SHREE L. R. TIWARI COLLEGE OF ENGINEERING

Approved by AICTE & DTE, Maharashtra State & Affiliated to University of Mumbai, NAAC Accredited, NBA Accredited program, ISO 9001:2015 Certified | DTE Code No: 3423, Recognized under Section 2(f) of the UGC Act 1956, Minority Status (Hindi Linguistic)

DEPARTMENT OF COMPUTER ENGINEERING

A Laboratory Manual for

Database Management System Lab (CSL402)

ACADEMIC YEAR: 2024-25

Course Name: Database Management Lab **Course Code: CSL402**

Name: _____

Semester: IV (Fourth) **Roll No.:** _____

Div.: _____ **Exam. Seat No.:** _____

Email ID: _____ **Mobile No.:** _____

VISION AND MISSION

Institution's

Vision	To be a world class institute and a front runner in educational and socioeconomic development of the nation by providing high quality technical education to students from all sections of society.
Mission	To provide superior learning experiences in a caring and conducive environment so as to empower students to be successful in life & contribute positively to society.
Quality Policy	We, at SHREE L. R. TIWARI COLLEGE OF ENGINEERING, shall dedicate and strive hard to continuously achieve academic excellence in the field of Engineering and to produce the most competent Engineers through objective & innovative teaching methods, consistent updating of facilities, welfare & quality improvement of the faculty & a system of continual process improvement.

Computer Engineering Department's

Vision	To be a department of high repute focused on quality education, training and skill development in the field of computer engineering to prepare professionals and entrepreneurs of high caliber with human values to serve our nation and globe.
Mission	**M1:** To provide fertile academic environment for the development of skilled professionals and empowered with knowledge, skills, values, and confidence to take the leadership role and to bridge the gap between industry institute and society in the field of Computer engineering.
	M2: To promote caring and interactive teaching practices in a rejoicing learning ambience with richly supported modern educational tools and techniques.
	M3: To enhance and revitalize research culture to provide practical exposure and to establish synergy between teaching and research and make it an enabler for speedy progress.
	M4: To pursue intensification of soft skills and personality development through interplay of achievers of all segments of our society.
	M5: To provide human values to students by promoting lifelong learning ability.
Program Educational Objectives	**PEO-1:** To prepare students for successful carrier in industry, research and institutions of higher learning.
	PEO-2: To encourage student to work in teams to address industrial and socially relevant problems/projects.
	PEO-3: To provide student with a sound mathematical, scientific and engineering fundamental necessary to formulate, analyze and solve engineering problems.
	PEO-4: To promote student awareness and commitment to lifelong learning and professional ethics during the course of professional practice.

SHREE L. R. TIWARI COLLEGE OF ENGINEERING

Approved by AICTE & DTE, Maharashtra State & Affiliated to University of Mumbai, NAAC Accredited, NBA Accredited program, ISO 9001:2015 Certified | DTE Code No: 3423, Recognized under Section 2(f) of the UGC Act 1956, Minority Status (Hindi Linguistic)

DEPARTMENT OF COMPUTER ENGINEERING

Certificate

*This is to certify that Mr. /Ms.*_____

Class _____ *Roll No.* _____ *Exam Seat No.* _____

of ***Fourth Semester of Degree in Computer Engineering*** *has completed the required number of Practical's / Term Work / Sessional in the subject* ***Database Management System Lab*** *from the* ***Department of Computer Engineering*** *during the academic year of* ***2024-2025*** *as prescribed in the curriculum.*

Lecturer in-Charge **Head of the Department** **Principal**

Date:

Seal of Institution

INSTRUCTION FOR STUDENTS

1) Listen carefully to the lecture given by teacher about importance of subject, instruments, procedure, method of continuous assessment, tentative plan of work in laboratory and total amount of work to be done in a semester.
2) Student shall undergo study visit of the laboratory for types of equipment, instruments, software to be used, before performing experiments.
3) Read the write up of each experiment to be performed, a day in advance.
4) Organize the work in the group and make a record of all observations.
5) Understand the purpose of experiment and its practical implications.
6) Write the answers of the questions allotted by the teacher during practical hours if possible or afterwards, but immediately.
7) Student should not hesitate to ask any difficulty faced during conduct of practical/exercise.
8) The student shall study all the questions given in the laboratory manual and practice to write the answers to these questions.
9) Student shall develop maintenance skills as expected by the industries.
10) Student should develop the habit of pocket discussion/group discussion related to the experiments/exercises so that exchanges of knowledge/skills could take place.
11) Student shall attempt to develop related hands-on-skills and gain confidence.
12) Student shall focus on development of skills rather than theoretical or codified knowledge.
13) Student shall visit the nearby workshops, workstation, industries, laboratories, technical exhibitions, trade fair etc. even not included in the Lab manual. In short, students should have exposure to the area of work right in the student hood. 14) Student shall insist for the completion of recommended laboratory work, industrial visits, answers to the given questions, etc.
15) Student shall develop the habit of evolving more ideas, innovations, skills etc. those included in the scope of the manual.
16) Student shall refer technical magazines, proceedings of the seminars, refer websites related to the scope of the subjects and update his knowledge and skills.
17) Student should develop the habit of not to depend totally on teachers but to develop self-learning techniques.
18) Student should develop the habit to react with the teacher without hesitation with respect to the academics involved.
19) Student should develop habit to submit the practical's, exercise continuously and progressively on the scheduled dates and should get the assessment done.
20) Student should be well prepared while submitting the write up of the exercise. This will develop the continuity of the studies and he/she will not be over loaded at the end of the term.

SHREE L. R. TIWARI COLLEGE OF ENGINEERING

Approved by AICTE & DTE, Maharashtra State & Affiliated to University of Mumbai, NAAC Accredited, NBA Accredited program, ISO 9001:2015 Certified | DTE Code No: 3423, Recognized under Section 2(f) of the UGC Act 1956, Minority Status (Hindi Linguistic)

DEPARTMENT OF COMPUTER ENGINEERING

RECORD OF PROGRESSIVE ASSESSMENTS

Student Name: _____ Roll No.: _____ (SE CS SEM-IV)

Course Name: Database Management System Lab Course Code: CSL402

Assessment of Experiments (A)

Sr. no.	Name of Experiments	Page No.	Date of Performance	Date of Submission	Assessment (out of 15)	Teacher's Signature and Remark	CO Covered
1	Design an Entity-Relationship Diagram						CO1
2	Mapping ER/EER to Relational schema model.						CO1
3	Create a database using Data Definition Language (DDL)						CO2
4	Apply DML Commands for the specified system						CO2
5	Perform DCL,TCL and DQL Commands						CO2
6	Perform Simple queries, string Manipulation operations and aggregate functions.						CO3
7	Implement various Join Operations						CO3
8	Perform Nested and Complex queries						CO3
9	Implement Procedure and Function						CO4
10	Implementation of Views and Triggers.						CO5
11	Case Study on NOSQL						CO4
Average Marks (Out of 15)							
Converted Marks (Out of 15) (A)							

Assessment of Assignments (B)

Sr. no.	Assignment	Page No.	Date of Display	Date of Completion	Assessment (Out of 12)	Teacher's Signature and Remark	CO Covered
1	Assignment No.1						CO1,CO2,CO3
2	Assignment No.2						CO4, CO5, C06
Average Marks (Out of 12)							
Converted Marks (Out of 5) (C)							

Assessments of Attendance (C)

DBMS Theory Attendance			DBMS Practical Attendance			AVG. Attendance % (TH+PR)	Attendance Marks (C) (Out of 5)
TH (out of)	TH attend.	TH %	PR (out of)	PR Attend.	PR %		

Total Term Work Marks: A+B+C= _____ (Out of 25)

Student Signature	Subject In-charge	Head of the Department

Programme Outcome (POs & PSOs)

Programme Outcomes are the skills and knowledge which the students have at the time of graduation. This will indicate what student can do from subject-wise knowledge acquired during the programme.

PO	Graduate Attributes	Description of the Programme outcome as defined by the NBA
PO-1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO-2	Problem analysis	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO-3	Design/development of solutions	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate 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 and research methods 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 modelling 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 diverse teams, and in multidisciplinary settings.
PO-10	Communication n	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO-11	Project management and finance	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and 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 (PSOs) defined by the programme. Baseline-Rational Unified Process (RUP)		
PSO-1	System Inception and Elaboration	"The graduate must be able to develop, deploy, test and maintain the software or computing hardware solutions to solve real life problems using state of the art technologies, standards, tools and programming paradigms."
PSO-2	System Construction	The graduate should be able to adapt Computer Engineering knowledge and skills to create career paths in industries or business organizations or institutes of repute.

Student's Signature

Course Objectives and Outcomes

Academic Year: 2024-25 **Class:** SE **Course Code:** CSL403
Program: Computer Engineering
Course Name: Database Management System Lab
Department: Computer Engineering **Sem.:** IV **Faculty:** Manali Parate

Course Outcomes:

CO's No.	Abbrev ation	Statement
CSL402.1	CO1	Design ER /EER diagram and convert to relational model for the real-world application.
CSL402.2	CO2	Apply DDL, DML, DCL and TCL commands
CSL402.3	CO3	Write simple and complex queries
CSL402.4	CO4	Use PL / SQL Constructs.
CSL402.5	CO5	Demonstrate the concept of concurrent transactions execution and frontend-backend connectivity

Course Prerequisite:

Sr. No.	Pre-requisite
1	Discrete Structure

Teaching and Examination Scheme:

Teaching Scheme (Hrs)			Credits Assigned				Examination Scheme								
Theory	Pract	Tut	Theory	TW/ Pract	Tut	Total	Theory					TW	Oral	Oral & Pract	Total
							Internal Assessment			End Sem. Exam	Exam Duration (in Hrs)				
							Test 1	Test 2	Avg.						
4	2	-	4	1	-	5	20	20	20	80	3	25	25	---	50

Term Work (Total 25 Marks) = (Experiments: 10 marks + Assignments: 10mark + Attendance: 05 marks (TH+PR)). **Oral/Practical exam** will be based on the above-mentioned experiment list and CSL402: Database Management System syllabus.

SHREE L. R. TIWARI COLLEGE OF ENGINEERING
Approved by AICTE & DTE, Maharashtra State & Affiliated to University of Mumbai, NAAC Accredited, NBA Accredited program,
ISO 9001:2015 Certified | DTE Code No: 3423, Recognized under Section 2(f) of the UGC Act 1956, Minority Status (Hindi Linguistic)
DEPARTMENT OF COMPUTER ENGINEERING

Course Exit Form

Student Name: _____ **Roll No.:** _____

Class/Semester: S.E/IV **Academic Year:2024-25**

Course Name: Database Management System Lab **Course Code: CSL402**

Judge your ability with regard to the following points on the scale of **1 (lowest) to 5 (highest),** based on the knowledge and skills you attained from this course.

Sr. No.	Your ability to	1 Lowest	2	3	4	5 Highest
1	**CSL402.1:** Design ER /EER diagram and convert to relational model for the real world application.					
2	**CSL402.2:** Apply DDL, DML, DCL and TCL commands					
3	**CSL403.2:** Write simple and complex queries					
4	**CSL402.4:** Use PL / SQL Constructs.					
5	**CSL402.5:** Demonstrate the concept of concurrent transactions execution and frontend-backend connectivity					

_____ _____

Student's Signature **Date**

EXPERIMENT NO 1

Title: Identify the case study and detail statement of problem. Design an Entity Relationship (ER) / Extended Entity-Relationship (EER) Model.

