# assignment4

Leng Seong Che

2023-11-07

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 <- "chicagocrime-403117"
```

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: chicagocrime-403117
```

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 'hamera0615@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(*) FROM crime
WHERE year = 2016;
```

Table 1: 1 records

| f0_ |
|---|
| 269841 |

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

Table 2: Displaying records 1 - 10

| primary_type | arrests |
|---|---|
| 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 |

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 arrests FROM crime
WHERE year = 2016 AND arrest = TRUE
GROUP BY hour
ORDER BY arrests DESC
LIMIT 1;
```

Table 3: 1 records

| hour | arrests |
|---|---|
| 10 | 5306 |

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 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 | 423 |
| 2003 | 379 |
| 2020 | 339 |
| 2004 | 293 |
| 2008 | 286 |
| 2016 | 286 |
| 2006 | 281 |
| 2005 | 281 |
| 2021 | 275 |

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 arrests FROM crime
WHERE (year = 2015 OR year = 2016) AND arrest = TRUE
GROUP BY year, district
ORDER BY arrests DESC;
```

Table 5: Displaying records 1 - 10

| year | district | arrests |
|---|---|---|
| 2015 | 11 | 8974 |
| 2016 | 11 | 6575 |
| 2015 | 7 | 5549 |
| 2015 | 15 | 4514 |

| year | district | arrests |
|---|---|---|
| 2015 | 6 | 4473 |
| 2015 | 25 | 4448 |
| 2015 | 4 | 4325 |
| 2015 | 8 | 4112 |
| 2016 | 7 | 3654 |
| 2015 | 10 | 3621 |

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

Execute the query.

```
dbGetQuery(con, query)
```

```
## # A tibble: 27 x 2
##    primary_type                     arrests
##    <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
```

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_query <- "SELECT * FROM crime"
crime <- tbl(con, sql(crime_query))
```

```
## 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, arrest == TRUE, district == 11) %>%
  group_by(primary_type) %>%
  summarise(arrests = n()) %>%
  arrange(desc(arrests))
```

```
## # Source:     SQL [?? x 2]
## # Database:   BigQueryConnection
## # Ordered by: desc(arrests)
##    primary_type                      arrests
##    <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 %>%
  filter(arrest == TRUE & district == 11) %>%
  group_by(primary_type, year) %>%
  summarise(arrests = n()) %>%
  arrange(year)
```

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

```
## # Source:     SQL [?? x 3]
## # Database:   BigQueryConnection
## # Groups:     primary_type
## # Ordered by: year
##    primary_type                year arrests
##    <chr>                      <int>   <int>
##  1 OFFENSE INVOLVING CHILDREN  2001      44
##  2 PROSTITUTION                2001     424
##  3 DECEPTIVE PRACTICE          2001      84
##  4 GAMBLING                    2001      71
##  5 BATTERY                     2001     962
##  6 INTIMIDATION                2001       3
##  7 SEX OFFENSE                 2001      19
##  8 CRIMINAL DAMAGE             2001     163
##  9 CRIM SEXUAL ASSAULT         2001      17
## 10 THEFT                       2001     419
## # i more rows
```

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

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

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

```
head(crime_arrest,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: year
##    primary_type                year arrests
##    <chr>                      <int>   <int>
##  1 DECEPTIVE PRACTICE          2001      84
##  2 THEFT                       2001     419
##  3 CRIM SEXUAL ASSAULT         2001      17
##  4 GAMBLING                    2001      71
##  5 INTIMIDATION                2001       3
##  6 ASSAULT                     2001     322
##  7 OFFENSE INVOLVING CHILDREN  2001      44
##  8 OTHER OFFENSE               2001     266
##  9 PROSTITUTION                2001     424
## 10 BATTERY                     2001     962
```

Close the connection.

```
dbDisconnect(con)
```