Introduction to SQL

Greg Ridgeway (gridge@upenn.edu)
Ruth Moyer (moyruth@upenn.edu)

July 31, 2018

The crime table in our Chicago crime database is not ideal. We're probably already unhappy about those dots in the column names. Also, it was overly complicated to extract the year from a date. There is also a lot of redundant information in the table.

Let's take a look at a few example rows.

Block	IUCR	Primary.Type	FBI.Code	X	Y
040XX W 26TH ST	0560	ASSAULT	08A	1150052	1886384
089XX S SOUTH CHICAGO AVE	0498	BATTERY	04B	1195182	1846473
052XX S HARPER AVE	2820	OTHER OFFENSE	26	1187140	1870924
033XX N TROY ST	2825	OTHER OFFENSE	26	1154836	1921848
015XX W 107TH ST	1310	CRIMINAL DAMAGE	14	1167706	1833732
0000X N LARAMIE AVE	2018	NARCOTICS	18	1141668	1900044
0000X N KEELER AVE	0554	ASSAULT	08A	1148347	1899920
026XX N ELSTON AVE	0560	ASSAULT	08A	1160777	1917685
076XX S ABERDEEN ST	0486	BATTERY	08B	1170265	1854235
3XX N SHEFFIELD AVE	1811	NARCOTICS	18	1169197	1915770

Note that whenever `IUCR` is 0560, then `Primary.Type` is ASSAULT and `FBI.Code` is 08A. There is no reason to store the IUCR code, the primary crime type, and the FBI code all in the same file. We should keep a separate table that links the IUCR codes, the primary crime types, and the FBI codes, but that we can keep separately. Note that it is essential to store the IUCR code in the crime table. Both IUR codes 2018 and 1811 both link to NARCOTICS and FBI code 18. If we deleted IUCR from the crime table and kept only the primary crime type, then we would lose some detailed information.

Let's start by reconnecting to the Chicago crime database

```
library(sqldf)
```

```
Loading required package: gsubfn
```

```
Loading required package: proto
```

```
Loading required package: RSQLite
```

```
con <- dbConnect(SQLite(), "chicagocrime.db")
```

The SQL keyword `DISTINCT` will filter out any duplicated rows in the result set so that every row is a unique combination of values.

```
res <- dbSendQuery(con, "
                    SELECT DISTINCT IUCR, [Primary.Type], [FBI.code]
```

```
                    FROM crime")
a <- fetch(res, n = -1)
print(a[1:5,])
dbClearResult(res)
```

```
  IUCR  Primary.Type FBI.Code
1 041A       BATTERY      04B
2 4625 OTHER OFFENSE       26
3 0486       BATTERY      08B
4 0460       BATTERY      08B
5 031A       ROBBERY       03
```

This creates a lookup table showing how IUCR links to the primary crime types and FBI codes. We should check that each IUCR code uniquely links to the primary type and FBI codes.

```
b <- table(a$IUCR)
print(b[b>1]) # do any show up more than once?
```

```
1030 1035 1710 1715 1725 1755 1780 2091 2092 2093 5114
   2    2    2    2    2    2    2    2    2    2    2
```

Let's start by examining codes 2091, 2092, and 2093.

```
res <- dbSendQuery(con, "SELECT COUNT(*) AS crimecount,
                                IUCR,
                                [Primary.Type],
                                [FBI.Code],
                                SUBSTR(Date,7,4) AS year
                         FROM crime
                         WHERE IUCR='2091' OR IUCR='2092' OR IUCR='2093'
                         GROUP BY IUCR,[Primary.Type],[FBI.Code],year
                         ORDER BY IUCR,[Primary.Type],[FBI.Code],year")
fetch(res, n = -1)
dbClearResult(res)
```

```
   crimecount IUCR Primary.Type FBI.Code year
1           1 2091    NARCOTICS       18 2012
2           2 2091    NARCOTICS       18 2013
3          26 2091    NARCOTICS       18 2014
4         135 2091    NARCOTICS       18 2015
5         118 2091    NARCOTICS       18 2016
6         112 2091    NARCOTICS       18 2017
7          41 2091    NARCOTICS       18 2018
8         389 2091    NARCOTICS       26 2001
9         267 2091    NARCOTICS       26 2002
10        238 2091    NARCOTICS       26 2003
11        288 2091    NARCOTICS       26 2004
12        253 2091    NARCOTICS       26 2005
13        232 2091    NARCOTICS       26 2006
```

14	221	2091	NARCOTICS	26	2007
15	225	2091	NARCOTICS	26	2008
16	246	2091	NARCOTICS	26	2009
17	208	2091	NARCOTICS	26	2010
18	178	2091	NARCOTICS	26	2011
19	205	2091	NARCOTICS	26	2012
20	195	2091	NARCOTICS	26	2013
21	166	2091	NARCOTICS	26	2014
22	28	2091	NARCOTICS	26	2015
23	126	2092	NARCOTICS	18	2015
24	212	2092	NARCOTICS	18	2016
25	373	2092	NARCOTICS	18	2017
26	300	2092	NARCOTICS	18	2018
27	1675	2092	NARCOTICS	26	2001
28	2373	2092	NARCOTICS	26	2002
29	2775	2092	NARCOTICS	26	2003
30	3094	2092	NARCOTICS	26	2004
31	3130	2092	NARCOTICS	26	2005
32	3049	2092	NARCOTICS	26	2006
33	2726	2092	NARCOTICS	26	2007
34	1523	2092	NARCOTICS	26	2008
35	1435	2092	NARCOTICS	26	2009
36	1056	2092	NARCOTICS	26	2010
37	767	2092	NARCOTICS	26	2011
38	672	2092	NARCOTICS	26	2012
39	679	2092	NARCOTICS	26	2013
40	542	2092	NARCOTICS	26	2014
41	237	2092	NARCOTICS	26	2015
42	1	2093	NARCOTICS	18	2008
43	1	2093	NARCOTICS	18	2010
44	2	2093	NARCOTICS	18	2011
45	16	2093	NARCOTICS	18	2012
46	16	2093	NARCOTICS	18	2013
47	15	2093	NARCOTICS	18	2014
48	321	2093	NARCOTICS	18	2015
49	842	2093	NARCOTICS	18	2016
50	994	2093	NARCOTICS	18	2017
51	520	2093	NARCOTICS	18	2018
52	972	2093	NARCOTICS	26	2001
53	866	2093	NARCOTICS	26	2002
54	968	2093	NARCOTICS	26	2003
55	864	2093	NARCOTICS	26	2004
56	839	2093	NARCOTICS	26	2005
57	909	2093	NARCOTICS	26	2006
58	1033	2093	NARCOTICS	26	2007
59	1209	2093	NARCOTICS	26	2008
60	1100	2093	NARCOTICS	26	2009
61	1017	2093	NARCOTICS	26	2010

```
62        934 2093    NARCOTICS       26 2011
63        935 2093    NARCOTICS       26 2012
64        760 2093    NARCOTICS       26 2013
65        676 2093    NARCOTICS       26 2014
66        332 2093    NARCOTICS       26 2015
```

These are all narcotics cases, but we see that in some years, these charges are marked as FBI code 18 (crimes of production, sale, use of drugs) and sometimes 26 (a miscellaneous category) (see CPD's crime type description). FBI code 26 appears more commonly, but the FBI code 26 appears to phase out after 2015. 2091 is a narcotics code for "forfeit property", 2092 is for "soliciting narcotics on a publicway," and 2093 is for "found suspect narcotics." It appears that CPD now interprets these crimes as being drug crimes. We'll just use code 18 for these crimes.

A similar story goes for IUCR crimes 1710, 1715, 1725, 1755, and 1780. These are all offenses involving children that prior to 2016 had been given the FBI miscellaneous code 26, but more recently has been coded as 20 (offenses against family). We'll code these using the more recent FBI code 20.

```
res <- dbSendQuery(con,
   "SELECT COUNT(*) AS crimecount,
           IUCR,
           [Primary.Type],
           [FBI.Code],
           SUBSTR(Date,7,4) AS year
    FROM crime
    WHERE IUCR IN ('1710','1715','1725','1755','1780')
    GROUP BY IUCR, [Primary.Type], [FBI.Code], year
    ORDER BY IUCR, [Primary.Type], [FBI.Code], year")
fetch(res, n = -1)
dbClearResult(res)
```

```
   crimecount IUCR               Primary.Type FBI.Code year
1           5 1710 OFFENSE INVOLVING CHILDREN       20 2013
2           1 1710 OFFENSE INVOLVING CHILDREN       20 2014
3          21 1710 OFFENSE INVOLVING CHILDREN       20 2015
4         275 1710 OFFENSE INVOLVING CHILDREN       20 2016
5         327 1710 OFFENSE INVOLVING CHILDREN       20 2017
6         182 1710 OFFENSE INVOLVING CHILDREN       20 2018
7         503 1710 OFFENSE INVOLVING CHILDREN       26 2001
8         506 1710 OFFENSE INVOLVING CHILDREN       26 2002
9         479 1710 OFFENSE INVOLVING CHILDREN       26 2003
10        427 1710 OFFENSE INVOLVING CHILDREN       26 2004
11        413 1710 OFFENSE INVOLVING CHILDREN       26 2005
12        392 1710 OFFENSE INVOLVING CHILDREN       26 2006
13        403 1710 OFFENSE INVOLVING CHILDREN       26 2007
14        337 1710 OFFENSE INVOLVING CHILDREN       26 2008
15        374 1710 OFFENSE INVOLVING CHILDREN       26 2009
16        362 1710 OFFENSE INVOLVING CHILDREN       26 2010
17        332 1710 OFFENSE INVOLVING CHILDREN       26 2011
```