Aim: Design an Entity-Relationship (ER) / Extended Entity-Relationship (EER) Model.

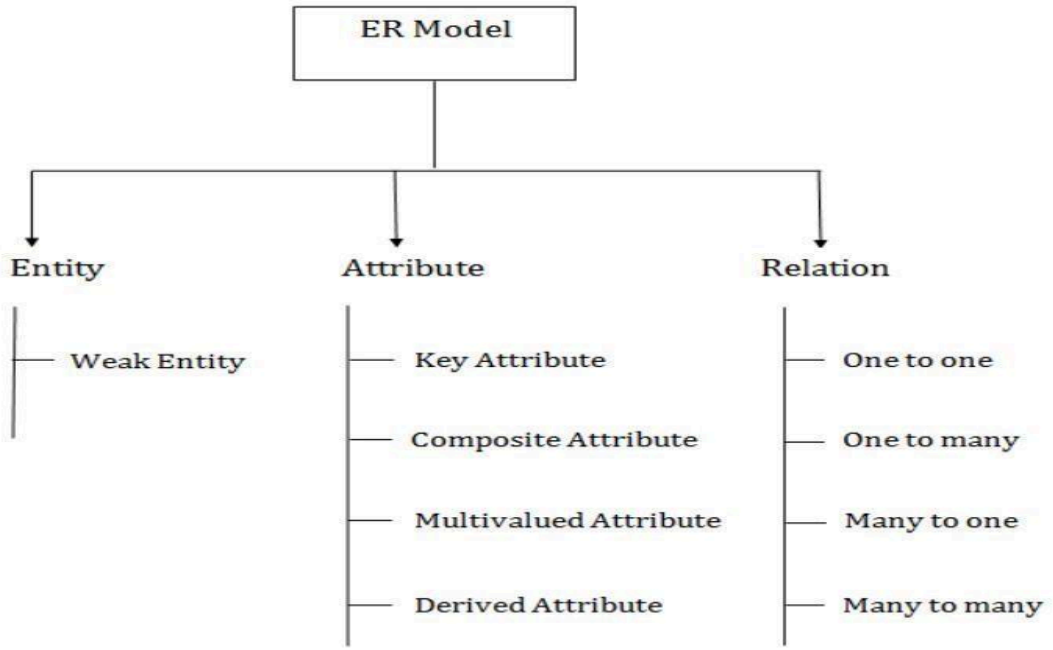
Theory:

ER (Entity Relationship) Diagram

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

COMPONENTS OF ER DIAGRAM:

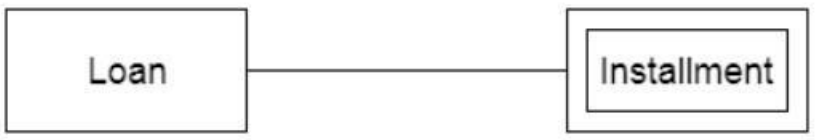


1. Entity:

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles. Consider an organization as an example-manager, product, employee, department etc. can be taken as an entity.

a. Weak Entity

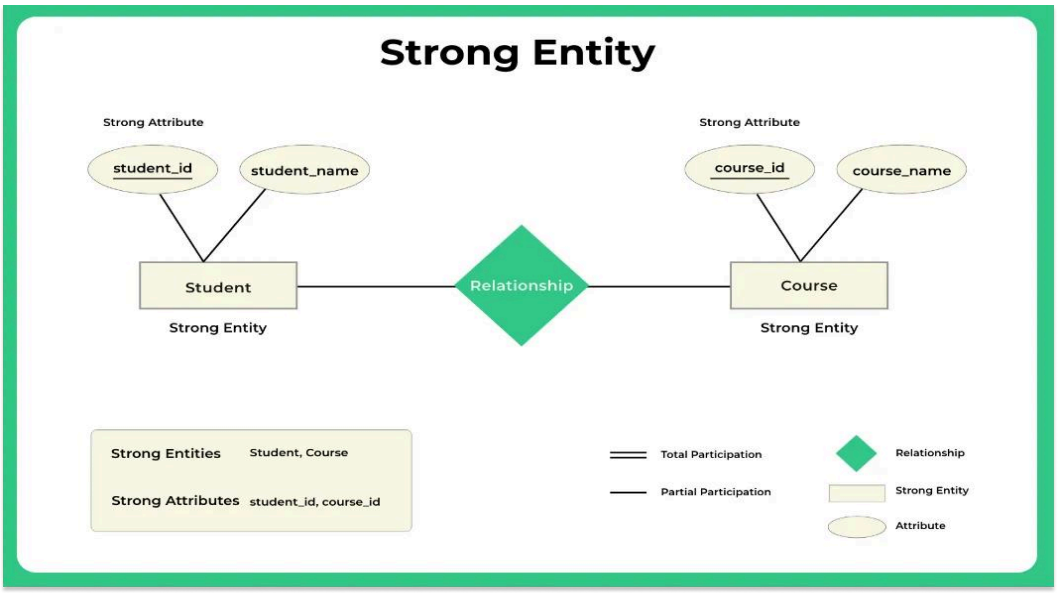
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



b. **Strong Entity:**

A strong entity is not dependent on any other entity in the schema. A strong entity will always have a primary key. Strong entities are represented by a single rectangle. The relationship of two strong entities is represented by a single diamond.

Various strong entities, when combined together, create a strong entity set.

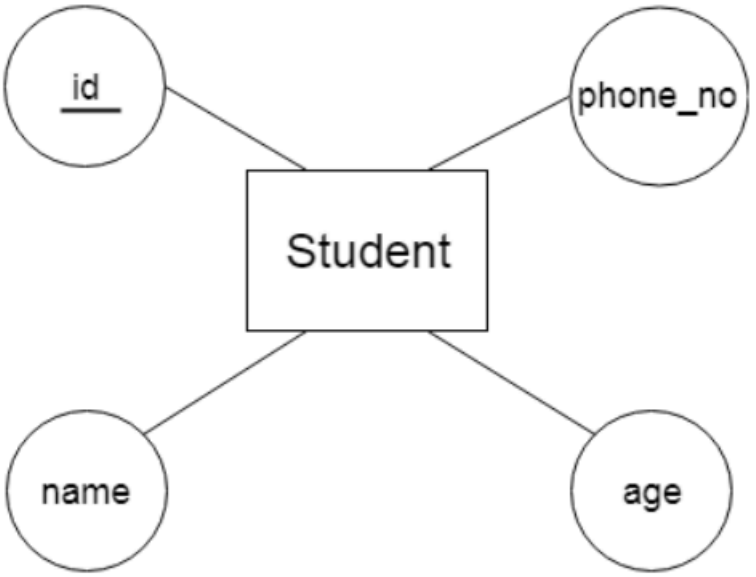


2. **Attribute**

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute. For example, id, age, contact number, name, etc. can be attributes of a student.

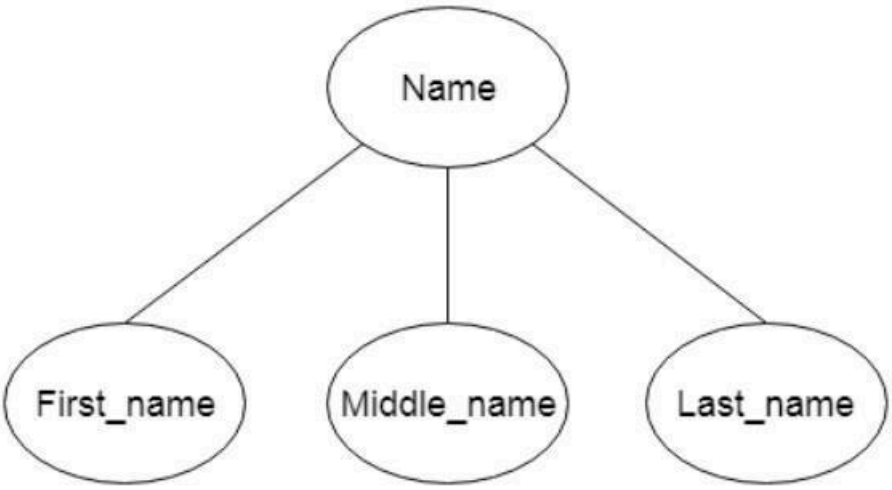
a. **Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. **Composite Attribute**

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

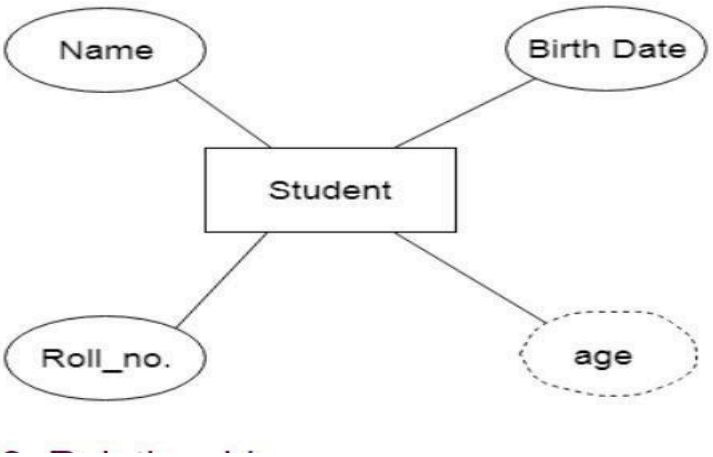


c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute. For example, a student can have more than one phone number.

d. Derived Attribute

An attribute that can be derived from another attribute is known as a derived attribute. It can be represented by a dashed ellipse. For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. **One-to-One Relationship**

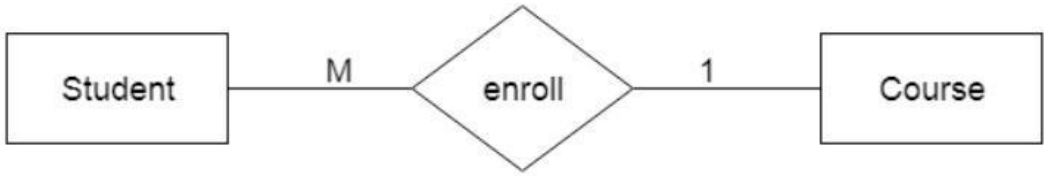
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship. For example, A female can marry to one male, and a male can marry to one female.

b. **One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a oneto-many relationship. For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

c. **Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a manyto-one relationship. For example, Student enrolls for only one course, but a course can have many students.



d. **Many-to-many relationship**

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship. For example, Employee can assign by many projects and project can have many employees.

Conclusion: Hence in this way we have studied case study and design of ER diagram

EXPERIMENT NO 2

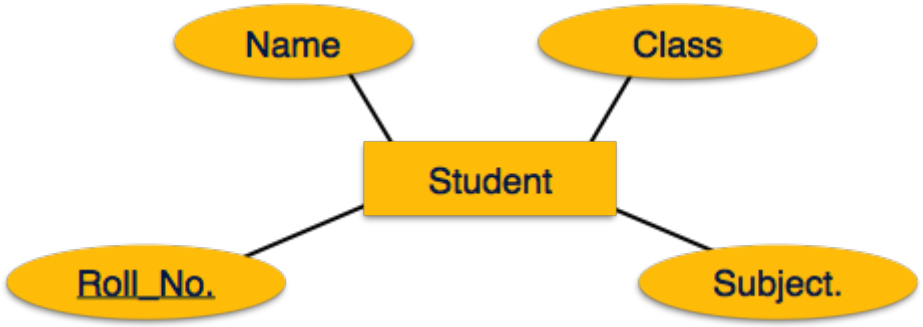
Title: Draw and explain how mapping ER/EER to Relational schema model.

