**Jennifer Faer**

**IS 5313 Structured Data and Querying**

**Dr. Bill Rosener**

**Fall 2024**

https://billrosener.com/teaching/is4293/assignments/assignment3/assignment3-description.html

**Chapter 3 Review Questions & SQL Code**

1. What does SQL stand for?

Structured Query Language (Page 151).

1. What is a data sublanguage?

A data sublanguage is “a language for defining and processing a database intended to be embedded in programs written in another language,” and it is an incomplete programming language because it only defines and processes data, and does not stand on its own. (page 151)

1. Explain the importance of SQL-92.

SQL-92 is the version of SQL that was adopted by ANSI in 1992. (page 151)

1. Why is it important to learn SQL?

SQL helps database managers to maximize their potential when working with database programs such as MS Access. (page 151)

1. Describe in your own words the purpose of the two business rules listed on page 158.

To recap, the two business rules are the following:

1. If an EMPLOYEE row is to be deleted and that row is connected to any ASSIGNMENT, the EMPLOYEE row deletion will be disallowed.
2. If a PROJECT row is deleted, then all the ASSIGNMENT rows that are connected to the deleted PROJECT row will also be deleted.

The purpose of the first rule is to prevent the loss or abandonment of assignments (work or work products) when an employee leaves an organization, by ensuring that the tasks are re-assigned to another employee. The second rule assures that employees are not working on unnecessary tasks and also maintains a clean database free of unnecessary assignments.

1. Why do some standard SQL-92 statements fail to run successfully in Microsoft Access?

MS Access defaults to SQL-89, which is slightly different from SQL-92. There is a setting to change this so that SQL-92 statements are accepted. (page 159)

Use the following tables for your answers to questions 3.7 through 3.48:

**PET\_OWNER (OwnerID, OwnerLastName, OwnerFirstName, OwnerPhone, OwnerEmail)**

**PET (PetID, PetName, PetType, PetBreed, PetDOB, OwnerID)**

Sample data for these tables are shown in Figures 3-27 and 3-28. For each SQL statement you write, show the results based on these data. If possible, run the statements you write for the questions that follow in an actual DBMS, as appropriate, to obtain results. Use data types that are consistent with the DBMS you are using. If you are not using an actual DBMS, consistently represent data types by using either the MySQL, Microsoft SQL Server, or Oracle Database data types shown in Figure 3-5.



1. Write an SQL CREATE TABLE statement to create the PET\_OWNER table, with OwnerID as a surrogate key. Justify your choices of column properties. If you are using an actual DBMS, also insert the data using SQL.

CREATE TABLE PET\_OWNER (

 OwnerID Int AUTO\_INCREMENT PRIMARY KEY,

 OwnerLastName Char(35) NOT NULL,

 OwnerFirstName Char(35) NOT NULL,

 OwnerPhone Char(12) NOT NULL UNIQUE,

 OwnerEmail VarChar(100) UNIQUE);

Owner ID is the primary key.

Owner names should never be null (a person always has a name), and 35 chars can accommodate longer or hyphenated names, and more inclusive to more cultures.

Owner phone is constrained to 12 chars to follow the format 123-456-7890 and assumes that these are US numbers. If the entity is likely to have clients with foreign cell phones (such as travelers) or the entity is not located in the US, this should be changed. Phone number also should not be duplicated in the system, and assumes that everyone has their own unique phone number. (This could be an issue if two pet owners from the same home want separate accounts, and they share a landline rather than have a cell phone. However, I am going to commit to this format for this assignment.)

Email is a variable character data type to save space in the database (emails can vary widely), should not be duplicated in the database, and is not required because many people don’t use email or want email from businesses.

1. Write an SQL CREATE TABLE statement to create the PET table without a referential integrity constraint on OwnerID in PET. Justify your choices of column properties. Why not make every column NOT NULL? If you are using an actual DBMS, also insert the data using SQL.

CREATE TABLE PET (

PetID Int AUTO\_INCREMENT PRIMARY KEY,

PetName Char(35) NOT NULL,

PetType Char(35) NOT NULL,

PetBreed Char(35),

PetDOB DATE,

OwnerID Int);

Making every column NOT NULL can be unnecessarily restrictive, as someone may not know their pet’s breed and it’s not necessary in most cases. Similarly, birthdate may not be known, but it may be worth altering the field so just an (approximate) year is included (see 3.11). OwnerID should probably be NOT NULL as well, but I have some reservations, such as scenarios where a pet may be abandoned, the owner passes away, or a stray is brought in. This depends on the purpose of the database and the information the organization needs.

1. Create a referential integrity constraint on OwnerID in PET. Assume that deletions should not cascade.

FOREIGN KEY (OwnerID) REFERENCES PET(OwnerID) ON DELETE RESTRICT;

1. Create a referential integrity constraint on OwnerID in PET. Assume that deletions should cascade.

FOREIGN KEY (OwnerID) REFERENCES PET(OwnerID) ON DELETE CASCADE;

The following table schema for the PET\_2 table is an alternate version of the PET table- use it to answer review questions 3.11 and 3.12:

**PET\_2 (PetName, PetType, PetBreed, PetDOB, OwnerID)**

1. Write the required SQL statements to create the PET\_2 table.

CREATE TABLE PET\_2 (

 PetID INT AUTO\_INCREMENT PRIMARY KEY,

 PetName Char(35) NOT NULL,

 PetType Char(25) NOT NULL,

 PetBreed Char(25),

 PetDOB VarChar(10),

 FOREIGN KEY (OwnerID));

1. Is PET or PET\_2 a better design? Explain your rationale.

PET\_2 is a better design because of the foreign key linking each pet record with a record from the PET\_OWNER table. The instructions did not specify a Primary Key in PET\_2 but I added one anyway; without the primary key in PET\_2, PET may be a better table, but having a foreign key is better database management for a scenario like this.

