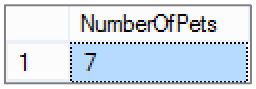
Requested solutions for Review Questions: 28-32,34-36,38-40,44-46 from Chapter 3. Write an SQL statement to count the number of pets.

```
SELECT    COUNT (*) AS NumberOfPets
FROM      PET;
```

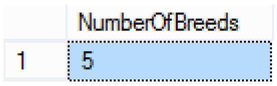


```
   NumberOfPets
1  7
```

3.28 Write an SQL statement to count the number of distinct breeds.

For SQL Server, Oracle Database and MySQL:

```
SELECT      Count( DISTINCT PetBreed) AS NumberOfPetBreeds
FROM        PET;
```



```
   NumberOfBreeds
1  5
```

For Microsoft Access:

The SQL solution requires the use of DISTINCT as part of the Count expression, but Microsoft Access SQL does not support this. (See "Does Not Work with Microsoft Access SQL" on p. 129.) However, there is a work around; we can use a subquery to determine the distinct PetBreeds, and then count that result:

```
SELECT    COUNT([PetBreed]) AS NumberOfBreeds
FROM      (SELECT DISTINCT PetBreed FROM PET);
```

The following table schema for the PET_3 table is another alternate version of the PET table:

PET_3 (PetID, PetName, PetType, PetBreed, PetDOB, PetWeight, OwnerID)

Data for PET_3 is shown if Figure 3-20.Except as specifically noted in the question itself, use the PET_3 for your answers to all the remaining Review Questions.

FIGURE 3-20

PET_3 Data

PetID	PetName	PetType	PetBreed	PetDOB	PetWeight	OwnerID
1	King	Dog	Std. Poodle	27-Feb-11	25.5	1
2	Teddy	Cat	Cashmere	01-Feb-12	10.5	2
3	Fido	Dog	Std. Poodle	17-Jul-10	28.5	1
4	AJ	Dog	Collie Mix	05-May-11	20.0	3
5	Cedro	Cat	Unknown	06-Jun-09	9.5	2
6	Wooley	Cat	Unknown	NULL	9.5	2
7	Buster	Dog	Border Collie	11-Dec-08	25.0	4

3.29 *Write the required SQL statements to create the PET_3 table. Assume that PetWeight is Numeric(4,1).*

For Microsoft Access:

```
CREATE TABLE PET_3(
   PetID          Int            NOT NULL,
   PetName        Char (50)      NOT NULL,
   PetType        Char (25)      NOT NULL,
   PetBreed       VarChar(100)   NULL,
   PetDOB         DateTime       NULL,
   PetWeight      Numeric        NULL,
   OwnerID        Int            NOT NULL,
   CONSTRAINT     PET_3_PK       PRIMARY KEY(PetID),
   CONSTRAINT     PET_3_OWNER_FK FOREIGN KEY(OwnerID)
                     REFERENCES PET_OWNER(OwnerID)
   );
```

For SQL Server:

```
CREATE TABLE PET_3(
   PetID          Int            NOT NULL IDENTITY(101,1),
   PetName        Char (50)      NOT NULL,
   PetType        Char (25)      NOT NULL,
   PetBreed       VarChar(100)   NULL,
   PetDOB         DateTime       NULL,
   PetWeight      Numeric(4,1)   NULL,
   OwnerID        Int         NOT NULL,
   CONSTRAINT     PET_3_PK    PRIMARY KEY(PetID),
   CONSTRAINT     PET_3_OWNER_FK FOREIGN KEY(OwnerID)
                     REFERENCES PET_OWNER(OwnerID)
                        ON DELETE CASCADE
   );
```

For Oracle Database:

The SQL CREATE TABLE commands shown for SQL Server 2014 will also work for Oracle Database with only two modifications. First, Oracle Database does not support ON UPDATE referential integrity actions. Second, Oracle Database uses SEQUENCES to set surrogate keys and set starting values and increment values. Therefore, the definitions of the OwnerID and PropertyID surrogate values should be written as:

```
CREATE TABLE PET_3(
   PetID          Int            NOT NULL,
   PetName        Char(50)       NOT NULL,
   PetType        Char(25)       NOT NULL,
   PetBreed       VarChar(100)   NULL,
   PetDOB         DateTime       NULL,
   PetWeight      Number(4,1) NULL,
   OwnerID        Int            NOT NULL,
   CONSTRAINT     PET_PK         PRIMARY KEY(PetID)
   CONSTRAINT     OWNER_FK       FOREIGN KEY(OwnerID)
                        REFERENCES PET_OWNER(OwnerID)
                              ON DELETE NO ACTION
   );

CREATE SEQUENCE seqPetID INCREMENT BY 1 START WITH 1;
```

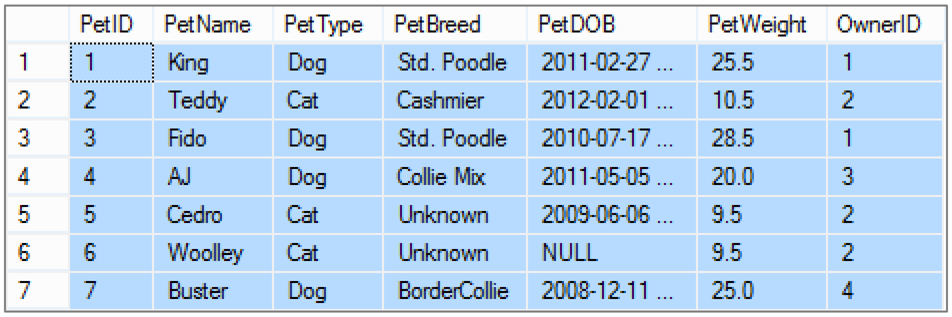
For MySQL:

MySQL uses the AUTO_INCREMENT keyword to implement surrogate keys. By default, AUTO_INCREMENT starts at 1 and increments by 1. Although the increment cannot be changed, the starting value can be reset using an ALTER command as shown below.

```
CREATE TABLE PET_3(
   PetID          Int               NOT NULL AUTO_INCREMENT,
   PetName        Char(50)          NOT NULL,
   PetType        Char(25)          NOT NULL,
   PetBreed       VarChar(100)      NULL,
   PetDOB         DateTime          NULL,
   PetWeight      Numeric(4,1)      NULL,
   OwnerID        Int               NOT NULL,
   CONSTRAINT     PET_3_PK          PRIMARY KEY(PetID),
   CONSTRAINT     PET_3_OWNER_FK    FOREIGN KEY(OwnerID)
                     REFERENCES PET_OWNER(OwnerID)
                        ON DELETE CASCADE
   );

ALTER TABLE PET AUTO_INCREMENT=1;
```

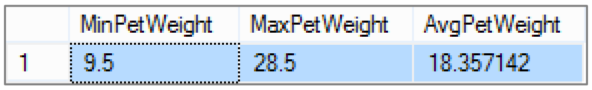
See Figure 3-20 for data for this table.



	PetID	PetName	PetType	PetBreed	PetDOB	PetWeight	OwnerID
1	1	King	Dog	Std. Poodle	2011-02-27 ...	25.5	1
2	2	Teddy	Cat	Cashmier	2012-02-01 ...	10.5	2
3	3	Fido	Dog	Std. Poodle	2010-07-17 ...	28.5	1
4	4	AJ	Dog	Collie Mix	2011-05-05 ...	20.0	3
5	5	Cedro	Cat	Unknown	2009-06-06 ...	9.5	2
6	6	Woolley	Cat	Unknown	NULL	9.5	2
7	7	Buster	Dog	BorderCollie	2008-12-11 ...	25.0	4

3.30 *Write an SQL statement to display the minimum, maximum, and average weight of dogs.*

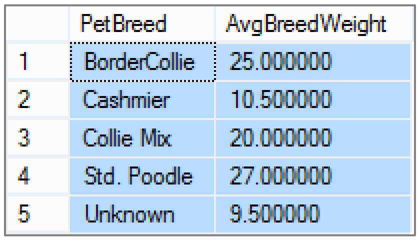
```
SELECT     MIN(PetWeight) AS MinPetWeight,
           MAX(PetWeight) AS MaxPetWeight,
           AVG(PetWeight) AS AvgPetWeight
FROM       PET_3;
```



