# CC5051NI Databases

## 100% Individual Coursework

**Autumn 2024**

**Credit: 15 Semester Long Module**

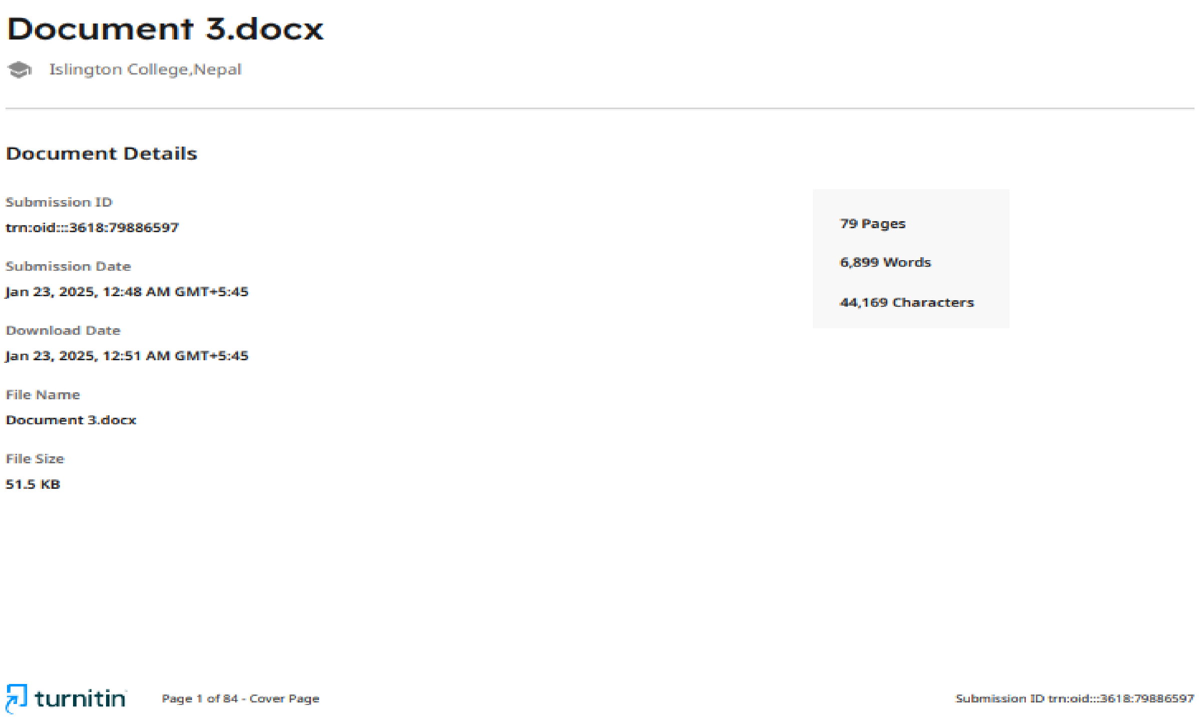
**Student Name:** Safal Kandel.

**London Met ID: 23056337**

**Assignment Submission Date:31/12/2024**.

**Word Count: 4594**

*I confirm that I understand my coursework needs to be submitted online via My Second Teacher 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 mark of zero will be awarded.*



# Document 3.docx

Islington College,Nepal

## Document Details

Submission ID
trn:oid:::3618:79886597

Submission Date
Jan 23, 2025, 12:48 AM GMT+5:45

Download Date
Jan 23, 2025, 12:51 AM GMT+5:45

File Name
Document 3.docx

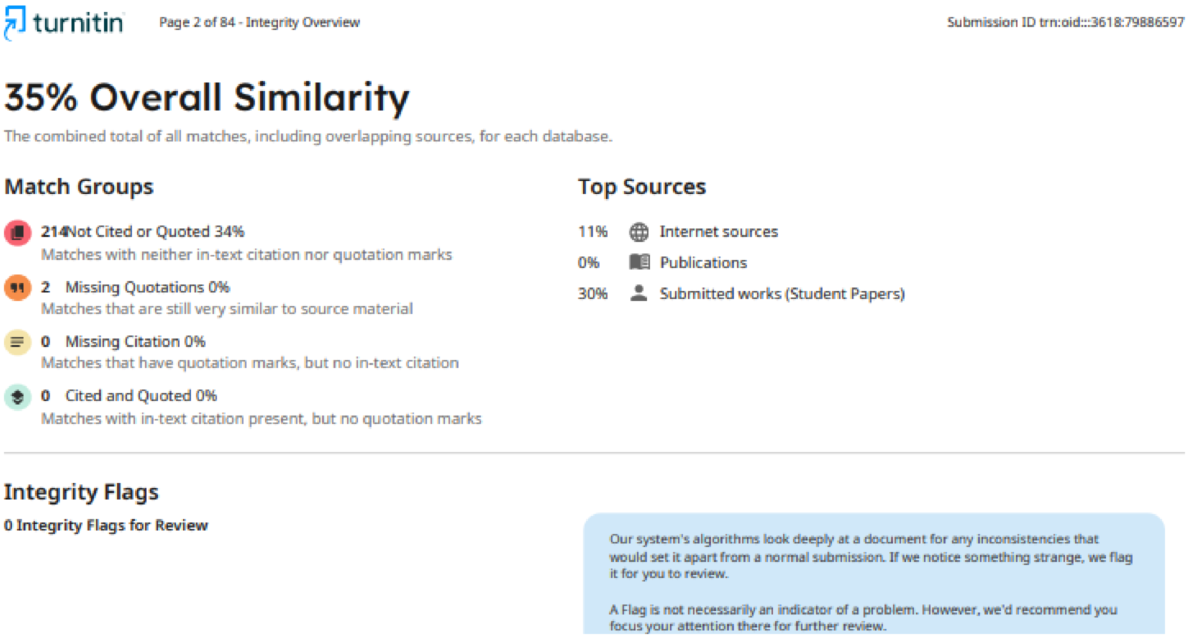
File Size
51.5 KB

79 Pages

6,899 Words

44,169 Characters

turnitin Page 1 of 84 - Cover Page Submission ID trn:oid:::3618:79886597



turnitin Page 2 of 84 - Integrity Overview Submission ID trn:oid:::3618:79886597

# 35% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Match Groups

- 214Not Cited or Quoted 34%
  Matches with neither in-text citation nor quotation marks
- 2 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

## Top Sources

| | |
|---|---|
| 11% | Internet sources |
| 0% | Publications |
| 30% | Submitted works (Student Papers) |

## Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

# Contents

# 1.Introduction

Ms. Mary has come with a revolutionary E-Classroom Platform that bridges the gaps in virtual education, integrating students, teachers, programs, and modules seamlessly into an academic ecosystem for continuous growth for Southwestern college established in 2018 AD. This digital solution, driven by this new-age, state-of-the-art platform, will bring a facelift to the evolving educational landscape at colleges, focusing on operation efficiency improvement. The E-Classroom Platform ensures that all processes regarding student admissions, module administration, assessment tracking, and the delivery of resources are smoothly executed and meaningful interactions among its key stakeholders are fostered by digitizing such processes. The structured yet flexible approach of the platform aspires to create an intuitive, engaging learning experience that will afford both students and educators the power to thrive in a modern, interconnected academic environment.

The platform digitizes core academic processes, including student admissions, module administration, assessment tracking, and resource delivery. By streamlining these operations, it eliminates unnecessary administrative burdens, allowing educators to focus on teaching and students to concentrate on learning. Moreover, it offers a highly structured yet adaptable framework, catering to the unique needs of institutions while ensuring that all stakeholders have a seamless and intuitive experience. Its design encourages collaboration and engagement, bridging gaps that often exist in virtual education.

More than just a tool, the E-Classroom Platform is an innovative solution that enhances both teaching and learning. It empowers students by providing them with access to interactive resources and tools that make learning more engaging, while educators are equipped with systems to efficiently manage their classes and track student progress. This flexible and intuitive approach ensures that the platform is not just a response to the current needs of education but a visionary step toward a more connected and impactful academic future.

*Figure 1: Southwestern College.*

## 2. Current Business Activities and Operations

The college currently operates multiple degree programs in various disciplines, such as BSc in Computing, Networking, and Multimedia. Key activities include:

### 2.1 Program Management:

Students enroll in one of several programs, each comprising mandatory modules that define their academic path.

### 2.2 Module Delivery and Assessment:

Each teaching module is assigned to certain teachers. There are modules between programs (such as Programming in Computing and Multimedia) where duplication of modules occurs. Each module has one or more assessments to be graded for performance measurement for students.

### 2.3 Resource Management:

Every module is equipped with resource-analyzed structure (for instance, video lectures, notes) which is required to be completed by the students in a prescribed order so that he/she **steps** up improve learning accordingly.

### 2.4 Announcements:

Most instructors provide reminders for their students through announcements related to the module for deadlines, additional resources, or changes in the syllabus. The system or partially digitized process suffers from inefficiencies, lack of scalability, and limited data integration. The proposed databasing system, therefore, would make all three possible through fully automating the operations while optimizing it.

## 3. Business Rules Derived from Operational Procedures

To maintain consistency and efficiency, the following business rules are proposed:

- One student can be enrolled in only one of the programs, and every program has many students.
- A program has many modules, and modules can be part of many programs.
- A teacher is assigned to teach specific modules, and a module consists of different teachers.
- A teacher can post announcements for their respective module only, and announcements can be posted about different modules.
- A module has single or multiple assessments, and each assessment is linked to only one module.
- Student can see result of each module.
- Every module can have multiple resources but resources will only belong to one module.
- Each assessment can have multiple results since every student taking an assessment will generate a different result.

# 4.Entities and Attributes

## 1. Student

| S.no | Attribute_Name | Datatype | Size | Constraint |
|---|---|---|---|---|
| 1 | Student_Id | number | 10 | Primary key |
| 2 | Student_Name | character | 50 | Not null |
| 3 | Student_Email | date | - | Not null |
| 4 | Student_Address | character | 100 | unique |

*Table 1- Student(Entities and Atrributes)*

## 2. Program

| S.no | Attribute_Name | Datetype | Size | Constraint |
|---|---|---|---|---|
| 1 | Program_id | Number | 10 | Primary key |
| 2 | Program_name | character | 50 | Not null |
| 3 | Program_Duration | character | 255 | Not null |
| 4 | Program_Title | Number | 3 | Not null |

*Table 2- Program Table.*

## 3. Module

| S.no | Attribute_Name | Datatype | Size | Constraint |
|---|---|---|---|---|
| 1 | Module_id | number | 10 | Primary key |
| 2 | Module Name | character | 50 | Not null |
| 3 | Credits | number | 10 | Not Null |
| 4 | Resource_id | number | 10 | Not Null |
| 5 | Resource title | character | 100 | Not null |
| 6 | Resource type | character | 10 | Not null |
| 7 | Resource status | character | 50 | Not null |
| 8 | Assessment_id | number | 10 | Not null |
| 9 | Assessment title | character | 50 | Not null |
| 10 | Assessment deadline | date | - | Not null |
| 11 | weightage | number | 3 | null |
| 12 | Result_id | number | 10 | Not Null |
| 13 | Result total mark | number | 5,2 | Not null |
| 14 | Result remark | character | 10 | Not null |
| 15 | Announcement_id | Number | 10 | Not null |
| 16 | Announcement_Title | character | 50 | Not null |
| 17 | Announcement date | date | - | Null allowed |
| 18 | Announcement description | character | 100 | Null allowed |
| 19 | Teacher_id | number | 10 | Not null |
| 20 | Teachers_name | character | 50 | Not null |
| 21 | Teachers_Email | character | 100 | Not Null |

*Table 3- Module Table.*

## 4. Initial Entity Relationship Diagram.

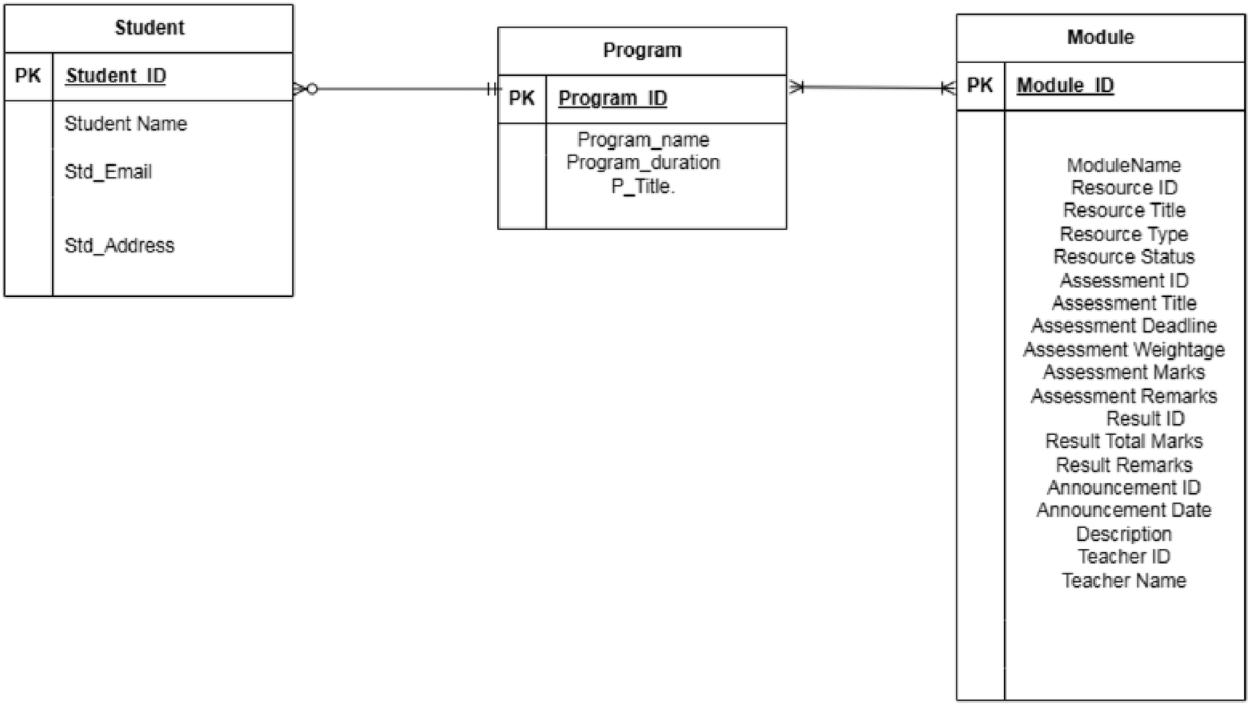


*Figure 2 Entity Relationship Diagram.*

## 5. Normalization

Normalization in database design is the process of organizing data into related, smaller tables to reduce data redundancy and improve data integrity. This has to do with breaking down a big table that could have some anomalies into efficient forms; this would be 1NF-just the atomic values and elimination of the repeating groups, 2NF-eliminate the partial dependencies by ensuring non-key attributes depend on the entirety of the primary key, and 3NF-transitive dependency would need elimination, making sure all the non-key attributes depend on a primary key. This process enhances consistency, reduces redundancy, and makes maintenance easier while the database remains scalable and efficient.

### UNF

**Unnormalized Form (UNF)** is the raw data representation where all information is stored in a single table with repeating groups or arrays, lacking structure or normalization. It often includes nested and duplicate data.

(student_id,student_Name,Student_Email,Student_Address,program_id,program_name,program_duration,program_Title{Module_id,Module Name,credits{Resource_id,Resource title,Resourse type,Resource Status},{Assessment_id,Assessment title,Assessment deadline,Weightage,Result id,Result total marks,Result remark},{Teacher_id,Teacher_name,Teacher_email{Announcement_id,Announcement_Title,Announcement_date, Announcement_Description }})

## 1NF

A relation violates the First Normal Form (1NF) if it contains composite attributes (attributes combining multiple pieces of information) or multi-valued attributes (attributes storing multiple values for a single entity). To comply with 1NF, each attribute must hold a single, atomic value, meaning each cell in the table contains only one value. This ensures the data is unambiguous and easy to query, update, and manage. For example, storing multiple phone numbers or subjects in a single attribute violates 1NF, but breaking them into separate rows for each value ensures the table adheres to 1NF, promoting clarity and eliminating redundancy (Geeksforgeeks, 2025)**.**

**Student-1**

(Student_id,Student_Name,Student_Email,Student_Addreess,program_id,program_name,program_duration,program_Title)

**Module-1**

(Student_ id,Module_Id,Module Name,credits)

**Resource-1**

(Student_ id,Module_Id,Resource Id, Resource ID,Resource title,Resourse type,Resource Status)

**Assessment-1**

(Student_ id,Module_Id,Assessment Id,Assessment title,Assessment deadline,Weightage,Result id,Result total marks,Result remark)

**Teacher-1**

(Student_ id,Module_Id,Teacher_id,Teacher_name,Teacher_email )

**Announcement-1**

(Student_id,Module_Id,Teacher_id,Announcement_id,Announcement_Title,Announcement_date,Announcement_Description)

## 2NF

The First Normal Form (1NF) focuses solely on eliminating repeating groups and ensuring that all attributes contain atomic (single) values, but it does not address redundancy. This is why the Second Normal Form (2NF) is introduced. A table is considered to be in 2NF if it satisfies two conditions: it is already in 1NF, and there are no partial dependencies. This means that every non-key attribute must be fully dependent on the entire primary key, rather than just a part of it. Partial dependency typically occurs in tables with composite primary keys, where some attributes depend only on a subset of the key rather than the full key. By removing partial dependencies, 2NF reduces redundancy and enhances data consistency (Chris, 2022).

**Checking Functional dependency:**

**Module:**

Module id→ module Name,credit.

Student_id→×

**Teacher:**

Teacher id→Teacher_id,Teacher_name,Teacher_email

Student_id→×

Module_id→×

**Announcement:**

teacher_id→×

student_id→×

module_id→×

Announcement_id-Announcement_Title,Announcement_date

**Resources:**

Resource_id→ Resource_title,Resource_Type,

Student_id,Module_id→Resource_status

**Assessment:**

Assessment_id → Assessment title,Assessment deadline,Weightage

Student_id, Module_id→ Result_id,Result total marks,Result remark

2NF-

**Student-2**

(Student_id,Student_Name,Student_Email,Student_Addreess,program_id,program_name,program_duration,program_Title)

**Module-2**

(Module_id, Module Name,Module credits)

**Student-module-2**

(Student_ id*,Module id*)

**Resource-2**

(Resource_id, Resource title,Resource Type)

**Student-module-Resource-2**

(Student_id*,Module_id*,Resource Status)

**Assessment-2**

(Assessment_id, Assessment title,Assessment deadline,Weightage)

**Student-module-Assessment-2**

(Student_ id*,Module_ id*,Assessment_ id,Result_id,Result total marks,Result remark)

**Teacher-2**

(Teacher_ id,Teacher_name,Teacher_email )

**Student-Module-teacher-2**

(Student_ id*,Module_id*,Teacher_ id*)

**Announcement-2**

(Announcement_id,Announcement Title,Announcement Date,Announcement Description)

**Student-annoucement-2**

(Student_id,Module_id,Teacher_ id,Announcement_ id)

### 3NF

**Third Normal Form (3NF)**: A relation is in Third Normal Form (3NF) if it satisfies the conditions of Second Normal Form (2NF) and eliminates transitive dependencies, meaning no non-key attribute depends on another non-key attribute. In 3NF, all non-key attributes must depend only on the primary key, ensuring that the relation is free from redundancy and anomalies caused by indirect dependencies. This normalization step improves data integrity and results in a well-organized and efficient database design, reducing the risk of inconsistencies during data updates or modifications (Geeksforgeeks, 2025).

3NF-

1. **Student Table**: This table holds information about students, like their ID, name, email, address, and the program they are enrolled in. The program_id links each student to a specific program.
2. **Program Table**: This table lists details about the programs available, such as the program’s name, duration, and title. Each program is uniquely identified by program_id.
3. **Module Table**: Modules, which are parts of a program, are listed here. It includes the module's name, ID, and credits. Each module has a unique module_id.
4. **Student-Module Table**: This is a connection table that links students to the modules they are taking. It records which student is taking which module by storing their respective IDs.
5. **Resource Table**: This table stores information about various resources available to students, such as their title and type. Each resource is identified by a resource_id.
6. **Student-Module-Resource Table**: This table tracks the status of resources assigned to students within specific modules. It connects students, modules, and the status of each resource they have access to.
7. **Student-Resource Table**: This table links students to specific resources, showing which resources are assigned to them in which modules.

8. **Assessment Table**: This table includes details about assessments, such as the assessment’s ID, title, deadline, and weightage (importance). Each assessment is identified by a assessment_id.

9. **Student-Module-Assessment Table**: This table maps students to the assessments in the modules they are taking, showing which assessment each student has for each module.

10. **Student-Module-Assessment-Result Table**: This table stores the results of students' assessments, including their marks and any feedback. It links students, modules, assessments, and results.

11. **Teacher Table**: This table holds information about teachers, like their ID, name, and email. Each teacher has a unique teacher_id.

12. **Student-Module-Teacher Table**: This table links students with the teachers for the modules they are enrolled in, showing which teacher is teaching which student in which module.

13. **Announcement Table**: This table includes announcements made by teachers, such as the title, date, and description. Each announcement has a unique announcement_id.

14. **Student-Announcement Table**: This table tracks which students have received which announcements from teachers for specific modules, linking students, modules, teachers, and announcements.

.

**Student-3**

(Student_id,Program id*,Student_Name,Student_Email,Student_Addreess)

**Program - 3**

(program_id,program_name,program_duration,program_Title)

**Module-3**

(Module_id, Module Name,Module credits)

**Student-module-3**

(Student_ id*,Module_id*)

**Resource-3**

(Resource id, Resource title,Resource Type)

**Student-module-Resource-3**

(Student_id*,Module_id*,Resource_id*,Resource Status)

**Assessment-3**

(Assessment_id, Assessment title,Assessment deadline,Weightage)

**Student-module-Assessment-Result-3**

(Student_id*,Module_id*, Assessment_id*,Result id)

**Result -3**

(Result_id,Result total marks,Result remark)

**Teacher-3**

(Teacher_ id,Teacher_name,Teacher_email )

**Student-Module-teacher-3**

(Student_ id*,Module_ id*,Teacher_ id*)

**Announcement-3**

(Announcement_id,Announcement Title,Announcement Date,Announcement Description)

**Student-annoucement-3**

(Student_id*,Module _id*,Teacher_ id*,Announcement_ id*)

## 6.Data Dictionary.

### 1. Student

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | student_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | student_name | Character | 50 | Not Null | - |
| 3 | enrollment_date | Date | - | Not Null | - |
| 4 | student_email | Varchar | 50 | Not Null, Unique | - |
| 5 | program_id | Number | 10 | Foreign Key (references Program-3) | - |

*Table 4- Student Table(Data Dictionary)*

### 2. Program

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | program_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | program_name | Character | 50 | Not Null | - |
| 3 | program_duration | Number | 3 | Not Null | - |
| 4 | program_title | Character | 100 | Not Null | - |

*Table 5- Program Table.*

### 3. Module

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | module_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | module_name | Character | 50 | Not Null | - |
| 3 | module_credits | Number | 3 | Not Null | - |

*Table 6- Module Table.*

### 4. Student-Module.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | student_id | Number | 10 | Foreign Key (references Student-3) | Part of Composite Primary Key |
| 2 | module_id | Number | 10 | Foreign Key (references Module-3) | Part of Composite Primary Key |

*Table 7- Student_Module Table.*

## 5. Resource

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | resource_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | resource_title | Character | 100 | Not Null | - |
| 3 | resource_type | Character | 50 | Not Null | - |

*Table 8- Resource Table.*

## 6. Student-Module-Resource.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | student_id | Number | 10 | Foreign Key (references Student-3) | Part of Composite Primary Key |
| 2 | module_id | Number | 10 | Foreign Key (references Module-3) | Part of Composite Primary Key |
| 3 | resource_id | Number | 10 | Foreign Key (references Resource-3) | Part of Composite Primary Key |
| 4 | resource_status | Character | 20 | Not Null | - |

*Table 9- Student-Module-Resource Table.*

## 7. Assessment.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | assessment_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | assessment_title | Character | 100 | Not Null | - |
| 3 | assessment_deadline | Date | - | Not Null | - |
| 4 | weightage | Number | 3 | Not Null | - |

*Table 10- Assessment Table.*

## 8. Student-Module-Assessment-Result.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | student_id | Number | 10 | Foreign Key (references Student-3) | Part of Composite Primary Key |
| 2 | module_id | Number | 10 | Foreign Key (references Module-3) | Part of Composite Primary Key |
| 3 | assessment_id | Number | 10 | Foreign Key (references Assessment-3) | Part of Composite Primary Key |
| 4 | result_id | Number | 10 | Foreign Key (references Result-3) | - |

*Table 11- Student-Module-Assessment Table.*

### 9. Result.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | result_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | result_total_marks | Number | 5 | Not Null | - |
| 3 | result_remark | Character | 100 | - | - |

*Table 12- Result Table.*

### 10. Teacher.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | teacher_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | teacher_name | Character | 50 | Not Null | - |
| 3 | teacher_email | Varchar | 50 | Not Null, Unique | - |

*Table 13- Teacher Table.*

### 11. Student-Module-Teacher.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | student_id | Number | 10 | Foreign Key (references Student-3) | Part of Composite Primary Key |
| 2 | module_id | Number | 10 | Foreign Key (references Module-3) | Part of Composite Primary Key |
| 3 | teacher_id | Number | 10 | Foreign Key (references Teacher-3) | Part of Composite Primary Key |

*Table 14- Student-Module-Teacher Table.*

### 12. Announcement.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | announcement_id | Number | 10 | Primary Key, Not Null, Unique | - |
| 2 | announcement_title | Character | 100 | Not Null | - |
| 3 | announcement_date | Date | - | Not Null | - |
| 4 | announcement_description | Character | 255 | - | - |

*Table 15- Announcement Table.*

## 13. Student-Announcement.

| Sno | Attribute | Datatype | Size | Constraints | Composite Constraint |
|---|---|---|---|---|---|
| 1 | student_id | Number | 10 | Foreign Key (references Student-3) | Part of Composite Primary Key |
| 2 | module_id | Number | 10 | Foreign Key (references Module-3) | Part of Composite Primary Key |
| 3 | teacher_id | Number | 10 | Foreign Key (references Teacher-3) | Part of Composite Primary Key |
| 4 | announcement_id | Number | 10 | Foreign Key (references Announcement-3) | Part of Composite Primary Key |

*Table 16- Student-Announcement Table.*

## 6. Final ERD

The final ERD represents a normalized and optimized database design, following all the requirements and business rules outlined for the "E-Classroom Platform." It reflects a structured relationship between entities and embodies all the constraints derived during the normalization process.

