Assignment 4

Due at 11:59pm on November 4. Nicole Blackburn

This is an individual assignment. Turn in this assignment as an HTML or PDF file to ELMS. Make sure to include the R Markdown or Quarto file that was used to generate it. Include the GitHub link for the repository containing these files.

GitHub link to repository

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.

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

```r
project <- "surv727-475919"
```

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

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

```
<BigQueryConnection>
  Dataset: bigquery-public-data.chicago_crime
  Billing: surv727-475919
```

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 'npbvbck26@gmail.com'.

Auto-refreshing stale OAuth token.

[1] "crime"
```

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

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

Write a first query that counts the number of rows of the ‘crime‘ table in the year 2015. Use code chunks with {sql connection = con} in order to write SQL code within the document.

```
SELECT count(primary_type) AS primary_count, count(*) AS overall_count -- counting non-missi
FROM crime
WHERE year = 2015
LIMIT 10;
```

Table 1: 1 records

primary_count	overall_count
264874	264874

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.

```
SELECT count(arrest) AS arrest_count, primary_type
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY primary_type
ORDER BY arrest_count DESC
LIMIT 10;
```

Table 2: Displaying records 1 - 10

arrest_count	primary_type
13327	NARCOTICS
10334	BATTERY
6522	THEFT
3724	CRIMINAL TRESPASS
3494	ASSAULT
3416	OTHER OFFENSE
2510	WEAPONS VIOLATION
1669	CRIMINAL DAMAGE
1116	PUBLIC PEACE VIOLATION
1098	MOTOR VEHICLE THEFT

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?

- hour = 19, or 7pm is associated with the most arrests, 3843.

```
SELECT count(arrest) AS arrest_count, EXTRACT(HOUR FROM date) AS hour
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY hour
```

```
ORDER BY arrest_count DESC
LIMIT 10;
```

Table 3: Displaying records 1 - 10

arrest_count	hour
3843	19
3482	18
3303	20
2962	21
2933	16
2896	22
2894	11
2821	17
2788	12
2775	14

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

```
SELECT year, count(arrest) AS homicide_arrest_count
FROM crime
WHERE primary_type = 'HOMICIDE' AND arrest = TRUE
GROUP BY year
ORDER BY homicide_arrest_count DESC
LIMIT 10;
```

Table 4: Displaying records 1 - 10

year	homicide_arrest_count
2001	431
2002	428
2003	386
2020	356
2022	321
2021	296
2004	294
2016	292
2008	288

year	homicide_arrest_count
2006	284

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.

```
SELECT year, district, count(arrest) AS arrest_count
FROM crime
WHERE (year = 2015 AND arrest = TRUE) OR (year = 2016 AND arrest = TRUE)
GROUP BY year, district
ORDER BY district, arrest_count DESC
LIMIT 10;
```

Table 5: Displaying records 1 - 10

year	district	arrest_count
2015	1	2802
2016	1	2548
2015	2	1940
2016	2	1704
2015	3	3047
2016	3	2362
2015	4	4326
2016	4	2841
2015	5	3087
2016	5	2704

- Based on the announcement example on campus, I added district to ORDER BY. Based on just the question wording, I would've left it out and arranged descending by arrest count.

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.

```
sql <- "SELECT count(arrest) AS arrest_count, primary_type
        FROM `bigquery-public-data.chicago_crime.crime`
        WHERE year = 2016 AND arrest = TRUE AND district = 11
        GROUP BY primary_type
        ORDER BY arrest_count DESC
        LIMIT 10"
```

Execute the query.

```
dbGetQuery(con, sql)
```

```
# A tibble: 10 x 2
   arrest_count primary_type
          <int> <chr>
 1         3634 NARCOTICS
 2          635 BATTERY
 3          511 PROSTITUTION
 4          303 WEAPONS VIOLATION
 5          255 OTHER OFFENSE
 6          207 ASSAULT
 7          205 CRIMINAL TRESPASS
 8          135 PUBLIC PEACE VIOLATION
 9          119 INTERFERENCE WITH PUBLIC OFFICER
10          106 CRIMINAL DAMAGE
```

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.

