More Hints on SQLite

Ted Laderas

August 17, 2016

Here are some more hints about SQLite/R to get you through the in-class assignments.

Difference between dates

You can find the difference between two dates by using the `julianday()` function in SQLite.

```
library(RSQLite)
```

```
## Loading required package: DBI
```

```
con <- dbConnect(SQLite(),dbname="patient.sqlite")

sqlStatement <- "select peh.*, julianday('2015-01-01') - julianday(peh.Admit_date)
    as TimeFromNewYear FROM
    patient_encounter_hosp as peh"

queryResult <- dbGetQuery(con, sqlStatement)
queryResult[1:10,]
```

```
##    patientid Event_ID encounter_type         outcome Admit_date
## 1          1      108             22             SNF 2014-01-01
## 2          1      109             22             SNF 2014-01-13
## 3          2     1333             22 Discharged Home 2014-01-01
## 4          3       71             22             SNF 2014-01-01
## 5          4      886             22 Discharged Home 2014-01-01
## 6          5       73             22 Discharged Home 2014-01-01
## 7          5       74             22 Discharged Home 2014-01-16
## 8          6       98             22             SNF 2014-01-01
## 9          6       99             22             SNF 2014-01-19
## 10         7      893             22           Rehab 2014-01-01
##    Discharge_date   Admit_source TimeFromNewYear
## 1      2014-01-08 Emergency Room             365
## 2      2014-01-20       Transfer             353
## 3      2014-01-13         Clinic             365
## 4      2014-01-07       Transfer             365
## 5      2014-01-07 Emergency Room             365
## 6      2014-01-08 Emergency Room             365
## 7      2014-01-25 Emergency Room             350
## 8      2014-01-08 Emergency Room             365
## 9      2014-02-01       Transfer             347
## 10     2014-01-16 Emergency Room             365
```

Recoding Values

Remember, you can recode values using the case statement, and you're not just limited to one WHEN statement!

```
sqlStatement <- "select pe.patientID, pe.reason, CASE
    WHEN reason = 3 THEN 1
    WHEN reason = 5 THEN 2
    WHEN reason = 10 THEN 3
    ELSE 0 END as recoded_reason
    FROM
    patient_encounter as pe"

queryResult <- dbGetQuery(con, sqlStatement)

#show those rows that have reason = 3 (and should have recoded_reason = 1)
reason3 <- queryResult[queryResult$reason == 3,]
reason3[1:10,]
```

```
##     patientID reason recoded_reason
## 57          2      3              1
## 71          2      3              1
## 75          2      3              1
## 416        19      3              1
## 454        25      3              1
## 479        27      3              1
## 589        33      3              1
## 847        53      3              1
## 896        57      3              1
## 932        60      3              1
```

```
reason5 <- queryResult[queryResult$reason == 5,]
reason5[1:10,]
```

```
##      patientID reason recoded_reason
## 128          7      5              2
## 225         10      5              2
## 424         21      5              2
## 436         21      5              2
## 440         22      5              2
## 814         51      5              2
## 1031        64      5              2
## 1309        82      5              2
## 1621       101      5              2
## 1702       104      5              2
```

Subqueries

Subqueries can be a useful way to break up a query when it makes sense to do a query in multiple steps. We can wrap any query in `()` (parentheses), and do queries on the returned table.

