islington college
(इस्लिङ्टन कलेज)

LONDON METROPOLITAN UNIVERSITY

Module Code & Module Title

CC5051NI Database

Assessment Weightage &Type

50% Individual Coursework

Year and Semester

2020 Spring

Student Name: Grisha Giri

Group: C17

London Met ID: 19033680

College ID: NP01CP4S200035

Assignment Due Date: 20th December, 2020

Assignment Submission Date: 20th 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.

ACKNOWLEDGEMENT

In the beginning of the coursework, I'd like to express my gratitude and sincerity to all the people who have helped me throughout the semester and the time I was completing my coursework. Without their guidance and suggestion, I'd have never reached the completion of my coursework.

Firstly, I'd like to express my gratitude towards my family who have been constantly supporting me throughout the time I've been doing my coursework. Without their support, I'd have never completed my coursework on time. They motivated me even when I was demotivated.

Secondly, I'm very thankful towards London Metropolitan University and Islington College to provide me with the opportunity to do this coursework. I want to express my gratitude towards the college and the university to provide me with the platform where I can study even during the global pandemic. In these difficult times of the lockdown, the college was successful in providing quality education.

I'd also like to thank our lecturers and tutors for constantly supporting us during these difficult times. I'd like to thank our module leader and lecturer Ms. Yunisha Bajracharya along with our tutors Mr. Aadesh Tandukar, Mr. Biwash Adhikari and Mr. Prashant Lal Shrestha for guiding me throughout the coursework. They have been guiding me and mentoring me throughout the semester. Hence, I'm very thankful to them.

Also, I wouldn't have been able to complete my coursework without my friends. That is why I am very grateful to my friends from providing me with the motivation and helping me throughout my problems that arose in my coursework. Their suggestions and motivation really helped me through the whole coursework doing period.

Thanking You,

Grisha Giri

C17

Table of Contents

1. Introduction 1

1.1. Introduction of the College 1

1.2. Current Business Activities and Operations 2

1.3. Business Rules 3

1.4. Creation of Entities and Attributes 4

1.5. Initial ERD 5

2. Normalization 7

2.1. UNF (Un-Normalized Normal Form) 7

2.2. 1NF (First Normal Form) 8

2.3. 2NF (Second Normal Form) 9

2.4. 3NF (Third Normal Form) 13

3. Final Entity Relationship Diagram 16

4. Database Implementation 18

4.1. Table Generation 19

4.2. Populating Data 35

5. Database Querying 73

5.1. Information Query 73

5.2. Transaction Query 86

6. Creation of Dump File 95

7. Drop Table 96

8. Conclusion 99

Bibliography 100

List of Figures

Figure 1: Initial ERD 5
Figure 2: Final ERD 16
Figure 3: Connect system 18
Figure 4: Connect User 18
Figure 5: Create Table Course 19
Figure 6: Create Table Specification 20
Figure 7: Create Table Specification Info 21
Figure 8: Create Table Class 21
Figure 9: Create Table Module 22
Figure 10: Create Table Module Info 23
Figure 11: Create Table Position 24
Figure 12: Create Table Instructor 25
Figure 13: Create Table Instructor_Info 26
Figure 14: Create Table Instructor Contact Info 27
Figure 15: Create Table Instructor Address 28
Figure 16: Create Table Instructor Address Type 29
Figure 17: Create Table Student 30
Figure 18: Create Table Marks 31
Figure 19: Create Table Student Contact Info 32
Figure 20: Create Table Student Address 33
Figure 21: Create Table Student Address Type 34
Figure 22: Inserting Values in Course 35
Figure 23: Course Table 36
Figure 24: Inserting Values in Specification 37
Figure 25: Specification Table 37
Figure 26: Inserting Values in Specification Info 38
Figure 27: Specification Info Table 39
Figure 28: Inserting Values in Class 40
Figure 29: Class Table 40

Figure 30: Inserting Values in Module 41
Figure 31: Module Table 42
Figure 32: Inserting Values in Module Info 43
Figure 33: Module Info Table 44
Figure 34: Inserting Values in Position 45
Figure 35: Position Table 45
Figure 36: Inserting Values in Instructor 47
Figure 37: Instructor Table 47
Figure 38: Inserting Values in Instructor Info 49
Figure 39: Instructor Info Table 49
Figure 40: Inserting Values in Instructor Contact Info 51
Figure 41: Instructor Contact Info Table 52
Figure 42: Inserting Values in Instructor Address1 54
Figure 43: Inserting Values in Instructor Address2 55
Figure 44: Instructor Address Table 55
Figure 45: Inserting Values in Instructor Address Type 58
Figure 46: Instructor Address Type Table 58
Figure 47: Inserting Values in Student 60
Figure 48: Student Table 61
Figure 49: Inserting Values in Marks 62
Figure 50: Marks Table 63
Figure 51: Inserting Values in Student Contact Info 65
Figure 52: Student Contact Info Table 66
Figure 53: Inserting Values in Student Address1 68
Figure 54: Inserting Values in Student Address2 69
Figure 55: Student Address Table 69
Figure 56: Inserting Values in Student Address Type 71
Figure 57: Student Address Type Table 72
Figure 58: Commit 72
Figure 59: Information Query1 74
Figure 60: Information Query 2 75

Figure 61: Information Query 3 76
Figure 62: Information Query 4 77
Figure 63: Information Query 5 79
Figure 64: Information Query 6 80
Figure 65: Information Query 7 82
Figure 66: Information Query 8 83
Figure 67: Information Query 9 84
Figure 68: Information Query 10 85
Figure 69: Transaction Query 1 87
Figure 70: Transaction Query 2 89
Figure 71: Transaction Query 3 90
Figure 72: Transaction query 4 92
Figure 73: Transaction Query 5 93
Figure 74: Transaction Query 6 94
Figure 75: Creating Dump file. 95
Figure 76: Drop Tables 98

List of Tables

Table 1: Entities and Attributes 4

1. Introduction

1.1. Introduction of the College

Islington College was established in 1996 as an associate college of Innovate Nepal Group (ING). Innovate Nepal Group (ING) is an investment holding company that focuses on innovating higher education in Nepal with foreign qualifications but with local context. The college is located in Kamalpokhari, Kathmandu. The college aims to deliver globally accepted degree programmes in direct partnership with London Metropolitan University. Since then the college has been providing quality education in IT and Business in Kathmandu at an affordable price.

Besides providing quality education, the college also keeps their students entertained with various events such as Britain Fair, AI Competitions, spring carnivals and other interactive talk shows. Furthermore, there are various libraries, playgrounds, cafeterias for students. The college has been working continuously to convert a shy high school student to a highly professional IT personnel. The courses offered in this college are approved by the Ministry of Education. Hence, the graduates can work for the government of the country or private companies nationally or internationally. The college have trained almost 3800 alumnus who are working in various reputed places making the college proud,

Since the IT and business industry is changing constantly around us, the main aim of the college is to help students train with the latest and most up-to-date skills and knowledge enabling them to be competitive and industry ready IT and business personnel. The vision of this college is to be the most prestigious college of not only Kathmandu but also of Nepal. The college have been working to produce industry ready graduates for a couple of decades now and the graduate employment rate of 98.6% stands as a proof. Islington have been providing and will always provide quality education with their team of highly trained professionals and the student friendly environment of the college. Along with the education, the college is also training the students to have leadership skills, responsiveness, professionalism, a sense of belonging and building trust and integrity.

1.2. Current Business Activities and Operations

Islington College has been satisfying the students and their parents by implementing following business activities and operations:

i. The college provides four degrees in IT i.e. BSc(Hons) Multimedia Technologies, BSc(Hons) Computing, BSc(Hons) Computer Networking & IT Security and MSc IT & Applied Security and four degrees in business i.e. BBA(International Business), BBA(Finance), BBA(Marketing) and MBA.
ii. The college facilitates their students with 6 blocks, 10 computer labs, 3 lecture halls, 3 cafeterias, 4 seminar rooms, 8 tutorial rooms, 2 audio video studios and 2 learning zones.
iii. The college has an excellent environment for students to read as well as libraries, basketball courts, cafeterias etc. where they can perform recreational activities.
iv. The college has many highly trained instructors who have taught over 3800 alumnus from the college.
v. There is a huge parking area in the college so that the instructors and students can park their vehicles.
vi. Different groups are assigned to students. One group may have 20-30 students. Each group has different time tables.
vii. Each week, for each module there is a lecture class taken by a certain lecturer, one tutorial class and one lab class both taken by the same tutor.
viii. The required data of all the instructors and students should be provided to the college. The contact information such as address, phone number, fax number, email address of each person should be stored.
ix. The instructor should have at least have a Bachelor's degree in the related field.
x. The student trying to enroll in the college should have passed high school with at least a B+ or else they have to add one more year in their respective bachelor's degrees.
xi. Various scholarship programmes are provided to the students according to their educational achievements and their financial conditions.

1.3. Business Rules

The college needs to follow some rules for the administration to run smoothly. The college implements various business rules some of which are listed below:

i. The college database should be able to keep track of all the associated people i.e. instructor and student.
ii. Each person must give their name, date of birth, gender, email address and address to be recorded in the database.
iii. Each person may or may not have multiple addresses, one of which must be a mailing address.
iv. A person may or may not have two phone numbers and the fax number is not necessary to provide.
v. Only one instructor could be a course leader for one course and there is one module leader for each module.
vi. A specification can have multiple modules and a module can be in different specifications.
vii. An instructor or a student can only be enrolled in one course.
viii. A course may or may not have multiple specifications.
ix. A course or a specification must have many students.
x. A module is taught in one particular class whereas multiple modules can be taught in a class.
xi. Student fee can differ from specification to specification and teacher's salary can differ by their position in the college.
xii. Each module has a specific amount of credit hours.

1.4. Creation of Entities and Attributes

Entities are distinguishable real like objects that exist. In a database, the objects that can be stored or retrieved are the only ones that could be called entities. Attributes describe the characteristics or properties of an entity in a database. An entity in a database consists of fixed sets of attributes. In relational models, entities are represented by tables whereas the column header of the table describes the attributes. (T, 2019)

After studying to the case study, the following entities with their corresponding attributes were separated:

Courses	Course ID(PK), Course name
Specification	Specification ID(PK), Specifications name, Fees, Course ID(FK), Module ID, Module name, Credit hours, Class ID, Class name, Block name
Instructor	Instructor ID(PK), Name, Date of Birth, Salary, Gender, Email address, Instructor position, Module ID(FK), Course ID(FK), Instructor Address ID, Country, Province, City, Street, Address Type House Number, Phone Number1, Phone Number2, Fax Number
Student	Student ID(PK), Name, Date of Birth, Gender, Email address, Semester, Enrollment date, Marks, Course ID(FK), Specification ID(FK), Student Address ID, Country, Province, City, Street, Address Type House Number, Phone Number1, Phone Number2, Fax Number

Table 1: Entities and Attributes

1.5. Initial ERD

The initial ERD created from the entities and attributes separated in Table 1 is shown below:

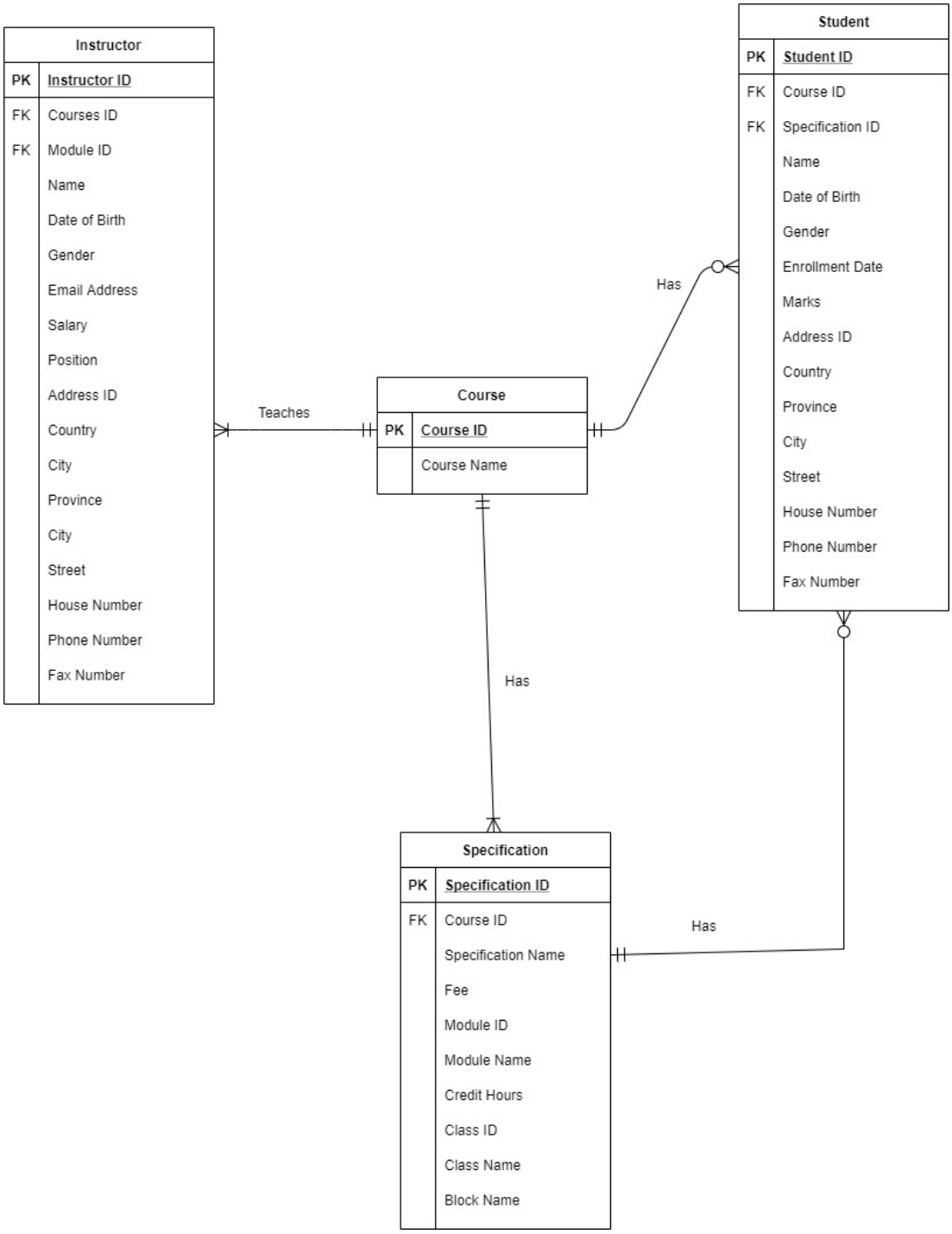

Figure 1: Initial ERD

Assumptions in Initial ERD

Following assumptions were made while showing the relationships in the initial erd:

- In one module, more than one instructor should be assigned into a module.
- One instructor may or may not be involved in more than one module.
- A student and instructor may have one or more addresses, but no two students or instructors can share an address.
- A student can only enroll in one course whereas one course must have more than one student.
- An instructor can only teach in one course but a course must have more than one instructor.
- A specification must have many modules but a module may or may not be included in many specifications.
- A course must have many specifications and one specification can only be included in one course.

The ERD shown above has many data redundancies and data anomalies. To remove the redundancies and anomalies, normalization must be done. As we can see in the initial ERD, there is many-to-many relationship within the entity i.e. between the attributes. A proper ERD must not have any many-to-many relationships. Furthermore, during the insertion of data, there occurs a lot of data redundancies. Also, there are update, delete and insert anomalies which makes the database unconventional.

Therefore, to remove redundancies, anomalies and for the easy insertion and retrieval of data, normalization is done.

2. Normalization

Normalization is the process of reorganizing data in a database so that it meets two basic requirements i.e. no redundancy of data, all data is stored in only one place and data dependencies are logical, all related data items are stored together. (Techopedia, 2020)

Normalization helps us to get data into simpler form that shows entity types, their respective attributes and their relationship between them to avoid unnecessary duplication of data. The process starts from pre-documented sets of attributes which is then tried to group and regroup without causing data inconsistencies in such a way that anomalies are avoided.

2.1. UNF (Un-Normalized Normal Form)

In UNF, we list out all the attributes and then name the entity. The repeating group are enclosed within { }. The unique identifier is chosen and underlined.

The scenario for UNF are:

i. In a course, there are multiple specifications. Hence, specifications is a repeating group.
ii. In a specification, there are multiple modules and students. Hence, module and student are two separate repeating groups under specification.
iii. In a module, multiple teachers are involved. Hence, the instructor is a repeating group under the module.
iv. A student and an instructor can have multiple addresses. Hence, address comes across as a repeating group for both student and instructor.
v. Course ID is chosen as a unique identifier.

Now, after applying UNF. We get,

Course(<u>Course ID</u>, Course Name, {Specification ID, Specification Name, Fees, {Module ID, Module Name, Credit Hours, Class Name, Class ID, Block Name, {Instructor ID, Name, Date of Birth, Salary, Gender, Email Address, Instructor Position, {Instructor Address ID, Address type, Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number}}}, {Student ID, Name, Date of Birth, Gender,

Email Address, Semester, Enrollment date, Marks, {Student Address ID, Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number}}})

2.2. 1NF (First Normal Form)

In 1NF, all the repeating groups are separated into different entities. Every repeating group will have their own unique identifier. But, the unique identifier from the UNF will also be listed as a composite key in the separated entities. The primary key, foreign key and composite key are separated in this step.

The scenario for 1NF is:

i. The repeating group i.e. Specifications, Module, Instructor, Instructor Address, Student and Student Address will be separated from courses.
ii. Course will still be an entity with Course ID as its primary key.
iii. All the new entities will have Course ID as composite primary key.
iv. Each entity will have their own primary key.

