LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

100% Individual Coursework

Autumn 2024

Credit: 15 Semester Long Module

Student Name: Carron Singh London

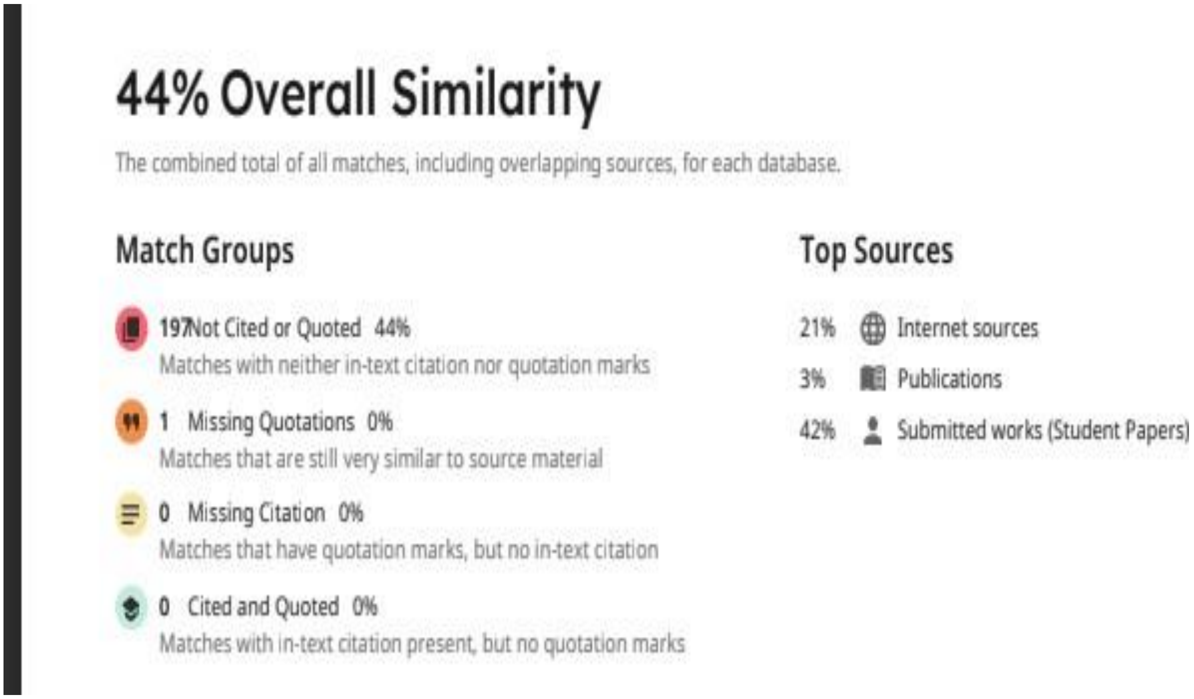
Met ID: 23050380

Assignment Submission Date:

31/12/2024

Word Count: 5800

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.



44% Overall Similarity

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

Match Groups

197Not Cited or Quoted 44%
Matches with neither in-text citation nor quotation marks

1 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

21%	Internet sources
3%	Publications
42%	Submitted works (Student Papers)

Table of Contents

Table of Figure

Table of Table

1. Introductions

1.1 Business Introduction

The Trinity International College, which is in Dillibazar, Kathmandu is one of the most well recognized higher-education providers in Nepal. It was founded with a long-term academic excellence in mind, and it also provides a high-quality education on an institutionally rigorous and student-centered framework. The College offers a very wide a range of courses given the +2 programs in Science, Management and Humanities, the undergraduate as well as the postgraduate courses affiliated to national and international examining bodies including Cambridge International Examinations in particular. The pedagogical model adopted by the College is based on comprehensive development, which makes it combine rigorous academic training and acquisition of practical skills. Instead, the curriculum and the instructional design are strategically designed to instill critical thinking, innovation, and adaptability, therefore, getting the learners ready to face the challenges in the future. Trinity International College has thus become a learning hub to students who aspire to achieve their goals in the domestic and international arena.

Figure 1 Trinity International College

Under this aspiration, this institution has various bachelor’s degrees to offer, such as Computing, Networking, Multimedia, and specialization in them. These programs are inclusive of core academic subjects, which cut across issues like programming, database systems, software engineering, and professional ethics.

Proposed Digital Learning Platform:

As stated by Ms. Mary, the school intends to establish a digital platform that would facilitate. communication, improve academic tracking, and manage other resources. Features include students- teacher management, enrollment in courses, assessments, and resources tracking.

1.2 Description of Current Business Activities and Operations

Since its inception, Trinity international college has taken the mission of ensuring its students are imparted with competencies needed in a competitive future. To realize this goal, the school takes up an integrated approach where intensive academic teaching is coupled with a wide coverage of the co-curricular and extra-curricular programs. The above-going projects which have already been undertaken are illustrative of this holistic student development undertaking:

- **Sports and Physical Well-being:** Sport activities in the form of football, basketball, and running further the idea of sports to keep students on the physical level, enhance cooperation and communication skills. A habitual practice accompanied by the desire to get better establishes an attitude of diligence which can be transferred to other areas of life.

- **Scholarships and Diversity:** The college gives scholarships in consideration of excellence and affordability. These scholarships are made to facilitate equal opportunity to higher education by all students regardless of their reasons for their economic and social status. Through the support, Trinity will allow students to follow their preferred careers without fear of financial impositions.

- **Creative Arts and Cultural Expression:** The same institution also provides organized ways in which the students can embark on arts and humanities. Such programs allow uncovering and fostering creativity. Sharing artistic work or performance with the outside world will encourage self-esteem and help appreciate the value of aestheticism much more.

- **Community Outreach:** Trinity partners with community national organizations in the goal of moving the environment stewardship and increasing populace wellbeing. Tree-planting campaigns, as well as health-awareness events are the activities that give the students opportunities to gain experience, the value of responsibility to the community, to develop interpersonal relationships and empathize.

- **International exposure:** The college also promotes interaction of students in international circles in the form of exchanges programs and in intercultural activities within and outside the country. These experiences expand the worldviews of participants, develop such competence called intercultural competency, and sharpen the participants in skills that are deemed to be crucially important to be a global citizen in the modern world.

1.3 Business Rules

- One student can be enrolled in one program, but a program can have many students.
- A student has many results, but one result belongs to one module.
- One program has many modules, but a module belongs to many programs.
- A module has many resources, but a resource belongs to only one module.
- A module has many assessments, but assessments belong to only one module.
- A teacher teaches one module, but one module has many teachers.
- An announcement for many modules can be posted but the announcements belong to them. respective modules.
- A result belongs to one assessment and an assessment has one result.

1.4 Assumptions

- The students are not allowed to pursue many programs.
- The format of the program (structuring in modules and assessment) does not change every year.
- The modules are ever present in the program duration.
- The teachers are subjected to certain modules. The resources must be consumed under a definite sequence.
- No student can be enrolled to repeat a module.

2. Initial Erd

2.1 Identification of Entities and Attributes

2.1.1 Student Entity

S. No.	Attribute Name	Data Type	Size	Constraint
1	Student_ID	Number	10	Primary Key
2	Student_Name	Character	50	Not Null
3	Date_of_Birth	Date	-	Not Null
4	Student_address	Character	40	Not Null
5	Student_Email	Character	40	Unique

Table 1 Student Entity

2.1.2 Program Entity

S. No.	Attribute Name	Data Type	Size	Constraint
1	Program_Name	Character	40	Not Null
2	Program_ID	Number	10	Primary Key
3	Program_duration	Date	-	Not Null
4	Program_credits	Number	3	Primary Key

Table 2 Program Entity

2.1.3 Module Entity

S. No.	Attribute Name	Data Type	Size	Constraint
1	Module_ID	Number	10	Primary Key
2	Module_Name	Character	50	Not Null
3	Credit_hours	Number	3	Not Null
4	Teacher_ID	Number	10	Unique
5	Teacher_Name	Character	60	Not Null
6	Teacher_Email	Character	50	Unique
7	Teacher_contact	Number	10	Unique
8	Announcement_ID	Number	10	Unique
9	Announcement_Date	Date	-	Not Null
10	Announcement_Details	Character	300	Not Null
11	Assessment_ID	Number	10	Unique
12	Assessment_Title	Character	50	Not Null
13	Assessment_Deadline	Date	-	Not Null
14	Assessment_weightage	Number	3	Not Null
15	Assessment_status	Character	20	Not Null
16	Obtained_Marks	Number	10	Not Null
17	Resource_ID	Number	10	Unique
18	Resource_Title	Character	100	Not Null
19	Resource_Type	Character	30	Not Null
20	Resource_duration	Number	4	Not Null
21	Resource_completion_status	Character	20	Not Null

Table 3 Module Entity

2.2 Initial ER Diagram

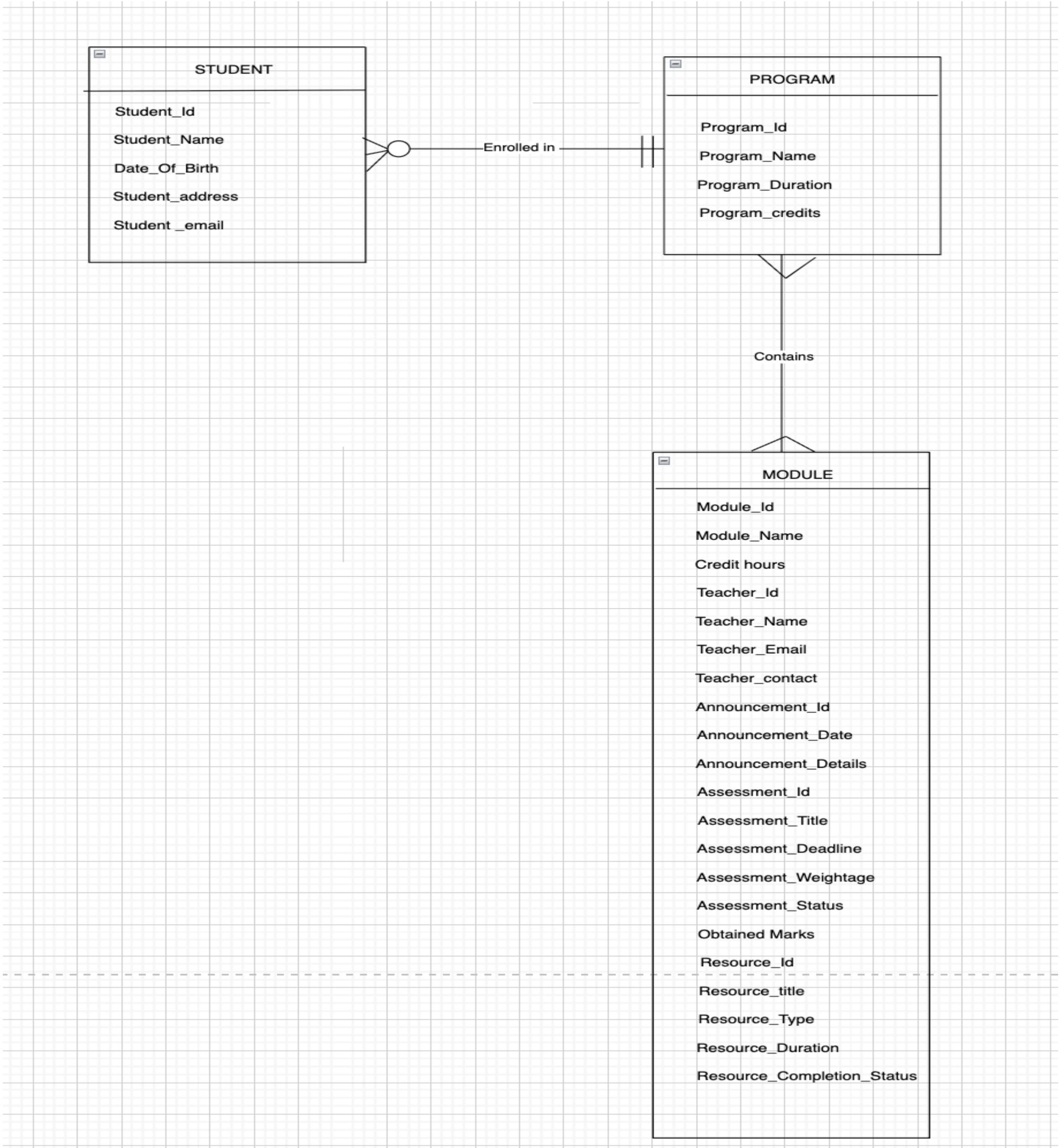


Figure 2 Initial ER Diagram

3. Normalization

Normalization process is an orderly way of data organization to ensure that traces of the databases are safe and without redundancy. This refinement will be done in several stages as it speaks of lessening redundancy and escaping invalid or anomalous relationships. In stages of UNF, 1NF, 2NF, and 3NF. Therefore, data-base designers can avoid the insertion, update and deletion anomalies using this methodology and at the same time reduce the storage space and hence improve the performance of the system.(Bharati & Podder, 2022)

Step 1: Unnormalized Form (UNF)

After initially normalizing a database, Repeat Groups and Multi-valued attributes are stored as an Unnormalized Form (UNF) and these are non-relational items. In this set up the data get maintained in its raw form with no normalization occurring. It claims the stored information such as the occurrences or sets of attribute values and normalization procedures, specifically, are not performed. As a result, the lack of normalization implies the unsystematic pattern of the data, with all kinds of information being represented in one table that will result in redundancy and inefficiency. The data is not separated into categories and thus the same information found in two or more instances in the same line thus creating a possibility of error and the lack of consistency. A point in case is a storage of a student record that can be covered with an enormous number of modules, tests, or materials; all these items are normally inserted into one row so management and updating activities turn out to be cumbersome. This design is normally used as a basis in disassembling and rearranging the data into smaller and interlinked tables that reduce repetitions and improve performance once the normalization processes are executed. (Rahmawati et al., 2023)

Example:

- Student row can have multiple modules, teachers, or assessments.
- For ex Student_ID: 001, Name: Mukesh , Module1: Database, Module2: OOP, Teacher1: Sabita, Teacher2: Vidit.
- This leads to nested groups within one line resulting in data that is not easy to maintain.
- UNF is the first step into applying 1NF, 2NF, 3NF to cleanup.

The UNF is :

Student(Student_Id,Student_name, Date_of_birth ,Student_address,Student_email,Program_Id,Program_name,Program_duration,Program_credits,{Module_Id,Module_name,Credit_hours,Teacher_Id,Teacher_name,Teacher_email,Teacher_contact,{Announcement_Id,Announcement_date,Announcement_details,{Assessment_Id,Assessment_title,Assessment_deadline,Assessment_weightage,Assessment_status,Obtained_marks},{Resource_ID,Resource_title,Resource_type,Resource_duration,Resource_completion_status}})

Step 2: First Normal Form (1NF)

Normalizing repetitive attributes, the steps in normalizing repetitive attributes are the division of instances in several tables which relate with each other through foreign keys. In this design, a table is a primary entity and contains only distinct information, whereas the rest of attributes (which are also a relational object) are presented in another table, which can be accessed through the primary-entity key, shown in this approach with the symbol *.

Example:

Student_id	Student_name	Student_address	Student_email	Date_of_birth
S001	Ronaldo Khatri	Kathmandu	ronaldo@gmail.com	2000-03-11

Table 4 1NF Example

Eliminating repeating groups:

Student(Student_ID, Student_name, Date_of_birth, Student_address, Student_email ,Program_ID, Program_name, Program_duration,Program_credit)
Student_Module(Student_ID*, Module_ID*, Module_name, Credit_hours)
Student_Module_Teacher(Student_ID*, Module_ID*, Teacher_ID*, Teacher_name, Teacher_Email, Teacher_Contact)
Student_Module_Teacher_Announcement(Student_ID*, Module_ID*, Teacher_ID*, Announcement_ID*, Announcement_Date, Announcement_Details)
Student_Module_Assessment(Student_ID*, Module_ID*, Assessment_ID*, Assessment_Title, Assessment_deadline, Assessment_Weightage, Assessment_status, Obtained_marks)
Student_Module_Resource(Student_ID*, Module_ID*, Resource_ID*, Resource_name, Resource_type, Resource_duration, Resource_Completion_status)

Key Changes:

- All the tables contain only atomic values, not any more nested repeating groups.
- Composite keys such as (Student_ID, Module_ID) must maintain relationships.

Step 3: Second Normal Form (2NF)

This step does partial dependency, wherein one attribute, not a key, depends only on a sub selection of a composite primary key, rather than on the entire compound. This situation is commonly found when an individual table has a combined primary key and when individual columns depend only on a single element of that composite. The eradication of partial dependency involves the division of existing tables into the ones that would have only one, and unique, primary key and all other non-key attributes in the table would only rely on such single key. In this reorganization, an easier and more rational association of data with keys is achieved.

Example – Before and After

Before 2NF (In 1NF):

Student_ID	Module_ID	Student_Name	Module_Name	Marks
S001	M101	Ankit	Database	80
S002	M102	Ankita	OOP	75

Table 5 Before 2NF (In 1NF) Example

In the relation schema to be considered, the composite key is given as (Student_ID, Module_ID). However, Student_Name is determined by Student_ID and hence is functionally dependent only on Student_ID and similarly Module_Name is determined by Module_ID and, hence, it is also functionally dependent only on Module_ID. Since Student_Name and Module_Name are not the full part of key, it can be concluded that schema has a partial dependency and, therefore, does not completely satisfy the second normal form.

After 2NF – Tables

1. Student Table

Student_ID	**Student_Name**
S001	Ankit
S002	Ankita

Table 6 After 2NF – Student Tables

2. Module Table

Module_ID	**Module_Name**
M101	Database
M102	OOP

Table 7 After 2NF – Module Table

3. Marks Table

Student_ID	**Module_ID**	**Marks**
S001	M101	80
S002	M102	75

Table 8 Marks Table

Key Changes:

- Put your data into different tables using identification keys as limits.
- Separate all attributes that depend on only one element of a multiple key combination.

Student_Module table

Seeing out partial dependencies by analyzing functional dependencies and removing those which are partial dependencies on composite primary keys.

Composite Key: (Student_ID, Module_ID)

Partial Dependencies:

Student_ID → No partial dependency exists in this case, as all non-prime attributes are fully functionally dependent on the entirety of the composite primary key.

Module_ID → Module_name, Credit_hours

Student_Module_Teacher table

Seeing out partial dependencies by analyzing functional dependencies and removing those which are partial dependencies on composite primary keys.

Composite Key: (Student_ID, Module_ID, Teacher_ID)

Partial Dependencies:

Student_ID, Module_ID → No partial dependency exists in this case, as all non-prime attributes are fully functionally dependent on the entirety of the composite primary key.

Student_ID, Teacher_ID → No partial dependency exists in this case, as all non-prime attributes are fully functionally dependent on the entirety of the composite primary key.

Module_ID, Teacher_ID → No partial dependency exists in this case, as all non-prime attributes are fully functionally dependent on the entirety of the composite primary key.

Teacher_ID → Teacher_name, Teacher_ email, Teacher_Contact

Student_Module_Teacher_Announcement table

Seeing out partial dependencies by analyzing functional dependencies and removing those which are partial dependencies on composite primary keys.

Composite Key: (Student_ID, Module_ID, Teacher_ID, Announcement_ID)

Partial Dependencies:

Announcement_ID → Announcement_name, Announcement_date

Other combinations (Student_ID, Module_ID, Teacher_ID, Announcement_ID) have no partial dependencies.

Student_Module_Assessment table

Seeing out partial dependencies by analyzing functional dependencies and removing those which are partial dependencies on composite primary keys.

Composite Key: (Student_ID, Module_ID, Assessment_ID)

Partial Dependencies:

Assessment_ID→Assessment_name, Assessment_Deadline

No partial dependencies on Student_ID or Module_ID alone

Student_Module_Resource table

Seeing out partial dependencies by analyzing functional dependencies and removing those which are partial dependencies on composite primary keys.

Composite Key: (Student_ID, Module_ID, Resource_ID)

Partial Dependencies:

Resource_ID → Resource_name, Resource_type, Resource_duration

No partial dependencies on Student_ID or Module_ID alone

Final 2NF Relations:

Student(Student_ID, Student_name, Date_of_birth, Student_address, Student_email, Program_ID*, Program_name, Program_duration, Program_credit)

Student_Module(Student_ID*, Module_ID*)

Module(Module_ID, Module_name, Credit_hours)

Student_Module_Teacher(Student_ID*, Module_ID*, Teacher_ID*)

Teacher(Teacher_ID, Teacher_name, Teacher_Email, Teacher_Contact)

Student_Module_Teacher_Announcement(Student_ID*, Module_ID*, Teacher_ID*, Announcement_ID*)

Announcement(Announcement_ID, Announcement_Title, Announcement_date, Announcement_details)

Student_Module_Assessment(Student_ID*, Module_ID*, Assessment_ID*, Marks_obtained, Assessment_Status)

Assessment(Assessment_ID, Assessment_Title, Assessment_deadline, Assessment_Weightage, Assessment_Status)

Student_Module_Resource(Student_ID*, Module_ID*, Resource_ID*)

Resource(Resource_ID, Resource_name, Resource_type, Resource_duration, Resource_Compeletion_status)

Step 4: Third Normal Form (3NF)

The process of transforming to third normal form (3NF) involves breaking down relational instances where the attribute dependencies cross over intermediate tuples as opposed to a direct association with the primary key. A transitive dependency exists in a situation in which the attribute and not the primary-key attribute constrains another attribute that lacks primary-key status.

Student

Student_ID → Student_Name, Date_of_Birth, Student_Address, Student_Email, Program_ID Program_ID → Program_Name, Program_Duration, Program_Credits

Transitive Dependency:

Student_ID → Program_ID → Program_Name, Program_Duration, Program_Credits

Transitive dependency exists. So, we separate program attributes.

Final Tables:
Student(Student_ID, Student_Name, Date_of_Birth, Student_Address, Student_Email,Program_ID)
Program(Program_ID, Program_Name, Program_Duration, Program_Credits)

No partial dependency exists in this case, as all non-prime attributes are fully functionally dependent on the entirety of the composite primary key.

Module Table

Module_ID → Module_Name, Credit_Hours

Each non-key attribute has a direct and full dependency on the primary key, placing the relation in 3NF.

Student_Module_Teacher Table

(Student_ID, Module_ID, Teacher_ID)

The relation contains only key attributes; therefore, it is already in 3NF.

Teacher Table

Teacher_ID → Teacher_Name, Teacher_Email, Teacher_Contact

There are no transitive dependencies hence, the relation is in Third Normal Form (3NF).

