**DDL AND DML**

CREATE TABLE EMP (

EMPNO NUMBER(4),

ENAME VARCHAR2(10),

DESIGNATION VARCHAR2(10),

SALARY NUMBER(8,2)

);

DESC EMP;

ALTER TABLE EMP MODIFY EMPNO NUMBER(6);

ALTER TABLE EMP ADD (

DOB DATE,

DOJ DATE

);

ALTER TABLE EMP DROP COLUMN DOJ;

ALTER TABLE EMP DROP COLUMN DOB;

ALTER TABLE EMP DROP COLUMN QUALIFICATION;

CREATE TABLE Student (

Id INTEGER,

LastName VARCHAR2(35) NOT NULL,

FirstName VARCHAR2(35) NOT NULL,

City VARCHAR2(35)

);

INSERT INTO Student VALUES (1, 'Hanks', 'Peter', 'New York');

-- This will fail due to NOT NULL constraint on LastName

-- INSERT INTO Student VALUES (2, NULL, 'Amanda', 'Florida');

CREATE TABLE ShirtBrands (

Id INTEGER,

BrandName VARCHAR2(40) UNIQUE,

Size VARCHAR2(30)

);

INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES (1, 'Pantaloons', '38');

INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES (2, 'Cantabil', '40');

-- This will fail due to duplicate BrandName 'Cantabil'

-- INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES (3, 'Raymond', '38');

-- INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES (4, 'Cantabil', '40');

CREATE TABLE Persons (

ID INT NOT NULL,

Name VARCHAR2(45) NOT NULL,

Age INT CHECK (Age >= 18)

);

INSERT INTO Persons(ID, Name, Age) VALUES (1, 'Robert', 28);

INSERT INTO Persons(ID, Name, Age) VALUES (2, 'Joseph', 35);

INSERT INTO Persons(ID, Name, Age) VALUES (3, 'Peter', 40);

-- This will fail due to CHECK constraint on Age

-- INSERT INTO Persons(ID, Name, Age) VALUES (4, 'Robert', 15);

CREATE TABLE Persons (

ID INT PRIMARY KEY,

Name VARCHAR2(45) NOT NULL,

Age INT,

City VARCHAR2(25)

);

INSERT INTO Persons(ID, Name, Age, City) VALUES (1, 'Robert', 15, 'Florida');

INSERT INTO Persons(ID, Name, Age, City) VALUES (2, 'Joseph', 35, 'California');

INSERT INTO Persons(ID, Name, Age, City) VALUES (3, 'Peter', 40, 'Alaska');

-- This will fail because ID=1 already exists (PRIMARY KEY violation)

-- INSERT INTO Persons(ID, Name, Age, City) VALUES (1, 'Stephen', 15, 'Florida');

**GUI APP**

import tkinter as tk

from tkinter import \*

import mysql.connector

def submitact():

user = Username.get()

passw = password.get()

print(f"The name entered by you is {user} {passw}")

logintodb(user, passw)

def logintodb(user, passw):

try:

# Connecting to database

if passw:

db = mysql.connector.connect(

host="localhost",

user=user,

password=passw,

database="College"

)

else:

db = mysql.connector.connect(

host="localhost",

user=user,

database="College"

)

cursor = db.cursor()

savequery = "SELECT \* FROM STUDENT"

cursor.execute(savequery)

myresult = cursor.fetchall()

for x in myresult:

print(x)

print("Query Executed successfully")

except Exception as e:

print("Error occurred:", e)

# GUI setup

root = tk.Tk()

root.geometry("300x200")

root.title("DBMS Login Page")

# Username label and entry

lblfrstrow = tk.Label(root, text="Username -")

lblfrstrow.place(x=50, y=20)

Username = tk.Entry(root, width=35)

Username.place(x=150, y=20, width=100)

# Password label and entry

lblsecrow = tk.Label(root, text="Password -")

lblsecrow.place(x=50, y=50)

password = tk.Entry(root, show="\*", width=35)

password.place(x=150, y=50, width=100)

# Submit button

submitbtn = tk.Button(root, text="Login", bg='blue', fg='white', command=submitact)

submitbtn.place(x=150, y=100, width=55)

root.mainloop()

**WHERE**

SELECT \* FROM TableName WHERE condition;

SELECT \* FROM officers

WHERE address = 'Lucknow' AND officer\_id < 5;

SELECT \* FROM officers

WHERE address = 'Lucknow' OR address = 'Mau';