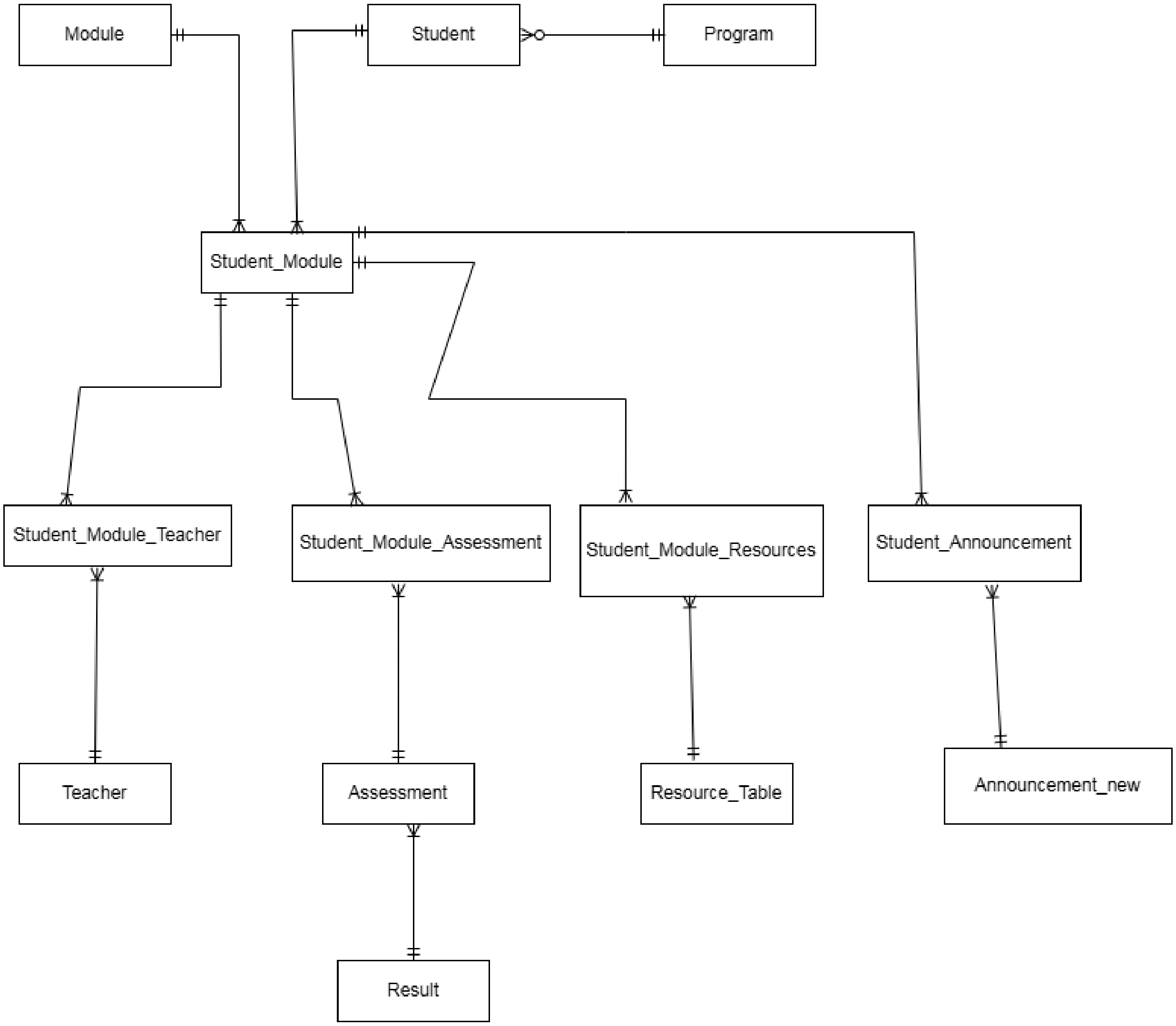
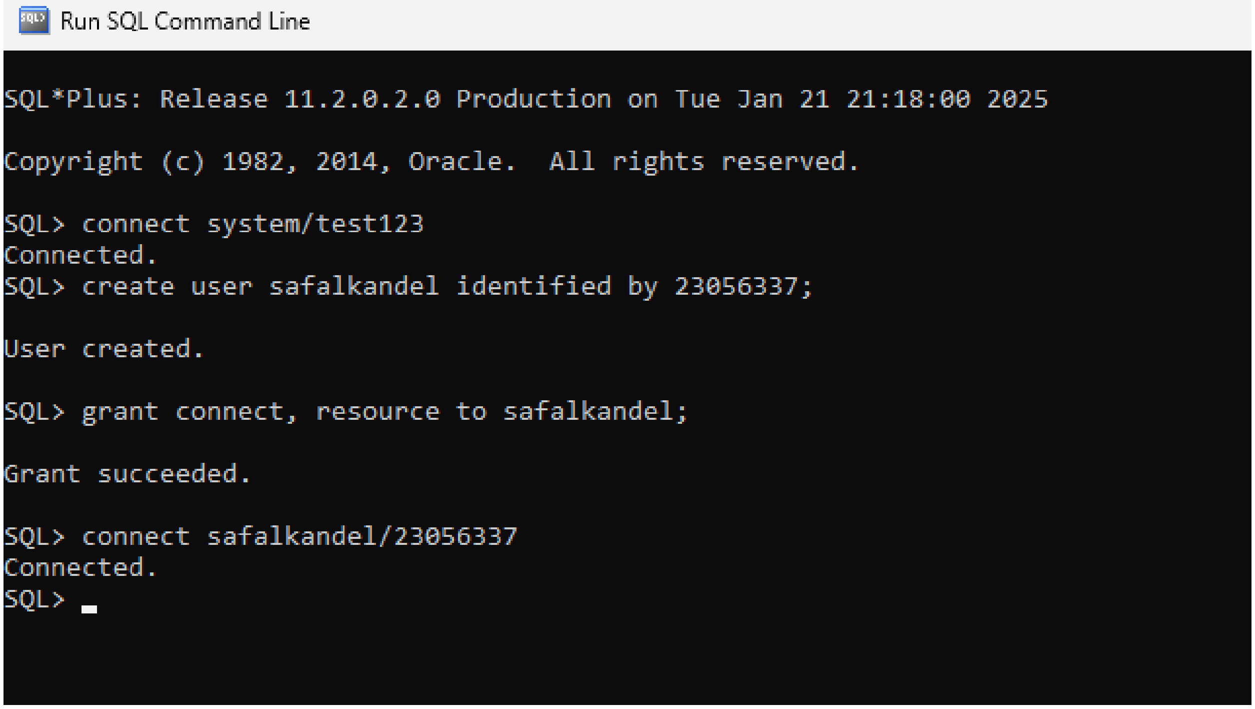


*Figure 3: Final ERD*

# 7. Implementation

## 1. Creating User.

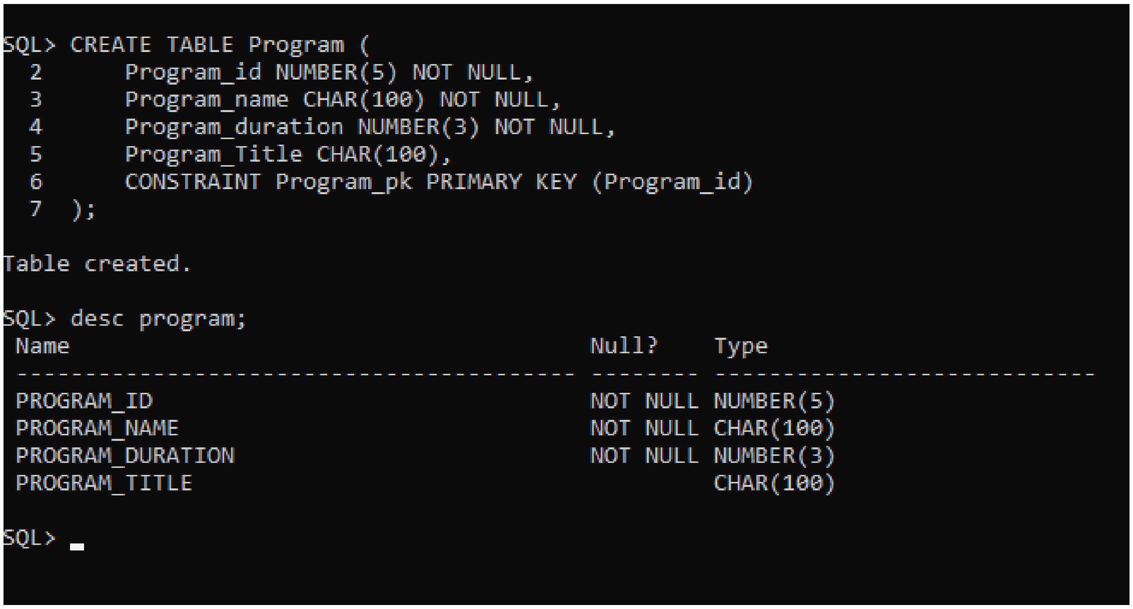


```
SQL*Plus: Release 11.2.0.2.0 Production on Tue Jan 21 21:18:00 2025

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

SQL> connect system/test123
Connected.
SQL> create user safalkandel identified by 23056337;

User created.

SQL> grant connect, resource to safalkandel;

Grant succeeded.

SQL> connect safalkandel/23056337
Connected.
SQL>
```

*Figure 4: Creating User.*

# 2.Creating Tables.

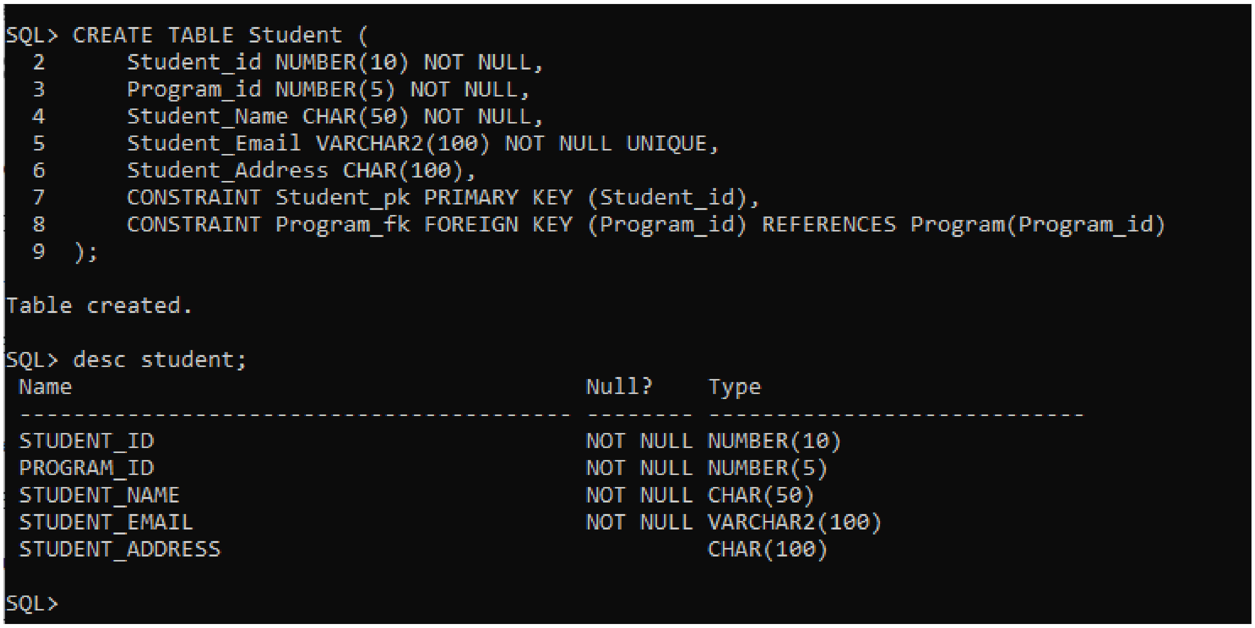
## 2.1 Create Program Table.



```
SQL> CREATE TABLE Program (
  2      Program_id NUMBER(5) NOT NULL,
  3      Program_name CHAR(100) NOT NULL,
  4      Program_duration NUMBER(3) NOT NULL,
  5      Program_Title CHAR(100),
  6      CONSTRAINT Program_pk PRIMARY KEY (Program_id)
  7  );

Table created.

SQL> desc program;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROGRAM_ID                                NOT NULL NUMBER(5)
 PROGRAM_NAME                              NOT NULL CHAR(100)
 PROGRAM_DURATION                          NOT NULL NUMBER(3)
 PROGRAM_TITLE                                      CHAR(100)

SQL>
```

*Figure 5: Creating Tables.*

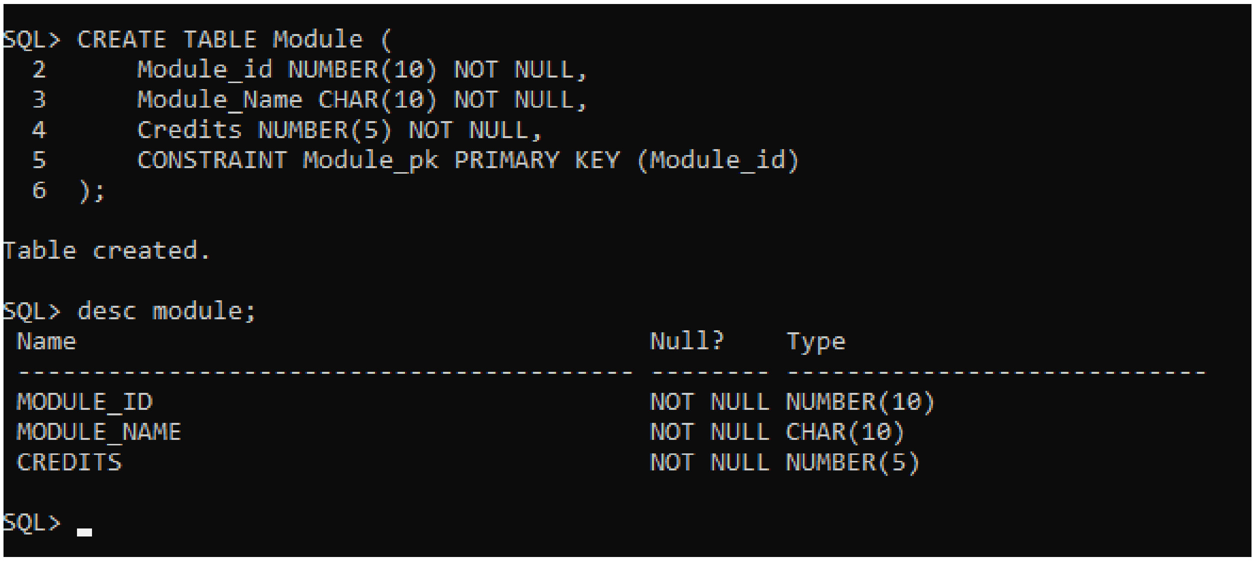
## 2.2 Create Student Table.



```
SQL> CREATE TABLE Student (
  2      Student_id NUMBER(10) NOT NULL,
  3      Program_id NUMBER(5) NOT NULL,
  4      Student_Name CHAR(50) NOT NULL,
  5      Student_Email VARCHAR2(100) NOT NULL UNIQUE,
  6      Student_Address CHAR(100),
  7      CONSTRAINT Student_pk PRIMARY KEY (Student_id),
  8      CONSTRAINT Program_fk FOREIGN KEY (Program_id) REFERENCES Program(Program_id)
  9  );

Table created.

SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(10)
 PROGRAM_ID                                NOT NULL NUMBER(5)
 STUDENT_NAME                              NOT NULL CHAR(50)
 STUDENT_EMAIL                             NOT NULL VARCHAR2(100)
 STUDENT_ADDRESS                                    CHAR(100)

SQL>
```

*Figure 6: Creating Student Table.*

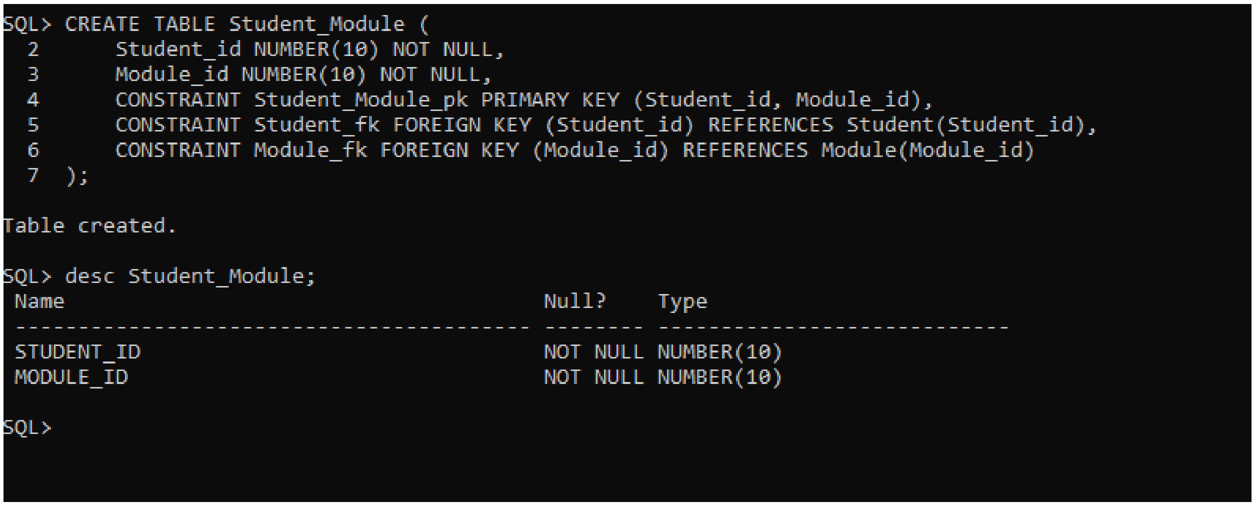
## 2.3 Create Module Table.



```
SQL> CREATE TABLE Module (
         Module_id NUMBER(10) NOT NULL,
         Module_Name CHAR(10) NOT NULL,
         Credits NUMBER(5) NOT NULL,
         CONSTRAINT Module_pk PRIMARY KEY (Module_id)
     );

Table created.

SQL> desc module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODULE_ID                                 NOT NULL NUMBER(10)
 MODULE_NAME                               NOT NULL CHAR(10)
 CREDITS                                   NOT NULL NUMBER(5)

SQL>
```

*Figure 7: Creating Module Table.*

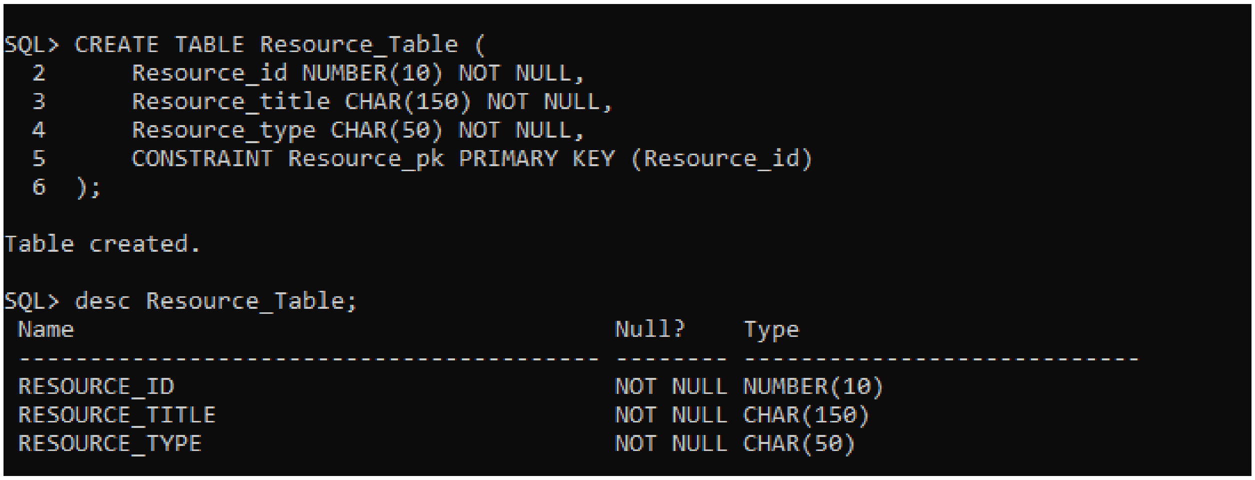
## 2.4 Create Student_Module Table.



```
SQL> CREATE TABLE Student_Module (
         Student_id NUMBER(10) NOT NULL,
         Module_id NUMBER(10) NOT NULL,
         CONSTRAINT Student_Module_pk PRIMARY KEY (Student_id, Module_id),
         CONSTRAINT Student_fk FOREIGN KEY (Student_id) REFERENCES Student(Student_id),
         CONSTRAINT Module_fk FOREIGN KEY (Module_id) REFERENCES Module(Module_id)
     );

Table created.

SQL> desc Student_Module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL NUMBER(10)

SQL>
```

*Figure 8: Creating Student_Module Table.*

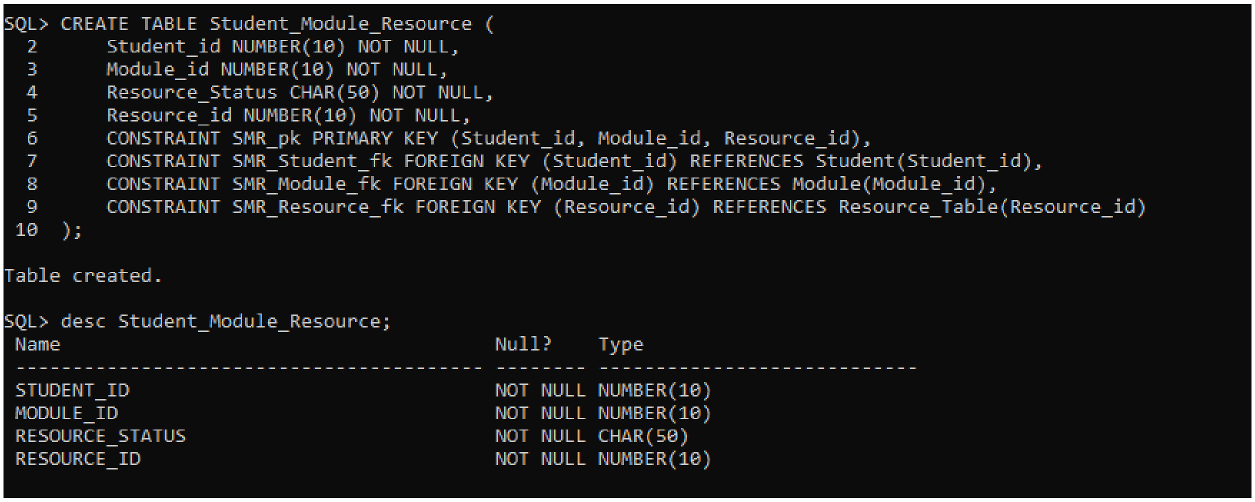
## 2.5 Create Resource_Table.



```
SQL> CREATE TABLE Resource_Table (
         Resource_id NUMBER(10) NOT NULL,
         Resource_title CHAR(150) NOT NULL,
         Resource_type CHAR(50) NOT NULL,
         CONSTRAINT Resource_pk PRIMARY KEY (Resource_id)
     );

Table created.

SQL> desc Resource_Table;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RESOURCE_ID                               NOT NULL NUMBER(10)
 RESOURCE_TITLE                            NOT NULL CHAR(150)
 RESOURCE_TYPE                             NOT NULL CHAR(50)
```

*Figure 9: Creating Resource Table.*

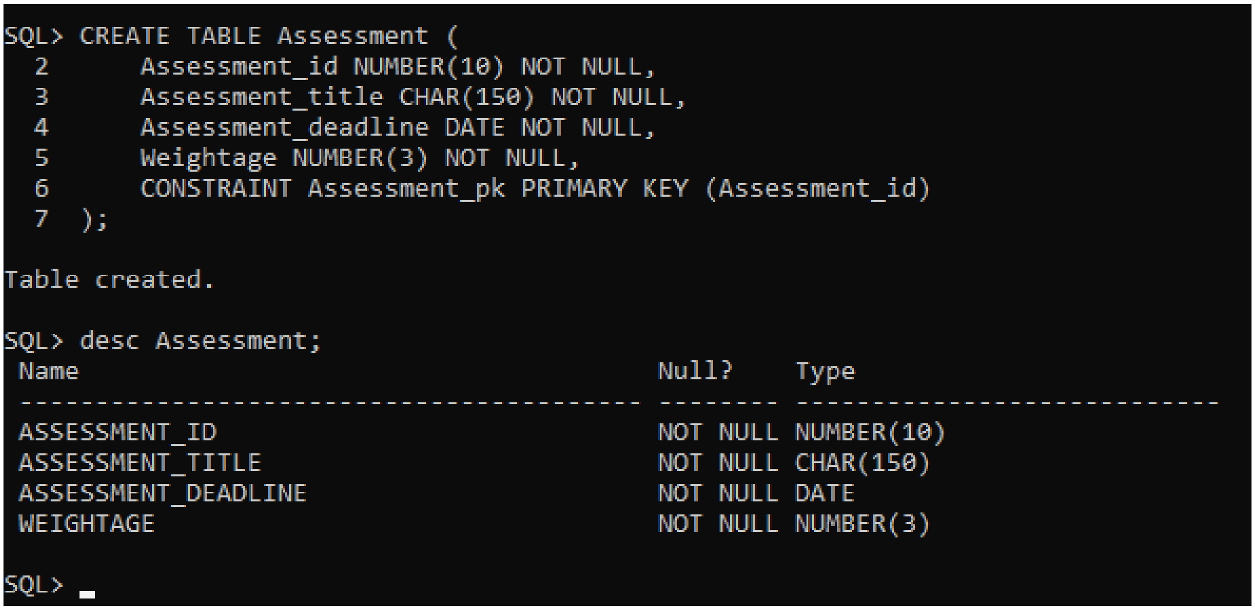
## 2.6 Create Student_Module_Resource Table.



```
SQL> CREATE TABLE Student_Module_Resource (
         Student_id NUMBER(10) NOT NULL,
         Module_id NUMBER(10) NOT NULL,
         Resource_Status CHAR(50) NOT NULL,
         Resource_id NUMBER(10) NOT NULL,
         CONSTRAINT SMR_pk PRIMARY KEY (Student_id, Module_id, Resource_id),
         CONSTRAINT SMR_Student_fk FOREIGN KEY (Student_id) REFERENCES Student(Student_id),
         CONSTRAINT SMR_Module_fk FOREIGN KEY (Module_id) REFERENCES Module(Module_id),
         CONSTRAINT SMR_Resource_fk FOREIGN KEY (Resource_id) REFERENCES Resource_Table(Resource_id)
     );

Table created.

SQL> desc Student_Module_Resource;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL NUMBER(10)
 RESOURCE_STATUS                           NOT NULL CHAR(50)
 RESOURCE_ID                               NOT NULL NUMBER(10)
```

