**Madiha Aimon Tappal**

[**madihaaimon@gamil.com**](mailto:madihaaimon@gamil.com)

**Data Engineering Batch – 1**

**Day – 3 Assignment**

Structured Query Language (SQL) was developed to work with relational databases that organize and store information in groups of columns and rows, called tables. They are “relational” because of relations linking data between different tables (think: Excel).

SQL has lasted for decades, and it’s become the industry standard because of its ease, versatility, and power in working with transactional data. SQL is easy to learn for beginners, and can take developers far in their careers. It’s still the best language for defining data architecture, and it remains more popular among data engineers and scientists than languages like Python or R.

**What is a Relational Database?**

A relational database is a database that exhibits two key SQL concepts: tables and data relations (hence the name “relational database”). Each table consists of rows and columns, and the configuration of the tables is known as its “schema.”

For example, here’s a graphical diagram of the schema of two tables we’ll use in this guide. Below each table is the list of the columns for that table, along with their corresponding data type. The two tables are related by the id column of the Animals table.

A helpful way to think about tables is to view them as a spreadsheet with columns for the information we want to track and rows for each data entry we want to store. We can create multiple sheets with different types of information inside each sheet, and then use SQL queries to link and work with the data across all of them.

**What Are the Data Languages Within SQL?**

SQL consists of three different types of underlying groups:

* **Data Definition Language (DDL)**
* **Data Manipulation Language (DML)**
* **Data/Transaction Control Language (DCL/TCL)**

DDL allows us to define what the structure of our databases looks like using commands such as CREATE and ALTER. We can imagine it as setting up and labeling shelves for our data and specifying how we want to organize it before moving and working with it.

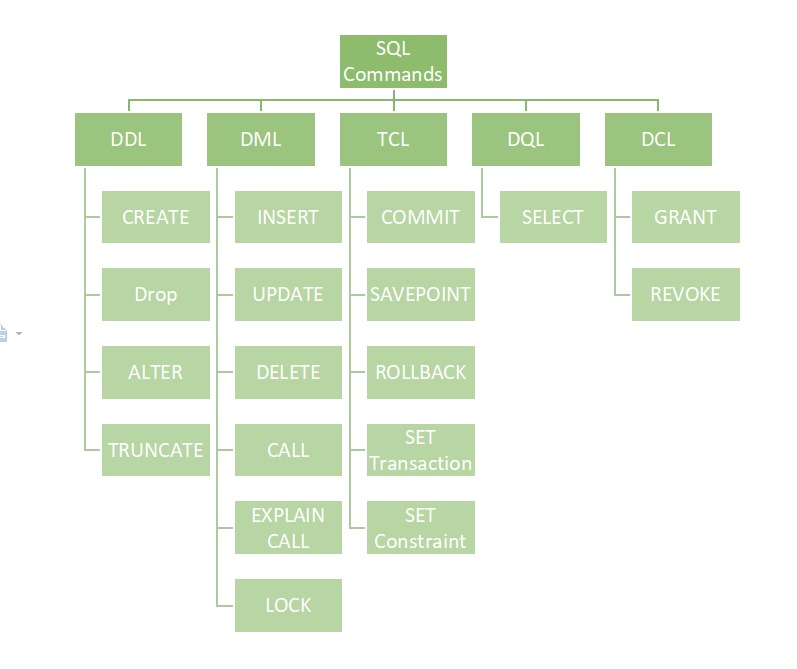
DML provides the methods for how to manipulate the data to actually do the adding, changing, and deleting through commands like SELECT, INSERT, UPDATE, and DELETE.

DCL/TCL enables us to specify who controls our databases with rights and permissions.

Lastly, there are also utility functions that provide us information, such as showing a list of tables or user permissions.

**These SQL commands are mainly categorized into five categories:**

* **DDL – Data Definition Language**
* **DQL – Data Query Language**
* **DML – Data Manipulation Language**
* **DCL – Data Control Language**
* **TCL – Transaction Control Language**

