

**Practical No. 2 NAME-Rathod Atul Ganesh**

**ROLL NO-45 DIV-A BATCH-A2**

1. Create deposit table create table deposit( actno int, cname text, bname text, amount text, adate int,

PRIMARY KEY

(actno)

);

1. Create deposit branch create table branch( bname text, city text

);

1. Create customers table create table customers( cname text, city text

);

1. Create borrow table create table borrow( loanno int, cname text, bname text, amount int,

PRIMARY KEY (loanno)

);

1. Insert values in all tables

#deposit insert into deposit values(123, "nilesh", "pune", 1000, '2022- 11-10'); insert into deposit values(456, "Anil", "pune", 1000, '2022-11-

11'); insert into deposit values(789, "siddharaj", "wagholi", 4500, '2022-

11-12');

#branch insert into branch values("pune", "pune"); insert into branch values("wagholi", "pune");

#customers

insert into customers values("atul", "nagar"); insert into customers values("Anil", "pune"); insert into customers values("siddharaj", "Wagholi"); #borrow insert into borrow values(987, "nilesh", "wagholi", 1000); insert into borrow values(654, "Raj", "wagholi", 1000); insert into borrow values(321, "Anil", "pune", 2000);

* 1. Display names of all branches located in city Bombay

SELECT bname FROM branch where city = "bombay";

* 1. Display account no. and amount of depositors.

SELECT actno, amount FROM deposit;

* 1. Update the city of customers Anil from Pune to Mumbai

UPDATE customers SET city="Mumbai" WHERE cname="Anil"; select \* from customers;

* 1. Find the number of depositors in the bank

SELECT COUNT(actno) FROM deposit;

* 1. Calculate Min,Max amount of customers.

MIN - SELECT MIN(amount) FROM deposit; MAX - SELECT MAX(amount) FROM deposit;

* 1. Create an index on deposit table syntax:

CREATE UNIQUE INDEX index\_name ON table\_name (column1, column2, ...);

Query:

CREATE INDEX depoIndex ON deposit(cname);

Display:

SHOW INDEX FROM deposit;

* 1. Create View on Borrow table syntax:

CREATE VIEW view\_name AS SELECT column1, column2, ... FROM table\_name WHERE condition;

Query:

CREATE VIEW viewDemo AS SELECT loanno, amount FROM borrow;

Display:

SELECT \* FROM viewDemo;

**Practical No. 3 NAME-Rathod Atul Ganesh**

**ROLL NO-45 DIV-A BATCH-A2**

1. create database atul;
2. use atul
3. create table departments(

departmentId int,

departmentName text,

primary key(departmentId)

)

4)

create table employees( empId int,

firstName text, lastName text,

departmentId int, primary key(empId), foreign key(departmentId) references departments(departmentId)

)

1. insert into departments values (1,"IT"), (2, "Design"), (3, "Testing")
2. insert into employees values (101, "atul", "Randhave", 1), (102, "siddharaj", "Gulve", 2), (104, "Ritesh", "Gaikwad", 3)

Queries

1. Give a Cartesian Product of Employees and Departments. (Cartesian Product) -> SELECT \* FROM employees CROSS JOIN departments;
2. Provide all details of employees whose department ID is greater than 1 along with their respective departments.(theta join)

-> SELECT \* FROM employees e INNER JOIN departments d

ON e.departmentID = d.departmentID AND e.departmentID > 1

1. Give all detils of employees. (Equi Join)

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

1. Give all details of employees using a Natural Join.(Natural Join)

-> SELECT \* FROM Employees e NATURAL JOIN Departments d;

1. 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;

1. 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;

1. 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

**Practical No. 4 NAME-Rathod Atul Ganesh**

**ROLL NO-45 DIV-A BATCH-A2**

1. CREATE TABLE borrower (roll\_no NUMERIC,name VARCHAR (25),dateofissue DATE, name\_of\_book VARCHAR (25),status VARCHAR (20));
2. CREATE TABLE fine( roll\_no NUMERIC,

date\_of\_return DATE,

amt NUMERIC);

1. INSERT INTO borrower VALUES(45,'ASHUTOSH','2022-08-01','HARRY POTTER','I'); INSERT INTO borrower VALUES(46,'ARYAN','2022-08-15','DARK MATTER','I');

INSERT INTO borrower VALUES(47,'atul','2022-08-24','SILENT HILL','I'); INSERT INTO borrower VALUES(48,'SANKET','2022-08-26','GOD OF WAR','I'); INSERT INTO borrower VALUES(49,'SARTHAK','2022-09-09','SPIDER-MAN','I');

1. DELIMITER //

CREATE PROCEDURE CalculateFine(IN p\_roll\_no INT, IN p\_book\_name VARCHAR(45)) BEGIN