*Figure 10: Creating Student_Module_Resource Table.*

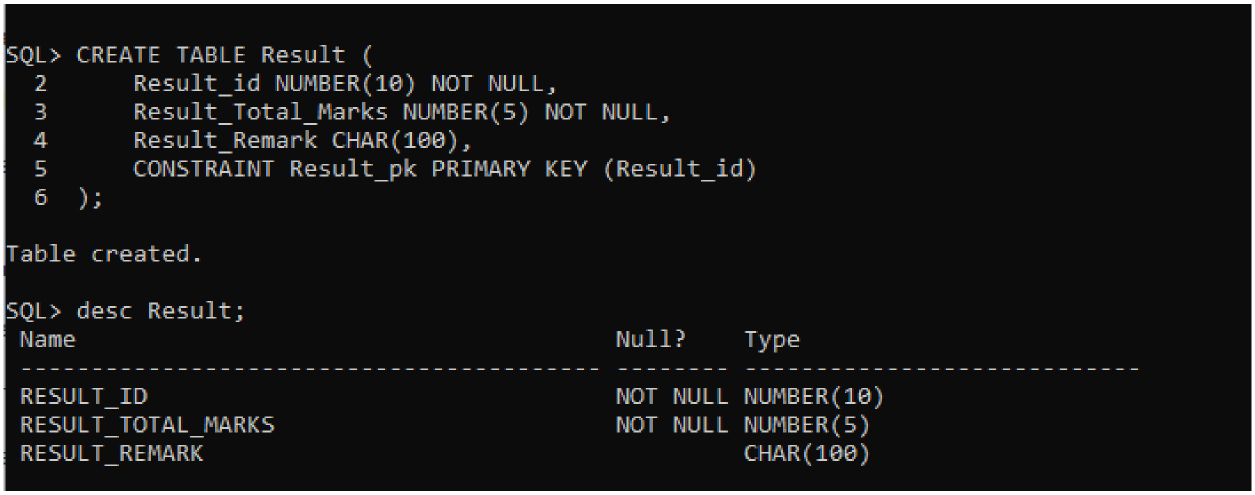
## 2.7 Create Assessment Table.



```
SQL> CREATE TABLE Assessment (
         Assessment_id NUMBER(10) NOT NULL,
         Assessment_title CHAR(150) NOT NULL,
         Assessment_deadline DATE NOT NULL,
         Weightage NUMBER(3) NOT NULL,
         CONSTRAINT Assessment_pk PRIMARY KEY (Assessment_id)
     );

Table created.

SQL> desc Assessment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ASSESSMENT_ID                             NOT NULL NUMBER(10)
 ASSESSMENT_TITLE                          NOT NULL CHAR(150)
 ASSESSMENT_DEADLINE                       NOT NULL DATE
 WEIGHTAGE                                 NOT NULL NUMBER(3)

SQL>
```

*Figure 11: Creating Assessment Table.*

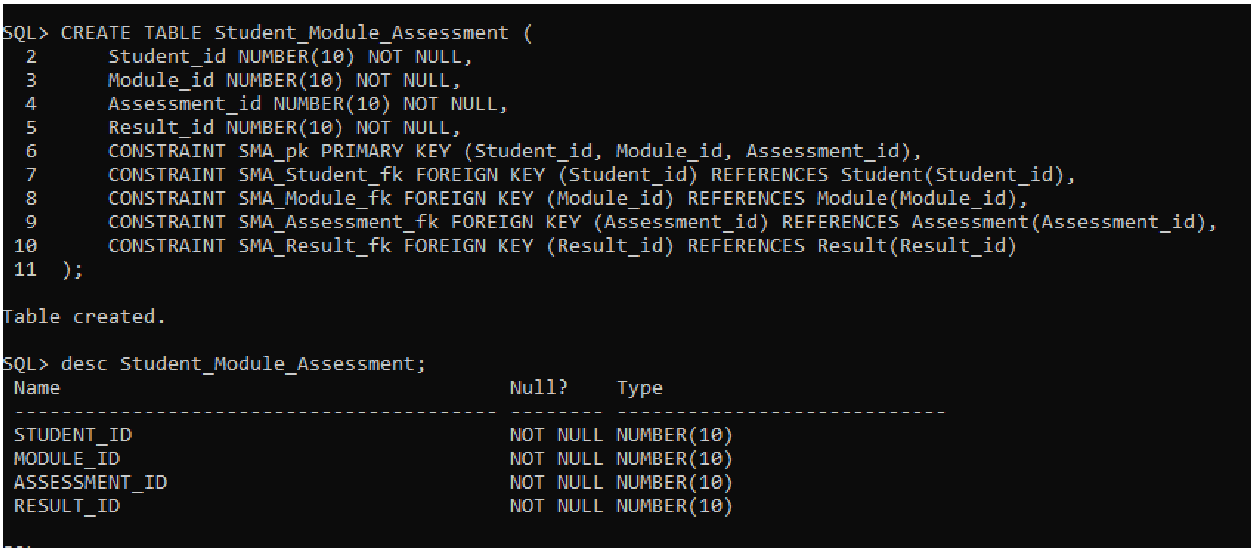
## 2.8 Create Result Table.



```
SQL> CREATE TABLE Result (
         Result_id NUMBER(10) NOT NULL,
         Result_Total_Marks NUMBER(5) NOT NULL,
         Result_Remark CHAR(100),
         CONSTRAINT Result_pk PRIMARY KEY (Result_id)
     );

Table created.

SQL> desc Result;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RESULT_ID                                 NOT NULL NUMBER(10)
 RESULT_TOTAL_MARKS                        NOT NULL NUMBER(5)
 RESULT_REMARK                                      CHAR(100)
```

*Figure 12: Creating Result Table.*

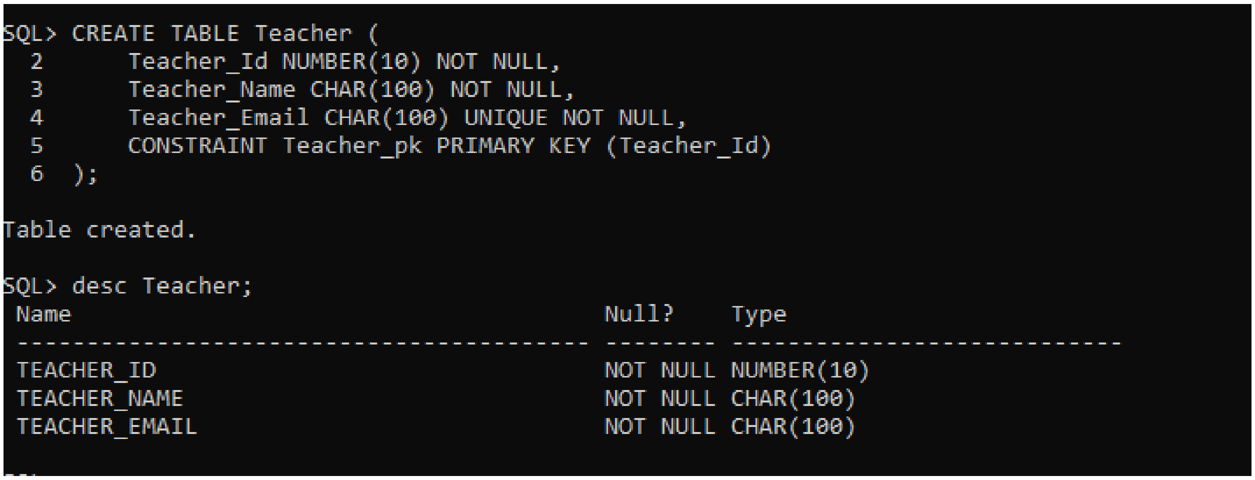
## 2.9 Create Student_Module_Assessment Table.



```
SQL> CREATE TABLE Student_Module_Assessment (
         Student_id NUMBER(10) NOT NULL,
         Module_id NUMBER(10) NOT NULL,
         Assessment_id NUMBER(10) NOT NULL,
         Result_id NUMBER(10) NOT NULL,
         CONSTRAINT SMA_pk PRIMARY KEY (Student_id, Module_id, Assessment_id),
         CONSTRAINT SMA_Student_fk FOREIGN KEY (Student_id) REFERENCES Student(Student_id),
         CONSTRAINT SMA_Module_fk FOREIGN KEY (Module_id) REFERENCES Module(Module_id),
         CONSTRAINT SMA_Assessment_fk FOREIGN KEY (Assessment_id) REFERENCES Assessment(Assessment_id),
         CONSTRAINT SMA_Result_fk FOREIGN KEY (Result_id) REFERENCES Result(Result_id)
     );

Table created.

SQL> desc Student_Module_Assessment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL NUMBER(10)
 ASSESSMENT_ID                             NOT NULL NUMBER(10)
 RESULT_ID                                 NOT NULL NUMBER(10)
```

*Figure 13: Creating Student_Module_Assessment Table.*

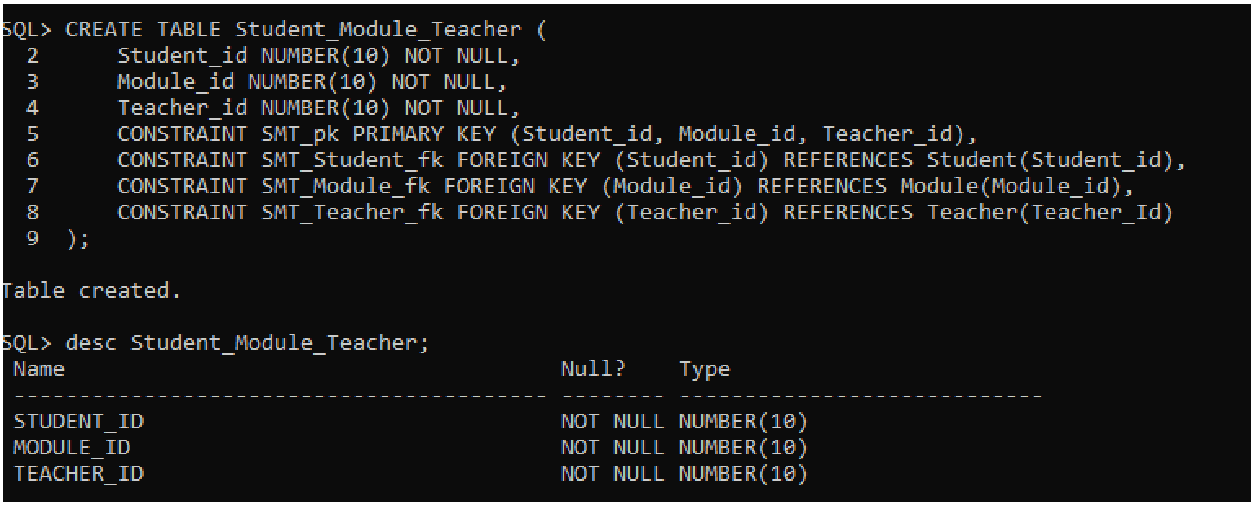
## 2.10 Create Teacher Table.



```
SQL> CREATE TABLE Teacher (
  2      Teacher_Id NUMBER(10) NOT NULL,
  3      Teacher_Name CHAR(100) NOT NULL,
  4      Teacher_Email CHAR(100) UNIQUE NOT NULL,
  5      CONSTRAINT Teacher_pk PRIMARY KEY (Teacher_Id)
  6  );

Table created.

SQL> desc Teacher;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TEACHER_ID                                NOT NULL NUMBER(10)
 TEACHER_NAME                              NOT NULL CHAR(100)
 TEACHER_EMAIL                             NOT NULL CHAR(100)
```

*Figure 14: Creating Teacher Table.*

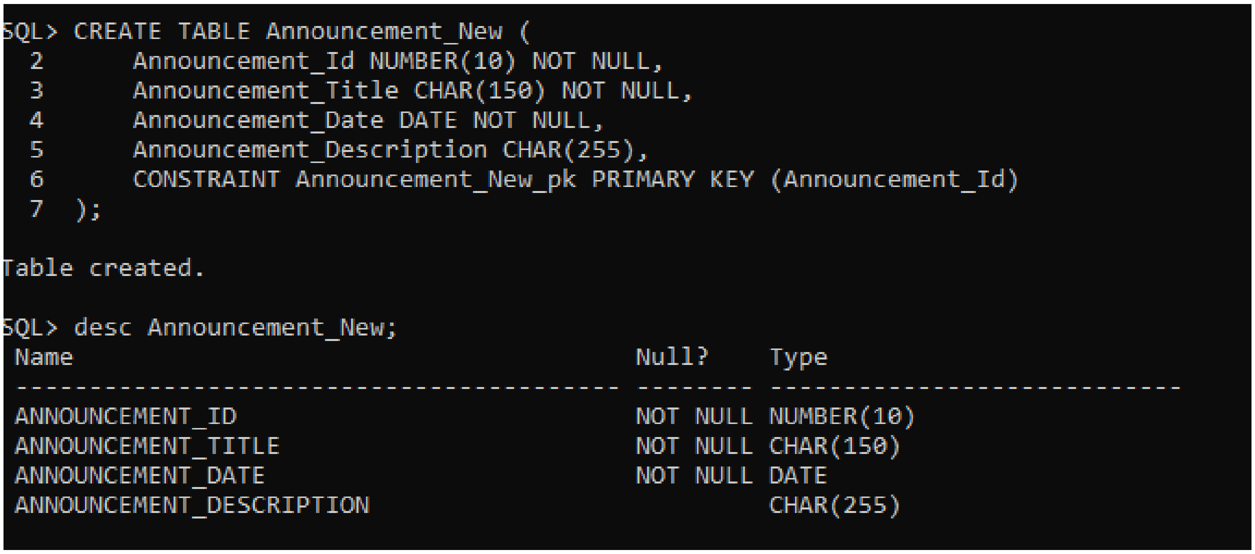
## 2.11 Create Student_Module_Teacher Table.



```
SQL> CREATE TABLE Student_Module_Teacher (
  2      Student_id NUMBER(10) NOT NULL,
  3      Module_id NUMBER(10) NOT NULL,
  4      Teacher_id NUMBER(10) NOT NULL,
  5      CONSTRAINT SMT_pk PRIMARY KEY (Student_id, Module_id, Teacher_id),
  6      CONSTRAINT SMT_Student_fk FOREIGN KEY (Student_id) REFERENCES Student(Student_id),
  7      CONSTRAINT SMT_Module_fk FOREIGN KEY (Module_id) REFERENCES Module(Module_id),
  8      CONSTRAINT SMT_Teacher_fk FOREIGN KEY (Teacher_id) REFERENCES Teacher(Teacher_Id)
  9  );

Table created.

SQL> desc Student_Module_Teacher;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL NUMBER(10)
 TEACHER_ID                                NOT NULL NUMBER(10)
```

*Figure 15: Create Student_Module_Teacher Table.*

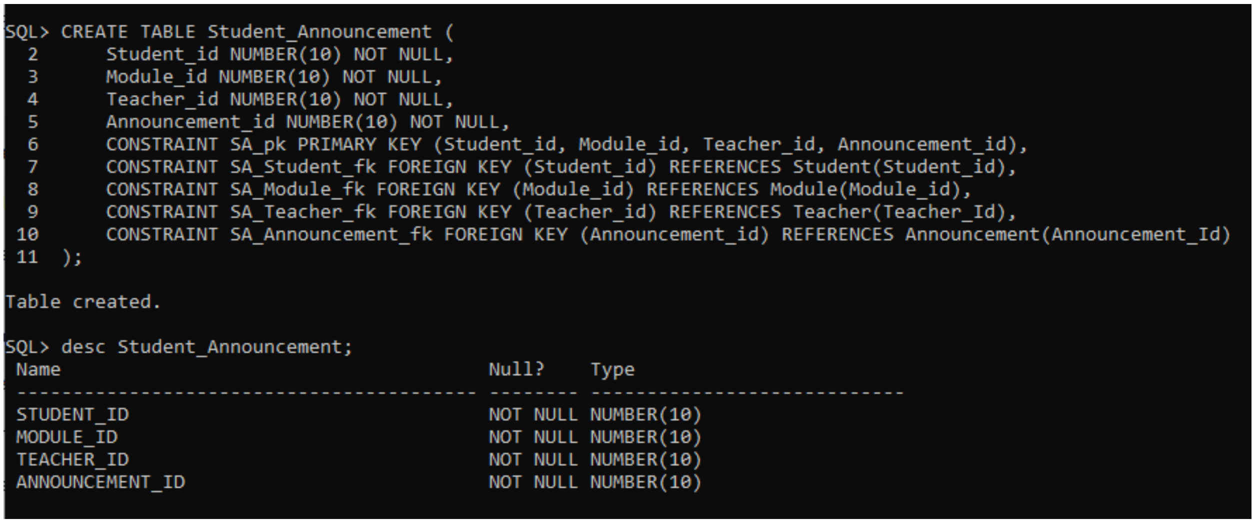
## 2.12 Create Announcement Table.



```
SQL> CREATE TABLE Announcement_New (
         Announcement_Id NUMBER(10) NOT NULL,
         Announcement_Title CHAR(150) NOT NULL,
         Announcement_Date DATE NOT NULL,
         Announcement_Description CHAR(255),
         CONSTRAINT Announcement_New_pk PRIMARY KEY (Announcement_Id)
     );

Table created.

SQL> desc Announcement_New;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ANNOUNCEMENT_ID                           NOT NULL NUMBER(10)
 ANNOUNCEMENT_TITLE                        NOT NULL CHAR(150)
 ANNOUNCEMENT_DATE                         NOT NULL DATE
 ANNOUNCEMENT_DESCRIPTION                           CHAR(255)
```

*Figure 16: Creating Announcement Table.*

## 2.13 Create Student_Announcement Table.

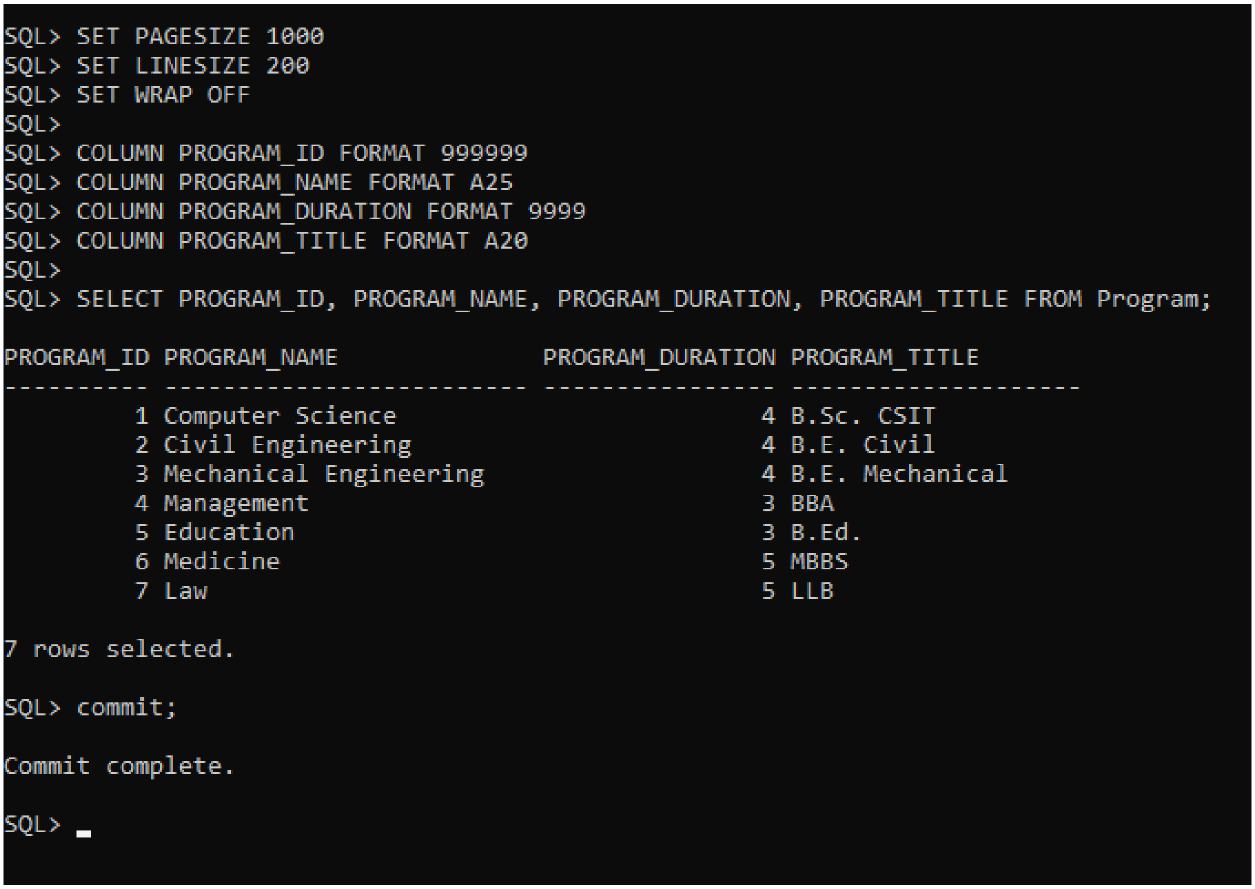


```
SQL> CREATE TABLE Student_Announcement (
         Student_id NUMBER(10) NOT NULL,
         Module_id NUMBER(10) NOT NULL,
         Teacher_id NUMBER(10) NOT NULL,
         Announcement_id NUMBER(10) NOT NULL,
         CONSTRAINT SA_pk PRIMARY KEY (Student_id, Module_id, Teacher_id, Announcement_id),
         CONSTRAINT SA_Student_fk FOREIGN KEY (Student_id) REFERENCES Student(Student_id),
         CONSTRAINT SA_Module_fk FOREIGN KEY (Module_id) REFERENCES Module(Module_id),
         CONSTRAINT SA_Teacher_fk FOREIGN KEY (Teacher_id) REFERENCES Teacher(Teacher_Id),
         CONSTRAINT SA_Announcement_fk FOREIGN KEY (Announcement_id) REFERENCES Announcement(Announcement_Id)
     );

Table created.

SQL> desc Student_Announcement;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL NUMBER(10)
 TEACHER_ID                                NOT NULL NUMBER(10)
 ANNOUNCEMENT_ID                           NOT NULL NUMBER(10)
```

*Figure 17: Creating Student_Announcement Table.*

## 3.Inserting the values

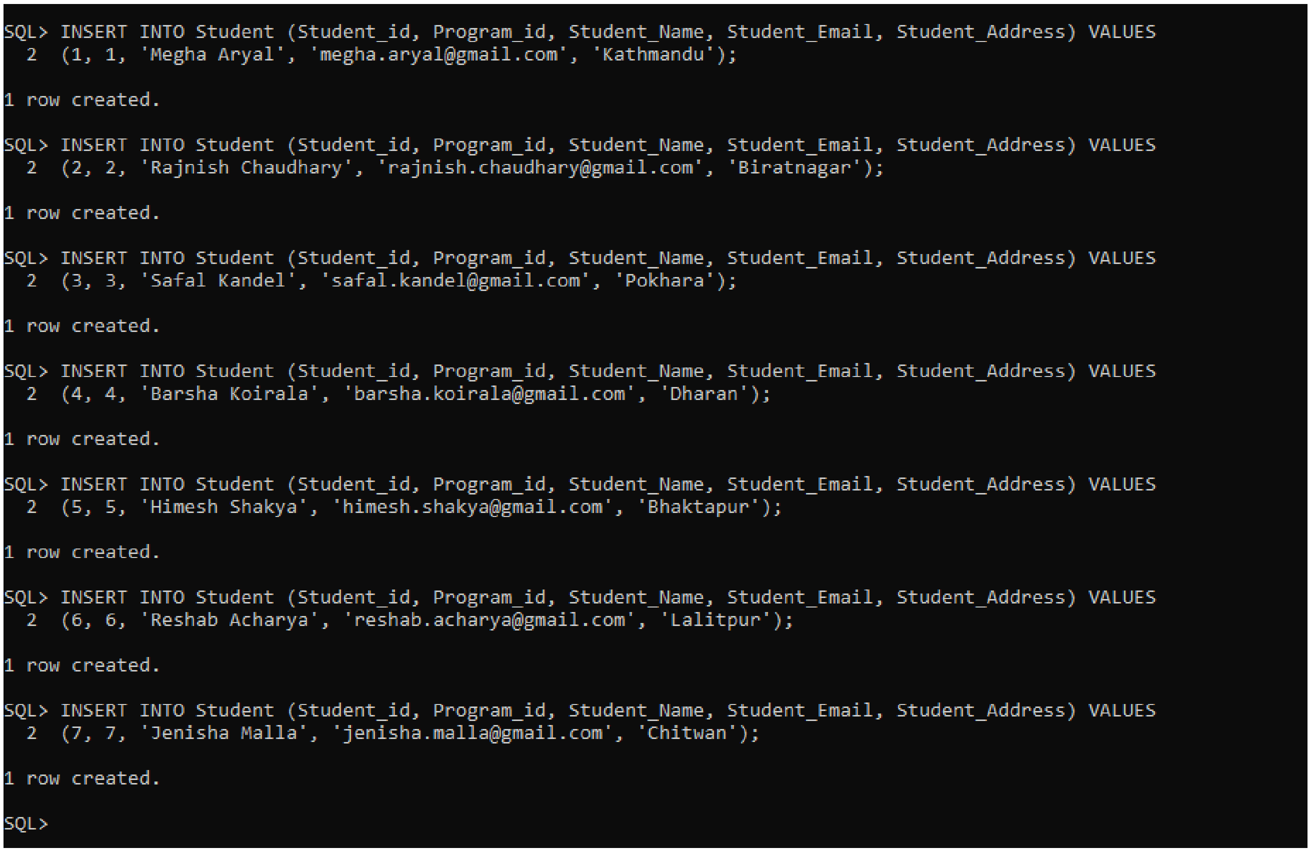
### 1)Inserting values in program table



```
SQL> SET PAGESIZE 1000
SQL> SET LINESIZE 200
SQL> SET WRAP OFF
SQL>
SQL> COLUMN PROGRAM_ID FORMAT 999999
SQL> COLUMN PROGRAM_NAME FORMAT A25
SQL> COLUMN PROGRAM_DURATION FORMAT 9999
SQL> COLUMN PROGRAM_TITLE FORMAT A20
SQL>
SQL> SELECT PROGRAM_ID, PROGRAM_NAME, PROGRAM_DURATION, PROGRAM_TITLE FROM Program;

PROGRAM_ID PROGRAM_NAME              PROGRAM_DURATION PROGRAM_TITLE
---------- ------------------------- ---------------- --------------------
         1 Computer Science                         4 B.Sc. CSIT
         2 Civil Engineering                        4 B.E. Civil
         3 Mechanical Engineering                   4 B.E. Mechanical
         4 Management                               3 BBA
         5 Education                                3 B.Ed.
         6 Medicine                                 5 MBBS
         7 Law                                      5 LLB

7 rows selected.

SQL> commit;

Commit complete.

SQL> _
```

