KPGR111 – Databases, Exam 2022

File attached : scheme_data_petClinic.sql, queries.sql, petClinic_solution.pdf, ERDiargram.mwb

1. Please read the PetClinic scenario above and address the following questions:

Problem description
Follow the given scenario (case) about PetClinic. Pet clinic has veterinarians, animals and pet's owners. Pet's owner visit the Pet Clinic to get pet treatment from veterinarians by registering appointments with pet and owner details. Pets clinics have to receive an appointment information date, time, room available,veterinarians name and type of treatment based.

a. Identify the entities that may be required for designing a relational database for PetClinic.

Solution:

Entities, the following are the entities of PetClinic:

Veterinarians: *vet*

Rooms: *room*

Times_slot: *timeslot*

Animal type: *specie*

Veterinarians Specialization: *vet_specialization*

address: *address*

Treatment type: *treatment*

Booking: *booking*

Pet's Owners: *owner*

Pets: *pet*

Appointments: *appointment*

"address, room, times_lot, and booking entities" are a table in the normalization form to remove redundancy in the PetClinic database.
"vet_specialization" to identify that the Veterinarians in pet clinics only treat animals within the specialize only in different types of animals.
Total of tables, 11.

b. Identify and list attributes associated with the relations identified in a.

Solution

Attributes, the following are the attributes of PetClinic:

vet (*vet_id, vet_name, vet_tele, vet_address*)

room (*room_name*)

timeslot (*timeslot_id, start_time, end_time*)

specie (*specie_id,specie_name*)

vet_specialization(*id, vet_id,specie_id*)

address (address_id, street, city, zip_code)

treatment (*treatment_id, treatment_no, treatment_name,price,specie_id*)

booking (*booking_id, owner_id, pet_id, booking_type, booking_on*)

owner (*owner_id, first_name, last_name, date_of_birth, address_id, owber_tele, email*)

pet (*pet_id, pet_name, owner_id, time_of_birth, specie_id*)

appointment (*apt_id, booking_id, date_of_birth, apt_note, room_name, vet_id, treatment_id, timeslot_id*)

c. Identify Candidate keys, Primary Keys and Foreign keys.

vet (***vet_id*** ***{PK}****{ Candidate Key }, vet_name,* ***vet_tele {Candidate Key}****, vet_address*)

room (***room_name {PK}*** *{Candidate Key}*)

timeslot (***timeslot_id {PK}*** *{Candidate Key},* ***start_time*** *{Candidate Key}, end_time****{Candidate Key}***)

specie (***specie_id {PK}*** *{Candidate Key} ,****specie_name {Candidate Key}***)

vet_specialization (***id {PK}****, vet_id {FK}, specie_id {FK}*)

treatment (***treatment_id {PK}*** *{Candidate Key}, treatment_no , treatment_name, price,specie_id*)

address (***address_id {PK}****, street, city, zip_code*)

owner (***owner_id {PK}*** *{ Candidate Key}, first_name, last_name, date_of_birth, address_id{FK}, owner_tele{ Candidate Key},* ***email {Candidate Key})***

pet (***pet_id {PK}*** *{Candidate Key}, pet_name,* ***owner_id {PK}*** *{Candidate Key}, time_of_birth, specie_id{FK}*)

booking (***booking_id {PK}*** *{Candidate Key}, pet_id {FK}, booking_type, booking_on*)

appointment (*apt_id) ,* ***booking_id {PK},*** *{Candidate Key}, date_of_birth, apt_note, room_name {FK}, vet_id {FK} , treatment_type_id {FK} , timeslot_id {FK}*)

Additional clarification: Here the **treatment entity** is a weak entity. It does have its own primary key. As I give booking_id from the booking table as the primary key (**treatment entity**) and identify more key attributes to others relationships. In order to allow a pet owner to bring one or more animals to an appointment. Therefore when the booking was set, one booking_id just for one animal. And the pet who has the same owner will be set to the same appointment_id. This can be easy to track which appointments that owner brings to more than one animal.

d. Design an ER model and define relationships between entities.