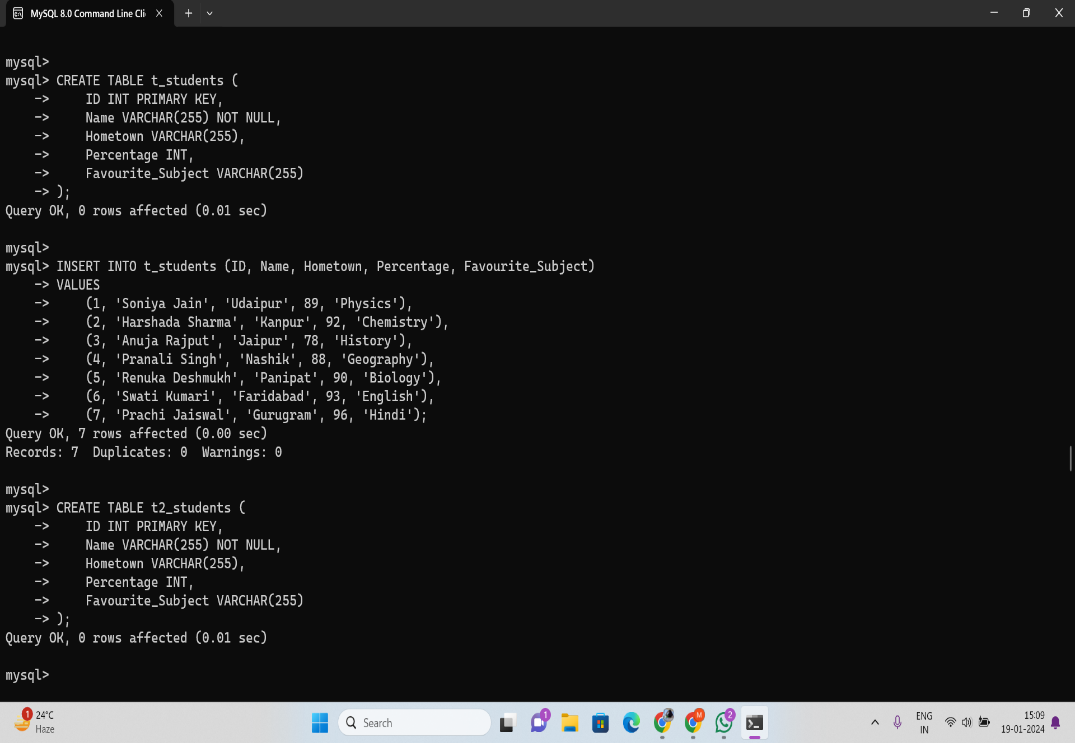


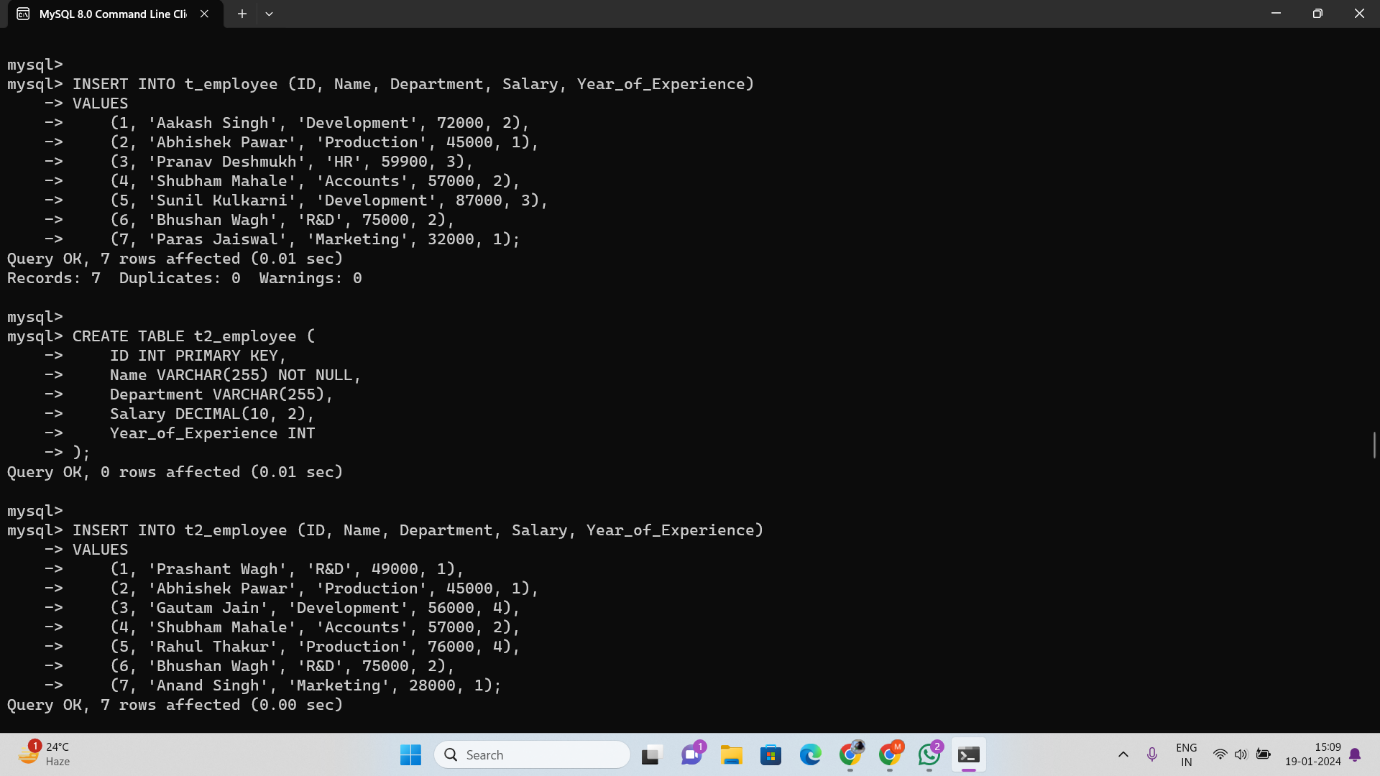
**DDL (Data Definition Language)**

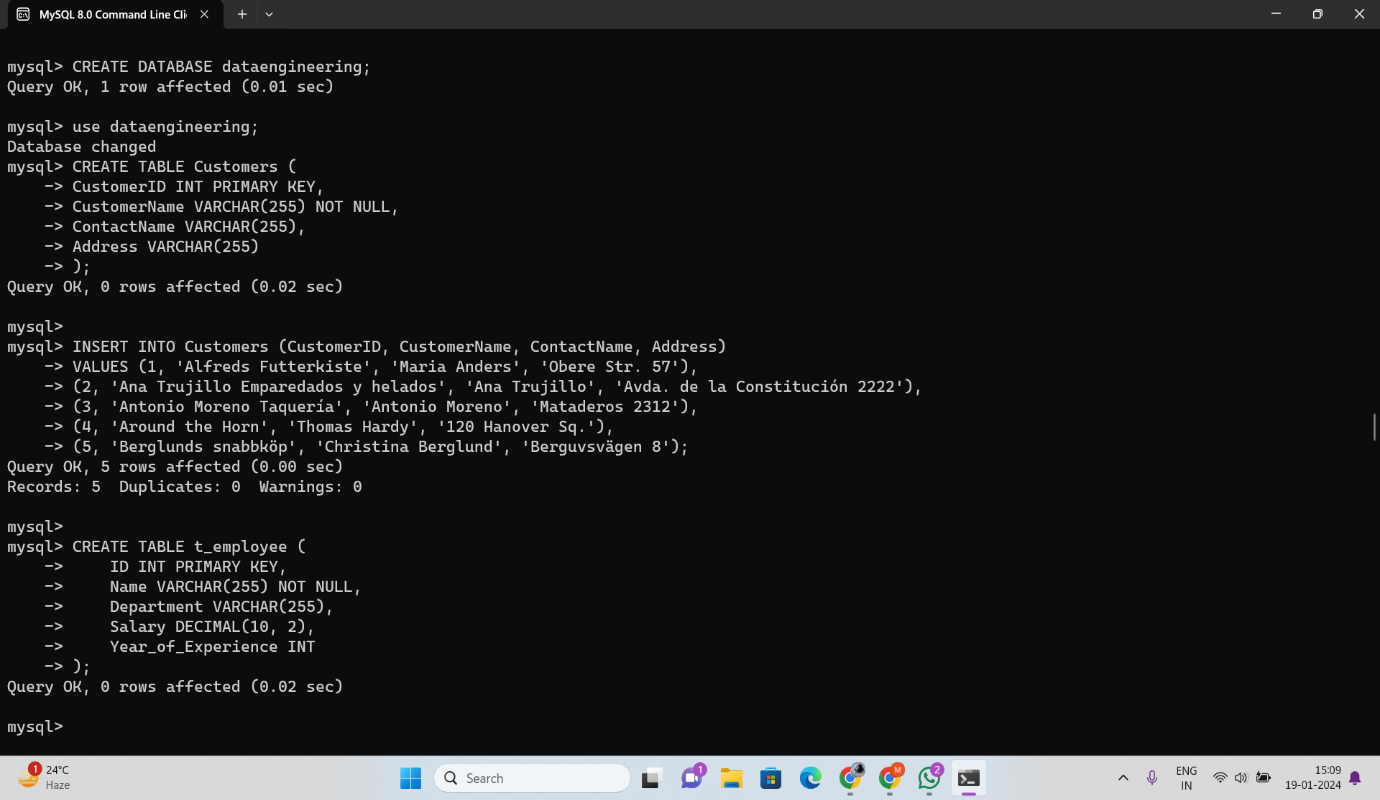
DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

**List of DDL commands:**

* **CREATE:** This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
* **DROP:** This command is used to delete objects from the database.
* **ALTER:** This is used to alter the structure of the database.
* **TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.
* **COMMENT:** This is used to add comments to the data dictionary.
* **RENAME:** This is used to rename an object existing in the database.



