# Day – 12

1. Create procedure or functions for employee table

Add 5000 bonus to all employee

Print same name employees

Print highest and lowest salary from employee table

SQL:

DROP TABLE IF EXISTS employee;

--Create employee table

CREATE TABLE employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

salary DECIMAL(10,2),

department VARCHAR(50)

);

-- Insert values into employee table

INSERT INTO employee VALUES

(1, 'Sai bavesh, 50000.00, 'IT'),

(2, 'rithwik', 65000.00, 'HR'),

(3, 'rohith', 72000.00, 'Finance'),

(4, 'revanth', 48000.00, 'IT'),

(5, 'charan', 55000.00, 'Marketing'),

(6, 'manish', 68000.00, 'HR');

-- Create procedure to add 5000 bonus to all employees

DELIMITER //

CREATE PROCEDURE AddBonusToAll()

BEGIN

UPDATE employee SET salary = salary + 5000;

SELECT \* FROM employee;

END //

DELIMITER ;

-- Create function to find same name employees

DELIMITER //

CREATE PROCEDURE FindSameNameEmployees()

BEGIN

SELECT emp\_name, COUNT(\*) as count

FROM employee

GROUP BY emp\_name

HAVING COUNT(\*) > 1;

END //

DELIMITER ;

--Create procedure to print highest and lowest salary

DELIMITER //

CREATE PROCEDURE GetSalaryRange()

BEGIN

SELECT

MAX(salary) as highest\_salary,

MIN(salary) as lowest\_salary

FROM employee;

END //

DELIMITER ;

Java code:

import java.sql.\*;

public class CallEmployeeProcedures {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/mydb";

String user = "root";

String password = "Ammanana@0310";

try (Connection con = DriverManager.getConnection(url, user, password)) {

// 1. employee table

System.out.println("Original Employee Data");

ResultSet originalRs = con.createStatement().executeQuery("SELECT \* FROM employee");

System.out.println("ID\tName\t\tSalary\t\tDepartment");

while (originalRs.next()) {

int id = originalRs.getInt("emp\_id");

String name = originalRs.getString("emp\_name");

double salary = originalRs.getDouble("salary");

String dept = originalRs.getString("department");

System.out.printf("%d\t%-15s\t%,.2f\t%s\n", id, name, salary, dept);

}

originalRs.close();

// 2. Add 5000 bonus

System.out.println("\nAdding 5000 Bonus");

CallableStatement addBonus = con.prepareCall("{CALL AddBonusToAll()}");

ResultSet bonusRs = addBonus.executeQuery();

System.out.println("ID\tName\t\tSalary\t\tDepartment");

while (bonusRs.next()) {

int id = bonusRs.getInt("emp\_id");

String name = bonusRs.getString("emp\_name");

double salary = bonusRs.getDouble("salary");

String dept = bonusRs.getString("department");

System.out.printf("%d\t%-15s\t%,.2f\t%s\n", id, name, salary, dept);

}

bonusRs.close();

// 2.Same name Employees procedure

System.out.println("\nEmployees with same names:");

CallableStatement sameNames = con.prepareCall("{CALL FindSameNameEmployees()}");

ResultSet nameRs = sameNames.executeQuery();

System.out.println("Name\t\tCount");

System.out.println("---------------------");

while (nameRs.next()) {

String name = nameRs.getString("emp\_name");

int count = nameRs.getInt("count");

System.out.printf("%-15s\t%d\n", name, count);

}

// 3.get salary range procedure

System.out.println("\nSalary Range:");

CallableStatement salaryRange = con.prepareCall("{CALL GetSalaryRange()}");

ResultSet rangeRs = salaryRange.executeQuery();

if (rangeRs.next()) {

double highest = rangeRs.getDouble("highest\_salary");

double lowest = rangeRs.getDouble("lowest\_salary");

System.out.printf("Highest Salary: %.2f\n", highest);

System.out.printf("Lowest Salary: %.2f\n", lowest);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

2.Create procedure or functions for Hospital table

1. print avg patient count on daily basis  
 2. print all the patients whose belong to same ward  
 3. arrange the patients list according their admission date

SQL:

CREATE DATABASE IF NOT EXISTS hospital\_db; USE hospital\_db;

CREATE TABLE staff(  
 staff\_id INT AUTO\_INCREMENT PRIMARY KEY,   
 username VARCHAR(50) UNIQUE NOT NULL,  
 password VARCHAR(100) NOT NULL,  
 role ENUM('Admin', 'Staff') NOT NULL );

INSERT INTO staff (username, password, role)   
VALUES ('admin', 'admin123', 'Admin');

-- Patients Table  
 CREATE TABLE patients (  
 patient\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,   
age INT,   
gender ENUM('Male', 'Female', 'Other'),   
phone VARCHAR(15),   
address VARCHAR(255)   
);

