CC5051NI Databases

100% Individual Coursework

Autumn 2024

Credit: 15 Semester Long Module

Student Name: Sujal Parajuli

London Met ID: 23050262

Assignment Submission Date: January 23

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

23050262 Sujal Parajuli (Database) 1.docx

Islington College,Nepal

Document Details

Submission ID

trn:oid:::3618:79887232

Submission Date

Jan 23, 2025, 1:09 AM GMT+5:45

Download Date

Jan 23, 2025, 1:15 AM GMT+5:45

File Name

23050262 Sujal Parajuli (Database) 1.docx

File Size

32.3 KB

69 Pages

4,027 Words

26,447 Characters

34% Overall Similarity

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

Match Groups

- 124 Not Cited or Quoted 32%
  Matches with neither in-text citation nor quotation marks
- 9 Missing Quotations 2%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

11% Internet sources

0% Publications

30% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

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

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

Table of Contents:

Table of Figure:

Table of Table:

1. Introduction

St. Mary's College of Technology was established in 2010 as one of the leading colleges in teaching technology to young students. Their college aims to bring in skilled tech professionals by practically teaching them with modern teaching methodology. They believe in making quality education available to all students. Their college has excellent computer labs, a big library, and modern classrooms that help students learn better.

Figure 1 College Picture

The college is also located on a beautiful campus, with facilities for all types of sports, including a basketball court and a gym. Here, one can enjoy both studies and sports.

The college has grown successfully over the years with the help of its founder and principal, Ms. Mary. She has very recently proposed starting an ambitious project for launching the "E-Classroom Platform" online. Such a project will revolutionize the area of education by connecting students, tutors, and academic programs in an integrated digital environment for effectively managing purposes. As a database designer, my responsibility is to develop a robust database system that will efficiently support the operational needs of this platform. This system will ensure smooth tracking and management of critical data, including student information, program detail, etc. The E-Classroom setup will likely cover several academic programs. In which we're talking Bachelor of Science tracks like Computing, Networking, and Multimedia, etc. The database system is designed to efficiently allocate teachers to classes and ensure that each course is well supported with qualified instructors. Each class will have mechanisms for assessing the performance of the students through various tests and assignments. Details regarding these assessments will be elaborately linked to their respective classes, providing essential information such as assessment ID, title, due date, and weightage.

2. Current Business Activities and operations:

Ms. Mary is trying to start an online platform that is an E-Classroom that will provide a virtual learning environment for a college and its individuals. It will manage students, teachers, and programs with courses such as BSc in Computing, Networking, and Multimedia. Every program will have modules, assessments, resources, and results to keep track of the progress of the students. Teachers will create modules, make announcements, and lead students through structured and efficient learning. It ensures that students go through the resources in order and helps them organize their work. It will make learning easier and more accessible for all concerned individuals.

3. Business Rule:

Business rules determine the way a database will arrange the data for presentation and process that data for accuracy and reliability. For this platform, they provide well-defined structures among students, programs, modules, assessments, and resources to allow for greater efficiency in tracking progress and assignment management to achieve a planned and proper end goal.

The E-Classroom Platform has established its own set of business rules to control the relationships between students, teachers, programs, and modules:

- One student must enroll in exactly one program whereas one program can have multiple enrolled students.
- One program has several compulsory modules that a student needs to complete.
- One module can be part of multiple programs, and one program can have multiple modules.
- One module can have multiple assessments, and each assessment belongs to only one module.

- To complete a module, a student needs to finish all the tasks and assessments of that module.

- One teacher can be assigned to teach multiple modules, and a single module can be taught by multiple teachers.

- One teacher post multiple announcement in his/her assigned modules.

- Once submitted, students cannot edit or delete the submission of their assessments.

- One module can have multiple resources, and multiple resources can be in a single module.

- Resources in a module must be completed in a linear fashion and are only unlocked once previous resources have been completed.

- One Teacher can give multiple announcements in any module they have been assigned to teach.

- One Teacher is assigned to grade the assessment of multiple students.

4. Assumption

For this case study, I have made some assumptions which have been integrated into the list of business rules which are given below:

a. One module can have multiple assessments, and each assessment belongs to only one module.

b. To complete a module, a student needs to finish all the tasks and assessments of that module.

c. One Teacher is assigned to grade the assessment of multiple students.

d. One module can have multiple resources, and multiple resources can be in a single module.

5. Initial ERD

a. Identification of Entities and Attributes:

In the project, the entities are all the things which are an integral part of the system, and which must be stored and maintained in the database. The attributes are the characteristics or details of each entity.

Student Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Student_ID	Number	10	Primary Key
2	Student_Name	Character	40	Not Null
3	DOB	Date	-	Not Null
4	Enrollment_Date	Date	-	Not Null
5	Std_Email	Character	30	Unique

Table 1 (Entity and Attribute table for Student)

Program Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Program_ID	Number	10	Primary Key
2	Program_Name	Character	40	Not Null
3	P_Duration	Date	-	Not Null
4	P_Description	Character	100	Not Null

Table 2 (Entity and Attribute table for Program)

Module Table

S.No.	Attribute Name	Data Type	Size	Constraint
1.	Module_ID	Number	10	Primary Key
2.	Module_Name	Character	40	Not Null
3.	M_Duration	Date	-	Not Null
4.	M_Description	Character	100	Not Null
5.	M_Credits	Number	10	No Null
6.	Teacher_ID	Number	10	Not Null
7.	Teacher_Name	Character	40	Not Null
8.	Teach_Specialization	Character	30	Not Null
9.	Teach_Email	Character	30	Unique
10.	Assessment_ID	Number	10	Not Null
11.	Asse_Description	Character	40	Not Null
12.	Asse_Weightage	Character	50	Not Null
13.	Asse_Deadline	Date	-	Not Null
14.	Asse_Status	Character	10	Not Null
15.	Marks_Obtained	Number	3	Not Null
16.	Grade_Obtained	Character	1	Not Null
17.	Announcement_ID	Number	10	Not Null
18.	Date_Posted	Date	-	Not Null
19.	Ann_Title	Character	20	Not Null
20.	Ann_Content	Character	100	Not Null
21.	Resource_ID	Number	10	Not Null
22.	R_Duration	Date	-	Not Null
23.	R_Title	Character	20	Not Null
24.	R_Type	Character	30	Not Null
25.	Sequence_Order	Character	30	Not Null

Table 3 (Entity and Attribute table for Module)

6. Initial Entity Relationship Diagram (E.R.D)

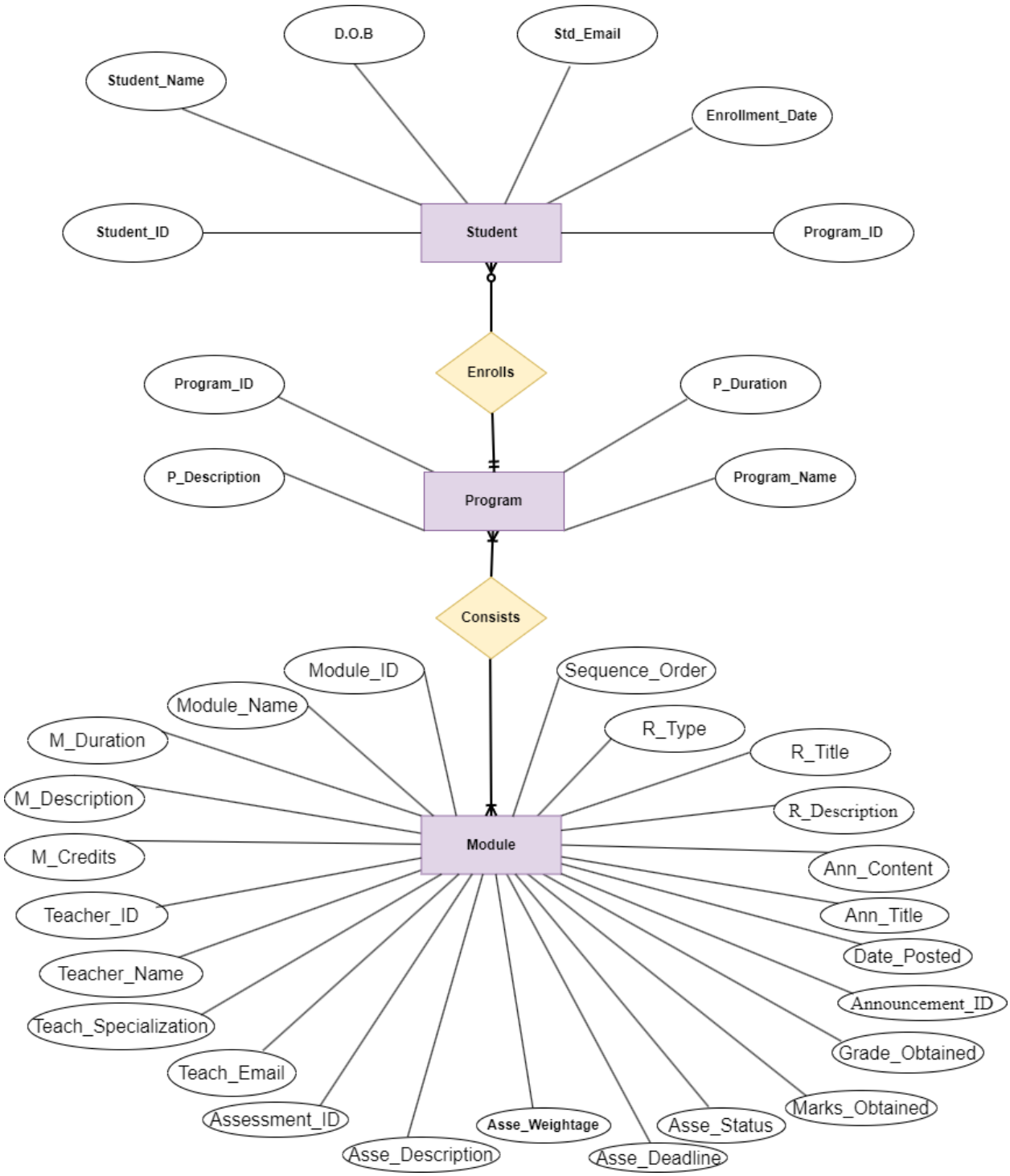

Figure 2 Initial ERD

7. Normalization:

It is a data organizing processing a database in such a manner that duplication reduces and accuracy increases. The normalizing of databases makes them perform optimally by reducing lots of problems during adding, deleting, or updating data. In normalization, data is organized in tables, and the relationship between different tables is established with the help of certain rules called normal forms. These rules keep the database organized, efficient, and error-free (Geeksofgeeks, 2024).