Student_Module_Teacher_Announcement Table

(Student_ID, Module_ID, Teacher_ID, Announcement_ID)

The relation contains only key attributes therefore it is already in 3NF.

Announcement Table

Announcement_ID → Announcement_Name, Announcement_Date

All attributes are fully functionally dependent on the primary key, satisfying 3NF

Student_Module_Assessment Table

(Student_ID, Module_ID, Assessment_ID)

The absence of transitive dependencies confirms that the relation is in Third Normal Form (3NF)

Assessment Table

Assessment_ID → Assessment_Id, Assessment_Deadline

All attributes are directly dependent on the primary key, ensuring compliance with 3NF

Student_Module_Resource Table

(Student_ID, Module_ID, Resource_ID)

As the relation contains only key attributes, it inherently satisfies Third Normal Form (3NF)

Resource Table

Resource_ID → Resource_Name, Resource_Type, Resource_Duration, Resource_completition_status

The relation is in 3NF, as no attribute depends on any non-key attribute.

Final 3NF Relations

Student(Student_ID, Student_Name, Date_of_Birth, Student_Address, Student_email , Program_ID*)
Program(Program_ID, Program_Name, Program_Duration,Program_credit)

Student_Module(Student_ID*, Module_ID*)
Module(Module_ID, Module_Name, Credit_Hours)

Student_Module_Teacher(Student_ID*, Module_ID*, Teacher_ID*)
Teacher(Teacher_ID, Teacher_Name, Teacher_Email, Teacher_contact)

Student_Module_Teacher_Announcement(Student_ID*, Module_ID*, Teacher_ID*, Announcement_ID*)
Announcement(Announcement_ID, Announcement_Date, Announcement_Details)

Student_Module_Assessment(Student_ID*, Module_ID*, Assessment_ID*, Obtained_Marks, Assessment_Status)
Assessment(Assessment_ID, Assessment_Title, Assessment_Deadline,Assessment_Weightage)

Student_Module_Resource(Student_ID*, Module_ID*, Resource_ID*)
Resource(Resource_ID, Resource_Title, Resource_Type, Resource_Duration,Resource_Completion_Status)

4. Data Dictionary and Final ERD

4.1 Data Dictionary

4.1.1 Student Table:

S.No	Attribute	Data Type	Size	Constraint	Description
1	Student_ID	Number	10	Primary Key	Every student has a special identification number.
2	Student_Name	VARCHAR	50	Not Null	Complete name of the student.
3	Date_of_Birth	DATE	-	Not Null	The student's birth date.
4	Student_address	VARCHAR	40	Not Null	The student’s location.
5	Student_Gmail	VARCHAR	40	Unique	Gmail of the student.
6	Program_Id	Number	10	Foreign Key	The students enrolled academic program.

Table 9 Data Dictionary Student Table

4.1.2 Program Table:

S.No	Attribute	Data Type	Size	Constraint	Description
1	Program_ID	Number	10	Primary Key, Not Null	Unique identifier for each program.
2	Program_Name	VARCHAR	40	Not Null	Name of the academic program.
3	Program_duration	Number	-	Not Null	Length of academic program as per university regulations.
4	Program_credits	Number	3	Not Null	Credit hours required.

Table 10 Program Table

4.1.3 Modules Table:

S.No	Attribute	Data Type	Size	Constraint	Description
1	Module_ID	Number	10	Primary Key, Not Null	Unique identifier for each module.
2	Module_Name	VARCHAR	50	Not Null	Name of the module.
3	Credit_hour	Number	-	Foreign Key, Not Null	Links to the associated program or credit reference.

Table 11 Modules Table

4.1.4 Student_ Module:

S. No.	Attribute Name	Data Type	Size	Constraint	Description
1	Student_id	Number	10	Foreign Key	References the unique ID of a student from the student's table.
2	Module_id	Number	10	Foreign Key	References the unique ID of a module from the Modules table.

Table 12 Student_ Module

4.1.5 Teacher Table:

S.No	Attribute	Data Type	Size	Constraint	Description
1	Teacher_ID	Number	10	Primary Key, Not Null	Unique identifier
2	Teacher_Name	Character	60	Not Null	Full name of the teacher
3	Teacher_Email	Character	50	Unique, Not Null	Email address of the teacher
4	Teacher_contact	Character	10	Unique	Contact number of the teacher

Table 13 Teacher Table

4.1.6 Student_Teacher_Module:

S. No.	Attribute Name	Data Type	Size	Constraint	Description
1	Teacher_id	Number	10	Foreign Key	References the unique ID of a teacher from the Teachers table.
2	Module_id	Number	10	Foreign Key	References the unique ID of a module from the Modules table.
3	Student_id	Number	10	Foreign Key	References the unique ID of a student from the student's table.

Table 14 Student_Teacher_Module

4.1.7 Assessments Table:

S.No	Attribute	Data Type	Size	Constraint	Description
1	Assessment_ID	Number	10	Primary Key, Not Null	Unique identifier for each assessment
2	Assessment_Title	Character	50	Not Null	Title of the assessment
3	Assessment_Deadline	DATE	-	Not Null	Deadline for submission
4	Assessment_Weightage	Number	3	Not Null	Average marking

Table 15 Assessments Table

4.1.8 Module_Student _Assessment:

S. No.	Attribute Name	Data Type	Size	Constraint	Description
1	Assessment_id	Number	10	Foreign Key	References the unique ID of an assessment from the Assessment table.
2	Module_id	Number	10	Foreign Key	References the unique ID of a module from the Modules table.
3	Student_id	Number	10	Foreign Key	References the unique ID of a student from the student's table.
4	Assessment_status	Character	20	Not Null	Indicates the status of the assessment (e.g., Submitted, Pending).
5	Obtained_marks	Number	10	Not Null	Marks obtained by the student in the assessment._

Table 16 Module_Student _Assessment

4.1.9 Resource Table:

S.No		Attribute	Data Type	Size	Constraint	Description
1		Resource_ID	Number	10	Primary Key, Not Null	Each resource has its own unique identifier
2		Resource_Title	Character	100	Not Null	Title of this resource
3		Resource_Type	Character	30	Not Null	Type of the resource (e.g., Video, PDF)
4		Resource_duration	Number	4	Not Null	Time/duration of the resource

Table 17 Resource Table

4.1.10 Resource_Module_Student:

S. No.	**Attribute Name**	**Data Type**	**Size**	**Constraint**	**Description**
1	Resource_id	Number	10	Primary Key	Unique identifier for the learning resource assigned to a student.
2	Module_id	Number	10	Foreign Key	References the unique ID of a module from the Modules table.
3	Student_id	Number	10	Foreign Key	References the unique ID of a student from the student's table.
4	Resource_completion_status	Character	20	Not Null	Indicates whether the student has completed the resource (e.g., Completed, In Progress).

Table 18 Resource_Module_Student

4.1.11 Announcements Table

S.No	Attribute	Data Type	Size	Constraint	Description
1	Announcement_ID	Number	10	Primary Key, Not Null	Each announcement has its own unique identifier
2	Announcement_Description	Character	300	Not Null	Title of the announcement
3	Announcement_Date	DATE	-	Not Null	Announcement date

Table 19 Announcements Table

4.1.12 Announcement_Module_Teacher:

S. No.	**Attribute Name**	**Data Type**	**Size**	**Constraint**	**Description**
1	Announcement_id	Number	10	Foreign Key	References the unique ID of an announcement from the Announcements table.
2	Module_id	Number	10	Foreign Key	References the unique ID of a module from the Modules table.
3	Student_id	Number	10	Foreign Key	References the unique ID of a student from the student's table.
4	Teacher_id	Number	10	Primary Key	Unique identifier for the teacher who made the announcement.

Table 20 Announcement_Module_Teacher:

4.2 Final ER Diagram

Diagram Description: The final ER diagram includes normalized entities with foreign key constraints, ensuring referential integrity and eliminating redundancy.

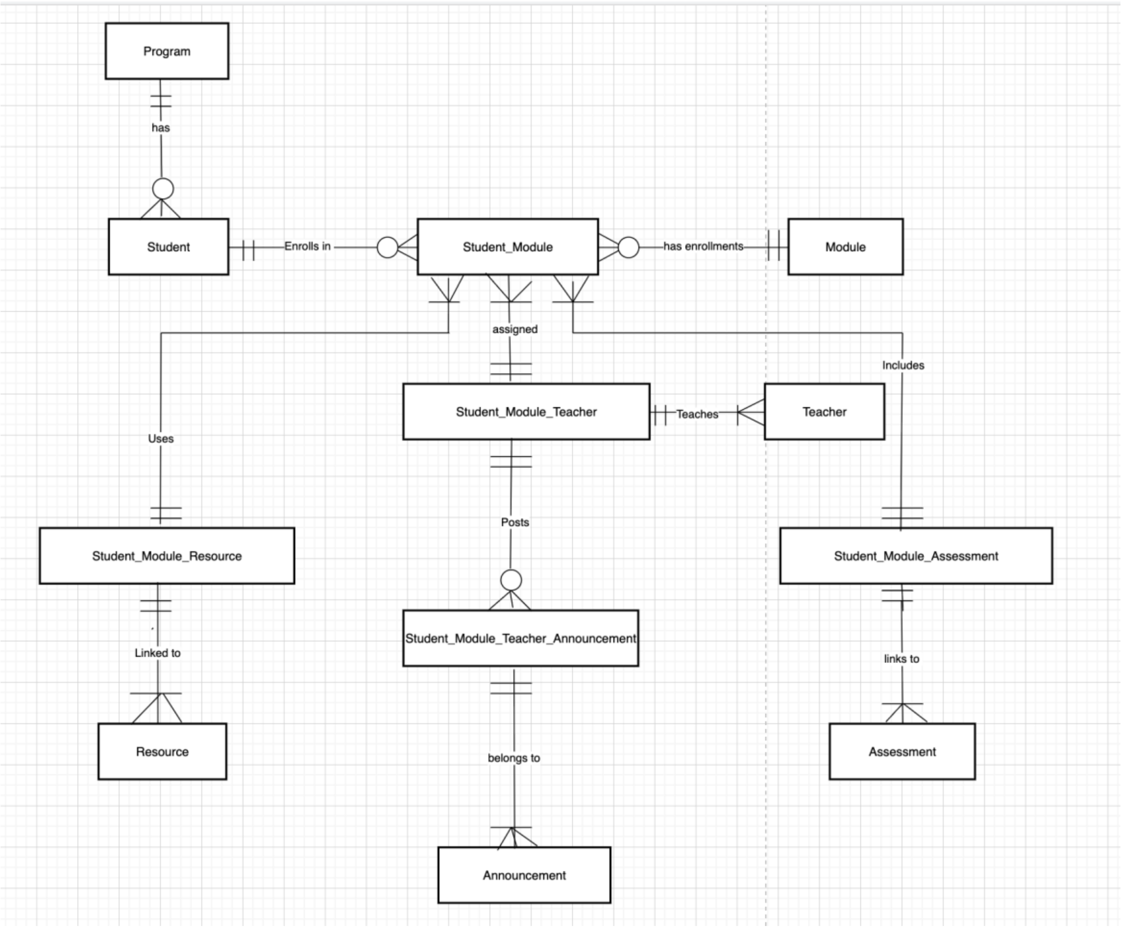
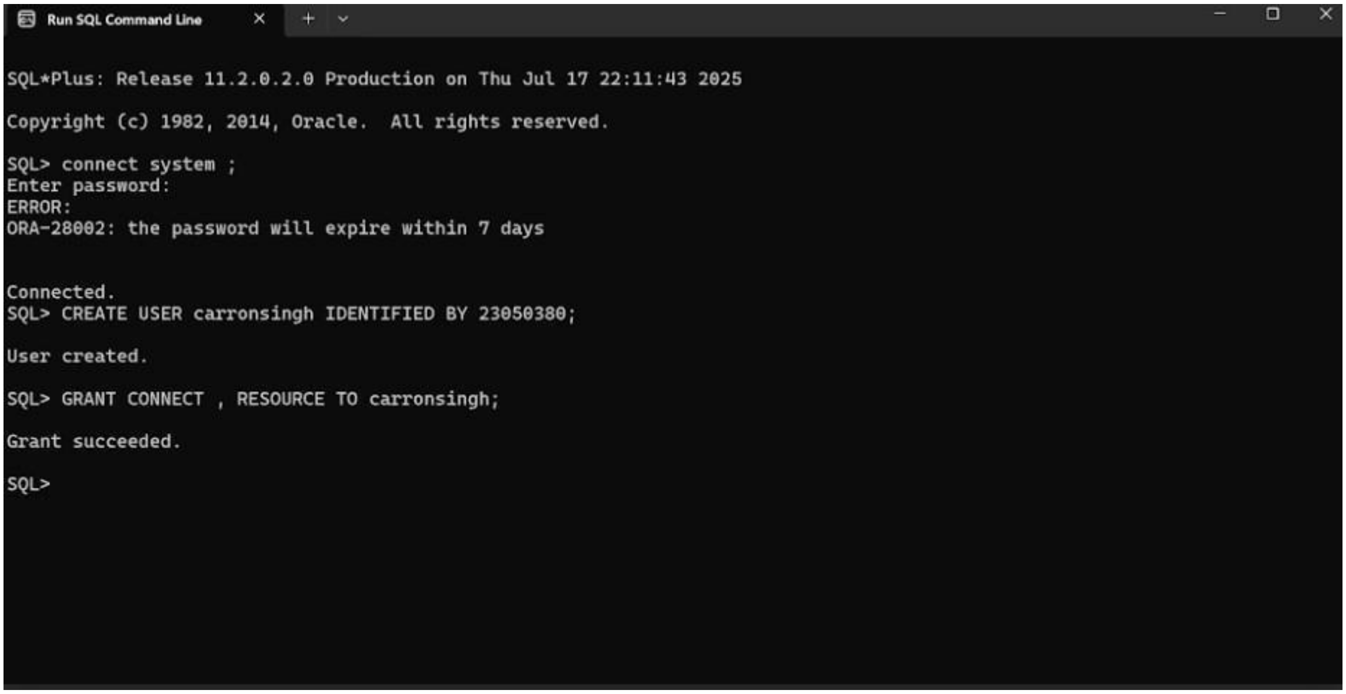


Figure 3 Final ER Diagram

5. Implementation

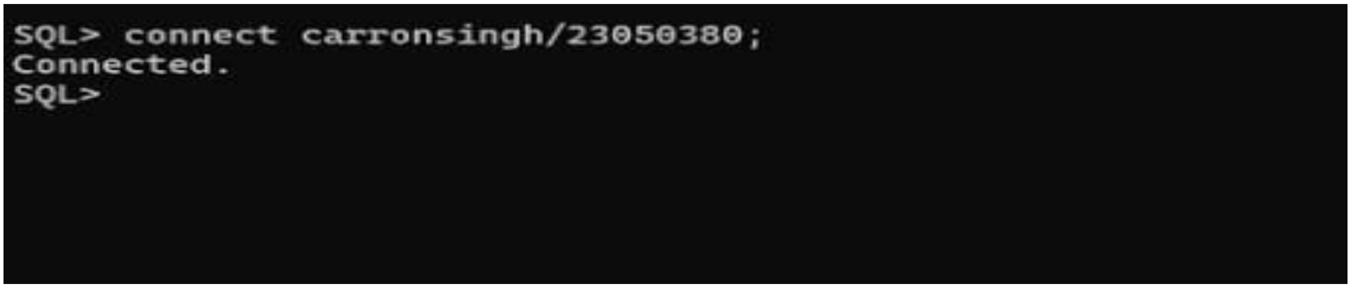
5.1 Creating a user with my name and password as my London Met Id assign.



Run SQL Command Line

```
SQL*Plus: Release 11.2.0.2.0 Production on Thu Jul 17 22:11:43 2025

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

SQL> connect system ;
Enter password:
ERROR:
ORA-28002: the password will expire within 7 days


Connected.
SQL> CREATE USER carronsingh IDENTIFIED BY 23050380;

User created.

SQL> GRANT CONNECT , RESOURCE TO carronsingh;

Grant succeeded.

SQL>
```

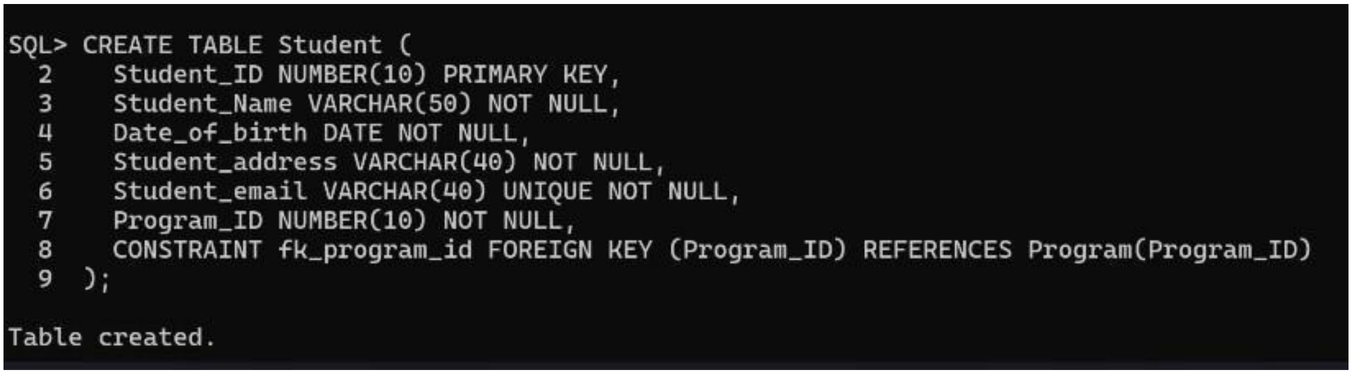
Figure 4 London Met Id assign.



```
SQL> connect carronsingh/23050380;
Connected.
SQL>
```

Figure 5 SQL Connected

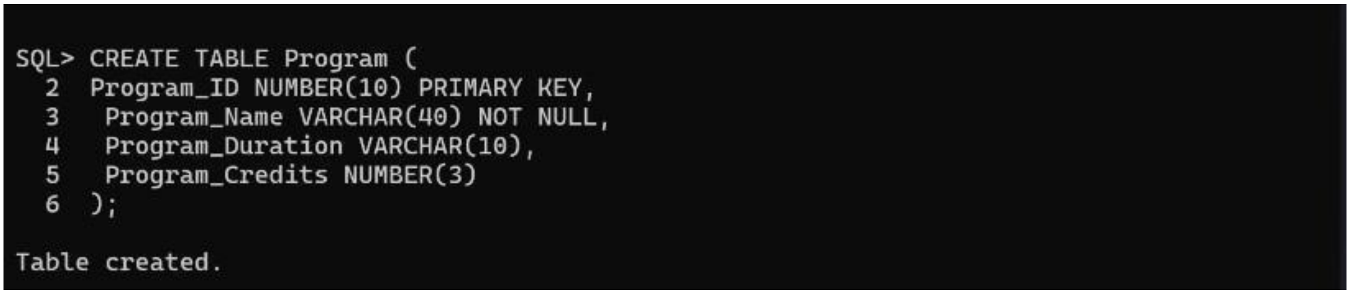
5.2 Creating Student table.



```
SQL> CREATE TABLE Student (
       Student_ID NUMBER(10) PRIMARY KEY,
       Student_Name VARCHAR(50) NOT NULL,
       Date_of_birth DATE NOT NULL,
       Student_address VARCHAR(40) NOT NULL,
       Student_email VARCHAR(40) UNIQUE NOT NULL,
       Program_ID NUMBER(10) NOT NULL,
       CONSTRAINT fk_program_id FOREIGN KEY (Program_ID) REFERENCES Program(Program_ID)
     );

Table created.
```

Figure 6 Creating Student table.

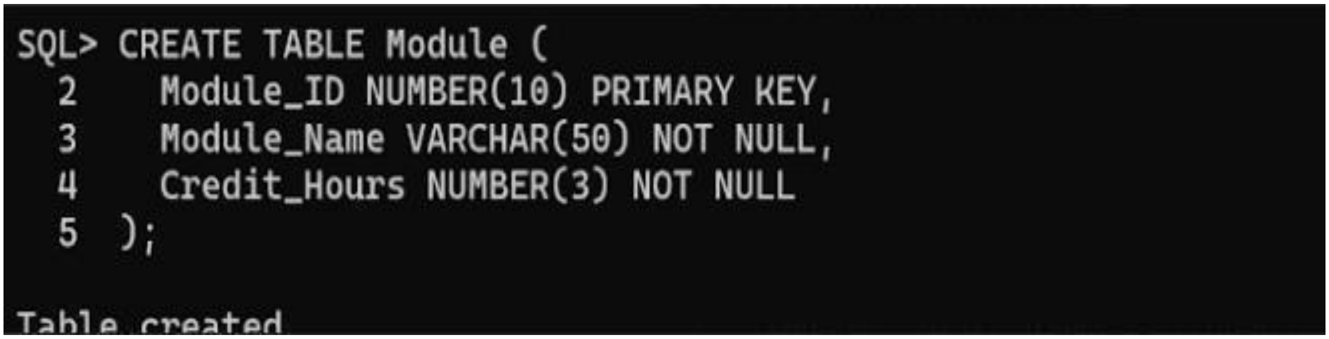
5.3 Creating Program table.



```
SQL> CREATE TABLE Program (
     Program_ID NUMBER(10) PRIMARY KEY,
      Program_Name VARCHAR(40) NOT NULL,
      Program_Duration VARCHAR(10),
      Program_Credits NUMBER(3)
     );

Table created.
```

Figure 7 Creating Program table

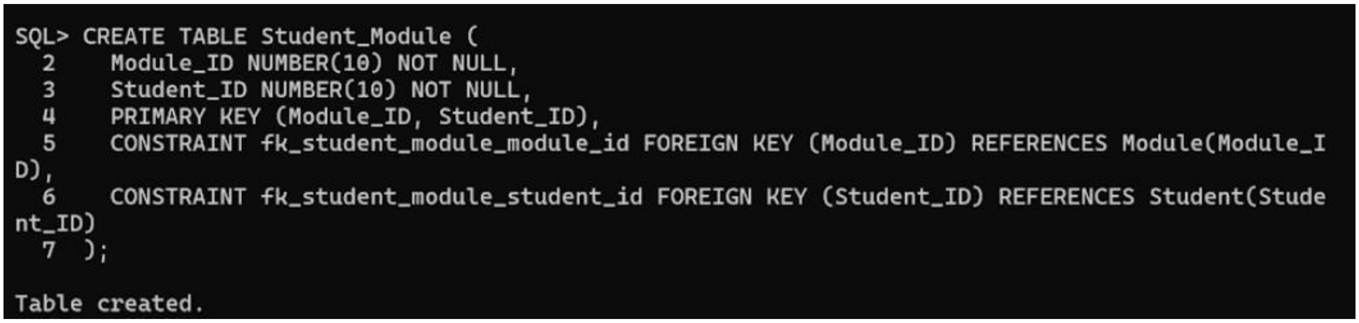
5.4 Creating Module table.



```
SQL> CREATE TABLE Module (
       Module_ID NUMBER(10) PRIMARY KEY,
       Module_Name VARCHAR(50) NOT NULL,
       Credit_Hours NUMBER(3) NOT NULL
     );

Table created
```

Figure 8 Creating Module table.