Solution

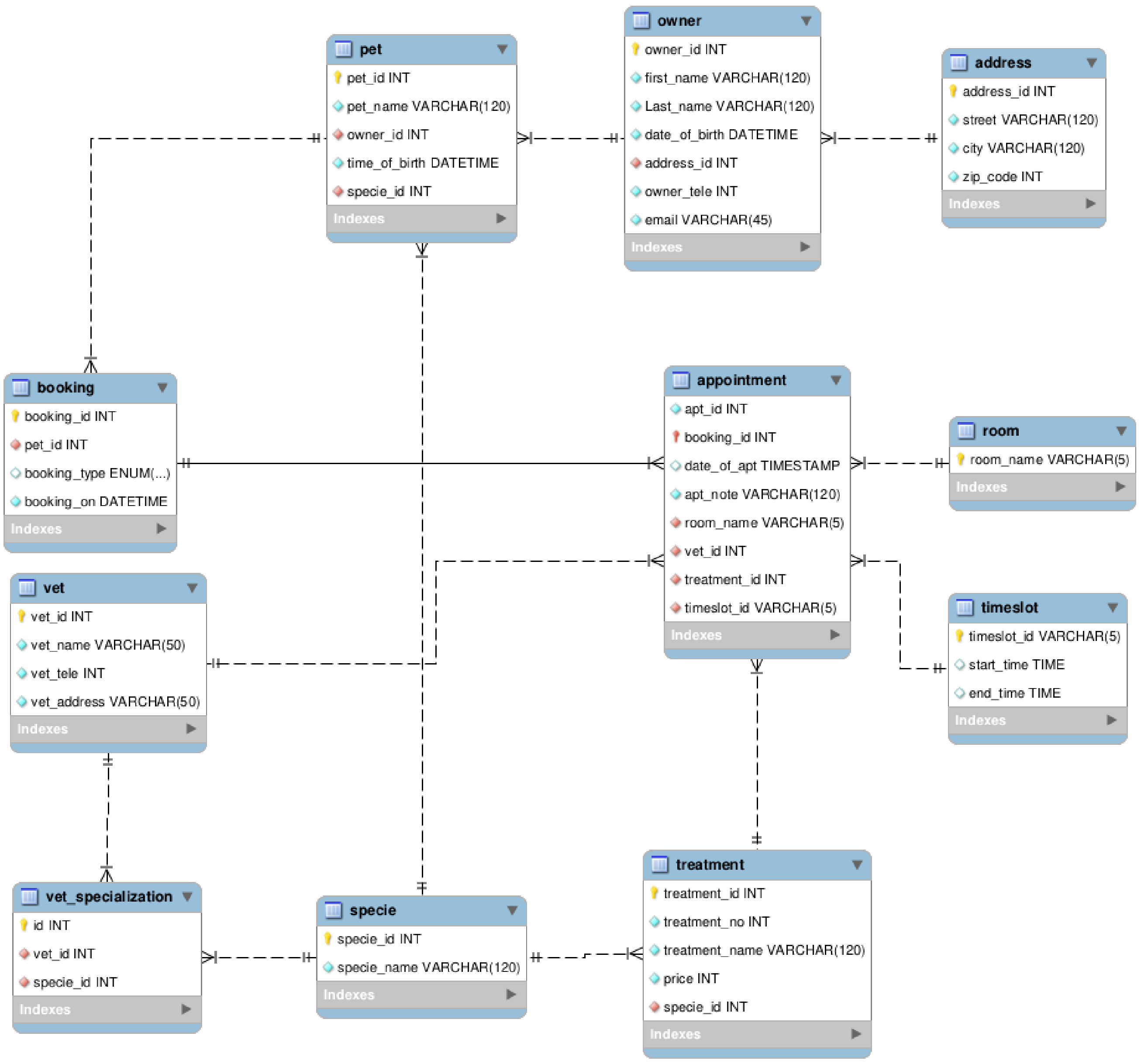


Figure d. PetClinic ER Diagram

e. Create an empty Database PetClinic and implement the database according to the design presented in ER Model (d) Note: Please make suitable assumptions for any missing requirements in the scenario.

