**Assignment 3**

**Name** : Jeevan Sai Badana

**Mail** : [jeevansai100@gmail.com](mailto:jeevansai100@gmail.com)

**Date** : 19-01-2024

Topic : SQL

|  |
| --- |
|  |
| **SQL\_D document** |

* First we can create database named as pet\_adoption

create database pet\_adoption;

* And then we can use this database with following query

use pet\_adoption;

* Next we can create tables with columns and proper datatypes

And we also show is our tables successfully created or not

CREATE TABLE animals

(id VARCHAR(100) Not null,

name VARCHAR(200),

breed VARCHAR(200),

color VARCHAR(200),

gender VARCHAR(200),

status TINYINT(1) NOT null

);

CREATE TABLE adoptions (

animal\_id varchar(100) NOT NULL,

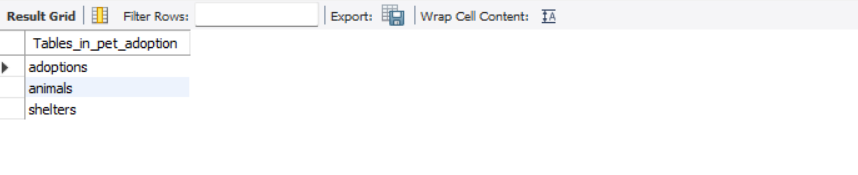
name varchar(200),

contact varchar(200),

date TIMESTAMP

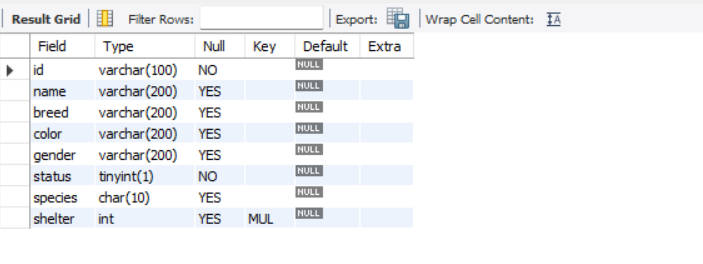
);

SHOW TABLES;



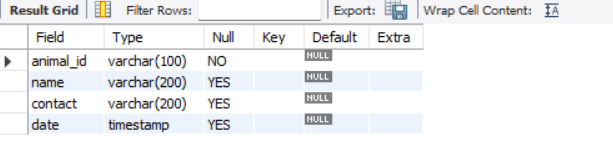
* Now we can see the columns also by using show

SHOW COLUMNS FROM animals;



* Likewise we show columns for other tables adoptions

SHOW COLUMNS FROM adoptions;



* Now we can insert values in animals table and see the whole table by using select statement

INSERT INTO animals (id, name, breed, color, gender, status)

VALUES

('89354034-20d9-4c3d-8195-3294bfd9dbc5', 'Bellyflop', 'Beagle', 'Brown', 'Male', 0),

('ae91cf1c-f972-4213-8160-6c04d935699c', 'Snowy', 'Husky', 'White', 'Female', 0),

('37df3388-b0f4-4f0d-b6ef-0d840923a4d8', 'Princess', 'Pomeranian', 'Black', 'Female', 0),

('94545432-d27a-4ac8-ab7c-3827d7535f3', 'Cricket', 'Chihuahua', 'Brown', 'Male', 0),

('a1e7a7fc-b429-41ec-9924-8bb39dd397c8', 'Princess', 'Poodle', 'Purple', 'Female', 0),

('5138ed53-2ab2-400b-973c-91186f8c673d', 'Spot', 'Dalmation', 'Black and White', 'Male', 0);

INSERT INTO animals (id, name, species, breed, color, gender, status)

VALUES ('11d6fa07-449f-4053-a7cb-ae4ec8570f3f', 'Meowmix', 'Cat', 'Munchkin', 'Yellow', 'Female', 0);

