**VAISHALI BOKADIYA**

**DAY 3 ASSESSMENT**

**SQL COMMANDS**

**SQL**

**MAJOR COMPONENTS OF SQL:**

* Data Definition Language (DDL):
  + It enables user to design and define the structure of the database.
  + It includes commands like CREATE, DROP, ALTER, RENAME etc.
* Data Manipulation Language (DML):
  + It enables user to update, delete and retrieve data from the database.
  + It includes commands like SELECT, INSERT, UPDATE, DELETE etc.
* Data Control Language (DCL):
  + It enables user to control the security of the database.
  + It includes commands like GRANT and REVOKE

**CREATE A DATABASE:**

CREATE DATABASE statement in SQL is used to create a database.

SYNTAX: CREATE DATABASE DATABASE\_NAME;

EXAMPLE:

CREATE DATABASE CUSTOMER\_DATABASE;

**USE DATABASE:**

USE statement in SQL is used to select a specific database on which the command will be executed.

SYNTAX: USE DATABASE\_NAME;

EXAMPLE:

USE CUSTOMER\_DATABASE;

**CREATE A TABLE:**

CREATE TABLE statement in SQL is used to create a table.

SYNTAX: CREATE TABLE TABLE\_NAME;

EXAMPLE:

CREATE TABLE CUSTOMERS (

CustomerId INT PRIMARY KEY NOT NULL,

CustomerName VARCHAR(50) NOT NULL,

ContactName VARCHAR(50),

Address VARCHAR(50),

City VARCHAR(20),

PostalCode VARCHAR(10),

Country VARCHAR(20)

);

CREATE TABLE CUSTOMERS2 (

CustomerId INT PRIMARY KEY NOT NULL,

CustomerName VARCHAR(50) NOT NULL,

ContactName VARCHAR(50),

Address VARCHAR(50),

City VARCHAR(20),

PostalCode VARCHAR(10),

Country VARCHAR(20)

);

**INSERT INTO TABLES:**

INSERT INTO statement in SQL is used to insert data into a table.

SYNTAX: INSERT INTO TABLE\_NAME (COLUMN1, COLUMN2…) VALUES ();

EXAMPLE:

INSERT INTO CUSTOMERS

(CustomerId, CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES

(101, 'Alfreds Futterkiste', 'Maria Anders', 'Obere Str. 57', 'Berlin', '12209', 'Germany' ),

(102, 'Ana Trujillo Emparedados y helados', 'Ana Trujillo', 'Avda. de la Constitución 2222', 'México D.F.', '05021', 'Mexico'),

(103, 'Antonio Moreno Taquería', 'Antonio Moreno', 'Mataderos 2312', 'México D.F.', '05023', 'Mexico'),

(104, 'Around the Horn', 'Thomas Hardy', '120 Hanover Sq.', 'London', 'WA1 1DP', 'UK'),

(105, 'Berglunds snabbköp', 'Christina Berglund', 'Berguvsvägen 8', 'Luleå', 'S-958 22', 'Sweden');

INSERT INTO CUSTOMERS2

(CustomerId, CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES

(201, 'John Doe', 'Jane Smith', '123 Main St', 'Anytown', '12345', 'USA'),

(202, 'Alice Johnson', 'Bob Miller', '456 Oak Ave', 'Somewhere', '67890', 'Canada'),

(203, 'Eva Brown', 'Michael White', '789 Pine Ln', 'Nowhere', '54321', 'Australia'),

(105, 'Berglunds snabbköp', 'Christina Berglund', 'Berguvsvägen 8', 'Luleå', 'S-958 22', 'Sweden');

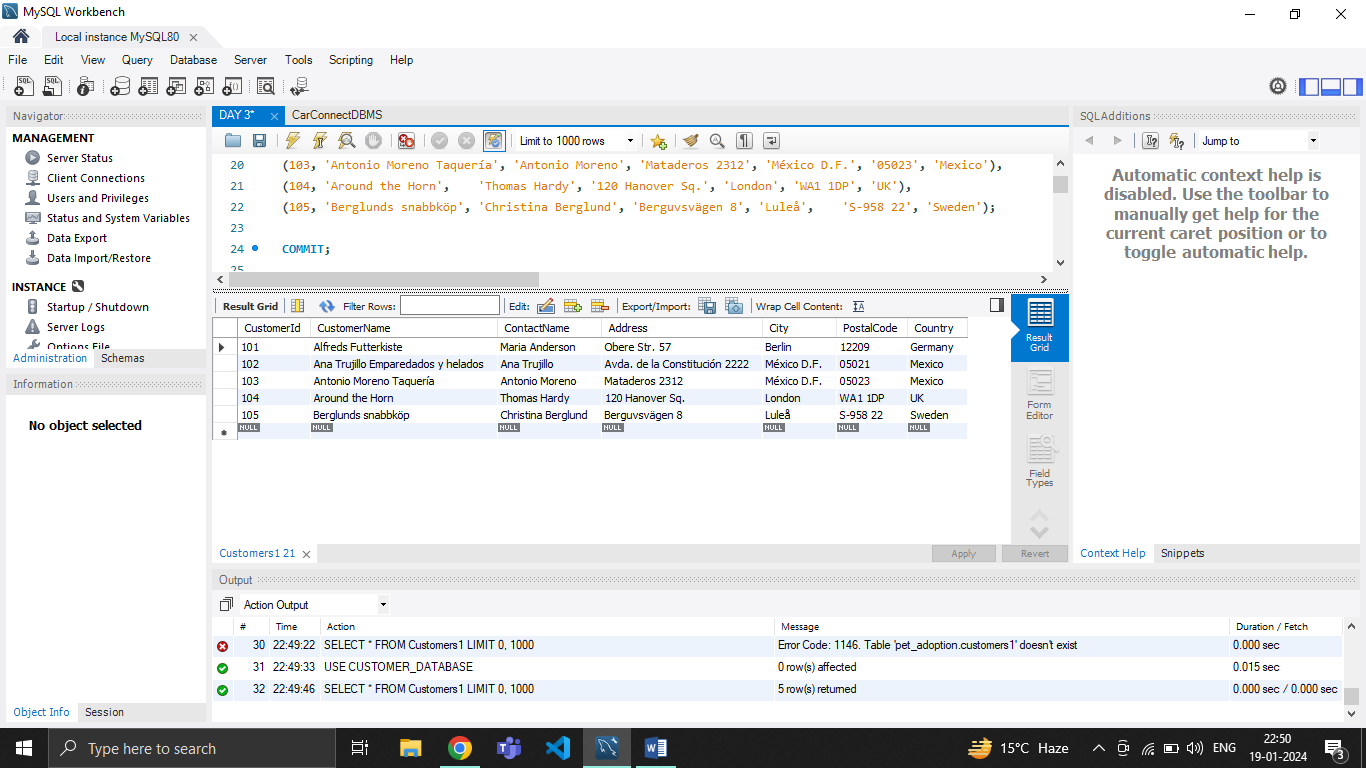
**SELECT:**

SELECT statement is used to query data from one or more tables.

SYNTAX: SELECT COLUMN\_NAME1, COLUMN\_NAME2… FROM TABLE\_NAME;

EXAMPLE:

SELECT \* FROM Customers;



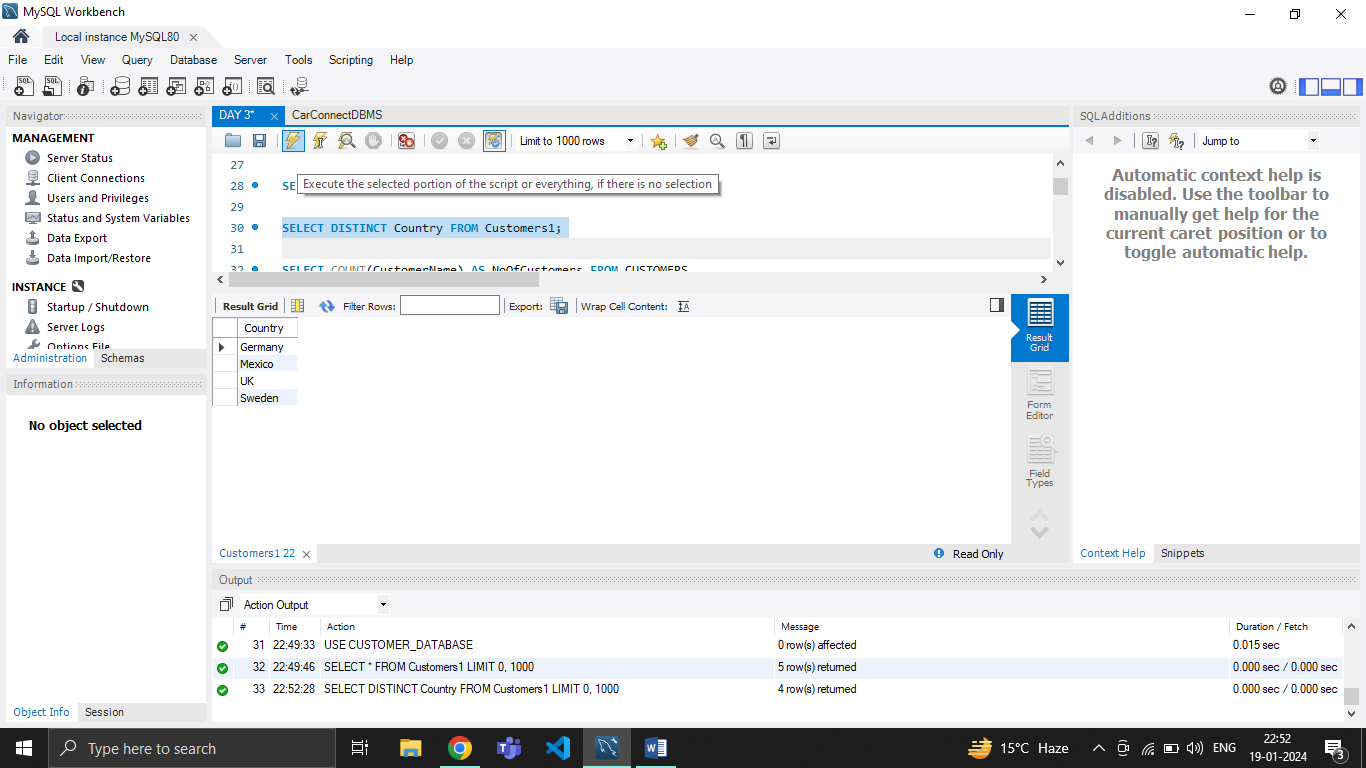
**SELECT DISTINCT:**

SELECT statement is used to distinct query data from one or more tables.

SYNTAX: SELECT DISTINCT COLUMN\_NAME FROM TABLE\_NAME;

EXAMPLE:

SELECT DISTINCT Country FROM Customers;



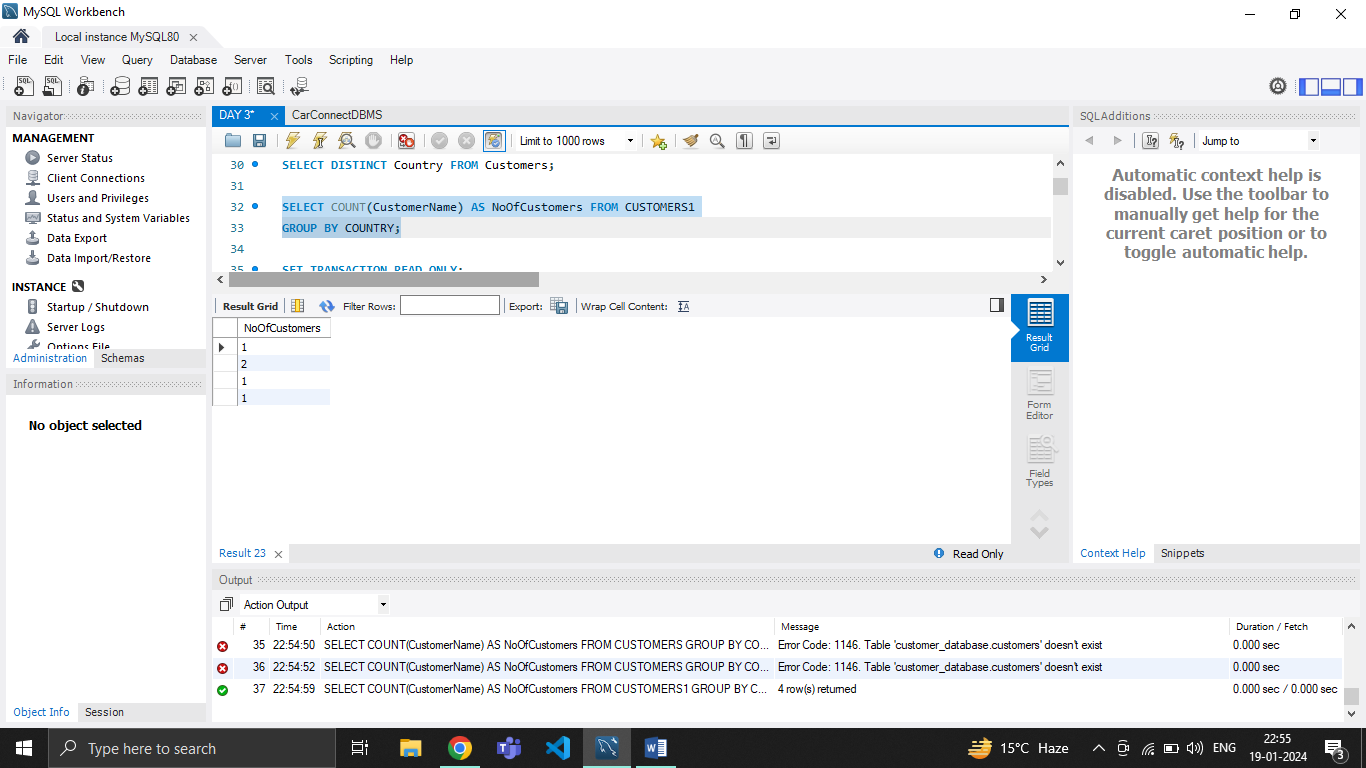
**GROUP BY:**

GROUP BY clause in SQL is used to group rows that have the same values in specified columns.

