

## Rule 1: The Information Rule:

```
SELECT * FROM animals WHERE Owner_id = 10;
```

The SQL statement "SELECT \* FROM animals WHERE Owner\_id = 10" retrieves all data from the "animals" table where the "Owner\_id" column matches the value of 10.

This statement is proof that data can be retrieved by matching identifiers in SQL. In this case, the identifier being used is the "Owner\_id" column. By specifying a value of 10 for this column, the statement instructs the database to retrieve all data from the "animals" table that is associated with the owner whose ID is 10.

This statement also demonstrates the use of a simple SELECT statement to retrieve data without reference to the underlying file system. The statement does not require any knowledge of the physical location or structure of the file system where the data is stored. Instead, it relies solely on the identifier (i.e. the "Owner\_id" column) to locate and retrieve the relevant data.

Overall, the statement "SELECT \* FROM animals WHERE Owner\_id = 10" is a clear example of how data can be retrieved in SQL by matching identifiers without reference to the underlying file system.

Before:

	Animal_id	Animal_type	Animal_breed	Animal_name	Animal_age	Owner_id
<input type="checkbox"/> Edit Copy Delete	1000	dog	Dobermann	Tor	6	10
<input type="checkbox"/> Edit Copy Delete	1001	dog	Dobermann	Alpha	4	10
<input type="checkbox"/> Edit Copy Delete	1002	dog	Dobermann	Ennis	1	10
<input type="checkbox"/> Edit Copy Delete	1003	dog	Maltipoo	Glory	6	1
<input type="checkbox"/> Edit Copy Delete	1008	sheep	Cheviot Sheep	Flaffy	2	6
<input type="checkbox"/> Edit Copy Delete	1091	Donkey	Irish donkey	Donny	5	7
<input type="checkbox"/> Edit Copy Delete	1111	Donkey	Irish donkey	Lilly	4	7
<input type="checkbox"/> Edit Copy Delete	1120	Horse	Thoroughbred Horse	Spirit	7	8
<input type="checkbox"/> Edit Copy Delete	1121	Donkey	Irish donkey	Hope	2	7
<input type="checkbox"/> Edit Copy Delete	1221	Horse	Thoroughbred Horse	Dakota	4	8
<input type="checkbox"/> Edit Copy Delete	1223	dog	American Bulldog	Jackson	4	5
<input type="checkbox"/> Edit Copy Delete	1870	cat	Sphynx cat	Archibald	7	6
<input type="checkbox"/> Edit Copy Delete	1875	cat	Sphynx cat	Amanda	8	6
<input type="checkbox"/> Edit Copy Delete	1953	dog	Maltipoo	Nency	4	4
<input type="checkbox"/> Edit Copy Delete	1977	sheep	Cheviot Sheep	Nick	2	6
<input type="checkbox"/> Edit Copy Delete	1978	sheep	Cheviot Sheep	Only	2	6
<input type="checkbox"/> Edit Copy Delete	1995	cat	Maine Coon	King	10	9
<input type="checkbox"/> Edit Copy Delete	2005	cat	Randall	Barbie	4	10

After:

	Animal_id	Animal_type	Animal_breed	Animal_name	Animal_age	Owner_id
<input type="checkbox"/> Edit Copy Delete	1000	dog	Dobermann	Tor	6	10
<input type="checkbox"/> Edit Copy Delete	1001	dog	Dobermann	Alpha	4	10
<input type="checkbox"/> Edit Copy Delete	1002	dog	Dobermann	Ennis	1	10
<input type="checkbox"/> Edit Copy Delete	2005	cat	Ragdoll	Barbie	4	10

## Rule 2: The Guaranteed Access Rule:

```
SELECT a.Animal_type, o.Owner_name FROM animals a JOIN animalowners o ON o.Owner_id = a.Owner_id WHERE a.Owner_id = 6;
```

The Guaranteed Access Rule in SQL states that data from a table can be accessed down to the field level using a combination of the table name, primary key, and column name.

