QBS181 Midterm

Mark Taylor

Setup

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

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

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(odbc)
library(sqldf)
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite
```

```
library(stringr)
library(SASxport)

# Connect to database
con <- DBI::dbConnect(odbc::odbc(),
                      Driver   = "/usr/local/lib/libmsodbcsql.17.dylib",
```

```
                      Server   = "qbs181-db.dartmouth.edu",
                      Database = "qbs181",
                      UID      = "mtaylor",
                      PWD      = "mtaylor@qbs181",
                      Port     = 40062)
```

1

```
DIQ <- read.xport("DIQ_I.XPT")

library(DBI)

#dbWriteTable(con, SQL("mtaylor.DIQ"), DIQ)
```

a) List the data-related issues you see in this data set

1. Many columns are coded, when really they should be using a descriptor value instead
2. Some columns containing meaningful numerical values also have codes intermixed into them, which could lead to confusion when analyzing if codes were to be interpreted as numerical values
3. Some columns that are coded would be much better suited to contain boolean values instead. Boolean values are easier to work with and more readable, and in these instances edge cases that cannot be described with a Boolean are neglible or can be expressed as NULL
4. Some columns contain their units in another column, most notably lengths of time that need corresponding units to make any sense. In one column there might be the value 1, but you need the corresponding column to know if that means 1 second, 1 minute, 1 hour, etc.
5. The column names are all encoded and not possible for a human to intuitively know what information they contain
6. There are many rows which are missing a lot of information, seeming to be filled mostly with NULL values

b) How will you address each data-related issue?

1. Because we know the descriptor associated with each code, in instances where we should replace a code with a descriptor we check each code and replace with the descriptor it should contain. For the most part, these are categories that should be readily understood and have semantic meaning
2. In an instance where codes occur in a column that should just contain values, we can check for the code and replace it with a meaningful value (or a NULL in many cases, as these often represent a lack of information). These codes are never in the range of possible values, so they can either be set to meaningful values (666 representing an infant being replaced with zero) or NULL (such as instances where a response is unknown or unanswered)
3. These instances are easy to interpret, usually we just need to simply replace 1s with TRUE and 2s with FALSE. We treat all other possibilities as NULLs in these instances, which will be justified in the next part.

4. For these instances, we need to first parse out the unit from the table containing the unit (as it typically is encoded). Then we can look through the table with the actual values and apply the appropriate units

5. I simply made new columns for everything as I went along, using names containing semantic information

6. For the most part I avoided imputing data for the sake of filling vacant values. In some instances there is a meaningful alternative to replace NULL with, such as FALSE, but for the most part it wasn't possible to come up with meaning values without risking confounding the data

1. This approach seemed best for a few reasons. For one, a human should be able to look at a table and understand it. Without descriptors, that is impossible. These instances usually contain categorical data, so it makes sense to separate rows into discrete groups and provide a description for them

2. Removing these codes from otherwise coherent information was an incredibly important step. While in all cases the codes were outside the range of possible values if someone working with the data were to try and take a mean for example, codes throughout the data could drastically throw it off. There were only a few situations where codes could be assigned comparable values, but for the most part we discarded these values to maintain the integrity of the data and its usefulness for analysis

3. Converting to a boolean makes the column much more intuitive to read and much easier to work with. In each column where this is the case, the column is asking a yes or no question. Yes corresponds to TRUE and No corresponds to FALSE. The notable edge cases here is for when subjects refuse to answer or do not know an answer. While in some contexts this might be a response worth analyzing, these two sub categories generally made up a very small amount of the data. As the goal was essentially to make columns as coherent as possible, if they are asking a yes or no question and the participant refuses to answer or doesn't know the answer, that would be considered a NULL value; trying to estimate what they might have said could lead to confounding.

4. This approach was essentially necessary to make certain columns meaningful. For example, time values were stored in one column and units were stored in the other, so it was necessary to unpack the units and apply them to the time values, otherwise they wouldn't make any sense in relation to each other.

5. The justification here is pretty straight-forward. Tables should be as human readable as they can if its a possibility. The variables names they had before had no obvious interpretation, so replacing them with human friendly names made the data easier to work with.

6. While I considered performing mean imputation in a lot of scenarios, for the most part I avoided generating any information that wasn't present in the data set. In many instances, there was a significant portion of the data that was missing. While generating similar information is tantalizing, in most cases there was so much missing data that confounding would have been extremely likely. I figured someone analyzing the data set would be much better off using the non-null values to avoid confounding.

2

Code to descriptor

