DBMS Project Report

PES University

Database Management Systems

UE20CS301

Submitted By

PES1UG20CS697 Thejaswi A

Animal shelter database

The database aims at maintaining records of animals and all the activities revolving around them. Be it, adoption, rescue, caring etc. Record of all the animal shelters along with all the other related entities is present in this centralized database. It even keeps track of all the employees, sources of income, monitoring Government bodies, philanthropists, etc. Querying is simplified since all the entities involved along with their relationships are simple. Retrieval of data is one of the main goals of a database and that is made possible with this.
Animal shelter database involves a lot of intricate complexities just like any other database. A consistent record of all the activities needs to be maintained and this can be problem because Animal shelters are very dynamic in nature. There are a lot of changes that happen all the time. Be it animals being adopted, neutered, euthanized etc, or Shelters receiving donations, funding etc. There is a possibility for loads of errors not only during the implementation of the database, but during the design itself. With all this in mind, a database with minimal error and maximum consistency has been designed.

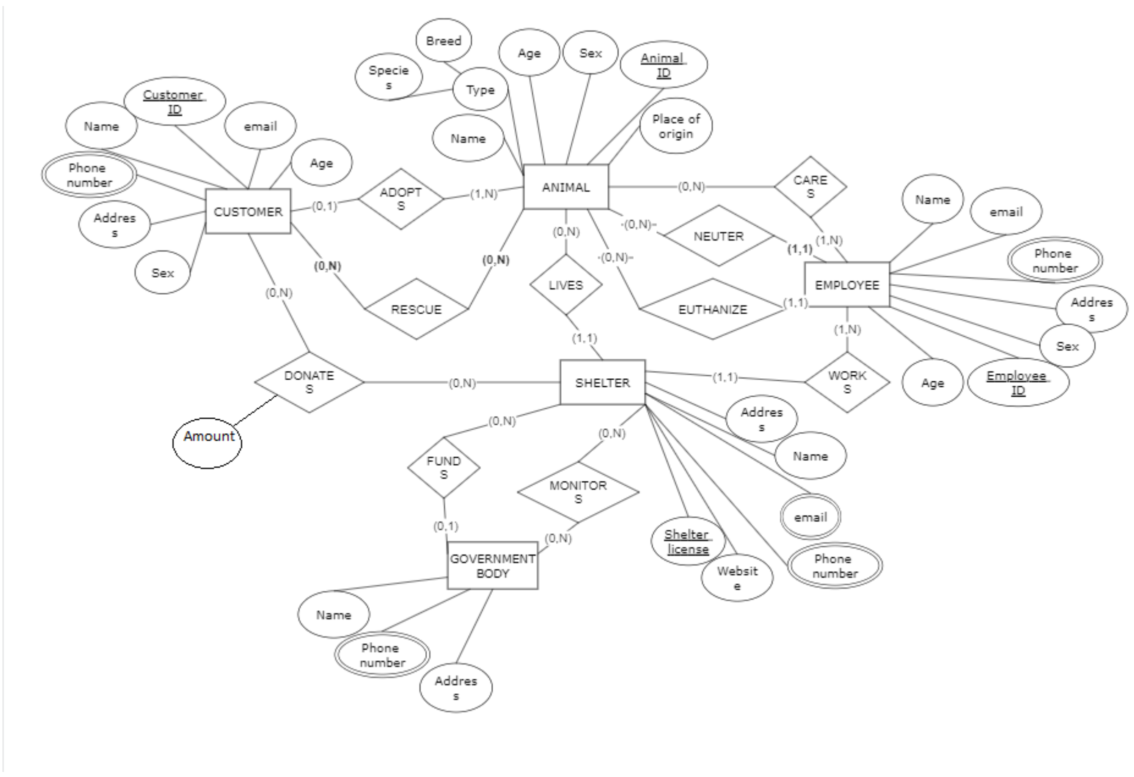
Index

Introduction

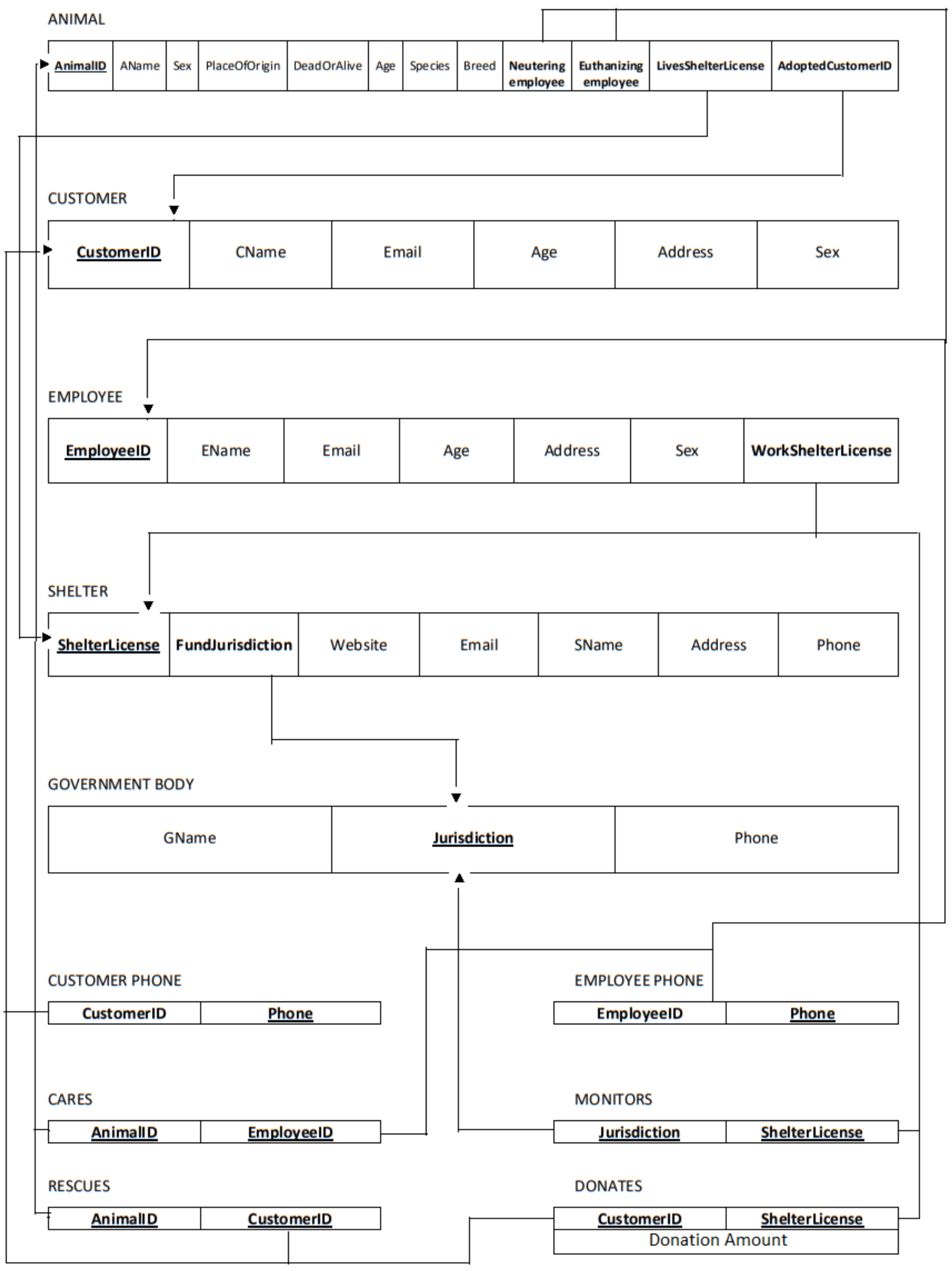
Animal Shelter database enables us to have a centralized system to have a check on activities related to animals conveniently. A lot of scandalous activities can be avoided if a proper record is maintained. Identifying animals or any other entities involved without a unique tag could be a challenging task and having a database where records are maintained with a unique ID makes it all easier as compared to the alternate.
The database required creation of 11 tables corresponding to 5 real life entities and relationships between them. All the attributes had to be carefully picked based on relevance and some of them had to be dropped because of redundancy. What I ended up with finally, was a carefully curated set of tables which could efficiently and precisely store and retrieve data.
Triggers have been created to assist during any possibility of errors while insertion, updating or deletion of values from the database. These prove to be very helpful when insertions happen in bulk.

