CC5051NA Databases Systems

50% Individual Coursework

2019-20 Autumn

Student Name: Summit Shrestha

London Met ID: 19032520

College ID: NP01CP4A190113

Assignment Due Date: 20th December, 2020

Assignment Submission Date: 20th December 2020

Word Count (Where Required):

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

1.2. Current Business Activities and Operations 2

1.3. Business Rules 3

1.4. Creation of entities and attributes 4

1.5. Initial ER Diagram 5

2. Part-1 Database Design 6

2.1. Assumptions 6

2.2. Normalization 6

2.2.1. UNF(Un-Normalized Form) 6

2.2.2 1NF (First Normal Form) 7

2.2.2. 2NF(Second Normal Form) 8

2.2.3. 3NF (Third Normal Form) 12

2.3. ER diagram of normalized database 15

3. Part-2 Database Implementation 17

3.1. Table Generation 17

3.2. Populating Database 34

3.3. Final Tables 60

4. Database Querying 69

4.1. Information Queries 69

4.1.1. List all the students with all their addresses with their phone numbers 69

4.1.2. List all the modules which are taught by more than one instructor 70

4.1.3. List the name of all the instructors whose name contains ‘s’ and salary is above 50,000 71

4.1.4. List the modules comes under the ‘Multimedia’ specification. 72

4.1.5. List the name of the head of modules with the list of his phone number 73

4.1.6. List all Students who have enrolled in ‘networking’ specifications 74

4.1.7. List the fax number of the instructor who teaches the ‘database’ module 74

4.1.8. List the specification falls under the BIT course. 75

4.1.9. List all the modules taught in any one particular class 76

4.1.10. List all the teachers with all their addresses who have ‘a’ at the end of their first names. 77

4.2. Creating Dump file 78

4.3. Drop Table 80

5. CONCLUSION 82

References 83

Table of Figures

Figure 1: Initial ER diagram 5
Figure 2: Normalized ER Diagram... 15
Figure 3: Create Table Person_DOB 18
Figure 4: Create table Person_House.. 18
Figure 5: Create Table Address 19
Figure 6: Create Table Person 20
Figure 7: Create table Student 21
Figure 8: Create table Student_Details 22
Figure 9: Create table Instructor_Salary 22

Figure 10: Create table Instructor 23
Figure 11: Create table Instructor_Details 24
Figure 12: Create Table Class 25
Figure 13:Create Table Session_Class 25
Figure 14: Create Table Module 26
Figure 15: Create Table Session_Module 27
Figure 16: Create table Module_Detais 28
Figure 17: Create Table Specification 28
Figure 18: Create table Specification_Details 29
Figure 19: Create table Course 30
Figure 20: Create table Course_Details 31
Figure 21: Create table Spec_Enrollment 32
Figure 22:Create Table Admission 32
Figure 23: Create Table Admission_Details 33
Figure 24: Inserting to Person_DOB 35
Figure 25: Inserting into Person_House 37
Figure 26: Inserting to Address 39
Figure 27: Inserting into Person 41
Figure 28: Inserting into Student 42
Figure 29: Inserting into Student_Details 43
Figure 30: Inserting into Instructor_Salary 44
Figure 31: Inserting into Instructor 45
Figure 32: Insertig into Instructor_Details 46
Figure 33: Inserting into Class 47
Figure 34: Inserting into Session_Class 48
Figure 35: Inserting into Module 49
Figure 36: Inserting into Session_Module 50
Figure 37: Inserting into Moduel_Details 51
Figure 38: Inserting into Specification 52
Figure 39: Inserting to Specification_Details 53
Figure 40: Inserting into Course 55
Figure 41:Inserting into Course_Details 56
Figure 42: Inserting into Spec_Enrollment 57
Figure 43: Inserting into Admission 58
Figure 44: Inserting into Admission_Details 59
Figure 45: Person_DOB 60
Figure 46: Person_House 61
Figure 47: Address 62
Figure 48:Person 62
Figure 49: Student_Details 63
Figure 50: Student 63
Figure 51: Instructor Salarya 63
Figure 52: Instructor 64
Figure 53: Instructor_Details 64
Figure 54: Class 64
Figure 55:: Session_Class 65
Figure 56: Module 65

Figure 57: Session_Module 65
Figure 58: Module_Details 66
Figure 59:Specification 66
Figure 60: Specification Details 66
Figure 61: Course 67
Figure 62: Course_Details 67
Figure 63: Spec_Enrollment 67
Figure 64: Admission 68
Figure 65: Admission_Details 68
Figure 66: Information Query No 1 69
Figure 67: Information Query No 2 70
Figure 68: Information Query No 3 71
Figure 69: Information Query No 4 72
Figure 70: Information Query No 5 73
Figure 71: Information Query No 6 74
Figure 72: Information Query No.7 75
Figure 73: Information Query No. 8 76
Figure 74: Infromation Query No. 9 76
Figure 75: Information Query No. 10 77
Figure 81: Creating dump file Softwarica.dmp 79
Figure 82: Droping all tables of the database 81

Table of Tables

Table 1: Entity and Attributes 4

1. College Introduction

1.1. Introduction to college

Softwarica College, established in 2010, is a pioneer in introducing British Education in Nepal. The college is working in collaboration with Coventry University which is one of the UK's leading Universities, ranked No.15 in the UK in the Guardian University Guide 2019 to offer a range of undergraduate programmes. It is located in Dillibazar, 15-20 minutes walking distance from the Putalisadak bus stop.

The main goal of Softwarica College is to provide education based on practical approach as combination of both academic and real life skills would help students towards their holistic development. The college recognizes the fact that in order to be successful in finding a rewarding career, real life skills are also essential. There are many programmes or courses offered by the college including BSc (Hons) Computing and BSc (Hons) Ethical Hacking & Cybersecurity. The college has crossed significant milestones in this short span of time. This is evident by the fact that the student number has already crossed 400 and is becoming very popular among students who aspire to gain a British Qualification. The current objective of the college is to graduate the students with more than one year of real time work experience facilitating them to sought positions in the middle level in companies both at home and abroad.

Softwarica College is one of the Best IT College in Nepal and has been satisfying students in terms of course delivery and other academic resources for over a decade.

1.2. Current Business Activities and Operations

Softwarica college has been satisfying students in terms of course delivery and other academic resources by using these current Business Activities:

i. The college provides computer labs, well equipped technical and networking labs, library with exhaustive collection of books on IT and Business, local and international experts on teaching faculties.

ii. It also provides real time work experience right after the completion of the first year, students are provided with internship opportunities with renowned Companies in Nepal. Similarly, after the completion of second year students are placed in various companies in Nepal and abroad in positions according to their skill level.

iii. The colleges lets the students enroll in many courses including BSc (Hons) IT, BBA, MBA, etc .The courses are divided into specification and the specifications are further divided into Modules.

iv. The college admission of student every year at the month of September-October. Students who have completed their +2 or A level course will be eligible for the admission.

v. After admission of a student is done then transfer of course cannot be done. However transfer from one specification of a course to another specification of the same course is only available until the 2nd teaching week of semester two.

vi. The instructor are given salary on the basis of their type.

vii. The modules are taught in a class which is spacious enough to fit 30 - 35 students comfortably.

viii. Finance Department is responsible for collecting the fees of the students and giving out salary to the instructors.

1.3. Business Rules

There are various rules that a college must follow. Some of them are listed below:

i. The college database should be able to keep track of address of all people.

ii. Out of all address details, one mailing address must be recorded.

iii. Each address consists of country, province, city, street, house number and a list of phone numbers to the location of the address and a list of numbers to the location of the address.

iv. The college contains many course each of which may offer any number of specifications.

v. Each specification contains several modules.

vi. An instructor can be associated to only one course but a course can have many instructors.

vii. Each course must have only one course leader.

viii. Each instructor can teach one module at a time but a module can be taught by many instructors.

ix. A student can enroll in only one course and each course can have many number of students.

x. During a session only one module can be taught by an instructor to a number a student in a class.

xi. A module can also be taught in multiple classes during multiple sessions by different instructor to different groups of students.

1.4. Creation of entities and attributes

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity. (Tutorialspoint, 2020)

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes. (Tutorialspoint, 2020)

Entity	Attributes
Person	Person_ID(PK), Name, Address, DOB(Date of Birth),
Student	Student_ID(PK), Person(ID), Course_ID(Fk) , Admission(Date), Fee
Course	Course_IT(PK), Instructor_ID(FK), Name, Specification, Module
Instructor	Instructor_ID(PK), Instructor_Type, Salary
Class	Class_ID(PK), Room_No
Session	Session_ID(PK), Course_ID(FK), Class_ID(FK)

Table 1: Entity and Attributes

1.5. Initial ER Diagram

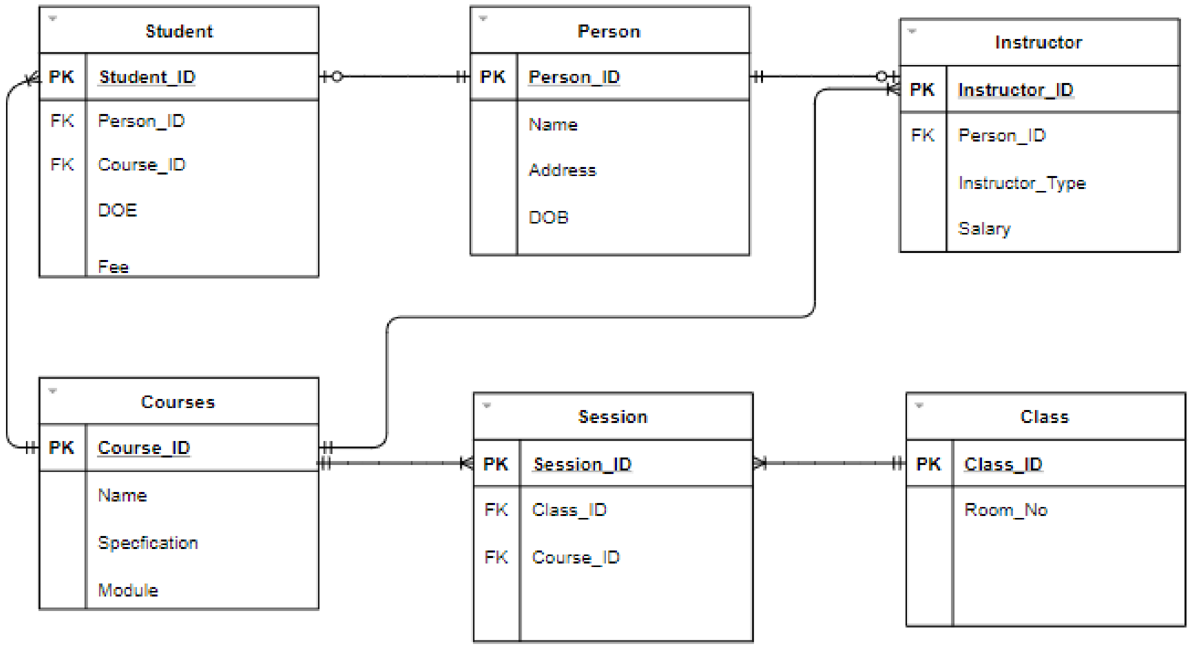


Figure 1: Initial ER diagram

2. Part-1 Database Design

2.1. Assumptions

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

- Student_ID has been assigned to Students and Instructor_ID has been assigned to Instructors.
- Every admission has Admit_ID, Student_ID, Date of Enrollment(DOE)
- Every course has Course_ID, Spec_ID, Course_Name
- Every specification has Spec_ID, Module_Code, Spec_Name
- Every session has Session_ID, Class_ID, Module_Code and Instructor_ID.
- Every Module has a module head who is also an instructor but manages other instructor of the module on the topic of how the knowledge would be given to students about the module.
- A person can either be a student or a instructor at a time but cannot be both.

2.2. Normalization

Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). (Chapple, 2020)

2.2.1. UNF(Un-Normalized Form)

Scenario for UNF

- Each person should register his/her detailed address along with a mailing address
- Each person should provide his/her Name, Age, Sex and DOB
- Each DOB of a person will have a DOB_ID
- Each Country of a person will have a Country_ID
- Each person should also provide their phone numbers and fax.
- The person can be a student or an instructor at a time.
- A student can enroll in only one course at a time.

- The data of Fee that a student must pay and the marks obtained by the student is must be recorded.
- An instructor can only teach one module.
- Every instructor is provided a salary in accordance to the type of instructor.
- During a session only one module can be taught by an instructor to a number a student in a class.

Showing Repeating Groups

People (Person_ID, Name, Age, Sex, DOB_ID, DOB, Country_ID, Country, Province, City, Street, House_number, Mailing_address, Fax, E-mail, Mobile_Number , { Student_ID, Fee, Mark}, {Admit_ID, DOE}, {Course_ID, Course_Name,Highest_Mark},{Spec_ID, Spec_name}, {Module_code, Module_Name}, {Instructor_ID, Instructor_Type, Salary}, {Class_ID, Room_no}, {Session_ID,})

2.2.2 1NF (First Normal Form)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values (Singh, n.d.). The 1NF is used to eliminate repeating groups in individual tables, create a separate table for each set of related data and identify each set of related data with a primary key.

Scenario for 1NF:

We can determine either a person is a student or an instructor with the Person_ID. Similarly, Admit_ID is related to a student that has been admitted to the college, Student_ID is related to the module a student gets enrolled in, Module_ID is related to the specificaiion that it belongs to, the instructor that it is taught by and class it is taught in. Spec_ID is also related to the course that it belongs to.

Entities:

People1 (Person_ID, Name, Age, Sex, DOB_ID, DOB, Country, Province, City, Street, House_No, Mailing_Address, Email, Fax)

Class1 (Class_ID, Room_No)

Instructor_details1 (Instructor_ID, Person_ID*, Instructor_Type, Salary)

Module_Details1 (Module_ID, Instructor_ID*, Module_Name)

Specification_Details1 (Spec_ID, Module_ID*, Spec_Name)

Course_details1 (Course_ID, Spec_ID*, Course_Name, Highest_Mark)

Student_details1 (Student_ID, Person_ID*, Spec_ID*, Fee, Mark)

Admission1 (Admit_ID, Student_ID*, DOE)

Session1 (Session_ID, Class_ID*, Module_ID*)

2.2.2. 2NF(Second Normal Form)

A table is said to be in Second Normal Form if it firstly meets all the requirements of the first normal form, then removes subsets of data that apply to multiple rows of a table and place them in separate tables and finally creates relationships between these new tables and their predecessors through the use of foreign keys. In 2NF all the non-key attributes are Fully Functionally Dependent on Primary Key and not on only a portion of Primary key. Partial Functional Dependencies are avoided because they result in data redundancy.

Scenario for 2NF:

The partial dependencies are firstly identified in 2NF. Thus identified Partial functional dependencies (PFDs) were eliminated by creating new entities and placing the attributes inside those new entities depending on the partial or full functional dependency.

Showing Partial Dependency:

For Student_details:

- Composite primary key Person_ID, Student_ID does not determine any attributes.
- Composite primary key Student_ID, Spec_ID determines Fee
- Student_ID determines the Mark

Person_ID, Student_ID →

Student_ID, Spec_ID → Fee

Student_ID → Mark

For Instructor_Details:

- Instructor_ID determines the Instructor_Type and Salary
- Composite primary key Instructor_ID, Person_ID does not determine any attributes

Instructor_ID → Instructor_Type, Salary

Instructor_ID, Person_ID →

For Course_Details:

- Course_ID determines the Course_Name, Highest Mark
- Composite primary key Course_ID, Spec_ID does not determine any attributes

Course_ID → Course_Name, Highest Mark

Course_ID, Spec_ID →

For Specification_Details:

- Spec_ID determines the Spec_Name
- Composite primary key Spec_ID, Module_ID does not determine any attributes

Spec_ID → Spec_Name

Spec_ID, Module_ID →

For Module_Details:

- Module_ID determines Module_Name
- Composite primary key Module_ID, Instructor_ID does not determine any attributes

Module_ID → Module_Name

Module_ID, Instructor_ID →

For Admission:

- Admit_ID determines Date of Enrollment(DOE)
- Composite primary key Admit_ID, Student_ID does not determine any attributes

Admit_ID, Student_ID →

Admit_ID → DOE

For Session:

- Composite primary key Session_ID, Class_ID does not determine any attributes
- Composite primary key Session_ID, Module_ID does not determine any attributes

Session_ID, Class_ID →

Session_ID, Module_ID →

Entities:

People2 (Person_ID, Name, Age, Sex, DOB_ID, DOB, Country, Province, City, Street, House_No, Mailing_Address, Email, Fax)

Student_Details2 (Student_ID* , Person_ID*)

Spec_Enrollment2 (Student_ID*, Spec_ID*, Fee)

Student2 (Student_ID, Mark)

Instructor_Details2 (Instructor_ID*, Person_ID*)

Instructor2 (Instructor_ID, Instructor_Type, Salary)

Course_Details2 (Course_ID*, Spec_ID*)

Course2 (Course_ID, Course_Name, Highest_Mark)

Specifcation_Details2 (Spec_ID*, Instructor_ID*)

Specification2 (Spec_ID, Spec_Name)

Module_Details2 (Module_ID*, Instructor_ID*)

Module2 (Module_ID, Module_Name)

Admission_Details2 (Admit_ID*, Student_ID*)

Admission2 (Admit_ID, DOE)

Session_Module2 (Session_ID*, Module_ID*)

Session_Class2 (Session_ID*, Class_ID*)

Class2 (Class_ID, Room_No)

2.2.3. 3NF (Third Normal Form)

A table design is said to be in 3NF if the table is firstly in 2NF and the transitive functional dependency does not exist. A transitive dependency in a database is an indirect relationship between values in the same table that causes a functional dependency (Chapple, 2020). To achieve the normalization standard of 3NF one must eliminate any transitive dependency.

Scenario for 3NF:

After removing partial dependencies in 2NF, the transitional functional dependencies were identified. Thus identified Transitional functional dependencies (TFDs) were eliminated by creating new entities and placing the attributes causing TFDs inside those new entities.

Showing Transitive Dependencies:

For People:

- Person_ID determines the Country_ID of a person and with the Country_ID we can easily determine Country, Province, City, Street, House_No, Mailing_Address

Person_ID → Country_ID, Country_ID → Country, Province, City, Street, House_No, Mailing_Address

- Country determines the House_No of a person and with the House_No we can determine the Phone_No and Fax

Country → House_No, House_No → Phone_No, Fax

- Person_ID determines the DOB_ID of a person and with DOB_ID we can determine the DOB and Age

Person_ID → DOB_ID, DOB_ID → DOB, Age

For Instructor:

- Instructor_ID determines Instructor_Type and with Instructor_Type we can determine the Salary.

Instructor_ID → Instructor_Type, Instructor_Type → Salary

The remaining entities do not have transitive dependency

Entities:

Person3 (Person_ID, Name, DOB_ID*, Country_ID*, Sex, Mobile_No, Email)

Address3 (Country_ID, Country, Province, City, Street, Mailing_Address)

Person_House3 (House_No, Phone_No, Fax)

Person_DOB3 (DOB_ID, DOB, Age)

Student_Details3 (Student_ID* , Person_ID*)

Spec_Enrollment3 (Student_ID*, Spec_ID*, Fee)

Student3 (Student_ID, Mark)

Instructor_Details3 (Instructor_ID*, Person_Id*)

Instructor3 (Instructor_ID, Instructor_Type*)

Instructor_Salary3 (Instructor_Type, Salary)

Course_Details3 (Course_ID*, Spec_ID*)

Course3 (Course_ID, Course_Name, Highest_Mark)

Specifcation_Details3 (Spec_ID*, Instructor_ID*)

Specification3 (Spec_ID, Spec_Name)

Module_Details3 (Module_ID*, Instructor_ID*)

Module3 (Module_ID, Module_Name)

Admission_Details3 (Admit_ID*, Student_ID*)

Admission3 (Admit_ID, DOE)

Session_Module3 (Session_ID*, Module_ID*)

Session_Class3 (Session_ID*, Class_ID*)

Class3 (Class_ID, Room_No)

2.3. ER diagram of normalized database

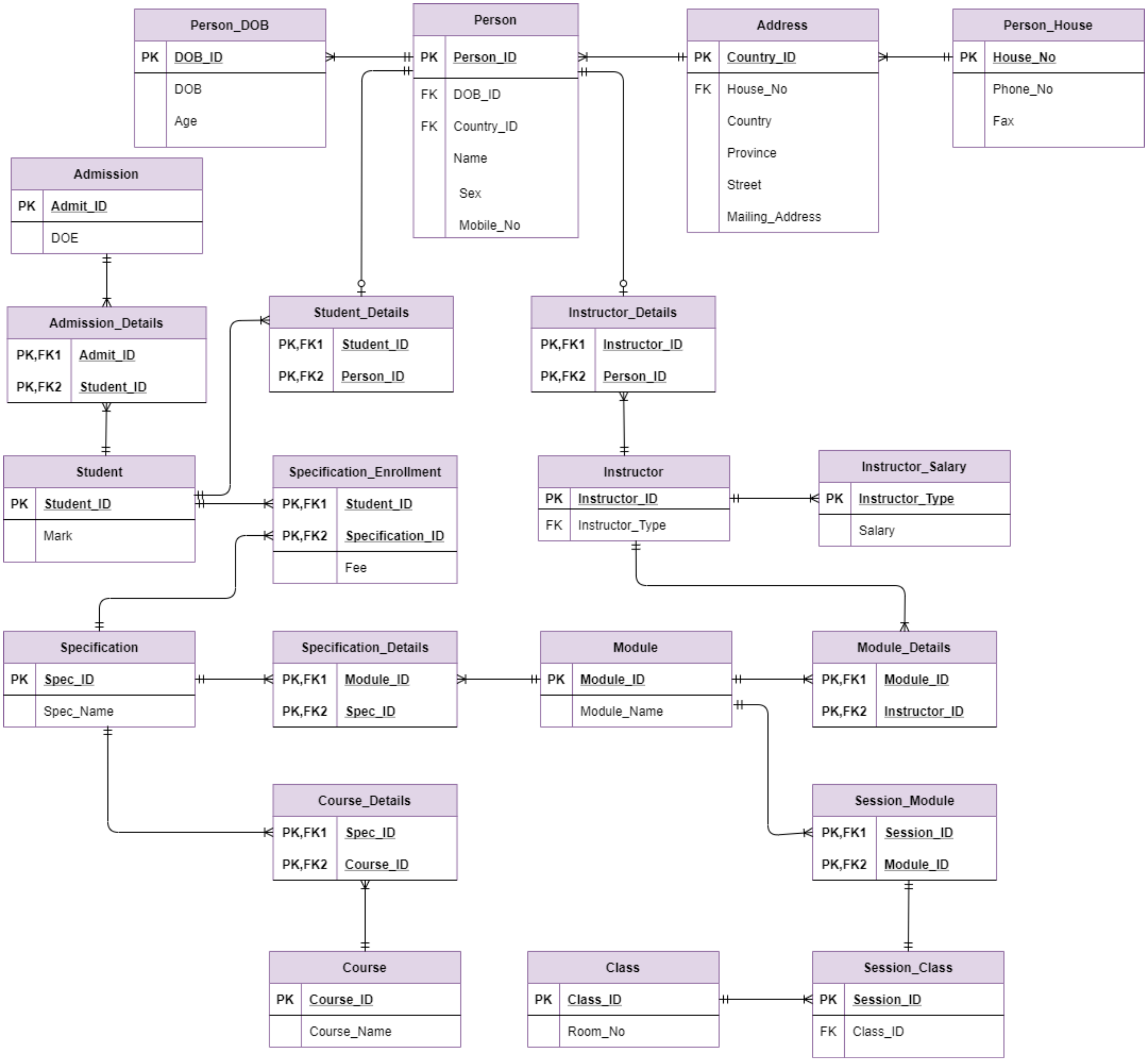


Figure 2: Normalized ER Diagram

3. Part-2 Database Implementation

3.1. Table Generation

Creating a basic table involves naming the table and defining its columns and each column's data type. CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement. (Tutorialspoint, n.d.).

ALTER TABLE modifies the design of a table after it has been created with the CREATE TABLE statement.

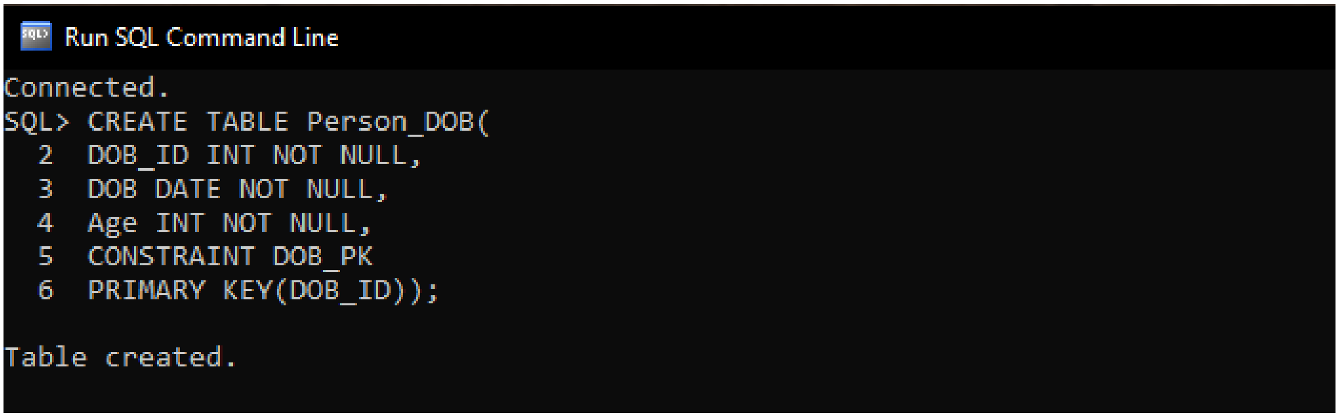
Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table. (tutorialspoint, 2019).

Primary Key is the constraint which uniquely defines the row of the table. Primary key value cannot be null and must be unique.

Foreign Key is the primary key in that exists in another table. Foreign Key cannot be null but its value can repeat.

Creating table for Person_DOB

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



```
Connected.
SQL> CREATE TABLE Person_DOB(
  2  DOB_ID INT NOT NULL,
  3  DOB DATE NOT NULL,
  4  Age INT NOT NULL,
  5  CONSTRAINT DOB_PK
  6  PRIMARY KEY(DOB_ID));

Table created.
```

Figure 3: Create Table Person_DOB

Creating table for Person_House

CREATE TABLE Person_House(

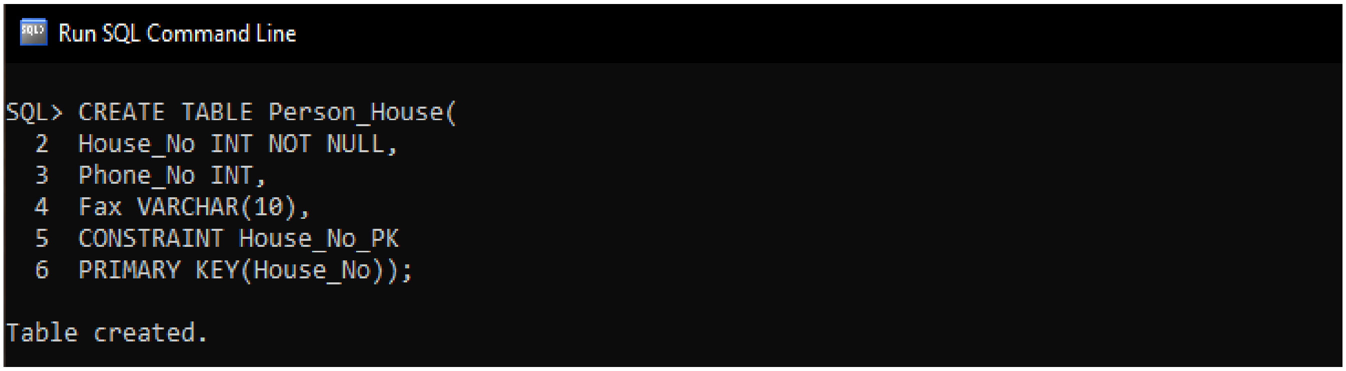
House_No INT NOT NULL,

Phone_No INT,

Fax VARCHAR(10),

CONSTRAINT House_No_PK

PRIMARY KEY(House_No));



```
SQL> CREATE TABLE Person_House(
  2  House_No INT NOT NULL,
  3  Phone_No INT,
  4  Fax VARCHAR(10),
  5  CONSTRAINT House_No_PK
  6  PRIMARY KEY(House_No));

Table created.
```

Figure 4: Create table Person_House

Creating table for Address

CREATE TABLE Address(

Country_ID INT NOT NULL,

Country VARCHAR(30) NOT NULL,

Province VARCHAR(30) NOT NULL,

City VARCHAR(30) NOT NULL,

Street VARCHAR(30) NOT NULL,

House_No INT NOT NULL,

Mailing_Address VARCHAR(30) NOT NULL,

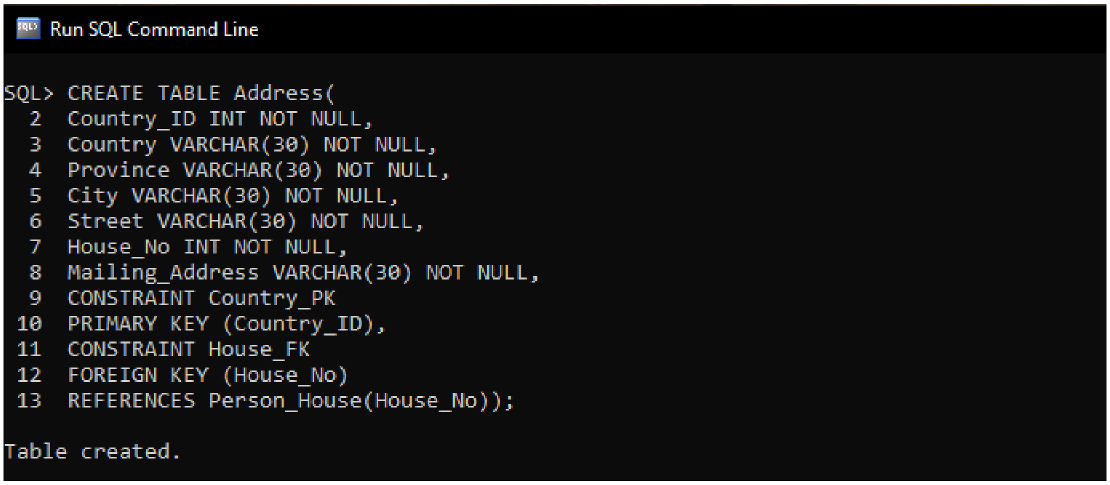
CONSTRAINT Country_PK

PRIMARY KEY (Country_ID),

CONSTRAINT House_FK

FOREIGN KEY (House_No)

REFERENCES Person_House(House_No));



```
SQL> CREATE TABLE Address(
  2  Country_ID INT NOT NULL,
  3  Country VARCHAR(30) NOT NULL,
  4  Province VARCHAR(30) NOT NULL,
  5  City VARCHAR(30) NOT NULL,
  6  Street VARCHAR(30) NOT NULL,
  7  House_No INT NOT NULL,
  8  Mailing_Address VARCHAR(30) NOT NULL,
  9  CONSTRAINT Country_PK
 10  PRIMARY KEY (Country_ID),
 11  CONSTRAINT House_FK
 12  FOREIGN KEY (House_No)
 13  REFERENCES Person_House(House_No));

Table created.
```

Figure 5: Create Table Address

Creating table for Person

CREATE TABLE Person(

Person_ID INT NOT NULL ,

Name VARCHAR(30) NOT NULL,

Sex VARCHAR(20) NOT NULL,

DOB_ID INT NOT NULL,

Country_ID INT NOT NULL,

MobileNo INT,

CONSTRAINT Person_PK

PRIMARY KEY(Person_ID),

CONSTRAINT DOB_FK

FOREIGN KEY(DOB_ID)