The SQL query "SELECT a.Animal\_type, o.Owner\_name FROM animals a JOIN animalowners o ON o.Owner\_id = a.Owner\_id WHERE a.Owner\_id = 6;" demonstrates this rule in the following ways:

1. The query specifies the table name and column names: The SELECT statement in the query specifies that we want to retrieve the "Animal\_type" column from the "animals" table and the "Owner\_name" column from the "animalowners" table. This is an example of using the table name and column name to access specific fields of data.
2. The query uses the primary key to join tables: The JOIN statement in the query uses the "Owner\_id" column from both tables to join the "animals" table with the "animalowners" table. This is an example of using the primary key to link related data from different tables.
3. The query retrieves unique values: The WHERE clause in the query specifies that we want to retrieve data only for the owner with an ID of 6. This is an example of using the primary key to retrieve unique values from the table.

Overall, this SQL query demonstrates the Guaranteed Access Rule by accessing specific fields of data, linking related data using the primary key, and retrieving unique values from the table using a simple SELECT statement.

Before:

Owner_id	Owner_name	Owner_email	Owner_phone
1	Nick Williams	n.williams98@gmail.com	833134415
2	Olesia Naumenko	olesia.naumenko@gmail.com	834756755
3	Carol Nordman	carolnordman45@gmail.com	833334455
4	Audrey O'Halloran	audrey.oh@gmail.com	833134495
5	John Potter	john.potter@gmail.com	831134691
6	Tommas Brown	tom_crazybear@gmail.com	833134422

Animal_id	Animal_type	Animal_breed	Animal_name	Animal_age	Owner_id
1000	dog	Dobermann	Tor	6	10
1001	dog	Dobermann	Alpha	4	10
1002	dog	Dobermann	Ennis	1	10
1003	dog	Maltipoo	Glory	6	1
1008	sheep	Cheviot Sheep	Flaffy	2	6
1091	Donkey	Irish donkey	Donny	5	7
1111	Donkey	Irish donkey	Lilly	4	7
1120	Horse	Thoroughbred Horse	Spirit	7	8

After:

Animal_type	Owner_name
sheep	Tommas Brown
cat	Tommas Brown
cat	Tommas Brown
sheep	Tommas Brown
sheep	Tommas Brown
sheep	Tommas Brown
Donkey	Tommas Brown
sheep	Tommas Brown
sheep	Tommas Brown

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Console

```
>SELECT a.Animal_type, o.Owner_name FROM animals a JOIN animalowners o ON o.Owner_id = a.Owner_id WHERE a.Owner_id = 6;
```

Rule 3: Systematic Treatment of Null Values:

```
SELECT Payment_id, Payment_method, COALESCE(Billing_date, 'N/A') AS Billing_date, COALESCE(Installment_ament, 'N/A') AS Installment_amount FROM payments;
```

The SQL query "Select Payment\_id, Payment\_method, COALESCE(Billing\_date, 'N/A') AS Billing\_date, COALESCE(Installment\_amount, 'N/A') AS Installment\_amount FROM payments;" demonstrates the Systematic Treatment of Null Values by using the COALESCE function to replace NULL values with the string "N/A" in the "Billing\_date" and "Installment\_amount" columns of the "payments" table.

This query ensures that any NULL values in the "Billing\_date" and "Installment\_amount" columns are treated consistently and represented in the output in the same way as other non-NULL values. By using COALESCE to replace NULL values with "N/A", we are treating both data types (date and numeric) where NULL appears in the same way.

In summary, the query demonstrates the Systematic Treatment of Null Values by using a consistent approach to handle NULL values across different columns in the "payments" table.