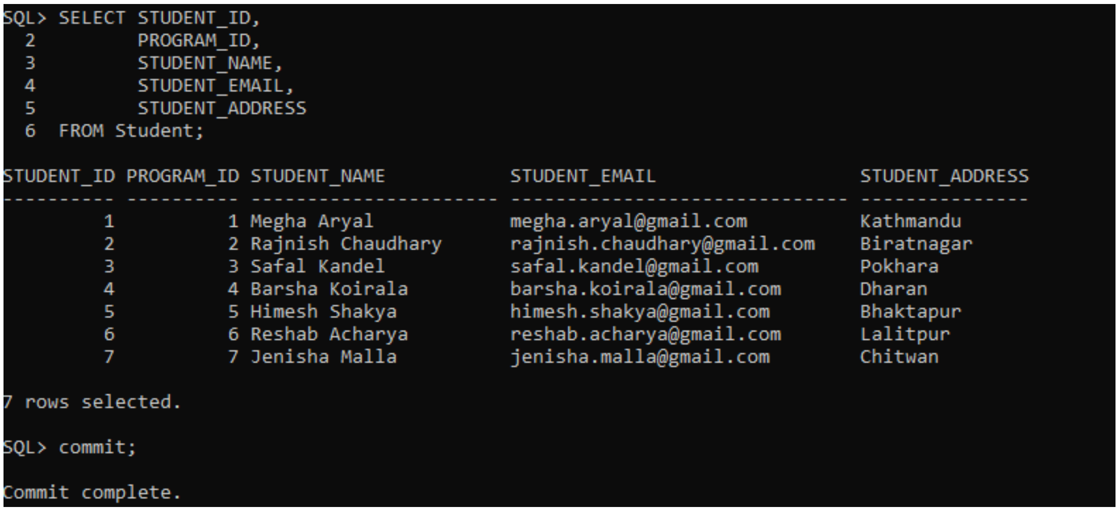
*Figure 18: Inserting in Program Table.*

## 2) Student table



```
SQL> INSERT INTO Student (Student_id, Program_id, Student_Name, Student_Email, Student_Address) VALUES
  2  (1, 1, 'Megha Aryal', 'megha.aryal@gmail.com', 'Kathmandu');

1 row created.

SQL> INSERT INTO Student (Student_id, Program_id, Student_Name, Student_Email, Student_Address) VALUES
  2  (2, 2, 'Rajnish Chaudhary', 'rajnish.chaudhary@gmail.com', 'Biratnagar');

1 row created.

SQL> INSERT INTO Student (Student_id, Program_id, Student_Name, Student_Email, Student_Address) VALUES
  2  (3, 3, 'Safal Kandel', 'safal.kandel@gmail.com', 'Pokhara');

1 row created.

SQL> INSERT INTO Student (Student_id, Program_id, Student_Name, Student_Email, Student_Address) VALUES
  2  (4, 4, 'Barsha Koirala', 'barsha.koirala@gmail.com', 'Dharan');

1 row created.

SQL> INSERT INTO Student (Student_id, Program_id, Student_Name, Student_Email, Student_Address) VALUES
  2  (5, 5, 'Himesh Shakya', 'himesh.shakya@gmail.com', 'Bhaktapur');

1 row created.

SQL> INSERT INTO Student (Student_id, Program_id, Student_Name, Student_Email, Student_Address) VALUES
  2  (6, 6, 'Reshab Acharya', 'reshab.acharya@gmail.com', 'Lalitpur');

1 row created.

SQL> INSERT INTO Student (Student_id, Program_id, Student_Name, Student_Email, Student_Address) VALUES
  2  (7, 7, 'Jenisha Malla', 'jenisha.malla@gmail.com', 'Chitwan');

1 row created.

SQL>
```

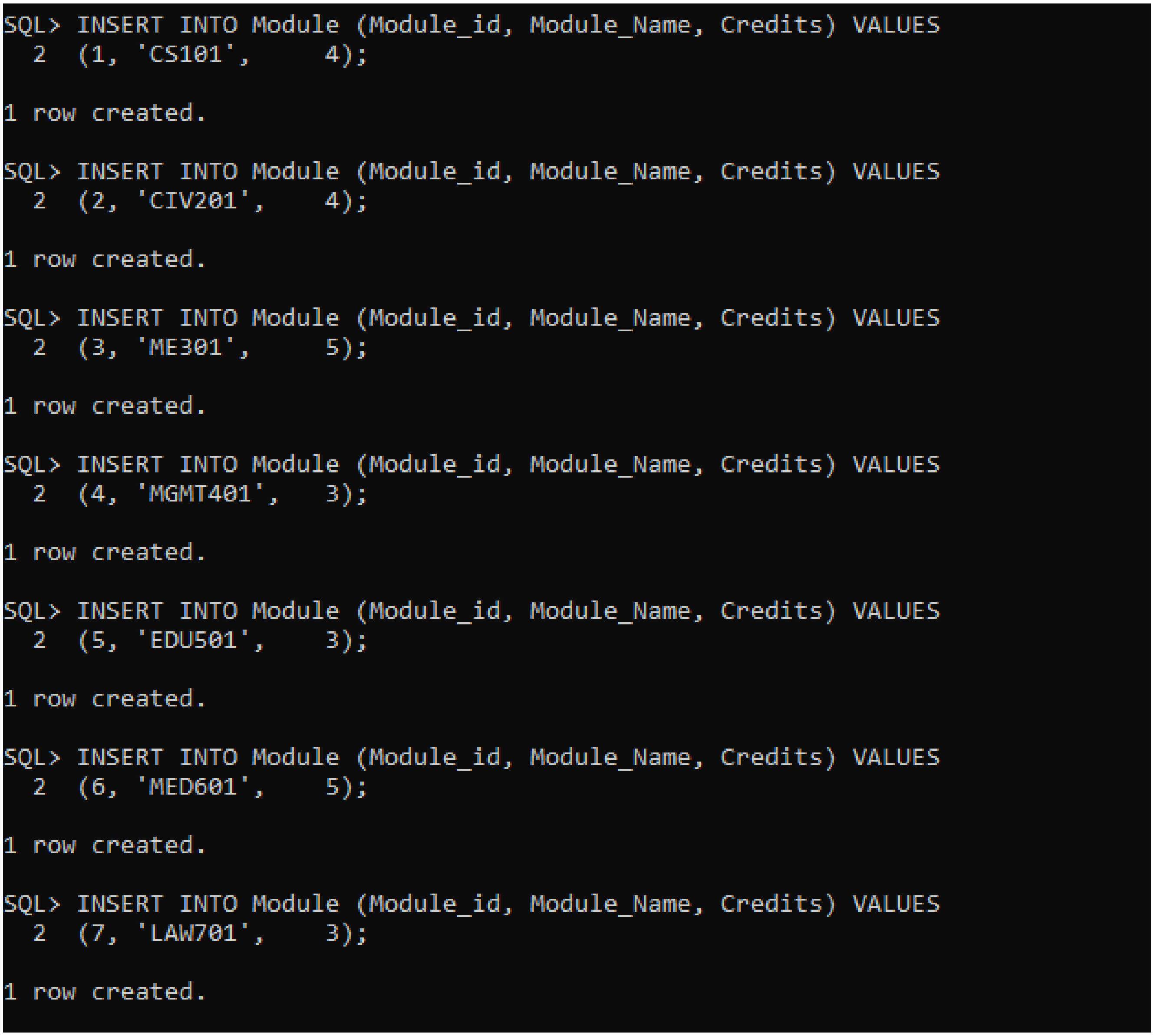
*Figure 19: Inserting in Student Table.*



```
SQL> SELECT STUDENT_ID,
            PROGRAM_ID,
            STUDENT_NAME,
            STUDENT_EMAIL,
            STUDENT_ADDRESS
     FROM Student;

STUDENT_ID PROGRAM_ID STUDENT_NAME          STUDENT_EMAIL                  STUDENT_ADDRESS
---------- ---------- --------------------- ------------------------------ ---------------
         1          1 Megha Aryal           megha.aryal@gmail.com          Kathmandu
         2          2 Rajnish Chaudhary     rajnish.chaudhary@gmail.com    Biratnagar
         3          3 Safal Kandel          safal.kandel@gmail.com         Pokhara
         4          4 Barsha Koirala        barsha.koirala@gmail.com       Dharan
         5          5 Himesh Shakya         himesh.shakya@gmail.com        Bhaktapur
         6          6 Reshab Acharya        reshab.acharya@gmail.com       Lalitpur
         7          7 Jenisha Malla         jenisha.malla@gmail.com        Chitwan

7 rows selected.

SQL> commit;

Commit complete.
```

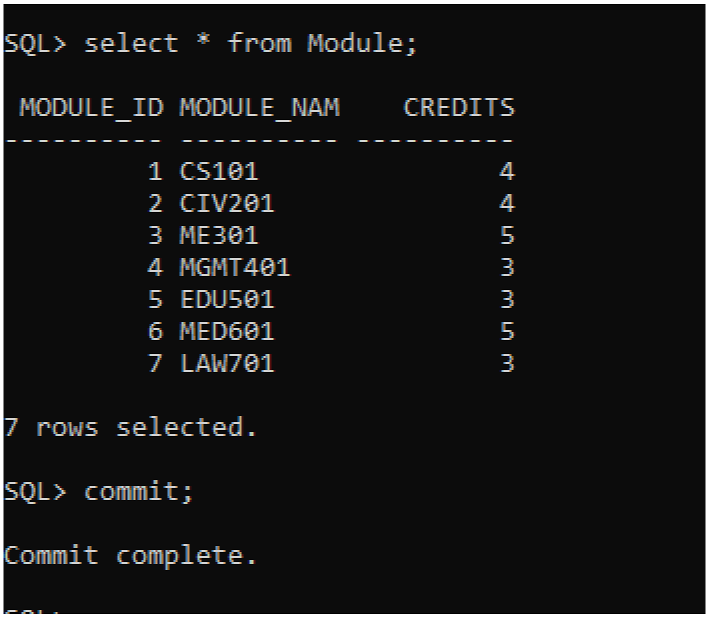
*Figure 20: Select from Student.*

### 3)module table



```
SQL> INSERT INTO Module (Module_id, Module_Name, Credits) VALUES
  2  (1, 'CS101',     4);

1 row created.

SQL> INSERT INTO Module (Module_id, Module_Name, Credits) VALUES
  2  (2, 'CIV201',    4);

1 row created.

SQL> INSERT INTO Module (Module_id, Module_Name, Credits) VALUES
  2  (3, 'ME301',     5);

1 row created.

SQL> INSERT INTO Module (Module_id, Module_Name, Credits) VALUES
  2  (4, 'MGMT401',   3);

1 row created.

SQL> INSERT INTO Module (Module_id, Module_Name, Credits) VALUES
  2  (5, 'EDU501',    3);

1 row created.

SQL> INSERT INTO Module (Module_id, Module_Name, Credits) VALUES
  2  (6, 'MED601',    5);

1 row created.

SQL> INSERT INTO Module (Module_id, Module_Name, Credits) VALUES
  2  (7, 'LAW701',    3);

1 row created.
```

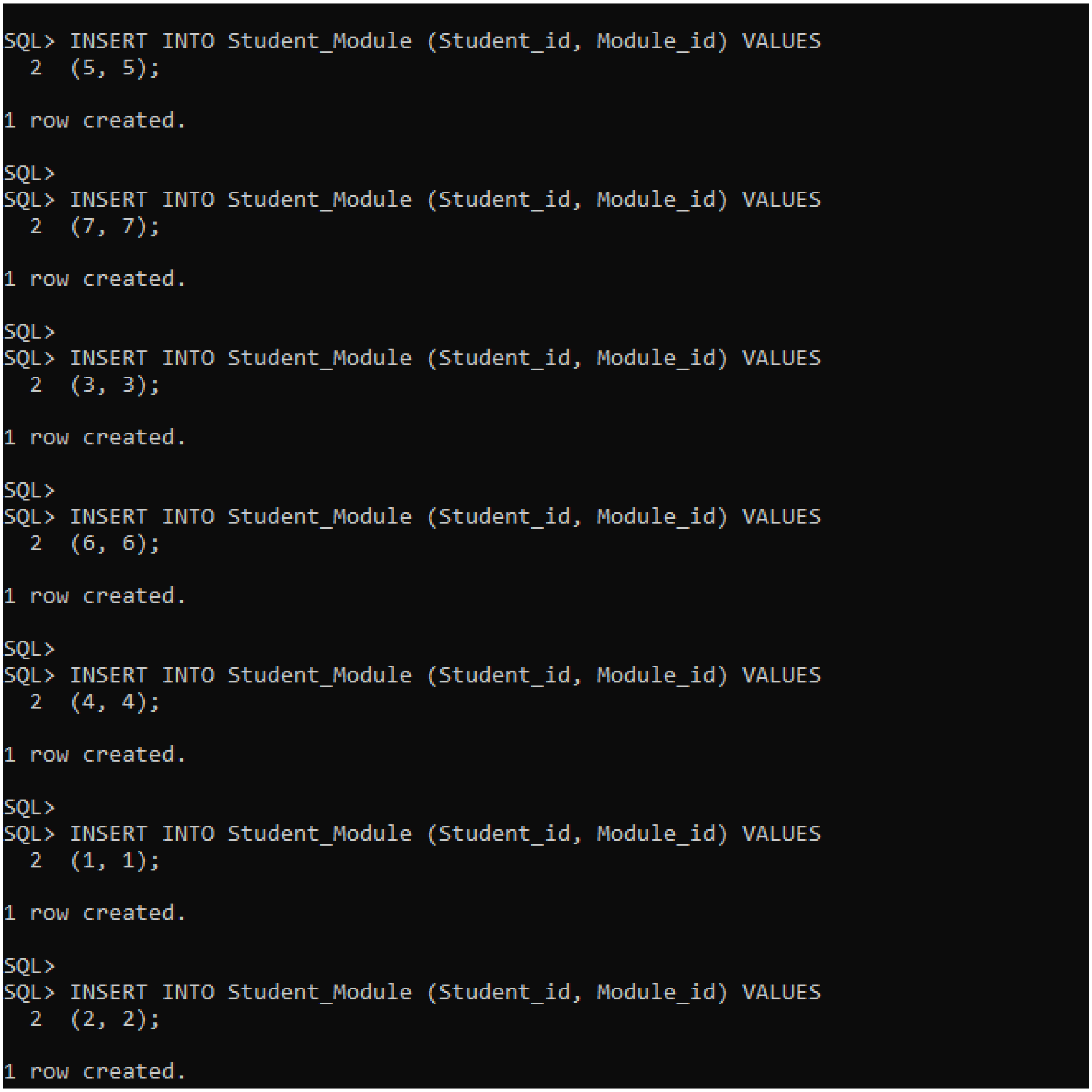
*Figure 21: Inserting in module Table.*



```
SQL> select * from Module;

 MODULE_ID MODULE_NAM    CREDITS
---------- ---------- ----------
         1 CS101               4
         2 CIV201              4
         3 ME301               5
         4 MGMT401             3
         5 EDU501              3
         6 MED601              5
         7 LAW701              3

7 rows selected.

SQL> commit;

Commit complete.
```

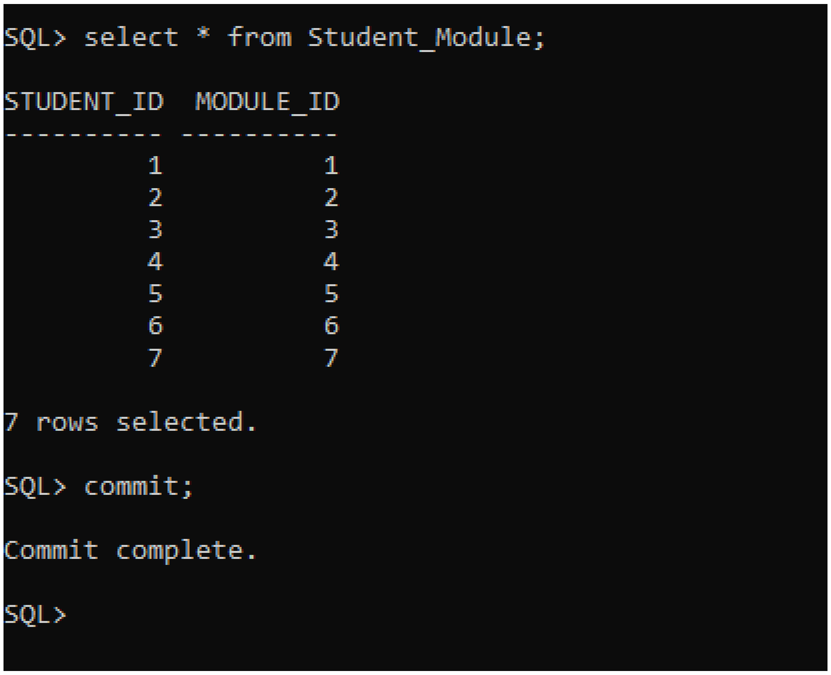
*Figure 22: Select from Module.*

### 4) Student Module



```
SQL> INSERT INTO Student_Module (Student_id, Module_id) VALUES
  2  (5, 5);

1 row created.

SQL>
SQL> INSERT INTO Student_Module (Student_id, Module_id) VALUES
  2  (7, 7);

1 row created.

SQL>
SQL> INSERT INTO Student_Module (Student_id, Module_id) VALUES
  2  (3, 3);

1 row created.

SQL>
SQL> INSERT INTO Student_Module (Student_id, Module_id) VALUES
  2  (6, 6);

1 row created.

SQL>
SQL> INSERT INTO Student_Module (Student_id, Module_id) VALUES
  2  (4, 4);

1 row created.

SQL>
SQL> INSERT INTO Student_Module (Student_id, Module_id) VALUES
  2  (1, 1);

1 row created.

SQL>
SQL> INSERT INTO Student_Module (Student_id, Module_id) VALUES
  2  (2, 2);

1 row created.
```

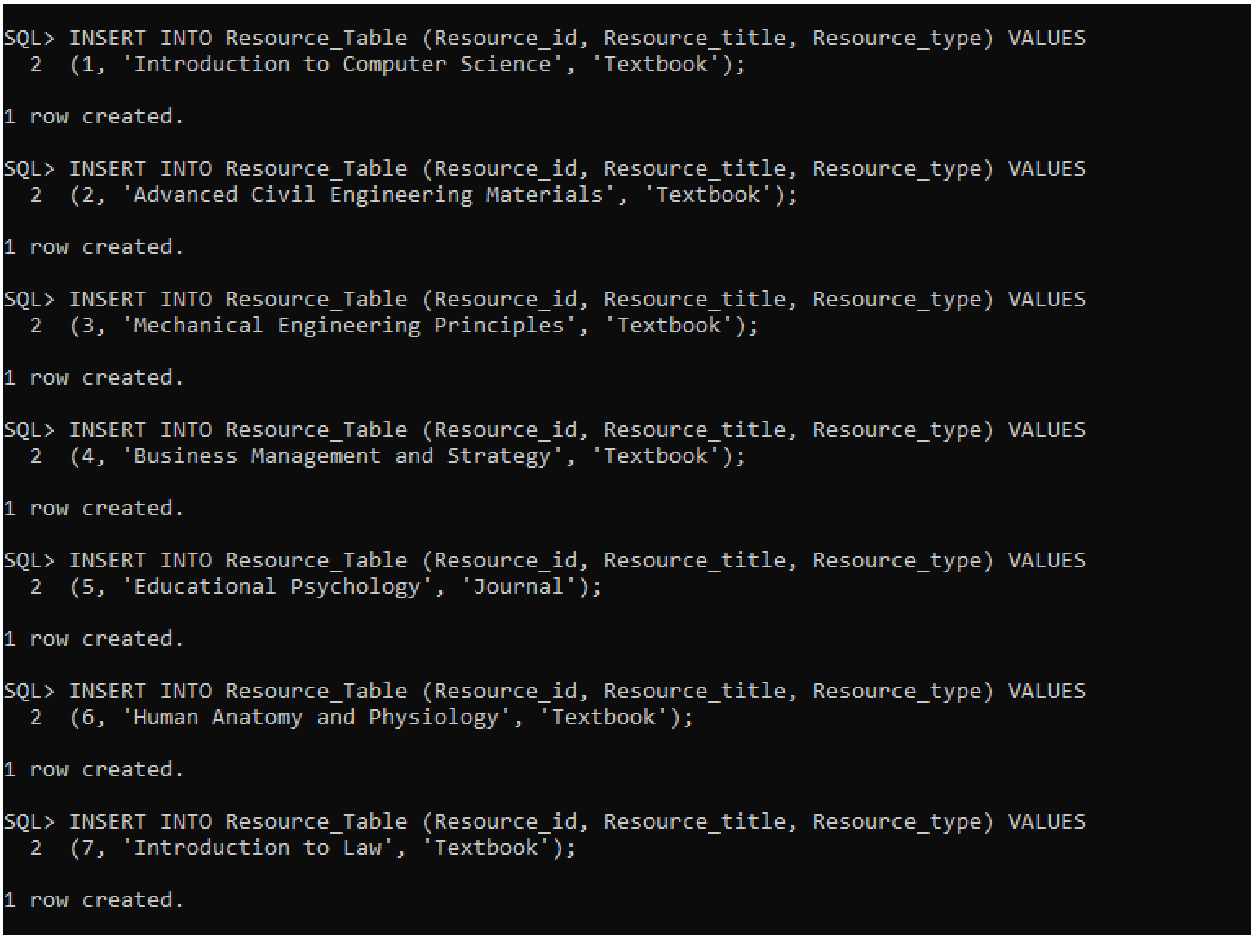
*Figure 23: Inserting into Student_Module Table.*



```
SQL> select * from Student_Module;

STUDENT_ID  MODULE_ID
---------- ----------
         1          1
         2          2
         3          3
         4          4
         5          5
         6          6
         7          7

7 rows selected.

SQL> commit;

Commit complete.

SQL>
```

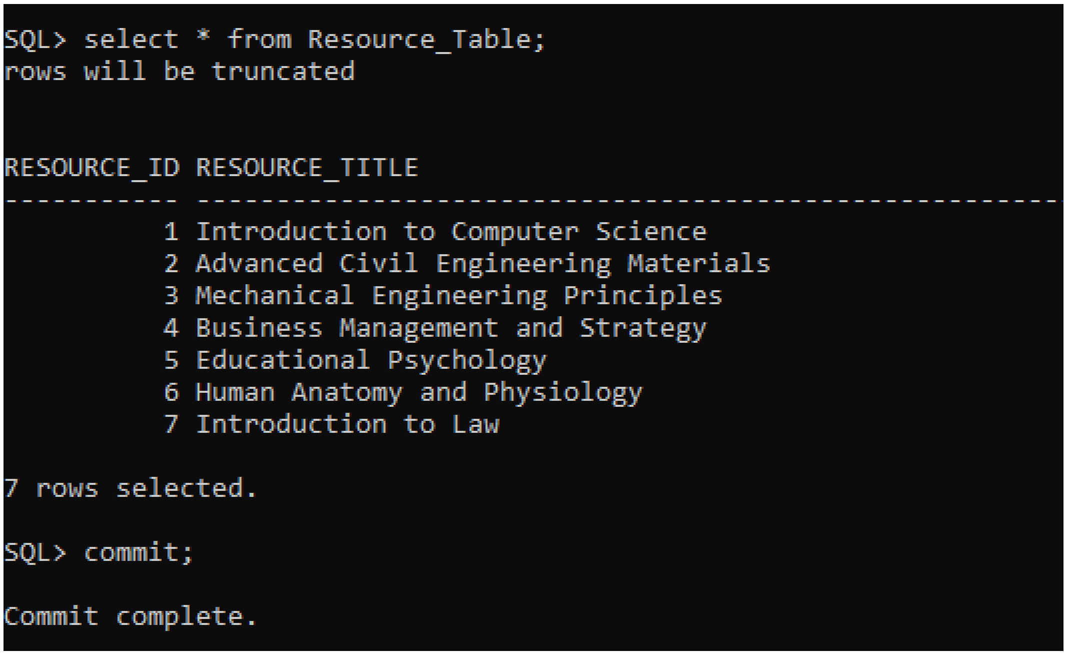
*Figure 24: Select Student_Module Table.*

## 5) Resource_Table



```
SQL> INSERT INTO Resource_Table (Resource_id, Resource_title, Resource_type) VALUES
  2  (1, 'Introduction to Computer Science', 'Textbook');

1 row created.

SQL> INSERT INTO Resource_Table (Resource_id, Resource_title, Resource_type) VALUES
  2  (2, 'Advanced Civil Engineering Materials', 'Textbook');

1 row created.

SQL> INSERT INTO Resource_Table (Resource_id, Resource_title, Resource_type) VALUES
  2  (3, 'Mechanical Engineering Principles', 'Textbook');

1 row created.

SQL> INSERT INTO Resource_Table (Resource_id, Resource_title, Resource_type) VALUES
  2  (4, 'Business Management and Strategy', 'Textbook');

1 row created.

SQL> INSERT INTO Resource_Table (Resource_id, Resource_title, Resource_type) VALUES
  2  (5, 'Educational Psychology', 'Journal');

1 row created.

SQL> INSERT INTO Resource_Table (Resource_id, Resource_title, Resource_type) VALUES
  2  (6, 'Human Anatomy and Physiology', 'Textbook');

1 row created.

SQL> INSERT INTO Resource_Table (Resource_id, Resource_title, Resource_type) VALUES
  2  (7, 'Introduction to Law', 'Textbook');

1 row created.
```

