# Assignment 4

**Due at 11:59pm on November 5.**

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 repo link: https://github.com/TKwapong/sql-assgn

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 <- "secret-code-439518-p2"
```

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: secret-code-439518-p2
```

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)
```

```
[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_cnt, count(*) AS overall_cnt
-- as helps rename
FROM crime
WHERE year = 2016
LIMIT 10;
--; colon sometimes not needed but conventionally used
```

Table 1: 1 records

| primary_cnt | overall_cnt |
|---|---|
| 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.

```sql
SELECT primary_type, count(arrest) AS arrests_cnt
-- as helps rename
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY primary_type
ORDER BY COUNT(*) DESC
LIMIT 20;
```

Table 2: Displaying records 1 - 10

| primary_type | arrests_cnt |
|---|---|
| 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?

```sql
SELECT EXTRACT(HOUR FROM date) AS hour , count(arrest) AS Arrest_cnts
FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY EXTRACT(HOUR FROM date)
ORDER BY COUNT(*) DESC
LIMIT 10;
```

Table 3: Displaying records 1 - 10

| hour | Arrest_cnts |
|---|---|
| 19 | 3843 |
| 18 | 3481 |
| 20 | 3302 |

| hour | Arrest_cnts |
|---|---|
| 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(arrest) AS HOMICIDE_cnts
FROM crime
WHERE primary_type = 'HOMICIDE' AND arrest = TRUE
GROUP BY year
ORDER BY COUNT(*) DESC;
```

Table 4: Displaying records 1 - 10

| year | HOMICIDE_cnts |
|---|---|
| 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  year, district, count(arrest) AS arrest_cnts
FROM crime
WHERE year BETWEEN 2015 AND 2016 AND arrest = TRUE
```

```
GROUP BY year, district
ORDER BY COUNT(*) DESC
LIMIT 10;
```

Table 5: Displaying records 1 - 10

| year | district | arrest_cnts |
|---|---|---|
| 2015 | 11 | 8974 |
| 2016 | 11 | 6575 |
| 2015 | 7 | 5549 |
| 2015 | 15 | 4514 |
| 2015 | 6 | 4474 |
| 2015 | 25 | 4450 |
| 2015 | 4 | 4325 |
| 2015 | 8 | 4113 |
| 2016 | 7 | 3655 |
| 2015 | 10 | 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(arrest) AS arrests_cnt
        FROM crime
        WHERE year = 2016 AND district = 11 AND arrest = TRUE
        GROUP BY primary_type
        ORDER BY COUNT(*) DESC
        LIMIT 20;"
dbGetQuery(con, sql)
```

```
# A tibble: 20 x 2
   primary_type                arrests_cnt
   <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
11 MOTOR VEHICLE THEFT                     98
12 THEFT                                   98
13 DECEPTIVE PRACTICE                      63
14 ROBBERY                                 56
15 GAMBLING                                32
16 HOMICIDE                                28
17 OFFENSE INVOLVING CHILDREN              25
18 BURGLARY                                22
19 LIQUOR LAW VIOLATION                    11
20 CRIM SEXUAL ASSAULT                     10
```

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
crime_tibble <- tbl(con, "crime")
str(crime_tibble)
```

```
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 "secret-code-439518-p2"
  .. .. ..@ 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" ...
```

```
crime_tibble %>%
  select(primary_type, year, arrest, district) %>%
  group_by(primary_type) %>%
  filter(year == 2016 & district == 11  & arrest == TRUE) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  head(20)
```

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

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

```
crime_tibble %>%
  select(primary_type, year, arrest, district) %>%
  group_by(primary_type) %>%
  filter(year == 2016 & district == 11 & arrest == TRUE) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  head(20)
```

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

```
   primary_type                     count
   <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 more rows
```

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

```
crime_tibble %>%
  group_by(primary_type, year) %>%
  filter(district == 11 & arrest == TRUE) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>%
  head(10)
```

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

```
# Source:     SQL [10 x 3]
# Database:   BigQueryConnection
# Groups:     primary_type
# Ordered by: desc(count)
   primary_type  year count
   <chr>        <int> <int>
 1 NARCOTICS     2005  9718
 2 NARCOTICS     2003  9562
 3 NARCOTICS     2002  9232
 4 NARCOTICS     2004  9083
 5 NARCOTICS     2006  8185
 6 NARCOTICS     2001  7979
 7 NARCOTICS     2007  7395
 8 NARCOTICS     2013  7234
```

```
 9 NARCOTICS     2014  6801
10 NARCOTICS     2009  5942
```

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

```
arrests_type <- crime_tibble %>%
  group_by(primary_type, year) %>%
  filter(district == 11 & arrest == TRUE) %>%
  summarize(count = n()) %>%
  arrange(desc(count))  %>%
  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.

```
arrests_type %>%
  head(10)
```

```
# A tibble: 10 x 3
# Groups:   primary_type [1]
   primary_type  year count
   <chr>        <int> <int>
 1 NARCOTICS     2005  9718
 2 NARCOTICS     2003  9562
 3 NARCOTICS     2002  9232
 4 NARCOTICS     2004  9083
 5 NARCOTICS     2006  8185
 6 NARCOTICS     2001  7979
 7 NARCOTICS     2007  7395
 8 NARCOTICS     2013  7234
 9 NARCOTICS     2014  6801
10 NARCOTICS     2009  5942
```

Close the connection.

```
dbDisconnect(con)
```