CREATE TABLE Employee (

EMPNO INT,

ENAME VARCHAR(50),

JOB VARCHAR(50),

MANAGER\_NO INT,

SAL DECIMAL(10, 2),

COMMISSION DECIMAL(10, 2)

* SQL CREATE DATABASE Statement

CREATE DATABASE *databasename*;

CREATE DATABASE practiceDB;

* SQL DROP DATABASE Statement

DROP DATABASE *databasename*;

DROP DATABASE practiceDB;

* SQL CREATE TABLE Statement

CREATE TABLE *table\_name*(  
*column1 datatype*,  
*column2 datatype*,  
*column3 datatype*,  
   ....  
);

**Ex:**

CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);

* SQL DROP TABLE Statement

DROP TABLE *table\_name*;

DROP TABLE Persons;

* SQL TRUNCATE TABLE

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

TRUNCATE TABLE *table\_name*;

* SELECT indicates which columns you'd like to view, and FROM identifies the table that they live in.

SELECT year,

month,

west

FROM *table\_name*;

* To select every column in a table, you can use \* instead of the column names:

SELECT \*

FROM *table\_name*;

* SQL INSERT INTO Statement

INSERT INTO *table\_name*  
VALUES (*value1*,*value2*,*value3*, ...);

INSERT INTO *Persons* VALUES (1, ‘Kumar’, ‘Ram’, ‘Vinobanagar’,'shimoga');

**DATABASE MANAGEMENT SYSTEM (BCS403)**

**Semester 4**

**PRACTICAL COMPONENT OF IPCC**

**Question 1**

Create a table called Employee & execute the following.

**Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)**

* **Create** a user and grant all permissions to the user.
* **Insert** the any three records in the employee table contains attributes

EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION and use **rollback.**

Check the result.

* Add **primary key constraint** and **not null constraint** to the employee table.
* Insert **null values** to the employee table and verify the result.

**Solution:**

CREATE TABLE Employee (

EMPNO INT,

ENAME VARCHAR (50),

JOB VARCHAR (50),

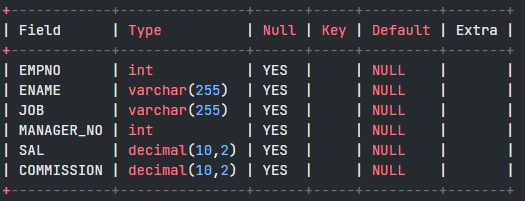
MANAGER\_NO INT,

SAL DECIMAL (10, 2),

COMMISSION DECIMAL (10, 2)

);

DESC Employee;



1. Create a user and grant all permissions to the user.

**Query:**

Type these two queries in the current user login:

CREATE USER IF NOT EXISTS 'dbuser1'@'localhost' IDENTIFIED BY ‘dbuser1’;

GRANT ALL PRIVILEGES ON Employee TO 'dbuser1'@'localhost';

Now logout and login with the new username as dbuser1 and password as dbuser1

Employee table should be visible in the new user dbuser1

**2. Insert** the any three records in the employee table contains attributes EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION and use **rollback.**

**Solution:**

INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION) VALUES (1, 'Kavana Shetty', 'Manager', NULL, 5000.00, 1000.00);

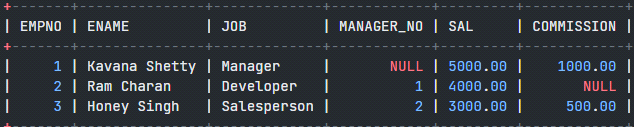
INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)

VALUES (2, 'Ram Charan', 'Developer', 1, 4000.00, NULL);

INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)

VALUES (3, 'Honey Singh', 'Salesperson', 2, 3000.00, 500.00);

SELECT \* FROM Employee;



**3.** Add **primary key constraint** and **not null constraint** to the employee table.

**Solution:**

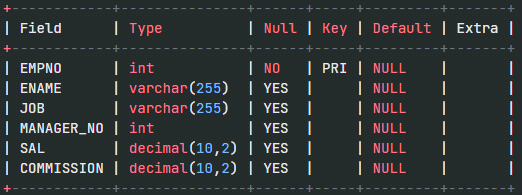
Add Primary Key Constraint

ALTER TABLE Employee

ADD CONSTRAINT pk\_employee PRIMARY KEY (EMPNO);

