Assignment 4

Sungjoo Cho

2023-11-07

GitHub Link

https://github.com/scho7/SURV727-Assignment4.git

Packages

```r
library(bigrquery)
library(DBI)
library(dbplyr)
library(dplyr)
```

Project

```r
# create project object
project <- "surv727-project"

# connect to a public database (Chicago crime database)
con <- dbConnect(
bigrquery::bigquery(),
project = "bigquery-public-data",
dataset = "chicago_crime",
billing = project
)

con
```

```
## <BigQueryConnection>
##   Dataset: bigquery-public-data.chicago_crime
##   Billing: surv727-project
```

```r
# check available tables
dbListTables(con)
```

```
## ! Using an auto-discovered, cached token.
```

```
##   To suppress this message, modify your code or options to clearly consent to
##   the use of a cached token.
##   See gargle's "Non-interactive auth" vignette for more details:

##   <https://gargle.r-lib.org/articles/non-interactive-auth.html>

## i The bigrquery package is using a cached token for 'sungjoocho7@gmail.com'.

## [1] "crime"
```

Write a first query that counts the number of rows of the 'crime' table in the year 2016. Use code chunks with {sql connection = con} in order to write SQL code within the document.

The number of rows of the 'crime' table in the year 2016 is 269841.

```
SELECT count(*)
FROM crime
WHERE year = 2016
LIMIT 10;
```

Table 1: 1 records

f0_
269841

Next, count the number of arrests grouped by primary_type in 2016. Note that is a somewhat similar task as above, with some adjustments on which rows should be considered. Sort the results, i.e. list the number of arrests in a descending order.

When sorting the count of arrests grouped by primary_type in descending order, the primary_type "NARCOTICS" had the highest number of arrests, totaling 13,327. It was followed by "BATTERY" with 10,332 arrests, and "THEFT" with 6,522 arrests.

```
# way 1
SELECT primary_type, count(*)
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY primary_type
ORDER BY count(*) DESC;
```

Table 2: Displaying records 1 - 10

primary_type	f0_
NARCOTICS	13327
BATTERY	10332
THEFT	6522
CRIMINAL TRESPASS	3724
ASSAULT	3492
OTHER OFFENSE	3415
WEAPONS VIOLATION	2511
CRIMINAL DAMAGE	1669

primary_type	f0_
PUBLIC PEACE VIOLATION	1116
MOTOR VEHICLE THEFT	1097

```
# way 2
arr_prm_type_2016 <- "SELECT primary_type, count(*)
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY primary_type
ORDER BY COUNT(*) DESC"

dbGetQuery(con, arr_prm_type_2016)
```

```
## # A tibble: 33 x 2
##    primary_type             f0_
##    <chr>                  <int>
##  1 NARCOTICS              13327
##  2 BATTERY                10332
##  3 THEFT                   6522
##  4 CRIMINAL TRESPASS       3724
##  5 ASSAULT                 3492
##  6 OTHER OFFENSE           3415
##  7 WEAPONS VIOLATION       2511
##  8 CRIMINAL DAMAGE         1669
##  9 PUBLIC PEACE VIOLATION  1116
## 10 MOTOR VEHICLE THEFT     1097
## # i 23 more rows
```

We can also use the date for grouping. Count the number of arrests grouped by hour of the day in 2016. You can extract the latter information from date via EXTRACT(HOUR FROM date). Which time of the day is associated with the most arrests?

When counting the number of arrests grouped by the hour of the day in 2016, it shows that the hour “10” is associated with the highest number of arrests, totaling 5,306. Following closely is the hour “11” with 5,200 arrests.

```
SELECT EXTRACT(HOUR FROM date), count(*)
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY EXTRACT(HOUR FROM date)
ORDER BY count(*) DESC;
```

Table 3: Displaying records 1 - 10

f0_	f1_
10	5306
11	5200
12	4942
7	4900
8	4735

f0_	f1_
9	4675
1	4288
6	4261
2	4029
3	3750

Focus only on HOMICIDE and count the number of arrests for this incident type, grouped by year. List the results in descending order.