DDL

```
DROP SCHEMA IF EXISTS PetClinic;
CREATE SCHEMA PetClinic;
use PetClinic;

CREATE TABLE vet
(vet_id          INT NOT NULL,
vet_name         VARCHAR(50) NOT NULL,
vet_tele         INT(10) NOT NULL,
vet_address      VARCHAR(50) NOT NULL,
CONSTRAINT vetPK PRIMARY KEY (vet_id));


CREATE TABLE room
(room_name       VARCHAR(5) NOT NULL,
CONSTRAINT roomPK PRIMARY KEY (room_name));


CREATE TABLE timeslot
(timeslot_id     VARCHAR(5) NOT NULL,
start_time       TIME,
end_time         TIME,
CONSTRAINT timeslotPK PRIMARY KEY (timeslot_id));


CREATE TABLE specie
(specie_id       INT NOT NULL,
specie_name      VARCHAR(120) NOT NULL,
CONSTRAINT specie PRIMARY KEY (specie_id));


CREATE TABLE vet_specialization
(id              INT NOT NULL,
vet_id           INT NOT NULL,
specie_id        INT NOT NULL,
CONSTRAINT vet_specializationPK PRIMARY KEY (id),
CONSTRAINT specieVets_IdFk FOREIGN KEY (vet_id) REFERENCES vet(vet_id),
CONSTRAINT specie_IdFk FOREIGN KEY(specie_id) REFERENCES specie(specie_id));


CREATE TABLE treatment(
treatment_id INT NOT NULL,
treatment_no INT NOT NULL,
treatment_name VARCHAR(120) NOT NULL,
price INT NOT NULL,
specie_id INT NOT NULL,
CONSTRAINT treatmentPK PRIMARY KEY (treatment_id),
CONSTRAINT TreatmentSpecies_IdFk FOREIGN KEY(specie_id) REFERENCES specie(specie_id)
);


CREATE TABLE address
(address_id      INT NOT NULL,
street           VARCHAR(120) NOT NULL,
city             VARCHAR(120) NOT NULL,
zip_code         INT NOT NULL,
CONSTRAINT address_IdPK PRIMARY KEY (address_id));


CREATE TABLE owner
(owner_id        INT NOT NULL,
first_name       VARCHAR(120) NOT NULL,
Last_name        VARCHAR(120) NOT NULL,
date_of_birth    DATETIME NOT NULL,
address_id       INT NOT NULL,
owner_tele       INT(10) NOT NULL,
email            VARCHAR(45) NOT NULL,
CONSTRAINT Owner_IdPk PRIMARY KEY (owner_Id),
CONSTRAINT Owner_address_IdFk FOREIGN KEY(address_id) REFERENCES address(address_id));


CREATE TABLE pet
(pet_id          INT NOT NULL,
pet_name         VARCHAR(120) NOT NULL,
owner_id         INT NOT NULL,
time_of_birth    DATETIME NOT NULL,
specie_id INT    NOT NULL,
CONSTRAINT pets_Id_Pk PRIMARY KEY (pet_id),
CONSTRAINT petsSpecies_Id FOREIGN KEY (specie_id) REFERENCES specie(specie_id),
CONSTRAINT petsOwner_Id FOREIGN KEY (owner_id) REFERENCES owner(owner_id));

CREATE TABLE booking
(booking_id      INT NOT NULL,
pet_id           INT NOT NULL,
booking_type     enum('mobile-call', 'in-person'),
booking_on       DATETIME NOT NULL,
CONSTRAINT bookingPK PRIMARY KEY (booking_id),
CONSTRAINT registers_pet_pet_idFK FOREIGN KEY (pet_id) REFERENCES pet(pet_id));
```

```
CREATE TABLE appointment(
apt_id INT NOT NULL,
booking_id INT NOT NULL,
date_of_apt TIMESTAMP,
apt_note VARCHAR(120) NOT NULL,
room_name VARCHAR(5) NOT NULL,
vet_id INT NOT NULL,
treatment_id INT NOT NULL,
timeslot_id VARCHAR(5) NOT NULL,
CONSTRAINT appoinmentPK PRIMARY KEY (booking_id),
FOREIGN KEY (Timeslot_Id) REFERENCES timeslot(timeslot_id),
CONSTRAINT AppointmentsApt_nameFk FOREIGN KEY(room_name) REFERENCES
room(room_name),
CONSTRAINT AppointmentsTreatment_no_Fk FOREIGN KEY( treatment_id) REFERENCES
treatment(treatment_id),
CONSTRAINT AppointmentsBooking_Id_IdFk FOREIGN KEY(booking_id) REFERENCES
booking(booking_id),
CONSTRAINT AppointmentsVet_Id_IdFk FOREIGN KEY(vet_id) REFERENCES vet(vet_id)
);
```

2. Create at least 10 dummy records in each Table. If you have a table where adding 10 records would not make any sense (not enough data), you may add less than 10 records for the table