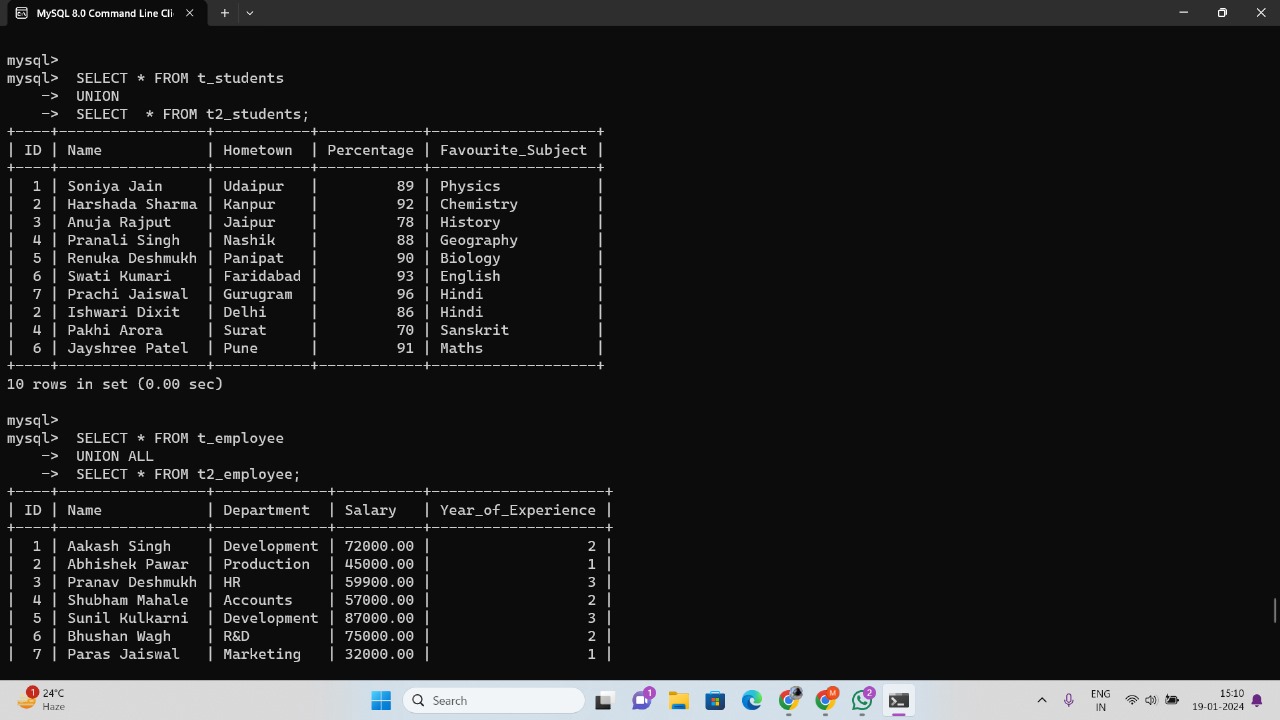
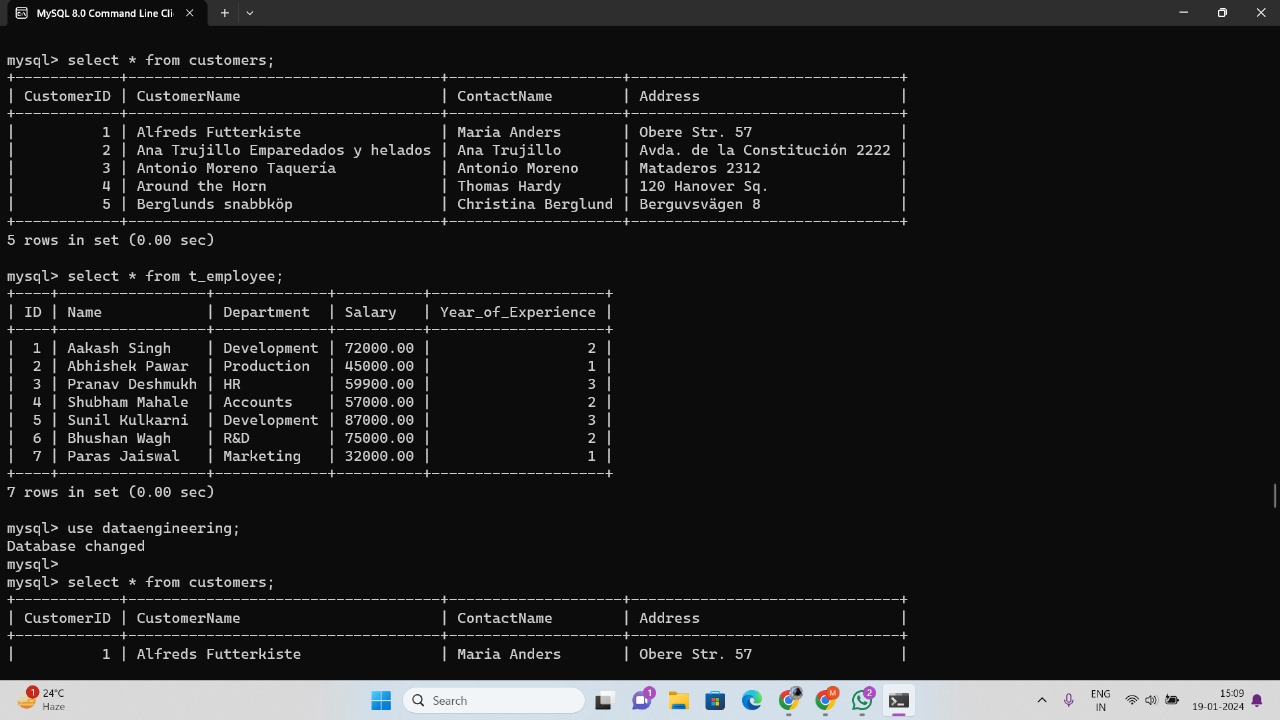
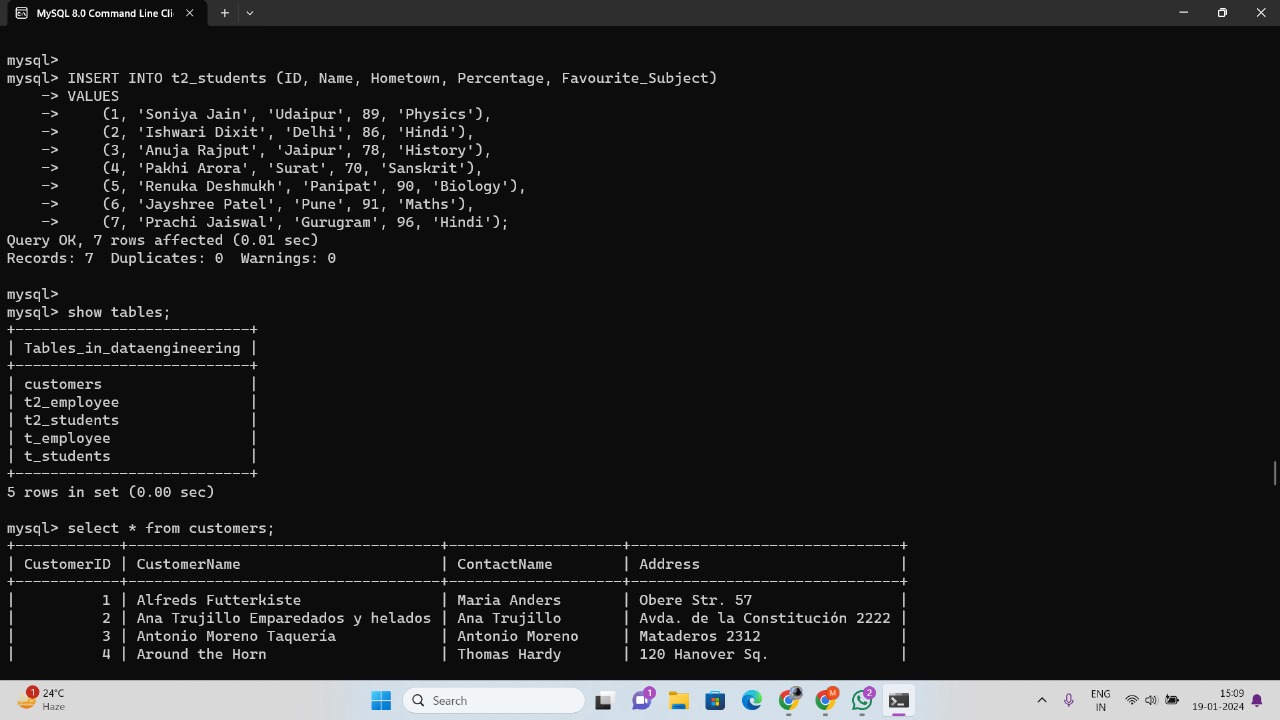




**DQL (Data Query Language)**

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DQL: SELECT: It is used to retrieve data from the databa



**DML(Data Manipulation Language)**

The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

**List of DML commands:**

INSERT: It is used to insert data into a table.

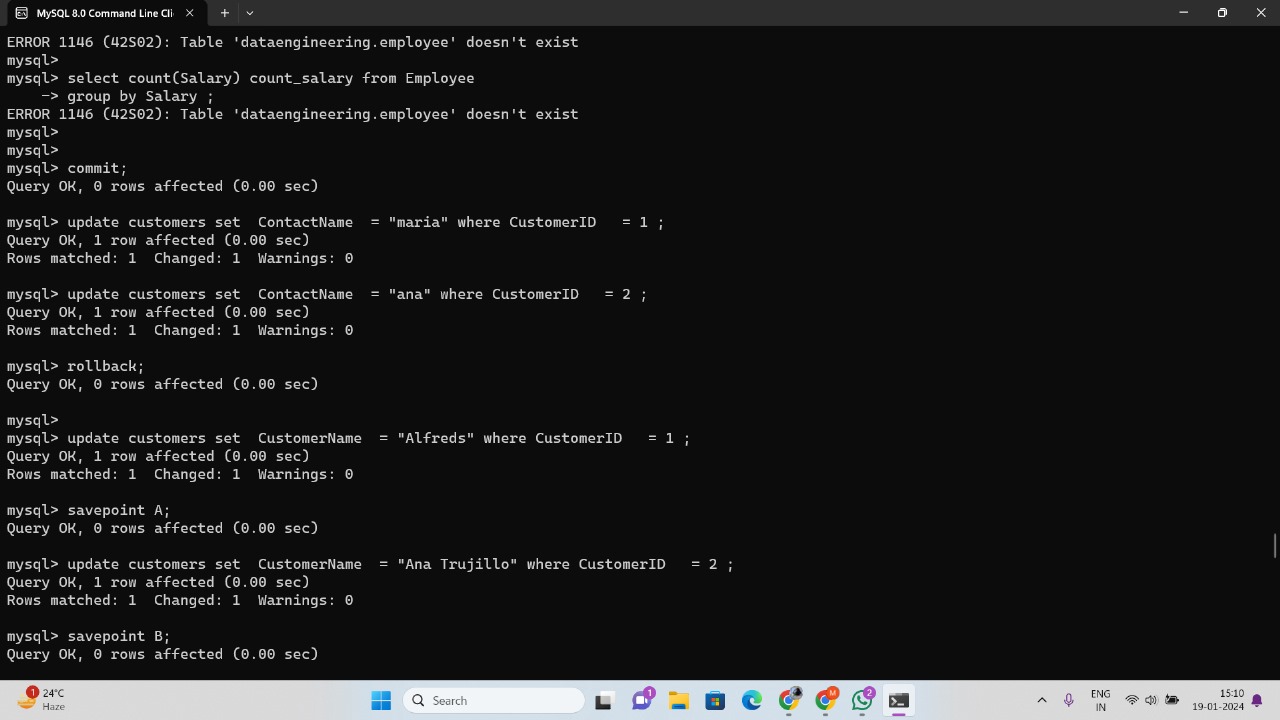
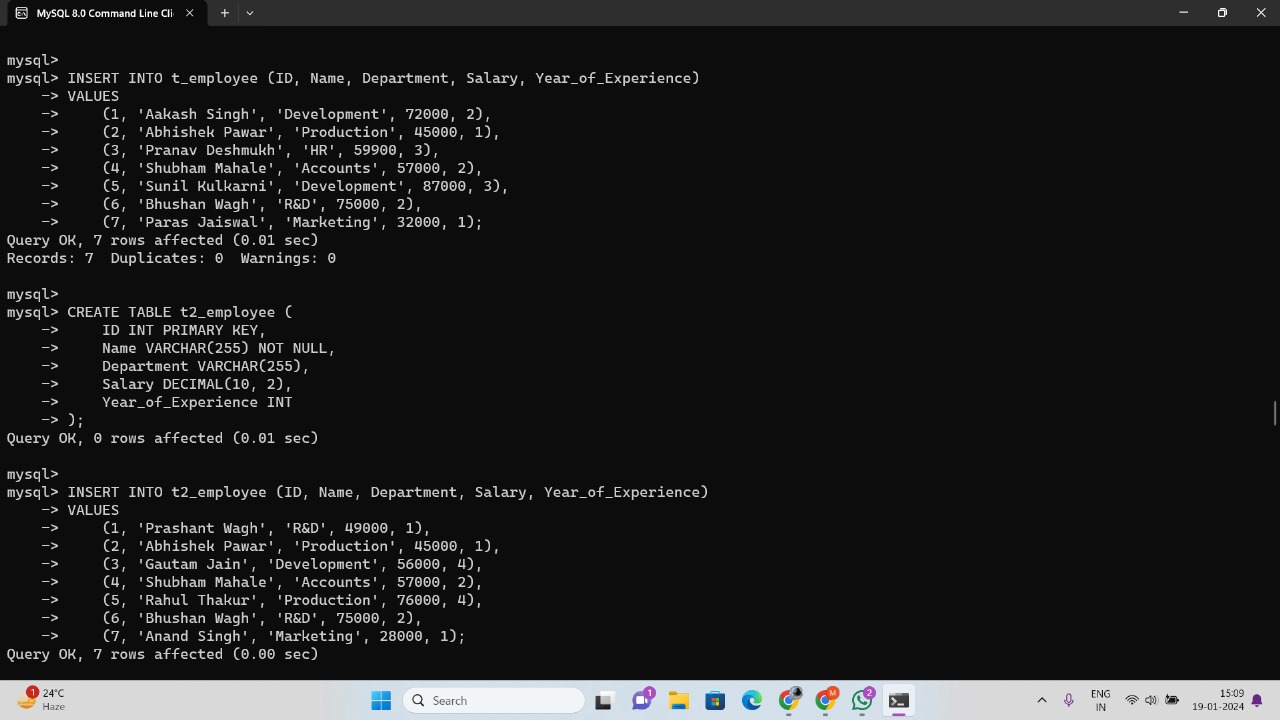
UPDATE: It is used to update existing data within a table.

