Module Code & Module Title

CC5051NT

Database

Assessment Type

100% Individual Coursework

2024/25 Autumn

Student Name: Bidur Siwakoti

London Met ID: 23049216

College ID: NP05CP4A230013

Submitted To: Ajay Raj Bhattarai

Word Count:10923

Assignment Due Date: Thursday, January 23, 2025

Assignment Submission Date: Thursday, January 23, 2025

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

coursework

Islington College,Nepal

Document Details

Submission ID trn:oid:::3618:79895290	59 Pages
Submission Date Jan 23, 2025, 3:44 AM GMT+5:45	6,709 Words
Download Date Jan 23, 2025, 3:47 AM GMT+5:45	
File Name coursework	
File Size 49.1 KB	

19% Overall Similarity

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

Match Groups

136 Not Cited or Quoted 17%

Matches with neither in-text citation nor quotation marks

4 Missing Quotations 1%

Matches that are still very similar to source material

0 Missing Citation 0%

Matches that have quotation marks, but no in-text citation

5 Cited and Quoted 1%

Matches with in-text citation present, but no quotation marks

Top Sources

2%	Internet sources
0%	Publications
18%	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.

Match Groups

136 Not Cited or Quoted 17%

Matches with neither in-text citation nor quotation marks

4 Missing Quotations 1%

Matches that are still very similar to source material

0 Missing Citation 0%

Matches that have quotation marks, but no in-text citation

5 Cited and Quoted 1%

Matches with in-text citation present, but no quotation marks

Top Sources

2%	Internet sources
0%	Publications
18%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1 **Submitted works**

islingtoncollege on 2024-12-30 **3%**

2 **Submitted works**

islingtoncollege on 2024-12-30 **2%**

3 **Submitted works**

University of Technology, Sydney on 2010-06-03 **1%**

4 **Submitted works**

islingtoncollege on 2024-12-30 **<1%**

5 **Submitted works**

Central Queensland University on 2018-05-20 **<1%**

6 **Submitted works**

University of Wollongong on 2023-03-26 **<1%**

7 **Submitted works**

Kaplan International Colleges on 2023-04-07 **<1%**

---

8 **Submitted works**

University of Nottingham on 2024-12-04 **<1%**

---

9 **Submitted works**

University of Massachusetts-Dartmouth on 2024-10-09 **<1%**

---

10 **Submitted works**

University of Technology, Sydney on 2013-10-28 **<1%**

11	Submitted works	University of London External System on 2013-03-19	<1%
12	Submitted works	Kaplan International Colleges on 2024-04-01	<1%
13	Submitted works	islingtoncollege on 2025-01-03	<1%
14	Submitted works	Kaplan International Colleges on 2023-06-28	<1%
15	Publication	Valluri, Sindhuja. "Enhancement of Performance of Micro Direct Ethanol Fuel Cell...	<1%
16	Submitted works	Ajman University of Science and Technology on 2024-11-21	<1%
17	Submitted works	De Montfort University on 2019-01-11	<1%
18	Submitted works	University of Arizona on 2013-04-08	<1%
19	Submitted works	University of Essex on 2006-03-28	<1%
20	Submitted works	islingtoncollege on 2025-01-03	<1%
21	Submitted works	Asia Pacific University College of Technology and Innovation (UCTI) on 2022-10-19	<1%

22 Submitted works

Institute of Advanced Technology on 2009-07-27 <1%

23 Submitted works

University of Huddersfield on 2013-04-15 <1%

24 Submitted works

University of Northampton on 2012-01-24 <1%

25	Internet	www.orafaq.com	<1%
26	Submitted works	University of Wales Institute, Cardiff on 2024-05-02	<1%
27	Submitted works	Queen Mary and Westfield College on 2025-01-10	<1%
28	Submitted works	School of Accounting & Management on 2010-07-29	<1%
29	Submitted works	University of Huddersfield on 2013-04-15	<1%
30	Submitted works	University Of Tasmania on 2024-05-24	<1%
31	Submitted works	The Robert Gordon University on 2024-11-04	<1%
32	Submitted works	Learnit Institute of Business and Technology on 2009-08-26	<1%
33	Submitted works	University of Huddersfield on 2013-04-15	<1%
34	Internet	vbook.pub	<1%
35	Submitted works	Central Queensland University on 2017-05-22	<1%

36 Submitted works

University of Arizona on 2012-10-29 <1%

37 Submitted works

University of East London on 2016-04-26 <1%

38 Submitted works

De Montfort University on 2024-11-14 <1%

39 Submitted works

Flinders University on 2024-05-13 <1%

40 Submitted works

Info Myanmar College on 2025-01-10 <1%

41 Submitted works

Institute of Advanced Technology on 2010-08-31 <1%

42 Submitted works

Middlesex University on 2014-04-27 <1%

43 Submitted works

Molloy College on 2022-09-29 <1%

44 Submitted works

Napier University on 2021-08-20 <1%

45 Submitted works

RDI Distance Learning on 2013-02-13 <1%

46 Submitted works

School of Accounting & Management on 2012-01-23 <1%

47 Submitted works

Study Group Australia on 2015-09-07 <1%

48 Submitted works

University of Arizona on 2012-10-29 <1%

49 Submitted works

University of Huddersfield on 2013-04-15 <1%

50 Submitted works

University of Northampton on 2024-09-01 <1%

51 Submitted works

University of Technology, Sydney on 2008-11-03 <1%

52 Submitted works

University of Wales Institute, Cardiff on 2023-08-16 <1%

53	Submitted works	University of Wales Institute, Cardiff on 2023-08-17	<1%
54	Submitted works	University of Westminster on 2012-05-01	<1%
55	Submitted works	Western Governors University on 2017-04-17	<1%
56	Submitted works	Western Governors University on 2017-04-24	<1%
57	Submitted works	islingtoncollege on 2024-12-30	<1%
58	Submitted works	islingtoncollege on 2025-01-02	<1%
59	Submitted works	American Intercontinental University Online on 2006-11-30	<1%
60	Submitted works	islingtoncollege on 2024-12-30	<1%
61	Submitted works	islingtoncollege on 2024-12-30	<1%

Table of Contents

Tables of Figures:

Tables of Tables:

1 Introduction

1.1 Introduction of the business and its forte

"Barcelona International Collage" is a famous and prestigious educational institution that has earned reputation for its academic excellence. It was established in 2018 and quickly become one of the favourites collages in its region. Barcelona International offers wide range of program. The main purpose this collage is to provide quality education and better services to student. The founder and CEO of Bright future collage Ms. Marry is the key person for the collage success. He is very passionate about improving the education system by using Morden technology. He is currently working to revounizes education system in his collage by introducing E-classroom which is a digital platform to modernize the classroom experiences for both student and teacher.

Figure 1: Barcelona International Collage.

In today's digital age, e-classroom platform like My Second Techer, Google Classroom is becoming more and more popular. CEO. Ms. Marry has decided to create an online platform for a college to manage student, teacher and programs. As a database designer, I have been given the task to build a reliable database system to support this idea. Here I have to manage things like the student, their enrolment, program structure, module assignment etc. The main purpose of this database is to create a system a reliable system that can manage all the administrative and academic task of the collage. The system should be capable to handle all the task like enrolling student in program, managing module assessment etc that are related with student or teacher. I should also think resources because they should be organized in proper sequence.

For this, We are using Oracle SQL PLUS, which is known for being secure, reliable and high performing. It is robust and widely used relational database management system. It serves as a foundation tool for database administrator and developer to execute SQL commands, manage database objects. One of the major strengths of Oracle Plus is its integration with Oracle database, enabling seamless access to robust database features. Whether for basic queries or complex administrative tasks, SQL Plus provides a reliable and flexible platform to meet diverse database needs.

1.2 Current Business Activities and Operations

Currently, the collage is offerings a wide range of activities to support both academic and personal development of the student. The educational activities in Barcelona international collage include physical classes, lab work, online classes and online resources for student. The collage also focus on extracurricular activities such as sport competitions and cultural event which helps student for all round development. The collage is also popular for scholarships program, which aim is to provide free education for such deserving student who have financial problem.

To further improve educational quality, CEO Mr. Marry has proposed the development of online learning platform which is a digital platform that helps student to get learning material online. The aim of this platform is to end the gap between traditional teaching and modern digital education. This platform will offer features like interactive lesson, discussion forums, quizzes and assignment enhancing the learning experiences for student who may not attend the physical class. The platform is designed to simplify the academic and administrative operation of collage. The primary activities of this platform are enrolling student in their interested program such as science, computing, networking etc. A single program contains multiple modules and multiple teachers associated with it. Teacher and module management is another important thing to be consider while making the database. A teacher can teach multiple modules and for a single module there are multiple teachers assigned. Modules are enrich with resources like lecture slides, videos, notes and textbook. The resources are carefully arranged with sequence number t make the learning in certain curve. For example, the student must complete first resources before going to second resources which helps student to understand thigs effectively. Finally, to evaluate student performance each modules have their respective coursework and exam. Coursework may be individual and group whereas exam may theory and MCQ. Once the assessment are checked the student worked are marked with grade and feedback. In conclusion, these all activities make the platform sustainable, well-structured and improve learning curve. The platform improve the efficiency for both academic and administrative function allowing the collage to run smoothly by cooperating with all the staff and student in the institution.

.

1.3 Business Rules

A statement that places restrictions on a certain feature of the database, such the properties of a given relationship or the contents of a field definition for a single field, is called a business rule. (eTutorials, 2022) These rules are essential to ensure consistency, efficiency and accuracy in how task is carried out within the a system or business environment. They are used to describe the relationships, behaviors or constraints within a system and help to maintain orders.

Some of the importances of business rules are listed below:

- Business rules standardize processes, ensuring all activities are carried out uniformly across the organization.
- It makes operations smooth by providing clear guidelines.
- Business rules minimize errors by setting clear constraints and expectations for how task should be performed.
- Business rules help all stakeholders to understand about the process and responsibilities.
- It ensures that organizational operations align with regulations, policies or industry standards.

The business rules used that are used while developing the database of E-Classroom are:

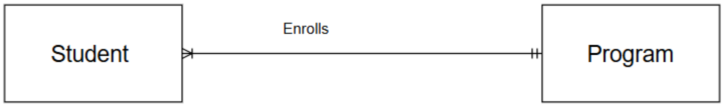
- Students must be enrolled in exactly in one program offered by collage at a time.
- Each program consists of multiple modules, and each module is mandatory for student enrolled in the program.
- Each module can be associated with multiple programs, allowing flexibility in curriculum design.
- Each module is assigned to one or more teacher based on their expertise.
- Every assessment must be associated with a specific module, and module can have multiple assessment.
- Each module has one or more resources.
- Each assessment can have only one result.
- Each teacher can make one or more announcements, and each announcement is made by exactly one teacher.

- Resources within a module must be accessed in predefined sequence. Student must complete one resources before proceeding to the next.
- Each assessment includes attributes like title, deadline, and weightage, which must be defined during its creations.
- Announcement are specific to a module and can only be viewed by student and teacher associated with that module.
- Each student's results must include the assessment details, totals mark obtained, and remarks indicating pass or fails.

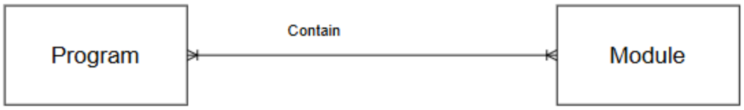
2 Identifications of Entities and Attributed

In any database, entities and attributes form the foundation for creating a well structured and efficient database system. Entities represent the core objective or concepts in system where as attributes define the specific details or properties associated with an entity. They describe the characteristics of an entity and provide the necessary data points for tracking and managing its information. For examples, in this system Student might be one entity and its attributes might be Student ID, Student Name, Address etc. Below are the entities and attributes that are fundamental to this E-Classroom System.

- **Relationships:** A database relationship is the logical connection between the tables in the give database. it helps us to understand how the tables are related with other in the database. in relational database it is the most important things. (indeed, 2023). In database there are many types of relation. Some of them are one-to-many, many-to-many, many-to-one. The relation that are used here are:
    - ➢ The relationship between Student and program is:



    - ➢ The relationship between Program and Module is:



1. Student

This tables stores the attributes of the student:

Table 1: Identification of Student attributes,

Attributes	Datatype	Constraints
studentID	NUMBER	PRIMARY KEY, NOT NULL,
Student_name	CHARATER (20)	NOT NULL
Phone_Number	CHARACTER (20)	NOT NULL
Email	CHARACTER(20)	NOT NULL, UNIQUE
Address	CHARACTER (20)	NOT NULL
DOB	DATE	NOT NULL

2. Program

This tables stores the attributes of the program with its datatype, constraints:

Table 2: Identification of Program attributes.

Attributes	Datatype	Constraints
program_ID	NUMBER	PRIMARY KEY, NOT NULL,
program_name	CHARACTER(20)	NOT NULL, UNIQUE
duration	NUMBER	NOT NULL
total_modules	NUMBER	NOT NULL
Program_Type	CHARACTER (20)	NOT NULL

3. module

This tables stores the attributes of the module with its datatype and constraints:

Table 3: Identification of module attributes.

Attributes	Datatype	Constraints
module_ID	CHARACTER (10)	PRIMARY KEY, NOT NULL
module_name	CHARACTER (20)	NOT NULL
credits_hours	NUMBER	NOT NULL
Module_Type	CHARACTER (20)	NOT NULL
Module_Desc	CHARACTER (25)	NULLABLE
resources_ID	CHARACTER (10)	NOT NULL
title	CHARACTER (25)	NOT NULL
resources_type	CHARACTER (50)	NOT NULL
duration	CHARACTER (20)	NOT NULL
Sequence_number	NUMBER	NOT NULL
teacher_ID	CHARACTER (10)	NOT NULL
teacher_name	CHARACTER (20)	NOT NULL
Teacher_email	CHARACTER (25)	NOT NULL, UNIQUE
Contact_number	CHARACTER (15)	NOT NULL
department	CHARACTER (20)	NOT NULL
specialization	CHARACTER (15)	NOT NULL
announcement_ID	CHARACTER (10)	NOT NULL
title	CHARACTER (25)	NOT NULL
Date_posted	DATE	NOT NULL

End_date	DATE	NOT NULL
Announcement_Desc	CHARACTER (20)	NOT NULL,
assessment_ID	CHARACTER (10)	NOT NULL
Assessment_name	CHARACTER (25)	NOT NULL
Asmt_posted_date	DATE	NOT NULL
Asmt_end_date	DATE	NOT NULL
Weightage	NUMBER	NOT NULL
result_ID	CHARACTER (10)	NOT NULL, UNIQUE
Mark_obtained	NUMBER(10)	NOT NULL
total_marks	NUMBER(10)	NOT NULL
remarks	CHARACTER (25)	NULLABLE
Feedback	CHARACTER (5)	NOT NULL

2.1 Initial ERD

An ERD (Entity Relationship Diagram) which is also know as entity relationship model is a visual representation that represent the relationship between physical object or real-world entity that can be places concepts or events. (Jacqueline Biscobing, 2024) ERDs are widely used for data modeling which help business processes and make the foundation for relational databases. The key components of the ERD are Entities, Attributes and Relationships. Entities are the real word object, people etc. For example, in the E-Classroom platform the entities are student, Program, Module, Teacher etc. Attributes are the characteristics or the behaviour of the entities. For examples, the attributes of the student are Student_id, Student_name, etc.

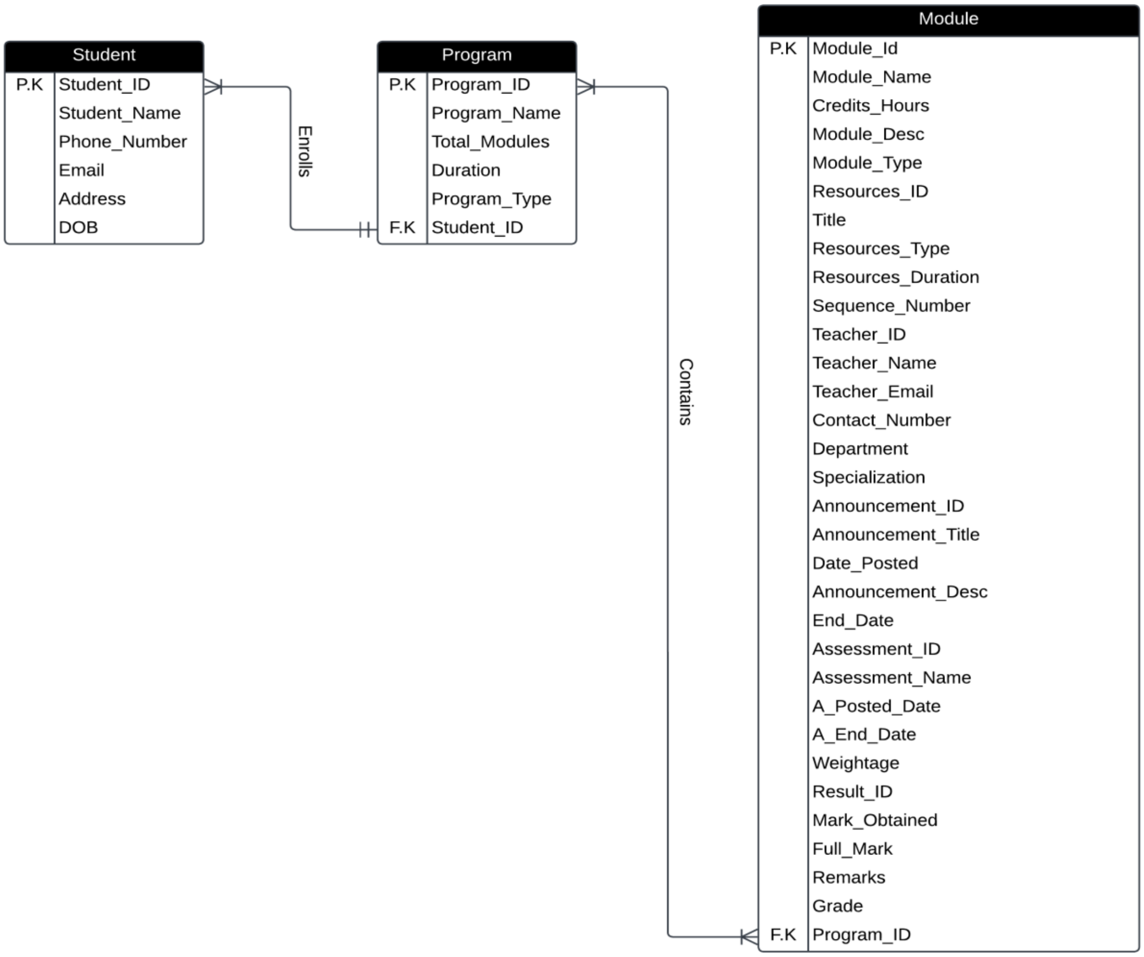


Figure 2: Initial ERD

3 Normalization

Normalization is the way of structuring a database to eliminate duplicate data by storing it in one place and making sure related data is kept together in a logical way. (Rouse, 2023). Simply normalization is the way to organize data is the database so that it is clean and efficient. The goal of normalization is to eliminate duplicate data and make everything is logically connected which helps to avoid inconsistences and make easier to manages and update the data. It is done multiple stage called Normal Forms, starting From UNF and going to higher levels like 1NF, 2NF, 3NF and sometime beyond.

Some of the advantages of normalization are:

- Data is stored only once, avoiding unnecessary duplicates and saving storage space.
- Since each piece of information is kept in only one point, it is easier to update or delete data without causing inconsistences.
- With fewer duplicate records, updates and modification becomes straightforward and less error prone.
- It helps to prevent from data anomalies like Insertion Anomaly, Deletion Anomaly and Update Anomaly.
- By logically organizing data, certain queries become faster and more efficient.

Normalization has some disadvantages, especially when it comes to practical use in certain situations. One major issue is that it can make queries more complicated because the data is split into multiple smaller tables. This means we need to used a lot of joins to fetch the required information for the database. Normalizing a database requires careful planning and understanding of data relationships, which can be time consuming during the design phase. Finally in some cases, like read intensive application or small datasets, denormalized structures might be more practical.

3.1 UNF (Unnormalized Form)

Unnormalized Form (UNF), which is also called as Non-First Normal Form (N1NF or NF2), refers to a database structure that doesn't follow any of the rules of normalization in the relational model. Essentially, it's a way of structuring data in a database without applying normalization principles. (DBpedia, 2023). Unnormalized Form is the starting point in the normalization process and typically represent raw, unorganized data collected from different sources. Its purpose is to collect all the data.

- **Repeating Groups:** Repeating groups occur when a single row contains multiple instances of the same set of characters, all linked to a single primary key. This happens when redundant data is stored in a way that multiple related values are grouped together within one record.

  The steps that are carried out during U.N.F are:
  - All the attributes of the entities are listed.
  - The Primary Key is identified.
  - Repeating groups are identified.

In UNF (Unnormalized Form), I have decided to make student_id and Program_id as the primary key because a single primary key student_id cannot uniquely identify or define all the non-key attributes in the table

- **UNF**

  Student(<u>student_id</u>, student_name, phone_number, email, address, DOB, <u>Program_ID</u>, Program_Name, total_modules, duration, program_types,**{**module_id, module_name, credits_hours, modules_desc, module_type**{**resources_id, title, resources_type, resources_duration, sequence_number**},{**teacher_id, teacher_name, teacher_email, contact_number, department, specialization,**{**announcement_id, announcement_title, date_posted, announcement_desc, end_date**}},{**Assessment_id, assessment_name, ass_posted_date, ass_end_date, weightage, result_id, mark_obtained, full_mark, remarks, Feedback **}})**

3.2 1NF (First Normal Form)

A relation is in 1NF if all its attributes contain only atomic (single) values, disallowing multi-valued, composite attributes, or their combinations. It requires that each attribute in a table contains only atomic (indivisible) values, meaning attributes cannot hold multiple values or combinations of values. (Gibbs, 2022) . The primary purpose of 1NF is to eliminate redundancy and improve data clarity by structuring information in a way that avoids multi valued or composite attributes. The key advantage of 1NF include reducing duplicate data, simplifying data retrieval, ensuring consistency, and improving scalability, as it allows for easier addition of new attributes or records without disrupting the database structure.

A table is considered to be in 1NF if it satisfies the following conditions:

- **Atomic values:** The column in the tables must contain independent data known as an atomic value. This means the data in each column cannot be further broken down into smaller components. For example, a single column should contain one date, one number, or one word, rather than multiple values or a composite piece of information.
- **No Repeating Group:** A set of attributes, fields or data that are being repeated throughout a database table is known as repeating group. Repeating group are sign of unnormalized data which can lead database to data redundancy and data inconsistency so it must be remove in first normal form.

Note:

The”-1” in the name of each entity is for the identification for the first normal form and is not the actual name of the entity.

The underlined attributes represents the primary key of the table.

Attributes marked with asterisk (*) represent the foreign key.

The process that is carried out to normalized the database of E-classroom Platform in 1NF:

- **Student-program-1 (**student_id, student_name, phone_number, email, address, DOB, Program_ID, Program_Name, total_modules, duration, program_types)

In the above entity Student-Program student_id and Program_id was selected as primary key because student_id can only define the attributes of student like student_name, Phone_number, email, address and DOB. So another Primary key Program_ID is introduced to define the attribute of program like Program_Name, total_modules, duration and Program_types.

- **Program-Module-1**(modules_id, modules_name, credits_hours, modules_desc, module_type, program_id*, Student_id*)

In the above entity Program-Module-1 Modules_ID is Primary key because it can uniquely identify the attributes related to module. To handle the many to may relationship between program and module program_id is selected as composite key. Similarly student_id is also selected as composite key as these key can transferred to other dependent tables. This approach follows the left to right key transfer rule.

- **Resources-1**(resource_id, title, resource_type, duration, sequence_number, module_id*, **student_id***)

In the above tables Resources-1, Resources_id is introduced as primary key to uniquely identify the each resources and its attributes like title, resource_type, duration and sequence_number. Module_id and student_ID are selected as composite key. Module_id identify the module related with the resources and student_id indicates which student is accessing that resources. By including module_id and student_id as part of the composite key the table ensures that resources are properly linked to both modules and student.

- **Module-Teacher-1** (Teacher_id, teacher_name, teacher_email, contact_number, department, specialization, module_id*)

In the above table, Module-Teacher-1 teacher_id is introduced as primary key to uniquely identify the each resources and its attributes. To handle many to may relationship between module and teacher module_id is selected as composite key.

- **Announcement**-1(announcement_id, title, date_posted, announcement_desc, end_date, teacher_id*, module_id*)

In the above table, announcement_id is primary key as it uniquely identify each announcement as teacher_id and module_id is foreign key that reference the teacher and modules tables.

- **Assessment-Result-1**(Assessment_id, assessement_name, ass_posted_date, ass_end_date, weightage, result_id, mark_obtained, full_mark, remarks, grade, module_id*, student_id*)

In the above table Assessment-Result, Assessment_id is primary key as it uniquely identify each assessment. Module id is selected as

The final tables in 1nf are:

- **Student-program-1 (**student_id, student_name, phone_number, email, address, DOB, Program_ID, Program_Name, total_modules, duration, program_types)
- **Program-Module-1**(modules_id, modules_name, credits_hours, modules_desc, module_type, program_id*, Student_id*)
- **Resources-1**(resource_id, title, resource_type, duration, sequence_number, module_id*, **student_id***)
- **Module-Teacher-1** (Teacher_id, teacher_name, teacher_email, contact_number, department, specialization, module_id*)
- **Teacher-Announcement**-1(announcement_id, Announcement_title , date_posted, announcement_desc, end_date, teacher_id*, module_id*)
- **Assessment-Result-1**(Assessment_id, assessement_name, ass_posted_date, ass_end_date, weightage, result_id, mark_obtained, full_mark, remarks, Feedback, module_id*, student_id*)

3.3 2NF (Second Normal Form)

“Second Normal form is a process of database normalization in which partial dependency is eliminated. It was first introduced by a famous relational database expert Edgar F. Codd.” (Fayard, 2025). Before performing 2NF the table must be in 1NF. The main goal is to eliminate partial dependency and making full functional dependency in primary key or entire key. The purpose of 2NF is to remove redundancy caused by partial dependencies, making the data structure more organized and efficient to update.

The conditions to be in 2NF are:

- ➢ It must be 1NF.
- ➢ There should no partial dependencies.

The key point that should be considered in 2NF are Functional dependencies and its types.

- **Functional Dependencies:** Functional dependencies are the basic concept in database development. It is used to developed relationship between attributes and are also used to check the state of normalization in database that helps to reduce data redundancy and helps to improve data integrity. There are three types of Functional dependencies they are full functional, partial functional and transitive functional dependency.
  - ➢ **Full Functional dependency: “**A dependency X → Y is a full functional dependency if Y is functionally dependent on X, and removing any attributes from X means Y is no longer dependent on X. In other words, Y depends on the entirety of X and not on any subset of it**”** (Agarwal, 2023). For Example, Let’s consider a relation which has three attributes EmployeeID, ProjectID and HoursWorked. If the composite key EmployeeID, ProjectID uniquely determines HoursWorked, then (EmployeeID, ProjectID) → HoursWorked is full functional dependency. Here HoursWorked depend on both key not and single key.
  - ➢ **Partial Functional Dependency: “**A dependency X→Y is partial functional dependency if Y is functionally dependent on X, but Y also depends on a proper subset of X. This means Y does not rely on the entire composite key, only on a part of it **”** (Agarwal, 2023) For examples, Let’s consider above relation and has an attributes EmployeeName and (EmployeeID, ProjectID) → EmployeeName, But EmployeeName can be determined By

only EmployeeId so this is partial dependency. In this condition EmployeeName does not depends on the full composite key.

The process that is carried out to normalized the database of E-classroom Platform from 1NF to 2NF:

Student-program-1 (student_id, student_name, phone_number, email, address, DOB, Program_ID, Program_Name, total_modules, duration, program_types)

In the above table Student_program there is a composite key formed of two attributes student_id and Program_Id. So, we should check for Partial functional and full functional dependency.

P.F.D: Student_id → Student_name, Phone_Number, Email, Address, DOB

Here, the key attributes student_id can uniquely identify the non-key attributes like student_name, phone_number, email, address and DOB. Hence partial functional dependency exist.

P.F.D: Program_ID →Program_name, total_modules, duration, program_types

Here, the key attributes Program_id can uniquely identify the attribute of program like program_name. total_modules, duration, and program_types. Hence here also partial functional dependency exists.

F.F.D: (Student_ID, Program_ID) → ××

Hence the new tables that are formed are

- **Student-2 (** student_id, student_name, phone_number, email, address, DOB)
- **Program-2** (Program_ID, Program_Name, total_modules, duration, program_types)
- **Student-program-2**(student_id*, program_id*)

Program-Module-1(modules_id, modules_name, credits_hours, modules_desc, module_type, program_id*, Student_id*)

In the above table, there is an composite key formed by the attributes module_id, program_id and student_id. To analyze the functional dependencies:

P.F.D: Module_id → modules_name, credits_hours, modules_desc, module_type

Here, the key attributes module_id can uniquely identify the non-key attributes like modules_name, credits_hours, modules_desc, module_type. Hence partial functional dependency exists.

F.F.D: (Modules_id. Program_id, Student_id) → ××

Hence the resulting 2NF tables are:

- **Module-2**(modules_id, modules_name, credits_hours, modules_desc, module_type)
- **Program_module-2**(module_id*, program_id*, Student_id*)

Resources-1(resource_id, title, resource_type, duration, sequence_number, module_id*, **student_id***)

In the above table, resources_id is primary key and student_id is composite key. Here we have to analyze the functional dependences:

P.F.D: Resources_id → title, resources_type, duration,

The non-key attributes title, resource_type, duration are fully determined by resource_id alone , which indicates a partial functional dependency

F.F.D: (Resources_id, Student_ID → Sequence_number

The composite key formed by resources_id and student_id uniquely determines the sequence_number. This ensures full functional dependency.

