Arka Educational & Cultural Trust (Regd.)
Jain Institute of Technology, Davangere
(A Unit of Jain Group of Institutions, Bangalore)
(Affiliated to VTU, Belagavi | Approved by AICTE, New Delhi | Recognized by Government of Karnataka)

Department of Computer Science and Engineering
(Accredited by NBA, New Delhi, validity up to 30.06.2026)

Database Management System

Laboratory Manual

Sub Code: BCS 403 Semester : IV

Compiled by

Prof. G C Divya | Prof. Latharani T R

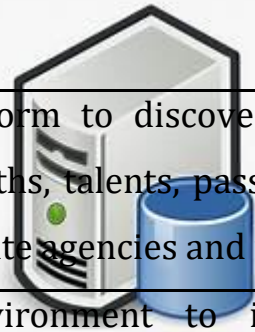
Vision & Mission of the Program

Vision of the Department:

To develop socially responsible computer engineers and entrepreneurs with strong academic excellence, technical backgrounds, research and innovation, intellectual skills and creativity to cater the needs of IT Industry and society by adopting professional ethics

Mission of the Department:

M1	To impart center of excellence by offering technical education and imbibing experiential learning skills to achieve teaching learning process.
M2	Providing a Platform to discover and engage research and innovation strengths, talents, passions through collaborations, government, private agencies and industries.
M3	Creating an environment to inculcate moral principles, professionalism and responsibilities towards the society.



POs	Program Outcomes
PO1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and computer science and engineering to the solution of complex engineering problems.
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex computer engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions:** Design solutions for complex computer engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern computer engineering and IT tools including prediction and modeling to complex computer engineering activities with an understanding of the limitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability:** Understand the impact of the professional computer engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the computer engineering practice.

PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex computer engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the computer engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PSOs	Program Specific Outcomes
PSO1	**Professional Skills:** The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer-based systems of varying complexity
PSO2	**Standard Practices:** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
PSO3	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

Sl. No	EXPERIMENT
1	Create a table called Employee & execute the following **Employee(EMPNO,ENAME,JOB, MANAGER_NO, SAL, COMMISSION)** 1. Create a user and grant all permissions to the user. 2. Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result. 3. Add primary key constraint and not null constraint to the employee table. 4. Insert null values to the employee table and verify the result.
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.
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 employeetable 3. Find the Maximum age from employee table. 4. Find the Minimum age from employeetable. 5. Find salaries of employee in Ascending Order. 6. Find grouped salaries of employees.
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)**
5	Create cursor for Employee table & extract the values from the table. Declare the variables ,Open the cursor & extrct the values from the cursor. Close the cursor. **Employee(E_id, E_name, Age, Salary)**
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.
7	Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations.

Experiment 1

Create a table called Employee & execute the following.

Employee (EMPNO, ENAME, JOB, MANAGER_NO, SAL, COMMISSION)

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

2. Insert the any three records in the employee table contain attributes EMPNO, ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result.

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

4. Insert null values to the employee table and verify the result.

Solution

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

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password123456';
GRANT ALL PRIVILEGES ON *.* TO 'new_user'@'localhost';
FLUSH PRIVILEGES;
```

2. **Create the Employee table and insert three records using rollback:**