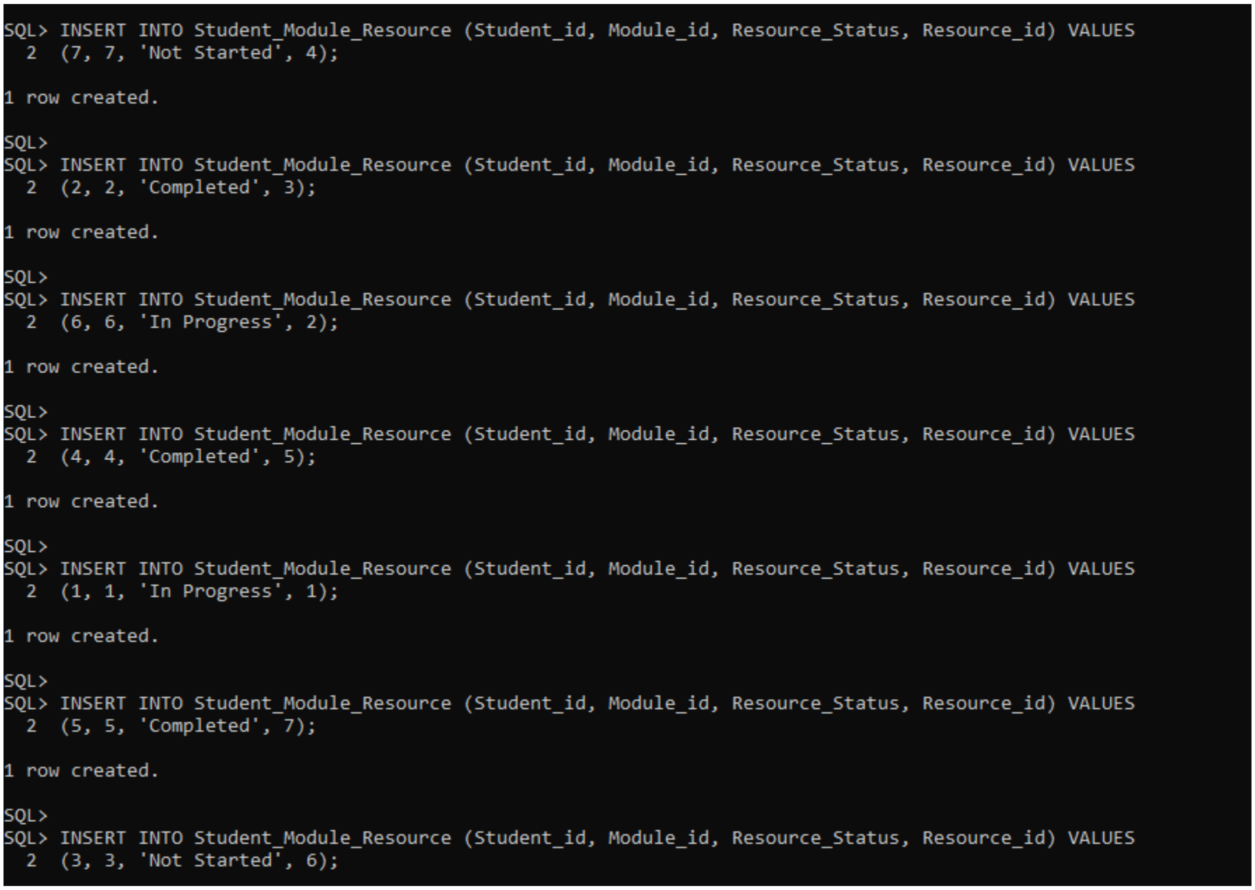
*Figure 25: Inserting into Resource Table.*



```
SQL> select * from Resource_Table;
rows will be truncated


RESOURCE_ID RESOURCE_TITLE
----------- -------------------------------------------------------
          1 Introduction to Computer Science
          2 Advanced Civil Engineering Materials
          3 Mechanical Engineering Principles
          4 Business Management and Strategy
          5 Educational Psychology
          6 Human Anatomy and Physiology
          7 Introduction to Law

7 rows selected.

SQL> commit;

Commit complete.
```

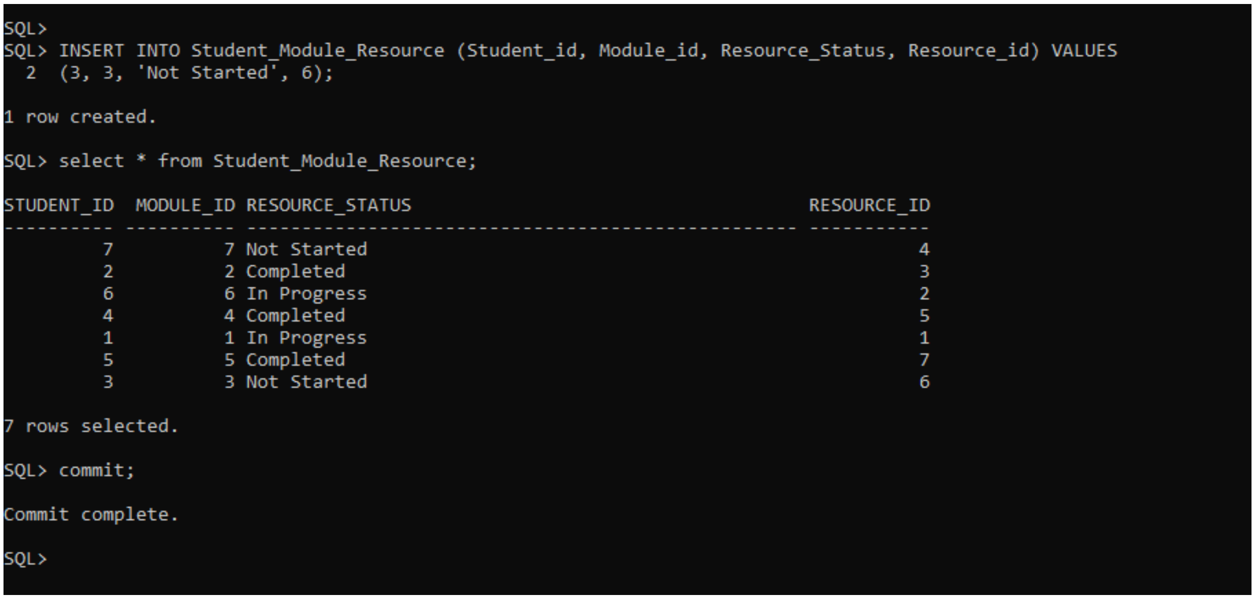
*Figure 26: Select from Resource Table.*

## 6)student module resource



```
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (7, 7, 'Not Started', 4);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (2, 2, 'Completed', 3);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (6, 6, 'In Progress', 2);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (4, 4, 'Completed', 5);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (1, 1, 'In Progress', 1);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (5, 5, 'Completed', 7);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (3, 3, 'Not Started', 6);
```

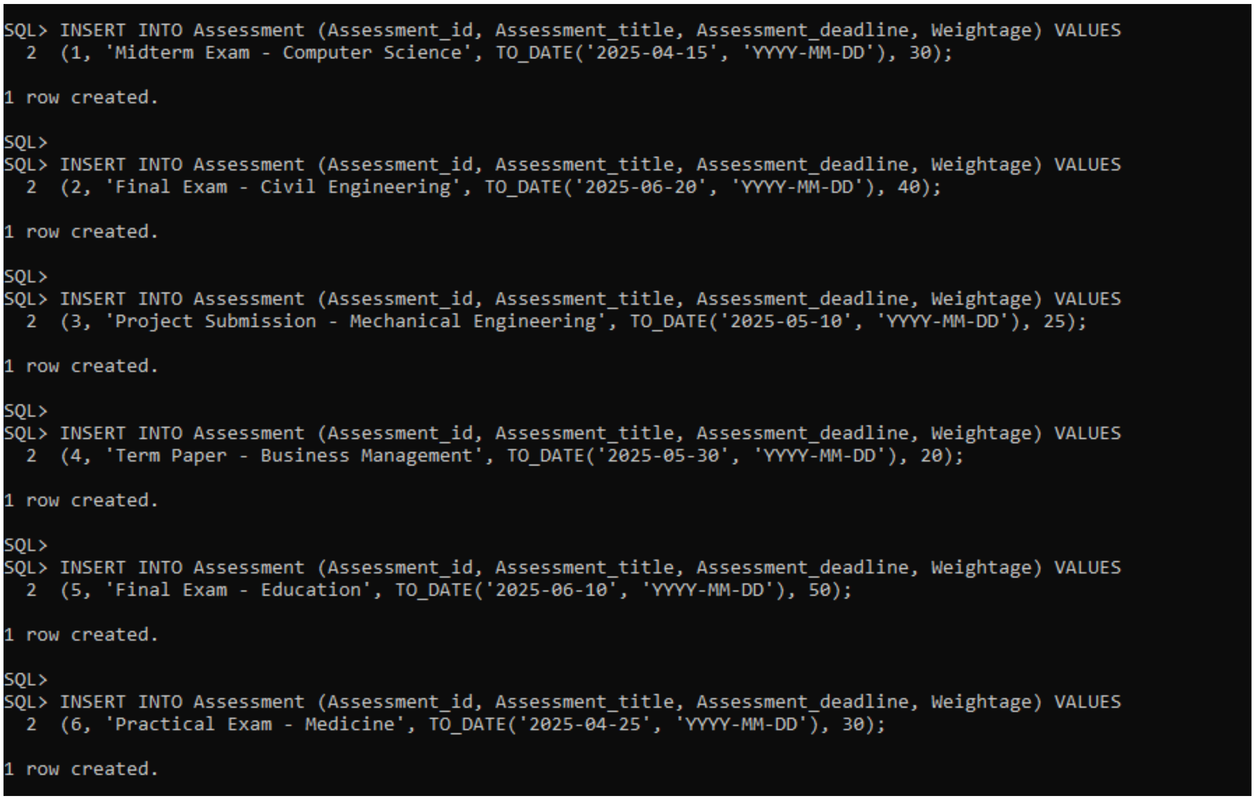
*Figure 27: Inserting into Student_Module_Resource Table.*



```
SQL>
SQL> INSERT INTO Student_Module_Resource (Student_id, Module_id, Resource_Status, Resource_id) VALUES
  2  (3, 3, 'Not Started', 6);

1 row created.

SQL> select * from Student_Module_Resource;

STUDENT_ID  MODULE_ID RESOURCE_STATUS                                    RESOURCE_ID
---------- ---------- -------------------------------------------------- -----------
         7          7 Not Started                                                  4
         2          2 Completed                                                    3
         6          6 In Progress                                                  2
         4          4 Completed                                                    5
         1          1 In Progress                                                  1
         5          5 Completed                                                    7
         3          3 Not Started                                                  6

7 rows selected.

SQL> commit;

Commit complete.

SQL>
```

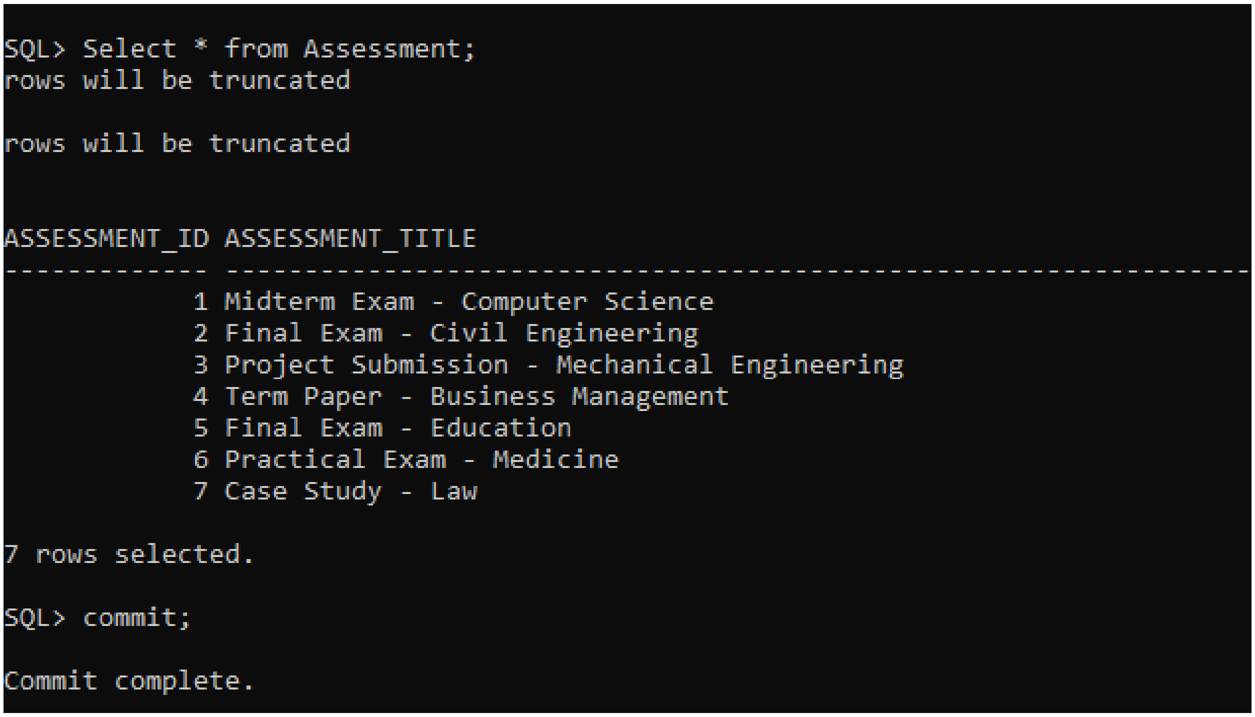
*Figure 28: Selecting From Student_Module_Resource Table.*

## 7)Assessment



```
SQL> INSERT INTO Assessment (Assessment_id, Assessment_title, Assessment_deadline, Weightage) VALUES
  2  (1, 'Midterm Exam - Computer Science', TO_DATE('2025-04-15', 'YYYY-MM-DD'), 30);

1 row created.

SQL>
SQL> INSERT INTO Assessment (Assessment_id, Assessment_title, Assessment_deadline, Weightage) VALUES
  2  (2, 'Final Exam - Civil Engineering', TO_DATE('2025-06-20', 'YYYY-MM-DD'), 40);

1 row created.

SQL>
SQL> INSERT INTO Assessment (Assessment_id, Assessment_title, Assessment_deadline, Weightage) VALUES
  2  (3, 'Project Submission - Mechanical Engineering', TO_DATE('2025-05-10', 'YYYY-MM-DD'), 25);

1 row created.

SQL>
SQL> INSERT INTO Assessment (Assessment_id, Assessment_title, Assessment_deadline, Weightage) VALUES
  2  (4, 'Term Paper - Business Management', TO_DATE('2025-05-30', 'YYYY-MM-DD'), 20);

1 row created.

SQL>
SQL> INSERT INTO Assessment (Assessment_id, Assessment_title, Assessment_deadline, Weightage) VALUES
  2  (5, 'Final Exam - Education', TO_DATE('2025-06-10', 'YYYY-MM-DD'), 50);

1 row created.

SQL>
SQL> INSERT INTO Assessment (Assessment_id, Assessment_title, Assessment_deadline, Weightage) VALUES
  2  (6, 'Practical Exam - Medicine', TO_DATE('2025-04-25', 'YYYY-MM-DD'), 30);

1 row created.
```

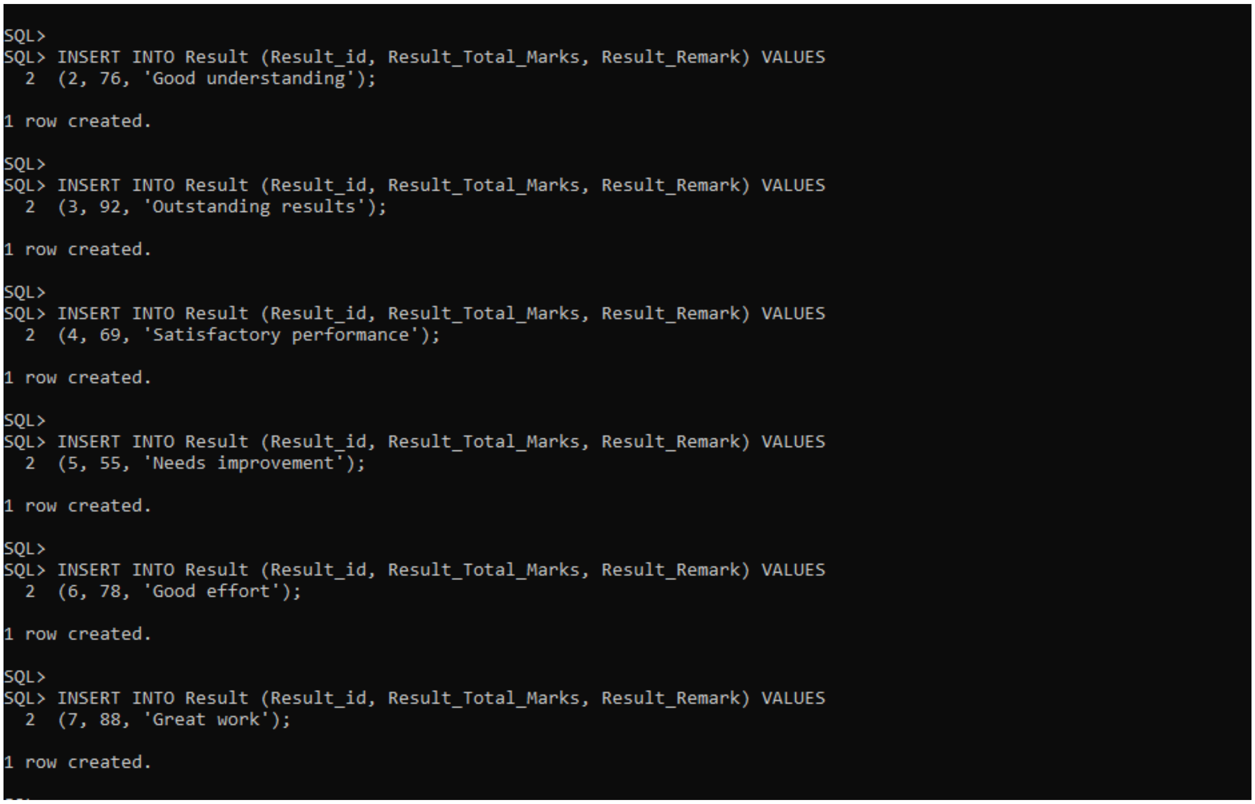
*Figure 29: Inserting into Assessment Table.*



```
SQL> Select * from Assessment;
rows will be truncated

rows will be truncated


ASSESSMENT_ID ASSESSMENT_TITLE
------------- -----------------------------------------------------------------
            1 Midterm Exam - Computer Science
            2 Final Exam - Civil Engineering
            3 Project Submission - Mechanical Engineering
            4 Term Paper - Business Management
            5 Final Exam - Education
            6 Practical Exam - Medicine
            7 Case Study - Law

7 rows selected.

SQL> commit;

Commit complete.
```

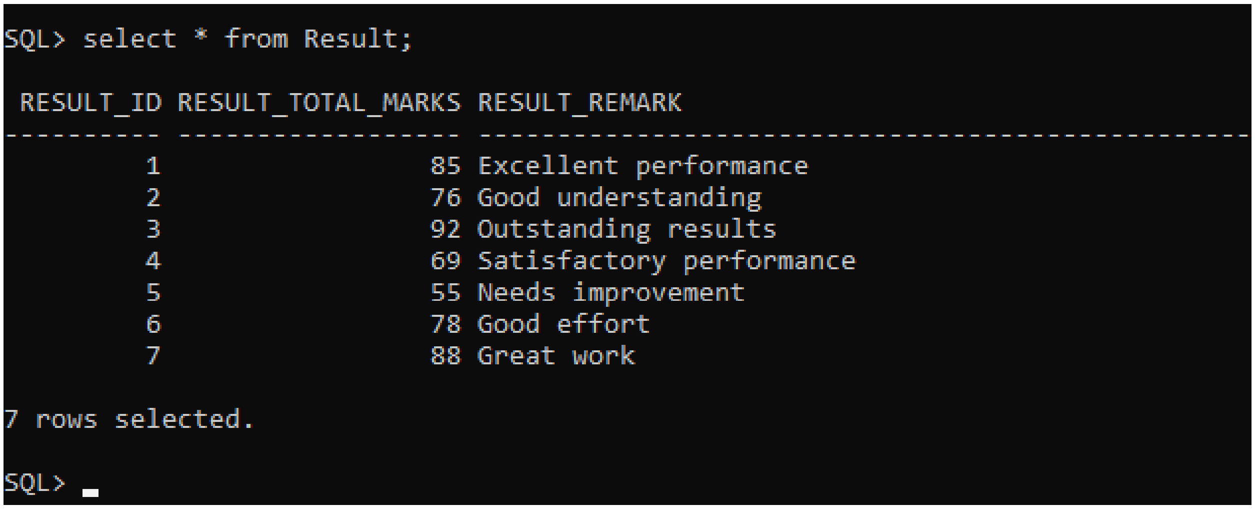
*Figure 30: Select from Assessment Table.*

## 8)Result



```
SQL>
SQL> INSERT INTO Result (Result_id, Result_Total_Marks, Result_Remark) VALUES
  2  (2, 76, 'Good understanding');

1 row created.

SQL>
SQL> INSERT INTO Result (Result_id, Result_Total_Marks, Result_Remark) VALUES
  2  (3, 92, 'Outstanding results');

1 row created.

SQL>
SQL> INSERT INTO Result (Result_id, Result_Total_Marks, Result_Remark) VALUES
  2  (4, 69, 'Satisfactory performance');

1 row created.

SQL>
SQL> INSERT INTO Result (Result_id, Result_Total_Marks, Result_Remark) VALUES
  2  (5, 55, 'Needs improvement');

1 row created.

SQL>
SQL> INSERT INTO Result (Result_id, Result_Total_Marks, Result_Remark) VALUES
  2  (6, 78, 'Good effort');

1 row created.

SQL>
SQL> INSERT INTO Result (Result_id, Result_Total_Marks, Result_Remark) VALUES
  2  (7, 88, 'Great work');

1 row created.
```

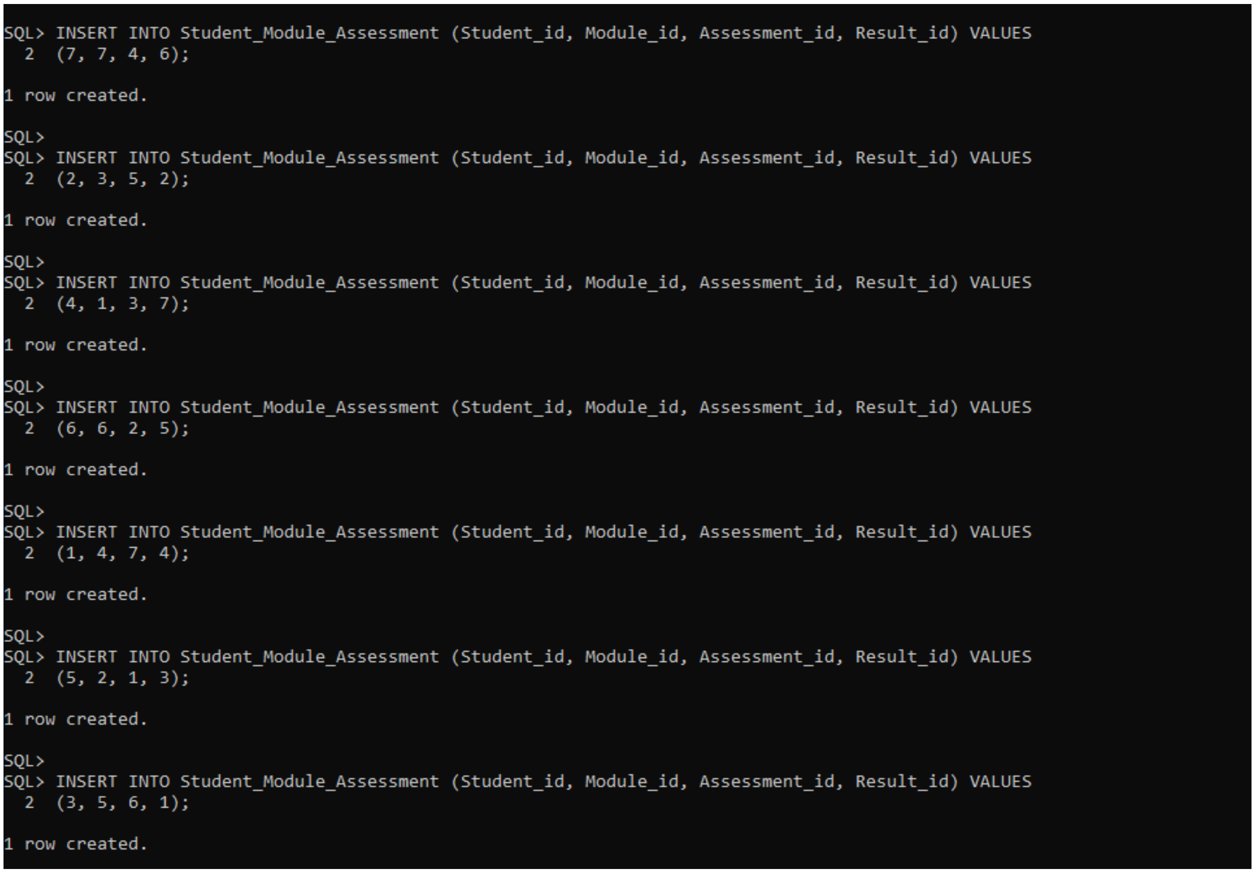
*Figure 31: Inserting into Result Table.*



```
SQL> select * from Result;

 RESULT_ID RESULT_TOTAL_MARKS RESULT_REMARK
---------- ------------------ --------------------------------------------------
         1                 85 Excellent performance
         2                 76 Good understanding
         3                 92 Outstanding results
         4                 69 Satisfactory performance
         5                 55 Needs improvement
         6                 78 Good effort
         7                 88 Great work

7 rows selected.

SQL> _
```