For example, we can do a join, and then select columns from that join. Note that unless we name the subquery using an alias, we do not use an alias to refer to the columns in the subquery.

```
sqlStatement <- "SELECT patientID, Actual_date, encounterName FROM (
                    SELECT pe.*, te.encounterName
                      FROM patient_encounter as pe,
                      t_encounter_type as te
                      WHERE pe.encounter_type = te.t_encounter_type)"

queryResult <- dbGetQuery(con, sqlStatement)

queryResult[1:30,]
```

```
##    patientID Actual_date        encounterName
## 1          1  2013-10-15 MD/PROV Office Visit
## 2          1  2013-12-01 MD/PROV Office Visit
## 3          1  2013-05-01 MD/PROV Office Visit
## 4          1  2013-09-04 MD/PROV Office Visit
## 5          1  2013-11-23 MD/PROV Office Visit
## 6          1  2013-06-11 MD/PROV Office Visit
## 7          1  2013-09-02 MD/PROV Office Visit
## 8          1  2013-10-31 MD/PROV Office Visit
## 9          1  2013-12-08 MD/PROV Office Visit
## 10         1  2013-02-02 MD/PROV Office Visit
## 11         1  2013-08-14 MD/PROV Office Visit
## 12         1  2013-05-04 MD/PROV Office Visit
## 13         1  2013-11-19 MD/PROV Office Visit
## 14         1  2013-12-06 MD/PROV Office Visit
## 15         1  2013-11-05 MD/PROV Office Visit
## 16         1  2013-08-11 MD/PROV Office Visit
## 17         1  2013-01-03 MD/PROV Office Visit
## 18         1  2013-11-07       Emergency Room
## 19         2  2013-02-11 MD/PROV Office Visit
## 20         2  2013-11-04 MD/PROV Office Visit
## 21         2  2013-05-06 MD/PROV Office Visit
## 22         2  2013-06-16 MD/PROV Office Visit
## 23         2  2013-01-22 MD/PROV Office Visit
## 24         2  2013-01-20 MD/PROV Office Visit
## 25         2  2013-08-03 MD/PROV Office Visit
## 26         2  2013-12-05 MD/PROV Office Visit
## 27         2  2013-03-09 MD/PROV Office Visit
## 28         2  2013-11-13 MD/PROV Office Visit
## 29         2  2013-04-04 MD/PROV Office Visit
## 30         2  2013-11-27 MD/PROV Office Visit
```

GROUP BY

Oftentimes, we need to calculate something by a patient id. For example, we might want to calculate the number of visits for each patient. A `GROUP BY` statement needs a function to aggregate those values, such as `COUNT`, `SUM`, or `MEAN`.

```
sqlStatement <- "SELECT patientID, COUNT(Actual_date) as numVisits FROM patient_encounter
                 GROUP BY patientID"

queryResult <- dbGetQuery(con, sqlStatement)
```

```
queryResult[1:20,]
```

```
##    patientID numVisits
## 1          1        18
## 2          2        64
## 3          3        29
## 4          4         6
## 5          5         5
## 6          6         5
## 7          7        16
## 8          8        32
## 9          9        21
## 10        10        34
## 11        11        18
## 12        12         4
## 13        13        30
## 14        14         4
## 15        15        41
## 16        16        25
## 17        17        46
## 18        18         6
## 19        19        15
## 20        20         1
```

Note that `GROUP BY` can be combined with a subquery, which can be very useful (this is a hint).

Finding a Set of Values in a Table

A fast way of finding entries in a table that satisfy a set of values is to do a join on the table with a temporary table that has the values that you're interested in.

```
## [1] TRUE
```

```
#let's define a new table in our database with the outcome values
#of interest
outcomesOfInterest <- c("SNF", "Rehab")
outcomeTable <- data.frame(outcomesOfInterest)
#show the data frame
outcomeTable
```

```
##   outcomesOfInterest
## 1                SNF
## 2             Rehab
```

```
#add the table to the database (note we don't disconnect, because we don't want
#to save this table)
dbWriteTable(con, name="outcomeTable", value=outcomeTable)
```

```
## [1] TRUE
```

```
sqlStatement <- "SELECT peh.* from patient_encounter_hosp as peh,
              outcomeTable as oc
              WHERE peh.outcome = oc.outcomesOfInterest"

queryResult <- dbGetQuery(con, sqlStatement)
queryResult[1:20,]
```

```
##    patientid Event_ID encounter_type outcome Admit_date Discharge_date
## 1          1      108             22     SNF 2014-01-01     2014-01-08
## 2          1      109             22     SNF 2014-01-13     2014-01-20
## 3          3       71             22     SNF 2014-01-01     2014-01-07
## 4          6       98             22     SNF 2014-01-01     2014-01-08
## 5          6       99             22     SNF 2014-01-19     2014-02-01
## 6          7      893             22   Rehab 2014-01-01     2014-01-16
## 7          8     2556             22     SNF 2014-01-01     2014-01-06
## 8          9      649             22     SNF 2014-01-01     2014-01-08
## 9         10      979             22     SNF 2014-01-01     2014-01-04
## 10        11     1815             22     SNF 2014-01-01     2014-01-02
## 11        13     1663             22     SNF 2014-01-01     2014-01-04
## 12        14      999             22     SNF 2014-01-01     2014-01-02
## 13        16       11             22   Rehab 2014-01-01     2014-01-08
## 14        17      587             22     SNF 2014-01-01     2014-01-03
## 15        18     1868             22     SNF 2014-01-01     2014-01-03
## 16        21     1180             22     SNF 2014-01-01     2014-02-21
## 17        21     1181             22     SNF 2014-03-04     2014-03-19
## 18        23     1920             22     SNF 2014-01-01     2014-01-07
## 19        30      512             22     SNF 2014-01-01     2014-01-04
## 20        33      732             22     SNF 2014-01-01     2014-01-05
##      Admit_source
## 1  Emergency Room
## 2        Transfer
## 3        Transfer
## 4  Emergency Room
## 5        Transfer
## 6  Emergency Room
## 7        Transfer
## 8  Emergency Room
## 9  Emergency Room
## 10       Transfer
## 11       Transfer
## 12 Emergency Room
## 13            SNF
## 14       Transfer
## 15 Emergency Room
## 16       Transfer
## 17       Transfer
## 18       Transfer
## 19 Emergency Room
## 20 Emergency Room
```

