Notes on Relational Databases:

www.pgrocer.net/Cis50/relation.html

Notes on Relational Databases:

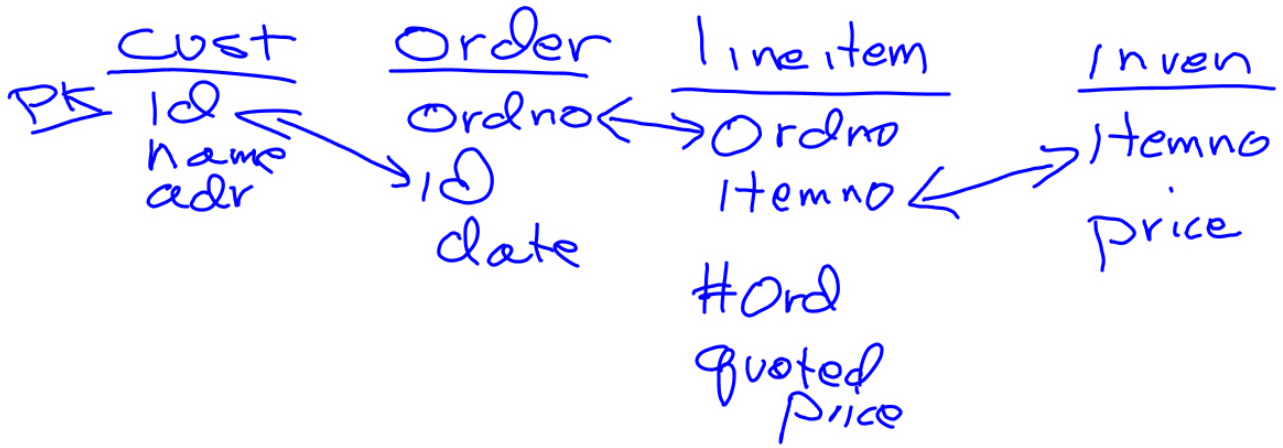
A database can consist of multiple tables/files. These tables/files are related to each other in some way so that the programmer or developer can access information from multiple table/files at the same time. A database management system is the database and its functionality. In today's PC environment, the model of relating the table/files within a database is the relational database model. Things that must be considered in a relational database:

- Analyze the data and determine how to design the files - consider whether the data is in a one to one relationship, a one to many relationship or a many to many relationship.
- Consider functional dependence: An attribute B, is functionally depended on another attribute, A if a value for A determines a single value for B at any one time.
- The primary key has all attributes in the table functionally dependent upon it - to simplify you can think of the primary key as the minimum collection of fields that will get you one and only one record from the table/file.
- We must also consider the rules of normalization and establish the relationships so that the database is in third normal form.

- A relation is in **first normal form** if it does not contain repeating groups.

- A relation is in **second normal form** if it is in first normal form and no non-key attribute is dependent on only a portion of the primary key. (Note: an attribute is a non-key attribute if it is not a part of the primary key).

- A relation is in **third normal form** if it is in second normal form and if the only determinants it contains are candidate keys. (Note: any attribute that determines another attribute is called a determinant).



9:48 AM
9/20/2018

Notes on Relational Databases:

www.pgrocer.net/Cis50/relation.html

Notes on Relational Databases:

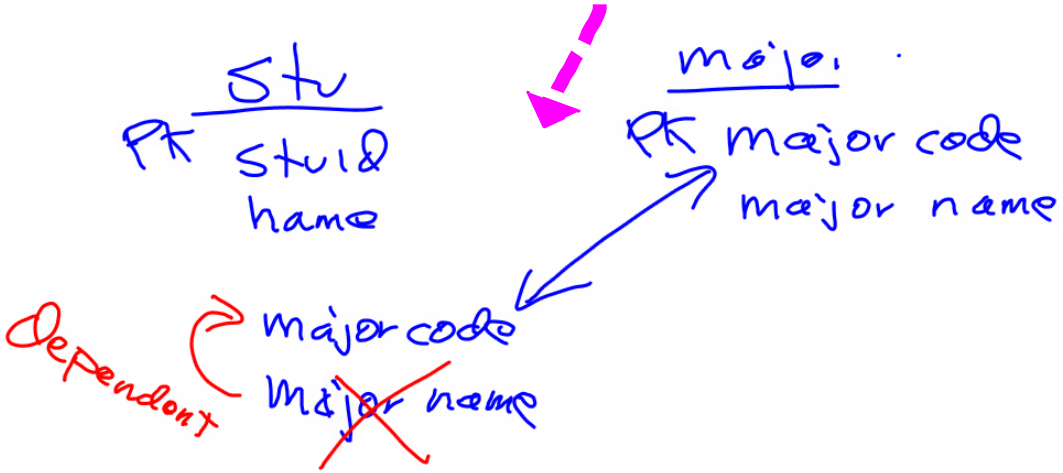
A database can consist of multiple tables/files. These tables/files are related to each other in some way so that the programmer or developer can access information from multiple table/files at the same time. A database management system is the database and its functionality. In today's PC environment, the model of relating the table/files within a database is the relational database model. Things that must be considered in a relational database:

- Analyze the data and determine how to design the files - consider whether the data is in a one to one relationship, a one to many relationship or a many to many relationship.
- Consider functional dependence: An attribute B, is functionally depended on another attribute, A if a value for A determines a single value for B at any one time.
- The primary key has all attributes in the table functionally dependent upon it - to simplify you can think of the primary key as the minimum collection of fields that will get you one and only one record from the table/file.
- We must also consider the rules of normalization and establish the relationships so that the database is in third normal form.

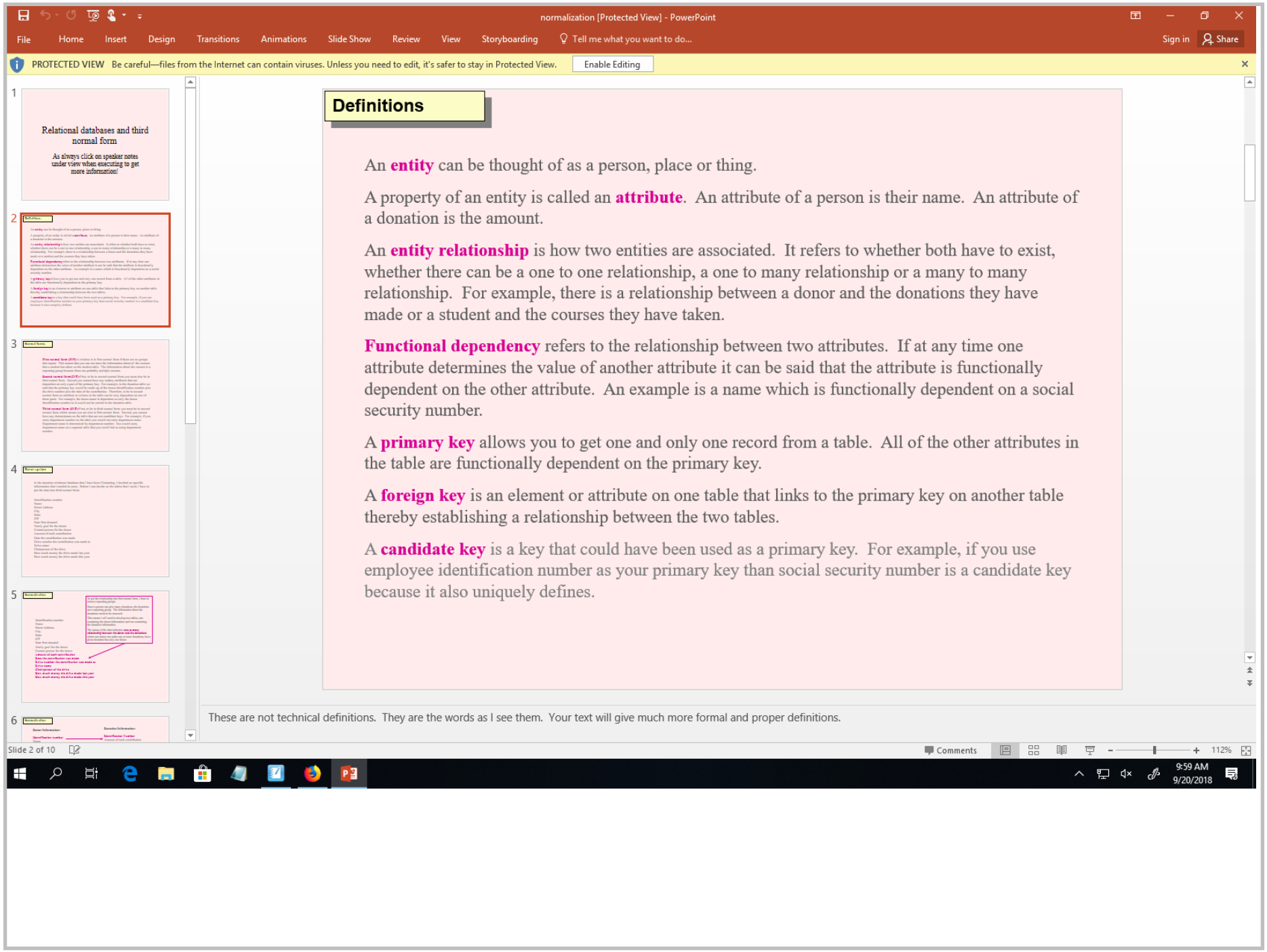
- A relation is in **first normal form** if it does not contain repeating groups.

- A relation is in **second normal form** if it is in first normal form and no non-key attribute is dependent on only a portion of the primary key. (Note: an attribute is a non-key attribute if it is not a part of the primary key).

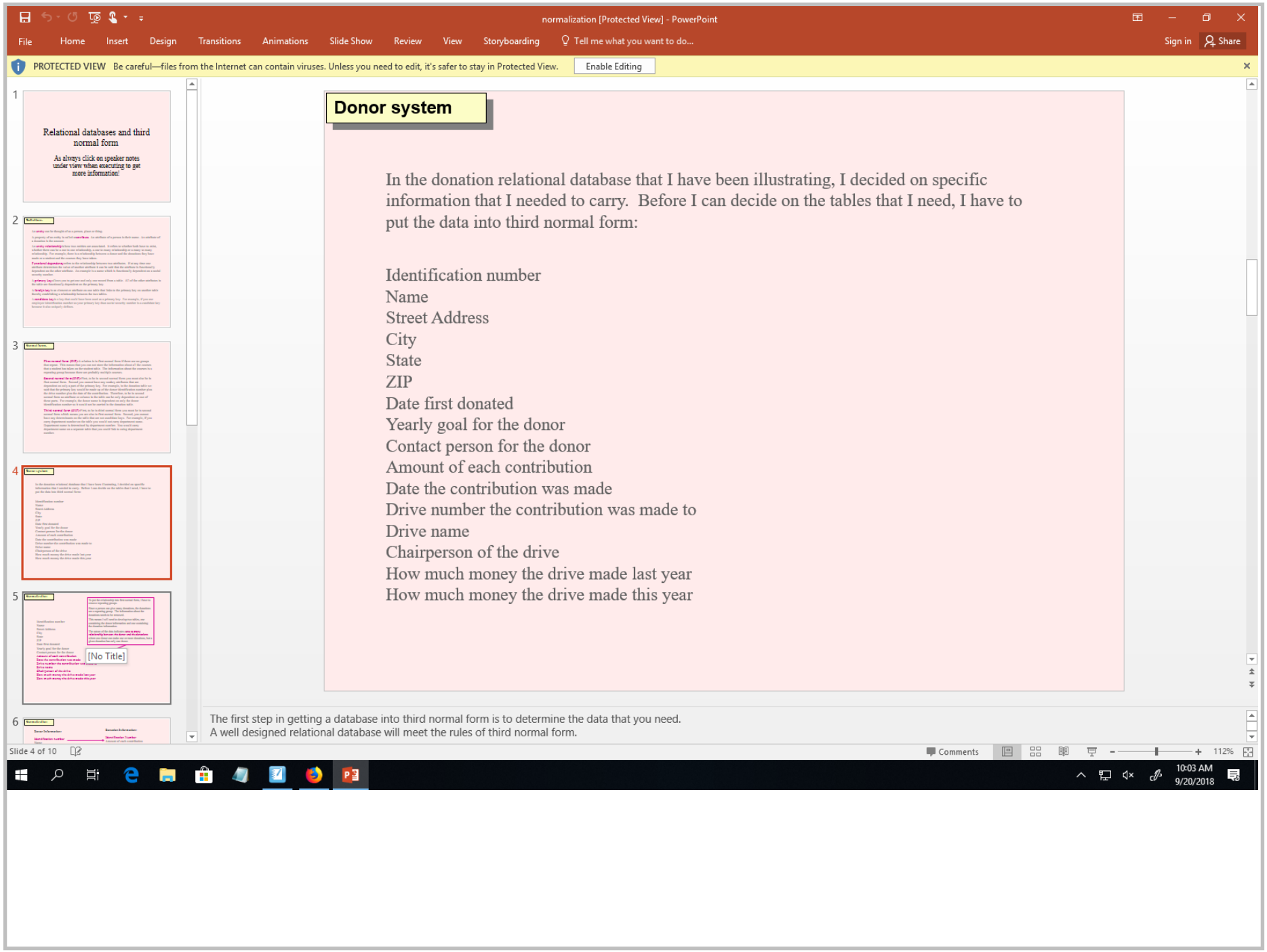
- A relation is in **third normal form** if it is in second normal form and if the only determinants it contains are candidate keys. (Note: any attribute that determines another attribute is called a determinant).