Aim: Mapping of ER to Relational schema model (Tables) **Theory:**

Converting ER Diagrams to Tables-

Mapping Entity:

An entity is a real-world object with some attributes.



Mapping Process (Algorithm)

Create table for each entity.

- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key.

Mapping Process

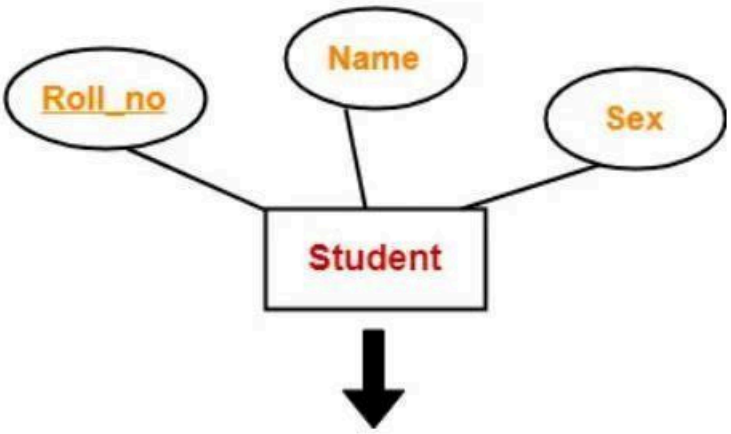
- Create table for a relationship.
- Add the primary keys of all participating Entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints.

Following rules are used for converting an ER diagram into the tables-

Rule-01: For Strong Entity Set with Only Simple Attributes-

A strong entity set with only simple attributes will require only one table in relational model. Attributes of the table will be the attributes of the entity set. The primary key of the table will be the key attribute of the entity set.

Example-



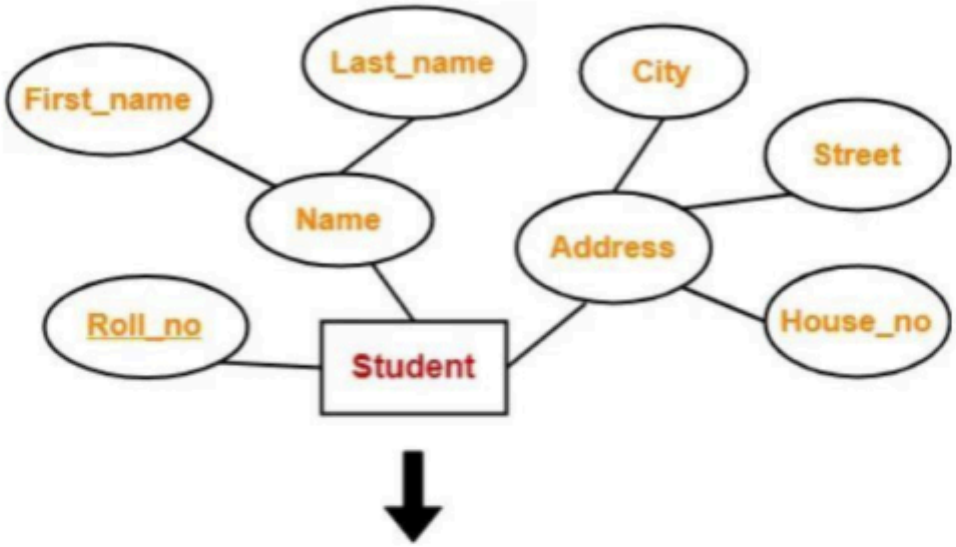
Roll_no	Name	Sex

Schema : Student (Roll_no , Name , Sex)

Rule-02: For Strong Entity Set with Composite Attributes-

A strong entity set with any number of composite attributes will require only one table in relational model. While conversion, simple attributes of the composite attributes are considered and not the composite attribute itself.

Example-



Roll_no	First_name	Last_name	House_no	Street	City

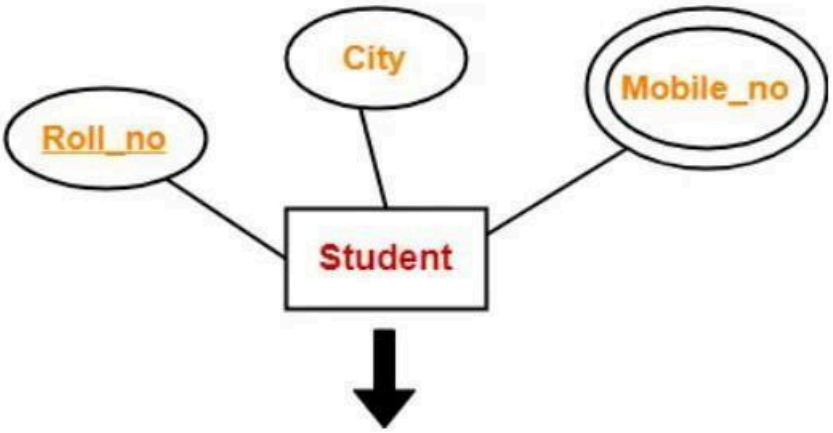
Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

Rule-03: For Strong Entity Set with Multi Valued Attributes-

A strong entity set with any number of multi valued attributes will require two tables in relational model.

- One table will contain all the simple attributes with the primary key.
- Other table will contain the primary key and all the multi valued attributes.

Example-



Roll_no	City

Roll_no	Mobile_no

Rule-04: Translating Relationship Set into a Table-

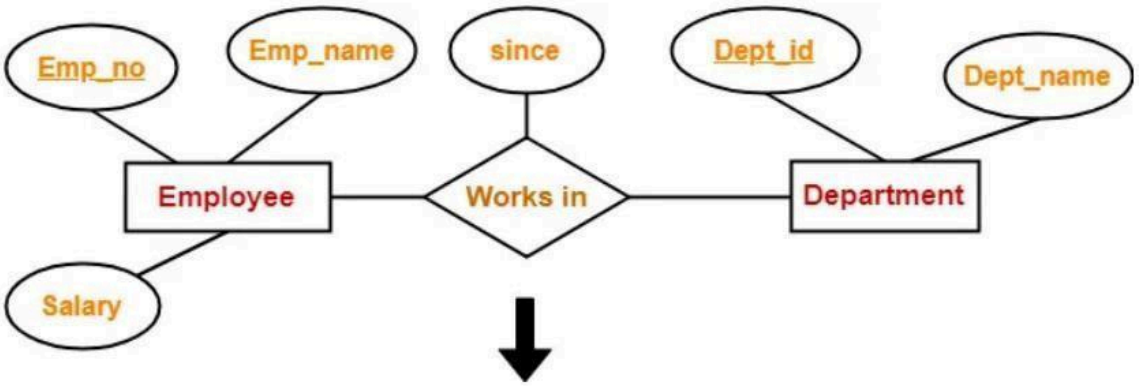
A relationship set will require one table in the relational model.

Attributes of the table are-

Primary key attributes of the participating entity set.

Its own descriptive attributes if any. Set of non-descriptive attributes will be the primary key.

Example-



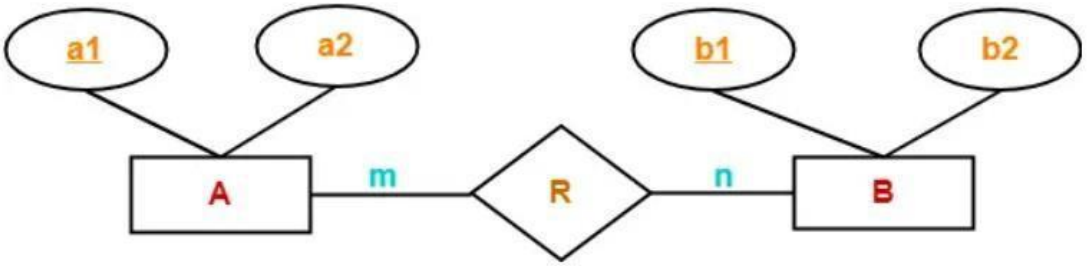
Emp_no	Dept_id	since

Schema : Works in (Emp_no , Dept_id , since)

Rule-05: For Binary Relationships with Cardinality Ratios-

The following four cases are possible-

Case-01: Binary relationship with cardinality ratio m:n



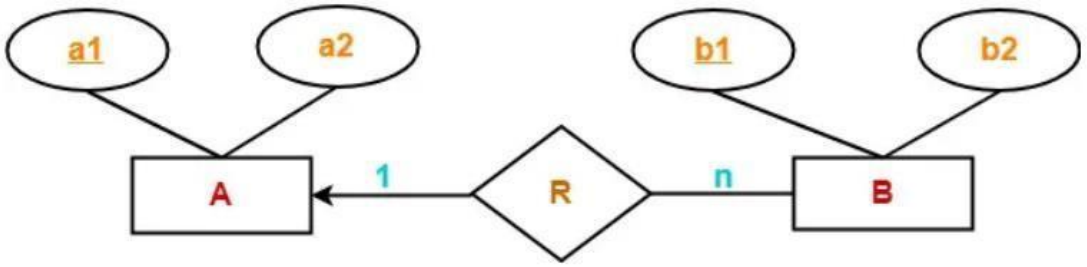
Here, three tables will be required-

A (a1 , a2)

R (a1 , b1)

B (b1 , b2)

Case-02: Binary relationship with cardinality ratio 1:n

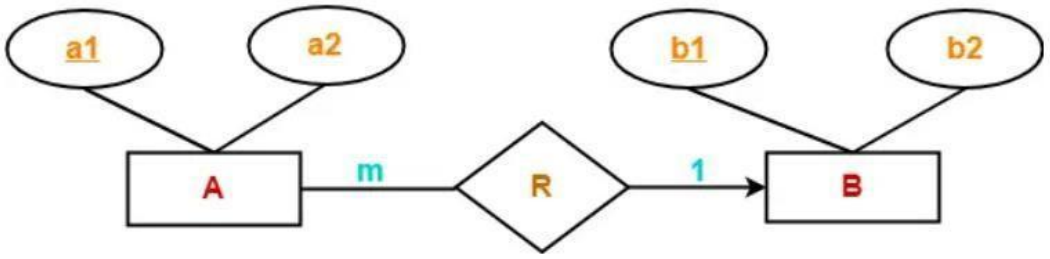


Here, two tables will be required-

A (a1 , a2)

BR (a1 , b1 , b2)

Case-03: Binary relationship with cardinality ratio m:1

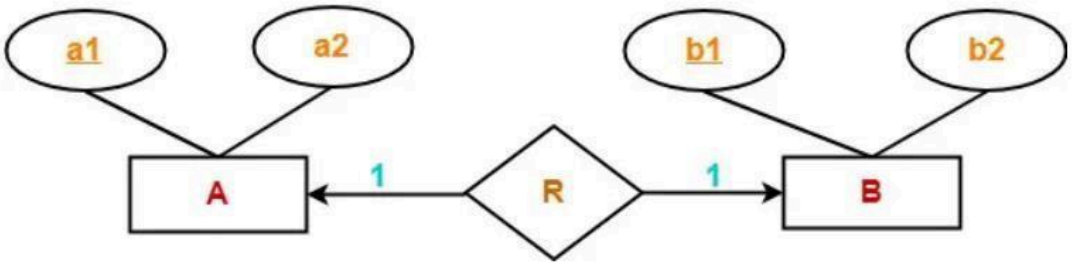


Here, two tables will be required-

AR (a1 , a2 , b1)

B (b1 , b2)

Case-04: Binary relationship with cardinality ratio 1:1



Way-01:

AR (a1 , a2 , b1)

B (b1 , b2)

Way-02:

A (a1 , a2)

BR (a1 , b1 , b2)

Rule-06: For Binary Relationship With Both Cardinality Constraints and Participation Constraints-

- Cardinality constraints will be implemented as discussed in Rule-05.
- Because of the total participation constraint, foreign key acquires NOT NULL constraint i.e. now foreign key can not be null.