Hence the resulting 2NF tables are:

- **Resources -2**(resource_id, title, resource_type, duration, module_id*)
- **Resources-student-2**(resources_id*, Student_id*, Sequence_number)

Module-Teacher-1 (Teacher_id, teacher_name, teacher_email, contact_number, department, specialization, module_id*)

In the above table, teacher_id is primary key and module_id is composite key. Here we have to analyze for partial functional dependency and full functional dependency:

P.F.D: Teacher_id → teacher_name, teacher_email, contact_number, department, specialization

The non key attributes teacher_name, teacher_email, contact_number, department and specialization are fully determined by teacher_id. This indicates a partial dependency.

F.F.D: (Teacher_id, Module_id) → ××

Here no other attributes depends on this composite key.

Hence the resulting tables in 2NF are:

- **Teacher-2**(Teacher_id, teacher_name, teacher_email, contact_number, department, specialization)
- **Module-Teacher-2**(teacher_id*, Module_id*)

Teacher-Announcement-1(announcement_id, Announcement_title,, date_posted, announcement_desc, end_date, teacher_id*, module_id*)

In the above table Teacher-Announcement-1, there is single primary key announcement_id. So all the non-key attributes i.e. title, date_posted, announcement_desc, end_date are fully functional dependent to announcement_id. Here, teacher_id and module_id remain as foreign keys to maintain relationships with their respective tables. Hence the resulting tables formed in 2NF is:

- **Announcement**-2(announcement_id, Announcement_title, date_posted, announcement_desc, end_date, teacher_id*, module_id*)

Assessment-Result-1(Assessment_id, assessement_name, ass_posted_date, ass_end_date, weightage, result_id, mark_obtained, full_mark, remarks, Feedback, module_id*, student_id*)

In the above table, Assessment-Result, there is composite key formed by module_id and Student_ID. So, we should check for partial dependency

P.F.D: Assessment_ID → assessement_name, ass_posted_date, ass_end_date, weightage

Here, the key attributes assessment_id can uniquely identify the non-key attributes like assessement_name, ass_posted_date, ass_end_date and weightage. Hence partial functional dependency exists.

F.F.D: (Assessment_id, module_id, student_id) → result_id, mark_obtained, full_mark, remarks, feedback

Here, assessment_id, module_id, student_id can gives result_id, mark_obtained, full_mark, remarks, and feedback. So here is full functional dependency.

Hence the resulting tables in 2NF are:

- **Assessment-2** (Assessment_ID assessement_name, ass_posted_date, ass_end_date, weightage)
- **Assessment-Result-2** (Assessment_id*, module_id*, student_id*, result_id, mark_obtained, full_mark, remarks, feedback)

The final tables in 2NF are:

- **Student-2 (** student_id, student_name, phone_number, email, address, DOB)
- **Program-2** (Program_ID, Program_Name, total_modules, duration, program_types)
- **Student-program-2**(student_id*, program_id*)
- **Module-2**(modules_id, modules_name, credits_hours, modules_desc, module_type)
- **Program_module-2**(module_id*, program_id*, Student_id*)
- **Resources -2**(resource_id, title, resource_type, duration, module_id*)
- **Resources-student-2**(resources_id*, Student_id*, Sequence_number)
- **Teacher-2**(Teacher_id, teacher_name, teacher_email, contact_number, department, specialization)
- **Module-Teacher-2**(teacher_id*, Module_id*)

- **Announcement-2** (announcement_id, Announcement_title, date_posted, announcement_desc, end_date, teacher_id*, module_id*)
- **Assessment-2** (Assessment_ID assessement_name, ass_posted_date, ass_end_date, weightage)
- **Assessment-Result-2** (Assessment_id, module_id*, student_id*, result_id, mark_obtained, full_mark, remarks, feedback)

3.4 3NF (Third Normal Form)

Third normal form is very important concept in database normalization that helps to remove unwanted dependency. It is builds upon first and second normal form mean the table must contain independent values in each cell and should not have any partial dependency. It make the database more normalized by eliminating transitive dependency.

Condition for 3 N.F:

- The table must be in Second Normal Form.
- It does not contain any transitive dependency.

The key concept that should consider in third normal form is transitive dependency.

Transitive functional Dependency: "A dependency X → Y is a transitive functional dependency if Y is functionally dependent on X through an intermediate attributes Z." (Agarwal, 2023) For example, let's us consider an table consisting of attributes like order_id, order_desc, price, customer_id and name. Here customer_id can uniquely identify the name of the customer. That why here is transitive dependency.

The process that is carried out to normalized the database of E-classroom Platform from 2NF to 3NF:

Student-2 (student_id, student_name, phone_number, email, address, DOB)

In above table,

Student_name → ××

The attributes student_name does not have ability to uniquely determine any other non-key attributes. Therefore, transitive dependency is absent here.

Phone_number → ××

Here, phone_number cannot uniquely identify any other non-key attributes, hence no transitive dependency exist here.

Email → ××

Here, email cannot uniquely identify any other non-key attributes. Hence on existence of transitive dependency.

DOB → ××

The attributes DOB cannot uniquely determine any other non-key attributes. Hence there is no presence of transitive dependency.

Therefore, the resulting table in 3NF:

- **Student-3 (**student_id, student_name, phone_number, email, address, DOB)

Program-2 (Program_ID, Program_Name, total_modules, duration, program_types)

In the above table,

Program_name →××

The attributes program_name cannot uniquely determine any other attributes. Thus, there is no transitive dependency.

Total_modules→××

The attribute total_modules cannot uniquely identify any other attributes. Thus, there is no transitive dependency.

Duration→××

The attribute duration cannot uniquely identify any other non attributes. Thus, there is no transitive dependency.

Program_types→××

The attribute program_types cannot uniquely identify any other attributes. Thus, there is no transitive dependency.

Hence the resulting table in 3NF:

- **Program-3** (Program_ID, Program_Name, total_modules, duration, program_types)
-

Student-program-2(student_id*, program_id*)

In the above table, Student_program there is no non key attribute. Therefore it has no transitive dependency. Hence the resulting table in 3NF is same

Module-2(modules_id, module_name, credits_hours, module_desc, module_type)

In the above table,

Module_name →××

The attribute module_name cannot able determine any other attributes. Hence it does not contribute to any transitive dependency.

Credits_hours → ××

The attribute credits_hours cannot able to uniquely identify any other non-key attributes. Hence it does not contribute to any transitive dependency.

Module_desc → ××

The attribute module_desc cannot able to uniquely identify any other non-key attribute. Hence it does not contribute to any transitive dependency.

Module_type →××

The attributes module_type cannot able to uniquely identify any other non-key attribute. Hence it does not contribute to any transitive dependency.

Hence the resulting table in 3NF:

- **Module-3**(modules_id, modules_name, credits_hours, modules_desc, module_type)

Program_module-3(module_id*, program_id*, Student_ID*)

In the above table, Program-Module, there are no any non-key attributes. So the table has no transitive dependency. Hence the table is in 3NF.

Resources -2(resource_id, title, resource_type, duration, module_id*)

In the above table,

Title → ××

Here, title cannot uniquely identify any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Resources_type → ××

Here, resources_type cannot uniquely identify any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Duration → ××

Here, duration cannot uniquely identify any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Hence, the resulting table in 3NF:

- ➢ **Resources -3**(resource_id, title, resource_type, duration, module_id*)

Resources-student-2(resources_id*, Student_id*, Sequence_number)

In above table,

Sequence_number →××

Here, sequence_number cannot uniquely identify any other non-key attributes. Hence her is no transitive dependency.

Hence the resulting table in 3NF:

- ➢ **Resources-student-3**(resources_id*, Student_id*, Sequence_number)

Teacher-2(Teacher_id, teacher_name, teacher_email, contact_number, department, specialization)

In above table,

Teacher_name → ××

Teacher_name cannot uniquely determine anu other non-key attributes. Hence it does not contribute to any transitive dependency.

Teacher_email → ××

Teacher_email cannot uniquely determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Contact_number → ××

Contact_number cannot uniquely determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Department → ××

Department cannot uniquely determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Specialization → ××

Specialization cannot uniquely determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Hence the resulting table in 3NF:

- **Teacher-3**(<u>Teacher_id,</u> teacher_name, teacher_email, contact_number, department, specialization)

Module-Teacher-2(<u>teacher_id,</u> Module_id)

In the above table, module-Teacher-2 there is no anykey attributes. So it has no any transitive dependency. Hence the table in 3NF remains same.

Announcement-2 (<u>announcement_id</u>, title, date_posted, announcement_desc, end_date, teacher_id*, module_id*)

In the above table,

Announcement_title, → ××

Announcement_title, cannot uniquely determine any other attributes. . Hence it does not contribute to any transitive dependency.

Date_posted → ××

Date_posted cannot uniquely determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Announcement_desc → ××

Announcement_desc cannot uniquely determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

End_date → ××

End_date cannot uniquely determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Hence, the resulting table in 3NF:

- **Announcement-3** (announcement_id, title, date_posted, announcement_desc, end_date, teacher_id*, module_id*)

Assessment-2 (Assessment_ID assessement_name, ass_posted_date, ass_end_date, weightage)

In the above table,

Assessment_name →××

Here, assessment_name cannot determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Ass_posted_date → ××

Here, ass_posted_date cannot determine any other non-key attributes. Hence it does not contribute to any transitive dependency.

Ass_end_date → ××

Here, ass_end_date cannot determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Weightage → ××

Here, weightage cannot determine any other non-key attributes. . Hence it does not contribute to any transitive dependency.

Hence the resulting table in 3NF is:

- **Assessment-2** (<u>Assessment_ID</u> assessement_name, ass_posted_date, ass_end_date, weightage)

Assessment-Result-2 (<u>Assessment_id</u>, <u>module_id</u>*, <u>student_id*</u>, result_id, mark_obtained, full_mark, remarks, feedback)

In the above table,

Result_id → remark, feedback

Here, result_id can determine remark and feedback. Hence here exist transitive dependency. So we can create new table as

Result-3 (<u>result_id</u>, remarks, feedback)

Mark_obtained →××

Here, mark_obtained cannot uniquely identify any other attributes. . Hence it does not contribute to any transitive dependency.

Mark_obtained →××

Here, mark_obtained cannot uniquely identify any other attributes. . Hence it does not contribute to any transitive dependency.

Full_mark →××

Here, full_mark cannot uniquely identify any other attributes. No transitive dependency.

Remarks → ××

Here, remarks cannot determine any other attributes. No transitive dependency

Feedback → ××

Here, feedback cannot determine any other attributes. So here is no transitive dependency.

Hence, the resulting tables in 3NF are:

- **Result-3** (result_id, remarks, feedback)
- **Assessment-Result-3** (Assessment_id, module_id, student_id, result_id, mark_obtained, full_mark)

The final tables in 3NF:

- **Student-3 (**student_id, student_name, phone_number, email, address, DOB)
- **Program-3** (Program_ID, Program_Name, total_modules, duration, program_types)
- **Student-program-3**(student_id*, program_id*)
- **Module-3**(modules_id, modules_name, credits_hours, modules_desc, module_type)
- **Program_module-3**(module_id*, program_id*, Student_ID*)
- **Resources -3**(resource_id, title, resource_type, duration, module_id*)
- **Resources-student-3**(resources_id*, Student_id*, Sequence_number)
- **Teacher-3**(Teacher_id, teacher_name, teacher_email, contact_number, department, specialization)
- **Module-Teacher-3**(teacher_id, Module_id)
- **Announcement-3** (announcement_id, Announcement_title, date_posted, announcement_desc, end_date, teacher_id*, module_id*)
- **Assessment-2** (Assessment_ID assessement_name, ass_posted_date, ass_end_date, weightage)
- **Assessment-Result-3** (Assessment_id, module_id, student_id, result_id, mark_obtained, full_mark)
- **Result-3** (result_id, remarks, feedback)

4 Final ERD

The final ERD is the ERD that is drawn after normalization. It is well structured and relationships are clearly defined with foregin key. In final ERD data is normalized and repeated data are eliminated. Data Integrity is also enforced by introducing proper foregin key. Overall this all help to improve the performance database. The final ERD for Barcelona International Collage's database is presented below:

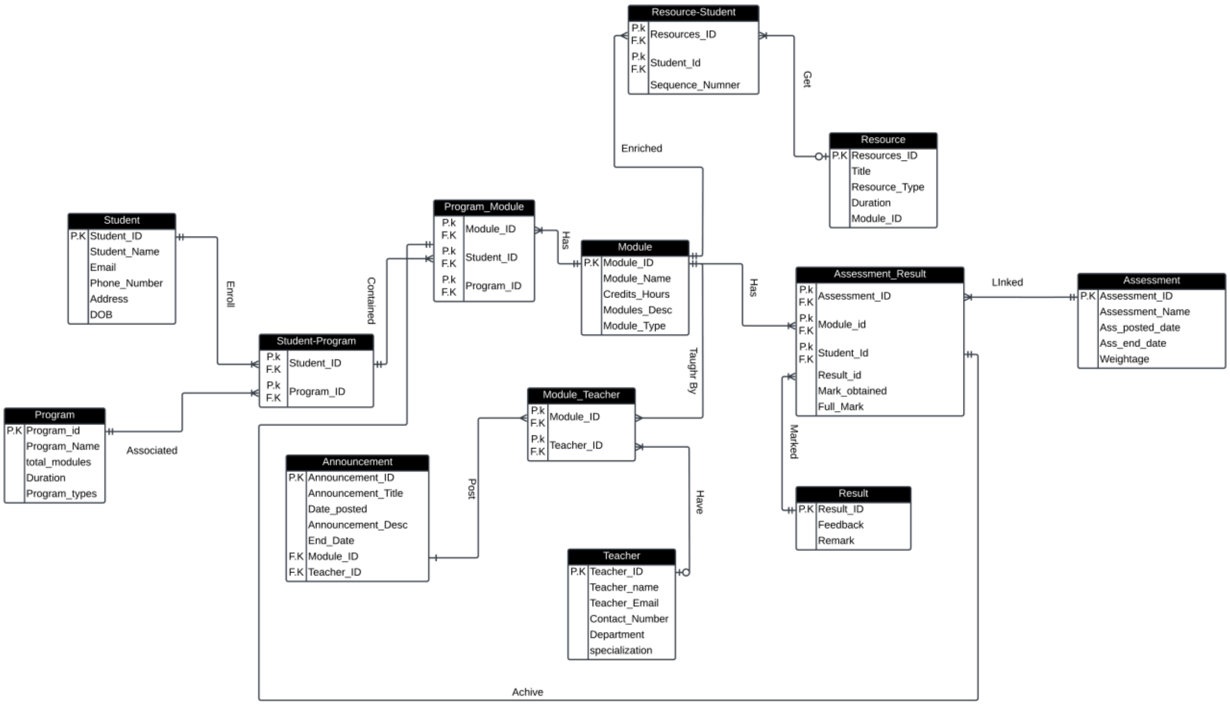


Figure 3: Final ERD.

4.1 Data Dictionary

A data Dictionary is a detailed documentation of metadata that explains the structure and purpose of the data being used. It was first introduced in the 1960s, where its was simple file documentation. Over the time, they developed into more advanced repositories, offering a comprehensive view of metadata. (Chia, 2023). It contains all the information about the table, its attributes and constraints. As it contains all the information of database schema, it an essential tool in data management which serves as a centralized repository. It generally includes description of the data elements, such as their names, types, formats, relationships and constraints.

The Data Dictionary that is used while developing the database of E-Classroom are:

1. Data Dictionary for Student Table

Table 4: Data dictionary for student Table.

S.N	Attributes	Datatype	Size	Constraints
1	studentID	NUMBER	10	PRIMARY KEY, NOT NULL
2	Student_name	CHARACTER	20	NOT NULL
3	Phone_Number	CHARACTER	20	NOT NULL
4	Email	CHARACTER	30	NOT NULL, UNIQUE
5	Address	CHARACTER	20	NOT NULL
6	DOB	DATE	N/A	NOT NULL,

2. Data Dictionary for Program Table

Table 5: Data Dictionary for program Table

S.N	Attributes	Datatype	Size	Constraints
1	program_ID	NUMBER	10	PRIMARY KEY, NOT NULL
2	program_name	CHARACTER	20	NOT NULL
3	duration	NUMBER	10	NOT NULL
4	total_modules	NUMBER	10	NOT NULL
5	Program_Type	CHARACTER	20	NOT NULL

3. Data Dictionary for Student_program tables

Table 6: Data Dictionary for Student_Program Table

S.N	Attributes	Datatype	Size	Constraints
1	program_ID	NUMBER	10	PRIMARY KEY, FOREIGN KEY, NOT NULL
2	Student_ID	NUMBER	10	PRIMARY KEY, FOREIGN KEY, NOT NULL

4. Data Dictionary for Module Table

Table 7: Data Dictionary for Module Table

S.N	Attributes	Datatype	Size	Constraints
1	module_ID	CHARACTER	10	PRIMARY KEY, NOT NULL
2	module_name	CHARACTER	20	NOT NULL
3	credits_hours	NUMBER	10	NOT NULL
4	Module_Type	CHARACTER	20	NOT NULL
5	Module_Desc	CHARACTER	35	NULLABLE

5. Data Dictionary for program-Module table

Table 8: Data Dictionary for Program-Module Table

S. N	Attributes	Datatype	Size	Constraints
1	module_ID	CHARACTER	10	PRIMARY KEY, FOREGIN KEY, NOT NULL
2	Student_ID	NUMBER	10	PRIMARY KEY, FOREGIN KEY, NOT NULL
2	Program_ID	NUMBER	10	PRIMARY KEY, FOREGIN KEY, NOT NULL

6. Data Dictionary for Resources Table

Table 9: Data Dictionary for Resources Table

S.N	Attribute Name	Datatype	Size	Constraints
1	resources_ID	CHARACTER	10	PRIMARY KEY, NOT NULL
2	title	CHARACTER	40	NOT NULL
3	resources_type	CHARACTER	10	NOT NULL
4	duration	NUMBER	10	NOT NULL
5	Module_id	CHARACTER	10	FOREGIN KEY, NOT NULL

7. Data Dictionary for Resources-Student Table

Table 10: Data Dictionary for Resource_Student Table

S.N	Attribute Name	Datatype	Size	Constraints
1	resources_ID	CHARACTER	10	PRIMARY KEY, NOT NULL
2	Student_ID	NUMBER	10	PRIMARY KEY
3	sequence_number	CHARACTER	10	NOT NULL

8. Data Dictionary for Teacher Table

Table 11: Data Dictionary for Teacher Table

S.N	Attributes	Datatype	Size	Constraints
1	teacher_ID	NUMBER	10	PRIMARY KEY, NOT NULL
2	teacher_name	CHARACTER	20	NOT NULL
3	Teacher_email	CHARACTER	25	NOT NULL, UNIQUE
4	Contact_number	CHARACTER	15	NOT NULL
5	department	CHARACTER	20	NOT NULL
6	specialization	CHARACTER	15	NOT NULL

9. Data Dictionary for Module-Teacher

Table 12: Data Dictionary for Module-Teacher Table

S. N	Attributes	Datatype	Size	Constraints
1	module_ID	CHARACTER	10	PRIMARY KEY, FOREGIN KEY, NOT NULL
2	Teacher_id	NUMBER	10	PRIMARY KEY, FOREGIN KEY, NOT NULL

10. Data Dictionary for Announcement Table

Table 13: Data Dictionary for Assessment Table

S.N	Attribute Name	Datatype	Size	Constraints
1	announcement_ID	CHARACTER	10	PRIMARY KEY, NOT NULL
2	title	CHARACTER	35	NOT NULL
3	date_posted	DATE	N/A	NOT NULL
4	end_date	DATE	N/A	NOT NULL
5	announcement_desc	CHARACTER	45	NOT NULL
6	teacher_ID	NUMBER	10	FOREGIN KEY, NOT NULL
7	module_ID	CHARACTER	10	FOREGIN KEY, NOT NULL

11. Data Dictionary for Assessment Table

Table 14: Data Dictionary for Assessment Table

S.N	Attributes	Datatype	Size	Constraints
1	assessment_ID	CHARACTER	10	PRIMARY KEY, NOT NULL
2	Assessment_name	CHARACTER	25	NOT NULL
3	Asmt_posted_date	DATE	N/A	NOT NULL
4	Asmt_end_date	DATE	N/A	NOT NULL
5	Weightage	NUMBER	10	NOT NULL

12. Data Dictionary for Result Table

Table 15: Data Dictionary for Result Table

S.N	Attribute Name	Datatype	Size	Constraints
1	result_ID	CHARACTER	10	PRIMARY KEY, NOT NULL
4	remarks	CHARACTER	25	NULLABLE
5	feedback	CHARACTER	25	NOT NULL

13. Assessment-Result

Table 16: Data Dictionary for Assessment_Result Table.

S.N	Attribute Name	Datatype	Size	Constraints
1	Assessment_ID	CHARACTER	10	PRIMARY KEY, NOT NULL, FOREIGN KEY
2	Module_ID	CHARACTER	10	PRIMARY KEY, NOT NULL, FOREIGN KEY
3	Student_ID	NUMBER	10	PRIMARY KEY, NOT NULL, FOREIGN KEY
4	Result_ID	CHARACTER	10	NOT NULL, UNIQUE, FOREIGN KEY
5	Mark_obtained	NUMBER	10	NOT NULL
6	Full_mark	NUMBER	10	NOT NULL

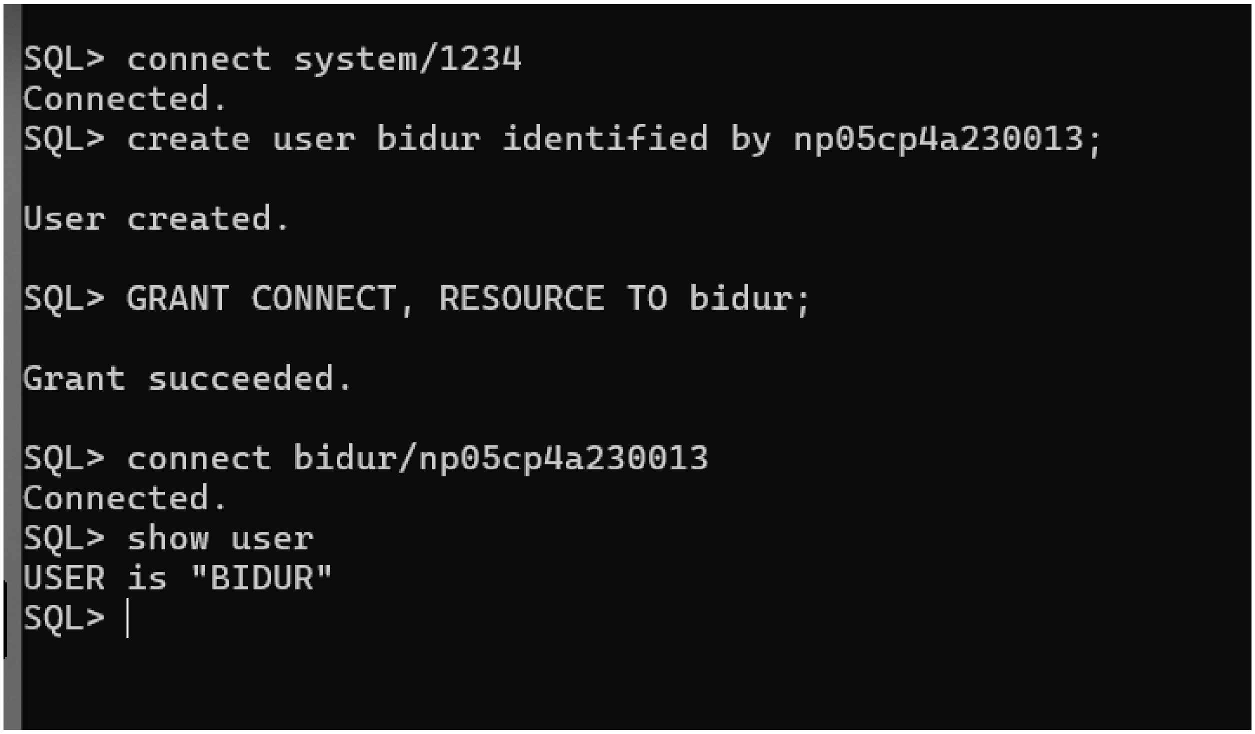
5 Implementation

For the implementation of database, we are using Oracle SQL plus database. Oracle SQL is a popular relational database and is renowned for its reliability, scalability and it is easy to use.

DDL Stands for Data Definition Language which is an subset for SQL and is used to define the structure of the database. The primary DDL command are create, alter, drop and truncate. To design the database of E-classroom platform a user with name bidur is created and is identified with password “np05cp4a230013”. After than the privilege is granted to the user

Creating user and Giving Privilege

```
CREATE user bidur IDENTIFIED BY np05cp4a230013;
GRANT CONNECT, RESOURCE TO bidur;
```

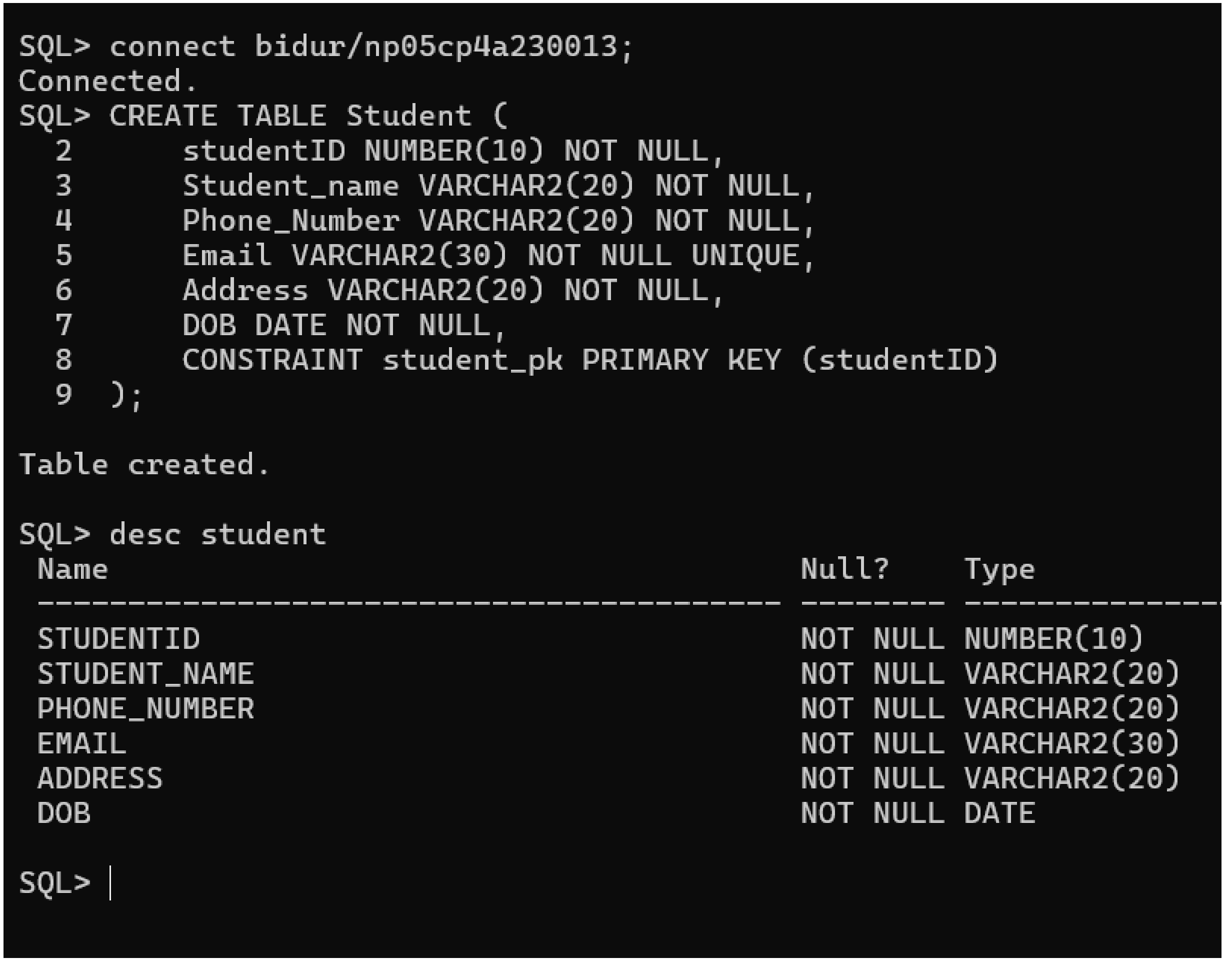


```
SQL> connect system/1234
Connected.
SQL> create user bidur identified by np05cp4a230013;

User created.

SQL> GRANT CONNECT, RESOURCE TO bidur;

Grant succeeded.

SQL> connect bidur/np05cp4a230013
Connected.
SQL> show user
USER is "BIDUR"
SQL>
```

Figure 4: Creating user and privilege management.

