FINAL PROJECT

Poultry Planner

AUTHOR: TARA MASSEY
INTRODUCTION TO DATABASES (CS 340)
SUMMER 2015

LIVE WEBSITE:
web.engr.oregonstate.edu/~masseyta/DBfinal/main.php

Table of Contents

I. Outline

My final project centers on planning a flock of backyard chickens. This endeavor has become increasingly popular in recent years, providing a source of eggs, meat, or simply new additions to the household in the form of a pet. The overall success of the flock depends on the owner's choices, however, and there can be great variation in flock temperament, care needs, and egg production based on the breeds the user selects. Luckily, raising backyard chickens is a relatively simple task provided the owner has done the appropriate research into which chicken breeds are suitable for their needs, their environment, or their currently existing flock.

My website, particularly the page View or Edit Your Flock, is aimed at allowing users to design their flock online prior to purchasing birds. This is an interesting prospect due to the vast varieties of chickens, unique care considerations, and ranges in egg production. The website educates prospective owners on various breeds they're considering adding to their flock, and provides information on how many eggs the owner can expect weekly with the current flock they have designed. This allows the user to plan a custom flock that provides them with the desired amount of eggs, and informs the user about special needs or considerations they may need to take into account.

II. Database Outline in Words

The Poultry Planner database consists of six entities. They are designed as follows:

1. Egg: Egg is written as 'eggTable' when accessing the table. This entity has all information about various combinations of eggs a chicken could lay. Egg has three attributes:
    a. eID : The egg identification key. This is an auto incrementing integer that represents the listed color and quantity combination inserted into the table. The egg identification key is the primary key.
    b. eggType: The color of the egg the chicken produces. This is a varchar(255), and may be entered as any string of user choice. The type of egg a chicken produces is required, and cannot be null.
    c. quantity: The amount of eggs the chicken produces per week. This is an integer value only. All chickens produce eggs, and the quantity cannot be null.

2. Purpose: Purpose is written a 'purposeTable' when accessing the table. This entity has all information about the purpose the chicken was bred for. Purpose has two attributes:
    a. pID: The purpose identification key. This is an auto incrementing integer that represents the purpose the chicken was bred for. The purpose identification key is the primary key.
    b. purpose: The primary purpose the chicken was bred for. This is a varchar(255) and may be entered as any string of user choice. All chickens were bred for a purpose, and it cannot be null.

3. Temperament: Temperament is written as ‘temperTable’ when accessing the table. This entity has all information about the primarily associated temperament of the chicken breed. Temperament has two attributes:
    a. tID: The temperament identification key. This is auto incrementing integer that represents a temperament that a chicken may have. The temperament identification key is the primary key.
    b. trait: Trait is the main personality trait that will provide a chicken with a temperament. This is a varchar(255) and may be entered as any string of user choice. Trait cannot be null.

4. Needs: Needs is written as ‘needsTable’ when accessing the table. This entity has all information about the various special care considerations a chicken breed. Needs has a Many-to-Many relationship with the breeds table, which will be discussed shortly. Needs has two attributes:
    a. nID: The need identification key. This is auto incrementing integer that represents a need or care concern. The need identification key is the primary key.
    b. need: Need is a special care consideration. A chicken breed many have many care needs, and a care need can be associated with many different chicken breeds. When adding to the needsTable only, the need cannot be null.

5. Flock: Flock is written as ‘flock’ when accessing the table. This entity contains the chicken breeds the user selects to join their flock. Flock has two attributes:
    a. fID: A chicken’s identification key. This auto incrementing integer represents a single chicken entered into the flock. This fID identification key is the primary key.
    b. id: The id is an integer that cannot be null. It is a foreign key reference to bID (breed’s identification key). Since a prospective owner can have many chickens of the same breed, the id is not unique.

6. Breed: Breed is written as ‘breedsTable’ when accessing the table. This entity contains all information that is required to be in a chicken breed. Breed has five attributes:
    a. bID: The breed’s identification key. This auto incrementing integer represents a single chicken breed. The bID identification key is the primary key.
    b. name: The name of the chicken breed. This name is a varchar(255), cannot be null, and must be unique.
    c. eID: An integer that is a foreign key reference to the egg entity (eggTable). The eID cannot be null.
    d. pID: An integer that is a foreign key reference to the purpose entity (purposeTable). The pID cannot be null.
    e. tID: An integer that is a foreign key reference to the temperament entity (temperTable). The tID cannot be null.

Furthermore, the Poultry Planner implements many different relationships between the entities:

Breeds may have many special needs – This is a Many-to-Many relationship. One breed may many different needs, and one need may be frequently seen among many different breeds.

In order to implement this relationship I created a joining table. This table is called the 'need_breed' table. It consists of the primary identification key from the breed table (bID), and primary identification key from the need table (nID).