SELECT \* FROM officers

WHERE (address = 'Mau' AND officer\_name = 'Ajeet')

OR (officer\_id < 5);

SELECT COUNT(emp\_name) FROM employees;

SELECT COUNT(\*) FROM employees

WHERE emp\_age > 32;

SELECT SUM(working\_hours) AS "Total working hours"

FROM employees;

SELECT AVG(working\_hours) AS Avg\_working\_hours

FROM employees;

**NOSQL**

mongosh "mongodb://localhost:27017"

use mydbnew

db.details.insertOne({ "website": "mywebsite" })

db.details.find()

**TRIGGER**

CREATE OR REPLACE TRIGGER trigger\_name

[BEFORE | AFTER] {INSERT | UPDATE | DELETE}

ON table\_name

[FOR EACH ROW | FOR EACH STATEMENT]

DECLARE

-- Variable declarations

BEGIN

-- Trigger logic

EXCEPTION

-- Exception handling (optional)

END;

CREATE TABLE poo (

rno NUMBER(5),

name VARCHAR2(10)

);

INSERT INTO poo VALUES (1, 'kala');

INSERT INTO poo VALUES (2, 'priya');

SELECT \* FROM poo;

CREATE OR REPLACE TRIGGER pool

BEFORE INSERT ON poo

FOR EACH ROW

DECLARE

existing\_rno poo.rno%TYPE;

CURSOR c IS SELECT rno FROM poo;

BEGIN

OPEN c;

LOOP

FETCH c INTO existing\_rno;

EXIT WHEN c%NOTFOUND;

IF :NEW.rno = existing\_rno THEN

RAISE\_APPLICATION\_ERROR(-20005, 'rno already exists');

END IF;

END LOOP;

CLOSE c;

END;

INSERT INTO poo VALUES (1, 'kala');

**INNER JOIN**

SELECT

officers.officer\_name,

officers.address,

students.course\_name

FROM

officers

INNER JOIN

students

ON

officers.officer\_id = students.student\_id;

SELECT

emp\_name,

city,

income

FROM

employees

WHERE

emp\_id IN (SELECT emp\_id FROM employees);

*SELECT emp\_name, city, income*

*FROM employees*

*WHERE emp\_id IN (*

*SELECT emp\_id*

*FROM employees*

*WHERE income > 50000*

*);*

**VIEW**

CREATE TABLE EMPLOYEE (

EMPLOYEE\_NAME VARCHAR2(10),

EMPLOYEE\_NO NUMBER(8),

DEPT\_NAME VARCHAR2(10),

DEPT\_NO NUMBER(5),

DATE\_OF\_JOIN DATE

);

DESC EMPLOYEE;

CREATE VIEW EMPVIEW AS

SELECT EMPLOYEE\_NAME, EMPLOYEE\_NO, DEPT\_NAME, DEPT\_NO, DATE\_OF\_JOIN

FROM EMPLOYEE;

DESC EMPVIEW;

SELECT \* FROM EMPVIEW;

INSERT INTO EMPVIEW

VALUES ('SRI', 120, 'CSE', 67, TO\_DATE('16-NOV-1981','DD-MON-YYYY'));

SELECT \* FROM EMPVIEW;

SELECT \* FROM EMPLOYEE;

DELETE FROM EMPVIEW WHERE EMPLOYEE\_NAME = 'SRI';

UPDATE EMPVIEW SET EMPLOYEE\_NAME = 'KAVI'

WHERE EMPLOYEE\_NAME = 'RAVI';

DROP VIEW EMPVIEW;

CREATE DATABASE indexes;

USE indexes;

CREATE TABLE employees (

employee\_id INT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

device\_serial VARCHAR(15),

salary INT

);

INSERT INTO employees VALUES

(1, 'John', 'Smith', 'ABC123', 60000),

(2, 'Jane', 'Doe', 'DEF456', 65000),

(3, 'Bob', 'Johnson', 'GHI789', 70000),

(4, 'Sally', 'Fields', 'JKL012', 75000),

(5, 'Michael', 'Smith', 'MNO345', 80000),

(6, 'Emily', 'Jones', 'PQR678', 85000),

(7, 'David', 'Williams', 'STU901', 90000),

(8, 'Sarah', 'Johnson', 'VWX234', 95000),

(9, 'James', 'Brown', 'YZA567', 100000);

CREATE INDEX salary ON employees(salary);

EXPLAIN SELECT \* FROM employees WHERE salary = 100000;

**DCL and TCL**

GRANT SELECT ON employee TO user1;

-- Output:

-- Command Successfully Completed

REVOKE SELECT ON employee FROM user1;

-- Output:

-- Command Successfully Completed

SAVEPOINT S1;

-- Output:

-- Savepoint created.