1. Write the SQL statements necessary to remove the PET\_OWNER table from the database. Assume that the referential integrity constraint is to be removed. **Do not run these commands in an actual database!**

ALTER TABLE PET

DROP FOREIGN KEY PET(OwnerID);

DROP TABLE PET\_OWNER;

1. Write the SQL statements necessary to remove the PET\_OWNER table from the database. Assume that the PET table is to be removed. **Do not run these commands in an actual database!**

ALTER TABLE PET

DROP FOREIGN KEY PET(OwnerID);

DROP TABLE PET;

DROP TABLE PET\_OWNER;

1. Write an SQL statement to display all columns of all rows of PET. Do not use the asterisk (\*) notation.

SELECT PetID, PetName, PetType, PetBreed, PetDOB, OwnerID FROM PET;

1. Write an SQL statement to display all columns of all rows of PET. Use the asterisk (\*) notation.

NOTE: Since this is the same as 3.15, I’m going to assume this should be for PET\_2.

SELECT PetID, PetName, PetType, PetBreed, PetDOB, OwnerID FROM PET\_2;

1. Write an SQL statement to display the breed and type of all pets.

SELECT PetBreed, PetType FROM PET;

1. Write an SQL statement to display the breed, type, and DOB of all pets having the type Dog.

SELECT PetBreed, PetType, PetDOB FROM PET WHERE PetType = ‘Dog’;

1. Write an SQL statement to display the PetBreed column of PET.

SELECT PetBreed FROM PET;

1. Write an SQL statement to display the PetBreed column of PET. Do not show duplicates.

SELECT DISTINCT PetBreed FROM PET;

1. Write an SQL statement to display the breed, type, and DOB for all pets having the type Dog and the breed Std. Poodle.

SELECT PetBreed, PetType, PetDOB FROM PET WHERE PetType = ‘Dog’ AND PetBreed = ‘Std. Poodle’;

1. Write an SQL statement to display the name, breed, and type for all pets that are not of type Cat, Dog, or Fish.

SELECT PetName, PetBreed, PetType FROM PET WHERE PetType NOT IN (‘Cat’, ‘Dog’, ‘Fish’);

1. Write an SQL statement to display the Pet ID, breed, and type for all pets having a four-character name starting with K. Note that the RTRIM function will be needed in the solution that uses a CHAR column, but not for one that uses a VARCHAR column.

SELECT PetID, PetBreed, PetType FROM PET WHERE RTRIM(PetName) LIKE 'K\_\_\_';

1. Write an SQL statement to display the last name, first name, and email of all owners who have an email address ending with somewhere.com. Assume that email account names can be any number of characters. Note that the RTRIM function will be needed in the solution that uses a CHAR column, but not for one that uses a VARCHAR column.

SELECT OwnerLastName, OwnerFirstName, OwnerEmail FROM PET\_OWNER WHERE OwnerEmail LIKE ‘%@somewhere.com’;

1. Write an SQL statement to display the last name, first name and email of any owner who has a NULL value for OwnerPhone.

SELECT OwnerLastName, OwnerFirstName, OwnerEmail FROM PET\_OWNER WHERE OwnerPhone IS NULL;

**SQL (Structured Query Language) Code**

After finishing the hands-on portion of Assignment 3, copy and paste your

SQL (Structured Query Language) code below.

1. SQL code to retrieve “All Customers” in the “CUSTOMER” table.

SELECT CUSTOMER.CustomerID, CUSTOMER.LastName, CUSTOMER.FIrstName, CUSTOMER.Address, CUSTOMER.City, CUSTOMER.State, CUSTOMER.ZIP FROM CUSTOMER;

2. SQL code to retrieve all customers from “Oklahoma” in the “CUSTOMER” table.

SELECT CUSTOMER.LastName, CUSTOMER.FIrstName, CUSTOMER.Address, CUSTOMER.City, CUSTOMER.State, CUSTOMER.ZIP

FROM CUSTOMER WHERE (((CUSTOMER.State)='OK'));

3. SQL code to retrieve all customers from “Tulsa” in the “CUSTOMER” table.

SELECT CUSTOMER.LastName, CUSTOMER.FIrstName, CUSTOMER.Address, CUSTOMER.City, CUSTOMER.State, CUSTOMER.ZIP, CUSTOMER.EmailAddress

FROM CUSTOMER WHERE (((CUSTOMER.City)='Tulsa'));

4. SQL code created to retrieve all “phone” contacts using a “JOIN” operation with the “CUSTOMER” and “CONTACT” table.

SELECT CUSTOMER.LastName, CUSTOMER.FIrstName, CONTACT.ContactDate, CONTACT.ContactType, CONTACT.Remarks

FROM CUSTOMER INNER JOIN CONTACT ON CUSTOMER.CustomerID = CONTACT.CustomerID WHERE (((CONTACT.ContactType)='Phone'));

5. SQL code created to retrieve all contacts created since January 1, 2024 using a “JOIN” operation with the “CUSTOMER” and “CONTACT” table.

SELECT CUSTOMER.LastName, CUSTOMER.FIrstName, CONTACT.ContactDate, CONTACT.ContactType, CONTACT.Remarks

FROM CUSTOMER INNER JOIN CONTACT ON CUSTOMER.CustomerID = CONTACT.CustomerID WHERE (((CONTACT.ContactDate)>=#1/1/2024#));