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.

Repository: https://github.com/sarahglidden97/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.

```r
project <- "surv-727-assignment-4"
```

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: surv-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 'sarahglidden97@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.

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

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.

```sql
SELECT primary_type AS primary_count, count(*) AS overall_count
FROM crime
WHERE year = 2016
GROUP BY primary_type
ORDER BY COUNT(*) DESC;
```

Table 2: Displaying records 1 - 10

primary_count	overall_count
THEFT	61626
BATTERY	50300
CRIMINAL DAMAGE	31018
DECEPTIVE PRACTICE	19433
ASSAULT	18743
OTHER OFFENSE	17309
BURGLARY	14289
NARCOTICS	13333
ROBBERY	11960
MOTOR VEHICLE THEFT	11285

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?

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

Table 3: Displaying records 1 - 10

date_count	overall_count
12	15852
18	15541
19	15097
20	14498
17	14341
15	14333
16	14247
0	13711
21	13482
14	13439

12PM is the time of day 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.

```sql
SELECT year, count(*) AS overall_count
FROM crime
WHERE primary_type = "HOMICIDE"
GROUP BY year
ORDER BY COUNT(*) DESC;
```

Table 4: Displaying records 1 - 10

year	overall_count
2021	813
2020	796
2016	790

year	overall_count
2022	737
2017	676
2001	667
2002	658
2023	638
2003	604
2018	600

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.

```sql
SELECT district,
      COUNTIF(year = 2015) AS count_arrests_2015,
      COUNTIF(year = 2016) AS count_arrests_2016
FROM crime
GROUP BY district
ORDER BY district
```

Table 5: Displaying records 1 - 10

district	count_arrests_2015	count_arrests_2016
NA	0	0
1	12005	13744
2	10780	11464
3	13090	12368
4	15893	15039
5	11371	11656
6	16085	16238
7	15779	14243
8	17351	17576
9	12766	12695

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_1 <- "SELECT primary_type, count(*) AS overall_count FROM crime WHERE district = 11 AN
```

Execute the query.

```
dbGetQuery(con, query_1)
```

```
# A tibble: 30 x 2
   primary_type        overall_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            612
10 PROSTITUTION                  511
# 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_tbl <- tbl(con, "crime")
```

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

```
crime_tbl %>%
  filter(district == 11 & year == 2016) %>%
  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    612
10 PROSTITUTION          511
# i more rows
```

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

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

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

```
# Source:     SQL [?? x 3]
# Database:   BigQueryConnection
# Groups:     year
# Ordered by: year
    year primary_type                     count
   <int> <chr>                            <int>
 1  2001 MOTOR VEHICLE THEFT               1183
 2  2001 BURGLARY                           866
 3  2001 LIQUOR LAW VIOLATION                49
 4  2001 BATTERY                           5938
 5  2001 THEFT                             3098
 6  2001 INTERFERENCE WITH PUBLIC OFFICER    17
 7  2001 HOMICIDE                            72
 8  2001 SEX OFFENSE                         67
 9  2001 KIDNAPPING                          36
10  2001 OTHER OFFENSE                     1150
# i more rows
```

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

```r
query_2 <- crime_tbl %>%
  filter(district == 11) %>%
  group_by(year, primary_type) %>%
  summarise(count = n()) %>%
  arrange(year) %>%
  collect
```

```
`summarise()` has grouped output by "year". 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
head(query_2, n = 10)
```

```
# A tibble: 10 x 3
# Groups:   year [1]
    year primary_type               count
   <int> <chr>                      <int>
 1  2001 PROSTITUTION                 424
 2  2001 ROBBERY                     1243
 3  2001 BURGLARY                     866
 4  2001 OFFENSE INVOLVING CHILDREN   140
 5  2001 KIDNAPPING                    36
 6  2001 HOMICIDE                      72
 7  2001 ARSON                         47
 8  2001 LIQUOR LAW VIOLATION          49
 9  2001 CRIMINAL SEXUAL ASSAULT        2
10  2001 OTHER OFFENSE               1150
```

Close the connection.

```r
dbDisconnect(con)
```