	MinPetWeight	MaxPetWeight	AvgPetWeight
1	9.5	28.5	18.357142

3.31 *Write an SQL statement to group the data by PetBreed and display the average Weight per breed.*

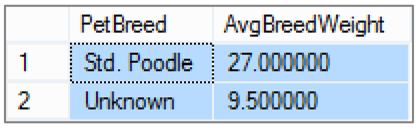
```
SELECT      PetBreed, AVG(PetWeight) AS AvgBreedWeight
FROM        PET_3
GROUP BY    PetBreed;
```



	PetBreed	AvgBreedWeight
1	BorderCollie	25.000000
2	Cashmier	10.500000
3	Collie Mix	20.000000
4	Std. Poodle	27.000000
5	Unknown	9.500000

3.32 *Answer question 3.32, but consider only breeds for which two or more pets are included in the database.*

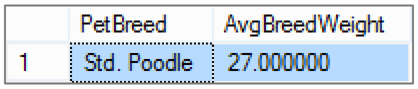
```
SELECT      PetBreed, AVG(PetWeight) AS AvgBreedWeight
FROM        PET_3
GROUP BY    PetBreed
HAVING      Count(*) > 1;
```



	PetBreed	AvgBreedWeight
1	Std. Poodle	27.000000
2	Unknown	9.500000

3.33 *Answer question 3.33, but do not consider any pet having the PetBreed value of Unknown.*

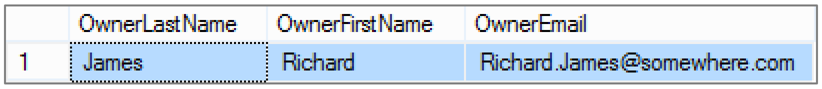
```
SELECT         PetBreed, AVG(PetWeight) AS AvgBreedWeight
FROM        PET_3
WHERE       PetBreed <> 'Unknown'
GROUP BY    PetBreed
HAVING      Count (*) > 1;
```



	PetBreed	AvgBreedWeight
1	Std. Poodle	27.000000

3.34 *Write an SQL statement to display the last name, first name, and email of any owners of cats. Use a subquery.*

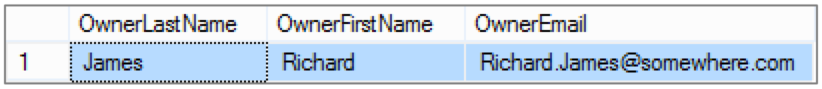
```
SELECT   OwnerLastName, OwnerFirstName, OwnerEmail
FROM     PET_OWNER
WHERE    OwnerID IN
            (SELECT   OwnerID
             FROM     PET
             WHERE       PetType = 'Cat');
```



	OwnerLastName	OwnerFirstName	OwnerEmail
1	James	Richard	Richard.James@somewhere.com

3.35 *Write an SQL statement to display the last name, first name, and email of any owners of cats with the name Teddy. Use a subquery.*

```
SELECT      OwnerLastName, OwnerFirstName, OwnerEmail
FROM        PET_OWNER
WHERE       OwnerID IN
            (SELECT  OwnerID
             FROM    PET
             WHERE      PetName= 'Teddy');
```



	OwnerLastName	OwnerFirstName	OwnerEmail
1	James	Richard	Richard.James@somewhere.com

The following table schema for the BREED Table shows a new table to be added to the PET database:

BREED (<u>BreedName</u>, MinWeight, MaxWeight, AverageLifeExpectancy)