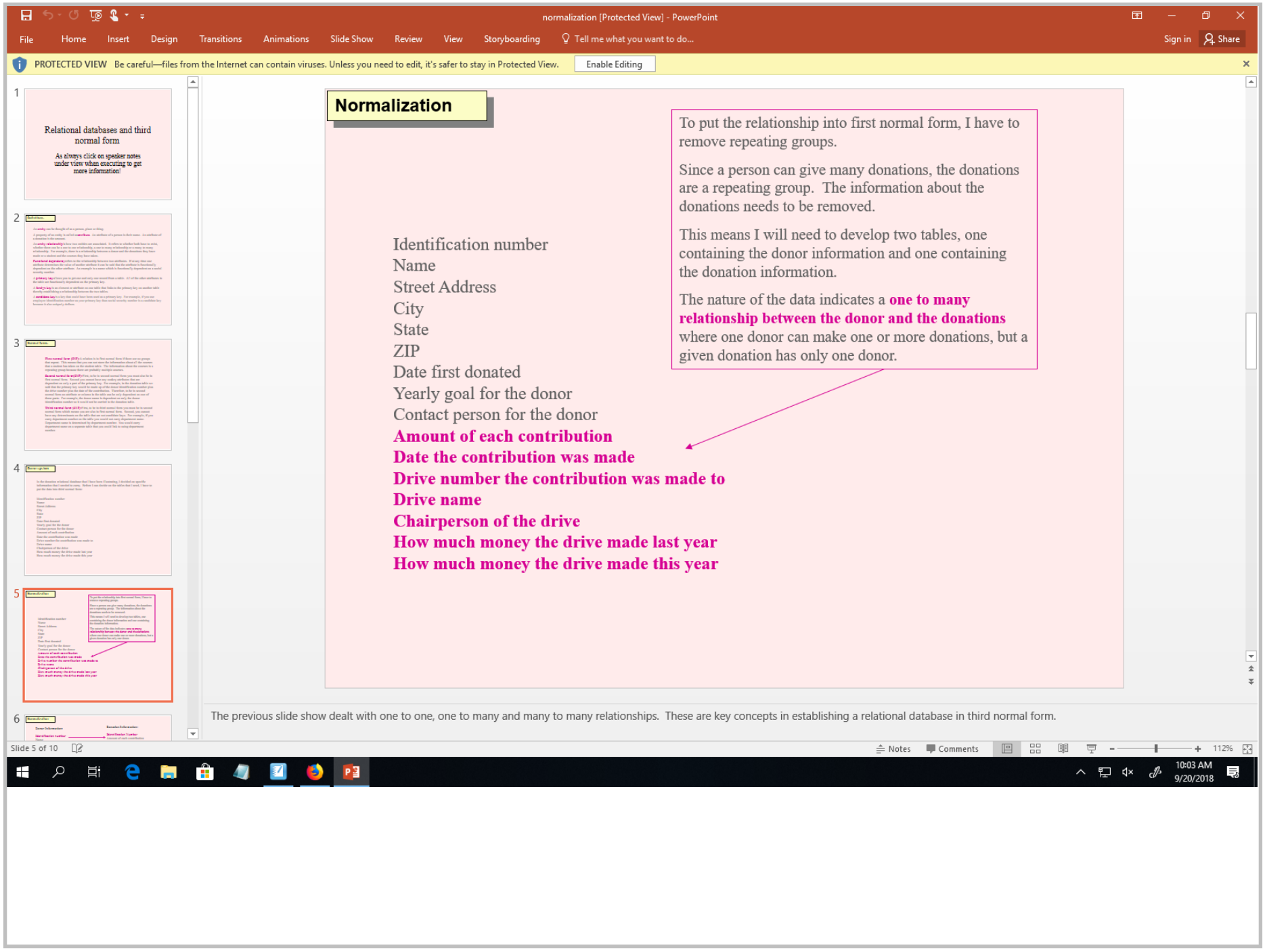
9:57 AM
9/20/2018



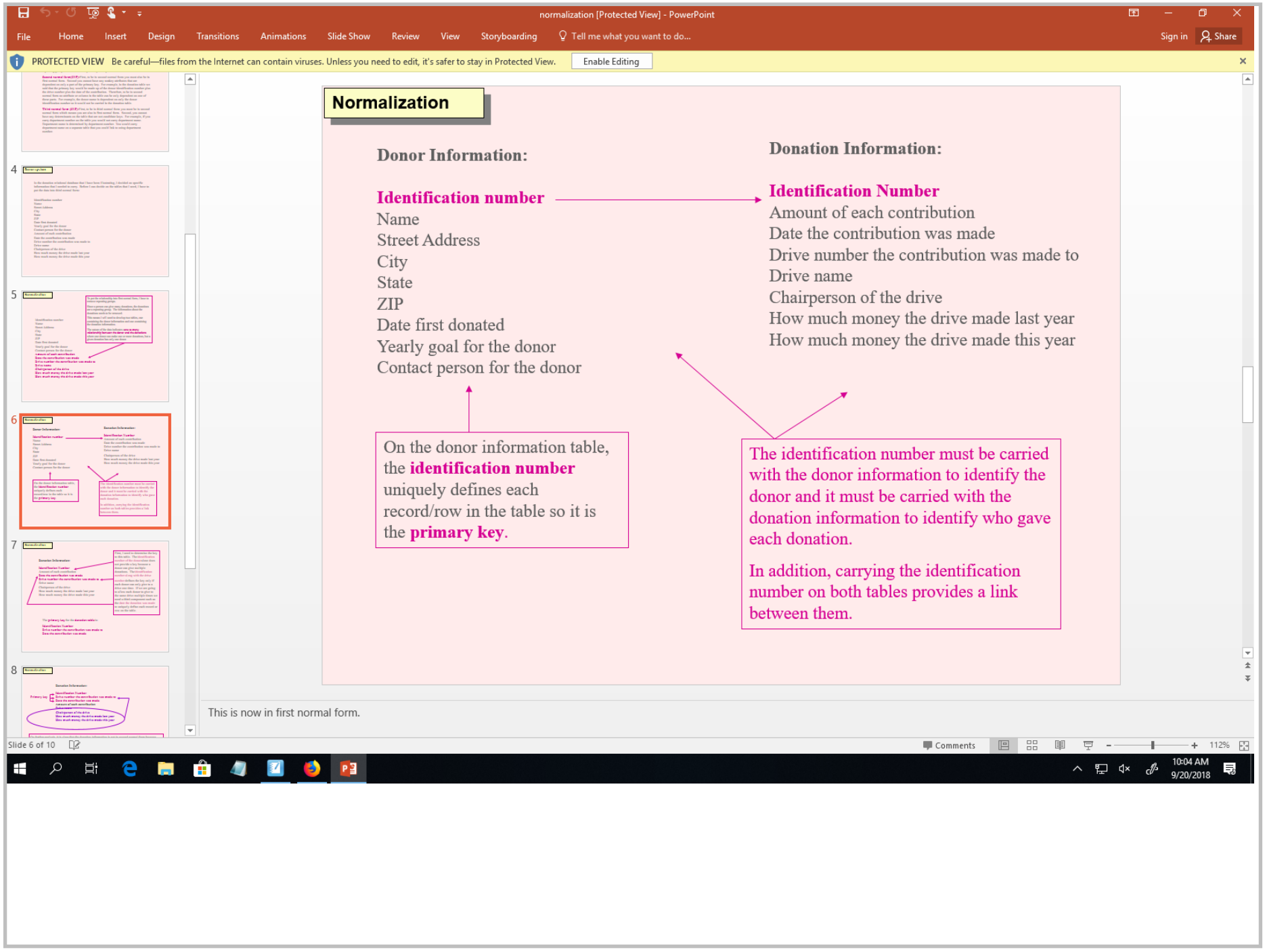
Title: Sep 20-9:59 AM (3 of 23)