Case-01: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From One Side-

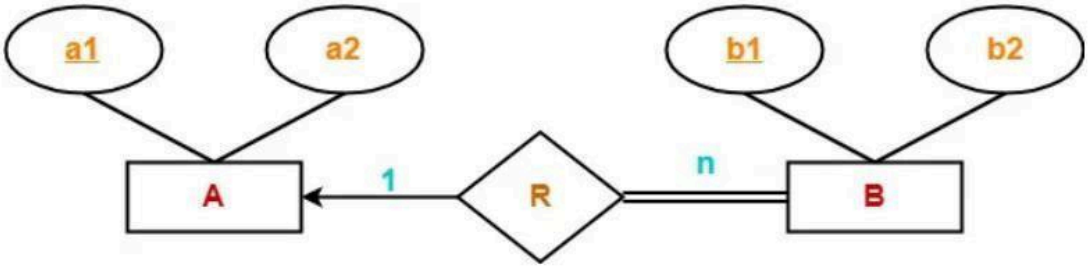
Because cardinality ratio = 1 : n , so we will combine the entity set B and relationship set R.

Then, two tables will be required-

A (a1 , a2)

BR (a1 , b1 , b2)

Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can't be null now.

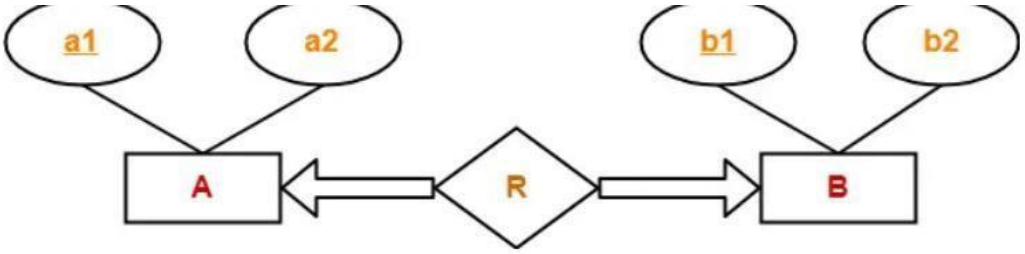


Case-02: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from Both Sides-

If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table.

Here, only one table is required.

ARB (a1, a2, b1, b2)



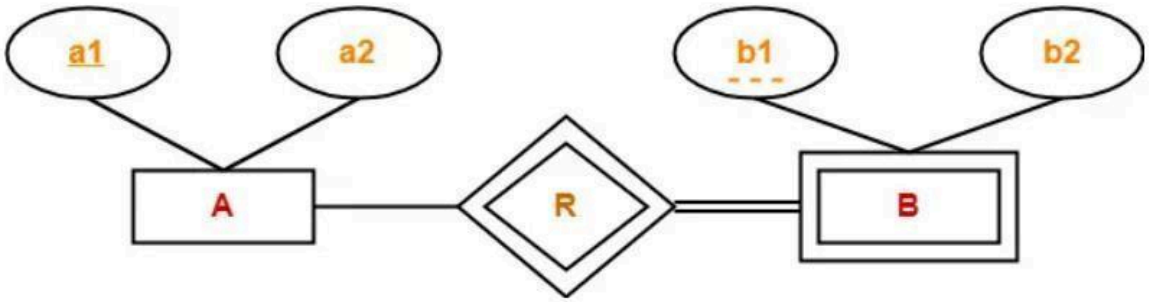
Rule-07: For Binary Relationship with Weak Entity Set-

Weak entity set always appears in association with identifying relationship with total participation constraint.

Here, two tables will be required-

A (a1, a2)

BR (a1, b1, b2)



Conclusion: Hence in this way we have studied mapping of ER to Relational Schema.

EXPERIMENT NO 3

Title: Create a database using Data Definition Language (DDL) and apply integrity constraints for the specified System

Aim: Perform Data Definition Language (DDL)for specified system.

Theory:

Structured Query Language (SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also, we can use this language to create a database. SQL uses certain commands like CREATE, DROP, INSERT, etc. to carry out the required tasks.

SQL commands are like instructions to a table. It is used to interact with the database with some operations. It is also used to perform specific tasks, functions, and queries of data. SQL can perform various tasks like creating a table, adding data to tables, dropping the table, modifying the table, set permission for users.

These SQL commands are mainly categorized into five categories:

1. DDL – Data Definition Language
2. DQL – Data Query Language
3. DML – Data Manipulation Language
4. DCL – Data Control Language
5. TCL – Transaction Control Language

DDL (Data Definition Language)

Following are the five DDL commands in SQL:

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

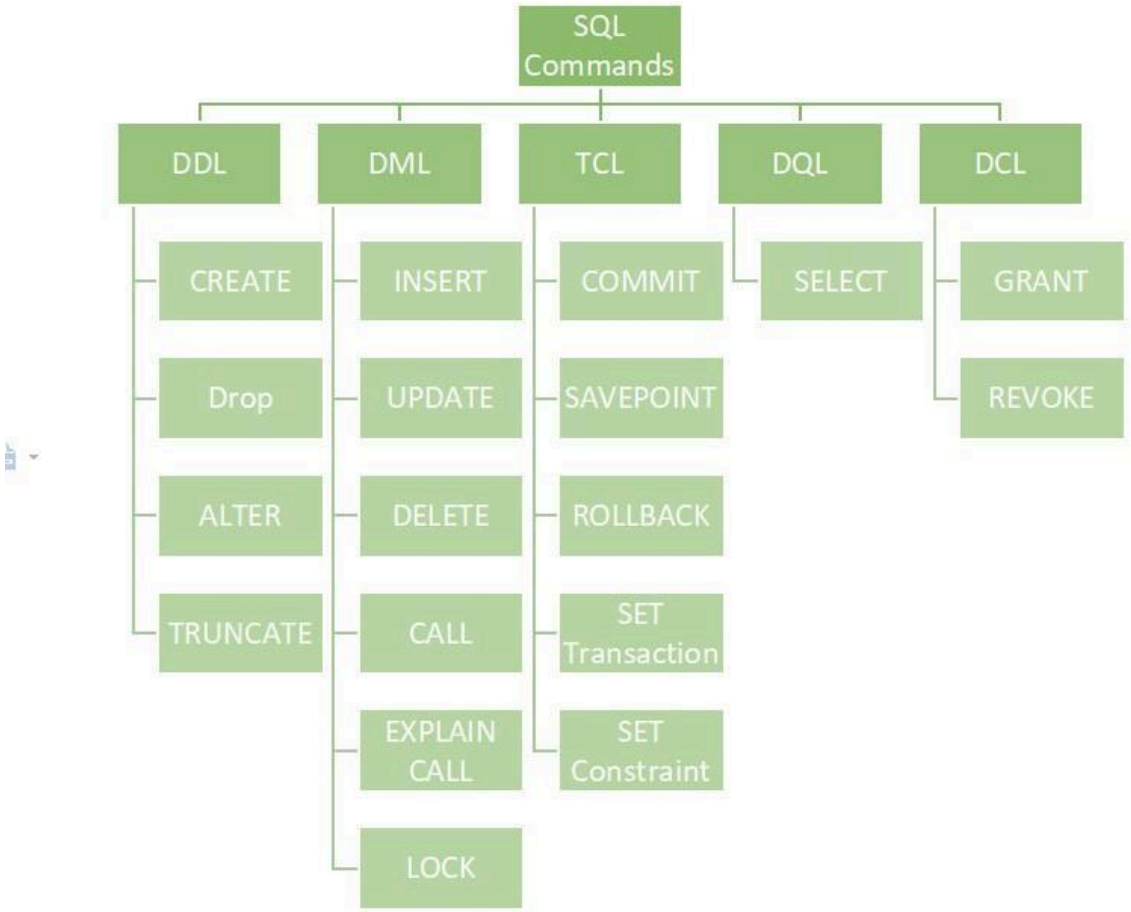
DROP: This command is used to delete objects from the database.

ALTER: This is used to alter the structure of the database.

TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.

COMMENT: This is used to add comments to the data dictionary.

RENAME: This is used to rename an object existing in the database.



CREATE TABLE

A **Table** is a combination of rows and columns. For creating a table, we have to define the structure of a table by adding names to columns and providing data type and size of data to be stored in columns.

Syntax:

```
CREATE table table_name
(
Column1 datatype (size), column2
datatype (size),
.
columnN datatype(size)
);
```

Example:

```
CREATE TABLE Customer (
   CustomerID INT PRIMARY KEY,
   CustomerName VARCHAR (50),
   LastName VARCHAR (50),
   Country VARCHAR (50),
   Age int (2),
  Phone int (10)
);
```

Customer

CustomerID	CustomerName	LastName	Country	Age	Phone
empty					

ALTER TABLE

A DBA can make changes to the table structure or column definitions after the table has been created in the database. The DDL command ALTER TABLE is used to perform such actions. Alter command provides multiple utilities exclusive for schema objects. The ALTER TABLE statement is used to add, drop, rename, and modify a column in a table.

ALTER TABLE EMP RENAME TO EMP_NEW;

ALTER TABLE EMP_NEW ADD (TESTCOL VARCHAR2 (100))

Example:

ALTER TABLE Student ADD

(AGE number(3),COURSE varchar(40));

Output:

ROLL_NO	NAME	AGE	COURSE
1	Ram		
2	Abhi		
3	Rahul		
4	Tanu		

DROP TABLE

The DROP TABLE statement is used to remove a table from the database. The dropped table and its data remain no longer available for selection.

`Syntax:`

DROP TABLE [TABLE NAME] [PURGE]

DROP TABLE emp_new;

Example : ALTER TABLE Student

DROP COLUMN COURSE;

Output:

ROLL_NO	NAME	AGE

1	Ram	
2	Abhi	

ROLL_NO	NAME	AGE
3	Rahul	
4	Tanu	

TRUNCATE

The major difference between TRUNCATE and DROP is that truncate is used to delete the data inside the table not the whole table.

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extent of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing several integrity-enforcing mechanisms.