SELECT \* FROM EMP;

-- Output:

-- EMPNO | ENAME | DESIGNATION | SALARY

-- 101 | NAGARAJAN | LECTURER | 16000

-- 102 | SARAVANAN | ASST. PROF | 16000

-- 104 | CHINNI | HOD, PROF | 45000

INSERT INTO EMP VALUES (105, 'PARTHASAR', 'STUDENT', 100);

-- Output:

-- 1 row created.

SELECT \* FROM EMP;

-- Output includes newly inserted row:

-- 105 | PARTHASAR | STUDENT | 100

ROLLBACK TO S1;

-- Output:

-- Rollback complete.

SELECT \* FROM EMP;

-- Output: Record with EMPNO 105 is removed

-- 101 | NAGARAJAN | LECTURER | 16000

-- 102 | SARAVANAN | ASST. PROF | 16000

-- 104 | CHINNI | HOD, PROF | 45000

COMMIT;

-- Output:

-- Commit complete

**XML**

CREATE TABLE person (

person\_id INT NOT NULL PRIMARY KEY,

fname VARCHAR(40) NULL,

lname VARCHAR(40) NULL,

created TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

<list>

<person person\_id="1" fname="Kapek" lname="Sainnouine"/>

<person person\_id="2" fname="Sajon" lname="Rondela"/>

<person person\_id="3">

<fname>Likame</fname>

<lname>Örrtmons</lname>

</person>

<person person\_id="4">

<fname>Slar</fname>

<lname>Manlanth</lname>

</person>

<person>

<field name="person\_id">5</field>

<field name="fname">Stoma</field>

<field name="lname">Milu</field>

</person>

<person>

<field name="person\_id">6</field>

<field name="fname">Nirtam</field>

<field name="lname">Sklöd</field>

</person>

<person person\_id="7">

<fname>Sungam</fname>

<lname>Dulbåd</lname>

</person>

<person person\_id="8" fname="Sraref" lname="Encmelt"/>

</list>

LOAD XML LOCAL INFILE 'C:/db/person.xml'

INTO TABLE person

ROWS IDENTIFIED BY '<person>';

SELECT \* FROM person;

SELECT

ExtractValue('<?xml version="1.0"?><person><fname>Ravi</fname></person>', '/person/fname') AS FirstName;

**NATURAL JOIN**

SELECT [column\_names | \*]

FROM table\_name1

NATURAL JOIN table\_name2;

/\* -- Table name: customer -- \*/

CREATE TABLE customer (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(55),

account INT,

email VARCHAR(55)

);

/\* -- Table name: balance -- \*/

CREATE TABLE balance (

id INT AUTO\_INCREMENT PRIMARY KEY,

account INT,

balance FLOAT(10, 2)

);

/\* -- Data for customer table -- \*/

INSERT INTO customer(customer\_name, account, email)

VALUES

('Stephen', 1030, 'stephen@javatpoint.com'),

('Jenifer', 2035, 'jenifer@javatpoint.com'),

('Mathew', 5564, 'mathew@javatpoint.com'),

('Smith', 4534, 'smith@javatpoint.com'),

('David', 7648, 'david@javatpoint.com');

/\* -- Data for balance table -- \*/

INSERT INTO balance(account, balance)

VALUES

(1030, 50000.00),

(2035, 230000.00),

(5564, 125000.00),

(4534, 80000.00),

(7648, 45000.00);

SELECT customer\_name, account

FROM customer NATURAL JOIN balance;

SELECT columns

FROM table1

RIGHT [OUTER] JOIN table2

ON table1.column = table2.column;

/\* -- Table: officers -- \*/

CREATE TABLE officers (

officer\_id INT,

officer\_name VARCHAR(100),

address VARCHAR(100)

);

/\* -- Table: students -- \*/

CREATE TABLE students (

student\_id INT,

student\_name VARCHAR(100),

course\_name VARCHAR(100)

);

INSERT INTO officers (officer\_id, officer\_name, address)

VALUES (1, 'John Doe', 'New York'),

(2, 'Jane Smith', 'Chicago');