<u>Solution</u>

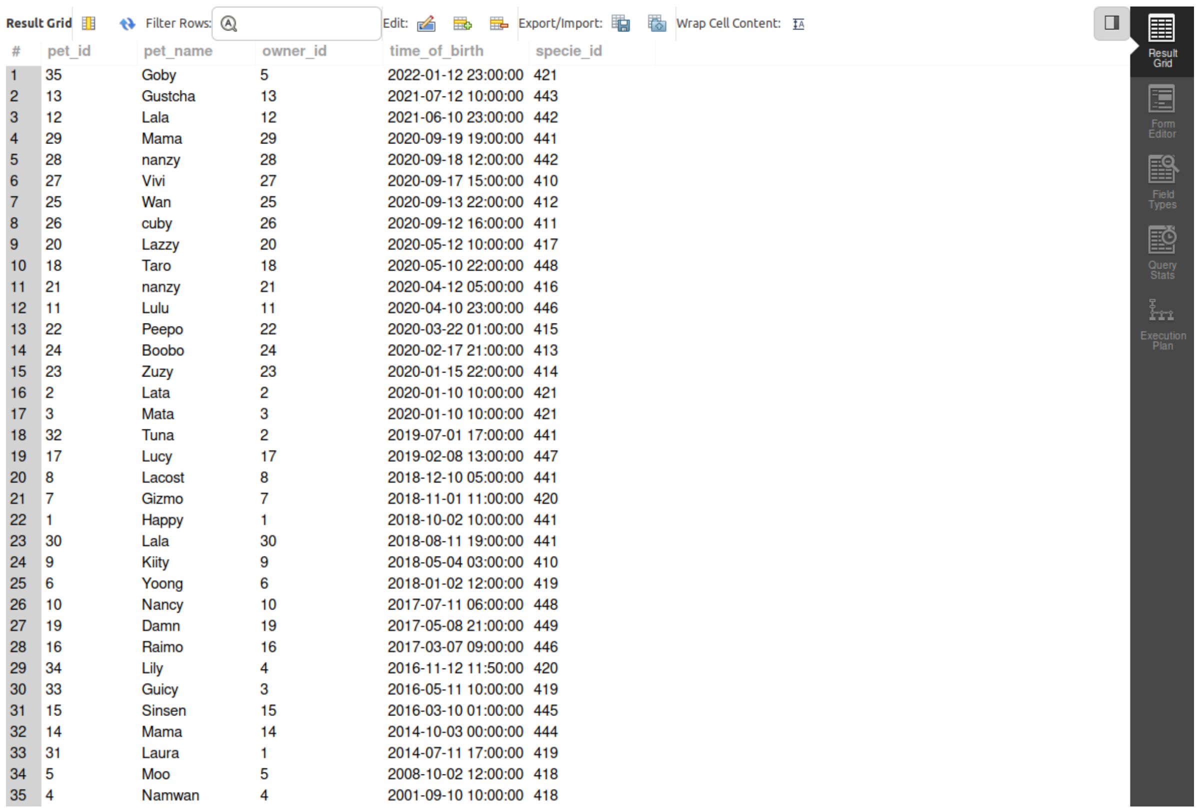
Sample Data attached: scheme_data.sql

3. Write SQL Queries to retrieve the following data from your PetClinic database. Include the SQLs in your delivery – not only the results.

Solution

a. List all registered pets, ordered by their birth date.

```
SELECT *
FROM pet
ORDER BY time_of_birth DESC;
```

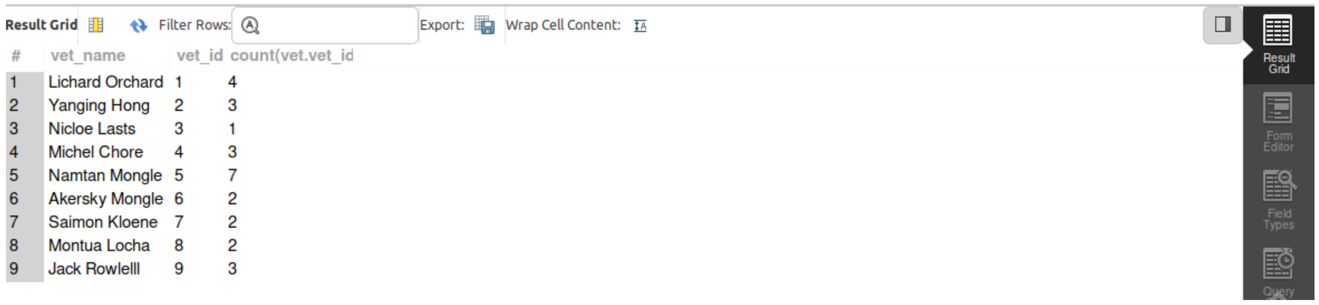


#	pet_id	pet_name	owner_id	time_of_birth	specie_id
1	35	Goby	5	2022-01-12 23:00:00	421
2	13	Gustcha	13	2021-07-12 10:00:00	443
3	12	Lala	12	2021-06-10 23:00:00	442
4	29	Mama	29	2020-09-19 19:00:00	441
5	28	nanzy	28	2020-09-18 12:00:00	442
6	27	Vivi	27	2020-09-17 15:00:00	410
7	25	Wan	25	2020-09-13 22:00:00	412
8	26	cuby	26	2020-09-12 16:00:00	411
9	20	Lazzy	20	2020-05-12 10:00:00	417
10	18	Taro	18	2020-05-10 22:00:00	448
11	21	nanzy	21	2020-04-12 05:00:00	416
12	11	Lulu	11	2020-04-10 23:00:00	446
13	22	Peepo	22	2020-03-22 01:00:00	415
14	24	Boobo	24	2020-02-17 21:00:00	413
15	23	Zuzy	23	2020-01-15 22:00:00	414
16	2	Lata	2	2020-01-10 10:00:00	421
17	3	Mata	3	2020-01-10 10:00:00	421
18	32	Tuna	2	2019-07-01 17:00:00	441
19	17	Lucy	17	2019-02-08 13:00:00	447
20	8	Lacost	8	2018-12-10 05:00:00	441
21	7	Gizmo	7	2018-11-01 11:00:00	420
22	1	Happy	1	2018-10-02 10:00:00	441
23	30	Lala	30	2018-08-11 19:00:00	441
24	9	Kiity	9	2018-05-04 03:00:00	410
25	6	Yoong	6	2018-01-02 12:00:00	419
26	10	Nancy	10	2017-07-11 06:00:00	448
27	19	Damn	19	2017-05-08 21:00:00	449
28	16	Raimo	16	2017-03-07 09:00:00	446
29	34	Lily	4	2016-11-12 11:50:00	420
30	33	Guicy	3	2016-05-11 10:00:00	419
31	15	Sinsen	15	2016-03-10 01:00:00	445
32	14	Mama	14	2014-10-03 00:00:00	444
33	31	Laura	1	2014-07-11 17:00:00	419
34	5	Moo	5	2008-10-02 12:00:00	418
35	4	Namwan	4	2001-09-10 10:00:00	418