Before:

		Payment_id	Billing_date	Billing_amount	Installment_amount	Payment_method	Appointment_id	Payment_status
<input type="checkbox"/>	Edit  Copy  Delete	102	2023-03-21 16:00:00	500	300	Apple Pay	1012	Done
<input type="checkbox"/>	Edit  Copy  Delete	103	2022-03-21 16:00:00	1000	600	MasterCard	1013	In progress
<input type="checkbox"/>	Edit  Copy  Delete	104	2022-05-05 11:00:00	100	0	Cash	1014	Done
<input type="checkbox"/>	Edit  Copy  Delete	105	2022-06-15 11:00:00	1000	100	Cash	1015	Done
<input type="checkbox"/>	Edit  Copy  Delete	106	2022-07-05 11:00:00	100	0	Apple pay	1016	Done
<input type="checkbox"/>	Edit  Copy  Delete	107	2022-08-05 11:00:00	750	0	MasterCard	1017	Done
<input type="checkbox"/>	Edit  Copy  Delete	108	2022-09-05 11:00:00	420	0	Cash	1018	Done
<input type="checkbox"/>	Edit  Copy  Delete	109	2022-10-05 11:00:00	370	0	Apple pay	1019	Done
<input type="checkbox"/>	Edit  Copy  Delete	110	2022-11-05 11:00:00	480	0	MasterCard	1022	Done
<input type="checkbox"/>	Edit  Copy  Delete	111	2022-12-05 11:00:00	900	0	Cash	1032	Done
<input type="checkbox"/>	Edit  Copy  Delete	112	2023-01-05 11:00:00	1004	0	Apple pay	1042	Done
<input type="checkbox"/>	Edit  Copy  Delete	113	2023-02-05 11:00:00	540	0	MasterCard	1052	Done
<input type="checkbox"/>	Edit  Copy  Delete	114	2023-04-05 11:00:00	300	0	Revolut	1053	Done
<input type="checkbox"/>	Edit  Copy  Delete	115	2023-05-05 11:00:00	100	0	Apple pay	1054	Done
<input type="checkbox"/>	Edit  Copy  Delete	116	2023-06-05 11:00:00	100	0	Cash	1055	Done
<input type="checkbox"/>	Edit  Copy  Delete	117	NULL	200	NULL	Cash	1056	In progress

After:

		Payment_id	Payment_method	Billing_date	Installment_amount
<input type="checkbox"/>	Edit  Copy  Delete	103	MasterCard	2022-03-21 16:00:00	600
<input type="checkbox"/>	Edit  Copy  Delete	104	Cash	2022-05-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	105	Cash	2022-06-15 11:00:00	100
<input type="checkbox"/>	Edit  Copy  Delete	106	Apple pay	2022-07-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	107	MasterCard	2022-08-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	108	Cash	2022-09-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	109	Apple pay	2022-10-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	110	MasterCard	2022-11-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	111	Cash	2022-12-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	112	Apple pay	2023-01-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	113	MasterCard	2023-02-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	114	Revolut	2023-04-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	115	Apple pay	2023-05-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	116	Cash	2023-06-05 11:00:00	0
<input type="checkbox"/>	Edit  Copy  Delete	117	Cash	N/A	N/A

#### Rule 4: Dynamic Online Catalog based on the relational model:

```
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH FROM INFORMATION_SCHEMA.COLUMNS WHERE  
TABLE_SCHEMA = 'veterinaryclinic' AND TABLE_NAME = 'animals';
```

This query demonstrates the concept of a dynamic online catalog based on the relational model in a few ways:

1. It uses the "INFORMATION\_SCHEMA.COLUMNS" table, which is a special table that stores metadata about the columns in all tables in the current database. This demonstrates that the meta data about the tables and their columns is stored in a separate table within the database.
2. The query selects specific columns from the INFORMATION\_SCHEMA.COLUMNS table, including TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE, and CHARACTER\_MAXIMUM\_LENGTH. This shows that the meta data stored in the INFORMATION\_SCHEMA can be used to retrieve information about the structure of the tables and the data types of their columns.
3. The "WHERE" clause filters the results to only show columns for the "animals" table in the "veterinaryclinic" database. This demonstrates that the metadata in the INFORMATION\_SCHEMA is specific to the database and can be used to retrieve information about individual tables within the database.

