DBMS Lab - Session-7

106122076 CSE-B
Lohesh S U

1. Write the following as triggers on the corresponding schema mentioned which you have already developed. In each case, disallow it if it does not satisfy the stated constraint. You may assume that the desired condition holds before any change to the database is attempted. Also, prefer to modify the database, even if it means inserting tuples with NULL or default values, rather than rejecting the attempted modification.

a. Assure that deleting details of an employee deletes his dependent records also.

```
CREATE TRIGGER delete_employee_cascade
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
    DELETE FROM Dependent WHERE Essn = OLD.Ssn;
END;
```

b. Whenever a department with exactly one project is shifted to a new location, ensure that the project is also shifted to the new location.

```
CREATE TRIGGER shift_project_with_department
AFTER UPDATE ON Dept_locations
FOR EACH ROW
BEGIN
    DECLARE project_count INT;
    SET project_count = (SELECT COUNT(*) FROM Project WHERE Dnum =
NEW.Dnumber);

    IF project_count = 1 THEN
       UPDATE Project SET Plocation = NEW.Dlocation WHERE Dnum = NEW.Dnumber;
    END IF;
END;
```

c. Assure at all times that there are no departments with more than 3 projects.

```
CREATE TRIGGER limit_department_projects
AFTER INSERT ON Project
FOR EACH ROW
BEGIN
```

```
    DECLARE project_count INT;
    SET project_count = (SELECT COUNT(*) FROM Project WHERE Dnum = NEW.Dnum);

    IF project_count > 3 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Departments
cannot have more than 3 projects';
    END IF;
END;
```

d. Assure that no employees work for more than one department.

```
CREATE TRIGGER limit_employee_departments
BEFORE INSERT ON Works_on
FOR EACH ROW
BEGIN
    DECLARE dept_count INT;
    SET dept_count = (SELECT COUNT(DISTINCT Dno) FROM Works_on JOIN
Project ON Works_on.Pno = Project.Pnumber WHERE Essn = NEW.Essn);

    IF dept_count > 1 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employees cannot
work for more than one department';
    END IF;
END;
```

e. Whenever a project is dropped, dissociate all the employees from the particular project.

```
CREATE TRIGGER dissociate_employees_on_project_delete
AFTER DELETE ON Project
FOR EACH ROW
BEGIN
    DELETE FROM Works_on WHERE Pno = OLD.Pnumber;
END;
```

f. When a new department is inaugurated, ensure that it is not co-located with any other departments.

```
CREATE TRIGGER prevent_co_location
BEFORE INSERT ON Department
FOR EACH ROW
BEGIN
    DECLARE location_count INT;
    SET location_count = (SELECT COUNT(*) FROM Dept_locations WHERE
```

```
Dlocation = (SELECT Dlocation FROM Dept_locations WHERE Dnumber =
NEW.Dnumber));

    IF location_count > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'New department
cannot be co-located with existing departments';
    END IF;
END;
```

g. For every employee, ensure that his dependent Birthdate is less than his Birthdate.

```
CREATE TRIGGER dependent_birthdate_check
BEFORE INSERT ON Dependent
FOR EACH ROW
BEGIN
    DECLARE employee_bdate DATE;
    SET employee_bdate = (SELECT Bdate FROM Employee WHERE Ssn = NEW.Essn);

    IF NEW.Bdate >= employee_bdate THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Dependent
birthdate must be less than employee birthdate';
    END IF;
END;
```

h. Increment 1000 rupees to the salary for those employees if any of his/her dependents expire.

```
CREATE TRIGGER increment_salary_on_dependent_expiry
AFTER DELETE ON Dependent
FOR EACH ROW
BEGIN
    UPDATE Employee SET Salary = Salary + 1000 WHERE Ssn = OLD.Essn;
END;
Triggers for Flight Schema
```

