Introduction to SQL, Part 2

Greg Ridgeway Ruth Moyer

2026-02-10

Table of contents

```
[1] TRUE
```

1 Creating an IUCR lookup table

The crime table in our Chicago crime database is not ideal. It is 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	PrimaryType	FBICode	Longitude	Latitude
040XX W 26TH ST	0560	ASSAULT	08A	-87.67741	41.90842

Block	IUCR	PrimaryType	FBICode	Longitude	Latitude
089XX S SOUTH CHICAGO AVE	0498	BATTERY	04B	-87.63394	41.88602
052XX S HARPER AVE	2820	OTHER OFFENSE	26	-87.62615	41.87183
033XX N TROY ST	2825	OTHER OFFENSE	26	-87.69560	41.85655
015XX W 107TH ST	1310	CRIMINAL DAMAGE	14	-87.59488	41.65512
0000X N LARAMIE AVE	2018	NARCOTICS	18	-87.76673	41.94523
0000X N KEELER AVE	0554	ASSAULT	08A	-87.61501	41.76935
026XX N ELSTON AVE	0560	ASSAULT	08A	-87.57389	41.76742
076XX S ABERDEEN ST	0486	BATTERY	08B	-87.64308	41.76094
3XX N SHEFFIELD AVE	1811	NARCOTICS	18	-87.70109	41.79261

Note that whenever `IUCR` is 0560, then `PrimaryType` is ASSAULT and `FBICode` 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. Note that it is essential to store the IUCR code in the crime table. Both IUCR 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. Here is Chicago PD's listing of FBI codes.

Aside from reducing database size, eliminating redundant information also provides "update consistency." In the table's current form, we could erroneously add a row that had `IUCR` 0560, `PrimaryType` as BATTERY, and `FBICode` 14.

Block	IUCR	PrimaryType	FBICode	Longitude	Latitude
040XX W 26TH ST	0560	ASSAULT	08A	-87.67741	41.90842
040XX W 26TH ST	0560	BATTERY	14	-87.67741	41.90842

The database would not complain even though this is an incorrect combination. IUCR code 0560 *must* link with ASSAULT and 08A. An IUCR lookup table avoids this possibility. The lookup table has each IUCR code showing up only once and always linking to the correct `PrimaryType` and `FBICode`.

IUCR	PrimaryType	FBICode
0486	BATTERY	08B
0498	BATTERY	04B
0554	ASSAULT	08A

IUCR	PrimaryType	FBICode
0560	ASSAULT	08A
1310	CRIMINAL DAMAGE	14
1811	NARCOTICS	18
2018	NARCOTICS	18
2820	OTHER OFFENSE	26
2825	OTHER OFFENSE	26

Then we can remove `PrimaryType` and `FBICode` from the crime table and look up the associated `PrimaryType` and `FBICode` from the IUCR lookup table whenever we need that information.

Let's start by reconnecting to the Chicago crime database.

```r
library(dplyr)
library(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.

```r
a <- dbGetQuery(con, "
  SELECT DISTINCT IUCR, PrimaryType, FBIcode
  FROM crime")
head(a)
```

```
  IUCR                PrimaryType FBICode
1 1582 OFFENSE INVOLVING CHILDREN      17
2 2017                  NARCOTICS      18
3 0326                    ROBBERY      03
4 0281        CRIM SEXUAL ASSAULT      02
5 1320            CRIMINAL DAMAGE      14
6 0810                      THEFT      06
```

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 a single primary type and a single FBI code.

```r
a |> count(IUCR) |> filter(n > 1)
```

```
   IUCR n
1  0261 2
2  0262 2
3  0263 2
4  0264 2
5  0265 2
6  0266 2
7  0271 2
8  0272 2
9  0273 2
10 0274 2
11 0275 2
12 0281 2
13 0291 2
14 1030 2
15 1035 2
16 1261 2
17 1537 2
18 1540 2
19 1541 2
20 1576 2
21 1581 2
22 1710 2
23 1715 2
24 1725 2
25 1750 2
26 1751 2
27 1752 2
28 1755 2
29 1780 2
30 1790 2
31 1792 2
32 2091 2
33 2092 2
34 2093 2
35 2820 2
36 2850 2
37 2851 2
38 2890 2
39 2895 2
40 3300 2
41 3400 2
42 3960 2
```

```
43 3961 2
44 3966 2
45 5114 2
```

Unfortunately, it looks like several IUCR codes have multiple values for `PrimaryType` and/or `FBICode`. Let's start by examining codes 2091, 2092, and 2093.

```
dbGetQuery(con, "
  SELECT COUNT(*) AS crimecount,
         IUCR,
         PrimaryType,
         FBICode,
         SUBSTR(Date, 7, 4) AS year
  FROM crime
  WHERE IUCR IN ('2091', '2092', '2093')
  GROUP BY IUCR, PrimaryType, year, FBICode
  ORDER BY IUCR, PrimaryType, year, FBICode")
```

```
   crimecount IUCR PrimaryType FBICode year
1         389 2091   NARCOTICS      26 2001
2         267 2091   NARCOTICS      26 2002
3         238 2091   NARCOTICS      26 2003
4         288 2091   NARCOTICS      26 2004
5         253 2091   NARCOTICS      26 2005
6         232 2091   NARCOTICS      26 2006
7         221 2091   NARCOTICS      26 2007
8         225 2091   NARCOTICS      26 2008
9         246 2091   NARCOTICS      26 2009
10        208 2091   NARCOTICS      26 2010
11        178 2091   NARCOTICS      26 2011
12          1 2091   NARCOTICS      18 2012
13        205 2091   NARCOTICS      26 2012
14          2 2091   NARCOTICS      18 2013
15        195 2091   NARCOTICS      26 2013
16         27 2091   NARCOTICS      18 2014
17        166 2091   NARCOTICS      26 2014
18        136 2091   NARCOTICS      18 2015
19         28 2091   NARCOTICS      26 2015
20        123 2091   NARCOTICS      18 2016
21        113 2091   NARCOTICS      18 2017
22        105 2091   NARCOTICS      18 2018
23        106 2091   NARCOTICS      18 2019
```

```
24          83 2091   NARCOTICS      18 2020
25          61 2091   NARCOTICS      18 2021
26          48 2091   NARCOTICS      18 2022
27          29 2091   NARCOTICS      18 2023
28          15 2091   NARCOTICS      18 2024
29           6 2091   NARCOTICS      18 2025
30        1675 2092   NARCOTICS      26 2001
31        2373 2092   NARCOTICS      26 2002
32        2775 2092   NARCOTICS      26 2003
33        3094 2092   NARCOTICS      26 2004
34        3130 2092   NARCOTICS      26 2005
35        3049 2092   NARCOTICS      26 2006
36        2726 2092   NARCOTICS      26 2007
37        1523 2092   NARCOTICS      26 2008
38        1435 2092   NARCOTICS      26 2009
39        1056 2092   NARCOTICS      26 2010
40         767 2092   NARCOTICS      26 2011
41         672 2092   NARCOTICS      26 2012
42         679 2092   NARCOTICS      26 2013
43         542 2092   NARCOTICS      26 2014
44         126 2092   NARCOTICS      18 2015
45         237 2092   NARCOTICS      26 2015
46         212 2092   NARCOTICS      18 2016
47         373 2092   NARCOTICS      18 2017
48         595 2092   NARCOTICS      18 2018
49         678 2092   NARCOTICS      18 2019
50         271 2092   NARCOTICS      18 2020
51          71 2092   NARCOTICS      18 2021
52         144 2092   NARCOTICS      18 2022
53         125 2092   NARCOTICS      18 2023
54          45 2092   NARCOTICS      18 2024
55          71 2092   NARCOTICS      18 2025
56         972 2093   NARCOTICS      26 2001
57         866 2093   NARCOTICS      26 2002
58         968 2093   NARCOTICS      26 2003
59         864 2093   NARCOTICS      26 2004
60         839 2093   NARCOTICS      26 2005
61         909 2093   NARCOTICS      26 2006
62        1033 2093   NARCOTICS      26 2007
63           2 2093   NARCOTICS      18 2008
64        1208 2093   NARCOTICS      26 2008
65           1 2093   NARCOTICS      18 2009
66        1099 2093   NARCOTICS      26 2009
```

```
67          2 2093   NARCOTICS      18 2010
68       1017 2093   NARCOTICS      26 2010
69          2 2093   NARCOTICS      18 2011
70        934 2093   NARCOTICS      26 2011
71         16 2093   NARCOTICS      18 2012
72        935 2093   NARCOTICS      26 2012
73         16 2093   NARCOTICS      18 2013
74        760 2093   NARCOTICS      26 2013
75         15 2093   NARCOTICS      18 2014
76        676 2093   NARCOTICS      26 2014
77        323 2093   NARCOTICS      18 2015
78        332 2093   NARCOTICS      26 2015
79        846 2093   NARCOTICS      18 2016
80       1000 2093   NARCOTICS      18 2017
81       1067 2093   NARCOTICS      18 2018
82       1052 2093   NARCOTICS      18 2019
83        760 2093   NARCOTICS      18 2020
84        776 2093   NARCOTICS      18 2021
85        634 2093   NARCOTICS      18 2022
86        641 2093   NARCOTICS      18 2023
87        693 2093   NARCOTICS      18 2024
88        477 2093   NARCOTICS      18 2025
```

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). 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 public way," and 2093 is for "found suspect narcotics." It appears that the CPD is now using the more specific FBI codes rather than the generic miscellaneous. The most practical decision is to use the most modern coding and use code 18 for these crimes.

A similar story applies to 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 have been coded as 20 (offenses against family). Again, it seems reasonable to use the most modern coding choice and use FBI code 20.