5.1 Table creation

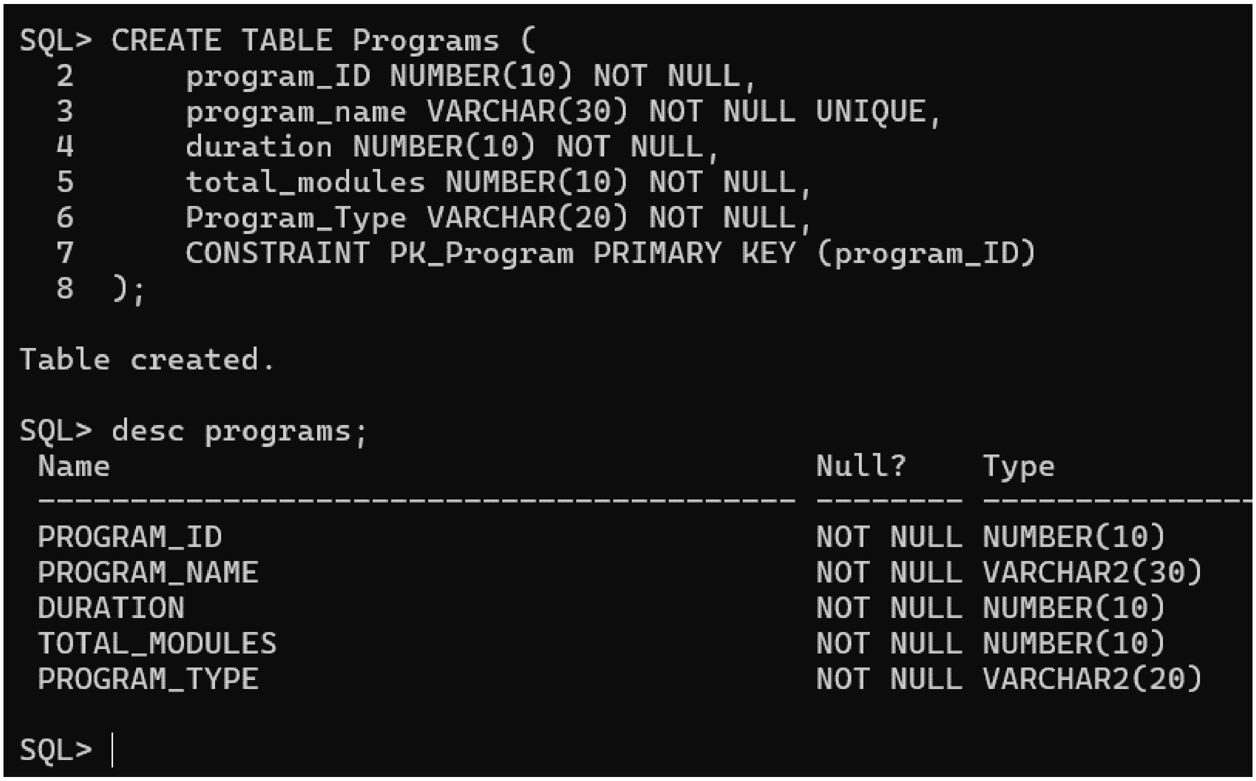
- **Creating Student Table**



```
SQL> connect bidur/np05cp4a230013;
Connected.
SQL> CREATE TABLE Student (
  2      studentID NUMBER(10) NOT NULL,
  3      Student_name VARCHAR2(20) NOT NULL,
  4      Phone_Number VARCHAR2(20) NOT NULL,
  5      Email VARCHAR2(30) NOT NULL UNIQUE,
  6      Address VARCHAR2(20) NOT NULL,
  7      DOB DATE NOT NULL,
  8      CONSTRAINT student_pk PRIMARY KEY (studentID)
  9  );

Table created.

SQL> desc student
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------
 STUDENTID                                 NOT NULL NUMBER(10)
 STUDENT_NAME                              NOT NULL VARCHAR2(20)
 PHONE_NUMBER                              NOT NULL VARCHAR2(20)
 EMAIL                                     NOT NULL VARCHAR2(30)
 ADDRESS                                   NOT NULL VARCHAR2(20)
 DOB                                       NOT NULL DATE

SQL> |
```

Figure 5: Student Table Creation.

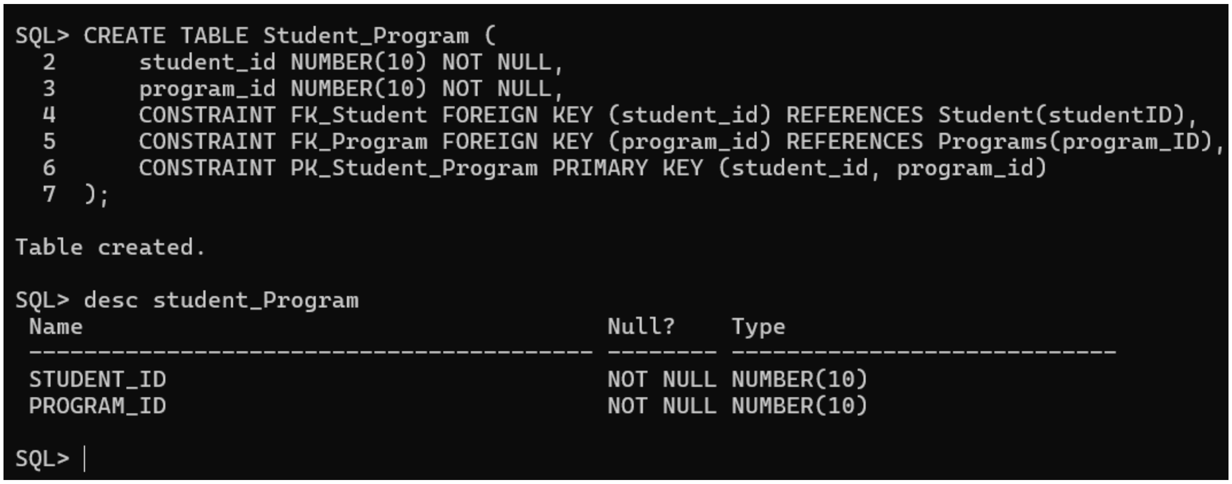
- **Creating Programs tables**



```
SQL> CREATE TABLE Programs (
  2      program_ID NUMBER(10) NOT NULL,
  3      program_name VARCHAR(30) NOT NULL UNIQUE,
  4      duration NUMBER(10) NOT NULL,
  5      total_modules NUMBER(10) NOT NULL,
  6      Program_Type VARCHAR(20) NOT NULL,
  7      CONSTRAINT PK_Program PRIMARY KEY (program_ID)
  8  );

Table created.

SQL> desc programs;
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------
 PROGRAM_ID                                NOT NULL NUMBER(10)
 PROGRAM_NAME                              NOT NULL VARCHAR2(30)
 DURATION                                  NOT NULL NUMBER(10)
 TOTAL_MODULES                             NOT NULL NUMBER(10)
 PROGRAM_TYPE                              NOT NULL VARCHAR2(20)

SQL> |
```

Figure 6:Creating Program tables.

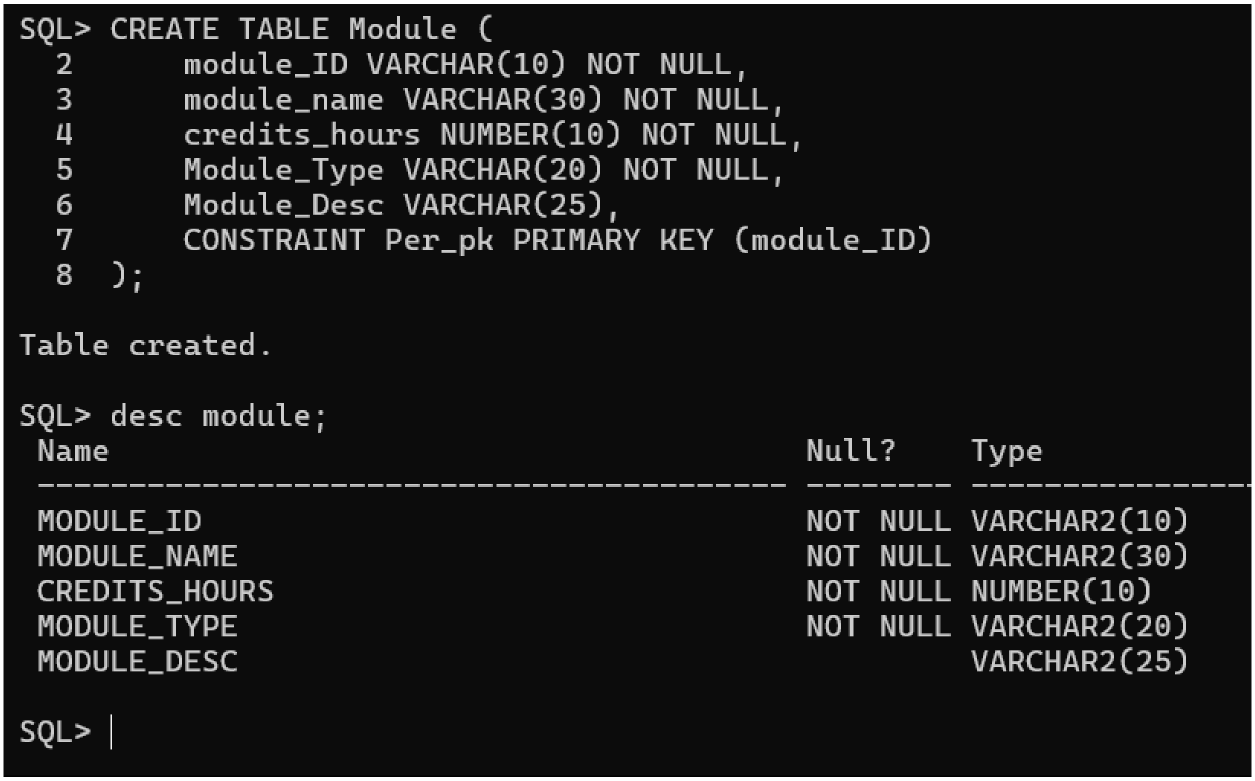
- **Creating student-program table.**



```
SQL> CREATE TABLE Student_Program (
  2      student_id NUMBER(10) NOT NULL,
  3      program_id NUMBER(10) NOT NULL,
  4      CONSTRAINT FK_Student FOREIGN KEY (student_id) REFERENCES Student(studentID),
  5      CONSTRAINT FK_Program FOREIGN KEY (program_id) REFERENCES Programs(program_ID),
  6      CONSTRAINT PK_Student_Program PRIMARY KEY (student_id, program_id)
  7  );

Table created.

SQL> desc student_Program
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(10)
 PROGRAM_ID                                NOT NULL NUMBER(10)

SQL> |
```

Figure 7:   Student_Program Tables.

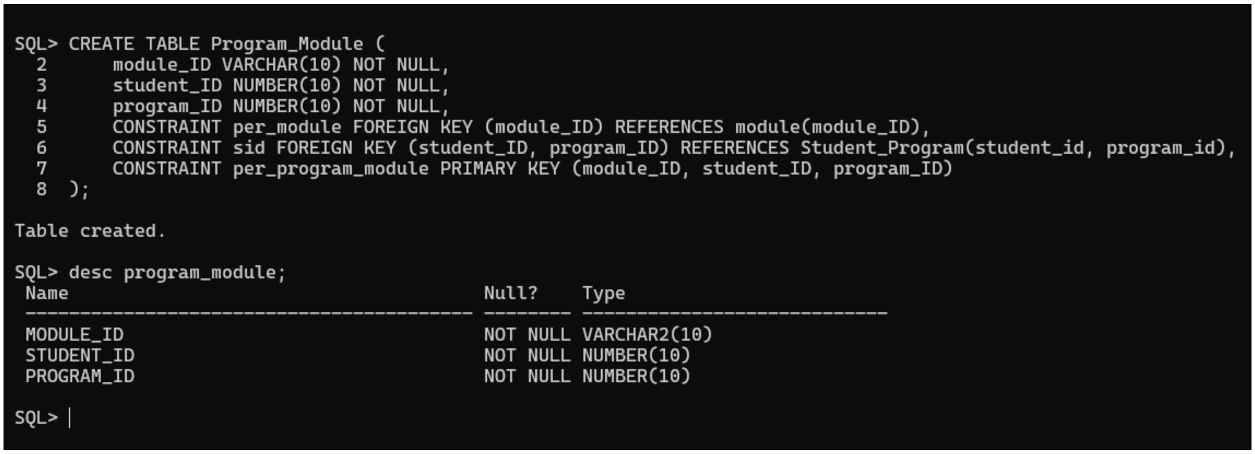
- **Creating Module table.**



```
SQL> CREATE TABLE Module (
  2      module_ID VARCHAR(10) NOT NULL,
  3      module_name VARCHAR(30) NOT NULL,
  4      credits_hours NUMBER(10) NOT NULL,
  5      Module_Type VARCHAR(20) NOT NULL,
  6      Module_Desc VARCHAR(25),
  7      CONSTRAINT Per_pk PRIMARY KEY (module_ID)
  8  );

Table created.

SQL> desc module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 MODULE_ID                                 NOT NULL VARCHAR2(10)
 MODULE_NAME                               NOT NULL VARCHAR2(30)
 CREDITS_HOURS                             NOT NULL NUMBER(10)
 MODULE_TYPE                               NOT NULL VARCHAR2(20)
 MODULE_DESC                                        VARCHAR2(25)

SQL>
```

Figure 8: Creating Module Table

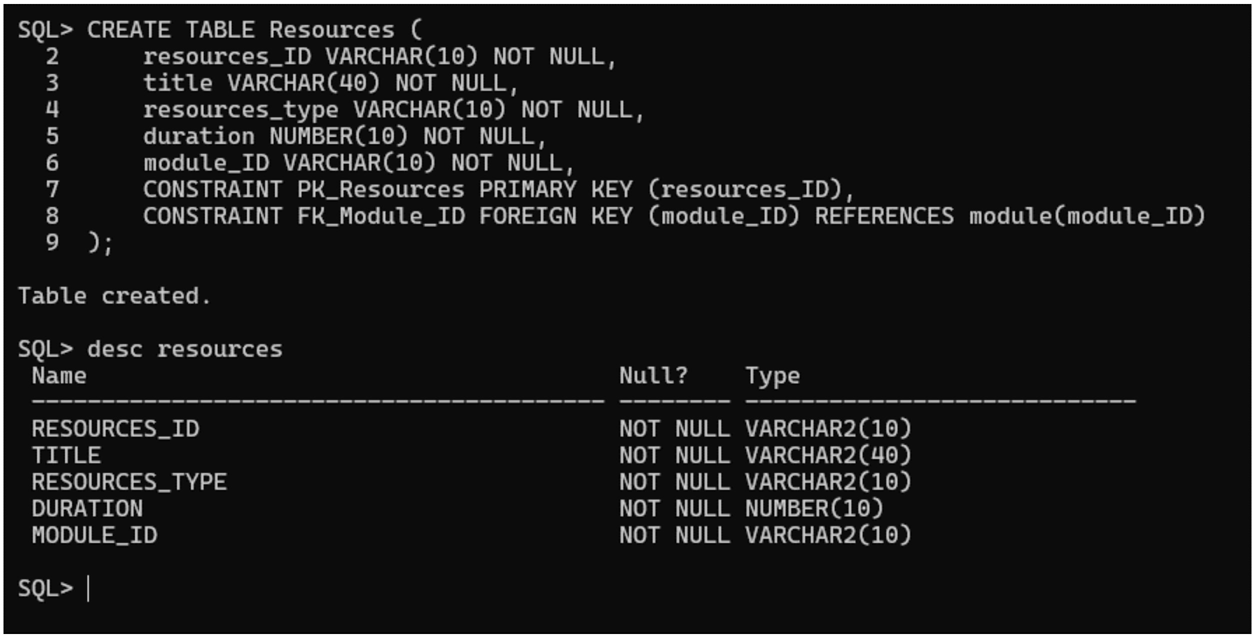
- **Creating Program-Module Table**



```
SQL> CREATE TABLE Program_Module (
  2      module_ID VARCHAR(10) NOT NULL,
  3      student_ID NUMBER(10) NOT NULL,
  4      program_ID NUMBER(10) NOT NULL,
  5      CONSTRAINT per_module FOREIGN KEY (module_ID) REFERENCES module(module_ID),
  6      CONSTRAINT sid FOREIGN KEY (student_ID, program_ID) REFERENCES Student_Program(student_id, program_id),
  7      CONSTRAINT per_program_module PRIMARY KEY (module_ID, student_ID, program_ID)
  8  );

Table created.

SQL> desc program_module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODULE_ID                                 NOT NULL VARCHAR2(10)
 STUDENT_ID                                NOT NULL NUMBER(10)
 PROGRAM_ID                                NOT NULL NUMBER(10)

SQL>
```

Figure 9: Creating Program_Module Table

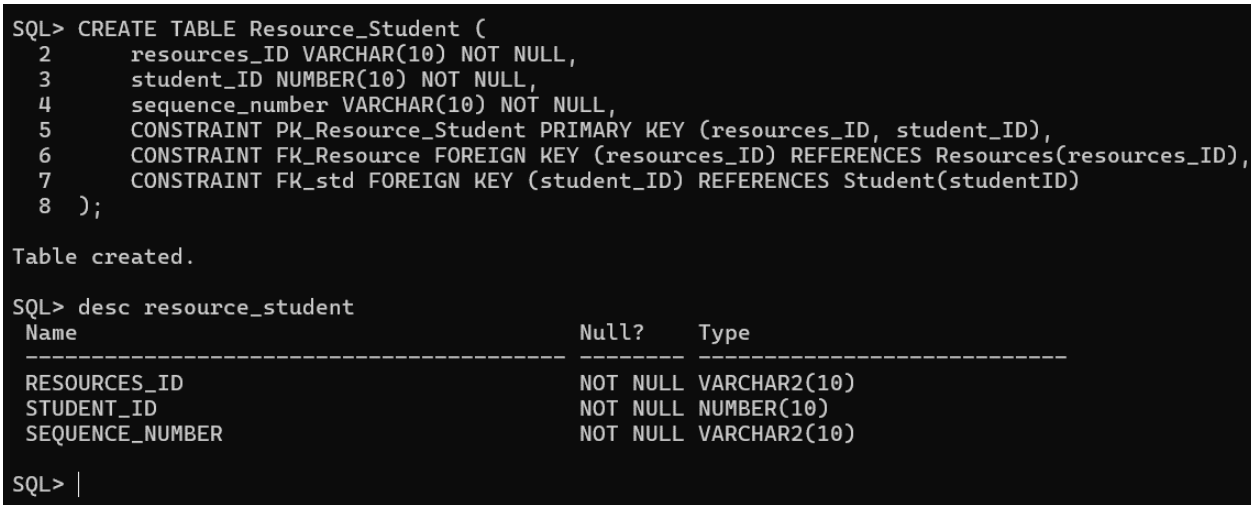
- **Creating Resources Table**



```
SQL> CREATE TABLE Resources (
  2      resources_ID VARCHAR(10) NOT NULL,
  3      title VARCHAR(40) NOT NULL,
  4      resources_type VARCHAR(10) NOT NULL,
  5      duration NUMBER(10) NOT NULL,
  6      module_ID VARCHAR(10) NOT NULL,
  7      CONSTRAINT PK_Resources PRIMARY KEY (resources_ID),
  8      CONSTRAINT FK_Module_ID FOREIGN KEY (module_ID) REFERENCES module(module_ID)
  9  );

Table created.

SQL> desc resources
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RESOURCES_ID                              NOT NULL VARCHAR2(10)
 TITLE                                     NOT NULL VARCHAR2(40)
 RESOURCES_TYPE                            NOT NULL VARCHAR2(10)
 DURATION                                  NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL VARCHAR2(10)

SQL> |
```

Figure 10: Creating Resources Table.

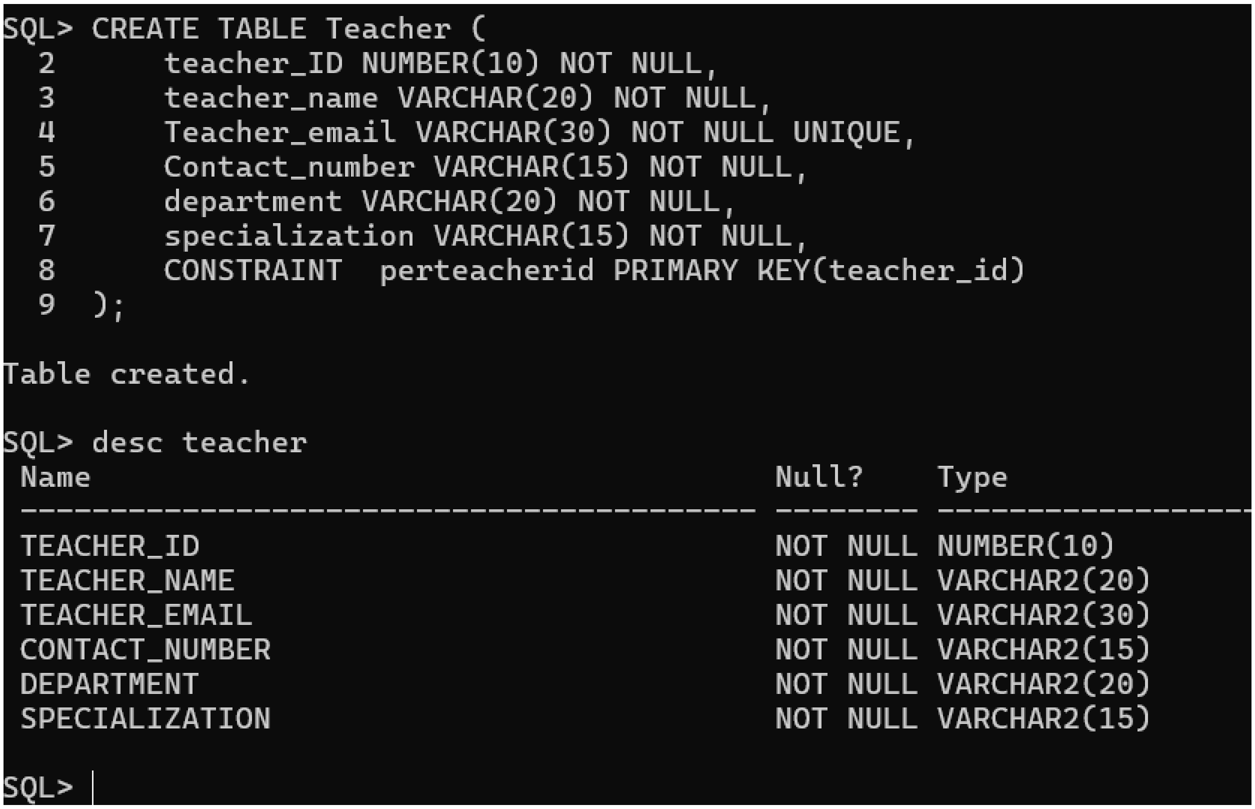
- **Creating Resources-Student**



```
SQL> CREATE TABLE Resource_Student (
  2      resources_ID VARCHAR(10) NOT NULL,
  3      student_ID NUMBER(10) NOT NULL,
  4      sequence_number VARCHAR(10) NOT NULL,
  5      CONSTRAINT PK_Resource_Student PRIMARY KEY (resources_ID, student_ID),
  6      CONSTRAINT FK_Resource FOREIGN KEY (resources_ID) REFERENCES Resources(resources_ID),
  7      CONSTRAINT FK_std FOREIGN KEY (student_ID) REFERENCES Student(studentID)
  8  );

Table created.

SQL> desc resource_student
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RESOURCES_ID                              NOT NULL VARCHAR2(10)
 STUDENT_ID                                NOT NULL NUMBER(10)
 SEQUENCE_NUMBER                           NOT NULL VARCHAR2(10)

SQL> |
```

Figure 11: Creating Resource_Student table.

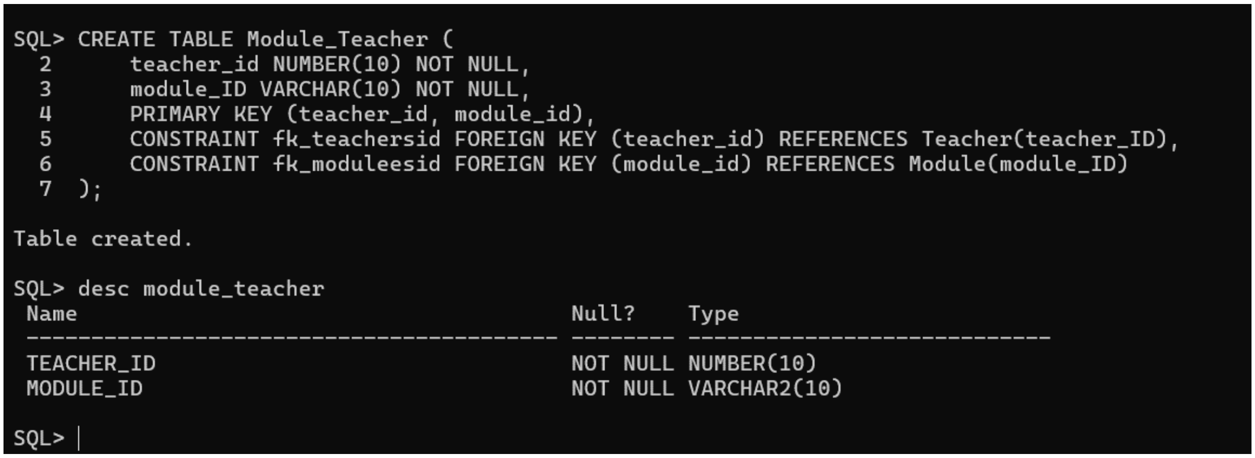
- **Creating teacher tables**



```
SQL> CREATE TABLE Teacher (
        teacher_ID NUMBER(10) NOT NULL,
        teacher_name VARCHAR(20) NOT NULL,
        Teacher_email VARCHAR(30) NOT NULL UNIQUE,
        Contact_number VARCHAR(15) NOT NULL,
        department VARCHAR(20) NOT NULL,
        specialization VARCHAR(15) NOT NULL,
        CONSTRAINT  perteacherid PRIMARY KEY(teacher_id)
    );

Table created.

SQL> desc teacher
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TEACHER_ID                                NOT NULL NUMBER(10)
 TEACHER_NAME                              NOT NULL VARCHAR2(20)
 TEACHER_EMAIL                             NOT NULL VARCHAR2(30)
 CONTACT_NUMBER                            NOT NULL VARCHAR2(15)
 DEPARTMENT                                NOT NULL VARCHAR2(20)
 SPECIALIZATION                            NOT NULL VARCHAR2(15)

SQL>
```

Figure 12: Creating teacher table.

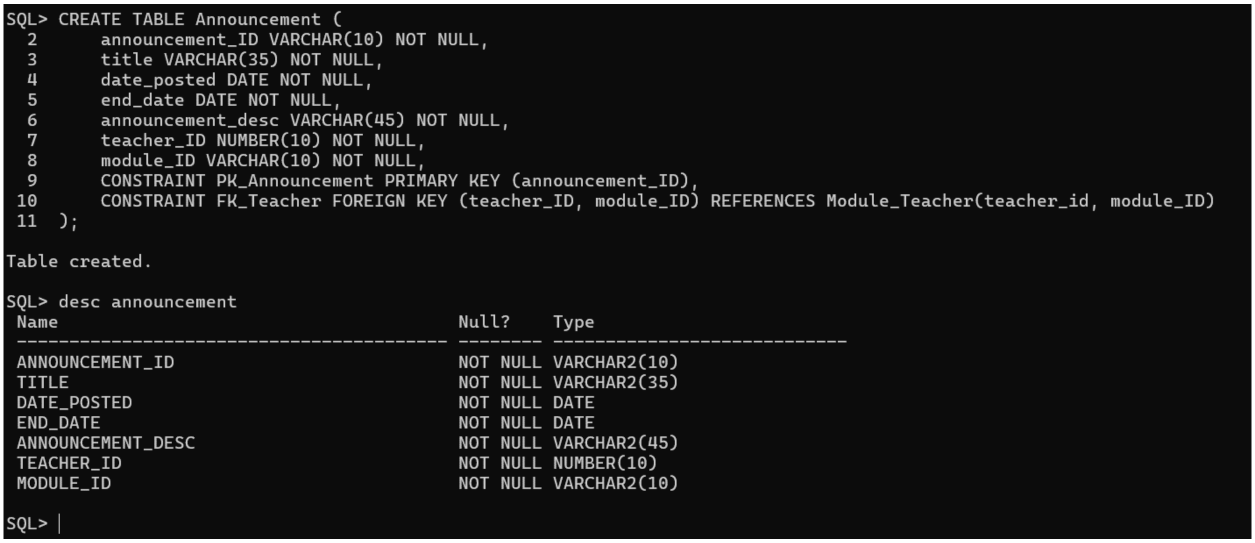
- **Creating Module-teacher tables**



```
SQL> CREATE TABLE Module_Teacher (
        teacher_id NUMBER(10) NOT NULL,
        module_ID VARCHAR(10) NOT NULL,
        PRIMARY KEY (teacher_id, module_id),
        CONSTRAINT fk_teachersid FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_ID),
        CONSTRAINT fk_moduleesid FOREIGN KEY (module_id) REFERENCES Module(module_ID)
    );

Table created.

SQL> desc module_teacher
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TEACHER_ID                                NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL VARCHAR2(10)

SQL>
```

Figure 13: Creating module-teacher table.