Data Model

Entity-Relationship diagram



Relation Schema


ANIMAL
AnimalID | AName | Sex | PlaceOfOrigin | DeadOrAlive | Age | Species | Breed | Neutering employee | Euthanizing employee | LivesShelterLicense | AdoptedCustomerID
CUSTOMER
CustomerID | CName | Email | Age | Address | Sex
EMPLOYEE
EmployeeID | EName | Email | Age | Address | Sex | WorkShelterLicense
SHELTER
ShelterLicense | FundJurisdiction | Website | Email | SName | Address | Phone
GOVERNMENT BODY
GName | Jurisdiction | Phone
CUSTOMER PHONE
CustomerID | Phone
EMPLOYEE PHONE
EmployeeID | Phone
CARES
AnimalID | EmployeeID
MONITORS
Jurisdiction | ShelterLicense
RESCUES
AnimalID | CustomerID
DONATES
CustomerID | ShelterLicense
Donation Amount

Conversion of E-R diagram to Relation Schema:

1. Relationships with 1:1 cardinality ratio:
   There were no relationships with 1:1 cardinality ratio
2. Relationships with 1:N cardinality ratio:
   1. Customer adopts animal
   2. Animal lives in Shelter
   3. Employee works in Shelter
   4. Employee neuters Animal
   5. Employee euthanizes Animal
   6. Government body funds Shelter

   Two relations were created for each such relationships with the entity on the N side taking the foreign key of the entity on the 1 side.

3. Relationships with M:N cardinality ratio:
   1. Customer donates to Shelter
   2. Government body monitors Shelter
   3. Employee cares for Animal
   4. Customer rescues Animal

   Three relations were created, two for each of the entities in the relationship and one for the relationship itself.

Relations:

1. **Animal**
   Primary key – Animal ID (int)
   Foreign keys – Employee ID of employee who neuters if neutered else NULL, Employee ID of employee who euthanizes if euthanized else NULL, Shelter license of the shelter where the animal lives and SSN of customer if the animal is adopted else NULL.
2. **Customer**
   Primary key – SSN (int)
3. **Employee**
   Primary key – Employee ID (int)
   Foreign key – Shelter license of the shelter where the employee works at.
4. **Shelter**
   Primary key – Shelter license (string)
   Foreign key – Jurisdiction of the Government body monitoring the shelter
5. **Government body**
   Primary key – Jurisdiction (string)
6. **Customer phone**
   Primary key – Phone number of the customers (int)
   Foreign key – SSN of the customer
7. **Employee phone**
   Primary key – Phone number of the employees (int)

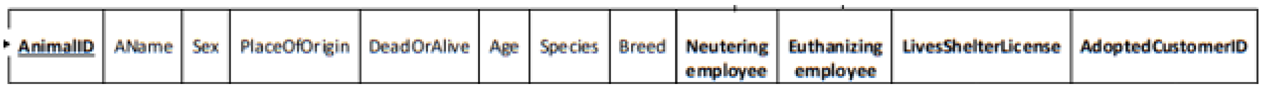
Foreign key – Employee ID of the employee

8. **Donations –** Relation for the M:N relationship between Customer and Shelter
   Primary key – (SSN, Shelter license)
   Foreign keys – SSN, Shelter license
9. **Rescues –** Relation for the M:N relationship between Customer and Animal
   Primary key – (SSN, Animal ID)
   Foreign keys – SSN, Animal ID
10. **Cares –** Relation for the M:N relationship between Employee and Animal
    Primary key – (Employee ID, Animal ID)
    Foreign keys – Employee ID, Animal ID
11. **Monitors –** Relation for the M:N relationship between Shelter and Government body
    Primary key – (Shelter license, Jurisdiction)
    Foreign keys – Shelter license, Jurisdiction

FD and Normalization

Functional dependencies

1. Animal

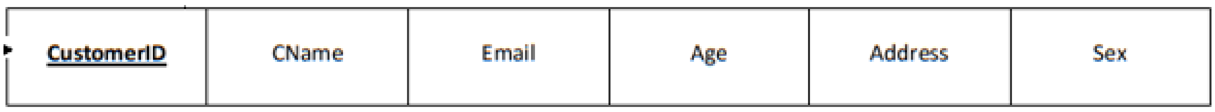


AnimalID	AName	Sex	PlaceOfOrigin	DeadOrAlive	Age	Species	Breed	Neutering employee	Euthanizing employee	LivesShelterLicense	AdoptedCustomerID

Functional dependencies:
AnimalID→{AName, Sex, PlaceOfOrigin, DeadOrAlive, Age, Species, Breed, NeuteringEmployee, Euthanizing Employee, LivesShelterlicense, AdoptedCustomerID}

The relation is in **BCNF**

2. Customer

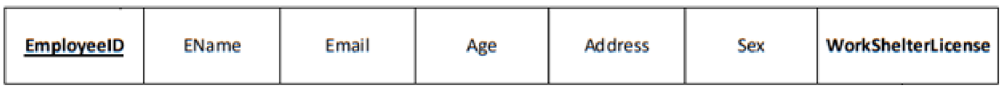


CustomerID	CName	Email	Age	Address	Sex

Functional dependencies:
CustomerID→{CName, Email, Age, Address, Sex}

The relation is in **BCNF**

3. Employee

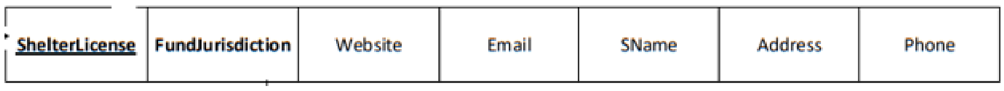


EmployeeID	EName	Email	Age	Address	Sex	WorkShelterLicense

Functional dependencies:
EmployeeID→{EName, Email, Age, Address, Sex, WorkShelterlicense}

The relation is in **BCNF**

4. Shelter

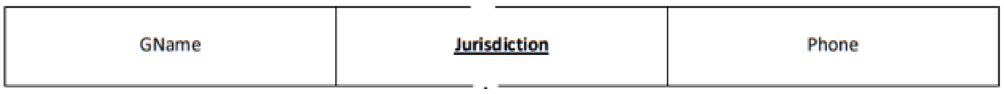


ShelterLicense	FundJurisdiction	Website	Email	SName	Address	Phone

Functional dependencies:
Shelterlicense→{FundJurisdiction, Website, Email, Sname, Address, Phone}

The relation is in **BCNF**

5. Government body

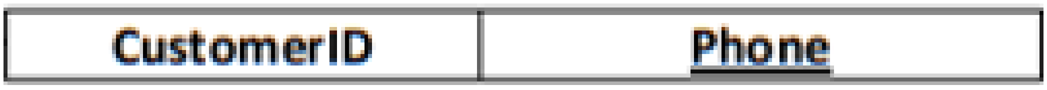


GName	Jurisdiction	Phone

Functional dependencies:
Jurisdiction→{GName, Phone}

The relation is in **BCNF**

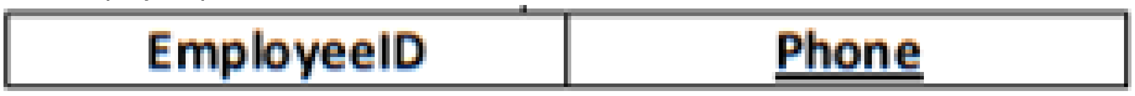
6. Customer phone



CustomerID	Phone

Functional dependencies:
Phone→{CustomerID}
The relation is in **BCNF**

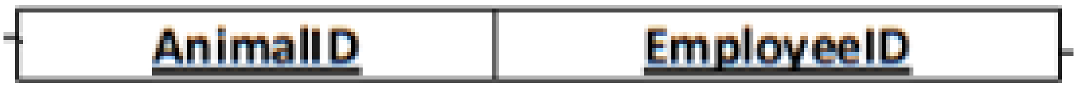
7. Employee phone



EmployeeID	Phone

Functional dependencies:
Phone→{EmployeeID}
The relation is in **BCNF**

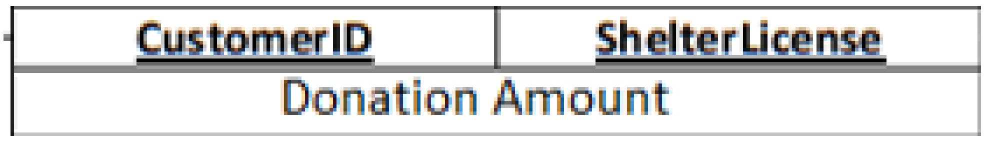
8. Cares



AnimalID	EmployeeID

Functional dependencies:
{AnimalID, EmployeeID}→{AnimalID, EmployeeID}
The relation is in **BCNF**

