Assignment 4

Due at 11:59pm on November 4.

Effy Tang

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.

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 <- "survmeth-727-assignment-4"
```

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
```

```
<BigQueryConnection>
  Dataset: bigquery-public-data.chicago_crime
  Billing: survmeth-727-assignment-4
```

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 'effygziven@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 = 2016
LIMIT 10;
```

Table 1: 1 records

primary_count	overall_count
269938	269938

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,
  COUNT(*) AS num_arrests
FROM crime
WHERE year = 2016
  AND arrest = TRUE
GROUP BY primary_type
ORDER BY num_arrests DESC
```

Table 2: Displaying records 1 - 10

primary_type	num_arrests
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?

```
SELECT
  EXTRACT(HOUR FROM date) AS hour_of_day,
  COUNT(*) AS num_arrests
FROM crime
```

```
WHERE year = 2016
  AND arrest = TRUE
GROUP BY hour_of_day
ORDER BY num_arrests DESC
```

Table 3: Displaying records 1 - 10

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

- This query shows the number of arrests by hour of the day (0-23, where 0 is midnight and 23 is 11 PM) in 2016, sorted from highest to lowest. The hour with the most arrests appears at the top of the results, which is 19.

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(*) AS num_homicide_arrests
FROM crime
WHERE primary_type = 'HOMICIDE'
  AND arrest = TRUE
GROUP BY year
ORDER BY num_homicide_arrests DESC
```

Table 4: Displaying records 1 - 10

year	num_homicide_arrests
2001	431
2002	428

year	num_homicide_arrests
2003	386
2020	356
2022	321
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,
  COUNT(*) AS num_arrests
FROM crime
WHERE year IN (2015, 2016)
  AND arrest = TRUE
GROUP BY year, district
ORDER BY num_arrests DESC
```

Table 5: Displaying records 1 - 10

year	district	num_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.

```
query <- "
  SELECT
    primary_type,
    COUNT(*) AS num_arrests
  FROM crime
  WHERE year = 2016
    AND district = 11
    AND arrest = TRUE
  GROUP BY primary_type
  ORDER BY num_arrests DESC
"

arrests_district_11 <- dbGetQuery(con, query)

arrests_district_11
```

```
# A tibble: 27 x 2
   primary_type                     num_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 17 more rows
```

Execute the query.

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.

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

```
crime_tbl <- tbl(con, "crime")

results_dplyr <- crime_tbl %>%
```

```
  filter(year == 2016,
         arrest == TRUE,
         district == 11) %>%
  group_by(primary_type) %>%
  summarise(arrest_count = n()) %>%
  arrange(desc(arrest_count))

results_dplyr
```

```
# 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
# i more rows
```

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

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

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

```
results_by_year <- crime_tbl %>%
  filter(arrest == TRUE,
         district == 11) %>%
  group_by(year, primary_type) %>%
  summarise(arrest_count = n()) %>%
  arrange(year) %>%
  collect()
```

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

```
results_by_year
```

```
# A tibble: 638 x 3
# Groups:   year [25]
    year primary_type           arrest_count
   <int> <chr>                         <int>
 1  2001 CRIMINAL DAMAGE                163
 2  2001 HOMICIDE                        49
 3  2001 KIDNAPPING                       4
 4  2001 OTHER OFFENSE                  266
 5  2001 ASSAULT                        322
 6  2001 WEAPONS VIOLATION              236
 7  2001 PUBLIC PEACE VIOLATION          34
 8  2001 ROBBERY                         97
 9  2001 LIQUOR LAW VIOLATION            49
10  2001 CRIMINAL TRESPASS              389
# i 628 more rows
```

```
head(results_by_year,10)
```

```
# A tibble: 10 x 3
# Groups:   year [1]
    year primary_type           arrest_count
   <int> <chr>                         <int>
 1  2001 CRIMINAL DAMAGE                163
 2  2001 HOMICIDE                        49
 3  2001 KIDNAPPING                       4
 4  2001 OTHER OFFENSE                  266
 5  2001 ASSAULT                        322
 6  2001 WEAPONS VIOLATION              236
 7  2001 PUBLIC PEACE VIOLATION          34
 8  2001 ROBBERY                         97
 9  2001 LIQUOR LAW VIOLATION            49
10  2001 CRIMINAL TRESPASS              389
```

Close the connection.

```
dbDisconnect(con)
```