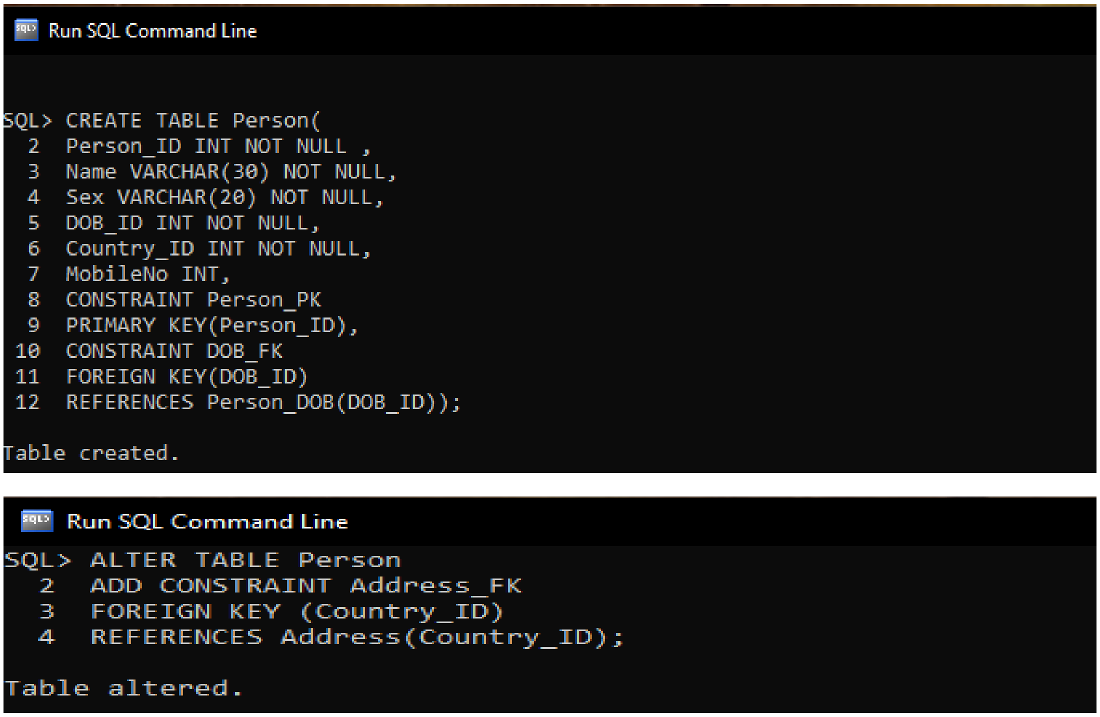
REFERENCES Person_DOB(DOB_ID));

ALTER TABLE Person

ADD CONSTRAINT Address_FK

FOREIGN KEY (Country_ID)

REFERENCES Address(Country_ID);



```
SQL> CREATE TABLE Person(
  2  Person_ID INT NOT NULL ,
  3  Name VARCHAR(30) NOT NULL,
  4  Sex VARCHAR(20) NOT NULL,
  5  DOB_ID INT NOT NULL,
  6  Country_ID INT NOT NULL,
  7  MobileNo INT,
  8  CONSTRAINT Person_PK
  9  PRIMARY KEY(Person_ID),
 10  CONSTRAINT DOB_FK
 11  FOREIGN KEY(DOB_ID)
 12  REFERENCES Person_DOB(DOB_ID));

Table created.
```

```
SQL> ALTER TABLE Person
  2  ADD CONSTRAINT Address_FK
  3  FOREIGN KEY (Country_ID)
  4  REFERENCES Address(Country_ID);

Table altered.
```

Figure 6: Create Table Person

Creating table for Student

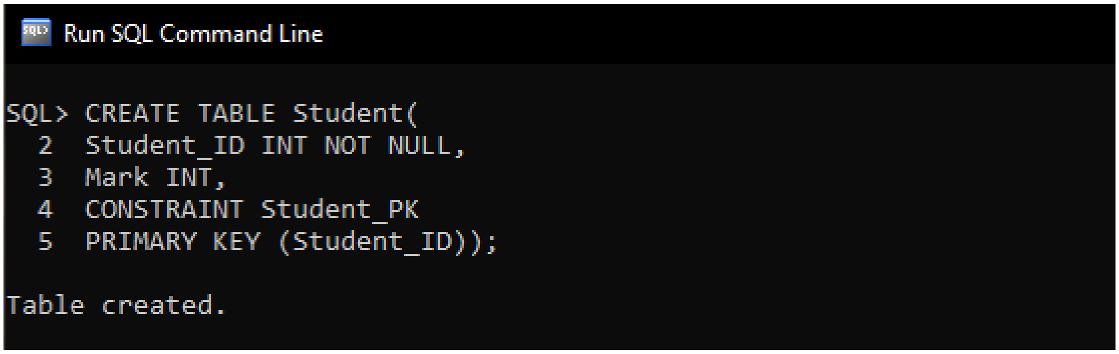
CREATE TABLE Student(

Student_ID INT NOT NULL,

Mark INT,

CONSTRAINT Student_PK

PRIMARY KEY (Student_ID));



```
SQL> CREATE TABLE Student(
     Student_ID INT NOT NULL,
     Mark INT,
     CONSTRAINT Student_PK
     PRIMARY KEY (Student_ID));

Table created.
```

Figure 7: Create table Student

Creating table for Student_Details

CREATE TABLE Student_Details(

Student_ID INT NOT NULL,

Person_ID INT NOT NULL,

CONSTRAINT Student_PK1

PRIMARY KEY (Student_ID, Person_ID),

CONSTRAINT Student_FK11

FOREIGN KEY (Student_ID)

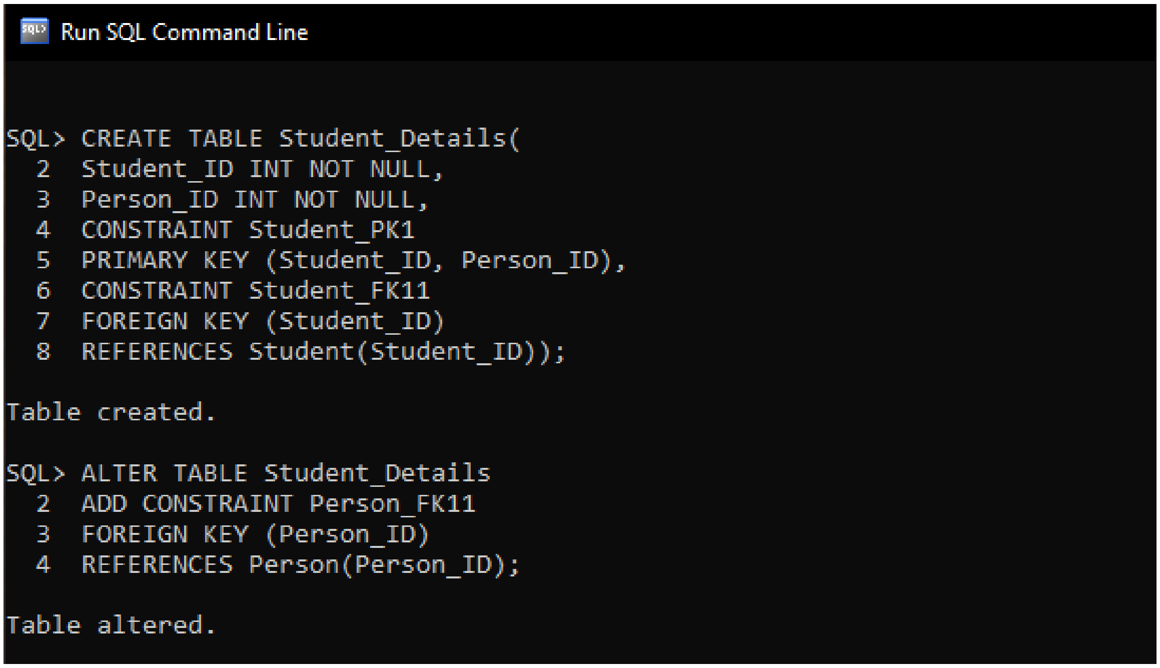
REFERENCES Student(Student_ID));

ALTER TABLE Student_Details

ADD CONSTRAINT Person_FK11

FOREIGN KEY (Person_ID)

REFERENCES Person(Person_ID);



```
SQL> CREATE TABLE Student_Details(
  2  Student_ID INT NOT NULL,
  3  Person_ID INT NOT NULL,
  4  CONSTRAINT Student_PK1
  5  PRIMARY KEY (Student_ID, Person_ID),
  6  CONSTRAINT Student_FK11
  7  FOREIGN KEY (Student_ID)
  8  REFERENCES Student(Student_ID));

Table created.

SQL> ALTER TABLE Student_Details
  2  ADD CONSTRAINT Person_FK11
  3  FOREIGN KEY (Person_ID)
  4  REFERENCES Person(Person_ID);

Table altered.
```

Figure 8: Create table Student_Details

Creating table for Instructor_Salary

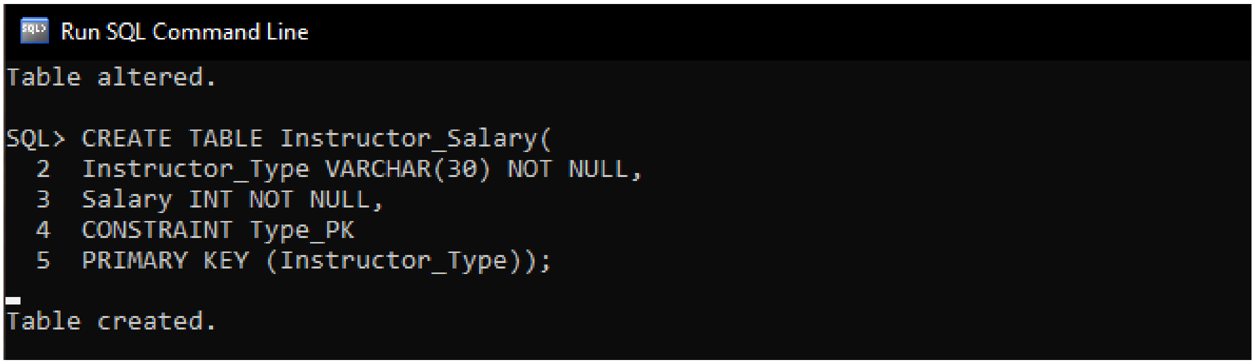
CREATE TABLE Instructor_Salary(

Instructor_Type VARCHAR(30) NOT NULL,

Salary INT NOT NULL,

CONSTRAINT Type_PK

PRIMARY KEY (Instructor_Type));



```
Table altered.

SQL> CREATE TABLE Instructor_Salary(
  2  Instructor_Type VARCHAR(30) NOT NULL,
  3  Salary INT NOT NULL,
  4  CONSTRAINT Type_PK
  5  PRIMARY KEY (Instructor_Type));

Table created.
```

Figure 9: Create table Instructor_Salary

Creating table for Instructor

CREATE TABLE Instructor(

Instructor_ID INT NOT NULL,

Instructor_Type VARCHAR(30) NOT NULL,

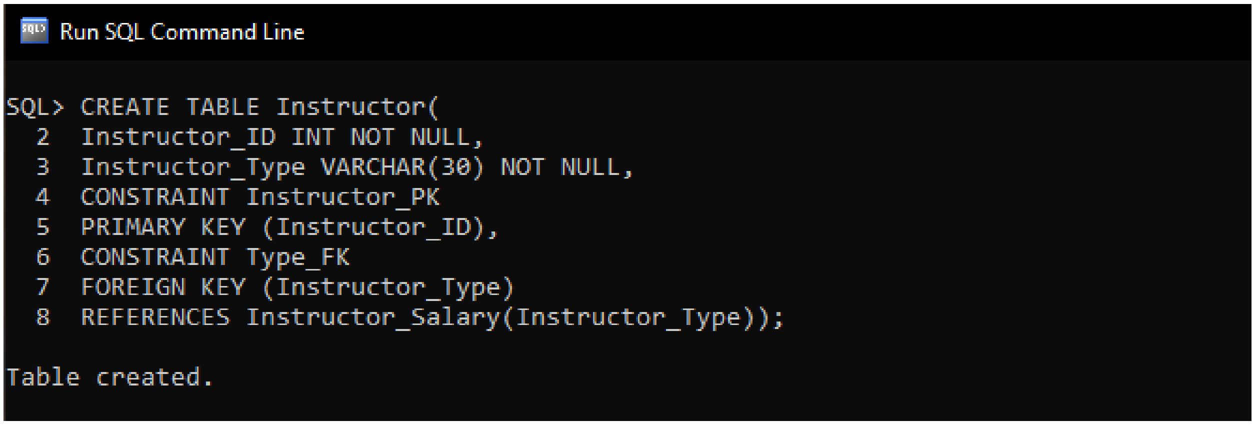
CONSTRAINT Instructor_PK

PRIMARY KEY (Instructor_ID),

CONSTRAINT Type_FK

FOREIGN KEY (Instructor_Type)

REFERENCES Instructor_Salary(Instructor_Type));



```
SQL> CREATE TABLE Instructor(
     Instructor_ID INT NOT NULL,
     Instructor_Type VARCHAR(30) NOT NULL,
     CONSTRAINT Instructor_PK
     PRIMARY KEY (Instructor_ID),
     CONSTRAINT Type_FK
     FOREIGN KEY (Instructor_Type)
     REFERENCES Instructor_Salary(Instructor_Type));

Table created.
```

Figure 10: Create table Instructor

Creating table for Instructor_Details

CREATE TABLE Instructor_Details(

Person_ID INT NOT NULL,

Instructor_ID INT NOT NULL,

CONSTRAINT Instructor_PK2

PRIMARY KEY (Person_ID, Instructor_ID),

CONSTRAINT Instructor_FK

FOREIGN KEY (Instructor_ID)

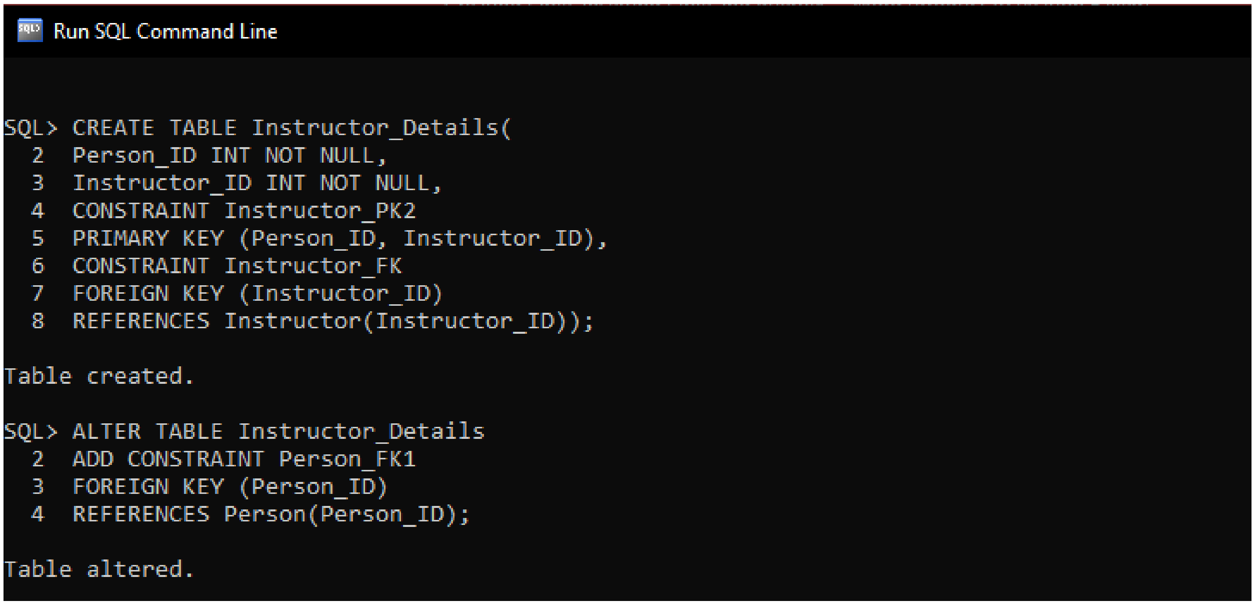
REFERENCES Instructor(Instructor_ID));

ALTER TABLE Instructor_Details

ADD CONSTRAINT Person_FK1

FOREIGN KEY (Person_ID)

REFERENCES Person(Person_ID);



```
SQL> CREATE TABLE Instructor_Details(
  2  Person_ID INT NOT NULL,
  3  Instructor_ID INT NOT NULL,
  4  CONSTRAINT Instructor_PK2
  5  PRIMARY KEY (Person_ID, Instructor_ID),
  6  CONSTRAINT Instructor_FK
  7  FOREIGN KEY (Instructor_ID)
  8  REFERENCES Instructor(Instructor_ID));

Table created.

SQL> ALTER TABLE Instructor_Details
  2  ADD CONSTRAINT Person_FK1
  3  FOREIGN KEY (Person_ID)
  4  REFERENCES Person(Person_ID);

Table altered.
```

Figure 11: Create table Instructor_Details

Creating table for Class

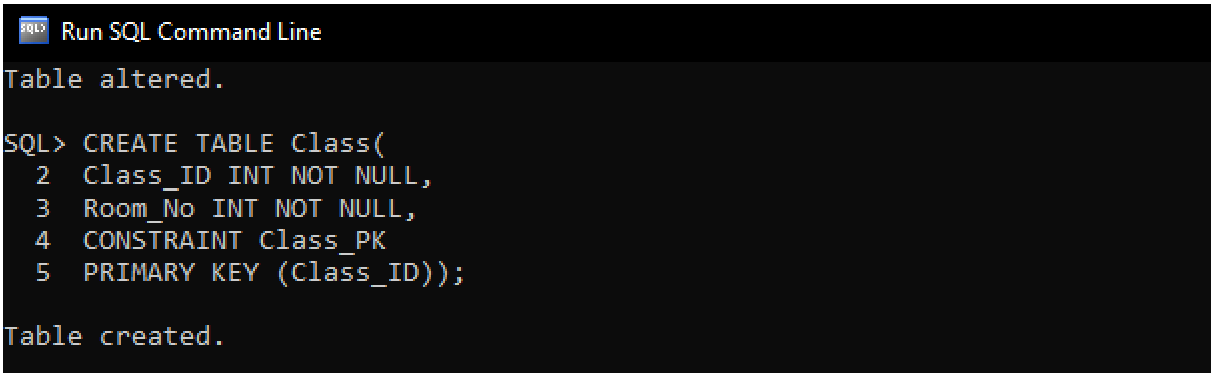
CREATE TABLE Class(

Class_ID INT NOT NULL,

Room_No INT NOT NULL,

CONSTRAINT Class_PK

PRIMARY KEY (Class_ID));



```
Table altered.

SQL> CREATE TABLE Class(
  2  Class_ID INT NOT NULL,
  3  Room_No INT NOT NULL,
  4  CONSTRAINT Class_PK
  5  PRIMARY KEY (Class_ID));

Table created.
```

Figure 12: Create Table Class

Creating table for Session_Class

CREATE TABLE Session_Class(

Session_ID INT NOT NULL,

Class_ID INT NOT NULL,

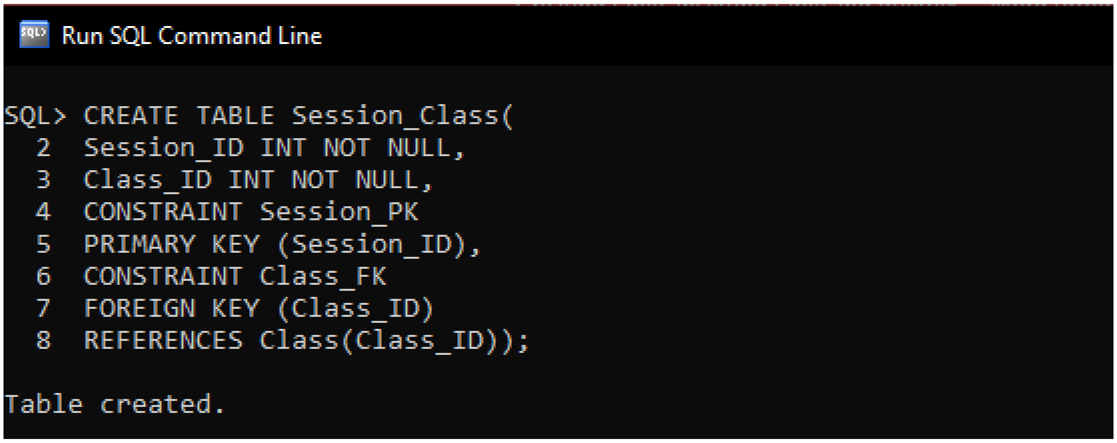
CONSTRAINT Session_PK

PRIMARY KEY (Session_ID),

CONSTRAINT Class_FK

FOREIGN KEY (Class_ID)

REFERENCES Class(Class_ID));



```
SQL> CREATE TABLE Session_Class(
  2  Session_ID INT NOT NULL,
  3  Class_ID INT NOT NULL,
  4  CONSTRAINT Session_PK
  5  PRIMARY KEY (Session_ID),
  6  CONSTRAINT Class_FK
  7  FOREIGN KEY (Class_ID)
  8  REFERENCES Class(Class_ID));

Table created.
```

Figure 13:Create Table Session_Class

Creating table for Module

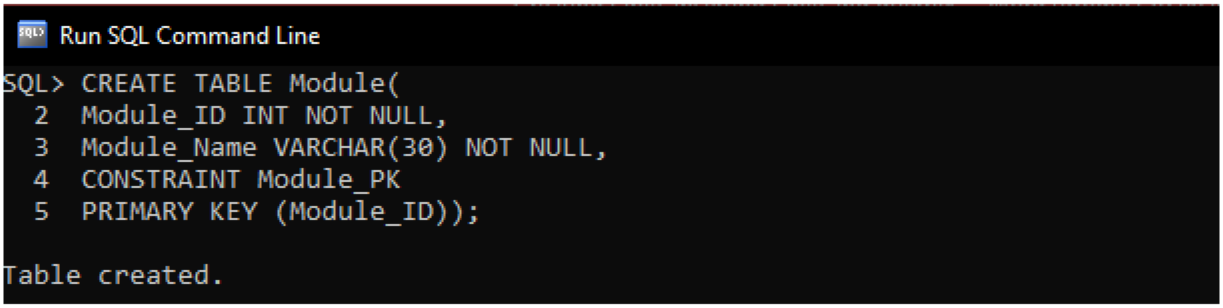
CREATE TABLE Module(

Module_ID INT NOT NULL,

Module_Name VARCHAR(30) NOT NULL,

CONSTRAINT Module_PK

PRIMARY KEY (Module_ID));

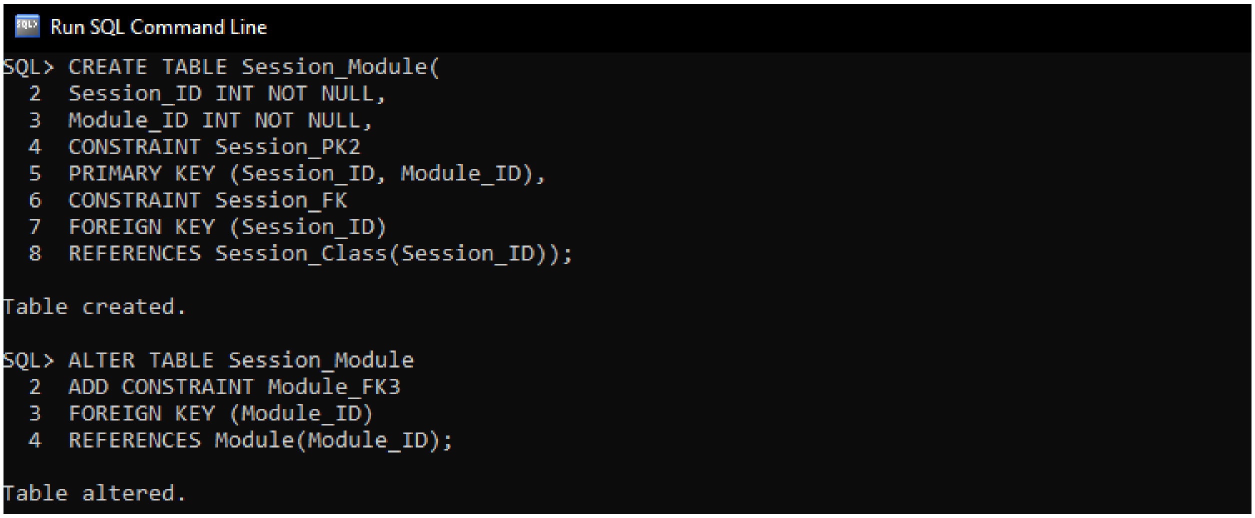


```
SQL> CREATE TABLE Module(
  2  Module_ID INT NOT NULL,
  3  Module_Name VARCHAR(30) NOT NULL,
  4  CONSTRAINT Module_PK
  5  PRIMARY KEY (Module_ID));

Table created.
```

Figure 14: Create Table Module

Creating table for Session_Module

```
CREATE TABLE Session_Module(
Session_ID INT NOT NULL,
Module_ID INT NOT NULL,
CONSTRAINT Session_PK2
PRIMARY KEY (Session_ID, Module_ID),
CONSTRAINT Session_FK
FOREIGN KEY (Session_ID)
REFERENCES Session_Class(Session_ID));


ALTER TABLE Session_Module
ADD CONSTRAINT Module_FK3
FOREIGN KEY (Module_ID)
REFERENCES Module(Module_ID);
```



```
SQL> CREATE TABLE Session_Module(
     Session_ID INT NOT NULL,
     Module_ID INT NOT NULL,
     CONSTRAINT Session_PK2
     PRIMARY KEY (Session_ID, Module_ID),
     CONSTRAINT Session_FK
     FOREIGN KEY (Session_ID)
     REFERENCES Session_Class(Session_ID));

Table created.

SQL> ALTER TABLE Session_Module
     ADD CONSTRAINT Module_FK3
     FOREIGN KEY (Module_ID)
     REFERENCES Module(Module_ID);

Table altered.
```

Figure 15: Create Table Session_Module

Creating table for Module_Details

CREATE TABLE Module_Details (

Instuctor_ID INT NOT NULL,

Module_ID INT NOT NULL,

CONSTRAINT Instuctor_PK3

PRIMARY KEY (Instuctor_ID, Module_ID),

CONSTRAINT Instuctor_FK

FOREIGN KEY (Instuctor_ID)

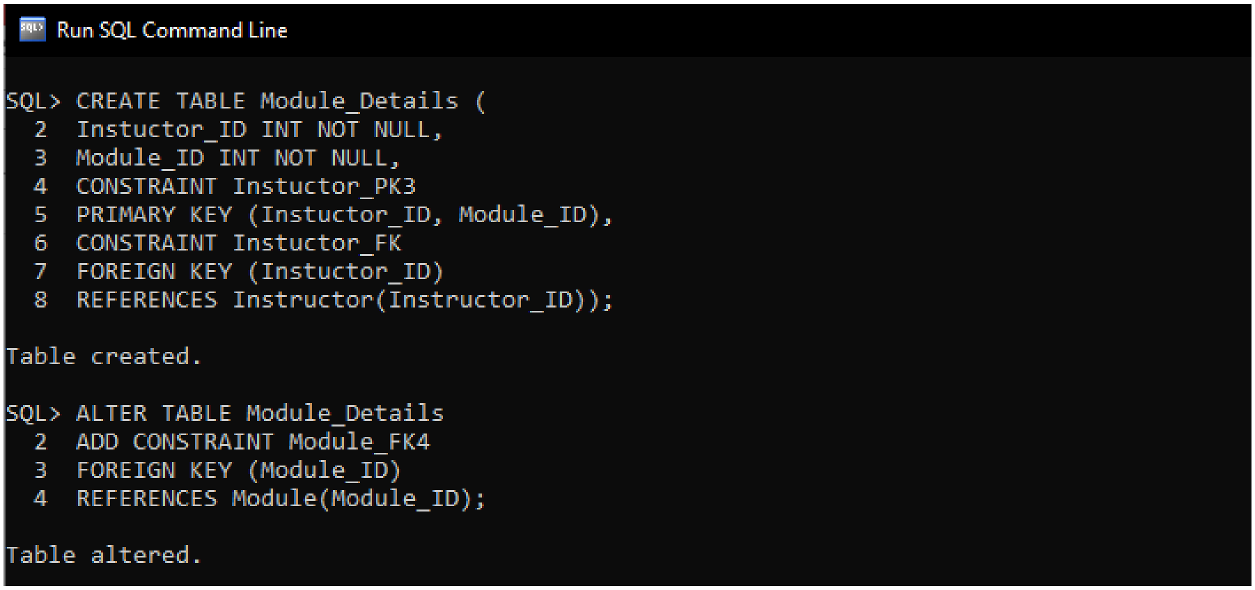
REFERENCES Instructor(Instructor_ID));

ALTER TABLE Module_Details

ADD CONSTRAINT Module_FK4

FOREIGN KEY (Module_ID)

REFERENCES Module(Module_ID);



```
SQL> CREATE TABLE Module_Details (
     Instuctor_ID INT NOT NULL,
     Module_ID INT NOT NULL,
     CONSTRAINT Instuctor_PK3
     PRIMARY KEY (Instuctor_ID, Module_ID),
     CONSTRAINT Instuctor_FK
     FOREIGN KEY (Instuctor_ID)
     REFERENCES Instructor(Instructor_ID));

Table created.

SQL> ALTER TABLE Module_Details
     ADD CONSTRAINT Module_FK4
     FOREIGN KEY (Module_ID)
     REFERENCES Module(Module_ID);

Table altered.
```

Figure 16: Create table Module_Detais

Creating table for Specification

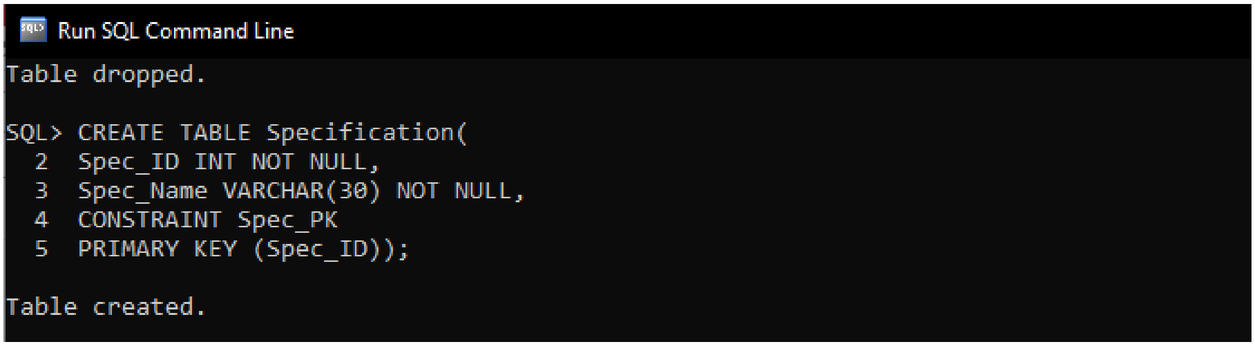
CREATE TABLE Specification(

Spec_ID INT NOT NULL,

Spec_Name VARCHAR(30) NOT NULL,

CONSTRAINT Spec_PK

PRIMARY KEY (Spec_ID));



```
Table dropped.

SQL> CREATE TABLE Specification(
     Spec_ID INT NOT NULL,
     Spec_Name VARCHAR(30) NOT NULL,
     CONSTRAINT Spec_PK
     PRIMARY KEY (Spec_ID));

Table created.
```

Figure 17: Create Table Specification

Creating table for Specification_Details

CREATE TABLE Specification_Details(

Module_ID INT NOT NULL,

Spec_ID INT NOT NULL,

CONSTRAINT Module_PK2

PRIMARY KEY (Module_ID, Spec_ID),

CONSTRAINT Module_FK

FOREIGN KEY (Module_ID)