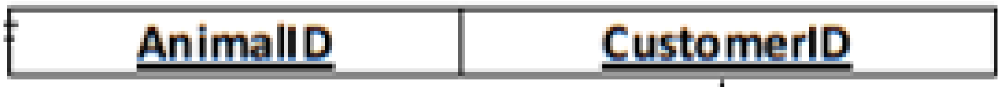
9. Donations



CustomerID	ShelterLicense
Donation Amount	

Functional dependencies:
{CustomerID, Shelterlicense}→Donation Amount
The relation is in **BCNF**

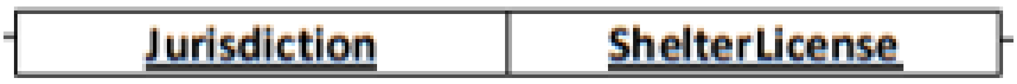
10. Rescues



AnimalID	CustomerID

Functional dependencies:
{AnimalID, CustomerID}→{AnimalID, CustomerID}
The relation is in **BCNF**

11. Monitors



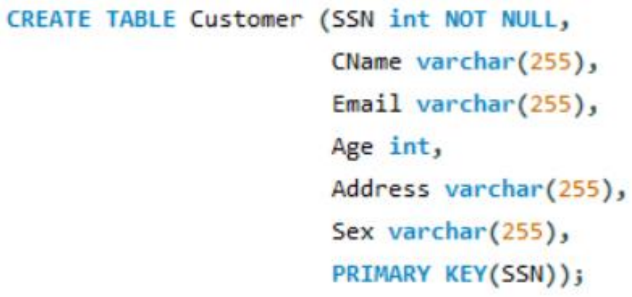
Jurisdiction	ShelterLicense

Functional dependencies:
{Jurisdiction, Shelterlicense}→{Jurisdiction, Shelterlicense}

The relation is in **BCNF**

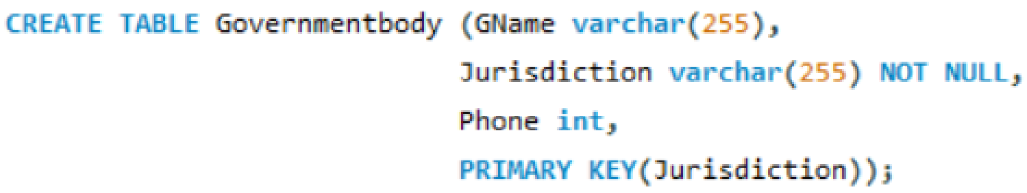
DDL

Customer



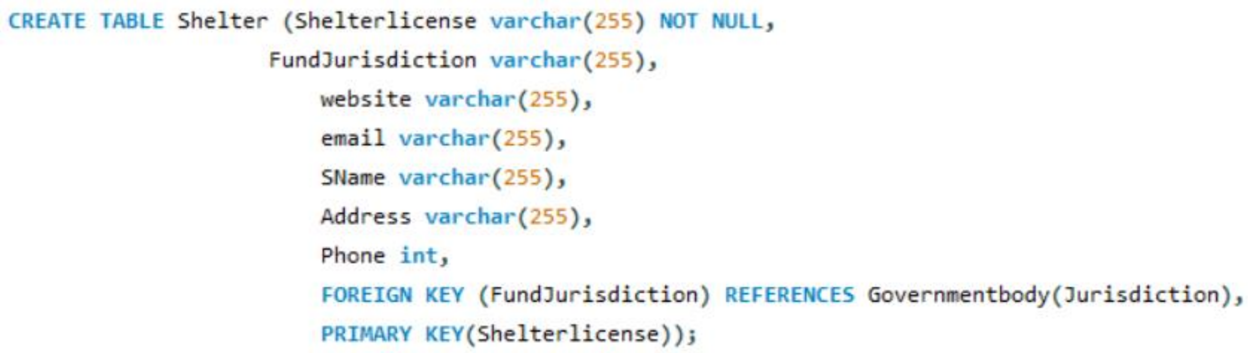
```
CREATE TABLE Customer (SSN int NOT NULL,
                       CName varchar(255),
                       Email varchar(255),
                       Age int,
                       Address varchar(255),
                       Sex varchar(255),
                       PRIMARY KEY(SSN));
```

Government body



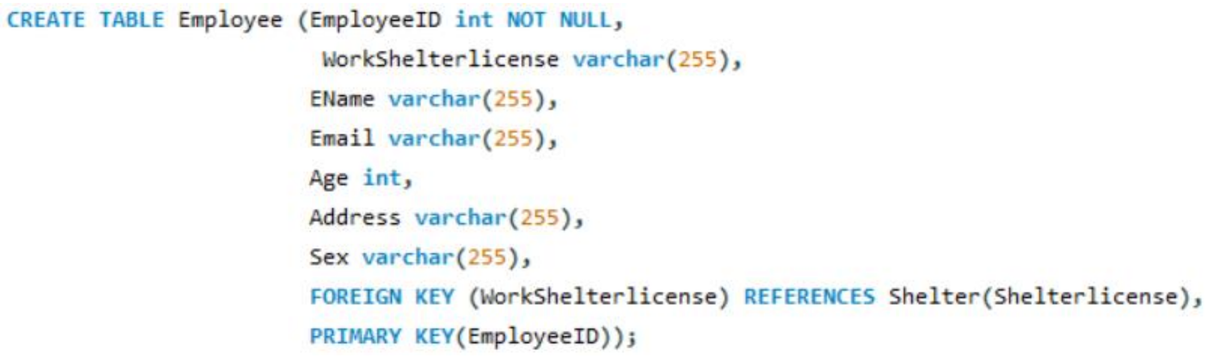
```
CREATE TABLE Governmentbody (GName varchar(255),
                             Jurisdiction varchar(255) NOT NULL,
                             Phone int,
                             PRIMARY KEY(Jurisdiction));
```

Shelter



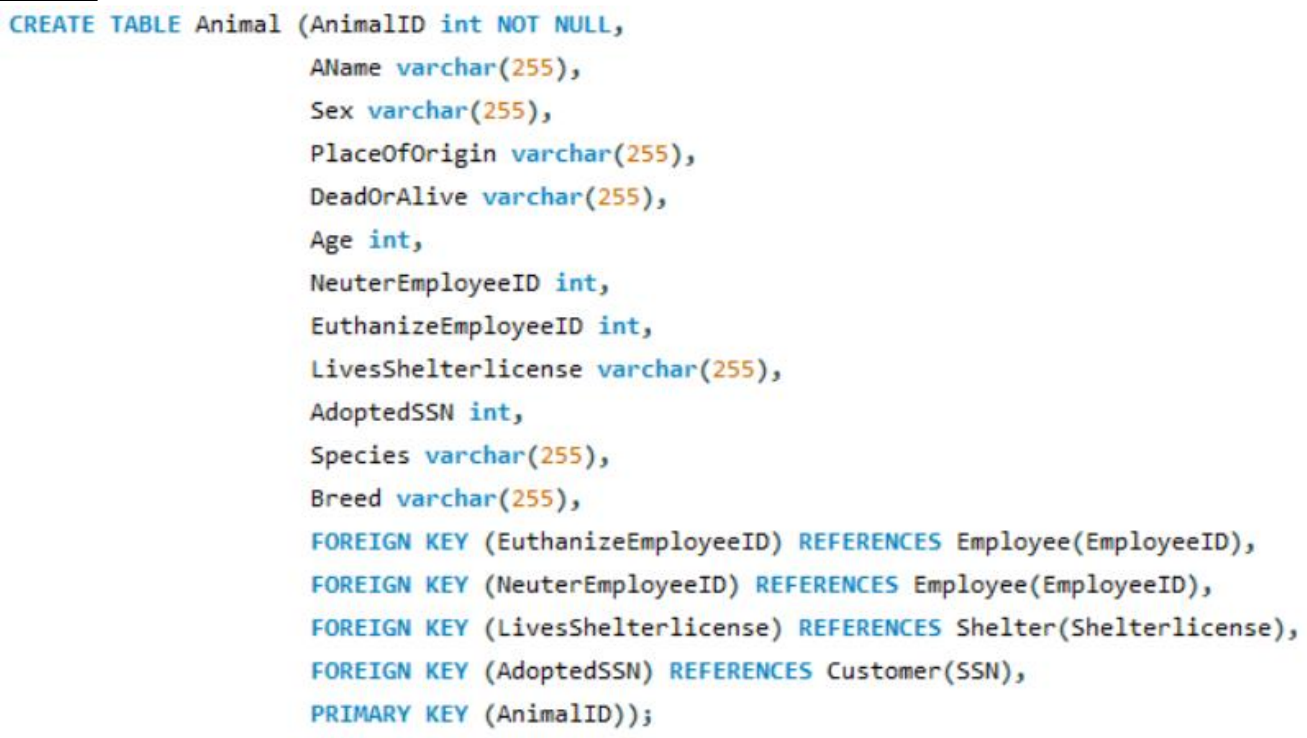
```
CREATE TABLE Shelter (Shelterlicense varchar(255) NOT NULL,
                      FundJurisdiction varchar(255),
                         website varchar(255),
                         email varchar(255),
                         SName varchar(255),
                         Address varchar(255),
                         Phone int,
                         FOREIGN KEY (FundJurisdiction) REFERENCES Governmentbody(Jurisdiction),
                         PRIMARY KEY(Shelterlicense));
```

