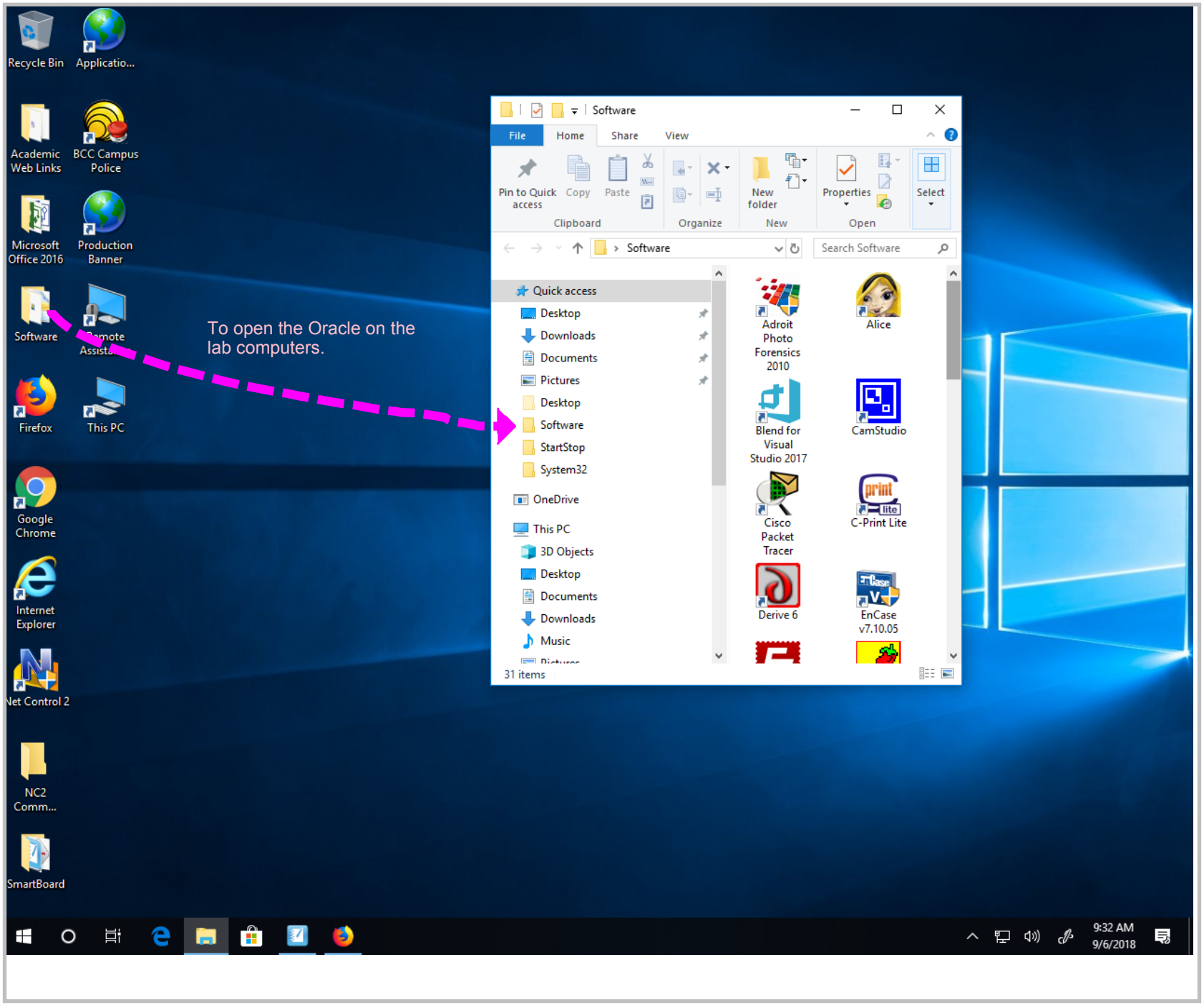
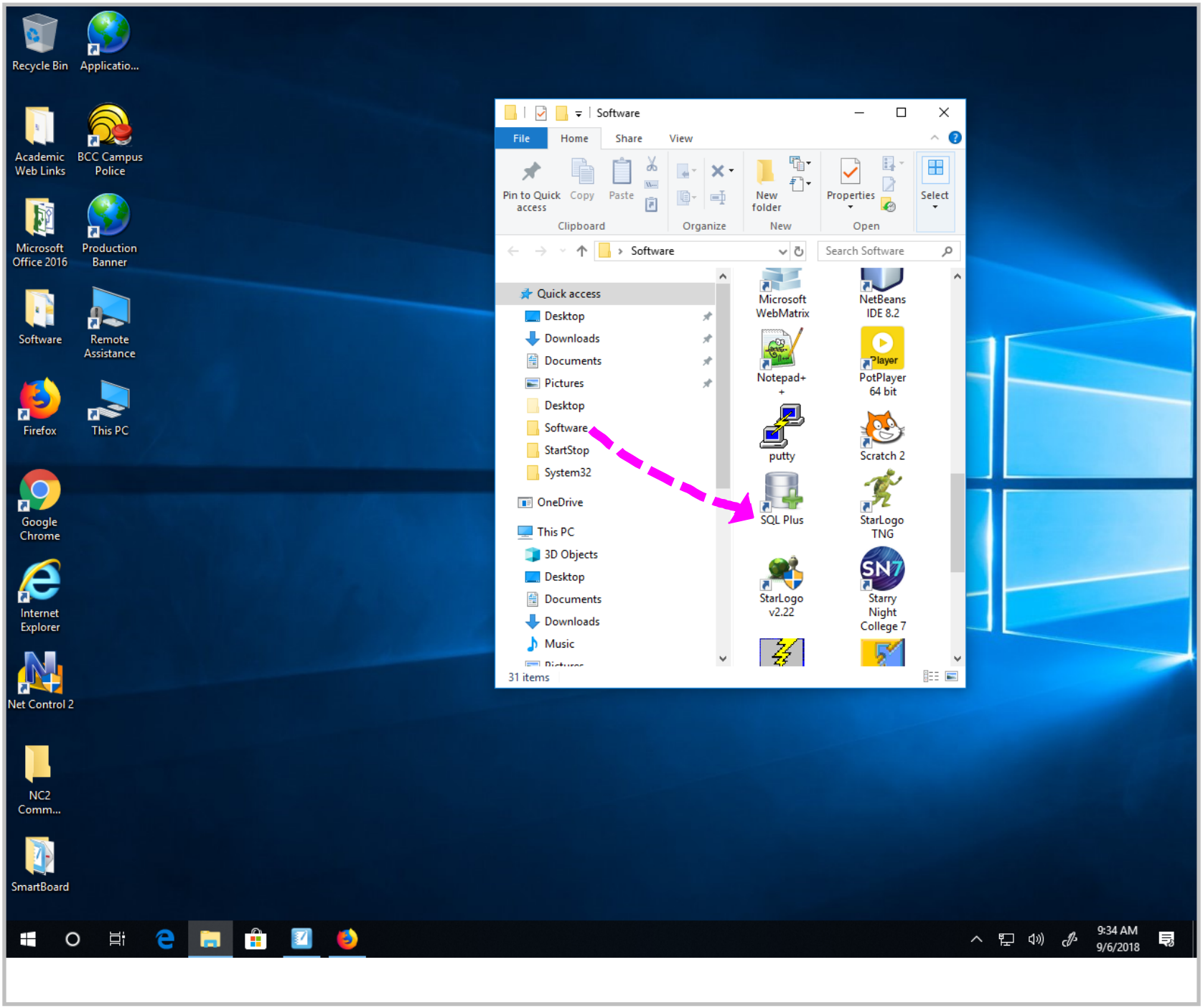
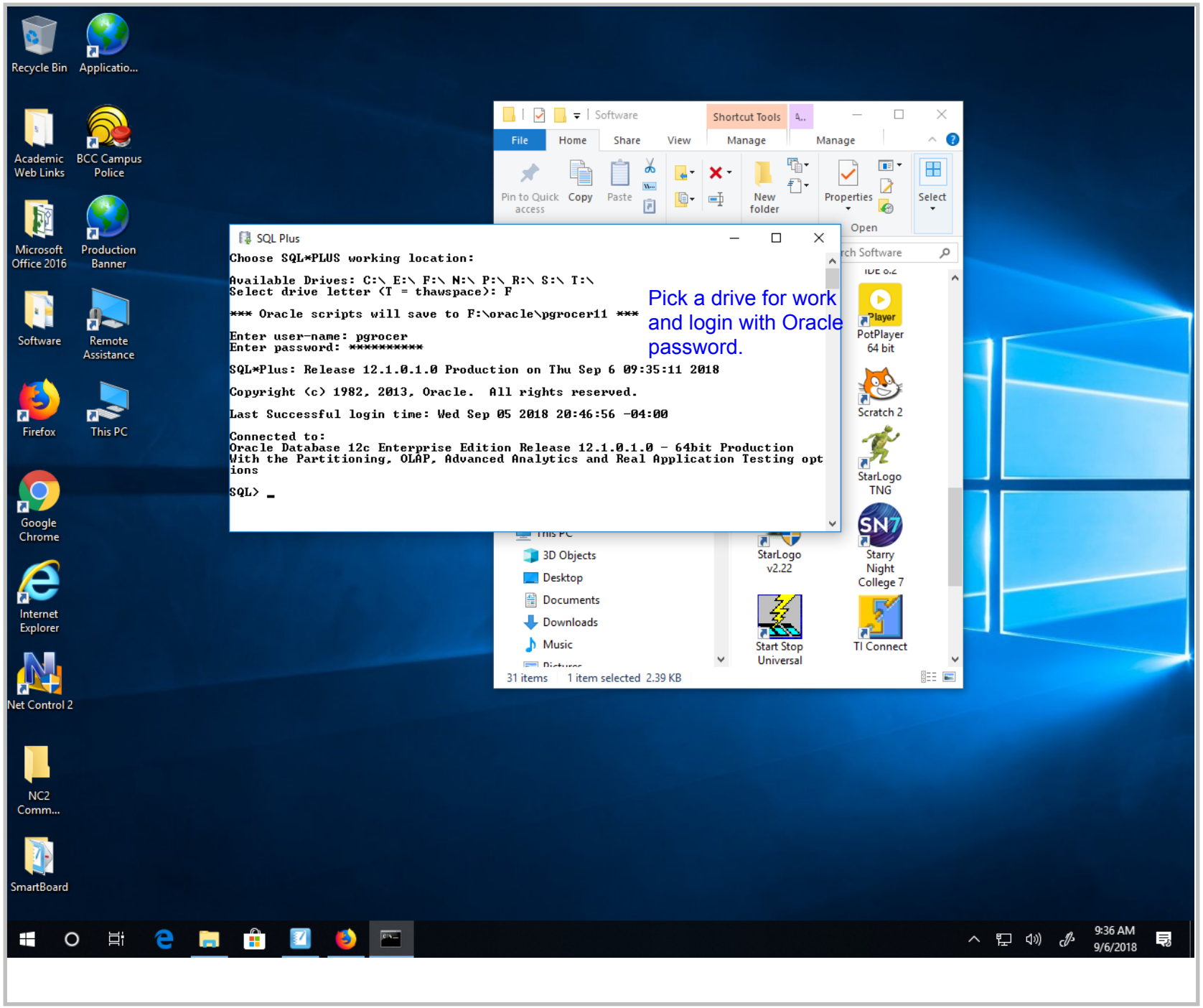


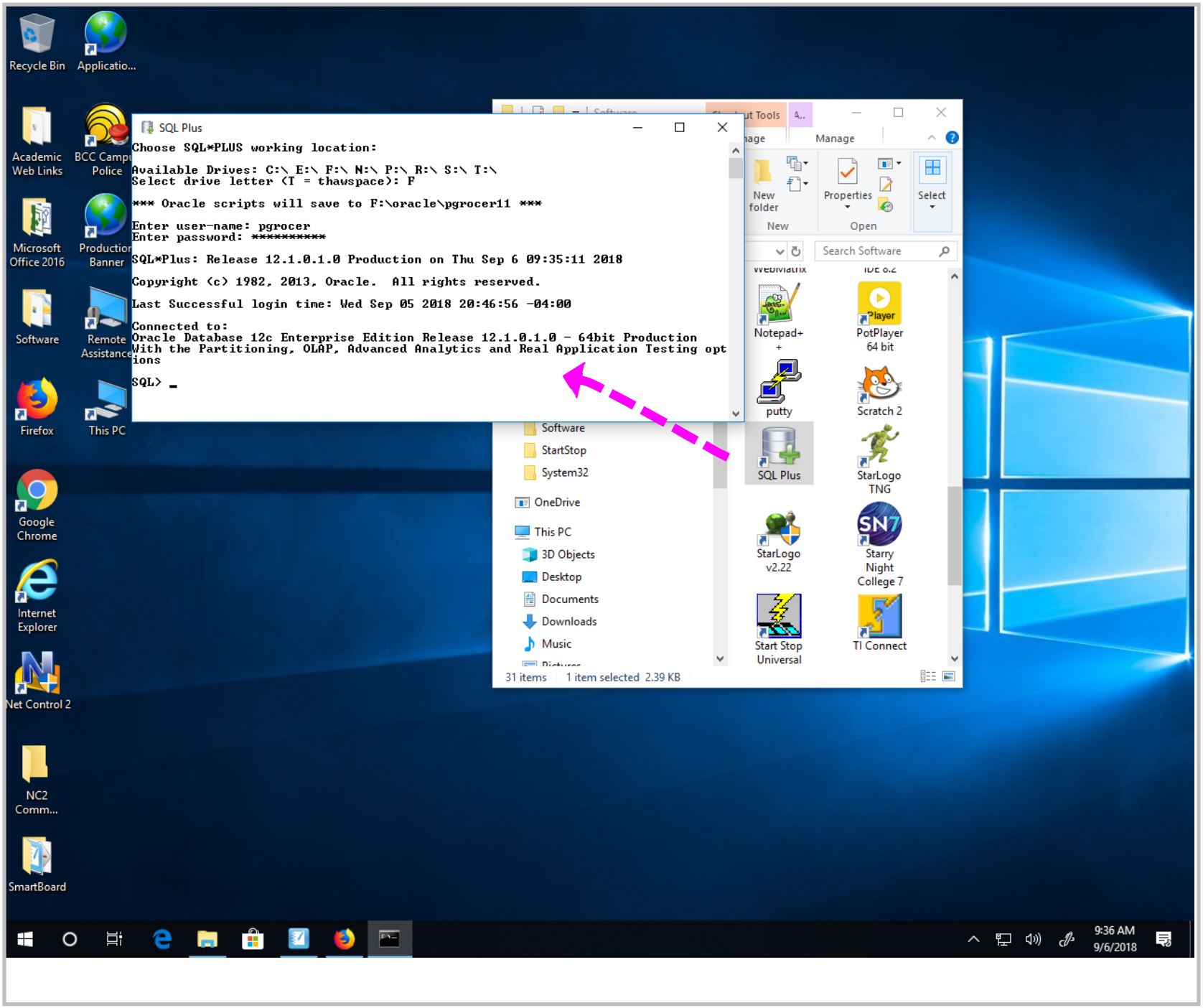
Title: Sep 6-9:28 AM (1 of 28)



