LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4001NA PROGRAMMING

50% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: SRIJAN ADHIKARI

Group:L1 C1

London Met ID:19031710

College ID:NP01CP4A190086

Assignment Due Date: 20th December 2020

Assignment Submission Date: 19th December 2020

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

List of Figures:

List of Tables:

Introduction:

Established in 1996, Islington College is a well-known all-over country for developing Industry ready graduates. It is centrally located in Kathmandu at KamalPokhari.

Partnered with one of the oldest institutions of London i.e., London Metropolitan University, Islington relies on a modern approach of studies and researches with open access and active learning. It offers various specifications on Business and IT Degree for undergraduates and postgraduates with high standard academics. Its degree and certificates are adorned with London Metropolitan University.

With a diversity of young talents and experienced faculties, Islington takes an innovative approach to provide a friendly environment for Students. With numbers of blocks, labs, learning zones, and cafeterias Islington facilitates students with the best place to work, study, and rest under a single roof.

Different events and carnivals keep the college environment vibrant and happening. With numbers of past events happened inside and outside of college, Islington offers refreshening events such as Carnivals, trips, fairs, sports, and a lot more celebrations quite often.

With the vision of being the most recognized and prestigious private college in Nepal, it has the following aims and objectives:

- To provide standard quality of British Education at affordable fees.
- To provide up to date education for students to cope up with current trends.
- To foster students' abilities to research and evaluate.
- To produce empowered graduates with self-service tools and solutions.
- To produce leaders who could take ownership of their profession.
- To foster the professional development of students with the help of Industry Expert's interactions.

Business Activities and Operations:

Islington College is a private institution in Nepal that falls under ING Group which is an investment company that focuses on innovating higher education in Nepal by blending international qualifications with local contexts. It performs following Business Activities and operations that matches the British standard in Nepal.

1. Islington facilitates students and faculties with 6 blocks,10 computer labs, 3 lecture halls, 3 cafeterias, 4 seminar rooms, 8 tutorial rooms,2 audio video studio and 2 learning zones.
2. All of the rooms are well facilitated with ACs and other essentials.
3. There are different halls and rooms where classes are carried out. Classes keep changing according to the routine or schedule.
4. There's a dedicated parking lot for college's staffs and students for all kind of vehicle.
5. There's a cafeteria which serves students and staffs with all kind of breakfast or lunch.
6. Islington records details of students and instructors such as Name, Address, DOB, Sex, Salary, Courses and so on.
7. 3 different types of classes are carried out i.e., Lecture, Tutorial and Lab. Lecture halls are for lecture classes, Tutorial rooms are for tutorial classes and computer labs are for lab classes. Seminars are done in Seminar rooms.
8. For every particular specification, a group (e.g., L1C1) is assigned to every student.
9. Timetable is made on the basis of groups. One or more groups are assigned to particular classes.
10. Timetable is different every day.
11. There are different instructors for different courses. There is a Course Leader for every course and a Module Leader for every module.
12. Islington offers bachelor's and master's degree for IT and Business courses. Courses are further divided into different major specifications which consists of multiple modules.

Business Rule:

Business rules are set of approved guidelines or framework within an organization. They are there to bring flexibility, to simplify daily operations in an organization.(Weerasinghe, 2018)

1. The student before enrolling into the college must fill the Enrollment Form with their details such as Name, Address, Enrolled Date, E-mail Address and the course they want to be enrolled to.
2. Similarly, when an instructor is hired, they must fill the Application Form including their details such as Name, Address, Salary, Hire Date which is stored in a database of an Islington College.
3. Fax Number and Secondary Phone Numbers are optional for students or instructors.
4. The address must be well detailed with Country, Province, Street, House Number, list of phone numbers or a list of fax numbers.
5. A student or instructor should mention their address type i.e., Temporary or Permanent.
6. Each course offers of multiple specifications.
7. A student can be enrolled in only one specification of a course.
8. An Instructor can be associated into a single course only.
9. A number of modules are taught in a class whereas each module is taught in particular class.
10. Each instructor is given a role on the basis of which they are getting salary.
11. Student’s fees differ on the basis of specifications.

Entities and Attributes:

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.(Tutorials Point , 2020)

Entities are real world thing, object, person which contains set of attributes.

Attributes are properties of Entities.

Entities	Attributes
Student	StudentID(PK), Name, Sex, DOB, Enrolled Date, StudentAddressID(FK), Email-Address,Marks Obtained.
Student Address	StudentAddressID(PK), Province, Country, City, Street, House Number, Phone Number, Fax.
Instructor	InstructorID(PK), CourseID(FK), Name, Salary, InstructorAddressID(FK),Hire Date, Email-Address,Role.
Instructor Address	InstructorAddressID(PK), Province, Country, City, Street, House Number, Phone Number, Fax.
Course	CourseID(PK), Course Name, Specification Name, Specification ID, Course Fees.
Module	ModuleID(PK), Module Name, ClassID, InstructorID(FK), Class Name.

Table 1 Entities and Attributes

Entity Relationship Diagram (ERD):

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.(Lucid Software Inc., 2020)
The diagram consists of Entities and their attributes showing a conceptual model for a Private College Management System.

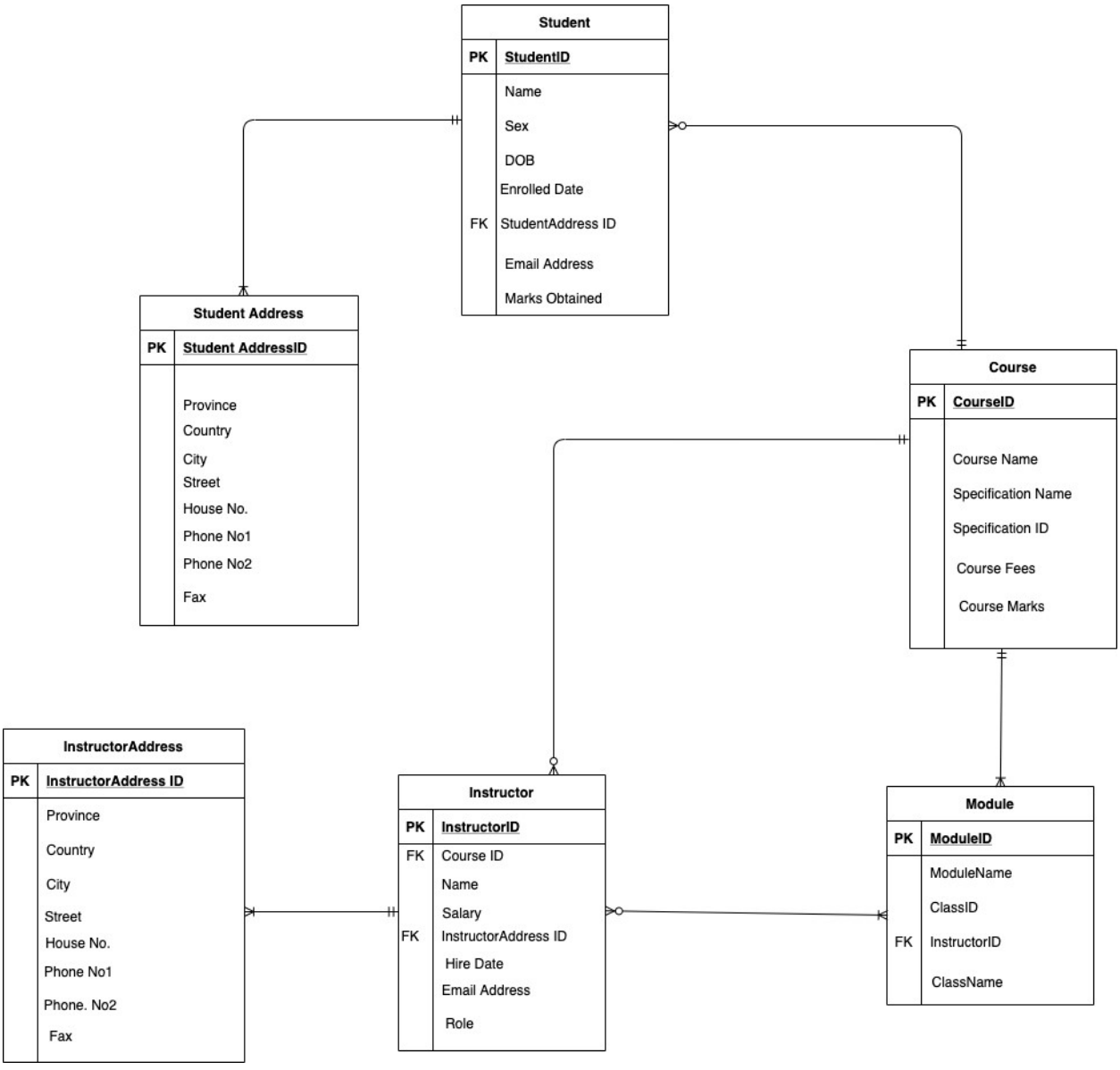

Figure 1 Initial ERD

Assumptions:

- Every Students have Student ID, Name, Sex, DOB, Enrolled Date, Email Address, Marks obtained and Address Details.
- Every Instructors have Instructor ID, Name, Sex, DOB, Hired Date, Salary and Address Details.
- An Instructor cannot be a student at the same time.
- A student can apply for only one degree and one course at a time.
- An instructor or a student have multiple addresses.
- An instructor or student may or may not have multiple phone number.

Problems with Initial ERD:

The ERD is a conceptual representation of a relational database for a Private College Management System. The ERD consists of number of anomalies along with a many-to-many relationship present in the model. The given module displays the presence of many-to-many relationship in between Instructor and module. Since the relational database doesn't allow many-to-many relationship between two entities as it causes the data to be less efficient. There's also a presence of fan trap in the module. The fan trap from course doesn't let to specify a particular instructor who is related to the particular course. These problems cause the database to be less efficient, redundant and in order to remove or minimize these problems, Normalization is done.

Normalization:

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.(Geeks for Geeks, 2019)
To Normalize a database, it is presented in Un-normalized form followed by 1st,2nd and 3rd Normal Forms.

UNF (Un-Normalized Form):

UNF is a preparatory stage for Normalization where there's presence of repeating data or repeating fields and there's no presence of unique keys. Repeating group is
an attribute that can have multiple values
associated with a single instance of some entity(FreeDictionary , 2020)

Scenarios for UNF:

1. A student can be enrolled in only one course resulting it to be a non-repeating group.
2. A student can be enrolled in only one specification of a course as well.
3. Course ID is a unique identifier.
4. A student has multiple addresses resulting it to be a repeating group.
5. A student studies multiple modules within a single specification.
6. An Instructor may be associated with multiple modules.
7. An Instructor has multiple addresses resulting it to be a repeating group as well.

Student (Student ID, Student Name,Sex,DOB, Enrolled Date, E-mail Address, Course_ID, CourseName, Course Fees, Marks Obtained, Specification ID,Specification Name {Student Address ID, Province,Country,City,Street,HouseNo., Phone No1,Phone No 2, Fax},{Module ID, Module Name, ClassID,ClassName ,{InstructorID, Name,Salary,HireDate,E-mail Address,Role,{ InstructorAddress ID, Province,Country,City,Street,HouseNo., Phone No1,PhoneNo2, Fax }}})

1NF (1st Normal Form):

A table is said to be in 1st Normal Form if it has only single valued attribute. 1NF doesn't allow the multi-valued attribute, composite attribute and their combinations.(JavaTPoint, 2018)
After identifications of repeating group from UNF, 1NF focuses on removal of repeating groups.

Scenarios for 1NF

Here, Student ID is a primary key. We can identify details of Students i.e., Name, Sex, DOB, Enrolled date with their Student ID. It is associated with Specification of a course. With Instructor, we can identify Instructor ID as a primary key which is also associated with Student ID, Course ID, and Module ID.
The database is organized in 1st Normal Form as:

Entities:

Student1(StudentID, Name, Sex, DOB, Enrolled Date,E-mailAddress , CourseID, CourseName,CourseFees, Marks Obtained,SpecificationID,SpecificationName)

Specification-1(Specification ID, Specification name, Specification Fees, Student ID*, CourseID*)

StudentAddress1(StudentAddressID, StudentID*, ,Province,Country, City, Street,HouseNo., Phone No1 , PhoneNo2, Fax , CourseID*)

Module1(ModuleID,StudentID*, CourseID*, ModuleName,ClassID,ClassName)

Instructor1(InstructorID, StudentID*, ModuleID*, CourseID*,Name,Salary,HireDate, E-mail Address,Role)

InstructorAddress1(InstructorAddressID, StudentID*, ModuleID*, InstructorID*, CourseID*, Province, Country, City, Street, HouseNumber, Phone No.1, Phone No2, Fax)

2NF (2nd Normal Form):

Despite of 1NF form, the relation still has redundancy. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.(Geeks for Geeks, 2019).