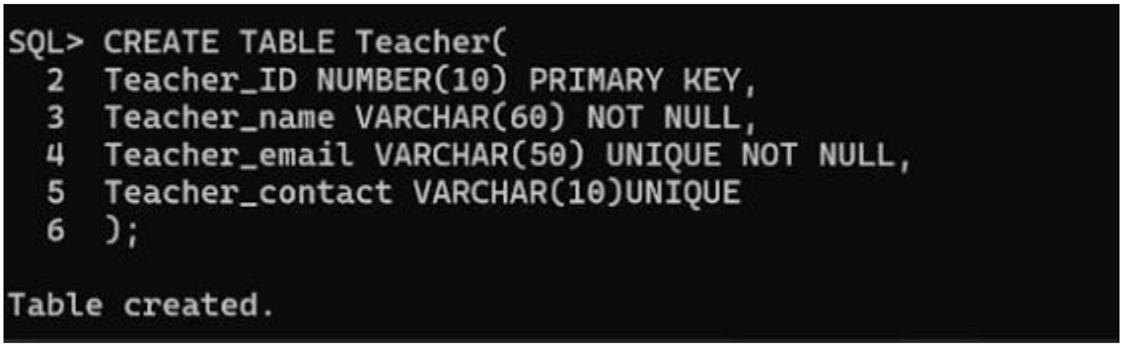
5.5 Creating Student Module table.



```
SQL> CREATE TABLE Student_Module (
       Module_ID NUMBER(10) NOT NULL,
       Student_ID NUMBER(10) NOT NULL,
       PRIMARY KEY (Module_ID, Student_ID),
       CONSTRAINT fk_student_module_module_id FOREIGN KEY (Module_ID) REFERENCES Module(Module_I
D),
       CONSTRAINT fk_student_module_student_id FOREIGN KEY (Student_ID) REFERENCES Student(Stude
nt_ID)
     );

Table created.
```

Figure 9 Creating Student Module table.

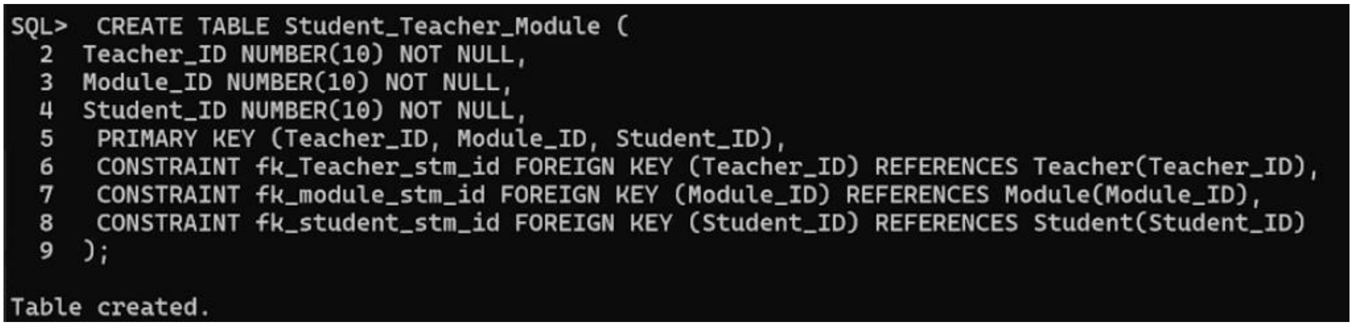
5.6 Creating Teacher table.



```
SQL> CREATE TABLE Teacher(
Teacher_ID NUMBER(10) PRIMARY KEY,
Teacher_name VARCHAR(60) NOT NULL,
Teacher_email VARCHAR(50) UNIQUE NOT NULL,
Teacher_contact VARCHAR(10)UNIQUE
);

Table created.
```

Figure 10 Creating Teacher table.

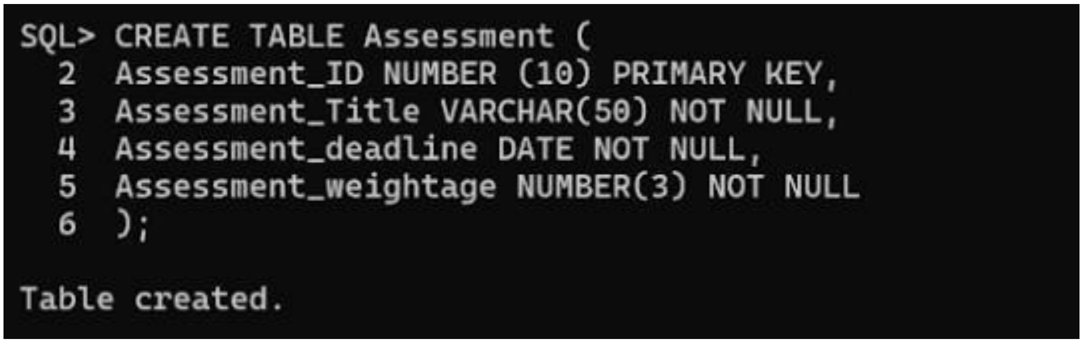
5.7 Creating Student_Teacher_Module table.



```
SQL>  CREATE TABLE Student_Teacher_Module (
Teacher_ID NUMBER(10) NOT NULL,
Module_ID NUMBER(10) NOT NULL,
Student_ID NUMBER(10) NOT NULL,
 PRIMARY KEY (Teacher_ID, Module_ID, Student_ID),
 CONSTRAINT fk_Teacher_stm_id FOREIGN KEY (Teacher_ID) REFERENCES Teacher(Teacher_ID),
 CONSTRAINT fk_module_stm_id FOREIGN KEY (Module_ID) REFERENCES Module(Module_ID),
 CONSTRAINT fk_student_stm_id FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
);

Table created.
```

Figure 11 Creating Student_Teacher_Module table.

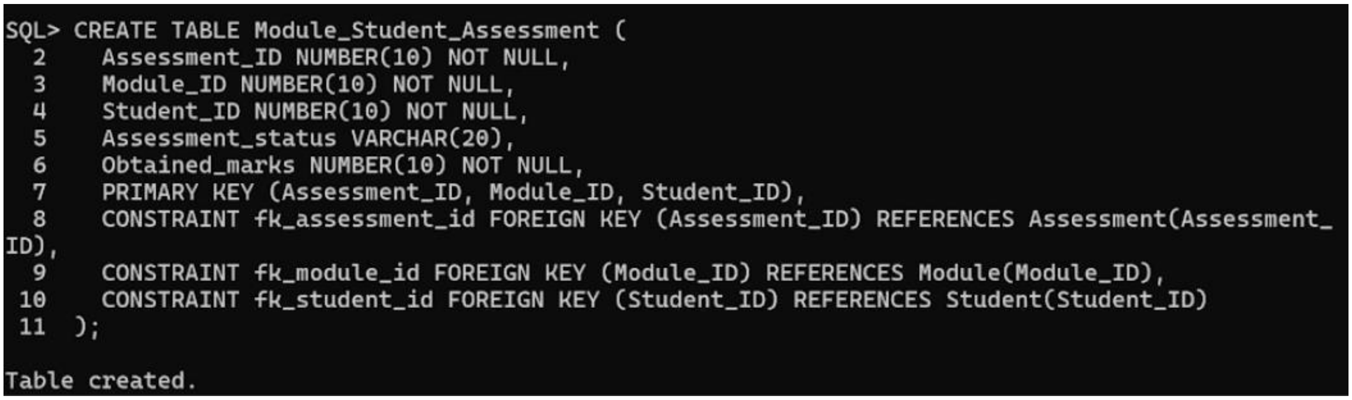
5.8 Creating Assessment table.



```
SQL> CREATE TABLE Assessment (
  2  Assessment_ID NUMBER (10) PRIMARY KEY,
  3  Assessment_Title VARCHAR(50) NOT NULL,
  4  Assessment_deadline DATE NOT NULL,
  5  Assessment_weightage NUMBER(3) NOT NULL
  6  );

Table created.
```

Figure 12 Creating Assessment table.

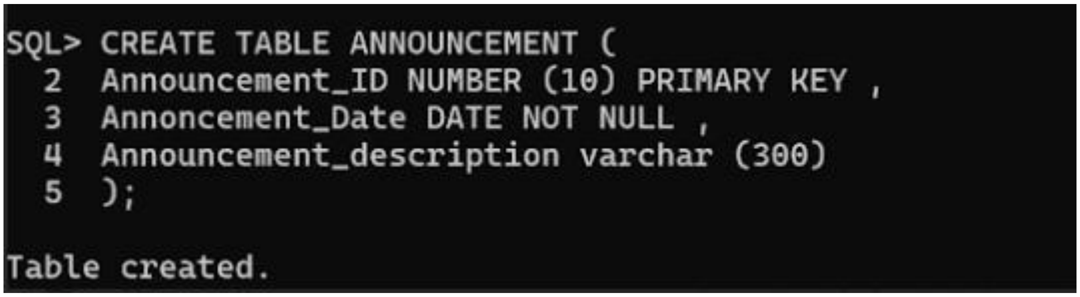
5.9 Creating Module_Student _Assessment table.



```
SQL> CREATE TABLE Module_Student_Assessment (
  2    Assessment_ID NUMBER(10) NOT NULL,
  3    Module_ID NUMBER(10) NOT NULL,
  4    Student_ID NUMBER(10) NOT NULL,
  5    Assessment_status VARCHAR(20),
  6    Obtained_marks NUMBER(10) NOT NULL,
  7    PRIMARY KEY (Assessment_ID, Module_ID, Student_ID),
  8    CONSTRAINT fk_assessment_id FOREIGN KEY (Assessment_ID) REFERENCES Assessment(Assessment_
ID),
  9    CONSTRAINT fk_module_id FOREIGN KEY (Module_ID) REFERENCES Module(Module_ID),
 10    CONSTRAINT fk_student_id FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
 11  );

Table created.
```

Figure 13 Creating Module_Student _Assessment table.

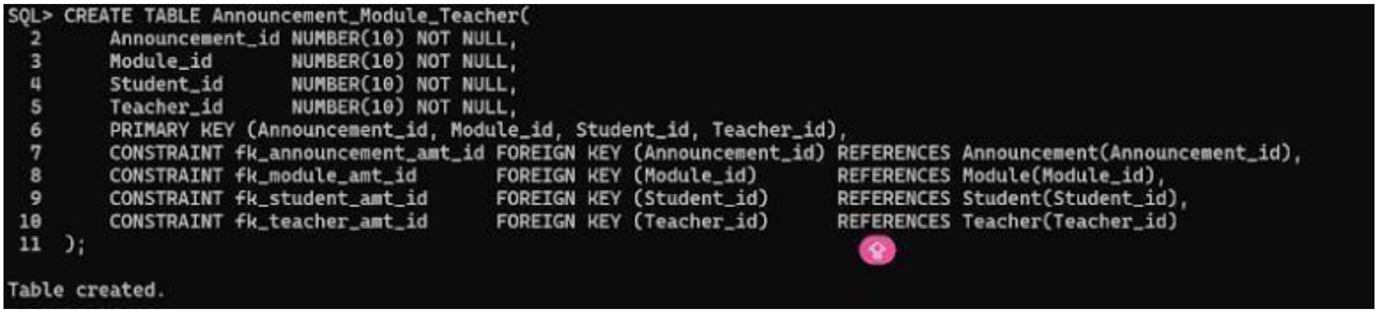
5.10 Creating Announcement table.



```
SQL> CREATE TABLE ANNOUNCEMENT (
     Announcement_ID NUMBER (10) PRIMARY KEY ,
     Annoncement_Date DATE NOT NULL ,
     Announcement_description varchar (300)
     );

Table created.
```

Figure 14 Creating Announcement table.

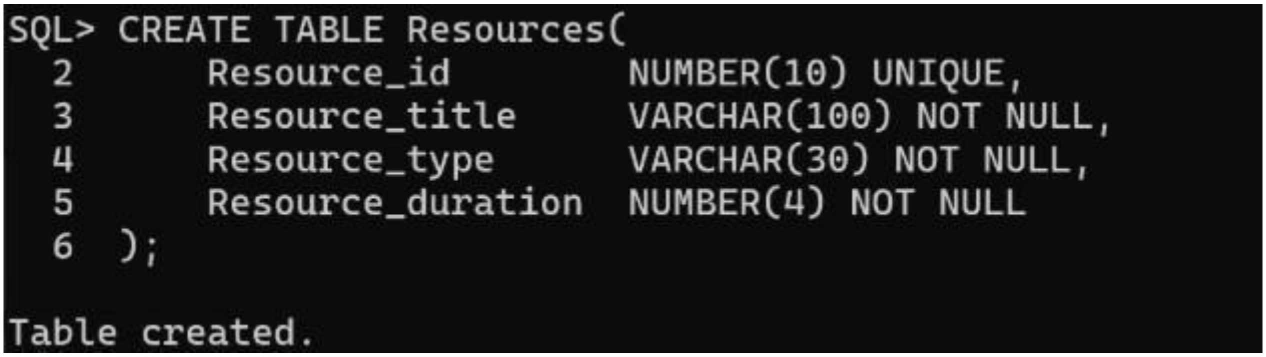
5.11 Creating Announcement_Module_Teacher table.



```
SQL> CREATE TABLE Announcement_Module_Teacher(
         Announcement_id NUMBER(10) NOT NULL,
         Module_id       NUMBER(10) NOT NULL,
         Student_id      NUMBER(10) NOT NULL,
         Teacher_id      NUMBER(10) NOT NULL,
         PRIMARY KEY (Announcement_id, Module_id, Student_id, Teacher_id),
         CONSTRAINT fk_announcement_amt_id FOREIGN KEY (Announcement_id) REFERENCES Announcement(Announcement_id),
         CONSTRAINT fk_module_amt_id       FOREIGN KEY (Module_id)       REFERENCES Module(Module_id),
         CONSTRAINT fk_student_amt_id      FOREIGN KEY (Student_id)      REFERENCES Student(Student_id),
         CONSTRAINT fk_teacher_amt_id      FOREIGN KEY (Teacher_id)      REFERENCES Teacher(Teacher_id)
     );

Table created.
```

Figure 15 Creating Announcement_Module_Teacher table.

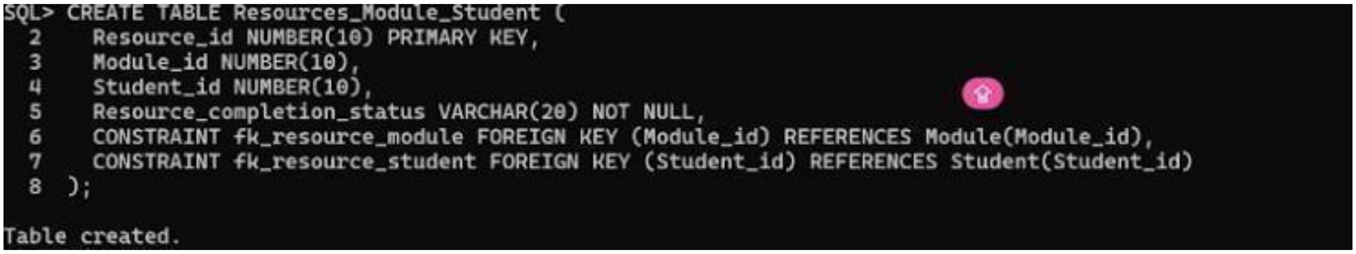
5.12 Creating Resources table.



```
SQL> CREATE TABLE Resources(
        Resource_id          NUMBER(10) UNIQUE,
        Resource_title       VARCHAR(100) NOT NULL,
        Resource_type        VARCHAR(30) NOT NULL,
        Resource_duration    NUMBER(4) NOT NULL
     );

Table created.
```

Figure 16 Creating Resources table.

5.13 Creating Resource _Module _Student table.

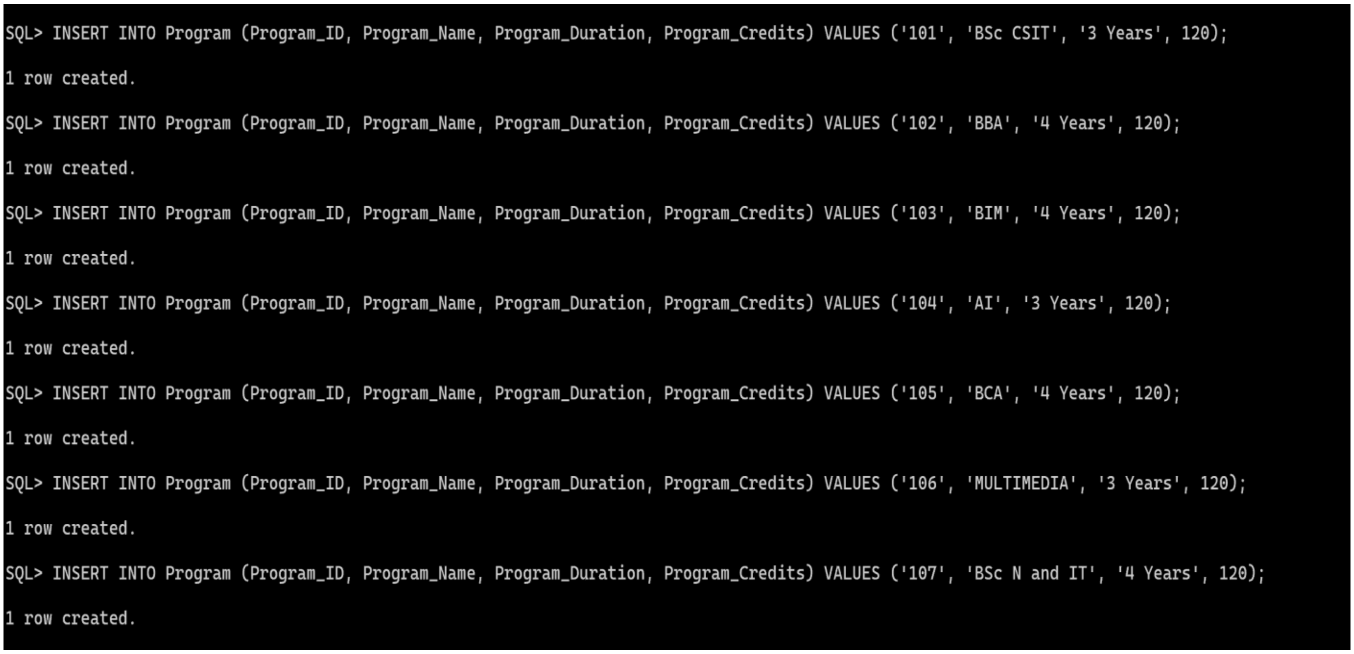


```
SQL> CREATE TABLE Resources_Module_Student (
       Resource_id NUMBER(10) PRIMARY KEY,
       Module_id NUMBER(10),
       Student_id NUMBER(10),
       Resource_completion_status VARCHAR(20) NOT NULL,
       CONSTRAINT fk_resource_module FOREIGN KEY (Module_id) REFERENCES Module(Module_id),
       CONSTRAINT fk_resource_student FOREIGN KEY (Student_id) REFERENCES Student(Student_id)
     );

Table created.
```

Figure 17 Creating Resource _Module _Student table.

