MDSR2e Ch 15: Database querying using SQL

Goals

- To

Required reading

- Chapter 5 of your textbook

New Code

- `mosaic::favstats(dataset$var)`, provides summary statistics for variable `var` from `dataset`

Before class

Let's start by loading a subset of data used for the story by doing the following command in R

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(mdsr)
library(dbplyr)
```

```
##
## Attaching package: 'dbplyr'
##
## The following objects are masked from 'package:dplyr':
##
##     ident, sql
```

```
library(DBI)

#| connect to the scidb server on Amazon Web Services
db <- dbConnect_scidb("airlines")
flights <- tbl(db, "flights")
carriers <- tbl(db, "carriers")

#| Section 20.1: From dplyr to SQL

q <- flights %>%
  filter(
    year == 2016 & month == 9,
    dest == "JFK"
  ) %>%
  inner_join(carriers, by = c("carrier" = "carrier")) %>%
  group_by(name) %>%
  summarize(
    N = n(),
    pct_ontime = sum(arr_delay <= 15) / n()
  ) %>%
  filter(N >= 100) %>%
  arrange(desc(pct_ontime))
head(q, 4)
```

```
## Warning: Missing values are always removed in SQL aggregation functions.
## Use `na.rm = TRUE` to silence this warning
## This warning is displayed once every 8 hours.
```

```
## # Source:     SQL [4 x 3]
## # Database:   mysql  [mdsr_public@mdsr.cdc7tgkkqd0n.us-east-1.rds.amazonaws.com:NA/airlines]
## # Ordered by: desc(pct_ontime)
##   name                        N pct_ontime
##   <chr>                 <int64>      <dbl>
## 1 Delta Air Lines Inc.      2396      0.869
## 2 Virgin America             347      0.833
## 3 JetBlue Airways           3463      0.817
## 4 American Airlines Inc.    1397      0.782
```

```
class(flights)
```

```
## [1] "tbl_MariaDBConnection" "tbl_dbi"               "tbl_sql"
## [4] "tbl_lazy"              "tbl"
```

```
#| for a MySQL database, dplyr translates pipeline to SQL
show_query(q)
```

```
## <SQL>
## SELECT
##   `name`,
##   COUNT(*) AS `N`,
##   SUM(`arr_delay` <= 15.0) / COUNT(*) AS `pct_ontime`
```

```
## FROM (
##   SELECT `LHS`.*, `name`
##   FROM (
##     SELECT *
##     FROM `flights`
##     WHERE (`year` = 2016.0 AND `month` = 9.0) AND (`dest` = 'JFK')
##   ) `LHS`
##   INNER JOIN `carriers`
##     ON (`LHS`.`carrier` = `carriers`.`carrier`)
## ) `q01`
## GROUP BY `name`
## HAVING (COUNT(*) >= 100.0)
## ORDER BY `pct_ontime` DESC
```

If we were to write the query, it'd be a bit more readable.

However, there is no chunk preview option with SQL code, so you have to (a) knit the document to check that it works, or (b) select "chunk output inline" under the settings button to get "run chunk" button in the sql chunk. The code below will show sql results if you leave off output.var, otherwise you need to actually print mydataframe in an R chunk.

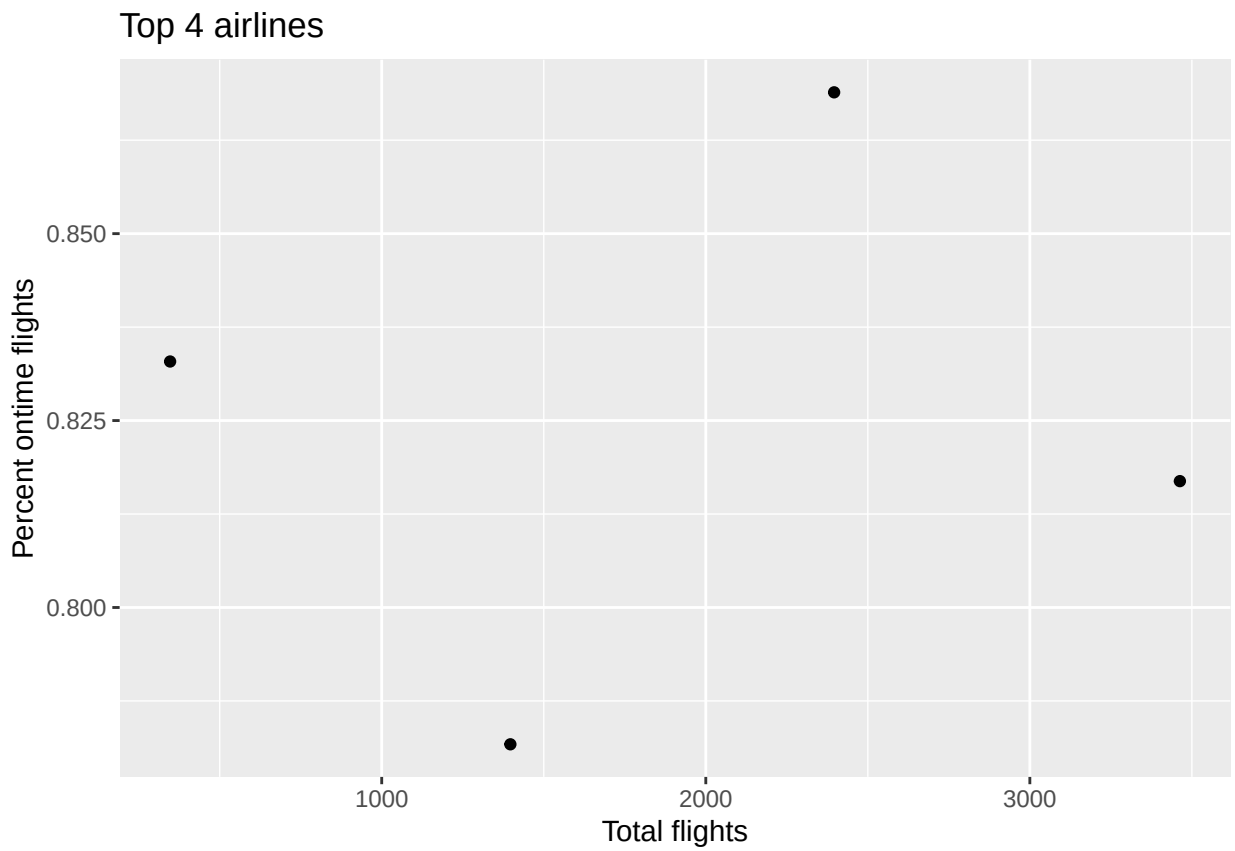
This post is helpful: https://irene.rbind.io/post/using-sql-in-rstudio/