i. Create a trigger that handles an update command to find the total salary of all pilots. Check the condition such that the new tuples inserted should not be null and salary should be more than 50,000.

```
CREATE TRIGGER update_total_pilot_salary
AFTER INSERT OR UPDATE ON Employee
FOR EACH ROW
BEGIN
```

```
    IF NEW.Salary IS NULL OR NEW.Salary <= 50000 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salary must not
be null and should be more than 50,000';
    END IF;

    -- Assuming 'Certified' table links pilots to aircraft
    IF EXISTS (SELECT 1 FROM Certified WHERE eid = NEW.eid) THEN
        -- Sum of salaries of all pilots
        SELECT SUM(Salary) INTO @total_salary FROM Employee WHERE eid
IN (SELECT eid FROM Certified);
    END IF;
END;
```

j. Create a trigger to set salary as 30,000 if there is a NULL present in it. Also, check whether a salary of a pilot is greater than the salary of a non-pilot.

```
CREATE TRIGGER set_salary_to_default
BEFORE INSERT OR UPDATE ON Employee
FOR EACH ROW
BEGIN
    IF NEW.Salary IS NULL THEN
        SET NEW.Salary = 30000;
    END IF;

    DECLARE pilot_salary INT;
    DECLARE non_pilot_salary INT;
    SET pilot_salary = (SELECT MIN(Salary) FROM Employee WHERE eid IN
(SELECT eid FROM Certified));
    SET non_pilot_salary = (SELECT MAX(Salary) FROM Employee WHERE
eid NOT IN (SELECT eid FROM Certified));

    IF pilot_salary < non_pilot_salary THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Pilot salary must
be greater than non-pilot salary';
    END IF;
END;
```

k. Create a trigger to foil any attempt to lower the salary of an employee.

```
CREATE TRIGGER prevent_salary_decrease
BEFORE UPDATE ON Employee
FOR EACH ROW
BEGIN
```

```
    IF NEW.Salary < OLD.Salary THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot lower the
salary of an employee';
    END IF;
END;
```

l. When inserting a new certification for an employee, check that the aircraft id exists in the Aircraft.

```
CREATE TRIGGER validate_aircraft_exists
BEFORE INSERT ON Certified
FOR EACH ROW
BEGIN
    IF (SELECT COUNT(*) FROM Aircraft WHERE aid = NEW.aid) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Aircraft ID does
not exist in Aircraft table';
    END IF;
END;
```

m. When making any modifications to the Aircraft table, check that the cruising range is greater than or equal to the distance of flights.

```
CREATE TRIGGER check_cruising_range
BEFORE UPDATE ON Aircraft
FOR EACH ROW
BEGIN
    IF EXISTS (SELECT 1 FROM Flights WHERE cruisingrange < distance
AND aid = NEW.aid) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cruising range
must be greater than or equal to flight distances';
    END IF;
END;
```

n. When a new certification is inserted into Certified, also insert an employee with the id of that employee and a NULL salary.

```
CREATE TRIGGER insert_employee_on_certification
AFTER INSERT ON Certified
FOR EACH ROW
BEGIN
    IF (SELECT COUNT(*) FROM Employee WHERE eid = NEW.eid) = 0 THEN
        INSERT INTO Employee (eid, ename, salary) VALUES (NEW.eid,
NULL, NULL);
    END IF;
```

```
END;
```

o. Terminate pilots and their certification when the pilot retires.

```
CREATE TRIGGER terminate_pilot_on_retirement
AFTER DELETE ON Employee
FOR EACH ROW
BEGIN
    DELETE FROM Certified WHERE eid = OLD.eid;
END;
```

p. Write a trigger for the condition mentioned: Suppose we want to prevent the average salary of an employee from dropping below Rs. 50,000.

```
CREATE TRIGGER prevent_avg_salary_drop
BEFORE INSERT OR UPDATE OR DELETE ON Employee
FOR EACH ROW
BEGIN
    DECLARE avg_salary DECIMAL(10,2);
    SET avg_salary = (SELECT AVG(Salary) FROM Employee);

    IF avg_salary < 50000 THEN
       SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Average salary of
employees must not drop below Rs. 50,000';
    END IF;
END;
```

