# 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.

GitHub Link: https://github.com/Zackwin73/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 <- "surv-727-475919"
```

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
```

```
<BigQueryConnection>
  Dataset: bigquery-public-data.chicago_crime
  Billing: surv-727-475919
```

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 'winogradza@gmail.com'.

Auto-refreshing stale OAuth token.

[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 |
|---:|---:|
| 264873 | 264873 |

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,
  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?

```sql
SELECT
  EXTRACT(HOUR FROM date) AS hour,
  COUNT(*) AS num_arrests
```

```
FROM crime
WHERE year = 2016
  AND arrest = TRUE
GROUP BY hour
ORDER BY hour ASC
```

Table 3: Displaying records 1 - 10

| hour | num_arrests |
|---|---:|
| 0 | 2174 |
| 1 | 1556 |
| 2 | 1256 |
| 3 | 980 |
| 4 | 677 |
| 5 | 517 |
| 6 | 781 |
| 7 | 1059 |
| 8 | 1433 |
| 9 | 1714 |

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_arrests
FROM crime
WHERE primary_type = 'HOMICIDE'
  AND arrest = TRUE
GROUP BY year
ORDER BY year DESC
```

Table 4: Displaying records 1 - 10

| year | num_arrests |
|---|---:|
| 2025 | 100 |
| 2024 | 213 |
| 2023 | 259 |
| 2022 | 321 |

| year | num_arrests |
|---|---|
| 2021 | 296 |
| 2020 | 356 |
| 2019 | 200 |
| 2018 | 255 |
| 2017 | 231 |
| 2016 | 292 |

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
  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.

Execute the query.

```r
library(DBI)
library(dbplyr)
library(bigrquery)

# Create the SQL query as a string
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
"

# Run the query and save results
arrests_d11_2016 <- dbGetQuery(con, query)

# Display top rows
head(arrests_d11_2016)
```

```
# A tibble: 6 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
```

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.

```r
# Map the 'crime' table from BigQuery to a lazy tibble
crime_tbl <- tbl(con, "crime")

# Use dplyr/dbplyr syntax to build the same query
arrests_d11_2016_dbplyr <- crime_tbl %>%
  filter(year == 2016,
         district == 11,
         arrest == TRUE) %>%
  group_by(primary_type) %>%
  summarise(num_arrests = n()) %>%
  arrange(desc(num_arrests))

# Display the first few rows
arrests_d11_2016_dbplyr %>% head()
```

```
# Source:     SQL [6 x 2]
# Database:   BigQueryConnection
# Ordered by: desc(num_arrests)
  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
```

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

```
library(dplyr)

crime_local <- crime_tbl %>%
  filter(year == 2016, district == 11) %>%
  collect()


arrests_d11_2016_dplyr <- crime_local %>%
  filter(year == 2016,
         district == 11,
         arrest == TRUE) %>%
  group_by(primary_type) %>%
  summarise(num_arrests = n()) %>%
  arrange(desc(num_arrests))

head(arrests_d11_2016_dplyr)
```

```
# A tibble: 6 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
```

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

```
arrests_d11_yearly_query <- crime_tbl %>%
  filter(district == 11, arrest == TRUE) %>%
  group_by(primary_type, year) %>%
  summarise(num_arrests = n()) %>%
  arrange(year)
```

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

```
arrests_d11_yearly <- arrests_d11_yearly_query %>% collect()
```

```
`summarise()` has grouped output by "primary_type". 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.

```
# Display the first ten rows of the locally saved data set
head(arrests_d11_yearly, 10)
```

```
# A tibble: 10 x 3
# Groups:   primary_type [10]
   primary_type                       year num_arrests
   <chr>                             <int>       <int>
 1 CRIMINAL DAMAGE                    2001         163
 2 DECEPTIVE PRACTICE                 2001          84
 3 INTERFERENCE WITH PUBLIC OFFICER   2001          14
 4 PROSTITUTION                       2001         424
 5 THEFT                              2001         419
 6 WEAPONS VIOLATION                  2001         236
 7 ASSAULT                            2001         322
 8 STALKING                           2001           1
 9 NARCOTICS                          2001        7979
10 LIQUOR LAW VIOLATION               2001          49
```

Close the connection.

```
dbDisconnect(con)
```