Employee



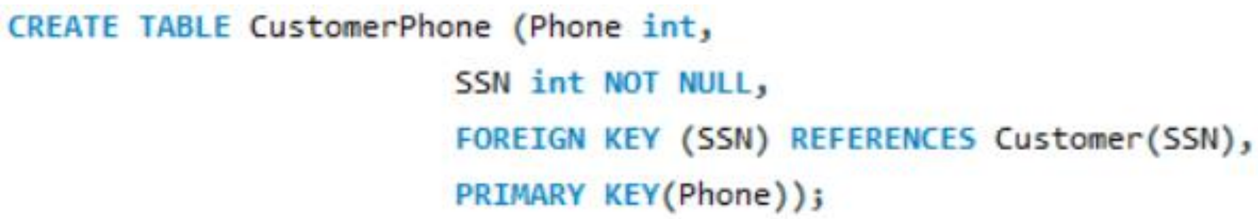
```
CREATE TABLE Employee (EmployeeID int NOT NULL,
                        WorkShelterlicense varchar(255),
                       EName varchar(255),
                       Email varchar(255),
                       Age int,
                       Address varchar(255),
                       Sex varchar(255),
                       FOREIGN KEY (WorkShelterlicense) REFERENCES Shelter(Shelterlicense),
                       PRIMARY KEY(EmployeeID));
```

Animal



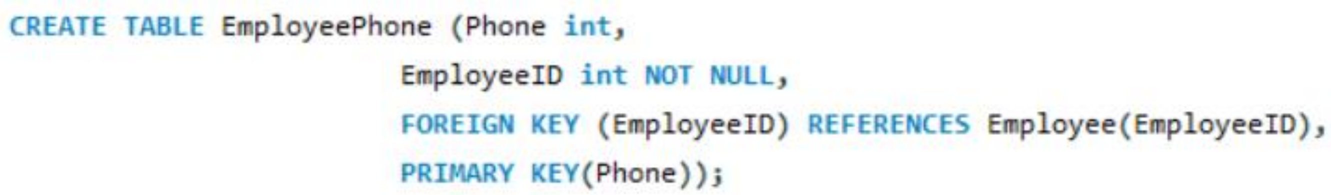
```
CREATE TABLE Animal (AnimalID int NOT NULL,
                     AName varchar(255),
                     Sex varchar(255),
                     PlaceOfOrigin varchar(255),
                     DeadOrAlive varchar(255),
                     Age int,
                     NeuterEmployeeID int,
                     EuthanizeEmployeeID int,
                     LivesShelterlicense varchar(255),
                     AdoptedSSN int,
                     Species varchar(255),
                     Breed varchar(255),
                     FOREIGN KEY (EuthanizeEmployeeID) REFERENCES Employee(EmployeeID),
                     FOREIGN KEY (NeuterEmployeeID) REFERENCES Employee(EmployeeID),
                     FOREIGN KEY (LivesShelterlicense) REFERENCES Shelter(Shelterlicense),
                     FOREIGN KEY (AdoptedSSN) REFERENCES Customer(SSN),
                     PRIMARY KEY (AnimalID));
```

Customer Phone



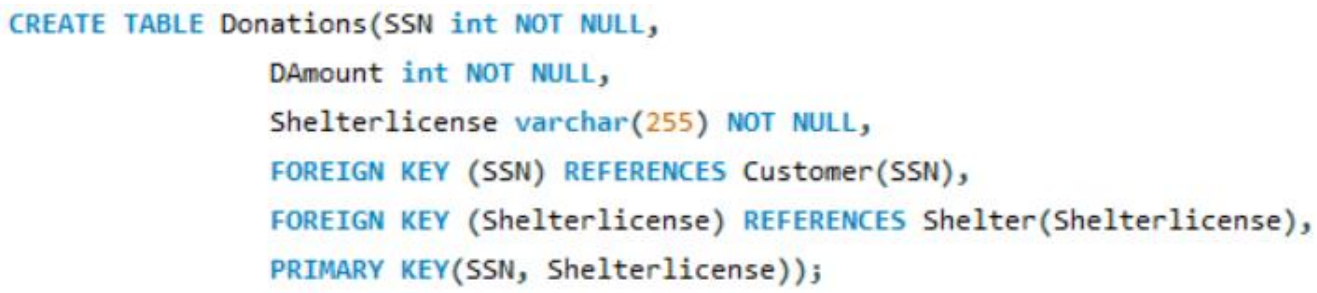
```
CREATE TABLE CustomerPhone (Phone int,
                            SSN int NOT NULL,
                            FOREIGN KEY (SSN) REFERENCES Customer(SSN),
                            PRIMARY KEY(Phone));
```

Employee Phone



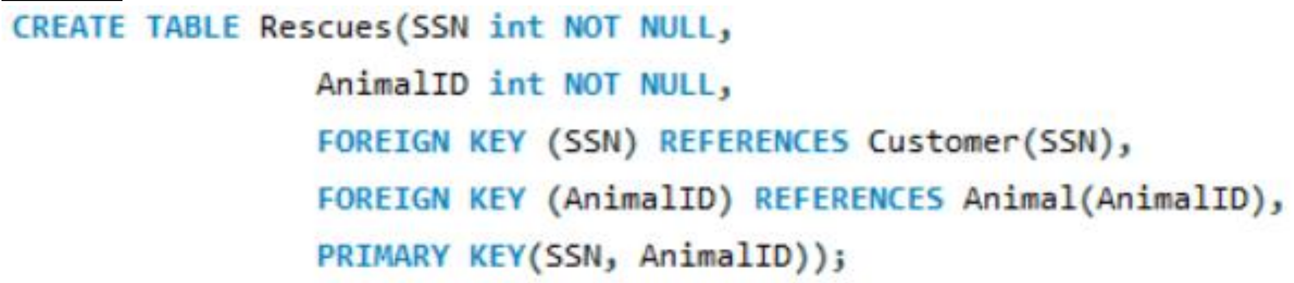
```
CREATE TABLE EmployeePhone (Phone int,
                            EmployeeID int NOT NULL,
                            FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
                            PRIMARY KEY(Phone));
```

Donations



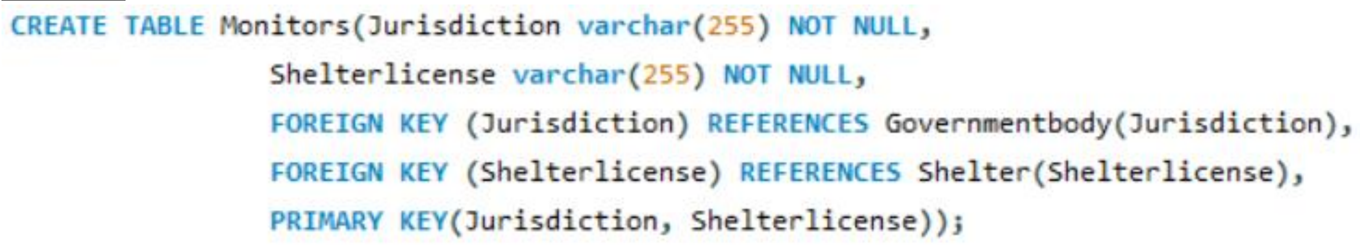
```
CREATE TABLE Donations(SSN int NOT NULL,
                       DAmount int NOT NULL,
                       Shelterlicense varchar(255) NOT NULL,
                       FOREIGN KEY (SSN) REFERENCES Customer(SSN),
                       FOREIGN KEY (Shelterlicense) REFERENCES Shelter(Shelterlicense),
                       PRIMARY KEY(SSN, Shelterlicense));
```

