Assignment 4

Kangrui Liu

2023-10-31

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/krliu67/Assignment_SURV727/tree/main/a4

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-test-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: surv-727-test-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 'kangruiliusz@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(*) count
FROM crime
WHERE year = 2016;
```

Table 1: 1 records

count
269840

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(*) count_pt FROM crime WHERE year = 2016 group by primary_type
ORDER BY count_pt DESC LIMIT 10;
```

Table 2: Displaying records 1 - 10

primary_type	count_pt
THEFT	61621
BATTERY	50300
CRIMINAL DAMAGE	31018
DECEPTIVE PRACTICE	19351
ASSAULT	18742
OTHER OFFENSE	17307
BURGLARY	14289

primary_type	count_pt
NARCOTICS	13333
ROBBERY	11960
MOTOR VEHICLE THEFT	11286

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) hour, count(*) count_hour FROM crime WHERE year = 2016 group by hour
ORDER BY count_hour DESC LIMIT 10;
```

Table 3: Displaying records 1 - 10

hour	count_hour
12	29489
9	26315
10	25400
8	24201
11	23217
7	21830
1	21153
2	20466
6	20330
3	20204

- 12:00:00 - 12:59:59 has the most arrests.

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(*) count_year FROM crime WHERE primary_type = "HOMICIDE" GROUP BY year
ORDER BY count_year DESC LIMIT 10;
```

Table 4: Displaying records 1 - 10

year	count_year
2021	810
2020	796
2016	790
2022	726
2017	676
2001	667
2002	658
2003	604
2018	601
2012	515

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(*) count_district FROM crime WHERE 2015<=year and year <= 2016
GROUP BY district,year ORDER BY count_district DESC LIMIT 10;
```

Table 5: Displaying records 1 - 10

year	district	count_district
2015	11	19535
2016	11	18654
2016	8	17569
2015	8	17345
2016	6	16226
2015	6	16072
2015	4	15887
2015	7	15777
2015	25	15081
2016	4	15030

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 year, district, primary_type, count(*) count_pt FROM crime WHERE year = 2016 and
district = 11 GROUP BY primary_type,year,district ORDER BY count_pt DESC LIMIT 10;"
```

Execute the query.

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

```
## # A tibble: 10 x 4
##     year district primary_type        count_pt
##    <int>    <int> <chr>                  <int>
##  1  2016       11 BATTERY                 3906
##  2  2016       11 NARCOTICS               3635
##  3  2016       11 THEFT                   2043
##  4  2016       11 CRIMINAL DAMAGE         1775
##  5  2016       11 ASSAULT                 1330
##  6  2016       11 OTHER OFFENSE           1045
##  7  2016       11 ROBBERY                 1007
##  8  2016       11 MOTOR VEHICLE THEFT      776
##  9  2016       11 DECEPTIVE PRACTICE       609
## 10  2016       11 PROSTITUTION             511
```

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(district == 11, year == 2016)%>% group_by(primary_type) %>% count(primary_type)
```

```
## # Source:   SQL [?? x 2]
## # Database: BigQueryConnection
## # Groups:   primary_type
##    primary_type                         n
##    <chr>                            <int>
##  1 DECEPTIVE PRACTICE                 609
##  2 HOMICIDE                            95
##  3 CRIM SEXUAL ASSAULT                101
##  4 NARCOTICS                         3635
##  5 ROBBERY                           1007
##  6 SEX OFFENSE                         45
##  7 INTERFERENCE WITH PUBLIC OFFICER   129
##  8 INTIMIDATION                         8
##  9 ARSON                               28
## 10 PUBLIC INDECENCY                     2
## # 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(district == 11) %>% group_by(year, primary_type) %>%
  summarize(arrest_count = n()) %>% arrange(year)
```

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

```
## # Source:     SQL [?? x 3]
## # Database:   BigQueryConnection
## # Groups:     year
## # Ordered by: year
##     year primary_type                     arrest_count
##    <int> <chr>                                   <int>
##  1  2001 WEAPONS VIOLATION                         316
##  2  2001 ASSAULT                                  1667
##  3  2001 BATTERY                                  5938
##  4  2001 PUBLIC PEACE VIOLATION                    128
##  5  2001 THEFT                                    3098
##  6  2001 INTERFERENCE WITH PUBLIC OFFICER           17
##  7  2001 KIDNAPPING                                36
##  8  2001 CRIMINAL DAMAGE                         2193
##  9  2001 GAMBLING                                  71
## 10  2001 INTIMIDATION                               8
## # i more rows
```

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

```
year_primary <- crime %>% filter(district == 11) %>% group_by(year, primary_type) %>%
    summarize(arrest_count = n()) %>% arrange(year) %>% data.frame()
```

```
## `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.

```
year_primary[c(1:10),]
```

```
##    year                     primary_type arrest_count
## 1  2001           PUBLIC PEACE VIOLATION          128
## 2  2001 INTERFERENCE WITH PUBLIC OFFICER           17
## 3  2001                         GAMBLING           71
## 4  2001                      INTIMIDATION            8
## 5  2001                          BATTERY         5938
## 6  2001                 CRIMINAL DAMAGE         2193
## 7  2001             LIQUOR LAW VIOLATION           49
## 8  2001                CRIMINAL TRESPASS          515
## 9  2001                         HOMICIDE           72
## 10 2001                       KIDNAPPING           36
```

Close the connection.

```
dbDisconnect(con)
```