MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AI&ML)

DATABASE MANAGEMENT SYSTEMS LAB

REGULATION- BT25

A.Y-2026-2027

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

INDEX

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

CERTIFICATE

This is to certify that this manual is a Bonafide record of practical work carried out in the

Database Management Systems Laboratory for the **B. Tech Computer Science Engineering (AI&ML) III Semester** Programme during the academic year **2026–2027.**

This manual has been prepared by **Ms.SK. RAHEEMA (Assistant Professor),** Department of Computer Science Engineering (AI&ML), with my own efforts and to the best of our knowledge.

Signature of Lab Faculty **Signature of HOD**

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

PREFACE

This Lab Manual entitled "DATABASE MANAGEMENT SYSTEMS Lab" is intended for the use of II B. Tech I Semester Computer Science and Engineering (AIML) students of Marri Laxman Reddy Institute of Technology and Management, Dundigal, Hyderabad. The main objective of the DBMS Lab is to provide hands-on experience in designing and managing databases using structured techniques. Students learn to develop database systems through E-R modeling, relational schema design, and normalization to ensure data integrity and efficiency. The lab enhances practical skills in SQL by practicing DDL and DML commands, complex queries, subqueries, joins, and aggregate functions. It also introduces advanced concepts such as views, triggers, stored procedures, and cursors for effective database automation and control. Overall, the lab prepares students to handle real-world data management problems and builds a strong foundation for database development and applications in industry.

By,

Ms. SK. Raheema

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

ACKNOWLEDGEMENT

It was really a good experience, working at Database Management Systems Lab. First, I would like to thank Dr. Mr. Parvez Mohammad, Assistant Professor, Department of Computer Science & Engineering, Marri Laxman Reddy Institute of technology & Management for giving the technical support in preparing the document.

I express my sincere thanks to Dr. B Ravi Prasad, Head of the Department of Computer Science & Engineering (AI&ML), Marri Laxman Reddy Institute of technology & Management, for his concern towards me and gave me opportunity to prepare Database Management Systems laboratory manual.

I am deeply indebted and gratefully acknowledge the constant support and valuable patronage of Dr. B Ravi Prasad, Dean Academics, Marri Laxman Reddy Institute of technology & Management. I am unboundedly grateful to him for timely corrections and scholarly guidance.

I express my heartful thanks to Dr. P. Sridhar, Director, and Dr. R. Murali Prasad, Principal, Marri Laxman Reddy Institute of technology & Management, for giving me this wonderful opportunity for preparing the Database Management Systems laboratory manual.

At last, but not the least I would like to thank the entire Computer Science & Engineering Department faculties those who had inspired and helped me to achieve my goal.

By,

Ms. SK. Raheema

Department of CSE (AI&ML)

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

GENERAL INSTRUCTIONS

1. Students are instructed to come to Database Management Systems laboratory on time. Late comers are not entertained in the lab.
2. Students should be punctual to the lab. If not, the conducted experiments will not be repeated.
3. Students are expected to come prepared at home with the experiments which are going to be performed.
4. Students are instructed to display their identity cards before entering into the lab.
5. Students are instructed not to bring mobile phones to the lab.
6. Any damage/loss of system parts like keyboard, mouse during the lab session, it is student's responsibility and penalty or fine will be collected from the student.
7. Students should update the records and lab observation books session wise. Before leaving the lab the student should get his/her lab observation book signed by the faculty.
8. Students should submit the lab records by the next lab to the concerned faculty members in the staff room for their correction and return.
9. Students should not move around the lab during the lab session.
10. If any emergency arises, the student should take the permission from faculty member concerned in written format.
11. The faculty members may suspend any student from the lab session on disciplinary grounds.
12. Never copy the output from other students. Write down your own outputs.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

SAFETY MEASURES

To ensure the safe and efficient use of the Computer Science and Engineering (AI&ML) laboratory, all students must strictly adhere to the following safety guidelines:

1. General Conduct

- Maintain silence and discipline during lab sessions.
- Do not bring food, drinks, or chewing gum into the lab.
- Use lab resources responsibly and follow all instructions provided by the instructor or lab assistant.

2. Electrical Safety

- Do not touch electrical switches, sockets, or plugs with wet hands.
- Avoid overloading power sockets with unauthorized devices.
- Immediately report any loose connections, sparks, or unusual noises from equipment.

3. Computer and Equipment Handling

- Handle all computer systems, keyboards, mice, and peripherals with care.
- Do not attempt to open or tamper with any hardware components.
- Use only the assigned computer system; do not switch systems without permission.

4. Software and Data Safety

- Use only authorized software installed by the lab administrator.
- Do not attempt to install, uninstall, or modify any software without approval.
- Save your work frequently and ensure backups of important files.

5. Cybersecurity and Network Usage

- Keep your login credentials confidential.
- Do not attempt to access restricted websites or servers.
- Avoid activities such as hacking, gaming, or the use of pirated content.

6. Emergency Preparedness

- Be familiar with the location of emergency exits, fire extinguishers, and first aid kits.
- In the event of a fire, electrical hazard, or any emergency, remain calm and inform the lab instructor immediately.
- Follow the evacuation procedure as instructed.

7. Post-Lab Procedures

- Log out of your session and shut down the system properly after use.
- Leave your workstation clean and organized.
- Return any borrowed materials or equipment to their proper place.

8. Hygiene and Cleanliness

- Wash or sanitize your hands before and after using shared devices.
- Do not write or place unnecessary items on the workstation.
- Report any spills or cleanliness issues to the lab staff.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

VISION & MISSION OF THE INSTITUTE

Vision of the Institute:

To be a globally recognized institution that fosters innovation, excellence, and leadership in education, research, and technology development, empowering students to create sustainable solutions for the advancement of society.

Mission of the Institute:

- To foster a transformative learning environment that empowers students to excel in engineering, innovation, and leadership.
- To produce skilled, ethical, and socially responsible engineers who contribute to sustainable technological advancements and address global challenges.
- To shape future leaders through cutting-edge research, industry collaboration, and community engagement.

VISION & MISSION OF THE DEPARTMENT

Vision of the Department:

To nurture globally competent professionals in Artificial Intelligence and Machine Learning through excellence in education, research, and innovation, committed to developing sustainable and impactful solutions for the betterment of society.

Mission of the Department:

- To provide a transformative learning environment that equips students with in-depth knowledge and practical skills in Artificial Intelligence and Machine Learning, fostering innovation, leadership, and lifelong learning.
- To advance AI and ML through cutting-edge research, strong industry collaboration, and community engagement, preparing students to address real-world challenges on a global scale.
- To produce competent and ethical AI professionals who contribute to technological progress while addressing societal and environmental challenges with sustainable solutions.
- To foster a research-driven culture by partnering with industry and academia, encouraging entrepreneurship, and engaging in community-centered technology development.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Program Educational Objectives (PEOs)

PEO 1	**Professional Competence** Graduates will possess strong theoretical and practical knowledge in Artificial Intelligence and Machine Learning, enabling them to solve complex real-world problems, pursue higher education, or excel in professional careers
PEO 2	**Innovation and Research Orientation:** Graduates will engage in innovative practices, cutting-edge research, and contribute to the advancement of AI and ML technologies through collaboration with industry and academia.
PEO 3	**Ethical and Social Responsibility:** Graduates will demonstrate ethical behavior, social responsibility, and environmental awareness in their professional conduct, contributing to sustainable development and societal well-being.
PEO 4	**Leadership and Lifelong Learning:** Graduates will exhibit leadership qualities, effective communication, and teamwork skills, and will continuously upgrade their knowledge to adapt to evolving technological landscapes.
PEO 5	**Entrepreneurial and Community Engagement:** Graduates will leverage entrepreneurial skills and a sense of civic responsibility to create AI-driven solutions that benefit local and global communities.

Course Outcomes:

- Understand conceptual database design using E–R modeling and relational database structure principles.
- Apply relational model concepts, normalization techniques, and dependency rules for efficient schema design.
- Execute database definition and manipulation commands for schema creation, data insertion, modification, and retrieval.
- Develop complex SQL queries using joins, subqueries, set operations, aggregate functions, views, and constraints.
- Implement database programming constructs including triggers, procedures, and cursors for automated data processing.

PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES:

PO 1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and engg. specialization to the solution of complex engineering problems.
PO 2: Problem analysis: Identify, formulate, research literature, and analyze engineering problems to arrive at substantiated conclusions using first principles of mathematics, natural, and engineering sciences.
PO 3: Design/development of solutions: Design solutions for complex engineering problems and design system components, processes to meet the specifications with consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4: Conduct investigations of complex problems: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
PO 10: Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions.

PO 11: Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

PO 12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

PSO 1: Able to identify, analyze and solve the problems related to Artificial Intelligence and Machine Learning by applying the fundamental knowledge of Computer Science and Engineering.

PSO 2: Build innovative tools and techniques to develop project models in the areas related to Deep Learning, Machine learning, Artificial Intelligence.

PSO 3: Make use of the Artificial Intelligence and Machine Learning knowledge to assess societal, environmental, health, safety issues, Sustainable development goals with professional ethics and can also pursue higher studies, involve in research activities, be employable or entrepreneur.

25X0580: DATA BASE MANAGEMENT SYSTEMS LAB

B.Tech. II Year I Sem.

L T P C

0 0 2 1

Course Objectives:

- Introduce ER data model, database design and normalization
- Learn SQL basics for data definition and data manipulation

Course Outcomes: After Completion of the Course, Students Should be able to:

- Understand conceptual database design using E–R modeling and relational database structure principles.
- Apply relational model concepts, normalization techniques, and dependency rules for efficient schema design.
- Execute database definition and manipulation commands for schema creation, data insertion, modification, and retrieval.
- Develop complex SQL queries using joins, subqueries, set operations, aggregate functions, views, and constraints.
- Implement database programming constructs including triggers, procedures, and cursors for automated data processing.

List of Experiments:

1. Concept design with E-R Model

2. Relational Model

3. Normalization

4. Practicing DDL commands

5. Practicing DML commands

6. A) Querying (using ANY, ALL, UNION, INTERSECT, JOIN, Constraints etc.) B) Nested, Correlated subqueries

7. Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.

8. Triggers (Creation of insert trigger, delete trigger, update trigger)

9. Procedures

10. Usage of Cursors

TEXT BOOKS:

1. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata Mc Graw Hill,3rd Edition

2. Database System Concepts, Silberschatz, Korth, McGraw Hill, V edition.

REFERENCES BOOKS:

1. Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel 7th Edition.

2. Fundamentals of Database Systems, Elmasri Navrate, Pearson Education

3. Introduction to Database Systems, C.J. Date, Pearson Education

4. Oracle for Professionals, The X Team, S. Shah and V. Shah, SPD.

5. Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL, Shah, PHI.

6. Fundamentals of Database Management Systems, M. L. Gillenson, Wiley Student Edition.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

DATABASE MANAGEMENT SYSTEMS LAB

Virtual lab details

Name of the Virtual Lab: DATABASE MANAGEMENT SYSTEMS LAB

Virtual Lab Host Institute: **Pune Institute of Computer Technology**

URL/Link to Lab: https://dmsl-virtual-lab.vercel.app/

Academic Year: 2026-2027.

Semester: III SEM

List of Experiments Available in Virtual Lab

DATABASE MANAGEMENT SYSTEMS LAB

LAB PLANNER

S.No	Experiment	CO	Virtual Lab Availability	Date planned	Date conducted
1	Concept Design using E–R Model	CO1			
2	Relational Model	CO1			
3	Normalization	CO2			
4	Practicing DDL commands	CO3	YES		
5	Practicing DML commands	CO3	YES		
6	**LAB INTERNAL-1**				
7	A) Querying (using ANY, ALL, UNION, INTERSECT, JOIN, Constraints etc.) B) Nested, Correlated subqueries	CO4			
8	Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.	CO4			
9	Triggers (Creation of insert trigger, delete trigger, update trigger)	CO5	YES		
10	Procedures	CO5	YES		
11	Usage of Cursors	CO5	YES		
12	**LAB INTERNAL-2**				

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML

DATABASE MANAGEMENT SYSTEMS LAB
LAB PLANNER

Date planned																																						
Date conducted																																					
Roll Number	ExpNo	CO	VL*	ExpNo	CO	VL*	ExpNo	CO	VL*	ExpNo	CO	VL*	ExpNo	CO	VL*		ExpNo	CO	VL*	ExpNo	CO	VL*	ExpNo	CO	VL*	ExpNo	CO	VL*	ExpNo	CO	VL*						
	1	1	Y	2	1	N	3	2	Y	4	2	N	5	3	Y		6	4	Y	7	4	Y	8	4	Y	9	5	N	10	5	Y						
	2	1	N	3	2	Y	4	2	N	5	3	Y	6	4	Y	M	7	4	Y	8	4	Y	9	5	N	10	5	Y	1	1	Y	**M**					
	3	2	Y	4	2	N	5	3	Y	6	4	Y	7	4	Y	I	8	4	Y	9	5	N	10	5	Y	1	1	Y	2	1	N	**I**					
	4	2	N	5	3	Y	6	4	Y	7	4	Y	8	4	Y	D	9	5	N	10	5	Y	1	1	Y	2	1	N	3	2	Y	D					
	5	3	Y	6	4	Y	7	4	Y	8	4	Y	9	5	N	-	10	5	Y	1	1	Y	2	1	N	3	2	Y	4	2	N	-					
	6	4	Y	7	4	Y	8	4	Y	9	5	N	10	5	Y	I	1	1	Y	2	1	N	3	2	Y	4	2	N	5	3	Y	**II**					

Note:VL*-Virtual Lab Availabilty

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

RUBRICS USED TO ASSESS LEARNINGS IN LABORATORIES

1. RUBRICS FOR DAY TO DAY EVALUATION

Parameter	Max Marks	Level-1 (Very Poor)	Level-2 (Poor)	Level-3 (Average)	Level-4 (Good)	Level-5 (Excellent)
Observation Book	05	No observations or irrelevant data. (0–1)	Incomplete or incorrect data. (2)	Basic values with some errors. (3)	Mostly correct with good format. (4)	Fully correct, clear, and well-formatted. (5)
Record Writing	05	Not submitted. (0-1)	Submitted but mostly incomplete. (2)	Submitted with some missing/wrong parts. (3)	Submitted with minor issues. (4)	Fully complete, correct algorithm & flowchart. (5)
Result	05	No result or major errors. (0–1)	Result partially obtained. (2)	Acceptable result with limited error. (3)	Near-correct result and reasonable error. (4)	Accurate result. (5)
Viva-Voce	05	Did not answer any questions. (1)	Answered very few questions. (2)	Answered some questions with help. (3)	Answered most questions correctly. (4)	Answered all questions accurately. (5)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

DATABASE MANAGEMENT SYSTEMS LAB

2. RUBRICS FOR INTERNAL EVALUATION

Criterion	Max Marks	Level-1 (Very Poor)	Level-2 (Poor)	Level-3 (Average)	Level-4 (Good)	Level-5 (Excellent)	
Design/Tool/Apparatus Selection	2 Marks	Incorrect tool/design and no reasoning. (0)	Tool/design selection attempted with unclear logic. (0.5)	Satisfactory selection with partial justification. (1)	Correct selection and proper analysis with few errors. (1.5)	Smart selection with accurate, relevant analysis. (2)	
Execution (Code/Debug/Run) /Analysis/Method Used	4 Marks	Did not attempt or completely failed to execute. (0)	Attempted but unable to proceed or with major errors. (1)	Partial execution with some logic/syntax errors. (2)	Mostly correct execution with minimal help. (3)	Fully correct and independently executed program. (4)	
Results& Documentation	2 Marks	Incomplete or poorly presented. (0)	Basic structure but lacks clarity or formatting. (0.5)	Complete but generic or with formatting issues. (1)	Well-structured and mostly clear. (1.5)	Well-organized, professional, and engaging documentation. (2)	
Viva-Voce (Understanding of Concepts)	2 Marks	No understanding; could not answer questions. (0)	Answered a few with difficulty. (0.5)	Answered half the questions with basic clarity. (1)	Good understanding with confident answers. (1.5)	Answered all questions with clarity and depth. (2)	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

DATABASE MANAGEMENT SYSTEMS LAB

3. RUBRICS FOR SEMESTER END EXAMINATIONS

Criterion	Max Marks	Level-1 (Very Poor *(0–2 marks)*	Level-2 (Poor) *(3–4 marks)*	Level-3 (Average) *(5–6 marks)*	Level-4 (Good) *(7–9 marks)*	Level-5 (Excellent) *(10–12 marks)*
Preparedness for the Experiment	12 marks	No clarity on objective or procedure. Unable to explain basics.	Limited idea of the objective/procedure. Needed prompting.	Has basic understanding; minor gaps in concept or preparation.	Well-prepared, with clear understanding of steps and background.	Fully prepared with strong conceptual clarity and confident explanation.
Performance in the Laboratory	12 marks	Unable to perform experiment. Relied entirely on examiner's help.	Performed with multiple errors and constant support.	Performed with some errors; required occasional help.	Performed mostly independently with minimal support.	Performed independently, efficiently, and with precision.
Calculations & Graphs	12 marks	No or incorrect calculations. Graphs missing or irrelevant.	Multiple calculation errors. Graphs/plots inaccurate or poorly labeled.	Calculations partially correct. Graphs present but with some flaws.	Correct calculations and graphs with minor errors.	Accurate calculations and well-labeled graphs with proper interpretation.
Results & Error Analysis	12 marks	No result or invalid result. No error analysis attempted.	Incorrect result with vague or no error discussion.	Acceptable result. Error analysis attempted but limited.	Correct result with sound error discussion.	Accurate result with detailed and relevant error analysis.
Viva-Voce (Subject Knowledge)	12 marks	Unable to answer any questions. No conceptual understanding.	Answered few questions with poor logic.	Answered half of the questions with average understanding.	Answered most questions with clarity and confidence.	Answered all questions with depth, clarity, and reasoning.