Rescues



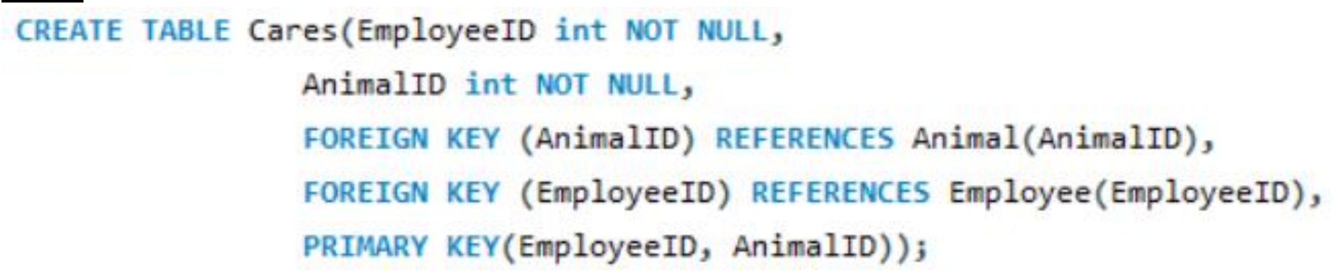
```
CREATE TABLE Rescues(SSN int NOT NULL,
                     AnimalID int NOT NULL,
                     FOREIGN KEY (SSN) REFERENCES Customer(SSN),
                     FOREIGN KEY (AnimalID) REFERENCES Animal(AnimalID),
                     PRIMARY KEY(SSN, AnimalID));
```

Monitors



```
CREATE TABLE Monitors(Jurisdiction varchar(255) NOT NULL,
                      Shelterlicense varchar(255) NOT NULL,
                      FOREIGN KEY (Jurisdiction) REFERENCES Governmentbody(Jurisdiction),
                      FOREIGN KEY (Shelterlicense) REFERENCES Shelter(Shelterlicense),
                      PRIMARY KEY(Jurisdiction, Shelterlicense));
```

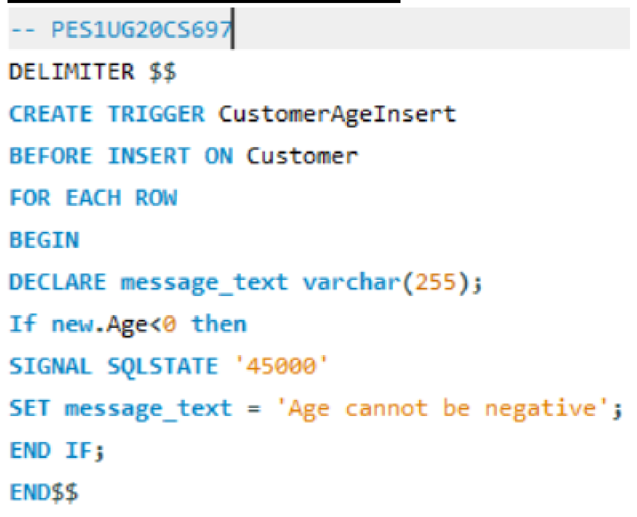
Cares



```
CREATE TABLE Cares(EmployeeID int NOT NULL,
                   AnimalID int NOT NULL,
                   FOREIGN KEY (AnimalID) REFERENCES Animal(AnimalID),
                   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
                   PRIMARY KEY(EmployeeID, AnimalID));
```

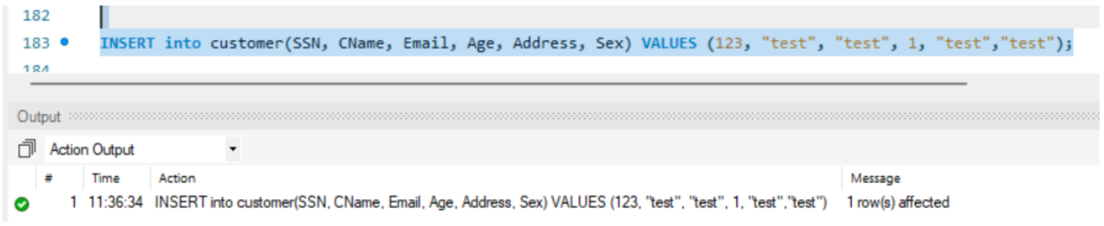
Triggers

1. **Trigger for Invalid Age entry**



```
-- PES1UG20CS697
DELIMITER $$
CREATE TRIGGER CustomerAgeInsert
BEFORE INSERT ON Customer
FOR EACH ROW
BEGIN
DECLARE message_text varchar(255);
If new.Age<0 then
SIGNAL SQLSTATE '45000'
SET message_text = 'Age cannot be negative';
END IF;
END$$
```

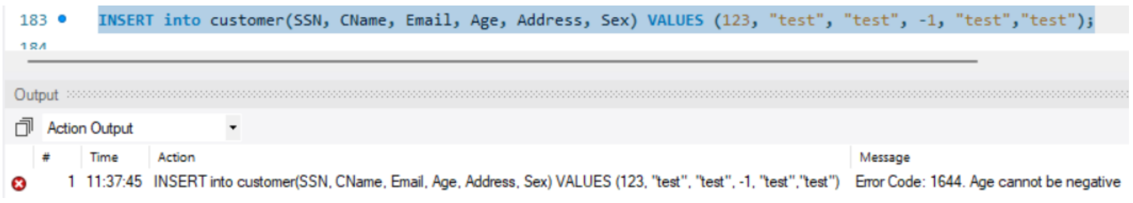
Valid insert –



```
INSERT into customer(SSN, CName, Email, Age, Address, Sex) VALUES (123, "test", "test", 1, "test","test");
```

```
Output
Action Output
#  Time      Action                                                                                          Message
1  11:36:34  INSERT into customer(SSN, CName, Email, Age, Address, Sex) VALUES (123, "test", "test", 1, "test","test")   1 row(s) affected
```

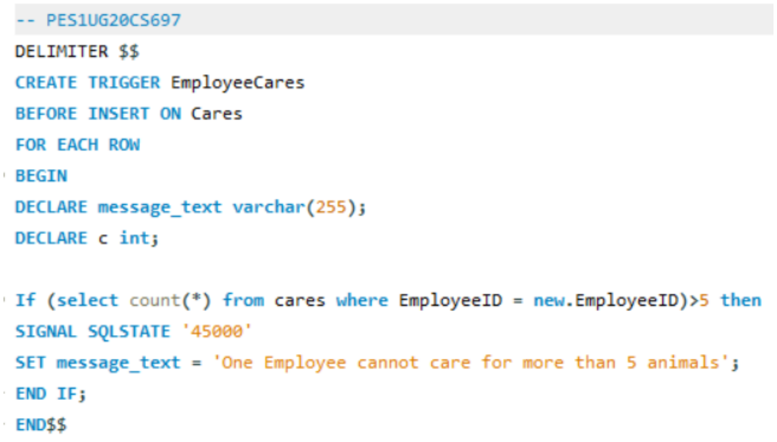
Invalid insert –



```
INSERT into customer(SSN, CName, Email, Age, Address, Sex) VALUES (123, "test", "test", -1, "test","test");
```

```
Output
Action Output
#  Time      Action                                                                                           Message
1  11:37:45  INSERT into customer(SSN, CName, Email, Age, Address, Sex) VALUES (123, "test", "test", -1, "test","test")   Error Code: 1644. Age cannot be negative
```

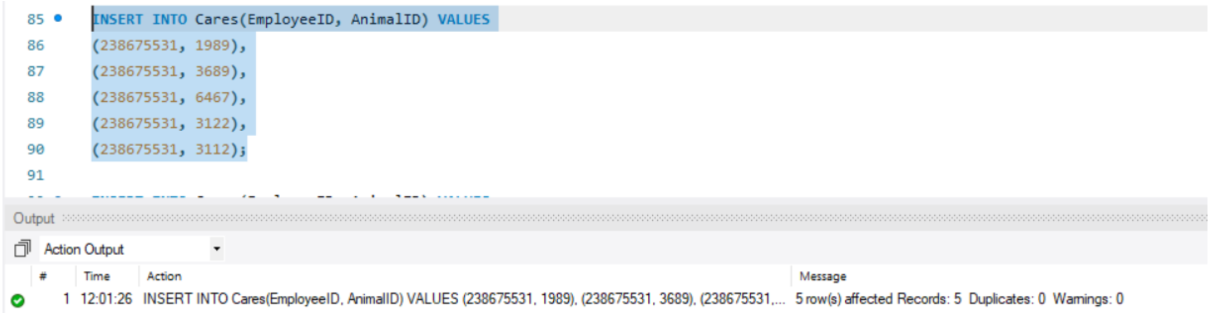
2. **Trigger for Cares. If Employee is in charge of caring for more than 5 animals a trigger is executed.**



```
-- PES1UG20CS697
DELIMITER $$
CREATE TRIGGER EmployeeCares
BEFORE INSERT ON Cares
FOR EACH ROW
BEGIN
DECLARE message_text varchar(255);
DECLARE c int;

If (select count(*) from cares where EmployeeID = new.EmployeeID)>5 then
SIGNAL SQLSTATE '45000'
SET message_text = 'One Employee cannot care for more than 5 animals';
END IF;
END$$
```