Each breed has a temperament – This is a Many-to-One relationship when viewed as breed to temperament. Each breed may have only one defining temperament trait (tID), and that temperament is not exclusive to that breed.

Each temperament can be found in many breeds – When one looks at the reverse of the above listed relationship, we see a One-to-Many relationship when viewed from temperament to breed. Each temperament trait can be found among many different breeds.

Each breed was bred for a purpose – This is a Many-to-One relationship when viewed as breed to purpose. Each breed has one primary purpose that they were bred for (pID), and that purpose is not unique to that breed.

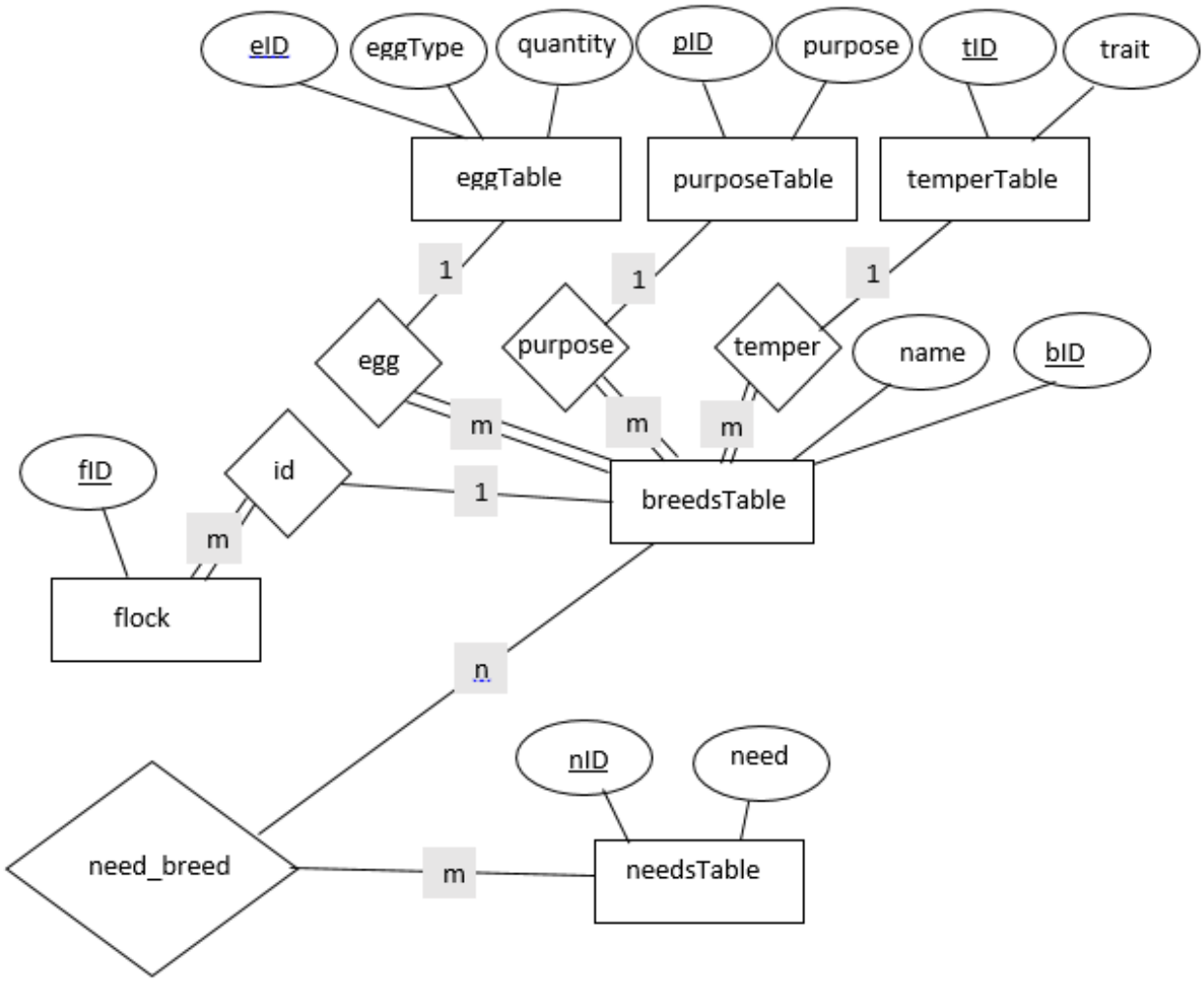
Each purpose was the start of many different breeds – When one looks at the reverse of the above listed relationship, we see a One-to-Many relationship when viewed from purpose to breed. A single purpose spawned the production of many different breeds. There are many breeds that fall under each purpose category.

Each breed has an egg Type – This is a Many-to-One relationship when viewed as breed to egg type. Each breed can only lay one combination of egg color and average quantity (eID).

