PRACTICAL-1 **DATE:**_____

AIM: Introduction to SQL. Data Definition in SQL (CREATE, ALTER and DROP), Data Types.

INPUT:-

i). Introduction to SQL :-

- SQL (Structured Query Language) is a standard language used to communicate with and manipulate databases. It allows users to create, retrieve, update and delete data within a database. It is essential for managing data in relational database management systems (RDBMS) such as MySQL, PostgreSQL, Oracle and SQL Server.

ii). Data Definition in SQL (DDL) :-

- Data Definition Language (DDL) is a subset of SQL that is used to define and manage databases objects, such as tables, indexes and schemas. The primary DDL commands include :

i). CREATE
ii). ALTER
iii). DROP
iv). INSERT
v). TRUNCATE

i). CREATE :-

- The 'CREATE' command is used to create new database objects such as tables, indexes and views.

Syntax :-

```
CREATE TABLE table_name (
    column1 datatype constraints
    column2 datatype constraints
);
```

Example :-

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    BirthDate DATE,
    HireDate DATE,
    Salary DECIMAL(10,2)
);
```

ii). ALTER :-

- The 'ALTER' command is used to modify the structure of an existing database object

Syntax :-

```
# Adding a column
ALTER TABLE table_name
ADD column_name datatype constraints;
# Modifying a column
ALTER TABLE table_name
MODIFY column_name new_datatype
                    new_constraints;
# Dropping a column
ALTER TABLE table_name
DROP COLUMN column_name;
```

Example :-

```
# Adding a column
ALTER TABLE Employees
ADD Email VARCHAR(100);
# Modifying a column
ALTER TABLE Employees
MODIFY Salary DECIMAL(12,2);
# Dropping a column
ALTER TABLE Employees
DROP COLUMN Email;
```

iii). DROP :-

- The 'DROP' command is used to delete existing database objects.

Syntax :-

```
# Dropping a Table
DROP TABLE table_name ;
# Dropping a column
ALTER TABLE table_name
DROP COLUMN column_name ;
```

Example :-

```
# Dropping a Table
DROP TABLE Employees ;
# Dropping a column
ALTER TABLE Employees
DROP COLUMN EMAIL ;
```

iv). INSERT :-

- The 'INSERT' command is used to add new rows of data into a table.

Syntax :-

```
INSERT INTO table_name VALUES
    (value1, value2 . . . . , valueN) ;
```

Example :-

```
INSERT INTO Employees VALUES (1,
    'John', 'Doe', '1985-01-01', '2020-05-15
    50000) ;
```

v). TRUNCATE :-

- The 'TRUNCATE' command is used to remove all rows from a table without logging the individual row deletions. It is faster than the 'DELETE' command because it does not generate individual row delete transactions. However, 'TRUNCATE' is a DDL command and cannot be rolled back it executed.

Syntax :- `TRUNCATE TABLE table_name ;`

Example :- `TRUNCATE TABLE Employees ;`

iii). Data Types in SQL :-

- Data types define the kind of data that can be stored in a column. Some common SQL data types include :

i). Numeric Data Types :-

- 'INT' → Integer values
- 'Float' → Floating-point numbers
- 'DECIMAL (p,s)' → Fixed-point numbers with precision 'p' and scale 's'.

ii). Character String Data Types :-

- 'CHAR(n)' → Fixed-length character strings.
- 'VARCHAR(n)' → Variable-length character strings.

iii). Date and Time Data Types :-

- 'DATE' → Date values
- 'TIME' → Time values
- 'DATETIME' → Date and Time values

iv). Boolean Data Types :-

- 'BOOLEAN' → True/False values

v). Binary Data Types :-

- 'BLOB' → Binary Large Objects for storing binary data.

vii. Other Data Types :-

- 'ENUM' → Enumeration of predefined values.
- 'SET' → A set of predefined values.

Example :-

```
CREATE TABLE Products (
        ProductID INT PRIMARY KEY,
        ProductName VARCHAR(100),
        Price DECIMAL(10,2),
        ReleaseDate DATE
);
```

SQL Queries Example :-

i). Create a Table :-

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100),
    PublishedYear INT,
    Genre VARCHAR(50)
);
```

ii). Insert Data :-

```
INSERT INTO Books(BookID, Title, Author, PublishedYear, Genre)
VALUES
(1, 'To Kill a Mockingbird', 'Harper Lee', 1960, 'Fiction'),
(2, '1984', 'George Orwell', 1949, 'Dystopian'),
(3, 'Moby Dick', 'Herman Melville', 1851, 'Adventure');
```

iii). Fetch Data :-

```
SELECT * FROM Books;
```

Output :-

BookID	Title	Author	Published Year
1	To Kill a Mockingbird	Harper Lee	1960
2	1984	George Orwell	1949
3	Moby Dick	Herman Melville	1851