EXPERIMENT – 1

Concept Design using E–R Model

Aim: To design an Entity–Relationship (E–R) model for a real-world application and represent entities, attributes, and relationships clearly.

The Entity–Relationship (E–R) Model is a high-level conceptual data model used to design and represent the structure of a database.

It describes:

- **Entities** (objects)
- **Attributes** (properties of entities)
- **Relationships** (association between entities)

1. Entity:

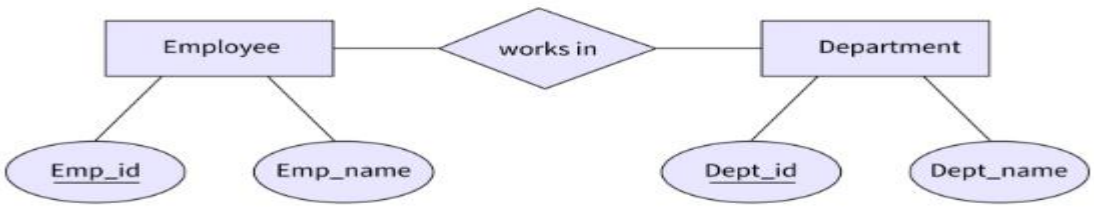
An entity is a real-world object that can be uniquely identified.

Examples:

- Student
- Course
- Employee

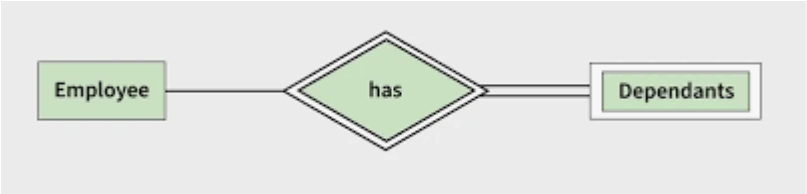
2. Types of Entities:

Strong Entity: A strong entity is an entity that has its own primary key and can exist independently.
Example: Student, Employee



Weak Entity: A weak entity is an entity that does not have a complete primary key and depends on another entity for its existence.

Example: Dependent of Employee



- Each employee can have **one or more dependents**
- **Dependents cannot exist independently** (they depend on Employee for identification)
- So, **Dependents is a weak entity**, and **Employee is a strong entity**

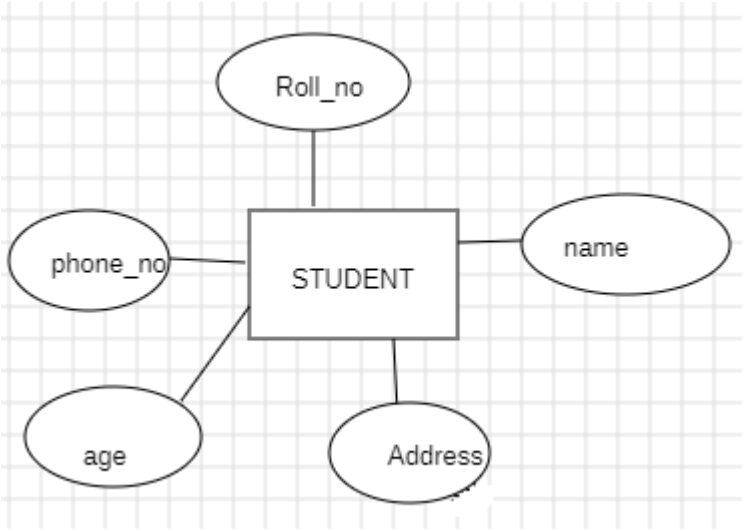
3. Attributes:

Attributes describe the properties of an entity.

Types of Attributes:

- **Simple:** Age, Name

- **Composite:** Address (Street, City)
- **Multi-valued:** Phone numbers
- **Derived:** Age (from DOB)

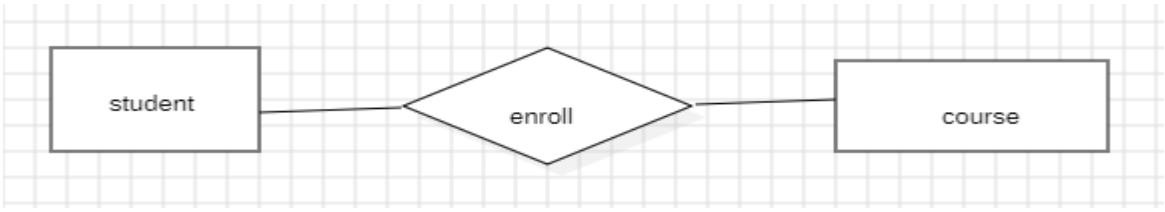


4. Relationship:

A relationship defines how entities are connected.

Examples:

- Student **enrolls in** Course
- Employee **works in** Department



Notation/Symbols of ER Diagram

- Rectangle → Entity
- Ellipse → Attribute
- Diamond → Relationship
- Double ellipse → Multivalued attribute
- Dashed ellipse → Derived attribute
- Double rectangle → Weak entity

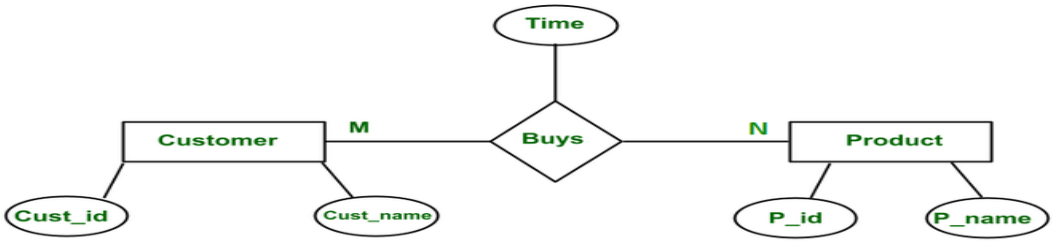
Key Attribute: A key attribute uniquely identifies an entity

Example: Student_ ID

Cardinality ratio:

Cardinality ratio defines the maximum number of relationship instances in which an entity can participate, such as One-to-One (1:1), One-to-Many (1:M), and Many-to-Many (M:N).

Example :



Example 1: ENROLLS (Student ↔ Course)
Many students → Many courses (**M:N**)

- A student can enroll in many courses
- A course can have many students

Example2: TEACHES (Faculty ↔ Class)

One faculty → Many classes (**1 : N)**

- One faculty teaches many classes

Total Participation:

Total participation means every entity must compulsorily participate in a relationship and is represented using double lines.

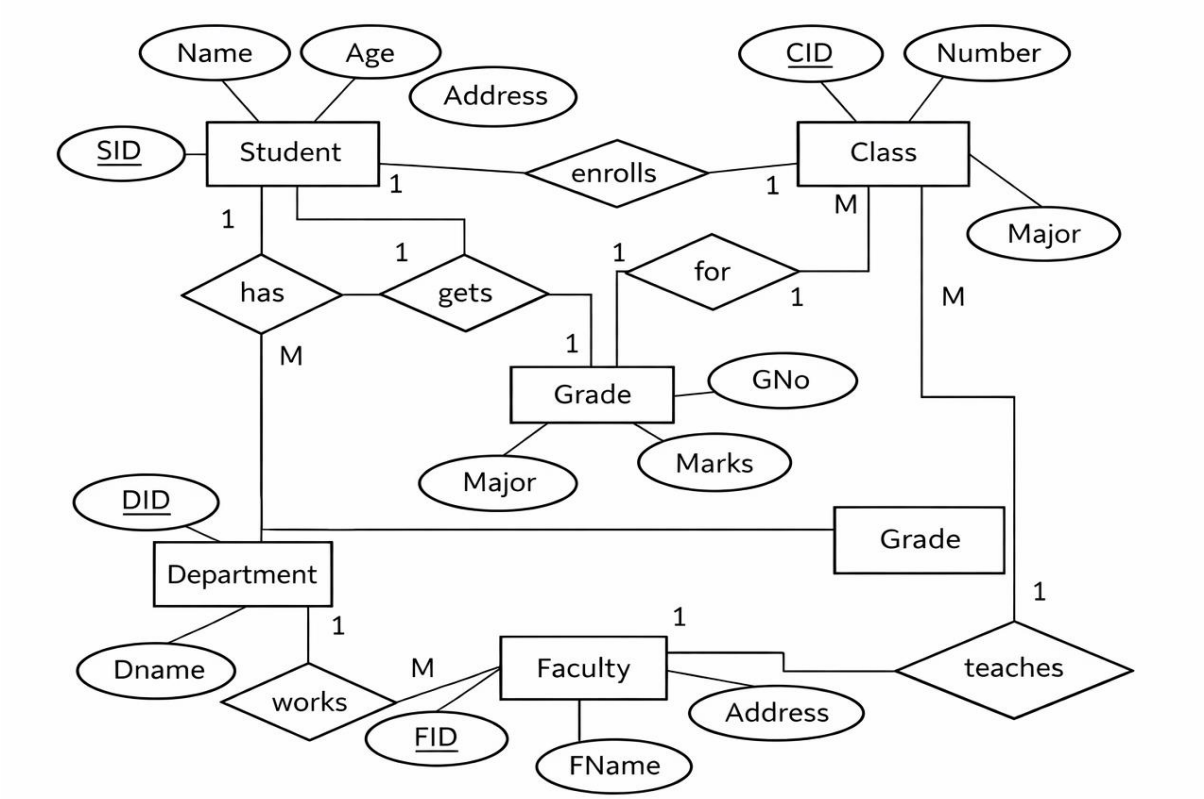
Example:
Every Student must enroll in at least one Course
→ Student has Total Participation in ENROLLS

Partial Participation:

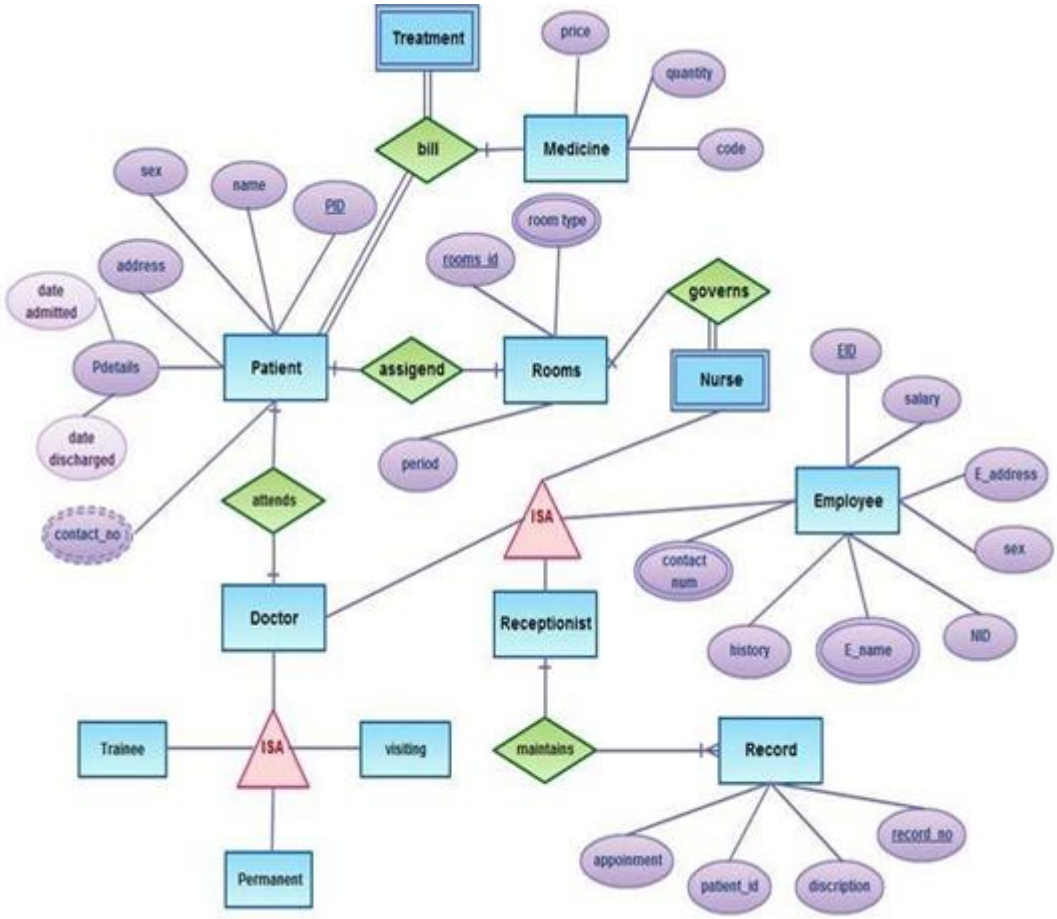
Partial participation means some entities may or may not participate in a relationship and is represented using a single line.

Example:
A Faculty may or may not teach a Course
→ Faculty has Partial Participation in TEACHES.

