Assignment 4

Due at 11:59pm on November 5.

Yujing Jiang

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 <- "sinuous-pact-439518-i2"
```

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: sinuous-pact-439518-i2
```

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
  'jiangyujing0817@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_
269922	269922

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 total_arrests
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY primary_type
ORDER BY total_arrests DESC;
```

Table 2: Displaying records 1 - 10

primary_type	total_arrests
NARCOTICS	13327
BATTERY	10333
THEFT	6522
CRIMINAL TRESPASS	3724
ASSAULT	3492
OTHER OFFENSE	3415
WEAPONS VIOLATION	2511
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 arrest_hour, COUNT(*) AS arrests
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY arrest_hour
ORDER BY arrests DESC;
```

Table 3: Displaying records 1 - 10

arrest_hour	arrests
19	3843
18	3481
20	3302
21	2961
16	2933
22	2896
11	2895
17	2820
12	2787
14	2774

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 arrests
FROM crime
WHERE primary_type = "HOMICIDE" AND arrest = TRUE
GROUP BY year
ORDER BY arrests DESC;
```

Table 4: Displaying records 1 - 10

year	arrests
2001	430
2002	427
2003	382
2020	349
2022	306
2004	294
2021	292
2016	289
2008	287
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 district, year, COUNT(*) AS arrests
FROM crime
WHERE year IN (2015,2016) AND arrest = TRUE
GROUP BY year, district
ORDER BY arrests DESC;
```

Table 5: Displaying records 1 - 10

district	year	arrests
11	2015	8974
11	2016	6575
7	2015	5549
15	2015	4514
6	2015	4474
25	2015	4450
4	2015	4325
8	2015	4113
7	2016	3655
10	2015	3622

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.

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

```
dbGetQuery(con, sql)
```

```
# A tibble: 32 x 2
   primary_type        arrests
   <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 22 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")
str(crime_tbl)
```

```
List of 2
 $ src       :List of 2
  ..$ con  :Formal class 'BigQueryConnection' [package "bigrquery"] with 7 slots
  .. .. ..@ project       : chr "bigquery-public-data"
  .. .. ..@ dataset       : chr "chicago_crime"
  .. .. ..@ billing       : chr "sinuous-pact-439518-i2"
  .. .. ..@ use_legacy_sql: logi FALSE
  .. .. ..@ page_size     : int 10000
  .. .. ..@ quiet         : logi NA
  .. .. ..@ bigint        : chr "integer"
  ..$ disco: NULL
  ..- attr(*, "class")= chr [1:4] "src_BigQueryConnection" "src_dbi" "src_sql" "src"
 $ lazy_query:List of 6
  ..$ x         : 'dbplyr_table_path' chr "`crime`"
  ..$ vars      : chr [1:22] "unique_key" "case_number" "date" "block" ...
  ..$ group_vars: chr(0)
  ..$ order_vars: NULL
  ..$ frame     : NULL
  ..$ is_view   : logi FALSE
  ..- attr(*, "class")= chr [1:3] "lazy_base_remote_query" "lazy_base_query" "lazy_query"
 - attr(*, "class")= chr [1:5] "tbl_BigQueryConnection" "tbl_dbi" "tbl_sql" "tbl_lazy" ...
```

```
class(crime_tbl)
```

```
[1] "tbl_BigQueryConnection" "tbl_dbi"                "tbl_sql"
[4] "tbl_lazy"               "tbl"
```

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

```
crime_tbl %>%
  summarise(total = n())
```

```
# Source:   SQL [1 x 1]
# Database: BigQueryConnection
    total
    <int>
1 8189954
```

```
crime_tbl %>%
  filter(year == 2016 & district == 11) %>%
  group_by(primary_type) %>%
  summarise(arrests = n()) %>%
  arrange(desc(arrests)) %>%
  print()
```

```
# Source:     SQL [?? x 2]
# Database:   BigQueryConnection
# Ordered by: desc(arrests)
   primary_type        arrests
   <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
```

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.

```r
arrest_numbers_year <- crime_tbl %>%
  filter(district == 11) %>%
  group_by(year, primary_type) %>%
  summarise(arrests = n(), .groups = 'drop') %>%
  arrange(year)
print(arrest_numbers_year)
```

```
# Source:     SQL [?? x 3]
# Database:   BigQueryConnection
# Ordered by: year
    year primary_type         arrests
   <int> <chr>                  <int>
 1  2001 HOMICIDE                  72
 2  2001 LIQUOR LAW VIOLATION      49
 3  2001 STALKING                   5
 4  2001 BATTERY                 5938
 5  2001 KIDNAPPING                36
 6  2001 CRIMINAL DAMAGE          2193
 7  2001 NARCOTICS                7979
 8  2001 PROSTITUTION              424
 9  2001 CRIM SEXUAL ASSAULT       101
10  2001 CRIMINAL TRESPASS         515
# i more rows
```

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

```r
head(arrest_numbers_year, 10)
```

```
# Source:     SQL [10 x 3]
# Database:   BigQueryConnection
# Ordered by: year
    year primary_type               arrests
   <int> <chr>                      <int>
 1  2001 BURGLARY                     866
 2  2001 ARSON                         47
 3  2001 MOTOR VEHICLE THEFT         1183
 4  2001 OFFENSE INVOLVING CHILDREN   140
 5  2001 THEFT                       3098
 6  2001 ROBBERY                     1243
 7  2001 OTHER OFFENSE               1150
```

```
 8  2001 SEX OFFENSE                    67
 9  2001 ASSAULT                      1667
10  2001 WEAPONS VIOLATION             316
```

Close the connection.

```
dbDisconnect(con)
```