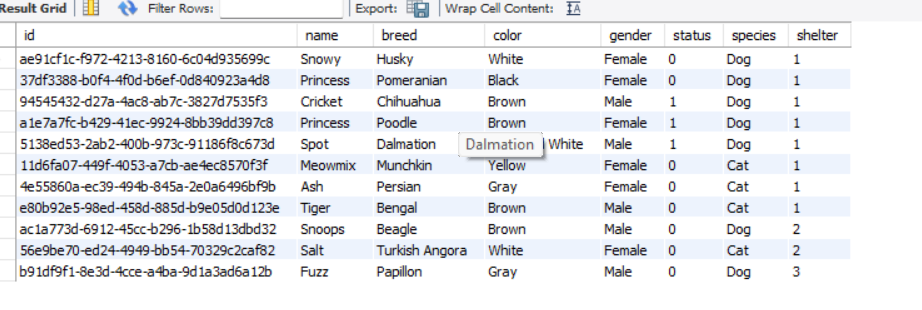
INSERT INTO animals (id, name, species, breed, color, gender, status)

VALUES ('4e55860a-ec39-494b-845a-2e0a6496bf9b', 'Ash', 'Cat', 'Persian', 'Gray', 'Female', 0);

INSERT INTO animals (id, name, species, breed, color, gender, status)

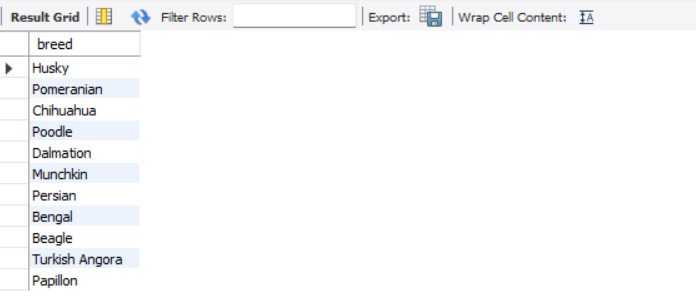
VALUES ('e80b92e5-98ed-458d-885d-b9e05d0d123e', 'Tiger', 'Cat', 'Bengal', 'Brown', 'Male', 0);

SELECT \* FROM animals;



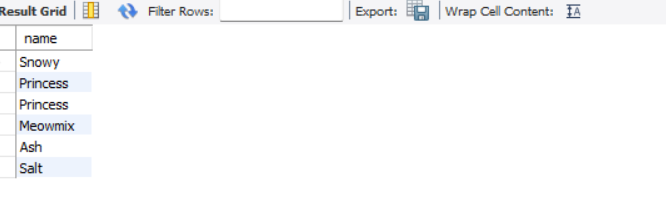
* We also retrieve specific column instead of getting everything

SELECT breed FROM animals;



* We also retrieve data on specific conditions by using where clause

SELECT name FROM animals WHERE gender = 'Female';



* We can update values using update statement

UPDATE animals SET status = 1 WHERE id = 'a1e7a7fc-b429-41ec-9924-8bb39dd397c8';



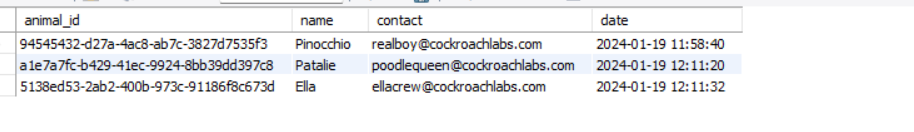
* We can delete the data by using delete statement

DELETE FROM animals WHERE id = '89354034-20d9-4c3d-8195-3294bfd9dbc5';



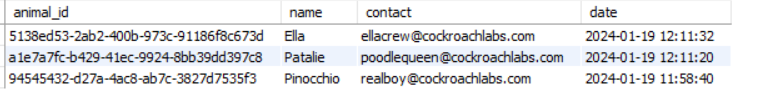
* We also insert values to adoption table and retrieve everything by using select

SELECT \* FROM adoptions;



* We also retrieve the data in certain order like accending or decending order

SELECT \* FROM adoptions ORDER BY date DESC;



