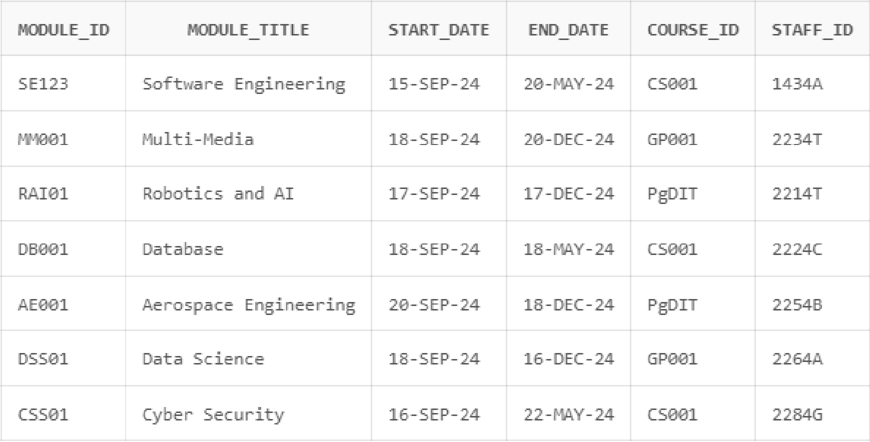
DEP_ID	MANAGER_STAFF_ID	START_DATE_MANAGER
CDM	1234J	18-AUG-23
HS	1826R	12-SEP-24
AAD	1435J	13-JAN-22
CIS	1235A	28-JUL-24
EE	2234D	18-SEP-23
BA	1596E	01-OCT-24

Course



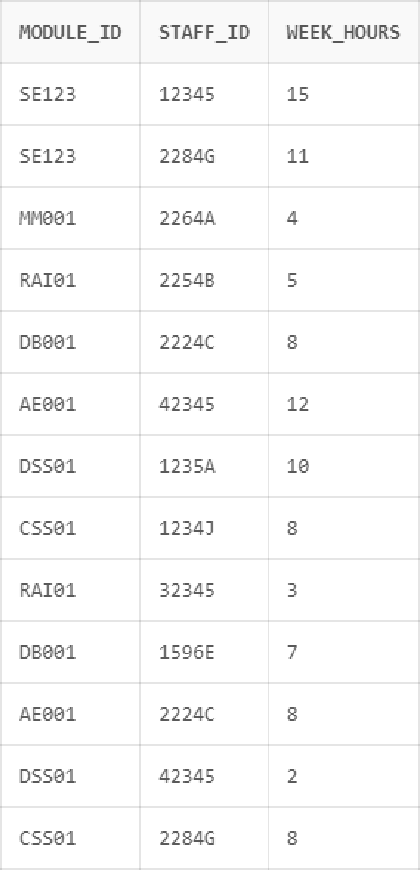
COURSE_ID	COURSE_TITLE	DURATION	DEP_ID	MANAGER_STAFF_ID
CS001	Computer Science	4 Years	CDM	12345
CS002	Cyber Security	3 Years	CIS	28345
GP001	Games Programming	5 Years	AAD	32345
PgDIT	Post Graduate D in Inform Tech	2.5 Years	CIS	42345
DS001	Data Science	3 Years	BA	52345