Genre
Fiction
Dystopian
Adventure

iv). ALTER (DELETE) COLUMN :-

```
ALTER TABLE Books
DROP COLUMN Author;
```

v). Fetch Data :-

```
SELECT * FROM Books;
```

Output :-

BookID	Title	Published Year	Genre
1	To Kill a Mocking Bird	1960	Fiction
2	1984	1949	Dystopian
3	Moby Dick	1851	Adventure

vii. DROP the table :-

```
DROP TABLE Books ;
```

Output :-

BookID	Title	Published Year	Genre

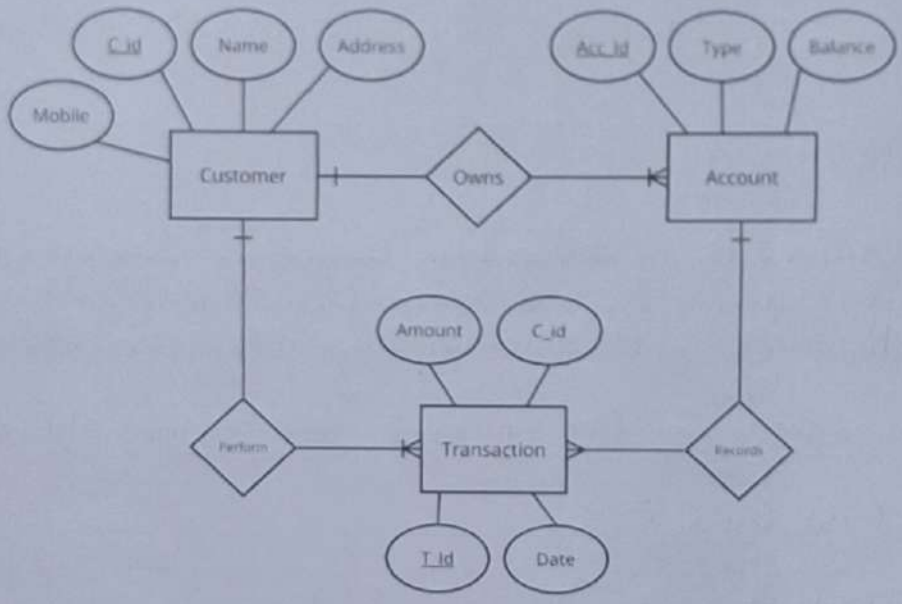
PRACTICAL-2 **DATE:**_____

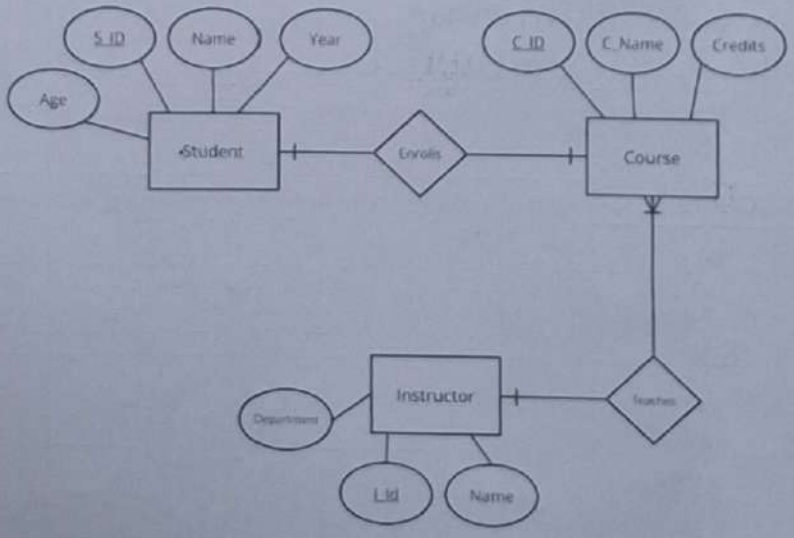
AIM: .Draw E-R diagram and convert entities and relationships to relationtable for a given scenario.

A. Two assignments shall be carried out i.e. consider two differentscenarios (eg. bank, college)

B. Write relational algebra queries for a given set of relations.

INPUT:-





A). Convert Entity and Relationship to relation table for a given scenario.

i). Bank System :-

a). Entities :-

- Customer (CustomerID, Name, Address, Mobile)
- Accounts (AccountID, AccountType, Balance, CustomerID)
  Foreign Key : CustomerID references Customer. (CustomerID
- Transactions (TransactionID, Date, Amount, AccountID)
  foreign key : AccountID references Account. AccountID

b). Relationship :-

- Owns : A relationship between Customer and Account.
- Performs : A relationship between Account and Transaction
- Records : A relationship between Account and Transaction

* Conversion of Entity and Relationship to Relational Table :-

a). Customer :-

Attribute	Type	Key
CustomerID	INT	Primary Key
Name	VARCHAR	
Address	VARCHAR	
Mobile	VARCHAR	

b). Account :-

Attribute	Type	Key
AccountID	INT	Primary Key
AccountType	VARCHAR	
Balance	DECIMAL	
CustomerID	INT	Foreign Key

c). Transaction :-

Attribute	Type	Key
TransactionID	INT	Primary Key
Date	DATE	
Amount	DECIMAL	
AccountID	INT	Foreign Key

d). Owns (Relationship) :-

Relationship Name	Primary Entity	Foreign Entity	Foreign Key
Owns	Customer	Account	'CustomerID' in Account References 'CustomerID' in Customer

e). Performs (Relationship) :-

Relationship Name	Primary Entity	Foreign Entity	Foreign Key.
Performs	Customer	Transaction	'CustomerID' in Transaction References 'CustomerID' in Customer

f). Records (Relationship) :-

Relationship Name	Primary Entity	Foreign Entity	Foreign Key
Records	Accounts	Transaction	'AccountID' in Transaction References 'AccountID' in Accounts

ii). College System :-

a). Entity :-

- Student (StudentID, Name, Age, Year)
- Course (CourseID, CourseName, Credits)
- Instructor (InstructorID, Name, Department)

b). Relationship :-

- Enrolls In : A relationship between Student and Course.
- Teaches : A relationship between Instructor and Course.