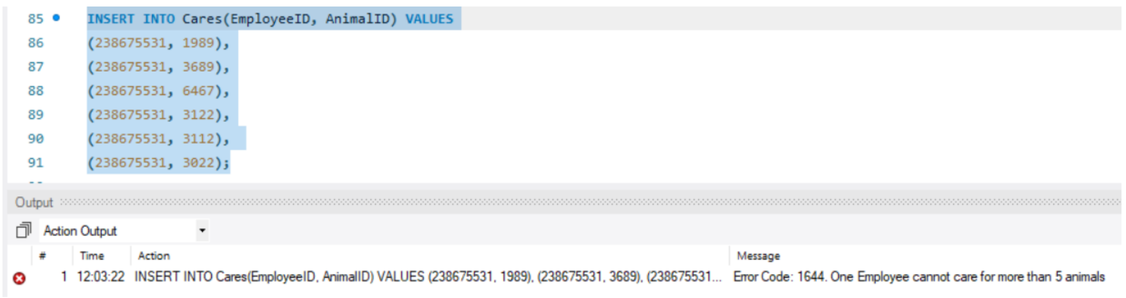
Valid insert –



```
INSERT INTO Cares(EmployeeID, AnimalID) VALUES
(238675531, 1989),
(238675531, 3689),
(238675531, 6467),
(238675531, 3122),
(238675531, 3112);
```

```
Output
Action Output
#  Time      Action                                                                                            Message
1  12:01:26  INSERT INTO Cares(EmployeeID, AnimalID) VALUES (238675531, 1989), (238675531, 3689), (238675531,...   5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0
```

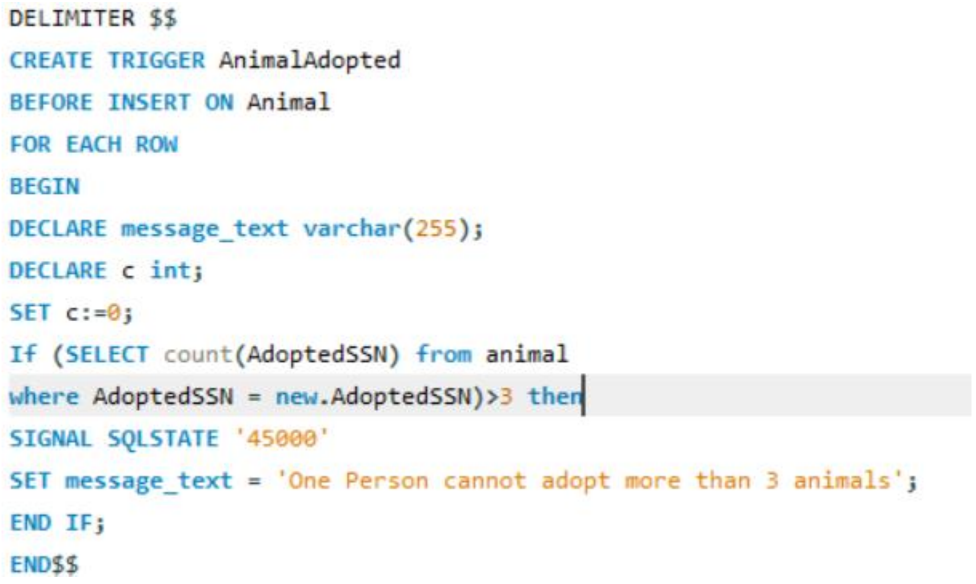
Invalid insert –



```
INSERT INTO Cares(EmployeeID, AnimalID) VALUES
(238675531, 1989),
(238675531, 3689),
(238675531, 6467),
(238675531, 3122),
(238675531, 3112),
(238675531, 3022);
```

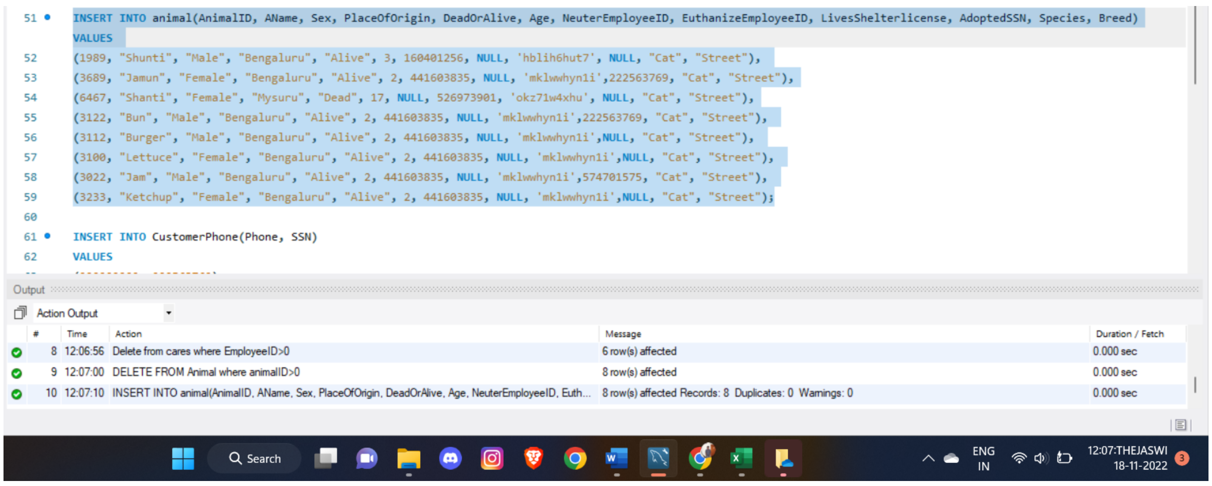
```
Output
Action Output
#  Time      Action                                                                                            Message
1  12:03:22  INSERT INTO Cares(EmployeeID, AnimalID) VALUES (238675531, 1989), (238675531, 3689), (238675531...   Error Code: 1644. One Employee cannot care for more than 5 animals
```

3. **<u>Trigger for Animal. One person cannot adopt more than 3 animals.</u>**



```
DELIMITER $$
CREATE TRIGGER AnimalAdopted
BEFORE INSERT ON Animal
FOR EACH ROW
BEGIN
DECLARE message_text varchar(255);
DECLARE c int;
SET c:=0;
If (SELECT count(AdoptedSSN) from animal
where AdoptedSSN = new.AdoptedSSN)>3 then
SIGNAL SQLSTATE '45000'
SET message_text = 'One Person cannot adopt more than 3 animals';
END IF;
END$$
```

Valid insert –

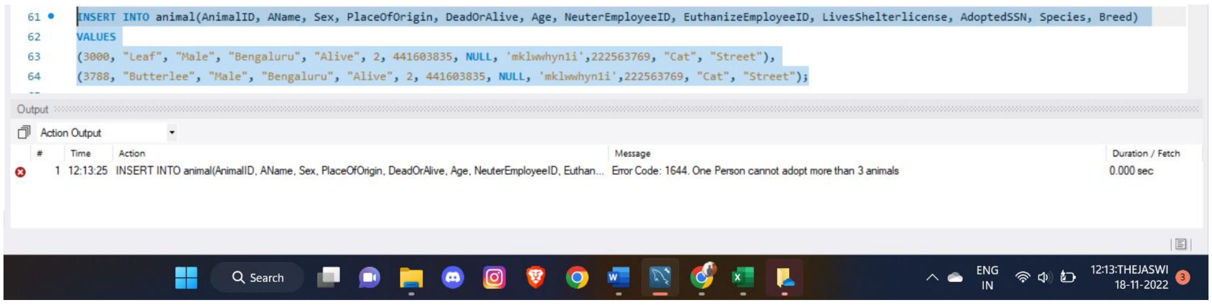


```
INSERT INTO animal(AnimalID, AName, Sex, PlaceOfOrigin, DeadOrAlive, Age, NeuterEmployeeID, EuthanizeEmployeeID, LivesShelterlicense, AdoptedSSN, Species, Breed)
VALUES
(1989, "Shunti", "Male", "Bengaluru", "Alive", 3, 160401256, NULL, 'hblih6hut7', NULL, "Cat", "Street"),
(3689, "Jamun", "Female", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',222563769, "Cat", "Street"),
(6467, "Shanti", "Female", "Mysuru", "Dead", 17, NULL, 526973901, 'okz71w4xhu', NULL, "Cat", "Street"),
(3122, "Bun", "Male", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',222563769, "Cat", "Street"),
(3112, "Burger", "Male", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',NULL, "Cat", "Street"),
(3100, "Lettuce", "Female", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',NULL, "Cat", "Street"),
(3022, "Jam", "Male", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',574701575, "Cat", "Street"),
(3233, "Ketchup", "Female", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',NULL, "Cat", "Street");

INSERT INTO CustomerPhone(Phone, SSN)
VALUES
```

```
Output
Action Output
 #   Time      Action                                                                              Message                                                Duration / Fetch
 8   12:06:56  Delete from cares where EmployeeID>0                                                6 row(s) affected                                      0.000 sec
 9   12:07:00  DELETE FROM Animal where animalID>0                                                 8 row(s) affected                                      0.000 sec
10   12:07:10  INSERT INTO animal(AnimalID, AName, Sex, PlaceOfOrigin, DeadOrAlive, Age, NeuterEmployeeID, Euth...  8 row(s) affected Records: 8  Duplicates: 0  Warnings: 0  0.000 sec
```

