Starting Steps:

Go to Command Prompt and type sqlplus

Type user as ‘system’ and the password that you set.

1.Create a table called Employee & execute the following.

Employee(EMPNO,ENAME,JOB, MANAGER_NO, SAL, COMMISSION)

1. Create a user and grant all permissions to the user.

2. Insert the any three records in the employee table contains attributes

EMPNO,ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback.

Check the result.

3. Add primary key constraint and not null constraint to the employee table.

4. Insert null values to the employee table and verify the result.

1.

```sql
-- Create a new user
CREATE USER Denzil IDENTIFIED BY MyPassword;

-- Grant all permissions to the new user
GRANT ALL PRIVILEGES TO Denzil;
```

2.

```sql
-- Create the Employee table
CREATE TABLE Employee (
    EMPNO NUMBER(5),
    ENAME VARCHAR2(30),
    JOB VARCHAR2(30),
    MANAGER_NO NUMBER(5),
    SAL DECIMAL(10),
    COMMISSION DECIMAL(10)
);

-- Insert records
INSERT INTO Employee VALUES ('001', 'AA', 'Job1', '1001', 25000, 2000);

INSERT INTO Employee VALUES ('002', 'BB', 'Job2', '1002', 30000, 2500);

INSERT INTO Employee VALUES ('003', 'CC', 'Job3', '1003', 35000, 3000);

--Display
DESC Employee;
```

```
SELECT * FROM Employee;
```

3.

```
-- Add NOT NULL constraints to columns
ALTER TABLE Employee
MODIFY(EMPNO NUMBER(5) PRIMARY KEY, ENAME VARCHAR2(30) NOT NULL);
```

4.

```
-- Insert records with null values(This should give an Error)
INSERT INTO Employee VALUES ('004', NULL, 'Job4', NULL, NULL, NULL);
```

2.Create a table called Employee that contain attributes EMPNO,ENAME,JOB, MGR,SAL &

execute the following.

1. Add a column commission with domain to the Employee table.

2. Insert any five records into the table.

3. Update the column details of job

4. Rename the column of Employ table using alter command.

5. Delete the employee whose Empno is 105.

1.

```
-- Create the Employee table
CREATE TABLE Employee (
    EMPNO NUMBER(5),
    ENAME VARCHAR2(30),
    JOB VARCHAR2(30),
    MGR_NO CHAR(5),
    SAL NUMBER(10)
);

--Display
DESC Employee;


-- Add the COMMISSION column
ALTER TABLE Employee
ADD (COMMISSION NUMBER(6));

--Display
DESC Employee;
```

2.

```sql
-- Insert records into the Employee table

INSERT INTO Employee VALUES ('101', 'AA', 'Manager', '102', 50000, 5000);

INSERT INTO Employee VALUES ('102', 'AB', 'Assistant Manager', '103', 40000, 4000);

INSERT INTO Employee VALUES ('103', 'AC', 'Clerk', '104', 30000, 3000);

INSERT INTO Employee VALUES ('104', 'AD', 'Secretary', '105', 25000, 2500);

INSERT INTO Employee VALUES ('105', 'AE', 'Intern', '106', 20000, 2000);

--Display
SELECT * FROM Employee;
```

3.

```sql
-- Update the JOB column details
UPDATE Employee
SET JOB = 'HR'
WHERE EMPNO = '101';

--Display
SELECT * FROM Employee;
```

4.

```sql
-- Rename the column ENAME to FULL_NAME
ALTER TABLE Employee
RENAME COLUMN MGR_NO TO MANAGER_NO;

--Display
DESC Employee;
```

5.

```sql
-- Delete the employee with EMPNO = '105'
DELETE FROM Employee
WHERE EMPNO = '105';

--Display
SELECT * FROM Employee;
```

3. Queries using aggregate functions(COUNT,AVG,MIN,MAX,SUM),Group by,Orderby.

Employee(E_id, E_name, Age, Salary)

1. Create Employee table containing all Records E_id, E_name, Age, Salary.

2. Count number of employee names from employeetable

3. Find the Maximum age from employee table.

4. Find the Minimum age from employeetable.

5. Find salaries of employee in Ascending Order.

6. Find grouped salaries of employees.

1.

```
-- Create the Employee table
CREATE TABLE Employee (
    E_id NUMBER(5) PRIMARY KEY,
    E_name VARCHAR(30),
    Age NUMBER(5),
    Salary NUMBER(10)
);

--Display
DESC Employee;

-- Insert records into the Employee table
INSERT INTO Employee VALUES (1, 'AA', 30, 55000);

INSERT INTO Employee VALUES (2, 'AB', 25, 45000);

INSERT INTO Employee VALUES (3, 'AC', 35, 60000);

INSERT INTO Employee VALUES (4, 'AD', 28, 50000);

INSERT INTO Employee VALUES (5, 'AE', 40, 70000);

--Display
SELECT * FROM Employee;
```