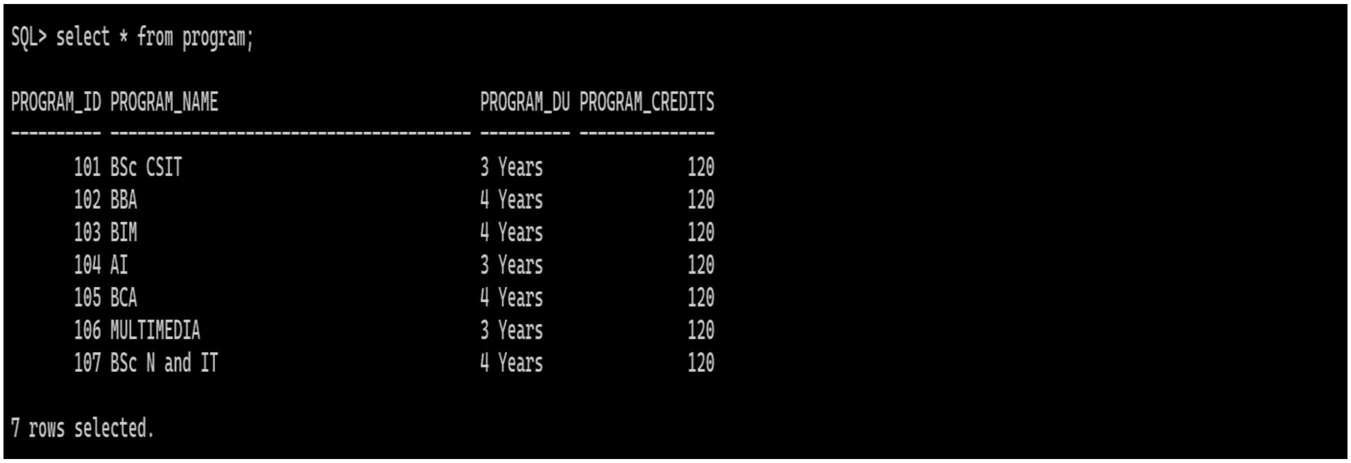
6. Inserting and Viewing data

6.1. Use “insert” command to insert data in program table.



```
SQL> INSERT INTO Program (Program_ID, Program_Name, Program_Duration, Program_Credits) VALUES ('101', 'BSc CSIT', '3 Years', 120);

1 row created.

SQL> INSERT INTO Program (Program_ID, Program_Name, Program_Duration, Program_Credits) VALUES ('102', 'BBA', '4 Years', 120);

1 row created.

SQL> INSERT INTO Program (Program_ID, Program_Name, Program_Duration, Program_Credits) VALUES ('103', 'BIM', '4 Years', 120);

1 row created.

SQL> INSERT INTO Program (Program_ID, Program_Name, Program_Duration, Program_Credits) VALUES ('104', 'AI', '3 Years', 120);

1 row created.

SQL> INSERT INTO Program (Program_ID, Program_Name, Program_Duration, Program_Credits) VALUES ('105', 'BCA', '4 Years', 120);

1 row created.

SQL> INSERT INTO Program (Program_ID, Program_Name, Program_Duration, Program_Credits) VALUES ('106', 'MULTIMEDIA', '3 Years', 120);

1 row created.

SQL> INSERT INTO Program (Program_ID, Program_Name, Program_Duration, Program_Credits) VALUES ('107', 'BSc N and IT', '4 Years', 120);

1 row created.
```

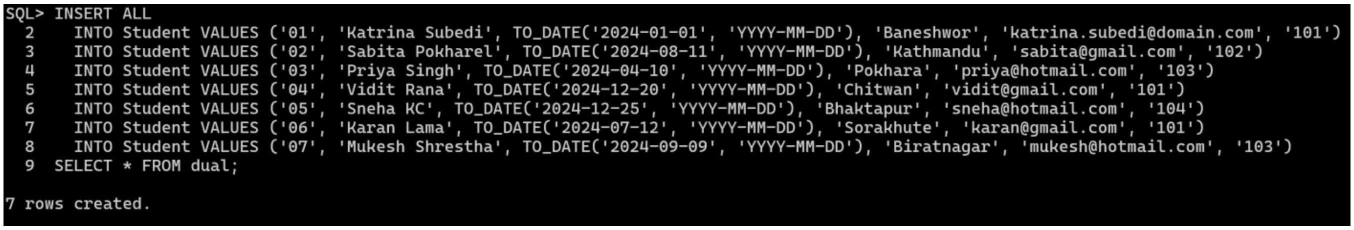
Figure 18 Use “insert” command to insert data in program table.



```
SQL> select * from program;

PROGRAM_ID PROGRAM_NAME                             PROGRAM_DU PROGRAM_CREDITS
---------- ---------------------------------------- ---------- ---------------
       101 BSc CSIT                                 3 Years                120
       102 BBA                                      4 Years                120
       103 BIM                                      4 Years                120
       104 AI                                       3 Years                120
       105 BCA                                      4 Years                120
       106 MULTIMEDIA                               3 Years                120
       107 BSc N and IT                             4 Years                120

7 rows selected.
```

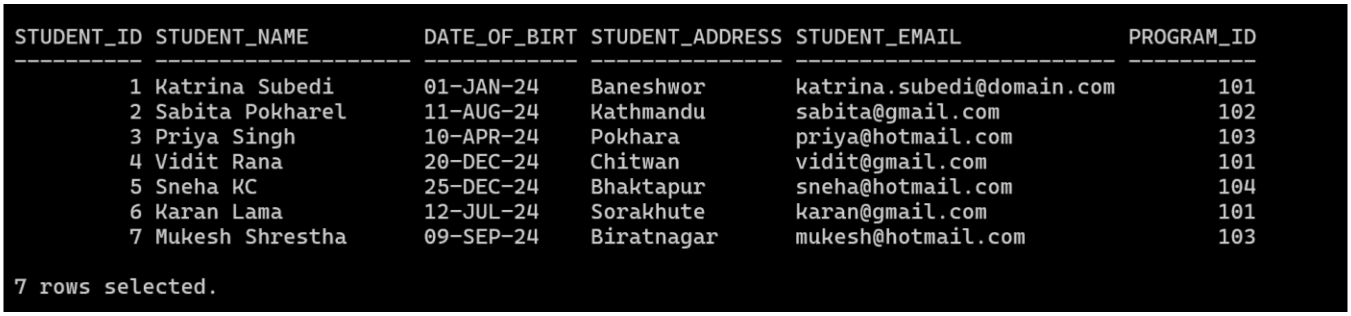
Figure 19 Viewing contents of program table.

6.2 insert" command to insert data in student table:



```
SQL> INSERT ALL
    INTO Student VALUES ('01', 'Katrina Subedi', TO_DATE('2024-01-01', 'YYYY-MM-DD'), 'Baneshwor', 'katrina.subedi@domain.com', '101')
    INTO Student VALUES ('02', 'Sabita Pokharel', TO_DATE('2024-08-11', 'YYYY-MM-DD'), 'Kathmandu', 'sabita@gmail.com', '102')
    INTO Student VALUES ('03', 'Priya Singh', TO_DATE('2024-04-10', 'YYYY-MM-DD'), 'Pokhara', 'priya@hotmail.com', '103')
    INTO Student VALUES ('04', 'Vidit Rana', TO_DATE('2024-12-20', 'YYYY-MM-DD'), 'Chitwan', 'vidit@gmail.com', '101')
    INTO Student VALUES ('05', 'Sneha KC', TO_DATE('2024-12-25', 'YYYY-MM-DD'), 'Bhaktapur', 'sneha@hotmail.com', '104')
    INTO Student VALUES ('06', 'Karan Lama', TO_DATE('2024-07-12', 'YYYY-MM-DD'), 'Sorakhute', 'karan@gmail.com', '101')
    INTO Student VALUES ('07', 'Mukesh Shrestha', TO_DATE('2024-09-09', 'YYYY-MM-DD'), 'Biratnagar', 'mukesh@hotmail.com', '103')
  SELECT * FROM dual;

7 rows created.
```

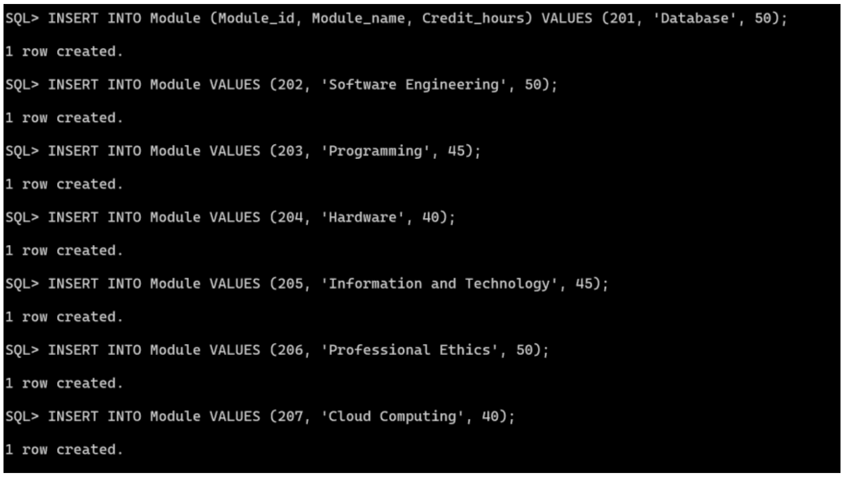
Figure 20 insert" command to insert data in student table:



```
STUDENT_ID STUDENT_NAME         DATE_OF_BIRT STUDENT_ADDRESS STUDENT_EMAIL             PROGRAM_ID
---------- -------------------- ------------ --------------- ------------------------- ----------
         1 Katrina Subedi       01-JAN-24    Baneshwor       katrina.subedi@domain.com        101
         2 Sabita Pokharel      11-AUG-24    Kathmandu       sabita@gmail.com                 102
         3 Priya Singh          10-APR-24    Pokhara         priya@hotmail.com                103
         4 Vidit Rana           20-DEC-24    Chitwan         vidit@gmail.com                  101
         5 Sneha KC             25-DEC-24    Bhaktapur       sneha@hotmail.com                104
         6 Karan Lama           12-JUL-24    Sorakhute       karan@gmail.com                  101
         7 Mukesh Shrestha      09-SEP-24    Biratnagar      mukesh@hotmail.com               103

7 rows selected.
```

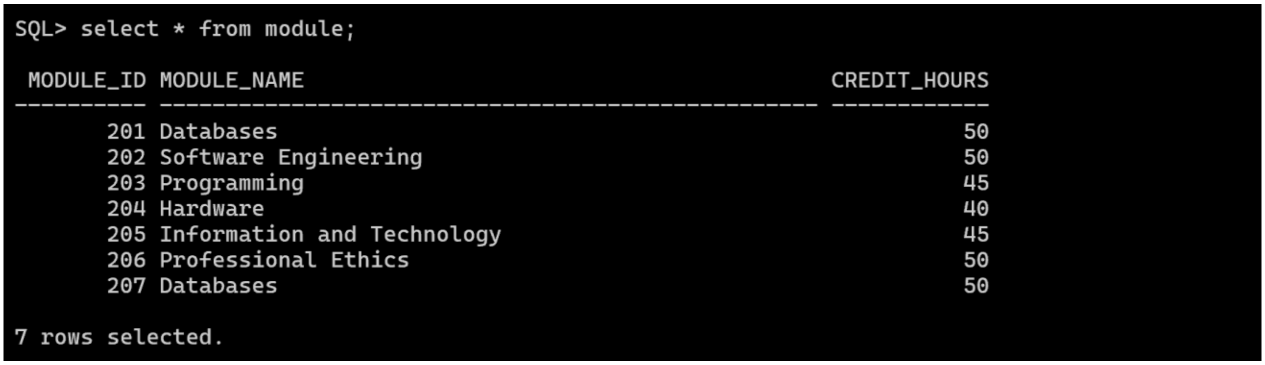
Figure 21 Viewing contents of student table

6.3 Use “insert” command to insert data in Module table:



```
SQL> INSERT INTO Module (Module_id, Module_name, Credit_hours) VALUES (201, 'Database', 50);

1 row created.

SQL> INSERT INTO Module VALUES (202, 'Software Engineering', 50);

1 row created.

SQL> INSERT INTO Module VALUES (203, 'Programming', 45);

1 row created.

SQL> INSERT INTO Module VALUES (204, 'Hardware', 40);

1 row created.

SQL> INSERT INTO Module VALUES (205, 'Information and Technology', 45);

1 row created.

SQL> INSERT INTO Module VALUES (206, 'Professional Ethics', 50);

1 row created.

SQL> INSERT INTO Module VALUES (207, 'Cloud Computing', 40);

1 row created.
```

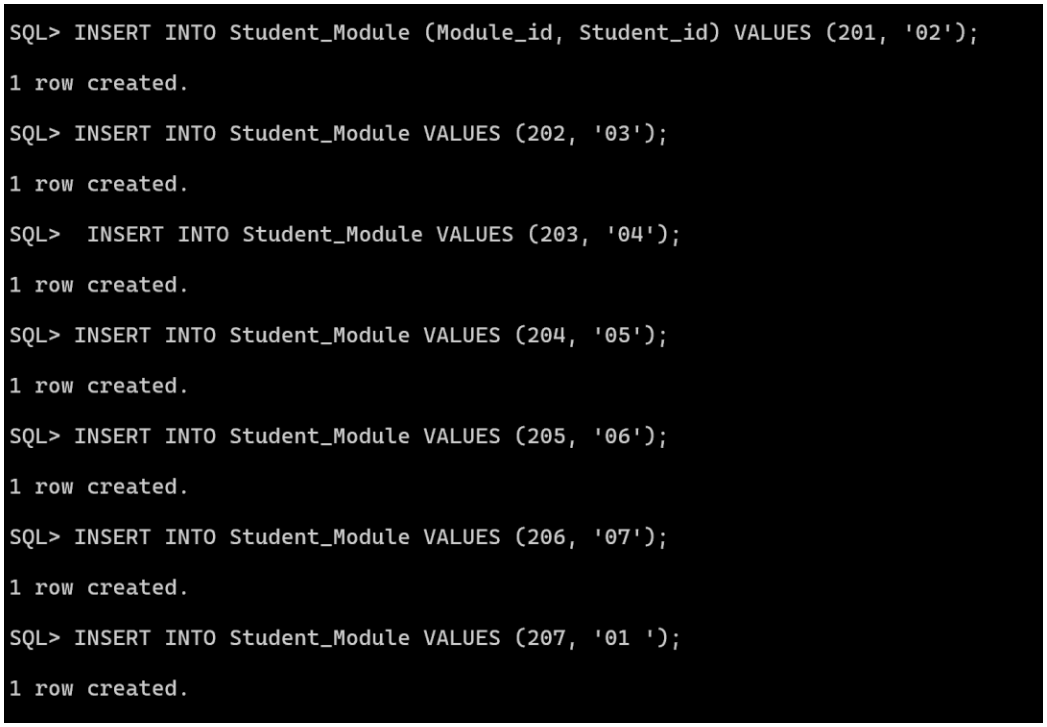
Figure 22 Module table.



```
SQL> select * from module;

 MODULE_ID MODULE_NAME                                        CREDIT_HOURS
---------- -------------------------------------------------- ------------
       201 Databases                                                    50
       202 Software Engineering                                         50
       203 Programming                                                  45
       204 Hardware                                                     40
       205 Information and Technology                                   45
       206 Professional Ethics                                          50
       207 Databases                                                    50

7 rows selected.
```

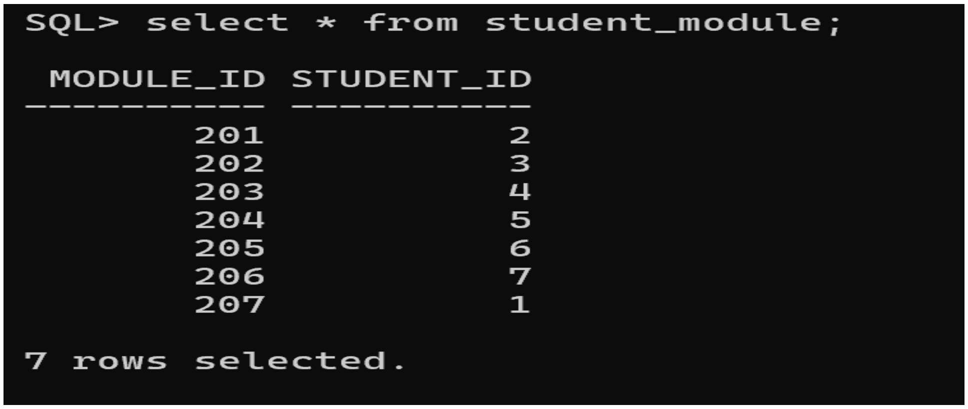
Figure 23 Viewing contents of module.

6.4 Use “insert” command to insert data in Student_ Module table:



```
SQL> INSERT INTO Student_Module (Module_id, Student_id) VALUES (201, '02');

1 row created.

SQL> INSERT INTO Student_Module VALUES (202, '03');

1 row created.

SQL>  INSERT INTO Student_Module VALUES (203, '04');

1 row created.

SQL> INSERT INTO Student_Module VALUES (204, '05');

1 row created.

SQL> INSERT INTO Student_Module VALUES (205, '06');

1 row created.

SQL> INSERT INTO Student_Module VALUES (206, '07');

1 row created.

SQL> INSERT INTO Student_Module VALUES (207, '01 ');

1 row created.
```

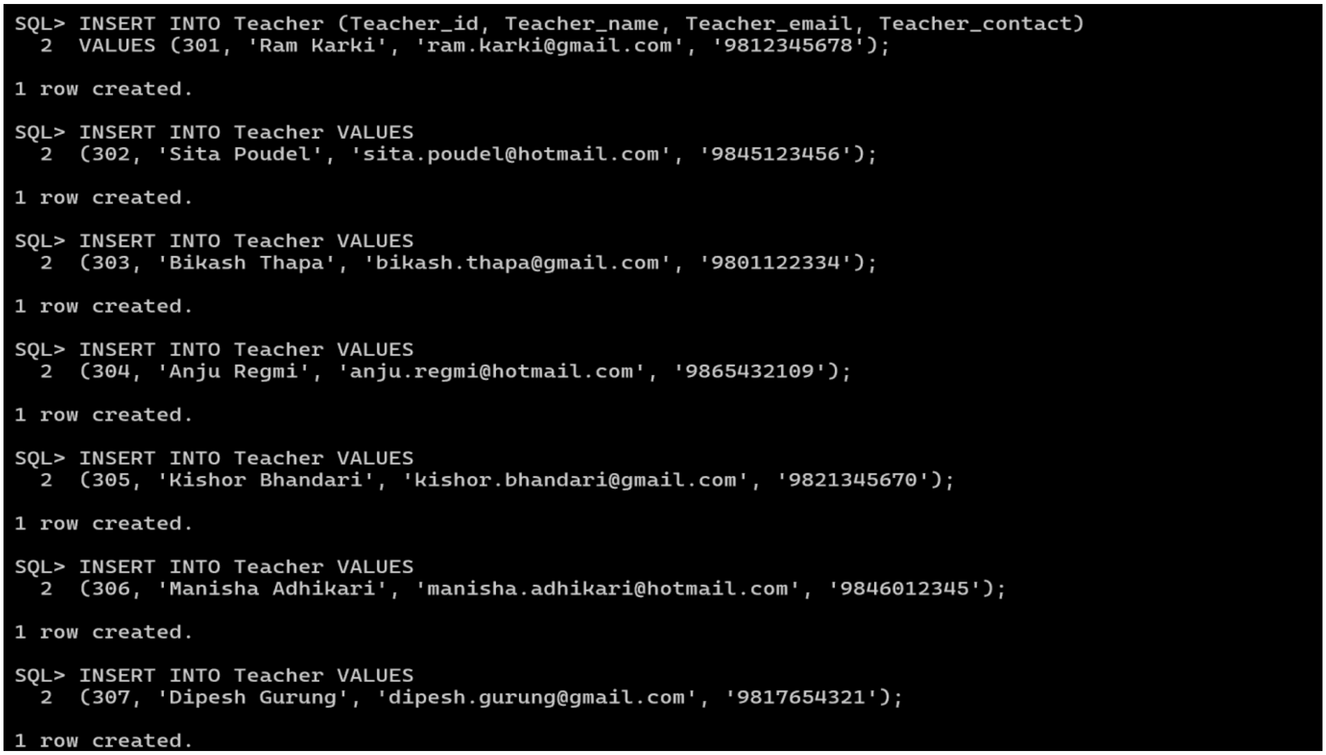
Figure 24 “insert” command to insert data in Student_ Module table:



```
SQL> select * from student_module;

 MODULE_ID STUDENT_ID
---------- ----------
       201          2
       202          3
       203          4
       204          5
       205          6
       206          7
       207          1

7 rows selected.
```

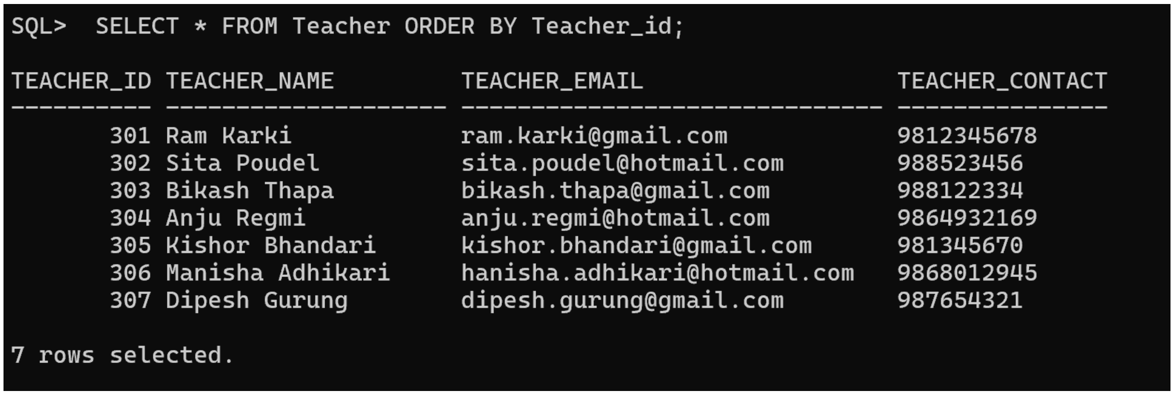
Figure 25 viewing Student_ Module table.

