**Madiha Aimon Tappal**

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

**Data Engineering Batch – 1**

**Day – 3 Assignment JOINS**

**SQL**

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 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

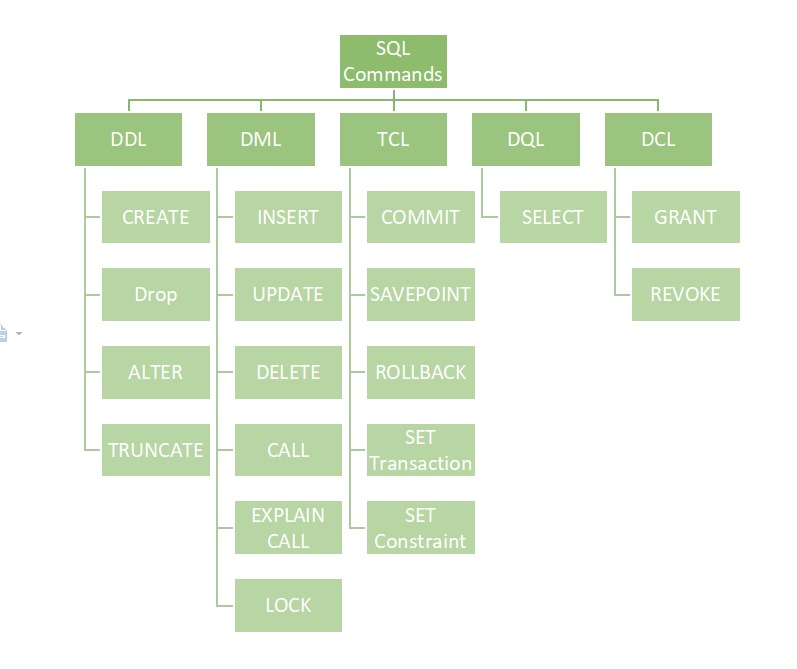
labeling shelves for our data and specifying how we want to organize it before moving and working with it.

**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.



**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 database

**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.

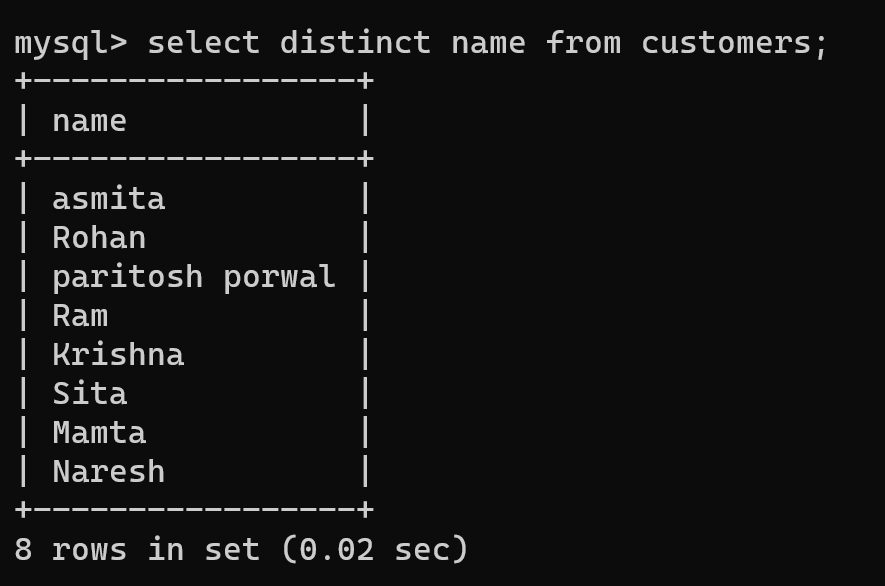
Syntax:

SAVEPOINT SAVEPOINT\_NAME;

