

ASSIGNMENT-3

SQL:

- Structured Query Language, is designed for managing and manipulating relational database systems.
- SQL is used for tasks such as querying data, updating data, inserting data, and managing database structures.

Types of SQL commands:

- 1) DDL (Data Definition Language)
- 2) DML (Data Manipulation Language)
- 3) DCL (Data Control Language)
- 4) TCL (Transcation Control Language)

1) Data Definition Language:

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.,
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.
- DDL commands are CREATE, ALTER, DROP, TRUNCATE

2) Data Manipulation Language:

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
- DML commands are INSERT, UPDATE, DELETE

3) Data Control Language:

- DCL commands are used to grant and take back authority from any database user.
- DCL commands are GRANT, REVOKE
- GRANT- It is used to give user access privileges to a database.
- REVOKE- It is used to take back permissions from the user.

4) Transaction Control Language:

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.
- TCL commands are COMMIT, ROLLBACK , SAVEPOINT

SQL Queries:

1) Create database:

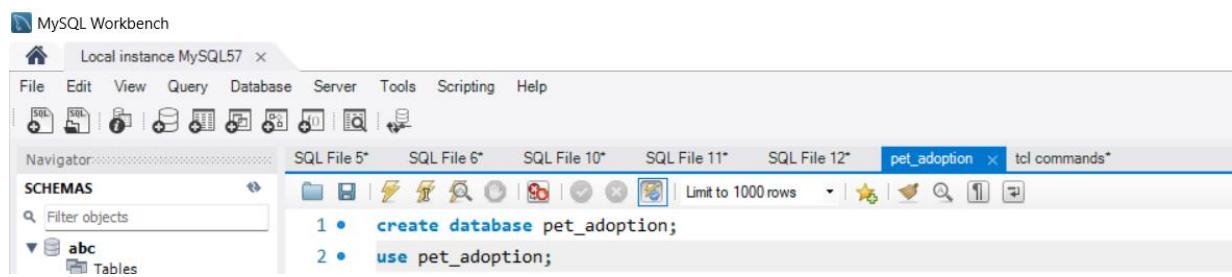
```
create database pet_adoption;
```

It creates the database named pet_adoption

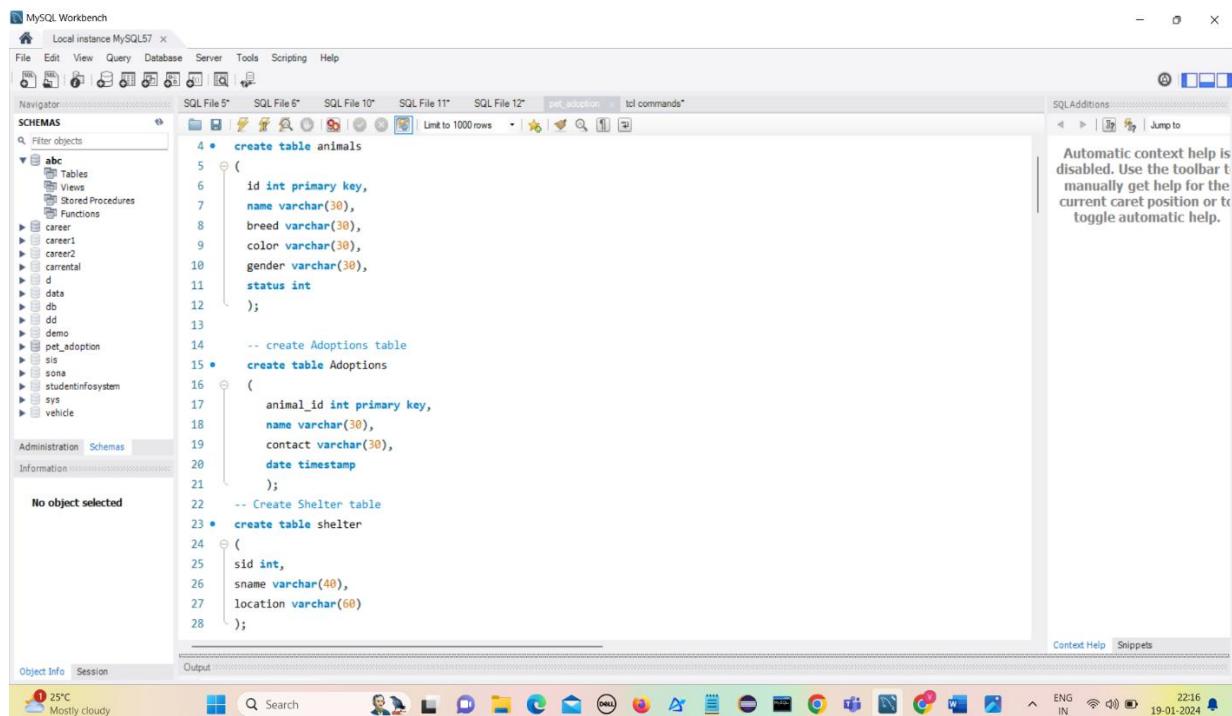
2) Use database:

```
use pet_adoption;
```

It makes pet_adoption database as the current working database on which operations are performed.



3) Create tables:



create command creates animals table, adoptions table and shelter table with different attributes.

4) Show tables in the database

It shows all the tables in the database

```
MySQL Workbench - Local instance MySQL57
```

File Edit View Query Database Server Tools Scripting Help

Schemas

- abc
- Tables
- Views
- Stored Procedures
- Functions
- career
- career1
- career2
- currental
- data
- dd
- demo
- pet_adoption
- Tables
- adoptions
- Columns
- animal_id
- name
- contact
- shelter_id

Administration Schemas Information

No object selected

SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* | sql commands*

```
27   location varchar(60)
28 );
29
30 -- Show all the tables in database
31 * show tables;
32 -- Show columns from tables
33 * show columns from animals;
34 * show columns from adoptions;
35 * show columns from shelter;
36
37 -- Insert records into animals table
38 insert into animals(id,name,breed,color,gender,status) values (1,'Bellyflop','Beagle','Brown','Male',0);
```

Result Grid | Filter Rows Export Wrap Cell Contents

Tables_in_pet_adoption	adoptions	animals	shelter

Result 82 x Output Action Output

Object Info Session

Upcoming Earnings Search

22:27 19-01-2024 ENG IN WiFi

5) Show columns from tables

It displays all the columns in the particular table.

```
MySQL Workbench - Local instance MySQL57
```

File Edit View Query Database Server Tools Scripting Help

Schemas

- abc
- Tables
- Views
- Stored Procedures
- Functions
- career
- career1
- career2
- currental
- data
- dd
- demo
- pet_adoption
- Tables
- adoptions
- Columns
- animal_id
- name
- contact
- shelter_id

Administration Schemas Information

No object selected

SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* | sql commands*

```
27   location varchar(60)
28 );
29
30 -- Show all the tables in database
31 * show tables;
32 -- Show columns from tables
33 * show columns from animals;
34 * show columns from adoptions;
35 * show columns from shelter;
36
37 -- Insert records into animals table
38 insert into animals(id,name,breed,color,gender,status) values (1,'Bellyflop','Beagle','Brown','Male',0);
```

Result Grid | Filter Rows Export Wrap Cell Contents

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI		
Animal_name	varchar(255)	YES			
breed	varchar(20)	YES	UNI		
color	varchar(20)	YES			
gender	varchar(20)	YES			
status	int(11)	YES			
price	double	YES			
species	varchar(30)	YES			
shelter_id	int(11)	YES			

Result 86 x Output Action Output

Object Info Session

Upcoming Earnings Search

22:28 19-01-2024 ENG IN WiFi

6) INSERT STATEMENT

Insert records into animals, adoptions and shelter table.

```
MySQL Workbench - Local instance MySQL57
```

File Edit View Query Database Server Tools Scripting Help

Schemas

- abc
- Tables
- Views
- Stored Procedures
- Functions
- career
- career1
- career2
- currental
- data
- dd
- demo
- pet_adoption
- Tables
- adoptions
- Columns
- animal_id
- name
- contact
- date
- shelter_id

Administration Schemas Information

No object selected

SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* | sql commands*

```
37
38 * insert into animals(id,name,breed,color,gender,status) values (1,'Bellyflop','Beagle','Brown','Male',0);
39 * insert into animals(id,name,breed,color,gender,status) values (2,'Snowy','Husky','White','Female',0),
40   ('Princess','Pomeranian','Black','Female',0),
41   ('Cricket','Chihuahua','Brown','Male',0),
42   ('Spot','Dalmatian','Black and white','Male',0);
43   ('Prince','Poodle','Purple','Female',0),
44   ('Fluffy','Bichon Frise','White','Female',0);
45
46 * insert into adoptions (animal_id,name,contact,date) values(1,'X','009090909090','2024-01-19 12:34:56');
47 * insert into adoptions (animal_id,name,contact,date) values(4,'Y','7070707070',now());
48 * select * from adoptions;
49
50 * insert into shelter (sid,sname,location) values (1, 'Animals 4 Homes', 'Red City'),
51   (2, 'Adopt A Buddy', 'Green Town'),
52   (3, 'Fluffy Animals', 'Blue Hills');
53 * select * from shelter;
```

Result Grid | Filter Rows Export Wrap Cell Contents

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI		
Animal_name	varchar(255)	YES			
breed	varchar(20)	YES	UNI		
color	varchar(20)	YES			
gender	varchar(20)	YES			

Result 86 x Output Action Output

Object Info Session

24°C Haze Search

22:48 19-01-2024 ENG IN WiFi

7) Select statement

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MySQL57, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (abc, career1, career2, currental, data, db, dd, demo, pet_adoption), Tables, Views, Stored Procedures, Functions.
- SQL Editor:** SQL File 5*, SQL File 6*, SQL File 10*, SQL File 11*, SQL File 12*, pet_adoption*, tc1 commands*. The code is:

```
55 -- Select all records from shelter
56 *
57 *   select * from shelter;
58 *
59 *   -- Select breeds of all dogs
60 *
61 *   select breed from animals;
```
- Result Grid:** shelter 87 x. The result shows three rows:

id	name	location
1	Animals 4 Homes	Red City
2	Adopt A Buddy	Green Town
3	Pluffy Animals	Blue Hills
- SQLAdditions:** Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
- Object Info:** Session, Output, Action Output.
- System Bar:** 24°C Haze, Search, 22:51, ENG IN, 19-01-2024.

