LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

CC5051NI – DATABASES SYSTEMS

50% Individual Coursework

2020-21 Autumn

Student Name: Siddhartha Ghimire

London Met ID: 19031691

College ID: NP01CP4A190148

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.

Table of Contents

Table of Figures

Chapter 1- Introduction

1.1. Introduction of the college

Islington College is located at Kamal Pokhari, Kathmandu. It was established in the year 1996 A.D. in collaboration with London Metropolitan University, UK. The College offers Bachelor's and Master's programs –BBA, MBA and IT (Islington, 2018). Islington conduct different types of extra-curricular activities per year like Autumn AI Competition, Online Mentorship Session, Hult Prize etc. Islington College aims is to provide the students with high-end state of the art IT and Business related education in the country (Islington, 2018).

1.2. Current Business Activities and Operations

Islington College which is also known for one of the best IT college in Nepal provides satisfaction to the students with the following current business activities:

i.) The College provides the students with different Bachelor's and Master's courses like MBA, BBA, IT.

ii.) Bachelor's course contains different specification like Computing, Networking, Multimedia, Marketing.

iii.) Each specification contains several modules like Database, Programming, etc.

iv.) Islington college has different type of classes with their respective code where each modules of the course are taught in the particular class.

v.) Islington College has hired the best instructors to guide the students in their respective course theoretically and practically as per their needs.

vi.) There are many instructors who are associated in different module. For each module there is a module leader.

vii.) A student can enroll for any one Bachelor's or Master's course which are specified.

viii.) The college provides identity card to the students and instructors so that their records are handled properly.

1.3. Business Rules

i.) Islington College keep track the address of all the student and teachers including the mailing address.

ii.) Address contains the name of country, province, city, street, house number, phone numbers, fax number where fax number is not mandatory to be provided by the students and instructors.

iii.) Islington College contains different courses where each course can offer any number of specifications and each specification contains several modules.

iv.) One module can be repeated under different specifications.

v.) Many teachers can be associated in a module, but a teacher can associated only in one module.

vi.) For each module, there is a module leader, and an instructor can be a leader of only one module.

vii.) Each instructor can teach any one or many modules at a time, and a module can be taught by many instructors.

viii.) A student should enroll in only one course and each course can have any number of students.

ix.) Each course contains different specification and so specification contains different fee structures.

x.) Each module is taught in any given particular class, but in each class a number of modules are taught.

1.4. Identification of Entities and Attributes

A single unique object in the real world that is being mastered which can be either a person, organization, object type or concept about which information is stored is known as Entities (IBM Knowledge, 2020).

A characteristic or trait of an entity type that describes the entity is known as Attributes (IBM Knowledge, 2020). For example, the student entity type has the student name attribute.

Entities	Attributes
Courses	Course ID(PK), Course Name, Course Type, Specification Name, Specification Fee
Teachers	Teachers ID(PK), Course ID(FK), Name, Sex, Joined Date, Teachers Type, Teacher Status, Salary
Students	Students ID(PK), Course ID(FK), Name, Sex, Joined Date, Student Status, Year
Modules	Module ID(PK), Course ID(FK), Module Name, Module Leader, Marks, Class
Address Student	AddressStd ID(PK), Student ID(FK), Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax No, Email
Address Teacher	AddressTec ID(PK), Teacher ID(FK), Country ID, Country, Province City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax No, Email

Table 1 Identification of Entities and Attributes

1.5. Initial ER Diagram

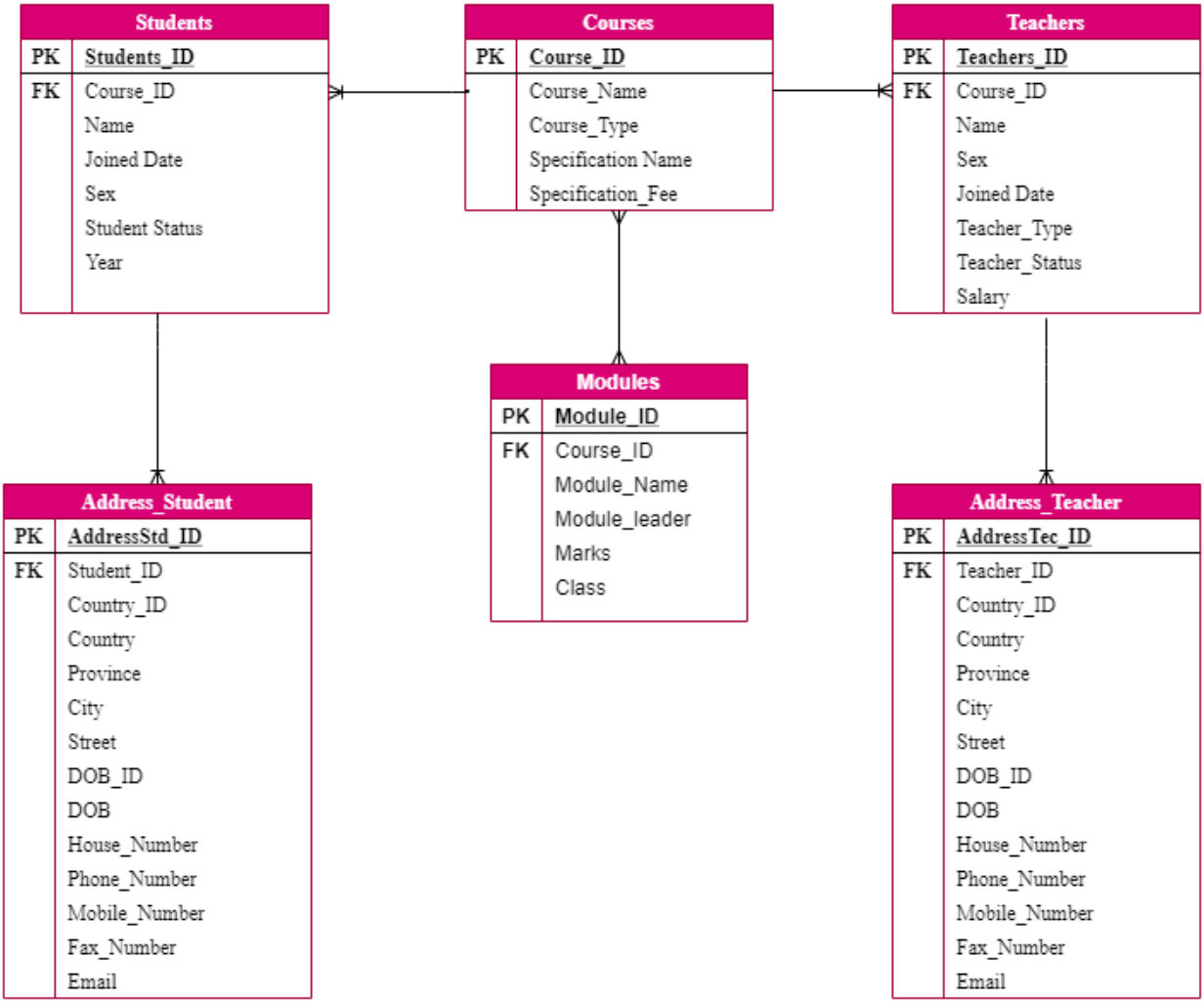


Figure 1 Initial ERD

The above ER Diagram is made gathering the information provided from the business rule. Given ER Diagram is not well managed and the relationship between the entities is also not properly mentioned. To solve all those problems arrived in the above ER Diagram, Normalization must be implemented, which provides a means to analyze business requirements so as to standardize organizational vocabulary, enforce business rules, and ensure adequate data quality (DATAVERSITY Education, 2020).

Chapter 2 – Normalization

2.1. Assumption

The assumptions made to justify the ER Diagram after Normalization are:

i.) Every Course has a Course ID, Course Name, Course Type, Specification Name and Specification Fees.

ii.) Every Student has a Student ID, Name, Sex, Joined Date, Student Status and Year.

iii.) Every Teacher has a Teacher ID, Name, Sex, Joined Date, Teacher Status, Teacher Type and Salary.

iv.) Every Module has a Module ID and Class.

v.) Every Student Address has AddressStd ID, Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number and Email

vi.) Every Teacher Address has AddressTec ID, Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number and Email

vii.) A specification has many modules and a module is taught by many teachers.

2.2. Normalization

Normalization is one of the key tenets in a relational model design. It is the process of removing redundancy in a table so that the table is easier to modify (Bala & Martin, 1997; Codd, 1970; Date, 2003; Mannino, 2006; Rob & Coronel, 2006). It usually takes part in dividing an entity table into one or more tables and creating relationships between the tables. The objective is to isolate data so that addition, deletion, and modification of an attribute can be made in just one table and then propagated through the rest of the database via the defined relationship (DATAVERSITY Education, 2020).

2.2.1. UNF (Un-Normalized Form)

Scenario for UNF:

i.) Each Course should provide the Course Name, Specification Name, Course Type and Specification fee.

ii.) A Course contains many teachers, students and modules.

iii.) Each student address and teacher address should provide detailed address and contact information of student and teacher: Mobile Number, Phone Number, Fax Number, Email, House Number, Street, Country, Province.

iv.) A student should provide his/her Name, Sex, Joined Date and Status.

v.) A teacher should provide his/her Name, Sex, Joined Date, Status, Type and Salary.

vi.) A student can have any course with any specification.

vii.) A teacher can teach any course with any specification.

viii.) A Student or Teacher can have multiple modules and a module are taught by multiple teachers.

ix.) Each Module has a module leader and module leader should be associated from a Teacher.

x.) Each Date of Birth of a student and a teacher will have a Date of Birth ID.

xi.) Each Country of a student and teacher will have a Country ID.

Repeating Groups:

Course(Course ID(PK), Course Name, Specification Name, Specification fee, Course Type, {Student ID(PK), Name, Joined Date, Sex, Student Status, Year, {AddressStd ID, Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email}},{Teachers ID(PK), Name, Joined Date, Sex, Teacher Type, Teacher Status, Salary, {AddressTec ID, Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email}}, {Module ID, Module Name, Module Leader, Marks, Class})

2.2.2. 1NF (First Normal Form)

The first normal form rule is that there shouldn't be any nesting or repeating groups in a table. The entity type that contains only one value for an attribute in an entity instance which ensures the application of first normal form for the entity type is known as first normal form (DATAVERSITY Education, 2020).

Scenario for 1NF:

i.) From the Course ID we can identify every detail of teachers, students and modules.

ii.) From the Student ID and Teachers ID we can identify every details of Student and Teachers Address

iii.) Those repeating groups from the UNF will be separated with different entities.

iv.) So formed entities will contains a different primary key and since Course ID is related to teacher, student and modules entity so it is added as foreign key and formed a composite primary key.

v.) Student ID and Teachers ID is related to Student Address and Teacher Address entity so it is added as foreign key and formed a composite primary key.

Entities:

Course-1 (Course ID(PK), Course Name, Specification Name, Specification fee, Course Type)

Student-1(Student ID(PK), Course ID(FK), Name, Joined Date, Sex, Student Status, Year)

Teachers-1(Teachers ID(PK), Course ID(FK), Name, Joined Date, Sex, Teacher Type,

Teacher Status, Salary)

Module-1 (Module ID(PK), Course ID(FK), Module Name, Module Leader, Marks, Class)

sAddress-1 (AddressStd ID(PK), Student ID(FK), Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email)

tAddress-1 (AddressTec ID(PK), Teachers ID(FK), Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email)

2.2.3. 2NF (Second Normal Form)

The second normal form rule is that the key attributes determine all non-key attributes. The second normal form deals the situation when the entity identifier contains two or more attributes, and the non-key attribute depends upon the part of the entity identifier (DATAVERSITY Education, 2020).

Scenario for 2NF:

i.) After the composite key which has been determined in above 1NF, will be checked whether its attributes functionally dependent to the given composite key of that entity or not which is also known as partial or full functional dependency.

ii.) The attributes formed from the 1NF will further separated into new entities depending upon the partial or full functional dependency

Showing Partial Dependency:

For Student Info:

i.) Student ID determines the Name, Sex, Joined Date, Student Status and Year.

ii.) Composite Primary Key Student ID, Course ID doesn't determine any of the student attributes

Student ID => Name, Sex, Joined Date, Student Status, Year

Student ID, Course ID =>

For Teacher Info:

i.) Teacher ID determines the Name, Sex, Joined Date, Teacher Type, Teacher Status and Salary.

ii.) Composite Primary Key Teacher ID, Course ID doesn't determine any of the teacher attributes.

Teacher ID => Name, Sex, Joined Date, Teacher Type, Teacher Status, Salary

Teacher ID, Course ID=>

For Module Details:

i.) Module ID determines the Module Leader and Class.

ii.) Composite Primary Key Module ID, Course ID determine the Module Name and Marks for each course.

Module ID => Class, Marks

Module ID, Course ID => Module Name, Module Leader

For sAddress Info:

i.) AddressStd ID determines the Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number and Email.

ii.) Composite Primary Key AddressStd ID, Student ID does not determine any of the sAddress attributes.

AddressStd ID => Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email

AddressStd ID, Student ID =>

For tAddress Info:

iii.) AddressTec ID determines the Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number and Email.

iv.) Composite Primary Key AddressTec ID, Teacher ID does not determine any of the tAddress attributes.

AddressTec ID => Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email

AddressTec ID, Teacher ID =>

Entities

Course-2 (Course ID(PK), Course Name, Specification Name, Specification fee, Course Type)

Student-2(Student ID(PK), Name, Sex, Joined Date, Student Status, Year)

Student Info-2(Student ID(FK), Course ID(FK))

Teacher-2(Teacher ID(PK), Name, Sex, Joined Date, Teacher Type, Teacher Status, Salary)

Teacher Info-2(Teacher ID(FK), Course ID(FK))

Module-2(Module ID(PK), Marks, Class)

Module details-2(Module ID(FK), Course ID(FK), Module Name, Module Leader)

sAddress-2(AddressStd ID(PK), Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email)

sAddress Info-2(AddressStd ID(FK), Student ID(FK))

tAddress-2(AddressTec ID(PK), Country ID, Country, Province, City, Street, DOB ID, DOB, House Number, Phone Number, Mobile Number, Fax Number, Email)

tAddress Info-2(AddressTec ID(FK), Teachers ID(FK))

2.2.4. 3NF (Third Normal Form)

The third normal form rule is that the non-key attributes should be independent. This normal form is violated when there exists a dependency among non-key attributes in the form of a transitive dependency (DATAVERSITY Education, 2020).

Scenario for 3NF:

i.) After the separation of an attributes based on Partial dependency in context of 2NF, the relationship between the values in the same table that cause a functional dependency which is also know as transitive dependency will be checked in 3NF.

ii.) The attributes formed from the 2NF will further separated into new entities depending upon the transitive dependency.

Showing Transitive Dependencies:

For sAddress:

i.) AddressStd ID determines the Date of Birth ID and Date of Birth ID determines the Date of Birth and Age of a student.

AddressStd ID -> DOB ID,

DOB ID -> DOB, Age

ii.) AddressStd ID determines the Country ID similarly Country ID determines the Country, Province, City, Street Number, House Number and House Number Provides the Phone Number and Fax Number of a student.

AddressStd ID -> Country ID,

Country ID-> Country, Province, city, Street No, House No

House No-> Phone No, Fax no

For tAddress:

i.) AddressTec ID determines the Date of Birth ID and Date of Birth ID determines the Date of Birth and Age of a teacher.

AddressTec ID -> DOB ID,

DOB ID-> DOB, Age

ii.) AddressTec ID determines the Country ID similarly Country ID determines the Country, Province, City, Street Number, House Number and House Number Provides the Phone Number and Fax Number of a teacher.

AddressTec ID -> Country ID,

Country ID-> Country, Province, city, Street No, House No

House No-> Phone No, Fax no

iii.) Similarly, Rest of the Entities does not have any Transitive Dependency

Entities:

Course-3 (Course ID(PK), Course Name, Specification Name, Specification fee, Course Type)

Module-3(Module ID(PK), Class, Marks)

Module details-3(Module ID(FK), Course ID(FK), Module Name, Module Leader)

Student-3(Student ID(PK), Name, Sex, Joined Date, Student Status, Year)

Student Info-3(Student ID(FK), Course ID(FK))

Teacher-3(Teacher ID(PK), Name, Sex, Joined Date, Teacher Type, Teacher Status, Salary)

Teacher Info-3(Teacher ID(FK), Course ID(FK))

sAddress-3(AddressStd ID(PK), Country ID(FK), DOB ID(FK), Mobile Number, Email)

sDOB-3(DOB ID(PK), DOB, Age)

sCountry-3 (Country ID(PK), Country, Province, City, Street Number, House Number (FK))

sResidence-3(House Number (PK), Phone Number, Fax Number)

sAddress Info-3(AddressStd ID(FK), Student ID(FK))

tAddress-3(AddressTec ID(PK), Country ID(FK), DOB ID(FK), Mobile Number, Email)

tDOB-3(DOB ID(PK), DOB, Age)

tCountry-3 (Country ID(PK), Country, Province, City, Street Number, House Number (FK))

tResidence-3(House Number (PK), Phone Number, Fax Number)

tAddress Info-3(AddressTec ID(FK), Teacher ID(FK))

2.3. ER Diagram of Normalized Database

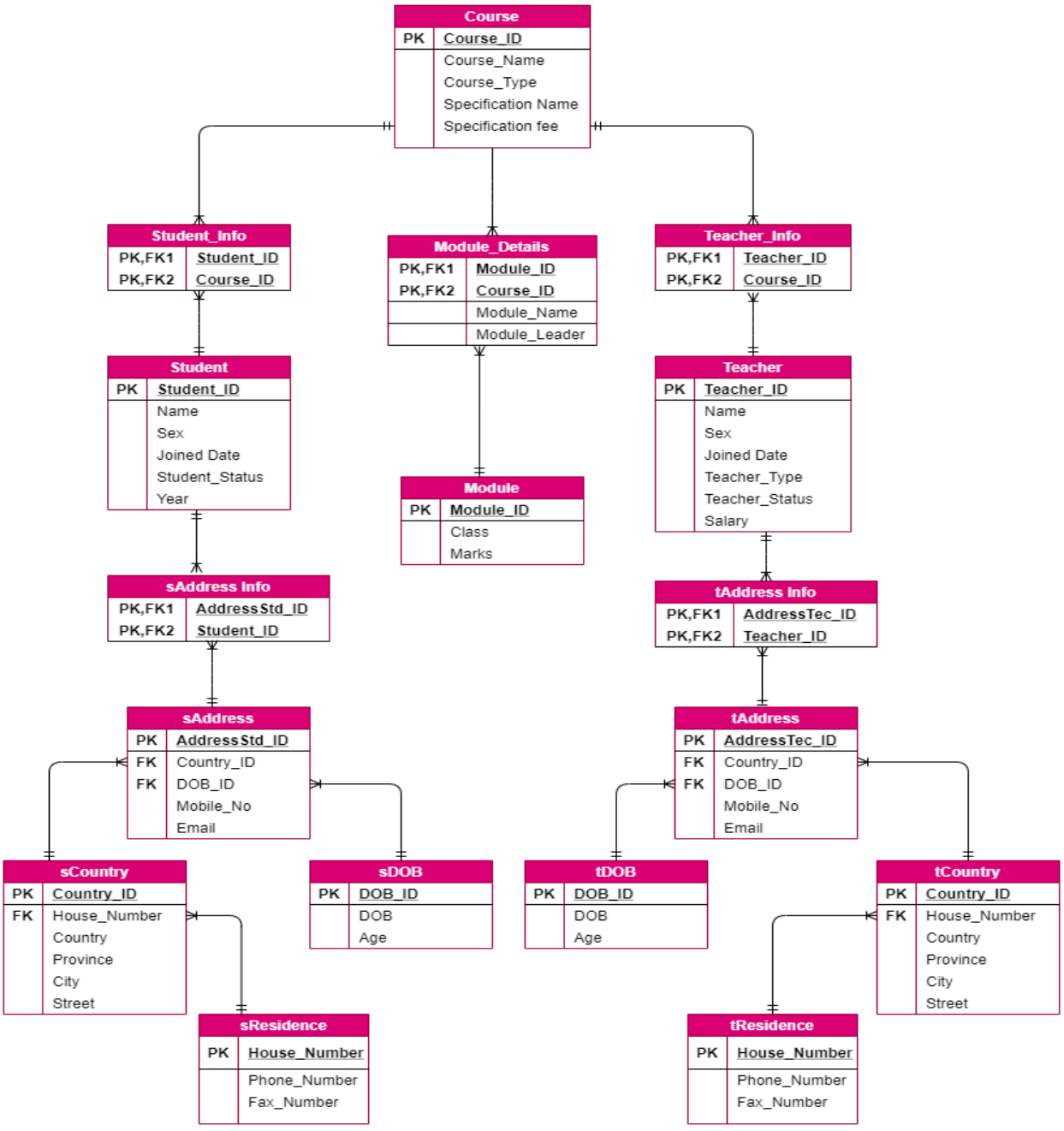


Figure 2 Final ERD

This final Entity Relationship Diagram has been implemented after the normalization of entities. This process enables a better representation of the user working requirement. The above normalization Entity Relationship Diagram improves the modeling effort by facilitating a better fit (DATAVERSITY Education, 2020).