Figure a

b. Which veterinarian has the highest number of registered appointments? If multiple veterinarians have the same highest number of registered appointments, you may select one of them.

```
select vet.vet_name ,vet.vet_id, count(vet.vet_id)
from appointment
join vet on vet.vet_id = appointment.vet_id
group by vet.vet_id
having count(vet.vet_id) = (
select max(count)
from (
select vet.vet_id, count(vet.vet_id) as count
from appointment
group by vet_id
) as count);
```

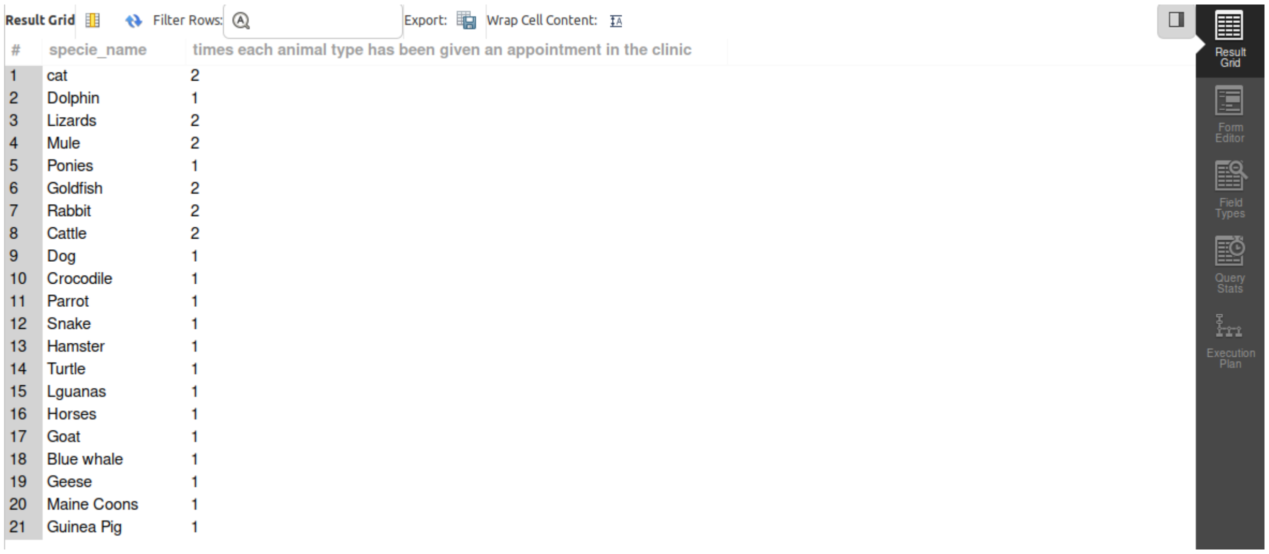


```
#  vet_name        vet_id  count(vet.vet_id
1  Lichard Orchard 1       4
2  Yanging Hong    2       3
3  Nicloe Lasts    3       1
4  Michel Chore    4       3
5  Namtan Mongle   5       7
6  Akersky Mongle  6       2
7  Saimon Kloene   7       2
8  Montua Locha    8       2
9  Jack Rowlelll   9       3
```

Figure b

c. Retrieve information about all animal types in the database, and how many times each animal type has been given an appointment in the clinic.

