LAB MANUAL

# DATABASE MANAGEMENT SYSTEMS

**BCS403**

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**Prepared By**
**Rakesh S Raj**
**Assistant Professor**
**Department of ISE, AIT**

**Adichunchanagiri Institute of Technology, Chickmaglur**

# DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

## DEPARTMENT VISION

To be recognized as a center of excellence in information technology and allied areas with quality learning and research environment

## DEPARTMENT MISSION

- Provide intellectual & professional leadership in ethical and social areas pertaining to information in contemporary society.
- Advancing the state of knowledge of information studies through research and development.
- Providing a platform to discuss cutting edge technologies.

## DEPARTMENT PROGRAM EDUCATIONAL OBJECTIVE

- **PEO1:** Graduates will be able to analyze, design and provide solutions to the problems in the field of information science and engineering
- **PEO2:** Graduates will be able to exhibit the quality of working in team and build leadership quality
- **PEO3:** Graduates will be able to learn new technologies that will be helpful in their professional success
- **PEO4:** To train students with good breadth of knowledge in core areas of Information

**PRACTICAL COMPONENT OF IPCC** *(May cover all / major modules)*

| Sl.NO | Experiments |
|---|---|
| 1 | Create a table called Employee & execute the following.<br>**Employee(EMPNO,ENAME,JOB, MANAGER_NO, SAL, COMMISSION)**<br>1. Create a user and grant all permissions to theuser.<br>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.<br>3. Add primary key constraint and not null constraint to the employee table.<br>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.<br>1. Add a column commission with domain to the Employeetable.<br>2. Insert any five records into the table.<br>3. Update the column details of job<br>4. Rename the column of Employ table using alter command.<br>5. Delete the employee whose Empno is 105. |
| 3 | Queries using aggregate functions(COUNT,AVG,MIN,MAX,SUM),Group by,Orderby.<br>**Employee(E_id, E_name, Age, Salary)**<br>1. Create Employee table containing all Records E_id, E_name, Age, Salary.<br>2. Count number of employee names from employeetable<br>3. Find the Maximum age from employee table.<br>4. Find the Minimum age from employeetable.<br>5. Find salaries of employee in Ascending Order.<br>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.<br>**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.<br>**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. |

**Course outcomes (Course Skill Set):**

At the end of the course, the student will be able to:

- Describe the basic elements of a relational database management system
- Design entity relationship for the given scenario.
- Apply various Structured Query Language (SQL) statements for database manipulation.
- Analyse various normalization forms for the given application.
- Develop database applications for the given real world problem.
- Understand the concepts related to NoSQL databases.

**Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum

1. Create a table called Employee & execute the following.

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

a) Create a user and grant all permissions to the user.
b) Insert the any three records in the employee table contains attributes EMPNO, ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result.
c) Add primary key constraint and not null constraint to the employee table.
d) Insert null values to the employee table and verify the result.

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

connect

username: system

password: tiger

SyntaX:

create user USERNAME identified by PASSWORD;

create user ait identified by ait123;

Syntax: grant connect to USERNAME;

grant connect to ait.

GRANT CONNECT, RESOURCE, DBA TO ait;

exit

connect

username: ait

password: ait123

**b. Insert the any three records in the employee table contains attributes**

```
create table employee2(
empno varchar(6),
ename varchar(10),
job char(10),
managerno varchar(6),
salary float,
commission varchar(5));

insert into employee2 values('emp101','Sachin','Developer','man201',65000,'25%');
insert into employee2 values('emp102','Sourav','Testing','man201',60000,'15%');
insert into employee2 values('em103','Rahul','Developer','man201',75000,'10%');
insert into employee2 values('em104','','Developer','man201',65000,'25%');

select *from employee2;

rollback
```

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

```
ALTER TABLE employee2
ADD PRIMARY KEY (empno);
desc employee2;
ALTER TABLE employee2
MODIFY ename int NOT NULL;
desc employee2;
```

**d. Insert null values to the employee table and verify the result.**

```
insert into employee2 values(' ','Rahul','Developer','man201',75000,'10%');
insert into employee2 values('em104',' ','Developer','man201',65000,'25%');
```

2. **Create a table called Employee that contain attributes EMPNO, ENAME,JOB, MGR,SAL &**

