Bachelor of Science
(B.Sc.- PCM)

DATABASE MANAGEMENT SYSTEM LAB
(DBSPDS201T24)

Self-Learning Material
(SEM II)

Jaipur National University
Centre for Distance and Online Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
&
NAAC A+ Accredited

TABLE OF CONTENTS

Course Introduction	i
Unit 1 Create tables with relevant foreign key constraints and populate with data	1
Unit 2 Populating the Tables with Data	2
Unit 3 Display the ssn, lname, fname, and address of employees who work in department number	4
Unit 4 Retrieve the name and salary of every employee from the employee table	4
Unit 5 Retrieve the ssn and department name for all employees	5
Unit 6 Retrieve the department number, the number of employees in each department, and their average salary.	5
Unit 7 Create and populate the table named “Employee” with data	7
Unit 8 Perform the following queries on the database.	8

EXPERT COMMITTEE

Prof. Sunil Gupta
(Dept. of Computer and Systems Sciences)
JNU, Jaipur

Dr. Shalini Rajawat
(Dept. of Computer and Systems Sciences)
JNU, Jaipur

COURSE COORDINATOR

Shish Kumar Dubey
(Dept. of Computer and Systems Sciences)
JNU, Jaipur

UNIT PREPARATION

Unit Writer(s)	Assisting & Proofreading	Unit Editor
Mr. Ram Lal Yadav Dept. of Computer and Systems Sciences, JNU, Jaipur (Unit 1-8)	Mr. Satender Singh (Dept. of Computer and Systems Sciences) JNU, Jaipur	Ms. Rachana Yadav (Dept. of Computer and Systems Sciences) JNU, Jaipur

Secretarial Assistance
Mr. Mukesh Sharma

COURSE INTRODUCTION

Welcome to the DBMS Lab course! This lab-based course is designed to provide you with hands-on experience in working with databases. You'll apply the concepts learned in your DBMS theory classes to practical scenarios, gaining valuable skills in creating, managing, and querying databases.

In this course, you'll work on various lab exercises and projects that involve designing database schemas, writing SQL queries, and performing data manipulation tasks. You'll also get the opportunity to experiment with different database management systems, exploring their features and capabilities.

By the end of the lab, you will have a practical understanding of how to implement and manage databases effectively, preparing you for real-world applications and challenges in the field of database management.

Course Outcomes:

At the completion of the course, a student will be able to:

1. Design and create database schemas based on practical requirements.
2. Write and execute SQL queries for data retrieval, insertion, update, and deletion.
3. Implement database constraints and manage data integrity.
4. Use database management tools to create, maintain, and optimize databases.
5. Perform data manipulation and analysis using various database functions and procedures.
6. Work with different database management systems to understand their features and capabilities.
7. Troubleshoot and resolve common database-related issues through practical problem-solving.
8. Apply practical knowledge from DBMS concepts to real-world database projects and scenarios.

Acknowledgements:

The content we have utilized is solely educational in nature. The copyright proprietors of the materials reproduced in this book have been tracked down as much as possible. The editors apologize for any violation that may have happened, and they will be happy to rectify any such material in later versions of this book.

1. Create tables with relevant foreign key constraints and populate with data.

1) Create the employee table

"CREATETABLE employee (
fnameVARCHAR(50) NOTNULL,
minitCHAR(1),
lnameVARCHAR(50) NOTNULL,
ssnCHAR(9) PRIMARY KEY,
bdateDATE,
address VARCHAR(100),
sex CHAR(1),
salary DECIMAL(10, 2),
superssnCHAR(9),
dnoINT,
FOREIGN KEY (superssn) REFERENCES employee(ssn),
FOREIGN KEY (dno) REFERENCES department(dnumber)
);"

2) Create the department table

"CREATETABLE department (
dnameVARCHAR(50) NOTNULL,
dnumberINTPRIMARY KEY,
mgrssnCHAR(9),
mgrstartdateDATE,
FOREIGN KEY (mgrssn) REFERENCES employee(ssn)
);"

3) Create the dept_locations table

"CREATETABLEdept_locations (
dnumberINT,
dlocationVARCHAR(50) NOTNULL,
FOREIGN KEY (dnumber) REFERENCES department(dnumber),
PRIMARY KEY (dnumber, dlocation)
);"

4) Create the project table

“CREATETABLE project (
pnameVARCHAR(50) NOTNULL,
pnumberINTPRIMARY KEY,
plocationVARCHAR(50),
dnumINT,
FOREIGN KEY (dnum) REFERENCES department(dnumber)
);”