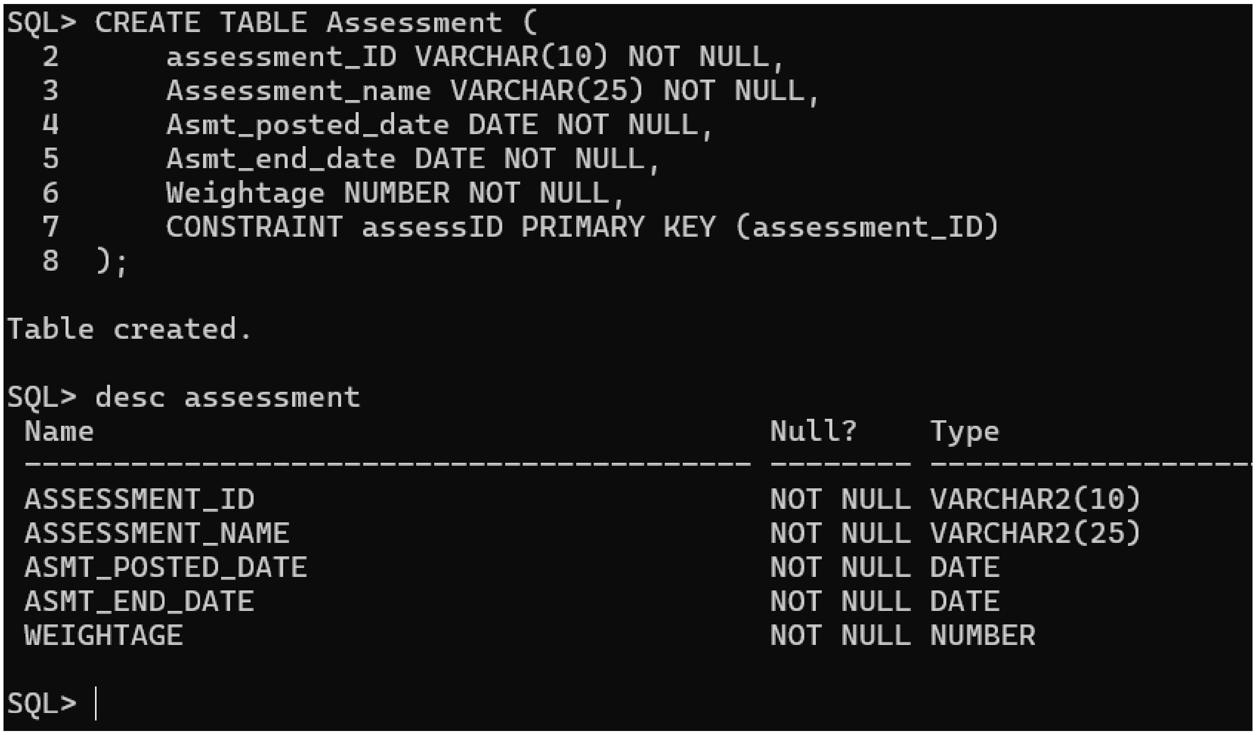
- **Creating Announcement table**



```
SQL> CREATE TABLE Announcement (
        announcement_ID VARCHAR(10) NOT NULL,
        title VARCHAR(35) NOT NULL,
        date_posted DATE NOT NULL,
        end_date DATE NOT NULL,
        announcement_desc VARCHAR(45) NOT NULL,
        teacher_ID NUMBER(10) NOT NULL,
        module_ID VARCHAR(10) NOT NULL,
        CONSTRAINT PK_Announcement PRIMARY KEY (announcement_ID),
        CONSTRAINT FK_Teacher FOREIGN KEY (teacher_ID, module_ID) REFERENCES Module_Teacher(teacher_id, module_ID)
    );

Table created.

SQL> desc announcement
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ANNOUNCEMENT_ID                           NOT NULL VARCHAR2(10)
 TITLE                                     NOT NULL VARCHAR2(35)
 DATE_POSTED                               NOT NULL DATE
 END_DATE                                  NOT NULL DATE
 ANNOUNCEMENT_DESC                         NOT NULL VARCHAR2(45)
 TEACHER_ID                                NOT NULL NUMBER(10)
 MODULE_ID                                 NOT NULL VARCHAR2(10)

SQL> |
```

Figure 14: Creating Announcement Table.

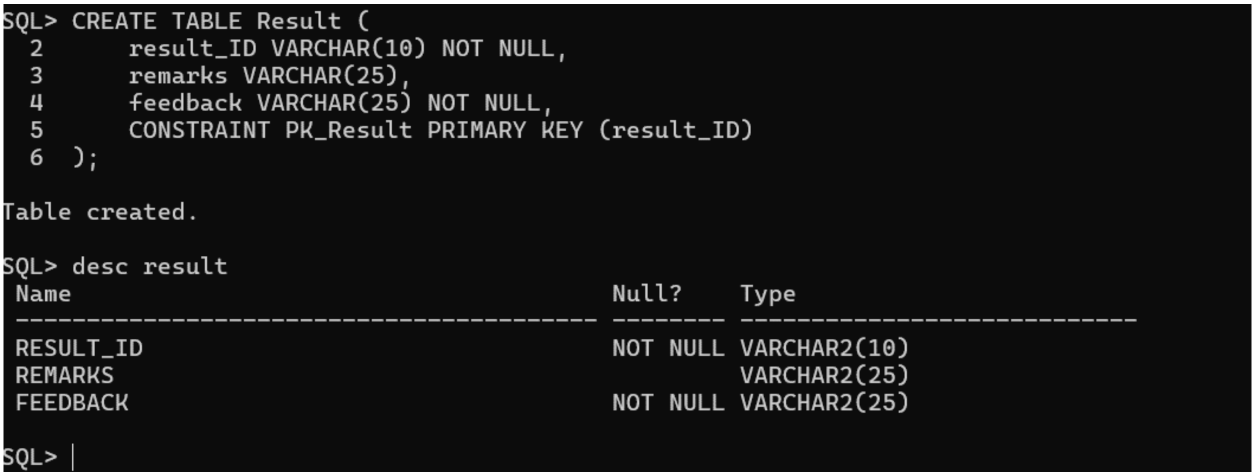
- **Creating assessment tables**



```
SQL> CREATE TABLE Assessment (
        assessment_ID VARCHAR(10) NOT NULL,
        Assessment_name VARCHAR(25) NOT NULL,
        Asmt_posted_date DATE NOT NULL,
        Asmt_end_date DATE NOT NULL,
        Weightage NUMBER NOT NULL,
        CONSTRAINT assessID PRIMARY KEY (assessment_ID)
    );

Table created.

SQL> desc assessment
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------
 ASSESSMENT_ID                             NOT NULL VARCHAR2(10)
 ASSESSMENT_NAME                           NOT NULL VARCHAR2(25)
 ASMT_POSTED_DATE                          NOT NULL DATE
 ASMT_END_DATE                             NOT NULL DATE
 WEIGHTAGE                                 NOT NULL NUMBER

SQL> |
```

Figure 15: Creating Assessment Table.

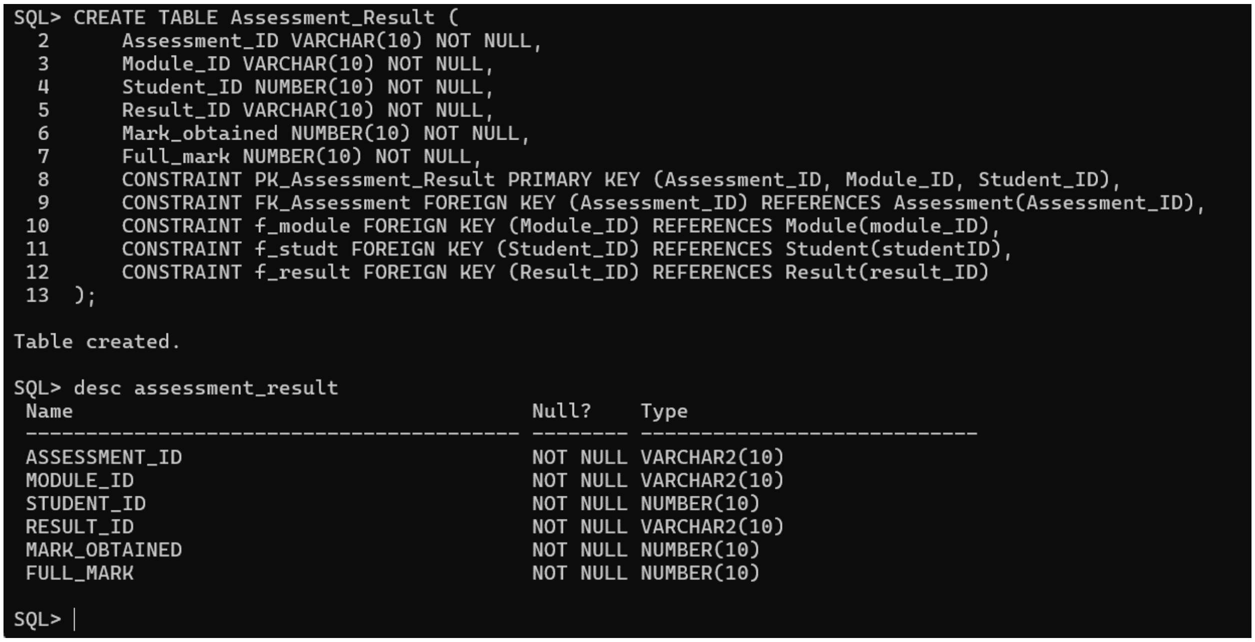
- **Creating result table**



```
SQL> CREATE TABLE Result (
         result_ID VARCHAR(10) NOT NULL,
         remarks VARCHAR(25),
         feedback VARCHAR(25) NOT NULL,
         CONSTRAINT PK_Result PRIMARY KEY (result_ID)
     );

Table created.

SQL> desc result
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RESULT_ID                                 NOT NULL VARCHAR2(10)
 REMARKS                                            VARCHAR2(25)
 FEEDBACK                                  NOT NULL VARCHAR2(25)

SQL> |
```

Figure 16: Creating Result Table.

- **Creating Assessment-Result Table**

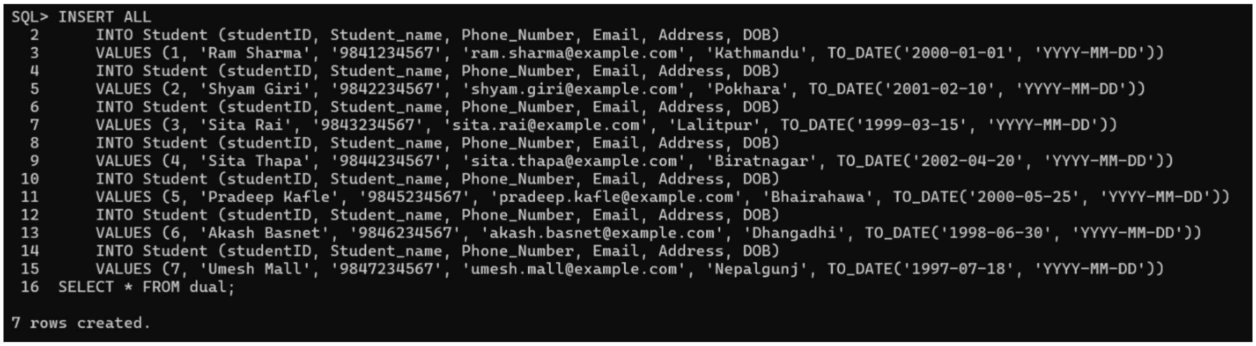


```
SQL> CREATE TABLE Assessment_Result (
         Assessment_ID VARCHAR(10) NOT NULL,
         Module_ID VARCHAR(10) NOT NULL,
         Student_ID NUMBER(10) NOT NULL,
         Result_ID VARCHAR(10) NOT NULL,
         Mark_obtained NUMBER(10) NOT NULL,
         Full_mark NUMBER(10) NOT NULL,
         CONSTRAINT PK_Assessment_Result PRIMARY KEY (Assessment_ID, Module_ID, Student_ID),
         CONSTRAINT FK_Assessment FOREIGN KEY (Assessment_ID) REFERENCES Assessment(Assessment_ID),
         CONSTRAINT f_module FOREIGN KEY (Module_ID) REFERENCES Module(module_ID),
         CONSTRAINT f_studt FOREIGN KEY (Student_ID) REFERENCES Student(studentID),
         CONSTRAINT f_result FOREIGN KEY (Result_ID) REFERENCES Result(result_ID)
     );

Table created.

SQL> desc assessment_result
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ASSESSMENT_ID                             NOT NULL VARCHAR2(10)
 MODULE_ID                                 NOT NULL VARCHAR2(10)
 STUDENT_ID                                NOT NULL NUMBER(10)
 RESULT_ID                                 NOT NULL VARCHAR2(10)
 MARK_OBTAINED                             NOT NULL NUMBER(10)
 FULL_MARK                                 NOT NULL NUMBER(10)

SQL> |
```

Figure 17:Creating Assessment_Result Table.

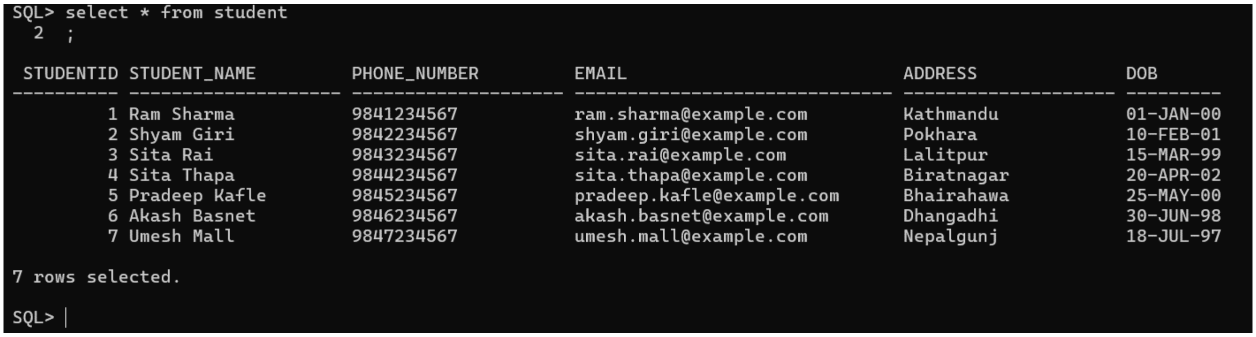
5.2 Inserting Data in the tables

- **Inserting data in student tables.**



```
SQL> INSERT ALL
        INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
        VALUES (1, 'Ram Sharma', '9841234567', 'ram.sharma@example.com', 'Kathmandu', TO_DATE('2000-01-01', 'YYYY-MM-DD'))
        INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
        VALUES (2, 'Shyam Giri', '9842234567', 'shyam.giri@example.com', 'Pokhara', TO_DATE('2001-02-10', 'YYYY-MM-DD'))
        INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
        VALUES (3, 'Sita Rai', '9843234567', 'sita.rai@example.com', 'Lalitpur', TO_DATE('1999-03-15', 'YYYY-MM-DD'))
        INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
        VALUES (4, 'Sita Thapa', '9844234567', 'sita.thapa@example.com', 'Biratnagar', TO_DATE('2002-04-20', 'YYYY-MM-DD'))
        INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
        VALUES (5, 'Pradeep Kafle', '9845234567', 'pradeep.kafle@example.com', 'Bhairahawa', TO_DATE('2000-05-25', 'YYYY-MM-DD'))
        INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
        VALUES (6, 'Akash Basnet', '9846234567', 'akash.basnet@example.com', 'Dhangadhi', TO_DATE('1998-06-30', 'YYYY-MM-DD'))
        INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
        VALUES (7, 'Umesh Mall', '9847234567', 'umesh.mall@example.com', 'Nepalgunj', TO_DATE('1997-07-18', 'YYYY-MM-DD'))
    SELECT * FROM dual;

7 rows created.
```

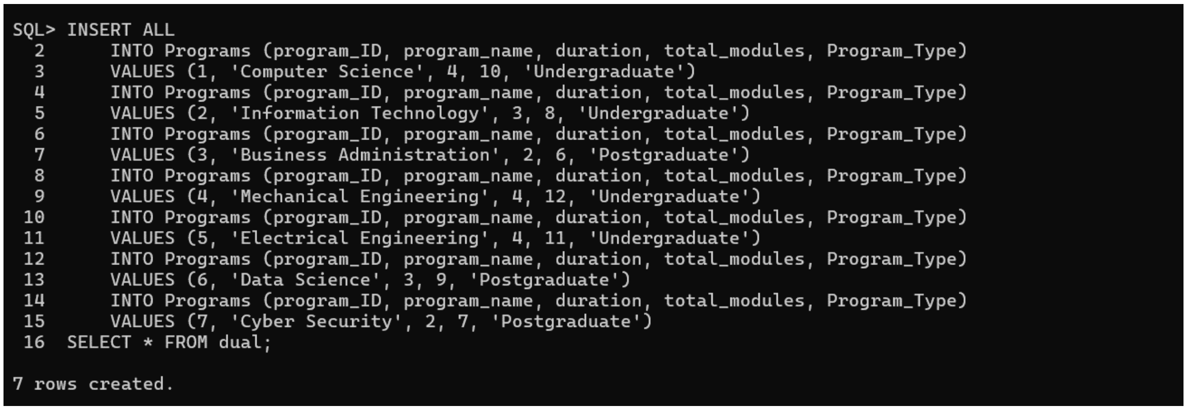
Figure 19: Inserting Data in Student Table.



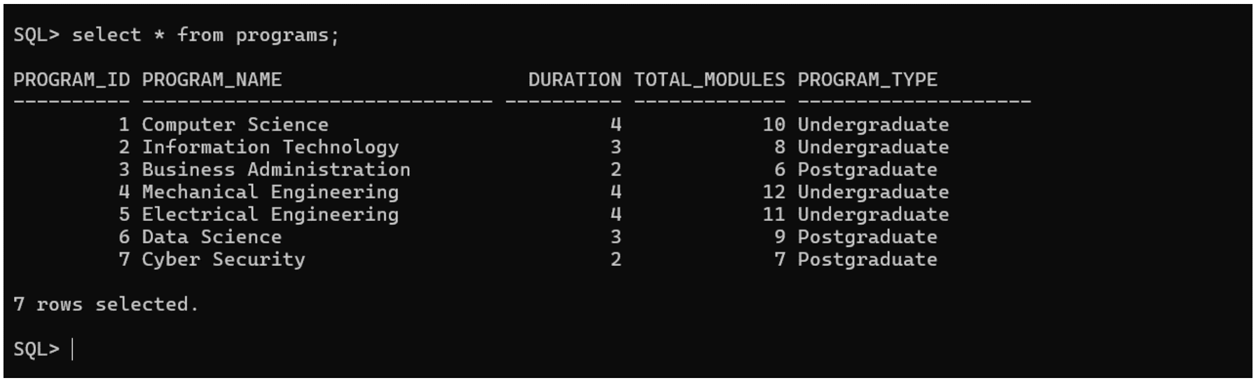
```
SQL> select * from student
     ;

 STUDENTID STUDENT_NAME         PHONE_NUMBER         EMAIL                          ADDRESS              DOB
---------- -------------------- -------------------- ------------------------------ -------------------- ---------
         1 Ram Sharma           9841234567           ram.sharma@example.com         Kathmandu            01-JAN-00
         2 Shyam Giri           9842234567           shyam.giri@example.com         Pokhara              10-FEB-01
         3 Sita Rai             9843234567           sita.rai@example.com           Lalitpur             15-MAR-99
         4 Sita Thapa           9844234567           sita.thapa@example.com         Biratnagar           20-APR-02
         5 Pradeep Kafle        9845234567           pradeep.kafle@example.com      Bhairahawa           25-MAY-00
         6 Akash Basnet         9846234567           akash.basnet@example.com       Dhangadhi            30-JUN-98
         7 Umesh Mall           9847234567           umesh.mall@example.com         Nepalgunj            18-JUL-97

7 rows selected.

SQL> |
```

Figure 18: Showing the inserted data from student Table.

- **Inserting data in program tables.**



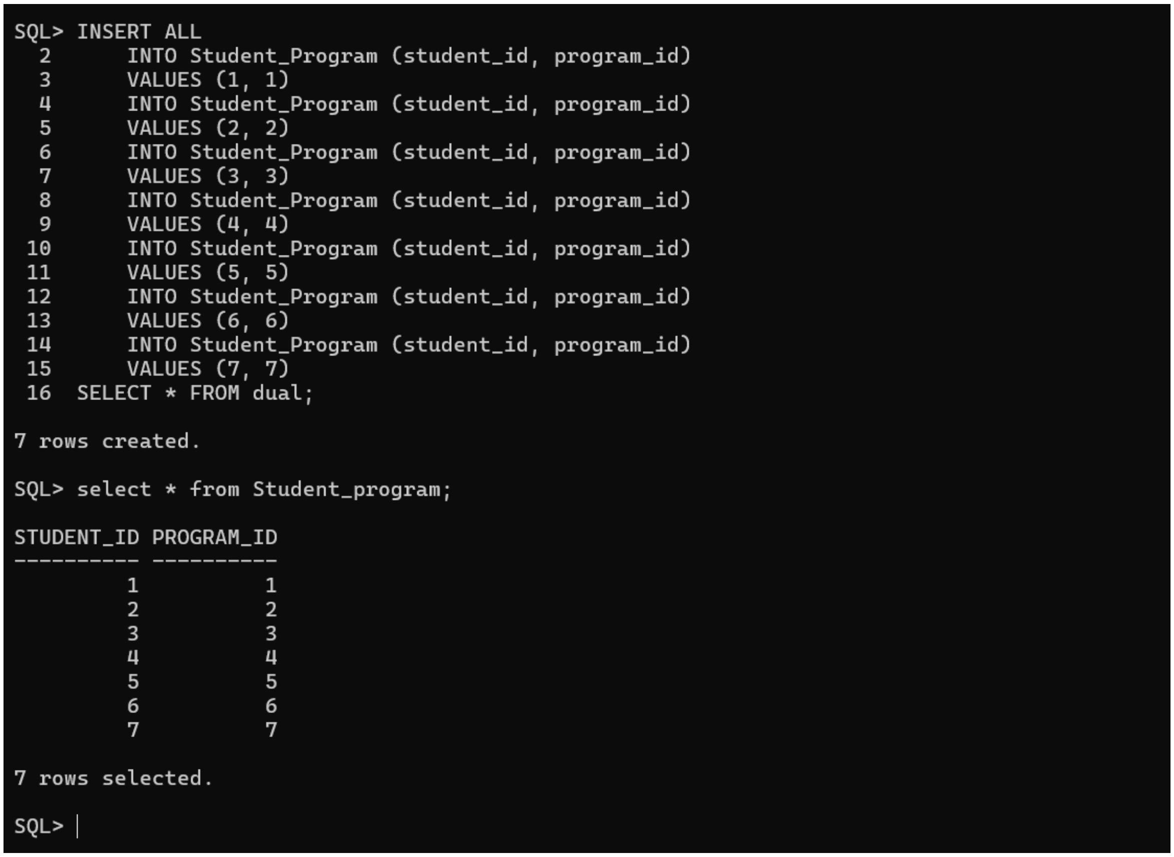
```
SQL> INSERT ALL
        INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
        VALUES (1, 'Computer Science', 4, 10, 'Undergraduate')
        INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
        VALUES (2, 'Information Technology', 3, 8, 'Undergraduate')
        INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
        VALUES (3, 'Business Administration', 2, 6, 'Postgraduate')
        INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
        VALUES (4, 'Mechanical Engineering', 4, 12, 'Undergraduate')
        INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
        VALUES (5, 'Electrical Engineering', 4, 11, 'Undergraduate')
        INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
        VALUES (6, 'Data Science', 3, 9, 'Postgraduate')
        INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
        VALUES (7, 'Cyber Security', 2, 7, 'Postgraduate')
    SELECT * FROM dual;

7 rows created.
```



```
SQL> select * from programs;

PROGRAM_ID PROGRAM_NAME                     DURATION TOTAL_MODULES PROGRAM_TYPE
---------- ------------------------------ ---------- ------------- --------------------
         1 Computer Science                        4            10 Undergraduate
         2 Information Technology                  3             8 Undergraduate
         3 Business Administration                 2             6 Postgraduate
         4 Mechanical Engineering                  4            12 Undergraduate
         5 Electrical Engineering                  4            11 Undergraduate
         6 Data Science                            3             9 Postgraduate
         7 Cyber Security                          2             7 Postgraduate

7 rows selected.

SQL> |
```

Figure 20: Inserting Data in program table.

- **Inserting data in student-program table.**

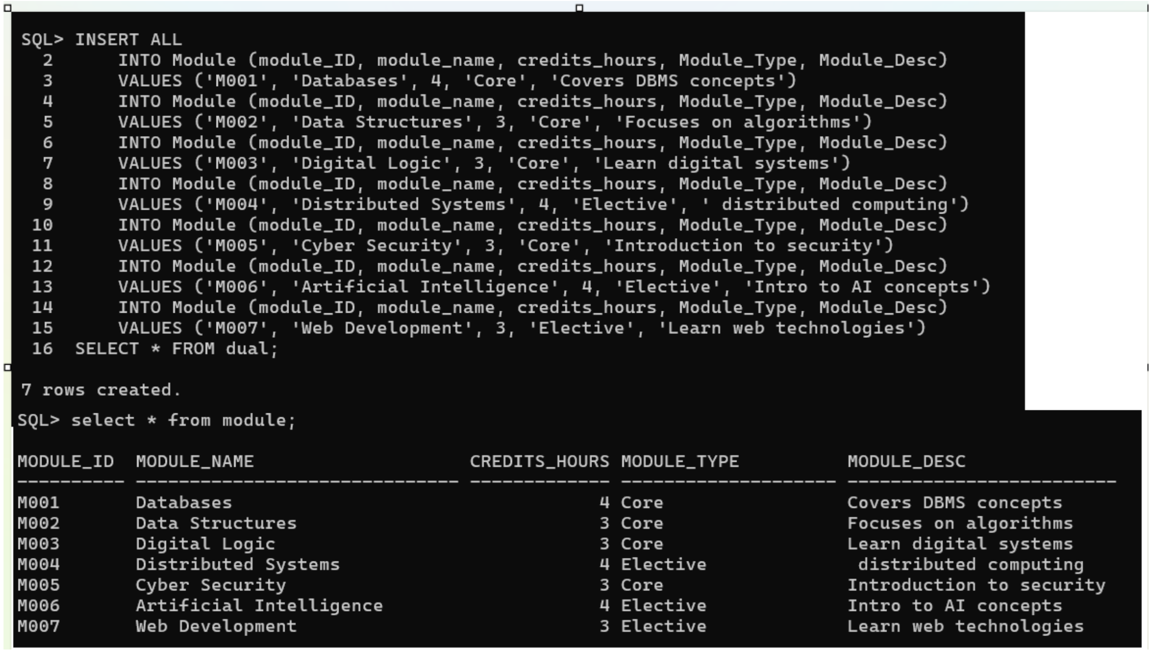


```
SQL> INSERT ALL
         INTO Student_Program (student_id, program_id)
         VALUES (1, 1)
         INTO Student_Program (student_id, program_id)
         VALUES (2, 2)
         INTO Student_Program (student_id, program_id)
         VALUES (3, 3)
         INTO Student_Program (student_id, program_id)
         VALUES (4, 4)
         INTO Student_Program (student_id, program_id)
         VALUES (5, 5)
         INTO Student_Program (student_id, program_id)
         VALUES (6, 6)
         INTO Student_Program (student_id, program_id)
         VALUES (7, 7)
     SELECT * FROM dual;

7 rows created.

SQL> select * from Student_program;

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

7 rows selected.

SQL> |
```

Figure 21: Inserting data in Student_Program Table.

\

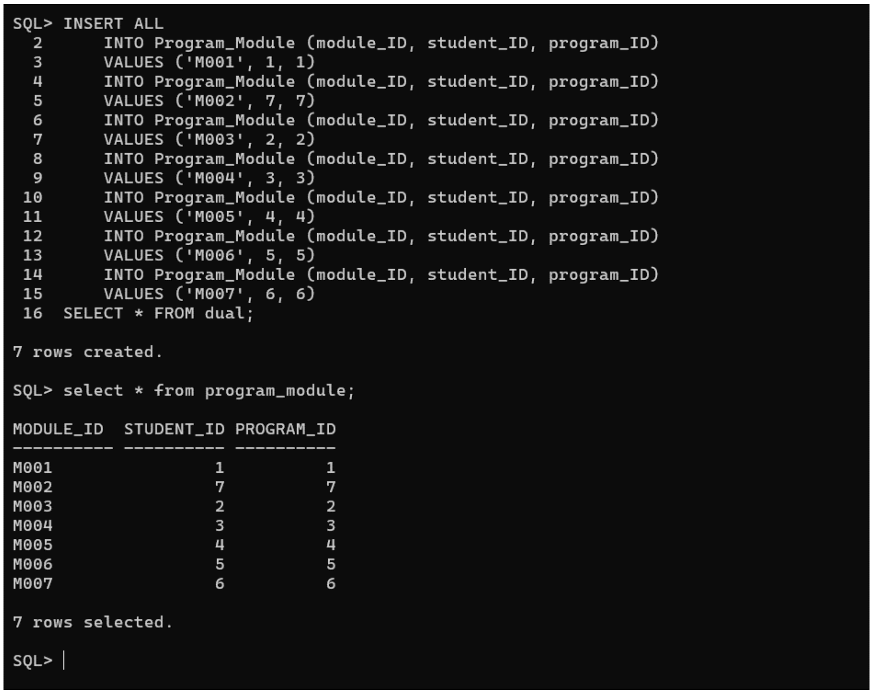
- **Inserting data in Module.**



```
SQL> INSERT ALL
        INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
        VALUES ('M001', 'Databases', 4, 'Core', 'Covers DBMS concepts')
        INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
        VALUES ('M002', 'Data Structures', 3, 'Core', 'Focuses on algorithms')
        INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
        VALUES ('M003', 'Digital Logic', 3, 'Core', 'Learn digital systems')
        INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
        VALUES ('M004', 'Distributed Systems', 4, 'Elective', ' distributed computing')
        INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
        VALUES ('M005', 'Cyber Security', 3, 'Core', 'Introduction to security')
        INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
        VALUES ('M006', 'Artificial Intelligence', 4, 'Elective', 'Intro to AI concepts')
        INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
        VALUES ('M007', 'Web Development', 3, 'Elective', 'Learn web technologies')
    SELECT * FROM dual;

7 rows created.

SQL> select * from module;

MODULE_ID  MODULE_NAME                    CREDITS_HOURS MODULE_TYPE          MODULE_DESC
---------- ------------------------------ ------------- -------------------- -------------------------
M001       Databases                                  4 Core                 Covers DBMS concepts
M002       Data Structures                            3 Core                 Focuses on algorithms
M003       Digital Logic                              3 Core                 Learn digital systems
M004       Distributed Systems                        4 Elective              distributed computing
M005       Cyber Security                             3 Core                 Introduction to security
M006       Artificial Intelligence                    4 Elective             Intro to AI concepts
M007       Web Development                            3 Elective             Learn web technologies
```

Figure 22: Inserting data in Module Table.

