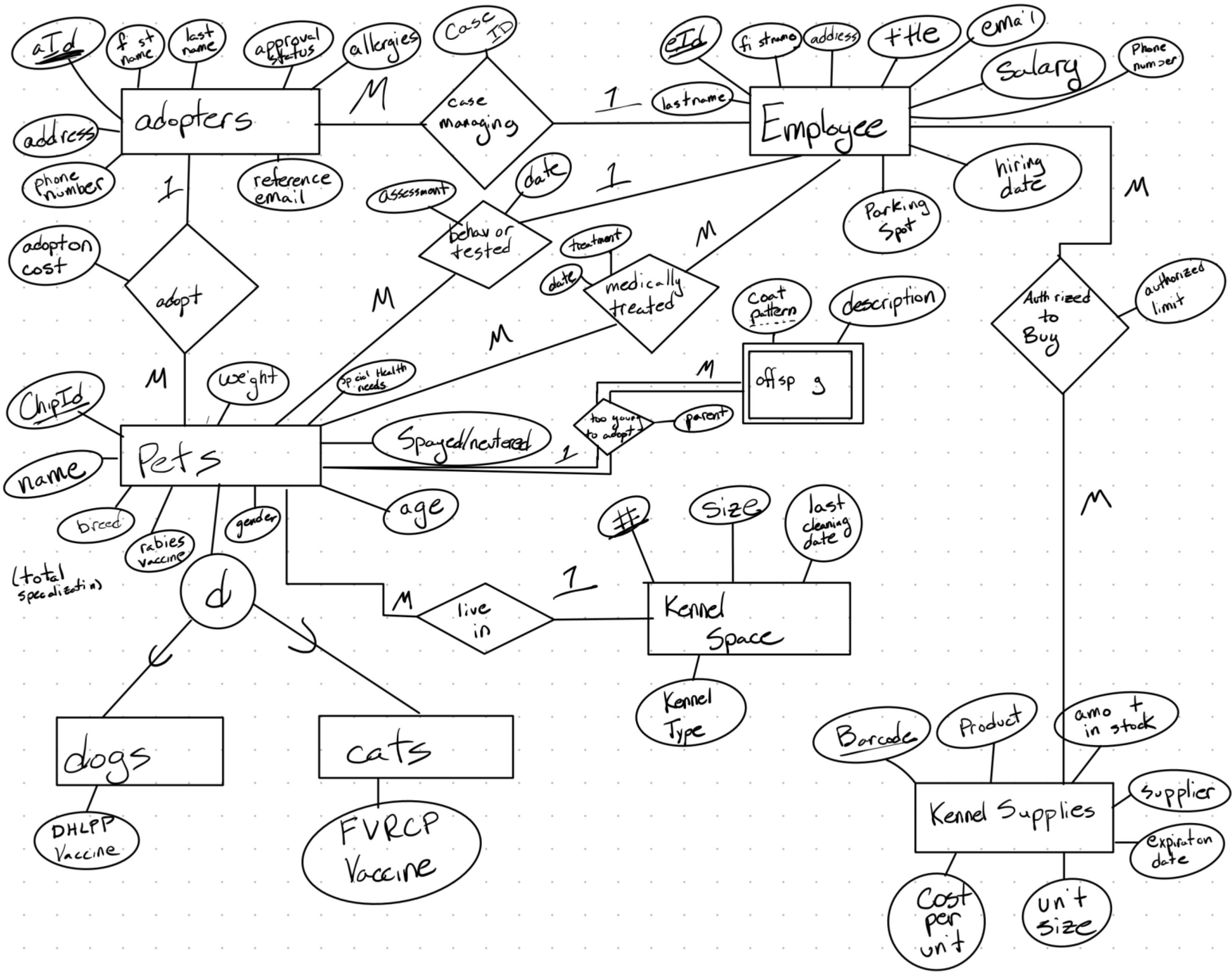
CS 4250: Animal Shelter Database Final Report

Group members: Xander Toepfer xtoepfer@csustan.edu, Kennedy Cunningham, kcunningham@csustan.edu, Melissa Villalovos mvillalovos1@csustan.edu

ER Diagram (Part 2)



Animal Shelter Database Explanation

Entities and Relationships

Employees contain the employees and the information about them. They are related to the adopters and 1 employee is assigned to each adopter as a case manager. The adopters have information regarding their identity, references, and contact information. The pets have information on their chip, age, weight, name, and other health information. The pets are sorted into cats and dogs, with no other kinds of animals housed at this shelter. Employees can be authorized to perform necessary medical procedures on pets, as well as behavior analysis to determine the possibility of adoption. Kennels are numbered and come in different types. The pets each have an assigned kennel. Supplies are identified by barcode and information is stored on them such as quantity, price, and supplier. Employees can be authorized to buy necessary supplies. The supplies include things like food and bowls for the kennels.

Constraints and Restrictions

Every employee must have a name associated with them in alphabet only and an employee ID entered as a positive integer value greater than 0. The names of employees, pets, and adopters must all be alphabet only. Employee salary must be a positive integer. The phone number for employees and adopters must be entered as a positive integer greater than 0. Every pets weight and age must be entered as a positive integer greater than 0. The amount of kennel supplies in stock and the cost per unit must be a positive integer greater than 0. The kennel space number must be a positive integer.

Summary/Relations (Part 3)

Summary

This project is to create a database that would be used by the managers of an animal shelter. This animal shelter would have cats and dogs to keep track of, along with where they are residing, potential adopters, and shelter supplies. Employees can be authorized to buy a certain number of supplies for the shelter. Adoption appointments can be held between employees and potential adopters to help them adopt a pet. The pets are sorted into cats and dogs, which can both potentially have offspring.

```
CREATE TABLE Employee
        ( eId VARCHAR(20),
        firstname VARCHAR(20),
        lastname VARCHAR (20),
        address VARCHAR (50),
        title VARCHAR (20),
        email VARCHAR (30),
        salary INTEGER,
        phoneNumber INTEGER (10),
        hiringDate DATE,
        parkingSpot INTEGER,
        PRIMARY KEY (eId));
```

```
CREATE TABLE Adopters

        ( aId VARCHAR (20),

        firstname VARCHAR(20),

        lastname VARCHAR (20),

        address VARCHAR (50),

        phoneNumber INTEGER (10),

        approvalStatus VARCHAR (10),

        allergies VARCHAR (50),

        referenceEmail VARCHAR (50),

        primary key (aId));


CREATE TABLE Kennel_Supplies

        (barcode INTEGER (20),

        product VARCHAR (30),

        unitSize VARCHAR (15),

        unitStock INTEGER,

        costPerUnit INTEGER,

        supplier VARCHAR (30),

        expireDate DATE,

        primary key (barcode));


CREATE TABLE Kennel_Space

        (kennelNum INTEGER (4),
	size VARCHAR (10),

	lastCleaning DATE,

	type VARCHAR (20),

	primary key (kennelNum));


CREATE TABLE Pets

	( chipId INTEGER (20),

	name VARCHAR (20),

	breed VARCHAR (20),

	weight INTEGER,

	sex VARCHAR (6).

	age INTEGER,

	rabiesVaccine DATE,

	spay/neuter DATE,

	specialNeeds VARCHAR (20),

	primary key (chipId));


CREATE TABLE Offspring

	(coatPattern VARCHAR (20),

	description VARCHAR (200),

	chipId INTEGER (20) NOT NULL,

	primary key (coatPattern, chipId),

	foreign key (chipId) references Pets
```

```
                on delete cascade);

CREATE TABLE dogs
        ( chipID INTEGER (20).
        DHLPP DATE,
        primary key (chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE cats
        ( chipID INTEGER (20).
        FVRCP DATE,
        primary key (chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE Case_managing
        (caseId INTEGER (20),
         eId VARCHAR(20),
        aId VARCHAR (20),
        primary key (caseId),
        foreign key (eId) references Employee,
        foreign key (aId) references Adopters);
```

```
CREATE TABLE Adopt

        ( aId VARCHAR (20),

        chipId INTEGER (20),

        adoptionCost DECIMAL (M, 2),

        primary key (aId),

        foreign key (aId) references Adopters,

        foreign key (chipId) references Pets);


CREATE TABLE Auth_to_buy

        ( eId VARCHAR(20),

        barcode VARCHAR(20),

        authorizedLimit DECIMAL (M, 2),

        primary key (eId, barcode),

        foreign key (eId) references Employee,

        foreign key (barcode) references Kennel_Supplies);


CREATE TABLE Med_treated

        ( eId VARCHAR (20),

        chipId INTEGER (20),

        date DATE,

        treatment VARCHAR (50),

        primary key (eId, chipId, treatment),
        foreign key (eId) references Employee,
        foreign key (chipId) references Pets);

CREATE TABLE Behavior_test
        ( eId VARCHAR (20),
        chipId INTEGER (20),
        date DATE,
        assessment VARCHAR (200),
        primary key (eId),
        foreign key (eId) references Employee,
        foreign key (chipId) references Pets);

CREATE TABLE Live_in
        ( chipId INTEGER (20),
        kennelNum INTEGER,
        primary key (kennelNum),
        foreign key (kennelNum) references Kennel_Space,
        foreign key (chipId) references Pets);
```

<u>Section 1: Initial Relations</u>

```
CREATE TABLE Employee
    ( eId VARCHAR(20),
    firstname VARCHAR(100),
    lastname VARCHAR (100),
    address VARCHAR (50),

    title VARCHAR (100),

    email VARCHAR (30),

    salary INTEGER,

    phoneNumber INTEGER (10),

    hiringDate DATE,

    parkingSpot INTEGER,

    PRIMARY KEY (eId));


CREATE TABLE Adopters

    ( aId VARCHAR (20),

    firstname VARCHAR(100),

    lastname VARCHAR (100),

    address VARCHAR (50),

    phoneNumber INTEGER (10),

    approvalStatus VARCHAR (10),

    allergies VARCHAR (50),

    referenceEmail VARCHAR (50),

    primary key (aId));


CREATE TABLE Kennel_Supplies

    (barcode INTEGER (20),

    product VARCHAR (30),
    unitSize VARCHAR (15),
    unitStock INTEGER,
    costPerUnit INTEGER,
    supplier VARCHAR (30),
    expireDate DATE,
    primary key (barcode));

CREATE TABLE Kennel_Space
    (kennelNum INTEGER (4),
    size VARCHAR (10),
    lastCleaning DATE,
    type VARCHAR (100),
    primary key (kennelNum));

CREATE TABLE Pets
    ( chipId INTEGER (20),
    name VARCHAR (100),
    breed VARCHAR (1000),
    weight INTEGER,
    sex VARCHAR (6).
    age INTEGER,
    rabiesVaccine DATE,
    spay/neuter DATE,
    specialNeeds VARCHAR (1000),
    primary key (chipId));

CREATE TABLE Offspring
    (coatPattern VARCHAR (100),
    description VARCHAR (1000),
    chipId INTEGER (20) NOT NULL,
    primary key (coatPattern, chipId),
    foreign key (chipId) references Pets
            on delete cascade);

CREATE TABLE dogs
    ( chipID INTEGER (20).
    DHLPP DATE,
    primary key (chipId),
    foreign key (chipId) references Pets
            on delete cascade);

CREATE TABLE cats
    ( chipID INTEGER (20).
    FVRCP DATE,
    primary key (chipId),
    foreign key (chipId) references Pets
```

```
                on delete cascade);

CREATE TABLE Case_managing
        (eId VARCHAR(20),
        aId VARCHAR (20),
        caseId INTEGER (20
        primary key (aId),
        foreign key (eId) references Employee,
        foreign key (aId) references Adopters);

CREATE TABLE Adopt
        ( aId VARCHAR (20),
        chipId INTEGER (20),
        adoptionCost DECIMAL (M, 2),
        primary key (chipId),
        foreign key (aId) references Adopters,
        foreign key (chipId) references Pets);

CREATE TABLE Auth_to_buy
        ( eId VARCHAR(20),
        barcode VARCHAR(20),
        authorizedLimit DECIMAL (M, 2),
        primary key (eId, barcode),
        foreign key (eId) references Employee,
        foreign key (barcode) references Kennel_Supplies);

CREATE TABLE Med_treated
        ( eId VARCHAR (20),
        chipId INTEGER (20),
        date DATE,
        treatment VARCHAR (1000),
        primary key (eId, chipId),
        foreign key (eId) references Employee,
        foreign key (chipId) references Pets);

CREATE TABLE Behavior_test
        (eId VARCHAR (20),
        chipId INTEGER (20),
        date DATE,
        assessment VARCHAR (1000),
        primary key (chipId),
        foreign key (eId) references Employee,
        foreign key (chipId) references Pets);

CREATE TABLE Live_in
        ( chipId INTEGER (20),
        kennelNum INTEGER,

        primary key (chipId),

        foreign key (kennelNum) references Kennel_Space,

        foreign key (chipId) references Pets);
```

Initial Relations/FDs/Normalization (Part 4)

Section 2: Functional Dependencies

Employee:

eId -> firstname
eId -> lastname
eId -> address
eId -> title
eId -> email
eId -> salary
eId -> phoneNumber
eId -> hiringDate
eId -> parkingSpot
email -> firstname
email -> lastname
email -> address
email -> title
email -> email
email -> salary
email -> phoneNumber
email -> hiringDate
email -> parkingSpot

Adopters:

aId -> firstname
aId -> lastname
aId -> address
aId -> phoneNumber
aId -> approvalStatus
aId -> allergies
aId -> referenceEmail

Kennel_Supplies:
barcode -> product
barcode -> unitSize
barcode -> unitStock
barcode -> costPerUnit
barcode -> supplier
barcode -> expireDate

Kennel_Space:
kennelNum -> size
kennelNum -> lastCleaning
kennelNum -> type

Pets:
chipId -> name
chipId -> breed
chipId -> weight
chipId -> sex
chipId -> age
chipId -> rabiesVaccine
chipId -> spay/neuter
chipId -> specialNeeds

Offspring:
chipId, coatPattern -> description

Dogs:
chipId -> DHLPP

Cats:
chipId -> FVRCP

Case_managing:
caseId ->eId
caseId ->aId

Adopt:
aId, chipId -> adoptionCost

Auth_to_buy:
eId, barcode -> authorizedLimit

Med_treated:
eId, chipId, treatment -> date

eId, chipId, date -> treatment

Behavior_test:
eId, chipId -> date
eId, chipId -> assessment

Live_in:
chipId -> kennelNum

Section 3: Normalization

Table Employee is in BCNF because there are no lists within the attribute fields, the primary key is not dependent on any other attribute.. It is also in 3NF due to the lack of transitive dependency.

Table Adopters is in BCNF because the primary key is not derived from other attributes. This table is also in 3NF because there is not any transitive dependencies.

Table Kennel_Supplies is in BCNF because every attribute is dependent on the respective primary key. It is also in 3NF because there isn't transitive dependency.

Table Kennel_space is in BCNF because there are no lists within the attribute fields, the primary key is not dependent on any other attribute. It is also in 3NF due to the lack of transitive dependency.

Table Pets is in BCNF because the primary key is not derived from other attributes. This table is also in 3NF because there are not any transitive dependencies.

Table Offspring is in BCNF because every attribute is associated with the primary key. It is also in 3NF as that is required to qualify the table to be in BCNF as well as the lack of transitive dependency.

Table dogs is in BCNF because every attribute is dependent on the respective primary key. It is also in 3NF because there isn't transitive dependency.

Table cats is in BCNF because there are no lists within the attribute fields, the primary key is not dependent on anything other. It is also in 3NF due to the lack of transitive dependency.

Table Case_managing is in BCNF because every attribute is associated with the primary key. It is also in 3NF as that is required to qualify the table to be in BCNF as well as the lack of transitive dependency.

Table Adopt is in BCNF because the primary key is not derived from other attributes. This table is also in 3NF because there are not any transitive dependencies.

Table Auth_to_buy is in BCNF because every attribute is associated with the primary key. It is also in 3NF as that is required to qualify the table to be in BCNF as well as the lack of transitive dependency.

Table Med_treated is in BCNF because is in BCNF because every attribute is dependent on the respective primary key. It is also in 3NF because there isn't transitive dependency.

Table Behavior_test is in BCNF because there are no lists within the attribute fields and the primary key is not dependent on any other attributes. It is also in 3NF due to the lack of transitive dependency.

Table Live_in is in BCNF the primary key is not derived from other attributes. This table is also in 3NF because there are not any transitive dependencies.