Verify primary key constraint

DESC Employee;



INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)

VALUES (1, 'Ranjan', 'Manager', NULL, 5000.00, 1000.00);

Duplicate entry '1' for key 'Employee.PRIMARY'

Since EMPNO field is the primary key it cannot have duplicate values, hence we see that the insert operation fails when provided with a duplicate value.

**Add Not Null Constraint**

ALTER TABLE Employee

MODIFY ENAME VARCHAR (255) NOT NULL,

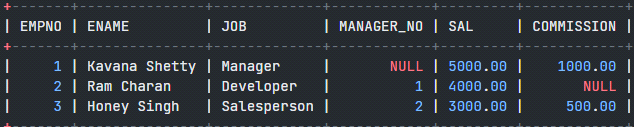
MODIFY JOB VARCHAR (255) NOT NULL,

MODIFY SAL DECIMAL (10, 2) NOT NULL;

INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)

VALUES (4, 'Ranjan', 'Manager', NULL, 5000.00, 1000.00);

SELECT \* FROM Employee;





INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)

VALUES (7, NULL, 'Tester', NULL, 3500.00, NULL);

ERROR 1048 (23000): Column 'ENAME' cannot be null

**Question 2**

Create a table called Employee that contain attributes EMPNO, ENAME, JOB, MGR, SAL & execute the following.

1. Add a column commission with domain to the Employee table.

2. Insert any five records into the table.

3. Update the column details of job

4. Rename the column of Employ table using alter command.

5. Delete the employee whose Empno is 105.

**Solution:**

CREATE TABLE Employee\_new (

EMPNO INT,

ENAME VARCHAR (50),

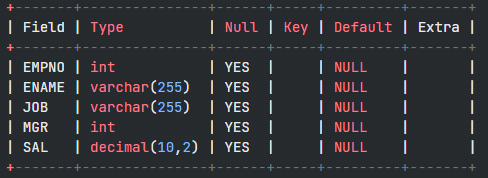
JOB VARCHAR (50),

MGR INT,

SAL DECIMAL (10, 2),

);

DESC Employee;

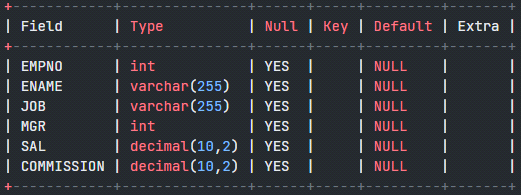


1. Add a column commission with domain to the Employee table.

ALTER TABLE Employee

ADD COLUMN COMMISSION DECIMAL (10, 2);

DESC Employee;



2. Insert any five records into the table.

INSERT INTO Employee (EMPNO, ENAME, JOB, MGR, SAL, COMMISSION)

VALUES

(101, 'Radha Bai', 'Manager', NULL, 5000.00, 1000.00),

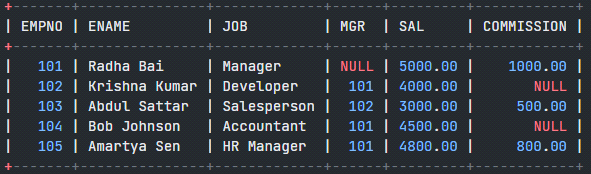
(102, 'Krishna Kumar', 'Developer', 101, 4000.00, NULL),

(103, 'Abdul Sattar', 'Salesperson', 102, 3000.00, 500.00),

(104, 'Bob Johnson', 'Accountant', 101, 4500.00, NULL),

(105, 'Amartya Sen', 'HR Manager', 101, 4800.00, 800.00);

SELECT \* FROM Employee;