```
dbGetQuery(con, "
  SELECT COUNT(*) AS crimecount,
         IUCR,
         PrimaryType,
         FBICode,
         SUBSTR(Date, 7, 4) AS year
  FROM crime
  WHERE IUCR IN ('1710','1715','1725','1755','1780')
  GROUP BY IUCR, PrimaryType, year, FBICode
  ORDER BY IUCR, PrimaryType, year, FBICode")
```

```
   crimecount IUCR                PrimaryType FBICode year
1         503 1710 OFFENSE INVOLVING CHILDREN      26 2001
2         506 1710 OFFENSE INVOLVING CHILDREN      26 2002
3         479 1710 OFFENSE INVOLVING CHILDREN      26 2003
4         427 1710 OFFENSE INVOLVING CHILDREN      26 2004
5         413 1710 OFFENSE INVOLVING CHILDREN      26 2005
6         392 1710 OFFENSE INVOLVING CHILDREN      26 2006
7         403 1710 OFFENSE INVOLVING CHILDREN      26 2007
8         337 1710 OFFENSE INVOLVING CHILDREN      26 2008
9         374 1710 OFFENSE INVOLVING CHILDREN      26 2009
10        362 1710 OFFENSE INVOLVING CHILDREN      26 2010
11          1 1710 OFFENSE INVOLVING CHILDREN      20 2011
12        331 1710 OFFENSE INVOLVING CHILDREN      26 2011
13          1 1710 OFFENSE INVOLVING CHILDREN      20 2012
14        333 1710 OFFENSE INVOLVING CHILDREN      26 2012
15          6 1710 OFFENSE INVOLVING CHILDREN      20 2013
16        270 1710 OFFENSE INVOLVING CHILDREN      26 2013
17          2 1710 OFFENSE INVOLVING CHILDREN      20 2014
18        315 1710 OFFENSE INVOLVING CHILDREN      26 2014
19         22 1710 OFFENSE INVOLVING CHILDREN      20 2015
20        265 1710 OFFENSE INVOLVING CHILDREN      26 2015
21        276 1710 OFFENSE INVOLVING CHILDREN      20 2016
22          8 1710 OFFENSE INVOLVING CHILDREN      26 2016
23        328 1710 OFFENSE INVOLVING CHILDREN      20 2017
24        334 1710 OFFENSE INVOLVING CHILDREN      20 2018
25        384 1710 OFFENSE INVOLVING CHILDREN      20 2019
26        289 1710 OFFENSE INVOLVING CHILDREN      20 2020
27        261 1710 OFFENSE INVOLVING CHILDREN      20 2021
28        289 1710 OFFENSE INVOLVING CHILDREN      20 2022
29        262 1710 OFFENSE INVOLVING CHILDREN      20 2023
30        349 1710 OFFENSE INVOLVING CHILDREN      20 2024
31        219 1710 OFFENSE INVOLVING CHILDREN      20 2025
32          4 1715 OFFENSE INVOLVING CHILDREN      26 2003
33          1 1715 OFFENSE INVOLVING CHILDREN      26 2006
34          1 1715 OFFENSE INVOLVING CHILDREN      26 2007
35          3 1715 OFFENSE INVOLVING CHILDREN      26 2008
36          2 1715 OFFENSE INVOLVING CHILDREN      26 2009
37          3 1715 OFFENSE INVOLVING CHILDREN      26 2010
```

```
38             2 1715 OFFENSE INVOLVING CHILDREN      26 2011
39             4 1715 OFFENSE INVOLVING CHILDREN      26 2012
40             1 1715 OFFENSE INVOLVING CHILDREN      26 2013
41             1 1715 OFFENSE INVOLVING CHILDREN      26 2015
42             1 1715 OFFENSE INVOLVING CHILDREN      20 2016
43             1 1715 OFFENSE INVOLVING CHILDREN      20 2017
44             2 1715 OFFENSE INVOLVING CHILDREN      20 2018
45             3 1715 OFFENSE INVOLVING CHILDREN      20 2019
46             2 1715 OFFENSE INVOLVING CHILDREN      20 2020
47             2 1715 OFFENSE INVOLVING CHILDREN      20 2021
48             1 1715 OFFENSE INVOLVING CHILDREN      20 2024
49             2 1725 OFFENSE INVOLVING CHILDREN      26 2002
50             4 1725 OFFENSE INVOLVING CHILDREN      26 2003
51             1 1725 OFFENSE INVOLVING CHILDREN      26 2004
52             5 1725 OFFENSE INVOLVING CHILDREN      26 2005
53             4 1725 OFFENSE INVOLVING CHILDREN      26 2006
54             9 1725 OFFENSE INVOLVING CHILDREN      26 2007
55             4 1725 OFFENSE INVOLVING CHILDREN      26 2008
56             3 1725 OFFENSE INVOLVING CHILDREN      26 2009
57            16 1725 OFFENSE INVOLVING CHILDREN      26 2010
58             9 1725 OFFENSE INVOLVING CHILDREN      26 2011
59             7 1725 OFFENSE INVOLVING CHILDREN      26 2012
60            12 1725 OFFENSE INVOLVING CHILDREN      26 2013
61             2 1725 OFFENSE INVOLVING CHILDREN      20 2014
62            12 1725 OFFENSE INVOLVING CHILDREN      26 2014
63             1 1725 OFFENSE INVOLVING CHILDREN      20 2015
64             9 1725 OFFENSE INVOLVING CHILDREN      26 2015
65             4 1725 OFFENSE INVOLVING CHILDREN      20 2016
66            15 1725 OFFENSE INVOLVING CHILDREN      20 2017
67             6 1725 OFFENSE INVOLVING CHILDREN      20 2018
68             7 1725 OFFENSE INVOLVING CHILDREN      20 2019
69             5 1725 OFFENSE INVOLVING CHILDREN      20 2020
70             1 1725 OFFENSE INVOLVING CHILDREN      20 2021
71             1 1725 OFFENSE INVOLVING CHILDREN      20 2022
72             4 1725 OFFENSE INVOLVING CHILDREN      20 2023
73             5 1725 OFFENSE INVOLVING CHILDREN      20 2024
74             3 1725 OFFENSE INVOLVING CHILDREN      20 2025
75            37 1755 OFFENSE INVOLVING CHILDREN      26 2002
76            75 1755 OFFENSE INVOLVING CHILDREN      26 2003
77            69 1755 OFFENSE INVOLVING CHILDREN      26 2004
78            64 1755 OFFENSE INVOLVING CHILDREN      26 2005
79            70 1755 OFFENSE INVOLVING CHILDREN      26 2006
80            59 1755 OFFENSE INVOLVING CHILDREN      26 2007
```

