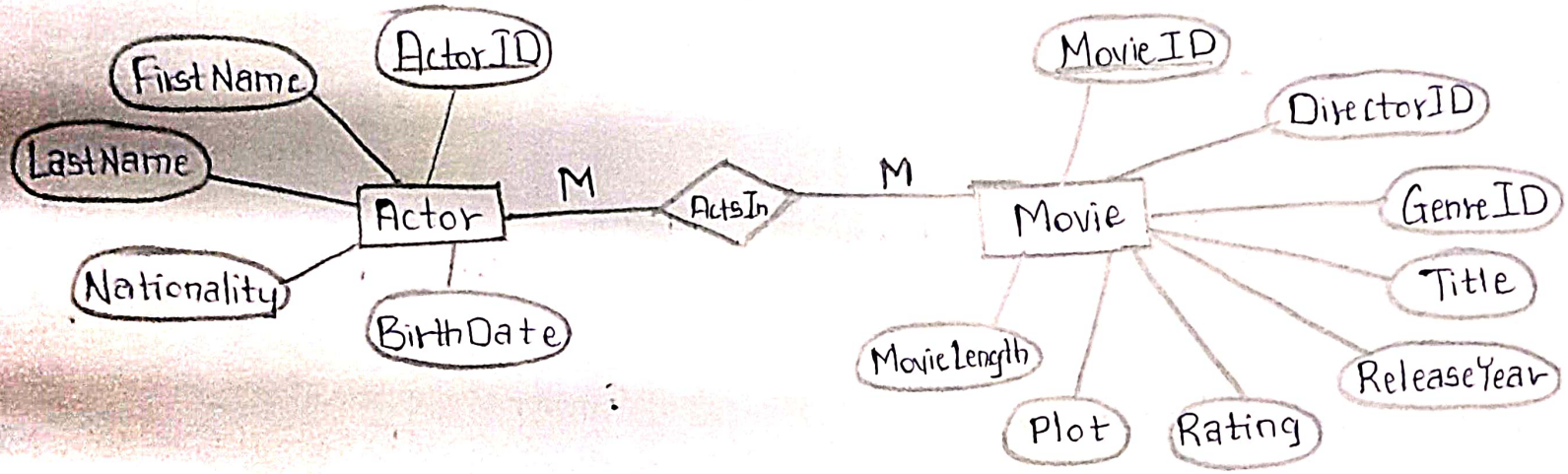
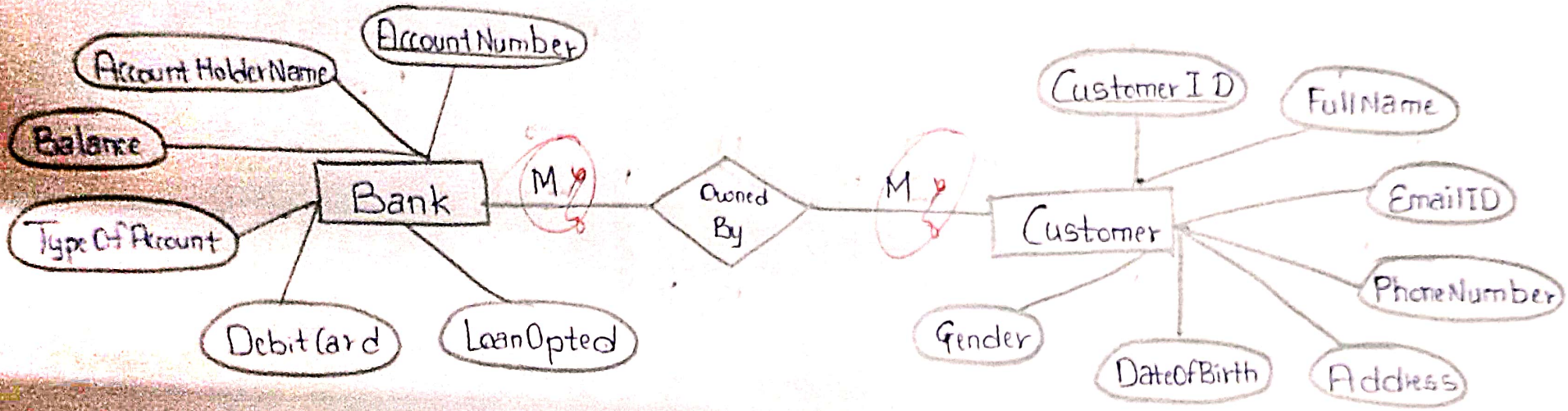
Abishekh-Villa- Roll No : 2471095

15-2-25 key?