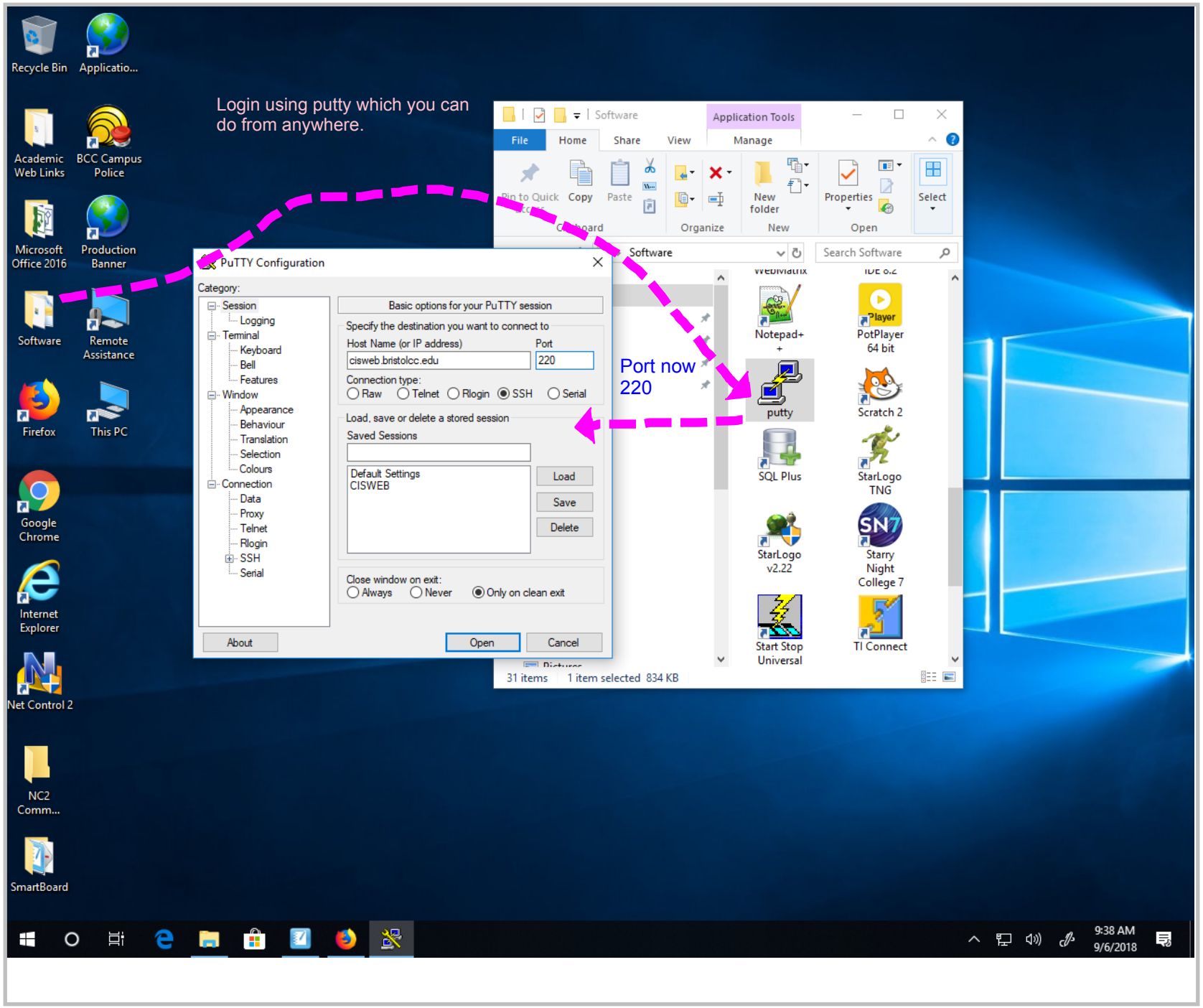
Title: Sep 6-9:34 AM (2 of 28)



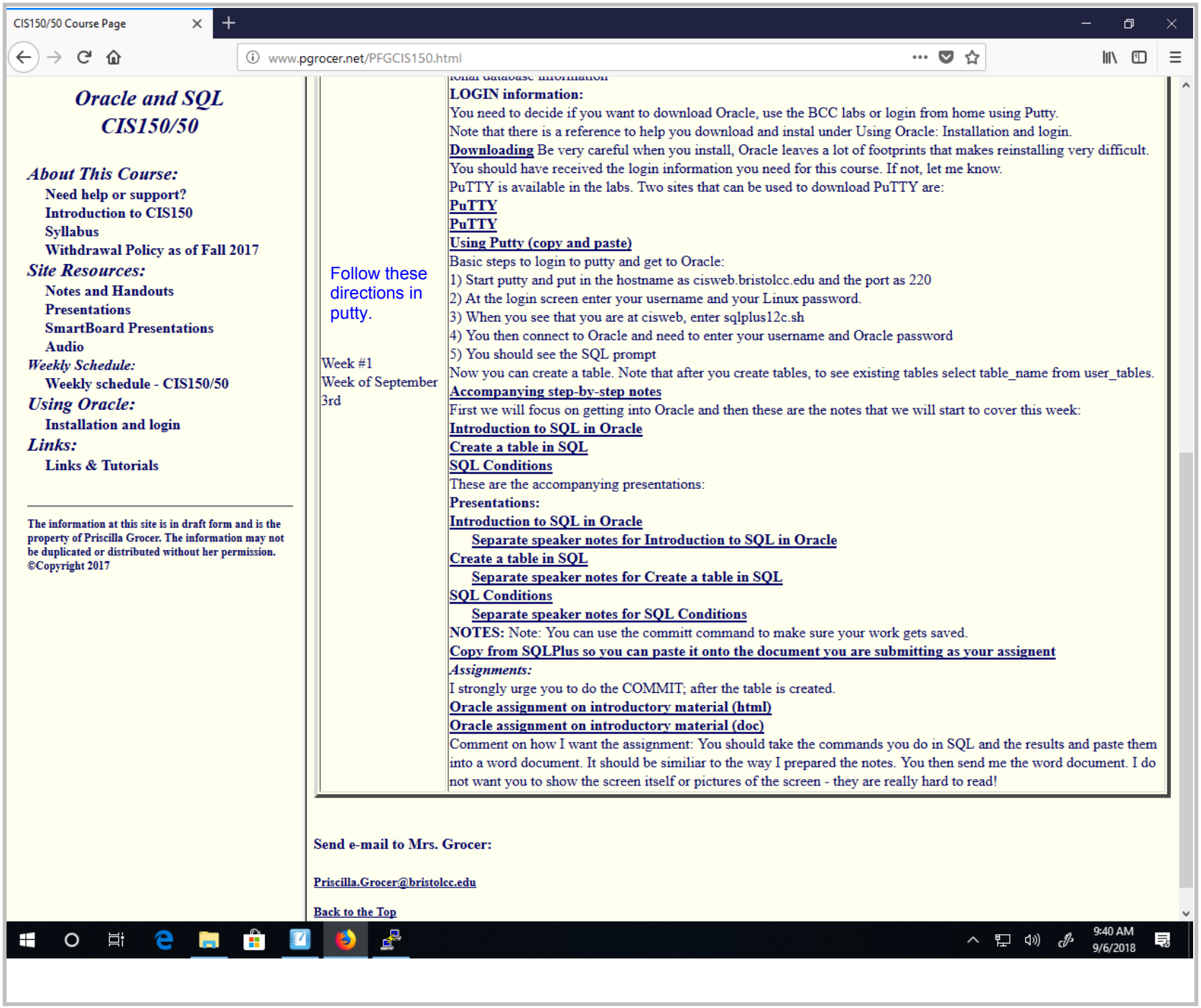
Title: Sep 6-9:36 AM (3 of 28)



Title: Sep 6-9:36 AM (4 of 28)

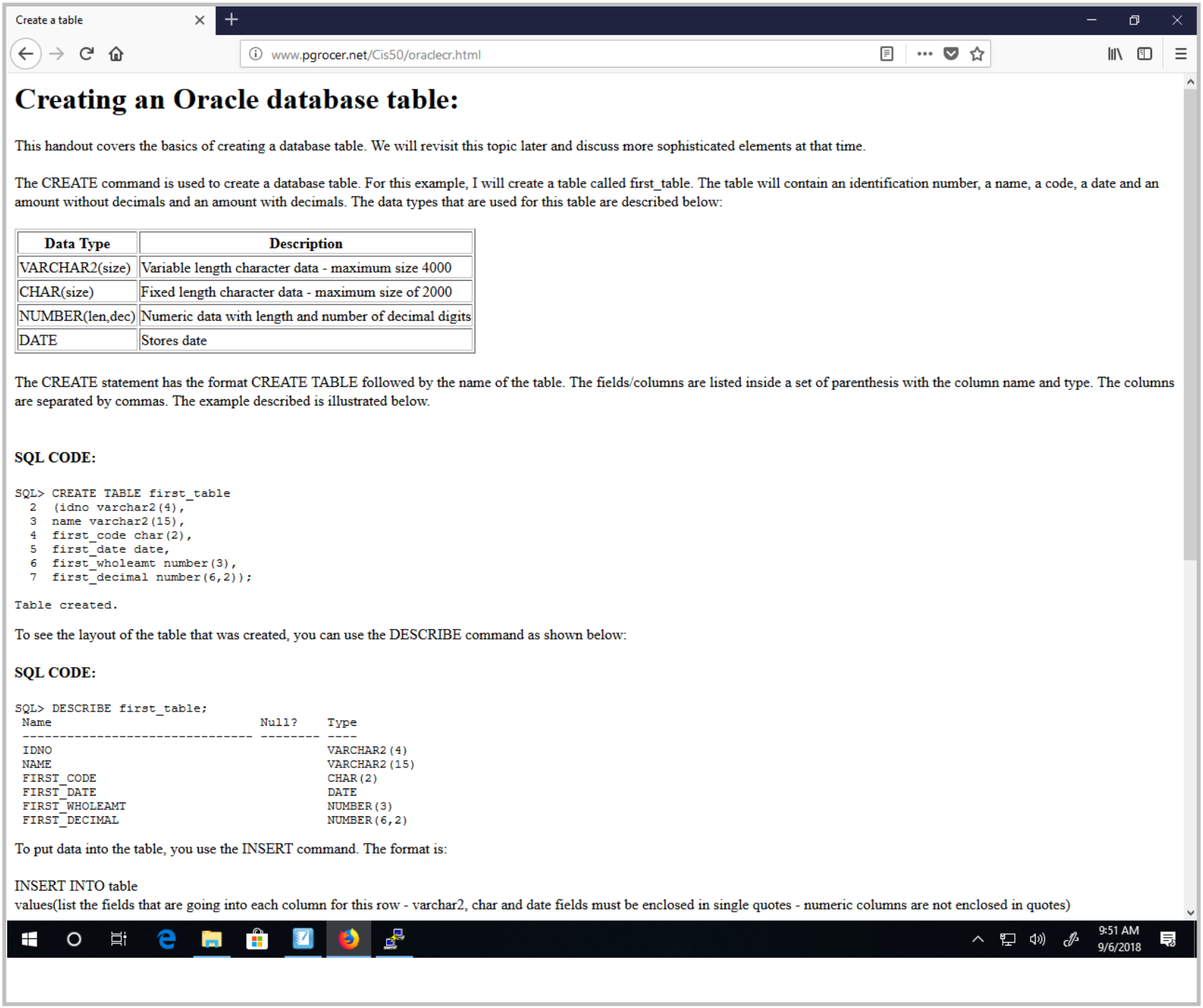


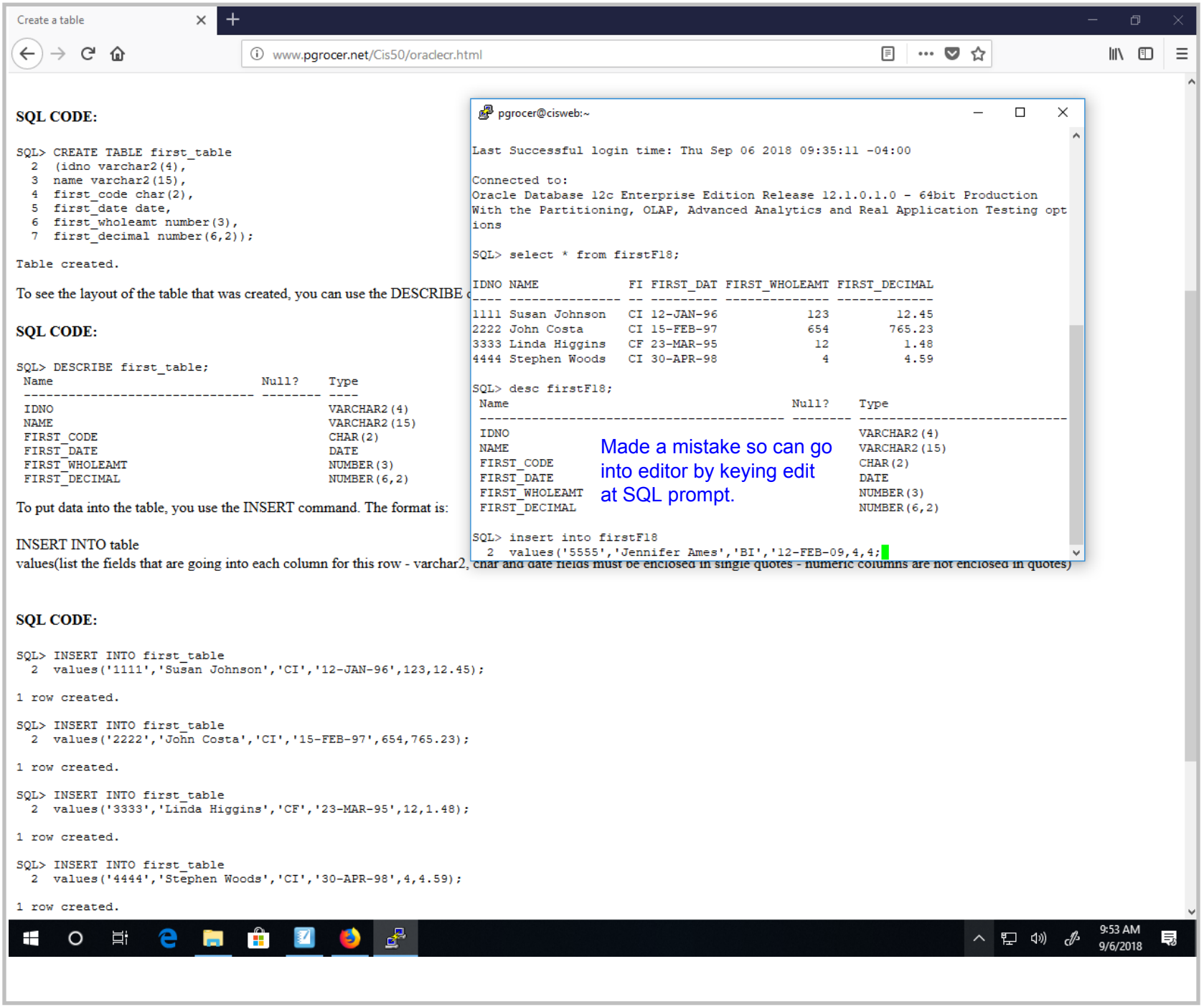
Title: Sep 6-9:38 AM (5 of 28)