```
81          49 1755 OFFENSE INVOLVING CHILDREN      26 2008
82          34 1755 OFFENSE INVOLVING CHILDREN      26 2009
83          52 1755 OFFENSE INVOLVING CHILDREN      26 2010
84          52 1755 OFFENSE INVOLVING CHILDREN      26 2011
85          39 1755 OFFENSE INVOLVING CHILDREN      26 2012
86          49 1755 OFFENSE INVOLVING CHILDREN      26 2013
87          43 1755 OFFENSE INVOLVING CHILDREN      26 2014
88           3 1755 OFFENSE INVOLVING CHILDREN      20 2015
89          32 1755 OFFENSE INVOLVING CHILDREN      26 2015
90          32 1755 OFFENSE INVOLVING CHILDREN      20 2016
91          46 1755 OFFENSE INVOLVING CHILDREN      20 2017
92          29 1755 OFFENSE INVOLVING CHILDREN      20 2018
93          38 1755 OFFENSE INVOLVING CHILDREN      20 2019
94          36 1755 OFFENSE INVOLVING CHILDREN      20 2020
95          36 1755 OFFENSE INVOLVING CHILDREN      20 2021
96          30 1755 OFFENSE INVOLVING CHILDREN      20 2022
97          59 1755 OFFENSE INVOLVING CHILDREN      20 2023
98          53 1755 OFFENSE INVOLVING CHILDREN      20 2024
99          41 1755 OFFENSE INVOLVING CHILDREN      20 2025
100         11 1780 OFFENSE INVOLVING CHILDREN      26 2001
101        166 1780 OFFENSE INVOLVING CHILDREN      26 2002
102        352 1780 OFFENSE INVOLVING CHILDREN      26 2003
103        559 1780 OFFENSE INVOLVING CHILDREN      26 2004
104        465 1780 OFFENSE INVOLVING CHILDREN      26 2005
105        504 1780 OFFENSE INVOLVING CHILDREN      26 2006
106        613 1780 OFFENSE INVOLVING CHILDREN      26 2007
107        624 1780 OFFENSE INVOLVING CHILDREN      26 2008
108        658 1780 OFFENSE INVOLVING CHILDREN      26 2009
109          2 1780 OFFENSE INVOLVING CHILDREN      20 2010
110        616 1780 OFFENSE INVOLVING CHILDREN      26 2010
111          1 1780 OFFENSE INVOLVING CHILDREN      20 2011
112        649 1780 OFFENSE INVOLVING CHILDREN      26 2011
113          2 1780 OFFENSE INVOLVING CHILDREN      20 2012
114        628 1780 OFFENSE INVOLVING CHILDREN      26 2012
115          1 1780 OFFENSE INVOLVING CHILDREN      20 2013
116        628 1780 OFFENSE INVOLVING CHILDREN      26 2013
117          2 1780 OFFENSE INVOLVING CHILDREN      20 2014
118        608 1780 OFFENSE INVOLVING CHILDREN      26 2014
119         17 1780 OFFENSE INVOLVING CHILDREN      20 2015
120        516 1780 OFFENSE INVOLVING CHILDREN      26 2015
121        540 1780 OFFENSE INVOLVING CHILDREN      20 2016
122         38 1780 OFFENSE INVOLVING CHILDREN      26 2016
123        415 1780 OFFENSE INVOLVING CHILDREN      20 2017
```

```
124         398 1780 OFFENSE INVOLVING CHILDREN      20 2018
125         341 1780 OFFENSE INVOLVING CHILDREN      20 2019
126         419 1780 OFFENSE INVOLVING CHILDREN      20 2020
127         393 1780 OFFENSE INVOLVING CHILDREN      20 2021
128         330 1780 OFFENSE INVOLVING CHILDREN      20 2022
129         260 1780 OFFENSE INVOLVING CHILDREN      20 2023
130         261 1780 OFFENSE INVOLVING CHILDREN      20 2024
131         208 1780 OFFENSE INVOLVING CHILDREN      20 2025
```

IUCR codes 1030 and 1035, which involve possession of incendiary devices, are now being coded as arson (09) rather than miscellaneous (26).

```
dbGetQuery(con, "
  SELECT COUNT(*) AS crimecount,
         IUCR,
         PrimaryType,
         FBICode,
         SUBSTR(Date,7,4) AS year
  FROM crime
  WHERE IUCR IN ('1030','1035')
  GROUP BY IUCR, PrimaryType, year, FBICode
  ORDER BY IUCR, PrimaryType, year, FBICode")
```

```
   crimecount IUCR PrimaryType FBICode year
1           6 1030       ARSON      26 2001
2           2 1030       ARSON      26 2002
3           5 1030       ARSON      26 2003
4           4 1030       ARSON      26 2004
5           3 1030       ARSON      26 2005
6           7 1030       ARSON      26 2006
7           5 1030       ARSON      26 2007
8           7 1030       ARSON      26 2008
9           5 1030       ARSON      26 2009
10          9 1030       ARSON      26 2010
11          5 1030       ARSON      26 2011
12          2 1030       ARSON      26 2012
13          6 1030       ARSON      26 2013
14          2 1030       ARSON      26 2014
15          5 1030       ARSON      26 2015
16          2 1030       ARSON      09 2016
17          1 1030       ARSON      26 2016
18          3 1030       ARSON      09 2017
```

```
19            1 1030        ARSON      09 2018
20            3 1030        ARSON      09 2019
21            4 1030        ARSON      09 2020
22            9 1030        ARSON      09 2021
23            4 1030        ARSON      09 2022
24            7 1030        ARSON      09 2023
25            4 1030        ARSON      09 2024
26            7 1030        ARSON      09 2025
27            7 1035        ARSON      26 2002
28            2 1035        ARSON      26 2004
29            3 1035        ARSON      26 2005
30            8 1035        ARSON      26 2006
31            6 1035        ARSON      26 2007
32            6 1035        ARSON      26 2008
33            4 1035        ARSON      26 2009
34            1 1035        ARSON      26 2010
35            1 1035        ARSON      26 2011
36            1 1035        ARSON      26 2012
37            1 1035        ARSON      09 2016
```

This all points to a modernization of FBI codes where Chicago adopted more specific FBI codes rather than placing them in the miscellaneous category.

Lastly, there are some inconsistent spellings of primary crime types. 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.

```r
dbGetQuery(con,
   "SELECT COUNT(*) AS crimecount,
           IUCR,
           PrimaryType,
           FBICode,
           SUBSTR(Date, 7, 4) AS year
    FROM crime
    WHERE IUCR='5114'
    GROUP BY IUCR, PrimaryType, FBICode, year")
```

```
  crimecount IUCR    PrimaryType FBICode 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         15 5114   NON-CRIMINAL      26 2018
9          1 5114   NON-CRIMINAL      26 2019
```

Criminal sexual assault also has an inconsistent spelling.

```
dbGetQuery(con, "
  SELECT COUNT(*) AS crimcount,
         PrimaryType,
         year
  FROM crime
  WHERE iucr IN ('0261','0263','0264','0265','0266','0271','0281','0291')
  GROUP BY PrimaryType, year
  ORDER BY year")
```

```
   crimcount              PrimaryType Year
1       1712      CRIM SEXUAL ASSAULT 2001
2         42 CRIMINAL SEXUAL ASSAULT 2001
3       1740      CRIM SEXUAL ASSAULT 2002
4         38 CRIMINAL SEXUAL ASSAULT 2002
5       1532      CRIM SEXUAL ASSAULT 2003
6         53 CRIMINAL SEXUAL ASSAULT 2003
7       1495      CRIM SEXUAL ASSAULT 2004
8         56 CRIMINAL SEXUAL ASSAULT 2004
9       1485      CRIM SEXUAL ASSAULT 2005
10        52 CRIMINAL SEXUAL ASSAULT 2005
11      1402      CRIM SEXUAL ASSAULT 2006
12        59 CRIMINAL SEXUAL ASSAULT 2006
13      1469      CRIM SEXUAL ASSAULT 2007
14        66 CRIMINAL SEXUAL ASSAULT 2007
15      1477      CRIM SEXUAL ASSAULT 2008
16        65 CRIMINAL SEXUAL ASSAULT 2008
17      1366      CRIM SEXUAL ASSAULT 2009
18        59 CRIMINAL SEXUAL ASSAULT 2009
19      1291      CRIM SEXUAL ASSAULT 2010
20        78 CRIMINAL SEXUAL ASSAULT 2010
21      1414      CRIM SEXUAL ASSAULT 2011
22        74 CRIMINAL SEXUAL ASSAULT 2011
23      1360      CRIM SEXUAL ASSAULT 2012
24        89 CRIMINAL SEXUAL ASSAULT 2012
```

```
25      1224      CRIM SEXUAL ASSAULT 2013
26       103 CRIMINAL SEXUAL ASSAULT 2013
27      1275      CRIM SEXUAL ASSAULT 2014
28       104 CRIMINAL SEXUAL ASSAULT 2014
29      1311      CRIM SEXUAL ASSAULT 2015
30       131 CRIMINAL SEXUAL ASSAULT 2015
31      1453      CRIM SEXUAL ASSAULT 2016
32       156 CRIMINAL SEXUAL ASSAULT 2016
33      1453      CRIM SEXUAL ASSAULT 2017
34       229 CRIMINAL SEXUAL ASSAULT 2017
35      1364      CRIM SEXUAL ASSAULT 2018
36       364 CRIMINAL SEXUAL ASSAULT 2018
37       884      CRIM SEXUAL ASSAULT 2019
38       771 CRIMINAL SEXUAL ASSAULT 2019
39        75      CRIM SEXUAL ASSAULT 2020
40      1169 CRIMINAL SEXUAL ASSAULT 2020
41      1516 CRIMINAL SEXUAL ASSAULT 2021
42      1591 CRIMINAL SEXUAL ASSAULT 2022
43      1646 CRIMINAL SEXUAL ASSAULT 2023
44      1576 CRIMINAL SEXUAL ASSAULT 2024
45      1034 CRIMINAL SEXUAL ASSAULT 2025
```

The conclusion of all of this is that if there is any inconsistency in the connection between `IUCR`, `PrimaryType`, and `FBICode`, then we should choose the most recent combination and delete the rest as options. The following SQL query finds for each IUCR the most recent year that it occurred in the dataset. Not all codes appear in the most recent year. Several IUCR codes last occurred before 2015.