EXAMPLE:

SELECT COUNT(CustomerName) AS NoOfCustomers FROM CUSTOMERS

GROUP BY COUNTRY;



**HAVING:**

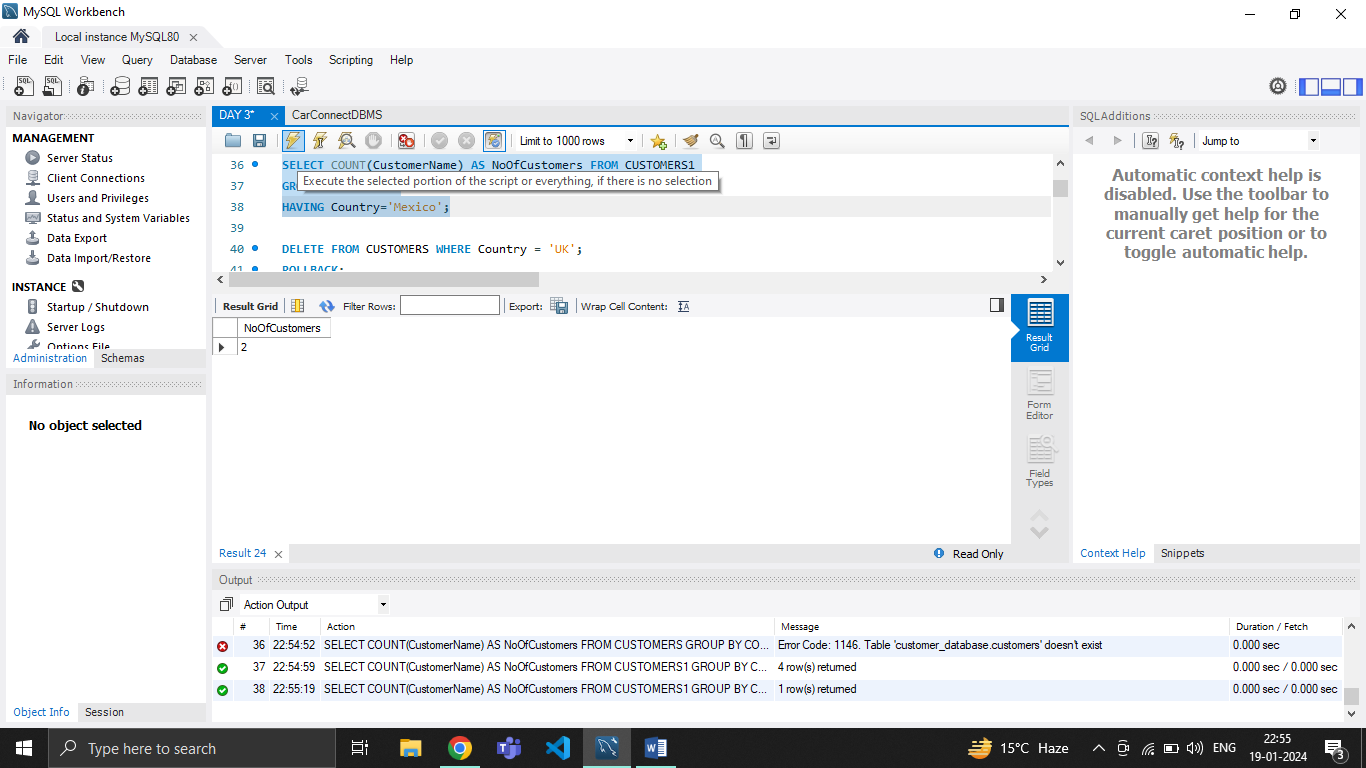
HAVING clause in SQL is used with the GROUP BY clause to filter the results of a query.

EXAMPLE:

SELECT COUNT(CustomerName) AS NoOfCustomers FROM CUSTOMERS

GROUP BY COUNTRY

HAVING Country='Mexico';



**DELETE:**

DELETE statement in SQL is used to delete records from a table.

SYNTAX: DELETE FROM table\_name WHERE condition;

EXAMPLE:

DELETE FROM CUSTOMERS WHERE Country = 'UK';

**UPDATE:**

UPDATE statement in SQL is used to modify existing records in a table.

EXAMPLE

UPDATE CUSTOMERS

SET ContactName='Maria Anderson'

WHERE CustomerId=101;

**RENAME A TABLE:**

RENAME TABLE statement in SQL is used to rename table.

SYNTAX: RENAME TABLE OLD\_TABLE\_NAME TO NEW\_TABLE\_NAME;

EXAMPLE:

RENAME TABLE CUSTOMERS TO CUSTOMERS1;

**UNION:**

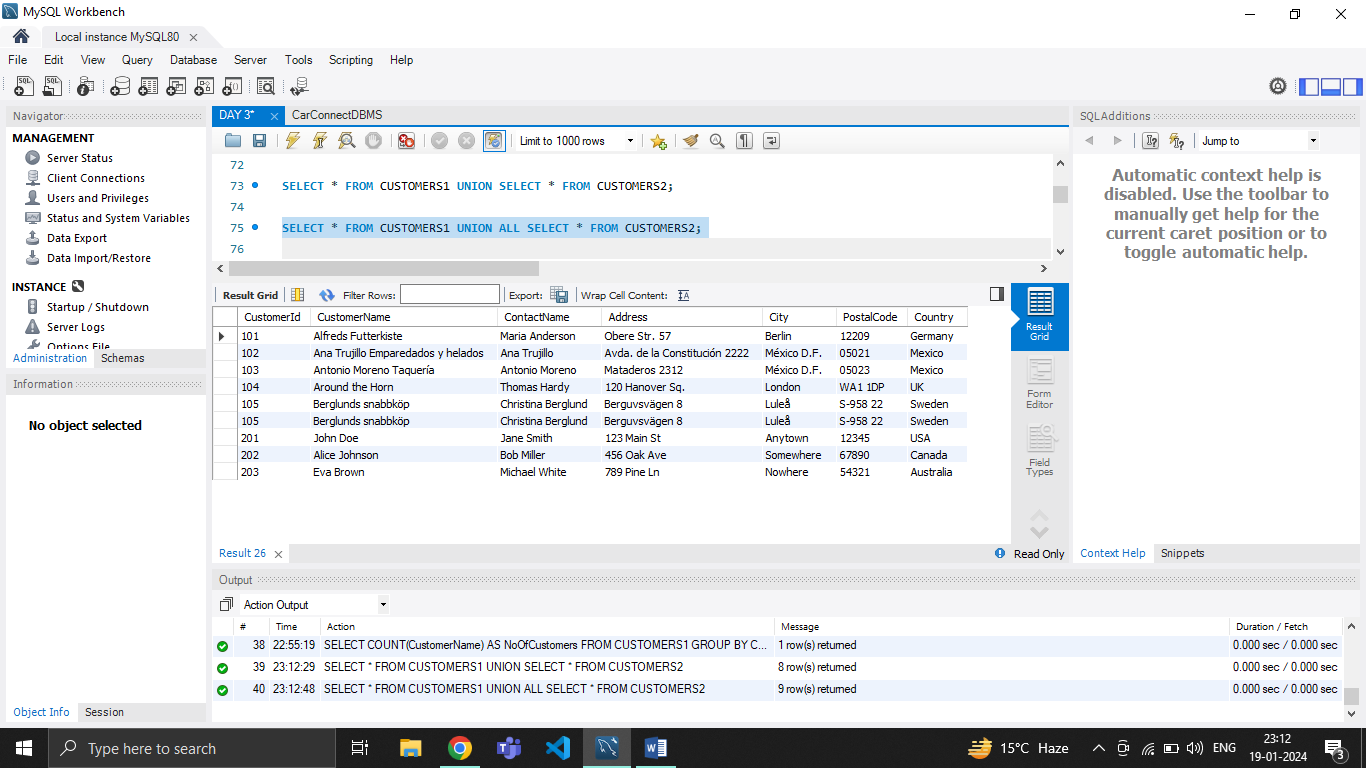
UNION operator in SQL is used to combine the results of two or more SELECT statements.

EXAMPLE:

SELECT \* FROM CUSTOMERS1

UNION

SELECT \* FROM CUSTOMERS2;



**UNION ALL:**

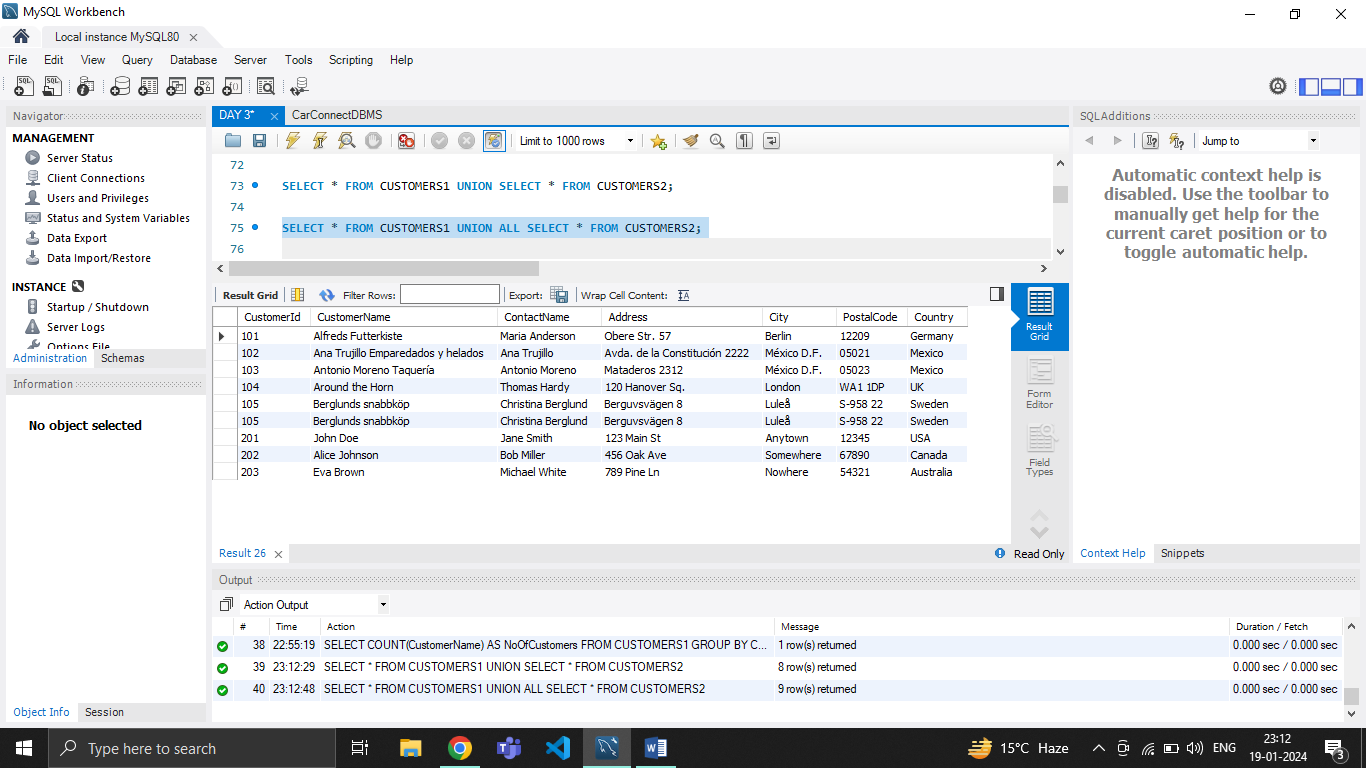
UNION ALL operator in SQL is used to combine the results of two or more SELECT statements without removing duplicates.