Module



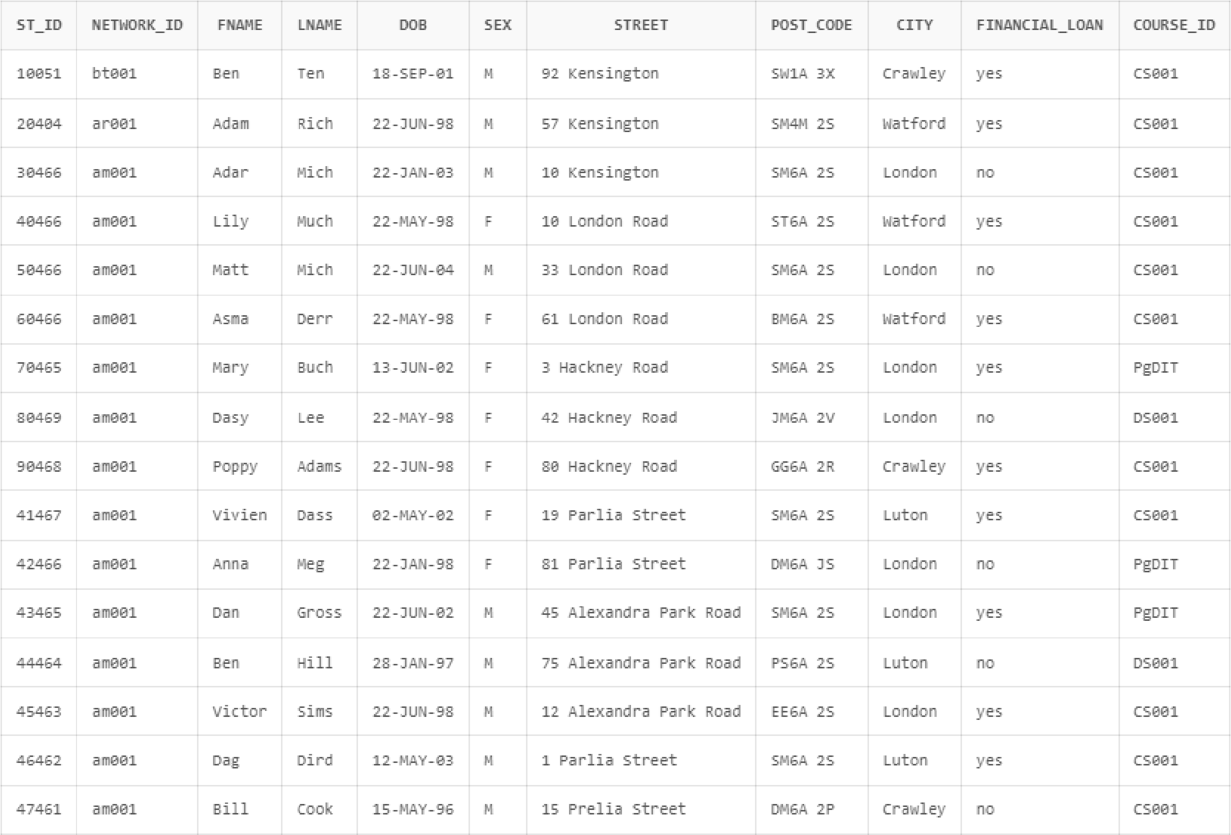
MODULE_ID	MODULE_TITLE	START_DATE	END_DATE	COURSE_ID	STAFF_ID
SE123	Software Engineering	15-SEP-24	20-MAY-24	CS001	1434A
MM001	Multi-Media	18-SEP-24	20-DEC-24	GP001	2234T
RAI01	Robotics and AI	17-SEP-24	17-DEC-24	PgDIT	2214T
DB001	Database	18-SEP-24	18-MAY-24	CS001	2224C
AE001	Aerospace Engineering	20-SEP-24	18-DEC-24	PgDIT	2254B
DSS01	Data Science	18-SEP-24	16-DEC-24	GP001	2264A
CSS01	Cyber Security	16-SEP-24	22-MAY-24	CS001	2284G