```
dbGetQuery(con, "
    SELECT IUCR, MAX(year) AS maxyear
    FROM crime
    GROUP BY IUCR")
```

```
    IUCR maxyear
1   0110    2025
2   0130    2022
3   0141    2022
4   0142    2025
5   0261    2025
6   0262    2025
7   0263    2025
8   0264    2025
```

9	0265	2025
10	0266	2025
11	0271	2025
12	0272	2024
13	0273	2025
14	0274	2024
15	0275	2025
16	0281	2025
17	0291	2025
18	0312	2025
19	0313	2025
20	031A	2025
21	031B	2025
22	0320	2025
23	0325	2025
24	0326	2025
25	0330	2025
26	0331	2025
27	0334	2025
28	0337	2025
29	033A	2025
30	033B	2025
31	0340	2025
32	041A	2025
33	041B	2025
34	0420	2025
35	0430	2025
36	0440	2025
37	0450	2025
38	0451	2021
39	0452	2025
40	0453	2025
41	0454	2025
42	0460	2025
43	0461	2025
44	0462	2025
45	0470	2025
46	0475	2025
47	0479	2025
48	0480	2021
49	0481	2015
50	0482	2025
51	0483	2025

52	0484	2025
53	0485	2025
54	0486	2025
55	0487	2025
56	0488	2025
57	0489	2024
58	0490	2006
59	0492	2005
60	0493	2006
61	0494	2006
62	0495	2025
63	0496	2025
64	0497	2025
65	0498	2025
66	0499	2009
67	0510	2020
68	051A	2025
69	051B	2025
70	0520	2025
71	0530	2025
72	0545	2025
73	0550	2025
74	0551	2025
75	0552	2025
76	0553	2025
77	0554	2025
78	0555	2025
79	0556	2025
80	0557	2025
81	0558	2025
82	0560	2025
83	0580	2025
84	0581	2025
85	0583	2025
86	0584	2025
87	0585	2018
88	0610	2025
89	0620	2025
90	0630	2025
91	0650	2025
92	0710	2025
93	0760	2025
94	0810	2025

95	0820	2025
96	0830	2016
97	0840	2014
98	0841	2014
99	0842	2014
100	0843	2014
101	0850	2025
102	0860	2025
103	0865	2025
104	0870	2025
105	0880	2025
106	0890	2025
107	0895	2025
108	0910	2025
109	0915	2025
110	0917	2025
111	0918	2025
112	0920	2025
113	0925	2025
114	0927	2025
115	0928	2025
116	0930	2025
117	0935	2025
118	0937	2025
119	0938	2025
120	1010	2025
121	1020	2025
122	1025	2025
123	1030	2025
124	1035	2016
125	1050	2025
126	1055	2025
127	1090	2025
128	1101	2025
129	1102	2025
130	1110	2025
131	1120	2025
132	1121	2025
133	1122	2025
134	1130	2025
135	1135	2025
136	1140	2025
137	1145	2025

138	1147	2025
139	1150	2025
140	1151	2025
141	1152	2025
142	1153	2025
143	1154	2025
144	1155	2025
145	1156	2025
146	1160	2018
147	1170	2025
148	1185	2025
149	1187	2025
150	1192	2025
151	1195	2025
152	1197	2025
153	1199	2025
154	1200	2025
155	1205	2025
156	1206	2025
157	1210	2025
158	1220	2025
159	1230	2023
160	1235	2024
161	1240	2025
162	1241	2025
163	1242	2025
164	1245	2025
165	1255	2018
166	1260	2025
167	1261	2025
168	1262	2025
169	1263	2025
170	1265	2024
171	1305	2025
172	1310	2025
173	1320	2025
174	1330	2025
175	1335	2025
176	1340	2025
177	1345	2025
178	1350	2025
179	1360	2025
180	1365	2025

181	1370	2025
182	1375	2025
183	141A	2025
184	141B	2025
185	141C	2025
186	142A	2025
187	142B	2025
188	1435	2025
189	143A	2025
190	143B	2025
191	143C	2025
192	1440	2008
193	1450	2025
194	1460	2025
195	1476	2023
196	1477	2025
197	1478	2025
198	1479	2025
199	1480	2025
200	1481	2025
201	1504	2025
202	1505	2025
203	1506	2025
204	1507	2025
205	1510	2017
206	1511	2020
207	1512	2024
208	1513	2025
209	1515	2023
210	1518	2025
211	1519	2025
212	1520	2025
213	1521	2006
214	1525	2019
215	1526	2018
216	1530	2024
217	1531	2025
218	1535	2025
219	1536	2025
220	1537	2024
221	1540	2025
222	1541	2025
223	1542	2015

224	1544	2025
225	1549	2025
226	1562	2025
227	1563	2025
228	1564	2024
229	1565	2025
230	1566	2025
231	1570	2025
232	1572	2007
233	1573	2025
234	1574	2024
235	1576	2024
236	1577	2025
237	1578	2005
238	1580	2023
239	1581	2025
240	1582	2025
241	1585	2025
242	1590	2025
243	1599	2025
244	1610	2008
245	1611	2012
246	1620	2008
247	1621	2009
248	1622	2008
249	1624	2008
250	1625	2005
251	1626	2009
252	1627	2011
253	1630	2007
254	1631	2011
255	1633	2004
256	1640	2008
257	1650	2008
258	1651	2021
259	1661	2025
260	1670	2024
261	1680	2025
262	1681	2006
263	1682	2022
264	1697	2002
265	1710	2025
266	1715	2024

267	1720	2025
268	1725	2025
269	1726	2025
270	1750	2025
271	1751	2025
272	1752	2025
273	1753	2025
274	1754	2025
275	1755	2025
276	1780	2025
277	1790	2025
278	1791	2025
279	1792	2025
280	1811	2025
281	1812	2025
282	1821	2025
283	1822	2025
284	1840	2025
285	1850	2025
286	1860	2023
287	1900	2025
288	2010	2025
289	2011	2025
290	2012	2025
291	2013	2025
292	2014	2025
293	2015	2025
294	2016	2025
295	2017	2025
296	2018	2025
297	2019	2025
298	2020	2025
299	2021	2025
300	2022	2025
301	2023	2025
302	2024	2025
303	2025	2025
304	2026	2025
305	2027	2025
306	2028	2025
307	2029	2025
308	2030	2024
309	2031	2025

310	2032	2025
311	2033	2025
312	2034	2025
313	2040	2025
314	2050	2025
315	2060	2016
316	2070	2022
317	2080	2024
318	2090	2025
319	2091	2025
320	2092	2025
321	2093	2025
322	2094	2024
323	2095	2025
324	2110	2025
325	2111	2014
326	2120	2007
327	2160	2025
328	2170	2025
329	2210	2025
330	2220	2025
331	2230	2025
332	2240	2023
333	2250	2025
334	2251	2023
335	2820	2025
336	2825	2025
337	2826	2025
338	2830	2025
339	2840	2025
340	2850	2025
341	2851	2025
342	2860	2025
343	2870	2025
344	2890	2025
345	2895	2024
346	2896	2025
347	2900	2025
348	3000	2025
349	3100	2025
350	3200	2025
351	3300	2025
352	3400	2020

353	3610	2024
354	3710	2025
355	3720	2020
356	3730	2025
357	3731	2025
358	3740	2016
359	3750	2025
360	3751	2021
361	3760	2025
362	3770	2016
363	3800	2025
364	3910	2024
365	3920	2025
366	3960	2025
367	3961	2024
368	3966	2022
369	3970	2024
370	3975	2016
371	3980	2019
372	4210	2025
373	4220	2025
374	4230	2025
375	4240	2025
376	4255	2025
377	4310	2025
378	4386	2025
379	4387	2025
380	4388	2025
381	4389	2025
382	4510	2024
383	4625	2025
384	4650	2025
385	4651	2025
386	4652	2025
387	4740	2025
388	4750	2021
389	4800	2025
390	4810	2025
391	4860	2025
392	5000	2025
393	5001	2025
394	5002	2025
395	5003	2025

```
396 5004     2025
397 5005     2002
398 5007     2025
399 5008     2013
400 5009     2025
401 500E     2025
402 500N     2025
403 5011     2025
404 5013     2025
405 501A     2025
406 501H     2025
407 502P     2025
408 502R     2025
409 502T     2025
410 5073     2018
411 5093     2018
412 5094     2017
413 5110     2025
414 5111     2025
415 5112     2025
416 5113     2017
417 5114     2019
418 5120     2018
419 5121     2024
420 5122     2025
421 5130     2025
422 5131     2025
423 5132     2025
424 9901     2001
```

Now that we have a query that tells us the most recent year for each IUCR code, we should look up what the `PrimaryType` and `FBICode` are for each `IUCR` in its most recent year. We are going to temporarily create a table with the results from the previous query using "Common Table Expressions" (CTE). A CTE is a temporary table that only lasts for the one query in which it is created. You can have multiple CTEs in one query. Also, here we have our first encounter with a `JOIN`. We will cover more about `JOIN` later in these notes. For now, study the query and see how it solves our problem. With a CTE (the part following the keyword `WITH`) we create a temporary table called `recentIUCR` that has two columns, `IUCR` and `maxyear`. Then the main query looks for rows in the `crime` table that match the rows in `recentIUCR`. When it finds a match, it merges in that crime's `PrimaryType` and `FBICode.` Since many crimes with the same value of `IUCR` show up, we use `DISTINCT` to keep just the unique combinations.