Each egg type can be produced by many different breeds – When one looks at the reverse of the above listed relationship, we see a One-to-Many relationship when viewed from egg type to breed. Though a breed can only lay one egg type, that type is not unique to that breed. Many breeds, for example, may produce brown eggs at an average of five per week.

Chickens are a specific breed – Each chicken in the flock can only be one breed of chicken (bID).

III. ER Diagram of the Database

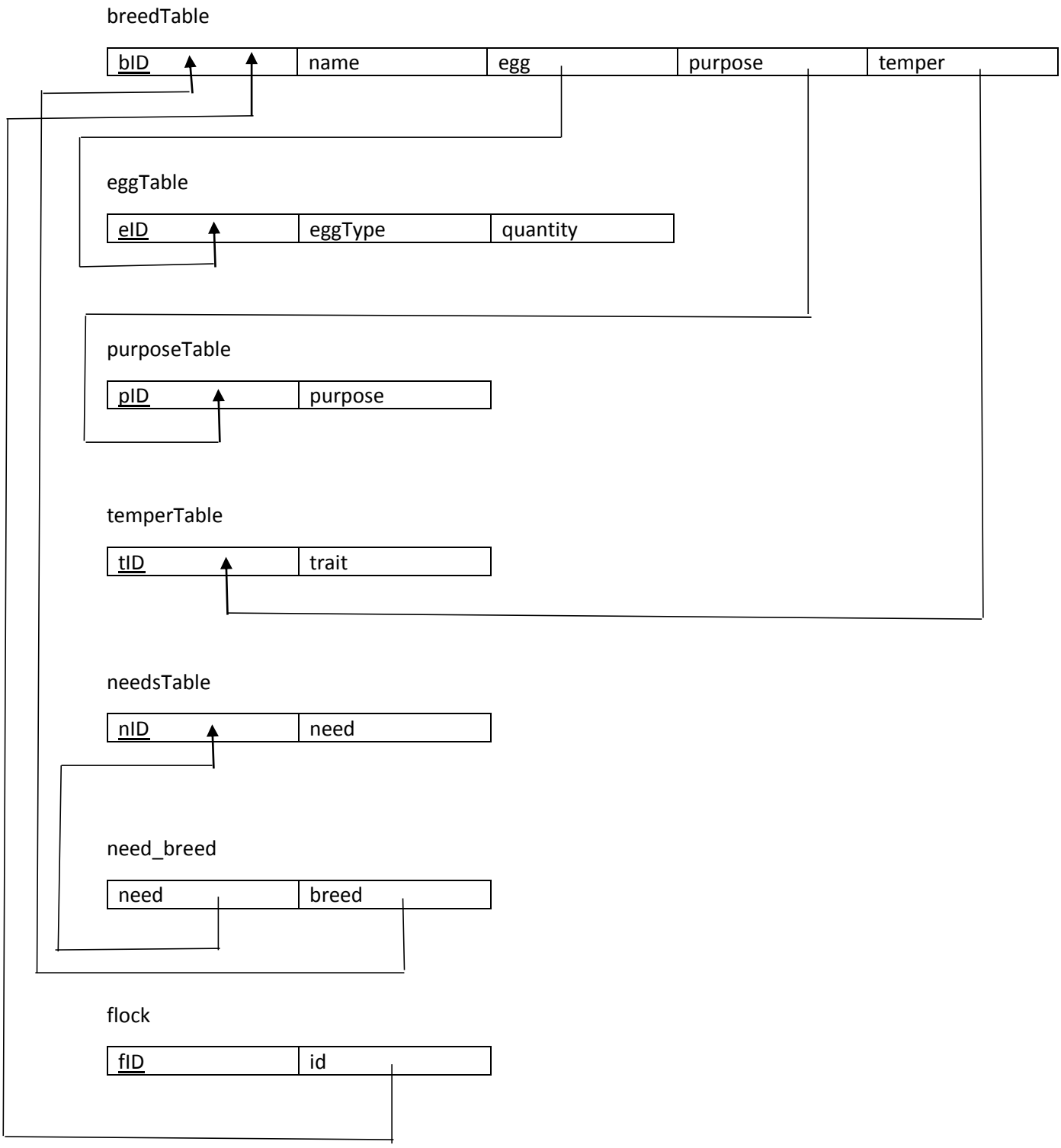


- **breedsTable is related to exactly one eggTable**
  - Every breed has exactly one egg
  - Total participation
- **eggTable is related to zero or more breedsTable**
  - Every egg can be produced by zero listed birds, or many listed birds
  - Ideally, this would be total participation, but since the user can add to the eggTable, per assignment instructions, and later choose not to attribute that type of egg to any birds, it is not total participation.
- **breedsTable is related to exactly one purposeTable**
  - Every breed has exactly one purpose
  - Total particpation
- **purposeTable is related zero or more breedsTable**
  - Every purpose can be had by zero listed birds, or many listed birds

    - Ideally, this would be total participation, but since the user can add to the purposeTable, per assignment instructions, and later choose not to attribute that type of purpose to any birds, it is not total participation.