TRUNCATE TABLE table_name; CREATE

TABLE CUSTOMERS (

```
   ID INT NOT NULL,
   NAME VARCHAR (20) NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR (25),
   SALARY DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)

VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'kaushik', 23, 'Kota', 2000.00);

The table will be created as follows −

```
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
```

Following query removes all the records of the customers table −

TRUNCATE TABLE CUSTOMERS;

SELECT * FROM CUSTOMERS;

Output: Empty set (0.00 sec)

Conclusion: Hence in this way we have created a database using the DDL.

EXPERIMENT NO 4

Title: Study and implement different DML commands in database system.

Aim: Apply DML Commands for the specified system

Theory:

DML commands in SQL are used to change the data present in the database tables, views, etc. These commands deal with inserting data into the tables, updating the data according to the conditions, and removing the data from the tables.

DML commands manipulate the data being stored in the tables (the tables that are defined using Data Definition Language Commands (DDL) commands). CREATE, ALTER, DROP, and TRUNCATE are DDL commands that are used to create the table structures.

When a table is created, we need to add data to it, this data is added using the INSERT DML command. Another scenario where DML commands are used is when we need to update or delete existing data based on some conditions. In this case, DML commands like UPDATE, and DELETE can be used.

SELECT command in SQL is considered in both DML and DQL (Data Query Language). Strictly speaking, it's part of DQL, but it is mostly considered under DML. SELECT is used to retrieve/fetch data from tables. We can give conditions to filter out the data being displayed.

DML commands are not auto-committed, which means the changes done using these commands are not saved into the database automatically. Users need to commit them manually. Changes done using DML commands in SQL can be rolled back.

The DML commands in SQL are as follows:

1. INSERT
2. UPDATE

3. DELETE
4. SELECT

INSERT

INSERT is a DML command in SQL that is used to insert new records into database tables. We can insert data in all the columns of a table or some specific columns using the INSERT command. While inserting records, the user should check if there are any integrity constraints like PRIMARY KEY, UNIQUE, NOT NULL, etc. on the table and insert records accordingly.

Syntax:

```
INSERT INTO table_name (column_name1, column_name2, column_name3, ....)

VALUES (value_1, value_2, value_3, ....);
```

Example:

```
INSERT INTO Students (Roll_no,Name,Age,Address,Date_of_Birth) VALUES (1, 'John', 16, 'Pune', '2006/06/06');

INSERT INTO Students (Roll_no,Name,Age,Address,Date_of_Birth) VALUES (2, 'Ajay', 15, 'Chennai', '2007/10/02');

INSERT INTO Students (Roll_no,Name,Age,Address,Date_of_Birth) VALUES (3, 'Ron', 12, 'Delhi', '2010/08/31');

INSERT INTO Students (Roll_no,Name,Age,Address,Date_of_Birth) VALUES (4, 'Joy', 16, 'Mumbai', '2006/01/01');

INSERT INTO Students (Roll_no,Name,Age,Address,Date_of_Birth) VALUES (5, 'Harry', 15, 'Hyderabad', '2007/12/31');
```

Output:

	Roll_no	Name	Age	Address	Date_of_Birth
1	1	John	16	Pune	2006-06-06
2	2	Ajay	15	Chennai	2007-10-02
3	3	Ron	12	Delhi	2010-08-31
4	4	Joy	16	Mumbai	2006-01-01
5	5	Harry	15	Hyderabad	2007-12-31

UPDATE

UPDATE is a DML command in SQL used to update existing records in the table.
We need to specify which records we want to update using the WHERE condition. WHERE is a clause in SQL used to filter data depending on the condition. The UPDATE command can update single or multiple records according to WHERE conditions.

Syntax:

UPDATE table_name SET [column_name1= value_1, column_name2= value_2,...] WHERE CONDITION;

Example:

Consider students who live in Pune and have moved to a new city and want to update their address to 'Nashik'. Let's achieve this using the UPDATE DML command.

Output:

	Roll_no	Name	Age	Address	Date_of_Birth
1	1	John	16	Pune	2006-06-06
2	2	Ajay	15	Chennai	2007-10-02
3	3	Ron	12	Delhi	2010-08-31
4	4	Joy	16	Mumbai	2006-01-01
5	5	Harry	15	Hyderabad	2007-12-31
6	6	Percy	15	Pune	2006-02-28
7	7	Jenny	15	NULL	2007-07-15

DELETE

DELETE is a DML command used to delete existing records from the table. This command is used to delete single or multiple records from the tables based on the condition given in the WHERE clause.

Syntax:

DELETE FROM table_Name WHERE condition; **Example:**

DELETE FROM Students WHERE Roll_no = 2; **Output:**

	Roll_no	Name	Age	Address	Date_of_Birth
1	1	John	16	Nashik	2006-06-06
2	3	Ron	12	Delhi	2010-08-31
3	4	Joy	16	Mumbai	2006-01-01
4	5	Harry	15	Hyderabad	2007-12-31
5	6	Percy	15	Nashik	2006-02-28
6	7	Jenny	15	NULL	2007-07-15

SELECT

SELECT is a DML command in SQL used to retrieve data from database tables. SELECT is one of the most important and most used commands in SQL. Although SELECT is most often considered as a **DML** command in SQL strictly, it is part of **DQL** as it is used to query the database. We can select the columns being displayed and apply conditions on the records being displayed using the SELECT **DML** command in SQL.

Syntax: SELECT column_name1, column_name2,....FROM table_name;

Example: SELECT * FROM Students

Output:

	Roll_no	Name	Age	Address	Date_of_Birth
1	1	John	16	Nashik	2006-06-06
2	3	Ron	12	Delhi	2010-08-31
3	4	Joy	16	Mumbai	2006-01-01
4	5	Harry	15	Hyderabad	2007-12-31
5	6	Percy	15	Nashik	2006-02-28
6	7	Jenny	15	NULL	2007-07-15

Conclusion: Hence in this way we have implemented DML commands**.**

EXPERIMENT NO 5

Title: Study and implement different data control and transaction control languages and different clauses in database system.

Aim: Perform DCL and TCL commands

Theory:

Data Control Language

DCL is used to access the stored data. It is used to revoke and grant the user the required access to a database. In the database, this language does not have the feature of rollback. It is a part of the structured query language (SQL).

It helps in controlling access to information stored in a database. It complements the data manipulation language and the data definition language. It is the simplest of three commands.

It provides the administrators, to remove and set database permissions to desired users as needed. These commands are employed to grant, remove and deny permissions to users for retrieving and manipulating a database. There are two relevant commands under this category: grant and revoke.

1. **GRANT**

GRANT is a command used to provide access or privileges on the database objects to the users.

Syntax

GRANT PRIVILEGES

ON OBJECT

TO USER;

To grant Select Privilege to a table named “tableName", the user name is "userName", and the following GRANT statement should be executed.

2. **SELECT:**

Syntax:

```
GRANT SELECT
ON tableName
TO 'userName'@'localhost';
```

Granting multiple privileges to a user:

To grant multiple Privileges to a user named “username" in table “tableName”, the following GRANT statement should be executed:

```
GRANT SELECT, INSERT, DELETE, UPDATE
ON tableName
TO 'userName'@'localhost';
```

To Grant all the privileges to a user named “userName” in a table “tableName”, the following Grant statement should be executed.

```
GRANT ALL
ON tableName TO 'userName'@'localhost';
```

REVOKE

To revoke some or all of these privileges. We have a revoke command. You can revoke any combination of SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER, or ALL.

Syntax:

```
REVOKE privileges ON object FROM user;
```

Suppose we need to revoke delete permission for the 'tableName' table' from a user named 'userNamed', the following would be the query.

REVOKE DELETE

ON tableName FROM userName;

Transaction Control Language

TCL includes statements that are used to manage the changes that are made from
DML statements. It enhances the transactional nature of SQL. The TCL commands in SQL are:

COMMIT: It's a SQL command used in the transaction tables or database to make the current transaction or database statement permanent. It shows the successful completion of a transaction. If we have successfully executed the transaction statement or a simple database query, we want to make the changes permanent. We need to perform the commit command to save the changes, and these changes become permanent for all users. Furthermore, once the commit command is executed in the database, we cannot regain its previous states in which it was earlier before the execution of the first statement.

Syntax: Commit;

ROLLBACK: Undoes any changes made to the database. ROLLBACK is the SQL command that is used for reverting changes performed by a transaction. When a ROLLBACK command is issued it reverts all the changes since the last COMMIT or ROLLBACK.

SYNTAX

SAVEPOINT: This command creates a point in your transaction to which you can roll back. It is a command in SQL that is used with the rollback command. It is a command in Transaction Control Language that is used to mark the transaction in a table.

ROLLBACK;

SAVEPOINT: This command creates a point in your transaction to which you can roll back. It is a command in SQL that is used with the rollback command. It is a command in Transaction Control Language that is used to mark the transaction in a table.

SYNTAX

SAVEPOINT some_name; table class,

id	name
1	Abhi
2	Adam
4	Alex

INSERT INTO class VALUES(5, 'Rahul');

COMMIT;

UPDATE class SET name = 'Abhijit' WHERE id = '5';

SAVEPOINT A;

INSERT INTO class VALUES (6, 'Chris');

SAVEPOINT B;

INSERT INTO class VALUES (7, 'Bravo');

SAVEPOINT C;

SELECT * FROM class;

id	name
1	Abhi
2	Adam

4	Alex

5	Abhijit
6	Chris
7	Bravo

ROLLBACK command to roll back the state of data to the save point

Syntax: ROLLBACK TO B;

SELECT * FROM class;

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit
6	Chris

again, use the ROLLBACK command to roll back the state of data to the save point A

Syntax: ROLLBACK TO A;

SELECT * FROM class;

id	name
1	Abhi
2	Adam
4	Alex
5	Abhijit

Conclusion: Hence in this way a DCL and TCL command will be implemented successfully.

EXPERIMENT NO 6

Title: Study and implement different, string manipulation operations and aggregate functions.

Aim: Perform String operations and Aggregate function on the user database.

Theory:

SQL String functions are the predefined functions that allow the database users for string manipulation. These functions only accept, process, and give results of the string data type.

There are many string functions available some are listed here:

1. LCASE ()
2. UCASE ()
3. LEN ()
4. MID ()
5. ROUND ()

1. **LCASE ()**: LCASE stands for Lowercase which is a scalar function used to convert strings of characters to lowercase.

Table Employee:

S. No.	Emp_ID	Name	Salary
1.	213	ABHAY	12000
2.	214	Aakash	15200
3.	215	bittu	13400

SELECT LCASE(Name) FROM Employee;
Output:

LCASE(Name)
abhay
aakash
bittu

2. **UCASE():** UCASE stands for Uppercase and this function is used to convert all the characters of a string to uppercase.

SELECT UCASE(Name) FROM Employee;
Output:

UCASE(Name)
ABHAY
AAKASH
BITTU

3. **LEN():** This function is used to get the length of any string value.

SELECT LENGTH(Name) FROM Employee;
Output:

LENGTH(Name)
5
6
5
4

4. **MID():** This scalar function is useful when we want to extract substrings from any column containing string values.

SELECT MID(Name, 3, 2) FROM Employee;

MID(Name,3,2)
ha
ka

tt

5. **ROUND ():** Round function is used when you need to round off any numeric value which is in decimal point values.

SELECT ROUND(Marks) FROM Student;
Output:

ROUND(Marks)
82
79
64
70

Aggregate Functions

These functions are used to perform various mathematical calculations on a single or group of values in the databases. The aggregate functions in SQL are very powerful to perform operations on the data. As we know about MS Excel functions, we can perform almost every mathematical calculation in Excel. This is just like MS Excel where we are also applying some formulas to the data stored in the databases. The aggregate functions return only a single value and these functions are also useful to summarize the data. When you start using these functions in SQL, you will get more familiar with the working of these functions. In aggregate functions, the NULL values are ignored while performing calculations except for the COUNT function.

1. **SUM():** Sum is a function that totals the numeric values of a column and gives us the output. SUM() is a mathematical function that adds all the values in a column and returns the SUM() of that column.

Employee table:

S. No.	Emp_ID	Name	Salary
1.	213	Abhay	12000
2.	214	Aakash	15200
3.	215	Bittu	13400
4.	216	Ravi	15000

SELECT SUM(Salary) FROM Employee;

Output:

SUM(Salary)
55600

2. **COUNT()**: The count function is very useful to get the total number of rows present in the table. You can also give a condition to count these rows and can also run it without a condition.

SELECT COUNT(Emp_ID) FROM Employee;
Output:

count(Emp_ID)
4

AVG(): As we already know how to calculate Average in Mathematics. So, this function does the same as we did in Maths to find the averages.

SELECT AVG(Salary) FROM Employee;

Output:

avg(Salary)
13900

3. **MIN():** The MIN() function returns the minimum value from a selected column. The minimum value can be extracted when there are integer values in that column.

SELECT MIN(Age) FROM Employee;

Output:

MIN(Age)
21

4. **MAX():** Max function becomes very important when we want to get the maximum value of any column. But this should be kept in mind, the MAX() function will work for integer values only.

SELECT MAX(Salary) FROM Employee; Output:

MAX(Salary)
15200

Conclusion: Hence in this way we have implemented different string and aggregate function.

EXPERIMENT NO 7

Title: Study and implement different Join operations in SQL.

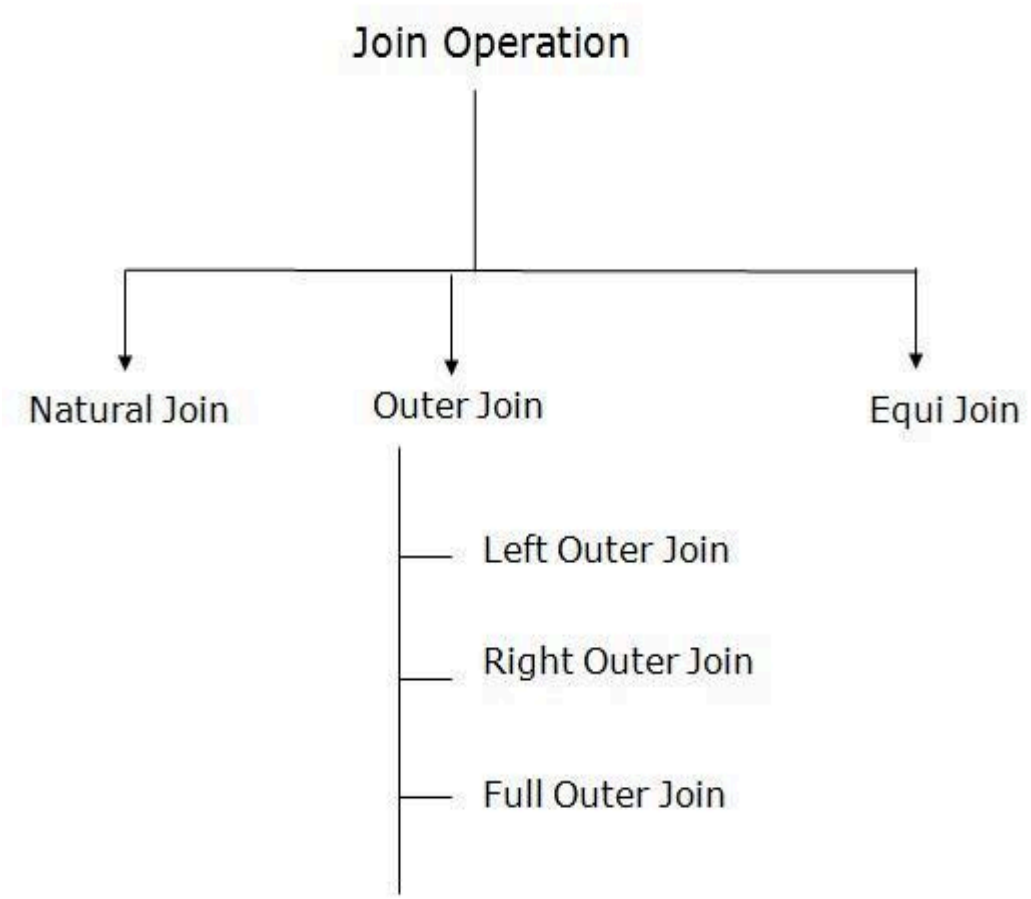
Aim: Perform various Join Operations.

Theory:

Join Operations:

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied.

Types of Join operations:



Create a table:

create a table student (name char(30), regno number(10));

Insert Values:

insert into student values ('hari', 1);

Insert into student values ('subbu', 2);

Insert into student values ('srinu', 3);

Output

Name	Regno
Hari	1
Subbu	2
Srinu	3

Create another table:

Create table marks(regno number(10), total number(10)); **Insert values:**

insert into marks values (1, 400);

Insert into marks values(2,450);

Insert into marks values (3, 300);

Output

Regno	Total
1	400

2	450
3	300

1. **Natural join** − If we join two tables on equal condition then it is called natural join or equi join. Generally, join is referred to as natural join.
Syntax:

   select columnname(s) from tablename1 join tablename2 on tablename1.columnname=tablename2.columnname; **Query**

   Select * from student join marks on student.regno = marks.regno;

   Output

Name	Regno	Regno	Total
Hari	1	1	400
Subbu	2	2	450

2. **Left join** − It is an extension of natural join to deal with missing values of relation.

   Query:

Select * from student left join marks on student.regno = marks.regno;

Output

Name	Regno	Regno	Total
Hari	1	1	400
Subbu	2	2	450

Srinu	3	NULL	NULL

3. **Right join** − Here all the tuples of table2 (right table) appear in the output. The mismatching values of table1 are filled with NULL

Query

Select * from student right join marks on student.regno = marks.regno;

Output

Name	Regno	Regno	Total
Hari	1	1	400
Subbu	2	2	450
NULL	NULL	NULL	NULL

4. **Full join** − Full outer join=left outer join U right outer join

Query

Select * from student full join on student.regno = marks.regno;

Output

Name	Regno	Regno	Total
Hari	1	1	400
Subbu	2	2	450
Srinu	3	NULL	NULL
NULL	NULL	5	350

Conclusion: Hence in this way we have implemented various join Operations such as left join, right join

EXPERIMENT NO 8

Title : Study and implement various Nested and Complex queries in sql

Aim: Perform Nested and Complex queries **Theory:**

SQL subqueries or nested queries are SQL statements where we need the results from our database after using multiple filters.A subquery is put to restrict the data pool for the main query i.e., the inner query gives us the data which is the pool for the main query.

Subqueries are compatible with almost all SQL statements, for example,

1. INSERT
2. UPDATE
3. DELETE
4. SELECT

Rules to Use Subqueries in SQL:

- Subqueries need to be enclosed in the Where clause and can be used with Insert, Update, Delete, and Select statements.
- We can use comparison operators for example: <, >, > =, < =, !=, IN , Between for the subqueries.
- The subquery is always executed first and then the main query. Subquery should be enclosed within parentheses.
- Subqueries are always to the right of the comparison operators. We can't use Order By clause in the subquery; instead, we can use the Group By clause.
- We should use single-row operators with single-row subqueries and vice versa.

- We can't use Between clause with a subquery, but we can use Between in a subquery.

Types of SQL Subqueries 1.
Single Row Subquery
Returns zero or one row in results.

2. Multiple Row Subquery
Returns one or more rows in results.

3. Multiple Column Subqueries Returns one or more columns

4. Correlated Subqueries
Returns one or more columns according to the main or the outer query, thus called a correlated subquery.

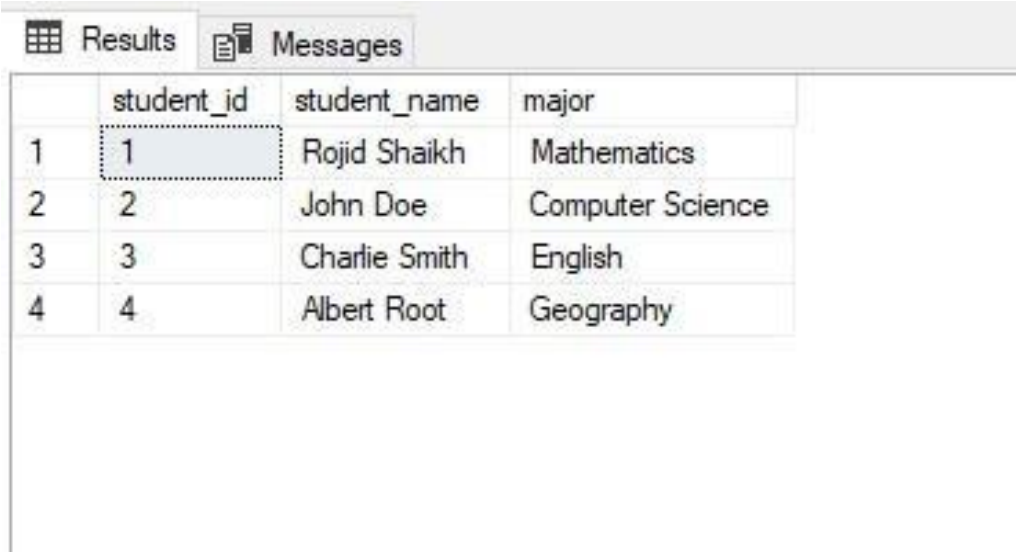
5. Nested Subqueries
We have queries within a query (inner and outer query).

Queries:

```
CREATE DATABASE Rojid00
USE Rojid00;
CREATE TABLE students (    student_id INT PRIMARY KEY,
student_name VARCHAR(50),    major VARCHAR(50));

INSERT INTO students (student_id, student_name, major )
VALUES
  (1, 'Rojid Shaikh', 'Mathematics'),
  (2, 'John Doe', 'Computer Science'),
  (3, 'Charlie Smith', 'English'),
  (4,'Albert Root', 'Geography')
```

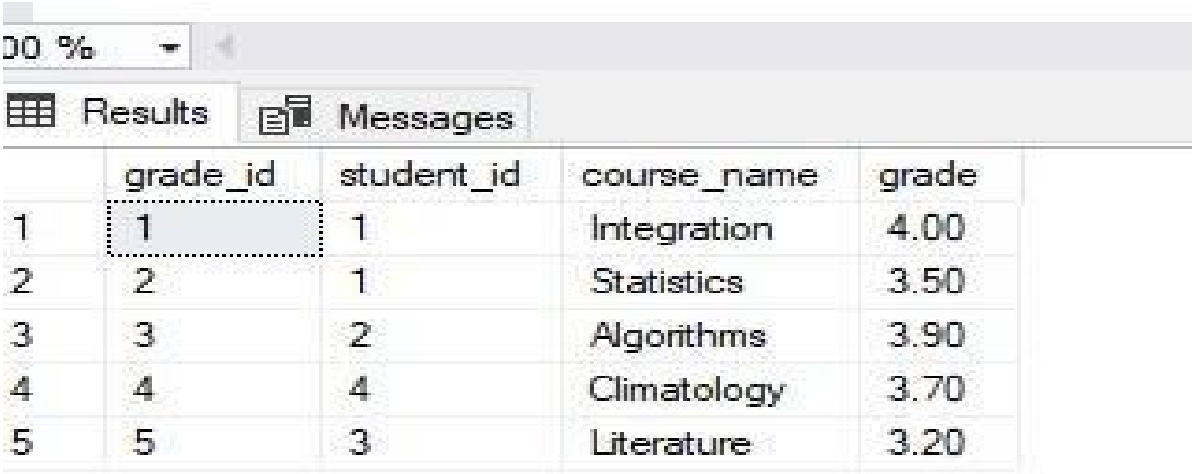
Select *from students


Results Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	2	John Doe	Computer Science
3	3	Charlie Smith	English
4	4	Albert Root	Geography