```
iucrLookupTable <- dbGetQuery(con, "
  WITH
     recentIUCR AS
        (SELECT IUCR, MAX(year) AS maxyear
         FROM crime
         GROUP BY IUCR)
  SELECT DISTINCT crime.IUCR,
                  crime.PrimaryType,
                  crime.FBICode
  FROM crime
  INNER JOIN recentIUCR
      ON crime.iucr = recentIUCR.iucr AND
         crime.year = recentIUCR.maxyear
  ORDER BY crime.IUCR
")

# check for a few IUCRs
iucrLookupTable |>
   filter(IUCR %in% c(2091,2092,2093,1030,1035,5114))
```

```
  IUCR  PrimaryType FBICode
1 1030        ARSON      09
2 1035        ARSON      09
3 2091    NARCOTICS      18
4 2092    NARCOTICS      18
5 2093    NARCOTICS      18
6 5114 NON-CRIMINAL      26
```

```
# make sure that each IUCR code shows up in only one row
#   should be empty
iucrLookupTable |>
   count(IUCR) |>
   filter(n > 1)
```

```
[1] IUCR n
<0 rows> (or 0-length row.names)
```

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 `iucrLookupTable` to the database, creating a new table called `iucr`.

```
# remove iucr table if it is there already
if(dbExistsTable(con,"iucr")) dbRemoveTable(con, "iucr")

# import the data frame into SQLite
dbWriteTable(con, "iucr", iucrLookupTable,
             row.names=FALSE)

# check
dbListFields(con,"iucr")
```

```
[1] "IUCR"        "PrimaryType" "FBICode"
```

```
# check whether the table looks correct
dbGetQuery(con, "SELECT * FROM iucr LIMIT 5")
```

```
  IUCR             PrimaryType FBICode
1 0110                HOMICIDE     01A
2 0130                HOMICIDE     01A
3 0141                HOMICIDE     01B
4 0142                HOMICIDE     01B
5 0261 CRIMINAL SEXUAL ASSAULT      02
```

Everything looks correct!

Note that we ran a SQL query to pull this lookup table into `iucrLookupTable`, then we wrote that table back to the database with `dbWriteTable()`. There really 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 directly in our database.

```
# remove iucr table if it is there already
if(dbExistsTable(con,"iucr")) dbRemoveTable(con, "iucr")

# use dbExecute() since we are creating a table, not retrieving data
dbExecute(con, "
  CREATE TABLE iucr AS
  WITH
     recentIUCR AS
        (SELECT IUCR, MAX(year) AS maxyear
         FROM crime
         GROUP BY IUCR)
  SELECT DISTINCT crime.IUCR, crime.PrimaryType, crime.FBICode
  FROM crime
  INNER JOIN recentIUCR
      ON crime.iucr = recentIUCR.iucr AND
         crime.year = recentIUCR.maxyear
  ORDER BY crime.IUCR
")
```

```
[1] 0
```

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

```
dbListTables(con)
```

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

1.1 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

2 SQL dates

SQLite has no special date/time data type. The `Date` column is currently stored in the `crime` table as plain 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).

```
dbGetQuery(con, "PRAGMA table_info(crime)")
```

```
   cid                 name   type notnull dflt_value pk
1    0                   ID    INT       0         NA  0
2    1           CaseNumber   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          PrimaryType   TEXT       0         NA  0
7    6          Description   TEXT       0         NA  0
8    7 LocationDescription   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        CommunityArea   TEXT       0         NA  0
15  14              FBICode   TEXT       0         NA  0
16  15          XCoordinate    INT       0         NA  0
17  16          YCoordinate    INT       0         NA  0
18  17                 Year    INT       0         NA  0
19  18            UpdatedOn   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 of the year in which 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 data frame 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.

```r
library(lubridate)
data <- dbGetQuery(con, "SELECT ID, Date FROM crime")
data |> head()
```

```
        ID                    Date
```

```
1 13311263 07/29/2022 03:39:00 AM
2 13053066 01/03/2023 04:44:00 PM
3 12131221 08/10/2020 09:45:00 AM
4 11227634 08/26/2017 10:00:00 AM
5 13203321 09/06/2023 05:00:00 PM
6 13204489 09/06/2023 11:00:00 AM
```

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 <- data |>
   mutate(datefix = mdy_hms(Date),
          datefix = as.character(datefix)) |> # convert to plain text
   # delete the original date from the data frame
   select(-Date)

# check that the reformatting worked
data |> head()
```

```
        ID             datefix
1 13311263 2022-07-29 03:39:00
2 13053066 2023-01-03 16:44:00
3 12131221 2020-08-10 09:45:00
4 11227634 2017-08-26 10:00:00
5 13203321 2023-09-06 17:00:00
6 13204489 2023-09-06 11:00:00
```

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"
```

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()`. Early programming language compilers limited functions to at most eight characters, so programmers got rather creative in shrinking complicated function descriptions down to eight characters.

The `STRFTIME()` function has two primary arguments (and some optional modifiers). The first is a format parameter in which you tell `STRFTIME()` what you want it to extract from the date. The second argument 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.

```
a <- dbGetQuery(con, "
  SELECT ID,
         datefix,
         STRFTIME('%w',datefix) AS weekday
  FROM DateFix")
a |> head()
```

```
        ID             datefix weekday
1 13311263 2022-07-29 03:39:00       5
2 13053066 2023-01-03 16:44:00       2
3 12131221 2020-08-10 09:45:00       1
4 11227634 2017-08-26 10:00:00       6
5 13203321 2023-09-06 17:00:00       3
6 13204489 2023-09-06 11:00:00       3
```

For the first date, 2022-07-29, `STRFTIME()` tells us that this was day 5 of the week, which is Friday (remember that 0 is Sunday).

`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 after noon.

```
dbGetQuery(con, "
  SELECT COUNT(*) as crimecount,
         CAST(STRFTIME('%w',datefix) AS INTEGER) AS weekday
  FROM DateFix
  WHERE (weekday>=1) AND (weekday<=5) AND
        (CAST(STRFTIME('%H',datefix) AS INTEGER) >= 12)
  GROUP BY weekday")
```

```
  crimecount weekday
1     746915       1
2     768661       2
3     774035       3
4     760030       4
5     810449       5
```

In the `SELECT` clause, we told SQLite to store the weekday as an integer. In the `WHERE` clause we extracted the hour (24-hour clock) so that we could make a numerical comparison with the number 12.

3 Creating the final table

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

Removing columns from tables in SQLite used to not be simple. Only after March 2021 could you run `ALTER TABLE crime DROP COLUMN Date` to remove a single column. We are going to use an old-school approach since we are going to make many changes to our database. We are going to rename the current `crime` table, then copy only the columns we want into a new `crime` table, while at the same time replacing the old format dates with dates in a more preferable format.

First, rename the `crime` table to `crime_old`, which we will delete as soon as we are done.

```
dbExecute(con, "ALTER TABLE crime RENAME TO crime_old")
```

```
[1] 0
```

There should be a new table.

```
dbListTables(con)
```

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

This will create our new `crime` table. It can take a few minutes.

```
dbExecute(con, "
   CREATE TABLE crime AS
   SELECT crime_old.ID,
          crime_old.CaseNumber,
          DateFix.datefix AS date,
          crime_old.Block,
          crime_old.IUCR,
          crime_old.Description,
          crime_old.LocationDescription,
          crime_old.Arrest,
          crime_old.Domestic,
          crime_old.Beat,
          crime_old.District,
          crime_old.Ward,
          crime_old.CommunityArea,
          crime_old.Latitude,
          crime_old.Longitude
   FROM crime_old
      INNER JOIN DateFix
        ON crime_old.ID=DateFix.ID")
```

```
[1] 0
```

This query requires a bit of discussion. First, note that the `FROM` clause joins two tables, `crime_old` and `DateFix`. The `ON` 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 have 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`. Are you not glad we fixed this way back when we first created our database? When we were cleaning up the Chicago crime CSV file we ran this code on the first line in the CSV file.

```
readLines(infile, n=1) |>
   gsub(",", ";", x=_) |> # separate with ;
   gsub(" ", "", x=_)  |> # SQL doesn't like field names with .,-,space
   writeLines(con=outfile)
```

R typically renames column names with spaces by replacing the spaces with periods. Right at the beginning we deleted any spaces in column names so that we get `CaseNumber` instead of `Case Number` or `Case.Number`.