Invalid insert –

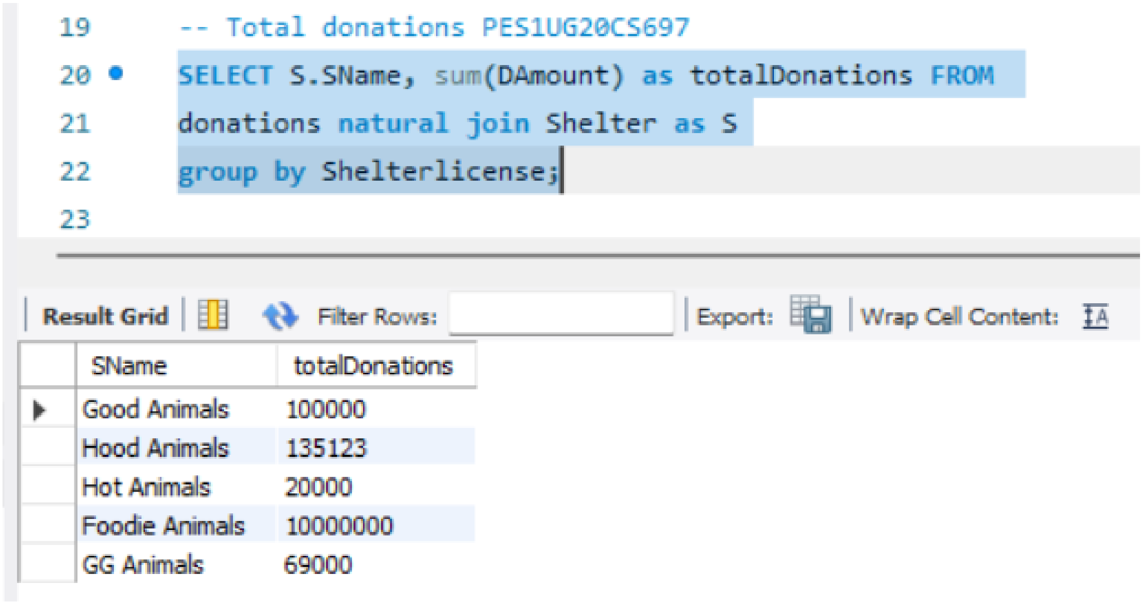


```
INSERT INTO animal(AnimalID, AName, Sex, PlaceOfOrigin, DeadOrAlive, Age, NeuterEmployeeID, EuthanizeEmployeeID, LivesShelterlicense, AdoptedSSN, Species, Breed)
VALUES
(3000, "Leaf", "Male", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',222563769, "Cat", "Street"),
(3788, "Butterlee", "Male", "Bengaluru", "Alive", 2, 441603835, NULL, 'mklwwhyn1i',222563769, "Cat", "Street");
```

```
Output
Action Output
 #  Time      Action                                                                                 Message                                                          Duration / Fetch
 1  12:13:25  INSERT INTO animal(AnimalID, AName, Sex, PlaceOfOrigin, DeadOrAlive, Age, NeuterEmployeeID, Euthan...  Error Code: 1644. One Person cannot adopt more than 3 animals  0.000 sec
```

SQL Queries

1. **Aggregate Query**
   Query – Display total donations received by each shelter

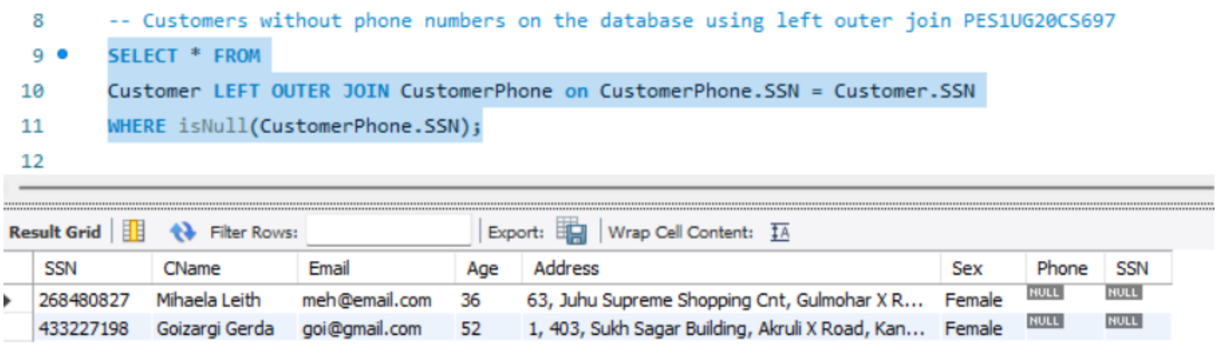


```sql
-- Total donations PES1UG20CS697
SELECT S.SName, sum(DAmount) as totalDonations FROM
donations natural join Shelter as S
group by Shelterlicense;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SName	totalDonations
Good Animals	100000
Hood Animals	135123
Hot Animals	20000
Foodie Animals	10000000
GG Animals	69000

2. **Outer join**
   Query – Display Customer details who's phone numbers are not available on the database.



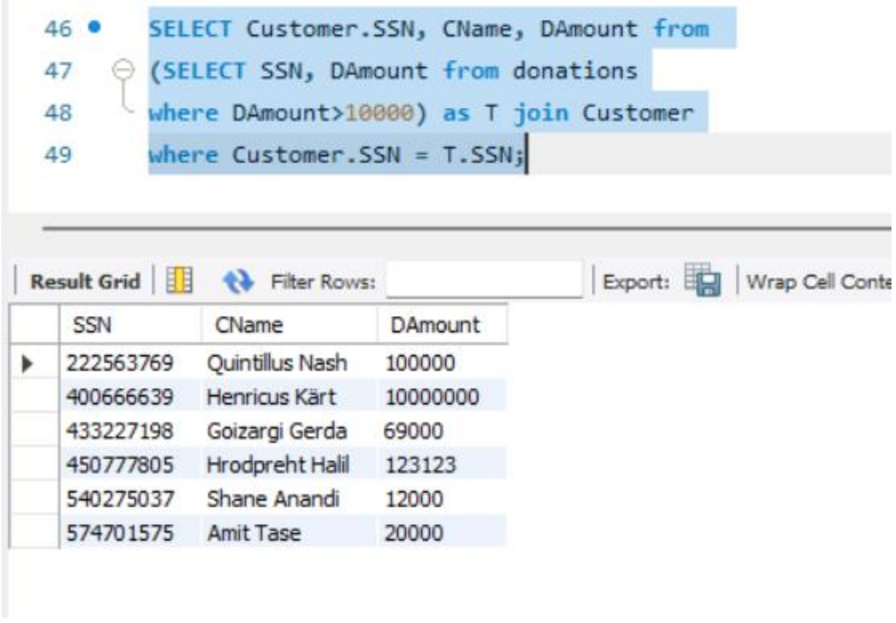
```sql
-- Customers without phone numbers on the database using left outer join PES1UG20CS697
SELECT * FROM
Customer LEFT OUTER JOIN CustomerPhone on CustomerPhone.SSN = Customer.SSN
WHERE isNull(CustomerPhone.SSN);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SSN	CName	Email	Age	Address	Sex	Phone	SSN
268480827	Mihaela Leith	meh@email.com	36	63, Juhu Supreme Shopping Cnt, Gulmohar X R...	Female	NULL	NULL
433227198	Goizargi Gerda	goi@gmail.com	52	1, 403, Sukh Sagar Building, Akruli X Road, Kan...	Female	NULL	NULL

3. **Nested Query**

Query – Display Name of the customer who donated more than 10000.



