LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5051NI Databases

100% Individual Coursework

Autumn 2024

Credit: 15 Semester Long Module

Student Name: Erika Shrestha

London Met ID: 23048598

Assignment Submission Date:23rd January 2025.

Word Count: 7740

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.

23048598ERIKASHRESTHAFINAL.docx

Islington College,Nepal

Document Details

Submission ID

trn:oid:::3618:79930045

Submission Date

Jan 23, 2025, 12:16 PM GMT+5:45

Download Date

Jan 23, 2025, 12:18 PM GMT+5:45

File Name

23048598ERIKASHRESTHAFINAL.docx

File Size

51.5 KB

98 Pages

6,783 Words

44,714 Characters

20% Overall Similarity

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

Match Groups

- 133 Not Cited or Quoted 19%
  Matches with neither in-text citation nor quotation marks
- 11 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

6% Internet sources

0% Publications

19% 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.

Abstract

The project “EVISION ROOM” is a multi-functional database system that provides our client with facilities like student and staff management, retrieving necessary related information and to properly monitor activities of the “Oak Tree Academy”.

This documentation accompanies the client to how the development and implementation of various data is done with proper manual guidance. There are several sections which also contribute to user with different sql languages.

In addition, the documentation explains the different entities required to create an appropriate database with the help of normalization and entity relationships diagram that will further help to perform actual implementation and practice of school data management.

Acknowledgement

I would like to express my heartfelt gratitude to Mr. Deepesh Raj Adhikari (Academic mentor of Databases) for guiding me throughout the coursework process. Along with the guidance, I am thankful for all the feedback received during the project completion.

In addition, I would also like to thank Mr. Aadesh Tandukar (Academic mentor of Databases) for helping me with unexpected coursework-related issues.

I am grateful to my friends and family for constantly being patient, supportive and encouraging.

Table of Contents

Table of Figures

Table of Tables

1. INTRODUCTION

The **"EVISION ROOM"** database, created by Contech Design Company acts as a data manager or guide for **Ms. Mary** to keep track of all related objects of the college and achieve its core objectives and goals. It aims to introduce the concept of database management system which involves practice of **normalization**, use of constraints for an e-classroom platform where the users can experience a user-friendly service. Upon successful completion of this subject, users can get a better understanding of what the business is and how each of its functionalities works.

1.1. DATABASE

The increased use in the application of computers is seen due to the introduction of a database approach in areas like business, e-classrooms, libraries, hospitals and many more. A general description of database comes from storing, modifying and managing related data in tabular form i.e. rows and columns where rows are values and columns are attributes. (Ramez Elmasri, 2015) A database can cope with both simplicity and complexity of data in various platforms. To support this, A large commercial database is MyTeacher where millions of active users like students and teachers interact with feedback, study slides for each subject areas and assignments portals.

1.2. DATABASE MANAGEMENT SYSTEM

Database is basically an alternative solution to the file-handling system and Database Management system is a platform to interact with databases using SQL (Structured

Query Language). (Ramez Elmasri, 2015)` DBMS like Oracle is used in this project where interaction between oracle databases is done by SQL + plus version 11.2.0.2.

Figure 1 Oracle Logo

Figure 2 Sql Plus Logo

1.3. OTHER TOOLS USED

Apart from the actual operational tools used in creating a database “EVISION ROOM”, a few tools were implemented to complete the documentation and create figures to support the statements involved in the project. For the preparation of documentation word (Microsoft 365) is used whereas draw.io is used for tables, diagrams and figure creation.

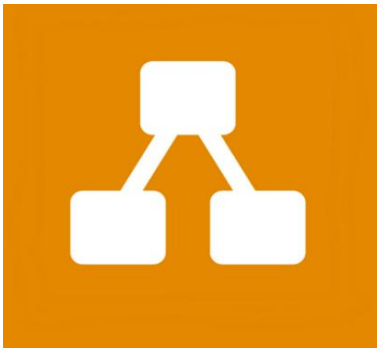


Figure 3 Draw.io Logo

Figure 4 Word Logo

1.4. BUSINESS AND ITS FORTE

“CONTECH designs” is a well-established company which specializes in introducing strong databases suitable for both small and large-scale enterprises. In this case, the company is planning to create a database called “EVISION ROOM” at request to our client “Ms. Mary” who insisted on a system for her e-classroom endeavor and expressed support for the success of the new innings.

Before 20th January of the year 2025, our mission is to make a full-fledge database with integrated functionalities just as Ms. Mary suggested involving access and organization of students, teachers and programs details and their allocation, provision of assessment, and grading results. Additionally, with the help of our database, students and teachers can smoothly interact with various means like feedback, announcements, and assignment markings.

1.5. CURRENT SCENARIO AND OPERATIONS

Ms. Mary runs a college “OAK TREE ACADEMY” with total students of 10 from the bachelor’s slot and 5 from master’s slot. The college offers various programs like BSCs in Computing, BSCs in Networking, BSCs in Multimedia and BSCs in Computing with Artificial Intelligence under IT bachelor’s degree whereas IT master’s degree offers two programs i.e. MSc in Data Analyst and MSc in Cyber Security with a total of 10 academic staff. The program contains different modules like java, fundamentals of computing, information system and many more. With that, new opportunities have been discussed and proceeded for scholarship awards categorized by academic performance and sport activities based where eligible students must hold dignified achievements in available sports like basketball, football, chess, long tennis etc.

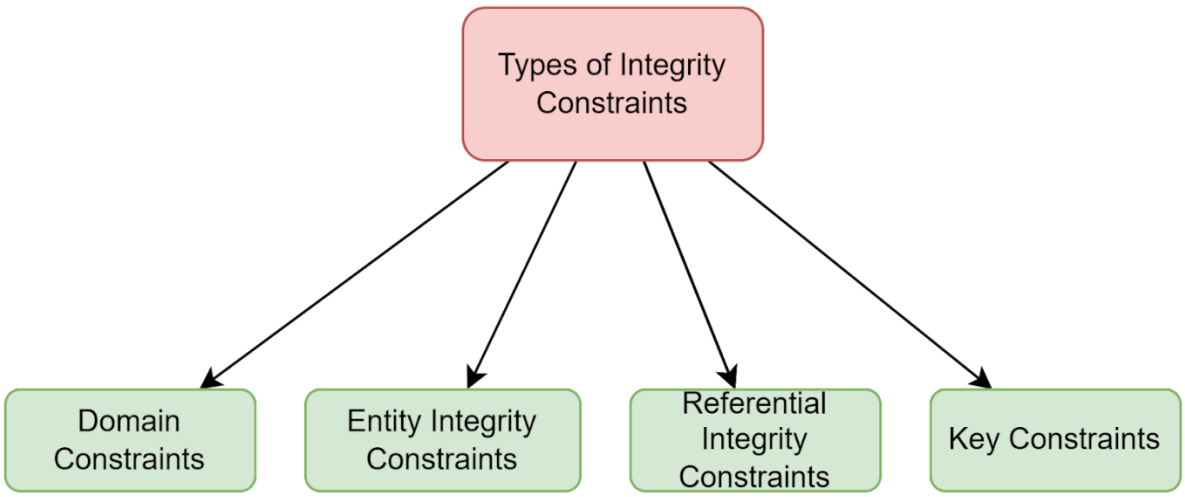
However, just after 5 months of college, difficulties in manual handling the records of students, teachers and programs containing various modules resulted in fluctuation of details, tracking problems, delays in updates and lack of proper course enrollments. The proper implementation of a structured and user-friendly environment must be carried out which will further help faculty staff in decision-making and improve the situation of the college’s operations. Apart from functionality, data integrity, data redundancy and data security are practiced which Ms. Mary, the principal of the college stated as her top priority.

2. IDENTIFICATION OF ENTITY AND ATTRIBUTES

In this section, we identify the **core objects** (entities) which relate to things in the real world and likewise provide properties (attributes) for each of them and the rules the attributes need to follow. According to the “OAK TREE ACADEMY”, involvement of both **tangible** and **intangible** entities is seen where tangible entities include Student, Teacher, Resource and Program, Module, Assessment, Announcement, Result fall under intangible entities.

2.1. CONSTRAINTS

A set of rules termed constraints are applied to necessary attributes of identified entities and limit the data format of a table which further improves the readability and accuracy of data.



Domain constraints → contact_no must be 10 digits strictly.

Entity Integrity constraints → Ensures every table has primary key like **student_id**, **program_id** and **module_code** and these values cannot be null.

Referential integrity constraints → ensures consistency between two tables using foreign key identified after normalization.

Key constraints → ensures uniqueness of records apart from primary key like **contact_no**, **teacher_id**, **email_address.**

KEY:

Primary key: it is a selected candidate key or a key attribute which single handily helps to acquire/retrieve all other related attributes(column). (Abraham Silberschatz, 2011) An entity can have one or multiple primary keys called a composite primary key. A primary key must always be unique and not null which itself is a constraint. A primary key is represented by (PK) or underlining.

Foreign key: They act as a connector between entities because foreign key represents primary key of another entity present in an entity. Usually, a foreign key is indicated as FK or *.

Not null: Nulling or not nulling a value depends on whether the entities are mandatory or optional relations. If an optional relation is seen, values can be null otherwise it cannot be null. NN or double N is used to represent not null.

Unique: Apart from primary keys, candidate keys can have unique data. Uniqueness of an attribute is denoted by U.

Primary key in the listed entities is mentioned below:

ENTITY	PRIMARY KEY
Student	student_id
Program	program_id
Module	module_code

Table 1 Core Entities with key attribute

2.2. ENITITY AND ATTRIBUTES

An entity is a **thing** or **object** in the real world that can be described as such which is called attributes (GeeksforGeeks, 2023). Entities are **recorded** in database which can be easily **distinguished** from **groups.**

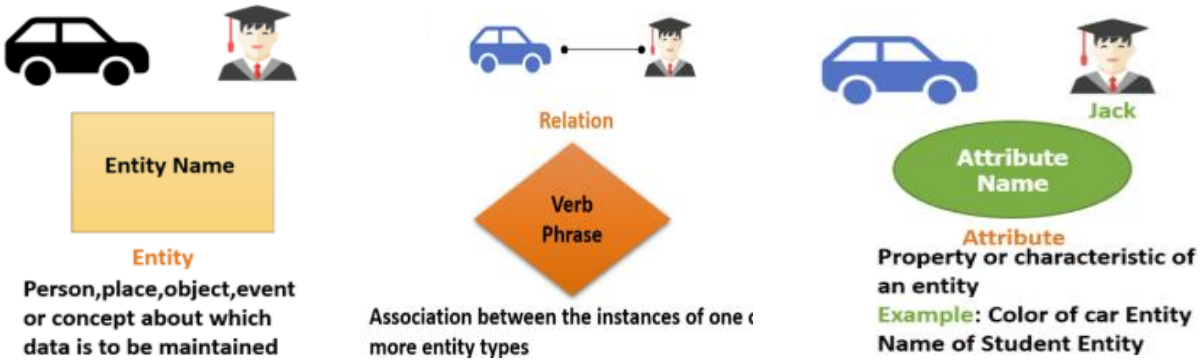


Figure 5 Entities and Attributes

STUDENT

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINTS
1	student_id	Number	7	Primary key
2	first_name	Character	15	Not null
3	middle_name	Character	15	
4	last_name	Character	15	Not null
5	home_address	Character	30	Not null
6	contact_no	Number	10	Unique
7	email_address	Character	15	Unique
8	gender	Character	6	Not null
9	age	Number	2	Not null

10	enrolled_date	Date		Not null

Table 2 Student entity: attributes

PROGRAM

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINTS
1	program_id	Number	7	Primary key
2	program_title	Character	15	Not null
3	program_length	Number	5	Not null
4	total_credit	Number	5	Not null
5	tuition_fee	Number	20	Not null
6	entry_requirement	Character	5	Not null
7	maximum_capacity	Number	20	Not null
8	Mode_of_delivery	Character	15	Not null

Table 3 Program entity: attributes

MODULE

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINTS
1	module_code	Number	7	Primary key
2	module_title	Character	15	Unique
3	credit_score	Number	5	Not null
4	duration_period	Number	5	Not null
5	academic_level	Number	5	Not null
6	teacher_id	Number	7	Unique
7	first_name	Character	15	Not null
8	middle_name	Character	15	

9	last_name	Character	15	Not null
10	home_address	Character	30	Not null
11	contact_no	Number	10	Unique
12	educational_background	Character	25	Not null
13	years_of_experience	Number	5	Not null
14	salary	Number	20	Not null
15	joined_date	Date		Not null
16	assessment_id	Number	8	Unique
17	assessment_title	Character	15	Not null
18	assigned_date	Date		Not null
29	due_date	Date		Not null
20	category	Character	10	Not null
21	weightage	Number	5	Not null
22	resource_id	Number	8	Unique
23	resource_title	Character	15	Not null
24	format_type	Character	10	Not null
25	duration	Number	5	Not null
26	result_code	Number	8	Unique
27	obtained_mark	Number	5	Not null
28	grade	Character	5	Not null
29	attempt_no	Number	5	Not null
30	result_released_date	Date		Not null
31	announcement_id	Number	8	Unique
32	notice_headline	Character	50	Not null
33	uploaded_date	Date		Not null
34	expired_date	Date		Not null

Table 4 Module entity: attributes

NOTE: The category of attributes is represented in the first row highlighted in grey whereas the second row contains a set of the entity's attributes.

3. BUSINESS RULE

This section elaborates **guidelines** of the system's daily operation, **assumption** made and **relation** between entities and how they are dependent on each other. With that, it helps to understand the requirements of the system and its core objectives and goals.

3.1. BUSINESS RULE FOR THE SYSTEM

- A student must enroll in one program only and each program should contain at least one student.
- A single module can belong to various programs and a program must contain at least one module.
- Every student is required to attend all modules of their respective program.
- A teacher can be responsible for single or many modules.
- A module is assigned to one or many teachers.
- Each teacher must assign assessments for their specific modules only.
- A module can include one or many assessments.
- Each result is associated with only one module.
- Students can view results of only completed assessments.
- A resource is associated with a single module.

- The programs must be related to information Technology subject only.
- Each module can contain at least one resource.
- Students must complete a resource to access another resource.
- A teacher can make announcements for their specific modules only.
- An announcement is associated with one module only.
- Each assessment must include attributes like ID, title, deadline and weightage.
- Resources must be marked as completed to proceed with the next resource.
- The new module may not decide which program or teachers it belongs for the moment.

3.2. SCENARIO ASSUMPTIONS

Assumption1: A module is taught by many teachers instead of a single teacher.

Assumption 2: Different programs are associated with common modules.

Assumption 3: Some modules may or may not be assigned to any teacher which generates optional modality.

Assumption 4: Some modules may or may not be associated with a program which also indicates a possibility of optional modality.

Assumption 5: Teacher and student may not have direct relationship but can be indirectly associated with the help of module.

3.3. CARDINALITY AND MODALITY

Cardinality is the **largest occurrence** of entities that are connected to each other whereas modalities describe whether the requirement of relationship between the entities is **optional** or **mandatory**. (Sudarshan, n.d.) Usually cardinality involves:

One-to-one (1: 1): This relation shows that at most one instance of two connected entities are associated. It is the rarest relationship to occur.

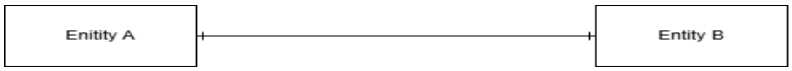


Figure 6 One-to-one relation

One-to-many (1: N): This relation occurs when at most one instance of entity A is associated with any number of entity B but at most one instance of entity B is associated with at most one instance of entity A (Alalouf, 2018).

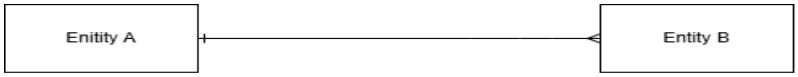


Figure 7 One-to-many relation

Many-to-many (M: N): This relation indicates that at most one instance of entity A is assigned with any number of instances of entity B and vice versa. A bridging entity is needed to solve this relationship.

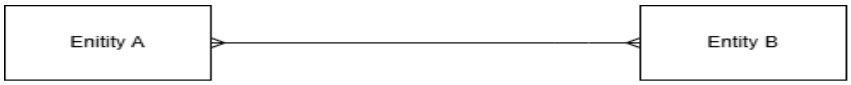


Figure 8 Many-to-many relation

Likewise, a modality can be of **two** types:

Optional (O): it means that the relationship between two entities may or may not be required.

Mandatory (|): it means that the relationship between two entities is required.

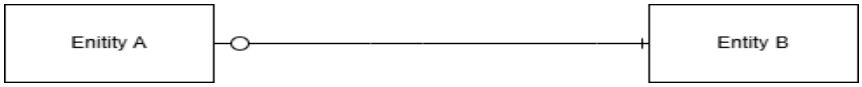


Figure 9 Optional and Mandatory relation

The diagram below shows the actual position of cardinality and modality:

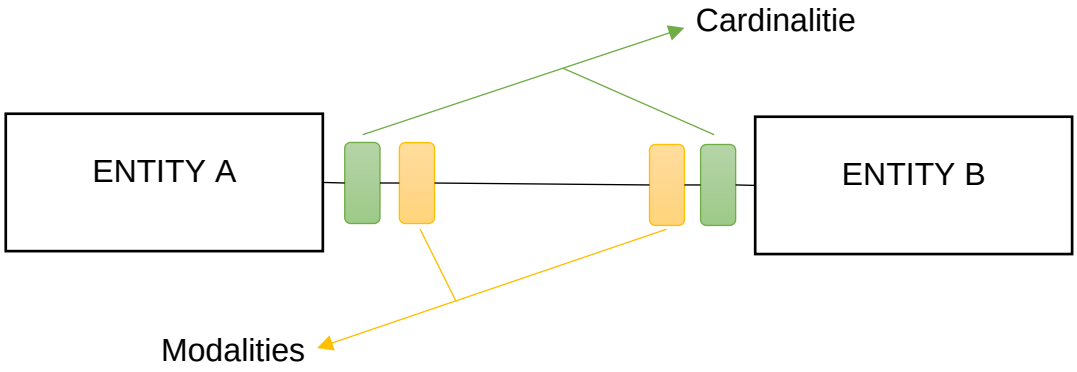


Figure 10 Cardinality and Modality position

NOTE: In **crow foot notation** one (1) can be represented by | a bar and many(N) as **“ ”.**

4. CONCEPTUAL DATABASE DESIGN

This section describes the importance of **data modelling** before initiating database development because it represents the requirement of the system like entities (**Strong** and **weak entities**), attributes, relations and constraints that were identified in section 2. The design is based on requirement gathering and **data analysis** to build a basis for the **physical database design**. The last part of the section contains initial er-diagram before any modification to give an overview of the core entities.

4.1. ENTITY-RELATION DIAGRAM

In the year 1976, the Entity-relation data model was first introduced by **Peter Chen** to provide a simple and understandable **working flow** of the system. Later its variation was further used to **sketch** real-world objects like people, events, non-living things and many more.

An er-diagram is a **visual representation** of identified entities with their respective attributes and applies connectivity to those entities according to their relation (Brown, 2024). In other words, it displays an overall logical structure of our database “EVISION ROOM”. This data model design contributes to building a blueprint for databases, provides clear vision of dependencies, determines flow of data, and helps in making relational databases.

4.2. ENTITY TYPES: STRONG ENTITIES VS WEAK ENTITIES

An entity includes two different types of characteristics that are strong and weak where each type determines their dependencies with one another.

4.2.1. STRONG ENTITIES

Strong entities are those entities which have key attributes and do not need any composite key to uniquely identify them. For example, entities like Student, Teacher, Program, Module. Strong entities are represented by a single line rectangle.

Therefore, listed Strong entities are mentioned below:

i. Student
ii. Teacher
iii. Program
iv. Module

4.2.2. WEAK ENTITIES

Weak entities are represented by a double line rectangle where they do need a composite key to uniquely identify them and are dependent on string entity. In this case, Assessment, Resource, Result, Announcement are identified as weak entities because they may have key attributes but need a key attribute of strong entities that is Module to identify them uniquely.

Therefore, listed weak entities are mentioned below:

i. Assessment
ii. Resource
iii. Result
iv. Announcement

A brief description of strong and weak entities:

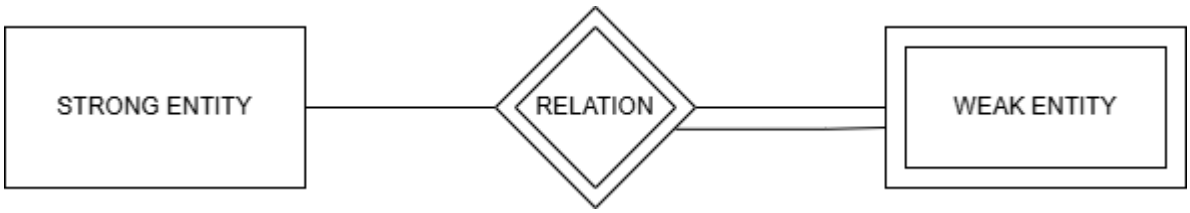


Figure 11 Strong and Weak Entity Relation

NOTE: The weak and string entities will be shown after normalization in final entity relation diagram

4.3. INITIAL ER-DIAGRAM WITH CORE ENTITIES

In the initial Er-diagram, our core entities are Student, Program and Module. Inclusion of all identified entities with the specification of foreign key and bridging entities will be done in logical database design.

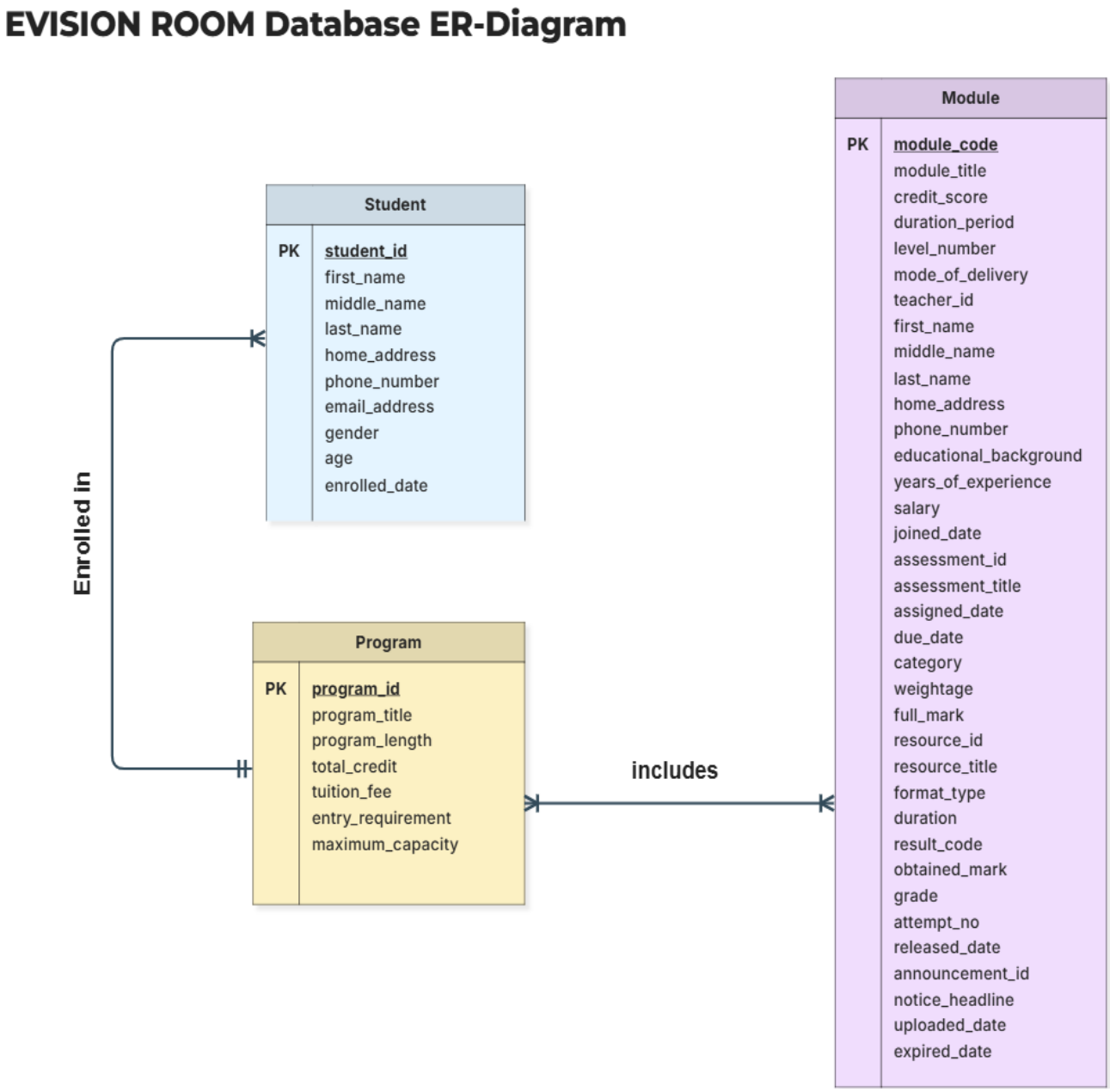


Figure 12 Initial Er-Diagram

5. NORMALIZATION

This section involves a process to avoid **data redundancy** and ensures each row with **unique data**. **Normalization** is done to also solve many-to-many relationships by introducing **bridging entities**. Additionally, completing and achieving a normalized form further helps in developing **final er-diagrams**.