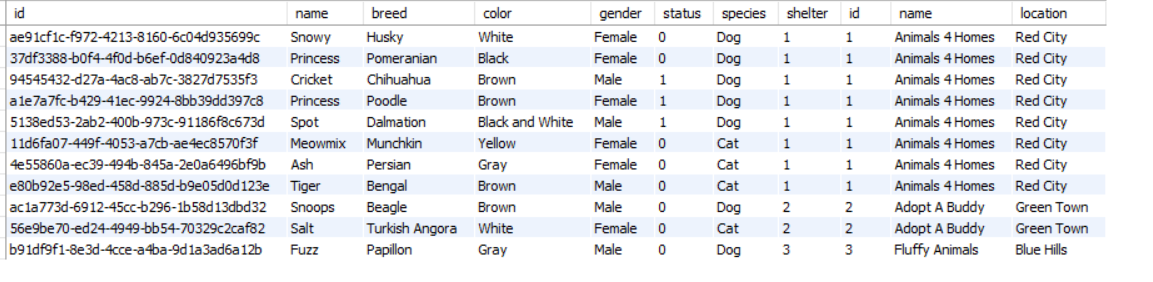
* We can add columns using alter statement

ALTER TABLE animals ADD COLUMN species char(10);



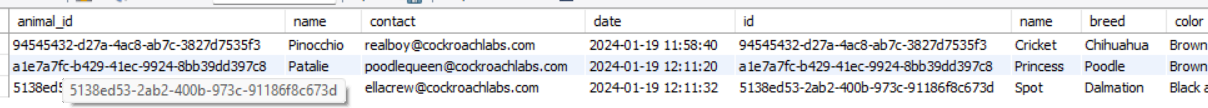
* Now we can perform some join operations by joining two tables

SELECT \* FROM animals JOIN shelters ON animals.shelter = shelters.id;



* We can also apply conditions for the join statements using where clause

SELECT \* FROM adoptions JOIN animals ON adoptions.animal\_id = animals.id WHERE animals.shelter = 1;



* Now we can create another database named as xstream\_db

CREATE DATABASE xstream\_db;

use xstream\_db;

* Now we create employee table and insert some values and retrieve everything using select

CREATE TABLE employee (emp\_id INT, emp\_name VARCHAR(255),

emp\_city VARCHAR(255),

emp\_country VARCHAR(255),

PRIMARY KEY (emp\_id));

INSERT INTO employee VALUES (101, 'Utkarsh Tripathi', 'Varanasi', 'India'),

(102, 'Abhinav Singh', 'Varanasi', 'India'),

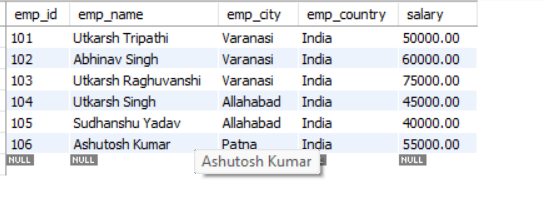
(103, 'Utkarsh Raghuvanshi', 'Varanasi', 'India'),

(104, 'Utkarsh Singh', 'Allahabad', 'India'),

(105, 'Sudhanshu Yadav', 'Allahabad', 'India'),

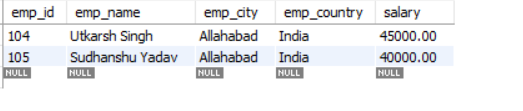
(106, 'Ashutosh Kumar', 'Patna', 'India');

SELECT \* FROM employee;



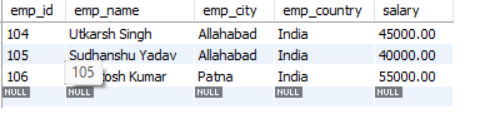
* Now we can perform different operations through different operators
* AND operator

SELECT \* FROM employee WHERE emp\_city = 'Allahabad' AND emp\_country = 'India';



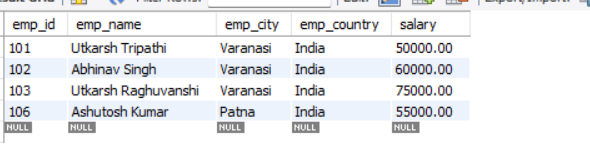
* IN operator

SELECT \* FROM employee WHERE emp\_city IN ('Allahabad', 'Patna');



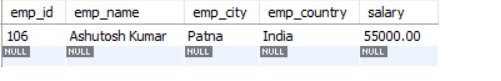
* NOT LIKE operator

SELECT \* FROM employee WHERE emp\_city NOT LIKE 'A%';