* Conversion of Entity and Relationship to Relational Table :-

a). Student :-

Attribute	Type	Key
StudentID	INT	Primary Key
Name	VARCHAR	
Age	INT	
Year	INT	

b). Course :-

Attribute	Type	Key
CourseID	INT	Primary Key
CourseName	VARCHAR	
Credits	INT	

c). Instructor :-

Attribute	Type	Key
InstructorID	INT	Primary Key
Name	VARCHAR	
Department	VARCHAR	

d). Enrolls (Relationship) :-

Attribute	Type	Key
StudentID	INT	Foreign Key (References 'Student.StudentID')
CourseID	INT	Foreign Key (References 'Course.CourseID')

e). Teaches (Relationship) :-

Attribute	Type	Key
InstructorID	INT	Foreign Key (References 'Instructor.InstructorID')
CourseID	INT	Foreign Key (References 'Course.CourseID')

B). Write relational algebra queries for a given set of relations.

i). Bank System :-

- Find all customers with a balance greater than $10,000

$$\sigma_{Balance > 10000}(Account) \bowtie Customer$$

- List all transactions of a accountID = 123

$$\sigma_{AccountID = 123}(Transaction)$$

- Find all accounts owned by customerID = 101

$$\sigma_{CustomerID = 101}(Account)$$

ii). College System :-

- Find all students enrolled in a specific course.

$$\sigma_{CourseID = 305}(Student\_Course) \bowtie Student$$

- List all courses taught by a specific instructor ID = 202

$$\sigma_{InstructorID = 202}(Teaches) \bowtie Course$$

- Find all instructors teaching in a specific department = 'Computer Science'

$$\sigma_{Department = 'Computer\ Science'}(Instructor) \bowtie Teaches \bowtie Course$$

PRACTICAL-3 **DATE:_____**

AIM: Design a Database and create required tables. For e.g. Bank, College Database.Perform the following: a.Viewing all databases, Creating a Database, Viewing all Tables in a Database, Creating Tables (With and Without Constraints), Inserting/Updating/Deleting Records in a Table, Apply the constraints like Primary Key , Foreign key, NOT NULL to the tables.

INPUT:-

i). Database :-

- A database is an organized collection of data, generally stored and accessed electronically from a computer system. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data. They can be managed using a Database Management System (DBMS).

ii). Table :-

- A table is a collection of related data held in a structured format within a database. It consists of rows and columns, where each row represents a unique record and each column represents a field in the record.

iii). Data Manipulation Language (DML) :-

- Data Manipulation Language (DML) is a subset of SQL used for adding, deleting, and modifying data in a database. DML commands are used to manipulate data stored in database objects like tables.

iv). Data Definition Language (DDL) :-

- DDL is a subset of SQL used to define and manages all database objects, such as tables, indexes and views. DDL commands are used to create, modify and delete database structure but do not typically manipulate the data within those structures.

vi). Keys :-

- Keys are attributes or sets of attributes that help to identify a row in a table. They are crucial for establishing relationships between tables and ensuring data integrity.

i). **Primary Key** : A column or a combination of columns that uniquely identifies each row in a table. Each table can have only one primary key, and it cannot contain null values.

ii). **Foreign Key** : A column or a combination of columns that establish a link between data in two tables. It refers to the primary key in another table, ensuring referential integrity.

iii). **Composite Key** : A primary key composed of two or more columns used to identify a record uniquely.

iv). **Alternate Key** : A candidate key that is not chosen as the primary key.

vii). DDL Commands :-

- DDL commands are used to define and manage database objects. Common DDL commands include :

i). CREATE : Creates a new table.
ii). ALTER : Modifies an existing database object.
iii). DROP : Deletes an existing database object.
iv). TRUNCATE : Removes all records from a table but does not delete the table structure.

vii. DML Commands :-

- DML command are used to manipulate the data within database objects. Common DML commands include :

i). INSERT : Adds new records to a table.
ii). UPDATE : Modifies existing records in a table.
iii). DELETE : Removes records from a table.
iv). SELECT : Retrieves data from one or more tables.