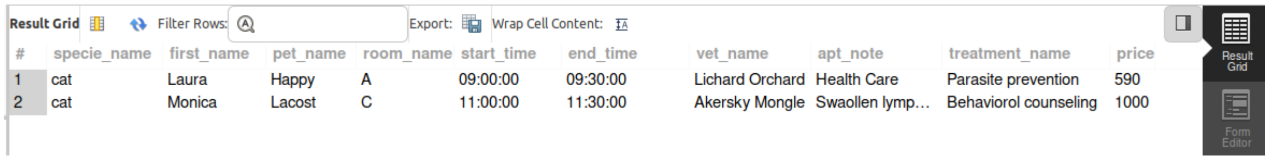
```
select
specie.specie_id,specie.specie_name,count(specie.specie_id) as
"times each animal type has been given an appointment in the clinic"
from specie
join pet on pet.specie_id = specie.specie_id
join booking on booking.pet_id = pet.pet_id
join appointment on appointment.booking_id = booking.booking_id
group by specie.specie_id;
```



```
#   specie_name   times each animal type has been given an appointment in the clinic
1   cat           2
2   Dolphin       1
3   Lizards       2
4   Mule          2
5   Ponies        1
6   Goldfish      2
7   Rabbit        2
8   Cattle        2
9   Dog           1
10  Crocodile     1
11  Parrot        1
12  Snake         1
13  Hamster       1
14  Turtle        1
15  Lguanas       1
16  Horses        1
17  Goat          1
18  Blue whale    1
19  Geese         1
20  Maine Coons   1
21  Guinea Pig    1
```

d. Retrieve a list of appointments where at least one cat is involved. (Use another animal than cat if you do not have cats in your database).

```
select specie.specie_name, owner.first_name,pet.pet_name,room.room_name,
timeslot.start_time, timeslot.end_time,
vet.vet_name,appointment.apt_note, treatment.treatment_name,
treatment.price
from appointment
join booking on booking.booking_id = appointment.booking_id
join pet on pet.pet_id = booking.pet_id
join specie on specie.specie_id = pet.specie_id
join owner on owner.owner_id = pet.owner_id
join treatment on treatment.treatment_id =
appointment.treatment_id
join timeslot on timeslot.timeslot_id = appointment.timeslot_id
join vet on vet.vet_id = appointment.vet_id
join room on room.room_name = appointment.room_name
where specie.specie_name = 'cat';
```

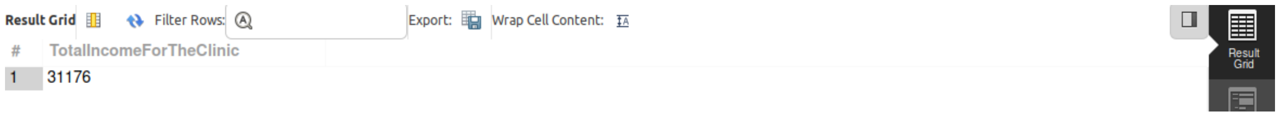


#	specie_name	first_name	pet_name	room_name	start_time	end_time	vet_name	apt_note	treatment_name	price
1	cat	Laura	Happy	A	09:00:00	09:30:00	Lichard Orchard	Health Care	Parasite prevention	590
2	cat	Monica	Lacost	C	11:00:00	11:30:00	Akersky Mongle	Swaollen lymp...	Behaviorol counseling	1000

Figure d

e. Retrieve the total income for the clinic, based on the prices for all treatments in all Appointments.

```
select sum(treatment.price)
from appointment
join treatment on treatment.treatment_id = appointment.treatment_id;
```