5.1. TYPE OF DEPENDENCIES IN UNNORMALIZATION

The **functional dependencies** and **transitive dependencies** may prevail in unnormalized data where in **full functional dependencies**, attributes depend on two or more composite keys. On the other hand, the **partial functional dependencies** means that attributes do not need additional key attributes to be dependent on. Transitive dependencies, another dependent type which works in format of A→B→C where non-key attributes must be dependent on candidate keys which in turn depends on key attribute (Primary key) (geeksforgeeks, 2024). This shows an indirect relationship between non-key attributes and primary key.

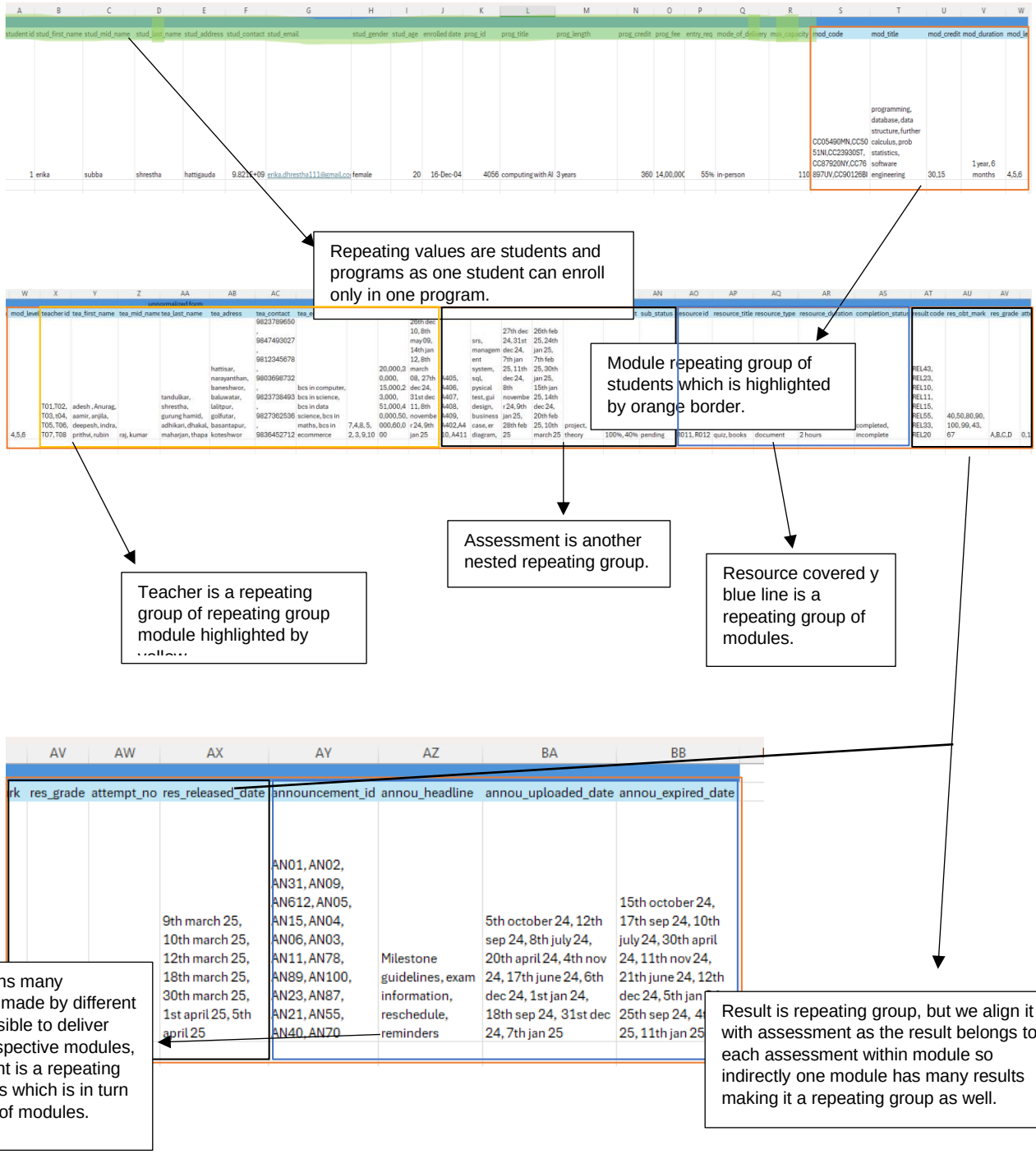
According to the normalization process, the functional dependencies and transitive dependencies occur in the second normalization form **2NF** and third normalization form **3NF.** The notation of functional dependencies is A□B, A,B□C and transitive dependencies is A□B B□C.

5.2. NORMALIZATION LEVEL

According to the entities and database involved in our database “EVISION ROOM”, there are four level breakdown of the process that is **UNF, 1NF, 2NF, 3NF.** To make the process easier and understandable, use of table structure for UNF is done with the help of Microsoft excel tool.

5.2.1. UNF (Unnormalized Form)

In this form, we determine the **repeating value** and **repeating group** where repeating values belong to those outside the curly brackets {} but inside small brackets () and repeating groups belong to value inside the curly brackets{}. To identify and differentiate each, use of a table and few row values is practiced.



UNF in brackets:

TABLE(**student_id,** student_name, stud_address, stud_contact, stud_email, stud_gender, stud_age, enrolled_date, prog_id, prog_title, prog_length, prog_credit, prog_fee, entry_req, max_capacity, mode_of_delivery{mod_code, mod_title, mod_credit, mod_duration, mod_level{teacher_id, tea_name, tea_address, tea_contact, tea_edu, tea_experience, salary, joined_date{ announcement_id, announcement_headline, uploaded_date, expired_date}}, {assessment_id, ass_title, assigned_date, due_date, category, weightage, result_code, obtained_mark, grade, attempt_no, released_date}, {resource_id, resource_title, resource_type, resource_duration}})

5.2.2. 1NF (First Normalized Form)

This form operates to separate each repeating groups in different tables (old and new) where old belongs to repeating values and new represent nested repeating groups is present (repeating group inside repeating group) i.e. **module, resource, teacher, assessment, resource, announcement, result.** Here, assessment and result attributes are kept side by side as result is influenced by each assessment within a module. Also, announcement is a repeating group of teachers which in turn is a repeating group of modules, so it acts as a nested repeating group of modules.

Repeating values in student table

- student (**student_id**, student_name, student_address, student_contact, student_email, student_gender, student_age, enrolled_date, program_id, program_title, program_length, program_credit, program_fee, entry_requirement, max_capacity, mode_of_delivery)

each repeating and nested repeating groups in separate table

- student_module(**student_id , module_code**, module_title, module_credit, module_duration, module_level)

- module_teacher(**student_id,module_code, teacher_id**, teacher_name, teacher_address, teacher_contact, teacher_educational_background, teacher_experience, salary, joined_date)

- teacher_announcement(**student_id,module_code,teacher_id,announcement_id**, announcement_headline, uploaded_date, expired_date)

- module_assessment(**student_id, module_code, assessment_id**, assessment_title, assigned_date, due_date, category, weightage, submission_status, result_code, obtained_mark, grade, attempt_no, result_released_date)

- module_resource(**student_id, module_code, resource_id**, resource_title, resource_type, resource_duration, completion_status)

5.2.3. 2NF (Second Normalized Form)

Since the 1NF is in full functional dependency and does not contain partial dependency in 2NF, we analyze and divide each non-key attributes regarding their dependencies with composite keys.

NOTE: The dependencies can only occur where composite keys reside.

In the student table there are no composite keys and only one primary key called student_id, no occurrence of partial dependencies can be seen. This means student_id gives each non-key attribute uniquely.

- student (**student_id**, student_name, student_address, student_contact, student_email, student_gender, student_age, enrolled_date, program_id, program_title, program_length, program_credit, program_fee, entry_requirement, max_capacity, mode_of_delivery)

Here, the dependencies are checked as the student_module table meets the criteria that is (composite keys).

- student_module(**student_id , module_code**, module_title, module_credit, module_duration, module_level)

  Since module_level is the level year. It is determined by students also as a module can be in different programs which might involve masters and bachelors.

  module_code → module_title, module_credit, module_duration

  student_id → ✕

  module_code, student_id → module_level

  The breakdown of student_module table is:
  module_details(**module_code**,module_title,module_credit, module_duration)

  student_module(**student_id , module_code**, module_level)

The attributes in module_teacher table have functional dependencies as three composite keys reside with many attributes. All the attributes like teacher_name,

teacher_address, teacher_contact, teacher_educational_background, teacher_experience, salary, joined_date are partially dependent on teacher_id.

- module_teacher(**<u>student_id,module_code,teacher_id</u>**,teacher_name, teacher_address, teacher_contact, teacher_educational_background, teacher_experience, salary, joined_date)

  teacher_id → teacher_name, teacher_address, teacher_contact, teacher_educational_background, teacher_experience, salary, joined_date

  teacher_id, module_code, student_id →✕

  student_id, teacher_id → ✕

  teacher_id,module_code → ✕

  student_id, module_code → ✕

  student_id → ✕

  module_code → ✕

  The table above table after dependence are:

  teacher_details(**<u>teacher_id</u>**,teacher_name,teacher_address, teacher_contact, teacher_edu, teacher_experience, salary, joined_date)

  teacher_module_student(**<u>teacher_id, module_code, student_id</u>**)

The next dependencies are identified in module_announcement table where non key attributes are announcement_headline, uploaded_date, expired_date where all attributes solely depend on only announcement_id.

- teacher_announcement(**student_id,module_code,teacher_id,announcement_id**, announcement_headline, uploaded_date, expired_date)

  announcement_id → announcement_headline, uploaded_date, expired_date

  student_id, module_code → ✕

  module_code, announcement_id → ✕

  teacher_id, announcement_id → ✕

  teacher_id, module_code → ✕

  teacher_id, student_id → ✕

  module_code, student_id, teacher_id → ✕

  teacher_id, announcement_id, student_id → ✕

  module_code, teacher_id, announcement_id → ✕

  module_code, student_id, announcement_id → ✕

  announcement_id, student_id → ✕

  student_id → ✕

  module_code → ✕

  teacher_id → ✕

  student_id, module_code, teacher_id, announcement_id → ✕

  The table below are after checking the dependencies:

announcement_details(**announcement_id**,announcement_headline, uploaded_date, expired_date)

student_module_teacher_announcement(**student_id,module_code, teacher_id, announcement_id**)

The module_assessment is now checked which includes attributes of both assessment and result where all attributes and result released date depends on assessment whereas the result_code, obtained_marks, grade, attempt_no depends equally on result_code and student_id, module_code and assessment_id.

- module_assessment(**student_id, module_code, assessment_id**, assessment_title, assigned_date, due_date, category, weightage, submission_status, result_code, obtained_mark, grade, attempt_no, result_released_date)

  assessment_id → assessment_title, assigned_date, due_date, category, weightage, result_released_date

  student_id, module_code, assessment_id → result_code, obtained_mark, grade, attempt_no, submission_status

  student_id, module_code → ✕

  module_code, assessment_id → ✕

  student_id, assessment_id → ✕

  student_id → ✕

module_code → ✕

The creation of two tables is seen where one has primary key assessment_id which includes only assessment details, another is composite key of assessment_id, student_id, module_code and result_code which showcase result details connected to student_id, module_code and assessment_id.

assessment_details(**assessment_id**,assessment_title,assigned_date, due_date, category, weightage, result_released_date)

student_module_assessment (**student_id, module_code, assessment_id,** result_code, obtained_mark, grade, attempt_no, submission_status)

The last check is done on module_resource table where non key attributes like resource_title, resource_type, resource_duration depend on only resource_id whereas completion_status depend on student_id, resource_id as the resource completion depends on each student's sequential process.

- module_resource(**student_id, module_code, resource_id**, resource_title, resource_type, resource_duration, completion_status)

resource_id → resource_title, resource_type, resource_duration

student_id → ✕

mod_code → ✕

resource_id, student_id → ✕

student_id, mod_code → ✕

mod_code, resource_id → ✕

resource_id, student_id, module_code → completion_status

The creation of the previous table is divided into two separate table where the first table contains only resource details, the next table is to track the status of student for a resource in a module.

resource_details(**resource_id**, resource_title, resource_type, resource_duration)

student_module_resource(**resource_id, student_id, module_code,** completion_status)

So, the identified tables from 1NF to 2NF are mentioned as follows:

- student (**student_id**, student_name, student_address, student_contact, student_email, student_gender, student_age, enrolled_date, program_id, program_title, program_length, program_credit, program_fee, entry_requirement, max_capacity, mode_of_delivery)

- module_details(**module_code**,module_title,module_credit, module_duration)

- student_module(**student_id , module_code**, module_level)

- teacher_details(**teacher_id**,teacher_name,teacher_address, teacher_contact, teacher_educational_background, teacher_experience, salary, joined_date)

- teacher_module_student(**teacher_id, module_code, student_id**)

- announcement_details(**announcement_id**,announcement_headline, uploaded_date, expired_date)

- student_module_teacher_announcement(**student_id,module_code, teacher_id, announcement_id**)

- assessment_details(**assessment_id**,assessment_title,assigned_date, due_date, category, weightage, result_released_date)

- student_module_assessment (**student_id, module_code, assessment_id,** result_code, obtained_mark, grade, attempt_no, submission_status)

- resource_details(**resource_id**,resource_title,resource_type, resource_duration)

- student_module_resource(**resource_id, student_id, module_code,** completion_status)

5.2.4. 3NF (Third Normalized Form))

The last form but not the least implemented form is the 3NF where transitive dependencies are identified like A → B → C is checked and separated in form A→B B→C where A = primary key attribute, B = candidate key attribute and C is a non-key attribute.

NOTE: The occurrence can only be checked on tables where non key attributes are more than one.

- student (**student_id**, student_name, student_address, student_contact, student_email, student_gender, student_age, enrolled_date, program_id, program_title, program_length, program_credit, program_fee, entry_requirement, max_capacity, mode_of_delivery)

  The transitive dependencies in the student table occur as a candidate key like program_id uniquely gives other non-key attributes like program_title, program_length, program_credit, program_fee, entry_requirement, max_capacity, mode_of_delivery.

  student_id → program_id and program_id → program_name, program_credit, program_fee, entry_req, max_capacity, mode_of_delivery so A→B B→C.

  student_details(**student_id**,student_name,student_address, student_contact,student_email,student_gender,enrolled_date, **program_id***)

  program_details(**program_id**,program_title,program_length, program_credit,program_fee,entry_requirement,max_capacity, mode_of_delivery)

- module_details(**module_code**,module_title,module_credit, module_duration)

  All non-key attributes directly depend on module_code so there is no transitive dependency.

- teacher_details(**teacher_id**,teacher_name,teacher_address, teacher_contact, teacher_educational_background, teacher_experience, salary, joined_date)

  All non-key attributes depend upon teacher_id so there is no transitive dependency.

- announcement_details(**announcement_id**,announcement_headline, uploaded_date, expired_date)

  All the non-key attributes depend on announcement_id so it is not in transitive dependency.

- assessment_details(**assessment_id**,assessment_title,assigned_date, due_date, category, weightage, result_released_date)

  All non-key attributes depend on directly to assessment_id so there are no transitive keys.

- student_module_assessment(**student_id,module_code, assessment_id,** result_code, obtained_mark, grade, attempt_no, submission_status)

The grade is determined by obtained_mark and obtained_mark is dependent on result_code which in turn is dependent on composite keys student_id, module_code and assessment_id so there is a transitive dependency of student_id, module_code, assessment_id → result_code → obtained_mark and obtained_mark → grade so A→B B→C.

student_module_assessment(**student_id,module_code, assessment_id, result_code*,**submission_status, attempt_no)

result_details(**result_code**, **obtained_mark***)

grade_details(**obtained_mark,** grade)

- resource_details(**resource_id**,resource_title,resource_type, resource_duration)

All non-key attributes directly depend on resource_id so there is no presence of transitive dependency.

Some tables do not need to be checked for transitive dependencies as they do not fit in with the criteria of more than **one non-key attribute**. The below mentioned tables are student_module, teacher_module_student,

- student_module(**student_id , module_code**, module_level)
- teacher_module_student(**teacher_id, mod_code, student_id**)
- student_module_teacher_announcement(**student_id,module_code, teacher_id, announcement_id**)
- student_module_resource(**resource_id, student_id, module_code,** completion_status)

The final tables after 3NF are:

- student_details(**student_id**,student_name,student_address, student_contact,student_email, student_gender, enrolled_date, **program_id***)
- program_details(**program_id**,program_title,program_length, program_credit,program_fee,entry_requirement,max_capacity, mode_of_delivery)
- module_details(**module_code**,module_title,module_credit, module_duration)
- student_module(**student_id , module_code**, module_level)
- teacher_details(**teacher_id**,teacher_name,teacher_address,teacher_contact, teacher_educational_background, teacher_experience, salary, joined_date)
- teacher_module_student(**teacher_id, module_code, student_id**)
- announcement_details(**announcement_id**,announcement_headline, uploaded_date, expired_date)
- student_module_teacher_announcement(**student_id,module_code, teacher_id, announcement_id**)
- assessment_details(**assessment_id**,assessment_title,assigned_date, due_date, category, weightage, result_released_date)
- student_module_assessment(**student_id,module_code,assessment_id, result_code*,**submission_status, attempt_no)
- result_details(**result_code**, **obtained_mark***)
- result_grade (**obtained_mark,** grade)
- resource_details(**resource_id**,resource_title,resource_type, resource_duration)
- student_module_resource(**resource_id,student_id,module_code,** completion_status)

6. DATA DICTIONARY

After normalization, each data element like title, type, attributes of the entities are stored in the tabular form as meta data of the database. The **data dictionary** also helps to locate relationships between entities (geeks, 2024).

6.1. META DATA

Table → Students

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	student_id	Character	7	Primary key	Unique identifier for each student
2	first_name	Character	15	Not null	First name of student
3	middle_name	Character	15		Optional middle name of student
4	last_name	Character	15	Not null	Last name of student
5	home_address	Character	30	Not null	Residence of student
6	contact_no	Number	10	Unique	Numeric value representing phone number
7	email_address	Character	15	Unique	Mailing address of student
8	gender	Character	6	Not null	Gender of student
9	age	Number	2	Not null	Numeric value representing age of students
10	enrolled_date	Date		Not null	Date when student enrolled

11	program_id	Character	7	Foreign key	Unique identifier for each program

Table 5 Table: Students

Where **previous student_name** = first_name + middle_name + last_name.

Table → Programs

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	program_id	Character	7	Primary key	Unique identifier for each program
2	program_title	Character	15	Not null	Name of the program
3	program_length	Number	5	Not null	duration to complete the program
4	total_credit	Number	5	Not null	Overall numeric value for credit of program
5	tuition_fee	Number	20	Not null	Total amount to study the program
6	entry_requirement	Character	15	Not null	criteria to take the program
7	maximum_capacity	Number	20	Not null	Numeric value for total program intake
8	Mode_of_delivery	Character	15	Not null	Medium to study the program

Table 6 Table: Programs

Table → Modules

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	module_code	Character	7	Primary key	Unique identifier of module
2	module_title	Character	15	Unique	Name of module
3	credit_score	Number	5	Not null	Credit of module
4	duration_period	Number	5	Not null	Duration to complete each module

Table 7 Table: Modules

Table → student_module

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	student_id	Character	7	Foreign key	Reference to unique identifier of student table's student_id
2	module_code	Character	7	Foreign key	Reference to unique identifier of module table's module_code
3	module_level	Number	5	Not null	Year level category of module

Table 8 Relation Table: student_module

Table → Teachers

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	teacher_id	Character	7	Primary key	Unique identifier of teachers
2	first_name	Character	15	Not null	First name of teacher
3	middle_name	Character	15		Optional middle name of teacher
4	last_name	Character	15	Not null	last name of teacher
5	home_address	Character	30	Not null	Residence of teacher
6	contact_no	Number	10	Unique	Numeric value representing phone number
7	educational_background	Character	25	Not null	Mailing address of teacher
8	years_of_experience	Number	5	Not null	Numeric value representing working experience of teacher
9	salary	Number	20	Not null	Numeric value representing salary of teacher
10	joined_date	Date		Not null	Date when teacher joined

Table 9 Table: Teachers

Table → student_module_teacher

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION	COMPOSITE CONSTRAINT
1	student_id	Number	7	Foreign key	Reference to unique identifier of student table's student_id	Primary key
2	module_code	Number	7	Foreign key	Reference to unique identifier of module table's module_code	
3	teacher_id	Number	7	Foreign key	Reference to unique identifier of teacher table's teacher_id	

Table 10 Relation Table: student_module_teacher

Table → Announcements

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	announcement_id	Character	8	Primary key	Unique identifier of announcement
2	notice_headline	Character	100	Not null	subject of announcement
3	uploaded_date	Date		Not null	Date when uploaded
4	expired_date	Date		Not null	Date when expired

Table 11 Relation Table: Announcements

Table → student_module_teacher_announcement

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION	COMPOSITE CONSTRAINT
1	student_id	Character	7	Foreign key	Reference to unique identifier of student table's student_id	Primary key
2	module_code	Character	7	Foreign key	Reference to unique identifier of module table's module_code	
3	teacher_id	Character	7	Foreign key	Reference to unique identifier of teacher table's teacher_id	
4	announcement_id	Character	8	Foreign key	Reference to unique identifier of announcement table's announcement_id	

Table 12 Relation Table: student_module_teacher_announcement

Table → Assessments

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	assessment_id	Character	8	Primary key	Unique identifier of assessment
2	assessment_title	Character	20	Not null	Name of assessment
3	assigned_date	Date		Not null	Date when assigned
4	due_date	Date		Not null	Deadline of assessment
5	category	Character	10	Not null	Type of assessment
6	weightage	Number	5	Not null	Numeric value for percentage of assessment
7	released_date	Date		Not null	Released date of result for assessment

Table 13 Table: Assessments

Table → Results

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	result_code	Character	8	Primary key	Unique identifier of result
2	obtained_mark	Number	5	Foreign key	References result_grade table obtained_mark

Table 14 Table: Results

Table → result_grade

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	obtained_mark	Number	5	Primary key	Assessment mark of student
2	grade	Character	5	Not null	Grade marked by obtained mark

Table 15 Relation Table: result_grade

Table → student_module_assessment

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION	COMPOSITE CONSTRAINT
1	student_id	Character	7	Foreign key	Reference to unique identifier of student table's student_id	Primary key
2	module_code	Character	7	Foreign key	Reference to unique identifier of module table's module_code	
3	assessment_id	Character	8	Foreign key	Reference to unique identifier of assessment table's assessment_id	
4	result_code	Character	8	Foreign key	Reference to unique identifier of result table's result_code	
5	Submission_status	Character	10	Not null	Assessment status by student	

6	attempt_no	Number	5	Not null	Numeric value representing attempt made by student for an assessment	

Table 16 Relation Table: student_module_assessmen_resultt

Table → Resources

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION
1	resource_id	Character	8	Primary key	Unique identifier of resource
2	resource_title	Character	30	Not null	Name of resource
3	format_type	Character	15	Not null	Type of resource
4	duration	Number	5	Not null	Duration of resource

Table 17 Table: Resources

Table → student_module_resource

S.NO	ATTRIBUTE NAME	DATA TYPE	SIZE	CONSTRAINT	DESCRIPTION	COMPOSITE CONSTRAINT
1	student_id	Character	7	Foreign key	Reference to unique identifier of student table's student_id	Primary key
2	module_code	Character	7	Foreign key	Reference to unique identifier of module table's module_code	
3	resource_id	Character	8	Foreign key	Reference to unique identifier of resource table's resource_id	
4	completion_status	Character	15	Not null	Resource completion status by student	

Table 18 Relation Table: student_module_resource

6.2. FINAL ER-DIAGRAM

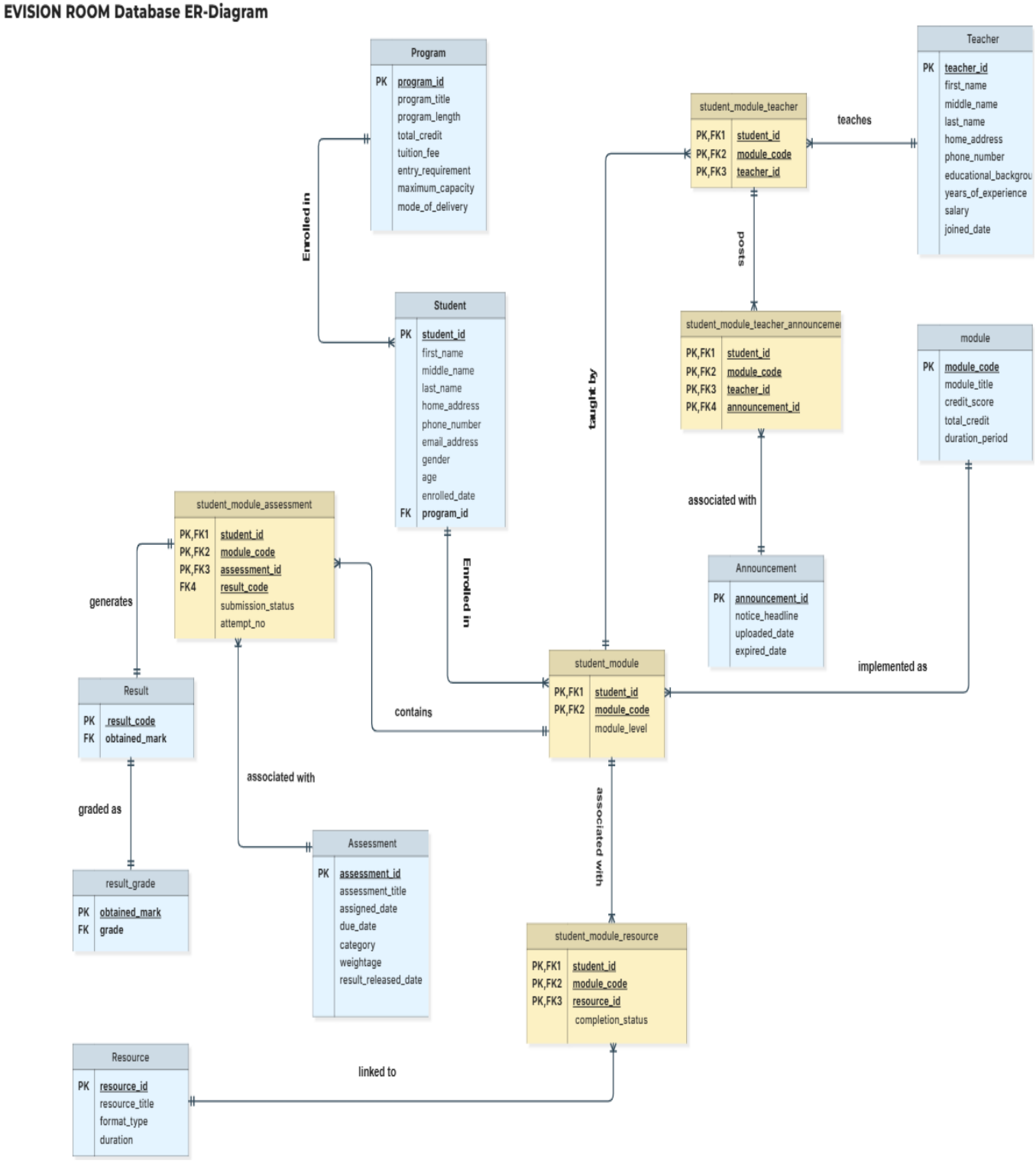


Figure 13 Final er diagram

NOTE: Blue is an individual entity whereas Yellow is connecting entities.