After applying 1NF. We get,

Course-1(Course ID, Course Name)

Specification-1(Specification ID, Specifications Name, Fees, Course ID*)

Module-1(Module ID, Module Name, Credit Hours, Class Name, Class ID, Block Name, Course ID*, Specification ID*)

Instructor-1(Instructor ID, Name, Date of Birth, Salary, Gender, Email Address, Instructor Position, Module ID*, Specification ID*, Course ID*)

Instructor Address-1(Instructor Address ID, Address Type, Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number, Instructor ID*, Module ID*, Specification ID*, Course ID*)

Student-1(Student ID, Name, Date of Birth, Gender, Email Address, Semester, Enrollment Date, Marks, Specification ID*, Course ID*)

Student Address-1(Student Address ID, Address Type, Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number, Student ID*, Specification ID*, Course ID*)

2.3. 2NF (Second Normal Form)

In 2NF, all the functional dependencies are removed. If a non-key is fully dependent upon one key, then it is called full functional dependency. If a non-key is dependent upon a part or more than one key, then it is called partial dependency. In this step, we identify functional dependencies and make each determinant the primary key of the new relation. Then, we place all attributes that depend on a given Determinant in Relation with that Determinant as non-key attributes.

The scenarios for 2NF are:

i. Dependencies are checked if there is a primary key and one or more composite keys.
ii. The composite keys are already determined in 1NF.
iii. The partial dependencies are checked in 2NF.
iv. The new entities are created after the attributes are separated from partial or full functional dependencies.

The partial and full functional dependencies are checked below:

For specification:

Specification ID → specifications name, fees

Course ID →

Specification ID, course ID→

For Module:

Module ID → module name, credit hours, block name, class ID, class name

Specification ID →

Course ID →

Module ID, Specification ID, course ID→

For Instructor:

Instructor ID → name, Date of Birth, salary, gender, email address, instructor position

Module ID →

Specification ID →

Course ID →

Instructor ID, Module ID, Specification ID, course ID→

For Instructor Address:

Assumption: To know about if an instructor address is permanent or temporary, we have to know both instructor ID and instructor address ID. Hence, address type is dependent on both of them.

Instructor address ID → Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number

Instructor ID → X

Module ID → X

Specification ID → X

Course ID → X

Instructor ID, Instructor address ID → address type

Instructor address ID, Instructor ID, Module ID, Specification ID, course ID→ X

For Student:

Assumption: To know somebody's marks, we need to know what specification and course. The student is enrolled into. Hence, marks is dependent on all three of their IDs.

Student ID → name, Date of Birth, gender, email address, semester, enrollment date

Specification ID → X

Course ID → X

Student ID, specification ID, course ID → marks

For Student Address:

Assumption: To know about if a student address is permanent or temporary, we have to know both student ID and student ID. Hence, address type is dependent on both of them.

Student Address ID → Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number

Student ID → X

Specification ID → X

Course ID → X

Student Address ID, Student ID → address type

Student Address ID, Student ID, specification ID, course ID → X

Since, there is no need of student address ID, student ID, specification id and course id in a single table. So, we can de-normalize that table and remove it from the entities list.

Similarly, there is no need of Instructor address ID, Instructor ID, Module ID, Specification ID, course ID in a single table. So, we can de-normalize that table and remove it from the entities list.

De-normalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. (GeeksForGeeks, 2019)

The final tables for 2NF are:

Course-2(Course ID, Course Name)

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

Specification info-2(Specification ID*, course ID*)

Module-2(Module ID, Module Name, Credit Hours, Class name, Class ID, Block name)

Module Info-2(Module ID*, Specification ID*, course ID*)

Instructor-2(Instructor ID, Name, Date of Birth, Salary, Gender, Email Address, Instructor Position)

Instructor Info-2(Instructor ID*, Module ID*, Specification ID*, Course ID*)

Instructor Address-2(Instructor Address ID, Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number)

Instructor Address Type-2(Instructor ID*, Instructor Address ID*, Address Type)

Student Info-2(Student ID, Name, Date of Birth, Gender, Email Address, Semester, Enrollment Date, Marks)

Marks-2(Student ID*, Specification ID*, Course ID*, Marks)

Student Address-2(Student Address ID, Country, Province, City, Street, House Number, Phone Number1, Phone Number2, Fax Number)

Student Address Type-2(Student Address ID*, Student ID*, Address Type)

2.4. 3NF (Third Normal Form)

In this step, transitive dependencies are detected and then separated to another relation. Transitive dependency exists when there is an intermediate dependency. When a non-key is dependent on another no-key, it is transitive dependency. Transitive dependency can only occur when there are three or more attributes.

The scenarios for 3NF are:

i. The entities with three or more attributes are checked for transitive dependency.
ii. New entities are created with the attributes separated after detection of transitive dependency.

The transitive dependency is tested below:

For Module:

Module ID determines the Class ID and Class ID helps us to know the Class Name and Block Name.

Module ID → Class ID → Class Name, Block Name

Module ID provides us the information of module name, credit hours and class ID.

Module ID → Module Name, Credit hours, Class ID

For Instructor:

Instructor ID gives instructor position and instructor position gives salary.

Instructor ID → Instructor Position → Salary

Instructor ID gives use information of name, DOB, gender, email address, instructor position

Instructor ID → Name, Date of Birth, Gender, Email Address, Instructor Position

For Instructor Address:

Instructor Address ID gives house number and house number gives Phone Number1, Phone Number2, Fax Number

Instructor Address ID → House Number → Phone Number1, Phone Number2, Fax Number

Instructor address ID gives Country, Province, City, Street, House Number

Instructor address ID → Country, Province, City, Street, House Number

For Student Address:

Student Address ID gives house number and house number gives Phone Number1, Phone Number2, Fax Number

Student Address ID → House Number → Phone Number1, Phone Number2, Fax Number

Student address ID gives Country, Province, City, Street, House Number

Student Address ID → Country, Province, City, Street, House Number

The final entities from 3NF are:

Course-3(Course ID, Course Name)

Specification-3(Specification ID, Specification Name, Fees)

Specification info-3(Specification ID*, Course ID*)

Module-3(Module ID, Module Name, Credit Hours, Class ID*)

Class-3(Class ID, Class Name, Block Name)

Module Info-3(Module ID*, Specification ID*, Course ID*)

Instructor-3(Instructor ID, Name, Date of Birth, Gender, Email Address, Instructor position*)

Position-3(Instructor Position, Salary)

Instructor Info-3(Instructor ID*, Module ID*, Specification ID*, Course ID*)

Instructor Address-3(Instructor address ID, Country, Province, City, Street, House Number*)

Instructor Contact Info-3(House Number, Phone Number1, Phone Number2, Fax Number)

Instructor Address Type-3(Instructor ID*, Instructor address ID*, Address Type)

Student-3(Student ID, Name, Date of Birth, Gender, Email Address, Semester, Enrollment Date)

Marks-3(Student ID*, Specification ID*, Course ID*, Marks)

Students Address-3(Student Address ID, Country, Province, City, Street, House Number*)

Student Contact Info-3(House Number, Phone Number1, Phone Number2, Fax Number)

Student Address Type-3(Student Address ID*, Student ID*, Address Type)

3. Final Entity Relationship Diagram

The following diagram represents the final ERD of the college record system that we prepared.

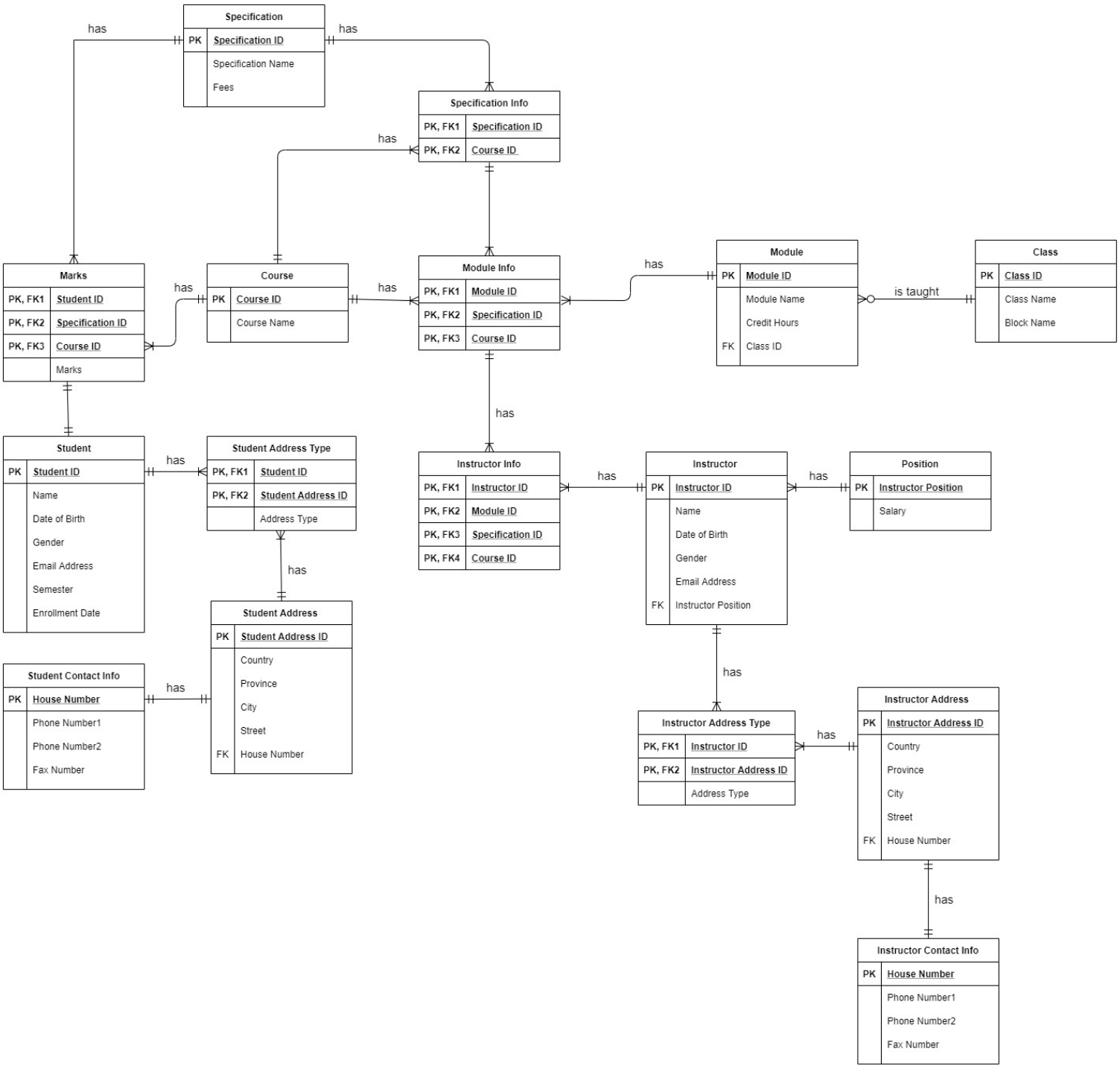

Figure 2: Final ERD

In the final ERD above, we can see proper relationships between the entities. The ERD above is made after normalization. The initial ERD had four entities namely Student, Instructor, Courses and Specification. After normalization, there are 17 entities including the main entities and their bridging entity. The address from both student and instructor has been separated into student address and instructor address respectively. Similarly, the instructor position has been separated. The module entity is also separated from specification and class is separated from module. The entities have various bridging entities connecting various other entities to avoid many to many relationships.

After the normalization, data redundancies and anomalies have been removed. Also, the chances of data duplication has been removed. The management of data would be faster as updating one entity wouldn't harm others. Also, deletion or insertion wouldn't be complicated as it was before. Also, entities can be joined when needed by using different commands.

4. Database Implementation

Oracle SQL plus is used to create tables and populate data into the. We use various SQL commands to create tables in SQL plus. But before that we have to connect to the system of SQL plus. We use the command “connect system” and then insert password to login into the system.

```
SQL> connect system
Enter password:
Connected.
SQL>
```

Figure 3: Connect system

Now, a user is created to create tables in SQL plus. The following command is used to create a user.

CREATE USER Coursework IDENTIFIED BY islington;

After the user is created, we login to the user using the “connect Coursework” command and after entering the password, we can login to the user.

```
SQL> CREATE USER Coursework IDENTIFIED BY islington;

User created.

SQL> GRANT CONNECT, RESOURCE TO Coursework;

Grant succeeded.

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

Figure 4: Connect User

4.1. Table Generation

DDL commands are used to create or modify the structure of database objects in a database. The three commonly used DDLs are: CREATE, ALTER and DROP. We use "CREATE TABLE" command to create a table. ALTER is used to modify a column or row in a table and DROP is used to drop a table or even a user.

Here, we have used the "CREATE TABLE" command to create a table. Various constraints are used to uniquely identify primary key and foreign key. Also, different data types such as VARCHAR2, NUMBER and DATE. Also, some of the attributes are NOT NULL and UNIQUE.

4.1.1. Create Table Course

CREATE TABLE Course(

Course_id VARCHAR2(10) NOT NULL,

Course_name VARCHAR2(20) NOT NULL,

CONSTRAINT Course_id_pk PRIMARY KEY(Course_id));

```
SQL> CREATE TABLE Course(
  2  Course_id VARCHAR2(10) NOT NULL,
  3  Course_name VARCHAR2(20) NOT NULL,
  4  CONSTRAINT Course_id_pk PRIMARY KEY(Course_id));

Table created.

SQL> DESC Course;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSE_ID                                 NOT NULL VARCHAR2(10)
 COURSE_NAME                               NOT NULL VARCHAR2(20)
```

Figure 5: Create Table Course

4.1.2. Create Table Specification

CREATE TABLE Specification(

Specification_id VARCHAR2(20) NOT NULL,

Specification_name VARCHAR2(20) NOT NULL,

Fees NUMBER(7),

CONSTRAINT specification_id_pk PRIMARY KEY(Specification_id));

```
SQL> CREATE TABLE Specification(
  2  Specification_id VARCHAR2(20) NOT NULL,
  3  Specification_name VARCHAR2(20) NOT NULL,
  4  Fees NUMBER(7),
  5  CONSTRAINT specification_id_pk PRIMARY KEY(Specification_id));

Table created.

SQL> DESC Specification;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SPECIFICATION_ID                          NOT NULL VARCHAR2(20)
 SPECIFICATION_NAME                        NOT NULL VARCHAR2(20)
 FEES                                               NUMBER(7)
```

Figure 6: Create Table Specification

4.1.3. Create Table Specification Info

CREATE TABLE Specification_Info(

Specification_id VARCHAR2(20) NOT NULL,

Course_id VARCHAR2(10) NOT NULL,

CONSTRAINT specs_course_id_pk PRIMARY KEY (Specification_id, Course_id),

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

CONSTRAINT course1_id_fk FOREIGN KEY(Course_id) REFERENCES Course(Course_id));

```
SQL> CREATE TABLE Specification_Info(
  2  Specification_id VARCHAR2(20) NOT NULL,
  3  Course_id VARCHAR2(10) NOT NULL,
  4  CONSTRAINT specs_course_id_pk PRIMARY KEY (Specification_id, Course_id),
  5  CONSTRAINT specs_id_fk FOREIGN KEY(Specification_id) REFERENCES Specification(Specification_id),
  6  CONSTRAINT course1_id_fk FOREIGN KEY(Course_id) REFERENCES Course(Course_id));

Table created.

SQL> DESC Specification_Info;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SPECIFICATION_ID                          NOT NULL VARCHAR2(20)
 COURSE_ID                                 NOT NULL VARCHAR2(10)
```

Figure 7: Create Table Specification Info

4.1.4. Create Table Class

CREATE TABLE Class(

Class_id VARCHAR2(10) NOT NULL,

Class_name VARCHAR2(20) NOT NULL,

Block_name VARCHAR2(20) NOT NULL,

CONSTRAINT class_id_pk PRIMARY KEY(Class_id));

```
SQL> CREATE TABLE Class(
  2  Class_id VARCHAR2(10) NOT NULL,
  3  Class_name VARCHAR2(20) NOT NULL,
  4  Block_name VARCHAR2(20) NOT NULL,
  5  CONSTRAINT class_id_pk PRIMARY KEY(Class_id));

Table created.

SQL> DESC Class;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CLASS_ID                                  NOT NULL VARCHAR2(10)
 CLASS_NAME                                NOT NULL VARCHAR2(20)
 BLOCK_NAME                                NOT NULL VARCHAR2(20)
```

Figure 8: Create Table Class

4.1.5. Create Table Module

CREATE TABLE Module(

Module_id VARCHAR2(10) NOT NULL,

Module_name VARCHAR2(20) NOT NULL,

Credit_hours NUMBER(3) NOT NULL,

Class_id VARCHAR2(10) NOT NULL,

CONSTRAINT module_id_pk PRIMARY KEY(Module_id),

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

```
SQL> CREATE TABLE Module(
  2  Module_id VARCHAR2(10) NOT NULL,
  3  Module_name VARCHAR2(20) NOT NULL,
  4  Credit_hours NUMBER(3) NOT NULL,
  5  Class_id VARCHAR2(10) NOT NULL,
  6  CONSTRAINT module_id_pk PRIMARY KEY(Module_id),
  7  CONSTRAINT class_id_fk FOREIGN KEY(Class_id) REFERENCES Class(Class_id));

Table created.