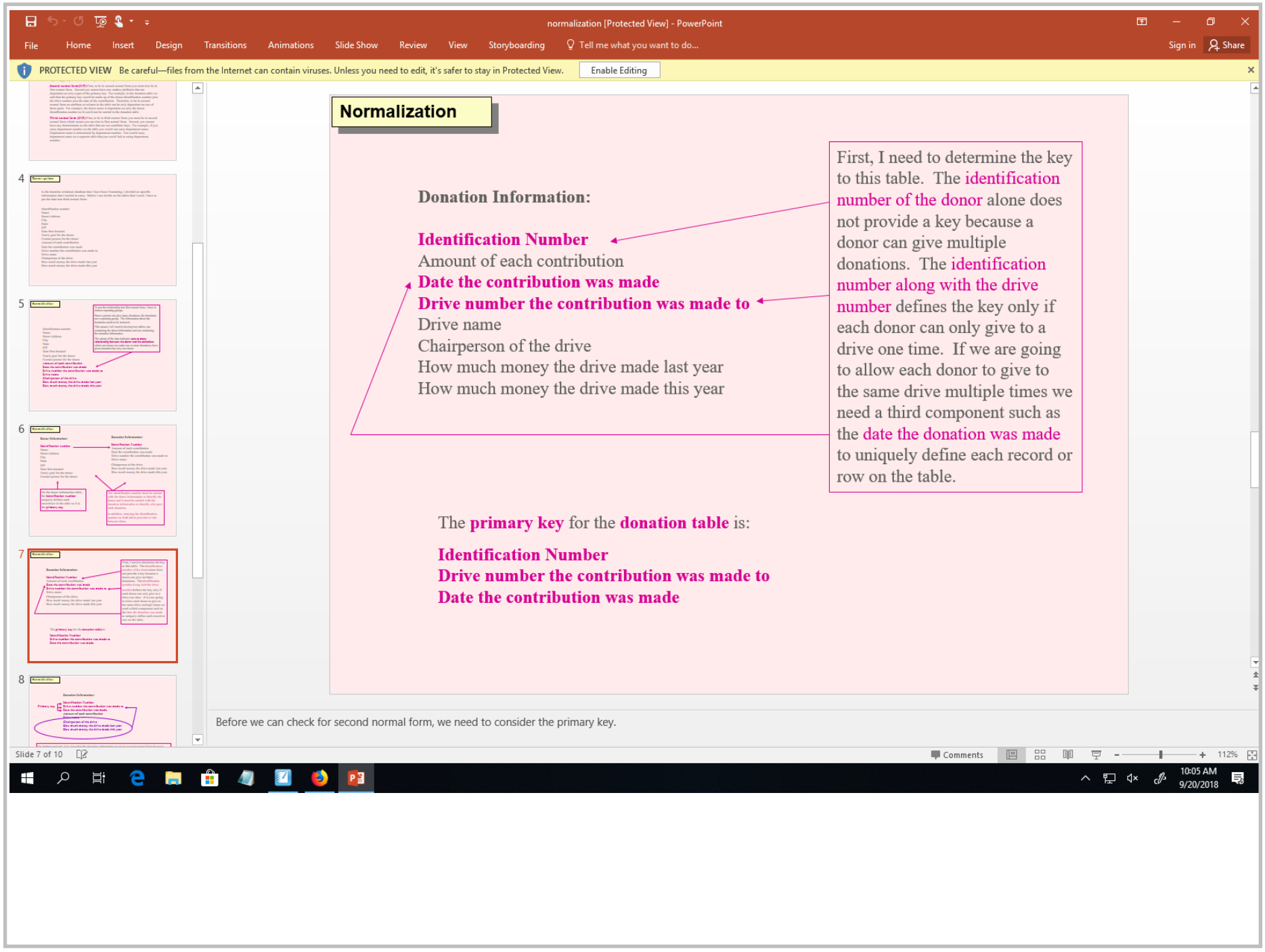


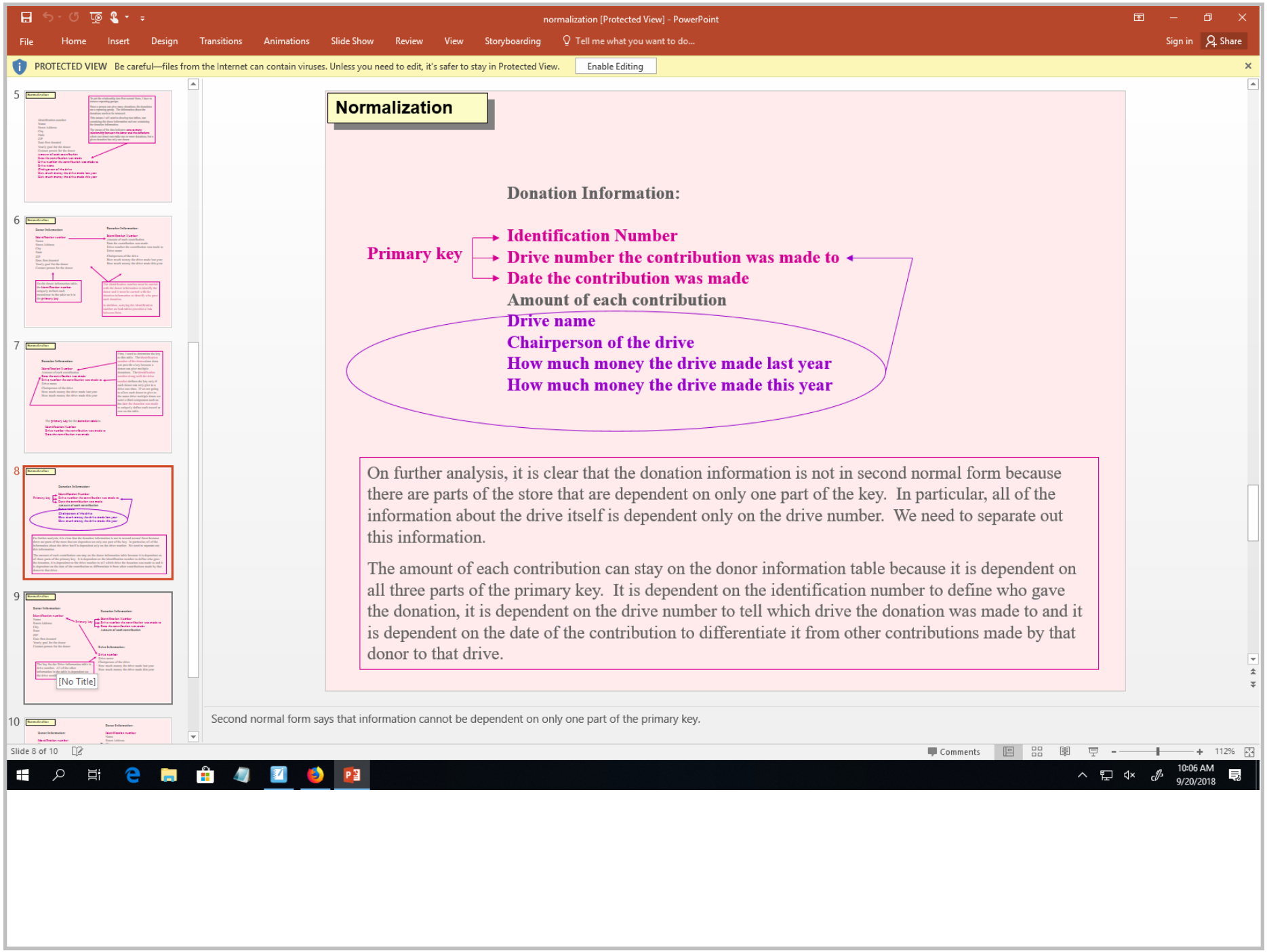
Title: Sep 20-10:03 AM (4 of 23)



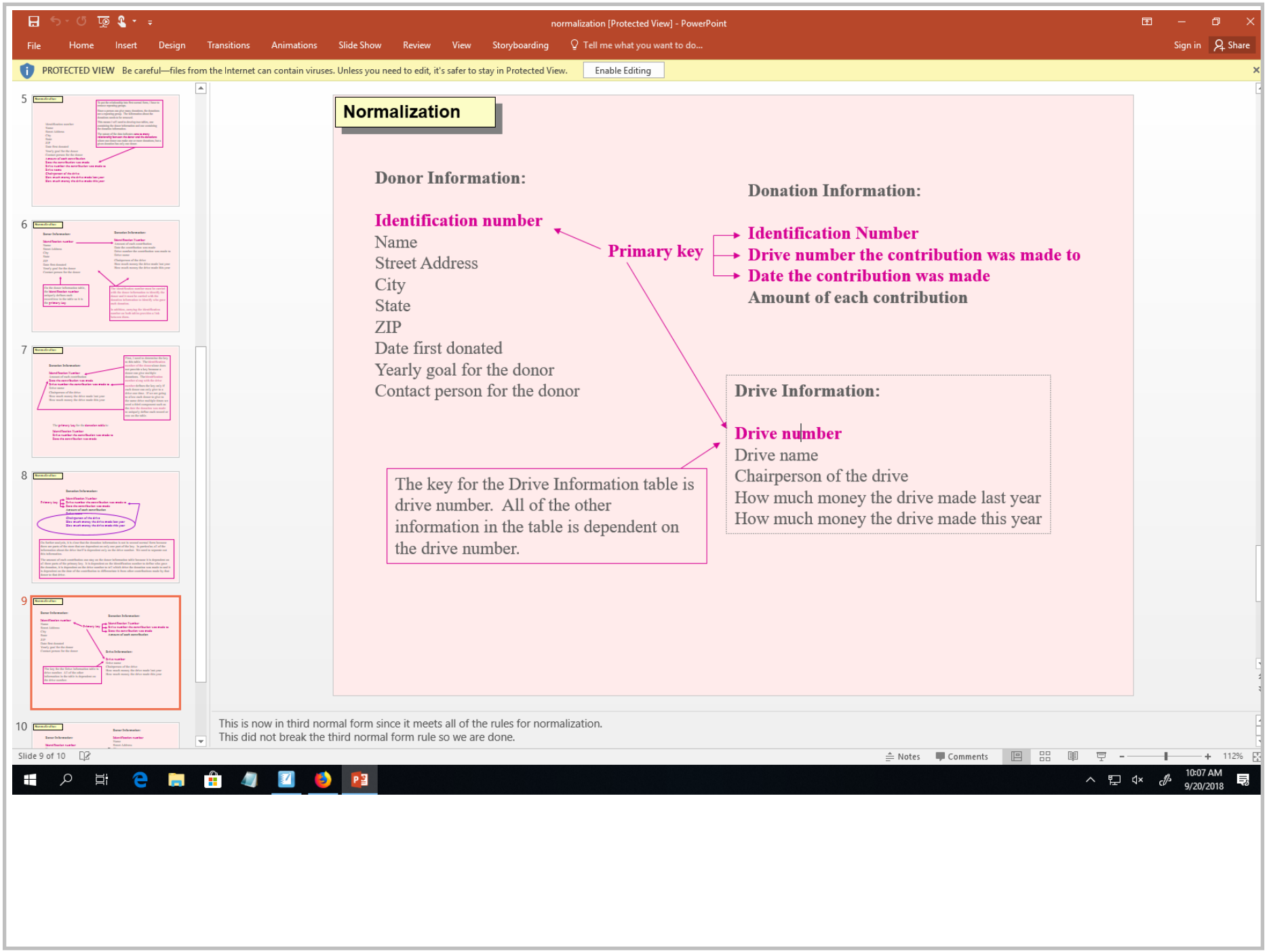
Title: Sep 20-10:03 AM (5 of 23)


normalization [Protected View] - PowerPoint
File Home Insert Design Transitions Animations Slide Show Review View Storyboarding Tell me what you want to do...
Sign in Share
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Normalization
Donor Information:
Identification number
Name
Street Address
City
State
ZIP
Date first donated
Yearly goal for the donor
Contact person for the donor
Donation Information:
Identification Number
Amount of each contribution
Date the contribution was made
Drive number the contribution was made to
Drive name
Chairperson of the drive
How much money the drive made last year
How much money the drive made this year
On the donor information table, the identification number uniquely defines each record/row in the table so it is the primary key.
The identification number must be carried with the donor information to identify the donor and it must be carried with the donation information to identify who gave each donation.
In addition, carrying the identification number on both tables provides a link between them.
This is now in first normal form.
Slide 6 of 10
Comments
112%
10:04 AM
9/20/2018

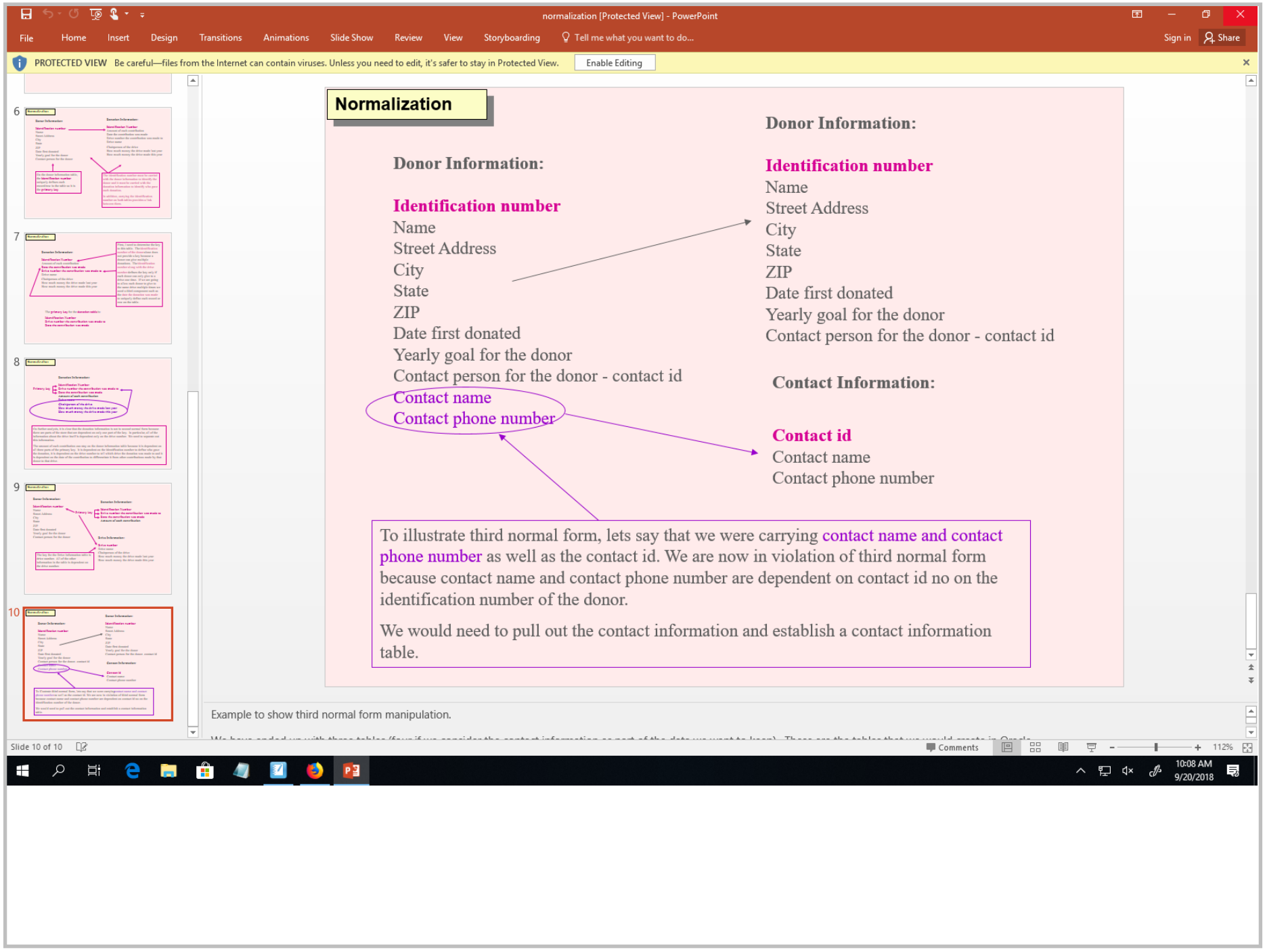

normalization [Protected View] - PowerPoint
File Home Insert Design Transitions Animations Slide Show Review View Storyboarding Tell me what you want to do...
Sign in Share
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing
Normalization
Donation Information:
Identification Number
Amount of each contribution
Date the contribution was made
Drive number the contribution was made to
Drive name
Chairperson of the drive
How much money the drive made last year
How much money the drive made this year
First, I need to determine the key to this table. The identification number of the donor alone does not provide a key because a donor can give multiple donations. The identification number along with the drive number defines the key only if each donor can only give to a drive one time. If we are going to allow each donor to give to the same drive multiple times we need a third component such as the date the donation was made to uniquely define each record or row on the table.
The primary key for the donation table is:
Identification Number
Drive number the contribution was made to
Date the contribution was made
Before we can check for second normal form, we need to consider the primary key.
Slide 7 of 10
Comments
112%
10:05 AM
9/20/2018



