

Asmita Porwal
Batch-1
Day-3
19/1/2023
Data engineering

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.

1. Select all columns

```
mysql> select * from customers;
```

customer_id	name	email	password
1	asmita	asmita.porwal@gmail.com	12345
2	Rohan	r@gmail.com	123
3	paritosh porwal	p@gmail.com	1234
4	Ram	ram@yahoo.com	123456
5	Krishna	k@yahoo.com	1234567
6	Sita	s@gmail.com	12345
7	Mamta	m@gmail.com	12345
8	Naresh	n@gmail.com	1234

```
8 rows in set (0.04 sec)
```

2. Select distinct statement

```
mysql> select distinct name from customers;
```

name
asmita
Rohan
paritosh porwal
Ram
Krishna
Sita
Mamta
Naresh

```
8 rows in set (0.02 sec)
```

3. Having clause

```
mysql> select * from payments;
```

payment_id	student_id	amount	payment_date
101	7	200	2019-02-01
102	7	200	2019-04-01
103	1	400	2020-03-01
106	10	400	2019-11-19
107	4	400	2021-09-07
108	6	400	2021-11-11
109	5	400	2021-04-19
110	9	400	2022-09-02
111	8	400	2023-02-14
112	3	400	2023-01-02
113	1	100	2023-12-26
114	3	400	2023-12-26

```
12 rows in set (0.03 sec)
```



```
mysql> select sum(amount) as total_amount, student_id from payments group by student_id having sum(amount)>600;
```

total_amount	student_id
800	3

```
1 row in set (0.02 sec)
```

4.Group by clause

```
mysql> select sum(amount) as total_amount, student_id from payments group by student_id;
```

total_amount	student_id
500	1
800	3
400	4
400	5
400	6
400	7
400	8
400	9
400	10

```
9 rows in set (0.00 sec)
```

5.Sql Transactions

```
mysql> create table customers(  
-> ID int not null,  
-> Name varchar(20) not null,  
-> Age int not null,  
-> Address char(25),  
-> salary decimal(18,2),  
-> primary key(ID)  
-> );  
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> insert into customers values  
-> (1,"Asmita",20,"Gwalior",600000.00),  
-> (2,"Amita",30,"Gwalior",200000.00),  
-> (3,"Smita",22,"Guna",10000.00),  
-> (4,"Ram",10,"Nagpur",700000.00),  
-> (5,"Sita",31,"Chennai",100000.00);  
Query OK, 5 rows affected (0.03 sec)  
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> Delete from customers where age=20;  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from customers;
```

ID	Name	Age	Address	salary
2	Amita	30	Gwalior	200000.00
3	Smita	22	Guna	10000.00
4	Ram	10	Nagpur	700000.00
5	Sita	31	Chennai	100000.00

```
4 rows in set (0.00 sec)
```

6.Commit

to save the changes

```
mysql> start transaction; delete from customers where age=30; Commit;  
Query OK, 0 rows affected (0.00 sec)
```

```
Query OK, 1 row affected (0.01 sec)
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from customers;
```

ID	Name	Age	Address	salary
3	Smita	22	Guna	10000.00
5	Sita	31	Chennai	100000.00

```
2 rows in set (0.00 sec)
```

7.Rollback

to roll back the changes

```
mysql> start transaction;Delete from customers where age=30;ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> select * from customers;
+----+-----+-----+-----+-----+
| ID | Name  | Age  | Address | salary |
+----+-----+-----+-----+-----+
| 2  | Amita | 30   | Gwalior | 200000.00 |
| 3  | Smita | 22   | Guna    | 10000.00  |
| 5  | Sita  | 31   | Chennai | 100000.00 |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

8.Savepoint

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

```
mysql> start transaction; savepoint sp1; update customers set age=33 where id=5; savepoint sp2; update customers set age=34 where i
d=5; savepoint sp3; update customers set age=35 where id=5;ROLLBACK TO sp2;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

Query OK, 0 rows affected (0.00 sec)

mysql> select * from customers;
+----+-----+-----+-----+-----+
| ID | Name  | Age  | Address | salary |
+----+-----+-----+-----+-----+
| 5  | Sita  | 33   | Chennai | 100000.00 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

9.Release Savepoint

```
mysql> start transaction; savepoint sp1; update customers set age=33 where id=5; savepoint sp2; update customers set age=34 where i
d=5; savepoint sp3; update customers set age=35 where id=5; release savepoint sp2;commit;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Query OK, 0 rows affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

10.Set Transaction

Place a name on a transaction