Overall, the query demonstrates how the INFORMATION\_SCHEMA can be used to access metadata about tables and their columns in a relational database, which supports the concept of a dynamic online catalog. The metadata in the INFORMATION\_SCHEMA allows for the structure of the database to be queried and analyzed dynamically, without having to rely on hard-coded information about the tables and their columns.

TABLE_NAME	COLUMN_NAME	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
animals	Animal_id	int	NULL
animals	Animal_type	varchar	100
animals	Animal_breed	varchar	100
animals	Animal_name	varchar	100
animals	Animal_age	int	NULL
animals	Owner_id	int	NULL

Rule 5: The Comprehensive Data Sub Language Rule: Every DBMS needs a language to modify data structures and query them; in the vast majority of cases it is SQL so some create statements for DDL and a number of queries for DML.

*DDL statements:*

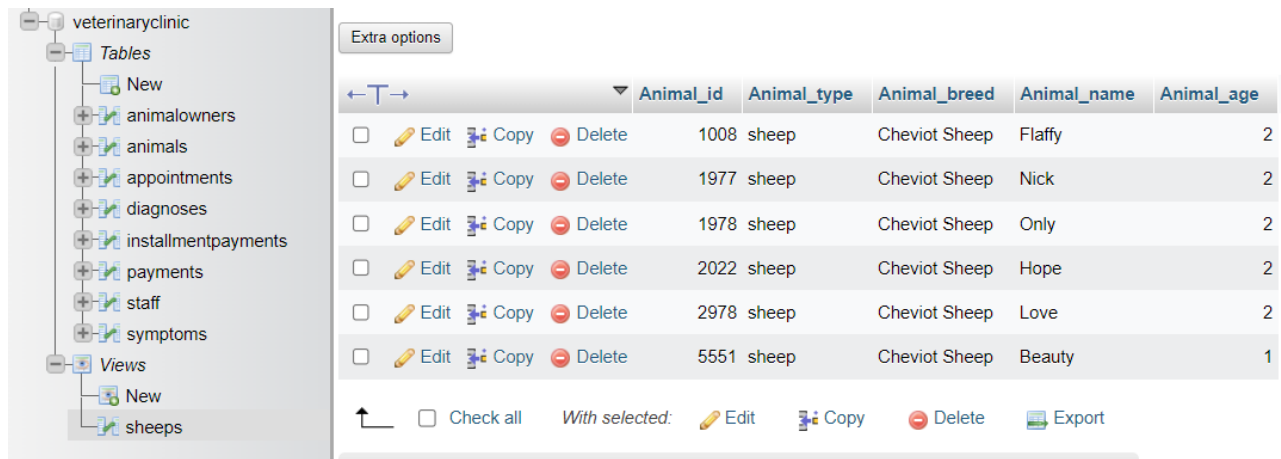
```
CREATE TABLE Staff ( Staff_id int, Name varchar(100), Position varchar(100), Phone int, PRIMARY KEY (Staff_id));  
ALTER TABLE diagnoses ADD FOREIGN KEY (Appointment_id) REFERENCES Appointments(Appointment_id);  
DROP TABLE nstaff;
```

*DML statements:*

```
SELECT * FROM `veterinaryclinic`.`staff` WHERE `Staff_id` = 71  
INSERT INTO staff (Staff_id, Name, Position, Phone) VALUES ('21', 'Mike Anderson', 'nurse', '0836275660');  
UPDATE payments SET Payment_method = "Cash" WHERE Payment_id = "114";  
DELETE FROM appointments WHERE Appointment_id="1062";
```

## Rule 6: The View Updating Rule:

```
CREATE VIEW sheeps AS SELECT Animal_id, Animal_type, Animal_breed, Animal_name, Animal_age FROM animals WHERE Animal_type = 'Sheep';
```