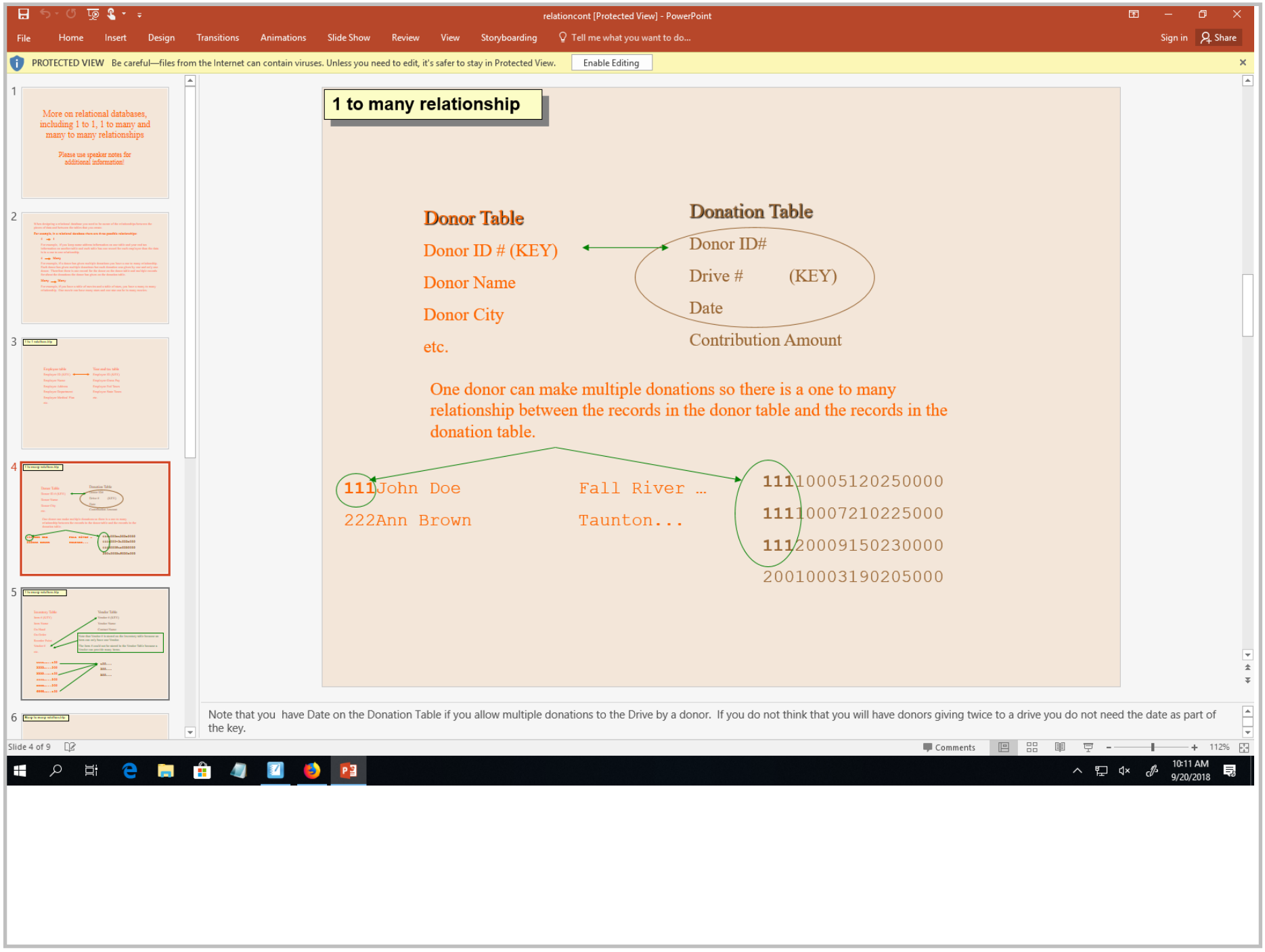
Title: Sep 20-10:06 AM (8 of 23)

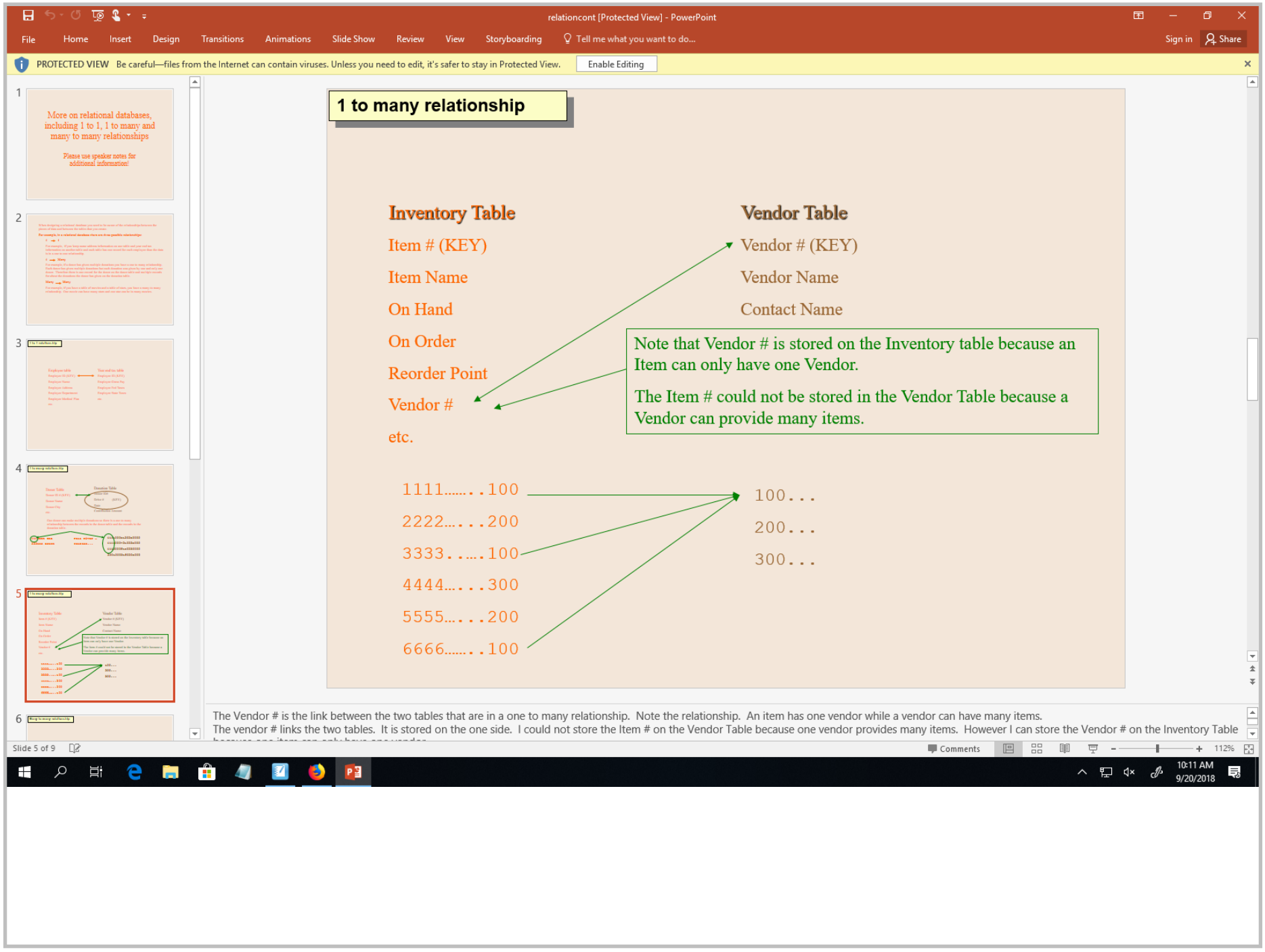


Title: Sep 20-10:07 AM (9 of 23)

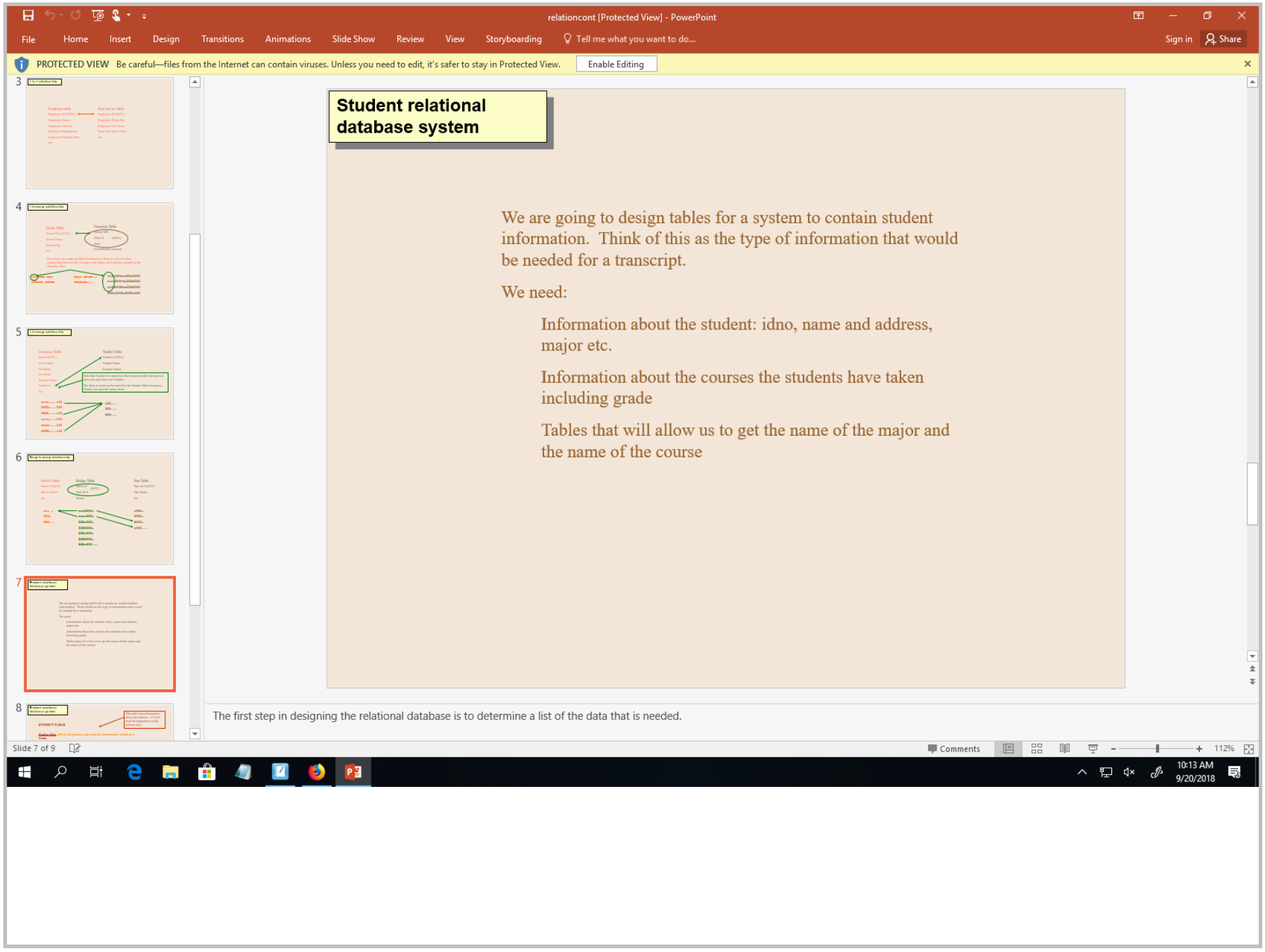


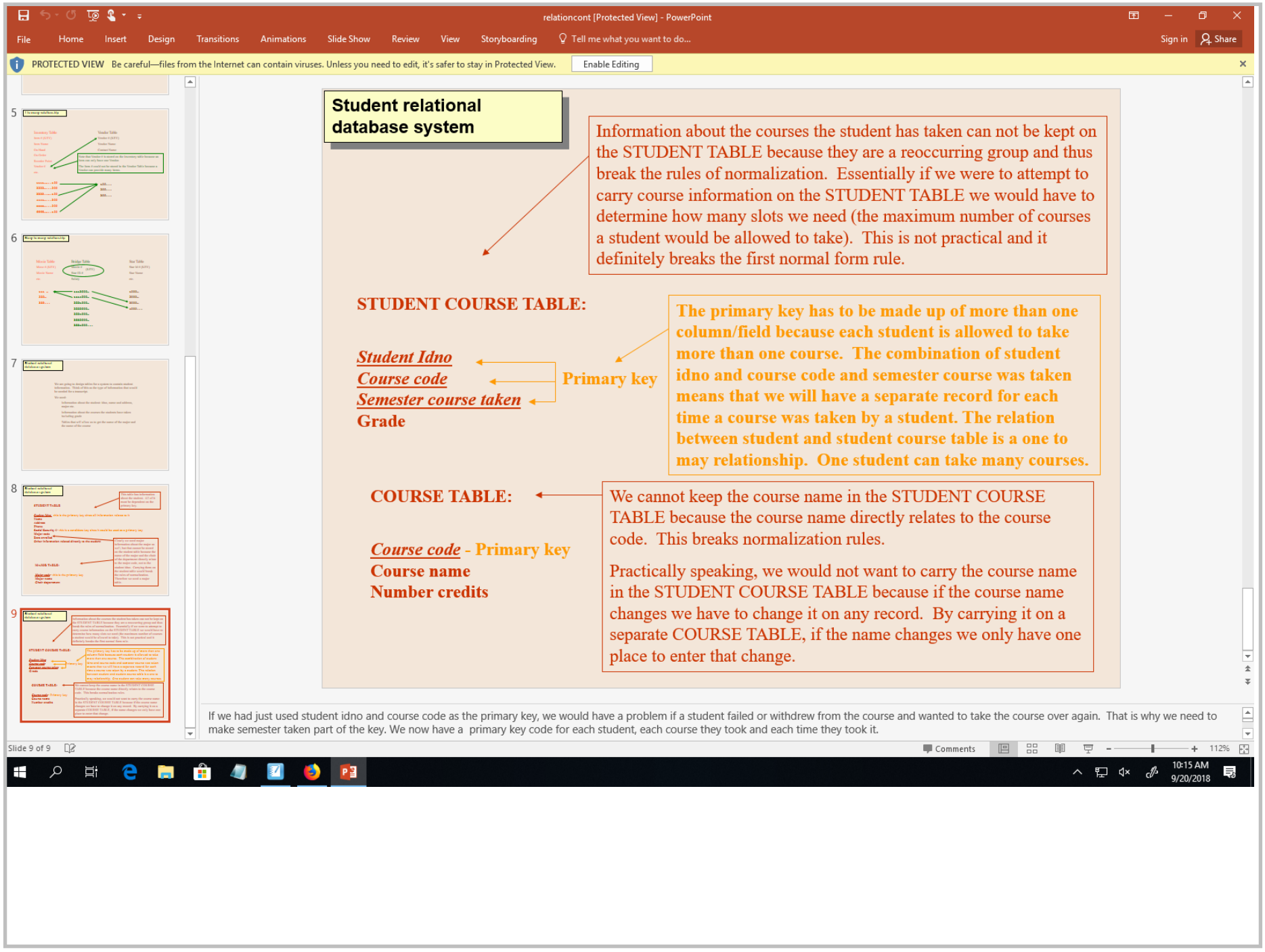
Title: Sep 20-10:08 AM (10 of 23)