1) 1.1 – Movie-Actor



12. Bank - Customer

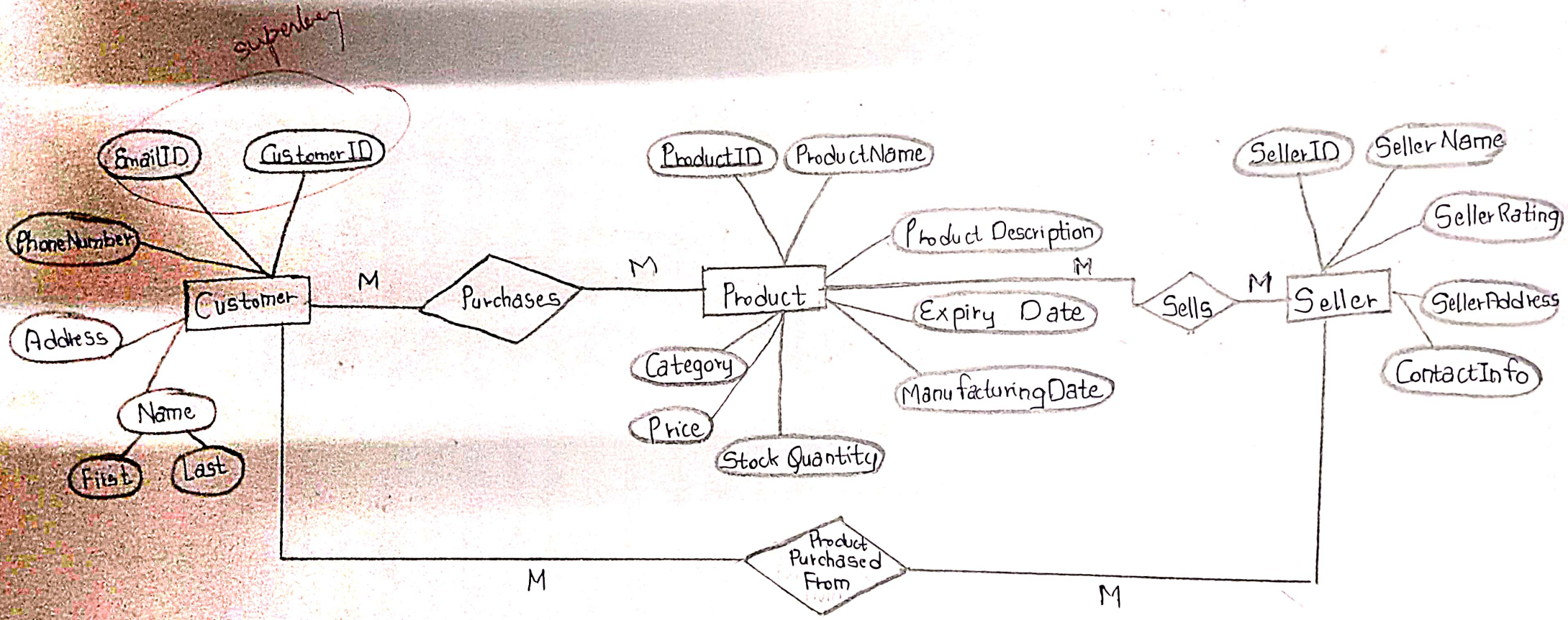


2.1. Product - Customer - Seller

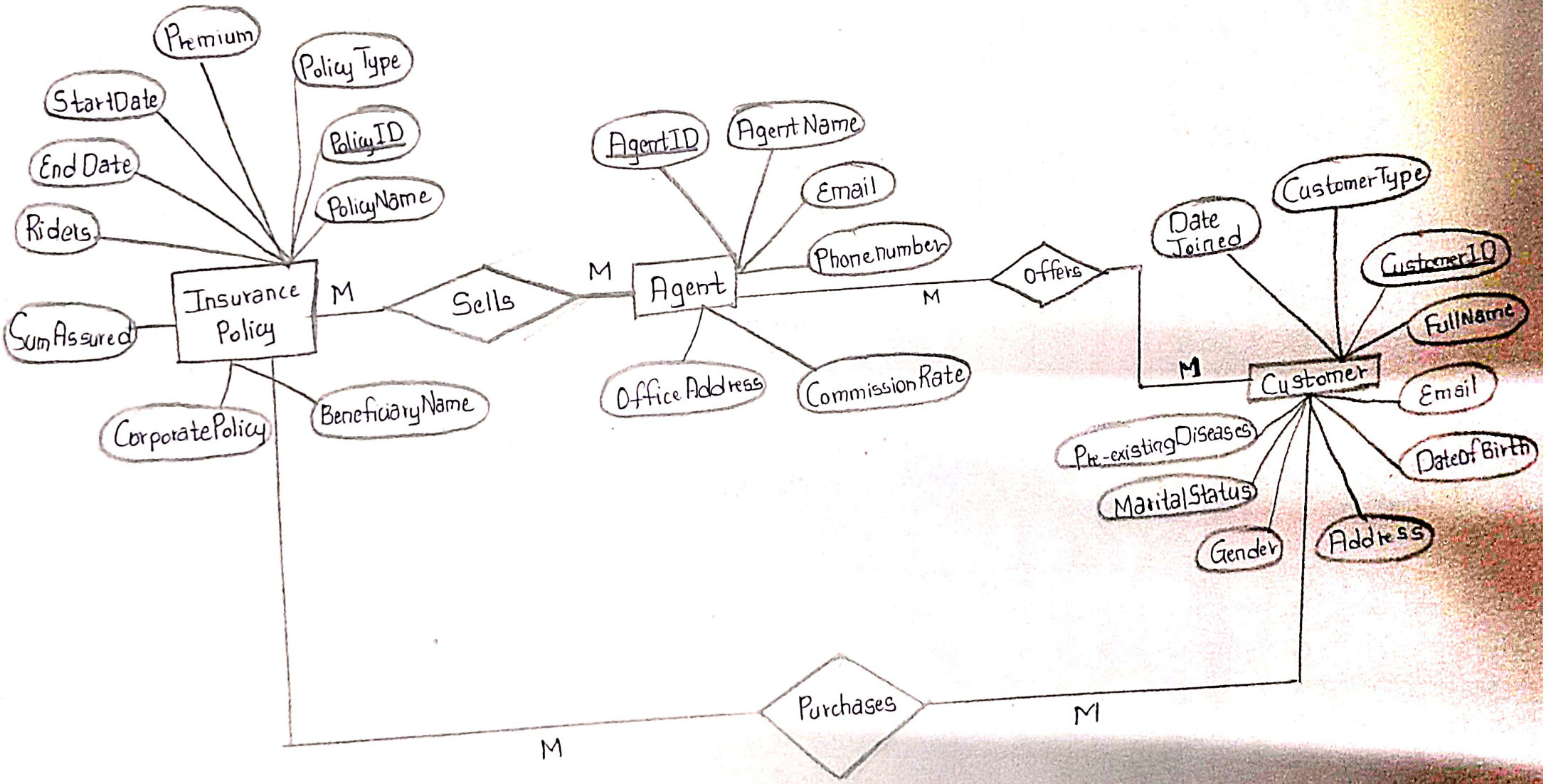
ASSIGNMENT No. 02

H Dishekh Uttad - Roll No. 24110141

15-2-2025



2) 1:2 – Policy – Agent – Customer



Roll No: 22F1095

Page No.
Date

Assignment No: 03

Working with relational algebra

Write relational algebra statements for relation Movie and Actor:

→

```
Create table Movie (Mid int not null primary key, Title
varchar (255) not null, RELEASE_year int check (RELEASE_year >=
1900), Budget decimal (12,2) check (Budget >= 0));

Create table Actor (Aid int not null primary key, Name varchar(255)
not null, Awards int check (Awards >= 0));

insert into Movie (Mid, Title, Release_year, Budget) values
(1, 'Lagaan', 2001, 25000000),
(2, 'Dangal', 2016, 30000000),
(3, '3 Idiots', 2009, 2500000),
(4, 'Kabhi Khushi Kabhie Gham', 2001, 60000000),
(5, 'Sholay', 1975, 2000000),
(6, 'Dilwale Dulhania Le Jayenge', 1995, 15000000),
(7, 'Kahaani', 2012, 15000000),
(8, 'PK', 2014, 70000000);

insert into Actor (Aid, Name, Awards) values
(1, 'Aamir Khan', 12),
(2, 'Shah Rukh Khan', 14),
(3, 'Salman Khan', 10),
(4, 'Deepika Padukone', 6),
(6, 'Kareena Kapoor', 9),
(7, 'Hrithik Roshan', 8),
```


PAGE No: 11

(8, 'Ranbir Kapoor', 5);

create table MovieActor (Mid int not null, Aid int not null, primary key (Mid, Aid), foreign key (Mid, Aid); foreign key (Mid) references Movie (Mid), foreign key (Aid) references Actor (Aid));

insert into MovieActor (Mid, Aid) values
(3,2),
(4,2),
(5,3),
(6,4),
(7,5),
(8,6),
(1,7),
(2,8);

3.1) Print all the movies released after year 2000.