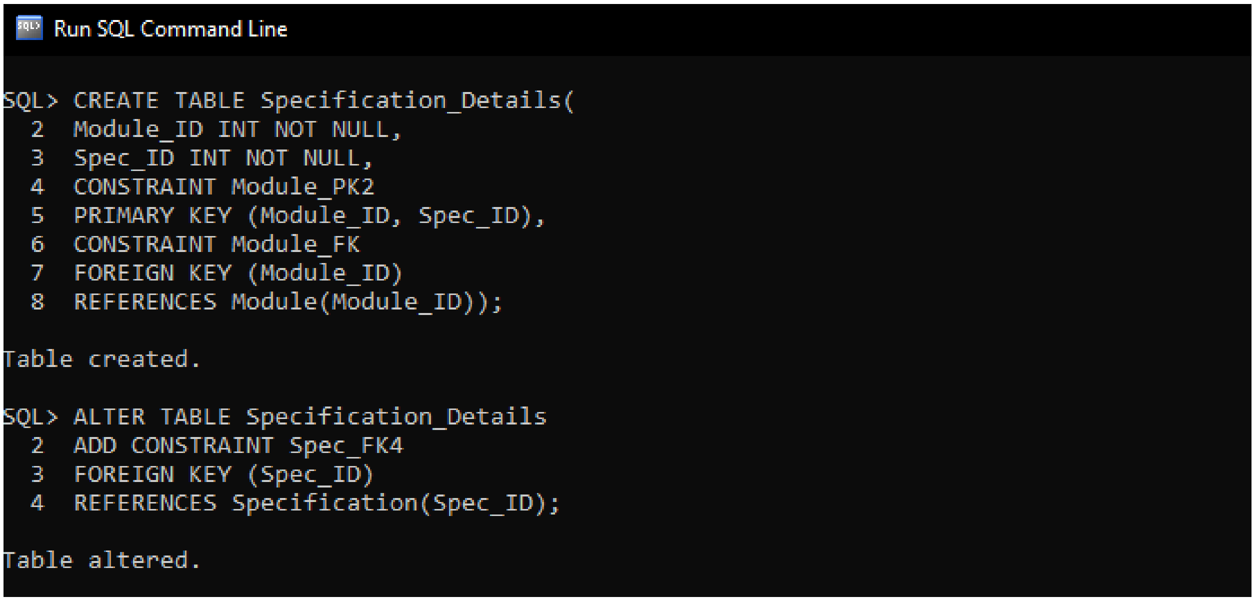
REFERENCES Module(Module_ID));

ALTER TABLE Specification_Details

ADD CONSTRAINT Spec_FK4

FOREIGN KEY (Spec_ID)

REFERENCES Specification(Spec_ID);



```
SQL> CREATE TABLE Specification_Details(
     Module_ID INT NOT NULL,
     Spec_ID INT NOT NULL,
     CONSTRAINT Module_PK2
     PRIMARY KEY (Module_ID, Spec_ID),
     CONSTRAINT Module_FK
     FOREIGN KEY (Module_ID)
     REFERENCES Module(Module_ID));

Table created.

SQL> ALTER TABLE Specification_Details
     ADD CONSTRAINT Spec_FK4
     FOREIGN KEY (Spec_ID)
     REFERENCES Specification(Spec_ID);

Table altered.
```

Figure 18: Create table Specification_Details

Creating table for Course

CREATE TABLE Course(

Course_ID INT NOT NULL,

Course_Name VARCHAR(30) NOT NULL,

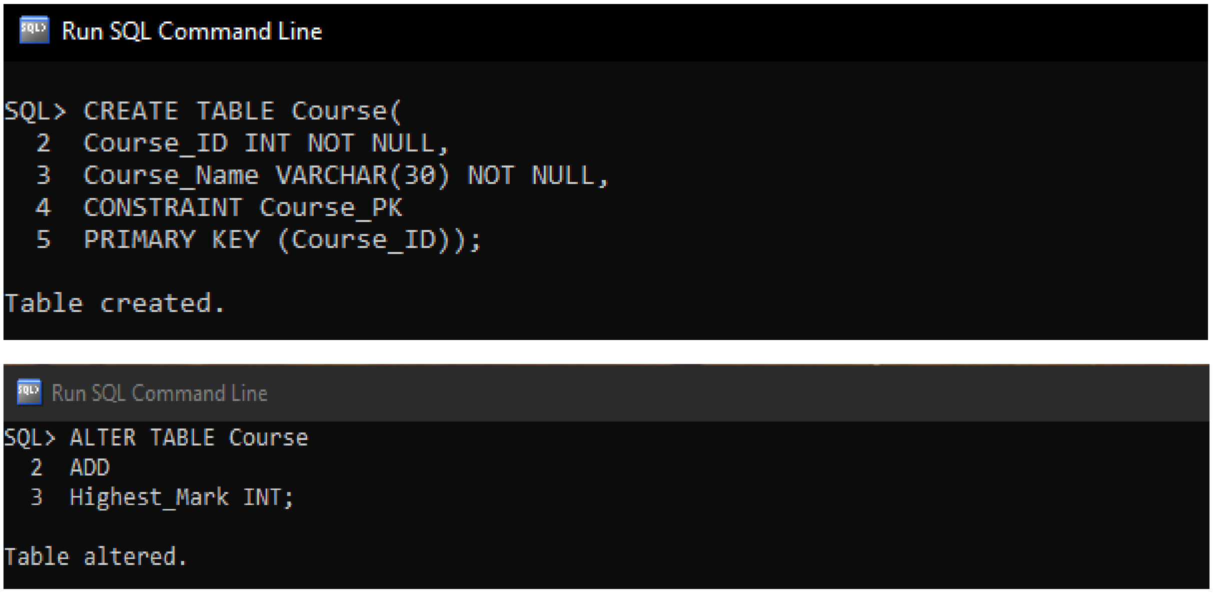
CONSTRAINT Course_PK

PRIMARY KEY (Course_ID));

ALTER TABLE Course

ADD

Highest_Mark INT;



```
SQL> CREATE TABLE Course(
  2  Course_ID INT NOT NULL,
  3  Course_Name VARCHAR(30) NOT NULL,
  4  CONSTRAINT Course_PK
  5  PRIMARY KEY (Course_ID));

Table created.
```

```
SQL> ALTER TABLE Course
  2  ADD
  3  Highest_Mark INT;

Table altered.
```

Figure 19: Create table Course

Creating table for Course_Details

CREATE TABLE Course_Details(

Course_ID INT NOT NULL,

Spec_ID INT NOT NULL,

CONSTRAINT Spec_PK2

PRIMARY KEY (Course_ID, Spec_ID),

CONSTRAINT Course_FK

FOREIGN KEY (Course_ID)

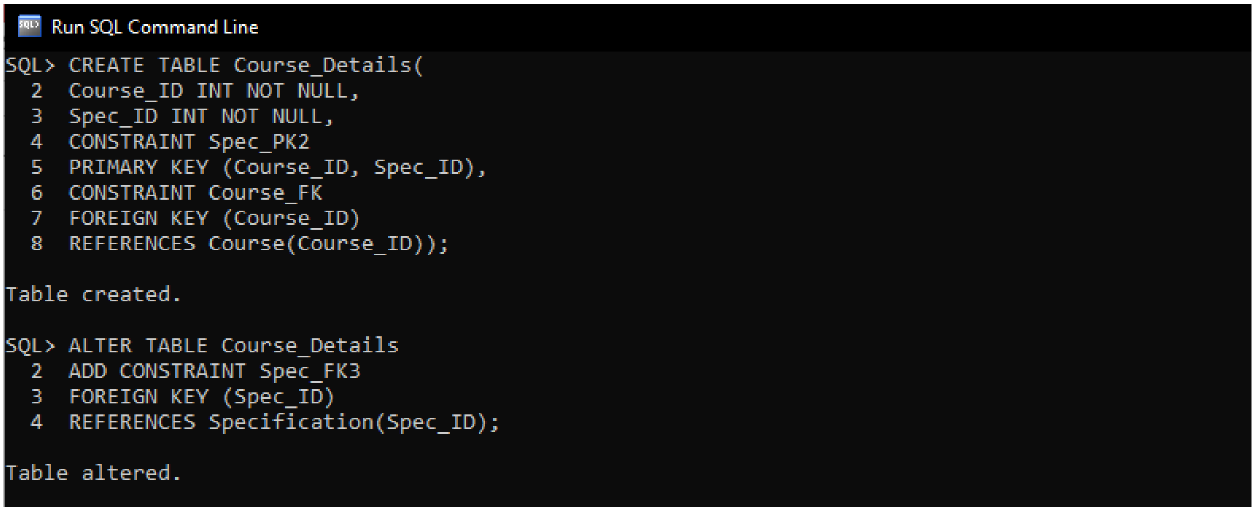
REFERENCES Course(Course_ID));

ALTER TABLE Course_Details

ADD CONSTRAINT Spec_FK3

FOREIGN KEY (Spec_ID)

REFERENCES Specification(Spec_ID);



Run SQL Command Line

```
SQL> CREATE TABLE Course_Details(
  2  Course_ID INT NOT NULL,
  3  Spec_ID INT NOT NULL,
  4  CONSTRAINT Spec_PK2
  5  PRIMARY KEY (Course_ID, Spec_ID),
  6  CONSTRAINT Course_FK
  7  FOREIGN KEY (Course_ID)
  8  REFERENCES Course(Course_ID));

Table created.

SQL> ALTER TABLE Course_Details
  2  ADD CONSTRAINT Spec_FK3
  3  FOREIGN KEY (Spec_ID)
  4  REFERENCES Specification(Spec_ID);

Table altered.
```

Figure 20: Create table Course_Details

Creating table for Spec_Enrollment

CREATE TABLE Spec_Enrollment (

Spec_ID INT NOT NULL,

Student_ID INT NOT NULL,

Fee INT NOT NULL,

CONSTRAINT Spec_PK22

PRIMARY KEY (Spec_ID, Student_ID),

CONSTRAINT Spec_FK15

FOREIGN KEY (Spec_ID)

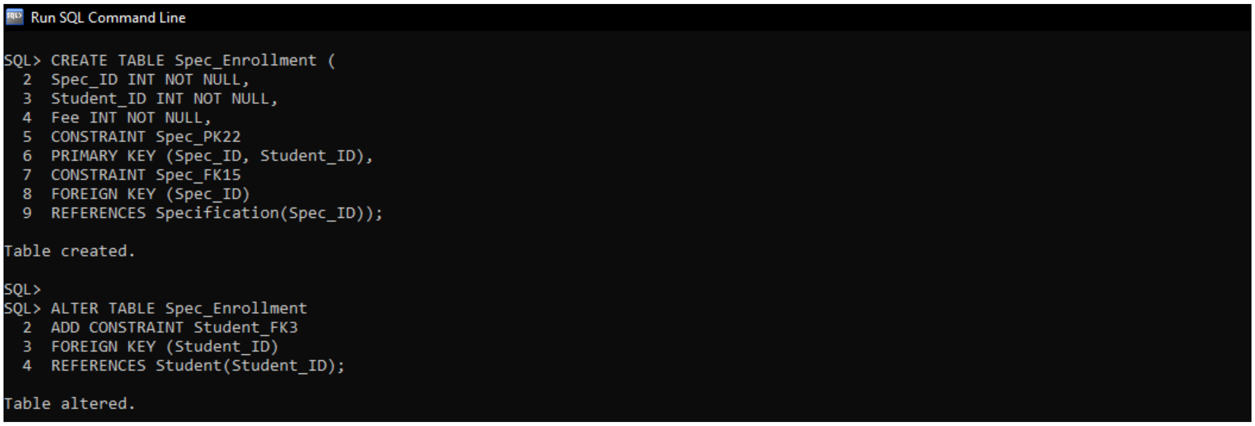
REFERENCES Specification(Spec_ID));

ALTER TABLE Spec_Enrollment

ADD CONSTRAINT Student_FK3

FOREIGN KEY (Student_ID)

REFERENCES Student(Student_ID);



```
SQL> CREATE TABLE Spec_Enrollment (
  2  Spec_ID INT NOT NULL,
  3  Student_ID INT NOT NULL,
  4  Fee INT NOT NULL,
  5  CONSTRAINT Spec_PK22
  6  PRIMARY KEY (Spec_ID, Student_ID),
  7  CONSTRAINT Spec_FK15
  8  FOREIGN KEY (Spec_ID)
  9  REFERENCES Specification(Spec_ID));

Table created.

SQL>
SQL> ALTER TABLE Spec_Enrollment
  2  ADD CONSTRAINT Student_FK3
  3  FOREIGN KEY (Student_ID)
  4  REFERENCES Student(Student_ID);

Table altered.
```

Figure 21: Create table Spec_Enrollment

Creating table for Admission

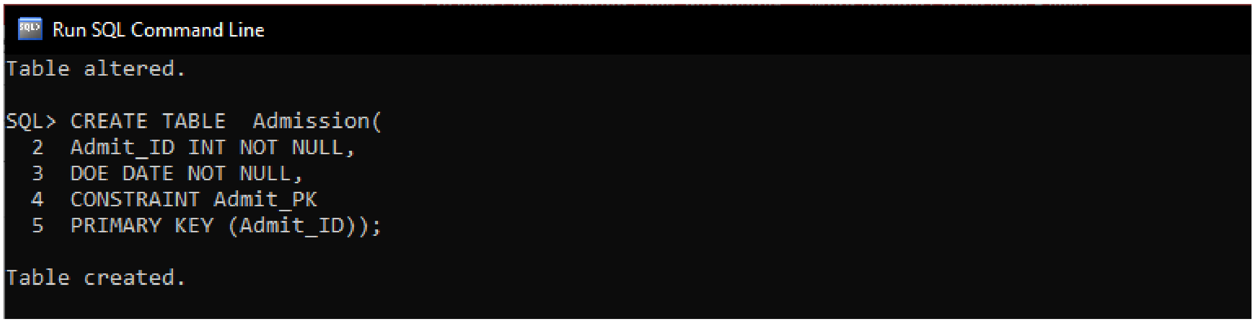
CREATE TABLE Admission(

Admit_ID INT NOT NULL,

DOE DATE NOT NULL,

CONSTRAINT Admit_PK

PRIMARY KEY (Admit_ID));



```
Table altered.

SQL> CREATE TABLE  Admission(
  2  Admit_ID INT NOT NULL,
  3  DOE DATE NOT NULL,
  4  CONSTRAINT Admit_PK
  5  PRIMARY KEY (Admit_ID));

Table created.
```

Figure 22:Create Table Admission

Creating table for Admission_Details

CREATE TABLE Admission_Details(

Admit_ID INT NOT NULL,

Student_ID INT NOT NULL,

CONSTRAINT Admit_PK2

PRIMARY KEY (Admit_ID, Student_ID),

CONSTRAINT Admit_FK

FOREIGN KEY (Admit_ID)

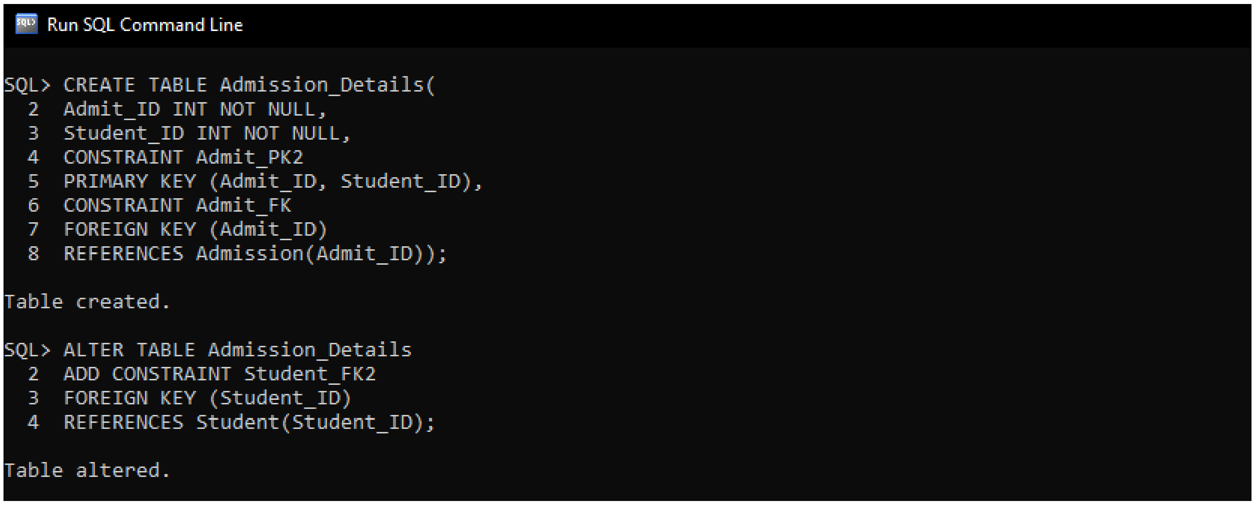
REFERENCES Admission(Admit_ID));

ALTER TABLE Admission_Details

ADD CONSTRAINT Student_FK2

FOREIGN KEY (Student_ID)

REFERENCES Student(Student_ID);



```
SQL> CREATE TABLE Admission_Details(
     Admit_ID INT NOT NULL,
     Student_ID INT NOT NULL,
     CONSTRAINT Admit_PK2
     PRIMARY KEY (Admit_ID, Student_ID),
     CONSTRAINT Admit_FK
     FOREIGN KEY (Admit_ID)
     REFERENCES Admission(Admit_ID));

Table created.

SQL> ALTER TABLE Admission_Details
     ADD CONSTRAINT Student_FK2
     FOREIGN KEY (Student_ID)
     REFERENCES Student(Student_ID);

Table altered.
```

Figure 23: Create Table Admission_Details

3.2. Populating Database

The INSERT INTO statement of SQL is used to insert a new row in a table. INSERT INTO … VALUES is the keyword telling the database system to place the values inside a table into its rows and columns according to how you want to enter them .There are two ways of using INSERT INTO statement for inserting rows:

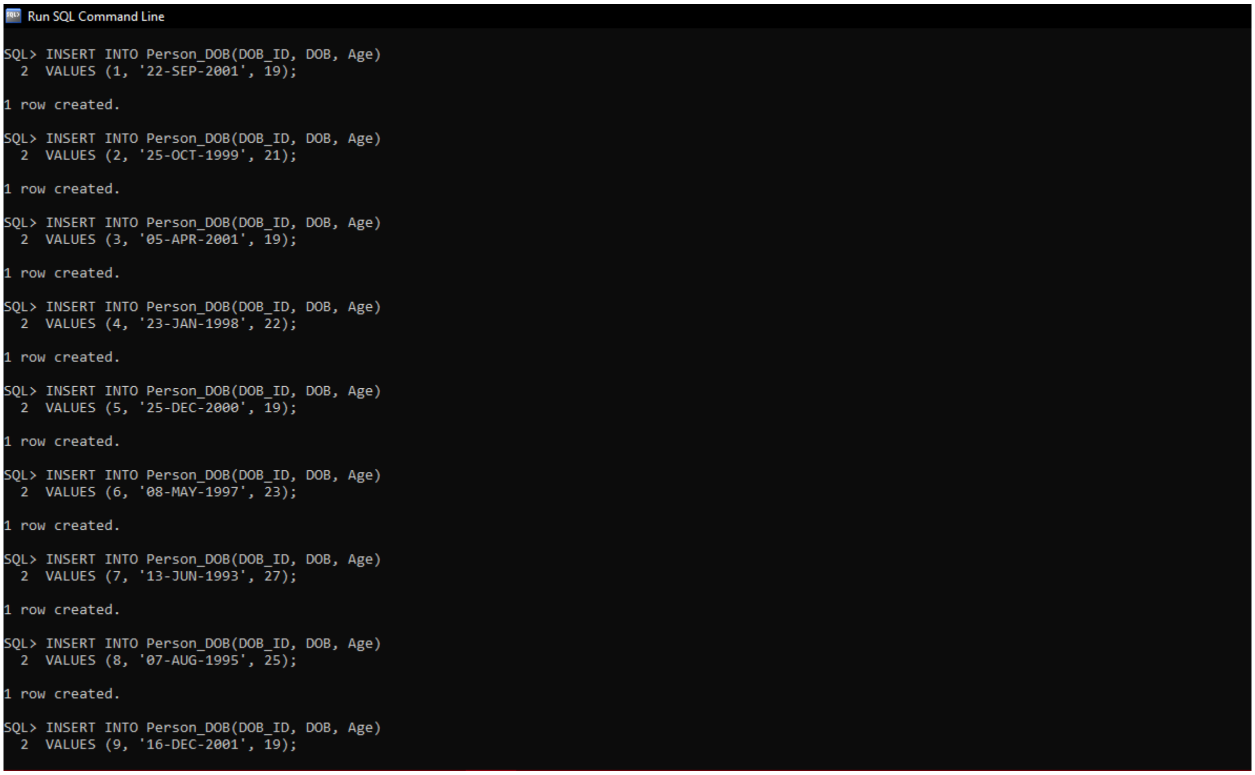
- **Only values:** First method is to specify only the value of data to be inserted without the column names. (GeeksforGeeks, 2019)
- **Column names and values both:** In the second method we will specify both the columns which we want to fill and their corresponding values. (GeeksforGeeks, 2019)

```
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (1, ‘22-SEP-2001’, 19);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (2, ‘25-Oct-1999’, 21);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (3, ‘05-Apr-2001’, 19);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (4, ‘16-Dec-2001’, 19);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (5, ’23-Jan-1998’, 22);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (6, ’25-Dec-2000’, 19);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (7, ’08-May-1997’, 23);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (8, ’13-Jun-1993’, 27);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (9, ’07-Aug-1995’, 25);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
VALUES (10, ’27-Nov-1992’, 28);
INSERT INTO Person_DOB(DOB_ID, DOB, Age)
```

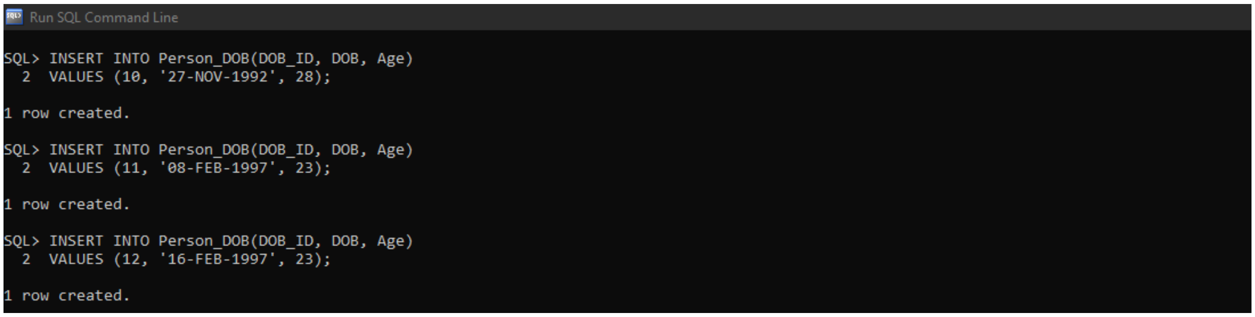
VALUES (11, ’08-Feb-1997’, 23);

INSERT INTO Person_DOB(DOB_ID, DOB, Age)

VALUES (12, ’16-Feb-1997’, 23);



```
SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (1, '22-SEP-2001', 19);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (2, '25-OCT-1999', 21);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (3, '05-APR-2001', 19);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (4, '23-JAN-1998', 22);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (5, '25-DEC-2000', 19);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (6, '08-MAY-1997', 23);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (7, '13-JUN-1993', 27);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (8, '07-AUG-1995', 25);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (9, '16-DEC-2001', 19);
```

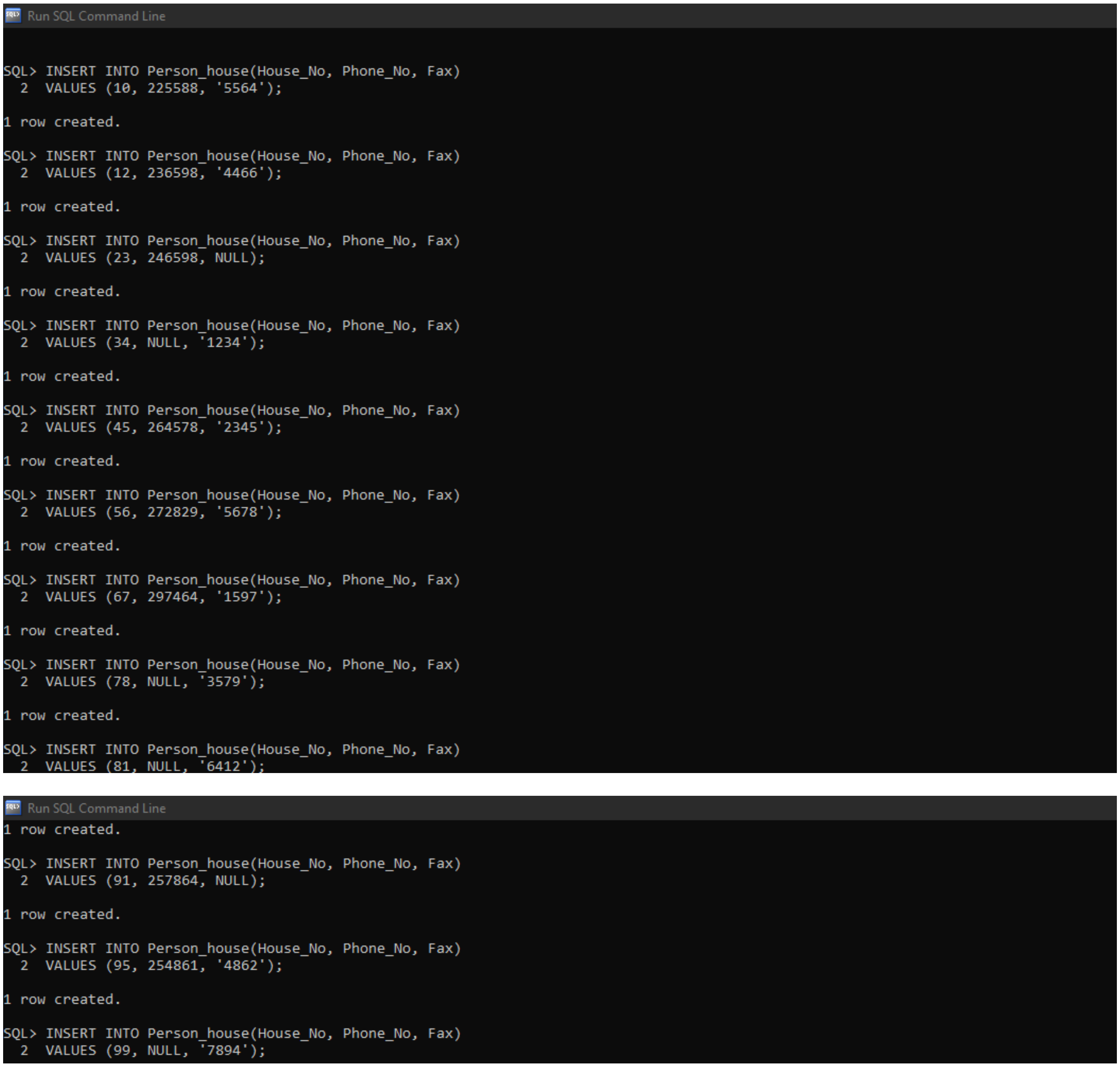


```
SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (10, '27-NOV-1992', 28);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (11, '08-FEB-1997', 23);

1 row created.

SQL> INSERT INTO Person_DOB(DOB_ID, DOB, Age)
  2  VALUES (12, '16-FEB-1997', 23);

1 row created.
```

Figure 24: Inserting to Person_DOB

Inserting Values to Person_house

```
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (10, 225588, ‘5564’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (12, 236598, ‘4466’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (23, 246598, NULL);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (34, 256867, ‘1234’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (45, 264578, ‘2345’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (56, 272829, ‘5678’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (67, 297464, ‘1597’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (78, 252498, ‘3579’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (81, 243159, ‘6412’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (91, 257864, ‘2684’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (95, 254861, ‘4862’);
INSERT INTO Person_house(House_No, Phone_No, Fax)
VALUES (99, 159648, ‘7894’);
```



```
SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (10, 225588, '5564');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (12, 236598, '4466');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (23, 246598, NULL);

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (34, NULL, '1234');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (45, 264578, '2345');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (56, 272829, '5678');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (67, 297464, '1597');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (78, NULL, '3579');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (81, NULL, '6412');
```

```
1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (91, 257864, NULL);

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (95, 254861, '4862');

1 row created.

SQL> INSERT INTO Person_house(House_No, Phone_No, Fax)
  2  VALUES (99, NULL, '7894');
```

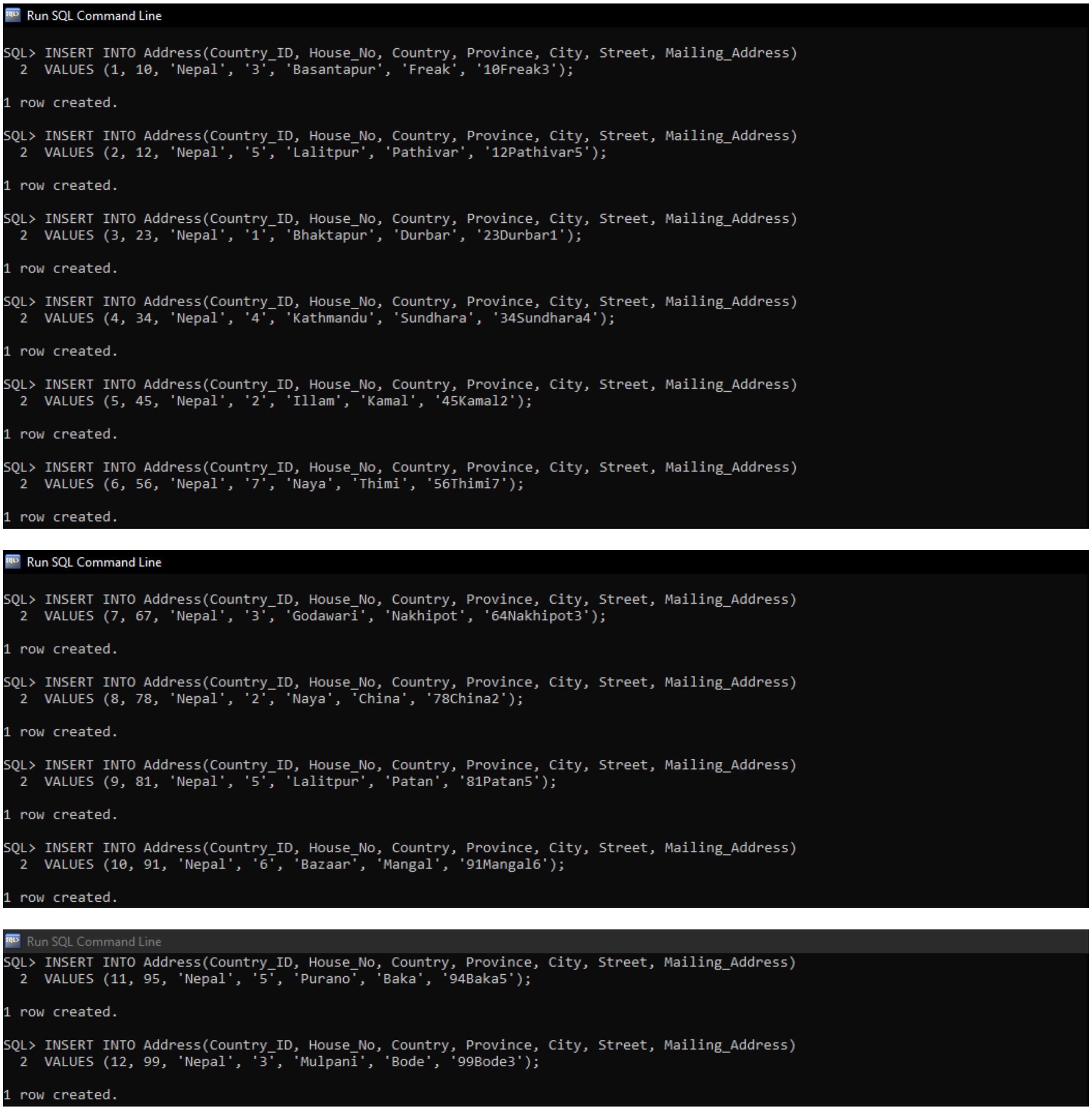
Figure 25: Inserting into Person_House

Inserting Values to Address

```
INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (1, 10, ‘Nepal’, ‘3’, ‘Basantapur’, ‘Freak’, ‘10Freak3’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (2, 12, ‘Nepal’, ‘5’, ‘Lalitpur’, ‘Pathivar’, ‘12Pathivar5’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (3, 23, ‘Nepal’, ‘1’, ‘Bhaktapur’, ‘Durbar’, ‘23Durbar1’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (4, 34, ‘Nepal’, ‘4’, ‘Kathmandu’, ‘Sundhara’, ‘34Sundhara4’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (5, 45, ‘Nepal’, ‘2’, ’Illam’, ‘Kamal’, ‘45Kamal2’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (6, 56, ‘Nepal’, ‘7’, ‘Naya’, ‘Thimi’, ‘56Thimi7’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (7, 67, ‘Nepal’, ‘3’, ‘Godawari’, ‘Nakhipot’, ‘64Nakhipot3’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (8, 78, ‘Nepal’, ‘2’, ‘Naya’, ‘China’, ‘78China2’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (9, 81, ‘Nepal’, ‘5’, ‘Lalitpur’, ‘Patan’, ‘81Patan5’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (10, 91, ‘Nepal’, ‘6’, ‘Bazaar’, ‘Mangal’, ‘91Mangal6’);

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street,
Mailing_Address)

VALUES (11, 95, ‘Nepal’, ‘5’, ‘Purano’, ‘Baka’, ‘94Baka5’);
```

INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)

VALUES (12, 99, ‘Nepal’, ‘3’, ‘Mulpani’, ‘Bode’, ‘99Bode3’);



```
SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (1, 10, 'Nepal', '3', 'Basantapur', 'Freak', '10Freak3');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (2, 12, 'Nepal', '5', 'Lalitpur', 'Pathivar', '12Pathivar5');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (3, 23, 'Nepal', '1', 'Bhaktapur', 'Durbar', '23Durbar1');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (4, 34, 'Nepal', '4', 'Kathmandu', 'Sundhara', '34Sundhara4');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (5, 45, 'Nepal', '2', 'Illam', 'Kamal', '45Kamal2');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (6, 56, 'Nepal', '7', 'Naya', 'Thimi', '56Thimi7');

1 row created.
```

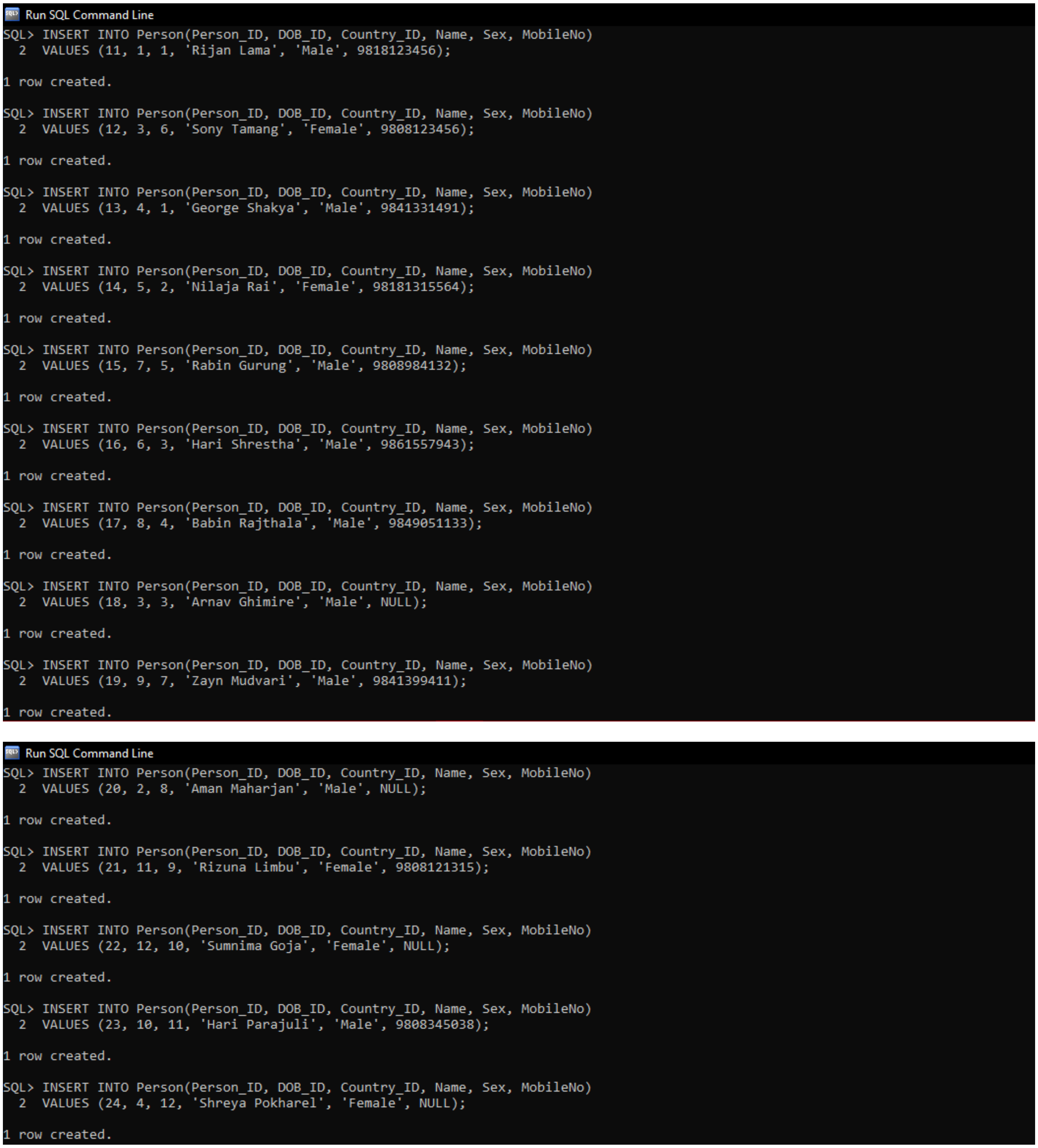
```
SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (7, 67, 'Nepal', '3', 'Godawari', 'Nakhipot', '64Nakhipot3');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (8, 78, 'Nepal', '2', 'Naya', 'China', '78China2');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (9, 81, 'Nepal', '5', 'Lalitpur', 'Patan', '81Patan5');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (10, 91, 'Nepal', '6', 'Bazaar', 'Mangal', '91Mangal6');

1 row created.
```

```
SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (11, 95, 'Nepal', '5', 'Purano', 'Baka', '94Baka5');

1 row created.

SQL> INSERT INTO Address(Country_ID, House_No, Country, Province, City, Street, Mailing_Address)
  2  VALUES (12, 99, 'Nepal', '3', 'Mulpani', 'Bode', '99Bode3');

1 row created.
```

Figure 26: Inserting to Address

Inserting Values to Person

```
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (11, 1, 1, ‘Rijan Lama’, ‘Male’, 9818123456);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (12, 3, 6, ‘Sony Tamang’, ‘Female’, 9808123456);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (13, 4, 1, ‘George Shakya’, ‘Male’, 9841331491);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (14, 5, 2, ‘Nilaja Rai’, ‘Female’, 9818131564);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (15, 7, 5, ‘Rabin Gurung’, ‘Male’, 9808984132);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (16, 6, 3, ‘Hari Shrestha’, ‘Male’, 9861557943);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (17, 8, 4, ‘Babin Rajthala’, ‘Male’, 9849051133);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (18, 3, 3, ‘Arnav Ghimire’, ‘Male’, NULL);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (19, 9, 7, ‘Zayn Mudvari’, ‘Male’, 9841399411);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (20, 2, 8, ‘Aman Maharjan’, ‘Male’, NULL);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (21, 11, 9, ‘Rizuna Limbu’, ‘Female’, 9808121315);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (22, 12, 10, ‘Sumnima Goja’, ‘Female’, NULL);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (23, 10, 11, ‘Hari Parajuli’, ‘Male’, 9808345038);
INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
VALUES (24, 4, 12, ‘Shreya Pokharel’, ‘Female’, NULL);
```



```
SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (11, 1, 1, 'Rijan Lama', 'Male', 9818123456);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (12, 3, 6, 'Sony Tamang', 'Female', 9808123456);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (13, 4, 1, 'George Shakya', 'Male', 9841331491);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (14, 5, 2, 'Nilaja Rai', 'Female', 98181315564);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (15, 7, 5, 'Rabin Gurung', 'Male', 9808984132);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (16, 6, 3, 'Hari Shrestha', 'Male', 9861557943);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (17, 8, 4, 'Babin Rajthala', 'Male', 9849051133);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (18, 3, 3, 'Arnav Ghimire', 'Male', NULL);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (19, 9, 7, 'Zayn Mudvari', 'Male', 9841399411);

1 row created.
```

```
SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (20, 2, 8, 'Aman Maharjan', 'Male', NULL);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (21, 11, 9, 'Rizuna Limbu', 'Female', 9808121315);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (22, 12, 10, 'Sumnima Goja', 'Female', NULL);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (23, 10, 11, 'Hari Parajuli', 'Male', 9808345038);

1 row created.

SQL> INSERT INTO Person(Person_ID, DOB_ID, Country_ID, Name, Sex, MobileNo)
  2  VALUES (24, 4, 12, 'Shreya Pokharel', 'Female', NULL);

1 row created.
```

Figure 27: Inserting into Person

Inserting Values to Student

INSERT INTO Student(Student_ID, Mark)

VALUES (21, 70);

INSERT INTO Student(Student_ID, Mark)

VALUES (22, 75);

INSERT INTO Student(Student_ID, Mark)

VALUES (23, 80);

INSERT INTO Student(Student_ID, Mark)

VALUES (24, 90);

INSERT INTO Student(Student_ID, Mark)

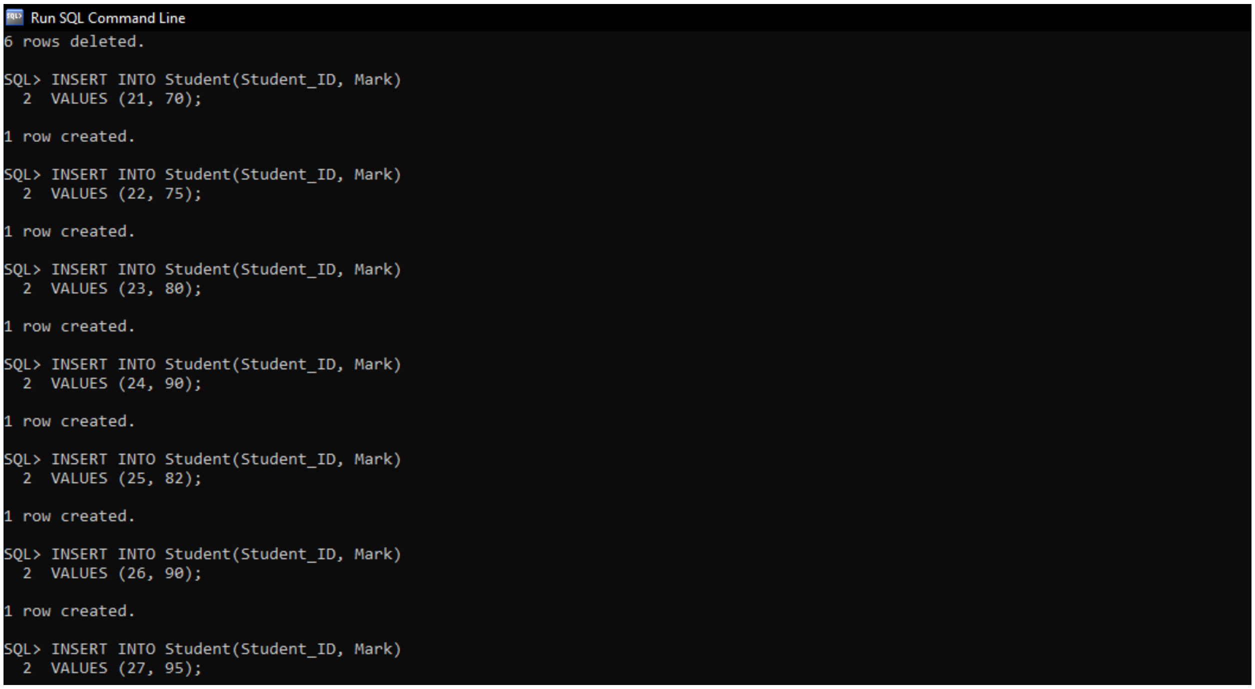
VALUES (25, 82);

INSERT INTO Student(Student_ID, Mark)

VALUES (26, 90);

INSERT INTO Student(Student_ID, Mark)

VALUES (27, 95);



```
6 rows deleted.

SQL> INSERT INTO Student(Student_ID, Mark)
  2  VALUES (21, 70);

1 row created.

SQL> INSERT INTO Student(Student_ID, Mark)
  2  VALUES (22, 75);

1 row created.

SQL> INSERT INTO Student(Student_ID, Mark)
  2  VALUES (23, 80);

1 row created.

SQL> INSERT INTO Student(Student_ID, Mark)
  2  VALUES (24, 90);

1 row created.

SQL> INSERT INTO Student(Student_ID, Mark)
  2  VALUES (25, 82);

1 row created.

SQL> INSERT INTO Student(Student_ID, Mark)
  2  VALUES (26, 90);

1 row created.

SQL> INSERT INTO Student(Student_ID, Mark)
  2  VALUES (27, 95);
```

Figure 28: Inserting into Student

Inserting Values to Student_Details

INSERT INTO Student_Details(Student_ID, Person_ID)

VALUES (21, 11);

INSERT INTO Student_Details(Student_ID, Person_ID)

VALUES (22, 12);

INSERT INTO Student_Details(Student_ID, Person_ID)

VALUES (23, 13);

INSERT INTO Student_Details(Student_ID, Person_ID)

VALUES (24, 16);

INSERT INTO Student_Details(Student_ID, Person_ID)

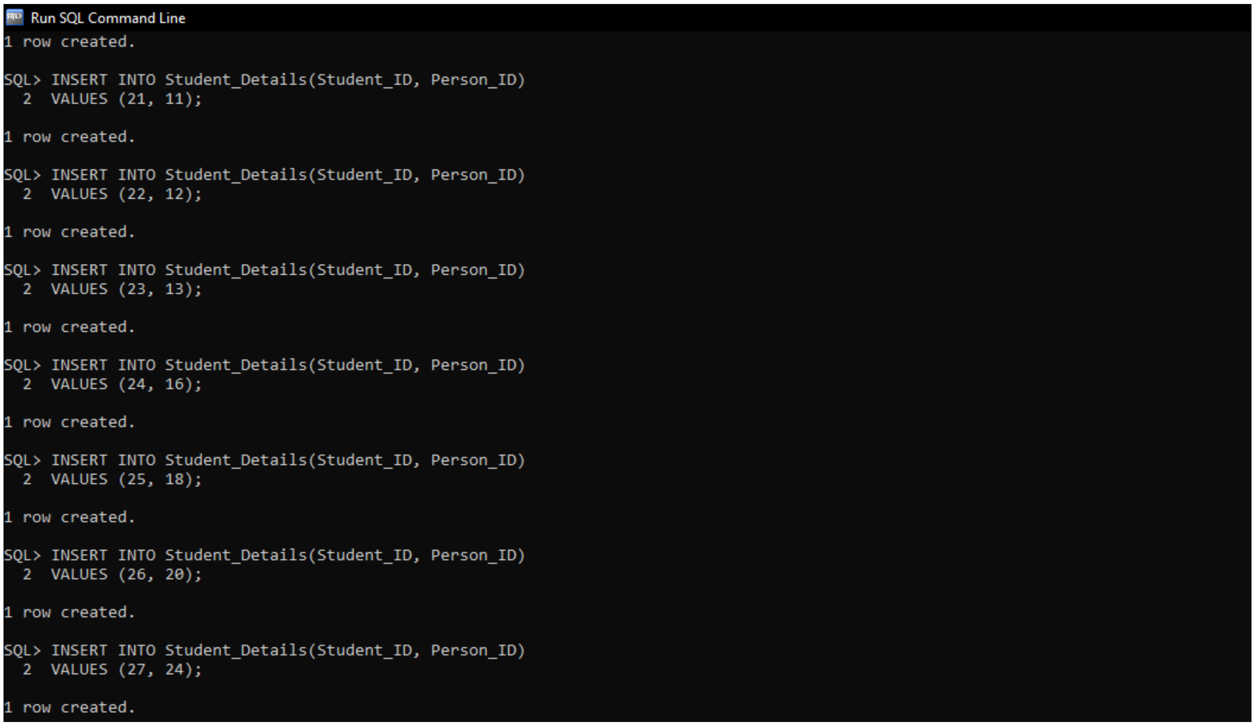
VALUES (25, 18);

INSERT INTO Student_Details(Student_ID, Person_ID)

VALUES (26, 20);

INSERT INTO Student_Details(Student_ID, Person_ID)

VALUES (27, 24);



```
1 row created.

SQL> INSERT INTO Student_Details(Student_ID, Person_ID)
  2  VALUES (21, 11);

1 row created.

SQL> INSERT INTO Student_Details(Student_ID, Person_ID)
  2  VALUES (22, 12);

1 row created.

SQL> INSERT INTO Student_Details(Student_ID, Person_ID)
  2  VALUES (23, 13);

1 row created.

SQL> INSERT INTO Student_Details(Student_ID, Person_ID)
  2  VALUES (24, 16);

1 row created.

SQL> INSERT INTO Student_Details(Student_ID, Person_ID)
  2  VALUES (25, 18);

1 row created.

SQL> INSERT INTO Student_Details(Student_ID, Person_ID)
  2  VALUES (26, 20);

1 row created.

SQL> INSERT INTO Student_Details(Student_ID, Person_ID)
  2  VALUES (27, 24);

1 row created.
```

Figure 29: Inserting into Student_Details

Inserting Values to Instructor_Salary

INSERT INTO Instructor_Salary(Instructor_Type, Salary)

VALUES (‘Course Leader’, 75000);

INSERT INTO Instructor_Salary(Instructor_Type, Salary)

VALUES (‘Modue Leader’, 60000);

INSERT INTO Instructor_Salary(Instructor_Type, Salary)

VALUES (‘Lecturer’, 45000);

INSERT INTO Instructor_Salary(Instructor_Type, Salary)

VALUES (‘Tutorial Mentor’, 49000);

INSERT INTO Instructor_Salary(Instructor_Type, Salary)

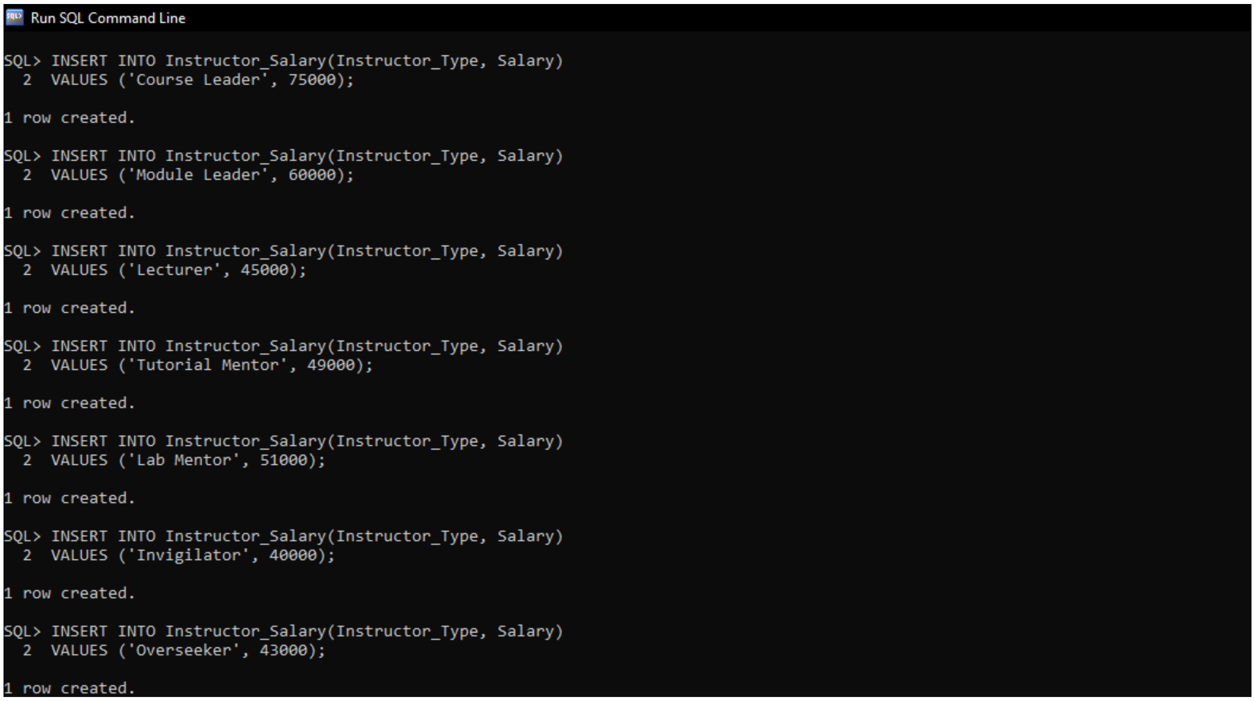
VALUES (‘Lab Mentor’, 51000);

INSERT INTO Instructor_Salary(Instructor_Type, Salary)

VALUES (‘Invigilator’, 40000);

INSERT INTO Instructor_Salary(Instructor_Type, Salary)

VALUES (‘Overseeker’, 43000);



```
SQL> INSERT INTO Instructor_Salary(Instructor_Type, Salary)
  2  VALUES ('Course Leader', 75000);

1 row created.

SQL> INSERT INTO Instructor_Salary(Instructor_Type, Salary)
  2  VALUES ('Module Leader', 60000);

1 row created.

SQL> INSERT INTO Instructor_Salary(Instructor_Type, Salary)
  2  VALUES ('Lecturer', 45000);

1 row created.

SQL> INSERT INTO Instructor_Salary(Instructor_Type, Salary)
  2  VALUES ('Tutorial Mentor', 49000);

1 row created.

SQL> INSERT INTO Instructor_Salary(Instructor_Type, Salary)
  2  VALUES ('Lab Mentor', 51000);

1 row created.

SQL> INSERT INTO Instructor_Salary(Instructor_Type, Salary)
  2  VALUES ('Invigilator', 40000);

1 row created.

SQL> INSERT INTO Instructor_Salary(Instructor_Type, Salary)
  2  VALUES ('Overseeker', 43000);

1 row created.
```

Figure 30: Inserting into Instructor_Salary

Inserting Values to Instructor

INSERT INTO Instructor(Instructor_ID, Instructor_Type)

VALUES (11, ‘Course Leader’);

INSERT INTO Instructor(Instructor_ID, Instructor_Type)

VALUES (12, ‘Module Leader’);

INSERT INTO Instructor(Instructor_ID, Instructor_Type)

VALUES (13, ‘Course Leader’);

INSERT INTO Instructor(Instructor_ID, Instructor_Type)

VALUES (14, ‘Module Leader’);

INSERT INTO Instructor(Instructor_ID, Instructor_Type)

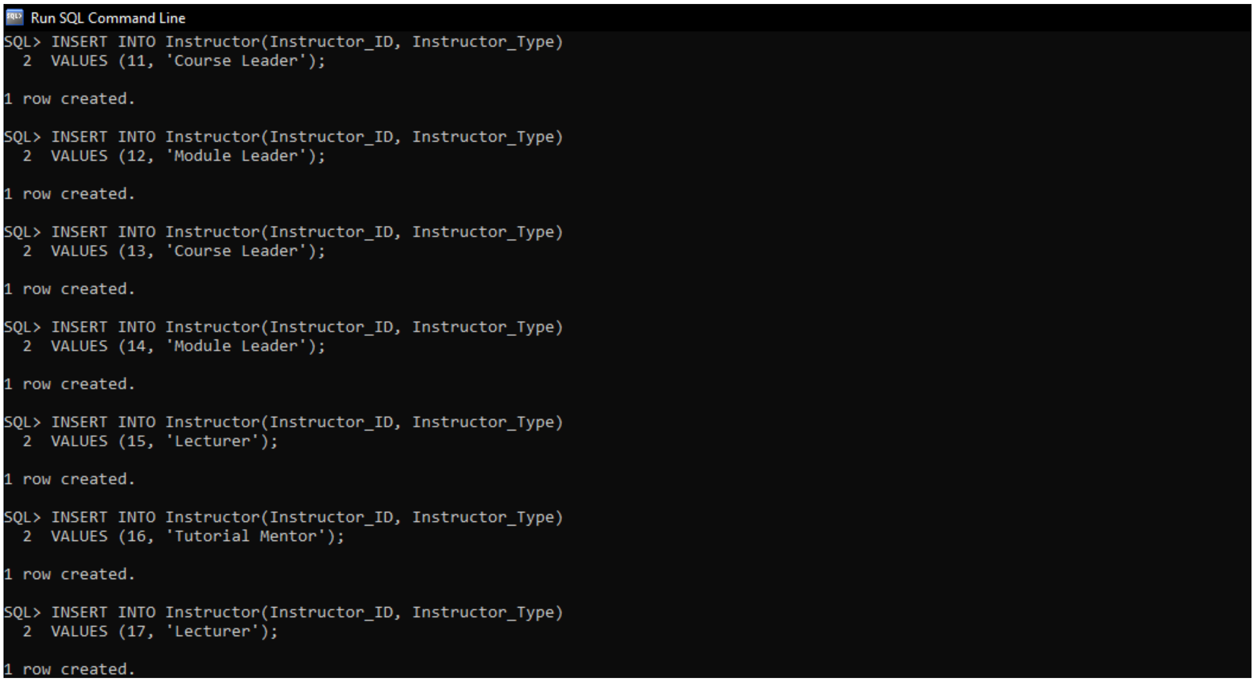
VALUES (15, ‘Lecturer’);

INSERT INTO Instructor(Instructor_ID, Instructor_Type)

VALUES (16, ‘Tutorial Mentor’);

INSERT INTO Instructor(Instructor_ID, Instructor_Type)

VALUES (17, ‘Lecturer’);



```
SQL> INSERT INTO Instructor(Instructor_ID, Instructor_Type)
  2  VALUES (11, 'Course Leader');

1 row created.

SQL> INSERT INTO Instructor(Instructor_ID, Instructor_Type)
  2  VALUES (12, 'Module Leader');

1 row created.

SQL> INSERT INTO Instructor(Instructor_ID, Instructor_Type)
  2  VALUES (13, 'Course Leader');

1 row created.

SQL> INSERT INTO Instructor(Instructor_ID, Instructor_Type)
  2  VALUES (14, 'Module Leader');

1 row created.

SQL> INSERT INTO Instructor(Instructor_ID, Instructor_Type)
  2  VALUES (15, 'Lecturer');

1 row created.

SQL> INSERT INTO Instructor(Instructor_ID, Instructor_Type)
  2  VALUES (16, 'Tutorial Mentor');

1 row created.

SQL> INSERT INTO Instructor(Instructor_ID, Instructor_Type)
  2  VALUES (17, 'Lecturer');

1 row created.
```

Figure 31: Inserting into Instructor

Inserting Values to Instructor_Details

INSERT INTO Instructor_Details(Instructor_ID, Person_ID)

VALUES (11, 21);

INSERT INTO Instructor_Details(Instructor_ID, Person_ID)

VALUES (12, 14);

INSERT INTO Instructor_Details(Instructor_ID, Person_ID)

VALUES (13, 22);

INSERT INTO Instructor_Details(Instructor_ID, Person_ID)

VALUES (14, 19);

INSERT INTO Instructor_Details(Instructor_ID, Person_ID)

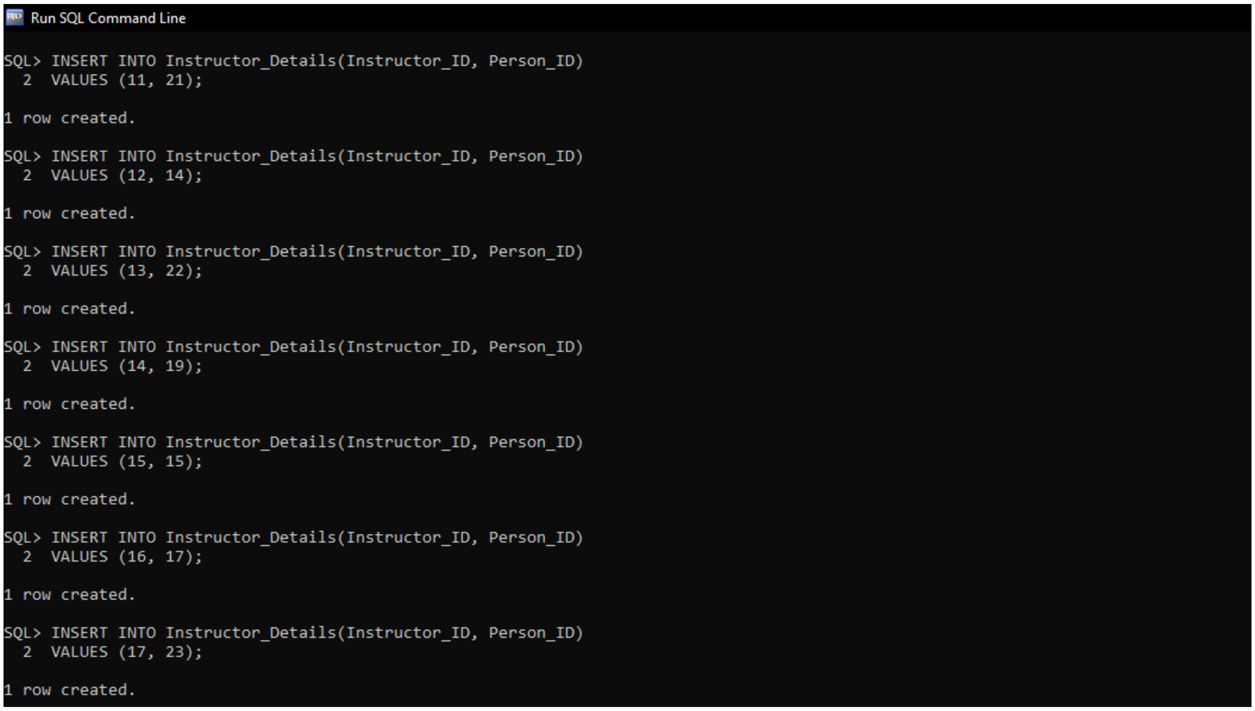
VALUES (15, 15);

INSERT INTO Instructor_Details(Instructor_ID, Person_ID)

VALUES (16, 17);

INSERT INTO Instructor_Details(Instructor_ID, Person_ID)

VALUES (17, 23);



```
SQL> INSERT INTO Instructor_Details(Instructor_ID, Person_ID)
  2  VALUES (11, 21);

1 row created.

SQL> INSERT INTO Instructor_Details(Instructor_ID, Person_ID)
  2  VALUES (12, 14);

1 row created.

SQL> INSERT INTO Instructor_Details(Instructor_ID, Person_ID)
  2  VALUES (13, 22);

1 row created.

SQL> INSERT INTO Instructor_Details(Instructor_ID, Person_ID)
  2  VALUES (14, 19);

1 row created.

SQL> INSERT INTO Instructor_Details(Instructor_ID, Person_ID)
  2  VALUES (15, 15);

1 row created.

SQL> INSERT INTO Instructor_Details(Instructor_ID, Person_ID)
  2  VALUES (16, 17);

1 row created.

SQL> INSERT INTO Instructor_Details(Instructor_ID, Person_ID)
  2  VALUES (17, 23);

1 row created.
```

Figure 32: Insertig into Instructor_Details

Inserting Values to Class

INSERT INTO Class(Class_ID, Room_No)

VALUES (10, 101);

INSERT INTO Class(Class_ID, Room_No)

VALUES (20, 102);

INSERT INTO Class(Class_ID, Room_No)

VALUES (30, 201);

INSERT INTO Class(Class_ID, Room_No)

VALUES (40, 202);

INSERT INTO Class(Class_ID, Room_No)