SQL> DESC Module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODULE_ID                                 NOT NULL VARCHAR2(10)
 MODULE_NAME                               NOT NULL VARCHAR2(20)
 CREDIT_HOURS                              NOT NULL NUMBER(3)
 CLASS_ID                                  NOT NULL VARCHAR2(10)
```

Figure 9: Create Table Module

4.1.6. Create Table Module_info

CREATE TABLE Module_Info(

Module_id VARCHAR2(10) NOT NULL,

Specification_id VARCHAR2(20) NOT NULL,

Course_id VARCHAR2(10) NOT NULL,

CONSTRAINT mod_specs_course_id_pk PRIMARY KEY (Module_id, Specification_id, Course_id),

CONSTRAINT module_id_fk FOREIGN KEY(Module_id) REFERENCES Module (Module_id),

CONSTRAINT spec_id_fk FOREIGN KEY(Specification_id) REFERENCES Specification (Specification_id),

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

```
SQL> CREATE TABLE Module_Info(
  2  Module_id VARCHAR2(10) NOT NULL,
  3  Specification_id VARCHAR2(20) NOT NULL,
  4  Course_id VARCHAR2(10) NOT NULL,
  5  CONSTRAINT mod_specs_course_id_pk PRIMARY KEY (Module_id, Specification_id, Course_id),
  6  CONSTRAINT module_id_fk FOREIGN KEY(Module_id) REFERENCES Module(Module_id),
  7  CONSTRAINT spec_id_fk FOREIGN KEY(Specification_id) REFERENCES Specification(Specification_id),
  8  CONSTRAINT course_id_fk FOREIGN KEY(Course_id) REFERENCES Course(Course_id));

Table created.

SQL> DESC Module_Info;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODULE_ID                                 NOT NULL VARCHAR2(10)
 SPECIFICATION_ID                          NOT NULL VARCHAR2(20)
 COURSE_ID                                 NOT NULL VARCHAR2(10)
```

Figure 10: Create Table Module Info

4.1.7. Create Table Position

CREATE TABLE Position(

Instructor_position VARCHAR2(20) NOT NULL,

Salary NUMBER(7) NOT NULL,

CONSTRAINT insp_id_pk PRIMARY KEY(Instructor_position));

```
SQL> CREATE TABLE Position(
  2  Instructor_position VARCHAR2(20) NOT NULL,
  3  Salary NUMBER(7) NOT NULL,
  4  CONSTRAINT insp_id_pk PRIMARY KEY(Instructor_position));

Table created.

SQL> DESC Position;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTOR_POSITION                       NOT NULL VARCHAR2(20)
 SALARY                                    NOT NULL NUMBER(7)
```

Figure 11: Create Table Position

4.1.8. Create Table Instructor

CREATE TABLE Instructor(

Instructor_id VARCHAR2(15) NOT NULL,

Name VARCHAR2(20) NOT NULL,

Date_of_Birth DATE NOT NULL,

Gender VARCHAR2(7) NOT NULL,

Email_Address VARCHAR2(40) NOT NULL,

Instructor_position VARCHAR2(20) NOT NULL,

CONSTRAINT ins_id_pk PRIMARY KEY(Instructor_ID),

CONSTRAINT ip_id_fk FOREIGN KEY(Instructor_position) REFERENCES Position (Instructor_position));

```
SQL> CREATE TABLE Instructor(
  2  Instructor_id VARCHAR2(15) NOT NULL,
  3  Name VARCHAR2(20) NOT NULL,
  4  Date_of_Birth DATE NOT NULL,
  5  Gender VARCHAR2(7) NOT NULL,
  6  Email_Address VARCHAR2(40) NOT NULL,
  7  Instructor_position VARCHAR2(20) NOT NULL,
  8  CONSTRAINT ins_id_pk PRIMARY KEY(Instructor_ID),
  9  CONSTRAINT ip_id_fk FOREIGN KEY(Instructor_position) REFERENCES Position(Instructor_position));

Table created.

SQL> DESC Instructor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTOR_ID                             NOT NULL VARCHAR2(15)
 NAME                                      NOT NULL VARCHAR2(20)
 DATE_OF_BIRTH                             NOT NULL DATE
 GENDER                                    NOT NULL VARCHAR2(7)
 EMAIL_ADDRESS                             NOT NULL VARCHAR2(40)
 INSTRUCTOR_POSITION                       NOT NULL VARCHAR2(20)
```

Figure 12: Create Table Instructor

4.1.9. Create Table Instructor_Info

CREATE TABLE Instructor_Info(

Instructor_id VARCHAR2(15) NOT NULL,

Module_id VARCHAR2(10) NOT NULL,

Specification_id VARCHAR2(20) NOT NULL,

Course_id VARCHAR2(10) NOT NULL,

CONSTRAINT ins_mod_specs_course_id_pk PRIMARY KEY (Instructor_id, Module_id, Specification_id, Course_id),

CONSTRAINT ins_id_fk FOREIGN KEY(Instructor_id) REFERENCES Instructor(Instructor_id),

CONSTRAINT module2_id_fk FOREIGN KEY(Module_id) REFERENCES Module (Module_id),

CONSTRAINT specs2_id_fk FOREIGN KEY(Specification_id) REFERENCES Specification(Specification_id),

CONSTRAINT course2_id_fk FOREIGN KEY(Course_id) REFERENCES Course (Course_id));

```
SQL> CREATE TABLE Instructor_Info(
  2  Instructor_id VARCHAR2(15) NOT NULL,
  3  Module_id VARCHAR2(10) NOT NULL,
  4  Specification_id VARCHAR2(20) NOT NULL,
  5  Course_id VARCHAR2(10) NOT NULL,
  6  CONSTRAINT ins_mod_specs_course_id_pk PRIMARY KEY (Instructor_id, Module_id, Specification_id, Course_id),
  7  CONSTRAINT ins_id_fk FOREIGN KEY(Instructor_id) REFERENCES Instructor(Instructor_id),
  8  CONSTRAINT module2_id_fk FOREIGN KEY(Module_id) REFERENCES Module(Module_id),
  9  CONSTRAINT specs2_id_fk FOREIGN KEY(Specification_id) REFERENCES Specification(Specification_id),
 10  CONSTRAINT course2_id_fk FOREIGN KEY(Course_id) REFERENCES Course(Course_id));

Table created.

SQL> DESC Instructor_Info;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTOR_ID                             NOT NULL VARCHAR2(15)
 MODULE_ID                                 NOT NULL VARCHAR2(10)
 SPECIFICATION_ID                          NOT NULL VARCHAR2(20)
 COURSE_ID                                 NOT NULL VARCHAR2(10)
```

Figure 13: Create Table Instructor_Info

4.1.10. Create Table Instructor Contact Info

CREATE TABLE Instructor_Contact_Info(

House_Number NUMBER(5) NOT NULL,

Phone_Number1 NUMBER(12) UNIQUE,

Phone_Number2 NUMBER(12) UNIQUE,

Fax_Number NUMBER(12) UNIQUE,

CONSTRAINT hn_id_pk PRIMARY KEY(House_Number));

```
SQL> CREATE TABLE Instructor_Contact_Info(
  2  House_Number NUMBER(5) NOT NULL,
  3  Phone_Number1 NUMBER(12) UNIQUE,
  4  Phone_Number2 NUMBER(12) UNIQUE,
  5  Fax_Number NUMBER(12) UNIQUE,
  6  CONSTRAINT hn_id_pk PRIMARY KEY(House_Number));

Table created.

SQL> DESC Instructor_Contact_Info;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 HOUSE_NUMBER                              NOT NULL NUMBER(5)
 PHONE_NUMBER1                                      NUMBER(12)
 PHONE_NUMBER2                                      NUMBER(12)
 FAX_NUMBER                                         NUMBER(12)
```

Figure 14: Create Table Instructor Contact Info

4.1.11. Create Table Instructor Address

CREATE TABLE Instructor_Address(

Instructor_Address_ID VARCHAR2(25) NOT NULL,

Country VARCHAR2(20) NOT NULL,

Province VARCHAR2(20) NOT NULL,

City VARCHAR2(20) NOT NULL,

Street VARCHAR2(20) NOT NULL,

House_Number NUMBER(5) NOT NULL,

CONSTRAINT ia_id_pk PRIMARY KEY(Instructor_Address_ID),

CONSTRAINT hn_id_fk FOREIGN KEY(House_Number) REFERENCES Instructor_Contact_Info(House_Number));

```
SQL> CREATE TABLE Instructor_Address(
  2  Instructor_Address_ID VARCHAR2(25) NOT NULL,
  3  Country VARCHAR2(20) NOT NULL,
  4  Province VARCHAR2(20) NOT NULL,
  5  City VARCHAR2(20) NOT NULL,
  6  Street VARCHAR2(20) NOT NULL,
  7  House_Number NUMBER(5) NOT NULL,
  8  CONSTRAINT ia_id_pk PRIMARY KEY(Instructor_Address_ID),
  9  CONSTRAINT hn_id_fk FOREIGN KEY(House_Number) REFERENCES Instructor_Contact_Info(House_Number));

Table created.

SQL> DESC Instructor_Address;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTOR_ADDRESS_ID                     NOT NULL VARCHAR2(25)
 COUNTRY                                   NOT NULL VARCHAR2(20)
 PROVINCE                                  NOT NULL VARCHAR2(20)
 CITY                                      NOT NULL VARCHAR2(20)
 STREET                                    NOT NULL VARCHAR2(20)
 HOUSE_NUMBER                              NOT NULL NUMBER(5)
```

Figure 15: Create Table Instructor Address

4.1.12. Create Table Instructor Address Type

CREATE TABLE Instructor_Address_Type(

Instructor_ID VARCHAR2(15) NOT NULL,

Instructor_Address_ID VARCHAR2(25) NOT NULL,

Address_Type VARCHAR2(15) NOT NULL,

CONSTRAINT ins_add_id_pk PRIMARY KEY (Instructor_ID, Instructor_Address_ID),

CONSTRAINT insi_id_fk FOREIGN KEY(Instructor_ID) REFERENCES Instructor (Instructor_ID),

CONSTRAINT insia_id_fk FOREIGN KEY(Instructor_Address_ID) REFERENCES Instructor_Address(Instructor_Address_ID));

```
SQL> CREATE TABLE Instructor_Address_Type(
  2  Instructor_ID VARCHAR2(15) NOT NULL,
  3  Instructor_Address_ID VARCHAR2(25) NOT NULL,
  4  Address_Type VARCHAR2(15) NOT NULL,
  5  CONSTRAINT ins_add_id_pk PRIMARY KEY (Instructor_ID, Instructor_Address_ID),
  6  CONSTRAINT insi_id_fk FOREIGN KEY(Instructor_ID) REFERENCES Instructor (Instructor_ID),
  7  CONSTRAINT insia_id_fk FOREIGN KEY(Instructor_Address_ID) REFERENCES Instructor_Address(Instructor_Address_ID));

Table created.

SQL> DESC Instructor_Address_Type;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTOR_ID                             NOT NULL VARCHAR2(15)
 INSTRUCTOR_ADDRESS_ID                     NOT NULL VARCHAR2(25)
 ADDRESS_TYPE                              NOT NULL VARCHAR2(15)
```

Figure 16: Create Table Instructor Address Type

4.1.13. Create Table Student

CREATE TABLE Student(

Student_id VARCHAR2(13) NOT NULL,

Name VARCHAR2(20) NOT NULL,

Date_of_Birth DATE NOT NULL,

Gender VARCHAR2(8) NOT NULL,

Email_Address VARCHAR2(40) NOT NULL,

Semester VARCHAR2(10) NOT NULL,

Enrollment_Date DATE NOT NULL,

CONSTRAINT st_id_pk PRIMARY KEY(Student_ID));

```
SQL> CREATE TABLE Student(
  2  Student_id VARCHAR2(13) NOT NULL,
  3  Name VARCHAR2(20) NOT NULL,
  4  Date_of_Birth DATE NOT NULL,
  5  Gender VARCHAR2(8) NOT NULL,
  6  Email_Address VARCHAR2(40) NOT NULL,
  7  Semester VARCHAR2(10) NOT NULL,
  8  Enrollment_Date DATE NOT NULL,
  9  CONSTRAINT st_id_pk PRIMARY KEY(Student_ID));

Table created.

SQL> DESC Student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL VARCHAR2(13)
 NAME                                      NOT NULL VARCHAR2(20)
 DATE_OF_BIRTH                             NOT NULL DATE
 GENDER                                    NOT NULL VARCHAR2(8)
 EMAIL_ADDRESS                             NOT NULL VARCHAR2(40)
 SEMESTER                                  NOT NULL VARCHAR2(10)
 ENROLLMENT_DATE                           NOT NULL DATE
```

Figure 17: Create Table Student

4.1.14. Create Table Marks

CREATE TABLE Marks(

Student_id VARCHAR2(13) NOT NULL,

Specification_id VARCHAR2(20) NOT NULL,

Course_id VARCHAR2(10) NOT NULL,

Marks NUMBER(5) NOT NULL,

CONSTRAINT st_spec_cou_id_pk PRIMARY KEY(Student_ID, Specification_id, Course_id),

CONSTRAINT stu_id_fk FOREIGN KEY(Student_id) REFERENCES Student (Student_id),

CONSTRAINT specs3_id_fk FOREIGN KEY(Specification_id) REFERENCES Specification(Specification_id),

CONSTRAINT course3_id_fk FOREIGN KEY(Course_id) REFERENCES Course(Course_id));

```
SQL> CREATE TABLE Marks(
  2  Student_id VARCHAR2(13) NOT NULL,
  3  Specification_id VARCHAR2(20) NOT NULL,
  4  Course_id VARCHAR2(10) NOT NULL,
  5  Marks NUMBER(5) NOT NULL,
  6  CONSTRAINT st_spec_cou_id_pk PRIMARY KEY(Student_ID, Specification_id, Course_id),
  7  CONSTRAINT stu_id_fk FOREIGN KEY(Student_id) REFERENCES Student(Student_id),
  8  CONSTRAINT specs3_id_fk FOREIGN KEY(Specification_id) REFERENCES Specification(Specification_id),
  9  CONSTRAINT course3_id_fk FOREIGN KEY(Course_id) REFERENCES Course(Course_id));

Table created.

SQL> DESC Marks;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL VARCHAR2(13)
 SPECIFICATION_ID                          NOT NULL VARCHAR2(20)
 COURSE_ID                                 NOT NULL VARCHAR2(10)
 MARKS                                     NOT NULL NUMBER(5)
```

Figure 18: Create Table Marks

4.1.15. Create Table Student Contact Info

CREATE TABLE Student_Contact_Info(

House_Number NUMBER(5) NOT NULL,

Phone_Number1 NUMBER(12) UNIQUE,

Phone_Number2 NUMBER(12) UNIQUE,

Fax_Number NUMBER(12) UNIQUE,

CONSTRAINT hn1_id_pk PRIMARY KEY(House_Number));

```
SQL> CREATE TABLE Student_Contact_Info(
  2  House_Number NUMBER(5) NOT NULL,
  3  Phone_Number1 NUMBER(12) UNIQUE,
  4  Phone_Number2 NUMBER(12) UNIQUE,
  5  Fax_Number NUMBER(12) UNIQUE,
  6  CONSTRAINT hn1_id_pk PRIMARY KEY(House_Number));

Table created.

SQL> DESC Student_Contact_Info;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 HOUSE_NUMBER                              NOT NULL NUMBER(5)
 PHONE_NUMBER1                                      NUMBER(12)
 PHONE_NUMBER2                                      NUMBER(12)
 FAX_NUMBER                                         NUMBER(12)
```

Figure 19: Create Table Student Contact Info

4.1.16. Create Table Student Address

CREATE TABLE Student_Address(

Student_Address_ID VARCHAR2(25) NOT NULL,

Country VARCHAR2(20) NOT NULL,

Province VARCHAR2(20) NOT NULL,

City VARCHAR2(20) NOT NULL,

Street VARCHAR2(20) NOT NULL,

House_Number NUMBER(5) NOT NULL,

CONSTRAINT sa_id_pk PRIMARY KEY(Student_Address_ID),

CONSTRAINT hn1_id_fk FOREIGN KEY(House_Number) REFERENCES Student_Contact_Info(House_Number));

```
SQL> CREATE TABLE Student_Contact_Info(
  2  House_Number NUMBER(5) NOT NULL,
  3  Phone_Number1 NUMBER(12) UNIQUE,
  4  Phone_Number2 NUMBER(12) UNIQUE,
  5  Fax_Number NUMBER(12) UNIQUE,
  6  CONSTRAINT hn1_id_pk PRIMARY KEY(House_Number));

Table created.

SQL> DESC Student_Contact_Info;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 HOUSE_NUMBER                              NOT NULL NUMBER(5)
 PHONE_NUMBER1                                      NUMBER(12)
 PHONE_NUMBER2                                      NUMBER(12)
 FAX_NUMBER                                         NUMBER(12)
```

Figure 20: Create Table Student Address

4.1.17. Create Table Student Address Type

CREATE TABLE Student_Address_Type(

Student_ID VARCHAR2(15) NOT NULL,

Student_Address_ID VARCHAR2(25) NOT NULL,

Address_Type VARCHAR2(15) NOT NULL,

CONSTRAINT sa_add_id_pk PRIMARY KEY(Student_ID, Student_Address_ID),

CONSTRAINT stia_id_fk FOREIGN KEY(Student_ID) REFERENCES Student (Student_ID),

CONSTRAINT sttia_id_fk FOREIGN KEY(Student_Address_ID) REFERENCES Student_Address(Student_Address_ID));

```
SQL> CREATE TABLE Student_Address_Type(
  2  Student_ID VARCHAR2(15) NOT NULL,
  3  Student_Address_ID VARCHAR2(25) NOT NULL,
  4  Address_Type VARCHAR2(15) NOT NULL,
  5  CONSTRAINT sa_add_id_pk PRIMARY KEY(Student_ID , Student_Address_ID),
  6  CONSTRAINT stia_id_fk FOREIGN KEY(Student_ID) REFERENCES Student(Student_ID),
  7  CONSTRAINT sttia_id_fk FOREIGN KEY(Student_Address_ID) REFERENCES Student_Address(Student_Address_ID));

Table created.

