Bachelor of Science in Computer Science and Information Technology

Laboratory Report On

Advanced Database

Submitted By :

Name : Basanta Rai
Semester : 8th
Section : B
Rollno : 23473/076

Submitted To :

Department Of Computer Science & IT

Lab 1

a. **Create table employee (name, department-id, job title)**

SQL QUERY:

CREATE DATABASE employee_db;

```
~/Code/database-docker-compose ⨗ docker exec -it mysqldb bash
bash-4.4# mysqli -u root -p
bash: mysqli: command not found
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 26
Server version: 8.0.34 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE employee_db;
Query OK, 1 row affected (0.01 sec)

mysql>
```

SQL QUERY:

CREATE TABLE employee(name VARCHAR(30), department_id INT, job_title VARCHAR(20));

```
mysql> USE employee_db;
Database changed
mysql> CREATE TABLE employee(name VARCHAR(30), department_id INT, job_title VARCHAR(20));
Query OK, 0 rows affected (0.02 sec)

mysql>
```

b. **Add primary key emp_id to employee**

SQL QUERY:

ALTER TABLE employee ADD emp_id INT PRIMARY KEY;

```
mysql> ALTER TABLE employee ADD emp_id INT PRIMARY KEY;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

c. Change job_title varchar(20) to varchar(30)

SQL QUERY:

ALTER TABLE employee MODIFY COLUMN job_title VARCHAR(30);

```
mysql> ALTER TABLE employee MODIFY COLUMN job_title VARCHAR(30);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

d. Insert 10 rows to employee

SQL QUERY:

INSERT INTO employee (emp_id, name, department_id, job_title) VALUES (1, "Basanta Rai", 3, "Intern"), (2, "Manjeet Shrestha", 4, "Developer"), (3, "Robin Devkota", 3, "Intern"), (4, "Sunil Chaudhary", 6, "Developer"), (5, "John Doe", 6, "Developer"), (6, "Jane Doe", 6, "Developer"), (7, "John Wick", 6, "Developer"), (8, "Primeagen", 6, "Developer"), (9, "TJ", 6, "Developer"), (10, "Tsoding", 6, "Developer");

```
mysql> INSERT INTO employee (emp_id, name, department_id, job_title) VALUES (1, "Basanta Rai", 3, "Intern"), (2, "Manjeet Shrestha", 4, "Developer"), (3, "Robin Devkota", 3, "Intern"), (4, "Sunil Chaudhary", 6, "Developer"), (5, "John Doe", 6, "Developer"), (6, "Jane Doe", 6, "Developer"), (7, "John Wick", 6, "Developer"), (8, "Primeagen", 6, "Developer"), (9, "TJ", 6, "Developer"), (10, "Tsoding", 6, "Developer");
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
```

e. Update emp_name whose emp_id is 5

SQL QUERY:

UPDATE employee SET name="John Caramack" WHERE emp_id=5;

```
mysql> UPDATE employee SET name="John Caramack" WHERE emp_id=5;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

f. Add foreign key department_id as reference key

SQL QUERY:

CREATE TABLE department(id INT PRIMARY KEY, name VARCHAR(20));

```
mysql> CREATE TABLE department(id INT PRIMARY KEY, name VARCHAR(20));
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

ALTER TABLE employee ADD CONSTRAINT fk_department FOREIGN KEY (department_id) REFERENCES department(id);

```
mysql> ALTER TABLE employee  ADD CONSTRAINT fk_department  FOREIGN KEY (department_id)  REFERENCES department(id);
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Lab 2

a. **Create table department**

SQL QUERY:

CREATE TABLE department(id INT PRIMARY KEY, name VARCHAR(20));

```
mysql> CREATE TABLE department(id INT PRIMARY KEY, name VARCHAR(20));
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

ALTER TABLE department ADD dept_type varchar(20);

```
mysql> ALTER TABLE department ADD dept_type varchar(20);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

b. **Insert Data**

SQL QUERY:

INSERT INTO department(id, name, dept_type)
VALUES (1, "React", "Senior"), (2, "Node", "Junior"), (3, "React Native", "Senior"),
(4, "React", "Junior"), (5, "Project Manager", "Senior"),
(6, "Project Manager", "Junior") ;

```
mysql> INSERT INTO department(id, name, dept_type)
    -> VALUES (1, "React", "Senior"), (2, "Node", "Junior"), (3, "React Native", "Senior"),
    -> (4, "React", "Junior"), (5, "Project Manager", "Senior"),
    -> (6, "Project Manager", "Junior") ;
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

c. **Update dept_type whose department_id =2**

SQL QUERY:

UPDATE department SET dept_type = "Mid Level" WHERE department_id = 5;

```
mysql> UPDATE department SET dept_type = "Mid Level" WHERE id=5;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Lab 3 : Create Trigger for table students.

SQL QUERY:

CREATE TABLE student(id INT PRIMARY KEY, name varchar(20), department_id INT);

```
mysql> CREATE TABLE student(id INT PRIMARY KEY, name varchar(20), department_id INT);
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

```
CREATE TABLE student_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    operation VARCHAR(10),
    operation_time DATETIME
    );
```

```
mysql> CREATE TABLE student_log (
    ->  log_id INT AUTO_INCREMENT PRIMARY KEY,
    -> student_id INT,
    -> operation VARCHAR(10),
    -> operation_time DATETIME
    -> );
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

```
DELIMITER //
    CREATE TRIGGER after_student_insert
    AFTER INSERT ON student
    FOR EACH ROW
    BEGIN
        INSERT INTO student_log (student_id, operation, operation_time)
        VALUES
        (NEW.id, 'INSERT', NOW());
    END//
DELIMITER ;
```

```
mysql> DELIMITER //
mysql> CREATE TRIGGER after_student_insert
    -> AFTER INSERT ON student
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO student_log (student_id, operation, operation_time)
    -> VALUES
    -> (NEW.id, 'INSERT', NOW());
    -> END//
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
```

SQL QUERY:

INSERT INTO student (id, name, department_id) VALUES (1 ,"Basanta Rai", 01);

```
mysql> INSERT INTO student (id,  name, department_id) VALUES (1 ,"Basanta Rai",  01);
Query OK, 1 row affected (0.01 sec)
```

SQL QUERY:

SELECT * FROM student_log;

```
mysql> SELECT * FROM student_log;
+--------+------------+-----------+---------------------+
| log_id | student_id | operation | operation_time      |
+--------+------------+-----------+---------------------+
|      1 |          1 | INSERT    | 2024-06-09 06:16:26 |
+--------+------------+-----------+---------------------+
1 row in set (0.00 sec)
```

Lab 4 : Creating Index

SQL QUERY:

```
CREATE TABLE employees(
        name VARCHAR(25) NOT NULL,
        address VARCHAR(25) NOT NULL,
        job_title VARCHAr(20) NOT NULL,
        department_id INT NOT NULL,
        salary DECIMAL(10,2) NOT NULL
```

```
mysql> CREATE TABLE employees(
    -> name VARCHAR(25) NOT NULL,
    -> address VARCHAR(25) NOT NULL,
    -> job_title VARCHAr(20) NOT NULL,
    -> department_id INT NOT NULL,
    -> salary DECIMAL(10,2) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

```
CREATE INDEX employee_address on employees(address);
```

```
mysql> CREATE INDEX employee_address on employees(address);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Lab 5 : Demonstrate deductive database query.

SQL QUERY:

CREATE DATABASE IF NOT EXISTS deductive_database;

```
mysql> CREATE DATABASE IF NOT EXISTS deductive_database;
Query OK, 1 row affected (0.01 sec)
```

SQL QUERY:

CREATE TABLE IF NOT EXISTS parents (
parent VARCHAR(255),
child VARCHAR(255)
);

```
mysql> CREATE TABLE IF NOT EXISTS parents (
    -> parent VARCHAR(255),
    -> child VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

DESCRIBE parents;

```
mysql> DESCRIBE parents;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| parent | varchar(255) | YES  |     | NULL    |       |
| child  | varchar(255) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

SQL QUERY:

CREATE TABLE IF NOT EXISTS ancestors (
ancestor VARCHAR(255),
descendant VARCHAR(255)

);

```
mysql> CREATE TABLE IF NOT EXISTS ancestors (
    -> ancestor VARCHAR(255),
    -> descendant VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

```
INSERT INTO parents (parent, child)
VALUES ('John', 'Alice'), ('Alice', 'Bob'), ('Bob', 'Charlie');
```

```
mysql> INSERT INTO parents (parent, child)
    -> VALUES ('John', 'Alice'), ('Alice', 'Bob'), ('Bob', 'Charlie');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

SQL QUERY:

```
DELIMITER $$
CREATE PROCEDURE populate_ancestors()
BEGIN
    DECLARE done BOOLEAN DEFAULT FALSE;
    DECLARE parent_name, child_name VARCHAR(255);
    DECLARE cur CURSOR FOR SELECT * FROM parents;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    TRUNCATE TABLE ancestors;
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO parent_name, child_name;
        IF done THEN
            LEAVE read_loop;
        END IF;
            INSERT INTO ancestors (ancestor, descendant)
            VALUES (parent_name, child_name);
```

```
            INSERT INTO ancestors (ancestor, descendant)
            SELECT a.ancestor, p.child
            FROM ancestors a
            JOIN parents p ON a.descendant = p.parent;
        END LOOP;
        CLOSE cur;
    END$$
    DELIMITER ;
```

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE populate_ancestors()
    -> BEGIN
    -> DECLARE done BOOLEAN DEFAULT FALSE;
    -> DECLARE parent_name, child_name VARCHAR(255);
    -> DECLARE cur CURSOR FOR SELECT * FROM parents;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    -> TRUNCATE TABLE ancestors;
    -> OPEN cur;
    -> read_loop: LOOP
    -> FETCH cur INTO parent_name, child_name;
    -> IF done THEN
    -> LEAVE read_loop;
    -> END IF;
    -> INSERT INTO ancestors (ancestor, descendant)
    -> VALUES (parent_name, child_name);
    ->
    -> INSERT INTO ancestors (ancestor, descendant)
    -> SELECT a.ancestor, p.child
    -> FROM ancestors a
    -> JOIN parents p ON a.descendant = p.parent;
    -> END LOOP;
    -> CLOSE cur;
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
```

SQL QUERY:

CALL populate_ancestors();

```
mysql> CALL populate_ancestors();
Query OK, 0 rows affected (0.14 sec)
```

SQL QUERY:

SELECT * from ancestors;

```
mysql> SELECT * from ancestors;
+----------+------------+
| ancestor | descendant |
+----------+------------+
| John     | Alice      |
| John     | Bob        |
| Alice    | Bob        |
| John     | Bob        |
| Alice    | Charlie    |
| John     | Charlie    |
| Bob      | Charlie    |
| John     | Bob        |
| John     | Charlie    |
| Alice    | Charlie    |
| John     | Charlie    |
+----------+------------+
11 rows in set (0.01 sec)
```

Lab 6 : Converting EER model to relational Model

Theory:

An Enhanced Entity-Relationship (EER) model is an extension of the traditional Entity-Relationship (ER) model that includes additional features to represent complex relationships between entities more accurately. It is a diagrammatic technique for displaying the Sub Class and Super Class; Specialization and Generalization; Union or Category; Aggregation etc.

Relational Diagram

Step 1: Mapping of Strong entity

Employee

Ssn	Bdate	Fname	Minit	Lname	Address	Salary	Sex

Department

Name	Number	Locations	Number_of_employees

Project

Name	Number	Locations

Step 2: Mapping of Weak entity

Employee

Ssn	Bdate	Fname	Minit	Lname	Address	Salary	Sex

Dependent

Name	Sex	Birth_Date	Relationship	Ssn

Step 3: Mapping of multivalued attributes

Department

Name	Number	Number_of_Employees

Location

Name	Number	Locations

Step 4: Mapping of composite attributes

Employee

Ssn	Bdate	Fname	Minit	Lname	Address	Salary	Sex

Step 5: Mapping of 1:1 Binary Relationship

Employee

Ssn	Bdate	Fname	Minit	Lname	Address	Salary	Sex

Department

Name	Number	Locations	Ssn

Step 6: Mapping of 1:N Binary relationship .

Employee dependents_of dependent Employee

Dependent

Name	Sex	Birth_Date	Relationship	Ssn

Department works_for employee

Department

Employee

Ssn	Bdate	Fname	Minit	Lname	Address	Salary	Sex	Name	Number

Supervisor Supervision Supervisee Supervisor

Supervisee

Ssn	Bdate	Fname	Minit	Lname	Address	Salary	Sex

Department controls project

Department

Project

Name	Number	Locations	Name	Number

Step 7: Mapping of M:N Binary relationship

Employee works_on project

Employee

Project

Name	Number	Locations

Works_on

Ssn	Name	Number	Hours

Step 8: Mapping of n-ary relationships

Step 9: Mapping of Specializations and generalizations

Step 10: Mapping of aggregations

Lab 7 : Create a Multimedia database.

