Assignment 4

Due at 11:59pm on November 7.

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 the repository:** https://github.com/annechang11/surv727-hw4

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-test-403119"
```

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-test-403119
```

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 'hsinwenc@umich.edu'.

[1] "crime"
```

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

https://console.cloud.google.com/marketplace/product/city-of-chicago-public-data/chicago-crime?project=surv-727-test-403119

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 (*)
FROM crime
```

```
WHERE year = 2016;
```

Table 1: 1 records

f0_
269840

- There are 269839 (or 269840) rows in the year 2016 in the “crime” table.

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 arrest = TRUE AND year = 2016
GROUP BY primary_type
ORDER BY count(*) DESC;
```

Table 2: Displaying records 1 - 10

primary_type	f0_
NARCOTICS	13327
BATTERY	10332
THEFT	6522
CRIMINAL TRESPASS	3724
ASSAULT	3492
OTHER OFFENSE	3415
WEAPONS VIOLATION	2511
CRIMINAL DAMAGE	1669
PUBLIC PEACE VIOLATION	1116
MOTOR VEHICLE THEFT	1097

- Please see the number of arrests in 2016 by primary type above.

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, count(*)
FROM crime
WHERE arrest = TRUE AND year = 2016
GROUP BY hour
ORDER BY count(*) DESC;
```

Table 3: Displaying records 1 - 10

hour	f0_
10	5306
11	5200
12	4941
7	4900
8	4735
9	4675
1	4288
6	4261
2	4029
3	3750

- From the list above, we can see that hour 10 has the highest number of arrests (5306), following by hour 11 (5200). Note that we can't distinguish AM and PM from the data though.

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 arrest = TRUE AND primary_type = 'HOMICIDE'
GROUP BY year
ORDER BY count(*) DESC;
```

Table 4: Displaying records 1 - 10

year	f0_
2001	430
2002	423
2003	379
2020	339
2004	293

year	f0_
2008	286
2016	286
2006	281
2005	281
2021	275

- The list above shows the numbers of "Homicide" arrests in each year. Year 2001 has the highest number of arrests for homicide from year 2001 to 2022.

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 arrest = TRUE AND year IN (2015, 2016)
GROUP BY year, district
ORDER BY count(*) DESC;
```

Table 5: Displaying records 1 - 10

year	district	f0_
2015	11	8974
2016	11	6575
2015	7	5549
2015	15	4514
2015	6	4473
2015	25	4448
2015	4	4325
2015	8	4112
2016	7	3654
2015	10	3621

- District 11 has the highest number of arrests in both 2015 and 2016.

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.

```
SELECT primary_type, count(*)
FROM crime
WHERE arrest = TRUE AND year = 2016 AND district = 11
GROUP BY primary_type
ORDER BY count(*) DESC;
```

Table 6: Displaying records 1 - 10

primary_type	f0_
NARCOTICS	3634
BATTERY	635
PROSTITUTION	511
WEAPONS VIOLATION	303
OTHER OFFENSE	255
ASSAULT	206
CRIMINAL TRESPASS	205
PUBLIC PEACE VIOLATION	135
INTERFERENCE WITH PUBLIC OFFICER	119
CRIMINAL DAMAGE	106

```
# store the above query in a R object

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

# run the query
dbGetQuery(con, query)
```

```
# A tibble: 27 x 2
   primary_type                      f0_
   <chr>                           <int>
 1 NARCOTICS                        3634
 2 BATTERY                           635
 3 PROSTITUTION                      511
 4 WEAPONS VIOLATION                 303
 5 OTHER OFFENSE                       255
 6 ASSAULT                             206
 7 CRIMINAL TRESPASS                   205
 8 PUBLIC PEACE VIOLATION              135
 9 INTERFERENCE WITH PUBLIC OFFICER    119
10 CRIMINAL DAMAGE                     106
# i 17 more rows
```

```
subtable <- dbGetQuery(con, query)
str(subtable) # a tibble of 27*2
```

```
tibble [27 x 2] (S3: tbl_df/tbl/data.frame)
 $ primary_type: chr [1:27] "NARCOTICS" "BATTERY" "PROSTITUTION" "WEAPONS VIOLATION" ...
 $ f0_         : int [1:27] 3634 635 511 303 255 206 205 135 119 106 ...
```

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')
```

```
Warning: <BigQueryConnection> uses an old dbplyr interface
i Please install a newer version of the package or contact the maintainer
This warning is displayed once every 8 hours.
```

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 & arrest == TRUE) %>%
  group_by(primary_type) %>%
  summarise(total = n())
```

```
# Source:   SQL [?? x 2]
# Database: BigQueryConnection
   primary_type          total
   <chr>                 <int>
 1 DECEPTIVE PRACTICE       63
 2 HOMICIDE                 28
 3 ARSON                      2
 4 PUBLIC INDECENCY           2
 5 WEAPONS VIOLATION        303
 6 OTHER OFFENSE            255
 7 PROSTITUTION             511
 8 PUBLIC PEACE VIOLATION   135
 9 THEFT                     98
10 MOTOR VEHICLE THEFT       98
# i more rows
```

```
# matches the sql result
```

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 & arrest == TRUE) %>%
  group_by(year, primary_type) %>%
  summarise(total = n()) %>%
  arrange(year, primary_type)
```

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

```
# Source:     SQL [?? x 3]
# Database:   BigQueryConnection
# Groups:     year
# Ordered by: year, primary_type
    year primary_type         total
   <int> <chr>                <int>
 1  2001 ARSON                   12
 2  2001 ASSAULT                322
 3  2001 BATTERY                962
 4  2001 BURGLARY                42
 5  2001 CRIM SEXUAL ASSAULT     17
 6  2001 CRIMINAL DAMAGE        163
 7  2001 CRIMINAL TRESPASS      389
 8  2001 DECEPTIVE PRACTICE      84
 9  2001 GAMBLING                71
10  2001 HOMICIDE                48
# i more rows
# note: I arranged the result by year from 2001 to 2022
# and then by primary_type alphebetically
```

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

```
sql_data <-
  crime %>%
  filter(district == 11 & arrest == TRUE) %>%
  group_by(year, primary_type) %>%
  summarise(total = n()) %>%
  arrange(year, primary_type) %>%
  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.

```
head(sql_data, 10)
```

```
# A tibble: 10 x 3
# Groups:   year [1]
    year primary_type        total
   <int> <chr>               <int>
 1  2001 ARSON                  12
 2  2001 ASSAULT               322
 3  2001 BATTERY               962
 4  2001 BURGLARY               42
 5  2001 CRIM SEXUAL ASSAULT    17
 6  2001 CRIMINAL DAMAGE       163
 7  2001 CRIMINAL TRESPASS     389
 8  2001 DECEPTIVE PRACTICE     84
 9  2001 GAMBLING               71
10  2001 HOMICIDE               48
```

Close the connection.

```
dbDisconnect(con)
```