Finding values using LIKE

We can do string matching using the `LIKE` statement. For example, we may want to return all values in a column that have `patient` in them. We can use the `%` as a wildcard to match multiple characters. For example, using `LIKE "patient%"` will match `patientID`, `patientSource`, and `patientName`, but not `patSource`.

Here we are selecting everything from the `patient` table that have postal codes that match `970%`.

```
sqlStatement <- "SELECT * from patient where postalcode LIKE '970%'"
queryResult <- dbGetQuery(con, sqlStatement)
queryResult[1:20,]
```

```
##    patientid GENDER First_name Last_name DOB age status PCP CM_ID
## 1       1214 Female      Gilly       Doe  NA  NA      4  14    NA
## 2       1079   Male       Gill       Doe  NA  NA      4  14    NA
## 3        154   Male       Gill       Doe  NA  NA      4  14    NA
## 4         96 Female      Gilly       Doe  NA  NA      4  14    NA
## 5       1426   Male       Gill       Doe  NA  NA      4  14    NA
## 6        369 Female      Gilly       Doe  NA  NA      4  14    NA
## 7        625 Female      Gilly       Doe  NA  NA      4  14    NA
## 8        728 Female      Gilly       Doe  NA  NA      4  14    NA
## 9        428 Female      Gilly       Doe  NA  NA      4  14    NA
## 10       483 Female      Gilly       Doe  NA  NA      4  14    NA
## 11       179 Female      Gilly       Doe  NA  NA      4  14    NA
## 12      1430   Male       Gill       Doe  NA  NA      4  14    NA
## 13       971 Female      Gilly       Doe  NA  NA      4  14    NA
## 14       200 Female      Gilly       Doe  NA  NA      4  14    NA
## 15      1584 Female      Gilly       Doe  NA  NA      4  14    NA
## 16      1077 Female      Gilly       Doe  NA  NA      4  14    NA
## 17      1485 Female      Gilly       Doe  NA  NA      4  14    NA
## 18       538   Male       Gill       Doe  NA  NA      4  14    NA
## 19      1028 Female      Gilly       Doe  NA  NA      4  14    NA
## 20       163   Male       Gill       Doe  NA  NA      4  14    NA
##    referraldate Insurance_ID race postalcode riskCat riskscore deleted
## 1            NA            3    1      97035    High         4       0
## 2            NA            3    3      97035    High         3       0
## 3            NA            3    3      97035    High         3       0
## 4            NA            3    4      97035    High         3       0
## 5            NA            3    3      97035    High         3       0
## 6            NA            3    1      97035  Normal         2       0
## 7            NA            3    1      97035    High         3       0
## 8            NA            3    1      97035    High         4       0
## 9            NA            3    1      97035    High         4       0
## 10           NA            3    1      97035    High         3       0
## 11           NA            3    1      97035    High         3       0
## 12           NA            3    1      97035    High         4       0
## 13           NA            3    1      97035    High         3       0
## 14           NA            3    1      97035    High         5       0
## 15           NA            3    1      97035    High         4       0
## 16           NA            3    1      97035    High         3       0
## 17           NA            3    1      97035  Normal         2       0
## 18           NA            3    1      97035    High         3       0
## 19           NA            3    1      97035  Normal         2       0
## 20           NA            3    1      97035    High         6       0
```

```
##    date_deleted facility source
## 1            NA      500   deid
## 2            NA      500   deid
## 3            NA      500   deid
## 4            NA      500   deid
## 5            NA      500   deid
## 6            NA      500   deid
## 7            NA      500   deid
## 8            NA      500   deid
## 9            NA      500   deid
## 10           NA      500   deid
## 11           NA      500   deid
## 12           NA      500   deid
## 13           NA      500   deid
## 14           NA      500   deid
## 15           NA      500   deid
## 16           NA      500   deid
## 17           NA      500   deid
## 18           NA      500   deid
## 19           NA      500   deid
## 20           NA      500   deid
```