```
SELECT DIQ010, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ010
ORDER BY DIQ010
```

Table 1: DIQ010 Count Check using SQL

DIQ010	CodeCount
1	856
2	8568
3	147
9	4

Code or Value	Value Description	Count
1	Yes	856
2	No	8568
3	Borderline	147
7	Refused	0
9	Don't know	4
.	Missing	0

Figure 1: DIQ010 Count Check from website

```
# Convert coded value to descriptor and use a more meaningful column name
DIQ$DiabetesDiagnosis <-
  ifelse(DIQ$DIQ010 == 1, "Yes",
  ifelse(DIQ$DIQ010 == 2, "No",
  ifelse(DIQ$DIQ010 == 3, "Borderline",
  ifelse(DIQ$DIQ010 == 4, "Don't know", NA
  ))))

sample(DIQ$DiabetesDiagnosis, 10)
```

```
##  [1] "No"  "No"  "No"  "Yes" "No"  "No"  "No"  "No"  "No"  "No"
```

```
SELECT DIQ230, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ230
ORDER BY DIQ230
```

Table 2: DIQ230 Count Check using SQL

DIQ230	CodeCount
NA	8722
1	276
2	64

DIQ230	CodeCount
3	86
4	85
5	330
9	12

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$LengthSinceSeenSpecialist <-
  ifelse(DIQ$DIQ230 == 1, "1 year ago or less",
  ifelse(DIQ$DIQ230 == 2, "More than 1 year ago but no more than 2 years ago",
  ifelse(DIQ$DIQ230 == 3, "More than 2 years ago but no more than 5 years ago",
  ifelse(DIQ$DIQ230 == 4, "More than 5 years ago",
  ifelse(DIQ$DIQ230 == 5, "Never", NA
  )))))

sample(DIQ$LengthSinceSeenSpecialist, 10)
```

```
##  [1] NA      NA      NA      NA      NA      "Never" NA      NA      NA
## [10] NA
```

```r
SELECT DIQ230, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ230
ORDER BY DIQ230
```

Table 3: DIQ230 Count Check using SQL

DIQ230	CodeCount
NA	8722
1	276
2	64
3	86
4	85
5	330
9	12

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$LengthSinceSpecialistSeen <-
  ifelse(DIQ$DIQ230 == 1, "1 year ago or less",
  ifelse(DIQ$DIQ230 == 2, "More than 1 year ago but no more than 2 years ago",
  ifelse(DIQ$DIQ230 == 3, "More than 2 years ago but no more than 5 years ago",
  ifelse(DIQ$DIQ230 == 4, "More than 5 years ago",
  ifelse(DIQ$DIQ230 == 5, "Never", NA
  )))))

sample(DIQ$LengthSinceSpecialistSeen, 10)
```

```
##  [1] NA
```

Code or Value	Value Description	Count
1	1 year ago or less	276
2	More than 1 year ago but no more than 2 years ago	64
3	More than 2 years ago but no more than 5 years ago	86
4	More than 5 years ago	85
5	Never	330
7	Refused	0
9	Don't know	12
.	Missing	8722

Figure 2: DIQ230 Count Check from website

Code or Value	Value Description	Count
1	1 year ago or less	276
2	More than 1 year ago but no more than 2 years ago	64
3	More than 2 years ago but no more than 5 years ago	86
4	More than 5 years ago	85
5	Never	330
7	Refused	0
9	Don't know	12
.	Missing	8722

Figure 3: DIQ230 Count Check from website