DELETE: It is used to delete records from a database table.

LOCK: Table control concurrency.

CALL: Call a PL/SQL or JAVA subprogram.

EXPLAIN PLAN: It describes the access path to data.



DCL (Data Control Language)

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

List of DCL commands:

GRANT: This command gives users access privileges to the database.

Syntax:

GRANT SELECT, UPDATE ON MY\_TABLE TO SOME\_USER, ANOTHER\_USER;

REVOKE: This command withdraws the user’s access privileges given by using the GRANT command.

Syntax:

REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;

TCL (Transaction Control Language)

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group are successfully completed. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure. You can explore more about transactions here. Hence, the following TCL commands are used to control the execution of a transaction:

BEGIN: Opens a Transaction.

COMMIT: Commits a Transaction.

Syntax:

COMMIT;

ROLLBACK: Rollbacks a transaction in case of any error occurs.

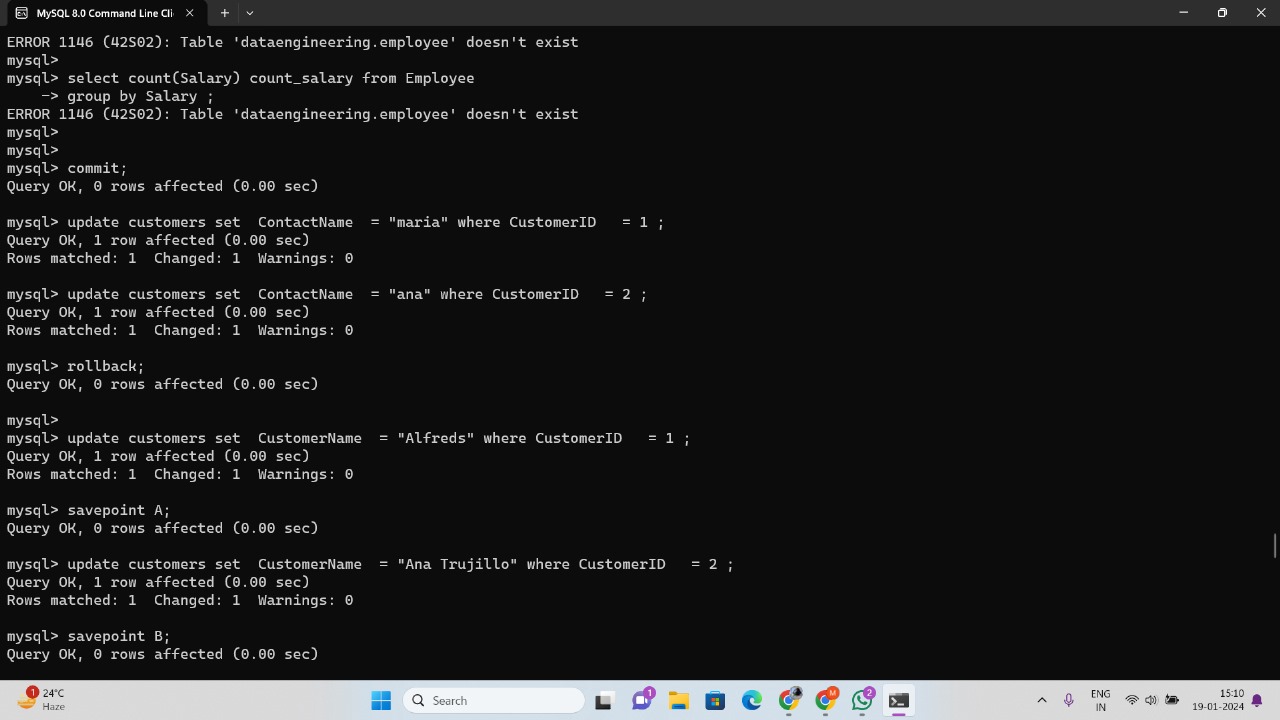
Syntax:

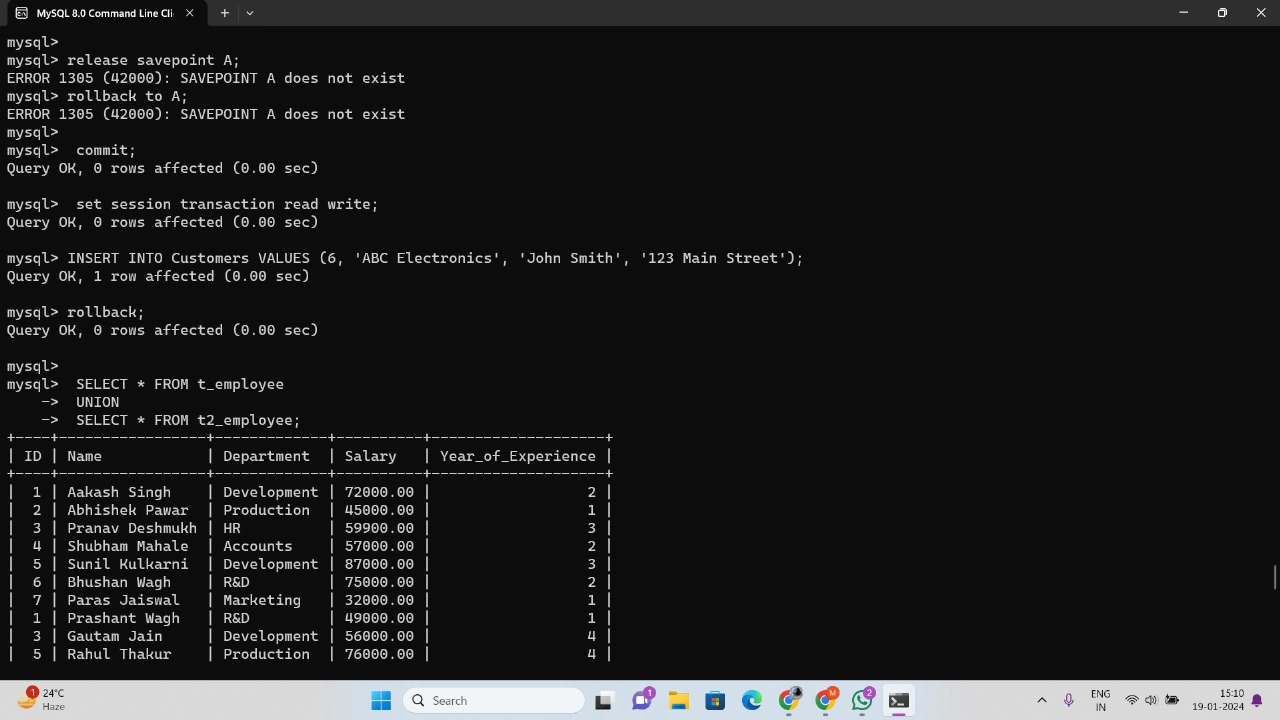
ROLLBACK;