```
CREATE TABLE Employee (
EMPNO INT,
ENAME VARCHAR(50),
JOB VARCHAR(50),
MANAGER_NO INT,
SAL DECIMAL(10, 2),
COMMISSION DECIMAL(10, 2)
);
```

desc Employee

Field	Type	Null	Key	Default
EMPNO	int(11)	Yes		*NULL*
ENAME	varchar(50)	Yes		*NULL*
JOB	varchar(50)	Yes		*NULL*
MANAGER_NO	int(11)	YES		*NULL*
SAL	decimal(10,2)	NO		*NULL*
COMMISSION	decimal(10,2)	YES		*NULL*

Insert three records

insert into Employee values (1, 'John Doe', 'Manager', NULL, 5000.00, 1000.00),

insert into Employee values (2, 'Jane Smith', 'Developer', 1, 4000.00, NULL),

insert into Employee values (3, 'Alice Johnson', 'Salesperson', 1, 3000.00, 500.00);

EMPNO	ENAME	JOB	MANAGER_NO	SAL	COMMISSION
1	John Doe	Manager	*NULL*	5000.00	1000.00
2	Jane Smith	Developer	1	4000.00	*NULL*
3	Alice Johnson	Salesperson	1	3000.00	500.00

ROLLBACK;

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

Adding primary key constraint

ALTER TABLE Employee ADD PRIMARY KEY (EMPNO);

Field	Type	Null	Key	Default
EMPNO	int(11)	NO	PRI	*NULL*
ENAME	varchar(50)	YES		*NULL*
JOB	varchar(50)	YES		*NULL*
MANAGER_NO	int(11)	YES		*NULL*
SAL	decimal(10,2)	YES		*NULL*
COMMISSION	decimal(10,2)	YES		*NULL*

Add not null constraint to the Employee table

```
ALTER TABLE Employee
MODIFY ENAME VARCHAR(50) NOT NULL,
MODIFY JOB VARCHAR(50) NOT NULL,
MODIFY SAL DECIMAL(10,2) NOT NULL;
```

Field	Type	Null	Key	Default
EMPNO	int(11)	NO	PRI	*NULL*
ENAME	varchar(50)	NO		*NULL*
JOB	varchar(50)	NO		*NULL*
MANAGER_NO	int(11)	YES		*NULL*
SAL	decimal(10,2)	NO		*NULL*
COMMISSION	decimal(10,2)	YES		*NULL*

4. **Attempt to insert null values into the Employee table:**

```
INSERT INTO Employee values(1,NULL,'intern',2,NULL,NULL);
```

Output:

Column 'ENAME' cannot be null

Experiment 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 (
   EMPNO INT,
   ENAME VARCHAR(50),
   JOB VARCHAR(50),
   MGR INT,
   SAL DECIMAL(10,2)
);
desc Employee
```

Field	Type	Null	Key	Default
Empno	varchar(50)	YES		*NULL*
ENAME	varchar(50)	YES		*NULL*
JOB	varchar(50)	YES		*NULL*
MGR	int(11)	YES		*NULL*
SAL	decimal(10,2)	YES		*NULL*

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

```
ALTER TABLE Employee
ADD COMMISSION DECIMAL(10,2);
```

Field	Type	Null	Key	Default
Empno	varchar(50)	YES		*NULL*
ENAME	varchar(50)	YES		*NULL*
JOB	varchar(50)	YES		*NULL*
MGR	int(11)	YES		*NULL*
SAL	decimal(10,2)	YES		*NULL*
COMMISSION	decimal(10,2)	YES		*NULL*

2. Insert any five records into the table.

INSERT INTO Employee (EMPNO, ENAME, JOB, MGR, SAL) VALUES (101, 'John Doe', 'Manager', NULL, 5000.00), (102, 'Jane Smith', 'Assistant', 101, 3000.00), (103, 'Michael Johnson', 'Clerk', 102, 2000.00), (104, 'Emily Brown', 'Manager', NULL, 5500.00), (105, 'Alex Clark', 'Assistant', 104, 3200.00);

EMPNO	ENAME	JOB	MGR	SAL	COMMISSION
101	John Doe	Manager	*NULL*	5000.00	*NULL*
102	Jane Smith	Assistant	101	3000.00	*NULL*
103	Michael Johnson	Clerk	102	2000.00	*NULL*
104	Emily Brown	Manager	*NULL*	5500.00	*NULL*
105	Alex Clark	Assistant	104	3200.00	*NULL*

3. Update the column details of job

```
UPDATE Employee
SET JOB = 'Supervisor'
WHERE EMPNO = 105;
```

EMPNO	ENAME	JOB	MGR	SAL	COMMISSION
101	John Doe	Manager	*NULL*	5000.00	*NULL*
102	Jane Smith	Assistant	101	3000.00	*NULL*
103	Michael Johnson	Clerk	102	2000.00	*NULL*
104	Emily Brown	Manager	*NULL*	5500.00	*NULL*
105	Alex Clark	Supervisor	104	3200.00	*NULL*

