Assignment 4

Due at 11:59pm on November 5.

Molly Fisch-Friedman

GitHub link: https://github.com/mollyfischfriedman/surv727-assignment4

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 <- "surv727-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: surv727-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
  'mollyfischfriedman@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), count(*)
FROM crime
WHERE year = 2016
LIMIT 10;
```

Table 1: 1 records

f0_	f1_
269921	269921

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(*)
FROM crime
WHERE year = 2016
GROUP BY primary_type
ORDER BY count(*) DESC
LIMIT 10;
```

Table 2: Displaying records 1 - 10

primary_type	f0_
THEFT	61623
BATTERY	50301
CRIMINAL DAMAGE	31018
DECEPTIVE PRACTICE	19394
ASSAULT	18743
OTHER OFFENSE	17308
BURGLARY	14289
NARCOTICS	13333
ROBBERY	11960
MOTOR VEHICLE THEFT	11285

In 2016, the category with the highest number of arrests in Chicago was 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?

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

Table 3: Displaying records 1 - 10

f0_	f1_
12	15859
18	15540
19	15098
20	14499
17	14343
15	14336
16	14248
0	13674
21	13483
14	13438

The 12:00-13:00 UTC hour is associated with the most arrests.

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(*)
FROM crime
WHERE primary_type = 'HOMICIDE'
GROUP BY year
ORDER BY count(*) DESC;
```

Table 4: Displaying records 1 - 10

year	f0_
2021	811
2020	796
2016	790
2022	732
2017	676
2001	667
2002	658
2023	628
2003	604
2018	600

2021 saw the most homicide arrests (811).

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(*)
FROM crime
WHERE year = 2015 OR year = 2016
GROUP BY year, district
ORDER BY count(*) DESC
LIMIT 10;
```

Table 5: Displaying records 1 - 10

year	district	f0__
2015	11	19535
2016	11	18657
2016	8	17578
2015	8	17351
2016	6	16233
2015	6	16077
2015	4	15888
2015	7	15778
2015	25	15088
2016	4	15038

The largest number of arrests in 2015 and 2016 occurred in 2015 in district 11 (19,535).

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 <- paste0("SELECT primary_type, count(*) FROM crime WHERE year = 2016 ",
                "AND district = 11 GROUP BY primary_type ORDER BY count(*) ",
                "DESC LIMIT 10;")
```

Execute the query.

```
dbGetQuery(con, query)
```

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

```
   primary_type          f0_
   <chr>                <int>
 1 BATTERY               3906
 2 NARCOTICS             3635
 3 THEFT                 2043
 4 CRIMINAL DAMAGE       1775
 5 ASSAULT               1330
 6 OTHER OFFENSE         1045
 7 ROBBERY               1007
 8 MOTOR VEHICLE THEFT    776
 9 DECEPTIVE PRACTICE     611
10 PROSTITUTION           511
```

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

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

```
crime %>%
  filter(year==2016 & district==11) %>%
  group_by(primary_type) %>%
  summarise(count=n()) %>%
  arrange(desc(count))
```

```
# Source:     SQL [?? x 2]
# Database:   BigQueryConnection
# Ordered by: desc(count)
   primary_type        count
   <chr>               <int>
 1 BATTERY              3906
 2 NARCOTICS            3635
 3 THEFT                2043
 4 CRIMINAL DAMAGE      1775
 5 ASSAULT              1330
 6 OTHER OFFENSE        1045
 7 ROBBERY              1007
 8 MOTOR VEHICLE THEFT   776
 9 DECEPTIVE PRACTICE    611
```

```
10 PROSTITUTION          511
# i more rows
```

This output matches the output from DBI.

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

```
crime %>%
  filter(district==11) %>%
  group_by(primary_type, year) %>%
  summarise(count=n()) %>%
  arrange(year)
```

```
`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 count
   <chr>                      <int> <int>
 1 ROBBERY                     2001  1243
 2 OTHER OFFENSE               2001  1150
 3 PROSTITUTION                2001   424
 4 DECEPTIVE PRACTICE          2001   346
 5 OFFENSE INVOLVING CHILDREN  2001   140
 6 INTIMIDATION                2001     8
 7 CRIM SEXUAL ASSAULT         2001   101
 8 GAMBLING                    2001    71
 9 THEFT                       2001  3098
10 SEX OFFENSE                 2001    67
# i more rows
```

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

```
dist11 <- crime %>%
  filter(district==11) %>%
  group_by(primary_type, year) %>%
  summarise(count=n()) %>%
  arrange(year)
```

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

```
head(dist11)
```

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

```
# Source:     SQL [6 x 3]
# Database:   BigQueryConnection
# Groups:     primary_type
# Ordered by: year
  primary_type         year count
  <chr>               <int> <int>
1 GAMBLING             2001    71
2 DECEPTIVE PRACTICE   2001   346
3 BATTERY              2001  5938
4 PROSTITUTION         2001   424
5 THEFT                2001  3098
6 CRIM SEXUAL ASSAULT  2001   101
```

Close the connection.

```
dbDisconnect(con)
```