5) Create the dependent table

“CREATETABLE dependent (
essnCHAR(9),
dependent_nameVARCHAR(50),
sex CHAR(1),
bdateDATE,
relationship VARCHAR(50),
FOREIGN KEY (essn) REFERENCES employee(ssn),
PRIMARY KEY (essn, dependent_name)
);”

2. Populating the Tables with Data:

a) Populate the department table

“INSERTINTO department (dname, dnumber, mgrssn, mgrstartdate) VALUES
('Research', 1, '123456789', '2020-01-01'),
('Administration', 2, '234567890', '2019-03-15'),
('Sales', 3, '345678901', '2021-06-01');”

b) Populate the employee table

“INSERTINTO employee (fname, minit, lname, ssn, bdate, address, sex, salary, superssn, dno) VALUES
('John', 'A', 'Doe', '123456789', '1980-01-01', '123 Elm St', 'M', 60000.00, NULL, 1),
('Jane', 'B', 'Smith', '234567890', '1985-02-14', '456 Oak St', 'F', 55000.00, '123456789', 2),

('Alice', 'C', 'Johnson', '345678901', '1990-05-21', '789 Pine St', 'F', 75000.00, '234567890', 3),
('Bob', 'D', 'Brown', '456789012', '1982-08-30', '101 Maple St', 'M', 50000.00, '345678901', 1);"

c) Populate the dept_locations table

"INSERTINTOdept_locations (dnumber, dlocation) VALUES
(1, 'New York'),
(2, 'Los Angeles'),
(3, 'Chicago');"

d) Populate the project table

"INSERTINTO project (pname, pnumber, plocation, dnum) VALUES
('Project A', 1, 'New York', 1),
('Project B', 2, 'Los Angeles', 2),
('Project C', 3, 'Chicago', 3);"

e) Populate the dependent table

"INSERTINTO dependent (essn, dependent_name, sex, bdate, relationship) VALUES
('123456789', 'Mary', 'F', '2005-06-15', 'Daughter'),
('234567890', 'Tom', 'M', '2010-08-25', 'Son'),
('345678901', 'Nancy', 'F', '2012-04-11', 'Daughter'),
('456789012', 'Paul', 'M', '2007-12-20', 'Son');"

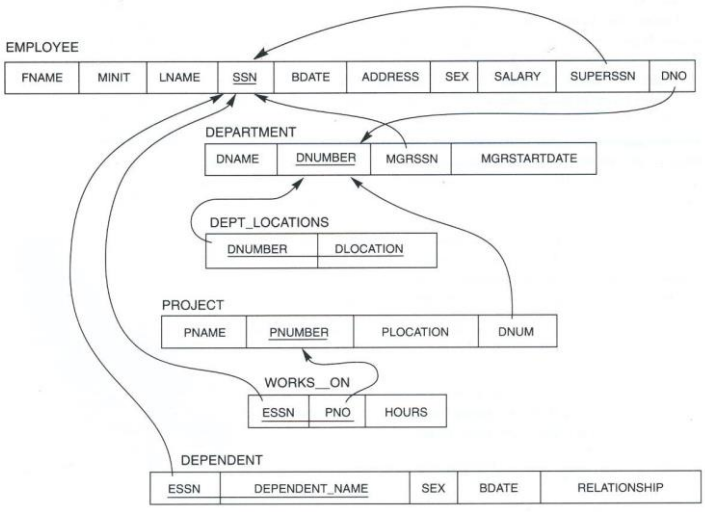

Figure: 1 Relational Database Schema

3. **Display the** ssn**,** lname**,** fname**, and** address **of employees who work in department number 3**

To display the ssn, lname, fname, and address of employees who work in department number 3 from the provided tables (employee, department, works_on), you can use a SQL query that joins these tables appropriately. Here's how you can do it:

“SELECTe.ssn, e.lname, e.fname, e.address
FROM employee e
JOINworks_on w ONe.ssn=w.essn
JOIN project p ONw.pno=p.pnumber
WHEREp.dnum=3;”

Explanation:

1. **Joining Tables**:
    - employee, works_on, and project tables are joined based on the relationships defined (works_on links employees to projects, and project identifies the department number).
2. **Filtering by Department**:
    - The WHERE clause filters results to only include employees who are working on projects belonging to department number 3 (p.dnum = 3).

4. **Retrieve the name and salary of every employee from the employee table**

To retrieve the name and salary of every employee from the employee table, you can use a simple SQL SELECT statement:

“SELECT fname, lname, salary

FROM employee;”

Explanation:

SELECT Statement: This query selects the fname, lname, and salary columns from the employee table.

To retrieve all distinct salary values from the employee table, you can use a simple SQL query with the DISTINCT keyword:

“SELECTDISTINCT salary
FROM employee;”

Explanation:

- **SELECT DISTINCT**: This query retrieves unique values of the salary column from the employee table.