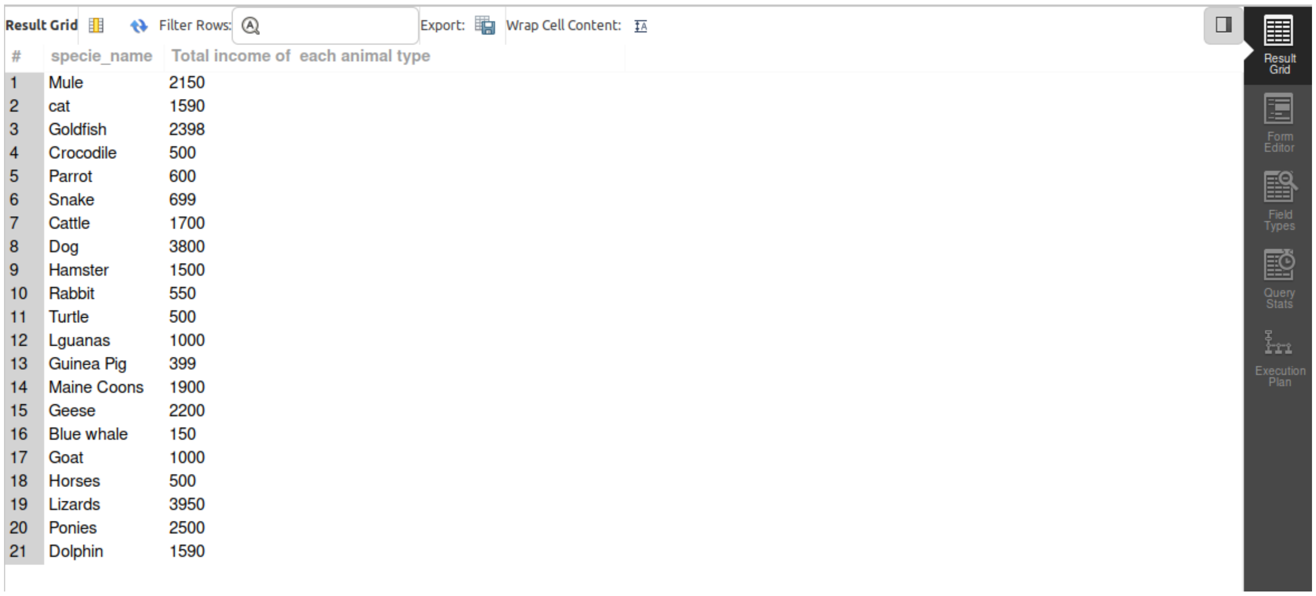
#	TotalIncomeForTheClinic
1	31176

Figure e

f. Create a view displaying all animal types and how much income each animal type has generated, based on the price of treatments in all appointments. Check to see if your answer corresponds with your result in f.

```
create view displaying_all_animal_type as
select specie.specie_name,sum(treatment.price) as TotalIncome
from appointment
join treatment on treatment.treatment_id = appointment.treatment_id
join booking on appointment.booking_id = booking.booking_id
join pet on pet.pet_id = booking.pet_id
join specie on specie.specie_id = pet.specie_id
group by specie.specie_name;

select *
from displaying_all_animal_type;
```



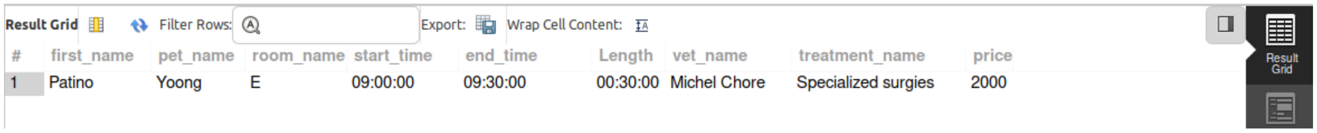
```
#   specie_name   Total income of  each animal type
1   Mule          2150
2   cat           1590
3   Goldfish      2398
4   Crocodile     500
5   Parrot        600
6   Snake         699
7   Cattle        1700
8   Dog           3800
9   Hamster       1500
10  Rabbit        550
11  Turtle        500
12  Lguanas       1000
13  Guinea Pig    399
14  Maine Coons   1900
15  Geese         2200
16  Blue whale    150
17  Goat          1000
18  Horses        500
19  Lizards       3950
20  Ponies        2500
21  Dolphin       1590
```

Figure F

g. Create two queries where you select information that you find interesting to retrieve from the case. Try to create queries that are a good representation of your level of understanding of SQL. In other words; you decide the level of complexity/difficulty in the queries.

g.1 calculate length of appointment for **“apt_id = 5”**

```
select owner.first_name,pet.pet_name,room.room_name, timeslot.start_time,
timeslot.end_time, timediff(end_time, start_time) as 'Length',
vet.vet_name, treatment.treatment_name, treatment.price
from appointment
join timeslot on timeslot.timeslot_id = appointment.timeslot_id
join room on room.room_name = appointment.room_name
join booking on booking.booking_id = appointment.booking_id
join pet on pet.pet_id = booking.pet_id
join owner on owner.owner_id = pet.owner_id
join vet on vet.vet_id = appointment.vet_id
join treatment on treatment.treatment_id = appointment.treatment_id
where apt_id = 5;
```

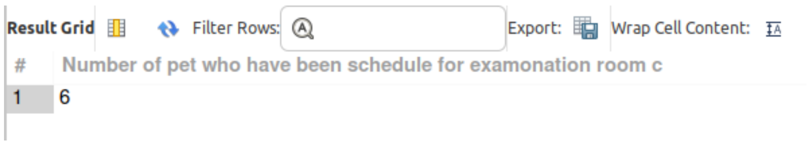


#	first_name	pet_name	room_name	start_time	end_time	Length	vet_name	treatment_name	price
1	Patino	Yoong	E	09:00:00	09:30:00	00:30:00	Michel Chore	Specialized surgies	2000

Figure g.1

g.2 Count the number of unique pet who have been schedule for examination for **"room c"**

```
select  count(distinct appointment.booking_id) as  "Number of pet who have
been schedule for examination room c"
from appointment
join booking on booking.booking_id = appointment.booking_id
join pet on pet.pet_id = booking.pet_id
join room on room.room_name = appointment.room_name
join timeslot on timeslot.timeslot_id = appointment.timeslot_id
where room.room_name = 'C';
```



#	Number of pet who have been schedule for examonation room c
1	6

Figure g.2

4. Write a paragraph explaining your database design choices and explain why you think your database tables are in good normalized forms. Note: There is no need to present all steps of normalization while designing the database itself.