→ select * from Movie where Release_year > 2000;

	Mid	Title	Release Year	Budget
1	1	Lagaan	2001	25000000.00
2	2	Dangal	2016	30000000.00
3	3	3 Idiots	2009	25000000.00
4	4	Kabhi Khushi Kabhie Gham	2001	60000000.00
5	7	Kahaani	2012	15000000.00
6	8	PK	2014	70000000.00

3.2) Print all the movies whose budget is minimum 5 crore.

→ select * from Movie where Budget >= 50000000;

	Mid	Title	Release_year	Budget
1	4	Kabhi Khushi Kabhie Ghom	2001	60000000.00
2	8	PK	2014	70000000.00

3.3) Print all the movies released between the year 1980 and 1990 with a budget less than 2 crore.

→ select * from Movie where Release_year BETWEEN 1980 AND 1990 AND Budget < 20000000;

Mid	Title	Release_year	Budget

* No records

3.4) Print all the actors without any awards

→ select * from Actor where Awards = 0;

Aid	Name	Awards

* No records

3.5) Print all the movies which are released before 1975 with a budget of 3 crore (30,000,000) or released after 2010 with a budget of 50 crore.

→ select * from Movie where (Release_year < 1975 AND Budget >= 30000000) OR (Release_year > 2010 AND Budget >= 50000000);

ROLL NO: 2471095

Page No.

Date

	Mid	Title	Release_year	Budget
1	8	PK	2014	70000000.00

Consider table student and teacher with name, age and address for both for atleast 5 tuples. Consider that a MSc student can teach a first year student. Print the result of union, intersection, difference and cross product of these two.

→

```
CREATE TABLE Student (Name varchar(255), Age INT,
Address varchar(255));

INSERT INTO Student (Name, Age, Address) values
('John Doe', 24, '123 Main st'),
('Alice Smith', 22, '456 Oak Ave'),
('Bob Johnson', 25, '789 Pine Rd'),
('Eve Davis', 23, '321 Maple Blvd'),
('Charlie Brown', 21, '654 Elm St');

CREATE TABLE Teacher (Name varchar(255), Age INT, Address varchar(255

INSERT INTO Teacher (Name, Age, Address) values
('Emma', 22, '654 Elm St'),
('Frank', 24, '111 Birch Dr'),
('Grace', 23, '222 Spruce Ct'),
('Hannah', 25, '333 Walnut Blvd'),
('Ian', 26, '444 Cherry Cir');
```

ROLL NO: 2471095

Page No.
Date

1) Print the result of UNION of 2 tables.

→
```
Select * from Student UNION
Select * from Teacher;
```

Name	Age	Address
Alice Smith	22	456 Oak Ave
Charlie	20	789 Pine Rd
David	21	321 Maple Blvd
Emma	22	654 Elm St
Frank	24	111 Birch Dr
Grace	23	222 Spruce Ct
Hannah	25	333 Walnut Blvd
Ian	26	444 Cherry Cir
John Doe	24	123 Main St

2) Print the result of INTERSECTION of 2 tables

→
```
select * from Student INTERSECTION
select * from Teacher;
```

Name	Age	Address
Emma	22	654 Elm St

3) Print the result of DIFFERENCE of 2 tables

→
```
select * from Student EXCEPT
select * from Teacher;
```

Name	Age	Address
John Doe	24	123 Main St
Alice Smith	22	456 Oak Ave
Charlie	20	789 Pine Rd
David	21	321 Maple Blvd

4) Print the CROSS PRODUCT of two tables

→ select s.name AS student_name, s.age AS student_age, s.address AS student_address, t.name AS teacher_name, t.age AS teacher_age, t.address AS teacher_address from Student s CROSS JOIN Teacher t;

student_name	student_age	student_address	teacher_name	teacher_age	teacher_add
John Doe	24	123 Main St	Emma	22	654 Elm St
John Doe	24	123 Main St	Frank	24	111 Birch Dr
John Doe	24	123 Main St	Grace	23	222 Spruce Ct
John Doe	24	123 Main St	Hannah	25	333 Walnut Blvd
John Doe	24	123 Main St	Ian	26	444 Cherry Cir
Alice Smith	22	456 Oak Ave	Emma	22	654 Elm St
Alice Smith	22	456 Oak Ave	Frank	24	111 Birch Dr
Alice Smith	22	456 Oak Ave	Grace	23	222 Spruce Ct
Alice Smith	22	456 Oak Ave	Hannah	25	333 Walnut Blvd
Alice Smith	22	456 Oak Ave	Ian	26	444 Cherry Cir
Charlie	20	789 Pine Rd	Emma	22	654 Elm St
Charlie	20	789 Pine Rd	Frank	24	111 Birch Dr
Charlie	20	789 Pine Rd	Grace	23	222 Spruce Ct
Charlie	20	789 Pine Rd	Hannah	25	333 Walnut Blvd
Charlie	20	789 Pine Rd	Ian	26	444 Cherry Cir
David	21	321 Maple Blvd	Emma	22	654 Elm St
David	21	321 Maple Blvd	Frank	24	111 Birch Dr
David	21	321 Maple Blvd	Grace	23	222 Spruce Ct
David	21	321 Maple Blvd	Hannah	25	333 Walnut Blvd
David	21	321 Maple Blvd	Ian	26	444 Cherry Cir
Emma	22	654 Elm St	Emma	22	654 Elm St
Emma	22	654 Elm St	Frank	24	111 Birch Dr
Emma	22	654 Elm St	Grace	23	222 Spruce St
Emma	22	654 Elm St	Hannah	25	333 Walnut Blvd
Emma	22	654 Elm St	Ian	26	444 Cherry Cir

Roll No : 2471095

Page No.

Date

Assignment No : 04

Working with SQL Server

Specify how to do the following using SQL server. Write proper SQL statements with necessary explanation.

4.1) Create a database containing two tables Movie and Actor with attributes given before.

→ Create database MovieActorDatabase;
Use MovieActorDatabase;
Create table Movie (Mid int not null primary key, Title varchar(255) not null, Release_year int check (Release_year >= 1990), Budget decimal(12,2) check (Budget >= '0'));
create table Actor (Aid int not null primary key, Name varchar(255) not null, Awards int check (Awards >= 0));

4.2) Specify the primary key for each table and add necessary constraints.

→ The primary key constraints are already specified in the tables. To ensure that the primary key values are unique and non-null, you can add the PRIMARY KEY constraint to the Mid and Aid columns in the Movie and Actor tables, respectively.
This ensures that each Mid and Aid will have unique and non-null values.

4.3) Add constraints for release year, budget, and awards ensuring no nulls.

→ Release year should be >= 1900 (using CHECK constraint).
Budget should be >= 0 (using CHECK constraint).
Awards should be >= 0 (using CHECK constraint).
Not null attributes on Title and Name to ensure no null values.