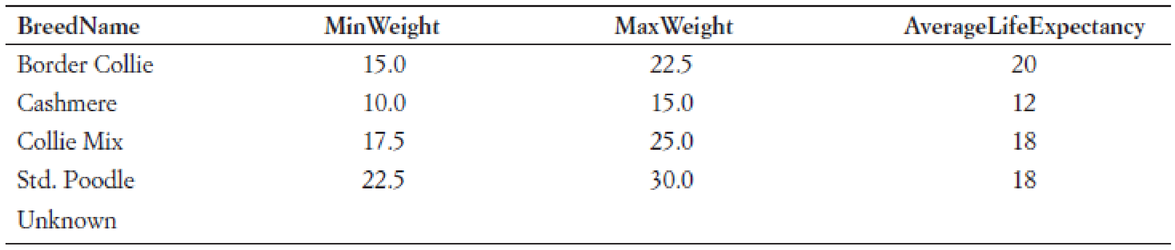
Assume that PetBreed in PET_3 is a foreign key that matches the primary key BreedName in BREED, and that BreedName in Breed is now a foreign key linking the two tables with the referential integrity constraint:

PetBreed in PET_3 must exist in BreedName in BREED

If needed, you may also assume that a similar referential integrity constraint exists between PET and BREED and between PET_2 and BREED. The BREED table data are shown in Figure 3-21.

FIGURE 3-21

BREED Data



BreedName	MinWeight	MaxWeight	AverageLifeExpectancy
Border Collie	15.0	22.5	20
Cashmere	10.0	15.0	12
Collie Mix	17.5	25.0	18
Std. Poodle	22.5	30.0	18
Unknown			

3.36 *Write SQL Statements to (1) create the BREED table, (2) insert the data in Figure3-20 into the BREED table, (3) alter the PET_3 table so that PetBreed is a foreign key referencing BreedName in BREED, and (4) to display the last name, first name, and email of any owner of a pet that has an AverageLifeExpectancy value greater than 15 using a subquery.*

For Microsoft Access:

Access SQL does not support the (m, n) extension of the Numeric data type. (See “Does Not Work with Microsoft Access ANSI-89 SQL” on p. 124.) Create the table with the following command, and then set the column properties in the GUI.

```
CREATE TABLE BREED(
   BreedName               VarChar(100)         NOT NULL,
   MinWeight               Numeric                    NULL,
   MaxWeight               Numeric                    NULL,
   AverageLifeExpectancy   Numeric                    NULL,
   CONSTRAINT              BREED_PK PRIMARY     KEY(BreedName)
   );
```

For SQL Server. Oracle Database and MySQL:

```
CREATE TABLE BREED(
   BreedName               VarChar(100)         NOT NULL,
   MinWeight               Numeric(4,1)         NULL,
   MaxWeight               Numeric(4,1)         NULL,
   AverageLifeExpectancy   Numeric(4,1)         NULL,
   CONSTRAINT              BREED_PK PRIMARY     KEY(BreedName)
   );
```

The Breed columns in PET and PET_3 will now become foreign keys, and if we haven't already included them in our CREATE TABLE statements, we will need to add an additional foreign key constraint to both tables. We will use the ALTER TABLE command as follows:

For Microsoft Access:

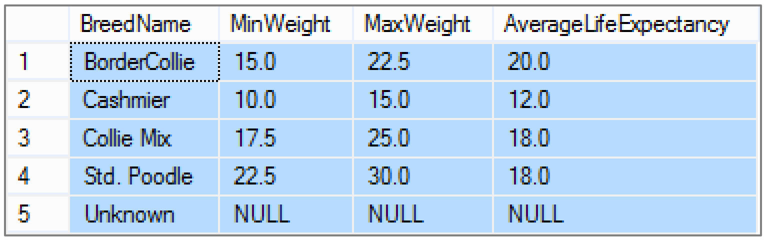
Microsoft Access does not support the ON DELETE and ON UPDATE constraint clauses.

```
ALTER TABLE PET
   ADD CONSTRAINT PET_BREED_FK FOREIGN KEY(Breed)
      REFERENCES BREED(BreedName);
ALTER TABLE PET_3
   ADD CONSTRAINT PET_3_BREED_FK FOREIGN KEY(Breed)
      REFERENCES BREED(BreedName);
```

For SQL Server, Oracle Database and MySQL:

```
ALTER TABLE PET
   ADD CONSTRAINT PET_BREED_FK FOREIGN KEY(PetBreed)
      REFERENCES BREED(BreedName)
         ON UPDATE CASCADE;
ALTER TABLE PET_3
   ADD CONSTRAINT PET_3_BREED_FK FOREIGN KEY(PetBreed)
      REFERENCES BREED(BreedName)
         ON UPDATE CASCADE;
```