EXAMPLE:

SELECT \* FROM CUSTOMERS1

UNION ALL

SELECT \* FROM CUSTOMERS2;



**INTERSECT:**

INTERSECT operator in SQL is used to combine the result sets of two SELECT statements and retrieve only the rows that are common to both result sets.

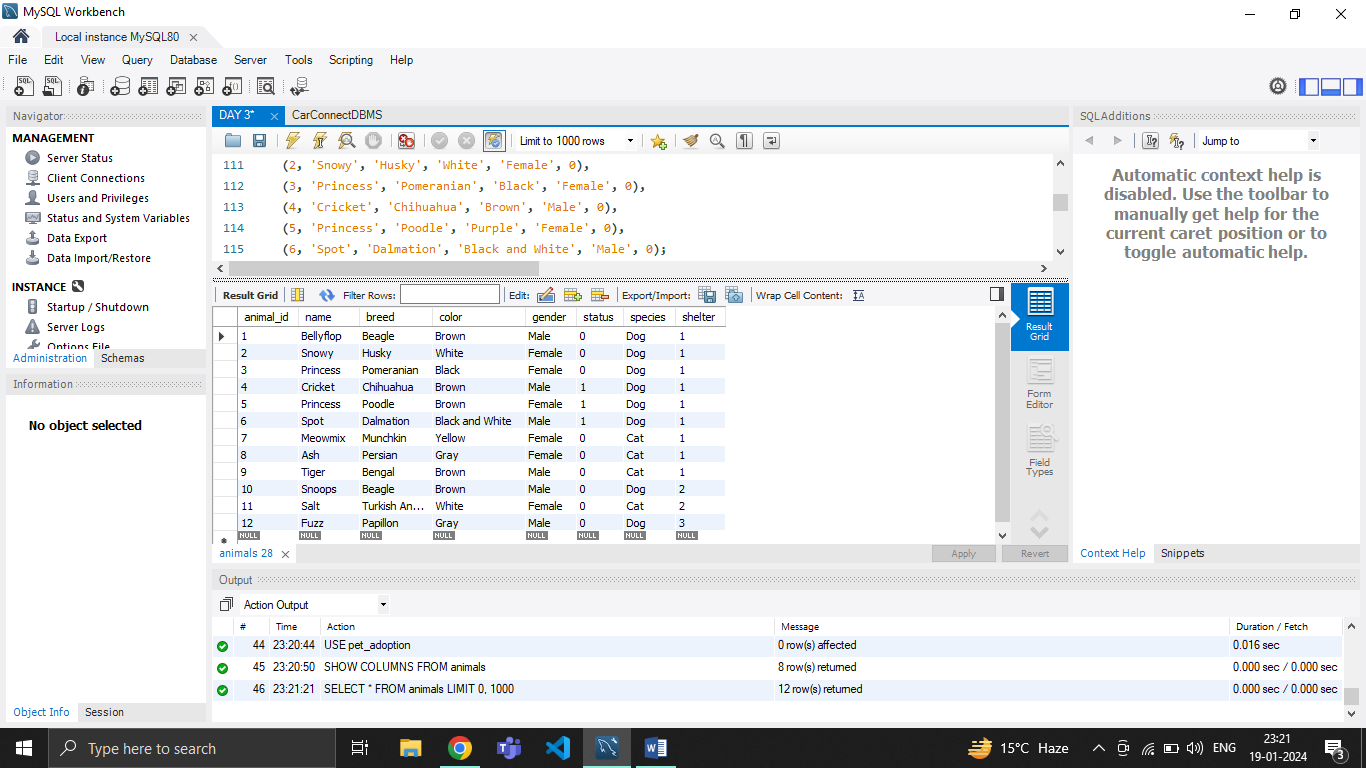
EXAMPLE:

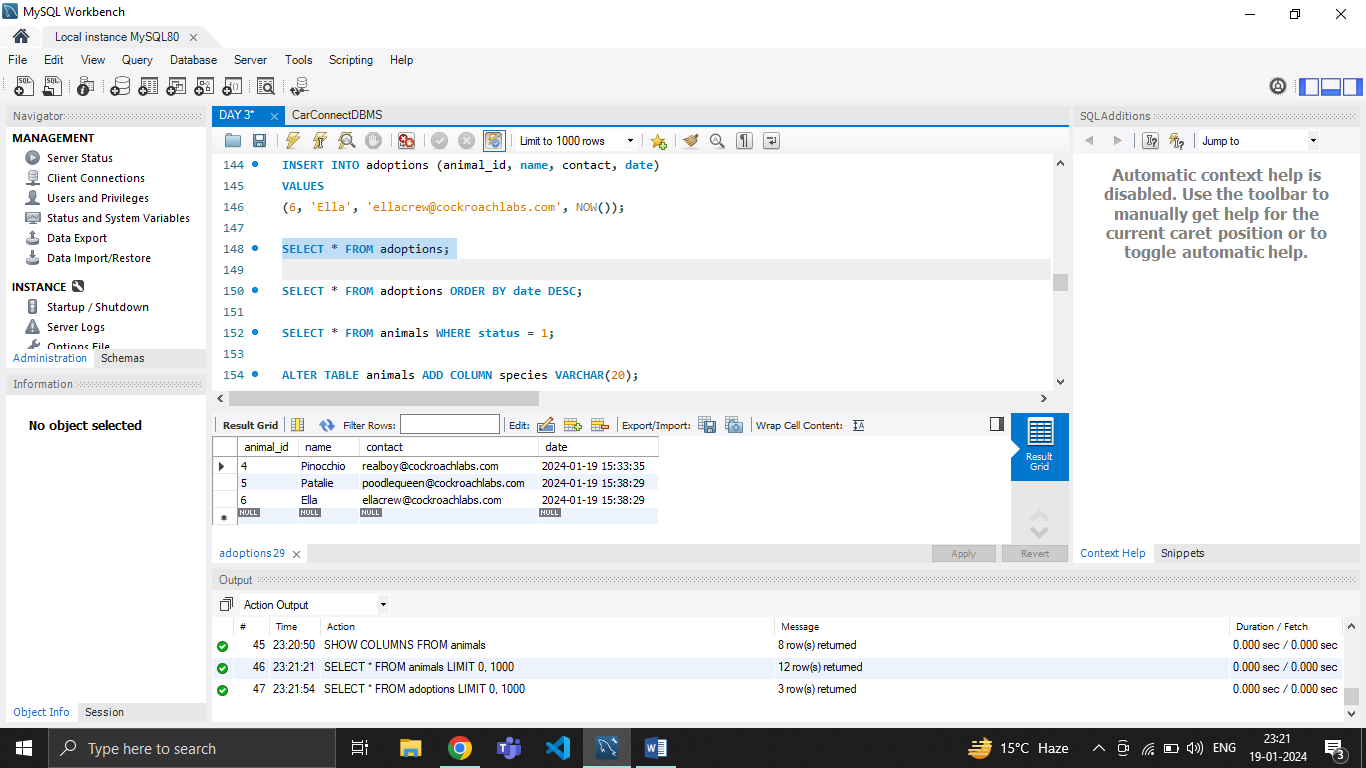
SELECT \* FROM CUSTOMERS1

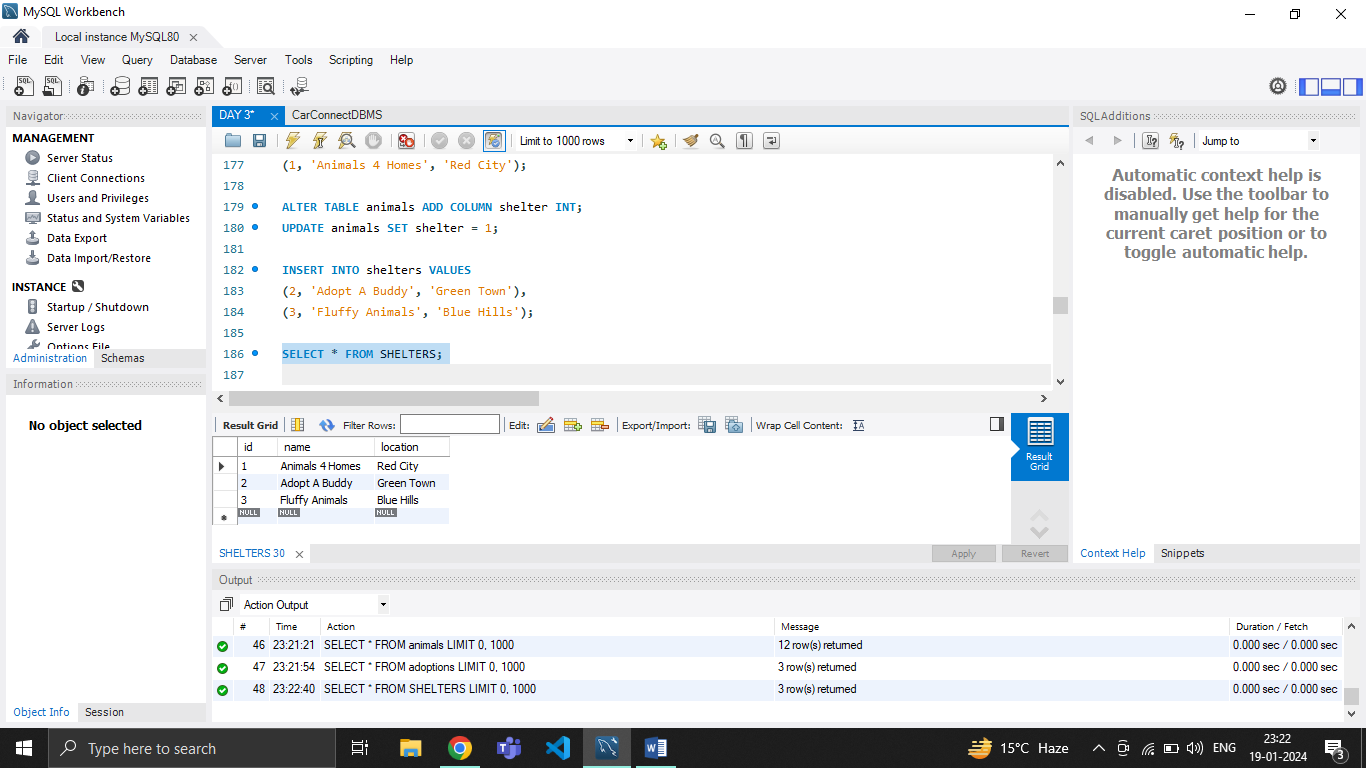
INTERSECT

SELECT \* FROM CUSTOMERS2;

**DATABASE:**







**WHERE:**

WHERE clause in SQL is used to filter rows from a table based on a specified condition.