7. IMPLEMENTION

In this section, physical database design is done with the help of provided tools. The main operations are done which involve **SELECT, CREATE, UPDATE AND INSERT.** The final tables created after normalization is implemented into the “EVISION ROOM” database and each valid data are inserted into rows to main referential integrity.

7.1. SQL KEYWORD

Usage of predefined keywords can be seen while developing the physical design database. The main basic keywords are **CONNECT** and **RESOURCE** in **DCL** apart from other SQL languages like **DDL, DML, TCL** which helps in **REVOKE** and **GRANT** permissions. DDL involves keywords to create a structural base of table, DML provides keyword which helps to manipulate data in the table whereas TCL is to record a transaction done in transaction query.

7.2. INITIAL STEP FOR CREATION OF EVISION ROOM DATABASE

The creation of individual plus bridging entities table is done to avoid redundancy, increase data integrity and accuracy.

7.2.1. CREATE, GRANT AND CONNECT TO USER

The process of creating a user who handles the further steps of **table creation**, **data insertion** and **implementing transaction** queries need to be done by the system which is the in-between users in term of access to the system. The system then grants permission with the required sql keyword (**mentioned in 7.2 SQL KEYWORD)** and proceeds to the next step that is table creation.

PRE-WRITTEN ROUGH NOTEPAD:

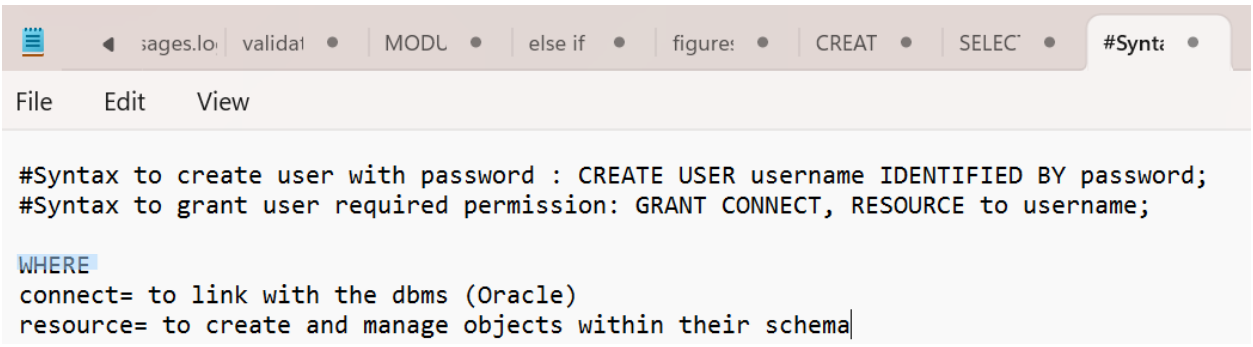


Figure 14 Screenshot: Notepad to grant permissions

ACTUAL IMPLEMENTATION:

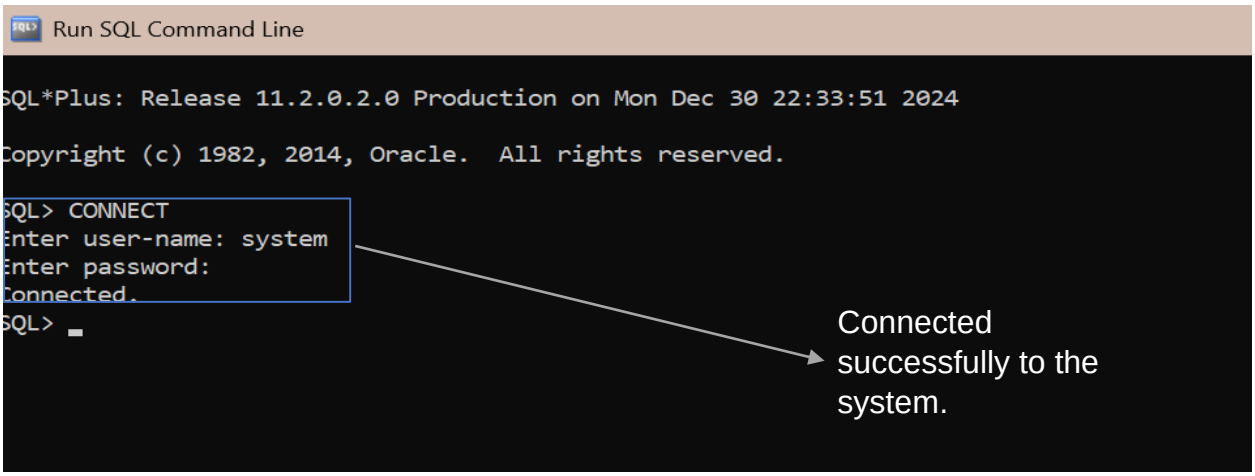


Figure 15 Screenshot: connect to system

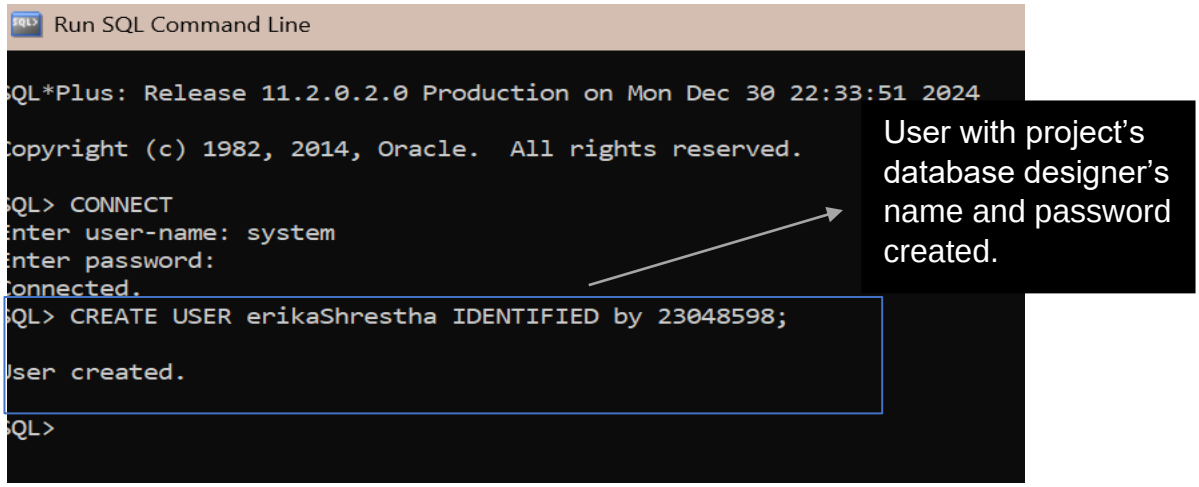


Figure 16 Screenshot: user and password creation

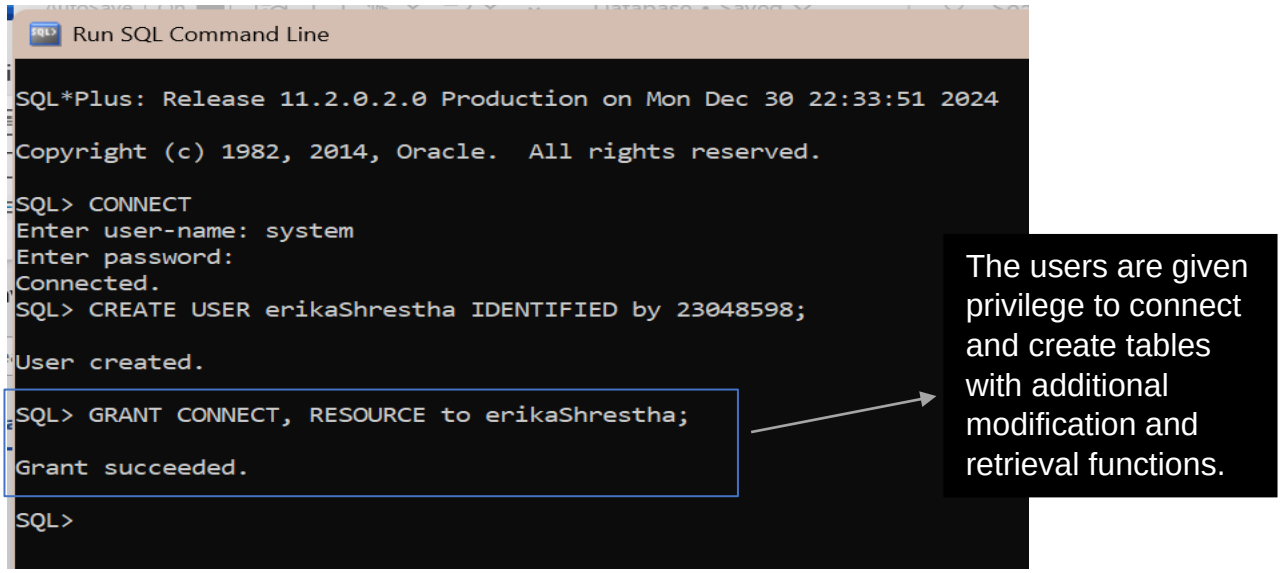


Figure 17 Screenshot: grant permission

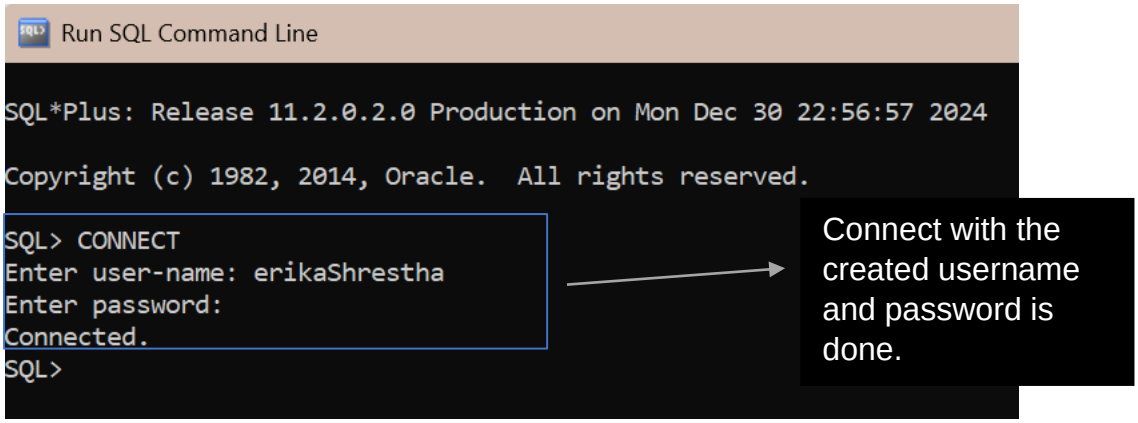


Figure 18 Screenshot: Connect to created user

7.2.2. REQUIRED ENTITIES TABLE CREATION

After connecting to the required user interface, we create the **structural base** of the tables needed for required entities and their relations with one another. This section involves the tables identified after **normalization 3NF** which is implemented with help of **creating syntax**.

PRE-WRITTEN ROUGH NOTEPAD:

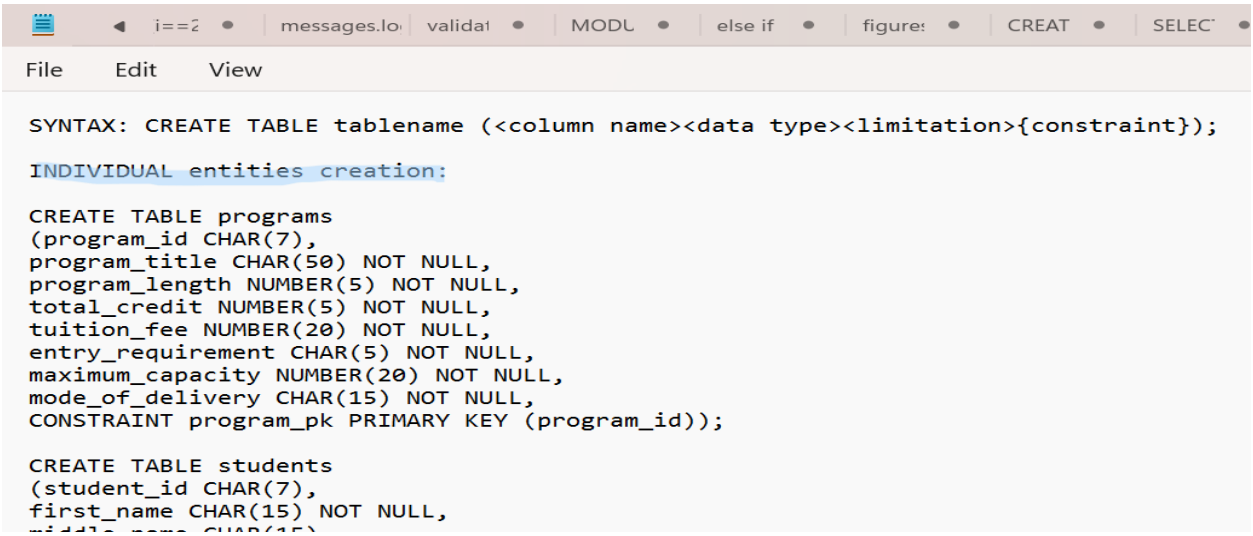


Figure 19 Screenshot: Notepad to create individual entity tables

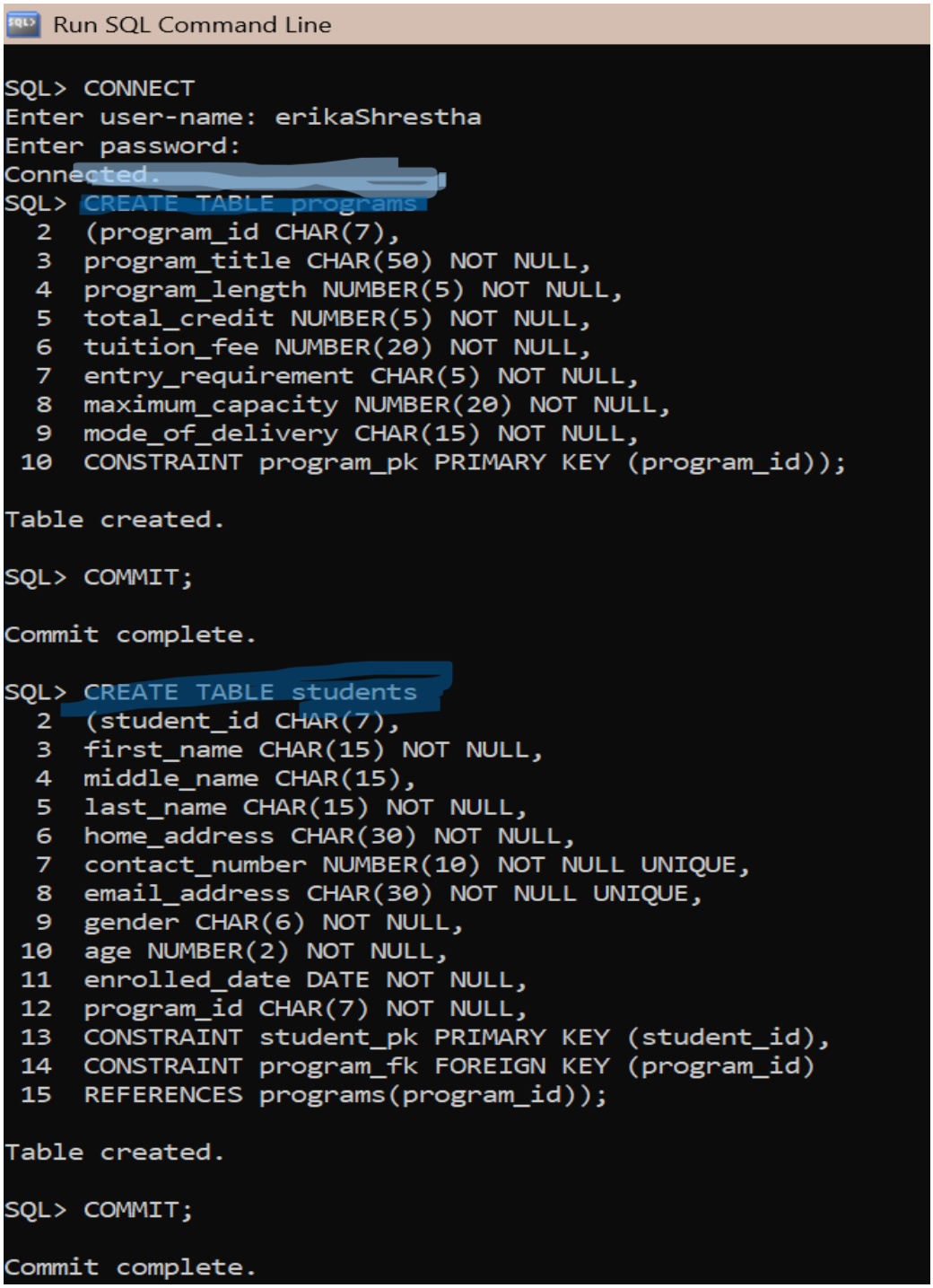


Figure 20 Screenshot: Create table programs and students

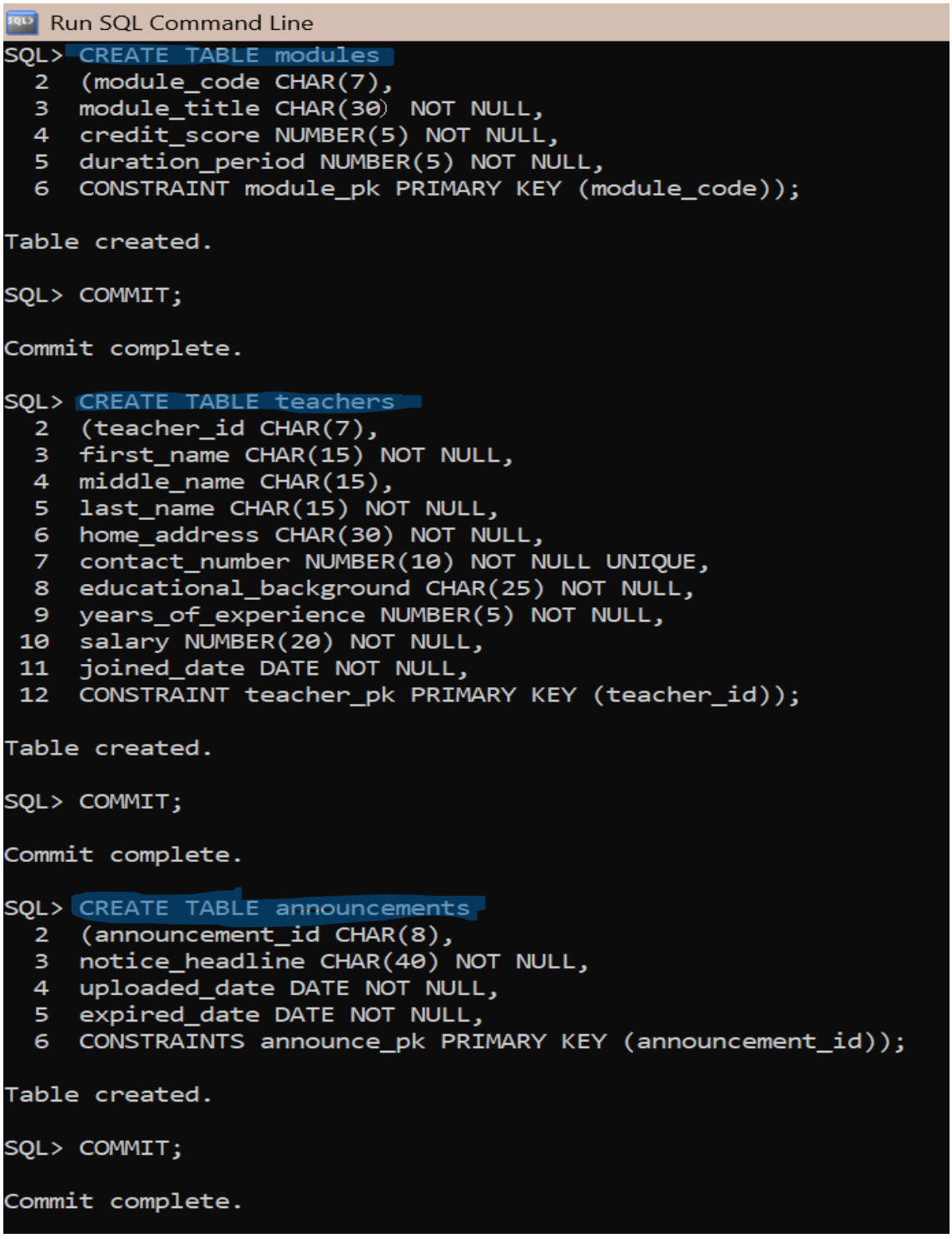


Figure 21 Screenshot: Create table modules, teachers and announcements

Run SQL Command Line

```
SQL> CREATE TABLE assessments
  2  (assessment_id CHAR(8),
  3  assessment_title CHAR(15) NOT NULL,
  4  assigned_date DATE NOT NULL,
  5  due_date DATE NOT NULL,
  6  category CHAR(10) NOT NULL,
  7  weightage NUMBER(5) NOT NULL,
  8  result_released_date DATE NOT NULL,
  9  CONSTRAINT assessment_pk PRIMARY KEY (assessment_id));

Table created.

SQL> COMMIT;

Commit complete.

SQL> CREATE TABLE resources
  2  (resource_id CHAR(8),
  3  resource_title CHAR(15) NOT NULL,
  4  format_type CHAR(10) NOT NULL,
  5  duration NUMBER(5) NOT NULL,
  6  CONSTRAINT resource_pk PRIMARY KEY (resource_id));

Table created.

SQL> COMMIT;

Commit complete.

SQL> CREATE TABLE result_grades
  2  (obtained_mark NUMBER(5),
  3  grade CHAR(5) NOT NULL,
  4  CONSTRAINT mark_pk PRIMARY KEY (obtained_mark));

Table created.

SQL> COMMIT;

Commit complete.
```