Scenario for 2NF:

Repeating groups that were separated in UNF were removed in 1NF Form. After composite keys are formed in 1NF, partial dependencies and full functional dependencies were separated.

Removal of partial dependency:

For Student Address:

StudentAddressID determines Province, Country, City, Street,HouseNo.,PhoneNo1, Phone No2 ,Fax.

StudentAddressID→Province, Country, City, Street, HouseNo., PhoneNo1, Phone No2,Fax

StudentID→X

CousreID → X

StudentAddressID,StudentID, CourseID→X

For Specification:

SpecificationID determines Specification Name, Fees.

Specification ID → Specification Name, fees

Student ID →

CourseID →

Specification ID, Student ID, CourseID →

For Module:

ModuleID determines Module Name and ClassID, Class Name.

ModuleID→ModuleName,ClassID,ClassName

StudentID→X

CourseID → X

ModuleID,StudentID, CourseID →X

For Instructor:

InstructorID determines Name,Salary,HireDate,E-mail Address,Role.

InstructorID→ Name,Salary,HireDate,E-mail Address,Role.

StudentID→X

ModuleID→X

CourseID → X

StudentID,ModuleID,InstructorID, CourseID →X

For InstructorAddress:

InstructorAddressID determines Province,Country, City, Street,HouseNo., Phone No1,PhoneNo2 and Fax.

InstructorAddressID→ Province,Country, City, Street,HouseNo., Phone No1,PhoneNo2.and Fax.

StudentID→X

InstructorID→X

ModuleID→X

CourseID →X

StudentID,InstructorID,ModuleID,InstructorAddressID, CourseID →X

From a Module Table, CourseID is denormalized from several tables i.e., Student Address, Module,to remove unnecessary data to minimize data redundancy. Similarly, StudentID,ModuleID,InstructorID, CourseID are removed from Instructor table as one copy of such data already exists so that the data is retrieved faster.
An optimization technique which helps to avoid costly joins in a relational database in which we add redundant data to one or more tables is called Denormalization.(Geeks for Geeks, 2019)

Entities:

Student2(StudentID,Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,CourseName,Marks Obtained)

StudentAddressInfo2(StudentAddressID,Province,Country,City,Street,HouseNo.,PhoneNo.,Fax)

StudentAddress2(StudentID*,StudentAddressID*)

Specification-2(Specification ID , Specification Name, Specification Fees)

Specification Info-2(Specification ID*, Student ID*, CourseID*)

ModuleInfo2(ModuleID,ModuleName,ClassID,ClassName)

Module2(ModuleID*,StudentID*)

InstructorInfo2(InstructorID,Name,Salary,HireDate,E-mail Address,Role)

Instructor2(InstructorID*,StudentID*,ModuleID*)

InstructorAddressInfo2(InstructorAddress,Province,Country, City, Street,HouseNo., Phone No1, PhoneNo2 and Fax)

InstructorAddress2(InstructorAddressID*, StudentID*, ModuleID*, InstructorID*)

3NF (3rd Normal Form):

After removal of partial dependency from 2NF, the relation is now

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.(Geeks for Geeks, 2019)

Scenario for 3NF:

After separation of Partial dependencies in 2NF, the relation is now checked for transitive dependency in 3NF given as:

Transitive Dependencies:

For Student:

StudentID determines StudentName,Sex,DOB,Enrolled Date,E-mailAddress , CourseID,CourseName,CourseFees,Marks Obtained,SpecificationID,SpecificationName.

CourseID→CourseName,CourseFees,CourseMarks

Thus can be splitted into StudentID→Name,Sex,DOB,EnrolledDate,E-mail Address,CourseID.

For Module:

ModuleID→ClassID→ClassName

Thus can be splitted into ModuleID→ModuleName,ClassID

Entities:

Student3(StudentID,Name,Sex,DOB,EnrolledDate,E-mailAddress,CourseID*,SpecificationID*, Marks Obtained)

Course3(CourseID,CourseName,CourseMarks)

Specification3(SpecificationID,SpecificationName,Specification Fees)

Specification Info-3(Specification ID*, Student ID*, CourseID*)

StudentAddressInfo3(StudentAddressID, City,Street,HouseNo.,PhoneNo.,Fax)

StudentAddress3(StudentAddressID*,StudentID*)

ModuleInfo3(ModuleID,ModuleName,ClassID*)

Class3(ClassID,ClassName)

Module3(ModuleID*,StudentID*)

InstructorInfo3(InstructorID,Name,Salary,HireDate,E-mail Address,Role)

Instructor3(InstructorID*,StudentID*,ModuleID*)

InstructorAddressInfo3(InstructorAddressID, City,Street,HouseNo.,PhoneNo.,Fax)

InstructorAddress3(InstructorAddressID*, StudentID*,ModuleID*,InstructorID*

Final ERD:

After Normalization, anomalies are removed or reduced resulting the data to be more efficient and less redundant. Initially, an ERD for a Private College Management was shown which was redundant and less efficient. It has anomalies as mentioned earlier. After the Normalization takes place, problems present earlier like fan traps, many to many relations are removed resulting the data to be more organized.

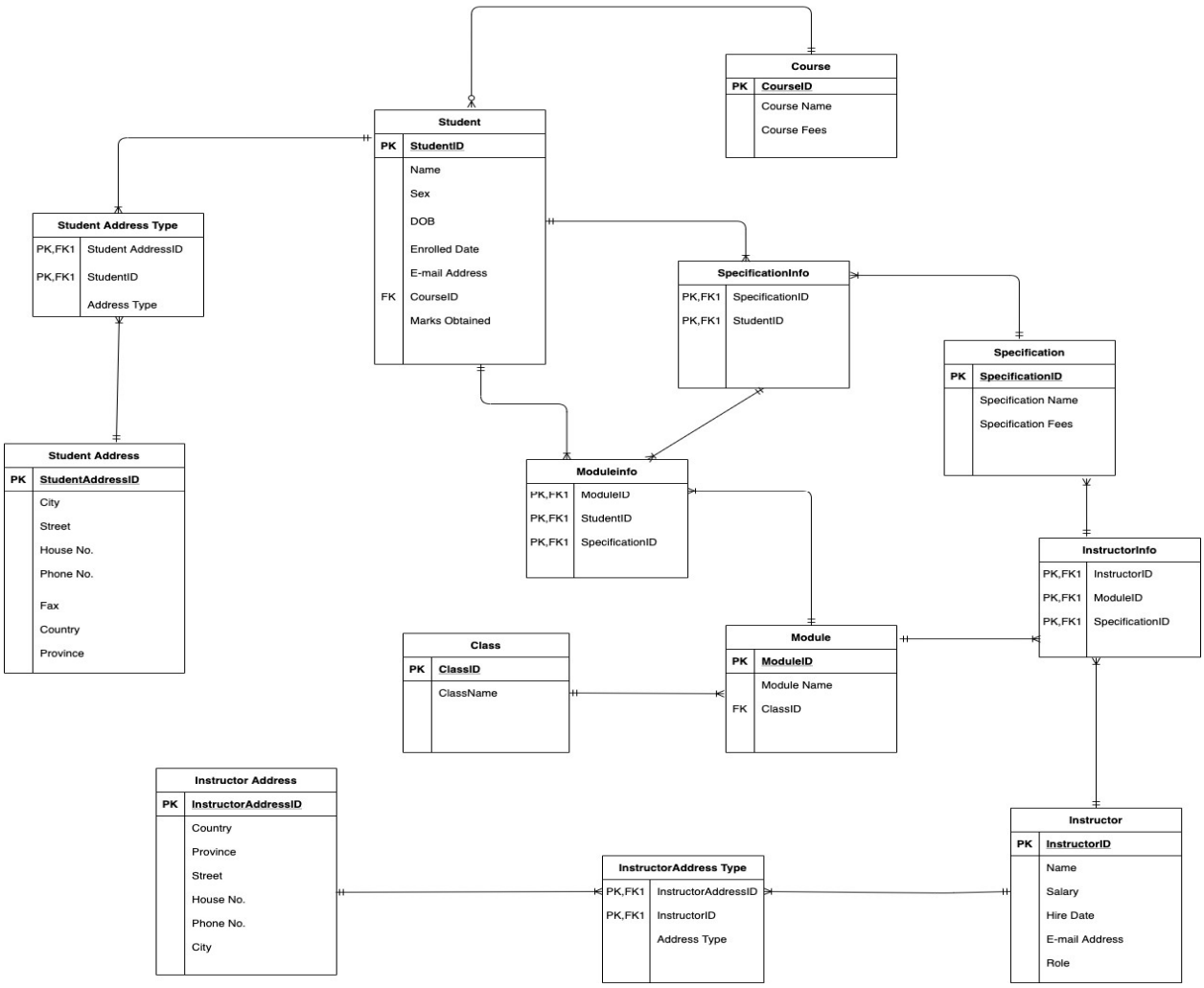

Figure 2 ERD after Normalization

3NF results into formation of new tables and relationships. Student table beholds the detail of student such as Name, Sex, DOB, E-mail and so on. A table called Course beholds the detail of Course such as Course ID, Course Name, and Course Marks. Similarly, Specification beholds the detail of Specifications. Likewise, StudentAddressInfo explains about student's address details. Module beholds attributes like Module ID, Module Name and Class ID. Instructor Table explains the detail of Instructor such as Name, Salary, Age, E-mail and Role. Instructor Address Info has Instructor Address details such as City, Country, Street, Phone Numbers and Fax.

As the database is transitive, composite keys play vital role by joining tables with each other. Composite key is any key having more than one attribute.(Singh) Composite key is when database consists of more than one column having primary keys. A partial dependency occurs when a non-key is dependent on only a part of composite key.

Before Normalization, the initial ERD had anomalies and many to many relationships resulting into less efficient data.

Normalization has been carried out through UNF, 1NF, 2NF and 3NF which generally included following process:

- Separation of Repeating group
- Minimization of redundancy
- Partial dependency by separation of Composite Keys
- Separation of non-key attributes

After Normalization, these problems are removed or reduced resulting the database to be more efficient, functional, consistent and free of redundant data.

Database Implementation

Database implementation is done using Structured Query Language (SQL). It is a standard language for storing, manipulating and retrieving data in Database.(w3schools, 2020)
A user named Islington is created after connecting it with the system. The system is then connected to the created user.

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL>
```

Figure 3 Connect with System

```
SQL> CREATE USER Islington IDENTIFIED BY Islington;

User created.

SQL>
```

Figure 4 Create User

```
SQL> connect Islington/Islington
Connected.
SQL>
```

Figure 5 Connect with User

Table Creations and Description:

1. **Course**

```
CREATE TABLE Course
(CourseID VARCHAR(10),
 CourseName VARCHAR(10) NOT NULL,
  CONSTRAINT  course_id_pk PRIMARY KEY (CourseID));
```

```
SQL> CREATE TABLE Course
  2       (CourseID VARCHAR(10),
  3        CourseName VARCHAR(10) NOT NULL,
  4        CONSTRAINT  course_id_pk PRIMARY KEY (CourseID));

Table created.

SQL>
```

Figure 6 Create table Course

```
SQL> DESC Course;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSEID                                  NOT NULL VARCHAR2(10)
 COURSENAME                                NOT NULL VARCHAR2(10)

SQL>
```

Figure 7 Description of Course

2. Specification

```
CREATE TABLE Specification
 (SpecificationID VARCHAR(10) ,
  SpecificationName VARCHAR(20) NOT NULL,
  SpecificationFees VARCHAR(10) NOT NULL,
  CONSTRAINT specifcation_id_pk PRIMARY KEY (SpecificationID));
```

```
SQL> CREATE TABLE Specification
  2       (SpecificationID VARCHAR(10) ,
  3        SpecificationName VARCHAR(20) NOT NULL,
  4        SpecificationFees VARCHAR(10) NOT NULL,
  5        CONSTRAINT specifcation_id_pk PRIMARY KEY (SpecificationID));

Table created.