4.4) Insert a record / tuple into both tables

→ Insert into Movie (Mid, Title, Release_year, Budget) values
(1, 'Lagaan', 2001, 25000000),
(2, 'Dangal', 2016, 30000000),
(3, '3 Idiots', 2009, 25000000),
(4, 'Kabhi Khushi Kabhie Gham', 2001, 60000000),
(5, 'Sholay', 1975, 20000000),
(6, 'Dilwale Dulhania Le Jayenge', 1995, 15000000),
(7, 'Kahaani', 2012, 15000000),
(8, 'PK', 2014, 70000000);

Insert into Actor (Aid, Name, Awards) values
(1, 'Aamir Khan', 12),
(2, 'Shah Rukh Khan', 14),
(3, 'Salman Khan', 10),
(4, 'Deepika Padukone', 6),
(5, 'Priyanka Chopra', 7); (6, 'Kareena Kapoor', 9), (7, 'Hritik Roshan', 8), (8, 'Ranbir Kapoor', 5);

4.5) Update the budget field and change the spelling of an actor's name.

→ UPDATE Movie SET Budget = 28000000 WHERE Mid = 3;
UPDATE Actor SET Name = 'Shah Rukh' where Aid = 2;

4.6) Delete a record / tuple.

→ Delete from Movie where Mid = 8;

4.7) Print the contents of both tables

→ select * from Movie; select * from Actor;

Page No.

Date

Outputs :-

4.1) and 4.2) and 4.3)

Mid Title Release_year Budget

Aid Name Awards

4.4)

	Mid	Title	Release_year	Budget
1	1	Lagaan	2001	25000000.00
2	2	Dangal	2016	30000000.00
3	3	3 Idiots	2009	25000000.00
4	4	Kabhi Khushi Kabhie Gham	2001	60000000.00
5	5	Sholay	1975	20000000.00
6	6	Dilwale Dulhania le Jayenge	1995	15000000.00
7	7	Kahaani	2012	15000000.00
8	8	PK	2014	70000000.00

	Aid	Name	Awards
1	1	Aamir Khan	12
2	2	Shah Rukh Khan	14
3	3	Salman Khan	10
4	4	Deepika Padukone	6
5	5	Priyanka Chopra	7
6	6	Kareena Kapoor	9
7	7	Hritik Roshan	8
8	8	Ranbir Kapoor	5

4.5)

	Mid	Title	Release year	Budget
1	1	Lagaan	2001	25000000.00
2	2	Dangal	2016	30000000.00
3	3	3 Idiots	2009	28000000.00
4	4	Kabhi Khushi Kabhie Gham	2001	60000000.00
5	5	Sholay	1975	20000000.00
6	6	Dilwale Dulhania le Jayenge	1995	15000000.00
7	7	Kahaani	2012	15000000.00
8	8	PK	2014	70000000.00

	Aid	Name	Awards
1	1	Aamir Khan	12
2	2	Shah Rukh	14
3	3	Salman Khan	10
4	4	Deepika Padukone	6
5	5	Priyanka Chopra	7
6	6	Kareena Kapoor	9
7	7	Hritik Roshan	8
8	8	Ranbir Kapoor	5

4.6) and 4.7)

	Mid	Title	Release year	Budget
1	1	Lagaan	2001	25000000.00
2	2	Dangal	2016	30000000.00
3	3	3 Idiots	2009	28000000.00
4	4	Kabhi Khushi Kabhie Gham	2001	60000000.00
5	5	Sholay	1975	20000000.00
6	6	Dilwale Dulhania le Jayenge	1995	15000000.00
7	7	Kahaani	2012	15000000.00

	Aid	Name	Awards
1	1	Aamir Khan	12
2	2	Shah Rukh	14
3	3	Salman Khan	10
4	4	Deepika Padukone	6
5	5	Priyanka Chopra	7
6	7	Hritik Roshan	8
7	8	Ranbir Kapoor	5

55)

Roll No : 2471095

Page No.

Date

ASSIGNMENT No. : 05

Movie-Actor : Simple commands to print different attributes of a table.

Write SQL statements to do the following :

```
Create DATABASE MovieActor ; USE MovieActor ;
Create table Movie (Mid int not null primary key , Title varchar(255)
not null , Release_year int check (Release_year >= 1900) ,
Budget decimal (12,2) check (Budget >= 0));

create table Actor (Aid int not null primary key , Name
varchar(255) not null , Awards int check (Awards >=0));

insert into Movie (Mid , Title , Release_year , Budget) values
(1 , 'Lagaan' , 2001 , 25000000) ,
(2 , 'Dangal' , 2016 , 30000000) ,
(3 , '3 Idiots' , 2009 , 25000000) ,
(4 , 'Kabhi Khushi Kabhie Gham' , 2001 , 60000000) ,
(5 , 'Sholay' , 1975 , 20000000) ,
(6 , 'Dilwale Dulhania Le Jayenge' , 1995 , 15000000) ,
(7 , 'Kahaani' , 2012 , 15000000) ,
(8 , 'PK' , 2014 , 70000000) ;

insert into Actor (Aid , Name , Awards) values
(1 , 'Aamir Khan' , 12)
(2 , 'Salman Khan' , 10) ,
(3 , 'Shah Rukh Khan' , 14) ,
(4 , 'Deepika Padukone' , 6) ,
(5 , 'Priyanka Chopra' , 7) ,
(6 , 'Kareena Kapoor' , 9) ,
(7 , 'Hritik Roshan' , 8) , (8 , 'Ranbir Kapoor' , 5) ;
```

create table MovieActor (Mid int not null, Aid int not null, primary key (Aid, Mid), foreign key (Mid) references Movie (Mid), foreign key (Aid) references Actor (Aid));

insert into MovieActor (Mid, Aid) values (3,2), (4,2), (5,3), (6,4), (7,5), (8,6), (1,7), (2,8);

5.1) Print the contents of all the tables

→ Select * from Movie;
Select * from Actor;
Select * from MovieActor;

Mid	Title	Release_year	Budget
1	Lagaan	2001	25000000.00
2	Dangal	2016	30000000.00
3	3 Idiots	2009	25000000.00
4	Kabhi Khushi Kabhie Gham	2001	60000000.00
5	Sholay	1975	20000000.00
6	Dilwale Dulhania Le Jayenge	1995	15000000.00
7	Kahaani	2012	15000000.00
8	PK	2014	70000000.00

Aid	Name	Awards
1	Aamir Khan	12
2	Shah Rukh Khan	14
3	Salman Khan	10
4	Deepika Padukone	6
5	Priyanka Chopra	7
6	Kareena Kapoor	9
7	Hritik Roshan	8
8	Ranbir Kapoor	5

Roll No : 2471095

Page No.
Date

5.2) Print the titles of those movies which released before 2020.

→

	Title
1	Lagaan
2	Dangal
3	3 Idiots
4	Kabhi Khushi Kabhie Gham
5	Sholay
6	Dilwale Dulhaniya Le Jayenge
7	Kahaani
8	PK

5.3) Print the actors who have won atleast 3 awards

→ Select Name from Actor where Awards >= 3 ;

	Name
1	Aamir Khan
2	Shah Rukh Khan
3	Salman Khan
4	Deepika Padukone
5	Priyanka Chopra
6	Kareena Kapoor
7	Hritik Roshan
8	Ranbir Kapoor

5.4) Print the movies which are released after 1947 and with a budget of not more than 1cr.