SYNTAX:

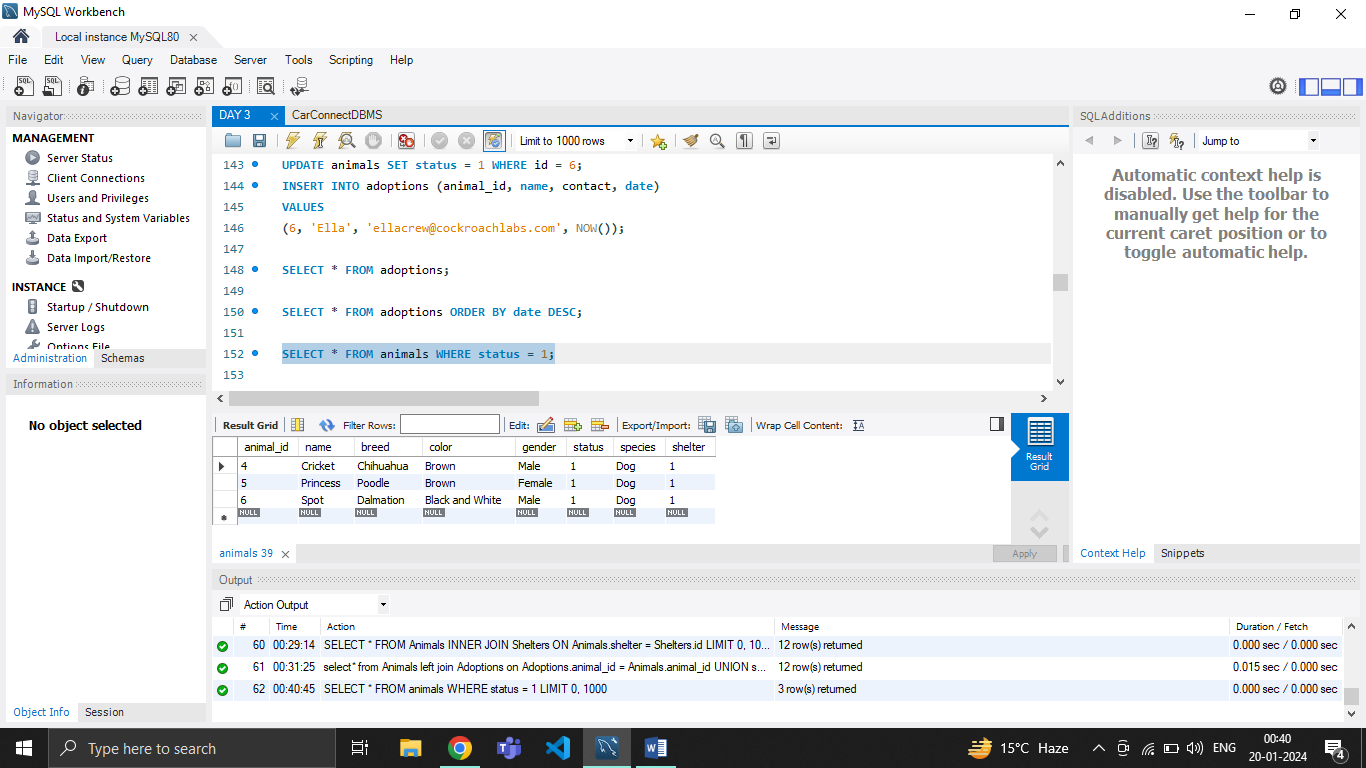
SELECT COLUMN1, COLUMN2,…

FROM TABLE\_NAME

WHERE CONDITION;

EXAMPLE:

SELECT \* FROM animals WHERE status = 1;

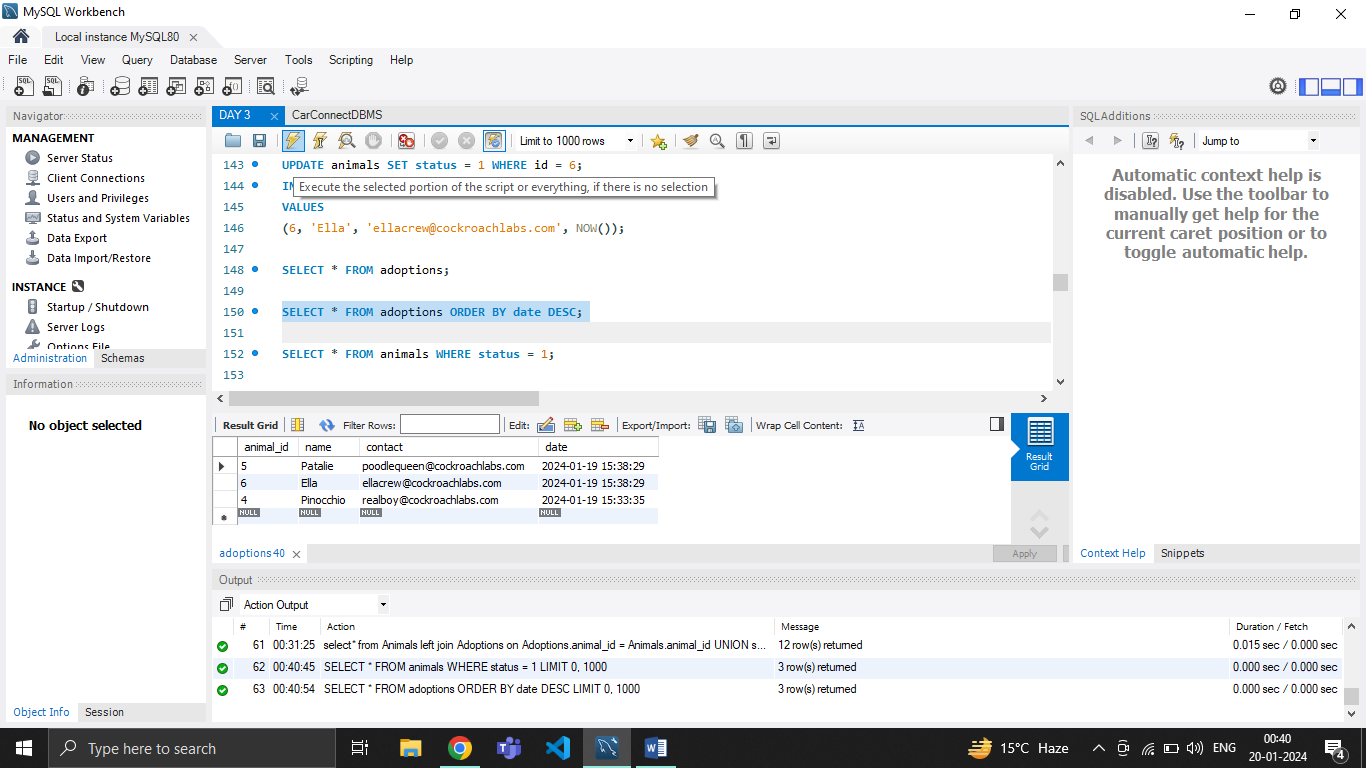
****

**ORDER BY:**

ORDER BY clause in SQL is used to sort the result set of a query based on one or more columns.

EXAMPLE:

SELECT \* FROM adoptions ORDER BY date DESC;

****

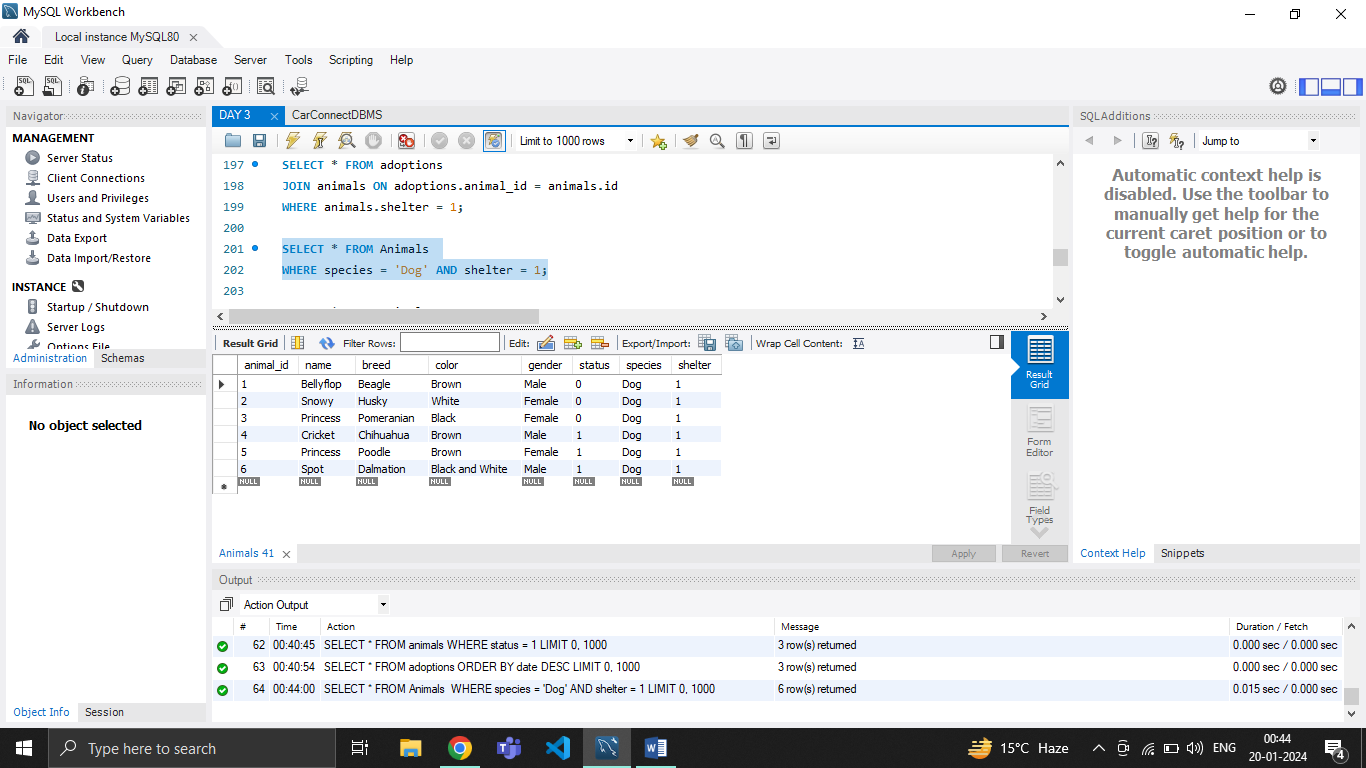
**AND:**

AND operator in SQL is used in the WHERE clause to combine multiple conditions, ensuring that all conditions must be true for a row to be included in the result set.

EXAMPLE:

SELECT \* FROM Animals

WHERE species = 'Dog' AND shelter = 1;



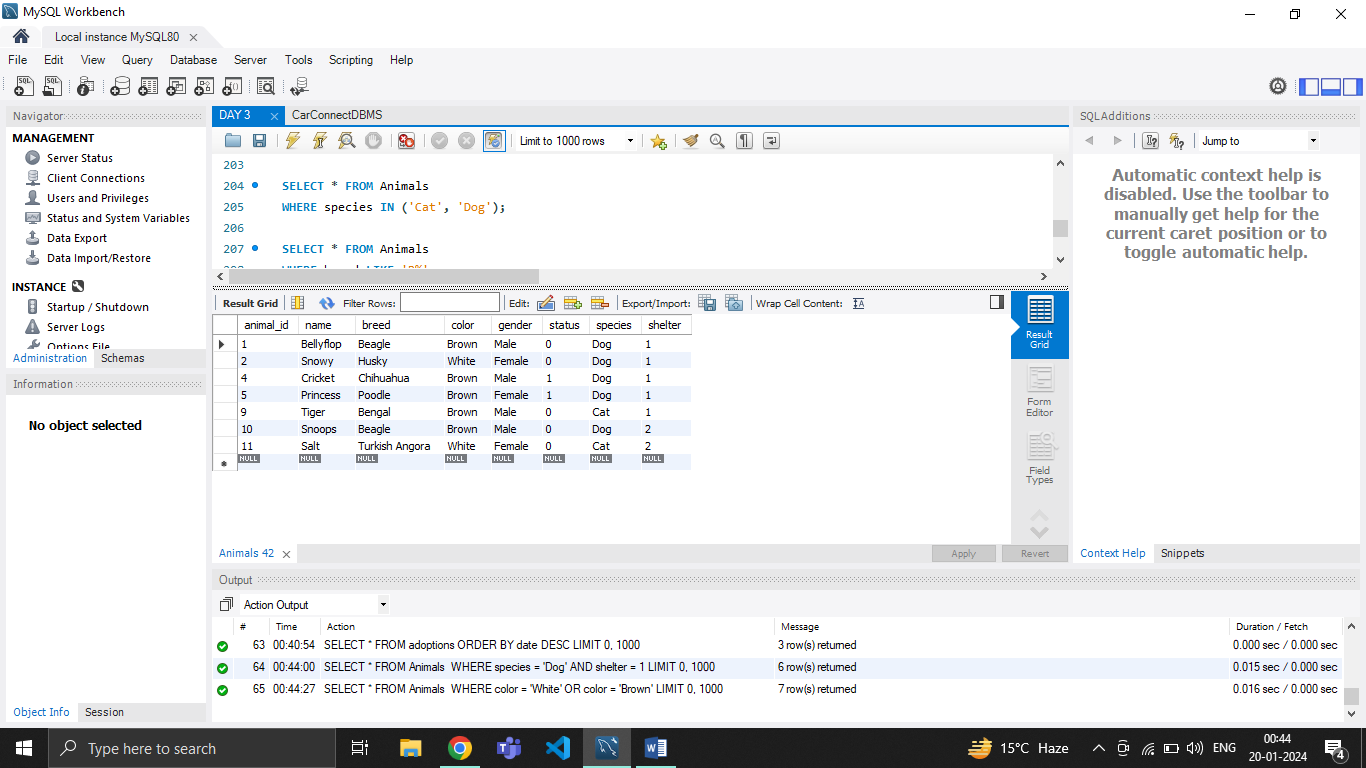
**OR:**

OR operator in SQL is used in the WHERE clause to combine multiple conditions, allowing a row to be included in the result set if at least one of the conditions is true.