```
mysql> SET TRANSACTION READ WRITE;UPDATE customers SET age= 20 WHERE id = 5; commit;
Query OK, 0 rows affected (0.02 sec)

Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET TRANSACTION READ Only; select * from customers; commit;
Query OK, 0 rows affected (0.00 sec)

+----+-----+-----+-----+-----+
| ID | Name | Age | Address | salary |
+----+-----+-----+-----+-----+
| 5 | Sita | 20 | Chennai | 100000.00 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

11.Set operators

11.Union

```
mysql> select * from teacher union select * from teachers_1;
```

teacher_id	first_name	last_name	email
1	asmita	porwal	asmita@gmail.com
2	Janvi	Singh	janvi@gmail.com
3	ritu	shrama	r@gmail.com
4	Tushar	Khurana	tushar@gmail.com
5	Shilpa	Singh	shilpa@gmail.com
6	Naresh	Gupta	naresh@gmail.com
7	Mamta	Gupta	mamta@gmail.com
8	Paritosh	Porwal	paritosh@gmail.com
9	Suman	Sharma	suman@gmail.com
10	Ramesh	Sood	ramesh@gmail.com
11	Amit	Kumar	amit.kumar@gmail.com
12	Preeti	Verma	preeti.verma@gmail.com
13	Rahul	Sharma	rahul.sharma@gmail.com
14	Neha	Singh	neha.singh@gmail.com
15	Vikas	Malhotra	vikas.malhotra@gmail.com

```
15 rows in set (0.01 sec)
```

12. Union all

```
mysql> select * from teacher union all select * from teachers_1;
```

teacher_id	first_name	last_name	email
1	asmita	porwal	asmita@gmail.com
2	Janvi	Singh	janvi@gmail.com
3	ritu	shrama	r@gmail.com
4	Tushar	Khurana	tushar@gmail.com
5	Shilpa	Singh	shilpa@gmail.com
6	Naresh	Gupta	naresh@gmail.com
7	Mamta	Gupta	mamta@gmail.com
8	Paritosh	Porwal	paritosh@gmail.com
9	Suman	Sharma	suman@gmail.com
10	Ramesh	Sood	ramesh@gmail.com
11	Amit	Kumar	amit.kumar@gmail.com
12	Preeti	Verma	preeti.verma@gmail.com
13	Rahul	Sharma	rahul.sharma@gmail.com
14	Neha	Singh	neha.singh@gmail.com
15	Vikas	Malhotra	vikas.malhotra@gmail.com

```
15 rows in set (0.00 sec)
```

13. Intersect


```
mysql> select * from teacher intersect select * from teachers_1;
+-----+-----+-----+-----+
| teacher_id | first_name | last_name | email          |
+-----+-----+-----+-----+
|          9 | Suman      | Sharma    | suman@gmail.com |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

14. Distinct Clause with Order By

```
mysql> select distinct Name, Age from customers order by Age;
+-----+-----+
| Name   | Age  |
+-----+-----+
| Ram    | 10   |
| Asmita | 20   |
| Sita   | 20   |
| Smita  | 22   |
| Amita  | 30   |
+-----+-----+
5 rows in set (0.01 sec)
```

15. New database

```
CREATE DATABASE pet_adoption;
```

16. use database

```
USE pet_adoption;
```

17. 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));
```

18.All tables

```
SHOW TABLES;
```

```
Database changed  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_pet_adoption |  
+-----+  
| adoptions               |  
| animals                 |  
| old_animals             |  
| shelters                |  
+-----+  
4 rows in set (0.03 sec)
```

19.show all columns

```
desc adoptions;
```

```
desc animals;
```

```
mysql> desc adoptions;
```

Field	Type	Null	Key	Default	Extra
animal_id	int	NO		NULL	
name	varchar(100)	YES		NULL	
contact	varchar(10)	YES		NULL	
date	timestamp	YES		NULL	

```
4 rows in set (0.01 sec)
```

```
mysql> desc animals;
```

Field	Type	Null	Key	Default	Extra
animal_id	int	NO		NULL	
name	varchar(100)	YES		NULL	
breed	varchar(100)	YES		NULL	
color	varchar(100)	YES		NULL	
gender	varchar(100)	YES		NULL	
status	int	YES		NULL	
species	varchar(100)	YES		NULL	
shelter	int	YES	MUL	NULL	

```
8 rows in set (0.00 sec)
```