6.5 Use “insert” command to insert data in Teacher table:



```
SQL> INSERT INTO Teacher (Teacher_id, Teacher_name, Teacher_email, Teacher_contact)
  2  VALUES (301, 'Ram Karki', 'ram.karki@gmail.com', '9812345678');

1 row created.

SQL> INSERT INTO Teacher VALUES
  2  (302, 'Sita Poudel', 'sita.poudel@hotmail.com', '9845123456');

1 row created.

SQL> INSERT INTO Teacher VALUES
  2  (303, 'Bikash Thapa', 'bikash.thapa@gmail.com', '9801122334');

1 row created.

SQL> INSERT INTO Teacher VALUES
  2  (304, 'Anju Regmi', 'anju.regmi@hotmail.com', '9865432109');

1 row created.

SQL> INSERT INTO Teacher VALUES
  2  (305, 'Kishor Bhandari', 'kishor.bhandari@gmail.com', '9821345670');

1 row created.

SQL> INSERT INTO Teacher VALUES
  2  (306, 'Manisha Adhikari', 'manisha.adhikari@hotmail.com', '9846012345');

1 row created.

SQL> INSERT INTO Teacher VALUES
  2  (307, 'Dipesh Gurung', 'dipesh.gurung@gmail.com', '9817654321');

1 row created.
```

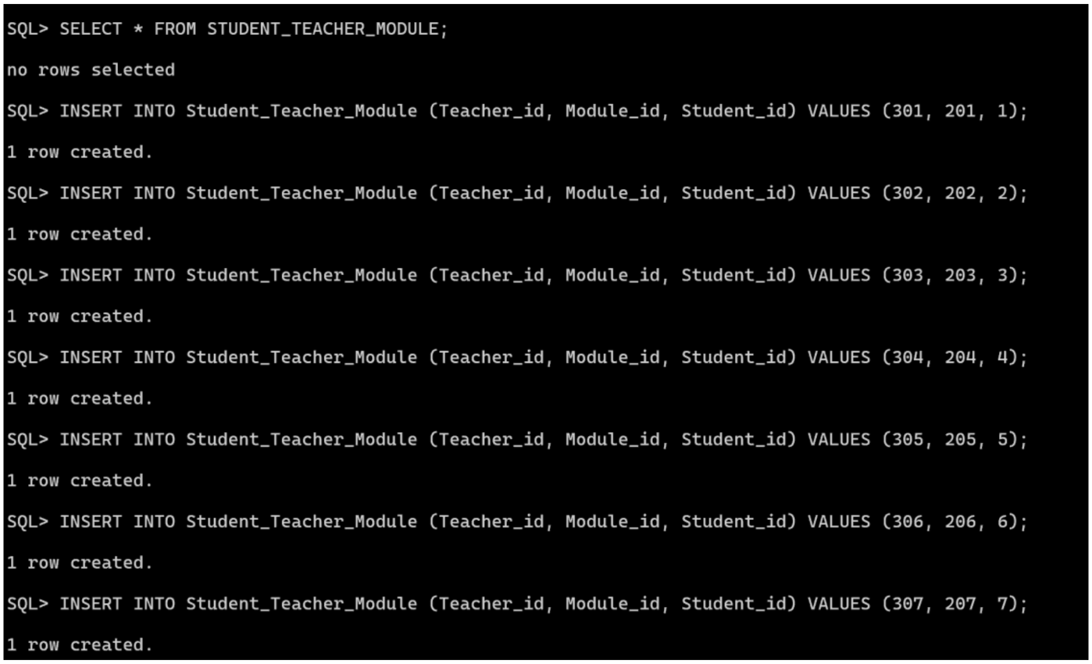
Figure 26 “insert” command to insert data in Teacher table:



```
SQL>  SELECT * FROM Teacher ORDER BY Teacher_id;

TEACHER_ID TEACHER_NAME         TEACHER_EMAIL                  TEACHER_CONTACT
---------- -------------------- ------------------------------ ---------------
       301 Ram Karki            ram.karki@gmail.com            9812345678
       302 Sita Poudel          sita.poudel@hotmail.com        988523456
       303 Bikash Thapa         bikash.thapa@gmail.com         988122334
       304 Anju Regmi           anju.regmi@hotmail.com         9864932169
       305 Kishor Bhandari      kishor.bhandari@gmail.com      981345670
       306 Manisha Adhikari     hanisha.adhikari@hotmail.com   9868012945
       307 Dipesh Gurung        dipesh.gurung@gmail.com        987654321

7 rows selected.
```

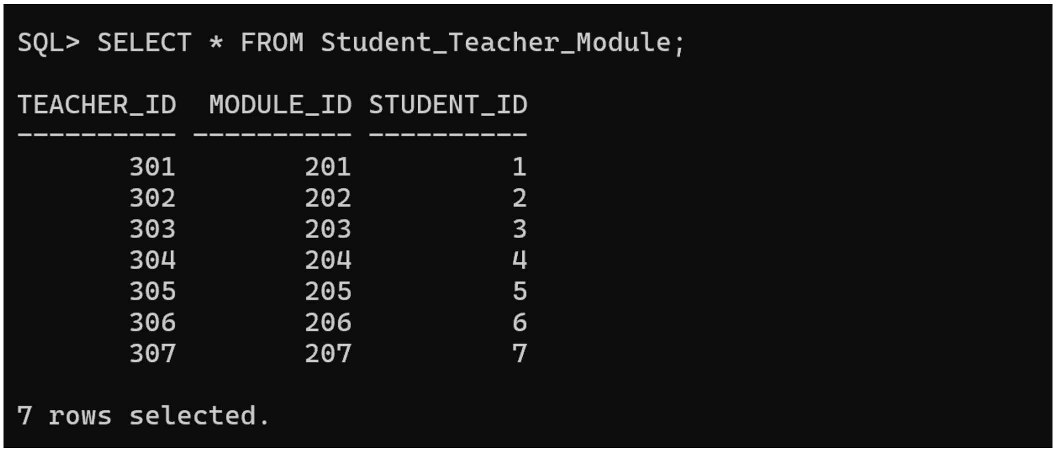
Figure 27viewing Teacher table.

6.6 Use “insert” command to insert data in Student_Teacher_Module table:



```
SQL> SELECT * FROM STUDENT_TEACHER_MODULE;

no rows selected

SQL> INSERT INTO Student_Teacher_Module (Teacher_id, Module_id, Student_id) VALUES (301, 201, 1);

1 row created.

SQL> INSERT INTO Student_Teacher_Module (Teacher_id, Module_id, Student_id) VALUES (302, 202, 2);

1 row created.

SQL> INSERT INTO Student_Teacher_Module (Teacher_id, Module_id, Student_id) VALUES (303, 203, 3);

1 row created.

SQL> INSERT INTO Student_Teacher_Module (Teacher_id, Module_id, Student_id) VALUES (304, 204, 4);

1 row created.

SQL> INSERT INTO Student_Teacher_Module (Teacher_id, Module_id, Student_id) VALUES (305, 205, 5);

1 row created.

SQL> INSERT INTO Student_Teacher_Module (Teacher_id, Module_id, Student_id) VALUES (306, 206, 6);

1 row created.

SQL> INSERT INTO Student_Teacher_Module (Teacher_id, Module_id, Student_id) VALUES (307, 207, 7);

1 row created.
```

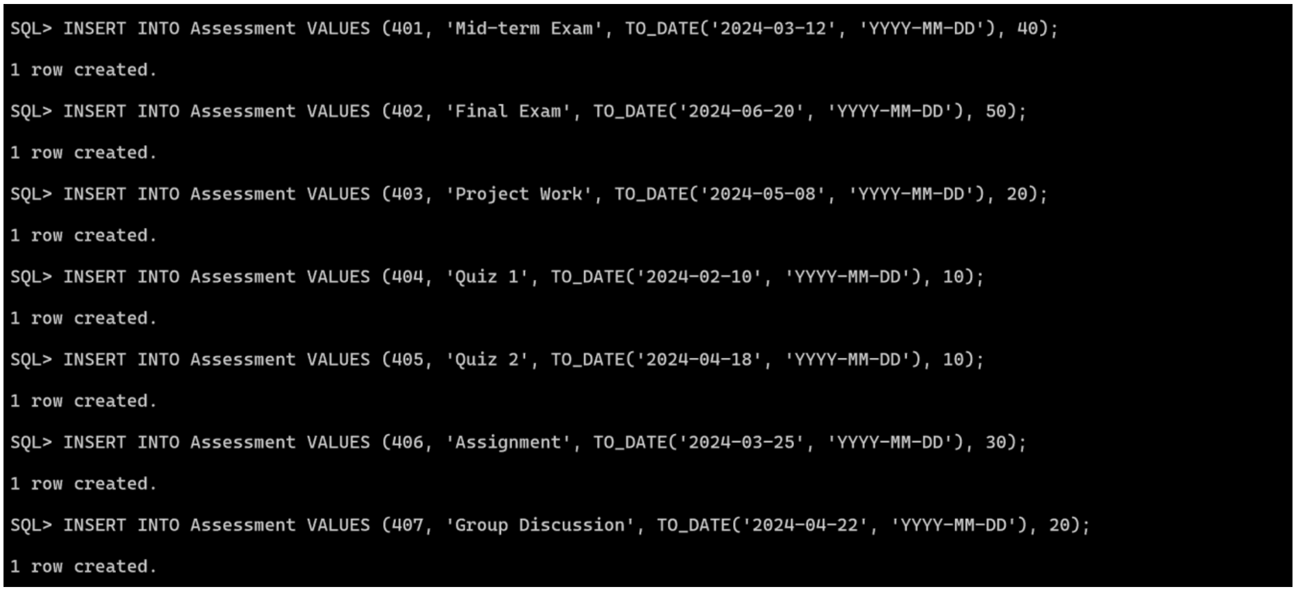
Figure 28 “insert” command to insert data in Student_Teacher_Module table.



```
SQL> SELECT * FROM Student_Teacher_Module;

TEACHER_ID  MODULE_ID STUDENT_ID
---------- ---------- ----------
       301        201          1
       302        202          2
       303        203          3
       304        204          4
       305        205          5
       306        206          6
       307        207          7

7 rows selected.
```

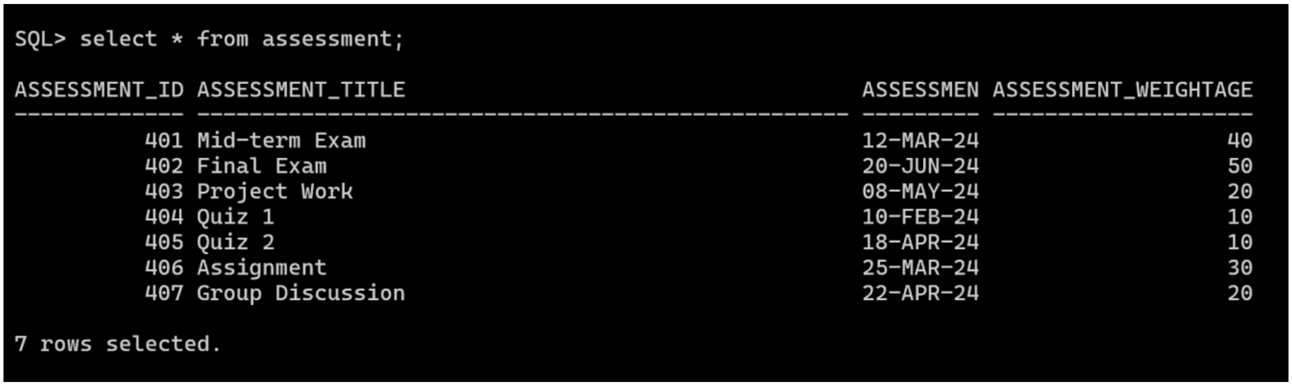
Figure 29 viewing Student_Teacher_Module table.

6.7 Use “insert” command to insert data in Assessment table:



```
SQL> INSERT INTO Assessment VALUES (401, 'Mid-term Exam', TO_DATE('2024-03-12', 'YYYY-MM-DD'), 40);

1 row created.

SQL> INSERT INTO Assessment VALUES (402, 'Final Exam', TO_DATE('2024-06-20', 'YYYY-MM-DD'), 50);

1 row created.

SQL> INSERT INTO Assessment VALUES (403, 'Project Work', TO_DATE('2024-05-08', 'YYYY-MM-DD'), 20);

1 row created.

SQL> INSERT INTO Assessment VALUES (404, 'Quiz 1', TO_DATE('2024-02-10', 'YYYY-MM-DD'), 10);

1 row created.

SQL> INSERT INTO Assessment VALUES (405, 'Quiz 2', TO_DATE('2024-04-18', 'YYYY-MM-DD'), 10);

1 row created.

SQL> INSERT INTO Assessment VALUES (406, 'Assignment', TO_DATE('2024-03-25', 'YYYY-MM-DD'), 30);

1 row created.

SQL> INSERT INTO Assessment VALUES (407, 'Group Discussion', TO_DATE('2024-04-22', 'YYYY-MM-DD'), 20);

1 row created.
```

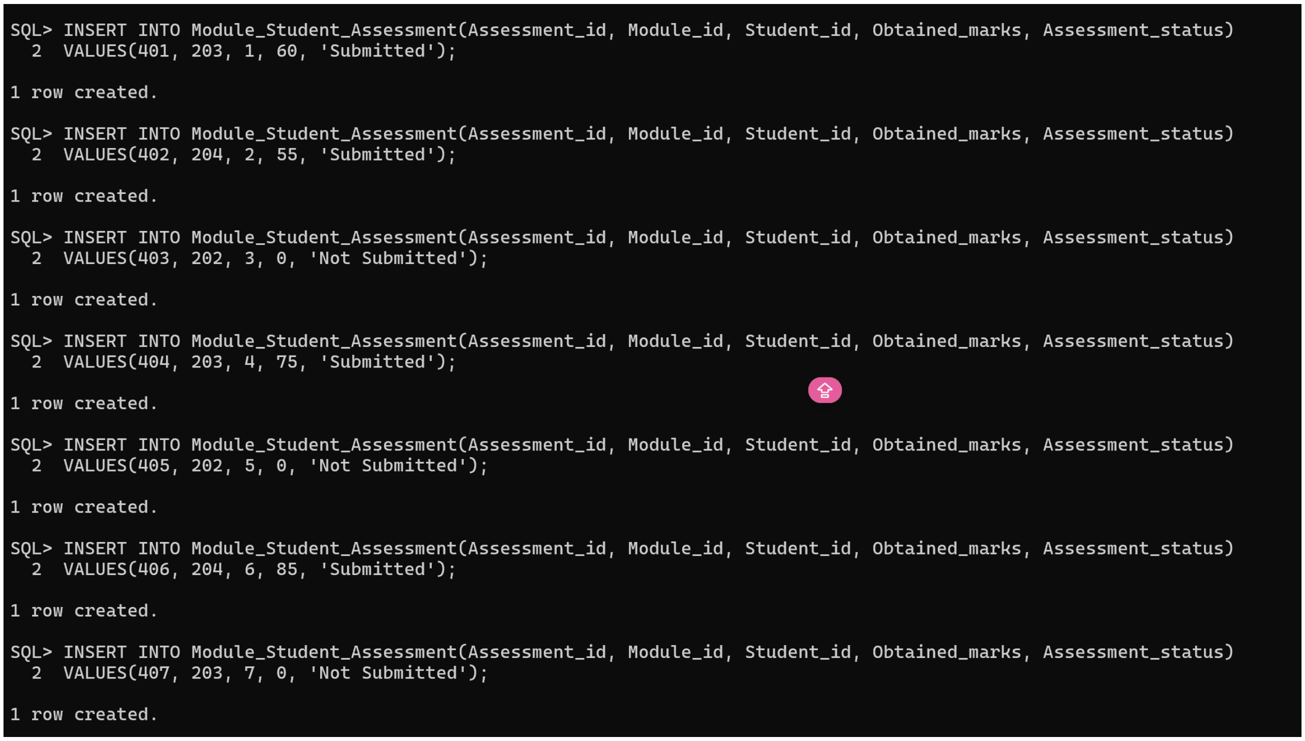
Figure 30 “insert” command to insert data in Assessment table:



```
SQL> select * from assessment;

ASSESSMENT_ID ASSESSMENT_TITLE                                   ASSESSMEN ASSESSMENT_WEIGHTAGE
------------- -------------------------------------------------- --------- --------------------
          401 Mid-term Exam                                      12-MAR-24                   40
          402 Final Exam                                         20-JUN-24                   50
          403 Project Work                                       08-MAY-24                   20
          404 Quiz 1                                             10-FEB-24                   10
          405 Quiz 2                                             18-APR-24                   10
          406 Assignment                                         25-MAR-24                   30
          407 Group Discussion                                   22-APR-24                   20

7 rows selected.
```

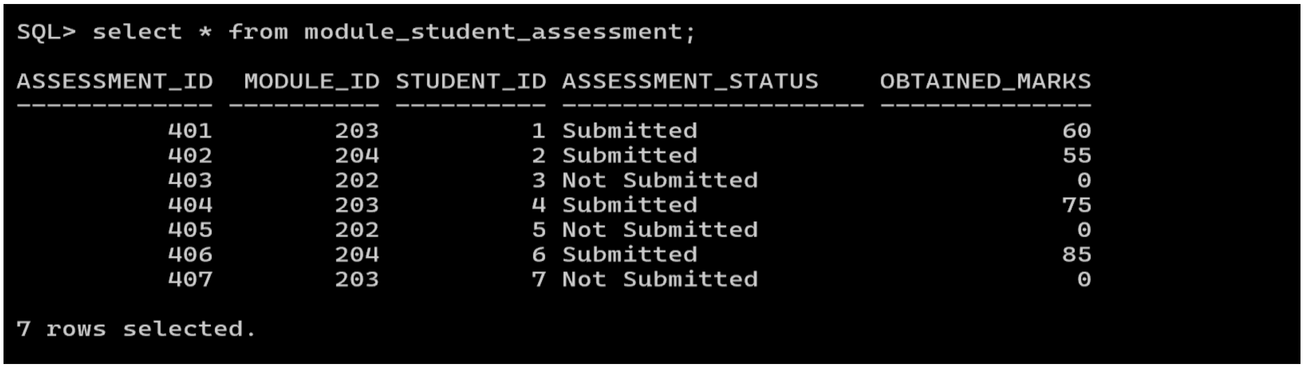
Figure 31 viewing Assessment table:

6.8 Use “insert” command to insert data in Module_Student_ Assessment table:



```
SQL> INSERT INTO Module_Student_Assessment(Assessment_id, Module_id, Student_id, Obtained_marks, Assessment_status)
  2  VALUES(401, 203, 1, 60, 'Submitted');

1 row created.

SQL> INSERT INTO Module_Student_Assessment(Assessment_id, Module_id, Student_id, Obtained_marks, Assessment_status)
  2  VALUES(402, 204, 2, 55, 'Submitted');

1 row created.

SQL> INSERT INTO Module_Student_Assessment(Assessment_id, Module_id, Student_id, Obtained_marks, Assessment_status)
  2  VALUES(403, 202, 3, 0, 'Not Submitted');

1 row created.

SQL> INSERT INTO Module_Student_Assessment(Assessment_id, Module_id, Student_id, Obtained_marks, Assessment_status)
  2  VALUES(404, 203, 4, 75, 'Submitted');

1 row created.

SQL> INSERT INTO Module_Student_Assessment(Assessment_id, Module_id, Student_id, Obtained_marks, Assessment_status)
  2  VALUES(405, 202, 5, 0, 'Not Submitted');

1 row created.

SQL> INSERT INTO Module_Student_Assessment(Assessment_id, Module_id, Student_id, Obtained_marks, Assessment_status)
  2  VALUES(406, 204, 6, 85, 'Submitted');

1 row created.

SQL> INSERT INTO Module_Student_Assessment(Assessment_id, Module_id, Student_id, Obtained_marks, Assessment_status)
  2  VALUES(407, 203, 7, 0, 'Not Submitted');

1 row created.
```

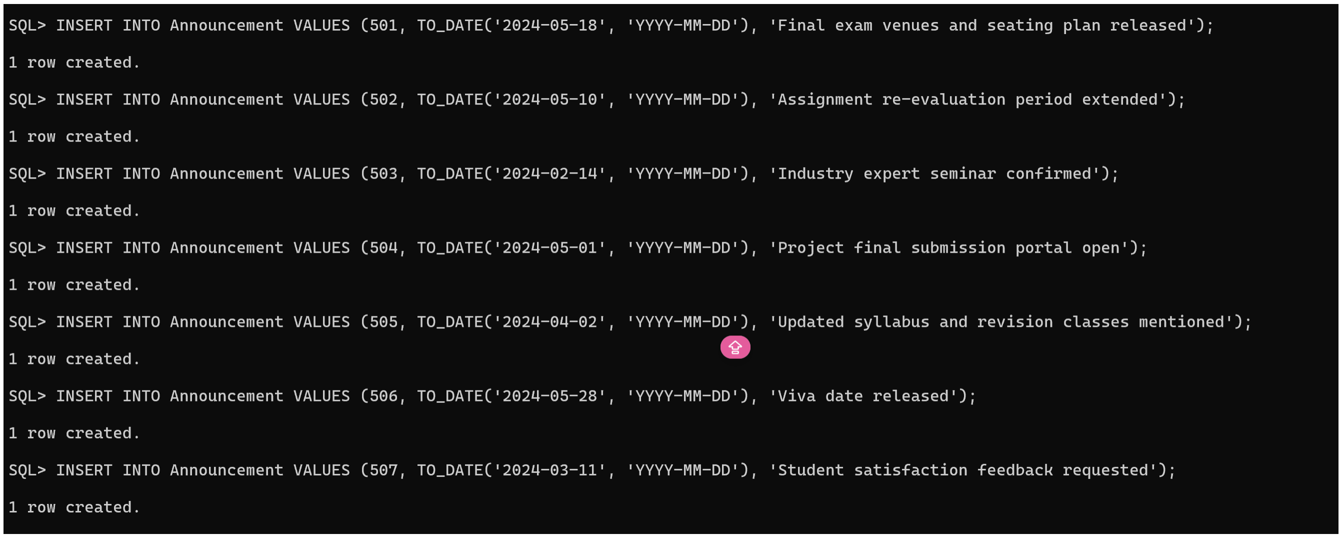
Figure 32 “insert” command to insert data in Module_Student_ Assessment table.



```
SQL> select * from module_student_assessment;

ASSESSMENT_ID  MODULE_ID STUDENT_ID ASSESSMENT_STATUS    OBTAINED_MARKS
------------- ---------- ---------- -------------------- --------------
          401        203          1 Submitted                        60
          402        204          2 Submitted                        55
          403        202          3 Not Submitted                     0
          404        203          4 Submitted                        75
          405        202          5 Not Submitted                     0
          406        204          6 Submitted                        85
          407        203          7 Not Submitted                     0

7 rows selected.
```

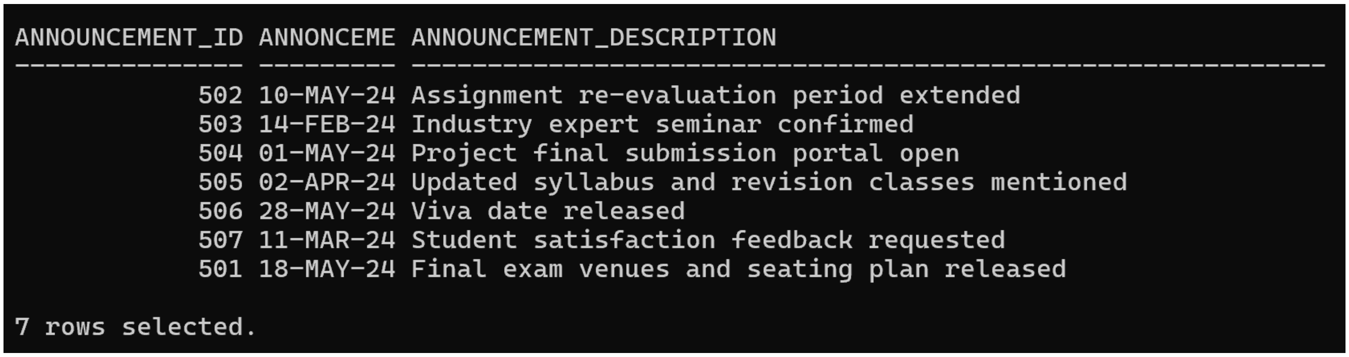
Figure 33 viewing Module_Student_ Assessment table.