**execute the following.**

a) **Add a column commission with domain to the Employee table.**
b) **Insert any five records into the table.**
c) **Update the column details of job**
d) **Rename the column of Employ table using alter command.**
e) **Delete the employee whose Empno is 105.**

```
create table employee3(
empno varchar(6) primary key,
ename varchar(10) not null,
job char(10),
managerno varchar(6),
salary float);

desc employee3;
```

a. **Syntax to add attribute/-column**

```
ALTER TABLE table_name
ADD column_name datatype;

ALTER TABLE employee3
ADD commission varchar(5);
```

b. **Insert any five records into the table.**

```
insert into employee3 values('emp101','Sachin','Developer','man201',65000,'10%');
insert into employee3 values('emp102','Sourav','Testing','man201',60000,'15%');
insert into employee3 values('emp103','Rahul','Developer','man201',75000,'20%');
insert into employee3 values('emp104','Virat','Developer','man201',65000,'25%');
insert into employee3 values('emp105','Rohit','Developer','man201',65000,'30%');
```

**c. Update the column details of job**

UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

update employee3

set job='Dev'

where job='Developer';

**d. To rename column name**

ALTER TABLE table_name

RENAME COLUMN old_name to new_name;

alter table employee3

rename column ename to emp_name;

To Rename table name

RENAME old_table _name To new_table_name ;

**e. Delete the employee whose empno is 105**

delete from emp3

where empno='emp105';

create table employee2(

empno varchar(6),

ename varchar(10),

job char(10),

managerno varchar(6),

```
salary float,
commission varchar(5));

insert into employee2 values('emp101','Sachin','Developer','man201',65000,'25%');
insert into employee2 values('emp102','Sourav','Testing','man201',60000,'15%');
insert into employee2 values('em103','Rahul','Developer','man201',75000,'10%');
insert into employee2 values('em104','','Developer','man201',65000,'25%');

select *from employee2;

rollback

ALTER TABLE employee2
ADD PRIMARY KEY (empno);

desc employee2;

ALTER TABLE employee2
MODIFY ename int NOT NULL;

desc employee2;
```

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

**Employee(E_id, E_name, Age, Salary)**

a) **Create Employee table containing all Records E_id, E_name, Age, Salary.**
b) **Count number of employee names from employeetable**
c) **Find the Maximum age from employee table.**
d) **Find the Minimum age from employeetable.**
e) **Find salaries of employee in Ascending Order.**
f) **Find grouped salaries of employees.**

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

```
create table employee3(
empno int primary key,
ename char(10),
age int,
salary float);

insert into employee3 values(101,'Virat',35,100000);
insert into employee3 values(102,'Rahul',51,100000);
insert into employee3 values(103,'Sachin',52,100000);
insert into employee3 values(104,'Rohit',38,80000);
insert into employee3 values(105,'Dhoni',45,80000);
insert into employee3 values(106,'Gill',35,80000);
```

b. **Count number of employee names from employeetable**

```
select count(*) as total_employees
from employee3;
```

c. **Find the Maximum age from employee table.**

```
select max(age) as Maximum_age
from employee3;
```

**d. Find the Minimum age from employeetable.**

```
select min(age) as Maximum_age
from employee3;
```

**e. Find salaries of employee in Ascending Order.**

```
select *
from employee3
order by salary desc;
```

**f. Find grouped salaries of employees.**

```
select salary,count(*)
from employee3
where salary>70000
group by salary;
```

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**

1.Create table

---

```
create table customer(
id int primary key,
name char(15),
age int,
address varchar(20),
salary number(5));
```

2.To enable dbms_output.put_line() function

---

```
SET SERVEROUTPUT ON
```

3.Create row level trigger

---

```
CREATE OR REPLACE TRIGGER salary_changes
  BEFORE DELETE OR INSERT OR UPDATE OF SALARY ON customer
  FOR EACH ROW
WHEN (new.id > 0)
DECLARE
    sal_diff number;
BEGIN
```

```
    sal_diff  := :new.salary  - :old.salary;
    dbms_output.put_line('Old salary: ' || :old.salary);
    dbms_output.put_line('  New salary: ' || :new.salary);
    dbms_output.put_line('  Difference ' || sal_diff);
END;
/
```

4.Create statement level Trigger

---

```
create or replace trigger cust1 before update on customer
begin
dbms_output.put_line('record updated');
end;
/
```