18	334	1710	OFFENSE INVOLVING CHILDREN	26	2012
19	272	1710	OFFENSE INVOLVING CHILDREN	26	2013
20	315	1710	OFFENSE INVOLVING CHILDREN	26	2014
21	267	1710	OFFENSE INVOLVING CHILDREN	26	2015
22	8	1710	OFFENSE INVOLVING CHILDREN	26	2016
23	1	1715	OFFENSE INVOLVING CHILDREN	20	2016
24	1	1715	OFFENSE INVOLVING CHILDREN	20	2017
25	1	1715	OFFENSE INVOLVING CHILDREN	20	2018
26	4	1715	OFFENSE INVOLVING CHILDREN	26	2003
27	1	1715	OFFENSE INVOLVING CHILDREN	26	2006
28	1	1715	OFFENSE INVOLVING CHILDREN	26	2007
29	3	1715	OFFENSE INVOLVING CHILDREN	26	2008
30	2	1715	OFFENSE INVOLVING CHILDREN	26	2009
31	3	1715	OFFENSE INVOLVING CHILDREN	26	2010
32	2	1715	OFFENSE INVOLVING CHILDREN	26	2011
33	4	1715	OFFENSE INVOLVING CHILDREN	26	2012
34	1	1715	OFFENSE INVOLVING CHILDREN	26	2013
35	1	1715	OFFENSE INVOLVING CHILDREN	26	2015
36	2	1725	OFFENSE INVOLVING CHILDREN	20	2014
37	1	1725	OFFENSE INVOLVING CHILDREN	20	2015
38	4	1725	OFFENSE INVOLVING CHILDREN	20	2016
39	15	1725	OFFENSE INVOLVING CHILDREN	20	2017
40	4	1725	OFFENSE INVOLVING CHILDREN	20	2018
41	2	1725	OFFENSE INVOLVING CHILDREN	26	2002
42	4	1725	OFFENSE INVOLVING CHILDREN	26	2003
43	1	1725	OFFENSE INVOLVING CHILDREN	26	2004
44	5	1725	OFFENSE INVOLVING CHILDREN	26	2005
45	4	1725	OFFENSE INVOLVING CHILDREN	26	2006
46	9	1725	OFFENSE INVOLVING CHILDREN	26	2007
47	4	1725	OFFENSE INVOLVING CHILDREN	26	2008
48	3	1725	OFFENSE INVOLVING CHILDREN	26	2009
49	16	1725	OFFENSE INVOLVING CHILDREN	26	2010
50	9	1725	OFFENSE INVOLVING CHILDREN	26	2011
51	7	1725	OFFENSE INVOLVING CHILDREN	26	2012
52	12	1725	OFFENSE INVOLVING CHILDREN	26	2013
53	12	1725	OFFENSE INVOLVING CHILDREN	26	2014
54	9	1725	OFFENSE INVOLVING CHILDREN	26	2015
55	2	1755	OFFENSE INVOLVING CHILDREN	20	2015
56	32	1755	OFFENSE INVOLVING CHILDREN	20	2016
57	45	1755	OFFENSE INVOLVING CHILDREN	20	2017
58	15	1755	OFFENSE INVOLVING CHILDREN	20	2018
59	37	1755	OFFENSE INVOLVING CHILDREN	26	2002
60	75	1755	OFFENSE INVOLVING CHILDREN	26	2003
61	69	1755	OFFENSE INVOLVING CHILDREN	26	2004
62	64	1755	OFFENSE INVOLVING CHILDREN	26	2005
63	70	1755	OFFENSE INVOLVING CHILDREN	26	2006
64	59	1755	OFFENSE INVOLVING CHILDREN	26	2007
65	49	1755	OFFENSE INVOLVING CHILDREN	26	2008

```
66         34 1755 OFFENSE INVOLVING CHILDREN       26 2009
67         52 1755 OFFENSE INVOLVING CHILDREN       26 2010
68         52 1755 OFFENSE INVOLVING CHILDREN       26 2011
69         39 1755 OFFENSE INVOLVING CHILDREN       26 2012
70         49 1755 OFFENSE INVOLVING CHILDREN       26 2013
71         43 1755 OFFENSE INVOLVING CHILDREN       26 2014
72         34 1755 OFFENSE INVOLVING CHILDREN       26 2015
73          1 1780 OFFENSE INVOLVING CHILDREN       20 2010
74          1 1780 OFFENSE INVOLVING CHILDREN       20 2011
75          2 1780 OFFENSE INVOLVING CHILDREN       20 2012
76         14 1780 OFFENSE INVOLVING CHILDREN       20 2015
77        539 1780 OFFENSE INVOLVING CHILDREN       20 2016
78        414 1780 OFFENSE INVOLVING CHILDREN       20 2017
79        232 1780 OFFENSE INVOLVING CHILDREN       20 2018
80         11 1780 OFFENSE INVOLVING CHILDREN       26 2001
81        166 1780 OFFENSE INVOLVING CHILDREN       26 2002
82        352 1780 OFFENSE INVOLVING CHILDREN       26 2003
83        559 1780 OFFENSE INVOLVING CHILDREN       26 2004
84        465 1780 OFFENSE INVOLVING CHILDREN       26 2005
85        504 1780 OFFENSE INVOLVING CHILDREN       26 2006
86        613 1780 OFFENSE INVOLVING CHILDREN       26 2007
87        624 1780 OFFENSE INVOLVING CHILDREN       26 2008
88        658 1780 OFFENSE INVOLVING CHILDREN       26 2009
89        617 1780 OFFENSE INVOLVING CHILDREN       26 2010
90        649 1780 OFFENSE INVOLVING CHILDREN       26 2011
91        628 1780 OFFENSE INVOLVING CHILDREN       26 2012
92        628 1780 OFFENSE INVOLVING CHILDREN       26 2013
93        609 1780 OFFENSE INVOLVING CHILDREN       26 2014
94        519 1780 OFFENSE INVOLVING CHILDREN       26 2015
95         38 1780 OFFENSE INVOLVING CHILDREN       26 2016
```

IUCR codes 1030 and 1035, which involve possession of incendiary devices, are now being coded as arson rather than miscellaneous.

```
res <- dbSendQuery(con,
   "SELECT COUNT(*) AS crimecount,
           IUCR,
           [Primary.Type],
           [FBI.Code],
           SUBSTR(Date,7,4) AS year
    FROM crime
    WHERE IUCR='1030' OR IUCR='1035'
    GROUP BY IUCR, [Primary.Type], [FBI.Code], year
    ORDER BY IUCR, [Primary.Type], [FBI.Code], year")
fetch(res, n = -1)
dbClearResult(res)
```

```
   crimecount IUCR Primary.Type FBI.Code year
1           2 1030        ARSON       09 2016
```

```
2           3 1030        ARSON       09 2017
3           1 1030        ARSON       09 2018
4           6 1030        ARSON       26 2001
5           2 1030        ARSON       26 2002
6           5 1030        ARSON       26 2003
7           4 1030        ARSON       26 2004
8           3 1030        ARSON       26 2005
9           7 1030        ARSON       26 2006
10          5 1030        ARSON       26 2007
11          7 1030        ARSON       26 2008
12          5 1030        ARSON       26 2009
13          9 1030        ARSON       26 2010
14          5 1030        ARSON       26 2011
15          2 1030        ARSON       26 2012
16          6 1030        ARSON       26 2013
17          2 1030        ARSON       26 2014
18          5 1030        ARSON       26 2015
19          1 1030        ARSON       26 2016
20          1 1035        ARSON       09 2016
21          7 1035        ARSON       26 2002
22          2 1035        ARSON       26 2004
23          3 1035        ARSON       26 2005
24          8 1035        ARSON       26 2006
25          6 1035        ARSON       26 2007
26          6 1035        ARSON       26 2008
27          4 1035        ARSON       26 2009
28          1 1035        ARSON       26 2010
29          1 1035        ARSON       26 2011
30          1 1035        ARSON       26 2012
```

Lastly, note that the spelling of the primary type for 5114 has changed to remove the extra spaces. Even though they differ only by a few spaces, SQL will conclude that these are different values.

```
res <- dbSendQuery(con,
   "SELECT COUNT(*) AS crimecount,
           IUCR,
           [Primary.Type],
           [FBI.Code],
           SUBSTR(Date,7,4) AS year
    FROM crime
    WHERE IUCR='5114'
    GROUP BY IUCR, [Primary.Type], [FBI.Code], year")
fetch(res, n = -1)
dbClearResult(res)
```

```
  crimecount IUCR   Primary.Type FBI.Code year
1          3 5114 NON - CRIMINAL       26 2013
2         10 5114 NON - CRIMINAL       26 2014
3         20 5114 NON - CRIMINAL       26 2015
```

```
4          5 5114 NON - CRIMINAL       26 2016
5          1 5114   NON-CRIMINAL       26 2015
6         14 5114   NON-CRIMINAL       26 2016
7          7 5114   NON-CRIMINAL       26 2017
8          9 5114   NON-CRIMINAL       26 2018
```

With questions about IUCR to FBI codes resolved, let's create the IUCR, primary type, and FBI code lookup table in our Chicago crime database. We can use `dbWriteTable()` to post our data frame `a` to the database creating a new table called `iucr`.

```
if(dbExistsTable(con,"iucr")) dbRemoveTable(con, "iucr")
# import the data frame into SQLite
dbWriteTable(con, "iucr", a,
             row.names=FALSE)
dbListFields(con,"iucr")
```

```
[1] "IUCR"         "Primary.Type" "FBI.Code"
```