<u>Solution</u>

I have created a set of relations PetClinic to represent the data model created. In step I validate the groupings of attributes in each relation using the rules of normalization form to ensure that the set of PetClinic relations has a minimal (to minimize data redundancy) yet sufficient number of attributes necessary to support the data requirements of the customer. Also the PetClinic relations have minimal data redundancy; most PetClinic relations are at least 3NF to avoid problems of update anomalies. However, some design redundancy to allow the joining of related relations. For example, based on design it was found out that normalization into 2NF and 3 NF form doesn't always provide maximum processing efficiency as it needs many of joining in related relations and get duplicate values as a result.

Follow the step of Normalization. Please see the given simple data **of 1NF form from PetClinic (Owner Entity and Pet Entity)**

Owner_Pet relation
Owner_Pet (owner_id, first_name , Last_name, date_of_birth,
street,zip_code, city, contactNr, email,
pet_id, pet_name, owner_id, time_of_birth,specie_id)

First step Pet Clinic DataSource was collected in the form of user requirement specification report (Unnormalized form) containing one and more repeating groups. Then we begin with the processing of normalization by first transferring the data from the source (for example, data entities form) into table format with row and column. To transfer the normalization form into first normal form, I identify and remove repeating groups within the table . By removing repeating groups in an attribute or group attribute within the table that appear to have multivalue. Next step would be First Normal Form (1NF). The simple data is **Owner_Pet** tables, some owners have more than 1 animal. The relation contains data describing owner name, pet names which is repeated pet name with the same owner several times. As a result, the **Owner_Pet** table relation contains data redundancy. If implemented, the 1NF relation would be subject to the update anomalies. To remove some of these, we must transform the relation into a second normalization form . We remove (Pet details) out of the **OwnerPet** relation by placing the repeating data along with a copy of the original key attribute (Owner_id) in the separate relation. With the help of functional dependencies. I identify a primary key for the relations. The simple data of the resulting 2NF relations is as follows.

Owner relation
Owner (owner_id, first_name , Last_name, date_of_birth,
street,zip_code, city, contactNr, email, pet_id)x`

Pet relation
Pet (owner_id, time_of_birth,specie_id)

To summarize, PetClinic is created mainly to remove data redundancy and to be easy to handle and update data. Each table has a primary key and foreign key as an attribute to identify the relationship between each table.

5. Choose two of the four ACID-properties for a DBMS, and describe how these two properties affect the use of the PetClinic database.

Solution

Isolation: Pet clinic databases may have many numbers of users accessing the same database and table at the same time. This issue often leads to concurrency situations. For example, if two receptionists are trying to book the appointment at the same date, same time and same vet, the system performs a transaction que to ensure the first user finishes the booking before the next can begins, it will make the booking of the other user be interrupted. That means they should not be affected by each other and transactions happen separately from one another. To summarize, transactions that happened in the PetClinic database have to be run at an isolation level to prevent losing the update that might happen when two transactions each select the same row, so lost updates do not happen at the default isolation level of read committed.

Durability: The result of a transaction or booking in PetClinic is completed and committed, its change must be stored permanently in the database, regardless of what happens in future transactions. This property is to ensure that the information that is saved in the pet clinic database is immutable until another update or transaction affects it. Once the change in PetClinic is committed, it will remain in this state even in the case of system failure. Because the completed transactions are recorded on permanent memory devices such as hard drives so the data still will be available so committed data is never lost, even MySQL is restarted.

6. Would It be a good idea to use NoSQL database instead of Relational Database for the clinic? Please provide your explanation using one or two paragraphs. You can also use examples as arguments to prove your opinion.

<u>Solution</u>

I'm not agree with the concept of using a NoSQL database with PetClinic. Even development with NoSQL can be faster and easier to scale than database development. However, MySQL Databases are best suited for complex queries while NoSQL are not suitable for complex queries.

PetClinic requires multiple-rows which are more suitable to use tables-base on the other hand NoSQL is either offered in terms of document and graph database. NoSQL is more complicated to find support from their developers. Also a lot of independent developers are there who can help with SQL databases. NoSQL also comes with the disadvantage of not using the full ACID principles which are important for safety and reliable platforms that use it widely.

References :

1. ^ Andy Oppel (October 22, 2015) “SQL: A Beginner's Guide, Fourth Edition”
2. ^ ValueLabs (2021-09-19) “Understanding Database isolation level via examples — MySQL and Postgres” https://amirsoleimani.medium.com/understanding-database-isolation-level-via-examples-mysql-and-postgres-a86b5502d404