Technically, `Beat`, `District`, `Ward`, and `CommunityArea` 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 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.

```
dbGetQuery(con, "
  SELECT *
  FROM crime
  LIMIT 10")
```

```
         ID CaseNumber                date                Block IUCR
1  13311263   JG503434 2022-07-29 03:39:00          023XX S TROY ST 1582
2  13053066   JG103252 2023-01-03 16:44:00 039XX W WASHINGTON BLVD 2017
3  12131221   JD327000 2020-08-10 09:45:00        015XX N DAMEN AVE 0326
4  11227634   JB147599 2017-08-26 10:00:00      001XX W RANDOLPH ST 0281
5  13203321   JG415333 2023-09-06 17:00:00         002XX N Wells st 1320
6  13204489   JG416325 2023-09-06 11:00:00           0000X E 8TH ST 0810
7  11695116   JC272771 2019-05-21 08:20:00   018XX S CALIFORNIA AVE 0620
8  12419690   JE295655 2021-07-07 10:30:00    132XX S GREENWOOD AVE 1544
9  12729745   JF279458 2022-06-14 14:47:00      035XX N CENTRAL AVE 0340
10 12835559   JF406130 2022-09-21 22:00:00         004XX E 69TH ST 0910
                    Description                    LocationDescription Arrest
1               CHILD PORNOGRAPHY                            RESIDENCE   true
2   MANUFACTURE / DELIVER - CRACK                             SIDEWALK   true
3  AGGRAVATED VEHICULAR HIJACKING                               STREET   true
4                  NON-AGGRAVATED                          HOTEL/MOTEL  false
5                      TO VEHICLE PARKING LOT / GARAGE (NON RESIDENTIAL)  false
```

```
6                           OVER $500 PARKING LOT / GARAGE (NON RESIDENTIAL)  false
7                     UNLAWFUL ENTRY                              RESIDENCE  false
8   SEXUAL EXPLOITATION OF A CHILD                              RESIDENCE  false
9   ATTEMPT STRONG ARM - NO WEAPON                                   BANK   true
10                        AUTOMOBILE                        OTHER (SPECIFY)   true
   Domestic Beat District Ward CommunityArea Latitude Longitude
1     false 1033      010   25            30       NA        NA
2     false 1122      011   28            26       NA        NA
3     false 1424      014    1            24 41.90842 -87.67741
4     false  122      001   42            32       NA        NA
5     false  122      001   42            32 41.88602 -87.63394
6     false  123      001    4            32 41.87183 -87.62615
7     false 1023      010   25            29 41.85655 -87.69560
8     false  533      005   10            54 41.65512 -87.59488
9     false 1633      016   30            15 41.94523 -87.76673
10    false  322      003    6            69 41.76935 -87.61501
```

Note that the dates are formatted properly and both `PrimaryType` and `FBICode` have been eliminated from the table. If everything looks as expected, then we can delete the `crime_old` and the `DateFix` tables.

```r
dbExecute(con, "DROP TABLE crime_old")
```

```
[1] 0
```

```r
dbExecute(con, "DROP TABLE DateFix")
```

```
[1] 0
```

```r
dbListTables(con)
```

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

After all this work, the size of the `chicagocrime.db` database file can become quite large. Our database file is now 3.4 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 does not 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 minute.

```
dbExecute(con, "VACUUM")
```

```
[1] 0
```

After `VACUUM`, our `chicagocrime.db` file is now 1.2 Gb... much better.

4 Joining data across tables

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

```
timeIUCRjoin <-
system.time(
{
   data <- dbGetQuery(con, "
       SELECT crime.CaseNumber,
              iucr.FBICode
       FROM   crime
         INNER JOIN iucr
           ON crime.iucr=iucr.iucr")
})
data |> head()
```

```
  CaseNumber FBICode
1   JG503434      17
2   JG103252      18
3   JD327000      03
4   JB147599      02
5   JG415333      14
6   JG416325      06
```

```
timeIUCRjoin
```

```
   user  system elapsed
  19.09    2.11   21.31
```

For each record in `crime`, SQLite looks up the crime's IUCR code in the `iucr` table and links in the FBI code. SQLite is fast. This query took 21.31 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. An index does not always make queries faster and can require storing a large index in some cases. Let's try this example.

```
dbExecute(con, "
   CREATE INDEX iucr_idx on iucr(iucr)")
```

```
[1] 0
```

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

```
timeIUCRjoinIndex <-
system.time(
{
   data <- dbGetQuery(con, "
       SELECT crime.CaseNumber,
              iucr.FBICode
       FROM   crime
         INNER JOIN iucr
           ON crime.iucr=iucr.iucr")
})
timeIUCRjoinIndex
```

```
   user  system elapsed 
  14.91    1.86   17.03 
```

That query now takes 17.03 seconds. Creating an index is not always worth it. If you have queries that are taking too long, it is worth experimenting with creating an index to see if it helps.

You may come across SQL queries that join two tables with a `WHERE` clause like this.

```
data <- dbGetQuery(con, "
    SELECT crime.CaseNumber,
           iucr.FBICode
    FROM crime, iucr
    WHERE crime.iucr=iucr.iucr")
```

Technically this is a legal SQL join query. However, 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. An `INNER JOIN` returns only the rows where the join keys (the columns we use to link tables like `crime.iucr`) exist in both tables. All other rows are dropped. SQL interprets joins using the WHERE clause implicitly as an `INNER JOIN`.

Generally, in social science, we do not want to drop a row simply because its IUCR code does not appear in the lookup table. We would probably rather code its `PrimaryType` and `FBICode` as missing rather than drop the row. A `LEFT JOIN` forces 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 `NA` for its `FBICode`. More precisely, `LEFT JOIN` is synonymous with a `LEFT OUTER JOIN` (the `OUTER` keyword is optional).

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.

```
dbGetQuery(con, "
    SELECT COUNT(*) AS crimecount,
           crime.Ward
    -- Use LEFT JOIN to link the two tables
    FROM crime
       LEFT JOIN iucr
          ON crime.iucr=iucr.iucr
    -- Use WHERE to filter cases we want
    WHERE iucr.PrimaryType='ASSAULT'
    GROUP BY crime.Ward")
```

```
   crimecount Ward
1       39807
2        7395    1
```

3	12065	10
4	7413	11
5	6520	12
6	5472	13
7	6540	14
8	14872	15
9	17957	16
10	21168	17
11	9332	18
12	4580	19
13	14607	2
14	20353	20
15	17680	21
16	6657	22
17	5659	23
18	20247	24
19	8053	25
20	9471	26
21	16997	27
22	23558	28
23	13488	29
24	17378	3
25	6628	30
26	6725	31
27	4245	32
28	4462	33
29	17117	34
30	6461	35
31	5242	36
32	14383	37
33	4641	38
34	4138	39
35	11962	4
36	5003	40
37	3974	41
38	12144	42
39	2859	43
40	4000	44
41	4712	45
42	6840	46
43	3556	47
44	5230	48
45	7106	49

```
46       13812    5
47        4561   50
48       20364    6
49       17963    7
50       18017    8
51       17820    9
```

Let's tabulate how many Part 1 crimes occur in each year. We will 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 a `LEFT JOIN` to link the tables.