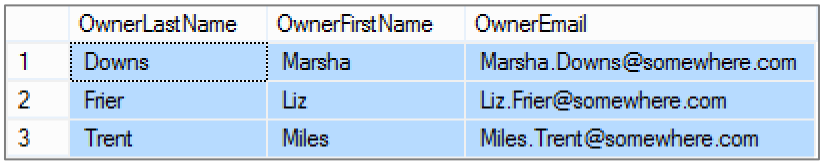
See Figure 3-21 for data for this table.



	BreedName	MinWeight	MaxWeight	AverageLifeExpectancy
1	BorderCollie	15.0	22.5	20.0
2	Cashmier	10.0	15.0	12.0
3	Collie Mix	17.5	25.0	18.0
4	Std. Poodle	22.5	30.0	18.0
5	Unknown	NULL	NULL	NULL

The query itself is:

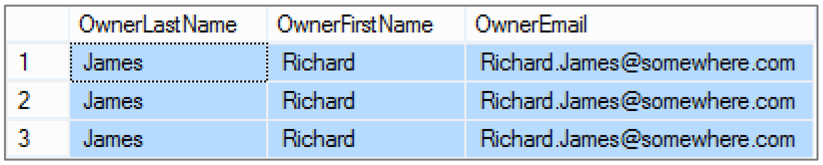
```
SELECT   OwnerLastName, OwnerFirstName, OwnerEmail
FROM     PET_OWNER
WHERE    OwnerID IN
         (SELECT  OwnerID
          FROM    PET
          WHERE      PetBreed IN
                  (SELECT  BreedName
                   FROM    BREED
                   WHERE      AverageLifeExpectancy > 15));
```



	OwnerLastName	OwnerFirstName	OwnerEmail
1	Downs	Marsha	Marsha.Downs@somewhere.com
2	Frier	Liz	Liz.Frier@somewhere.com
3	Trent	Miles	Miles.Trent@somewhere.com

3.37 Answer question 3.35, but use a join using JOIN ON syntax.

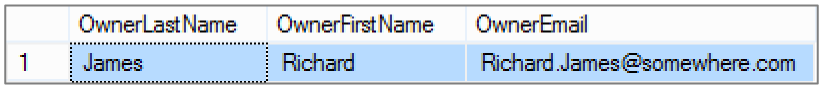
```
SELECT   OwnerLastName, OwnerFirstName, OwnerEmail
FROM     PET_OWNER as PO INNER JOIN PET as P
      ON PO.OwnerID = P.OwnerID
WHERE    PetType = 'Cat';
```



	OwnerLastName	OwnerFirstName	OwnerEmail
1	James	Richard	Richard.James@somewhere.com
2	James	Richard	Richard.James@somewhere.com
3	James	Richard	Richard.James@somewhere.com

We can also use DISTINCT to remove duplicate lines:

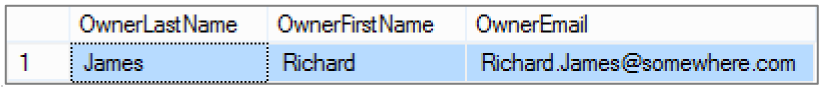
```
SELECT   DISTINCT OwnerLastName, OwnerFirstName, OwnerEmail
FROM     PET_OWNER as PO INNER JOIN PET as P
      ON PO.OwnerID = P.OwnerID
WHERE    PetType = 'Cat';
```



	OwnerLastName	OwnerFirstName	OwnerEmail
1	James	Richard	Richard.James@somewhere.com

3.38 Answer question 3.36, but use a join using JOIN ON syntax.

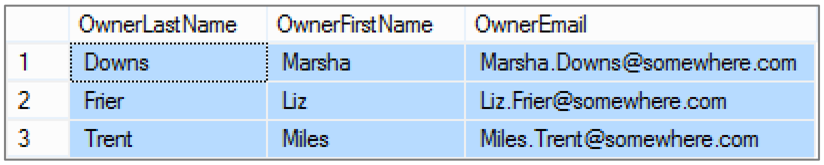
```
SELECT   OwnerLastName, OwnerFirstName, OwnerEmail
FROM     PET_OWNER INNER JOIN PET
      ON PET_OWNER.OwnerID = PET.OwnerID
WHERE    PetName = 'Teddy';
```