This statement selects all the records from shelter table

8) SELECT DISTINCT

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MySQL57, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (abc, career1, career2, currental, data, db, dd, demo, pet_adoption), Tables, Views, Stored Procedures, Functions.
- SQL Editor:** SQL File 5*, SQL File 6*, SQL File 10*, SQL File 11*, SQL File 12*, pet_adoption*, tc1 commands*. The code is:

```
61 -- DISTINCT
62
63 *   select distinct name from animals where genders='female';
64 *
65 *   count of distinct names of animals
```
- Result Grid:** shelter 87 x. The result shows three distinct rows:

id	name	location
1	Animals 4 Homes	Red City
2	Adopt A Buddy	Green Town
3	Pluffy Animals	Blue Hills
- SQLAdditions:** Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
- Object Info:** Session, Output, Action Output.
- System Bar:** 24°C Haze, Search, 22:53, ENG IN, 19-01-2024.

It selects all the distinct values i.e., distinct names of all female dogs

9) Count of distinct names of animals

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MySQL57, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (abc, career1, career2, currental, data, db, dd, demo, pet_adoption), Tables, Views, Stored Procedures, Functions.
- SQL Editor:** SQL File 5*, SQL File 6*, SQL File 10*, SQL File 11*, SQL File 12*, pet_adoption*, tc1 commands*. The code is:

```
66
67   -- count of distinct names of animals
68 *   select count(distinct Animal_name) as distinct_count from animals;
69
```
- Result Grid:** Result 88 x. The result shows one row:

distinct_count
10
- Object Info:** Session, Output, Action Output.
- System Bar:** 24°C Haze, Search, 22:53, ENG IN, 19-01-2024.

This query gives the count of distinct animal names from animals table.

10) ALTER COMMANDS

-- ALTER command ADD column

```
alter table animals add price float;
```

It adds a column price in animals table

-- ALTER command change column datatype

```
alter table animals modify column price double;
```

It changes the data type of price from float to double

-- ALTER command add constraints on column

```
alter table animals add constraint unique_constraint unique(breed);
```

It adds a unique constraint to breed column which doesnot allow repeated values

-- ALTER command DROP column

```
alter table animals drop column price;
```

It drops the price column from animals table

-- ALTER command RENAME column

```
alter table animals CHANGE name Animal_name VARCHAR(255);
```

It renames the name in animals table to Animal_name

-- TRUNCATE command

```
truncate table animals;
```

It truncates animals table

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following code:

```
73
74 -- DDL commands
75 -- ALTER command ADD column
76 • alter table animals add price float;
77 -- ALTER command change column datatype
78 • alter table animals modify column price double;
79 -- ALTER command add constraints on column
80 • alter table animals add constraint unique_constraint unique(breed);
81 -- ALTER command DROP column
82 • alter table animals drop column price;
83 -- ALTER command RENAME column
84 • ALTER TABLE animals CHANGE name Animal_name VARCHAR(255);
85 -- TRUNCATE command
86 • truncate table animals;
87
88
89
```

The Result Grid below shows a single row with the value 10.

11) DML Commands

-- Update statement

```
update animals set color='White' where breed='Poodle';
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MySQL57, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (abc, career1, career2, currental, d, data, db, dd, demo, pet_adoption (adoptions, Columns, animal_id, name, contact)).
- SQL Editor:** pet_adoption tab, SQL File 12*, 92 -- DML commands, 93 -- Update, 94 update animals set color='White' where breed='Poodle';, 95 select * from animals;, 96 update animals set price=12000.80 where id>1.
- Result Grid:** Shows a table with columns: id, Animal_name, breed, color, gender, status, price, species, shelter_id. The data includes rows for various animals like Bellyflop, Snowy, Princess, Cricket, Fuzz, Papillon, Salt, Snoppo, Tiger, Ash, and Meowmix.
- Right Panel:** SQLAdditions, Result Grid, Form Editor, Field Types, Query Stats, Execution Plan.
- Bottom:** Taskbar showing various application icons and system status.

It updates the color of animal to White whose breed is Poodle.

-- Delete statement

delete from animals where breed='Dalmation'; It deletes animal whose breed is Dalmation from animals table

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MySQL57, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (abc, career1, career2, currental, d, data, db, dd, demo, pet_adoption (adoptions, Columns, animal_id, name, contact)).
- SQL Editor:** pet_adoption tab, SQL File 12*, 97, 98 -- Delete, 99 delete from animals where breed='Dalmation';, 100 select * from animals;, 101.
- Result Grid:** Shows a table with columns: id, Animal_name, breed, color, gender, status, price, species, shelter_id. The data includes rows for various animals like Bellyflop, Snowy, Princess, Cricket, Fuzz, Papillon, Salt, Snoppo, Tiger, Ash, and Meowmix.
- Right Panel:** SQLAdditions, Result Grid, Form Editor, Field Types, Query Stats, Execution Plan.
- Bottom:** Taskbar showing various application icons and system status.

12) LOGICAL OPERATORS

1) AND operator

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL57, pet_adoption.
- SQL Editor:** pet_adoption*, SQL commands*. The code is:

```
134 -- LOGICAL OPERATORS
135 -- 1) AND operator
136 -- Retrieve animals that are both black and female.
137 • select * from animals where color='black' and gender='female';
```
- Result Grid:** Shows a table with columns: id, Animal_name, breed, color, gender, status, price, species, shelter_id. The data is:

id	Animal_name	breed	color	gender	status	price	species	shelter_id
3	Princess	Pomeranian	Black	Female	0	23459.122	dog	1
- Toolbar:** Result Grid, Filter Rows, Edit, Export/Import, Wrap Cell Content, SQLAdditions, Jump to, Context Help, Snippets.
- System Bar:** 24°C Haze, Search, Task View, Mail, File, Internet Explorer, Firefox, Google Chrome, Microsoft Edge, 23:26, 19-01-2024.

This select statement retrieves animals that are both black and female. AND operator returns records if both conditions are true. It returns TRUE if both Boolean expressions are TRUE.

2) IN operator

Retrieve animals with specific breeds from a list.

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL57, pet_adoption.
- SQL Editor:** pet_adoption*, SQL commands*. The code is:

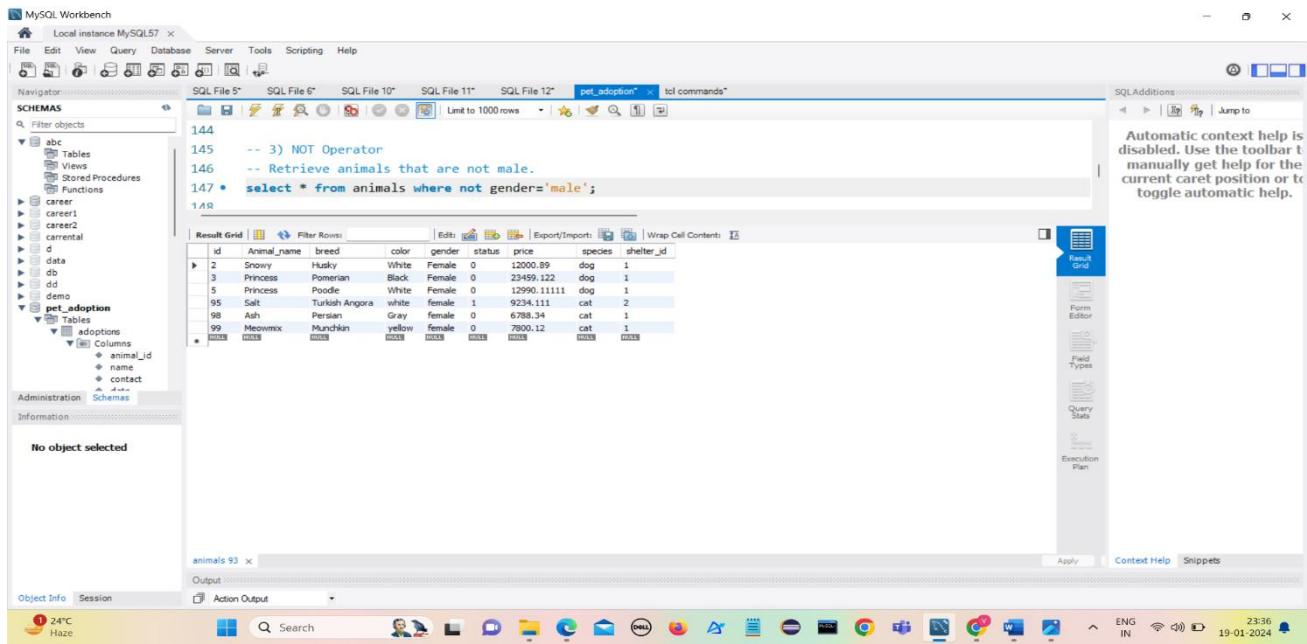
```
140 -- 2) IN operator
141 -- Retrieve animals with specific breeds from a list.
142 • select * from animals where breed in ('Beagle','Husky','Poodle');
```
- Result Grid:** Shows a table with columns: id, Animal_name, breed, color, gender, status, price, species, shelter_id. The data is:

id	Animal_name	breed	color	gender	status	price	species	shelter_id
1	Belly flop	Beagle	Brown	Male	1	12345.88	dog	1
2	Snowy	Husky	White	Female	0	12000.89	dog	1
5	Princess	Poodle	White	Female	0	12990.1111	dog	1
- Toolbar:** Result Grid, Filter Rows, Edit, Export/Import, Wrap Cell Content, SQLAdditions, Jump to, Context Help, Snippets.
- System Bar:** 24°C Haze, Search, Task View, Mail, File, Internet Explorer, Firefox, Google Chrome, Microsoft Edge, 23:32, 19-01-2024.

This select statement returns all the animals whose breed is in Beagle or Husky or Poodle. It returns TRUE if the operand is equal to one of a list of expressions.

3) NOT operator

Retrieve animals that are not male.



The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- 3) NOT Operator
-- Retrieve animals that are not male.
select * from animals where not gender='male';
```

