LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

100% Individual Coursework

Autumn 2024

Credit: 15 Semester Long Module

Student Name: Priya Soni

London Met ID: 23048457

Assignment Submission Date: December 30, 2024

Word Count:

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

Document Details

Submission ID
trn:oid:::3618:75023890

Submission Date
Dec 31, 2024, 12:13 AM GMT+5:45

Download Date
Dec 31, 2024, 12:15 AM GMT+5:45

File Name
Database milestone 2.docx

File Size
26.0 KB

35 Pages

2,573 Words

16,732 Characters

17% Overall Similarity

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

Match Groups

- **36** Not Cited or Quoted 16%
  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
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

5% Internet sources
1% Publications
17% 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 figures

Table of Tables

1. Introduction

1.1 Introduction of business and its forte

Islington College was established in 1996 and has since become one of the leading colleges in the field of technology education. It is in Kamal Pokhari, Kathmandu. The tremendous growth of this college in the last decade speaks volumes about its commitment to excellence in the field of technology education. The undergraduate programs include a Bachelor of Science in Computing, Bachelor of Science in Networking, Bachelor of Science in Multimedia, Bachelor of Science in Artificial Intelligence, and Bachelor of Science in Cybersecurity. Each program is designed in collaboration with industry experts, thus assuring its applicability and usefulness. The main course, being the Computing program, has consistently achieved an employment rate of 80% within six months of graduating, with graduates being placed in top companies. What makes Islington College succeed is the founder and chairman, Mr. Slav Budhathoki. His remarkable journey of establishing Islington College is no less impressive than the institution itself.

Figure 1: Representing Islington College

Ms. Marry is an entrepreneur who, after finishing her master's degree in educational technology at Stanford University and an MBA at MIT, worked for 15 years in educational technology innovation. She held key positions at several major EdTech companies, where she spearheaded the development of new learning management systems and online education platforms. MS. Mary's view of education is much broader than just a regular school subject. Ms. Mary envisions education to be one of them, and the number of people who are seeking flexibility in learning will increase. She has thrown her biggest project so far: the complete E-Classroom Platform. The e-classroom platform is an innovative digital learning management system initiated by entrepreneur Ms. Mary to provide both students and teachers with a comprehensive online and digital educational environment. We are aware that throughout the years, "E-classroom platform" has taken place in the educational environment continuously, so Ms. Mary would want to launch online applications for colleges that will specialize in managing students, teachers and programmers. The platform E-classroom aspires to take traditional classrooms to a higher level of professional and friendly interaction between the students and the teachers and also makes them facilitate easy resource sharing along with administrative learning activities in a virtually formed learning space. An area where the platform proves to be strong is how it fulfills its purposes of providing knowledge to learners by restructuring resources and facilitating assessment management. The proposed E-classroom platform, having key features like the following, will :

- Easy management of all embedded programs and modules.
- Effective and efficient mode of resource delivery system
- Methods used for tracking and grading in assessment
- Systems used for tracking and reporting progress
- The ideal broadcasting of information among students and teachers
- Order in learning courses.

1.2 Current business activities and operations

The E-classroom platform manages several interconnected educational operations. The platform manages programs that provide a variety of degrees like BSc in Computing, BSc in networking and BSc in multimedia. It also oversees several modules for a particular program and manages module sharing across programs. It monitors student resources usage and control various resources and their characteristics. This platform is involved in several assessments of each module and deals with the assignment of the due dates together with weight distribution. It produces evaluation reports of the students and keeps alerts related to the module. This platform supports messages from the teachers and the students and make sure that the communication is sent to the designated recipients only. It Maintains module-specific resources and announcements, announcements by teachers, module-based notifications and targeted delivery of communications for teachers and students.

The E_Classroom platform helps in organized learning with the help of a resource management system. This means that educational materials need to be completed in order. Assessment management keeps a close eye on how students are doing. Teachers are given specific modules to work on. They oversee delivering content and evaluating students; they can also post announcements for their modules. This combined system makes a connected online learning environment that manages all educational parts while keeping clear communication between teachers and students.

1.3 Business Rules

- The platform should manage at least one module in every program.
- Students must provide email address and contact number.
- Programs must have a specified duration and description.
- Programs can have multiple modules and one module ca belong to various programs.
- Each student needs to be enrolled in only one program at a time.
- Each student and teachers need to have a unique identification number.
- Each module shall be assigned at least to one teacher.
- Each module should have at least one assessment components.
- Each resource shall be completed in sequence order and should have a unique identifier within its module.
- Resource should specify type and duration.
- Every assessment must be assigned to precisely one module
- Assessment should have specified deadlines.
- Students' performance in each assessment needs to be tracked properly.
- One teacher can be assigned to many modules but only assigned teacher can post announcements for their modules.
- Teachers are responsible for marking assessment in their module.
- Every student's performance in the assessment must be kept on record.
- One program can be enrolled in at any one time.
- Every student should have unique identification.
- The students must complete all the modules that are compulsory in their enrolled program.
- All the programs operate should have their unique identification.
- Every module has an assigned credit value.
- Every program has certain duration for completion.

2. Identification of Entities and Attributes

a. Program

S. No.	Attribute Name	Data	Size	Constraints
1	Program ID	Number	10	Primary key
2	Program Name	Character	40	Not Null
3	Description	Character	100	Not Null
4	Duration	Date	-	Not Null

Table 1: Representing initial Program.

b. Student

S. No.	Attribute Name	Data	Size	Constraints
1	Student ID	Number	30	Primary key
2	Student Name	Character	40	Not Null
3	Contact Number	Number	10	Unique
4	Email Address	Character	40	Unique

Table 2: Representing initial Student.

c. Module

S. No.	Attribute Name	Data	Size	Constraints
1	Module ID	Number	30	Primary key
2	Module Name	Character	40	Not Null
3	Credits	Number	10	Not Null

Table 3: Representing initial Module.