EXAMPLE:

SELECT \* FROM Animals

WHERE color = 'White' OR color = 'Brown';



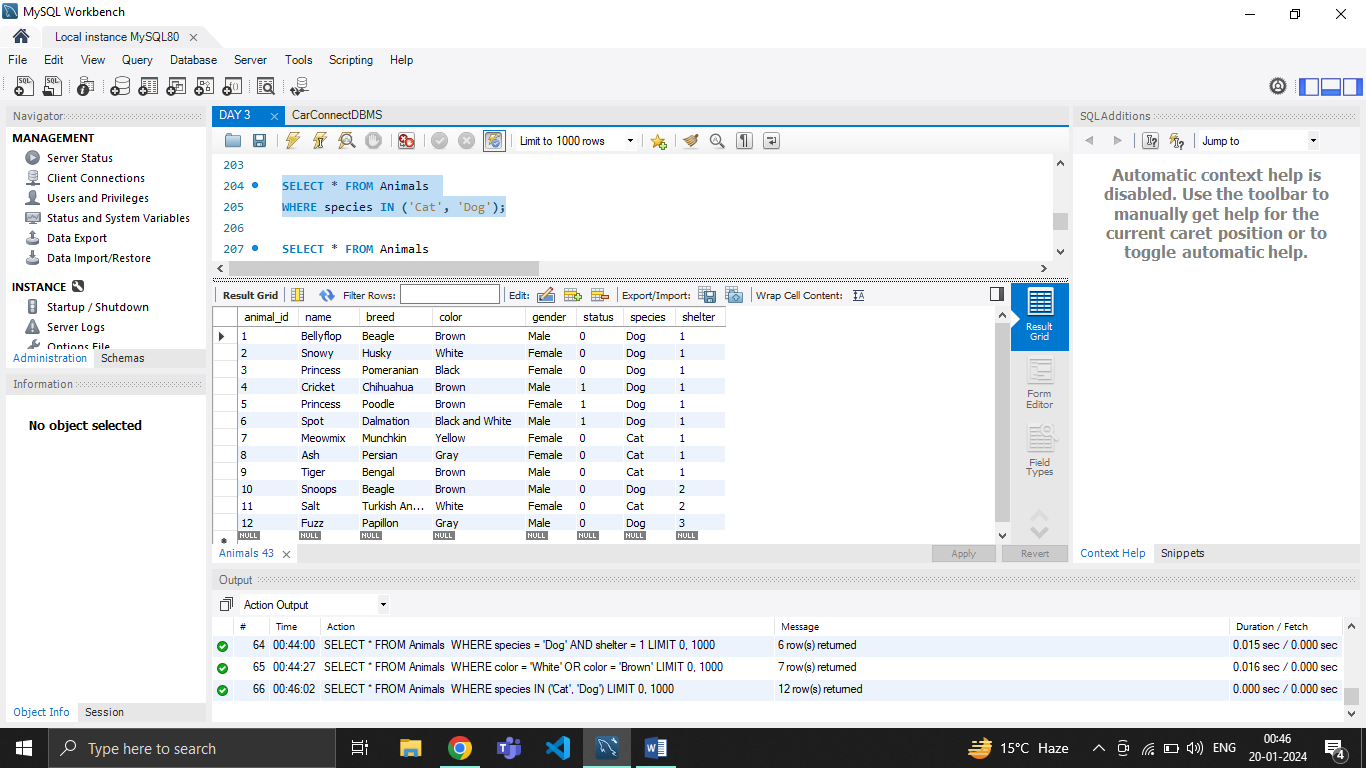
**IN:**

IN operator in SQL is used to filter rows based on a list of values.

EXAMPLE:

SELECT \* FROM Animals

WHERE species IN ('Cat', 'Dog');



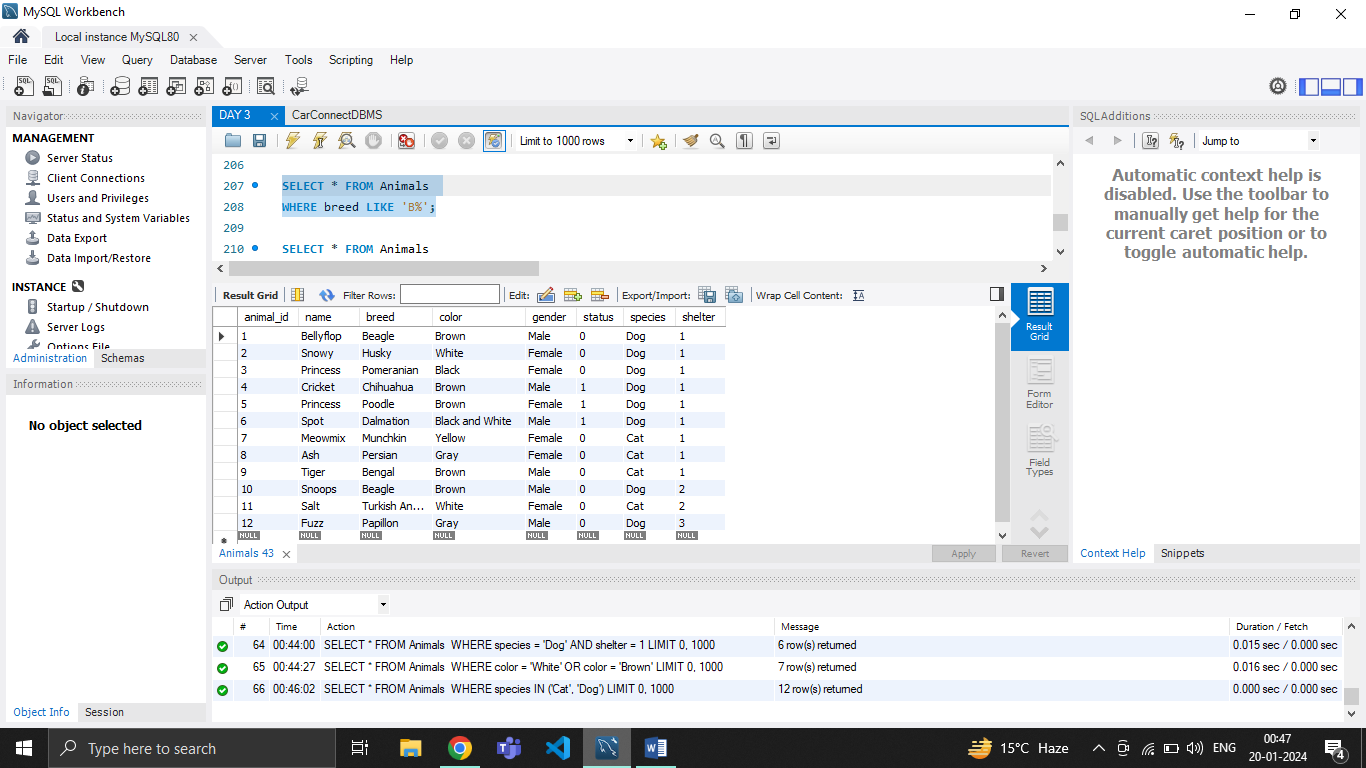
**LIKE:**

LIKE operator in SQL is used in the WHERE clause to search for a specified pattern in a column.

EXAMPLE:

SELECT \* FROM Animals

WHERE breed LIKE 'B%';



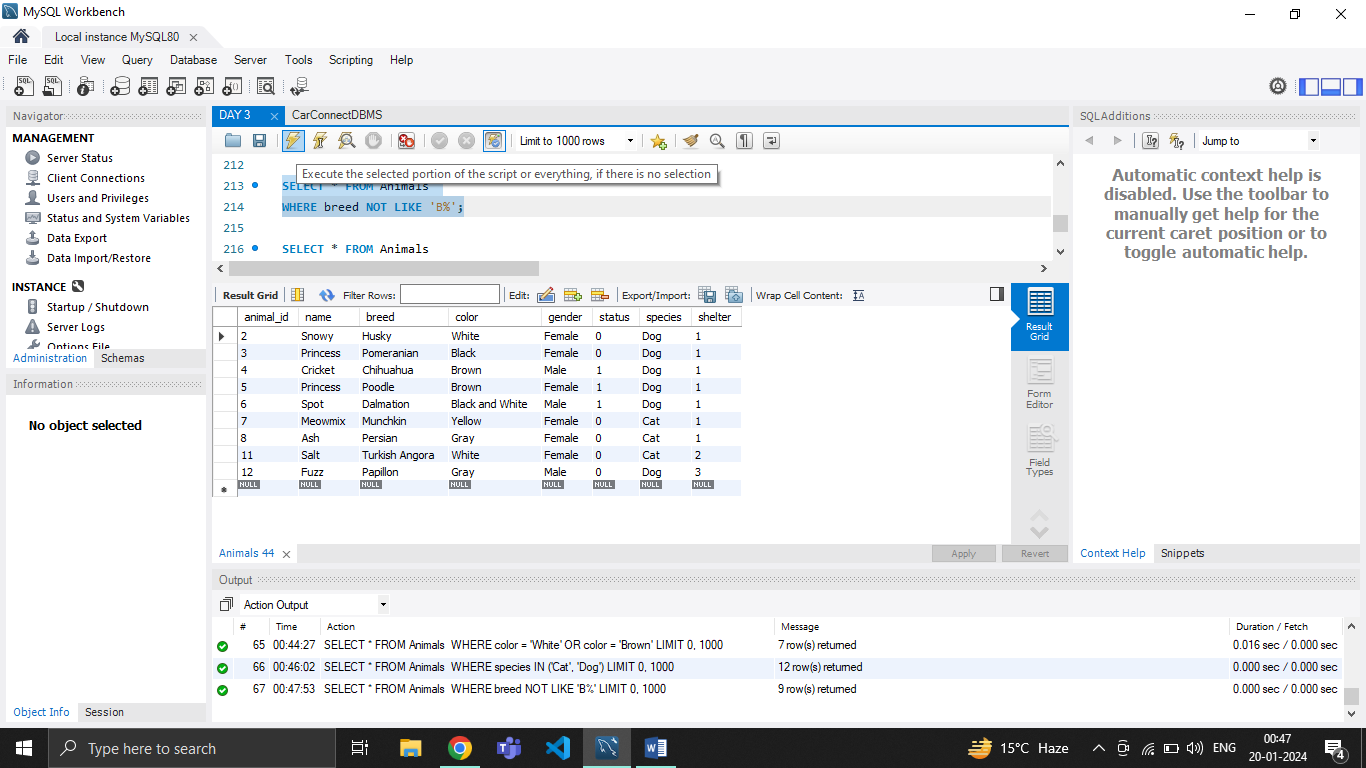
**NOT LIKE:**

NOTLIKE operator in SQL is used in the WHERE clause to filter rows based on a pattern that does not match the specified condition.

EXAMPLE:

SELECT \* FROM Animals

WHERE breed NOT LIKE 'B%';



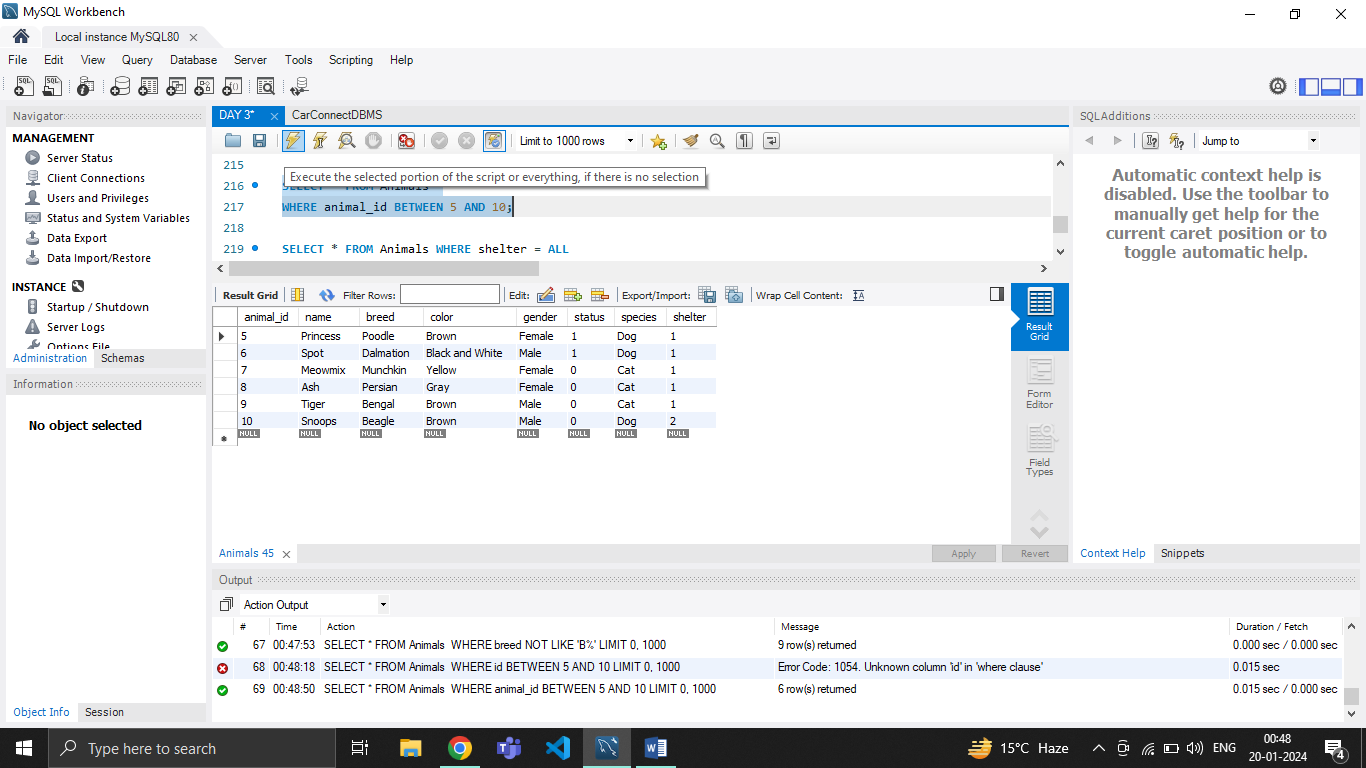
**BETWEEN:**

BETWEEN operator in SQL is used in the WHERE clause to filter rows based on a range of values.