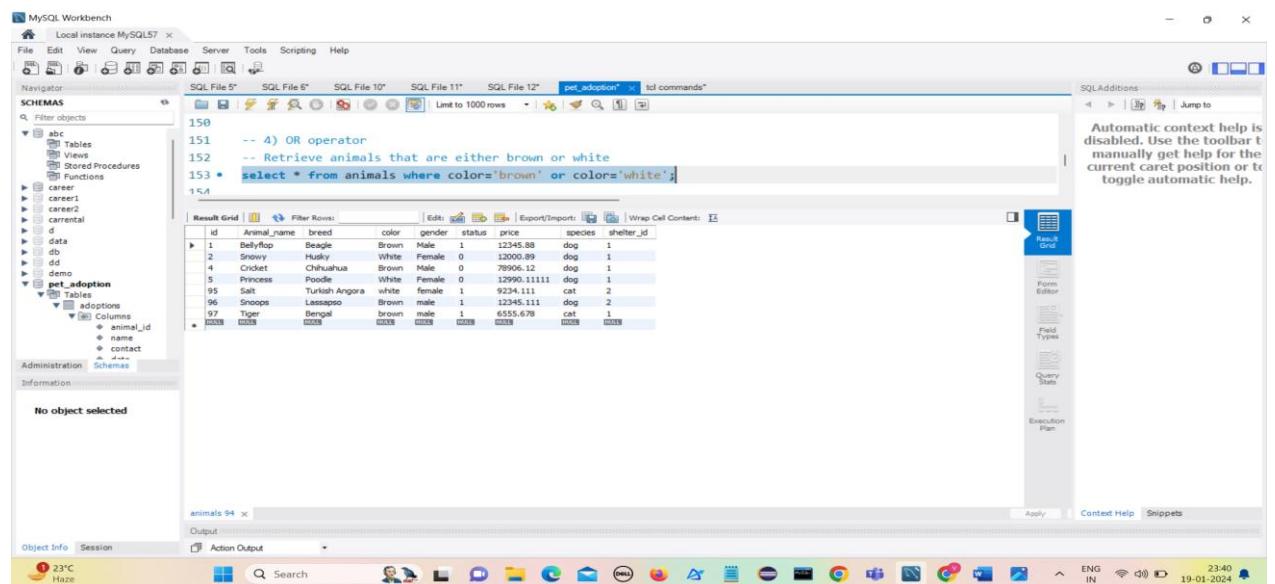
The results grid displays the following data:

ID	Animal_name	Breed	Color	Gender	Status	Price	Species	Shelter_ID
2	Snowy	Husky	White	Female	0	12000.89	Dog	1
5	Princess	Pomeranian	Black	Female	0	23459.122	Dog	1
5	Princess	Pomeranian	White	Female	0	12000.11111	Dog	1
95	Salt	Turkish Angora	White	Female	1	9234.111	Cat	2
98	Ash	Persian	Grey	Female	0	6788.34	Cat	1
99	Mexomix	Munchkin	Yellow	Female	0	7800.12	Cat	1

It returns all the animals that are not male i.e., female. It Reverses the value of any other Boolean operator.

4) OR operator

Retrieve animals that are either brown or white



The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```
-- 4) OR operator
-- Retrieve animals that are either brown or white
select * from animals where colors='brown' or colors='white';
```

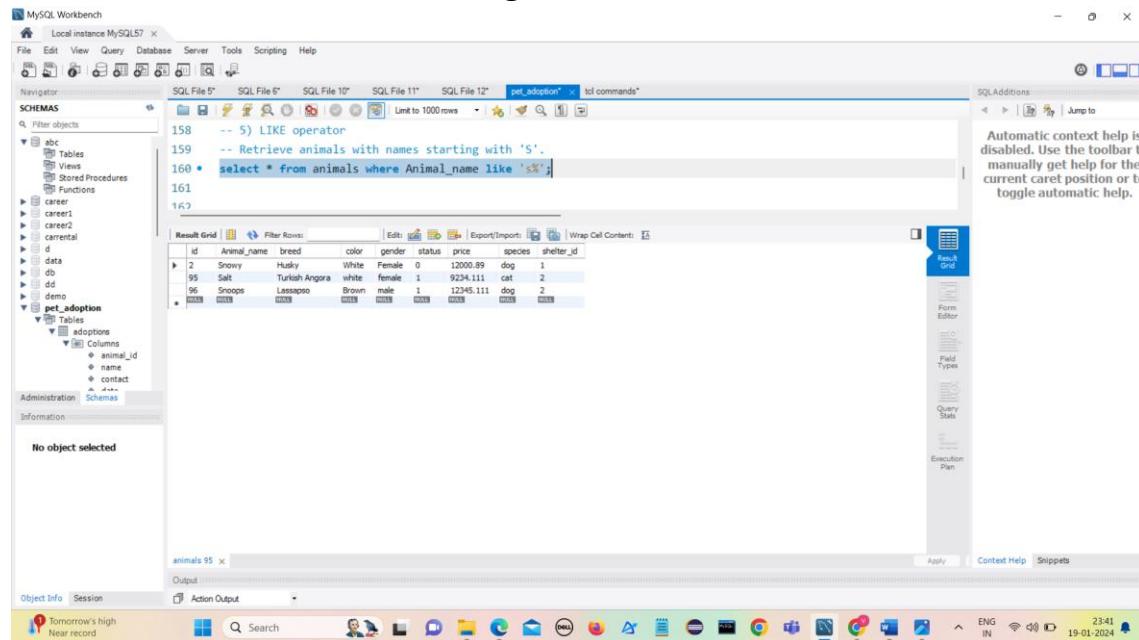
The results grid displays the following data:

ID	Animal_name	Breed	Color	Gender	Status	Price	Species	Shelter_ID
1	Bellyflop	Beagle	Brown	Male	1	12345.88	Dog	1
2	Snowy	Husky	White	Female	0	12000.89	Dog	1
4	Cricket	Chihuahua	Brown	Male	0	78906.12	Dog	1
5	Princess	Poodle	White	Female	0	12000.11111	Dog	1
14	Urma	Turkish Angora	White	Female	1	9234.111	Cat	2
96	Snoops	Lhasapoo	Brown	Male	1	12345.111	Dog	2
97	Tiger	Bengal	Brown	Male	1	6555.678	Cat	1

This query returns all animals whose color is brown or white. It returns TRUE if either Boolean expression is TRUE.

5) LIKE operator

Retrieve animals with names starting with ‘S’



The screenshot shows the MySQL Workbench interface with the following details:

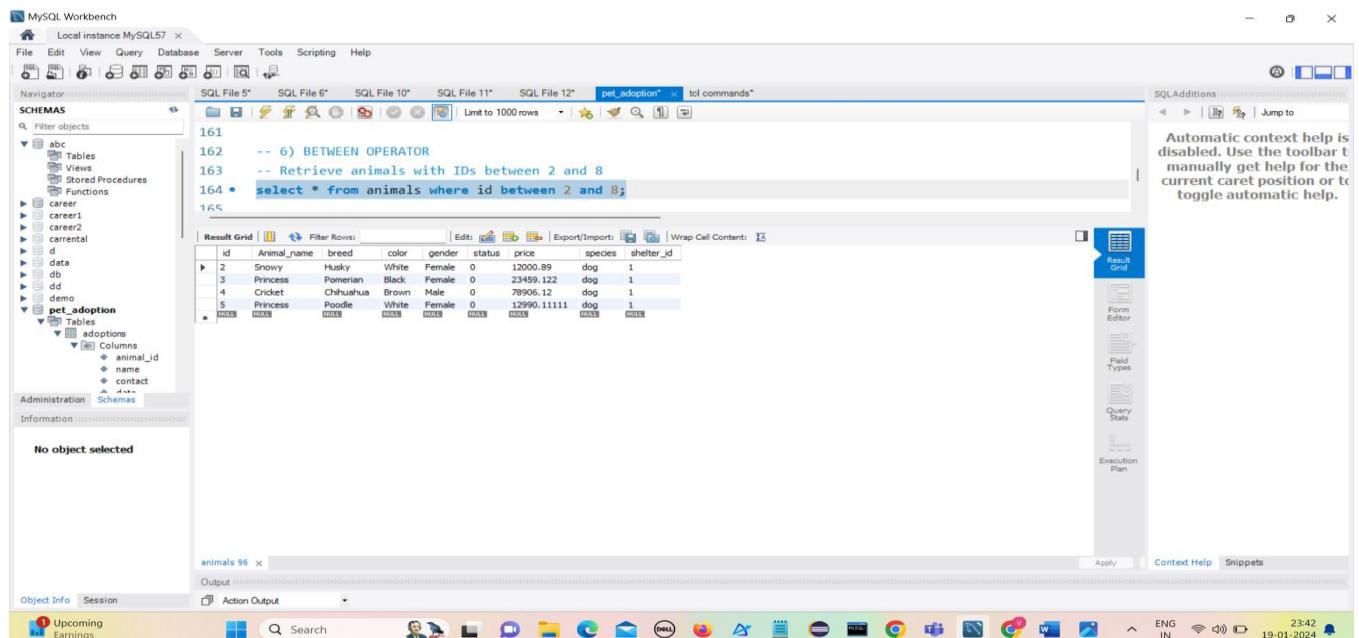
- Navigator:** Shows the schema structure, including the `pet_adoption` database and its tables (`adoptions`, `animals`, `contact`).
- SQL Editor:** Contains the following SQL code:

```
-- 5) LIKE operator
-- Retrieve animals with names starting with 'S'.
select * from animals where Animal_name like 'S%';
```
- Result Grid:** Displays the results of the query, showing 95 rows of animal data. The columns are: id, Animal_name, breed, color, gender, status, price, species, and shelter_id.
- Output:** Shows the output of the query, which is the same as the Result Grid.
- System Bar:** Shows the date and time as 19-01-2024 23:41.

This query returns animals whose name starts with ‘S’. It returns TRUE if the operand matches a pattern.

6) BETWEEN OPERATOR

Retrieve animals with ids between 2 and 8



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure, including the `pet_adoption` database and its tables (`adoptions`, `animals`, `contact`).
- SQL Editor:** Contains the following SQL code:

```
-- 6) BETWEEN OPERATOR
-- Retrieve animals with IDs between 2 and 8
select * from animals where id between 2 and 8;
```
- Result Grid:** Displays the results of the query, showing 96 rows of animal data. The columns are: id, Animal_name, breed, color, gender, status, price, species, and shelter_id.
- Output:** Shows the output of the query, which is the same as the Result Grid.
- System Bar:** Shows the date and time as 19-01-2024 23:42.

This query gives animals whose IDs lie between 2 and 8. It returns TRUE if the operand is within a range.

7) ALL Operator

Retrieve animals with id greater than all ids in adoption table

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** abc, career, career1, career2, currental, d, data, db, dd, demo, pet_adoption.
- Tables:** pet_adoption (Columns: animal_id, name, contact, address, city, state, zip, phone, email, notes).
- Query:** SQL File 12* (pet_adoption*)
- Code:**