2.1 Assumption for initial ERD

- One program can be enrolled in at any one time.
- Programs can have multiple modules.
- Every student should have unique identification.
- The students must complete all the modules that are compulsory in their enrolled program.
- All the programs operate should have their unique identification.
- Every module has an assigned credit value.
- Every program has certain duration for completion.

2.2 Initial Entity Relationship Diagram

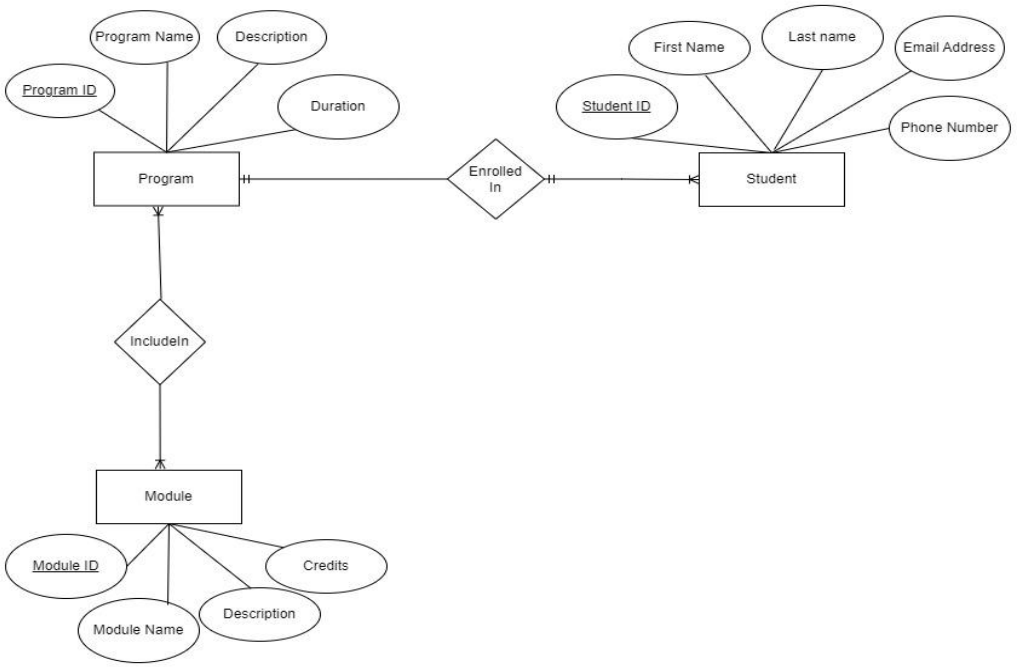

Figure 2: Representing Initial ERD.

3. Normalization

UNF (Unnormalized Form) is raw data with a lot of redundancy. There are two types of repetition in this form: single-instance repeating values and multiple-instance repeating groups. This initial state of data organization is inefficient and prone to data anomalies, which is the starting point for normalization. (Anon., 2024)

3.1 Unnormalized Form(UNF)

Student {StudentID, StudenName, StudentContactnumber, StudentEmaiAddress, ProgramID, ProgramName, ProgramDescription, ProgramDuration, {ModuleID, ModuleName, ModuleCredits, TeacherID, TeacherName, TeacherPhoneNumber, TeacherEmail, AssessmentID, AssessmentTitle, AssessmentDeadline, MarksObtained, ResourceID, ResourceTitle, ResourceType, ResourceSequenceOrder, Resource Duration, { AnnouncementID, AnnouncementTittle, AnnouncementContent, AnnouncementDate }}}

As we know, UNF consists of only one table, here it is Student. So, basically the above section describes the relations and normal forms starting from an unnormalized table called Students. It contains information about students, programs, modules, teachers, assessments, resources and announcement.

- Assign
  Primary Key [PK]
  Foreign key [FK]

3.1 First Normal Form (1NF)

First Normal Form (1NF) requires that each table cell contains exactly one atomic (indivisible) value, and every column has a distinct name. This eliminates repeating groups and ensures data is stored in its most basic form. (Anon., 2024)

- **STUDENT** (StudentID [PK], StudentName, StudentEmail, Student ContactNumber,ProgramID [FK])

- **PROGRAM** (ProgramID [PK], ProgramName, ProgramDescription, ProgramDuration, ModuleID [FK])

- **MODULE** (ModuleID [PK] , ModuleName, ModuleCredits, ProgramID [FK])

- **TEACHER** (TeacherID [PK], TeacherName, TeacherEmail, TeacherContact, Number, ModuleID [FK])

- **ASSESSMENT** (AssessmentID [PK], ModuleID [FK] , AssessmentTitle, AssessmentDeadline, MarkObtained, StudentID [FK])

- **Announcement** (AnnouncementID [PK] , ModuleID [FK] , TeacherID [FK] , AnnouncementDate , AnnouncementContent)

- **RESOURCE** (ResourceID [PK] , ModuleID[FK] , ResourceTitle, ResourceType, ResourceSequenceOrder)

3.3 Second Normal Form (2NF)

Second Normal Form. 2NF is based on 1NF and ensures that all non-key attributes depend on the whole primary key, not just part of it. Thus, second normal form eliminates partial dependencies, where columns relate only to some parts of a composite primary key. To check this, we see if a composite key depends on only one key or all keys.

1. Checking partial functional dependency on Student

StudentID = StudentName, StudentEmailAddress, StudentContactNumber, ProgramID

Here, we can see each non-key attribute of student is fully dependent on primary key which is eliminating the possibility of partial dependency.