SQL> DESC Student_Address_Type;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL VARCHAR2(15)
 STUDENT_ADDRESS_ID                        NOT NULL VARCHAR2(25)
 ADDRESS_TYPE                              NOT NULL VARCHAR2(15)
```

Figure 21: Create Table Student Address Type

4.2. Populating Data

'INSERT' command is used to insert values into the created table. 'COMMIT' command is used to save the inserted data. The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command. (TutorialsPoint, 2020). 'SELECT *' is used to select all the data from the table.

4.2.1. Inserting Value in Course

```
INSERT ALL
INTO Course(Course_id, Course_name) VALUES ('BI1', 'B.Sc.IT')
INTO Course(Course_id, Course_name) VALUES ('MI1', 'M.Sc.IT')
INTO Course(Course_id, Course_name) VALUES ('BB1', 'BBA')
INTO Course(Course_id, Course_name) VALUES ('MB1', 'MBA')
SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Course(Course_id, Course_name) VALUES ('BI1', 'B.Sc.IT')
  3  INTO Course(Course_id, Course_name) VALUES ('MI1', 'M.Sc.IT')
  4  INTO Course(Course_id, Course_name) VALUES ('BB1', 'BBA')
  5  INTO Course(Course_id, Course_name) VALUES ('MB1', 'MBA')
  6  SELECT * FROM dual;

4 rows created.
```

Figure 22: Inserting Values in Course

SELECT * FROM Course;

```
SQL> SELECT * FROM Course;

COURSE_ID  COURSE_NAME
---------- --------------------
BI1        B.Sc.IT
MI1        M.Sc.IT
BB1        BBA
MB1        MBA
```

Figure 23: Course Table

4.2.2. Inserting Value in Specification

INSERT ALL

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('C2', 'Computing', '115000')

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('M2', 'Multimedia', '120000')

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('N2', 'Networking', '117000')

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('MI2', 'IT Security', '90000')

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('IB2', 'Intl Business', '100000')

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('MK2', 'Marketing', '105000')

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('F2', 'Finance', '103000')

INTO Specification(Specification_id, Specification_name, Fees) VALUES ('MB2', 'Business', '95000')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('C2', 'Computing', '115000')
  3  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('M2', 'Multimedia', '120000')
  4  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('N2', 'Networking', '117000')
  5  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('MI2', 'IT Security', '90000')
  6  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('IB2', 'Intl Business', '100000')
  7  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('MK2', 'Marketing', '105000')
  8  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('F2', 'Finance', '103000')
  9  INTO Specification(Specification_id, Specification_name, Fees) VALUES ('MB2', 'Business', '95000')
 10  SELECT * FROM dual;

8 rows created.
```

Figure 24: Inserting Values in Specification

SELECT * FROM Specification;

```
SQL> SELECT * FROM Specification;

SPECIFICATION_ID     SPECIFICATION_NAME         FEES
-------------------- -------------------- ----------
C2                   Computing                115000
M2                   Multimedia               120000
N2                   Networking               117000
MI2                  IT Security               90000
IB2                  Intl Business            100000
MK2                  Marketing                105000
F2                   Finance                  103000
MB2                  Business                  95000

8 rows selected.
```

Figure 25: Specification Table

4.2.3. Inserting Value in Specification Info

INSERT ALL

INTO Specification_Info(Specification_id, Course_id) VALUES ('C2', 'BI1')

INTO Specification_Info(Specification_id, Course_id) VALUES ('M2', 'BI1')

INTO Specification_Info(Specification_id, Course_id) VALUES ('N2', 'BI1')

INTO Specification_Info(Specification_id, Course_id) VALUES ('MI2', 'MI1')

INTO Specification_Info(Specification_id, Course_id) VALUES ('IB2', 'BB1')

INTO Specification_Info(Specification_id, Course_id) VALUES ('MK2', 'BB1')

INTO Specification_Info(Specification_id, Course_id) VALUES ('F2', 'BB1')

INTO Specification_Info(Specification_id, Course_id) VALUES ('MB2', 'MB1')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Specification_Info(Specification_id, Course_id) VALUES ('C2', 'BI1')
  3  INTO Specification_Info(Specification_id, Course_id) VALUES ('M2', 'BI1')
  4  INTO Specification_Info(Specification_id, Course_id) VALUES ('N2', 'BI1')
  5  INTO Specification_Info(Specification_id, Course_id) VALUES ('MI2', 'MI1')
  6  INTO Specification_Info(Specification_id, Course_id) VALUES ('IB2', 'BB1')
  7  INTO Specification_Info(Specification_id, Course_id) VALUES ('MK2', 'BB1')
  8  INTO Specification_Info(Specification_id, Course_id) VALUES ('F2', 'BB1')
  9  INTO Specification_Info(Specification_id, Course_id) VALUES ('MB2', 'MB1')
 10  SELECT * FROM dual;

8 rows created.
```

Figure 26: Inserting Values in Specification Info

SELECT * FROM Specification_Info;

```
SQL> SELECT * FROM Specification_Info;

SPECIFICATION_ID     COURSE_ID
-------------------- ----------
C2                   BI1
M2                   BI1
N2                   BI1
MI2                  MI1
IB2                  BB1
MK2                  BB1
F2                   BB1
MB2                  MB1

8 rows selected.
```

Figure 27: Specification Info Table

4.2.4. Inserting Value in Class

INSERT ALL

INTO Class(Class_id, Class_name, Block_name) VALUES ('LT01', 'Buckingham Palace', 'London')

INTO Class(Class_id, Class_name, Block_name) VALUES ('LTO8', 'Lecture Theatre', 'Alumni')

INTO Class(Class_id, Class_name, Block_name) VALUES ('TR07', 'Kanchanjunga', 'Nepal')

INTO Class(Class_id, Class_name, Block_name) VALUES ('LT02', 'Nottingham', 'UK')

INTO Class(Class_id, Class_name, Block_name) VALUES ('TR12', 'Gorkha', 'UK')

INTO Class(Class_id, Class_name, Block_name) VALUES ('TR01', 'Kantipur', 'Nepal')

INTO Class(Class_id, Class_name, Block_name) VALUES ('SR03', 'Piccadilly Circus', 'London')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Class(Class_id, Class_name, Block_name) VALUES ('LT01', 'Buckingham Palace', 'London')
  3  INTO Class(Class_id, Class_name, Block_name) VALUES ('LTO8', 'Lecture Theatre', 'Alumni')
  4  INTO Class(Class_id, Class_name, Block_name) VALUES ('TR07', 'Kanchanjunga', 'Nepal')
  5  INTO Class(Class_id, Class_name, Block_name) VALUES ('LT02', 'Nottingham', 'UK')
  6  INTO Class(Class_id, Class_name, Block_name) VALUES ('TR12', 'Gorkha', 'UK')
  7  INTO Class(Class_id, Class_name, Block_name) VALUES ('TR01', 'Kantipur', 'Nepal')
  8  INTO Class(Class_id, Class_name, Block_name) VALUES ('SR03', 'Piccadilly Circus', 'London')
  9  SELECT * FROM dual;

7 rows created.
```

Figure 28: Inserting Values in Class

SELECT * FROM Class;

```
SQL> SELECT * FROM Class;

CLASS_ID   CLASS_NAME           BLOCK_NAME
---------- -------------------- --------------------
LT01       Buckingham Palace    London
LTO8       Lecture Theatre      Alumni
TR07       Kanchanjunga         Nepal
LT02       Nottingham           UK
TR12       Gorkha               UK
TR01       Kantipur             Nepal
SR03       Piccadilly Circus    London

7 rows selected.
```

Figure 29: Class Table

4.2.5. Inserting Value in Module

INSERT ALL

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('D3', 'Database', 30, 'LT01')

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('SE3', 'Software Eng', 45, 'TR12')

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('SCM3', 'Cyber Security Mgmt', 15, 'SR03')

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('EH3', 'Ethical Hacking', 20, 'TR07')

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('AF3', 'Accounting', 15, 'LTO8')

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('DM3', '3D Modelling', 30, 'LT01')

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('ES3', 'Economics', 25, 'SR03')

INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('SM3', 'Service Marketing', 40, 'LT02')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('D3', 'Database', 30, 'LT01')
  3  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('SE3', 'Software Eng', 45, 'TR12')
  4  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('SCM3', 'Cyber Security Mgmt', 15, 'SR03')
  5  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('EH3', 'Ethical Hacking', 20, 'TR07')
  6  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('AF3', 'Accounting', 15, 'LTO8')
  7  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('DM3', '3D Modelling', 30, 'LT01')
  8  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('ES3', 'Economics', 25, 'SR03')
  9  INTO Module(Module_id, Module_name, Credit_hours, Class_id) VALUES ('SM3', 'Service Marketing', 40, 'LT02')
 10  SELECT * FROM dual;
```

Figure 30: Inserting Values in Module

SELECT * FROM Module;

```
SQL> SELECT * FROM Module;

MODULE_ID  MODULE_NAME          CREDIT_HOURS CLASS_ID
---------- -------------------- ------------ ----------
D3         Database                       30 LT01
SE3        Software Eng                   45 TR12
SCM3       Cyber Security Mgmt            15 SR03
EH3        Ethical Hacking                20 TR07
AF3        Accounting                     15 LTO8
DM3        3D Modelling                   30 LT01
ES3        Economics                      25 SR03
SM3        Service Marketing              40 LT02

8 rows selected.
```

Figure 31: Module Table

4.2.6. Inserting Value in Module Info

INSERT ALL

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('D3', 'C2', 'BI1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SE3', 'C2', 'BI1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SCM3', 'C2', 'BI1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('D3', 'N2', 'BI1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SCM3', 'MI2', 'MI1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('ES3', 'MB2', 'MB1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('EH3', 'N2', 'BI1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('AF3', 'F2', 'BB1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('AF3', 'MK2', 'BB1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('DM3', 'M2', 'BB1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SM3', 'MK2', 'BB1')

INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('ES3', 'IB2', 'BB1')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('D3', 'C2', 'BI1')
  3  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SE3', 'C2', 'BI1')
  4  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SCM3', 'C2', 'BI1')
  5  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('D3', 'N2', 'BI1')
  6  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SCM3', 'MI2', 'MI1')
  7  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('ES3', 'MB2', 'MB1')
  8  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('EH3', 'N2', 'BI1')
  9  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('AF3', 'F2', 'BB1')
 10  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('AF3', 'MK2', 'BB1')
 11  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('DM3', 'M2', 'BB1')
 12  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('SM3', 'MK2', 'BB1')
 13  INTO Module_Info(Module_id, Specification_id, Course_id) VALUES ('ES3', 'IB2', 'BB1')
 14  SELECT * FROM dual;

12 rows created.
```

Figure 32: Inserting Values in Module Info

SELECT * FROM Module_Info;

```
SQL> SELECT * FROM Module_Info;

MODULE_ID  SPECIFICATION_ID     COURSE_ID
---------- -------------------- ----------
D3         C2                   BI1
SE3        C2                   BI1
SCM3       C2                   BI1
D3         N2                   BI1
SCM3       MI2                  MI1
ES3        MB2                  MB1
EH3        N2                   BI1
AF3        F2                   BB1
AF3        MK2                  BB1
DM3        M2                   BB1
SM3        MK2                  BB1

MODULE_ID  SPECIFICATION_ID     COURSE_ID
---------- -------------------- ----------
ES3        IB2                  BB1

12 rows selected.
```

Figure 33: Module Info Table

4.2.7. Inserting Values in Position

INSERT ALL

INTO Position(Instructor_position, salary) VALUES ('Course Leader', 95000)

INTO Position(Instructor_position, salary) VALUES ('Module Leader', 93500)

INTO Position(Instructor_position, salary) VALUES ('Lecturer', 90500)

INTO Position(Instructor_position, salary) VALUES ('Tutor', 91000)

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Position(Instructor_position, salary) VALUES ('Course Leader', 95000)
  3  INTO Position(Instructor_position, salary) VALUES ('Module Leader', 93500)
  4  INTO Position(Instructor_position, salary) VALUES ('Lecturer', 90500)
  5  INTO Position(Instructor_position, salary) VALUES ('Tutor', 91000)
  6  SELECT * FROM dual;

4 rows created.
```

Figure 34: Inserting Values in Position

SELECT * FROM Position;

```
SQL> SELECT * FROM Position;

INSTRUCTOR_POSITION      SALARY
-------------------- ----------
Course Leader             95000
Module Leader             93500
Lecturer                  90500
Tutor                     91000
```

Figure 35: Position Table

4.2.8. Inserting Values in Instructor

```
INSERT ALL

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('YB4', 'Yurisha Banjade', '01-Jan-95', 'Female', 'yurisha.banjade@islingtoncollege.edu.np', 'Module Leader')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('UN4', 'Umesh Nepal', '23-Mar-97', 'Male', 'umesh.nepal@islingtoncollege.edu.np', 'Module Leader')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SY4', 'Saroj Yadav', '10-Dec-89', 'Male', 'saroj.yadav@islingtoncollege.edu.np', 'Tutor')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('BA4', 'Biwash Adhikari', '12-Jan-97', 'Male', 'biwash.adhikari@islingtoncollege.edu.np', 'Course Leader')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SS4', 'Supriya Shrestha', '18-Jun-98', 'Female', 'supriya.shrestha@islingtoncollege.edu.np', 'Lecturer')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SA4', 'Sonique Adhikari', '23-Feb-94', 'Female', 'sonique.adhikari@islingtoncollege.edu.np', 'Course Leader')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('RK4', 'Rohan Khatiwada', '12-Aug-89', 'Male', 'rohan.khatiwada@islingtoncollege.edu.np', 'Module Leader')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('GB4', 'Gopal Bhandari', '29-Apr-91', 'Male', 'gopal.bhandari@islingtoncollege.edu.np', 'Tutor')
```

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('RG4', 'Raju Gaire', '31-Jan-85', 'Male', 'raju.gaire@islingtoncollege.edu.np', 'Course Leader')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SK4', 'Surendra KC', '27-Jul-87', 'Male', 'surendra.kc@islingtoncollege.edu.np', 'Module Leader')

INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SP4', 'Sangita Poudyal', '13-Dec-1989', 'Female', 'sangita.poudyal@islingtoncollege.edu.np', 'Lecturer')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('YB4', 'Yurisha Banjade',
'01-Jan-95', 'Female', 'yurisha.banjade@islingtoncollege.edu.np', 'Module Leader')
  3  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('UN4', 'Umesh Nepal', '23-
Mar-97', 'Male', 'umesh.nepal@islingtoncollege.edu.np', 'Module Leader')
  4  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SY4', 'Saroj Yadav', '10-
Dec-89', 'Male', 'saroj.yadav@islingtoncollege.edu.np', 'Tutor')
  5  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('BA4', 'Biwash Adhikari',
'12-Jan-97', 'Male', 'biwash.adhikari@islingtoncollege.edu.np', 'Course Leader')
  6  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SS4', 'Supriya Shrestha',
'18-Jun-98', 'Female', 'supriya.shrestha@islingtoncollege.edu.np', 'Lecturer')
  7  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SA4', 'Sonique Adhikari',
'23-Feb-94', 'Female', 'sonique.adhikari@islingtoncollege.edu.np', 'Course Leader')
  8  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('RK4', 'Rohan Khatiwada',
'12-Aug-89', 'Male', 'rohan.khatiwada@islingtoncollege.edu.np', 'Module Leader')
  9  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('GB4', 'Gopal Bhandari', '
29-Apr-91', 'Male', 'gopal.bhandari@islingtoncollege.edu.np', 'Tutor')
 10  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('RG4', 'Raju Gaire', '31-J
an-85', 'Male', 'raju.gaire@islingtoncollege.edu.np', 'Course Leader')
 11  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SK4', 'Surendra KC', '27-
Jul-87', 'Male', 'surendra.kc@islingtoncollege.edu.np', 'Module Leader')
 12  INTO Instructor(Instructor_id, Name, Date_of_Birth, Gender, Email_Address, Instructor_position) VALUES ('SP4', 'Sangita Poudyal',
'13-Dec-1989', 'Female', 'sangita.poudyal@islingtoncollege.edu.np', 'Lecturer')
 13  SELECT * FROM dual;
```

Figure 36: Inserting Values in Instructor

SELECT * FROM Instructor;

```
SQL> SELECT * FROM Instructor;

INSTRUCTOR_ID   NAME                DATE_OF_B GENDER  EMAIL_ADDRESS                          INSTRUCTOR_POSITION
--------------- ------------------- --------- ------- -------------------------------------- --------------------
YB4             Yurisha Banjade     01-JAN-95 Female  yurisha.banjade@islingtoncollege.edu.np  Module Leader
UN4             Umesh Nepal         23-MAR-97 Male    umesh.nepal@islingtoncollege.edu.np      Module Leader
SY4             Saroj Yadav         10-DEC-89 Male    saroj.yadav@islingtoncollege.edu.np      Tutor
BA4             Biwash Adhikari     12-JAN-97 Male    biwash.adhikari@islingtoncollege.edu.np  Course Leader
SS4             Supriya Shrestha    18-JUN-98 Female  supriya.shrestha@islingtoncollege.edu.np Lecturer
SA4             Sonique Adhikari    23-FEB-94 Female  sonique.adhikari@islingtoncollege.edu.np Course Leader
RK4             Rohan Khatiwada     12-AUG-89 Male    rohan.khatiwada@islingtoncollege.edu.np  Module Leader
GB4             Gopal Bhandari      29-APR-91 Male    gopal.bhandari@islingtoncollege.edu.np   Tutor
RG4             Raju Gaire          31-JAN-85 Male    raju.gaire@islingtoncollege.edu.np       Course Leader
SK4             Surendra KC         27-JUL-87 Male    surendra.kc@islingtoncollege.edu.np      Module Leader
SP4             Sangita Poudyal     13-DEC-89 Female  sangita.poudyal@islingtoncollege.edu.np  Lecturer
```