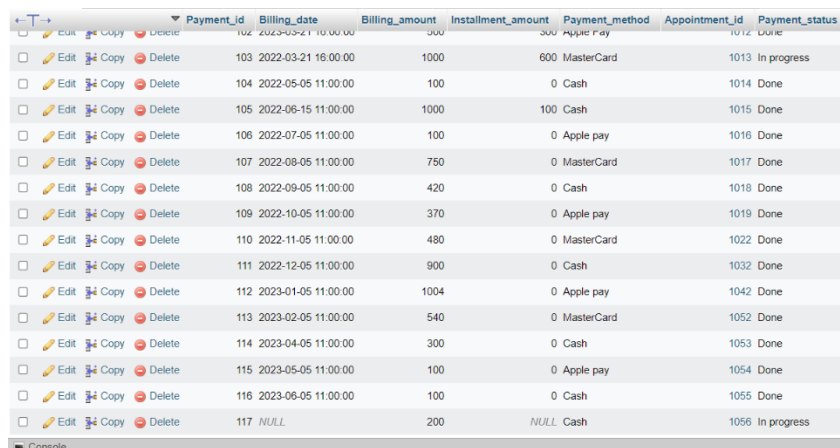


	Animal_id	Animal_type	Animal_breed	Animal_name	Animal_age
<input type="checkbox"/> Edit Copy Delete	1008	sheep	Cheviot Sheep	Flaffy	2
<input type="checkbox"/> Edit Copy Delete	1977	sheep	Cheviot Sheep	Nick	2
<input type="checkbox"/> Edit Copy Delete	1978	sheep	Cheviot Sheep	Only	2
<input type="checkbox"/> Edit Copy Delete	2022	sheep	Cheviot Sheep	Hope	2
<input type="checkbox"/> Edit Copy Delete	2978	sheep	Cheviot Sheep	Love	2
<input type="checkbox"/> Edit Copy Delete	5551	sheep	Cheviot Sheep	Beauty	1

## Rule 7: High Level Insert Update and Delete Rule:

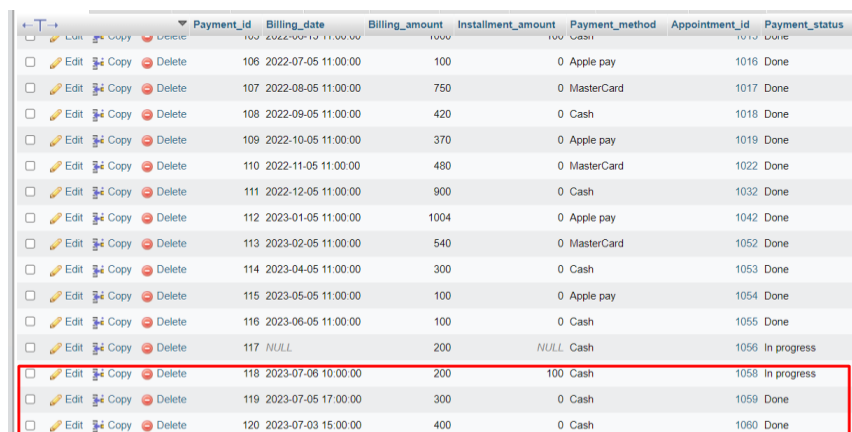
```
INSERT INTO payments VALUES  
( '118', '2023-07-06 10:00:00', '200', '100', 'Cash', '1058', 'In progress'),  
( '119', '2023-07-05 17:00:00', '300', '0', 'Cash', '1059', 'Done'),  
( '120', '2023-07-03 15:00:00', '400', '0', 'Cash', '1060', 'Done');
```

Before:



Payment_id	Billing_date	Billing_amount	Installment_amount	Payment_method	Appointment_id	Payment_status
102	2023-03-21 10:00:00	500	500	Apple pay	1012	Done
103	2022-03-21 16:00:00	1000	600	MasterCard	1013	In progress
104	2022-05-05 11:00:00	100	0	Cash	1014	Done
105	2022-06-15 11:00:00	1000	100	Cash	1015	Done
106	2022-07-05 11:00:00	100	0	Apple pay	1016	Done
107	2022-08-05 11:00:00	750	0	MasterCard	1017	Done
108	2022-09-05 11:00:00	420	0	Cash	1018	Done
109	2022-10-05 11:00:00	370	0	Apple pay	1019	Done
110	2022-11-05 11:00:00	480	0	MasterCard	1022	Done
111	2022-12-05 11:00:00	900	0	Cash	1032	Done
112	2023-01-05 11:00:00	1004	0	Apple pay	1042	Done
113	2023-02-05 11:00:00	540	0	MasterCard	1052	Done
114	2023-04-05 11:00:00	300	0	Cash	1053	Done
115	2023-05-05 11:00:00	100	0	Apple pay	1054	Done
116	2023-06-05 11:00:00	100	0	Cash	1055	Done
117	NULL	200	NULL	Cash	1056	In progress

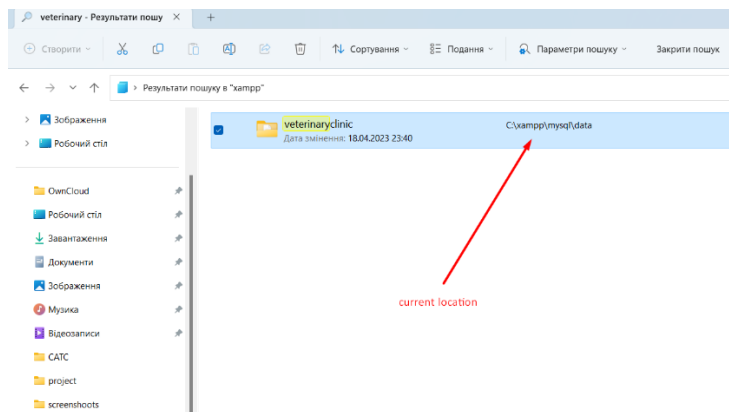
After:



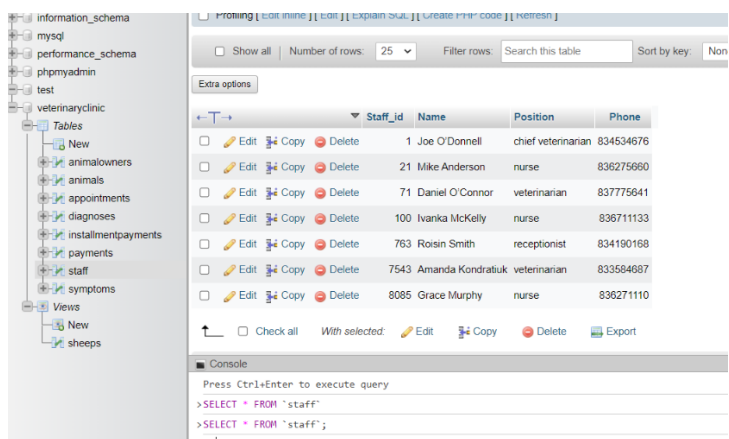
Payment_id	Billing_date	Billing_amount	Installment_amount	Payment_method	Appointment_id	Payment_status
102	2023-03-21 10:00:00	500	500	Apple pay	1012	Done
106	2022-07-05 11:00:00	100	0	Apple pay	1016	Done
107	2022-08-05 11:00:00	750	0	MasterCard	1017	Done
108	2022-09-05 11:00:00	420	0	Cash	1018	Done
109	2022-10-05 11:00:00	370	0	Apple pay	1019	Done
110	2022-11-05 11:00:00	480	0	MasterCard	1022	Done
111	2022-12-05 11:00:00	900	0	Cash	1032	Done
112	2023-01-05 11:00:00	1004	0	Apple pay	1042	Done
113	2023-02-05 11:00:00	540	0	MasterCard	1052	Done
114	2023-04-05 11:00:00	300	0	Cash	1053	Done
115	2023-05-05 11:00:00	100	0	Apple pay	1054	Done
116	2023-06-05 11:00:00	100	0	Cash	1055	Done
117	NULL	200	NULL	Cash	1056	In progress
118	2023-07-06 10:00:00	200	100	Cash	1058	In progress
119	2023-07-05 17:00:00	300	0	Cash	1059	Done
120	2023-07-03 15:00:00	400	0	Cash	1060	Done