EXAMPLE:

SELECT \* FROM Animals

WHERE id BETWEEN 5 AND 10;



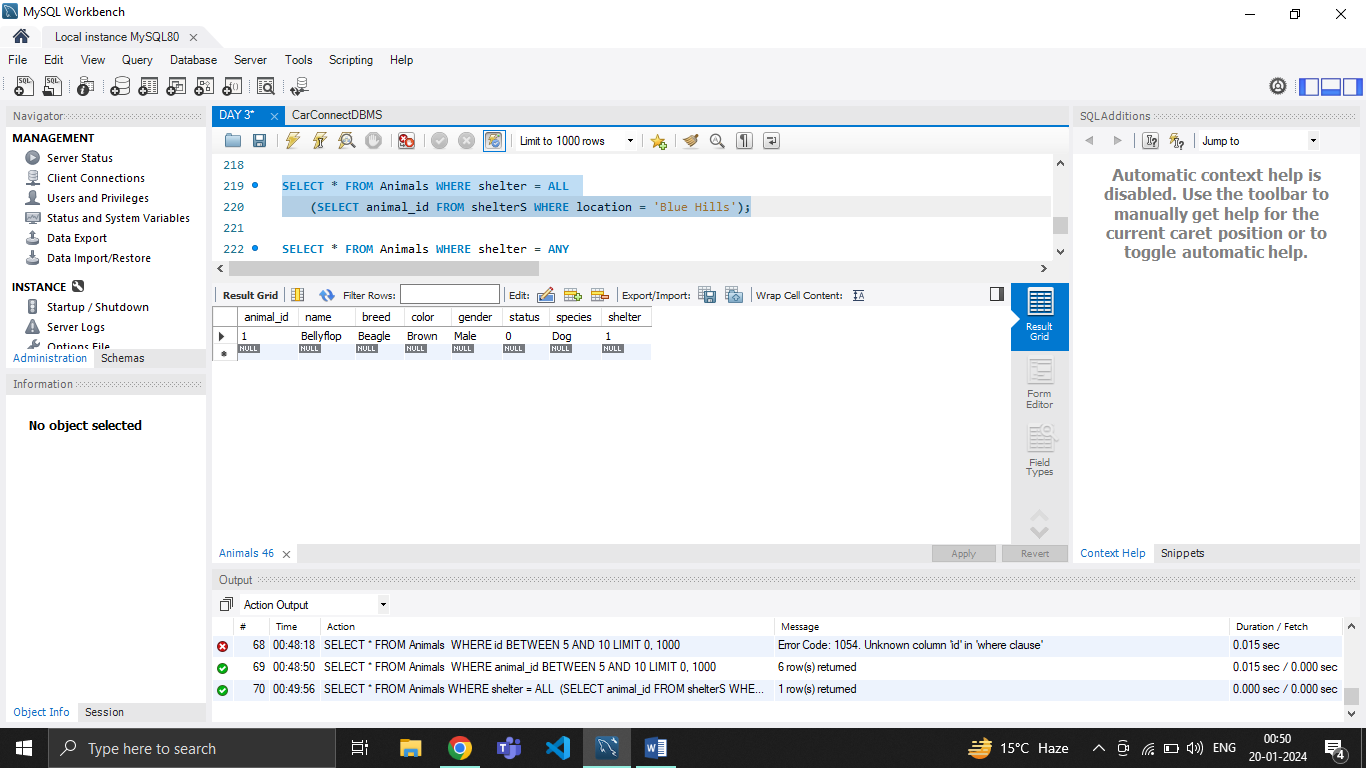
**ALL:**

ALL operator in SQL is used to compare a value to all values in a result set or a subquery. It returns true if the specified logical condition is true for all rows in the result set.

EXAMPLE:

SELECT \* FROM Animals WHERE shelter = ALL

(SELECT id FROM shelters WHERE location = 'Blue Hills');



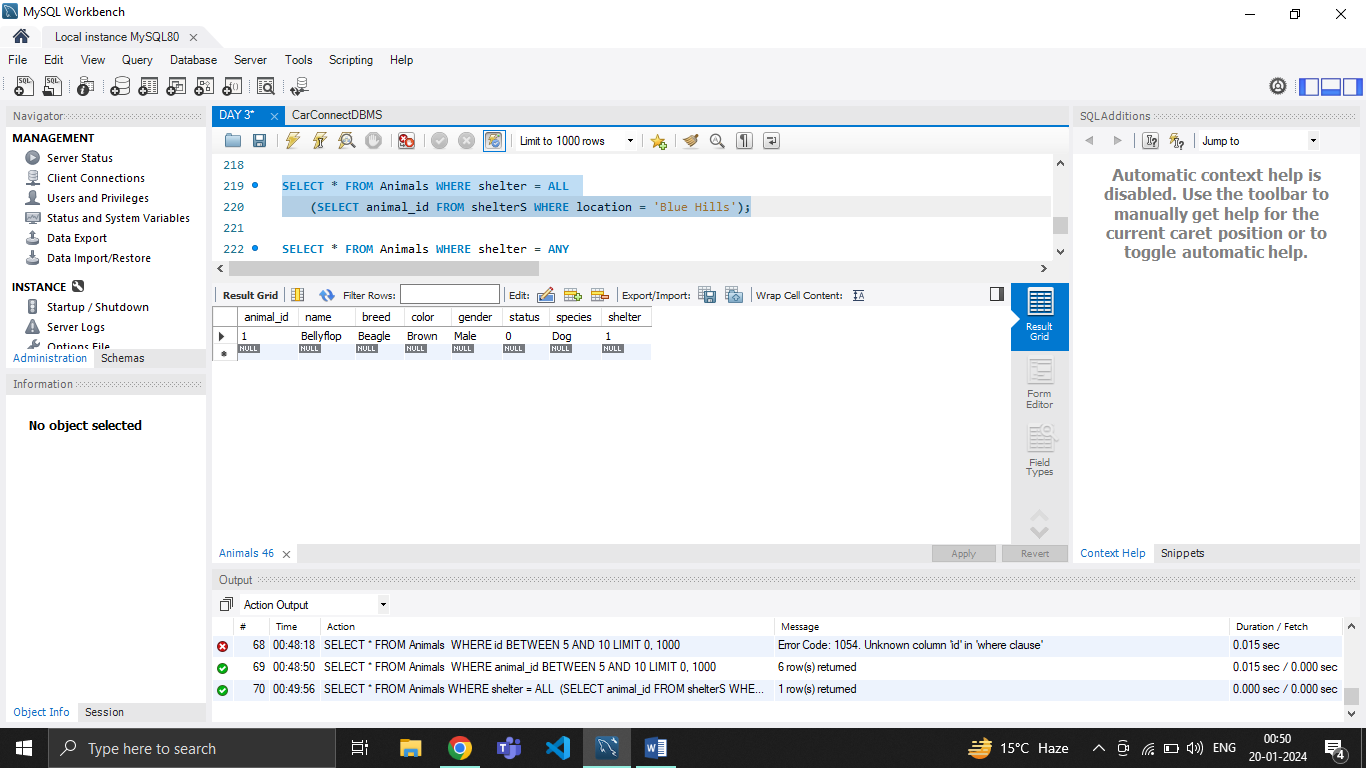
**ANY:**

ANY operator in SQL is used to compare a value to any value in a result set or a subquery. It returns true if the specified logical condition is true for at least one row in the result set.

EXAMPLE:

SELECT \* FROM Animals WHERE shelter = ANY

(SELECT id FROM shelters WHERE location = 'Blue Hills');



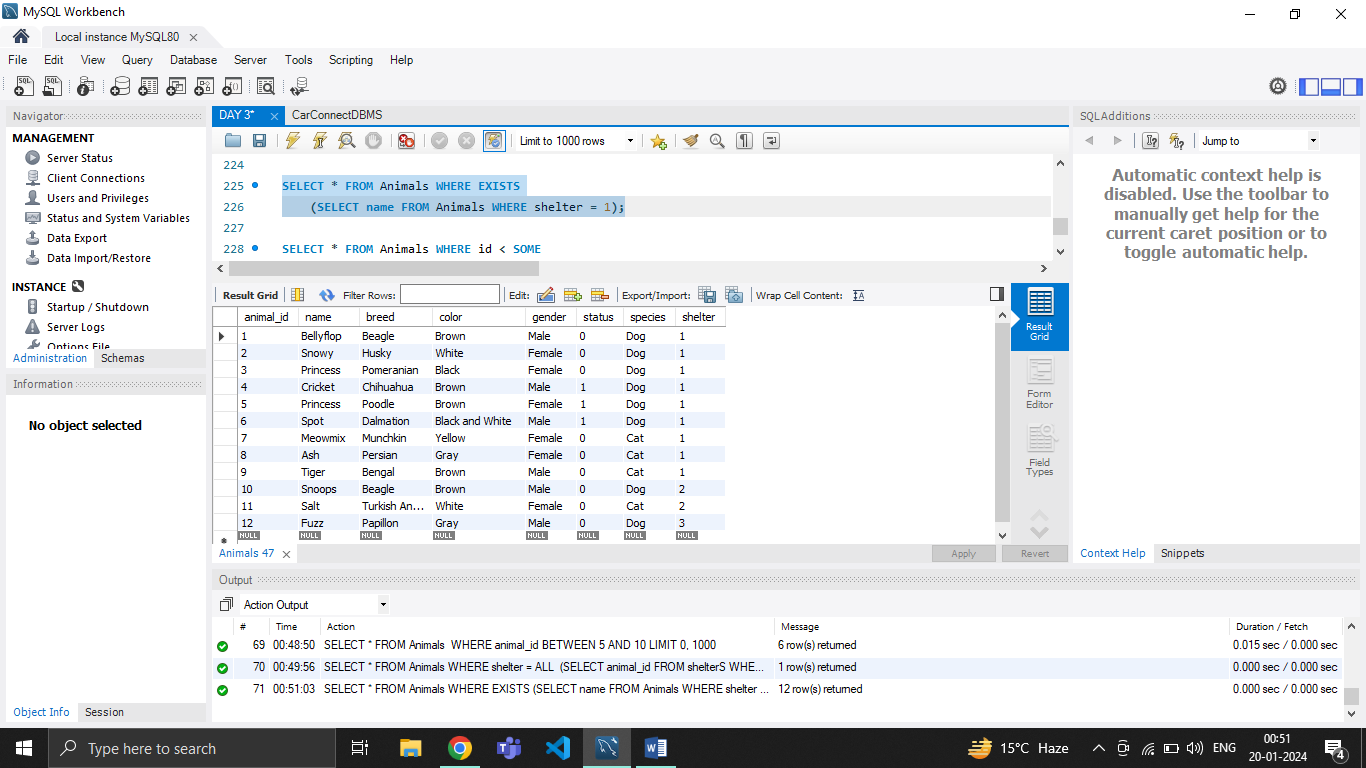
**EXISTS:**

EXISTS operator in SQL is used in the WHERE clause to check if a subquery returns any rows. It returns true if the subquery returns one or more rows and false if the subquery result set is empty.