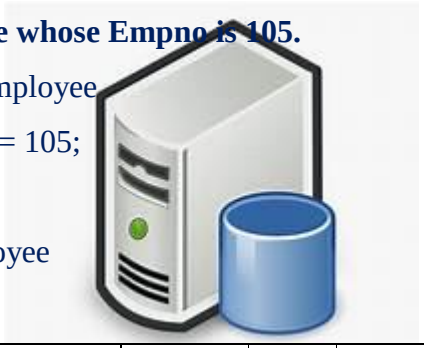
4. Rename the column of Employ table using alter command.

```
ALTER TABLE Employee
CHANGE COLUMN Empno Empno2 VARCHAR(50);
```

Empno2	ENAME	JOB	MGR	SAL	COMMISSION
101	John Doe	Manager	*NULL*	5000.00	*NULL*
102	Jane Smith	Assistant	101	3000.00	*NULL*
103	Michael Johnson	Clerk	102	2000.00	*NULL*
104	Emily Brown	Manager	*NULL*	5500.00	*NULL*
105	Alex Clark	Supervisor	104	3200.00	*NULL*

5. **Delete the employee whose Empno is 105.**

```
DELETE FROM Employee
WHERE EMPNO2 = 105;
```



```
Select * from Employee
```

Empno2	ENAME	JOB	MGR	SAL	COMMISSION
101	John Doe	Manager	*NULL*	5000.00	*NULL*
102	Jane Smith	Assistant	101	3000.00	*NULL*
103	Michael Johnson	Clerk	102	2000.00	*NULL*
104	Emily Brown	Manager	*NULL*	5500.00	*NULL*

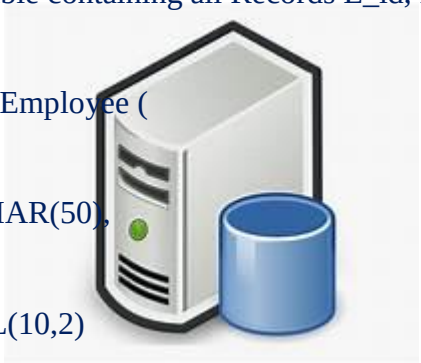
Experiment 3

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 employeetable**
3. **Find the Maximum age from employee table.**
4. **Find the Minimum age from employeetable.**
5. **Find salaries of employee in Ascending Order.**
6. **Find grouped salaries of employees.**

Solution

1. Create Employee table containing all Records E_id, E_name, Age, Salary.



```
CREATE TABLE Employee (
  E_id INT,
  E_name VARCHAR(50),
 Age INT,
 Salary DECIMAL(10,2)
);
```

Field	Type	Null	Key	Default
E_id	int(11)	YES		*NULL*
E_name	varchar(50)	YES		*NULL*
Age	int(11)	YES		*NULL*
Salary	decimal(10,2)	YES		*NULL*

insert into Employee(E_id, E_name, Age, Salary) values (1, 'John',35,40000), (2,'Alice',32,35700), (3,'Bob',28,28000), (4,'Charles',30,32000);

E_id	E_name	Age	Salary
1	John	35	40000.00
2	Alice	32	35700.00
3	Bob	28	28000.00
4	Charles	30	32000.00

2. Count number of employee names from employee table

```
SELECT COUNT(E_name) AS total_employees
FROM Employee;
```

total_employees
4

3. Find the Maximum age from employee table.

```
SELECT MAX(Age) AS max_age
FROM Employee;
```

max_age
35

4. Find the Minimum age from employeetable.

```
SELECT MIN(Age) AS min_age
FROM Employee;
```

min_age
28

5. Find salaries of employee in Ascending Order.

```
SELECT Salary
FROM Employee
ORDER BY Salary ASC;
```

Salary_1
28000.00
32000.00
35700.00
40000.00

6. Find grouped salaries of employees.

insert into Employee values (6, 'Peter',38,40000);

