# Practical No:-02

C:\Users\Swati>mysql -u root -p

Enter password: root

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 9

Server version: 8.0.35 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

CREATE DATABASE Nikhil;

USE Nikhil;

CREATE TABLE Deposit(

actno int PRIMARY KEY, cname varchar(200), bname varchar(200), amount int,

adate date

);

CREATE TABLE Branch(

bname varchar(200), city varchar(200)

);

CREATE TABLE Customer(

Cname varchar(200), city varchar(200)

);

CREATE TABLE Borrow(

loanno int,

cname varchar(200), bname varchar(200), amount int

);

INSERT INTO Deposit(actno,cname,bname,amount,adate) VALUES

(1234,'Anil', 'Wagholi', 1000, '2015-08-11'),

(9122, 'Ram', 'Hadapsar',2000, '2011-04-18'),

(7066, 'Rohit', 'Karolbagh', 3000, '2001-04-18'),

(9223, 'Sunil','Thane', 4000, '2016-07-22');

INSERT INTO Branch(bname,city) VALUES

('Wagholi', 'pune'),

('hadapsar', 'pune'),

('karolbagh', 'Delhi'),

('Thane', 'Mumbai');

INSERT INTO Customer (Cname, city) VALUES

('Anil', 'pune'),

('Ram','pune'),

('Rohit', 'Delhi'),

('Sunil', 'Mumbai');

INSERT INTO Borrow(loanno, cname, bname, amount) VALUES

(5556, 'Ram', 'Hadapsar', 20000),

(9875, 'Rohit','Karolbagh', 69000),

(5265, 'Sunil', 'Thane', 100000);

SELECT adate FROM Deposit WHERE cname='Anil';

UPDATE Deposit SET amount= 20000

WHERE actno= 1234;

SELECT cname FROM Customer WHERE city= 'Pune';

SELECT city FROM Branch WHERE bname='Karolbagh';

SELECT COUNT(Cname) FROM Customer

DELETE FROM borrow WHERE cname = 'Sunil'; CREATE VIEW ViewDemo AS

SELECT actno, Cname FROM Deposit

select \* from ViewDemo;

# Practical No:03

CREATE DATABASE NikhilRai;

USE NikhilRai;

CREATE TABLE Departments(

DepartmentID INT,

DepartmentName VARCHAR(100), PRIMARY KEY(DepartmentID)

);

CREATE TABLE Employees(

EmployeeID INT ,

FirstName VARCHAR(100), LastName VARCHAR(100),

DepartmentID INT,

PRIMARY KEY(EmployeeID),

FOREIGN KEY(DepartmentID) REFERENCES Departments(DepartmentID)

);

INSERT INTO Departments VALUES

(100, 'HR'),

(105, 'Sales'),

(110, 'IT'),

(115, 'Marketing');

INSERT INTO Employees VALUES

(421, 'Nikhil', 'Rai', 100),

(632, 'Mayur', 'Gulve', 105),

(712, 'Ritesh', 'Gaikwad', 100),

(892, 'Vishal','Gaikwad', 110);

-- 1.Give a Cartesian Product of Employees and Departments. (Cartesian Product) SELECT \* FROM Employees CROSS JOIN Departments;

-- Inner Join

-- 2.Provide all details of employees whose department ID is greater than 101 along with their respective departments.(theta join)

SELECT \* FROM Employees e INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID AND e.DepartmentID > 101;

-- 3.Give all detils of employees. (Equi Join)

SELECT \* FROM Employees e INNER JOIN Departments d ON e.DepartmentID = d.DepartmentID;

-- 4.Give all details of employees using a Natural Join.(Natural Join) SELECT \* FROM Employees e NATURAL JOIN Departments d;

-- outer join

-- 5.Show all employees and their respective department details, if available. (Left Outer Join) SELECT \* FROM Employees e LEFT JOIN Departments d

ON e.DepartmentID = d.DepartmentID;

-- 6.Show all departments and their respective employees, if available. (Right Outer Join) SELECT \* FROM Employees e RIGHT JOIN Departments d

ON e.DepartmentID = d.DepartmentID;

-- 7.Give the full details of employees along with their respective departments. (Full Outer Join) SELECT \* FROM Employees e

