**Lab Exercise 3 - Practicing Integrity Constraints in MySQL**

**Objective:**

To understand and apply various **integrity constraints** in MySQL such as PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, CHECK, and DEFAULT.

**Software Required:**

* MySQL Server / MySQL Workbench / phpMyAdmin
* SQL-compatible terminal or IDE

**Part A: Create a Database**

CREATE DATABASE ConstraintLab;

USE ConstraintLab;

**Part B: Create Tables with Constraints**

**Task 1: Create Department Table**

CREATE TABLE Department (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50) UNIQUE NOT NULL

);

**Constraints Used:**

* PRIMARY KEY: Ensures unique department ID.
* UNIQUE: Ensures no duplicate department names.
* NOT NULL: Ensures department name is always provided.

**Task 2: Create Employee Table**

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50) NOT NULL,

emp\_email VARCHAR(100) UNIQUE,

emp\_salary DECIMAL(10,2) CHECK (emp\_salary >= 3000),

emp\_join\_date DATE DEFAULT CURRENT\_DATE,

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id)

);

**Constraints Used:**

* PRIMARY KEY: Unique employee ID.
* NOT NULL: Employee name cannot be empty.
* UNIQUE: Prevents duplicate emails.
* CHECK: Salary must be at least 3000.
* DEFAULT: Join date defaults to today’s date.
* FOREIGN KEY: Links each employee to a department.

**Part C: Insert Data and Observe Constraints**

**Task 3: Insert Valid Data**

INSERT INTO Department (dept\_id, dept\_name) VALUES (1, 'HR'), (2, 'Engineering');

INSERT INTO Employee (emp\_id, emp\_name, emp\_email, emp\_salary, dept\_id)

VALUES

(101, 'Alice', 'alice@example.com', 5000, 1),

(102, 'Bob', 'bob@example.com', 4500, 2);

**Task 4: Try Inserting Invalid Data**

1. Duplicate primary key:

INSERT INTO Employee (emp\_id, emp\_name, emp\_email, emp\_salary, dept\_id)

VALUES (101, 'Charlie', 'charlie@example.com', 4000, 2); -- Should fail

1. NULL in NOT NULL column:

INSERT INTO Employee (emp\_id, emp\_name, emp\_email, emp\_salary, dept\_id)

VALUES (103, NULL, 'null@example.com', 4000, 1); -- Should fail

1. Duplicate email:

INSERT INTO Employee (emp\_id, emp\_name, emp\_email, emp\_salary, dept\_id)

VALUES (104, 'David', 'alice@example.com', 4000, 1); -- Should fail

1. Salary below check constraint:

INSERT INTO Employee (emp\_id, emp\_name, emp\_email, emp\_salary, dept\_id)

VALUES (105, 'Eva', 'eva@example.com', 2500, 1); -- Should fail

1. Invalid foreign key:

INSERT INTO Employee (emp\_id, emp\_name, emp\_email, emp\_salary, dept\_id)

VALUES (106, 'Frank', 'frank@example.com', 4000, 5); -- Should fail

**Part D: View and Verify Data**

**Task 5: View Valid Records**

SELECT \* FROM Employee;

**Task 6: View Departments**

SELECT \* FROM Department;

**Learning Outcomes:**

By the end of this lab, you will be able to:

* Apply various integrity constraints while creating tables
* Understand how constraints prevent invalid or inconsistent data
* Observe the effects of violated constraints through insert operations
* Strengthen your foundation in relational database design and integrity