```
CREATE TABLE grades (    grade_id INT PRIMARY
KEY, student_id INT,    course_name VARCHAR(50),
grade DECIMAL(3, 2),
  FOREIGN KEY (student_id) REFERENCES students(student_id));


INSERT INTO grades (grade_id, student_id, course_name, grade)
VALUES
  (1, 1, 'Integration', 4.0), (2, 1, 'Statistics', 3.5),
  (3, 2, 'Algorithms', 3.9), (4, 4, 'Climatology', 3.7),
  (5, 3, 'Literature', 3.2);
```



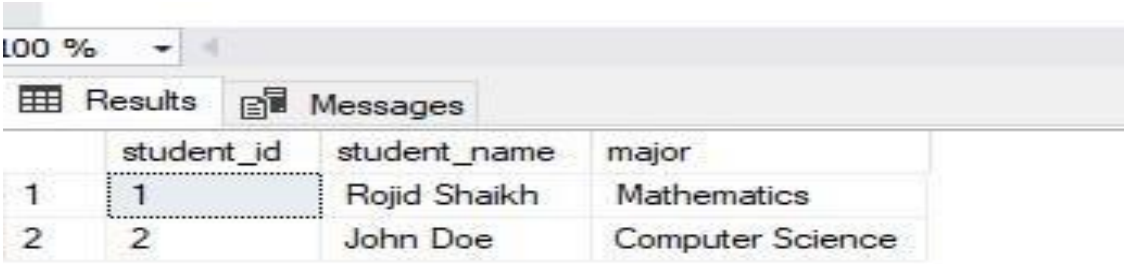
00 %

Results | Messages

	grade_id	student_id	course_name	grade
1	1	1	Integration	4.00
2	2	1	Statistics	3.50
3	3	2	Algorithms	3.90
4	4	4	Climatology	3.70
5	5	3	Literature	3.20

SELECT *FROM students

WHERE major IN ('Mathematics', 'Computer Science');



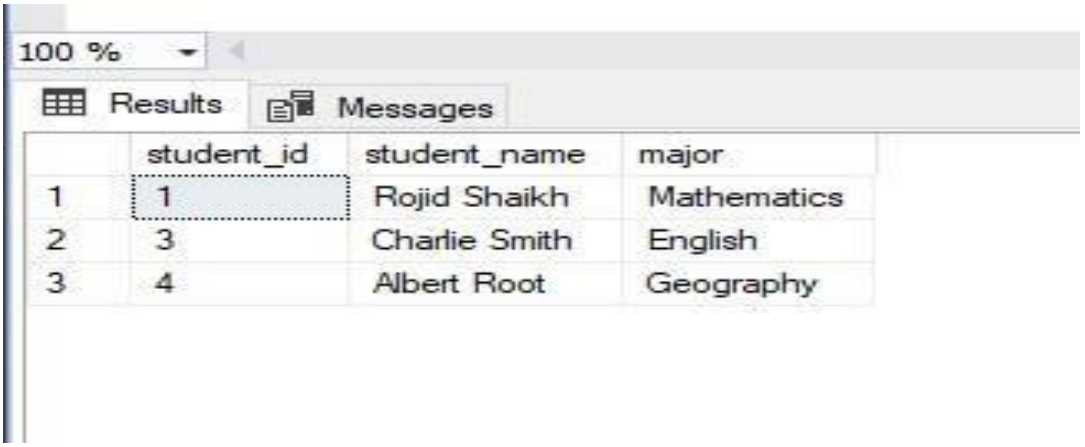
100 %

Results | Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	2	John Doe	Computer Science

SELECT *FROM students

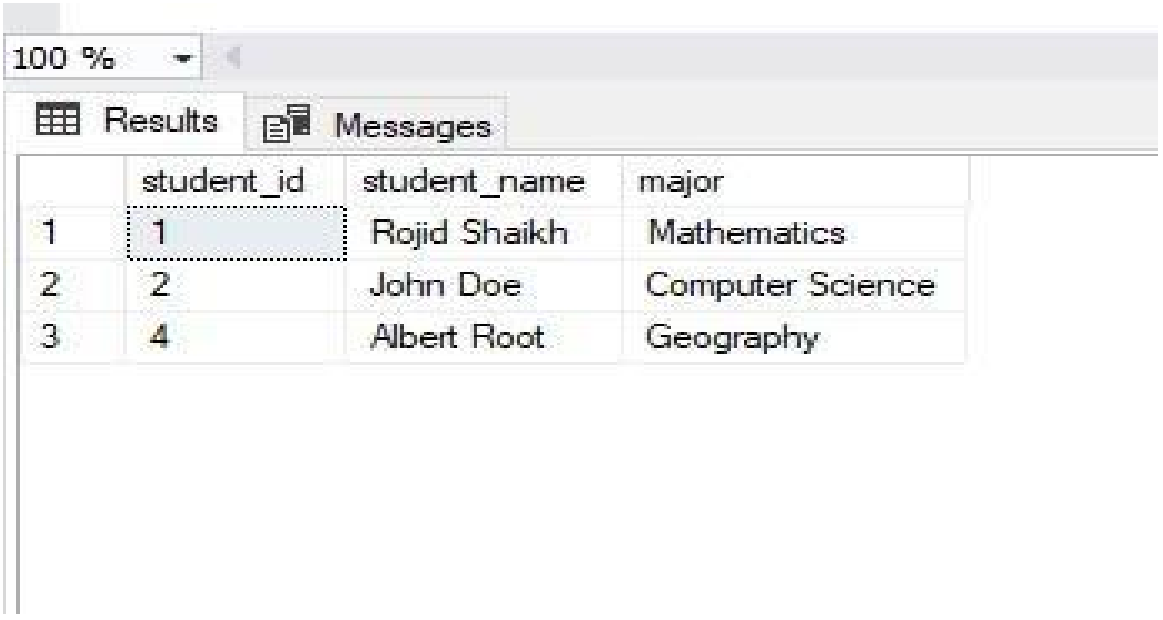
WHERE major NOT IN ('Computer Science');



	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	3	Charlie Smith	English
3	4	Albert Root	Geography

SELECT *FROM students

WHERE student_id = ANY (SELECT student_id FROM grades WHERE grade >= 3.7);



	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics
2	2	John Doe	Computer Science
3	4	Albert Root	Geography

SELECT *FROM students

WHERE student_id = ALL (SELECT student_id FROM grades WHERE grade >= 4.0);

```
SELECT *FROM students
WHERE student_id = ALL (SELECT student_id FROM grades WHERE grade >= 4.0);
```

100 %

Results | Messages

	student_id	student_name	major
1	1	Rojid Shaikh	Mathematics

SELECT student_id, student_name, major,

(SELECT AVG(grade) FROM grades WHERE grades.student_id = students.student_id) AS avg_grade

FROM students;

100 %

Results | Messages

	student_id	student_name	major	avg_grade
1	1	Rojid Shaikh	Mathematics	3.750000
2	2	John Doe	Computer Science	3.900000
3	3	Charlie Smith	English	3.200000
4	4	Albert Root	Geography	3.700000

Conclusion: Hence in this way nested and complex queries are implemented successfully.

EXPERIMENT NO 9

Title: Study and implement procedure and function in database system**.**

Aim: Perform Procedure and Function.

Theory:

A function calculates the results of a program based on the inputs provided, whereas a procedure is used to perform some tasks in a specific order.

What is a Procedure?

A procedure is a set of instructions which takes input and performs a certain task. In SQL, procedures do not return a value. In Java, procedures and functions are same and also called subroutines.

In SQL, a procedure is basically a precompiled statement which is stored inside the database. Therefore, a procedure is sometimes also called a stored procedure.

A procedure always has a name, list of parameters, and compiled SQL statements.

What is Function?

A function, in the context of computer programming languages, is a set of instructions which takes some input and performs certain tasks. In SQL, a function returns a value. In other words, a function is a tool in SQL that is used to calculate anything to produce an output for the provided inputs. In SQL queries, when a function is called, it returns the resulting value. It also controls to the calling function. However, in a function, we cannot use some DML statements like Insert, Delete, Update, etc.

Also, a function can be called through a procedure. Based on definition, there are two types of functions namely, predefined function and user defined function.

CREATE PROCEDURE HelloWorldprocedure

```
AS
PRINT “Hello World”;
exec HelloWorldprocedure
CREATE FUNCTION
dbo.helloworldfunction()

RETURNS varchar(20)
AS
BEGIN
RETURN “Hello world”; END
select dbo.helloworldfunction() as regards
```

Write a program to Calculate a factorial using procedure

```
Create procedure factor(@number int) as
begin
Declare @i int = 1,@result int=1 while
(@i&lt;=@number)
Begin
Set @result = @result * @i
Set @i += 1
End
Select @result
End
```

Exec factor 5

Conclusion: Hence in this way a procedure and function performed successfully.

EXPERIMENT NO 10

Title : Study and implement Views and Triggers in database system

Aim: Perform View and Trigger

Theory:

Views:

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax

```
CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

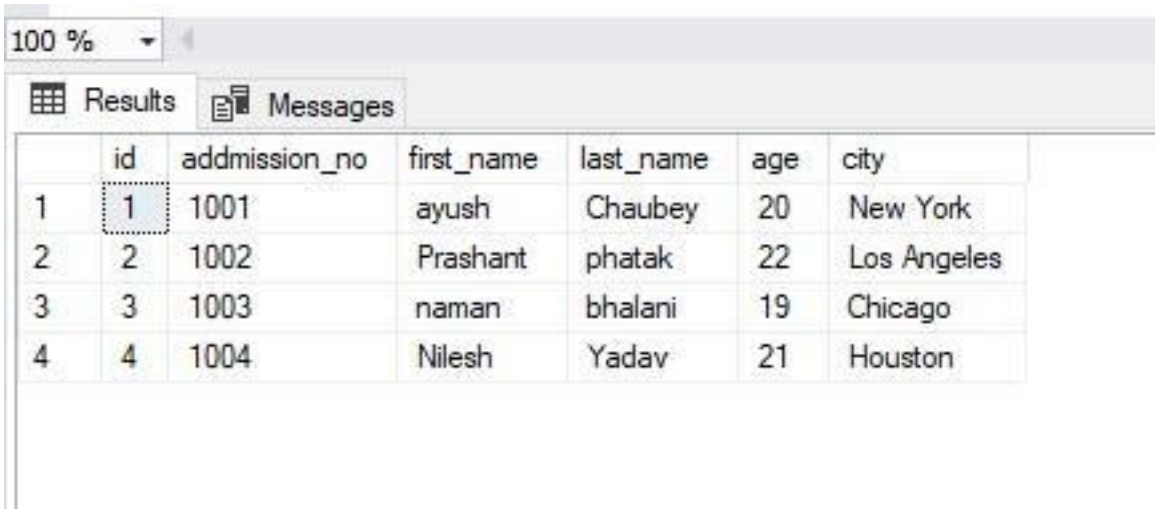
Example:

```
-- create the Student_details table CREATE
TABLE Student_details (
   id INTEGER,
addmission_no INTEGER,
first_name VARCHAR(10),
last_name VARCHAR(20),
age INTEGER,
   city VARCHAR(20)
);
```

-- insert some data into the Student_details table

```
INSERT INTO Student_details (id, addmission_no, first_name, last_name, age,
city)
VALUES
   (1, 1001, 'ayush', 'Chaubey', 20, 'New York'),
   (2, 1002, 'Prashant', 'phatak', 22, 'Los Angeles'),
   (3, 1003, 'naman', 'bhalani', 19, 'Chicago'),
   (4, 1004, 'Nilesh', 'Yadav', 21, 'Houston');

SELECT * FROM Student_details;
```