5.Insertion of values

---

```
insert into customer values(201,'Ramesh',30,'Vijaypura,Ckm',60000);
insert into customer values(202,'Sumesh',25,'KHB,Mysore',70000);
insert into customer values(203,'Bharathi',30,'Vijaypura,Ckm',60000);
insert into customer values(204,'Rama',30,'Vijaypura,Ckm',60000);
```

6.Updating the salary

---

```
UPDATE customer
  SET SALARY=salary+15000
```

WHERE id=201;

7. Deleting a row

---

delete customer where id=201;

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**

```
create table emp5(empid int primary key,ename char(15),age int,salary number(5));

insert into emp5 values(101,'Suman Shekar',28,10000);
insert into emp5 values(102,'Prashanth',30,10000);
insert into emp5 values(103,'Rahul',22,10000);
insert into emp5 values(104,'Shrinidhi',26,10000);
insert into emp5 values(105,'Nikhil',30,10000);


DECLARE
  CURSOR c1 IS SELECT * FROM   emp5;
  cempid emp5.empid%TYPE;
  cempname emp5.ename%TYPE;
  cage emp5.age%TYPE;
  csalary emp5.salary%TYPE;


BEGIN
  OPEN c1;
  LOOP
    FETCH c1 INTO  cempid,cempname,cage,csalary;
    EXIT WHEN c1%NOTFOUND;
    dbms_output.Put_line(cempid||' '||cempname||' '||cage||' '||csalary);
  END LOOP;
  CLOSE c1;
END;
```

/

set pagesize 200

set serveroutput on

**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.**

```
CREATE DATABASE ROLLCALL;

USE ROLLCALL;

-- 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
);

-- Insert common data into O_RollCall
INSERT INTO O_RollCall (student_id, student_name, birth_date)
VALUES
    (1, 'Sachin', '1995-08-15'),
    (3, 'Rahul', '1990-12-10');

-- Insert sample records into N_RollCall
INSERT INTO N_RollCall (student_id, student_name, birth_date)
```

```
VALUES
    (1, 'Sachin', '1995-08-15'),  -- Common record with O_RollCall
    (2, 'Virat', '1998-03-22'),
    (3, 'Rahul', '1990-12-10'), -- Common record with O_RollCall
    (4, 'Rohit', '2000-05-18'),
    (5, 'Sourav', '1997-09-03');

-- merge_rollcall_data stored procedure
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 ;


--execute the merge_rollcall_data stored procedure
CALL merge_rollcall_data();
```

```
-- Select all records from O_RollCall
SELECT * FROM O_RollCall;
```

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

```
use bookDB

db.createCollection("books")

db.books.insertMany([
  {
    title: "DBMS",
    author: "Elmasri and Navathe",
    category: "SQL Programming",
    year: 2008
  },
  {
    title: "JavaScript",
    author: "Robert Sebesta",
    category: "Full Stack Programming",
    year: 2008
  },
  {
    title: "Python",
    author: "Mark Lutz",
    category: "Programming",
    year: 1994
  },
  {
    title: "Introduction to Algorithms",
    author: "Thomas H. Cormen",
    category: "Algorithms",
```

```
    year: 1990
  },
  {
    title: "Python Crash Course",
    author: "Eric Matthes",
    category: "Python",
    year: 2015
  }
])


// Display all documents in the 'ProgrammingBooks' collection
db.books.find().pretty()

db.books.insertOne({
  title: "The Pragmatic Programmer: Your Journey to Mastery",
  author: "David Thomas, Andrew Hunt",
  category: "Software Development",
  year: 1999
})

db.ProgrammingBooks.find().pretty()

//find books published after the year 2000
db.books.find({ year: { $gt: 2000 } }).pretty()

//change the author of a book
db.books.updateOne(
  { title: "Javascript" },
  { $set: { author: "Robert C. Martin" } }
```

```
)

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

//update multiple books
db.ProgrammingBooks.updateMany(
  { year: { $lt: 2010 } },
  { $set: { category: "Classic Programming Books" } }
)

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

//Delete a Single Document
db.ProgrammingBooks.deleteOne({ title: "JavaScript" })

//delete multiple documents
db.ProgrammingBooks.deleteMany({ year: { $lt: 1995 } })

//delete multiple documents
db.ProgrammingBooks.deleteMany({})

//delete collection
db.ProgrammingBooks.drop()
```