Check whether the table looks correct.

```
fetch(dbSendQuery(con, "SELECT * FROM iucr LIMIT 5"))
```

```
  IUCR  Primary.Type FBI.Code
1 041A       BATTERY      04B
2 4625 OTHER OFFENSE       26
3 0486       BATTERY      08B
4 0460       BATTERY      08B
5 031A       ROBBERY       03
```

Everything looks okay. However, the dots in the column names are rather tiresome. We should take this opportunity to give them names that are more appropriate for working in SQL. Also, there was no need to pull the table into R, only to post it right back into the database. We can use a `CREATE TABLE` clause to create this lookup table instead.

```
if(dbExistsTable(con,"iucr")) dbRemoveTable(con, "iucr")
res <- dbSendQuery(con, "
                  CREATE TABLE iucr AS
                  SELECT DISTINCT IUCR as iucr,
                                  [Primary.Type] AS PrimaryType,
                                  [FBI.Code] AS FBIcode
                  FROM crime")
```

To finalize the lookup table, we just need to clear out those rows that we do not want, those involving FBI code 26 and the removal of the spaces for IUCR 5114.

```
res <- dbSendQuery(con, "
   DELETE FROM iucr
   WHERE (FBICode='26') AND
         (iucr IN ('1030','1035',
                   '1710','1715','1725','1755','1780',
                   '2091','2092','2093'))")
res <- dbSendQuery(con, "
```

```
                    DELETE FROM iucr
                    WHERE (iucr='5114') AND
                          (PrimaryType='NON - CRIMINAL')")
```

We now see that our database has two tables, the original `crime` table and the new `iucr` lookup table.

```
dbListTables(con)
```

```
[1] "crime" "iucr"
```

Exercises

With the new table `iucr` in the database complete the following exercises.

1. Print out all of the rows in iucr
2. Print out all the IUCR codes for "KIDNAPPING"
3. How many IUCR codes are there for "ASSAULT"?
4. Try doing the prior exercise again using COUNT(*) if you did not use it the first time

SQL dates

SQLite has no special data/time data type. The `Date` column is currently stored in the `crime` table as text. The `PRAGMA` statement is a way to modify or query the SQLite database itself. Here we can ask SQLite the data types it is using to store each of the columns. All the entries, including `Date` are stored as text, integers, or doubles (numbers with decimal points).

```
res <- dbSendQuery(con, "PRAGMA table_info(crime)")
fetch(res, n = -1)
dbClearResult(res)
```

```
   cid                 name   type notnull dflt_value pk
1    0                   ID    INT       0         NA  0
2    1          Case.Number   TEXT       0         NA  0
3    2                 Date   TEXT       0         NA  0
4    3                Block   TEXT       0         NA  0
5    4                 IUCR   TEXT       0         NA  0
6    5         Primary.Type   TEXT       0         NA  0
7    6          Description   TEXT       0         NA  0
8    7 Location.Description   TEXT       0         NA  0
9    8               Arrest   TEXT       0         NA  0
10   9             Domestic   TEXT       0         NA  0
11  10                 Beat    INT       0         NA  0
12  11             District   TEXT       0         NA  0
13  12                 Ward   TEXT       0         NA  0
```

```
14  13       Community.Area   TEXT       0         NA  0
15  14             FBI.Code   TEXT       0         NA  0
16  15         X.Coordinate    INT       0         NA  0
17  16         Y.Coordinate    INT       0         NA  0
18  17                 Year    INT       0         NA  0
19  18           Updated.On   TEXT       0         NA  0
20  19             Latitude DOUBLE       0         NA  0
21  20            Longitude DOUBLE       0         NA  0
22  21             Location   TEXT       0         NA  0
```

The standard date format in computing is yyyy-mm-dd hh:mm:ss, where the hours are on the 24-hour clock (so no AM/PM). The reason for this format is that you can sort the data in this format to get events in order. For some reason, the producers of the Chicago crime dataset did not use this standard format. If you sort events in the current database then all the January events will come first (regardless in what year they occurred) and any events occurring at 1pm will show up before those occurring at 2am. Putting the dates in a standard format also allows us to use some useful SQLite date functions for extracting the year, day of the week, time of day, and other features of the date and time.

The plan is to create a dataframe in R with each crime's `ID` and `Date`. Then we will use `lubridate` to clean up the dates and put them in the standard format. Then we will push a new table into the database containing each crime's `ID` and its newly formatted date.

```
library(lubridate)
```

```
Attaching package: 'lubridate'
```

```
The following object is masked from 'package:base':

    date
```

```
res <- dbSendQuery(con, "SELECT ID, Date FROM crime")
data <- fetch(res, n = -1)
dbClearResult(res)
data[1:5,]
```

```
        ID                   Date
1 10000092 03/18/2015 07:44:00 PM
2 10000094 03/18/2015 11:00:00 PM
3 10000095 03/18/2015 10:45:00 PM
4 10000096 03/18/2015 10:30:00 PM
5 10000097 03/18/2015 09:00:00 PM
```

Since the dates are in mm/dd/yyyy hh:mm:ss format, we will use `mdy_hms()` from the `lubridate` package to clean these up. Fortunately, this function can also handle the AM/PM.

```
data$datefix <- mdy_hms(data$Date)
# convert to plain text
data$datefix <- as.character(data$datefix)
# check that the reformatting worked
data[1:5,]
```

```
# delete the original date from the data frame
data$Date <- NULL
```

```
        ID                    Date             datefix
1 10000092 03/18/2015 07:44:00 PM 2015-03-18 19:44:00
2 10000094 03/18/2015 11:00:00 PM 2015-03-18 23:00:00
3 10000095 03/18/2015 10:45:00 PM 2015-03-18 22:45:00
4 10000096 03/18/2015 10:30:00 PM 2015-03-18 22:30:00
5 10000097 03/18/2015 09:00:00 PM 2015-03-18 21:00:00
```

With the dates in standard format, let's push the fixed dates table to the database.

```
# remove DateFix table if it already exists
if(dbExistsTable(con,"DateFix")) dbRemoveTable(con, "DateFix")
# save a table with ID and the properly formatted date
dbWriteTable(con, "DateFix", data, row.names=FALSE)
dbListTables(con)
```

```
[1] "DateFix" "crime"   "iucr"
```

We now see that our database now has three tables with the addition of the new `DateFix` table.

Before we used `SUBSTR()` to extract the year from the date. That was not very elegant and required figuring out which characters held the four characters representing the year. Even though SQLite does not have a date/time type, it does have some functions that help us work with dates. We will use SQLite's `STRFTIME()` function. It stands for "string format time". It is a decades old function that you will find in almost all languages. Even R has its own version of `strftime()`. The `STRFTIME()` function has two parameters. The first is a format parameter in which you tell `STRFTIME()` what you want it to extract from the date and the second parameter is the column containing the dates. There are a lot of options for the format parameter. For example, you can extract just the year (%Y), just the month (%m), just the minute (%M), the day of the week (%w) with Sunday represented as 0 and Saturday as 6, or the week of the year (%W). You can also combine to get, for example, the year and month (%Y-%m). You can find a complete listing here.

Let's write a query to test out `STRFTIME()`. Here we will select some dates from `DateFix` and determine on which day of the week the crime occurred.

```
res <- dbSendQuery(con, "
                   SELECT ID,
                          datefix,
                          STRFTIME('%w',datefix) AS weekday
                   FROM DateFix")
a <- fetch(res, n = 10)
a
dbClearResult(res)
```

```
         ID             datefix weekday
1  10000092 2015-03-18 19:44:00       3
2  10000094 2015-03-18 23:00:00       3
3  10000095 2015-03-18 22:45:00       3
4  10000096 2015-03-18 22:30:00       3
```

```
5  10000097 2015-03-18 21:00:00       3
6  10000098 2015-03-18 22:00:00       3
7  10000099 2015-03-18 23:00:00       3
8  10000100 2015-03-18 21:35:00       3
9  10000101 2015-03-18 22:09:00       3
10 10000104 2015-03-18 21:25:00       3
```

For the first date, 2015-03-18, `STRFTIME()` tells us that this was day 3 of the week, which is Wednesday.

`STRFTIME()` always returns values that are text. That is, if you ask for the year using `STRFTIME('%Y',datefix)` and you get values like 2017 and 2018, your results will be character strings rather than numeric. You will have to convert them using `as.numeric()` in R or, preferably, using a `CAST()` expression in SQL. `CAST()` is particularly useful if you want to select records that, say, occur after 2010 or after noon.

Let's count cases that occurred between Monday and Friday.

```
res <- dbSendQuery(con, "
                   SELECT COUNT(*),
                          CAST(STRFTIME('%w',datefix) AS INTEGER) AS weekday
                   FROM DateFix
                   WHERE (weekday>=1) AND (weekday<=5)
                   GROUP BY weekday")
fetch(res, n = 10)
dbClearResult(res)
```

```
  COUNT(*) weekday
1   939220       1
2   954750       2
3   960458       3
4   950447       4
5  1002251       5
```

In the `SELECT` clause we went ahead and told SQLite to store the weekday as an integer.

Creating the final table

Now we can put it all together, drop columns we don't want, remove redundant information, and clean up the dates.