SQL>
```

Figure 8 Create table Specification

```
SQL> DESC Specification;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SPECIFICATIONID                           NOT NULL VARCHAR2(10)
 SPECIFICATIONNAME                         NOT NULL VARCHAR2(20)
 SPECIFICATIONFEES                         NOT NULL VARCHAR2(10)

SQL>
```

Figure 9 Describe Specification

3. Student

```
CREATE TABLE Student
(StudentID VARCHAR(10),
Name VARCHAR(20) NOT NULL,
Sex VARCHAR(10) NOT NULL,
DOB DATE NOT NULL,
EnrolledDate DATE NOT NULL,
EmailAddress VARCHAR(25) NOT NULL,
CourseID VARCHAR(10),
MarksObtained NUMBER(5),
CONSTRAINT  student_id_pk PRIMARY KEY (StudentID),
CONSTRAINT  courses_id_pk  FOREIGN KEY(CourseID) REFERENCES
Course(CourseID));
```

```
SQL>
SQL>   CREATE TABLE Student
  2    (StudentID VARCHAR(10),
  3    Name VARCHAR(20) NOT NULL,
  4    Sex VARCHAR(10) NOT NULL,
  5    DOB DATE NOT NULL,
  6    EnrolledDate DATE NOT NULL,
  7    EmailAddress VARCHAR(25) NOT NULL,
  8    CourseID VARCHAR(10),
  9    MarksObtained NUMBER(5),
 10    CONSTRAINT  student_id_pk PRIMARY KEY (StudentID),
 11    CONSTRAINT  courses_id_pk  FOREIGN KEY(CourseID) REFERENCES Course(CourseID));

Table created.
```

Figure 10 Create table Student

```
SQL> DESC Student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENTID                                 NOT NULL VARCHAR2(10)
 NAME                                      NOT NULL VARCHAR2(20)
 SEX                                       NOT NULL VARCHAR2(10)
 DOB                                       NOT NULL DATE
 ENROLLEDDATE                              NOT NULL DATE
 EMAILADDRESS                              NOT NULL VARCHAR2(25)
 COURSEID                                           VARCHAR2(10)
 MARKSOBTAINED                                      NUMBER(5)

SQL>
```

Figure 11 Describe Student

4. Specification Info

CREATE TABLE SpecificationInfo

(SpecificationID VARCHAR(10),

StudentID VARCHAR(10),

CourseID VARCHAR(10),

CONSTRAINT specs_std_pk PRIMARY KEY(SpecificationID,StudentID,CourseID),

CONSTRAINT specs_id_fk FOREIGN KEY (SpecificationID) REFERENCES Specification(SpecificationID),

CONSTRAINT student_id_fk FOREIGN KEY(StudentID) REFERENCES Student(StudentID),

CONSTRAINT course_id_fk FOREIGN KEY(CourseID) REFERENCES Course(CourseID));

```
SQL> CREATE TABLE SpecificationInfo
  2  (SpecificationID VARCHAR(10),
  3      StudentID VARCHAR(10),
  4     CourseID VARCHAR(10),
  5      CONSTRAINT specs_std_pk PRIMARY KEY(SpecificationID,StudentID,CourseID),
  6      CONSTRAINT specs_id_fk FOREIGN KEY (SpecificationID) REFERENCES Specification(SpecificationID),
  7      CONSTRAINT student_id_fk FOREIGN KEY(StudentID) REFERENCES Student(StudentID),
  8       CONSTRAINT course_id_fk FOREIGN KEY(CourseID) REFERENCES Course(CourseID));

Table created.

SQL>
```

Figure 12 Create table Specification

```
SQL> DESC SpecificationInfo;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SPECIFICATIONID                           NOT NULL VARCHAR2(10)
 STUDENTID                                 NOT NULL VARCHAR2(10)
 COURSEID                                  NOT NULL VARCHAR2(10)

SQL>
```

Figure 13 Describe Specification

5. Instructor

```
CREATE TABLE Instructor
(InstructorID VARCHAR(10),
 Name VARCHAR(20) NOT NULL,
 Salary NUMBER(10) NOT NULL,
 HireDate DATE NOT NULL,
 EmailAddress VARCHAR(25),
 Role VARCHAR(20) NOT NULL,
CONSTRAINT  instructor_id_pk PRIMARY KEY (InstructorID));
```

```
SQL> CREATE TABLE Instructor
  2  (InstructorID VARCHAR(10),
  3  Name VARCHAR(20) NOT NULL,
  4  Salary NUMBER(10) NOT NULL,
  5  HireDate DATE NOT NULL,
  6   EmailAddress VARCHAR(25),
  7   Role VARCHAR(20) NOT NULL,
  8   CONSTRAINT  instructor_id_pk PRIMARY KEY (InstructorID));

Table created.

SQL>
```

Figure 14 Create table Instructor

```
SQL> DESC Instructor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTORID                              NOT NULL VARCHAR2(10)
 NAME                                      NOT NULL VARCHAR2(20)
 SALARY                                    NOT NULL NUMBER(10)
 HIREDATE                                  NOT NULL DATE
 EMAILADDRESS                                       VARCHAR2(25)
 ROLE                                      NOT NULL VARCHAR2(20)

SQL>
```

Figure 15 Describe Instructor

6. Instructor Address

```
CREATE TABLE InstructorAddress
   (InstructorAddressID VARCHAR(10),
  Country VARCHAR(20) NOT NULL,
  Province VARCHAR(20) NOT NULL,
  City VARCHAR(20) NOT NULL,
  Street VARCHAR(20) NOT NULL,
  HouseNumber NUMBER(10) NOT NULL,
  PhoneNo1 NUMBER(10) NOT NULL,
  PhoneNo2 NUMBER(10) NOT NULL,
  Fax VARCHAR(20),
  CONSTRAINT  instructor_address_id_pk PRIMARY KEY (InstructorAddressID));
```

```
SQL> CREATE TABLE InstructorAddress
  2        (InstructorAddressID VARCHAR(10),
  3       Country VARCHAR(20) NOT NULL,
  4       Province VARCHAR(20) NOT NULL,
  5       City VARCHAR(20) NOT NULL,
  6       Street VARCHAR(20) NOT NULL,
  7       HouseNumber NUMBER(10) NOT NULL,
  8       PhoneNo1 NUMBER(10) NOT NULL,
  9       PhoneNo2 NUMBER(10) NOT NULL,
 10       Fax VARCHAR(20),
 11       CONSTRAINT  instructor_address_id_pk PRIMARY KEY (InstructorAddressID));

Table created.

SQL>
```

Figure 16 Create table InstructorAddress

```
SQL> DESC InstructorAddress;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTORADDRESSID                       NOT NULL VARCHAR2(10)
 COUNTRY                                   NOT NULL VARCHAR2(20)
 PROVINCE                                  NOT NULL VARCHAR2(20)
 CITY                                      NOT NULL VARCHAR2(20)
 STREET                                    NOT NULL VARCHAR2(20)
 HOUSENUMBER                               NOT NULL NUMBER(10)
 PHONENO1                                  NOT NULL NUMBER(10)
 PHONENO2                                  NOT NULL NUMBER(10)
 FAX                                                VARCHAR2(20)

SQL>
```

Figure 17 Describe InstructorAddress

7. Instructor Address Type

Create Table InstructorAddressType

(InstructorAddressID VARCHAR(10),

InstructorID VARCHAR(10),

AddressType VARCHAR(15),

CONSTRAINT ins_add_pk PRIMARY KEY(InstructorAddressID,InstructorID),

CONSTRAINT ins_add_fk FOREIGN KEY (InstructorAddressID) REFERENCES InstructorAddress(InstructorAddressID),

CONSTRAINT ins_id_fk FOREIGN KEY(InstructorID) REFERENCES Instructor(InstructorID));

```
SQL> Create Table InstructorAddressType
  2  (InstructorAddressID VARCHAR(10),
  3      InstructorID VARCHAR(10),
  4      AddressType VARCHAR(15),
  5      CONSTRAINT ins_add_pk PRIMARY KEY(InstructorAddressID,InstructorID),
  6      CONSTRAINT ins_add_fk FOREIGN KEY (InstructorAddressID) REFERENCES InstructorAddress(InstructorAddressID),
  7      CONSTRAINT ins_id_fk FOREIGN KEY(InstructorID) REFERENCES Instructor(InstructorID));

Table created.

SQL>
```

Figure 18 Create Table InstrctorAddressType

```
SQL> DESC InstructorAddressType;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTORADDRESSID                       NOT NULL VARCHAR2(10)
 INSTRUCTORID                              NOT NULL VARCHAR2(10)
 ADDRESSTYPE                                        VARCHAR2(15)

SQL>
```

Figure 19 Describe InstructorAddress Type

8. Class

CREATE TABLE Class

(ClassID VARCHAR(10),

ClassName VARCHAR(20) NOT NULL,

CONSTRAINT class_id_pk PRIMARY KEY (ClassID));

```
SQL> CREATE TABLE Class
  2        (ClassID VARCHAR(10),
  3        ClassName VARCHAR(20) NOT NULL,
  4        CONSTRAINT  class_id_pk PRIMARY KEY (ClassID));

Table created.

SQL>
```

Figure 20 Create table Class

```
SQL> DESC Class;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CLASSID                                   NOT NULL VARCHAR2(10)
 CLASSNAME                                 NOT NULL VARCHAR2(20)

SQL>
```

Figure 21 Describe Class

9. Module

CREATE TABLE Module

(ModuleID VARCHAR(10),

ModuleName VARCHAR(20) NOT NULL,

ClassID VARCHAR(10),

CONSTRAINT module_id_pk PRIMARY KEY (ModuleID),

CONSTRAINT class_id_fk FOREIGN KEY(ClassID) REFERENCES Class(ClassID));

```
SQL> CREATE TABLE Module
  2       (ModuleID VARCHAR(10),
  3       ModuleName VARCHAR(20) NOT NULL,
  4       ClassID VARCHAR(10),
  5       CONSTRAINT  module_id_pk PRIMARY KEY (ModuleID),
  6        CONSTRAINT  class_id_fk FOREIGN KEY(ClassID) REFERENCES Class(ClassID));

Table created.

SQL>
```

Figure 22 Create table Module

```
SQL> DESC Module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODULEID                                  NOT NULL VARCHAR2(10)
 MODULENAME                                NOT NULL VARCHAR2(20)
 CLASSID                                            VARCHAR2(10)

SQL>
```

Figure 23 Describe Module

10. Instructor Info

CREATE Table InstructorInfo
(InstructorID VARCHAR(10),
ModuleID VARCHAR(10),
SpecificationID VARCHAR(10),
CONSTRAINT modulespecs_inst_pk PRIMARY KEY(InstructorID,ModuleID,SpecificationID),
CONSTRAINT ins_id1_fk FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID),
CONSTRAINT module_id_fk FOREIGN KEY(ModuleID) REFERENCES Module(ModuleID),
CONSTRAINT specs_id1_fk FOREIGN KEY (SpecificationID) REFERENCES Specification(SpecificationID));

```
SQL> CREATE Table InstructorInfo
  2  (InstructorID VARCHAR(10),
  3      ModuleID VARCHAR(10),
  4      SpecificationID VARCHAR(10),
  5      CONSTRAINT modulespecs_inst_pk PRIMARY KEY(InstructorID,ModuleID,SpecificationID),
  6      CONSTRAINT ins_id1_fk FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID),
  7      CONSTRAINT module_id_fk FOREIGN KEY(ModuleID) REFERENCES Module(ModuleID),
  8       CONSTRAINT specs_id1_fk FOREIGN KEY (SpecificationID) REFERENCES Specification(SpecificationID));

Table created.

SQL>
```

Figure 24 Create table InstructorInfo

```
SQL> DESC InstructorInfo;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTORID                              NOT NULL VARCHAR2(10)
 MODULEID                                  NOT NULL VARCHAR2(10)
 SPECIFICATIONID                           NOT NULL VARCHAR2(10)

SQL>
```

Figure 25 Describe InstructorInfo

11. ModuleInfo

```
CREATE TABLE ModuleInfo
 (ModuleID VARCHAR(10),
   StudentID VARCHAR(10),
   SpecificationID VARCHAR(10),
   CONSTRAINT module_stdspecs_id_pk PRIMARY KEY(ModuleID,StudentID),
   CONSTRAINT module_id2_fk FOREIGN KEY (ModuleID) REFERENCES
Module(ModuleID),
   CONSTRAINT specs_id2_fk FOREIGN KEY (SpecificationID) REFERENCES
Specification(SpecificationID),
   CONSTRAINT student_id2_fk FOREIGN KEY(StudentID) REFERENCES
Student(StudentID));
```

```
SQL> CREATE TABLE ModuleInfo
  2   (ModuleID VARCHAR(10),
  3      StudentID VARCHAR(10),
  4      SpecificationID VARCHAR(10),
  5      CONSTRAINT module_stdspecs_id_pk PRIMARY KEY(ModuleID,StudentID),
  6      CONSTRAINT module_id2_fk FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID),
  7      CONSTRAINT specs_id2_fk FOREIGN KEY (SpecificationID) REFERENCES Specification(SpecificationID),
  8      CONSTRAINT student_id2_fk FOREIGN KEY(StudentID) REFERENCES Student(StudentID));

Table created.