```
##  [2] NA
##  [3] NA
##  [4] "More than 1 year ago but no more than 2 years ago"
##  [5] NA
##  [6] "More than 5 years ago"
##  [7] NA
##  [8] NA
##  [9] NA
## [10] NA
```

```
SELECT DIQ291, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ291
ORDER BY DIQ291
```

Table 4: DIQ291 Count Check using SQL

DIQ291	CodeCount
NA	8934
1	213
2	182
3	32
4	5
5	9
6	88
77	2
99	110

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$DoctorA1CGoal <-
  ifelse(DIQ$DIQ291 == 1, "Less than 6",
  ifelse(DIQ$DIQ291 == 2, "Less than 7",
  ifelse(DIQ$DIQ291 == 3, "Less than 8",
  ifelse(DIQ$DIQ291 == 4, "Less than 9",
  ifelse(DIQ$DIQ291 == 5, "Less than 10",
  ifelse(DIQ$DIQ291 == 6, "Provider did not specify goal",
  ifelse(DIQ$DIQ291 == 77, "Refused",
  ifelse(DIQ$DIQ291 == 99, "Don't know", NA
  ))))))))

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DoctorA1CGoal[!is.na(DIQ$DoctorA1CGoal)], 10)
```

```
##  [1] "Less than 6"  "Less than 6"  "Less than 7"  "Don't know"   "Less than 7"
##  [6] "Less than 8"  "Less than 6"  "Less than 6"  "Less than 10" "Less than 6"
```

```
SELECT DIQ360, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ360
ORDER BY DIQ360
```

Code or Value	Value Description	Count
1	Less than 6	213
2	Less than 7	182
3	Less than 8	32
4	Less than 9	5
5	Less than 10	9
6	Provider did not specify goal	88
77	Refused	2
99	Don't know	110
.	Missing	8934

Figure 4: DIQ291 Count Check from website

Table 5: DIQ360 Count Check using SQL

DIQ360	CodeCount
NA	8729
1	82
2	440
3	129
4	116
5	64
9	15

Code or Value	Value Description	Count
1	Less than 1 month	82
2	1-12 months	440
3	13-24 months	129
4	Greater than 2 years	116
5	Never	64
7	Refused	0
9	Don't know	15
.	Missing	8729

Figure 5: DIQ360 Count Check from website

```r
sample(DIQ$LastPupilDilationExam, 10)
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

Code mixed in with values

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID040 >= 2 AND DID040 <= 78 THEN '2-78'
               ELSE DID040
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 6: DID040 Count Check using SQL

range	CodeCount
NA	8722
2-78	833
666	1
80	7
999	12

Code or Value	Value Description	Count
2 to 78	Range of Values	833
80	80 years or older	7
666	Less than 1 year	1
777	Refused	0
999	Don't know	12
.	Missing	8722

Figure 6: DID040 Count Check from website

```r
# Values not related to age should be replaced
DIQ$DiagnosisAge <-
  ifelse(DIQ$DID040 == 666, 0,
  ifelse(DIQ$DID040 == 999, NA,
  ifelse((DIQ$DID040 >= 2 & DIQ$DID040 <= 78), DIQ$DID040, NA)
  ))
```

```r
# Most samples are NA, so we exclude them when sampling
sample(DIQ$DiagnosisAge[!is.na(DIQ$DiagnosisAge)], 10)
```

```
##  [1] 52 40 40 65 55 64 33 49 52 44
```

```sql
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID250 >= 1 AND DID250 <= 60 THEN '1-60'
               WHEN DID250 like '%9999%' THEN NULL
               ELSE DID250
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 7: DID250 Count Check using SQL

range	CodeCount
NA	8935
0	13
1-60	627

Code or Value	Value Description	Count
1 to 60	Range of Values	627
0	None	13
7777	Refused	0
9999	Don't know	3
.	Missing	8932