SQL Database Schema/Table Definitions/Data (Part 5)

Part 1: Relation Schemas from Part 4

CREATE TABLE Employee

(eId VARCHAR(20),

firstname VARCHAR(100),

lastname VARCHAR (100),

address VARCHAR (50),

title VARCHAR (100),

email VARCHAR (30),

salary INTEGER,

phoneNumber INTEGER (10),

hiringDate DATE,

parkingSpot INTEGER,

```
        PRIMARY KEY (eId));

CREATE TABLE Adopters
        ( aId VARCHAR (20),
        firstname VARCHAR(100),
        lastname VARCHAR (100),
        address VARCHAR (50),
        phoneNumber INTEGER (10),
        approvalStatus VARCHAR (10),
        allergies VARCHAR (50),
        referenceEmail VARCHAR (50),
        primary key (aId));

CREATE TABLE Kennel_Supplies
        (barcode INTEGER (20),
        product VARCHAR (30),
        unitSize VARCHAR (15),
        unitStock INTEGER,
        costPerUnit INTEGER,
        supplier VARCHAR (30),
        expireDate DATE,
        primary key (barcode));
```

```
CREATE TABLE Kennel_Space
        (kennelNum INTEGER (4),
        size VARCHAR (10),
        lastCleaning DATE,
        type VARCHAR (100),
        primary key (kennelNum));

CREATE TABLE Pets
        ( chipId INTEGER (20),
        name VARCHAR (100),
        breed VARCHAR (1000),
        weight INTEGER,
        sex VARCHAR (6).
        age INTEGER,
        rabiesVaccine DATE,
        spay/neuter DATE,
        specialNeeds VARCHAR (1000),
        primary key (chipId));

CREATE TABLE Offspring
        (coatPattern VARCHAR (100),
        description VARCHAR (1000),
        chipId INTEGER (20) NOT NULL,
        primary key (coatPattern, chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE dogs
        ( chipID INTEGER (20).
        DHLPP DATE,
        primary key (chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE cats
        ( chipID INTEGER (20).
        FVRCP DATE,
        primary key (chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE Case_managing
        (eId VARCHAR(20),
        aId VARCHAR (20),
        caseId INTEGER (20
        primary key (aId),
        foreign key (eId) references Employee,
        foreign key (aId) references Adopters);

CREATE TABLE Adopt
        ( aId VARCHAR (20),
        chipId INTEGER (20),
        adoptionCost DECIMAL (M, 2),
        primary key (chipId),
        foreign key (aId) references Adopters,
        foreign key (chipId) references Pets);

CREATE TABLE Auth_to_buy
        ( eId VARCHAR(20),
        barcode VARCHAR(20),
        authorizedLimit DECIMAL (M, 2),
        primary key (eId, barcode),
        foreign key (eId) references Employee,
        foreign key (barcode) references Kennel_Supplies);

CREATE TABLE Med_treated
        ( eId VARCHAR (20),
        chipId INTEGER (20),
        date DATE,
	treatment VARCHAR (1000),
	primary key (eId, chipId),
	foreign key (eId) references Employee,
	foreign key (chipId) references Pets);

CREATE TABLE Behavior_test
	(eId VARCHAR (20),
	chipId INTEGER (20),
	date DATE,
	assessment VARCHAR (1000),
	primary key (chipId),
	foreign key (eId) references Employee,
	foreign key (chipId) references Pets);

CREATE TABLE Live_in
	( chipId INTEGER (20),
	kennelNum INTEGER,
	primary key (chipId),
	foreign key (kennelNum) references Kennel_Space,
	foreign key (chipId) references Pets);
```

Part 2: SQL Create Table Commands for Part 5 and notes on minor changes

```
CREATE TABLE `xtoepfer`.`Employee` (`eId` INT NOT NULLAUTO_INCREMENT , `firstName` VARCHAR(100) NOT NULL , `lastName`VARCHAR(100) NOT NULL , `address` VARCHAR(100) NOT NULL , `title`VARCHAR(100) NOT NULL , `email` VARCHAR(100) NOT NULL , `salary`INT NOT NULL , `phoneNumber` BIGINT NOT NULL , `hiringDate` DATE NOTNULL , `parkingSpot` INT NOT NULL , PRIMARY KEY (`eId`)) ENGINE =InnoDB;, PRIMARY KEY (`eId`)) ENGINE =InnoDB;
```

Changed eID to int to allow auto inc. Extended data length to 100 on varchar fields

Phone number became bigint to fit data

```
CREATE TABLE `xtoepfer`.`Adopters` (`aId` INT NOT NULLAUTO_INCREMENT , `firstName` VARCHAR(100) NOT NULL , `lastName`VARCHAR(100) NOT NULL , `address` VARCHAR(100) NOT NULL ,`phoneNumber` BIGINT NOT NULL , `approvalStatus` VARCHAR(10) NOT NULL, `allergies` VARCHAR(100) NULL DEFAULT NULL , `referenceEmail`VARCHAR(100) NOT NULL , PRIMARY KEY (`aId`)) ENGINE = InnoDB;
```

Same changes as employee

Phone number became bigint to fit data

```
CREATE TABLE `xtoepfer`.`Kennel_Supplies` (`barcode` INT NOT NULL ,`product` VARCHAR(100) NOT NULL , `unitSize` VARCHAR(15) NOT NULL ,`unitStock` INT NOT NULL , `costPerUnit` DECIMAL(M,2) NOT NULL , `supplier`VARCHAR(100) NOT NULL , `expireDate` DATE NOT NULL , PRIMARY KEY(`barcode`)) ENGINE = InnoDB;
```

Same as above

```
CREATE TABLE `xtoepfer`.`Kennel_Space` (`kennelNum` INT NOT NULL ,`size` VARCHA
R(10) NOT NULL , `lastCleaning` DATE NULL , `type`VARCHAR(100) NOT NULL , PRIM
ARY KEY (`kennelNum`)) ENGINE = InnoDB;
```

```
CREATE TABLE `xtoepfer`.`Pets` (`chipId` INT NOT NULLAUTO_INCREMENT , `name` V
ARCHAR(100) NOT NULL , `breed`VARCHAR(1000) NOT NULL , `weightInPounds` INT N
OT NULL , `sex`VARCHAR(6) NOT NULL , `age` INT NOT NULL , `rabiesVaccine` DATEN
ULL , `spay/neuter` DATE NULL , `specialNeeds` VARCHAR(1000) NULL, PRIMARY KEY
(`chipId`)) ENGINE = InnoDB;
```

Weight has become weightInPounds

```
CREATE TABLE `xtoepfer`.`Offspring` (`coatPattern` VARCHAR(100) NOTNULL , `chipId` I
NT NOT NULL , `description` VARCHAR(1000) NOT NULL, PRIMARY KEY (`coatPattern`,
 `chipId`)) ENGINE = InnoDB;
ALTER TABLE `Offspring` ADD  FOREIGN KEY (`chipId`) REFERENCES`Pets`(`chipId`) O
N DELETE CASCADE ON UPDATE CASCADE;
```

```
CREATE TABLE `xtoepfer`.`Dogs` (`chipId` INT NOT NULL , `DHLPP` DATENULL , PRIM
ARY KEY (`chipId`)) ENGINE = InnoDB;
ALTER TABLE `Dogs` ADD  FOREIGN KEY (`chipId`) REFERENCES`Pets`(`chipId`) ON D
ELETE CASCADE ON UPDATE CASCADE;
```

CREATE TABLE `xtoepfer`.`Cats` (`chipId` INT NOT NULL , `FVRCP` DATENULL , PRIM ARY KEY (`chipId`)) ENGINE = InnoDB;

ALTER TABLE `Cats` ADD FOREIGN KEY (`chipId`) REFERENCES`Pets`(`chipId`) ON D ELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE `xtoepfer`.`Case_Managing` (`eId` INT NOT NULL , `aId`INT NOT NULL , `caseId` INT NOT NULL , PRIMARY KEY (`aId`)) ENGINE =InnoDB;

ALTER TABLE `Case_Managing` ADD FOREIGN KEY (`aId`) REFERENCES`Adopters`(`aI d`) ON DELETE CASCADE ON UPDATE CASCADE; ALTER TABLE`Case_Managing` AD D FOREIGN KEY (`eId`) REFERENCES `Employee`(`eId`)ON DELETE CASCADE ON UP DATE CASCADE;

All Id types are of type int

CREATE TABLE `xtoepfer`.`Adopt` (`aId` INT NOT NULL , `chipId` INTNOT NULL , `adopt ionCost` DECIMAL(M,2) NOT NULL DEFAULT '0.00' ,PRIMARY KEY (`chipId`)) ENGINE = InnoDB;

CREATE TABLE `xtoepfer`.`Adopt` (`aId` INT NOT NULL , `chipId` INTNOT NULL , `adopt ionCost` DECIMAL(8,2) NOT NULL DEFAULT '0.00' ,PRIMARY KEY (`chipId`)) ENGINE = InnoDB;

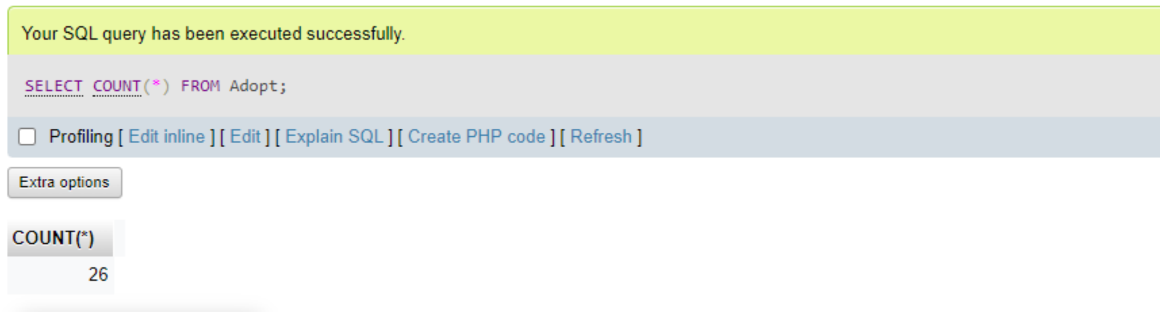
```
ALTER TABLE `Adopt` ADD  FOREIGN KEY (`aId`) REFERENCES`Adopters`(`aId`) ON D
ELETE CASCADE ON UPDATE CASCADE; ALTER TABLE`Adopt` ADD  FOREIGN KEY
(`chipId`) REFERENCES `Pets`(`chipId`) ONDELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE `xtoepfer`.`Auth_to_Buy` (`eId` INT NOT NULL ,`barcode` INT NOT NUL
L , `authorizedLimit` DECIMAL(10,2) NOT NULLDEFAULT '0.00' , PRIMARY KEY (`eId`, `
barcode`)) ENGINE = InnoDB;
ALTER TABLE `Auth_to_Buy` ADD  FOREIGN KEY (`eId`) REFERENCES`Employee`(`eId`
) ON DELETE CASCADE ON UPDATE CASCADE; ALTER TABLE`Auth_to_Buy` ADD  F
OREIGN KEY (`barcode`) REFERENCES`Kennel_Supplies`(`barcode`) ON DELETE CASCA
DE ON UPDATE CASCADE;

CREATE TABLE `xtoepfer`.`Med_Treated` (`eId` INT NOT NULL , `chipId`INT NOT NULL ,
 `date` DATE NOT NULL , `treatment` VARCHAR(1000) NOTNULL , PRIMARY KEY (`eId`
, `chipId`)) ENGINE = InnoDB;
ALTER TABLE `Med_Treated` ADD  FOREIGN KEY (`eId`) REFERENCES`Employee`(`eId`
) ON DELETE CASCADE ON UPDATE CASCADE; ALTER TABLE`Med_Treated` ADD  F
OREIGN KEY (`chipId`) REFERENCES `Pets`(`chipId`)ON DELETE CASCADE ON UPDAT
E CASCADE;

CREATE TABLE `xtoepfer`.`Behavior_Test` (`eId` INT NOT NULL ,`chipId` INT NOT NULL
 , `date` DATE NOT NULL , `assessment`VARCHAR(1000) NOT NULL , PRIMARY KEY (`c
hipId`)) ENGINE = InnoDB;
```

ALTER TABLE `Behavior_Test` ADD FOREIGN KEY (`eId`) REFERENCES`Employee`(`eI d`) ON DELETE CASCADE ON UPDATE CASCADE; ALTER TABLE`Behavior_Test` ADD FOREIGN KEY (`chipId`) REFERENCES`Pets`(`chipId`) ON DELETE CASCADE ON UPD ATE CASCADE;

CREATE TABLE `xtoepfer`.`Live_In` (`chipId` INT NOT NULL ,`kennelNum` INT NOT NUL L , PRIMARY KEY (`chipId`)) ENGINE = InnoDB;

ALTER TABLE `Live_In` ADD FOREIGN KEY (`kennelNum`) REFERENCES`Kennel_Spac e`(`kennelNum`) ON DELETE CASCADE ON UPDATE CASCADE;ALTER TABLE `Live_In ` ADD FOREIGN KEY (`chipId`) REFERENCES`Pets`(`chipId`) ON DELETE CASCADE O N UPDATE CASCADE;

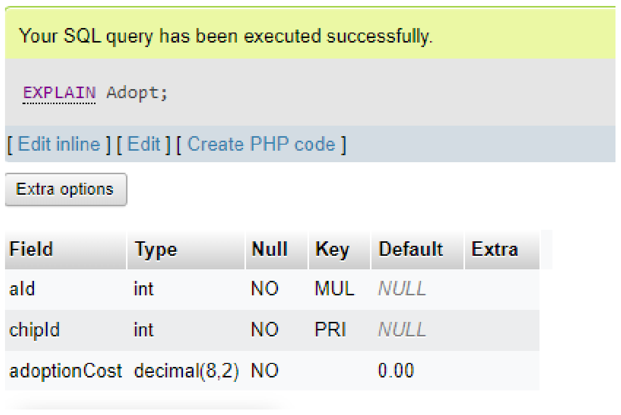
Part 3: Screenshots of explain command and output of select count(*)



Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Adopt;
```

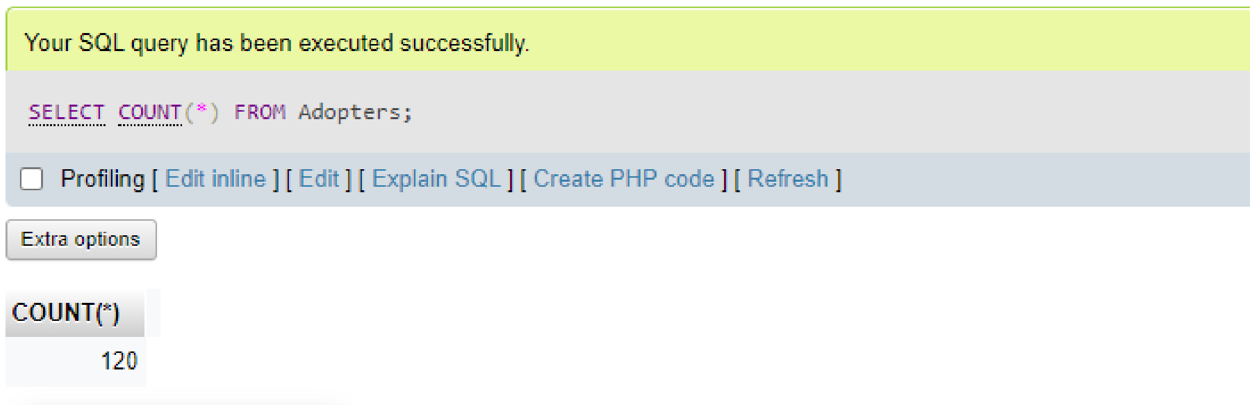
COUNT(*)
26



Your SQL query has been executed successfully.