100 %

Results Messages

	id	addmission_no	first_name	last_name	age	city
1	1	1001	ayush	Chaubey	20	New York
2	2	1002	Prashant	phatak	22	Los Angeles
3	3	1003	naman	bhalani	19	Chicago
4	4	1004	Nilesh	Yadav	21	Houston

-- create the fees table

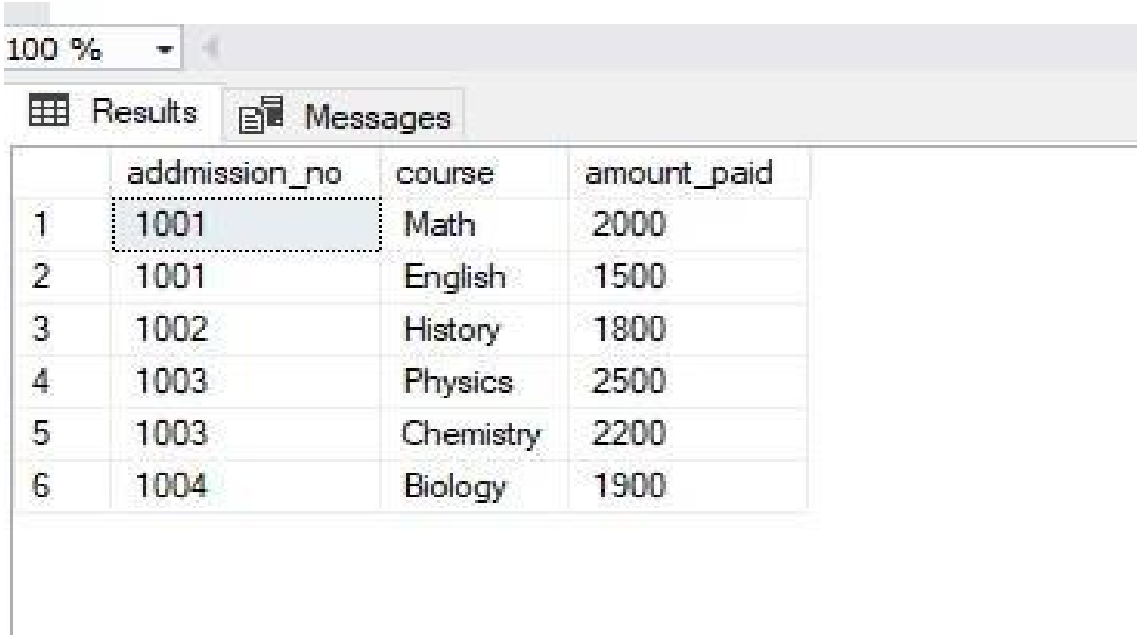
```
CREATE TABLE fees (
addmission_no INTEGER,     course
VARCHAR(20),
   amount_paid INTEGER
);
```

-- insert some data into the fees table

```
INSERT INTO fees (addmission_no, course, amount_paid)
VALUES
   (1001, 'Math', 2000),
```

```
    (1001, 'English', 1500),
    (1002, 'History', 1800),
    (1003, 'Physics', 2500),

    (1003, 'Chemistry', 2200),
    (1004, 'Biology', 1900);
SELECT * FROM fees;
```

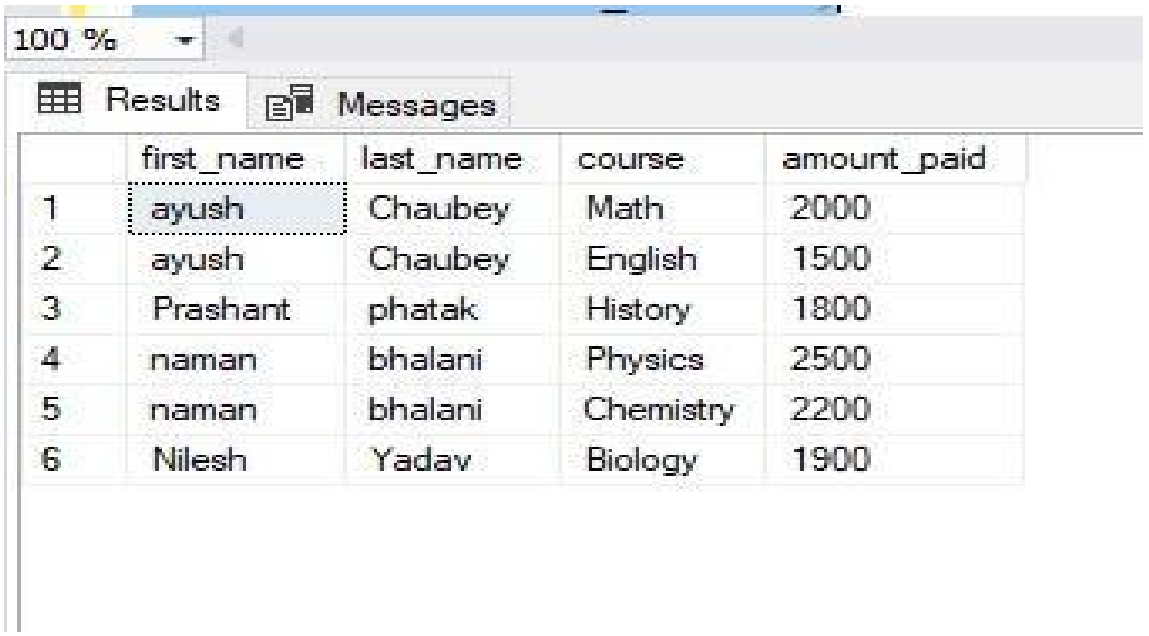

100 %

Results | Messages

	addmission_no	course	amount_paid
1	1001	Math	2000
2	1001	English	1500
3	1002	History	1800
4	1003	Physics	2500
5	1003	Chemistry	2200
6	1004	Biology	1900

```
-- create the course_enrolled view
CREATE VIEW course_enrolled
AS
SELECT first_name, last_name, course, amount_paid
FROM Student_details AS S
INNER JOIN fees AS F
ON S.addmission_no = F.addmission_no;


-- select all rows from the course_enrolled view
SELECT * FROM course_enrolled;
```



100 %

Results | Messages

	first_name	last_name	course	amount_paid
1	ayush	Chaubey	Math	2000
2	ayush	Chaubey	English	1500
3	Prashant	phatak	History	1800
4	naman	bhalani	Physics	2500
5	naman	bhalani	Chemistry	2200
6	Nilesh	Yadav	Biology	1900

TRIGGER:

A trigger is a set of SQL statements that reside in system memory with unique names. It is a specialized category of stored procedure that is called automatically when a database server event occurs. Each trigger is always associated with a table.

A trigger is called a special procedure because it cannot be called directly like a stored procedure. The key distinction between the trigger and procedure is that a trigger is called automatically when a data modification event occurs against a table. A stored procedure, on the other hand, must be invoked directly.

Advantages of Triggers

1. Triggers help us to enforce data integrity.
2. Triggers help us to validate data before inserted or updated.
3. Triggers help us to keep a log of records.

Disadvantages of Triggers

1. Triggers only allow using extended validations.

2. Triggers are invoked automatically, and their execution is invisible to the user
3. Triggers may increase the overhead of the database server.

Syntax of Trigger

```
CREATE TRIGGER schema.trigger_name
ON table_name
AFTER  {INSERT, UPDATE, DELETE}
[NOT FOR REPLICATION]
AS
{SQL_Statements}
```

Example of Trigger

```
CREATE TABLE Employee
(
  Id INT PRIMARY KEY,
  Name VARCHAR(45),
  Salary INT,
  Gender VARCHAR(12),
  DepartmentId INT
)

INSERT INTO Employee VALUES (1,'Steffan', 82000, 'Male', 3),
(2,'Amelie', 52000, 'Female', 2),
(3,'Antonio', 25000, 'male', 1),
(4,'Marco', 47000, 'Male', 2),
(5,'Eliana', 46000, 'Female', 3)

SELECT * FROM Employee;
```

Id	Name	Salary	Gender	DepartmentId
1	Steffan	82000	Male	3
2	Amelie	52000	Female	2
3	Antonio	25000	male	1
4	Marco	47000	Male	2
5	Eliana	46000	Female	3
6	Peter	62000	Male	3

also create another table named 'Employee_Audit_Test' to automatically store transaction records of each operation, such as INSERT, UPDATE, or DELETE on the Employee table:

```
CREATE TABLE Employee_Audit_Test
(
Id int IDENTITY,
Audit_Action text
)
```

Now create a trigger that stores transaction records of each insert operation on the Employee table into the Employee_Audit_Test table. Here we are going to create the insert trigger using the below statement:

```
CREATE TRIGGER trInsertEmployee
ON Employee
FOR INSERT
AS
BEGIN
  Declare @Id int
  SELECT @Id = Id from inserted
  INSERT INTO Employee_Audit_Test
```

```
  VALUES ('New employee with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is
added at ' + CAST(Getdate() AS VARCHAR(22)))   END
```

Try to add new record in the table.

```
INSERT INTO Employee VALUES (6,'Peter', 62000, 'Male', 3)
```

Id	Audit_Action
1	New employee with Id = 6 is added at Mar 24 2021 2:08PM

Now create another trigger to store transaction records of each delete operation on the Employee table into the Employee_Audit_Test table

```
CREATE TRIGGER trDeleteEmployee
ON Employee
FOR DELETE
AS
BEGIN
  Declare @Id int
  SELECT @Id = Id from deleted
  INSERT INTO Employee_Audit_Test
  VALUES ('An existing employee with Id = ' + CAST(@Id AS
VARCHAR(10)) + ' is deleted at ' + CAST(Getdate() AS VARCHAR(22)))
END
```

After creating a trigger, we will delete a record from the Employee table:

```
DELETE FROM Employee WHERE Id = 2;
```

Id	Audit_Action
1	New employee with Id = 6 is added at Mar 24 2021 2:08PM
2	An existing employee with Id = 2 is deleted at Mar 25 2021 12:26PM

create database BHAVIN

```
CREATE TABLE Employee_new
(
Id INT PRIMARY KEY,
Name VARCHAR(45),
Salary INT,
Gender VARCHAR(12),
DepartmentId INT
) ;
INSERT INTO Employee_new VALUES (1,'Steffan', 82000, 'Male', 3),
(2,'Amelie', 52000, 'Female', 2),
(3,'Antonio', 25000, 'male', 1),
(4,'Marco', 47000, 'Male', 2), (5,'Eliana',
46000, 'Female', 3)
select* from Employee_new;

CREATE TABLE Employee_Audit_Test1
(
Id int IDENTITY,
Audit_Action text
)
CREATE TRIGGER trInsertEmployee
ON Employee_new
FOR INSERT
AS
BEGIN
Declare @Id int
```

```
SELECT @Id = Id from inserted --show entry in different table
INSERT INTO Employee_Audit_Test1
VALUES ('New employee with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is added at ' +
CAST(Getdate() AS VARCHAR(22)))
END
INSERT INTO Employee_new VALUES (6,'Peter', 62000, 'Male', 3)


Delete from Employee_new


Where id = 3 ;

UPDATE Employee_new
SET Salary = 55000, Gender = 'Male'
WHERE Id = 2;

DROP TRIGGER trInsertEmployee;
DROP TABLE Employee_Audit_Test1;

select * from Employee_Audit_Test1
```

Conclusion: Hence in this way the View and trigger is implemented successfully.