Figure 7: DID250 Count Check from website

```r
# Values not related to age should be replaced
DIQ$DoctorVisitsLastYear <-
  ifelse(DIQ$DID250 == '9999', NA, DIQ$DID250)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DoctorVisitsLastYear[!is.na(DIQ$DoctorVisitsLastYear)], 10)
```

```
##  [1]  3  2  3 10  7  3  3  4  3  2
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN CAST(DIQ280 AS DECIMAL(22,8)) >= 2.0 AND CAST(DIQ280 AS DECIMAL(22,8)) <= 18.5 THEN
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 8: DIQ280 Count Check using SQL

range	CodeCount
NA	9171
2-18.5	404

Code or Value	Value Description	Count
2 to 18.5	Range of Values	404
777	Refused	2
999	Don't know	235
.	Missing	8934

Figure 8: DIQ280 Count Check from website

```
# Values not related to age should be replaced
DIQ$LastA1CLevel <-
  ifelse(DIQ$DIQ280 == '999' | DIQ$DIQ280 == '777', NA, DIQ$DIQ280)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$LastA1CLevel[!is.na(DIQ$LastA1CLevel)], 10)
```

```
##  [1] 7.5 5.6 6.0 9.3 7.0 6.7 5.1 7.0 8.0 7.1
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DIQ300S >= 80 AND DIQ300S <= 201 THEN '80-201'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 9: DIQ300S Count Check using SQL

range	CodeCount
NA	9036
80-201	539

Code or Value	Value Description	Count
80 to 201	Range of Values	539
7777	Refused	1
9999	Don't know	305
.	Missing	8730

Figure 9: DIQ300S Count Check from website

```
# Values not related to age should be replaced
DIQ$MostRecentSBP <-
  ifelse((DIQ$DIQ300S >= 80 & DIQ$DIQ300S <= 201), DIQ$DIQ300S, NA
  )

# Most samples are NA, so we exclude them when sampling
sample(DIQ$MostRecentSBP[!is.na(DIQ$MostRecentSBP)], 10)
```

```
##  [1] 146 104 132 110 122 118 123 120 120 134
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DIQ300D >= 17 AND DIQ300D <= 251 THEN '17-251'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 10: DIQ300D Count Check using SQL

range	CodeCount
NA	9060
17-251	515

```
# Values not related to age should be replaced
DIQ$MostRecentDBP <-
  ifelse((DIQ$DIQ300D >= 80 & DIQ$DIQ300D <= 201), DIQ$DIQ300D, NA
```

Code or Value	Value Description	Count
17 to 251	Range of Values	515
7777	Refused	2
9999	Don't know	328
.	Missing	8730

Figure 10: DIQ300D Count Check from website

```
  )

# Most samples are NA, so we exclude them when sampling
sample(DIQ$MostRecentDBP[!is.na(DIQ$MostRecentDBP)], 10)
```

```
##  [1]  80  98  84  85  96  85  90  88 107  90
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID310S >= 80 AND DID310S <= 175 THEN '80-175'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 11: DIQ300S Count Check using SQL

range	CodeCount
NA	9239
80-175	336

Code or Value	Value Description	Count
80 to 175	Range of Values	336
6666	Provider did not specify goal	308
7777	Refused	2
9999	Don't know	200
.	Missing	8729

Figure 11: DID310S Count Check from website


```
# Values not related to age should be replaced
DIQ$DoctorRecommendedSBP <-
  ifelse((DIQ$DID310S >= 80 & DIQ$DID310S <= 175), DIQ$DIQ300S, NA
  )
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID310D >= 18 AND DID310D <= 140 THEN '18-140'
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 12: DID310D Count Check using SQL

range	CodeCount
NA	9260
18-140	315

Code or Value	Value Description	Count
18 to 140	Range of Values	315
6666	Provider did not specify goal	308
7777	Refused	2
9999	Don't know	221
.	Missing	8729

Figure 12: DID310D Count Check from website