Figure 37: Instructor Table

4.2.9. Inserting Values in Instructor Info

```
INSERT ALL

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('BA4', 'D3', 'C2', 'BI1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('RK4', 'ES3', 'MB2', 'MB1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('BA4', 'EH3', 'N2', 'BI1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('RK4', 'SM3', 'MK2', 'BB1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SP4', 'SCM3', 'C2', 'BI1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('YB4', 'D3', 'C2', 'BI1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SY4', 'SM3', 'MK2', 'BB1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SS4', 'SE3', 'C2', 'BI1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('GB4', 'SCM3', 'MI2', 'MI1')

INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SK4', 'ES3', 'MB2', 'MB1')

SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('BA4', 'D3', 'C2', 'BI1')
  3  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('RK4', 'ES3', 'MB2', 'MB1')
  4  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('BA4', 'EH3', 'N2', 'BI1')
  5  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('RK4', 'SM3', 'MK2', 'BB1')
  6  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SP4', 'SCM3', 'C2', 'BI1')
  7  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('YB4', 'D3', 'C2', 'BI1')
  8  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SY4', 'SM3', 'MK2', 'BB1')
  9  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SS4', 'SE3', 'C2', 'BI1')
 10  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('GB4', 'SCM3', 'MI2', 'MI1')
 11  INTO Instructor_Info(Instructor_id, Module_id, Specification_id, Course_id) VALUES ('SK4', 'ES3', 'MB2', 'MB1')
 12  SELECT * FROM dual;

10 rows created.
```

Figure 38: Inserting Values in Instructor Info

SELECT * FROM Instructor_Info;

```
SQL> SELECT * FROM Instructor_Info;

INSTRUCTOR_ID   MODULE_ID  SPECIFICATION_ID     COURSE_ID
--------------- ---------- -------------------- ----------
BA4             D3         C2                   BI1
RK4             ES3        MB2                  MB1
BA4             EH3        N2                   BI1
RK4             SM3        MK2                  BB1
SP4             SCM3       C2                   BI1
YB4             D3         C2                   BI1
SY4             SM3        MK2                  BB1
SS4             SE3        C2                   BI1
GB4             SCM3       MI2                  MI1
SK4             ES3        MB2                  MB1

10 rows selected.
```

Figure 39: Instructor Info Table

4.2.10. Inserting Values in Instructor Contact Info

INSERT ALL

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (91, 9876543210, '', 135441)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (100, 9875643210, 9808765432, '')

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (120, 9867653800, 9872136289, '')

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (122, 9808765497, 9877766537, 546434)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (130, 9856427075, '', '')

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (134, 9870685358, '', 779765)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (140, 9808654795, 9863548726, '')

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (169, 9813568420, 9894650283, 134686)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (170, 9800585386, '', 344769)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (181, 9863468368, '', 576542)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (191, 9876558643, '', 341467)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (240, 9808639273, 9897564827, '')

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (270, 9867364820, '', '')

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (394, 9812345678, 9808734828, '')

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (476, 9823461720, 9837582701, 452768)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (578, 9865724209, '', 786553)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (666, 9876543211, '', 354879)

INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (787, 9876543336, 9803758269, '')

```
SQL> INSERT ALL
  2  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (91, 9876543210, '', 135441)
  3  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (100, 9875643210, 9808765432, '')
  4  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (120, 9867653800, 9872136289, '')
  5  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (122, 9808765497, 9877766537, 546434)
  6  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (130, 9856427075, '', '')
  7  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (134, 9870685358, '', 779765)
  8  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (140, 9808654795, 9863548726, '')
  9  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (169, 9813568420, 9894650283, 134686)
 10  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (170, 9800585386, '', 344769)
 11  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (181, 9863468368, '', 576542)
 12  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (191, 9876558643, '', 341467)
 13  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (240, 9808639273, 9897564827, '')
 14  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (270, 9867364820, '', '')
 15  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (394, 9812345678, 9808734828, '')
 16  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (476, 9823461720, 9837582701, 452768)
 17  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (578, 9865724209, '', 786553)
 18  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (666, 9876543211, '', 354879)
 19  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (787, 9876543336, 9803758269, '')
 20  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (998, 9876541167, 9878262620, 437688)
 21  INTO Instructor_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1074, 9800987668, '', 354656)
 22  SELECT * FROM dual;

20 rows created.
```

Figure 40: Inserting Values in Instructor Contact Info

SELECT * FROM Instructor_Contact_Info;

```
SQL> SELECT * FROM Instructor_Contact_Info;

HOUSE_NUMBER PHONE_NUMBER1 PHONE_NUMBER2 FAX_NUMBER
------------ ------------- ------------- ----------
          91    9876543210                   135441
         100    9875643210    9808765432
         120    9867653800    9872136289
         122    9808765497    9877766537     546434
         130    9856427075
         134    9870685358                   779765
         140    9808654795    9863548726
         169    9813568420    9894650283     134686
         170    9800585386                   344769
         181    9863468368                   576542
         191    9876558643                   341467

HOUSE_NUMBER PHONE_NUMBER1 PHONE_NUMBER2 FAX_NUMBER
------------ ------------- ------------- ----------
         240    9808639273    9897564827
         270    9867364820
         394    9812345678    9808734828
         476    9823461720    9837582701     452768
         578    9865724209                   786553
         666    9876543211                   354879
         787    9876543336    9803758269
         998    9876541167    9878262620     437688
        1074    9800987668                   354656

20 rows selected.
```

Figure 41: Instructor Contact Info Table

4.2.11. Inserting Values in Instructor Address

```
INSERT ALL

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A51', 'Nepal', 'Bagmati', 'Kathmandu', 'Thamel', 100)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A52', 'Nepal', 'Bagmati', 'Kathmandu', 'Bagbazar', 181)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A53', 'Nepal', 'Gandaki', 'Pokhara', 'Prithivi Chowk', 240)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A54', 'Nepal', 'Bagmati', 'Kathmandu', 'Kuleshwor', 122)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A55', 'Nepal', 'Province 1', 'Jhapa', 'Dhulabari', 394)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A56', 'Nepal', 'Bagmati', 'Kathmandu', 'Kalanki', 134)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A57', 'Nepal', 'Lumbini', 'Dang', 'Dang Besi', 476)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A58', 'Nepal', 'Bagmati', 'Kathmandu', 'Baluwatar', 140)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A59', 'Nepal', 'Lumbini', 'Butwal', 'New Road', 578)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A510', 'Nepal', 'Bagmati', 'Kathmandu', 'Indrachowk', 170)

SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A51', 'Nepal', 'Bagmati', '
Kathmandu', 'Thamel', 100)
  3  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A52', 'Nepal', 'Bagmati', '
Kathmandu', 'Bagbazar', 181)
  4  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A53', 'Nepal', 'Gandaki', '
Pokhara', 'Prithivi Chowk', 240)
  5  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A54', 'Nepal', 'Bagmati', '
Kathmandu', 'Kuleshwor', 122)
  6  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A55', 'Nepal', 'Province 1'
, 'Jhapa', 'Dhulabari', 394)
  7  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A56', 'Nepal', 'Bagmati', '
Kathmandu', 'Kalanki', 134)
  8  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A57', 'Nepal', 'Lumbini', '
Dang', 'Dang Besi', 476)
  9  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A58', 'Nepal', 'Bagmati', '
Kathmandu', 'Baluwatar', 140)
 10  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A59', 'Nepal', 'Lumbini', '
Butwal', 'New Road', 578)
 11  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A510', 'Nepal', 'Bagmati',
'Kathmandu', 'Indrachowk', 170)
 12  SELECT * FROM dual;

10 rows created.
```

Figure 42: Inserting Values in Instructor Address1

INSERT ALL

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A511', 'Nepal', 'Province 1', 'Illam', 'Chiya Bagan', 666)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A512', 'Nepal', 'Bagmati', 'Kathmandu', 'Dillibazar', 91)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A513', 'Nepal', 'Bagmati', 'Kathmandu', 'Balkhu', 130)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A514', 'Nepal', 'Bagmati', 'Kathmandu', 'Sitapaila', 120)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A515', 'Nepal', 'Gandaki', 'Pokhara', 'Lakeside', 270)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A516', 'Nepal', 'Province 2', 'Chitwan', 'Rampur', 787)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A517', 'Nepal', 'Bagmati', 'Kathmandu', 'Kamalpokhari', 191)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A518', 'Nepal', 'Sudarpashim', 'Dhangadhi', 'Hasanpur', 998)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A519', 'Nepal', 'Bagmati', 'Hetauda', 'Chitipur', 1074)

INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A520', 'Nepal', 'Bagmati', 'Kathmandu', '169', 169)

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A511', 'Nepal', 'Province 1
', 'Illam', 'Chiya Bagan', 666)
  3  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A512', 'Nepal', 'Bagmati',
'Kathmandu', 'Dillibazar', 91)
  4  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A513', 'Nepal', 'Bagmati',
'Kathmandu', 'Balkhu', 130)
  5  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A514', 'Nepal', 'Bagmati',
'Kathmandu', 'Sitapaila', 120)
  6  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A515', 'Nepal', 'Gandaki',
'Pokhara', 'Lakeside', 270)
  7  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A516', 'Nepal', 'Province 2
', 'Chitwan', 'Rampur', 787)
  8  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A517', 'Nepal', 'Bagmati',
'Kathmandu', 'Kamalpokhari', 191)
  9  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A518', 'Nepal', 'Sudarpashi
m', 'Dhangadhi', 'Hasanpur', 998)
 10  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A519', 'Nepal', 'Bagmati',
'Hetauda', 'Chitipur', 1074)
 11  INTO Instructor_Address(Instructor_Address_ID, Country, Province, City, Street, House_number) VALUES ('A520', 'Nepal', 'Bagmati',
'Kathmandu', '169', 169)
 12  SELECT * FROM dual;

10 rows created.
```

Figure 43: Inserting Values in Instructor Address2

SELECT * FROM Instructor_Address;

```
SQL> SELECT * FROM Instructor_Address;

INSTRUCTOR_ADDRESS_ID     COUNTRY              PROVINCE             CITY                 STREET               HOUSE_NUMBER
------------------------- -------------------- -------------------- -------------------- -------------------- ------------
A51                       Nepal                Bagmati              Kathmandu            Thamel                        100
A52                       Nepal                Bagmati              Kathmandu            Bagbazar                      181
A53                       Nepal                Gandaki              Pokhara              Prithivi Chowk                240
A54                       Nepal                Bagmati              Kathmandu            Kuleshwor                     122
A55                       Nepal                Province 1           Jhapa                Dhulabari                     394
A56                       Nepal                Bagmati              Kathmandu            Kalanki                       134
A57                       Nepal                Lumbini              Dang                 Dang Besi                     476
A58                       Nepal                Bagmati              Kathmandu            Baluwatar                     140
A59                       Nepal                Lumbini              Butwal               New Road                      578
A510                      Nepal                Bagmati              Kathmandu            Indrachowk                    170
A511                      Nepal                Province 1           Illam                Chiya Bagan                   666

INSTRUCTOR_ADDRESS_ID     COUNTRY              PROVINCE             CITY                 STREET               HOUSE_NUMBER
------------------------- -------------------- -------------------- -------------------- -------------------- ------------
A512                      Nepal                Bagmati              Kathmandu            Dillibazar                     91
A513                      Nepal                Bagmati              Kathmandu            Balkhu                        130
A514                      Nepal                Bagmati              Kathmandu            Sitapaila                     120
A515                      Nepal                Gandaki              Pokhara              Lakeside                      270
A516                      Nepal                Province 2           Chitwan              Rampur                        787
A517                      Nepal                Bagmati              Kathmandu            Kamalpokhari                  191
A518                      Nepal                Sudarpashim          Dhangadhi            Hasanpur                      998
A519                      Nepal                Bagmati              Hetauda              Chitipur                     1074
A520                      Nepal                Bagmati              Kathmandu            169                           169
```

Figure 44: Instructor Address Table

4.2.12. Inserting Values in Instructor Address Type

```
INSERT ALL

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('YB4', 'A51', 'Temporary')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('YB4', 'A53', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('UN4', 'A52', 'Temporary')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('UN4', 'A55', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SY4', 'A54', 'Temporary')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SY4', 'A57', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('BA4', 'A56', 'Temporary')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('BA4', 'A59', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SS4', 'A58', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SA4', 'A510', 'Temporary')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SA4', 'A515', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RK4', 'A512', 'Temporary')
```

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RK4', 'A516', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('GB4', 'A513', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RG4', 'A514', 'Temporary')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RG4', 'A519', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SK4', 'A517', 'Temporary')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SK4', 'A518', 'Permanent')

INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SP4', 'A520', 'Temporary')

I/NTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SP4', 'A511', 'Permanent')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('YB4', 'A51', 'Temporary')
  3  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('YB4', 'A53', 'Permanent')
  4  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('UN4', 'A52', 'Temporary')
  5  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('UN4', 'A55', 'Permanent')
  6  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SY4', 'A54', 'Temporary')
  7  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SY4', 'A57', 'Permanent')
  8  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('BA4', 'A56', 'Temporary')
  9  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('BA4', 'A59', 'Permanent')
 10  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SS4', 'A58', 'Permanent')
 11  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SA4', 'A510', 'Temporary')
 12  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SA4', 'A515', 'Permanent')
 13  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RK4', 'A512', 'Temporary')
 14  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RK4', 'A516', 'Permanent')
 15  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('GB4', 'A513', 'Permanent')
 16  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RG4', 'A514', 'Temporary')
 17  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('RG4', 'A519', 'Permanent')
 18  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SK4', 'A517', 'Temporary')
 19  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SK4', 'A518', 'Permanent')
 20  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SP4', 'A520', 'Temporary')
 21  INTO Instructor_Address_Type(Instructor_ID, Instructor_Address_ID, Address_Type) VALUES ('SP4', 'A511', 'Permanent')
 22  SELECT * FROM dual;

20 rows created.
```

Figure 45: Inserting Values in Instructor Address Type

SELECT * FROM Instructor_Address_Type;

```
SQL> SELECT * FROM Instructor_Address_Type;

INSTRUCTOR_ID   INSTRUCTOR_ADDRESS_ID     ADDRESS_TYPE
--------------- ------------------------ ---------------
YB4             A51                      Temporary
YB4             A53                      Permanent
UN4             A52                      Temporary
UN4             A55                      Permanent
SY4             A54                      Temporary
SY4             A57                      Permanent
BA4             A56                      Temporary
BA4             A59                      Permanent
SS4             A58                      Permanent
SA4             A510                     Temporary
SA4             A515                     Permanent

INSTRUCTOR_ID   INSTRUCTOR_ADDRESS_ID     ADDRESS_TYPE
--------------- ------------------------ ---------------
RK4             A512                     Temporary
RK4             A516                     Permanent
GB4             A513                     Permanent
RG4             A514                     Temporary
RG4             A519                     Permanent
SK4             A517                     Temporary
SK4             A518                     Permanent
SP4             A520                     Temporary
SP4             A511                     Permanent

20 rows selected.
```

Figure 46: Instructor Address Type Table

4.2.13. Inserting Values in Student

```
INSERT ALL

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('GG6', 'Grisha Giri', '12-Jan-02', 'Female', 'grisha.giri@islingtoncollege.edu.np', '3rd', '12-Feb-19')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('SA6', 'Srijan Adhikari', '27-Nov-00', 'Male', 'srijan.adhikari@islingtoncollege.edu.np', '5th', '15-Nov-18')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('RS6', 'Rojan Shrestha', '28-Jul-02', 'Male', 'rojan.shrestha@islingtoncollege.edu.np', '2nd', '16-Nov-19')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('AD6', 'Aditi Dhakal', '23-Jun-01', 'Female', 'aditi.dhakal@islingtoncollege.edu.np', '3rd', '16-Feb-19')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('LK6', 'Labbi Karmasta', '30-Jan-00', 'Female', 'labbi.karmasta@islingtoncollege.edu.np', '3rd', '11-Feb-19')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('PP6', 'Prince Panjiyar', '23-Mar-99', 'Male', 'prince.panjiyar@islingtoncollege.edu.np', '4th', '13-Feb-18')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('RP6', 'Rohit Parajuli', '12-Oct-98', 'Male', 'rohit.parajuli@islingtoncollege.edu.np', '6th', '23-Nov-17')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('PG6', 'Pragati Gautam', '13-Sep-99', 'Female', 'pragati.gautam@islingtoncollege.edu.np', '6th', '24-Nov-17')
```

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('AB6', 'Alisha Basnet', '28-Aug-00', 'Female', 'alisha.basnet@islingtoncollege.edu.np', '5th', '15-Nov-18')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('RB6', 'Roman Bhandari', '08-Feb-00', 'Male', 'roman.bhandari@islingtoncollege.edu.np', '4th', '12-Feb-18')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('AG6', 'Ashish Gautam', '25-Apr-99', 'Male', 'ashish.gautam@islingtoncollege.edu.np', '6th', '23-Nov-17')

INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('SR6', 'Simran Ranjit', '15-Oct-98', 'Female', 'simran.ranjit@islingtoncollege.edu.np', '6th', '24-Nov-17')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('GG6', 'Grisha Giri', '12-
Jan-02', 'Female', 'grisha.giri@islingtoncollege.edu.np', '3rd', '12-Feb-19')
  3  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('SA6', 'Srijan Adhikari',
'27-Nov-00', 'Male', 'srijan.adhikari@islingtoncollege.edu.np', '5th', '15-Nov-18')
  4  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('RS6', 'Rojan Shrestha', '
28-Jul-02', 'Male', 'rojan.shrestha@islingtoncollege.edu.np', '2nd', '16-Nov-19')
  5  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('AD6', 'Aditi Dhakal', '23
-Jun-01', 'Female', 'aditi.dhakal@islingtoncollege.edu.np', '3rd', '16-Feb-19')
  6  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('LK6', 'Labbi Karmasta', '
30-Jan-00', 'Female', 'labbi.karmasta@islingtoncollege.edu.np', '3rd', '11-Feb-19')
  7  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('PP6', 'Prince Panjiyar',
'23-Mar-99', 'Male', 'prince.panjiyar@islingtoncollege.edu.np', '4th', '13-Feb-18')
  8  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('RP6', 'Rohit Parajuli', '
12-Oct-98', 'Male', 'rohit.parajuli@islingtoncollege.edu.np', '6th', '23-Nov-17')
  9  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('PG6', 'Pragati Gautam', '
13-Sep-99', 'Female', 'pragati.gautam@islingtoncollege.edu.np', '6th', '24-Nov-17')
 10  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('AB6', 'Alisha Basnet', '2
8-Aug-00', 'Female', 'alisha.basnet@islingtoncollege.edu.np', '5th', '15-Nov-18')
 11  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('RB6', 'Roman Bhandari', '
08-Feb-00', 'Male', 'roman.bhandari@islingtoncollege.edu.np', '4th', '12-Feb-18')
 12  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('AG6', 'Ashish Gautam', '2
5-Apr-99', 'Male', 'ashish.gautam@islingtoncollege.edu.np', '6th', '23-Nov-17')
 13  INTO Student(Student_id, Name, Date_of_Birth, Gender, Email_Address, Semester, Enrollment_Date) VALUES ('SR6', 'Simran Ranjit', '1
5-Oct-98', 'Female', 'simran.ranjit@islingtoncollege.edu.np', '6th', '24-Nov-17')
 14  SELECT * FROM dual;

12 rows created.
```

Figure 47: Inserting Values in Student

SELECT * FROM Student;

```
SQL> SELECT * FROM Student;

STUDENT_ID    NAME                DATE_OF_B GENDER   EMAIL_ADDRESS                          SEMESTER   ENROLLMEN
------------ -------------------- --------- -------- -------------------------------------- ---------- ---------
GG6           Grisha Giri         12-JAN-02 Female   grisha.giri@islingtoncollege.edu.np    3rd        12-FEB-19
SA6           Srijan Adhikari     27-NOV-00 Male     srijan.adhikari@islingtoncollege.edu.np 5th       15-NOV-18
RS6           Rojan Shrestha      28-JUL-02 Male     rojan.shrestha@islingtoncollege.edu.np 2nd        16-NOV-19
AD6           Aditi Dhakal        23-JUN-01 Female   aditi.dhakal@islingtoncollege.edu.np   3rd        16-FEB-19
LK6           Labbi Karmasta      30-JAN-00 Female   labbi.karmasta@islingtoncollege.edu.np 3rd        11-FEB-19
PP6           Prince Panjiyar     23-MAR-99 Male     prince.panjiyar@islingtoncollege.edu.np 4th       13-FEB-18
RP6           Rohit Parajuli      12-OCT-98 Male     rohit.parajuli@islingtoncollege.edu.np 6th        23-NOV-17
PG6           Pragati Gautam      13-SEP-99 Female   pragati.gautam@islingtoncollege.edu.np 6th        24-NOV-17
AB6           Alisha Basnet       28-AUG-00 Female   alisha.basnet@islingtoncollege.edu.np  5th        15-NOV-18
RB6           Roman Bhandari      08-FEB-00 Male     roman.bhandari@islingtoncollege.edu.np 4th        12-FEB-18
AG6           Ashish Gautam       25-APR-99 Male     ashish.gautam@islingtoncollege.edu.np  6th        23-NOV-17

STUDENT_ID    NAME                DATE_OF_B GENDER   EMAIL_ADDRESS                          SEMESTER   ENROLLMEN
------------ -------------------- --------- -------- -------------------------------------- ---------- ---------
SR6           Simran Ranjit       15-OCT-98 Female   simran.ranjit@islingtoncollege.edu.np  6th        24-NOV-17

12 rows selected.
```

Figure 48: Student Table

4.2.14. Inserting Values in Marks

INSERT ALL

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('GG6', 'C2', 'BI1', 70)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('SA6', 'N2', 'BI1', 87)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('RS6', 'F2', 'BB1', 71)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('AD6', 'M2', 'BI1', 90)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('LK6', 'C2', 'BI1', 86)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('PP6', 'MB2', 'MB1', 69)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('PG6', 'M2', 'BI1', 65)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('RP6', 'MI2', 'MI1', 80)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('AB6', 'MK2', 'BB1', 68)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('RB6', 'F2', 'BB1', 85)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('AG6', 'IB2', 'BB1', 95)

INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('SR6', 'N2', 'BI1', 77)

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('GG6', 'C2', 'BI1', 70)
  3  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('SA6', 'N2', 'BI1', 87)
  4  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('RS6', 'F2', 'BB1', 71)
  5  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('AD6', 'M2', 'BI1', 90)
  6  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('LK6', 'C2', 'BI1', 86)
  7  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('PP6', 'MB2', 'MB1', 69)
  8  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('PG6', 'M2', 'BI1', 65)
  9  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('RP6', 'MI2', 'MI1', 80)
 10  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('AB6', 'MK2', 'BB1', 68)
 11  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('RB6', 'F2', 'BB1', 85)
 12  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('AG6', 'IB2', 'BB1', 95)
 13  INTO Marks(Student_id, Specification_id, Course_id, Marks) VALUES ('SR6', 'N2', 'BI1', 77)
 14  SELECT * FROM dual;

12 rows created.
```

Figure 49: Inserting Values in Marks

SELECT * FROM Marks;

```
SQL> SELECT * FROM Marks;

STUDENT_ID    SPECIFICATION_ID     COURSE_ID       MARKS
------------- -------------------- ---------- ----------
GG6           C2                   BI1                70
SA6           N2                   BI1                87
RS6           F2                   BB1                71
AD6           M2                   BI1                90
LK6           C2                   BI1                86
PP6           MB2                  MB1                69
PG6           M2                   BI1                65
RP6           MI2                  MI1                80
AB6           MK2                  BB1                68
RB6           F2                   BB1                85
AG6           IB2                  BB1                95

STUDENT_ID    SPECIFICATION_ID     COURSE_ID       MARKS
------------- -------------------- ---------- ----------
SR6           N2                   BI1                77

12 rows selected.
```

Figure 50: Marks Table

4.2.15. Inserting Values in Student Contact Info

```
INSERT ALL

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1001, 9873697100, 9860469853, 045768)

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1010, 9808761098, 9865077564, 054376)

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1021, 9873777779, 9818248640, '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1042, 9808980780, '', '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1056, 9823456718, 9849644291, '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1069, 9800776698, 9860033237, 065456)

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1075, 9875675534, 9840098585, '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1078, 9809887666, '', '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1089, 9878900760, '', 065454)

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1093, 9887653700, '', '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1096, 9868890665, 9841724570, 065477)

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number)
VALUES (1098, 9851078766, '', 045376)
```

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (2020, 9807610886, '', 023547)

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (3078, 9825987059, 9841808752, 054342)

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (4099, 9843819860, 9860848077, '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (5081, 9841709854, '', '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (6060, 9813126745, 9848819580, '')

INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (7071, 9846871009, '', 078564)

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1001, 9873697100, 9860469853, 045768)
  3  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1010, 9808761098, 9865077564, 054376)
  4  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1021, 9873777779, 9818248640, '')
  5  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1042, 9808980780, '', '')
  6  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1056, 9823456718, 9849644291, '')
  7  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1069, 9800776698, 9860033237, 065456)
  8  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1075, 9875675534, 9840098585, '')
  9  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1078, 9809887666, '', '')
 10  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1089, 9878900760, '', 065454)
 11  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1093, 9887653700, '', '')
 12  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1096, 9868890665, 9841724570, 065477)
 13  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (1098, 9851078766, '', 045376)
 14  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (2020, 9807610886, '', 023547)
 15  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (3078, 9825987059, 9841808752, 054342)
 16  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (4099, 9843819860, 9860848077, '')
 17  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (5081, 9841709854, '', '')
 18  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (6060, 9813126745, 9848819580, '')
 19  INTO Student_Contact_Info(House_number, Phone_Number1, Phone_Number2, Fax_Number) VALUES (7071, 9846871009, '', 078564)
 20  SELECT * FROM dual;

18 rows created.
```

Figure 51: Inserting Values in Student Contact Info

SELECT * FROM Student_Contact_Info;

```
SQL> SELECT * FROM Student_Contact_Info;

HOUSE_NUMBER PHONE_NUMBER1 PHONE_NUMBER2 FAX_NUMBER
------------ ------------- ------------- ----------
        1001    9873697100    9860469853      45768
        1010    9808761098    9865077564      54376
        1021    9873777779    9818248640
        1042    9808980780
        1056    9823456718    9849644291
        1069    9800776698    9860033237      65456
        1075    9875675534    9840098585
        1078    9809887666
        1089    9878900760                    65454
        1093    9887653700
        1096    9868890665    9841724570      65477

HOUSE_NUMBER PHONE_NUMBER1 PHONE_NUMBER2 FAX_NUMBER
------------ ------------- ------------- ----------
        1098    9851078766                    45376
        2020    9807610886                    23547
        3078    9825987059    9841808752      54342
        4099    9843819860    9860848077
        5081    9841709854
        6060    9813126745    9848819580
        7071    9846871009                    78564

18 rows selected.
```

Figure 52: Student Contact Info Table

4.2.16. Inserting Values in Student Address

```
INSERT ALL

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A71', 'Nepal', 'Bagmati', 'Kathmandu', 'Kuleshwor', 1056)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A72', 'Nepal', 'Lumbini', 'Butwal', 'Buddhanagar', 2020)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A73', 'Nepal', 'Province 2', 'Birgunj', 'Ghantaghar', 3078)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A74', 'Nepal', 'Bagmati', 'Kathmandu', 'Balkhu', 1069)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A75', 'Nepal', 'Bagmati', 'Kathmandu', 'Dllibazar', 1021)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A76', 'Nepal', 'Gandaki', 'Pokhara', 'Nayabazar', 4099)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A77', 'Nepal', 'Bagmati', 'Kathmandu', 'New Road', 1078)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A78', 'Nepal', 'Province 2', 'Nawalparasi', 'Parasi', 5081)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A79', 'Nepal', 'Bagmati', 'Kathmandu', 'Durbarmarg', 1021)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A710', 'Nepal', 'Gandaki', 'Gorkha', 'Khaireni', 6060)

SELECT * FROM dual;
```

```
SQL> INSERT ALL
  2  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A71', 'Nepal', 'Bagmati', 'Kathma
ndu', 'Kuleshwor', 1056)
  3  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A72', 'Nepal', 'Lumbini', 'Butwal
', 'Buddhanagar', 2020)
  4  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A73', 'Nepal', 'Province 2', 'Bir
gunj', 'Ghantaghar', 3078)
  5  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A74', 'Nepal', 'Bagmati', 'Kathma
ndu', 'Balkhu', 1069)
  6  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A75', 'Nepal', 'Bagmati', 'Kathma
ndu', 'Dllibazar', 1021)
  7  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A76', 'Nepal', 'Gandaki', 'Pokhar
a', 'Nayabazar', 4099)
  8  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A77', 'Nepal', 'Bagmati', 'Kathma
ndu', 'New Road', 1078)
  9  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A78', 'Nepal', 'Province 2', 'Naw
alparasi', 'Parasi', 5081)
 10  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A79', 'Nepal', 'Bagmati', 'Kathma
ndu', 'Durbarmarg', 1021)
 11  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A710', 'Nepal', 'Gandaki', 'Gorkh
a', 'Khaireni', 6060)
 12  SELECT * FROM dual;

10 rows created.
```

Figure 53: Inserting Values in Student Address1

INSERT ALL

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A711', 'Nepal', 'Bagmati', 'Kathmandu', 'Thapathali', 1098)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A712', 'Nepal', 'Bagmati', 'Kathmandu', 'Thamel', 1010)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A713', 'Nepal', 'Bagmati', 'Kathmandu', 'Putalisadak', 1093)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A714', 'Nepal', 'Bagmati', 'Kathmandu', 'Baneshwor', 1075)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A715', 'Nepal', 'Bagmati', 'Kathmandu', 'Tinkune', 1042)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A716', 'Nepal', 'Bagmati', 'Kathmandu', 'Asan', 1096)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A717', 'Nepal', 'Bagmati', 'Kathmandu', 'Koteshwor', 1089)

INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A718', 'Nepal', 'Province 1', 'Biratnagar', 'Buspark', 7071)

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A711', 'Nepal', 'Bagmati', 'Kathm
andu', 'Thapathali', 1098)
  3  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A712', 'Nepal', 'Bagmati', 'Kathm
andu', 'Thamel', 1010)
  4  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A713', 'Nepal', 'Bagmati', 'Kathm
andu', 'Putalisadak', 1093)
  5  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A714', 'Nepal', 'Bagmati', 'Kathm
andu', 'Baneshwor', 1075)
  6  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A715', 'Nepal', 'Bagmati', 'Kathm
andu', 'Tinkune', 1042)
  7  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A716', 'Nepal', 'Bagmati', 'Kathm
andu', 'Asan', 1096)
  8  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A717', 'Nepal', 'Bagmati', 'Kathm
andu', 'Koteshwor', 1089)
  9  INTO Student_Address(Student_Address_ID, Country, Province, City, Street, House_number) VALUES ('A718', 'Nepal', 'Province 1', 'Bi
ratnagar', 'Buspark', 7071)
 10  SELECT * FROM dual;

8 rows created.
```

Figure 54: Inserting Values in Student Address2

SELECT * FROM Student_Address;

```
SQL> SELECT * FROM Student_Address;

STUDENT_ADDRESS_ID        COUNTRY              PROVINCE             CITY                 STREET               HOUSE_NUMBER
------------------------- -------------------- -------------------- -------------------- -------------------- ------------
A71                       Nepal                Bagmati              Kathmandu            Kuleshwor                    1056
A72                       Nepal                Lumbini              Butwal               Buddhanagar                  2020
A73                       Nepal                Province 2           Birgunj              Ghantaghar                   3078
A74                       Nepal                Bagmati              Kathmandu            Balkhu                       1069
A75                       Nepal                Bagmati              Kathmandu            Dllibazar                    1021
A76                       Nepal                Gandaki              Pokhara              Nayabazar                    4099
A77                       Nepal                Bagmati              Kathmandu            New Road                     1078
A78                       Nepal                Province 2           Nawalparasi          Parasi                       5081
A79                       Nepal                Bagmati              Kathmandu            Durbarmarg                   1021
A710                      Nepal                Gandaki              Gorkha               Khaireni                     6060
A711                      Nepal                Bagmati              Kathmandu            Thapathali                   1098

STUDENT_ADDRESS_ID        COUNTRY              PROVINCE             CITY                 STREET               HOUSE_NUMBER
------------------------- -------------------- -------------------- -------------------- -------------------- ------------
A712                      Nepal                Bagmati              Kathmandu            Thamel                       1010
A713                      Nepal                Bagmati              Kathmandu            Putalisadak                  1093
A714                      Nepal                Bagmati              Kathmandu            Baneshwor                    1075
A715                      Nepal                Bagmati              Kathmandu            Tinkune                      1042
A716                      Nepal                Bagmati              Kathmandu            Asan                         1096
A717                      Nepal                Bagmati              Kathmandu            Koteshwor                    1089
A718                      Nepal                Province 1           Biratnagar           Buspark                      7071

18 rows selected.
```

Figure 55: Student Address Table

4.2.17. Inserting Values in Student Address Type

```
INSERT ALL
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('GG6', 'A71', 'Permanent')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('SA6', 'A74', 'Temporary')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('SA6', 'A72', 'Permanent')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RS6', 'A75', 'Permanent')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AD6', 'A77', 'Temporary')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AD6', 'A73', 'Permanent')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('LK6', 'A79', 'Temporary')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('LK6', 'A76', 'Permanent')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('PP6', 'A711', 'Permanent')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RP6', 'A712', 'Temporary')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RP6', 'A78', 'Permanent')
INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('PG6', 'A713', 'Permanent')
```

INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AB6', 'A714', 'Temporary')

INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AB6', 'A710', 'Permanent')

INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RB6', 'A715', 'Temporary')

INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RB6', 'A718', 'Permanent')

INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AG6', 'A716', 'Permanent')

INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('SR6', 'A717', 'Permanent')

SELECT * FROM dual;

```
SQL> INSERT ALL
  2  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('GG6', 'A71', 'Permanent')
  3  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('SA6', 'A74', 'Temporary')
  4  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('SA6', 'A72', 'Permanent')
  5  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RS6', 'A75', 'Permanent')
  6  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AD6', 'A77', 'Temporary')
  7  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AD6', 'A73', 'Permanent')
  8  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('LK6', 'A79', 'Temporary')
  9  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('LK6', 'A76', 'Permanent')
 10  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('PP6', 'A711', 'Permanent')
 11  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RP6', 'A712', 'Temporary')
 12  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RP6', 'A78', 'Permanent')
 13  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('PG6', 'A713', 'Permanent')
 14  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AB6', 'A714', 'Temporary')
 15  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AB6', 'A710', 'Permanent')
 16  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RB6', 'A715', 'Temporary')
 17  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('RB6', 'A718', 'Permanent')
 18  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('AG6', 'A716', 'Permanent')
 19  INTO Student_Address_Type(Student_ID, Student_Address_ID, Address_Type) VALUES ('SR6', 'A717', 'Permanent')
 20  SELECT * FROM dual;

18 rows created.
```

Figure 56: Inserting Values in Student Address Type

SELCT * FROM Student_Address_Type;

```
SQL> SELECT * FROM Student_Address_Type;

STUDENT_ID      STUDENT_ADDRESS_ID        ADDRESS_TYPE
--------------- ------------------------- ---------------
GG6             A71                       Permanent
SA6             A74                       Temporary
SA6             A72                       Permanent
RS6             A75                       Permanent
AD6             A77                       Temporary
AD6             A73                       Permanent
LK6             A79                       Temporary
LK6             A76                       Permanent
PP6             A711                      Permanent
RP6             A712                      Temporary
RP6             A78                       Permanent

STUDENT_ID      STUDENT_ADDRESS_ID        ADDRESS_TYPE
--------------- ------------------------- ---------------
PG6             A713                      Permanent
AB6             A714                      Temporary
AB6             A710                      Permanent
RB6             A715                      Temporary
RB6             A718                      Permanent
AG6             A716                      Permanent
SR6             A717                      Permanent

18 rows selected.
```

Figure 57: Student Address Type Table

4.2.18. Commit

COMMIT;

```
SQL> COMMIT;

Commit complete.
```

Figure 58: Commit

5. Database Querying

5.1. Information Query

i. List all the students with all their addresses with their phone numbers.

