--program 04--

create database Employee\_db

USE Employee\_db;

CREATE TABLE CUSTOMERS(

ID INT PRIMARY KEY,

NAME VARCHAR(255),

AGE INT,

ADDRESS VARCHAR(255),

SALARY DECIMAL(10, 2));

INSERT INTO CUSTOMERS (ID, NAME, AGE,ADDRESS,SALARY) VALUES

(002, 'AJAY', 19,'Marballi' ,50000.00),

(007, 'DARSHAN', 20,'Mysore' ,60000.00),

(009, 'DEEKSHITH', 21,'Kollegal', 70000.00),

(010, 'DHAKSHATH', 22,'Mandya', 45000.00),

(039, 'PRAJWAL', 45,'Chamarajanagara' ,80000.00);

SELECT \* FROM CUSTOMERS

DELIMITER //

CREATE TRIGGER after\_insert\_salary\_difference

AFTER INSERT ON CUSTOMERS

FOR EACH ROW

BEGIN

SET @my\_sal\_diff = CONCAT('Salary inserted is ', NEW.SALARY);

END//

DELIMITER ;

INSERT INTO CUSTOMERS VALUES (001, 'amit kumar', 38,'bijapur', 90000.00)

SELECT@my\_sal\_diff as SALARY\_INSERTED

DELIMITER //

CREATE TRIGGER after\_update\_salary\_difference

AFTER UPDATE ON CUSTOMERS

FOR EACH ROW

BEGIN

DECLARE old\_salary DECIMAL(10, 2);

DECLARE new\_salary DECIMAL(10, 2);

SET old\_salary = OLD.SALARY;

SET new\_salary = NEW.SALARY;

SET @my\_sal\_diff = CONCAT('salary difference after update is ', NEW.SALARY - OLD.SALARY);

END;

// DELIMITER ;

INSERT INTO CUSTOMERS VALUES (6, 'amit kumar', 38,'bijapur', 90000.00)

UPDATE CUSTOMERS

SET SALARY=100000

WHERE ID=6

SELECT @my\_sal\_diff AS salary\_differnce

DELIMITER //

CREATE TRIGGER after\_delete\_salary\_difference

AFTER DELETE ON CUSTOMERS

FOR EACH ROW

BEGIN

SET @my\_sal\_diff = CONCAT('Salary deleted is ', OLD.SALARY);

END//

DELIMITER ;

select \* from CUSTOMERS

DELETE from CUSTOMERS where ID=1

SELECT @my\_sal\_diff AS Deleted\_Salary

---- program 5----

CREATE DATABASE Employee\_db

USE Employee\_db

CREATE TABLE ED1 (

E\_id INT,

E\_name VARCHAR(255),

Age INT,

Salary DECIMAL(10, 2)

);

INSERT INTO ED1 (E\_id, E\_name, Age, Salary) VALUES

(1, 'AJAY', 30, 50000.00),

(2, 'DARSHAN', 25, 45000.00),

(3, 'DEEKSHITH', 35, 62000.00),

(4, 'DHAKSHATH', 28, 52000.00),

(5, 'PRAJWAL', 32, 58000.00);

SELECT \* FROM ED1

DELIMITER $$

CREATE PROCEDURE gettable(INOUT Tableontents VARCHAR(4000))

BEGIN

DECLARE finished INTEGER DEFAULT 0;

DECLARE content VARCHAR(100) DEFAULT '';

DECLARE curName CURSOR FOR

SELECT CONCAT(E\_id, ' , ', E\_name, Age, ' , ', Salary)

FROM ED1

ORDER BY E\_id DESC;

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET finished = 1;

OPEN curName;

getName: LOOP

FETCH curName INTO content;

IF finished = 1 THEN

LEAVE getName;

END IF;

SET Tableontents = CONCAT(content, ';', Tableontents);

END LOOP getName;

CLOSE curName;

END$$

DELIMITER ;

SET @Tableontents = "";

CALL gettable(@Tableontents);

SELECT @Tableontents;

----P6---

CREATE DATABASE RC;

USE RC;

CREATE TABLE N\_RollCall (

student\_id INT PRIMARY KEY,

student\_name VARCHAR (255),

birth\_date DATE);

CREATE TABLE O\_RollCall (

student\_id INT PRIMARY KEY,

student\_name VARCHAR (255),

birth\_date DATE);

INSERT INTO O\_RollCall (student\_id, student\_name, birth\_date) VALUES

(5, 'Deekshith', '2004-08-05'),

(10, 'Dhakshath', '2004-03-01');

SELECT \* FROM O\_Rollcall

INSERT INTO N\_RollCall (student\_id, student\_name, birth\_date) VALUES

(2, 'Ajay', '2004-08-05'),

(8, 'Dhanush', '2004-04-03'),

(5, 'Vinay', '2004-09-03');

SELECT \* FROM N\_Rollcall

DELIMITER //

CREATE PROCEDURE merge\_rollcall\_data()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE n\_id INT;

DECLARE n\_name VARCHAR(255);

DECLARE n\_birth\_date DATE;

DECLARE n\_cursor CURSOR FOR

SELECT student\_id, student\_name, birth\_date FROM N\_RollCall;

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET done = TRUE;

OPEN n\_cursor;

cursor\_loop: LOOP

FETCH n\_cursor INTO n\_id, n\_name, n\_birth\_date;

IF done THEN

LEAVE cursor\_loop;

END IF;

IF NOT EXISTS (

SELECT 1 FROM O\_RollCall WHERE student\_id = n\_id)

THEN INSERT INTO O\_RollCall (student\_id, student\_name, birth\_date)

VALUES (n\_id, n\_name, n\_birth\_date);

END IF;

END LOOP;

CLOSE n\_cursor;

END //

DELIMITER ;

CALL merge\_rollcall\_data();

SELECT \* FROM O\_RollCall;

=== p07---=====

**use bookDB**

db.ProgrammingBooks.insertMany([

{

title: "Clean Code: A Handbook of Agile Software Craftsmanship",

author: "Robert C. Martin",

category: "Software Development",

year: 2008

})]

* db.programmingBooks.insertone([])

**db.ProgrammingBooks.find(). pretty ()**

**db.ProgrammingBooks.find({year: { $gt: 2000 } }). pretty ()**

**db.ProgrammingBooks.updateOne(**

**{ title: "Clean Code: A Handbook of Agile Software Craftsmanship" },**

**{ $set: { author: "Robert C. Martin (Uncle Bob)" } } )**

**db.ProgrammingBooks.find({ year: { $eq: 2008 } }).pretty()**

db.ProgrammingBooks.updateMany( { year: { $lt: 2010 } },

{ $set: { category: "Classic Programming Books" } } )

db.ProgrammingBooks.find({ year: { $lt: 2010 } }).pretty()

**db.ProgrammingBooks.deleteOne({ title: "JavaScript: The Good Parts" })**

**db.ProgrammingBooks.deleteMany({ year: { $lt: 1995 } })**

**db.ProgrammingBooks.deleteMany({})**

**db.ProgrammingBooks.find().pretty()**