```
# Values not related to age should be replaced
DIQ$DoctorRecommendedDBP <-
  ifelse((DIQ$DID310D >= 80 & DIQ$DID310D <= 201), DIQ$DID310D, NA
  )

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DoctorRecommendedDBP[!is.na(DIQ$DoctorRecommendedDBP)], 10)
```

```
##  [1] 80 80 80 80 80 85 80 80 80 80
```

Code should be replaced with booleans

```
SELECT DIQ160, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ160
ORDER BY DIQ160
```

Table 13: DIQ160 Count Check using SQL

DIQ160	CodeCount
NA	3530
1	513
2	5521
9	11

Code or Value	Value Description	Count
1	Yes	513
2	No	5521
7	Refused	0
9	Don't know	11
.	Missing	3530

Figure 13: DIQ160 Count Check from website

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$EverDiagnosedWithPrediabetes <-
  ifelse(DIQ$DIQ160 == 1, TRUE,
  ifelse(DIQ$DIQ160 == 2, FALSE, NA
  ))

sample(DIQ$EverDiagnosedWithPrediabetes[!is.na(DIQ$EverDiagnosedWithPrediabetes)], 10)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
SELECT DIQ160, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ160
ORDER BY DIQ160
```

Table 14: DIQ160 Count Check using SQL

DIQ160	CodeCount
NA	3530
1	513
2	5521
9	11

Code or Value	Value Description	Count
1	Yes	513
2	No	5521
7	Refused	0
9	Don't know	11
.	Missing	3530

Figure 14: DIQ160 Count Check from website

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$EverDiagnosedWithPrediabetes <-
  ifelse(DIQ$DIQ160 == 1, TRUE,
  ifelse(DIQ$DIQ160 == 2, FALSE, NA
  ))

sample(DIQ$EverDiagnosedWithPrediabetes[!is.na(DIQ$EverDiagnosedWithPrediabetes)], 10)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
SELECT DIQ170, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ170
ORDER BY DIQ170
```

Table 15: DIQ170 Count Check using SQL

DIQ170	CodeCount
NA	3389
1	899
2	5268
9	19

Code or Value	Value Description	Count
1	Yes	899
2	No	5268
7	Refused	0
9	Don't know	19
.	Missing	3389

Figure 15: DIQ170 Count Check from website

```
# Convert code to descriptive intervals and use a more meaningful column name
```

```sql
SELECT DIQ172, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ172
ORDER BY DIQ172
```

Table 16: DIQ172 Count Check using SQL

DIQ172	CodeCount
NA	3389
1	1588
2	4510
7	1
9	87

Code or Value	Value Description	Count
1	Yes	1588
2	No	4510
7	Refused	1
9	Don't know	87
.	Missing	3389

Figure 16: DIQ172 Count Check from website

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$FeelCouldHaveDiabetesRisk <-
  ifelse(DIQ$DIQ172 == 1, TRUE,
  ifelse(DIQ$DIQ172 == 2, FALSE, NA
  ))

sample(DIQ$FeelCouldHaveDiabetesRisk[!is.na(DIQ$FeelCouldHaveDiabetesRisk)], 10)
```

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

```sql
SELECT DIQ180, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ180
ORDER BY DIQ180
```

Table 17: DIQ180 Count Check using SQL

DIQ180	CodeCount
NA	3389
1	2836
2	3167
7	2
9	181

Code or Value	Value Description	Count
1	Yes	2836
2	No	3167
7	Refused	2
9	Don't know	181
.	Missing	3389