Removing columns in tables in SQLite is not simple. SQLite does not have `DROP` statement like some SQL implementations have. Instead, we're going to rename the current `crime` table, then copy only the columns we want into a new `crime` table.

First rename the `crime` table to a `crime_old` table that we will delete as soon as we're done.

```
res <- dbSendQuery(con, "
                   ALTER TABLE crime RENAME TO crime_old")
dbClearResult(res)
```

There should be a new table.

```
dbListTables(con)
```

```
[1] "DateFix"   "crime_old" "iucr"
```

We're going to need to list all the variables we want to keep, but let's make R do all the work.

```
a <- dbListFields(con,"crime_old")
paste(a,collapse=",")
```

```
[1] "ID,Case.Number,Date,Block,IUCR,Primary.Type,Description,Location.Description,Arrest,Domesti
```

Now we will create our new `crime`.

```
res <- dbSendQuery(con, "
                    CREATE TABLE crime AS
                    SELECT crime_old.ID,
                           crime_old.[Case.Number] AS CaseNumber,
                           DateFix.datefix AS date,
                           crime_old.Block,
                           crime_old.IUCR,
                           crime_old.Description,
                           crime_old.[Location.Description] AS LocationDescription,
                           crime_old.Arrest,
                           crime_old.Domestic,
                           crime_old.Beat,
                           crime_old.District,
                           crime_old.Ward,
                           crime_old.[Community.Area] AS CommunityArea,
                           crime_old.[X.Coordinate] AS XCoordinate,
                           crime_old.[Y.Coordinate] AS YCoordinate,
                           crime_old.Latitude,
                           crime_old.Longitude
                    FROM crime_old, DateFix
                    WHERE crime_old.ID=DateFix.ID")
dbClearResult(res)
```

This query requires a bit of discussion. First, note that the `FROM` clause includes two tables, `crime_old` and `DateFix`. The `WHERE` clause tells SQLite how to link these two tables together. It says that if there is a row in `crime_old` with a particular `ID`, then it can find its associated row in the `DateFix` table by finding the matching value in the `DateFix`'s `ID` column. For every column in the `SELECT` clause, we've included the table from where SQLite should find the column. Technically we only need to prefix the column with the table name when there might be confusion. For example, both `crime_old` and `DateFix` have a column called `ID`. However, we like to be explicit in complicated queries to remind ourselves from where all the data comes. You can also see in this `SELECT` query why periods in column names cause problems. SQL uses the period to separate the table name from the column name. If we were to include `Case.Number` in a `SELECT` statement, then SQL would think we had a table called `Case` with a column called `Number`. Let's fix this once and for all here by renaming all the columns with periods in their names. In the `SELECT` clause we have not included any of the columns with redundant information like `Primary.Type`, `FBI.Code`, and

`Location`. Technically, `Beat`, `District`, `Ward`, and `Community.Area` are all redundant information once we have `Latitude` and `Longitude`. However, "spatial joins," linking coordinates to spatial areas, is computationally expensive so that it is most likely more efficient to simply leave this redundant information here. Lastly, note that the first line is a `CREATE TABLE` statement that will store the results of this query in a new table called `crime`.

Let's look at the newly cleaned up table.

```
res <- dbSendQuery(con, "
                    SELECT *
                    FROM crime")
fetch(res, n = 10)
dbClearResult(res)
```

```
         ID CaseNumber                date                   Block IUCR
1  10000092   HY189866 2015-03-18 19:44:00         047XX W OHIO ST 041A
2  10000094   HY190059 2015-03-18 23:00:00 066XX S MARSHFIELD AVE 4625
3  10000095   HY190052 2015-03-18 22:45:00  044XX S LAKE PARK AVE 0486
4  10000096   HY190054 2015-03-18 22:30:00   051XX S MICHIGAN AVE 0460
5  10000097   HY189976 2015-03-18 21:00:00        047XX W ADAMS ST 031A
6  10000098   HY190032 2015-03-18 22:00:00    049XX S DREXEL BLVD 0460
7  10000099   HY190047 2015-03-18 23:00:00      070XX S MORGAN ST 0486
8  10000100   HY189988 2015-03-18 21:35:00    042XX S PRAIRIE AVE 0486
9  10000101   HY190020 2015-03-18 22:09:00    036XX S WOLCOTT AVE 1811
10 10000104   HY189964 2015-03-18 21:25:00    097XX S PRAIRIE AVE 0460
                    Description     LocationDescription Arrest Domestic
1            AGGRAVATED: HANDGUN                  STREET  false    false
2               PAROLE VIOLATION                  STREET   true    false
3        DOMESTIC BATTERY SIMPLE               APARTMENT  false     true
4                        SIMPLE               APARTMENT  false    false
5                  ARMED: HANDGUN                SIDEWALK  false    false
6                        SIMPLE               APARTMENT  false    false
7        DOMESTIC BATTERY SIMPLE               APARTMENT  false     true
8        DOMESTIC BATTERY SIMPLE               APARTMENT  false     true
9  POSS: CANNABIS 30GMS OR LESS                  STREET   true    false
10                       SIMPLE RESIDENCE PORCH/HALLWAY  false    false
   Beat District Ward CommunityArea XCoordinate YCoordinate Latitude
1  1111      011   28            25     1144606     1903566 41.89140
2   725      007   15            67     1166468     1860715 41.77337
3   222      002    4            39     1185075     1875622 41.81386
4   225      002    3            40     1178033     1870804 41.80080
5  1113      011   28            25     1144920     1898709 41.87806
6   223      002    4            39     1183018     1872537 41.80544
7   733      007   17            68     1170859     1858210 41.76640
8   213      002    3            38     1178746     1876914 41.81755
9   912      009   11            59     1164279     1880656 41.82814
10  511      005    6            49     1179637     1840444 41.71745
   Longitude
1  -87.74438
```

```
2  -87.66532
3  -87.59664
4  -87.62262
5  -87.74335
6  -87.60428
7  -87.64930
8  -87.61982
9  -87.67278
10 -87.61766
```

If everything looks as expected, then we can delete the `crime_old` and the `DateFix` tables.

```
res <- dbSendQuery(con, "DROP TABLE crime_old")
dbClearResult(res)
res <- dbSendQuery(con, "DROP TABLE DateFix")
dbClearResult(res)
dbListTables(con)
```

```
[1] "crime" "iucr"
```

After all this work, the size of the `chicagocrime.db` database file can become quite large. Our file is now 2.7 Gb, much larger than the size of the file we downloaded from the City of Chicago open data site. Even though we have deleted the `crime_old` and `DateFix` tables, SQLite simply marks them as deleted, but doesn't necessarily give up the space that it had allocated for their storage. It holds onto that space in case the user needs it. The `VACUUM` statement will clean up unused space, but it can take a while. It is not essential, so run this when you won't be working with your data for a little while.

```
system.time(
res <- dbSendQuery(con, "VACUUM")
)
dbClearResult(res)
```

```
   user  system elapsed
   3.47   25.68   35.95
```

After `VACUUM` our `chicagocrime.db` file is now 1 Gb... much better.

Joining data across tables

Now with data split across tables, we need to link tables together in order to get information. Let's extract the first 10 crime incident with their case numbers and FBI codes. Since `FBI.Code` is no longer in the `crime` table we need to add the table `iucr` to the `FROM` clause and link the two tables in the `WHERE` clause.

```
timeIUCRjoin <-
system.time(
{
   res <- dbSendQuery(con, "
                       SELECT crime.CaseNumber,
```

```
                        iucr.FBICode
                  FROM  crime,
                        iucr
                  WHERE crime.iucr=iucr.iucr")
   data <- fetch(res, n = -1)
})
dbClearResult(res)
data[1:10,]
```

```
   CaseNumber FBIcode
1    HY189866     04B
2    HY190059      26
3    HY190052     08B
4    HY190054     08B
5    HY189976      03
6    HY190032     08B
7    HY190047     08B
8    HY189988     08B
9    HY190020      18
10   HY189964     08B
```

For each record SQLite looks up the crime's IUCR code in the `iucr` table and links in the FBI code. SQLite is fast. This query took 9.08 seconds, but this linking does take time especially for really large datasets and large lookup tables. For the above query, SQLite scans through the `iucr` table until it finds the right IUCR code. This is not very efficient. If you were to look up the word "query" in the dictionary, you would not start on page 1 and scan through every word until you arrived at "query". Instead you would start about two-thirds of the way through the dictionary, see if the words are before or after "query" and revise your search until you find the word. Rather than search hundreds of pages, you might only need to look at nine pages.

In the same way, we can create an "index" for the `iucr` table to help speed up the search. It does not always make the queries faster and can require storing a large index in some cases. Let's try on this example.

```
res <- dbSendQuery(con, "
                   CREATE INDEX iucr_idx on iucr(iucr)")
dbClearResult(res)
```

Let's rerun the query now and see if it made a difference.

```
timeIUCRjoinIndex <-
system.time(
{
   res <- dbSendQuery(con, "
                  SELECT crime.CaseNumber,
                        iucr.FBICode
                  FROM  crime,
                        iucr
                  WHERE crime.iucr=iucr.iucr")
   data <- fetch(res, n = -1)
```

```
})
dbClearResult(res)
```

That query now takes 8.81 seconds. Creating an index is not always worth it. If you have queries that are taking too long, it's worth experimenting with creating an index to see if it helps.