20.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);
```

21.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;
```

```
mysql> SELECT * FROM animals;
```

animal_id	name	breed	color	gender	status	species	shelter
1	Bellyflop	Beagle	Brown	Male	0	Dog	1
2	Snowy	Husky	Brown	Female	1	Dog	1
3	Princess	Pomeranian	Brown	Female	0	Dog	1
4	Cricket	Chihuahua	Brown	Male	0	Dog	1
5	Princess	Poodle	Brown	Female	0	Dog	1
6	Spot	Dalmation	Black and White	Male	0	Dog	1
7	Meowmix	Munchkin	Yellow	Female	0	Cat	1
8	Ash	Persian	Gray	Female	0	Cat	1
9	Tiger	Bengal	Brown	Male	0	Cat	1
10	Snoops	Beagle	Brown	Male	0	Dog	2
11	Salt	Turkish Angora	White	Female	0	Cat	2
12	Fuzz	Papillon	Gray	Male	0	Dog	3

```
12 rows in set (0.01 sec)
```

```
mysql> SELECT breed FROM animals;
```

breed
Beagle
Husky
Pomeranian
Chihuahua
Poodle
Dalmation
Munchkin
Persian
Bengal
Beagle
Turkish Angora
Papillon

```
12 rows in set (0.00 sec)
```

```
mysql> SELECT name FROM animals WHERE gender = 'Female';
```

name
Snowy
Princess
Princess
Meowmix
Ash
Salt

```
6 rows in set (0.00 sec)
```

```
mysql> SELECT animal_id FROM animals WHERE status = 0;
```

animal_id
1
3
4
5
6
7
8
9
10
11
12

```
11 rows in set (0.00 sec)
```

22.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;
```

23.Order by

```
SELECT * FROM adoptions ORDER BY date DESC;
```

```
mysql> SELECT * FROM adoptions ORDER BY date DESC;
```

animal_id	name	contact	date
8	Ella	9009787866	2024-01-19 15:27:16
7	Pinocchio	9826976856	2024-01-19 15:26:30

```
2 rows in set (0.00 sec)
```

24.Alter

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

25.Turn off safe update

```
SET sql_safe_updates = FALSE;
UPDATE animals SET shelter = 1;
```

26.Indexing

```
CREATE INDEX animal_shelter ON animals (shelter);
```

27.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;
```

```
mysql> SELECT * FROM animals JOIN shelters ON animals.shelter = shelters.id;
```

animal_id	name	breed	color	gender	status	species	shelter	id	name	location
1	Bellyflop	Beagle	Brown	Male	0	Dog	1	1	Animals 4 Homes	Red City
2	Snowy	Husky	Brown	Female	1	Dog	1	1	Animals 4 Homes	Red City
3	Princess	Pomeranian	Brown	Female	0	Dog	1	1	Animals 4 Homes	Red City
4	Cricket	Chihuahua	Brown	Male	0	Dog	1	1	Animals 4 Homes	Red City
5	Princess	Poodle	Brown	Female	0	Dog	1	1	Animals 4 Homes	Red City
6	Spot	Dalmation	Black and White	Male	0	Dog	1	1	Animals 4 Homes	Red City
7	Meowmix	Munchkin	Yellow	Female	0	Cat	1	1	Animals 4 Homes	Red City
8	Ash	Persian	Gray	Female	0	Cat	1	1	Animals 4 Homes	Red City
9	Tiger	Bengal	Brown	Male	0	Cat	1	1	Animals 4 Homes	Red City
10	Snoops	Beagle	Brown	Male	0	Dog	2	2	Adopt A Buddy	Green Town
11	Salt	Turkish Angora	White	Female	0	Cat	2	2	Adopt A Buddy	Green Town
12	Fuzz	Papillon	Gray	Male	0	Dog	3	3	Fluffy Animals	Blue Hills

```
12 rows in set (0.01 sec)
```

```
mysql> SELECT * FROM adoptions JOIN animals ON adoptions.animal_id = animals.animal_id WHERE animals.shelter = 1;
```

animal_id	name	contact	date	animal_id	name	breed	color	gender	status	species	shelter
7	Pinocchio	9826976856	2024-01-19 15:26:30	7	Meowmix	Munchkin	Yellow	Female	0	Cat	1
8	Ella	9009787866	2024-01-19 15:27:16	8	Ash	Persian	Gray	Female	0	Cat	1

```
2 rows in set (0.00 sec)
```

28.Like operator

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

```
mysql> select * from adoptions where name like '%a%';
```

animal_id	name	contact	date
8	Ella	9009787866	2024-01-19 15:27:16

```
1 row in set (0.00 sec)
```