7.1. Un-Normalized Form (UNF):

- Here Data is stored in the form it is collected, and no effort is made to eliminate redundancy or guarantee atomicity.
- It allows repeating groups and non-atomic attributes (attributes with multiple values or sets).

Organizing data for UNF:

Student: (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID, Program_Name, P_Duration, P_Description {Module_ID, Module_Name, M_Duration, M_Description, M_Credits { Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email, { Announcement_ID, Date_Posted, Ann_Title, Ann_Content}},

Resource_ID, R_Description, R_Title, R_Type, Sequence_Order}, {Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline, Asse_Status, Marks_Obtained, Grade_Obtained}})

7.2. First Normalization Form (1NF):

Rules for 1NF

- No repeating values in a group
- No repeating groups

UNF to 1NF:

- Repeating groups (nested sets like {Student}, {Module}) were separated into different rows.
- All attributes were atomic; for example, no {Resource_ID, Title} in one field.

First Normal Form Tables:

i. **Student Table for 1NF:**

- **Student-1: (**Student_ID, Student_Name, DOB, Enrollment_Date, DOB, Std_Email, Program_ID, Program_Name, P_Duration, P_Description)

ii. **Student and Module Table for 1NF:**

- **Module_Student1: (**Module_ID, Module_Name, M_Duration, M_Credits, M_Description, Student_ID*)

iii. **Student, Module and Teacher Table for 1NF:**

- **Student_Module_Teacher1: (**Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email, Student_ID*, Module_ID***)**

iv. **Student, Teacher and Announcement Table for 1NF:**

- **Student_Teacher_Announcement1: (**Announcement_ID, Date_Posted, Ann_Title, Ann_Content, Student_ID*, Module_ID*, Teacher_ID***)**

v. **Resource, Module and Student Table for 1NF:**

- **Resource_Module_Student1: (**Resource_ID, R_Duration, R_Title, R_Type, Sequence_Order, Student_ID*, Module_ID***)**

vi. **Student, Module and Assessment Table for 1NF:**

- **Assessment_Module_Student1: (**Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline, Asse_Status, Marks_Obtained, Grade_Obtained, Student_ID*, Module_ID*)

7.3. Second Normalization Form (2NF):

Rules for 2NF:

- If it is necessary, we need to split the composite keys into separate tables.
- All non-keys should depend on all primary keys.

Transiting from 1NF to 2NF:

- Identifying and removing many-to-many relationships among entities by creating linking entities or adding a bridging entity if needed.

- Attributes depending on the composite key partially are transferred to other tables.

Process for Conversion of Second Normal Form:

i. Student Table for 2NF:

- **Student2:** (Student_ID, Student_Name, D.O.B, Enrollment_Date, Std_Email, Program_ID, Program_Name, P_Description, P_Duration)

Note: 1NF table of Student is already in 2NF because it has only one primary key i.e. Student_ID and it does not have any composite key.

ii. Module_Student Table for 2NF:

- Module_ID ⟹ Module_Name (partially dependent on Module_ID)
- Module_ID ⟹ M_Description (It has partial dependency with Module_ID)
- Module_ID ⟹ M_Credit (It has partial dependency with Module_ID)
- Module_ID, Student_ID ⟶ X
- Student_ID ⟶ X
  - **Module2: (**Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
  - **Module_Student2: (**Module_ID, Student_ID**)**

iii. Student_Module_Teacher Table for 2NF:

- Teacher_ID ⟹ Teacher_Name (partially dependent on Teacher_ID)
- Teacher_ID ⟹ Specialization (It has partial dependency with Teacher_ID)
- Teacher_ID ⟹ Teach_Email (It has partial dependency with Teacher_ID)
- Module_ID, Teacher_ID ⟶ X
- Module_ID ⟶ X
  - **Teacher2: (**Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email**)**
  - **Student_Module_Teacher2: (**Teacher_ID, Module_ID, Student_ID**)**

iv. Student_Module_Announcement Table for 2NF:

- Announcement_ID ⟹ Ann_Title (partially dependent on Announcement_ID)
- Announcement_ID ⟹ Date_Posted (Partial dependency with Announcement_ID)
- Announcement_ID ⟹ Ann_Content (Partial dependency with Announcement_ID)
- Student_ID ⟶ X
- Module_ID ⟶ X
- Teacher_ID, Module_ID ⟶ X
- Teacher_ID, Module_ID, Announcement_ID ⟶ X
  - **Announcement2: (**Announcement_ID, Ann_Title, Date_Posted, Ann_Content**)**
  - **Student_Teacher_Announcement2: (**Teacher_ID, Announcement _ID, Module_ID, Student_ID**)**

v. Resource_Module_Student Table for 2NF:

- Resource_ID ⟹ R_Duration (partially dependent on Resource _ID)
- Resource_ID ⟹ R_Title (It has partial dependency with Resource _ID)
- Resource_ID ⟹ R_Type (It has partial dependency with Resource _ID)
- Resource_ID ⟹ Sequence_Order (It has partial dependency with Resource _ID)
- Student_ID ⟶ X
- Module_ID ⟶ X
- Resource_ID, Student_ID, Module_ID ⟶ X
  - **Resource2: (**Resource_ID, R_Duration, R_Title, R_Type, Sequence_Order**)**
  - **Student_Module_Resource2: (**Resource _ID, Module_ID, Student_ID**)**

vi. Assessment_Module_Student Table for 2NF:

- Assessment_ID, Module_ID, Student_ID ⟹ Grade_Obtained (Fully dependent)
- Assessment_ID, Module_ID, Student_ID ⟹ Marks_Obtained (Fully dependent)
- Assessment_ID, Module_ID, Student_ID ⟹ Asse_Status (Fully dependent)
- Assessment_ID ⟹ Asse_Weightage (It has partial dependency with Assessment _ID)
- Assessment_ID ⟹ Asse_Description (It has partial dependency with Assessment _ID)
- Assessment_ID ⟹ Asse_Deadline (It has partial dependency with Assessment _ID)
- Assessment_ID ⟹ Date_Published (It has partial dependency with Result_ID)
- Module_ID, Student_ID ⟶ X
- Student_ID ⟶ X
- Module_ID ⟶ X
  - **Assessment2: (**Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
  - **Assessment_Module_Student2: (**Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)

Final Second Normal Form Tables:

1. **Student2:** (Student_ID, Student_Name, D.O.B, Enrollment_Date, Std_Email, Program_ID, Program_Name, P_Description, P_Duration)
2. **Module2:** (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
3. **Module_Student2:** (Module_ID, Student_ID)
4. **Teacher2:** (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
5. **Student_Module_Teacher2:** (Teacher_ID, Module_ID, Student_ID)
6. **Announcement2:** (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
7. **Student_Teacher_Announcement2:** (Teacher_ID, Announcement _ID, Module_ID, Student_ID)
8. **Resources2:** (Resource_ID, R_Duration, R_Title, R_Type, Sequence_Order)
9. **Resource_Module_Student 2:** (Resource _ID, Module_ID, Student_ID)
10. **Assessment2:** (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
11. **Assessment _Module_Student2:** (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)

7.4. Third Normalization Form (3NF):

Rules for 3NF:

- Removal of transitive dependency and non-key attributes must not depend upon other non-key attributes.
- Ensure that each non-key attribute depends upon the primary key only (Chris, 2022).

Transformation from 2NF to 3NF:

- Getting rid of transitive dependencies, like when Program_Name depends on Student_ID indirectly through Program_ID.
- Make distinct relations for dependencies or derived attributes.

Process for conversion of Third Normal Form:

1. **Student Table for 3NF:**

In the 2NF table of Student, Program_Name, P_Duration and P_Description depends directly on the Program_ID, which depends upon Student_ID:

Student_ID ⟹ Program_ID ⟹ Program_Name, P_Duration and P_Description

To solve this issue, the Student Table and the Program Table are separated into different tables. Therefore, in 3NF the structure will look like:

- **Student3**: (Student_ID, Student_Name, D.O.B, Enrollment_Date, Std_Email, Program_ID*)
- **Program3:** (Program_ID, P_Description, P_Duration, Program_Name)

2. **Module Table for 3NF:**

Transitivity dependency is not here because all non-key attributes depend directly upon Module_ID.

- **Module3**: **(**Module_ID, Module_Name, M_Duration, M_Credits, M_Description)

3. **Module_Student:**

   It is automatically in 3NF because non key attributes do not exist.

   - **Module_Student3**: **(**Student_ID, Module_ID**)**

4. **Teacher:**

   There is no transitivity dependency because all non-key attributes depend directly upon Teacher_ID.

   - **Teacher3: (**Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email**)**

5. **Student_Module_Teacher:**

   It is automatically in 3NF because non key attributes do not exist.