Chapter 3 – Implementation

3.1. Table Creation

To create a table inside database ‘CREATE TABLE’ command is used which is one of the most complex statement in SQL. To create a table following things must be implemented: name of the table, name of the field and definition of every field. The ALTER TABLE statement is used to add, delete, modify columns or constraints in an existing table (W3Schools, 2020). A primary key is a column or set of columns that uniquely identify each row in a table (Techopedia, 2019). A foreign key is a field or collection of field in one table which refers to the primary key in another table (W3School, 2020).

Creating Table Course:

```
CREATE TABLE Course(

Course_ID INT NOT NULL,

Course_Name VARCHAR(200) NOT NULL,

Course_Type VARCHAR(200) NOT NULL,

Specification_Name VARCHAR(200) NOT NULL,

Specification_Fees INT NOT NULL,

CONSTRAINT Course_PK

PRIMARY KEY(Course_ID));
```

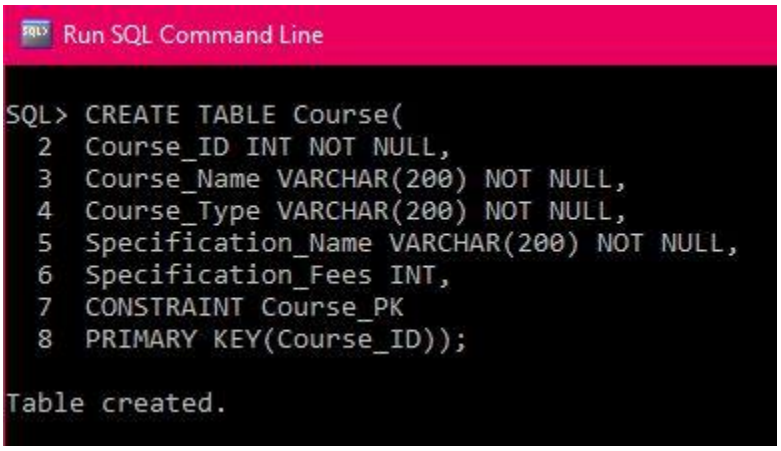


Figure 3 Create table Course

Creating table Student:

CREATE TABLE Student(

Student_ID INT NOT NULL,

Name VARCHAR(200) NOT NULL,

Sex VARCHAR(10) NOT NULL,

Joined_Date DATE NOT NULL,

Student_Status VARCHAR(10) NOT NULL,

Year VARCHAR(10) NOT NULL,

CONSTRAINT Student_PK

PRIMARY KEY(Student_ID));

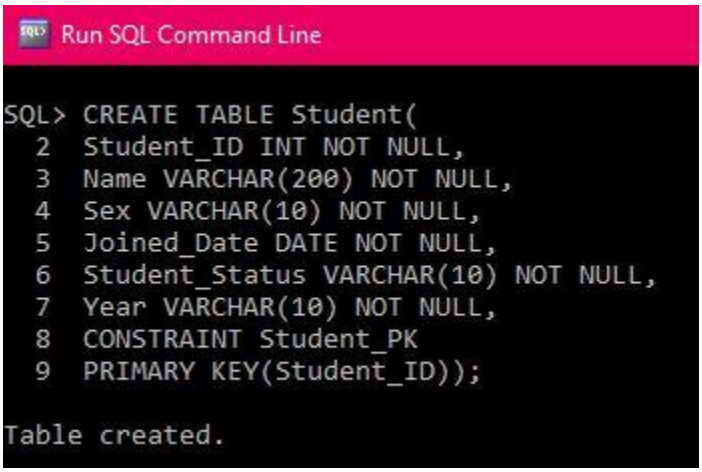


Figure 4 Create table Student

Creating table Teacher:

CREATE TABLE Teacher(

Teacher_ID INT NOT NULL,

Name VARCHAR(200) NOT NULL,

Sex VARCHAR(10) NOT NULL,

Joined_Date DATE NOT NULL,

Teacher_Type VARCHAR(20) NULL,

Teacher_Status VARCHAR(20) NOT NULL,

Salary VARCHAR(200) NOT NULL,

CONSTRAINT Teacher_PK

PRIMARY KEY(Teacher_ID));

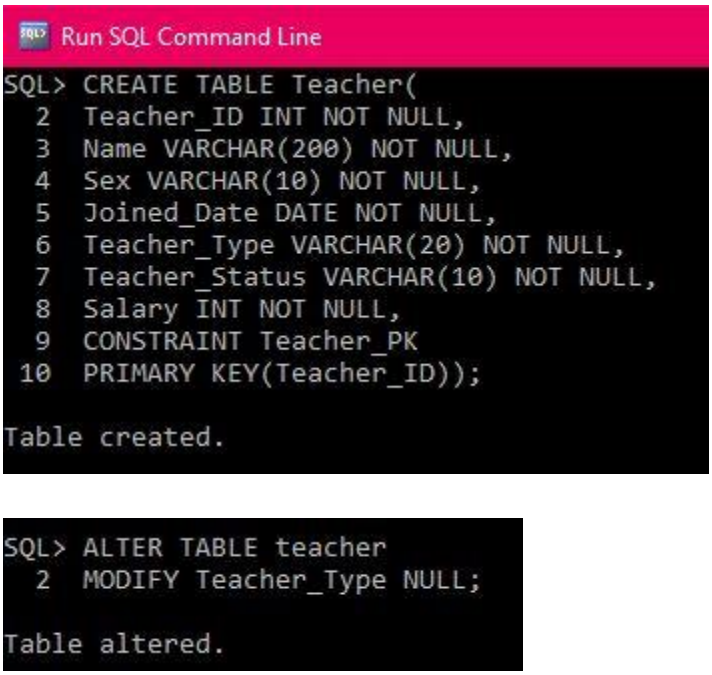


Figure 5 Create table Teacher

Creating table Module:

CREATE TABLE Module(

Module_ID INT NOT NULL,

Class VARCHAR(50) NOT NULL,

Marks INT,

CONSTRAINT Module_PK

PRIMARY KEY(Module_ID));

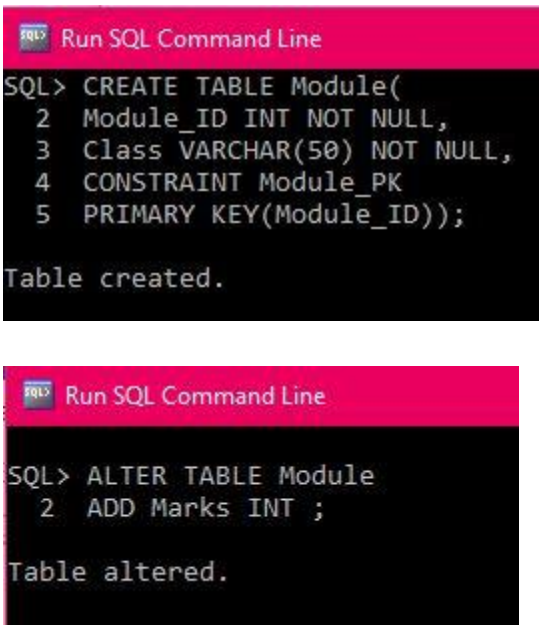


Figure 6 Create table Module

Creating table Student_Info:

CREATE TABLE Student_Info(

Student_ID INT NOT NULL,

Course_ID INT NOT NULL,

CONSTRAINT StdInfo_PK

PRIMARY KEY(Student_ID, Course_ID),

CONSTRAINT StdInfo_FK

FOREIGN KEY(Student_ID) REFERENCES Student(Student_ID),

CONSTRAINT StdInfo_FK1

FOREIGN KEY(Course_ID) REFERENCES Course(Course_ID));

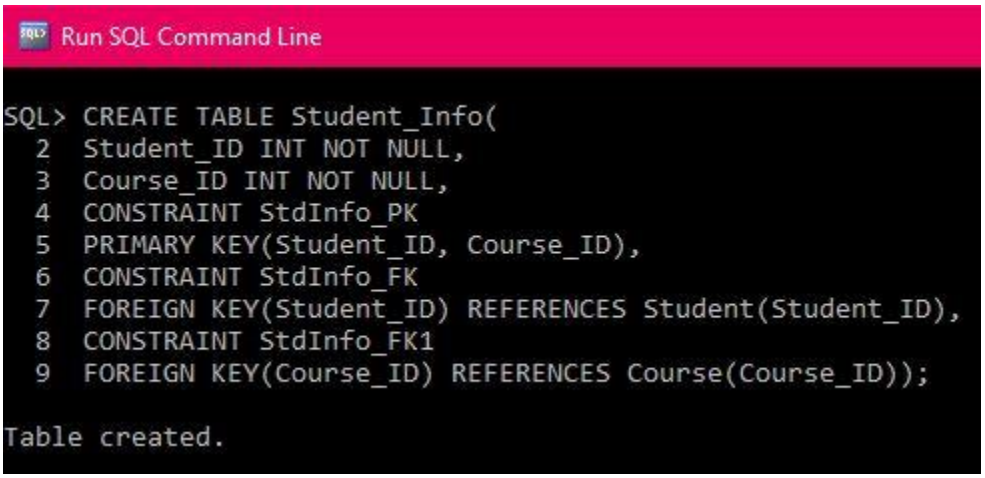


Figure 7 Create table Student_Info

Creating table Teacher_Info:

CREATE TABLE Teacher_Info(

Teacher_ID INT NOT NULL,

Course_ID INT NOT NULL,

CONSTRAINT TecInfo_PK

PRIMARY KEY(Teacher_ID, Course_ID),

CONSTRAINT TecInfo_FK

FOREIGN KEY(Teacher_ID) REFERENCES Teacher(Teacher_ID),

CONSTRAINT TecInfo_FK1

FOREIGN KEY(Course_ID) REFERENCES Course(Course_ID));

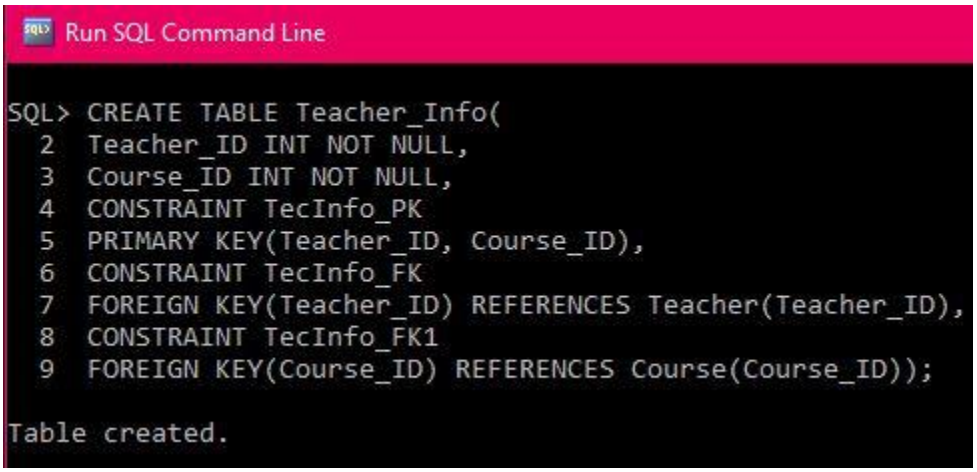


Figure 8 Create table Teacher_Info

Creating table Module_Details:

CREATE TABLE Module_Details(

Module_ID INT NOT NULL,

Course_ID INT NOT NULL,

Module_Name VARCHAR(200) NOT NULL,

Module_Leader VARCHAR(200) NOT NULL,

CONSTRAINT ModDetails_PK

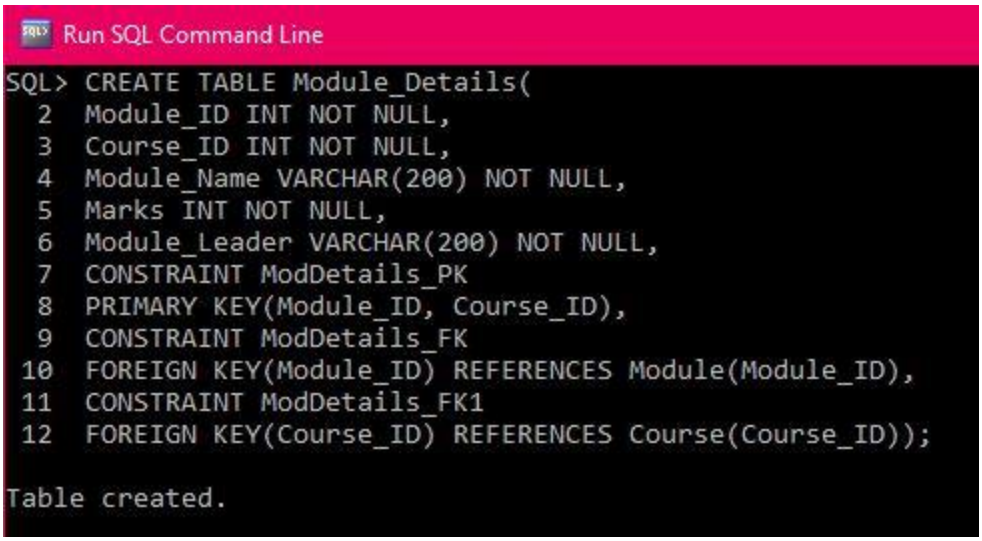
PRIMARY KEY(Module_ID, Course_ID),

CONSTRAINT ModDetails_FK

FOREIGN KEY(Module_ID) REFERENCES Module(Module_ID),

CONSTRAINT ModDetails_FK1

FOREIGN KEY(Course_ID) REFERENCES Course(Course_ID));



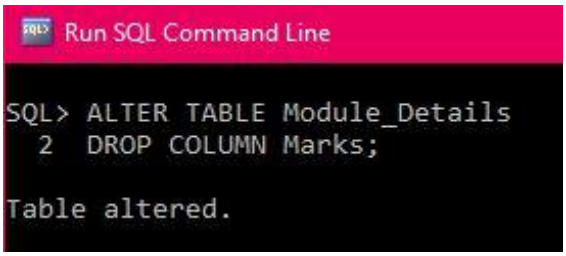


Figure 9 Create table Module_Details

Creating table sResidence:

CREATE TABLE sResidence(

House_Number INT NOT NULL,

Phone_Number INT,

Fax_Number INT,

CONSTRAINT sHouseNumber_PK

PRIMARY KEY(House_Number));

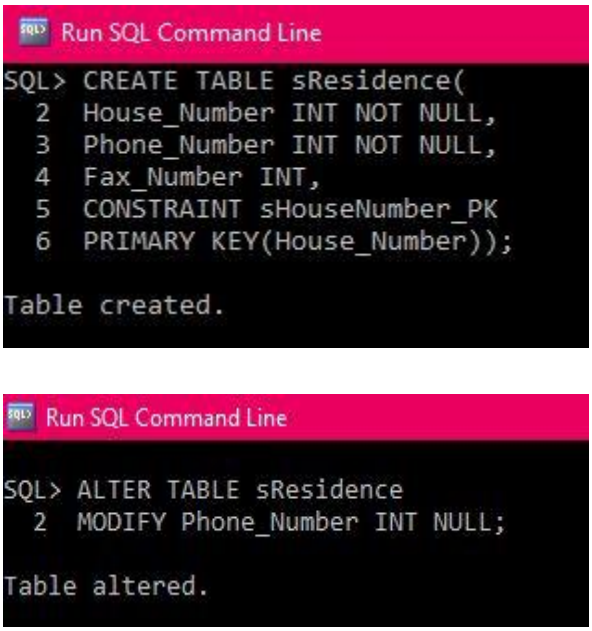


Figure 10 Create table sResidence

Creating table tResidence:

CREATE TABLE tResidence(

House_Number INT NOT NULL,

Phone_Number INT NOT NULL,

Fax_Number INT,

CONSTRAINT tHouseNumber_PK

PRIMARY KEY(House_Number));

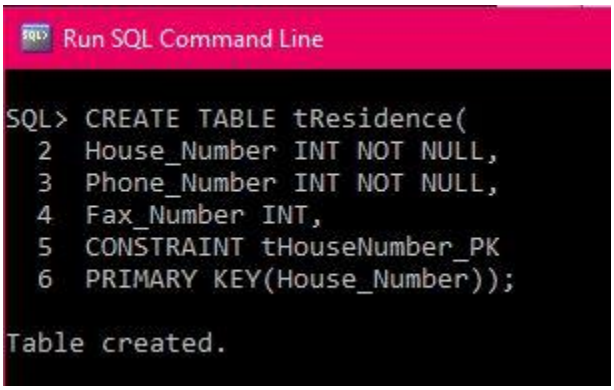


Figure 11 Create table tResidence

Creating table sCountry:

CREATE TABLE sCountry(

Country_ID INT NOT NULL,

House_Number INT NOT NULL,

Country VARCHAR(200) NOT NULL,

Province INT NOT NULL,

City VARCHAR(200) NOT NULL,

Street VARCHAR(200) NOT NULL,

CONSTRAINT sCountry_PK

PRIMARY KEY(Country_ID),

CONSTRAINT sHouseNumber_FK

FOREIGN KEY(House_Number) REFERENCES sResidence(House_Number));

```
Run SQL Command Line
SQL> CREATE TABLE sCountry(
  2  Country_ID INT NOT NULL,
  3  House_Number INT NOT NULL,
  4  Country VARCHAR(200) NOT NULL,
  5  Province INT NOT NULL,
  6  City VARCHAR(200) NOT NULL,
  7  Street VARCHAR(200) NOT NULL,
  8  CONSTRAINT sCountry_PK
  9  PRIMARY KEY(Country_ID),
 10  CONSTRAINT sHouseNumber_FK
 11  FOREIGN KEY(House_Number) REFERENCES sResidence(House_Number));

Table created.
```

Figure 12 Create table sCountry

Creating table tCountry:

CREATE TABLE tCountry(

Country_ID INT NOT NULL,

House_Number INT NOT NULL,

Country VARCHAR(200) NOT NULL,

Province INT NOT NULL,

City VARCHAR(200) NOT NULL,

Street VARCHAR(200) NOT NULL,

CONSTRAINT tCountry_PK

PRIMARY KEY(Country_ID),

CONSTRAINT tHouseNumber_FK

FOREIGN KEY(House_Number) REFERENCES tResidence(House_Number));

Figure 13 Create table tCountry

Creating table sDOB:

CREATE TABLE sDOB(

DOB_ID INT NOT NULL,

DOB DATE NOT NULL,

Age INT NOT NULL,

CONSTRAINT DOB_PK

PRIMARY KEY(DOB_ID));