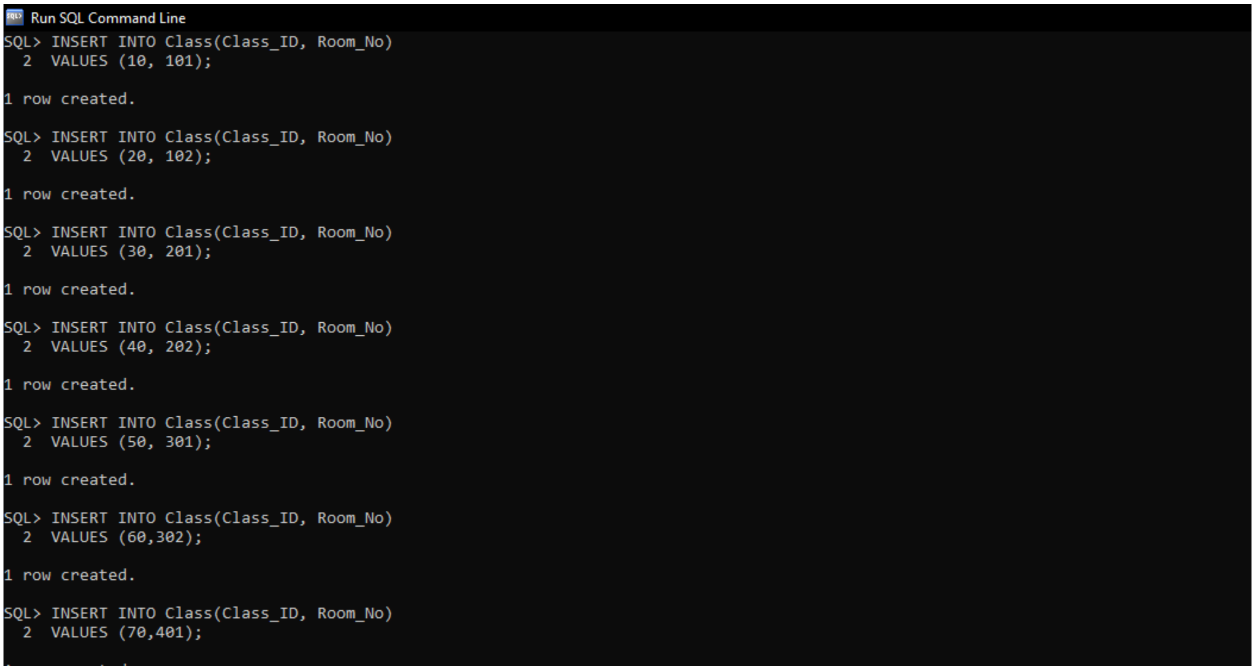
VALUES (50, 301);

INSERT INTO Class(Class_ID, Room_No)

VALUES (60,302);

INSERT INTO Class(Class_ID, Room_No)

VALUES (70,401);



```
SQL> INSERT INTO Class(Class_ID, Room_No)
  2  VALUES (10, 101);

1 row created.

SQL> INSERT INTO Class(Class_ID, Room_No)
  2  VALUES (20, 102);

1 row created.

SQL> INSERT INTO Class(Class_ID, Room_No)
  2  VALUES (30, 201);

1 row created.

SQL> INSERT INTO Class(Class_ID, Room_No)
  2  VALUES (40, 202);

1 row created.

SQL> INSERT INTO Class(Class_ID, Room_No)
  2  VALUES (50, 301);

1 row created.

SQL> INSERT INTO Class(Class_ID, Room_No)
  2  VALUES (60,302);

1 row created.

SQL> INSERT INTO Class(Class_ID, Room_No)
  2  VALUES (70,401);
```

Figure 33: Inserting into Class

Inserting Values to Session_Class

INSERT INTO Session_Class(Session_ID, Class_ID)

VALUES (1, 10);

INSERT INTO Session_Class(Session_ID, Class_ID)

VALUES (2, 20);

INSERT INTO Session_Class(Session_ID, Class_ID)

VALUES (3, 30);

INSERT INTO Session_Class(Session_ID, Class_ID)

VALUES (4, 40);

INSERT INTO Session_Class(Session_ID, Class_ID)

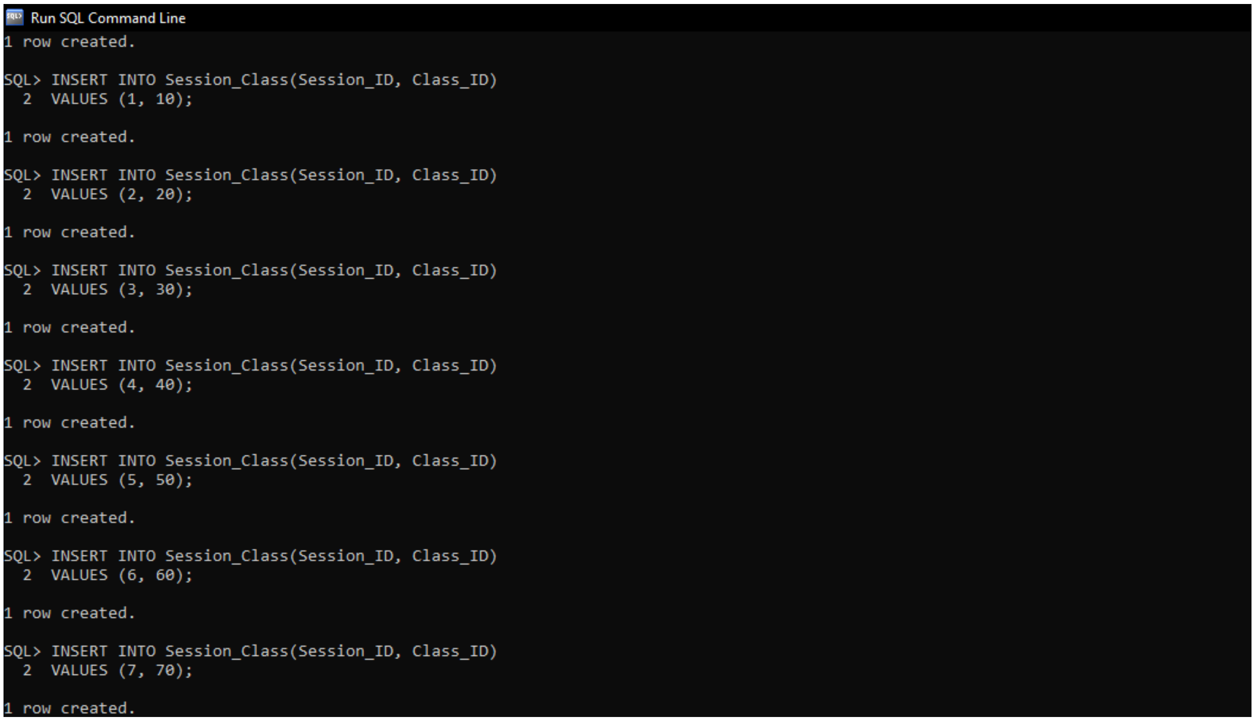
VALUES (5, 50);

INSERT INTO Session_Class(Session_ID, Class_ID)

VALUES (6, 60);

INSERT INTO Session_Class(Session_ID, Class_ID)

VALUES (7, 70);



```
1 row created.

SQL> INSERT INTO Session_Class(Session_ID, Class_ID)
  2  VALUES (1, 10);

1 row created.

SQL> INSERT INTO Session_Class(Session_ID, Class_ID)
  2  VALUES (2, 20);

1 row created.

SQL> INSERT INTO Session_Class(Session_ID, Class_ID)
  2  VALUES (3, 30);

1 row created.

SQL> INSERT INTO Session_Class(Session_ID, Class_ID)
  2  VALUES (4, 40);

1 row created.

SQL> INSERT INTO Session_Class(Session_ID, Class_ID)
  2  VALUES (5, 50);

1 row created.

SQL> INSERT INTO Session_Class(Session_ID, Class_ID)
  2  VALUES (6, 60);

1 row created.

SQL> INSERT INTO Session_Class(Session_ID, Class_ID)
  2  VALUES (7, 70);

1 row created.
```

Figure 34: Inserting into Session_Class

Inserting Values to Module

INSERT INTO Module(Module_ID, Module_Name)

VALUES (2, ‘Software Engineering’);

INSERT INTO Module(Module_ID, Module_Name)

VALUES (3, ‘Database’);

INSERT INTO Module(Module_ID, Module_Name)

VALUES (7, ‘Security’);

INSERT INTO Module(Module_ID, Module_Name)

VALUES (8, ‘3D Modelling’);

INSERT INTO Module(Module_ID, Module_Name)

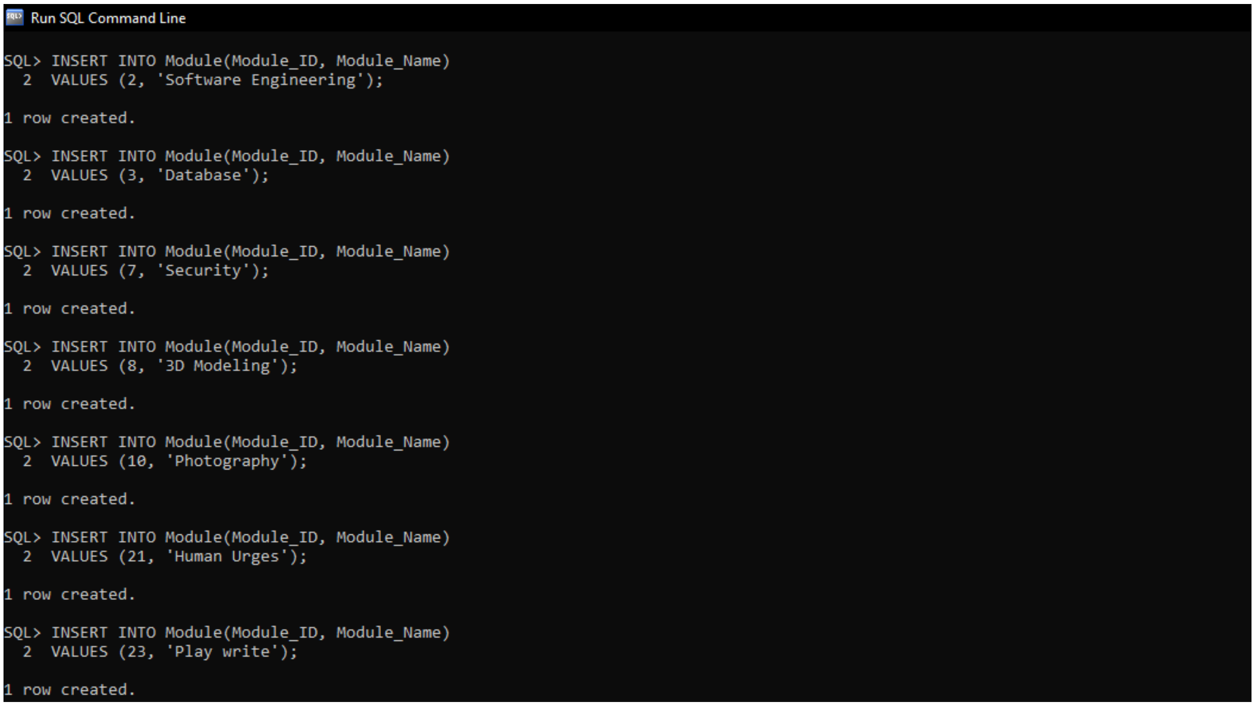
VALUES (10, ‘Photography’);

INSERT INTO Module(Module_ID, Module_Name)

VALUES (21, ‘Human urges’);

INSERT INTO Module(Module_ID, Module_Name)

VALUES (23, ‘Play write’);



```
SQL> INSERT INTO Module(Module_ID, Module_Name)
  2  VALUES (2, 'Software Engineering');

1 row created.

SQL> INSERT INTO Module(Module_ID, Module_Name)
  2  VALUES (3, 'Database');

1 row created.

SQL> INSERT INTO Module(Module_ID, Module_Name)
  2  VALUES (7, 'Security');

1 row created.

SQL> INSERT INTO Module(Module_ID, Module_Name)
  2  VALUES (8, '3D Modeling');

1 row created.

SQL> INSERT INTO Module(Module_ID, Module_Name)
  2  VALUES (10, 'Photography');

1 row created.

SQL> INSERT INTO Module(Module_ID, Module_Name)
  2  VALUES (21, 'Human Urges');

1 row created.

SQL> INSERT INTO Module(Module_ID, Module_Name)
  2  VALUES (23, 'Play write');

1 row created.
```

Figure 35: Inserting into Module

Inserting Values to Session_Module

INSERT ALL

INTO Session_Module(Session_ID, Module_ID) VALUES (1, 2)

INTO Session_Module(Session_ID, Module_ID) VALUES (2, 3)

INTO Session_Module(Session_ID, Module_ID) VALUES (3, 7)

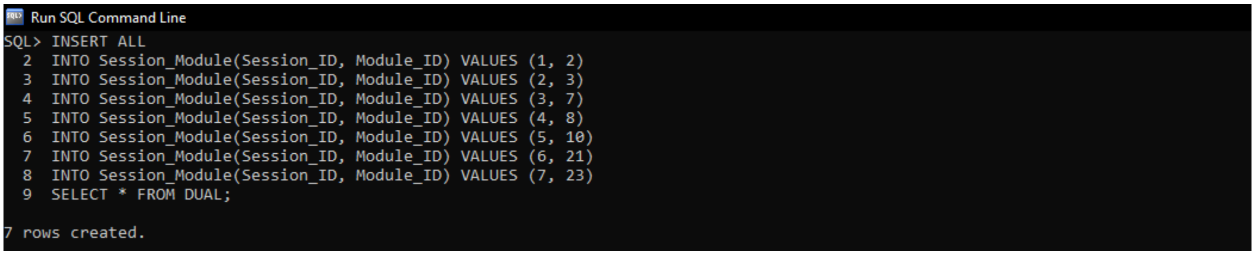
INTO Session_Module(Session_ID, Module_ID) VALUES (4, 8)

INTO Session_Module(Session_ID, Module_ID) VALUES (5, 10)

INTO Session_Module(Session_ID, Module_ID) VALUES (6, 21)

INTO Session_Module(Session_ID, Module_ID) VALUES (7, 23)

SELECT * FROM DUAL;



```
SQL> INSERT ALL
     INTO Session_Module(Session_ID, Module_ID) VALUES (1, 2)
     INTO Session_Module(Session_ID, Module_ID) VALUES (2, 3)
     INTO Session_Module(Session_ID, Module_ID) VALUES (3, 7)
     INTO Session_Module(Session_ID, Module_ID) VALUES (4, 8)
     INTO Session_Module(Session_ID, Module_ID) VALUES (5, 10)
     INTO Session_Module(Session_ID, Module_ID) VALUES (6, 21)
     INTO Session_Module(Session_ID, Module_ID) VALUES (7, 23)
     SELECT * FROM DUAL;

7 rows created.
```

Figure 36: Inserting into Session_Module

Inserting Values to Module_Details

INSERT INTO Module_Details(Module_ID, Instuctor_ID)

VALUES (2, 16);

INSERT INTO Module_Details(Module_ID, Instuctor_ID)

VALUES (3, 12);

INSERT INTO Module_Details(Module_ID, Instuctor_ID)

VALUES (7, 14);

INSERT INTO Module_Details(Module_ID, Instuctor_ID)

VALUES (8, 13);

INSERT INTO Module_Details(Module_ID, Instuctor_ID)

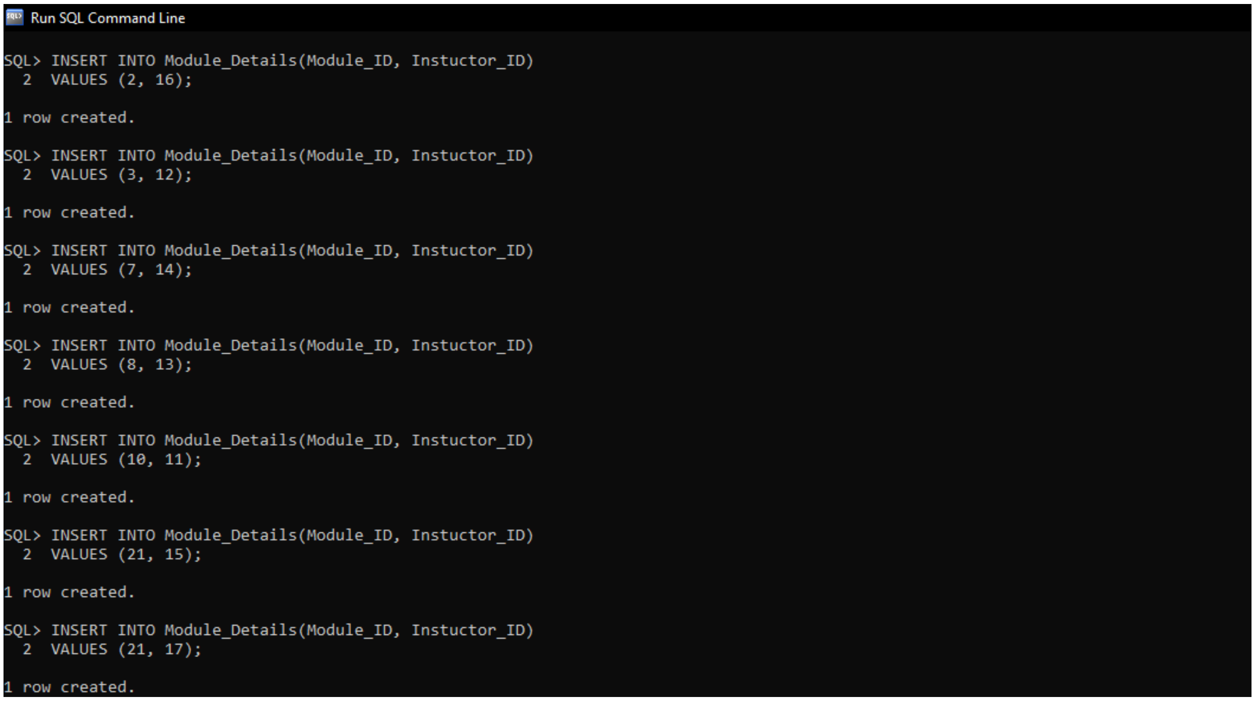
VALUES (10, 11);

INSERT INTO Module_Details(Module_ID, Instuctor_ID)

VALUES (21, 15);

INSERT INTO Module_Details(Module_ID, Instuctor_ID)

VALUES (21, 17);



```
SQL> INSERT INTO Module_Details(Module_ID, Instuctor_ID)
  2  VALUES (2, 16);

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Instuctor_ID)
  2  VALUES (3, 12);

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Instuctor_ID)
  2  VALUES (7, 14);

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Instuctor_ID)
  2  VALUES (8, 13);

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Instuctor_ID)
  2  VALUES (10, 11);

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Instuctor_ID)
  2  VALUES (21, 15);

1 row created.

SQL> INSERT INTO Module_Details(Module_ID, Instuctor_ID)
  2  VALUES (21, 17);

1 row created.
```

Figure 37: Inserting into Moduel_Details

Inserting Values to Specification

INSERT INTO Specification(Spec_ID, Spec_Name)

VALUES (3, ‘Computing’);

INSERT INTO Specification(Spec_ID, Spec_Name)

VALUES (4, ‘Networking’);

INSERT INTO Specification(Spec_ID, Spec_Name)

VALUES (5, ‘Multimedia’);

INSERT INTO Specification(Spec_ID, Spec_Name)

VALUES (10, ‘Anthropology’);

INSERT INTO Specification(Spec_ID, Spec_Name)

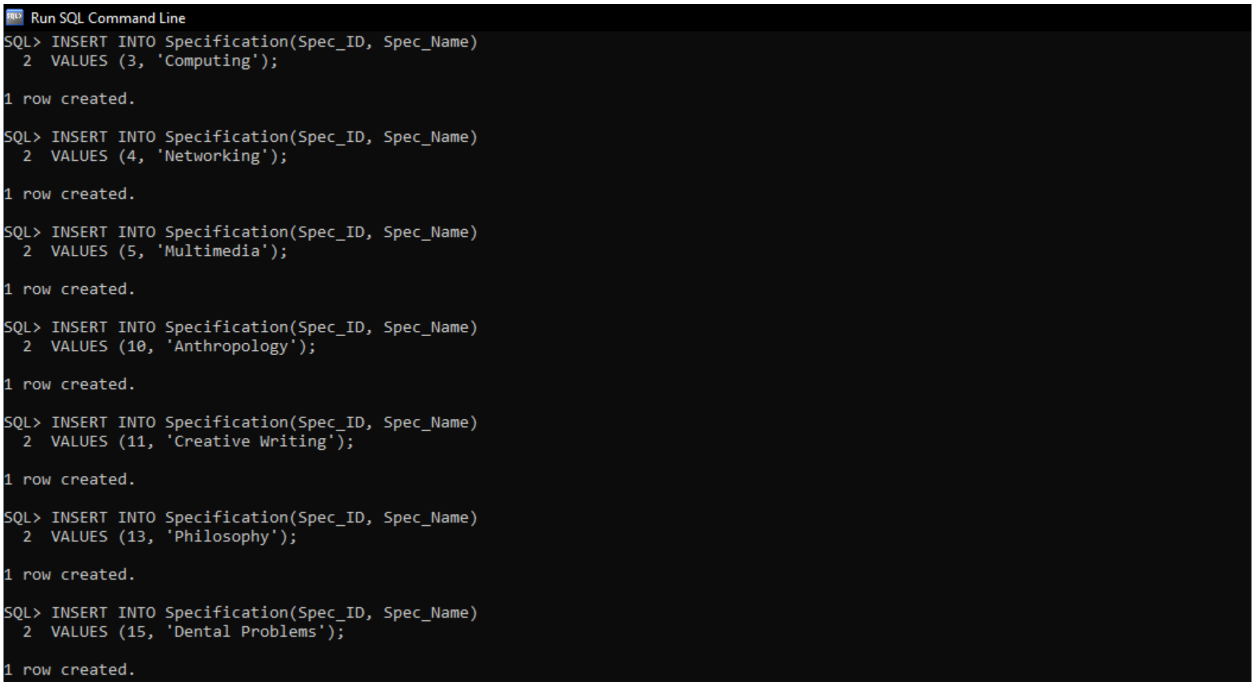
VALUES (11, ‘Creative Writing’);

INSERT INTO Specification(Spec_ID, Spec_Name)

VALUES (13, ‘Philosophy’);

INSERT INTO Specification(Spec_ID, Spec_Name)

VALUES (15, ‘Dental Problems’);



```
SQL> INSERT INTO Specification(Spec_ID, Spec_Name)
  2  VALUES (3, 'Computing');

1 row created.

SQL> INSERT INTO Specification(Spec_ID, Spec_Name)
  2  VALUES (4, 'Networking');

1 row created.

SQL> INSERT INTO Specification(Spec_ID, Spec_Name)
  2  VALUES (5, 'Multimedia');

1 row created.

SQL> INSERT INTO Specification(Spec_ID, Spec_Name)
  2  VALUES (10, 'Anthropology');

1 row created.

SQL> INSERT INTO Specification(Spec_ID, Spec_Name)
  2  VALUES (11, 'Creative Writing');

1 row created.

SQL> INSERT INTO Specification(Spec_ID, Spec_Name)
  2  VALUES (13, 'Philosophy');

1 row created.

SQL> INSERT INTO Specification(Spec_ID, Spec_Name)
  2  VALUES (15, 'Dental Problems');

1 row created.
```

Figure 38: Inserting into Specification

Inserting Values to Specification_Details

INSERT ALL

INTO Specification_Details(Spec_ID, Module_ID) VALUES (3, 2)

INTO Specification_Details(Spec_ID, Module_ID) VALUES (3, 3)

INTO Specification_Details(Spec_ID, Module_ID) VALUES (4, 7)

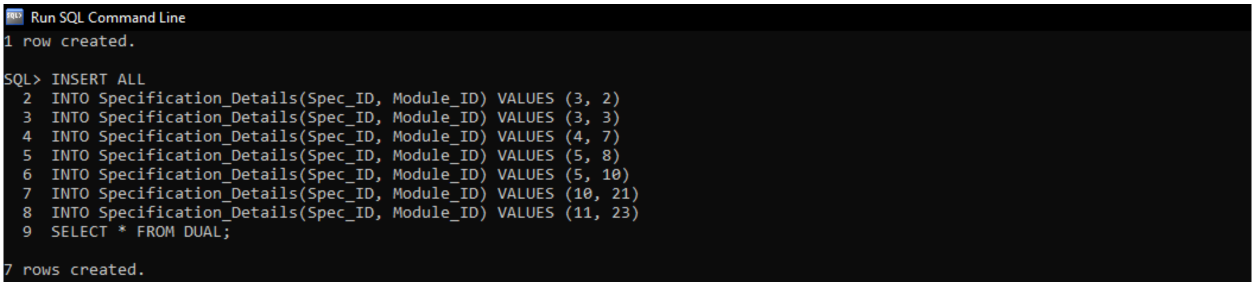
INTO Specification_Details(Spec_ID, Module_ID) VALUES (5, 8)

INTO Specification_Details(Spec_ID, Module_ID) VALUES (5, 10)

INTO Specification_Details(Spec_ID, Module_ID) VALUES (10, 21)

INTO Specification_Details(Spec_ID, Module_ID) VALUES (11, 23)

SELECT * FROM DUAL;

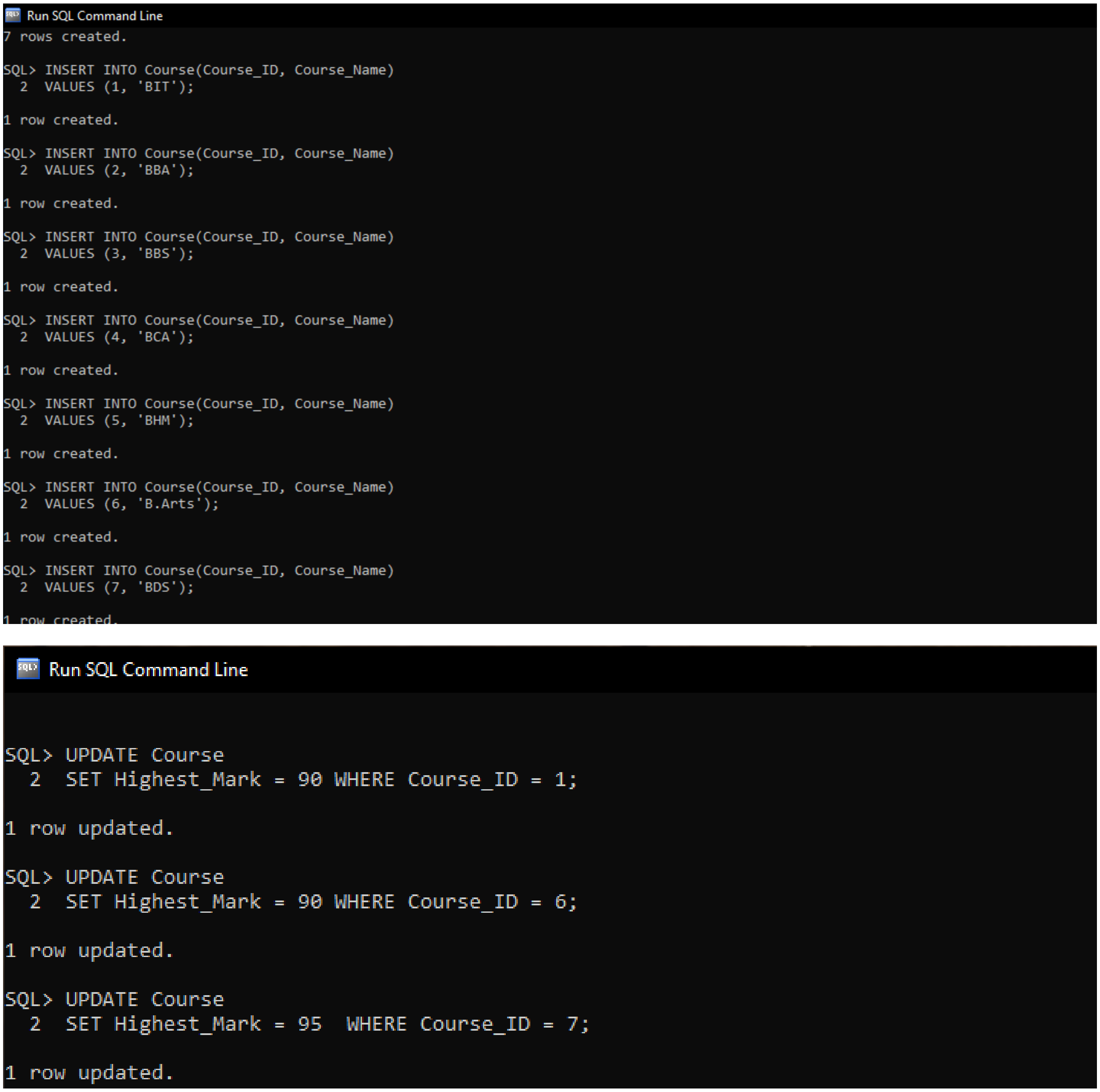


```
1 row created.

SQL> INSERT ALL
  2  INTO Specification_Details(Spec_ID, Module_ID) VALUES (3, 2)
  3  INTO Specification_Details(Spec_ID, Module_ID) VALUES (3, 3)
  4  INTO Specification_Details(Spec_ID, Module_ID) VALUES (4, 7)
  5  INTO Specification_Details(Spec_ID, Module_ID) VALUES (5, 8)
  6  INTO Specification_Details(Spec_ID, Module_ID) VALUES (5, 10)
  7  INTO Specification_Details(Spec_ID, Module_ID) VALUES (10, 21)
  8  INTO Specification_Details(Spec_ID, Module_ID) VALUES (11, 23)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 39: Inserting to Specification_Details

Inserting Values to Course

```
INSERT INTO Course(Course_ID, Course_Name)
VALUES (1, ‘BIT’);
INSERT INTO Course(Course_ID, Course_Name)
VALUES (2, ‘BBA’);
INSERT INTO Course(Course_ID, Course_Name)
VALUES (3, ‘BBS’);
INSERT INTO Course(Course_ID, Course_Name)
VALUES (4, ‘BCA’);
INSERT INTO Course(Course_ID, Course_Name)
VALUES (5, ‘BHM’
INSERT INTO Course(Course_ID, Course_Name)
VALUES (6, ‘B.Arts’);
INSERT INTO Course(Course_ID, Course_Name)
VALUES (7, ‘BDS’);

UPDATE Course
SET Highest_Mark = 90 WHERE Course_ID = 1;
UPDATE Course
SET Highest_Mark = 90 WHERE Course_ID = 6;
UPDATE Course
SET Highest_Mark = 95  WHERE Course_ID = 7;
```



```
7 rows created.

SQL> INSERT INTO Course(Course_ID, Course_Name)
  2  VALUES (1, 'BIT');

1 row created.

SQL> INSERT INTO Course(Course_ID, Course_Name)
  2  VALUES (2, 'BBA');

1 row created.

SQL> INSERT INTO Course(Course_ID, Course_Name)
  2  VALUES (3, 'BBS');

1 row created.

SQL> INSERT INTO Course(Course_ID, Course_Name)
  2  VALUES (4, 'BCA');

1 row created.

SQL> INSERT INTO Course(Course_ID, Course_Name)
  2  VALUES (5, 'BHM');

1 row created.

SQL> INSERT INTO Course(Course_ID, Course_Name)
  2  VALUES (6, 'B.Arts');

1 row created.

SQL> INSERT INTO Course(Course_ID, Course_Name)
  2  VALUES (7, 'BDS');

1 row created.
```

```
SQL> UPDATE Course
  2  SET Highest_Mark = 90 WHERE Course_ID = 1;

1 row updated.

SQL> UPDATE Course
  2  SET Highest_Mark = 90 WHERE Course_ID = 6;

1 row updated.

