**CREATE TABLE `pets` (**

petid int,

name varchar(20),

owner varchar(20),

species varchar(20),

gender char(1)

);

INSERT INTO pets VALUES (1001, 'Tommy', 'Raj', 'Dog', 'M'),(1010, 'Fluffy', 'Harold', 'cat', 'F'),(1020, 'Chirpy', 'Gwen', 'bird', 'F'), (1025, 'Ginger', 'Glen', 'bird', 'F') ;

select Distinct name from firsttest.pets;

SELECT petid, name FROM firsttest.pets WHERE (gender ="F");

SELECT petid, name FROM firsttest.pets WHERE (gender ="F" AND species ="bird");

SELECT petid, name, species , gender FROM firsttest.pets WHERE (gender ="M" OR species ="bird");

SELECT name, species, gender FROM firsttest.pets ORDER BY name DESC;

**CONSTRAINTS**

**Primary**

create table firsttest.emp (empid int primary key Not Null, name varchar(20));

insert into firsttest.emp values (1001 , 'Raj'), (1002 , 'Ram'), (1003 , 'Ravi');

select \* from emp;

insert into firsttest.emp values(NULL, 'Renu');

**Foreign key**

create table firsttest.branch (id int primary key Not Null, deptname varchar(20), empid int, age int);

insert into firsttest.branch values (10 , 'IT', 1001, 25), (11 , 'MECH', 1002, 28), (12 , 'IT', 1003, 40);

select \* from firsttest.branch;

alter table branch add constraint adding\_foreignkey foreign key (empid) references emp (empid);

desc branch;

insert into firsttest.emp values (1005, 'IT');

insert into firsttest.branch values (9, 'IT', 1006, 44);

ERROR OCCURS, SO ADD ONLY EMPID WHICH IS PRESENT IN EMP TABLE.

**Unique**

create table firsttest.branchone (id int, deptname varchar(20), empid int, empage int);

insert into firsttest.branchone values (10 , 'IT', 1001, 25), (11 , 'MECH', 1002, 29), (12 , 'IT', 1003, 40);

select \* from firsttest.branchone;

alter table branchone add constraint unique (id);

insert into firsttest.branchone values (10 , 'IT', 1004, 44);

**Check**

SELECT \* FROM firsttest.branch;

alter table firsttest.branch add constraint chk\_empid CHECK (empid>=1005);

alter table firsttest.branch add constraint chk\_empid CHECK (empid<=1010);

**DEFAULT**

SELECT \* FROM firsttest.emp;

ALTER TABLE firsttest.emp ALTER COLUMN name SET DEFAULT "AUTO";

insert into firsttest.emp (empid) values (1014);

SELECT \* FROM firsttest.emp;

**JOINS**

use firsttest;

create table students (student\_id int, stud\_firstname varchar(20), stud\_lastname varchar(20), city varchar(20));

insert into students values (1, 'Melvin', 'Putin', 'France'), (2, 'Michael','Clark', 'Australia'),(3,'Ethon','Miller', 'England'),(4,'Mark','Strauss','America');

select \* from students;

create table technologies (student\_id int, tech\_id int, institute\_name varchar(20), technology varchar(20));

insert into technologies values (1,1,'Java Institute', 'JAVA'), (2,2,'Chroma Campus','Angular'),(3,3, 'Cetpa Infotech', 'Big Data'), (4,4,'Aptron Institute','IOS');

select \* from technologies;

create table contact (college\_id int, cellphone int, homephone int);

insert into contact values (1, 983547898, 7837647), (2, 674837645, 989338), (3,758393939,872678);

select \* from contact;

**INNER JOIN**

**Basic INNER JOIN between two tables**

SELECT students.stud\_firstname, students.stud\_lastname, students.city, technologies.technology

FROM students

INNER JOIN technologies

ON students.student\_id = technologies.tech\_id;

the keyword "ON" is used to specify the condition that determines how the tables should be joined.

**Inner Join with Group By Clause**

SELECT students.student\_id, technologies.institute\_name, students.city, technologies.technology

FROM students

INNER JOIN technologies

ON students.student\_id = technologies.tech\_id GROUP BY institute\_name;

Groups the results by the institute\_name column from the technologies table.