SQL>
```

Figure 26 Create table ModuleInfo

```
SQL> DESC ModuleInfo;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODULEID                                  NOT NULL VARCHAR2(10)
 STUDENTID                                 NOT NULL VARCHAR2(10)
 SPECIFICATIONID                                    VARCHAR2(10)

SQL>
```

Figure 27 Describe ModuleInfo

12. Student Address

```
CREATE TABLE StudentAddress
(StudentAddressID VARCHAR(10),
Country VARCHAR(10) NOT NULL,
Province VARCHAR(20) NOT NULL,
City VARCHAR(20) NOT NULL,
Street VARCHAR(20) NOT NULL,
HouseNumber NUMBER(10) NOT NULL,
PhoneNo1  NUMBER(10) NOT NULL,
PhoneNo2  NUMBER(10) NOT NULL,
Fax VARCHAR(20) ,
CONSTRAINT  student_address_id_pk PRIMARY KEY (StudentAddressID));
```

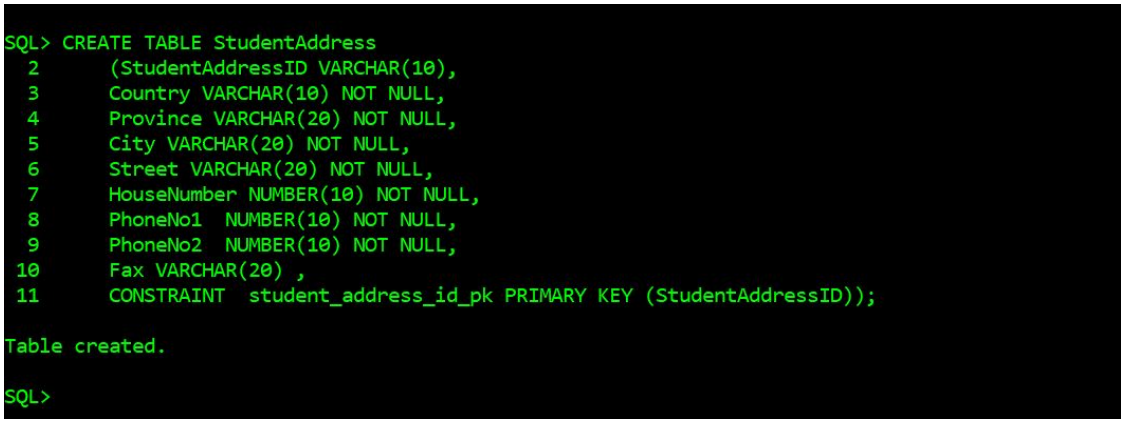

Figure 28 Create table StudentAddress

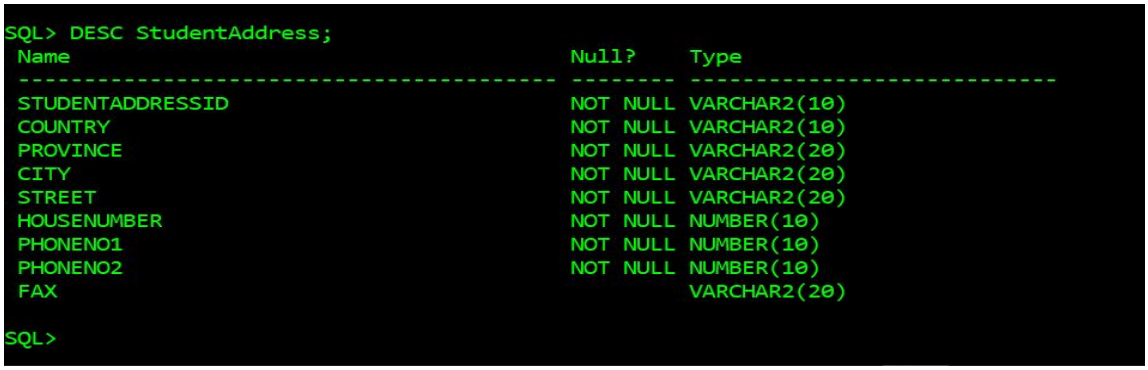

Figure 29 Describe StudentAddress

13. Student Address Type

```
CREATE TABLE StudentAddressType
(StudentAddressID VARCHAR(10),
   StudentID VARCHAR(10),
   AddressType VARCHAR(15),
   CONSTRAINT std_add_pk PRIMARY KEY(StudentAddressID,StudentID),
   CONSTRAINT std_add_fk FOREIGN KEY (StudentAddressID) REFERENCES
StudentAddress(StudentAddressID),
   CONSTRAINT Std_id_fk FOREIGN KEY(StudentID) REFERENCES Student(StudentID));
```

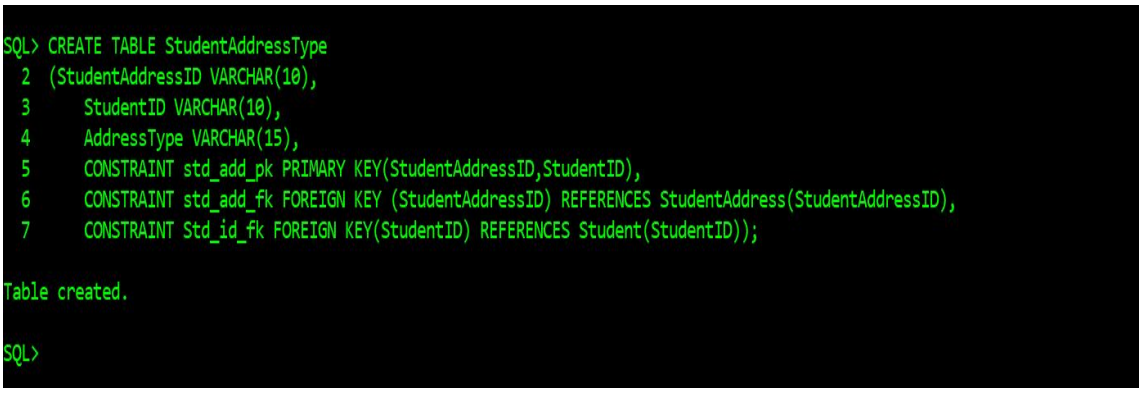

Figure 30 Create table StudentAddressType

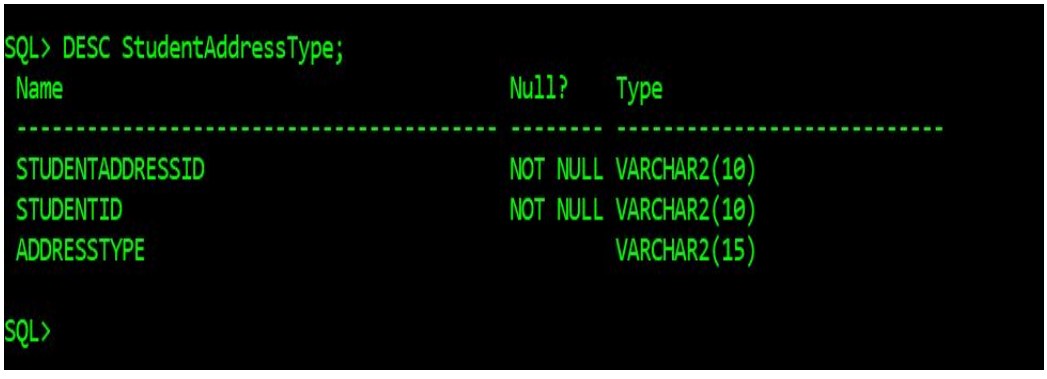

Figure 31 Describe StudentAddressType

Populating Data into Tables

Insertion and Selection

SQL was used to create tables which are now populated using SQL as well. Simple command which goes by INSERT helps to insert values into the table. Populated Tables were later made permanent using a command called COMMIT. A populated table is then displayed by using SELECT * FROM TABLE query.

1. **Course**

```
INSERT ALL
INTO Course(CourseID, CourseName) VALUES ('I1','BIT')
INTO Course(CourseID, CourseName) VALUES ('B1','Business')
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2       INTO Course(CourseID, CourseName) VALUES ('I1','BIT')
  3       INTO Course(CourseID, CourseName) VALUES ('B1','Business')
  4       SELECT * FROM dual;

2 rows created.

SQL>
```

Figure 32 Insert into Courses

```
SQL> SELECT * FROM Course;

COURSEID   COURSENAME
---------- ----------
I1         BIT
B1         Business

SQL>
```

Figure 33 Selection from Course

2. Specification

```
INSERT ALL
INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES
('C1','Computing',1300000)
INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES
('N1','Networking',1000000)
INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES
('M1','Multimedia',1200000)
INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES
('B1','MBA',700000)
INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES
('M2','Marketing',700000)
INTO Specification(SpecificationID, SpecificationName, SpecificationFees)  VALUES
('F1','Finance',800000)
INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES
('IB1','IB',1500000)
 SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES ('C1','Computing',1300000)
  3  INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES ('N1','Networking',1000000)
  4  INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES ('M1','Multimedia',1200000)
  5  INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES ('B1','MBA',700000)
  6  INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES ('M2','Marketing',700000)
  7  INTO Specification(SpecificationID, SpecificationName, SpecificationFees)  VALUES ('F1','Finance',800000)
  8  INTO Specification(SpecificationID, SpecificationName, SpecificationFees) VALUES ('IB1','IB',1500000)
  9   SELECT * FROM dual;

7 rows created.
```

Figure 34 Insert into Specification

```
SQL> SELECT * FROM Specification;

SPECIFICAT SPECIFICATIONNAME     SPECIFICAT
---------- --------------------- ----------
C1         Computing             1300000
N1         Networking            1000000
M1         Multimedia            1200000
B1         MBA                   700000
M2         Marketing             700000
F1         Finance               800000
IB1        IB                    1500000

7 rows selected.

SQL>
```

Figure 35 Selection from Specification

3. **Student**

```
INSERT ALL
INTO Student(StudentID,
Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('SA1','Srijan
Adhikari', 'M','27-NOV-01', ' 21-APR-20', 'srijan.adhikari@gmail.com', 'I1',499)
INTO Student(StudentID,
Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES
('GG1','Grisha Giri', 'F','25-MAY-01', '20-APR-20 ', 'grishagiri@gmail.com', 'B1',456)
INTO Student(StudentID,
Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('RS1','Rajin
Shrestha', 'M','23-JUN-02', '19-APR-20 ','shrestha.rajin@gmail.com', 'B1',300)
INTO Student(StudentID,
Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES
('HK1','Helina Khanal', 'F','05-MAY-01', '18-APR-20 ', 'heli.khanal@gmail.com', 'B1', 480)
INTO Student(StudentID,
Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('PS1','Prajna
Subedi', 'F','06-APR-03', '16-MAY-19 ','prajnasubedi@gmail.com', 'I1',460)
INTO Student(StudentID,
Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('RS2','Rojan
Shrestha', 'M','12-DEC-99', '18-NOV-18 ' ,'rojanshrestha@gmail.com', 'I1',469)
INTO Student(StudentID,
Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES
('PS2','Priyanka Shrestha', 'F','02-FEB-99', '17-DEC-17 ', 'srpriyanka@gmail.com', 'I1', 459)
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Student(StudentID, Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('SA1','Srijan Adhikari', 'M','27-NOV-01', ' 21-APR-20', 'srijan.adhikari
@gmail.com', 'I1',499)
  3  INTO Student(StudentID, Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('GG1','Grisha Giri', 'F','25-MAY-01', '20-APR-20 ', 'grishagiri@gmail.co
m', 'B1',456)
  4  INTO Student(StudentID, Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('RS1','Rajin Shrestha', 'M','23-JUN-02', '19-APR-20 ','shrestha.rajin@gm
ail.com', 'B1',300)
  5  INTO Student(StudentID, Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('HK1','Helina Khanal', 'F','05-MAY-01', '18-APR-20 ', 'heli.khanal@gmail
.com', 'B1', 480)
  6  INTO Student(StudentID, Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('PS1','Prajna Subedi', 'F','06-APR-03', '16-MAY-19 ','prajnasubedi@gmail
.com', 'I1',460)
  7  INTO Student(StudentID, Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('RS2','Rojan Shrestha', 'M','12-DEC-99', '18-NOV-18 ' ,'rojanshrestha@gm
ail.com', 'I1',469)
  8  INTO Student(StudentID, Name,Sex,DOB,EnrolledDate,EmailAddress,CourseID,MarksObtained) VALUES ('PS2','Priyanka Shrestha', 'F','02-FEB-99', '17-DEC-17 ', 'srpriyanka@gm
ail.com', 'I1', 459)
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 36 Insert into Students

```
SQL> SELECT * FROM Student;

STUDENTID  NAME               SEX        DOB       ENROLLEDD EMAILADDRESS             COURSEID   MARKSOBTAINED
---------- ------------------ ---------- --------- --------- ------------------------ ---------- -------------
SA1        Srijan Adhikari    M          27-NOV-01 21-APR-20 srijan.adhikari@gmail.com I1                   499
GG1        Grisha Giri        F          25-MAY-01 20-APR-20 grishagiri@gmail.com     B1                   456
RS1        Rajin Shrestha     M          23-JUN-02 19-APR-20 shrestha.rajin@gmail.com B1                   300
HK1        Helina Khanal      F          05-MAY-01 18-APR-20 heli.khanal@gmail.com    B1                   480
PS1        Prajna Subedi      F          06-APR-03 16-MAY-19 prajnasubedi@gmail.com   I1                   460
RS2        Rojan Shrestha     M          12-DEC-99 18-NOV-18 rojanshrestha@gmail.com  I1                   469
PS2        Priyanka Shrestha  F          02-FEB-99 17-DEC-17 srpriyanka@gmail.com     I1                   459

7 rows selected.

SQL>
```