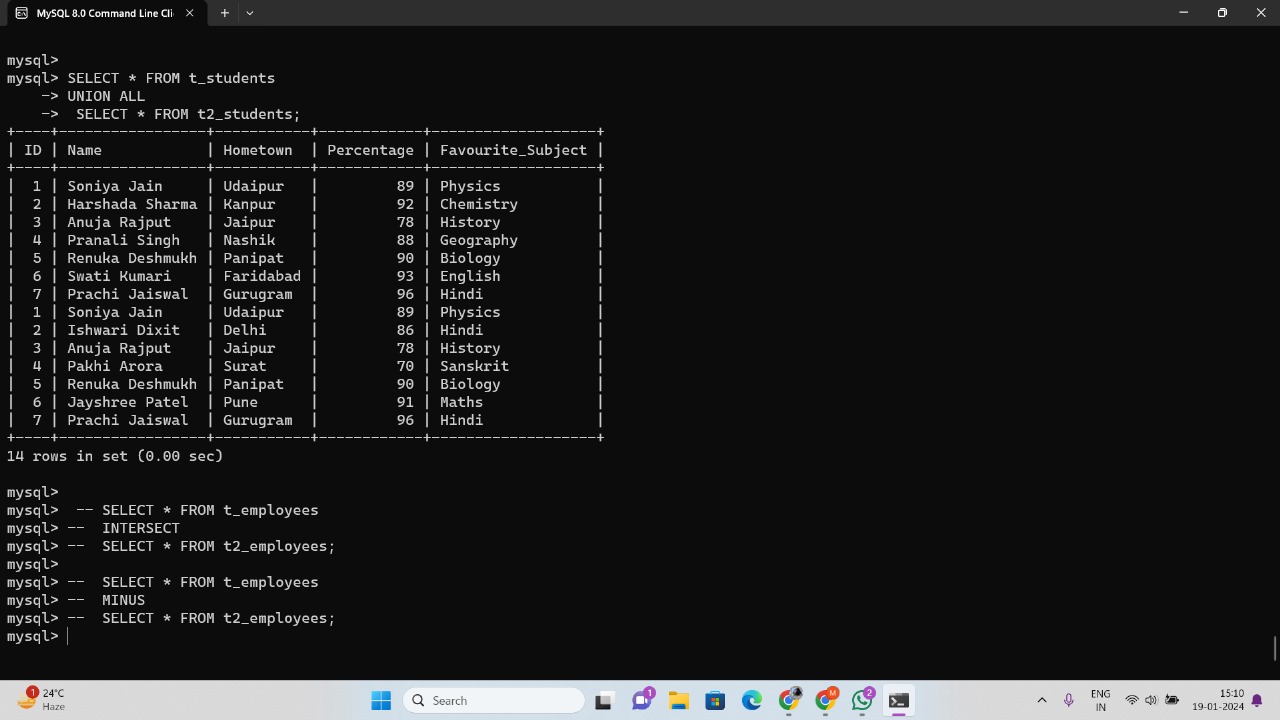
SAVEPOINT: Sets a save point within a transaction.

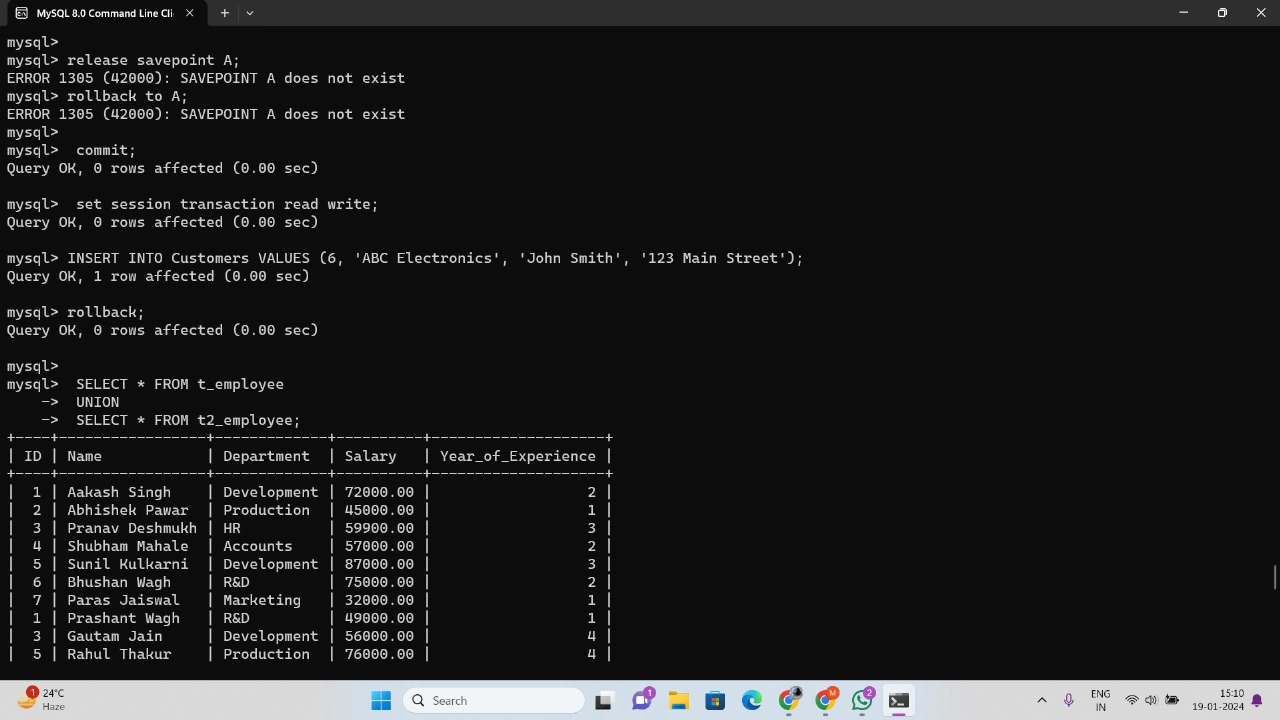
Syntax:

SAVEPOINT SAVEPOINT\_NAME;









**Set Up Dog Shelter Database: CREATE DATABASE**

First, we need to create a database for our data. We’ll name it pet\_adoption using the following SQL statement:

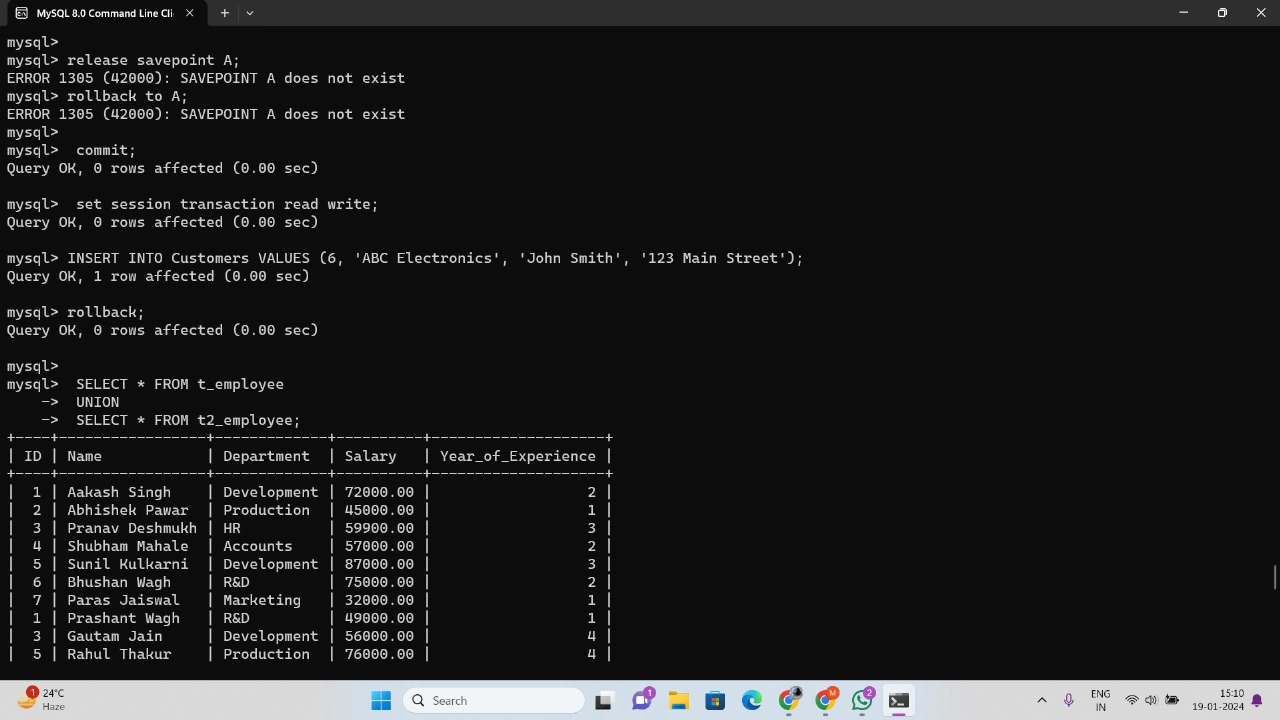
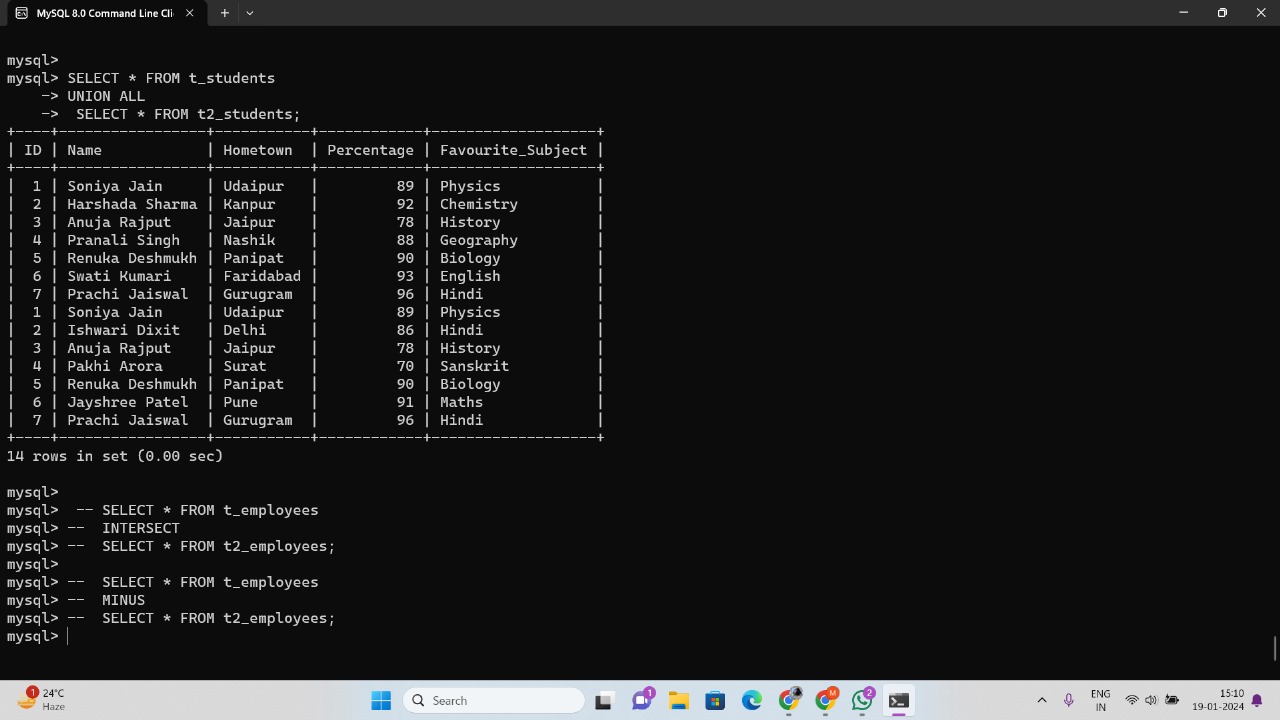
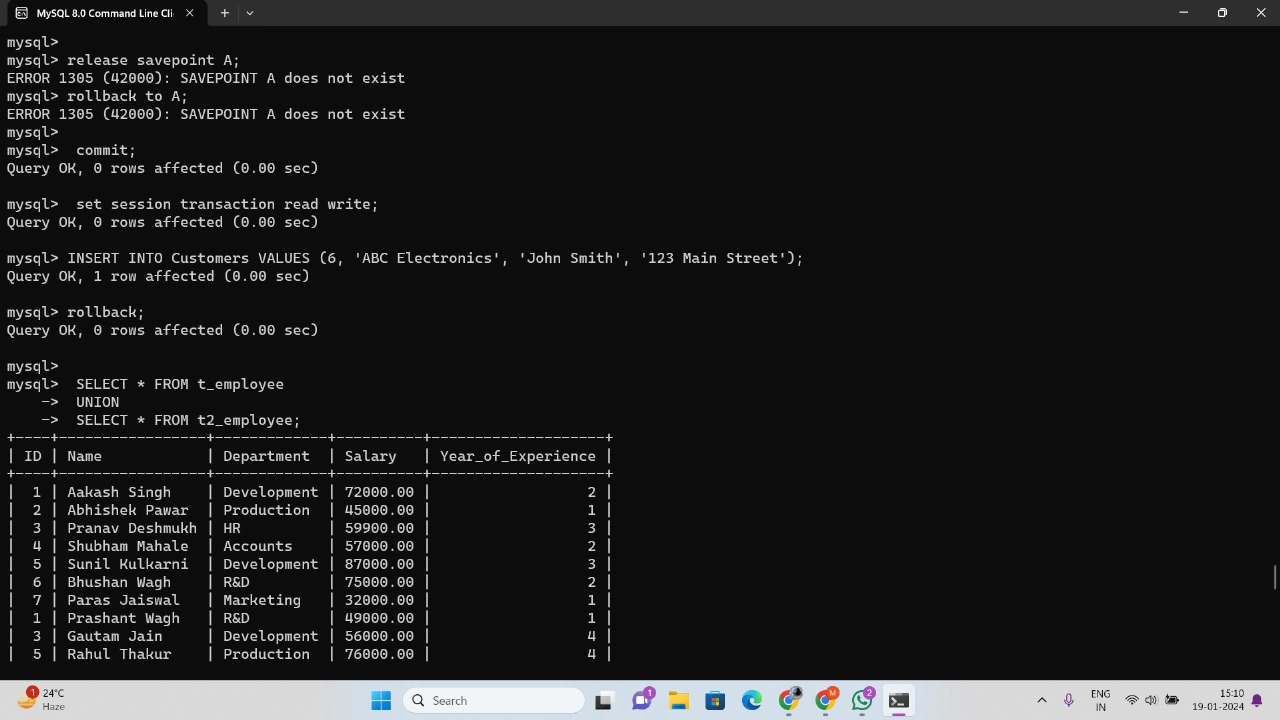
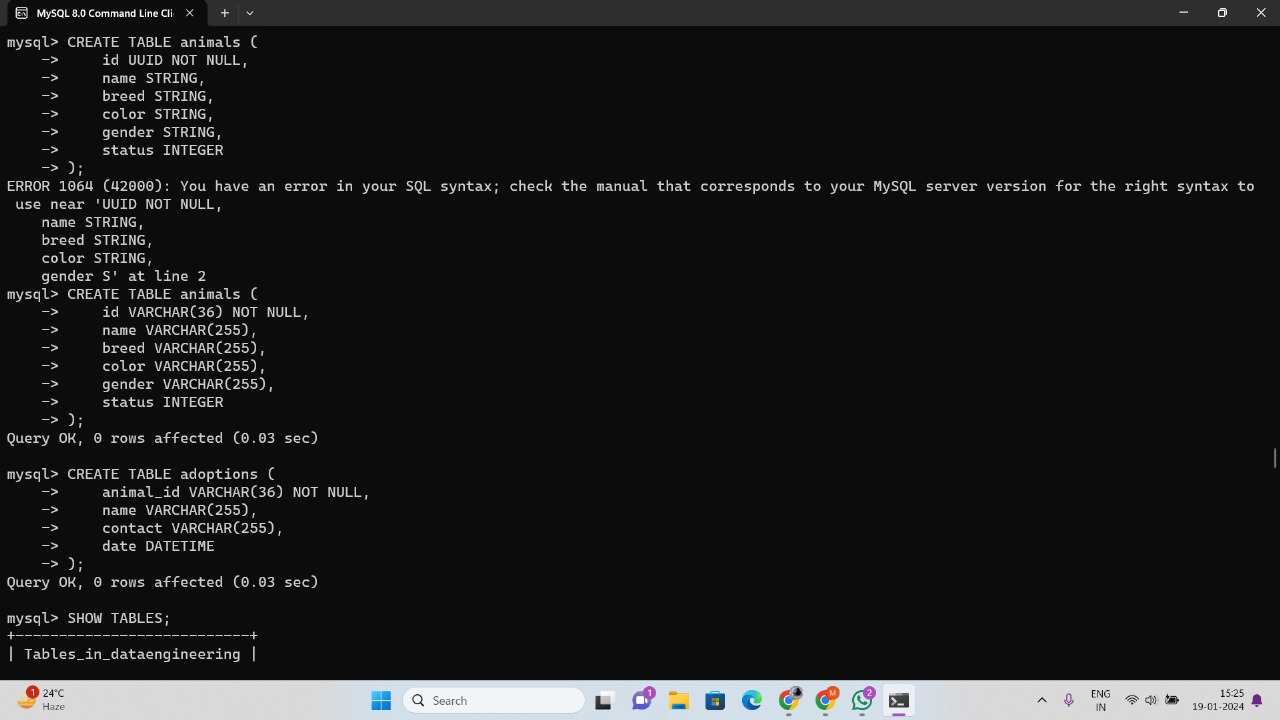
CREATE DATABASE pet\_adoption;