SQL> UPDATE Course
  2  SET Highest_Mark = 95  WHERE Course_ID = 7;

1 row updated.
```

Figure 40: Inserting into Course

Inserting Values to Course_Details

INSERT ALL

INTO Course_Details(Course_ID, Spec_ID) VALUES (1, 3)

INTO Course_Details(Course_ID, Spec_ID) VALUES (1, 4)

INTO Course_Details(Course_ID, Spec_ID) VALUES (1, 5)

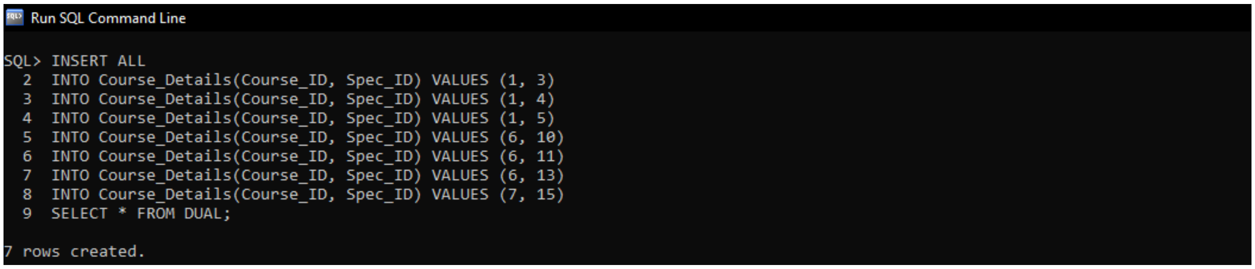
INTO Course_Details(Course_ID, Spec_ID) VALUES (6, 10)

INTO Course_Details(Course_ID, Spec_ID) VALUES (6, 11)

INTO Course_Details(Course_ID, Spec_ID) VALUES (6, 13)

INTO Course_Details(Course_ID, Spec_ID) VALUES (7, 15)

SELECT * FROM DUAL;



```
SQL> INSERT ALL
  2  INTO Course_Details(Course_ID, Spec_ID) VALUES (1, 3)
  3  INTO Course_Details(Course_ID, Spec_ID) VALUES (1, 4)
  4  INTO Course_Details(Course_ID, Spec_ID) VALUES (1, 5)
  5  INTO Course_Details(Course_ID, Spec_ID) VALUES (6, 10)
  6  INTO Course_Details(Course_ID, Spec_ID) VALUES (6, 11)
  7  INTO Course_Details(Course_ID, Spec_ID) VALUES (6, 13)
  8  INTO Course_Details(Course_ID, Spec_ID) VALUES (7, 15)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 41:Inserting into Course_Details

Inserting Values to Spec_Enrollment

INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)

VALUES (21, 3, 75000);

INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)

VALUES (22, 4, 80000);

INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)

VALUES (23, 5, 80000);

INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)

VALUES (24, 3, 75000);

INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)

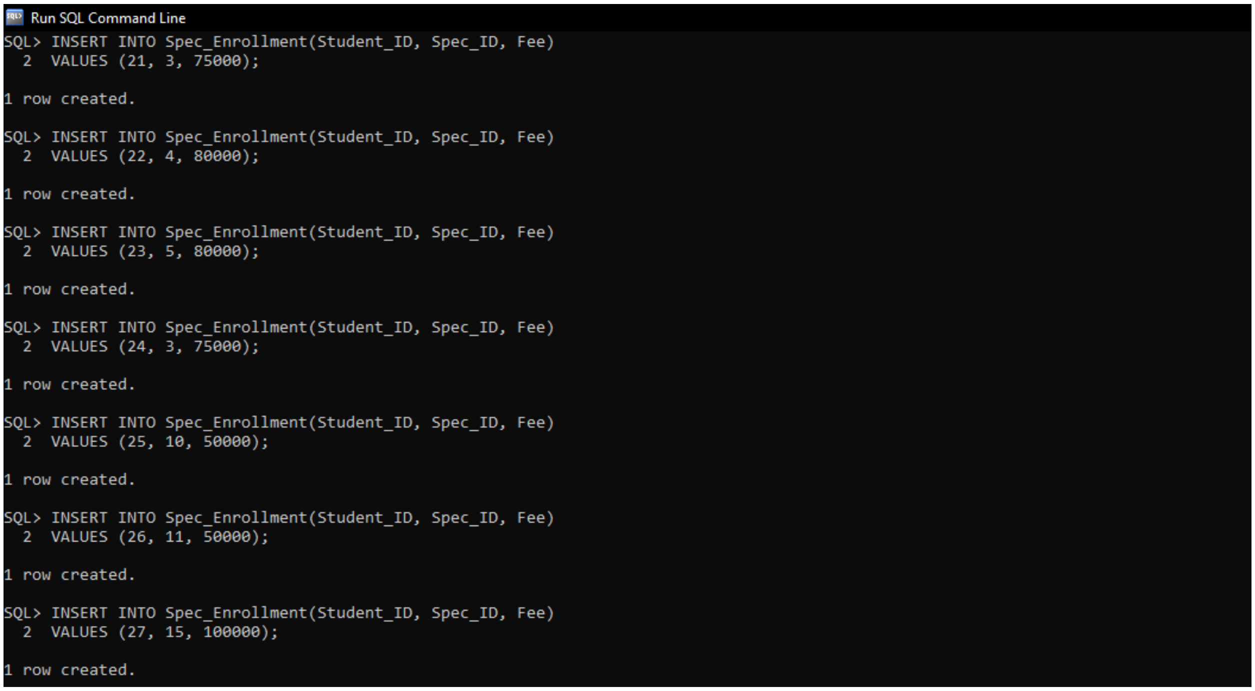
VALUES (25, 10, 50000);

INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)

VALUES (26, 11, 50000);

INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)

VALUES (27, 15, 100000);



```
SQL> INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)
  2  VALUES (21, 3, 75000);

1 row created.

SQL> INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)
  2  VALUES (22, 4, 80000);

1 row created.

SQL> INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)
  2  VALUES (23, 5, 80000);

1 row created.

SQL> INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)
  2  VALUES (24, 3, 75000);

1 row created.

SQL> INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)
  2  VALUES (25, 10, 50000);

1 row created.

SQL> INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)
  2  VALUES (26, 11, 50000);

1 row created.

SQL> INSERT INTO Spec_Enrollment(Student_ID, Spec_ID, Fee)
  2  VALUES (27, 15, 100000);

1 row created.
```

Figure 42: Inserting into Spec_Enrollment

Inserting Values to Admission

INSERT INTO Admission(Admit_ID, DOE)

VALUES (1, ’10-APR-2018’);

INSERT INTO Admission(Admit_ID, DOE)

VALUES (2, ’15-JUL-2018’);

INSERT INTO Admission(Admit_ID, DOE)

VALUES (3, ’23-NOV-2018’);

INSERT INTO Admission(Admit_ID, DOE)

VALUES (4, ’07-JAN-2019’);

INSERT INTO Admission(Admit_ID, DOE)

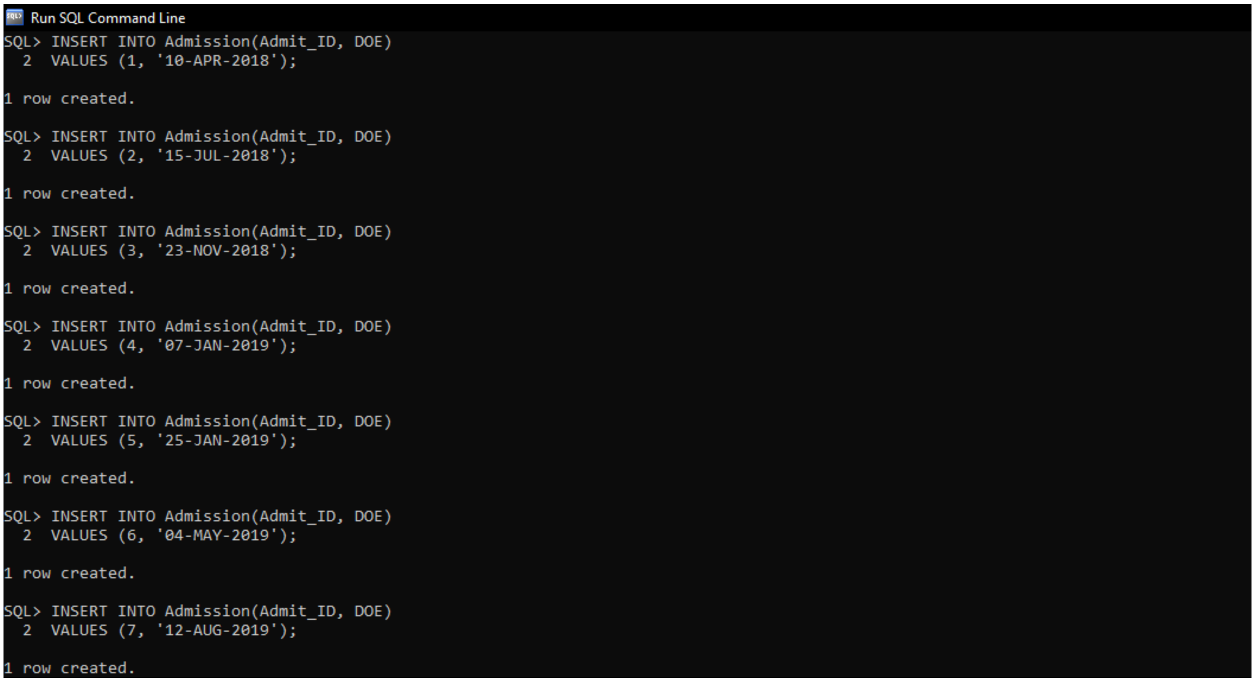
VALUES (5, ’25-JAN-2019’);

INSERT INTO Admission(Admit_ID, DOE)

VALUES (6, ’04-MAY-2019’);

INSERT INTO Admission(Admit_ID, DOE)

VALUES (7, ’12-AUG-2019’);



```
SQL> INSERT INTO Admission(Admit_ID, DOE)
  2  VALUES (1, '10-APR-2018');

1 row created.

SQL> INSERT INTO Admission(Admit_ID, DOE)
  2  VALUES (2, '15-JUL-2018');

1 row created.

SQL> INSERT INTO Admission(Admit_ID, DOE)
  2  VALUES (3, '23-NOV-2018');

1 row created.

SQL> INSERT INTO Admission(Admit_ID, DOE)
  2  VALUES (4, '07-JAN-2019');

1 row created.

SQL> INSERT INTO Admission(Admit_ID, DOE)
  2  VALUES (5, '25-JAN-2019');

1 row created.

SQL> INSERT INTO Admission(Admit_ID, DOE)
  2  VALUES (6, '04-MAY-2019');

1 row created.

SQL> INSERT INTO Admission(Admit_ID, DOE)
  2  VALUES (7, '12-AUG-2019');

1 row created.
```

Figure 43: Inserting into Admission

Inserting Values to Admission_Details

INSERT INTO Admission_Details(Admit_ID, Student_ID)

VALUES (1, 21);

INSERT INTO Admission_Details(Admit_ID, Student_ID)

VALUES (2, 23);

INSERT INTO Admission_Details(Admit_ID, Student_ID)

VALUES (3, 25);

INSERT INTO Admission_Details(Admit_ID, Student_ID)

VALUES (4, 22);

INSERT INTO Admission_Details(Admit_ID, Student_ID)

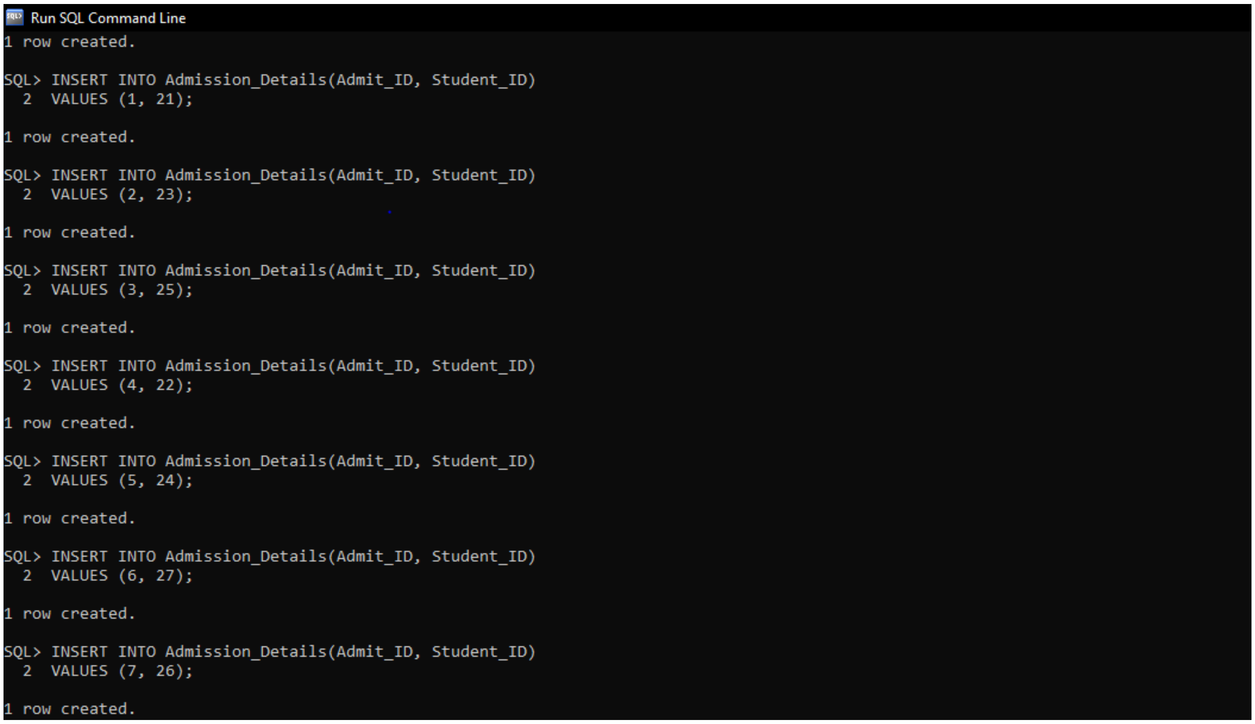
VALUES (5, 24);

INSERT INTO Admission_Details(Admit_ID, Student_ID)

VALUES (6, 27);

INSERT INTO Admission_Details(Admit_ID, Student_ID)

VALUES (7, 26);



```
1 row created.

SQL> INSERT INTO Admission_Details(Admit_ID, Student_ID)
  2  VALUES (1, 21);

1 row created.

SQL> INSERT INTO Admission_Details(Admit_ID, Student_ID)
  2  VALUES (2, 23);

1 row created.

SQL> INSERT INTO Admission_Details(Admit_ID, Student_ID)
  2  VALUES (3, 25);

1 row created.

SQL> INSERT INTO Admission_Details(Admit_ID, Student_ID)
  2  VALUES (4, 22);

1 row created.

SQL> INSERT INTO Admission_Details(Admit_ID, Student_ID)
  2  VALUES (5, 24);

1 row created.

SQL> INSERT INTO Admission_Details(Admit_ID, Student_ID)
  2  VALUES (6, 27);

1 row created.

SQL> INSERT INTO Admission_Details(Admit_ID, Student_ID)
  2  VALUES (7, 26);

1 row created.
```

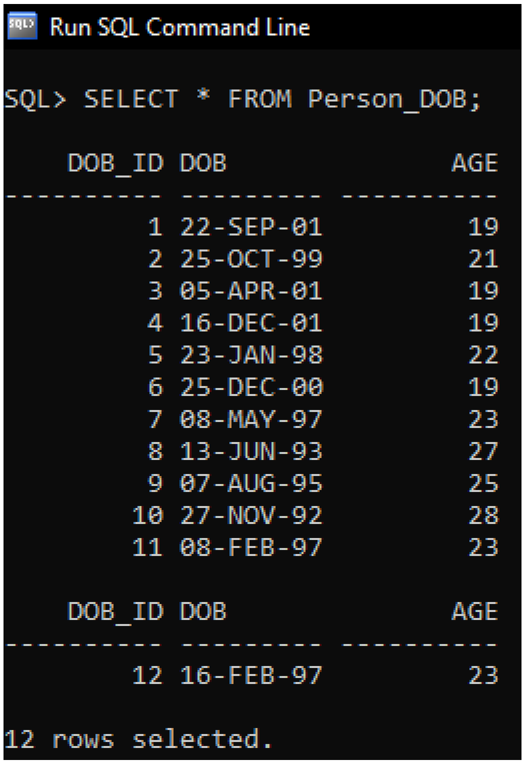
Figure 44: Inserting into Admission_Details

3.3. Final Tables

The SELECT statement is used to select data from a database.The data returned is stored in a result table, called the result-set. (W3schools, 2018)

Person_DOB

SELECT * FROM Person_DOB;

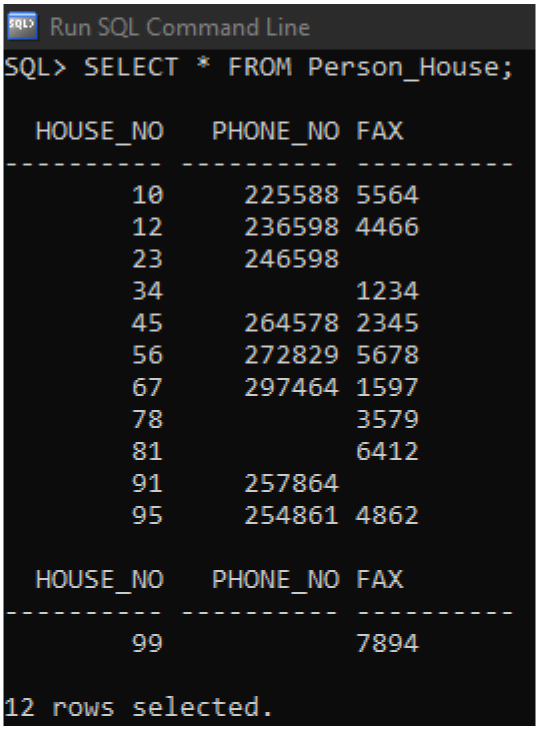


```
SQL> SELECT * FROM Person_DOB;

    DOB_ID DOB              AGE
---------- --------- ----------
         1 22-SEP-01         19
         2 25-OCT-99         21
         3 05-APR-01         19
         4 16-DEC-01         19
         5 23-JAN-98         22
         6 25-DEC-00         19
         7 08-MAY-97         23
         8 13-JUN-93         27
         9 07-AUG-95         25
        10 27-NOV-92         28
        11 08-FEB-97         23

    DOB_ID DOB              AGE
---------- --------- ----------
        12 16-FEB-97         23

12 rows selected.
```

Figure 45: Person_DOB

Person_House

SELECT * FROM Person_House;

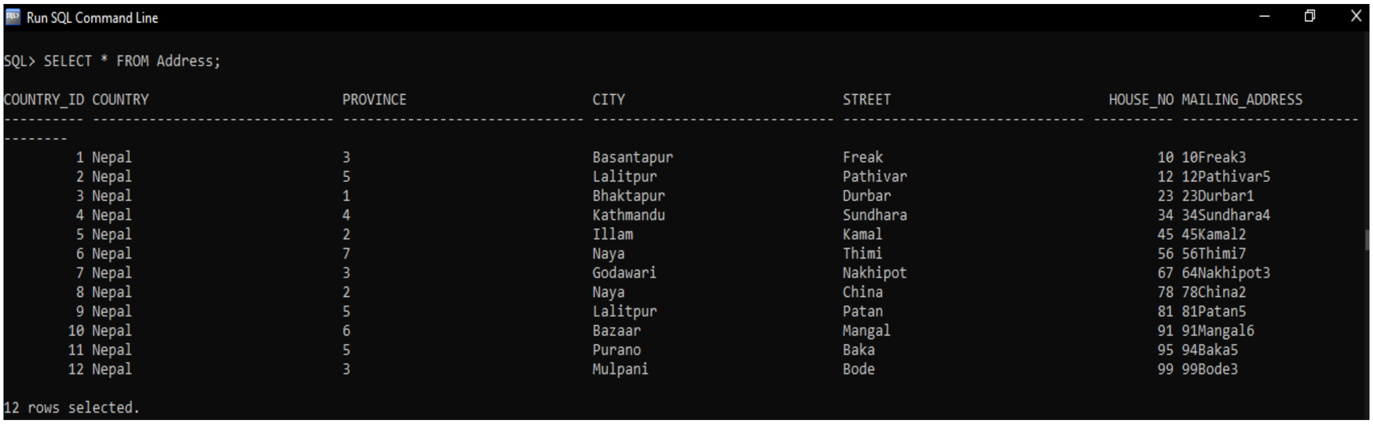


```
SQL> SELECT * FROM Person_House;

  HOUSE_NO   PHONE_NO FAX
---------- ---------- ----------
        10     225588 5564
        12     236598 4466
        23     246598
        34            1234
        45     264578 2345
        56     272829 5678
        67     297464 1597
        78            3579
        81            6412
        91     257864
        95     254861 4862

  HOUSE_NO   PHONE_NO FAX
---------- ---------- ----------
        99            7894

12 rows selected.
```

Figure 46: Person_House

Address

SELECT * FROM Address;

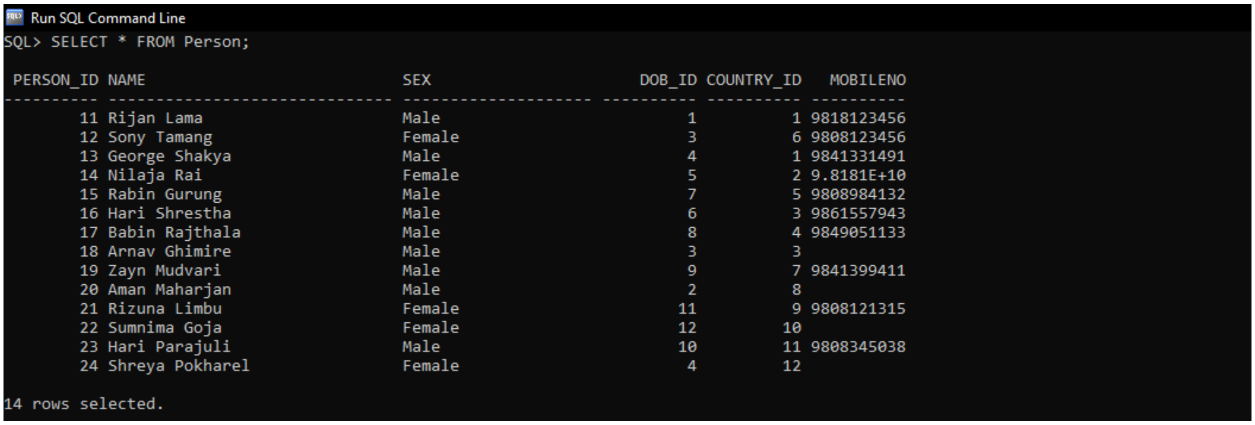


```
SQL> SELECT * FROM Address;

COUNTRY_ID COUNTRY                      PROVINCE                     CITY                         STREET                        HOUSE_NO MAILING_ADDRESS
--------- ---------------------------- ---------------------------- ---------------------------- ---------------------------- --------- --------------------
--------
        1 Nepal                        3                            Basantapur                   Freak                               10 10Freak3
        2 Nepal                        5                            Lalitpur                     Pathivar                            12 12Pathivar5
        3 Nepal                        1                            Bhaktapur                    Durbar                              23 23Durbar1
        4 Nepal                        4                            Kathmandu                    Sundhara                            34 34Sundhara4
        5 Nepal                        2                            Illam                        Kamal                               45 45Kamal2
        6 Nepal                        7                            Naya                         Thimi                               56 56Thimi7
        7 Nepal                        3                            Godawari                     Nakhipot                            67 64Nakhipot3
        8 Nepal                        2                            Naya                         China                               78 78China2
        9 Nepal                        5                            Lalitpur                     Patan                               81 81Patan5
       10 Nepal                        6                            Bazaar                       Mangal                              91 91Mangal6
       11 Nepal                        5                            Purano                       Baka                                95 94Baka5
       12 Nepal                        3                            Mulpani                      Bode                                99 99Bode3

12 rows selected.
```

Figure 47: Address

Person

SELECT * FROM Person;

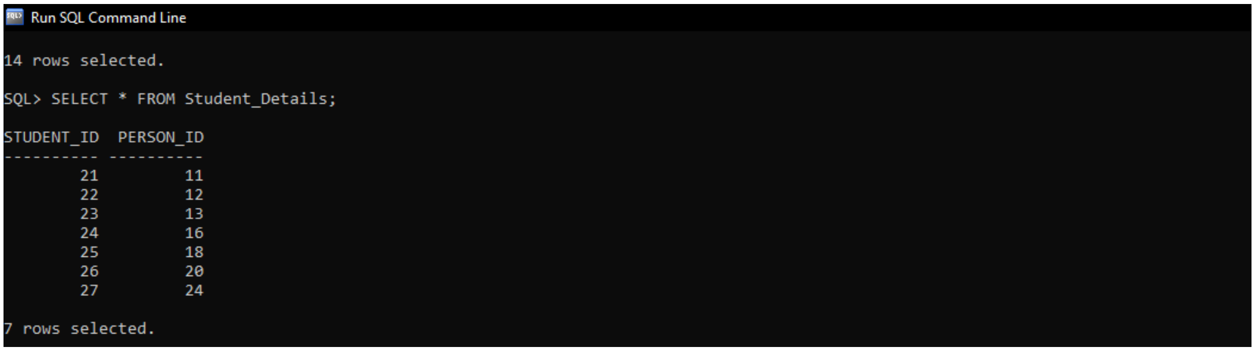


```
SQL> SELECT * FROM Person;

 PERSON_ID NAME                           SEX                      DOB_ID COUNTRY_ID   MOBILENO
---------- ------------------------------ -------------------- ---------- ---------- ----------
        11 Rijan Lama                     Male                          1          1 9818123456
        12 Sony Tamang                    Female                        3          6 9808123456
        13 George Shakya                  Male                          4          1 9841331491
        14 Nilaja Rai                     Female                        5          2 9.8181E+10
        15 Rabin Gurung                   Male                          7          5 9808984132
        16 Hari Shrestha                  Male                          6          3 9861557943
        17 Babin Rajthala                 Male                          8          4 9849051133
        18 Arnav Ghimire                  Male                          3          3
        19 Zayn Mudvari                   Male                          9          7 9841399411
        20 Aman Maharjan                  Male                          2          8
        21 Rizuna Limbu                   Female                       11          9 9808121315
        22 Sumnima Goja                   Female                       12         10
        23 Hari Parajuli                  Male                         10         11 9808345038
        24 Shreya Pokharel                Female                        4         12

14 rows selected.
```

Figure 48:Person

Student_Details

SELECT * FROM Student_Details;

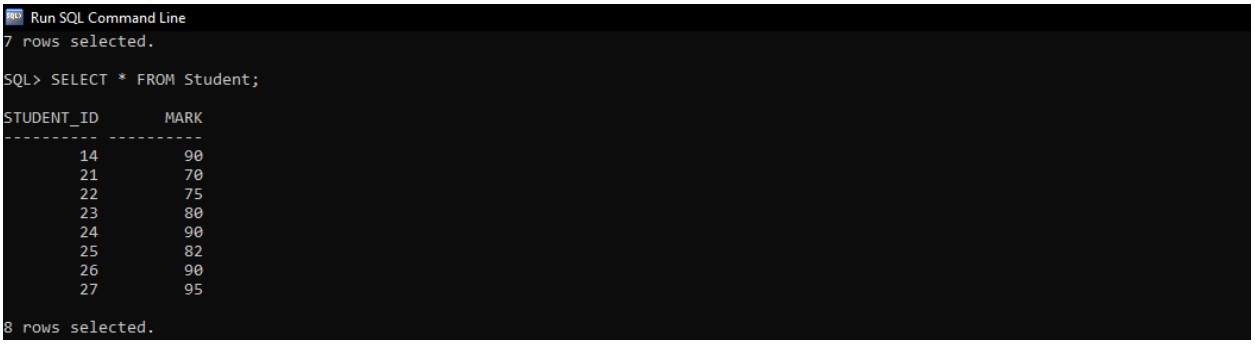


```
14 rows selected.

SQL> SELECT * FROM Student_Details;

STUDENT_ID  PERSON_ID
---------- ----------
        21         11
        22         12
        23         13
        24         16
        25         18
        26         20
        27         24

7 rows selected.
```

Figure 49: Student_Details

Student

SELECT * FROM Student;

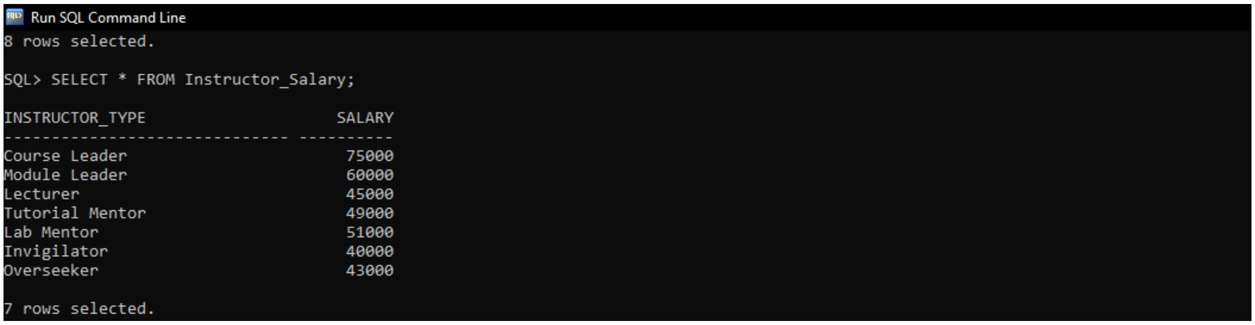


```
7 rows selected.

SQL> SELECT * FROM Student;

STUDENT_ID       MARK
---------- ----------
        14         90
        21         70
        22         75
        23         80
        24         90
        25         82
        26         90
        27         95

8 rows selected.
```

Figure 50: Student

Instructor_Salary

SELECT * FROM Instructor_Salary;

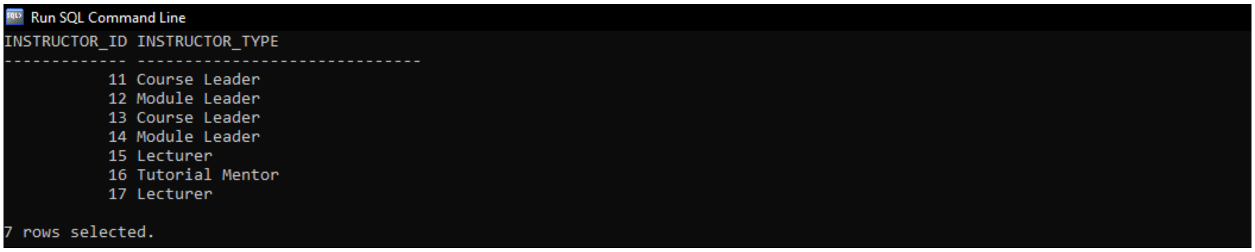


```
8 rows selected.

SQL> SELECT * FROM Instructor_Salary;

INSTRUCTOR_TYPE                    SALARY
------------------------------ ----------
Course Leader                       75000
Module Leader                       60000
Lecturer                            45000
Tutorial Mentor                     49000
Lab Mentor                          51000
Invigilator                         40000
Overseeker                          43000

7 rows selected.
```

Figure 51: Instructor Salarya

Instructor

SELECT * FROM Instructor;

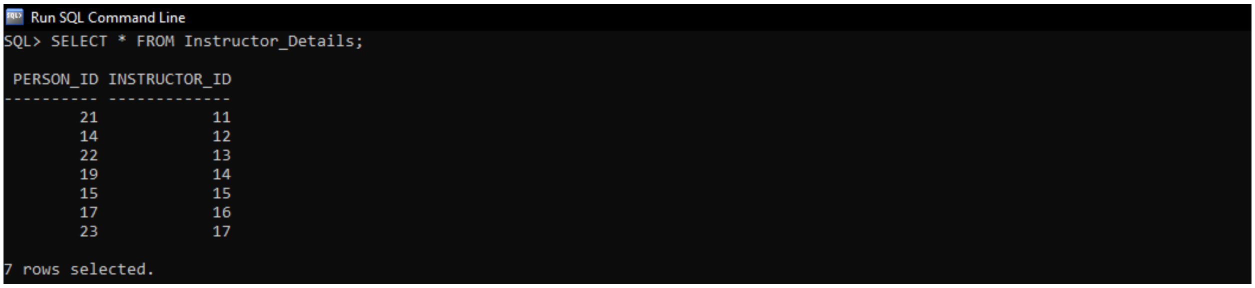


```
INSTRUCTOR_ID INSTRUCTOR_TYPE
------------- ------------------------------
           11 Course Leader
           12 Module Leader
           13 Course Leader
           14 Module Leader
           15 Lecturer
           16 Tutorial Mentor
           17 Lecturer

7 rows selected.
```