Figure 37 Selection from Students

4. **SpecificationInfo**

```
INSERT ALL
INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('SA1', 'N1', 'I1')
INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('GG1', 'C1', 'I1')
INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('RS1', 'C1', 'I1')
INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('HK1', 'N1', 'I1')
INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('PS1', 'F1', 'B1')
INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('RS2', 'M2', 'B1')
INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('PS2', 'IB1', 'B1')
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('SA1', 'N1', 'I1')
  3  INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('GG1', 'C1', 'I1')
  4  INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('RS1', 'C1', 'I1')
  5  INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('HK1', 'N1', 'I1')
  6  INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('PS1', 'F1', 'B1')
  7  INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('RS2', 'M2', 'B1')
  8  INTO SpecificationInfo(StudentID,SpecificationID,CourseID) VALUES ('PS2', 'IB1', 'B1')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 38 Insert into SpecificationInfo

```
SQL> SELECT * FROM SpecificationInfo;

SPECIFICAT STUDENTID
---------- ----------
N1         SA1
C1         GG1
C1         RS1
N1         HK1
F1         PS1
M2         RS2
IB1        PS2

7 rows selected.

SQL>
```

Figure 39 Selection from InstructorInfo

5. Instructor

```
INSERT ALL
INTO Instructor(InstructorID, Name,Salary,HireDate,EmailAddress,Role) VALUES
('BS6','Bhim Sunar',90000,'19-MAY-18', 'bhim.sunar@gmail.com', 'ModuleLeader')
INTO Instructor (InstructorID, Name,Salary,HireDate,EmailAddress,Role) VALUES
('HT1','HrishavTandukar',70000,'18-APR-17',  'rishav.tandukar@gmail.com', 'Lecturer')
INTO Instructor (InstructorID, Name,Salary,HireDate,EmailAddress,Role) VALUES
('DS1','Dipeshwor Silwal',10000,'04-JAN-19', 'dipeshworsilwal@gmail.com', 'Module Leader')
INTO Instructor (InstructorID, Name,Salary,HireDate,EmailAddress,Role) VALUES
('AT1','Adesh Tandukar',80000,'06-APR-19', 'adesh.tandukar@gmail.com', 'Tutor')
INTO Instructor (InstructorID, Name,Salary,HireDate,EmailAddress,Role) VALUES
('BA1','Biwash Adhikari',80000,'05-MAY-17', 'biwash.adhikari@gmail.com', 'Tutor')
INTO Instructor (InstructorID, Name,Salary,HireDate,EmailAddress,Role) VALUES
('SR1','SanjayaRegmi',111000,'10-FEB-15', 'regmisanjaya@gmail.com', 'Course Leader')
INTO Instructor (InstructorID, Name,Salary,HireDate,EmailAddress,Role) VALUES
('WM1','Weenit Maharjan',65000,'10-FEB-15', ' wmmaharjan@gmail.com', 'Lecturer')
SELECT * FROM dual;
```

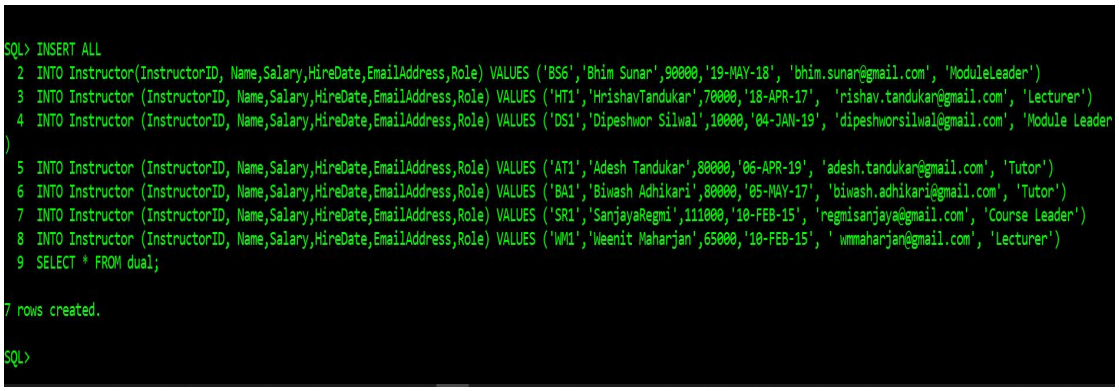

Figure 40 Insert into Instructor

```
SQL> SELECT * FROM Instructor;

INSTRUCTOR NAME                 SALARY HIREDATE  EMAILADDRESS             ROLE
---------- -------------------- ---------- --------- ------------------------ --------------------
BS6        Bhim Sunar            90000 19-MAY-18 bhim.sunar@gmail.com      ModuleLeader
HT1        HrishavTandukar       70000 18-APR-17 rishav.tandukar@gmail.com Lecturer
DS1        Dipeshwor Silwal      10000 04-JAN-19 dipeshworsilwal@gmail.com Module Leader
AT1        Adesh Tandukar        80000 06-APR-19 adesh.tandukar@gmail.com  Tutor
BA1        Biwash Adhikari       80000 05-MAY-17 biwash.adhikari@gmail.com Tutor
SR1        SanjayaRegmi         111000 10-FEB-15 regmisanjaya@gmail.com    Course Leader
WM1        Weenit Maharjan       65000 10-FEB-15  wmmaharjan@gmail.com     Lecturer

7 rows selected.

SQL>
```

Figure 41 Selection from Instructor

6. InstructorAddress

```
INSERT ALL
INTO
InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('KK1','Nepal','Bagmati', ' Kathmandu', ' Kamalmarg', 1000, 985627878, 9877626261, 65776)
INTO
InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('PK1','Nepal','Gandaki', ' Pokhara', ' Lekhnath', 1200, 986111999, '' ,64888)
INTO
InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('BT1','Nepal','Lumbini', ' Butwal', ' Tilottama', 999, 985155222,'',56992)
INTO
InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('BH1','Nepal','Karnali', ' Birendranagar', ' Hariharpur', 872, 985125255, 9988654450,77092)
INTO
InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('HN1','Nepal','Bagmati', ' Hetauda', ' Nibuatar', 10009, 9875252773, 9000888099, 12001)
INTO
InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('KP1','Nepal','Bagmati', ' Kathmandu', ' Putalisadak', 9090, 986525522, 9877778778,22900)
INTO
InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('BT2','Nepal','Province no.1', ' Biratnagar', ' BusPark', 2828, 9842820000,'',22800)
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('KK1','Nepal','Bagmati', ' Kathmandu', ' Kamalmarg',
 1000, 985627878, 9877626261, 65776)
  3  INTO InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('PK1','Nepal','Gandaki', ' Pokhara', ' Lekhnath', 12
00, 986111999, '' ,64888)
  4  INTO InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('BT1','Nepal','Lumbini', ' Butwal', ' Tilottama', 99
9, 985155222,'',56992)
  5  INTO InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('BH1','Nepal','Karnali', ' Birendranagar', ' Harihar
pur', 872, 985125255, 9988654450,77092)
  6  INTO InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('HN1','Nepal','Bagmati', ' Hetauda', ' Nibuatar', 10
009, 9875252773, 9000888099, 12001)
  7  INTO InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('KP1','Nepal','Bagmati', ' Kathmandu', ' Putalisadak
', 9090, 986525522, 9877778778,22900)
  8  INTO InstructorAddress(InstructorAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('BT2','Nepal','Province no.1', ' Biratnagar', ' BusP
ark', 2828, 9842820000,'',22800)
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 42 Insert into InstructorAddress

```
SQL> SELECT * FROM InstructorAddress;

INSTRUCTOR COUNTRY              PROVINCE             CITY                 STREET               HOUSENUMBER   PHONENO1   PHONENO2 FAX
---------- -------------------- -------------------- -------------------- -------------------- ----------- ---------- ---------- --------------------
KK1        Nepal                Bagmati              Kathmandu            Kamalmarg                   1000  985627878 9877626261 65776
PK1        Nepal                Gandaki              Pokhara              Lekhnath                    1200  986111999            64888
BT1        Nepal                Lumbini              Butwal               Tilottama                    999  985155222            56992
BH1        Nepal                Karnali              Birendranagar        Hariharpur                   872  985125255 9988654450 77092
HN1        Nepal                Bagmati              Hetauda              Nibuatar                   10009 9875252773 9000888099 12001
KP1        Nepal                Bagmati              Kathmandu            Putalisadak                 9090  986525522 9877778778 22900
BT2        Nepal                Province no.1        Biratnagar           BusPark                     2828 9842820000            22800

7 rows selected.

SQL>
```

Figure 43 Selection from InstructorAddress

7. **InstructorAddress Type**

INSERT ALL

INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('KK1','BS6', 'Temporary')

INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('PK1','HT1', 'Permanent')

INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('BT1','DS1', 'Permanent')

INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('BH1','BA1', ' Permanent ')

INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('HN1','AT1', ' Permanent ')

INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('KP1','SR1', 'Temporary')

INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('BT2','BS6', ' Permanent ')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('KK1','BS6', 'Temporary')
  3  INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('PK1','HT1', 'Permanent')
  4  INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('BT1','DS1', 'Permanent')
  5  INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('BH1','BA1', ' Permanent ')
  6  INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('HN1','AT1', ' Permanent ')
  7  INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('KP1','SR1', 'Temporary')
  8  INTO InstructorAddressType(InstructorAddressID, InstructorID,AddressType) VALUES ('BT2','BS6', ' Permanent ')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 44 Insert into InstructorAddressType

```
SQL> SELECT * FROM InstructorAddressType;

INSTRUCTOR INSTRUCTOR ADDRESSTYPE
---------- ---------- ---------------
KK1        BS6        Temporary
PK1        HT1        Permanent
BT1        DS1        Permanent
BH1        BA1         Permanent
HN1        AT1         Permanent
KP1        SR1        Temporary
BT2        BS6         Permanent

7 rows selected.

SQL>
```

Figure 45 Selection from InstructorAddressType

8. Class

```
INSERT ALL
INTO Class(ClassID, ClassName) VALUES ('K1','Kantipur')
INTO Class(ClassID, ClassName) VALUES ('C1','Cambridge')
INTO Class(ClassID, ClassName) VALUES ('Y1','York')
INTO Class(ClassID, ClassName) VALUES ('P1','Pokhara')
INTO Class(ClassID, ClassName) VALUES ('P2','Patan')
INTO Class(ClassID, ClassName) VALUES ('O1','Oxford')
INTO Class(ClassID, ClassName) VALUES ('N1','Nottingham')
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Class(ClassID, ClassName) VALUES ('K1','Kantipur')
  3  INTO Class(ClassID, ClassName) VALUES ('C1','Cambridge')
  4  INTO Class(ClassID, ClassName) VALUES ('Y1','York')
  5  INTO Class(ClassID, ClassName) VALUES ('P1','Pokhara')
  6  INTO Class(ClassID, ClassName) VALUES ('P2','Patan')
  7  INTO Class(ClassID, ClassName) VALUES ('O1','Oxford')
  8  INTO Class(ClassID, ClassName) VALUES ('N1','Nottingham')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 46 Insert into Class

```
SQL> SELECT * FROM Class;

CLASSID    CLASSNAME
---------- --------------------
K1         Kantipur
C1         Cambridge
Y1         York
P1         Pokhara
P2         Patan
O1         Oxford
N1         Nottingham

7 rows selected.