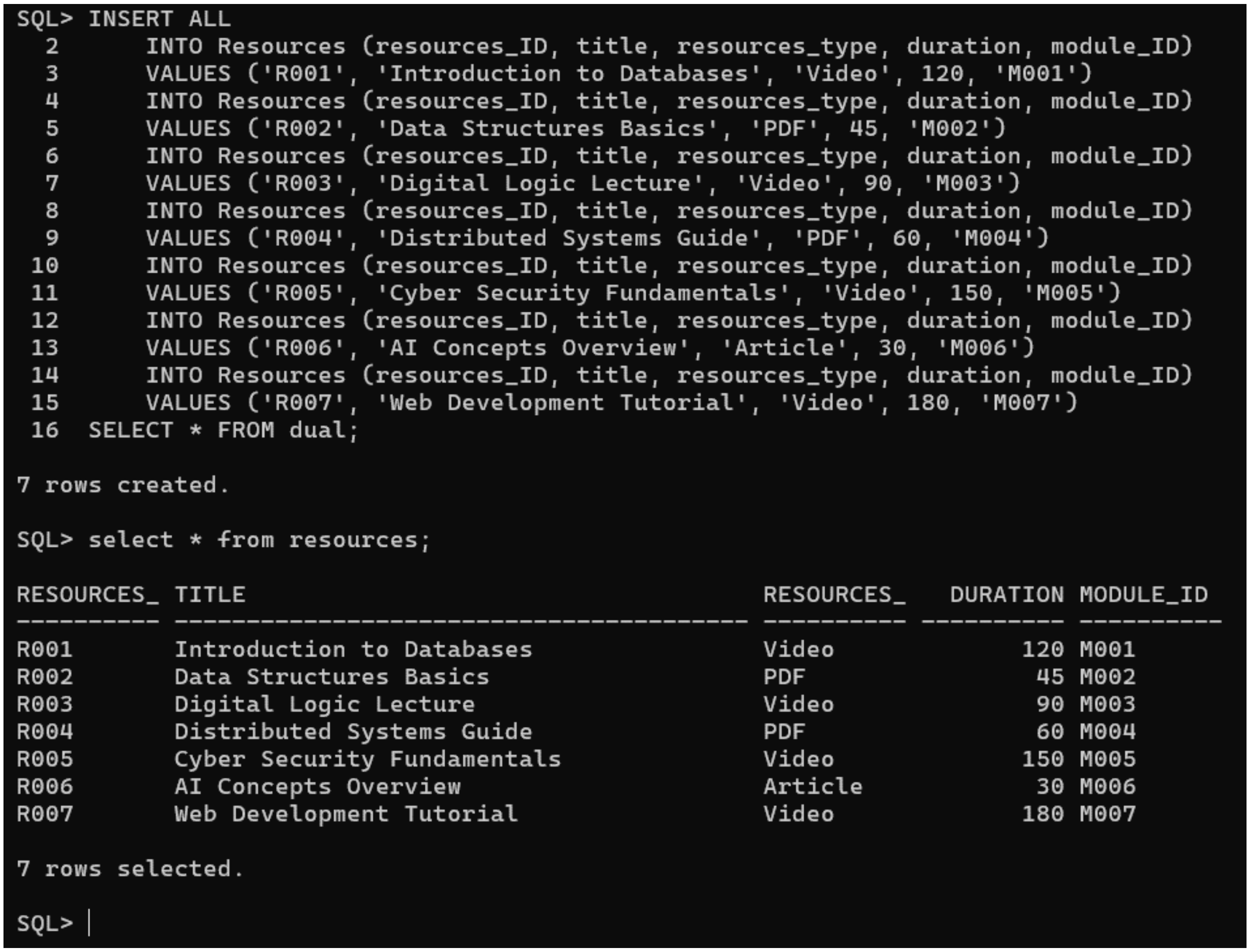
- **Inserting data in program-module tables.**



```
SQL> INSERT ALL
        INTO Program_Module (module_ID, student_ID, program_ID)
        VALUES ('M001', 1, 1)
        INTO Program_Module (module_ID, student_ID, program_ID)
        VALUES ('M002', 7, 7)
        INTO Program_Module (module_ID, student_ID, program_ID)
        VALUES ('M003', 2, 2)
        INTO Program_Module (module_ID, student_ID, program_ID)
        VALUES ('M004', 3, 3)
        INTO Program_Module (module_ID, student_ID, program_ID)
        VALUES ('M005', 4, 4)
        INTO Program_Module (module_ID, student_ID, program_ID)
        VALUES ('M006', 5, 5)
        INTO Program_Module (module_ID, student_ID, program_ID)
        VALUES ('M007', 6, 6)
    SELECT * FROM dual;

7 rows created.

SQL> select * from program_module;

MODULE_ID  STUDENT_ID PROGRAM_ID
---------- ---------- ----------
M001                1          1
M002                7          7
M003                2          2
M004                3          3
M005                4          4
M006                5          5
M007                6          6

7 rows selected.

SQL> |
```

Figure 23: Inserting Data in Program_Module.

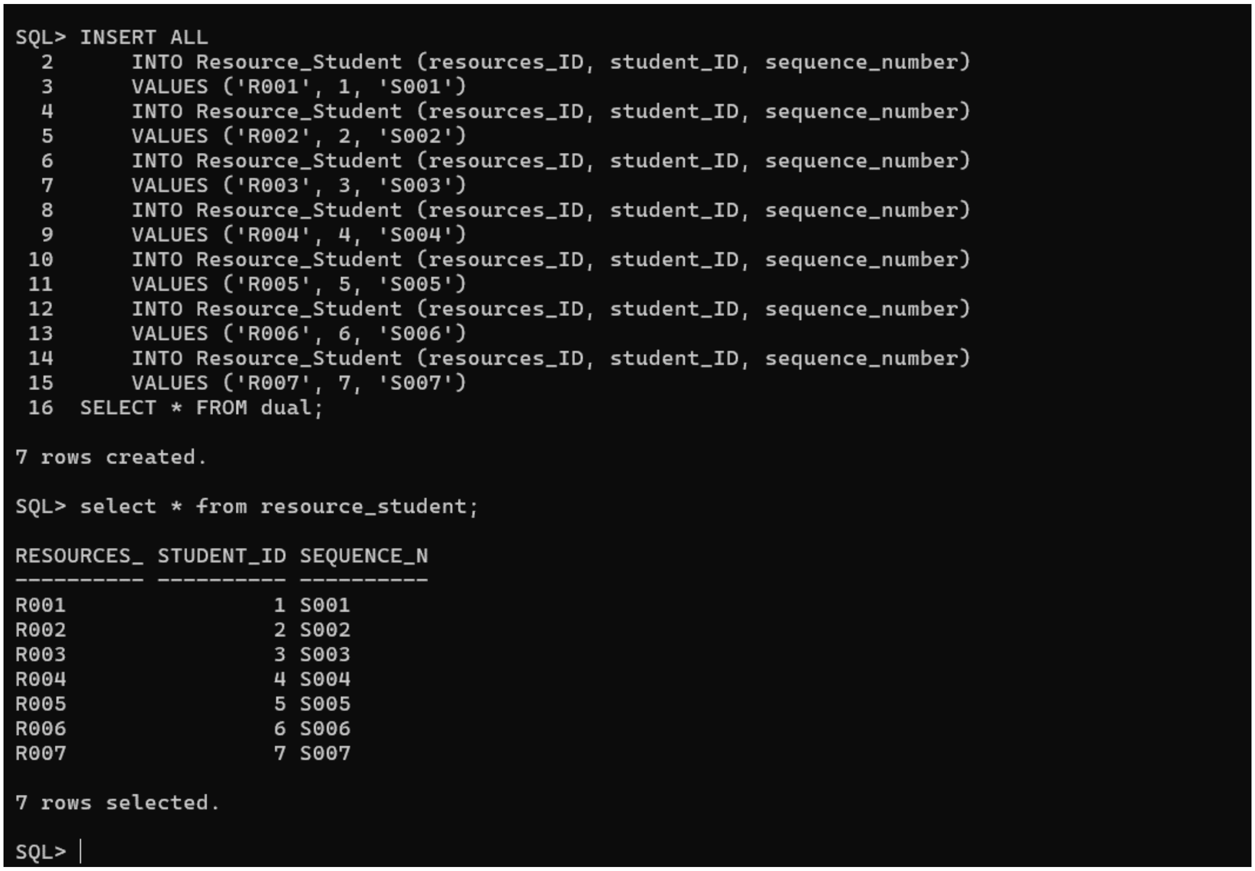
- **Inserting data in Resources table.**



```
SQL> INSERT ALL
         INTO Resources (resources_ID, title, resources_type, duration, module_ID)
         VALUES ('R001', 'Introduction to Databases', 'Video', 120, 'M001')
         INTO Resources (resources_ID, title, resources_type, duration, module_ID)
         VALUES ('R002', 'Data Structures Basics', 'PDF', 45, 'M002')
         INTO Resources (resources_ID, title, resources_type, duration, module_ID)
         VALUES ('R003', 'Digital Logic Lecture', 'Video', 90, 'M003')
         INTO Resources (resources_ID, title, resources_type, duration, module_ID)
         VALUES ('R004', 'Distributed Systems Guide', 'PDF', 60, 'M004')
         INTO Resources (resources_ID, title, resources_type, duration, module_ID)
         VALUES ('R005', 'Cyber Security Fundamentals', 'Video', 150, 'M005')
         INTO Resources (resources_ID, title, resources_type, duration, module_ID)
         VALUES ('R006', 'AI Concepts Overview', 'Article', 30, 'M006')
         INTO Resources (resources_ID, title, resources_type, duration, module_ID)
         VALUES ('R007', 'Web Development Tutorial', 'Video', 180, 'M007')
     SELECT * FROM dual;

7 rows created.

SQL> select * from resources;

RESOURCES_ TITLE                                    RESOURCES_   DURATION MODULE_ID
---------- ---------------------------------------- ---------- ---------- ----------
R001       Introduction to Databases                Video             120 M001
R002       Data Structures Basics                   PDF                45 M002
R003       Digital Logic Lecture                    Video              90 M003
R004       Distributed Systems Guide                PDF                60 M004
R005       Cyber Security Fundamentals              Video             150 M005
R006       AI Concepts Overview                     Article            30 M006
R007       Web Development Tutorial                 Video             180 M007

7 rows selected.

SQL> |
```

Figure 24: Inserting Data in Resources Table.

- **Inserting data in Resources-Student table.**

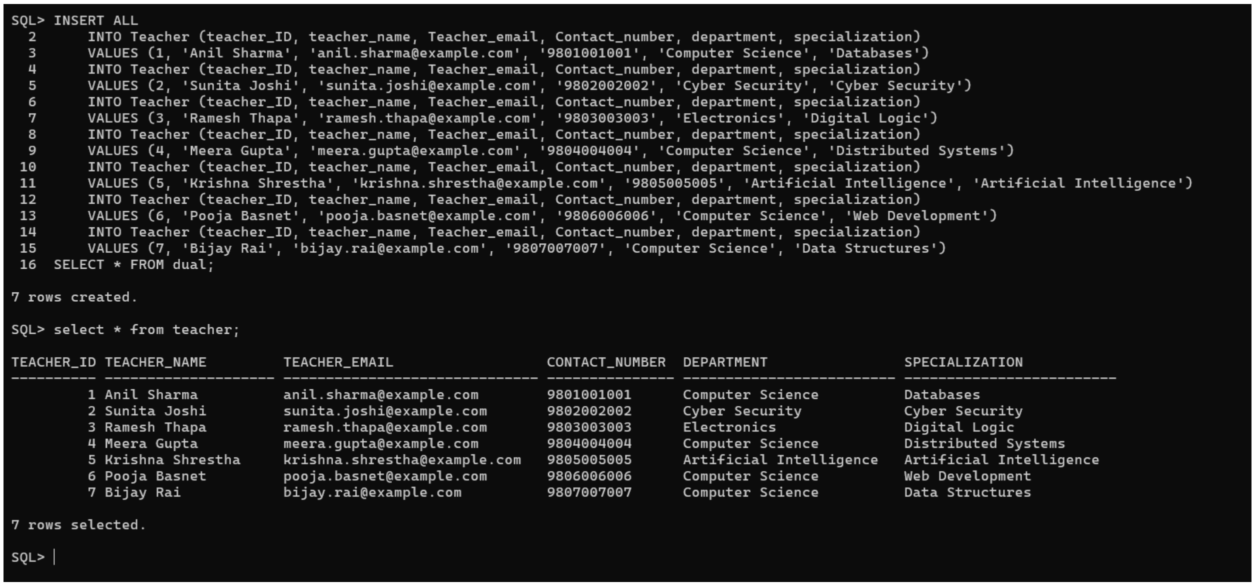


```
SQL> INSERT ALL
         INTO Resource_Student (resources_ID, student_ID, sequence_number)
         VALUES ('R001', 1, 'S001')
         INTO Resource_Student (resources_ID, student_ID, sequence_number)
         VALUES ('R002', 2, 'S002')
         INTO Resource_Student (resources_ID, student_ID, sequence_number)
         VALUES ('R003', 3, 'S003')
         INTO Resource_Student (resources_ID, student_ID, sequence_number)
         VALUES ('R004', 4, 'S004')
         INTO Resource_Student (resources_ID, student_ID, sequence_number)
         VALUES ('R005', 5, 'S005')
         INTO Resource_Student (resources_ID, student_ID, sequence_number)
         VALUES ('R006', 6, 'S006')
         INTO Resource_Student (resources_ID, student_ID, sequence_number)
         VALUES ('R007', 7, 'S007')
     SELECT * FROM dual;

7 rows created.

SQL> select * from resource_student;

RESOURCES_ STUDENT_ID SEQUENCE_N
---------- ---------- ----------
R001                1 S001
R002                2 S002
R003                3 S003
R004                4 S004
R005                5 S005
R006                6 S006
R007                7 S007

7 rows selected.

SQL> |
```

Figure 25: Inserting Data in Resource_Student

.

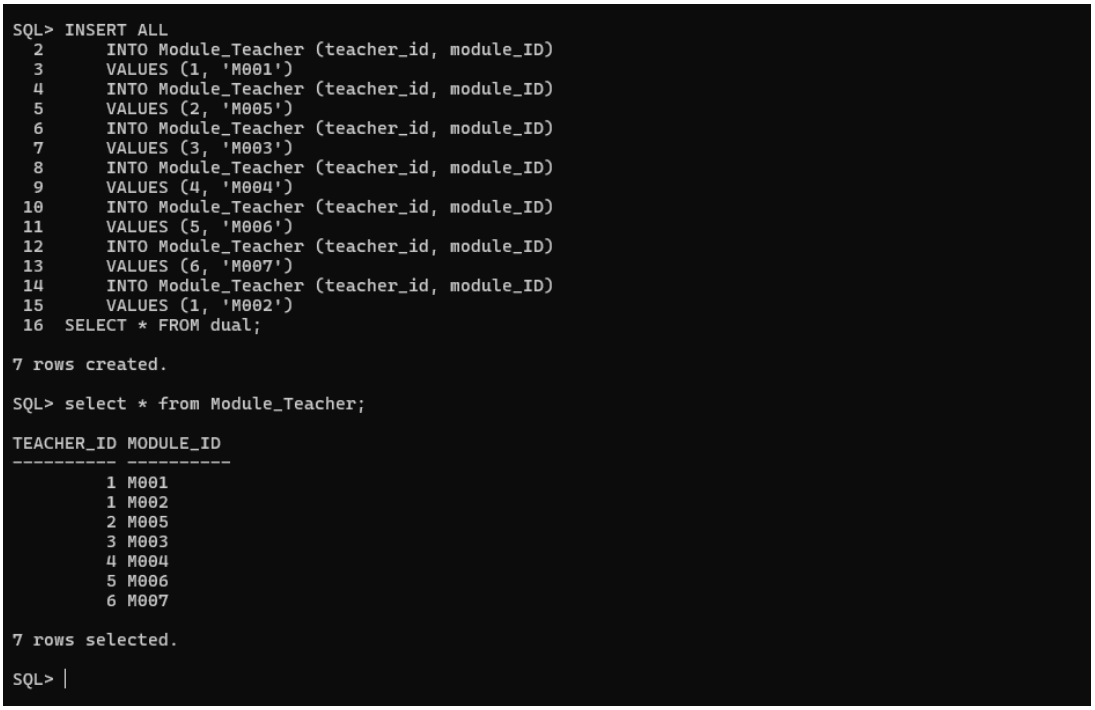
- **Inserting data in Teacher Tables.**



```
SQL> INSERT ALL
  2      INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
  3      VALUES (1, 'Anil Sharma', 'anil.sharma@example.com', '9801001001', 'Computer Science', 'Databases')
  4      INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
  5      VALUES (2, 'Sunita Joshi', 'sunita.joshi@example.com', '9802002002', 'Cyber Security', 'Cyber Security')
  6      INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
  7      VALUES (3, 'Ramesh Thapa', 'ramesh.thapa@example.com', '9803003003', 'Electronics', 'Digital Logic')
  8      INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
  9      VALUES (4, 'Meera Gupta', 'meera.gupta@example.com', '9804004004', 'Computer Science', 'Distributed Systems')
 10      INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
 11      VALUES (5, 'Krishna Shrestha', 'krishna.shrestha@example.com', '9805005005', 'Artificial Intelligence', 'Artificial Intelligence')
 12      INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
 13      VALUES (6, 'Pooja Basnet', 'pooja.basnet@example.com', '9806006006', 'Computer Science', 'Web Development')
 14      INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
 15      VALUES (7, 'Bijay Rai', 'bijay.rai@example.com', '9807007007', 'Computer Science', 'Data Structures')
 16  SELECT * FROM dual;

7 rows created.

SQL> select * from teacher;

TEACHER_ID TEACHER_NAME         TEACHER_EMAIL                  CONTACT_NUMBER  DEPARTMENT                SPECIALIZATION
---------- -------------------- ------------------------------ --------------- ------------------------- -------------------------
         1 Anil Sharma          anil.sharma@example.com        9801001001      Computer Science          Databases
         2 Sunita Joshi         sunita.joshi@example.com       9802002002      Cyber Security            Cyber Security
         3 Ramesh Thapa         ramesh.thapa@example.com       9803003003      Electronics               Digital Logic
         4 Meera Gupta          meera.gupta@example.com        9804004004      Computer Science          Distributed Systems
         5 Krishna Shrestha     krishna.shrestha@example.com   9805005005      Artificial Intelligence   Artificial Intelligence
         6 Pooja Basnet         pooja.basnet@example.com       9806006006      Computer Science          Web Development
         7 Bijay Rai            bijay.rai@example.com          9807007007      Computer Science          Data Structures

7 rows selected.

SQL> |
```

Figure 26: Inserting data in Teacher Table.

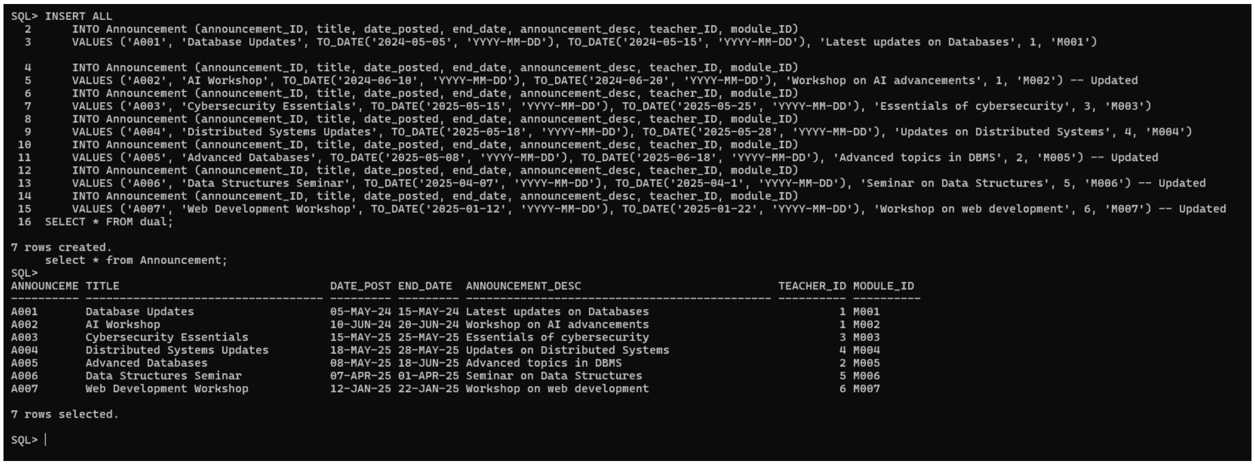
- **Inserting data in Module-Teacher Tables.**



```
SQL> INSERT ALL
  2      INTO Module_Teacher (teacher_id, module_ID)
  3      VALUES (1, 'M001')
  4      INTO Module_Teacher (teacher_id, module_ID)
  5      VALUES (2, 'M005')
  6      INTO Module_Teacher (teacher_id, module_ID)
  7      VALUES (3, 'M003')
  8      INTO Module_Teacher (teacher_id, module_ID)
  9      VALUES (4, 'M004')
 10      INTO Module_Teacher (teacher_id, module_ID)
 11      VALUES (5, 'M006')
 12      INTO Module_Teacher (teacher_id, module_ID)
 13      VALUES (6, 'M007')
 14      INTO Module_Teacher (teacher_id, module_ID)
 15      VALUES (1, 'M002')
 16  SELECT * FROM dual;

7 rows created.

SQL> select * from Module_Teacher;

TEACHER_ID MODULE_ID
---------- ----------
         1 M001
         1 M002
         2 M005
         3 M003
         4 M004
         5 M006
         6 M007

7 rows selected.

SQL> |
```

Figure 27: Inserting Data in Module_Teacher Table.

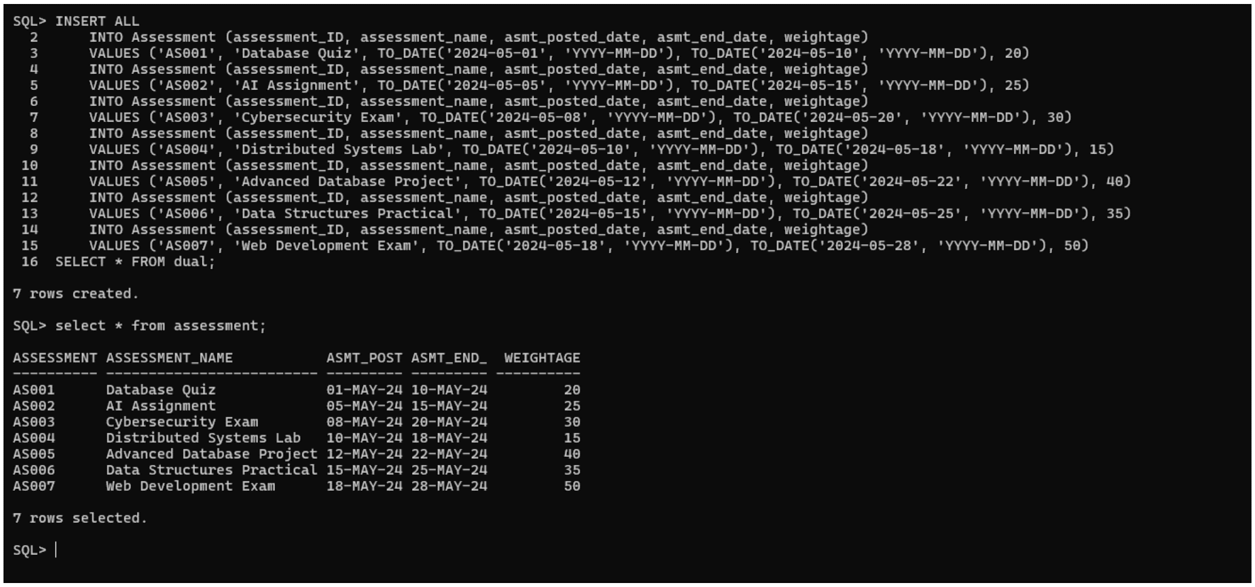
- **Inserting data in Announcement Tables.**



```
SQL> INSERT ALL
         INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
         VALUES ('A001', 'Database Updates', TO_DATE('2024-05-05', 'YYYY-MM-DD'), TO_DATE('2024-05-15', 'YYYY-MM-DD'), 'Latest updates on Databases', 1, 'M001')

         INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
         VALUES ('A002', 'AI Workshop', TO_DATE('2024-06-10', 'YYYY-MM-DD'), TO_DATE('2024-06-20', 'YYYY-MM-DD'), 'Workshop on AI advancements', 1, 'M002') -- Updated
         INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
         VALUES ('A003', 'Cybersecurity Essentials', TO_DATE('2025-05-15', 'YYYY-MM-DD'), TO_DATE('2025-05-25', 'YYYY-MM-DD'), 'Essentials of cybersecurity', 3, 'M003')
         INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
         VALUES ('A004', 'Distributed Systems Updates', TO_DATE('2025-05-18', 'YYYY-MM-DD'), TO_DATE('2025-05-28', 'YYYY-MM-DD'), 'Updates on Distributed Systems', 4, 'M004')
         INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
         VALUES ('A005', 'Advanced Databases', TO_DATE('2025-05-08', 'YYYY-MM-DD'), TO_DATE('2025-06-18', 'YYYY-MM-DD'), 'Advanced topics in DBMS', 2, 'M005') -- Updated
         INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
         VALUES ('A006', 'Data Structures Seminar', TO_DATE('2025-04-07', 'YYYY-MM-DD'), TO_DATE('2025-04-1', 'YYYY-MM-DD'), 'Seminar on Data Structures', 5, 'M006') -- Updated
         INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
         VALUES ('A007', 'Web Development Workshop', TO_DATE('2025-01-12', 'YYYY-MM-DD'), TO_DATE('2025-01-22', 'YYYY-MM-DD'), 'Workshop on web development', 6, 'M007') -- Updated
     SELECT * FROM dual;

7 rows created.
     select * from Announcement;
SQL>
ANNOUNCEME TITLE                              DATE_POST END_DATE  ANNOUNCEMENT_DESC                                TEACHER_ID MODULE_ID
---------- ---------------------------------- --------- --------- ------------------------------------------------ ---------- ----------
A001       Database Updates                   05-MAY-24 15-MAY-24 Latest updates on Databases                               1 M001
A002       AI Workshop                        10-JUN-24 20-JUN-24 Workshop on AI advancements                               1 M002
A003       Cybersecurity Essentials           15-MAY-25 25-MAY-25 Essentials of cybersecurity                               3 M003
A004       Distributed Systems Updates        18-MAY-25 28-MAY-25 Updates on Distributed Systems                            4 M004
A005       Advanced Databases                 08-MAY-25 18-JUN-25 Advanced topics in DBMS                                   2 M005
A006       Data Structures Seminar            07-APR-25 01-APR-25 Seminar on Data Structures                                5 M006
A007       Web Development Workshop           12-JAN-25 22-JAN-25 Workshop on web development                               6 M007

7 rows selected.

SQL> |
```

Figure 28: Inserting Data in Announcement Table.