```
EXPLAIN Adopt;
```

Field	Type	Null	Key	Default	Extra
aId	int	NO	MUL	NULL	
chipId	int	NO	PRI	NULL	
adoptionCost	decimal(8,2)	NO		0.00	

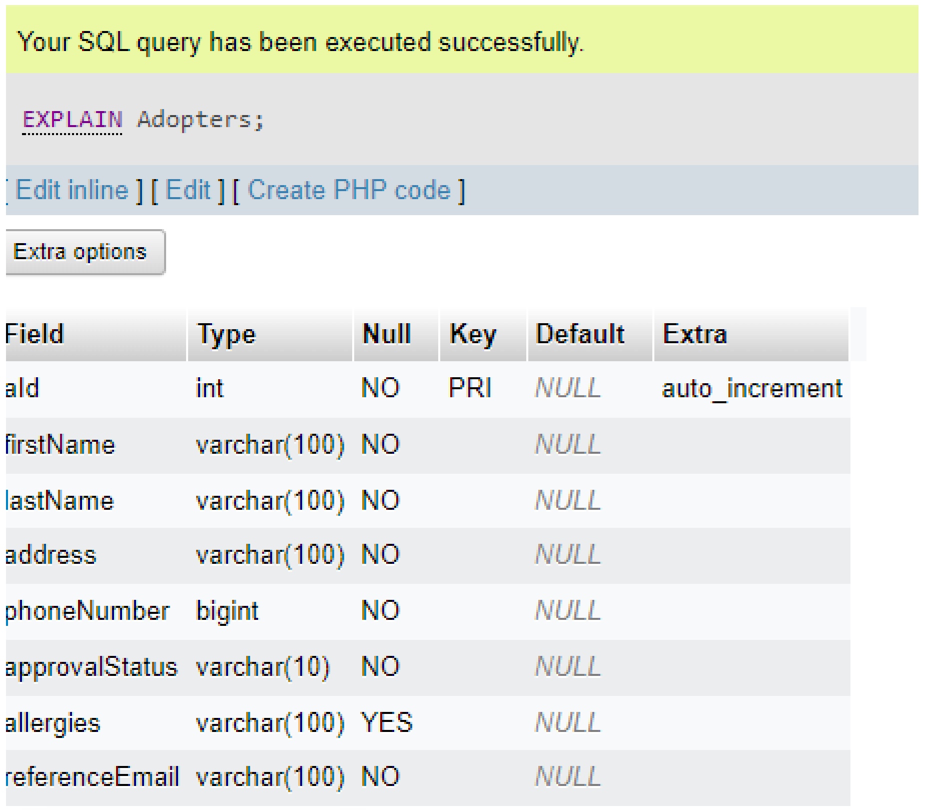

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Adopters;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

COUNT(*)
120

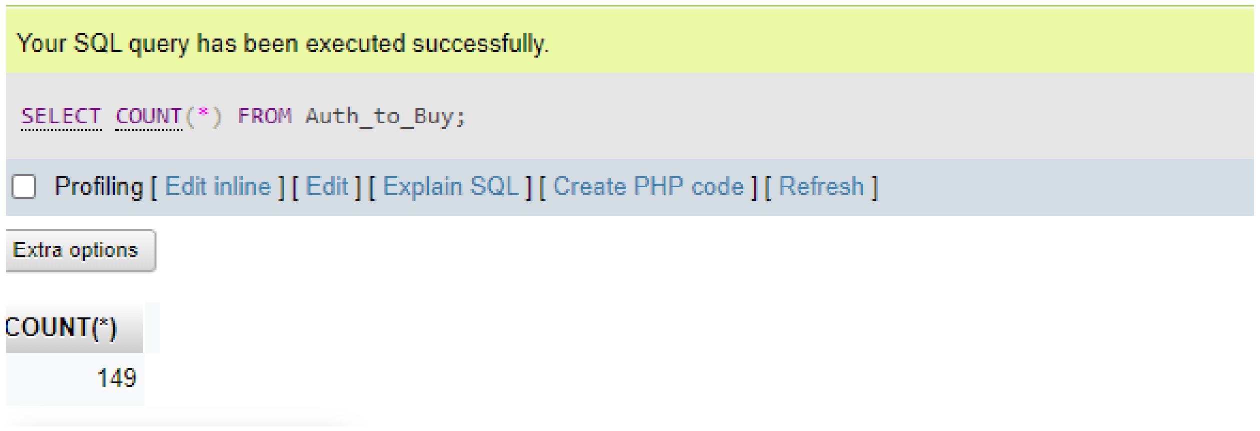

Your SQL query has been executed successfully.

```
EXPLAIN Adopters;
```

[Edit inline] [Edit] [Create PHP code]

Extra options

Field	Type	Null	Key	Default	Extra
aId	int	NO	PRI	*NULL*	auto_increment
firstName	varchar(100)	NO		*NULL*	
lastName	varchar(100)	NO		*NULL*	
address	varchar(100)	NO		*NULL*	
phoneNumber	bigint	NO		*NULL*	
approvalStatus	varchar(10)	NO		*NULL*	
allergies	varchar(100)	YES		*NULL*	
referenceEmail	varchar(100)	NO		*NULL*	

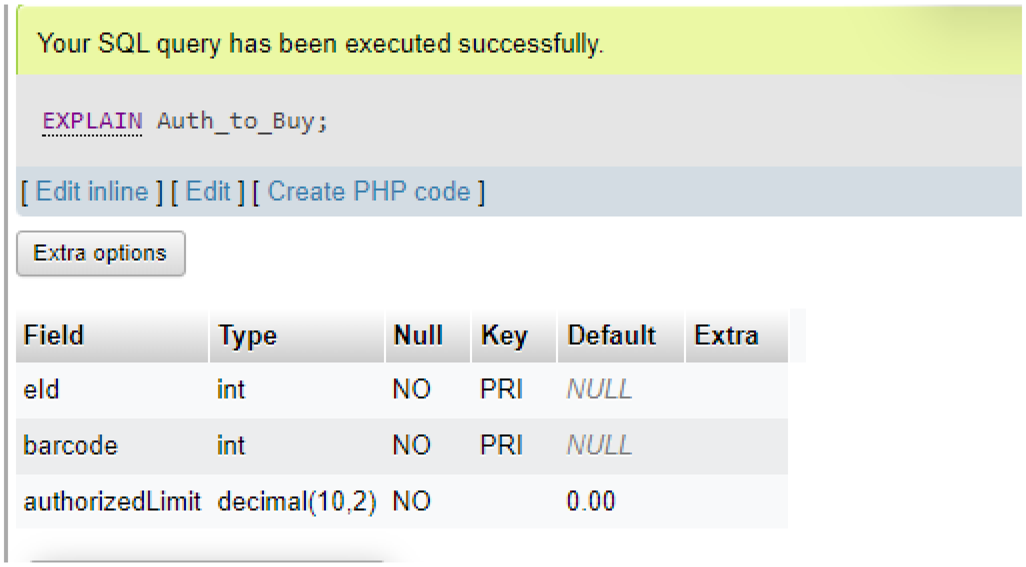

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Auth_to_Buy;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

COUNT(*)
149

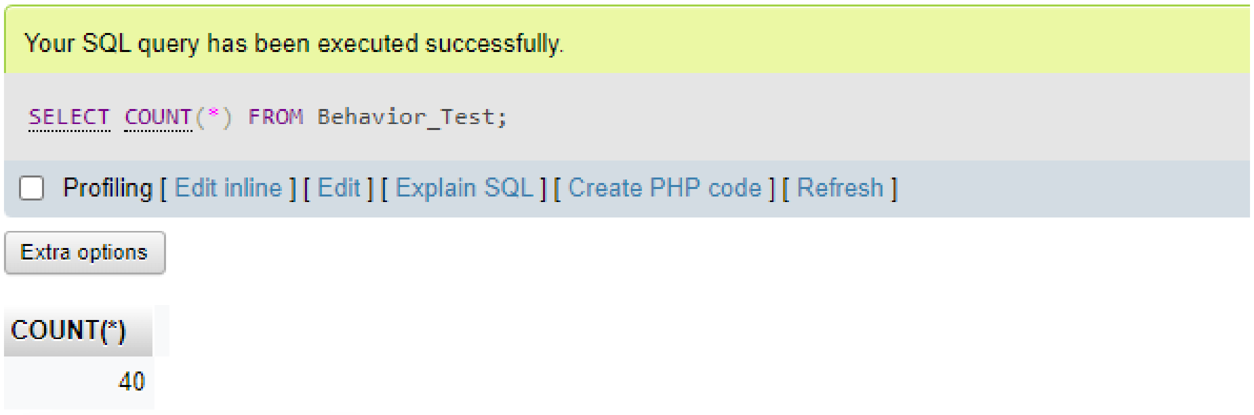

Your SQL query has been executed successfully.

```
EXPLAIN Auth_to_Buy;
```

[Edit inline] [Edit] [Create PHP code]

Extra options

Field	Type	Null	Key	Default	Extra
eId	int	NO	PRI	NULL	
barcode	int	NO	PRI	NULL	
authorizedLimit	decimal(10,2)	NO		0.00	

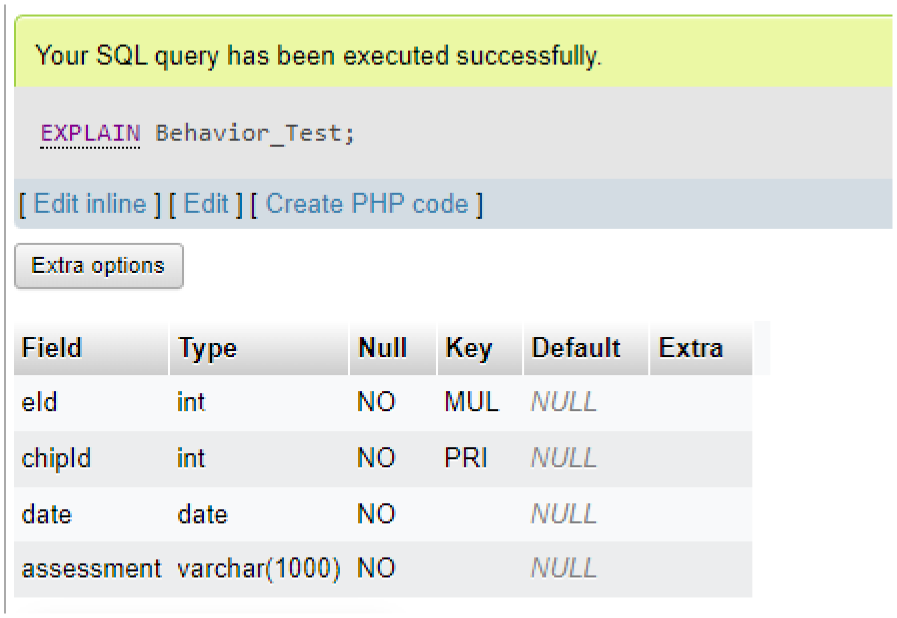

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Behavior_Test;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

COUNT(*)
40

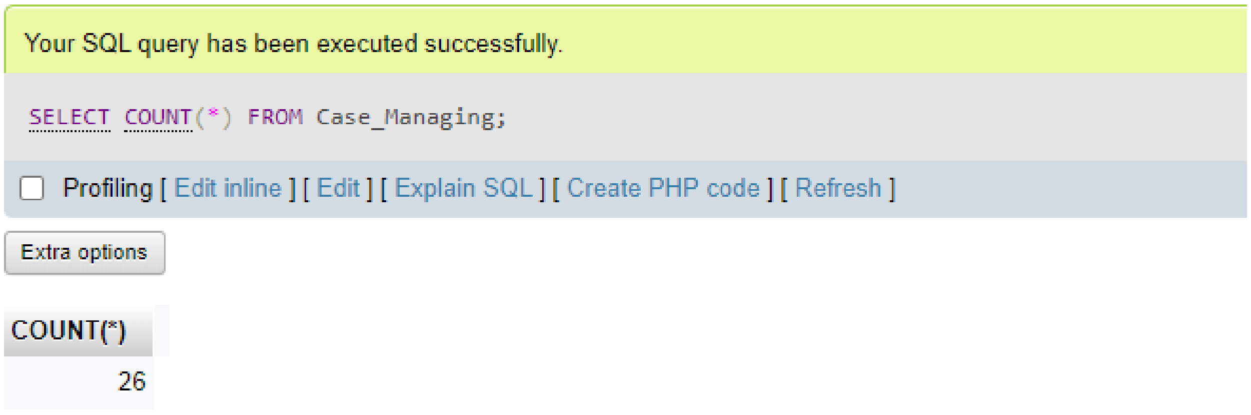

Your SQL query has been executed successfully.

```
EXPLAIN Behavior_Test;
```

[Edit inline] [Edit] [Create PHP code]

Extra options

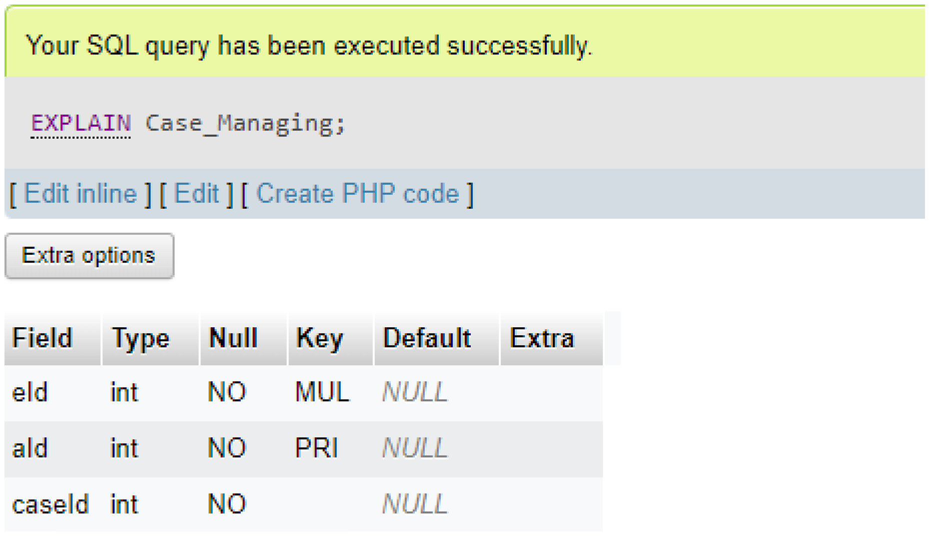
Field	Type	Null	Key	Default	Extra
eId	int	NO	MUL	NULL	
chipId	int	NO	PRI	NULL	
date	date	NO		NULL	
assessment	varchar(1000)	NO		NULL	


Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Case_Managing;
```

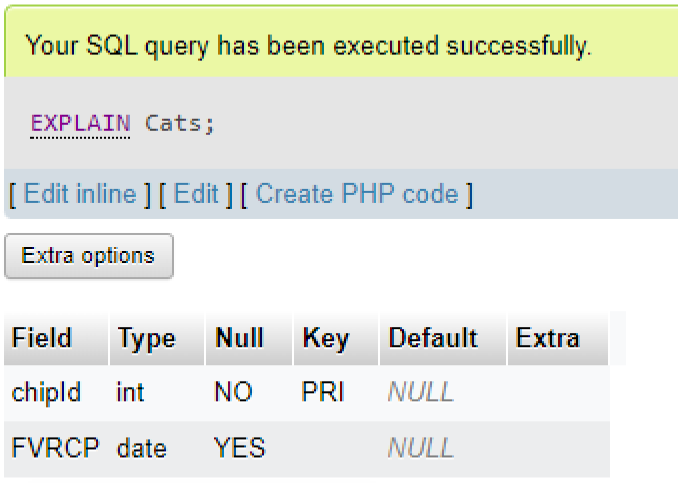
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options

COUNT(*)
26


Your SQL query has been executed successfully.