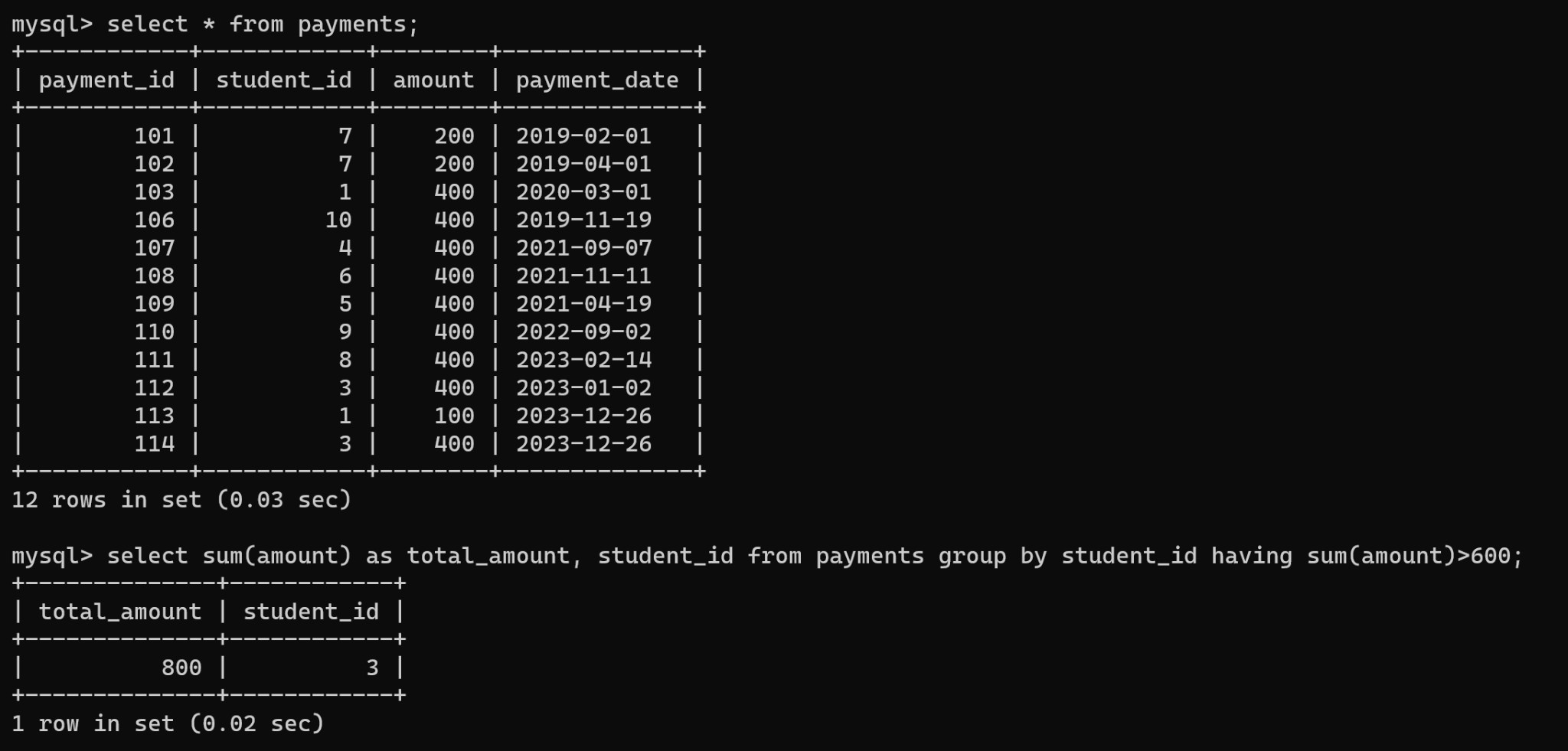
1. Select all columns



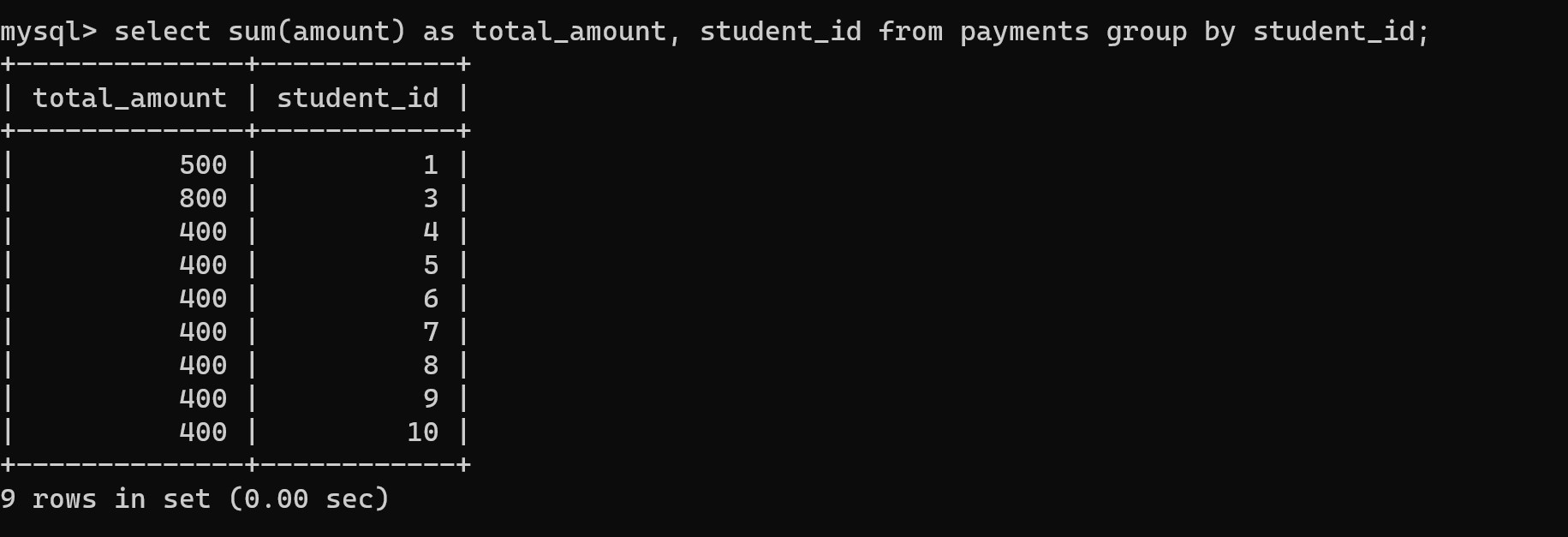
1. Select distinct statement



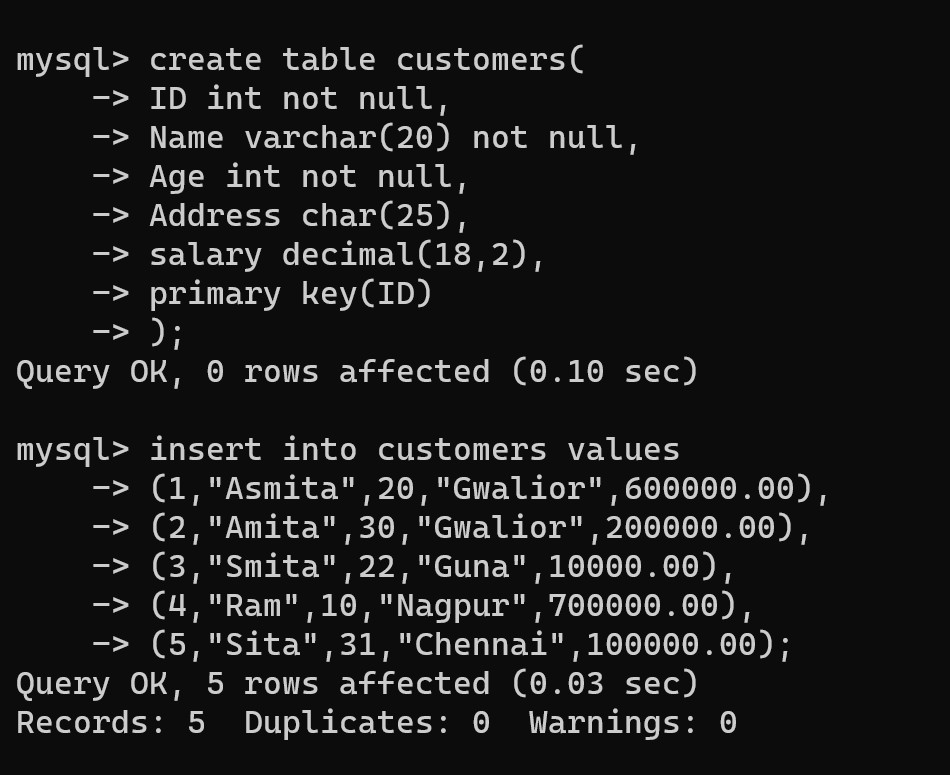
1. Having clause

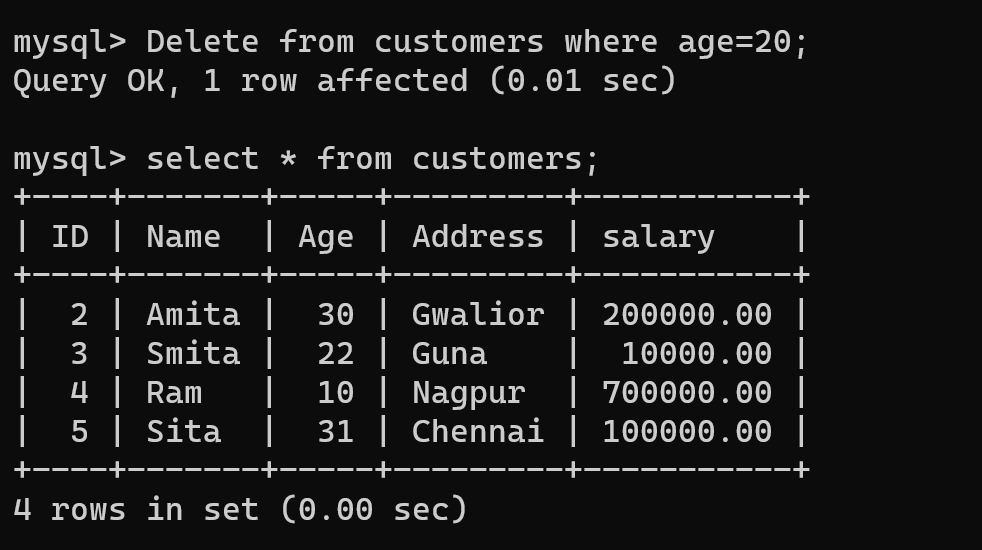


1. Group by clause



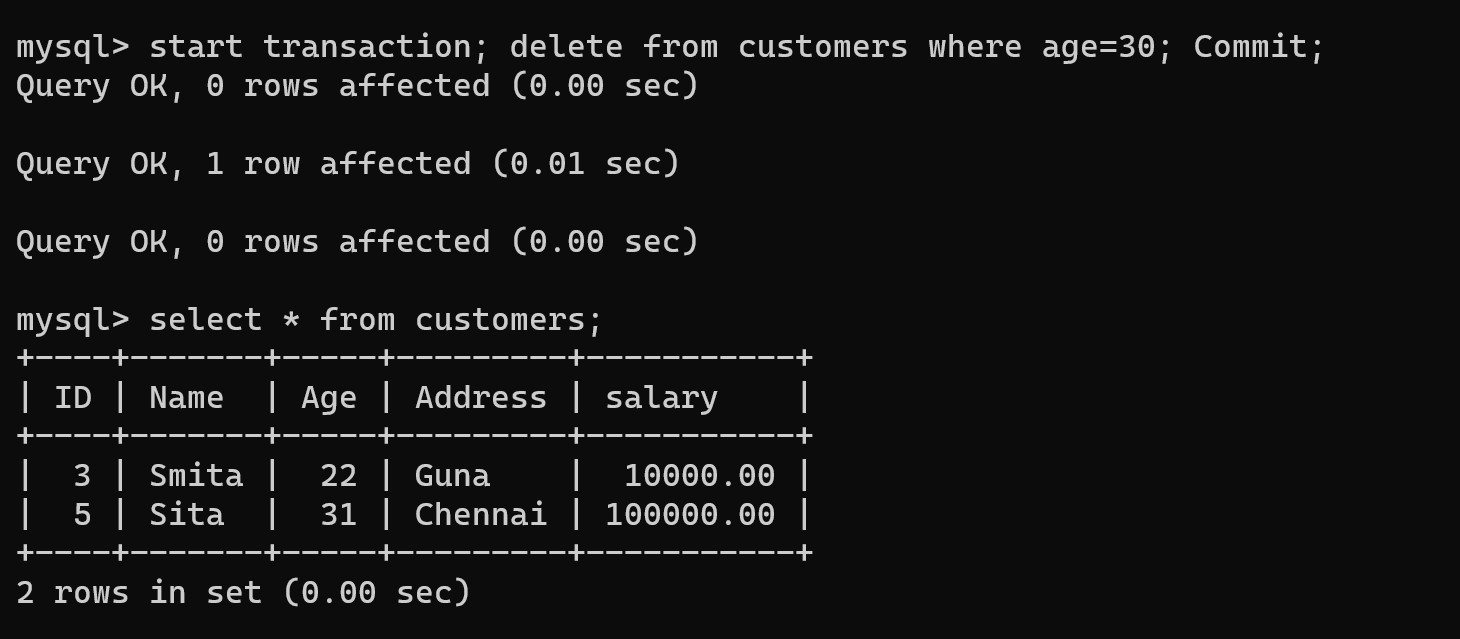