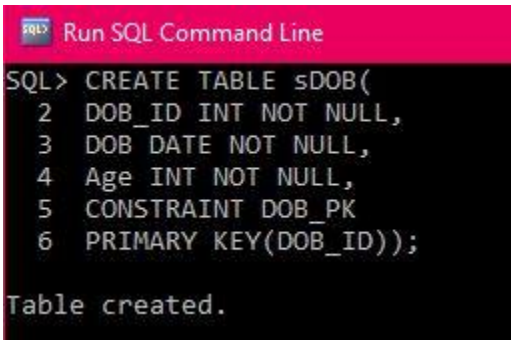


Figure 14 Create table sDOB

Creating table tDOB:

```
CREATE TABLE tDOB(
DOB_ID INT NOT NULL,
DOB DATE NOT NULL,
Age INT NOT NULL,
CONSTRAINT tDOB_PK
PRIMARY KEY(DOB_ID));
```

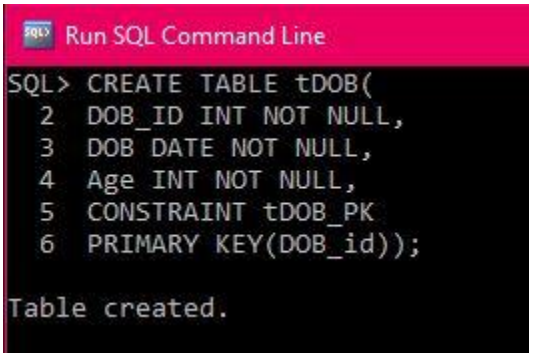


Figure 15 Create table tDOB

Creating table sAddress:

```
CREATE TABLE sAddress(
AddressStd_ID INT NOT NULL,
Country_ID INT NOT NULL,
DOB_ID INT NOT NULL,
Mobile_Number INT NOT NULL,
Email VARCHAR(200) NOT NULL,
CONSTRAINT AddressSTd_PK
PRIMARY KEY(AddressStd_ID),
```

CONSTRAINT Country_FK

FOREIGN KEY(Country_ID) REFERENCES sCountry(Country_ID),

CONSTRAINT DOB_FK

FOREIGN KEY(DOB_ID) REFERENCES sDOB(DOB_ID));

```
Run SQL Command Line
SQL> CREATE TABLE sAddress(
  2  AddressStd_ID INT NOT NULL,
  3  Country_ID INT NOT NULL,
  4  DOB_ID INT NOT NULL,
  5  Mobile_Number INT NOT NULL,
  6  Email VARCHAR(200) NOT NULL,
  7  CONSTRAINT AddressSTd_PK
  8  PRIMARY KEY(AddressStd_ID),
  9  CONSTRAINT Country_FK
 10  FOREIGN KEY(Country_ID) REFERENCES sCountry(Country_ID),
 11  CONSTRAINT DOB_FK
 12  FOREIGN KEY(DOB_ID) REFERENCES sDOB(DOB_ID));

Table created.
```

Figure 16 Create table sAddress

Creating table tAddress:

CREATE TABLE tAddress(

AddressTec_ID INT NOT NULL,

Country_ID INT NOT NULL,

DOB_ID INT NOT NULL,

Mobile_Number INT NOT NULL,

Email VARCHAR(200) NOT NULL,

CONSTRAINT AddressTec_PK

PRIMARY KEY(AddressTec_ID),

CONSTRAINT tCountry_FK

FOREIGN KEY(Country_ID) REFERENCES tCountry(Country_ID),

CONSTRAINT tDOB_FK

FOREIGN KEY(DOB_ID) REFERENCES tDOB(DOB_ID));

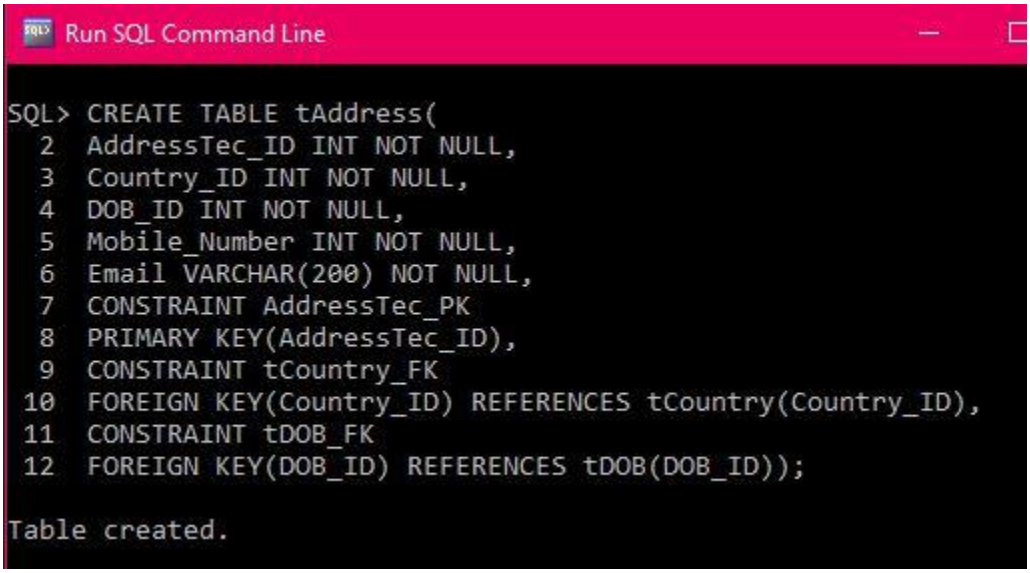


Figure 17 Create table tAddress

Creating table sAddress_Info(

CREATE TABLE sAddress_Info(

AddressStd_ID INT NOT NULL,

Student_ID INT NOT NULL,

CONSTRAINT sAddress_PK

PRIMARY KEY(AddressStd_ID, Student_ID),

CONSTRAINT AddressStd_FK

FOREIGN KEY(AddressStd_ID) REFERENCES sAddress(AddressStd_ID),

CONSTRAINT Student_FK

FOREIGN KEY(Student_ID) REFERENCES Student(Student_ID));

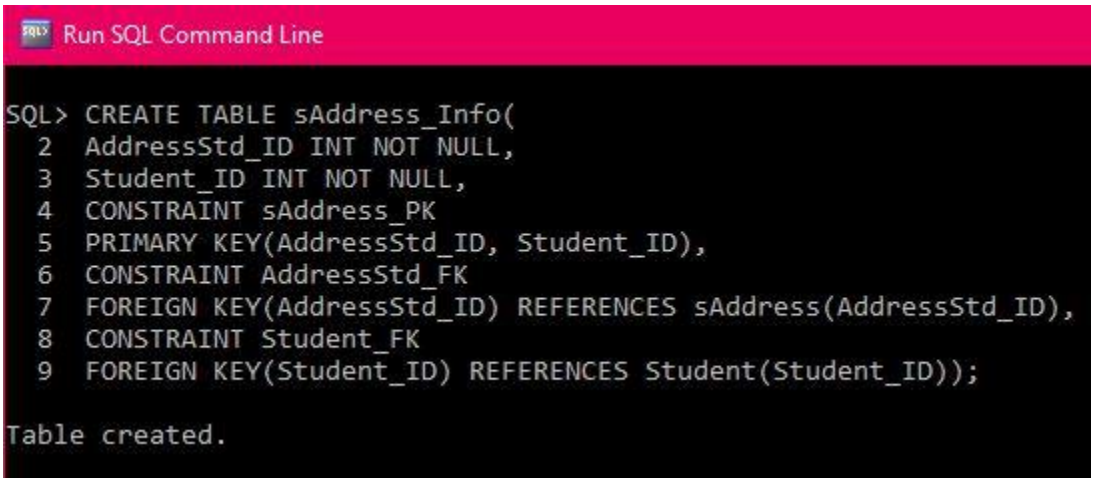


Figure 18 Create table sAddress_Info

Creating table tAddress_Info(

CREATE TABLE tAddress_Info(

AddressTec_ID INT NOT NULL,

Teacher_ID INT NOT NULL,

CONSTRAINT tAddress_PK

PRIMARY KEY(AddressTec_ID, Teacher_ID),

CONSTRAINT AddressTec_FK

FOREIGN KEY(AddressTec_ID) REFERENCES tAddress(AddressTec_ID),

CONSTRAINT Teacher_FK

FOREIGN KEY(Teacher_ID) REFERENCES Teacher(Teacher_ID));

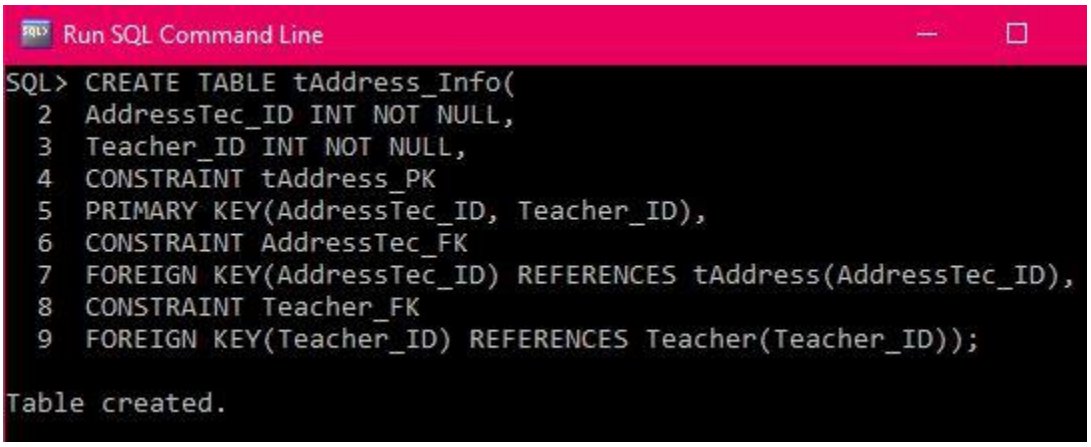


Figure 19 Create table tAddress_Info

3.2. Populating Database Table

'INSERT INTO' command is used to insert data into the tables. 'COMMIT' command is used to save the inserted data permanently. By using 'INSERT INTO' and 'COMMIT' command in Oracle SQL Plus all the data will be stored and saved and secured.

Inserting values in Course table:

INSERT INTO Course(Course_ID, Course_Name, Course_Type, Specification_Name, Specification_Fees)VALUES(001, 'BIT', 'Bachelor', 'Computing', 115000);

INSERT INTO Course(Course_ID, Course_Name, Course_Type, Specification_Name, Specification_Fees)VALUES(002, 'BIT', 'Bachelor', 'Networking', 115000);

INSERT INTO Course(Course_ID, Course_Name, Course_Type, Specification_Name, Specification_Fees)VALUES(003, 'BIT', 'Bachelor', 'Multimedia', 115000);

INSERT INTO Course(Course_ID, Course_Name, Course_Type, Specification_Name, Specification_Fees)VALUES(004, 'MSc IT', 'Master', 'Security', 215000);

INSERT INTO Course(Course_ID, Course_Name, Course_Type, Specification_Name, Specification_Fees)VALUES(005, 'BBA', 'Bachelor', 'Marketing', 110000);

INSERT INTO Course(Course_ID, Course_Name, Course_Type, Specification_Name, Specification_Fees)VALUES(006, 'BBA', 'Bachelor', 'Finance', 110000);

INSERT INTO Course(Course_ID, Course_Name, Course_Type, Specification_Name, Specification_Fees)VALUES(007, ‘MBA’, ‘Master’, ‘Accounting’, 210000);

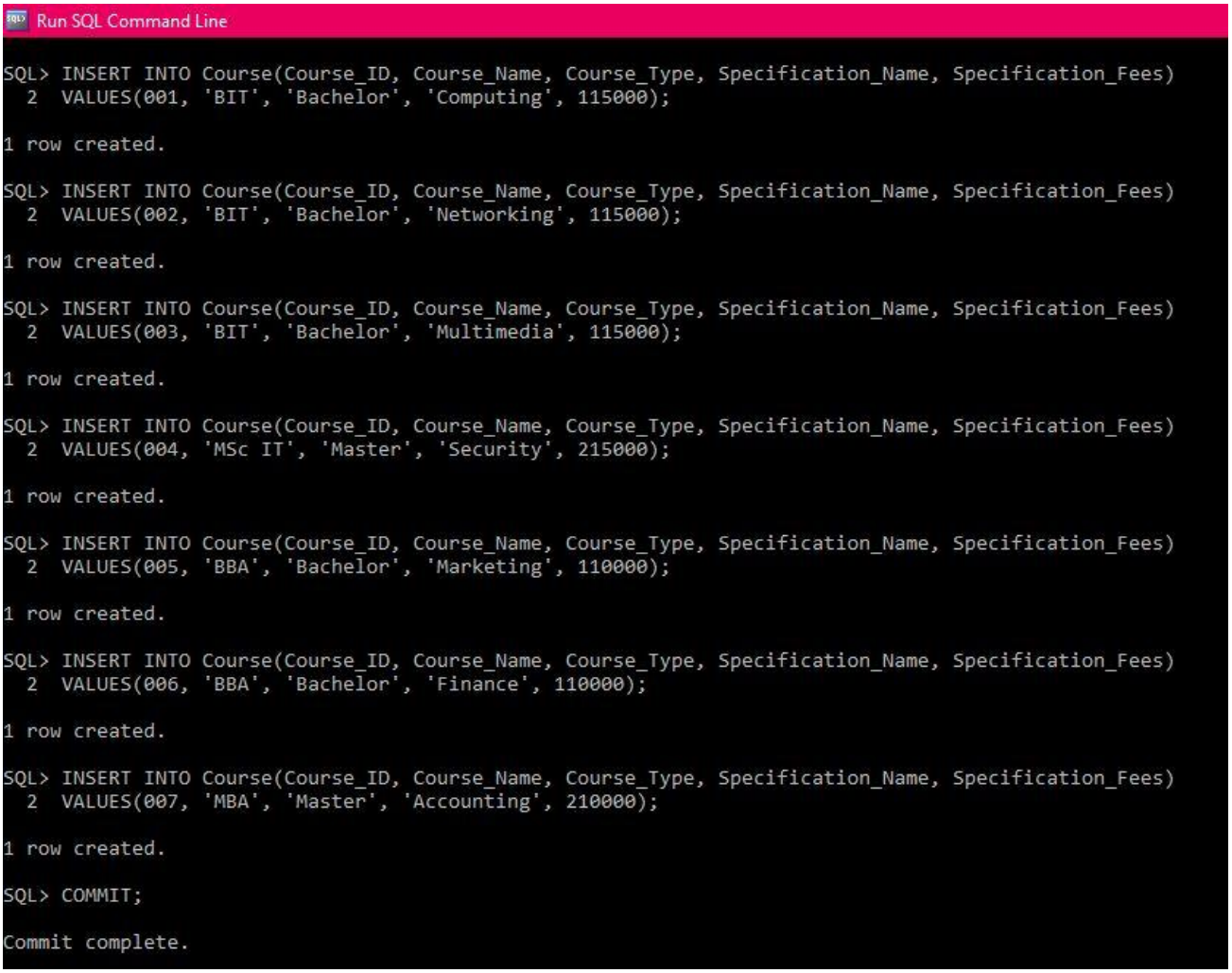


Figure 20 Insert Into Course

Inserting values in Student table:

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)VALUES(101, ‘Siddhartha Ghimire’, ‘Male’, ‘01-Jan-2019’, ‘Active’, ’Second’);

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)VALUES(102, ‘Rihan Ojha’, ‘Male’, ‘01-Jan-2018’, ‘Active’, ’Third’);

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)VALUES(103, ‘Chirag Timalsina’, ‘Male’, ‘01-Jan-2018’, ‘Inactive’, ‘Third’);

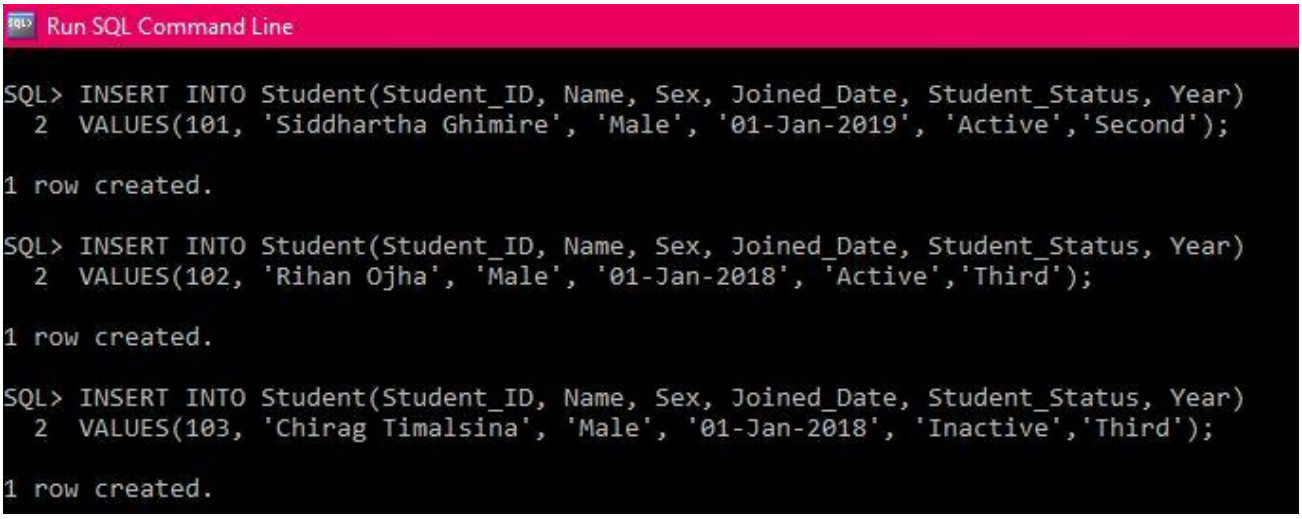
```
INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(108, 'Rohan Gurung', 'Male', '01-Jan-2019', 'Active', 'Second');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(109, 'Bishnu Thapa', 'Male', '01-Jan-2020', 'Inactive', 'First');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(110, 'Ashish Shrestha', 'Male', '01-Jan-2018', 'Active', 'Third');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(111, 'Aayusha Shrestha', 'Female', '01-Jan-2020', 'Active', 'First');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(112, 'Shiv Thakur', 'Male', '01-Jan-2019', 'Inactive', 'Second');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(113, 'Sahayog G.C.', 'Male', '01-Jan-2018', 'Inactive', 'Third');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(114, 'Shrijana Rokaya', 'Female', '01-Jan-2019', 'Active', 'Second');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(115, 'Bikram Rathor', 'Male', '01-Jan-2018', 'Active', 'Third');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(116, 'Piyush Sharma', 'Male', '02-Jan-2019', 'Inactive', 'Second');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(117, 'Prajeet Thakur', 'Male', '01-Jan-2018', 'Active', 'Third');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(118, 'Azan Ahmad', 'Male', '01-Feb-2020', 'Inactive', 'First');

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status,
Year)VALUES(119, 'Smriti Thapa', 'Female', '12-Jan-2018', 'Active', 'Third');
```

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)VALUES(104, ‘Priyanka Chopra’, ‘Female’, ‘01-Jan-2020’, ‘Active’, ‘First’);

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)VALUES(105, ‘Katrina Kaif’, ‘Female’, ‘01-Jan-2019’, ‘Active’, ‘Second’);

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)VALUES(106, ‘Dwayne Johnson’, ‘Male’, ‘01-Jan-2020’, ‘Inactive’, ’First’);

INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)VALUES(107, ‘Simon Wilson’, ‘Male’, ‘01-Jan-2018’, ‘Inactive’, ’Third’);



```
SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(108, 'Rohan Gurung', 'Male', '01-Jan-2019', 'Active', 'Second');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(109, 'Bishnu Thapa', 'Male', '01-Jan-2020', 'Inactive', 'First');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(110, 'Ashish Shrestha', 'Male', '01-Jan-2018', 'Active', 'Third');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(111, 'Aayusha Shrestha', 'Female', '01-Jan-2020', 'Active', 'First');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(112, 'Shiv Thakur', 'Male', '01-Jan-2019', 'Inactive', 'Second');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(113, 'Sahayog G.C.', 'Male', '01-Jan-2018', 'Inactive', 'Third');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(114, 'Shrijana Rokaya', 'Female', '01-Jan-2019', 'Active', 'Second');

1 row created.
```

```
SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(115, 'Bikram Rathor', 'Male', '01-Jan-2018', 'Active', 'Third');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(116, 'Piyush Sharma', 'Male', '02-Jan-2019', 'Inactive', 'Second');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(117, 'Prajeet Thakur', 'Male', '01-Jan-2018', 'Active', 'Third');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(118, 'Azan Ahmad', 'Male', '01-Feb-2020', 'Inactive', 'First');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(119, 'Smriti Thapa', 'Female', '12-Jan-2018', 'Active', 'Third');

1 row created.
```

```
SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(104, 'Priyanka Chopra', 'Female', '01-Jan-2020', 'Active','First');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(105, 'Katrina Kaif', 'Female', '01-Jan-2019', 'Active','Second');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(106, 'Dwayne Johnson', 'Male', '01-Jan-2020', 'Inactive','First');

1 row created.

SQL> INSERT INTO Student(Student_ID, Name, Sex, Joined_Date, Student_Status, Year)
  2  VALUES(107, 'Simon Wilson', 'Male', '01-Jan-2018', 'Inactive','Third');

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 21 Insert into Student

Inserting values in Teacher table:

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(201, ‘Steve Jobs’,’Male’,’01-Feb-2000’, ‘Module Leader’, ’Active’, 60000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(202, ‘Sabin Bharati’,’Male’,’01-Jan-2005’, ‘Module Leader’, ’Active’, 35000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(203, ‘Mark Zuckerburg’,’Male’,’01-Dec-1999’, ‘Module Leader’, ’Active’, 75000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(204, ‘Ramesh Thapa’,’Male’,’01-Feb-2013’, ‘Module Leader’, ’Active’, 48000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(205, ‘Jayesh Karmacharya’,’Male’,’01-May-2002’, ‘Module Leader’, ’Active’, 40000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(206, ‘Sabina Tamrakar’,’Female’,’01-Feb-2010’, ‘Module Leader’, ’Active’, 52000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(207, ‘Wiz Khalif’,’Male’,’01-Jun-2018’, ‘Module Leader’, ’Active’, 55000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(208, ‘Anup Thakur’,’Male’,’01-Jun-2018’, ‘Module Leader’, ’Active’, 54000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(209, ‘Ram Nepal’,’Male’,’01-Sep-2018’, ‘Module Leader’, ’Active’, 20000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(215, ‘Sameer Khadka’,’Male’,’01-Sep-2018’, ‘Instructor’, ’Active’, 62000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(216, ‘Piyush Khadka’,’Male’,’01-Feb-2017’, ‘Instructor’, ’Active’, 68000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(217, ‘Ramesh Karmacharya’,’Male’,’01-Sep-2017’, ‘Instructor’, ’Inactive’, 51000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(218, ‘Neha Khadka’,’Female’,’12-Sep-2019’, ‘Instructor’, ’Active’, 64000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(219, ‘Rabin Bajracharya’,’Male’,’01-Sep-2020’, ‘Instructor’, ’Active’, 60000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(220, ‘Simran Sha’,’Female’,’01-Nov-2018’, ‘Instructor’, ’Inactive’, 65000);

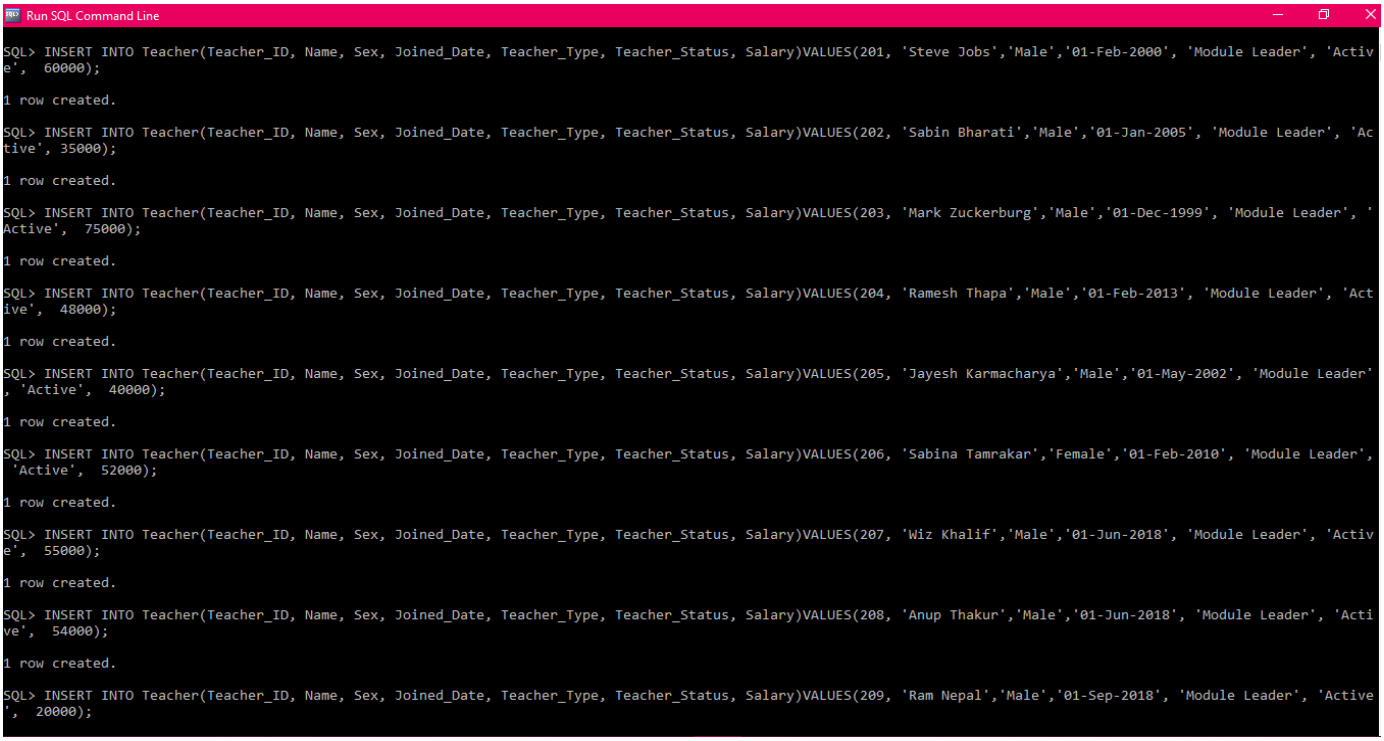
INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(210, ‘Utsav Rajput’,’Male’,’01-Jun-2013’, ‘Module Leader’, ’Active’, 15000);

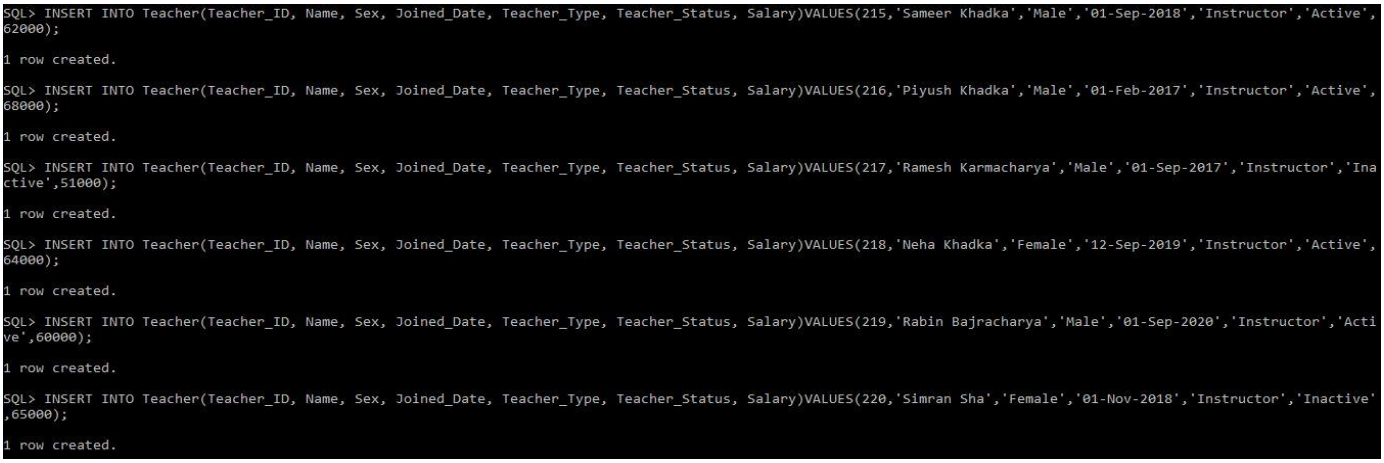
INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(211, ‘Yubraj Khadka’,’Male’,’01-Sep-2005’, ‘Module Leader’, ’Active’, 59000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(212, ‘Hari Ghimire’,’Male’,’01-Jan-2020’, ‘Module Leader’, ’Active’, 50000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(213, ‘Bom Bahadur’,’Male’,’01-Dec-2019’, ‘Module Leader’, ’Active’, 54000);

INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(214, ‘Govind Chaudhary’,’Male’,’08-Jun-2019’, ‘Module Leader’, ’Active’, 50000);





```
1 row created.

SQL> INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(210, 'Utsav Rajput','Male','01-Jun-2013', 'Module Leader', 'Act
ive',  15000);

1 row created.

SQL> INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(211, 'Yubraj Khadka','Male','01-Sep-2005', 'Module Leader', 'Ac
tive',  59000);

1 row created.

SQL> INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(212, 'Hari Ghimire','Male','01-Jan-2020', 'Module Leader', 'Act
ive',  50000);

1 row created.

SQL> INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(213, 'Bom Bahadur','Male','01-Dec-2019', 'Module Leader', 'Acti
ve',  54000);

1 row created.

SQL> INSERT INTO Teacher(Teacher_ID, Name, Sex, Joined_Date, Teacher_Type, Teacher_Status, Salary)VALUES(214, 'Govind Chaudhary','Male','08-Jun-2019', 'Module Leader',
'Active',  50000);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 22 Insert into Teacher

Inserting values in Module table:

INSERT INTO Module(Module_ID, Class, Marks)VALUES(51, ‘Kanchanjanga’, 70);

INSERT INTO Module(Module_ID, Class, Marks)VALUES(52, ‘Bristol’, 82);

INSERT INTO Module(Module_ID, Class, Marks)VALUES(53, ‘Kingstone’, 42);

INSERT INTO Module(Module_ID, Class, Marks)VALUES(54, ‘Pokhara’, 37);

INSERT INTO Module(Module_ID, Class, Marks)VALUES(55, ‘Buckingham’, 82);

Run SQL Command Line

```
SQL> INSERT INTO Module(Module_ID, Class, Marks)VALUES(51,'Kanchanjanga',70);

1 row created.

SQL> INSERT INTO Module(Module_ID, Class, Marks)VALUES(52,'Bristol',82);

1 row created.

SQL> INSERT INTO Module(Module_ID, Class, Marks)VALUES(53,'Kingstone',42);

1 row created.

SQL> INSERT INTO Module(Module_ID, Class, Marks)VALUES(54,'Pokhara',37);

1 row created.

SQL> INSERT INTO Module(Module_ID, Class, Marks)VALUES(55,'Buckingham',82);

1 row created.
```

```
SQL> COMMIT;

Commit complete.
```

Figure 23 Insert into Module

Inserting values in Student_Info table:

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(101, 1);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(102, 2);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(103, 3);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(104, 4);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(105, 5);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(106, 6);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(107, 7);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(115, 2);

```
INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(116, 2);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(117, 2);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(118, 2);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(119, 2);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(108, 1);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(109, 3);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(110, 2);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(111, 5);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(112, 4);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(113, 7);

INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(114, 6);
```

Run SQL Command Line

```
SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(101, 1);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(102, 2);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(103, 3);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(104, 4);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(105, 5);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(106, 6);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(107, 7);

1 row created.
```

```
SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(115, 2);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(116, 2);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(117, 2);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(118, 2);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(119, 2);

1 row created.
```

```
SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(108, 1);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(109, 3);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(110, 2);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(111, 5);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(112, 4);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(113, 7);

1 row created.

SQL> INSERT INTO Student_Info(Student_ID, Course_ID)VALUES(114, 6);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 24 Insert into Student_Info

Inserting values in Teacher_Info table:

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(201, 1);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(201, 3);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(202, 7);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(203, 1);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(204, 5);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(204, 6);

```
INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(205, 1);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(205, 2);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(206, 3);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(207, 7);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(208, 2);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(218, 6);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(216, 4);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(217, 5);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(215, 6);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(219, 6);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(220, 6);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(209, 4);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(210, 2);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(211, 4);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(212, 5);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(213, 6);

INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(214, 3);
```


Run SQL Command Line
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(201, 1);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(201, 3);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(202, 7);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(203, 1);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(204, 5);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(204, 6);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(205, 1);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(205, 2);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(206, 3);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(207, 7);
1 row created.
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(208, 2);
1 row created.

```
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(218, 6);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(216, 4);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(217, 5);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(215, 6);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(219, 6);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(220, 6);

1 row created.
```

```
SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(209, 4);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(210, 2);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(211, 4);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(212, 5);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(213, 6);

1 row created.

SQL> INSERT INTO Teacher_Info(Teacher_ID, Course_ID) VALUES(214, 3);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 25 Insert into Teacher_Info

Inserting values in Module_Details table:

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 1, ‘Database’, ’Mark Zuckerburg’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(52, 1, ‘Programming’, ’Steve Jobs’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(53, 1, ‘Information System’, ’Jayesh Karmacharya’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(54, 2, ‘Information System’, ’Jayesh Karmacharya’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(55, 2, ‘Networking Concepts’, ’Utsav Rajput’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 2, ‘Communications Engineering ’, ’Anup Thakur’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 3, ‘Digital Design’, ’Govind Chaudhary’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(53, 3, ‘Programming’, ’Steve Jobs’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(54, 3, ‘Drawing and Character Design ’, ’Sabina Tamrakar’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 4, ‘Cyber Security’, ’Ram Nepal’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(55, 4, ‘Work Related Learning’, ’Yubraj Khadka’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 5, ‘Economics and Society, ’Ramesh Thapa’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(53, 5, ‘The Corporate Environment’, ’Hari Ghimire’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 6, ‘Economics and Society, ’Ramesh Thapa’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(55, 6, ‘Choices and Changes’, ’Bom Bahadur’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 7, ‘Leadership and Strategic Management’, ’Sabin Bharati’);

INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(52, 7, ‘Management Learning’, ’Wiz Khalif’);

Select Run SQL Command Line

```
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 1, 'Database', 'Mark Zuckerburg');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(52, 1, 'Programming', 'Steve Jobs');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(53, 1, 'Information System', 'Jayesh Karmacharya');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(54, 2, 'Information System', 'Jayesh Karmacharya');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(55, 2, 'Networking Concepts', 'Utsav Rajput');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 2, 'Communications Engineering ', 'Anup Thakur');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 3, 'Digital Design', 'Govind Chaudhary');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(53, 3, 'Programming', 'Steve Jobs');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(54, 3, 'Drawing and Character Design', 'Sabina Tamrakar');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 4, 'Cyber Security', 'Ram Nepal');
1 row created.
SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(55, 4, 'Work Related Learning', 'Yubraj Khadka');
1 row created.
```

```
Select Run SQL Command Line

SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 5, 'Economics and Society', 'Ramesh Thapa');

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(53, 5, 'The Corporate Environment', 'Hari Ghimire');

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 6, 'Economics and Society', 'Ramesh Thapa');

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(55, 6, 'Choices and Changes', 'Bom Bahadur');

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(51, 7, 'Leadership and Strategic Management', 'Sabin Bharati');

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Course_ID, Module_Name, Module_Leader)VALUES(52, 7, 'Management Learning', 'Wiz Khalif');

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 26 Insert into Module_Details

Inserting values in sResidence table:

INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(21,NULL,NULL);

INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(22,9864567899, 6120);

INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(23,NULL,6121);

INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(24,9800818055,6122);

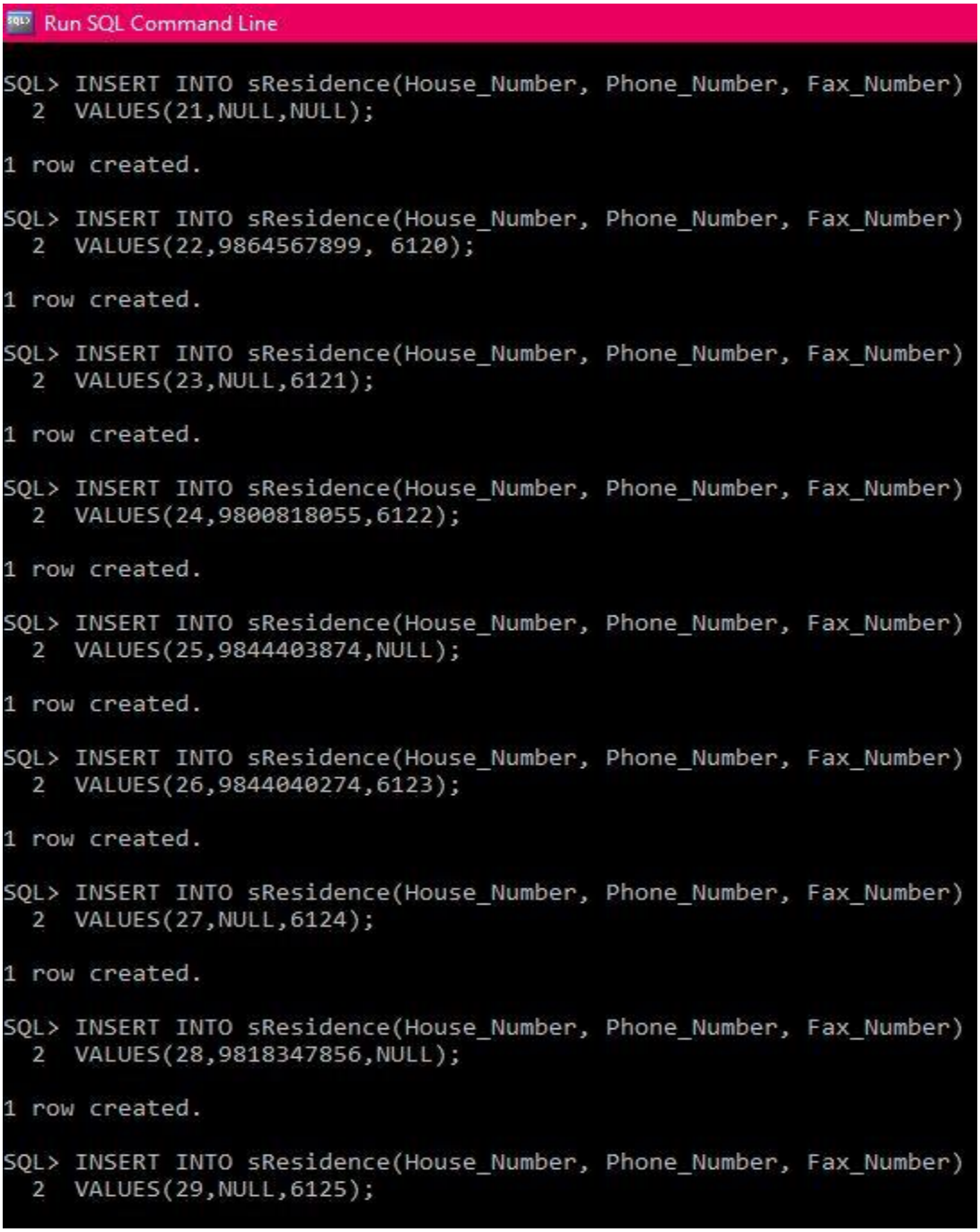
INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(25,9844403874,NULL);

INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(26,9844040274,6123);

```
INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(27,NULL,6124);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(28,9818347856,NULL);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(29,NULL,6125);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(35,9836363444,6130);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(36,9836363474,6131);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(37,9836363244,6132);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(38,9838363444,6133);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(39,9826363444,6134);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(40,9836373444,6135);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(30,9876567890,6126);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(31,9876345672,6127);

INSERT INTO sResidence(House_Number, Phone_Number,
Fax_Number)VALUES(32,NULL,6128);
```

INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(33,9856782234,NULL);

INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)VALUES(34,NULL,6129);



```
SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(35,9836363444,6130);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(36,9836363474,6131);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(37,9836363244,6132);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(38,9838363444,6133);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(39,9826363444,6134);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(40,9836373444,6135);

1 row created.
```

```
1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(30,9876567890,6126);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(31,9876345672,6127);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(32,NULL,6128)
  3  ;

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(33,9856782234,NULL);

1 row created.