To answer this query, we first have to join the Student table, Student Address Type table, Student Address and Student Contact Info using the "JOIN" command. Then we select the required rows from their respective tables i.e. Name from Student, Country, Province, City, Street, House Number from Student Address and Phone Number1 and Phone Number2 from Student Contact Info.

Following query is used to list all the students with their phone numbers and addresses.

```
SELECT s.Name, sa.Country, sa.Province, sa.City, sa.Street, sa.House_Number, sat. Address_Type, sci.Phone_Number1, sci.Phone_Number2

FROM Student s

JOIN Student_Address_Type sat

ON s.Student_ID = sat.Student_ID

JOIN Student_Address sa

ON sat.Student_Address_ID = sa.Student_Address_ID

JOIN Student_Contact_Info sci

ON sci.House_Number = sa.House_Number

ORDER BY Name;
```

```
SQL> SELECT s.Name, sa.Country, sa.Province, sa.City, sa.Street, sa.House_Number, sat. Address_Type, sci.Phone_Number1, sci.Phone_Numbe
r2
  2  FROM Student s
  3  JOIN Student_Address_Type sat
  4  ON s.Student_ID = sat.Student_ID
  5  JOIN Student_Address sa
  6  ON sat.Student_Address_ID = sa.Student_Address_ID
  7  JOIN Student_Contact_Info sci
  8  ON sci.House_Number = sa.House_Number
  9  ORDER BY Name;

NAME                 COUNTRY              PROVINCE             CITY                 STREET               HOUSE_NUMBER ADDRESS_TYPE    P
HONE_NUMBER1 PHONE_NUMBER2
-------------------- -------------------- -------------------- -------------------- -------------------- ------------ --------------- -
------------ -------------
Aditi Dhakal         Nepal                Province 2           Birgunj              Ghantaghar                   3078 Permanent
         9825987059    9841808752
Aditi Dhakal         Nepal                Bagmati              Kathmandu            New Road                     1078 Temporary
         9809887666
Alisha Basnet        Nepal                Gandaki              Gorkha               Khaireni                     6060 Permanent
         9813126745    9848819580
Alisha Basnet        Nepal                Bagmati              Kathmandu            Baneshwor                    1075 Temporary
         9875675534    9840098585
Ashish Gautam        Nepal                Bagmati              Kathmandu            Asan                         1096 Permanent
         9868890665    9841724570
Grisha Giri          Nepal                Bagmati              Kathmandu            Kuleshwor                    1056 Permanent
         9823456718    9849644291
Labbi Karmasta       Nepal                Bagmati              Kathmandu            Durbarmarg                   1021 Temporary
         9873777779    9818248640
Labbi Karmasta       Nepal                Gandaki              Pokhara              Nayabazar                    4099 Permanent
         9843819860    9860848077
Pragati Gautam       Nepal                Bagmati              Kathmandu            Putalisadak                  1093 Permanent
         9887653700
Prince Panjiyar      Nepal                Bagmati              Kathmandu            Thapathali                   1098 Permanent
         9851078766
Rohit Parajuli       Nepal                Bagmati              Kathmandu            Thamel                       1010 Temporary
         9808761098    9865077564

NAME                 COUNTRY              PROVINCE             CITY                 STREET               HOUSE_NUMBER ADDRESS_TYPE    P
HONE_NUMBER1 PHONE_NUMBER2
-------------------- -------------------- -------------------- -------------------- -------------------- ------------ --------------- -
------------ -------------
Rohit Parajuli       Nepal                Province 2           Nawalparasi          Parasi                       5081 Permanent
         9841709854
Rojan Shrestha       Nepal                Bagmati              Kathmandu            Dllibazar                    1021 Permanent
         9873777779    9818248640
Roman Bhandari       Nepal                Bagmati              Kathmandu            Tinkune                      1042 Temporary
         9808980780
Roman Bhandari       Nepal                Province 1           Biratnagar           Buspark                      7071 Permanent
         9846871009
Simran Ranjit        Nepal                Bagmati              Kathmandu            Koteshwor                    1089 Permanent
         9878900760
Srijan Adhikari      Nepal                Bagmati              Kathmandu            Balkhu                       1069 Temporary
         9800776698    9860033237
Srijan Adhikari      Nepal                Lumbini              Butwal               Buddhanagar                  2020 Permanent
         9807610886

18 rows selected.
```

Figure 59: Information Query1

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

To answer this query, we have to join the Instructor Info and Module table. Then, we have to select module id from instructor info, module name from module and count the instructor id from instructor info table. Then we have to group by module id and module name having count of module id more than 1.

The command to list all the modules taught by more than one instructor is:

```
SELECT ii.Module_ID,m.Module_Name,Count(ii.Instructor_ID) AS "No of Instructor"
FROM Instructor_Info ii
JOIN Module m
ON m.Module_ID = ii.Module_ID
GROUP BY ii.Module_ID,m.Module_Name
HAVING COUNT(ii.Module_ID)>1;
```

```
SQL> SELECT ii.Module_ID,m.Module_Name,Count(ii.Instructor_ID) AS "No of Instructor"
  2  FROM Instructor_Info ii
  3  JOIN Module m
  4  ON m.Module_ID = ii.Module_ID
  5  GROUP BY ii.Module_ID,m.Module_Name
  6  HAVING COUNT(ii.Module_ID)>1;

MODULE_ID  MODULE_NAME          No of Instructor
---------- -------------------- ---------------
SCM3       Cyber Security Mgmt                2
ES3        Economics                          2
D3         Database                           2
SM3        Service Marketing                  2
```

Figure 60: Information Query 2

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

To answer this query, we first join the Instructor and Position table. Then, we select Name from the instructor table and the salary from the position table. We select data where the name contains a letter ‘s’ and salary is above 50,000.

The command for this query is:

```
SELECT i.Name, p.Salary

FROM Instructor i

JOIN Position p

ON  i.Instructor_Position = p.Instructor_Position

WHERE LOWER(i.Name) LIKE '%s%'

AND p.Salary > 50000;
```

```
SQL> SELECT i.Name, p.Salary
  2  FROM Instructor i
  3  JOIN Position p
  4  ON  i.Instructor_Position = p.Instructor_Position
  5  WHERE LOWER(i.Name) LIKE '%s%'
  6  AND p.Salary > 50000;

NAME                    SALARY
-------------------- ----------
Yurisha Banjade          93500
Umesh Nepal              93500
Saroj Yadav              91000
Biwash Adhikari          95000
Supriya Shrestha         90500
Sonique Adhikari         95000
Surendra KC              93500
Sangita Poudyal          90500

8 rows selected.
```

Figure 61: Information Query 3

iv. <u>List the modules comes under the 'Multimedia' specification.</u>

Tables i.e. module, module info and specification are joined. And then, specification name from specification and module name from module is selected. Only modules which came under Multimedia were selected using "WHERE".

The command to list module that come under "Multimedia" is:

```
SELECT s.Specification_Name, m.Module_Name

FROM Module m

JOIN Module_Info mi

ON m.Module_ID = mi.Module_ID

JOIN Specification s

ON mi.Specification_ID = s.Specification_ID

WHERE Specification_Name = 'Multimedia';
```

```
SQL> SELECT s.Specification_Name, m.Module_Name
  2  FROM Module m
  3  JOIN Module_Info mi
  4  ON m.Module_ID = mi.Module_ID
  5  JOIN Specification s
  6  ON mi.Specification_ID = s.Specification_ID
  7  WHERE Specification_Name = 'Multimedia';

SPECIFICATION_NAME   MODULE_NAME
-------------------- --------------------
Multimedia           3D Modelling
```

Figure 62: Information Query 4

v. <u>List the name of the head of modules with the list of his phone number.</u>

Here, we have to join Instructor, Instructor Address type, Instructor Address and Instructor Contact Info. Then, Name is selected from instructor, Address Type from Instructor Address Type and Phone number1, Phone number2 from Instructor Contact Info. Using “WHERE”, we select the data of the instructor who is the head of modules. We use “ORDER BY” to order the name by alphabetical order.

The command to answer this query is:

```
SELECT i.Name, iat.Address_Type, ici.Phone_Number1, ici.Phone_Number2
FROM Instructor i
JOIN Instructor_Address_Type iat
ON i.Instructor_ID = iat.Instructor_ID
JOIN Instructor_Address ia
ON iat.Instructor_Address_ID = ia.Instructor_Address_ID
JOIN Instructor_Contact_Info ici
ON ici.House_Number = ia.House_Number
WHERE Instructor_Position = 'Module Leader'
ORDER BY i.Name;
```

```
SQL> SELECT i.Name, iat.Address_Type, ici.Phone_Number1, ici.Phone_Number2
  2  FROM Instructor i
  3  JOIN Instructor_Address_Type iat
  4  ON i.Instructor_ID = iat.Instructor_ID
  5  JOIN Instructor_Address ia
  6  ON iat.Instructor_Address_ID = ia.Instructor_Address_ID
  7  JOIN Instructor_Contact_Info ici
  8  ON ici.House_Number = ia.House_Number
  9  WHERE Instructor_Position = 'Module Leader'
 10  ORDER BY i.Name;

NAME                 ADDRESS_TYPE    PHONE_NUMBER1 PHONE_NUMBER2
-------------------- --------------- ------------- -------------
Rohan Khatiwada      Temporary          9876543210
Rohan Khatiwada      Permanent          9876543336    9803758269
Surendra KC          Temporary          9876558643
Surendra KC          Permanent          9876541167    9878262620
Umesh Nepal          Permanent          9812345678    9808734828
Umesh Nepal          Temporary          9863468368
Yurisha Banjade      Permanent          9808639273    9897564827
Yurisha Banjade      Temporary          9875643210    9808765432

8 rows selected.
```

Figure 63: Information Query 5

vi. List all Students who have enrolled in 'networking' specifications.

Here, we join student, marks and specification tables. We select the Name attribute from Student and Specification from Specification. We use "WHERE" to select data where the specification name is Networking.

The command to list all students enrolled in 'networking' is:

```
SELECT s.Name, sp.Specification_name

FROM Student s

JOIN Marks m

ON m.Student_ID = s.Student_ID

JOIN Specification sp

ON sp.Specification_ID = m.Specification_ID

WHERE sp.Specification_Name = 'Networking';
```

```
SQL> SELECT s.Name, sp.Specification_name
  2  FROM Student s
  3  JOIN Marks m
  4  ON m.Student_ID = s.Student_ID
  5  JOIN Specification sp
  6  ON sp.Specification_ID = m.Specification_ID
  7  WHERE sp.Specification_Name = 'Networking';

NAME                 SPECIFICATION_NAME
-------------------- --------------------
Srijan Adhikari      Networking
Simran Ranjit        Networking
```

Figure 64: Information Query 6

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

To answer this query, we join Instructor Contact Info, Instructor Address, Instructor Address type, instructor, instructor info and module. Then we select Name from Instructor, Address Type from Instructor Address type and Fax Number from Instructor Contact info. Then, to select the instructor who teaches the database module, we use the "WHERE" clause. "ORDER BY" is used to order names by alphabetical order.

```
SELECT i.Name, iat.Address_Type, ici.Fax_Number

FROM Instructor_Contact_Info ici

JOIN Instructor_Address ia

ON ici.House_Number = ia.House_Number

JOIN Instructor_Address_Type iat

ON iat.Instructor_Address_ID = ia.Instructor_Address_ID

JOIN Instructor i

ON i.Instructor_ID = iat.Instructor_ID

JOIN Instructor_Info ii

ON i.Instructor_ID =  ii.Instructor_ID

JOIN Module m

ON m.Module_ID =  ii.Module_ID

WHERE m.Module_Name = 'Database'

ORDER BY i.Name;
```

```
SQL> SELECT i.Name, iat.Address_Type, ici.Fax_Number
  2  FROM Instructor_Contact_Info ici
  3  JOIN Instructor_Address ia
  4  ON ici.House_Number = ia.House_Number
  5  JOIN Instructor_Address_Type iat
  6  ON iat.Instructor_Address_ID = ia.Instructor_Address_ID
  7  JOIN Instructor i
  8  ON i.Instructor_ID = iat.Instructor_ID
  9  JOIN Instructor_Info ii
 10  ON i.Instructor_ID =  ii.Instructor_ID
 11  JOIN Module m
 12  ON m.Module_ID =  ii.Module_ID
 13  WHERE m.Module_Name = 'Database'
 14  ORDER BY i.Name;

NAME                 ADDRESS_TYPE    FAX_NUMBER
-------------------- -------------- ----------
Biwash Adhikari      Temporary          779765
Biwash Adhikari      Permanent          786553
Yurisha Banjade      Permanent
Yurisha Banjade      Temporary
```

Figure 65: Information Query 7

viii. List the specification falls under the BIT course.

For this query, Specification table, specification info and course table are joined. Specification name from specification and course name from course are retrieved. "WHERE" is used to select the specification which only falls under the 'B.Sc.IT' course. "ORDER BY" is used to manage the specification name by alphabetical order.

The command to list the specification falling under 'B.Sc.IT' course is:

```
SELECT s.Specification_Name, c.Course_Name

FROM Specification s

JOIN Specification_Info si

ON s.Specification_ID = si.Specification_ID

JOIN Course c

ON si.Course_ID = c.Course_ID

WHERE c.Course_Name = 'B.Sc.IT'

ORDER BY s.Specification_Name;
```

```
SQL> SELECT s.Specification_Name, c.Course_Name
  2  FROM Specification s
  3  JOIN Specification_Info si
  4  ON s.Specification_ID = si.Specification_ID
  5  JOIN Course c
  6  ON si.Course_ID = c.Course_ID
  7  WHERE c.Course_Name = 'B.Sc.IT'
  8  ORDER BY s.Specification_Name;

SPECIFICATION_NAME   COURSE_NAME
-------------------- --------------------
Computing            B.Sc.IT
Multimedia           B.Sc.IT
Networking           B.Sc.IT
```

Figure 66: Information Query 8

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

Here, distinct module name and class ID is selected from the Module table. The module name and class ID is grouped using "GROUPED BY". Class ID having count one is selected and the class_id is ordered by using "ORDERED BY".

The command to list the module taught in any one particular class is:

```
SELECT DISTINCT(Module_Name), Class_ID

FROM Module

GROUP BY Module_Name, Class_ID

HAVING COUNT(Class_ID)=1

ORDER BY Class_ID;
```

```
SQL> SELECT DISTINCT(Module_Name), Class_ID
  2  FROM Module
  3  GROUP BY Module_Name, Class_ID
  4  HAVING COUNT(Class_ID)=1
  5  ORDER BY Class_ID;

MODULE_NAME          CLASS_ID
-------------------- ----------
3D Modelling         LT01
Database             LT01
Service Marketing    LT02
Accounting           LTO8
Cyber Security Mgmt  SR03
Economics            SR03
Ethical Hacking      TR07
Software Eng         TR12

8 rows selected.
```

Figure 67: Information Query 9

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

For this query, instructor, instructor address type and address table are joined using “JOIN”. Then, we select Name from Instructor, Country, Province, City, Street, and House Number from Instructor Address and Address Type from Instructor Address Type. Then “SUBSTR” and “INSTR” are used to split the name into first and last names. “WHERE” is used to select the data where the first name has ‘a’ at the end of it.

The command to answer this query is:

```
SELECT i.Name, ia.Country, ia.Province, ia.City, ia.Street, ia.House_Number, iat.Address_type

FROM Instructor i

JOIN Instructor_Address_Type iat

ON i.Instructor_ID = iat.Instructor_ID

JOIN Instructor_Address ia

ON iat.Instructor_Address_ID = ia.Instructor_Address_ID

WHERE SUBSTR(i.Name, 1, INSTR(i.Name, ' ') - 1) LIKE '%a';
```

```
SQL> SELECT i.Name, ia.Country, ia.Province, ia.City, ia.Street, ia.House_Number, iat.Address_type
  2  FROM Instructor i
  3  JOIN Instructor_Address_Type iat
  4  ON i.Instructor_ID = iat.Instructor_ID
  5  JOIN Instructor_Address ia
  6  ON iat.Instructor_Address_ID = ia.Instructor_Address_ID
  7  WHERE SUBSTR(i.Name, 1, INSTR(i.Name, ' ') - 1) LIKE '%a';

NAME                 COUNTRY              PROVINCE             CITY                 STREET               HOUSE_NUMBER ADDRESS_TYPE
-------------------- -------------------- -------------------- -------------------- -------------------- ------------ ---------------
Yurisha Banjade      Nepal                Bagmati              Kathmandu            Thamel                        100 Temporary
Yurisha Banjade      Nepal                Gandaki              Pokhara              Prithivi Chowk                240 Permanent
Supriya Shrestha     Nepal                Bagmati              Kathmandu            Baluwatar                     140 Permanent
Surendra KC          Nepal                Bagmati              Kathmandu            Kamalpokhari                  191 Temporary
Surendra KC          Nepal                Sudarpashim          Dhangadhi            Hasanpur                      998 Permanent
Sangita Poudyal      Nepal                Bagmati              Kathmandu            169                           169 Temporary
Sangita Poudyal      Nepal                Province 1           Illam                Chiya Bagan                   666 Permanent

7 rows selected.
```

Figure 68: Information Query 10

5.2. Transaction Query

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

Here, student, marks, specification and course are joined. To reduce the fees of students enrolled in computing, "CASE" is used. The reduced fees are named "Discounted Fees". Name from student, course name from course and fees from specification are selected.