SQL Queries :-

i). CREATE Database Bank ;

ii).
```
CREATE table Customers (
    Customer_id Primary key Identity(1,1),
    name varchar(20) Not Null,
    Contactno varchar(10) Not Null
);
```

iii).
```
INSERT INTO Customers (name, Contactno) VALUES
('John Cina', '1234567890'),
('Joe Biden', '246801214'),
('Michael Jackson', '3456780912');
```

iv). Select * from Customers ;

Customer_id	Name	Contactno
1	John Cina	1234567890
2	Joe Biden	246801214
3	Michael Jackson	3456780912

v).
```
CREATE table Accounts (
    acc_id int Primary key Identity(1,1),
    Customer_id int,
    acc_type VARCHAR(10) NOT NULL,
    Balance Decimal(10,2) NOT NULL,
    Foreign Key ((ustomer_id) References
                        (ustomer ((ustomer_id)
);
```

vi).
```
INSERT INTO Accounts ((ustomer_id, acc_type, Balance)
VALUES
    (1, 'Savings', 1000.00)
    (2, '(urrent', 2500.50)
    (3, 'Salary', 1500.75);
```

vii).
```
SELECT * FROM Accounts;
```

acc_id	(ustomer_id	acc_type	Balance
1	1	Savings	1000.00
2	2	(urrent	2500.50
3	3	Salary	1500.75

viii).
```
CREATE Table Transactions (
    t_id int Primary key Identity(1,1),
    acc_id int,
    transaction_type VARCHAR(20) Not Null,
    amount Decimal(10,2) Not Null,
    Foreign Key (acc_id) References Accounts(acc_id)
);
```

ix.
```
INSERT INTO Transactions (acc_id, transaction_type,
                          amount) VALUES
    (1, 'Deposit', 200.00),
    (2, 'Withdrawal', -50.75),
    (3, 'Deposit', 300.00);
```

x.
```
SELECT * FROM Transactions ;
```

t_id	acc_id	transaction_type	Amount
1	1	Deposit	200.00
2	2	Withdrawal	-50.75
3	3	Deposit	300.00

xi.
```
UPDATE Customers set name = "Justin Bieber"
  Where Customer_id = 1 ;
```

xii.
```
SELECT * FROM Customers ;
```

Customer_id	Name	Contactno
1	Justin Bieber	1234567890
2	Joe Biden	2468012 14
3	Michael Jackson	3456780912

xiii.
```
DELETE FROM Transactions where t_id=3 ;
```

xiv.
```
SELECT * FROM Transactions ;
```

t_id	ac_id	transaction_type	Amount
1	1	Deposit	200.00
2	2	Withdrawal	-50.75

PRACTICAL-4 **DATE:**_____

AIM: For a given set of relation schemes, create tables and perform the following Simple Queries, Simple Queries with Aggregate functions, Queries with Aggregate functions (group by and having clause), Queries involving- Date Functions, String Functions , Math Functions

INPUT:-

i). Relation Schema :-

- A Relation Schema is a blueprint or structure that defines a relation in a relational database. It specifies the name of the relation and the names and types of its attributes.

- Key components of a relation schema include :

i). Relation Name
ii). Attributes
iii). Domain
iv). Keys

ii). Aggregate Functions :-

i). COUNT() : Returns the number of rows.
ii). AVG() : Returns the average value.
iii). SUM() : Returns the sum of values.
iv). Min() : Returns the minimum value.
v). Max() : Returns the maximum value.

iii). Date Functions :-

i). Year() : Extracts the year from a date.
ii). CURDATE() : Returns the current date.

iv). String Functions :-

i). LIKE : Used to search for a specified pattern in column
ii). CONCAT() : Concatenates two or more strings.

v). Math Functions :-

i). ROUND() : Rounds a number to a specific number of decimal places.
ii). ABS() : Returns the absolute value of a number.
iii). CEIL() : Returns the smallest integer value greater than or equal to a number.
iv). FLOOR() : Returns the largest integer value less than or equal to a number.

SQL Queries :-

i) Create Customers, Accounts and Transactions table.

```
// Customers Table
CREATE TABLE Customers (
        CustomerID INT AUTO_INCREMENT PRIMARY KEY,
        FirstName VARCHAR(50) NOT NULL,
        LastName VARCHAR(50) NOT NULL,
        DateofBirth DATE NOT NULL,
        Gender CHAR(1),
        Mobile VARCHAR(10) NOT NULL
    );
// Accounts Table
CREATE TABLE Accounts (
        AccountID INT AUTO_INCREMENT PRIMARY KEY
        CustomerID INT,
        AccountType VARCHAR(20) NOT NULL,
        Balance DECIMAL(15,2),
```

```
                FOREIGN KEY (CustomerID) REFERENCES
                                    Customers (CustomerID)
            );
// Transactions Table
CREATE TABLE Transactions (
        TransactionID INT AUTO_INCREMENT PRIMARY KEY,
        AccountID INT,
        TransactionDate DATE NOT NULL,
        Amount Decimal(15,2) NOT NULL,
        TransactionType VARCHAR(20) NOT NULL,
        FOREIGN KEY (AccountID) REFERENCES
                                    Accounts(AccountID)
    );
```