StaffModule



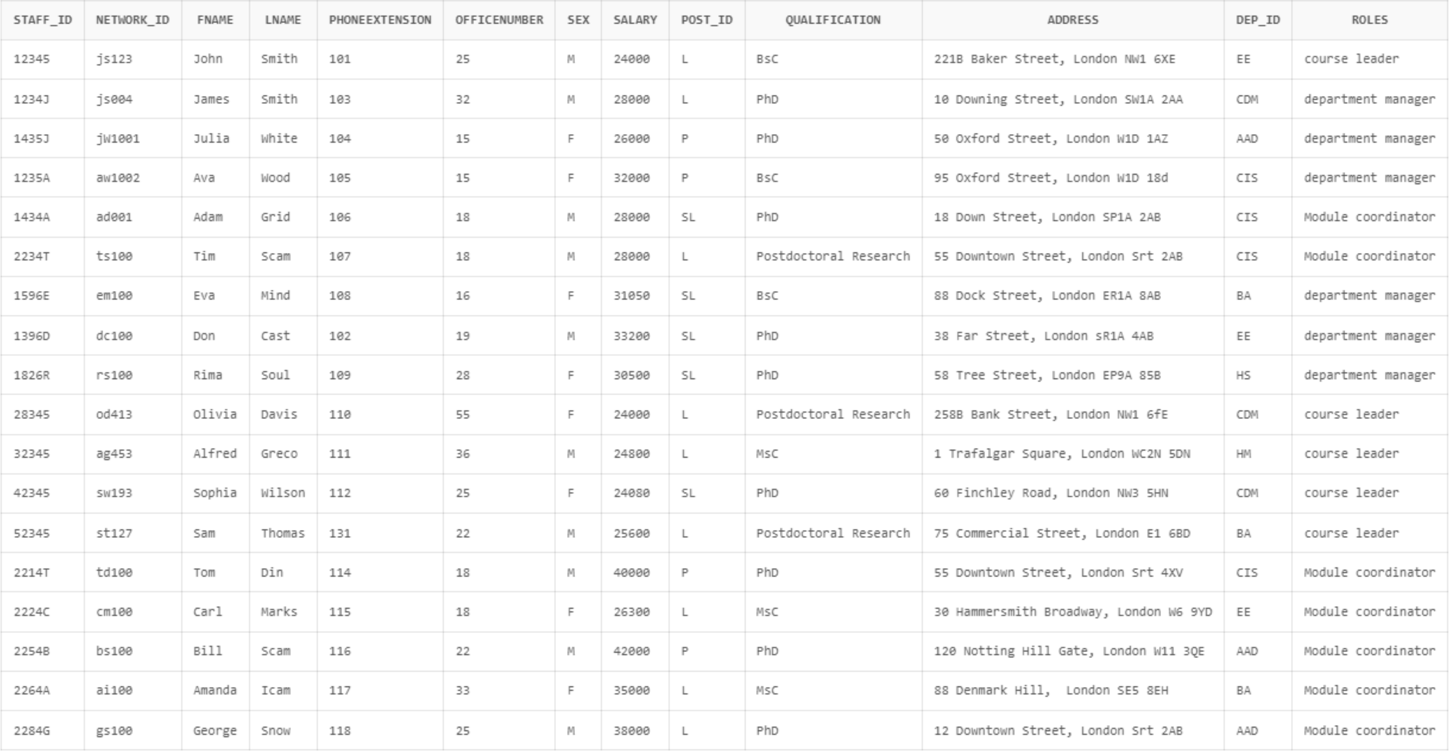
MODULE_ID	STAFF_ID	WEEK_HOURS
SE123	12345	15
SE123	2284G	11
MM001	2264A	4
RAI01	2254B	5
DB001	2224C	8
AE001	42345	12
DSS01	1235A	10
CSS01	1234J	8
RAI01	32345	3
DB001	1596E	7
AE001	2224C	8
DSS01	42345	2
CSS01	2284G	8