2. Checking partial functional dependency on Program

ProgrmID = ProgramName, ProgramDescription, ProgramDuration, ModuleID

Since, we can see partial dependency on Program table so, now we can see split program table to remove partial dependency in module.

- **Splitting Program table into**:
  i) Program = ProgrmID, ProgramName, ProgramDescription, ProgramDuration
  ii) Program_Module = ProgramID, ModuleID

3. Checking partial functional dependency on Module

ModuleID = ModuleName, ModuleCredit, ProgramID

Here, we can see each non-key attribute of Module is fully dependent on primary key which is eliminating the possibility of partial dependency.

4. Checking partial functional dependency on Teacher

TeacherID = TeacherName, TeacherEmail, TeacherContactNumber, ModuleID

Since, we can see each attribute on the teacher table is given by TeacherID itself, the foreign key i.e. the ModuleID doesn't return any non-key attribute which has probability of having partial dependency so, it remove from the teacher table.

- **Splitting the Teacher table:**
  i) Teacher = TeacherID, TeacherName, TeacherEmail, TeacherContactNumber
  ii) Teacher_Module = TeacherID, ModuleID

5. Checking partial functional dependency on Assessment

AssessmentID = ModuleID , AssessmentTitle, AssessmentDeadline, MarkObtained, StudentID

Since, we can see MarkOdtained of Assessment depend on both StudentID and AssessmsntID which has probability of having partial dependency so, now we can see split program table to remove partial dependency in assessment.

- **Splitting the Assessment table:**

  i) Assessment = AssessmentID , AssessmentTitle, AssessmentDeadline, ModuleID

  ii) Student_Assessment = StudentID, ModuleID, MarkObtained

6. Checking partial functional dependency on Announcement

AnnouncementID = ModuleID , TeacherID , AnnouncementDate , AnnouncementContent

Here, we can see each non-key attribute of announcement is fully dependent on primary key which is eliminating the possibility of partial dependency.

7. Checking partial functional dependency on Resource

ResourceID = ModuleID, ResourceTitle, ResourceType, ResourceSequenceOrder

Here, we can see each non-key attribute of resource is fully dependent on primary key which is eliminating the possibility of partial dependency.

3.4 Third Normal Form (3NF)

Third Normal Form (3NF) builds on 2NF by getting rid of transitive dependencies. Transitive dependencies happen when non-key attributes rely on other non-key attributes instead of depending directly on the primary key. This makes sure that each non-key column depends only on the primary key.

Checking partial Transitive dependency on Student

a. StudentID = StudentName, StudentEmailAddress, StudentContactNumber, ProgramID

No transitive dependency found.

b. ProgrmID = ProgramName, ProgramDescription, ProgramDuration

No transitive dependency found.

c. Program_Module = ProgramID, ModuleID

No transitive dependency found.

d. ModuleID = ModuleName, ModuleCredit, ProgramID

No transitive dependency found.

e. TeacherID = TeacherName, TeacherEmail, TeacherContactNumber

No transitive dependency found

f. Teacher_Module = TeacherID, ModuleID

No transitive dependency found

g. AssessmentID = AssessmentTitle, AssessmentDeadline, ModuleID

No transitive dependency found

h. Student_Assessment = StudentID, ModuleID, MarkObtained

No transitive dependency found

i. AnnouncementID = ModuleID , TeacherID , AnnouncementDate , AnnouncementContent

No transitive dependency found

j. ResourceID = ModuleID, ResourceTitle, ResourceType, ResourceSequence

Transitive Dependency found.

As, we can see except resource table, the other table does not contain transitive dependency there we can see ResourceSequenceOrder is dependent ModuleID, so to remove the transitive dependency splitting resource table.

- Splitting the Resource table

i) Resource = ResourceID, ModuleID, ResourceTitle, ResourceType

ii) Module_Resource = ModuleID, ResouceID, ResourceSequenceOrder.

3.4 Final Normalized Table

The final normal forms help in reducing data redundancy, increasing data consistency, and improving database performance. However, higher levels of normalization may lead to more complex database designs and queries. In any case, there should be a balance between normalization and practicality while designing a database.

1. Student(StudentID [PK], StudentName, StudentEmail, Student ContactNumber,ProgramID [FK]**)**

2. Program(ProgrmID [PK], ProgramName, ProgramDescription, ProgramDuration**)**

3. Program_Module(ProgramID [FK], ModuleID [FK]**)**

4. Module(ModuleID [PK], ModuleName, ModuleCredit, ProgramID [FK]**)**

5. Teacher(TeacherID [PK], TeacherName, TeacherEmail, TeacherContactNumber**)**

6. Teacher_Module(TeacherID [FK], ModuleID [FK]**)**

7. Assessment(AssessmentID [PK], AssessmentTitle, AssessmentDeadline, ModuleID [FK]**)**

8. Student_Assessment(StudentID [FK], ModuleID [FK], MarkObtained**)**

9. Announcement(AnnouncementID [PK], ModuleID [FK], TeacherID [FK] , AnnouncementDate , AnnouncementContent**)**

10. Resource(ResourceID [PK], ModuleID [FK], ResourceTitle, ResourceType**)**

11. Module_Resource (ModuleID [FK], ResouceID [FK], ResourceSequenceOrder**)**

4. Data Dictionary And Final ERD

4.1 Data Dictionary

1. Student – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	StudentID	Number	10	Primary key
2	StudentName	Character	40	Not Null
3	StudentEmailAddress	Character	10	Unique
4	StudentContactNumber	Number	10	Unique
5.	ProgramID	Number	10	Foreign key

Table 4: Representing Student.