   - **Student_Module_Teacher3: (**Teacher_ID, Module_ID, Student_ID**)**

6. **Announcement:**

   Transitivity dependency is not here because all non-key attributes depend directly upon Announcement_ID.

   - **Announcement3: (**Announcement_ID, Ann_Title, Date_Posted, Ann_Content**)**

7. **Student_Teacher_Announcement:**

   It is automatically in 3NF because non key attributes do not exist.

   - **Student_Teacher_Announcement3: (**Announcement _ID, Module_ID, Student_ID, Teacher_ID**)**

8. **Resources:**

   Transitivity dependency is not here because all non-key attributes depend directly upon Resource_ID.

   - **Resource3: (**Resource_ID, R_Duration, R_Title, R_Type, Sequence_Order**)**

9. **Student_Module_Resource:**

   It is automatically in 3NF because non key attributes do not exist.

   - **Student_Module_Resource3**: **(**Resource _ID, Module_ID, Student_ID**)**

10. **Assessment:**

    Transitivity dependency is not here because all non-key attributes depend directly upon Assessment_ID.

    - **Assessment3: (**Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)

11. **Assessment_Module_Student:**

    It is automatically in 3NF because non key attributes do not exist.

    - **Assessment_Module_Student3: (**Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained))

Final Third Normal Form Tables:

1. **Student3:** (Student_ID, Student_Name, D.O.B, Enrollment_Date, Std_Email, Program_ID*)
2. **Program3:** (Program_ID, P_Description, P_Duration, Program_Name)
3. **Module3:** (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
4. **Module_Student3**: (Student_ID, Module_ID)
5. **Teacher3:** (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
6. **Student_Module_Teacher3:** (Teacher_ID, Module_ID, Student_ID)
7. **Announcement3**: (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
8. **Student_Teacher_Announcement3:** (Announcement _ID, Module_ID, Student_ID, Teacher_ID)
9. **Resources3:** (Resource_ID, R_Duration, R_Title, R_Type)
10. **Student_Module_Resource3:** (Resource _ID, Module_ID, Student_ID)
11. **Assessment3:** (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
12. **Assessment_Module_Student3:** (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained))

8. Final ERD:

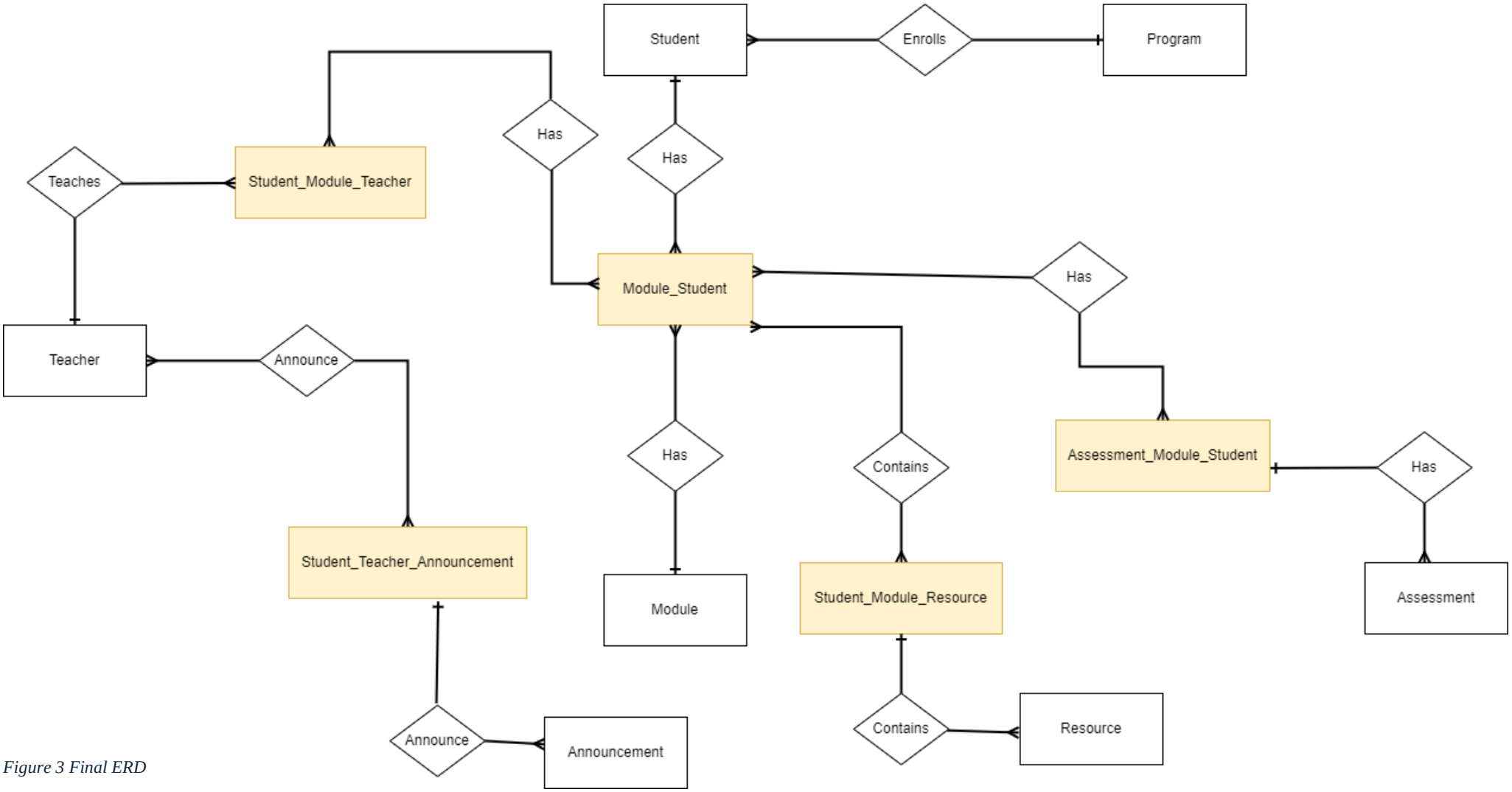

Figure 3 Final ERD

9. Data Dictionary:

Student Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Student_ID	Number	10	Primary Key
2	Student_Name	Character	40	Not Null
3	DOB	Date	-	Not Null
4	Enrollment_Date	Date	-	Not Null
5	Std_Email	Character	30	Unique
6	Program_ID	Number	10	Foreign Key

Table 4 Data Dictionary of Student

Program Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Program_ID	Number	10	Primary Key
2	Program_Name	Character	40	Not Null
3	P_Duration	Date	-	Not Null
4	P_Description	Character	100	Not Null

Table 5 Data Dictionary of Table

Module Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Module_ID	Number	10	Primary Key
2	Module_Name	Character	40	Not Null
3	M_Duration	Date	-	Not Null
4	M_Credits	Number	5	Not Null
5	M_Description	Character	100	Not Null

Table 6 Data Dictionary of Module

Module_Student:

S.No.	Attribute Name	Data Type	Size	Constraint	Composite Constraint
1	Student_ID	Number	10	Foreign Key	Primary Key
2	Module_ID	Number	10	Foreign Key	

Table 7 Data Dictionary of Module_Student

Teacher Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Teacher_ID	Number	10	Primary Key
2	Teacher_Name	Character	40	Not Null
3	Teach_Specialization	Character	50	Not Null
4	Teach_Email	Character	30	Unique

*Table 8 Data Dictionary of **Teacher***

Student_Module_Teacher:

S.No.	Attribute Name	Data Type	Size	Constraint	Composite Constraint
1	Teacher_ID	Number	10	Foreign Key	Primary Key
2	Module_ID	Number	10	Foreign Key	
3	Student_ID	Number	10	Foreign Key	

*Table 9 Data Dictionary of **Student_Module_Teacher***

Announcement Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Announcement_ID	Number	10	Primary Key
2	Date_Posted	Date	-	Not Null
3	Ann_Title	Character	20	Not Null
4	Ann_Content	Character	100	Not Null

*Table 10 Data Dictionary of **Announcement***

Student_Teacher_Announcement:

S.No.	Attribute Name	Data Type	Size	Constraint	Composite Constraint
1	Announcement_ID	Number	10	Foreign Key	Primary Key
2	Module_ID	Number	10	Foreign Key	
3	Student_ID	Number	10	Foreign Key	

*Table 11 Data Dictionary of **Student_Teacher_Announcement***

Resources Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Resource_ID	Number	10	Primary Key
2	R_Duration	Date	-	Not Null
3	R_Title	Character	20	Not Null
4	R_Type	Character	30	Not Null
5	Sequence_Order	Character	30	Not Null

*Table 12 Data Dictionary of **Resource***

Student_Module_Resource:

S.No.	Attribute Name	Data Type	Size	Constraint	Composite Constraint
1	Resource_ID	Number	10	Foreign Key	Primary Key
2	Module_ID	Number	10	Foreign Key	
3	Student_ID	Number	10	Foreign Key	

*Table 13 Data Dictionary of **Student_Module_Resource***

Assessment Table:

S.No.	Attribute Name	Data Type	Size	Constraint
1	Assessment_ID	Number	10	Primary Key
2	Asse_Description	Character	40	Not Null
3	Asse_Weightage	Number	3	Not Null

*Table 14 Data Dictionary of **Assessment***

Assessment_Module_Student:

S.No.	Attribute Name	Data Type	Size	Constraint	Composite Constraint
1	Assessment_ID	Number	10	Foreign Key	Primary Key
2	Module_ID	Number	10	Foreign Key	
3	Student_ID	Number	10	Foreign Key	
4	Asse_Status	Character	10	Not Null	
5	Marks_Obtained	Number	3	Not Null	
6	Grade_Obtained	Character	1	Not Null	