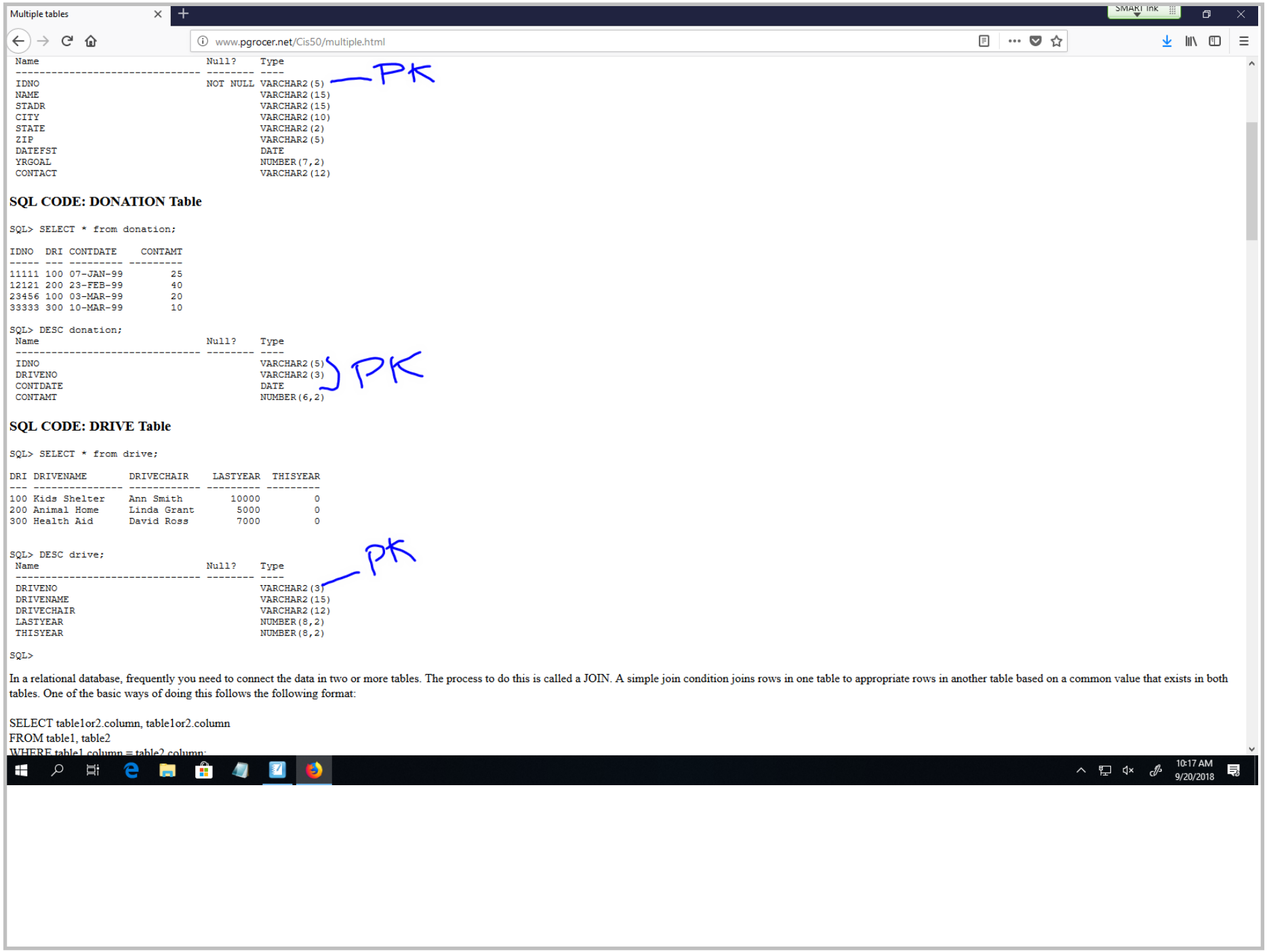

relationcont [Protected View] - PowerPoint
File Home Insert Design Transitions Animations Slide Show Review View Storyboarding Tell me what you want to do...
Sign in Share
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
1 to many relationship
Donor Table
Donor ID # (KEY)
Donor Name
Donor City
etc.
Donation Table
Donor ID#
Drive # (KEY)
Date
Contribution Amount
One donor can make multiple donations so there is a one to many relationship between the records in the donor table and the records in the donation table.
111John Doe
Fall River …
222Ann Brown
Taunton...
111 10005120250000
111 10007210225000
111 20009150230000
20010003190205000
Note that you have Date on the Donation Table if you allow multiple donations to the Drive by a donor. If you do not think that you will have donors giving twice to a drive you do not need the date as part of the key.
Slide 4 of 9
Comments
112%
10:11 AM
9/20/2018



Title: Sep 20-10:11 AM (12 of 23)


relationcont [Protected View] - PowerPoint
File Home Insert Design Transitions Animations Slide Show Review View Storyboarding Tell me what you want to do...
Sign in Share
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Student relational database system
We are going to design tables for a system to contain student information. Think of this as the type of information that would be needed for a transcript.
We need:
Information about the student: idno, name and address, major etc.
Information about the courses the students have taken including grade
Tables that will allow us to get the name of the major and the name of the course
The first step in designing the relational database is to determine a list of the data that is needed.
Slide 7 of 9
Comments
112%
10:13 AM
9/20/2018


relationcont [Protected View] - PowerPoint
File Home Insert Design Transitions Animations Slide Show Review View Storyboarding Tell me what you want to do...
Sign in Share
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing
Student relational database system
Information about the courses the student has taken can not be kept on the STUDENT TABLE because they are a reoccurring group and thus break the rules of normalization. Essentially if we were to attempt to carry course information on the STUDENT TABLE we would have to determine how many slots we need (the maximum number of courses a student would be allowed to take). This is not practical and it definitely breaks the first normal form rule.
STUDENT COURSE TABLE:
Student Idno
Course code
Semester course taken
Primary key
Grade
The primary key has to be made up of more than one column/field because each student is allowed to take more than one course. The combination of student idno and course code and semester course was taken means that we will have a separate record for each time a course was taken by a student. The relation between student and student course table is a one to may relationship. One student can take many courses.
COURSE TABLE:
We cannot keep the course name in the STUDENT COURSE TABLE because the course name directly relates to the course code. This breaks normalization rules.
Course code - Primary key
Course name
Number credits
Practically speaking, we would not want to carry the course name in the STUDENT COURSE TABLE because if the course name changes we have to change it on any record. By carrying it on a separate COURSE TABLE, if the name changes we only have one place to enter that change.
If we had just used student idno and course code as the primary key, we would have a problem if a student failed or withdrew from the course and wanted to take the course over again. That is why we need to make semester taken part of the key. We now have a primary key code for each student, each course they took and each time they took it.
Slide 9 of 9
Comments
112%
10:15 AM
9/20/2018




```
 Name                           Null?    Type
 ------------------------------ -------- ----
 IDNO                           NOT NULL VARCHAR2(5)
 NAME                                    VARCHAR2(15)
 STADR                                   VARCHAR2(15)
 CITY                                    VARCHAR2(10)
 STATE                                   VARCHAR2(2)
 ZIP                                     VARCHAR2(5)
 DATEFST                                 DATE
 YRGOAL                                  NUMBER(7,2)
 CONTACT                                 VARCHAR2(12)
```

SQL CODE: DONATION Table

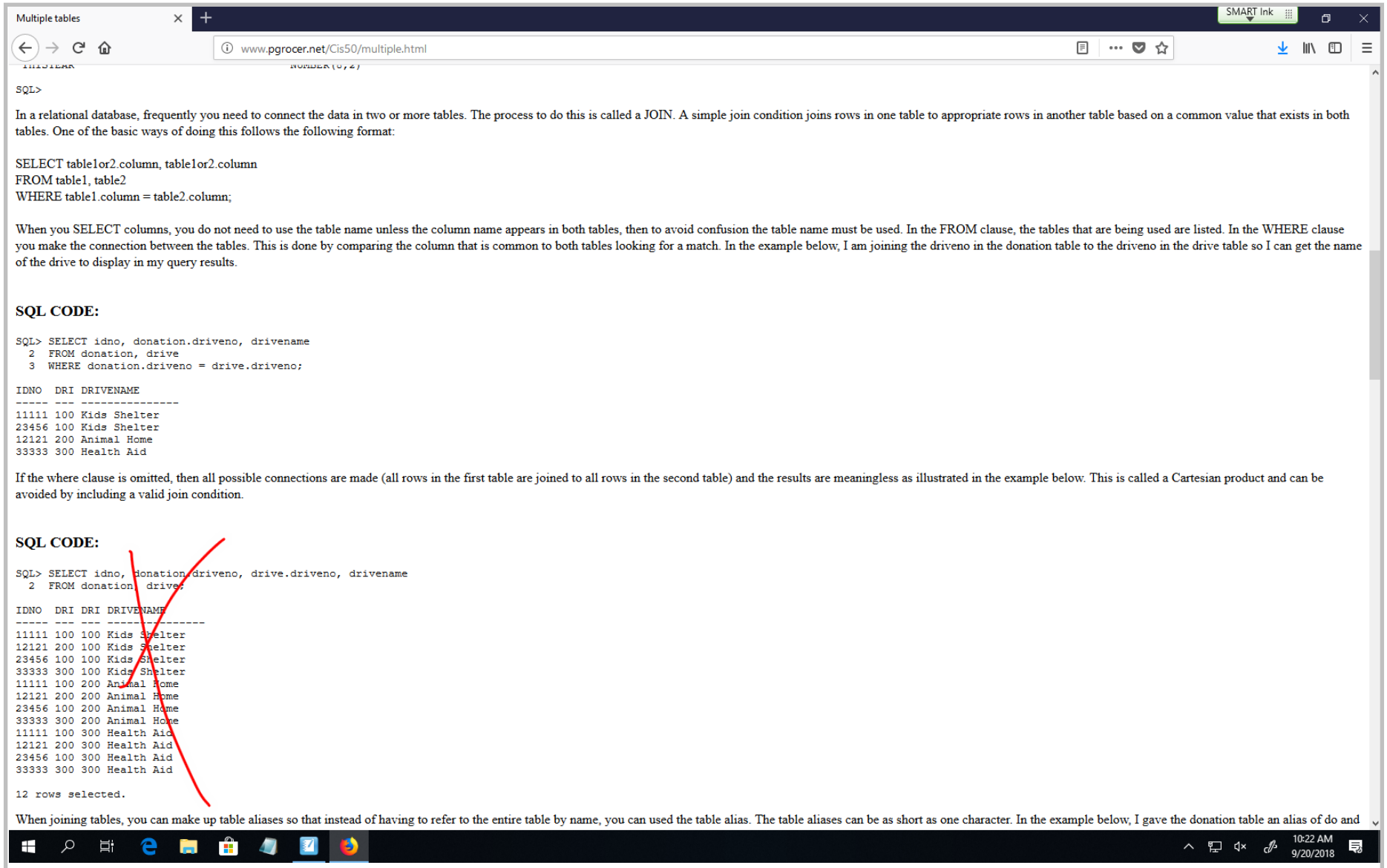
```
SQL> SELECT * from donation;

IDNO  DRI CONTDATE    CONTAMT
----- --- --------- ---------
11111 100 07-JAN-99        25
12121 200 23-FEB-99        40
23456 100 03-MAR-99        20
33333 300 10-MAR-99        10

SQL> DESC donation;
 Name                           Null?    Type
 ------------------------------ -------- ----
 IDNO                                    VARCHAR2(5)
 DRIVENO                                 VARCHAR2(3)
 CONTDATE                                DATE
 CONTAMT                                 NUMBER(6,2)
```