Figure 17: DIQ180 Count Check from website

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$HadBloodTestInLast3Years <-
  ifelse(DIQ$DIQ180 == 1, TRUE,
  ifelse(DIQ$DIQ180 == 2, FALSE, NA
  ))

sample(DIQ$HadBloodTestInLast3Years[!is.na(DIQ$HadBloodTestInLast3Years)], 10)
```

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

```
SELECT DIQ050, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ050
ORDER BY DIQ050
```

Table 18: DIQ050 Count Check using SQL

DIQ050	CodeCount
1	256
2	9316
7	1
9	2

Code or Value	Value Description	Count
1	Yes	256
2	No	9316
7	Refused	1
9	Don't know	2
.	Missing	0

Figure 18: DIQ050 Count Check from website

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$TakingInsulinNow <-
  ifelse(DIQ$DIQ050 == 1, TRUE,
  ifelse(DIQ$DIQ050 == 2, FALSE, NA
  ))

sample(DIQ$TakingInsulinNow[!is.na(DIQ$TakingInsulinNow)], 10)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
SELECT DIQ070, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ070
ORDER BY DIQ070
```

Table 19: DIQ070 Count Check using SQL

DIQ070	CodeCount
NA	8060
1	643
2	870
7	1
9	1

Code or Value	Value Description	Count
1	Yes	643
2	No	870
7	Refused	1
9	Don't know	1
.	Missing	8060

Figure 19: DIQ070 Count Check from website

```sql
SELECT DIQ240, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ240
ORDER BY DIQ240
```

Table 20: DIQ240 Count Check using SQL

DIQ240	CodeCount
NA	8722
1	643
2	210

Code or Value	Value Description	Count
1	Yes	643
2	No	210
7	Refused	0
9	Don't know	0
.	Missing	8722

Figure 20: DIQ240 Count Check from website

```r
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$HasDiabetesDoctor <-
  ifelse(DIQ$DIQ240 == 1, TRUE,
  ifelse(DIQ$DIQ240 == 2, FALSE, NA
  ))

sample(DIQ$HasDiabetesDoctor[!is.na(DIQ$HasDiabetesDoctor)], 10)
```

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

```sql
SELECT DIQ275, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ275
ORDER BY DIQ275
```

Table 21: DIQ275 Count Check using SQL

DIQ275	CodeCount
NA	8722
1	641
2	156
9	56

Code or Value	Value Description	Count
1	Yes	641
2	No	156
7	Refused	0
9	Don't know	56
.	Missing	8722

Figure 21: DIQ275 Count Check from website

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$DrCheckedA1CInPastYear <-
  ifelse(DIQ$DIQ275 == 1, TRUE,
  ifelse(DIQ$DIQ275 == 2, FALSE, NA
  ))

sample(DIQ$DrCheckedA1CInPastYear[!is.na(DIQ$DrCheckedA1CInPastYear)], 10)
```

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

Units needed from one column

```
SELECT DIQ260U, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ260U
ORDER BY DIQ260U
```

Table 22: DIQ260U Count Check using SQL

DIQ260U	CodeCount
NA	8914
1	412
2	150
3	65
4	34

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$BloodGlucoseCheckFrequencyUnits <-
  ifelse(DIQ$DIQ260U == 1, "daily",
  ifelse(DIQ$DIQ260U == 2, "weekly",
  ifelse(DIQ$DIQ260U == 3, "monthly",
  ifelse(DIQ$DIQ260U == 4, "yearly", NA
```

Code or Value	Value Description	Count
1	Per day	412
2	Per week	150
3	Per month	65
4	Per year	34
.	Missing	8914