LEFT JOIN Departments d ON e.DepartmentID = d.DepartmentID UNION

SELECT \* FROM Employees e

RIGHT JOIN Departments d ON e.DepartmentID = d.DepartmentID;

-- 8.Provide the first names and last names of HR employees without using a join. (Subquery) SELECT FirstName, LastName FROM Employees

WHERE DepartmentID = (SELECT DepartmentID FROM Departments WHERE DepartmentName = 'HR');

-- 9.Create and display a view of the Employees table. (Views) CREATE VIEW EmployeeView AS

SELECT e.EmployeeID, e.FirstName, e.LastName, d.DepartmentName FROM Employees e

INNER JOIN Departments d ON e.DepartmentID = d.DepartmentID; SELECT \* FROM EmployeeView;

# Practical No:-04

CREATE DATABASE Nikhil;

USE Nikhil;

CREATE TABLE borrower(roll\_no INT,

name VARCHAR(50),

dateofissue DATE,

name\_of\_book VARCHAR(50), status varchar(1)

);

INSERT INTO borrower VALUES

(101, 'Nikhil', '2022-08-01', 'DBMS', 'I'),

(102, 'Vishal', '2022-09-18', 'IOT', 'I'),

(103, 'Radhika', '2022-10-02', 'CNS', 'I'),

(104, 'Ram', '2022-09-05', 'TOC', 'I');

CREATE TABLE fine(roll\_no INT,

date\_of\_return DATE, amt INT);

DELIMITER //

CREATE PROCEDURE CalculateFine(IN p\_roll\_no INT, IN p\_book\_name VARCHAR(50)) BEGIN

DECLARE v\_issue\_date DATE; DECLARE v\_days INT; DECLARE v\_fine INT;

-- Retrive the issue date from the Borrower table SELECT 'dateofissue' INTO v\_issue\_date

FROM borrower

WHERE roll\_no = p\_roll\_no

AND 'name\_of\_book' = p\_book\_name;

-- calculate the number of days since issue

SET v\_days = DATEDIFF(CURDATE(), v\_issue\_date);

-- Apply fine rules

IF v\_days BETWEEN 15 AND 30 THEN

SET v\_fine = v\_days \* 5; ELSEIF v\_days > 30 THEN SET v\_fine = v\_days \* 50; ELSE

SET v\_fine = 0; END IF;

-- Insert the fine record into the Fine table if (v\_fine is not null) then

INSERT INTO fine(roll\_no, date\_of\_return, amt)

VALUES (p\_roll\_no, CURDATE(), v\_fine);

-- Update Status in Borrower table UPDATE borrower

SET status = 'R'

WHERE roll\_no = p\_roll\_no

AND `name\_of\_book` = p\_book\_name; End if;

END //

DELIMITER ;

-- Fine Table before procedure call:

SELECT \* FROM Fine;

-- Procedure is Called

CALL CalculateFine(104,'TOC');

-- Fine Table after procedure call:

SELECT \* FROM fine;

-- Borrower Table After submitting the book, status will change from I to R. SELECT \* FROM borrower;

# Practical No:-05

1. USE Nikhil;
2. CREATE TABLE Stud\_Marks ( Roll int, name VARCHAR(50), total\_marks INT);
3. INSERT INTO Stud\_Marks VALUES (1, "Nikhil", 991),(2, "Vishal", 912),(3, "Ritesh", 840),(4, "Mayur", 800);

4)CREATE TABLE Result ( Roll INT,

Name VARCHAR(50), Class VARCHAR(50)

);

1. DELIMITER //

CREATE PROCEDURE proc\_Grade(

OUT p\_name VARCHAR(50),

OUT p\_total\_marks INT, IN p\_Roll INT,

OUT p\_Class VARCHAR(50)

)

BEGIN

SELECT name, total\_marks INTO p\_name, p\_total\_marks FROM Stud\_Marks WHERE Roll = p\_Roll;

IF p\_total\_marks >= 990 THEN SET p\_class = "Distinction";

ELSEIF p\_total\_marks >=900 AND p\_total\_marks <= 989 THEN SET p\_class = "First Class";

ELSEIF p\_total\_marks >=825 AND p\_total\_marks <= 899 THEN SET p\_class = "Higher Second Class";