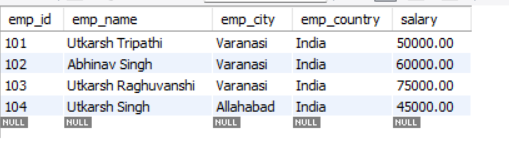
* LIKE operator

SELECT \* FROM employee WHERE emp\_city LIKE 'P%';



* BETWEEN operator

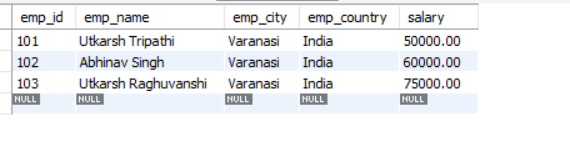
SELECT \* FROM employee WHERE emp\_id BETWEEN 101 AND 104;



* retrieve employee data who live in varanasi using ANY

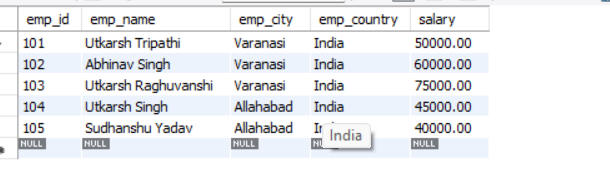
SELECT \* FROM employee WHERE emp\_id = ANY

(SELECT emp\_id FROM employee WHERE emp\_city = 'Varanasi');



* SELECT emp\_name FROM employee WHERE EXISTS

(SELECT emp\_id FROM employee WHERE emp\_city = 'Patna');



* Now we create students table and insert values into students table and retrieve everything

CREATE TABLE students (

ROLL\_NO INT,

NAME VARCHAR(50),

ADDRESS VARCHAR(100),

PHONE VARCHAR(20),

AGE INT

);

INSERT INTO students (ROLL\_NO, NAME, ADDRESS, PHONE, AGE)

VALUES

(1, 'Shubham Kumar', '123 Main Street, Bangalore', '9876543210', 23),

(2, 'Shreya Gupta', '456 Park Road, Mumbai', '9876543211', 23),

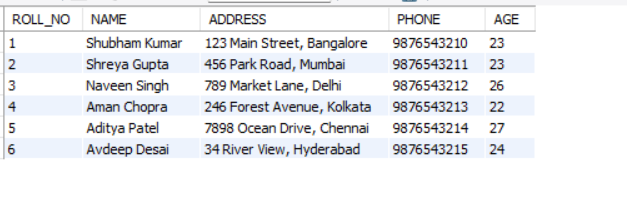
(3, 'Naveen Singh', '789 Market Lane, Delhi', '9876543212', 26),

(4, 'Aman Chopra', '246 Forest Avenue, Kolkata', '9876543213', 22),

(5, 'Aditya Patel', '7898 Ocean Drive, Chennai', '9876543214', 27),

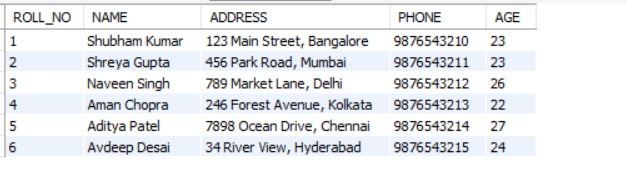
(6, 'Avdeep Desai', '34 River View, Hyderabad', '9876543215', 24);

Select \* from students;



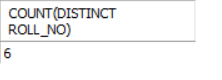
* Using distinct keyword we don’t retrieve repeated data

SELECT DISTINCT \* FROM students;



* We can use count to get total sum

SELECT COUNT(DISTINCT ROLL\_NO) FROM Students;



**SQL\_Day document**

* Here we can first create customer table and insert some values into the customer table and retrieve everything

CREATE TABLE customer (

CustomerID INT,

CustomerName VARCHAR(255),

ContactName VARCHAR(255),

Address VARCHAR(255)

);

INSERT INTO customer (CustomerID, CustomerName, ContactName, Address)