E_id	E_name	Age	Salary
1	John	35	40000.00
2	Alice	32	35700.00
3	Bob	28	28000.00
4	Charles	30	32000.00
6	peter	38	40000.00

SELECT Salary, COUNT(*) AS num_employees
FROM Employee
GROUP BY Salary;

Salary	num_employees
28000.00	1
32000.00	1
35700.00	1
40000.00	2

Experiment 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

```
CREATE TABLE CUSTOMERS (
    ID INT PRIMARY KEY,
    NAME VARCHAR(50),
    AGE INT,
    ADDRESS VARCHAR(100),
    SALARY DECIMAL(10, 2)
);
```

Field	Type	Null	Key	Default
ID	int(11)	NO	PRI	*NULL*
NAME	varchar(50)	YES		*NULL*
AGE	int(11)	YES		*NULL*
ADDRESS	varchar(100)	YES		*NULL*
SALARY	decimal(10,2)	YES		*NULL*

Step 1: Create an Audit Table

First, create an audit table to store the salary differences and other relevant information.

```
CREATE TABLE customers_audit (
    id INT AUTO_INCREMENT PRIMARY KEY,
    operation VARCHAR(10),
    customer_id INT,
    old_salary DECIMAL(10,2),
    new_salary DECIMAL(10,2),
    salary_difference DECIMAL(10,2),
```

```
    operation_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Step 2: Create Triggers

Next, create the triggers for INSERT, UPDATE, and DELETE operations to log changes to the customers_audit table.

```
DELIMITER //

CREATE TRIGGER trg_customers_insert
AFTER INSERT ON CUSTOMERS
FOR EACH ROW
BEGIN
    INSERT INTO customers_audit (operation, customer_id, new_salary)
    VALUES ('INSERT', NEW.ID, NEW.SALARY);
END//

DELIMITER ;
```

Step 3: Trigger for UPDATE

```
DELIMITER //

CREATE TRIGGER trg_customers_update
AFTER UPDATE ON CUSTOMERS
FOR EACH ROW
BEGIN
    DECLARE v_salary_diff DECIMAL(10,2);
    SET v_salary_diff = NEW.SALARY - OLD.SALARY;
    INSERT INTO customers_audit (operation, customer_id, old_salary, new_salary,
      salary_difference)
    VALUES ('UPDATE', NEW.ID, OLD.SALARY, NEW.SALARY, v_salary_diff);
END//

DELIMITER ;
```

Step 3: Trigger for Delete

```
DELIMITER //

CREATE TRIGGER trg_customers_delete
AFTER DELETE ON CUSTOMERS
FOR EACH ROW
BEGIN
   INSERT INTO customers_audit (operation, customer_id, old_salary)
   VALUES ('DELETE', OLD.ID, OLD.SALARY);
END//