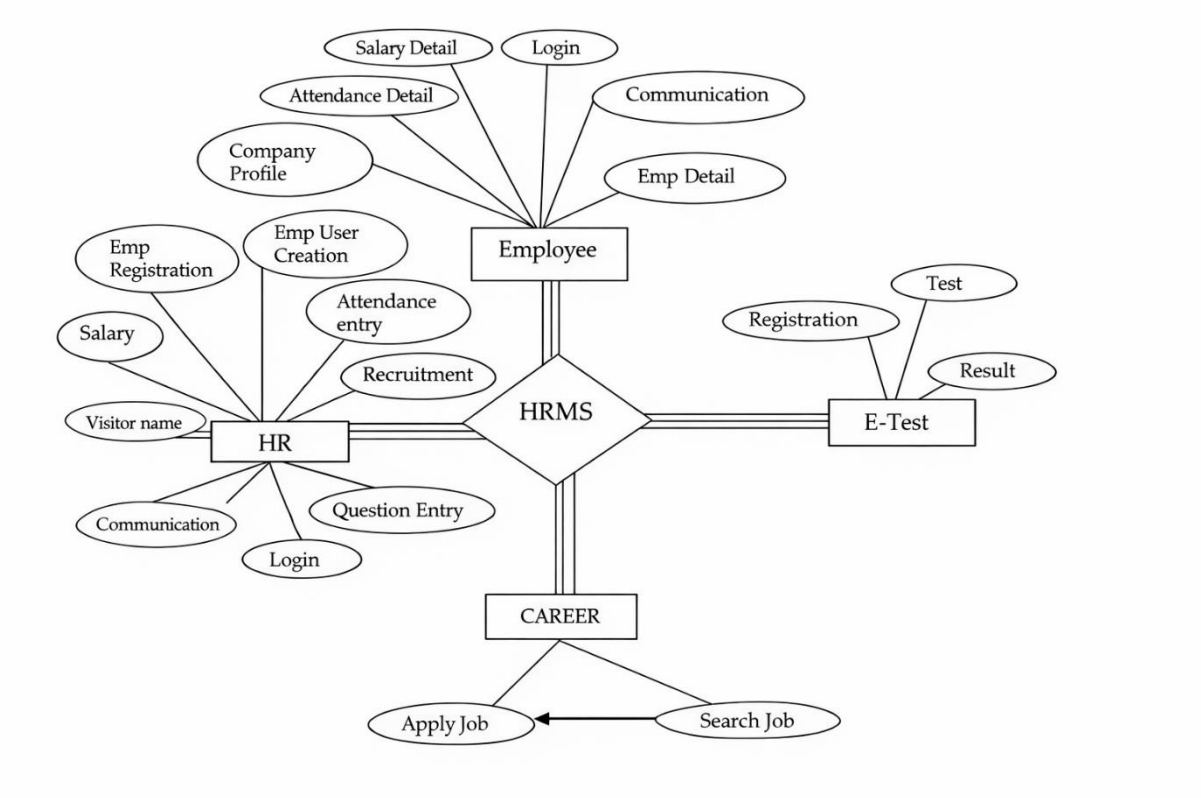
1. Draw the ER Diagram for UNIVERSITY



2. Draw the ER Diagram for Hospital management Systems.



3. Draw the ER Diagram for HR management Systems



VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is a Database?	CO1	Remember
2	What is DBMS?	CO1	Remember
3	What is an E-R model?	CO1	Remember
4	What is an entity set?	CO1	Remember
5	What are types of relationships?	CO1	Remember
6	Explain 1:1, 1:M, and M: N relationships with examples.	CO1	Understanding
7	How is a multivalued attribute represented?	CO1	Understanding
8	How is a derived attribute represented?	CO1	Understanding
9	Can a relationship have attributes? Give an example.	CO1	Understanding
10	Why is E-R modeling important in database design?	CO1	Understanding
11	Give a real-world example of an E-R model.	CO1	Apply
12	What is an identifying relationship?	CO1	Remember
13	What happens if relationships are not properly defined?	CO1	Apply
14	How do you identify entities in a real-world problem?	CO1	Apply
15	How do you decide cardinality between two entities?	CO1	Apply
16	What is the difference between entity and table?	CO1	Understanding
17	Can an entity exist without attributes? Why?	CO1	Understanding
18	How do you represent a weak entity in an E-R diagram?	CO1	Apply
19	How is a composite attribute represented?	CO1	Understanding
20	Difference between total and partial participation?	CO1	Understanding

EXPERIMENT – 2

Relational Model

AIM: To study the relational model and represent data in the form of tables using keys, constraints, and relationships.

The relational model is a high-level data model used to organize data in the form of tables (relations). Each table consists of rows (tuples) and columns (attributes). It was proposed by E.F. Codd.

In this model, data is stored in a structured format and relationships between tables are established using keys such as primary key and foreign key.

Simple Table Representation

STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Tom	New York	1234567890
2	John	Los Angeles	2233657960
3	Alice	Chicago	1754689650

Relation: A table in a database.

Tuple: A row in a table representing a record.

Attribute: A column in a table representing a property.

Domain: The set of possible values for an attribute.

Degree: Number of attributes in a relation.

Cardinality: Number of tuples in a relation.

Keys

- **Super Key**: A set of one or more attributes that uniquely identifies a tuple.

  EX : For the STUDENT table above, STUD_NO and PHONE, as this combination uniquely identifies a student.

- **Candidate Key**: The minimal set of attributes that can uniquely identify a tuple is known as a **candidate key.**

  EX: For the STUDENT table above, STUD_NO can be a candidate key, as it uniquely identifies each record.

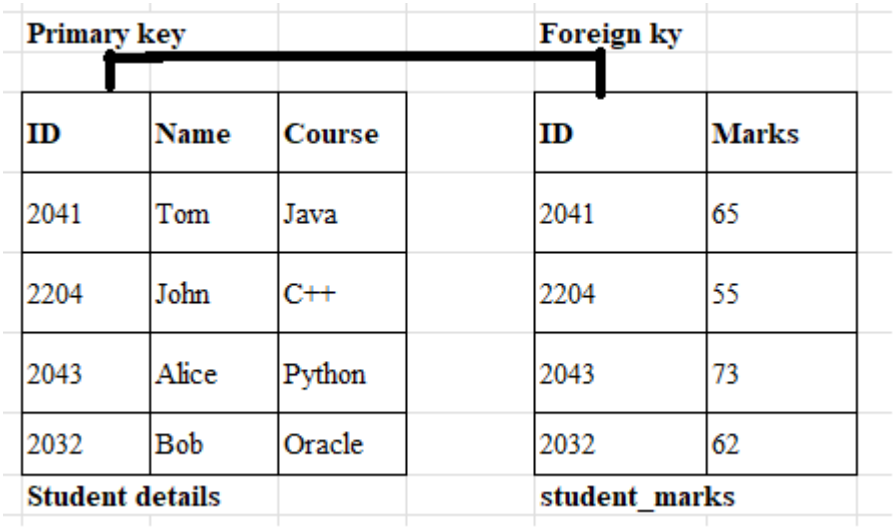
- **Primary Key**: A **primary key** is chosen from the set of candidate keys to uniquely identify each record in a table.

  For example, in the STUDENT table, both STUD_NO and STUD_PHONE can be candidate keys, but STUD_NO is selected as the primary key.

- **Alternate Key**: An **alternate key** is any candidate key in a table that is not chosen as the primary key. In other words, all the keys that are not selected as the primary key are considered alternate keys.

  Ex: In the STUDENT table, both STUD_NO and PHONE are candidate keys. If STUD_NO is chosen as the primary key, then PHONE would be considered an alternate key.

- **Foreign Key**: An attribute that refers to the primary key of another table.(or) A **foreign key** is an attribute in one table that refers to the primary key in another table.



ID	Name	Course
2041	Tom	Java
2204	John	C++
2043	Alice	Python
2032	Bob	Oracle

Student details

ID	Marks
2041	65
2204	55
2043	73
2032	62

student_marks

Integrity Constraints

Integrity constraints are rules used to maintain accuracy and consistency of data in a database.

1. Domain Constraint: Ensures that an attribute contains valid values only

- Each column has a data type and range

Example:

Age → should be a number (not text)

Marks → between 0 and 100

- Invalid values are not allowed

2. Entity Integrity Constraint: Ensures that the Primary Key is NOT NULL

- Every table must have a primary key
- Each record must be uniquely identified

Example:

Student_ID cannot be NULL

- No two rows can have the same primary key

3. Referential Integrity Constraint: Maintains correct relationship between tables

- Foreign Key must match a Primary Key in another table
  Or it can be NULL

Example:

Student_ID in ENROLLMENT must exist in STUDENT table

- Prevents invalid or unrelated data

Schema and Instance

1. Relation Schema: Structure (design) of a table

Example:

STUDENT (Student_ID, Name, Age)

- This shows how the table is defined

2. Instance: Actual data stored in the table at a particular time

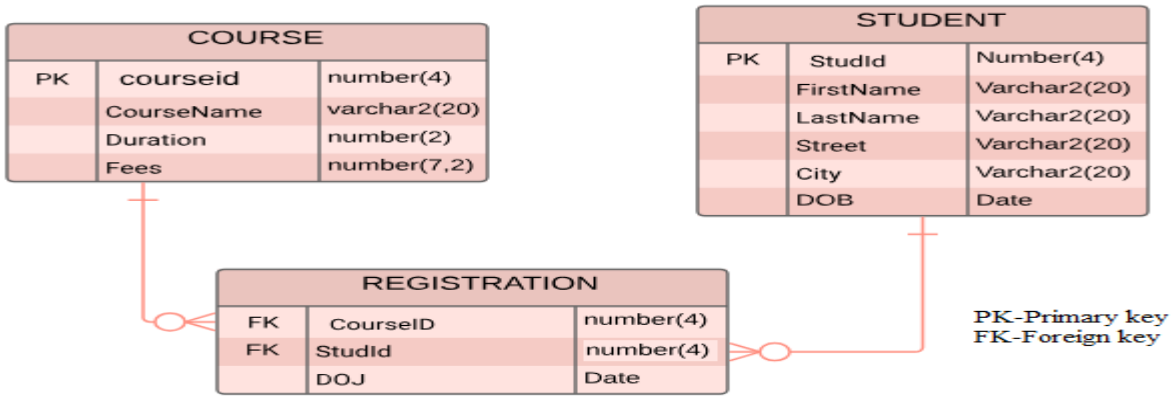
Example:

Student_ID	Name	Age
101	Ram	20
102	Ravi	21

- **Data can change over time**

Relational Database Structure

In the relational model, data is stored in tables with rows and columns. Tables are related to each other using keys, especially foreign keys, to maintain relationships between data.



COURSE → REGISTRATION
Near REGISTRATION → crow's foot (many)
Near COURSE → single line (one)
i. e, One course → many registrations (1 : M)
STUDENT → REGISTRATION
Near REGISTRATION → **many**
Near STUDENT → **one**
i. e., One student → many registrations (**1 : M**)
STUDENT ↔ COURSE
Many-to-Many (M : N)

Mapping ER Model to Relational Model

- Strong entities are represented as separate tables.
- Weak entities are represented as tables with a foreign key referencing the strong entity.
- The primary key of a weak entity is formed by combining its partial key with the primary key of the strong entity.
- 1:1 and 1:M relationships are represented using foreign keys, while M:N relationships are represented using separate tables.
- Attributes are represented as columns in tables.
- Composite attributes are divided into simple attributes.
- Multivalued attributes are represented as separate tables.
- Derived attributes are generally not stored in the database.

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is a relational model?	CO1	Remember
2	What is a relation in DBMS?	CO1	Remember
3	What is a table in relational model?	CO1	Remember
4	What is a tuple?	CO1	Remember
5	Why are keys important in relational model?	CO1	Understand
6	What is a composite key?	CO1	Remember
7	How is a strong entity converted into a table?	CO1	Understand
8	How is a weak entity represented in relational model?	CO1	Understand
9	How is a one-to-many relationship represented as tables?	CO1	Understand
10	How is a many-to-many relationship represented?	CO1	Understand
11	What is a composite attribute? How is it represented in tables?	CO1	Understand
12	What is a multi-valued attribute? How is it handled?	CO1	Understand
13	What is a derived attribute? Give an example.	CO1	Understand
14	Why derived attributes are usually not stored?	CO1	Understand
15	How do you handle multi-valued attributes in relational model?	CO1	Apply
16	What is referential integrity?	CO1	Remember
17	What happens if referential integrity is violated?	CO1	Understand
18	What is redundancy in relational databases?	CO1	Remember
19	Why do we convert ER model to relational model?	CO1	Understand
20	What is the difference between schema and instance in a relational model?	CO1	Understand

EXPERIMENT – 3

Normalization

AIM: To study normalization and apply normalization techniques to organize data in a database and eliminate redundancy.

Normalization:

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller tables and defining relationships between them.

The main objective of normalization is to eliminate data anomalies such as insertion, deletion, and update anomalies.

Types of Anomalies

1. Insertion Anomaly

Difficulty in inserting data without other related data.

2. Deletion Anomaly

Loss of important data when some data is deleted.

3. Update Anomaly

Inconsistency in data when updating values.

Normalization Forms

First Normal Form (1NF)

- All attributes must have atomic (single) values
- No repeating groups or multivalued attributes

Example:

Before 1NF:

Student_ID	Student_Name	Courses
101	Ravi	DBMS, SQL
102	Anu	Java, Python

Problem: **Courses column has multiple values** → violates 1NF

After 1NF:

We split multi-valued attributes into separate row

Student

Student_ID	Student_Name	Courses
101	Ravi	DBMS
101	Ravi	SQL
102	Anu	Java
102	Anu	Python

- Each field contains atomic values
- No repeating groups

Second Normal Form (2NF)

- Must be in 1NF
- No partial dependency (non-key depends on part of key)

Example:

Before 2NF:

Student_ID	course_ID	Student_Name
101	C1	Ravi
101	C2	Ravi
102	C1	Anu
102	C3	Anu

Problem: **Student_Name depends only on Student_ID**, not on full key (Student_ID + Course_ID)

After 2NF:

STUDENT

Student_ID	Student_Name
101	Ravi
102	Anu

ENROLLMENT

Student_ID	course_ID
101	C1
101	C2
102	C1
102	C3

- No partial dependency
- Non-key attributes depend on full key

Third Normal Form (3NF)

- Must be in 2NF
- No transitive dependency

Example:

Before 3NF:

Student_ID	Dept_ID	Dept_Name
101	D1	CSE
102	D2	ECE
103	D1	CSE

Problem: **Dept_Name depends on Dept_ID**, not directly on Student_ID
→ Transitive dependency

After 3NF:

STUDENT

Student_ID	Dept_ID
101	D1
102	D2
103	D1

DEPARTMENT

Dept_ID	Dept_Name
D1	CSE
D2	ECE

- No transitive dependency
- Clean separation of data

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is normalization in DBMS?	CO2	Remember
2	Why is normalization important?	CO2	Understand
3	What are data anomalies?	CO2	Remember
4	How does normalization reduce redundancy?	CO2	Understand
5	What are the objectives of normalization?	CO2	Remember
6	Name the common normal forms used in DBMS.	CO2	Remember
7	What is first normal form (1NF)?	CO2	Remember
8	What is second normal form (2NF)?	CO2	Remember
9	What is third normal form (3NF)?	CO2	Remember
10	What is BCNF (Boyce-Codd Normal Form)?	CO2	Remember
11	What is a partial dependency?	CO2	Understand
12	What is a transitive dependency?	CO2	Understand
13	How are primary keys used during normalization?	CO2	Understand
14	Give an example of a table in 3NF.	CO2	Apply
15	How is 2NF different from 1NF?	CO2	Analyze
16	How is 3NF different from 2NF?	CO2	Analyze
17	How do foreign keys help maintain relationships after normalization?	CO2	Understand
18	What is the role of normalization in maintaining data integrity?	CO2	Understand
19	What problems occur if normalization is not done?	CO2	Understand
20	How do you decide how many normal forms are required?	CO2	Evaluate

EXPERIMENT – 4

Practicing DDL commands

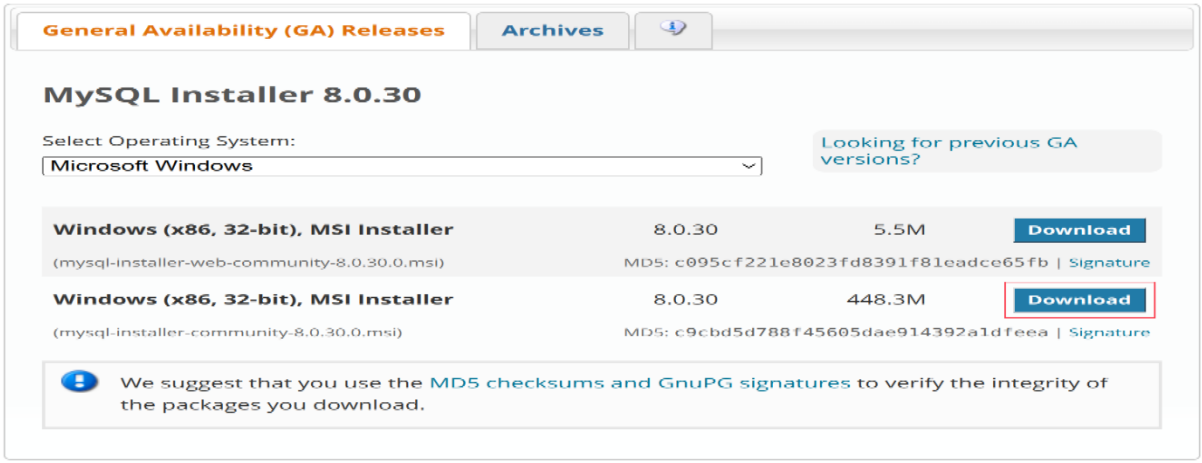
AIM: To study and implement DDL (Data Definition Language) commands using MySQL.