- **breedsTable is related to exactly one temperTable**
    - Every breed has exactly one temperament
    - Total participation
- **temperTable is related to zero or more breedsTable**
    - Every temperament can be had by zero listed birds, or many listed birds
    - Ideally, this would be total participation, but since the user can add to the temperTable, per assignment instructions, and later choose not to attribute that type of temperament to any birds, it is not total participation.
- **flock is related to exactly one breedTable**
    - A chicken in the flock can have only one breed
    - Total participation
- **breedsTable is related to zero or more flock**
    - A breed can be found no chickens, or many chickens, of the flock
- **breedsTable is related to zero or more needsTable**
    - A breed can have no special needs, or many special needs
    - Not required to have special care needs, not total participation
- **needsTable is related to zero or more breedsTable**
    - A need can be found no birds, or many birds
    - Ideally, this would be total participation, but since the user can add to the needsTable, per assignment instructions, and later choose not to attribute that type of special care need to any birds, it is not total participation.

IV. Database Schema



V. Queries

Section One: Table Creation Queries

Entity Table Creation

```
CREATE TABLE eggTable (
eID int AUTO_INCREMENT NOT NULL,
eggType varchar(255) NOT NULL,
quantity int NOT NULL,
PRIMARY KEY(eID)
)ENGINE = InnoDB;

CREATE TABLE purposeTable (
pID int AUTO_INCREMENT NOT NULL,
purpose varchar(255) NOT NULL,
PRIMARY KEY(pID),
UNIQUE KEY (purpose)
)ENGINE = InnoDB;

CREATE TABLE temperTable(
tID int AUTO_INCREMENT NOT NULL,
trait varchar(255) NOT NULL,
PRIMARY KEY(tID),
UNIQUE KEY (trait)
)ENGINE = InnoDB;

CREATE TABLE needsTable (
nID int AUTO_INCREMENT NOT NULL,
need varchar(255) NOT NULL,
```

```
PRIMARY KEY(nID),
UNIQUE KEY (need)
)ENGINE = InnoDB;

CREATE TABLE breedsTable (
bID int AUTO_INCREMENT NOT NULL,
name varchar(255) NOT NULL,
egg int NOT NULL,
purpose int NOT NULL,
temper int NOT NULL,
FOREIGN KEY(egg) REFERENCES eggTable(eID),
FOREIGN KEY(purpose) REFERENCES purposeTable(pID),
FOREIGN KEY(temper) REFERENCES temperTable(tID),
PRIMARY KEY(bID),
UNIQUE KEY (name)
)ENGINE = InnoDB;

CREATE TABLE flock(
fID INT NOT NULL AUTO_INCREMENT,
id INT NOT NULL,
PRIMARY KEY (fID),
FOREIGN KEY ( id ) REFERENCES breedsTable( bID )
) ENGINE = INNODB;
```

Many-to-Many Relationship Table Creation

```
CREATE TABLE need_breed(
breed INT NOT NULL ,
need INT NOT NULL ,
FOREIGN KEY ( breed ) REFERENCES breedsTable( bID ) ,
```

```
FOREIGN KEY ( need ) REFERENCES needsTable( nID )
) ENGINE = INNODB;
```

Initial Starting Data: Table Insertion Code

eggTable General Code:

```
INSERT INTO eggTable (eggType, quantity) VALUES ([color], [quantity]);
```

eggTable Code Used to Insert Starting Data:

```
INSERT INTO eggTable (eggType, quantity) VALUES ("White", 2);
INSERT INTO eggTable (eggType, quantity) VALUES ("White", 3);
INSERT INTO eggTable (eggType, quantity) VALUES ("White", 5);
INSERT INTO eggTable (eggType, quantity) VALUES ("brown", 2);
INSERT INTO eggTable (eggType, quantity) VALUES ("brown", 3);
INSERT INTO eggTable (eggType, quantity) VALUES ("brown", 5);
INSERT INTO eggTable (eggType, quantity) VALUES ("green", 2);
INSERT INTO eggTable (eggType, quantity) VALUES ("green", 3);
INSERT INTO eggTable (eggType, quantity) VALUES ("green", 5);
```

purposeTable General Code:

```
INSERT INTO purposeTable (purpose) VALUES ([purpose]);
```

purposeTable Code Used to Insert Starting Data:

```
INSERT INTO purposeTable (purpose) VALUES ("egg production");
INSERT INTO purposeTable (purpose) VALUES ("meat production");
INSERT INTO purposeTable (purpose) VALUES ("all purpose");
INSERT INTO purposeTable (purpose) VALUES ("show");
```

temperTable General Code:

INSERT INTO temperTable (trait) VALUES ([trait]);

temperTable Code Used to Insert Starting Data:

INSERT INTO temperTable (trait) VALUES ("calm");

INSERT INTO temperTable (trait) VALUES ("active");

INSERT INTO temperTable (trait) VALUES ("aggressive");

INSERT INTO temperTable (trait) VALUES ("flighty");

needsTable General Code:

INSERT INTO needsTable (need) VALUES ([need]);

needsTable Code Used to Insert Starting Data:

INSERT INTO needsTable (need) VALUES ("no cold weather");

INSERT INTO needsTable (need) VALUES ("no hot weather");

INSERT INTO needsTable (need) VALUES ("no rainy weather");

INSERT INTO needsTable (need) VALUES ("poor eyesight accomodations");

INSERT INTO needsTable (need) VALUES ("poor flight ability accomodations");

INSERT INTO needsTable (need) VALUES ("feathered foot care required");

INSERT INTO needsTable (need) VALUES ("needs flock of same breed");

breedsTable General Code:

INSERT INTO breedsTable(name, egg, purpose, temper)

VALUES ([breedName]

(SELECT eID FROM eggTable WHERE eggType= [color] AND quantity= [quantityAmount]),

(SELECT pID FROM purposeTable WHERE purpose= [purpose]),

(SELECT tID FROM temperTable WHERE trait= [trait]));

breedsTable Code Used to Insert Starting Data:

```
INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Araucana",
(SELECT eID FROM eggTable WHERE eggType= "green" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Australorp",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "active"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Brahma",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 3),
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Cochin",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "show"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Cornish",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "meat production"),
```

```
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Dominique",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "active"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Easter Egger",
(SELECT eID FROM eggTable WHERE eggType= "green" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Houdan",
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "show"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Jersey Giant",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 3),
(SELECT pID FROM purposeTable WHERE purpose= "all purpose"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Lakenvelder",
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 3),
```

```
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "flighty"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Leghorn",
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "active"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Modern Game",
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "meat production"),
(SELECT tID FROM temperTable WHERE trait= "active"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Orpington",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "all purpose"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Plymouth Rock",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "all purpose"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Polish",
```

```
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "egg production"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Russian Orloff",
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "meat production"),
(SELECT tID FROM temperTable WHERE trait= "aggressive"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Sebright",
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "show"),
(SELECT tID FROM temperTable WHERE trait= "flighty"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Shamo",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "meat production"),
(SELECT tID FROM temperTable WHERE trait= "aggressive"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Silkie",
(SELECT eID FROM eggTable WHERE eggType= "white" AND quantity= 2),
(SELECT pID FROM purposeTable WHERE purpose= "show"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
```

```
VALUES ("Sussex",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "all purpose"),
(SELECT tID FROM temperTable WHERE trait= "calm"));

INSERT INTO breedsTable(name, egg, purpose, temper)
VALUES ("Wyandotte",
(SELECT eID FROM eggTable WHERE eggType= "brown" AND quantity= 5),
(SELECT pID FROM purposeTable WHERE purpose= "all purpose"),
(SELECT tID FROM temperTable WHERE trait= "calm"));
```

need_breed General Code:

```
INSERT INTO need_breed(breed, need)
VALUES (
(SELECT bID FROM breedsTable WHERE name= [breedName]),
(SELECT nID FROM needsTable WHERE need= [need]));
```

need_breed Code Used to Insert Starting Data:

```
INSERT INTO need_breed(breed, need)
VALUES (
(SELECT bID FROM breedsTable WHERE name= "Brahma"),
(SELECT nID FROM needsTable WHERE need= "feathered foot care required"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Brahma"),
(SELECT nID FROM needsTable WHERE need= "no rainy weather"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Cochin"),
```

```
(SELECT nID FROM needsTable WHERE need= "no rainy weather"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Cornish"),
(SELECT nID FROM needsTable WHERE need= "no hot weather"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Cornish"),
(SELECT nID FROM needsTable WHERE need= "no cold weather"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Houdan"),
(SELECT nID FROM needsTable WHERE need= "poor eyesight accomodations"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Houdan"),
(SELECT nID FROM needsTable WHERE need= "needs flock of same breed"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Lakenvelder"),
(SELECT nID FROM needsTable WHERE need= "no cold weather"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Polish"),
(SELECT nID FROM needsTable WHERE need= "poor eyesight accomodations"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Silkie"),
(SELECT nID FROM needsTable WHERE need= "poor eyesight accomodations"));
```

```
INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Silkie"),
(SELECT nID FROM needsTable WHERE need= "needs flock of same breed"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Silkie"),
(SELECT nID FROM needsTable WHERE need= "feathered foot care required"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Silkie"),
(SELECT nID FROM needsTable WHERE need= "poor flight ability accomodations"));

INSERT INTO need_breed(breed, need)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Sussex"),
(SELECT nID FROM needsTable WHERE need= "no cold weather"));
```

flock General Code:

```
INSERT INTO flock(id)
VALUES ((SELECT bID FROM breedsTable WHERE name= [breedName]));
```

flock Code Used to Insert Starting Data:

```
INSERT INTO flock(id)
VALUES ((SELECT bID FROM breedsTable WHERE name= "Silkie"));
```