Creating a database doesn’t automatically set it as the active database, so now let’s select the pet\_adoption database with the USE command:

USE pet\_adoption;

Create the animal table with the following command:

CREATE TABLE animals (id UUID NOT NULL, name STRING, breed STRING, color STRING, gender STRING, status INTEGER);



Create this adoptions table using the following command:

CREATE TABLE adoptions (animal\_id UUID NOT NULL, name STRING, contact STRING, date TIMESTAMP);

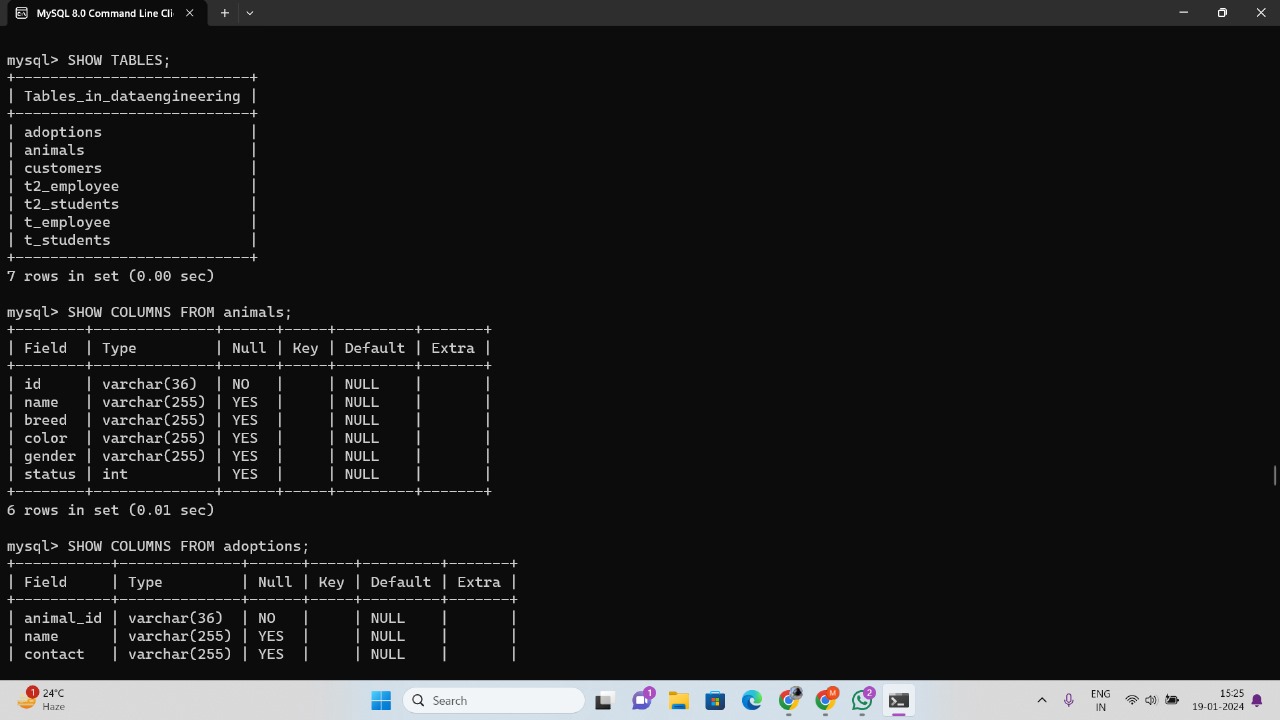
Run this command to get the list of tables in the current database and check that we have both the animals table and the adoptions table:

SHOW TABLES;

If you see both tables, then you can run these two statements to make sure that the columns of each are correct:

SHOW COLUMNS FROM animals;

SHOW COLUMNS FROM adoptions;



Add Dogs to Database: INSERT

INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('89354034-20d9-4c3d-8195-3294bfd9dbc5', 'Bellyflop', 'Beagle', 'Brown', 'Male', 0);

Exercise 1: Add All Dogs to the Database

Here is the list of all of the dogs waiting in line. Try putting them into the system based on the above SQL statement.

Did you get it? The SQL INSERT statements should look like this:

INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('89354034-20d9-4c3d-8195-3294bfd9dbc5', 'Bellyflop', 'Beagle', 'Brown', 'Male', 0);

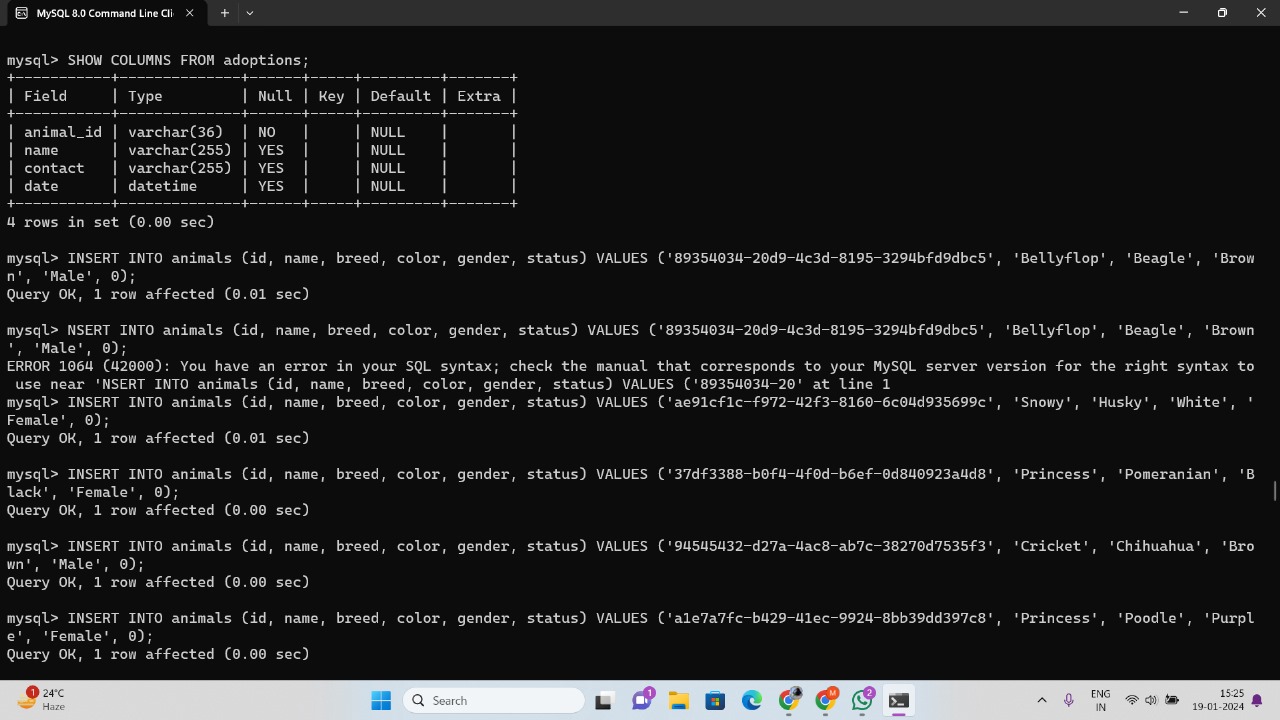
INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('ae91cf1c-f972-42f3-8160-6c04d935699c', 'Snowy', 'Husky', 'White', 'Female', 0);

INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('37df3388-b0f4-4f0d-b6ef-0d840923a4d8', 'Princess', 'Pomeranian', 'Black', 'Female', 0);

INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('94545432-d27a-4ac8-ab7c-38270d7535f3', 'Cricket', 'Chihuahua', 'Brown', 'Male', 0);

INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('a1e7a7fc-b429-41ec-9924-8bb39dd397c8', 'Princess', 'Poodle', 'Purple', 'Female', 0);

INSERT INTO animals (id, name, breed, color, gender, status) VALUES ('5138ed53-2ab2-400b-973c-91186f8c673d', 'Spot', 'Dalmation', 'Black and White', 'Male', 0);



Retrieve List of Dogs: SELECT \* FROM

With the full list of dogs added to our database, we can try running some SELECT queries to look through them. The following are some small examples of possible SQL statements to run.