2.

```
-- Count the number of employees
SELECT COUNT(E_name) AS NumberOfEmployees
FROM Employee;
```

3.

```sql
-- Find the maximum age
SELECT MAX(Age) AS MaxAge
FROM Employee;
```

4.

```sql
-- Find the minimum age
SELECT MIN(Age) AS MinAge
FROM Employee;
```

5.

```sql
-- Find salaries in ascending order
SELECT Salary
FROM Employee
ORDER BY Salary ASC;
```

4. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old & new Salary.

CUSTOMERS(ID,NAME,AGE,ADDRESS,SALARY)

1. Create the custormer table

```sql
-- Create the CUSTOMERS table
CREATE TABLE CUSTOMERS (
    ID INT ,
    NAME VARCHAR(10),
    AGE INT,
    ADDRESS VARCHAR2(30),
    SALARY DECIMAL(10, 2)
);
```

2.Insert 5 Records

```sql
-- Insert records into the CUSTOMERS table
INSERT INTO CUSTOMERS VALUES (1, 'AA', 30, 'Manglore', 55000.00);

INSERT INTO CUSTOMERS VALUES (2, 'AB', 25, 'Vitla', 45000.00);

INSERT INTO CUSTOMERS VALUES (3, 'AC', 35, 'Banglore', 60000.00);

INSERT INTO CUSTOMERS VALUES (4, 'AD', 28, 'Dehli', 50000.00);
INSERT INTO CUSTOMERS VALUES (5, 'AE', 40, 'Mumbai', 70000.00);

--Display
DESC Customers;

SELECT * FROM Customers;
```

3.Set serveroutput on

```
-- Enable server output to view results
SET SERVEROUTPUT ON;
```

4.Create the Trigger

```
CREATE OR REPLACE TRIGGER display_salary_changes
AFTER DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE sal_diff number;
BEGIN
sal_diff := :NEW.salary - :OLD.salary;
dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary);
dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

4.Insert and Update values

```
-- Insert a new row into the CUSTOMERS table
INSERT INTO CUSTOMERS VALUES (6, 'AF', 45, 'Mysore', 65000.00);

-- Update a row to trigger salary change
UPDATE CUSTOMERS
SET SALARY = 72000.00
WHERE ID = 5;
```

5. Create cursor for Employee table & extract the values from the table. Declare the variables

,Open the cursor & extrct the values from the cursor. Close the cursor.

Employee(E_id, E_name, Age, Salary)

1.Create Table

```
-- Create the Employee table
```

```
CREATE TABLE Employee (
    E_id INT,
    E_name VARCHAR(20),
    Age NUMBER(5),
    Salary DECIMAL(10, 2)
);

-- Insert records into the Employee table
INSERT INTO Employee VALUES (1, 'AA', 30, 55000);

INSERT INTO Employee VALUES (2, 'AB', 25, 45000);

INSERT INTO Employee VALUES (3, 'AC', 35, 60000);

INSERT INTO Employee VALUES (4, 'AD', 28, 50000);

INSERT INTO Employee VALUES (5, 'AE', 40, 70000);

--Display
DESC Employee;

SELECT * FROM Employee;
```

2.Set serveroutput on

```
-- Enable server output to view results
SET SERVEROUTPUT ON;
```

3. Create Cursor

```
DECLARE
    -- Declare variables to hold cursor values
    empid Employee.E_id%TYPE;
    empname Employee.E_name%TYPE;
    empsal Employee.salary%TYPE;

    -- Declare the cursor
    CURSOR emp_cursor IS
        SELECT E_id, E_name, Salary
        FROM Employee;
BEGIN
    -- Open the cursor
    OPEN emp_cursor;

    -- Loop to fetch and display each row
    LOOP
```

```
        FETCH emp_cursor INTO empid, empname, empsal;
        EXIT WHEN emp_cursor%NOTFOUND;

        IF(empsal>50000)THEN
            DBMS_OUTPUT.PUT_LINE(empid || ' ' || empname || ' ' || empsal);
        ELSE
            DBMS_OUTPUT.PUT_LINE(empname || 'Salary Less Than 50000');
        END IF;
    END LOOP;