VALUES

(1, 'Alfreds Futterkiste', 'Maria Anders', 'Obere Str. 57'),

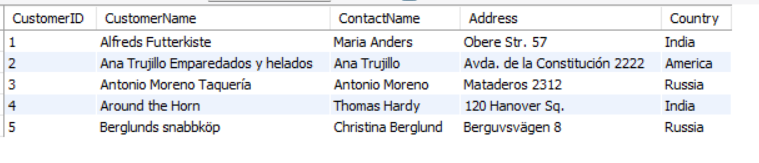
(2, 'Ana Trujillo Emparedados y helados', 'Ana Trujillo', 'Avda. de la Constitución 2222'),

(3, 'Antonio Moreno Taquería', 'Antonio Moreno', 'Mataderos 2312'),

(4, 'Around the Horn', 'Thomas Hardy', '120 Hanover Sq.'),

(5, 'Berglunds snabbköp', 'Christina Berglund', 'Berguvsvägen 8');

SELECT \* FROM Customer;

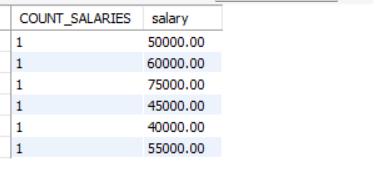
****

* Here we can write group by statement and the noted pointed here is group by is always write after the aggrgrate functions

SELECT COUNT(\*) AS COUNT\_SALARIES, salary

FROM EMPLOYEE

GROUP BY salary;



* Now we create customers table and insert values in it and retrieve everything

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 VALUES

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

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

(3, 'Kaushik', 23, 'Kota', 2000.00),

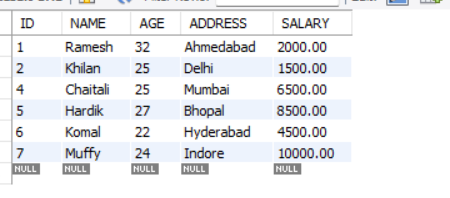
(4, 'Chaitali', 25, 'Mumbai', 6500.00),

(5, 'Hardik', 27, 'Bhopal', 8500.00),

(6, 'Komal', 22, 'Hyderabad', 4500.00),

(7, 'Muffy', 24, 'Indore', 10000.00);

select \* from customers;



* Now we perform Commit and roleback operations

DELETE FROM CUSTOMERS WHERE AGE = 23;

COMMIT;

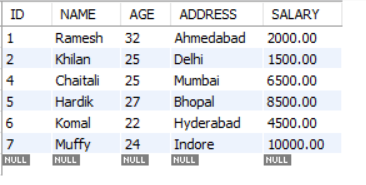
select \* from customers;

START TRANSACTION;

DELETE FROM CUSTOMERS WHERE AGE = 25;

ROLLBACK;

* Here the data is deleted and successfully rollbacked so we retrieve data we can get everything without deleted



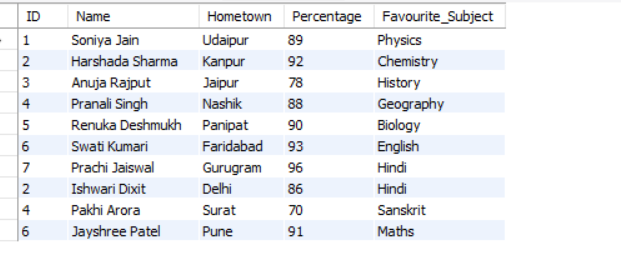
* SET operators in sql
* First we create t\_employees and insert some values
* Next we create t2\_employees and insert some values
* And then we create t\_students and insert some values
* And lastly we create t2\_students and insert values and we are perfoming set operations for these tables

UNION OPERATION

* It gives data in both tables and does not repeat the data

SELECT \*FROM t\_employees UNION SELECT \*FROM t2\_employees;

SELECT \*FROM t\_students UNION SELECT \*FROM t2\_students;

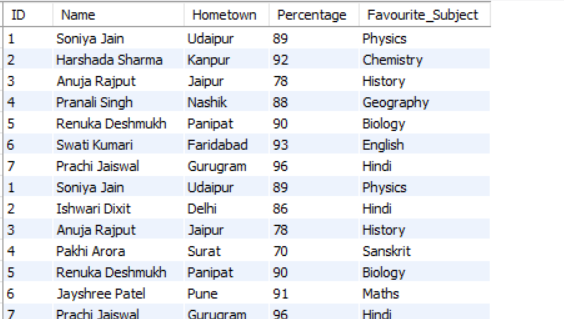


UNION ALL

* Here also the data is retrieve from both tables and values is repeated

SELECT \*FROM t\_employees UNION ALL SELECT \*FROM t2\_employees;