When counting the number of arrests for the 'HOMICIDE' incident type, it shows that in the year 2001, there was the highest number of arrests, totaling 430 cases. Following closely, in the year 2002, there were 423 cases of arrests for 'HOMICIDE'.

```
SELECT year, count(*)
FROM crime
WHERE primary_type = 'HOMICIDE' AND arrest = TRUE
GROUP BY year
ORDER BY count(*) DESC;
```

Table 4: Displaying records 1 - 10

year	f0_
2001	430
2002	423
2003	379
2020	339
2004	293
2008	286
2016	286
2006	281
2005	281
2021	275

Find out which districts have the highest numbers of arrests in 2015 and 2016. That is, count the number of arrests in 2015 and 2016, grouped by year and district. List the results in descending order.

In 2016, District 11 had the highest number of arrests, totaling 6,575 cases. Similarly, in 2015, District 11 also had the highest number of arrests, with 8,974 cases.

```
SELECT year, district, count(*)
FROM crime
WHERE (year = 2015 OR year = 2016) AND arrest = TRUE
GROUP BY year, district
ORDER BY year DESC, count(*) DESC;
```

Table 5: Displaying records 1 - 10

year	district	f0_
2016	11	6575
2016	7	3654
2016	6	3447
2016	15	3072
2016	10	2951
2016	8	2948
2016	25	2947
2016	4	2837
2016	5	2701
2016	9	2592

Lets switch to writing queries from within R via the DBI package. Create a query object that counts the number of arrests grouped by primary_type of district 11 in year 2016. The results should be displayed in descending order. Execute the query.

In District 11 in the year 2016, the primary_type "NARCOTICS" had the highest number of cases, totaling 3,634, followed by "BATTERY" with 635 cases.

```r
arr_dis11_2016 <- "SELECT primary_type, count(*)
FROM crime
WHERE (district = 11 AND year = 2016) AND arrest = TRUE
GROUP BY primary_type
ORDER BY count(*) DESC"

dbGetQuery(con, arr_dis11_2016)
```

```
## # A tibble: 27 x 2
##    primary_type                       f0_
##    <chr>                            <int>
##  1 NARCOTICS                         3634
##  2 BATTERY                            635
##  3 PROSTITUTION                       511
##  4 WEAPONS VIOLATION                  303
##  5 OTHER OFFENSE                      255
##  6 ASSAULT                            206
##  7 CRIMINAL TRESPASS                  205
##  8 PUBLIC PEACE VIOLATION             135
##  9 INTERFERENCE WITH PUBLIC OFFICER   119
## 10 CRIMINAL DAMAGE                    106
## # i 17 more rows
```

Try to write the very same query, now using the dbplyr package. For this, you need to first map the crime table to a tibble object in R.

```r
# map the `crime` table to a tibble object in R.
crime_data <- tbl(con, "crime")
```

```
## Warning: <BigQueryConnection> uses an old dbplyr interface
## i Please install a newer version of the package or contact the maintainer
## This warning is displayed once every 8 hours.
```

```r
str(crime_data)
```

```
## List of 2
##  $ unique_key          :List of 2
##   ..$ con  :Formal class 'BigQueryConnection' [package "bigrquery"] with 7 slots
##   .. .. ..@ project       : chr "bigquery-public-data"
##   .. .. ..@ dataset       : chr "chicago_crime"
##   .. .. ..@ billing       : chr "surv727-project"
##   .. .. ..@ use_legacy_sql: logi FALSE
##   .. .. ..@ page_size     : int 10000
##   .. .. ..@ quiet         : logi NA
##   .. .. ..@ bigint        : chr "integer"
##   ..$ disco: NULL
##   ..- attr(*, "class")= chr [1:4] "src_BigQueryConnection" "src_dbi" "src_sql" "src"
##  $ case_number         :List of 5
##   ..$ x         : 'ident' chr "crime"
##   ..$ vars      : chr [1:22] "unique_key" "case_number" "date" "block" ...
##   ..$ group_vars: chr(0)
##   ..$ order_vars: NULL
##   ..$ frame     : NULL
##   ..- attr(*, "class")= chr [1:3] "lazy_base_remote_query" "lazy_base_query" "lazy_query"
##  - attr(*, "class")= chr [1:5] "tbl_BigQueryConnection" "tbl_dbi" "tbl_sql" "tbl_lazy" ...
```

```r
class(crime_data)
```

```
## [1] "tbl_BigQueryConnection" "tbl_dbi"                "tbl_sql"
## [4] "tbl_lazy"               "tbl"
```

```r
head(crime_data)
```

```
## # Source:   SQL [6 x 22]
## # Database: BigQueryConnection
##   unique_key case_number date                block            iucr  primary_type
##        <int> <chr>       <dttm>              <chr>            <chr> <chr>
## 1       5039 HR349878    2009-05-30 11:53:00 009XX E 104TH ST 0110  HOMICIDE
## 2        747 G194349     2001-04-06 02:00:00 104XX S STATE ST 0110  HOMICIDE
## 3      26773 JF247428    2022-05-18 10:06:00 002XX W 105TH ST 0110  HOMICIDE
## 4       2014 HJ193229    2003-02-19 10:35:00 105XX S STATE ST 0110  HOMICIDE
## 5    5779659 HN574317    2007-09-06 09:30:00 103XX S COTTAGE~ 0261  CRIM SEXUAL~
## 6    2897697 HJ570499    2003-07-30 10:00:00 006XX E 103RD ST 0263  CRIM SEXUAL~
## # i 16 more variables: description <chr>, location_description <chr>,
## #   arrest <lgl>, domestic <lgl>, beat <int>, district <int>, ward <int>,
## #   community_area <int>, fbi_code <chr>, x_coordinate <dbl>,
## #   y_coordinate <dbl>, year <int>, updated_on <dttm>, latitude <dbl>,
## #   longitude <dbl>, location <chr>
```