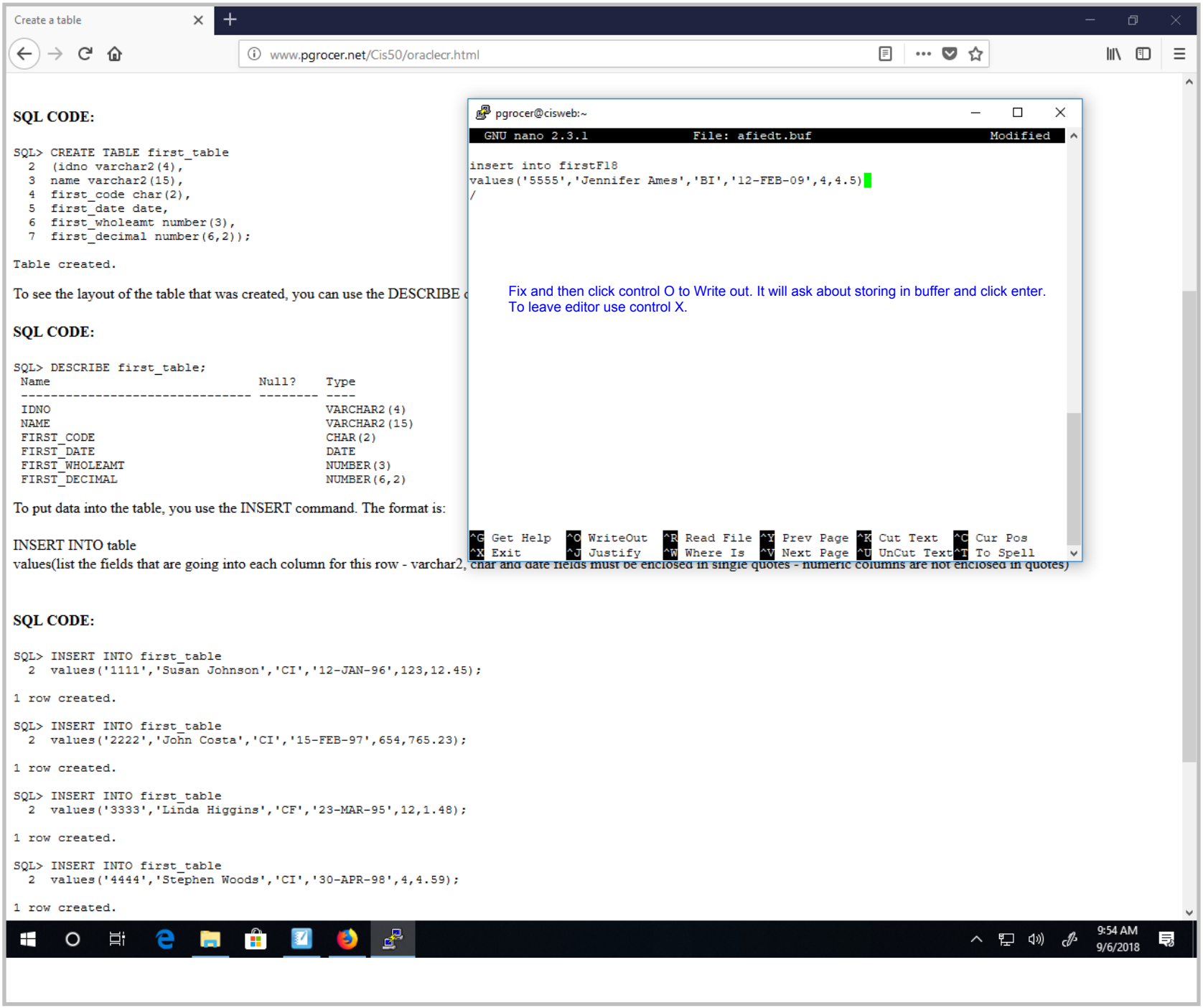


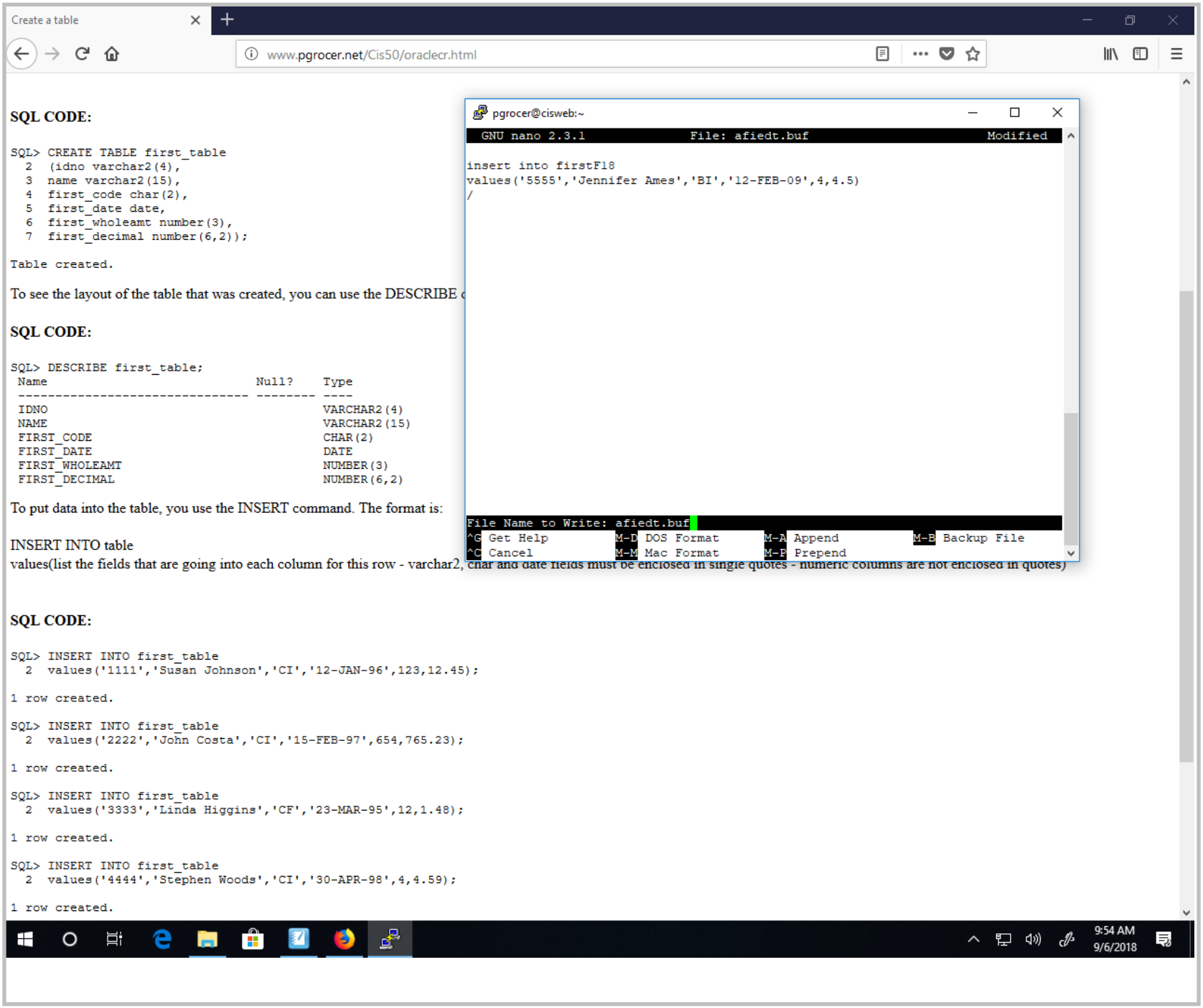
Title: Sep 6-9:40 AM (6 of 28)

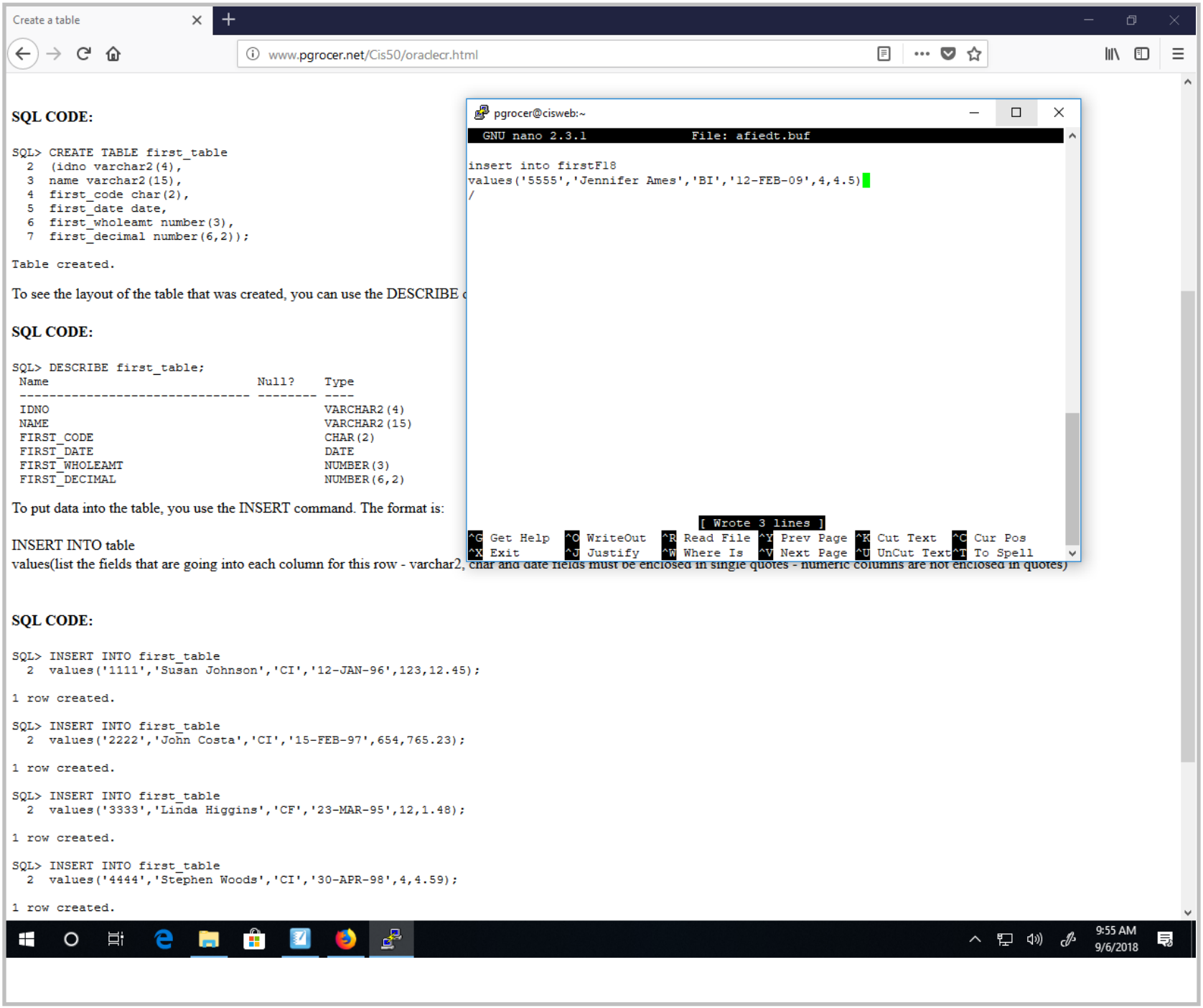
Title: Sep 6-9:49 AM (7 of 28)

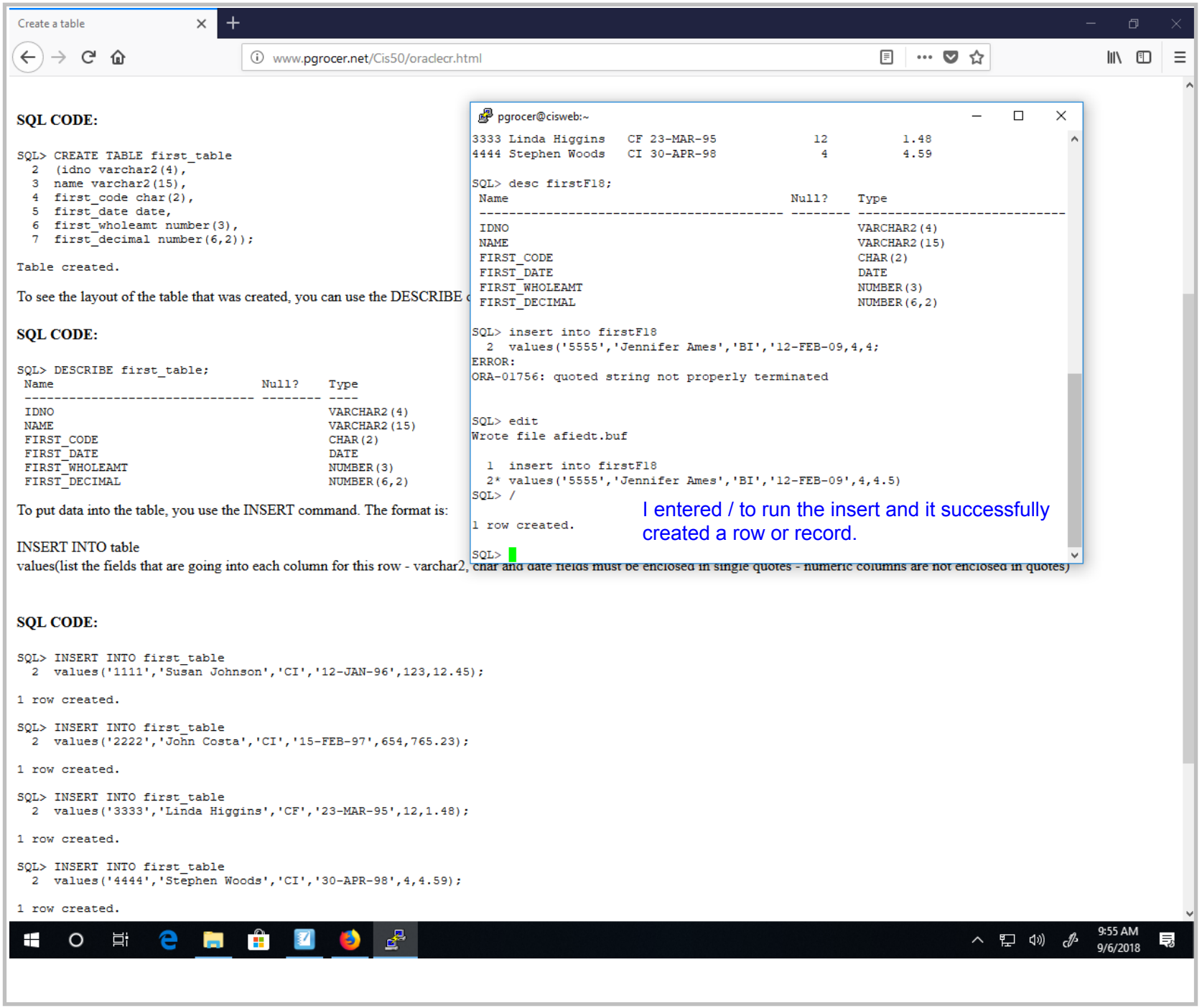

Create a table
www.pgrocer.net/Cis50/oraclecr.html
Creating an Oracle database table:
This handout covers the basics of creating a database table. We will revisit this topic later and discuss more sophisticated elements at that time.
The CREATE command is used to create a database table. For this example, I will create a table called first_table. The table will contain an identification number, a name, a code, a date and an amount without decimals and an amount with decimals. The data types that are used for this table are described below:
Data Type
Description
VARCHAR2(size)
Variable length character data - maximum size 4000
CHAR(size)
Fixed length character data - maximum size of 2000
NUMBER(len,dec)
Numeric data with length and number of decimal digits
DATE
Stores date
The CREATE statement has the format CREATE TABLE followed by the name of the table. The fields/columns are listed inside a set of parenthesis with the column name and type. The columns are separated by commas. The example described is illustrated below.
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE command as shown below:
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
9:51 AM
9/6/2018