*Table 15 Data Dictionary of **Assessment_Module_Student***

10.Implementation for creating tables:

10.1. Creating Student Table:

```
SQL> CREATE TABLE Student (
  2      Student_ID INT PRIMARY KEY,
  3      Program_ID INT,
  4      Student_Name VARCHAR(40) NOT NULL,
  5      D_O_B DATE NOT NULL,
  6      Enrollment_Date DATE NOT NULL,
  7      Std_Email VARCHAR(40) NOT NULL,
  8      FOREIGN KEY (Program_ID) REFERENCES Program(Program_ID)
  9  );

Table created.
```

Figure 4 Creating Student Table

10.2. Creating Program Table:

```
Enter user-name: sujallparajuli
Enter password:

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

SQL> CREATE TABLE Program (
  2      Program_ID INT PRIMARY KEY,
  3      P_Description VARCHAR(100) NOT NULL,
  4      P_Duration INT NOT NULL,
  5      Program_Name VARCHAR(40) NOT NULL
  6  );
```

Figure 5 Creating Program Table

10.3. Creating Module Table:

```
SQL> CREATE TABLE Module (
  2      Module_ID INT PRIMARY KEY,
  3      Module_Name VARCHAR(50) NOT NULL,
  4      M_Duration INT NOT NULL,
  5      M_Credits INT NOT NULL,
  6      M_Description VARCHAR(100) NOT NULL
  7  );

Table created.
```

Figure 6 Creating Module table

10.4. Creating Module_Student Table:

```
SQL> CREATE TABLE Module_Student (
  2      Student_ID INT,
  3      Module_ID INT,
  4      PRIMARY KEY (Student_ID, Module_ID),
  5      FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID),
  6      FOREIGN KEY (Module_ID) REFERENCES Module(Module_ID)
  7  );

Table created.
```

Figure 7 Creating Module_Student Table

10.5. Creating Teacher Table:

```
SQL> CREATE TABLE Teacher (
  2      Teacher_ID INT PRIMARY KEY,
  3      Teacher_Name VARCHAR(40) NOT NULL,
  4      Teach_Specialization VARCHAR(30) NOT NULL,
  5      Teach_Email VARCHAR(30) UNIQUE NOT NULL
  6  );

Table created.
```

Figure 8 Creating Teacher Table

10.6. Creating Student_Module_Teacher Table:

```
SQL> CREATE TABLE Student_Module_Teacher (
  2      Teacher_ID INT,
  3      Module_ID INT,
  4      Student_ID INT,
  5      PRIMARY KEY (Teacher_ID, Module_ID, Student_ID),
  6      FOREIGN KEY (Teacher_ID) REFERENCES Teacher(Teacher_ID),
  7      FOREIGN KEY (Module_ID) REFERENCES Module(Module_ID),
  8      FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
  9  );

Table created.
```

Figure 9 Creating Student_Teacher_Announcement Table

10.7. Creating Assessment Table:

```
SQL> CREATE TABLE Assessment (
  2       Assessment_ID NUMBER(10) PRIMARY KEY,
  3       Asse_Description VARCHAR2(100) NOT NULL,
  4       Asse_Weightage VARCHAR2(50) NOT NULL,
  5       Asse_Deadline DATE NOT NULL
  6  );

Table created.
```

Figure 10 Creating Assessment Table:

10.8. Creating Assessment_Module_Student Table:

```
SQL> CREATE TABLE Assessment_Module_Student (
  2       Assessment_ID NUMBER(10) NOT NULL,
  3       Module_ID NUMBER(10) NOT NULL,
  4       Student_ID NUMBER(10) NOT NULL,
  5       Asse_Status VARCHAR2(15) NOT NULL,
  6       Marks_Obtained NUMBER(3) NOT NULL,
  7       Grade_Obtained VARCHAR2(1) NOT NULL,
  8       CONSTRAINT PK_Assessment_Module_Student PRIMARY KEY (Assessment_ID, Module_ID, Student_ID),
  9       CONSTRAINT FK_Assessment FOREIGN KEY (Assessment_ID) REFERENCES Assessment(Assessment_ID),

 10       CONSTRAINT FK_Module FOREIGN KEY (Module_ID) REFERENCES Module(Module_ID),
 11       CONSTRAINT FK_Student FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
 12  );

Table created.
```

Figure 11 Creating Assessment_Module_Student Table

10.9. Creating Announcement Table:

```
SQL> CREATE TABLE Announcement (
  2      Announcement_ID INT PRIMARY KEY,
  3      Ann_Title VARCHAR(20) NOT NULL,
  4      Date_Posted DATE NOT NULL,
  5      Ann_Content VARCHAR(100) NOT NULL
  6  );

Table created.
```

Figure 12 Creating Announcement Table:

10.10. Creating Student_Teacher_Announcement Table:

```
SQL> CREATE TABLE Student_Teacher_Announcement (
  2      Announcement_ID INT,
  3      Module_ID INT,
  4      Student_ID INT,
  5      Teacher_ID INT,
  6      PRIMARY KEY (Announcement_ID, Module_ID, Student_ID, Teacher_ID),
  7      FOREIGN KEY (Announcement_ID) REFERENCES Announcement(Announcement_ID),
  8      FOREIGN KEY (Module_ID) REFERENCES Module(Module_ID),
  9      FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID),
 10      FOREIGN KEY (Teacher_ID) REFERENCES Teacher(Teacher_ID)
 11  );

Table created.
```

Figure 13 Creating Student_Teacher_Announcement Table

10.11. Creating Resources Table:

```
SQL> CREATE TABLE Resources (
  2      Resource_ID INT PRIMARY KEY,
  3      R_Duration INT NOT NULL,
  4      R_Title VARCHAR(20) NOT NULL,
  5      R_Type VARCHAR(30) NOT NULL
  6  );

Table created.
```

Figure 14 Creating Resources Table:

10.12. Creating Student_Module_Resource Table:

```
SQL> CREATE TABLE Student_Module_Resource (
  2      Resource_ID INT,
  3      Module_ID INT,
  4      Student_ID INT,
  5      PRIMARY KEY (Resource_ID, Module_ID, Student_ID),
  6      FOREIGN KEY (Resource_ID) REFERENCES Resources(Resource_ID),
  7      FOREIGN KEY (Module_ID) REFERENCES Module(Module_ID),
  8      FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
  9  );

Table created.
```

Figure 15 Creating Student_Module_Resource

10.13. Creating Showing final table after creation:

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ANNOUNCEMENT                   TABLE
ASSESSMENT                     TABLE
ASSESSMENT_MODULE_STUDENT      TABLE
MODULE                         TABLE
MODULE_STUDENT                 TABLE
PROGRAM                        TABLE
RESOURCES                      TABLE
STUDENT                        TABLE
STUDENT_MODULE_RESOURCE        TABLE
STUDENT_MODULE_TEACHER         TABLE
STUDENT_TEACHER_ANNOUNCEMENT   TABLE

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
TEACHER                        TABLE

12 rows selected.
```

Figure 16 Creating Showing final table after creation:

11.Implementation for inserting data in the tables:

11.1. Inserting Program data:

```
SQL> INSERT ALL
  2  INTO Program (Program_ID, Program_Name, P_Duration, P_Description)
  3  VALUES (101, 'BSC Hons Computing', 3, 'Bachelor of Science Honours in Computing')
  4  INTO Program (Program_ID, Program_Name, P_Duration, P_Description)
  5  VALUES (102, 'BSC Hons Networking', 3, 'Bachelor of Science Honours in Networking')
  6  INTO Program (Program_ID, Program_Name, P_Duration, P_Description)
  7  VALUES (103, 'BSC Hons AI', 3, 'Bachelor of Science Honours in Artificial Intelligence')
  8  INTO Program (Program_ID, Program_Name, P_Duration, P_Description)
  9  VALUES (104, 'BSC Hons Multimedia', 3, 'Bachelor of Science Honours in Multimedia')
 10  INTO Program (Program_ID, Program_Name, P_Duration, P_Description)
 11  VALUES (105, 'BSC Hons Data Science', 3, 'Bachelor of Science Honours in Data Science')
 12  INTO Program (Program_ID, Program_Name, P_Duration, P_Description)
 13  VALUES (106, 'BSC Hons Cybersecurity', 3, 'Bachelor of Science Honours in Cybersecurity')
 14  INTO Program (Program_ID, Program_Name, P_Duration, P_Description)
 15  VALUES (107, 'BSC Hons Software Engineering', 3, 'Bachelor of Science Honours in Software
Engineering')
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 17Inserting Program data:

```
SQL> select * from program;

PROGRAM_ID P_DESCRIPTION                                                                                P_DURATION PROGRAM_NAME
---------- ---------------------------------------------------------------------------------------------- ---------- ----------------------------------------
       101 Bachelor of Science Honours in Computing                                                            3 BSC Hons Computing
       102 Bachelor of Science Honours in Networking                                                           3 BSC Hons Networking
       103 Bachelor of Science Honours in Artificial Intelligence                                              3 BSC Hons AI
       104 Bachelor of Science Honours in Multimedia                                                           3 BSC Hons Multimedia
       105 Bachelor of Science Honours in Data Science                                                         3 BSC Hons Data Science
       106 Bachelor of Science Honours in Cybersecurity                                                        3 BSC Hons Cybersecurity
       107 Bachelor of Science Honours in Software Engineering                                                 3 BSC Hons Software Engineering

7 rows selected.
```