SQL QUERY:

```
CREATE DATABASE IF NOT EXISTS multimedia_database;
USE multimedia_database;
```

```
mysql> CREATE DATABASE IF NOT EXISTS multimedia_database;
Query OK, 1 row affected (0.01 sec)

mysql> USE multimedia_database;
Database changed
mysql>
```

SQL QUERY:

```
CREATE TABLE IF NOT EXISTS images(
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    image_data LONGBLOB
    );
```

```
mysql> CREATE TABLE IF NOT EXISTS images(
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(255),
    -> image_data LONGBLOB
    -> );
Query OK, 0 rows affected (0.03 sec)
```

SQL QUERY:

```
CREATE TABLE IF NOT EXISTS audio(
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    audio_data LONGBLOB
    );
```

```
mysql> CREATE TABLE IF NOT EXISTS audio(
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(255),
    -> audio_data LONGBLOB
    -> );
Query OK, 0 rows affected (0.03 sec)
```

SQL QUERY:

```
CREATE TABLE IF NOT EXISTS video(
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    video_data LONGBLOB
    );
```

```
mysql> CREATE TABLE IF NOT EXISTS video(
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(255),
    -> video_data LONGBLOB
    -> );
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

```
SHOW tables;
```

```
mysql> SHOW tables;
+--------------------------------+
| Tables_in_multimedia_database |
+--------------------------------+
| audio                          |
| images                         |
| video                          |
+--------------------------------+
3 rows in set (0.00 sec)
```

Inserting Image files

SQL QUERY:

INSERT INTO images (name, image_data) VALUES ('Image 1',
LOAD_FILE("/home/basanta/Pictures/wallpapers/void_wallpaper.jpg"));

```
mysql> INSERT INTO images (name, image_data)
    -> VALUES ('Image 1', LOAD_FILE('/home/basanta/Pictures/wallpapers/void_wallpaper.jpg'));
Query OK, 1 row affected (0.01 sec)
```

Inserting Video files

SQL QUERY:

INSERT INTO audio (name, audio_data)
VALUES ('Audio 1', LOAD_FILE("/home/basanta/Code/pp/static/alarm.mp3"));

```
mysql> INSERT INTO audio (name, audio_data)  VALUES ('Audio 1', LOAD_FILE("/home/basanta/Code/pp/static/alarm.mp3"));
Query OK, 1 row affected (0.01 sec)
```

Inserting Video files

SQL QUERY:

INSERT INTO video (name, video_data)
VALUES ('video 1', LOAD_FILE("/home/basanta/Downloads/let her go - passenger.mp4"));

```
mysql> INSERT INTO video (name, video_data)  VALUES ('video 1', LOAD_FILE("/home/basanta/Downloads/let her go - passenger.mp4"));
Query OK, 1 row affected (0.00 sec)
```

Database

Images:

	id	name	image_data
Edit Copy Delete	1	Image 1	[BLOB - 318.8 KiB]
Edit Copy Delete	2	Image 1	[BLOB - 318.8 KiB]

Audio:

	id	name	audio_data
Edit Copy Delete	1	Audio 1	[BLOB - 51.2 KiB]

Video:

	id	name	video_data
Edit Copy Delete	1	Video 1	[BLOB - 51.2 KiB]

Lab 8 : Demonstrate Spatial DB (GIS)

SQL QUERY:

SHOW VARIABLES LIKE 'have_%';

```
mysql> SHOW VARIABLES LIKE 'have_%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| have_compress           | YES      |
| have_dynamic_loading    | YES      |
| have_geometry           | YES      |
| have_openssl            | YES      |
| have_profiling          | YES      |
| have_query_cache        | NO       |
| have_rtree_keys         | YES      |
| have_ssl                | YES      |
| have_statement_timeout  | YES      |
| have_symlink            | DISABLED |
+-------------------------+----------+
10 rows in set (0.01 sec)
```

SQL QUERY:

CREATE DATABASE IF NOT EXISTS spatial_database;

USE spatial_database;

```
mysql> CREATE DATABASE IF NOT EXISTS spatial_database;
Query OK, 1 row affected (0.01 sec)

mysql> USE spatial_database;
Database changed
mysql>
```

SQL QUERY:

CREATE TABLE locations (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(255),
coordinates POINT
);

```
mysql> CREATE TABLE locations (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> name VARCHAR(255),
    -> coordinates POINT
    -> );
Query OK, 0 rows affected (0.02 sec)
```

SQL QUERY:

INSERT INTO locations (name, coordinates) VALUES
('Location A', POINT(1.2345, 2.3456)),
('Location B', POINT(3.4567, 4.5678)),
('Location C', POINT(5.6789, 6.7890));