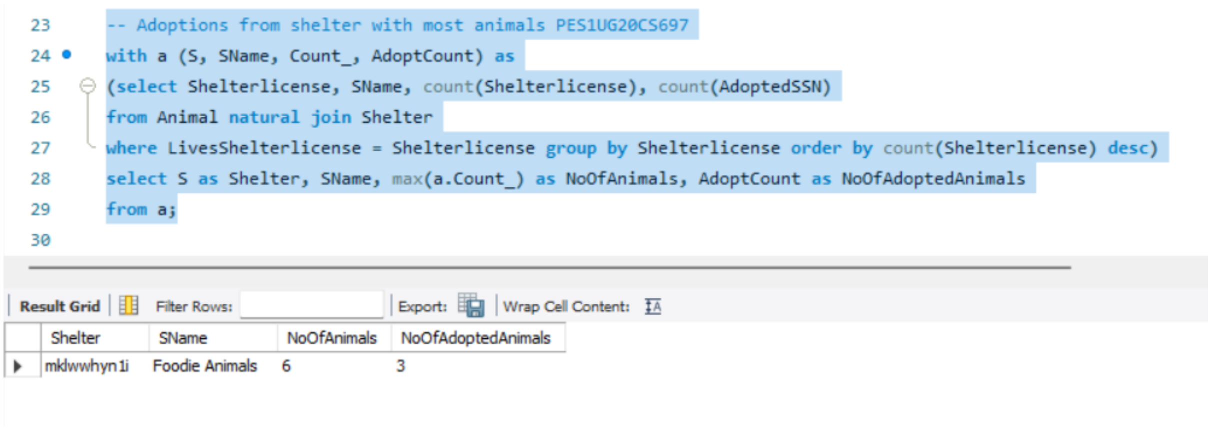
```
SELECT Customer.SSN, CName, DAmount from
(SELECT SSN, DAmount from donations
where DAmount>10000) as T join Customer
where Customer.SSN = T.SSN;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

SSN	CName	DAmount
222563769	Quintillus Nash	100000
400666639	Henricus Kärt	10000000
433227198	Goizargi Gerda	69000
450777805	Hrodpreht Halil	123123
540275037	Shane Anandi	12000
574701575	Amit Tase	20000

4.

Query – Display number of animals adopted from the Shelter with maximum number of animals.



```
-- Adoptions from shelter with most animals PES1UG20CS697
with a (S, SName, Count_, AdoptCount) as
(select Shelterlicense, SName, count(Shelterlicense), count(AdoptedSSN)
from Animal natural join Shelter
where LivesShelterlicense = Shelterlicense group by Shelterlicense order by count(Shelterlicense) desc)
select S as Shelter, SName, max(a.Count_) as NoOfAnimals, AdoptCount as NoOfAdoptedAnimals
from a;
```

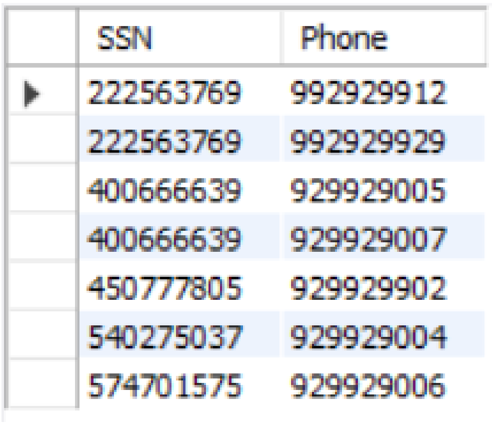
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Shelter	SName	NoOfAnimals	NoOfAdoptedAnimals
mklwwhyn1i	Foodie Animals	6	3

Execution Plan before and after

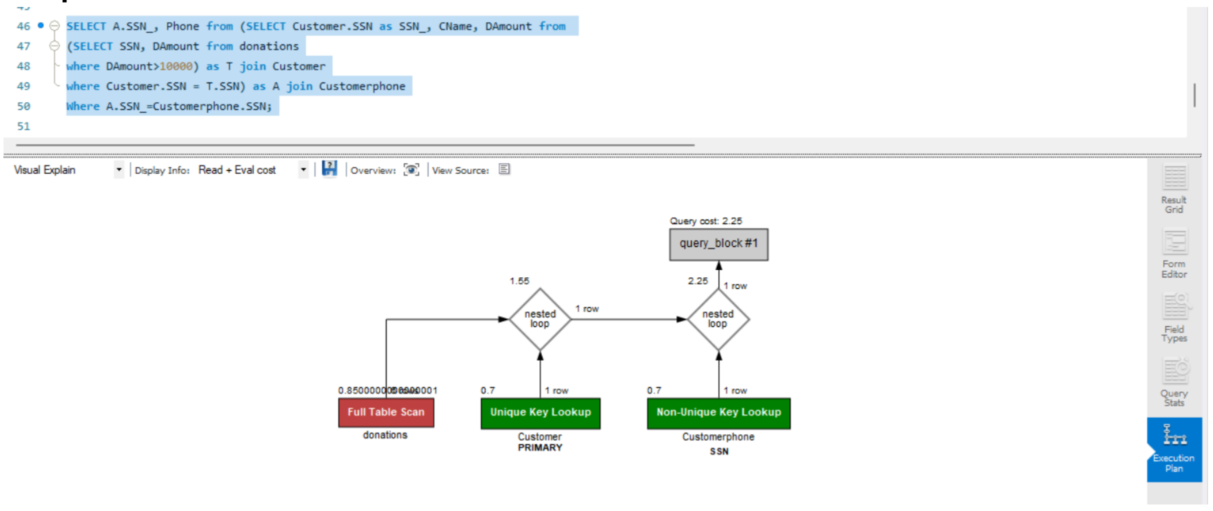
Query – Display SSN and Donation amount of customers who donated more than 10000

Output –



SSN	Phone
222563769	992929912
222563769	992929929
400666639	929929005
400666639	929929007
450777805	929929902
540275037	929929004
574701575	929929006

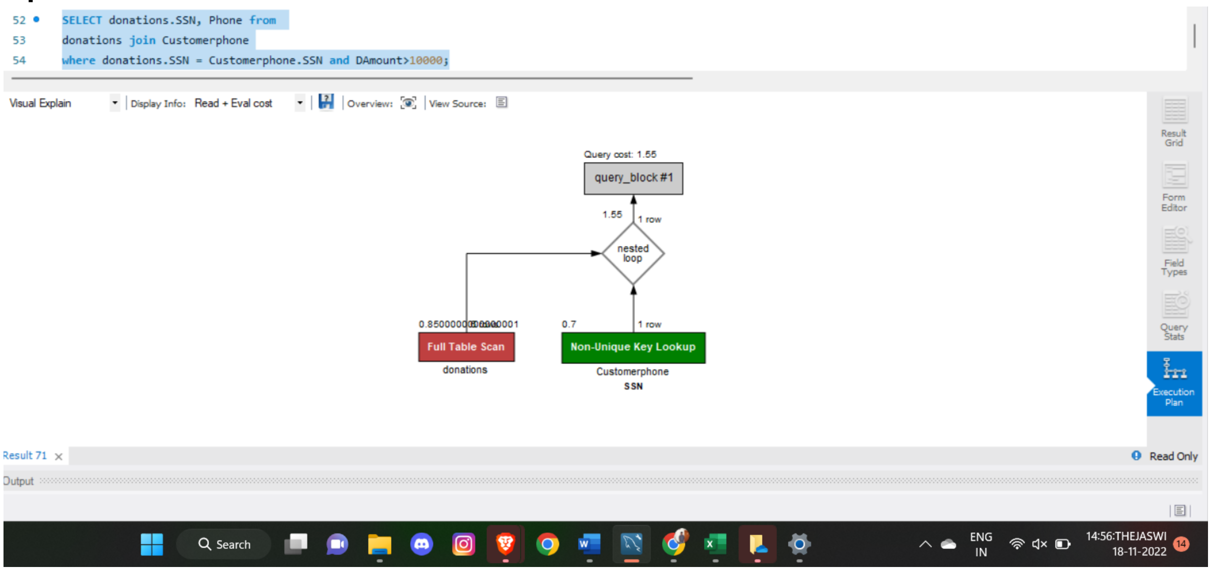
Unoptimized –



```
SELECT A.SSN_, Phone from (SELECT Customer.SSN as SSN_, CName, DAmount from
(SELECT SSN, DAmount from donations
where DAmount>10000) as T join Customer
where Customer.SSN = T.SSN) as A join Customerphone
Where A.SSN_=Customerphone.SSN;
```

Visual Explain | Display Info: Read + Eval cost

Optimized –



```
SELECT donations.SSN, Phone from
donations join Customerphone
where donations.SSN = Customerphone.SSN and DAmount>10000;
```

Visual Explain | Display Info: Read + Eval cost