The command to answer this transaction query is:

```
SELECT s.Name, c.Course_Name, sp.Fees,
CASE sp.Specification_Name WHEN 'Computing' THEN 0.9*sp.Fees
ELSE sp.Fees END "Discounted Fees"
FROM Student s
JOIN Marks m
ON s.Student_ID = m.Student_ID
JOIN Specification sp
ON sp.Specification_ID = m.Specification_ID
JOIN Course c
ON c.Course_ID = m.Course_ID;
```

```
SQL> SELECT s.Name, c.Course_Name, sp.Fees,
  2  CASE sp.Specification_Name WHEN 'Computing' THEN 0.9*sp.Fees
  3  ELSE sp.Fees END "Discounted Fees"
  4  FROM Student s
  5  JOIN Marks m
  6  ON s.Student_ID = m.Student_ID
  7  JOIN Specification sp
  8  ON sp.Specification_ID = m.Specification_ID
  9  JOIN Course c
 10  ON c.Course_ID = m.Course_ID;

NAME                 COURSE_NAME                FEES Discounted Fees
-------------------- -------------------- ---------- ---------------
Alisha Basnet        BBA                      105000          105000
Aditi Dhakal         B.Sc.IT                  120000          120000
Ashish Gautam        BBA                      100000          100000
Grisha Giri          B.Sc.IT                  115000          103500
Labbi Karmasta       B.Sc.IT                  115000          103500
Pragati Gautam       B.Sc.IT                  120000          120000
Prince Panjiyar      MBA                       95000           95000
Roman Bhandari       BBA                      103000          103000
Rohit Parajuli       M.Sc.IT                   90000           90000
Rojan Shrestha       BBA                      103000          103000
Srijan Adhikari      B.Sc.IT                  117000          117000

NAME                 COURSE_NAME                FEES Discounted Fees
-------------------- -------------------- ---------- ---------------
Simran Ranjit        B.Sc.IT                  117000          117000

12 rows selected.
```

Figure 69: Transaction Query 1

ii. 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.

Here, Phone Number list having null value i.e. Phone Number2 is selected from Student Contact Info and "NVL" is used to replace the null value with 1234567890 and is named "Contact Details"

The following command answers the query:

```
SELECT Phone_Number2,

NVL(Phone_Number2, 1234567890) AS "Contact Details"

FROM Student_Contact_info;
```

```
SQL> SELECT Phone_Number2,
  2  NVL(Phone_Number2, 1234567890) AS "Contact Details"
  3  FROM Student_Contact_info;

PHONE_NUMBER2 Contact Details
------------- ---------------
   9860469853      9860469853
   9865077564      9865077564
   9818248640      9818248640
                   1234567890
   9849644291      9849644291
   9860033237      9860033237
   9840098585      9840098585
                   1234567890
                   1234567890
                   1234567890
   9841724570      9841724570

PHONE_NUMBER2 Contact Details
------------- ---------------
                   1234567890
                   1234567890
   9841808752      9841808752
   9860848077      9860848077
                   1234567890
   9848819580      9848819580
                   1234567890

18 rows selected.
```

Figure 70: Transaction Query 2

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

Here, to find the number of weeks since the students enrolled in the course, SYSDATE is subtracted by the Enrollment Date and the difference is divided by 7. The column is named 'No, of weeks since Enrollment'. Name is also selected from Student.

The command of this query is:

```
SELECT Name, (SYSDATE-Enrollment_Date)/7 AS "No. of weeks since Enrollment"
FROM Student;
```

```
SQL> SELECT Name, (SYSDATE-Enrollment_Date)/7 AS "No. of weeks since Enrollment"
  2  FROM Student;

NAME                 No. of weeks since Enrollment
-------------------- -----------------------------
Grisha Giri                             96.5600579
Srijan Adhikari                         109.274344
Rojan Shrestha                          56.9886293
Aditi Dhakal                            95.9886293
Labbi Karmasta                           96.702915
Prince Panjiyar                         148.560058
Rohit Parajuli                          160.274344
Pragati Gautam                          160.131486
Alisha Basnet                           109.274344
Roman Bhandari                          148.702915
Ashish Gautam                           160.274344

NAME                 No. of weeks since Enrollment
-------------------- -----------------------------
Simran Ranjit                           160.131486

12 rows selected.
```

Figure 71: Transaction Query 3

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

The following command shows the name of instructor who got equal salary and work in same specification:

```
SELECT i1.Name, x1.Salary, x1.Specification_Name
    From (
SELECT * FROM (
SELECT i.Instructor_ID, p.Salary, s.Specification_Name
 FROM Instructor i
    JOIN Instructor_Info ii ON ii.Instructor_ID = i.Instructor_ID
    JOIN Specification s ON s.Specification_ID = ii.Specification_ID
    JOIN Position p ON p.Instructor_Position = i.Instructor_Position
GROUP BY s.Specification_Name, p.Salary, i.Instructor_ID) y1
    WHERE (
  SELECT COUNT(*) FROM (
    SELECT i.Instructor_ID, p.Salary, s.Specification_Name
 FROM Instructor i
    JOIN Instructor_Info ii ON ii.Instructor_ID = i.Instructor_ID
    JOIN Specification s ON s.Specification_ID = ii.Specification_ID
    JOIN Position p ON p.Instructor_Position = i.Instructor_Position
GROUP BY s.Specification_Name, p.Salary, i.Instructor_ID) y2
WHERE y1.Salary = y2.Salary
AND
```

y1.Specification_Name =y2.Specification_Name) >1

) x1

JOIN Instructor i1 on x1.Instructor_ID = i1.Instructor_ID

ORDER BY x1.Salary, x1.specification_Name, i1.Name;

```
SQL> SELECT i1.Name, x1.Salary, x1.Specification_Name
  2          From (
  3  SELECT * FROM (
  4  SELECT i.Instructor_ID, p.Salary, s.Specification_Name
  5   FROM Instructor i
  6         JOIN Instructor_Info ii ON ii.Instructor_ID = i.Instructor_ID
  7         JOIN Specification s ON s.Specification_ID = ii.Specification_ID
  8         JOIN Position p ON p.Instructor_Position = i.Instructor_Position
  9  GROUP BY s.Specification_Name, p.Salary, i.Instructor_ID) y1
 10         WHERE (
 11      SELECT COUNT(*) FROM (
 12          SELECT i.Instructor_ID, p.Salary, s.Specification_Name
 13   FROM Instructor i
 14         JOIN Instructor_Info ii ON ii.Instructor_ID = i.Instructor_ID
 15         JOIN Specification s ON s.Specification_ID = ii.Specification_ID
 16         JOIN Position p ON p.Instructor_Position = i.Instructor_Position
 17  GROUP BY s.Specification_Name, p.Salary, i.Instructor_ID) y2
 18  WHERE y1.Salary = y2.Salary
 19  AND
 20  y1.Specification_Name =y2.Specification_Name) >1
 21  ) x1
 22  JOIN Instructor i1 on x1.Instructor_ID = i1.Instructor_ID
 23  ORDER BY x1.Salary, x1.specification_Name, i1.Name;

NAME                    SALARY SPECIFICATION_NAME
-------------------- ---------- --------------------
Sangita Poudyal           90500 Computing
Supriya Shrestha          90500 Computing
Rohan Khatiwada           93500 Business
Surendra KC               93500 Business
```

Figure 72: Transaction query 4

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

Here, course and marks are joined. Course name from course is selected. The total students are counted using “COUNT” and maximum marks are calculated using “MAX”. The data is grouped using “GROUP BY” on Course Name.

The following command answer the query:

```
SELECT c.Course_Name, COUNT(m.Student_ID) AS "Total Students", MAX(m.Marks) AS "Highest Marks"
FROM Course c
JOIN Marks m
ON c.Course_ID = m.Course_ID
GROUP BY Course_Name;
```

```
SQL> SELECT c.Course_Name, COUNT(m.Student_ID) AS "Total Students", MAX(m.Marks) AS "Highest Marks"
  2  FROM Course c
  3  JOIN Marks m
  4  ON c.Course_ID = m.Course_ID
  5  GROUP BY Course_Name;

COURSE_NAME          Total Students Highest Marks
-------------------- -------------- -------------
MBA                               1            69
M.Sc.IT                           1            80
B.Sc.IT                           6            90
BBA                               4            95
```

Figure 73: Transaction Query 5

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

Name and instructor position is selected from the Instructor table. To select the name of instructors who are the only course leader, “WHERE” is used.

The following command lists the instructor who are also a course leader:

```
SELECT Name, Instructor_Position

FROM Instructor

WHERE Instructor_Position = 'Course Leader';
```

```
SQL> SELECT Name, Instructor_Position
  2  FROM Instructor
  3  WHERE Instructor_Position = 'Course Leader';

NAME                 INSTRUCTOR_POSITION
-------------------- --------------------
Biwash Adhikari      Course Leader
Sonique Adhikari     Course Leader
Raju Gaire           Course Leader
```

Figure 74: Transaction Query 6

6. Creation of Dump File

A database dump is a major output of data that can help users to either back up or duplicate a database. This can be considered part of the more general term data dump, which involves revealing a set of stored data from a given technology. (Techopedia, 2020)

A dump file can be imported to any other user's database by using various commands. This helps to transport the data to other users easily.

To create a dump file, we first open the command prompt and enter the path where the dump file will be stored. Then, we use 'exp' to export the data into the dump file in the provided location. The command is shown below:

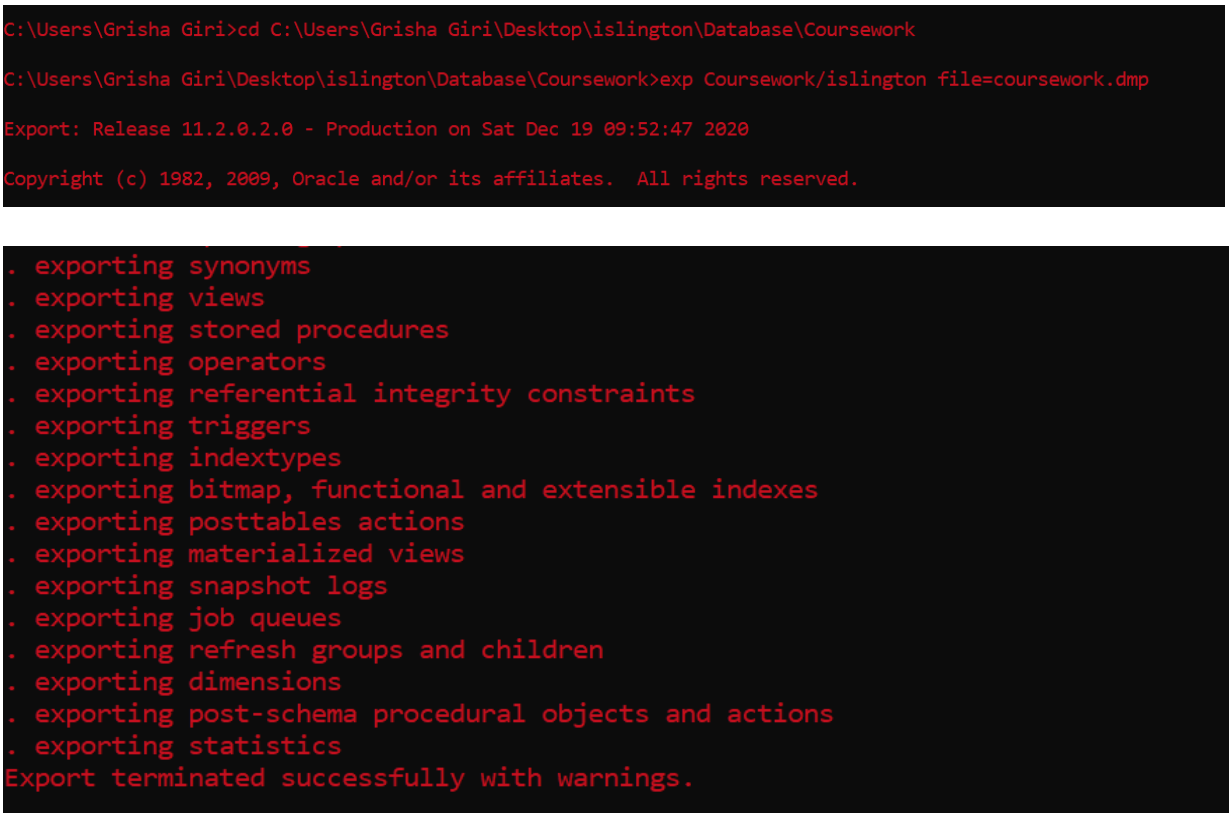

Figure 75: Creating Dump file.

If the creation of the dump file is successful, then 'Export terminated successfully with warnings' is shown at the end. Now that the dump file is created, we can now import the data of the dump file using 'imp'.

7. Drop Table

All the created tables are dropped using “DROP TABLE”. Following commands were used to drop the tables:

- DROP TABLE Student_Address_Type;
- DROP TABLE Student_Address;
- DROP TABLE Student_Contact_Info;
- DROP TABLE Marks;
- DROP TABLE Student;
- DROP TABLE Instructor_Address_Type;
- DROP TABLE Instructor_Address;
- DROP TABLE Instructor_Contact_Info;
- DROP TABLE Instructor_Info;
- DROP TABLE Instructor;
- DROP TABLE Position;
- DROP TABLE Module_Info;
- DROP TABLE Module;
- DROP TABLE Class;
- DROP TABLE Class;
- DROP TABLE Specification_info;
- DROP TABLE Course;

```
SQL> DROP TABLE Student_Address_Type;

Table dropped.

SQL> DROP TABLE Student_Address;

Table dropped.

SQL> DROP TABLE Student_Contact_Info;

Table dropped.

SQL> DROP TABLE Marks;

Table dropped.

SQL> DROP TABLE Student;

Table dropped.

SQL> DROP TABLE Instructor_Address_Type;

Table dropped.

SQL> DROP TABLE Instructor_Address;

Table dropped.
```

```
SQL> DROP TABLE Instructor_Contact_Info;

Table dropped.

SQL> DROP TABLE Instructor_Info;

Table dropped.

SQL> DROP TABLE Instructor;

Table dropped.
```

```
SQL> DROP TABLE Position;

Table dropped.

SQL> DROP TABLE Module_Info;

Table dropped.

SQL> DROP TABLE Module;

Table dropped.

SQL> DROP TABLE Class;

Table dropped.
```

```
SQL> DROP TABLE Specification_info;

Table dropped.

SQL> DROP TABLE Specification;

Table dropped.

SQL> DROP TABLE Course;

Table dropped.
```

Figure 76: Drop Tables

8. Conclusion

The coursework was related to making a college record system. Normalization was done and the database was prepared in Oracle SQL Plus. The coursework has been more about learning myself more than completing it. During the coursework, I've learned about many things about normalization and the steps to complete it. The learning experience has been really fun and helpful for future purposes. Whenever I had any problem, I researched on many websites learning new things every time. The research process might have been hard but it was a really fun process. The teachers and my friends were really helpful whenever I was struck in a problem with my coursework.

The main objective of the coursework was to prepare entities and attributes for a college record system, which was Islington College in this case and normalize the data to remove redundancies and anomalies and prepare a database of the normalized data. Various steps were carried out in the normalization process. The initial ERD had four entities but after normalization there were 17 entities with both main entities and bridging entities. The bridging entities connected two or more main entities avoiding making any many-to-many relationships which helped in data insertion or deletion later in the database. The database was created in SQL Plus. Various commands were used. There were transaction and information queries given in the coursework to be solved which was also done. I learned more about normalization than I used to before this coursework. Learning more about normalization was very fun as it is a fun topic to even research. Creating the correct UNF took some time and there were a lot of ups and downs during the process but I eventually got to the end point of this. At first, knowing what was functional dependency or transitive dependency was very hard. But with practice, learning about dependencies was very easy.

There were a lot of things this coursework taught me. This coursework taught me the importance of researching and the importance of normalization in databases. Moreover, it also made me discover new queries and commands to use. This coursework taught me how to make ERDs which can be applicable in Software Engineering for the construction of ERD in that coursework. Various concepts of database module can be used in Software Engineering. Overall, later in life we may have to manage a lot of data. At the end, this coursework taught me how to organize and store data with less redundancies.

Bibliography

GeeksForGeeks. 2019. GeeksForGeeks. [Online] September 25, 2019. [Cited: December 19, 2020.] https://www.geeksforgeeks.org/denormalization-in-databases/.

T, Neha. 2019. BinaryTerms. [Online] December 10, 2019. [Cited: December 13, 2020.]

Techopedia. 2020. Techopedia. [Online] 2020. [Cited: December 15, 2020.] https://www.techopedia.com/definition/1221/normalization.

—. 2020. Techopedia. [Online] 2020. [Cited: December 19, 2020.] https://www.techopedia.com/definition/23340/database-dump.

TutorialsPoint. 2020. TutorialsPoint. [Online] 2020. [Cited: December 19, 2020.] https://www.tutorialspoint.com/sql/sql-transactions.htm.