DELIMITER ;
```

Inserting values to customers table

insert into CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) values (1,'Peter',35, '2nd cross vijaynagar', 100000), (2,'Jhew',42, 'Dollars Colony', 92000), (3,'Pinto',26, 'RR Nagar', 45000);

Output:

3 rows inserted

SELECT * FROM `customers`;

NAME	ID	AGE	ADDRESS	SALARY
Peter	1	35	2nd cross vijaynagar	100000.00
Jhew	2	42	Dollars Colony	92000.00
Pinto	3	26	RR Nagar	45000.00

Checking the trigger on insert operation, created by displaying the customers_audit table

SELECT * FROM customers_audit

ID	OPERATION	CUSTOMER_ID	OLD_SALARY	NEW_SALARY	SALARY_DIFFERENCE	OPERATION_TIME
1	INSERT	1	NULL	100000.00	NULL	2024-05-20 14:20:09
2	INSERT	2	NULL	92000.00	NULL	2024-05-20 14:20:09
3	INSERT	3	NULL	45000.00	NULL	2024-05-20 14:20:09

Updating the salary of an employee with id 2

update customers set salary = 150000 where id = 2;

SELECT * FROM customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Peter	35	2nd cross vijaynagar	100000.00
2	Jhew	42	Dollars Colony	92000.00
3	Pinto	26	RR Nagar	45000.00

Checking the trigger on update operation, created by displaying the customers_audit table

SELECT * FROM customers_audit;

id	operation	customer_id	old_salary	new_salary	salary_difference	operation_time
1	INSERT	1	NULL	100000.00	NULL	2024-05-20 14:20:09
2	INSERT	2	NULL	92000.00	NULL	2024-05-20 14:20:09
3	INSERT	3	NULL	45000.00	NULL	2024-05-20 14:20:09
4	UPDATE	2	92000.00	150000.00	58000.00	2024-05-20 14:57:56

Delete a row with id 1

delete from customers where id=1

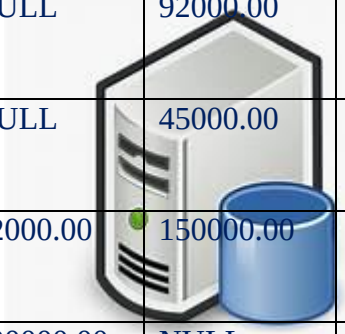
Output:

1 row deleted. (Query took 0.1224 seconds.)

Checking the trigger on delete operation, created by displaying the customers_audit table

SELECT * FROM `customers_audit`

ID	OPERATION	CUSTOMER_ID	OLD_SALARY	NEW_SALARY	SALARY_DIFFERENCE	OPERATION_TIME
1	INSERT	1	NULL	100000.00	NULL	2024-05-20 14:20:09
2	INSERT	2	NULL	92000.00	NULL	2024-05-20 14:20:09
3	INSERT	3	NULL	45000.00	NULL	2024-05-20 14:20:09
4	UPDATE	2	92000.00	150000.00	58000.00	2024-05-20 14:57:56
5	DELETE	1	100000.00	NULL	NULL	2024-05-20 15:03:03



Experiment 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

Create the Employee table

```
CREATE TABLE Employee (
    E_id INT PRIMARY KEY,
    E_name VARCHAR(255),
    Age INT,
    Salary DECIMAL(10, 2)
);
```

desc employee;

E_id	int(11)	NO	PRI	NULL
E_name	varchar(100)	YES		NULL
Age	int(11)	YES		NULL
Salary	decimal(10,2)	YES		NULL

Insert some sample data

```
INSERT INTO Employee (E_id, E_name, Age, Salary)
VALUES (1, 'Alice', 30, 50000),(2, 'Bob', 28, 48000), (3, 'Charlie', 25, 45000);
```

Procedure to create routine:

```
DELIMITER //
CREATE PROCEDURE your_procedure_name()
BEGIN
    -- Procedure logic here
```

```
    SELECT * FROM your_table_name;
END //

DELIMITER ;
```

Edit in procedure code and execute the procedure

Create the stored procedure to work with the cursor:

```
BEGIN
    -- Declare variables to hold employee data
    DECLARE v_E_id INT;
    DECLARE v_E_name VARCHAR(100);
    DECLARE v_Age INT;
    DECLARE v_Salary DECIMAL(10,2);

    -- Variable to control cursor loop
    DECLARE done INT DEFAULT 0;

    -- Declare a cursor for the Employee table
    DECLARE employee_cursor CURSOR FOR
    SELECT E_id, E_name, Age, Salary FROM Employee;

    -- Declare a handler to handle the end of the cursor
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- Open the cursor
    OPEN employee_cursor;

    -- Fetch the data from the cursor
    read_loop: LOOP
        FETCH employee_cursor INTO v_E_id, v_E_name, v_Age, v_Salary;
        IF done THEN
            LEAVE read_loop;
        END IF;
```

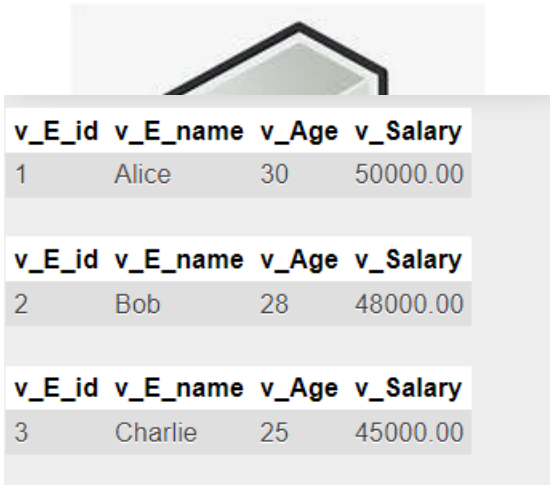
```
    -- Do something with the fetched data
    -- Here we will just select it to display
    SELECT v_E_id, v_E_name, v_Age, v_Salary;
  END LOOP read_loop;