→ No records found.

Page No.
Date

ASSIGNMENT No : 06

Car-Driver : Aggregation

Write SQL statements to do the following :

→ CREATE DATABASE CarDrivers ; USE CarDrivers ;

create table Car (Rno int not null primary key , Company varchar(255) not null , Model varchar (100) not null , Price decimal (12,2) check (Price >= 0));

create table Driver (Id int not null primary key , License_No varchar (50) unique not null , Name varchar (255) not null , Age int check (Age >= 18)) ;

insert into Car (Rno , Company , Model , Price) values
(101, 'Toyota', 'Camry', 25000) ,
(102, 'Ford', 'Mustang', 55000) ,
(103, 'Tesla', 'Model 3', 45000) ,
(104, 'Honda', 'Civic', 23000) ,
(105, 'Chevrolet', 'Malibu', 27000) ,
(106, 'BMW', 'X5', 70000) ,
(107, 'Audi', 'A4', 45000) ,
(108, 'Nissan', 'Altima', 23000) ;

insert into Driver (Id , License_No , Name , Age) values
(201, 'DL1234', 'Jane', 30) ,
(202, 'DL5678', 'John', 35) ,
(203, 'DL9910', 'Alex', 40) ,
(204, 'DL1122', 'Emily', 28) ,

(205, 'DL3344', 'Michael', 32),
(206, 'DL5566', 'Sophia', 27),
(207, 'DL7788', 'David', 33),
(208, 'DL9900', 'Olivia', 29);

Create table CarDriver (Rno int not null, Id int not null, primary key (Rno, Id), foreign key (Rno) references Car(Rno), foreign key (Id) references Driver(Id));

insert into CarDriver (Rno, Id) values
(101, 201), (102, 202), (103, 203), (104, 204), (105, 205), (106, 206), (107, 207), (108, 208);

6.1) Print the contents of all the tables

→
```
Select * from Car;
select * from Driver;
select * from CarDriver;
```

	Rno	Company	Model	Price
1	101	Toyota	Camry	25000.00
2	102	Ford	Mustang	55000.00
3	103	Tesla	Model 3	45000.00
4	104	Honda	Civic	23000.00
5	105	Chevrolet	Malibu	27000.00
6	106	BMW	X5	70000.00
7	107	Audi	A4	45000.00
8	108	Nissan	Altima	23000.00

Page No.
Date

	Id	License_No	Name	Age
1	201	DL1234	Jane	30
2	202	DL5678	John	35
3	203	DL8910	Alex	40
4	204	DL1122	Emily	28
5	205	DL33421	Michael	32
6	206	DL5566	Sophia	27
7	207	DL7788	David	33
8	208	DL9900	Olivia	29

	Rno	Id
1	101	201
2	102	202
3	103	203
4	104	204
5	105	205
6	106	206
7	107	207
8	108	208

6.2) Print how many cars are manufactured by BMW.

→ select COUNT(*) AS BMW_Cars from Car where Company = 'BMW';

	BMW_Cars
1	1

6.3) Print which company has manufactured the car with highest price.

→ select Company from Car where Price = (Select Max(Price) from Car);

	Company
1	BMW

6.4) Print the name of the youngest driver

```
SELECT Name from Driver WHERE
Age = (select MIN(Age) from Driver);
```

	Name
1	Sophia

6.5) Print the average age of the driver

```
SELECT AVG(Age) AS Average_age FROM DRIVER;
```

	Average_Age
1	31

Page No.
Date

ASSIGNMENT No : 07

Book - Author : wild card character, use of AND / OR.

Write SQL statements to do the following :

```
CREATE DATABASE BookAuthor ; USE BookAuthor ;

Create table Book (Bid int not null primary key , Title varchar(255)
not null , Publisher varchar(255) not null , Price decimal (10,2)
check (Price >= 0)) ;
Create table Author (Aid not null int primary key , Name varchar(255)
not null , Num_Books int check (Num_Books >= 0)) ;

insert into Book (Bid, Title , Publisher , Price) values
(1 , 'The Great Gatsby' , 'Scribner' , 10.99) ,
(2, '1984' , 'Secker & Warburg' , 12.99) ,
(3 , 'To Kill ' , ' J.B. Lippin ' , 9.99) ,
(4 , ' The Catcher' , ' Little , Brown & Co ' , 14.99) ,
(5 , 'Moby-Dick ' , 'Harper ' , 18.99) ,
(6 , ' Pride & Prejudice ' , ' T. Egerton ' , 11.49) ,
(7 , ' War and Peace ' , ' The Russian ' , 20.99) ,
(8 , ' The Hobbit ' , 'George Allen' , 15.49) ;

insert into Author (Aid , Name , Num_Books) values
(1 , ' F. Scott ' , 5) ,
(2 , ' George Orwell ' , 10) ,
(3 , ' Harper Lee ' , 2) ,
(4 , ' Herman Melville ' , 6) ,
(5 , ' J.D. Salinger ' , 3) ,
(6 , ' Jane Austen ' , 7) ,
(7 , ' Leo Tolstoy ' , 15) ,
(8 , ' J.R.R. Tolkien ' , 30) ;
```

Page No.
Date

Create table BookAuthor (Bid int not null, Aid int not null, primary key (Bid, Aid), foreign key (Bid) references Book(Bid), foreign key (Aid) references Author (Aid));

insert into BookAuthor (Bid, Aid) values (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8);

7.1) Print the contents of all the tables.

→
```
select * from Book;
select * from Author;
Select * from BookAuthor;
```

	Bid	Title	Publisher	Price
1	1	The Great Gatsby	Scribner	10.99
2	2	1984	Secker & Warburg	12.99
3	3	To Kill	J.B. Lippin	9.99
4	4	The Catcher	Little, Brow & Co	14.99
5	5	Moby-Dick	Harper	18.99
6	6	Pride and Prejudice	T. Egerton	11.49
7	7	War and Peace	The Russian	20.99
8	8	The Hobbit	George Allen	15.49

	Aid	Name	Num_Books
1	1	F. Scott	5
2	2	George Orwell	10
3	3	Harper Lee	2
4	4	J.D. Salinger	3
5	5	Herman Melville	6
6	6	Jane Austen	7
7	7	Leo Tolstoy	15
8	8	J.R.R. Tolkien	30

	Bid	Aid
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8

7.2) Print all the books starting with BW.

→ select Title from Book where Title LIKE 'BW%';

	Title
1	War and Peace

7.3) Print all the books published by 'Harper' such that the price is more than 15.

→ Select Title FROM Book where Publisher = 'Harper Everest Publications' AND Price > 15;

	Title
1	Moby-Dick

7.4) Print the author whose has written atleast 3 books

→ select Name from Author where Num_Books >=3;

	Name
1	F. Scott
2	George Orwell
3	J.D. Salinger
4	Herman Melville
5	Jane Austen
6	Leo Tolstoy

7	J.R.R. Tolkien

7.5) Print the author whose name starts with 'G' or has written 2 books.