1. Sql Transactions





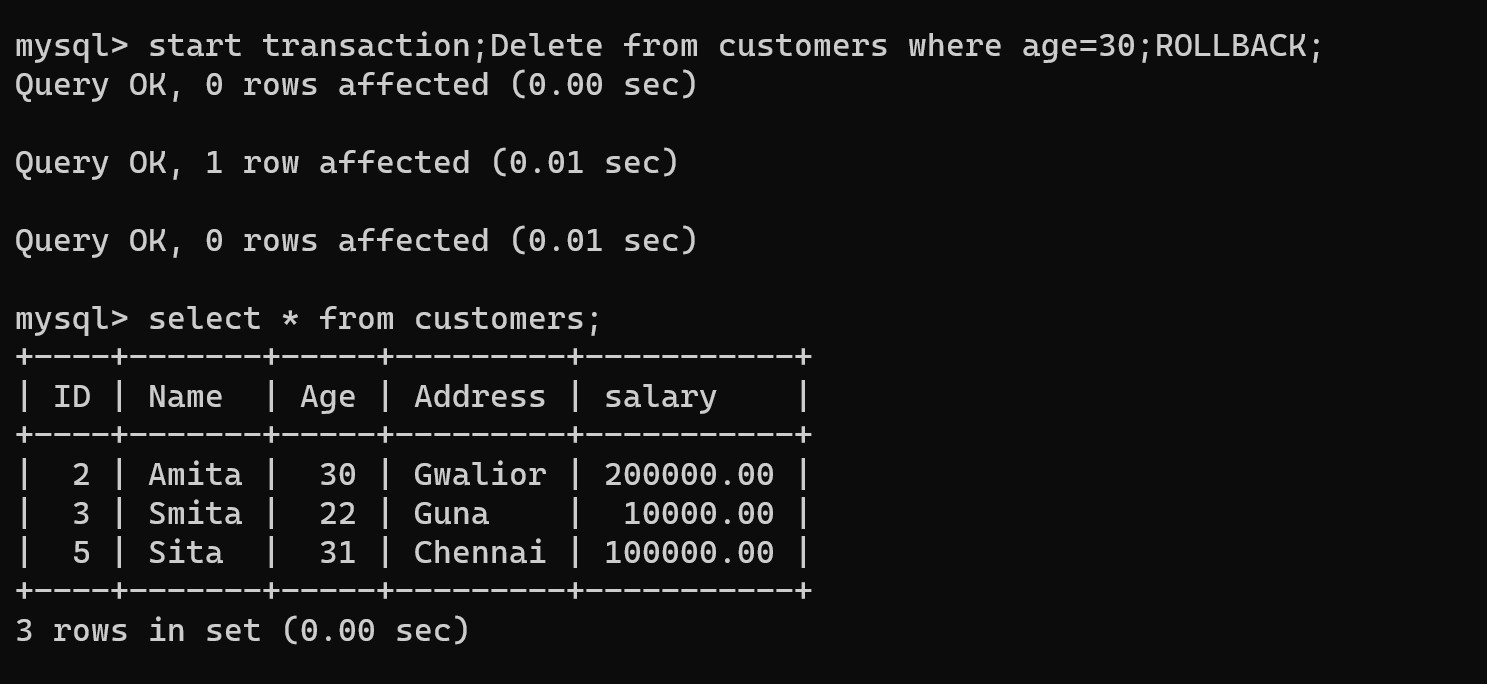
1. Commit

to save the changes



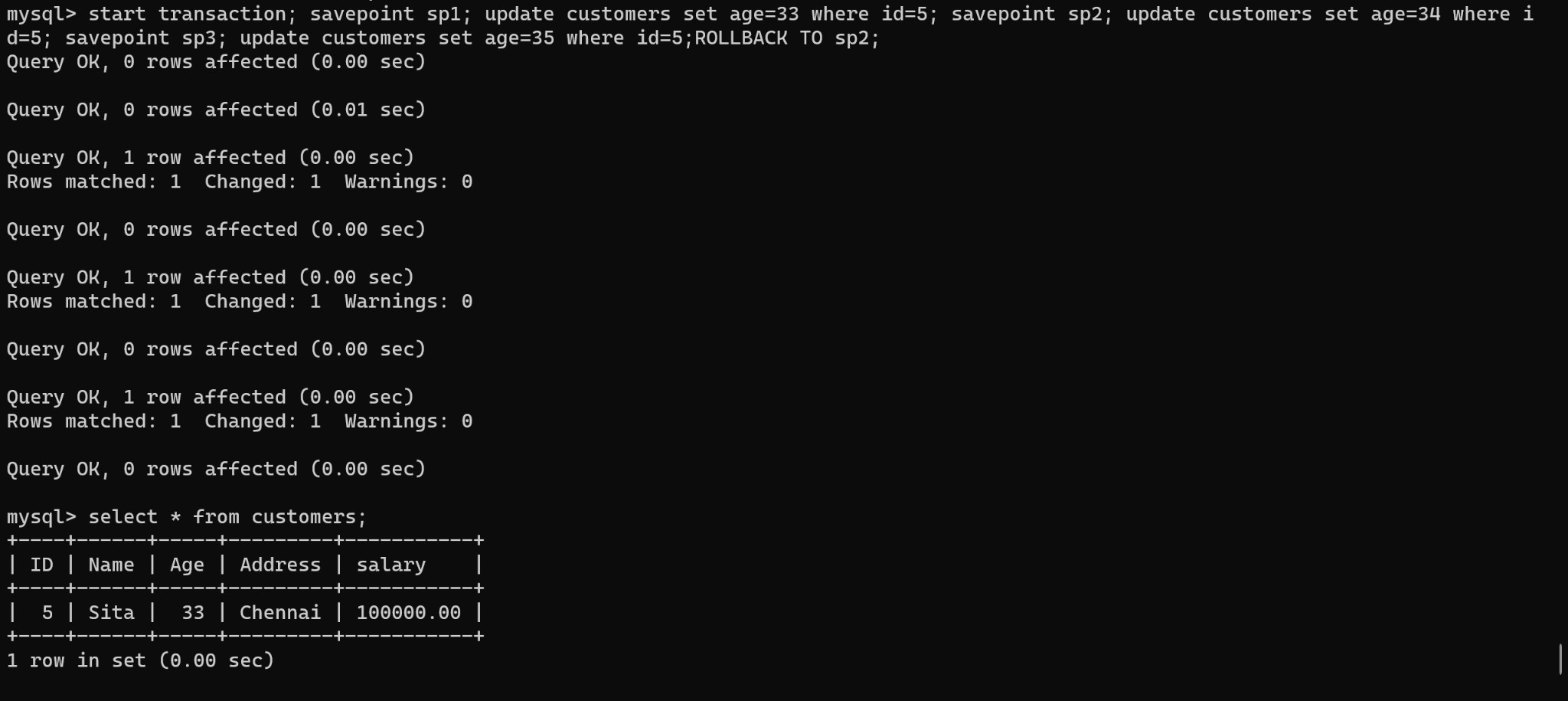
1. Rollback

to roll back the changes

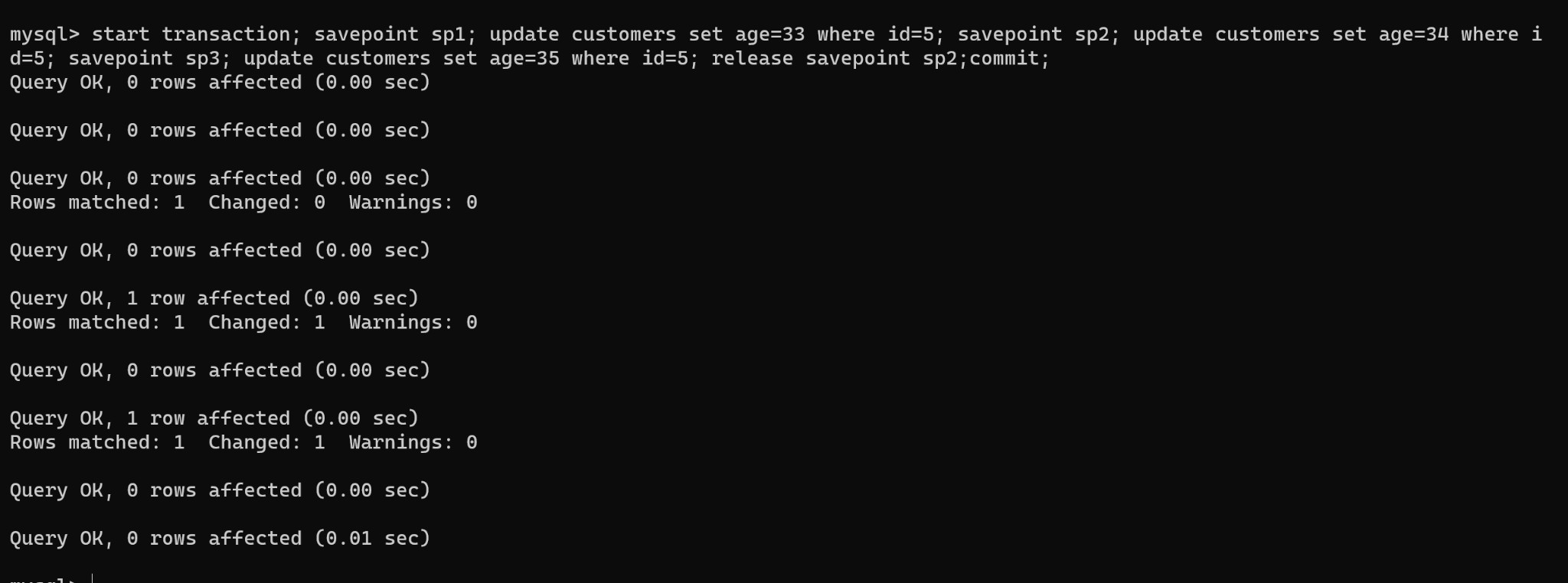


1. Savepoint

creates points within the groups of transactions in which to ROLLBACK.