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

```
                          type year crimecount
1                        ARSON 2001       1011
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        373
19                       ARSON 2019        376
20                       ARSON 2020        588
```

```
21                     ARSON 2021         530
22                     ARSON 2022         422
23                     ARSON 2023         513
24                     ARSON 2024         482
25                     ARSON 2025         250
26                   ASSAULT 2001        7871
27                   ASSAULT 2002        7721
28                   ASSAULT 2003        7372
29                   ASSAULT 2004        7331
30                   ASSAULT 2005        6754
31                   ASSAULT 2006        6597
32                   ASSAULT 2007        6335
33                   ASSAULT 2008        6250
34                   ASSAULT 2009        6000
35                   ASSAULT 2010        5278
36                   ASSAULT 2011        5157
37                   ASSAULT 2012        4873
38                   ASSAULT 2013        4268
39                   ASSAULT 2014        4337
40                   ASSAULT 2015        4480
41                   ASSAULT 2016        5713
42                   ASSAULT 2017        5793
43                   ASSAULT 2018        6002
44                   ASSAULT 2019        5842
45                   ASSAULT 2020        6265
46                   ASSAULT 2021        7242
47                   ASSAULT 2022        7281
48                   ASSAULT 2023        7712
49                   ASSAULT 2024        7905
50                   ASSAULT 2025        4345
51                   BATTERY 2001       16388
52                   BATTERY 2002       15196
53                   BATTERY 2003       12477
54                   BATTERY 2004       11529
55                   BATTERY 2005       11327
56                   BATTERY 2006       11001
57                   BATTERY 2007       11153
58                   BATTERY 2008       10805
59                   BATTERY 2009       10142
60                   BATTERY 2010        9432
61                   BATTERY 2011        8402
62                   BATTERY 2012        8005
63                   BATTERY 2013        6634
```

```
64                      BATTERY 2014       6577
65                      BATTERY 2015       7018
66                      BATTERY 2016       8085
67                      BATTERY 2017       7845
68                      BATTERY 2018       7734
69                      BATTERY 2019       7858
70                      BATTERY 2020       8319
71                      BATTERY 2021       8346
72                      BATTERY 2022       7495
73                      BATTERY 2023       8080
74                      BATTERY 2024       8182
75                      BATTERY 2025       4597
76                     BURGLARY 2001      26014
77                     BURGLARY 2002      25623
78                     BURGLARY 2003      25157
79                     BURGLARY 2004      24564
80                     BURGLARY 2005      25503
81                     BURGLARY 2006      24324
82                     BURGLARY 2007      24858
83                     BURGLARY 2008      26218
84                     BURGLARY 2009      26767
85                     BURGLARY 2010      26422
86                     BURGLARY 2011      26620
87                     BURGLARY 2012      22844
88                     BURGLARY 2013      17894
89                     BURGLARY 2014      14569
90                     BURGLARY 2015      13184
91                     BURGLARY 2016      14289
92                     BURGLARY 2017      13001
93                     BURGLARY 2018      11747
94                     BURGLARY 2019       9639
95                     BURGLARY 2020       8758
96                     BURGLARY 2021       6661
97                     BURGLARY 2022       7594
98                     BURGLARY 2023       7486
99                     BURGLARY 2024       8425
100                    BURGLARY 2025       5679
101    CRIMINAL SEXUAL ASSAULT 2001       1814
102    CRIMINAL SEXUAL ASSAULT 2002       1839
103    CRIMINAL SEXUAL ASSAULT 2003       1617
104    CRIMINAL SEXUAL ASSAULT 2004       1583
105    CRIMINAL SEXUAL ASSAULT 2005       1562
106    CRIMINAL SEXUAL ASSAULT 2006       1488
```

```
107    CRIMINAL SEXUAL ASSAULT 2007       1565
108    CRIMINAL SEXUAL ASSAULT 2008       1566
109    CRIMINAL SEXUAL ASSAULT 2009       1450
110    CRIMINAL SEXUAL ASSAULT 2010       1397
111    CRIMINAL SEXUAL ASSAULT 2011       1516
112    CRIMINAL SEXUAL ASSAULT 2012       1468
113    CRIMINAL SEXUAL ASSAULT 2013       1355
114    CRIMINAL SEXUAL ASSAULT 2014       1398
115    CRIMINAL SEXUAL ASSAULT 2015       1461
116    CRIMINAL SEXUAL ASSAULT 2016       1627
117    CRIMINAL SEXUAL ASSAULT 2017       1697
118    CRIMINAL SEXUAL ASSAULT 2018       1742
119    CRIMINAL SEXUAL ASSAULT 2019       1673
120    CRIMINAL SEXUAL ASSAULT 2020       1255
121    CRIMINAL SEXUAL ASSAULT 2021       1530
122    CRIMINAL SEXUAL ASSAULT 2022       1606
123    CRIMINAL SEXUAL ASSAULT 2023       1668
124    CRIMINAL SEXUAL ASSAULT 2024       1598
125    CRIMINAL SEXUAL ASSAULT 2025       1041
126                   HOMICIDE 2001        667
127                   HOMICIDE 2002        657
128                   HOMICIDE 2003        601
129                   HOMICIDE 2004        454
130                   HOMICIDE 2005        451
131                   HOMICIDE 2006        472
132                   HOMICIDE 2007        448
133                   HOMICIDE 2008        513
134                   HOMICIDE 2009        461
135                   HOMICIDE 2010        438
136                   HOMICIDE 2011        437
137                   HOMICIDE 2012        514
138                   HOMICIDE 2013        430
139                   HOMICIDE 2014        427
140                   HOMICIDE 2015        496
141                   HOMICIDE 2016        786
142                   HOMICIDE 2017        672
143                   HOMICIDE 2018        588
144                   HOMICIDE 2019        499
145                   HOMICIDE 2020        787
146                   HOMICIDE 2021        806
147                   HOMICIDE 2022        730
148                   HOMICIDE 2023        632
149                   HOMICIDE 2024        589
```

```
150                   HOMICIDE 2025        264
151        MOTOR VEHICLE THEFT 2001      27555
152        MOTOR VEHICLE THEFT 2002      25121
153        MOTOR VEHICLE THEFT 2003      22749
154        MOTOR VEHICLE THEFT 2004      22805
155        MOTOR VEHICLE THEFT 2005      22497
156        MOTOR VEHICLE THEFT 2006      21818
157        MOTOR VEHICLE THEFT 2007      18573
158        MOTOR VEHICLE THEFT 2008      18881
159        MOTOR VEHICLE THEFT 2009      15482
160        MOTOR VEHICLE THEFT 2010      19029
161        MOTOR VEHICLE THEFT 2011      19388
162        MOTOR VEHICLE THEFT 2012      16490
163        MOTOR VEHICLE THEFT 2013      12582
164        MOTOR VEHICLE THEFT 2014       9911
165        MOTOR VEHICLE THEFT 2015      10068
166        MOTOR VEHICLE THEFT 2016      11285
167        MOTOR VEHICLE THEFT 2017      11380
168        MOTOR VEHICLE THEFT 2018       9985
169        MOTOR VEHICLE THEFT 2019       8978
170        MOTOR VEHICLE THEFT 2020       9962
171        MOTOR VEHICLE THEFT 2021      10605
172        MOTOR VEHICLE THEFT 2022      21472
173        MOTOR VEHICLE THEFT 2023      29253
174        MOTOR VEHICLE THEFT 2024      21709
175        MOTOR VEHICLE THEFT 2025      10731
176 OFFENSE INVOLVING CHILDREN 2001        380
177 OFFENSE INVOLVING CHILDREN 2002        383
178 OFFENSE INVOLVING CHILDREN 2003        386
179 OFFENSE INVOLVING CHILDREN 2004        366
180 OFFENSE INVOLVING CHILDREN 2005        354
181 OFFENSE INVOLVING CHILDREN 2006        327
182 OFFENSE INVOLVING CHILDREN 2007        318
183 OFFENSE INVOLVING CHILDREN 2008        239
184 OFFENSE INVOLVING CHILDREN 2009        248
185 OFFENSE INVOLVING CHILDREN 2010        244
186 OFFENSE INVOLVING CHILDREN 2011        221
187 OFFENSE INVOLVING CHILDREN 2012        233
188 OFFENSE INVOLVING CHILDREN 2013        218
189 OFFENSE INVOLVING CHILDREN 2014        239
190 OFFENSE INVOLVING CHILDREN 2015        253
191 OFFENSE INVOLVING CHILDREN 2016        244
192 OFFENSE INVOLVING CHILDREN 2017        297
```

```
193 OFFENSE INVOLVING CHILDREN 2018        312
194 OFFENSE INVOLVING CHILDREN 2019        258
195 OFFENSE INVOLVING CHILDREN 2020        251
196 OFFENSE INVOLVING CHILDREN 2021        226
197 OFFENSE INVOLVING CHILDREN 2022        235
198 OFFENSE INVOLVING CHILDREN 2023        204
199 OFFENSE INVOLVING CHILDREN 2024        181
200 OFFENSE INVOLVING CHILDREN 2025         97
201                  RITUALISM 2001          8
202                  RITUALISM 2002          1
203                  RITUALISM 2003          1
204                  RITUALISM 2004          1
205                  RITUALISM 2005          2
206                  RITUALISM 2006          6
207                  RITUALISM 2007          1
208                  RITUALISM 2020          1
209                    ROBBERY 2001      18441
210                    ROBBERY 2002      18523
211                    ROBBERY 2003      17332
212                    ROBBERY 2004      15978
213                    ROBBERY 2005      16047
214                    ROBBERY 2006      15969
215                    ROBBERY 2007      15450
216                    ROBBERY 2008      16703
217                    ROBBERY 2009      15981
218                    ROBBERY 2010      14275
219                    ROBBERY 2011      13983
220                    ROBBERY 2012      13484
221                    ROBBERY 2013      11819
222                    ROBBERY 2014       9800
223                    ROBBERY 2015       9638
224                    ROBBERY 2016      11960
225                    ROBBERY 2017      11881
226                    ROBBERY 2018       9681
227                    ROBBERY 2019       7995
228                    ROBBERY 2020       7855
229                    ROBBERY 2021       7920
230                    ROBBERY 2022       8964
231                    ROBBERY 2023      11052
232                    ROBBERY 2024       9116
233                    ROBBERY 2025       3970
234                      THEFT 2001      99290
235                      THEFT 2002      98334
```

```
236                              THEFT 2003      98876
237                              THEFT 2004      95464
238                              THEFT 2005      85684
239                              THEFT 2006      86241
240                              THEFT 2007      85156
241                              THEFT 2008      88437
242                              THEFT 2009      80977
243                              THEFT 2010      76758
244                              THEFT 2011      75153
245                              THEFT 2012      75464
246                              THEFT 2013      71536
247                              THEFT 2014      61569
248                              THEFT 2015      57353
249                              THEFT 2016      61625
250                              THEFT 2017      64386
251                              THEFT 2018      65290
252                              THEFT 2019      62498
253                              THEFT 2020      41350
254                              THEFT 2021      40822
255                              THEFT 2022      54899
256                              THEFT 2023      57490
257                              THEFT 2024      60495
258                              THEFT 2025      35635
```

4.1 Exercises

5. Count the number of arrests for "MOTOR VEHICLE THEFT"
6. Which District has the most thefts?. You can first try doing this with a mix of SQL and R. Once you do that, try finding another solution that only uses SQL (and two CTEs in a `WITH` clause separated by a comma).