In the previous query we used the `WHERE` clause to join the two tables together. Most SQL programmers prefer using `JOIN` rather than using the `WHERE` clause. The primary reason is readability. The thinking is that the `WHERE` clause should really be about filtering which cases to include, while joining tables is quite a different operation. There are also several different kinds of joins. What should the query return if a crime has an IUCR code that does not appear in the `iucr` table? `JOIN`s more carefully define the desired behavior. Here we provide an example of an inner join.

```
res <- dbSendQuery(con, "
                    SELECT crime.CaseNumber,
                           iucr.FBICode
                    FROM crime
                       INNER JOIN iucr
                       ON crime.iucr=iucr.iucr")
data <- fetch(res, n = -1)
dbClearResult(res)
```

Note how we modified the `FROM` clause. Rather than listing the tables, we tell SQL to join the `crime` table with the `iucr` table using the `iucr` columns to link them together. The `INNER JOIN` will drop any record in the `crime` table that has an IUCR code that cannot be found in the `iucr` lookup table. Using `LEFT OUTER JOIN` will force every record in `crime` (the "left" table) to appear in the final result set even if it cannot find an IUCR code in `iucr`. It will simply report `NULL` for its `FBICode`.

For a helpful, visual description of the different kinds of joins, visit this site.

Let's determine how many assaults occurred in each ward. Since the crime type is stored in `iucr.PrimaryType`, we need to join the tables.

```
res <- dbSendQuery(con, "
                    SELECT COUNT(*) AS crimecount,
                           crime.Ward
                    FROM crime
                       INNER JOIN iucr
                       ON crime.iucr=iucr.iucr
                    WHERE iucr.PrimaryType='ASSAULT'
                    GROUP BY crime.Ward")
fetch(res, n = -1)
dbClearResult(res)
```

```
   crimecount Ward
1       39803
2        5516    1
3        8602   10
4        5583   11
5        4564   12
6        4090   13
```

```
7        4503   14
8       12056   15
9       12275   16
10      15399   17
11       6886   18
12       3372   19
13      12855    2
14      14644   20
15      12334   21
16       4632   22
17       4184   23
18      14439   24
19       5255   25
20       7065   26
21      11017   27
22      16444   28
23       9661   29
24      12655    3
25       5132   30
26       4982   31
27       3167   32
28       3139   33
29      13373   34
30       4812   35
31       3384   36
32       9964   37
33       3458   38
34       2853   39
35       7562    4
36       3466   40
37       2756   41
38       8256   42
39       2101   43
40       2631   44
41       3289   45
42       4754   46
43       2575   47
44       3454   48
45       4972   49
46       9924    5
47       3179   50
48      13625    6
49      12363    7
50      12290    8
51      12425    9
```

Let's tabulate how many Part 1 crimes occur in each year. We'll use `PrimaryType` to give useful labels, `STRFTIME()` to extract the year in which each crime occurred, `FBICode` to pick out the Part 1

crimes, and an `INNER JOIN` to link the tables.

```
res <- dbSendQuery(con, "
                   SELECT iucr.PrimaryType    AS type,
                          STRFTIME('%Y',date) AS year,
                          COUNT(*)            AS crimecount
                   FROM crime
                      INNER JOIN iucr
                      ON crime.iucr=iucr.iucr
                   WHERE FBICode IN ('01A','02','03','04A','04B','05','06','07','09')
                   GROUP BY type, year")
fetch(res, n = -1)
dbClearResult(res)
```

```
                        type year crimecount
1                     ARSON 2001       1010
2                     ARSON 2002       1032
3                     ARSON 2003        955
4                     ARSON 2004        778
5                     ARSON 2005        691
6                     ARSON 2006        726
7                     ARSON 2007        712
8                     ARSON 2008        644
9                     ARSON 2009        616
10                    ARSON 2010        522
11                    ARSON 2011        504
12                    ARSON 2012        469
13                    ARSON 2013        364
14                    ARSON 2014        397
15                    ARSON 2015        453
16                    ARSON 2016        516
17                    ARSON 2017        444
18                    ARSON 2018        186
19                  ASSAULT 2001       7871
20                  ASSAULT 2002       7721
21                  ASSAULT 2003       7372
22                  ASSAULT 2004       7331
23                  ASSAULT 2005       6753
24                  ASSAULT 2006       6597
25                  ASSAULT 2007       6335
26                  ASSAULT 2008       6250
27                  ASSAULT 2009       6000
28                  ASSAULT 2010       5277
29                  ASSAULT 2011       5157
30                  ASSAULT 2012       4873
31                  ASSAULT 2013       4268
32                  ASSAULT 2014       4337
33                  ASSAULT 2015       4480
```

34		ASSAULT	2016	5710
35		ASSAULT	2017	5794
36		ASSAULT	2018	3299
37		BATTERY	2001	16389
38		BATTERY	2002	15199
39		BATTERY	2003	12476
40		BATTERY	2004	11530
41		BATTERY	2005	11329
42		BATTERY	2006	11002
43		BATTERY	2007	11153
44		BATTERY	2008	10803
45		BATTERY	2009	10146
46		BATTERY	2010	9435
47		BATTERY	2011	8403
48		BATTERY	2012	8009
49		BATTERY	2013	6632
50		BATTERY	2014	6578
51		BATTERY	2015	7019
52		BATTERY	2016	8086
53		BATTERY	2017	7845
54		BATTERY	2018	4078
55		BURGLARY	2001	26011
56		BURGLARY	2002	25623
57		BURGLARY	2003	25156
58		BURGLARY	2004	24564
59		BURGLARY	2005	25504
60		BURGLARY	2006	24324
61		BURGLARY	2007	24858
62		BURGLARY	2008	26218
63		BURGLARY	2009	26766
64		BURGLARY	2010	26422
65		BURGLARY	2011	26619
66		BURGLARY	2012	22843
67		BURGLARY	2013	17894
68		BURGLARY	2014	14570
69		BURGLARY	2015	13183
70		BURGLARY	2016	14287
71		BURGLARY	2017	12992
72		BURGLARY	2018	6057
73	CRIM SEXUAL	ASSAULT	2001	1796
74	CRIM SEXUAL	ASSAULT	2002	1830
75	CRIM SEXUAL	ASSAULT	2003	1595
76	CRIM SEXUAL	ASSAULT	2004	1570
77	CRIM SEXUAL	ASSAULT	2005	1546
78	CRIM SEXUAL	ASSAULT	2006	1466
79	CRIM SEXUAL	ASSAULT	2007	1538
80	CRIM SEXUAL	ASSAULT	2008	1525
81	CRIM SEXUAL	ASSAULT	2009	1414

82		CRIM SEXUAL ASSAULT	2010	1351
83		CRIM SEXUAL ASSAULT	2011	1477
84		CRIM SEXUAL ASSAULT	2012	1415
85		CRIM SEXUAL ASSAULT	2013	1291
86		CRIM SEXUAL ASSAULT	2014	1331
87		CRIM SEXUAL ASSAULT	2015	1383
88		CRIM SEXUAL ASSAULT	2016	1533
89		CRIM SEXUAL ASSAULT	2017	1599
90		CRIM SEXUAL ASSAULT	2018	795
91		HOMICIDE	2001	667
92		HOMICIDE	2002	656
93		HOMICIDE	2003	601
94		HOMICIDE	2004	453
95		HOMICIDE	2005	451
96		HOMICIDE	2006	470
97		HOMICIDE	2007	447
98		HOMICIDE	2008	513
99		HOMICIDE	2009	460
100		HOMICIDE	2010	438
101		HOMICIDE	2011	436
102		HOMICIDE	2012	504
103		HOMICIDE	2013	421
104		HOMICIDE	2014	424
105		HOMICIDE	2015	494
106		HOMICIDE	2016	779
107		HOMICIDE	2017	669
108		HOMICIDE	2018	288
109		MOTOR VEHICLE THEFT	2001	27549
110		MOTOR VEHICLE THEFT	2002	25121
111		MOTOR VEHICLE THEFT	2003	22748
112		MOTOR VEHICLE THEFT	2004	22805
113		MOTOR VEHICLE THEFT	2005	22497
114		MOTOR VEHICLE THEFT	2006	21818
115		MOTOR VEHICLE THEFT	2007	18573
116		MOTOR VEHICLE THEFT	2008	18881
117		MOTOR VEHICLE THEFT	2009	15482
118		MOTOR VEHICLE THEFT	2010	19029
119		MOTOR VEHICLE THEFT	2011	19388
120		MOTOR VEHICLE THEFT	2012	16492
121		MOTOR VEHICLE THEFT	2013	12582
122		MOTOR VEHICLE THEFT	2014	9912
123		MOTOR VEHICLE THEFT	2015	10070
124		MOTOR VEHICLE THEFT	2016	11296
125		MOTOR VEHICLE THEFT	2017	11414
126		MOTOR VEHICLE THEFT	2018	5264
127	OFFENSE	INVOLVING CHILDREN	2001	359
128	OFFENSE	INVOLVING CHILDREN	2002	367
129	OFFENSE	INVOLVING CHILDREN	2003	378