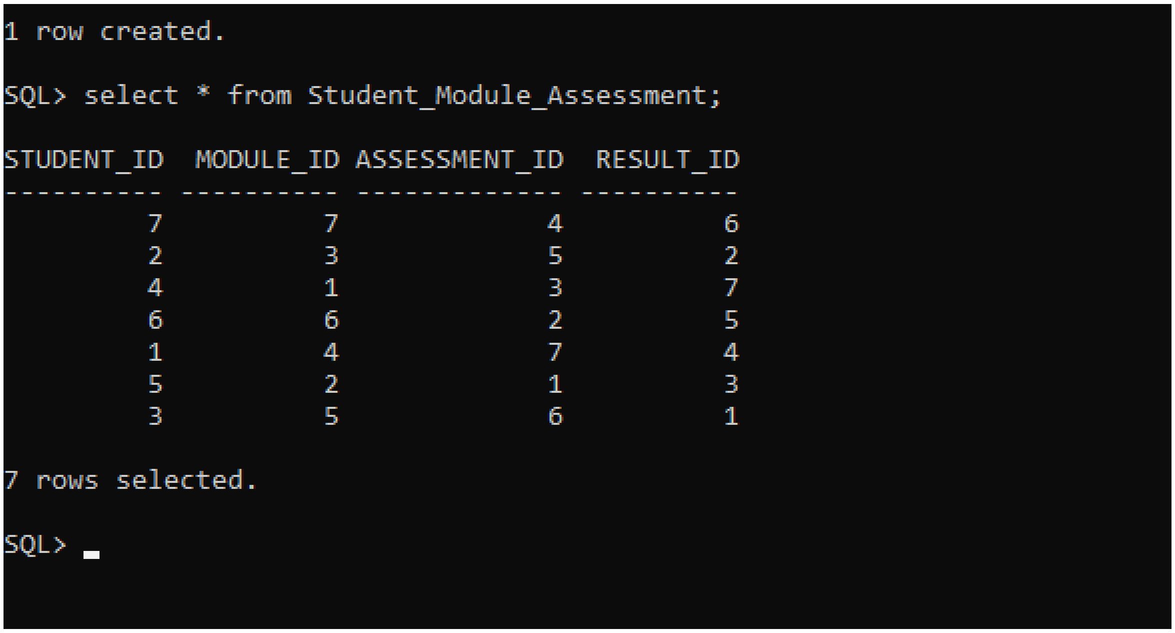
*Figure 32: Select from Results.*

### 9)student_ module _assessment



```
SQL> INSERT INTO Student_Module_Assessment (Student_id, Module_id, Assessment_id, Result_id) VALUES
  2  (7, 7, 4, 6);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Assessment (Student_id, Module_id, Assessment_id, Result_id) VALUES
  2  (2, 3, 5, 2);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Assessment (Student_id, Module_id, Assessment_id, Result_id) VALUES
  2  (4, 1, 3, 7);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Assessment (Student_id, Module_id, Assessment_id, Result_id) VALUES
  2  (6, 6, 2, 5);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Assessment (Student_id, Module_id, Assessment_id, Result_id) VALUES
  2  (1, 4, 7, 4);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Assessment (Student_id, Module_id, Assessment_id, Result_id) VALUES
  2  (5, 2, 1, 3);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Assessment (Student_id, Module_id, Assessment_id, Result_id) VALUES
  2  (3, 5, 6, 1);

1 row created.
```

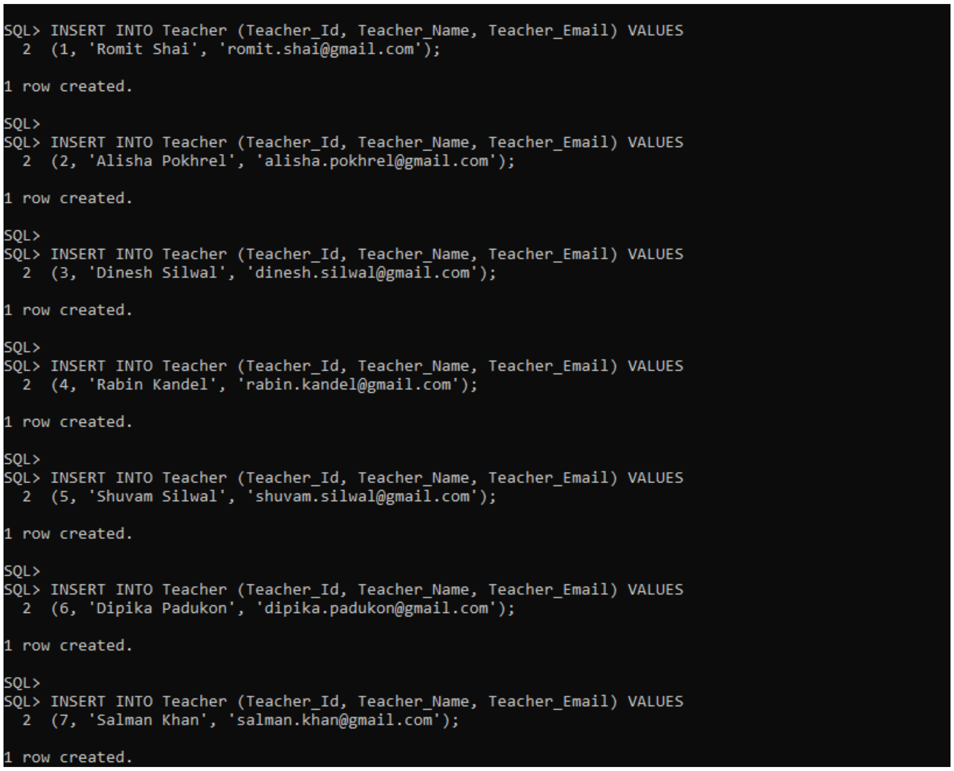
*Figure 33: Inserting into Student_Module_Assessment.*



```
1 row created.

SQL> select * from Student_Module_Assessment;

STUDENT_ID  MODULE_ID ASSESSMENT_ID  RESULT_ID
---------- ---------- ------------- ----------
         7          7             4          6
         2          3             5          2
         4          1             3          7
         6          6             2          5
         1          4             7          4
         5          2             1          3
         3          5             6          1

7 rows selected.

SQL>
```

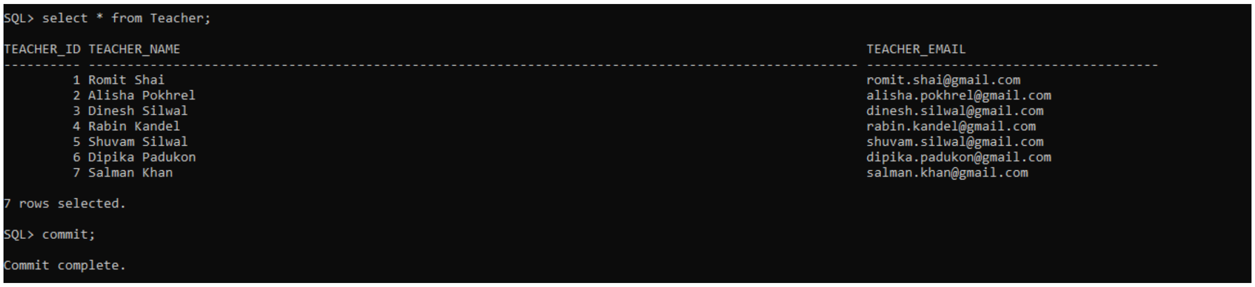
*Figure 34: Selecting From Student_Module_Assessment.*

## 10)Teacher



```
SQL> INSERT INTO Teacher (Teacher_Id, Teacher_Name, Teacher_Email) VALUES
  2  (1, 'Romit Shai', 'romit.shai@gmail.com');

1 row created.

SQL>
SQL> INSERT INTO Teacher (Teacher_Id, Teacher_Name, Teacher_Email) VALUES
  2  (2, 'Alisha Pokhrel', 'alisha.pokhrel@gmail.com');

1 row created.

SQL>
SQL> INSERT INTO Teacher (Teacher_Id, Teacher_Name, Teacher_Email) VALUES
  2  (3, 'Dinesh Silwal', 'dinesh.silwal@gmail.com');

1 row created.

SQL>
SQL> INSERT INTO Teacher (Teacher_Id, Teacher_Name, Teacher_Email) VALUES
  2  (4, 'Rabin Kandel', 'rabin.kandel@gmail.com');

1 row created.

SQL>
SQL> INSERT INTO Teacher (Teacher_Id, Teacher_Name, Teacher_Email) VALUES
  2  (5, 'Shuvam Silwal', 'shuvam.silwal@gmail.com');

1 row created.

SQL>
SQL> INSERT INTO Teacher (Teacher_Id, Teacher_Name, Teacher_Email) VALUES
  2  (6, 'Dipika Padukon', 'dipika.padukon@gmail.com');

1 row created.

SQL>
SQL> INSERT INTO Teacher (Teacher_Id, Teacher_Name, Teacher_Email) VALUES
  2  (7, 'Salman Khan', 'salman.khan@gmail.com');

1 row created.
```

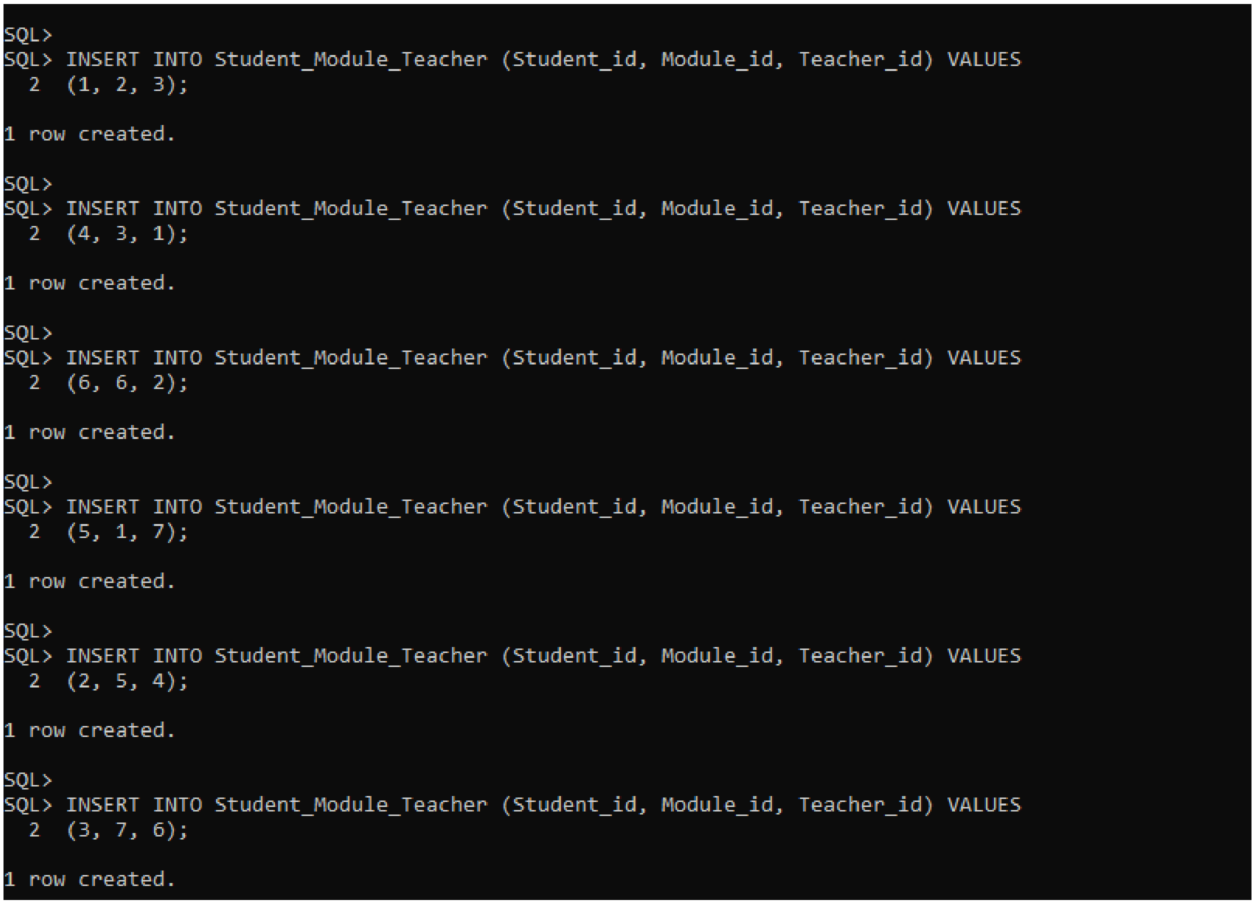
*Figure 35: Insert into Teacher.*



```
SQL> select * from Teacher;

TEACHER_ID TEACHER_NAME                                                                                         TEACHER_EMAIL
---------- ---------------------------------------------------------------------------------------------------- --------------------------------------
         1 Romit Shai                                                                                           romit.shai@gmail.com
         2 Alisha Pokhrel                                                                                       alisha.pokhrel@gmail.com
         3 Dinesh Silwal                                                                                        dinesh.silwal@gmail.com
         4 Rabin Kandel                                                                                         rabin.kandel@gmail.com
         5 Shuvam Silwal                                                                                        shuvam.silwal@gmail.com
         6 Dipika Padukon                                                                                       dipika.padukon@gmail.com
         7 Salman Khan                                                                                          salman.khan@gmail.com

7 rows selected.

SQL> commit;

Commit complete.
```

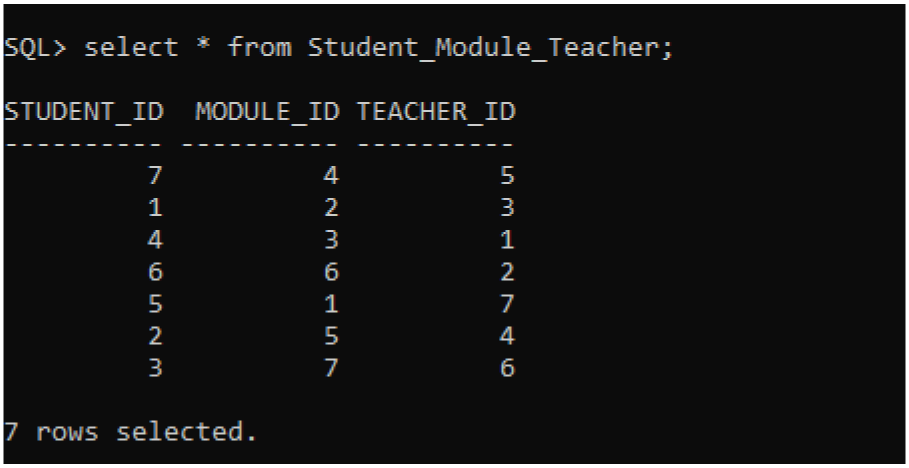
*Figure 36: Select from Teacher.*

### 11) student module teacher



```
SQL>
SQL> INSERT INTO Student_Module_Teacher (Student_id, Module_id, Teacher_id) VALUES
  2  (1, 2, 3);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Teacher (Student_id, Module_id, Teacher_id) VALUES
  2  (4, 3, 1);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Teacher (Student_id, Module_id, Teacher_id) VALUES
  2  (6, 6, 2);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Teacher (Student_id, Module_id, Teacher_id) VALUES
  2  (5, 1, 7);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Teacher (Student_id, Module_id, Teacher_id) VALUES
  2  (2, 5, 4);

1 row created.

SQL>
SQL> INSERT INTO Student_Module_Teacher (Student_id, Module_id, Teacher_id) VALUES
  2  (3, 7, 6);

1 row created.
```

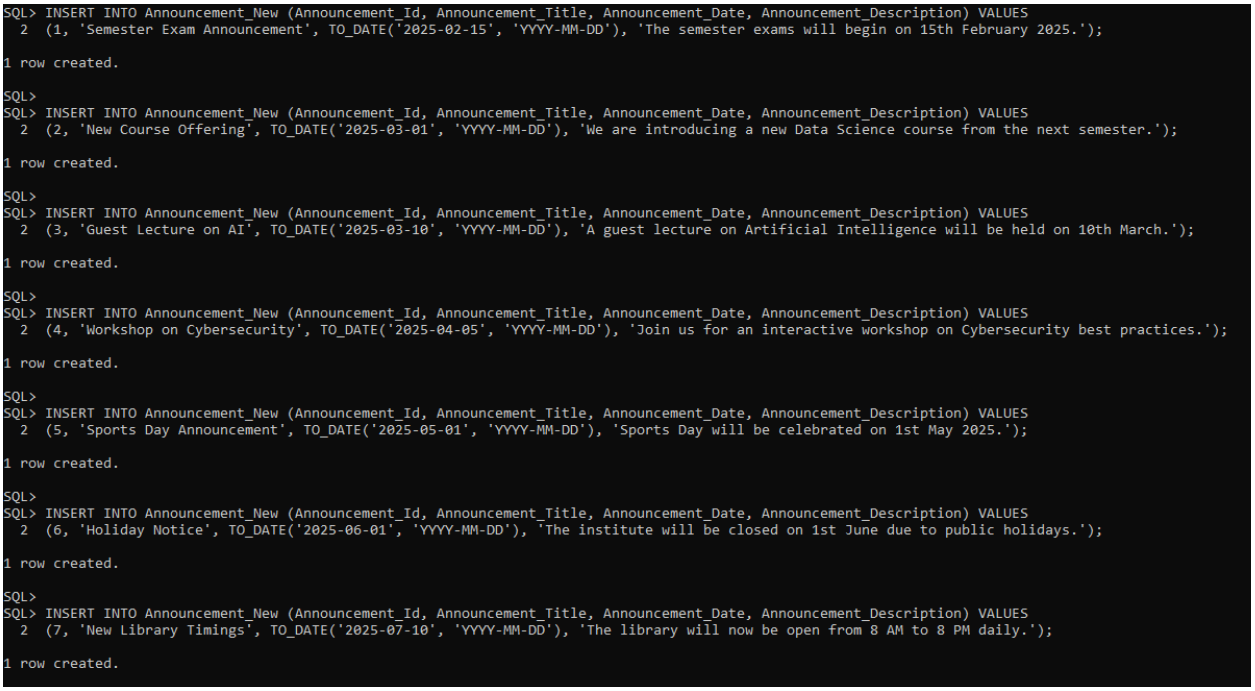
*Figure 37: Insert into Student_Module_Teacher Table.*



```
SQL> select * from Student_Module_Teacher;

STUDENT_ID  MODULE_ID TEACHER_ID
---------- ---------- ----------
         7          4          5
         1          2          3
         4          3          1
         6          6          2
         5          1          7
         2          5          4
         3          7          6

7 rows selected.
```

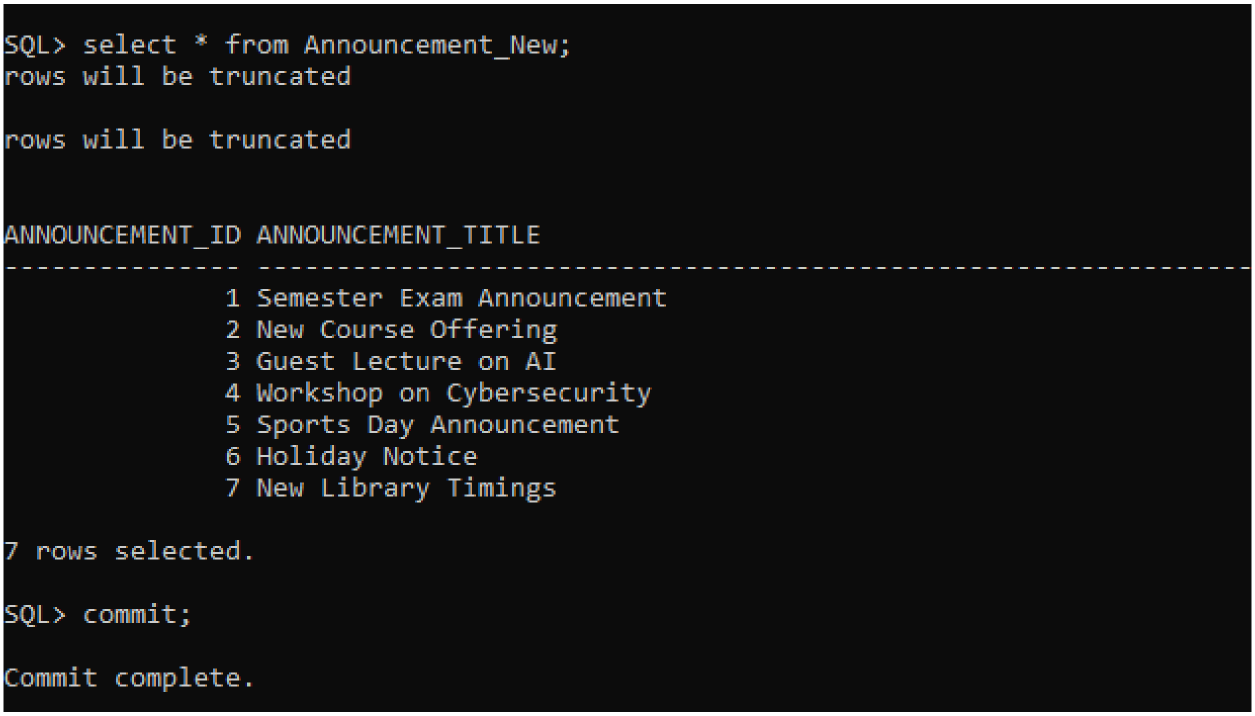
*Figure 38:Select from student module teacher Table.*

## 12)Announcement New



```
SQL> INSERT INTO Announcement_New (Announcement_Id, Announcement_Title, Announcement_Date, Announcement_Description) VALUES
  2  (1, 'Semester Exam Announcement', TO_DATE('2025-02-15', 'YYYY-MM-DD'), 'The semester exams will begin on 15th February 2025.');

1 row created.

SQL>
SQL> INSERT INTO Announcement_New (Announcement_Id, Announcement_Title, Announcement_Date, Announcement_Description) VALUES
  2  (2, 'New Course Offering', TO_DATE('2025-03-01', 'YYYY-MM-DD'), 'We are introducing a new Data Science course from the next semester.');

1 row created.

SQL>
SQL> INSERT INTO Announcement_New (Announcement_Id, Announcement_Title, Announcement_Date, Announcement_Description) VALUES
  2  (3, 'Guest Lecture on AI', TO_DATE('2025-03-10', 'YYYY-MM-DD'), 'A guest lecture on Artificial Intelligence will be held on 10th March.');

1 row created.

SQL>
SQL> INSERT INTO Announcement_New (Announcement_Id, Announcement_Title, Announcement_Date, Announcement_Description) VALUES
  2  (4, 'Workshop on Cybersecurity', TO_DATE('2025-04-05', 'YYYY-MM-DD'), 'Join us for an interactive workshop on Cybersecurity best practices.');

1 row created.

SQL>
SQL> INSERT INTO Announcement_New (Announcement_Id, Announcement_Title, Announcement_Date, Announcement_Description) VALUES
  2  (5, 'Sports Day Announcement', TO_DATE('2025-05-01', 'YYYY-MM-DD'), 'Sports Day will be celebrated on 1st May 2025.');

1 row created.

SQL>
SQL> INSERT INTO Announcement_New (Announcement_Id, Announcement_Title, Announcement_Date, Announcement_Description) VALUES
  2  (6, 'Holiday Notice', TO_DATE('2025-06-01', 'YYYY-MM-DD'), 'The institute will be closed on 1st June due to public holidays.');

1 row created.

SQL>
SQL> INSERT INTO Announcement_New (Announcement_Id, Announcement_Title, Announcement_Date, Announcement_Description) VALUES
  2  (7, 'New Library Timings', TO_DATE('2025-07-10', 'YYYY-MM-DD'), 'The library will now be open from 8 AM to 8 PM daily.');

1 row created.
```

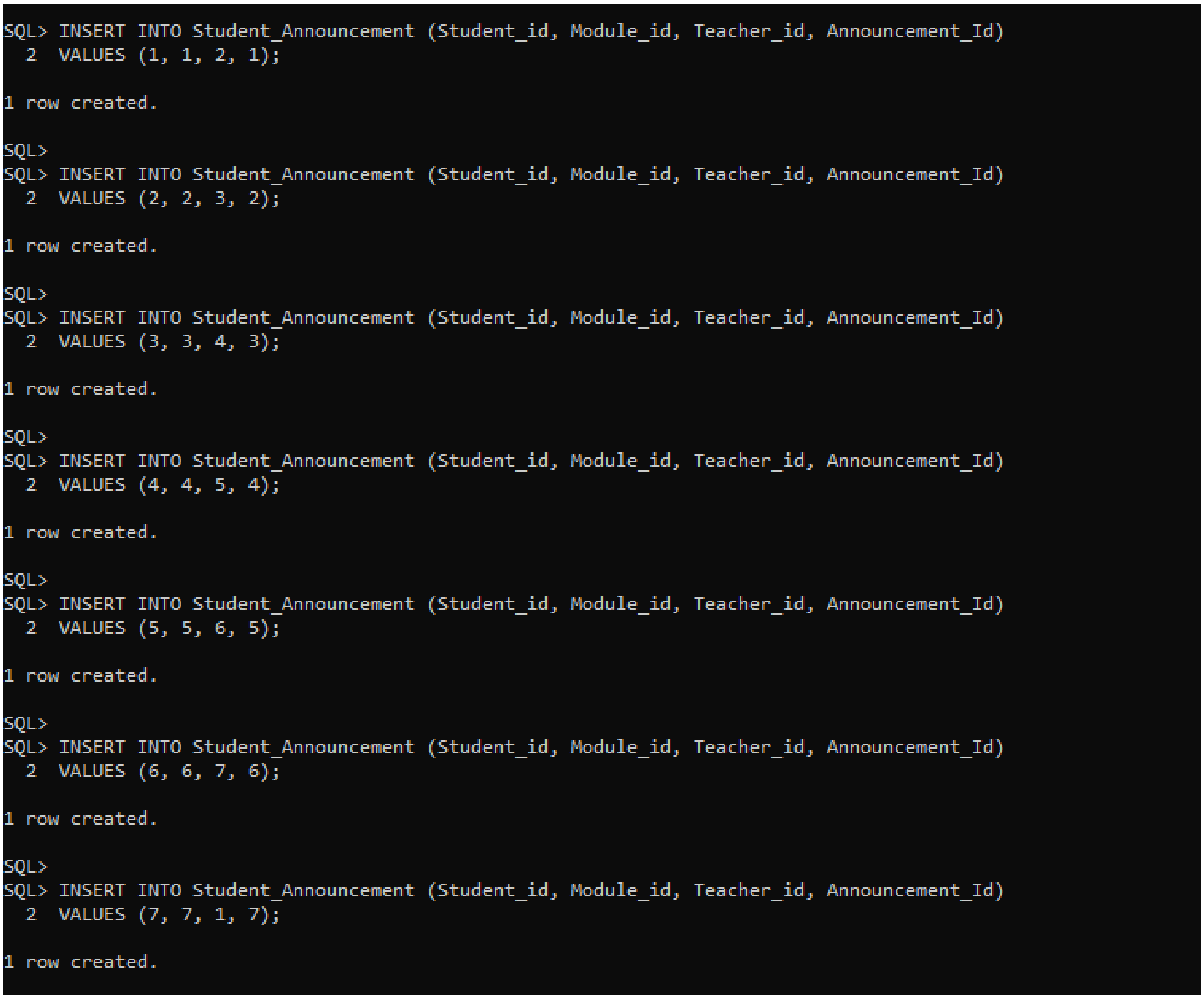
*Figure 39: Inserting into Announcement Table.*



```
SQL> select * from Announcement_New;
rows will be truncated

rows will be truncated


ANNOUNCEMENT_ID ANNOUNCEMENT_TITLE
--------------- ----------------------------------------------------------------
              1 Semester Exam Announcement
              2 New Course Offering
              3 Guest Lecture on AI
              4 Workshop on Cybersecurity
              5 Sports Day Announcement
              6 Holiday Notice
              7 New Library Timings

7 rows selected.

SQL> commit;

Commit complete.
```