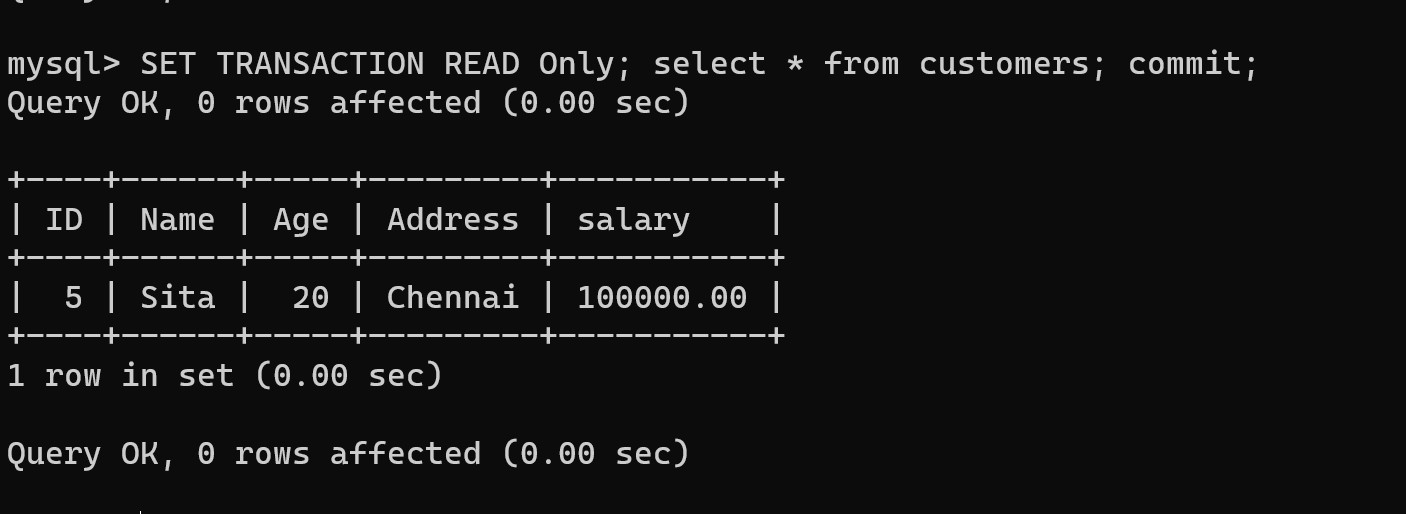
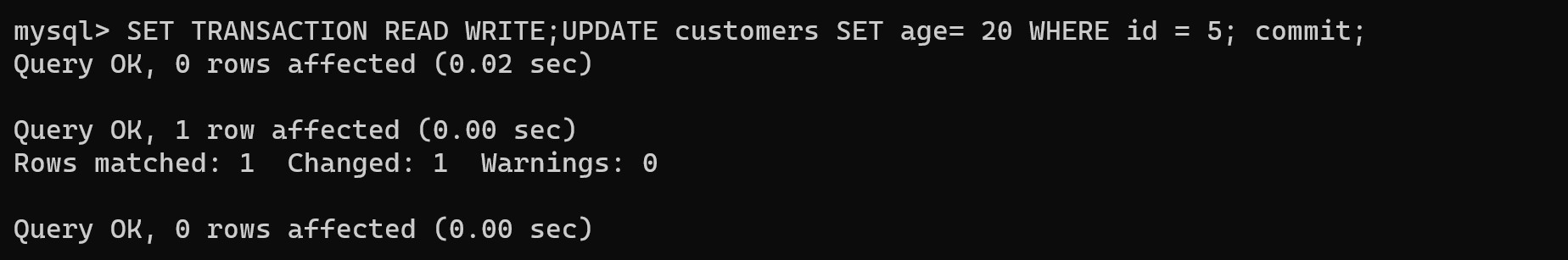