2. Write the following as Cursors on the corresponding Schema.

q. Develop a stored procedure to insert a new attribute ‘address’ in DEPENDENT and update the same as that of the employee’s address.

```
DELIMITER //

CREATE PROCEDURE UpdateDependentAddress()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE emp_ssn CHAR(9);
    DECLARE emp_address VARCHAR(100);
    DECLARE cur CURSOR FOR SELECT Ssn, Address FROM Employee;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
```

```
    read_loop: LOOP
        FETCH cur INTO emp_ssn, emp_address;
        IF done THEN
            LEAVE read_loop;
        END IF;
        UPDATE Dependent SET Address = emp_address WHERE Essn = emp_ssn;
    END LOOP;

    CLOSE cur;
END //
```

DELIMITER ;

r. Develop a stored procedure to display the fname, ssn, and salary, grade of an employee. Handle the condition such that if salary of an employee is 1 - 10000, assign grade3, grade2 if salary in between 10000 and 50000, and grade1 if salary > 50000. Handle exception with an error message when an invalid case occurs.

```
DELIMITER //

CREATE PROCEDURE DisplayEmployeeGrade()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE emp_fname VARCHAR(50);
    DECLARE emp_ssn CHAR(9);
    DECLARE emp_salary INT;
    DECLARE emp_grade VARCHAR(10);
    DECLARE cur CURSOR FOR SELECT Fname, Ssn, Salary FROM Employee;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO emp_fname, emp_ssn, emp_salary;
        IF done THEN
            LEAVE read_loop;
        END IF;

        IF emp_salary BETWEEN 1 AND 10000 THEN
            SET emp_grade = 'grade3';
        ELSEIF emp_salary BETWEEN 10001 AND 50000 THEN
            SET emp_grade = 'grade2';
```

```
        ELSEIF emp_salary > 50000 THEN
            SET emp_grade = 'grade1';
        ELSE
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid salary range';
        END IF;

        SELECT emp_fname, emp_ssn, emp_salary, emp_grade;
    END LOOP;

    CLOSE cur;
END //

DELIMITER ;
```

s. Create a stored procedure to display deptno, avgsalary, and #employees in each department. Handle exceptions with an error message when invalid deptno is given.

```
DELIMITER //

CREATE PROCEDURE DisplayDepartmentStats()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE deptno INT;
    DECLARE avg_salary DECIMAL(10,2);
    DECLARE num_employees INT;
    DECLARE cur CURSOR FOR
        SELECT Dno, AVG(Salary), COUNT(*) FROM Employee
        JOIN Department ON Employee.Dno = Department.Dnumber
        GROUP BY Dno;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO deptno, avg_salary, num_employees;
        IF done THEN
            LEAVE read_loop;
        END IF;

        SELECT deptno, avg_salary, num_employees;
    END LOOP;

    CLOSE cur;
```

```
END //

DELIMITER ;
```

Cursors and Stored Procedures for Flight Schema
t. Develop a stored procedure to update an employee record given the employee id. Print a message after the update is successfully done with exception handling for an invalid employee id.

```
DELIMITER //

CREATE PROCEDURE UpdateEmployeeRecord(IN emp_id INT, IN new_salary INT)
BEGIN
    DECLARE emp_count INT;

    SET emp_count = (SELECT COUNT(*) FROM Employee WHERE eid = emp_id);

    IF emp_count = 0 THEN
       SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid employee id';
    ELSE
       UPDATE Employee SET Salary = new_salary WHERE eid = emp_id;
       SELECT CONCAT('Employee with ID ', emp_id, ' updated
successfully.') AS Message;
    END IF;
END //

DELIMITER ;
```

u. Develop a stored procedure to display the name and salary of each employee from the employee table. Handle the condition such that if the salary of an employee is above 50,000, rank them as Grade ‘A’, else as Grade ‘B’.

```
DELIMITER //

CREATE PROCEDURE DisplayEmployeeGradeBySalary()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE emp_name VARCHAR(50);
    DECLARE emp_salary INT;
    DECLARE emp_grade CHAR(1);
    DECLARE cur CURSOR FOR SELECT CONCAT(Fname, ' ', Lname), Salary
FROM Employee;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
```

```
    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO emp_name, emp_salary;
        IF done THEN
            LEAVE read_loop;
        END IF;

        IF emp_salary > 50000 THEN
            SET emp_grade = 'A';
        ELSE
            SET emp_grade = 'B';
        END IF;

        SELECT emp_name, emp_salary, emp_grade;
    END LOOP;

    CLOSE cur;
END //

DELIMITER ;
```