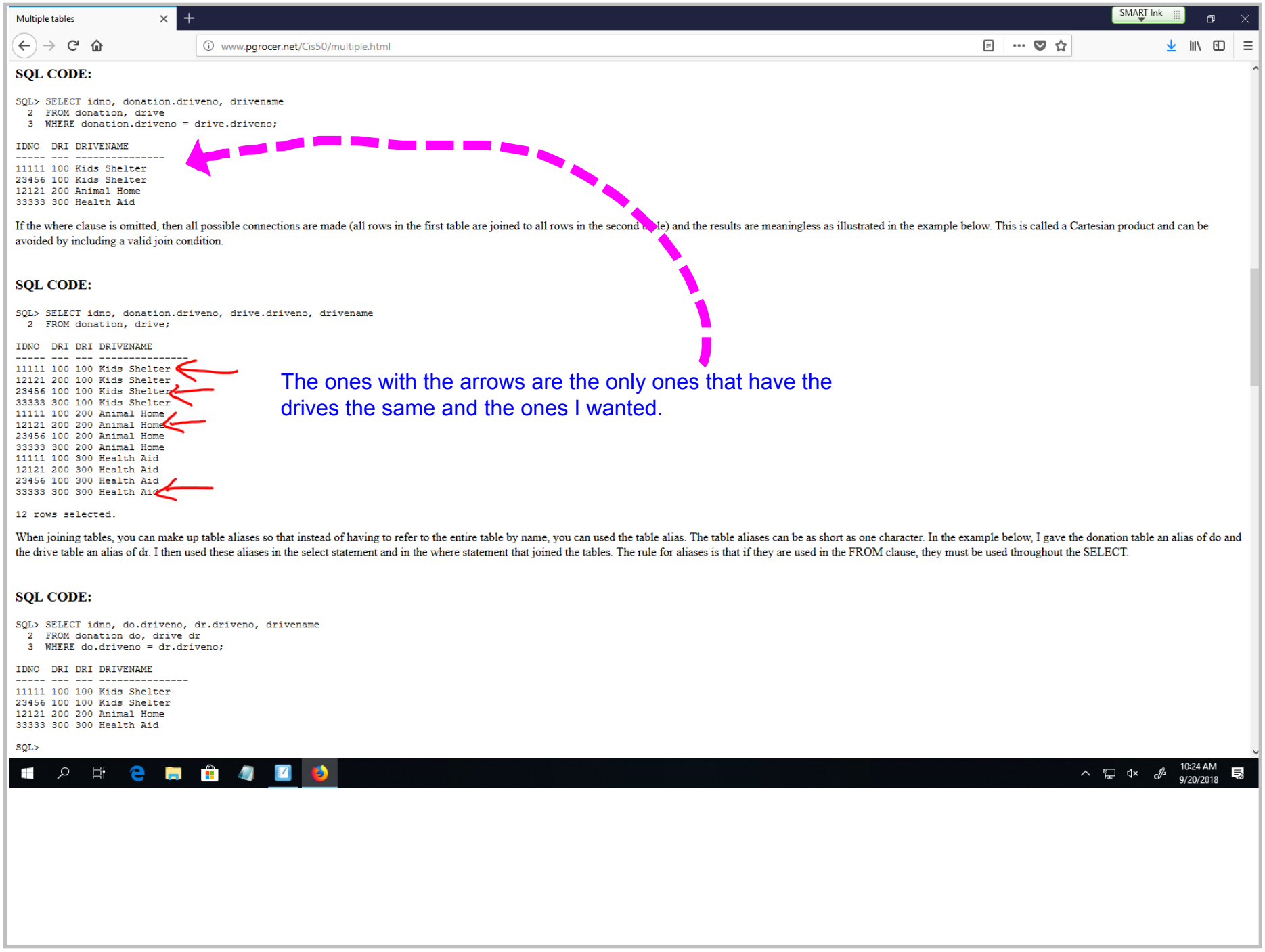
SQL CODE: DRIVE Table

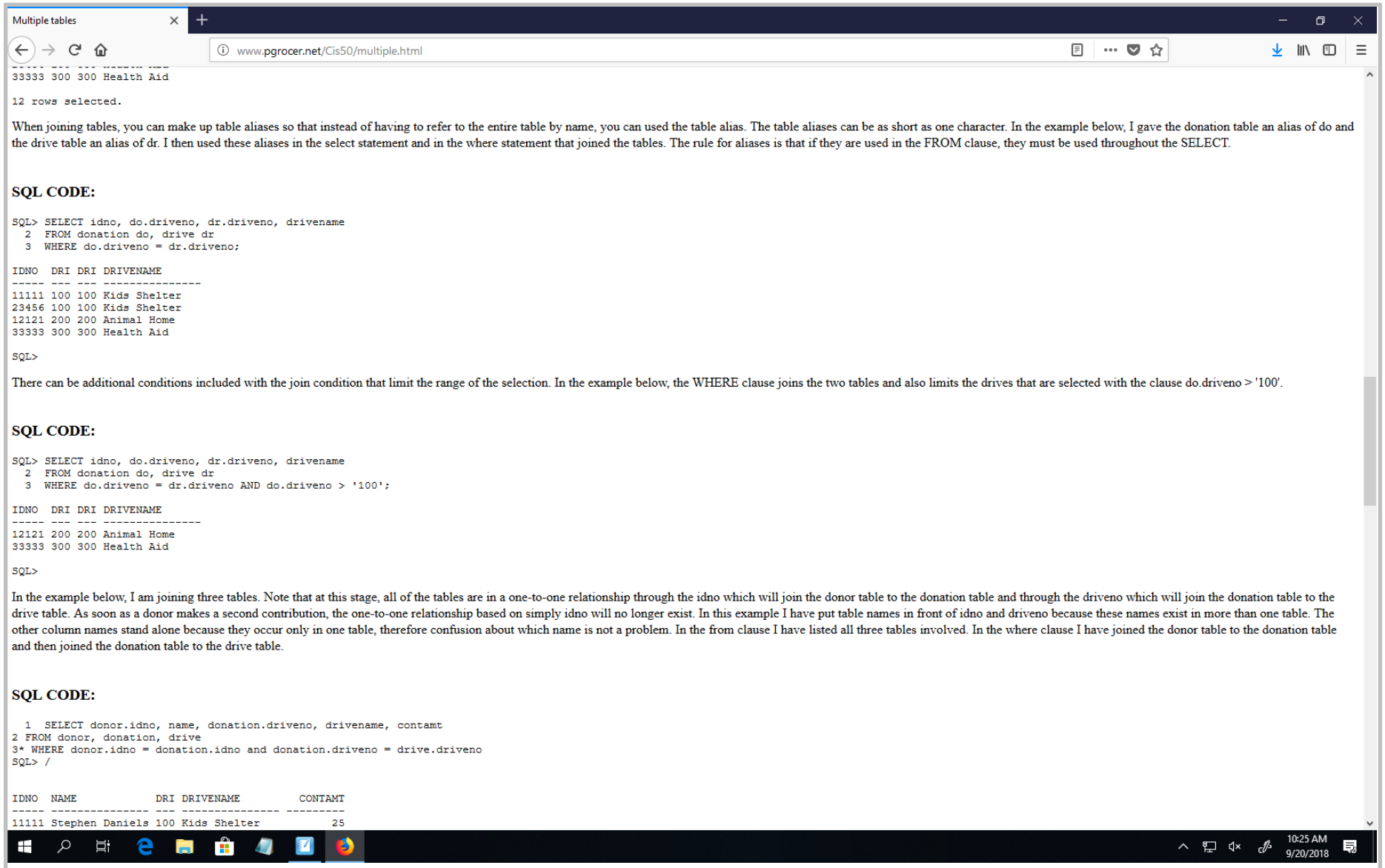
```
SQL> SELECT * from drive;

DRI DRIVENAME       DRIVECHAIR    LASTYEAR  THISYEAR
--- --------------- ------------ --------- ---------
100 Kids Shelter    Ann Smith        10000         0
200 Animal Home     Linda Grant       5000         0
300 Health Aid      David Ross        7000         0


SQL> DESC drive;
 Name                           Null?    Type
 ------------------------------ -------- ----
 DRIVENO                                 VARCHAR2(3)
 DRIVENAME                               VARCHAR2(15)
 DRIVECHAIR                              VARCHAR2(12)
 LASTYEAR                                NUMBER(8,2)
 THISYEAR                                NUMBER(8,2)

SQL>
```

In a relational database, frequently you need to connect the data in two or more tables. The process to do this is called a JOIN. A simple join condition joins rows in one table to appropriate rows in another table based on a common value that exists in both tables. One of the basic ways of doing this follows the following format:

SELECT table1or2.column, table1or2.column
FROM table1, table2
WHERE table1.column = table2.column;


Multiple tables
SMART Ink
www.pgrocer.net/Cis50/multiple.html
SQL>
In a relational database, frequently you need to connect the data in two or more tables. The process to do this is called a JOIN. A simple join condition joins rows in one table to appropriate rows in another table based on a common value that exists in both tables. One of the basic ways of doing this follows the following format:
SELECT table1or2.column, table1or2.column
FROM table1, table2
WHERE table1.column = table2.column;
When you SELECT columns, you do not need to use the table name unless the column name appears in both tables, then to avoid confusion the table name must be used. In the FROM clause, the tables that are being used are listed. In the WHERE clause you make the connection between the tables. This is done by comparing the column that is common to both tables looking for a match. In the example below, I am joining the driveno in the donation table to the driveno in the drive table so I can get the name of the drive to display in my query results.
SQL CODE:
SQL> SELECT idno, donation.driveno, drivename
2 FROM donation, drive
3 WHERE donation.driveno = drive.driveno;
IDNO DRI DRIVENAME
11111 100 Kids Shelter
23456 100 Kids Shelter
12121 200 Animal Home
33333 300 Health Aid
If the where clause is omitted, then all possible connections are made (all rows in the first table are joined to all rows in the second table) and the results are meaningless as illustrated in the example below. This is called a Cartesian product and can be avoided by including a valid join condition.
SQL CODE:
SQL> SELECT idno, donation.driveno, drive.driveno, drivename
2 FROM donation, drive;
IDNO DRI DRI DRIVENAME
11111 100 100 Kids Shelter
12121 200 100 Kids Shelter
23456 100 100 Kids Shelter
33333 300 100 Kids Shelter
11111 100 200 Animal Home
12121 200 200 Animal Home
23456 100 200 Animal Home
33333 300 200 Animal Home
11111 100 300 Health Aid
12121 200 300 Health Aid
23456 100 300 Health Aid
33333 300 300 Health Aid
12 rows selected.
When joining tables, you can make up table aliases so that instead of having to refer to the entire table by name, you can used the table alias. The table aliases can be as short as one character. In the example below, I gave the donation table an alias of do and
10:22 AM
9/20/2018


Multiple tables
SMART Ink
www.pgrocer.net/Cis50/multiple.html
SQL CODE:
SQL> SELECT idno, donation.driveno, drivename
2 FROM donation, drive
3 WHERE donation.driveno = drive.driveno;
IDNO DRI DRIVENAME
11111 100 Kids Shelter
23456 100 Kids Shelter
12121 200 Animal Home
33333 300 Health Aid
If the where clause is omitted, then all possible connections are made (all rows in the first table are joined to all rows in the second table) and the results are meaningless as illustrated in the example below. This is called a Cartesian product and can be avoided by including a valid join condition.
SQL CODE:
SQL> SELECT idno, donation.driveno, drive.driveno, drivename
2 FROM donation, drive;
IDNO DRI DRI DRIVENAME
11111 100 100 Kids Shelter
12121 200 100 Kids Shelter
23456 100 100 Kids Shelter
33333 300 100 Kids Shelter
11111 100 200 Animal Home
12121 200 200 Animal Home
23456 100 200 Animal Home
33333 300 200 Animal Home
11111 100 300 Health Aid
12121 200 300 Health Aid
23456 100 300 Health Aid
33333 300 300 Health Aid
12 rows selected.
The ones with the arrows are the only ones that have the drives the same and the ones I wanted.
When joining tables, you can make up table aliases so that instead of having to refer to the entire table by name, you can used the table alias. The table aliases can be as short as one character. In the example below, I gave the donation table an alias of do and the drive table an alias of dr. I then used these aliases in the select statement and in the where statement that joined the tables. The rule for aliases is that if they are used in the FROM clause, they must be used throughout the SELECT.
SQL CODE:
SQL> SELECT idno, do.driveno, dr.driveno, drivename
2 FROM donation do, drive dr
3 WHERE do.driveno = dr.driveno;
IDNO DRI DRI DRIVENAME
11111 100 100 Kids Shelter
23456 100 100 Kids Shelter
12121 200 200 Animal Home
33333 300 300 Health Aid
SQL>
10:24 AM
9/20/2018