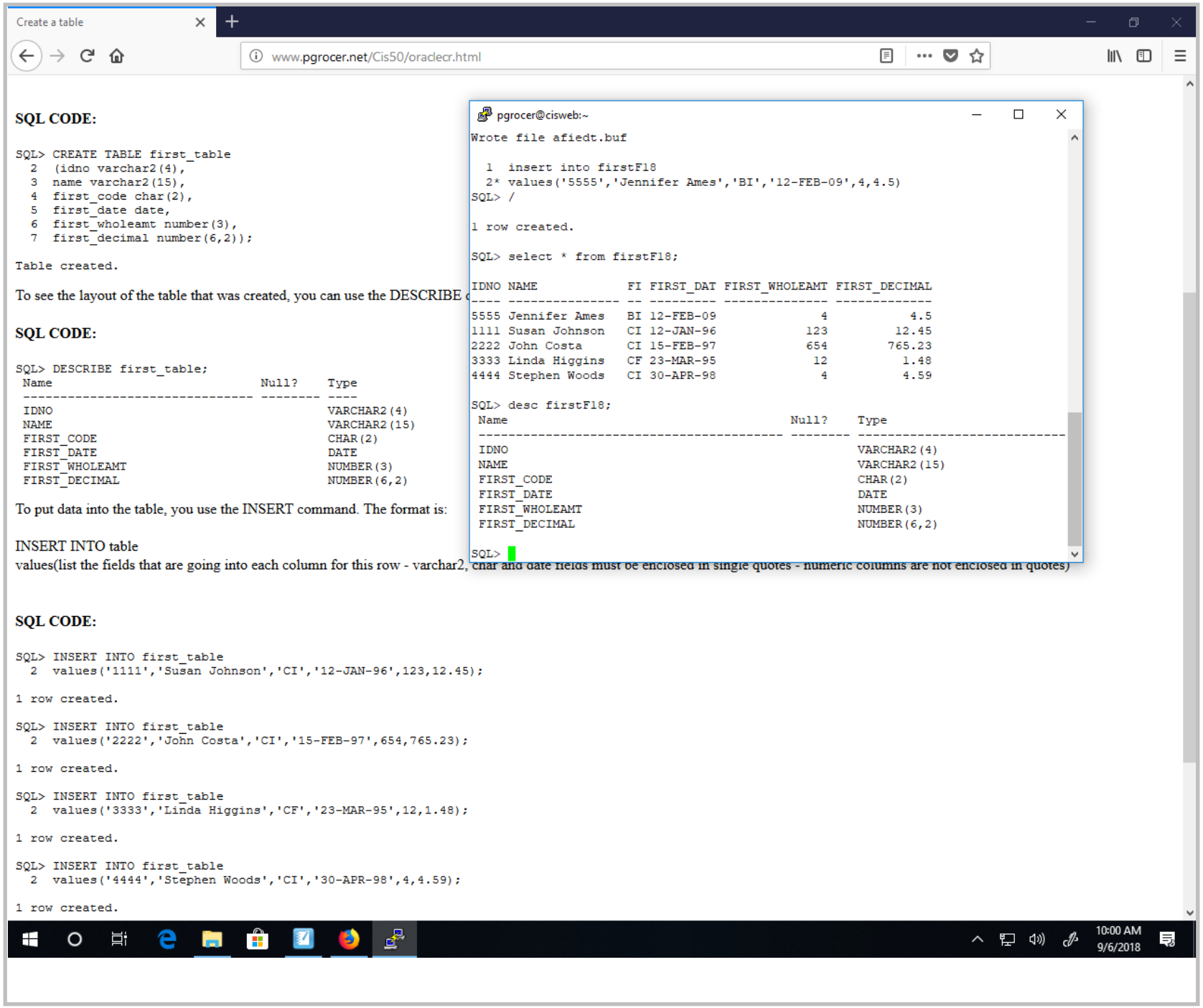

Create a table
www.pgrocer.net/Cis50/oraclecr.html
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
SQL CODE:
SQL> INSERT INTO first_table
2 values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);
1 row created.
SQL> INSERT INTO first_table
2 values('2222','John Costa','CI','15-FEB-97',654,765.23);
1 row created.
SQL> INSERT INTO first_table
2 values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);
1 row created.
SQL> INSERT INTO first_table
2 values('4444','Stephen Woods','CI','30-APR-98',4,4.59);
1 row created.
pgrocer@cisweb:~
Last Successful login time: Thu Sep 06 2018 09:35:11 -04:00
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions
SQL> select * from firstF18;
IDNO NAME FI FIRST_DAT FIRST_WHOLEAMT FIRST_DECIMAL
1111 Susan Johnson CI 12-JAN-96 123 12.45
2222 John Costa CI 15-FEB-97 654 765.23
3333 Linda Higgins CF 23-MAR-95 12 1.48
4444 Stephen Woods CI 30-APR-98 4 4.59
SQL> desc firstF18;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
Made a mistake so can go into editor by keying edit at SQL prompt.
SQL> insert into firstF18
2 values('5555','Jennifer Ames','BI','12-FEB-09,4,4;
9:53 AM
9/6/2018

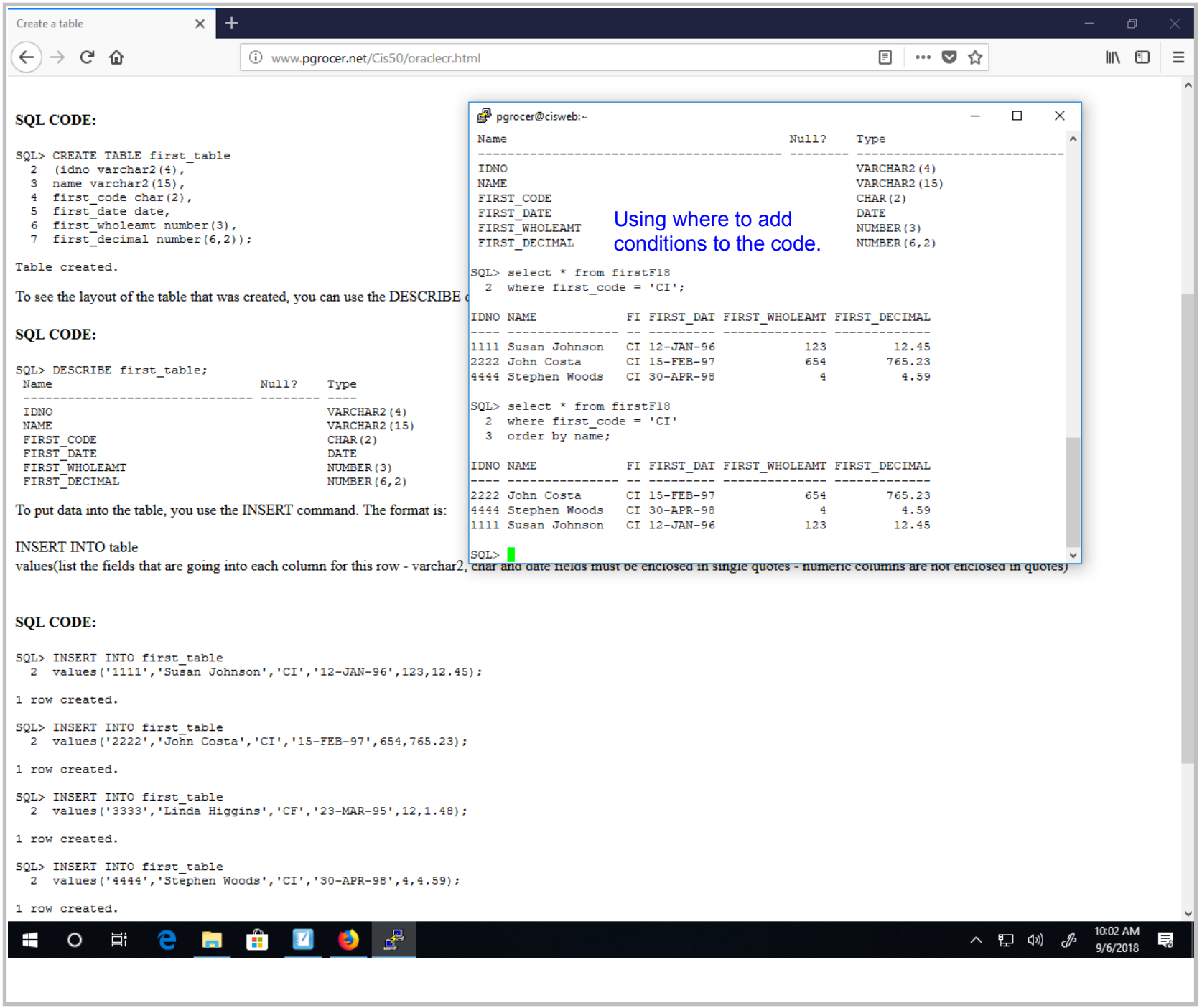

Create a table
www.pgrocer.net/Cis50/oraclecr.html
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
SQL CODE:
SQL> INSERT INTO first_table
2 values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);
1 row created.
SQL> INSERT INTO first_table
2 values('2222','John Costa','CI','15-FEB-97',654,765.23);
1 row created.
SQL> INSERT INTO first_table
2 values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);
1 row created.
SQL> INSERT INTO first_table
2 values('4444','Stephen Woods','CI','30-APR-98',4,4.59);
1 row created.
pgrocer@cisweb:~
GNU nano 2.3.1 File: afiedt.buf Modified
insert into firstF18
values('5555','Jennifer Ames','BI','12-FEB-09',4,4.5)
/
Fix and then click control O to Write out. It will ask about storing in buffer and click enter.
To leave editor use control X.
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
9:54 AM
9/6/2018

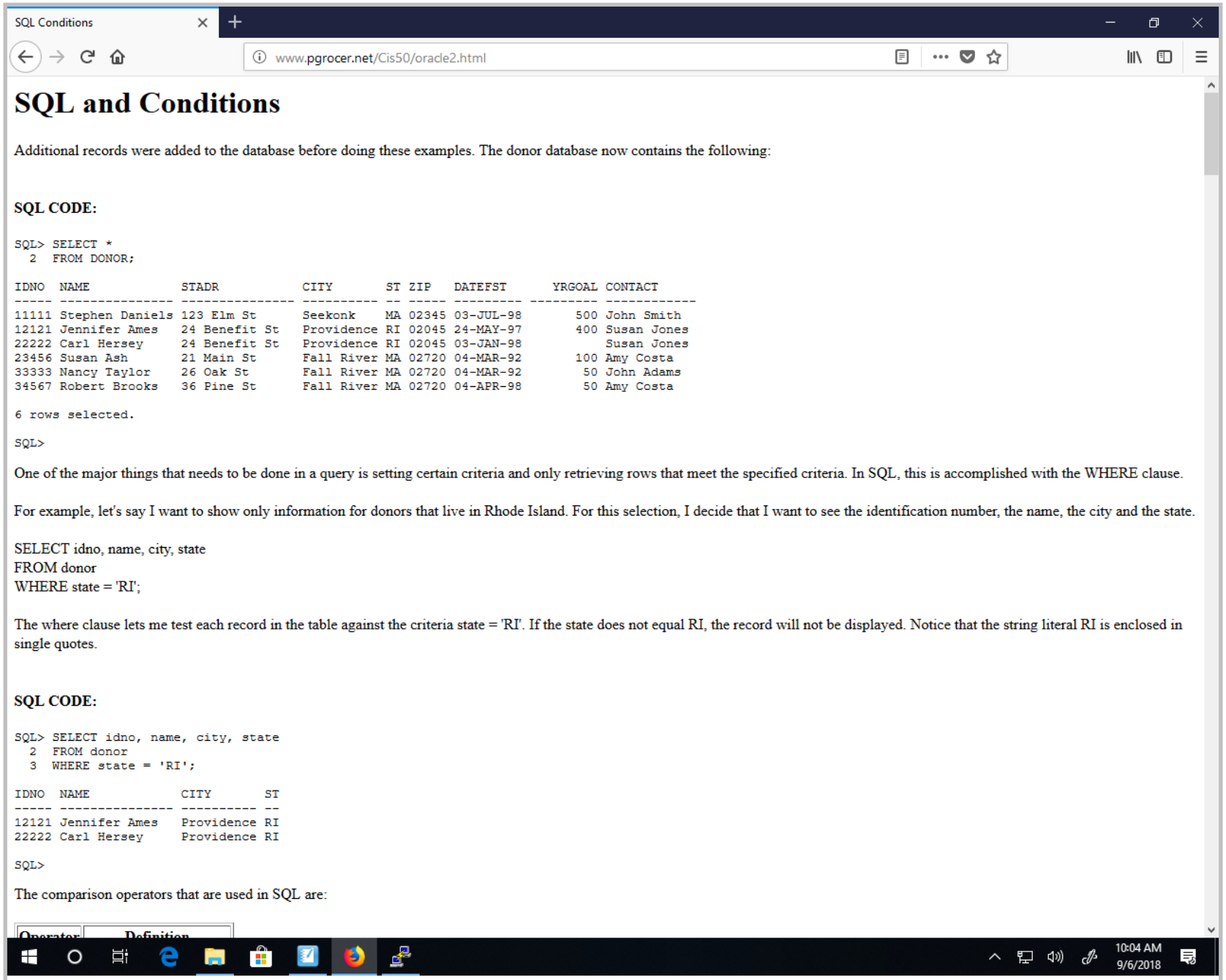

Create a table
www.pgrocer.net/Cis50/oraclecr.html
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
SQL CODE:
SQL> INSERT INTO first_table
2 values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);
1 row created.
SQL> INSERT INTO first_table
2 values('2222','John Costa','CI','15-FEB-97',654,765.23);
1 row created.
SQL> INSERT INTO first_table
2 values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);
1 row created.
SQL> INSERT INTO first_table
2 values('4444','Stephen Woods','CI','30-APR-98',4,4.59);
1 row created.
pgrocer@cisweb:~
GNU nano 2.3.1 File: afiedt.buf Modified
insert into firstF18
values('5555','Jennifer Ames','BI','12-FEB-09',4,4.5)
/
File Name to Write: afiedt.buf
^G Get Help M-D DOS Format M-A Append M-B Backup File
^C Cancel M-M Mac Format M-P Prepend
9:54 AM
9/6/2018