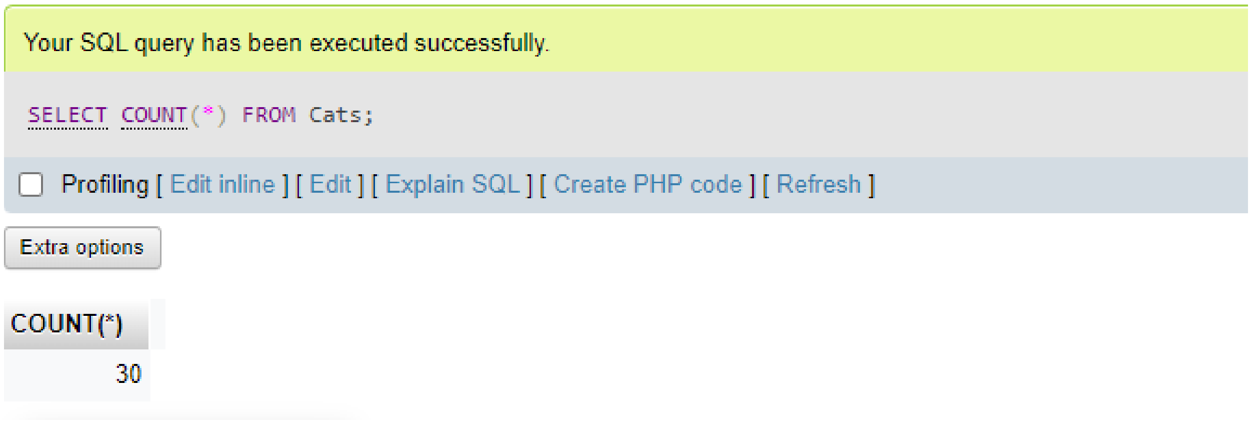
```
EXPLAIN Case_Managing;
```

Field	Type	Null	Key	Default	Extra
eId	int	NO	MUL	NULL	
aId	int	NO	PRI	NULL	
caseId	int	NO		NULL	


Your SQL query has been executed successfully.

```
EXPLAIN Cats;
```

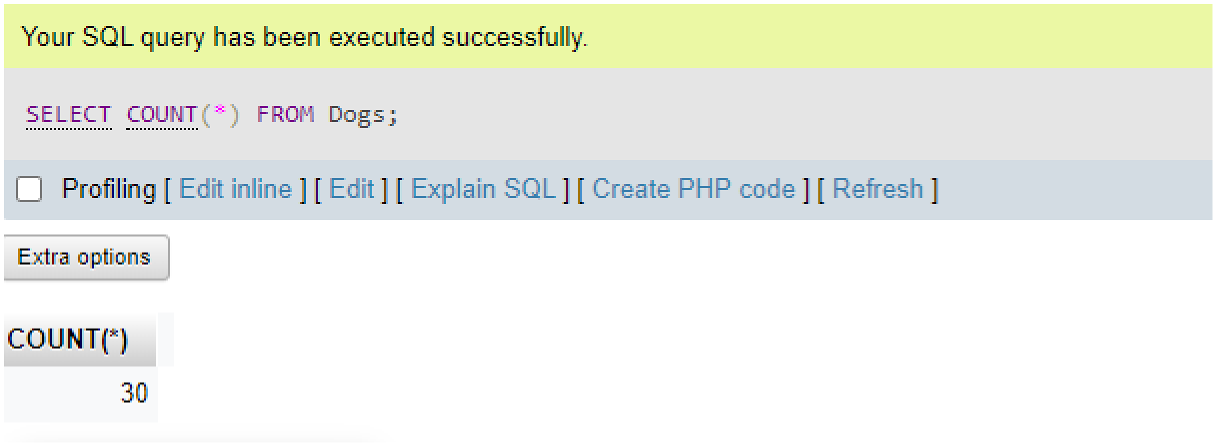
Field	Type	Null	Key	Default	Extra
chipId	int	NO	PRI	NULL	
FVRCP	date	YES		NULL	


Your SQL query has been executed successfully.

```sql
SELECT COUNT(*) FROM Cats;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options

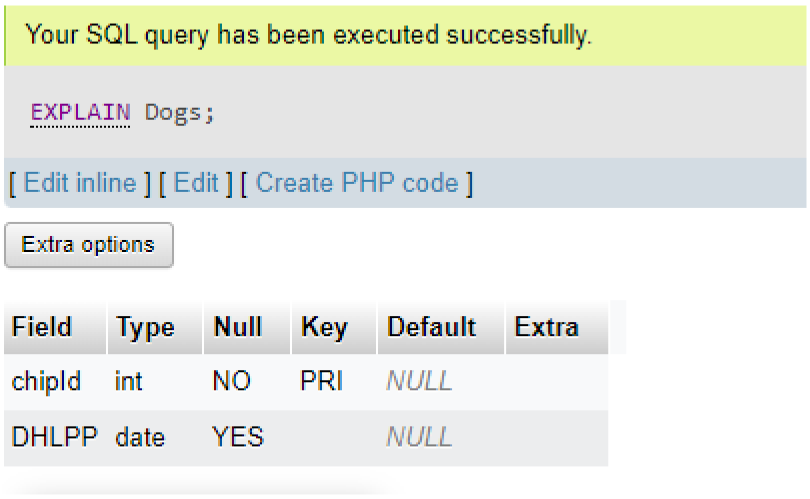
COUNT(*)
30


Your SQL query has been executed successfully.

```sql
SELECT COUNT(*) FROM Dogs;
```

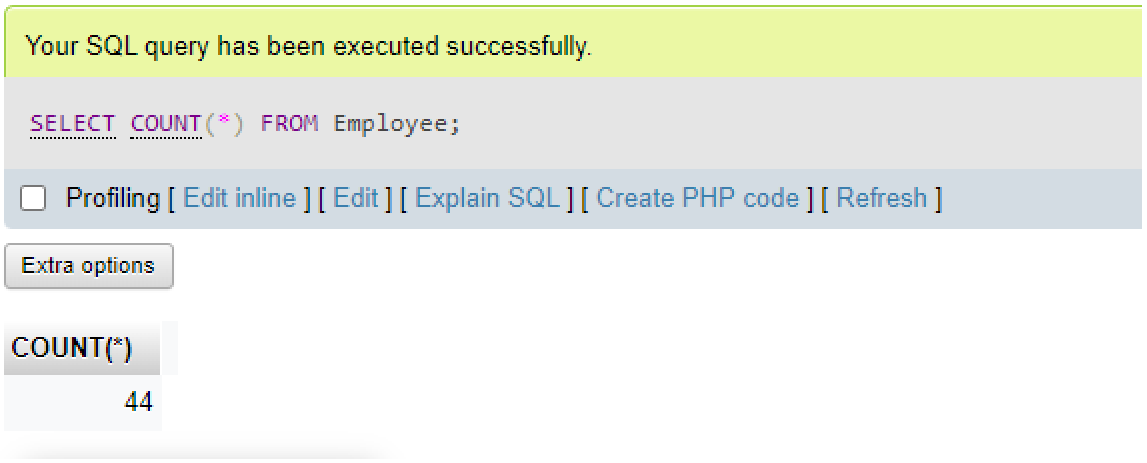
☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options

COUNT(*)
30


Your SQL query has been executed successfully.

```sql
EXPLAIN Dogs;
```

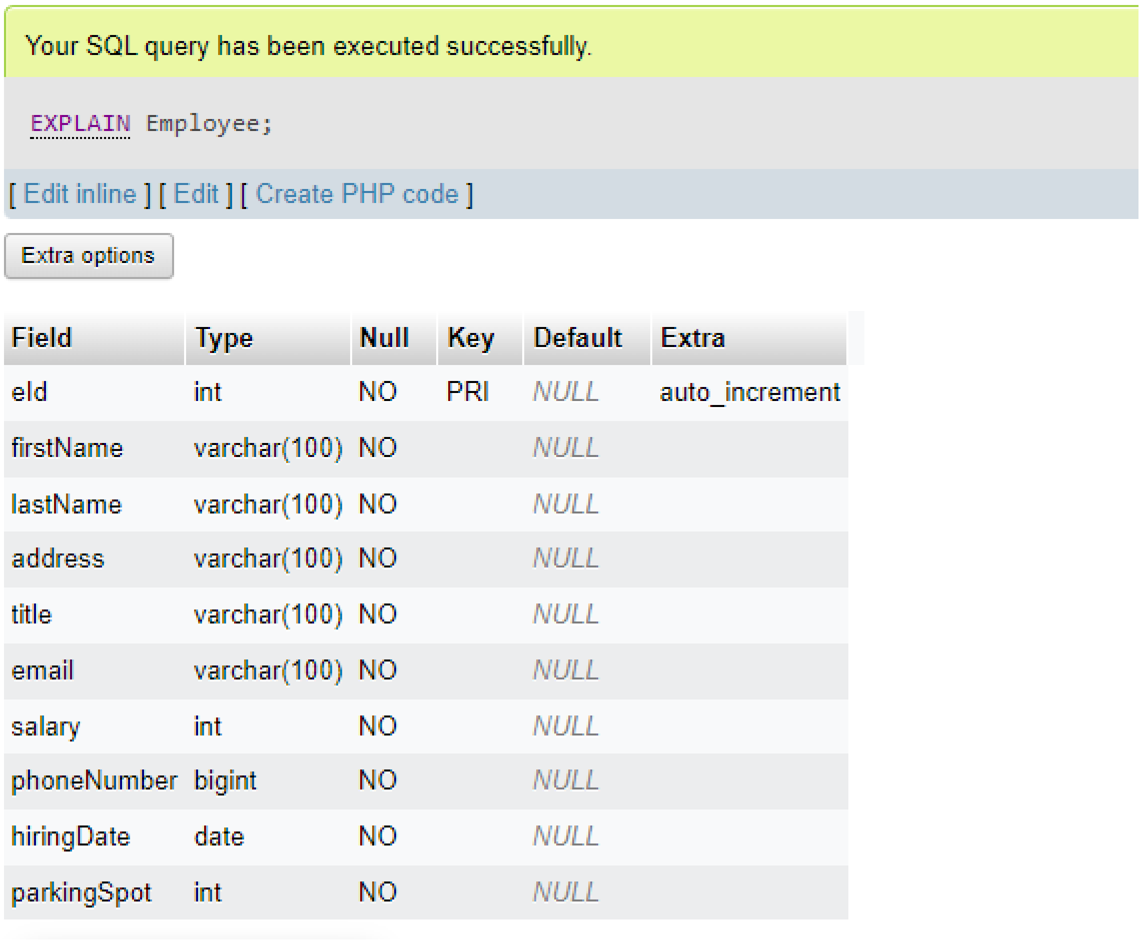
Field	Type	Null	Key	Default	Extra
chipId	int	NO	PRI	*NULL*	
DHLPP	date	YES		*NULL*	


Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Employee;
```

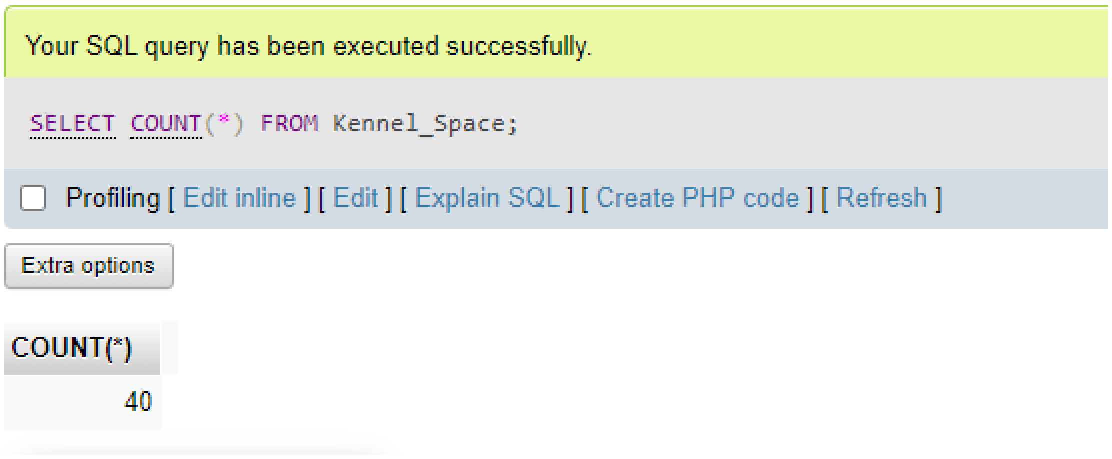
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options

COUNT(*)
44


Your SQL query has been executed successfully.

```
EXPLAIN Employee;
```

Field	Type	Null	Key	Default	Extra
eId	int	NO	PRI	NULL	auto_increment
firstName	varchar(100)	NO		NULL	
lastName	varchar(100)	NO		NULL	
address	varchar(100)	NO		NULL	
title	varchar(100)	NO		NULL	
email	varchar(100)	NO		NULL	
salary	int	NO		NULL	
phoneNumber	bigint	NO		NULL	
hiringDate	date	NO		NULL	
parkingSpot	int	NO		NULL	

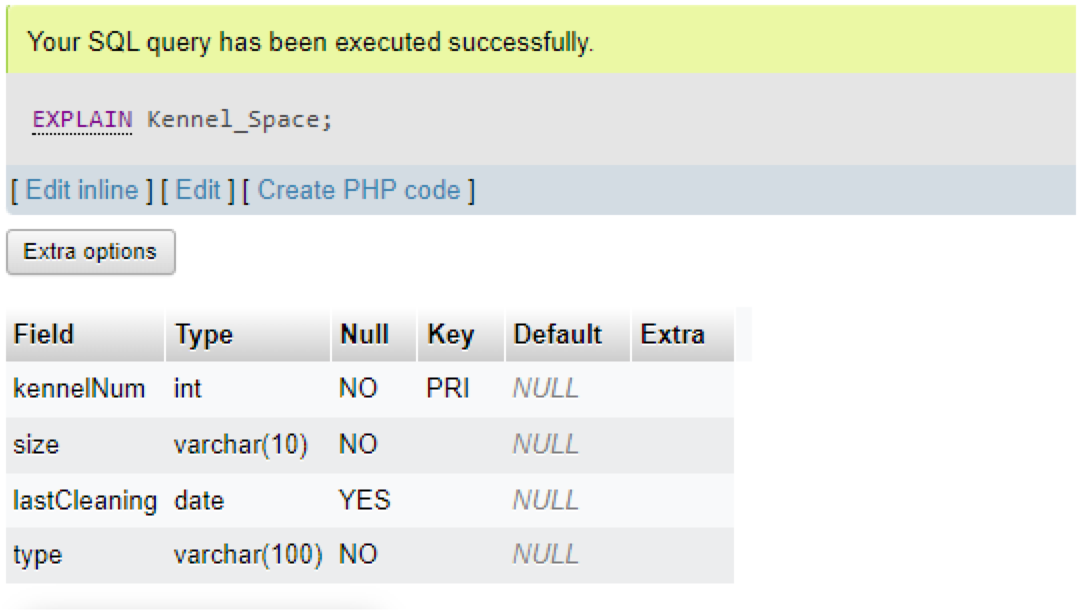

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Kennel_Space;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

COUNT(*)
40

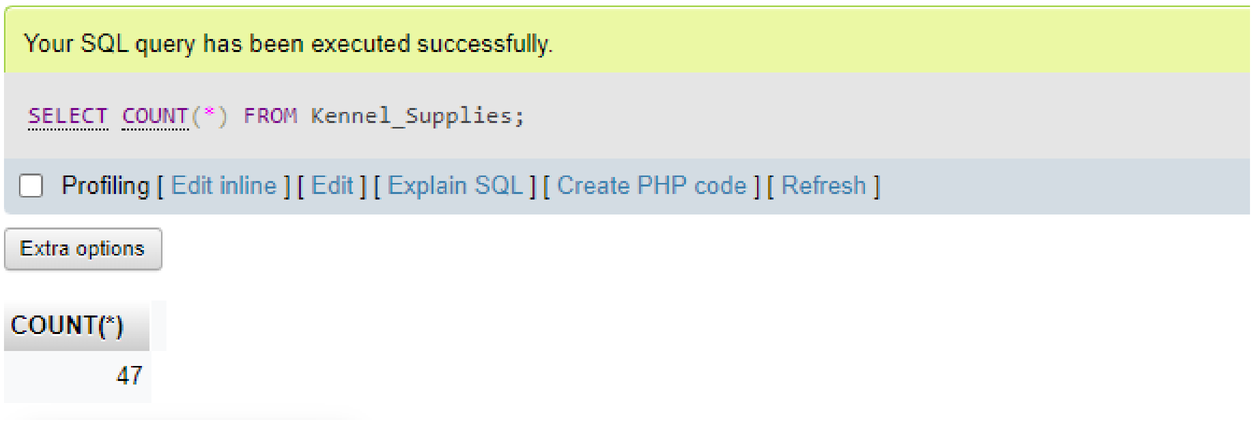

Your SQL query has been executed successfully.

```
EXPLAIN Kennel_Space;
```

[Edit inline] [Edit] [Create PHP code]

Extra options

Field	Type	Null	Key	Default	Extra
kennelNum	int	NO	PRI	*NULL*	
size	varchar(10)	NO		*NULL*	
lastCleaning	date	YES		*NULL*	
type	varchar(100)	NO		*NULL*	

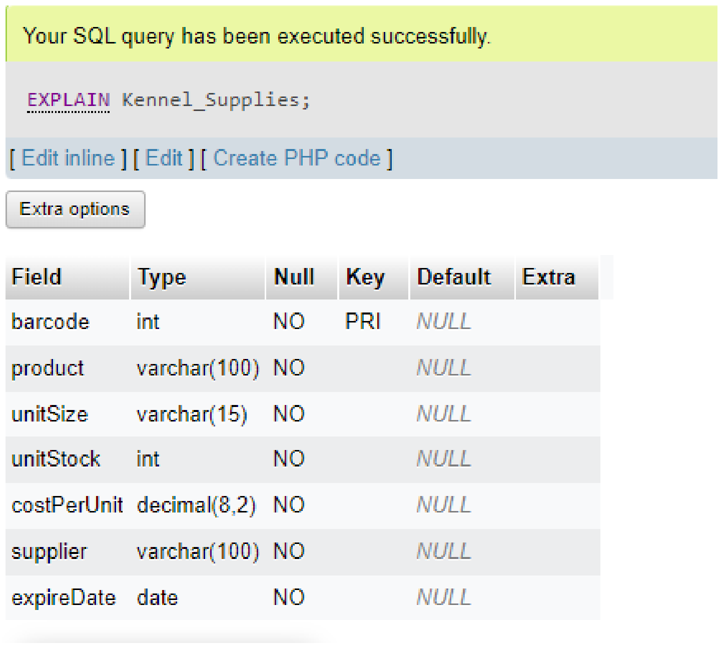

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Kennel_Supplies;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