SQL>
```

Figure 47 Selection from Class

9. Module

```
INSERT ALL
INTO Module(ModuleID, ModuleName,ClassID) VALUES ('DB1','Database','K1')
INTO Module(ModuleID, ModuleName,ClassID) VALUES ('EP1','Emerging
Programming','P1')
INTO Module(ModuleID, ModuleName,ClassID) VALUES ('SE1','Software Engineering','P2')
INTO Module(ModuleID, ModuleName,ClassID) VALUES ('NOS1',' NOS','O1')
INTO Module(ModuleID, ModuleName,ClassID) VALUES ('LPS1',' Logic and Problems','N1')
INTO Module(ModuleID, ModuleName,ClassID) VALUES ('LIN1',' Linux','Y1')
INTO Module(ModuleID, ModuleName,ClassID) VALUES ('SHA1',' Software','Y1')
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Module(ModuleID, ModuleName,ClassID) VALUES ('DB1','Database','K1')
  3  INTO Module(ModuleID, ModuleName,ClassID) VALUES ('EP1','Emerging Programming','P1')
  4  INTO Module(ModuleID, ModuleName,ClassID) VALUES ('SE1','Software Engineering','P2')
  5  INTO Module(ModuleID, ModuleName,ClassID) VALUES ('NOS1',' NOS','O1')
  6  INTO Module(ModuleID, ModuleName,ClassID) VALUES ('LPS1',' Logic and Problems','N1')
  7  INTO Module(ModuleID, ModuleName,ClassID) VALUES ('LIN1',' Linux','Y1')
  8  INTO Module(ModuleID, ModuleName,ClassID) VALUES ('SHA1',' Software','Y1')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 48 Insertion into Module

```
SQL> SELECT * FROM Module;

MODULEID   MODULENAME           CLASSID
---------- -------------------- ----------
DB1        Database             K1
EP1        Emerging Programming P1
SE1        Software Engineering P2
NOS1        NOS                 O1
LPS1        Logic and Problems  N1
LIN1        Linux               Y1
SHA1        Software            Y1

7 rows selected.

SQL>
```

Figure 49 Selection from Module

10. InstructorInfo

```
INSERT ALL
INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('SHA1','BS6', 'C1')
INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('LPS1','BS6', 'C1')
INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('DB1','DS1', 'F1')
INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('EP1','BA1', 'IB1')
INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('LPS1','AT1', 'N1')
INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('LIN1','SR1', 'M1')
INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('NOS1','WM1', 'B1')
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('SHA1','BS6', 'C1')
  3  INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('LPS1','BS6', 'C1')
  4  INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('DB1','DS1', 'F1')
  5  INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('EP1','BA1', 'IB1')
  6  INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('LPS1','AT1', 'N1')
  7  INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('LIN1','SR1', 'M1')
  8  INTO InstructorInfo(ModuleID,InstructorID,SpecificationID) VALUES ('NOS1','WM1', 'B1')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 50 Insert into InstructorInfo

```
SQL> SELECT * FROM InstructorInfo;

INSTRUCTOR MODULEID   SPECIFICAT
---------- ---------- ----------
BS6        SHA1       C1
BS6        LPS1       C1
DS1        DB1        F1
BA1        EP1        IB1
AT1        LPS1       N1
SR1        LIN1       M1
WM1        NOS1       B1

7 rows selected.

SQL>
```

Figure 51 Selection from InstructorInfo

11. ModuleInfo

```
INSERT ALL
INTO ModuleInfo(ModuleID, StudentID,SpecificationID) VALUES ('NOS1','SA1', 'N1')
INTO ModuleInfo (ModuleID, StudentID,SpecificationID)  VALUES ('LPS1','GG1', 'C1')
INTO ModuleInfo (ModuleID, StudentID,SpecificationID) VALUES ('DB1','RS1', 'C1')
INTO ModuleInfo (ModuleID, StudentID,SpecificationID) VALUES ('EP1','HK1', 'N1')
INTO ModuleInfo (ModuleID, StudentID,SpecificationID) VALUES ('LPS1','PS1', 'F1')
INTO ModuleInfo (ModuleID, StudentID,SpecificationID) VALUES ('LIN1','RS2', 'M2')
INTO ModuleInfo (ModuleID, StudentID,SpecificationID) VALUES ('SHA1','PS2', 'IB1')
SELECT * FROM dual;
```

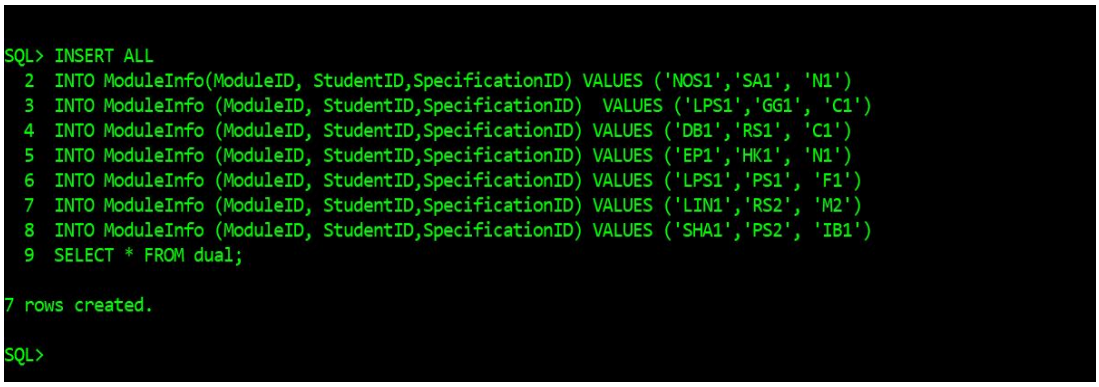

Figure 52 Insert into ModuleInfo

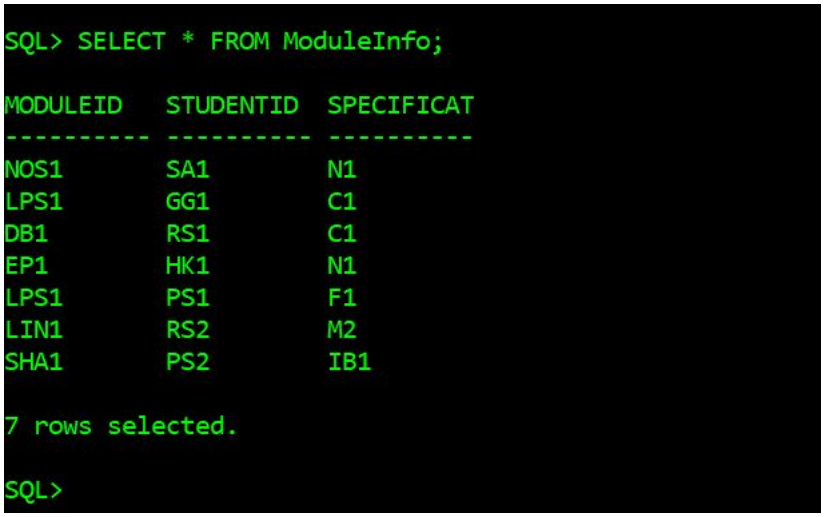

Figure 53 Selection from ModuleInfo

12. StudentAddress

```
INSERT ALL
INTO
StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,Phon
eNo2,Fax) VALUES ('HN1','Nepal','Bagmati', ' Hetauda', ' Namtar', 205, 982929228, " , 898882)
INTO
StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,Phon
eNo2,Fax) VALUES ('PL1','Nepal','Gandaki', ' Pokhara', ' Lekhnath', 2002, 982777272,
97332233,929992)
INTO
StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,Phon
eNo2,Fax) VALUES ('BJ1','Nepal','Karnali', ' Birendranagar', ' Jarbuta', 19199, 988981823,
98833203, 879882)
INTO
StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,Phon
eNo2,Fax) VALUES ('KK1','Nepal','Bagmati', ' Kathmandu', ' Kamalpokhari', 98888,
987622212, 09882211, 982222)
INTO
StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,Phon
eNo2,Fax) VALUES ('KP1','Nepal','Bagmati', 'Kathmandu ' , 'Putalisadak', 909009, 989997122,
09922111, 328892)
INTO
StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,Phon
eNo2,Fax) VALUES ('RD1','Nepal','Lumbini', ' Rupandehi', ' Devdaha', 2002, 9982882291,
96645342, 909992)
INTO
StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,Phon
eNo2,Fax) VALUES ('KD1','Nepal','Bagmati', ' Kathmandu', ' Dillibazar', 565772, 982451272,
",122242)
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('HN1','Nepal','Bagmati', ' Hetauda', ' Namtar', 205, 98292
9228, '' , 898882)
  3  INTO StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('PL1','Nepal','Gandaki', ' Pokhara', ' Lekhnath', 2002, 98
2777272, 97332233,929992)
  4  INTO StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('BJ1','Nepal','Karnali', ' Birendranagar', ' Jarbuta', 191
99, 988981823, 98833203, 879882)
  5  INTO StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('KK1','Nepal','Bagmati', ' Kathmandu', ' Kamalpokhari', 98
888, 987622212, 09882211, 982222)
  6  INTO StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('KP1','Nepal','Bagmati', 'Kathmandu ' , 'Putalisadak', 909
009, 989997122, 09922111, 328892)
  7  INTO  StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('RD1','Nepal','Lumbini', ' Rupandehi', ' Devdaha', 2002,
9982882291, 96645342, 909992)
  8  INTO StudentAddress(StudentAddressID,Country,Province,City,Street,HouseNumber,PhoneNo1,PhoneNo2,Fax) VALUES ('KD1','Nepal','Bagmati', ' Kathmandu', ' Dillibazar', 5657
72, 982451272, '',122242)
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 54 Insert into StudentAddress

```
SQL> SELECT * FROM StudentAddress;

STUDENTADD COUNTRY    PROVINCE            CITY                STREET              HOUSENUMBER   PHONENO1   PHONENO2 FAX
---------- ---------- ------------------- ------------------- ------------------- ----------- ---------- ---------- --------------------
HN1        Nepal      Bagmati             Hetauda             Namtar                      205  982929228             898882
PL1        Nepal      Gandaki             Pokhara             Lekhnath                   2002  982777272   97332233 929992
BJ1        Nepal      Karnali             Birendranagar       Jarbuta                   19199  988981823   98833203 879882
KK1        Nepal      Bagmati             Kathmandu           Kamalpokhari              98888  987622212    9882211 982222
KP1        Nepal      Bagmati             Kathmandu           Putalisadak              909009  989997122    9922111 328892
RD1        Nepal      Lumbini             Rupandehi           Devdaha                    2002 9982882291   96645342 909992
KD1        Nepal      Bagmati             Kathmandu           Dillibazar               565772  982451272             122242

7 rows selected.

SQL>
```

Figure 55 Selection from StudentAddress

13. StudentAddressType

```
INSERT ALL
INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES
('HN1','SA1', 'Permanent')
INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES
('PL1','GG1', 'Permanent')
INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('BJ1','RS1',
'Permanent')
INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES
('KK1','HK1', 'Temporary')
INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('KP1','PS1',
'Temporary')
INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES
('RD1','RS2', 'Temporary')
INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES
('KD1','PS2', 'Temporary')
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('HN1','SA1', 'Permanent')
  3  INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('PL1','GG1', 'Permanent')
  4  INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('BJ1','RS1', 'Permanent')
  5  INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('KK1','HK1', 'Temporary')
  6  INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('KP1','PS1', 'Temporary')
  7  INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('RD1','RS2', 'Temporary')
  8  INTO StudentAddressType (StudentAddressID, StudentID,AddressType) VALUES ('KD1','PS2', 'Temporary')
  9  SELECT * FROM dual;

7 rows created.

SQL>
```

Figure 56 Insert into StudentAddressType

```
SQL> SELECT * FROM StudentAddressType;

STUDENTADD STUDENTID  ADDRESSTYPE
---------- ---------- ---------------
HN1        SA1        Permanent
PL1        GG1        Permanent
BJ1        RS1        Permanent
KK1        HK1        Temporary
KP1        PS1        Temporary
RD1        RS2        Temporary
KD1        PS2        Temporary

7 rows selected.

SQL>
```