130	OFFENSE INVOLVING	CHILDREN	2004	348
131	OFFENSE INVOLVING	CHILDREN	2005	336
132	OFFENSE INVOLVING	CHILDREN	2006	300
133	OFFENSE INVOLVING	CHILDREN	2007	297
134	OFFENSE INVOLVING	CHILDREN	2008	212
135	OFFENSE INVOLVING	CHILDREN	2009	206
136	OFFENSE INVOLVING	CHILDREN	2010	195
137	OFFENSE INVOLVING	CHILDREN	2011	164
138	OFFENSE INVOLVING	CHILDREN	2012	151
139	OFFENSE INVOLVING	CHILDREN	2013	127
140	OFFENSE INVOLVING	CHILDREN	2014	134
141	OFFENSE INVOLVING	CHILDREN	2015	156
142	OFFENSE INVOLVING	CHILDREN	2016	141
143	OFFENSE INVOLVING	CHILDREN	2017	204
144	OFFENSE INVOLVING	CHILDREN	2018	105
145		RITUALISM	2001	8
146		RITUALISM	2002	1
147		RITUALISM	2003	1
148		RITUALISM	2004	1
149		RITUALISM	2005	2
150		RITUALISM	2006	6
151		RITUALISM	2007	1
152		ROBBERY	2001	18441
153		ROBBERY	2002	18522
154		ROBBERY	2003	17332
155		ROBBERY	2004	15978
156		ROBBERY	2005	16047
157		ROBBERY	2006	15968
158		ROBBERY	2007	15450
159		ROBBERY	2008	16703
160		ROBBERY	2009	15980
161		ROBBERY	2010	14274
162		ROBBERY	2011	13983
163		ROBBERY	2012	13485
164		ROBBERY	2013	11820
165		ROBBERY	2014	9799
166		ROBBERY	2015	9638
167		ROBBERY	2016	11960
168		ROBBERY	2017	11877
169		ROBBERY	2018	5156
170		THEFT	2001	99264
171		THEFT	2002	98327
172		THEFT	2003	98875
173		THEFT	2004	95463
174		THEFT	2005	85685
175		THEFT	2006	86240
176		THEFT	2007	85156
177		THEFT	2008	88432

```
178                     THEFT 2009      80972
179                     THEFT 2010      76754
180                     THEFT 2011      75148
181                     THEFT 2012      75459
182                     THEFT 2013      71530
183                     THEFT 2014      61557
184                     THEFT 2015      57332
185                     THEFT 2016      61588
186                     THEFT 2017      64296
187                     THEFT 2018      33244
```

Exercises

5. Count the number of arrests for "MOTOR VEHICLE THEFT"
6. Which District has the most thefts?

Subqueries

Sometimes we would like to use the results of one query as part of another query. You can put `SELECT` statements inside `FROM` statements to accomplish this. We'll use this method to see if addresses are always geocoded to the same coordinates. Here are the unique combinations of addresses and coordinates. We'll just show here the first 20.

```
res <- dbSendQuery(con, "
                    SELECT DISTINCT Block, XCoordinate, YCoordinate
                    FROM crime")
fetch(res, n = 20)
dbClearResult(res)
```

```
                               Block XCoordinate YCoordinate
1                       047XX W OHIO ST     1144606     1903566
2                 066XX S MARSHFIELD AVE     1166468     1860715
3                  044XX S LAKE PARK AVE     1185075     1875622
4                   051XX S MICHIGAN AVE     1178033     1870804
5                       047XX W ADAMS ST     1144920     1898709
6                    049XX S DREXEL BLVD     1183018     1872537
7                      070XX S MORGAN ST     1170859     1858210
8                    042XX S PRAIRIE AVE     1178746     1876914
9                    036XX S WOLCOTT AVE     1164279     1880656
10                   097XX S PRAIRIE AVE     1179637     1840444
11 130XX S DR MARTIN LUTHER KING JR DR     1180907     1818839
12                  078XX S VINCENNES AVE     1175130     1853144
13                   086XX S EXCHANGE AVE     1197309     1848290
14                    014XX S ASHLAND AVE     1165950     1893388
15                    051XX W CHICAGO AVE     1141741     1904839
```

```
16                077XX S KINGSTON AVE     1194535     1854110
17                   024XX W NORTH AVE     1159959     1910569
18                 069XX S LOOMIS BLVD     1168192     1858832
19               105XX S LAFAYETTE AVE     1177790     1835106
20                 087XX S KIMBARK AVE     1186312     1847473
```

We would like to know if `Block` shows up multiple times in these results or just one time. We use the results of this query in the `FROM` clause and count up the frequency of each `Block`.

```
res <- dbSendQuery(con, "
                   SELECT COUNT(*), Block
                   FROM
                      (SELECT DISTINCT block,
                                       XCoordinate,
                                       YCoordinate
                       FROM crime)
                   GROUP BY block")
fetch(res, n = 20)
dbClearResult(res)
```

```
   COUNT(*)            block
1        28   0000X E 100 PL
2        22   0000X E 100 ST
3        67 0000X E 100TH PL
4        54 0000X E 100TH ST
5         7   0000X E 101 PL
6        21   0000X E 101 ST
7        57 0000X E 101ST PL
8        49 0000X E 101ST ST
9        13   0000X E 102 PL
10       13   0000X E 102 ST
11       65 0000X E 102ND PL
12       58 0000X E 102ND ST
13        8   0000X E 103 PL
14       12   0000X E 103 ST
15       34 0000X E 103RD PL
16       54 0000X E 103RD ST
17        8   0000X E 104 ST
18       32 0000X E 104TH ST
19        8   0000X E 105 ST
20       34 0000X E 105TH ST
```

Clearly, the coordinates are not unique to each address. This suggests that the coordinates have greater spatial resolution than the addresses imply. The addresses are "rounded" to provide some privacy, but the coordinates appear to point to more specific places.

After completing the final exercise, remember to run `dbDisconnect(con)` to disconnect from the database.

Exercise

As a final exercise, which does not involve a subquery,

7. Count the number of assaults, since 2010, that occurred on Fridays and Saturdays, after 6pm, reporting the date, day of week, hour of the day, and year

Solutions

1. Print out all of the rows in iucr

```
res <- dbSendQuery(con, "SELECT * from iucr")
fetch(res, n=-1)
dbClearResult(res)
```

```
    iucr                       PrimaryType FBIcode
1   041A                           BATTERY     04B
2   4625                      OTHER OFFENSE      26
3   0486                           BATTERY     08B
4   0460                           BATTERY     08B
5   031A                           ROBBERY      03
6   1811                         NARCOTICS      18
7   1320                   CRIMINAL DAMAGE      14
8   2825                      OTHER OFFENSE      26
9   143A                  WEAPONS VIOLATION      15
10  0860                             THEFT      06
11  0610                           BURGLARY      05
12  0910                MOTOR VEHICLE THEFT      07
13  0890                             THEFT      06
14  0470            PUBLIC PEACE VIOLATION      24
15  1140                DECEPTIVE PRACTICE      12
16  0320                           ROBBERY      03
17  0430                           BATTERY     04B
18  141A                  WEAPONS VIOLATION      15
19  0560                           ASSAULT     08A
20  1350                  CRIMINAL TRESPASS      26
21  0312                           ROBBERY      03
22  1305                   CRIMINAL DAMAGE      14
23  1310                   CRIMINAL DAMAGE      14
24  0420                           BATTERY     04B
25  0281                CRIM SEXUAL ASSAULT      02
26  3760  INTERFERENCE WITH PUBLIC OFFICER      24
27  2170                         NARCOTICS      18
28  1020                             ARSON      09
29  0483                           BATTERY     04B
30  0520                           ASSAULT     04A
31  502R                      OTHER OFFENSE      26
```

32	0326	ROBBERY	03
33	0810	THEFT	06
34	0820	THEFT	06
35	0650	BURGLARY	05
36	2820	OTHER OFFENSE	26
37	2027	NARCOTICS	18
38	1150	DECEPTIVE PRACTICE	11
39	0453	BATTERY	04B
40	0840	THEFT	06
41	2024	NARCOTICS	18
42	0497	BATTERY	04B
43	4387	OTHER OFFENSE	26
44	0340	ROBBERY	03
45	1154	DECEPTIVE PRACTICE	11
46	2250	LIQUOR LAW VIOLATION	22
47	4255	KIDNAPPING	26
48	2050	NARCOTICS	18
49	1330	CRIMINAL TRESPASS	26
50	1340	CRIMINAL DAMAGE	14
51	1562	SEX OFFENSE	17
52	1345	CRIMINAL DAMAGE	14
53	1130	DECEPTIVE PRACTICE	11
54	1153	DECEPTIVE PRACTICE	11
55	0484	BATTERY	08B
56	0620	BURGLARY	05
57	0880	THEFT	06
58	0870	THEFT	06
59	2826	OTHER OFFENSE	26
60	0454	BATTERY	08B
61	0330	ROBBERY	03
62	0630	BURGLARY	05
63	1152	DECEPTIVE PRACTICE	11
64	2091	NARCOTICS	18
65	1821	NARCOTICS	18
66	0920	MOTOR VEHICLE THEFT	07
67	1822	NARCOTICS	18
68	2014	NARCOTICS	18
69	1110	DECEPTIVE PRACTICE	11
70	0496	BATTERY	04B
71	0440	BATTERY	08B
72	1582	OFFENSE INVOLVING CHILDREN	17
73	2017	NARCOTICS	18
74	051A	ASSAULT	04A
75	1812	NARCOTICS	18
76	1122	DECEPTIVE PRACTICE	10
77	0545	ASSAULT	08A
78	5000	OTHER OFFENSE	26
79	5011	OTHER OFFENSE	26