Figure 22 Screenshot: create table assessments, resources and result_grades

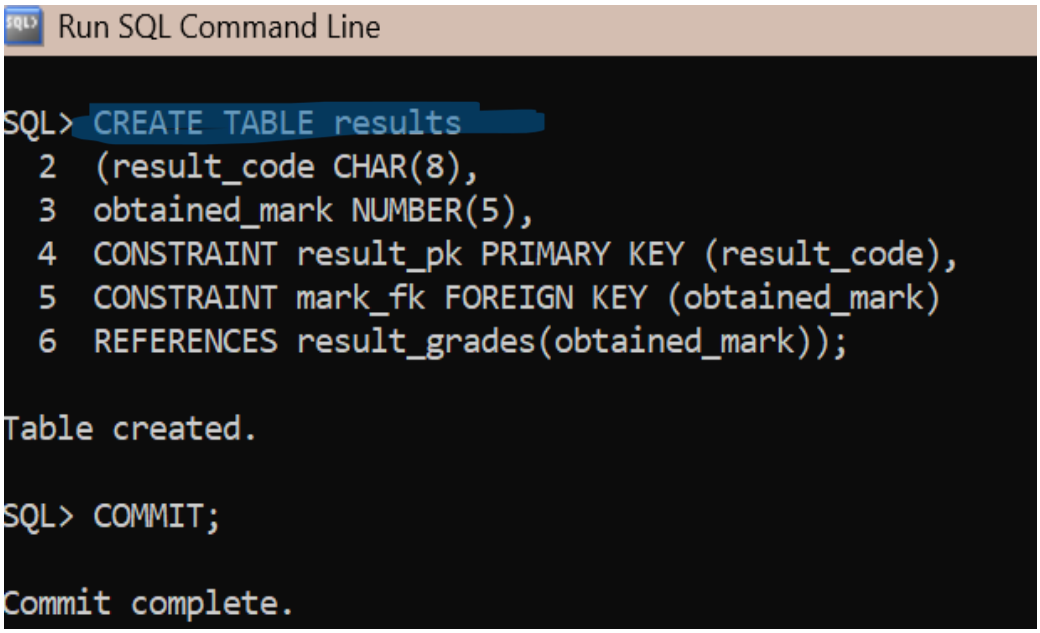


Figure 23 Screenshot: Create table results

PRE-WRITTEN ROUGH NOTEPAD:

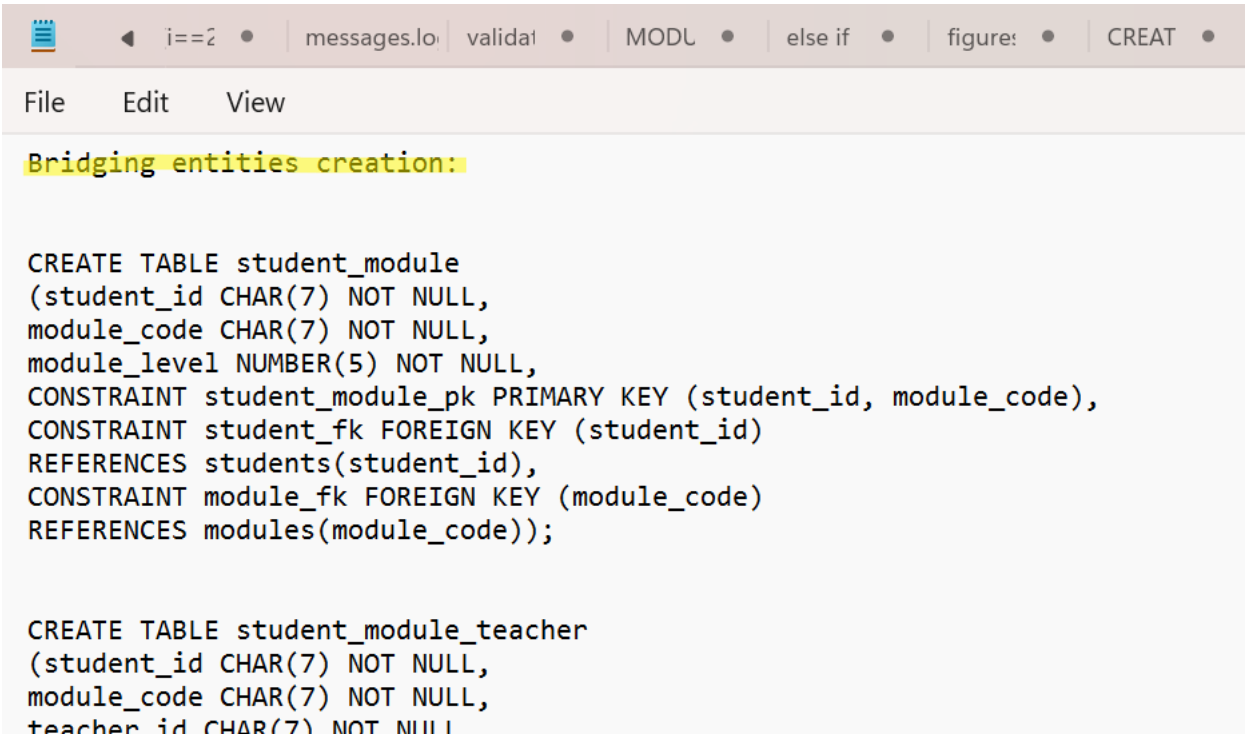


Figure 24 Screenshot: Notepad to create bridging entity tables

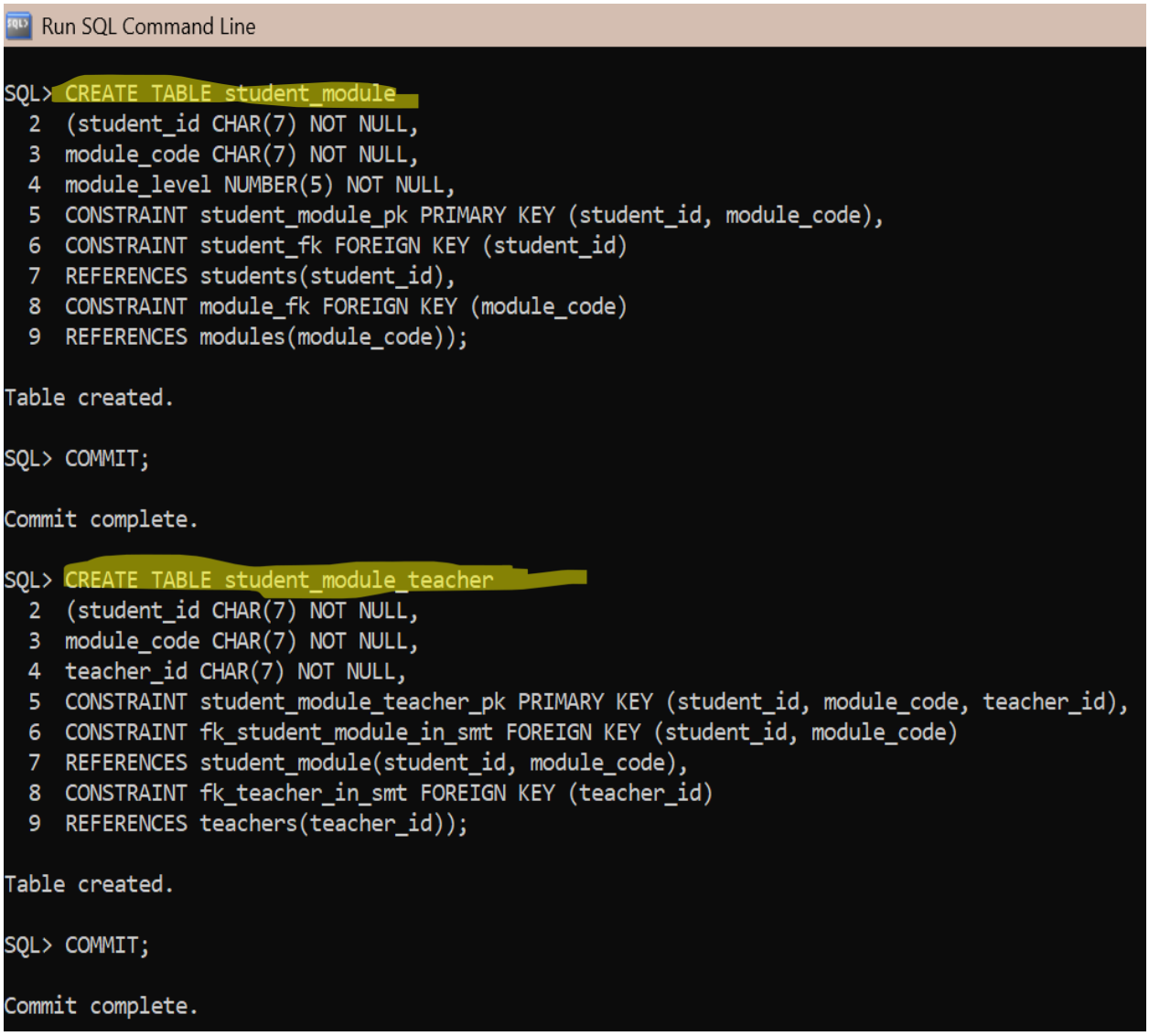


Figure 25 Screenshot: Create table student_module and student_module_teacher

Run SQL Command Line

```
SQL> CREATE TABLE smt_announcement
  2  (student_id CHAR(7) NOT NULL,
  3  module_code CHAR(7) NOT NULL,
  4  teacher_id CHAR(7) NOT NULL,
  5  announcement_id CHAR(8) NOT NULL,
  6  CONSTRAINT smt_announcement_pk PRIMARY KEY (student_id, module_code, teacher_id, announcement_id),
  7  CONSTRAINT fk_smt_in_smt_announcement FOREIGN KEY (student_id, module_code, teacher_id)
  8  REFERENCES student_module_teacher(student_id, module_code, teacher_id),
  9  CONSTRAINT fk_announ_in_smt_announcement FOREIGN KEY (announcement_id)
 10  REFERENCES announcements(announcement_id));

Table created.

SQL> COMMIT;

Commit complete.

SQL> CREATE TABLE student_module_assessment
  2  (student_id CHAR(7) NOT NULL,
  3  module_code CHAR(7) NOT NULL,
  4  assessment_id CHAR(8) NOT NULL,
  5  result_code CHAR(8) NOT NULL,
  6  submission_status CHAR(20) NOT NULL,
  7  attempt_number NUMBER(5) NOT NULL,
  8  CONSTRAINT sm_assessment_pk PRIMARY KEY (student_id, module_code, assessment_id),
  9  CONSTRAINT fk_sm_in_sm_assess FOREIGN KEY (student_id, module_code)
 10  REFERENCES student_module(student_id, module_code),
 11  CONSTRAINT fk_assessment_in_sm_assess FOREIGN KEY (assessment_id)
 12  REFERENCES assessments(assessment_id),
 13  CONSTRAINT fk_result_in_sm_assess FOREIGN KEY (result_code)
 14  REFERENCES results(result_code));

Table created.

SQL> COMMIT;

Commit complete.
```

Figure 26 Screenshot: Create table student_module_teacher_announcement and student_module_assessment

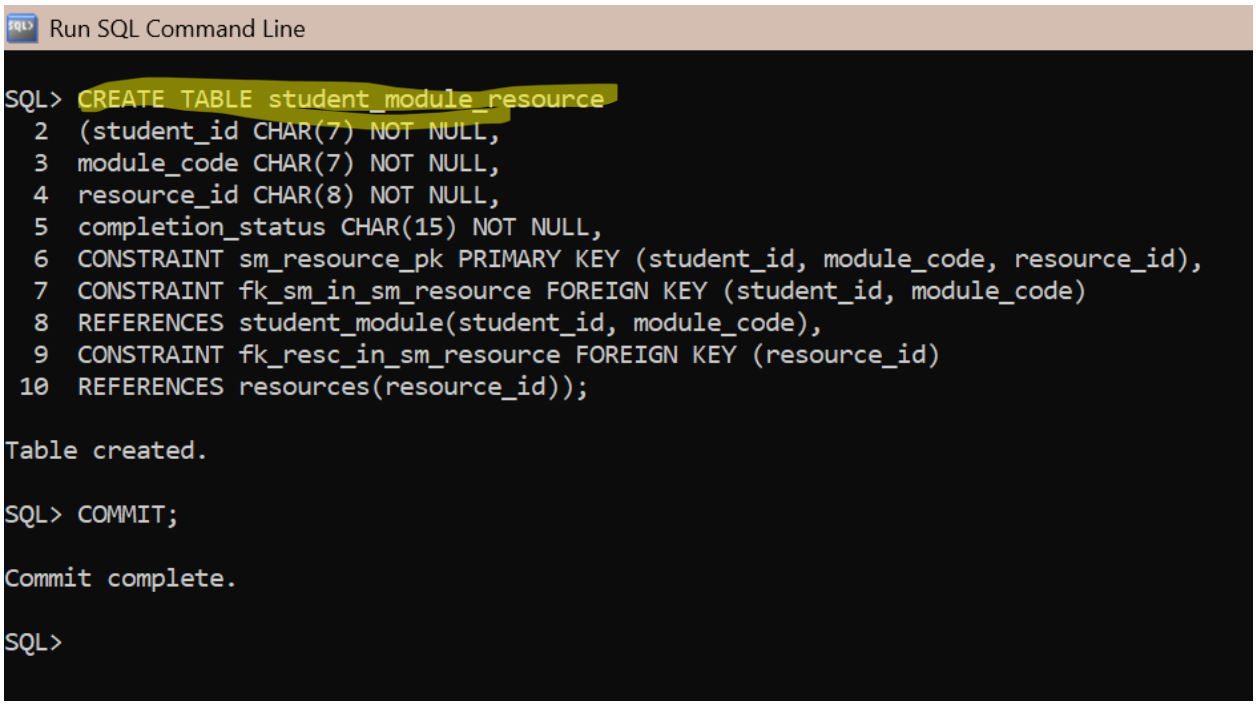


Figure 27 Screenshot: Create table: student_module_resource

Finalized all tables:

PRE-WRITTEN ROUGH NOTEPAD:

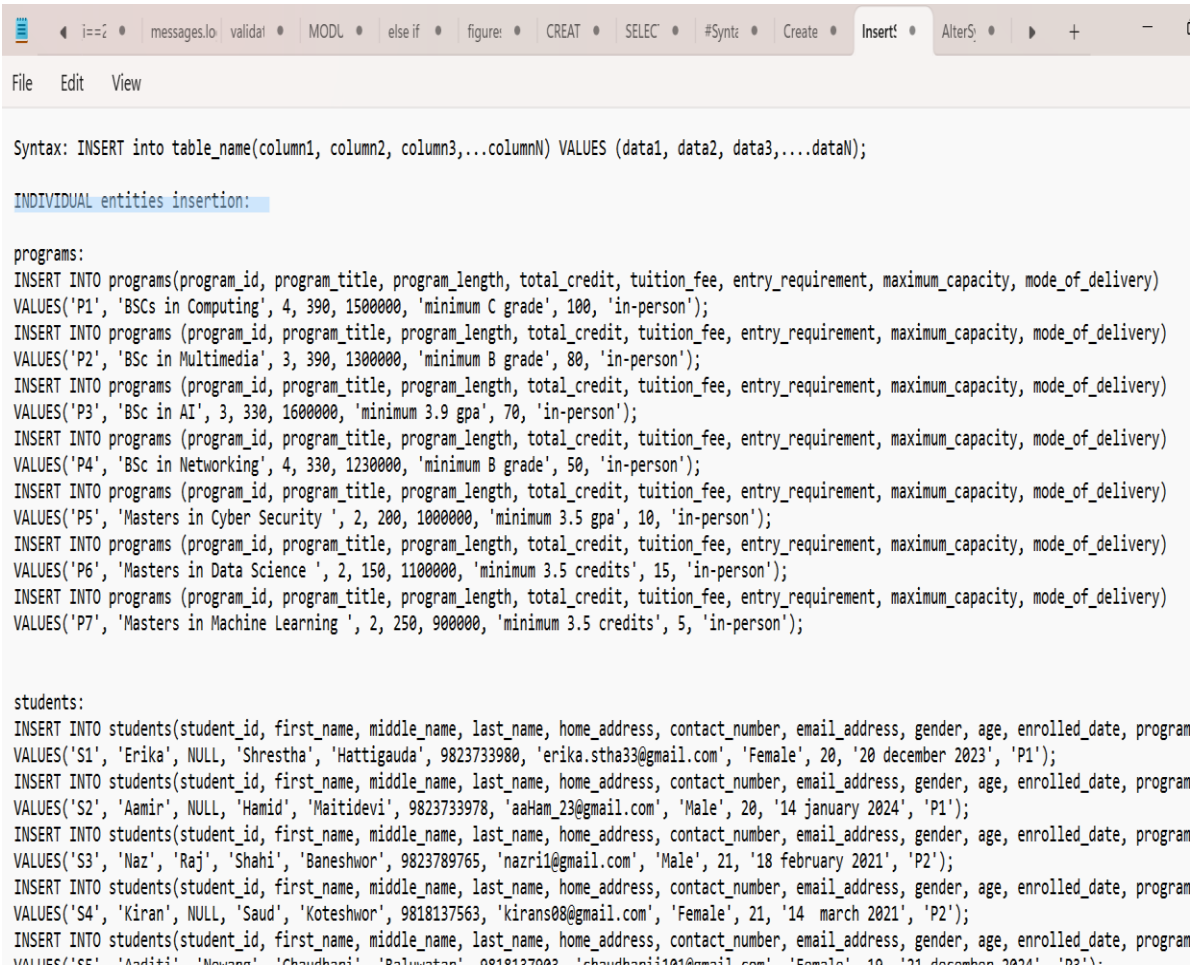
```
SELECT table_name FROM user_tables;
```

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
------------------------------
PROGRAMS
STUDENTS
MODULES
TEACHERS
ANNOUNCEMENTS
ASSESSMENTS
RESOURCES
RESULT_GRADES
RESULTS
STUDENT_MODULE
STUDENT_MODULE_TEACHER
SMT_ANNOUNCEMENT
STUDENT_MODULE_ASSESSMENT
STUDENT_MODULE_RESOURCE

14 rows selected.
```

Figure 28 Screenshot: Finalized implemented entities

7.2.3. INSERT ROWS INTO TABLES

PRE-WRITTEN ROUGH NOTEPAD:



File Edit View

```
Syntax: INSERT into table_name(column1, column2, column3,...columnN) VALUES (data1, data2, data3,....dataN);

INDIVIDUAL entities insertion:

programs:
INSERT INTO programs(program_id, program_title, program_length, total_credit, tuition_fee, entry_requirement, maximum_capacity, mode_of_delivery)
VALUES('P1', 'BSCs in Computing', 4, 390, 1500000, 'minimum C grade', 100, 'in-person');
INSERT INTO programs (program_id, program_title, program_length, total_credit, tuition_fee, entry_requirement, maximum_capacity, mode_of_delivery)
VALUES('P2', 'BSc in Multimedia', 3, 390, 1300000, 'minimum B grade', 80, 'in-person');
INSERT INTO programs (program_id, program_title, program_length, total_credit, tuition_fee, entry_requirement, maximum_capacity, mode_of_delivery)
VALUES('P3', 'BSc in AI', 3, 330, 1600000, 'minimum 3.9 gpa', 70, 'in-person');
INSERT INTO programs (program_id, program_title, program_length, total_credit, tuition_fee, entry_requirement, maximum_capacity, mode_of_delivery)
VALUES('P4', 'BSc in Networking', 4, 330, 1230000, 'minimum B grade', 50, 'in-person');
INSERT INTO programs (program_id, program_title, program_length, total_credit, tuition_fee, entry_requirement, maximum_capacity, mode_of_delivery)
VALUES('P5', 'Masters in Cyber Security ', 2, 200, 1000000, 'minimum 3.5 gpa', 10, 'in-person');
INSERT INTO programs (program_id, program_title, program_length, total_credit, tuition_fee, entry_requirement, maximum_capacity, mode_of_delivery)
VALUES('P6', 'Masters in Data Science ', 2, 150, 1100000, 'minimum 3.5 credits', 15, 'in-person');
INSERT INTO programs (program_id, program_title, program_length, total_credit, tuition_fee, entry_requirement, maximum_capacity, mode_of_delivery)
VALUES('P7', 'Masters in Machine Learning ', 2, 250, 900000, 'minimum 3.5 credits', 5, 'in-person');