CREATE TABLE doctors (   
doctor\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(100),   
specialization VARCHAR(100),   
phone VARCHAR(15)  
 );

CREATE TABLE appointments (|  
appointment\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 patient\_id INT, doctor\_id INT,   
appointment\_date DATE,  
 reason VARCHAR(255),   
FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id),   
FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id) );

CREATE TABLE billing (   
billing\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 patient\_id INT,  
appointment\_id INT,  
 total\_amount DECIMAL(10,2),  
 discharge\_date DATE,   
summary TEXT,   
FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id),   
FOREIGN KEY (appointment\_id) REFERENCES appointments(appointment\_id) );  
DELIMITER //  
-- First drop the procedure if it exists   
DROP PROCEDURE IF EXISTS GetAveragePatientCount//  
-- Then create the procedure   
CREATE PROCEDURE GetAveragePatientCount()   
BEGIN

SELECT

ROUND(AVG(patient\_count), 2) AS average\_patients\_per\_day

FROM (

SELECT

DATE(created\_at) AS day,

COUNT(\*) AS patient\_count

FROM patients

GROUP BY DATE(created\_at)

) AS daily\_counts;

END //

DELIMITER ;

-- Procedure for patients in same ward

DELIMITER //

CREATE PROCEDURE GetPatientsByWard(IN ward\_name VARCHAR(100))

BEGIN  
 SELECT   
 patient\_id, name, age, gender

FROM patients

WHERE ward = ward\_name

ORDER BY name;

END //

DELIMITER ;

-- Procedure for patients sorted by admission date

DELIMITER //

CREATE PROCEDURE GetPatientsByAdmissionDate()

BEGIN

SELECT

patient\_id, name, age, gender, admission\_date

FROM patients

ORDER BY admission\_date DESC;

END //  
DELIMITER ;

ALTER TABLE patients ADD COLUMN ward VARCHAR(50);

ALTER TABLE patients ADD COLUMN admission\_date DATETIME DEFAULT CURRENT\_TIMESTAMP;

ALTER TABLE patients ADD COLUMN created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP;

DESCRIBE patients;

Java code:

import java.sql.Connection;

import java.sql.SQLException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws SQLException {

Scanner sc = new Scanner(System.in);

System.out.print("Username: ");

String username = sc.nextLine();

System.out.print("Password: ");

String password = sc.nextLine();

if (!LoginService.login(username, password)) {

System.out.println("Login failed. Exiting...");

return;

}

// Initialize services with connection

try {

Connection conn = Modules.getConnection();

PatientService.init(conn);

AppointmentService.init(conn);

BillingService.init(conn);

ReportService.init(conn);

while (true) {

printMainMenu();

int choice = getIntInput(sc, "Choose option: ");

switch (choice) {

case 1 -> PatientService.registerPatient(sc);

case 2 -> PatientService.viewPatients();

case 3 -> AppointmentService.bookAppointment(sc);

case 4 -> BillingService.generateBill(sc);

case 5 -> handleReportsMenu(sc);

case 6 -> {

System.out.println(" Goodbye!");

conn.close(); // Close connection before exit

return;

}

default -> System.out.println("Invalid option");

}

}

} catch (Exception e) {

System.err.println("Database connection error:");

e.printStackTrace();

}

}

private static void printMainMenu() {

System.out.println("\n=== Hospital Management System ===");

System.out.println("1. Register Patient");

System.out.println("2. View Patients");

System.out.println("3. Book Appointment");

System.out.println("4. Generate Bill");

System.out.println("5. Reports");

System.out.println("6. Exit");

}

private static void handleReportsMenu(Scanner sc) throws SQLException {

while (true) {

System.out.println("\n=== Report Options ===");

System.out.println("1. Average Patient Count");

System.out.println("2. Patients by Ward");

System.out.println("3. Patients by Admission Date");

System.out.println("4. Return to Main Menu");

int reportChoice = getIntInput(sc, "Select report type: ");

switch (reportChoice) {

case 1 -> ReportService.showAveragePatientCount();

case 2 -> {

sc.nextLine(); // Clear buffer

String ward = getStringInput(sc, "Enter ward name: ");

ReportService.showPatientsByWard(ward);

}

case 3 -> ReportService.showPatientsByAdmissionDate();  
case 4 -> { return; }

default -> System.out.println("Invalid report choice!");

}

}

}

private static int getIntInput(Scanner sc, String prompt) {

System.out.print(prompt);

while (!sc.hasNextInt()) {

sc.next(); // Discard invalid input

System.out.print("Please enter a number: ");

}

int input = sc.nextInt();

sc.nextLine(); // Consume newline

return input;

}  
private static String getStringInput(Scanner sc, String prompt) {

System.out.print(prompt);

return sc.nextLine();

}  
}