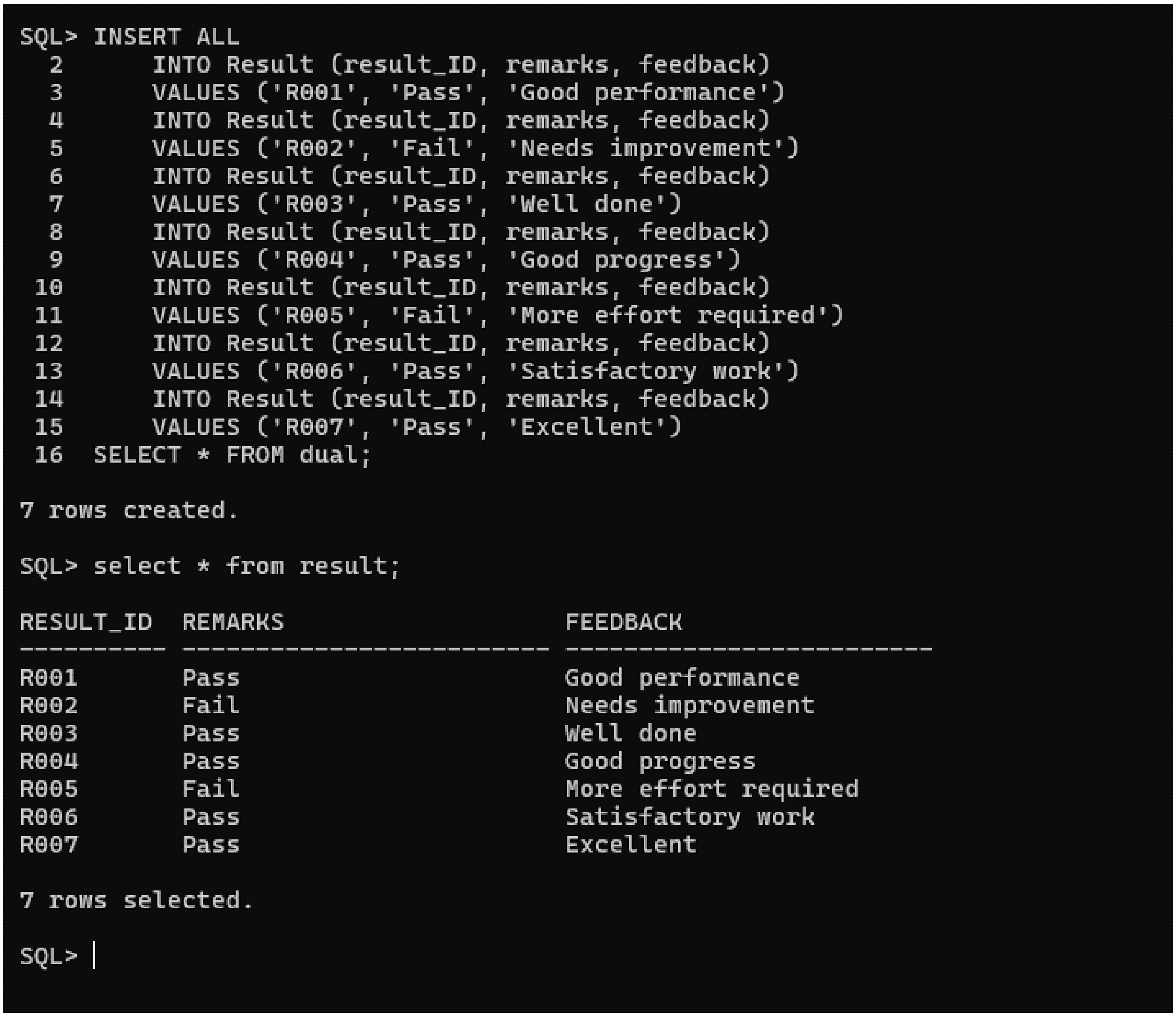
- **Inserting data in Assessment Tables.**



```
SQL> INSERT ALL
         INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
         VALUES ('AS001', 'Database Quiz', TO_DATE('2024-05-01', 'YYYY-MM-DD'), TO_DATE('2024-05-10', 'YYYY-MM-DD'), 20)
         INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
         VALUES ('AS002', 'AI Assignment', TO_DATE('2024-05-05', 'YYYY-MM-DD'), TO_DATE('2024-05-15', 'YYYY-MM-DD'), 25)
         INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
         VALUES ('AS003', 'Cybersecurity Exam', TO_DATE('2024-05-08', 'YYYY-MM-DD'), TO_DATE('2024-05-20', 'YYYY-MM-DD'), 30)
         INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
         VALUES ('AS004', 'Distributed Systems Lab', TO_DATE('2024-05-10', 'YYYY-MM-DD'), TO_DATE('2024-05-18', 'YYYY-MM-DD'), 15)
         INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
         VALUES ('AS005', 'Advanced Database Project', TO_DATE('2024-05-12', 'YYYY-MM-DD'), TO_DATE('2024-05-22', 'YYYY-MM-DD'), 40)
         INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
         VALUES ('AS006', 'Data Structures Practical', TO_DATE('2024-05-15', 'YYYY-MM-DD'), TO_DATE('2024-05-25', 'YYYY-MM-DD'), 35)
         INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
         VALUES ('AS007', 'Web Development Exam', TO_DATE('2024-05-18', 'YYYY-MM-DD'), TO_DATE('2024-05-28', 'YYYY-MM-DD'), 50)
     SELECT * FROM dual;

7 rows created.

SQL> select * from assessment;

ASSESSMENT ASSESSMENT_NAME           ASMT_POST ASMT_END_  WEIGHTAGE
---------- ------------------------- --------- --------- ----------
AS001      Database Quiz             01-MAY-24 10-MAY-24         20
AS002      AI Assignment             05-MAY-24 15-MAY-24         25
AS003      Cybersecurity Exam        08-MAY-24 20-MAY-24         30
AS004      Distributed Systems Lab   10-MAY-24 18-MAY-24         15
AS005      Advanced Database Project 12-MAY-24 22-MAY-24         40
AS006      Data Structures Practical 15-MAY-24 25-MAY-24         35
AS007      Web Development Exam      18-MAY-24 28-MAY-24         50

7 rows selected.

SQL> |
```

Figure 29: Inserting Data in Assessment Table.

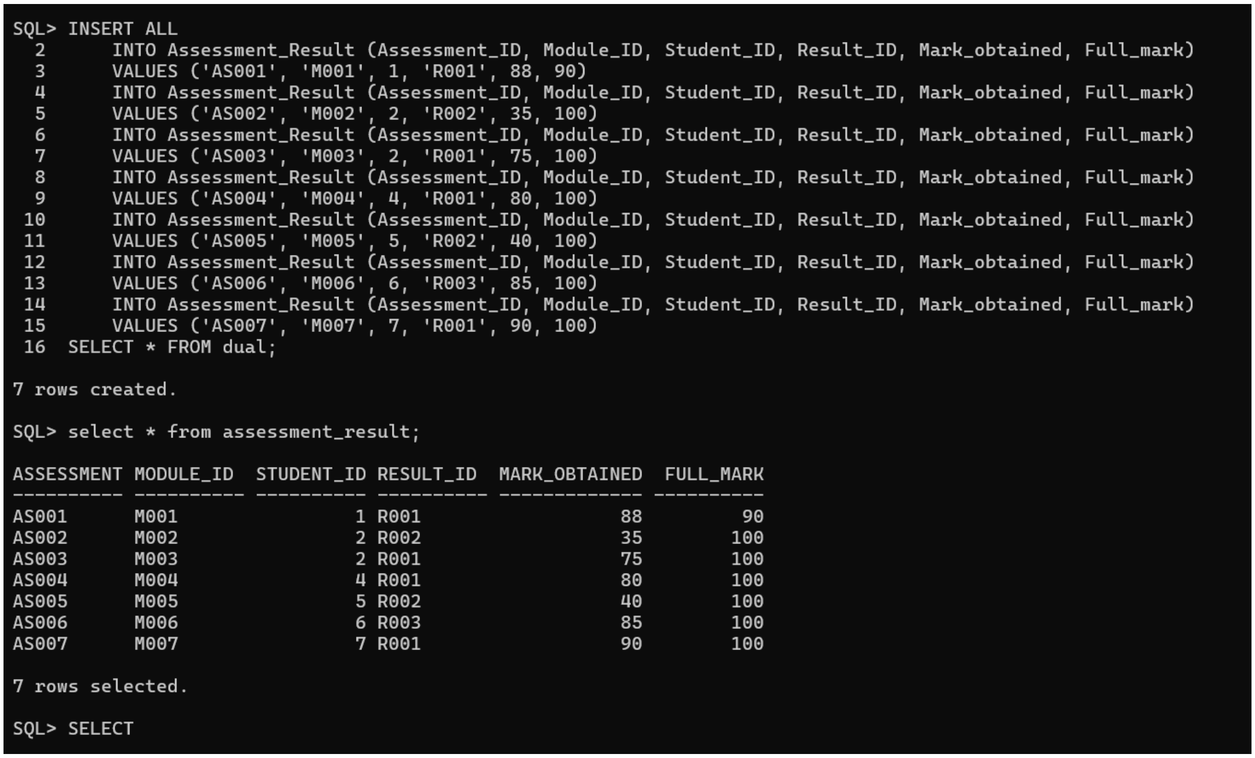
- **Inserting data in Result Tables.**



```
SQL> INSERT ALL
  2      INTO Result (result_ID, remarks, feedback)
  3      VALUES ('R001', 'Pass', 'Good performance')
  4      INTO Result (result_ID, remarks, feedback)
  5      VALUES ('R002', 'Fail', 'Needs improvement')
  6      INTO Result (result_ID, remarks, feedback)
  7      VALUES ('R003', 'Pass', 'Well done')
  8      INTO Result (result_ID, remarks, feedback)
  9      VALUES ('R004', 'Pass', 'Good progress')
 10      INTO Result (result_ID, remarks, feedback)
 11      VALUES ('R005', 'Fail', 'More effort required')
 12      INTO Result (result_ID, remarks, feedback)
 13      VALUES ('R006', 'Pass', 'Satisfactory work')
 14      INTO Result (result_ID, remarks, feedback)
 15      VALUES ('R007', 'Pass', 'Excellent')
 16  SELECT * FROM dual;

7 rows created.

SQL> select * from result;

RESULT_ID  REMARKS                    FEEDBACK
---------- -------------------------- -------------------------
R001       Pass                       Good performance
R002       Fail                       Needs improvement
R003       Pass                       Well done
R004       Pass                       Good progress
R005       Fail                       More effort required
R006       Pass                       Satisfactory work
R007       Pass                       Excellent

7 rows selected.

SQL> |
```

Figure 30: Inserting Data in Result Table.

- **Inserting Data in Assessment-Result Table**



```
SQL> INSERT ALL
         INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID, Mark_obtained, Full_mark)
         VALUES ('AS001', 'M001', 1, 'R001', 88, 90)
         INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID, Mark_obtained, Full_mark)
         VALUES ('AS002', 'M002', 2, 'R002', 35, 100)
         INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID, Mark_obtained, Full_mark)
         VALUES ('AS003', 'M003', 2, 'R001', 75, 100)
         INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID, Mark_obtained, Full_mark)
         VALUES ('AS004', 'M004', 4, 'R001', 80, 100)
         INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID, Mark_obtained, Full_mark)
         VALUES ('AS005', 'M005', 5, 'R002', 40, 100)
         INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID, Mark_obtained, Full_mark)
         VALUES ('AS006', 'M006', 6, 'R003', 85, 100)
         INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID, Mark_obtained, Full_mark)
         VALUES ('AS007', 'M007', 7, 'R001', 90, 100)
     SELECT * FROM dual;

7 rows created.

SQL> select * from assessment_result;

ASSESSMENT MODULE_ID  STUDENT_ID RESULT_ID  MARK_OBTAINED  FULL_MARK
---------- ---------- ---------- ---------- ------------- ----------
AS001      M001                1 R001                  88         90
AS002      M002                2 R002                  35        100
AS003      M003                2 R001                  75        100
AS004      M004                4 R001                  80        100
AS005      M005                5 R002                  40        100
AS006      M006                6 R003                  85        100
AS007      M007                7 R001                  90        100

7 rows selected.

SQL> SELECT
```

Figure 31: Inserting Data in Assesment_Table.

6 Database Querying

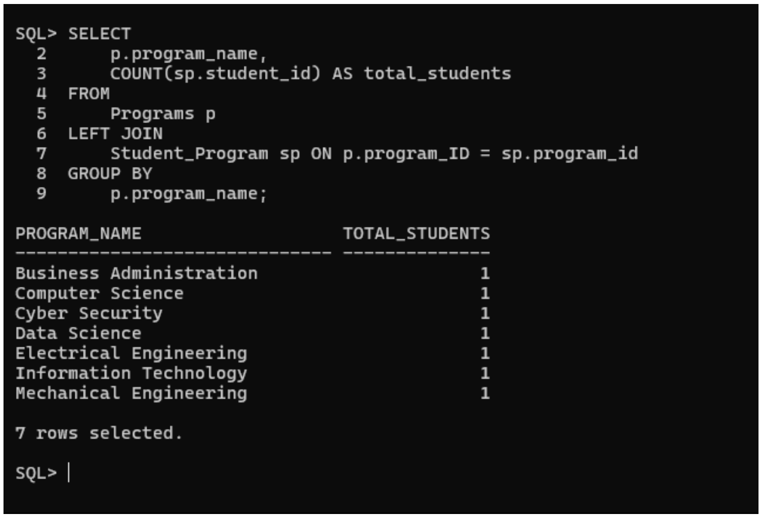
Database query are performed to get the specific data from database. It involves the use of Query language like SQL and is the most used and common language to interact with database. In this coursework we have to perform informational and transactional query.

6.1 Information query

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

```
SELECT
    p.program_name,
    COUNT(sp.student_id) AS total_students
FROM
    Programs p
LEFT JOIN
    Student_Program sp ON p.program_ID = sp.program_id
GROUP BY
    p.program_name;
```

This SQL query list each program name from program table along with the total number of the student enrolled on it. It uses left join with student_program to select all the program even those program where student are not enrolled and display their count. The count function in this query calculates the total number of student enrolled in respective program.



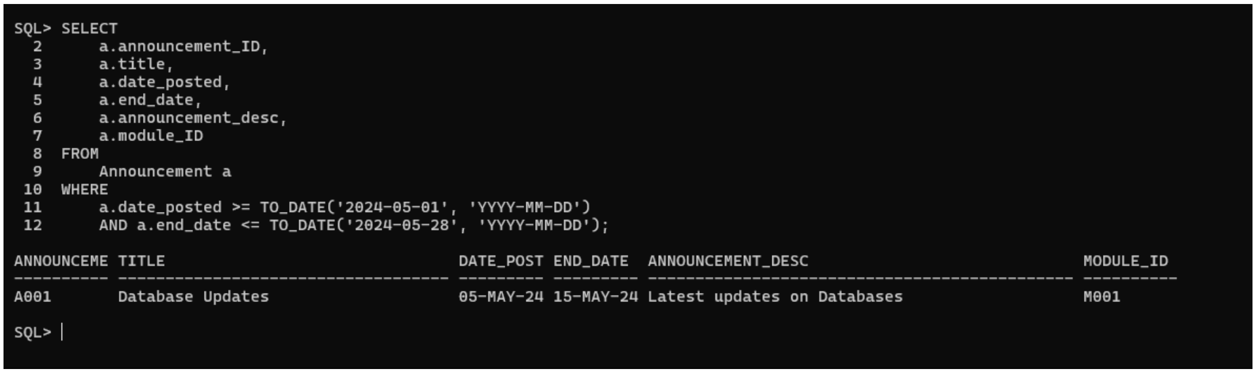
```
SQL> SELECT
         p.program_name,
         COUNT(sp.student_id) AS total_students
     FROM
         Programs p
     LEFT JOIN
         Student_Program sp ON p.program_ID = sp.program_id
     GROUP BY
         p.program_name;

PROGRAM_NAME                   TOTAL_STUDENTS
------------------------------ --------------
Business Administration                     1
Computer Science                            1
Cyber Security                              1
Data Science                                1
Electrical Engineering                      1
Information Technology                      1
Mechanical Engineering                      1

7 rows selected.

SQL> |
```

Figure 32: Information Query 1.

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

```
SELECT
    a.announcement_ID,
    a.title,
    a.date_posted,
    a.end_date,
    a.announcement_desc,
    a.module_ID
FROM
    Announcement a
WHERE
    a.date_posted >= TO_DATE('2024-05-01', 'YYYY-MM-DD')
    AND a.end_date <= TO_DATE('2024-05-28', 'YYYY-MM-DD');
```

The above quey retrieves all the details of announcement that were posted between May 1 2024 and May 28 2024. It selects columns like announcement_id, title, date_posted, end_date, announcement_desc and module_id. The where clause in this query checks if the announcement meets the requirement.



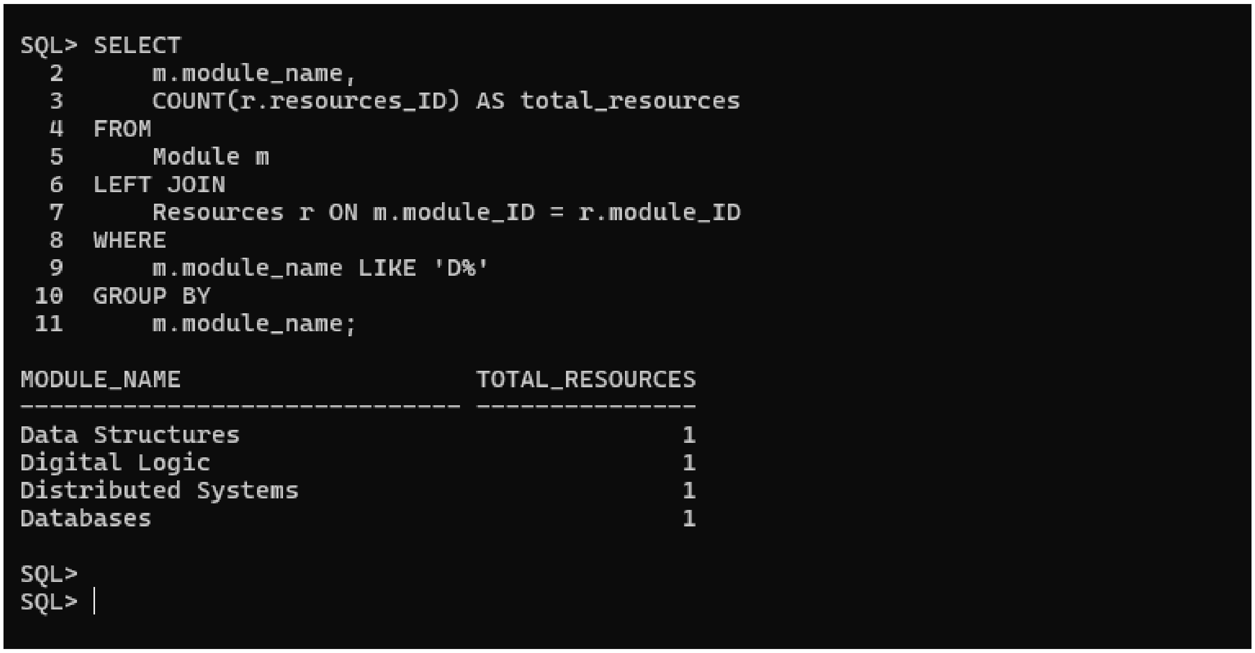
```
SQL> SELECT
         a.announcement_ID,
         a.title,
         a.date_posted,
         a.end_date,
         a.announcement_desc,
         a.module_ID
     FROM
         Announcement a
     WHERE
         a.date_posted >= TO_DATE('2024-05-01', 'YYYY-MM-DD')
         AND a.end_date <= TO_DATE('2024-05-28', 'YYYY-MM-DD');

ANNOUNCEME TITLE                               DATE_POST END_DATE  ANNOUNCEMENT_DESC                             MODULE_ID
---------- ----------------------------------- --------- --------- --------------------------------------------- ----------
A001       Database Updates                    05-MAY-24 15-MAY-24 Latest updates on Databases                   M001

SQL> |
```

Figure 33: Information Query 2

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

```
SELECT
    m.module_name,
    COUNT(r.resources_ID) AS total_resources
FROM
    Module m
LEFT JOIN
    Resources r ON m.module_ID = r.module_ID
WHERE
    m.module_name LIKE 'D%'
GROUP BY
    m.module_name;
```

The above query retrieves all the module that start with D and the total numbers of resources for that particular module. It select module_name from the module table and count the resources_id from the resources table using left join.



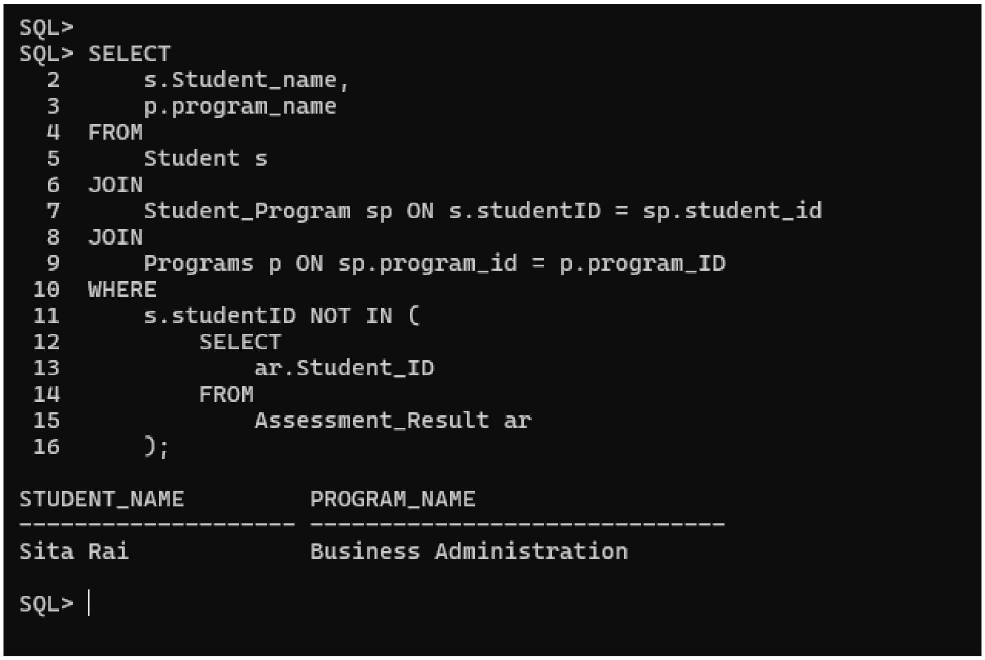
```
SQL> SELECT
         m.module_name,
         COUNT(r.resources_ID) AS total_resources
     FROM
         Module m
     LEFT JOIN
         Resources r ON m.module_ID = r.module_ID
     WHERE
         m.module_name LIKE 'D%'
     GROUP BY
         m.module_name;

MODULE_NAME                    TOTAL_RESOURCES
------------------------------ ---------------
Data Structures                              1
Digital Logic                                1
Distributed Systems                          1
Databases                                    1

SQL>
SQL>
```

Figure 34: Information Query 3.

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

```
SELECT
    s.Student_name,
    p.program_name
FROM
    Student s
JOIN
    Student_Program sp ON s.studentID = sp.student_id
JOIN
    Programs p ON sp.program_id = p.program_ID
WHERE
    s.studentID NOT IN (
        SELECT
            ar.Student_ID
        FROM
            Assessment_Result ar
    );
```

The query list the name of the student who are enrolled in program and have not submitted any assessment. It selects student_name from the student table and program_name from program tables. Than it joins student with student_program using student_ID and joins student_program with programs using program_id. The where clause filters student who have not submitted assessment.



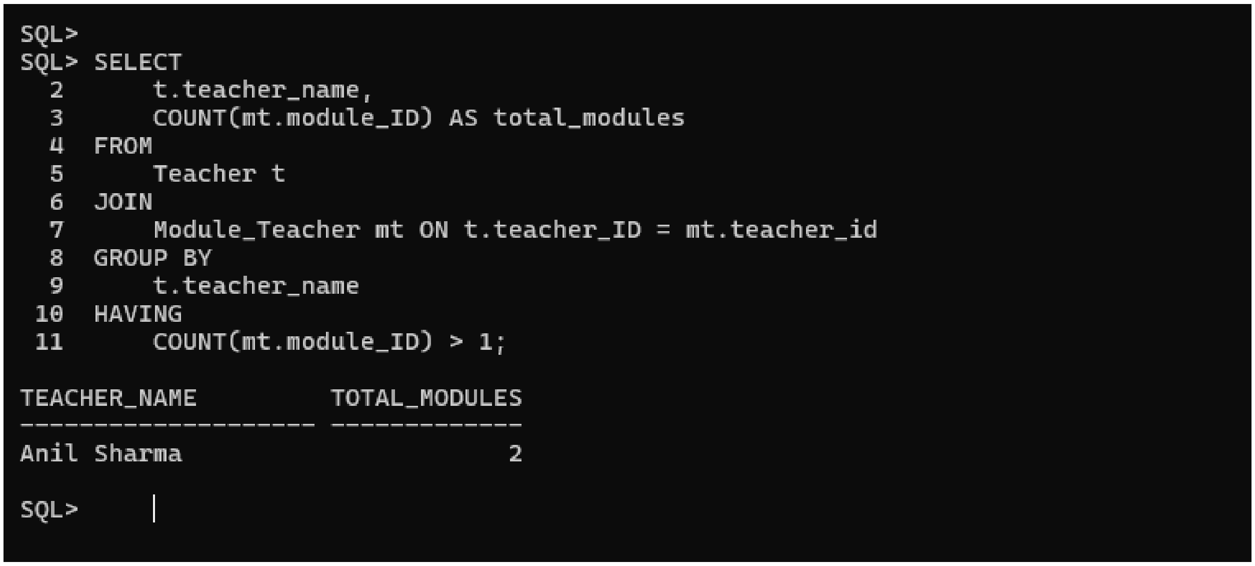
```
SQL>
SQL> SELECT
        s.Student_name,
        p.program_name
    FROM
        Student s
    JOIN
        Student_Program sp ON s.studentID = sp.student_id
    JOIN
        Programs p ON sp.program_id = p.program_ID
    WHERE
        s.studentID NOT IN (
            SELECT
                ar.Student_ID
            FROM
                Assessment_Result ar
        );

STUDENT_NAME         PROGRAM_NAME
-------------------- ------------------------------
Sita Rai             Business Administration

SQL> |
```

Figure 35: Information Query 4

5. List all the teachers who teach more than one module.

```
SELECT
   t.teacher_name,
   COUNT(mt.module_ID) AS total_modules
FROM
   Teacher t
JOIN
   Module_Teacher mt ON t.teacher_ID = mt.teacher_id
GROUP BY
   t.teacher_name
HAVING
   COUNT(mt.module_ID) > 1;
```

The query retrieves the name of the teacher and total modules they teaches and only display those teacher who teaches more than one module. It select teacher_name from the teacher table and counts the module_id from the module_teacher table. The query joins these two tables using teacher_id and then group by teacher_name.



```
SQL>
SQL> SELECT
         t.teacher_name,
         COUNT(mt.module_ID) AS total_modules
     FROM
         Teacher t
     JOIN
         Module_Teacher mt ON t.teacher_ID = mt.teacher_id
     GROUP BY
         t.teacher_name
     HAVING
         COUNT(mt.module_ID) > 1;

TEACHER_NAME         TOTAL_MODULES
-------------------- -------------
Anil Sharma                      2

SQL>
```

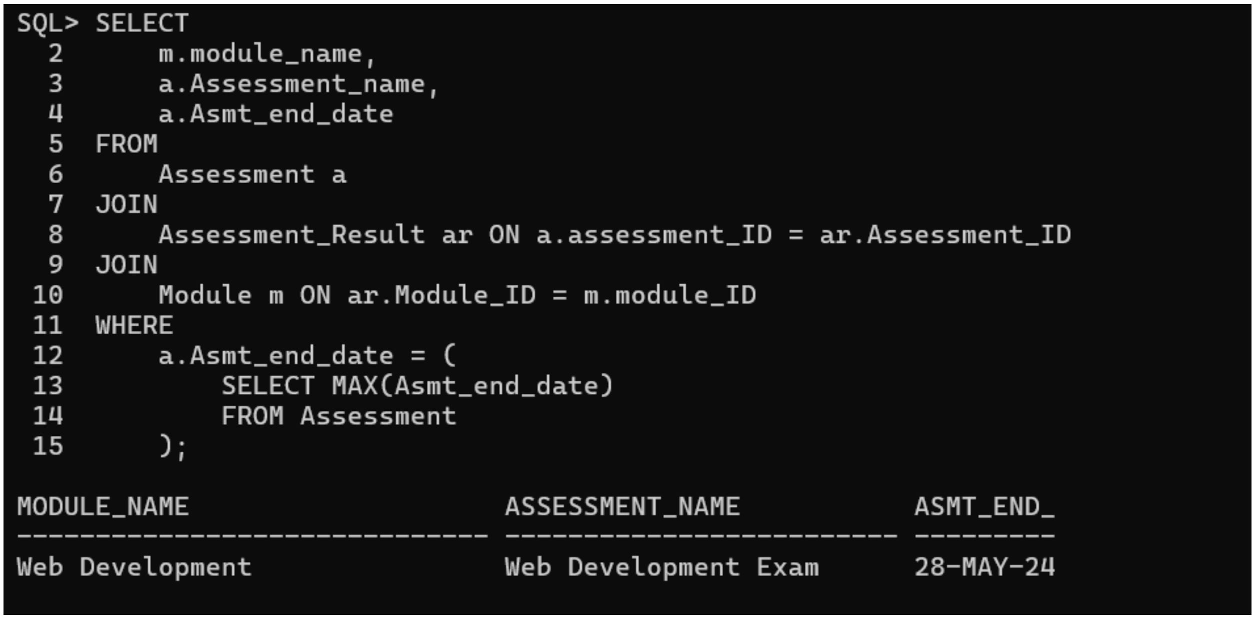
Figure 36: Information Query 5

6.2 Transaction query

1. Identify the module that has the latest assessment deadline.

```
SELECT
   m.module_name,
   a.Assessment_name,
   a.Asmt_end_date
FROM
   Assessment a
JOIN
   Assessment_Result ar ON a.assessment_ID = ar.Assessment_ID
JOIN
   Module m ON ar.Module_ID = m.module_ID
WHERE
   a.Asmt_end_date = (
      SELECT MAX(Asmt_end_date)
      FROM Assessment
   );
```

The query selects module name, assessment name and assessment end date for the assessment with the lasts deadline. It display the assessment which have lasts deadline. It join assessment, assessmen_result and module_id form their respective table. The where clause filters the assessment and only display the assessment that meets criteria.