11.2. Inserting Student data:

```
SQL> INSERT ALL
  2  INTO Student (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID)
  3  VALUES (201, 'Aarav Sharma', TO_DATE('2004-01-15', 'YYYY-MM-DD'), TO_DATE('2023-08-01', 'YYYY-MM-DD'), 'aarav.sharma@gmail.com', 101)
  4  INTO Student (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID)
  5  VALUES (202, 'Prisha Poudel', TO_DATE('2004-02-16', 'YYYY-MM-DD'), TO_DATE('2023-08-02', 'YYYY-MM-DD'), 'prisha.poudel@gmail.com', 101)
  6  INTO Student (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID)
  7  VALUES (203, 'Arjun Adhikari', TO_DATE('2004-03-17', 'YYYY-MM-DD'), TO_DATE('2023-08-03', 'YYYY-MM-DD'), 'arjun.adhikari@gmail.com', 102)
  8  INTO Student (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID)
  9  VALUES (204, 'Siddhartha Bhattarai', TO_DATE('2004-04-18', 'YYYY-MM-DD'), TO_DATE('2023-08-04', 'YYYY-MM-DD'), 'siddhartha.b@gmail.com', 102)
 10  INTO Student (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID)
 11  VALUES (205, 'Aanya Thapa', TO_DATE('2004-05-19', 'YYYY-MM-DD'), TO_DATE('2023-08-01', 'YYYY-MM-DD'), 'aanya.thapa@gmail.com', 103)
 12  INTO Student (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID)
 13  VALUES (206, 'Rohan KC', TO_DATE('2004-06-20', 'YYYY-MM-DD'), TO_DATE('2023-08-04', 'YYYY-MM-DD'), 'rohan.kc@gmail.com', 103)
 14  INTO Student (Student_ID, Student_Name, DOB, Enrollment_Date, Std_Email, Program_ID)
 15  VALUES (207, 'Krisha Gurung', TO_DATE('2004-07-21', 'YYYY-MM-DD'), TO_DATE('2023-08-05', 'YYYY-MM-DD'), 'krisha.gurung@gmail.com', 104)
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 18 Inserting Student data:

Updated Insertion:

```
SQL> UPDATE student SET PROGRAM_ID = 101 WHERE STUDENT_ID = 201;

1 row updated.

SQL> UPDATE student SET PROGRAM_ID = 101 WHERE STUDENT_ID = 202;

1 row updated.

SQL> UPDATE student SET PROGRAM_ID = 102 WHERE STUDENT_ID = 203;

1 row updated.

SQL> UPDATE student SET PROGRAM_ID = 102 WHERE STUDENT_ID = 204;

1 row updated.

SQL> UPDATE student SET PROGRAM_ID = 103 WHERE STUDENT_ID = 205;

1 row updated.

SQL> UPDATE student SET PROGRAM_ID = 103 WHERE STUDENT_ID = 206;

1 row updated.

SQL> UPDATE student SET PROGRAM_ID = 104 WHERE STUDENT_ID = 207;

1 row updated.
```

```
SQL> select * from student;

STUDENT_ID STUDENT_NAME                                DOB       ENROLLMEN STD_EMAIL                      PROGRAM_ID
---------- ---------------------------------------- --------- --------- ------------------------------ ----------
       201 Aarav Sharma                               15-JAN-04 01-AUG-23 aarav.sharma@gmail.com                101
       202 Prisha Poudel                              16-FEB-04 02-AUG-23 prisha.poudel@gmail.com               102
       203 Arjun Adhikari                             17-MAR-04 03-AUG-23 arjun.adhikari@gmail.com              103
       204 Siddhartha Bhattarai                       18-APR-04 04-AUG-23 siddhartha.b@gmail.com                104
       205 Aanya Thapa                                19-MAY-04 01-AUG-23 aanya.thapa@gmail.com                 105
       206 Rohan KC                                   20-JUN-04 04-AUG-23 rohan.kc@gmail.com                    106
       207 Krisha Gurung                              21-JUL-04 05-AUG-23 krisha.gurung@gmail.com               107

7 rows selected.
```

11.3. Inserting Module data:

```
SQL> INSERT ALL
  2  INTO Module (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
  3  VALUES (301, 'Database Systems', 6, 12, 'Advanced Database Management Systems')
  4  INTO Module (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
  5  VALUES (302, 'Data Structures', 6, 12, 'Advanced Data Structures and Algorithms')
  6  INTO Module (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
  7  VALUES (303, 'Software Engineering', 6, 12, 'Software Development Life Cycle')
  8  INTO Module (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
  9  VALUES (304, 'Cloud Computing', 6, 12, 'Cloud Infrastructure and Services')
 10  INTO Module (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
 11  VALUES (305, 'Deep Learning', 6, 12, 'Neural Networks and Deep Learning')
 12  INTO Module (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
 13  VALUES (306, 'Web Development', 6, 12, 'Full Stack Web Development')
 14  INTO Module (Module_ID, Module_Name, M_Duration, M_Credits, M_Description)
 15  VALUES (307, 'Network Security', 6, 12, 'Network Security and Cryptography')
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 19 Inserting Module

```
SQL> select * from module;

 MODULE_ID MODULE_NAME                                M_DURATION  M_CREDITS M_DESCRIPTION
---------- ---------------------------------------- ---------- ---------- ----------------------------------------
       301 Database Systems                                  6         12 Advanced Database Management Systems
       302 Data Structures                                   6         12 Advanced Data Structures and Algorithms
       303 Software Engineering                              6         12 Software Development Life Cycle
       304 Cloud Computing                                   6         12 Cloud Infrastructure and Services
       305 Deep Learning                                     6         12 Neural Networks and Deep Learning
       306 Web Development                                   6         12 Full Stack Web Development
       307 Network Security                                  6         12 Network Security and Cryptography

7 rows selected.
```

11.4. Inserting Module_Student data:

```
SQL> INSERT ALL
  2  INTO Module_Student (Student_ID, Module_ID)
  3  VALUES (201, 301)
  4  INTO Module_Student (Student_ID, Module_ID)
  5  VALUES (201, 302)
  6  INTO Module_Student (Student_ID, Module_ID)
  7  VALUES (202, 301)
  8  INTO Module_Student (Student_ID, Module_ID)
  9  VALUES (203, 303)
 10  INTO Module_Student (Student_ID, Module_ID)
 11  VALUES (204, 304)
 12  INTO Module_Student (Student_ID, Module_ID)
 13  VALUES (205, 305)
 14  INTO Module_Student (Student_ID, Module_ID)
 15  VALUES (206, 305)
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 20 Inserting Module_Student data:

```
SQL> select * from module_student;

STUDENT_ID  MODULE_ID
---------- ----------
       201        301
       201        302
       202        301
       203        303
       204        304
       205        305
       206        305

7 rows selected.
```

11.5. Inserting Resources data:

```
SQL> INSERT ALL
  2  INTO Resources (Resource_ID, R_Duration, R_Title, R_Type)
  3  VALUES (701, 60, 'DB Fundamentals', 'PDF')
  4  INTO Resources (Resource_ID, R_Duration, R_Title, R_Type)
  5  VALUES (702, 45, 'Advanced SQL', 'PDF')
  6  INTO Resources (Resource_ID, R_Duration, R_Title, R_Type)
  7  VALUES (703, 90, 'Database Design', 'VIDEO')
  8  INTO Resources (Resource_ID, R_Duration, R_Title, R_Type)
  9  VALUES (704, 60, 'Deep Learning', 'PDF')
 10  INTO Resources (Resource_ID, R_Duration, R_Title, R_Type)
 11  VALUES (705, 120, 'Network Security', 'VIDEO')
 12  INTO Resources (Resource_ID, R_Duration, R_Title, R_Type)
 13  VALUES (706, 45, 'Web Dev Basics', 'PDF')
 14  INTO Resources (Resource_ID, R_Duration, R_Title, R_Type)
 15  VALUES (707, 90, 'Cloud Computing', 'VIDEO')
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 21 Inserting Resources data:

```
SQL> select * from resources;

RESOURCE_ID R_DURATION R_TITLE              R_TYPE
----------- ---------- -------------------- ------------------------------
        701         60 DB Fundamentals      PDF
        702         45 Advanced SQL         PDF
        703         90 Database Design      VIDEO
        704         60 Deep Learning        PDF
        705        120 Network Security     VIDEO
        706         45 Web Dev Basics       PDF
        707         90 Cloud Computing      VIDEO

7 rows selected.
```

11.6. Inserting Student_Module_Resource data:

```
SQL> INSERT ALL
  2  INTO Student_Module_Resource (Resource_ID, Module_ID, Student_ID)
  3  VALUES (701, 301, 201)
  4  INTO Student_Module_Resource (Resource_ID, Module_ID, Student_ID)
  5  VALUES (702, 301, 201)
  6  INTO Student_Module_Resource (Resource_ID, Module_ID, Student_ID)
  7  VALUES (701, 301, 202)
  8  INTO Student_Module_Resource (Resource_ID, Module_ID, Student_ID)
  9  VALUES (704, 305, 205)
 10  INTO Student_Module_Resource (Resource_ID, Module_ID, Student_ID)
 11  VALUES (704, 305, 206)
 12  INTO Student_Module_Resource (Resource_ID, Module_ID, Student_ID)
 13  VALUES (705, 307, 204)
 14  INTO Student_Module_Resource (Resource_ID, Module_ID, Student_ID)
 15  VALUES (706, 306, 207)
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 22 Inserting Student_Module_Resource data:

```
SQL> select * from Student_Module_Resource;

RESOURCE_ID  MODULE_ID STUDENT_ID
----------- ---------- ----------
        701        301        201
        702        301        201
        701        301        202
        704        305        205
        704        305        206
        705        307        204
        706        306        207

7 rows selected.
```

11.7. Inserting Teacher data:

```
SQL> INSERT ALL
  2  INTO Teacher (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
  3  VALUES (401, 'Dr. Ramesh Prasad', 'Database Systems', 'ramesh.p@college.edu')
  4  INTO Teacher (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
  5  VALUES (402, 'Prof. Sita Devi', 'Software Engineering', 'sita.d@college.edu')
  6  INTO Teacher (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
  7  VALUES (403, 'Dr. Hari Kumar', 'Artificial Intelligence', 'hari.k@college.edu')
  8  INTO Teacher (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
  9  VALUES (404, 'Dr. Maya Singh', 'Web Technologies', 'maya.s@college.edu')
 10  INTO Teacher (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
 11  VALUES (405, 'Prof. Raj Sharma', 'Network Security', 'raj.s@college.edu')
 12  INTO Teacher (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
 13  VALUES (406, 'Dr. Anita Pant', 'Cloud Computing', 'anita.p@college.edu')
 14  INTO Teacher (Teacher_ID, Teacher_Name, Teach_Specialization, Teach_Email)
 15  VALUES (407, 'Prof. Binod Thapa', 'Data Structures', 'binod.t@college.edu')
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 23 Inserting Teacher data:

```
SQL> select * from teacher;

TEACHER_ID TEACHER_NAME                          TEACH_SPECIALIZATION          TEACH_EMAIL
---------- ------------------------------------- ----------------------------- ----------------------------
       401 Dr. Ramesh Prasad                     Database Systems              ramesh.p@college.edu
       402 Prof. Sita Devi                       Software Engineering          sita.d@college.edu
       403 Dr. Hari Kumar                        Artificial Intelligence       hari.k@college.edu
       404 Dr. Maya Singh                        Web Technologies              maya.s@college.edu
       405 Prof. Raj Sharma                      Network Security              raj.s@college.edu
       406 Dr. Anita Pant                        Cloud Computing               anita.p@college.edu
       407 Prof. Binod Thapa                     Data Structures               binod.t@college.edu

7 rows selected.
```

11.8. Inserting Student_Module_Teacher data:

```
SQL> INSERT ALL
  2  INTO Student_Module_Teacher (Teacher_ID, Module_ID, Student_ID)
  3  VALUES (401, 301, 201)
  4  INTO Student_Module_Teacher (Teacher_ID, Module_ID, Student_ID)
  5  VALUES (401, 301, 202)
  6  INTO Student_Module_Teacher (Teacher_ID, Module_ID, Student_ID)
  7  VALUES (402, 303, 203)
  8  INTO Student_Module_Teacher (Teacher_ID, Module_ID, Student_ID)
  9  VALUES (403, 305, 205)
 10  INTO Student_Module_Teacher (Teacher_ID, Module_ID, Student_ID)
 11  VALUES (403, 305, 206)
 12  INTO Student_Module_Teacher (Teacher_ID, Module_ID, Student_ID)
 13  VALUES (404, 306, 207)
 14  INTO Student_Module_Teacher (Teacher_ID, Module_ID, Student_ID)
 15  VALUES (405, 307, 204)
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 24 Inserting Student_Module_Teacher data:

```
SQL> select * from Student_Module_Teacher;

TEACHER_ID  MODULE_ID STUDENT_ID
---------- ---------- ----------
       401        301        201
       401        301        202
       402        303        203
       403        305        205
       403        305        206
       404        306        207
       405        307        204

7 rows selected.
```

11.9. Inserting Announcement data:

```
SQL> INSERT ALL
  2  INTO Announcement (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
  3  VALUES (601, 'DB Mid-Term Update', TO_DATE('2024-05-02', 'YYYY-MM-DD'), 'Mid-term exam postponed to May 20th')
  4  INTO Announcement (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
  5  VALUES (602, 'Project Guidelines', TO_DATE('2024-05-10', 'YYYY-MM-DD'), 'Updated project guidelines posted')
  6  INTO Announcement (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
  7  VALUES (603, 'Deep Learning Wksp', TO_DATE('2024-05-15', 'YYYY-MM-DD'), 'Additional workshop scheduled')
  8  INTO Announcement (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
  9  VALUES (604, 'Network Sec Seminar', TO_DATE('2024-05-20', 'YYYY-MM-DD'), 'Industry expert session')
 10  INTO Announcement (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
 11  VALUES (605, 'Web Dev Hackathon', TO_DATE('2024-05-25', 'YYYY-MM-DD'), '24-hour coding challenge')
 12  INTO Announcement (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
 13  VALUES (606, 'Cloud Comp Workshop', TO_DATE('2024-05-30', 'YYYY-MM-DD'), 'AWS certification prep')
 14  INTO Announcement (Announcement_ID, Ann_Title, Date_Posted, Ann_Content)
 15  VALUES (607, 'Data Struct Tutorial', TO_DATE('2024-06-05', 'YYYY-MM-DD'), 'Extra help session')
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 25 Inserting Announcement data:

```
SQL> select * from announcement;

ANNOUNCEMENT_ID ANN_TITLE           DATE_POST ANN_CONTENT
--------------- -------------------- --------- ------------------------------------
            601 DB Mid-Term Update   02-MAY-24 Mid-term exam postponed to May 20th
            602 Project Guidelines   10-MAY-24 Updated project guidelines posted
            603 Deep Learning Wksp   15-MAY-24 Additional workshop scheduled
            604 Network Sec Seminar  20-MAY-24 Industry expert session
            605 Web Dev Hackathon    25-MAY-24 24-hour coding challenge
            606 Cloud Comp Workshop  30-MAY-24 AWS certification prep
            607 Data Struct Tutorial 05-JUN-24 Extra help session

7 rows selected.
```

11.10. Inserting Student_Teacher_Announcement data:

```
SQL> INSERT ALL
  2  INTO Student_Teacher_Announcement (Announcement_ID, Module_ID, Student_ID, Teacher_ID)
  3  VALUES (601, 301, 201, 401)
  4  INTO Student_Teacher_Announcement (Announcement_ID, Module_ID, Student_ID, Teacher_ID)
  5  VALUES (601, 301, 202, 401)
  6  INTO Student_Teacher_Announcement (Announcement_ID, Module_ID, Student_ID, Teacher_ID)
  7  VALUES (602, 303, 203, 402)
  8  INTO Student_Teacher_Announcement (Announcement_ID, Module_ID, Student_ID, Teacher_ID)
  9  VALUES (603, 305, 205, 403)
 10  INTO Student_Teacher_Announcement (Announcement_ID, Module_ID, Student_ID, Teacher_ID)
 11  VALUES (603, 305, 206, 403)
 12  INTO Student_Teacher_Announcement (Announcement_ID, Module_ID, Student_ID, Teacher_ID)
 13  VALUES (604, 307, 204, 405)
 14  INTO Student_Teacher_Announcement (Announcement_ID, Module_ID, Student_ID, Teacher_ID)
 15  VALUES (605, 306, 207, 404)
 16  SELECT * FROM DUAL;

7 rows created.
```

Figure 26 Inserting Announcement_Module_Student data

```
SQL> select * from Student_Teacher_Announcement;

ANNOUNCEMENT_ID  MODULE_ID STUDENT_ID TEACHER_ID
--------------- ---------- ---------- ----------
            601        301        201        401
            601        301        202        401
            602        303        203        402
            603        305        205        403
            603        305        206        403
            604        307        204        405
            605        306        207        404

7 rows selected.
```

11.11. Inserting Assessment data:

```
SQL> INSERT ALL
  2  INTO Assessment (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
  3  VALUES (10, 'Programming Assignment', '40', TO_DATE('2024-05-10', 'YYYY-MM-DD'))
  4  INTO Assessment (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
  5  VALUES (11, 'Database Project', '40', TO_DATE('2024-06-01', 'YYYY-MM-DD'))
  6  INTO Assessment (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
  7  VALUES (12, 'Professional Ethics Exam', '50', TO_DATE('2024-05-15', 'YYYY-MM-DD'))
  8  INTO Assessment (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
  9  VALUES (13, 'Networking Lab', '70', TO_DATE('2024-06-05', 'YYYY-MM-DD'))
 10  INTO Assessment (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
 11  VALUES (14, 'Multimedia Presentation', '30', TO_DATE('2024-05-25', 'YYYY-MM-DD'))
 12  INTO Assessment (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
 13  VALUES (15, 'Cloud Computing Assignment', '80', TO_DATE('2024-06-10', 'YYYY-MM-DD'))
 14  INTO Assessment (Assessment_ID, Asse_Description, Asse_Weightage, Asse_Deadline)
 15  VALUES (16, 'Final Exam for Computing', '100', TO_DATE('2024-06-15', 'YYYY-MM-DD'))
 16  SELECT * FROM dual;

7 rows created.
```

Figure 27 Inserting Assessment data:

```
SQL> SELECT * FROM ASSESSMENT;

ASSESSMENT_ID ASSE_DESCRIPTION                                                                          ASSE_WEIGHTAGE                                   ASSE_DEAD
------------- ----------------------------------------------------------------------------------------- ---------------------------------------------- ---------
           10 Programming Assignment                                                                   40                                               10-MAY-24
           11 Database Project                                                                         40                                               01-JUN-24
           12 Professional Ethics Exam                                                                 50                                               15-MAY-24
           13 Networking Lab                                                                           70                                               05-JUN-24
           14 Multimedia Presentation                                                                  30                                               25-MAY-24
           15 Cloud Computing Assignment                                                               80                                               10-JUN-24
           16 Final Exam for Computing                                                                 100                                              15-JUN-24

7 rows selected.
```