**Sometimes, the name of the columns is the same in both the tables. In that case, we can use a USING keyword to access the records. The following query explains it more clearly:**

**Inner Join with USING clause**

SELECT student\_id, institute\_name, city, technology

FROM students

INNER JOIN technologies

USING (student\_id);

The WHERE clause enables you to return the filter result.

**Inner Join with WHERE Clause**

SELECT tech\_id, institute\_name, city, technology

FROM students

INNER JOIN technologies

USING (student\_id) WHERE technology = "Java";

**Inner Join Multiple Tables**

SELECT student\_id, institute\_name, city, technology, cellphone

FROM students

INNER JOIN technologies USING (student\_id)

INNER JOIN contact ORDER BY student\_id;

**LEFT OUTER JOIN**

create table orders (OrderID int, OwnerName varchar(20), CustomerID int, OrderDate date, Price int);

create table customers (CustomerID int, CustomerName varchar(30), Occupation varchar(20), Income int, Qualification varchar(20) );

insert into orders values (1001,'Raj',2,'2020-03-20',3000),(1002,'Shankar',4,'2020-02-15',2500),(1003,'Sheik',5,'2020-01-31',5000),(1004,'Fazar',2,'2020-03-10',1500),(1005,'Shekar',1,'2020-02-20',4500);

insert into customers values (1,'Alfreds','Developer',20000,'BTech'), (2, 'Mark','Engineer',40000,'BTech'),(3,'Reyan','Scientist',60000,'MSc'),(4, 'Shane','Businessman',10000,'MBA'),(5,'Adam','Manager',80000,'MBA'),(6,'Ricky','Cricketer',20000,'BTech');

select \* from orders;

select \* from customers;

**LEFT JOIN clause for joining two tables**

SELECT \*

FROM Customers

LEFT OUTER JOIN Orders ON Customers.customerid = Orders.customerid;

**LEFT JOIN with USING Clause**

SELECT \*

FROM Customers

LEFT OUTER JOIN Orders USING (customerid);

**MySQL LEFT JOIN with Group By Clause**

SELECT Customers.customerId, Customers.CustomerName, COUNT(Orders.OrderId) AS order\_count

FROM Customers

LEFT JOIN Orders ON Customers.customerId = Orders.customerId

GROUP BY Customers.customerId, Customers.CustomerName;

**LEFT JOIN with WHERE Clause**

SELECT \*

FROM Customers

LEFT OUTER JOIN Orders ON Customers.customerid = Orders.customerid

WHERE Orders.orderdate >= '2020-02-15';

**LEFT JOIN with HAVING Clause**

SELECT Customers.customerid, Customers.customername, COUNT(Orders.orderid) AS order\_count

FROM Customers

LEFT OUTER JOIN Orders ON Customers.customerid = Orders.customerid

GROUP BY Customers.customerid

HAVING COUNT(Orders.orderid) > 0;

**MySQL LEFT JOIN Multiple Tables**

SELECT \*

FROM Customers

LEFT OUTER JOIN Orders ON Customers.customerId = Orders.customerId

AND Orders.OrderDate >= '2020-02-15'

WHERE Customers.Qualification = 'BTech';

**RIGHT OUTER JOIN**

This retrieves all rows from the right table Orders table and the matching rows from left table the Customers table.

**Basic Right Outer Join:**

SELECT \*

FROM Customers

RIGHT OUTER JOIN Orders ON Customers.customerid = Orders.customerid;

**Right Outer Join with WHERE Clause:**

SELECT \*

FROM Customers

RIGHT OUTER JOIN Orders ON Customers.customerid = Orders.customerid

WHERE Customers.Qualification = 'BTech';

**Right Outer Join with USING Clause:**

SELECT \*

FROM Customers

**RIGHT OUTER JOIN Orders USING (customerid) ;**

**Right Outer Join with HAVING Clause and GROUP BY:**

SELECT Customers.customername,Customers.CUSTOMERID,COUNT(Orders.orderid) AS order\_count

FROM Customers

right OUTER JOIN Orders ON Customers.customerid = Orders.customerid

GROUP BY Customers.customername,Customers.CUSTOMERID

HAVING COUNT(Orders.orderid) >=1;

**RIGHT JOIN Multiple Tables**

SELECT CUSTOMERS.CustomerID,CustomerName,Occupation,Qualification,OrderDate

FROM Customers

RIGHT OUTER JOIN Orders ON Customers.customerid = Orders.customerid

AND Orders.OrderDate >= '2020-02-15'