students:
INSERT INTO students(student_id, first_name, middle_name, last_name, home_address, contact_number, email_address, gender, age, enrolled_date, program
VALUES('S1', 'Erika', NULL, 'Shrestha', 'Hattigauda', 9823733980, 'erika.stha33@gmail.com', 'Female', 20, '20 december 2023', 'P1');
INSERT INTO students(student_id, first_name, middle_name, last_name, home_address, contact_number, email_address, gender, age, enrolled_date, program
VALUES('S2', 'Aamir', NULL, 'Hamid', 'Maitidevi', 9823733978, 'aaHam_23@gmail.com', 'Male', 20, '14 january 2024', 'P1');
INSERT INTO students(student_id, first_name, middle_name, last_name, home_address, contact_number, email_address, gender, age, enrolled_date, program
VALUES('S3', 'Naz', 'Raj', 'Shahi', 'Baneshwor', 9823789765, 'nazri1@gmail.com', 'Male', 21, '18 february 2021', 'P2');
INSERT INTO students(student_id, first_name, middle_name, last_name, home_address, contact_number, email_address, gender, age, enrolled_date, program
VALUES('S4', 'Kiran', NULL, 'Saud', 'Koteshwor', 9818137563, 'kirans08@gmail.com', 'Female', 21, '14  march 2021', 'P2');
INSERT INTO students(student_id, first_name, middle_name, last_name, home_address, contact_number, email_address, gender, age, enrolled_date, program
```

Figure 29 Screenshot: Notepad to individual entities insertion

TABLE students:

PRE-WRITTEN ROUGH NOTEPAD:

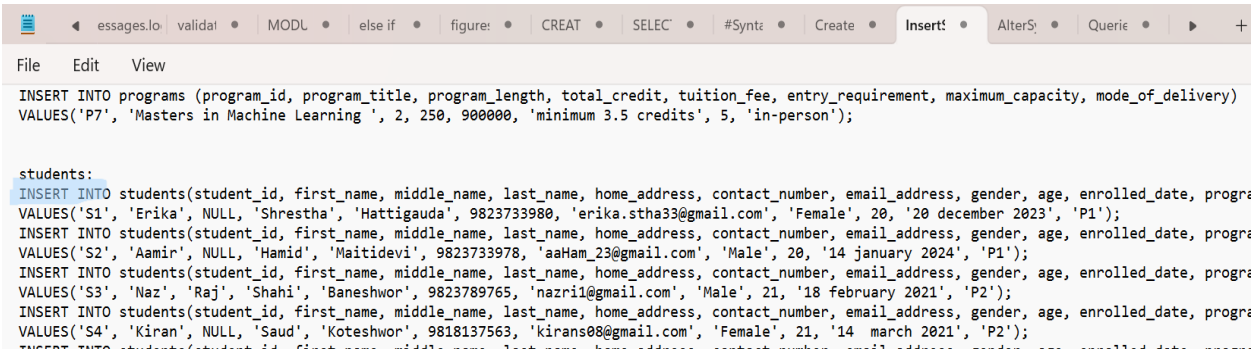


Figure 30 Screenshot: Notepad to insert into students

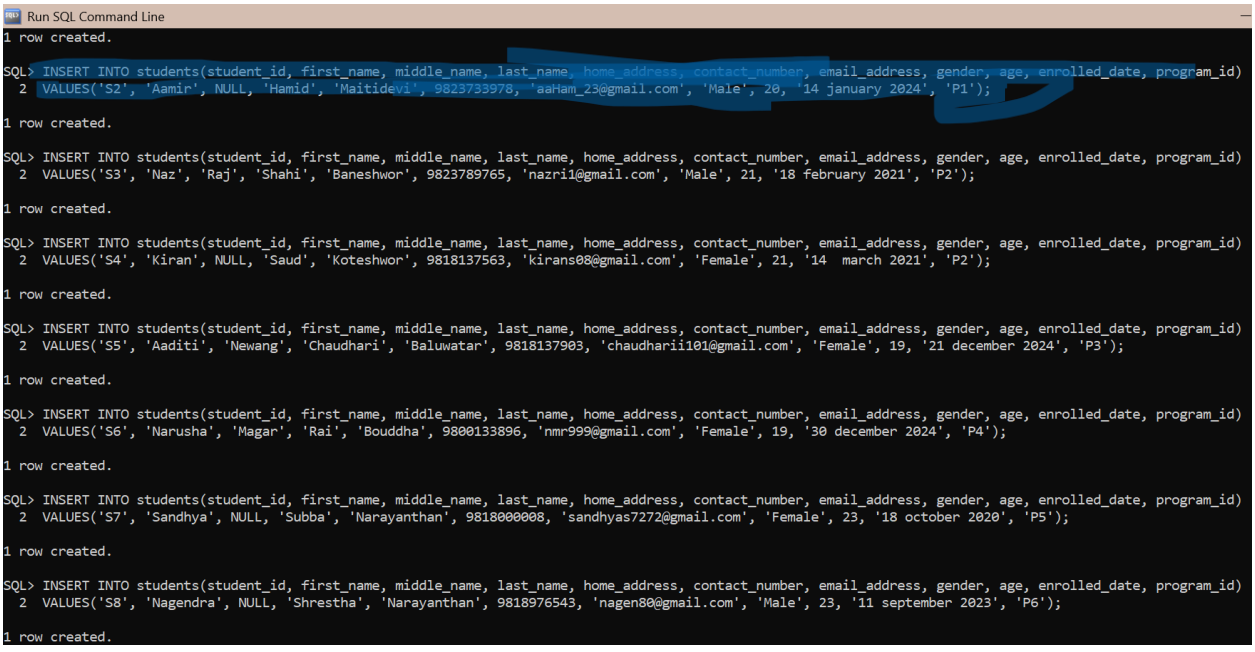


Figure 31 Screenshot: Actual implementation of students insertion

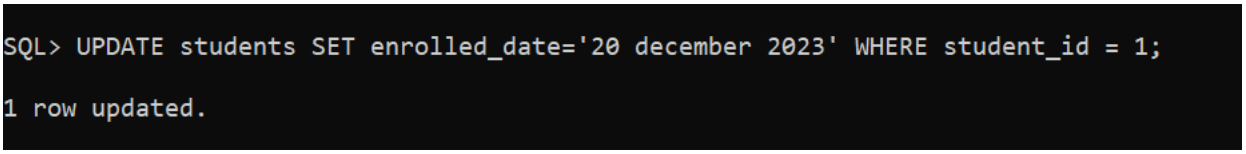


Figure 32 Screenshot: update student values

TABLE programs:

PRE-WRITTEN ROUGH NOTEPAD:

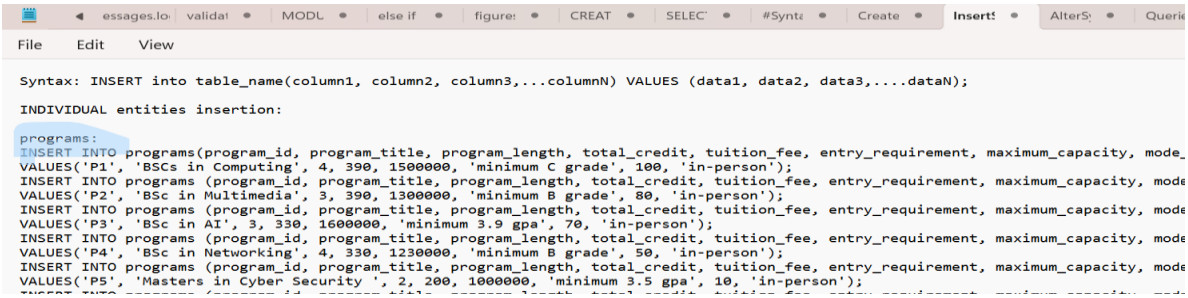


Figure 33 Screenshot: Notepad to insert program values

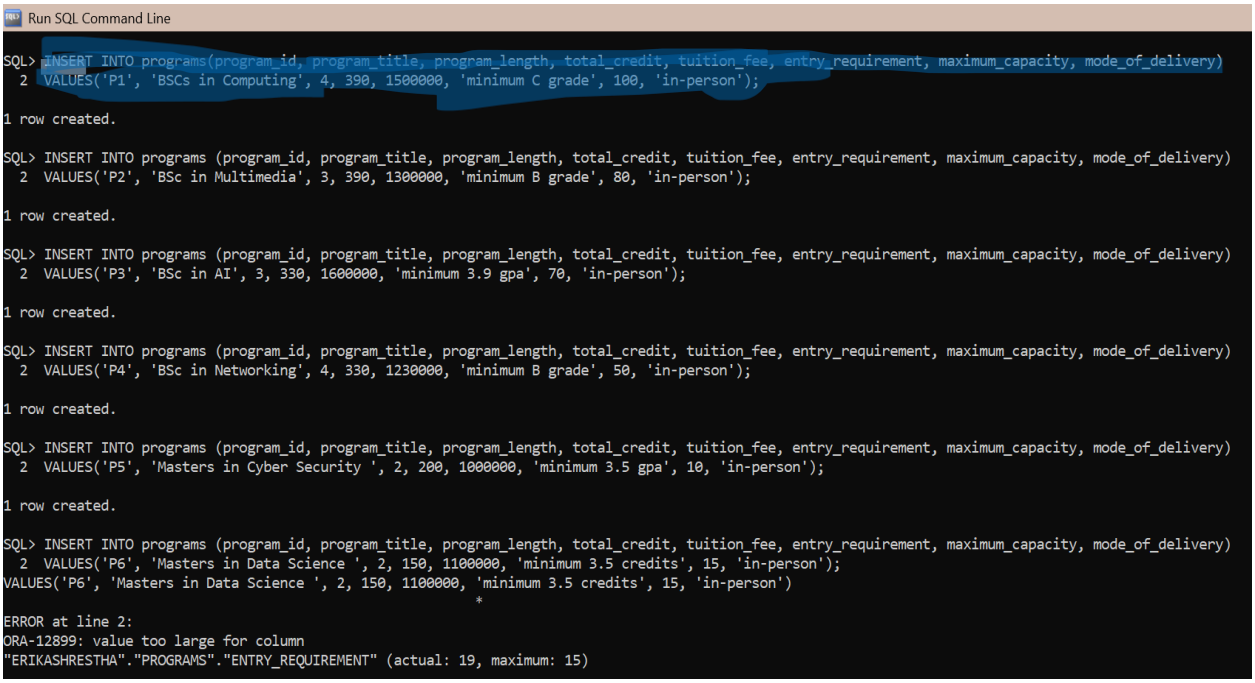


Figure 34 Screenshot: Actual implementation of programs insertion

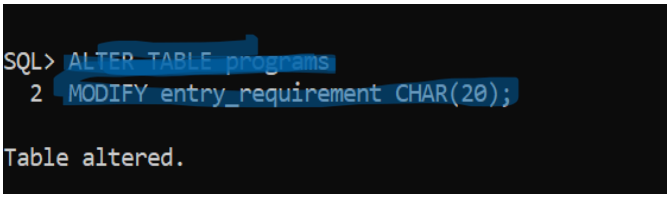


Figure 35 Screenshot: Alter program columns

TABLE modules:

PRE-WRITTEN ROUGH NOTEPAD:

```
Modules:
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M1', 'Further Calculus', 15, 6);
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M2', 'Probability Statistics', 15, 6);
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M3', 'Database', 15, 6);
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M4', 'Data structure', 30, 12);
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M5', 'Java programming', 30, 12);
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M6', 'Software Engineering', 30, 12);
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M7', 'Information system', 15, 6);
INSERT INTO modules(module_code, module_title, credit_score, duration_period)
VALUES('M8', 'Fundamentals of computing', 15, 6);
```

Figure 36 Screenshot: Notepad to insert modules values

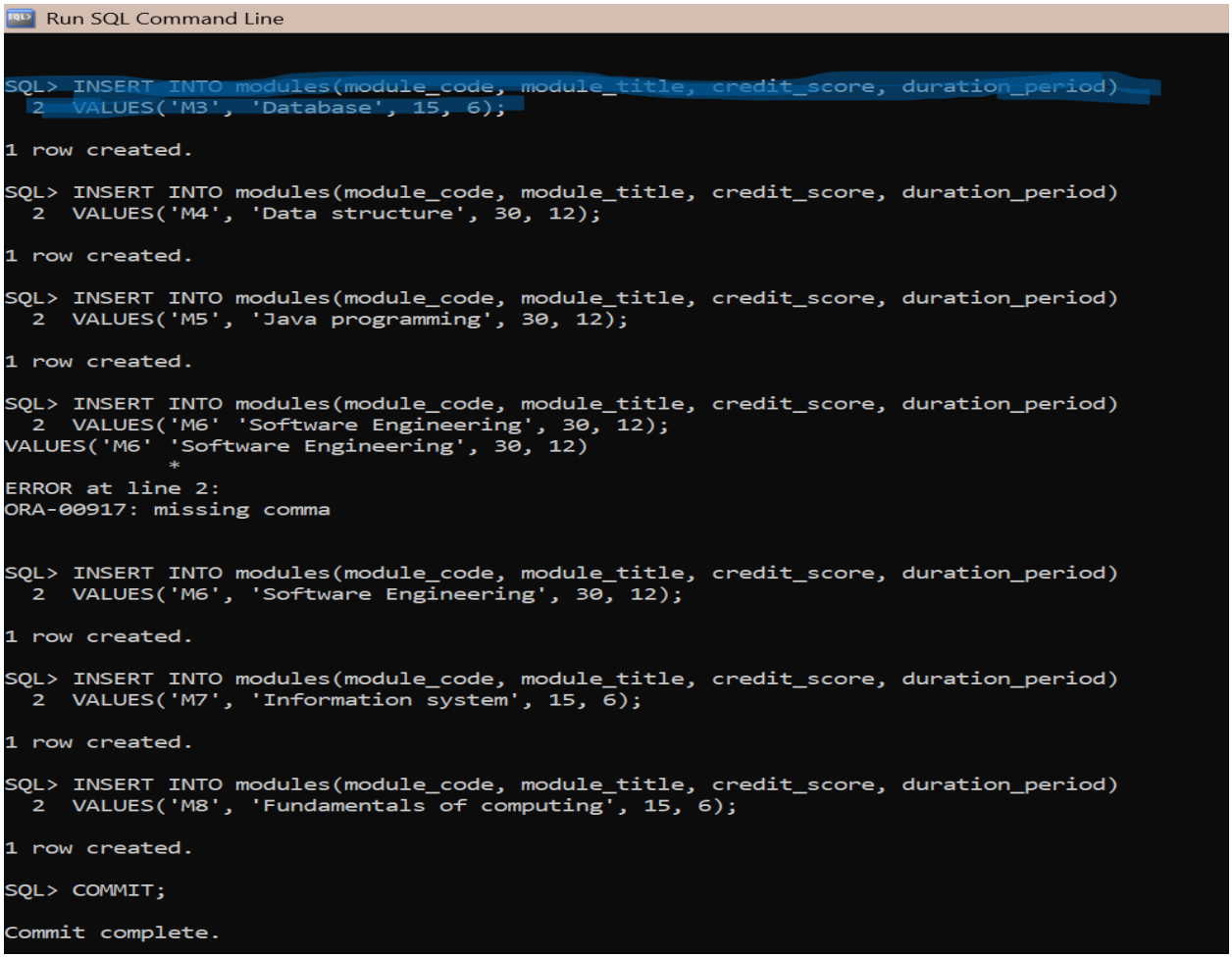


Figure 37 Screenshot: Actual implementation of modules insertion

TABLE **teachers:**

PRE-WRITTEN ROUGH NOTEPAD:

```
teachers:
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T1', 'Anuja', 'Singh', 'Bhattarai', 'Nagarkot', 9823738392, 'bIT', 5, 20000, '4 january 2023');
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T2', 'Prithvi',NULL, 'Maharjan', 'Basantapur', 9823738391, 'bsc in computing', 2, 30000, '20 january 2022');
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T3', 'Deepesh', 'raj', 'adhikari', 'Budhanilkantha', 9823738310, 'bsc in data analyst', 4, 35000, '16 june 2021');
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T4', 'Indra', NULL, 'Dhakal', 'Budhanilkantha', 9823738312, 'bsc in mathematics', 3,25000, '1 july 2023');
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T5', 'Sushil', 'Prasad', 'Koirala', 'Bouddha', 9823789034, 'bsc in mathematics', 4,40000, '1 december 2020');
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T6', 'Rubin', NULL, 'Thapa', 'Baluwatar', 9823789033, 'bsc in ecommerce', 6,50000, '28 october 2024');
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T7', 'Prasika',NULL, 'Kc', 'Baneshwor', 9823789055, 'bsc in IoT', 1,15000, '5 october 2020');
INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_date)
VALUES('T8', 'Numang',NULL, 'Gurung', 'Baneshwor', 9823789044, 'bsc in IoT', 3,20000, '5 october 2022');
```

Figure 38 Screenshot: Notepad to insert teachers values

```
Run SQL Command Line

Commit complete.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T1', 'Anuja', 'Singh', 'Bhattarai', 'Nagarkot', 9823738392, 'bIT', 5, 20000, '4 january 2023');

1 row created.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T2', 'Prithvi',NULL, 'Maharjan', 'Basantapur', 9823738391, 'bsc in computing', 2, 30000, '20 january 2022');

1 row created.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T3', 'Deepesh', 'raj', 'adhikari', 'Budhanilkantha', 9823738310, 'bsc in data analyst', 4, 35000, '16 june 2021');

1 row created.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T4', 'Indra', NULL, 'Dhakal', 'Budhanilkantha', 9823738312, 'bsc in mathematics', 3,25000, '1 july 2023');

1 row created.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T5', 'Sushil', 'Prasad', 'Koirala', 'Bouddha', 9823789034, 'bsc in mathematics', 4,40000, '1 december 2020');

1 row created.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T6', 'Rubin', NULL, 'Thapa', 'Baluwatar', 9823789033, 'bsc in ecommerce', 6,50000, '28 october 2024');

1 row created.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T7', 'Prasika',NULL, 'Kc', 'Baneshwor', 9823789055, 'bsc in IoT', 1,15000, '5 october 2020');

1 row created.

SQL> INSERT INTO teachers(teacher_id, first_name, middle_name, last_name, home_address, contact_number, educational_background, years_of_experience, salary, joined_
  2  VALUES('T8', 'Numang',NULL, 'Gurung', 'Baneshwor', 9823789044, 'bsc in IoT', 3,20000, '5 october 2022');

1 row created.

SQL> COMMIT;
```

Figure 39 Screenshot: Actual implementation of teachers insertion

TABLE announcements:

PRE-WRITTEN ROUGH NOTEPAD:

```
announcements:
INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
VALUES('AN1', 'Class postponed', '1 may 2025', '10 may 2025');
INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
VALUES('AN2', 'Class postponed', '18 january 2025', '11 february 2025');
INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
VALUES('AN3', 'Coursework rules', '8 march 2025', '10 march 2025');
INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
VALUES('AN4', 'Coursework rules', '8 march 2025', '10 march 2025');
INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
VALUES('AN5', 'College parking space', '8 september 2023', '12 november 2023');
```

Figure 40 Screenshot: Notepad to insert announcement values

```
Run SQL Command Line
SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN2', 'Class postponed', '18 january 2025', '11 february 2025');

1 row created.

SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN3', 'Coursework rules', '8 march 2025', '10 march 2025');

1 row created.

SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN4', 'Coursework rules', '8 march 2025', '10 march 2025');

1 row created.

SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN5', 'College parking space', '8 september 2023', '12 november 2023');

1 row created.

SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN6', 'College reopens', '1 november 2023', '12 november 2023');

1 row created.

SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN7', 'Milestone guidelines', '21 june 2024', '2 july 2024');

1 row created.

SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN8', 'Milestone guidelines', '21 august 2024', '2 september 2024');

1 row created.

SQL> INSERT INTO announcements(announcement_id, notice_headline, uploaded_date, expired_date)
  2  VALUES('AN9', 'upcoming event', '21 august 2025', '2 september 2025');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

Figure 41 Screenshot: Actual implementation of announcements insertion

TABLE assessments:

PRE-WRITTEN ROUGH NOTEPAD:

```
assessments:
INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
VALUES('AS1', 'invoice creation', '9 december 2024', '24 december 2024', 'individual', 30, '8 october 2025');
INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
VALUES('AS2', 'college app', '28 december 2024', '4 january 2025', 'individual', 40, '10 january 2025');
INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
VALUES('AS3', 'class test', '11 december 2024', '11 december 2024', 'group', 100, '30 january 2025');
INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
VALUES('AS4', 'srs', '20 december 2024', '1 january 2025', 'group', 20, '8 january 2025');
INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
VALUES('AS5', 'crud operations', '5 november 2023', '11 november 2023', 'individual', 60, '14 december 2023');
INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
VALUES('AS6', 'Dashboard creation', '5 november 2024', '11 november 2024', 'individual', 40, '14 december 2024');
INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
VALUES('AS7', 'searching algorithms', '15 november 2025', '21 november 2025', 'group', 40, '14 december 2025');
```

Figure 42 Screenshot: Notepad to insert assessment values

```
Run SQL Command Line
Table altered.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS1', 'invoice creation', '9 december 2024', '24 december 2024', 'individual', 30, '8 october 2025');

1 row created.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS2', 'college app', '28 december 2024', '4 january 2025', 'individual', 40, '10 january 2025');

1 row created.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS3', 'class test', '11 december 2024', '11 december 2024', 'group', 100, '30 january 2025');

1 row created.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS4', 'srs', '20 december 2024', '1 january 2025', 'group', 20, '8 january 2025');

1 row created.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS5', 'crud operations', '5 november 2023', '11 november 2023', 'individual', 60, '14 december 2023');

1 row created.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS6', 'Dashboard creation', '5 november 2024', '11 november 2024', 'individual', 40, '14 december 2024');

1 row created.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS7', 'searching algorithms', '15 november 2025', '21 november 2025', 'group', 40, '14 december 2025');

1 row created.

SQL> INSERT INTO assessments(assessment_id, assessment_title, assigned_date, due_date, category, weightage, result_released_date)
  2  VALUES('AS8', 'database creation', '15 november 2015', '21 november 2015', 'individual', 70, '24 december 2015');

1 row created.

SQL> COMMIT;
```

Figure 43Screenshot: Actual implementation of assessments insertion

```
SQL> ALTER TABLE assessments
  2  MODIFY assessment_title CHAR(20);

Table altered.
```

Figure 44 Alter assessment columns

TABLE result_grade:

PRE-WRITTEN ROUGH NOTEPAD:

```
result_grades:
INSERT INTO result_grades(obtained_mark, grade)
VALUES (0, 'F');
INSERT INTO result_grades(obtained_mark, grade)
VALUES (10, 'F');
INSERT INTO result_grades(obtained_mark, grade)
VALUES (20, 'F');
```

Figure 45 Screenshot: Notepad to insert results_grade values

```
SQL> INSERT INTO result_grades(obtained_mark, grade)
  2  VALUES (40, 'P');

1 row created.

SQL> INSERT INTO result_grades(obtained_mark, grade)
  2  VALUES (50, 'D' );

1 row created.

SQL> INSERT INTO result_grades(obtained_mark, grade)
  2  VALUES (60, 'C');

1 row created.

SQL> INSERT INTO result_grades(obtained_mark, grade)
  2  VALUES (70, 'B');

1 row created.

SQL> INSERT INTO result_grades (obtained_mark, grade)
  2  VALUES (80, 'A');

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 46 Screenshot: Actual implementation of results_grade insertion

TABLE: Results

PRE-WRITTEN ROUGH NOTEPAD:

```
results:
INSERT INTO results(result_code, obtained_mark)
VALUES ('RR01', 50);
INSERT INTO results(result_code, obtained mark)
VALUES ('RR02', 50);
INSERT INTO results(result_code, obtained_mark)
VALUES ('RR03', 60);
INSERT INTO results(result_code, obtained_mark)
```

Figure 47 Screenshot: Notepad to insert into results values

```
SQL> INSERT INTO results(result_code, obtained_mark)
  2  VALUES ('RR03', 60);

1 row created.

SQL> INSERT INTO results(result_code, obtained_mark)
  2  VALUES('RR04', 70);

1 row created.

SQL> INSERT INTO results(result_code, obtained_mark)
  2  VALUES ('RR05', 80);

1 row created.

SQL> INSERT INTO results(result_code, obtained_mark)
  2  VALUES ('RR06', 60);

1 row created.

SQL> INSERT INTO results(result_code, obtained_mark)
  2  VALUES ('RR07', 80);

1 row created.

SQL> INSERT INTO results(result_code, obtained_mark)
  2  VALUES ('RR08', 70);

1 row created.

SQL> INSERT INTO results(result_code, obtained_mark)
  2  VALUES ('RR09', 40);

1 row created.

SQL> COMMIT;
```

Figure 48 Screenshot: Actual implementation of results insertion

TABLE resources:

PRE-WRITTEN ROUGH NOTEPAD:

```
resources:
INSERT INTO resources(resource_id, resource_title, format_type, duration)
VALUES ('R1', 'Introduction to Database', 'pdf', 100);
INSERT INTO resources(resource_id, resource_title, format_type, duration)
VALUES ('R2', 'Intro to Programming', 'mp4', 1);
INSERT INTO resources(resource_id, resource_title, format_type, duration)
VALUES ('R3', 'Variables and Data Types', 'documentation', 140);
```

Figure 49 Screenshot: Notepad to insert into resource values

```
Run SQL Command Line
                                        *
ERROR at line 2:
ORA-12899: value too large for column "ERIKASHRESTHA"."RESOURCES"."FORMAT_TYPE" (actual: 13, maximum: 10)


SQL> ALTER TABLE resources
  2  MODIFY format_type CHAR(15);

Table altered.

SQL> INSERT INTO resources(resource_id, resource_title, format_type, duration)
  2  VALUES ('R3', 'Variables and Data Types', 'documentation', 140);

1 row created.

SQL> INSERT INTO resources(resource_id, resource_title, format_type, duration)
  2  VALUES ('R4', 'Functions and Procedures', 'pdf', 100);

1 row created.

SQL> INSERT INTO resources(resource_id, resource_title, format_type, duration)
  2  VALUES ('R5', 'ER Diagrams Tutorial', 'mp4', 1);

1 row created.

SQL> INSERT INTO resources(resource_id, resource_title, format_type, duration)
  2  VALUES ('R6', 'SQL Basics', 'documentation', 140);

1 row created.

SQL> INSERT INTO resources(resource_id, resource_title, format_type, duration)
  2  VALUES ('R7', 'Normalization Techniques', 'documentation', 140);

1 row created.

SQL> INSERT INTO resources(resource_id, resource_title, format_type, duration)
  2  VALUES ('R8', 'Control Structures', 'pdf', 100);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 50 Screenshot: Actual Implementation of resource insertion

```
SQL> ALTER TABLE resources
  2  MODIFY resource_title CHAR(30);

Table altered.

SQL>
```

Figure 51Alter resources columns

TABLE student_module:

PRE-WRITTEN ROUGH NOTEPAD:

File Edit View

```
Bridging entities insertion:

student_module:
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S1','M1',4);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S1','M2',6);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S2','M3',6);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S2','M4',4);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S2','M5',4);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S3','M1',4);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S3','M6',4);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S3','M7',4);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S4','M8',5);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S4','M2',4);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S5','M3',6);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S5','M8',5);
INSERT INTO student_module(student_id, module_code, Module_level)
VALUES('S6','M6',4);
INSERT INTO student_module(student_id, module_code, Module_level)
```

Figure 52 Screenshot: Notepad to insert into student_module values

Figure 53 Screenshot: Actual implementation of student_module insertion

TABLE student_module_teacher_announcement:

PRE-WRITTEN ROUGH NOTEPAD:

```
student_module_teacher
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S1','M1','T1');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S1','M2','T8');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S2','M3','T2');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S2','M4','T3');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S2','M5','T3');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S3','M1','T5');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S3','M6','T4');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
VALUES('S3','M7','T4');
INSERT INTO student_module_teacher(student_id, module_code, teacher_id)
```

Figure 54 Screenshot: Notepad to insert into student_module_teacher values

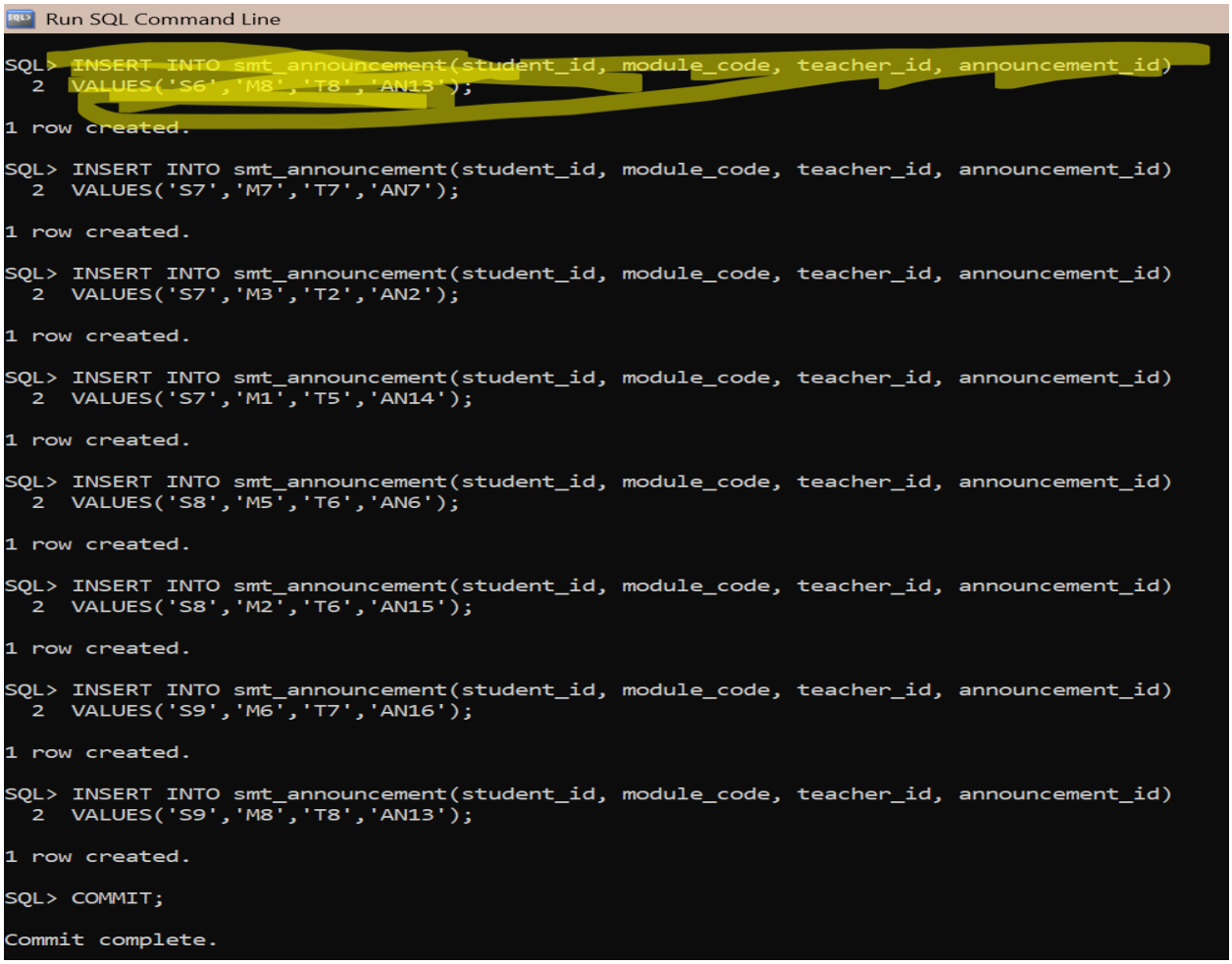


Figure 55 Screenshot: Actual implementation of student_module_teacher insertion

TABLE student_module_teacher:

PRE-WRITTEN ROUGH NOTEPAD:

```
smt_announcement
INSERT INTO smt_announcement(student_id, module_code, teacher_id, announcement_id)
VALUES('S1','M1','T1','AN1');
INSERT INTO smt_announcement(student_id, module_code, teacher_id, announcement_id)
VALUES('S1','M2','T8','AN10');
INSERT INTO smt_announcement(student_id, module_code, teacher_id, announcement_id)
VALUES('S2','M3','T2','AN2');
INSERT INTO smt_announcement(student_id, module_code, teacher_id, announcement_id)
VALUES('S2','M4','T3','AN4');
INSERT INTO smt_announcement(student_id, module_code, teacher_id, announcement_id)
```

Figure 56 Screenshot: Notepad to insert into student_module_teacher values

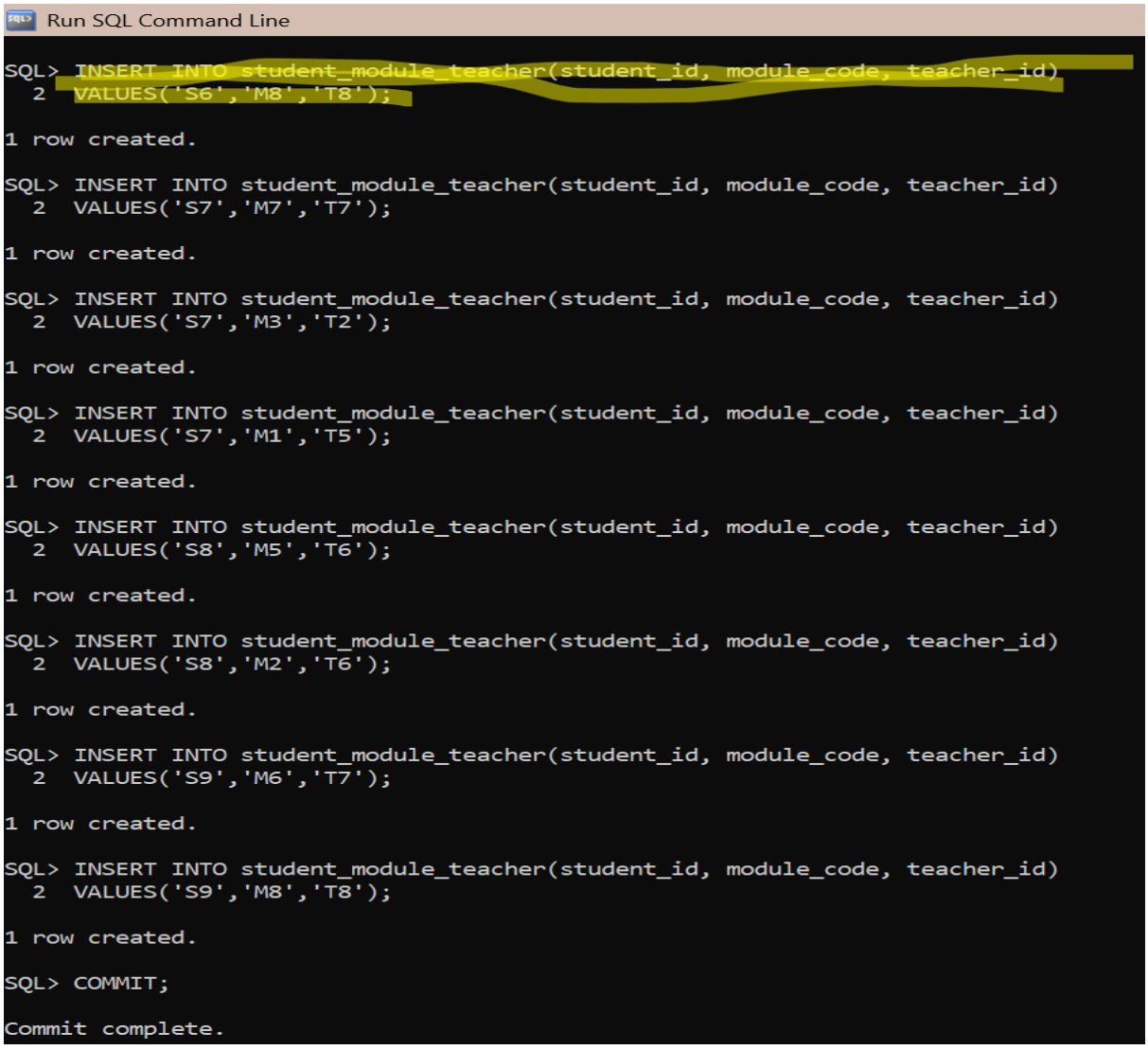


Figure 57 Screenshot: Actual implementation of student_module_teacher insertion

TABLE student_module_assessment:

PRE-WRITTEN ROUGH NOTEPAD:

```
student_module_assessment:
INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
VALUES('S1','M1','AS3','RR01','submitted',1);
INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
VALUES('S1','M2','AS9','RR02','submitted',3);
INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
VALUES('S2','M3','AS8','RR03','submitted',2);
INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
VALUES('S2','M4','AS2','RR04','submitted',1);
INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
```

Figure 58 Screenshot: Notepad to insert into student_module_assessment values

```
Run SQL Command Line

SQL> INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
  2  VALUES('S7','M7','AS6', 'RR07','submitted',3);

1 row created.

SQL> INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
  2  VALUES('S7','M3','AS11','RR14','not submitted',1);

1 row created.

SQL> INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
  2  VALUES('S7','M1','AS3','RR15','not submitted',1);

1 row created.

SQL> INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
  2  VALUES('S8','M5','AS5','RR05','submitted',1);

1 row created.

SQL> INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
  2  VALUES('S8','M2','AS9','RR09','submitted',1);

1 row created.

SQL> INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
  2  VALUES('S9','M6','AS4','RR12','not submitted',1);

1 row created.

SQL> INSERT INTO student_module_assessment(student_id, module_code, assessment_id, result_code, submission_status, attempt_number)
  2  VALUES('S9','M8','AS1','RR15','not submitted',3);

1 row created.

SQL> COMMIT;

Commit complete.
```

Figure 59 Screenshot: Actual implementation of student_module_assessment insertion

TABLE student_module_resource:

PRE-WRITTEN ROUGH NOTEPAD:

```
student_module_resource:
INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
VALUES('S1','M1', 'R9', 'completed');
INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
VALUES('S1','M2', 'R10', 'not completed');
INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
VALUES('S2','M3', 'R1', 'completed');
INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
VALUES('S2','M4', 'R3', 'completed');
INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
```

Figure 60 Screenshot: Notepad to insert into student_module_resource values

```
Run SQL Command Line
  2  VALUES('S6','M6', 'R13', 'completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S6','M8', 'R4', 'not completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S7','M7', 'R15', 'not completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S7','M3', 'R16', 'completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S7','M1', 'R9', 'completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S8','M5', 'R2', 'completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S8','M2', 'R10', 'completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S9','M6', 'R13', 'completed');

1 row created.

SQL> INSERT INTO student_module_resource(student_id, module_code, resource_id, completion_status)
  2  VALUES('S9','M8', 'R8', 'completed');

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

Figure 61 Screenshot: Actual implementation of student_module_resource insertion

7.2.4. SELECT ALL DETAILS FROM ALL TABLES

After inserting the required data(rows) in the tables. To view the details, **SELECT** keyword is used to view the details of every column.

SELECT * FROM programs

```
SQL> SELECT *
  2  FROM programs;

PROGRAM PROGRAM_TITLE                                PROGRAM_LENGTH TOTAL_CREDIT TUITION_FEE ENTRY_REQUIREMENT    MAXIMUM_CAPACITY MODE_OF_DELIVER
------- ---------------------------------------------- -------------- ------------ ----------- -------------------- ---------------- ---------------
P1      BSCs in Computing                                           4          390     1500000 minimum C grade                100 in-person
P2      BSc in Multimedia                                           3          390     1300000 minimum B grade                 80 in-person
P3      BSc in AI                                                   3          330     1600000 minimum 3.9 gpa                 70 in-person
P4      BSc in Networking                                           4          330     1230000 minimum B grade                 50 in-person
P5      Masters in Cyber Security                                   2          200     1000000 minimum 3.5 gpa                 10 in-person
P6      Masters in Data Science                                     2          150     1100000 minimum 3.5 credits             15 in-person
P7      Masters in Machine Learning                                 2          250      900000 minimum 3.5 credits              5 in-person

7 rows selected.

SQL>
```

Figure 62 Screenshot: Select programs table

SELECT * FROM students

```
SQL> SELECT *
  2  FROM students;

STUDENT FIRST_NAME     MIDDLE_NAME     LAST_NAME      HOME_ADDRESS                CONTACT_NUMBER EMAIL_ADDRESS                GENDER       AGE ENROLLED_ PROGRAM
------- -------------- -------------- -------------- ---------------------------- -------------- ------------------------------ ------ ---------- --------- -------
S1      Erika                         Shrestha       Hattigauda                      9823733980 erika.stha33@gmail.com         Female       20 20-DEC-23 P1
S2      Aamir                         Hamid          Maitidevi                       9823733978 aaHam_23@gmail.com             Male         20 14-JAN-24 P1
S3      Naz            Raj            Shahi          Baneshwor                       9823789765 nazri1@gmail.com               Male         21 18-FEB-21 P2
S4      Kiran                         Saud           Koteshwor                       9818137563 kirans08@gmail.com             Female       21 14-MAR-21 P2
S5      Aaditi         Newang         Chaudhari      Baluwatar                       9818137903 chaudharii101@gmail.com        Female       19 21-DEC-24 P3
S6      Narusha        Magar          Rai            Bouddha                         9800133896 nmr999@gmail.com               Female       19 30-DEC-24 P4
S7      Sandhya                       Subba          Narayanthan                     9818000008 sandhyas7272@gmail.com         Female       23 18-OCT-20 P5
S8      Nagendra                      Shrestha       Narayanthan                     9818976543 nagen80@gmail.com              Male         23 11-SEP-23 P6
S9      Ram                           Shrestha       Gokarneshwor                    9818022208 ram@gmail.com                  Male         23 17-JUN-24 P7

9 rows selected.
```

Figure 63 Screenshot: Select students table

SELECT * FROM modules

```
SQL> SELECT *
  2  FROM modules;

MODULE_ MODULE_TITLE                    CREDIT_SCORE DURATION_PERIOD
------- ------------------------------ ------------ ---------------
M1      Further Calculus                         15               6
M2      Probability Statistics                   15               6
M3      Database                                 15               6
M4      Data structure                           30              12
M5      Java programming                         30              12
M6      Software Engineering                     30              12
M7      Information system                       15               6
M8      Fundamentals of computing                15               6

8 rows selected.

SQL>
```

Figure 64 Screenshot: Select modules table

SELECT * FROM teachers

```
SQL> SELECT *
  2  FROM teachers;

TEACHER FIRST_NAME    MIDDLE_NAME   LAST_NAME     HOME_ADDRESS                CONTACT_NUMBER EDUCATIONAL_BACKGROUND    YEARS_OF_EXPERIENCE    SALARY JOINED_DATE
------- ------------- ------------- ------------- --------------------------- -------------- ------------------------ ------------------- ---------- -----------
T1      Anuja         Singh         Bhattarai     Nagarkot                     9823738392 bIT                                        5      20000 04-JAN-23
T2      Prithvi                     Maharjan      Basantapur                   9823738391 bsc in computing                           2      30000 20-JAN-22
T3      Deepesh       raj           adhikari      Budhanilkantha               9823738310 bsc in data analyst                        4      35000 16-JUN-21
T4      Indra                       Dhakal        Budhanilkantha               9823738312 bsc in mathematics                         3      25000 01-JUL-23
T5      Sushil        Prasad        Koirala       Bouddha                      9823789034 bsc in mathematics                         4      40000 01-DEC-20
T6      Rubin                       Thapa         Baluwatar                    9823789033 bsc in ecommerce                           6      50000 28-OCT-24
T7      Prasika                     Kc            Baneshwor                    9823789055 bsc in IoT                                 1      15000 05-OCT-20
T8      Numang                      Gurung        Baneshwor                    9823789044 bsc in IoT                                 3      20000 05-OCT-22

8 rows selected.
```

Figure 65 Screenshot: Select teachers table

SELECT * FROM announcements

```
SQL> SELECT *
  2  FROM announcements;

ANNOUNCEMENT_ID NOTICE_HEADLINE                      UPLOADED_DATE   EXPIRED_DATE
--------------- ------------------------------------ --------------- ---------------
AN1             Class postponed                      01-MAY-24       03-MAY-24
AN2             Class postponed                      18-JAN-25       11-FEB-25
AN3             Coursework rules                     08-MAR-25       10-MAR-25
AN4             Coursework rules                     08-MAR-25       10-MAR-25
AN5             College parking space                08-SEP-23       12-NOV-23
AN6             College reopens                      01-NOV-23       12-NOV-23
AN7             Milestone guidelines                 21-JUN-24       02-JUL-24
AN8             Milestone guidelines                 21-AUG-24       02-SEP-24
AN9             upcoming event                       26-MAY-24       30-MAY-24
AN10            upcoming event                       05-MAY-24       07-MAY-24
AN11            upcoming event                       29-NOV-25       02-DEC-25
AN12            College reopens                      01-NOV-23       12-NOV-23
AN13            Class postponed                      18-JAN-23       11-FEB-23
AN14            Milestone guidelines                 10-JAN-23       08-FEB-23
AN15            College parking space                08-SEP-22       12-NOV-22
AN16            Coursework rules                     15-MAY-24       20-MAY-24

16 rows selected.

SQL>
```

Figure 66 Screenshot: Select announcements table

SELECT * FROM assessments

```
SQL> SELECT *
  2  FROM assessments;

ASSESSME ASSESSMENT_TITLE     ASSIGNED_ DUE_DATE  CATEGORY    WEIGHTAGE RESULT_RELEASED_DATE
-------- -------------------- --------- --------- ---------- ---------- ---------------------
AS1      invoice creation     09-DEC-24 24-DEC-24 individual         30 08-OCT-25
AS2      college app          28-DEC-24 04-JAN-25 individual         40 10-JAN-25
AS3      class test           11-DEC-24 11-DEC-24 group             100 30-JAN-25
AS4      srs                  20-DEC-24 01-JAN-25 group              20 08-JAN-25
AS5      crud operations      05-NOV-23 11-NOV-23 individual         60 14-DEC-23
AS6      Dashboard creation   05-NOV-24 11-NOV-24 individual         40 14-DEC-24
AS7      searching algorithms 15-NOV-25 21-NOV-25 group              40 14-DEC-25
AS8      database creation    15-NOV-15 21-NOV-15 individual         70 24-DEC-15
AS9      basic integration    05-NOV-24 11-NOV-24 individual         60 14-DEC-24
AS10     gantt chart          05-SEP-24 10-SEP-24 individual         30 02-DEC-24
AS11     normalization        28-DEC-24 04-JAN-25 individual         40 10-JAN-25

11 rows selected.

SQL>
```

Figure 67 Screenshot: Select assessments table

SELECT * FROM resources

```
SQL> SELECT *
  2  FROM resources;

RESOURCE RESOURCE_TITLE                FORMAT_TYPE       DURATION
-------- ----------------------------- --------------- ----------
R1       Introduction to Database      pdf                    100
R2       Intro to Programming          mp4                      1
R3       Variables and Data Types      documentation          140
R4       Functions and Procedures      pdf                    100
R5       ER Diagrams Tutorial          mp4                      1
R6       SQL Basics                    documentation          140
R7       Normalization Techniques      documentation          140
R8       Control Structures            pdf                    100
R9       Intro to integration          pdf                    100
R10      Statistics                    pdf                    100
R11      Collection data type          pdf                    100
R12      CPM usage                     mp4                      1
R13      Project management            documentation          140
R14      Partial derivative            pdf                    100
R15      Standard deviation            mp4                      1
R16      Sorting and searching         documentation          140

16 rows selected.
```

Figure 68 Screenshot: Select resources table

SELECT * FROM results

```
SQL> SELECT *
  2  FROM results;

RESULT_CODE          OBTAINED_MARK
-------------------- -------------
RR01                            50
RR03                            60
RR04                            70
RR05                            80
RR06                            60
RR07                            80
RR08                            70
RR09                            40
RR02                            50
RR10                            50
RR11                            40
RR12                            20
RR13                            10
RR14                            40
RR15                             0

15 rows selected.
```

Figure 69 Screenshot: Select results table

SELECT * FROM result_grades

```
SQL> SELECT *
  2  FROM result_grades;

OBTAINED_MARK GRADE
------------- -----
           40 P
           50 D
           60 C
           70 B
           80 A
           30 F
           20 F
           10 F
            0 F

9 rows selected.
```

Figure 70 Screenshot: Select result_grades table

SELECT * FROM student_module

```
SQL> SELECT *
  2  FROM student_module;

STUDENT MODULE_CODE     MODULE_LEVEL
------- --------------- ------------
S1      M1                         4
S1      M2                         6
S2      M3                         6
S2      M4                         4
S2      M5                         4
S3      M1                         4
S3      M6                         4
S3      M7                         4
S4      M8                         5
S4      M2                         4
S5      M3                         6
S5      M8                         5
S6      M6                         4
S6      M8                         5
S7      M7                         6
S7      M3                         6
S7      M1                         5
S8      M5                         6
S8      M2                         6
S9      M6                         4
S9      M8                         5

21 rows selected.
```

Since student can enroll in many modules and modules can also have many same students

Figure 71 Screenshot: Select student_module table

SELECT * FROM student_module_teacher

```
SQL> SELECT *
  2  FROM student_module_teacher;

STUDENT_ID      MODULE_CODE     TEACHER_ID
--------------- --------------- ---------------
S1              M1              T1
S1              M2              T8
S2              M3              T2
S2              M4              T3
S2              M5              T3
S3              M1              T5
S3              M6              T4
S3              M7              T4
S4              M8              T5
S4              M2              T2
S5              M3              T2
S5              M8              T8
S6              M6              T1
S6              M8              T8
S7              M7              T7
S7              M3              T2
S7              M1              T5
S8              M5              T6
S8              M2              T6
S9              M6              T7
S9              M8              T8

21 rows selected.
```

The teacher teaches many modules.

Figure 72 Screenshot: Select student_module_teacher table

SELECT * FROM student_module_teacher_announcement

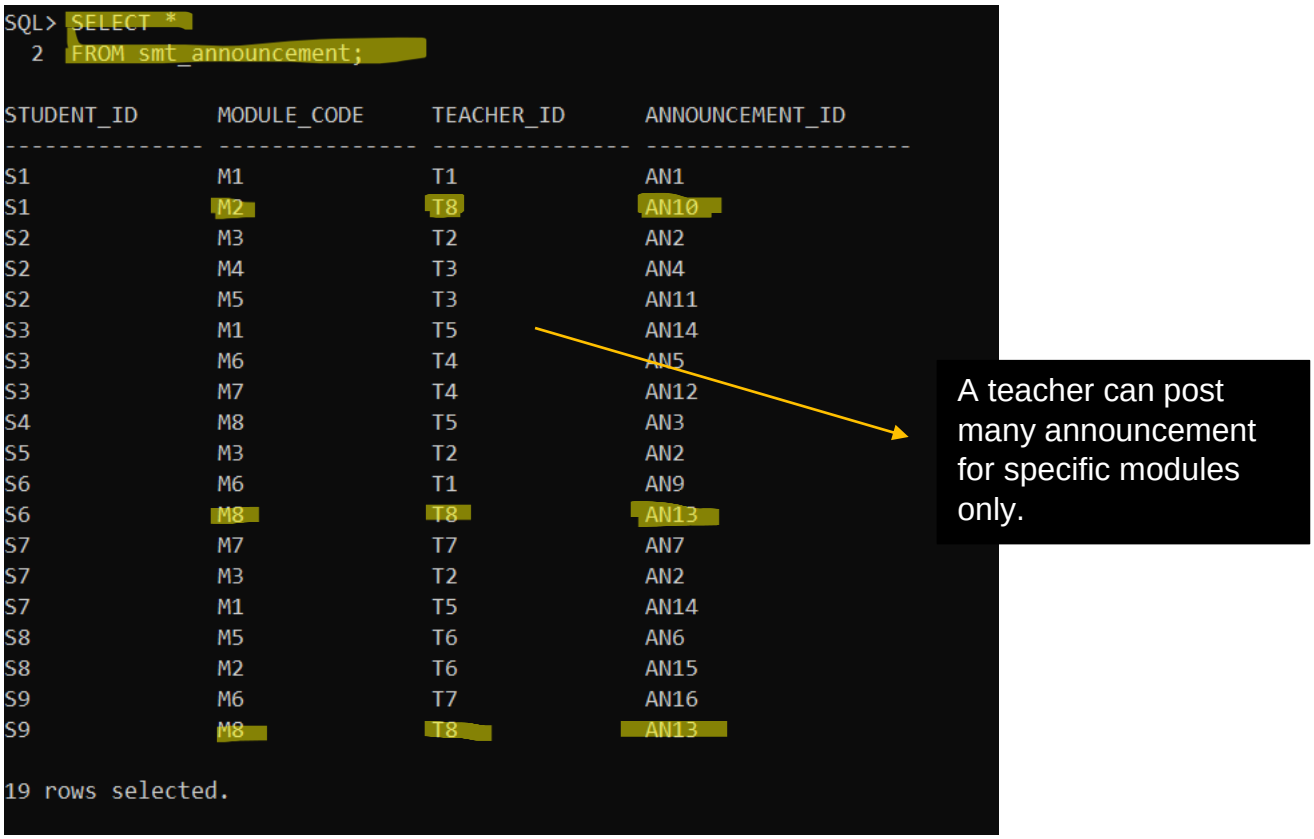


Figure 73 Screenshot: Select student_module_teacher_announcement table

SELECT * FROM student_module_assessment

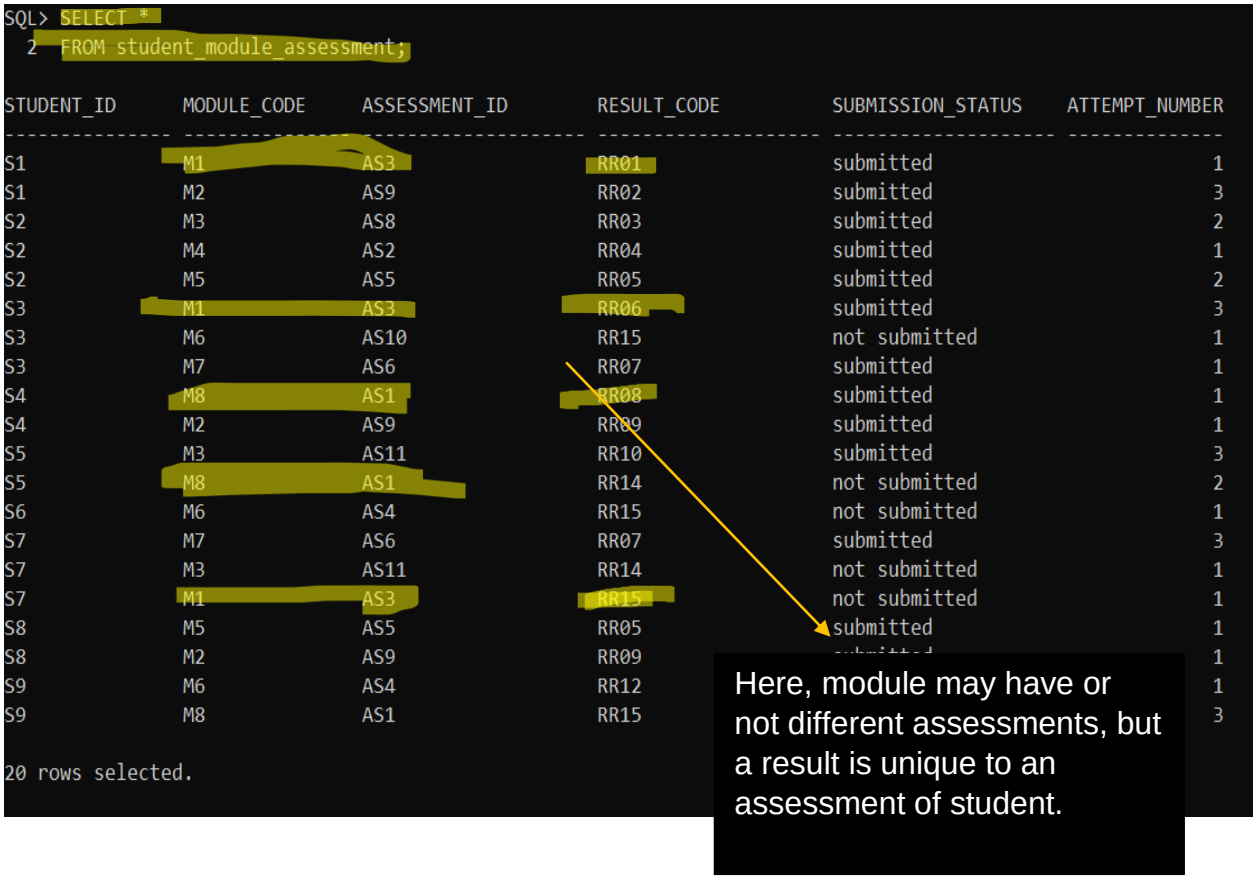


Figure 74 Screenshot: Select student_module_assessment table

SELECT * FROM student_module_resource