11.12. Inserting Assessment_Module_Student data:

```
SQL> INSERT ALL
  2  INTO Assessment_Module_Student (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)
  3  VALUES (10, 301, 201, 'Submitted', 85, 'A')
  4  INTO Assessment_Module_Student (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)
  5  VALUES (10, 301, 202, 'Submitted', 78, 'B')
  6  INTO Assessment_Module_Student (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)
  7  VALUES (11, 302, 201, 'Submitted', 92, 'A')
  8  INTO Assessment_Module_Student (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)
  9  VALUES (11, 302, 202, 'Submitted', 84, 'B')
 10  INTO Assessment_Module_Student (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)
 11  VALUES (12, 303, 201, 'Not Submitted', 0, 'E')
 12  INTO Assessment_Module_Student (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)
 13  VALUES (12, 303, 202, 'Not Submitted', 0, 'E')
 14  INTO Assessment_Module_Student (Assessment_ID, Module_ID, Student_ID, Asse_Status, Marks_Obtained, Grade_Obtained)
 15  VALUES (13, 304, 201, 'Not Submitted', 0, 'E')
 16  SELECT * FROM dual;

7 rows created.
```

Figure 28 Inserting Assessment_Module_Student data:

```
SQL> SELECT * FROM ASSESSMENT_Module_Student;

ASSESSMENT_ID  MODULE_ID STUDENT_ID ASSE_STATUS     MARKS_OBTAINED G
------------- ---------- ---------- --------------- -------------- -
           10        301        201 Submitted                  85 A
           10        301        202 Submitted                  78 B
           11        302        201 Submitted                  92 A
           11        302        202 Submitted                  84 B
           12        303        201 Not Submitted               0 E
           12        303        202 Not Submitted               0 E
           13        304        201 Not Submitted               0 E

7 rows selected.
```

12. Query regarding database:

1. Information Query:

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

```
SQL> Select P.Program_Name, Count(S.Student_ID) As Total
  2  From Program P
  3  Left Join Student S
  4  On P.Program_ID = S.Program_ID
  5  Group by P.Program_Name;

PROGRAM_NAME                                   TOTAL
---------------------------------------- ----------
BSC Hons Data Science                             1
BSC Hons Cybersecurity                            1
BSC Hons Software Engineering                     1
BSC Hons Networking                               1
BSC Hons AI                                       1
BSC Hons Multimedia                               1
BSC Hons Computing                                1

7 rows selected.
```

Figure 29 Query One: listing students enrolled in a program and total program

Entered Query:

Select P.Program_Name, Count(S.Student_ID) As Total

From Program P

Left Join Student S

On P.Program_ID = S.Program_ID

Group by P.Program_Name;

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

```
SQL> SELECT A.Announcement_ID, M.Module_Name, A.Ann_Title, A.Date_Posted as dates
  2  FROM Announcement A
  3  JOIN Student_Teacher_Announcement N ON A.Announcement_ID = N.Announcement_ID
  4  JOIN Module M ON N.Module_ID = M.Module_ID
  5  WHERE A.Date_Posted >= TO_DATE('01-MAY-2024', 'DD-MON-YYYY')
  6  AND A.Date_Posted <= TO_DATE('28-MAY-2024', 'DD-MON-YYYY')
  7  AND M.Module_Name = 'Software Engineering';

ANNOUNCEMENT_ID MODULE_NAME                            ANN_TITLE            DATES
--------------- -------------------------------------- -------------------- ---------
            602 Software Engineering                   Project Guidelines   10-MAY-24
```

Figure 30 Announcement for a particular module i: e Software Engineering

Entered Query:

```
SELECT A.Announcement_ID, M.Module_Name, A.Ann_Title, A.Date_Posted as dates
FROM Announcement A
JOIN Student_Teacher_Announcement N ON A.Announcement_ID = N.Announcement_ID
JOIN Module M ON N.Module_ID = M.Module_ID
WHERE A.Date_Posted >= TO_DATE('01-MAY-2024', 'DD-MON-YYYY')
AND A.Date_Posted <= TO_DATE('28-MAY-2024', 'DD-MON-YYYY')
AND M.Module_Name = 'Software Engineering';
```

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

```
SQL> SELECT A.Module_Name, COUNT(B.Resource_ID) as SUM
  2  From Module A
  3  LEFT JOIN Student_Module_Resource B ON A.Module_ID = B.Module_ID
  4  Where A.Module_Name LIKE 'D%'
  5  Group by A.Module_Name;

MODULE_NAME                                       SUM
------------------------------------ ----------
Data Structures                                     0
Database Systems                                    3
Deep Learning                                       2
```

Figure 31 Module name that starts from D

Entered Query:

```
SELECT A.Module_Name, COUNT(B.Resource_ID) as SUM
From Module A
LEFT JOIN Student_Module_Resource B ON A.Module_ID = B.Module_ID
Where A.Module_Name LIKE 'D%'
Group by A.Module_Name;
```

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

```
SQL>
SQL> SELECT s.Student_ID, s.Student_Name, p.Program_Name,m.Module_Name, ms.Asse_Status AS status
  2  FROM Student s
  3  INNER JOIN Program p ON s.Program_ID = p.Program_ID
  4  INNER JOIN Module_Student sm ON s.Student_ID = sm.Student_ID
  5  INNER JOIN Module m ON sm.Module_ID = m.Module_ID
  6  LEFT JOIN Assessment_Module_Student ms ON s.Student_ID = ms.Student_ID AND m.Module_ID = ms.Module_ID
  7  WHERE ms.Asse_Status = 'Not Submitted' OR ms.Asse_Status IS NULL;

STUDENT_ID STUDENT_NAME                            PROGRAM_NAME                           MODULE_NAME                STATUS
---------- ---------------------------------------- ---------------------------------------- ----------------------------------------
       204 Siddhartha Bhattarai                     BSC Hons Multimedia                    Cloud Computing
       206 Rohan KC                                 BSC Hons Cybersecurity                 Deep Learning
       205 Aanya Thapa                              BSC Hons Data Science                  Deep Learning
       203 Arjun Adhikari                           BSC Hons AI                            Software Engineering
```

Figure 32 All students along with their enrolled program who have not submitted any assessments

Entered Query:

```
SELECT s.Student_ID, s.Student_Name, p.Program_Name,m.Module_Name,
ms.Asse_Status AS status

FROM Student s

INNER JOIN Program p ON s.Program_ID = p.Program_ID

INNER JOIN Module_Student sm ON s.Student_ID = sm.Student_ID

INNER JOIN Module m ON sm.Module_ID = m.Module_ID

LEFT JOIN Assessment_Module_Student ms ON s.Student_ID = ms.Student_ID

AND m.Module_ID = ms.Module_ID

WHERE ms.Asse_Status = 'Not Submitted' OR ms.Asse_Status IS NULL;
```

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

```
SQL> Select T.Teacher_Name, Count(M.Module_ID) As SUM
  2  From Teacher T
  3  join Student_Module_Teacher M ON T.Teacher_ID = M.Teacher_ID
  4  Group by  T.Teacher_Name
  5  HAVING COUNT(M.Module_ID) > 1;

TEACHER_NAME                                       SUM
------------------------------------ ----------
Dr. Ramesh Prasad                                    2
Dr. Hari Kumar                                       2
```

Figure 33 teachers who teach more than one module

Entered Query:

```
Select T.Teacher_Name, Count(M.Module_ID) As SUM From Teacher T
join Student_Module_Teacher M ON T.Teacher_ID = M.Teacher_ID
Group by  T.Teacher_Name
HAVING COUNT(M.Module_ID) > 1;
```

13.Transaction Query:

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

```
SQL> SELECT m.Module_Name, a.Asse_Deadline
  2  FROM Module m
  3  JOIN Assessment_Module_Student ma ON m.Module_ID = ma.Module_ID
  4  JOIN Assessment a ON ma.Assessment_ID = a.Assessment_ID
  5  WHERE a.Asse_Deadline = (SELECT MAX(Asse_Deadline)
  6  FROM Assessment
  7  );

no rows selected
```

Figure 34 module that has the latest assessment deadline

Entered Query:

```
SELECT m.Module_Name, a.Asse_Deadline
FROM Module m
JOIN Assessment_Module_Student ma ON m.Module_ID = ma.Module_ID
JOIN Assessment a ON ma.Assessment_ID = a.Assessment_ID
WHERE a.Asse_Deadline = (SELECT MAX(Asse_Deadline)
FROM Assessment);
```

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

```
SQL> SELECT *
  2  FROM (
  3      SELECT s.Student_Name, SUM(ams.Marks_Obtained) AS Total_Score
  4      FROM Student s
  5      JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
  6      WHERE ams.Marks_Obtained IS NOT NULL
  7      GROUP BY s.Student_Name
  8      ORDER BY Total_Score DESC
  9  )
 10  WHERE ROWNUM <= 3;

STUDENT_NAME                           TOTAL_SCORE
-------------------------------------- -----------
Aarav Sharma                                   177
Prisha Poudel                                  162
```

Figure 35 top three students who have the highest total score across all modules

Entered Query:

```
SELECT *
FROM (
   SELECT s.Student_Name, SUM(ams.Marks_Obtained) AS Total_Score
   FROM Student s
   JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
   WHERE ams.Marks_Obtained IS NOT NULL
   GROUP BY s.Student_Name
   ORDER BY Total_Score DESC
)
WHERE ROWNUM <= 3;
```

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

```
SQL> SELECT p.Program_Name AS Program_Name,
  2         COUNT(a.Assessment_ID) AS Total_Assessments_Count,
  3         AVG(ams.Marks_Obtained) AS Average_Marks_Obtained
  4  FROM Program p
  5  JOIN Student s ON p.Program_ID = s.Program_ID
  6  JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
  7  JOIN Assessment a ON ams.Assessment_ID = a.Assessment_ID
  8  WHERE ams.Marks_Obtained IS NOT NULL
  9  GROUP BY p.Program_Name;

PROGRAM_NAME                           TOTAL_ASSESSMENTS_COUNT AVERAGE_MARKS_OBTAINED
-------------------------------------- ----------------------- ----------------------
BSC Hons Networking                                          3                     54
BSC Hons Computing                                           4                  44.25
```