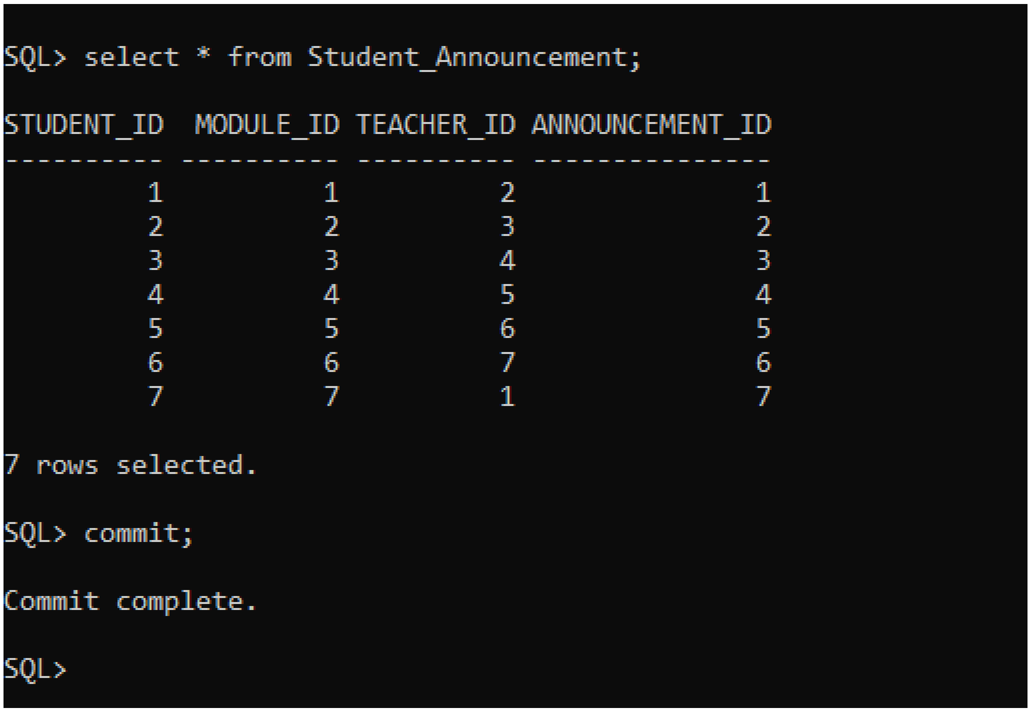
*Figure 40: Selecting From Announcement table.*

## 13)student announcement



```
SQL> INSERT INTO Student_Announcement (Student_id, Module_id, Teacher_id, Announcement_Id)
  2  VALUES (1, 1, 2, 1);

1 row created.

SQL>
SQL> INSERT INTO Student_Announcement (Student_id, Module_id, Teacher_id, Announcement_Id)
  2  VALUES (2, 2, 3, 2);

1 row created.

SQL>
SQL> INSERT INTO Student_Announcement (Student_id, Module_id, Teacher_id, Announcement_Id)
  2  VALUES (3, 3, 4, 3);

1 row created.

SQL>
SQL> INSERT INTO Student_Announcement (Student_id, Module_id, Teacher_id, Announcement_Id)
  2  VALUES (4, 4, 5, 4);

1 row created.

SQL>
SQL> INSERT INTO Student_Announcement (Student_id, Module_id, Teacher_id, Announcement_Id)
  2  VALUES (5, 5, 6, 5);

1 row created.

SQL>
SQL> INSERT INTO Student_Announcement (Student_id, Module_id, Teacher_id, Announcement_Id)
  2  VALUES (6, 6, 7, 6);

1 row created.

SQL>
SQL> INSERT INTO Student_Announcement (Student_id, Module_id, Teacher_id, Announcement_Id)
  2  VALUES (7, 7, 1, 7);

1 row created.
```

*Figure 41: Insert into Student_Announcement Table.*



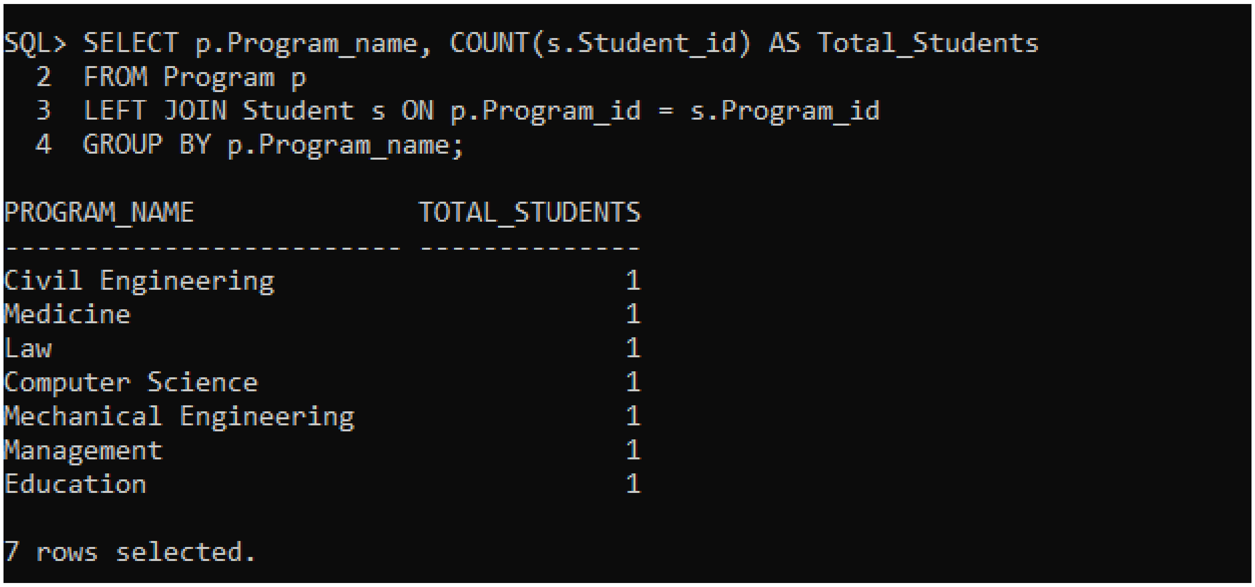
```
SQL> select * from Student_Announcement;

STUDENT_ID  MODULE_ID TEACHER_ID ANNOUNCEMENT_ID
---------- ---------- ---------- ---------------
         1          1          2               1
         2          2          3               2
         3          3          4               3
         4          4          5               4
         5          5          6               5
         6          6          7               6
         7          7          1               7

7 rows selected.

SQL> commit;

Commit complete.

SQL>
```

*Figure 42: Select from student_Announcement Table.*

# 4)Query

## 4.1 Information query

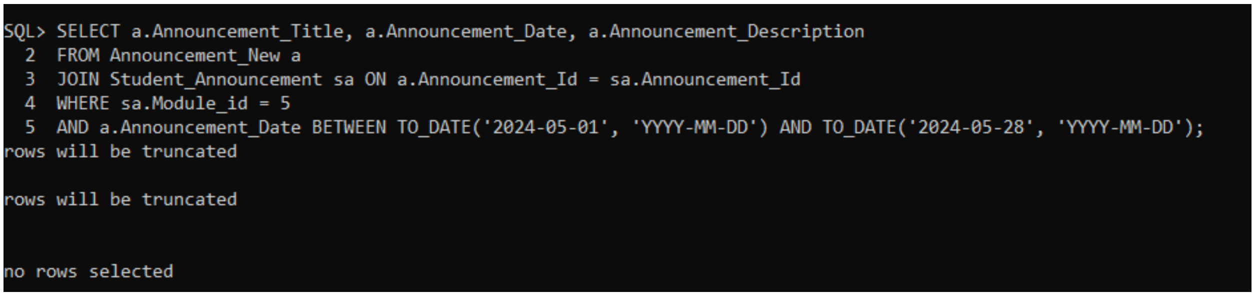
**1) List the programs that are available in the college and the total number of students enrolled in each.**



```
SQL> SELECT p.Program_name, COUNT(s.Student_id) AS Total_Students
     FROM Program p
     LEFT JOIN Student s ON p.Program_id = s.Program_id
     GROUP BY p.Program_name;

PROGRAM_NAME              TOTAL_STUDENTS
------------------------- --------------
Civil Engineering                      1
Medicine                               1
Law                                    1
Computer Science                       1
Mechanical Engineering                 1
Management                             1
Education                              1

7 rows selected.
```

*Figure 43: Information query"1".*

**2) List all the announcements made for a particular module starting from 1st May 2024 to 28th May 2024.**

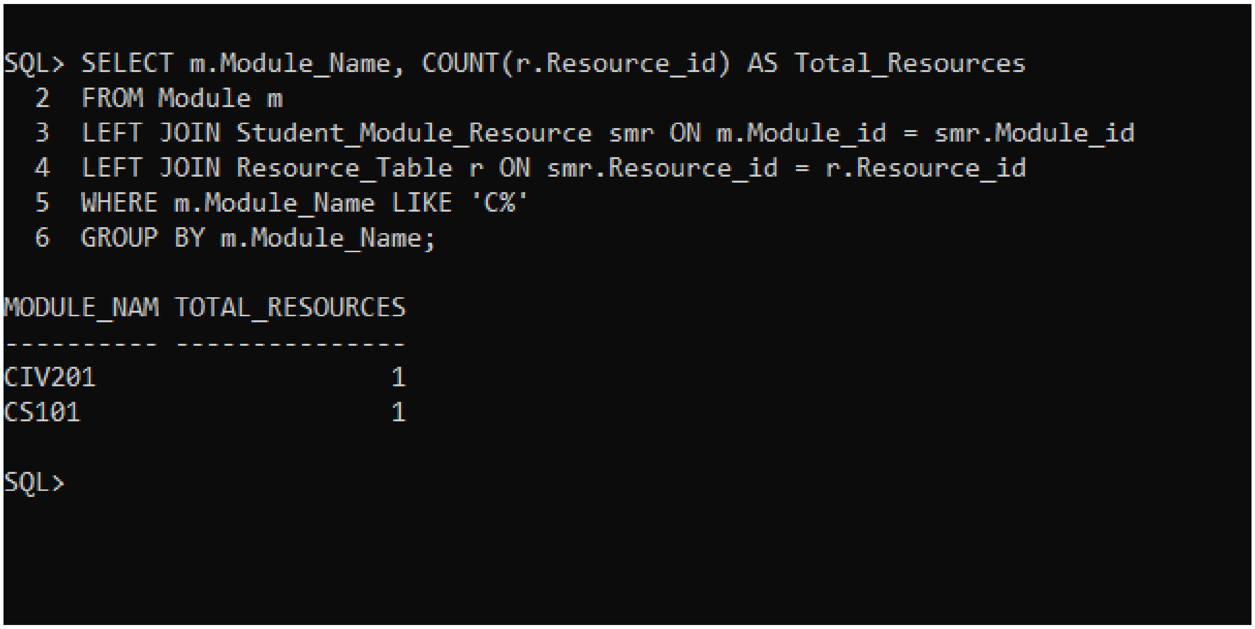


```
SQL> SELECT a.Announcement_Title, a.Announcement_Date, a.Announcement_Description
     FROM Announcement_New a
     JOIN Student_Announcement sa ON a.Announcement_Id = sa.Announcement_Id
     WHERE sa.Module_id = 5
     AND a.Announcement_Date BETWEEN TO_DATE('2024-05-01', 'YYYY-MM-DD') AND TO_DATE('2024-05-28', 'YYYY-MM-DD');
rows will be truncated

rows will be truncated


no rows selected
```

*Figure 44: Information query "2".*

As I have no data entry starting from 1st May 2024 to 28th May 2024 so as a result there is no rows selected.

**3) List the names of all modules that begin with the letter 'C', along with the total number of resources uploaded for those modules**

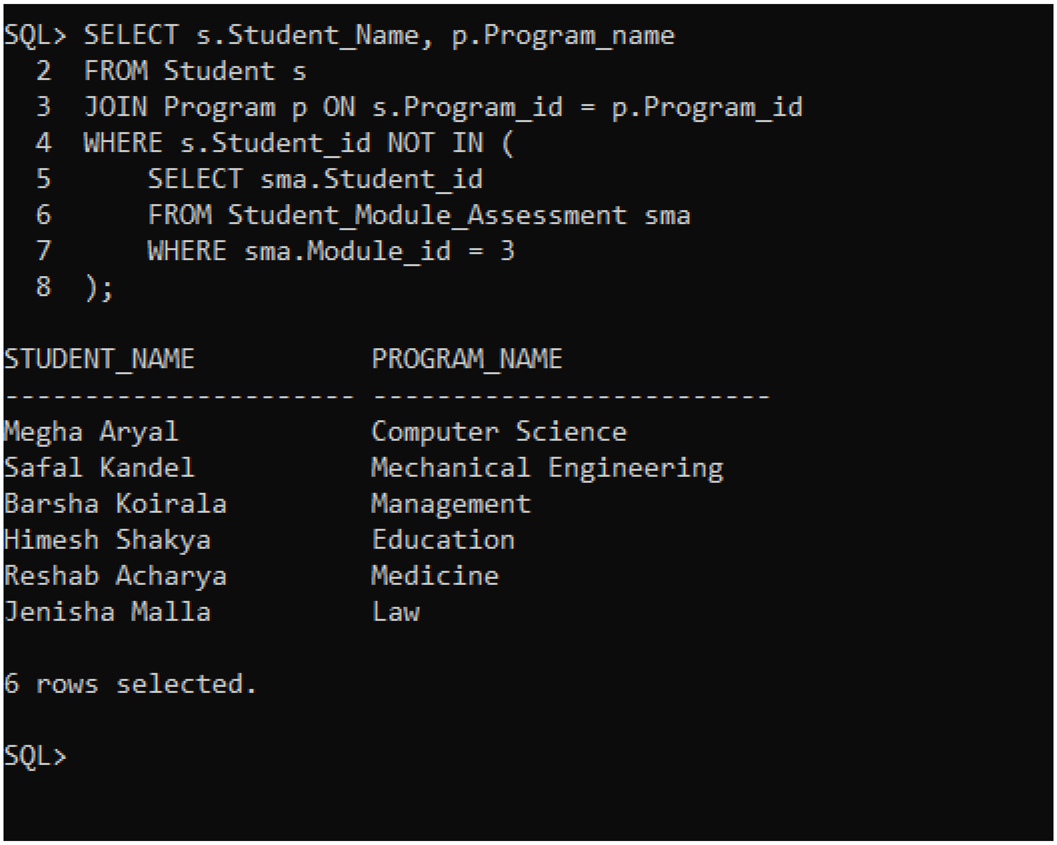


```
SQL> SELECT m.Module_Name, COUNT(r.Resource_id) AS Total_Resources
     FROM Module m
     LEFT JOIN Student_Module_Resource smr ON m.Module_id = smr.Module_id
     LEFT JOIN Resource_Table r ON smr.Resource_id = r.Resource_id
     WHERE m.Module_Name LIKE 'C%'
     GROUP BY m.Module_Name;

MODULE_NAM TOTAL_RESOURCES
---------- ---------------
CIV201                   1
CS101                    1

SQL>
```

*Figure 45: Information query "3".*

There in question it is asked to list all the modules that begin with “D” but I have not inserted any module starting with letter “D” so here I have performed a query using  letter “C”.

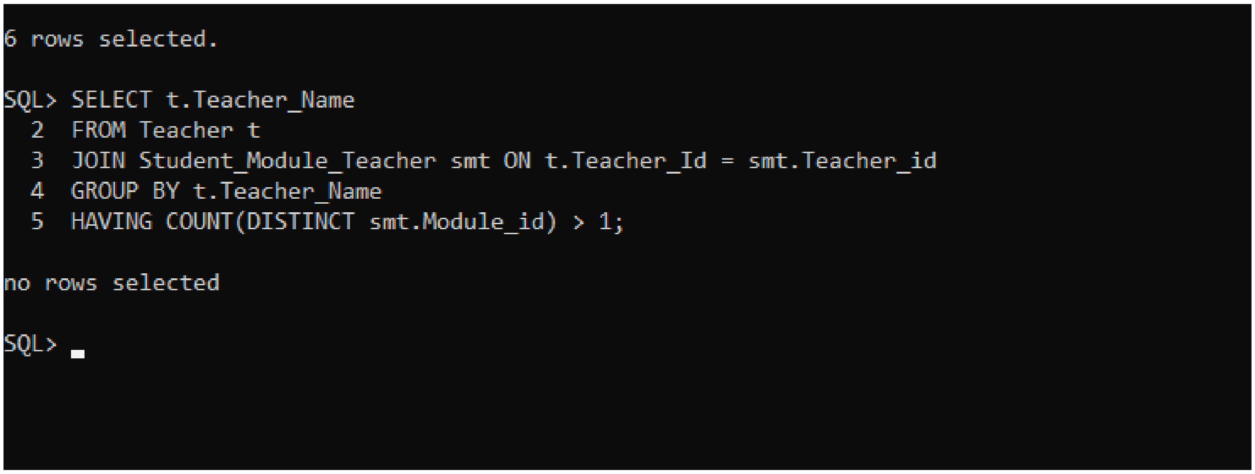
**4) List the names of all students along with their enrolled program who have not submitted any assessments for a particular module.**



```
SQL> SELECT s.Student_Name, p.Program_name
     FROM Student s
     JOIN Program p ON s.Program_id = p.Program_id
     WHERE s.Student_id NOT IN (
         SELECT sma.Student_id
         FROM Student_Module_Assessment sma
         WHERE sma.Module_id = 3
     );

STUDENT_NAME           PROGRAM_NAME
---------------------- -------------------------
Megha Aryal            Computer Science
Safal Kandel           Mechanical Engineering
Barsha Koirala         Management
Himesh Shakya          Education
Reshab Acharya         Medicine
Jenisha Malla          Law

6 rows selected.

SQL>
```

*Figure 46: Information query "4".*

**5) List all the teachers who teach more than one module**



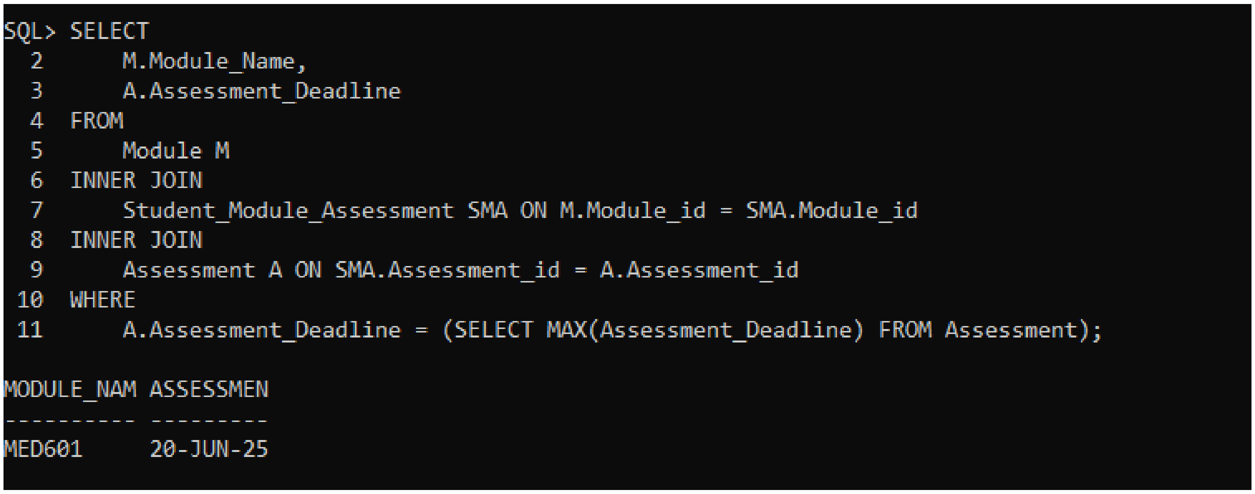
```
6 rows selected.

SQL> SELECT t.Teacher_Name
     FROM Teacher t
     JOIN Student_Module_Teacher smt ON t.Teacher_Id = smt.Teacher_id
     GROUP BY t.Teacher_Name
     HAVING COUNT(DISTINCT smt.Module_id) > 1;

no rows selected

SQL>
```

*Figure 47: Information query "5".*

Here in my database system one teacher is supposed to teach only one module so there is no rows selected to list all the teachers who teach more than one module.

## 4.2 Transaction Query

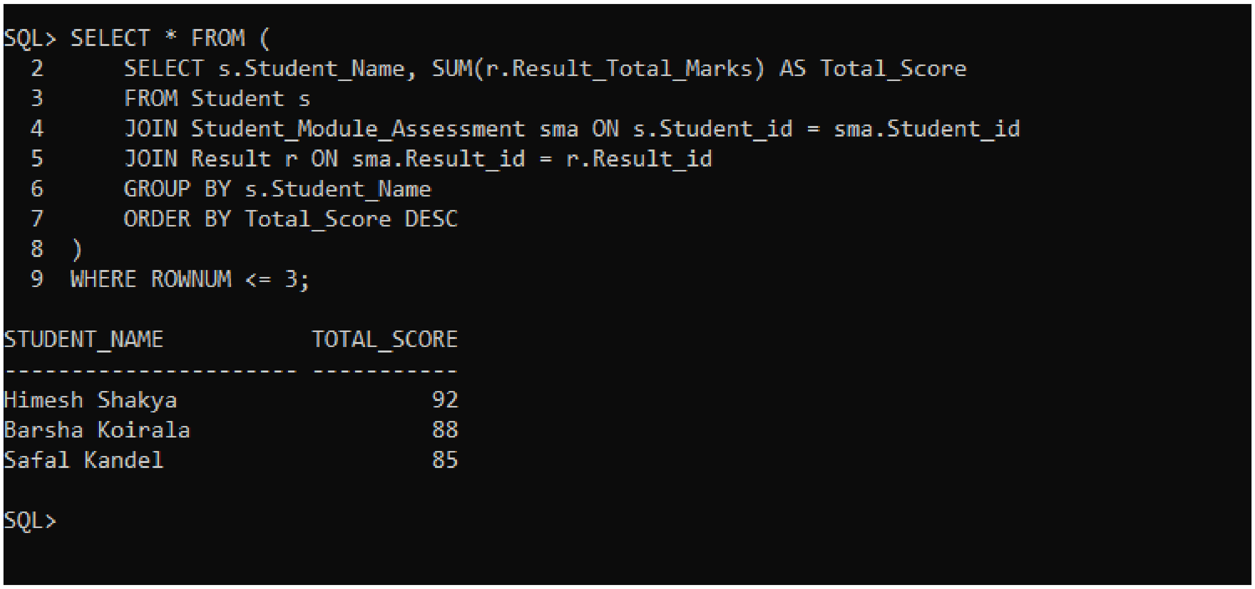
### 1) Identify the module that has the latest assessment deadline



```
SQL> SELECT
         M.Module_Name,
         A.Assessment_Deadline
     FROM
         Module M
     INNER JOIN
         Student_Module_Assessment SMA ON M.Module_id = SMA.Module_id
     INNER JOIN
         Assessment A ON SMA.Assessment_id = A.Assessment_id
     WHERE
         A.Assessment_Deadline = (SELECT MAX(Assessment_Deadline) FROM Assessment);

MODULE_NAM ASSESSMEN
---------- ---------
MED601     20-JUN-25
```

*Figure 48: Transaction query "1".*

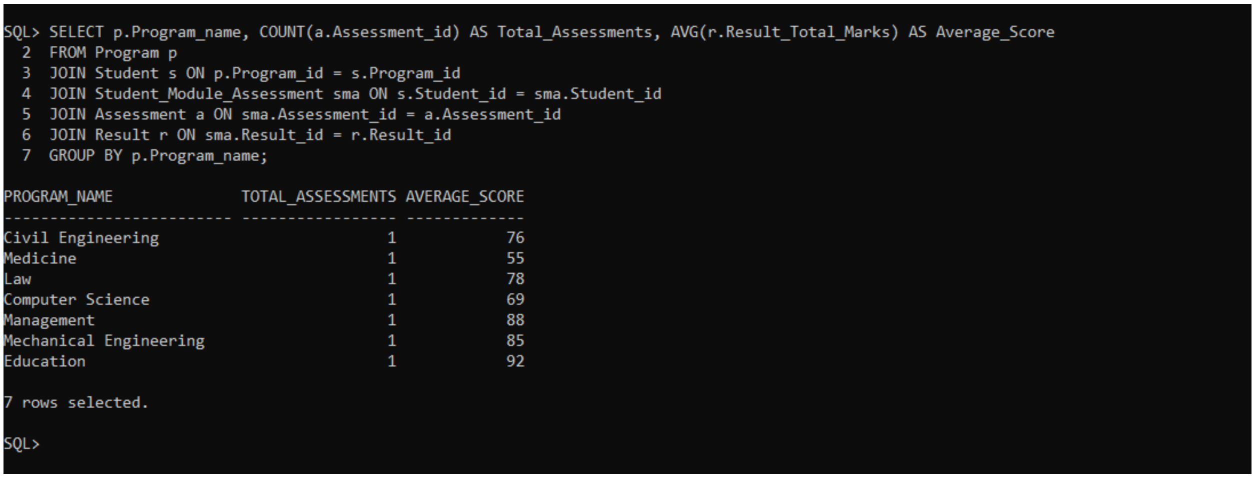
### 2) Find the top three students who have the highest total score across all modules



```
SQL> SELECT * FROM (
         SELECT s.Student_Name, SUM(r.Result_Total_Marks) AS Total_Score
         FROM Student s
         JOIN Student_Module_Assessment sma ON s.Student_id = sma.Student_id
         JOIN Result r ON sma.Result_id = r.Result_id
         GROUP BY s.Student_Name
         ORDER BY Total_Score DESC
     )
     WHERE ROWNUM <= 3;

STUDENT_NAME          TOTAL_SCORE
--------------------- -----------
Himesh Shakya                  92
Barsha Koirala                 88
Safal Kandel                   85

SQL>
```