5. Retrieve the ssn **and** department name **for all employees.**

To retrieve the ssn and department name for all employees from the provided tables (employee and department), you'll need to join these tables together based on the dno (department number) attribute. Here's how you can construct the query:

“SELECTe.ssn, d.dnameASdepartment_name
FROM employee e
JOIN department d ONe.dno=d.dnumber;”

Explanation:

- **JOIN Statement**: This query joins the employee table (e) with the department table (d) using a JOIN condition ON e.dno = d.dnumber.
- **Selecting Columns**: It selects the ssn from the employee table and renames d.dname as department_name from the department table.
- **Result**: This will retrieve the ssn (Social Security Number) and the corresponding department name for each employee, based on their dno association with a department in the department table.

6. Retrieve the department number, the number of employees in each department, and their average salary.

To retrieve the department number, the number of employees in each department, and their average salary, you can use a SQL query that joins the department and employee tables together and aggregates the data using GROUP BY. Here's how you can construct the query:

“SELECTd.dnumberASdepartment_number,
COUNT(e.ssn) ASnumber_of_employees,
AVG(e.salary) ASaverage_salary
FROM department d

LEFTJOIN employee e ONd.dnumber=e.dno
GROUPBYd.dnumber;”

Explanation:

- **SELECT Statement**:
  - d.dnumber AS department_number: Selects the department number from the department table and aliases it as department_number.
  - COUNT(e.ssn) AS number_of_employees: Counts the number of employees (ssn column) in each department.
  - AVG(e.salary) AS average_salary: Calculates the average salary of employees (salary column) in each department.
- **LEFT JOIN**: Joins the department table (d) with the employee table (e) based on d.dnumber = e.dno. This ensures all departments are included in the result, even if they have no employees (thanks to LEFT JOIN).
- **GROUP BY**: Groups the result set by d.dnumber, which is the department number. This is necessary whenever using aggregate functions (COUNT, AVG) alongside regular columns.
- **Result**: This query will produce a result set where each row represents a department, displaying the department number (department_number), the number of employees (number_of_employees), and the average salary (average_salary) of employees in that department.

To find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary across all employees, you can use aggregate functions in a single SQL query. Here's how you can construct the query:

“SELECT
SUM(salary) AStotal_salaries,
MAX(salary) ASmax_salary,
MIN(salary) ASmin_salary,
AVG(salary) ASaverage_salary
FROM employee;”

Explanation:

- **SELECT Statement**:
  - SUM(salary) AS total_salaries: Calculates the sum of all salaries (salary column) in the employee table and aliases it as total_salaries.
  - MAX(salary) AS max_salary: Finds the maximum salary (salary column) among all employees and aliases it as max_salary.
  - MIN(salary) AS min_salary: Determines the minimum salary (salary column) among all employees and aliases it as min_salary.
  - AVG(salary) AS average_salary: Computes the average salary (salary column) of all employees and aliases it as average_salary.
- **FROM Clause**: Specifies the employee table from which to retrieve the salary data.
- **Result**: This query will produce a single row result set with the total sum of salaries (total_salaries), the maximum salary (max_salary), the minimum salary (min_salary), and the average salary (average_salary) of all employees in the company.

Output:

The output will look something like this:

diff

Copy code

total_salaries	max_salary	min_salary	average_salary
240000.00	75000.00	50000.00	60000.00

Each column will display the respective calculated values with clear headings: total_salaries, max_salary, min_salary, and average_salary.

7. Create and populate the table named "Employee" with data.

Creating the Employees Table:

"CREATETABLE Employees (

EmployeeIDINT AUTO_INCREMENT PRIMARY KEY,

FirstNameVARCHAR(50) NOTNULL,

LastNameVARCHAR(50) NOTNULL,

Email VARCHAR(100) NOTNULL,

HireDateDATENOTNULL,

JobTitleVARCHAR(50) NOTNULL,
Salary DECIMAL(10, 2) NOTNULL
);”

Populating the Employees Table with Sample Data:

INSERTINTO Employees (FirstName, LastName, Email, HireDate, JobTitle, Salary) VALUES
('John', 'Doe', 'john.doe@example.com', '2020-01-15', 'Software Engineer', 75000.00),
('Jane', 'Smith', 'jane.smith@example.com', '2019-03-12', 'Project Manager', 85000.00),
('Michael', 'Brown', 'michael.brown@example.com', '2018-07-23', 'Data Analyst', 65000.00),
('Emily', 'Davis', 'emily.davis@example.com', '2021-05-30', 'UX Designer', 70000.00),
('David', 'Wilson', 'david.wilson@example.com', '2017-11-19', 'DevOps Engineer', 80000.00),
('Sarah', 'Miller', 'sarah.miller@example.com', '2022-02-25', 'QA Engineer', 72000.00),
('Robert', 'Moore', 'robert.moore@example.com', '2016-09-13', 'Systems Administrator', 78000.00),
('Laura', 'Taylor', 'laura.taylor@example.com', '2020-06-18', 'Technical Writer', 65000.00),
('James', 'Anderson', 'james.anderson@example.com', '2015-04-10', 'Product Manager', 90000.00),
('Patricia', 'Thomas', 'patricia.thomas@example.com', '2023-01-05', 'HR Specialist', 62000.00);