SQL> INSERT INTO sResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(34,NULL,6129);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 27 Insert into sResidence

Inserting values in tResidence table:

INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(71,NULL,NULL);

INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(72,9823456890, 6130);

INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(85,9823456290, 6140);

```
INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(86,9823426890, 6141);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(87,9833456890, 6142);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(88,9823556890, 6143);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(89,9823455890, 6144);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(90,9823486890, 6145);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(73,NULL,6131);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(74,9809876543,6132);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(75,9856789012,NULL);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(76,9867890123,6133);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(77,NULL,6134);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(78,9878901234,NULL);

INSERT INTO tResidence(House_Number, Phone_Number,
Fax_Number)VALUES(79,NULL,6135);
```

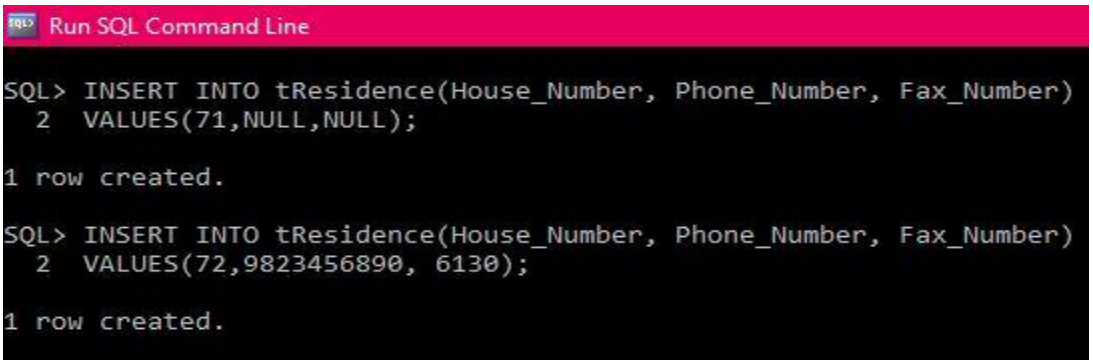
INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(80,9872345456,6136);

INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(81,9876654567,6137);

INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(82,NULL,6138);

INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(83,9856798765,NULL);

INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)VALUES(84,NULL,6139);



```
SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(85,9823456290, 6140);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(86,9823426890, 6141);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(87,9833456890, 6142);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(88,9823556890, 6143);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(89,9823455890, 6144);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(90,9823486890, 6145);

1 row created.
```

```
SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(73,NULL,6131);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(74,9809876543,6132);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(75,9856789012,NULL);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(76,9867890123,6133);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(77,NULL,6134);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(78,9878901234,NULL);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(79,NULL,6135);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(80,9872345456,6136);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(81,9876654567,6137);

1 row created.
```

```
SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(82,NULL,6138);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(83,9856798765,NULL);

1 row created.

SQL> INSERT INTO tResidence(House_Number, Phone_Number, Fax_Number)
  2  VALUES(84,NULL,6139);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 28 Insert into tResidence

Inserting values in sCountry table:

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(171, 21, ‘Nepal’, 3, ‘Lalitpur’, ‘Satdobato-15’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(172, 22, ‘Nepal’, 3, ‘Kathmandu’, ‘Koteshwor-10’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(173, 23, ‘Nepal’, 2, ‘Sindhuli’, ‘Kamalamai-8’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(174, 24, ‘India’, 5, ‘Mumbai’, ‘Jaypur-6’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(175, 25, ‘India’, 5, ‘Mumbai’, ‘Delhi-15’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(176, 26, ‘USA’, 8, ‘Hollywood’, ‘Gantok-5’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(177, 27, ‘Mexico’, 3, ‘Sinaloa’, ‘Cartel-2’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(178, 28, ‘Nepal’, 3, ‘Kathmandu’, ‘Balaju-3’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(179, 29, ‘Nepal’, 3, ‘Lalitpur’, ‘Balkumari-15’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(185, 35, ‘Nepal’, 3, ‘Lalitpur’, ‘Balkumari-16’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(186, 36, ‘Nepal’, 4, ‘Pokhara’, ‘Lakeside-15’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(187, 37, ‘Nepal’, 6, ‘Jhapa’, ‘Birtamod-2’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(188, 38, ‘Nepal’, 3, ‘Lalitpur’, ‘Sanepa-7’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(189, 39, ‘Nepal’, 3, ‘Lalitpur’, ‘Dhobighat-18’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(180, 30, ‘Nepal’, 4, ‘Chitwan’, ‘Bharatpur-7’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(181, 31, ‘Nepal’, 6, ‘Pokhara’, ‘Sarangkot-9’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(182, 32, ‘Nepal’, 4, ‘Janakpur’, ‘Bardibas-11’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(183, 33, ‘Nepal’, 3, ‘Lalitpur’, ‘Patan-2’);

INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(184, 34, ‘Nepal’, 3, ‘Lalitpur’, ‘Jawalakhel-8’);

Run SQL Command Line

```
SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(171, 21, 'Nepal', 3, 'Lalitpur', 'Satdobato-15');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(172, 22, 'Nepal', 3, 'Kathmandu', 'Koteshwor-10');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(173, 23, 'Nepal', 2, 'Sindhuli', 'Kamalamai-8');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(174, 24, 'India', 5, 'Mumbai', 'Jaypur-6');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(175, 25, 'India', 5, 'Mumbai', 'Delhi-15');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(176, 26, 'USA', 8, 'Hollywood', 'Gantok-5');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(177, 27, 'Mexico', 3, 'Sinaloa', 'Cartel-2');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(178, 28, 'Nepal', 3, 'Kathmandu', 'Balaju-3');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(179, 29, 'Nepal', 3, 'Lalitpur', 'Balkumari-15');

1 row created.
```

```
SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(185, 35, 'Nepal', 3, 'Lalitpur', 'Balkumari-16');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(186, 36, 'Nepal', 4, 'Pokhara', 'Lakeside-15');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(187, 37, 'Nepal', 6, 'Jhapa', 'Birtamod-2');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(188, 38, 'Nepal', 3, 'Lalitpur', 'Sanepa-7');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(189, 39, 'Nepal', 3, 'Lalitpur', 'Dhobighat-18');

1 row created.
```

```
SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(180, 30, 'Nepal', 4, 'Chitwan', 'Bharatpur-7');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(181, 31, 'Nepal', 6, 'Pokhara', 'Sarangkot-9');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(182, 32, 'Nepal', 4, 'Janakpur', 'Bardibas-11');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(183, 33, 'Nepal', 3, 'Lalitpur', 'Patan-2');

1 row created.

SQL> INSERT INTO sCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(184, 34, 'Nepal', 3, 'Lalitpur', 'Jawalakhel-8');

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 29 Insert into sCountry

Inserting values in tCountry table:

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(121, 71, ‘USA’, 3, ‘Los Angels’, ‘Maroon-15’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(122, 72, ‘Nepal’, 3, ‘Kathmandu’, ‘Koteshwor-11’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(123, 73, ‘Australia’, 2, ‘Sydney’, ‘Come-8’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(124, 74, ‘Nepal’, 5, ‘Heatauda’, ‘Cement-6’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(125, 75, ‘India’, 5, ‘Mumbai’, ‘Delhi-5’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(126, 76, ‘Nepal’, 6, ‘Pokhara’, ‘Gantok-5’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(127, 77, ‘Mexico’, 3, ‘Sinaloa’, ‘Cartel-3’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(128, 78, ‘Nepal’, 3, ‘Kathmandu’, ‘Balaju-4’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(129, 79, ‘Nepal’, 3, ‘Lalitpur’, ‘Balkumari-16’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(135, 85, ‘Nepal’, 3, ‘Lalitpur’, ‘Balkumari-6’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(136, 86, ‘Nepal’, 4, ‘Pokhara’, ‘Lakeside-3’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(137, 87, ‘Nepal’, 6, ‘Jhapa’, ‘Birtamod-8’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(138, 88, ‘Nepal’, 5, ‘Chitwan’, ‘Tadi-16’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(139, 89, ‘Nepal’, 3, ‘Lalitpur’, ‘Sanepa-2’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(140, 90, ‘Nepal’, 3, ‘Lalitpur’, ‘Jwalakhel-15’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(130, 80, ‘Nepal’, 4, ‘Chitwan’, ‘Bharatpur-7’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(131, 81, ‘Nepal’, 6, ‘Pokhara’, ‘Sarangkot-10’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(132, 82, ‘Nepal’, 4, ‘Janakpur’, ‘Bardibas-10’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(133, 83, ‘Nepal’, 3, ‘Lalitpur’, ‘Patan-4’);

INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)VALUES(134, 84, ‘Nepal’, 3, ‘Lalitpur’, ‘Jawalakhel-9’);

Run SQL Command Line

```
SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(121, 71, 'USA', 3, 'Los Angels', 'Maroon-15');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(122, 72, 'Nepal', 3, 'Kathmandu', 'Koteshwor-11');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(123, 73, 'Australia', 2, 'Sydney', 'Come-8');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(124, 74, 'Nepal', 5, 'Heatauda', 'Cement-6');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(125, 75, 'India', 5, 'Mumbai', 'Delhi-5');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(126, 76, 'Nepal', 6, 'Pokhara', 'Gantok-5');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(127, 77, 'Mexico', 3, 'Sinaloa', 'Cartel-3');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(128, 78, 'Nepal', 3, 'Kathmandu', 'Balaju-4');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(129, 79, 'Nepal', 3, 'Lalitpur', 'Balkumari-16');

1 row created.
```

```
SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(135, 85, 'Nepal', 3, 'Lalitpur', 'Balkumari-6');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(136, 86, 'Nepal', 4, 'Pokhara', 'Lakeside-3');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(137, 87, 'Nepal', 6, 'Jhapa', 'Birtamod-8');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(138, 88, 'Nepal', 5, 'Chitwan', 'Tadi-16');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(139, 89, 'Nepal', 3, 'Lalitpur', 'Sanepa-2');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(140, 90, 'Nepal', 3, 'Lalitpur', 'Jwalakhel-15');

1 row created.
```

```
SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(130, 80, 'Nepal', 4, 'Chitwan', 'Bharatpur-7');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(131, 81, 'Nepal', 6, 'Pokhara', 'Sarangkot-10');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(132, 82, 'Nepal', 4, 'Janakpur', 'Bardibas-10');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(133, 83, 'Nepal', 3, 'Lalitpur', 'Patan-4');

1 row created.

SQL> INSERT INTO tCountry(Country_ID, House_Number, Country, Province, City, Street)
  2  VALUES(134, 84, 'Nepal', 3, 'Lalitpur', 'Jawalakhel-9');

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 30 Insert into tCountry

Inserting values in sDOB table:

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(11, ‘01-Jan-2000’, 20);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(12, ‘01-Jan-2001’, 19);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(13, ‘10-Jan-1999’, 21);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(14, ‘09-Sep-2000’, 20);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(15, ‘07-Dec-2001’, 19);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(16, ‘05-Oct-2000’, 20);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(25, ‘15-Oct-2000’, 20);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(26, ‘05-Jan-2001’, 19);

```
INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(27, '22-Oct-1999', 21);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(28, '05-Oct-1998', 22);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(29, '08-May-2000', 20);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(17, '25-Feb-1998', 22);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(18, '01-Nov-2001', 19);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(19, '25-Jan-1999', 21);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(20, '04-Sep-2002', 18);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(21, '01-Jan-2000', 20);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(22, '19-Aug-2001', 19);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(23, '24-Nov-1999', 21);

INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(24, '01-Dec-2002', 18);
```

Run SQL Command Line

```
SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(11, '01-Jan-2000', 20);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(12, '01-Jan-2001', 19);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(13, '10-Jan-1999', 21);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(14, '09-Sep-2000', 20);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(15, '07-Dec-2001', 19);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(16, '05-Oct-2000', 20);

1 row created.
```

```
SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(25, '15-Oct-2000', 20);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(26, '05-Jan-2001', 19);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(27, '22-Oct-1999', 21);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(28, '05-Oct-1998', 22);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(29, '08-May-2000', 20);

1 row created.
```

```
SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(17, '25-Feb-1998', 22);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(18, '01-Nov-2001', 19);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(19, '25-Jan-1999', 21);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(20, '04-Sep-2002', 18);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(21, '01-Jan-2000', 20);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(22, '19-Aug-2001', 19);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(23, '24-Nov-1999', 21);

1 row created.

SQL> INSERT INTO sDOB(DOB_ID, DOB, Age)VALUES(24, '01-Dec-2002', 18);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 31 Insert into sDOB

Inserting values in tDOB table:

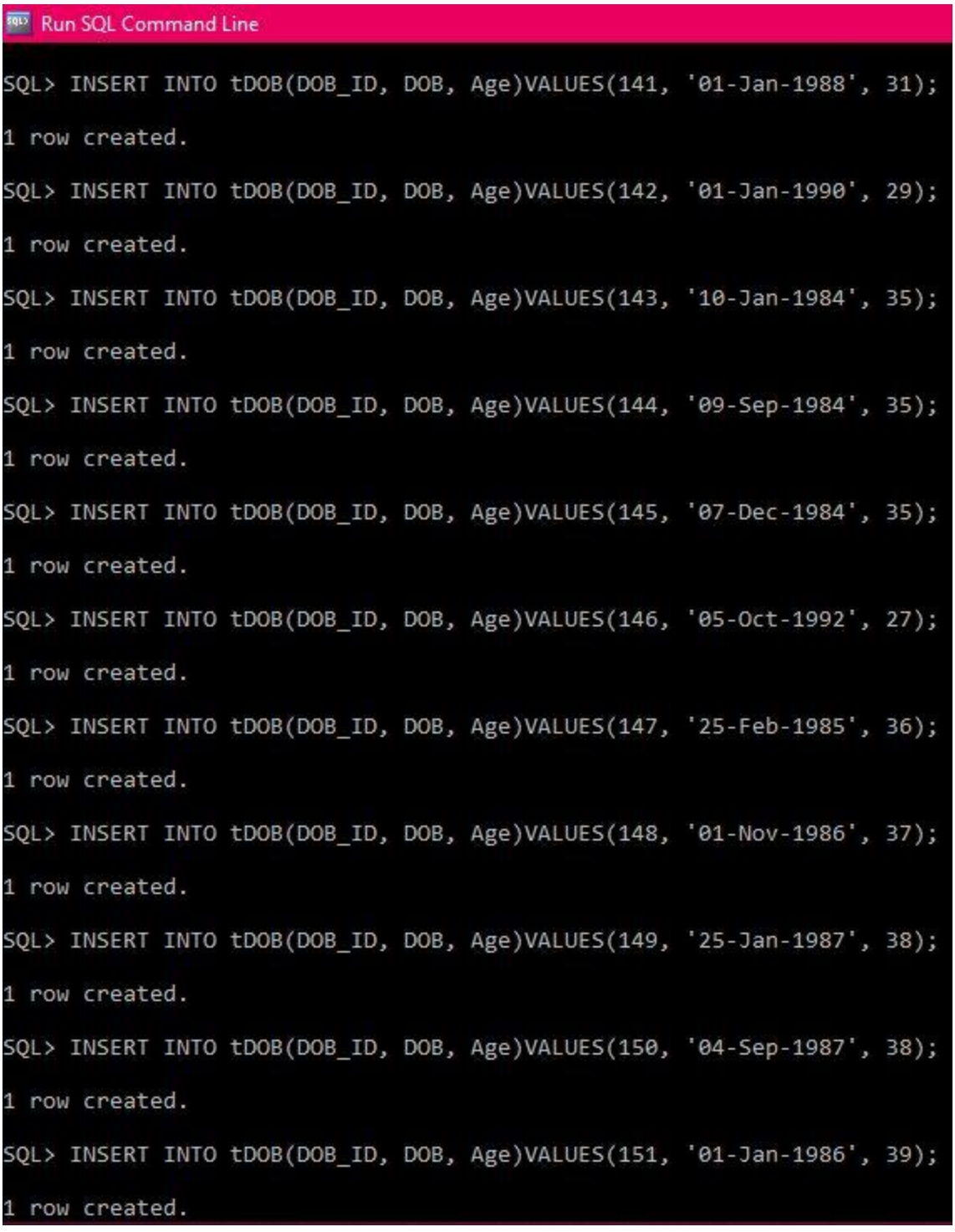
INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(141, ‘01-Jan-1988’, 31);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(142, ‘01-Jan-1990’, 29);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(143, ‘10-Jan-1984’, 35);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(144, ‘09-Sep-1984’, 35);

```
INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(145, '07-Dec-1984', 35);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(146, '05-Oct-1992', 27);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(147, '25-Feb-1985', 36);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(148, '01-Nov-1986', 37);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(149, '25-Jan-1987', 38);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(150, '04-Sep-1987', 38);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(151, '01-Jan-1986', 39);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(155, '01-Sep-1985', 36);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(156, '12-Oct-1983', 41);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(157, '13-Nov-1982', 42);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(158, '01-Dec-1984', 40);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(159, '06-Feb-1985', 39);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(160, '09-Jan-1982', 42);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(152, '19-Aug-1985', 40);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(153, '24-Nov-1985', 40);

INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(154, '01-Dec-1986', 39);
```


Run SQL Command Line
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(141, '01-Jan-1988', 31);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(142, '01-Jan-1990', 29);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(143, '10-Jan-1984', 35);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(144, '09-Sep-1984', 35);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(145, '07-Dec-1984', 35);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(146, '05-Oct-1992', 27);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(147, '25-Feb-1985', 36);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(148, '01-Nov-1986', 37);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(149, '25-Jan-1987', 38);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(150, '04-Sep-1987', 38);
1 row created.
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(151, '01-Jan-1986', 39);
1 row created.

```
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(155, '01-Sep-1985', 36);

1 row created.

SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(156, '12-Oct-1983', 41);

1 row created.

SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(157, '13-Nov-1982', 42);

1 row created.

SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(158, '01-Dec-1984', 40);

1 row created.

SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(159, '06-Feb-1985', 39);

1 row created.

SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(160, '09-Jan-1982', 42);

1 row created.
```

```
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(152, '19-Aug-1985', 40);

1 row created.

SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(153, '24-Nov-1985', 40);

1 row created.

SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(154, ‘01-Dec-1986’, 39);
ERROR:
ORA-01756: quoted string not properly terminated


SQL>
SQL> INSERT INTO tDOB(DOB_ID, DOB, Age)VALUES(154, '01-Dec-1986', 39);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 32 Insert into tDOB

Inserting values in sAddress table:

```
INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(301,171,11,9812398745,'siddhartha@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(302,172,12,9813458769,'rihan@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(303,173,13,9817658903,'chirag@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(304,174,14,9812674583,'priyanka@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(305,175,15,9659872153,'katrina@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(306,176,16,9812365984,'dwayne@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(307,177,17,9812323458,'simon@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(308,178,18,9812398456,'rohan@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(309,179,19,98167842985,'bishnu@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(310,180,20,9822234445,'ashish@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(311,181,21,98144466783,'aayusha@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(315,185,25,98134466783,'bikram@gmail.com');
```

```
INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(316,186,26,98142466783,'piyush@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(317,187,27,98144766783,'prajeet@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(318,188,28,98144486783,'azan@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(319,189,29,98144469783,'smriti@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(312,182,22,9812455567,'shiv@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(313,183,23,9811199986,'sahayog@gmail.com');

INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(314,184,24,9814333565,'shrijana@gmail.com');
```

```
Run SQL Command Line
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(301,171,11,9812398745,'siddhartha@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(302,172,12,9813458769,'rihan@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(303,173,13,9817658903,'chirag@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(304,174,14,9812674583,'priyanka@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(305,175,15,9659872153,'katrina@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(306,176,16,9812365984,'dwayne@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(307,177,17,9812323458,'simon@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(308,178,18,9812398456,'rohan@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(309,179,19,98167842985,'bishnu@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(310,180,20,9822234445,'ashish@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(311,181,21,98144466783,'aayusha@gmail.com');
1 row created.
```

```
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(315,185,25,98134466783,'bikram@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(316,186,26,98142466783,'piyush@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(317,187,27,98144766783,'prajeet@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(318,188,28,98144486783,'azan@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(319,189,29,98144469783,'smriti@gmail.com');
1 row created.
```

```
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(312,182,22,9812455567,'shiv@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(313,183,23,9811199986,'sahayog@gmail.com');
1 row created.
SQL> INSERT INTO sAddress(AddressStd_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(314,184,24,9814333565,'shrijana@gmail.com');
1 row created.
SQL> COMMIT;
Commit complete.
```

Figure 33 Insert Into sAddress

Inserting values in tAddress table:

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(401,121,141,9812398745,'steve@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(402,122,142,9813458769,'sabin@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(403,123,143,9817658903,'mark@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(404,124,144,9812674583,'ramesh@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(405,125,145,9659872153,'jayesh@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(406,126,146,9812365984,'sabina@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(407,127,147,9812323458,'wiz@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(408,128,148,9812398456,'anup@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(409,129,149,98167842985,'ram@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(410,130,150,9822234445,'utsav@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(411,131,151,98144466783,'yubraj@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(415,135,155,98244466783,'sameer@gmail.com');

```
INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(416,136,156,98264466783,'piyush@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(417,137,157,98247466783,'ramesh@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(418,138,158,98244966783,'neha@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(419,139,159,98244406783,'rabin@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(420,140,160,98244466793,'simran@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(412,132,152,9812455567,'hari@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(413,133,153,9811199986,'bom@gmail.com');

INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number,
Email)VALUES(414,134,154,9814333565,'govind@gmail.com');
```

```
Run SQL Command Line
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(401,121,141,9812398745,'steve@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(402,122,142,9813458769,'sabin@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(403,123,143,9817658903,'mark@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(404,124,144,9812674583,'ramesh@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(405,125,145,9659872153,'jayesh@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(406,126,146,9812365984,'sabina@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(407,127,147,9812323458,'wiz@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(408,128,148,9812398456,'anup@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(409,129,149,98167842985,'ram@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(410,130,150,9822234445,'utsav@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(411,131,151,98144466783,'yubraj@gmail.com');
1 row created.
```

```
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(415,135,155,98244466783,'sameer@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(416,136,156,98264466783,'piyush@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(417,137,157,98247466783,'ramesh@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(418,138,158,98244966783,'neha@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(419,139,159,98244406783,'rabin@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(420,140,160,98244466793,'simran@gmail.com');
1 row created.
```

```
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(412,132,152,9812455567,'hari@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(413,133,153,9811199986,'bom@gmail.com');
1 row created.
SQL> INSERT INTO tAddress(AddressTec_ID, Country_ID, DOB_ID, Mobile_Number, Email)VALUES(414,134,154,9814333565,'govind@gmail.com');
1 row created.
SQL> COMMIT;
Commit complete.
```

Figure 34 Insert into tAddress

Inserting values in sAddress_Info table:

```
INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(301,101);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(302,102);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(303,103);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(304,104);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(305,105);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(306,106);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(307,107);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(308,108);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(309,109);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(310,110);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(311,111);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(315,115);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(316,116);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(317,117);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(318,118);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(319,119);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(312,112);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(313,113);

INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(314,114);
```

Run SQL Command Line

```
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(301,101);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(302,102);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(303,103);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(304,104);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(305,105);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(306,106);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(307,107);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(308,108);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(309,109);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(310,110);

1 row created.

SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(311,111);

1 row created.
```

```
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(315,115);
1 row created.
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(316,116);
1 row created.
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(317,117);
1 row created.
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(318,118);
1 row created.
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(319,119);
1 row created.
```

```
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(312,112);
1 row created.
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(313,113);
1 row created.
SQL> INSERT INTO sAddress_Info(AddressStd_ID, Student_ID)VALUES(314,114);
1 row created.
SQL> COMMIT;
Commit complete.
```

Figure 35 Insert into sAddress_Info

Inserting values in tAddress_Info table:

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(401,201);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(402,202);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(403,203);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(404,204);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(405,205);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(406,206);

```
INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(407,207);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(408,208);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(409,209);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(410,210);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(411,211);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(415,215);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(416,216);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(417,217);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(418,218);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(419,219);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(420,220);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(412,212);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(413,213);

INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(414,214);
```

Run SQL Command Line

```
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(401,201);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(402,202);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(403,203);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(404,204);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(405,205);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(406,206);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(407,207);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(408,208);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(409,209);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(410,210);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(411,211);
1 row created.
```

```
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(415,215);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(416,216);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(417,217);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(418,218);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(419,219);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(420,220);
1 row created.
```

```
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(412,212);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(413,213);
1 row created.
SQL> INSERT INTO tAddress_Info(AddressTec_ID, Teacher_ID)VALUES(414,214);
1 row created.
SQL> COMMIT;
Commit complete.
```

Figure 36 Insert into tAddress_Info

3.3. Final Table

'SELECT' command is implemented to provide the inserted data in Oracle SQL Plus.

Course Table:

SELECT * FROM Course;

```
SQL> SELECT * FROM Course;
COURSE_ID | COURSE_NAME     | COURSE_TYPE     | SPECIFICATION_NAME   | SPECIFICATION_FEES
--------- | --------------- | --------------- | -------------------- | ------------------
        1 | BIT             | Bachelor        | Computing            |             115000
        2 | BIT             | Bachelor        | Networking           |             115000
        3 | BIT             | Bachelor        | Multimedia           |             115000
        4 | MSc IT          | Master          | Security             |             215000
        5 | BBA             | Bachelor        | Marketing            |             110000
        6 | BBA             | Bachelor        | Finance              |             110000
        7 | MBA             | Master          | Accounting           |             210000

7 rows selected.
```

Figure 37 Select from Course

Student Table:

SELECT * FROM Student;

```
SQL> SELECT * FROM Student;

STUDENT_ID | NAME            | SEX        | JOINED_DA | STUDENT_ST | YEAR
---------- | --------------- | ---------- | --------- | ---------- | ----------
       116 | Piyush Sharma   | Male       | 02-JAN-19 | Inactive   | Second
       101 | Siddhartha Ghim | Male       | 01-JAN-19 | Active     | Second
       102 | Rihan Ojha      | Male       | 01-JAN-18 | Active     | Third
       103 | Chirag Timalsin | Male       | 01-JAN-18 | Inactive   | Third
       104 | Priyanka Chopra | Female     | 01-JAN-20 | Active     | First
       105 | Katrina Kaif    | Female     | 01-JAN-19 | Active     | Second
       106 | Dwayne Johnson  | Male       | 01-JAN-20 | Inactive   | First
       107 | Simon Wilson    | Male       | 01-JAN-18 | Inactive   | Third
       108 | Rohan Gurung    | Male       | 01-JAN-19 | Active     | Second
       109 | Bishnu Thapa    | Male       | 01-JAN-20 | Inactive   | First
       110 | Ashish Shrestha | Male       | 01-JAN-18 | Active     | Third
       111 | Aayusha Shresth | Female     | 01-JAN-20 | Active     | First
       112 | Shiv Thakur     | Male       | 01-JAN-19 | Inactive   | Second
       113 | Sahayog G.C.    | Male       | 01-JAN-18 | Inactive   | Third
       114 | Shrijana Rokaya | Female     | 01-JAN-19 | Active     | Second
       115 | Bikram Rathor   | Male       | 01-JAN-18 | Active     | Third
       117 | Prajeet Thakur  | Male       | 01-JAN-18 | Active     | Third
       118 | Azan Ahmad      | Male       | 01-FEB-20 | Inactive   | First
       119 | Smriti Thapa    | Female     | 12-JAN-18 | Active     | Third

19 rows selected.
```

Figure 38 Select from Student

Teacher Table:

SELECT * FROM Teacher;

TEACHER_ID	NAME	SEX	JOINED_DA	TEACHER_TYPE	TEACHER_ST	SALARY
215	Sameer Khadka	Male	01-SEP-18	Instructor	Active	62000
216	Piyush Khadka	Male	01-FEB-17	Instructor	Active	68000
217	Ramesh Karmacha	Male	01-SEP-17	Instructor	Inactive	51000
218	Neha Khadka	Female	12-SEP-19	Instructor	Active	64000
219	Rabin Bajrachar	Male	01-SEP-20	Instructor	Active	60000
220	Simran Sha	Female	01-NOV-18	Instructor	Inactive	65000
201	Steve Jobs	Male	01-FEB-00	Module Leader	Active	60000
202	Sabin Bharati	Male	01-JAN-05	Module Leader	Active	50000
203	Mark Zuckerburg	Male	01-DEC-99	Module Leader	Active	75000
204	Ramesh Thapa	Male	01-FEB-13	Module Leader	Active	48000
205	Jayesh Karmacha	Male	01-MAY-02	Module Leader	Active	40000
206	Sabina Tamrakar	Female	01-FEB-10	Module Leader	Active	52000
207	Wiz Khalif	Male	01-JUN-18	Module Leader	Active	55000
208	Anup Thakur	Male	01-JUN-18	Module Leader	Active	54000
209	Ram Nepal	Male	01-SEP-18	Module Leader	Active	20000
210	Utsav Rajput	Male	01-JUN-13	Module Leader	Active	15000
211	Yubraj Khadka	Male	01-SEP-05	Module Leader	Active	59000
212	Hari Ghimire	Male	01-JAN-20	Module Leader	Active	50000
213	Bom Bahadur	Male	01-DEC-19	Module Leader	Active	54000
214	Govind Chaudhar	Male	08-JUN-19	Module Leader	Active	50000

20 rows selected.

Figure 39 Select from Teacher

Module Table:

SELECT * FROM Module;

```
SQL> SELECT * FROM Module;
MODULE_ID | CLASS          |   MARKS
--------- | -------------- | -------
       51 | Kanchanjanga   |      70
       52 | Bristol        |      82
       53 | Kingstone      |      42
       54 | Pokhara        |      37
       55 | Buckingham     |      82

5 rows selected.
```

Figure 40 Select from Module

Student_Info Table:

SELECT * FROM Student_Info;

```
SQL> SELECT * FROM Student_Info;

STUDENT_ID |  COURSE_ID
---------- | ----------
       101 |          1
       102 |          2
       103 |          3
       104 |          4
       105 |          5
       106 |          6
       107 |          7
       108 |          1
       109 |          3
       110 |          2
       111 |          5
       112 |          4
       113 |          7
       114 |          6
       115 |          2
       116 |          2
       117 |          2
       118 |          2
       119 |          2

19 rows selected.
```

Figure 41 Select from Student_Info

Teacher_Info Table:

SELECT * FROM Teacher_Info;

```
SQL> SELECT * FROM Teacher_Info;

TEACHER_ID |  COURSE_ID
---------- | ----------
       201 |          1
       201 |          3
       202 |          7
       203 |          1
       204 |          5
       204 |          6
       205 |          1
       205 |          2
       206 |          3
       207 |          7
       208 |          2
       209 |          4
       210 |          2
       211 |          4
       212 |          5
       213 |          6
       214 |          3
       215 |          6
       216 |          4
       217 |          5
       218 |          6
       219 |          6
       220 |          6

23 rows selected.
```

Figure 42 Select from Teacher_Info

Module_Details Table:

SELECT * FROM Module_Details;

```
SQL> SELECT * FROM Module_Details;
MODULE_ID | COURSE_ID | MODULE_NAME                          | MODULE_LEADER
--------- | --------- | ------------------------------------ | ------------------
       51 |         1 | Database                             | Mark Zuckerburg
       52 |         1 | Programming                          | Steve Jobs
       53 |         1 | Information System                   | Jayesh Karmacharya
       54 |         2 | Information System                   | Jayesh Karmacharya
       55 |         2 | Networking Concepts                  | Utsav Rajput
       51 |         2 | Communications Engineering           | Anup Thakur
       51 |         3 | Digital Design                       | Govind Chaudhary
       53 |         3 | Programming                          | Steve Jobs
       54 |         3 | Drawing and Character Design         | Sabina Tamrakar
       51 |         4 | Cyber Security                       | Ram Nepal
       55 |         4 | Work Related Learning                | Yubraj Khadka
       51 |         5 | Economics and Society                | Ramesh Thapa
       53 |         5 | The Corporate Environment            | Hari Ghimire
       51 |         6 | Economics and Society                | Ramesh Thapa
       55 |         6 | Choices and Changes                  | Bom Bahadur
       51 |         7 | Leadership and Strategic Management  | Sabin Bharati
       52 |         7 | Management Learning                  | Wiz Khalif

17 rows selected.
```

Figure 43 Select from Module_Details

sResidence Table:

SELECT * FROM sResidence;

```
SQL> SELECT * FROM sResidence;

HOUSE_NUMBER | PHONE_NUMBER | FAX_NUMBER
------------ | ------------ | ----------
          21 |              |
          22 |   9864567899 |       6120
          23 |              |       6121
          24 |   9800818055 |       6122
          25 |   9844403874 |
          26 |   9844040274 |       6123
          27 |              |       6124
          28 |   9818347856 |
          29 |              |       6125
          30 |   9876567890 |       6126
          31 |   9876345672 |       6127
          32 |              |       6128
          33 |   9856782234 |
          34 |              |       6129
          35 |   9836363444 |       6130
          36 |   9836363474 |       6131
          37 |   9836363244 |       6132
          38 |   9838363444 |       6133
          39 |   9826363444 |       6134
          40 |   9836373444 |       6135

20 rows selected.
```

Figure 44 Select from sResidence

tResidence Table:

SELECT * FROM tResidence;

```
SQL> SELECT * FROM tResidence;

HOUSE_NUMBER | PHONE_NUMBER | FAX_NUMBER
------------ | ------------ | ----------
          71 |              |
          72 |   9823456890 |       6130
          73 |              |       6131
          74 |   9809876543 |       6132
          75 |   9856789012 |
          76 |   9867890123 |       6133
          77 |              |       6134
          78 |   9878901234 |
          79 |              |       6135
          80 |   9872345456 |       6136
          81 |   9876654567 |       6137
          82 |              |       6138
          83 |   9856798765 |
          84 |              |       6139
          85 |   9823456290 |       6140
          86 |   9823426890 |       6141
          87 |   9833456890 |       6142
          88 |   9823556890 |       6143
          89 |   9823455890 |       6144
          90 |   9823486890 |       6145

20 rows selected.
```

Figure 45 Select from tResidence

sCountry Table:

SELECT * FROM sCountry;

```
SQL> SELECT * FROM sCountry;

COUNTRY_ID | HOUSE_NUMBER | COUNTRY    |   PROVINCE | CITY       | STREET
---------- | ------------ | ---------- | ---------- | ---------- | ---------------
-------------------------------------------------------------------------------
       171 |           21 | Nepal      |          3 | Lalitpur   | Satdobato-15
       172 |           22 | Nepal      |          3 | Kathmandu  | Koteshwor-10
       173 |           23 | Nepal      |          2 | Sindhuli   | Kamalamai-8
       174 |           24 | India      |          5 | Mumbai     | Jaypur-6
       175 |           25 | India      |          5 | Mumbai     | Delhi-15
       176 |           26 | USA        |          8 | Hollywood  | Gantok-5
       177 |           27 | Mexico     |          3 | Sinaloa    | Cartel-2
       178 |           28 | Nepal      |          3 | Kathmandu  | Balaju-3
       179 |           29 | Nepal      |          3 | Lalitpur   | Balkumari-15
       180 |           30 | Nepal      |          4 | Chitwan    | Bharatpur-7
       181 |           31 | Nepal      |          6 | Pokhara    | Sarangkot-9
       182 |           32 | Nepal      |          4 | Janakpur   | Bardibas-11
       183 |           33 | Nepal      |          3 | Lalitpur   | Patan-2
       184 |           34 | Nepal      |          3 | Lalitpur   | Jawalakhel-8
       185 |           35 | Nepal      |          3 | Lalitpur   | Balkumari-16
       186 |           36 | Nepal      |          4 | Pokhara    | Lakeside-15
       187 |           37 | Nepal      |          6 | Jhapa      | Birtamod-2
       188 |           38 | Nepal      |          3 | Lalitpur   | Sanepa-7
       189 |           39 | Nepal      |          3 | Lalitpur   | Dhobighat-18

19 rows selected.
```

Figure 46 Select from sCountry

tCountry Table:

SELECT * FROM tCountry;

```
SQL> SELECT * FROM tCountry;

COUNTRY_ID | HOUSE_NUMBER | COUNTRY    |   PROVINCE | CITY       | STREET
---------- | ------------ | ---------- | ---------- | ---------- | ----------------
-------------------------------------------------------------------------------------
       121 |           71 | USA        |          3 | Los Angels | Maroon-15
       122 |           72 | Nepal      |          3 | Kathmandu  | Koteshwor-11
       123 |           73 | Australia  |          2 | Sydney     | Come-8
       124 |           74 | Nepal      |          5 | Heatauda   | Cement-6
       125 |           75 | India      |          5 | Mumbai     | Delhi-5
       126 |           76 | Nepal      |          6 | Pokhara    | Gantok-5
       127 |           77 | Mexico     |          3 | Sinaloa    | Cartel-3
       128 |           78 | Nepal      |          3 | Kathmandu  | Balaju-4
       129 |           79 | Nepal      |          3 | Lalitpur   | Balkumari-16
       130 |           80 | Nepal      |          4 | Chitwan    | Bharatpur-7
       131 |           81 | Nepal      |          6 | Pokhara    | Sarangkot-10
       132 |           82 | Nepal      |          4 | Janakpur   | Bardibas-10
       133 |           83 | Nepal      |          3 | Lalitpur   | Patan-4
       134 |           84 | Nepal      |          3 | Lalitpur   | Jawalakhel-9
       135 |           85 | Nepal      |          3 | Lalitpur   | Balkumari-6
       136 |           86 | Nepal      |          4 | Pokhara    | Lakeside-3
       137 |           87 | Nepal      |          6 | Jhapa      | Birtamod-8
       138 |           88 | Nepal      |          5 | Chitwan    | Tadi-16
       139 |           89 | Nepal      |          3 | Lalitpur   | Sanepa-2
       140 |           90 | Nepal      |          3 | Lalitpur   | Jwalakhel-15

20 rows selected.
```

Figure 47 Select from tCountry

sDOB Table:

SELECT * FROM sDOB;