```
SELECT
  c.name,
  SUM(1) AS N,
  SUM(arr_delay <= 15) / SUM(1) AS pct_ontime
FROM flights AS f
JOIN carriers AS c ON f.carrier = c.carrier
WHERE year = 2016 AND month = 9
  AND dest = 'JFK'
GROUP BY name
HAVING N >= 100
ORDER BY pct_ontime DESC
LIMIT 0,4;
```

```
mydataframe
```

```
##                      name    N pct_ontime
## 1   Delta Air Lines Inc. 2396     0.8689
## 2         Virgin America  347     0.8329
## 3        JetBlue Airways 3463     0.8169
## 4 American Airlines Inc. 1397     0.7817
```

```
ggplot(mydataframe, aes(x = N, y = pct_ontime)) +
  geom_point() +
  xlab("Total flights") +
  ylab("Percent ontime flights") +
  ggtitle("Top 4 airlines")
```



An alternative if the SQL code is stored elsewhere:

```
-- !preview conn=dbConnect_scidb("airlines")

SELECT
  c.name,
  SUM(1) AS N,
  SUM(arr_delay <= 15) / SUM(1) AS pct_ontime
FROM flights AS f
JOIN carriers AS c ON f.carrier = c.carrier
WHERE year = 2016 AND month = 9
  AND dest = 'JFK'
GROUP BY name
HAVING N >= 100
ORDER BY pct_ontime DESC
LIMIT 0,4;
```

```
test
```

```
##                     name    N pct_ontime
## 1   Delta Air Lines Inc. 2396     0.8689
## 2         Virgin America  347     0.8329
## 3        JetBlue Airways 3463     0.8169
## 4 American Airlines Inc. 1397     0.7817
```

Can also insert SQL code inside dbGetQuery():

```
test2 <- dbGetQuery(db,'
SELECT
  c.name,
  SUM(1) AS N,
  SUM(arr_delay <= 15) / SUM(1) AS pct_ontime
FROM flights AS f
JOIN carriers AS c ON f.carrier = c.carrier
WHERE year = 2016 AND month = 9
  AND dest = \'JFK\'
GROUP BY name
HAVING N >= 100
ORDER BY pct_ontime DESC
LIMIT 0,4;
')

test2
```

```
##                      name    N pct_ontime
## 1   Delta Air Lines Inc. 2396     0.8689
## 2         Virgin America  347     0.8329
## 3        JetBlue Airways 3463     0.8169
## 4 American Airlines Inc. 1397     0.7817
```

```
translate_sql(mean(arr_delay, na.rm = TRUE))

#| translation does not always work, but can pass unknown functions through
my_paste <- paste0
translate_sql(my_paste("this", "is", "a", "string"))
carriers %>%
  mutate(name_code = my_paste(name, "(", carrier, ")"))

#| how to make it work - use CONCAT, the MySQL equivalent to paste0, or
#| use collect to break the MySQL connection and return a tbl_df
class(carriers)

carriers %>%
  mutate(name_code = CONCAT(name, "(", carrier, ")"))

carriers %>%
  collect() %>%
  mutate(name_code = my_paste(name, "(", carrier, ")"))

#| Section 15.2: Flat-file databases

carriers %>%
  object.size() %>%
  print(units = "Kb")  # doesn't match size in book

carriers %>%
  collect() %>%
  object.size() %>%
  print(units = "Kb")
```

```r
#| this code produces error: cannot allocate vector of size 762.9 Mb
n <- 100 * 1e6
x <- matrix(runif(n), ncol = 100)
dim(x)
print(object.size(x), units = "Mb")

#| Section 15.3: The SQL universe
```