*Figure 49: Transaction query "2".*

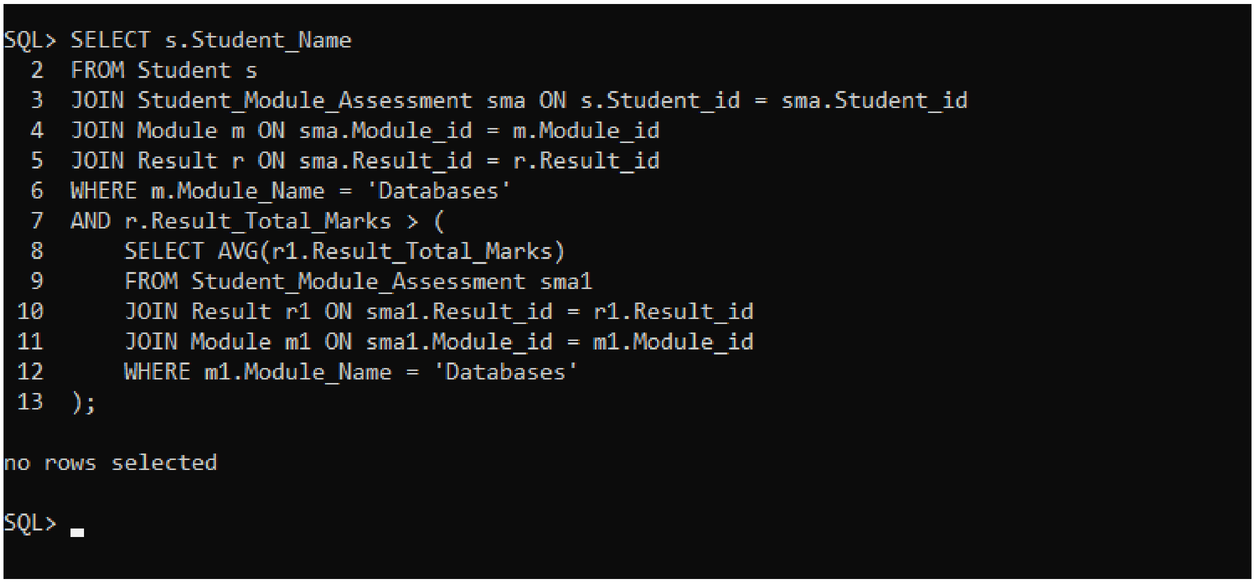
**3) Find the total number of assessments for each program and the average score across all assessments in those programs**



```
SQL> SELECT p.Program_name, COUNT(a.Assessment_id) AS Total_Assessments, AVG(r.Result_Total_Marks) AS Average_Score
     FROM Program p
     JOIN Student s ON p.Program_id = s.Program_id
     JOIN Student_Module_Assessment sma ON s.Student_id = sma.Student_id
     JOIN Assessment a ON sma.Assessment_id = a.Assessment_id
     JOIN Result r ON sma.Result_id = r.Result_id
     GROUP BY p.Program_name;

PROGRAM_NAME              TOTAL_ASSESSMENTS AVERAGE_SCORE
------------------------- ----------------- -------------
Civil Engineering                         1            76
Medicine                                  1            55
Law                                       1            78
Computer Science                          1            69
Management                                1            88
Mechanical Engineering                    1            85
Education                                 1            92

7 rows selected.

SQL>
```

*Figure 50:Transaction query "3".*

**4) List the students who have scored above the average score in the 'Databases' module.**

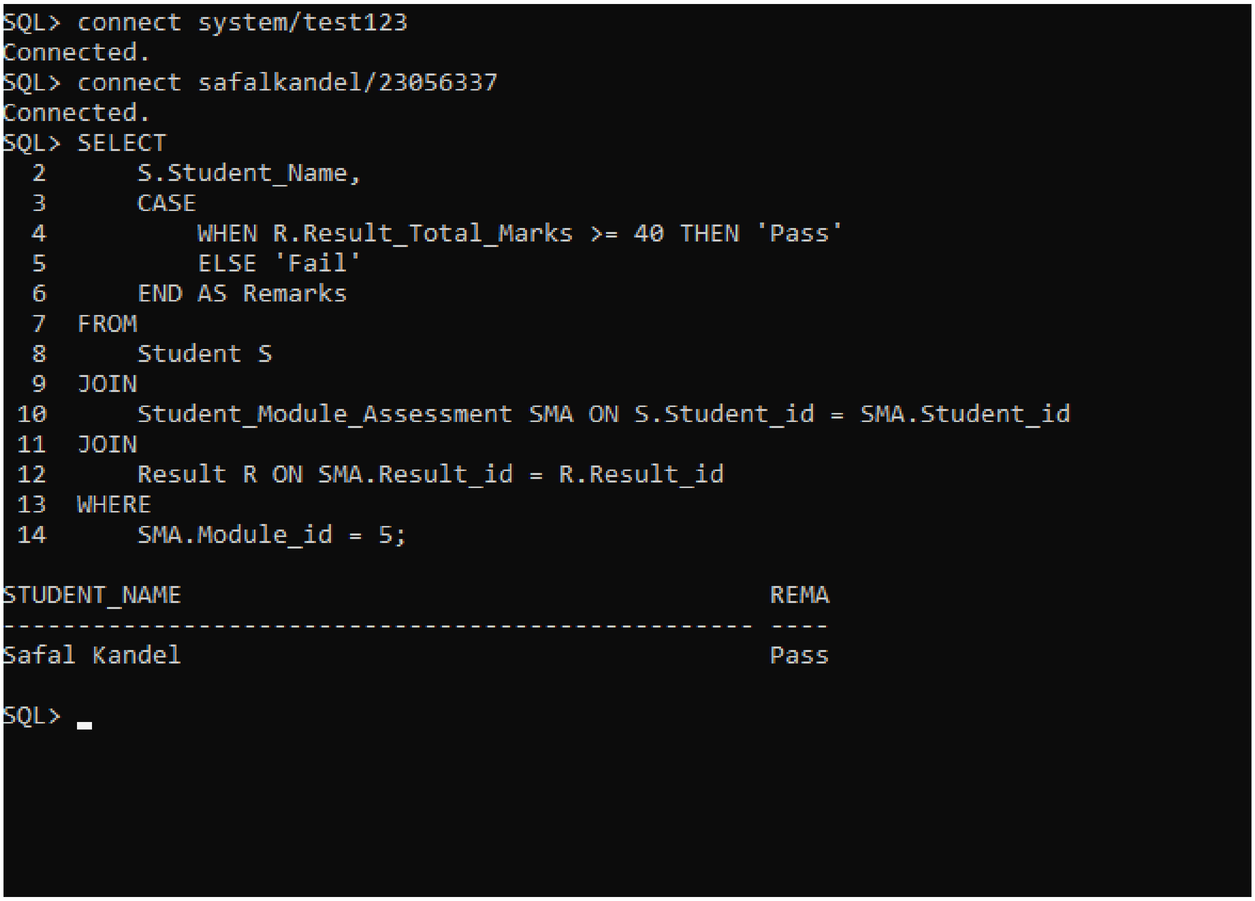


```
SQL> SELECT s.Student_Name
     FROM Student s
     JOIN Student_Module_Assessment sma ON s.Student_id = sma.Student_id
     JOIN Module m ON sma.Module_id = m.Module_id
     JOIN Result r ON sma.Result_id = r.Result_id
     WHERE m.Module_Name = 'Databases'
     AND r.Result_Total_Marks > (
         SELECT AVG(r1.Result_Total_Marks)
         FROM Student_Module_Assessment sma1
         JOIN Result r1 ON sma1.Result_id = r1.Result_id
         JOIN Module m1 ON sma1.Module_id = m1.Module_id
         WHERE m1.Module_Name = 'Databases'
     );

no rows selected

SQL>
```

*Figure 51: Transaction query "4".*

In my database system Module named Databases doesn't exists.

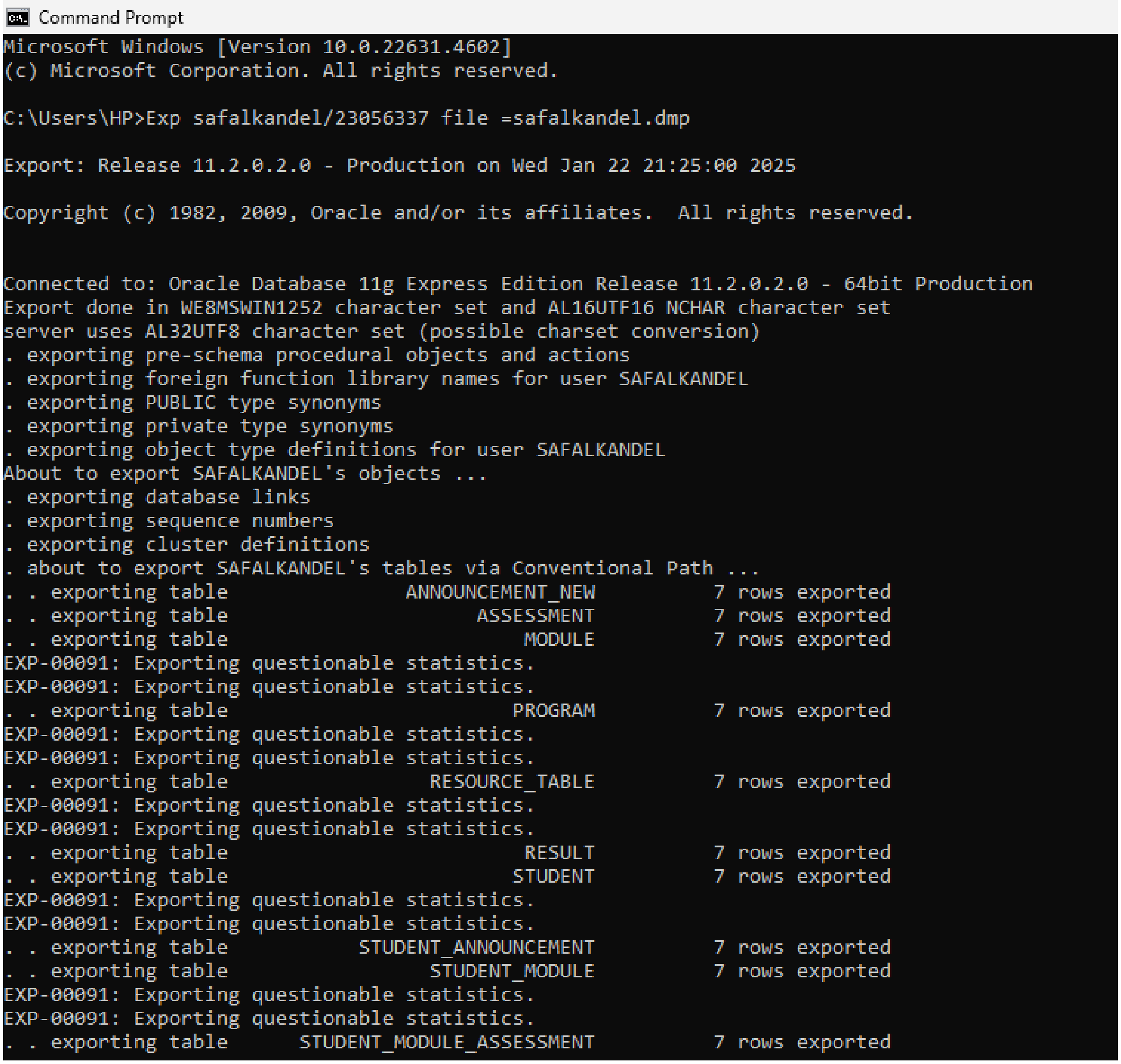
**5) Display whether a student has passed or failed as remarks as per their total aggregate marks obtained in a particular module.**



```
SQL> connect system/test123
Connected.
SQL> connect safalkandel/23056337
Connected.
SQL> SELECT
         S.Student_Name,
         CASE
             WHEN R.Result_Total_Marks >= 40 THEN 'Pass'
             ELSE 'Fail'
         END AS Remarks
     FROM
         Student S
     JOIN
         Student_Module_Assessment SMA ON S.Student_id = SMA.Student_id
     JOIN
         Result R ON SMA.Result_id = R.Result_id
     WHERE
         SMA.Module_id = 5;

STUDENT_NAME                                       REMA
-------------------------------------------------- ----
Safal Kandel                                       Pass

SQL>
```

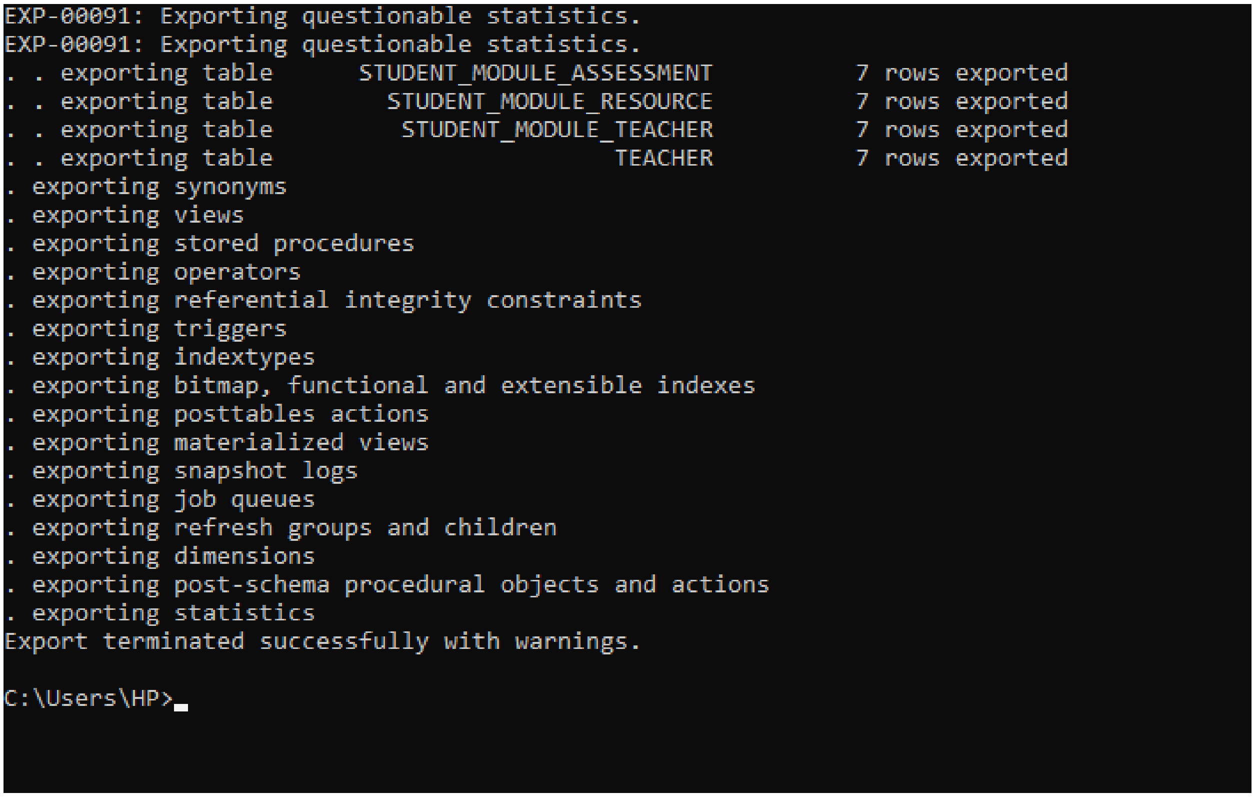
*Figure 52:Transaction query"5".*

## Dump file



```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>Exp safalkandel/23056337 file =safalkandel.dmp

Export: Release 11.2.0.2.0 - Production on Wed Jan 22 21:25:00 2025

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 SAFALKANDEL
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user SAFALKANDEL
About to export SAFALKANDEL's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export SAFALKANDEL's tables via Conventional Path ...
. . exporting table               ANNOUNCEMENT_NEW          7 rows exported
. . exporting table                     ASSESSMENT          7 rows exported
. . exporting table                         MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        PROGRAM          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 RESOURCE_TABLE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         RESULT          7 rows exported
. . exporting table                        STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table           STUDENT_ANNOUNCEMENT          7 rows exported
. . exporting table                 STUDENT_MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table      STUDENT_MODULE_ASSESSMENT          7 rows exported
```

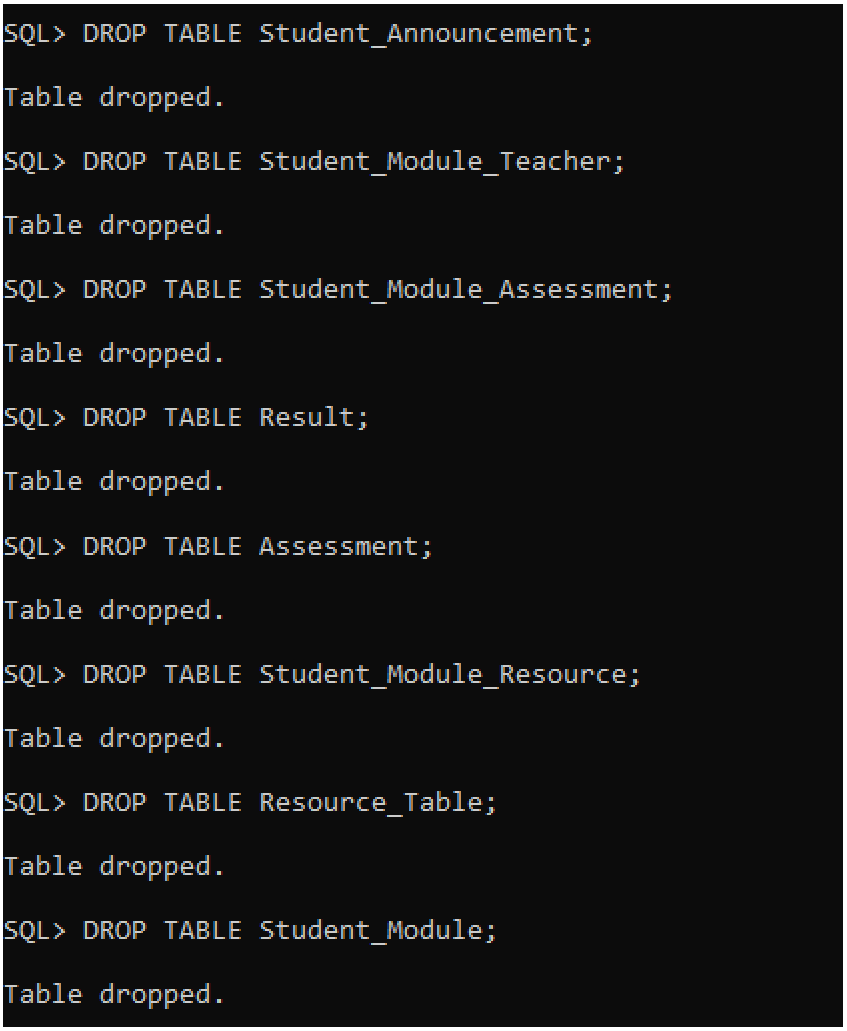
*Figure 53: Dump File I*



```
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table      STUDENT_MODULE_ASSESSMENT          7 rows exported
. . exporting table        STUDENT_MODULE_RESOURCE          7 rows exported
. . exporting table         STUDENT_MODULE_TEACHER          7 rows exported
. . exporting table                        TEACHER          7 rows exported
. 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.

C:\Users\HP>
```

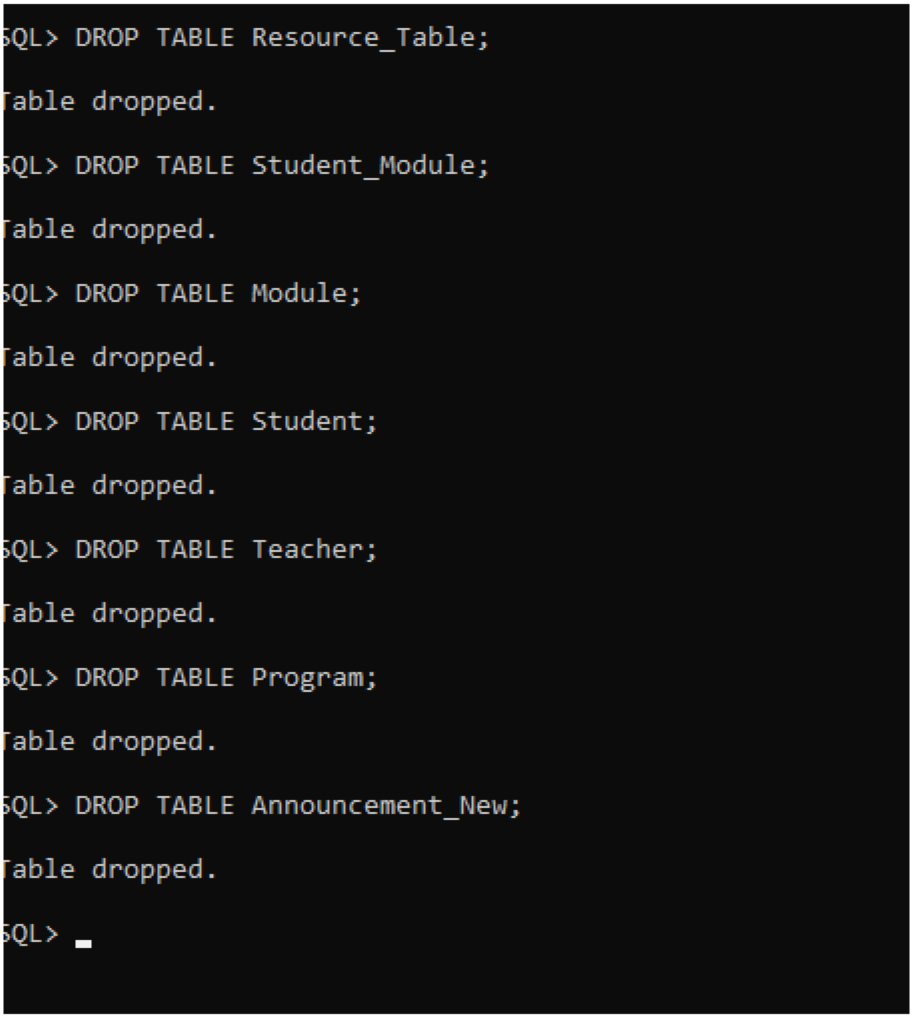
*Figure 54: Dump File II.*

## Drop table



```
SQL> DROP TABLE Student_Announcement;

Table dropped.

SQL> DROP TABLE Student_Module_Teacher;

Table dropped.

SQL> DROP TABLE Student_Module_Assessment;

Table dropped.

SQL> DROP TABLE Result;

Table dropped.

SQL> DROP TABLE Assessment;

Table dropped.

SQL> DROP TABLE Student_Module_Resource;

Table dropped.

SQL> DROP TABLE Resource_Table;

Table dropped.

SQL> DROP TABLE Student_Module;

Table dropped.
```

*Figure 55: Dropping Table (I).*



```
;QL> DROP TABLE Resource_Table;

Table dropped.

;QL> DROP TABLE Student_Module;

Table dropped.

;QL> DROP TABLE Module;

Table dropped.

;QL> DROP TABLE Student;

Table dropped.

;QL> DROP TABLE Teacher;

Table dropped.

;QL> DROP TABLE Program;

Table dropped.

;QL> DROP TABLE Announcement_New;

Table dropped.

;QL> _
```

*Figure 56: Dropping Table (II).*

## 8)Critical Evaluation

### Learning from the Coursework.

It was during this course that I learned most about database design and implementation in developing a very robust e-classroom platform system. Having understood entities, attributes, and relationships, I also learned normalizing data structures into Third Normal Form, 3NF, and implementing them in Oracle SQL. Preparing ERDs further enabled me to visualize the flow of data in order to obtain appropriately logical arrangement for entities. This assignment on normalization and querying indeed gave insight into how theoretical concepts could actually be applied to solving real world-type problems and helped me boost my critical and problem-solving aspect of thinking.

### Challenges Faced:

There were ups and downs in the journey, with much of the integrity of data compromised while establishing relationships and constraints, especially in complex entities, such as assessments and resources, that may need a highly efficient writing of Oracle SQL queries for advanced functionalities and transactions where one needed syntax and logical accuracy to be maniacally attentive. It was also very time draining to adhere to the demand of course-required detailed documentation and screenshots for every step involved in creating the object. Besides, balancing theoretical understanding with technical accuracy became exasperating in itself. It was not easy going.

### Overall Experience:

Overall, this course was an extremely enriching experience in many ways; indeed, it enhanced both my technical and analytical abilities. Here was the opportunity to put academic knowledge into practice in one large project, and it was a real exhaustive experience preparatory to the database management challenges awaiting me in the real world. The setbacks notwithstanding, there is gratification in designing something from scratch and in observing its progress.

## 9. Conclusion

The coursework designates the development of a robust database system that would serve the needs of the "E-Classroom Platform," conforming to proper normalisation principles while observing the set business rules. This involves analysing the operational needs by developing entity-relationship diagrams, further developing the structure for consistency and efficiency in the data. We reduced data redundancy, improved data integrity, and established relationships between key entities such as programs, modules, students, teachers, assessments, and resources with the development of a fully normalized database.

Also, the use of Oracle SQL in implementing this database demonstrated how to apply theoretical concepts to make such a system workable, scalable, and pertinent to practical needs. The incorporation of structured queries and comprehensive test data was the validation that this system was indeed capable of managing complex educational operations efficiently. This project has also laid a very solid foundation for enhancements that could be made in the future to this platform, aside from bringing forth the importance of proper database design in creating dynamic digital environments. Knowledge gained from this coursework strengthened our understanding of database principles and their critical role in modern information systems.

## Bibliography

Chris, K. (2022, 12 21). *freeCodeCamp*. Retrieved from https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/

Geeksforgeeks. (2025, 01 09). Retrieved from Geeksforgeeks: https://www.geeksforgeeks.org/first-normal-form-1nf/