Create a table
www.pgrocer.net/Cis50/oraclecr.html
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
SQL CODE:
SQL> INSERT INTO first_table
2 values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);
1 row created.
SQL> INSERT INTO first_table
2 values('2222','John Costa','CI','15-FEB-97',654,765.23);
1 row created.
SQL> INSERT INTO first_table
2 values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);
1 row created.
SQL> INSERT INTO first_table
2 values('4444','Stephen Woods','CI','30-APR-98',4,4.59);
1 row created.
pgrocer@cisweb:~
GNU nano 2.3.1 File: afiedt.buf
insert into firstF18
values('5555','Jennifer Ames','BI','12-FEB-09',4,4.5)
/
[Wrote 3 lines]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
9:55 AM
9/6/2018


Create a table
www.pgrocer.net/Cis50/oraclecr.html
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
SQL CODE:
SQL> INSERT INTO first_table
2 values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);
1 row created.
SQL> INSERT INTO first_table
2 values('2222','John Costa','CI','15-FEB-97',654,765.23);
1 row created.
SQL> INSERT INTO first_table
2 values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);
1 row created.
SQL> INSERT INTO first_table
2 values('4444','Stephen Woods','CI','30-APR-98',4,4.59);
1 row created.
pgrocer@cisweb:~
3333 Linda Higgins CF 23-MAR-95 12 1.48
4444 Stephen Woods CI 30-APR-98 4 4.59
SQL> desc firstF18;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
SQL> insert into firstF18
2 values('5555','Jennifer Ames','BI','12-FEB-09,4,4;
ERROR:
ORA-01756: quoted string not properly terminated
SQL> edit
Wrote file afiedt.buf
1 insert into firstF18
2* values('5555','Jennifer Ames','BI','12-FEB-09',4,4.5)
SQL> /
1 row created.
SQL>
I entered / to run the insert and it successfully created a row or record.
9:55 AM
9/6/2018


Create a table
www.pgrocer.net/Cis50/oraclecr.html
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
SQL CODE:
SQL> INSERT INTO first_table
2 values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);
1 row created.
SQL> INSERT INTO first_table
2 values('2222','John Costa','CI','15-FEB-97',654,765.23);
1 row created.
SQL> INSERT INTO first_table
2 values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);
1 row created.
SQL> INSERT INTO first_table
2 values('4444','Stephen Woods','CI','30-APR-98',4,4.59);
1 row created.
pgrocer@cisweb:~
Wrote file afiedt.buf
1 insert into firstF18
2* values('5555','Jennifer Ames','BI','12-FEB-09',4,4.5)
SQL> /
1 row created.
SQL> select * from firstF18;
IDNO NAME FI FIRST_DAT FIRST_WHOLEAMT FIRST_DECIMAL
5555 Jennifer Ames BI 12-FEB-09 4 4.5
1111 Susan Johnson CI 12-JAN-96 123 12.45
2222 John Costa CI 15-FEB-97 654 765.23
3333 Linda Higgins CF 23-MAR-95 12 1.48
4444 Stephen Woods CI 30-APR-98 4 4.59
SQL> desc firstF18;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
SQL>
10:00 AM
9/6/2018


Create a table
www.pgrocer.net/Cis50/oraclecr.html
SQL CODE:
SQL> CREATE TABLE first_table
2 (idno varchar2(4),
3 name varchar2(15),
4 first_code char(2),
5 first_date date,
6 first_wholeamt number(3),
7 first_decimal number(6,2));
Table created.
To see the layout of the table that was created, you can use the DESCRIBE
SQL CODE:
SQL> DESCRIBE first_table;
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
To put data into the table, you use the INSERT command. The format is:
INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)
SQL CODE:
SQL> INSERT INTO first_table
2 values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);
1 row created.
SQL> INSERT INTO first_table
2 values('2222','John Costa','CI','15-FEB-97',654,765.23);
1 row created.
SQL> INSERT INTO first_table
2 values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);
1 row created.
SQL> INSERT INTO first_table
2 values('4444','Stephen Woods','CI','30-APR-98',4,4.59);
1 row created.
pgrocer@cisweb:~
Name Null? Type
IDNO VARCHAR2(4)
NAME VARCHAR2(15)
FIRST_CODE CHAR(2)
FIRST_DATE DATE
FIRST_WHOLEAMT NUMBER(3)
FIRST_DECIMAL NUMBER(6,2)
Using where to add conditions to the code.
SQL> select * from firstF18
2 where first_code = 'CI';
IDNO NAME FI FIRST_DAT FIRST_WHOLEAMT FIRST_DECIMAL
1111 Susan Johnson CI 12-JAN-96 123 12.45
2222 John Costa CI 15-FEB-97 654 765.23
4444 Stephen Woods CI 30-APR-98 4 4.59
SQL> select * from firstF18
2 where first_code = 'CI'
3 order by name;
IDNO NAME FI FIRST_DAT FIRST_WHOLEAMT FIRST_DECIMAL
2222 John Costa CI 15-FEB-97 654 765.23
4444 Stephen Woods CI 30-APR-98 4 4.59
1111 Susan Johnson CI 12-JAN-96 123 12.45
SQL>
10:02 AM
9/6/2018



SQL and Conditions

Additional records were added to the database before doing these examples. The donor database now contains the following:

SQL CODE:

```
SQL> SELECT *
  2  FROM DONOR;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       500 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       400 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       100 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98        50 Amy Costa

6 rows selected.

SQL>
```

One of the major things that needs to be done in a query is setting certain criteria and only retrieving rows that meet the specified criteria. In SQL, this is accomplished with the WHERE clause.

For example, let's say I want to show only information for donors that live in Rhode Island. For this selection, I decide that I want to see the identification number, the name, the city and the state.

SELECT idno, name, city, state
FROM donor
WHERE state = 'RI';

The where clause lets me test each record in the table against the criteria state = 'RI'. If the state does not equal RI, the record will not be displayed. Notice that the string literal RI is enclosed in single quotes.

SQL CODE:

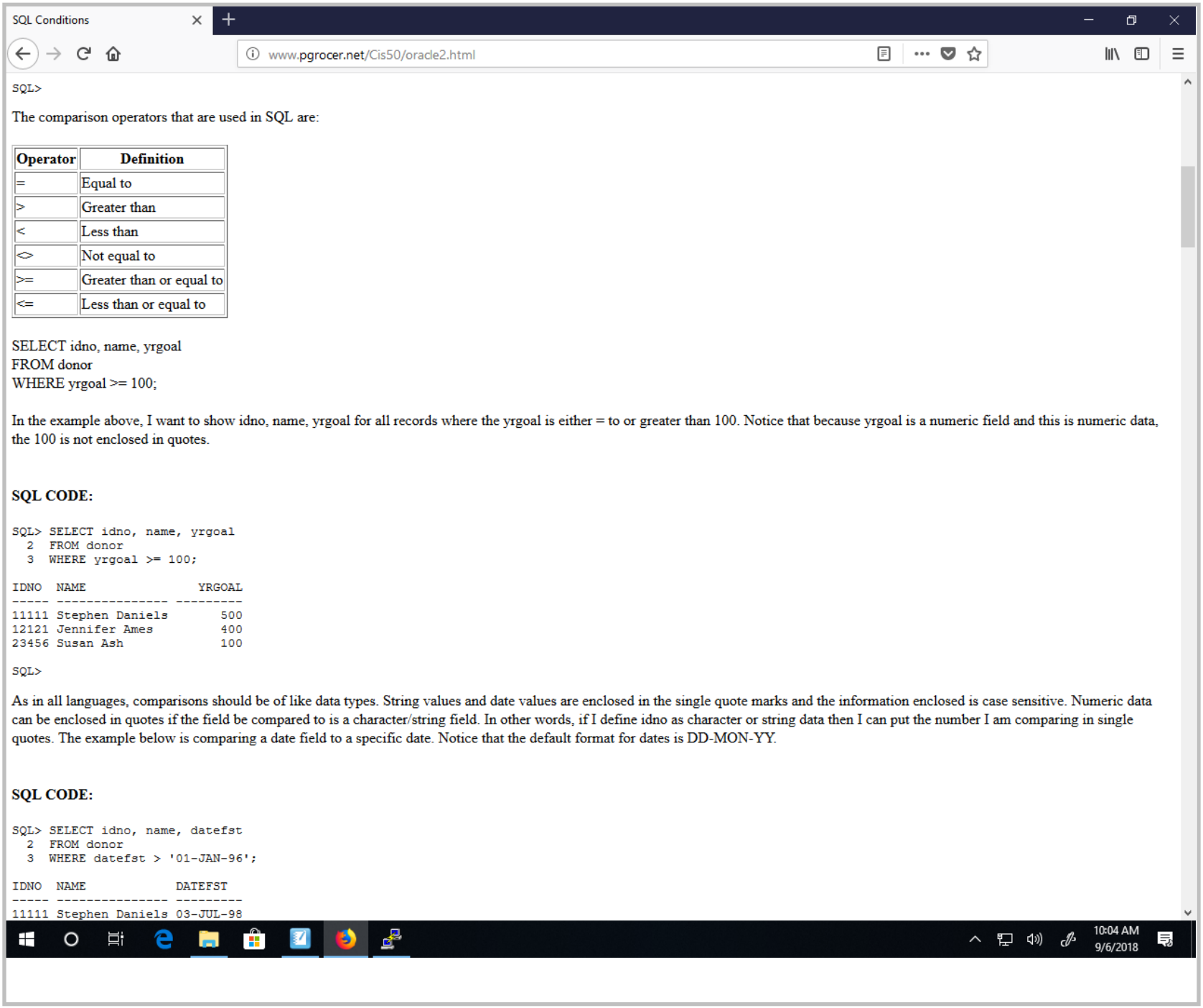
```
SQL> SELECT idno, name, city, state
  2  FROM donor
  3  WHERE state = 'RI';

IDNO  NAME            CITY       ST
----- --------------- ---------- --
12121 Jennifer Ames   Providence RI
22222 Carl Hersey     Providence RI

SQL>
```

The comparison operators that are used in SQL are:

Operator	Definition



SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

```
SQL>
```

The comparison operators that are used in SQL are:

Operator	Definition
=	Equal to
>	Greater than
<	Less than
<>	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

SELECT idno, name, yrgoal
FROM donor
WHERE yrgoal >= 100;

In the example above, I want to show idno, name, yrgoal for all records where the yrgoal is either = to or greater than 100. Notice that because yrgoal is a numeric field and this is numeric data, the 100 is not enclosed in quotes.

SQL CODE:

```
SQL> SELECT idno, name, yrgoal
  2  FROM donor
  3  WHERE yrgoal >= 100;

IDNO  NAME                YRGOAL
----- --------------- ---------
11111 Stephen Daniels       500
12121 Jennifer Ames         400
23456 Susan Ash             100

SQL>
```

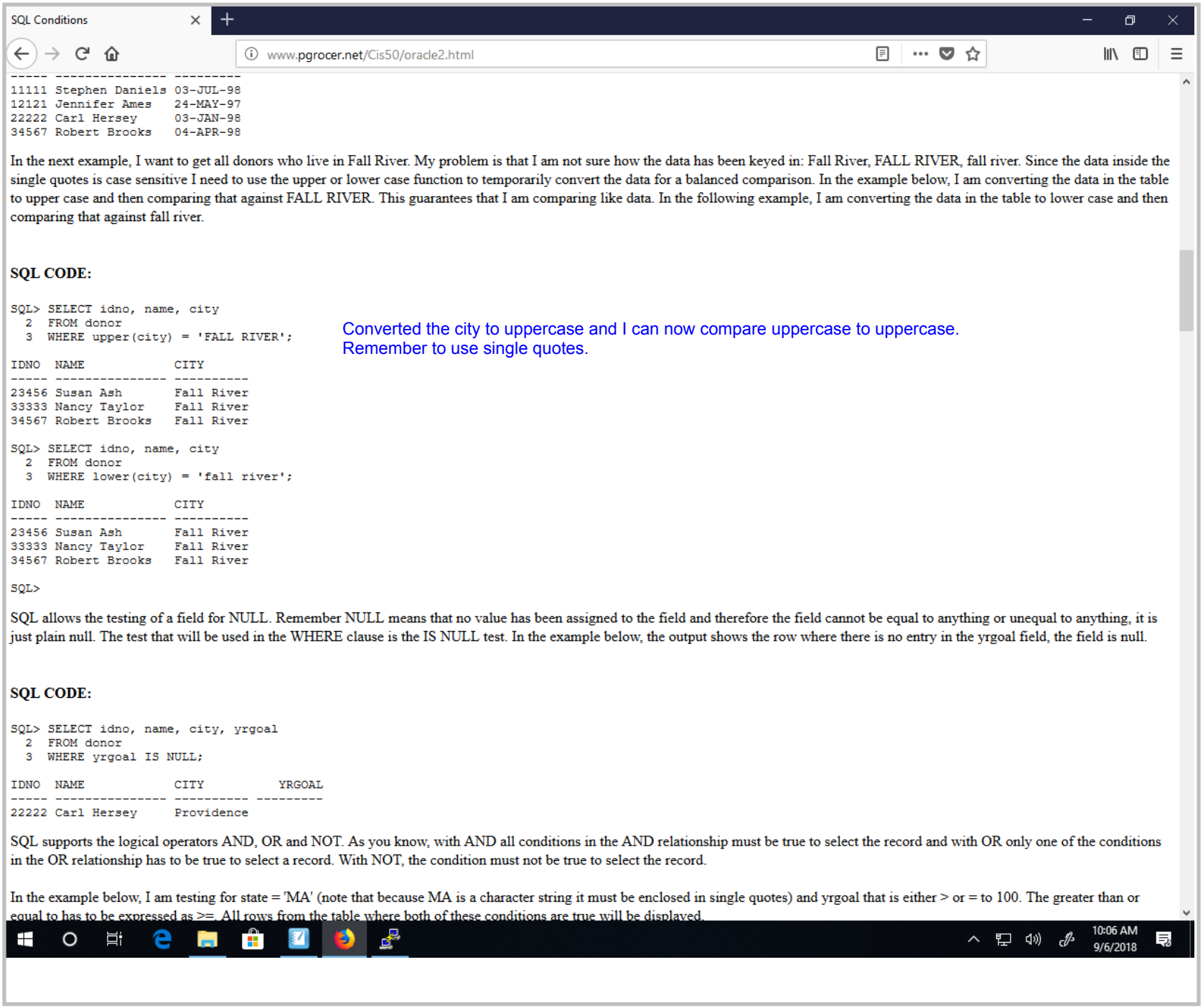
As in all languages, comparisons should be of like data types. String values and date values are enclosed in the single quote marks and the information enclosed is case sensitive. Numeric data can be enclosed in quotes if the field be compared to is a character/string field. In other words, if I define idno as character or string data then I can put the number I am comparing in single quotes. The example below is comparing a date field to a specific date. Notice that the default format for dates is DD-MON-YY.