6.9 Use “insert” command to insert data in Announcement table:



```
SQL> INSERT INTO Announcement VALUES (501, TO_DATE('2024-05-18', 'YYYY-MM-DD'), 'Final exam venues and seating plan released');

1 row created.

SQL> INSERT INTO Announcement VALUES (502, TO_DATE('2024-05-10', 'YYYY-MM-DD'), 'Assignment re-evaluation period extended');

1 row created.

SQL> INSERT INTO Announcement VALUES (503, TO_DATE('2024-02-14', 'YYYY-MM-DD'), 'Industry expert seminar confirmed');

1 row created.

SQL> INSERT INTO Announcement VALUES (504, TO_DATE('2024-05-01', 'YYYY-MM-DD'), 'Project final submission portal open');

1 row created.

SQL> INSERT INTO Announcement VALUES (505, TO_DATE('2024-04-02', 'YYYY-MM-DD'), 'Updated syllabus and revision classes mentioned');

1 row created.

SQL> INSERT INTO Announcement VALUES (506, TO_DATE('2024-05-28', 'YYYY-MM-DD'), 'Viva date released');

1 row created.

SQL> INSERT INTO Announcement VALUES (507, TO_DATE('2024-03-11', 'YYYY-MM-DD'), 'Student satisfaction feedback requested');

1 row created.
```

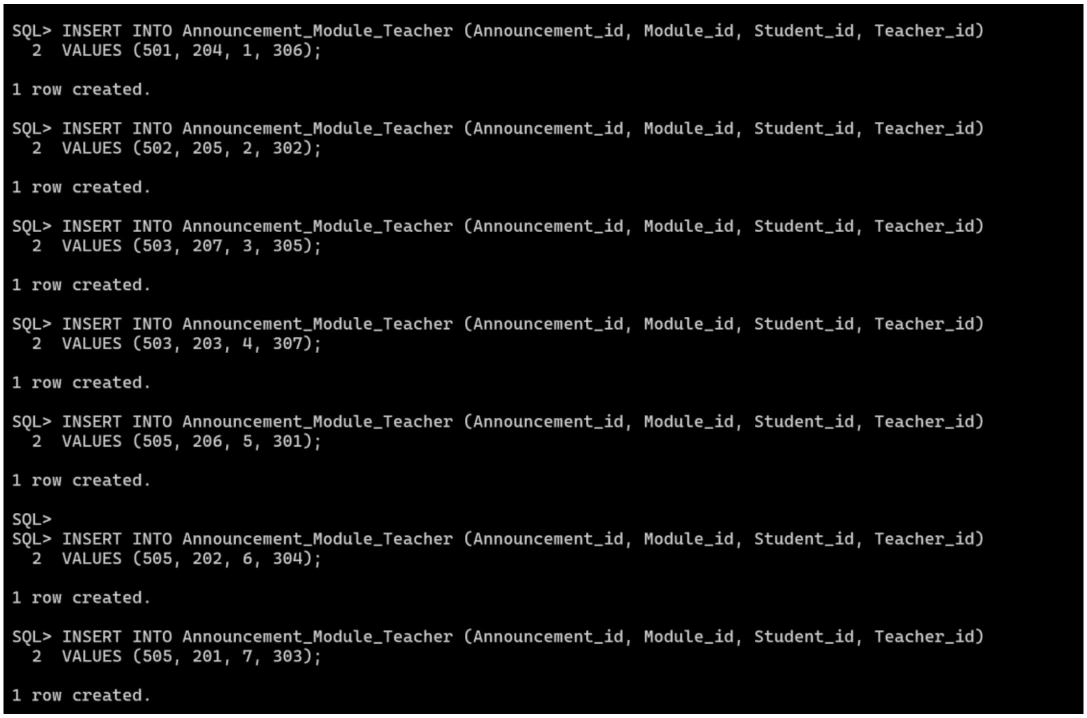
Figure 34 “insert” command to insert data in Announcement table.



```
ANNOUNCEMENT_ID ANNONCEME ANNOUNCEMENT_DESCRIPTION
--------------- --------- ------------------------------------------------------------
            502 10-MAY-24 Assignment re-evaluation period extended
            503 14-FEB-24 Industry expert seminar confirmed
            504 01-MAY-24 Project final submission portal open
            505 02-APR-24 Updated syllabus and revision classes mentioned
            506 28-MAY-24 Viva date released
            507 11-MAR-24 Student satisfaction feedback requested
            501 18-MAY-24 Final exam venues and seating plan released

7 rows selected.
```

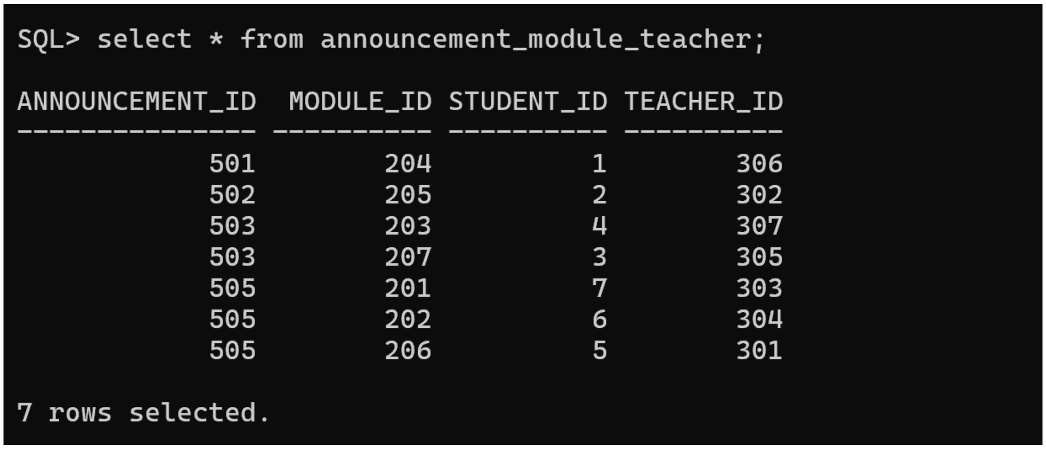
Figure 35 viewing Announcement table.

6.10 Use “insert” command to insert data in Announcement_Module _ Teacher table:



```
SQL> INSERT INTO Announcement_Module_Teacher (Announcement_id, Module_id, Student_id, Teacher_id)
  2  VALUES (501, 204, 1, 306);

1 row created.

SQL> INSERT INTO Announcement_Module_Teacher (Announcement_id, Module_id, Student_id, Teacher_id)
  2  VALUES (502, 205, 2, 302);

1 row created.

SQL> INSERT INTO Announcement_Module_Teacher (Announcement_id, Module_id, Student_id, Teacher_id)
  2  VALUES (503, 207, 3, 305);

1 row created.

SQL> INSERT INTO Announcement_Module_Teacher (Announcement_id, Module_id, Student_id, Teacher_id)
  2  VALUES (503, 203, 4, 307);

1 row created.

SQL> INSERT INTO Announcement_Module_Teacher (Announcement_id, Module_id, Student_id, Teacher_id)
  2  VALUES (505, 206, 5, 301);

1 row created.

SQL>
SQL> INSERT INTO Announcement_Module_Teacher (Announcement_id, Module_id, Student_id, Teacher_id)
  2  VALUES (505, 202, 6, 304);

1 row created.

SQL> INSERT INTO Announcement_Module_Teacher (Announcement_id, Module_id, Student_id, Teacher_id)
  2  VALUES (505, 201, 7, 303);

1 row created.
```

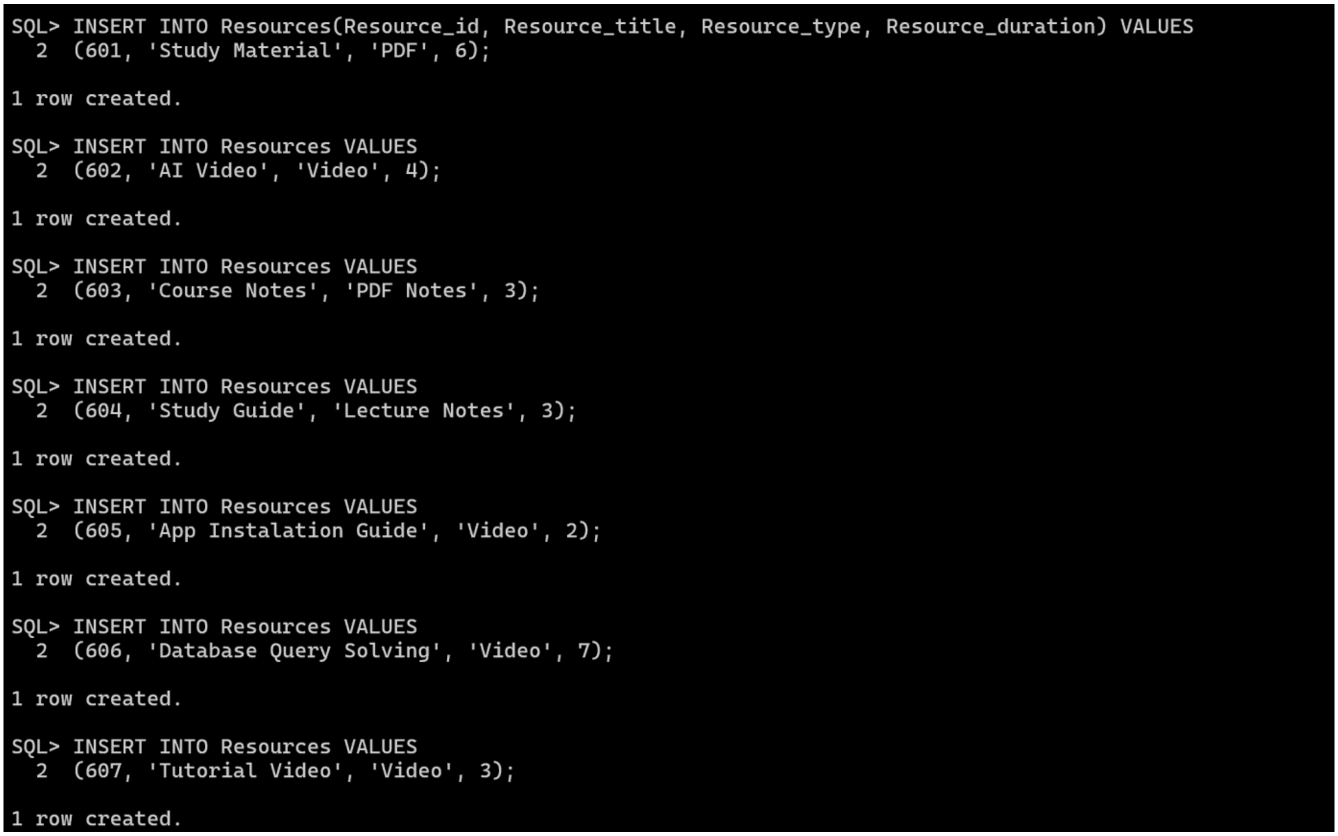
Figure 36 Announcement_Module _Teacher table.



```
SQL> select * from announcement_module_teacher;

ANNOUNCEMENT_ID  MODULE_ID STUDENT_ID TEACHER_ID
--------------- ---------- ---------- ----------
            501        204          1        306
            502        205          2        302
            503        203          4        307
            503        207          3        305
            505        201          7        303
            505        202          6        304
            505        206          5        301

7 rows selected.
```

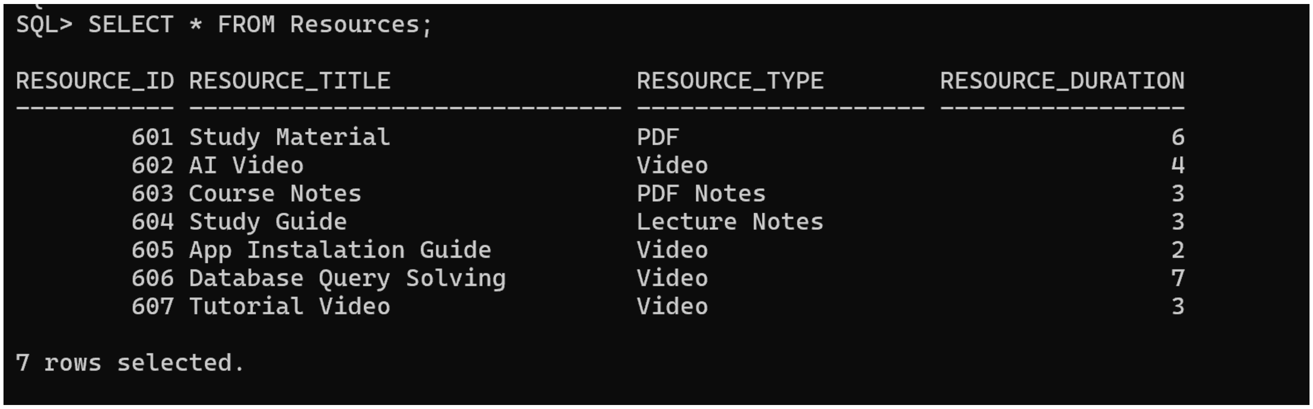
Figure 37 viewing Announcement_Module _Teacher table.

6.11 Use “insert” command to insert data in Resources table:



```
SQL> INSERT INTO Resources(Resource_id, Resource_title, Resource_type, Resource_duration) VALUES
  2  (601, 'Study Material', 'PDF', 6);

1 row created.

SQL> INSERT INTO Resources VALUES
  2  (602, 'AI Video', 'Video', 4);

1 row created.

SQL> INSERT INTO Resources VALUES
  2  (603, 'Course Notes', 'PDF Notes', 3);

1 row created.

SQL> INSERT INTO Resources VALUES
  2  (604, 'Study Guide', 'Lecture Notes', 3);

1 row created.

SQL> INSERT INTO Resources VALUES
  2  (605, 'App Instalation Guide', 'Video', 2);

1 row created.

SQL> INSERT INTO Resources VALUES
  2  (606, 'Database Query Solving', 'Video', 7);

1 row created.

SQL> INSERT INTO Resources VALUES
  2  (607, 'Tutorial Video', 'Video', 3);

1 row created.
```

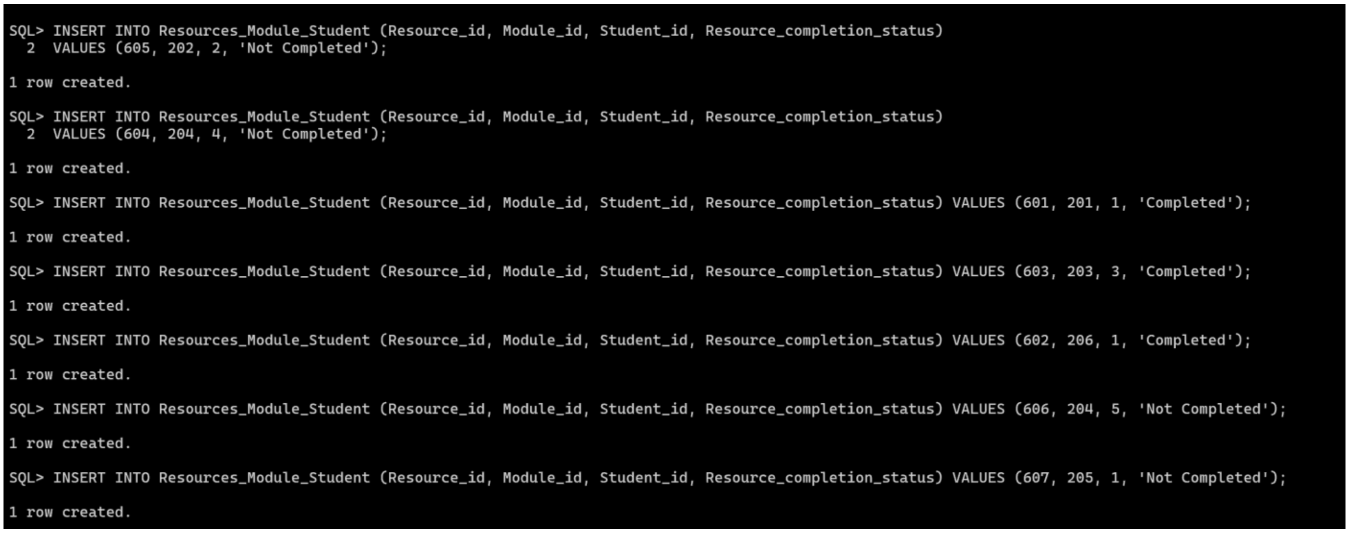
Figure 38 “insert” command to insert data in Resources table:



```
SQL> SELECT * FROM Resources;

RESOURCE_ID RESOURCE_TITLE                 RESOURCE_TYPE        RESOURCE_DURATION
----------- ------------------------------ -------------------- -----------------
        601 Study Material                 PDF                                  6
        602 AI Video                       Video                                4
        603 Course Notes                   PDF Notes                            3
        604 Study Guide                    Lecture Notes                        3
        605 App Instalation Guide          Video                                2
        606 Database Query Solving         Video                                7
        607 Tutorial Video                 Video                                3

7 rows selected.
```

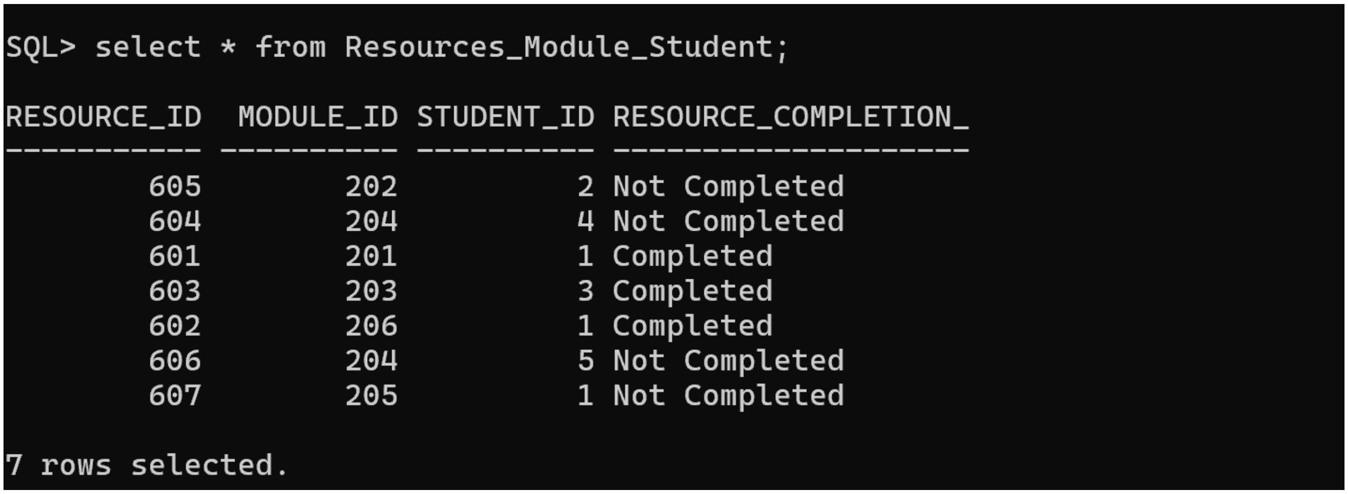
Figure 39 viewing Resources table:

6.12 Use “insert” command to insert data in Resource_Module_Student table:



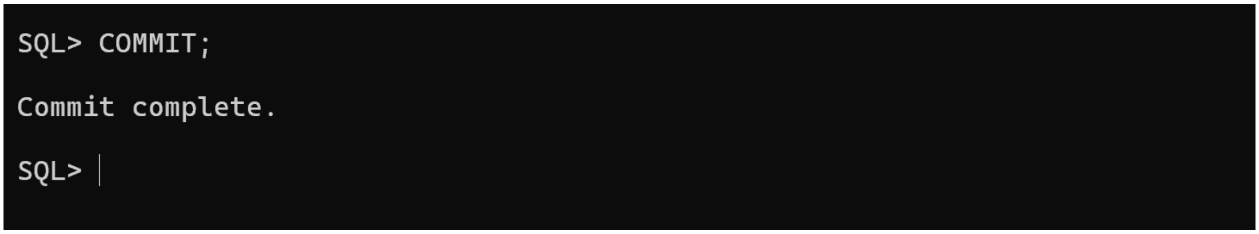
```
SQL> INSERT INTO Resources_Module_Student (Resource_id, Module_id, Student_id, Resource_completion_status)
  2  VALUES (605, 202, 2, 'Not Completed');

1 row created.

SQL> INSERT INTO Resources_Module_Student (Resource_id, Module_id, Student_id, Resource_completion_status)
  2  VALUES (604, 204, 4, 'Not Completed');

1 row created.

SQL> INSERT INTO Resources_Module_Student (Resource_id, Module_id, Student_id, Resource_completion_status) VALUES (601, 201, 1, 'Completed');

1 row created.

SQL> INSERT INTO Resources_Module_Student (Resource_id, Module_id, Student_id, Resource_completion_status) VALUES (603, 203, 3, 'Completed');

1 row created.

SQL> INSERT INTO Resources_Module_Student (Resource_id, Module_id, Student_id, Resource_completion_status) VALUES (602, 206, 1, 'Completed');

1 row created.

SQL> INSERT INTO Resources_Module_Student (Resource_id, Module_id, Student_id, Resource_completion_status) VALUES (606, 204, 5, 'Not Completed');

1 row created.

SQL> INSERT INTO Resources_Module_Student (Resource_id, Module_id, Student_id, Resource_completion_status) VALUES (607, 205, 1, 'Not Completed');

1 row created.
```

Figure 40 “insert” command to insert data in Resource_Module_Student table:



```
SQL> select * from Resources_Module_Student;

RESOURCE_ID  MODULE_ID STUDENT_ID RESOURCE_COMPLETION_
----------- ---------- ---------- --------------------
        605        202          2 Not Completed
        604        204          4 Not Completed
        601        201          1 Completed
        603        203          3 Completed
        602        206          1 Completed
        606        204          5 Not Completed
        607        205          1 Not Completed

7 rows selected.
```

Figure 41 viewing resource_Module_Student table:



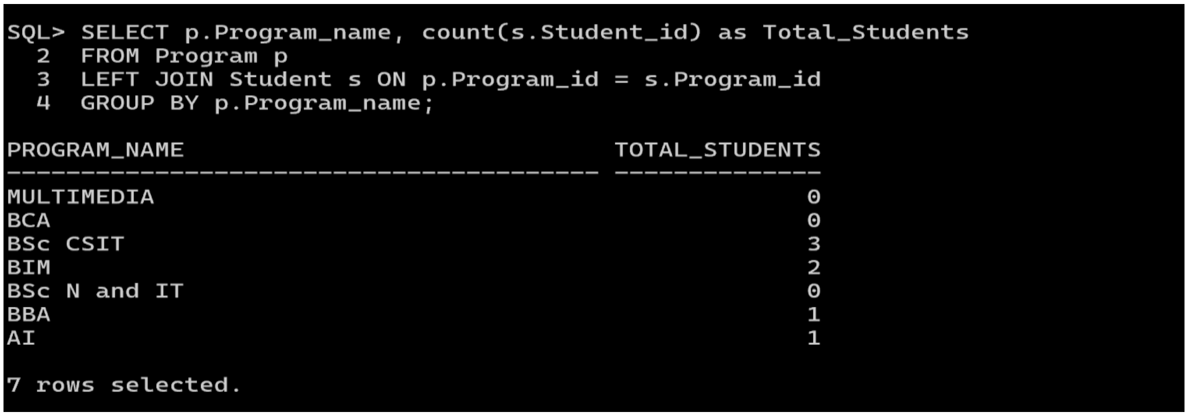
```
SQL> COMMIT;

Commit complete.

SQL>
```

Figure 42 saving SQL.