Again, count the number of arrests grouped by primary_type of district 11 in year 2016, now using dplyr syntax.

In District 11 in the year 2016, the primary_type “NARCOTICS” had the highest number of cases, totaling 3,634, followed by “BATTERY” with 635 cases. These values match those obtained using the SQL code.

```
crime_data %>%
  filter(district == 11, year == 2016, arrest == TRUE) %>%
  group_by(primary_type) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # Source:     SQL [?? x 2]
## # Database:   BigQueryConnection
## # Ordered by: desc(count)
##    primary_type                      count
##    <chr>                             <int>
##  1 NARCOTICS                          3634
##  2 BATTERY                             635
##  3 PROSTITUTION                        511
##  4 WEAPONS VIOLATION                   303
##  5 OTHER OFFENSE                       255
##  6 ASSAULT                             206
##  7 CRIMINAL TRESPASS                   205
##  8 PUBLIC PEACE VIOLATION              135
##  9 INTERFERENCE WITH PUBLIC OFFICER    119
## 10 CRIMINAL DAMAGE                     106
## # i more rows
```

Count the number of arrests grouped by primary_type and year, still only for district 11. Arrange the result by year.

The code below displays the count of arrests grouped by primary_type and year specifically for District 11. For instance, in the year 2023, there were 403 arrests related to “WEAPONS VIOLATION.”

```
crime_data %>%
  filter(district == 11, arrest == TRUE) %>%
  group_by(year, primary_type) %>%
  summarize(count = n()) %>%
  arrange(desc(year))
```

```
## `summarise()` has grouped output by "year". You can override using the
## `.groups` argument.
```

```
## # Source:     SQL [?? x 3]
## # Database:   BigQueryConnection
## # Groups:     year
## # Ordered by: desc(year)
##     year primary_type           count
##    <int> <chr>                  <int>
##  1  2023 STALKING                   3
##  2  2023 PUBLIC PEACE VIOLATION    27
##  3  2023 MOTOR VEHICLE THEFT        48
##  4  2023 CRIMINAL SEXUAL ASSAULT     1
##  5  2023 CRIMINAL DAMAGE            51
##  6  2023 BATTERY                   298
##  7  2023 CRIMINAL TRESPASS          43
##  8  2023 ROBBERY                    26
##  9  2023 BURGLARY                    5
## 10  2023 DECEPTIVE PRACTICE         27
## # i more rows
```

Assign the results of the query above to a local R object.