Section Two: General Use Queries

Get all Chicken Breed Information : Search by Breed Name (Full Query)

```
SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose,
temperTable.trait, needsTable.need FROM breedsTable
    INNER JOIN eggTable on breedsTable.egg = eggTable.eID
    INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID
    INNER JOIN temperTable on breedsTable.temper = temperTable.tID
    INNER JOIN need_breed on breedsTable.bID = need_breed.breed
    INNER JOIN needsTable on needsTable.nID = need_breed.need
    WHERE breedsTable.name = [searchCriteria];
```

Get all Chicken Breed Information : Search by Breed Name

Get all information about breed except needs:

```
SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose,
temperTable.trait FROM breedsTable
    INNER JOIN eggTable on breedsTable.egg = eggTable.eID
    INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID
    INNER JOIN temperTable on breedsTable.temper = temperTable.tID
    WHERE breedsTable.name = [searchCriteria]
```

For the breed that has populated information, if there are needs, get as many as are listed:

```
SELECT needsTable.need FROM needsTable
    INNER JOIN need_breed on needsTable.nID = need_breed.need
    INNER JOIN breedsTable on breedsTable.bID = need_breed.breed
    WHERE breedsTable.name = [searchCriteria];
```

Get all Chicken Breed Information : Search by Egg Color

Get all information about breed except needs:

```
SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose,
temperTable.trait FROM breedsTable
```

INNER JOIN eggTable on breedsTable.egg = eggTable.eID

INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable on breedsTable.temper = temperTable.tID

WHERE eggTable.eggType = [searchCriteria];

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

SELECT needsTable.need FROM needsTable

INNER JOIN need_breed on needsTable.nID = need_breed.need

INNER JOIN breedsTable on breedsTable.bID = need_breed.breed

INNER JOIN eggTable on eggTable.eID = breedsTable.egg

WHERE eggTable.eggType = [searchCriteria]

AND breedsTable.name = [current breed's name];

Get all Chicken Breed Information : Search by Egg Quantity

Get all information about breed except needs:

SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose, temperTable.trait FROM breedsTable

INNER JOIN eggTable on breedsTable.egg = eggTable.eID

INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable on breedsTable.temper = temperTable.tID

WHERE eggTable.quantity = [amountDesired];

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

SELECT needsTable.need FROM needsTable

INNER JOIN need_breed on needsTable.nID = need_breed.need

INNER JOIN breedsTable on breedsTable.bID = need_breed.breed

INNER JOIN eggTable on eggTable.eID = breedsTable.egg

WHERE eggTable.quantity = [amountDesired]

AND breedsTable.name = [current breed's name];

Get all Chicken Breed Information : Search by Chicken Purpose

Get all information about breed except needs:

SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose, temperTable.trait FROM breedsTable

INNER JOIN eggTable on breedsTable.egg = eggTable.eID

INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable on breedsTable.temper = temperTable.tID

WHERE purposeTable.purpose = [searchCriteria];

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

SELECT needsTable.need FROM needsTable

INNER JOIN need_breed on needsTable.nID = need_breed.need

INNER JOIN breedsTable on breedsTable.bID = need_breed.breed

INNER JOIN purposeTable on purposeTable.pID = breedsTable.purpose

WHERE purposeTable.purpose = [searchCriteria]

AND breedsTable.name = [current breed's name];

Get all Chicken Breed Information : Search by Chicken Temperament

Get all information about breed except needs:

SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose, temperTable.trait FROM breedsTable

INNER JOIN eggTable on breedsTable.egg = eggTable.eID

INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable on breedsTable.temper = temperTable.tID

WHERE trait = [searchCriteria];

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

```
SELECT needsTable.need FROM needsTable
    INNER JOIN need_breed on needsTable.nID = need_breed.need
    INNER JOIN breedsTable on breedsTable.bID = need_breed.breed
    INNER JOIN temperTable on temperTable.tID = breedsTable.temper
    WHERE trait = [searchCriteria]
    AND breedsTable.name = [current breed's name];
```

Get all Chicken Breed Information : Search by Special Care Needs

```
SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose,
temperTable.trait FROM breedsTable
    INNER JOIN eggTable on breedsTable.egg = eggTable.eID
    INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID
    INNER JOIN temperTable on breedsTable.temper = temperTable.tID
    INNER JOIN need_breed on breedsTable.bID = need_breed.breed
    INNER JOIN needsTable on needsTable.nID = need_breed.need
    WHERE needsTable.need = [searchCriteria];
```

```
SELECT needsTable.need FROM needsTable
    INNER JOIN need_breed on needsTable.nID = need_breed.need
    INNER JOIN breedsTable on breedsTable.bID = need_breed.breed
    WHERE needsTable.need = [searchCriteria]
    AND breedsTable.name = [current breed's name];
```