Installation of MySQL

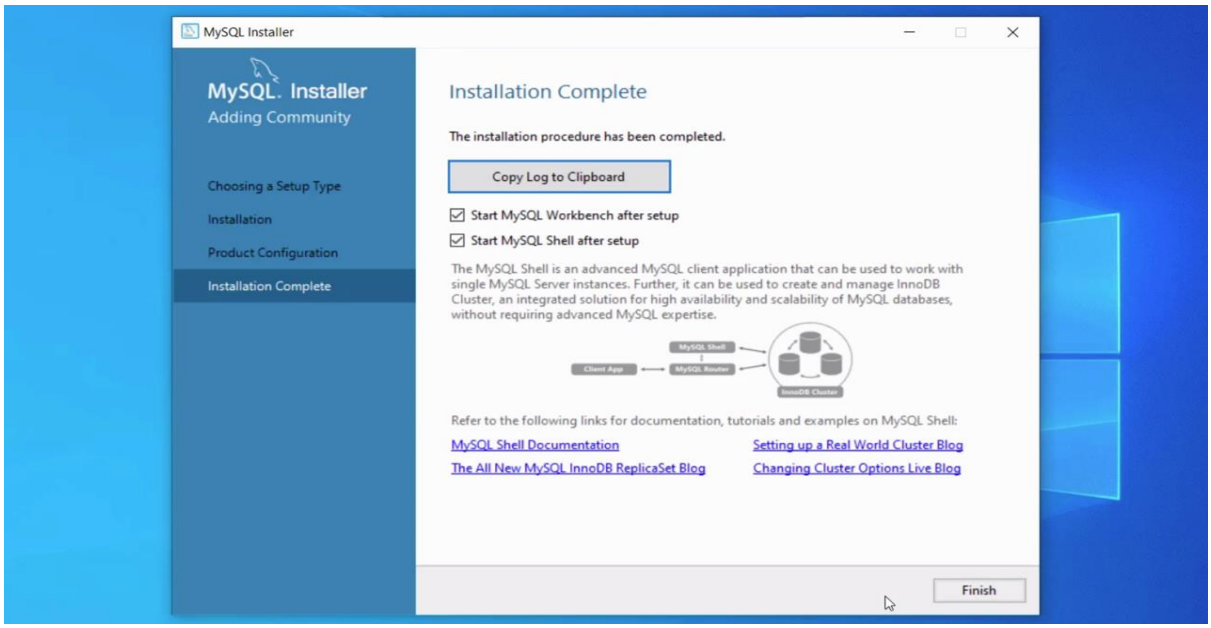
Installation of MySQL. In this week you will learn creating databases, how to create tables, altering the tables, dropping tables and databases if not required. You will also try truncate, rename commands etc

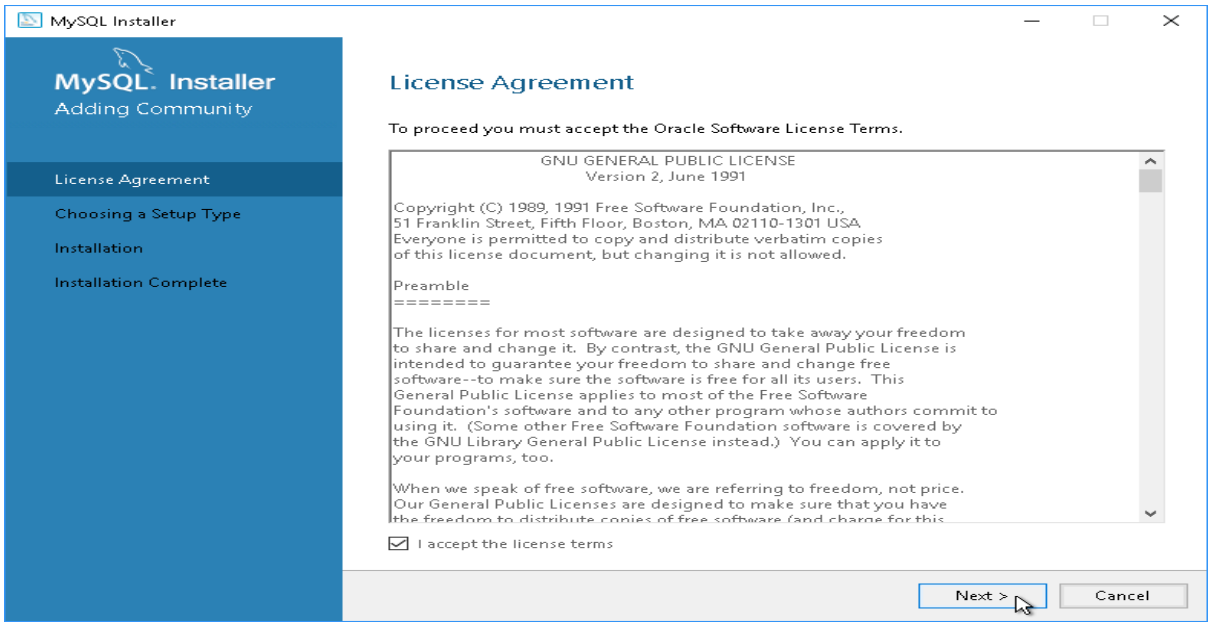
Installation of MySQL 8.0 Client

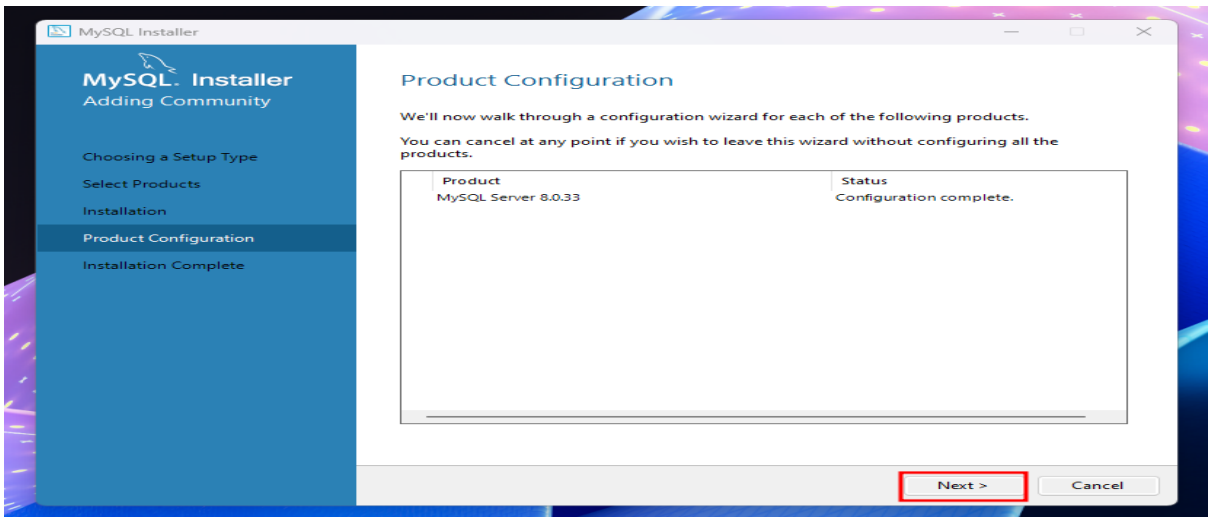
Step 1: Download MySQL Installer



Step 2: Run the Installer





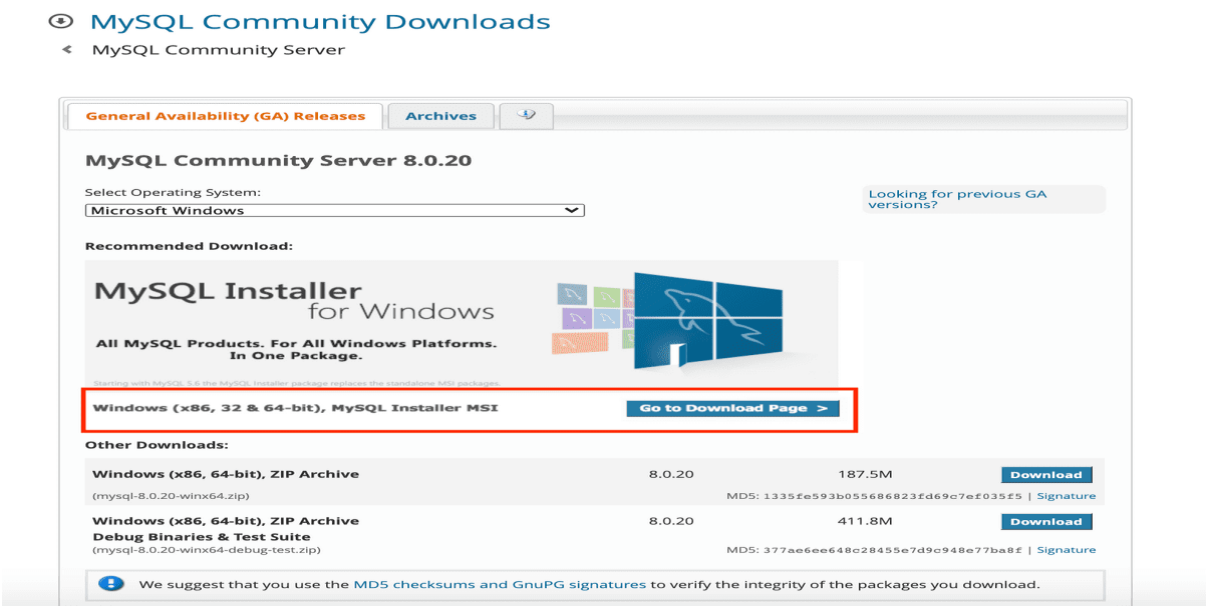


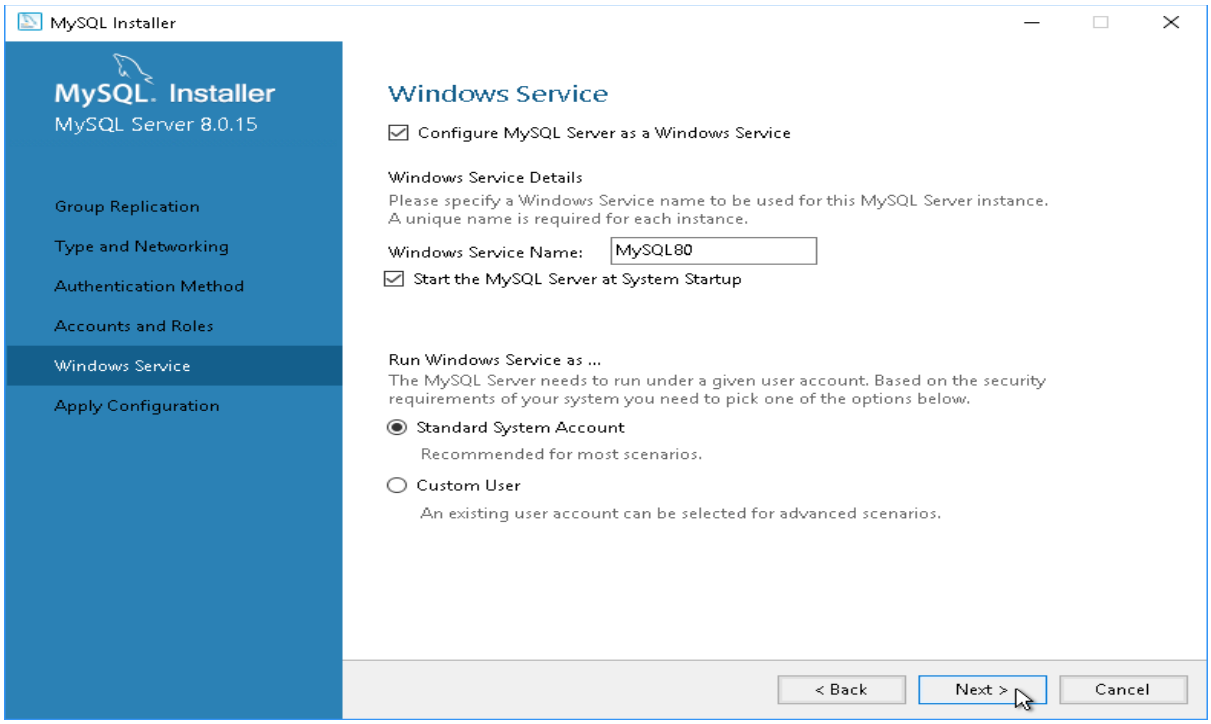
Step 3: Choose Setup Type

Select Developer Default (Recommended).

(This installs Server + Client + Workbench)

Click Next.





Step 4: Install Products

Click Execute to start installation.

Wait until installation completes.

Click Next.

Step 5: Configure MySQL Server

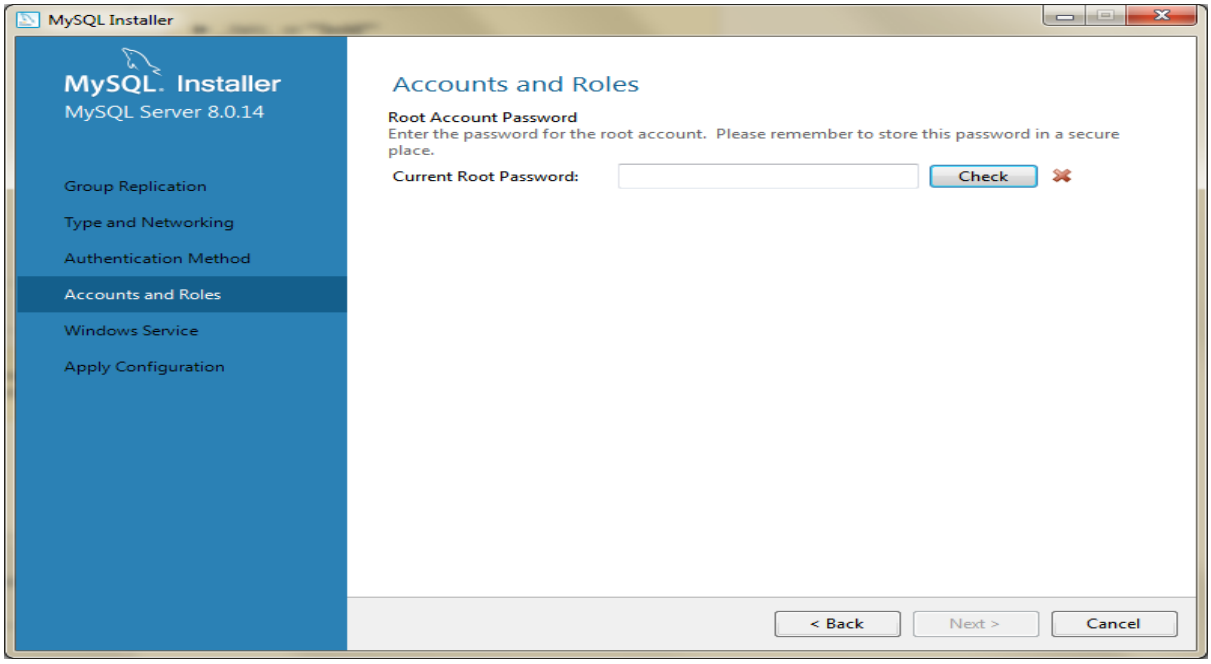
Select Development Computer.