2. Program – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	ProgramID	Number	10	Primary key
2	ProgramName	Character	40	Not Null
3	ProgramDescription	Character	40	Not Null
4	ProgramDuration	Date	-	Not Null

Table 5: Representing Program.

3. Program_Module – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	ProgramID	Number	10	Foreign key
2	ModuleID	Number	10	Foreign key

Table 6: Representing Program_Module.

4. Module – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	ModuleID	Number	10	Primary key
2	ModuleName	Character	40	Not Null
3	ModuleCredits	Character	10	Not Null
4	ProgramID	Number	10	Foreign key

Table 7: Representing Module

5. Teacher – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	TeacherID	Number	10	Primary key
2	TeacherName	Character	40	Not Null
3	TeacherEmailAddress	Character	10	Unique

4	TeacherContactNumber	Number	10	Unique

Table 8: Representing Teacher.

6. Teacher_Module – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	TeacheerID	Number	10	Foreign key
2	ModuleID	Number	10	Foreign key

Table 9: Representing Teacher_Module

7. Student_Assessment - Entity

S. No.	Attribute Name	Data	Size	Constraints
1	AssessmentID	Number	10	Primary key
2	AssessmentTittle	Character	40	Not Null
3	AssessmentDeadline	Date	-	Not Null
4	ModuleID	Number	10	Foreign Key

Table 10:Representing Student_Assessment.

8. Assessment – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	StudentID	Number	10	Foreign key
2	MarkOdtained	Number	40	Not Null
3	ModuleID	Number	10	Foreign key

Table 11: Representing Assessment.

9. Announcement – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	AnnouncementID	Number	10	Primary key
2	ModuleID	Number	10	Foreign Key
3	AnnouncementDate	Date	-	Not Null
4	AnnouncementContent	Character	50	Not Null
5.	TeacherID	Number	10	Foreign key

Table 12: Representing Announcement.

10. Resource – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	ResourceID	Number	10	Primary key
2	ResourceTittle	Character	40	Not Null
3	ResourceType	Character	40	Not Null
4	ModuleID	Number	10	Foreign key

Table 13: Representing Resource.

11. Module_ Resource – Entity

S. No.	Attribute Name	Data	Size	Constraints
1	ResourceID	Number	10	Foregin key
2	ResourceSequenceOrder	Number	10	Not Null
3	ModuleID	Number	10	Foreign key

Table 14: Representing Module_Resource.

4.2 Final Entity Relationship Diagram

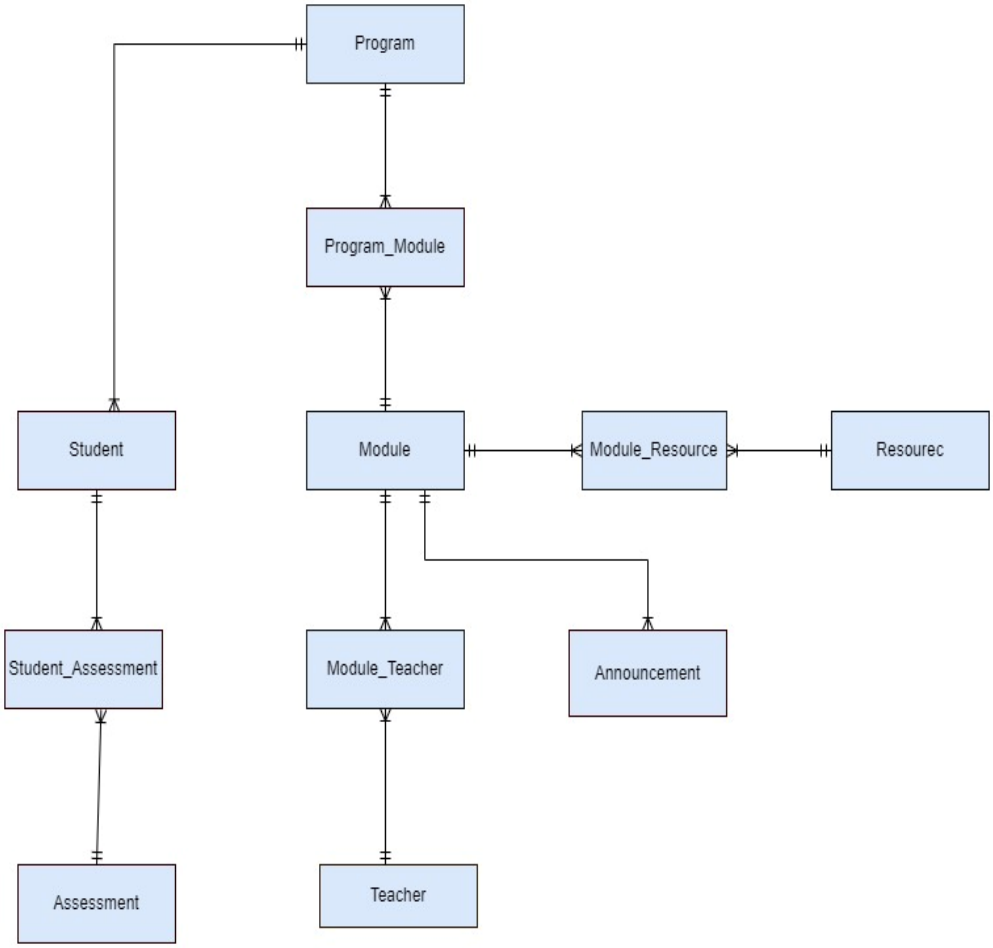

Figure 3: Final ERD.