```
165
166
167 -- 7) ALL operator
168 -- Retrieve animals with id greater than all ids in adoption table
169 • select * from animals where id > all (select animal_id from adoptions);
170
```
- Result Grid:** Shows a table with columns: id, Animal_name, breed, color, gender, status, price, species, shelter_id. The data includes rows for Snowy, Princess, Cricket, Princess, and several others.
- Status Bar:** Shows system information like battery level (23%), network (Haze), and date/time (19-01-2024 23:44).

This query returns all animals whose id is greater than all ids in adoption table. It returns TRUE if all of a set of comparisons are TRUE.

8) ANY Operator

Retrieve animals with id greater than any ids in adoption table

The screenshot shows the MySQL Workbench interface with the following details:

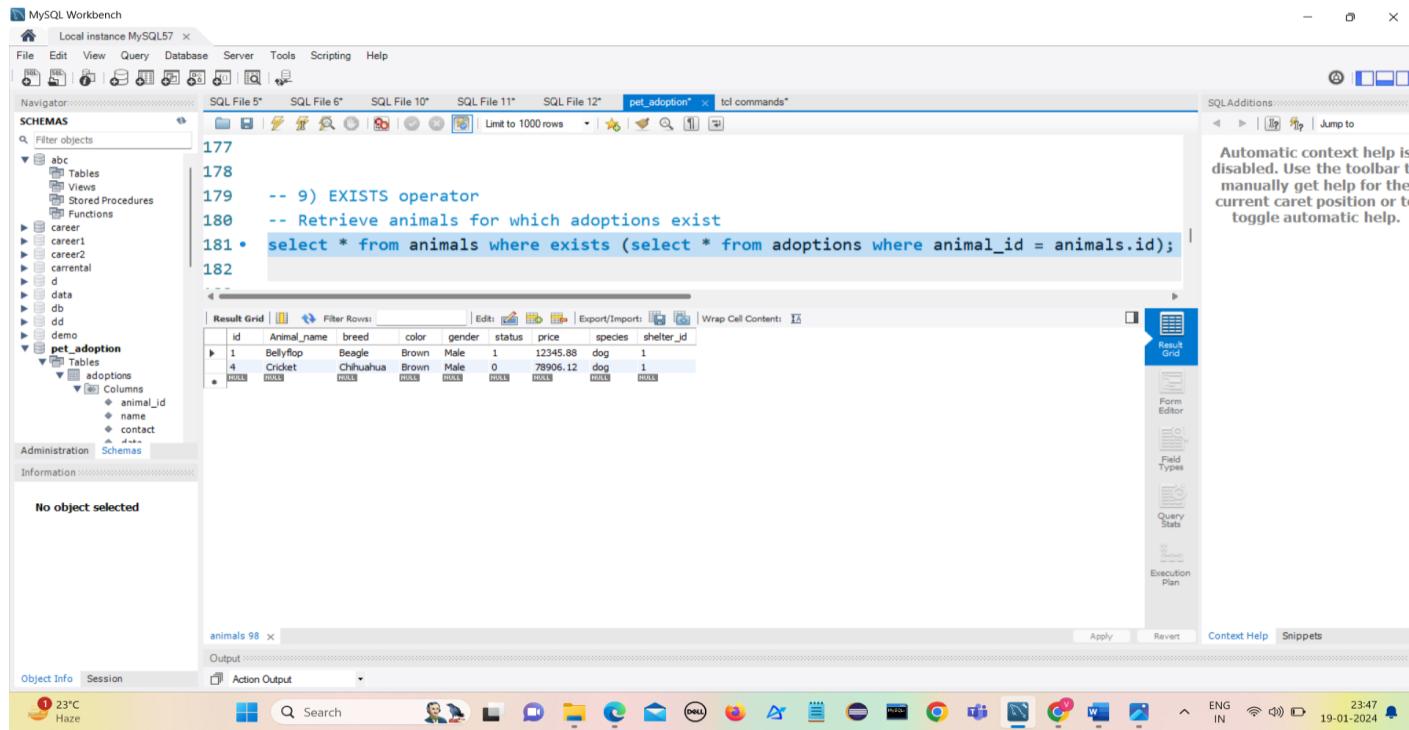
- Schemas:** abc, career, career1, career2, currental, d, data, db, dd, demo, pet_adoption.
- Tables:** pet_adoption (Columns: animal_id, name, contact, address, city, state, zip, phone, email, notes).
- Query:** SQL File 12* (pet_adoption*)
- Code:**

```
172
173 -- 8) ANY operator
174 -- Retrieve animals with id greater than any ids in adoption table
175 • select * from animals where id > any (select animal_id from adoptions);
176
177
```
- Result Grid:** Shows a table with columns: id, Animal_name, breed, color, gender, status, price, species, shelter_id. The data includes rows for Snowy, Princess, Cricket, Princess, and several others.
- Status Bar:** Shows system information like battery level (23%), network (Haze), and date/time (19-01-2024 23:44).

This query returns all animals with id greater than any ids in adoption table. It returns TRUE if any one of a set of comparisons is TRUE.

9) EXISTS Operator

Retrieve animals for which adoptions exist



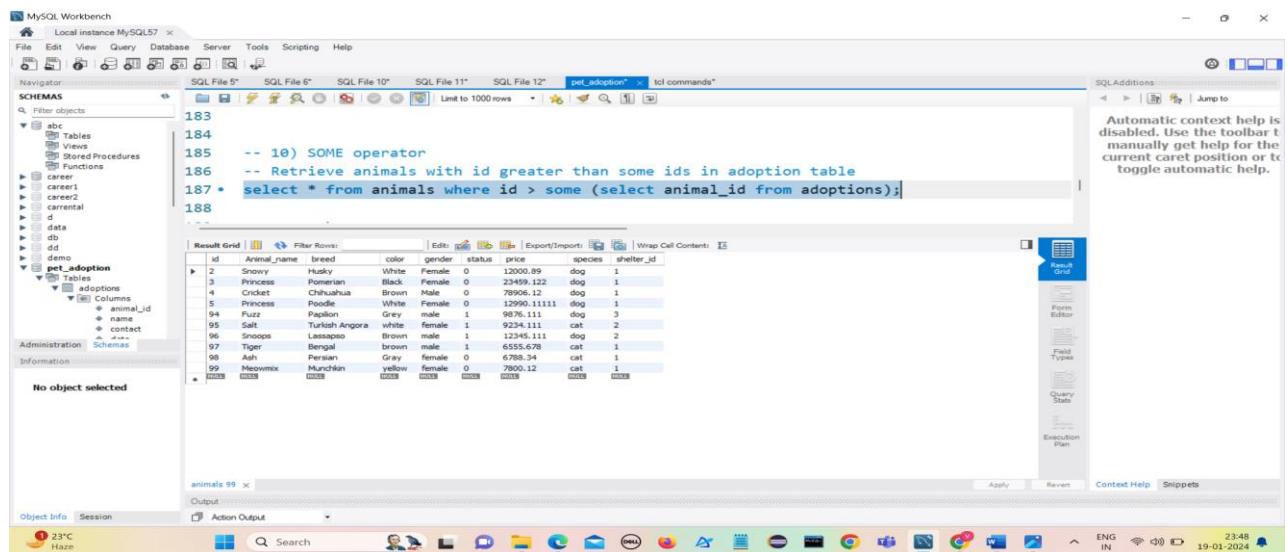
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** abc, career, career1, career2, career3, d, data, db, demo, pet_adoption, pet_adoption (Tables: adoptions, Columns: animal_id, name, contact).
- Query Editor:** SQL File 12* tab selected. The query is:

```
-- 9) EXISTS operator
-- Retrieve animals for which adoptions exist
select * from animals where exists (select * from adoptions where animal_id = animals.id);
```
- Result Grid:** Shows the results of the query, displaying 4 rows of animal data.
- Toolbar:** Includes icons for Save, Undo, Redo, Copy, Paste, and Print.
- Help:** A note says "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- System Bar:** Shows the date and time (19-01-2024, 23:47), system status (23°C Haze), and taskbar icons.

It returns all the records of animals that are available for adoption. It returns TRUE if a subquery contains any rows.

10) SOME Operator



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** abc, career, career1, career2, career3, d, data, db, demo, pet_adoption, pet_adoption (Tables: adoptions, Columns: animal_id, name, contact).
- Query Editor:** SQL File 12* tab selected. The query is:

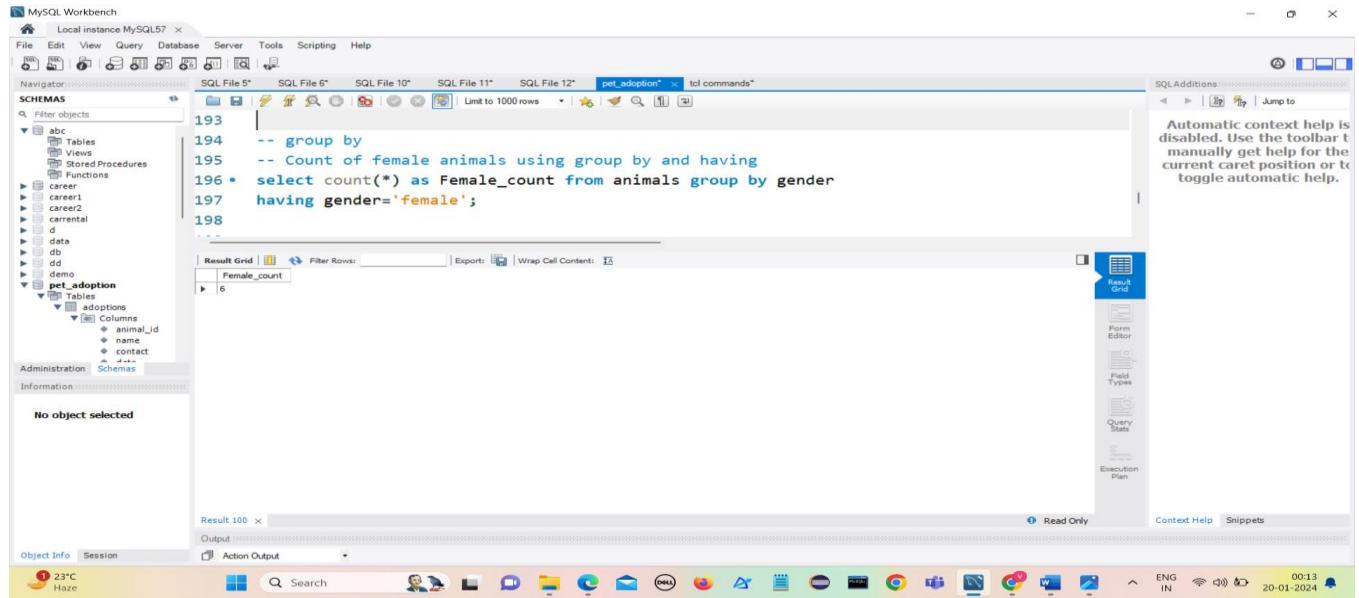
```
-- 10) SOME operator
-- Retrieve animals with id greater than some ids in adoption table
select * from animals where id > some (select animal_id from adoptions);
```
- Result Grid:** Shows the results of the query, displaying 99 rows of animal data.
- Toolbar:** Includes icons for Save, Undo, Redo, Copy, Paste, and Print.
- Help:** A note says "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- System Bar:** Shows the date and time (19-01-2024, 23:48), system status (23°C Haze), and taskbar icons.

This query returns animals with id greater than some ids in adoption table. It returns TRUE if some of a set of comparisons are TRUE.

13) GROUP BY AND HAVING CLAUSE

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions

Count of female animals using group by and having clause