80	1505	PROSTITUTION	16
81	1120	DECEPTIVE PRACTICE	10
82	5007	OTHER OFFENSE	26
83	1121	DECEPTIVE PRACTICE	10
84	1751	OFFENSE INVOLVING CHILDREN	20
85	5002	OTHER OFFENSE	26
86	1750	OFFENSE INVOLVING CHILDREN	20
87	1365	CRIMINAL TRESPASS	26
88	5111	OTHER OFFENSE	26
89	0313	ROBBERY	03
90	4510	OTHER OFFENSE	26
91	1792	KIDNAPPING	20
92	0530	ASSAULT	04A
93	0930	MOTOR VEHICLE THEFT	07
94	5004	SEX OFFENSE	17
95	141C	WEAPONS VIOLATION	15
96	0915	MOTOR VEHICLE THEFT	07
97	1506	PROSTITUTION	16
98	0337	ROBBERY	03
99	2023	NARCOTICS	18
100	2021	NARCOTICS	18
101	0110	HOMICIDE	01A
102	1585	SEX OFFENSE	17
103	0554	ASSAULT	08A
104	1512	PROSTITUTION	16
105	0495	BATTERY	04B
106	1210	DECEPTIVE PRACTICE	11
107	033A	ROBBERY	03
108	1360	CRIMINAL TRESPASS	26
109	0479	BATTERY	04B
110	1477	WEAPONS VIOLATION	15
111	3731	INTERFERENCE WITH PUBLIC OFFICER	24
112	0265	CRIM SEXUAL ASSAULT	02
113	1206	DECEPTIVE PRACTICE	11
114	2016	NARCOTICS	18
115	2018	NARCOTICS	18
116	1661	GAMBLING	19
117	2026	NARCOTICS	18
118	1242	DECEPTIVE PRACTICE	11
119	1754	OFFENSE INVOLVING CHILDREN	02
120	1563	SEX OFFENSE	17
121	2028	NARCOTICS	18
122	2025	NARCOTICS	18
123	143C	WEAPONS VIOLATION	15
124	141B	WEAPONS VIOLATION	15
125	1507	PROSTITUTION	16
126	502P	OTHER OFFENSE	26
127	3710	INTERFERENCE WITH PUBLIC OFFICER	24

128	051B	ASSAULT	04A
129	0498	BATTERY	04B
130	0266	CRIM SEXUAL ASSAULT	02
131	0865	THEFT	06
132	0325	ROBBERY	03
133	1752	OFFENSE INVOLVING CHILDREN	20
134	1513	PROSTITUTION	16
135	2034	NARCOTICS	18
136	2210	LIQUOR LAW VIOLATION	22
137	1335	CRIMINAL TRESPASS	26
138	1155	DECEPTIVE PRACTICE	11
139	2890	PUBLIC PEACE VIOLATION	26
140	0850	THEFT	06
141	2022	NARCOTICS	18
142	5001	OTHER OFFENSE	26
143	2230	LIQUOR LAW VIOLATION	22
144	1156	DECEPTIVE PRACTICE	11
145	3730	INTERFERENCE WITH PUBLIC OFFICER	24
146	1670	GAMBLING	19
147	4210	KIDNAPPING	26
148	0917	MOTOR VEHICLE THEFT	07
149	3960	INTIMIDATION	26
150	2851	PUBLIC PEACE VIOLATION	26
151	0584	STALKING	26
152	1220	DECEPTIVE PRACTICE	11
153	4230	KIDNAPPING	26
154	143B	WEAPONS VIOLATION	15
155	0263	CRIM SEXUAL ASSAULT	02
156	4650	OTHER OFFENSE	26
157	3100	PUBLIC PEACE VIOLATION	24
158	2870	PUBLIC PEACE VIOLATION	24
159	1185	DECEPTIVE PRACTICE	11
160	1170	DECEPTIVE PRACTICE	11
161	1753	OFFENSE INVOLVING CHILDREN	02
162	1195	DECEPTIVE PRACTICE	11
163	2900	WEAPONS VIOLATION	15
164	2850	PUBLIC PEACE VIOLATION	26
165	3300	PUBLIC PEACE VIOLATION	26
166	1090	ARSON	09
167	2015	NARCOTICS	18
168	0580	STALKING	08A
169	0482	BATTERY	04B
170	1200	DECEPTIVE PRACTICE	13
171	0291	CRIM SEXUAL ASSAULT	02
172	1245	DECEPTIVE PRACTICE	11
173	5110	OTHER OFFENSE	26
174	501A	OTHER OFFENSE	26
175	2093	NARCOTICS	18

176	0461	BATTERY	04B
177	2220	LIQUOR LAW VIOLATION	22
178	0553	ASSAULT	04A
179	0935	MOTOR VEHICLE THEFT	07
180	1261	DECEPTIVE PRACTICE	11
181	1570	SEX OFFENSE	17
182	2012	NARCOTICS	18
183	1535	OBSCENITY	17
184	1460	WEAPONS VIOLATION	15
185	031B	ROBBERY	03
186	2020	NARCOTICS	18
187	1544	SEX OFFENSE	17
188	0558	ASSAULT	04A
189	1790	OFFENSE INVOLVING CHILDREN	20
190	5131	OTHER OFFENSE	26
191	1536	PUBLIC INDECENCY	17
192	0841	THEFT	06
193	1450	WEAPONS VIOLATION	15
194	2031	NARCOTICS	18
195	0275	CRIM SEXUAL ASSAULT	02
196	0552	ASSAULT	04A
197	0555	ASSAULT	04A
198	5112	OTHER OFFENSE	26
199	2013	NARCOTICS	18
200	1055	HUMAN TRAFFICKING	26
201	3970	INTIMIDATION	26
202	4651	OTHER OFFENSE	26
203	2860	PUBLIC PEACE VIOLATION	24
204	0550	ASSAULT	04A
205	0485	BATTERY	04B
206	4386	OTHER OFFENSE	26
207	4310	OTHER OFFENSE	26
208	1478	CONCEALED CARRY LICENSE VIOLATION	15
209	0925	MOTOR VEHICLE THEFT	07
210	3750	INTERFERENCE WITH PUBLIC OFFICER	24
211	4652	OTHER OFFENSE	26
212	0462	BATTERY	04B
213	0334	ROBBERY	03
214	2830	OTHER OFFENSE	17
215	501H	OTHER OFFENSE	26
216	0487	BATTERY	04B
217	5130	OTHER OFFENSE	26
218	0557	ASSAULT	04A
219	2090	NARCOTICS	18
220	0551	ASSAULT	04A
221	1590	SEX OFFENSE	17
222	1540	OBSCENITY	17
223	1720	OFFENSE INVOLVING CHILDREN	20

224	0583	STALKING	08A
225	2110	NARCOTICS	18
226	502T	OTHER OFFENSE	26
227	3800	INTERFERENCE WITH PUBLIC OFFICER	26
228	1025	ARSON	09
229	142A	WEAPONS VIOLATION	15
230	1205	DECEPTIVE PRACTICE	11
231	1480	CONCEALED CARRY LICENSE VIOLATION	15
232	2011	NARCOTICS	18
233	0264	CRIM SEXUAL ASSAULT	02
234	0488	BATTERY	04B
235	1241	DECEPTIVE PRACTICE	11
236	1240	DECEPTIVE PRACTICE	11
237	1566	SEX OFFENSE	17
238	041B	BATTERY	04B
239	0261	CRIM SEXUAL ASSAULT	02
240	1050	HUMAN TRAFFICKING	26
241	1525	PROSTITUTION	16
242	1564	SEX OFFENSE	17
243	0331	ROBBERY	03
244	2840	PUBLIC PEACE VIOLATION	24
245	1900	OTHER NARCOTIC VIOLATION	18
246	0142	HOMICIDE	01B
247	2029	NARCOTICS	18
248	4389	OTHER OFFENSE	26
249	1375	CRIMINAL DAMAGE	14
250	0937	MOTOR VEHICLE THEFT	07
251	1651	GAMBLING	19
252	4388	OTHER OFFENSE	26
253	1580	SEX OFFENSE	17
254	1565	SEX OFFENSE	17
255	1511	PROSTITUTION	16
256	2032	NARCOTICS	18
257	500N	OTHER OFFENSE	26
258	1537	OFFENSE INVOLVING CHILDREN	16
259	1151	DECEPTIVE PRACTICE	11
260	4240	KIDNAPPING	26
261	1541	OBSCENITY	17
262	4860	OTHER OFFENSE	26
263	1526	PROSTITUTION	16
264	0842	THEFT	06
265	1479	CONCEALED CARRY LICENSE VIOLATION	15
266	0475	BATTERY	08B
267	2010	NARCOTICS	18
268	1791	OFFENSE INVOLVING CHILDREN	20
269	0843	THEFT	06
270	1680	GAMBLING	19
271	0271	CRIM SEXUAL ASSAULT	02