    -- Close the cursor
    CLOSE emp_cursor;
END;
/
```

6. Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

1.Create Two Tables

```
-- Create the N_RollCall table
CREATE TABLE N_RollCall (
    USN INT,
    Name VARCHAR(30),
    Age NUMBER(5)
);

-- Create the O_RollCall table
CREATE TABLE O_RollCall (
    USN INT,
    Name VARCHAR(30),
    Age NUMBER(5)
);
```

2. Insert Values

```
-- Insert records into N_RollCall
INSERT INTO N_RollCall VALUES (1, 'AA', 25);

INSERT INTO N_RollCall VALUES (2, 'AB', 18);

INSERT INTO N_RollCall VALUES (3, 'AC', 40);

-- Insert records into O_RollCall
```

```
INSERT INTO O_RollCall VALUES (2, 'AB', 18);

INSERT INTO O_RollCall VALUES (4, 'AD', 32);

INSERT INTO O_RollCall VALUES (5, 'AE', 36);

--Display

SELECT * FROM N_RollCall;

SELECT * FROM O_RollCall;
```

3.Set serveroutput on

```
-- Enable server output to view results
SET SERVEROUTPUT ON;
```

4. Create Cursor

```
DECLARE
    -- Cursor to select records from O_RollCall based on ID
    CURSOR o_cursor (roll CHARACTER) IS
        SELECT * FROM O_RollCall
        WHERE USN = roll;

    -- Cursor to select all records from N_RollCall
    CURSOR n_cursor IS
        SELECT * FROM N_RollCall;

    -- Record type variables for N_RollCall and O_RollCall
    n_record N_RollCall%ROWTYPE;
    o_record O_RollCall%ROWTYPE;
BEGIN
    -- Open the cursor to fetch records from N_RollCall
    OPEN n_cursor;

    -- Loop through each record in N_RollCall
    LOOP
        FETCH n_cursor INTO n_record;
        EXIT WHEN n_cursor%NOTFOUND;

        -- Display the record details
        DBMS_OUTPUT.PUT_LINE(n_record.USN || ' ' || n_record.Name || ' ' ||
n_record.Age);

        -- Open the o_cursor with the ID from N_RollCall
```

```
         OPEN o_cursor(n_record.USN);
        FETCH o_cursor INTO o_record;

        -- Check if the record exists in O_RollCall
        IF o_cursor%NOTFOUND THEN
            -- Insert the record if it does not exist
            INSERT INTO O_RollCall VALUES (n_record.USN, n_record.Name, 
n_record.Age);
        ELSE
            -- Display a message if the record already exists
            DBMS_OUTPUT.PUT_LINE(o_record.USN || ' data already exists');
        END IF;

        -- Close the o_cursor
        CLOSE o_cursor;
    END LOOP;

    -- Close the n_cursor
    CLOSE n_cursor;

    DBMS_OUTPUT.PUT_LINE('Data merge completed.');
END;
/
```

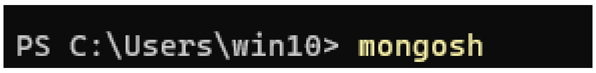
5. Display Merged values

```
--Display

SELECT * FROM O_RollCall;
```

7. Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations.

1.Creation (Type mongosh in the command terminal)



```
PS C:\Users\win10> mongosh
```

2. Initialize

```
// Switch to a database (it will be created if it does not exist)
use myDatabase;
```

3. CRUD Operations

```
// Create a collection named "employees"
db.createCollection("employees");
```

```
// Insert a single document
db.employees.insertOne({
    empID: 1,
    name: "AA",
    position: "Developer",
    salary: 50000
});

// Insert multiple documents
db.employees.insertMany([
    { empID: 2, name: "AB", position: "Designer", salary: 45000 },
    { empID: 3, name: "AC", position: "Manager", salary: 60000 }
]);

// Find all documents
db.employees.find();

// Find a document by a specific field
db.employees.find({ empID: 1 });

// Update a single document
db.employees.updateOne(
    { empID: 1 },
    { $set: { salary: 55000 } }
);

//Display Changes
db.employees.find({ empID: 1 });

// Update multiple documents
db.employees.updateMany(
    {},
    { $set: { status:"Active" } }
);

//Display Changes
db.employees.find({ status:"Active" });

// Delete a single document
db.employees.deleteOne({ empID: 1 });

db.employees.find().limit(1);
```