1. Release Savepoint

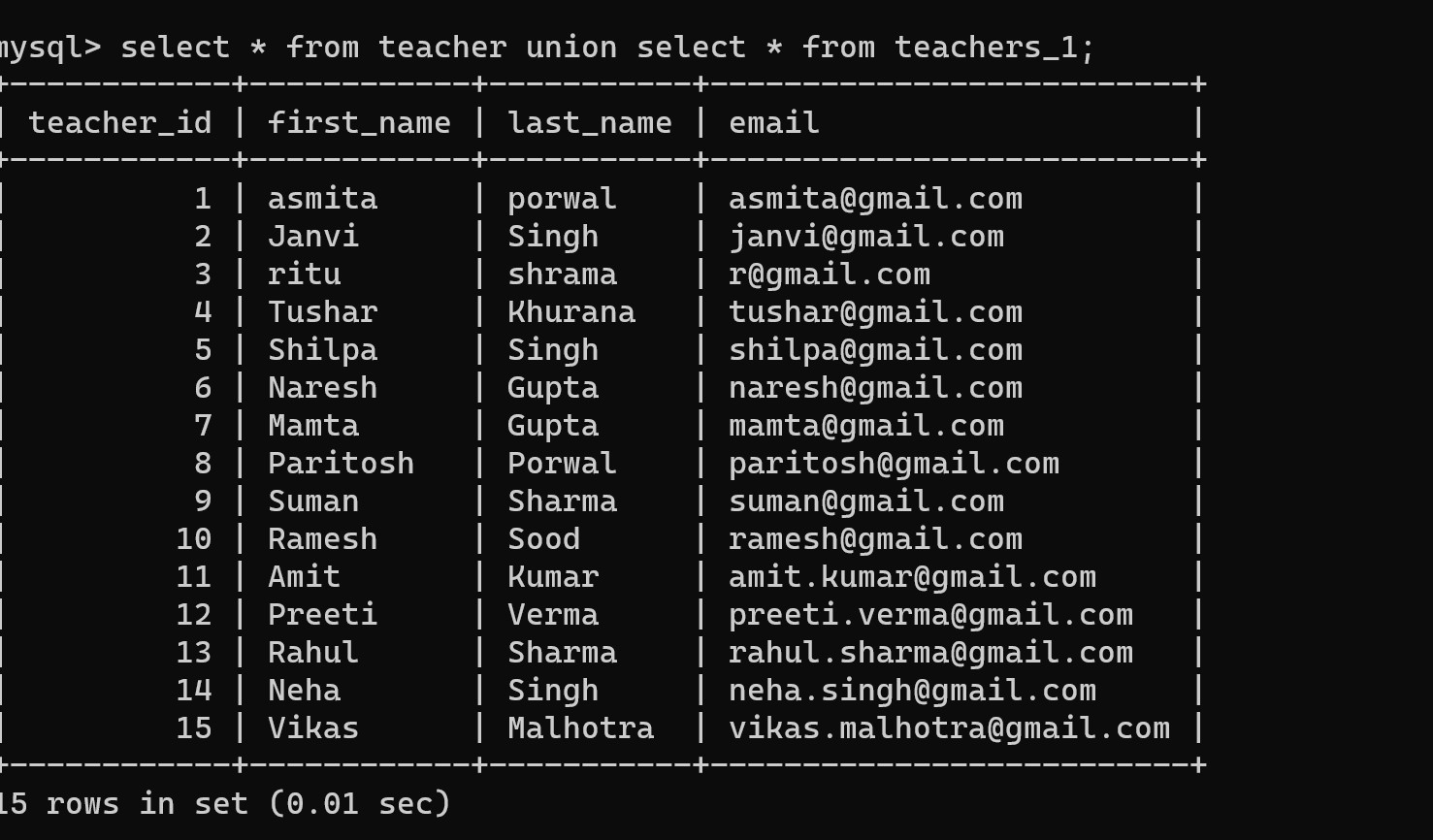


1. Set Transaction

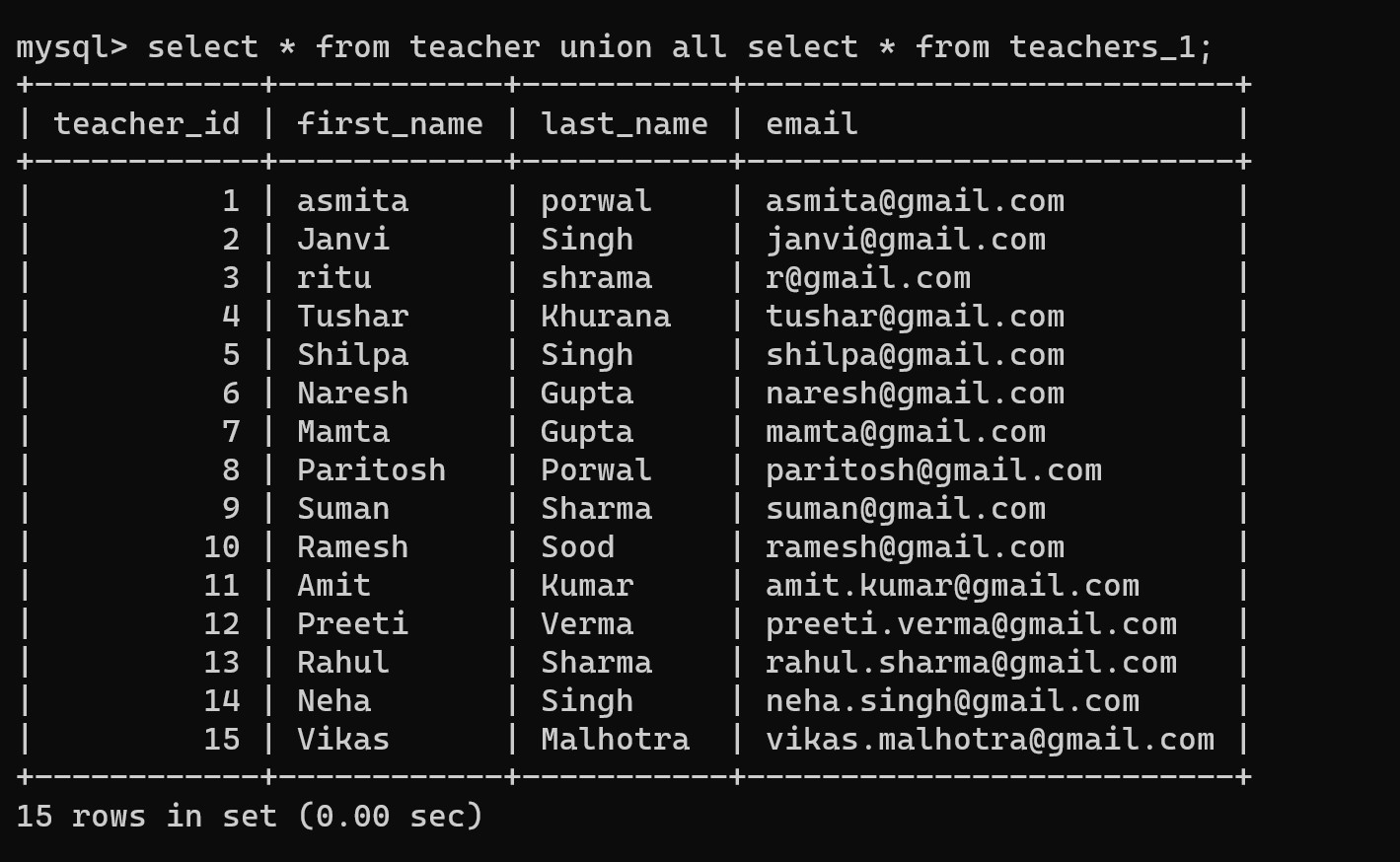
Place a name on a transaction



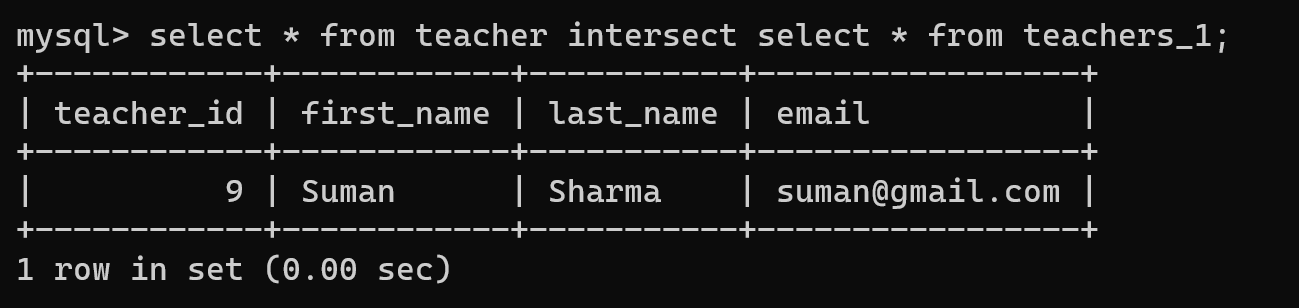
1. Set operators 11.Union



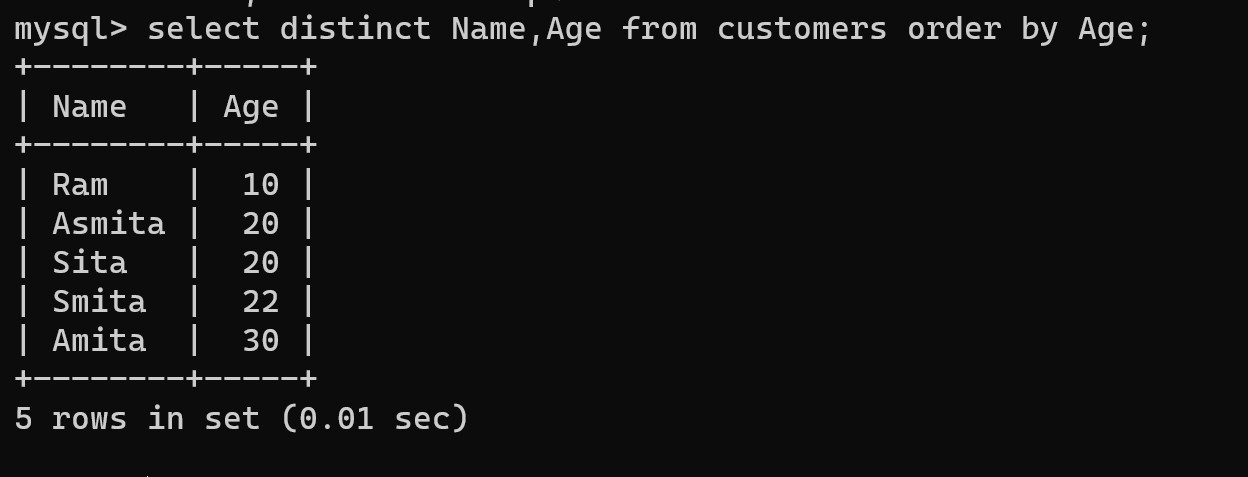
1. Union all



1. Intersect



1. Distinct Clause with Order By



## New database

CREATE DATABASE pet\_adoption;

## use database

USE pet\_adoption;

## Create table

CREATE TABLE animals (animal\_id int NOT NULL, name varchar(100),

breed varchar(100), color varchar(100),

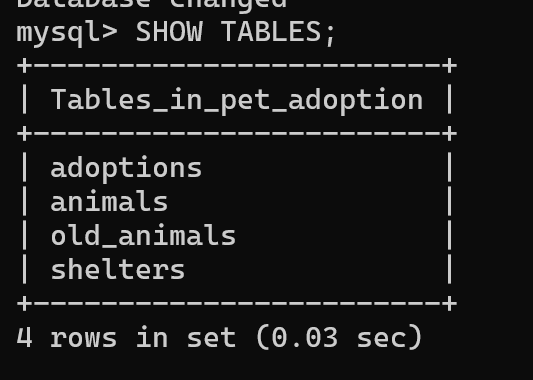
gender varchar(100), status INTEGER);

CREATE TABLE adoptions (animal\_id int NOT NULL, name varchar(100), contact varchar(10), date TIMESTAMP);

CREATE TABLE shelters (id INTEGER, name varchar(100), location varchar(100));

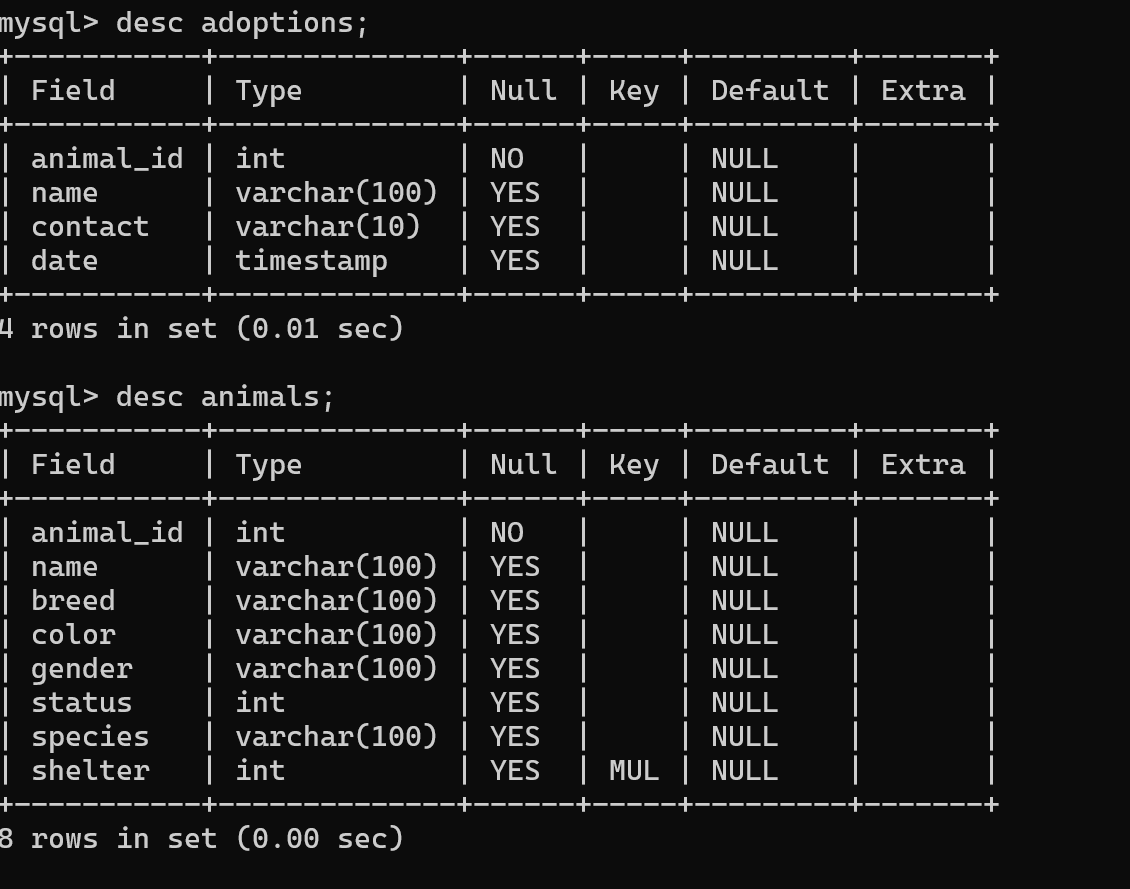
## All tables

SHOW TABLES;



## show all columns

desc adoptions; desc animals;



## Insert values

-- INSERT INTO animals (animal\_id, name, breed, color, gender, status) VALUES

-- (1, 'Bellyflop', 'Beagle', 'Brown', 'Male', 0),

-- (2, 'Snowy', 'Husky', 'White', 'Female', 0),

-- (3, 'Princess', 'Pomeranian', 'Black', 'Female', 0),

-- (4,'Cricket', 'Chihuahua', 'Brown', 'Male', 0),

-- (5,'Princess', 'Poodle', 'Purple', 'Female', 0),

-- (6,'Spot', 'Dalmation', 'Black and White', 'Male', 0);

INSERT INTO animals (animal\_id, name, species, breed, color, gender, status) VALUES (7, 'Meowmix', 'Cat', 'Munchkin', 'Yellow', 'Female', 0);

-- INSERT INTO animals (animal\_id, name, species, breed, color, gender, status) VALUES (8, 'Ash', 'Cat', 'Persian', 'Gray', 'Female', 0);

-- INSERT INTO animals (animal\_id, name, species, breed, color, gender, status) VALUES (9, 'Tiger', 'Cat', 'Bengal', 'Brown', 'Male', 0);

INSERT INTO shelters (id, name, location) VALUES (1, 'Animals 4 Homes', 'Red City');

-- INSERT INTO shelters (id, name, location) VALUES (2, 'Adopt A Buddy', 'Green Town');

-- INSERT INTO shelters (id, name, location) VALUES (3, 'Fluffy Animals', 'Blue Hills');