```
mysql> INSERT INTO locations (name, coordinates) VALUES
    -> ('Location A', POINT(1.2345, 2.3456)),
    -> ('Location B', POINT(3.4567, 4.5678)),
    -> ('Location C', POINT(5.6789, 6.7890));
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

SQL QUERY:

SELECT name FROM locations WHERE ST_DISTANCE(coordinates, POINT(1.0, 2.0)) < 1.0;

```
mysql> SELECT name FROM locations WHERE ST_DISTANCE(coordinates, POINT(1.0, 2.0)) < 1.0;
+------------+
| name       |
+------------+
| Location A |
+------------+
1 row in set (0.01 sec)
```

Lab 9 : Demonstrate Temporal DB

SQL QUERY:

CREATE DATABASE IF NOT EXISTS basanta_temporal_database;

USE basanta_temporal_database;

```
MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS basanta_temporal_database;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>       USE basanta_temporal_database;
Database changed
```

SQL QUERY:

CREATE TABLE basanta_temporal_table (

id INT AUTO_INCREMENT PRIMARY KEY,

data VARCHAR(255),

valid_from TIMESTAMP(6) GENERATED ALWAYS AS ROW START,

valid_to TIMESTAMP(6) GENERATED ALWAYS AS ROW END,

PERIOD FOR SYSTEM_TIME(valid_from, valid_to)

) WITH SYSTEM VERSIONING;

```
MariaDB [basanta_temporal_database]> CREATE TABLE basanta_temporal_table (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> data VARCHAR(255),
    -> valid_from TIMESTAMP(6) GENERATED ALWAYS AS ROW START,
    -> valid_to TIMESTAMP(6) GENERATED ALWAYS AS ROW END,
    -> PERIOD FOR SYSTEM_TIME(valid_from, valid_to)
    -> ) WITH SYSTEM VERSIONING;
Query OK, 0 rows affected (0.009 sec)
```

Inserting data

SQL QUERY:

INSERT INTO basanta_temporal_table (data)

VALUES

('Test data 01'),

('Test data 02');

```
MariaDB [basanta_temporal_database]> INSERT INTO basanta_temporal_table (data)
    -> VALUES
    -> ('Test data 01'),
    -> ('Test data 02');
Query OK, 2 rows affected (0.003 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Updating data

SQL QUERY:

UPDATE basanta_temporal_table SET data = “Test data 01 updated” WHERE id = 1;

```
MariaDB [basanta_temporal_database]> UPDATE basanta_temporal_table
    -> SET
    -> data = “Test data 01 updated” WHERE id = 1;
Query OK, 1 row affected (0.005 sec)
Rows matched: 1  Changed: 1  Inserted: 1  Warnings: 0
```

Delete data

SQL QUERY:

DELETE FROM basanta_temporal_table WHERE id=2;

```
MariaDB [basanta_temporal_database]> DELETE FROM basanta_temporal_table WHERE id=2;
Query OK, 1 row affected (0.003 sec)
```

View Current Data

SQL QUERY:

SELECT * from basanta_temporal_table;

```
MariaDB [basanta_temporal_database]> SELECT * from basanta_temporal_table;
+----+---------------------+----------------------------+----------------------------+
| id | data                | valid_from                 | valid_to                   |
+----+---------------------+----------------------------+----------------------------+
|  1 | Test data 01 updated | 2024-06-09 13:02:14.544555 | 2038-01-19 08:59:07.999999 |
+----+---------------------+----------------------------+----------------------------+
1 row in set (0.000 sec)
```

View Historical Data

SQL QUERY:

SELECT * from basanta_temporal_table FOR SYSTEM_TIME ALL;

```
MariaDB [basanta_temporal_database]> SELECT * from basanta_temporal_table FOR SYSTEM_TIME ALL;
+----+---------------------+----------------------------+----------------------------+
| id | data                | valid_from                 | valid_to                   |
+----+---------------------+----------------------------+----------------------------+
|  1 | Test data 01        | 2024-06-09 13:02:00.794172 | 2024-06-09 13:02:14.544555 |
|  1 | Test data 01 updated | 2024-06-09 13:02:14.544555 | 2038-01-19 08:59:07.999999 |
|  2 | Test data 02        | 2024-06-09 13:02:00.794172 | 2024-06-09 13:02:25.573430 |
+----+---------------------+----------------------------+----------------------------+
3 rows in set (0.000 sec)
```

Drop Database

SQL QUERY:

DROP DATABASE IF EXISTS temporal_database;