Student



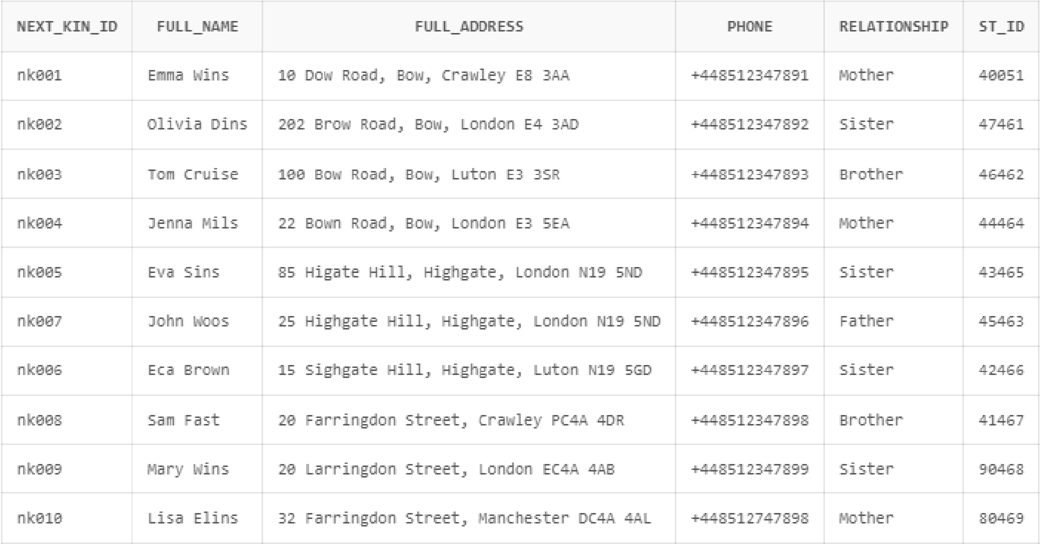
ST_ID	NETWORK_ID	FNAME	LNAME	DOB	SEX	STREET	POST_CODE	CITY	FINANCIAL_LOAN	COURSE_ID
10051	bt001	Ben	Ten	18-SEP-01	M	92 Kensington	SW1A 3X	Crawley	yes	CS001
20404	ar001	Adam	Rich	22-JUN-98	M	57 Kensington	SM4M 2S	Watford	yes	CS001
30466	am001	Adar	Mich	22-JAN-03	M	10 Kensington	SM6A 2S	London	no	CS001
40466	am001	Lily	Much	22-MAY-98	F	10 London Road	ST6A 2S	Watford	yes	CS001
50466	am001	Matt	Mich	22-JUN-04	M	33 London Road	SM6A 2S	London	no	CS001
60466	am001	Asma	Derr	22-MAY-98	F	61 London Road	BM6A 2S	Watford	yes	CS001
70465	am001	Mary	Buch	13-JUN-02	F	3 Hackney Road	SM6A 2S	London	yes	PgDIT
80469	am001	Dasy	Lee	22-MAY-98	F	42 Hackney Road	JM6A 2V	London	no	DS001
90468	am001	Poppy	Adams	22-JUN-98	F	80 Hackney Road	GG6A 2R	Crawley	yes	CS001
41467	am001	Vivien	Dass	02-MAY-02	F	19 Parlia Street	SM6A 2S	Luton	yes	CS001
42466	am001	Anna	Meg	22-JAN-98	F	81 Parlia Street	DM6A JS	London	no	PgDIT
43465	am001	Dan	Gross	22-JUN-02	M	45 Alexandra Park Road	SM6A 2S	London	yes	PgDIT
44464	am001	Ben	Hill	28-JAN-97	M	75 Alexandra Park Road	PS6A 2S	Luton	no	DS001
45463	am001	Victor	Sims	22-JUN-98	M	12 Alexandra Park Road	EE6A 2S	London	yes	CS001
46462	am001	Dag	Dird	12-MAY-03	M	1 Parlia Street	SM6A 2S	Luton	yes	CS001
47461	am001	Bill	Cook	15-MAY-96	M	15 Prelia Street	DM6A 2P	Crawley	no	CS001

