Assignment 4

Due at 11:59pm on November 4.

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.

Link to github repo: https://github.com/danapopky/Assignment-4

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.

```
project <- "surv727-assignment-4-475919"
```

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

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

```
)
#con
```

DP NOTE: MUST PRESS ENTER IN THE CONSOLE AFTER RUNNING THIS LINE OF CODE TO CONTINUE.

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 'danapopky@gmail.com'.

[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 2016. 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 primary_type, COUNTIF(arrest) AS arrests2016
FROM crime
WHERE year = 2016
GROUP BY primary_type
ORDER BY arrests2016 DESC
LIMIT 10;
```

Table 2: Displaying records 1 - 10

primary_type	arrests2016
NARCOTICS	13327
BATTERY	10334
THEFT	6522
CRIMINAL TRESPASS	3724
ASSAULT	3494
OTHER OFFENSE	3416
WEAPONS VIOLATION	2510
CRIMINAL DAMAGE	1669
PUBLIC PEACE VIOLATION	1116
MOTOR VEHICLE THEFT	1098

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?

7pm (or, 19:00) is the hour of the day most associated with the most arrests.

```
SELECT
  EXTRACT(HOUR FROM date) as hour,
  COUNTIF(arrest) AS arrests2016
```

```
FROM crime
WHERE year = 2016
GROUP BY hour
ORDER BY arrests2016 DESC
LIMIT 24;
```

Table 3: Displaying records 1 - 10

hour	arrests2016
19	3843
18	3482
20	3303
21	2962
16	2933
22	2896
11	2894
17	2821
12	2788
14	2775

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, COUNTIF(arrest) AS homicides
FROM crime
WHERE primary_type = 'HOMICIDE'
GROUP BY year
ORDER BY homicides DESC
LIMIT 10;
```

Table 4: Displaying records 1 - 10

year	homicides
2001	431
2002	428
2003	386
2020	356
2022	321

year	homicides
2021	296
2004	294
2016	292
2008	288
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, COUNTIF(arrest) AS arrests
FROM crime
WHERE year IN (2015, 2016)
GROUP BY year, district
ORDER BY arrests DESC
LIMIT 10;
```

Table 5: Displaying records 1 - 10

year	district	arrests
2015	11	8975
2016	11	6578
2015	7	5549
2015	15	4514
2015	6	4476
2015	25	4451
2015	4	4326
2015	8	4115
2016	7	3656
2015	10	3628

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.

```
sql <-   "SELECT primary_type, COUNTIF(arrest) AS arrests
  FROM crime
  WHERE year = 2016 AND district = 11
  GROUP BY primary_type
  ORDER BY arrests DESC"

dbGetQuery(con, sql)
```

```
# A tibble: 30 x 2
   primary_type                     arrests
   <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
# i 20 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.

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

```
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-assignment-4-475919"
  .. .. ..@ 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(crime_table )
```

```
[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.

```
crime_table %>%
  select(primary_type, arrest, district, year) %>%
  filter(year == 2016, district == 11) %>%
  group_by(primary_type) %>%
  summarise(arrests = sum(as.integer(arrest), na.rm = TRUE)) %>%
  arrange(desc(arrests)) %>%
  head(10)
```

```
# Source:     SQL [?? x 2]
# Database:   BigQueryConnection
# Ordered by: desc(arrests)
   primary_type                      arrests
   <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`.

```
crime_table %>%
  select(primary_type, arrest, district, year) %>%
  filter(district == 11) %>%
  group_by(year, primary_type) %>%
  summarise(arrests = sum(as.integer(arrest), na.rm = TRUE)) %>%
  arrange(desc(year), desc(arrests)) %>%
  head(50)
```

```
`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), desc(arrests)
    year primary_type                      arrests
   <int> <chr>                               <int>
 1  2025 NARCOTICS                           2077
 2  2025 BATTERY                              359
 3  2025 WEAPONS VIOLATION                    306
 4  2025 OTHER OFFENSE                        119
 5  2025 ASSAULT                               98
 6  2025 CRIMINAL TRESPASS                     59
 7  2025 INTERFERENCE WITH PUBLIC OFFICER      53
 8  2025 CRIMINAL DAMAGE                       51
 9  2025 PROSTITUTION                          46
10  2025 MOTOR VEHICLE THEFT                   42
# i more rows
```

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

```
sql_data <-
crime_table %>%
  select(primary_type, arrest, district, year) %>%
  filter(district == 11) %>%
  group_by(year, primary_type) %>%
  summarise(arrests = sum(as.integer(arrest), na.rm = TRUE)) %>%
  arrange(desc(year), desc(arrests)) %>%
  head(50)
```

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

```
head(sql_data)
```

```
`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), desc(arrests)
   year primary_type      arrests
  <int> <chr>               <int>
1  2025 NARCOTICS            2077
2  2025 BATTERY               359
3  2025 WEAPONS VIOLATION     306
4  2025 OTHER OFFENSE         119
5  2025 ASSAULT                98
6  2025 CRIMINAL TRESPASS      59
```

Close the connection.

```
dbDisconnect(con)
```