```
SQL> SELECT *
  2  FROM student_module_resource;

STUDENT_ID      MODULE_CODE     RESOURCE_ID         COMPLETION_STATUS
--------------- --------------- ------------------- -------------------------
S1              M1              R9                  completed
S1              M2              R10                 not completed
S2              M3              R1                  completed
S2              M4              R3                  completed
S2              M5              R7                  not completed
S3              M1              R14                 not completed
S3              M6              R13                 completed
S3              M7              R11                 completed
S4              M8              R4                  not completed
S4              M2              R15                 not completed
S5              M3              R6                  completed
S5              M8              R1                  completed
S6              M6              R13                 completed
S6              M8              R4                  not completed
S7              M7              R15                 not completed
S7              M3              R16                 completed
S7              M1              R9                  completed
S8              M5              R2
S8              M2              R10
S9              M6              R13
S9              M8              R8

21 rows selected.
```

Here, all the modules have more than one unique resources.

Figure 75 Screenshot: Select student_module_resource table

7.2.5. DROP TABLES FROM USER

PRE-WRITTEN ROUGH NOTEPAD:

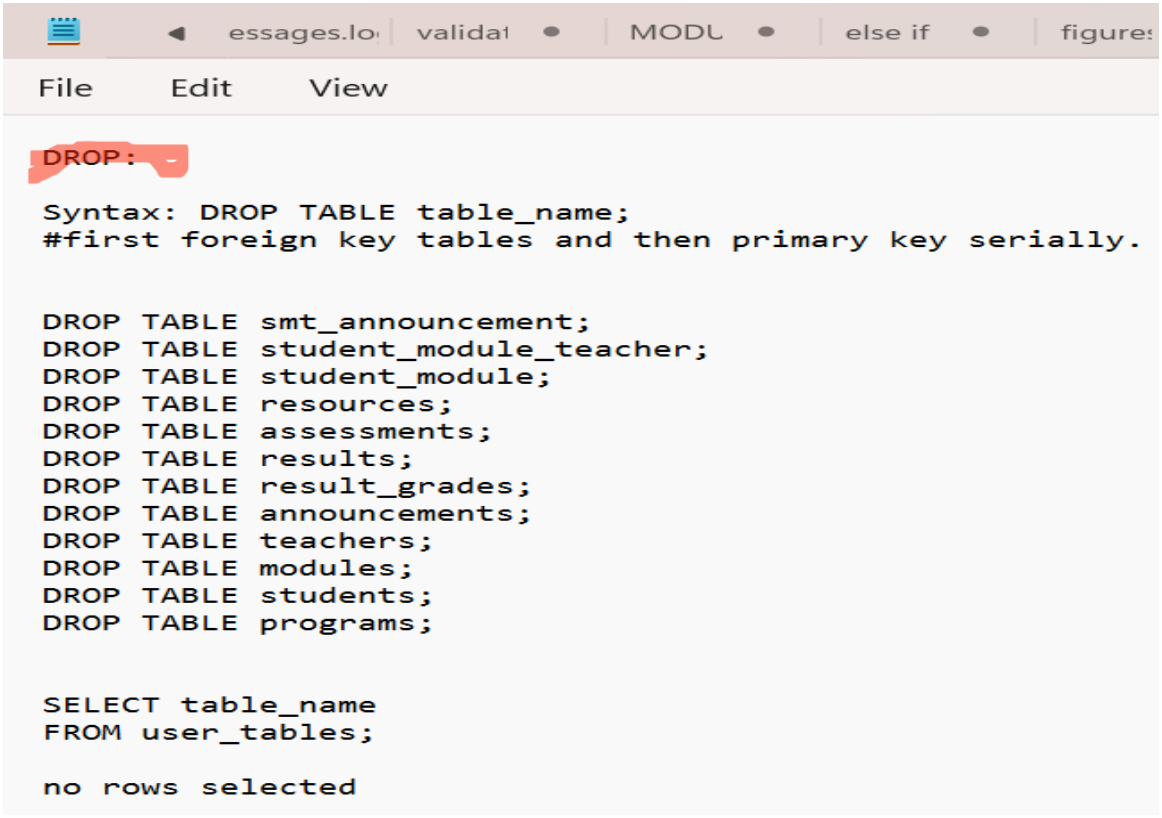


Figure 76 Screenshot: Notepad to drop tables

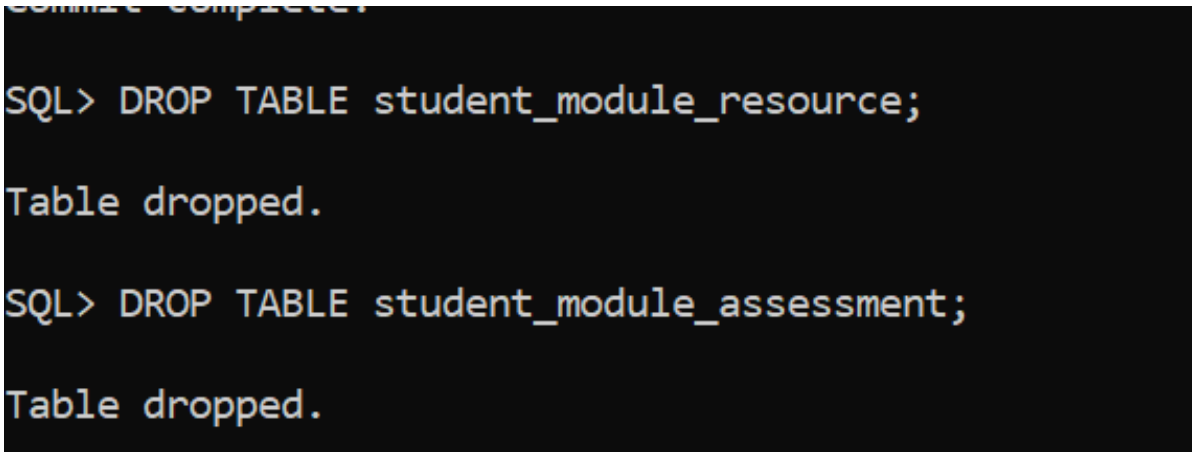


Figure 77 Screenshot: Actual implementation of drop student_module_resource and student_module_assessment tables

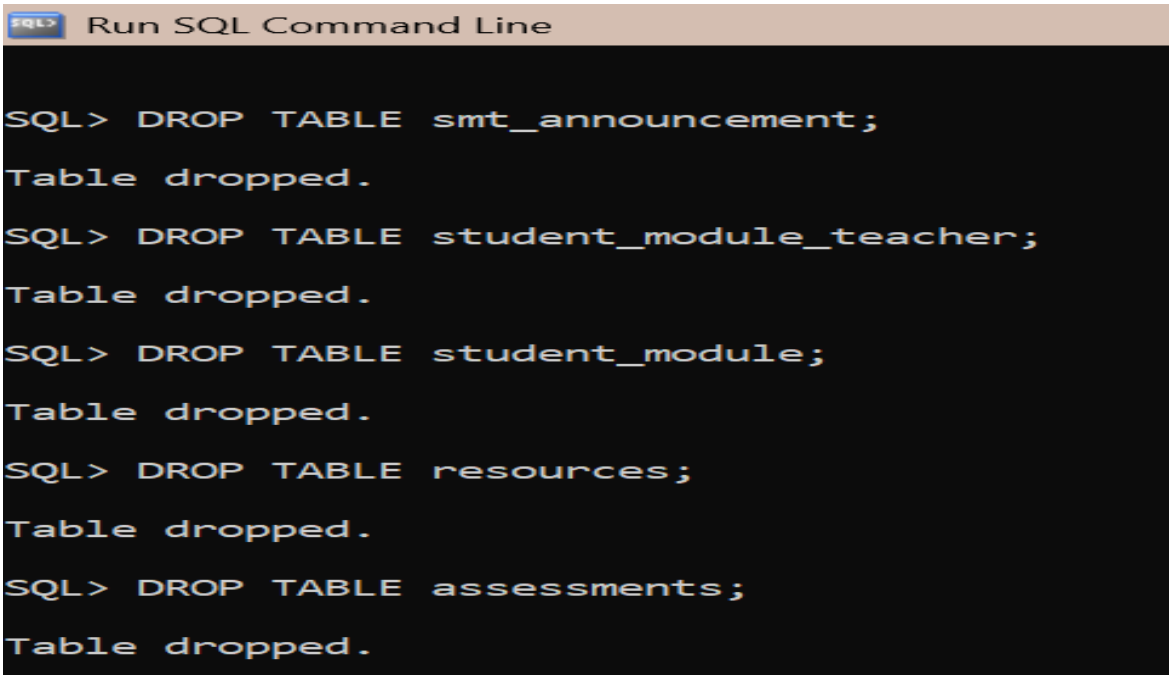


Figure 78 Screenshot: Actual implementation of drop smt_announcement, student_module_teacher, student_module, resources and assessments tables

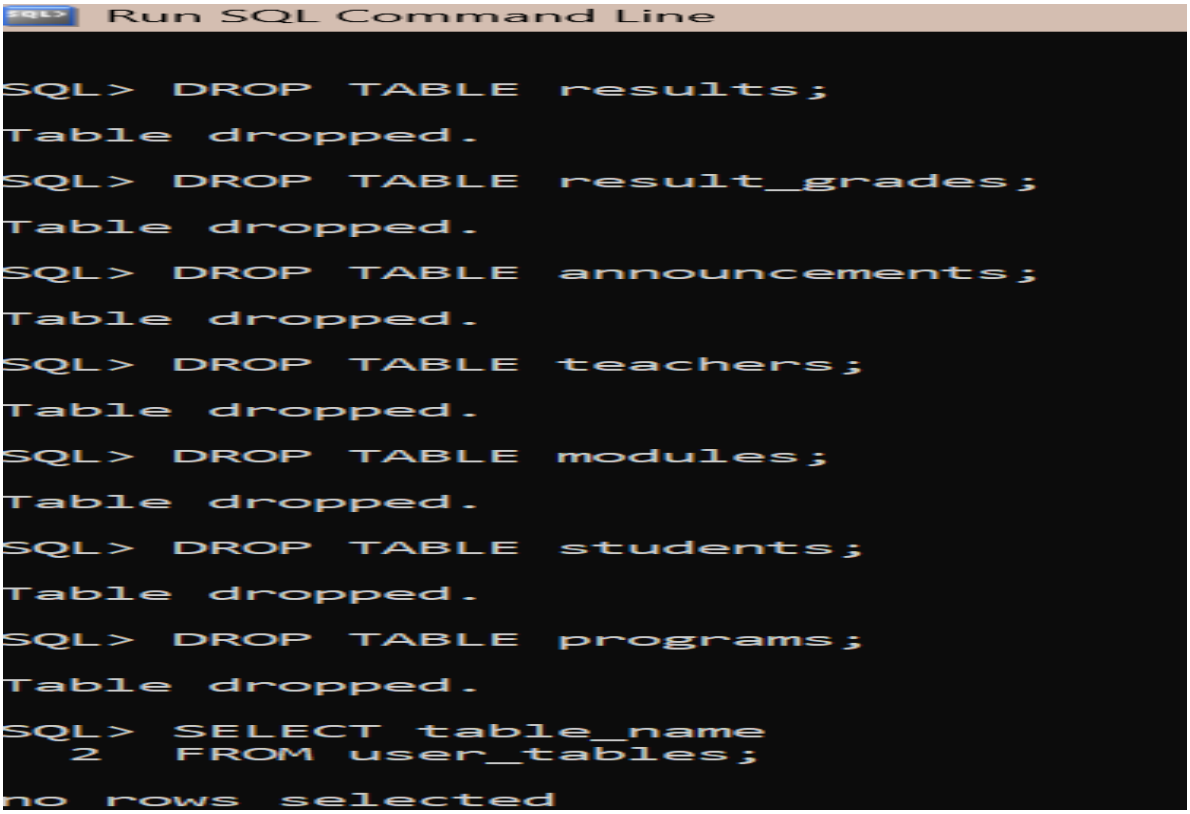


Figure 79 Screenshot: Actual implementation of drop results, result_grades, announcements, teachers, modules, students, programs tables

7.2.6. DUMP FILE CREATION AND OTHER COMMANDS

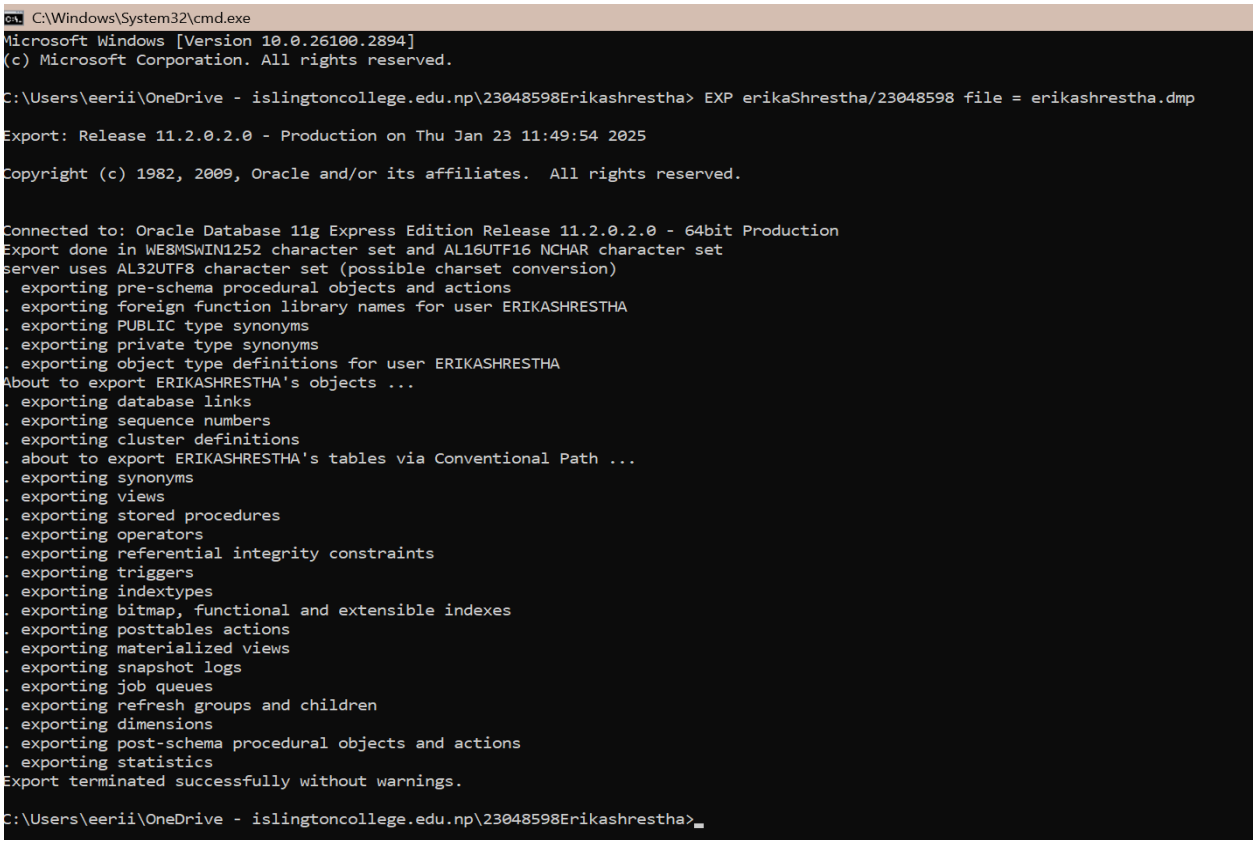


Figure 80 Screenshot: Dump file creation

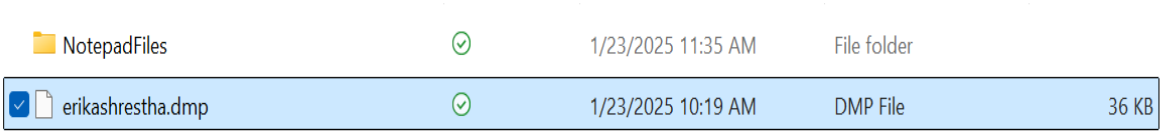


Figure 81 Screenshot: Dump file in folder

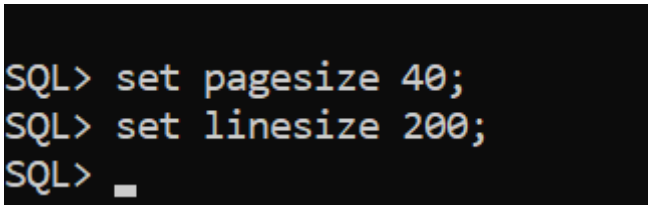


Figure 82 Screenshot: set linesize and pagesize

7.3. SCREENSHOT OF QUERIES APPLIED IN DATABASE

The queries are divided into two sub-parts **information** and **transaction** queries where information queries involve retrieving data from the table like a **QnA** whereas the transaction queries are to follow **TCL** (Transaction control language) while retrieving the data.

7.3.7. INFORMATION QUERY

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

PRE-WRITTEN ROUGH NOTEPAD:

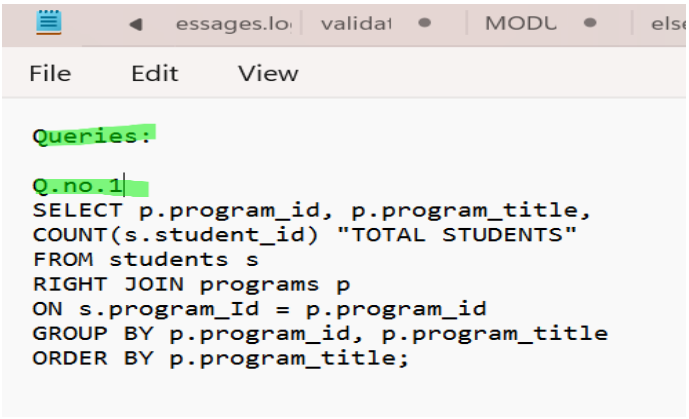


Figure 83 Screenshot: Notepad to information query 1

Result:

```
SQL> SELECT p.program_id, p.program_title,
  2  COUNT(s.student_id) "TOTAL STUDENTS"
  3  FROM students s
  4  RIGHT JOIN programs p
  5  ON s.program_Id = p.program_id
  6  GROUP BY p.program_id, p.program_title
  7  ORDER BY p.program_title;

PROGRAM PROGRAM_TITLE                                    TOTAL STUDENTS
------- -------------------------------------------------- --------------
P1      BSCs in Computing                                             2
P3      BSc in AI                                                     1
P2      BSc in Multimedia                                             2
P4      BSc in Networking                                             1
P5      Masters in Cyber Security                                     1
P6      Masters in Data Science                                       1
P7      Masters in Machine Learning                                   1

7 rows selected.

SQL>
```

Figure 84 Screenshot: Result of information query 1

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

```
Q.no.2
SELECT DISTINCT m.module_code, a.announcement_id, m.module_title, a.notice_headline, a.uploaded_date
FROM modules m
JOIN student_module sm
ON m.module_code = sm.module_code
JOIN smt_announcement smt_a
ON m.module_code = smt_a.module_code
JOIN announcements a
ON smt_a.announcement_id = a.announcement_id
WHERE m.module_code
IN (SELECT smt_a.module_code
FROM smt_announcement smt_a
JOIN announcements a
ON smt_a.announcement_id= a.announcement_id
WHERE a.uploaded_date
BETWEEN TO_DATE('01-may-24','DD-MON-YY')
AND TO_DATE('28-may-24','DD-MM-YY'))
AND a.uploaded_date BETWEEN TO_DATE('01-may-24','DD-MON-YY')
AND TO_DATE('28-may-24','DD-MM-YY')
ORDER BY m.module_code;
```

Figure 85 Screenshot: Notepad to information query 2

```
SQL> SELECT DISTINCT m.module_code, a.announcement_id, m.module_title, a.notice_headline, a.uploaded_date
  2  FROM modules m
  3  JOIN student_module sm
  4  ON m.module_code = sm.module_code
  5  JOIN smt_announcement smt_a
  6  ON m.module_code = smt_a.module_code
  7  JOIN announcements a
  8  ON smt_a.announcement_id = a.announcement_id
  9  WHERE m.module_code
 10  IN (SELECT smt_a.module_code
 11  FROM smt_announcement smt_a
 12  JOIN announcements a
 13  ON smt_a.announcement_id= a.announcement_id
 14  WHERE a.uploaded_date
 15  BETWEEN TO_DATE('01-may-24','DD-MON-YY')
 16  AND TO_DATE('28-may-24','DD-MM-YY'))
 17  AND a.uploaded_date BETWEEN TO_DATE('01-may-24','DD-MON-YY')
 18  AND TO_DATE('28-may-24','DD-MM-YY')
 19  ORDER BY m.module_code;

MODULE_CODE  ANNOUNCEMENT_ID MODULE_TITLE               NOTICE_HEADLINE                      UPLOADED_DATE
-----------  -------------- --------------------------- ------------------------------------ --------------
M1           AN1            Further Calculus            Class postponed                      01-MAY-24
M2           AN10           Probability Statistics      upcoming event                       05-MAY-24
M6           AN16           Software Engineering        Coursework rules                     15-MAY-24
M6           AN9            Software Engineering        upcoming event                       26-MAY-24

SQL>
```

Figure 86 Screenshot: Result of information query 2

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

```
Q.no.3
SELECT m.module_title,
COUNT (r.resource_id) "TOTAL RESOURCES"
FROM modules m
JOIN student_module_resource smr
ON m.module_code = smr.module_code
JOIN resources r
ON r.resource_id = smr.resource_id
WHERE m.module_title like 'D%'
GROUP BY m.module_code,m.module_title
ORDER BY m.module_title;
```

Figure 87 Screenshot: Notepad to information query 3

```
SQL> SELECT m.module_title,
  2  COUNT (r.resource_id) "TOTAL RESOURCES"
  3  FROM modules m
  4  JOIN student_module_resource smr
  5  ON m.module_code = smr.module_code
  6  JOIN resources r
  7  ON r.resource_id = smr.resource_id
  8  WHERE m.module_title like 'D%'
  9  GROUP BY m.module_code,m.module_title
 10  ORDER BY m.module_title;