```
SQL> SELECT * FROM sDOB;

    DOB_ID | DOB       |        AGE
---------- | --------- | ----------
        11 | 01-JAN-00 |         20
        12 | 01-JAN-01 |         19
        13 | 10-JAN-99 |         21
        14 | 09-SEP-00 |         20
        15 | 07-DEC-01 |         19
        16 | 05-OCT-00 |         20
        17 | 25-FEB-98 |         22
        18 | 01-NOV-01 |         19
        19 | 25-JAN-99 |         21
        20 | 04-SEP-02 |         18
        21 | 01-JAN-00 |         20
        22 | 19-AUG-01 |         19
        23 | 24-NOV-99 |         21
        24 | 01-DEC-02 |         18
        25 | 15-OCT-00 |         20
        26 | 05-JAN-01 |         19
        27 | 22-OCT-99 |         21
        28 | 05-OCT-98 |         22
        29 | 08-MAY-00 |         20

19 rows selected.
```

Figure 48 Select from sDOB

tDOB Table:

SELECT * FROM tDOB;

```
SQL> SELECT * FROM tDOB;

    DOB_ID | DOB       |        AGE
---------- | --------- | ----------
       141 | 01-JAN-88 |         31
       142 | 01-JAN-90 |         29
       143 | 10-JAN-84 |         35
       144 | 09-SEP-84 |         35
       145 | 07-DEC-84 |         35
       146 | 05-OCT-92 |         27
       147 | 25-FEB-85 |         36
       148 | 01-NOV-86 |         37
       149 | 25-JAN-87 |         38
       150 | 04-SEP-87 |         38
       151 | 01-JAN-86 |         39
       152 | 19-AUG-85 |         40
       153 | 24-NOV-85 |         40
       154 | 01-DEC-86 |         39
       155 | 01-SEP-85 |         36
       156 | 12-OCT-83 |         41
       157 | 13-NOV-82 |         42
       158 | 01-DEC-84 |         40
       159 | 06-FEB-85 |         39
       160 | 09-JAN-82 |         42

20 rows selected.
```

Figure 49 Select from tDOB

sAddress Table:

SELECT * FROM sAddress;

```
SQL> SELECT * FROM sAddress;

ADDRESSSTD_ID | COUNTRY_ID |     DOB_ID | MOBILE_NUMBER | EMAIL
------------- | ---------- | ---------- | ------------- | ----------------------
------------------------------------------------------------------------------------
          301 |        171 |         11 |    9812398745 | siddhartha@gmail.com
          302 |        172 |         12 |    9813458769 | rihan@gmail.com
          303 |        173 |         13 |    9817658903 | chirag@gmail.com
          304 |        174 |         14 |    9812674583 | priyanka@gmail.com
          305 |        175 |         15 |    9659872153 | katrina@gmail.com
          306 |        176 |         16 |    9812365984 | dwayne@gmail.com
          307 |        177 |         17 |    9812323458 | simon@gmail.com
          308 |        178 |         18 |    9812398456 | rohan@gmail.com
          309 |        179 |         19 |    9.8168E+10 | bishnu@gmail.com
          310 |        180 |         20 |    9822234445 | ashish@gmail.com
          311 |        181 |         21 |    9.8144E+10 | aayusha@gmail.com
          312 |        182 |         22 |    9812455567 | shiv@gmail.com
          313 |        183 |         23 |    9811199986 | sahayog@gmail.com
          314 |        184 |         24 |    9814333565 | shrijana@gmail.com
          315 |        185 |         25 |    9.8134E+10 | bikram@gmail.com
          316 |        186 |         26 |    9.8142E+10 | piyush@gmail.com
          317 |        187 |         27 |    9.8145E+10 | prajeet@gmail.com
          318 |        188 |         28 |    9.8144E+10 | azan@gmail.com
          319 |        189 |         29 |    9.8144E+10 | smriti@gmail.com

19 rows selected.
```

Figure 50 Select from sAddress

tAddress Table:

SELECT * FROM tAddress;

```
SQL> SELECT * FROM tAddress;

ADDRESSTEC_ID | COUNTRY_ID |     DOB_ID | MOBILE_NUMBER | EMAIL
------------- | ---------- | ---------- | ------------- | --------------------
------------------------------------------------------------------------------

          401 |        121 |        141 |    9812398745 | steve@gmail.com
          402 |        122 |        142 |    9813458769 | sabin@gmail.com
          403 |        123 |        143 |    9817658903 | mark@gmail.com
          404 |        124 |        144 |    9812674583 | ramesh@gmail.com
          405 |        125 |        145 |    9659872153 | jayesh@gmail.com
          406 |        126 |        146 |    9812365984 | sabina@gmail.com
          407 |        127 |        147 |    9812323458 | wiz@gmail.com
          408 |        128 |        148 |    9812398456 | anup@gmail.com
          409 |        129 |        149 |    9.8168E+10 | ram@gmail.com
          410 |        130 |        150 |    9822234445 | utsav@gmail.com
          411 |        131 |        151 |    9.8144E+10 | yubraj@gmail.com
          412 |        132 |        152 |    9812455567 | hari@gmail.com
          413 |        133 |        153 |    9811199986 | bom@gmail.com
          414 |        134 |        154 |    9814333565 | govind@gmail.com
          415 |        135 |        155 |    9.8244E+10 | sameer@gmail.com
          416 |        136 |        156 |    9.8264E+10 | piyush@gmail.com
          417 |        137 |        157 |    9.8247E+10 | ramesh@gmail.com
          418 |        138 |        158 |    9.8245E+10 | neha@gmail.com
          419 |        139 |        159 |    9.8244E+10 | rabin@gmail.com
          420 |        140 |        160 |    9.8244E+10 | simran@gmail.com

20 rows selected.
```

Figure 51 Select from tAddress

sAddress_Info Table:

SELECT * FROM sAddress_Info;

```
SQL> SELECT * FROM sAddress_Info;

ADDRESSSTD_ID | STUDENT_ID
------------- | ----------
          301 |        101
          302 |        102
          303 |        103
          304 |        104
          305 |        105
          306 |        106
          307 |        107
          308 |        108
          309 |        109
          310 |        110
          311 |        111
          312 |        112
          313 |        113
          314 |        114
          315 |        115
          316 |        116
          317 |        117
          318 |        118
          319 |        119

19 rows selected.
```

Figure 52 Select from sAddress_Info

tAddress_Info Table:

SELECT * FROM tAddress_Info;

```
SQL> SELECT * FROM tAddress_Info;

ADDRESSTEC_ID | TEACHER_ID
------------- | ----------
          401 |        201
          402 |        202
          403 |        203
          404 |        204
          405 |        205
          406 |        206
          407 |        207
          408 |        208
          409 |        209
          410 |        210
          411 |        211
          412 |        212
          413 |        213
          414 |        214
          415 |        215
          416 |        216
          417 |        217
          418 |        218
          419 |        219
          420 |        220

20 rows selected.
```

Figure 53 Select tAddress_Info

Chapter 4 – Information and Transaction Queries

4.1. Information Queries

i.) List all the student with all their address with their phone number?

➔ SELECT Student.Name,sCountry.Country, sCountry.Province, sCountry.City, sCountry.Street, sCountry.House_Number, sResidence.Phone_Number, sResidence.Fax_Number

FROM Student

JOIN sAddress_Info ON Student.Student_ID=sAddress_Info.Student_ID

JOIN sAddress ON sAddress_Info.AddressStd_ID=sAddress.AddressStd_ID

```
JOIN sCountry ON sAddress.Country_ID=sCountry.Country_ID

JOIN sResidence ON sCountry.House_Number=sResidence.House_Number;
```

```
SQL> SELECT Student.Name,sCountry.Country, sCountry.Province, sCountry.City, sCountry.Street, sCountry.House_Number,
  2  sResidence.Phone_Number, sResidence.Fax_Number
  3  FROM Student
  4  JOIN sAddress_Info ON Student.Student_ID=sAddress_Info.Student_ID
  5  JOIN sAddress ON sAddress_Info.AddressStd_ID=sAddress.AddressStd_ID
  6  JOIN sCountry ON sAddress.Country_ID=sCountry.Country_ID
  7  JOIN sResidence ON sCountry.House_Number=sResidence.House_Number;

NAME               | COUNTRY         |   PROVINCE | CITY            | STREET          | HOUSE_NUMBER | PHONE_NUMBER | FAX_NUMBER
-------------------- | --------------- | ---------- | --------------- | --------------- | ------------ | ------------ | ----------
Siddhartha Ghimire | Nepal           |          3 | Lalitpur        | Satdobato-15    |           21 |              |
Rihan Ojha         | Nepal           |          3 | Kathmandu       | Koteshwor-10    |           22 |   9864567899 |       6120
Chirag Timalsina   | Nepal           |          2 | Sindhuli        | Kamalamai-8     |           23 |              |       6121
Priyanka Chopra    | India           |          5 | Mumbai          | Jaypur-6        |           24 |   9800818055 |       6122
Katrina Kaif       | India           |          5 | Mumbai          | Delhi-15        |           25 |   9844403874 |
Dwayne Johnson     | USA             |          8 | Hollywood       | Gantok-5        |           26 |   9844040274 |       6123
Simon Wilson       | Mexico          |          3 | Sinaloa         | Cartel-2        |           27 |              |       6124
Rohan Gurung       | Nepal           |          3 | Kathmandu       | Balaju-3        |           28 |   9818347856 |
Bishnu Thapa       | Nepal           |          3 | Lalitpur        | Balkumari-15    |           29 |              |       6125
Ashish Shrestha    | Nepal           |          4 | Chitwan         | Bharatpur-7     |           30 |   9876567890 |       6126
Aayusha Shrestha   | Nepal           |          6 | Pokhara         | Sarangkot-9     |           31 |   9876345672 |       6127
Shiv Thakur        | Nepal           |          4 | Janakpur        | Bardibas-11     |           32 |              |       6128
Sahayog G.C.       | Nepal           |          3 | Lalitpur        | Patan-2         |           33 |   9856782234 |
Shrijana Rokaya    | Nepal           |          3 | Lalitpur        | Jawalakhel-8    |           34 |              |       6129
Bikram Rathor      | Nepal           |          3 | Lalitpur        | Balkumari-16    |           35 |   9836363444 |       6130
Piyush Sharma      | Nepal           |          4 | Pokhara         | Lakeside-15     |           36 |   9836363474 |       6131
Prajeet Thakur     | Nepal           |          6 | Jhapa           | Birtamod-2      |           37 |   9836363244 |       6132
Azan Ahmad         | Nepal           |          3 | Lalitpur        | Sanepa-7        |           38 |   9838363444 |       6133
Smriti Thapa       | Nepal           |          3 | Lalitpur        | Dhobighat-18    |           39 |   9826363444 |       6134

19 rows selected.
```

Figure 54 Information query 1

In this query Student Name, Country, Province, City, Street, House Number, Phone Number, Fax Number where selected from Student, sCountry and sResidence tables using JOIN to find out the total students with all their address along with their phone number.

ii.) List all the Modules which are taught by more then one Instructor?

➔SELECT Module_Details.Module_ID,Course.Specification_Name,
Module_Details.Module_Name, COUNT (Teacher.Teacher_ID) AS "Number OF Teachers"
FROM Course
JOIN Module_Details ON Course.Course_ID=Module_Details.Course_ID
JOIN Teacher_Info ON Course.Course_ID=Teacher_Info.Course_ID
JOIN Teacher ON Teacher_Info.Teacher_ID=Teacher.Teacher_ID
GROUP BY Module_Details.Module_ID,Course.Specification_Name,
Module_Details.Module_Name
HAVING COUNT (Teacher.Teacher_ID)>1
ORDER BY Course.Specification_Name;

```
SQL> SELECT Module_Details.Module_ID,Course.Specification_Name,Module_Details.Module_Name,
  2  COUNT(Teacher.Teacher_ID)AS "Number OF Teachers" FROM Course
  3  JOIN Module_Details ON Course.Course_ID=Module_Details.Course_ID
  4  JOIN Teacher_Info ON Course.Course_ID=Teacher_Info.Course_ID
  5  JOIN Teacher ON Teacher_Info.Teacher_ID=Teacher.Teacher_ID
  6  GROUP BY Module_Details.Module_ID,Course.Specification_Name, Module_Details.Module_Name
  7  HAVING COUNT(Teacher.Teacher_ID)>1
  8  ORDER BY Course.Specification_Name;

 MODULE_ID | SPECIFICATION_NAME   | MODULE_NAME              | Number OF Teachers
---------- | -------------------- | ------------------------ | ------------------
        51 | Accounting           | Leadership and Strategic |                  2
        52 | Accounting           | Management Learning      |                  2
        51 | Computing            | Database                 |                  3
        52 | Computing            | Programming              |                  3
        53 | Computing            | Information System       |                  3
        51 | Finance              | Economics and Society    |                  6
        55 | Finance              | Choices and Changes      |                  6
        51 | Marketing            | Economics and Society    |                  3
        53 | Marketing            | The Corporate Environment|                  3
        51 | Multimedia           | Digital Design           |                  3
        53 | Multimedia           | Programming              |                  3
        54 | Multimedia           | Drawing and Character Des|                  3
        51 | Networking           | Communications Engineerin|                  3
        54 | Networking           | Information System       |                  3
        55 | Networking           | Networking Concepts      |                  3
        51 | Security             | Cyber Security           |                  3
        55 | Security             | Work Related Learning    |                  3

17 rows selected.
```

Figure 55 Information query 2

In this query Teacher ID, Module ID, Module Name, Specification Name attributes were selected from Teacher, Course and Module Details tables using JOIN, COUNT, GROUP BY, HAVING, ORDER BY functions to find out the total modules which are taught by more then one Teachers.

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

➔SELECT Name, Salary FROM Teacher WHERE Name LIKE ‘S%’ AND Salary > 50000;

```
SQL> SELECT Name, Salary FROM Teacher WHERE Name LIKE 'S%' AND Salary > 50000;

NAME                 |     SALARY
-------------------- | ----------
Sameer Khadka        |      62000
Simran Sha           |      65000
Steve Jobs           |      60000
Sabina Tamrakar      |      52000

4 rows selected.
```

Figure 56 Information query 3

In this query Teacher Name and Salary were selected from Teacher Table whose Name starts with s and Salary is greater then 50,000 using WHERE clause.

iv.) List the Modules comes under the Multimedia Specification?

➔ SELECT Course.Specification_Name, Module_Details.Module_Name FROM Course JOIN Module_Details ON Course.Course_ID=Module_Details.Course_ID WHERE Specification_Name=’Multimedia’;

```
Run SQL Command Line

SQL> SELECT Course.Specification_Name, Module_Details.Module_Name FROM Course
  2  JOIN Module_Details ON Course.Course_ID=Module_Details.Course_ID
  3  WHERE Specification_Name='Multimedia';

SPECIFICATION_NAME   | MODULE_NAME
-------------------- | ----------------------------------
Multimedia           | Digital Design
Multimedia           | Programming
Multimedia           | Drawing and Character Design

3 rows selected.
```

Figure 57 Information query 4

In this query Specification Name and Module Name attributes were selected from Course and Module Details tables using JOIN and WHERE clause to find out the total modules that falls under Multimedia Specification.

v.) List the name of head of modules with the list of phone number?

➔SELECT Teacher.Name, Teacher.Teacher_Type, tResidence.Phone_Number

FROM Teacher

JOIN tAddress_Info ON Teacher.Teacher_ID = tAddress_Info.Teacher_ID

JOIN tAddress ON tAddress_Info.AddressTec_ID = tAddress.AddressTec_ID

JOIN tCountry ON tAddress.Country_ID = tCountry.Country_ID

JOIN tResidence ON tCountry.House_Number = tResidence.House_Number

WHERE Teacher_Type=’Module Leader’;

```
SQL> SELECT Teacher.Name, Teacher.Teacher_Type, tResidence.Phone_Number
  2  FROM Teacher
  3  JOIN tAddress_Info ON Teacher.Teacher_ID = tAddress_Info.Teacher_ID
  4  JOIN tAddress ON tAddress_Info.AddressTec_ID = tAddress.AddressTec_ID
  5  JOIN tCountry ON tAddress.Country_ID = tCountry.Country_ID
  6  JOIN tResidence ON tCountry.House_Number = tResidence.House_Number
  7  WHERE Teacher_Type='Module Leader';

NAME               | TEACHER_TYPE         | PHONE_NUMBER
-------------------- | -------------------- | ------------
Steve Jobs         | Module Leader        |
Sabin Bharati      | Module Leader        |   9823456890
Mark Zuckerburg    | Module Leader        |
Ramesh Thapa       | Module Leader        |   9809876543
Jayesh Karmacharya | Module Leader        |   9856789012
Sabina Tamrakar    | Module Leader        |   9867890123
Wiz Khalif         | Module Leader        |
Anup Thakur        | Module Leader        |   9878901234
Ram Nepal          | Module Leader        |
Utsav Rajput       | Module Leader        |   9872345456
Yubraj Khadka      | Module Leader        |   9876654567
Hari Ghimire       | Module Leader        |
Bom Bahadur        | Module Leader        |   9856798765
Govind Chaudhary   | Module Leader        |

14 rows selected.
```

Figure 58 Information query 5

In this query Teacher Name, Teacher Type and Phone Number attributes where selected from Teacher and tResidence tables using JOIN and WHERE clause to find out the module leaders with all their phone number.

vi.) List all Students who are enrolled in Networking Specification?

➔SELECT Course.Specification_Name, Student.Name FROM Course

JOIN Student_Info ON Course.Course_ID = Student_Info.Course_ID

JOIN Student ON Student_Info.Student_ID = Student.Student_ID

WHERE Specification_Name=’Networking’;

```
SQL> SELECT Course.Specification_Name, Student.Name FROM Course
  2  JOIN Student_Info ON Course.Course_ID = Student_Info.Course_ID
  3  JOIN Student ON Student_Info.Student_ID = Student.Student_ID
  4  WHERE Specification_Name='Networking';

SPECIFICATION_NAME   | NAME
-------------------- | --------------------
Networking           | Piyush Sharma
Networking           | Rihan Ojha
Networking           | Ashish Shrestha
Networking           | Bikram Rathor
Networking           | Prajeet Thakur
Networking           | Azan Ahmad
Networking           | Smriti Thapa

7 rows selected.
```

Figure 59 Information query 6

In this query Specification Name, Student Name are selected from Course and Student tables using JOIN and WHERE clause to find out the total students who are enrolled in Networking Specification.

vii.) List the Fax Number of Instructor who teaches the Database Module?

➔SELECT Teacher.Teacher_ID,Module_Details.Module_Name,tResidence.Fax_Number FROM Course

```
JOIN Module_Details ON Course.Course_ID=Module_Details.Course_ID

JOIN Teacher_Info ON Course.Course_ID=Teacher_Info.Course_ID

JOIN Teacher on Teacher_Info.Teacher_ID=Teacher.Teacher_ID

JOIN tAddress_Info ON Teacher.Teacher_ID = tAddress_Info.Teacher_ID

JOIN tAddress ON tAddress_Info.AddressTec_ID = tAddress.AddressTec_ID

JOIN tCountry ON tAddress.Country_ID = tCountry.Country_ID

JOIN tResidence ON tCountry.House_Number = tResidence.House_Number

WHERE Module_Name='Database';
```

```
SQL> SELECT Teacher.Teacher_ID, Module_Details.Module_Name,tResidence.Fax_Number FROM Course
  2  JOIN Module_Details ON Course.Course_ID=Module_Details.Course_ID
  3  JOIN Teacher_Info ON Course.Course_ID=Teacher_Info.Course_ID
  4  JOIN Teacher on Teacher_Info.Teacher_ID=Teacher.Teacher_ID
  5  JOIN tAddress_Info ON Teacher.Teacher_ID = tAddress_Info.Teacher_ID
  6  JOIN tAddress ON tAddress_Info.AddressTec_ID = tAddress.AddressTec_ID
  7  JOIN tCountry ON tAddress.Country_ID = tCountry.Country_ID
  8  JOIN tResidence ON tCountry.House_Number = tResidence.House_Number
  9  WHERE Module_Name='Database';