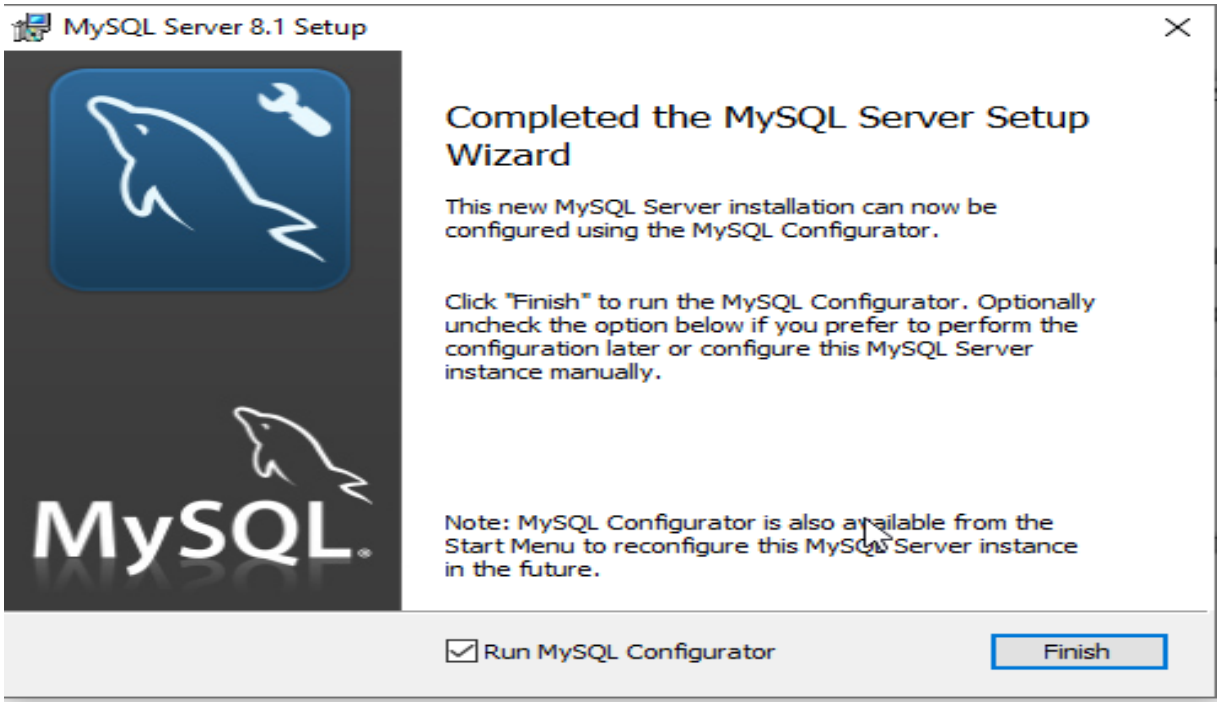
Keep default Port Number: 3306.

Choose authentication method.

Set Root Password.

Click Execute → Finish.





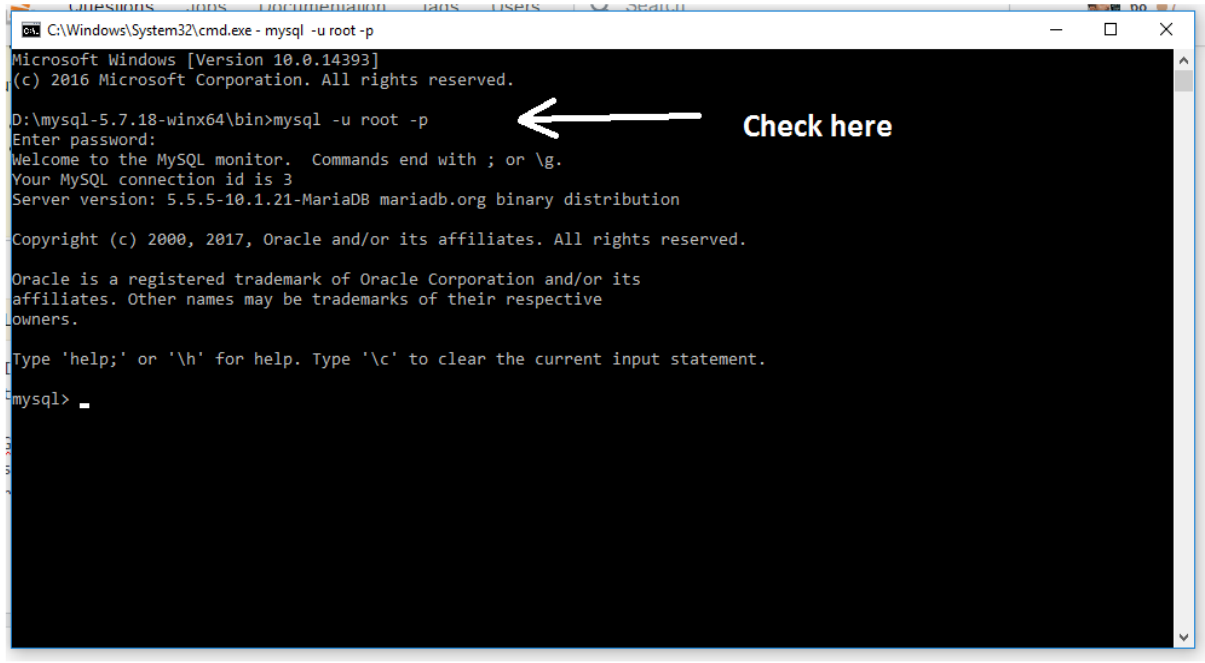
Step 6: Open MySQL 8.0 Client

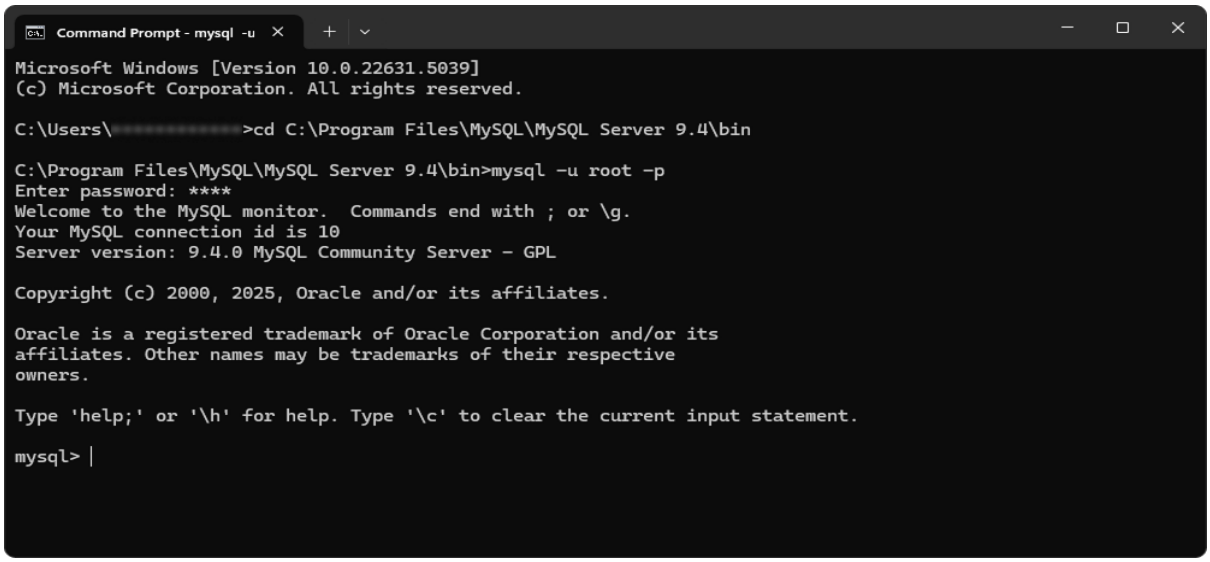
Go to Start Menu.

Click MySQL 8.0 Command Line Client.

Enter the root password.

mysql> prompt appears.





Thus, MySQL 8.0 Client was successfully installed and connected to the MySQL Server.

Practicing DDL commands:

DDL (Data Definition Language)

DDL commands are used to define and modify the structure of database objects like tables, schemas, etc.

DDL Commands

- CREATE – Creates database or table
- ALTER – Modifies table structure
- DROP – Deletes table/database
- TRUNCATE – Removes all records from a table
- RENAME – Renames a table

CREATE DATABASE

```
CREATE DATABASE CollegeDB;
USE CollegeDB;
```

CREATE TABLE

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT
);
```

Output:

```
mysql> Desc student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StudentID | int         | NO   | PRI | NULL    |       |
| Name      | varchar(50) | YES  |     | NULL    |       |
| Age       | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.07 sec)
```

ALTER TABLE (ADD COLUMN)

ALTER TABLE Student ADD Course VARCHAR(20);

Output:

```
mysql> Desc student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StudentID | int         | NO   | PRI | NULL    |       |
| Name      | varchar(50) | YES  |     | NULL    |       |
| Age       | int         | YES  |     | NULL    |       |
| Course    | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

RENAME TABLE

RENAME TABLE Student TO Student_Info;

Output:

```
mysql> Desc student_info;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StudentID | int         | NO   | PRI | NULL    |       |
| Name      | varchar(50) | YES  |     | NULL    |       |
| Age       | int         | YES  |     | NULL    |       |
| Course    | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

TRUNCATE TABLE

TRUNCATE TABLE Student_Info;

Output:

Query ok ,0 rows affected(0.05 sec)

DROP TABLE

DROP TABLE Student_Info;

Output:

```
mysql> Desc student_info;
ERROR 1146 (42S02): Table 'collegedb.student_info' doesn't exist
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is MySQL?	CO3	Remember
2	What is a database in MySQL?	CO3	Remember
3	How do you create a database?	CO3	Apply
4	How do you select a database to use?	CO3	Apply
5	How can you view all databases in MySQL?	CO3	Apply
6	How do you create a table?	CO3	Apply
7	What is a primary key and how do you define it in a table?	CO3	Understand
8	What is the difference between CHAR and VARCHAR data types?	CO3	Understand
9	How do you alter a table in MySQL?	CO3	Apply
10	How do you add a new column to an existing table?	CO3	Apply
11	How do you drop a column from a table?	CO3	Apply
12	How do you rename a table?	CO3	Apply
13	What is the difference between DROP and TRUNCATE commands?	CO3	Evaluate
14	Can a primary key be NULL? Explain.	CO3	Understand
15	What is the syntax of CREATE DATABASE command?	CO3	Remember
16	What is the syntax of DROP TABLE command?	CO3	Remember
17	What is a NOT NULL constraint?	CO3	Remember
18	What is TRUNCATE command and how does it work?	CO3	Understand
19	What are DDL commands and give examples?	CO3	Remember
20	Why is it important to use constraints while creating tables?	CO3	Understand

EXPERIMENT – 5

Practicing DML commands

AIM: To study and implement **Data Manipulation Language (DML)** commands in MySQL.

DML (Data Manipulation Language) is used to manipulate data stored in tables.

DML Commands:

- **INSERT** – Add new records
- **SELECT** – Retrieve data
- **UPDATE** – Modify data
- **DELETE** – Remove data

1. CREATE TABLE

```
CREATE TABLE Student (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(50),
   Age INT,
   Course VARCHAR(20)
);
```

Output:

```
mysql> Desc student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StudentID | int         | NO   | PRI | NULL    |       |
| Name      | varchar(50) | YES  |     | NULL    |       |
| Age       | int         | YES  |     | NULL    |       |
| Course    | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

2. INSERT DATA

```
INSERT INTO Student VALUES
(101, 'Asha', 20, 'BCA'),
(102, 'Ravi', 21, 'BSc'),
(103, 'Kiran', 19, 'BCom');
```

Output:

Query OK, 3 rows affected (0.05 sec)

Records: 3 Duplicates: 0 Warnings: 0

3. SELECT DATA

SELECT * FROM Student;

Output:

```
+-----------+-------+------+---------+
| StudentID | Name  | Age  | Course  |
+-----------+-------+------+---------+
|       101 | Asha  |   20 | BCA     |
|       102 | Ravi  |   21 | BSc     |
|       103 | Kiran |   19 | BCom    |
+-----------+-------+------+---------+
3 rows in set (0.01 sec)
```

4. UPDATE DATA

UPDATE Student
SET Age = 22
WHERE StudentID = 102;

Output:

Query OK, 1 row affected (0.03 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> SELECT * FROM Student;
+-----------+-------+------+---------+
| StudentID | Name  | Age  | Course  |
+-----------+-------+------+---------+
|       101 | Asha  |   20 | BCA     |
|       102 | Ravi  |   22 | BSc     |
|       103 | Kiran |   19 | BCom    |
+-----------+-------+------+---------+
3 rows in set (0.00 sec)
```

5. DELETE DATA

DELETE FROM Student
WHERE StudentID = 103;

Output:

Query OK, 1 row affected (0.01 sec)

```
mysql> SELECT * FROM Student;
+-----------+------+------+---------+
| StudentID | Name | Age  | Course  |
+-----------+------+------+---------+
|       101 | Asha |   20 | BCA     |
|       102 | Ravi |   22 | BSc     |
+-----------+------+------+---------+
2 rows in set (0.00 sec)
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is DML in DBMS?	CO3	Remember
2	What happens if you run UPDATE or DELETE without WHERE clause?	CO3	Evaluate
3	Why is DML important in database management?	CO3	Understand
4	What are the main types of DML commands?	CO3	Remember
5	Give examples of DML commands.	CO3	Understand
6	What is the purpose of the SELECT command?	CO3	Apply
7	How do you select all columns from a table?	CO3	Apply
8	How do you select specific columns from a table?	CO3	Apply
9	How do you filter data using WHERE clause?	CO3	Apply
10	What is the purpose of the INSERT command?	CO3	Understand
11	How do you insert a single row into a table?	CO3	Apply
12	How do you insert multiple rows at once?	CO3	Apply
13	What happens if you insert NULL into a NOT NULL column?	CO3	Understand
14	Can you insert values into selected columns only? How?	CO3	Apply
15	What is the purpose of the UPDATE command?	CO3	Understand
16	How do you update a single row in a table?	CO3	Apply
17	How do you update multiple rows in a table?	CO3	Apply
18	What is the purpose of the DELETE command?	CO3	Understand
19	How do you delete all rows from a table?	CO3	Apply
20	How do you delete a single row?	CO3	Apply

EXPERIMENT – 6

A) Querying (using ANY, ALL, UNION, INTERSECT, JOIN, Constraints etc.)

B) Nested, Correlated subqueries

AIM: To study and implement SQL queries using constraints, joins, set operations, and subqueries in MySQL.

CONSTRAINTS: Constraints are rules applied on table columns to ensure data integrity and accuracy.

Types of Constraints

- PRIMARY KEY
- FOREIGN KEY
- NOT NULL
- UNIQUE

Example:

```
CREATE TABLE Department (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(50) UNIQUE
);
```

Output:

Query OK, 0 rows affected (0.09 sec)

```
CREATE TABLE Student (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(50) NOT NULL,
   DeptID INT,
   FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);
```

Output:

Query OK, 0 rows affected (0.09 sec)

Insert:

INSERT INTO Department (DeptID, DeptName) VALUES

(1, 'CSE'),

(2, 'ECE'),

(3, 'MECH');

Output:

```
mysql> select * from Department;
+--------+----------+
| DeptID | DeptName |
+--------+----------+
|      1 | CSE      |
|      2 | ECE      |
|      3 | MECH     |
+--------+----------+
3 rows in set (0.00 sec)
```

INSERT INTO Student (StudentID, Name, DeptID) VALUES

(101, 'John', 1),

(102, 'Alice', 2),

(103, 'Bob', 1),

(104, 'David', NULL);

Output:

```
mysql> select * from student;
+-----------+-------+--------+
| StudentID | Name  | DeptID |
+-----------+-------+--------+
|       101 | John  |      1 |
|       102 | Alice |      2 |
|       103 | Bob   |      1 |
|       104 | David |   NULL |
+-----------+-------+--------+
4 rows in set (0.00 sec)
```

JOIN OPERATIONS

Definition: JOIN is used to combine data from two or more tables based on a related column.

Types of JOINS :

1. INNER JOIN

Returns only matching records from both tables

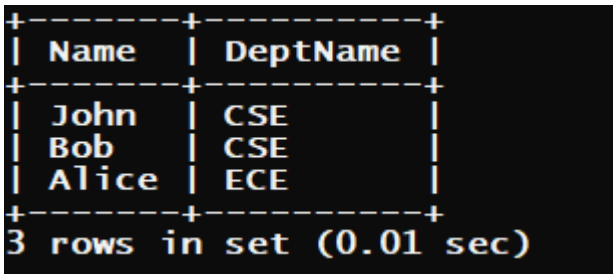
SELECT s.Name, d.DeptName

FROM Student s

INNER JOIN Department d

ON s.DeptID = d.DeptID;

Output:



```
+-------+----------+
| Name  | DeptName |
+-------+----------+
| John  | CSE      |
| Bob   | CSE      |
| Alice | ECE      |
+-------+----------+
3 rows in set (0.01 sec)
```

- Only matching records are shown
- David is NOT shown because DeptID is NULL

2. LEFT JOIN

Returns all records from left table + matching from right

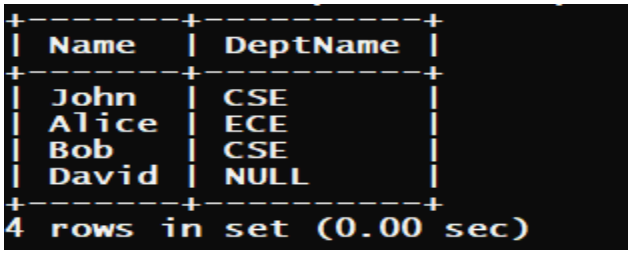
SELECT s.Name, d.DeptName

FROM Student s

LEFT JOIN Department d

ON s.DeptID = d.DeptID;

Output:



```
+-------+----------+
| Name  | DeptName |
+-------+----------+
| John  | CSE      |
| Alice | ECE      |
| Bob   | CSE      |
| David | NULL     |
+-------+----------+
4 rows in set (0.00 sec)
```

LEFT JOIN shows all students, even if no department.

3. RIGHT JOIN

Returns all records from right table + matching from left

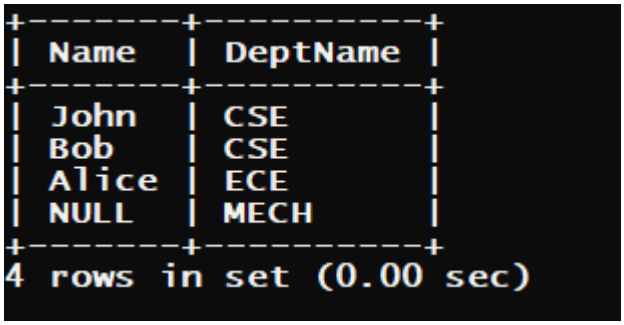
SELECT s.Name, d.DeptName

FROM Student s

RIGHT JOIN Department d

ON s.DeptID = d.DeptID;

Output:



```
+-------+----------+
| Name  | DeptName |
+-------+----------+
| John  | CSE      |
| Bob   | CSE      |
| Alice | ECE      |
| NULL  | MECH     |
+-------+----------+
4 rows in set (0.00 sec)
```

- Show all departments
- Show student names only if they belong to that department
- MECH has no students → so **NULL**

4. FULL JOIN (Important Note for MySQL)

MySQL does NOT support FULL JOIN directly

Use UNION of LEFT + RIGHT JOIN

SELECT s.Name, d.DeptName

FROM Student s

LEFT JOIN Department d

ON s.DeptID = d.DeptID

UNION

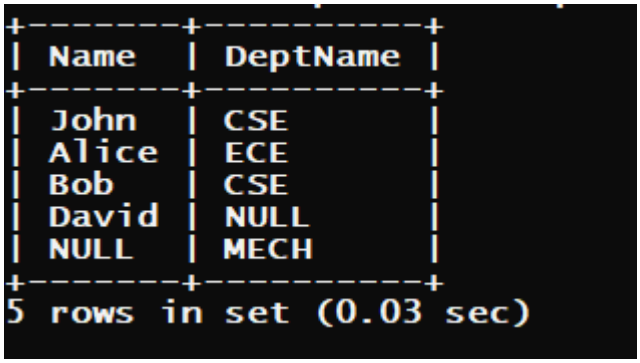
SELECT s.Name, d.DeptName

FROM Student s

RIGHT JOIN Department d

ON s.DeptID = d.DeptID;

Output:



```
+-------+----------+
| Name  | DeptName |
+-------+----------+
| John  | CSE      |
| Alice | ECE      |
| Bob   | CSE      |
| David | NULL     |
| NULL  | MECH     |
+-------+----------+
5 rows in set (0.03 sec)
```

- Show all students
- Show all departments
- Even if no match exists

SET OPERATIONS

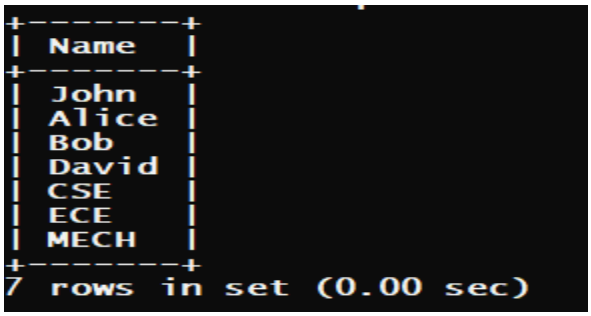
UNION: UNION is used to combine results of two queries and return unique values (duplicates are removed).

SELECT Name FROM Student

UNION

SELECT DeptName FROM Department;

Output:



```
+-------+
| Name  |
+-------+
| John  |
| Alice |
| Bob   |
| David |
| CSE   |
| ECE   |
| MECH  |
+-------+
7 rows in set (0.00 sec)
```

INTERSECT

SELECT Name

FROM Student

WHERE Name IN (SELECT DeptName FROM Department);

Output:

Empty set (0.01 sec)

NESTED SUBQUERIES

Definition: A subquery is a query inside another query.

Example

SELECT Name

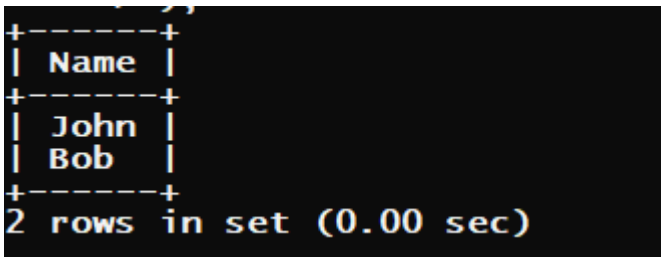
FROM Student

WHERE DeptID IN (

SELECT DeptID FROM Department WHERE DeptName = 'CSE'

);

Output:



CORRELATED SUBQUERIES

Definition: A correlated subquery depends on the outer query and executes repeatedly.

Example

SELECT Name

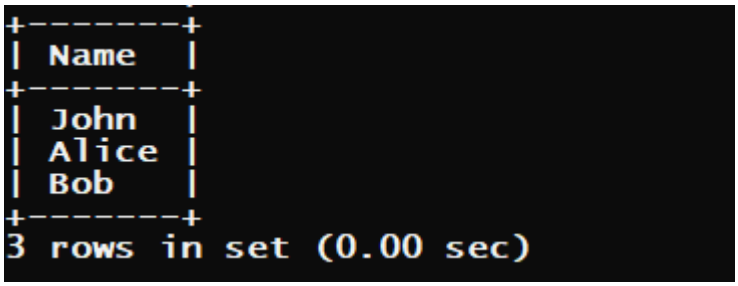
FROM Student s

WHERE DeptID = (

SELECT DeptID

```
FROM Department d
   WHERE s.DeptID = d.DeptID
);
```

Output:

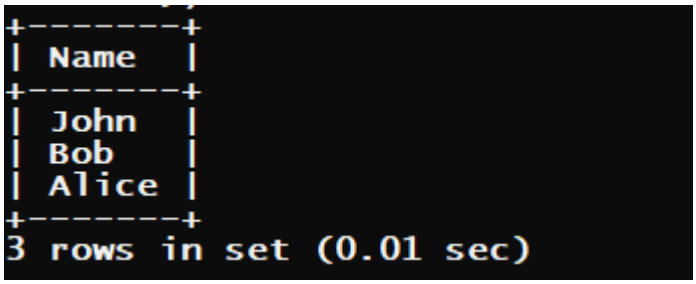


ANY, ALL, EXISTS OPERATORS

ANY

```
SELECT Name
FROM Student
WHERE DeptID = ANY (
   SELECT DeptID FROM Department
);
```

Output:



ALL

```
SELECT Name
FROM Student
WHERE DeptID > ALL (
   SELECT DeptID FROM Department);
```

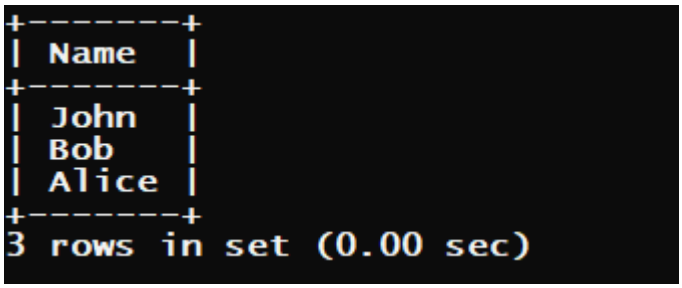
Output:

Empty set (0.01 sec)

EXISTS

```
SELECT Name
FROM Student s
WHERE EXISTS (
    SELECT * FROM Department d
    WHERE s.DeptID = d.DeptID
);
```

Output:



VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What are the types of SQL queries?	CO4	Remember
2	What is the difference between a simple query and a subquery?	CO4	Understand
3	What is the difference between UNION and UNION ALL?	CO4	Analyze
4	How do you use a subquery in the WHERE clause?	CO4	Apply
5		CO4	Understand
6	What is the ANY operator in SQL?	CO4	Remember
7	What is the ALL operator in SQL?	CO4	Remember
8	How can INTERSECT be implemented in MySQL?	CO4	Evaluate
9	Which is better: JOIN or subquery?	CO4	Analyze
10	What is the difference between INNER JOIN and LEFT JOIN?	CO4	Analyze
11	What is the difference between UNION and INTERSECT?	CO4	Analyze
12		CO4	Understand
13	What is the purpose of the LIKE operator?	CO4	Understand
14	How do you use BETWEEN operator in SQL?	CO4	Apply
15	What is the difference between ORDER BY and GROUP BY?	CO4	Analyze
16	How do you write a query using UNION?	CO4	Apply
17	What is a correlated subquery?	CO4	Understand
18	What is the difference between EXISTS and IN?	CO4	Analyze
19	Why do we use DISTINCT in queries?	CO4	Understand
20	How do constraints affect the output of queries?	CO4	Understand

EXPERIMENT – 7

Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.

Aim: Practice queries using Aggregate functions (COUNT, SUM, AVG, MAX, and MIN), GROUP BY, HAVING Clause and Creation and dropping of VIEWS.

Create Sales table

```
CREATE TABLE Sales (
    SaleID INT,
    ProductName VARCHAR(50),
    Amount DECIMAL(10, 2),
    SaleDate DATE
);
```

Insert sample data

```
INSERT INTO Sales (SaleID, ProductName, Amount, SaleDate)
VALUES
(1, 'Laptop', 1200.00, '2025-01-01'),
(2, 'Smartphone', 800.00, '2025-01-03'),
(3, 'Tablet', 500.00, '2025-01-04'),
(4, 'Laptop', 1100.00, '2025-02-01'),
(5, 'Smartphone', 750.00, '2025-02-03'),
(6, 'Tablet', 600.00, '2025-02-05');
```

1. COUNT: Total number of sales

SELECT COUNT(*) AS TotalSales FROM Sales;

```
mysql> SELECT COUNT(*) AS TotalSales FROM Sales;
+------------+
| TotalSales |
+------------+
|          6 |
+------------+
1 row in set (0.04 sec)
```

2. SUM: Total revenue from all sales

SELECT SUM(Amount) AS TotalRevenue FROM Sales;

```
mysql> SELECT SUM(Amount) AS TotalRevenue FROM Sales;
+--------------+
| TotalRevenue |
+--------------+
|      4950.00 |
+--------------+
1 row in set (0.01 sec)
```

3. AVG: Average sale amount

SELECT AVG(Amount) AS AverageSale FROM Sales;

```
mysql> SELECT AVG(Amount) AS AverageSale FROM Sales;
+-------------+
| AverageSale |
+-------------+
|  825.000000 |
+-------------+
1 row in set (0.00 sec)
```

4. MAX: Highest sale amount

SELECT MAX(Amount) AS MaxSale FROM Sales;

```
mysql> SELECT MAX(Amount) AS MaxSale FROM Sales;
+---------+
| MaxSale |
+---------+
| 1200.00 |
+---------+
1 row in set (0.01 sec)
```

5. MIN: Lowest sale amount

SELECT MIN(Amount) AS MinSale FROM Sales;

```
mysql> SELECT MIN(Amount) AS MinSale FROM Sales;
+---------+
| MinSale |
+---------+
|  500.00 |
+---------+
1 row in set (0.00 sec)
```

6. Group sales by ProductName and calculate the total sales

SELECT ProductName, SUM(Amount) AS TotalSales

FROM Sales

GROUP BY ProductName;

```
mysql> SELECT ProductName, SUM(Amount) AS TotalSales
    -> FROM Sales
    -> GROUP BY ProductName;
+-------------+------------+
| ProductName | TotalSales |
+-------------+------------+
| Laptop      |    2300.00 |
| Smartphone  |    1550.00 |
| Tablet      |    1100.00 |
+-------------+------------+
```

7. Group sales by ProductName and filter those with total sales greater than 1500

SELECT ProductName, SUM(Amount) AS TotalSales

FROM Sales

GROUP BY ProductName

HAVING SUM(Amount) > 1500;

```
+-------------+------------+
| ProductName | TotalSales |
+-------------+------------+
| Laptop      |    2300.00 |
| Smartphone  |    1550.00 |
+-------------+------------+
2 rows in set (0.01 sec)
```

8. Create a View for total sales per product

CREATE VIEW TotalSalesView AS

SELECT ProductName, SUM(Amount) AS TotalSales

FROM Sales

GROUP BY ProductName;

Output: query ok , 0 rows effected.

9. View the created view

SELECT * FROM TotalSalesView;

```
mysql> SELECT * FROM TotalSalesView;
+-------------+------------+
| ProductName | TotalSales |
+-------------+------------+
| Laptop      |    2300.00 |
| Smartphone  |    1550.00 |
| Tablet      |    1100.00 |
+-------------+------------+
3 rows in set (0.01 sec)
```

10. Drop the View if it's no longer needed

DROP VIEW TotalSalesView;

Output:

```
mysql> SELECT * FROM TotalSalesView;
ERROR 1146 (42S02): Table 'college.totalsalesview' doesn't exist
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	List common aggregate functions in SQL.	CO4	Remember
2	Why aggregate functions are used in SQL?	CO4	Understand
3	How does sum () handle NULL values?	CO4	Remember
4	What is the difference between COUNT (*) and COUNT (column_ name)?	CO4	Understand
5	Write the syntax to apply group by clause in SQL	CO4	Remember
6	Write the syntax to apply having clause in SQL.	CO4	Remember
7	How does GROUP BY handle NULL values?	CO4	Remember
8	What is the difference between using WHERE and HAVING with aggregate functions? Give examples.	CO4	Analyze
9	How do aggregate functions handle duplicate values?	CO4	Remember
10	What is the purpose of GROUP BY Clause?	CO4	Understand
11	What is a view in SQL?	CO4	Remember
12	Can we use aliases in the HAVING clause?	CO4	Remember
13	Can we insert, update, or delete data using a view?	CO4	Understand
14	What is a simple view and how is it different from a complex view?	CO4	Understand
15	Can we use aggregate functions without GROUP BY ? Example?	CO4	Apply
16	What are the limitations of views in SQL?	CO4	Understand
17	Can we create a view from multiple tables?	CO4	Apply
18	What is the order of execution: WHERE, GROUP BY, HAVING ?	CO4	Analyze
19	Can a view call another view?	CO4	Remember
20	What is the impact of changing the base table on a view?	CO4	Analyze

EXPERIMENT – 8

Triggers (Creation of insert trigger, delete trigger, update trigger)

Aim: To create and implement INSERT, DELETE, and UPDATE triggers in MySQL and observe their effect on data manipulation operations.