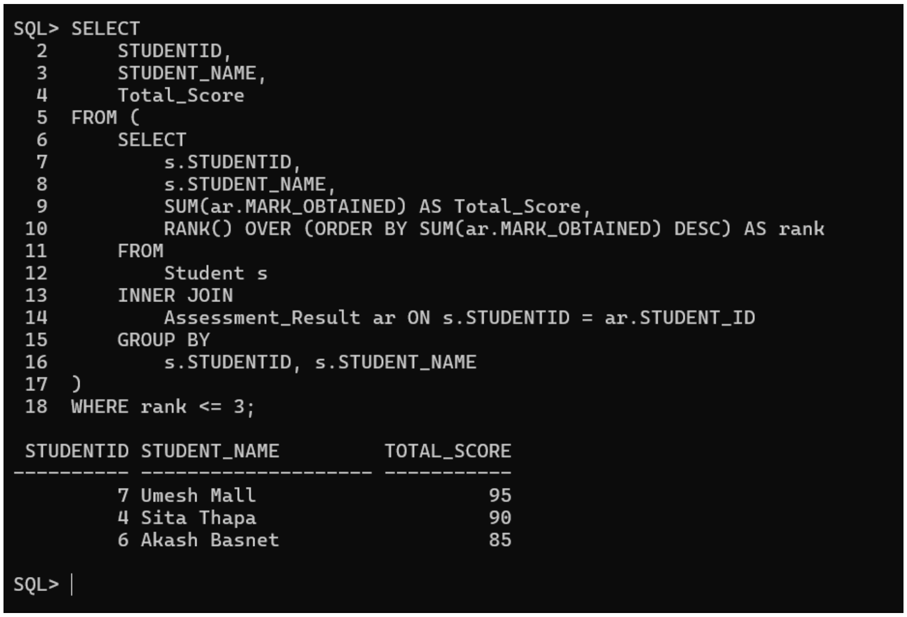
```
SQL> SELECT
         m.module_name,
         a.Assessment_name,
         a.Asmt_end_date
     FROM
         Assessment a
     JOIN
         Assessment_Result ar ON a.assessment_ID = ar.Assessment_ID
     JOIN
         Module m ON ar.Module_ID = m.module_ID
     WHERE
         a.Asmt_end_date = (
             SELECT MAX(Asmt_end_date)
             FROM Assessment
         );

MODULE_NAME                    ASSESSMENT_NAME           ASMT_END_
------------------------------ ------------------------- ---------
Web Development                Web Development Exam      28-MAY-24
```

Figure 37: Transaction Query 1.

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

```
SELECT
    STUDENTID,
    STUDENT_NAME,
    Total_Score
FROM (
    SELECT
        s.STUDENTID,
        s.STUDENT_NAME,
        SUM(ar.MARK_OBTAINED) AS Total_Score,
        RANK() OVER (ORDER BY SUM(ar.MARK_OBTAINED) DESC) AS rank
    FROM
        Student s
    INNER JOIN
        Assessment_Result ar ON s.STUDENTID = ar.STUDENT_ID
    GROUP BY
        s.STUDENTID, s.STUDENT_NAME
)
WHERE rank <= 3;
```

The query retrieves the top three student with have scored the highest total score across all the assessment. It join the student table with Assessment_Result table using student_ID and calculates the total scores for each student and groups the data by student_id and student_name. the where clause filters the results to include only top three student.



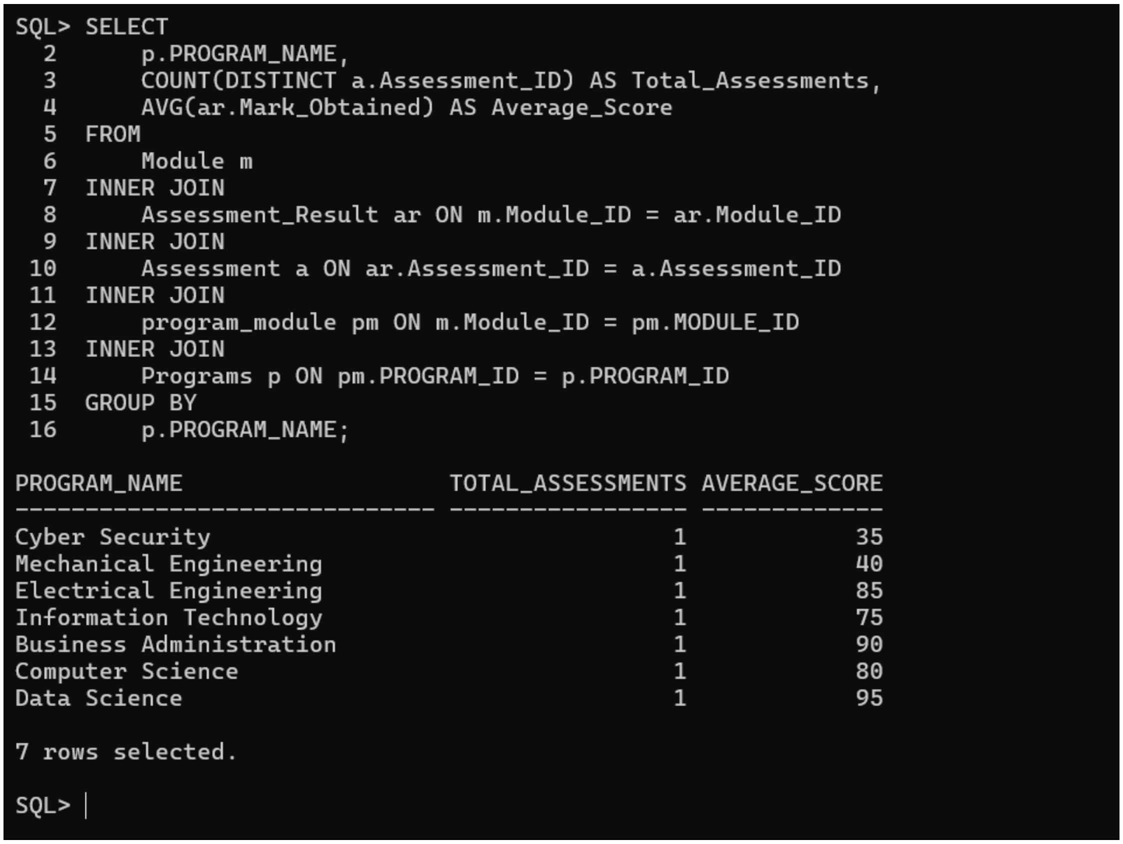
```
SQL> SELECT
         STUDENTID,
         STUDENT_NAME,
         Total_Score
     FROM (
         SELECT
             s.STUDENTID,
             s.STUDENT_NAME,
             SUM(ar.MARK_OBTAINED) AS Total_Score,
             RANK() OVER (ORDER BY SUM(ar.MARK_OBTAINED) DESC) AS rank
         FROM
             Student s
         INNER JOIN
             Assessment_Result ar ON s.STUDENTID = ar.STUDENT_ID
         GROUP BY
             s.STUDENTID, s.STUDENT_NAME
     )
     WHERE rank <= 3;

 STUDENTID STUDENT_NAME         TOTAL_SCORE
---------- -------------------- -----------
         7 Umesh Mall                    95
         4 Sita Thapa                    90
         6 Akash Basnet                  85

SQL> |
```

Figure 38: Transaction Query 2.

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

```
SELECT
   p.PROGRAM_NAME,
   COUNT(DISTINCT a.Assessment_ID) AS Total_Assessments,
   AVG(ar.Mark_Obtained) AS Average_Score
FROM
   Module m
INNER JOIN
   Assessment_Result ar ON m.Module_ID = ar.Module_ID
INNER JOIN
   Assessment a ON ar.Assessment_ID = a.Assessment_ID
INNER JOIN
   program_module pm ON m.Module_ID = pm.MODULE_ID
INNER JOIN
   Programs p ON pm.PROGRAM_ID = p.PROGRAM_ID
GROUP BY
   p.PROGRAM_NAME;
```

This query display the total number of assessment for each program and the average scores for each program. It combines data from multiples tables i.e. module, assessment_result and program_module and program. For each assessment the query calculates total number of assessment and the average mark obtained by the student. The result are grouped by program name.



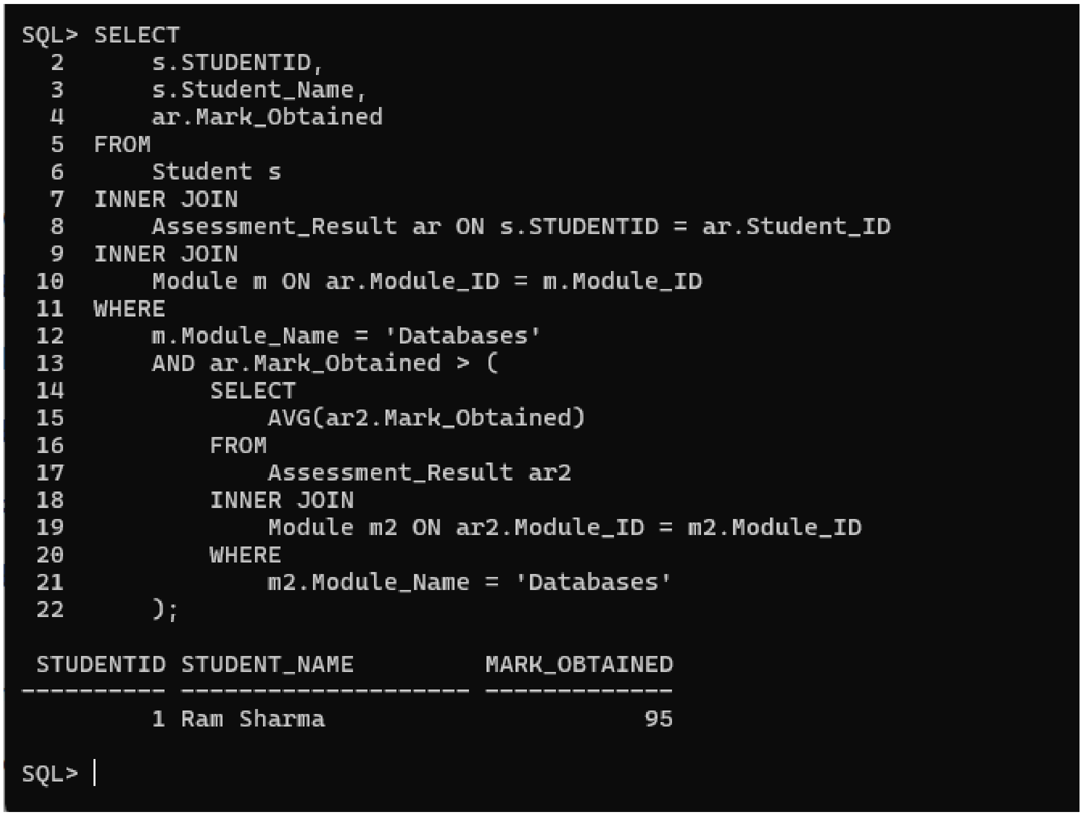
```
SQL> SELECT
         p.PROGRAM_NAME,
         COUNT(DISTINCT a.Assessment_ID) AS Total_Assessments,
         AVG(ar.Mark_Obtained) AS Average_Score
     FROM
         Module m
     INNER JOIN
         Assessment_Result ar ON m.Module_ID = ar.Module_ID
     INNER JOIN
         Assessment a ON ar.Assessment_ID = a.Assessment_ID
     INNER JOIN
         program_module pm ON m.Module_ID = pm.MODULE_ID
     INNER JOIN
         Programs p ON pm.PROGRAM_ID = p.PROGRAM_ID
     GROUP BY
         p.PROGRAM_NAME;

PROGRAM_NAME                   TOTAL_ASSESSMENTS AVERAGE_SCORE
------------------------------ ----------------- -------------
Cyber Security                                 1            35
Mechanical Engineering                         1            40
Electrical Engineering                         1            85
Information Technology                         1            75
Business Administration                        1            90
Computer Science                               1            80
Data Science                                   1            95

7 rows selected.

SQL>
```

Figure 39: Transaction Query 3.

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

```
SELECT
    s.STUDENTID,
    s.Student_Name,
    ar.Mark_Obtained
FROM
    Student s
INNER JOIN
    Assessment_Result ar ON s.STUDENTID = ar.Student_ID
INNER JOIN
    Module m ON ar.Module_ID = m.Module_ID
WHERE
    m.Module_Name = 'Databases'
    AND ar.Mark_Obtained > (
        SELECT
            AVG(ar2.Mark_Obtained)
        FROM
            Assessment_Result ar2
        INNER JOIN
            Module m2 ON ar2.Module_ID = m2.Module_ID
        WHERE
            m2.Module_Name = 'Databases'
    );
```

The query list all the student who have scored above the average mark in the database module. It first joins student, Assessment_Result and module tables to access required information. The where clause in the query filter the result and display only those student who have scored higher than the average. The subquery calculates the average mark obtained by the all student in the database modules.



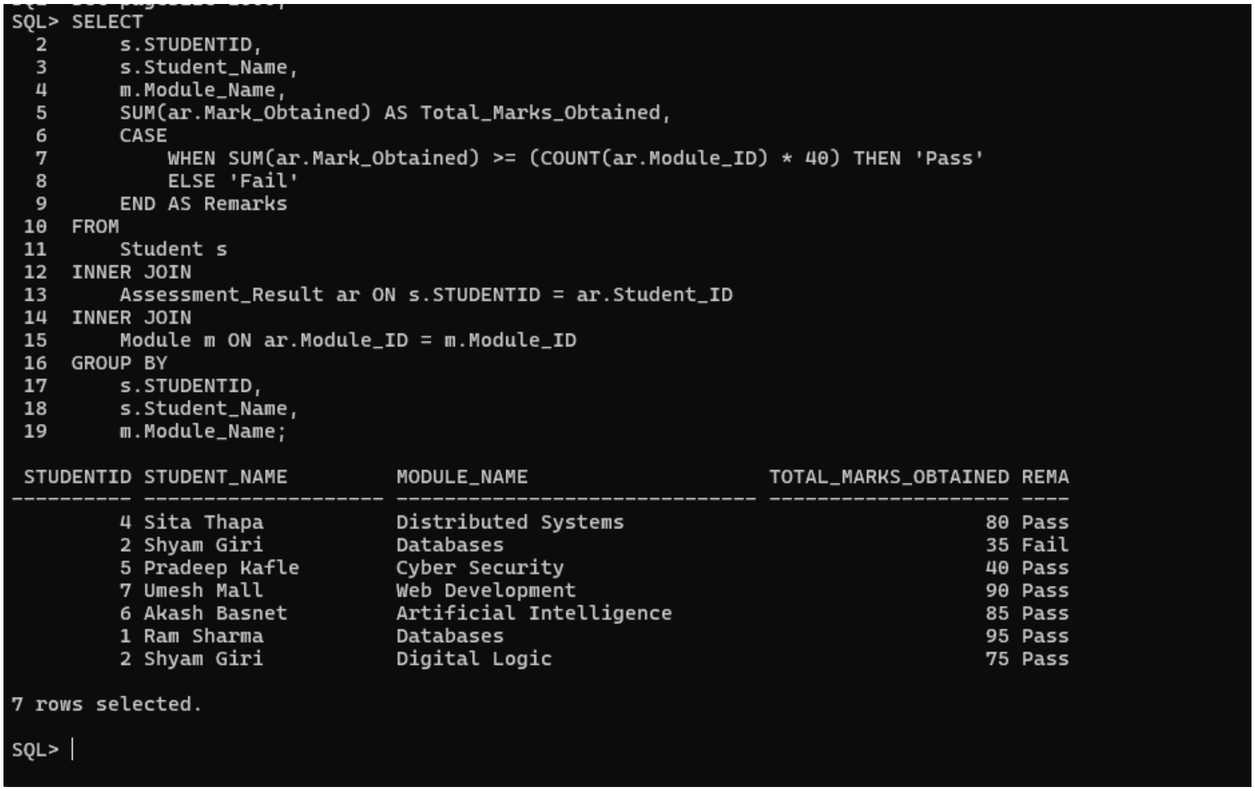
```
SQL> SELECT
         s.STUDENTID,
         s.Student_Name,
         ar.Mark_Obtained
     FROM
         Student s
     INNER JOIN
         Assessment_Result ar ON s.STUDENTID = ar.Student_ID
     INNER JOIN
         Module m ON ar.Module_ID = m.Module_ID
     WHERE
         m.Module_Name = 'Databases'
         AND ar.Mark_Obtained > (
             SELECT
                 AVG(ar2.Mark_Obtained)
             FROM
                 Assessment_Result ar2
             INNER JOIN
                 Module m2 ON ar2.Module_ID = m2.Module_ID
             WHERE
                 m2.Module_Name = 'Databases'
         );

 STUDENTID STUDENT_NAME         MARK_OBTAINED
---------- -------------------- -------------
         1 Ram Sharma                      95

SQL>
```

Figure 40: Transaction Query 4.

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

```
SELECT
   s.STUDENTID,
   s.Student_Name,
   m.Module_Name,
   SUM(ar.Mark_Obtained) AS Total_Marks_Obtained,
   CASE
      WHEN SUM(ar.Mark_Obtained) >= (COUNT(ar.Module_ID) * 40) THEN 'Pass'
      ELSE 'Fail'
   END AS Remarks
FROM
   Student s
INNER JOIN
   Assessment_Result ar ON s.STUDENTID = ar.Student_ID
INNER JOIN
   Module m ON ar.Module_ID = m.Module_ID
GROUP BY
   s.STUDENTID,
   s.Student_Name,
   m.Module_Name;
```

The query calculates the total mark obtained by the each student in particular module and determines whether they passed or failed. It joins student, Assessment_Result and module tables to gather student names, modules names and mark obtained. The total mark are sum for each student and module and case statement check the total mark is greater than 40 percent. If it is below forty percent than is marked as fail. student and module and case statement check the total mark is greater than 40 percent. If it is below forty percent than is marked as fail.



```
SQL> SELECT
         s.STUDENTID,
         s.Student_Name,
         m.Module_Name,
         SUM(ar.Mark_Obtained) AS Total_Marks_Obtained,
         CASE
             WHEN SUM(ar.Mark_Obtained) >= (COUNT(ar.Module_ID) * 40) THEN 'Pass'
             ELSE 'Fail'
         END AS Remarks
     FROM
         Student s
     INNER JOIN
         Assessment_Result ar ON s.STUDENTID = ar.Student_ID
     INNER JOIN
         Module m ON ar.Module_ID = m.Module_ID
     GROUP BY
         s.STUDENTID,
         s.Student_Name,
         m.Module_Name;

 STUDENTID STUDENT_NAME         MODULE_NAME                    TOTAL_MARKS_OBTAINED REMA
---------- -------------------- ------------------------------ -------------------- ----
         4 Sita Thapa           Distributed Systems                              80 Pass
         2 Shyam Giri           Databases                                        35 Fail
         5 Pradeep Kafle        Cyber Security                                   40 Pass
         7 Umesh Mall           Web Development                                  90 Pass
         6 Akash Basnet         Artificial Intelligence                          85 Pass
         1 Ram Sharma           Databases                                        95 Pass
         2 Shyam Giri           Digital Logic                                    75 Pass

7 rows selected.

SQL> |
```

Figure 41: Transaction Query 5.

7 Critical Evaluation

7.1 Critical Evaluation of Module.

The module “Database System” is one of the most interesting modules in this semester. The total credits hour of this module is 15 and its module code is CC5051NT. That’s why it’s only a semester module. The most important things about this module is that it provides the theoretical concepts of database as well as how to use those concepts in real word project for the design and development of functional and robust database. At starting we developed a solid foundation on relational database, which is very necessary for any IT professional. Later, we start learning normalization. I think the most important and most valuable concept from database is normalization. In normalization we studied multiples concepts like functional dependency, way to transfer key, how to make composite key, way to fin the FFD, PFD etc. We start normalization from UNF to 3NF. At some time, it was to frustrating to understand the core concepts of normalization.

Normalization is very crucial in real worlds project because it helps to minimize data redundancy, improves data integrity and the query performance can be increased. Normalized database is generally more flexible and easier to scale and are free from common anomalies like insertion, deletion and update anomalies. We also covered about ERD (Entity Relationship Diagram) which is the graphical representation of database. ERD is very useful for visualizing the relationship between different entities in the database. For practical implementation we used Oracle SQL Plus database that gives us hand on experiences on writing SQL queries.

Finally, I would to express my sincere gratitude to Mr. Ajayraj Bhattrai the module leaders for his guidance and support throughout the course. His insights and encouragement have made this learning experience more fruitful. I would also like to thank my friends who helped me to understand the difficult concepts. Their support and guidance made a huge difference. To conclude the “Database system” module was a great learning experience. While sometime it seems challenging but with the helps of the module leader and friends it became easier. It had provided us with the solid foundation, essential knowledge and important concept that we can use to move forward in our studies and career. We believe that the skills we learned from this module can be extremely useful

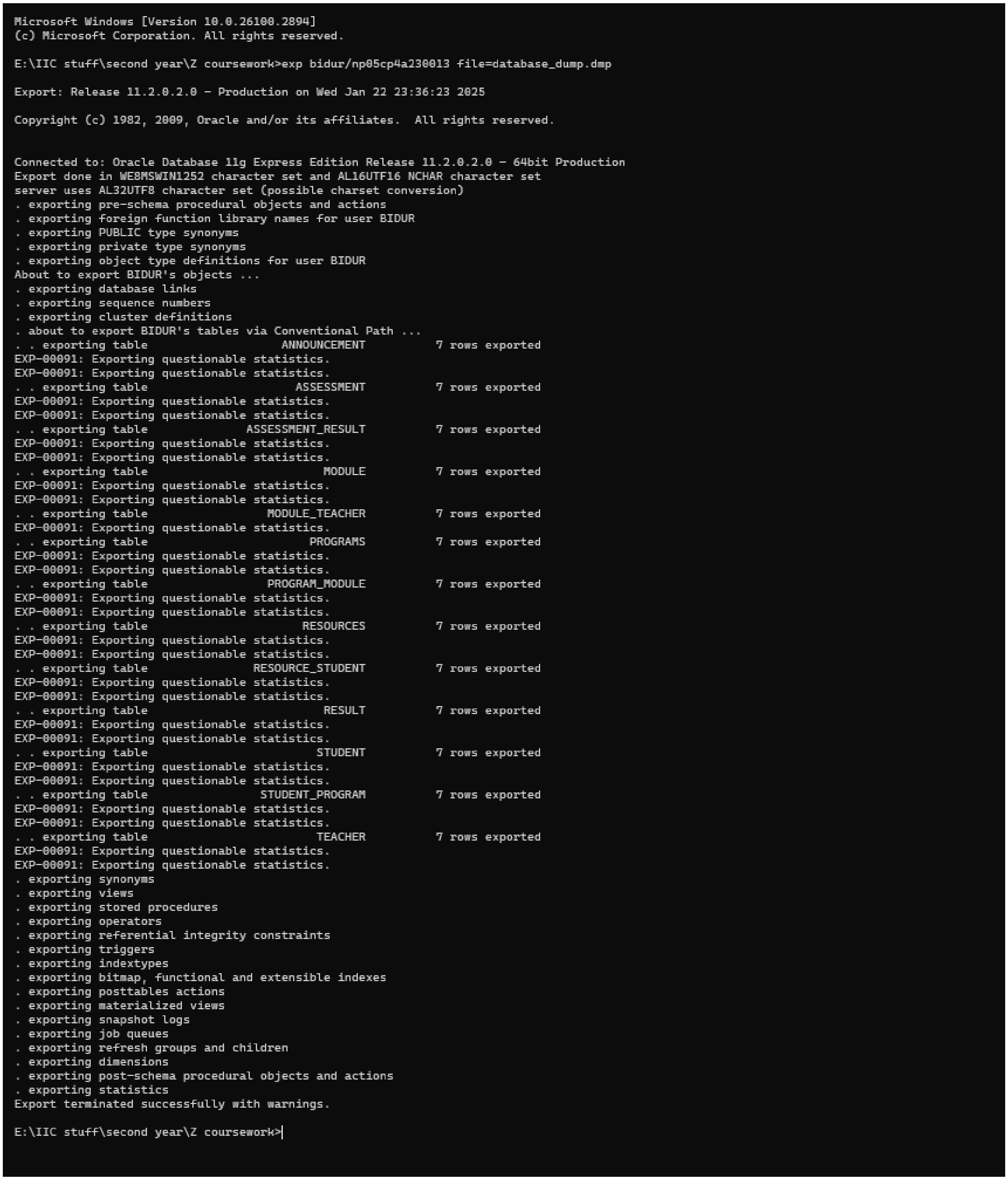
7.2 Critical Evaluation of Coursework

The coursework of “Database System” was one of the most important and necessary coursework that connects the gaps between theoretical knowledge and real worlds problem. The coursework was designed in such a way that it can reflect everything that are in module. In coursework we are supposed to designed the functional and robust database for Electronic-Classroom. It provides an opportunity to test the concept that we learned during lecture. The initial task was to identify the entities, attributes, relationship from the scenario. After that we perform normalization along with final ERD and implementation of database in oracle SQL plus. The hand on experiences in SQL query, normalization of database, designing of ERD wad truly rewarding. However, in some area of coursework like normalization and feeding sample data are too challenges. We need to feed data multiple time as the inserted sample data are not align with the query we need to perform. Additionally, Ten SQL query are given us to solve after feeding data which is good way to interact with the database and also help to understand the working process of query.

Lastly, I would like to express sincere gratitude to my module leader Mr. Ajayraj Bhattrai for his guidance throughout the coursework. His timely feedback and remarks help us to complete this coursework in time. I am also thankful for my friends who helps me to tackle some of the challenging part of this coursework. In, conclusion the coursework was too challenging at beginning but with proper research and understanding the coursework seems possible. This was the most valuable learning experiences that fills the gap between the theoretical concepts and real worlds context. Finally, the coursework provided me with a strong and solid foundation on database design, development and management and I am vey confident that it will helps in my future career. The problem and challenges that I faced during the coursework are only way that make the outcome more rewarding and I am very proud what I have achieved from this coursework as well as module too.

8 Drop Query and Database Dump File Creation.

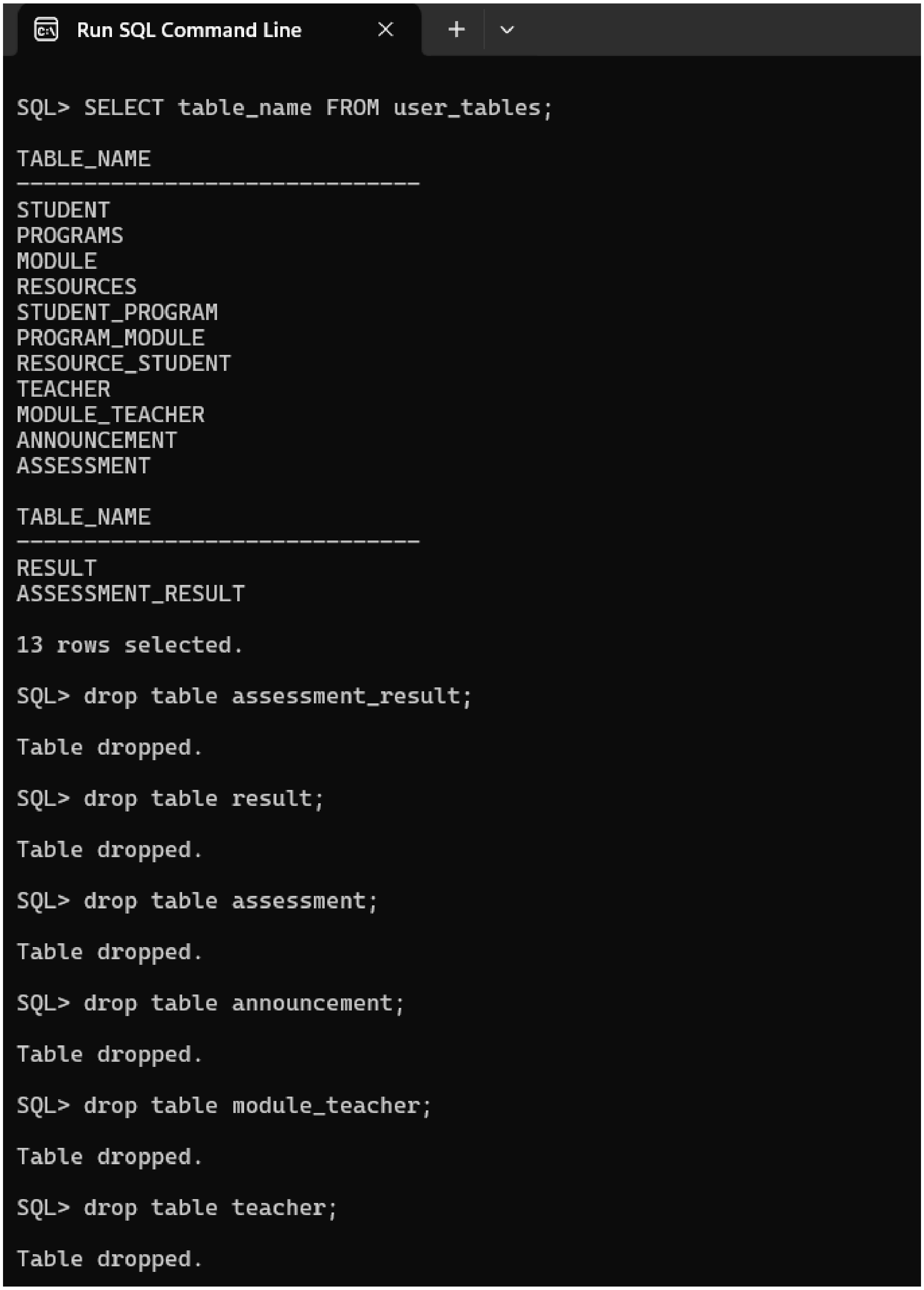
8.1 Database Dump File Creation



```
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

E:\IIC stuff\second year\Z coursework>exp bidur/np05cp4a230013 file=database_dump.dmp

Export: Release 11.2.0.2.0 - Production on Wed Jan 22 23:36:23 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 BIDUR
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user BIDUR
About to export BIDUR's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export BIDUR's tables via Conventional Path ...
. . exporting table                   ANNOUNCEMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                     ASSESSMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table              ASSESSMENT_RESULT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 MODULE_TEACHER          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                       PROGRAMS          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                 PROGRAM_MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                      RESOURCES          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table               RESOURCE_STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         RESULT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                STUDENT_PROGRAM          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                        TEACHER          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.

E:\IIC stuff\second year\Z coursework>
```

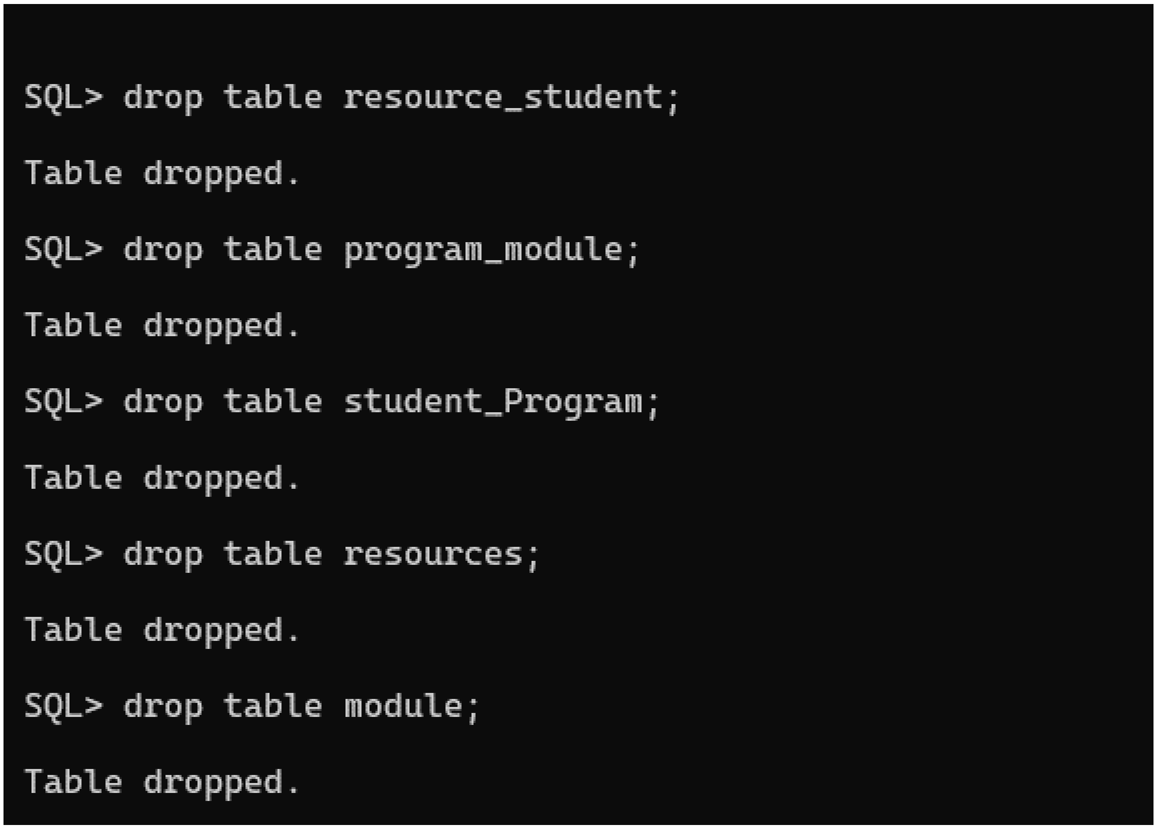
Figure 42: Creating Dump File.