→ Select Name from Author where Name LIKE 'G%' OR Num_Books = 2 ;

	Name
1	Harper Lee
2	George Orwell

Roll No : 2471095

Page No.
Date

Assignment No : 08

Product - Customer and Customer - Bank Accounts : order by / group by

Write SQL statements to do the following :

→
```
CREATE DATABASE CustomerProducts;
USE CustomerProducts;

Create table Customer (Cid int not null primary key,
Name varchar(255) not null, Address varchar(255) not null,
Bill_Amount decimal(10,2) check (Bill_Amount >= 0));

Create table Products (Pid int not null primary key,
Name varchar(255) not null, Quantity int check (Quantity >= 0),
Price_Per_Unit decimal check (Price_Per_Unit >= 0));

insert into Customer (Cid, Name, Address, Bill_Amount) values
(1, 'Alice', 'Elm St', 150.75),
(2, 'Bob', 'Oak St', 230.50),
(3, 'Charlie', 'Pine St', 120.00),
(4, 'Diana', 'Maple St', 175.30),
(5, 'Eve', 'Birch St', 99.99),
(6, 'Frank', 'Cedar St', 210.10),
(7, 'Grace', 'Willow St', 300.00),
(8, 'Henry', 'Fir St', 125.80);

insert into Products (Pid, Name, Quantity, Price_Per_Unit)
values (1, 'Laptop', 10, 550.00),
(2, 'Smartphone', 25, 300.00),
(3, 'Headphones', 30, 50.00),
```

(4, 'Tablet', 15, 350.00),
(5, 'Keyboard', 100, 20.00),
(6, 'Mouse', 150, 15.00),
(7, 'Monitor', 30, 200.00),
(8, 'Speaker', 40, 75.00);

create table CustomerProducts (Cid int not null, Pid not null int, primary key (Cid, Pid), foreign key (Cid) references Customer (Pid), foreign key (Pid) references Products (Pid));

insert into CustomerProducts (Cid, Pid) values (1,1), (1,3), (2,2), (2,4), (3,6), (4,5), (5,7), (6,8), (7,1), (8,3);

8-1) Print the contents of all the tables

→ select * from Customer;
select * from Products;
select * from CustomerProducts;

	Cid	Name	Address	Bill_Amount
1	1	Alice	Elm St	150.75
2	2	Bob	Oak St	230.50
3	3	Charlie	Pine St	120.00
4	4	Diana	Maple St	175.30
5	5	Eve	Birch St	99.99
6	6	Frank	Cedar St	210.10
7	7	Grace	Willow St	300.00
8	8	Henry	Fir St	125.80

	Pid	Name	Quantity	Price_Per_Unit
1	1	Laptop	10	550
2	2	Smartphone	25	300
3	3	Headphones	50	50
4	4	Tablet	15	350
5	5	Keyboard	100	20
6	6	Mouse	150	15
7	7	Monitor	30	200
8	8	Speaker	40	75

	Cid	Pid
1	1	1
2	1	3
3	2	2
4	2	4
5	3	6
6	4	5
7	5	7
8	6	8
9	7	1
10	8	3

3.2) Print names of all the customers area wise as per their address

→ select Name, Address from Customer ORDER BY Address;

	Name	Address
1	~~Alice~~ Eve	Birch St
2	Frank	Cedar St
3	Alice	Elm St

4	Henry	Fir St
5	Diana	Maple St
6	Bob	Oak St
7	Charlie	Pine St
8	Grace	Willow St

8.3) Print names of all the products price-wise whose quantity is more than 5. Use order by / group by as per the need where essential.

→ select Name, Quantity, Price_Per_Unit from Products where Quantity > 5 order by Price_Per_Unit DESC;

	Name	Quantity	Price_Per_Unit
1	Laptop	10	550
2	Tablet	15	350
3	Smartphone	25	300
4	Monitor	30	200
5	Speaker	40	75
6	Headphones	50	50
7	Keyboard	100	20
8	Mouse	150	15

→
```
CREATE DATABASE CustomerBankAccounts ;
USE CustomerBankAccounts ;

create table BankCustomer ( Cid int primary key , Name
varchar (255) not null , Address varchar (255) not null ,
Nominee varchar (255)) ;

create table BankAccount ( Anum int primary key , Type
varchar (50) check ( Type IN ('Savings' , 'Current' , 'Fixed Deposit'))
Min_Balance decimal (10,2) check (Min_Balance >= 0)) ;

insert into BankCustomer (Cid , Name , Address , Nominee) values
(1 , 'Amit Sharma' , 'Pune' , 'Rekha Sharma') ,
(2 , 'Priya Mehta' , 'Mumbai' , 'Rohan Mehta') ,
(3 , 'Rahul Verma' , 'Delhi' , 'Sunita Verma') ,
(4 , 'Neha Gupta' , 'Nagpur' , 'Aarav Gupta') ,
(5 , 'Vikram Singh' , 'Kolkata' , 'Ravi Singh') ,
(6 , 'Sneha Patil' , 'Pune' , 'Karan Patil') ,
(7 , 'Manoj Deshmukh' , 'Mumbai' , 'Nisha Deshmukh') ,
(8 , 'Ritu Malhotra' , 'Chennai' , 'Anil Malhotra') ;

insert into BankAccount (Anum , Type , Min_Balance) values
(101 , 'Savings' , 1000.00) ,
(102 , 'Current' , 5000.00) ,
(103 , 'Fixed Deposit' , 25000.00) ,
(104 , 'Savings' , 2000.00) ,
(105 , 'Current' , 1000.00) ,
(106 , 'Fixed Deposit' , 50000.00) ,
(107 , 'Savings' , 1500.00) ,
(108 , 'Current' , 7500.00) ;
```

Page No.
Date

```
create table CustomerBankAccount ( Cid INT , Anum INT ,
primary key (Cid, Anum) , foreign key (Cid) references
BankCustomer (Cid) , foreign key (Anum) references
BankAccount (Anum)) ;

insert into CustomerBankAccount (Cid , Anum) values
(1,101) , ( 2,102) , (3,103) , (4,104) , (5,105) , (6,106),
(7,107) , (8,108) ;
```

8.1) Print the contents of all the tables

```
select * from CustomerAccount ;
select * from BankAccount ;
select * from CustomerBankAccount ;
```

	Cid	Name	Address	Nominee
1	~~Amit S~~ 1	Amit Sharma	Pune	Rekha Sharma
2	2	Priya Mehta	Mumbai	Rohan Mehta
3	3	Rahul Verma	Delhi	Sunita Verma
4	4	Neha Gupta	Nagpur	Aarav Gupta
5	5	Vikram Singh	Kolkata	Ravi Singh
6	6	Sneha Patil	Pune	Karan Patil
7	7	Manoj Deshmukh	Mumbai	Nisha Deshmukh
8	8	Ritu Malhotra	Chennai	Anil Malhotra

	Anum	Type	Min_Balance
1	101	Savings	1000.00
2	102	Current	5000.00
3	103	Fixed Deposit	25000.00
4	104	Savings	2000.00

5	105	Current	10000.00
6	106	Fixed Deposit	50000.00
7	107	Savings	1500.00
8	108	Current	7500.00

	Cid	Anum
1	1	101
2	2	102
3	3	103
4	4	104
5	5	105
6	6	106
7	7	107
8	8	108

8.2) Print all the customers whose names start with R and stay in Delhi along with their nominee.