Figure 22: DIQ260U Count Check from website

```
  ))))

# Most samples are NA, so we exclude them when sampling
sample(DIQ$BloodGlucoseCheckFrequencyUnits[!is.na(DIQ$BloodGlucoseCheckFrequencyUnits)], 10)
```

```
##  [1] "weekly"  "monthly" "weekly"  "weekly"  "weekly"  "daily"   "daily"
##  [8] "daily"   "daily"   "daily"
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID260 >= 1 AND DID260 <= 15 THEN '1-15'
               WHEN DID260 = 0 THEN 'Never'
               ELSE DID260
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 23: DID260 Count Check using SQL

range	CodeCount
NA	8726
1-15	661
Never	188

Code or Value	Value Description	Count
1 to 15	Range of Values	661
0	Never	188
777	Refused	0
999	Don't know	0
.	Missing	8726

Figure 23: DID260 Count Check from website

```
          )))))
DIQ$BloodGlucoseCheckFrequency <- days(DIQ$BloodGlucoseCheckFrequency)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$BloodGlucoseCheckFrequency[!is.na(DIQ$BloodGlucoseCheckFrequency)], 10)
```

```
##  [1] "2d 0H 0M 0S"  "1d 0H 0M 0S"  "1d 0H 0M 0S"  "1d 0H 0M 0S"  "31d 0H 0M 0S"
##  [6] "1d 0H 0M 0S"  "1d 0H 0M 0S"  "4d 0H 0M 0S"  "1d 0H 0M 0S"  "21d 0H 0M 0S"
```

```
SELECT DIQ060U, COUNT(*) as CodeCount
FROM qbs181.mtaylor.DIQ
GROUP BY DIQ060U
ORDER BY DIQ060U
```

Table 24: DIQ060U Count Check using SQL

DIQ060U	CodeCount
NA	9326
1	30
2	219

Code or Value	Value Description	Count
1	Months	30
2	Years	219
.	Missing	9326

Figure 24: DIQ060U Count Check from website

```
# Convert code to descriptive intervals and use a more meaningful column name
DIQ$DurationTakingInsulinUnits <-
  ifelse(DIQ$DIQ260U == 1, "monthly",
  ifelse(DIQ$DIQ260U == 2, "yearly", NA
  ))

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DurationTakingInsulinUnits[!is.na(DIQ$DurationTakingInsulinUnits)], 10)
```

```
##  [1] "monthly" "monthly" "monthly" "monthly" "yearly"  "yearly"  "monthly"
##  [8] "monthly" "monthly" "monthly"
```

```
SELECT D.range, COUNT(*) as CodeCount
FROM (SELECT CASE
               WHEN DID060 >= 1 AND DID060 <= 55 THEN '1-55'
               WHEN DID060 = 666 THEN 'Less than 1 month'
```

```
               ELSE NULL
             END as range
      FROM qbs181.mtaylor.DIQ) D
GROUP BY D.range
ORDER BY D.range
```

Table 25: DID060 Count Check using SQL

range	CodeCount
NA	9321
1-55	249
Less than 1 month	5

Code or Value	Value Description	Count
1 to 55	Range of Values	249
666	Less than 1 month	5
777	Refused	0
999	Don't know	2
.	Missing	9319

Figure 25: DID060 Count Check from website

```
# Values not related to age should be replaced

DIQ <- DIQ %>%
  mutate(DurationTakingInsulin =
           ifelse((DurationTakingInsulinUnits == "monthly"),DID060,
           ifelse((DurationTakingInsulinUnits == "yearly"),DID060*12, NA
           )))
DIQ$DurationTakingInsulin <- months(DIQ$DurationTakingInsulin)

# Most samples are NA, so we exclude them when sampling
sample(DIQ$DurationTakingInsulin[!is.na(DIQ$DurationTakingInsulin)], 10)
```

```
##  [1] "5m 0d 0H 0M 0S"  "1m 0d 0H 0M 0S"  "5m 0d 0H 0M 0S"  "10m 0d 0H 0M 0S"
##  [5] "1m 0d 0H 0M 0S"  "5m 0d 0H 0M 0S"  "6m 0d 0H 0M 0S"  "16m 0d 0H 0M 0S"
##  [9] "2m 0d 0H 0M 0S"  "10m 0d 0H 0M 0S"
```