ELSE

SET p\_class = "Not Classified"; END IF;

INSERT INTO Result (Roll, Name, Class) VALUES (p\_Roll, p\_name, p\_Class); END //

DELIMITER ;

1. SELECT \* FROM stud\_Marks;
2. CALL proc\_Grade(@name, @marks, 1, @class);
3. CALL proc\_Grade(@name, @marks, 2, @class);
4. SELECT \* FROM result;

# Practical No:-06

1. Create Database nikhil
2. use nikhil;
3. Create N\_RollCall table

-> CREATE TABLE N\_RollCall (

RollNumber INT PRIMARY KEY, Name VARCHAR(25)

);

1. Create O\_RollCall table

-> CREATE TABLE O\_RollCall (

RollNumber INT PRIMARY KEY, Name VARCHAR(25)

);

1. Insert sample data into N\_RollCall table

-> INSERT INTO N\_RollCall (RollNumber, Name) VALUES (1, 'nikhil'), (2, 'Mayur'), (3, 'Ritesh');

1. Insert sample data into O\_RollCall table

-> INSERT INTO O\_RollCall (RollNumber, Name) VALUES (1, 'nikhil'), (4, 'Rahul'), (5, 'Chetan');

1. Create the Procedure

-> DELIMITER //

CREATE PROCEDURE MergeData() BEGIN

DECLARE rln INT;

DECLARE nm VARCHAR(25);

-- Declare a cursor for N\_RollCall

DECLARE cur CURSOR FOR SELECT RollNumber, Name FROM N\_RollCall;

-- Open the cursor OPEN cur;

-- Loop through the cursor read\_loop: LOOP

FETCH cur INTO rln, nm;

-- Check if the data already exists in O\_RollCall

IF NOT EXISTS (SELECT 1 FROM O\_RollCall WHERE RollNumber = rln AND Name = nm) THEN

-- Insert the data into O\_RollCall

INSERT INTO O\_RollCall (RollNumber, Name) VALUES (rln, nm); END IF;

END LOOP;

-- Close the cursor CLOSE cur;

END //

1. Call procedure CALL MergeData();
2. Select data from O\_RollCall to verify the merge SELECT \* FROM O\_RollCall;

# Practical No:-07

create database Nikhil;

use Nikhil;

CREATE TABLE library (

Book\_no INT PRIMARY KEY,

Book\_name VARCHAR(255), author VARCHAR(255)

);

CREATE TABLE library\_audit( action\_type VARCHAR(50), Book\_no INT,

old\_Book\_name VARCHAR(50), old\_Author VARCHAR(50)

);

INSERT INTO library(Book\_no,Book\_name, Author)

VALUES(421,'Maths','Ajay'),(534,'Scince','Robin'),(792,'Physics','Chetan'),(999,'English','Ritesh');

);

select \* from library;

select \* from library\_audit;

DELIMITER //

CREATE TRIGGER library\_update\_trigger BEFORE UPDATE ON library

FOR EACH ROW BEGIN

INSERT INTO library\_audit (action\_type, Book\_no,

old\_Book\_name, old\_Author

)

values('update', OLD.Book\_no, old.Book\_name,old.Author); END//

DELIMITER //

CREATE TRIGGER library\_delete\_trigger BEFORE DELETE ON library

FOR EACH ROW BEGIN

INSERT INTO library\_audit (action\_type, Book\_no,

old\_Book\_name, old\_Author

)

values('delete', OLD.Book\_no, old.Book\_name,old.Author); END//

DELIMITER ;

UPDATE library

SET Book\_name = 'Science' WHERE Book\_no = 534;

select \* from library\_audit ;

DELETE FROM library WHERE Book\_no =999;

select \* from library\_audit ;

DELIMITER //

CREATE TRIGGER library\_insert\_trigger AFTER INSERT ON library

FOR EACH ROW BEGIN

INSERT INTO library\_audit (action\_type, Book\_no,

old\_Book\_name, old\_Author

)

values('insert', NEW.Book\_no, NEW.Book\_name,NEW.Author); END//

DELIMITER;

INSERT INTO library(Book\_no,Book\_name, Author) VALUES(1001,'History','Vijay');