```
MySQL Workbench - Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands*
schemas abc
Tables Views Stored Procedures Functions
career career1 career2 currental d data db demo pet_adoption
Tables Columns animal_id name contact
Result Grid Filter Rows: Export Wrap Cell Contents
Female_count
6
Result 100 x
Output
Object Info Session Action Output
23°C Haze Search
20-01-2024 00:13
ENG IN WiFi

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
193
194 -- group by
195 -- Count of female animals using group by and having
196 select count(*) as Female_count from animals group by gender
197 having gender='female';
198
```

This screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
193
194 -- group by
195 -- Count of female animals using group by and having
196 select count(*) as Female_count from animals group by gender
197 having gender='female';
198
```

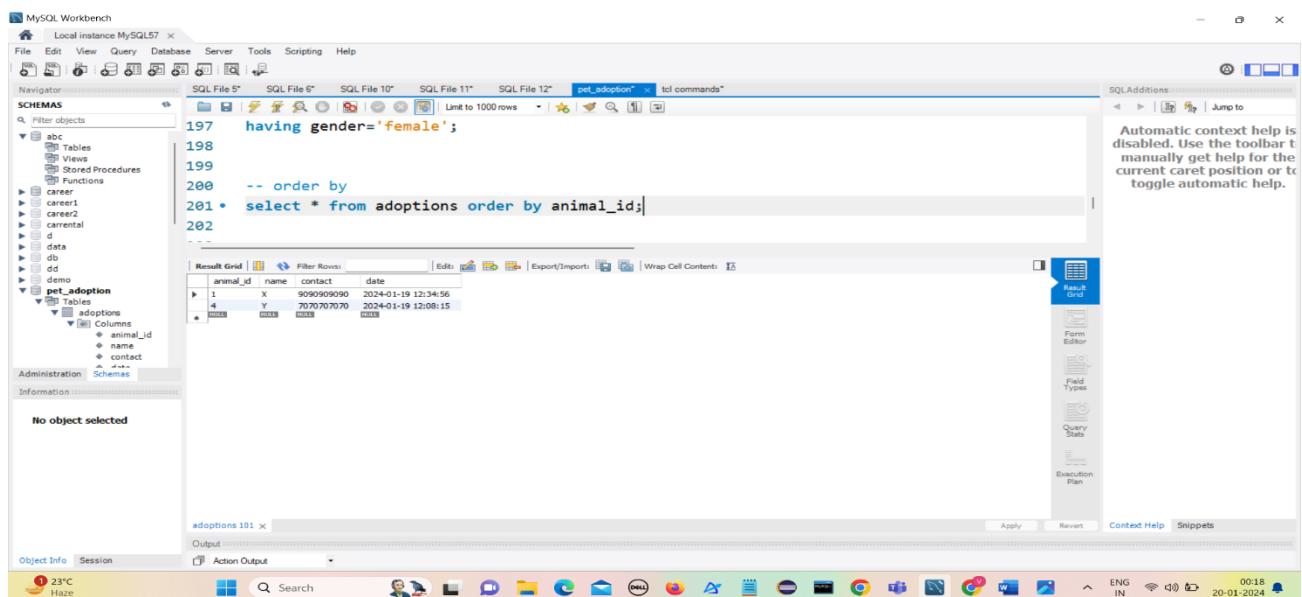
The result grid shows a single row with the value 6 for 'Female_count'. The status bar at the bottom right indicates the date and time as 20-01-2024 00:13.

This query returns the count of all the female animals using group by and having clause.

14) ORDER BY CLAUSE

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

Retrieve all the adoptions in the increasing order of animal_id



```
MySQL Workbench - Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas SQL File 5* SQL File 6* SQL File 10* SQL File 11* SQL File 12* pet_adoption* tcl commands*
schemas abc
Tables Views Stored Procedures Functions
career career1 career2 currental d data db demo pet_adoption
Tables Columns animal_id name contact
Result Grid Filter Rows: Export Wrap Cell Contents
animal_id name contact date
1 X 9090909090 2024-01-19 12:34:56
2 Y 7070707070 2024-01-19 12:08:15
Adoptions 101 x
Output
Object Info Session Action Output
23°C Haze Search
20-01-2024 00:18
ENG IN WiFi

```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
197 having gender='female';
198
199
200 -- order by
201 select * from adoptions order by animal_id;
202
```

This screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
197 having gender='female';
198
199
200 -- order by
201 select * from adoptions order by animal_id;
202
```

The result grid shows two rows of adoption data. The status bar at the bottom right indicates the date and time as 20-01-2024 00:18.

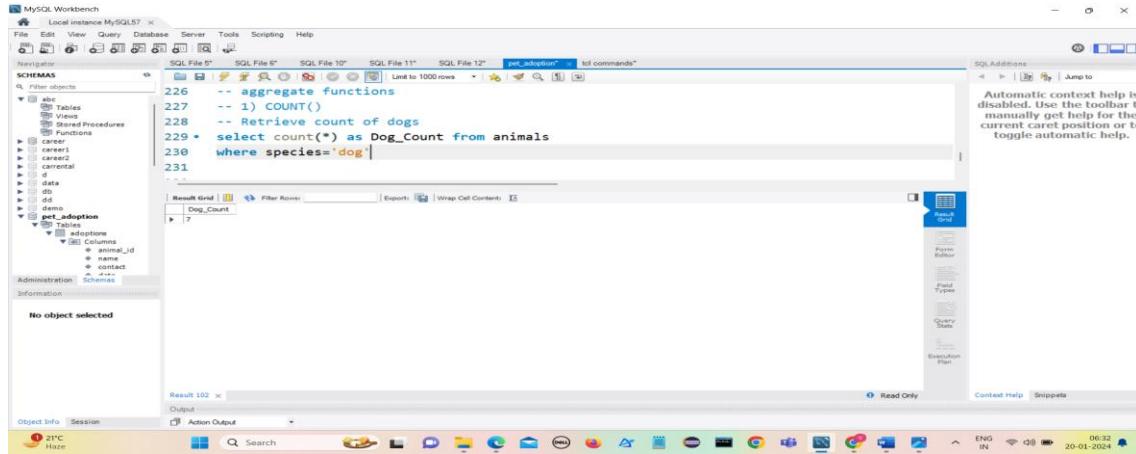
This query returns all the adoptions in the increasing order of animal_id.

15) AGGREGATE FUNCTIONS

1) COUNT()

The **COUNT()** function returns the number of rows that matches a specified criterion.

-- Retrieve count of dogs



A screenshot of the MySQL Workbench interface. The SQL editor tab contains the following code:

```
226 -- aggregate functions
227 -- 1) COUNT()
228 -- Retrieve count of dogs
229 * select count(*) as Dog_Count from animals
230 where species='dog'
231
```

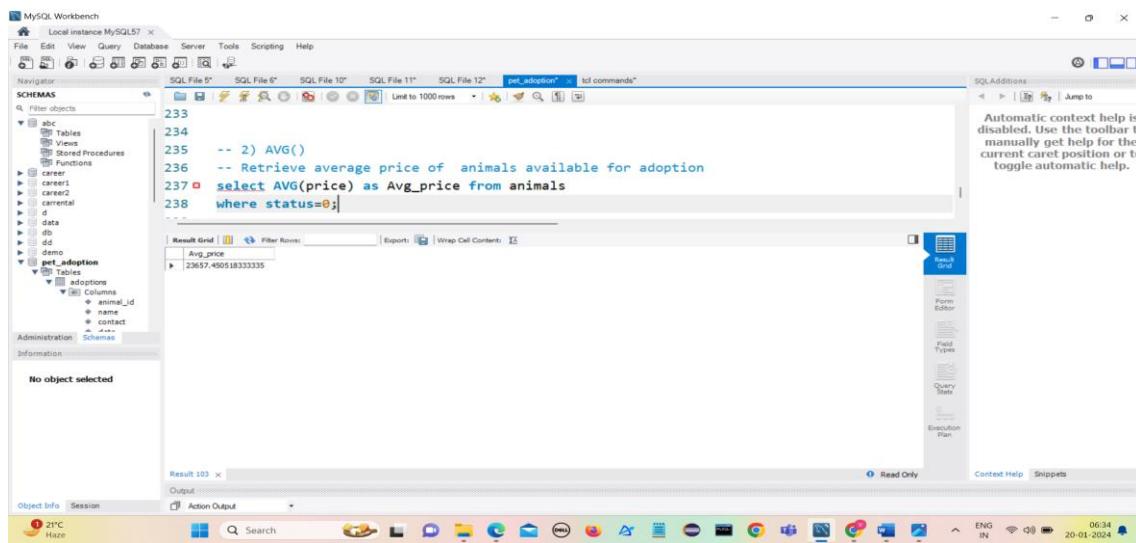
The result grid shows a single row with the value 7, labeled "Dog_Count". The status bar at the bottom right indicates the date and time as 20-01-2024 06:32.

This query retrieves the number of dogs in animal table.

2) AVG()

The **AVG()** function returns the average value of a numeric column.

Retrieve average price of animals available for adoption



A screenshot of the MySQL Workbench interface. The SQL editor tab contains the following code:

```
233
234
235 -- 2) AVG()
236 -- Retrieve average price of animals available for adoption
237 * select AVG(price) as Avg_price from animals
238 where status=0;
```

The result grid shows a single row with the value 23657.490518333333, labeled "Avg_price". The status bar at the bottom right indicates the date and time as 20-01-2024 06:34.

This query returns the average price of all animals that are available for adoption using **AVG()** function.

3) MAX()

The MAX() function returns the largest value of the selected column.

Retrieve maximum id of animals which are male.

A screenshot of the MySQL Workbench interface. The query editor window shows the following SQL code:

```
243
244 -- 3) MAX()
245 -- Retrieve maximum id of animals which are male
246 • select max(id) as Maximum_id from animals
247 where gender='male';
248
```

The result grid shows one row with the value 97 under the column 'Maximum_id'. The status bar at the bottom right indicates the date and time as 20-01-2024 06:38.

This query returns maximum id of animals which are male using MAX() function.

4) MIN()

The MIN() function returns the smallest value of the selected column.