ii). Insert data into Customers, Accounts and Transactions.

```
// Customers
INSERT INTO Customers (FirstName, LastName, DateofBirth,
                        Gender, Mobile) VALUES
 ('John', 'Doe', '1980-01-05', 'M', '1234567890'),
 ('Jane', 'Smith', '1985-05-23', 'F', '0987654321');
// Accounts
 INSERT INTO Accounts (CustomerID, AccountType, Balance)
      VALUES
  (1, 'Savings', 5000.00),
  (2, 'Checking', 1500.00),
  (1, 'Checking', 2000.00);
// Transactions
 INSERT INTO Transactions (AccountID, TransactionDate,
                    Amount, TransactionType) VALUES
  (1, '2024-01-10', 500.00, 'Deposit'),
  (2, '2024-01-15', 200.00, 'Withdrawal'),
  (3, '2024-01-20', 300.00, 'Deposit');
```

iii). Simple Queries :-

a). View all Customers :

```
SELECT * FROM Customers ;
```

CustomerID	FirstName	LastName	DateofBirth	Gender	Mobile
1	John	Doe	1980-01-15	M	1234567890
2	Jane	Smith	1985-05-23	F	0987654321

b). View all Accounts :

```
SELECT * FROM Accounts ;
```

AccountID	CustomerID	AccountType	Balance
1	1	Savings	5000.00
2	2	Checking	1500.00
3	1	Checking	2000.00

c). View all Transactions :

```
SELECT * FROM Transactions ;
```

TransactionID	AccountID	TransactionDate	Amount	TransactionType
1	1	2024-01-10	500.00	Deposit
2	2	2024-01-15	200.00	Withdrawal
3	3	2024-01-20	300.00	Deposit

iv). Simple Queries with Aggregate Functions :-

a). Total balance in all accounts :

```
SELECT SUM(Balance) AS TotalBalance FROM Accounts;
```

TotalBalance
8500.00

b). Average balance in Savings accounts :

```
SELECT AVG(Balance) As AverageSavingsBalance FROM
ACCOUNTS WHERE AccountType = 'Savings';
```

AverageSavingsBalance
5000.00

v). Queries with Aggregate Functions (GROUP BY and Having) :-

a). Total balance per account type :

```
SELECT AccountType, SUM(Balance) AS TotalBalance
FROM ACCOUNTS
GROUP BY AccountType;
```

AccountType	TotalBalance
Savings	5000.00
Checking	3500.00

b). Total balance per customer :

```
SELECT CustomerID, SUM(Balance) As TotalBalance
FROM Accounts GROUP BY CustomerID
HAVING SUM(Balance) > 3000;
```

CustomerID	TotalBalance
1	7000.00

vi). Queries Involving Date Functions :-

a). Transactions in the last month :

```
SELECT * FROM Transactions
WHERE TransactionDate >= DATEADD(MONTH, -1, GETDATE());
```

TransactionID	AccountID	TransactionDate	Amount	TransactionType
1	1	2024-01-10	500.00	Deposit
2	2	2024-01-15	200.00	Withdrawal
3	3	2024-01-20	300.00	Deposit

b). Customer's ages :

```
SELECT FirstName, LastName,
DATEDIFF(YEAR, DateOfBirth, GETDATA()) AS Age
FROM Customers ;
```

FirstName	LastName	Age
John	Doe	44
Jane	Smith	39

vii). Queries Involving String Functions :-

a). Customer's full names :

```
SELECT CONCAT(FirstName,' ', LastName) AS FullName
FROM Customers ;
```

FullName
John Doe
Jane Smith

viii). Queries Involving Math Functions :-

a). Round the balance to the nearest whole number :

```
SELECT AccountID, ROUND(Balance, 0) AS RoundedBalance
FROM Accounts;
```

AccountID	RoundedBalance
1	5000
2	1500
3	2000

b). Find the absolute value of transactions :

```
SELECT TransactionID, ABS(Amount) AS AbsoluteAmount
FROM Transactions;
```

TransactionID	AbsoluteAmount
1	500.00
2	200.00
3	300.00

PRACTICAL-5 **DATE:**_____

AIM: Perform the following:

a. Altering a Table,Dropping/Truncating/Renaming Tables,
b.Backing up / Restoring a Database.
c.Use of grant, revoke.

INPUT:-

Step-1 : Create Customer, Accounts and Transactions table.

```
// Customer Table
CREATE TABLE Customer (
        CustomerID INT AUTO_INCREMENT PRIMARY KEY,
        FirstName VARCHAR(50) NOT NULL,
        LastName VARCHAR(50) NOT NULL,
        DateOfBirth DATE NOT NULL,
        Gender CHAR(1),
        Phone VARCHAR(10) NOT NULL
    );
// Accounts Table
CREATE TABLE Accounts (
        AccountID INT AUTO_INCREMENT PRIMARY KEY,
        CustomerID INT,
        AccountType VARCHAR(20) NOT NULL,
        Balance DECIMAL(15,2) NOT NULL,
        FOREIGN KEY (CustomerID) REFERENCES
                            Customer (CustomerID)
    );
// Transactions Table
CREATE TABLE Transactions (
        TransactionID INT AUTO_INCREMENT PRIMARY KEY,
        AccountID INT,
        TransactionDate DATE NOT NULL,
        Amount DECIMAL(15,2) NOT NULL,
        TransactionType VARCHAR(20) NOT NULL,
        FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);
```