```
MariaDB [basanta_temporal_database]> DROP DATABASE IF EXISTS temporal_database;
```

Lab 10 : Create Table using another table

SQL QUERY:

CREATE TABLE employee_duplicate AS SELECT emp_id, emp_name FROM employee;

```
mysql> CREATE TABLE employee_duplicate AS SELECT emp_id, name FROM employee;
Query OK, 0 rows affected (0.26 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

SQL QUERY:

SELECT * FROM employee_duplicate;

```
mysql> DESCRIBE employee_duplicate;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| emp_id | int         | NO   |     | NULL    |       |
| name   | varchar(30) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Lab 11 : SQL Truncate Table

SQL QUERY:

TRUNCATE TABLE employee;

```
mysql> TRUNCATE TABLE employee;
Query OK, 0 rows affected (0.15 sec)
```

SQL QUERY:

SELECT * FROM test_table;

```
mysql> SELECT * FROM employee;
Empty set (0.00 sec)
```

Lab 12 : Perform CRUD operations in NOSQL

 # Show all databases

QUERY:

show dbs;

```
test> show dbs;
admin    40.00 KiB
config  108.00 KiB
local    72.00 KiB
test>
```

 # Use database

QUERY:

use basanta;

```
test> use basanta;
switched to db basanta
basanta>
```

 # Create collection

QUERY:

db.createCollection("basanta_users");

```
basanta> db.createCollection("basanta_users");
{ ok: 1 }
basanta>
```

 # Insert single data into collection

QUERY:

db.basanta_users.insertOne({"name": "basanta", "age": 23});

```
basanta> db.basanta_users.insertOne({"name": "basanta", "age": 23});
{
  acknowledged: true,
  insertedId: ObjectId('666548c93f61b05f06cdcdf6')
}
basanta>
```

 # Insert multiple data into collection

QUERY:

```
db.basanta_users.insertMany([
  {"name": "ashmita", "age": 20},
  {"name": "raj", "age": 25}
]);
```

```
basanta> db.basanta_users.insertMany([
...     {"name": "ashmita", "age": 20},
...     {"name": "raj", "age": 25}
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('666548c93f61b05f06cdcdf7'),
    '1': ObjectId('666548c93f61b05f06cdcdf8')
  }
}
basanta>
```

 # Find all the objects/data in the given collection

QUERY:

```
db.basanta_users.find();
```

```
basanta> db.basanta_users.find();
[
  {
    _id: ObjectId('666548c93f61b05f06cdcdf6'),
    name: 'basanta',
    age: 23
  },
  {
    _id: ObjectId('666548c93f61b05f06cdcdf7'),
    name: 'ashmita',
    age: 20
  },
  { _id: ObjectId('666548c93f61b05f06cdcdf8'), name: 'raj', age: 25 }
]
basanta>
```

 # Find single data

QUERY:

```
db.basanta_users.find({"name": "ashmita"});
```

```
basanta> db.basanta_users.find({"name": "ashmita"});
[
  {
    _id: ObjectId('666548c93f61b05f06cdcdf7'),
    name: 'ashmita',
    age: 20
  }
]
basanta>
```

 # Update single data in collection

QUERY:

db.basanta_users.updateOne({"name": "basanta"}, {$set: {"address": "Kathmandu"}});

```
basanta> db.basanta_users.updateOne({"name": "basanta"}, {$set: {"address": "Kathmandu"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
basanta>
```

 # Update many data in collection

QUERY:

db.basanta_users.updateMany({"name": "basanta"}, {$set: {"age": 24, "address": "Pokhara"}});

```
basanta> db.basanta_users.updateMany({"name": "basanta"}, {$set: {"age": 24, "address": "Pokhara"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
basanta>
```

 # Delete a data in collection

QUERY:

db.basanta_users.deleteOne({"name": "basanta"});

```
basanta> db.basanta_users.deleteOne({"name": "basanta"});
{ acknowledged: true, deletedCount: 1 }
basanta>
```

 # Delete many data in collection

QUERY:

db.basanta_users.deleteMany({"age": {$lt: 30}});

```
basanta> db.basanta_users.deleteMany({"age": {$lt: 30}});
{ acknowledged: true, deletedCount: 2 }
basanta>
```

 # Drop collection

QUERY:

db.basanta_users.drop();

```
basanta> db.basanta_users.drop();
true
basanta>
```

 # Drop database

QUERY:

db.dropDatabase();

```
basanta> db.dropDatabase();
{ ok: 1, dropped: 'basanta' }
basanta>
```