	OwnerLastName	OwnerFirstName	OwnerEmail
1	James	Richard	Richard.James@somewhere.com

3.39 *Answer part (4) of question 3.37, but use joins using JOIN ON syntax.*

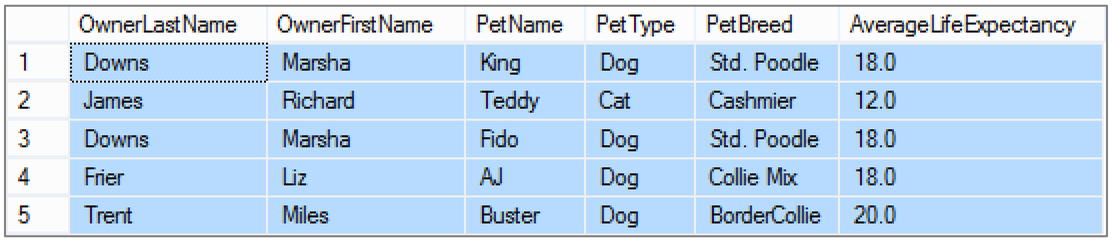
```
SELECT   DISTINCT OwnerLastName, OwnerFirstName, OwnerEmail
FROM     (PET_OWNER as PO JOIN PET as P
            ON PO.OwnerID = P.OwnerID)
               JOIN BREED as B
                  ON P.PetBreed = B.BreedName
WHERE    AverageLifeExpectancy > 15;
```



	OwnerLastName	OwnerFirstName	OwnerEmail
1	Downs	Marsha	Marsha.Downs@somewhere.com
2	Frier	Liz	Liz.Frier@somewhere.com
3	Trent	Miles	Miles.Trent@somewhere.com

3.40 *Write an SQL statement to display the OwnerLastName, OwnerFirstName, PetName, PetType, PetBreed, and AverageLifeExpectancy for pets with a known PetBreed.*

```
SELECT   OwnerLastName, OwnerFirstName,
         PetName, PetType, PetBreed,
         AverageLifeExpectancy
FROM     PET_OWNER JOIN PET
   ON    PET_OWNER.OwnerID = PET.OwnerID
            JOIN BREED
               ON PET.PetBreed = BREED.BreedName
WHERE    PetBreed <> 'Unknown';
```



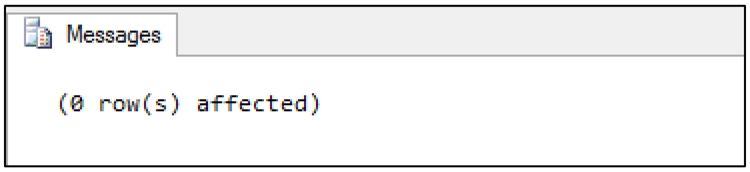
	OwnerLastName	OwnerFirstName	PetName	PetType	PetBreed	AverageLifeExpectancy
1	Downs	Marsha	King	Dog	Std. Poodle	18.0
2	James	Richard	Teddy	Cat	Cashmier	12.0
3	Downs	Marsha	Fido	Dog	Std. Poodle	18.0
4	Frier	Liz	AJ	Dog	Collie Mix	18.0
5	Trent	Miles	Buster	Dog	BorderCollie	20.0

3.44 *Explain what will happen if you leave the WHERE clause off your answer to question 3.44.*

All pets would have the Breed 'Std. Poodle'

3.45 *Write an SQL statement to delete all rows of pets of type Anteater. What will happen if you forget to code the WHERE clause in this statement?*

```
DELETE
FROM     PET
WHERE    PetType = 'Anteater';
```



```
(0 row(s) affected)
```

As there are NO anteaters in the PET_3 table, in PET, no rows are affected.

If the WHERE clause is omitted, we'd delete *all* the rows in PET—***DO NOT*** run that command