Multiple tables
www.pgrocer.net/Cis50/multiple.html
33333 300 300 Health Aid
12 rows selected.
When joining tables, you can make up table aliases so that instead of having to refer to the entire table by name, you can used the table alias. The table aliases can be as short as one character. In the example below, I gave the donation table an alias of do and the drive table an alias of dr. I then used these aliases in the select statement and in the where statement that joined the tables. The rule for aliases is that if they are used in the FROM clause, they must be used throughout the SELECT.
SQL CODE:
SQL> SELECT idno, do.driveno, dr.driveno, drivename
2 FROM donation do, drive dr
3 WHERE do.driveno = dr.driveno;
IDNO DRI DRI DRIVENAME
11111 100 100 Kids Shelter
23456 100 100 Kids Shelter
12121 200 200 Animal Home
33333 300 300 Health Aid
SQL>
There can be additional conditions included with the join condition that limit the range of the selection. In the example below, the WHERE clause joins the two tables and also limits the drives that are selected with the clause do.driveno > '100'.
SQL CODE:
SQL> SELECT idno, do.driveno, dr.driveno, drivename
2 FROM donation do, drive dr
3 WHERE do.driveno = dr.driveno AND do.driveno > '100';
IDNO DRI DRI DRIVENAME
12121 200 200 Animal Home
33333 300 300 Health Aid
SQL>
In the example below, I am joining three tables. Note that at this stage, all of the tables are in a one-to-one relationship through the idno which will join the donor table to the donation table and through the driveno which will join the donation table to the drive table. As soon as a donor makes a second contribution, the one-to-one relationship based on simply idno will no longer exist. In this example I have put table names in front of idno and driveno because these names exist in more than one table. The other column names stand alone because they occur only in one table, therefore confusion about which name is not a problem. In the from clause I have listed all three tables involved. In the where clause I have joined the donor table to the donation table and then joined the donation table to the drive table.
SQL CODE:
1 SELECT donor.idno, name, donation.driveno, drivename, contamt
2 FROM donor, donation, drive
3* WHERE donor.idno = donation.idno and donation.driveno = drive.driveno
SQL> /
IDNO NAME DRI DRIVENAME CONTAMT
11111 Stephen Daniels 100 Kids Shelter 25
10:25 AM
9/20/2018

Multiple tables

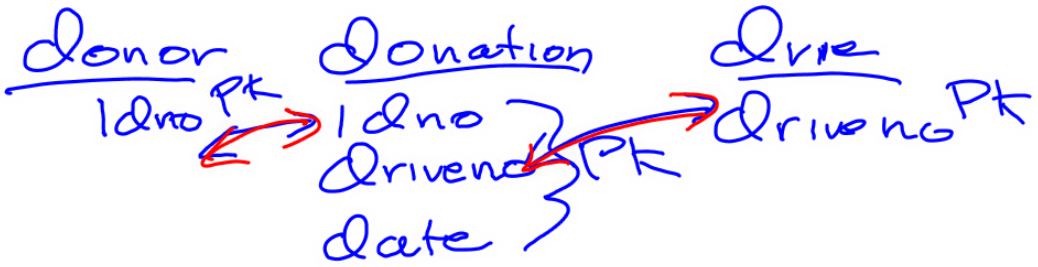
www.pgrocer.net/Cis50/multiple.html

drive table. As soon as a donor makes a second contribution, the one-to-one relationship based on simply idno will no longer exist. In this example I have put table names in front of idno and driveno because these names exist in more than one table. The other column names stand alone because they occur only in one table, therefore confusion about which name is not a problem. In the from clause I have listed all three tables involved. In the where clause I have joined the donor table to the donation table and then joined the donation table to the drive table.

SQL CODE:

Joining 3 tables.

```
  1  SELECT donor.idno, name, donation.driveno, drivename, contamt
2 FROM donor, donation, drive
3* WHERE donor.idno = donation.idno and donation.driveno = drive.driveno
SQL> /

IDNO  NAME            DRI DRIVENAME         CONTAMT
----- --------------- --- --------------- ---------
11111 Stephen Daniels 100 Kids Shelter           25
23456 Susan Ash       100 Kids Shelter           20
12121 Jennifer Ames   200 Animal Home            40
33333 Nancy Taylor    300 Health Aid             10
```



JOIN Explanation:

The joins that we are going to look at are equijoins, non-equijoins, outer joins and self joins.

Equijoin

An equijoin is the kind that has been illustrated so far in this handout. Other names for equijoins are simple joins or inner joins. An equijoin means that values in the tables being compared must be equal.

Non-equijoin

A non-equijoin means that there is not a direct join between two tables. The relationship might be that you relate a number on one table to a range on another. For example one table might have a student's grade and the other might have a range of 1 thru 6 for elementary, 7 thru 8 for junior high etc. You could then compare the student's grade to the low and high for each grade in the range and determine the group the student is in.

SELECT grade.name, grade.gradelevel, schools.groupname
FROM grade, schools
WHERE grade.gradelevel BETWEEN schools.lowgrade AND schools.highgrade;

Outer join

An outer join can be used to display records that do not satisfy the join condition. To do this, we put the join operator which is a plus sign in parenthesis in the where clause on the side of the join that is missing the information. The result is the creation of null rows on the side that is missing the information which are joined with the side that contains information you need to see. For this example, I added a record to the drive table that contained a drive that did not exist in the donation table because no one had given to that drive.

SQL CODE:

```
SQL> select * from drive;
```

10:28 AM
9/20/2018

Multiple tables

www.pgrocer.net/Cis50/multiple.html

The joins that we are going to look at are equijoins, non-equijoins, outer joins and self joins.

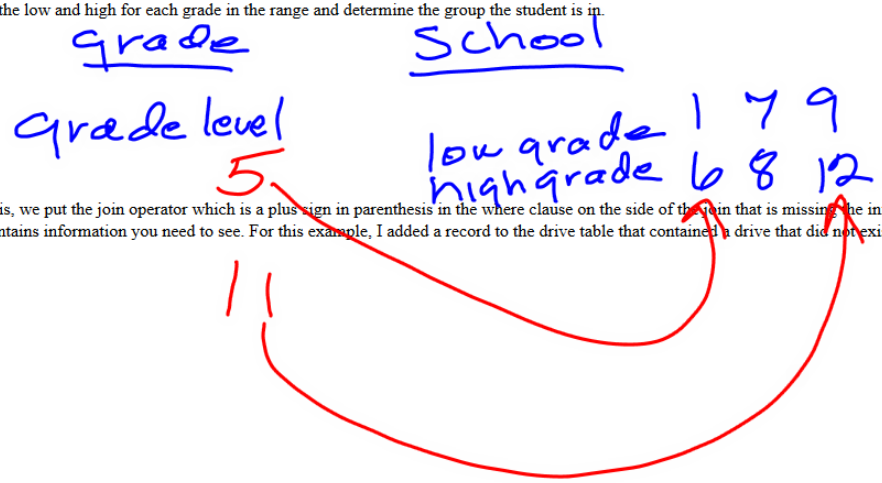
Equijoin

An equijoin is the kind that has been illustrated so far in this handout. Other names for equijoins are simple joins or inner joins. An equijoin means that values in the tables being compared must be equal.

Non-equijoin

A non-equijoin means that there is not a direct join between two tables. The relationship might be that you relate a number on one table to a range on another. For example one table might have a student's grade and the other might have a range of 1 thru 6 for elementary, 7 thru 8 for junior high etc. You could then compare the student's grade to the low and high for each grade in the range and determine the group the student is in.

```
SELECT grade.name, grade.gradelevel, schools.groupname
FROM grade, schools
WHERE grade.gradelevel BETWEEN schools.lowgrade AND schools.highgrade;
```



Outer join

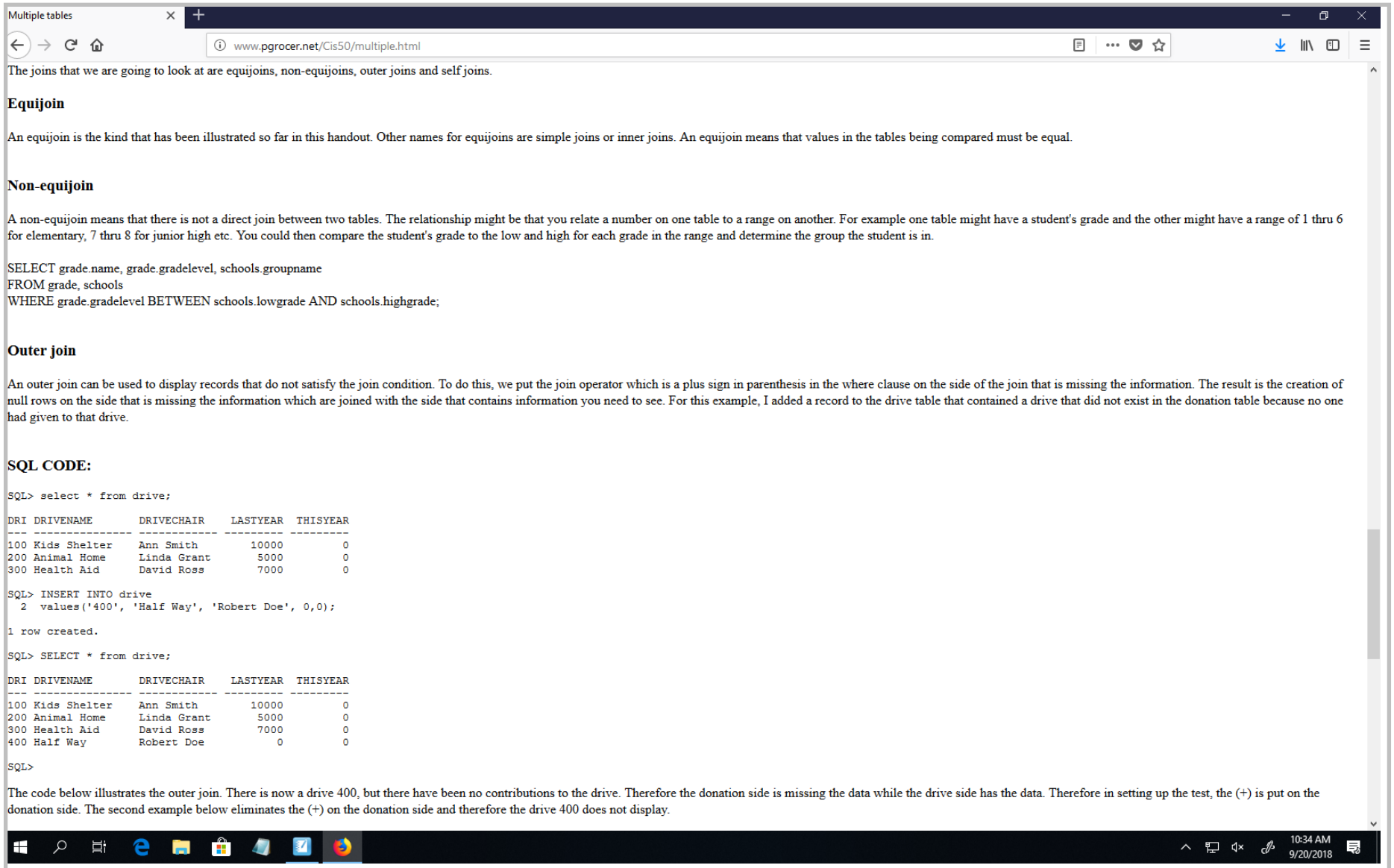
An outer join can be used to display records that do not satisfy the join condition. To do this, we put the join operator which is a plus sign in parenthesis in the where clause on the side of the join that is missing the information. The result is the creation of null rows on the side that is missing the information which are joined with the side that contains information you need to see. For this example, I added a record to the drive table that contained a drive that did not exist in the donation table because no one had given to that drive.

SQL CODE:

```
SQL> select * from drive;

DRI DRIVENAME       DRIVECHAIR    LASTYEAR  THISYEAR
--- --------------- ------------ --------- ---------
100 Kids Shelter    Ann Smith        10000         0
200 Animal Home     Linda Grant       5000         0
300 Health Aid      David Ross        7000         0

SQL> INSERT INTO drive
  2  values('400', 'Half Way', 'Robert Doe', 0,0);

1 row created.

SQL> SELECT * from drive;

DRI DRIVENAME       DRIVECHAIR    LASTYEAR  THISYEAR
--- --------------- ------------ --------- ---------
100 Kids Shelter    Ann Smith        10000         0
200 Animal Home     Linda Grant       5000         0
300 Health Aid      David Ross        7000         0
400 Half Way        Robert Doe           0         0

SQL>
```

The code below illustrates the outer join. There is now a drive 400, but there have been no contributions to the drive. Therefore the donation side is missing the data while the drive side has the data. Therefore in setting up the test, the (+) is put on the donation side. The second example below eliminates the (+) on the donation side and therefore the drive 400 does not display.

10:32 AM
9/20/2018



The joins that we are going to look at are equijoins, non-equijoins, outer joins and self joins.

Equijoin

An equijoin is the kind that has been illustrated so far in this handout. Other names for equijoins are simple joins or inner joins. An equijoin means that values in the tables being compared must be equal.