SELECT \*FROM t\_students UNION ALL SELECT \*FROM t2\_students;

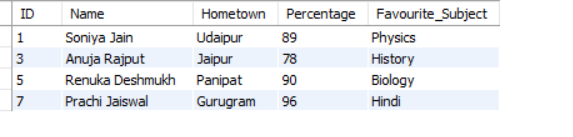


INTERSECT

* The data which is present in common of both tables that data is only retrieved

SELECT \*FROM t\_employees INTERSECT SELECT \*FROM t2\_employees;

SELECT \*FROM t\_students INTERSECT SELECT \*FROM t2\_students;



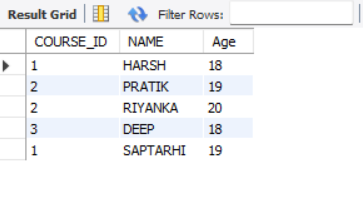
**SQL JOINS**

INNER JOIN

* It joins both table until condition satisfies
* The result of an INNER JOIN is a new table that contains only the rows where there is a match in the specified columns from both tables. If there is no match, the row is excluded from the result set.

select studentcourse.COURSE\_ID,student.NAME,student.Age

from studentcourse join student on studentcourse.ROLL\_NO=student.ROLL\_NO;

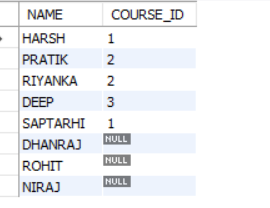


LEFT JOIN

* This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null.

select student.NAME,studentcourse.COURSE\_ID from student

left join studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO

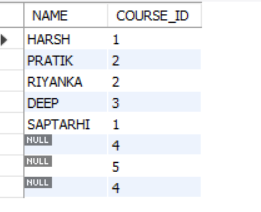


RIGHT JOIN

* RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null.

select student.NAME,studentcourse.COURSE\_ID from student right join

studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO;



FULL JOIN

* FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values

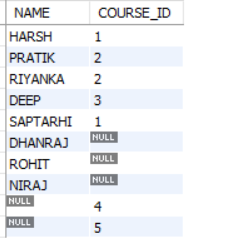
select student.NAME,studentcourse.COURSE\_ID from student

left join studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO

union

select student.NAME,studentcourse.COURSE\_ID from student right join

studentcourse on student.ROLL\_NO=studentcourse.ROLL\_NO;



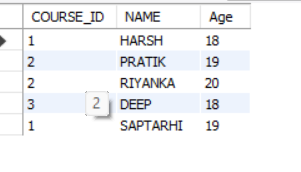
NATURAL JOIN

* In a NATURAL JOIN, the join is based on all columns with the same name in both tables. You don't need to explicitly specify the columns or conditions for the join.

SELECT studentcourse.COURSE\_ID, student.NAME, student.Age

FROM studentcourse

NATURAL JOIN student;



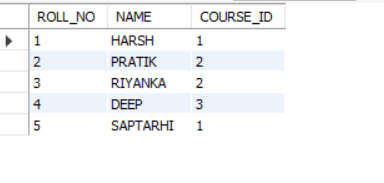
EQUI JOIN

* In an EQUI join, you typically join tables based on the equality of columns. In this example, I'll join the Student and StudentCourse tables based on the common column ROLL\_NO.

SELECT Student.ROLL\_NO, Student.NAME, StudentCourse.COURSE\_ID

FROM Student

JOIN StudentCourse ON Student.ROLL\_NO = StudentCourse.ROLL\_NO;



NON-EQUII JOIN

* In a NON-EQUI join, you join tables based on conditions other than equality. In this example, I'll use a condition where the Age of the student is greater than 18.

SELECT Student.ROLL\_NO, Student.NAME, StudentCourse.COURSE\_ID

FROM Student

JOIN StudentCourse ON Student.Age > 18;