EXAMPLE:

SELECT \* FROM Animals WHERE EXISTS

(SELECT name FROM Animals WHERE shelter = 1);



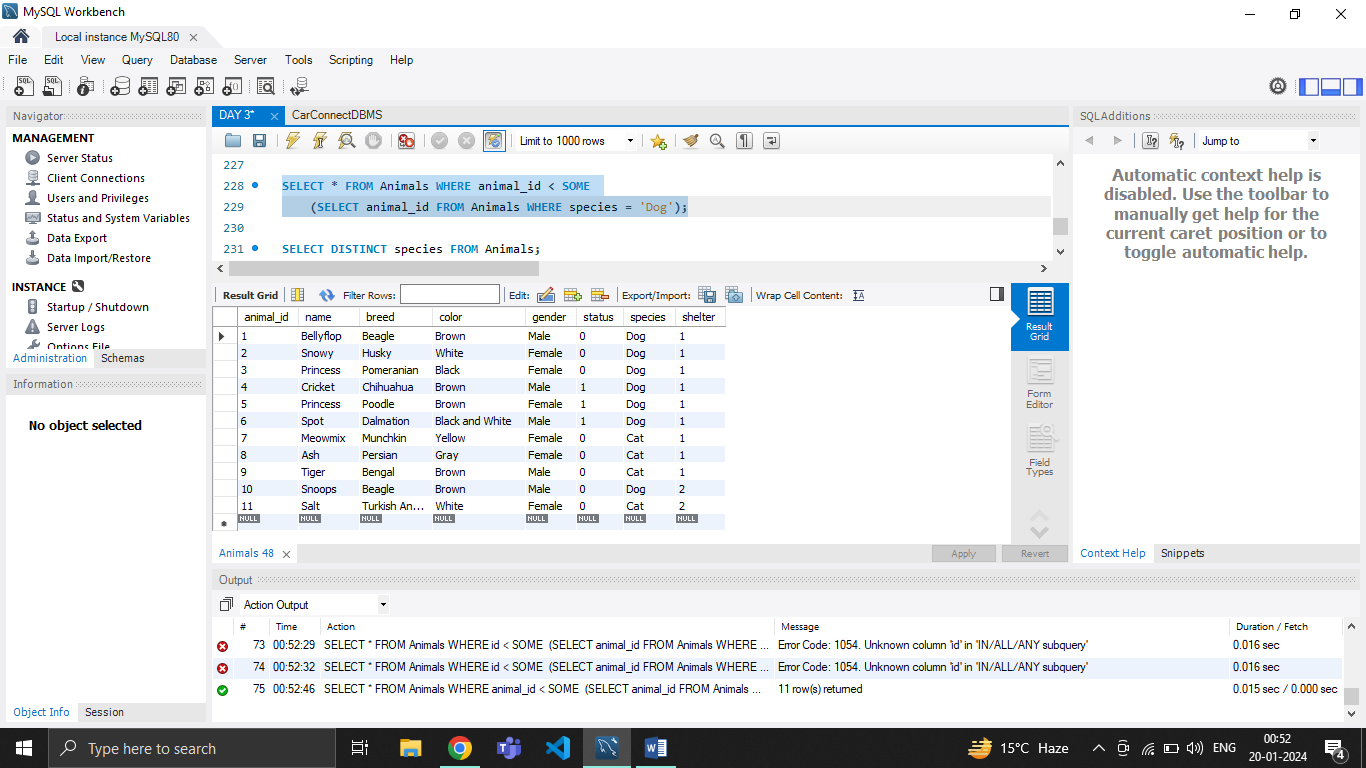
**SOME:**

SOME operator in SQL is a synonym for the ANY operator. It is used to compare a value to any value in a result set or a subquery and returns true if the specified logical condition is true for at least one row in the result set.

EXAMPLE:

SELECT \* FROM Animals WHERE id < SOME

(SELECT id FROM Animals WHERE species = 'Dog');



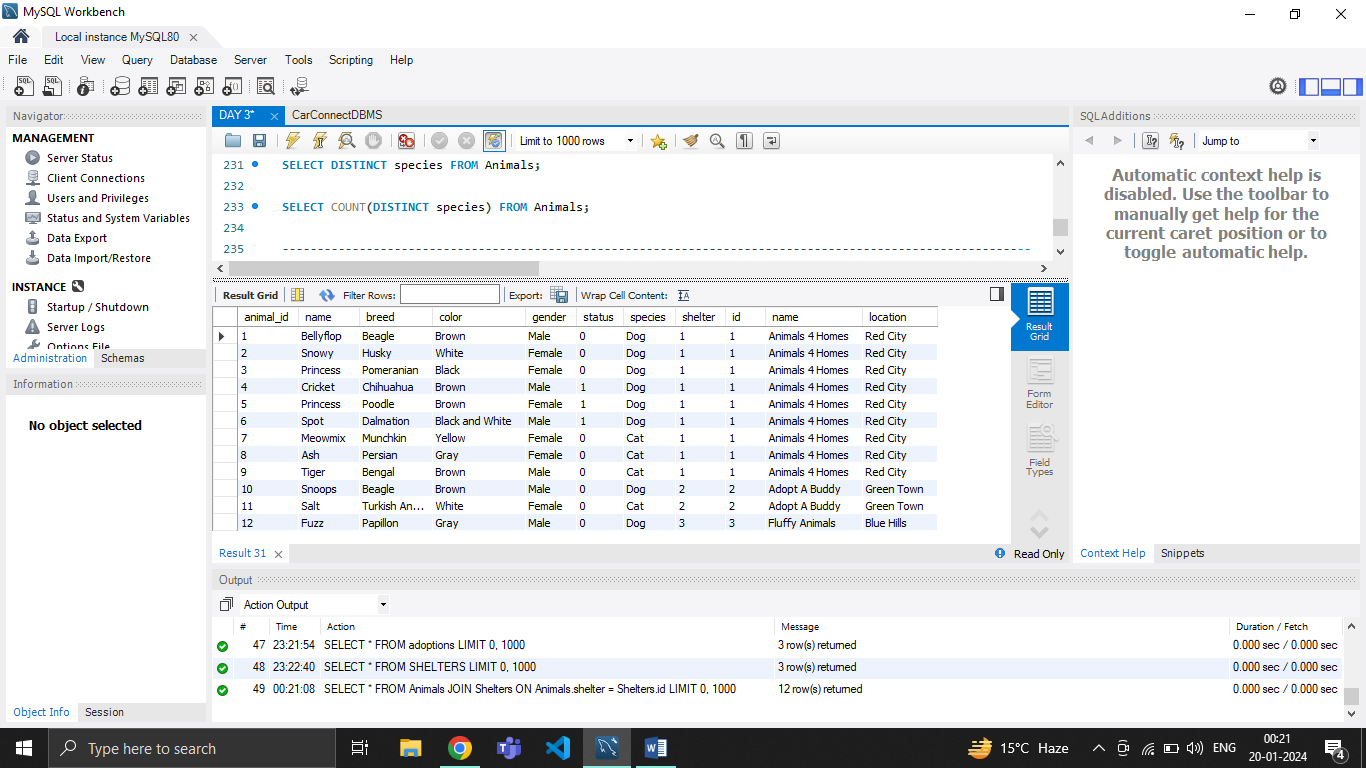
**EQUI JOIN:**

An EQUI JOIN is a type of join in SQL that combines rows from two or more tables based on a common column between them.

EXAMPLE:

SELECT \* FROM Animals

JOIN Shelters ON Animals.shelter = Shelters.id;



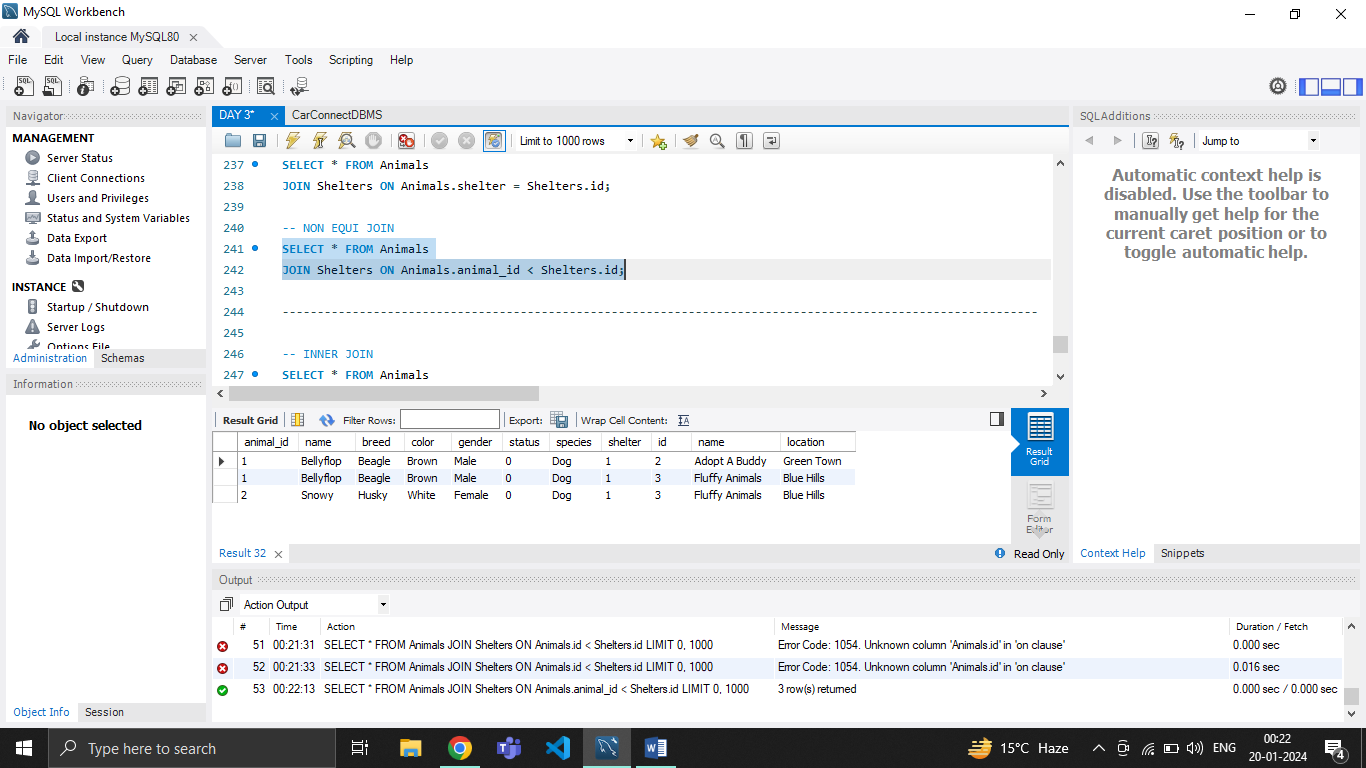
**NON EQUI JOIN:**

A NON EQUI JOIN is a type of join in SQL where the condition for combining rows from two or more tables involves a comparison other than equality.

EXAMPLE:

SELECT \* FROM Animals

JOIN Shelters ON Animals.id < Shelters.id;

****

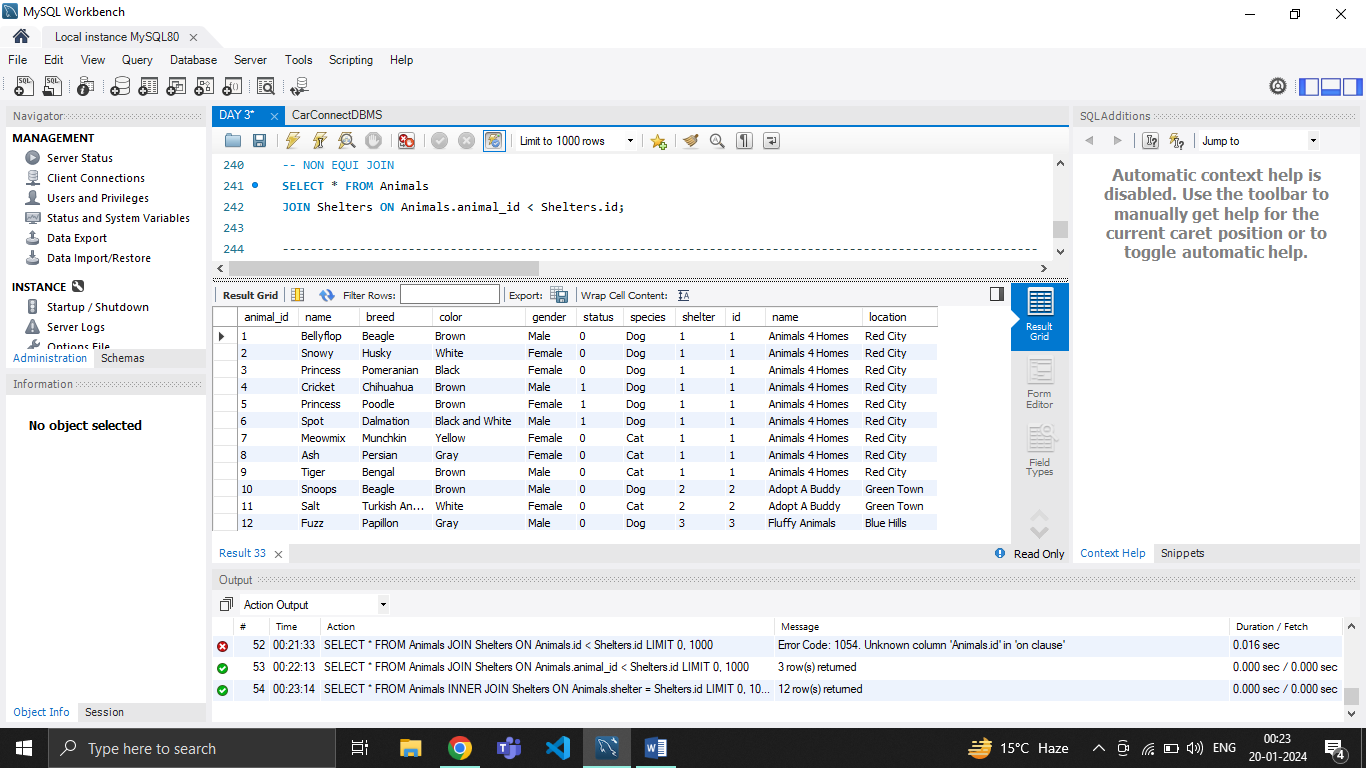
**INNER JOIN:**

INNERJOIN is a type of join in SQL that retrieves rows from two or more tables based on a specified condition and only includes rows where the condition is true. The condition typically involves matching values in columns between the tables.