Step-2: Inserting Data into Tables.

```
// Customers Table
INSERT INTO Customers
    (FirstName, LastName, DateOfBirth, Gender, PhoneNumber)
VALUES
('John', 'Doe', '1980-01-05', 'Male', '1234567890'),
('Jane', 'Smith', '1985-05-23', 'Female', '0987654321');

// Accounts Table
INSERT INTO Accounts
    (CustomerID, AccountType, Balance) VALUES
(1, 'Savings', 5000.00),
(2, 'Checking', 1500.00),
(1, 'Checking', 2000.00);

// Transactions Table
INSERT INTO Transactions
    (AccountID, TransactionDate, Amount, TransactionType)
VALUES
(1, '2024-01-10', 500.00, 'Deposit'),
(2, '2024-01-15', 200.00, 'Withdrawal'),
(3, '2024-01-20', 300.00, 'Deposit');
```

Step-3: Fetch data

```
// Customer Table
SELECT * FROM Customers;
// Accounts Table
SELECT * FROM Accounts;
// Transactions Table
SELECT * FROM Transactions;
```

// Customers Table

CustomerID	FirstName	LastName	DateofBirth	Gender	PhoneNumber
1	John	Doe	1980-01-15	Male	1234567890
2	Jane	Smith	1985-05-23	Female	0987654321

// Accounts Table

AccountID	CustomerID	AccountType	Balance
1	1	Savings	5000.00
2	2	Checking	1500.00
3	1	Checking	2000.00

// Transactions Table

TransactionID	AccountID	TransactionDate	Amount	TransactionType
1	1	2024-01-10	500.00	Deposit
2	2	2024-01-15	200.00	Withdrawal
3	3	2024-01-20	300.00	Deposit

---

9. Altering, Dropping / Truncating / Renaming Tables :-

i). Altering a Table :-

```
ALTER TABLE Customers
ADD Address VARCHAR(255);
```

CustomerID	Firstname	Lastname	DateofBirth	Gender	PhoneNumber	Address
1	John	Doe	1980-01-15	Male	1234567890	Null
2	Jane	Smith	1985-05-23	Female	0987654321	Null

ii). Dropping a Table :-

```
DROP TABLE Transactions;
```

iii). Truncating a Table :-

```
TRUNCATE TABLE Accounts;
```

AccountID	CustomerID	AccountType	Balance

iv). Renaming a Table :-

```
ALTER TABLE Customers
RENAME TO BankCustomers;
```

b). Backing UP / Restoring a Database :-

i). Backing Up a Database :-

```
BACKUP DATABASE BankDB
TO DISK = 'C:\backups\BankDB.bak';
```

ii). Restore a Database :-

```
RESTORE DATABASE BankDB
FROM DISK = 'C:\backups\BankDB.bak';
```

c). Use of grant and Revoke :-

i). Grant Priviliges :-

```
GRANT SELECT, INSERT, UPDATE ON BankDB.* To
'bank_user'@'localhost' IDENTIFIED BY 'password';
```

ii). Revoke Priviliges :-

```
REVOKE INSERT, UPDATE ON BankDB.* FROM 'bank_user'
@'localhost';
```

PRACTICAL-6 **DATE:_____**

AIM: Join Queries- Inner Join, Outer Join Subqueries- With IN clause, With EXISTS clause

INPUT:-

Step-1 : Create BankCustomers, Accounts, Transactions table.

```
// BankCustomers Table
CREATE TABLE BankCustomers (
        CustomerID INT AUTO_INCREMENT PRIMARY KEY,
        FirstName VARCHAR(50) NOT NULL,
        LastName VARCHAR(50) NOT NULL,
        DOB DATE NOT NULL,
        Gender CHAR(1),
        PhoneNumber VARCHAR(10) NOT NULL
  );
// Accounts Table
CREATE TABLE Accounts (
        AccountID INT AUTO_INCREMENT PRIMARY KEY,
        CustomerID INT,
        AccountType VARCHAR(20) NOT NULL,
        Balance DECIMAL(15,2) NOT NULL,
        FOREIGN KEY (CustomerID) REFERENCES
                        BankCustomers (CustomerID)
  );
// Transactions Table
CREATE TABLE Transactions (
        TransactionID INT AUTO_INCREMENT PRIMARY KEY,
        AccountID INT,
        TransactionDate DATE NOT NULL,
        Amount DECIMAL(15,2) NOT NULL,
        TransactionType VARCHAR(20) NOT NULL,
        FOREIGN KEY (AccountID) REFERENCES
                        Accounts (AccountID)
```

Step-2 : Insert data into tables.

```
// BankCustomers Table
INSERT INTO BankCustomers
  (FirstName, LastName, DOB, Gender, PhoneNumber) VALUES
('John', 'Doe', '1980-01-15', 'M', '123456789'),
('Jane', 'Smith', '1985-05-23', 'F', '2345678901'),
('Alan', 'Brown', '1990-08-12', 'M', '3456789012');
// Accounts Table
INSERT INTO Accounts
  (CustomerID, AccountType, Balance) VALUES
(1, 'Savings', 5000.00),
(2, 'Checking', 1500.00),
(1, 'Checking', 2000.00);
// Transactions Table
INSERT INTO Transactions
  (AccountID, TransactionDate, Amount, TransactionType)
VALUES
  (1, '2024-01-10', 500.00, 'Deposit'),
  (2, '2024-01-15', 200.00, 'Withdrawal'),
  (3, '2024-01-20', 300.00, 'Deposit');
```