Figure 52: Instructor

Instructor_Details

SELECT * FROM Instructor_Details;

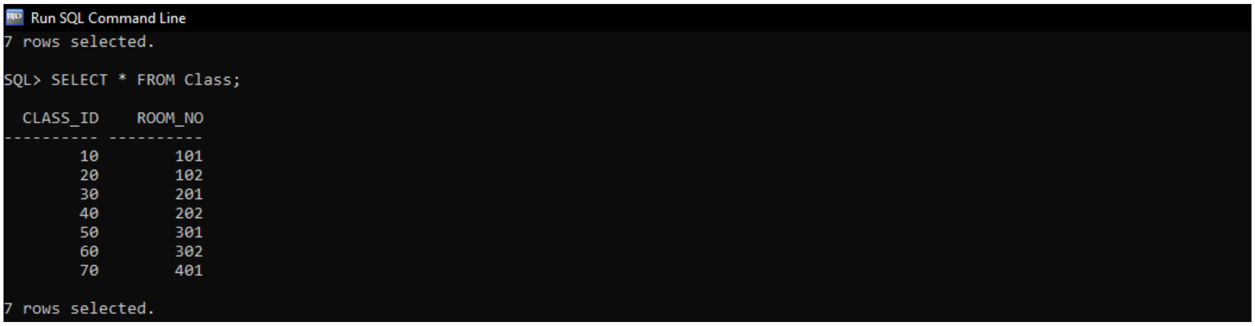


```
SQL> SELECT * FROM Instructor_Details;

 PERSON_ID INSTRUCTOR_ID
---------- -------------
        21            11
        14            12
        22            13
        19            14
        15            15
        17            16
        23            17

7 rows selected.
```

Figure 53: Instructor_Details

Class

SELECT * FROM Class;

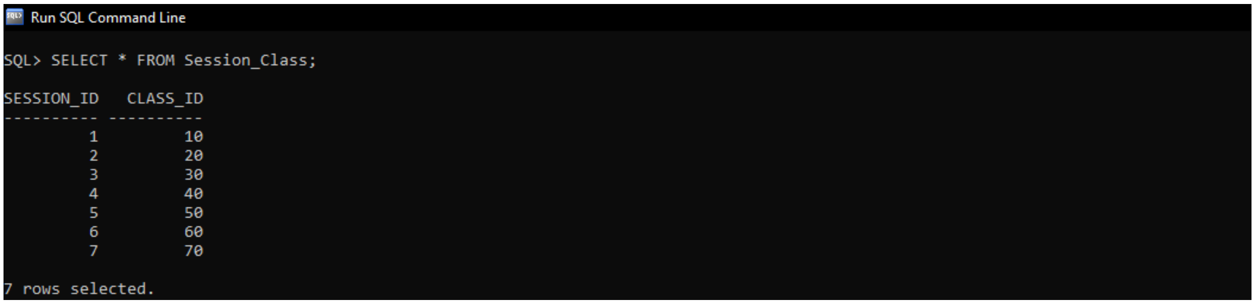


```
7 rows selected.

SQL> SELECT * FROM Class;

  CLASS_ID    ROOM_NO
---------- ----------
        10        101
        20        102
        30        201
        40        202
        50        301
        60        302
        70        401

7 rows selected.
```

Figure 54: Class

Session_Class

SELECT * FROM Session_Class;

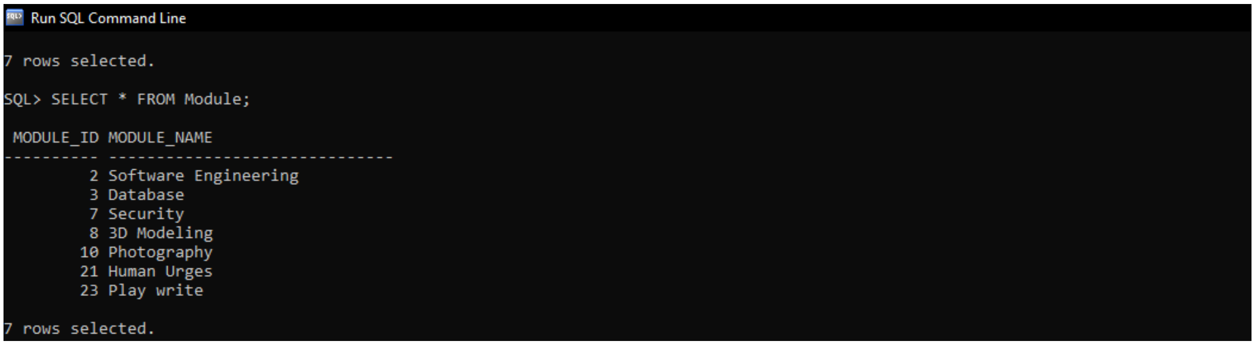


```
SQL> SELECT * FROM Session_Class;

SESSION_ID   CLASS_ID
---------- ----------
         1         10
         2         20
         3         30
         4         40
         5         50
         6         60
         7         70

7 rows selected.
```

Figure 55:: Session_Class

Module

SELECT * FROM Module;

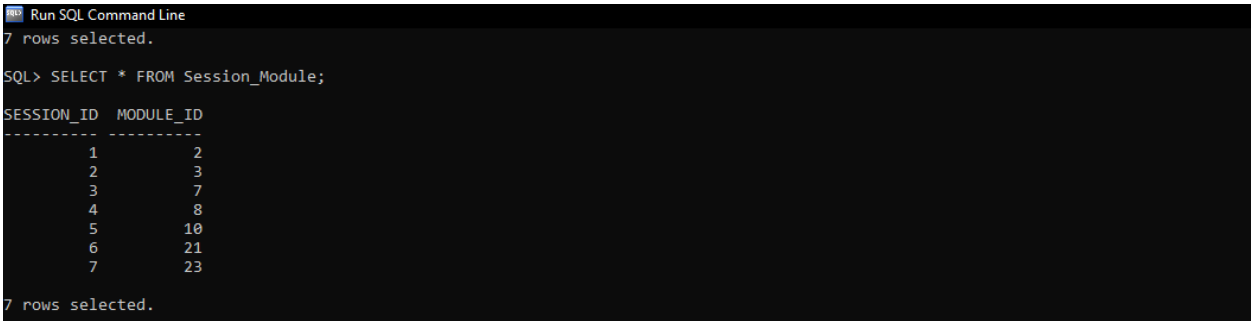


```
7 rows selected.

SQL> SELECT * FROM Module;

 MODULE_ID MODULE_NAME
---------- ------------------------------
         2 Software Engineering
         3 Database
         7 Security
         8 3D Modeling
        10 Photography
        21 Human Urges
        23 Play write

7 rows selected.
```

Figure 56: Module

Session_Module

SELECT * FROM Session_Module;

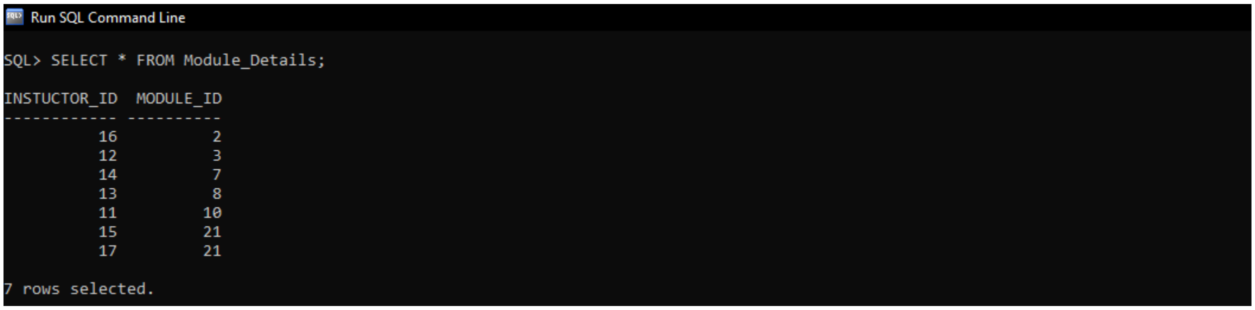


```
7 rows selected.

SQL> SELECT * FROM Session_Module;

SESSION_ID  MODULE_ID
---------- ----------
         1          2
         2          3
         3          7
         4          8
         5         10
         6         21
         7         23

7 rows selected.
```

Figure 57: Session_Module

Module_Details

SELECT * FROM Module_Details;



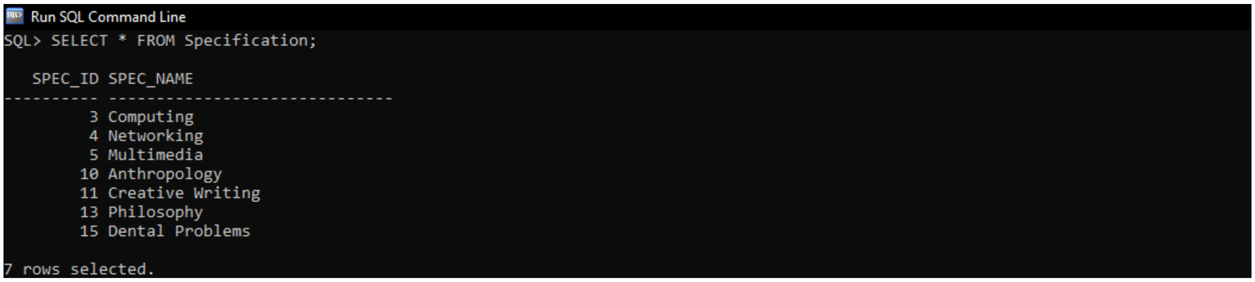
Run SQL Command Line

```
SQL> SELECT * FROM Module_Details;

INSTUCTOR_ID  MODULE_ID
------------ ----------
          16          2
          12          3
          14          7
          13          8
          11         10
          15         21
          17         21

7 rows selected.
```

Figure 58: Module_Details

Specification

SELECT * FROM Specification;



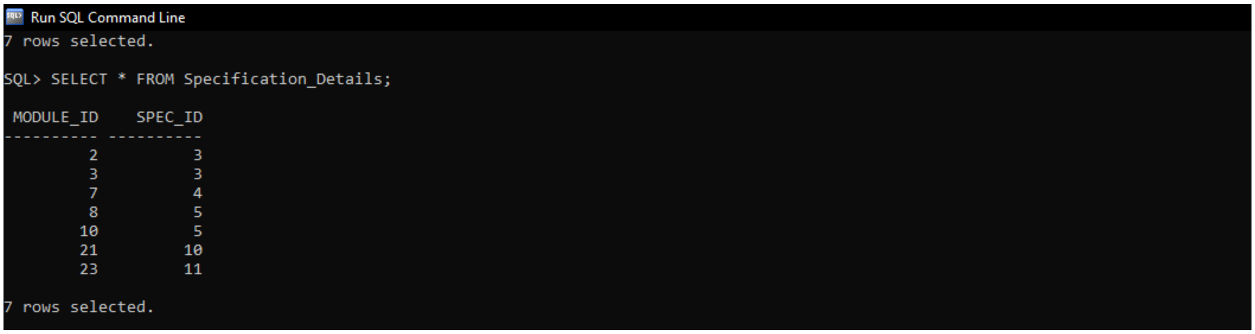
Run SQL Command Line

```
SQL> SELECT * FROM Specification;

   SPEC_ID SPEC_NAME
---------- ------------------------------
         3 Computing
         4 Networking
         5 Multimedia
        10 Anthropology
        11 Creative Writing
        13 Philosophy
        15 Dental Problems

7 rows selected.
```

Figure 59:Specification

Specification_Details

SELECT * FROM Specification_Details;



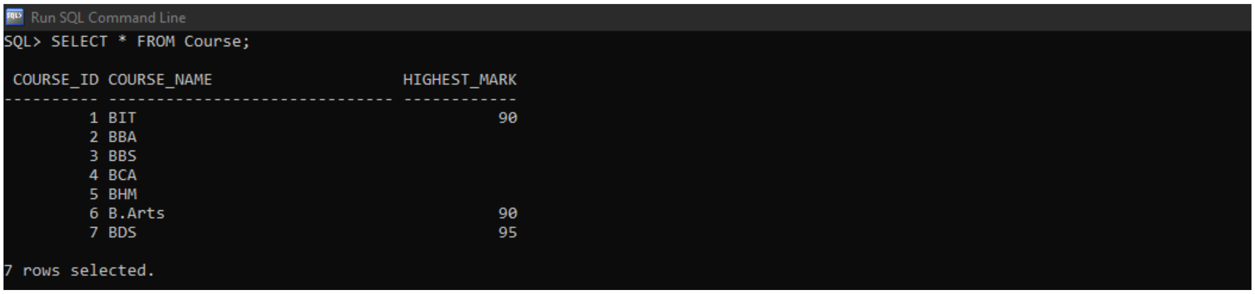
Run SQL Command Line

```
7 rows selected.

SQL> SELECT * FROM Specification_Details;

 MODULE_ID    SPEC_ID
---------- ----------
         2          3
         3          3
         7          4
         8          5
        10          5
        21         10
        23         11

7 rows selected.
```

Figure 60: Specification Details

Course

SELECT * FROM Course;

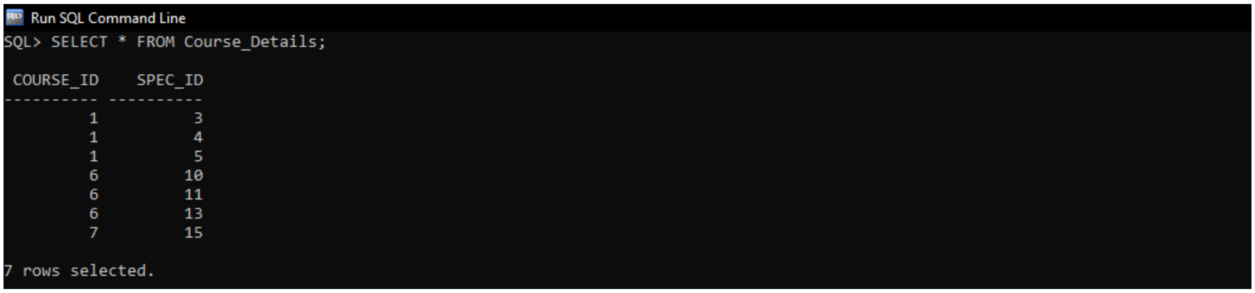


```
SQL> SELECT * FROM Course;

 COURSE_ID COURSE_NAME                    HIGHEST_MARK
---------- ------------------------------ ------------
         1 BIT                                      90
         2 BBA
         3 BBS
         4 BCA
         5 BHM
         6 B.Arts                                   90
         7 BDS                                      95

7 rows selected.
```

Figure 61: Course

Course_Details

SELECT * FROM Course_Details;

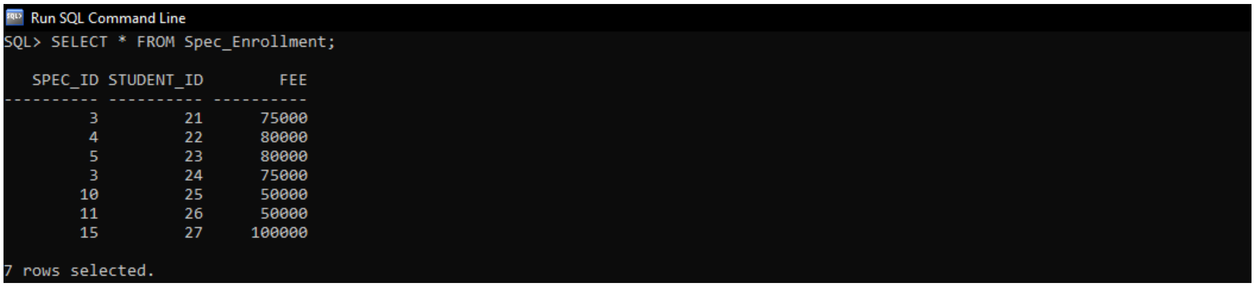


```
SQL> SELECT * FROM Course_Details;

 COURSE_ID    SPEC_ID
---------- ----------
         1          3
         1          4
         1          5
         6         10
         6         11
         6         13
         7         15

7 rows selected.
```

Figure 62: Course_Details

Spec_Enrollment

SELECT * FROM Spec_Enrollment;

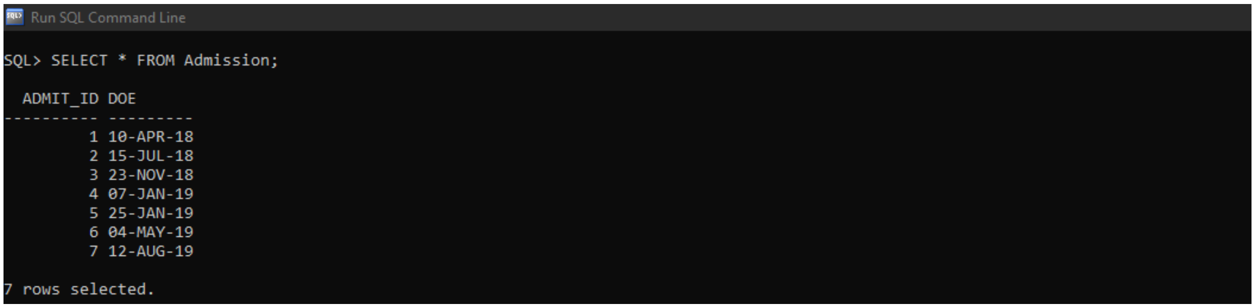


```
SQL> SELECT * FROM Spec_Enrollment;

   SPEC_ID STUDENT_ID        FEE
---------- ---------- ----------
         3         21      75000
         4         22      80000
         5         23      80000
         3         24      75000
        10         25      50000
        11         26      50000
        15         27     100000

7 rows selected.
```

Figure 63: Spec_Enrollment

Admission

SELECT * FROM Admission;

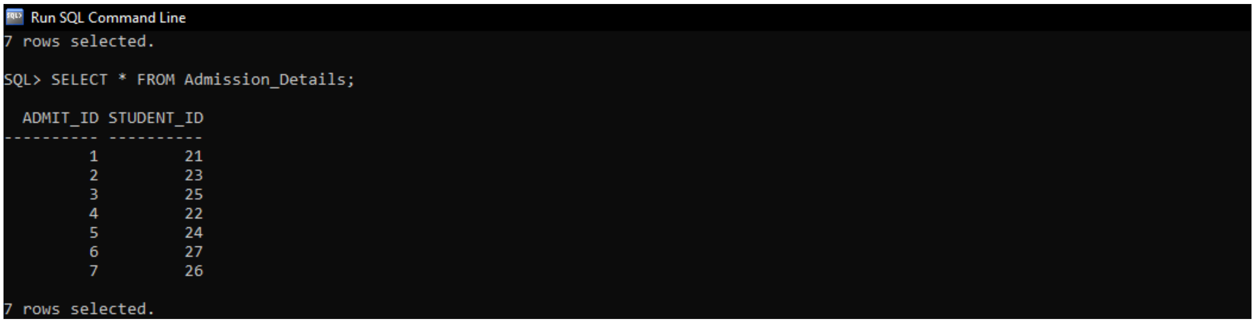


```
SQL> SELECT * FROM Admission;

  ADMIT_ID DOE
---------- ---------
         1 10-APR-18
         2 15-JUL-18
         3 23-NOV-18
         4 07-JAN-19
         5 25-JAN-19
         6 04-MAY-19
         7 12-AUG-19

7 rows selected.
```

Figure 64: Admission

Admission_Details

SELECT * FROM Admission_Details;



```
7 rows selected.

SQL> SELECT * FROM Admission_Details;

  ADMIT_ID STUDENT_ID
---------- ----------
         1         21
         2         23
         3         25
         4         22
         5         24
         6         27
         7         26

7 rows selected.
```

Figure 65: Admission_Details

4. Database Querying

4.1. Information Queries

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

SELECT Student_Details.Student_ID, Person.Name, Address.Country, Address.Province, Address.City, Address.Street, Person_House.House_No, Person_House.Phone_No FROM Student_Details FULL OUTER JOIN

Person

ON Student_Details.Person_ID = Person.Person_ID FULL OUTER JOIN

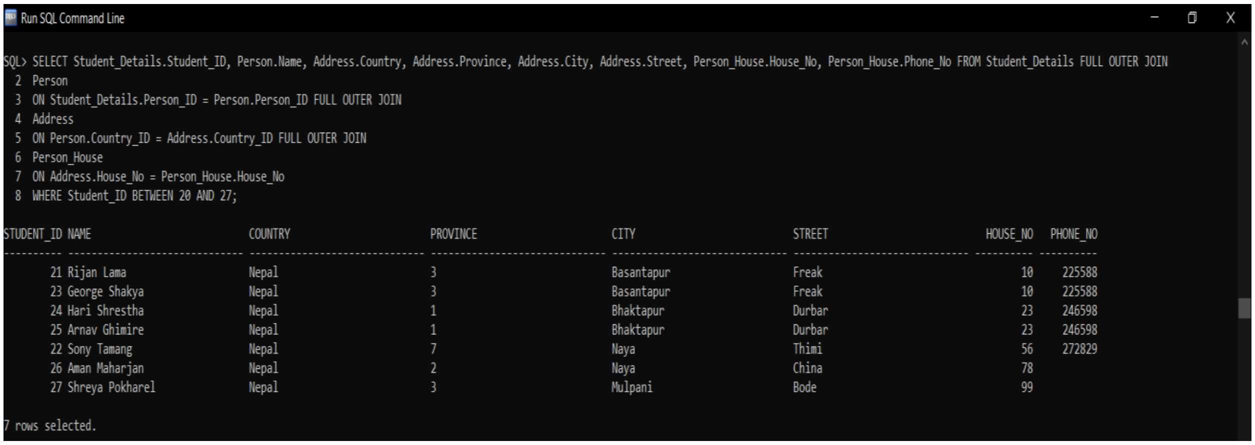
Address

ON Person.Country_ID = Address.Country_ID FULL OUTER JOIN

Person_House

ON Address.House_No = Person_House.House_No

WHERE Student_ID BETWEEN 20 AND 27;



```
SQL> SELECT Student_Details.Student_ID, Person.Name, Address.Country, Address.Province, Address.City, Address.Street, Person_House.House_No, Person_House.Phone_No FROM Student_Details FULL OUTER JOIN
Person
ON Student_Details.Person_ID = Person.Person_ID FULL OUTER JOIN
Address
ON Person.Country_ID = Address.Country_ID FULL OUTER JOIN
Person_House
ON Address.House_No = Person_House.House_No
WHERE Student_ID BETWEEN 20 AND 27;

STUDENT_ID NAME                           COUNTRY                        PROVINCE                       CITY                           STREET                           HOUSE_NO   PHONE_NO
---------- ------------------------------ ------------------------------ ------------------------------ ------------------------------ ------------------------------ ---------- ----------
        21 Rijan Lama                     Nepal                          3                              Basantapur                     Freak                                  10     225588
        23 George Shakya                  Nepal                          3                              Basantapur                     Freak                                  10     225588
        24 Hari Shrestha                  Nepal                          1                              Bhaktapur                      Durbar                                 23     246598
        25 Arnav Ghimire                  Nepal                          1                              Bhaktapur                      Durbar                                 23     246598
        22 Sony Tamang                    Nepal                          7                              Naya                           Thimi                                  56     272829
        26 Aman Maharjan                  Nepal                          2                              Naya                           China                                  78
        27 Shreya Pokharel                Nepal                          3                              Mulpani                        Bode                                   99

7 rows selected.
```

Figure 66: Information Query No 1

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

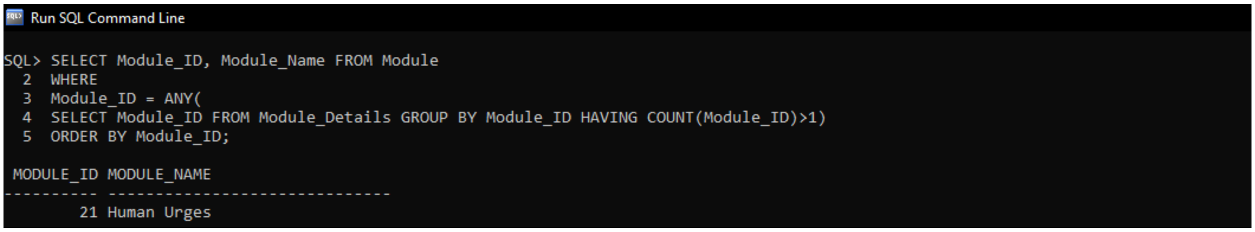
SELECT Module_ID, Module_Name FROM Module

WHERE

Module_ID = ANY(

SELECT Module_ID FROM Module_Details GROUP BY Module_ID HAVING COUNT(Module_ID)>1)

ORDER BY Module_ID;



```
SQL> SELECT Module_ID, Module_Name FROM Module
  2  WHERE
  3  Module_ID = ANY(
  4  SELECT Module_ID FROM Module_Details GROUP BY Module_ID HAVING COUNT(Module_ID)>1)
  5  ORDER BY Module_ID;

 MODULE_ID MODULE_NAME
---------- ------------------------------
        21 Human Urges
```

Figure 67: Information Query No 2

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

SELECT Instructor.Instructor_ID, Person.Name, Instructor.Instructor_Type, Instructor_Salary.Salary FROM Person

JOIN

Instructor_Details

ON Person.Person_ID = Instructor_Details.Person_ID

JOIN

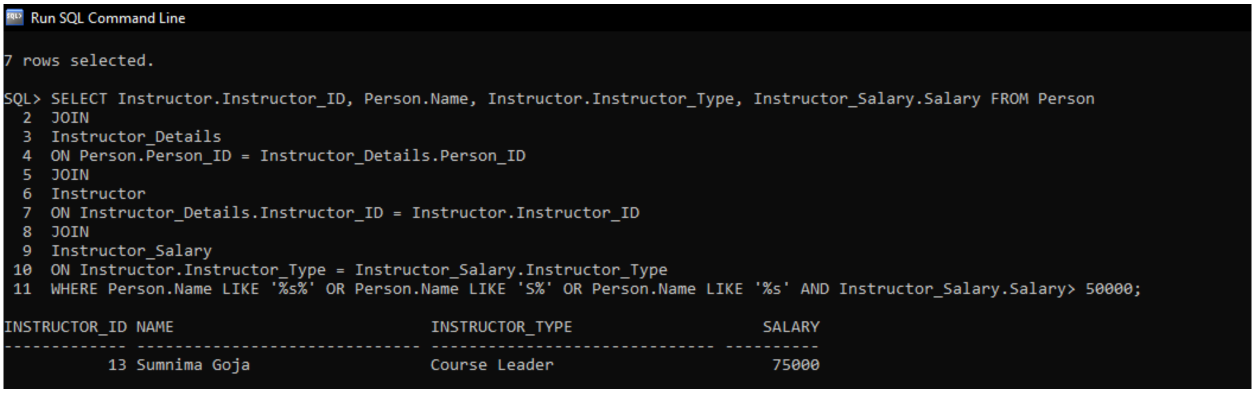
Instructor

ON Instructor_Details.Instructor_ID = Instructor.Instructor_ID

JOIN

Instructor_Salary

ON Instructor.Instructor_Type = Instructor_Salary.Instructor_Type



```
7 rows selected.

SQL> SELECT Instructor.Instructor_ID, Person.Name, Instructor.Instructor_Type, Instructor_Salary.Salary FROM Person
     JOIN
     Instructor_Details
     ON Person.Person_ID = Instructor_Details.Person_ID
     JOIN
     Instructor
     ON Instructor_Details.Instructor_ID = Instructor.Instructor_ID
     JOIN
     Instructor_Salary
     ON Instructor.Instructor_Type = Instructor_Salary.Instructor_Type
     WHERE Person.Name LIKE '%s%' OR Person.Name LIKE 'S%' OR Person.Name LIKE '%s' AND Instructor_Salary.Salary> 50000;

INSTRUCTOR_ID NAME                           INSTRUCTOR_TYPE                    SALARY
------------- ------------------------------ ------------------------------ ----------
           13 Sumnima Goja                   Course Leader                       75000
```

Figure 68: Information Query No 3

4.1.4. List the modules comes under the ‘Multimedia’ specification.

SELECT Module.Module_ID, Module.Module_Name, Specification.Spec_Name FROM Module

JOIN

Specification_Details

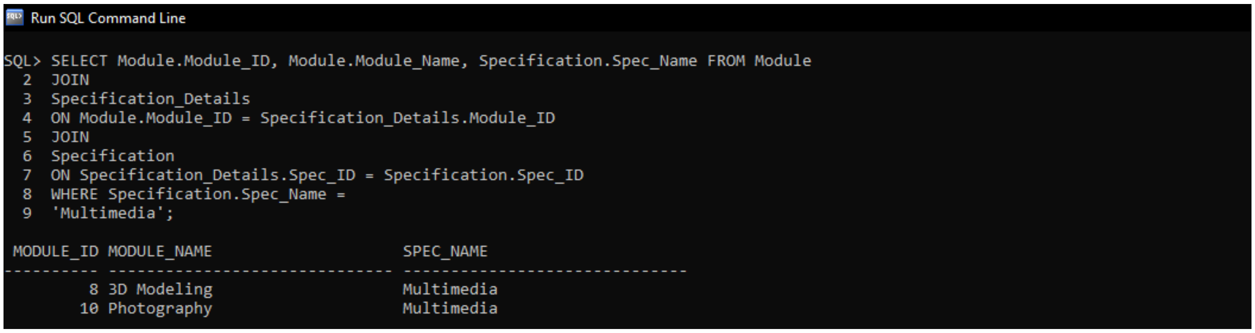
ON Module.Module_ID = Specification_Details.Module_ID

JOIN

Specification

ON Specification_Details.Spec_ID = Specification.Spec_ID

WHERE Specification.Spec_Name = ‘Multimedia’;



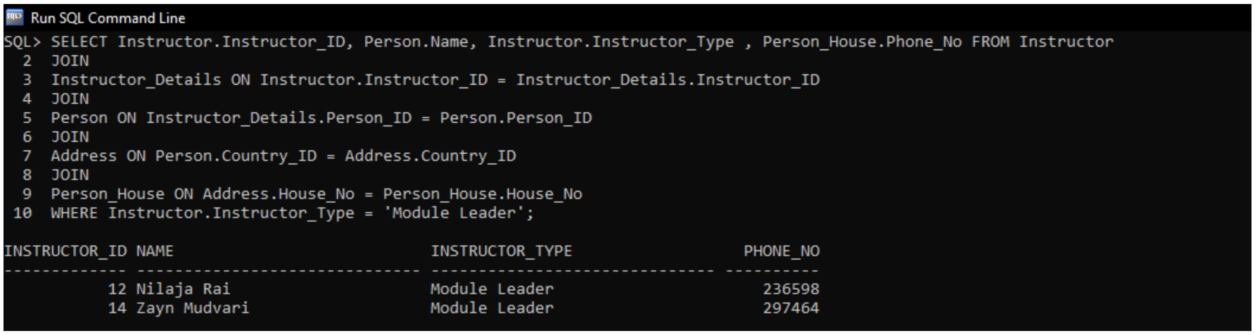
Run SQL Command Line

```
SQL> SELECT Module.Module_ID, Module.Module_Name, Specification.Spec_Name FROM Module
  2  JOIN
  3  Specification_Details
  4  ON Module.Module_ID = Specification_Details.Module_ID
  5  JOIN
  6  Specification
  7  ON Specification_Details.Spec_ID = Specification.Spec_ID
  8  WHERE Specification.Spec_Name =
  9  'Multimedia';

 MODULE_ID MODULE_NAME                    SPEC_NAME
---------- ------------------------------ ------------------------------
         8 3D Modeling                    Multimedia
        10 Photography                    Multimedia
```

