Sample Problem statements for DMS Practical exam

Problem 1: College Admission System

Schema:

- Student(sid INT, name VARCHAR(50), gender VARCHAR(10), dept_id INT)
- Department(dept_id INT, dept_name VARCHAR(50), intake INT)

Questions:

1. Create tables with appropriate keys and constraints.
2. Add 5 students and 3 departments.
3. Display names of all male students and their department names.
4. List departments with more than 2 students using GROUP BY and HAVING.
5. Update the intake to increase by 10% for all departments.

Problem 2: Online Retail Store

Schema:

- Customers(cust_id INT, name VARCHAR(50), city VARCHAR(30))
- Orders(order_id INT, cust_id INT, amount DECIMAL(10,2), order_date DATE)

Questions:

1. Create both tables with appropriate constraints.
2. Insert at least 4 customers and 5 orders.
3. Display customer names who placed orders above ₹5000.
4. List total order amount placed by each customer in descending order.
5. Retrieve customers who haven't placed any orders.

Problem 3: Bookstore Inventory

Schema:

- Books(book_id INT, title VARCHAR(100), price DECIMAL(8,2), pub_year INT)
- Sales(sale_id INT, book_id INT, quantity INT, sale_date DATE)

Questions:

1. Create tables with suitable constraints.
2. Insert 4 books and 5 sales records.

3. Display titles of books sold in the year 2024.
4. Show total sales revenue for each book using SUM(price * quantity).
5. Find the title of the most sold book using ORDER BY and LIMIT.

Problem 4: Airline Reservation

Schema:

- Flights(flight_id INT, source VARCHAR(30), destination VARCHAR(30), fare DECIMAL(6,2))
- Passengers(pid INT, name VARCHAR(50), flight_id INT, travel_date DATE)

Questions:

1. Create both tables with constraints.
2. Insert 3 flights and 5 passenger bookings.
3. List all passengers travelling to 'Delhi'.
4. Show flight-wise passenger count.
5. Increase fare by 10% for flights having more than 2 bookings.

Problem 5: Employee Performance Tracker

Schema:

- Employee(emp_id INT, name VARCHAR(50), designation VARCHAR(30), salary INT)
- Performance(emp_id INT, month VARCHAR(15), rating INT)

Questions:

1. Create schema and insert sample data.
2. Find employees with average rating > 4.
3. Display highest rated employee each month.
4. List employees who never received a rating using NOT IN.
5. Display total salary to be paid for 'Manager' designation employees.

Procedure:

A company wants to give a bonus of ₹5000 to employees whose salaries are less than ₹30,000. The HR department maintains a database of employee records.

Schema:

- Employees(emp_id INT PRIMARY KEY, name VARCHAR(50), salary INT, bonus INT DEFAULT 0)

Tasks:
1.Write a **stored procedure using a cursor** that:

- Retrieves all employees with salary < ₹30,000
- Adds ₹5000 to their bonus column
- Displays their name and updated bonus value

2. A library tracks borrowed books and their return status. A fine of ₹2 is applied for each day after the due date.

Schema:

- Borrowers(borrow_id INT PRIMARY KEY, student_name VARCHAR(50), due_date DATE, return_date DATE, fine INT DEFAULT 0)

Task:
Write a **stored procedure using a cursor** to:

- Loop through all records in Borrowers
- For each student who returned the book late, calculate the number of overdue days
- Multiply overdue days by ₹2 and update the fine column
- Show a message like: Fine of ₹20 updated for Rahul Singh

Trigger

1: Track Salary Updates

Context:
A company wants to **maintain a log of all salary changes** for employees. Every time an employee's salary is updated, the old and new values should be stored in a separate table for audit purposes.

Tables:

- employees(emp_id INT PRIMARY KEY, name VARCHAR(50), salary DECIMAL(10,2))
- salary_log(log_id INT AUTO_INCREMENT PRIMARY KEY, emp_id INT, old_salary DECIMAL(10,2), new_salary DECIMAL(10,2), change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP)

Objective:
Create a **BEFORE UPDATE trigger** on the employees table that:

- Captures the old and new salary values whenever salary is updated
- Inserts them into the salary_log table

Problem statements based on MongoDB

Student Performance Tracker

Context:
A university wants to track students' marks across various subjects using MongoDB.

Collection: students

Sample Document:

```
{
  "roll_no": 101,
  "name": "Ankita Desai",
  "department": "IT",
  "marks": [
    { "subject": "DBMS", "score": 78 },
    { "subject": "AI", "score": 89 },
    { "subject": "OS", "score": 91 }
  ]
}
```

Tasks:

1. Insert at least 5 student documents with varying subjects and marks.
2. Retrieve all students with more than 85 in "AI".
3. Update the DBMS score of student roll_no: 101 to 85.
4. Delete a student with roll_no: 105.
5. Use aggregation to find the **average score in OS** across all students.

: Online Bookstore Database

Context:
An online bookstore wants to manage books, authors, and price data.

Collection: books

Sample Document:

```
{
  "title": "The MongoDB Guide",
  "author": "Ravi Joshi",
```

"price": 499,

"category": "Database",

"ratings": [4, 5, 5, 3]

}

Tasks:

1. Insert 5 books with details like title, author, price, category, and rating array.
2. Find all books priced under ₹500.
3. Update the price of a book titled "The MongoDB Guide" to ₹450.
4. Delete all books from category "Old Stock".
5. Use aggregation to calculate the **average rating per book**.

4: Hospital Patient Records System

Context:
A hospital wants to store and analyze basic patient treatment information.

Collection: patients

Sample Document

{

"patient_id": "P1001",

"name": "Rohan Kulkarni",

"age": 45,

"department": "Cardiology",

"treatments": [

{ "treatment": "ECG", "cost": 1200 },

{ "treatment": "Angiography", "cost": 15000 }

]

}

Tasks:

1. Insert 4–5 patient documents with multiple treatments.
2. Retrieve all patients from "Cardiology".
3. Add a new treatment for patient "P1001".

4. Delete records of patients older than 80 years.
5. Use aggregation to compute the **total treatment cost per patient**.

Problem 4: Movie Ratings and Reviews

Context:
A movie platform stores user reviews and wants to perform analysis on the data.

Collection: movies

Sample Document:

```
{
  "movie_id": 1,
  "title": "Interstellar",
  "genre": "Sci-Fi",
  "release_year": 2014,
  "ratings": [
    { "user": "user1", "score": 5 },
    { "user": "user2", "score": 4 }
  ]
}
```

Tasks:

1. Insert at least 5 movie documents with ratings.
2. Find all movies released after 2010 in the "Sci-Fi" genre.
3. Update the title of a movie from "Inception" to "Inception (2010)".
4. Delete all movies with an average rating below 3.
5. Use aggregation to calculate the **average score of each movie**.