Get all Chicken Breed Information : Search by **NOT IN** Egg Color

Get all information about breed except needs:

```
SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose,
temperTable.trait
    FROM breedsTable
```

INNER JOIN eggTable ON breedsTable.egg = eggTable.eID

INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable ON breedsTable.temper = temperTable.tID

WHERE breedsTable.bID NOT

IN (SELECT breedsTable.bID FROM breedsTable

INNER JOIN eggTable ON breedsTable.egg = eggTable.eID

WHERE eggTable.eggType = [searchCriteria]);

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

SELECT needsTable.need FROM needsTable

INNER JOIN need_breed on needsTable.nID = need_breed.need

INNER JOIN breedsTable on breedsTable.bID = need_breed.breed

INNER JOIN eggTable on eggTable.eID = breedsTable.egg

WHERE breedsTable.bID NOT

IN (SELECT breedsTable.bID FROM breedsTable

INNER JOIN eggTable ON breedsTable.egg = eggTable.eID

WHERE eggTable.eggType = [searchCriteria])

AND breedsTable.name = [current breed's name];

Get all Chicken Breed Information : Search by **NOT IN** Egg Amount

Get all information about breed except needs:

SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose, temperTable.trait

FROM breedsTable

INNER JOIN eggTable ON breedsTable.egg = eggTable.eID

INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable ON breedsTable.temper = temperTable.tID

WHERE breedsTable.bID NOT

```
IN (SELECT breedsTable.bID FROM breedsTable
INNER JOIN eggTable ON breedsTable.egg = eggTable.eID
WHERE eggTable.quantity = [searchAmount]);
```

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

```
SELECT needsTable.need FROM needsTable
    INNER JOIN need_breed on needsTable.nID = need_breed.need
    INNER JOIN breedsTable on breedsTable.bID = need_breed.breed
    INNER JOIN eggTable on eggTable.eID = breedsTable.egg
    WHERE breedsTable.bID NOT
    IN (SELECT breedsTable.bID FROM breedsTable
    INNER JOIN eggTable ON breedsTable.egg = eggTable.eID
    WHERE eggTable.quantity = [searchAmount])
    AND breedsTable.name = [current breed's name];
```

Get all Chicken Breed Information : Search by **NOT IN** purpose

Get all information about breed except needs:

```
SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose,
temperTable.trait
    FROM breedsTable
    INNER JOIN eggTable ON breedsTable.egg = eggTable.eID
    INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID
    INNER JOIN temperTable ON breedsTable.temper = temperTable.tID
    WHERE breedsTable.bID NOT
    IN (SELECT breedsTable.bID FROM breedsTable
    INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID
    WHERE purposeTable.purpose = [searchCriteria]);
```

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

```
SELECT needsTable.need FROM needsTable
    INNER JOIN need_breed on needsTable.nID = need_breed.need
    INNER JOIN breedsTable on breedsTable.bID = need_breed.breed
    INNER JOIN purposeTable on purposeTable.pID = breedsTable.purpose
    WHERE breedsTable.bID NOT
    IN (SELECT breedsTable.bID FROM breedsTable
    INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID
    WHERE purposeTable.purpose = [searchCriteria])
    AND breedsTable.name = [current breed's name];
```

Get all Chicken Breed Information : Search by **NOT IN** Temperament

Get all information about breed except needs:

```
SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose,
temperTable.trait
    FROM breedsTable
    INNER JOIN eggTable ON breedsTable.egg = eggTable.eID
    INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID
    INNER JOIN temperTable ON breedsTable.temper = temperTable.tID
    WHERE breedsTable.bID NOT
    IN (SELECT breedsTable.bID FROM breedsTable
    INNER JOIN temperTable ON breedsTable.temper = temperTable.tID
    WHERE temperTable.trait = [searchCriteria]);
```

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

```
SELECT needsTable.need FROM needsTable
    INNER JOIN need_breed on needsTable.nID = need_breed.need
```

INNER JOIN breedsTable on breedsTable.bID = need_breed.breed

INNER JOIN temperTable on temperTable.tID = breedsTable.temper

WHERE breedsTable.bID NOT

IN (SELECT breedsTable.bID FROM breedsTable

INNER JOIN temperTable ON breedsTable.temper = temperTable.tID

WHERE temperTable.trait = [searchCriteria])

AND breedsTable.name = [currently selected breed's name];

Get all Chicken Breed Information : Search by **NOT IN** needs

Get all information about breed except needs:

SELECT breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose, temperTable.trait, needsTable.need

FROM breedsTable

INNER JOIN eggTable ON breedsTable.egg = eggTable.eID

INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable ON breedsTable.temper = temperTable.tID

INNER JOIN need_breed ON breedsTable.bID = need_breed.breed

INNER JOIN needsTable ON need_breed.need = needsTable.nID

WHERE breedsTable.bID NOT

IN (SELECT breedsTable.bID FROM breedsTable

INNER JOIN need_breed ON breedsTable.bID = need_breed.breed

INNER JOIN needsTable ON need_breed.need = needsTable.nID

WHERE needsTable.need = [searchCriteria]);

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

SELECT needsTable.need FROM needsTable

INNER JOIN need_breed on needsTable.nID = need_breed.need

INNER JOIN breedsTable on breedsTable.bID = need_breed.breed

```
    WHERE breedsTable.bID NOT
    IN (SELECT breedsTable.bID FROM breedsTable
    INNER JOIN need_breed ON breedsTable.bID = need_breed.breed
    INNER JOIN needsTable ON need_breed.need = needsTable.nID
    WHERE needsTable.need = [searchCriteria])
    AND breedsTable.name = [currently selected breed's name];
```

Aggregate Query: Average Amount of Eggs by Temperament

```
SELECT temperTable.trait, AVG(eggTable.quantity)
    FROM breedsTable
    INNER JOIN eggTable ON breedsTable.egg = eggTable.eID
    INNER JOIN temperTable ON breedsTable.temper = temperTable.tID
    GROUP BY temperTable.trait;
```

Aggregate Query: Average Amount of Eggs by Purpose

```
SELECT purposeTable.purpose, AVG(eggTable.quantity)
    FROM breedsTable
    INNER JOIN eggTable ON breedsTable.egg = eggTable.eID
    INNER JOIN purposeTable ON breedsTable.purpose = purposeTable.pID
    GROUP BY purposeTable.purpose;
```

Aggregate Query: Total Number of Chickens in Flock

```
SELECT flock.id, breedsTable.bID, COUNT(flock.fID)
    FROM breedsTable
    INNER JOIN eggTable ON breedsTable.egg = eggTable.eID
    INNER JOIN flock ON breedsTable.bID = flock.id;
```

Aggregate Query: Total Amount of Eggs Produced by Flock

```
SELECT flock.id, breedsTable.bID, SUM(eggTable.quantity)
```

FROM breedsTable

INNER JOIN flock ON breedsTable.bID = flock.id

INNER JOIN eggTable ON breedsTable.egg = eggTable.eID;

Delete From Flock

DELETE FROM flock WHERE flock.fID = [selected row's fID];

Get All Flock Information

All information but needs:

SELECT breedsTable.bID, flock.id, flock.fID, breedsTable.name, eggTable.eggType, eggTable.quantity, purposeTable.purpose, temperTable.trait FROM breedsTable

INNER JOIN eggTable on breedsTable.egg = eggTable.eID

INNER JOIN purposeTable on breedsTable.purpose = purposeTable.pID

INNER JOIN temperTable on breedsTable.temper = temperTable.tID

INNER JOIN flock on breedsTable.bID = flock.fID;

For the breed that has populated information, if there are needs, get as many as are listed before moving onto next breed that fits search criteria:

SELECT needsTable.need FROM needsTable

INNER JOIN need_breed on needsTable.nID = need_breed.need

INNER JOIN breedsTable on breedsTable.bID = need_breed.breed

WHERE breedsTable.name = [currently selected breed's name];

User Adding to a Table

INSERT INTO eggTable (eggType, quantity) VALUES ([color], [quantity]);

INSERT INTO purposeTable (purpose) VALUES ([purpose]);

INSERT INTO temperTable (trait) VALUES ([trait]);

INSERT INTO breedsTable(name, egg, purpose, temper)

VALUES ([breedName],

(SELECT eID FROM eggTable WHERE eggType= [color] AND quantity= [quantity]),

(SELECT pID FROM purposeTable WHERE purpose= [purpose]),

(SELECT tID FROM temperTable WHERE trait= [trait]));

INSERT INTO flock(id) VALUES ((SELECT bID FROM breedsTable WHERE name= [breedName]));

Adding to Joining Table for Many-to-Many

INSERT INTO need_breed(breed, need)

VALUES ((SELECT bID FROM breedsTable WHERE name= [breedName]),

(SELECT nID FROM needsTable WHERE need= [need]));

Queries for Making Dropdown Options

SELECT DISTINCT eggType, quantity FROM eggTable;

SELECT DISTINCT purpose FROM purposeTable;

SELECT DISTINCT trait FROM temperTable;

SELECT DISTINCT need FROM needsTable;

SELECT DISTINCT name FROM breedsTable;