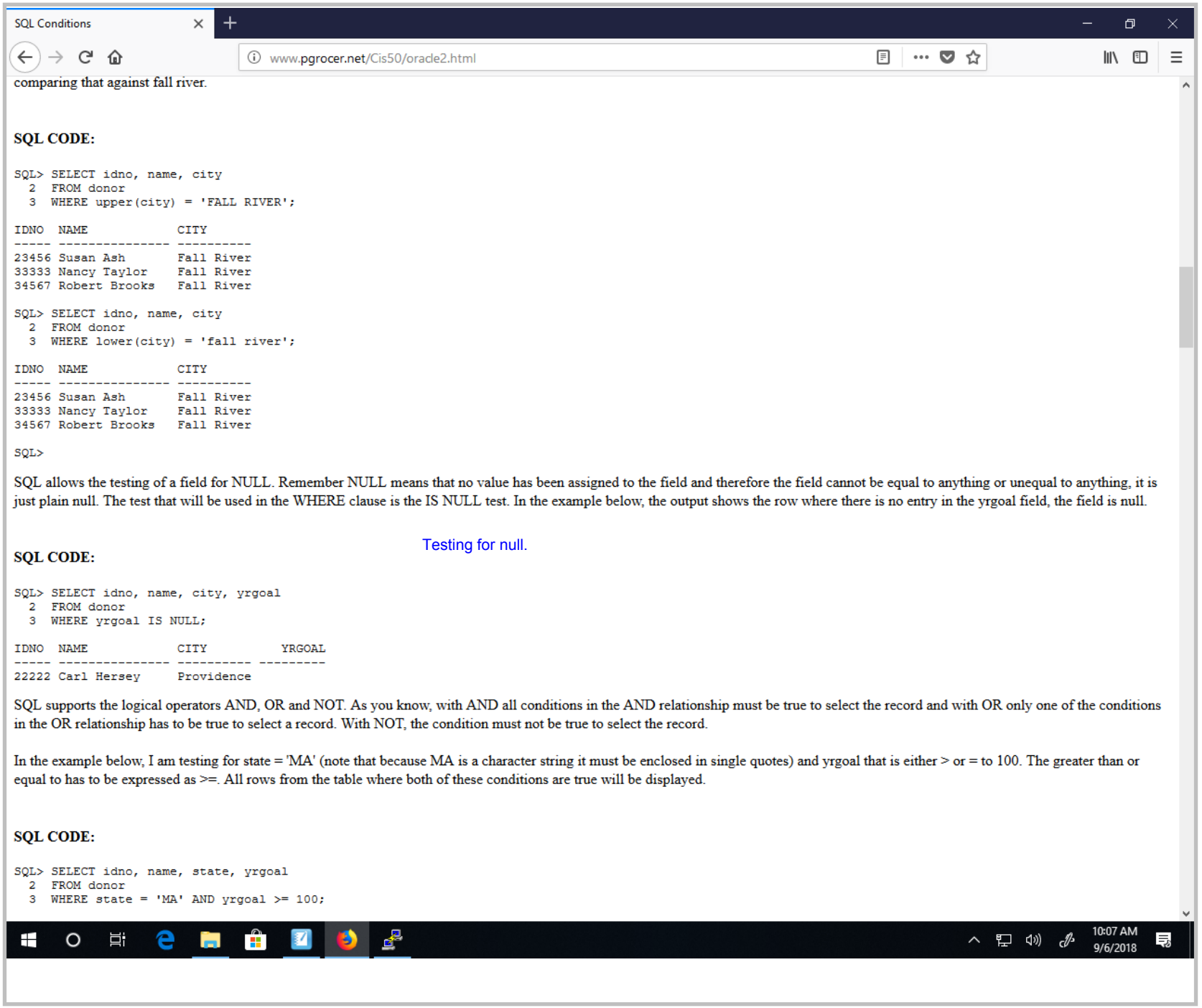
SQL CODE:

```
SQL> SELECT idno, name, datefst
  2  FROM donor
  3  WHERE datefst > '01-JAN-96';

IDNO  NAME            DATEFST
----- --------------- ---------
11111 Stephen Daniels 03-JUL-98
```

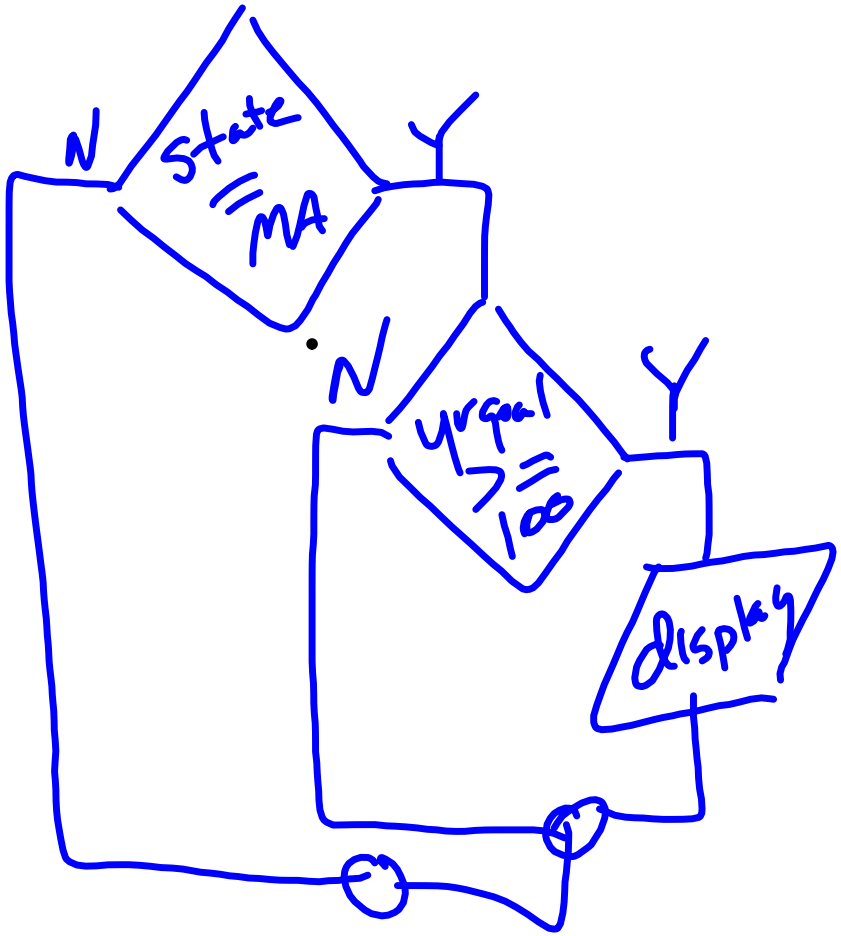
10:04 AM
9/6/2018


SQL Conditions
www.pgrocer.net/Cis50/oracle2.html
----- --------------- ---------
11111 Stephen Daniels 03-JUL-98
12121 Jennifer Ames 24-MAY-97
22222 Carl Hersey 03-JAN-98
34567 Robert Brooks 04-APR-98
In the next example, I want to get all donors who live in Fall River. My problem is that I am not sure how the data has been keyed in: Fall River, FALL RIVER, fall river. Since the data inside the single quotes is case sensitive I need to use the upper or lower case function to temporarily convert the data for a balanced comparison. In the example below, I am converting the data in the table to upper case and then comparing that against FALL RIVER. This guarantees that I am comparing like data. In the following example, I am converting the data in the table to lower case and then comparing that against fall river.
SQL CODE:
SQL> SELECT idno, name, city
2 FROM donor
3 WHERE upper(city) = 'FALL RIVER';
Converted the city to uppercase and I can now compare uppercase to uppercase.
Remember to use single quotes.
IDNO NAME CITY
----- --------------- ----------
23456 Susan Ash Fall River
33333 Nancy Taylor Fall River
34567 Robert Brooks Fall River
SQL> SELECT idno, name, city
2 FROM donor
3 WHERE lower(city) = 'fall river';
IDNO NAME CITY
----- --------------- ----------
23456 Susan Ash Fall River
33333 Nancy Taylor Fall River
34567 Robert Brooks Fall River
SQL>
SQL allows the testing of a field for NULL. Remember NULL means that no value has been assigned to the field and therefore the field cannot be equal to anything or unequal to anything, it is just plain null. The test that will be used in the WHERE clause is the IS NULL test. In the example below, the output shows the row where there is no entry in the yrgoal field, the field is null.
SQL CODE:
SQL> SELECT idno, name, city, yrgoal
2 FROM donor
3 WHERE yrgoal IS NULL;
IDNO NAME CITY YRGOAL
----- --------------- ---------- ---------
22222 Carl Hersey Providence
SQL supports the logical operators AND, OR and NOT. As you know, with AND all conditions in the AND relationship must be true to select the record and with OR only one of the conditions in the OR relationship has to be true to select a record. With NOT, the condition must not be true to select the record.
In the example below, I am testing for state = 'MA' (note that because MA is a character string it must be enclosed in single quotes) and yrgoal that is either > or = to 100. The greater than or equal to has to be expressed as >=. All rows from the table where both of these conditions are true will be displayed.
10:06 AM
9/6/2018

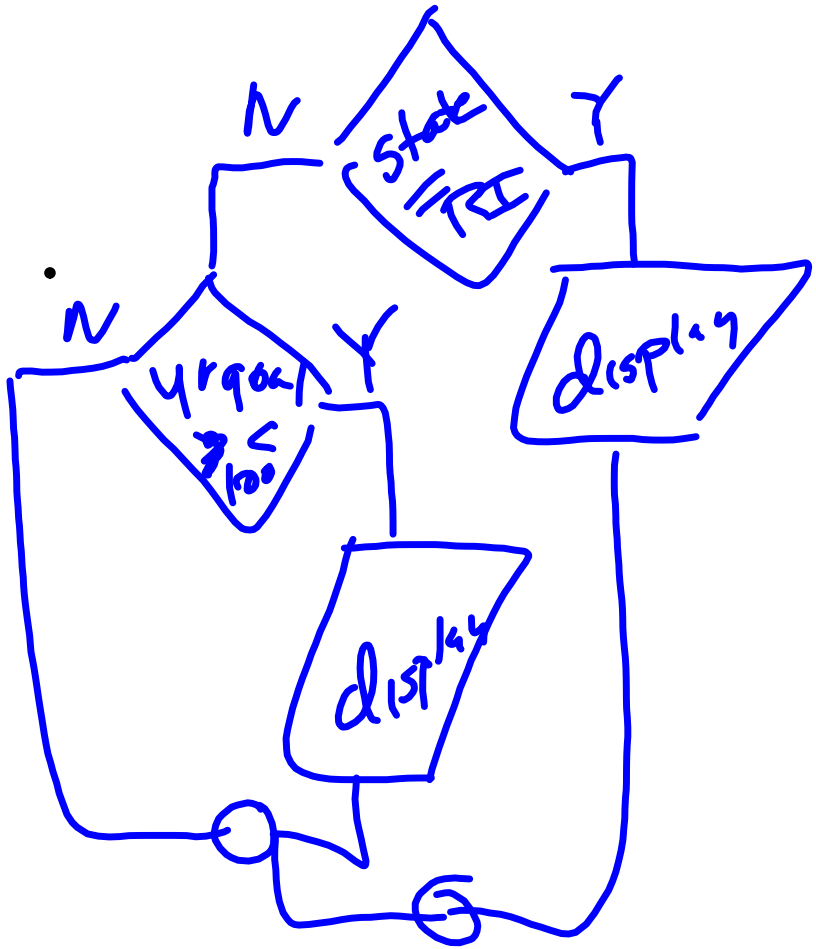

SQL Conditions
www.pgrocer.net/Cis50/oracle2.html
comparing that against fall river.
SQL CODE:
SQL> SELECT idno, name, city
2 FROM donor
3 WHERE upper(city) = 'FALL RIVER';
IDNO NAME CITY
23456 Susan Ash Fall River
33333 Nancy Taylor Fall River
34567 Robert Brooks Fall River
SQL> SELECT idno, name, city
2 FROM donor
3 WHERE lower(city) = 'fall river';
IDNO NAME CITY
23456 Susan Ash Fall River
33333 Nancy Taylor Fall River
34567 Robert Brooks Fall River
SQL>
SQL allows the testing of a field for NULL. Remember NULL means that no value has been assigned to the field and therefore the field cannot be equal to anything or unequal to anything, it is just plain null. The test that will be used in the WHERE clause is the IS NULL test. In the example below, the output shows the row where there is no entry in the yrgoal field, the field is null.
Testing for null.
SQL CODE:
SQL> SELECT idno, name, city, yrgoal
2 FROM donor
3 WHERE yrgoal IS NULL;
IDNO NAME CITY YRGOAL
22222 Carl Hersey Providence
SQL supports the logical operators AND, OR and NOT. As you know, with AND all conditions in the AND relationship must be true to select the record and with OR only one of the conditions in the OR relationship has to be true to select a record. With NOT, the condition must not be true to select the record.
In the example below, I am testing for state = 'MA' (note that because MA is a character string it must be enclosed in single quotes) and yrgoal that is either > or = to 100. The greater than or equal to has to be expressed as >=. All rows from the table where both of these conditions are true will be displayed.
SQL CODE:
SQL> SELECT idno, name, state, yrgoal
2 FROM donor
3 WHERE state = 'MA' AND yrgoal >= 100;
10:07 AM
9/6/2018

```
SQL> SELECT idno, name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'MA' AND yrgoal >= 100;

IDNO  NAME            ST    YRGOAL
----- --------------- -- ---------
11111 Stephen Daniels MA        500
23456 Susan Ash       MA        100
```



```
SQL> SELECT name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'RI' OR yrgoal < 100;
```



SQL CODE:

```
SQL> list
  1  SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE yrgoal >100 AND state = 'RI' OR contact = 'Amy Costa'
SQL> /

NAME           CITY       ST DATEFST      YRGOAL CONTACT
-------------- ---------- -- --------- --------- ------------
Jennifer Ames  Providence RI 24-MAY-97       400 Susan Jones
Susan Ash      Fall River MA 04-MAR-92       100 Amy Costa
Robert Brooks  Fall River MA 04-APR-98        50 Amy Costa
```

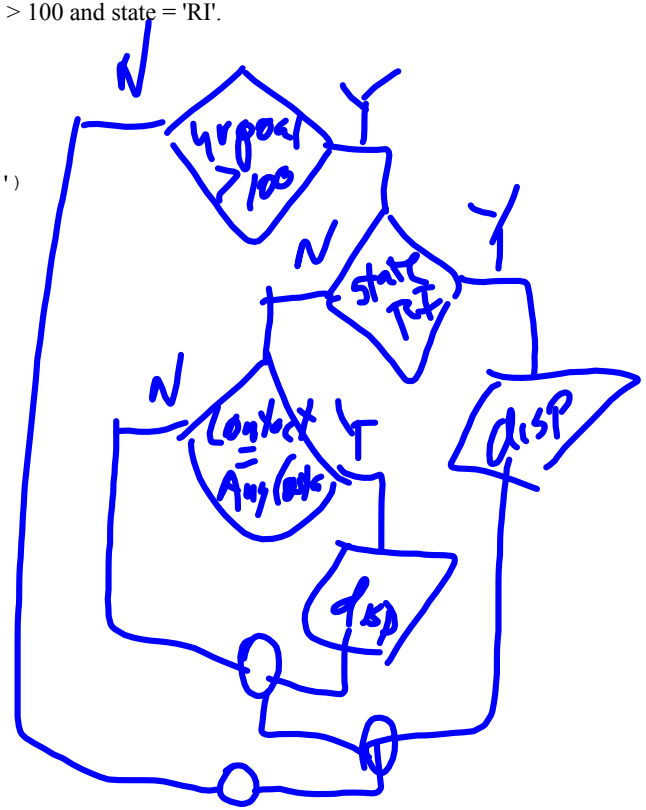
When you look at the results of the SQL query above, the first record meets the criteria yrgoal > 100 and state = 'RI', the second and third record meet the criteria contact = 'Amy Costa'.