Retrieve minimum price spent for male dogs

A screenshot of the MySQL Workbench interface. The query editor window shows the following SQL code:

```
251
252 -- 4) MIN()
253 -- Retrieve minimum price spent for male dogs
254 • select min(price) as Minimum_price from animals
255 where gender='male' and species='dog';
256
```

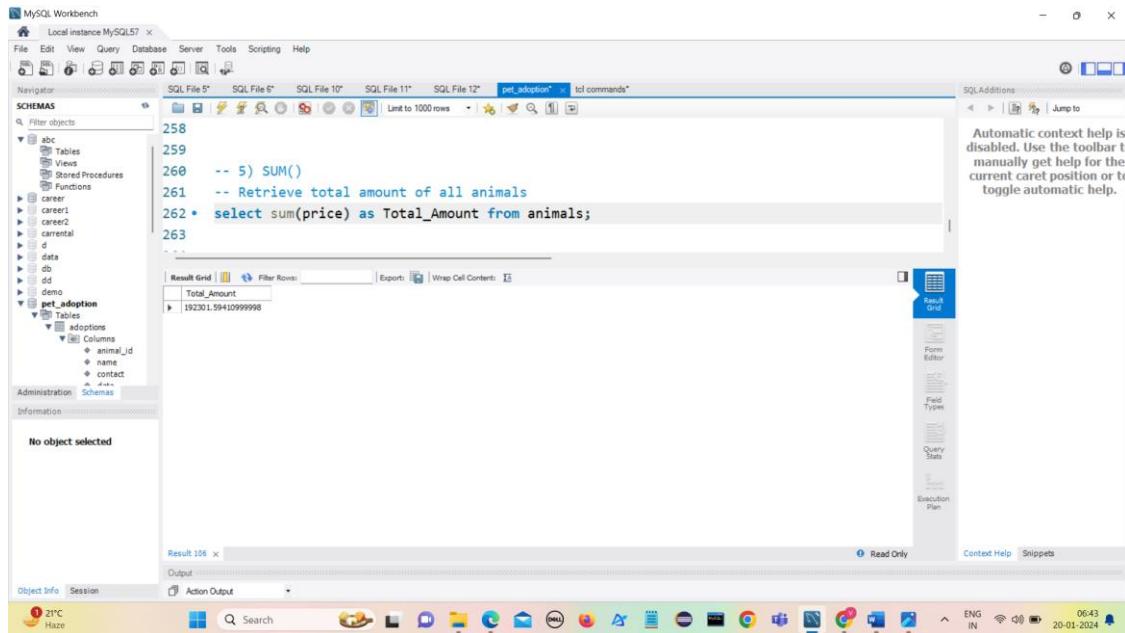
The result grid shows one row with the value 9876.111 under the column 'Minimum_price'. The status bar at the bottom right indicates the date and time as 20-01-2024 06:40.

This query returns the minimum price that is spent on dogs which are dogs using MIN() function.

5) SUM()

The SUM() function returns the total sum of a numeric column.

Retrieve total amount of all animals



```
MySQL Workbench - Local instance MySQL57 - pet_adoption* - tcl commands*
```

```
258
259
260 -- 5) SUM()
261 -- Retrieve total amount of all animals
262 * select sum(price) as Total_Amount from animals;
263
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
-- 5) SUM()
-- Retrieve total amount of all animals
select sum(price) as Total_Amount from animals;
```

The result grid shows a single row with the value 19230.159410999998.

This query returns the total amount on animals that can be cats or dogs.

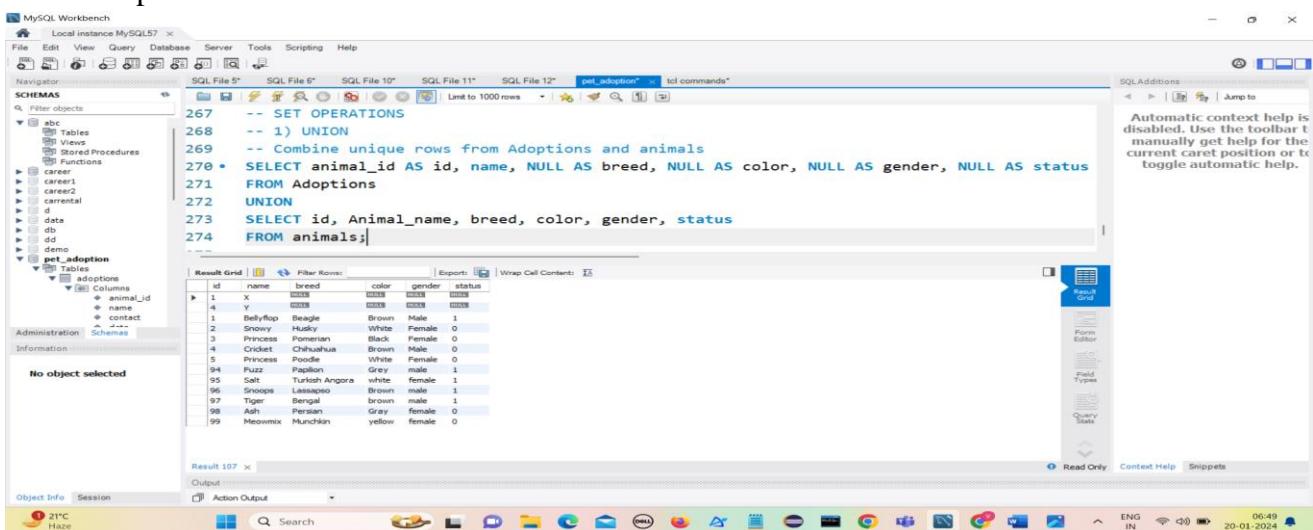
16) SET Operations in SQL

SET operators are special type of operators which are used to combine the result of two queries.

1) UNION

UNION will be used to combine the result of two select statements.

Duplicate rows will be eliminated from the results obtained after performing the UNION operation.



```
MySQL Workbench - Local instance MySQL57 - pet_adoption* - tcl commands*
```

```
267 -- SET OPERATIONS
268 -- 1) UNION
269 -- Combine unique rows from Adoptions and animals
270 * SELECT animal_id AS id, name, NULL AS breed, NULL AS color, NULL AS gender, NULL AS status
  FROM Adoptions
  UNION
273   SELECT id, Animal_name, breed, color, gender, status
  FROM animals;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
-- SET OPERATIONS
-- 1) UNION
-- Combine unique rows from Adoptions and animals
SELECT animal_id AS id, name, NULL AS breed, NULL AS color, NULL AS gender, NULL AS status
  FROM Adoptions
  UNION
  SELECT id, Animal_name, breed, color, gender, status
  FROM animals;
```

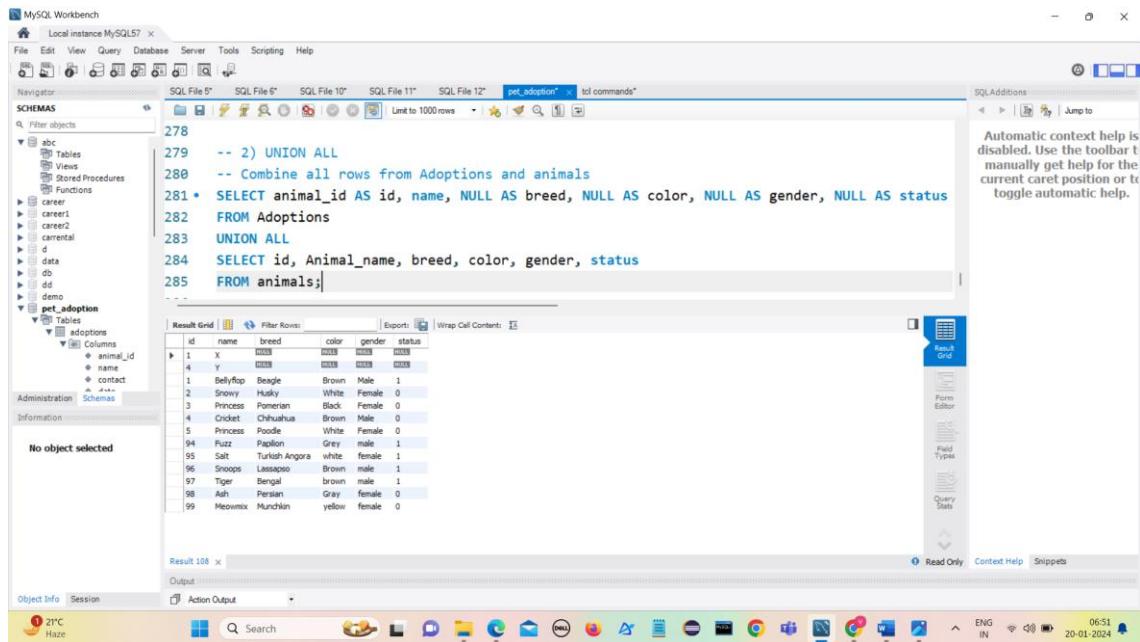
The result grid displays 99 rows of animal data, combining data from the 'Adoptions' and 'animals' tables.

This query returns unique rows from both animals and adoptions table.

2) UNION ALL

This operator combines all the records from both the queries.

Duplicate rows will be not be eliminated from the results obtained after performing the UNION ALL operation.



The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a query:

```
-- 2) UNION ALL
-- Combine all rows from Adoptions and animals
SELECT animal_id AS id, name, NULL AS breed, NULL AS color, NULL AS gender, NULL AS status
FROM Adoptions
UNION ALL
SELECT id, Animal_name, breed, color, gender, status
FROM animals;
```

The Result Grid shows the combined data from both tables:

id	name	breed	color	gender	status
1	X			Male	0
4	Y			Female	0
1	SallyFlop	Beagle	Brown	Male	1
2	Sammy	Shih Tzu	White	Female	0
3	Princess	Pomeranian	Black	Female	0
4	Cricket	Chihuahua	Brown	Male	0
5	Princess	Poodle	White	Female	0
94	Fuzz	Papillon	Grey	male	1
95	Salt	Turkish Angora	white	female	1
96	Snoopy	Lhasapso	Brown	male	1
97	Tiger	Bengal	brown	male	1
98	Ash	Persian	Gray	female	0
99	Meowmix	Munchkin	yellow	female	0

This query returns all the rows from Animals and Adoptions table without removing duplicate elements.

17) Transaction Control Commands

Transactional control commands are only used with the **DML Commands** such as - INSERT, UPDATE and DELETE. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

1) COMMIT

The COMMIT command is the transactional command used to save changes invoked by a transaction.

```
create database abc;
use abc;
```

Create the database abc and make abc database as the current working database on which operations can be performed.

Now create a table named customer using create statement and insert values into the customer table.