v. Develop a stored procedure that builds a name list of all employees who are certified for a Boeing aircraft and handle an exception with an error message.

```
DELIMITER //

CREATE PROCEDURE ListCertifiedBoeingEmployees()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE emp_name VARCHAR(50);
    DECLARE cur CURSOR FOR
        SELECT CONCAT(E.Fname, ' ', E.Lname)
        FROM Employee E
        JOIN Certified C ON E.eid = C.eid
        JOIN Aircraft A ON C.aid = A.aid
        WHERE A.aname LIKE '%Boeing%';
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
```

```
        FETCH cur INTO emp_name;
        IF done THEN
           LEAVE read_loop;
        END IF;

        SELECT emp_name;
    END LOOP;

    CLOSE cur;
END //

DELIMITER ;
```

3. On the Company Relational Schema, execute the following queries.

3a. Display all odd-numbered alternate records from ‘Employee’ table.

```
SELECT *
FROM Employee
WHERE MOD(id, 2) = 1;
```

3b. Display all even-numbered alternate records from ‘Employee’ table.

```
SELECT *
FROM Employee
WHERE MOD(id, 2) = 0;
```

3c. Find year from birth date when the date is a VARCHAR column instead of the proper DATE data type.

```
SELECT SUBSTRING(Bdate, 1, 4) AS Year
FROM Employee;
```

3d. Select the first 3 characters of the first name.

```
SELECT LEFT(Fname, 3)
FROM Employee;
```

3e. Find duplicate rows in a table of your choice.

```
SELECT Fname, COUNT(*)
FROM Employee
GROUP BY Fname
HAVING COUNT(*) > 1;
```

3f. Delete the duplicate records retrieved using the above query without using a temporary table.

```
DELETE e1
FROM Employee e1
JOIN Employee e2
ON e1.id > e2.id AND e1.Fname = e2.Fname;
```

3g. Delete the duplicate records retrieved using the above query using a temporary table.

```
CREATE TEMPORARY TABLE temp_employee AS
SELECT MIN(id) AS id FROM Employee
GROUP BY Fname;
DELETE FROM Employee
WHERE id NOT IN (SELECT id FROM temp_employee);
```

3h. Extract the 3rd maximum salary. Also find nth max salary.

```
SELECT DISTINCT Salary
FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET 2;


SELECT DISTINCT Salary
FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET n-1;
```

3i. Get first 3 max salaries. Also find first n max salaries.

```
-- For first 3 max salaries
SELECT DISTINCT Salary
FROM Employee
ORDER BY Salary DESC
LIMIT 3;

-- For first n max salaries, replace `n` with the desired limit.
SELECT DISTINCT Salary
FROM Employee
ORDER BY Salary DESC
LIMIT n;
```

3j. Display year, month, day as separate attributes from employee's date of birth.

```
SELECT YEAR(Bdate) AS Year, MONTH(Bdate) AS Month, DAY(Bdate) AS Day
FROM Employee;
```

3k. Retrieve the date part of the date or datetime expression.

```
SELECT DATE(Bdate)
FROM Employee;
```

3l. Get position of 'a' in name 'Sundar Pitchai' from employee table.

```
SELECT POSITION('a' IN 'Sundar Pitchai')
FROM Employee;
```

3m. Get fname from employee table after removing white spaces from left side.

```
SELECT LTRIM(Fname)
FROM Employee;
```

o. Get fname from the employee table after replacing 'o' with '*'

```
SELECT REPLACE(fname, 'o', '*') AS modified_fname
FROM Employee;
```

p. Get fname and lname as a single attribute from the employee table separated by '_'

```
SELECT CONCAT(fname, '_', lname) AS full_name
FROM Employee;
```

q. Find all employee records containing the word "Jai", regardless of whether it was stored as JAI, Jai, or jai

```
SELECT *
FROM Employee
WHERE LOWER(fname) LIKE '%jai%';
```