In this second example the parenthesis are included so I will get records where yrgoal is always greater than 100 and either the state = 'RI' or the contact = 'Amy Costa'. In this case, only one record met the criteria. It had yrgoal > 100 and state = 'RI'.

SQL CODE:

```
SQL> list
  1  SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE yrgoal > 100 AND (state = 'RI' OR contact = 'Amy Costa')
SQL> /

NAME           CITY       ST DATEFST      YRGOAL CONTACT
-------------- ---------- -- --------- --------- ------------
Jennifer Ames  Providence RI 24-MAY-97       400 Susan Jones
```

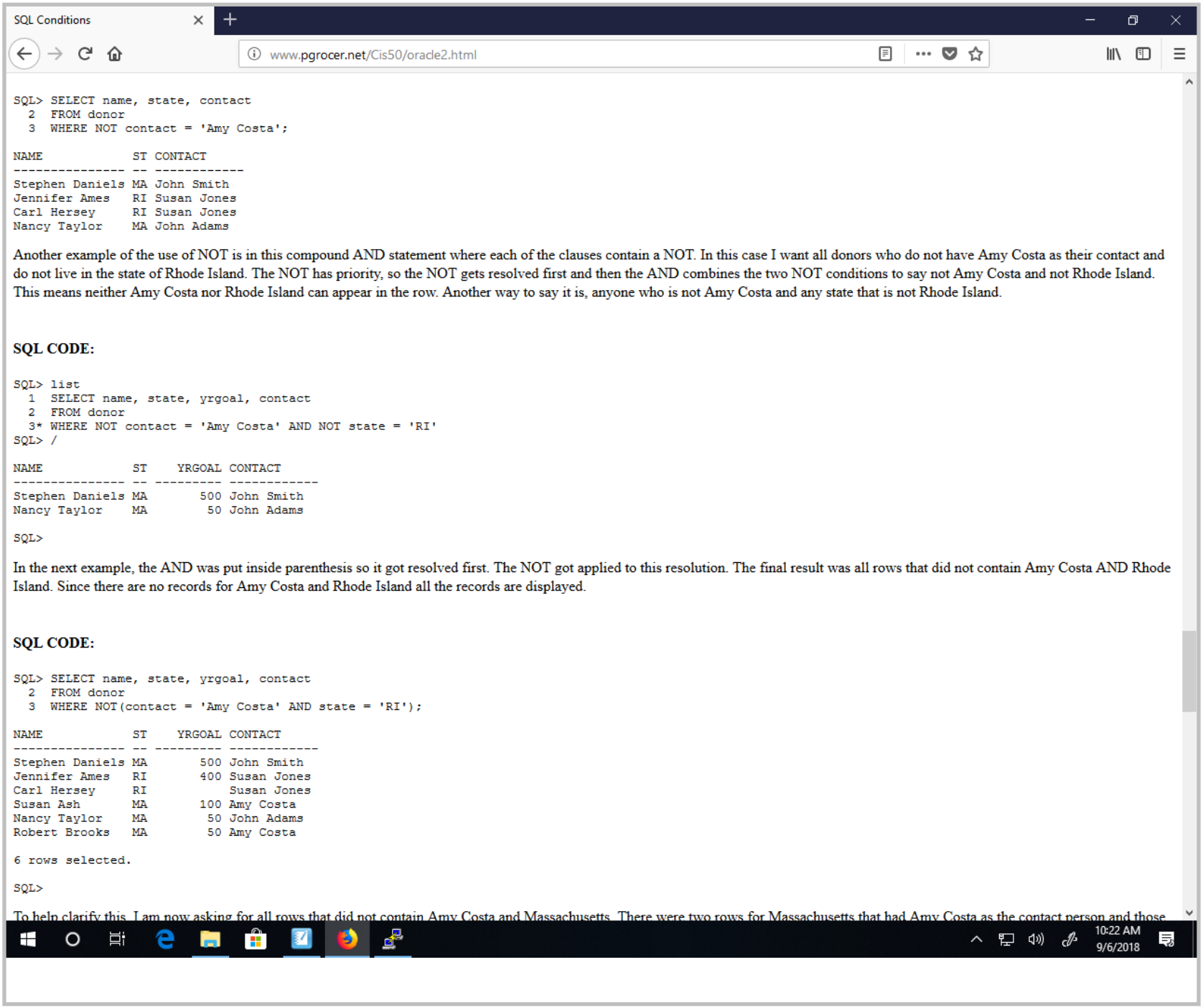


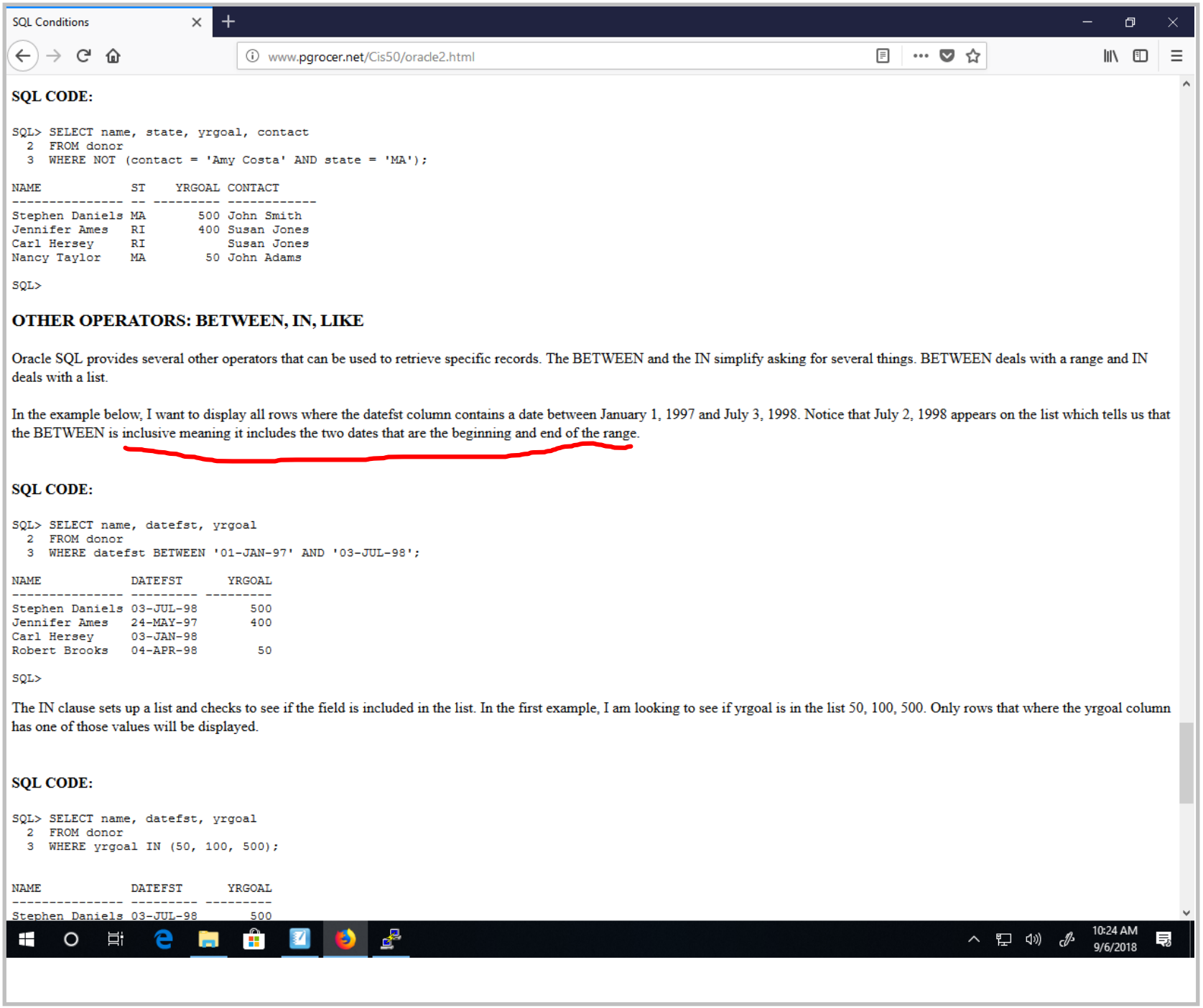
Another example of a combined AND and OR is shown below. In this example I want state of Rhode Island with a date of first contribution greater than January 1, 1998 or the state of Massachusetts with a date for first contribution greater than January 1, 1996. Notice that no parenthesis are needed because the two ANDs get resolved first and then the results hit the OR test. Parenthesis could have been inserted around the two AND combinations for clarity.

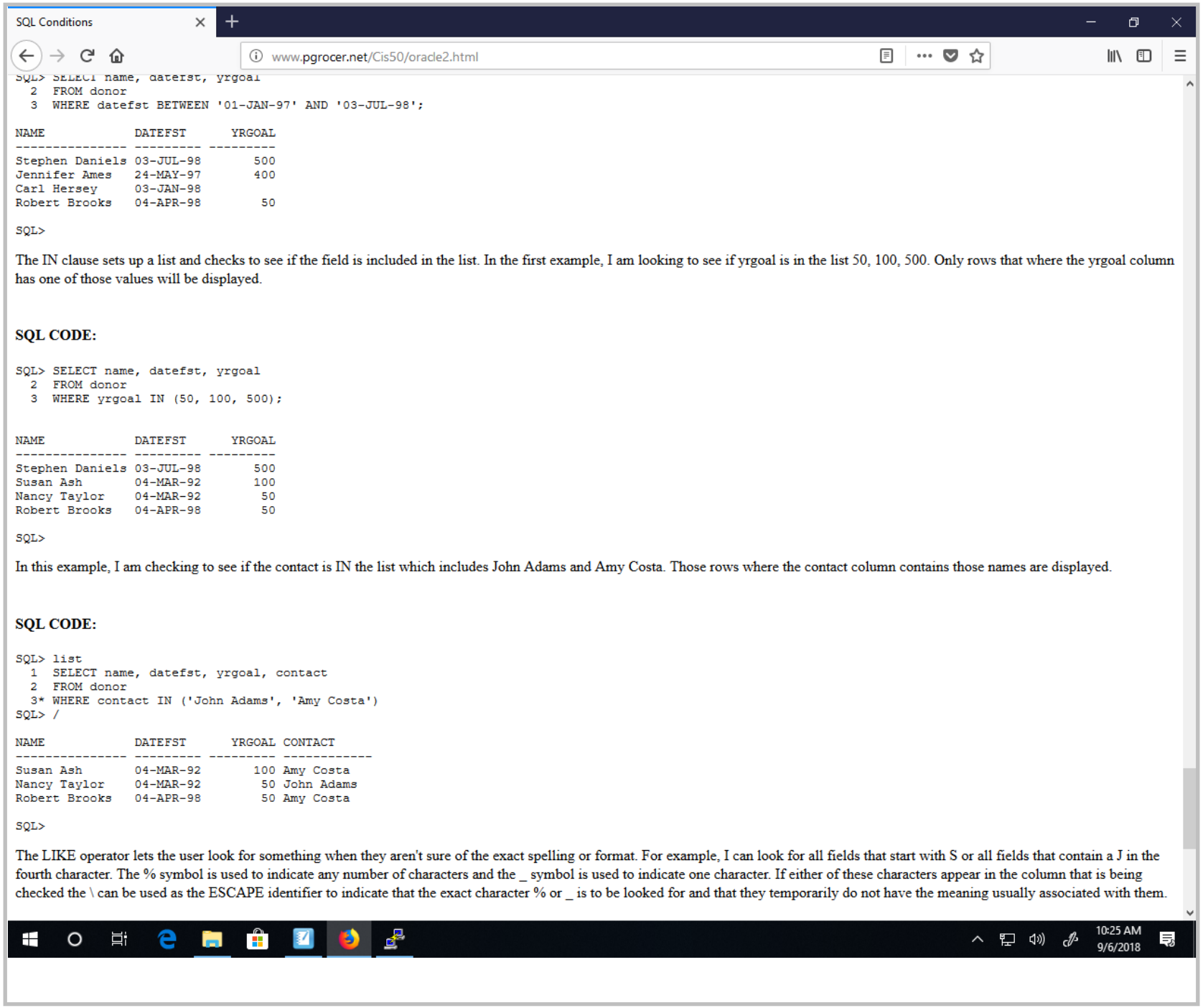
SQL CODE:

```
SELECT name, city, state, datefst, yrgoal, contact
FROM donor
WHERE state = 'RI' AND datefst > '01-JAN-98' OR state = 'MA' AND datefst > '01-JAN-96';

NAME            CITY       ST DATEFST      YRGOAL CONTACT
--------------- ---------- -- --------- ------- ------------
Stephen Daniels Seekonk    MA 03-JUL-98       500 John Smith
Carl Hersey     Providence RI 03-JAN-98           Susan Jones
Robert Brooks   Fall River MA 04-APR-98        50 Amy Costa
```