7. Using SQL to solve the following questions :

7.1 Information query:

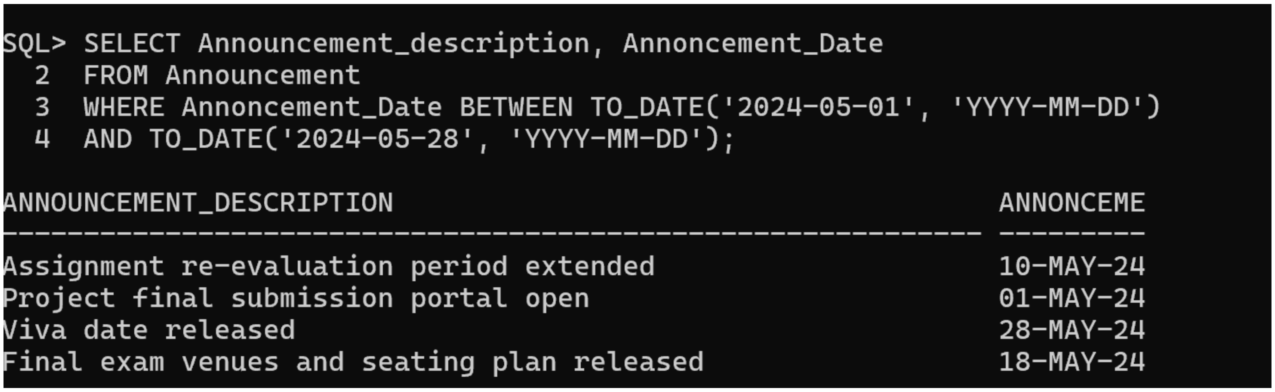
7.1.1. List the programs that are available in the college and the total number of students enrolled in each.



```
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
---------------------------------------- --------------
MULTIMEDIA                                            0
BCA                                                   0
BSc CSIT                                              3
BIM                                                   2
BSc N and IT                                          0
BBA                                                   1
AI                                                    1

7 rows selected.
```

Figure 43 List the programs that are available in the college and the total number of students enrolled in each.

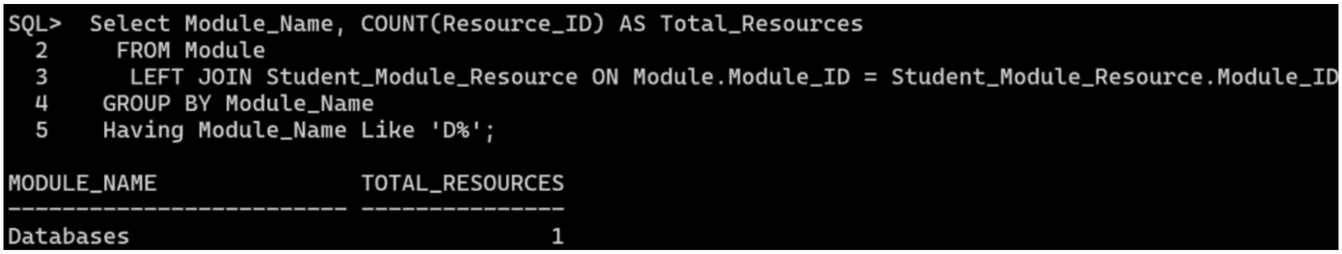
7.1.2 List all the announcements made for a particular module starting from 1st May 2024 to 28th May 2024.



```
SQL> SELECT Announcement_description, Annoncement_Date
     FROM Announcement
     WHERE Annoncement_Date BETWEEN TO_DATE('2024-05-01', 'YYYY-MM-DD')
     AND TO_DATE('2024-05-28', 'YYYY-MM-DD');

ANNOUNCEMENT_DESCRIPTION                                     ANNONCEME
------------------------------------------------------------ ---------
Assignment re-evaluation period extended                     10-MAY-24
Project final submission portal open                         01-MAY-24
Viva date released                                           28-MAY-24
Final exam venues and seating plan released                  18-MAY-24
```

Figure 44 List all the announcements made for a particular module starting from 1st May 2024 to 28th May 2024.

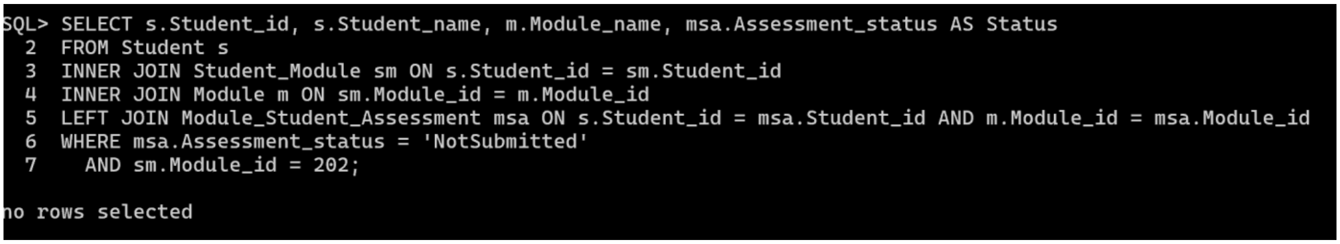
7.1.3 List the names of all modules that begin with the letter 'D', along with the total number of resources uploaded for those modules.



```
SQL>  Select Module_Name, COUNT(Resource_ID) AS Total_Resources
        FROM Module
         LEFT JOIN Student_Module_Resource ON Module.Module_ID = Student_Module_Resource.Module_ID
       GROUP BY Module_Name
       Having Module_Name Like 'D%';

MODULE_NAME               TOTAL_RESOURCES
------------------------- ---------------
Databases                               1
```

Figure 45 List the names of all modules that begin with the letter 'D', along with the total number of resources uploaded for those modules.

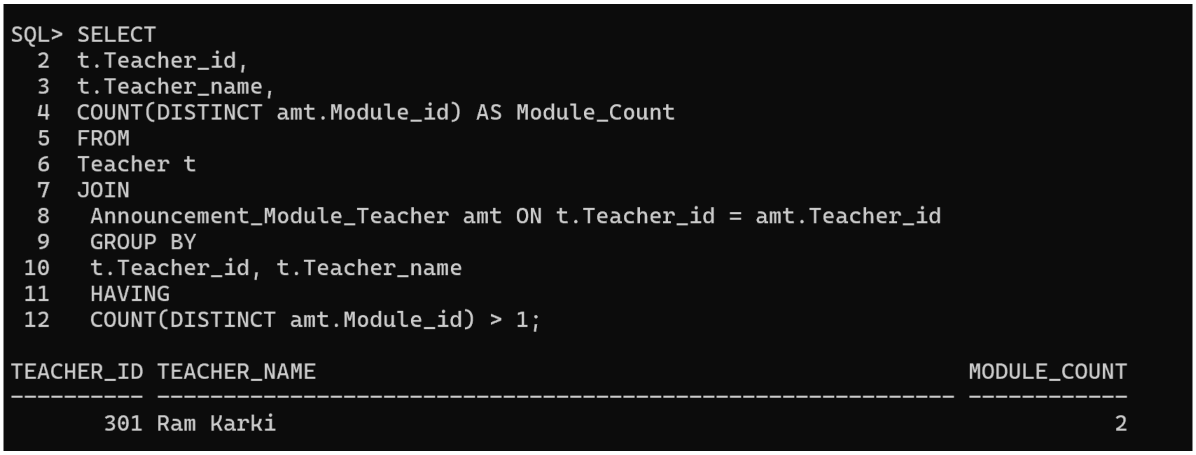
7.1.4 List the names of all students along with their enrolled program who have not submitted any assessments for a particular module.



```
SQL> SELECT s.Student_id, s.Student_name, m.Module_name, msa.Assessment_status AS Status
     FROM Student s
     INNER JOIN Student_Module sm ON s.Student_id = sm.Student_id
     INNER JOIN Module m ON sm.Module_id = m.Module_id
     LEFT JOIN Module_Student_Assessment msa ON s.Student_id = msa.Student_id AND m.Module_id = msa.Module_id
     WHERE msa.Assessment_status = 'NotSubmitted'
       AND sm.Module_id = 202;

no rows selected
```

Figure 46 List the names of all students along with their enrolled program who have not submitted any assessments for a particular module.

7.1.5 List of all the teachers who teach more than one module.

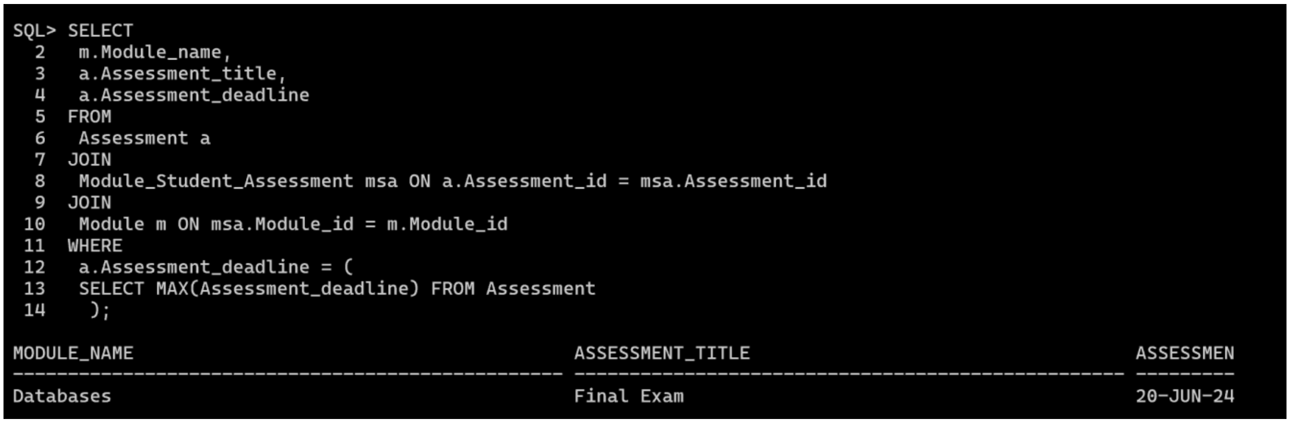


```
SQL> SELECT
     t.Teacher_id,
     t.Teacher_name,
     COUNT(DISTINCT amt.Module_id) AS Module_Count
     FROM
     Teacher t
     JOIN
      Announcement_Module_Teacher amt ON t.Teacher_id = amt.Teacher_id
      GROUP BY
      t.Teacher_id, t.Teacher_name
      HAVING
      COUNT(DISTINCT amt.Module_id) > 1;

TEACHER_ID TEACHER_NAME                                                MODULE_COUNT
---------- ------------------------------------------------------------ ------------
       301 Ram Karki                                                              2
```

Figure 47 List of all the teachers who teach more than one module.

7.2 Transaction query:

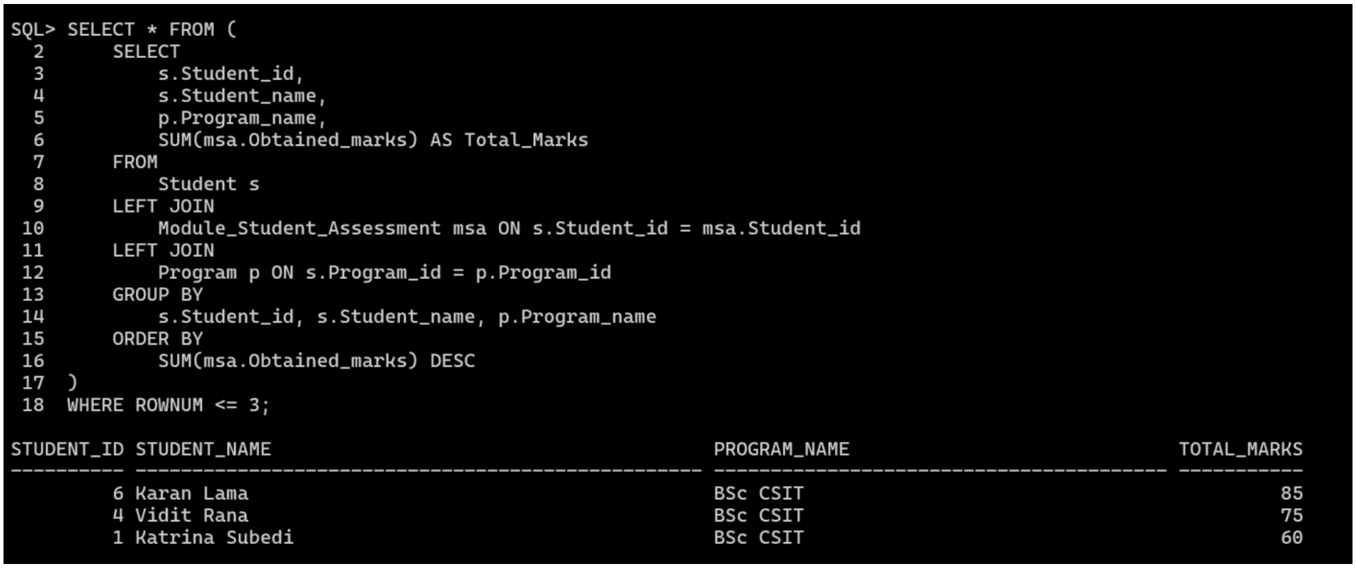
7.2.1 Identify the module that has the latest assessment deadline.



```
SQL> SELECT
      m.Module_name,
      a.Assessment_title,
      a.Assessment_deadline
     FROM
      Assessment a
     JOIN
      Module_Student_Assessment msa ON a.Assessment_id = msa.Assessment_id
     JOIN
      Module m ON msa.Module_id = m.Module_id
     WHERE
      a.Assessment_deadline = (
      SELECT MAX(Assessment_deadline) FROM Assessment
       );

MODULE_NAME                                        ASSESSMENT_TITLE                                   ASSESSMEN
-------------------------------------------------- -------------------------------------------------- ---------
Databases                                          Final Exam                                         20-JUN-24
```

Figure 48 Identify the module that has the latest assessment deadline.

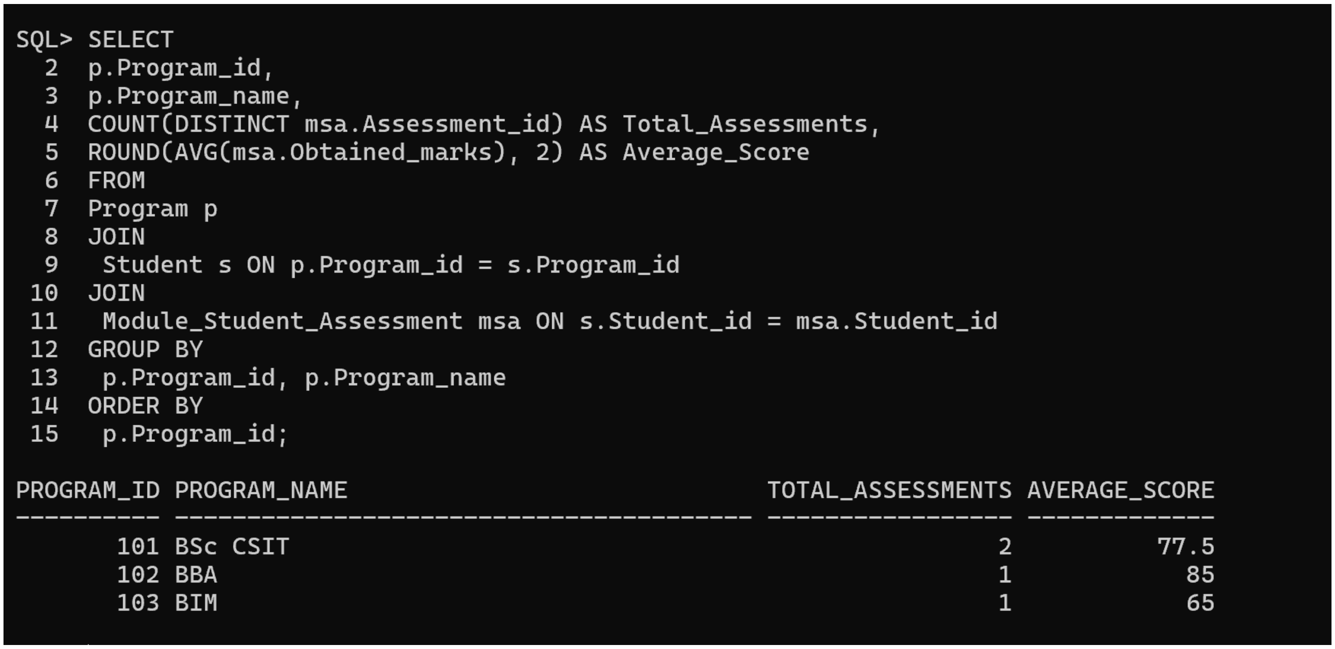
7.2.2 Find the top three students who have the highest total score across all modules.



```
SQL> SELECT * FROM (
         SELECT
             s.Student_id,
             s.Student_name,
             p.Program_name,
             SUM(msa.Obtained_marks) AS Total_Marks
         FROM
             Student s
         LEFT JOIN
             Module_Student_Assessment msa ON s.Student_id = msa.Student_id
         LEFT JOIN
             Program p ON s.Program_id = p.Program_id
         GROUP BY
             s.Student_id, s.Student_name, p.Program_name
         ORDER BY
             SUM(msa.Obtained_marks) DESC
     )
     WHERE ROWNUM <= 3;

STUDENT_ID STUDENT_NAME                                       PROGRAM_NAME                             TOTAL_MARKS
---------- -------------------------------------------------- ---------------------------------------- -----------
         6 Karan Lama                                         BSc CSIT                                          85
         4 Vidit Rana                                         BSc CSIT                                          75
         1 Katrina Subedi                                     BSc CSIT                                          60
```

Figure 49 Find the top three students who have the highest total score across all modules.

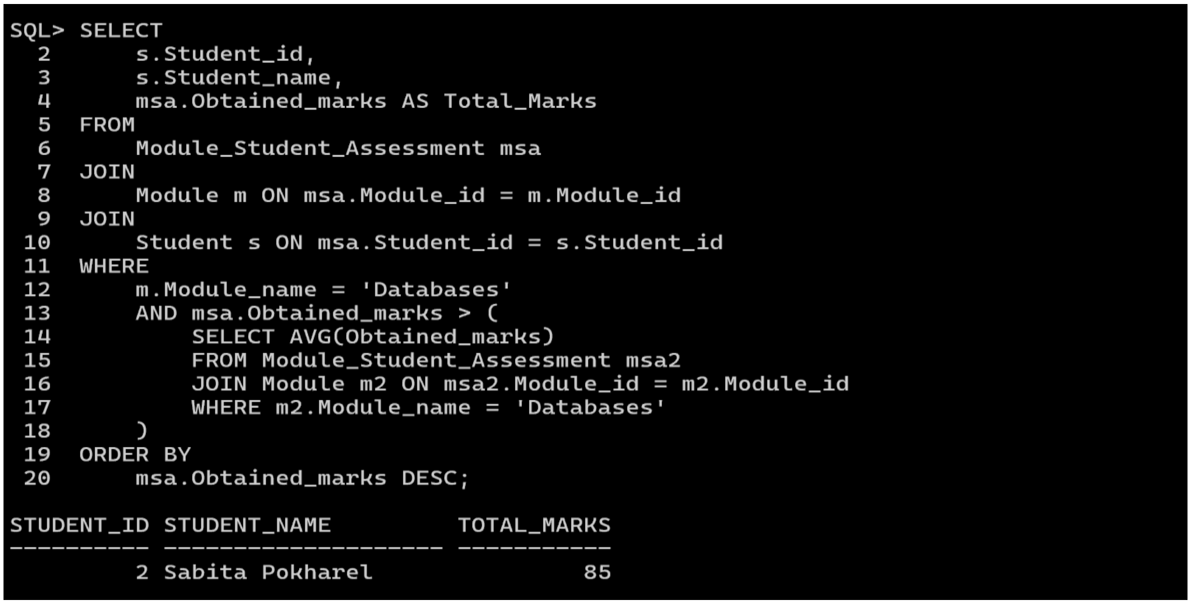
7.2.3. Find the total number of assessments for each program and the average score across all assessments in those programs.



```
SQL> SELECT
     p.Program_id,
     p.Program_name,
     COUNT(DISTINCT msa.Assessment_id) AS Total_Assessments,
     ROUND(AVG(msa.Obtained_marks), 2) AS Average_Score
     FROM
     Program p
     JOIN
      Student s ON p.Program_id = s.Program_id
     JOIN
      Module_Student_Assessment msa ON s.Student_id = msa.Student_id
     GROUP BY
      p.Program_id, p.Program_name
     ORDER BY
      p.Program_id;

PROGRAM_ID PROGRAM_NAME                             TOTAL_ASSESSMENTS AVERAGE_SCORE
---------- ---------------------------------------- ----------------- -------------
       101 BSc CSIT                                                 2          77.5
       102 BBA                                                      1            85
       103 BIM                                                      1            65
```

Figure 50 Find the total number of assessments for each program and the average score across all assessments in those programs.