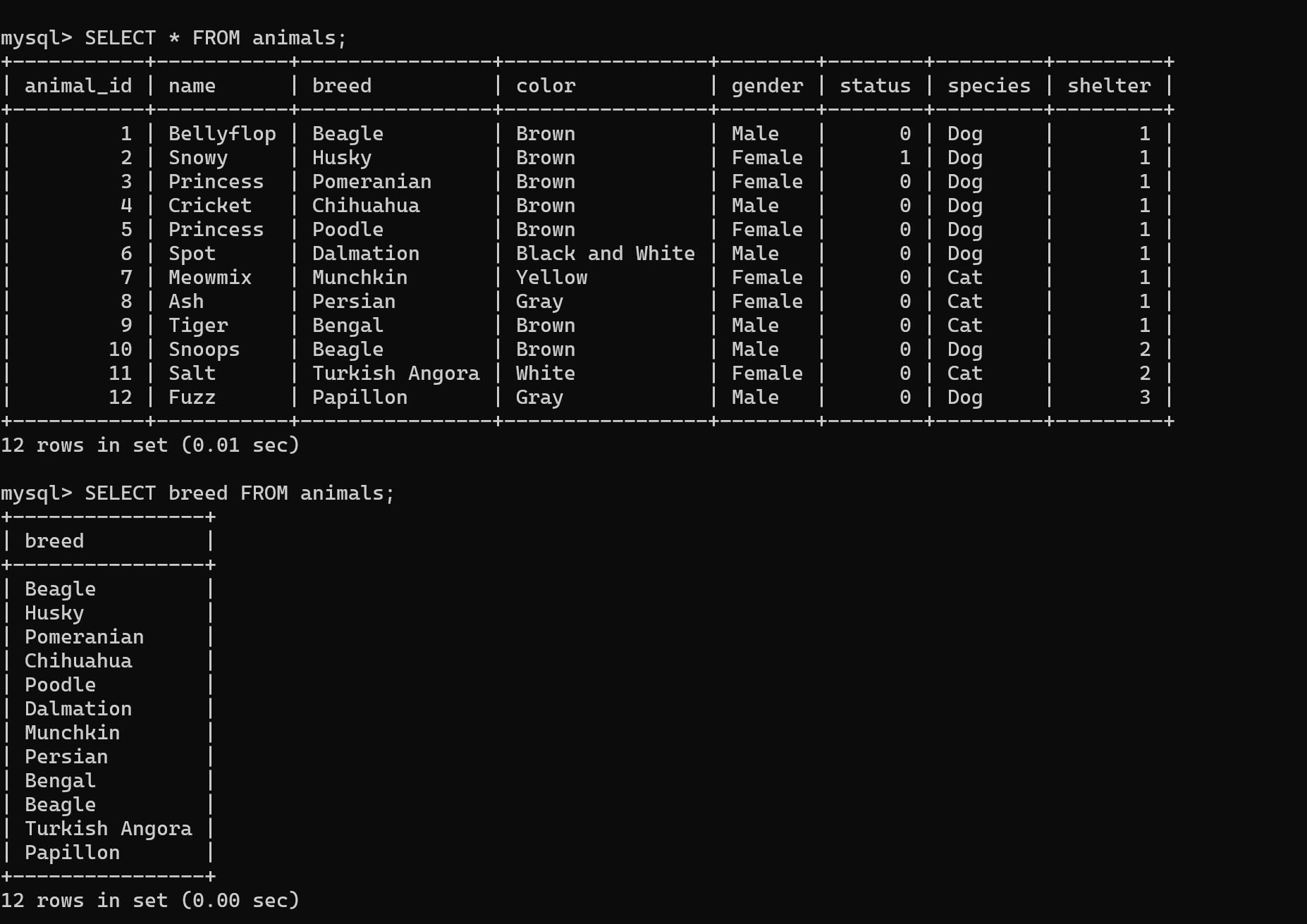
-- INSERT INTO animals (animal\_id, name, shelter, species, breed, color, gender, status) VALUES (10, 'Snoops', 2, 'Dog', 'Beagle', 'Brown', 'Male', 0);

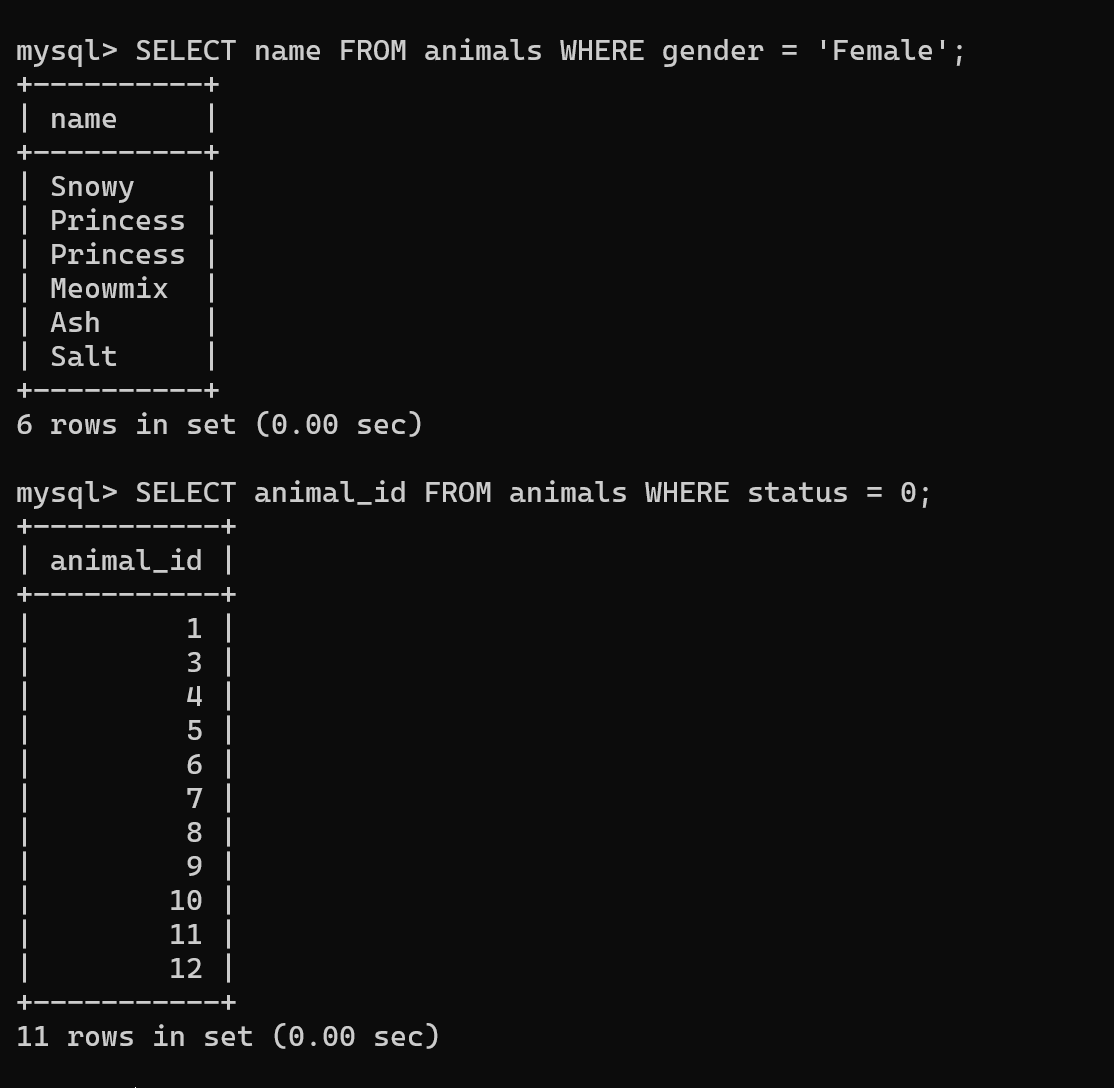
-- INSERT INTO animals (animal\_id, name, shelter, species, breed, color, gender, status) VALUES (11, 'Salt', 2, 'Cat', 'Turkish Angora', 'White', 'Female', 0);

-- INSERT INTO animals (animal\_id, name, shelter, species, breed, color, gender, status) VALUES (12, 'Fuzz', 3, 'Dog', 'Papillon', 'Gray', 'Male', 0);

1. **show animal table** SELECT \* FROM animals; SELECT breed FROM animals;

SELECT name FROM animals WHERE gender = 'Female'; SELECT animal\_id FROM animals WHERE status = 0;



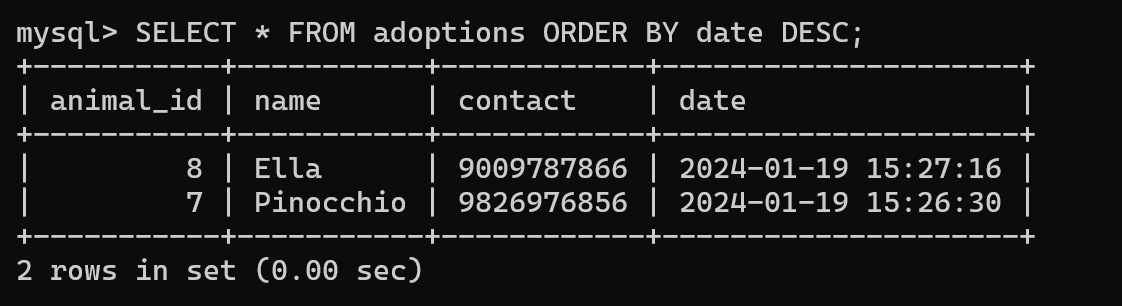


## Update query

UPDATE animals SET color = 'Brown' WHERE animal\_id = 2 LIMIT 1; UPDATE animals SET color = 'Brown' WHERE name = 'Princess' LIMIT 1; UPDATE animals SET color = 'Brown' WHERE color = 'Purple'LIMIT 1; UPDATE animals SET status = 1 WHERE animal\_id =2 LIMIT 1;