Step-3 : Fetch the data

```
// BankCustomers Table
SELECT * FROM BankCustomers;
// Accounts Table
SELECT * FROM Accounts;
// Transactions Table
SELECT * FROM Transactions;
```

// Bank Customers Table

CustomerID	FirstName	LastName	DoB	Gender	PhoneNumber
1	John	Doe	1980-01-15	M	1234567890
2	Jane	Smith	1985-05-23	F	2345678901
3	Alan	Brown	1990-08-12	M	3456789012

// Accounts Table

AccountID	CustomerID	AccountType	Balance
1	1	Savings	5000.00
2	2	Checking	1500.00
3	1	Checking	2000.00

// Transactions Table

TransactionID	AccountID	TransactionDate	Amount	TransactionType
1	1	2024-01-10	500.00	Deposit
2	2	2024-01-15	200.00	Withdrawal
3	3	2024-01-20	300.00	Deposit

---

a). Join Queries :-

i). Inner Join :-

```
SELECT
    BankCustomers.CustomerID,
    BankCustomers.FirstName,
    BankCustomers.LastName,
    Accounts.AccountType,
    Accounts.Balance
FROM
    BankCustomers
INNER JOIN
    Accounts
ON
    BankCustomers.CustomerID =
                    Accounts.CustomerID ;
```

CustomerID	FirstName	LastName	AccountType	Balance
1	John	Doe	Savings	5000.00
2	Jane	Smith	Checking	1500.00
1	John	Doe	Checking	2000.00

ii). Left Join :-

```
SELECT
    BankCustomers.CustomerID,
    BankCustomers.FirstName,
    BankCustomers.LastName,
    Accounts.AccountType,
    Accounts.Balance
FROM
    BankCustomers
LEFT JOIN
    Accounts
ON
    BankCustomers.CustomerID = Accounts.CustomerID ;
```

CustomerID	FirstName	LastName	AccountType	Balance
1	John	Doe	Savings	5000.00
1	John	Doe	Checking	2000.00
2	Jane	Smith	Checking	1500.00
3	Alan	Brown	NULL	NULL

iii). Right Join :-

```
SELECT
    BankCustomers.CustomerID,
    BankCustomers.FirstName,
    BankCustomers.LastName,
    Accounts.AccountType,
    Accounts.Balance
FROM
    BankCustomers
RIGHTJOIN
    Accounts
ON
    BankCustomers.CustomerID = Accounts.CustomerID ;
```

CustomerID	FirstName	LastName	AccountType	Balance
1	John	Doe	Savings	5000.00
2	Jane	Smith	Checking	1500.00
1	John	Doe	Checking	2000.00

iv). Full-Outer Join :-

```
SELECT
    BankCustomers.CustomerID,
    BankCustomers.FirstName,
    BankCustomers.LastName,
    Accounts.AccountType,
    Accounts.Balance
FROM
    BankCustomers
FULL OUTER JOIN
    Accounts
ON
    BankCustomers.CustomerID =
                    Accounts.CustomerID;
```

CustomerID	FirstName	LastName	AccountType	Balance
1	John	Doe	Savings	5000.00
2	Jane	Smith	Checking	1500.00
1	John	Doe	Checking	2000.00
3	Alan	Brocon	NULL	NULL

V). CROSS Join :-

```
SELECT
    BankCustomers.FirstName,
    BankCustomers.LastName,
    Accounts.AccountType,
    Accounts.Balance
FROM
    BankCustomers
CROSS JOIN
    Accounts;
```

FirstName	LastName	AccountType	Balance
John	Doe	Savings	5000.00
John	Doe	Checking	2000.00
Jane	Smith	Savings	5000.00
Jane	Smith	Checking	2000.00
Alan	Brown	Savings	5000.00
Alan	Brown	Checking	2000.00

vii. Self-Join :-

```
SELECT
    a1.AccountID AS AccountID1,
    a1.CustomerID AS CustomerID1,
    a1.AccountType AS AccountType1,
    a1.Balance AS Balance1,
    a2.AccountID AS AccountID2,
    a2.CustomerID AS CustomerID2,
    a2.AccountType AS AccountType2,
    a2.Balance AS Balance2
FROM
    Accounts a1
INNER JOIN
    Accounts a2
ON
    a1.CustomerID = a2.CustomerID
    AND a1.Balance > a2.Balance;
```

AccountID1	CustomerID1	AccountType1	Balance1	AccountID2	CustomerID2	AccountType2	Balance2
1	1	Savings	5000.00	3	1	Checking	2000.00

b). Subqueries with IN and Exists Clause :-

i). IN Clause :-

```
SELECT
    FirstName,
    LastName,
FROM
    BankCustomers
WHERE
    CustomerID IN (
        SELECT DISTINCT CustomerID
        FROM Accounts
        WHERE AccountID IN(
            SELECT AccounID
            FROM Transactions
        )
    );
```