A **trigger** is a database object in **MySQL** that automatically executes (fires) when a specified event occurs on a table.

Triggers are mainly used for:

- Maintaining **audit logs**
- Enforcing **business rules**
- Automatic updates of related tables

Types of Triggers

1. **INSERT Trigger** – Executes when a new record is inserted
2. **UPDATE Trigger** – Executes when a record is updated
3. **DELETE Trigger** – Executes when a record is deleted

Trigger Timing

- **BEFORE Trigger** – Executes before the event
- **AFTER Trigger** – Executes after the event

Create Main Table

```
CREATE TABLE Student (
   ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Marks INT
);
```

Output: Query OK, 0 rows affected (0.01 sec)

Create Log Table

```
CREATE TABLE Student_Log (
   LogID INT AUTO_INCREMENT PRIMARY KEY,

Action VARCHAR(50),
   StudentID INT,
   ActionTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Output: Query OK, 0 rows affected (0.01 sec)

INSERT Trigger

Create INSERT Trigger

```
DELIMITER //
CREATE TRIGGER insert_trigger
AFTER INSERT ON Student
FOR EACH ROW
BEGIN
   INSERT INTO Student_Log(Action, StudentID)
   VALUES ('INSERT', NEW.ID);
END;
//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.01 sec)

Insert Data

```
INSERT INTO Student VALUES (101, 'John', 85);
```

Output: Query OK, 1 row affected (0.01 sec)

```
SELECT * FROM Student;
```

```
mysql> SELECT * FROM Student;
+-----+------+-------+
| ID  | Name | Marks |
+-----+------+-------+
| 101 | John |    85 |
+-----+------+-------+
1 row in set (0.01 sec)
```

SELECT * FROM Student_Log;

```
mysql> SELECT * FROM Student_Log;
+-------+--------+-----------+---------------------+
| LogID | Action | StudentID | ActionTime          |
+-------+--------+-----------+---------------------+
|     1 | INSERT |       101 | 2026-03-28 10:22:16 |
+-------+--------+-----------+---------------------+
1 row in set (0.00 sec)
```

UPDATE Trigger

Create UPDATE Trigger

```
DELIMITER //
CREATE TRIGGER update_trigger
AFTER UPDATE ON Student
FOR EACH ROW
BEGIN
   INSERT INTO Student_Log(Action, StudentID)
   VALUES ('UPDATE', NEW.ID);
END;
//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.01 sec)

Update Data

UPDATE Student SET Marks = 90 WHERE ID = 101;

OUTPUT: Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

Output

SELECT * FROM Student;

```
mysql> SELECT * FROM Student;
+-----+------+-------+
| ID  | Name | Marks |
+-----+------+-------+
| 101 | John |    90 |
+-----+------+-------+
1 row in set (0.00 sec)
```

SELECT * FROM Student_Log;

```
mysql> SELECT * FROM Student_Log;
+-------+--------+-----------+---------------------+
| LogID | Action | StudentID | ActionTime          |
+-------+--------+-----------+---------------------+
|     1 | INSERT |       101 | 2026-03-28 10:22:16 |
|     2 | UPDATE |       101 | 2026-03-28 10:26:16 |
+-------+--------+-----------+---------------------+
2 rows in set (0.00 sec)
```

Step 6: DELETE Trigger

Create DELETE Trigger

```
DELIMITER //

CREATE TRIGGER delete_trigger
AFTER DELETE ON Student
FOR EACH ROW
BEGIN
   INSERT INTO Student_Log(Action, StudentID)
   VALUES ('DELETE', OLD.ID);
END;//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.00 sec)

Delete Data

DELETE FROM Student WHERE ID = 101;

Output: Query OK, 0 rows affected (0.00 sec)

SELECT * FROM Student;

Output: Empty set (0.00 sec)

SELECT * FROM Student_Log;

```
mysql> SELECT * FROM Student_Log;
+-------+--------+-----------+---------------------+
| LogID | Action | StudentID | ActionTime          |
+-------+--------+-----------+---------------------+
|     1 | INSERT |       101 | 2026-03-28 10:22:16 |
|     2 | UPDATE |       101 | 2026-03-28 10:26:16 |
|     3 | DELETE |       101 | 2026-03-28 10:32:05 |
+-------+--------+-----------+---------------------+
3 rows in set (0.00 sec)
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is a trigger in SQL?	CO5	Remember
2	List the SQL events that can activate a trigger.	CO5	Remember
3	On which database objects can triggers be created?	CO5	Remember
4	What keyword is used to create a trigger in SQL?	CO5	Remember
5	Can triggers be created on views?	CO5	Remember
6	What is the syntax keyword used to drop a trigger?	CO5	Remember
7	Are triggers automatically executed or manually executed?	CO5	Remember
8	Can triggers be used with INSERT operations?	CO5	Remember
9	Can triggers be used with DELETE operations?	CO5	Remember
10	What is the difference between BEFORE and AFTER trigger?	CO5	Understand
11	Can we use multiple triggers on a table?	CO5	Remember
12	What are the types of triggers in SQL?	CO5	Remember
13	What is the difference between a trigger and a stored procedure?	CO5	Understand
14	Can a table have multiple triggers for the same event?	CO5	Remember
15	What are the limitations or disadvantages of using triggers?	CO5	Understand
16	How do you disable or drop a trigger?	CO5	Remember
17	What is a BEFORE trigger used for?	CO5	Remember
18	Can triggers be used to enforce constraints?	CO5	Remember
19	What is a recursive trigger? How is it handled?	CO5	Understand
20	What is a row-level and statement-level trigger?	CO5	Remember

EXPERIMENT-9

Stored Procedures

Aim: To create and execute stored procedures in MySQL for performing database operations with and without parameters.

A **Stored Procedure** is a precompiled collection of SQL statements stored in the database and executed as a single unit.

Stored procedures help to:

- Reduce **code redundancy**
- Improve **performance**
- Provide **security** by restricting direct table access

Types of Parameters

1. **IN** – Input parameter (default)
2. **OUT** – Output parameter
3. **INOUT** – Both input and output

Procedure / Steps

Step 1: Create Database

```
CREATE DATABASE college;
USE college;
```

Step 2: Create Table

```
CREATE TABLE Student (
   ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Marks INT
);
```

Output: Query OK, 0 rows affected (0.01 sec)

Step 3: Insert Sample Data

```
INSERT INTO Student VALUES
(101, 'John', 85),
(102, 'Alice', 90),
(103, 'Bob', 75);
```

Output: Query OK, 3 rows affected (0.01 sec)

Records: 3 Duplicates: 0 Warnings: 0

Step 4: Procedure without Parameters

Create Procedure

```
DELIMITER //

CREATE PROCEDURE ShowStudents()
BEGIN
   SELECT * FROM Student;
END;
//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.01 sec)

Execute Procedure

```
CALL ShowStudents();
```

```
mysql> CALL ShowStudents();
+-----+-------+-------+
| ID  | Name  | Marks |
+-----+-------+-------+
| 101 | John  |    85 |
| 102 | Alice |    90 |
| 103 | Bob   |    75 |
+-----+-------+-------+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Step 5: Procedure with IN Parameter

Create Procedure

```
DELIMITER //

CREATE PROCEDURE GetStudent(IN sid INT)
BEGIN
   SELECT * FROM Student WHERE ID = sid;
END;
//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.01 sec)

Execute Procedure

```
CALL GetStudent(102);
```

Output:

```
mysql> CALL GetStudent(102);
+-----+-------+-------+
| ID  | Name  | Marks |
+-----+-------+-------+
| 102 | Alice |    90 |
+-----+-------+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Step 6: Procedure with OUT Parameter

Create Procedure

```
DELIMITER //
CREATE PROCEDURE GetMarks(IN sid INT, OUT smarks INT)
BEGIN
   SELECT Marks INTO smarks
   FROM Student
   WHERE ID = sid;
END;
//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.01 sec)

Execute Procedure

CALL GetMarks(101, @m);

Output: Query OK, 1 row affected (0.01 sec)
SELECT @m;

Output:

```
mysql> SELECT @m;
+------+
| @m   |
+------+
|   85 |
+------+
1 row in set (0.00 sec)
```

Step 7: Procedure with INOUT Parameter

Create Procedure

```
DELIMITER //
CREATE PROCEDURE UpdateMarks(INOUT m INT)
BEGIN
   SET m = m + 10;
END;
//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.01 sec)

Execute Procedure

SET @marks = 80;
CALL UpdateMarks(@marks);

Output: Query OK, 0 rows affected (0.01 sec)
SELECT @marks;

```
mysql> SELECT @marks;
+--------+
| @marks |
+--------+
|     90 |
+--------+
1 row in set (0.00 sec)
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is a stored procedure in SQL?	CO5	Remember
2	What are the advantages of using stored procedures?	CO5	Understand
3	How is a procedure different from a function?	CO5	Understand
4	What are the types of parameters used in procedures?	CO5	Remember
5	Can a procedure return a value? How?	CO5	Remember
6	How do you pass parameters to a procedure?	CO5	Remember
7	How can you modify or update an existing procedure?	CO5	Remember
8	Can a procedure contain control structures like IF, WHILE, etc.?	CO5	Remember
9	What is the purpose of the DELIMITER in MySQL procedures?	CO5	Remember
10	Can a procedure call another procedure?	CO5	Remember
11	What is the syntax to create a stored procedure in SQL?	CO5	Remember
12	What is the difference between CREATE PROCEDURE and CREATE FUNCTION?	CO5	Understand
13	What are IN parameters in a procedure?	CO5	Remember
14	How do you execute a stored procedure in SQL?	CO5	Remember
15	Can a procedure have multiple parameters?	CO5	Remember
16	Can a procedure handle exceptions in SQL?	CO5	Remember
17	What is the purpose of BEGIN…END block in procedures?	CO5	Remember
18	Can a stored procedure return a result set?	CO5	Remember
19	Can a procedure perform DML operations like INSERT, UPDATE, DELETE?	CO5	Remember
20	What is the purpose of using stored procedures over dynamic SQL?	CO5	Understand

EXPERIMENT-10

Usage of Cursors

Aim: To create and use cursors in MySQL for processing data row-by-row from a table.

A **cursor** is a database object used to retrieve data from a result set **one row at a time**.

Unlike normal SQL queries (which work on entire tables), cursors allow:

- Row-by-row processing
- Complex logic implementation
- Iterative operations

Steps in Using a Cursor

1. Declare cursor
2. Open cursor
3. Fetch data
4. Close cursor

Steps:

Step 1: Create Database

```
CREATE DATABASE college;
USE college;
```

Step 2: Create Table

```
CREATE TABLE Student (
   ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Marks INT
);
```

Output: Query OK, 0 rows affected (0.01 sec)

Step 3: Insert Data

```
INSERT INTO Student VALUES
(101, 'John', 85),

(102, 'Alice', 90),
(103, 'Bob', 75);
```

Output: Query OK, 3 rows affected (0.01 sec)

Records: 3 Duplicates: 0 Warnings: 0

Step 4: Create Result Table

```
CREATE TABLE Result (
   ID INT,
   Name VARCHAR(50),
   Grade CHAR(1)
);
```

Output: Query OK, 0 rows affected (0.01 sec)

Step 5: Create Cursor Procedure

Create Procedure Using Cursor

```
DELIMITER //
CREATE PROCEDURE GenerateGrades()
BEGIN
   DECLARE done INT DEFAULT 0;
   DECLARE sid INT;
   DECLARE sname VARCHAR(50);
   DECLARE smarks INT;
   DECLARE cur CURSOR FOR

        SELECT ID, Name, Marks FROM Student;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
   OPEN cur;
   read_loop: LOOP
      FETCH cur INTO sid, sname, smarks;
      IF done = 1 THEN
         LEAVE read_loop;
      END IF
```

```
    IF smarks >= 90 THEN
      INSERT INTO Result VALUES (sid, sname, 'A');
    ELSEIF smarks >= 75 THEN
      INSERT INTO Result VALUES (sid, sname, 'B');
    ELSE
      INSERT INTO Result VALUES (sid, sname, 'C');
    END IF;
  END LOOP;

  CLOSE cur;
END;
//
DELIMITER ;
```

Output: Query OK, 0 rows affected (0.01 sec)

Step 6: Execute Procedure

```
CALL GenerateGrades();
```

Output: Query OK, 0 rows affected (0.01 sec)