COUNT(*)
47



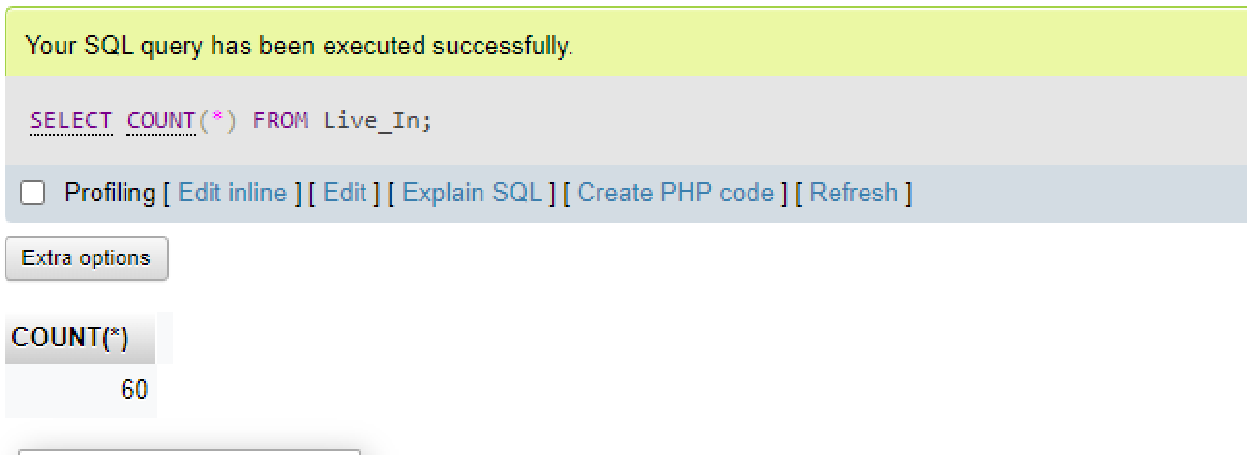
Your SQL query has been executed successfully.

```
EXPLAIN Kennel_Supplies;
```

[Edit inline] [Edit] [Create PHP code]

Extra options

Field	Type	Null	Key	Default	Extra
barcode	int	NO	PRI	*NULL*	
product	varchar(100)	NO		*NULL*	
unitSize	varchar(15)	NO		*NULL*	
unitStock	int	NO		*NULL*	
costPerUnit	decimal(8,2)	NO		*NULL*	
supplier	varchar(100)	NO		*NULL*	
expireDate	date	NO		*NULL*	



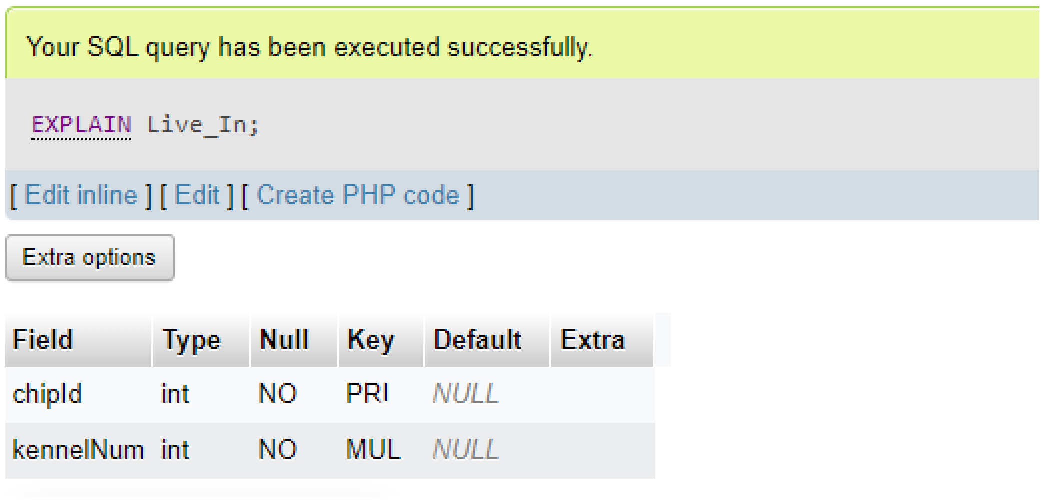
Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Live_In;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Extra options

COUNT(*)
60



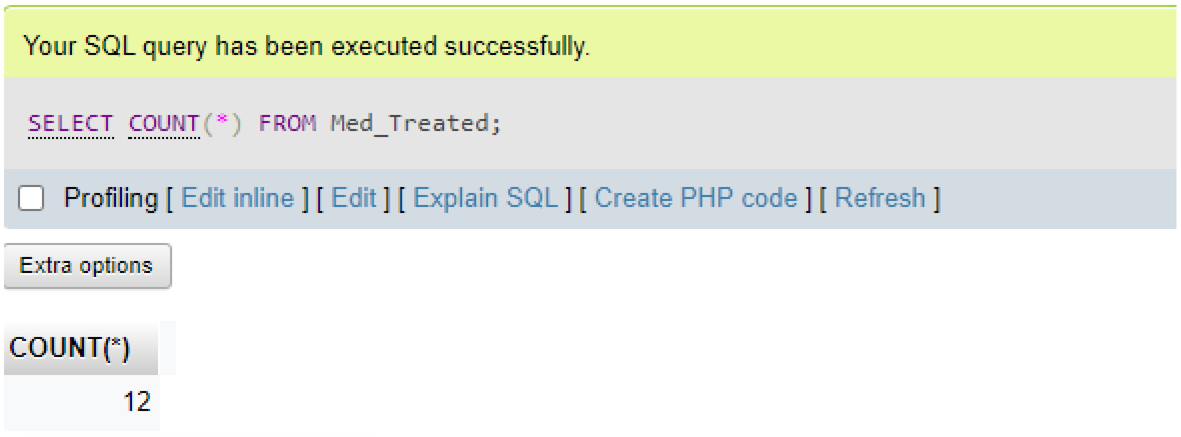
Your SQL query has been executed successfully.

```
EXPLAIN Live_In;
```

[Edit inline] [Edit] [Create PHP code]

Extra options

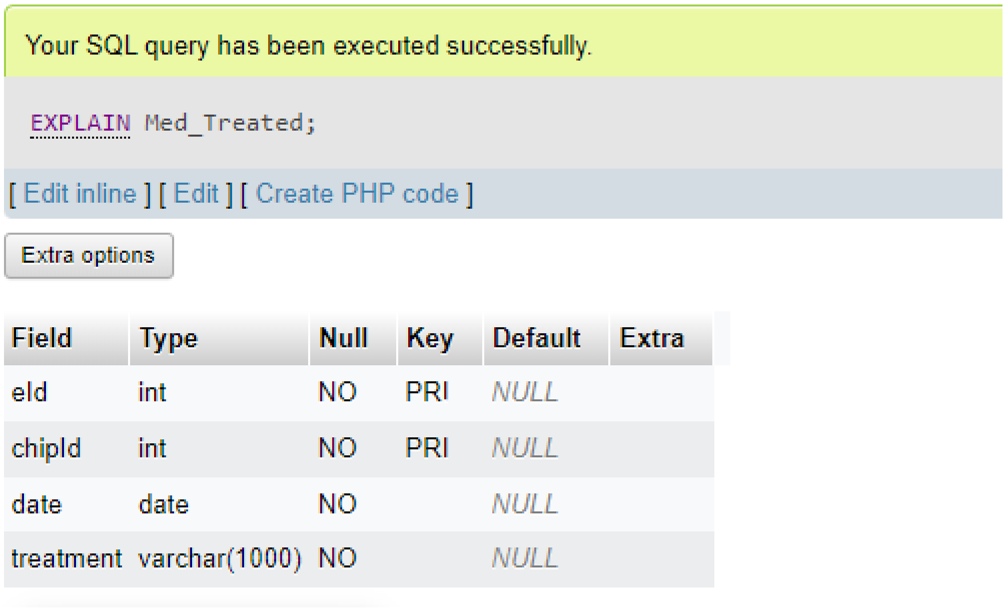
Field	Type	Null	Key	Default	Extra
chipId	int	NO	PRI	*NULL*	
kennelNum	int	NO	MUL	*NULL*	


Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Med_Treated;
```

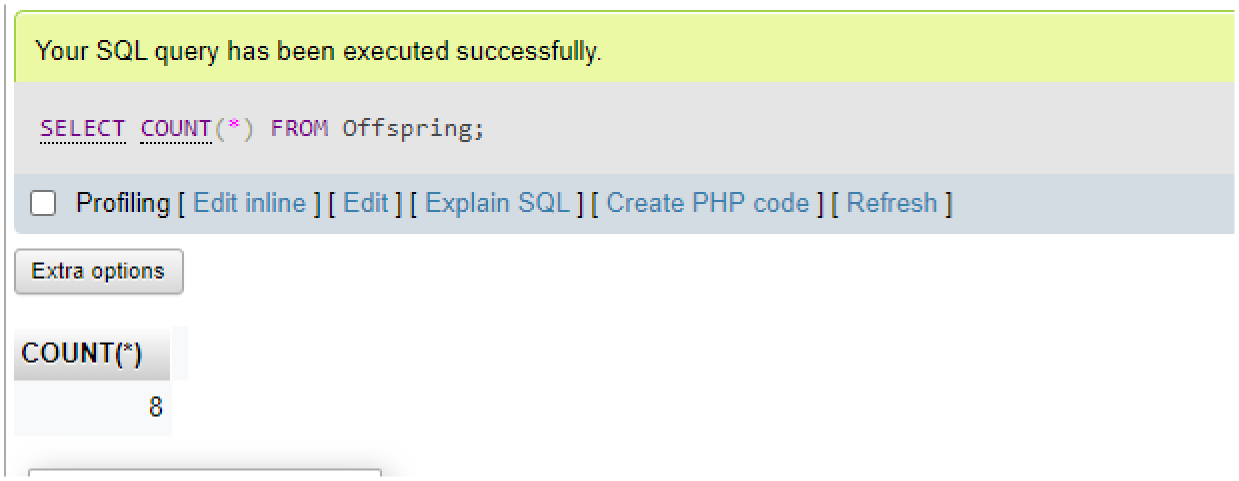
COUNT(*)
12



Your SQL query has been executed successfully.

```
EXPLAIN Med_Treated;
```

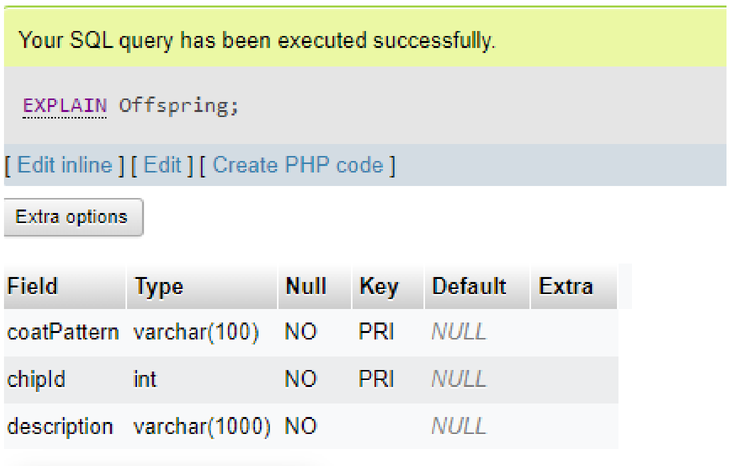
Field	Type	Null	Key	Default	Extra
eId	int	NO	PRI	*NULL*	
chipId	int	NO	PRI	*NULL*	
date	date	NO		*NULL*	
treatment	varchar(1000)	NO		*NULL*	


Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Offspring;
```

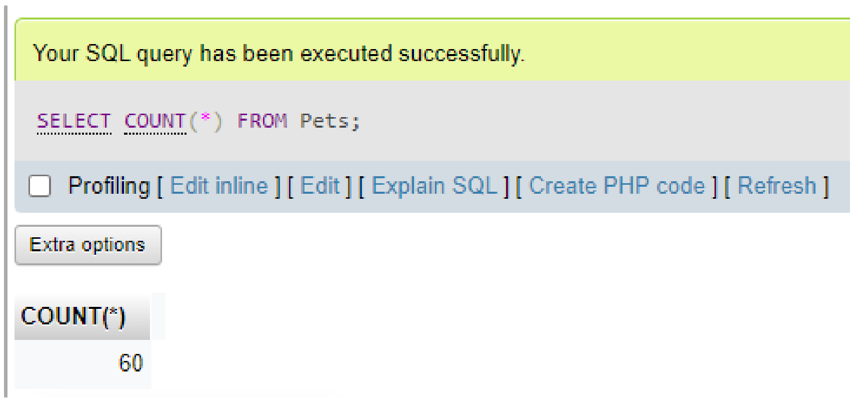
COUNT(*)
8



Your SQL query has been executed successfully.

```
EXPLAIN Offspring;
```

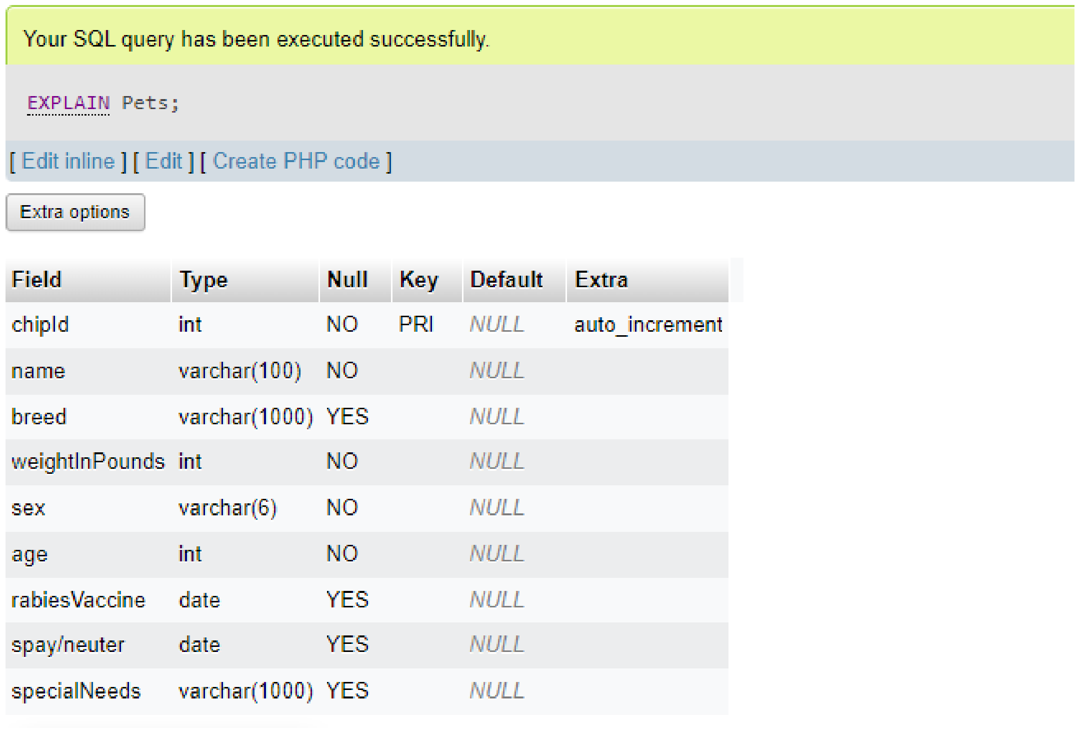
Field	Type	Null	Key	Default	Extra
coatPattern	varchar(100)	NO	PRI	*NULL*	
chipId	int	NO	PRI	*NULL*	
description	varchar(1000)	NO		*NULL*	


Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Extra options

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM Pets;
```

COUNT(*)
60



Your SQL query has been executed successfully.

