**Extra Lab Practice…**

**1.Introduction to sql…**

Lab-1:

-> CREATE TABLE students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(100) NOT NULL,

);

INSERT INTO `students` (`student\_id`, `student\_name`, `age`, `class`, `address`) VALUES (1, 'rutvi', '20', 'bca', 'abc');

->INSERT INTO `students` (`student\_id`, `student\_name`, `age`, `class`, `address`) VALUES (2, 'hiral', '15', '2th', 'pqr');

->INSERT INTO `students` (`student\_id`, `student\_name`, `age`, `class`, `address`) VALUES (3, 'priyanshi', '45', '1st', 'xyz');

->INSERT INTO `students` (`student\_id`, `student\_name`, `age`, `class`, `address`) VALUES (4, 'vaishvi', '56', 'bba', 'mno');

->INSERT INTO `students` (`student\_id`, `student\_name`, `age`, `class`, `address`) VALUES (5, 'abc', '50', 'bcom', 'qwty');

Lab-2:

->select \* from students;

**2.SQL Syntax…**

Lab-1:

->SELECT student\_name,age FROM students;

Lab-2:

->SELECT age FROM students where age>10;

**3.SQL Contraints…**

Lab-1:

-> CREATE TABLE teachers (

teacher\_id INT PRIMARY KEY,

teacher\_name VARCHAR(100) NOT NULL,

subject VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE

);

Lab-2:

-> CREATE TABLE students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(100) NOT NULL,

);

ALTER TABLE students

ADD CONSTRAINT fk\_teacher

FOREIGN KEY (teacher\_id) REFERENCES teachers(teacher\_id);

**4.** **Main SQL Commands and Sub-commands (DDL)..**

Lab-1**:**

->CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

course\_credits INT

);

Lab-2:

-> CREATE DATABASE university\_db;

**5. ALTER Command..**

Lab-1:

-> ALTER TABLE courses ADD COLUMN course\_duration VARCHAR(50);

Lab-2:

ALTER TABLE courses DROP COLUMN course\_credits;

**6.** **DROP Command..**

Lab-1**:**

**->**ALTER TABLE students DROP FOREIGN KEY fk\_teacher;

->DROP TABLE teachers;

Lab-2:

-> DROP TABLE students;

**7. Data Manipulation Language (DML)…**

Lab-1:

->INSERT INTO courses(course\_name)VALUES(‘rutvi’);

INSERT INTO courses(course\_name)VALUES(‘hiral’);

INSERT INTO courses(course\_name)VALUES(‘priyanshi’);

Lab-2:

-> UPDATE courses SET course\_duration = '6 months' WHERE course\_id = 1;

Lab-3:

-> DELETE FROM course WHERE course\_id = 1;

**8. Data Query Language (DQL)..**

Lab-1:

->select \* from courses;

Lab-2:

-> SELECT \* FROM courses ORDER BY course\_duration DESC;

Lab-3:

-> SELECT \* FROM courses LIMIT 2;

**9.** **Data Control Language (DCL)..**

Lab-1:

->GRANT SELECT ON courses TO user1;

Lab-2:

->REVOKE SELECT ON courses TO user1;

**10.** **Transaction Control Language (TCL)..**

Lab-1:

->START TRANSACTION;

INSERT INTO courses (course\_name, course\_duration)VALUES ('Networking', '3');

INSERT INTO courses (course\_name, course\_duration)VALUES ('Networking', '3');

COMMIT;

Lab-2:

-> INSERT INTO courses (course\_name, course\_duration)VALUES ('Networking', '3');

ROLLBACK;

Lab-3:

-> START TRANSACTION;

SAVEPOINT before\_insert;

INSERT INTO courses (course\_name, course\_duration)

VALUES ('Java Advanced', 12);

ROLLBACK TO before\_insert;

COMMIT;

**11.** **SQL Joins..**

Lab-1:

-> CREATE TABLE employess (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100)

);

CREATE TABLE department (

dpart\_id INT PRIMARY KEY,

dpat\_name VARCHAR(100),

emp\_id int

);

ALTER TABLE department ADD CONSTRAINT emp\_id FOREIGN KEY(emp  
\_id)REFERENCES employees(emp\_id);

SELECT

employees.emp\_id,

employees.emp\_name,

department.department\_name

FROM

employees

INNER JOIN

department ON employees.emp\_id = department.department\_id;

lab-2:

-> SELECT

department.department\_id,

department.department\_name,

employees.emp\_id,

employees.emp\_name

FROM

department

LEFT JOIN

employees ON department.emp\_id = employees.emp\_id;

**12.** **SQL Group By..**

Lab-1:

-> SELECT \*,

COUNT(\*) AS employee\_count

FROM

employees

GROUP BY

emp\_id;

Lab-2:

-> SELECT emp\_id, AVG(salary) AS average\_salary FROM department GROUP BY emp\_id LIMIT 1;

13. **SQL Stored Procedure..**

Lab-1:

-> DELIMITER $$

CREATE PROCEDURE GetEmployeesByDepartment(IN dept\_id INT)

BEGIN

SELECT \*

FROM employees

WHERE department\_id = dept\_id;

END $$

DELIMITER ;

Lab-2:

-> DELIMITER $$

CREATE PROCEDURE GetCourseDetails(IN input\_course\_id INT)

BEGIN

SELECT \*

FROM courses

WHERE course\_id = input\_course\_id;

END $$

DELIMITER ;

**14. SQL View..**

Lab-1:

-> CREATE VIEW employee\_department\_view AS

SELECT

e.emp\_id,

e.emp\_name,

e.salary,

d.department\_id,

d.department\_name

FROM

employees e

JOIN

department d ON d.department\_id = d.department\_id;

Lab-2:

-> CREATE VIEW employee\_department\_view AS

SELECT

e.emp\_id,

e.emp\_name,

e.salary,

d.department\_id,

d.department\_name

FROM

employees e

JOIN

department d ON d.department\_id = d.department\_id

WHERE

e.salary >= 50000;

**15.sql trigger..**

Lab-1:

-> CREATE TABLE employee\_log (

log\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_id INT,

emp\_name VARCHAR(100),

action\_time DATETIME,

action\_type VARCHAR(20)

);

DELIMITER $$

CREATE TRIGGER log\_new\_employee

AFTER INSERT ON employees

FOR EACH ROW

BEGIN

INSERT INTO employee\_log(emp\_id, emp\_name, action\_time, action\_type)

VALUES (NEW.emp\_id, NEW.emp\_name, NOW(), 'INSERT');

END $$

DELIMITER ;

Lab-2:

-> ALTER TABLE employees

ADD COLUMN last\_modified DATETIME;

DELIMITER $$

CREATE TRIGGER update\_last\_modified

BEFORE UPDATE ON employees

FOR EACH ROW

BEGIN

SET NEW.last\_modified = NOW();

END $$

DELIMITER ;

16. **Introduction to PL/SQL..**

Lab-1:

->