TEACHER_ID | MODULE_NAME     | FAX_NUMBER
---------- | --------------- | ----------
       201 | Database        |
       203 | Database        |       6131
       205 | Database        |

3 rows selected.
```

Figure 60 Information query 7

In this query Module Name and Fax Number attributes are selected from Module Details and tResidence tables using JOIN and WHERE clause to find out the instructor's Fax Number who teaches database module.

viii.) List the specification falls under the BIT Course?

➔SELECT Course_Name, Specification_Name FROM Course WHERE Course_Name = 'BIT';

```
Run SQL Command Line

SQL> SELECT Course_Name, Specification_Name FROM Course
  2  WHERE Course_Name ='BIT';

COURSE_NAM | SPECIFICATION_NAME
---------- | --------------------
BIT        | Computing
BIT        | Networking
BIT        | Multimedia

3 rows selected.
```

Figure 61 Information query 8

In this query Course Name and Specification Name attributes were selected from the Course table using SELECT statement and WHERE clause to find the Specification that falls under BIT.

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

➔SELECT Module_Details.Module_Name, Module.Class FROM Module_Details

JOIN Module ON Module_Details.Module_ID = Module.Module_ID

WHERE Class = ‘Kanchanjanga’;

```
SQL> SELECT Module_Details.Module_Name, Module.Class FROM Module_Details
  2  JOIN Module ON Module_Details.Module_ID = Module.Module_ID
  3  WHERE Class='Kanchanjanga';

MODULE_NAME                       | CLASS
--------------------------------- | ---------------------------------
Database                          | Kanchanjanga
Communications Engineering        | Kanchanjanga
Digital Design                    | Kanchanjanga
Cyber Security                    | Kanchanjanga
Economics and Society             | Kanchanjanga
Economics and Society             | Kanchanjanga
Leadership and Strategic Management | Kanchanjanga

7 rows selected.
```

Figure 62 Information query 9

In this query Module Name and Class attributes were selected from Module Details and Module tables using JOIN and WHERE clause to find all the modules which are taught in Kanchanjanga class.

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

➔SELECT Teacher.Name,tCountry.Country, tCountry.Province, tCountry.City, tCountry.Street, tCountry.House_Number, tAddress.Email

FROM Teacher

JOIN tAddress_Info ON Teacher.Teacher_ID=tAddress_Info.Teacher_ID

JOIN tAddress ON tAddress_Info.AddressTec_ID=tAddress.AddressTec_ID

JOIN tCountry ON tAddress.Country_ID=tCountry.Country_ID

JOIN tResidence ON tCountry.House_Number=tResidence.House_Number

WHERE Name LIKE ‘%a’;

```
SQL> SELECT Teacher.Name,tCountry.Country,
  2   tCountry.Province, tCountry.City, tCountry.Street, tCountry.House_Number, tAddress.Email
  3  FROM Teacher
  4  JOIN tAddress_Info ON Teacher.Teacher_ID=tAddress_Info.Teacher_ID
  5  JOIN tAddress ON tAddress_Info.AddressTec_ID=tAddress.AddressTec_ID
  6  JOIN tCountry ON tAddress.Country_ID=tCountry.Country_ID
  7  JOIN tResidence ON tCountry.House_Number=tResidence.House_Number
  8  WHERE Name LIKE '%a';

NAME               | COUNTRY        |   PROVINCE | CITY           | STREET          | HOUSE_NUMBER | EMAIL
-------------------- | -------------- | ---------- | -------------- | --------------- | ------------ | ------------------
Sameer Khadka      | Nepal          |          3 | Lalitpur       | Balkumari-6     |           85 | sameer@gmail.com
Piyush Khadka      | Nepal          |          4 | Pokhara        | Lakeside-3      |           86 | piyush@gmail.com
Ramesh Karmacharya | Nepal          |          6 | Jhapa          | Birtamod-8      |           87 | ramesh@gmail.com
Neha Khadka        | Nepal          |          5 | Chitwan        | Tadi-16         |           88 | neha@gmail.com
Rabin Bajracharya  | Nepal          |          3 | Lalitpur       | Sanepa-2        |           89 | rabin@gmail.com
Simran Sha         | Nepal          |          3 | Lalitpur       | Jwalakhel-15    |           90 | simran@gmail.com
Ramesh Thapa       | Nepal          |          5 | Heatauda       | Cement-6        |           74 | ramesh@gmail.com
Jayesh Karmacharya | India          |          5 | Mumbai         | Delhi-5         |           75 | jayesh@gmail.com
Yubraj Khadka      | Nepal          |          6 | Pokhara        | Sarangkot-10    |           81 | yubraj@gmail.com

9 rows selected.
```

Figure 63 Information query 10

In this query Teacher Name, Country, Province, City, Street, House Number and Email attributes were selected from Teacher, tCountry and tAddress tables using JOIN and

WHERE clause to find all the teachers with their addresses who have ‘a’ at the end of their name.

4.2. Transaction Queries

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

➔ SELECT Course.Course_Name, Course.Specification_Name, Student.Name, (Course.Specification_Fees)AS "Original Fees", CASE Course.Specification_Name

WHEN 'Computing' THEN Specification_Fees*0.9

ELSE Specification_Fees

END AS Reduced_Fee FROM Course

JOIN Student_Info ON Course.Course_ID=Student_Info.Course_ID

JOIN Student ON Student_Info.Student_ID=Student.Student_ID

ORDER BY Reduced_Fee;

```
SQL> SELECT Course.Course_Name,Course.Specification_Name,Student.Name,
  2  (Course.Specification_Fees)AS "Original Fees", CASE Course.Specification_Name
  3  WHEN 'Computing' THEN Specification_Fees*0.9
  4  ELSE Specification_Fees
  5  END AS Reduced_Fee FROM Course
  6  JOIN Student_Info ON Course.Course_ID=Student_Info.Course_ID
  7  JOIN Student ON Student_Info.Student_ID=Student.Student_ID
  8  ORDER BY Reduced_Fee;

COURSE_NAME     | SPECIFICATION_NAME   | NAME            | Original Fees | REDUCED_FEE
--------------- | -------------------- | --------------- | ------------- | -----------
BIT             | Computing            | Siddhartha Ghim |        115000 |      103500
BIT             | Computing            | Rohan Gurung    |        115000 |      103500
BBA             | Finance              | Shrijana Rokaya |        110000 |      110000
BBA             | Marketing            | Aayusha Shresth |        110000 |      110000
BBA             | Finance              | Dwayne Johnson  |        110000 |      110000
BBA             | Marketing            | Katrina Kaif    |        110000 |      110000
BIT             | Networking           | Ashish Shrestha |        115000 |      115000
BIT             | Networking           | Smriti Thapa    |        115000 |      115000
BIT             | Multimedia           | Chirag Timalsin |        115000 |      115000
BIT             | Networking           | Rihan Ojha      |        115000 |      115000
BIT             | Networking           | Piyush Sharma   |        115000 |      115000
BIT             | Multimedia           | Bishnu Thapa    |        115000 |      115000
BIT             | Networking           | Azan Ahmad      |        115000 |      115000
BIT             | Networking           | Prajeet Thakur  |        115000 |      115000
BIT             | Networking           | Bikram Rathor   |        115000 |      115000
MBA             | Accounting           | Sahayog G.C.    |        210000 |      210000
MBA             | Accounting           | Simon Wilson    |        210000 |      210000
MSc IT          | Security             | Priyanka Chopra |        215000 |      215000
MSc IT          | Security             | Shiv Thakur     |        215000 |      215000

19 rows selected.
```

Figure 64 Transaction query 1

In this query Course Name, Specification Name, Student Name, Specification Fees attributes were selected from Course, Student tables using JOIN, CASE, ORDER BY function to show the students, course they enroll in and their fees also reducing 10% of the fees if they are enrolled in a computing course.

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?

➔SELECT COALESCE(Phone_Number,1234567890)AS Contact_Details FROM tResidence;

```
SQL> SELECT COALESCE(Phone_Number,1234567890)AS Contact_Details FROM tResidence;

CONTACT_DETAILS
---------------
     1234567890
     9823456890
     1234567890
     9809876543
     9856789012
     9867890123
     1234567890
     9878901234
     1234567890
     9872345456
     9876654567
     1234567890
     9856798765
     1234567890
     9823456290
     9823426890
     9833456890
     9823556890
     9823455890
     9823486890

20 rows selected.
```

Figure 65 Transaction query 2

In this query Phone Number attribute is selected from tResidence tables using SELECT and COALESCE statement to place the default Number 1234567890 if the list of phone numbers to the location of the teacher residence is empty and the column name as 'Contact_Details'.

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

➔SELECT Name, ROUND((SYSDATE-Joined_Date)/7,0) AS WEEKS FROM Student;

```
SQL> SELECT Name, ROUND((SYSDATE-Joined_Date)/7,0) AS WEEKS FROM Student;

NAME                 |      WEEKS
-------------------- | ----------
Piyush Sharma        |         102
Siddhartha Ghimire   |         102
Rihan Ojha           |         154
Chirag Timalsina     |         154
Priyanka Chopra      |          50
Katrina Kaif         |         102
Dwayne Johnson       |          50
Simon Wilson         |         154
Rohan Gurung         |         102
Bishnu Thapa         |          50
Ashish Shrestha      |         154
Aayusha Shrestha     |          50
Shiv Thakur          |         102
Sahayog G.C.         |         154
Shrijana Rokaya      |         102
Bikram Rathor        |         154
Prajeet Thakur       |         154
Azan Ahmad           |          46
Smriti Thapa         |         153

19 rows selected.
```

Figure 66 Transaction query 3

In this query Student Name and Joined Date Attributes were used from Student tables using SELECT and ROUND Statement to find the student's total number of weeks since they have enrolled in the course by subtracting Joined Date from System Date(Present Date) and dividing it by total number of days in a week i.e. 7 and round of by 0.

iv.) Show the name of the instructors who got equal salary and work in the same Specification?

➔SELECT Course.Specification_Name, Teacher.Name, Teacher.Salary FROM Course

JOIN Teacher_Info ON Course.Course_ID=Teacher_Info.Course_ID

JOIN Teacher ON Teacher_Info.Teacher_ID=Teacher.Teacher_ID

WHERE Salary=50000 AND Specification_Name= 'Finance';

```
SQL> SELECT Course.Specification_Name, Teacher.Name, Teacher.Salary FROM Course
  2  JOIN Teacher_Info ON Course.Course_ID=Teacher_Info.Course_ID
  3  JOIN Teacher ON Teacher_Info.Teacher_ID=Teacher.Teacher_ID
  4  WHERE Salary=50000 AND Specification_Name='Finance';

SPECIFICATION_NAME   | NAME                 |     SALARY
-------------------- | -------------------- | ----------
Finance              | Ramesh Thapa         |      50000
Finance              | Hari Ghimire         |      50000
Finance              | Bom Bahadur          |      50000
Finance              | Govind Chaudhary     |      50000

4 rows selected.
```

Figure 67 Transaction query 4

In this query Specification Name, Teacher Name and Salary where selected from Course and Teacher Tables using JOIN and Where clause to find the name of the teacher who got equal salary and work in the same specification.

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

➔ SELECT Course.Course_Name, Course.Specification_Name, Count (Student.Student_ID) AS “Total Number Of Students”, Max (Module.Marks) AS “Highest Marks” From Course

JOIN Module_Details ON Course.Course_ID=Module_Details.Course_ID

Join Module ON Module_Details.Module_ID=Module.Module_ID

Join Student_Info ON Course.Course_ID=Student_Info.Course_ID

Join Student ON Student_Info.Student_ID=Student.Student_ID

GROUP BY Course.Course_Name, Course.Specification_Name

ORDER BY Course.Course_Name;

```
Select Run SQL Command Line

SQL> SELECT Course.Course_Name,Course.Specification_Name,
  2  Count(Student.Student_ID)AS "Total Number Of Students",
  3  Max(Module.Marks)AS "Highest Marks" FROM Course
  4  JOIN Module_Details On Course.Course_ID=Module_Details.Course_ID
  5  JOIN Module ON Module_Details.Module_ID=Module.Module_ID
  6  JOIN Student_Info ON Course.Course_ID=Student_Info.Course_ID
  7  JOIN Student ON Student_Info.Student_ID=Student.Student_ID
  8  GROUP BY Course.Course_Name,Course.Specification_Name
  9  ORDER BY Course.Course_Name;

COURSE_NAME     | SPECIFICATION_NAME   | Total Number Of Students | Highest Marks
--------------- | -------------------- | ------------------------ | -------------
BBA             | Finance              |                        4 |            82
BBA             | Marketing            |                        4 |            70
BIT             | Computing            |                        6 |            82
BIT             | Multimedia           |                        6 |            70
BIT             | Networking           |                       21 |            82
MBA             | Accounting           |                        4 |            82
MSc IT          | Security             |                        4 |            82

7 rows selected.
```

Figure 68 Transaction query 5

In this query Course Name, Specification Name, Student ID and Marks attributes were selected from Course, Module, Student tables using JOIN, Count, Max function to find all the course and specification with the total number of students enrolled course name and the highest marks obtained.

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

➔SELECT * FROM Teacher WHERE Teacher_Type=’Module Leader’;

```
SQL> SELECT * FROM Teacher WHERE Teacher_Type='Module Leader';

TEACHER_ID | NAME           | SEX        | JOINED_DA | TEACHER_TYPE       | TEACHER_ST |     SALARY
---------- | -------------- | ---------- | --------- | ------------------ | ---------- | ----------
       201 | Steve Jobs     | Male       | 01-FEB-00 | Module Leader      | Active     |      60000
       202 | Sabin Bharati  | Male       | 01-JAN-05 | Module Leader      | Active     |      50000
       203 | Mark Zuckerburg| Male       | 01-DEC-99 | Module Leader      | Active     |      75000
       204 | Ramesh Thapa   | Male       | 01-FEB-13 | Module Leader      | Active     |      48000
       205 | Jayesh Karmacha| Male       | 01-MAY-02 | Module Leader      | Active     |      40000
       206 | Sabina Tamrakar| Female     | 01-FEB-10 | Module Leader      | Active     |      52000
       207 | Wiz Khalif     | Male       | 01-JUN-18 | Module Leader      | Active     |      55000
       208 | Anup Thakur    | Male       | 01-JUN-18 | Module Leader      | Active     |      54000
       209 | Ram Nepal      | Male       | 01-SEP-18 | Module Leader      | Active     |      20000
       210 | Utsav Rajput   | Male       | 01-JUN-13 | Module Leader      | Active     |      15000
       211 | Yubraj Khadka  | Male       | 01-SEP-05 | Module Leader      | Active     |      59000
       212 | Hari Ghimire   | Male       | 01-JAN-20 | Module Leader      | Active     |      50000
       213 | Bom Bahadur    | Male       | 01-DEC-19 | Module Leader      | Active     |      54000
       214 | Govind Chaudhar| Male       | 08-JUN-19 | Module Leader      | Active     |      50000

14 rows selected.
```

Figure 69 Transaction query 6

In this query all the attributes inside the teacher table were selected using SELECT and WHERE clause to list all the teacher who are also a module leader.

4.3. Creation of Dump File

```
F:\Course_Work>exp coursework/coursework file=coursework.dmp

Export: Release 11.2.0.2.0 - Production on Fri Dec 18 21:13:42 2020

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


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)

About to export specified users ...
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user COURSEWORK
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user COURSEWORK
About to export COURSEWORK's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export COURSEWORK's tables via Conventional Path ...
. . exporting table                         COURSE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         MODULE          5 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 MODULE_DETAILS         17 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                       SADDRESS         19 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                  SADDRESS_INFO         19 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                       SCOUNTRY         19 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                           SDOB         19 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
```

```
. . exporting table                      SRESIDENCE          20 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         STUDENT          19 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                    STUDENT_INFO          19 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                       TADDRESS          20 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                  TADDRESS_INFO          20 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                       TCOUNTRY          20 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                           TDOB          20 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        TEACHER          20 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                   TEACHER_INFO          23 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                     TRESIDENCE          20 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
```

```
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.
```

Figure 70 Creation of Dump file

4.4. Drop tables

```
DROP TABLE Course;

DROP TABLE Module;

DROP TABLE Module_Details;

DROP TABLE sAddress;

DROP TABLE sAddress_Info;

DROP TABLE sCountry;

DROP TABLE sDOB;

DROP TABLE sResidence;

DROP TABLE Student;

DROP TABLE Student_Info;

DROP TABLE tAddress;

DROP TABLE tAddress_Info;

DROP TABLE tCountry;

DROP TABLE tDOB;

DROP TABLE Teacher;

DROP TABLE Teacher_Info;

DROP TABLE tResidence;
```

Chapter 5- Conclusion

The Coursework which is been implemented in this report is all about the Islington College that can now able to handles every record of student and teacher from Database using Oracle SQL Plus. This report is been finalized after the great research, consulting with the instructor (Mrs. Yunisha Bajracharya) time to time, patience and hard work. Learning the Concepts of Business Rules, Relationship in ERD, Normalization, Data Implementation, Populating Database, Information and Transaction query was quite fun and thrilling. This report is done after getting the concepts of Scenario provided from the Coursework. From this report I have learnt to distinguish DDL and DML, Types Of JOIN, Use of Normalization (UNF, 1NF, 2NF, 3NF), Familiar with SQL Plus, Implementation of Business Rules and Assumption, Identification of Entities, ERD before and after Normalization, Relationship in ERD and many others. This Coursework can be used in all college to handle every data of student's and teacher's in a systematic way including their course, modules, address, specification, salary, course fee and so on. The normalization was done correcting five times using different methods and finally it is been implemented. I am mostly familiar with the Normalization from UNF to 3NF. Now I can able to Separate the Repeating Groups, Super Type and Subtype, Partial dependency, Transitive dependency and so on.

Last but not the least I want to thank my every instructor who provided the time in this Pandemic Situation to guide me in every difficulty and solve the problems.

Chapter 6- References

References

DATAVERSITY Education, L., 2020. *Normalizing With Entity Relation.* [Online]
Available at: https://tdan.com/normalizing-with-entity-relationship-diagramming/4583#:~:text=Since%20an%20ERD%20also%20utilizes,model%2C%20and%20speed%20its%20implementation.
[Accessed 4 12 2020].

IBM Knowledge, C., 2020. *Key concepts.* [Online]
Available at: https://www.ibm.com/support/knowledgecenter/de/SSWSR9_11.6.0/com.ibm.mdmhs.overview.doc/entityconcepts.html
[Accessed 26 11 2020].

Islington, C., 2018. *Programmes.* [Online]
Available at: https://islington.edu.np/programmes/
[Accessed 25 11 2020].

Islington, C., 2018. *Vision, Mission, Values.* [Online]
Available at: https://islington.edu.np/about/vision-mission-values/
[Accessed 25 11 2020].

Techopedia, 2019. *What Is Primary Key?.* [Online]
Available at: https://www.techopedia.com/definition/5547/primary-key#:~:text=A%20primary%20key%20is%20a,parse%20data%20within%20the%20table.&text=It%20must%20contain%20a%20unique,It%20cannot%20contain%20null%20values.
[Accessed 9 12 2020].

W3School, 2020. *SQL Foreign Key Constraints.* [Online]
Available at: https://www.w3schools.com/sql/sql_foreignkey.asp
[Accessed 9 12 2020].

W3Schools, 2020. *SQL ALTER TABLE STATEMENT.* [Online]
Available at:

https://www.w3schools.com/sql/sql_alter.asp#:~:text=The%20ALTER%20TABLE%20statement%20is,constraints%20on%20an%20existing%20table.
[Accessed 9 12 2020].