Figure 57 Selection fron StudentAddressType

These Populated tables are then set as permanent value by using a COMMIT query.

```
SQL> commit;

Commit complete.

SQL>
```

Figure 58 Commit

Information and Transaction Queries

Information Queries:

1. **List all the students with all their addresses with their phone numbers.**

```
SELECT  s.Name ,
ad.AddressType,sa.Country,sa.Province,sa.City,sa.Street,sa.HouseNumber,sa.PhoneNo1,sa.Phon
eNo2,sa.fax
FROM Student s
JOIN StudentAddressType ad
ON s.StudentID = ad.StudentID
JOIN StudentAddress sa
ON sa.StudentAddressID = ad.StudentAddressID;
```

```
SQL> SELECT  s.Name , ad.AddressType,sa.Country,sa.Province,sa.City,sa.Street,sa.HouseNumber,sa.PhoneNo1,sa.PhoneNo2,sa.fax
  2  FROM Student s
  3  JOIN StudentAddressType ad
  4  ON s.StudentID = ad.StudentID
  5  JOIN StudentAddress sa
  6  ON sa.StudentAddressID = ad.StudentAddressID;

NAME              ADDRESSTYPE     COUNTRY    PROVINCE           CITY               STREET             HOUSENUMBER   PHONENO1   PHONENO2 FAX
----------------- --------------- ---------- ------------------ ------------------ ------------------ ----------- ---------- ---------- --------------------
Srijan Adhikari   Permanent       Nepal      Bagmati             Hetauda            Namtar                     205  982929228           898882
Grisha Giri       Permanent       Nepal      Gandaki             Pokhara            Lekhnath                  2002  982777272   97332233 929992
Rajin Shrestha    Permanent       Nepal      Karnali             Birendranagar      Jarbuta                  19199  988981823   98833203 879882
Helina Khanal     Temporary       Nepal      Bagmati             Kathmandu          Kamalpokhari             98888  987622212    9882211 982222
Prajna Subedi     Temporary       Nepal      Bagmati            Kathmandu          Putalisadak              909009  989997122    9922111 328892
Rojan Shrestha    Temporary       Nepal      Lumbini             Rupandehi          Devdaha                   2002 9982882291   96645342 909992
Priyanka Shrestha Temporary       Nepal      Bagmati             Kathmandu          Dillibazar              565772  98451272           122242

7 rows selected.

SQL>
```

Figure 59 Information Query 1

2. List all the modules which are taught by more than one instructor.

```
Select ii.ModuleID, m.ModuleName,Count(ii.InstructorID) "Number of Instructors"
   FROM InstructorInfo ii
   JOIN Module m
    ON m.ModuleID = ii.ModuleID
    GROUP BY ii.ModuleID,m.ModuleName
    HAVING COUNT(ii.ModuleID)>1;
```

```
SQL> Select ii.ModuleID, m.ModuleName,Count(ii.InstructorID) "Number of Instructors"
  2  FROM InstructorInfo ii
  3  JOIN Module m
  4  ON m.ModuleID = ii.ModuleID
  5  GROUP BY ii.ModuleID,m.ModuleName
  6  HAVING COUNT(ii.ModuleID)>1;

MODULEID   MODULENAME          Number of Instructors
---------- ------------------- --------------------
LPS1        Logic and Problems                    2

SQL>
```

Figure 60 Information Query 2

3. **List the name of all the instructors whose name contains ‘s’ and salary is above 50,000.**

```
SELECT Name,Salary
FROM Instructor
WHERE LOWER(Name) LIKE '%s%'
AND Salary> 50000;
```

```
SQL> SELECT Name,Salary
  2  FROM Instructor
  3  WHERE LOWER(Name) LIKE '%s%'
  4  AND Salary> 50000;

NAME                    SALARY
-------------------- ----------
Bhim Sunar               90000
HrishavTandukar          70000
Adesh Tandukar           80000
Biwash Adhikari          80000
SanjayaRegmi            111000

SQL>
```

Figure 61 Information Query 3

4. **List the modules comes under the 'Multimedia' specification.**

```
SELECT m.ModuleName
FROM Module m JOIN ModuleInfo mi
ON m.ModuleID = mi.ModuleID JOIN Specification sp
 ON mi.SpecificationID = sp.SpecificationID
 WHERE SpecificationName = 'Multimedia';
```

```
SQL> UPDATE ModuleInfo
  2  SET SpecificationID = 'M1'
  3  WHERE StudentID = 'RS2';

1 row updated.

SQL> SELECT * FROM ModuleInfo;

MODULEID   STUDENTID  SPECIFICAT
---------- ---------- ----------
NOS1       SA1        N1
LPS1       GG1        C1
DB1        RS1        C1
EP1        HK1        N1
LPS1       PS1        F1
LIN1       RS2        M1
SHA1       PS2        IB1

7 rows selected.

SQL>
```

Figure 62 Updation required for IQ4

```
SQL> SELECT m.ModuleName
  2   FROM Module m JOIN ModuleInfo mi
  3  ON m.ModuleID = mi.ModuleID JOIN Specification sp
  4  ON mi.SpecificationID = sp.SpecificationID
  5  WHERE SpecificationName = 'Multimedia';

MODULENAME
--------------------
 Linux

SQL>
```

Figure 63 Information Query 4

5. List the name of the head of modules with the list of his phone number.

```
SELECT i.Name,i.Role,ia.PhoneNo1,ia.PhoneNo2, iat.AddressType
FROM Instructor i JOIN InstructorAddressType iat
ON i.InstructorID = iat.InstructorID
JOIN InstructorAddress ia
ON ia.InstructorAddressID = iat.InstructorAddressID
WHERE Role= 'ModuleLeader';
```

```
SQL> UPDATE Instructor
  2  SET Role = 'ModuleLeader'
  3  WHERE InstructorID= 'DS1';

1 row updated.
```

Figure 64 Updation required on IQ5

```
SQL> SELECT i.Name,i.Role,ia.PhoneNo1,ia.PhoneNo2, iat.AddressType
  2  FROM Instructor i JOIN InstructorAddressType iat
  3  ON i.InstructorID = iat.InstructorID
  4  JOIN InstructorAddress ia
  5  ON ia.InstructorAddressID = iat.InstructorAddressID
  6  WHERE Role= 'ModuleLeader';

NAME                ROLE                  PHONENO1   PHONENO2 ADDRESSTYPE
------------------- -------------------- ---------- ---------- ---------------
Bhim Sunar          ModuleLeader          985627878 9877626261 Temporary
Dipeshwor Silwal    ModuleLeader          985155222            Permanent
Bhim Sunar          ModuleLeader         9842820000             Permanent

SQL>
```

Figure 65 Information Query 5

6. List all Students who have enrolled in ‘networking’ specifications.

```
SELECT s.Name,sp.SpecificationName
 FROM Student s JOIN SpecificationInfo si
ON s.StudentID = si.StudentID
JOIN Specification sp
ON sp.SpecificationID = si.SpecificationID
WHERE sp.SpecificationName= 'Networking';
```

```
SQL> SELECT s.Name,sp.SpecificationName
  2  FROM Student s JOIN SpecificationInfo si
  3  ON s.StudentID = si.StudentID
  4  JOIN Specification sp
  5  ON sp.SpecificationID = si.SpecificationID
  6  WHERE sp.SpecificationName= 'Networking';

NAME                 SPECIFICATIONNAME
-------------------- --------------------
Srijan Adhikari      Networking
Helina Khanal        Networking

SQL>
```

Figure 66 Information Query 6

7. **List the fax number of the instructor who teaches the 'database' module.**

```
SELECT i.Name,ia.Fax,m.ModuleName
FROM Instructor i JOIN InstructorAddressType iat
ON i.InstructorID = iat.InstructorID
JOIN InstructorAddress ia
ON ia.InstructorAddressID = iat.InstructorAddressID
JOIN InstructorInfo ii
ON ii.InstructorID = i.InstructorID
JOIN Module m
ON m.ModuleID=ii.ModuleID
WHERE m.ModuleName='Database';
```

```
SQL> SELECT i.Name,ia.Fax,m.ModuleName
  2  FROM Instructor i JOIN InstructorAddressType iat
  3  ON i.InstructorID = iat.InstructorID
  4  JOIN InstructorAddress ia
  5  ON ia.InstructorAddressID = iat.InstructorAddressID
  6  JOIN InstructorInfo ii
  7  ON ii.InstructorID = i.InstructorID
  8  JOIN Module m
  9  ON m.ModuleID=ii.ModuleID
 10  WHERE m.ModuleName='Database';

NAME                 FAX                  MODULENAME
-------------------- -------------------- --------------------
Dipeshwor Silwal     56992                Database

SQL>
```

Figure 67 Information Query 7

8. List the specification falls under the BIT course.

```
SELECT sp.SpecificationName,c.CourseName
FROM Specification sp JOIN SpecificationInfo si
ON sp.SpecificationID = si.SpecificationID
JOIN Course c
ON c.CourseID = si.CourseID
WHERE CourseName='BIT';
```

```
SQL> SELECT sp.SpecificationName,c.CourseName
  2  FROM Specification sp JOIN SpecificationInfo si
  3  ON sp.SpecificationID = si.SpecificationID
  4  JOIN Course c
  5  ON c.CourseID = si.CourseID
  6  WHERE CourseName='BIT';

SPECIFICATIONNAME    COURSENAME
-------------------- ----------
Networking           BIT
Computing            BIT
Computing            BIT
Networking           BIT

SQL>
```

Figure 68 Information Query 8

9. List all the modules taught in any one particular class.

```
SELECT DISTINCT(m.ModuleName),ClassID
FROM Module m
GROUP BY m.ModuleName,ClassID
HAVING COUNT (m.ClassID)=1;
```

```
SQL> SELECT DISTINCT(m.ModuleName),ClassID
  2  FROM Module m
  3  GROUP BY m.ModuleName,ClassID
  4  HAVING COUNT (m.ClassID)=1;

MODULENAME           CLASSID
-------------------- ----------
Emerging Programming P1
Database             K1
Software Engineering P2
 Linux               Y1
 NOS                 O1
 Logic and Problems  N1
 Software            Y1

7 rows selected.

SQL>
```

Figure 69 Information Query 9

10. List all the teachers with all their addresses who have ‘a’ at the end of their first names.

```
SELECT i.Name,iad.InstructorAddressID
FROM Instructor i
JOIN InstructorAddressType iadt
ON i.InstructorID = iadt.InstructorID
JOIN InstructorAddress iad
ON iadt.InstructorAddressID = iad.InstructorAddressID
WHERE LOWER(SUBSTR(i.Name,1,INSTR(i.Name, ' ') -1)) LIKE '%a';
```

```
SQL> SELECT i.Name,iad.InstructorAddressID
  2  FROM Instructor i
  3  JOIN InstructorAddressType iadt
  4  ON i.InstructorID = iadt.InstructorID
  5  JOIN InstructorAddress iad
  6  ON iadt.InstructorAddressID = iad.InstructorAddressID
  7  WHERE LOWER(SUBSTR(i.Name,1,INSTR(i.Name, ' ') -1)) LIKE '%a';

NAME                 INSTRUCTOR
-------------------- ----------
Sanjaya Regmi        KP1

SQL>
```

Figure 70 Information Query 10

Transaction Queries:

1. **Show the students, course they enroll in and their fees. Reduce 10% of the fees if they are enrolled in a computing course.**

```
SELECT s.Name, c.CourseName, sp.SpecificationFees,
DECODE(sp.SpecificationName, 'Computing', 0.9*sp.SpecificationFees,
sp.SpecificationFees) RevisedFees
FROM Student s
JOIN SpecificationInfo si
ON s.StudentID = si.StudentID
JOIN Specification sp ON sp.SpecificationID = si.SpecificationID
JOIN Course c
ON c.CourseID = si.CourseID;
```

```
SQL> SELECT s.Name, c.CourseName, sp.SpecificationFees,
  2   DECODE(sp.SpecificationName, 'Computing', 0.9*sp.SpecificationFees, sp.SpecificationFees) RevisedFees
  3  FROM Student s
  4  JOIN SpecificationInfo si
  5   ON s.StudentID = si.StudentID
  6   JOIN Specification sp ON sp.SpecificationID = si.SpecificationID
  7  JOIN Course c
  8  ON c.CourseID = si.CourseID;

NAME                COURSENAME SPECIFICAT REVISEDFEES
------------------- ---------- ---------- -----------
Rajin Shrestha      BIT        1300000        1170000
Grisha Giri         BIT        1300000        1170000
Helina Khanal       BIT        1000000        1000000
Srijan Adhikari     BIT        1000000        1000000
Rojan Shrestha      Business   700000          700000
Prajna Subedi       Business   800000          800000
Priyanka Shrestha   Business   1500000        1500000

7 rows selected.
```