## Order by

SELECT \* FROM adoptions ORDER BY date DESC;



## Alter

ALTER TABLE animals ADD COLUMN species varchar(100); ALTER TABLE animals ADD COLUMN shelter INTEGER;

## Turn off safe update

SET sql\_safe\_updates = FALSE; UPDATE animals SET shelter = 1;

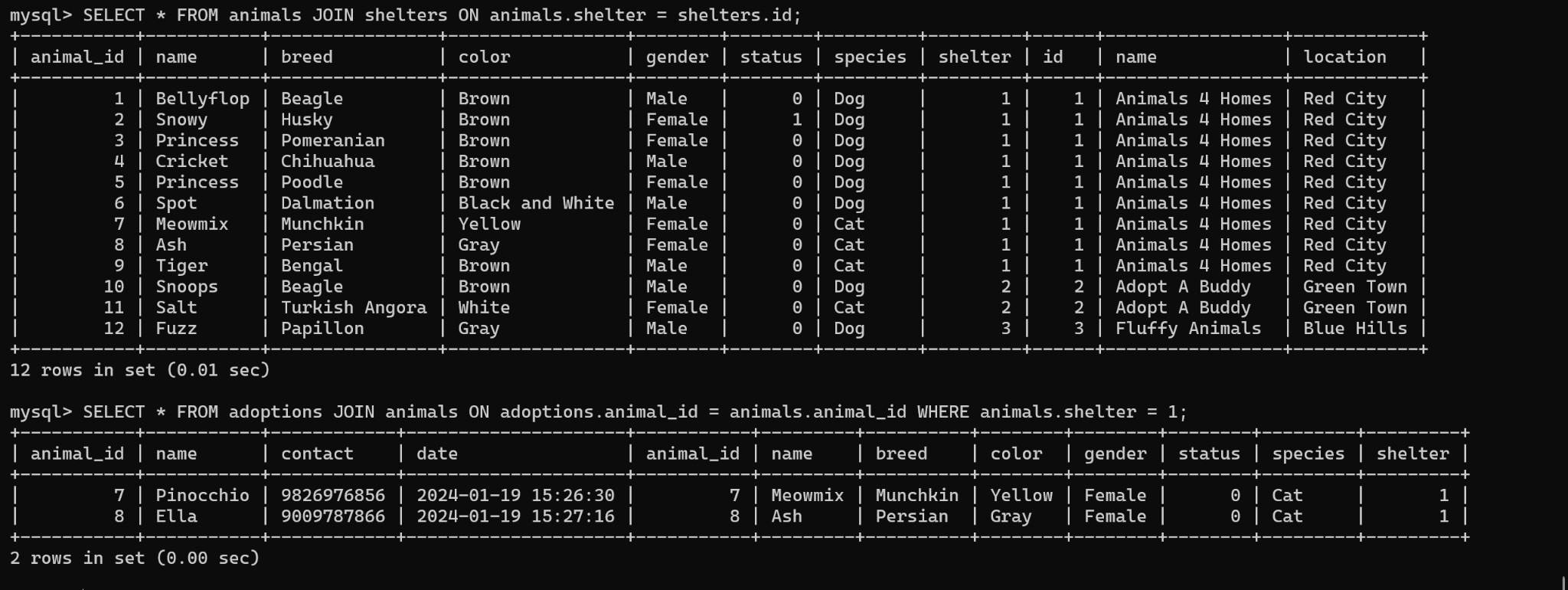
## Indexing

CREATE INDEX animal\_shelter ON animals (shelter);

## Join

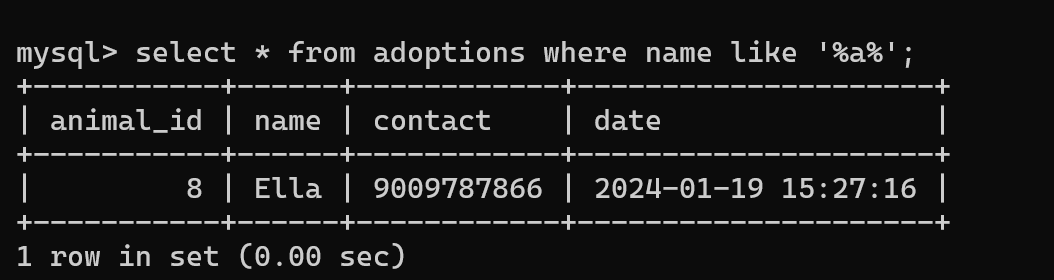
SELECT \* FROM animals JOIN shelters ON animals.shelter = shelters.id;

SELECT \* FROM adoptions JOIN animals ON adoptions.animal\_id = animals.animal\_id WHERE animals.shelter = 1;



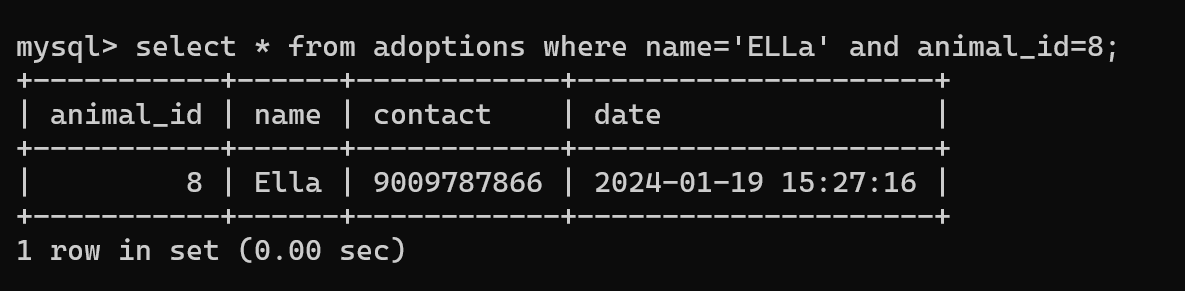
## Like operator

select \* from adoptions where name like '%a%';



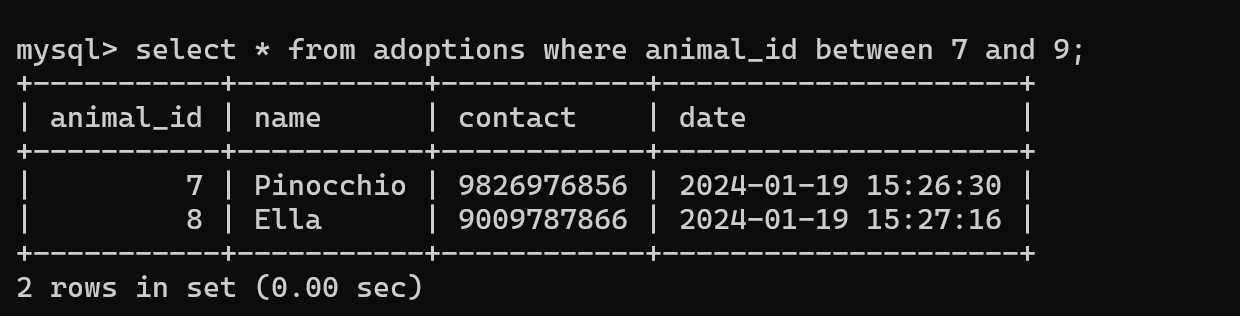
## And operator

select \* from adoptions where name='ELLa' and animal\_id=8;



## Between operator

select \* from adoptions where animal\_id between 7 and 9;