8.2 Drop Query



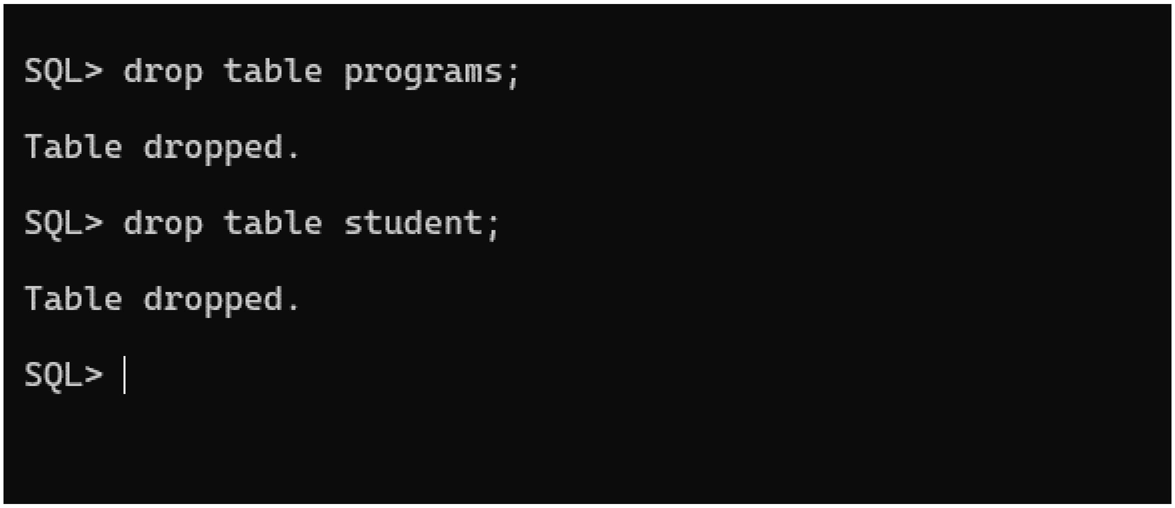
```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
------------------------------
STUDENT
PROGRAMS
MODULE
RESOURCES
STUDENT_PROGRAM
PROGRAM_MODULE
RESOURCE_STUDENT
TEACHER
MODULE_TEACHER
ANNOUNCEMENT
ASSESSMENT

TABLE_NAME
------------------------------
RESULT
ASSESSMENT_RESULT

13 rows selected.

SQL> drop table assessment_result;

Table dropped.

SQL> drop table result;

Table dropped.

SQL> drop table assessment;

Table dropped.

SQL> drop table announcement;

Table dropped.

SQL> drop table module_teacher;

Table dropped.

SQL> drop table teacher;

Table dropped.
```

Figure 43: Drop query 1 out of 3



```
SQL> drop table resource_student;

Table dropped.

SQL> drop table program_module;

Table dropped.

SQL> drop table student_Program;

Table dropped.

SQL> drop table resources;

Table dropped.

SQL> drop table module;

Table dropped.
```

Figure 44: Drop query 2 out of 3



```
SQL> drop table programs;

Table dropped.

SQL> drop table student;

Table dropped.

SQL> |
```

Figure 45: Drop query 3 out of 3

9 References

eTutorials, 2022. *What us business Rules ?.* [Online]
Available at:
https://etutorials.org/SQL/Database+design+for+mere+mortals/Part+II+The+Design+Process/Chapter+11.+Business+Rules/What+Are+Business+Rules/
[Accessed 17 12 2024].

Agarwal, K., 2023. *Functional dependency.* [Online]
Available at: https://takeuforward.org/dbms/functional-dependency
[Accessed 25 12 2024].

Chia, A., 2023. *What is data dictionary?.* [Online]
Available at: https://www.splunk.com/en_us/blog/learn/data-dictionary.html
[Accessed 30 12 2024].

DBpedia, 2023. *About: Unnormalized Form.* [Online]
Available at: https://dbpedia.org/page/Unnormalized_form
[Accessed 20 12 2024].

Fayard, M., 2024. *What is third normal form ?.* [Online]
Available at: https://www.datacamp.com/tutorial/third-normal-form
[Accessed 21 12 2024].

Gibbs, M., 2022. *First Noraml Form.* [Online]
Available at: https://study.com/academy/lesson/first-normal-form-in-dbms-with-examples.html
[Accessed 20 12 2024].

indeed, 2023. *What Is A Relationship In Database? (Definition And Types).* [Online]
Available at: https://in.indeed.com/career-advice/career-development/what-is-relationship-in-database#:~:text=A%20relationship%20in%20databases%20is,answer%20some%20frequently%20asked%20questions.
[Accessed 21 12 2024].

Jacqueline Biscobing, K. T. H., 2024. *ERD.* [Online]
Available at: https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-

diagram-ERD
[Accessed 09 08 2025].

Kozubek, A., 2020. *Normalization in Relational Databases.* [Online]
Available at: https://vertabelo.com/blog/normalization-1nf-2nf-3nf/
[Accessed 21 12 2024].

Rouse, M., 2023. *What is Normalization?*. [Online]
Available at: https://www.techopedia.com/definition/1221/normalization
[Accessed 20 12 2024].

10 Appendix

The query used to create tables: -

- **Creating student table**

```
CREATE TABLE Student (

   studentID NUMBER(10) NOT NULL,

   Student_name VARCHAR2(20) NOT NULL,

   Phone_Number VARCHAR2(20) NOT NULL,

   Email VARCHAR2(30) NOT NULL UNIQUE,

   Address VARCHAR2(20) NOT NULL,

   DOB DATE NOT NULL,

   CONSTRAINT student_pk PRIMARY KEY (studentID)

);
```

- **Creating Programs table**

```
CREATE TABLE Programs (

   program_ID NUMBER(10) NOT NULL,

   program_name VARCHAR(30) NOT NULL UNIQUE,

   duration NUMBER(10) NOT NULL,

   total_modules NUMBER(10) NOT NULL,

   Program_Type VARCHAR(20) NOT NULL,

   CONSTRAINT PK_Program PRIMARY KEY (program_ID)

);
```

- **Creating student_Program table**

```
CREATE TABLE Student_Program (

   student_id NUMBER(10) NOT NULL,

   program_id NUMBER(10) NOT NULL,

   CONSTRAINT FK_Student FOREIGN KEY (student_id) REFERENCES Student(studentID),

   CONSTRAINT FK_Program FOREIGN KEY (program_id) REFERENCES Programs(program_ID),

   CONSTRAINT PK_Student_Program PRIMARY KEY (student_id, program_id)

);
```

- **Creating module table**

```
CREATE TABLE Module (

   module_ID VARCHAR(10) NOT NULL,

   module_name VARCHAR(30) NOT NULL,

   credits_hours NUMBER(10) NOT NULL,

   Module_Type VARCHAR(20) NOT NULL,

   Module_Desc VARCHAR(25),

   CONSTRAINT Per_pk PRIMARY KEY (module_ID)

);
```

- **Creating Program_module table**

```
CREATE TABLE Program_Module (
    module_ID VARCHAR(10) NOT NULL,
    student_ID NUMBER(10) NOT NULL,
    program_ID NUMBER(10) NOT NULL,
    CONSTRAINT per_module FOREIGN KEY (module_ID) REFERENCES module(module_ID),
    CONSTRAINT sid FOREIGN KEY (student_ID, program_ID) REFERENCES Student_Program(student_id, program_id),
    CONSTRAINT per_program_module PRIMARY KEY (module_ID, student_ID, program_ID)
);
```

- **Creating student table**

```
CREATE TABLE Resources (
    resources_ID VARCHAR(10) NOT NULL,
    title VARCHAR(40) NOT NULL,
    resources_type VARCHAR(10) NOT NULL,
    duration NUMBER(10) NOT NULL,
    module_ID VARCHAR(10) NOT NULL,
    CONSTRAINT PK_Resources PRIMARY KEY (resources_ID),
    CONSTRAINT FK_Module_ID FOREIGN KEY (module_ID) REFERENCES module(module_ID)
);
```

- **Creating Resource_Student table**

```
CREATE TABLE Resource_Student (
   resources_ID VARCHAR(10) NOT NULL,
   student_ID NUMBER(10) NOT NULL,
   sequence_number VARCHAR(10) NOT NULL,
   CONSTRAINT PK_Resource_Student PRIMARY KEY (resources_ID, student_ID),
       CONSTRAINT FK_Resource FOREIGN KEY (resources_ID) REFERENCES Resources(resources_ID),
   CONSTRAINT FK_std FOREIGN KEY (student_ID) REFERENCES Student(studentID)
);
```

- **Creating teacher table**

```
CREATE TABLE Teacher (
   teacher_ID NUMBER(10) NOT NULL,
   teacher_name VARCHAR(20) NOT NULL,
   Teacher_email VARCHAR(30) NOT NULL UNIQUE,
   Contact_number VARCHAR(15) NOT NULL,
   department VARCHAR(20) NOT NULL,
   specialization VARCHAR(15) NOT NULL,
   CONSTRAINT  perteacherid PRIMARY KEY(teacher_id)
);
```

- **Creating Module_Teacher table**

```
CREATE TABLE Module_Teacher (
   teacher_id NUMBER(10) NOT NULL,
   module_ID VARCHAR(10) NOT NULL,
   PRIMARY KEY (teacher_id, module_id),
          CONSTRAINT fk_teachersid FOREIGN KEY (teacher_id) REFERENCES Teacher(teacher_ID),
         CONSTRAINT fk_moduleesid FOREIGN KEY (module_id) REFERENCES Module(module_ID)
);
```

- **Creating Announcement table**

```
CREATE TABLE Announcement (
   announcement_ID VARCHAR(10) NOT NULL,
   title VARCHAR(35) NOT NULL,
   date_posted DATE NOT NULL,
   end_date DATE NOT NULL,
   announcement_desc VARCHAR(45) NOT NULL,
   teacher_ID NUMBER(10) NOT NULL,
   module_ID VARCHAR(10) NOT NULL,
   CONSTRAINT PK_Announcement PRIMARY KEY (announcement_ID),
     CONSTRAINT FK_Teacher FOREIGN KEY (teacher_ID, module_ID) REFERENCES Module_Teacher(teacher_id, module_ID)
);
```

- **Creating Assessment table**

```
CREATE TABLE Assessment (
   assessment_ID VARCHAR(10) NOT NULL,
   Assessment_name VARCHAR(25) NOT NULL,
   Asmt_posted_date DATE NOT NULL,
   Asmt_end_date DATE NOT NULL,
   Weightage NUMBER NOT NULL,
   CONSTRAINT assessID PRIMARY KEY (assessment_ID)
);
```

- **Creating Result table**

```
CREATE TABLE Result (
   result_ID VARCHAR(10) NOT NULL,
   remarks VARCHAR(25),
   feedback VARCHAR(25) NOT NULL,
   CONSTRAINT PK_Result PRIMARY KEY (result_ID)
);
```

- **Creating Assessment_Result table**

```
CREATE TABLE Assessment_Result (
   Assessment_ID VARCHAR(10) NOT NULL,
   Module_ID VARCHAR(10) NOT NULL,
   Student_ID NUMBER(10) NOT NULL,
   Result_ID VARCHAR(10) NOT NULL,
   Mark_obtained NUMBER(10) NOT NULL,
   Full_mark NUMBER(10) NOT NULL,
    CONSTRAINT PK_Assessment_Result PRIMARY KEY (Assessment_ID, Module_ID, Student_ID),
     CONSTRAINT FK_Assessment FOREIGN KEY (Assessment_ID) REFERENCES Assessment(Assessment_ID),
   CONSTRAINT f_module FOREIGN KEY (Module_ID) REFERENCES Module(module_ID),
   CONSTRAINT f_studt FOREIGN KEY (Student_ID) REFERENCES Student(studentID),
   CONSTRAINT f_result FOREIGN KEY (Result_ID) REFERENCES Result(result_ID)
);
```

The query used to insert data:

1. **Inserting data in student Table.**

```
INSERT ALL
   INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
    VALUES (1, 'Ram Sharma', '9841234567', 'ram.sharma@example.com', 'Kathmandu', TO_DATE('2000-01-01', 'YYYY-MM-DD'))
   INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
     VALUES (2, 'Shyam Giri', '9842234567', 'shyam.giri@example.com', 'Pokhara', TO_DATE('2001-02-10', 'YYYY-MM-DD'))
   INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
   VALUES (3, 'Sita Rai', '9843234567', 'sita.rai@example.com', 'Lalitpur', TO_DATE('1999-03-15', 'YYYY-MM-DD'))
   INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
```

```
    VALUES (4, 'Sita Thapa', '9844234567', 'sita.thapa@example.com', 'Biratnagar', TO_DATE('2002-04-20', 'YYYY-MM-DD'))
  INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
    VALUES (5, 'Pradeep Kafle', '9845234567', 'pradeep.kafle@example.com', 'Bhairahawa', TO_DATE('2000-05-25', 'YYYY-MM-DD'))
  INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
    VALUES (6, 'Akash Basnet', '9846234567', 'akash.basnet@example.com', 'Dhangadhi', TO_DATE('1998-06-30', 'YYYY-MM-DD'))
  INTO Student (studentID, Student_name, Phone_Number, Email, Address, DOB)
    VALUES (7, 'Umesh Mall', '9847234567', 'umesh.mall@example.com', 'Nepalgunj', TO_DATE('1997-07-18', 'YYYY-MM-DD'))
SELECT * FROM dual;
```

2. **Inserting data in Program Table.**

```
INSERT ALL
  INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
  VALUES (1, 'Computer Science', 4, 10, 'Undergraduate')
  INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
  VALUES (2, 'Information Technology', 3, 8, 'Undergraduate')
  INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
  VALUES (3, 'Business Administration', 2, 6, 'Postgraduate')
  INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
  VALUES (4, 'Mechanical Engineering', 4, 12, 'Undergraduate')
  INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
```

```
    VALUES (5, 'Electrical Engineering', 4, 11, 'Undergraduate')
    INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
    VALUES (6, 'Data Science', 3, 9, 'Postgraduate')
    INTO Programs (program_ID, program_name, duration, total_modules, Program_Type)
    VALUES (7, 'Cyber Security', 2, 7, 'Postgraduate')
SELECT * FROM dual;
```

3. Inserting data in student_Program Table.

```
INSERT ALL
    INTO Student_Program (student_id, program_id)
    VALUES (1, 1)
    INTO Student_Program (student_id, program_id)
    VALUES (2, 2)
    INTO Student_Program (student_id, program_id)
    VALUES (3, 3)
    INTO Student_Program (student_id, program_id)
    VALUES (4, 4)
    INTO Student_Program (student_id, program_id)
    VALUES (5, 5)
    INTO Student_Program (student_id, program_id)
    VALUES (6, 6)
    INTO Student_Program (student_id, program_id)
```

```
   VALUES (7, 7)
SELECT * FROM dual;
```

4. Inserting data in Module Table.

```
INSERT ALL
   INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
   VALUES ('M001', 'Databases', 4, 'Core', 'Covers DBMS concepts')
   INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
   VALUES ('M002', 'Data Structures', 3, 'Core', 'Focuses on algorithms')
   INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
   VALUES ('M003', 'Digital Logic', 3, 'Core', 'Learn digital systems')
   INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
   VALUES ('M004', 'Distributed Systems', 4, 'Elective', ' distributed computing')
   INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
   VALUES ('M005', 'Cyber Security', 3, 'Core', 'Introduction to security')
   INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
   VALUES ('M006', 'Artificial Intelligence', 4, 'Elective', 'Intro to AI concepts')
   INTO Module (module_ID, module_name, credits_hours, Module_Type, Module_Desc)
   VALUES ('M007', 'Web Development', 3, 'Elective', 'Learn web technologies')
SELECT * FROM dual;
```

5. Inserting data in Program_Module Table.

```
INSERT ALL
   INTO Program_Module (module_ID, student_ID, program_ID)
   VALUES ('M001', 1, 1)
   INTO Program_Module (module_ID, student_ID, program_ID)
   VALUES ('M002', 7, 7)
   INTO Program_Module (module_ID, student_ID, program_ID)
   VALUES ('M003', 2, 2)
   INTO Program_Module (module_ID, student_ID, program_ID)
   VALUES ('M004', 3, 3)
   INTO Program_Module (module_ID, student_ID, program_ID)
   VALUES ('M005', 4, 4)
   INTO Program_Module (module_ID, student_ID, program_ID)
   VALUES ('M006', 5, 5)
   INTO Program_Module (module_ID, student_ID, program_ID)
   VALUES ('M007', 6, 6)
SELECT * FROM dual;
```

6. Inserting data in Resources Table.

```
INSERT ALL
   INTO Resources (resources_ID, title, resources_type, duration, module_ID)
   VALUES ('R001', 'Introduction to Databases', 'Video', 120, 'M001')
   INTO Resources (resources_ID, title, resources_type, duration, module_ID)
   VALUES ('R002', 'Data Structures Basics', 'PDF', 45, 'M002')
   INTO Resources (resources_ID, title, resources_type, duration, module_ID)
   VALUES ('R003', 'Digital Logic Lecture', 'Video', 90, 'M003')
   INTO Resources (resources_ID, title, resources_type, duration, module_ID)
   VALUES ('R004', 'Distributed Systems Guide', 'PDF', 60, 'M004')
   INTO Resources (resources_ID, title, resources_type, duration, module_ID)
   VALUES ('R005', 'Cyber Security Fundamentals', 'Video', 150, 'M005')
   INTO Resources (resources_ID, title, resources_type, duration, module_ID)
   VALUES ('R006', 'AI Concepts Overview', 'Article', 30, 'M006')
   INTO Resources (resources_ID, title, resources_type, duration, module_ID)
   VALUES ('R007', 'Web Development Tutorial', 'Video', 180, 'M007')
SELECT * FROM dual;
```

7. Inserting data in Resource_Student Table.

```
INSERT ALL
   INTO Resource_Student (resources_ID, student_ID, sequence_number)
   VALUES ('R001', 1, 'S001')
   INTO Resource_Student (resources_ID, student_ID, sequence_number)
   VALUES ('R002', 2, 'S002')
   INTO Resource_Student (resources_ID, student_ID, sequence_number)
   VALUES ('R003', 3, 'S003')
   INTO Resource_Student (resources_ID, student_ID, sequence_number)
   VALUES ('R004', 4, 'S004')
   INTO Resource_Student (resources_ID, student_ID, sequence_number)
   VALUES ('R005', 5, 'S005')
   INTO Resource_Student (resources_ID, student_ID, sequence_number)
   VALUES ('R006', 6, 'S006')
   INTO Resource_Student (resources_ID, student_ID, sequence_number)
   VALUES ('R007', 7, 'S007')
SELECT * FROM dual;
```

8. Inserting data in Teacher Table.

```
INSERT ALL
     INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
    VALUES (1, 'Anil Sharma', 'anil.sharma@example.com', '9801001001', 'Computer Science', 'Databases')
     INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
   VALUES (2, 'Sunita Joshi', 'sunita.joshi@example.com', '9802002002', 'Cyber Security', 'Cyber Security')
     INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
     VALUES (3, 'Ramesh Thapa', 'ramesh.thapa@example.com', '9803003003', 'Electronics', 'Digital Logic')
     INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)
    VALUES (4, 'Meera Gupta', 'meera.gupta@example.com', '9804004004', 'Computer Science', 'Distributed Systems')

    INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)

    VALUES (5, 'Krishna Shrestha', 'krishna.shrestha@example.com', '9805005005', 'Artificial Intelligence', 'Artificial Intelligence')

    INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)

    VALUES (6, 'Pooja Basnet', 'pooja.basnet@example.com', '9806006006', 'Computer Science', 'Web Development')

    INTO Teacher (teacher_ID, teacher_name, Teacher_email, Contact_number, department, specialization)

    VALUES (7, 'Bijay Rai', 'bijay.rai@example.com', '9807007007', 'Computer Science', 'Data Structures')

SELECT * FROM dual;
```

9. Inserting data in Module_Teacher Table.

```
INSERT ALL

    INTO Module_Teacher (teacher_id, module_ID)

    VALUES (1, 'M001')

    INTO Module_Teacher (teacher_id, module_ID)

    VALUES (2, 'M005')

    INTO Module_Teacher (teacher_id, module_ID)

    VALUES (3, 'M003')
    INTO Module_Teacher (teacher_id, module_ID)

    VALUES (4, 'M004')

    INTO Module_Teacher (teacher_id, module_ID)

    VALUES (5, 'M006')

    INTO Module_Teacher (teacher_id, module_ID)

    VALUES (6, 'M007')

    INTO Module_Teacher (teacher_id, module_ID)

    VALUES (1, 'M002')

SELECT * FROM dual;
```

10. Inserting data in Announcement Table.

```
INSERT ALL

    INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)

        VALUES ('A001', 'Database Updates', TO_DATE('2024-05-05', 'YYYY-MM-DD'), TO_DATE('2024-05-15', 'YYYY-MM-DD'), 'Latest updates on Databases', 1, 'M001')

    INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)

        VALUES ('A002', 'AI Workshop', TO_DATE('2024-06-10', 'YYYY-MM-DD'), TO_DATE('2024-06-20', 'YYYY-MM-DD'), 'Workshop on AI advancements', 1, 'M002') -- Updated

    INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)
```

```
  VALUES ('A003', 'Cybersecurity Essentials', TO_DATE('2025-05-15', 'YYYY-MM-DD'), TO_DATE('2025-05-25', 'YYYY-MM-DD'), 'Essentials of cybersecurity', 3, 'M003')

  INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)

  VALUES ('A004', 'Distributed Systems Updates', TO_DATE('2025-05-18', 'YYYY-MM-DD'), TO_DATE('2025-05-28', 'YYYY-MM-DD'), 'Updates on Distributed Systems', 4, 'M004')

  INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)

   VALUES ('A005', 'Advanced Databases', TO_DATE('2025-05-08', 'YYYY-MM-DD'), TO_DATE('2025-06-18', 'YYYY-MM-DD'), 'Advanced topics in DBMS', 2, 'M005') -- Updated

  INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)

  VALUES ('A006', 'Data Structures Seminar', TO_DATE('2025-04-07', 'YYYY-MM-DD'), TO_DATE('2025-04-1', 'YYYY-MM-DD'), 'Seminar on Data Structures', 5, 'M006') -- Updated

  INTO Announcement (announcement_ID, title, date_posted, end_date, announcement_desc, teacher_ID, module_ID)

 VALUES ('A007', 'Web Development Workshop', TO_DATE('2025-01-12', 'YYYY-MM-DD'), TO_DATE('2025-01-22', 'YYYY-MM-DD'), 'Workshop on web development', 6, 'M007') -- Updated

SELECT * FROM dual;
```

11. Inserting data in Assessment Table.

```
INSERT ALL

  INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)
```

```
    VALUES ('AS001', 'Database Quiz', TO_DATE('2024-05-01', 'YYYY-MM-DD'), TO_DATE('2024-05-10', 'YYYY-MM-DD'), 20)

   INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)

    VALUES ('AS002', 'AI Assignment', TO_DATE('2024-05-05', 'YYYY-MM-DD'), TO_DATE('2024-05-15', 'YYYY-MM-DD'), 25)

   INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)

   VALUES ('AS003', 'Cybersecurity Exam', TO_DATE('2024-05-08', 'YYYY-MM-DD'), TO_DATE('2024-05-20', 'YYYY-MM-DD'), 30)

   INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)

  VALUES ('AS004', 'Distributed Systems Lab', TO_DATE('2024-05-10', 'YYYY-MM-DD'), TO_DATE('2024-05-18', 'YYYY-MM-DD'), 15)

   INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)

  VALUES ('AS005', 'Advanced Database Project', TO_DATE('2024-05-12', 'YYYY-MM-DD'), TO_DATE('2024-05-22', 'YYYY-MM-DD'), 40)

   INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)

  VALUES ('AS006', 'Data Structures Practical', TO_DATE('2024-05-15', 'YYYY-MM-DD'), TO_DATE('2024-05-25', 'YYYY-MM-DD'), 35)

   INTO Assessment (assessment_ID, assessment_name, asmt_posted_date, asmt_end_date, weightage)

  VALUES ('AS007', 'Web Development Exam', TO_DATE('2024-05-18', 'YYYY-MM-DD'), TO_DATE('2024-05-28', 'YYYY-MM-DD'), 50)
```

```
SELECT * FROM dual;
```

12. Inserting data in Assessment_Result Table.

```
INSERT ALL
        INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID,
Mark_obtained, Full_mark)
   VALUES ('AS001', 'M001', 1, 'R001', 88, 90)
        INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID,
Mark_obtained, Full_mark)
   VALUES ('AS002', 'M002', 2, 'R002', 35, 100)
        INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID,
Mark_obtained, Full_mark)
   VALUES ('AS003', 'M003', 2, 'R001', 75, 100)
        INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID,
Mark_obtained, Full_mark)
   VALUES ('AS004', 'M004', 4, 'R001', 80, 100)
        INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID,
Mark_obtained, Full_mark)
   VALUES ('AS005', 'M005', 5, 'R002', 40, 100)
        INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID,
Mark_obtained, Full_mark)
   VALUES ('AS006', 'M006', 6, 'R003', 85, 100)
        INTO Assessment_Result (Assessment_ID, Module_ID, Student_ID, Result_ID,
Mark_obtained, Full_mark)
   VALUES ('AS007', 'M007', 7, 'R001', 90, 100)
```

```
SELECT * FROM dual;
```

13. Inserting data in Result Table.

```
INSERT ALL
   INTO Result (result_ID, remarks, feedback)
   VALUES ('R001', 'Pass', 'Good performance')
   INTO Result (result_ID, remarks, feedback)
   VALUES ('R002', 'Fail', 'Needs improvement')
   INTO Result (result_ID, remarks, feedback)
   VALUES ('R003', 'Pass', 'Well done')
   INTO Result (result_ID, remarks, feedback)
   VALUES ('R004', 'Pass', 'Good progress')
   INTO Result (result_ID, remarks, feedback)
   VALUES ('R005', 'Fail', 'More effort required')
   INTO Result (result_ID, remarks, feedback)
   VALUES ('R006', 'Pass', 'Satisfactory work')
   INTO Result (result_ID, remarks, feedback)
   VALUES ('R007', 'Pass', 'Excellent')
SELECT * FROM dual;
```