AcademiStaff



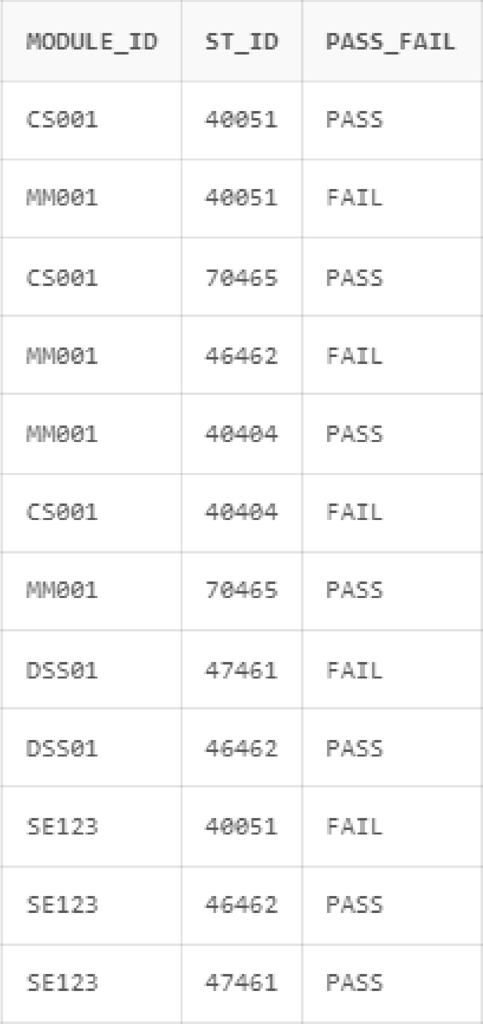
STAFF_ID	NETWORK_ID	FNAME	LNAME	PHONEEXTENSION	OFFICENUMBER	SEX	SALARY	POST_ID	QUALIFICATION	ADDRESS	DEP_ID	ROLES
12345	js123	John	Smith	101	25	M	24000	L	BsC	221B Baker Street, London NW1 6XE	EE	course leader
1234J	js004	James	Smith	103	32	M	28000	L	PhD	10 Downing Street, London SW1A 2AA	CDM	department manager
1435J	jW1001	Julia	White	104	15	F	26000	P	PhD	50 Oxford Street, London W1D 1AZ	AAD	department manager
1235A	aw1002	Ava	Wood	105	15	F	32000	P	BsC	95 Oxford Street, London W1D 18d	CIS	department manager
1434A	ad001	Adam	Grid	106	18	M	28000	SL	PhD	18 Down Street, London SP1A 2AB	CIS	Module coordinator
2234T	ts100	Tim	Scam	107	18	M	28000	L	Postdoctoral Research	55 Downtown Street, London Srt 2AB	CIS	Module coordinator
1596E	em100	Eva	Mind	108	16	F	31050	SL	BsC	88 Dock Street, London ER1A 8AB	BA	department manager
1396D	dc100	Don	Cast	102	19	M	33200	SL	PhD	38 Far Street, London sR1A 4AB	EE	department manager
1826R	rs100	Rima	Soul	109	28	F	30500	SL	PhD	58 Tree Street, London EP9A 85B	HS	department manager
28345	od413	Olivia	Davis	110	55	F	24000	L	Postdoctoral Research	258B Bank Street, London NW1 6fE	CDM	course leader
32345	ag453	Alfred	Greco	111	36	M	24800	L	MsC	1 Trafalgar Square, London WC2N 5DN	HM	course leader
42345	sw193	Sophia	Wilson	112	25	F	24080	SL	PhD	60 Finchley Road, London NW3 5HN	CDM	course leader
52345	st127	Sam	Thomas	131	22	M	25600	L	Postdoctoral Research	75 Commercial Street, London E1 6BD	BA	course leader
2214T	td100	Tom	Din	114	18	M	40000	P	PhD	55 Downtown Street, London Srt 4XV	CIS	Module coordinator
2224C	cm100	Carl	Marks	115	18	F	26300	L	MsC	30 Hammersmith Broadway, London W6 9YD	EE	Module coordinator
2254B	bs100	Bill	Scam	116	22	M	42000	P	PhD	120 Notting Hill Gate, London W11 3QE	AAD	Module coordinator
2264A	ai100	Amanda	Icam	117	33	F	35000	L	MsC	88 Denmark Hill, London SE5 8EH	BA	Module coordinator
2284G	gs100	George	Snow	118	25	M	38000	L	PhD	12 Downtown Street, London Srt 2AB	AAD	Module coordinator