r. Find the number of employees according to gender whose DOB is between 05/01/1980 to 31/12/2024

```
SELECT gender, COUNT(*) AS number_of_employees
FROM Employee
WHERE DOB BETWEEN '1980-01-05' AND '2024-12-31'
```

```
GROUP BY gender;
```

s. Retrieve the MySQL username and password

```
SELECT user, host
FROM mysql.user;
```

t. Find all the employee first names whose name consists of three or more words

```
SELECT fname
FROM Employee
WHERE LENGTH(fname) - LENGTH(REPLACE(fname, ' ', '')) >= 2;
```

u. Get employee details from the employee table whose first name ends with 'n' and contains 4 letters

```
SELECT *
FROM Employee
WHERE fname LIKE '___n';
```

v. Get employee details from the employee table whose joining month is “January”

```
SELECT *
FROM Employee
WHERE MONTHNAME(joining_date) = 'January';
```

w. Fetch data that are common in two query results

```
-- Assuming two queries: Query1 and Query2
SELECT *
FROM (Query1)
INTERSECT
SELECT *
FROM (Query2);
```

x. Get first names of employees who have '*' in last_name

```
SELECT fname
FROM Employee
WHERE lname LIKE '%*%';
```

y. Find department from the dept table after replacing special characters with a white space

```
SELECT REPLACE(dept_name, '[^a-zA-Z0-9]', ' ') AS modified_dept_name
FROM Department;
```

z. Retrieve the number of employees joined with respect to a particular year and month

```
SELECT YEAR(joining_date) AS year, MONTH(joining_date) AS month,
COUNT(*) AS number_of_employees
FROM Employee
GROUP BY YEAR(joining_date), MONTH(joining_date);
```

aa. Extract characters within a specified range of length from the department field

```
SELECT SUBSTRING(dept_name, start_position, length) AS substring_dept_name
FROM Department;
```

bb. Convert the name of the employee to lowercase and then as uppercase

```
SELECT LOWER(fname) AS lower_name, UPPER(fname) AS upper_name
FROM Employee;
```

cc. Select FIRST n records from a department table

```
SELECT *
FROM Department
LIMIT n;
```

dd. Select LAST n records from a department table

```
SELECT *
FROM Department
ORDER BY dept_id DESC
LIMIT n;
```

ee. Select first name from the employee table which contains only numbers

```
SELECT fname
FROM Employee
WHERE fname REGEXP '^[0-9]+$';
```

ff. Get fname, lname from the employee table as separate rows

```
SELECT fname AS name FROM Employee
UNION ALL
SELECT lname FROM Employee;
```

gg. Create an empty table emptem with the same structure as emp

```
CREATE TABLE emptem LIKE emp;
```

hh. If there are two tables emp1 and emp2, and both have common records, fetch all the records, but common records only once

```
SELECT * FROM emp1
UNION
SELECT * FROM emp2;
```

ii. Extract only common records from two tables emp1 and emp2

```
SELECT * FROM emp1
INTERSECT
SELECT * FROM emp2;
```

jj. Retrieve all records of emp1 that should not be present in emp2

```
SELECT * FROM emp1
WHERE emp1.id NOT IN (SELECT id FROM emp2);
```

kk. Find rows that contain at least one of the two words ‘mysql’, ‘oracle’

```
SELECT *
FROM table_name
WHERE column_name LIKE '%mysql%' OR column_name LIKE '%oracle%';
```

ll. In a string attribute of the company schema, match the following using regular expression

i) Beginning of the string.

```
SELECT *
FROM Employee
WHERE column_name REGEXP '^pattern';
```

ii) Match any character (including carriage return and newline).

```
SELECT *
FROM Employee
WHERE column_name REGEXP '.';
```

iii) Match the end of a string.

```
SELECT *
FROM Employee
WHERE column_name REGEXP 'pattern$';
```

iv) Any sequence of zero or more characters.

```
SELECT *
FROM Employee
WHERE column_name REGEXP '.*';
```

v) Either of the sequences xy or abc.

```
SELECT *
FROM Employee
WHERE column_name REGEXP '(xy|abc)';
```