272	0918	MOTOR VEHICLE THEFT	07
273	0895	THEFT	06
274	2080	NARCOTICS	18
275	2095	NARCOTICS	18
276	4220	KIDNAPPING	26
277	1530	PROSTITUTION	16
278	1549	PROSTITUTION	16
279	1710	OFFENSE INVOLVING CHILDREN	20
280	0273	CRIM SEXUAL ASSAULT	02
281	2019	NARCOTICS	18
282	033B	ROBBERY	03
283	1850	NARCOTICS	18
284	1265	CRIMINAL DAMAGE	14
285	1235	DECEPTIVE PRACTICE	11
286	1780	OFFENSE INVOLVING CHILDREN	20
287	1370	CRIMINAL DAMAGE	14
288	3000	PUBLIC PEACE VIOLATION	26
289	2240	LIQUOR LAW VIOLATION	22
290	0452	BATTERY	04B
291	1010	ARSON	09
292	2094	NARCOTICS	18
293	1255	DECEPTIVE PRACTICE	11
294	5003	OTHER OFFENSE	26
295	0556	ASSAULT	04A
296	5093	NON-CRIMINAL	26
297	5132	OTHER OFFENSE	26
298	2092	NARCOTICS	18
299	0480	BATTERY	04B
300	0581	STALKING	08A
301	3751	INTERFERENCE WITH PUBLIC OFFICER	24
302	0585	NON-CRIMINAL	26
303	1135	DECEPTIVE PRACTICE	11
304	2033	NARCOTICS	18
305	0489	BATTERY	04B
306	1515	PROSTITUTION	16
307	3610	OTHER OFFENSE	24
308	0274	CRIM SEXUAL ASSAULT	02
309	1576	SEX OFFENSE	17
310	2160	NARCOTICS	26
311	3920	INTERFERENCE WITH PUBLIC OFFICER	26
312	0481	BATTERY	04B
313	2040	NARCOTICS	18
314	0510	RITUALISM	04B
315	3975	INTIMIDATION	26
316	5009	OTHER OFFENSE	26
317	1755	OFFENSE INVOLVING CHILDREN	20
318	4810	OTHER OFFENSE	26
319	5013	OTHER OFFENSE	26

320	3966	INTIMIDATION	26
321	2070	NARCOTICS	18
322	1840	NARCOTICS	18
323	5073	NON-CRIMINAL (SUBJECT SPECIFIED)	26
324	3910	INTERFERENCE WITH PUBLIC OFFICER	26
325	4800	OTHER OFFENSE	26
326	5121	OTHER OFFENSE	26
327	5114	NON-CRIMINAL	26
328	0927	MOTOR VEHICLE THEFT	07
329	1035	ARSON	09
330	1260	DECEPTIVE PRACTICE	11
331	0830	THEFT	06
332	0450	BATTERY	04B
333	3740	INTERFERENCE WITH PUBLIC OFFICER	24
334	1725	OFFENSE INVOLVING CHILDREN	20
335	1435	WEAPONS VIOLATION	15
336	2895	PUBLIC PEACE VIOLATION	26
337	500E	OTHER OFFENSE	26
338	5094	NON-CRIMINAL	26
339	2060	NARCOTICS	18
340	1520	PROSTITUTION	16
341	1230	DECEPTIVE PRACTICE	11
342	3720	INTERFERENCE WITH PUBLIC OFFICER	24
343	1030	ARSON	09
344	0272	CRIM SEXUAL ASSAULT	02
345	1481	NON-CRIMINAL	15
346	1715	OFFENSE INVOLVING CHILDREN	20
347	1510	PROSTITUTION	16
348	5120	OTHER OFFENSE	26
349	5122	OTHER OFFENSE	26
350	0451	BATTERY	04B
351	5113	NON-CRIMINAL	26
352	3961	INTIMIDATION	26
353	1682	OTHER OFFENSE	19
354	1476	WEAPONS VIOLATION	15
355	2111	NARCOTICS	26
356	1626	GAMBLING	19
357	1630	GAMBLING	19
358	1622	GAMBLING	19
359	1611	GAMBLING	19
360	1440	WEAPONS VIOLATION	15
361	2030	NARCOTICS	18
362	0262	CRIM SEXUAL ASSAULT	02
363	5008	OTHER OFFENSE	26
364	142B	WEAPONS VIOLATION	15
365	0938	MOTOR VEHICLE THEFT	07
366	1578	SEX OFFENSE	17
367	1624	GAMBLING	19

```
368 0499                           BATTERY     04B
369 1631                          GAMBLING      19
370 5005                       SEX OFFENSE      17
371 1650                          GAMBLING      19
372 0494                         RITUALISM     08B
373 0130                          HOMICIDE     01A
374 1610                          GAMBLING      19
375 0141                          HOMICIDE     01B
376 3770  INTERFERENCE WITH PUBLIC OFFICER      26
377 1572                       SEX OFFENSE      17
378 1621                          GAMBLING      19
379 4740                     OTHER OFFENSE      26
380 3980                      INTIMIDATION      26
381 1160                DECEPTIVE PRACTICE      11
382 1574                       SEX OFFENSE      17
383 1625                          GAMBLING      19
384 1620                          GAMBLING      19
385 1531                      PROSTITUTION      16
386 0490                         RITUALISM     04B
387 2120                         NARCOTICS      26
388 0928               MOTOR VEHICLE THEFT      07
389 2251              LIQUOR LAW VIOLATION      22
390 1681                          GAMBLING      19
391 3400            PUBLIC PEACE VIOLATION      26
392 3200            PUBLIC PEACE VIOLATION      26
393 0492                         RITUALISM     04B
394 9901                DOMESTIC VIOLENCE     08B
395 1627                          GAMBLING      19
396 0493                         RITUALISM     04B
397 1697                          GAMBLING      19
398 4750                     OTHER OFFENSE      26
399 1860                         NARCOTICS      18
400 1633                          GAMBLING      19
401 1640                          GAMBLING      19
402 1521                      PROSTITUTION      16
```

2. Print out all the IUCR codes for "KIDNAPPING"

```
res <- dbSendQuery(con, "
   SELECT iucr
   FROM iucr
   WHERE Primarytype='KIDNAPPING'")
fetch(res, n=-1)
dbClearResult(res)
```

```
  iucr
1 4255
2 1792
3 4210
```

```
4 4230
5 4240
6 4220
```

3. How many IUCR codes are there for "ASSAULT"?

```
res <- dbSendQuery(con, "
   SELECT *
   FROM iucr
   WHERE Primarytype='ASSAULT'")
fetch(res, n=-1)
dbClearResult(res)
```

```
   iucr PrimaryType FBIcode
1  0560     ASSAULT     08A
2  0520     ASSAULT     04A
3  051A     ASSAULT     04A
4  0545     ASSAULT     08A
5  0530     ASSAULT     04A
6  0554     ASSAULT     08A
7  051B     ASSAULT     04A
8  0553     ASSAULT     04A
9  0558     ASSAULT     04A
10 0552     ASSAULT     04A
11 0555     ASSAULT     04A
12 0550     ASSAULT     04A
13 0557     ASSAULT     04A
14 0551     ASSAULT     04A
15 0556     ASSAULT     04A
```

4. Try doing the prior exercise again using COUNT(*) if you did not use it the first time

```
res <- dbSendQuery(con, "
   SELECT COUNT(*)
   FROM iucr
   WHERE PrimaryType='ASSAULT'")
fetch(res, n=-1)
dbClearResult(res)
```

```
  COUNT(*)
1       15
```

5. Count the number of arrests for "MOTOR VEHICLE THEFT"

```
res <- dbSendQuery(con, "
   SELECT COUNT(*)
   FROM crime
      INNER JOIN iucr ON
         crime.iucr=iucr.iucr
   WHERE crime.Arrest='true' AND
         iucr.PrimaryType='MOTOR VEHICLE THEFT'")
```

```
fetch(res, n=-1)
dbClearResult(res)
```

```
  COUNT(*)
1    28496
```

6. Which District has the most thefts?

```
res <- dbSendQuery(con, "
   SELECT COUNT(*) AS crimecount,
          District
   FROM crime
      INNER JOIN iucr ON
         crime.iucr=iucr.iucr
   WHERE iucr.PrimaryType='THEFT'
   GROUP BY District")
a <- fetch(res, n=-1)
dbClearResult(res)

subset(a, crimecount==max(crimecount))
# or
a[which.max(a$crimecount),]
```

```
   crimecount District
18     122957      018
   crimecount District
18     122957      018
```

7. Count the number of assaults, since 2010, that occurred on Fridays and Saturdays, after 6pm, reporting the date, day of week, hour of the day, and year

```
#  count 1) assaults
#        2) since 2016 on
#        3) Fridays and Saturdays
#        4) after 6pm
# report 5) count,
#        6) date,
#        7) day of week, and
#        8) hour of the day
#        9) year
res <- dbSendQuery(con, "
   SELECT COUNT(*),
          DATE(crime.Date) AS crimdate,
          CAST(STRFTIME('%w',crime.Date) AS INTEGER) AS weekday,
          CAST(STRFTIME('%H',crime.Date) AS INTEGER) AS hour,
          CAST(STRFTIME('%Y',crime.Date) AS INTEGER) AS year
   FROM   crime
             INNER JOIN iucr ON
                crime.iucr=iucr.iucr
   WHERE  iucr.PrimaryType='ASSAULT' AND
```

```
          year>=2016 AND
          weekday>=5 AND
          hour>=18
   GROUP BY crimdate, weekday, hour, year")
fetch(res, n = 20)
dbClearResult(res)
```

```
   COUNT(*)   crimdate weekday hour year
1         2 2016-01-01       5   18 2016
2         3 2016-01-01       5   19 2016
3         1 2016-01-01       5   20 2016
4         3 2016-01-01       5   21 2016
5         1 2016-01-01       5   22 2016
6         3 2016-01-01       5   23 2016
7         2 2016-01-02       6   18 2016
8         2 2016-01-02       6   19 2016
9         2 2016-01-02       6   20 2016
10        1 2016-01-02       6   21 2016
11        2 2016-01-02       6   22 2016
12        1 2016-01-02       6   23 2016
13        6 2016-01-08       5   18 2016
14        2 2016-01-08       5   19 2016
15        1 2016-01-08       5   21 2016
16        4 2016-01-08       5   23 2016
17        2 2016-01-09       6   18 2016
18        2 2016-01-09       6   19 2016
19        4 2016-01-09       6   20 2016
20        2 2016-01-09       6   21 2016
```

```
dbDisconnect(con)
```