INSERT INTO students (student\_id, student\_name, course\_name)

VALUES (1, 'Tom', 'Math'),

(2, 'Jerry', 'Science'),

(3, 'Spike', 'History');

SELECT officers.officer\_name, officers.address, students.course\_name, students.student\_name

FROM officers RIGHT JOIN students ON officers.officer\_id = students.student\_id;

SELECT column\_name(s)

FROM table\_name1, table\_name2

WHERE table\_name1.column\_name = table\_name2.column\_name;

SELECT cust.customer\_name, bal.balance

FROM customer AS cust, balance AS bal

WHERE cust.account = bal.account;

**PROCEDURE**

SQL> SET SERVEROUTPUT ON;

SQL> CREATE OR REPLACE PROCEDURE PROC AS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello from procedure...');

END;

SQL> EXECUTE PROC;

SQL> CREATE TABLE student (

regno NUMBER(4),

name VARCHAR2(20),

mark1 NUMBER(3),

mark2 NUMBER(3),

mark3 NUMBER(3),

mark4 NUMBER(3),

mark5 NUMBER(3)

);

SQL> INSERT INTO student VALUES (101, 'Priya', 78, 88, 77, 60, 89);

SQL> INSERT INTO student VALUES (102, 'Surya', 99, 77, 69, 81, 99);

SQL> INSERT INTO student VALUES (103, 'Suryapriya', 100, 90, 97, 89, 91);

SQL> DECLARE

CURSOR c\_mark IS

SELECT \* FROM student

WHERE mark1 >= 40 AND mark2 >= 40 AND mark3 >= 40 AND mark4 >= 40 AND mark5 >= 40;

tot NUMBER(3);

ave NUMBER(5,2);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('regno name mark1 mark2 mark3 mark4 mark5 total average');

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------');

FOR student IN c\_mark LOOP

tot := student.mark1 + student.mark2 + student.mark3 + student.mark4 + student.mark5;

ave := tot / 5;

DBMS\_OUTPUT.PUT\_LINE(

student.regno || RPAD(student.name, 15) || RPAD(student.mark1, 6) ||

RPAD(student.mark2, 6) || RPAD(student.mark3, 6) || RPAD(student.mark4, 6) ||

RPAD(student.mark5, 6) || RPAD(tot, 8) || RPAD(ave, 5)

);

END LOOP;

END;

SQL> CREATE TABLE phonebook (

phone\_no NUMBER(6) PRIMARY KEY,

username VARCHAR2(30),

doorno VARCHAR2(10),

street VARCHAR2(30),

place VARCHAR2(30),

pincode CHAR(6)

);

SQL> INSERT INTO phonebook VALUES (20312, 'Vijay', '120/5D', 'Bharathi Street', 'NGO Colony', '629002');

SQL> INSERT INTO phonebook VALUES (29467, 'Vasanth', '39D4', 'RK Bhavan', 'Sarakkal Vilai', '629002');

SQL> CREATE OR REPLACE FUNCTION findAddress(phone IN NUMBER) RETURN VARCHAR2 AS

address VARCHAR2(100);

BEGIN

SELECT username || ',' || doorno || ',' || street || ',' || place || ',' || pincode

INTO address

FROM phonebook

WHERE phone\_no = phone;

RETURN address;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Address not found';

END;

SQL> DECLARE

address VARCHAR2(100);

BEGIN

address := findAddress(20312);

DBMS\_OUTPUT.PUT\_LINE(address);

END;

**FOREIGN**

-- DEPARTMENT

CREATE TABLE Department(

Id INT PRIMARY KEY,

Name NVARCHAR(50)

);

-- Insert some test data in Department Table

INSERT INTO Department VALUES (10, 'IT');

INSERT INTO Department VALUES (20, 'HR');

INSERT INTO Department VALUES (30, 'INFRA');

-- EMPLOYEES

CREATE TABLE Employees(

Id INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

DepartmentID INT

);

-- Adding the Foreign Key Constraint

ALTER TABLE Employees ADD FOREIGN KEY (DepartmentId) REFERENCES Department(Id);

-- Insert some test data in Employees Table

INSERT INTO Employees VALUES (101, 'Anurag', 10);

INSERT INTO Employees VALUES (102, 'Pranaya', 20);

INSERT INTO Employees VALUES (103, 'Hina', 30);

-- Delete from Parent Table

DELETE FROM Department WHERE Id = 10;