MySQL Workbench - Local instance MySQL57

SQL File 5*

```

1 • create database abc;
2 • use abc;
3 -- Create table Customer
4 • create table customer
5 (
6     id int not null,
7     name varchar(20) not null,
8     age int not null,
9     address char(25),
10    salary decimal(18,2),
11    primary key(id)
12 );
13 -- Insert records into customer table
14 • insert into customer values
15 (1, 'Ramesh', 32, 'Ahmedabad', 2000.00),
16 (2, 'Khalan', 25, 'Delhi', 1500.00),
17 (3, 'Kaushik', 23, 'Kota', 2000.00),
18 (4, 'Chaitali', 25, 'Mumbai', 6500.00),
19 (5, 'Hardik', 27, 'Bhopal', 8500.00),
20 (6, 'Komal', 22, 'Hyderabad', 4500.00),
21 (7, 'Muffy', 24, 'Indore', 10000.00);
22 -- Retrieve all the details of customers
23 • select * from customer;
24

```

Output

#	Time	Action	Message	Duration / Fetch
145	17:09:18	Action		
145	17:09:37	select * from customer LIMIT 0, 1000	Error Code: 1146. Table 'abc.customers' doesn't exist 7 row(s) returned	0.000 sec / 0.000 sec

No object selected

Object Info Session

System tray icons: 30°C Partly sunny, Search, Home, Task View, Mail, Dell, Firefox, File Explorer, Taskbar, 07:08, ENG IN, 20-01-2024

Retrieve all the details from customer table using select statement

MySQL Workbench - Local instance MySQL57

SQL File 6*

```

16 (2, 'Khalan', 25, 'Delhi', 1500.00),
17 (3, 'Kaushik', 23, 'Kota', 2000.00),
18 (4, 'Chaitali', 25, 'Mumbai', 6500.00),
19 (5, 'Hardik', 27, 'Bhopal', 8500.00),
20 (6, 'Komal', 22, 'Hyderabad', 4500.00),
21 (7, 'Muffy', 24, 'Indore', 10000.00);
22 -- Retrieve all the details of customers
23 • select * from customer;
24

```

Result Grid

ID	Name	Age	Address	Salary
1	Ramesh	32	Ahmedabad	2000
2	Khalan	25	Delhi	1500
3	Kaushik	23	Kota	2000
4	Chaitali	25	Mumbai	6500
5	Hardik	27	Bhopal	8500
6	Komal	22	Hyderabad	4500
7	Muffy	24	Indore	10000
8	None	None	None	None

customer 7

Output

#	Time	Action	Message	Duration / Fetch
145	17:09:18	Action		
145	17:09:37	select * from customer LIMIT 0, 1000	Error Code: 1146. Table 'abc.customers' doesn't exist 7 row(s) returned	0.000 sec / 0.000 sec

No object selected

Object Info Session

System tray icons: 30°C Partly sunny, Search, Home, Task View, Mail, Dell, Firefox, File Explorer, Taskbar, 17:10, ENG IN, 19-01-2024

Use commit statement

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** abc
- SQL Editor:** pet_adoption tab
- Code:**

```
21 (7, 'Muffy', 24, 'Indore', 10000.00);
22 -- Retrieve all the details of customers
23 • select * from customer;
24 -- COMMIT STATEMENT
25 -- Delete records from customers of age 25 and commit changes in the database
26 • delete from customer where age=25;
27 • commit;
28 • select * from customer;
29 |
```
- Result Grid:** Shows a table with columns id, name, age, address, salary. Data:

id	name	age	address	salary
1	Ramesh	32	Ahmedabad	2000.00
3	Kaushik	23	Kota	2000.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00
- Output:** Action Output shows three log entries:
 - 147 17:11:33 delete from customer where age=25: 2 row(s) affected
 - 148 17:11:33 commit: 0 row(s) affected
 - 149 17:11:58 select * from customer LIMIT 0, 1000: 5 row(s) returned
- System Bar:** Shows system icons, a search bar, and a taskbar with various application icons.

2) ROLLBACK

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.

The screenshot shows the MySQL Workbench interface with the following details:

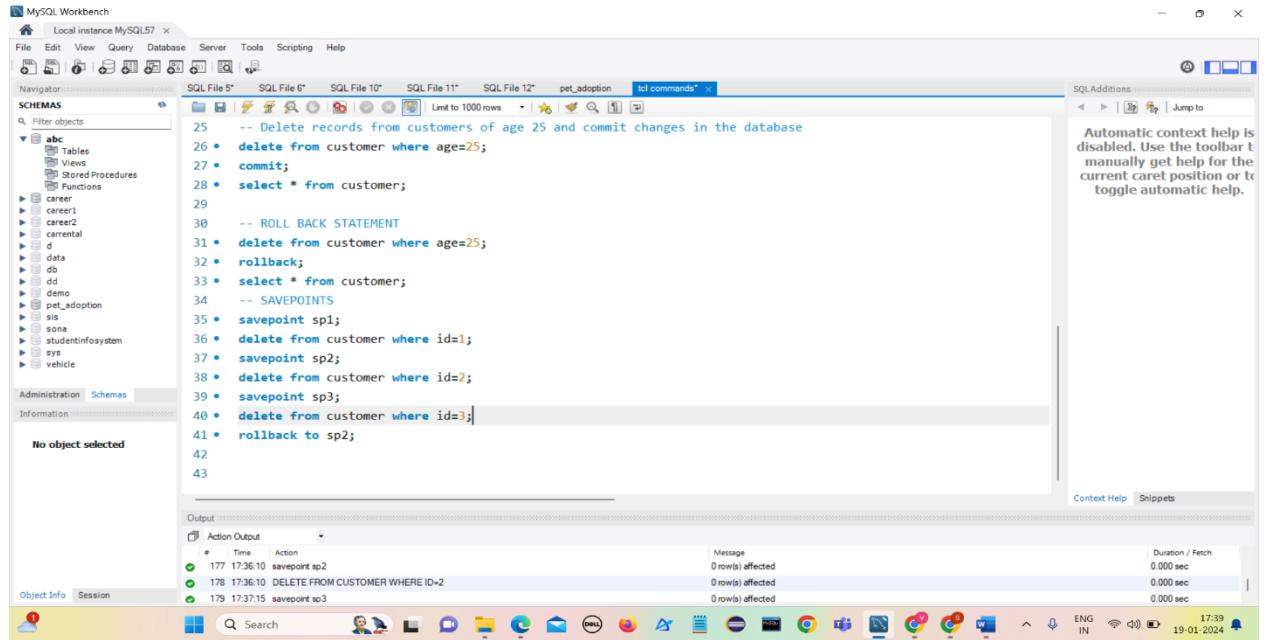
- Schemas:** abc
- SQL Editor:** pet_adoption tab
- Code:**

```
25 -- Delete records from customers of age 25 and commit changes in the database
26 • delete from customer where age=25;
27 • commit;
28 • select * from customer;
29
30 -- ROLL BACK STATEMENT
31 • delete from customer where age=25;
32 • rollback;
33 • select * from customer;
```
- Result Grid:** Shows a table with columns id, name, age, address, salary. Data:

id	name	age	address	salary
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00
- Output:** Action Output shows two log entries:
 - 166 17:29:13 insert into customer values(2, 'Khilan', 25, 'Delhi', 1500.00), (4, 'Chaitali', 25, 'Mumbai', 6500.00): 2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0
 - 167 17:29:25 select * from customer LIMIT 0, 1000: 7 row(s) returned
- System Bar:** Shows system icons, a search bar, and a taskbar with various application icons.

3) SAVEPOINT

A SAVEPOINT is a logical rollback point in a transaction.



The screenshot shows the MySQL Workbench interface with a transaction script in the SQL editor. The script includes a delete operation, a commit, a rollback, and three savepoints (sp1, sp2, sp3) used to mark points for potential rollbacks. The output pane shows the execution of these statements and their results.

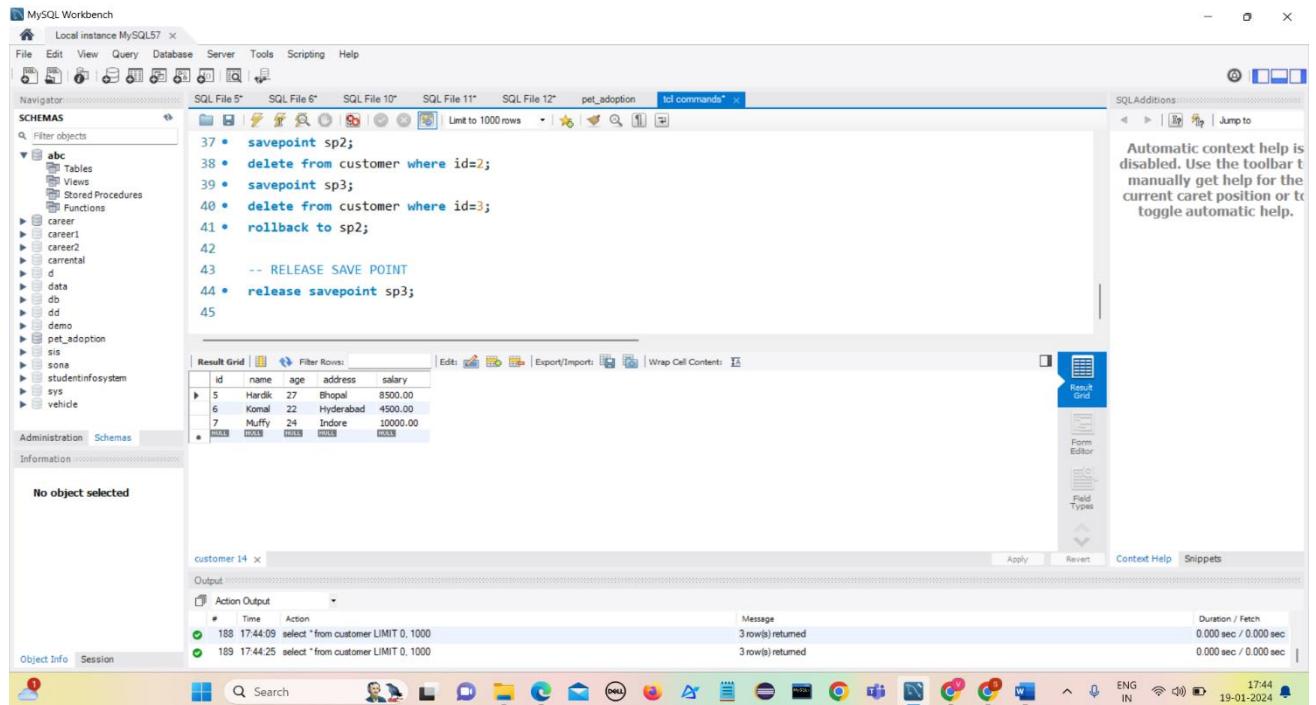
```
25 -- Delete records from customers of age 25 and commit changes in the database
26 • delete from customer where age=25;
27 • commit;
28 • select * from customer;
29
30 -- ROLL BACK STATEMENT
31 • delete from customer where age=25;
32 • rollback;
33 • select * from customer;
34 -- SAVEPOINTS
35 • savepoint sp1;
36 • delete from customer where id=1;
37 • savepoint sp2;
38 • delete from customer where id=2;
39 • savepoint sp3;
40 • delete from customer where id=3;
41 • rollback to sp2;
42
43
```

Output:

Action	Time	Message	Duration / Fetch
177 17:36:10	savepoint sp2	0 row(s) affected	0.000 sec
178 17:36:10	DELETE FROM CUSTOMER WHERE ID=2	0 row(s) affected	0.000 sec
179 17:37:15	savepoint sp3	0 row(s) affected	0.000 sec

4) RELEASE SAVEPOINT

The RELEASE SAVEPOINT command is used to remove an existing SAVEPOINT.