29.And operator

select * from adoptions where name='ELLa' and animal_id=8;

```
mysql> select * from adoptions where name='ELLa' and animal_id=8;
```

animal_id	name	contact	date
8	Ella	9009787866	2024-01-19 15:27:16

```
1 row in set (0.00 sec)
```

30.Between operator

select * from adoptions where animal_id between 7 and 9;

```
mysql> select * from adoptions where animal_id between 7 and 9;
```

animal_id	name	contact	date
7	Pinocchio	9826976856	2024-01-19 15:26:30
8	Ella	9009787866	2024-01-19 15:27:16

```
2 rows in set (0.00 sec)
```

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.

```
mysql> select a.name,a.animal_id,a.breed,s.name as shelter_name from animals as a inner join shelters as s on a.shelter=s.id;
```

name	animal_id	breed	shelter_name
Bellyflop	1	Beagle	Animals 4 Homes
Snowy	2	Husky	Animals 4 Homes
Princess	3	Pomeranian	Animals 4 Homes
Cricket	4	Chihuahua	Animals 4 Homes
Princess	5	Poodle	Animals 4 Homes
Spot	6	Dalmation	Animals 4 Homes
Meowmix	7	Munchkin	Animals 4 Homes
Ash	8	Persian	Animals 4 Homes
Tiger	9	Bengal	Animals 4 Homes
Snoops	10	Beagle	Adopt A Buddy
Salt	11	Turkish Angora	Adopt A Buddy
Fuzz	12	Papillon	Fluffy Animals

12 rows in set (0.00 sec)

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.

```
mysql> select a.name,a.animal_id,a.breed from animals as a left join shelters as s on a.shelter=s.id;
```

name	animal_id	breed
Bellyflop	1	Beagle
Snowy	2	Husky
Princess	3	Pomeranian
Cricket	4	Chihuahua
Princess	5	Poodle
Spot	6	Dalmation
Meowmix	7	Munchkin
Ash	8	Persian
Tiger	9	Bengal
Snoops	10	Beagle
Salt	11	Turkish Angora
Fuzz	12	Papillon

12 rows in set (0.00 sec)

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.

```
mysql> select s.name, a.breed from animals as a right join shelters as s on a.shelter=s.id;
```

name	breed
Animals 4 Homes	Beagle
Animals 4 Homes	Husky
Animals 4 Homes	Pomeranian
Animals 4 Homes	Chihuahua
Animals 4 Homes	Poodle
Animals 4 Homes	Dalmation
Animals 4 Homes	Munchkin
Animals 4 Homes	Persian
Animals 4 Homes	Bengal
Adopt A Buddy	Beagle
Adopt A Buddy	Turkish Angora
Fluffy Animals	NULL

```
12 rows in set (0.00 sec)
```

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.

```
mysql> select s.name, a.breed from animals as a left join shelters as s on a.shelter=s.id union select s.name, a.breed from animals as a right join shelters as s on a.shelter=s.id;
```

name	breed
Animals 4 Homes	Beagle
Animals 4 Homes	Husky
Animals 4 Homes	Pomeranian
Animals 4 Homes	Chihuahua
Animals 4 Homes	Poodle
Animals 4 Homes	Dalmation
Animals 4 Homes	Munchkin
Animals 4 Homes	Persian
Animals 4 Homes	Bengal
Adopt A Buddy	Beagle
Adopt A Buddy	Turkish Angora
Fluffy Animals	NULL

```
12 rows in set (0.01 sec)
```

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

EQUI JOIN creates a JOIN for equality or matching column(s) values of the relative tables. EQUI 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 (=).

```
mysql> select s.name, a.breed from animals as a join shelters as s on a.shelter=s.id;
```

name	breed
Animals 4 Homes	Beagle
Animals 4 Homes	Husky
Animals 4 Homes	Pomeranian
Animals 4 Homes	Chihuahua
Animals 4 Homes	Poodle
Animals 4 Homes	Dalmation
Animals 4 Homes	Munchkin
Animals 4 Homes	Persian
Animals 4 Homes	Bengal
Adopt A Buddy	Beagle
Adopt A Buddy	Turkish Angora

```
11 rows in set (0.00 sec)
```

NON EQUI JOIN

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

```
mysql> select s.name, a.breed from animals as a join shelters as s on a.shelter>s.id;
```

name	breed
Animals 4 Homes	Beagle
Animals 4 Homes	Turkish Angora

```
2 rows in set (0.00 sec)
```