5. Implementation

i. Table Creation and DESC Table

- Program

```
SQL> CREATE TABLE Program (
  2      ProgramID NUMBER(10) PRIMARY KEY,
  3      ProgramName VARCHAR2(40) NOT NULL,
  4      ProgramDescription VARCHAR2(400) NOT NULL,
  5      ProgramDuration DATE NOT NULL
  6  );

Table created.

SQL> DESC Program;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROGRAMID                                 NOT NULL NUMBER(10)
 PROGRAMNAME                               NOT NULL VARCHAR2(40)
 PROGRAMDESCRIPTION                        NOT NULL VARCHAR2(400)
 PROGRAMDURATION                           NOT NULL DATE
```

Figure 4: Create table Program.

- Student

```
SQL> CREATE TABLE Student (
  2      StudentID NUMBER(10) PRIMARY KEY,
  3      StudentName VARCHAR2(40) NOT NULL,
  4      StudentEmailAddress VARCHAR2(100) UNIQUE NOT NULL,
  5      StudentContactNumber NUMBER(10) UNIQUE NOT NULL,
  6      ProgramID NUMBER(10),
  7      FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID)
  8  );

Table created.

SQL> DESC Student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENTID                                 NOT NULL NUMBER(10)
 STUDENTNAME                               NOT NULL VARCHAR2(40)
 STUDENTEMAILADDRESS                       NOT NULL VARCHAR2(100)
 STUDENTCONTACTNUMBER                      NOT NULL NUMBER(10)
 PROGRAMID                                          NUMBER(10)
```

Figure 5: Create table Student.

- Module

```
SQL> CREATE TABLE Module (
  2      ModuleID NUMBER(10) PRIMARY KEY,
  3      ModuleName VARCHAR2(40) NOT NULL,
  4      ModuleCredits VARCHAR2(10) NOT NULL,
  5      ProgramID NUMBER(10),
  6      FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID)
  7  );

Table created.

SQL> DESC Module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODULEID                                  NOT NULL NUMBER(10)
 MODULENAME                                NOT NULL VARCHAR2(40)
 MODULECREDITS                             NOT NULL VARCHAR2(10)
 PROGRAMID                                          NUMBER(10)
```

Figure 6: Create table Module.

- Program_Module

```
SQL> CREATE TABLE Program_Module (
  2      ProgramID NUMBER(10),
  3      ModuleID NUMBER(10),
  4      PRIMARY KEY (ProgramID, ModuleID),
  5      FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID),
  6      FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
  7  );

Table created.

SQL> DESC Program_Module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROGRAMID                                 NOT NULL NUMBER(10)
 MODULEID                                  NOT NULL NUMBER(10)
```

Figure 7: Create table Program_Module.

- Teacher

```
SQL> CREATE TABLE Teacher (
  2      TeacherID NUMBER(10) PRIMARY KEY,
  3      TeacherName VARCHAR2(40) NOT NULL,
  4      TeacherEmailAddress VARCHAR2(100) UNIQUE NOT NULL,
  5      TeacherContactNumber NUMBER(10) UNIQUE NOT NULL
  6  );

Table created.

SQL> DESC Teacher;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TEACHERID                                 NOT NULL NUMBER(10)
 TEACHERNAME                               NOT NULL VARCHAR2(40)
 TEACHEREMAILADDRESS                       NOT NULL VARCHAR2(100)
 TEACHERCONTACTNUMBER                      NOT NULL NUMBER(10)
```

Figure 8: Create table Teacher.

- Teacher_Module

```
SQL> CREATE TABLE Teacher_Module (
  2      TeacherID NUMBER(10),
  3      ModuleID NUMBER(10),
  4      PRIMARY KEY (TeacherID, ModuleID),
  5      FOREIGN KEY (TeacherID) REFERENCES Teacher(TeacherID),
  6      FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
  7  );

Table created.

SQL> DESC Teacher_Module;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TEACHERID                                 NOT NULL NUMBER(10)
 MODULEID                                  NOT NULL NUMBER(10)
```

Figure 9: Create Teacher_Module.

- Student_Assessment

```
SQL> CREATE TABLE Student_Assessment (
  2      AssessmentID NUMBER(10) PRIMARY KEY,
  3      AssessmentTittle VARCHAR2(40) NOT NULL,
  4      AssessmentDeadline DATE NOT NULL,
  5      ModuleID NUMBER(10),
  6      FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
  7  );

Table created.

SQL> DESC Student_Assessment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 ASSESSMENTID                              NOT NULL NUMBER(10)
 ASSESSMENTTITTLE                          NOT NULL VARCHAR2(40)
 ASSESSMENTDEADLINE                        NOT NULL DATE
 MODULEID                                           NUMBER(10)
```

Figure 10: Create Student_Assessment.

- Assessment

```
SQL> CREATE TABLE Assessment (
  2      StudentID NUMBER(10),
  3      MarkObtained NUMBER(30) NOT NULL,
  4      ModuleID NUMBER(10),
  5      PRIMARY KEY (StudentID, ModuleID),
  6      FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
  7      FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
  8  );

Table created.

SQL> DESC Assessment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 STUDENTID                                 NOT NULL NUMBER(10)
 MARKOBTAINED                              NOT NULL NUMBER(30)
 MODULEID                                  NOT NULL NUMBER(10)
```

Figure 11: Create table Assessment.

- Announcement

```
SQL> CREATE TABLE Announcement (
  2      AnnouncementID NUMBER(10) PRIMARY KEY,
  3      ModuleID NUMBER(10),
  4      AnnouncementDate DATE NOT NULL,
  5      AnnouncementContent VARCHAR2(500) NOT NULL,
  6      TeacherID NUMBER(10),
  7      FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID),
  8      FOREIGN KEY (TeacherID) REFERENCES Teacher(TeacherID)
  9  );

Table created.

SQL> DESC Announcement;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 ANNOUNCEMENTID                            NOT NULL NUMBER(10)
 MODULEID                                           NUMBER(10)
 ANNOUNCEMENTDATE                          NOT NULL DATE
 ANNOUNCEMENTCONTENT                       NOT NULL VARCHAR2(500)
 TEACHERID                                          NUMBER(10)
```

Figure 12: Create table Announcement.