→ Select Name, Nominee from BankCustomer where Name LIKE 'R%' AND Address LIKE '%DELHI%';

	Name	Nominee
1	Rahul Verma	Sunita Verma

.3) Print the account numbers as per type along with the minimum balance. Use of order by / group by / where as per the need where essential.

→ select Type, Anum, Min_Balance from BankAccount order by Type ;

	Type	Anum	Min_Balance
1	Current	102	5000.00
2	Current	105	10000.00
3	Current	108	7500.00
4	Fixed Deposit	106	50000.00
5	Fixed Deposit	103	25000.00
6	Savings	104	2000.00
7	Savings	107	1500.00
8	Savings	101	1000.00

Roll No: 2471095

Page No.
Date

Assignment No : 09

Singers - Songs : Normalization

Write SQL statements to do the following :

```
CREATE DATABASE SingersSongs ;
USE SingersSongs ;

create table Singer (Sid int primary key , Name varchar(255) not null , Gender varchar(10) check (Gender IN ('Male','Female','Other')) , No_Of_Songs int check (No_Of_Songs >= 0)) ;

create table Songs (Snid int primary key , Lyricist varchar(255) not null , Composer varchar(255) not null , Release_Year int check (Release_Year >= 1900)) ;

insert into Singe (Sid , Name , Gender , No_Of_Songs) values
(1 , 'Arijit Singh' , 'Male' , 200) ,
(2 , 'Shreya Ghoshal' , 'Female' , 300) ,
(3 , 'Lata Mangeshkar' , 'Female' , 8000) ,
(4 , 'Kishore Kumar' , 'Male' , 800) ,
(5 , 'Neha Kakkar' , 'Female' , 150) ,
(6 , 'Sonu Nigam' , 'Male' , 600) ,
(7 , 'Asha Bhosle' , 'Female' , 700) ,
(8 , 'Mohit Chauhan' , 'Male' , 250) ;

insert into Songs (Snid , Lyricist , Composer , Release_Year)
values (1 , 'Kumaar' , 'Pritam' , 2014) ,
(2 , 'Javed' , 'Shankar' , 2001) ,
(3 , 'Gulzar' , 'Rahman' , 2007) ,
(4 , 'Sameer' , 'Nadeem' , 1990) ;
```

(5, 'Ravindra', 'Jain', 1980),
(6, 'Kumaar', 'Himesh', 2012),
(7, 'Shailendra', 'Burman', 1959),
(8, 'Irshad', 'Pritam', 2014);

create table SingerSongs (Sid, Snid) values
(1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7),
(8,8);

9.1) Print the content of all the tables

→
```
select * from Singer;
select * from Songs;
select * from SingerSongs;
```

	Sid	Name	Gender	No_Of_Songs
1	1	Arijit Singh	Male	200
2	2	Shreya Ghoshal	Female	300
3	3	Lata Mangeshkar	Female	1000
4	4	Kishore Kumar	Male	800
5	5	Neha Kakkar	Female	150
6	6	Sonu Nigam	Male	600
7	7	Asha Bhosle	Female	700
8	8	Mohit Chauhan	Male	250

	Snid	Lyricist	Composer	Release_Year
1	1	Kumaar	Pritam	2014
2	2	Javed	Shankar	2001
3	3	Gulzar	Rahman	2007
4	4	Sameer	Nadeem	1990

5	5	Ravindra	Jain	1980
6	6	Kumaar	Himesh	2012
7	7	Shailendra	Burman	1959
8	8	Irshad	Pritam	2014

	Sid	Snid
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8

9.2) Print the names of all the singers gender-wise.

→ Select Gender, Name from Singer order by Gender;

	Gender	Name
1	Female	Shreya Ghoshal
2	Female	Lata Mangeshkar
3	Female	Neha Kakkar
4	Female	Asha Bhosle
5	Male	Mohit Chauhan
6	Male	Arijit Singh
7	Male	Sonu Nigam
8	Male	Kishore Kumar

9.3) Print names of all the composers in the ascending order of release year.

→ select Composer, Releaser_Year from Songs order by Release_Year ASC ;

	Composer	Release_Year
1	Burman	1959
2	Jain	1980
3	Nadeem	1990
4	Shankar	2001
5	Rahman	2007
6	Himesh	2012
7	Pritam	2014
8	Pritam	2014

9.4) Print all the songs year-wise whose lyricist is Gulzar and Composer is Rahman. Use order by /group by as per the need where essential.

→ select * from Songs where Lyricist = 'Gulzar' AND Composer = 'Rahman' order by Release_Year;

	Sbid	Lyricist	Composer	Release_Year
1	3	Gulzar	Rahman	2007

Roll No : 2471095

Page No.
Date

ASSIGNMENT No : 10

Doctor - Patient : where
Indexing , user creation

Write SQL statements to do the following :

→ CREATE DATABASE DoctorPatients ;

USE DoctorPatients ;

create table Doctor (Did int primary key , Name varchar(255) Not null , Specialization varchar (265) Not null , Registration_Number varchar (50) unique Not null) ;

create table Patient (Pid int primary key , Name varchar (255) not null , Fees decimal (10,2) CHECK (Fees >= 0)) ;

insert into Doctor (Did , Specialization , Registration_Number) values (1, 'Dr Amit Sharma' , 'Cardiologist' , 'REG12345') ,
(2 , 'Dr. Priya Mehta' , 'Neurologist' , 'REG67890') ,
(3 , 'Dr. Rahul Verma' , 'Dermatologist' , 'REG11223') ,
(4 , 'Dr. Neha Gupta' , 'Orthopedic' , 'REG44556') ,
(5 , 'Dr. Vikram Singh' , 'Pediatrician' , 'REG77889') ,
(6 , 'Dr. Sneha Patil' , 'Gynecologist' , 'REG99001') ,
(7 , 'Dr. Manoj Deshmukh' , 'ENT Specialist' , 'REG22334') ,
(8 , 'Dr. Ritu Malhotra' , 'General Physician' , 'REG55667') ;

insert into Patient (Pid , Name , Fees) values
(101 , 'Rohan Kumar' , 500.00) ,
(102 , 'Ananya Singh' , 700.00) ,
(103 , 'Karan Patel' , 400.00) ,
(104 , 'Sonya Nath' , 500.00) ;

```
(105, 'Meera Joshi', 900.00),
(106, 'Vikash Yadav', 600.00),
(107, 'Amitabh Roy', 1000.00),
(108, 'Priyanka Das', 750.00);

create table DoctorPatient ( Did int, Pid int,
Visit_Date not null, primary key (Did, Pid, Visit_Date),
foreign key (Did) references Doctor (Did),
foreign key (Pid) references Patient (Pid));

insert into DoctorPatient (Did, Pid, Visit_Date) values
(1, 101, '2025-02-10'),
(2, 102, '2025-02-11'),
(3, 103, '2025-02-12'),
(4, 104, '2025-02-13'),
(5, 105, '2025-02-14'),
(6, 106, '2025-02-15'),
(7, 107, '2025-02-16'),
(8, 108, '2025-02-17');
```