SQL Conditions
www.pgrocer.net/Cis50/oracle2.html
SQL> SELECT name, state, contact
2 FROM donor
3 WHERE NOT contact = 'Amy Costa';
NAME ST CONTACT
Stephen Daniels MA John Smith
Jennifer Ames RI Susan Jones
Carl Hersey RI Susan Jones
Nancy Taylor MA John Adams
Another example of the use of NOT is in this compound AND statement where each of the clauses contain a NOT. In this case I want all donors who do not have Amy Costa as their contact and do not live in the state of Rhode Island. The NOT has priority, so the NOT gets resolved first and then the AND combines the two NOT conditions to say not Amy Costa and not Rhode Island. This means neither Amy Costa nor Rhode Island can appear in the row. Another way to say it is, anyone who is not Amy Costa and any state that is not Rhode Island.
SQL CODE:
SQL> list
1 SELECT name, state, yrgoal, contact
2 FROM donor
3* WHERE NOT contact = 'Amy Costa' AND NOT state = 'RI'
SQL> /
NAME ST YRGOAL CONTACT
Stephen Daniels MA 500 John Smith
Nancy Taylor MA 50 John Adams
SQL>
In the next example, the AND was put inside parenthesis so it got resolved first. The NOT got applied to this resolution. The final result was all rows that did not contain Amy Costa AND Rhode Island. Since there are no records for Amy Costa and Rhode Island all the records are displayed.
SQL CODE:
SQL> SELECT name, state, yrgoal, contact
2 FROM donor
3 WHERE NOT(contact = 'Amy Costa' AND state = 'RI');
NAME ST YRGOAL CONTACT
Stephen Daniels MA 500 John Smith
Jennifer Ames RI 400 Susan Jones
Carl Hersey RI Susan Jones
Susan Ash MA 100 Amy Costa
Nancy Taylor MA 50 John Adams
Robert Brooks MA 50 Amy Costa
6 rows selected.
SQL>
To help clarify this, I am now asking for all rows that did not contain Amy Costa and Massachusetts. There were two rows for Massachusetts that had Amy Costa as the contact person and those
10:22 AM
9/6/2018


SQL Conditions
www.pgrocer.net/Cis50/oracle2.html
SQL CODE:
SQL> SELECT name, state, yrgoal, contact
2 FROM donor
3 WHERE NOT (contact = 'Amy Costa' AND state = 'MA');
NAME ST YRGOAL CONTACT
Stephen Daniels MA 500 John Smith
Jennifer Ames RI 400 Susan Jones
Carl Hersey RI Susan Jones
Nancy Taylor MA 50 John Adams
SQL>
OTHER OPERATORS: BETWEEN, IN, LIKE
Oracle SQL provides several other operators that can be used to retrieve specific records. The BETWEEN and the IN simplify asking for several things. BETWEEN deals with a range and IN deals with a list.
In the example below, I want to display all rows where the datefst column contains a date between January 1, 1997 and July 3, 1998. Notice that July 2, 1998 appears on the list which tells us that the BETWEEN is inclusive meaning it includes the two dates that are the beginning and end of the range.
SQL CODE:
SQL> SELECT name, datefst, yrgoal
2 FROM donor
3 WHERE datefst BETWEEN '01-JAN-97' AND '03-JUL-98';
NAME DATEFST YRGOAL
Stephen Daniels 03-JUL-98 500
Jennifer Ames 24-MAY-97 400
Carl Hersey 03-JAN-98
Robert Brooks 04-APR-98 50
SQL>
The IN clause sets up a list and checks to see if the field is included in the list. In the first example, I am looking to see if yrgoal is in the list 50, 100, 500. Only rows that where the yrgoal column has one of those values will be displayed.
SQL CODE:
SQL> SELECT name, datefst, yrgoal
2 FROM donor
3 WHERE yrgoal IN (50, 100, 500);
NAME DATEFST YRGOAL
Stephen Daniels 03-JUL-98 500
10:24 AM
9/6/2018




```
SQL> SELECT name, datefst, yrgoal
  2  FROM donor
  3  WHERE datefst BETWEEN '01-JAN-97' AND '03-JUL-98';

NAME            DATEFST      YRGOAL
--------------- --------- ---------
Stephen Daniels 03-JUL-98        500
Jennifer Ames   24-MAY-97        400
Carl Hersey     03-JAN-98
Robert Brooks   04-APR-98         50

SQL>
```

The IN clause sets up a list and checks to see if the field is included in the list. In the first example, I am looking to see if yrgoal is in the list 50, 100, 500. Only rows that where the yrgoal column has one of those values will be displayed.

SQL CODE:

```
SQL> SELECT name, datefst, yrgoal
  2  FROM donor
  3  WHERE yrgoal IN (50, 100, 500);


NAME            DATEFST      YRGOAL
--------------- --------- ---------
Stephen Daniels 03-JUL-98        500
Susan Ash       04-MAR-92        100
Nancy Taylor    04-MAR-92         50
Robert Brooks   04-APR-98         50

SQL>
```

In this example, I am checking to see if the contact is IN the list which includes John Adams and Amy Costa. Those rows where the contact column contains those names are displayed.

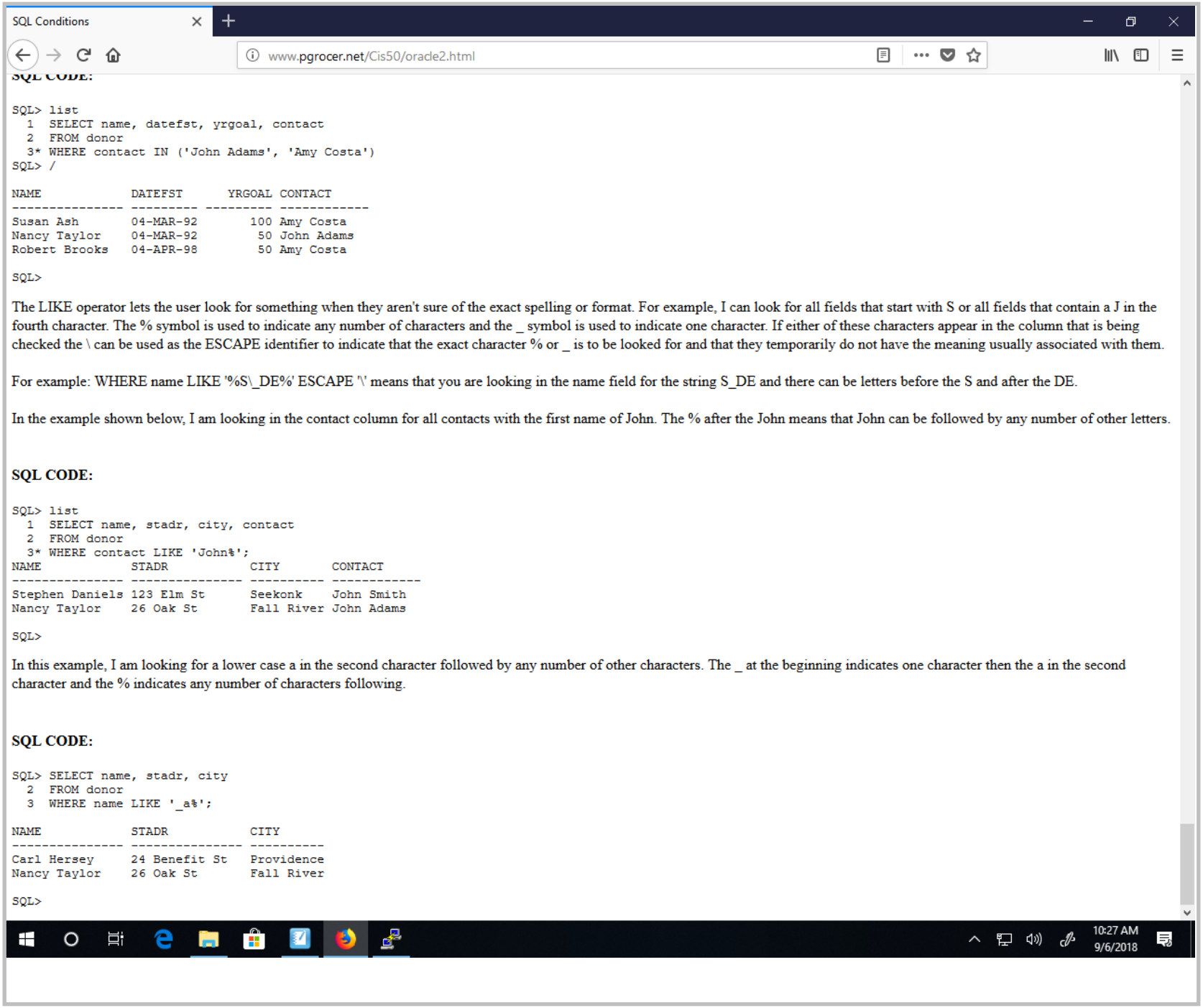
SQL CODE:

```
SQL> list
  1  SELECT name, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE contact IN ('John Adams', 'Amy Costa')
SQL> /

NAME            DATEFST      YRGOAL CONTACT
--------------- --------- --------- ------------
Susan Ash       04-MAR-92        100 Amy Costa
Nancy Taylor    04-MAR-92         50 John Adams
Robert Brooks   04-APR-98         50 Amy Costa

SQL>
```

The LIKE operator lets the user look for something when they aren't sure of the exact spelling or format. For example, I can look for all fields that start with S or all fields that contain a J in the fourth character. The % symbol is used to indicate any number of characters and the _ symbol is used to indicate one character. If either of these characters appear in the column that is being checked the \ can be used as the ESCAPE identifier to indicate that the exact character % or _ is to be looked for and that they temporarily do not have the meaning usually associated with them.




```
SQL> list
  1  SELECT name, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE contact IN ('John Adams', 'Amy Costa')
SQL> /

NAME            DATEFST      YRGOAL CONTACT
--------------- --------- --------- ------------
Susan Ash       04-MAR-92        100 Amy Costa
Nancy Taylor    04-MAR-92         50 John Adams
Robert Brooks   04-APR-98         50 Amy Costa

SQL>
```

The LIKE operator lets the user look for something when they aren't sure of the exact spelling or format. For example, I can look for all fields that start with S or all fields that contain a J in the fourth character. The % symbol is used to indicate any number of characters and the _ symbol is used to indicate one character. If either of these characters appear in the column that is being checked the \ can be used as the ESCAPE identifier to indicate that the exact character % or _ is to be looked for and that they temporarily do not have the meaning usually associated with them.

For example: WHERE name LIKE '%S_DE%' ESCAPE '\' means that you are looking in the name field for the string S_DE and there can be letters before the S and after the DE.

In the example shown below, I am looking in the contact column for all contacts with the first name of John. The % after the John means that John can be followed by any number of other letters.

SQL CODE:

```
SQL> list
  1  SELECT name, stadr, city, contact
  2  FROM donor
  3* WHERE contact LIKE 'John%';
NAME            STADR          CITY       CONTACT
--------------- -------------- ---------- ------------
Stephen Daniels 123 Elm St     Seekonk    John Smith
Nancy Taylor    26 Oak St      Fall River John Adams

SQL>
```

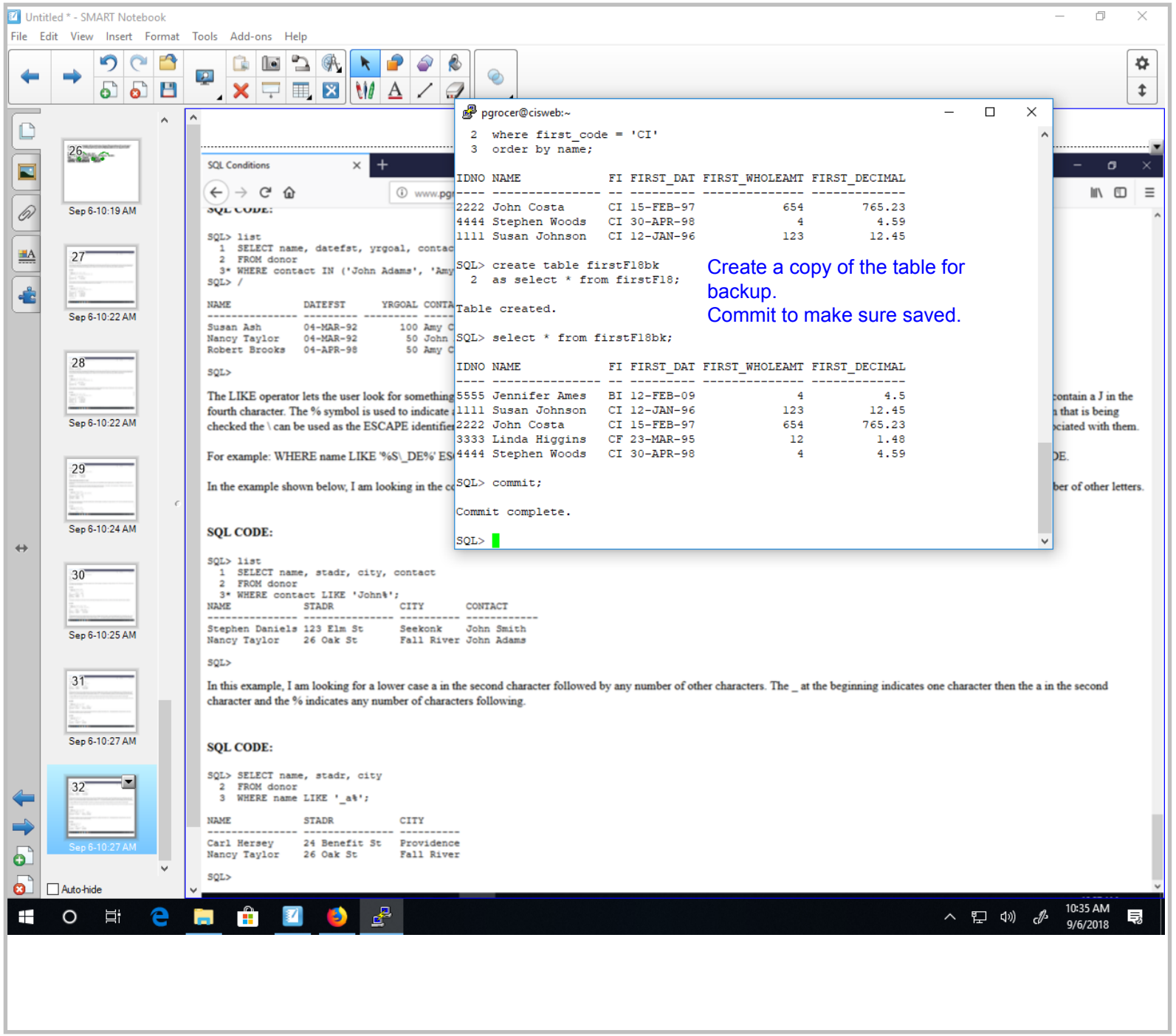
In this example, I am looking for a lower case a in the second character followed by any number of other characters. The _ at the beginning indicates one character then the a in the second character and the % indicates any number of characters following.

SQL CODE:

```
SQL> SELECT name, stadr, city
  2  FROM donor
  3  WHERE name LIKE '_a%';

NAME            STADR          CITY
--------------- -------------- ----------
Carl Hersey     24 Benefit St  Providence
Nancy Taylor    26 Oak St      Fall River

SQL>
```



Title: Sep 6-10:35 AM (28 of 28)