Assignment 4

Kate Lamoreaux

2023-10-25

Link to github repository: https://github.com/katelmrx/SURV727_Assignment4

In this notebook we will use Google BigQuery, "Google's fully managed, petabyte scale, low cost analytics data warehouse". Some instruction on how to connect to Google BigQuery can be found here: https://db.rstudio.com/databases/big-query/.

You will need to set up a Google account with a project to be able to use this service. We will be using a public dataset that comes with 1 TB/mo of free processing on Google BigQuery. As long as you do not repeat the work in this notebook constantly, you should be fine with just the free tier.

Go to https://console.cloud.google.com and make sure you are logged in a non-university Google account. **This may not work on a university G Suite account because of restrictions on those accounts.** Create a new project by navigating to the dropdown menu at the top (it might say "Select a project") and selecting "New Project" in the window that pops up. Name it something useful.

```
project <- "surv727-fcdd"
```

After you have initialized a project, paste your project ID into the following chunk.

```
con <- dbConnect(
  bigrquery::bigquery(),
  project = "bigquery-public-data",
  dataset = "chicago_crime",
  billing = project
)
con
```

We will connect to a public database, the Chicago crime database, which has data on crime in Chicago.

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

We can look at the available tables in this database using `dbListTables`.

Note: When you run this code, you will be sent to a browser and have to give Google permissions to Tidyverse API Packages. **Make sure you select all to give access or else your code will not run.**

```
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 'katelmrx@gmail.com'.
## [1] "crime"
```

Information on the ‘crime‘ table can be found here:

https://cloud.google.com/bigquery/public-data/chicago-crime-data

```sql
SELECT COUNT(*) as COUNT
FROM crime
WHERE year = 2016
LIMIT 10;
```

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.

Table 1: 1 records

COUNT
269854

According to our first query, there are 269,854 rows in the crime table for the year 2016.

```sql
SELECT primary_type, COUNT(*) as COUNT
FROM crime
WHERE year= 2016 and arrest = TRUE
GROUP BY primary_type
ORDER BY COUNT(*) DESC
LIMIT 10;
```

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.

Table 2: Displaying records 1 - 10

primary_type	COUNT
NARCOTICS	13327
BATTERY	10332
THEFT	6522
CRIMINAL TRESPASS	3724
ASSAULT	3492
OTHER OFFENSE	3415
WEAPONS VIOLATION	2511
CRIMINAL DAMAGE	1669
PUBLIC PEACE VIOLATION	1116
MOTOR VEHICLE THEFT	1097

Arrests for narcotics, battery, and theft were the most common in 2016.

####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?

```
SELECT EXTRACT(HOUR FROM date) AS hour_of_day, COUNT(*) as COUNT
FROM crime
WHERE year= 2016 and arrest = TRUE
GROUP BY hour_of_day
ORDER BY COUNT(*) DESC
LIMIT 12;
```

Table 3: Displaying records 1 - 10

hour_of_day	COUNT
10	5306
11	5200
12	4944
7	4900
8	4735
9	4675
1	4288
6	4261
2	4029
3	3750

Later hours of the data are associated with more arrests. Hour 10 is associated with the most arrests.

```

SELECT YEAR, COUNT(*) as COUNT
FROM crime
WHERE primary_type='HOMICIDE' and arrest = TRUE
GROUP BY year
ORDER BY COUNT(*) DESC
LIMIT 20;
```

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

Table 4: Displaying records 1 - 10

YEAR	COUNT
2001	430
2002	424
2003	379
2020	341
2004	293
2008	286
2016	286
2005	281
2006	281
2022	280

Years 2001-2003 had the highest number of arrests for homicides.

```
SELECT YEAR, DISTRICT, COUNT(*) as COUNT
FROM crime
WHERE arrest = TRUE AND (YEAR = 2015 OR YEAR = 2016)
GROUP BY YEAR, DISTRICT
ORDER BY COUNT(*) DESC
LIMIT 20;
```

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.

Table 5: Displaying records 1 - 10

YEAR	DISTRICT	COUNT
2015	11	8974
2016	11	6575
2015	7	5549
2015	15	4514
2015	6	4473
2015	25	4448
2015	4	4325
2015	8	4112
2016	7	3654
2015	10	3621

Districts 11 and 7, respectively, had the highest and second-highest numbers of arrests in both 2015 and 2016.

```
dbiquery1 <- "SELECT primary_type, COUNT(*) as COUNT
        FROM crime
        WHERE (arrest = TRUE AND year = 2016) AND district = 11
        GROUP BY primary_type
        ORDER BY COUNT(*) DESC
        LIMIT 10"
```

Let's 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. The above code assigns `dbiquery1` as a query object that counts the number of arrests in descending order grouped by `primary_type` within District 11 in the year 2016.

```
dbGetQuery(con, dbiquery1)
```

Execute the query.

```
## # A tibble: 10 x 2
##    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
```

The above code uses the function dbGetQuery() to execute query `dbiquery1`. 2016 arrests in District 11 were mostly for narcotics, battery, and prostitution.

```
crimetibble <- tbl(con, 'crime')
```

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.

```
## 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.
```

The above code maps the `crime` table to a tibble object in R, titled `crimetibble`.

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

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

```
## # 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
```

The above code counts the number of arrests in descending order grouped by `primary_type` within District 11 in the year 2016, using `dplyr` syntax. Again, we see the same numbers as we did in DBI query `dbiquery1`.

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

Count the number of arrests grouped by primary_type and year, still only for district 11. Arrange the result by 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 WEAPONS VIOLATION            437
##  2  2023 PROSTITUTION                 143
##  3  2023 CRIMINAL TRESPASS             51
##  4  2023 HOMICIDE                      17
##  5  2023 OFFENSE INVOLVING CHILDREN    10
##  6  2023 SEX OFFENSE                    5
##  7  2023 ASSAULT                      114
##  8  2023 ROBBERY                       28
##  9  2023 OTHER OFFENSE                189
## 10  2023 LIQUOR LAW VIOLATION           5
## # i more rows
```

This code counts the number of arrests grouped by `primary_type` and `year`, still only for district 11. With the results arranged by `year`, however, the count of arrests is no longer in descending order.

```
arrestsbytypeandyear_chidistrict11 <- crimetibble %>%
  filter(arrest == TRUE, district == 11) %>%
  group_by(year, primary_type) %>%
  summarize(count = n()) %>%
  arrange(desc(year))
```

Assign the results of the query above to a local R object. The above code assigns the results of our prior SQL query using `dplyr` syntax to a local R object titled, arrestsbytypeandyear_chidistrict11.

```
head(arrestsbytypeandyear_chidistrict11, 10)
```

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

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

```
## # Source:     SQL [10 x 3]
## # Database:   BigQueryConnection
## # Groups:     year
## # Ordered by: desc(year)
##     year primary_type                      count
##    <int> <chr>                             <int>
##   1  2023 INTERFERENCE WITH PUBLIC OFFICER     43
##   2  2023 OFFENSE INVOLVING CHILDREN           10
##   3  2023 PROSTITUTION                        143
##   4  2023 THEFT                                27
##   5  2023 CRIMINAL DAMAGE                      56
##   6  2023 CONCEALED CARRY LICENSE VIOLATION     5
##   7  2023 NARCOTICS                          1415
##   8  2023 WEAPONS VIOLATION                   437
##   9  2023 STALKING                              3
##  10  2023 PUBLIC PEACE VIOLATION               28
```

The above code confirms I pulled the data into my local environment by printing the first ten rows of my new R object, arrestsbytypeandyear_chidistrict11.

```
#code to close database connection
dbDisconnect(con)
```

Close the connection.