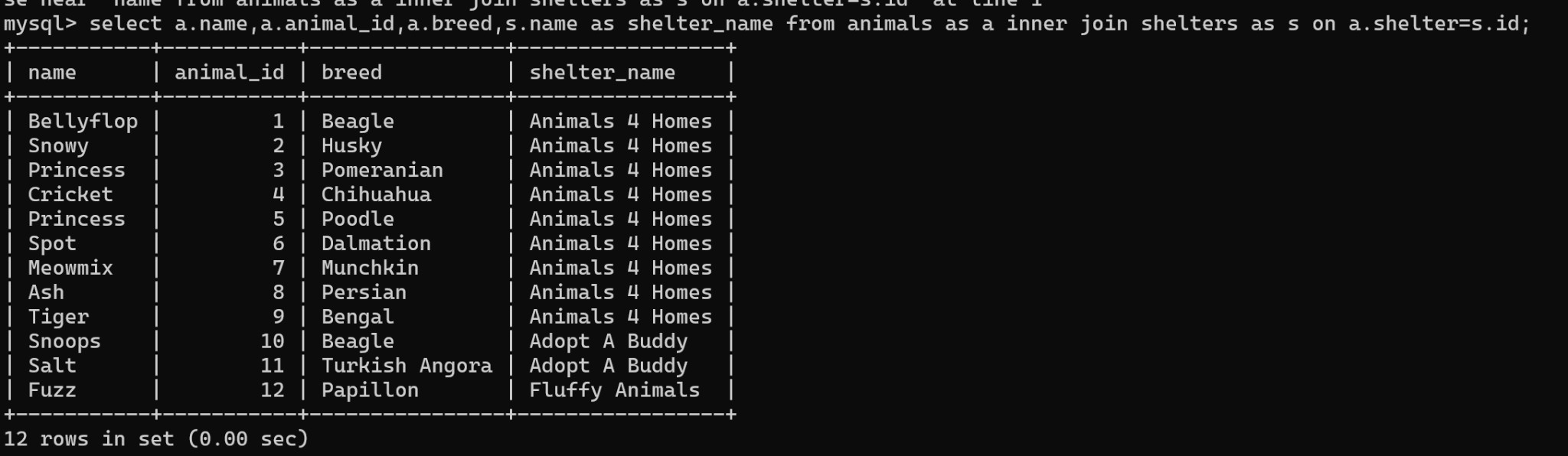
## Sql Joins

It is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

* + INNER JOIN
  + LEFT JOIN
  + RIGHT JOIN
  + FULL JOIN
  + NATURAL JOIN

# INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.



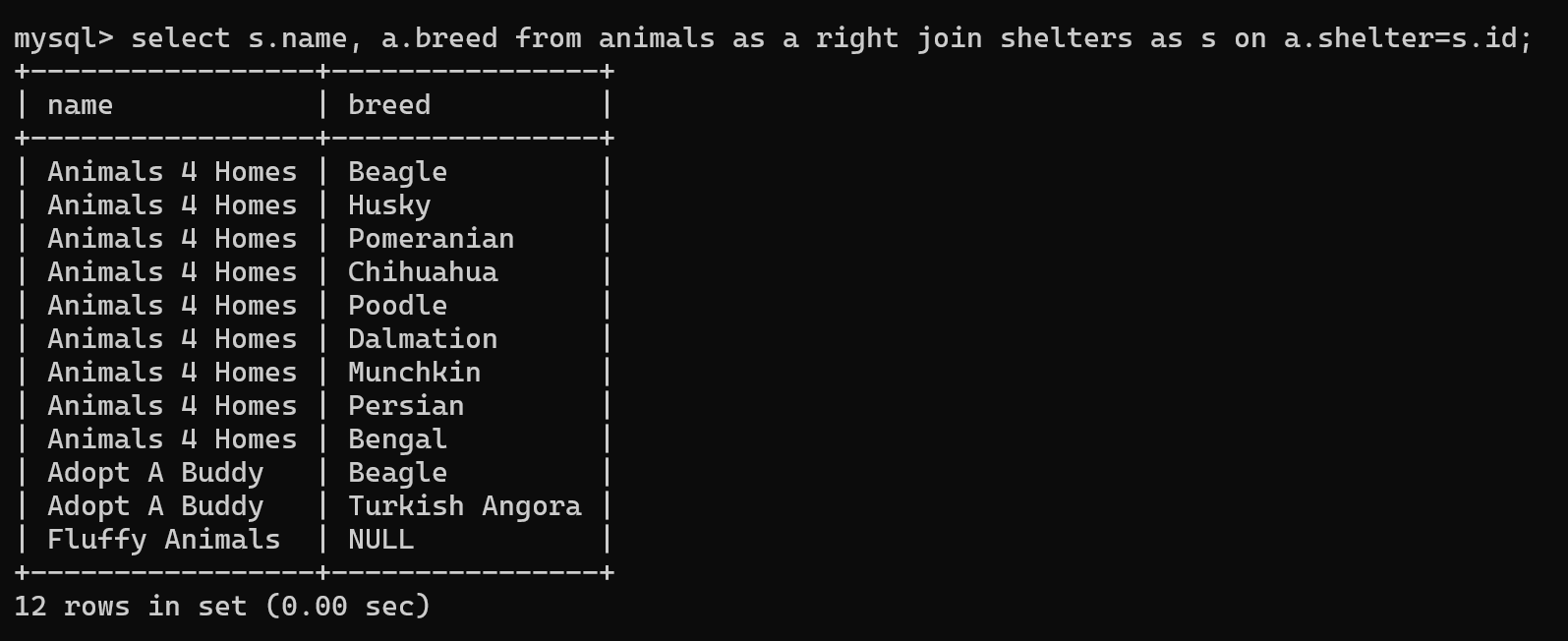
# LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.



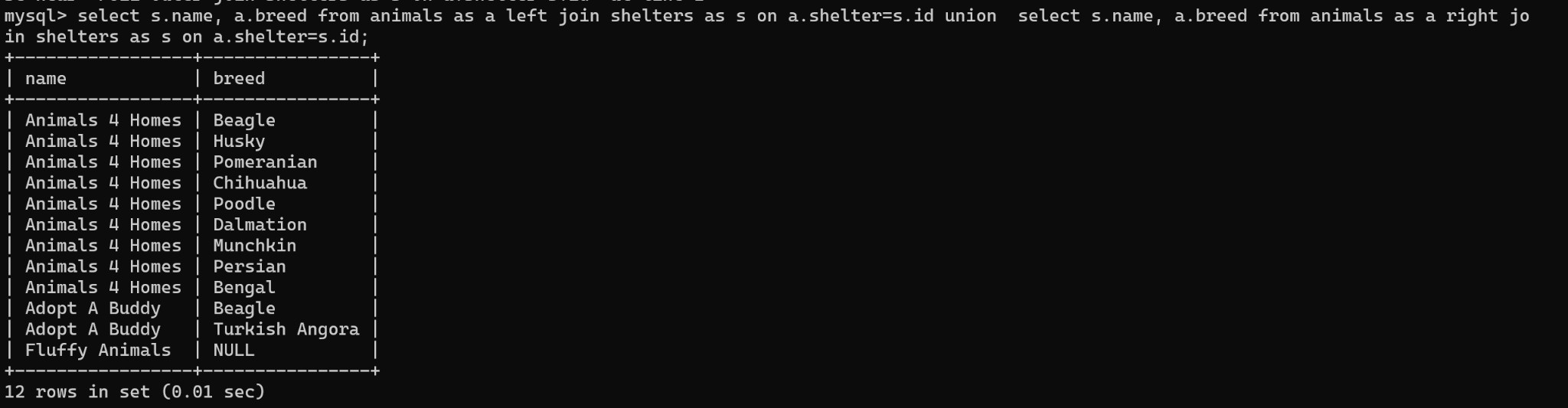
# RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.



# FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.



# NATURAL JOIN

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having the same name and data type of columns and that column should be present in both tables.

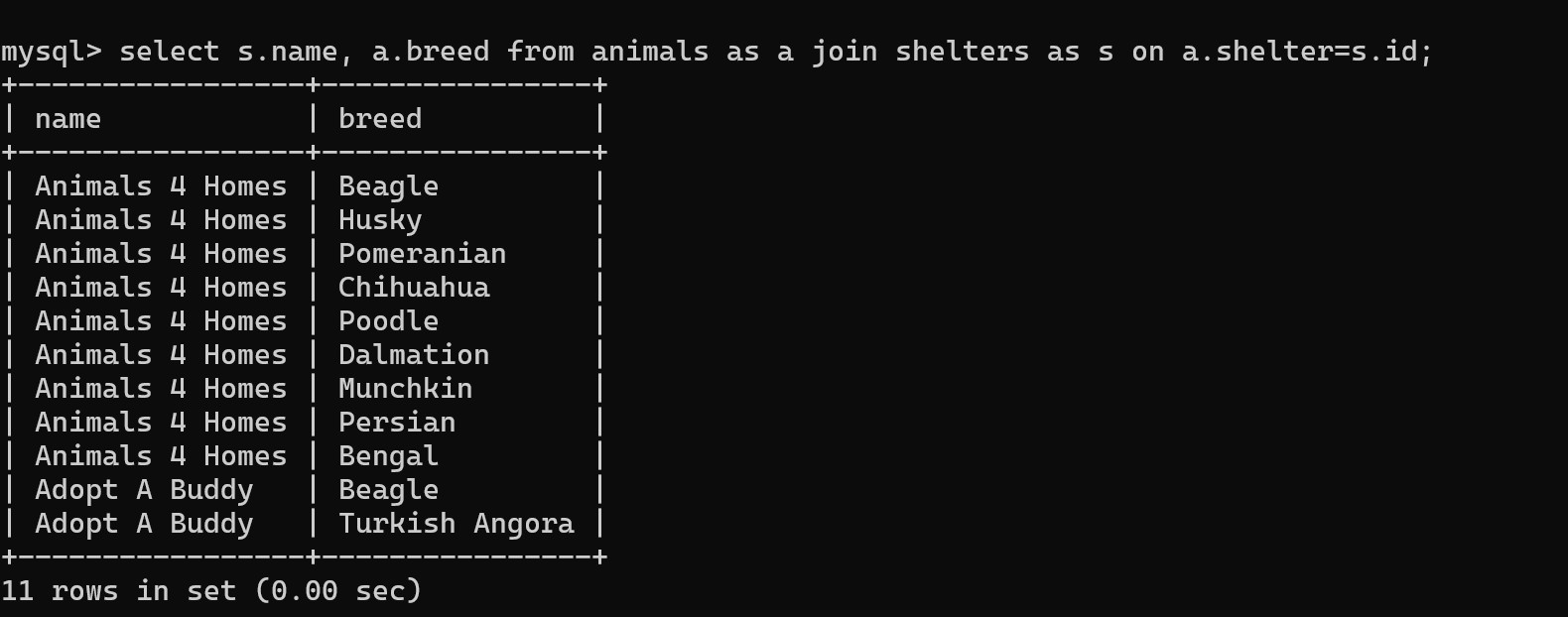
Both tables must have at least one common column with the same column name and same data type.

The two tables are joined using Cross join.

DBMS will look for a common column with the same name and data type Tuples having exactly the same values in common columns are kept in the result.

# EQUI JOIN

EǪUI JOIN creates a JOIN for equality or matching column(s) values of the relative tables. EǪUI JOIN also creates JOIN by using JOIN with ON and then providing the names of the columns with their relative tables to check equality using equal sign (=).



# NON EQUI JOIN

NON EǪUI JOIN performs a JOIN using comparison operators other than equal(=) signs like >, <, >=, <= with conditions.