```
EXPLAIN Pets;
```

Field	Type	Null	Key	Default	Extra
chipId	int	NO	PRI	*NULL*	auto_increment
name	varchar(100)	NO		*NULL*	
breed	varchar(1000)	YES		*NULL*	
weightInPounds	int	NO		*NULL*	
sex	varchar(6)	NO		*NULL*	
age	int	NO		*NULL*	
rabiesVaccine	date	YES		*NULL*	
spay/neuter	date	YES		*NULL*	
specialNeeds	varchar(1000)	YES		*NULL*	

Part 4: Examples of commands that were run to test the database

```
INSERT INTO `Employee` (`eId`, `firstName`, `lastName`, `address`, `title`, `email`, `salary`,
`phoneNumber`, `hiringDate`, `parkingSpot`) VALUES ('1', 'John', 'Doe', '13 First St. Turlock
CA', 'Senior Kennel Cleaner', 'johndoe@mail.com', '15000', '2095555555', '2022-11-25', '1');
```

I had accidentally made address an int at first, so it is correct now

```
INSERT INTO `Employee` (`eId`, `firstName`, `lastName`, `address`, `title`, `email`, `salary`,
`phoneNumber`, `hiringDate`, `parkingSpot`) VALUES (NULL, 'Jane', 'Dawson', '123 Second
St. Turlock CA', 'Senior Adoption Supervisor ', 'janedawson@mail.com', '25000', '2095551241',
'2022-11-25', '2');
```

This automatically inserted eId incremented from the last value, as expected

```
UPDATE `Employee` SET `salary` = '29000' WHERE `Employee`.`eId` = 2
```

Properly updated the salary where eId = 2

```
DELETE FROM Employee WHERE `Employee`.`eId` = 1"
```

```
INSERT INTO `Adopters` (`aId`, `firstName`, `lastName`, `address`, `phoneNumber`,
`approvalStatus`, `allergies`, `referenceEmail`) VALUES (NULL, 'Berry', 'Finch', '1 This St.
Town CA', '2095551254', 'approved', NULL, 'finch@mail.com');
```

Auto incremented the id to be 1

UPDATE `Adopters` SET `address` = '1 That St. Town CA' WHERE`Adopters`.`aId` = 1

DELETE FROM Adopters WHERE `Adopters`.`aId` = 1"

INSERT INTO `Kennel_Supplies` (`barcode`, `product`, `unitSize`, `unitStock`, `costPerUnit`, `supplier`, `expireDate`) VALUES ('1', 'dog chow', '16 oz', '4', '12', 'chow inc', '2023-11-24');

UPDATE `Kennel_Supplies` SET `unitSize` = '32 oz' WHERE`Kennel_Supplies`.`barcode` = 1

DELETE FROM Kennel_Supplies WHERE `Kennel_Supplies`.`barcode` = 1"

INSERT INTO `Kennel_Space` (`kennelNum`, `size`, `lastCleaning`, `type`) VALUES ('2', 'small', '2022-11-25', 'Wall locker');

UPDATE `Kennel_Space` SET `size` = 'single' WHERE`Kennel_Space`.`kennelNum` = 2

DELETE FROM Kennel_Space WHERE `Kennel_Space`.`kennelNum` = 2

INSERT INTO `Pets` (`chipId`, `name`, `breed`, `weightInPounds`, `sex`, `age`, `rabiesVaccine`, `spay/neuter`, `specialNeeds`) VALUES ('1', 'fido ', 'husky', '45', 'male', '3', '2022-11-25', '2022-11-25', NULL);

INSERT INTO `Dogs` (`chipId`, `DHLPP`) VALUES ('1', '2022-11-25');

```
DELETE FROM Pets WHERE `Pets`.`chipId` = 1
```

Cascade delete works as expected

```
INSERT INTO `Pets` (`chipId`, `name`, `breed`, `weightInPounds`, `sex`, `age`,
`rabiesVaccine`, `spay/neuter`, `specialNeeds`) VALUES ('2', 'Barb', 'Shorthair Cat', '8', 'female',
'9', '2022-11-25', NULL, NULL);
```

```
INSERT INTO `Cats` (`chipId`, `FVRCP`) VALUES ('2', '2018-11-08');
```

```
INSERT INTO `Offspring` (`coatPattern`, `chipId`, `description`) VALUES ('black with white
paws', '2', 'goofy looking little fellow');
```

```
UPDATE `Pets` SET `chipId` = '3' WHERE `Pets`.`chipId` = 2
```

Update cascaded as expected

```
DELETE FROM Pets WHERE `Pets`.`chipId` = 3
```

Delete cascaded properly

```
INSERT INTO `Adopters` (`aId`, `firstName`, `lastName`, `address`, `phoneNumber`,
`approvalStatus`, `allergies`, `referenceEmail`) VALUES (NULL, 'Paul', 'Bunyan', '12 This St.
Town CA', '2095559265', 'approved', NULL, 'bunyan@mail.com');
```

```
INSERT INTO `Employee` (`eId`, `firstName`, `lastName`, `address`, `title`, `email`, `salary`,
`phoneNumber`, `hiringDate`, `parkingSpot`) VALUES (NULL, 'Mary', 'Madrigal', '13 A St.
Town CA', 'Head of Adoption', 'mad@mail.com', '50000', '2095559847', '2014-11-18', '1');
```

```
INSERT INTO `Pets` (`chipId`, `name`, `breed`, `weightInPounds`, `sex`, `age`,
`rabiesVaccine`, `spay/neuter`, `specialNeeds`) VALUES (NULL, 'Jeremy', 'Shitzu ', '11', 'male',
'1', '2022-11-01', '2022-11-01', NULL);
```

```
INSERT INTO `Adopt` (`aId`, `chipId`, `adoptionCost`) VALUES ('3', '4', '96.00');
```

```
UPDATE `Adopt` SET `adoptionCost` = '99.00' WHERE `Adopt`.`chipId` = 4;
```

```
INSERT INTO `Kennel_Supplies` (`barcode`, `product`, `unitSize`, `unitStock`, `costPerUnit`,
`supplier`, `expireDate`) VALUES ('2', 'cat chow', '24 oz', '6', '24', 'cat chow inc', '2026-11-03');
```

```
INSERT INTO `Auth_to_Buy` (`eId`, `barcode`, `authorizedLimit`) VALUES ('3', '2', '200.00');
```

```
INSERT INTO `Employee` (`eId`, `firstName`, `lastName`, `address`, `title`, `email`, `salary`,
`phoneNumber`, `hiringDate`, `parkingSpot`) VALUES ('4', 'Tim', 'Burton', '12 Never St. City
CA', 'Head Veterenarian ', 'burton@mail.com', '95000', '2095559564', '2014-11-12', '9');
```

```
INSERT INTO `Med_Treated` (`eId`, `chipId`, `date`, `treatment`) VALUES ('4', '4', '2022-11-
25', 'Treated broken leg');
```

```
INSERT INTO `Behavior_Test` (`eId`, `chipId`, `date`, `assessment`) VALUES ('4', '4', '2022-
11-25', 'Not tolerant of other animals, loves kids ');

INSERT INTO `Kennel_Space` (`kennelNum`, `size`, `lastCleaning`, `type`) VALUES ('2',
'small', '2022-11-24', 'carry kennel');

INSERT INTO `Live_In` (`chipId`, `kennelNum`) VALUES ('4', '2');

DELETE FROM Employee WHERE `Employee`.`eId` = 4

DELETE FROM Adopters WHERE `Adopters`.`aId` = 3

DELETE FROM Pets WHERE `Pets`.`chipId` = 4

DELETE FROM Employee WHERE `Employee`.`eId` = 2

DELETE FROM Employee WHERE `Employee`.`eId` = 3

DELETE FROM Kennel_Space WHERE `Kennel_Space`.`kennelNum` = 2
```

5: Where we got our data

For the main portions of our data, we used the website mockaroo.com. Using this online tool, we created the fields for each table and gave the types of data we wanted it to automatically generate. There were some fields that could not be automatically generated because of their specific qualities, so for those we opted to leave the fields blank. After importing CSV files from the mock data generator, we then came back and manually filled in the fields that needed it, such as dog breeds. Update statements were used for some fields that were identical for multiple rows, and the rest were manually inserted one by one. Certain relations such as Adopting do not have many tuples because this would not be a very large table given our real-world structure. The number of pets were currently have would be limited by real world space and not every animal would be in the process of adoption.

6: Sample of tuples

```
  Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM ADOPT LIMIT 10;
```

```
aId     chipId  adoptionCost
6       1       93.43
13      2       89.01
17      4       98.21
18      5       85.80
34      11      89.72
```

```
35	12	85.36

37	17	86.89

40	18	93.84

43	19	86.61

47	22	85.10


   Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM Adopters LIMIT 10;
```

```
aId	firstName	lastName	addressphoneNumber approvalStatusallergies
	referenceEmail
1	Alvis	Flewin 49041 Merry Hill	4736204577	Approved
	aflewin0@yolasite.com
2	Odetta	Pillinger	80980 Drewry Crossing	6325174835	Approved
	opillinger1@symantec.com
3	Maude Jacobovitch	7968 Prentice Place	5968781644	Approved
	mjacobovitch2@aol.com
4	Junette Perassi 92 Dahle Circle	8602944742	Approved	Allergic to cats
	jperassi3@vk.com
5	Alyson Yukhtin	82 Atwood Park	5677322735	Failed
	ayukhtin4@acquirethisname.com
```

```
6       Welch  Denslow        0 Knutson Pass          3322188986    Approved
        wdenslow5@comcast.net
7       Sholom          Boustred        82402 Talmadge Plaza          6671067630    Approved
        sboustred6@163.com
8       Leland Simcock        558 Brentwood Place  3712572473    Approved
        lsimcock7@foxnews.com
9       Carolyn         Gensavage       36432 Forster Parkway         9447361154    Approved
        cgensavage8@yale.edu
10      Consalve        Edden  700 Burrows Trail      3849820136    Approved
        cedden9@umich.edu

  Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM `Auth_to_Buy` LIMIT 10;
```

```
eId     barcode         authorizedLimit
11      1       1000.00
11      2       1000.00
11      3       1000.00
11      4       1000.00
11      5       1000.00
11      6       1000.00
11      7       1000.00
```

```
11      8       1000.00

11      9       1000.00

11      10      1000.00


   Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)


SELECT * FROM `Behavior_Test` LIMIT 10;


eId     chipId  date      assessment

18      1       2021-08-04    positive, no concern

18      2       2021-06-18    positive, no concern

18      3       2021-09-19    positive, no concern

18      4       2021-08-13    positive, no concern

18      5       2021-07-10    not compatible with other house pets

18      6       2021-11-22    positive, no concern

18      7       2021-07-17    positive, no concern

18      8       2021-06-14    positive, no concern

18      9       2021-07-02    positive, no concern

18      10      2021-06-08    positive, no concern


   Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)


SELECT * FROM `Case_Managing` LIMIT 10;
```

eId	aId	caseId
4	6	22
3	13	10
3	17	21
3	18	12
3	21	15
3	34	11
4	35	17
1	37	2
1	40	4
4	43	18

Showing rows 0 - 9 (10 total, Query took 0.0002 seconds.)

```
SELECT * FROM `Cats` LIMIT 10;
```

chipId	FVRCP
21	2021-09-20
22	2021-09-05
23	2021-05-24
24	2022-02-03
25	2022-02-11

```
26       2022-02-28

27       2022-10-04

28       2021-09-21

29       2022-05-06

30       2021-11-27


   Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)


SELECT * FROM `Dogs` LIMIT 10;


chipId  DHLPP

1        2022-06-23

2        2022-07-13

3        2022-07-13

4        2022-07-05

5        2022-07-11

6        2022-07-12

7        2021-10-13

8        2022-04-19

9        2022-03-09

10       2022-03-08


   Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM `Employee` LIMIT 10;

eId     firstName       lastName        addresstitle    email   salary  phoneNumber hiringDate
        parkingSpot
1       Rudiger         Ortner  12914 Ridgeview Way     Adoption Supervisor
        rortner0@liveinternet.ru        18186   8644879169      2021-07-15      565
2       Issiah  Aleksich        648 Russell Parkway     Head Vet
        ialeksich1@surveymonkey.com     102934          7762571931      2020-08-04      716
3       Madelon         De Courtney     75886 New Castle Plaza          Adoption Supervisor
        mdecourtney2@pbs.org            188284          1908461546      2021-08-19      305
4       Marty   Mottram         1 Starling Crossing     Adoption Supervisor
        mmottram3@redcross.org  194469          8065641486      2019-12-26      61
5       Jyoti   Garrand         08 Luster Alley         Accounting      jgarrand4@unicef.org 130678
        1902650017      2019-10-15      370
6       Minda   Klimkowski      458 Prairieview Circle          Head of Care
        mklimkowski5@gov.uk             77880   8893324090      2017-11-21      81
7       Alfonse         Sermin  3 Banding Hill  Human Resources Manager
        asermin6@myspace.com            142463          2089254324      2020-02-05      406
8       Neils   Fallen  531 Nelson Lane         Vet     nfallen7@wiley.com      86011   9365243973
        2020-12-31      246
9       Henka   Wallbridge      40065 Starling Court    Accountant II   hwallbridge8@mit.edu
        105501          9933941380      2018-10-11      970
```

```
10      Vickie  Marmyon     72 Vernon Pass      Vet     vmarmyon9@icio.us   24650
        2069300034  2021-02-10  143
```

```
  Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM `Kennel_Space` LIMIT 10;
```

```
kennelNum   size    lastCleaning    type
1       S       2022-02-23  carry
2       S       2022-09-01  carry
3       S       2022-05-14  carry
4       S       2022-09-23  carry
5       S       2021-12-07  carry
6       S       2021-12-28  carry
7       S       2022-04-28  carry
8       S       2022-04-09  carry
9       S       2022-07-30  carry
10      S       2022-10-17  carry
```

```
  Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM `Kennel_Supplies` LIMIT 10;
```

```
barcode         product         unitSize        unitStock       costPerUnit     supplier
        expireDate
1       dry dog chow    10lbs   35      25.00   dog chow inc    2023-10-14
2       dry cat chow    8lbs    6       24.00   cat chow inc    2026-11-03
3       wet dog food    40 cans         67      123.81 dog chow inc     2024-02-03
4       wet cat food    45 cans         72      136.98 cat chow inc     2022-12-27
5       dog tags        1       32      113.32 dog supply inc 2023-06-11
6       water bowl      1       18      131.00 dog supply inc 2023-06-13
7       food bowl       1       45      38.87   dog supply inc 2022-12-10
8       auto feeder     1       31      141.32 dog supply inc 2023-06-17
9       dog collar      1       58      54.29   dog supply inc 2023-06-22
10      dog treats      20      56      101.54 dog supply inc 2023-03-28

  Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM `Live_In` LIMIT 10;
```

```
chipId  kennelNum
20      1
26      2
36      2
37      2
28      3
```

```
32      4

33      4

25      6

30      7

38      7


  Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM `Med_Treated` LIMIT 10;
```

```
eId     chipId  date       treatment

2       1       2022-07-05  Tail removed after injury

2       6       2022-08-17  Given stitches for leg wound

2       18      2022-07-13  Treated for broken leg

8       1       2022-09-22  Given cone for 2 weeks to treat rash

8       6       2022-08-10  Treated for bite wound

8       9       2022-07-12  Stitched leg from fence wound

8       16      2022-06-14  Treated for ear infection

8       35      2022-08-16  Treated for bite wound

10      9       2022-08-09  Treated for fleas

10      15      2022-07-20  Treated for bite wound


  Showing rows 0 -  7 (8 total, Query took 0.0001 seconds.)
```

```
SELECT * FROM `Offspring` LIMIT 10;
```

```
coatPattern      chipId  description

Black with white bow tie        28      Six toes on front paws

Black with white paws           28      Cannot sit still, attacks its siblings

Black with white spot between eyes   28      One really floppy ear, the other sticks straight u...

Brown with Black paws           3       Whines until he is held

Brown with black spot on left side   3       Does NOT like to be held

Brown with black tipped ears 3         Looks angry, but that's just because of the eyebro...