  -- Close the cursor
  CLOSE employee_cursor;
END
```

Call the stored procedure:

```
CALL ExtractEmployeeData();
```

Output:



v_E_id	v_E_name	v_Age	v_Salary
1	Alice	30	50000.00

v_E_id	v_E_name	v_Age	v_Salary
2	Bob	28	48000.00

v_E_id	v_E_name	v_Age	v_Salary
3	Charlie	25	45000.00

Experiment 5

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

Creation of Tables:

```
CREATE TABLE N_RollCall (
    student_id INT,
    rollcall_date DATE,
    status VARCHAR(10),
    PRIMARY KEY (student_id, rollcall_date)
);

CREATE TABLE O_RollCall (
    student_id INT,
    rollcall_date DATE,
    status VARCHAR(10),
    PRIMARY KEY (student_id, rollcall_date)
);
```

Insertion of values

```
-- Insert sample data into N_RollCall
INSERT INTO N_RollCall (student_id, rollcall_date, status) VALUES
(1, '2024-06-25', 'Present'),
(2, '2024-06-25', 'Absent'),
(3, '2024-06-25', 'Present');

-- Insert sample data into O_RollCall
INSERT INTO O_RollCall (student_id, rollcall_date, status) VALUES
(1, '2024-06-25', 'Present');
```

Creation of Procedure

```
DELIMITER $$

CREATE PROCEDURE merge_rollcall()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_student_id INT;
    DECLARE v_rollcall_date DATE;
    DECLARE v_status VARCHAR(10);

    -- Declare a cursor to select data from N_RollCall
    DECLARE cur_n_rollcall CURSOR FOR
        SELECT student_id, rollcall_date, status FROM N_RollCall;

    -- Declare a continue handler for cursor
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    -- Open the cursor
    OPEN cur_n_rollcall;

    -- Loop through each row in N_RollCall
    read_loop: LOOP
        FETCH cur_n_rollcall INTO v_student_id, v_rollcall_date, v_status;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Check if the record already exists in O_RollCall
        IF NOT EXISTS (
            SELECT 1 FROM O_RollCall
            WHERE student_id = v_student_id
            AND rollcall_date = v_rollcall_date
        ) THEN
```

```
            -- Insert the record into O_RollCall
            INSERT INTO O_RollCall (student_id, rollcall_date, status)
            VALUES (v_student_id, v_rollcall_date, v_status);
        END IF;
    END LOOP;

    -- Close the cursor
    CLOSE cur_n_rollcall;
END$$

DELIMITER ;
```

Experiment 6

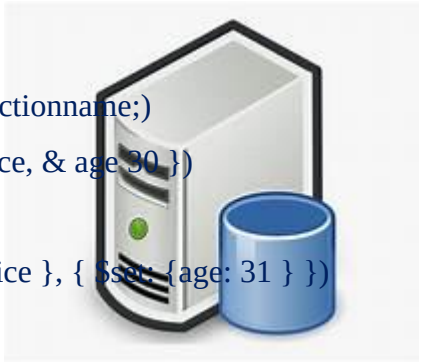
Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations

- https://www.mongodb.com/try/download/community
- https://www.mongodb.com/try/download/shell
- Set Environmental Path

Mongod –version

Mongosh

Use dbname



db.createCollection(collectionname;)

db.sana.insert({name :Alice, & age 30 })

db.sana.find()

db.sana.update({name:Alice }, { $set: {age: 31 } })