NextKin



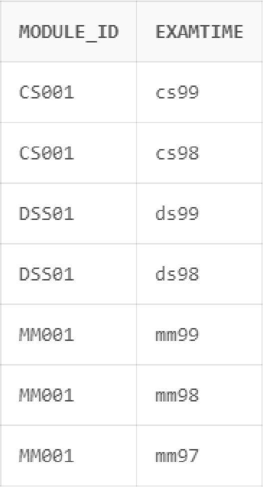
NEXT_KIN_ID	FULL_NAME	FULL_ADDRESS	PHONE	RELATIONSHIP	ST_ID
nk001	Emma Wins	10 Dow Road, Bow, Crawley E8 3AA	+448512347891	Mother	40051
nk002	Olivia Dins	202 Brow Road, Bow, London E4 3AD	+448512347892	Sister	47461
nk003	Tom Cruise	100 Bow Road, Bow, Luton E3 3SR	+448512347893	Brother	46462
nk004	Jenna Mils	22 Bown Road, Bow, London E3 5EA	+448512347894	Mother	44464
nk005	Eva Sins	85 Higate Hill, Highgate, London N19 5ND	+448512347895	Sister	43465
nk007	John Woos	25 Highgate Hill, Highgate, London N19 5ND	+448512347896	Father	45463
nk006	Eca Brown	15 Sighgate Hill, Highgate, Luton N19 5GD	+448512347897	Sister	42466
nk008	Sam Fast	20 Farringdon Street, Crawley PC4A 4DR	+448512347898	Brother	41467
nk009	Mary Wins	20 Larringdon Street, London EC4A 4AB	+448512347899	Sister	90468
nk010	Lisa Elins	32 Farringdon Street, Manchester DC4A 4AL	+448512747898	Mother	80469

Performance



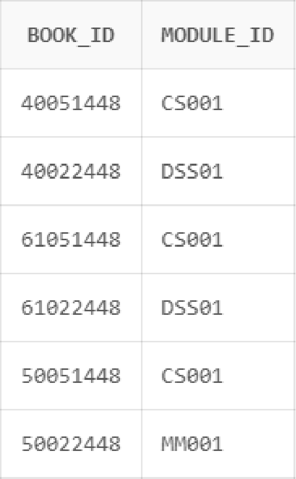
MODULE_ID	ST_ID	PASS_FAIL
CS001	40051	PASS
MM001	40051	FAIL
CS001	70465	PASS
MM001	46462	FAIL
MM001	40404	PASS
CS001	40404	FAIL
MM001	70465	PASS
DSS01	47461	FAIL
DSS01	46462	PASS
SE123	40051	FAIL
SE123	46462	PASS
SE123	47461	PASS

ExamModule



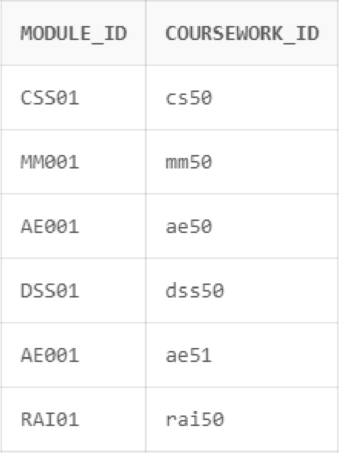
MODULE_ID	EXAMTIME
CS001	cs99
CS001	cs98
DSS01	ds99
DSS01	ds98
MM001	mm99
MM001	mm98
MM001	mm97

BookModule



BOOK_ID	MODULE_ID
40051448	CS001
40022448	DSS01
61051448	CS001
61022448	DSS01
50051448	CS001
50022448	MM001

CourseworkModule



MODULE_ID	COURSEWORK_ID
CSS01	cs50
MM001	mm50
AE001	ae50
DSS01	dss50
AE001	ae51
RAI01	rai50

I assume that the tables Book, Exam, and Coursework already exist with all the atributes inside, which is why only their IDs are referenced in the tables shown here.

Part 5 – Demonstrate Database Application

You are required to demonstrate your database application to your Tutor during your usual tutorial time in Week 12. **(5 marks) Video presentation must be uploaded into WebLearn along with your report**.

Conclusion

This project provided a valuable opportunity to explore the fundamentals of databases and gain a clear understanding of how to design an efficient database from the ground up. One of the main challenges was performing normalization, which involved minimizing data overlap and duplication. Another difficult aspect was creating the Enhanced Entity-Relationship (EER) diagram, as it required defining all the relationships between entities and ensuring logical connections. Setting up the tables and populating them with data was also challenging, especially since it was my first time using SQL development tools. However, querying the database to retrieve data turned out to be the most straightforward and enjoyable part of the project.