- Resources

```
SQL> CREATE TABLE Resources (
  2      ResourceID NUMBER(10) PRIMARY KEY,
  3      ResourceTittle VARCHAR2(40) NOT NULL,
  4      ResourceType VARCHAR2(400) NOT NULL,
  5      ModuleID NUMBER(10),
  6      FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
  7  );

Table created.

SQL> DESC Resources;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 RESOURCEID                                NOT NULL NUMBER(10)
 RESOURCETITTLE                            NOT NULL VARCHAR2(40)
 RESOURCETYPE                              NOT NULL VARCHAR2(400)
 MODULEID                                           NUMBER(10)
```

Figure 13: Create table Resources.

- Module_Resources

```
SQL> CREATE TABLE Module_Resources (
  2      ResourceID NUMBER(10),
  3      ResourceSequenceOrder NUMBER(30) NOT NULL,
  4      ModuleID NUMBER(10),
  5      PRIMARY KEY (ResourceID, ModuleID),
  6      FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID),
  7      FOREIGN KEY (ModuleID) REFERENCES Module(ModuleID)
  8  );

Table created.

SQL> DESC Module_Resources;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 RESOURCEID                                NOT NULL NUMBER(10)
 RESOURCESEQUENCEORDER                     NOT NULL NUMBER(30)
 MODULEID                                  NOT NULL NUMBER(10)
```

Figure 14: Create Module_Resources.

ii. Data Insertion and Display Inserted Data

- Programs

```
SQL> INSERT ALL
  2     INTO Program VALUES(1, 'Computer Science', 'Bachelor degree in Computer Science', TO_DATE('2024-09-01', 'YYYY-MM-DD'))
  3     INTO Program VALUES(2, 'Data Science', 'Master degree in Data Science', TO_DATE('2024-09-01', 'YYYY-MM-DD'))
  4     INTO Program VALUES(3, 'Business Administration', 'Bachelor in Business Administration', TO_DATE('2024-09-01', 'YYYY-MM-DD')
)
  5     INTO Program VALUES(4, 'Digital Marketing', 'Advanced Certificate in Digital Marketing', TO_DATE('2024-09-01', 'YYYY-MM-DD')
)
  6     INTO Program VALUES(5, 'Software Engineering', 'Master in Software Engineering', TO_DATE('2024-09-01', 'YYYY-MM-DD'))
  7     INTO Program VALUES(6, 'Cybersecurity', 'Master in Cybersecurity', TO_DATE('2024-09-01', 'YYYY-MM-DD'))
  8     INTO Program VALUES(7, 'Artificial Intelligence', 'Master in AI', TO_DATE('2024-09-01', 'YYYY-MM-DD'))
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 15: Insert Program.

```
SQL> SELECT * FROM Program;

 PROGRAMID PROGRAMNAME                              PROGRAMDESCRIPTION                                 PROGRAMDU
---------- --------------------------------------- -------------------------------------------------- ---------
         1 Computer Science                        Bachelor degree in Computer Science                01-SEP-24
         2 Data Science                            Master degree in Data Science                      01-SEP-24
         3 Business Administration                 Bachelor in Business Administration                01-SEP-24
         4 Digital Marketing                       Advanced Certificate in Digital Marketing          01-SEP-24
         5 Software Engineering                    Master in Software Engineering                     01-SEP-24
         6 Cybersecurity                           Master in Cybersecurity                            01-SEP-24
         7 Artificial Intelligence                 Master in AI                                       01-SEP-24

7 rows selected.
```

Figure 16: Display Program.

- Student

```
SQL> INSERT ALL
  2  INTO Student VALUES (1, 'John Doe', 'john.doe@email.com', 1234567890, 1)
  3  INTO Student VALUES (2, 'Jane Smith', 'jane.smith@email.com', 2345678901, 1)
  4  INTO Student VALUES (3, 'Bob Johnson', 'bob.johnson@email.com', 3456789012, 2)
  5  INTO Student VALUES (4, 'Alice Brown', 'alice.brown@email.com', 4567890123, 2)
  6  INTO Student VALUES (5, 'Charlie Wilson', 'charlie.wilson@email.com', 5678901234, 3)
  7  INTO Student VALUES (6, 'Diana Miller', 'diana.miller@email.com', 6789012345, 3)
  8  INTO Student VALUES (7, 'Edward Davis', 'edward.davis@email.com', 7890123456, 4)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 17: Inserting Student.

```
SQL> SET LINESIZE 200;
SQL> SET PAGESIZE 50;
SQL> SELECT * FROM STUDENT;

 STUDENTID STUDENTNAME                        STUDENTEMAILADDRESS                                STUDENTCONTACTNUMBER  PROGRAMI
D
---------- --------------------------------- -------------------------------------------------- -------------------- ---------
-
         1 John Doe                          john.doe@email.com                                            1234567890
1
         2 Jane Smith                        jane.smith@email.com                                          2345678901
1
         3 Bob Johnson                       bob.johnson@email.com                                         3456789012
2
         4 Alice Brown                       alice.brown@email.com                                         4567890123
2
         5 Charlie Wilson                    charlie.wilson@email.com                                      5678901234
3
         6 Diana Miller                      diana.miller@email.com                                        6789012345
3
         7 Edward Davis                      edward.davis@email.com                                        7890123456
4

7 rows selected.
```

Figure 18: Display Student.