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

-- Retrieve 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 \* 45; 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);

End if; END //

DELIMITER ;

-- Update Status in Borrower table UPDATE borrower

SET status = 'R'

WHERE roll\_no = p\_roll\_no

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

1. Call the Procedure

CALL CalculateFine(45, 'HARRY POTTER');

**Practical No. 5 NAME-Rathod Atul Ganesh**

**ROLL NO-45 DIV-A BATCH-A2**

1. CREATE TABLE Stud\_Marks ( Roll int,

name VARCHAR(45),

total\_marks INT

);

1. CREATE TABLE Result ( Roll INT,

Name VARCHAR(45), Class VARCHAR(45)

);

**3)**

DELIMITER //

CREATE PROCEDURE proc\_Grade( OUT p\_name VARCHAR(45),

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

OUT p\_Class VARCHAR(45)

) 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 ;

**4)** call the Procedure

CALL proc\_Grade (@name, @marks, 1, @Class);

**Practical No. 6 NAME-Rathod Atul Ganesh**

**ROLL NO-45 DIV-A BATCH-A2**

1. Create Database atul;
2. Use atul;
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, 'nilesh'), (2, 'siddharaj'), (3, 'Ritesh');

1. Insert sample data into O\_RollCall table

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

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;

THEN

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

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

-- 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. class procedure

-> CALL MergeData();

1. Select data from O\_RollCall to verify the merge

-> SELECT \* FROM O\_RollCall;

**Practical No. 7 NAME-Rathod Atul Ganesh**

**ROLL NO-45 DIV-A BATCH-A2**

* 1. create database atul;
  2. use atul;
  3. create library table

->

CREATE TABLE library ( book\_id INT PRIMARY KEY, title VARCHAR(255), author VARCHAR(255)

);

* 1. Create library\_audit table

->

CREATE TABLE library\_audit ( action\_type VARCHAR(10), book\_id INT, old\_title

VARCHAR(255),

old\_author VARCHAR(255)

);

* 1. Insert values into library table

->

INSERT INTO library (book\_id, title, author) VALUES

(1, 'bhagwat geeta', 'nilesh'),

(2, 'Hitman', 'siddharaj'),

(3, 'king kohli', 'Ritesh'),

(4, 'Missile Man of India', 'Nikhil'), (5, 'chandrayaan-3', 'krushna');

* 1. Display the library and library\_audit

-> select \* from library; select \* from library\_audit;

* 1. Create TRIGGER for update statement

|  |  |
| --- | --- |
| -> | DELIMITER // |
| -> | CREATE TRIGGER library\_update\_trigger |
|  | BEFORE UPDATE ON library |
|  | FOR EACH ROW |
|  | BEGIN |
|  | INSERT INTO library\_audit (action\_type, book\_id, old\_title, old\_author) |
|  | VALUES ('UPDATE', OLD.book\_id, OLD.title, OLD.author); |
|  | END// |

* 1. Create TRIGGER for Delete statement

-> DELIMITER //

-> CREATE TRIGGER library\_delete\_trigger BEFORE DELETE ON library

FOR EACH ROW BEGIN

INSERT INTO library\_audit (action\_type, book\_id, old\_title, old\_author) VALUES ('DELETE', OLD.book\_id, OLD.title, OLD.author);

END

//

* 1. Change the DELIMITER

-> DELIMITER ;

* 1. UPDATE library table

-> UPDATE library

SET author = 'nilesh' WHERE book\_id = 1;

* 1. See the changes in library\_audit

-> select \* from library\_audit ;

* 1. Delete a row from libary table

-> DELETE FROM library WHERE book\_id = 2;

* 1. See the changes in library\_audit

-> select \* from library\_audit ;

* 1. Creare Table for After trigger CREATE TABLE library\_audit\_after ( action\_type VARCHAR(10),

book\_id INT, new\_title

VARCHAR(255), new\_author VARCHAR(255)

);

1. After Trigger

-> DELIMITER //

-> CREATE TRIGGER library\_insert\_trigger AFTER INSERT ON library

FOR EACH ROW BEGIN

INSERT INTO library\_audit\_after (action\_type, book\_id, new\_title, new\_author) VALUES ('INSERT', NEW.book\_id, NEW.title, NEW.author);

END

//

1. INSERT INTO library (book\_id, title, author) VALUES

(1, 'bhagwat geeta', 'nilesh'),

1. select \* from library\_audit\_after;
2. Statement Level DELIMITER //

CREATE TRIGGER library\_insert\_statement\_trigger AFTER INSERT ON library

BEGIN

INSERT INTO library\_audit\_after (action\_type, book\_id, new\_title, new\_author) VALUES ('INSERT', NEW.book\_id, NEW.title, NEW.author);

END //