Figure 36 total number of assessments for each program and the average score across all assessments in those programs

Entered Query:

```
SELECT p.Program_Name AS Program_Name,
    COUNT(a.Assessment_ID) AS Total_Assessments_Count,
    AVG(ams.Marks_Obtained) AS Average_Marks_Obtained
FROM Program p
JOIN Student s ON p.Program_ID = s.Program_ID
JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
JOIN Assessment a ON ams.Assessment_ID = a.Assessment_ID
WHERE ams.Marks_Obtained IS NOT NULL
GROUP BY p.Program_Name;
```

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

```
SQL> SELECT s.Student_Name AS Student_Name, ams.Marks_Obtained AS Marks_Obtained
  2  FROM Student s
  3  JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
  4  JOIN Module m ON ams.Module_ID = m.Module_ID
  5  WHERE m.Module_Name = 'Database Systems'
  6  AND ams.Marks_Obtained > (
  7      SELECT AVG(ams2.Marks_Obtained)
  8      FROM Assessment_Module_Student ams2
  9      JOIN Module m2 ON ams2.Module_ID = m2.Module_ID
 10      WHERE m2.Module_Name = 'Database Systems'
 11  );

STUDENT_NAME                             MARKS_OBTAINED
---------------------------------------- --------------
Aarav Sharma                                         85
```

Figure 37 total number of assessments for each program and the average score across all assessments in those programs

Entered Query:

```
SELECT s.Student_Name AS Student_Name, ams.Marks_Obtained AS Marks_Obtained
FROM Student s
JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
JOIN Module m ON ams.Module_ID = m.Module_ID
WHERE m.Module_Name = 'Database Systems'
AND ams.Marks_Obtained > (
    SELECT AVG(ams2.Marks_Obtained)
    FROM Assessment_Module_Student ams2
    JOIN Module m2 ON ams2.Module_ID = m2.Module_ID
    WHERE m2.Module_Name = 'Database Systems'
);
```

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

```
SQL> SELECT s.Student_Name AS Student_Name, m.Module_Name AS Module_Name,
  2  SUM(ams.Marks_Obtained) AS Total_Marks_Obtained,
  3  CASE
  4    WHEN SUM(ams.Marks_Obtained) >= 40 THEN 'Pass'
  5             ELSE 'Fail'
  6    END AS Pass_Fail_Status
  7  FROM Student s
  8  JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
  9  JOIN Module m ON ams.Module_ID = m.Module_ID
 10  GROUP BY s.Student_Name, m.Module_Name
 11  ORDER BY s.Student_Name;

STUDENT_NAME                           MODULE_NAME                            TOTAL_MARKS_OBTAINED PASS
-------------------------------------- -------------------------------------- -------------------- ----
Aarav Sharma                           Cloud Computing                                          11 Fail
Aarav Sharma                           Data Structures                                          92 Pass
Aarav Sharma                           Database Systems                                         85 Pass
Aarav Sharma                           Software Engineering                                     11 Fail
Prisha Poudel                          Data Structures                                          84 Pass
Prisha Poudel                          Database Systems                                         78 Pass
Prisha Poudel                          Software Engineering                                     11 Fail

7 rows selected.
```

Figure 38 student has passed or failed as remarks as per their total aggregate marks obtained in a particular module

Entered Query:

```
SELECT s.Student_Name AS Student_Name, m.Module_Name AS Module_Name,
SUM(ams.Marks_Obtained) AS Total_Marks_Obtained,
CASE
  WHEN SUM(ams.Marks_Obtained) >= 40 THEN 'Pass'
        ELSE 'Fail'
  END AS Pass_Fail_Status
FROM Student s
JOIN Assessment_Module_Student ams ON s.Student_ID = ams.Student_ID
JOIN Module m ON ams.Module_ID = m.Module_ID
GROUP BY s.Student_Name, m.Module_Name
ORDER BY s.Student_Name;
```

14.Critical Evaluation:

14.1. Overview of Module:

The Database Design and Implementation module is vital for understanding how to organize and manage data within computer systems. It covers fundamental database concepts, including:

- Data organization and storage: How to structure and store data effectively.
- Database creation: Basic principles of building databases.
- Normalization: Organizing data to reduce redundancy and improve efficiency.
- SQL queries: Using SQL to interact with and manipulate data (Ana L.C. Bazzan, n.d.).

This module will be practical experience for the students to build a real-world database, for example, an e-commerce platform. This will show the students how applicable database concepts are in other fields, such as:

- E-learning: Building online learning platforms.
- Healthcare: Patient records and medical data management.
- Finance: Handling financial transactions and data analysis.
- Business: Data-driven decisions and reports.
- Interconnections with Other Subjects

The database module has very strong interconnections with other subjects within IT and business:

- Software Engineering: Efficient code that interfaces with data should be written based on database design.
- Data Structures and Algorithms: Advanced data structures such as trees and graphs are used to optimize data storage in databases.
- Business Management: Database skills will be crucial in making informed business decisions based on the analysis of data.

14.2. Coursework Critical Assessment:

St. Mary's College E-Learning Platform Project

I built a database for an online learning website called St. Mary's College. This website would help students learn through the use of modern technology. Although the project was new and challenging for me, I completed it with the help of my teacher.

The major steps involved in the project were that first, I identified the basic elements of the database, which included student information, courses, and study programs. From these, I created an Entity Relationship Diagram, which is referred to as an ERD. I then decomposed the big data tables into smaller ones by a process called normalization, and information was thus handled easily. Some problems did arise at this stage; I just solved them through an inquiry from my teacher and looking online for a solution. Once the normalization to 3NF was done, the final ERD was produced.

The next major activity was to create the database commands in SQL. I have created various types of commands like CREATE statements to create database tables; INSERT statements for creating students and courses information. Each table needed at least 7 rows of data. COMMIT statements to save all the changes. For the last part of this project, I wrote an evaluation on two main topics: How this module of database relates to other subjects; Detailed review of the coursework.

I faced many challenges, but at the end of it all, I successfully built a working e-learning platform database. This project taught me a lot about database design and management.

15.Screenshot of Dump File:

```
Command Prompt

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

C:\Users\Nitro>cd C:\sujalparajuli

C:\sujalparajuli>exp sujallparajuli/23050262 file = sujalparajuli.dmp

Export: Release 11.2.0.2.0 - Production on Thu Jan 23 00:30:42 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 SUJALLPARAJULI
. exporting PUBLIC type synonyms
. exporting private type synonyms
. exporting object type definitions for user SUJALLPARAJULI
About to export SUJALLPARAJULI's objects ...
. exporting database links
. exporting sequence numbers
. exporting cluster definitions
. about to export SUJALLPARAJULI's tables via Conventional Path ...
. . exporting table                   ANNOUNCEMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                     ASSESSMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table      ASSESSMENT_MODULE_STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
EXP-00091: Exporting questionable statistics.
. . exporting table                         MODULE          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                 MODULE_STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        PROGRAM          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                      RESOURCES          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        STUDENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table        STUDENT_MODULE_RESOURCE          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table         STUDENT_MODULE_TEACHER          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table   STUDENT_TEACHER_ANNOUNCEMENT          7 rows exported
EXP-00091: Exporting questionable statistics.
. . exporting table                        TEACHER          7 rows exported
EXP-00091: Exporting questionable statistics.
. exporting synonyms
. exporting views
. exporting stored procedures
. exporting operators
. exporting referential integrity constraints
. exporting triggers
. exporting indextypes
. exporting bitmap, functional and extensible indexes
. exporting posttables actions
. exporting materialized views
. exporting snapshot logs
. exporting job queues
. exporting refresh groups and children
. exporting dimensions
. exporting post-schema procedural objects and actions
. exporting statistics
Export terminated successfully with warnings.

C:\sujalparajuli>C:\sujalparajuli
```

Figure 39 Screenshot of Dump File

16. Screenshot of Dropping all Tables:

```
SQL*Plus: Release 11.2.0.2.0 Production on Thu Jan 23 00:34:50 2025

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

Enter user-name: sujallparajuli
Enter password:

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

SQL> drop table Assessment_Module_Student;

Table dropped.

SQL> drop table Assessment;

Table dropped.

SQL> drop table Student_Module_Resource;

Table dropped.

SQL> drop table Resource;
drop table Resource
           *
ERROR at line 1:
ORA-00903: invalid table name


SQL> drop table Resources;

Table dropped.

SQL> drop table Student_Teacher_Announcement;

Table dropped.

SQL> drop table Announcement;

Table dropped.

SQL> drop table Student_Module_Teacher;

Table dropped.

SQL> drop table Teacher;

Table dropped.

SQL> drop table Module_Student;

Table dropped.

SQL> drop table Module;

Table dropped.

SQL> drop table Student;

Table dropped.

SQL> drop table Program;

Table dropped.

SQL> |
```

Figure 40 Screenshot of Dropping all Tables

17.References

Ana L.C. Bazzan, M. d. (n.d.). *Database Module*. Retrieved from sciencedirect: https://www.sciencedirect.com/topics/computer-science/database-module

Chris, K. (2022, December 21). *Database Normalization – Normal Forms 1nf 2nf 3nf Table Examples*. Retrieved from FreeCodeCamp: https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/

Geeksofgeeks. (2024, july 23). *Normal form in DBMS*. Retrieved from Geeksofgeeks: https://www.geeksforgeeks.org/normal-forms-in-dbms/