MODULE_TITLE                    TOTAL RESOURCES
------------------------------ ---------------
Data structure                               1
Database                                     3
```

Figure 88 Screenshot: Result of information query 3

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

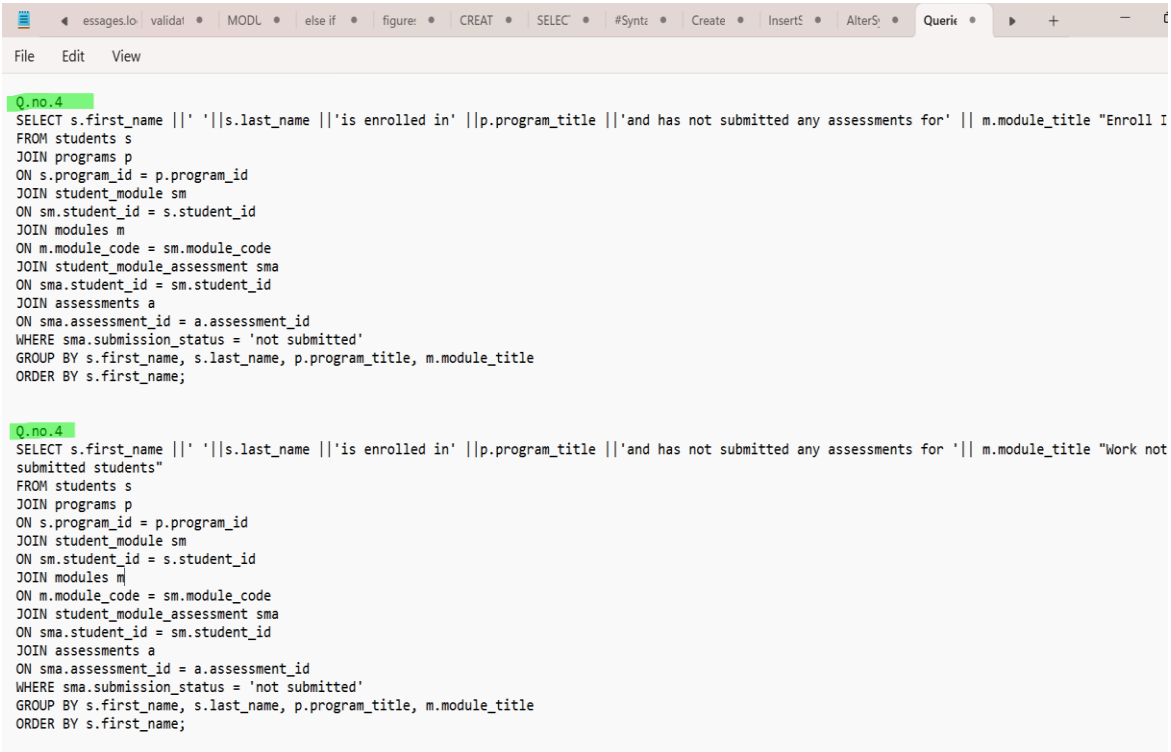


Figure 89 Screenshot: Notepad to information query 4

```
SQL> SELECT s.first_name || s.last_name || ' is enrolled in ' || p.program_title "Work not submitted students"
  2  FROM students s
  3  JOIN programs p
  4  ON s.program_id = p.program_id
  5  JOIN student_module sm
  6  ON sm.student_id = s.student_id
  7  JOIN student_module_assessment sma
  8  ON sma.student_id = sm.student_id
  9  JOIN assessments a
 10  ON sma.assessment_id = a.assessment_id
 11  WHERE sma.submission_status = 'not submitted'
 12  GROUP BY s.first_name, s.last_name, p.program_title
 13  ORDER BY s.first_name;

Work not submitted students
-------------------------------------------------------------------------------------------
Aaditi          Chaudhari       is enrolled in BSc in AI
Narusha         Rai             is enrolled in BSc in Networking
Naz             Shahi           is enrolled in BSc in Multimedia
Ram             Shrestha        is enrolled in Masters in Machine Learning
Sandhya         Subba           is enrolled in Masters in Cyber Security
```

Figure 90 Screenshot: Result of information query 4 part 1

```
SQL> SELECT s.first_name ||' '||s.last_name ||'is enrolled in' ||p.program_title ||'and has not submitted any assessments for '|| m.module_title "Work not submitted student
s"
  2  FROM students s
  3  JOIN programs p
  4  ON s.program_id = p.program_id
  5  JOIN student_module sm
  6  ON sm.student_id = s.student_id
  7  JOIN modules m
  8  ON m.module_code = sm.module_code
  9  JOIN student_module_assessment sma
 10  ON sma.student_id = sm.student_id
 11  JOIN assessments a
 12  ON sma.assessment_id = a.assessment_id
 13  WHERE sma.submission_status = 'not submitted'
 14  GROUP BY s.first_name, s.last_name, p.program_title, m.module_title
 15  ORDER BY s.first_name;

Work not submitted students
-------------------------------------------------------------------------------------------------------------------------------------------------------
Aaditi          Chaudhari       is enrolled inBSc in AI                                 and has not submitted any assessments for Database
Aaditi          Chaudhari       is enrolled inBSc in AI                                 and has not submitted any assessments for Fundamentals of computing
Narusha         Rai             is enrolled inBSc in Networking                         and has not submitted any assessments for Fundamentals of computing
Narusha         Rai             is enrolled inBSc in Networking                         and has not submitted any assessments for Software Engineering
Naz             Shahi           is enrolled inBSc in Multimedia                         and has not submitted any assessments for Further Calculus
Naz             Shahi           is enrolled inBSc in Multimedia                         and has not submitted any assessments for Information system
Naz             Shahi           is enrolled inBSc in Multimedia                         and has not submitted any assessments for Software Engineering
Ram             Shrestha        is enrolled inMasters in Machine Learning               and has not submitted any assessments for Fundamentals of computing
Ram             Shrestha        is enrolled inMasters in Machine Learning               and has not submitted any assessments for Software Engineering
Sandhya         Subba           is enrolled inMasters in Cyber Security                  and has not submitted any assessments for Database
Sandhya         Subba           is enrolled inMasters in Cyber Security                  and has not submitted any assessments for Further Calculus
Sandhya         Subba           is enrolled inMasters in Cyber Security                  and has not submitted any assessments for Information system

12 rows selected.
```

Figure 91 Screenshot: Result of information query 5 part 2

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

```
Q.no.5
SELECT t.first_name||' '||t.last_name||' has '||t.educational_background||' degree with '||t.years_of_experience||' year experience '||' who teaches '||
COUNT (DISTINCT m.module_code)||' modules. ' "Teacher Details"
FROM teachers t
JOIN student_module_teacher smt
ON t.teacher_Id = smt.teacher_id
JOIN student_module sm
ON sm.module_code = smt.module_code
JOIN modules m
ON m.module_code = sm.module_code
GROUP BY t.teacher_id, t.first_name, t.last_name, t.educational_background, t.years_of_experience
HAVING COUNT (DISTINCT m.module_code) >1
ORDER BY t.first_name;
```

Figure 92 Screenshot: Notepad to information query 5

```
SQL> SELECT t.first_name||' '||t.last_name||' has '||t.educational_background||' degree with '||t.years_of_experience||' year experience '||' who teaches '||
  2  COUNT (DISTINCT m.module_code)||' modules. ' "Teacher Details"
  3  FROM teachers t
  4  JOIN student_module_teacher smt
  5  ON t.teacher_Id = smt.teacher_id
  6  JOIN student_module sm
  7  ON sm.module_code = smt.module_code
  8  JOIN modules m
  9  ON m.module_code = sm.module_code
 10  GROUP BY t.teacher_id, t.first_name, t.last_name, t.educational_background, t.years_of_experience
 11  HAVING COUNT (DISTINCT m.module_code) >1
 12  ORDER BY t.first_name;

Teacher Details
--------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------
Anuja           Bhattarai       has bIT                 degree with 5 year experience  who teaches 2 modules.
Deepesh         adhikari        has bsc in data analyst     degree with 4 year experience  who teaches 2 modules.
Indra           Dhakal          has bsc in mathematics      degree with 3 year experience  who teaches 2 modules.
Numang          Gurung          has bsc in IoT              degree with 3 year experience  who teaches 2 modules.
Prasika         Kc              has bsc in IoT              degree with 1 year experience  who teaches 2 modules.
Prithvi         Maharjan        has bsc in computing        degree with 2 year experience  who teaches 2 modules.
Rubin           Thapa           has bsc in ecommerce        degree with 6 year experience  who teaches 2 modules.
Sushil          Koirala         has bsc in mathematics      degree with 4 year experience  who teaches 2 modules.

8 rows selected.
```

Figure 93 Screenshot: Result of information query 5

7.3.8. TRANSACTION QUERY

- **Identify the module that has the latest assessment deadline.**

```
Transaction queries

Q.no.1
SELECT *
FROM
(SELECT m.module_title ||' has the latest due date which is '|| TO_CHAR(a.due_date, 'YYYY-MM-DD') "Latest Assessment Deadline"
FROM modules m
JOIN student_module sm ON m.module_code = sm.module_code
JOIN student_module_assessment sma ON sm.module_code = sma.module_code
JOIN assessments a ON a.assessment_id = sma.assessment_id
GROUP BY m.module_title, a.due_date
ORDER BY a.due_date DESC)
WHERE ROWNUM = 1;

COMMIT;
```

Figure 94 Screenshot: Notepad to transaction query 1

```
SQL> SELECT *
  2  FROM
  3  (SELECT m.module_title ||' has the latest due date which is '|| TO_CHAR(a.due_date, 'YYYY-MM-DD') "Latest Assessment Deadline"
  4  FROM modules m
  5  JOIN student_module sm ON m.module_code = sm.module_code
  6  JOIN student_module_assessment sma ON sm.module_code = sma.module_code
  7  JOIN assessments a ON a.assessment_id = sma.assessment_id
  8  GROUP BY m.module_title, a.due_date
  9  ORDER BY a.due_date DESC)
 10  WHERE ROWNUM = 1;

Latest Assessment Deadline
--------------------------------------------------------------------------------
Data structure               has the latest due date which is 2025-01-04

SQL> COMMIT;

Commit complete.

SQL>
```

Figure 95 Screenshot: Result of transaction query 1

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

```
Q.no.2
SELECT *
FROM
(SELECT s.first_name||' '||s.last_name|| 'has the highest score '|| SUM(r.obtained_mark) "TOP THREE STUDENTS"
FROM students s
JOIN student_module sm
ON sm.student_id= s.student_id
JOIN student_module_assessment sma
ON sma.student_id = sm.student_id
AND sma.module_code = sm.module_code
JOIN results r
ON r.result_code = sma.result_code
GROUP BY s.first_name, s.last_name
ORDER BY SUM(r.obtained_mark) desc)
WHERE ROWNUM <=3;

COMMIT;
```

Figure 96 Screenshot: Notepad to transaction query 2

```
SQL> SELECT *
  2  FROM
  3  (SELECT s.first_name||' '||s.last_name|| 'has the highest score '|| SUM(r.obtained_mark) "TOP THREE STUDENTS"
  4  FROM students s
  5  JOIN student_module sm
  6  ON sm.student_id= s.student_id
  7  JOIN student_module_assessment sma
  8  ON sma.student_id = sm.student_id
  9  AND sma.module_code = sm.module_code
 10  JOIN results r
 11  ON r.result_code = sma.result_code
 12  GROUP BY s.first_name, s.last_name
 13  ORDER BY SUM(r.obtained_mark) desc)
 14  WHERE ROWNUM <=3;

TOP THREE STUDENTS
--------------------------------------------------------------------------------
Aamir           Hamid          has the highest score 210
Naz             Shahi          has the highest score 140
Sandhya         Subba          has the highest score 120

SQL> COMMIT;

Commit complete.
```

Figure 97 Screenshot: Result of transaction query 2

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

```
Q.no.3
SELECT p.program_title||' program has '||COUNT(DISTINCT sma.assessment_id)||' assessments'|| ' with an average score of '|| ROUND(AVG(r.obtained_mark)) "Total
assessments"
FROM students s
RIGHT JOIN programs p
ON s.program_id = p.program_id
JOIN student_module sm
ON sm.student_id = s.student_id
JOIN student_module_assessment sma
ON sma.student_id = sm.student_id
JOIN results r
ON r.result_code = sma.result_code
GROUP BY p.program_title
ORDER BY p.program_title;

COMMIT;
```

Figure 98 Screenshot: Notepad to transaction query 3

```
SQL> SELECT p.program_title||' program has '||COUNT(DISTINCT sma.assessment_id)||' assessments'|| ' with an average score of '|| ROUND(AVG(r.obtained_mark)) "Total assessme
nts"
  2  FROM students s
  3  RIGHT JOIN programs p
  4  ON s.program_id = p.program_id
  5  JOIN student_module sm
  6  ON sm.student_id = s.student_id
  7  JOIN student_module_assessment sma
  8  ON sma.student_id = sm.student_id
  9  JOIN results r
 10  ON r.result_code = sma.result_code
 11  GROUP BY p.program_title
 12  ORDER BY p.program_title;

Total assessments
--------------------------------------------------------------------------------------------------------------------------------------------------------
---------
BSCs in Computing                          program has 5 assessments with an average score of 64
BSc in AI                                  program has 2 assessments with an average score of 45
BSc in Multimedia                          program has 5 assessments with an average score of 49
BSc in Networking                          program has 1 assessments with an average score of 0
Masters in Cyber Security                  program has 3 assessments with an average score of 40
Masters in Data Science                    program has 2 assessments with an average score of 60
Masters in Machine Learning                program has 2 assessments with an average score of 10

7 rows selected.

SQL> COMMIT;

Commit complete.
```

Figure 99 Screenshot: Result of transaction query 3

- **List the students who have scared above the average score in the ‘Databases’ module.**

```
Q.no.4
SELECT s.first_name|| ' '|| s.last_name || ' scores above average ' || AVG(r.obtained_mark)||' in '||' '|| m.module_title "Above average scorer students"
FROM students s
JOIN student_module sm
ON sm.student_id = s.student_id
JOIN student_module_assessment sma
ON sma.student_id = sm.student_id
JOIN modules m
ON m.module_code = sm.module_code
AND sm.module_code = sma.module_code
JOIN results r
ON r.result_code = sma.result_code
WHERE m.module_title = 'Database'
GROUP BY s.first_name, s.last_name, r.obtained_mark, m.module_title
HAVING AVG(r.obtained_mark) >
(SELECT AVG(r2.obtained_mark)
FROM results r2
JOIN student_module_assessment sma2
ON sma2.result_code = r2.result_code
JOIN student_module sm2
ON sm2.student_id = sma2.student_id
AND sm2.module_code = sma2.module_code
JOIN modules m2
ON m2.module_code = sm2.module_code
WHERE m2.module_title = 'Database')
ORDER BY s.first_name;

COMMIT;
```

Figure 100 Screenshot: Notepad to transaction query 4

```
SQL> SELECT s.first_name|| ' '|| s.last_name || ' scores above average ' || AVG(r.obtained_mark)||' in '||' '|| m.module_title "Above average scorer students"
  2  FROM students s
  3  JOIN student_module sm
  4  ON sm.student_id = s.student_id
  5  JOIN student_module_assessment sma
  6  ON sma.student_id = sm.student_id
  7  JOIN modules m
  8  ON m.module_code = sm.module_code
  9  AND sm.module_code = sma.module_code
 10  JOIN results r
 11  ON r.result_code = sma.result_code
 12  WHERE m.module_title = 'Database'
 13  GROUP BY s.first_name, s.last_name, r.obtained_mark, m.module_title
 14  HAVING AVG(r.obtained_mark) >
 15  (SELECT AVG(r2.obtained_mark)
 16  FROM results r2
 17  JOIN student_module_assessment sma2
 18  ON sma2.result_code = r2.result_code
 19  JOIN student_module sm2
 20  ON sm2.student_id = sma2.student_id
 21  AND sm2.module_code = sma2.module_code
 22  JOIN modules m2
 23  ON m2.module_code = sm2.module_code
 24  WHERE m2.module_title = 'Database')
 25  ORDER BY s.first_name;

Above average scorer students
--------------------------------------------------------------------------------------------------------------------------------
Aamir           Hamid           scores above average 60 in  Database

SQL> COMMIT;

Commit complete.
```

Figure 101 Screenshot: Result of transaction query 4

- **Display whether a student has passed or failed as remarks as per their total aggregate marks obtained in a particular module. (total obtained marks equal or above 40 is marked as pass, below 40 or null values are marked as fail.)**

```
Q.no.5
SELECT s.first_name||''||s.last_name "Student name", m.module_title "Module title",
SUM(r.obtained_mark) "Total mark",
'PASS' "Remarks"
FROM students s
JOIN student_module sm
ON s.student_id = sm.student_id
JOIN student_module_assessment sma
ON sm.student_id = sma.student_id
JOIN modules m
ON sm.module_code = m.module_code
AND sm.module_code = sma.module_code
JOIN results r
ON r.result_code = sma.result_code
GROUP BY s.first_name, s.last_name, m.module_title
HAVING SUM(r.obtained_mark)>=40
UNION ALL
SELECT s.first_name||''||s.last_name "Student name", m.module_title "Module title",
SUM(r.obtained_mark) "Total mark",
'FAIL' "remarks"
FROM students s
JOIN student_module sm
ON s.student_id = sm.student_id
JOIN student_module_assessment sma
ON sm.student_id = sma.student_id
JOIN modules m
ON sm.module_code = m.module_code
AND sm.module_code = sma.module_code
JOIN results r
ON r.result_code = sma.result_code
GROUP BY s.first_name, s.last_name, m.module_title
HAVING SUM(r.obtained_mark)<40;

COMMIT;
```

Figure 102 Screenshot: Notepad to transaction query 5

Run SQL Command Line

```
SQL> SELECT s.first_name||''||s.last_name "Student name", m.module_title "Module title",
  2  SUM(r.obtained_mark) "Total mark",
  3  'PASS' "Remarks"
  4  FROM students s
  5  JOIN student_module sm
  6  ON s.student_id = sm.student_id
  7  JOIN student_module_assessment sma
  8  ON sm.student_id = sma.student_id
  9  JOIN modules m
 10  ON sm.module_code = m.module_code
 11  AND sm.module_code = sma.module_code
 12  JOIN results r
 13  ON r.result_code = sma.result_code
 14  GROUP BY s.first_name, s.last_name, m.module_title, r.obtained_mark
 15  HAVING SUM(r.obtained_mark)>=40
 16  UNION ALL
 17  SELECT s.first_name||''||s.last_name "Student name", m.module_title "Module title",
 18  SUM(r.obtained_mark) "Total mark",
 19  'FAIL' "remarks"
 20  FROM students s
 21  JOIN student_module sm
 22  ON s.student_id = sm.student_id
 23  JOIN student_module_assessment sma
 24  ON sm.student_id = sma.student_id
 25  JOIN modules m
 26  ON sm.module_code = m.module_code
 27  AND sm.module_code = sma.module_code
 28  JOIN results r
 29  ON r.result_code = sma.result_code
 30  GROUP BY s.first_name, s.last_name, m.module_title, r.obtained_mark
 31  HAVING SUM(r.obtained_mark)<40;

Student name                  Module title                  Total mark Remarks
----------------------------- ----------------------------- ---------- ---------------
Erika          Shrestha       Probability Statistics                50 PASS
Kiran          Saud           Fundamentals of computing             70 PASS
Naz            Shahi          Further Calculus                      60 PASS
Aaditi         Chaudhari      Database                              50 PASS
Kiran          Saud           Probability Statistics                40 PASS
Nagendra       Shrestha       Java programming                      80 PASS
Naz            Shahi          Information system                    80 PASS
Sandhya        Subba          Information system                    80 PASS
Nagendra       Shrestha       Probability Statistics                40 PASS
Aamir          Hamid          Java programming                      80 PASS
```

Figure 103 Screenshot: Result of transaction query 5 part 1

```
 25  JOIN modules m
 26  ON sm.module_code = m.module_code
 27  AND sm.module_code = sma.module_code
 28  JOIN results r
 29  ON r.result_code = sma.result_code
 30  GROUP BY s.first_name, s.last_name, m.module_title
 31  HAVING SUM(r.obtained_mark)<40;

Student name                   Module title                  Total mark Remarks
------------------------------ ----------------------------- ---------- --------------
Sandhya        Subba           Database                              40 PASS
Aamir          Hamid           Java programming                      80 PASS
Aamir          Hamid           Database                              60 PASS
Kiran          Saud            Probability Statistics                40 PASS
Nagendra       Shrestha        Java programming                      80 PASS
Naz            Shahi           Information system                    80 PASS
Aaditi         Chaudhari       Fundamentals of computing             40 PASS
Erika          Shrestha        Probability Statistics                50 PASS
Kiran          Saud            Fundamentals of computing             70 PASS
Nagendra       Shrestha        Probability Statistics                40 PASS
Aamir          Hamid           Data structure                        70 PASS
Erika          Shrestha        Further Calculus                      50 PASS
Naz            Shahi           Further Calculus                      60 PASS
Aaditi         Chaudhari       Database                              50 PASS
Sandhya        Subba           Information system                    80 PASS
Sandhya        Subba           Further Calculus                       0 FAIL
Naz            Shahi           Software Engineering                   0 FAIL
Narusha        Rai             Software Engineering                   0 FAIL
Ram            Shrestha        Software Engineering                  20 FAIL
Ram            Shrestha        Fundamentals of computing              0 FAIL

20 rows selected.

SQL> COMMIT;

Commit complete.
```

Figure 104 Screenshot: Result of transaction query 5 part 2

8. CRITICAL EVALUATION

This section evaluates the usages and relation of the database project with other subjects. The aim is to critically analysis, evaluate and understand more about the database project.

8.1. USAGE

The database system developed for this coursework performs excellent yet simple and user-friendly interface ensuring ease of use even for non-programmer users. In addition, the SQL language is a simple language and like English which is why the retrieval of data can be done with few practices.

With the help of referential constraints and keys, the implementation of data integrity and minimized redundancy is seen which leads to saving storage and reducing manual errors.

In addition, performing multiple testing resulted in less unexpected errors as the problems were solved.

8.2. RELATION WITH OTHER SUBJECTS

The database was a creation of mixed knowledge and logic applied from different subjects which directly related to it whereas some indirectly.

8.2.1. PROGRAMMING (JAVA, PYTHON)

The implementation of queries and constraint validation like NUMBER for age, CHARACTER for genders acts like a data type just as in java and python programming.

8.2.2. MATHEMATICS (SETS/JOINS)

The queries were mostly made with the help of joints which were derived from sets used in mathematic calculations. The join combines two or more tables into one categorized by union, left, right, inner etc (GeeksforGeeks, 2024).

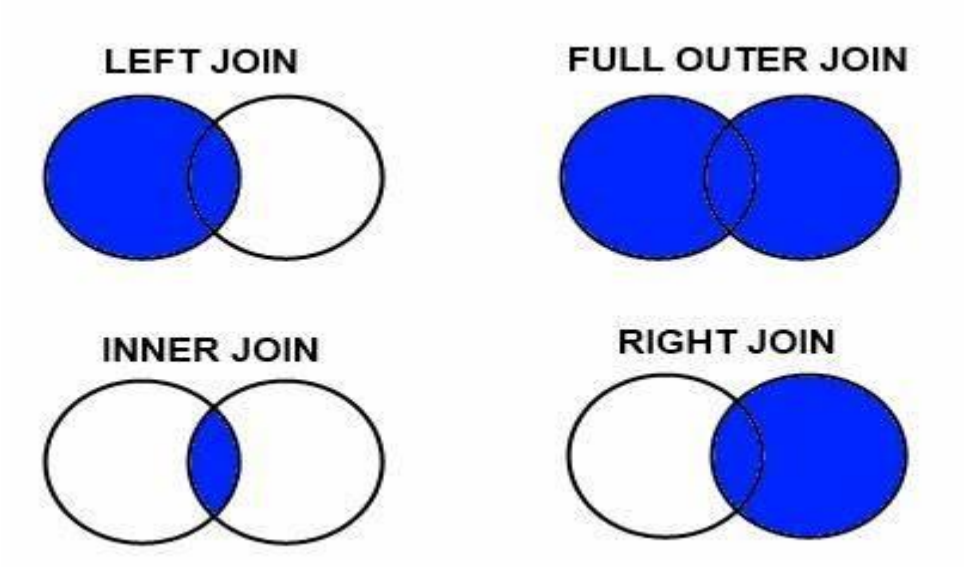


Figure 105 joins implemented

8.2.3. PROJECT MANAGEMENT

The management of the project started from breaking the task into smaller milestones to highlight important tasks and to complete within the time frame. It was practiced making the project successful, complete and functional.

9. CONCLUSION

The project "**EVISION ROOM"** database has been **successfully completed** with all the mentioned features for the E-PLATFORM SYSTEM and Ms. Mary stated that the system ran smoothly and provided user friendly environment. The problem faced by the school no longer **prevails**. Overall, the project strongly shows and practices relationship between them in relation to the business rules and proper normalization.

With that I would like to conclude that the project "EVISION ROOM" has successfully complete to help Ms. Mary for her upcoming days and wish her with prosperity and well-being.

10. REFERENCE

Abraham Silberschatz, H. F. K. S. S., 2011. *Database System Concepts.* 6th ed. New York: McGraw-Hill Companies.

Alalouf, A., 2018. *Cardinality in Data Modelling,* Philadelphia: Temple University.

Brown, F., 2024. *Guru99.* [Online]
Available at: https://www.guru99.com/er-diagram-tutorial-dbms.html
[Accessed 21 January 2025].

GeeksforGeeks, 2023. *Entity in DBMS.* [Online]
Available at: https://www.geeksforgeeks.org/entity-in-dbms/
[Accessed 21 January 2025].

GeeksforGeeks, 2024. *Joins in DBMS.* [Online]
Available at: https://www.geeksforgeeks.org/joins-in-dbms/
[Accessed 23 January 2025].

geeksforgeeks, 2024. *What is functional dependencies in dbms?.* [Online]
Available at: https://www.geeksforgeeks.org/what-is-functional-dependency-in-dbms/
[Accessed 29 dec 2024].

geeks, g. f., 2024. *What is data dictionary in dbms?.* [Online]
Available at: https://www.geeksforgeeks.org/what-is-data-dictionary/
[Accessed 30 dec 2024].

Ramez Elmasri, S. B. N., 2015. *Fundamentals Of Database Systems.* 7th ed. Texas: Pearson.

Sudarshan, K. a., n.d. Chapter-6: Database Design Using the E-R Model. In: K. a. Sudarshan, ed. *Database System Concepts.* s.l.:Silberschatz, p. 80.