Figure 71 Transaction Query 1

2. **Place the default Number 1234567890 if the list of phone numbers to the location of the address is empty and give the column name as 'Contact details.**

```
SELECT PhoneNo2,
NVL(PhoneNo2,1234567890) AS "ContactDetails"
 FROM InstructorAddress;
```

```
SQL> SELECT PhoneNo2,
  2  NVL(PhoneNo2,1234567890) AS "ContactDetails"
  3  FROM InstructorAddress;

  PHONENO2 ContactDetails
---------- --------------
9877626261     9877626261
               1234567890
               1234567890
9988654450     9988654450
9000888099     9000888099
9877778778     9877778778
               1234567890

7 rows selected.

SQL>
```

Figure 72 Transaction Query 2

3. **Show the name of all the students with the number of weeks since they have enrolled in the course.**

SELECT NAME,(SYSDATE-EnrolledDate)/7 AS "Weeks since Enrollment”
FROM Student;

```
SQL> SELECT NAME,(SYSDATE-EnrolledDate)/7 AS "Weeks since Enrollment"
  2  FROM Student;

NAME                Weeks since Enrollment
------------------- ----------------------
Srijan Adhikari                 34.4018767
Grisha Giri                     34.5447338
Rajin Shrestha                  34.6875909
Helina Khanal                   34.8304481
Prajna Subedi                   83.1161624
Rojan Shrestha                  108.687591
Priyanka Shrestha               156.687591

7 rows selected.

SQL>
```

Figure 73 Transaction Query 3

4. **Show the name of the instructors who got equal salary and work in the same specification.**

```
SELECT i1.Name, i1.Salary, x1.SpecificationName
 From ( SELECT * FROM ( SELECT i.InstructorID, i.Salary, s.SpecificationName
 FROM Instructor i JOIN InstructorInfo ii
ON ii.InstructorID = i.InstructorID JOIN Specification s
 ON s.SpecificationID = ii.SpecificationID
GROUP BY s.SpecificationName, i.Salary, i.InstructorID) y1
 WHERE ( SELECT COUNT(*) FROM ( SELECT i.InstructorID, i.Salary, s.SpecificationName
FROM Instructor i JOIN InstructorInfo ii
ON ii.InstructorID = i.InstructorID JOIN Specification s
 ON s.SpecificationID = ii.SpecificationID
GROUP BY s.SpecificationName, i.Salary, i.InstructorID) y2
WHERE y1.Salary = y2.Salary AND y1.SpecificationName =y2.SpecificationName) >1 ) x1
JOIN Instructor i1 on x1.InstructorID = i1.InstructorID ORDER BY i1.Salary,
x1.SpecificationName, i1.Name;
```

```
SQL> SELECT i1.Name, i1.Salary, x1.SpecificationName
  2   From ( SELECT * FROM ( SELECT i.InstructorID, i.Salary, s.SpecificationName
  3   FROM Instructor i JOIN InstructorInfo ii
  4  ON ii.InstructorID = i.InstructorID JOIN Specification s
  5   ON s.SpecificationID = ii.SpecificationID
  6  GROUP BY s.SpecificationName, i.Salary, i.InstructorID) y1
  7   WHERE ( SELECT COUNT(*) FROM ( SELECT i.InstructorID, i.Salary, s.SpecificationName
  8  FROM Instructor i JOIN InstructorInfo ii
  9  ON ii.InstructorID = i.InstructorID JOIN Specification s
 10   ON s.SpecificationID = ii.SpecificationID
 11  GROUP BY s.SpecificationName, i.Salary, i.InstructorID) y2
 12  WHERE y1.Salary = y2.Salary AND y1.SpecificationName =y2.SpecificationName) >1 ) x1
 13  JOIN Instructor i1 on x1.InstructorID = i1.InstructorID ORDER BY i1.Salary, x1.SpecificationName, i1.Name;

NAME                   SALARY SPECIFICATIONNAME
-------------------- ---------- --------------------
Adesh Tandukar          80000 IB
Biwash Adhikari         80000 IB

SQL>
```

Figure 74 Transaction Query 4

5. **List all the courses with the total number of students enrolled course name and the highest marks obtained.**

```
SELECT c.CourseName,COUNT(s.StudentID) AS "Total No. of
Students",MAX(s.MarksObtained) AS "Highest Mark"
FROM Student s
JOIN Course c
ON c.CourseID = s.CourseID
GROUP BY CourseName;
```

```
SQL> SELECT c.CourseName,COUNT(s.StudentID) AS "Total No. of Students",MAX(s.MarksObtained) AS "Highest Mark"
  2  FROM Student s
  3  JOIN Course c
  4  ON c.CourseID = s.CourseID
  5  GROUP BY CourseName;

COURSENAME Total No. of Students Highest Mark
---------- -------------------- ------------
BIT                           4          499
Business                      3          480

SQL>
```

Figure 75 Transaction Query 5

6. List all the instructors who are also a course leader.

```
SELECT NAME,Role
FROM Instructor
WHERE Role= 'Course Leader';
```

```
SQL> SELECT NAME,Role
  2  FROM Instructor
  3  WHERE Role= 'Course Leader';

NAME                 ROLE
-------------------- --------------------
Sanjaya Regmi        Course Leader

SQL>
```

Figure 76 Transaction Query 6

Creation of Dump File:

Data are dumped in SQL. SQL Dump performs logical backups, producing a set of SQL statements that can be executed to reproduce the original database object definitions and table data.(Oracle Corporation, 2020)

```
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\acer-i7>cd C:\Users\acer-i7\Desktop\Database\SQL File

C:\Users\acer-i7\Desktop\Database\SQL File>exp Islington/Islington file = courswork.dmp

Export: Release 11.2.0.2.0 - Production on Sat Dec 19 14:10:05 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user ISLINGTON
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user ISLINGTON
About to export ISLINGTON's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export ISLINGTON's tables via Conventional Path ...
. . exporting table                          CLASS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         COURSE          2 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                     INSTRUCTOR          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table              INSTRUCTORADDRESS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table          INSTRUCTORADDRESSTYPE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                INSTRUCTORINFO          7 rows exported
```

Figure 77 Creation of Dump File i

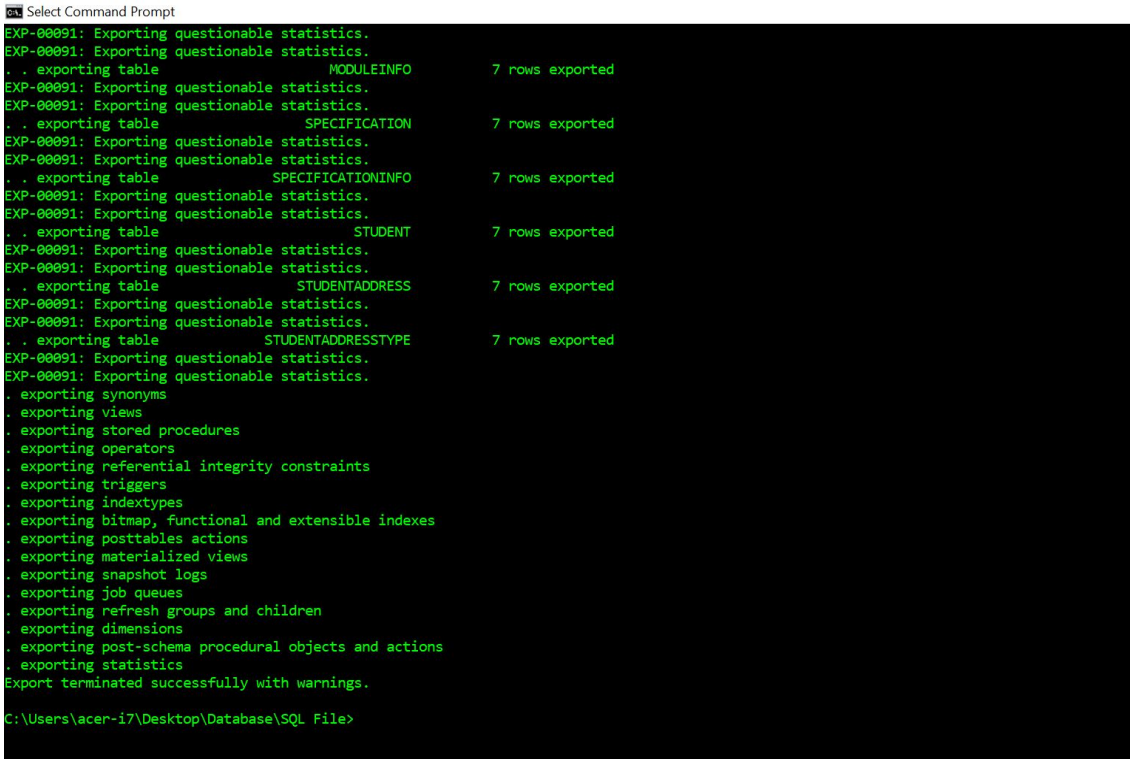

Figure 78 Creation of dump file ii

Drop Tables:

Tables with database are dropped in following order;

```
DROP TABLE SpecificationInfo;

DROP TABLE ModuleInfo;

DROP TABLE InstructorInfo;

DROP TABLE StudentAddressType;

DROP TABLE InstructorAddressType;

DROP TABLE Student;

DROP TABLE Course;

DROP TABLE Module;

DROP TABLE Class;

DROP TABLE Specification;

DROP TABLE Instructor;

DROP TABLE InstructorAddress;

DROP TABLE StudentAddress;
```

CONCLUSION:

The coursework was full of learning opportunity for me. To complete the research work, I had to briefly study about Entity Relationship Diagrams, Normalization and queries of SQL.

The report consists of a research works that was done on a private college named as Islington College. It briefly explains how it operates and other business rules. The report consists of a DBMS allowing entities and relations among them forming tables.

The database of a college was designed using Entity Relationship Diagram. Problems that were raised on a database from Entity Relationship were removed or reduced using Normalization. Problems were then solved resulting into final database. Those data were entered in SQL using different queries. Different queries in order to find out different database were entered in SQL which gives required database.

Overall, the report explains a Database Management system of a Private College . Learning opportunities on new topics like Normalization were difficult initially. To match with the given scenario for the coursework was another hassle. Normalization was a new thing for me which took me some time to adapt. Queries on mySQL were another problem. Very small mistakes led to error time and again.

With help of instructors, sites, and books it was very helpful for me to solve the doubts I've had.

The coursework shows the essentiality of Database management system in a real world. It shows how Normalization removes redundancy, makes data more efficient and consistent. The coursework shows the application of Normalization in a College Management System. Similarly it is also applicable to Bank Management System, Universities, Markets, Banks and different Management System.

With completion of the coursework, I've come to know deeply about concept of Database management system. I am known about normal forms and how Normalization is carried out step by step. Similarly, I've become known about how a management system works in a college, how it operates and how database is managed. I'm now familiar with mySQL that is used to communicate with database.

REFERENCES:

FreeDictionary . (2020). Retrieved from Farlex Inc.: https://encyclopedia2.thefreedictionary.com/
Geeks for Geeks. (2019, November 21). Retrieved from Geeks for Geeks website: https://www.geeksforgeeks.org/
Geeks for Geeks. (2019, November 25). Retrieved from Geeks for Geeks Web Site: https://www.geeksforgeeks.org/
Geeks for Geeks. (2019, September 25). Retrieved from Geeks for Geeks website: https://www.geeksforgeeks.org/
Geeks for Geeks. (2019, July 31). Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/
JavaTPoint. (2018). Retrieved from JavaTPoint Website: https://www.javatpoint.com/
Lucid Software Inc. (2020). Retrieved from Lucid Chart Website.
Oracle Corporation. (2020). Retrieved December 19, 2020, from Oracle Web site: https://dev.mysql.com/
Singh, C. (n.d.). *Beginners Book*. Retrieved 2020, from Beginners Book Web Site: http://www.beginnersbook.com
Tutorials Point . (2020). Retrieved from Tutorial Point Website.
w3schools. (2020). Retrieved from W3Schools Website: http:/www.w3schools.com
Weerasinghe, T. (2018, December 5). *Slide Share* . Retrieved from Slide Share Website: https://www.slideshare.net/TharinduWeerasinghe