Non-equijoin

A non-equijoin means that there is not a direct join between two tables. The relationship might be that you relate a number on one table to a range on another. For example one table might have a student's grade and the other might have a range of 1 thru 6 for elementary, 7 thru 8 for junior high etc. You could then compare the student's grade to the low and high for each grade in the range and determine the group the student is in.

```
SELECT grade.name, grade.gradelevel, schools.groupname
FROM grade, schools
WHERE grade.gradelevel BETWEEN schools.lowgrade AND schools.highgrade;
```

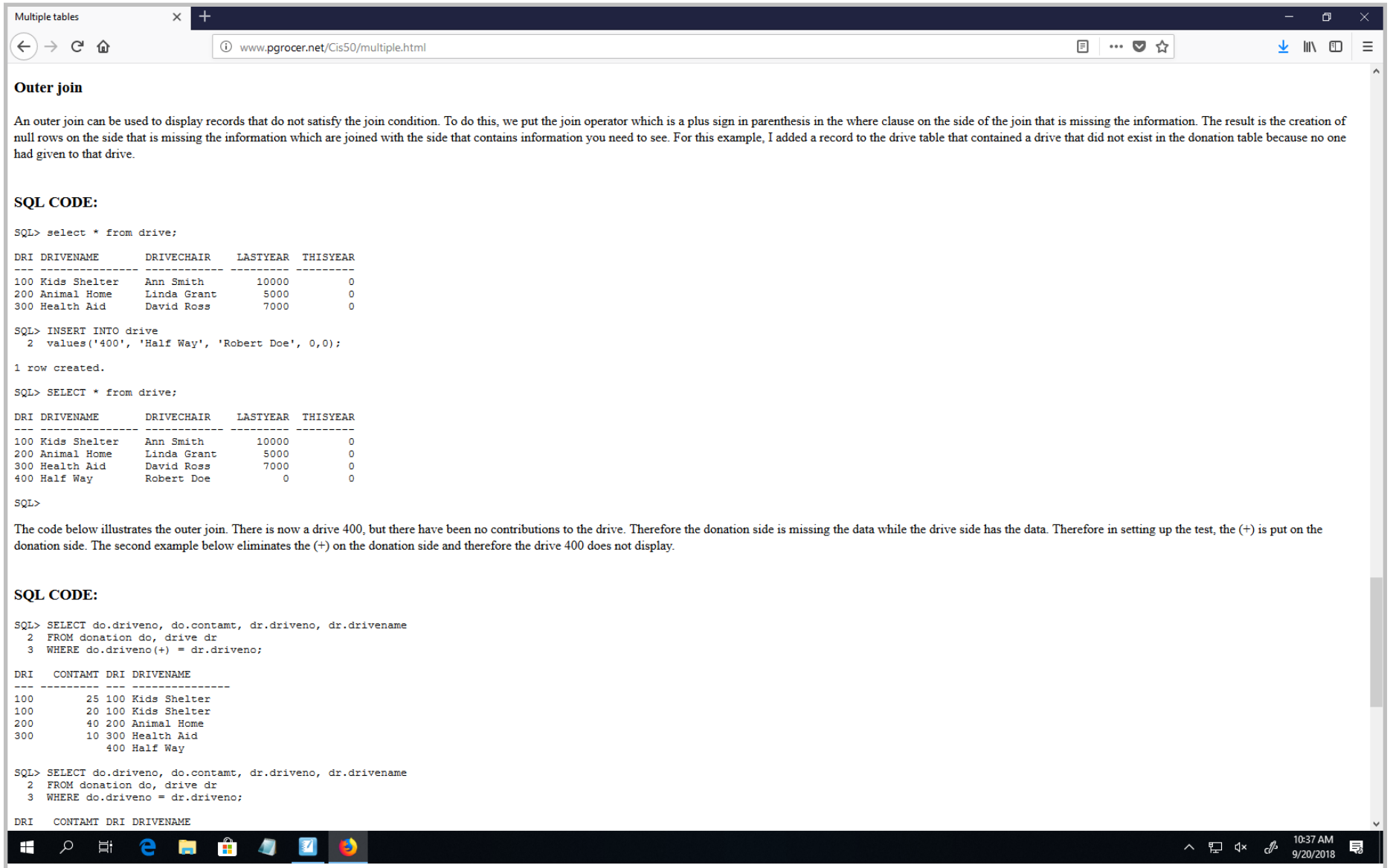
Outer join

An outer join can be used to display records that do not satisfy the join condition. To do this, we put the join operator which is a plus sign in parenthesis in the where clause on the side of the join that is missing the information. The result is the creation of null rows on the side that is missing the information which are joined with the side that contains information you need to see. For this example, I added a record to the drive table that contained a drive that did not exist in the donation table because no one had given to that drive.

SQL CODE:

```
SQL> select * from drive;

DRI DRIVENAME       DRIVECHAIR    LASTYEAR  THISYEAR
--- --------------- ------------ --------- ---------
100 Kids Shelter    Ann Smith        10000         0
200 Animal Home     Linda Grant       5000         0
300 Health Aid      David Ross        7000         0

SQL> INSERT INTO drive
  2  values('400', 'Half Way', 'Robert Doe', 0,0);

1 row created.

SQL> SELECT * from drive;

DRI DRIVENAME       DRIVECHAIR    LASTYEAR  THISYEAR
--- --------------- ------------ --------- ---------
100 Kids Shelter    Ann Smith        10000         0
200 Animal Home     Linda Grant       5000         0
300 Health Aid      David Ross        7000         0
400 Half Way        Robert Doe           0         0

SQL>
```

The code below illustrates the outer join. There is now a drive 400, but there have been no contributions to the drive. Therefore the donation side is missing the data while the drive side has the data. Therefore in setting up the test, the (+) is put on the donation side. The second example below eliminates the (+) on the donation side and therefore the drive 400 does not display.



Outer join

An outer join can be used to display records that do not satisfy the join condition. To do this, we put the join operator which is a plus sign in parenthesis in the where clause on the side of the join that is missing the information. The result is the creation of null rows on the side that is missing the information which are joined with the side that contains information you need to see. For this example, I added a record to the drive table that contained a drive that did not exist in the donation table because no one had given to that drive.

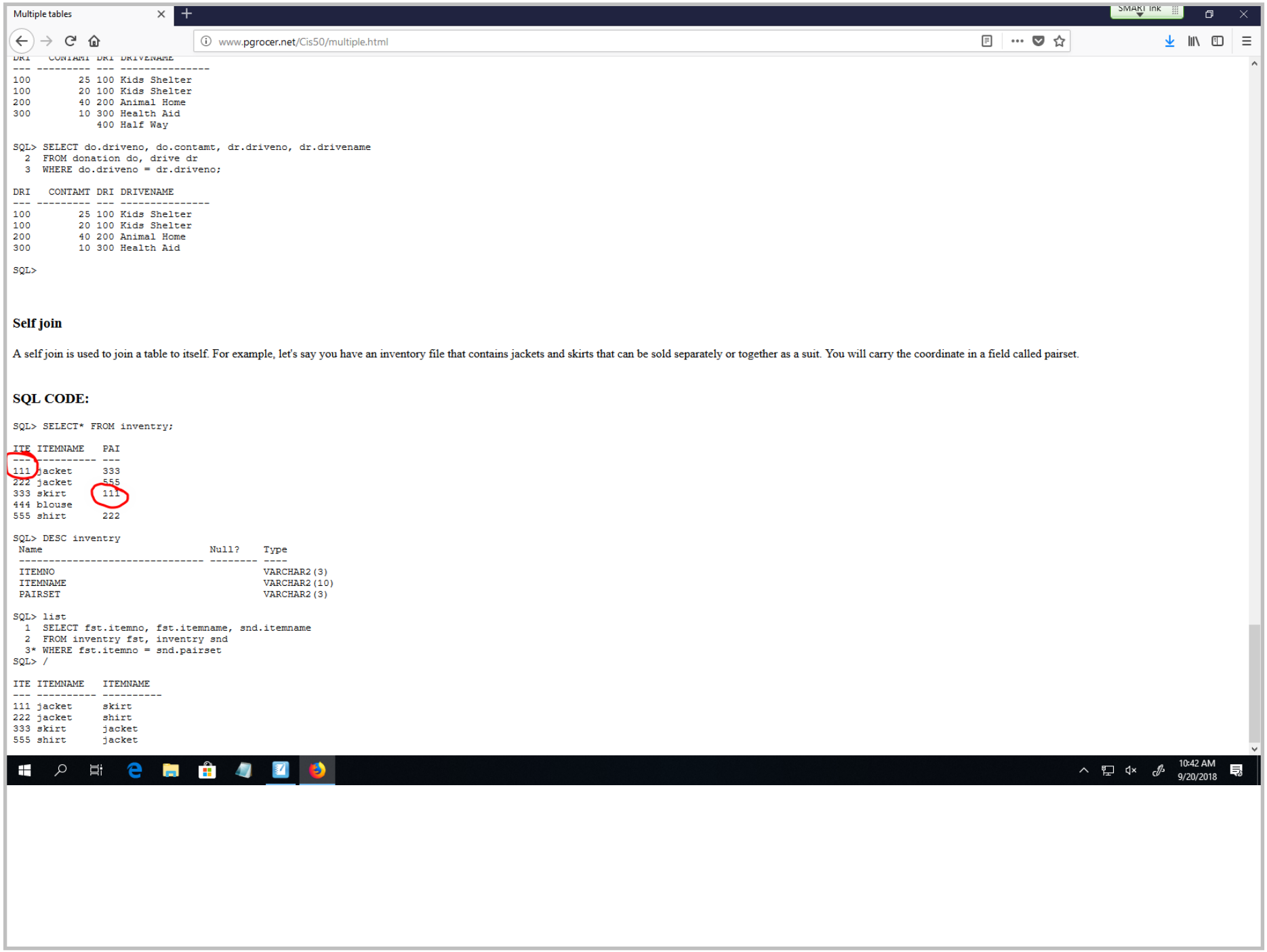
SQL CODE:

```
SQL> select * from drive;

DRI DRIVENAME       DRIVECHAIR    LASTYEAR  THISYEAR
--- --------------- ------------ --------- ---------
100 Kids Shelter    Ann Smith        10000         0
200 Animal Home     Linda Grant       5000         0
300 Health Aid      David Ross        7000         0

SQL> INSERT INTO drive
  2  values('400', 'Half Way', 'Robert Doe', 0,0);

1 row created.

SQL> SELECT * from drive;

DRI DRIVENAME       DRIVECHAIR    LASTYEAR  THISYEAR
--- --------------- ------------ --------- ---------
100 Kids Shelter    Ann Smith        10000         0
200 Animal Home     Linda Grant       5000         0
300 Health Aid      David Ross        7000         0
400 Half Way        Robert Doe           0         0

SQL>
```

The code below illustrates the outer join. There is now a drive 400, but there have been no contributions to the drive. Therefore the donation side is missing the data while the drive side has the data. Therefore in setting up the test, the (+) is put on the donation side. The second example below eliminates the (+) on the donation side and therefore the drive 400 does not display.

SQL CODE:

```
SQL> SELECT do.driveno, do.contamt, dr.driveno, dr.drivename
  2  FROM donation do, drive dr
  3  WHERE do.driveno(+) = dr.driveno;

DRI   CONTAMT DRI DRIVENAME
--- --------- --- ---------------
100        25 100 Kids Shelter
100        20 100 Kids Shelter
200        40 200 Animal Home
300        10 300 Health Aid
              400 Half Way

SQL> SELECT do.driveno, do.contamt, dr.driveno, dr.drivename
  2  FROM donation do, drive dr
  3  WHERE do.driveno = dr.driveno;

DRI   CONTAMT DRI DRIVENAME
```


Multiple tables
www.pgrocer.net/Cis50/multiple.html
DRI CONTAMT DRI DRIVENAME
--- --------- --- ---------------
100 25 100 Kids Shelter
100 20 100 Kids Shelter
200 40 200 Animal Home
300 10 300 Health Aid
400 Half Way
SQL> SELECT do.driveno, do.contamt, dr.driveno, dr.drivename
2 FROM donation do, drive dr
3 WHERE do.driveno = dr.driveno;
DRI CONTAMT DRI DRIVENAME
--- --------- --- ---------------
100 25 100 Kids Shelter
100 20 100 Kids Shelter
200 40 200 Animal Home
300 10 300 Health Aid
SQL>
Self join
A self join is used to join a table to itself. For example, let's say you have an inventory file that contains jackets and skirts that can be sold separately or together as a suit. You will carry the coordinate in a field called pairset.
SQL CODE:
SQL> SELECT* FROM inventry;
ITE ITEMNAME PAI
--- ---------- ---
111 jacket 333
222 jacket 555
333 skirt 111
444 blouse
555 shirt 222
SQL> DESC inventry
Name Null? Type
ITEMNO VARCHAR2(3)
ITEMNAME VARCHAR2(10)
PAIRSET VARCHAR2(3)
SQL> list
1 SELECT fst.itemno, fst.itemname, snd.itemname
2 FROM inventry fst, inventry snd
3* WHERE fst.itemno = snd.pairset
SQL> /
ITE ITEMNAME ITEMNAME
--- ---------- ----------
111 jacket skirt
222 jacket shirt
333 skirt jacket
555 shirt jacket
10:42 AM
9/20/2018