Get the full list of all properties of all dogs (defaults to a limit of 100 rows):

SELECT \* FROM animals;

Get the breeds of all dogs:

SELECT breed FROM animals;

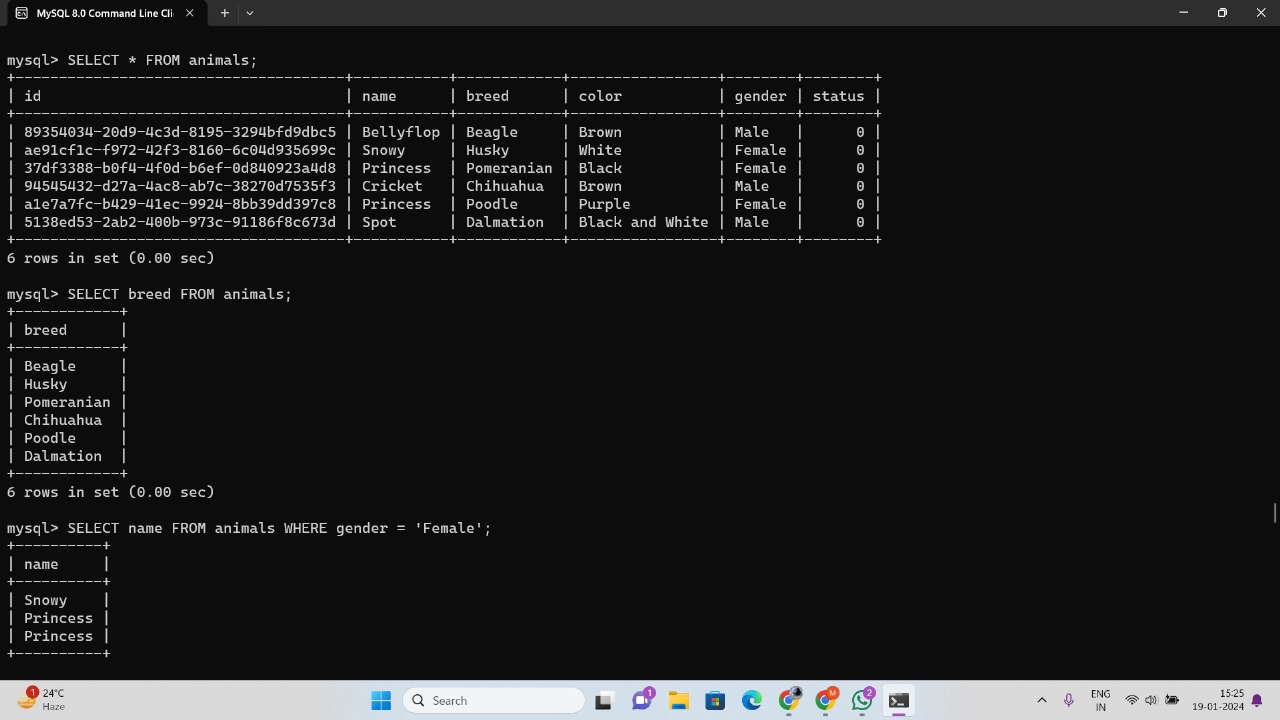
Get the names of only female dogs by including a WHERE clause:

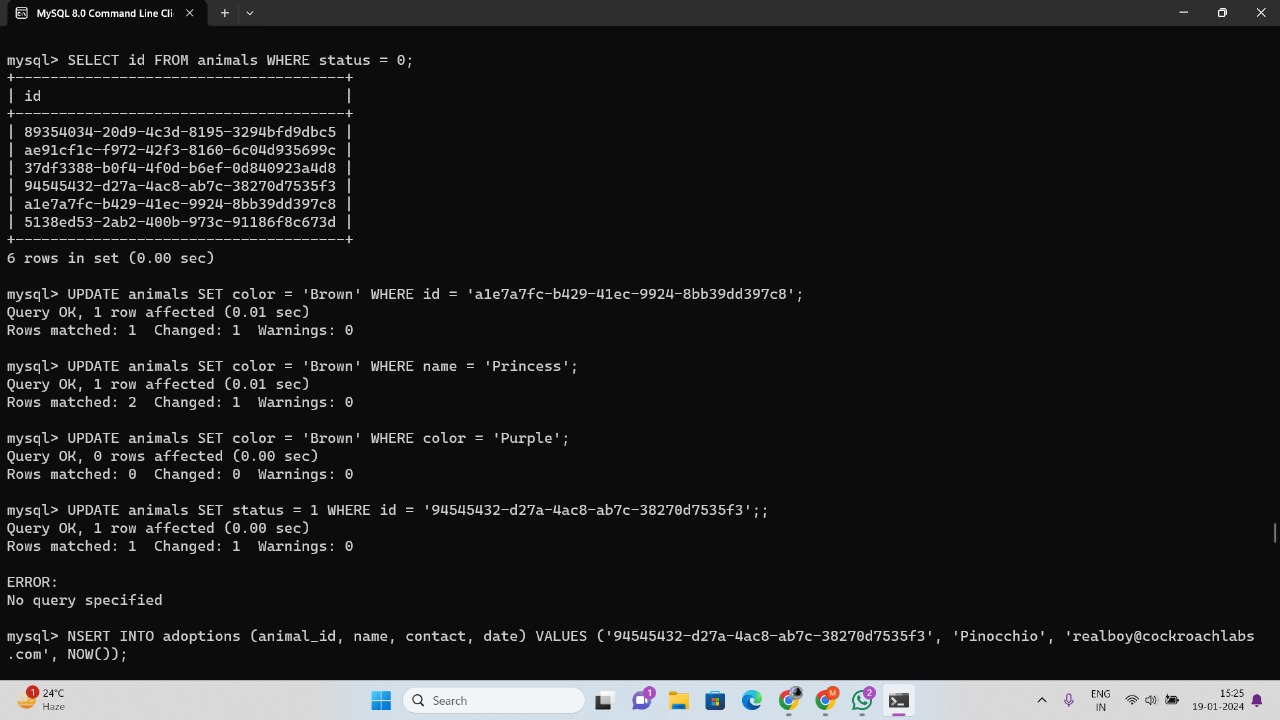
SELECT name FROM animals WHERE gender = 'Female';

Get the IDs of dogs up for adoption:

SELECT id FROM animals WHERE status = 0;

For practice, test and get creative with your SQL statements to get a feel for using SELECT statements with WHERE clauses.





Update Dogs’ Information: UPDATE & DELETE FROM

UPDATE animals SET color = 'Brown' WHERE id = 'a1e7a7fc-b429-41ec-9924-8bb39dd397c8';

UPDATE animals SET color = 'Brown' WHERE name = 'Princess';

The first statement will look like this, using Cricket’s id field:

UPDATE animals SET status = 1 WHERE id = '94545432-d27a-4ac8-ab7c-38270d7535f3';

The adoption SQL statement should look like the following:

INSERT INTO adoptions (animal\_id, name, contact, date) VALUES ('94545432-d27a-4ac8-ab7c-38270d7535f3', 'Pinocchio', 'realboy@cockroachlabs.com', NOW());

UPDATE animals SET status = 1 WHERE id = 'a1e7a7fc-b429-41ec-9924-8bb39dd397c8';

INSERT INTO adoptions (animal\_id, name, contact, date) VALUES ('a1e7a7fc-b429-41ec-9924-8bb39dd397c8', 'Patalie', 'poodlequeen@cockroachlabs.com', NOW());

UPDATE animals SET status = 1 WHERE id = '5138ed53-2ab2-400b-973c-91186f8c673d';

INSERT INTO adoptions (animal\_id, name, contact, date) VALUES ('5138ed53-2ab2-400b-973c-91186f8c673d', 'Ella', 'ellacrew@cockroachlabs.com', NOW());