FirstName	LastName
John	Doe
Jane	Smith

ii). Exists Clause :-

```
SELECT
    FirstName,
    LastName
From
    BankCustomers bc
WHERE
    EXISTS (
        Select 1
        FROM Accounts a
```

```
    JOIN Transactions t ON a.AccountID = t.AccountID
    WHERE bc.CustomerID = a.CustomerID
);
```

FirstName	LastName
John	Doe
Jane	Smith

PRACTICAL-7 **DATE:**_____

AIM:. For a given set of relation tables perform the following:

i. Creating views (with and without check option),

ii. Dropping views, Selecting from a view

INPUT:-

Step-1 : Create BankCustomers, Accounts and Transactions table

```
// BankCustomers Table
CREATE TABLE BankCustomers (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    Email VARCHAR(100)
);
// Accounts Table
CREATE TABLE Accounts (
    AccountID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT,
    AccountType VARCHAR(20),
    Balance DECIMAL(10,2),
    FOREIGN KEY (CustomerID) REFERENCES BankCustomers
                                        (CustomerID)
);
// Transaction Table
CREATE TABLE Transaction (
    TransactionID INT AUTO_INCREMENT PRIMARY KEY,
    AccountID INT,
    TransactionDate DATE,
    Amount DECIMAL(10,2),
    TransactionType VARCHAR(10),
    FOREIGN KEY (AccountID) REFERENCES Accounts
                                        (AccountID)
);
```

Step-2 : Insert data into BankCustomers, Accounts and Transaction table.

```
// BankCustomers Table
INSERT INTO BankCustomers (FirstName, LastName, DateOfBirth,
                           Email)
  VALUES
    ('John', 'Doe', '1980-05-15', 'john.doe@gmail.com'),
    ('Jane', 'Smith', '1990-07-22', 'jane.smith@gmail.com'),
    ('Alan', 'Brocon', '1985-09-30', 'alan.brocon@gmail.com');
// Accounts Table
INSERT INTO Accounts (CustomerID, AccountType, Balance)
  VALUES
      (1, 'Savings', 5000.00),
      (1, 'Checking', 2000.00),
      (2, 'Checking', 1500.00),
      (3, 'Savings', 4000.00);
// Transactions Table
INSERT INTO Transactions (AccountID, TransactionDate, Amount,
                          TransactionType)
  VALUES
      (1, '2024-08-01', 1000.00, 'Deposit'),
      (2, '2024-08-02', 500.00, 'Withdrawal'),
      (3, '2024-08-03', 200.00, 'Deposit'),
      (4, '2024-08-04', 800.00, 'Withdrawal');
```

Step-3 : Fetch the data

```
// BankCustomers Table
SELECT * FROM BankCustomers;
// Accounts Table
SELECT * FROM Accounts;
// Transactions Table
SELECT * FROM Transactions;
```

// Bank Customers Table

CustomerID	FirstName	LastName	DateofBirth	Email
1	John	Doe	1980-05-15	john.doe@gmail.com
2	Jane	Smith	1990-07-22	jane.smith@gmail.com
3	Alan	Brown	1985-09-30	alan.brown@gmail.

// Accounts Table

CustomerID	AccountID	AccountType	Balance
1	1	Savings	5000.00
1	2	Checking	2000.00
2	3	Checking	1500.00
3	4	Savings	4000.00

// Transactions Table

TransactionID	AccountID	TransactionDate	Amount	TransactionDate
1	1	'2024-08-01'	1000.00	Deposit
2	2	'2024-08-02'	500.00	Withdrawal
3	3	'2024-08-03'	200.00	Deposit
4	4	'2024-08-04'	800.00	Withdrawal

---

a). Creating Views :-

i). Creating a Simple view :-

```
CREATE VIEW CustomerAccountsSummary AS
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    a.AccountType,
    a.Balance
FROM
    BankCustomers c
INNER JOIN
    Accounts a ON c.CustomerID = a.CustomerID ;
```

ii). Creating a view with 'CHECK OPTION' :-

```
CREATE VIEW HighBalanceAccounts AS
SELECT
    AccountID,
    CustomerID,
    AccountType,
    Balance
FROM
    Accounts
WHERE
    Balance > 3000
WITH CHECK OPTION ;
```

b). Dropping / Selecting a view :-

i). Selecting from a view :-

```
SELECT * FROM CustomerAccountsSummary ;
```

```
SELECT * FROM HighBalanceAccounts ;
```

CustomerID	FirstName	LastName	AccountType	Balance
1	John	Doe	Savings	5000.00
1	John	Doe	Checking	2000.00
2	Jane	Smith	Checking	1500.00
3	Alan	Brown	Savings	4000.00

AccountID	CustomerID	AccountType	Balance
1	1	Savings	5000.00
3	1	Checking	4000.00

ii). Dropping a view :-

```
DROP VIEW CustomerAccountsSummary;
```

```
DROP VIEW HighBalanceAccounts;
```