## Rule 8: Physical Data Independence:

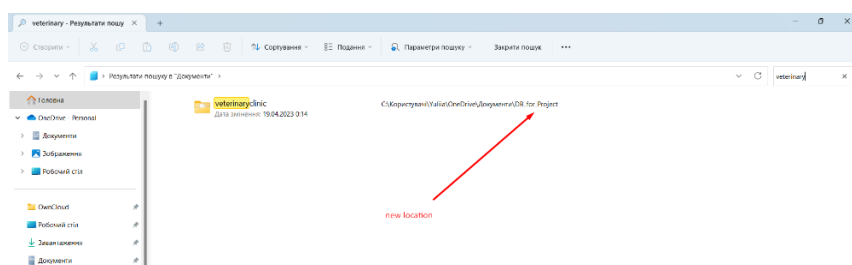
### Current Location of My DataBase:



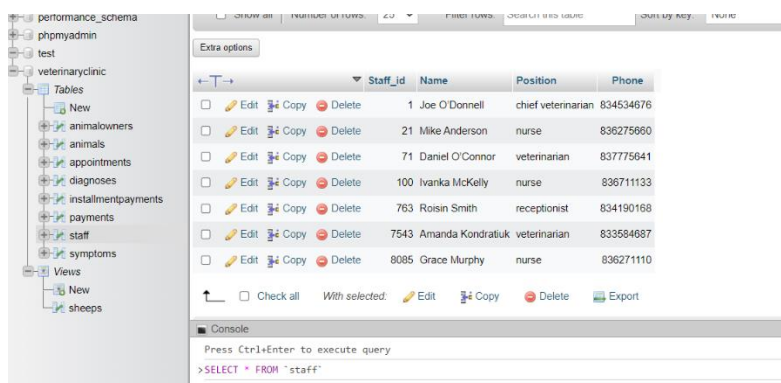
`SELECT * FROM `veterinaryclinic`.`staff` WHERE `Staff_id` = 71` works:



### New Location:



### Still works:



## Rule 9: Logical Data Independence:

We have a table called "Staff" with the following structure:

```
CREATE TABLE Staff (  
    Staff_id int,  
    Name varchar(100),  
    Position varchar(100),  
    Phone int,  
    PRIMARY KEY (Staff_id)  
);
```

Now let's say we want to add a new column to this table called "Email". We can do this using the following ALTER TABLE statement:

```
ALTER TABLE Staff ADD Email varchar(100);
```

Now we can run a SELECT query against the original columns:

```
SELECT Staff_id, Name, Position, Phone FROM Staff;
```

And we can also run a SELECT query against the original columns plus the new "Email" column:

```
SELECT Staff_id, Name, Position, Phone, Email FROM Staff;
```

The screenshot displays a database management interface. On the left, a tree view shows the database structure, including tables like 'animalowners', 'animals', 'appointments', 'diagnoses', 'installmentpayments', 'payments', 'staff', and 'symptoms'. The 'staff' table is selected. The main area shows the 'Staff' table with columns: Staff\_id, Name, Position, Phone, and Email. The data is as follows:

Staff_id	Name	Position	Phone	Email
1	Joe O'Donnell	chief veterinarian	834534676	NULL
21	Mike Anderson	nurse	836275660	NULL
71	Daniel O'Connor	veterinarian	837775641	NULL
100	Ivanka McKelly	nurse	836711133	NULL
763	Roisin Smith	receptionist	834190168	NULL
7543	Amanda Kondratiuk	veterinarian	833584687	NULL
8085	Grace Murphy	nurse	836271110	NULL

Below the table, a console window shows the following SQL queries:

```
> SELECT Staff_id, Name, Position, Phone FROM Staff;  
= SELECT Staff_id, Name, Position, Phone, Email FROM Staff;  
> SELECT * FROM `staff`  
> ALTER TABLE Staff ADD Email varchar(100);  
> SELECT Staff_id, Name, Position, Phone, Email FROM Staff;
```



## Rule 10: Integrity Independence:

To this Rule, I created two tables with a primary key-foreign key relationship, and then execute a query that uses the foreign key to retrieve data from the related table.

```
CREATE TABLE Staff ( Staff_id int, Name varchar(100), Position varchar(100), Phone int, PRIMARY KEY (Staff_id));
```

```
CREATE TABLE Appointments (
    Appointment_id int,
    Appointment_type varchar(100),
    Appointment_time datetime,
    Animal_id int,
    Symptom_id int,
    Staff_id int,
    PRIMARY KEY (Appointment_id)
);
```

```
INSERT INTO staff (Staff_id, Name, Position, Phone) VALUES ('763', 'Roisin Smith', 'receptionist', '0834190168');
```

```
INSERT INTO appointments (Appointment_id, Appointment_type, Appointment_time, Animal_id, Symptom_id, Staff_id ) VALUES ('1010', 'online', '2022-02-11 12:00:00', '1000', '1', '71');
```

In this example, we have two tables, Staff and Appointments, with a primary key-foreign key relationship between them. The Staff table has a primary key column Staff\_id, and the Appointments table has a foreign key column Staff\_id that references the Staff table.

I can then execute a query that retrieves the appointment information for a specific staff\_id:

```
SELECT Appointment_time, Staff_id FROM appointments WHERE Staff_id = 1;
```

The screenshot shows a database management tool interface. On the left is a tree view of the database structure, including a 'veterinaryclinic' database with tables like 'appointments', 'staff', and 'symptoms'. The main area displays the results of a query in a table with two columns: 'Appointment\_time' and 'Staff\_id'. The results show 11 rows of data, all with 'Staff\_id' equal to 1. At the bottom, a console window shows the executed query: 'SELECT Appointment\_time, Staff\_id FROM appointments WHERE Staff\_id = 1;'. The console also includes a prompt for the next command and a status bar indicating 'Press Ctrl+Enter to execute query'.

	Appointment_time	Staff_id
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2022-03-22 11:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2022-06-22 12:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2023-05-12 11:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2022-03-22 12:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2023-05-12 13:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2022-06-22 14:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2023-05-12 15:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2021-03-22 11:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2023-02-11 12:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2022-03-11 12:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2023-06-23 12:00:00	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2022-09-11 12:00:00	1

```
> SELECT Appointment_time, Staff_id FROM appointments WHERE Staff_id = 1;
> |
```



### Rule 11: Distributed Independence:

Suppose that the database is distributed across two physical locations, with the staff table stored on one server and the appointments table stored on another server. Despite this physical separation, queries should still work in the same way. For example, suppose we want to retrieve a list of appointments for a specific staff member:

```
SELECT * FROM appointments WHERE staff_id = 7543;
```

This query should work regardless of whether the appointments table is stored on the same server as the staff table or on a different server. The primary/foreign key relationship between the two tables ensures that the query will return accurate results even if the data is distributed across multiple locations.

### Rule 12 Non Subversion Rule:

The Non-Subversion Rule in this database would mean that data values cannot be changed by code other than SQL. This is important in preventing hacking and injection of SQL from sources other than the DBMS. One example of this in the VeterinaryClinic database could be ensuring that only authorized users have access to modify data, and implementing security measures to prevent unauthorized access.