Figure 69: Information Query No 4

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

```
SELECT Instructor.Instructor_ID, Person.Name, Instructor.Instructor_Type , Person_House.Phone_No FROM Instructor

JOIN

Instructor_Details ON Instructor.Instructor_ID = Instructor_Details.Instructor_ID

JOIN

Person ON Instructor_Details.Person_ID = Person.Person_ID

JOIN

Address ON Person.Country_ID = Address.Country_ID

JOIN

Person_House ON Address.House_No = Person_House.House_No

WHERE Instructor.Instructor_Type = ‘ Module Leader’;
```



```
SQL> SELECT Instructor.Instructor_ID, Person.Name, Instructor.Instructor_Type , Person_House.Phone_No FROM Instructor
     JOIN
     Instructor_Details ON Instructor.Instructor_ID = Instructor_Details.Instructor_ID
     JOIN
     Person ON Instructor_Details.Person_ID = Person.Person_ID
     JOIN
     Address ON Person.Country_ID = Address.Country_ID
     JOIN
     Person_House ON Address.House_No = Person_House.House_No
     WHERE Instructor.Instructor_Type = 'Module Leader';

INSTRUCTOR_ID NAME                           INSTRUCTOR_TYPE                  PHONE_NO
------------- ------------------------------ ------------------------------ ----------
           12 Nilaja Rai                     Module Leader                      236598
           14 Zayn Mudvari                   Module Leader                      297464
```

Figure 70: Information Query No 5

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

SELECT Student.Student_ID, Person.Name, Specification.Spec_Name FROM Person

JOIN

Student_Details ON Person.Person_ID = Student_Details.Person_ID

JOIN

Student ON Student_Details.Student_ID = Student.Student_ID

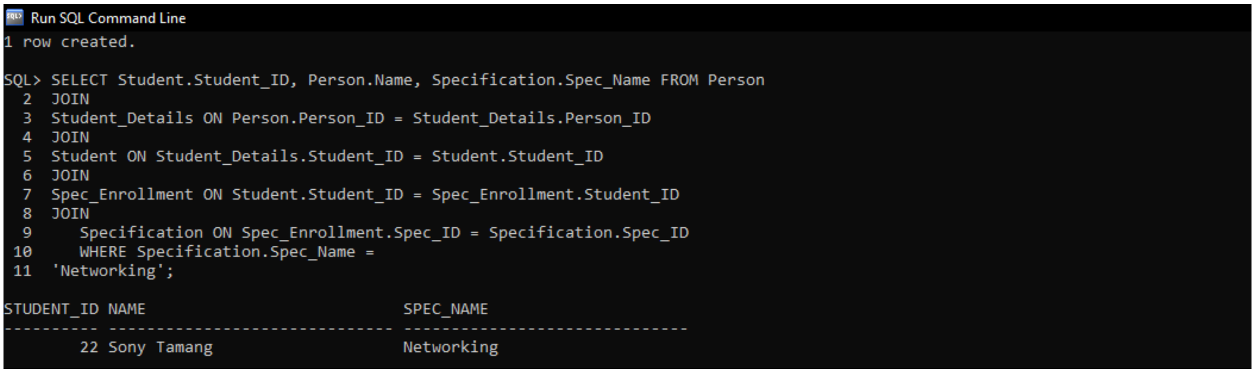
JOIN

Spec_Enrollment ON Student.Student_ID = Spec_Enrollment.Student_ID

JOIN

Specification ON Spec_Enrollment.Spec_ID = Specification.Spec_ID

WHERE Specification.Spec_Name = ‘Networking’;



```
1 row created.

SQL> SELECT Student.Student_ID, Person.Name, Specification.Spec_Name FROM Person
     JOIN
     Student_Details ON Person.Person_ID = Student_Details.Person_ID
     JOIN
     Student ON Student_Details.Student_ID = Student.Student_ID
     JOIN
     Spec_Enrollment ON Student.Student_ID = Spec_Enrollment.Student_ID
     JOIN
        Specification ON Spec_Enrollment.Spec_ID = Specification.Spec_ID
        WHERE Specification.Spec_Name =
     'Networking';

STUDENT_ID NAME                           SPEC_NAME
---------- ------------------------------ ------------------------------
        22 Sony Tamang                    Networking
```

Figure 71: Information Query No 6

4.1.7. List the fax number of the instructor who teaches the ‘database’ module.

SELECT Person.Name , Module.Module_Name, Person_House.Fax FROM Module

JOIN

Module_Details ON Module.Module_ID= Module_Details.Module_ID

JOIN

Instructor ON Module_Details.Instuctor_ID = Instructor.Instructor_ID

JOIN

Instructor_Details ON Instructor.Instructor_ID = Instructor_Details.Instructor_ID

JOIN

Person ON Instructor_Details.Person_ID = Person.Person_ID

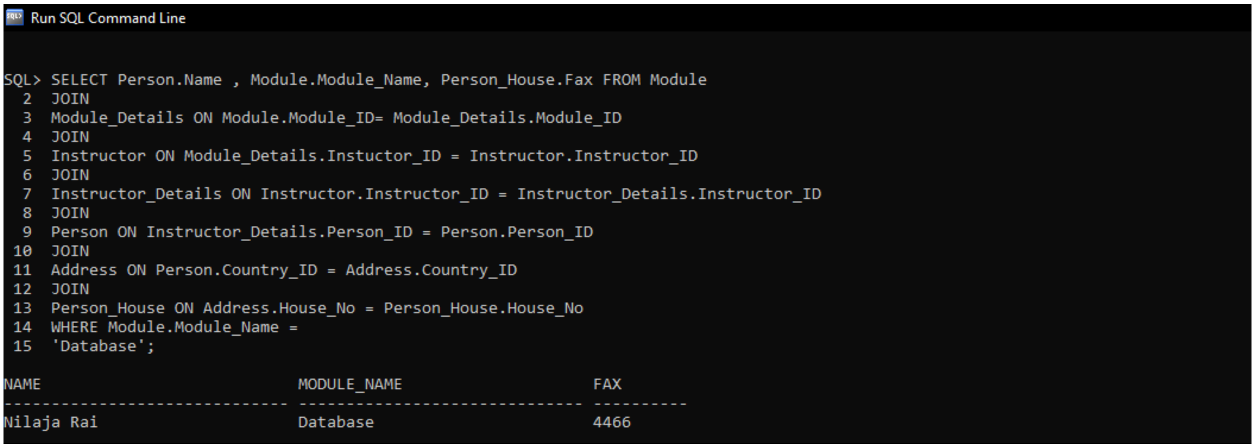
JOIN

Address ON Person.Country_ID = Address.Country_ID

JOIN

Person_House ON Address.House_No = Person_House.House_No

WHERE Module.Module_Name = ‘Database’;



```
SQL> SELECT Person.Name , Module.Module_Name, Person_House.Fax FROM Module
     JOIN
     Module_Details ON Module.Module_ID= Module_Details.Module_ID
     JOIN
     Instructor ON Module_Details.Instuctor_ID = Instructor.Instructor_ID
     JOIN
     Instructor_Details ON Instructor.Instructor_ID = Instructor_Details.Instructor_ID
     JOIN
     Person ON Instructor_Details.Person_ID = Person.Person_ID
     JOIN
     Address ON Person.Country_ID = Address.Country_ID
     JOIN
     Person_House ON Address.House_No = Person_House.House_No
     WHERE Module.Module_Name =
     'Database';

NAME                           MODULE_NAME                    FAX
------------------------------ ------------------------------ ----------
Nilaja Rai                     Database                       4466
```

Figure 72: Information Query No.7

4.1.8. List the specification falls under the BIT course.

SELECT Specification.Spec_Name, Course.Course_Name FROM Specification

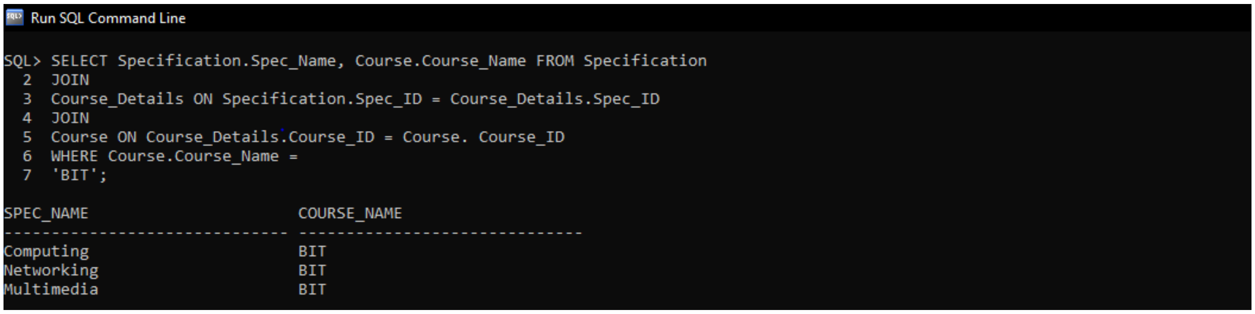
JOIN

Course_Details ON Specification.Spec_ID = Course_Details.Spec_ID

JOIN

Course ON Course_Details.Course_ID = Course. Course_ID

WHERE Course.Course_Name = ‘BIT’;



```
SQL> SELECT Specification.Spec_Name, Course.Course_Name FROM Specification
     JOIN
     Course_Details ON Specification.Spec_ID = Course_Details.Spec_ID
     JOIN
     Course ON Course_Details.Course_ID = Course. Course_ID
     WHERE Course.Course_Name =
     'BIT';

SPEC_NAME                      COURSE_NAME
------------------------------ ------------------------------
Computing                      BIT
Networking                     BIT
Multimedia                     BIT
```

Figure 73: Information Query No. 8

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

SELECT Class.Room_No , Module.Module_Name FROM Class

JOIN

Session_Class ON Class.Class_ID = Session_Class.Class_ID

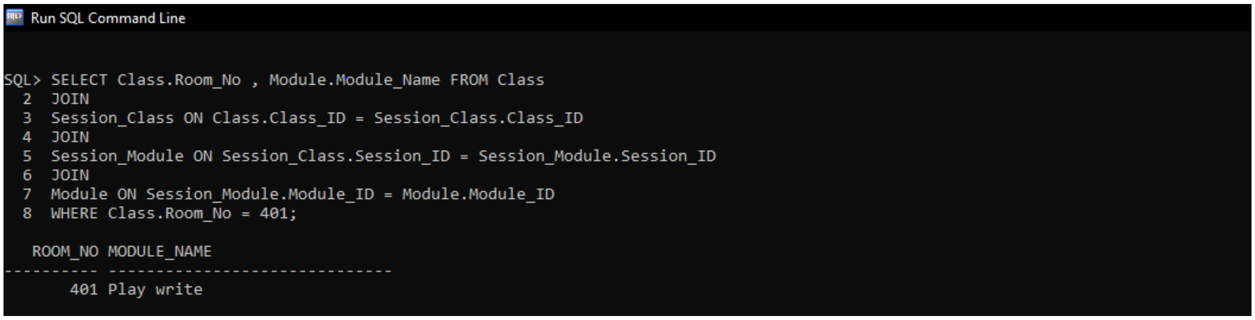
JOIN

Session_Module ON Session_Class.Session_ID = Session_Module.Session_ID

JOIN

Module ON Session_Module.Module_ID = Module.Module_ID

WHERE Class.Room_No = 401;



```
SQL> SELECT Class.Room_No , Module.Module_Name FROM Class
     JOIN
     Session_Class ON Class.Class_ID = Session_Class.Class_ID
     JOIN
     Session_Module ON Session_Class.Session_ID = Session_Module.Session_ID
     JOIN
     Module ON Session_Module.Module_ID = Module.Module_ID
     WHERE Class.Room_No = 401;

   ROOM_NO MODULE_NAME
---------- ------------------------------
       401 Play write
```

Figure 74: Infromation Query No. 9

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

SELECT Person.Name, Address.Country, Address.Province, Address.City, Address.Street, Address.Mailing_Address FROM Address

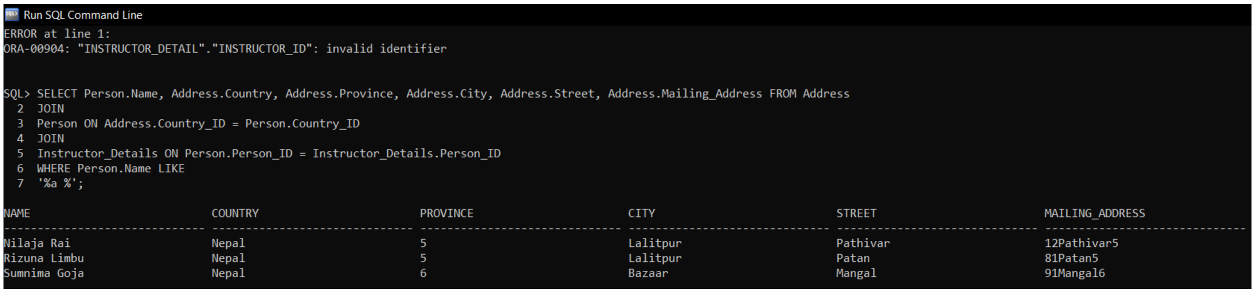
JOIN

Person ON Address.Country_ID = Person.Country_ID

JOIN

Instructor_Details ON Person.Person_ID = Instructor_Details.Person_ID

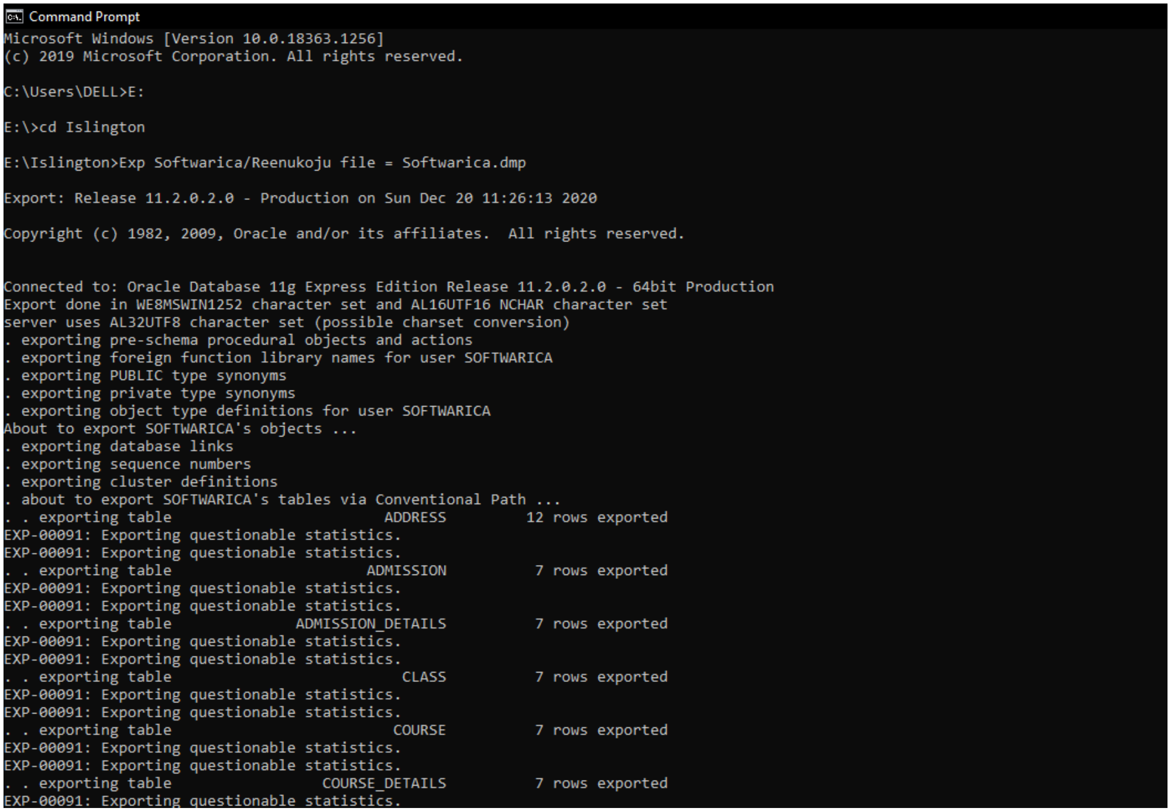
WHERE Person.Name LIKE ‘%a %’;



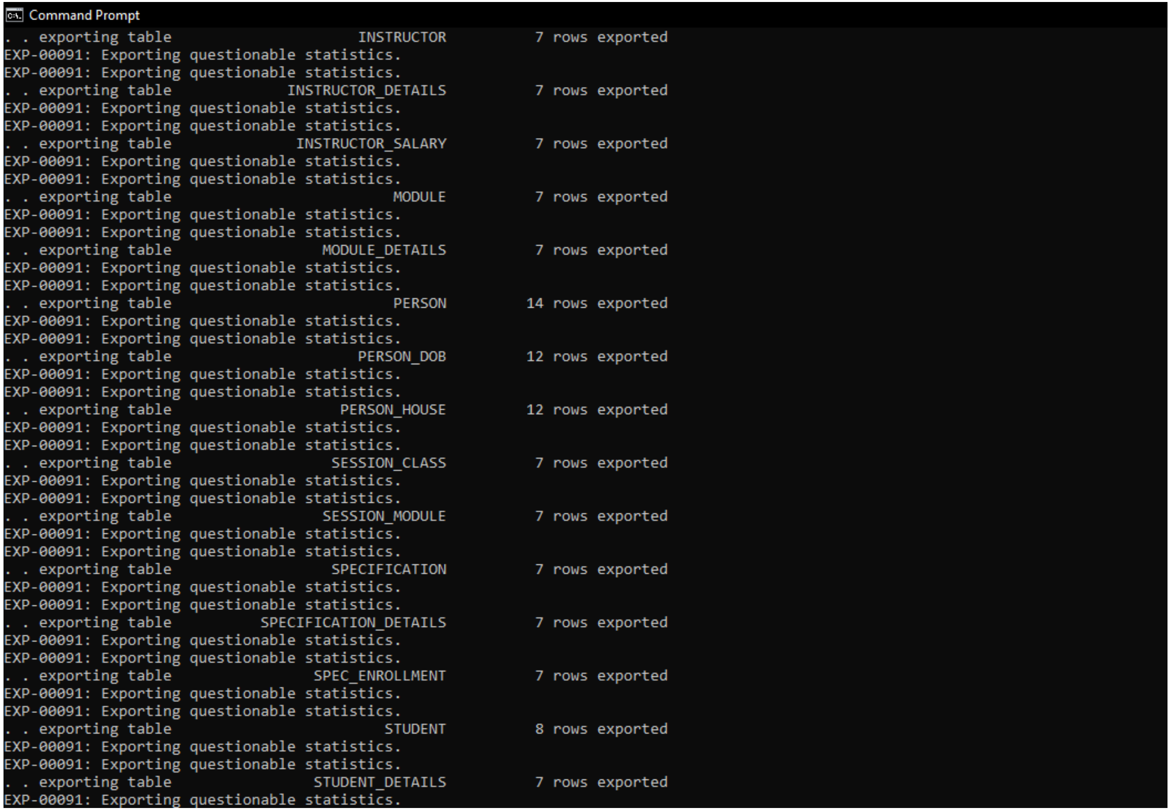
```
ERROR at line 1:
ORA-00904: "INSTRUCTOR_DETAIL"."INSTRUCTOR_ID": invalid identifier


SQL> SELECT Person.Name, Address.Country, Address.Province, Address.City, Address.Street, Address.Mailing_Address FROM Address
  2  JOIN
  3  Person ON Address.Country_ID = Person.Country_ID
  4  JOIN
  5  Instructor_Details ON Person.Person_ID = Instructor_Details.Person_ID
  6  WHERE Person.Name LIKE
  7  '%a %';

NAME                           COUNTRY                        PROVINCE                       CITY                           STREET                         MAILING_ADDRESS
------------------------------ ------------------------------ ------------------------------ ------------------------------ ------------------------------ ------------------------------
Nilaja Rai                     Nepal                          5                              Lalitpur                       Pathivar                       12Pathivar5
Rizuna Limbu                   Nepal                          5                              Lalitpur                       Patan                          81Patan5
Sumnima Goja                   Nepal                          6                              Bazaar                         Mangal                         91Mangal6
```

Figure 75: Information Query No. 10

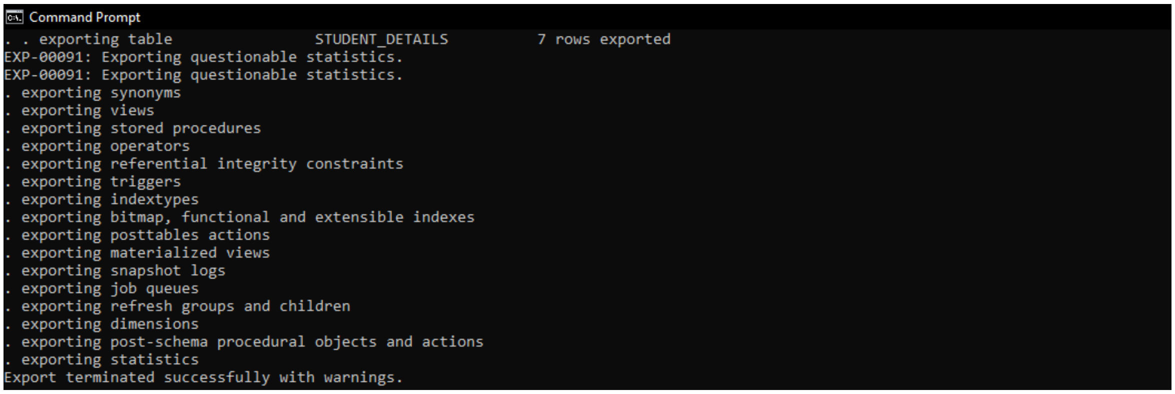
4.2. Creating Dump file



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

C:\Users\DELL>E:

E:\>cd Islington

E:\Islington>Exp Softwarica/Reenukoju file = Softwarica.dmp

Export: Release 11.2.0.2.0 - Production on Sun Dec 20 11:26:13 2020

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


Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
server uses AL32UTF8 character set (possible charset conversion)
. exporting pre-schema procedural objects and actions
. exporting foreign function library names for user SOFTWARICA
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user SOFTWARICA
About to export SOFTWARICA's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export SOFTWARICA's tables via Conventional Path ...
. . exporting table                        ADDRESS         12 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                      ADMISSION          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table              ADMISSION_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                          CLASS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         COURSE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 COURSE_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
```



```
. . exporting table                     INSTRUCTOR          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table             INSTRUCTOR_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table              INSTRUCTOR_SALARY          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 MODULE_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         PERSON         14 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                     PERSON_DOB         12 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                   PERSON_HOUSE         12 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                  SESSION_CLASS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 SESSION_MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                  SPECIFICATION          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table          SPECIFICATION_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                SPEC_ENROLLMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        STUDENT          8 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                STUDENT_DETAILS          7 rows exported
EXP-00091: Exporting questionable statistics.
```



```
. . exporting table                STUDENT_DETAILS          7 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 76: Creating dump file Softwarica.dmp

4.3. Drop Table

DROP TABLE Student_Details;

DROP TABLE Instructor_Details;

DROP TABLE Specification_Details;

DROP TABLE Module_Details;

DROP TABLE Session_Module;

DROP TABLE Course_Details;

DROP TABLE Admission_Details;

DROP TABLE Session_Class;

DROP TABLE Person;

DROP TABLE Person_DOB;

DROP TABLE Address;

DROP TABLE Person_House;

DROP TABLE Instructor;

DROP TABLE Instructor_Salary;

DROP TABLE Class;

DROP TABLE Module;

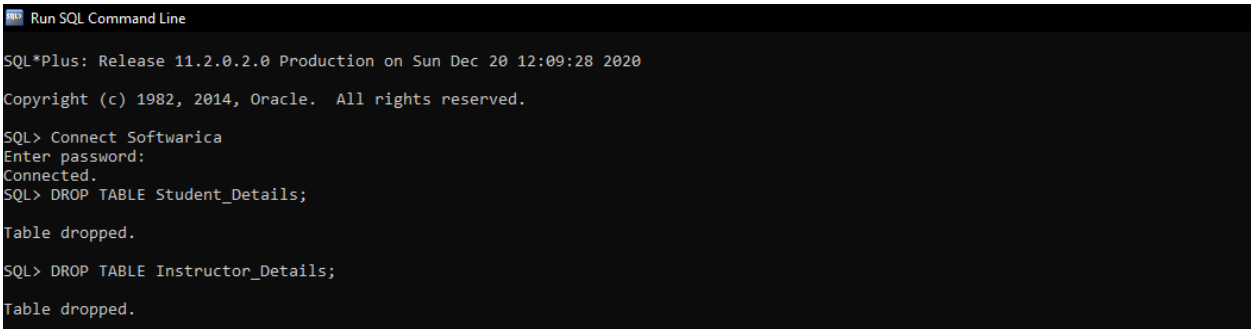
DROP TABLE Course;

DROP TABLE Admission;

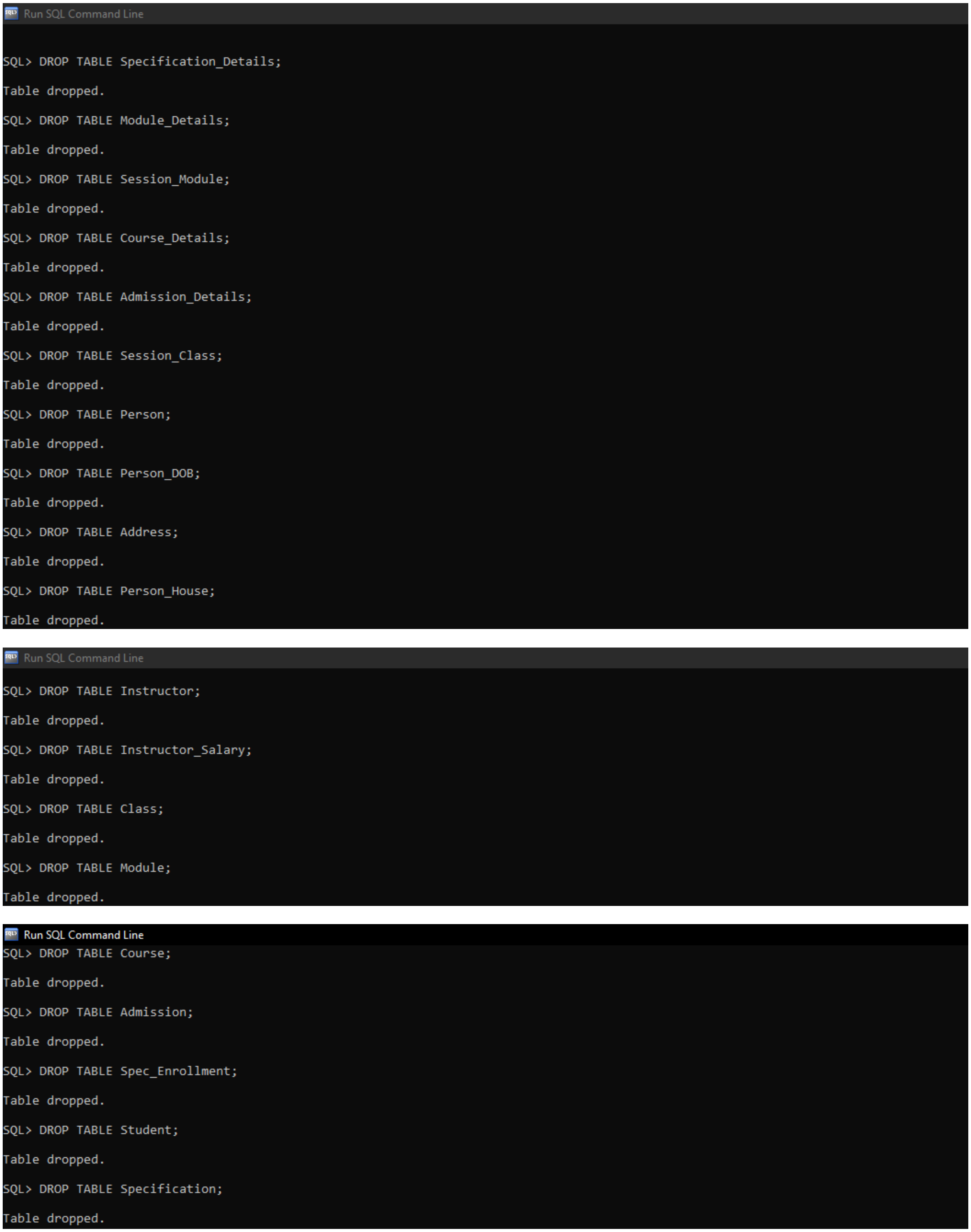
DROP TABLE Spec_Enrollment;

DROP TABLE Student;

DROP TABLE Specification;



```
SQL*Plus: Release 11.2.0.2.0 Production on Sun Dec 20 12:09:28 2020

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> Connect Softwarica
Enter password:
Connected.
SQL> DROP TABLE Student_Details;

Table dropped.

SQL> DROP TABLE Instructor_Details;

Table dropped.
```



```
SQL> DROP TABLE Specification_Details;

Table dropped.

SQL> DROP TABLE Module_Details;

Table dropped.

SQL> DROP TABLE Session_Module;

Table dropped.

SQL> DROP TABLE Course_Details;

Table dropped.

SQL> DROP TABLE Admission_Details;

Table dropped.

SQL> DROP TABLE Session_Class;

Table dropped.

SQL> DROP TABLE Person;

Table dropped.

SQL> DROP TABLE Person_DOB;

Table dropped.

SQL> DROP TABLE Address;

Table dropped.

SQL> DROP TABLE Person_House;

Table dropped.
```

```
SQL> DROP TABLE Instructor;

Table dropped.

SQL> DROP TABLE Instructor_Salary;

Table dropped.

SQL> DROP TABLE Class;

Table dropped.

SQL> DROP TABLE Module;

Table dropped.
```

```
SQL> DROP TABLE Course;

Table dropped.

SQL> DROP TABLE Admission;

Table dropped.

SQL> DROP TABLE Spec_Enrollment;

Table dropped.

SQL> DROP TABLE Student;

Table dropped.

SQL> DROP TABLE Specification;

Table dropped.
```

Figure 77: Droping all tables of the database

5. CONCLUSION

There were many difficulties that I faced while doing the coursework. The Online classes were difficult to understand. Therefore I had to watch the videos repeatedly to understand the session. It was a relief to find the recorded videos in Google Classroom without the videos I'd be clueless.

The main problem that I faced doing the coursework was normalizing the database from its un-normalized form to the third normalized form. The basics of normalizations were very blurry to me at the first. Later on by repeatedly trying to understand the theory of normalization through slides, videos and online sites like tutorialspoint, w3schools, and also the help of the tutors, I finally learned the normalization process and normalized the database.

Another problem that I faced was doing the transactional queries some of them were easy but the rest were harder to do. It took me a long period of time to do the hard ones but the rest of them were solved in minutes.

The coursework has acknowledged me about the database management system in any kind of company and has made me capable of identifiying the entities and attributes to create entity relationship diagram and normalize them up to its third normalized form using the scenarios set up by the company.

While doing the coursework I faced a lot of problems but with the help of tutors, lecture and slides in google classroom and a bit of research the problems were solved and the coursework was completed. During this process I learned a lot about creating and managing a database of a college. The database that was created could only store the data of students, teachers, courses but there are also cafeteria services, other workers whose data should be stored. But with the knowledge that I gained through the coursework I know how it works.

References

Chapple, M., 2020. *Lifewire.* [Online] Available at: https://www.lifewire.com/database-normalization-basics-1019735
[Accessed 6 December 2020].

GeeksforGeeks, 2019. *GeeksforGeeks.* [Online] Available at: https://www.geeksforgeeks.org/sql-insert-statement/
[Accessed 14 12 2020].

Singh, C., n.d. *Beginnersbook.* [Online] Available at: https://beginnersbook.com/2015/05/normalization-in-dbms/
[Accessed 6 December 2020].

tutorialspoint, 2019. *tutorialspoint.com.* [Online] Available at: https://www.tutorialspoint.com/sql/sql-constraints.htm
[Accessed 13 12 2020].

Tutorialspoint, 2020. *Tutorialspoint.* [Online] Available at: https://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm#:~:text=for%20designing%20databases.-,Entity,that%20give%20them%20their%20identity.
[Accessed 4 December 2020].

Tutorialspoint, n.d. *Tutorialspoint.* [Online] Available at: https://www.tutorialspoint.com/sql/sql-create-table.htm#:~:text=Creating%20a%20basic%20table%20involves%20naming%20the%20table,table_name%28%20column1%20datatype%2C%20column2%20datatype%2C%20column3%20datatype%2C%20..
[Accessed 13 12 2020].

W3schools, 2018. *W3schools.* [Online] Available at: https://www.w3schools.com/sql/sql_select.asp
[Accessed 14 12 2020].