**ADBMS Lab schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.NO** | **LAB EXPERIMENTS** | **DATE** | **Approx.Date** | **Actual date** |
| **1** | **Introduction to Relational Databases & ER Model** | **Week 1,2** | **5/02/2025,12/02/2025** | **5/02/2025,12/02/2025** |
| **2** | **Implementing Constraints in Relational Databases** | **Week 3,4** | **26/002/2025,05/03/2025** | **26/002/2025,05/03/2025** |
| **3** | **Advanced SQL - Joins and Subqueries** | **Week 5,6** | **12/03/2025,19/03/2025** |  |
| **4** | **Working with Views and Triggers**  **Transaction Management and Concurrency Control** | **Week7,8** | **26/03/2025,02/04/2025** |  |
| **5** | **Object-Oriented Databases and Complex Data Types**  **NoSQL Database (MongoDB) Basics** | **Week 9**  **Week 10** | **09/04/2025,16/04,2025** |  |

**LIST OF LAB EXPERIMENTS**

**MODULE 1( LAB CYCLE 1) (CO1)**

**Lab Experiment 1: Introduction to Relational Databases & ER Model**

1. Understand the basics of relational databases and their applications.
2. Design an Entity-Relationship (ER) diagram for a simple scenario (e.g., Library Management System).
3. Convert the ER diagram to a relational schema and implement it in SQL.

**Lab Experiment 2: Implementing Constraints in Relational Databases**

1. Create a database with various constraints (Primary Key, Foreign Key, Unique, Not Null).
2. Demonstrate the enforcement of entity integrity and referential integrity using SQL.

**MODULE 2 LAB CYCLE 1) (CO2)**

**Lab Experiment 3: Advanced SQL - Joins and Subqueries**

* Implement different types of joins (INNER, LEFT, RIGHT, FULL OUTER).
* Use subqueries (both nested and correlated) .

**Experiment3.1: Retrieve Orders Issued by a Specific Salesman**

Create Salesman and Orders tables with the following sample data:

**Salesman Table:**

| **salesman\_id** | **name** | **city** | **commission** |
| --- | --- | --- | --- |
| 5001 | AKASH | NY | 0.15 |
| 5002 | AMAN | Paris | 0.13 |
| 5005 | FIDHA | London | 0.11 |

**Orders Table:**

| **ord\_no** | **purch\_amt** | **ord\_date** | **customer\_id** | **salesman\_id** |
| --- | --- | --- | --- | --- |
| 70008 | 5760 | 2012-09-10 | 3002 | 5001 |
| 70010 | 1983.43 | 2012-10-10 | 3004 | 5002 |
| 70003 | 2480.4 | 2012-10-10 | 3009 | 5005 |
| 70012 | 250.45 | 2012-06-27 | 3008 | 5002 |
| 70013 | 3045.6 | 2012-04-25 | 3002 | 5001 |

**Find all orders issued by salesman 'AKASH '(USE SUBQUERY CONCEPT)**

**Expected Result:** Retrieve all orders associated with 'Akash'.

**Experiment 3.2:Use the emp and dept tables with appropriate sample data.**

**Find employees whose salaries are above the average salary in their department. (Use a subquery)**

**Expected result:** List employees earning more than the average salary in their respective departments.

Experiment 3.3: **Retrieve Employees Working in a Specific Location.**

**Find customers with the minimum age. Use subquery**

**MODULE 3 ( LAB CYCLE 2) (CO3)**

**Lab Experiment 5: Working with Views and Triggers**

Create and manage views in SQL, including updatable views.

**Experiment 5.1** :Develop a view named employee\_contact\_info that combines data from the employees and departments tables.

**Experiment 5.2 Create** an updatable view and use it to modify data in the underlying table.

Implement triggers to automatically handle insert, update, or delete actions.

**{Experiment 5.3: Implementing an Audit Trail for Employee Salary Changes**

**C**reate a trigger that logs changes to employee salaries, maintaining an audit trail for monitoring salary adjustments.

**NOTE:Create the employees Table:**Design a table to store employee information, including fields like id, name, position, and salary.**Create the salary\_audit Table:**Design a table to log salary changes, including fields like audit\_id, employee\_id, old\_salary, new\_salary, and change\_date.**Create a Trigger to Log Salary Changes:**

Develop a BEFORE UPDATE trigger on the employees table that captures the old salary before any update and inserts a record into the salary\_audit table.

EXPECTED RESULT:Whenever an employee's salary is updated, a record of the change is automatically stored in the salary\_audit table, providing a history of salary modifications.

* **Experiment 5.4: Preventing Deletion of Critical Product Records**

**Create the products Table:**Design a table to store product details, including id, name, price, and a critical flag indicating if the product is essential.**Insert Sample Data:**Populate the products table with sample records, marking certain products as critical.

Develop a BEFORE DELETE trigger on the products table that checks if the critical flag is set. If so, the trigger prevents the deletion operation.

**EXPECTED RESULT:** Attempts to delete critical products are blocked, preserving essential data within the database.

* **Lab Experiment 5.5: Automatically Updating Stock Quantities After Sales**

**C**reate a trigger that automatically updates the stock quantity of products in the inventory after a sale is recorded.

**Create the products Table:**Design a table to store product information, including id, name, and stock\_quantity.

**Create the sales Table:**Design a table to record sales transactions, including sale\_id, product\_id, quantity\_sold, and sale\_date.

Develop an AFTER INSERT trigger on the sales table that automatically updates the stock\_quantity in the products table by subtracting the quantity\_sold whenever a new sale is recorded.

**Lab Experiment 6: Transaction Management and Concurrency Control**

* Implement basic transaction control (BEGIN, COMMIT, ROLLBACK).

**MODULE 5 ( LAB CYCLE 2) (CO5)**

**Lab Experiment 9: Object-Oriented Databases and Complex Data Types April 2**

* Implement complex data types (arrays, multisets, structured types) in SQL.
* Work with inheritance and object identity in SQL databases.

**Lab Experiment 10: Build an application using MYSQL Database or MongoDB April 3**

* **Design and build a simple database system and demonstrate competence with the fundamental tasks involved with modeling, designing, and implementing a DBMS**..