We can chain multiple `LIKE` clauses using boolean operators such as `AND`, `OR`, and `NOT`. Unfortunately, SQLite does not understand the `ANY` clause, which would simplify our searching.

```
sqlStatement <- "SELECT * from patient where postalcode LIKE '970%' AND
                 riskCat NOT LIKE 'Norm%'"
queryResult <- dbGetQuery(con, sqlStatement)
queryResult[1:20,]
```

```
##    patientid GENDER First_name Last_name DOB age status PCP CM_ID
## 1       1214 Female      Gilly       Doe  NA  NA      4  14    NA
## 2       1079   Male       Gill       Doe  NA  NA      4  14    NA
## 3        154   Male       Gill       Doe  NA  NA      4  14    NA
## 4         96 Female      Gilly       Doe  NA  NA      4  14    NA
## 5       1426   Male       Gill       Doe  NA  NA      4  14    NA
## 6        625 Female      Gilly       Doe  NA  NA      4  14    NA
## 7        728 Female      Gilly       Doe  NA  NA      4  14    NA
## 8        428 Female      Gilly       Doe  NA  NA      4  14    NA
## 9        483 Female      Gilly       Doe  NA  NA      4  14    NA
## 10       179 Female      Gilly       Doe  NA  NA      4  14    NA
## 11      1430   Male       Gill       Doe  NA  NA      4  14    NA
## 12       971 Female      Gilly       Doe  NA  NA      4  14    NA
## 13       200 Female      Gilly       Doe  NA  NA      4  14    NA
## 14      1584 Female      Gilly       Doe  NA  NA      4  14    NA
## 15      1077 Female      Gilly       Doe  NA  NA      4  14    NA
## 16       538   Male       Gill       Doe  NA  NA      4  14    NA
## 17       163   Male       Gill       Doe  NA  NA      4  14    NA
## 18       831 Female      Gilly       Doe  NA  NA      4  14    NA
## 19      1113 Female      Gilly       Doe  NA  NA      4  14    NA
## 20       197   Male       Gill       Doe  NA  NA      4  14    NA
##    referraldate Insurance_ID race postalcode riskCat riskscore deleted
## 1            NA            3    1      97035    High         4       0
```

```
## 2            NA            3    3      97035    High         3       0
## 3            NA            3    3      97035    High         3       0
## 4            NA            3    4      97035    High         3       0
## 5            NA            3    3      97035    High         3       0
## 6            NA            3    1      97035    High         3       0
## 7            NA            3    1      97035    High         4       0
## 8            NA            3    1      97035    High         4       0
## 9            NA            3    1      97035    High         3       0
## 10           NA            3    1      97035    High         3       0
## 11           NA            3    1      97035    High         4       0
## 12           NA            3    1      97035    High         3       0
## 13           NA            3    1      97035    High         5       0
## 14           NA            3    1      97035    High         4       0
## 15           NA            3    1      97035    High         3       0
## 16           NA            3    1      97035    High         3       0
## 17           NA            3    1      97035    High         6       0
## 18           NA            3    1      97035    High         3       0
## 19           NA            3    1      97035    High         3       0
## 20           NA            3    3      97035    High         3       0
##    date_deleted facility source
## 1            NA      500   deid
## 2            NA      500   deid
## 3            NA      500   deid
## 4            NA      500   deid
## 5            NA      500   deid
## 6            NA      500   deid
## 7            NA      500   deid
## 8            NA      500   deid
## 9            NA      500   deid
## 10           NA      500   deid
## 11           NA      500   deid
## 12           NA      500   deid
## 13           NA      500   deid
## 14           NA      500   deid
## 15           NA      500   deid
## 16           NA      500   deid
## 17           NA      500   deid
## 18           NA      500   deid
## 19           NA      500   deid
## 20           NA      500   deid
```