- Module

```
SQL> INSER ALL
SP2-0042: unknown command "INSER ALL" - rest of line ignored.
SQL> INSERT ALL
  2   INTO Module VALUES (1, 'Programming Basics', '10', 1)
  3   INTO Module VALUES (2, 'Database Systems', '15', 1)
  4   INTO Module VALUES (3, 'Data Analytics', '20', 2)
  5   INTO Module VALUES (4, 'Machine Learning', '15', 2)
  6   INTO Module VALUES (5, 'Marketing Principles', '10', 3)
  7   INTO Module VALUES (6, 'Business Ethics', '10', 3)
  8   INTO Module VALUES (7, 'Digital Strategy', '15', 4)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 19: Insert Module.

```
SQL> SELECT * FROM MODULE;

  MODULEID MODULENAME                         MODULECREDITS                                          PROGRAMID
---------- ------------------------------------ ---------------------------------------------------- ----------
         1 Programming Basics                   10                                                          1
         2 Database Systems                     15                                                          1
         3 Data Analytics                       20                                                          2
         4 Machine Learning                     15                                                          2
         5 Marketing Principles                 10                                                          3
         6 Business Ethics                      10                                                          3
         7 Digital Strategy                     15                                                          4

7 rows selected.
```

Figure 20: Display Module.

- Teacher

```
SQL> INSERT ALL
  2   INTO Teacher VALUES (1, 'Prof Anderson', 'anderson@faculty.edu', 1122334455)
  3   INTO Teacher VALUES (2, 'Dr Martinez', 'martinez@faculty.edu', 2233445566)
  4   INTO Teacher VALUES (3, 'Prof Wilson', 'wilson@faculty.edu', 3344556677)
  5   INTO Teacher VALUES (4, 'Dr Brown', 'brown@faculty.edu', 4455667788)
  6   INTO Teacher VALUES (5, 'Prof Clark', 'clark@faculty.edu', 5566778899)
  7   INTO Teacher VALUES (6, 'Dr Lee', 'lee@faculty.edu', 6677889900)
  8   INTO Teacher VALUES (7, 'Prof Zhang', 'zhang@faculty.edu', 7788990011)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 21: Insert Teacher.

```
SQL> SET LINESIZE 200;
SQL> SET PAGESIZE 50;
SQL> SELECT * FROM Teacher;

 TEACHERID TEACHERNAME                          TEACHEREMAILADDRESS                               TEACHERCONTACTNUMBER
---------- ------------------------------------ ------------------------------------------------- --------------------
         1 Prof Anderson                        anderson@faculty.edu                                        1122334455
         2 Dr Martinez                          martinez@faculty.edu                                        2233445566
         3 Prof Wilson                          wilson@faculty.edu                                          3344556677
         4 Dr Brown                             brown@faculty.edu                                           4455667788
         5 Prof Clark                           clark@faculty.edu                                           5566778899
         6 Dr Lee                               lee@faculty.edu                                             6677889900
         7 Prof Zhang                           zhang@faculty.edu                                           7788990011

7 rows selected.
```

Figure 22: Display Teacher.

- Module_Teacher

```
SQL> INSERT ALL
  2   INTO Teacher_Module VALUES (1, 1)
  3   INTO Teacher_Module VALUES (1, 2)
  4   INTO Teacher_Module VALUES (2, 3)
  5   INTO Teacher_Module VALUES (2, 4)
  6   INTO Teacher_Module VALUES (3, 5)
  7   INTO Teacher_Module VALUES (4, 6)
  8   INTO Teacher_Module VALUES (5, 7)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 23: Insert Module_Teacher.

```
SQL> SELECT * FROM Teacher_Module;

 TEACHERID   MODULEID
---------- ----------
         1          1
         1          2
         2          3
         2          4
         3          5
         4          6
         5          7

7 rows selected.
```

Figure 24: Display Teacher_Module.

- Student_Assessment

```
SQL> INSERT ALL
  2   INTO Student_Assessment VALUES (1, 'Programming Project', TO_DATE('2024-10-15', 'YYYY-MM-DD'), 1)
  3   INTO Student_Assessment VALUES (2, 'Database Design', TO_DATE('2024-11-01', 'YYYY-MM-DD'), 2)
  4   INTO Student_Assessment VALUES (3, 'Data Analysis Report', TO_DATE('2024-10-20', 'YYYY-MM-DD'), 3)
  5   INTO Student_Assessment VALUES (4, 'ML Algorithm Implementation', TO_DATE('2024-11-15', 'YYYY-MM-DD'), 4)
  6   INTO Student_Assessment VALUES (5, 'Marketing Plan', TO_DATE('2024-10-30', 'YYYY-MM-DD'), 5)
  7   INTO Student_Assessment VALUES (6, 'Ethics Case Study', TO_DATE('2024-11-10', 'YYYY-MM-DD'), 6)
  8   INTO Student_Assessment VALUES (7, 'Digital Campaign', TO_DATE('2024-10-25', 'YYYY-MM-DD'), 7)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 25: Student_Assessment.

```
SQL> SET LINESIZE 200;
SQL> SET PAGESIZE 50;
SQL> SELECT * FROM Student_Assessment;

ASSESSMENTID ASSESSMENTTITTLE                       ASSESSMEN   MODULEID
------------ ------------------------------------- --------- ----------
           1 Programming Project                   15-OCT-24          1
           2 Database Design                       01-NOV-24          2
           3 Data Analysis Report                  20-OCT-24          3
           4 ML Algorithm Implementation           15-NOV-24          4
           5 Marketing Plan                        30-OCT-24          5
           6 Ethics Case Study                     10-NOV-24          6
           7 Digital Campaign                      25-OCT-24          7

7 rows selected.
```

Figure 26: Display Student_Assessment.