WHERE Customers.qualification = 'MBA';

**SELF JOIN**

SELECT e1.customername as EMPLOYEE , e2.ownername as MANAGER

FROM customers e1

JOIN orders e2 ON e1.customerid = e2.customerid

**Self Join with WHERE Clause:**

SELECT e1.customername as EMPLOYEE , e2.ownername as MANAGER

FROM customers e1

JOIN orders e2 ON e1.customerid = e2.customerid

WHERE e1.qualification = 'BTECH';

**CROSS JOIN**

SELECT e1.CustomerID, e1.customerName, e2.price, e2.orderDate

FROM customers e1

CROSS JOIN orders e2;

**REGEX**

SELECT stud\_firstname FROM students WHERE stud\_firstname REGEXP '^m';

SELECT stud\_firstname FROM students WHERE stud\_firstname REGEXP 'n$';

SELECT city FROM students WHERE city REGEXP 'ca|ra';

**Retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any character, followed by** **the letter ‘a’**

SELECT stud\_lastname FROM students WHERE stud\_lastname REGEXP '[d-t].[r]' ;

Query to find all the names starting with e to o and ending with 'n' −

SELECT stud\_firstname FROM students WHERE stud\_firstname REGEXP '^[e-o].\*n$';

the asterisk (\*) is a quantifier that means "zero or more occurrences of the preceding element."

**Regex functions**

SELECT \* FROM students WHERE REGEXP\_LIKE(stud\_firstname, '^[e-h]');

SELECT REGEXP\_REPLACE(city, 'france', 'England') FROM students;

SELECT REGEXP\_INSTR(stud\_firstname, 'Ethon') FROM students;

SELECT REGEXP\_SUBSTR(stud\_firstname, 'Ethon') AS extracted\_name FROM students;

**SUBQUERIES**

Using a subquery, list the name of the employees, paid more than Michael from employees.

SELECT first\_name,last\_name, salary FROM EMPLOYEES WHERE salary >(SELECT salary FROM EMPLOYEES WHERE first\_name='Michael');

**MySQL Subqueries: Scalar operand**

create table Department (Department\_ID int, Department\_NAME varchar(30));

insert into Department values ( 10, 'SDLC'),(20,'VLED'),(40,'WERS');

**Retrieve the position of job for the employees with the corresponding manager id.**

SELECT (

SELECT job\_id

FROM employees where manager\_id = 102

) AS JOB\_POSITION;

SELECT (

SELECT job\_id

FROM employees where manager\_id = 104

) AS JOB\_POSITION;

SELECT (

SELECT job\_id

FROM employees where manager\_id = 107

) AS JOB\_POSITION;

**MySQL Subqueries: Using Comparisons**

**suppose you want to find the employee id, first\_name, last\_name, and salaries for employees whose average salary is higher than the average salary throughout the company**

SELECT employee\_id,first\_name,last\_name,salary FROM employees WHERE salary > (SELECT AVG(SALARY) FROM employees);

**Subqueries with ALL, ANY, IN, or SOME**

**Find all employees whose salary is greater than all the salaries in the department with department\_id 40.**

SELECT \*

FROM employees

WHERE salary > ALL (SELECT salary FROM employees WHERE department\_id = 40);

**Find all employees whose salary is greater than the highest salary in the department with department\_id 40.**

SELECT \*

FROM employees

WHERE salary > ANY (SELECT MAX(salary) FROM employees WHERE department\_id = 40);

**Finding Employees with Specific MANAGER\_ID returned by a subquery.**

SELECT first\_name

FROM employees

WHERE MANAGER\_ID IN (SELECT MANAGER\_ID FROM employees WHERE department\_id = 40);

**Subqueries with EXISTS or NOT EXISTS**

**find employees (employee\_id, first\_name, last\_name, job\_id, department\_id) who have at least one person reporting to them.**

SELECT employee\_id, first\_name, last\_name, job\_id, department\_id

FROM employees E

WHERE EXISTS (SELECT \* FROM department WHERE department\_id = E.department\_id);

SELECT department\_id, department\_name

FROM department d

WHERE NOT EXISTS (SELECT \* FROM employees WHERE department\_id = d.department\_id);