10-1) Print the contents of all the tables

→

```
Select * from Doctor;
Select * from Patient;
Select * from DoctorPatient;
```

	Did	Name	Specialization	Registration_Number
1	1	Dr. Amit Sharma	Cardiologist	REG12345
2	2	Dr. Priya Mehta	Neurologist	REG67890
3	3	Dr. Rahul Verma	Dermatologist	REG11223

Roll No: 2471095

Page No.
Date

4	4	Dr. Neha Gupta	Orthopedic	REG44556
5	5	Dr. Vikram Singh	Pediatrician	REG77889
6	6	Dr. Sneha Patil	Gynecologist	REG9901
7	7	Dr. Manoj Deshmukh	ENT Specialist	REG22334
8	8	Dr. Ritu Malhotra	General Physician	REG55667

	Pid	Name	Fees
1	101	Rohan Kumar	500.00
2	102	Ananya Singh	700.00
3	103	Karan Patel	400.00
4	104	Sanya Arora	800.00
5	105	Vikash Yadav	600.00
6	106	Meera Joshi	900.00
7	107	Amitabh Roy	1000.00
8	108	Priyanka Das	750.00

	Did	Pid	Visit_Date
1	1	101	2025-02-10
2	2	102	2025-02-11
3	3	103	2025-02-12
4	4	104	2025-02-13
5	5	105	2025-02-14
6	6	106	2025-02-15
7	7	107	2025-02-16
8	8	108	2025-02-17

9-2) Print names of all the surgeons with registration number.

→ select Name, Registration_Number from Doctor;

	Name	Registration_Number
1	Dr. Amit Sharma	REG12345
2	Dr. Priya Mehta	REG67890
3	Dr. Rahul Verma	REG 11223
4	Dr. Neha Gupta	REG 44556
5	Dr. Vikram Singh	REG 77889
6	Dr. Sneha Patil	REG 99001
7	Dr. Manoj Deshmukh	REG 22334
8	Dr. Ritu Malhotra	REG 55667

9-3) Print names of the patient in the ascending order of fees.

→ select Name, Fees from Patient order by Fees ASC;

	Name	Fees
1	Karan Patel	400.00
2	Rohan Kumar	500.00
3	Vikash Yadav	600.00
4	Ananya Singh	700.00
5	Priyanka Das	750.00
6	Sanya Arora	800.00
7	Meera Joshi	900.00
8	Amitabh Roy	1000.00

1) What is the use of a relationship table ?

→ A relationship table (also called a junction table or associative table) is used in databases to establish and maintain many-to-many relationships between two tables.

Example : A Singer can sing multiple Songs.
A Song can be sung by multiple Singers.

To handle this, we create a relationship table SingerSongs:

```
CREATE TABLE SingerSongs ( Sid INT, Snid INT,
PRIMARY KEY (Sid, Snid), FOREIGN KEY (Sid) REFERENCES
Singer (Sid), FOREIGN KEY (Snid) REFERENCES Songs (Snid));
```

This table connects singers and songs without duplicating data.

- Benefits of a relationship table :
  - Avoids data redundancy.
  - Maintains data integrity.
  - Supports complex queries efficiently.

2) What is Normalization?

→ Normalization is the process of organizing a database to reduce redundancy and improve data integrity by dividing tables into smaller, structured tables.

• Normalization Forms:

① 1NF (First Normal Form) – Ensure all columns contain atomic (indivisible) values.

② 2NF (Second Normal Form) – Remove partial dependencies; every non-key column should depend on the entire primary key.

③ 3NF (Third Normal Form) – Remove transitive dependencies; non-key columns should depend only on the primary key.

④ BCNF (Boyce-Codd Normal Form) – A stronger version of 3NF to handle anomalies in composite keys.

• Example of Normalization: Unnormalized Table (Repeating Data):

Order ID	Customer	Product	Price
101	Alice	Laptop	50000
101	Alice	Mouse	1000

Normalized Tables (Using 2NF):

① Customer Table

Customer ID	Customer
1	Alice

② Order Table

OrderID	CustomerID
101	1

③ Product Table

ProductID	Product	Price
201	Laptop	50000
202	Mouse	1000

④ Order_Product Table (Relationship Table)

OrderID	ProductID
101	201
101	202

This structure eliminates redundant data, making updates and queries more efficient.

Roll No: 2471095

3) Explain Indexing

→ Indexing is a database optimization technique used to speed up search queries by creating a data structure that allows quick lookups.

• Types of Indexes :

① Primary Index - Automatically created on a primary key.
② Unique Index - Ensures unique values in a column.
③ Clustered Index - Stores table rows in a sorted order.
④ Non-clustered Index - Stores pointers to the actual data.
⑤ Composite Index - Created on multiple columns.

• Example : CREATE INDEX idx_customer_name ON BankCustomer (Name);

This index speeds up searches on the Name column.

• Advantages :

- Faster searches (SELECT queries).
- Improved sorting performance (ORDER BY).
- Efficient filtering using WHERE conditions.

• Disadvantages :

- Indexes consume extra storage.
- Slower INSERT, UPDATE and DELETE operations due to the index updates.

4) Explain How to Create a Database User

→ To create a database user, we use the CREATE USER command in SQL.

- Example (MySQL / PostgreSQL):

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password';
```

This creates a user named new_user with a password.

- Assign Privileges :

```
GRANT ALL PRIVILEGES ON my_database.* TO 'new_user'@'localhost';
```

This allows new_user to access and modify my_database.

- Viewing Users :

```
Select User From mysql.user;
```

This display all users in MySQL.

- Deleting a User :

```
DROP user 'new_user'@'localhost';
```

5) Explain : GRANT and REVOKE

→ GRANT Command – Gives specific priveleges to a user.
Example :
GRANT SELECT , INSERT , UPDATE ON BankCustomer TO 'new_user' @ 'localhost';
This allows new_user to read , insert , and update records in the BankCustomer table.

REVOKE Command – Removes priveleges from a user.
Example :
REVOKE INSERT , UPDATE ON BankCustomer FROM 'new_user' @ 'localhost' ;
This removes INSERT and UPDATE privileges from new_user.

• Common Privileges :
- SELECT – Read data.
- INSERT – Add data.
- UPDATE – Modify data.
- DELETE – Remove data
- ALL PRIVILEGES – Full access.