The screenshot shows the MySQL Workbench interface with a transaction script. It includes a savepoint (sp2), a delete operation, another savepoint (sp3), another delete operation, a rollback to sp2, and finally a release savepoint (sp3). The output pane shows the execution of these statements and their results, including a result grid showing customer data.

```
37 • savepoint sp2;
38 • delete from customer where id=2;
39 • savepoint sp3;
40 • delete from customer where id=3;
41 • rollback to sp2;
42
43 -- RELEASE SAVE POINT
44 • release savepoint sp3;
45
```

Result Grid:

id	name	age	address	salary
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Hyderabad	4500.00
7	Muffy	24	Indore	10000.00

Output:

Action	Time	Message	Duration / Fetch
188 17:44:09	select * from customer LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
189 17:44:25	select * from customer LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

18) JOINS

Join statement is used to combine data or rows from two or more tables based on a common field between them.

1) 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.

Create a table xyz and use the database to make it as the current working database.

The screenshot shows the MySQL Workbench interface. In the 'Schemas' tree on the left, under the 'abc' schema, there is a 'Tables' node which contains 'adoptions', 'animals', and 'shelter'. In the central query editor, the following SQL commands are run:

```
1 • create database xyz;
2 • use xyz;
```

The output window shows the results of these commands:

Action	Time	Message	Duration / Fetch
1 • create database xyz;	07:09:08	Query OK, 1 row affected (0.00 sec)	0.000 sec
2 • use xyz;	07:09:15	Database changed	0.000 sec
3 • select * from customer LIMIT 0, 1000	07:09:25	Query OK, 0 rows affected (0.00 sec)	0.000 sec
4 • select * from customer LIMIT 0, 1000	07:32:43	Query OK, 0 rows affected (0.00 sec)	0.000 sec
5 • use xyz;	07:32:53	Database changed	0.000 sec

Create two tables Student and Studentcourse using create table statement.

The screenshot shows the MySQL Workbench interface. In the 'Schemas' tree on the left, under the 'abc' schema, there is a 'Tables' node which contains 'adoptions', 'animals', and 'shelter'. In the central query editor, the following SQL commands are run:

```
3 -- create student table
4 • create table student
5 (
6   rollno int primary key,
7   name varchar(30),
8   address varchar(30),
9   phone char(10),
10  age int
11 );
12 • create table studentcourse
13 (
14   coursedid int primary key,
15   rollno int
16 );
```

The output window shows the results of these commands:

Action	Time	Message	Duration / Fetch
1 • use abc;	07:09:15	Query OK, 1 row affected (0.00 sec)	0.000 sec
2 • create database xyz;	07:09:25	Query OK, 1 row affected (0.00 sec)	0.000 sec
3 • use xyz;	07:32:43	Database changed	0.000 sec
4 • create table student (rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int)	07:32:53	Query OK, 0 rows affected (0.00 sec)	0.000 sec
5 • create table studentcourse (coursedid int primary key, rollno int)	07:38:37	Query OK, 0 rows affected (0.04 sec)	0.047 sec

Insert records into Student and Studentcourse table using insert statement.

```
15 rollno int
16 );
17 -- insert records into student table
18 • insert into student values(1,'ganga','Bodhan','7878787878',22),
19 (2,'Vinith','Nizamabad','1212121212',26),
20 (3,'Lehka','Hyderabad','6565656565',24),
21 (4,'Ram','pune','9898989898',19),
22 (5,'Mona','Mumbai','2121212121',10);
-- insert records into studentcourse table
24 • insert into studentcourse values(101,1),
25 (102,3),
26 (103,4),
27 (104,2),
28 (105,1);
```

Output:

#	Time	Action	Message	Duration / Fetch
4	07:32:43	create database xyz	1row(s) affected	0.000 sec
5	07:32:53	use xyz	0row(s) affected	0.000 sec
6	07:37:13	create table student (rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int)	0row(s) affected	0.047 sec
7	07:38:37	create table studentcourse (coursed int primary key, rollno int)	0row(s) affected	0.047 sec
8	07:42:37	insert into student values(1,'ganga','Bodhan','7878787878',22), (2,'Vinith','Nizamabad','1212121212',26), (3,'Lehka','Hyderabad','6565656565',24), (4,'Ram','pune','9898989898',19), (5,'Mona','Mumbai','2121212121',10)	5row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
9	07:43:45	insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)	5row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec

Retrieve records from student and studentcourse table

```
22 (5,'Mona','Mumbai','2121212121',10);
23 -- insert records into studentcourse table
24 • insert into studentcourse values(101,1),
25 (102,3),
26 (103,4),
27 (104,2),
28 (105,1);
29 • select * from student;
```

Result Grid:

rollno	name	address	phone	age
1	ganga	Bodhan	7878787878	22
2	Vinith	Nizamabad	1212121212	26
3	Lehka	Hyderabad	6565656565	24
4	Ram	pune	9898989898	19
5	Mona	Mumbai	2121212121	10

Output:

#	Time	Action	Message	Duration / Fetch
5	07:32:53	use xyz	0row(s) affected	0.000 sec
6	07:37:13	create table student (rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int)	0row(s) affected	0.047 sec
7	07:38:37	create table studentcourse (coursed int primary key, rollno int)	0row(s) affected	0.047 sec
8	07:42:37	insert into student values(1,'ganga','Bodhan','7878787878',22), (2,'Vinith','Nizamabad','1212121212',26), (3,'Lehka','Hyderabad','6565656565',24), (4,'Ram','pune','9898989898',19), (5,'Mona','Mumbai','2121212121',10)	5row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
9	07:43:45	insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)	5row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
10	07:45:14	select * from student LIMIT 0, 1000	5row(s)/returned	0.000 sec / 0.000 sec

The screenshot shows the MySQL Workbench interface with a local instance of MySQL57. In the Navigator pane, the schema 'pet_adoption' is selected, containing tables like 'student', 'course', and 'studentcourse'. A SQL editor tab named 'joins*' contains the following code:

```

23 -- insert records into studentcourse table
24 * insert into studentcourse values(101,1),
25 (102,3),
26 (103,4),
27 (104,2),
28 (105,1);
29 * select * from student;
30 * select * from studentcourse;

```

The Result Grid shows the data inserted into the 'studentcourse' table:

rollno	courseid
101	1
102	3
103	4
104	2
105	1

The Output pane displays the execution log:

Action	Time	Message	Duration / Fetch
create table student (rollno int primary key, name varchar(30), address varchar(30), phone char(10), age int)	6 07-37-13	0 row(s) affected	0.047 sec
create table studentcourse (coursed int primary key, rollno int)	7 07-38-37	0 row(s) affected	0.047 sec
insert into student values('ganga','Bodhan','7878787878',22, (2,'Virth','Nizamabad'))	8 07-42-37	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)	9 07-43-45	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
select * from student LIMIT 0, 1000	10 07-45-14	5 row(s) returned	0.000 sec / 0.000 sec
select * from studentcourse LIMIT 0, 1000	11 07-45-47	5 row(s) returned	0.000 sec / 0.000 sec

Inner join

The screenshot shows the MySQL Workbench interface with a local instance of MySQL57. In the Navigator pane, the schema 'pet_adoption' is selected. A SQL editor tab named 'joins*' contains the following code:

```

30 * select * from studentcourse;
31
32
33
34
35 -- Names and age of students enrolled in different courses
36 * select s.name , s.age from student s inner join studentcourse c
37 on s.rollno=c.rollno;

```

The Result Grid shows the names and ages of students enrolled in different courses:

name	age
ganga	22
Lekha	24
Ran	19
Winn	26
ganga	22

The Output pane displays the execution log:

Action	Time	Message	Duration / Fetch
create table studentcourse (coursed int primary key, rollno int)	7 07-38-37	0 row(s) affected	0.047 sec
insert into student values('ganga','Bodhan','7878787878',22, (2,'Virth','Nizamabad'))	8 07-42-37	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
insert into studentcourse values(101,1), (102,3), (103,4), (104,2), (105,1)	9 07-43-45	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
select * from student LIMIT 0, 1000	10 07-45-14	5 row(s) returned	0.000 sec / 0.000 sec
select * from studentcourse LIMIT 0, 1000	11 07-45-47	5 row(s) returned	0.000 sec / 0.000 sec
select s.name , s.age from student s inner join studentcourse c on s.rollno=c.rollno LIMIT 0, 1000	12 07-50-14	5 row(s) returned	0.000 sec / 0.000 sec

This query gives the names and age of students enrolled in different courses.

2) 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*.

The screenshot shows the MySQL Workbench interface with a query editor window titled "pet_adoption". The code is:

```

37     on s.rollno=c.rollno;
38
39
40
41 -- LEFT JOIN
42 * select s.name , s.age from student s
43 left join studentcourse c
44 on s.rollno=c.rollno;

```

The result grid shows the following data:

name	age
ganga	22
Lekha	24
Ram	19
Vrith	26
ganga	22
Mona	10

The status bar at the bottom right shows the date and time as 20-01-2024 08:24.

3) 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*

The screenshot shows the MySQL Workbench interface with a query editor window titled "pet_adoption". The code is:

```

42 * select s.name , s.age from student s
43 left join studentcourse c
44 on s.rollno=c.rollno;
45 -- RIGHT JOIN
46 * select s.name , s.age from student s
47 right join studentcourse c
48 on s.rollno=c.rollno;
49
50

```

The result grid shows the following data:

name	age
ganga	22
Lekha	24
Ram	19
Vrith	26
ganga	22

The status bar at the bottom right shows the date and time as 20-01-2024 08:26.

Vuthur Sriganga
Data Engineering Batch-1