- Assessment

```
SQL> INSERT ALL
  2   INTO Assessment VALUES (1, 85, 1)
  3   INTO Assessment VALUES (2, 92, 1)
  4   INTO Assessment VALUES (3, 88, 2)
  5   INTO Assessment VALUES (4, 95, 2)
  6   INTO Assessment VALUES (5, 90, 3)
  7   INTO Assessment VALUES (6, 87, 3)
  8   INTO Assessment VALUES (7, 93, 4)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 27: Insert Assessment.

```
SQL> SELECT * FROM Assessment;

 STUDENTID MARKOBTAINED   MODULEID
---------- ------------ ----------
         1           85          1
         2           92          1
         3           88          2
         4           95          2
         5           90          3
         6           87          3
         7           93          4

7 rows selected.
```

Figure 28: Display Assessment.

- Announcement

```
SQL> INSERT ALL
  2   INTO Announcement VALUES (1, 1, TO_DATE('2024-09-01', 'YYYY-MM-DD'), 'Welcome to Programming Basics', 1)
  3   INTO Announcement VALUES (2, 2, TO_DATE('2024-09-02', 'YYYY-MM-DD'), 'Database Project Guidelines', 1)
  4   INTO Announcement VALUES (3, 3, TO_DATE('2024-09-03', 'YYYY-MM-DD'), 'Data Analytics Workshop', 2)
  5   INTO Announcement VALUES (4, 4, TO_DATE('2024-09-04', 'YYYY-MM-DD'), 'ML Project Deadline Extended', 2)
  6   INTO Announcement VALUES (5, 5, TO_DATE('2024-09-05', 'YYYY-MM-DD'), 'Guest Speaker Next Week', 3)
  7   INTO Announcement VALUES (6, 6, TO_DATE('2024-09-06', 'YYYY-MM-DD'), 'Case Study Submission Guidelines', 4)
  8   INTO Announcement VALUES (7, 7, TO_DATE('2024-09-07', 'YYYY-MM-DD'), 'Digital Marketing Tools Workshop', 5)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 29: Insert Announcement.

```
SQL> SET LINESIZE 200;
SQL> SET PAGESIZE 50;
SQL> SELECT * FROM Announcement;

ANNOUNCEMENTID   MODULEID ANNOUNCEM ANNOUNCEMENTCONTENT                                                TEACHERID
-------------- ---------- --------- -------------------------------------------------- ----------
             1          1 01-SEP-24 Welcome to Programming Basics                                      1
             2          2 02-SEP-24 Database Project Guidelines                                        1
             3          3 03-SEP-24 Data Analytics Workshop                                            2
             4          4 04-SEP-24 ML Project Deadline Extended                                       2
             5          5 05-SEP-24 Guest Speaker Next Week                                            3
             6          6 06-SEP-24 Case Study Submission Guidelines                                   4
             7          7 07-SEP-24 Digital Marketing Tools Workshop                                   5

7 rows selected.
```

Figure 30: Display Announcement.

- Resources

```
SQL> INSERT ALL
  2  INTO Resources VALUES (1, 'Programming Basics Slides', 'PDF', 1)
  3  INTO Resources VALUES (2, 'Database Design Guide', 'PDF', 2)
  4  INTO Resources VALUES (3, 'Data Analytics Tutorial', 'Video', 3)
  5  INTO Resources VALUES (4, 'ML Algorithms Guide', 'PDF', 4)
  6  INTO Resources VALUES (5, 'Marketing Case Studies', 'PDF', 5)
  7  INTO Resources VALUES (6, 'Ethics Framework Document', 'PDF', 6)
  8  INTO Resources VALUES (7, 'Digital Marketing Tools Guide', 'PDF', 7)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 31: Insert Resources.

```
SQL> SET LINESIZE 200;
SQL> SET PAGESIZE 50;
SQL> SELECT * FROM Resources;

RESOURCEID RESOURCETITTLE                          RESOURCETYPE                                MODULEID
---------- --------------------------------------- ------------------------------------------ ----------
         1 Programming Basics Slides               PDF                                                 1
         2 Database Design Guide                   PDF                                                 2
         3 Data Analytics Tutorial                 Video                                               3
         4 ML Algorithms Guide                     PDF                                                 4
         5 Marketing Case Studies                  PDF                                                 5
         6 Ethics Framework Document               PDF                                                 6
         7 Digital Marketing Tools Guide           PDF                                                 7

7 rows selected.
```

Figure 32: Display Resources.

- Module_Resources

```
SQL> INSERT ALL
  2   INTO Module_Resources VALUES (1, 1, 1)
  3   INTO Module_Resources VALUES (2, 1, 2)
  4   INTO Module_Resources VALUES (3, 1, 3)
  5   INTO Module_Resources VALUES (4, 1, 4)
  6   INTO Module_Resources VALUES (5, 1, 5)
  7   INTO Module_Resources VALUES (7, 1, 7)
  8   INTO Module_Resources VALUES (6, 1, 6)
  9  SELECT * FROM DUAL;

7 rows created.
```

Figure 33: Module_Resources.

```
SQL> SELECT * FROM Module_Resources;

RESOURCEID RESOURCESEQUENCEORDER   MODULEID
---------- --------------------- ----------
         1                     1          1
         2                     1          2
         3                     1          3
         4                     1          4
         5                     1          5
         6                     1          6
         7                     1          7

7 rows selected.
```

Figure 34: Display Module_Resources.

6. References

Anon., 2024. *normal-forms-in-dbms.* [Online]

Available at: https://www.geeksforgeeks.org/normal-forms-in-dbms/

[Accessed 23 July 2024].