EXAMPLE:

SELECT \* FROM Animals

INNER JOIN Shelters ON Animals.shelter = Shelters.id;



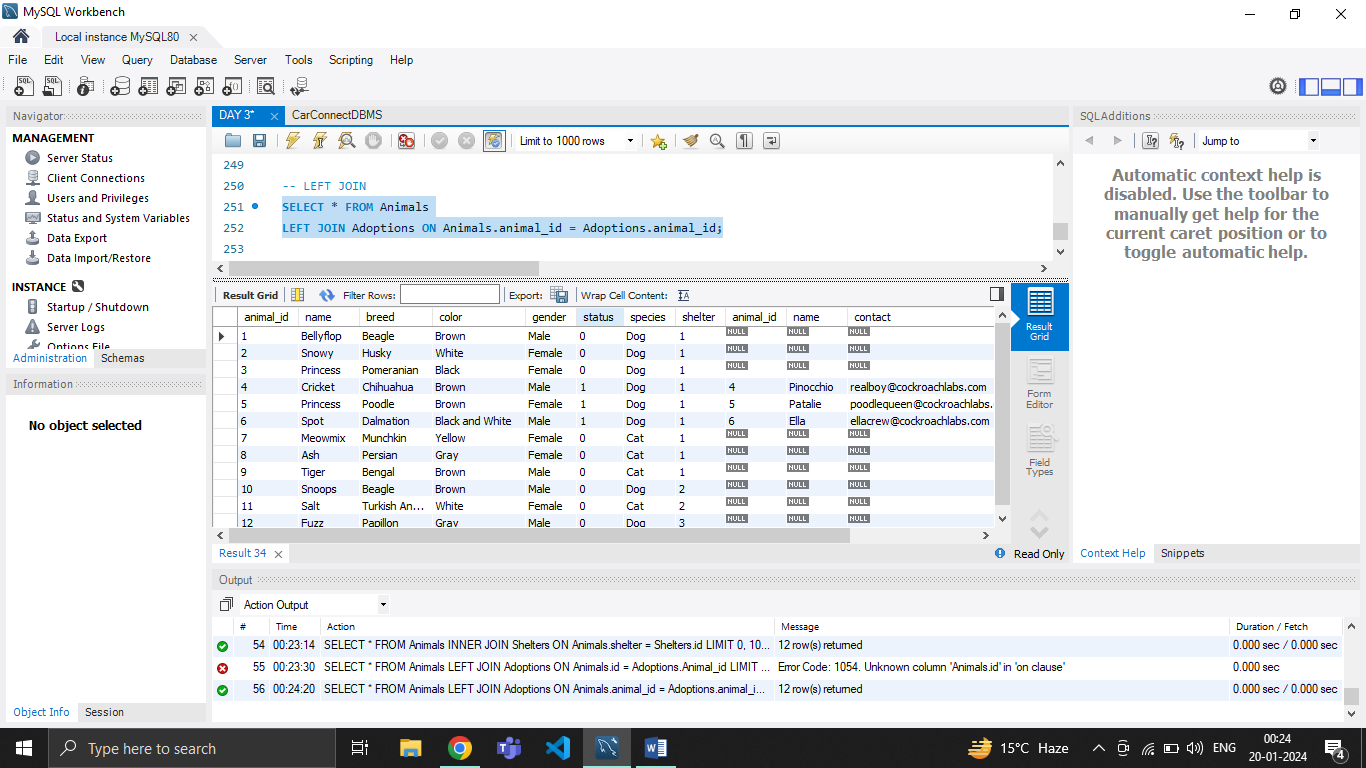
**LEFT JOIN:**

LEFT JOIN is a type of join in SQL that retrieves all rows from the left table and the matching rows from the right table. If there is no match in the right table, NULL values are returned for columns from the right table.

EXAMPLE:

SELECT \* FROM Animals

LEFT JOIN Adoptions ON Animals.id = Adoptions.animal\_id;



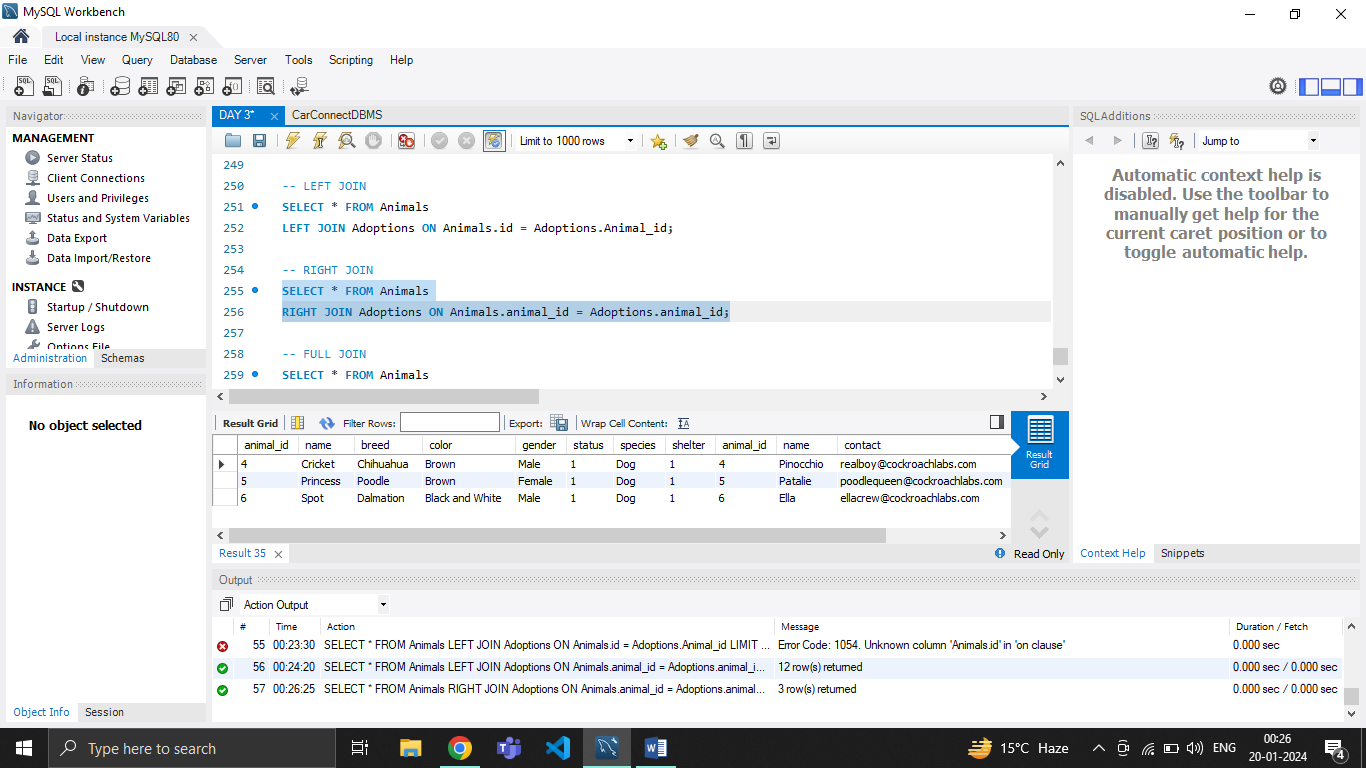
**RIGHT JOIN:**

RIGHT JOIN is a type of join in SQL that retrieves all rows from the right table and the matching rows from the left table. If there is no match in the left table, NULL values are returned for columns from the left table.

EXAMPLE:

SELECT \* FROM Animals

RIGHT JOIN Adoptions ON Animals.id = Adoptions.animal\_id;



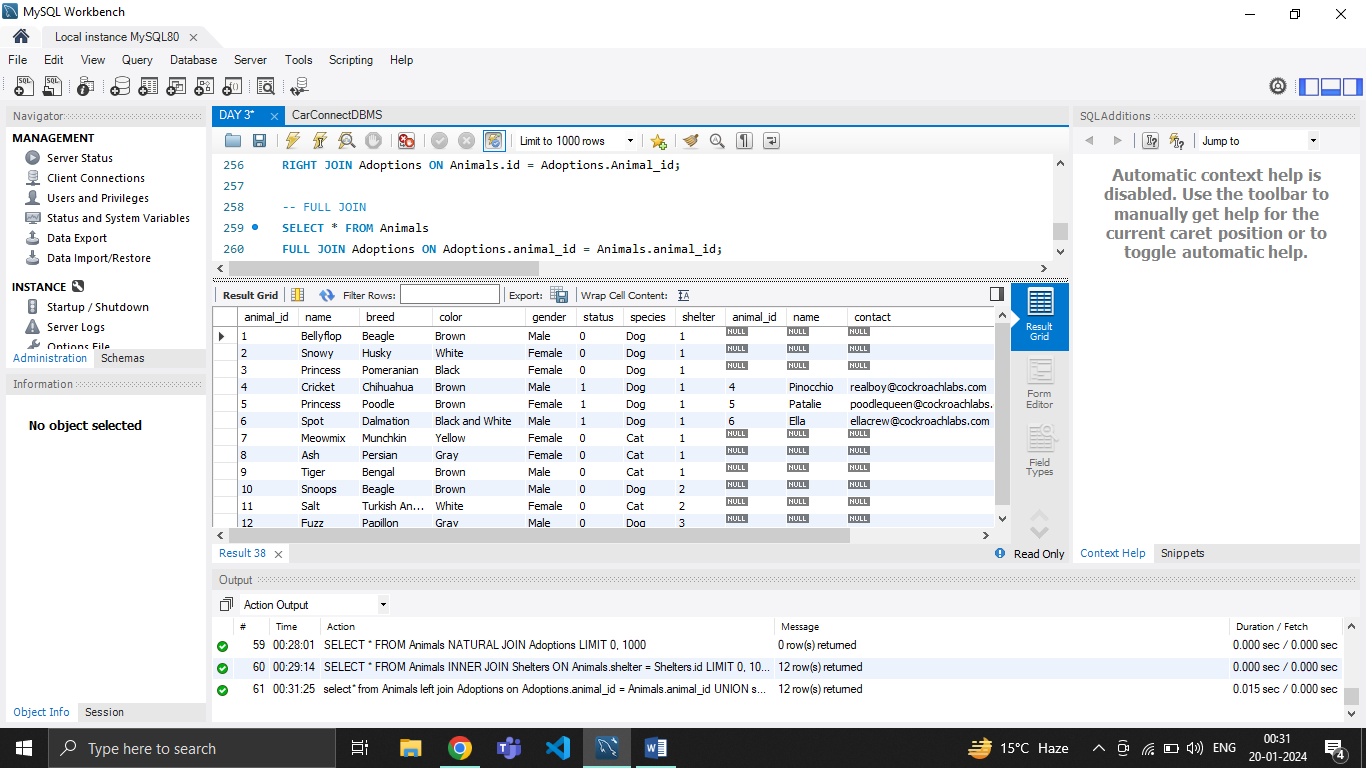
**FULL JOIN:**

FULL JOIN is a type of join in SQL that combines the results of both **LEFT JOIN** and **RIGHT JOIN**. It returns all rows from both tables, matching them where possible, and filling in with NULLs where there is no match.

EXAMPLE:

SELECT \* FROM Animals

FULL JOIN Adoptions ON Adoptions.animal\_id = Animals.id;



**NATURAL JOIN:**

NATURALJOIN in SQL is a type of join that automatically joins tables based on columns with matching names.

EXAMPLE:

SELECT \* FROM Animals

NATURAL JOIN Shelters;