Explanation:

1. **Creating the Employees Table**:
    - EmployeeID is the primary key and auto-increments with each new record.
    - FirstName, LastName, Email, HireDate, JobTitle, and Salary are fields that store the employee's details.
2. **Populating the Employees Table**:
    - The INSERT INTO statement is used to add 10 sample employee records.
    - Each record includes a first name, last name, email, hire date, job title, and salary.

8. Perform the following queries on the database:

1) Write query to Select all employees:

“SELECT * FROM Employees;”

2) Find employees hired after January 1, 2020:

“SELECT * FROM Employees

WHERE HireDate> '2020-01-01';”

3) List employees with a salary greater than $70,000:

“SELECT * FROM Employees
WHERE Salary > 70000;”

4) Count the number of employees:

“SELECT COUNT(*) AS TotalEmployees FROM Employees;”

5) Find the highest salary in the table:

“SELECT MAX(Salary) AS HighestSalary FROM Employees;”

6) Find the average salary of employees:

“SELECT AVG(Salary) AS AverageSalary FROM Employees;”

7) List employees sorted by their hire date in descending order:

“SELECT * FROM Employees
ORDER BY HireDate DESC;”

8) Find employees whose last name starts with 'D':

“SELECT * FROM Employees
WHERE LastName LIKE 'D%';”

9) Select employees with the job title 'Software Engineer':

“SELECT * FROM Employees
WHERE JobTitle = 'Software Engineer';”

10) Find employees who have 'Manager' in their job title:

“SELECT * FROM Employees
WHERE JobTitle LIKE '%Manager%';”

11) Calculate the total payroll of the company:

“SELECT SUM(Salary) AS TotalPayroll FROM Employees;”

12) Find the employee with the earliest hire date:

“SELECT * FROM Employees
ORDER BY HireDate ASC
LIMIT 1;”

13) Count employees in each job title:

“SELECT JobTitle, COUNT(*) AS NumberOfEmployees
FROM Employees
GROUP BY JobTitle;”

14) Find employees hired in the year 2021:

“SELECT * FROM Employees
WHERE YEAR(HireDate) = 2021;”

15) Find employees with email addresses from the 'example.com' domain:

“SELECT * FROM Employees
WHERE Email LIKE '%@example.com';”

16) List employees with salaries between $60,000 and $80,000:

“SELECT * FROM Employees
WHERE Salary BETWEEN 60000 AND 80000;”

17) Find employees who were hired in the last 6 months:

“SELECT * FROM Employees
WHERE HireDate> DATE_SUB(CURDATE(), INTERVAL 6 MONTH);”

18) Calculate the average salary by job title:

“SELECT JobTitle, AVG(Salary) AS AverageSalary

FROM Employees
GROUP BY JobTitle;”

19) List the top 3 highest-paid employees:

“SELECT * FROM Employees
ORDER BY Salary DESC
LIMIT 3;”

20) Find employees with the job title 'Engineer' but not 'Software Engineer':

“SELECT * FROM Employees
WHERE JobTitle LIKE '%Engineer%' AND JobTitle != 'Software Engineer';”

21) Find employees hired in the same month but different years:

“SELECT FirstName, LastName, HireDate
FROM Employees
WHERE MONTH(HireDate) = MONTH(CURDATE());”

22) Calculate the percentage of total salary each employee earns:

“SELECT FirstName, LastName, Salary,
(Salary / (SELECT SUM(Salary) FROM Employees) * 100) AS SalaryPercentage
FROM Employees;”

23) Rank employees by their salary:

“SELECT FirstName, LastName, Salary,
RANK() OVER (ORDER BY Salary DESC) AS SalaryRank
FROM Employees;”

24) Select employees who have the same salary:

“SELECT E1.FirstName, E1.LastName, E1.Salary
FROM Employees E1
JOIN Employees E2 ON E1.Salary = E2.Salary AND E1.EmployeeID != E2.EmployeeID;”

25) Find the longest tenured employee:

“SELECT FirstName, LastName, HireDate
FROM Employees
ORDER BY HireDate ASC
LIMIT 1;”