We can assign the query above to a local R object ‘crim_query’.

```
crim_query <- crime_data %>%
  filter(district == 11, arrest == TRUE) %>%
  group_by(year, primary_type) %>%
  summarize(count = n()) %>%
  arrange(desc(year)) %>%
  collect()
```

```
## ‘summarise()‘ has grouped output by "year". You can override using the
## ‘.groups‘ argument.
```

Confirm that you pulled the data to the local environment by displaying the first ten rows of the saved data set.

The code below shows the first ten rows of the saved data set (crim_query) in the local environment.

```
head(crim_query, 10)
```

```
## # A tibble: 10 x 3
## # Groups:   year [1]
##     year primary_type               count
##    <int> <chr>                      <int>
##  1  2023 OTHER OFFENSE                176
##  2  2023 LIQUOR LAW VIOLATION           5
##  3  2023 SEX OFFENSE                    4
##  4  2023 WEAPONS VIOLATION            403
##  5  2023 PROSTITUTION                 139
##  6  2023 THEFT                         25
##  7  2023 OFFENSE INVOLVING CHILDREN     8
##  8  2023 NARCOTICS                   1224
##  9  2023 ROBBERY                       26
## 10  2023 CRIMINAL TRESPASS             43
```

```
str(crim_query)
```

```
## gropd_df [588 x 3] (S3: grouped_df/tbl_df/tbl/data.frame)
##  $ year        : int [1:588] 2023 2023 2023 2023 2023 2023 2023 2023 2023 2023 ...
##  $ primary_type: chr [1:588] "OTHER OFFENSE" "LIQUOR LAW VIOLATION" "SEX OFFENSE" "WEAPONS VIOLATION
##  $ count       : int [1:588] 176 5 4 403 139 25 8 1224 26 43 ...
##  - attr(*, "groups")= tibble [23 x 2] (S3: tbl_df/tbl/data.frame)
##   ..$ year : int [1:23] 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 ...
##   ..$ .rows: list<int> [1:23]
##   .. ..$ : int [1:25] 564 565 566 567 568 569 570 571 572 573 ...
##   .. ..$ : int [1:26] 538 539 540 541 542 543 544 545 546 547 ...
##   .. ..$ : int [1:25] 513 514 515 516 517 518 519 520 521 522 ...
##   .. ..$ : int [1:25] 488 489 490 491 492 493 494 495 496 497 ...
##   .. ..$ : int [1:26] 462 463 464 465 466 467 468 469 470 471 ...
##   .. ..$ : int [1:26] 436 437 438 439 440 441 442 443 444 445 ...
##   .. ..$ : int [1:25] 411 412 413 414 415 416 417 418 419 420 ...
##   .. ..$ : int [1:25] 386 387 388 389 390 391 392 393 394 395 ...
##   .. ..$ : int [1:26] 360 361 362 363 364 365 366 367 368 369 ...
##   .. ..$ : int [1:27] 333 334 335 336 337 338 339 340 341 342 ...
##   .. ..$ : int [1:27] 306 307 308 309 310 311 312 313 314 315 ...
##   .. ..$ : int [1:26] 280 281 282 283 284 285 286 287 288 289 ...
##   .. ..$ : int [1:26] 254 255 256 257 258 259 260 261 262 263 ...
##   .. ..$ : int [1:27] 227 228 229 230 231 232 233 234 235 236 ...
##   .. ..$ : int [1:25] 202 203 204 205 206 207 208 209 210 211 ...
##   .. ..$ : int [1:27] 175 176 177 178 179 180 181 182 183 184 ...
##   .. ..$ : int [1:26] 149 150 151 152 153 154 155 156 157 158 ...
##   .. ..$ : int [1:27] 122 123 124 125 126 127 128 129 130 131 ...
##   .. ..$ : int [1:27] 95 96 97 98 99 100 101 102 103 104 ...
##   .. ..$ : int [1:24] 71 72 73 74 75 76 77 78 79 80 ...
##   .. ..$ : int [1:23] 48 49 50 51 52 53 54 55 56 57 ...
##   .. ..$ : int [1:25] 23 24 25 26 27 28 29 30 31 32 ...
##   .. ..$ : int [1:22] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..@ ptype: int(0)
##   ..- attr(*, ".drop")= logi TRUE
```

Close the connection.

```
dbDisconnect(con)
```