Solid Black     28      Sleeps all day and pounces all night

Solid Brown     3       Active and curious


  Showing rows 0 -  9 (10 total, Query took 0.0002 seconds.)
```

```
SELECT * FROM `Pets` LIMIT 10;
```

```
chipId          name    breed   weightInPounds          sex     age     rabiesVaccine spay/neuter

        specialNeeds

1       Johnny          Boxer           70      M       12      2020-09-19      2022-05-25

2       Luella          Corgis          38      F       9       2020-04-24      2022-06-24

3       Aeriell         Share-Pei       47      F       8       2021-06-07      2022-09-14
```

4	Tarrah	King Charles Spaniel	67	F	13	2020-05-27	2021-09-04
	Missing Front Left Leg						
5	Deva	Shar-Pei	46	F	9	2022-07-13	2021-09-02
6	Vernice	Dalmation	70	F	3	2022-07-25	2022-07-25
7	Kally	Terrier	23	F	14	2021-10-13	2022-06-08
8	Bendick	Doberman	70	M	12	2020-08-23	2022-01-01
9	Margarita	Foxhound	63	F	1	2018-08-23	2021-10-30
10	Vinni	Schnauzer	72	F	15	2022-02-06	2021-12-10

SQL Queries on Database (Part 6)

1:

```
CREATE TABLE Employee
        ( eId INTEGER(20),
        firstname VARCHAR(100),
        lastname VARCHAR (100),
        address VARCHAR (50),
        title VARCHAR (100),
        email VARCHAR (30),
        salary INTEGER,
        phoneNumber BIGINT (10),
        hiringDate DATE,
        parkingSpot INTEGER,
        PRIMARY KEY (eId));

CREATE TABLE Adopters
        ( aId INTEGER (20),
        firstname VARCHAR(100),
        lastname VARCHAR (100),
        address VARCHAR (50),
        phoneNumber BIGINT (10),
        approvalStatus VARCHAR (10),
        allergies VARCHAR (50),
        referenceEmail VARCHAR (50),
        primary key (aId));

CREATE TABLE Kennel_Supplies
        (barcode INTEGER (20),
        product VARCHAR (30),
        unitSize VARCHAR (15),
        unitStock INTEGER,
        costPerUnit INTEGER,
        supplier VARCHAR (30),
        expireDate DATE,
        primary key (barcode));
```

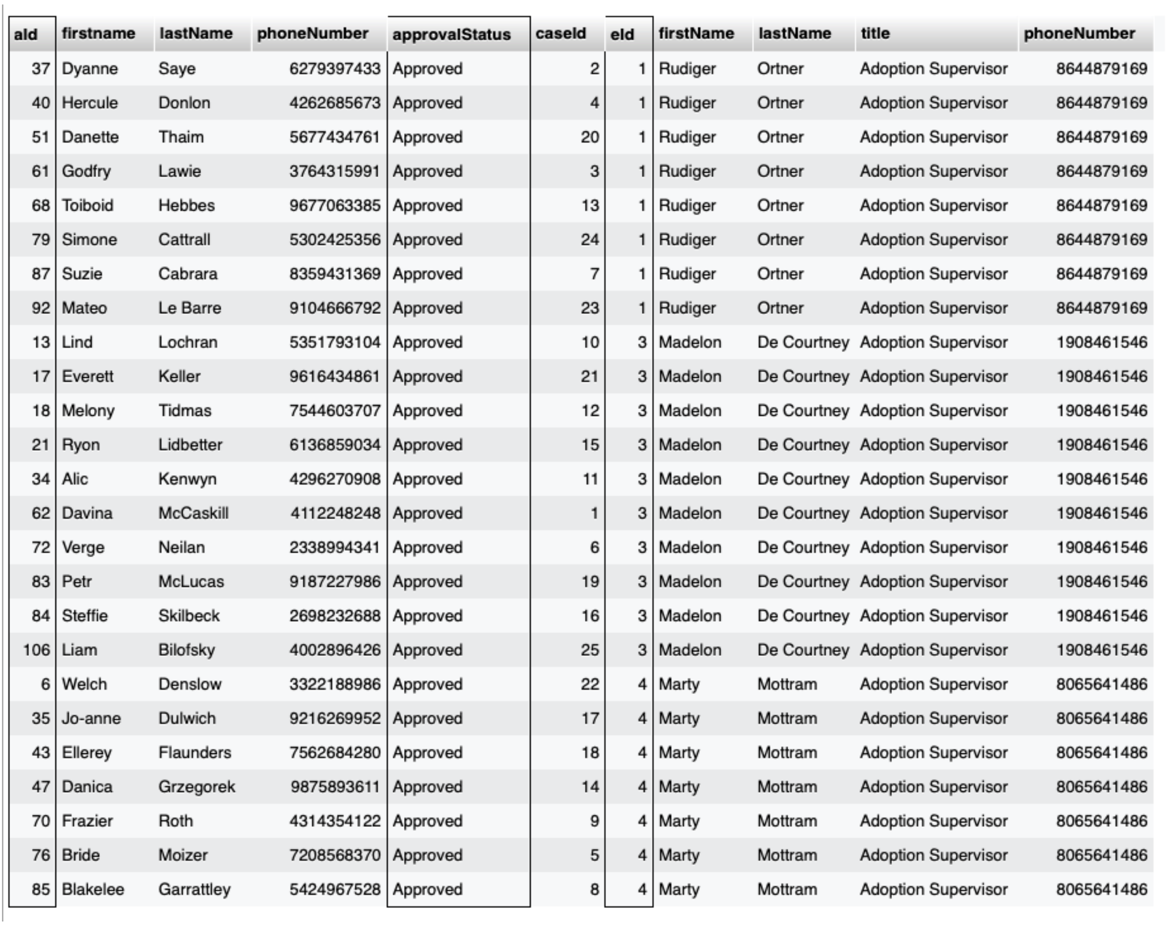
```
CREATE TABLE Kennel_Space
        (kennelNum INTEGER (4),
        size VARCHAR (10),
        lastCleaning DATE,
        type VARCHAR (100),
        primary key (kennelNum));

CREATE TABLE Pets
        ( chipId INTEGER (20),
        name VARCHAR (100),
        breed VARCHAR (1000),
        weight INTEGER,
        sex VARCHAR (6).
        age INTEGER,
        rabiesVaccine DATE,
        spay/neuter DATE,
        specialNeeds VARCHAR (1000),
        primary key (chipId));

CREATE TABLE Offspring
        (coatPattern VARCHAR (100),
        description VARCHAR (1000),
        chipId INTEGER (20) NOT NULL,
        primary key (coatPattern, chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE dogs
        ( chipID INTEGER (20).
        DHLPP DATE,
        primary key (chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE cats
        ( chipID INTEGER (20).
        FVRCP DATE,
        primary key (chipId),
        foreign key (chipId) references Pets
                on delete cascade);

CREATE TABLE Case_managing
        (eId INTEGER(20),
        aId VARCHAR (20),
        caseId INTEGER (20
        primary key (aId),
        foreign key (eId) references Employee,
        foreign key (aId) references Adopters);

CREATE TABLE Adopt
        ( aId INTEGER (20),
        chipId INTEGER (20),
        adoptionCost DECIMAL (M, 2),
        primary key (chipId),
        foreign key (aId) references Adopters,
        foreign key (chipId) references Pets);

CREATE TABLE Auth_to_buy
        ( eId INTEGER(20),
        barcode VARCHAR(20),
        authorizedLimit DECIMAL (M, 2),
        primary key (eId, barcode),
        foreign key (eId) references Employee,
        foreign key (barcode) references Kennel_Supplies);

CREATE TABLE Med_treated
        ( eId INTEGER (20),
        chipId INTEGER (20),
        date DATE,
        treatment VARCHAR (1000),
        primary key (eId, chipId),
        foreign key (eId) references Employee,
        foreign key (chipId) references Pets);

CREATE TABLE Behavior_test
        (eId INTEGER(20),
        chipId INTEGER (20),
        date DATE,
        assessment VARCHAR (1000),
        primary key (chipId),
        foreign key (eId) references Employee,
        foreign key (chipId) references Pets);

CREATE TABLE Live_in
        ( chipId INTEGER (20),
        kennelNum INTEGER,
        primary key (chipId),
        foreign key (kennelNum) references Kennel_Space,
        foreign key (chipId) references Pets);
```

2:

Query 1:

```
SELECT A.aId, A.firstname, A.lastName, A.phoneNumber, A.approvalStatus, CM.caseId,
E.eId, E.firstName, E.lastName, E.title, E.phoneNumber
FROM Adopters A, Case_Managing CM, Employee E
WHERE E.eId = CM.eId AND A.aId = CM.aId AND A.approvalStatus = 'Approved'
ORDER BY E.eId;
```

This query finds the basic information and phone number for each employee and adopter currently involved in the process of adopting a pet. This could be used to quickly find the proper contact information for each case. The phone number of the adopter might be needed by the case manager, and the phone number of the case manager could be passed on to the adopter if the need arose.

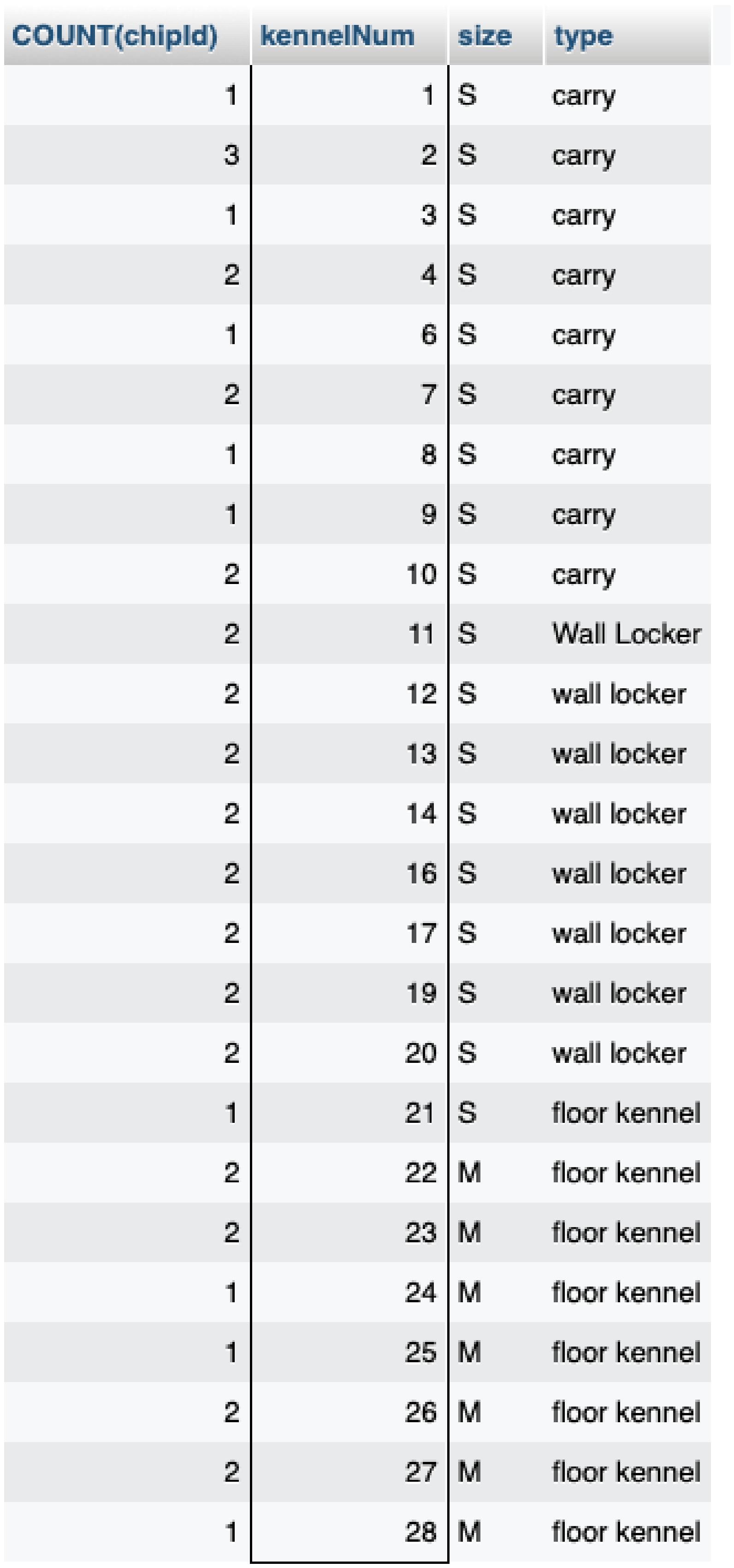


aId	firstname	lastName	phoneNumber	approvalStatus	caseId	eId	firstName	lastName	title	phoneNumber
37	Dyanne	Saye	6279397433	Approved	2	1	Rudiger	Ortner	Adoption Supervisor	8644879169
40	Hercule	Donlon	4262685673	Approved	4	1	Rudiger	Ortner	Adoption Supervisor	8644879169
51	Danette	Thaim	5677434761	Approved	20	1	Rudiger	Ortner	Adoption Supervisor	8644879169
61	Godfry	Lawie	3764315991	Approved	3	1	Rudiger	Ortner	Adoption Supervisor	8644879169
68	Toiboid	Hebbes	9677063385	Approved	13	1	Rudiger	Ortner	Adoption Supervisor	8644879169
79	Simone	Cattrall	5302425356	Approved	24	1	Rudiger	Ortner	Adoption Supervisor	8644879169
87	Suzie	Cabrara	8359431369	Approved	7	1	Rudiger	Ortner	Adoption Supervisor	8644879169
92	Mateo	Le Barre	9104666792	Approved	23	1	Rudiger	Ortner	Adoption Supervisor	8644879169
13	Lind	Lochran	5351793104	Approved	10	3	Madelon	De Courtney	Adoption Supervisor	1908461546
17	Everett	Keller	9616434861	Approved	21	3	Madelon	De Courtney	Adoption Supervisor	1908461546
18	Melony	Tidmas	7544603707	Approved	12	3	Madelon	De Courtney	Adoption Supervisor	1908461546
21	Ryon	Lidbetter	6136859034	Approved	15	3	Madelon	De Courtney	Adoption Supervisor	1908461546
34	Alic	Kenwyn	4296270908	Approved	11	3	Madelon	De Courtney	Adoption Supervisor	1908461546
62	Davina	McCaskill	4112248248	Approved	1	3	Madelon	De Courtney	Adoption Supervisor	1908461546
72	Verge	Neilan	2338994341	Approved	6	3	Madelon	De Courtney	Adoption Supervisor	1908461546
83	Petr	McLucas	9187227986	Approved	19	3	Madelon	De Courtney	Adoption Supervisor	1908461546
84	Steffie	Skilbeck	2698232688	Approved	16	3	Madelon	De Courtney	Adoption Supervisor	1908461546
106	Liam	Bilofsky	4002896426	Approved	25	3	Madelon	De Courtney	Adoption Supervisor	1908461546
6	Welch	Denslow	3322188986	Approved	22	4	Marty	Mottram	Adoption Supervisor	8065641486
35	Jo-anne	Dulwich	9216269952	Approved	17	4	Marty	Mottram	Adoption Supervisor	8065641486
43	Ellerey	Flaunders	7562684280	Approved	18	4	Marty	Mottram	Adoption Supervisor	8065641486
47	Danica	Grzegorek	9875893611	Approved	14	4	Marty	Mottram	Adoption Supervisor	8065641486
70	Frazier	Roth	4314354122	Approved	9	4	Marty	Mottram	Adoption Supervisor	8065641486
76	Bride	Moizer	7208568370	Approved	5	4	Marty	Mottram	Adoption Supervisor	8065641486
85	Blakelee	Garrattley	5424967528	Approved	8	4	Marty	Mottram	Adoption Supervisor	8065641486

Query 2:

```
SELECT COUNT(chipId), LI.kennelNum, KS.size, KS.type
FROM Live_In LI, Kennel_Space KS
WHERE LI.kennelNum = KS.kennelNum
GROUP BY LI.kennelNum;
```

This query gives a quick look at how many pets are already living in a particular kennel. Some animals do better when they are not alone, but crowding can become an issue. This would allow the workers to quickly determine how full a kennel already is, along with how much room might be available.

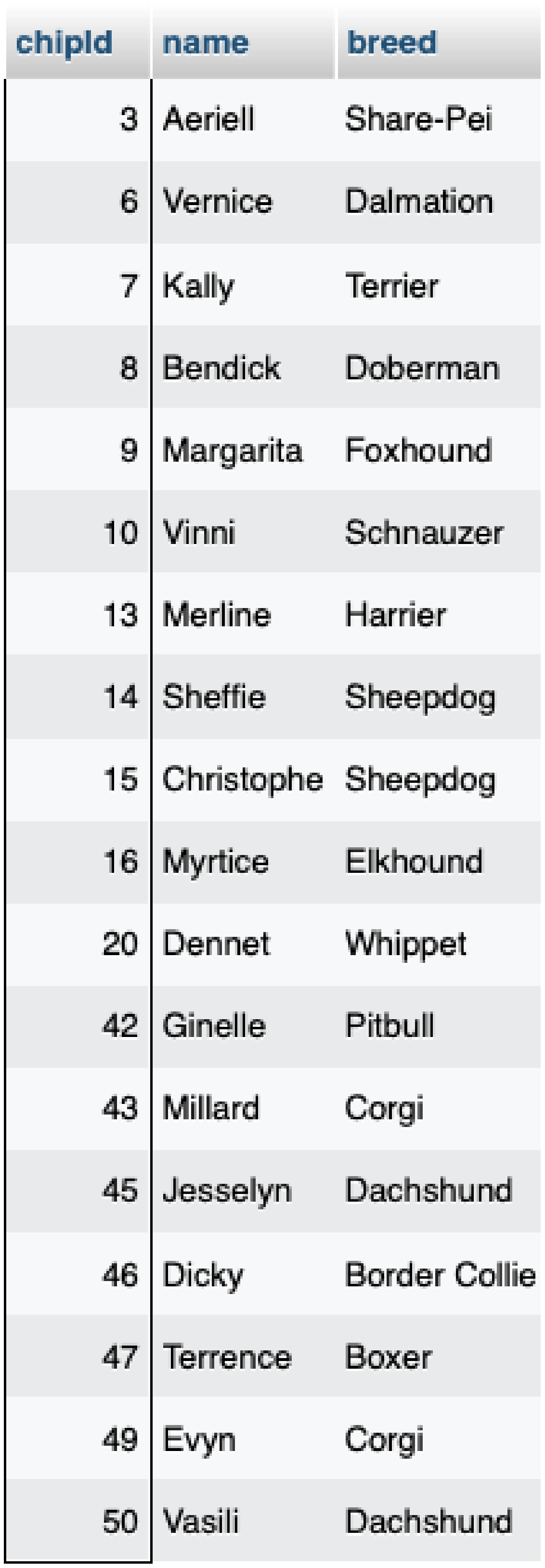


COUNT(chipId)	kennelNum	size	type
1	1	S	carry
3	2	S	carry
1	3	S	carry
2	4	S	carry
1	6	S	carry
2	7	S	carry
1	8	S	carry
1	9	S	carry
2	10	S	carry
2	11	S	Wall Locker
2	12	S	wall locker
2	13	S	wall locker
2	14	S	wall locker
2	16	S	wall locker
2	17	S	wall locker
2	19	S	wall locker
2	20	S	wall locker
1	21	S	floor kennel
2	22	M	floor kennel
2	23	M	floor kennel
1	24	M	floor kennel
1	25	M	floor kennel
2	26	M	floor kennel
2	27	M	floor kennel
1	28	M	floor kennel

Query 3:

```
SELECT P.chipId, P.name, P.breed

FROM Pets P

WHERE P.chipId

NOT IN(

        SELECT P.chipId

        FROM Pets P, Cats C

        WHERE P.chipId = C.chipId

)

AND P.chipId

NOT IN(

        SELECT A.chipId

        FROM Adopt A

);
```

This query finds dogs that are not currently being adopted. This would be important for someone with a cat allergy, or maybe even just someone who specifically wants a dog. There are two subqueries to find which animals are cats and which are currently being adopted. The query then only shows results for pets not in the two subqueries. This leaves a list of dogs that are not currently being adopted.



chipId	name	breed
3	Aeriell	Share-Pei
6	Vernice	Dalmation
7	Kally	Terrier
8	Bendick	Doberman
9	Margarita	Foxhound
10	Vinni	Schnauzer
13	Merline	Harrier
14	Sheffie	Sheepdog
15	Christophe	Sheepdog
16	Myrtice	Elkhound
20	Dennet	Whippet
42	Ginelle	Pitbull
43	Millard	Corgi
45	Jesselyn	Dachshund
46	Dicky	Border Collie
47	Terrence	Boxer
49	Evyn	Corgi
50	Vasili	Dachshund

Query 4:

SELECT ALL Med_Treated.eId, Employee.lastName, treatment, Med_Treated.chipId,

Pets.name, Pets.breed, date

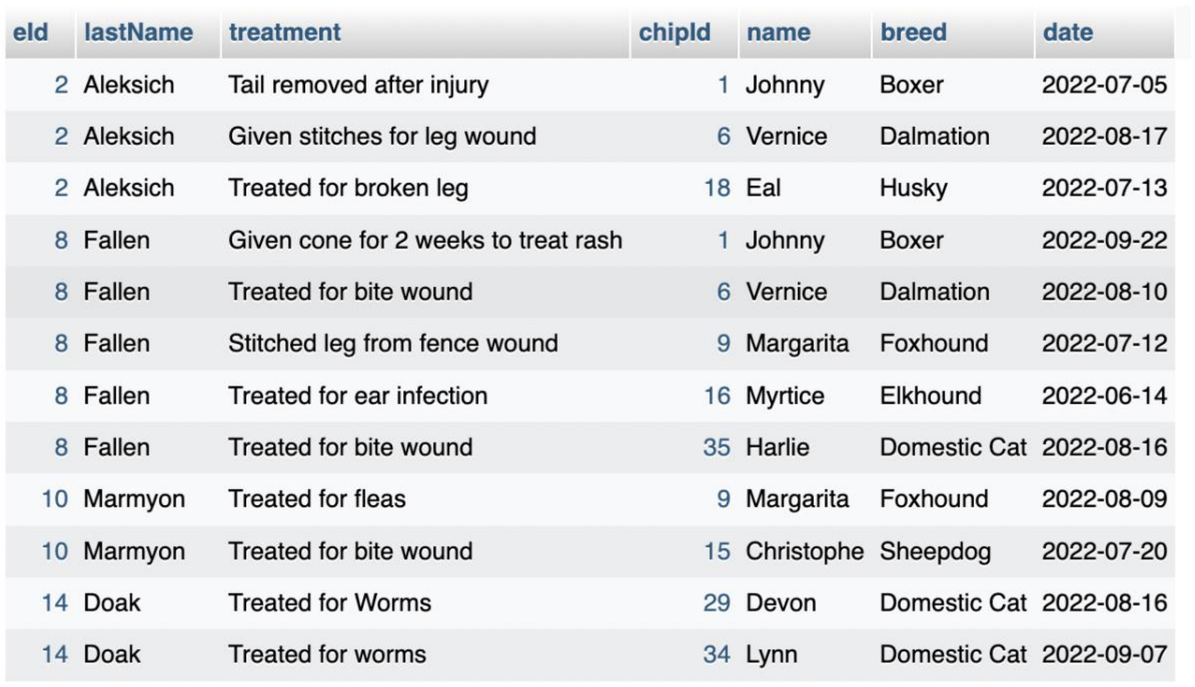
FROM Med_Treated

JOIN Pets

ON Med_Treated.chipId=Pets.chipId

JOIN Employee

ON Med_Treated.eId=Employee.eId



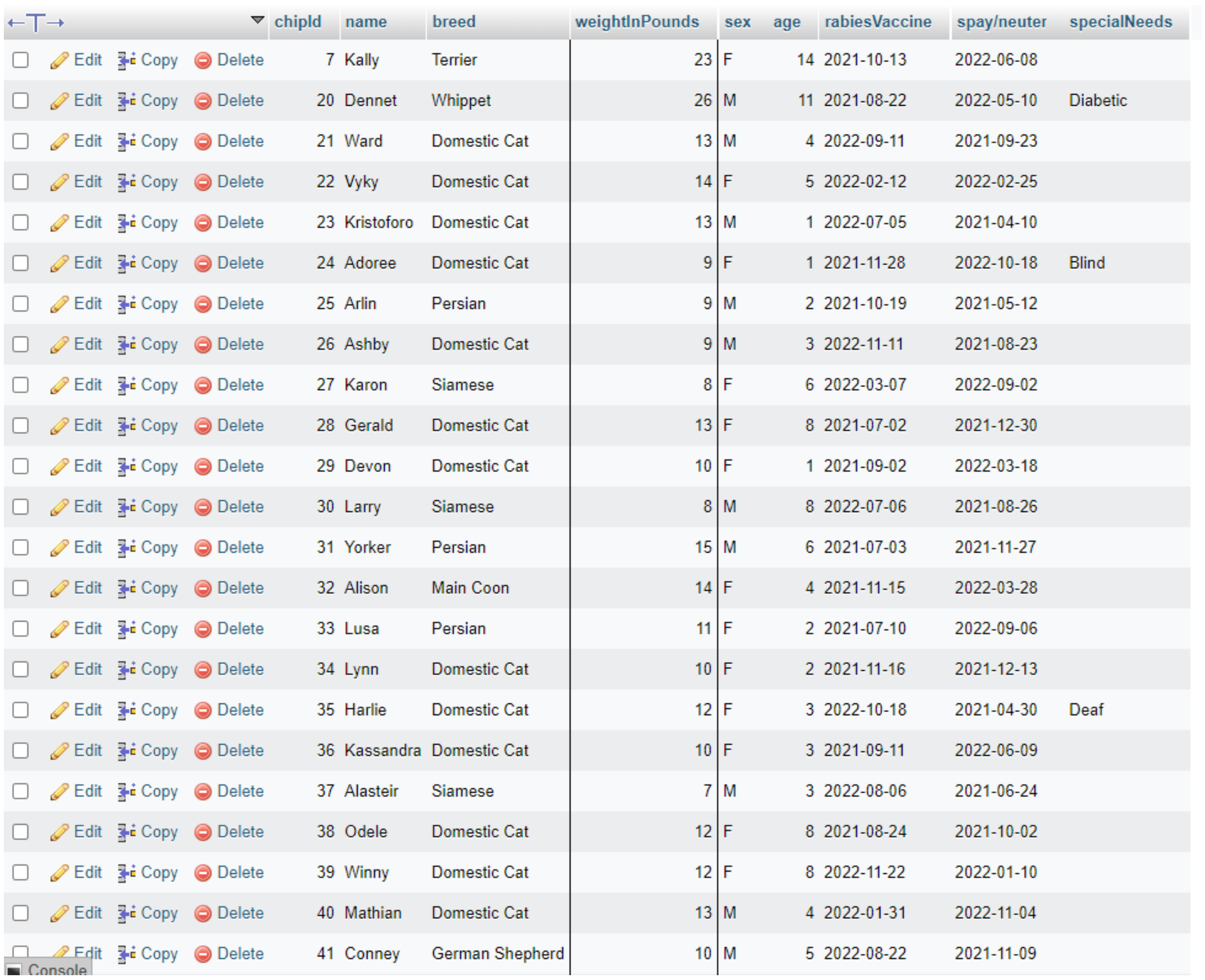
eId	lastName	treatment	chipId	name	breed	date
2	Aleksich	Tail removed after injury	1	Johnny	Boxer	2022-07-05
2	Aleksich	Given stitches for leg wound	6	Vernice	Dalmation	2022-08-17
2	Aleksich	Treated for broken leg	18	Eal	Husky	2022-07-13
8	Fallen	Given cone for 2 weeks to treat rash	1	Johnny	Boxer	2022-09-22
8	Fallen	Treated for bite wound	6	Vernice	Dalmation	2022-08-10
8	Fallen	Stitched leg from fence wound	9	Margarita	Foxhound	2022-07-12
8	Fallen	Treated for ear infection	16	Myrtice	Elkhound	2022-06-14
8	Fallen	Treated for bite wound	35	Harlie	Domestic Cat	2022-08-16
10	Marmyon	Treated for fleas	9	Margarita	Foxhound	2022-08-09
10	Marmyon	Treated for bite wound	15	Christophe	Sheepdog	2022-07-20
14	Doak	Treated for Worms	29	Devon	Domestic Cat	2022-08-16
14	Doak	Treated for worms	34	Lynn	Domestic Cat	2022-09-07

This query displays the description of any medical treatment that was done on any of the animals that needed assistance (this includes the animal chip Id, name, and breed) along with the date the treatment was done by a specific employee at the shelter. All shelters are required to keep record of the vet treatments/medications due and given. This query also displays the employee Id and their last name. Knowing which employee issued the treatment would allow the shelter manager to hold anyone accountable who may have mis-administered a treatment.

<u>Query 5:</u>

SELECT * FROM Pets

WHERE weightInPounds < (SELECT AVG(weightInPounds) FROM PETS);



chipId	name	breed	weightInPounds	sex	age	rabiesVaccine	spay/neuter	specialNeeds
7	Kally	Terrier	23	F	14	2021-10-13	2022-06-08	
20	Dennet	Whippet	26	M	11	2021-08-22	2022-05-10	Diabetic
21	Ward	Domestic Cat	13	M	4	2022-09-11	2021-09-23	
22	Vyky	Domestic Cat	14	F	5	2022-02-12	2022-02-25	
23	Kristoforo	Domestic Cat	13	M	1	2022-07-05	2021-04-10	
24	Adoree	Domestic Cat	9	F	1	2021-11-28	2022-10-18	Blind
25	Arlin	Persian	9	M	2	2021-10-19	2021-05-12	
26	Ashby	Domestic Cat	9	M	3	2022-11-11	2021-08-23	
27	Karon	Siamese	8	F	6	2022-03-07	2022-09-02	
28	Gerald	Domestic Cat	13	F	8	2021-07-02	2021-12-30	
29	Devon	Domestic Cat	10	F	1	2021-09-02	2022-03-18	
30	Larry	Siamese	8	M	8	2022-07-06	2021-08-26	
31	Yorker	Persian	15	M	6	2021-07-03	2021-11-27	
32	Alison	Main Coon	14	F	4	2021-11-15	2022-03-28	
33	Lusa	Persian	11	F	2	2021-07-10	2022-09-06	
34	Lynn	Domestic Cat	10	F	2	2021-11-16	2021-12-13	
35	Harlie	Domestic Cat	12	F	3	2022-10-18	2021-04-30	Deaf
36	Kassandra	Domestic Cat	10	F	3	2021-09-11	2022-06-09	
37	Alasteir	Siamese	7	M	3	2022-08-06	2021-06-24	
38	Odele	Domestic Cat	12	F	8	2021-08-24	2021-10-02	
39	Winny	Domestic Cat	12	F	8	2022-11-22	2022-01-10	
40	Mathian	Domestic Cat	13	M	4	2022-01-31	2022-11-04	
41	Conney	German Shepherd	10	M	5	2022-08-22	2021-11-09	

This query shows how adopters who have living restrictions, such as living in an apartment complex that only allows animals under a specific weight. This then selects the below average weight compared to the other animals in the shelter by using AVG. Finally, showing all the pet’s information allows the adopter to view more features about animals that meet their requirements. This can provide an example of the size of their potential animal.

<u>Failed attempt:</u>

This was my first attempt at creating query 3. I realized it was flawed because there is no need to retrieve data from the adopters table. I was thinking that we needed to check for who was allergic to cats, but in the real-world scenario, this query would only be made if we already were assuming that we wanted to find available dogs. There was also no way to join these two tables, which is what really tipped me off to the fact that this query was nonsensical.

```
SELECT A.aId, a.firstName, A.lastName, A.phoneNumber, P.chipId, P.name, P.breed
FROM Pets P, Adopters A
WHERE A.allergies='Allergic to cats'
AND
P.chipId
NOT IN(
        SELECT P.chipId
        FROM Pets P, Cats C
        WHERE P.chipId = C.chipId
)
AND P.chipId
NOT IN(
        SELECT A.chipId
        FROM Adopt A);
```

Contributions:

3. User ability/missing elements:

Somebody, such as an animal shelter manager, who is working in the domain of our application would be able to use our web-enabled database. Our interface contains most elements that an actual animal shelter database would have. After viewing a few real world animal shelter websites, the only aspect that most other applications included that ours didn't was pictures for each animal in the shelter. However, that wasn't very ideal for our project since we used mock data. Other than that, there are no other missing elements.

<u>URL</u>: https://hopper.csustan.edu/~xtoepfer/adoption.html

4. Safety Checks

Before the code makes the call to the database with the query, it uses the stripos function to check the contents of the query. The results of these checks are stored in variables. Then these variables are used within a set of if statements to determine the response from the webpage. If the common 1=1 or a=a SQL injection statements are in the query, the query is not processed and the user is given a warning. This does not prevent all SQL injection, but it is a deterrent for some of the most common attempts.

The stripos calls are made on lines 9-13 and the if/else statements are on lines 23, 26, 29, 33, and 67.

5. Sample SQL Query

Update pets set name='Johnny' where chipId=1;

Select chipId, name from Pets where chipId=1;

These queries was tested ad hoc and worked.

7. Extra Functionality

Along with the checks for the SQL injection statement, there are also checks for the words update, insert, select, and delete. If update or insert are in the query, the query is run and a warning is presented to the user that tells them to be mindful of their actions, since they are being allowed to modify data. If the word delete is in the query, the query is not processed and the user is notified that they should not delete data. If the keyword select is in the query, the query is run and the data returned and displayed in a clear lines chart. In any other case, the user is alerted that there seems to be something wrong, and nothing is run. There is also a button on this page to return to the original homepage to make overall navigation easier.

We will also need to include copies of the source code once we have it finalized and know we are not making any more changes.