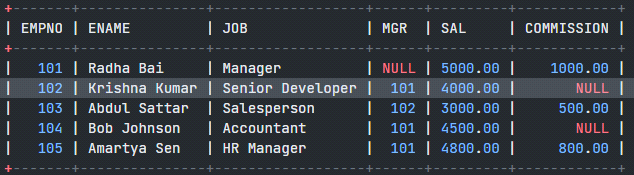
3. Update the column details of job

UPDATE Employee

SET JOB = 'Senior Developer'

WHERE EMPNO = 102;

SELECT \* FROM Employee;

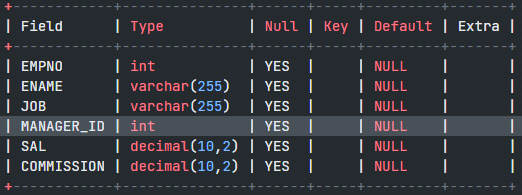


4. Rename the column of Employ table using alter command.

ALTER TABLE Employee

CHANGE COLUMN MGR MANAGER\_ID INT;

DESC Employee;

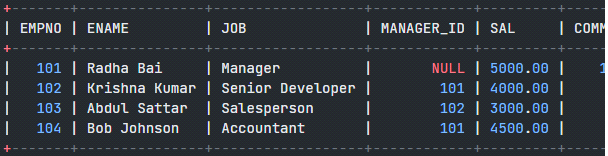


5. Delete the employee whose Empno is 105.

DELETE FROM Employee

WHERE EMPNO = 105;

SELECT \* FROM Employee;



**Question 3**

Queries using aggregate functions (COUNT, AVG, MIN, MAX, SUM), Group by, Orderby.

Employee (E\_id, E\_name, Age, Salary)

1. Create Employee table containing all Records E\_id, E\_name, Age, Salary.

2. Count number of employee names from employee table

3. Find the Maximum age from employee table.

4. Find the Minimum age from employee table.

5. Find salaries of employee in Ascending Order.

6. Find grouped salaries of employees.

**Solution:**

**1. Creating the Employee Table**

CREATE TABLE Employee (

E\_id INT PRIMARY KEY,

E\_name VARCHAR (255),

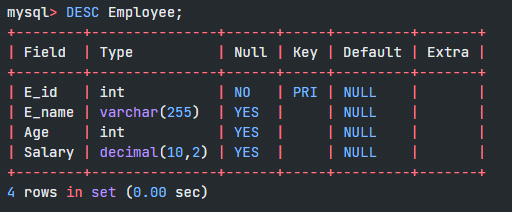
Age INT,

Salary DECIMAL (10, 2)

);

DESC Employee;

O/P:



**2. Insert 6 Records into the Employee Table**

INSERT INTO Employee (E\_id, E\_name, Age, Salary)

VALUES

(1, 'Samarth', 30, 50000.00),

(2, 'Ramesh Kumar', 25, 50000.00),

(3, 'Seema Banu', 35, 60000.00),

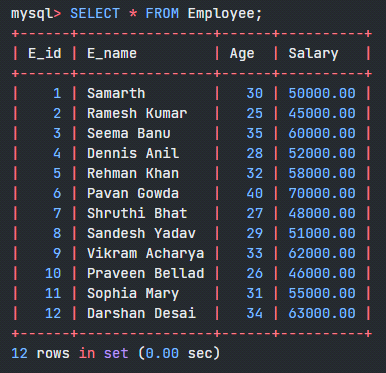
(4, 'Dennis Anil', 28, 58000.00),

(5, 'Rehman Khan', 32, 58000.00),

(6, 'Pavan Gowda', 40, 70000.00);

SELECT \* from Employee;

**O/P:**

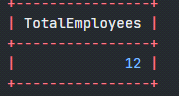


**3. Count Number of Employee Names**

SELECT COUNT(E\_name) AS TotalEmployees

FROM Employee;

O/P:



**4. Find the Maximum Age**

SELECT MAX(Age) AS MaxAge

FROM Employee;

O/P:



**5. Find the Minimum Age**

SELECT MIN(Age) AS MinAge

FROM Employee;

O/P:



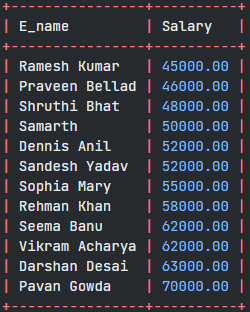
**6. Find Salaries of Employees in Ascending Order**

SELECT E\_name, Salary

FROM Employee

ORDER BY Salary ASC; //Also replace ASC by DESC for descending order

O/P:



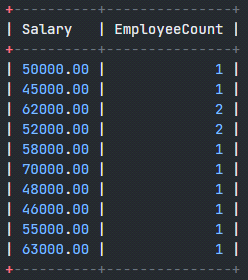
**7. Find Grouped Salaries of Employees**

SELECT Salary, COUNT(\*) AS EmployeeCount

FROM Employee

GROUP BY Salary;

O/P:



In these queries:

* **COUNT(E\_name)** counts the number of non-NULL values in the **E\_name** column.
* **MAX(Age)** finds the maximum age among the employees.
* **MIN(Age)** finds the minimum age among the employees.
* **ORDER BY Salary ASC** sorts the employees based on their salaries in ascending order.
* **GROUP BY Salary** groups employees by their salaries and counts the number of employees for each salary.

**Question 4**

Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old & new Salary.

**CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)**

**Solution:-**

**1. Create the CUSTOMERS Table**

CREATE TABLE CUSTOMERS (

ID INT PRIMARY KEY AUTO\_INCREMENT,

NAME VARCHAR (255),

AGE INT,

ADDRESS VARCHAR (255),

SALARY DECIMAL (10, 2)

);

**2. Create Trigger for INSERT Operation**

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 ;

**3. Create Trigger for UPDATE Operation**

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 ;

**4. Create Trigger for DELETE Operation**

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 ;

**5. Testing the Trigger:**

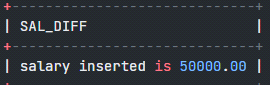
**[execute below two queries together]**

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

VALUES ('Shankara', 35, '123 Main St', 50000.00);

SELECT @my\_sal\_diff AS SAL\_DIFF;

O/P:



**[execute below two queries together]**

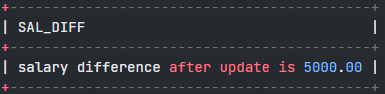
UPDATE CUSTOMERS

SET SALARY = 55000.00

WHERE ID = 1;

SELECT @my\_sal\_diff AS SAL\_DIFF;

O/P:



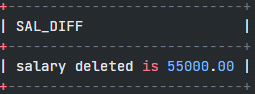
**[execute below two queries together]**

DELETE FROM CUSTOMERS

WHERE ID = 1;

SELECT @my\_sal\_diff AS SAL\_DIFF;

O/P:



**Question 5**

Create cursor for Employee table & extract the values from the table. Declare the variables,Open the cursor & extract the values from the cursor. Close the cursor.

Employee (E\_id, E\_name, Age, Salary)

**Solution:**

**1. Creating the Employee Table and insert few records**

CREATE TABLE Employee (

E\_id INT PRIMARY KEY AUTO\_INCREMENT,

E\_name VARCHAR(255),

Age INT,

Salary DECIMAL(10, 2)

);

INSERT INTO Employee (E\_id, E\_name, Age, Salary)

VALUES

(1, 'Samarth', 30, 50000.00),

(2, 'Ramesh Kumar', 25, 45000.00),

(3, 'Seema Banu', 35, 62000.00),

(4, 'Dennis Anil', 28, 52000.00),

(5, 'Rehman Khan', 32, 58000.00);

**2. Create a Stored Procedure with Cursor**

DELIMITER //

CREATE PROCEDURE fetch\_employee\_data()

BEGIN

**-- Declare variables to store cursor values**

DECLARE emp\_id INT;

DECLARE emp\_name VARCHAR(255);

DECLARE emp\_age INT;

DECLARE emp\_salary DECIMAL(10, 2);

**-- Declare a cursor for the Employee table**

DECLARE emp\_cursor CURSOR FOR

SELECT E\_id, E\_name, Age, Salary

FROM Employee;