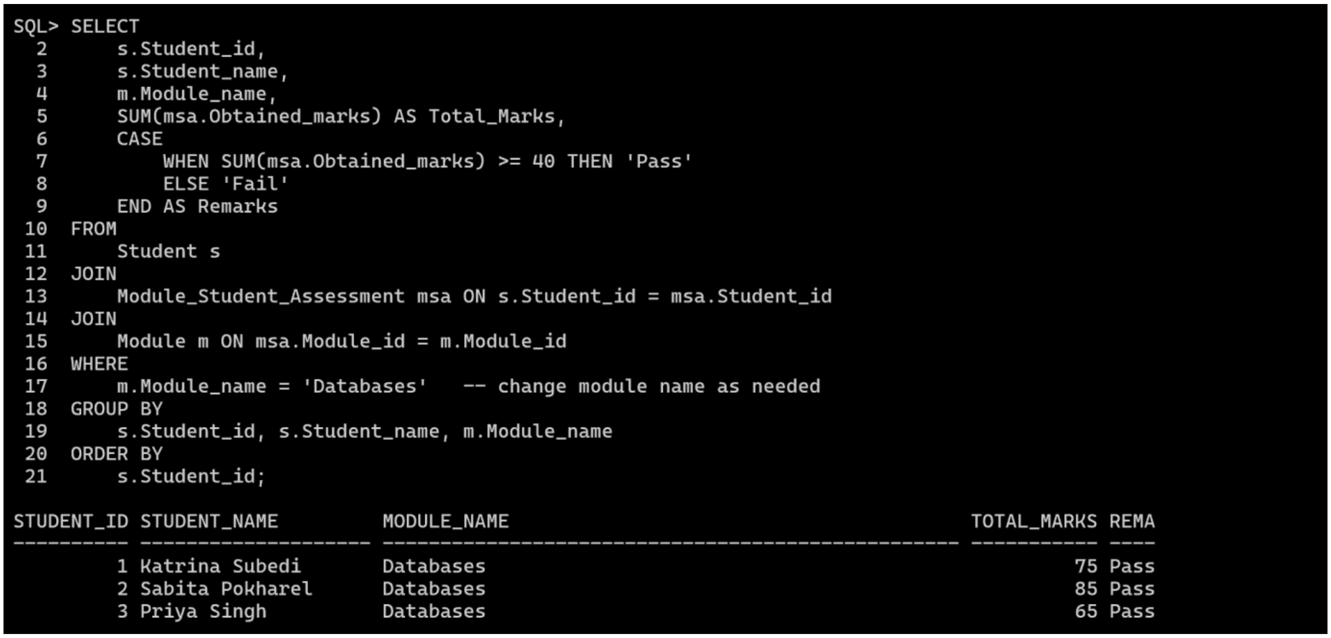
7.2.4 List the students who have scored above the average score in the ‘Databases’ module.



```
SQL> SELECT
         s.Student_id,
         s.Student_name,
         msa.Obtained_marks AS Total_Marks
     FROM
         Module_Student_Assessment msa
     JOIN
         Module m ON msa.Module_id = m.Module_id
     JOIN
         Student s ON msa.Student_id = s.Student_id
     WHERE
         m.Module_name = 'Databases'
         AND msa.Obtained_marks > (
             SELECT AVG(Obtained_marks)
             FROM Module_Student_Assessment msa2
             JOIN Module m2 ON msa2.Module_id = m2.Module_id
             WHERE m2.Module_name = 'Databases'
         )
     ORDER BY
         msa.Obtained_marks DESC;

STUDENT_ID STUDENT_NAME         TOTAL_MARKS
---------- -------------------- -----------
         2 Sabita Pokharel               85
```

Figure 51 List the students who have scored above the average score in the ‘Databases’ module.

7.2.5 Display whether a student has passed or failed as remarks as per their total aggregate marks obtained in a particular module. (NOTE: Consider total aggregate marks equal to or above 40 is pass , below 40 is fail



```
SQL> SELECT
         s.Student_id,
         s.Student_name,
         m.Module_name,
         SUM(msa.Obtained_marks) AS Total_Marks,
         CASE
             WHEN SUM(msa.Obtained_marks) >= 40 THEN 'Pass'
             ELSE 'Fail'
         END AS Remarks
     FROM
         Student s
     JOIN
         Module_Student_Assessment msa ON s.Student_id = msa.Student_id
     JOIN
         Module m ON msa.Module_id = m.Module_id
     WHERE
         m.Module_name = 'Databases'   -- change module name as needed
     GROUP BY
         s.Student_id, s.Student_name, m.Module_name
     ORDER BY
         s.Student_id;

STUDENT_ID STUDENT_NAME         MODULE_NAME                                        TOTAL_MARKS REMA
---------- -------------------- -------------------------------------------------- ----------- ----
         1 Katrina Subedi       Databases                                                   75 Pass
         2 Sabita Pokharel      Databases                                                   85 Pass
         3 Priya Singh          Databases                                                   65 Pass
```

Figure 52 Display whether a student has passed or failed as remarks as per their total aggregate marks obtained in a particular module.

8. Critical Evaluation

8.1 Critical Evaluation of module, its usage and relation with other subjects

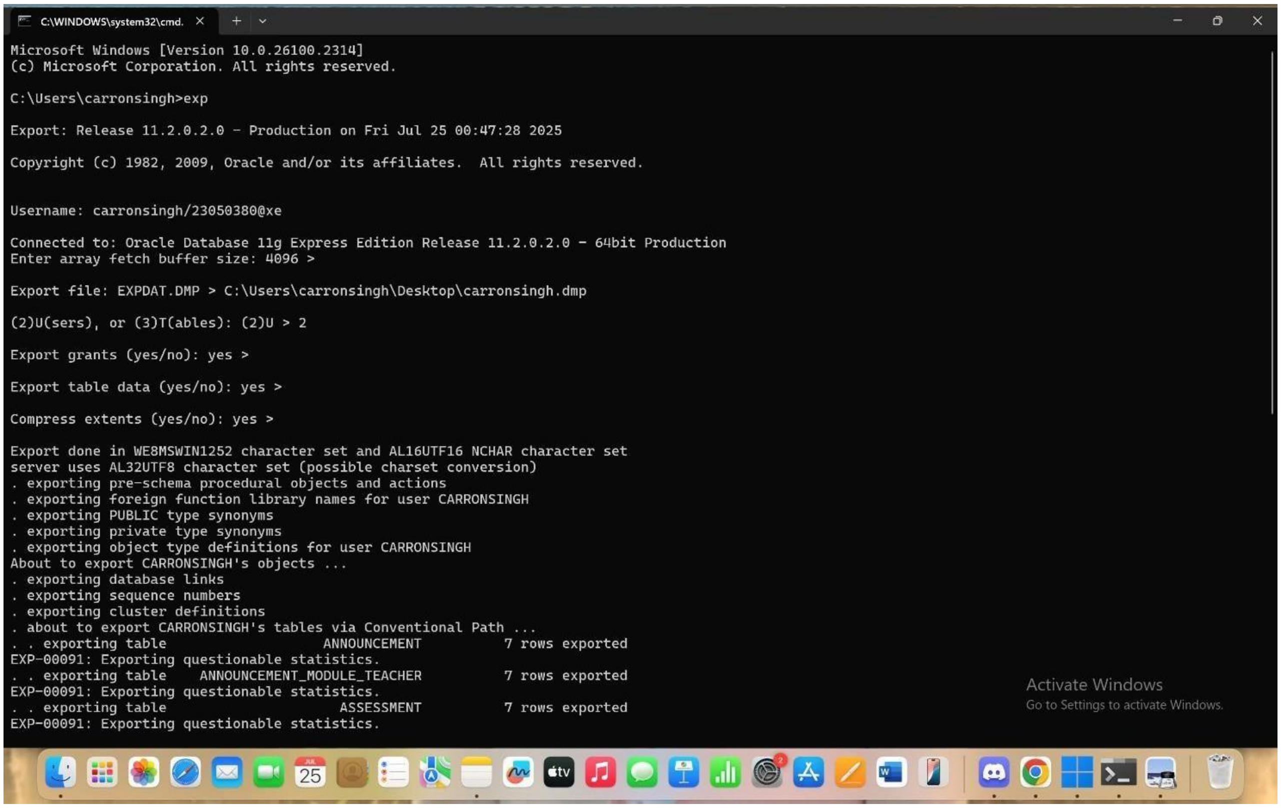
Through the module CC5051NI Database Systems, I gained much-needed knowledge on the theory of relational databases, database normalization and Structured Query Language (SQL). The normalization of data within the module by taking UNF to 3NF forms assisted me to understand data anomaly and model data structures to enhance data integrity and efficiency. The competencies mentioned can be directly applied to the software engineering trade, where the concepts of database design form the basis of backend system design, and programming modules guided the standard coding of connecting applications to the use of databases through SQL or Object-Relational Mapping (ORM) abstractions. As a result, the module knowledge can be applied to the field of full-stack development and can be informed project design in several other modules.

8.2 Critical Assessment of coursework

The course project provided logical, direct exposure to database design and implementation, at the same time improving theoretical understanding but serving to focus more on practice rather than theory with the usage of SQL Plus and the creation of dump-files. The functional processes, which were to create normalized tables, establish primary and foreign key between the tables and functional dependency principles, were specifically educative.

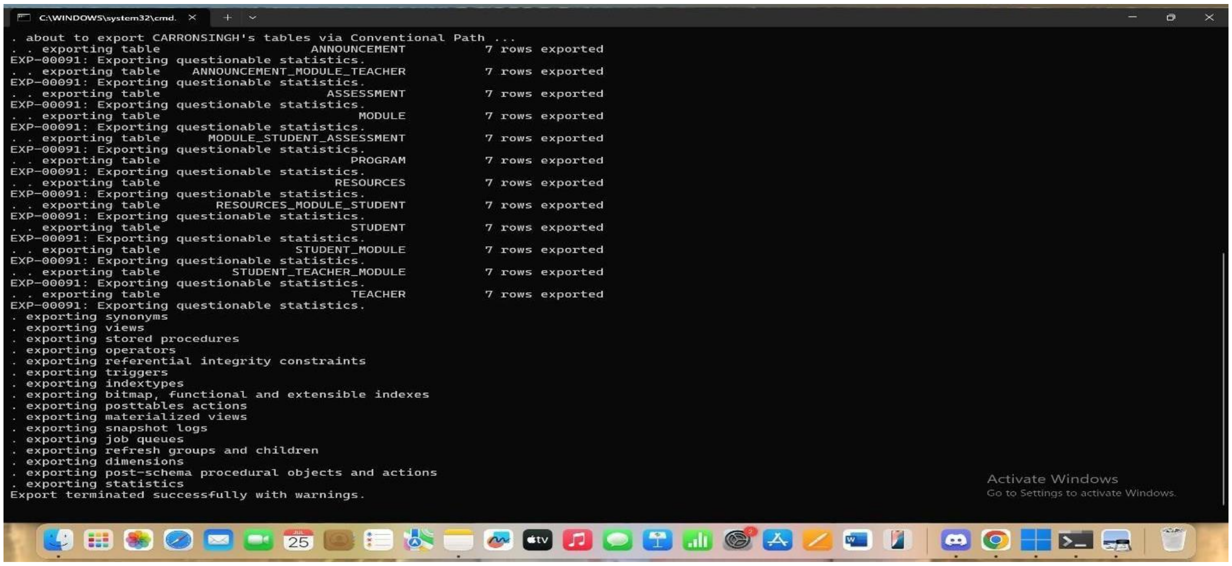
However, handling dependency on tables introducing and deletion often proved to be a difficult one. Major steps were required to achieve a precise sequencing of operations, as well as the arrest of referential integrity through continuous effort and diagnostics. Despite such challenges, the apparent situation after having established ERD development, data-injection, construction of queries, and retrieval of dump attained technological confidence in ways, which were signified highly. The peaceful balance between the theory and the execution of SQL as a practical exploration made the course alluring and thorough.

9. Dump file and drop



```
Microsoft Windows [Version 10.0.26100.2314]
(c) Microsoft Corporation. All rights reserved.

C:\Users\carronsingh>exp

Export: Release 11.2.0.2.0 - Production on Fri Jul 25 00:47:28 2025

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


Username: carronsingh/23050380@xe

Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
Enter array fetch buffer size: 4096 >

Export file: EXPDAT.DMP > C:\Users\carronsingh\Desktop\carronsingh.dmp

(2)U(sers), or (3)T(ables): (2)U > 2

Export grants (yes/no): yes >

Export table data (yes/no): yes >

Compress extents (yes/no): yes >

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 CARRONSINGH
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user CARRONSINGH
About to export CARRONSINGH's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export CARRONSINGH's tables via Conventional Path ...
. . exporting table                   ANNOUNCEMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table    ANNOUNCEMENT_MODULE_TEACHER          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                     ASSESSMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
```

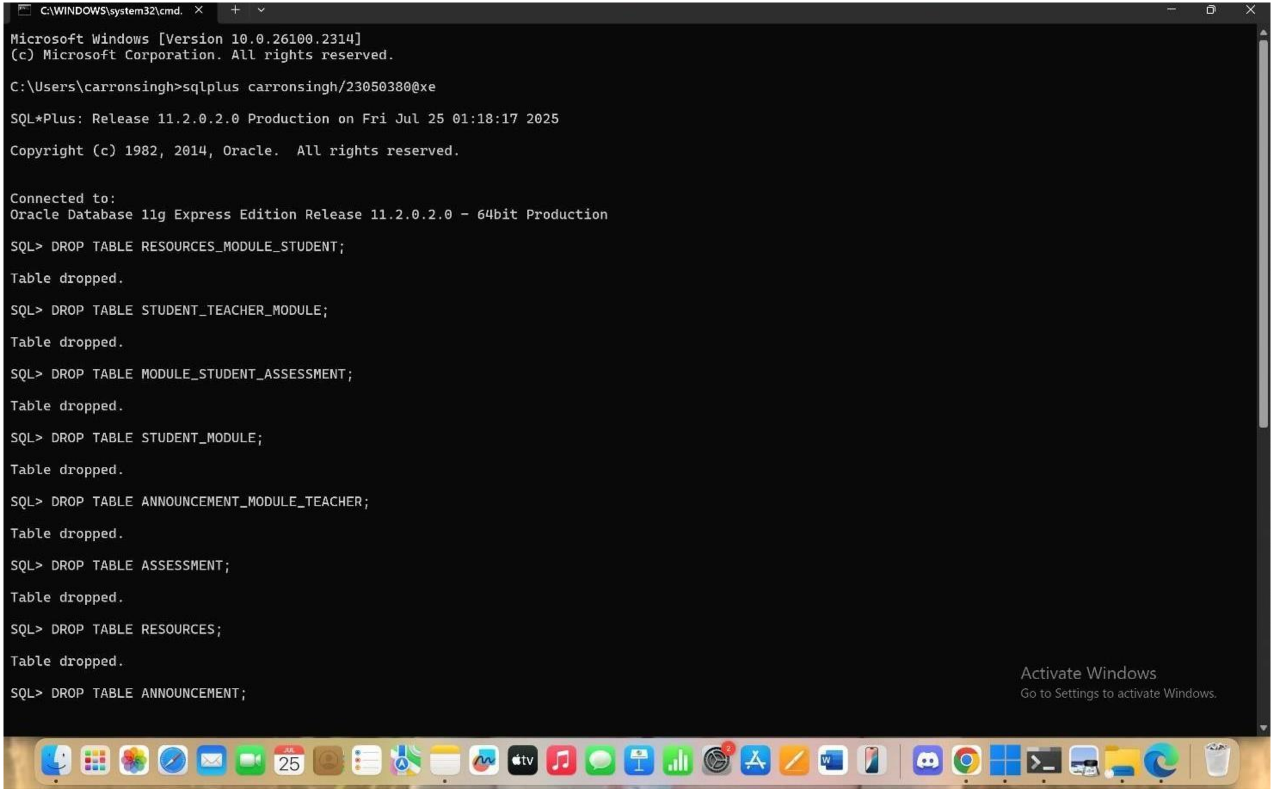
Figure 53 Dump file.



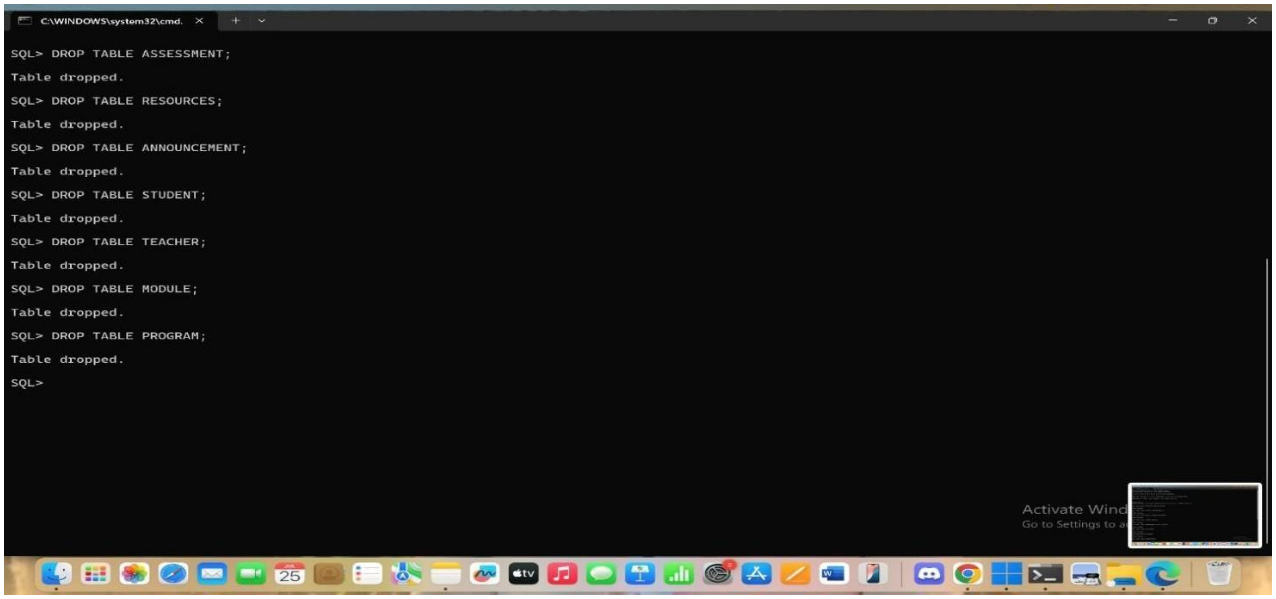
```
. about to export CARRONSINGH's tables via Conventional Path ...
. . exporting table                   ANNOUNCEMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table    ANNOUNCEMENT_MODULE_TEACHER          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                     ASSESSMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                         MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table      MODULE_STUDENT_ASSESSMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        PROGRAM          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                      RESOURCES          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table       RESOURCES_MODULE_STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                 STUDENT_MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table         STUDENT_TEACHER_MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        TEACHER          7 rows exported
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 54 Dump file.

Figure 55 drop file.



```
Microsoft Windows [Version 10.0.26100.2314]
(c) Microsoft Corporation. All rights reserved.

C:\Users\carronsingh>sqlplus carronsingh/23050380@xe

SQL*Plus: Release 11.2.0.2.0 Production on Fri Jul 25 01:18:17 2025

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


Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> DROP TABLE RESOURCES_MODULE_STUDENT;

Table dropped.

SQL> DROP TABLE STUDENT_TEACHER_MODULE;

Table dropped.

SQL> DROP TABLE MODULE_STUDENT_ASSESSMENT;

Table dropped.

SQL> DROP TABLE STUDENT_MODULE;

Table dropped.

SQL> DROP TABLE ANNOUNCEMENT_MODULE_TEACHER;

Table dropped.

SQL> DROP TABLE ASSESSMENT;

Table dropped.

SQL> DROP TABLE RESOURCES;

Table dropped.

SQL> DROP TABLE ANNOUNCEMENT;
```



```
SQL> DROP TABLE ASSESSMENT;
Table dropped.
SQL> DROP TABLE RESOURCES;
Table dropped.
SQL> DROP TABLE ANNOUNCEMENT;
Table dropped.
SQL> DROP TABLE STUDENT;
Table dropped.
SQL> DROP TABLE TEACHER;
Table dropped.
SQL> DROP TABLE MODULE;
Table dropped.
SQL> DROP TABLE PROGRAM;
Table dropped.
SQL>
```

Figure 56 drop file.

10. References

Author links open overlay panelYongchun Xu a 1 et al. (2023) Material database construction for data-driven computing via a continuous path-following method, Composite Structures. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0263822323005317 (Accessed: 23 January 2025).

Singh, J.P. (2023) Enhancing Database Security: A machine learning approach to anomaly detection in NoSQL systems. https://publications.dlpress.org/index.php/ijic/article/view/68.

Bharati, S. and Podder, P. (2022) 'Machine and deep learning for IoT security and privacy: applications, challenges, and future directions,' Security and Communication Networks, 2022, pp. 1–41. https://doi.org/10.1155/2022/8951961.

Rahmawati, N.T. et al. (2023) 'ANALISIS PERANCANGAN DATABASE MANAGEMEN SISTEM UNTUK SISTEM PENUNJANG
PROSES BISNIS WEDANG UWUH INSTAN,' TEKNIMEDIA Teknologi Informasi Dan Multimedia, 4(1), pp. 61–
69. https://doi.org/10.46764/teknimedia.v4i1.104.

Kind, M.C. et al. (2019) 'easyaccess: Enhanced SQL command line interpreter for astronomical surveys,' The Journal of Open-Source Software, 4(33), p. 1022. https://doi.org/10.21105/joss.01022.

Xu, Y. et al. (2023). Material database construction for data-driven computing via a continuous path-following method, Composite Structures. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0263822323005317 (Accessed: 26th July 2025).
Singh, J.P. (2023). Enhancing Database Security: A machine learning approach to anomaly detection in NoSQL systems. Available at: https://publications.dlpress.org/index.php/ijic/article/view/68 (Accessed: 26th July 2025).

Bharati, S. and Podder, P. (2022). Machine and deep learning for IoT security and privacy: applications, challenges, and future directions, Security and Communication Networks. Available at:

https://doi.org/10.1155/2022/8951961 (Accessed: 26th July 2025).

Rahmawati, N.T. et al. (2023). Analisis Perancangan Database Managemen Sistem untuk Sistem Penunjang Proses Bisnis Wedang Uwuh Instan, TEKNIMEDIA Teknologi Informasi dan Multimedia, (1), pp. 61–69. Available at: https://doi.org/10.46764/teknimedia.v4i1.104 (Accessed: 26th July 2025).

Kind, M.C. et al. (2019). easyaccess: Enhanced SQL command line interpreter for astronomical surveys, The Journal of Open-Source Software, 4(33), p. 1022. Available at: https://doi.org/10.21105/joss.01022 (Accessed: 26th July 2025).

Date, C.J. (2003). An Introduction to Database Systems. 8th end. Boston: Pearson Education.

Elmasri, R. and Navathe, S.B. (2015). Fundamentals of Database Systems. 7th edn. Pearson Education.

Connolly, T. and Begg, C. (2014). Database Systems: A Practical Approach to Design, Implementation, and Management. 6th edn. Harlow: Pearson Education.

Silberschatz, A., Korth, H.F., and Sudarshan, S. (2019). Database System Concepts. 7th edn. New York: McGraw-Hill Education.

Coronel, C. and Morris, S. (2017). Database Systems: Design, Implementation, & Management. 12th edn. Boston: Cengage Learning.