```
crim <- tbl(con, "crime")
str(crim)
```

```
List of 2
 $ src       :List of 2
  ..$ con  :Formal class 'BigQueryConnection' [package "bigrquery"] with 7 slots
  .. .. ..@ project       : chr "bigquery-public-data"
  .. .. ..@ dataset       : chr "chicago_crime"
  .. .. ..@ billing       : chr "surv727-475919"
  .. .. ..@ 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"
 $ lazy_query:List of 6
```

```
  ..$ x         : 'dbplyr_table_path' chr "`crime`"
  ..$ vars      : chr [1:22] "unique_key" "case_number" "date" "block" ...
  ..$ group_vars: chr(0)
  ..$ order_vars: NULL
  ..$ frame     : NULL
  ..$ is_view   : logi FALSE
  ..- 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" ...
```

```
class(crim)
```

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

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

```
crim |>
  select(arrest, primary_type, year, district) |>
  filter(year == 2016, arrest == TRUE, district == 11) |>
  group_by(primary_type) |>
  summarise(arrest_count = n()) |>
  arrange(desc(arrest_count)) |>
  head(10)
```

```
# Source:     SQL [?? x 2]
# Database:   BigQueryConnection
# Ordered by: desc(arrest_count)
   primary_type                     arrest_count
   <chr>                                   <int>
 1 NARCOTICS                                3634
 2 BATTERY                                   635
 3 PROSTITUTION                              511
 4 WEAPONS VIOLATION                         303
 5 OTHER OFFENSE                             255
 6 ASSAULT                                   207
 7 CRIMINAL TRESPASS                         205
 8 PUBLIC PEACE VIOLATION                    135
 9 INTERFERENCE WITH PUBLIC OFFICER          119
10 CRIMINAL DAMAGE                           106
```

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

```r
crim |>
  select(arrest, primary_type, year, district) |>
  filter(arrest == TRUE, district == 11) |>
  group_by(primary_type, year) |>
  summarise(arrest_count = n()) |>
  arrange(year) |>
  head(10)
```

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

```
# Source:     SQL [?? x 3]
# Database:   BigQueryConnection
# Groups:     primary_type
# Ordered by: year
   primary_type          year arrest_count
   <chr>                <int>        <int>
 1 WEAPONS VIOLATION     2001          236
 2 GAMBLING              2001           71
 3 OTHER OFFENSE         2001          266
 4 PROSTITUTION          2001          424
 5 LIQUOR LAW VIOLATION  2001           49
 6 CRIMINAL DAMAGE       2001          163
 7 STALKING              2001            1
 8 ROBBERY               2001           97
 9 DECEPTIVE PRACTICE    2001           84
10 ASSAULT               2001          322
```

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

```r
object <-
  crim |>
  select(arrest, primary_type, year, district) |>
  filter(arrest == TRUE, district == 11) |>
  group_by(primary_type, year) |>
  summarise(arrest_count = n()) |>
  arrange(year) |>
  head(10) |>
  collect()
```

```
`summarise()` has grouped output by "primary_type". 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.

```r
object |>
  slice_head(n = 10)
```

```
# A tibble: 10 x 3
# Groups:   primary_type [10]
   primary_type          year arrest_count
   <chr>                <int>        <int>
 1 ASSAULT               2001          322
 2 CRIMINAL DAMAGE       2001          163
 3 DECEPTIVE PRACTICE    2001           84
 4 GAMBLING              2001           71
 5 LIQUOR LAW VIOLATION  2001           49
 6 OTHER OFFENSE         2001          266
 7 PROSTITUTION          2001          424
 8 ROBBERY               2001           97
 9 STALKING              2001            1
10 WEAPONS VIOLATION     2001          236
```

Close the connection.

```r
dbDisconnect(con)
```