Step 7: View Output

```
SELECT * FROM Result;
```

```
mysql> SELECT * FROM Result;
+------+-------+-------+
| ID   | Name  | Grade |
+------+-------+-------+
|  101 | John  | B     |
|  102 | Alice | A     |
|  103 | Bob   | B     |
+------+-------+-------+
3 rows in set (0.00 sec)
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is a Cursor in DBMS?	CO5	Remember
2	Types of Cursors in DBMS?	CO5	Remember
3	What is a Cursor Variable?	CO5	Remember
4	Difference between Static and Dynamic Cursors?	CO5	Understand
5	Explain Cursor Attributes?	CO5	Understand
6	What are the advantages and disadvantages of using cursors?	CO5	
7	What is a FOR loop in context with cursors?	CO5	Remember
8	What is the difference between a read-only cursor and an updatable cursor?	CO5	Understand
9	Can a cursor be used for INSERT, UPDATE, or DELETE operations?	CO5	Remember
10	How is a cursor used to retrieve data from multiple tables?	CO5	Understand
11	What is the purpose of using a cursor in DBMS?	CO5	Understand
12	What is the syntax to declare a cursor?	CO5	Remember
13	Can a cursor fetch one row at a time?	CO5	Remember
14	Can a cursor be used with joins?	CO5	Remember
15	Can cursors be used with stored procedures?	CO5	Remember
16	What is the difference between forward-only and scrollable cursors?	CO5	Understand
17	What is a cursor pointer?	CO5	Remember
18	What is an implicit cursor?	CO5	Remember
19	What is a weak cursor?	CO5	Remember
20	What is a strong cursor?	CO5	Remember

Design a Banking Database System

Aim

To design a database for a banking system to manage customers, accounts, transactions, and loans.

Entities

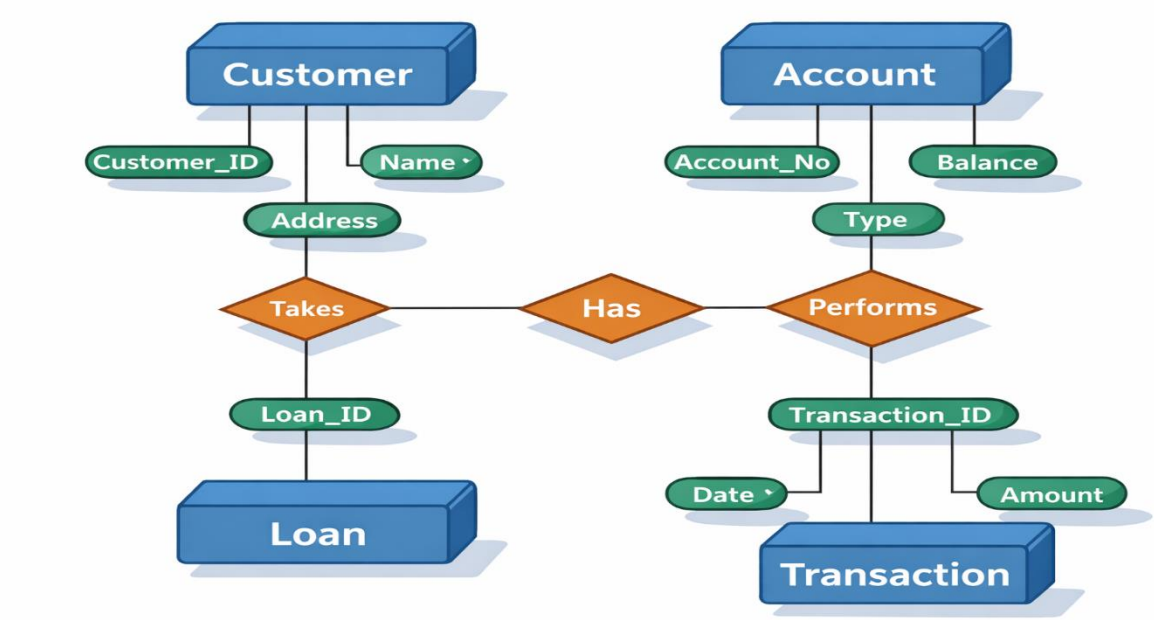
- **Customer** (Customer_ID, Name, Address, Phone)
- **Account** (Account_No, Type, Balance)
- **Transaction** (Transaction_ID, Date, Amount, Type)
- **Loan** (Loan_ID, Amount, Type)

Primary keys:

- **ustomer(Customer_ID)**
- **Account(Account_No)**
- **Transaction(Transaction_ID)**
- **Loan(Loan_ID)**

Relationships

- A customer can have **multiple accounts**
- An account can have **multiple transactions**
- A customer can have **multiple loans**



Relational Schema

- Customer(Customer_ID, Name, Address, Phone)
- Account(Account_No, Type, Balance, Customer_ID)
- Transaction(Transaction_ID, Date, Amount, Type, Account_No)
- Loan(Loan_ID, Amount, Type, Customer_ID)

Normalization

- **1NF:** No repeating groups
- **2NF:** No partial dependency
- **3NF:** No transitive dependency

Database is in **3NF**

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is a relationship in an E–R model?	CO2	
2	What is a foreign key?	CO2	
3	What is normalization?	CO2	
4	What is Third Normal Form (3NF)?	CO2	
5	How does your design reduce data redundancy?	CO2	
6	What happens if foreign keys are not used?	CO2	
7	How are transactions represented in your database?	CO2	
8	How can you improve this banking database system further?	CO2	
9	Can one account belong to multiple customers? Why?	CO2	
10	What is an entity in DBMS?	CO2	
11	What is Fourth Normal Form (4NF)?	CO2	
12	What is Fifth Normal Form (5NF)?	CO2	
13	Why is Customer_ID used as a primary key?	CO2	
14	Why is normalization important in your database design?	CO2	
15	What is the difference between DBMS and RDBMS?	CO2	
16	What is an instance of a database?	CO2	
17	What is data independence?	CO2	
18	What is a data model?	CO2	
19	Difference between primary key and unique key?	CO2	
20	What are the components of an E–R diagram?	CO2	

Hospital Data Management using SQL Queries

Aim: To develop SQL queries to manage and analyze hospital data efficiently.

Tables Used

1. Patient

- **Patient_ID (PK)**
- **Name**
- **Age**
- **Gender**
- **Disease**

2. Doctor

- **Doctor_ID (PK)**
- **Name**
- **Specialization**

3. Appointment

- **Appointment_ID (PK)**
- **Patient_ID (FK)**
- **Doctor_ID (FK)**
- **Date**

1. Create Tables

```
CREATE TABLE Patient (
Patient_ID INT PRIMARY KEY,
Name VARCHAR(50),
Age INT,
Gender VARCHAR(10),
Disease VARCHAR(50)
);
```

```
CREATE TABLE Doctor (
Doctor_ID INT PRIMARY KEY,
Name VARCHAR(50),
Specialization VARCHAR(50)
);
```

CREATE TABLE Appointment (
Appointment_ID INT PRIMARY KEY,
Patient_ID INT,
Doctor_ID INT,
Date DATE,
FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID)
);

2. Insert Sample Data

INSERT INTO Patient VALUES (1, 'Ravi', 30, 'Male', 'Fever');
INSERT INTO Patient VALUES (2, 'Anita', 25, 'Female', 'Cold');

INSERT INTO Doctor VALUES (101, 'Dr. Kumar', 'General');
INSERT INTO Doctor VALUES (102, 'Dr. Meena', 'ENT');

INSERT INTO Appointment VALUES (1001, 1, 101, '2024-03-01');
INSERT INTO Appointment VALUES (1002, 2, 102, '2024-03-02');

3. Display All Patients

SELECT * FROM Patient;

Output:

```
mysql> SELECT * FROM Patient;
+------------+-------+------+--------+---------+
| Patient_ID | Name  | Age  | Gender | Disease |
+------------+-------+------+--------+---------+
|          1 | Ravi  |   30 | Male   | Fever   |
|          2 | Anita |   25 | Female | Cold    |
+------------+-------+------+--------+---------+
2 rows in set (0.01 sec)
```

4. Find Patients with Specific Disease

SELECT Name FROM Patient WHERE Disease = 'Fever';

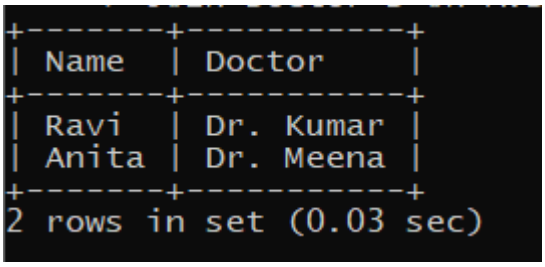
Output:

```
mysql> SELECT Name FROM Patient WHERE Disease = 'Fever';
+------+
| Name |
+------+
| Ravi |
+------+
1 row in set (0.02 sec)
```

5. Join Query (Patient + Doctor)

```
SELECT P.Name, D.Name AS Doctor
    FROM Patient P
    JOIN Appointment A ON P.Patient_ID = A.Patient_ID
    JOIN Doctor D ON A.Doctor_ID = D.Doctor_ID;
```

Output:

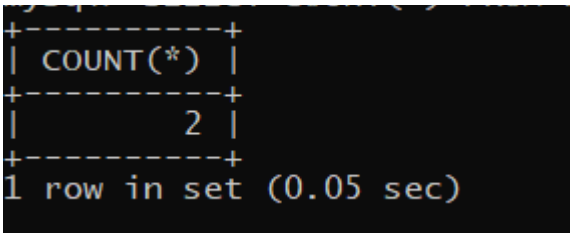


```
+-------+-----------+
| Name  | Doctor    |
+-------+-----------+
| Ravi  | Dr. Kumar |
| Anita | Dr. Meena |
+-------+-----------+
2 rows in set (0.03 sec)
```

6. Count Number of Patients

```
SELECT COUNT(*) FROM Patient;
```

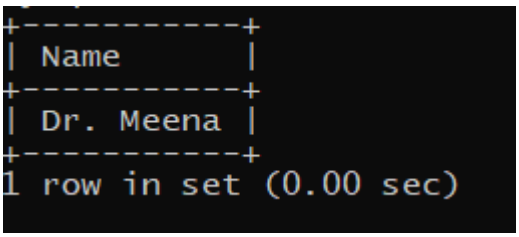
Output:



```
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
1 row in set (0.05 sec)
```

7. Find Doctors by Specialization

```
SELECT Name FROM Doctor WHERE Specialization = 'ENT';
```



```
+-----------+
| Name      |
+-----------+
| Dr. Meena |
+-----------+
1 row in set (0.00 sec)
```

8. Update Patient Disease

UPDATE Patient SET Disease = 'Flu' WHERE Patient_ID = 1;

Output:

Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

9. Delete Record

DELETE FROM Appointment WHERE Patient_ID = 2;
DELETE FROM Patient WHERE Patient_ID = 2;

Output:

```
mysql> SELECT * FROM Patient;
+------------+------+------+--------+---------+
| Patient_ID | Name | Age  | Gender | Disease |
+------------+------+------+--------+---------+
|          1 | Ravi |   30 | Male   | Flu     |
+------------+------+------+--------+---------+
1 row in set (0.00 sec)
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is a table?	CO3	Remember
2	What is a record (tuple)?	CO3	Remember
3	What is the purpose of the SELECT statement?	CO3	Understanding
4	What is the difference between WHERE and HAVING?	CO3	Understanding
5	What is a JOIN in SQL?	CO3	Understanding
6	What is INNER JOIN?	CO3	Understanding
7	What is LEFT JOIN?	CO3	Understanding
8	What is GROUP BY clause?	CO3	Understanding
9	What is COUNT() function used for?	CO3	Understanding
10	How do you find all patients treated by a specific doctor?	CO3	Applying
11	How do SQL queries help in hospital management?	CO3	Understanding
12	What happens if foreign keys are not used?	CO3	Analyzing
13	How do you find patients with a specific disease?	CO3	Applying
14	What are the advantages of using SQL in hospital systems?	CO3	Understanding
15	How can you improve this hospital database system?	CO3	creating
16	How do you ensure data integrity in your database?	CO3	Analyzing
17	What is an attribute?	CO3	Remember
18	How do you display doctor-wise patient details?	CO3	Applying
19	What tables are used in your hospital database?	CO3	Remember
20	How do you handle large hospital data efficiently?	CO3	Analyzing

Employee Data Management using SQL Queries

Aim: To develop and execute SQL queries to manage and analyze employee data including attendance, payroll, and performance.

Tables Used

1. Employee

- Emp_ID (PK)
- Name
- Department
- Salary

2. Attendance

- Attendance_ID (PK)
- Emp_ID (FK)
- Date
- Status (Present/Absent)

3. Payroll

- Payroll_ID (PK)
- Emp_ID (FK)
- Basic_Salary
- Bonus
- Deductions

4. Performance

- Performance_ID (PK)
- Emp_ID (FK)
- Rating
- Remarks

SQL Queries

1. Create Tables

```
CREATE TABLE Employee (
    Emp_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Department VARCHAR(50),
    Salary DECIMAL(10,2)
);
```

```
CREATE TABLE Attendance (
    Attendance_ID INT PRIMARY KEY,
    Emp_ID INT,
    Date DATE,
    Status VARCHAR(10),
    FOREIGN KEY (Emp_ID) REFERENCES Employee(Emp_ID)
);

CREATE TABLE Payroll (
    Payroll_ID INT PRIMARY KEY,
    Emp_ID INT,
    Basic_Salary DECIMAL(10,2),
    Bonus DECIMAL(10,2),
    Deductions DECIMAL(10,2),
    FOREIGN KEY (Emp_ID) REFERENCES Employee(Emp_ID)
);

CREATE TABLE Performance (
    Performance_ID INT PRIMARY KEY,
    Emp_ID INT,
    Rating INT,
    Remarks VARCHAR(100),
    FOREIGN KEY (Emp_ID) REFERENCES Employee(Emp_ID)
);
```

2. Insert Sample Data

```
INSERT INTO Employee VALUES (1, 'Rahul', 'HR', 30000);
INSERT INTO Employee VALUES (2, 'Sneha', 'IT', 40000);

INSERT INTO Attendance VALUES (101, 1, '2024-03-01', 'Present');
INSERT INTO Attendance VALUES (102, 2, '2024-03-01', 'Absent');

INSERT INTO Payroll VALUES (201, 1, 30000, 2000, 500);
INSERT INTO Payroll VALUES (202, 2, 40000, 3000, 700);

INSERT INTO Performance VALUES (301, 1, 4, 'Good');
INSERT INTO Performance VALUES (302, 2, 5, 'Excellent');
```

3. Display All Employees

SELECT * FROM Employee;

Output:

```
mysql> SELECT * FROM Employee;
+--------+-------+------------+----------+
| Emp_ID | Name  | Department | Salary   |
+--------+-------+------------+----------+
|      1 | Rahul | HR         | 30000.00 |
|      2 | Sneha | IT         | 40000.00 |
+--------+-------+------------+----------+
2 rows in set (0.00 sec)
```

4. Employees Present Today

```
SELECT E.Name
FROM Employee E
JOIN Attendance A ON E.Emp_ID = A.Emp_ID
WHERE A.Status = 'Present';
```

Output:

```
+-------+
| Name  |
+-------+
| Rahul |
+-------+
1 row in set (0.00 sec)
```

5. Calculate Net Salary

```
SELECT Emp_ID, (Basic_Salary + Bonus – Deductions) AS Net_Salary
FROM Payroll;
```

Output:

```
+--------+------------+
| Emp_ID | Net_Salary |
+--------+------------+
|      1 |   31500.00 |
|      2 |   42300.00 |
+--------+------------+
2 rows in set (0.01 sec)
```

6. Employee Performance Details

```
SELECT E.Name, P.Rating, P.Remarks
FROM Employee E
JOIN Performance P ON E.Emp_ID = P.Emp_ID;
```

Output:

```
+-------+--------+-----------+
| Name  | Rating | Remarks   |
+-------+--------+-----------+
| Rahul |      4 | Good      |
| Sneha |      5 | Excellent |
+-------+--------+-----------+
2 rows in set (0.00 sec)
```

7. Count Employees in Each Department

```
SELECT Department, COUNT(*)
FROM Employee
GROUP BY Department;
```

Output:

```
+------------+----------+
| Department | COUNT(*) |
+------------+----------+
| HR         |        1 |
| IT         |        1 |
+------------+----------+
2 rows in set (0.02 sec)
```

8. Update Employee Salary

```
UPDATE Employee SET Salary = 35000 WHERE Emp_ID = 1;
```

Output:

```
mysql> select * FROM Employee;
+--------+-------+------------+----------+
| Emp_ID | Name  | Department | Salary   |
+--------+-------+------------+----------+
|      1 | Rahul | HR         | 35000.00 |
|      2 | Sneha | IT         | 40000.00 |
+--------+-------+------------+----------+
2 rows in set (0.00 sec)
```

VIVA QUESTIONS

S.No	Question	CO	Blooms Taxonomy
1	What is DBMS?	CO4	Remember
2	What is the purpose of the SELECT statement?	CO4	Remember
3	What are the tables used in your employee database?	CO4	Understand
4	Why is Emp_ID used as a primary key?	CO4	Understand
5	What is the difference between INNER JOIN and OUTER JOIN?	CO4	Understand
6	How do you calculate net salary in SQL?	CO4	Apply
7	Difference between 3NF and BCNF?	CO4	Understand
8	What is denormalization?	CO4	Understand
9	How does payroll help in employee management?	CO4	Analyze
10	What is the role of performance table?	CO4	Analyze
11	How can you improve this employee database system?	CO4	Evaluate
12	How do you find employees who are present on a specific day?	CO4	Apply
13	What happens if foreign keys are not used?	CO4	Understand
14	How do SQL queries help in data analysis?	CO4	Understand
15	What is functional dependency?	CO4	Remember
16	What is transitive dependency?	CO4	Remember
17	Difference between DDL, DML, and DCL?	CO4	Understand
18	Difference between DELETE, TRUNCATE, and DROP?	CO4	Understand
19	What is a subquery?	CO4	Understand
20	What is a JOIN? Types of joins	CO4	Understand