5 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 will use this method to see if addresses are always geocoded to the same coordinates. Here are the unique combinations of addresses and coordinates. We will just show the first 20.

```
dbGetQuery(con, "
   SELECT DISTINCT Block, Longitude, Latitude
   FROM crime
   LIMIT 20")
```

```
                     Block Longitude Latitude
1           023XX S TROY ST        NA       NA
2   039XX W WASHINGTON BLVD        NA       NA
3         015XX N DAMEN AVE -87.67741 41.90842
4       001XX W RANDOLPH ST        NA       NA
5          002XX N Wells st -87.63394 41.88602
6            0000X E 8TH ST -87.62615 41.87183
7    018XX S CALIFORNIA AVE -87.69560 41.85655
8     132XX S GREENWOOD AVE -87.59488 41.65512
9       035XX N CENTRAL AVE -87.76673 41.94523
10          004XX E 69TH ST -87.61501 41.76935
11        070XX S CLYDE AVE -87.57389 41.76742
12      073XX S EMERALD AVE -87.64308 41.76094
13       055XX S ALBANY AVE -87.70109 41.79261
14          040XX W 59TH ST -87.72327 41.78593
15          002XX W 47TH ST -87.63191 41.80913
16       044XX S KEDZIE AVE -87.70416 41.81281
17          004XX E 88TH ST -87.61318 41.73470
18      020XX N KIMBALL AVE -87.71191 41.91849
19    101XX S LAFAYETTE AVE -87.62480 41.71004
20        105XX S PERRY AVE -87.62578 41.70301
```

The crime table has at least one row with each of these combinations of `Block`, `Longitude`, and `Latitude`.

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

```
dbGetQuery(con, "
   SELECT COUNT(*) AS Blockcount,
          Block
   FROM
     (SELECT DISTINCT block,
                      Longitude,
                      Latitude
      FROM crime)
   GROUP BY block
```

```
  ORDER BY blockcount DESC
  LIMIT 20")
```

```
   Blockcount                 block
1         117      034XX N CLARK ST
2         108      048XX N BROADWAY
3         106     016XX W HOWARD ST
4         105    002XX N PULASKI RD
5         104   013XX W RANDOLPH ST
6         103      044XX N BROADWAY
7         100      028XX N CLARK ST
8         100      024XX N CLARK ST
9          97     010XX W ARGYLE ST
10         96      045XX N BROADWAY
11         95   045XX N SHERIDAN RD
12         94   0000X W DIVISION ST
13         93    031XX W MADISON ST
14         93      031XX S GREEN ST
15         93 015XX N KINGSBURY ST
16         93   001XX W DIVISION ST
17         92     027XX W CERMAK RD
18         90    054XX W MADISON ST
19         87    049XX W MADISON ST
20         87   008XX W RANDOLPH ST
```

Clearly, the coordinates are not unique to each address. The addresses are "rounded" to provide some privacy, but the coordinates appear to be scattered. Why? The Chicago data portal notes "This location is shifted from the actual location for partial redaction but falls on the same block."

Rather than place subqueries in the `FROM` clause, the more modern preference is to use Common Table Expressions like we did earlier. Rewritten as a CTE:

```
dbGetQuery(con, "
 WITH
    XYBlockUnique AS
      (SELECT DISTINCT block,
              Longitude,
              Latitude
       FROM crime)
 SELECT COUNT(*) AS blockcount,
        block
 FROM XYBlockUnique
 GROUP BY block
 ORDER BY blockcount DESC
 LIMIT 20")
```

```
   blockcount                block
1         117      034XX N CLARK ST
2         108      048XX N BROADWAY
3         106     016XX W HOWARD ST
4         105    002XX N PULASKI RD
5         104   013XX W RANDOLPH ST
6         103      044XX N BROADWAY
7         100      028XX N CLARK ST
8         100      024XX N CLARK ST
9          97     010XX W ARGYLE ST
10         96      045XX N BROADWAY
11         95   045XX N SHERIDAN RD
12         94   0000X W DIVISION ST
13         93    031XX W MADISON ST
14         93      031XX S GREEN ST
15         93 015XX N KINGSBURY ST
16         93   001XX W DIVISION ST
17         92     027XX W CERMAK RD
18         90    054XX W MADISON ST
19         87    049XX W MADISON ST
20         87   008XX W RANDOLPH ST
```

If you are going to use the CTE or subquery in multiple queries, then it is better to `CREATE TEMPORARY TABLE`, which we will encounter later.

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

5.1 Exercise

As a final exercise that does not involve a subquery:

7. Count the number of assaults, since 2016, that occurred on Fridays and Saturdays, after 6pm, reporting the date, day of week, hour of the day, and year

6 Solutions

1. Print out all of the rows in iucr

```
dbGetQuery(con, "
  SELECT * from iucr
  LIMIT 20")
```

```
   IUCR             PrimaryType FBICode
1  0110                HOMICIDE     01A
2  0130                HOMICIDE     01A
3  0141                HOMICIDE     01B
4  0142                HOMICIDE     01B
5  0261 CRIMINAL SEXUAL ASSAULT      02
6  0262 CRIMINAL SEXUAL ASSAULT      02
7  0263 CRIMINAL SEXUAL ASSAULT      02
8  0264 CRIMINAL SEXUAL ASSAULT      02
9  0265 CRIMINAL SEXUAL ASSAULT      02
10 0266 CRIMINAL SEXUAL ASSAULT      02
11 0271 CRIMINAL SEXUAL ASSAULT      02
12 0272 CRIMINAL SEXUAL ASSAULT      02
13 0273 CRIMINAL SEXUAL ASSAULT      02
14 0274 CRIMINAL SEXUAL ASSAULT      02
15 0275 CRIMINAL SEXUAL ASSAULT      02
16 0281 CRIMINAL SEXUAL ASSAULT      02
17 0291 CRIMINAL SEXUAL ASSAULT      02
18 0312                 ROBBERY      03
19 0313                 ROBBERY      03
20 031A                 ROBBERY      03
```

2. Print out all the IUCR codes for “KIDNAPPING”

```
dbGetQuery(con, "
   SELECT iucr
   FROM iucr
   WHERE PrimaryType='KIDNAPPING'")
```

```
  IUCR
1 1792
2 4210
3 4220
```

```
4 4230
5 4240
6 4255
```

3. How many IUCR codes are there for “ASSAULT”?

```
dbGetQuery(con, "
   SELECT *
   FROM iucr
   WHERE PrimaryType='ASSAULT'")
```

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

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

```
dbGetQuery(con, "
   SELECT COUNT(*)
   FROM iucr
   WHERE PrimaryType='ASSAULT'")
```

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

5. Count the number of arrests for “MOTOR VEHICLE THEFT”

```r
dbGetQuery(con, "
   SELECT COUNT(*) as MVTArrestCount
   FROM crime
      INNER JOIN iucr ON
         crime.iucr=iucr.iucr
   WHERE crime.Arrest='true' AND
         iucr.PrimaryType='MOTOR VEHICLE THEFT'")
```

```
  MVTArrestCount
1          32533
```

6. Which District has the most thefts?

```r
a <- dbGetQuery(con, "
   SELECT COUNT(*) AS crimecount,
          District
   FROM crime
      INNER JOIN iucr ON
         crime.iucr=iucr.iucr
   WHERE iucr.PrimaryType='THEFT'
   GROUP BY District")

a |>
   filter(crimecount==max(crimecount))
```

```
  crimecount District
1     159430      018
```

```r
# or
a |>
   slice_max(crimecount, with_ties=TRUE)
```

```
  crimecount District
1     159430      018
```

```r
# or with a CTE
dbGetQuery(con, "
WITH
   -- first CTE counts thefts by district
   DistrictCountCTE AS
```

```
      (SELECT COUNT(*) AS crimecount,
              District
       FROM crime
          INNER JOIN iucr ON
             crime.iucr=iucr.iucr
       WHERE iucr.PrimaryType='THEFT'
       GROUP BY District),
   -- second CTE finds the max theft count
   MaxCountCTE AS
      (SELECT MAX(crimecount) AS MaxCrimeCount
       FROM DistrictCountCTE)
-- main query selects the district(s) matching the max
SELECT District, crimecount
FROM DistrictCountCTE
   INNER JOIN MaxCountCTE
      ON DistrictCountCTE.crimecount = MaxCountCTE.MaxCrimeCount
")
```

```
  District crimecount
1      018     159430
```

7. Count the number of assaults, since 2016, 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
dbGetQuery(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
   LIMIT 20")
```

```
   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)
```