**-- Declare a continue handler for the cursor**

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET @finished = 1;

**-- Open the cursor**

OPEN emp\_cursor;

**-- Initialize a variable to control cursor loop**

SET @finished = 0;

**-- Loop through the cursor results**

cursor\_loop: LOOP

**-- Fetch the next row from the cursor into variables**

FETCH emp\_cursor INTO emp\_id, emp\_name, emp\_age, emp\_salary;

**-- Check if no more rows to fetch**

IF @finished = 1 THEN

LEAVE cursor\_loop;

END IF;

**-- Output or process each row (for demonstration, print the values)**

SELECT CONCAT('Employee ID: ', emp\_id, ', Name: ', emp\_name, ', Age: ', emp\_age, ', Salary: ', emp\_salary) AS Employee\_Info;

END LOOP;

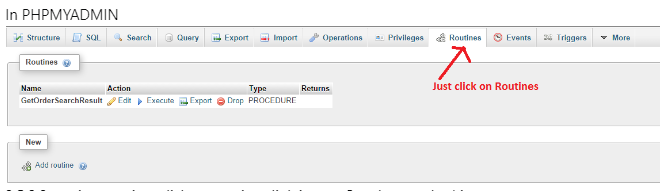
**-- Close the cursor**

CLOSE emp\_cursor;

END;//

DELIMITER ;

To check strored procedure, select the database (not the table) and then click on Routines\*\*\*\*\*\*\*



**3. Execute the Stored Procedure**

CALL 'fetch\_employee\_data'();

**Question 6**

Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N\_RollCall with the data available in the table O\_RollCall. If the data in the first table already exist in the second table, then that data should be skipped.

**Solution:**

To accomplish this task, use a stored procedure with a parameterized cursor to merge data from one table (N\_RollCall) into another table (O\_RollCall) while skipping existing data. Iterate through the records of N\_RollCall and insert them into O\_RollCall only if they do not already exist.

**1. Create the Tables**

First, create the N\_RollCall and O\_RollCall tables with similar structure:

solution:

**-- Create N\_RollCall table**

CREATE TABLE N\_RollCall (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(255),

birth\_date DATE

);

**-- Create O\_RollCall table with common data**

CREATE TABLE O\_RollCall (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(255),

birth\_date DATE

);

**2. Add Sample Records to both tables**

**-- Insert common data into O\_RollCall**

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

VALUES

(1, 'Shivanna', '1995-08-15'),

(3, 'Cheluva', '1990-12-10');

**-- Insert sample records into N\_RollCall**

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

VALUES

(1, 'Shivanna', '1995-08-15'), -- Common record with O\_RollCall

(2, 'Bhadramma', '1998-03-22'),

(3, 'Cheluva', '1990-12-10'), -- Common record with O\_RollCall

(4, 'Devendra', '2000-05-18'),

(5, 'Eshwar', '1997-09-03');

**3. Define the Stored Procedure**

Define the **merge\_rollcall\_data** stored procedure to merge records from N\_RollCall into O\_RollCall, skipping existing records:

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 cursor for N\_RollCall table**

DECLARE n\_cursor CURSOR FOR

SELECT student\_id, student\_name, birth\_date

FROM N\_RollCall;

**-- Declare handler for cursor**

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET done = TRUE;

**-- Open the cursor**

OPEN n\_cursor;

**-- Start looping through cursor results**

cursor\_loop: LOOP

**-- Fetch data from cursor into variables**

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

**-- Check if no more rows to fetch**

IF done THEN

LEAVE cursor\_loop;

END IF;

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

IF NOT EXISTS (

SELECT 1

FROM O\_RollCall

WHERE student\_id = n\_id

) THEN

**-- Insert the record into O\_RollCall**

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

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

END IF;

END LOOP;

**-- Close the cursor**

CLOSE n\_cursor;

END//

DELIMITER ;

**4. Execute the Stored Procedure**

CALL 'merge\_rollcall\_data'();

**5. Verify Records in O\_RollCall**

**-- Select all records from O\_RollCall**

SELECT \* FROM O\_RollCall;

