SQL: Exercises

You can download this .qmd file from here. Just hit the Download Raw File button.

The code in 15_SQL.qmd walked us through many of the examples in MDSR Chapter 15; now, we present a set of practice exercises in converting from the tidyverse to SQL.

```
library(tidyverse)
library(mdsr)
library(dbplyr)
library(DBI)
```

```
# connect to the database which lives on a remote server maintained by
#   St. Olaf's IT department
library(RMariaDB)
con <- dbConnect(
  MariaDB(), host = "mdb.stolaf.edu",
  user = "ruser", password = "ruserpass",
  dbname = "flight_data"
)
```

On Your Own - Extended Example from MDSR

Refer to Section 15.5 in MDSR, where they attempt to replicate some of FiveThirtyEight's analyses. The MDSR authors provide a mix of SQL and R code to perform their analyses, but the code will not work if you simply cut-and-paste as-is into R. Your task is to convert the book code into something that actually runs, and then *apply it to data from 2024*. Very little of the code needs to be adjusted; it mostly needs to be repackaged.

Hints:

- use `dbGetQuery()`

- note that what they call `carrier` is just called `Reporting_Airline` in the `flightdata` table; you don't have to merge in a `carrier` table, although it's unfortunate that the `Reporting_Airline` codes are a bit cryptic

1. Below Figure 15.1, the MDSR authors first describe how to plot slowest and fastest airports. Instead of using *target time*, which has a complex definition, we will use *arrival time*, which oversimplifies the situation but gets us in the ballpark. Duplicate the equivalent of the table below for 2024 using the code in MDSR:

```
# A tibble: 30 × 3
   dest  avgDepartDelay avgArrivalDelay
   <chr>          <dbl>           <dbl>
 1 ORD            14.3            13.1
 2 MDW            12.8             7.40
 3 DEN            11.3             7.60
 4 IAD            11.3             7.45
 5 HOU            11.3             8.07
 6 DFW            10.7             9.00
 7 BWI            10.2             6.04
 8 BNA             9.47            8.94
 9 EWR             8.70            9.61
10 IAH             8.41            6.75
# 20 more rows
```

2. Following the table above, the MDSR authors mimic one more FiveThirtyEight table which ranks carriers by time added vs. typical and time added vs. target. In this case, we will find average arrival delay after controlling for the routes flown. Again, duplicate the equivalent of the table below for 2024 using the code in MDSR:

```
# A tibble: 14 × 5
   carrier carrier_name                numRoutes numFlights wAvgDelay
   <chr>   <chr>                           <int>      <dbl>     <dbl>
 1 VX      Virgin America                     72      57510   -2.69
 2 FL      AirTran Airways Corporation       170      79495   -1.55
 3 AS      Alaska Airlines Inc.              242     160257   -1.44
 4 US      US Airways Inc.                   378     414665   -1.31
 5 DL      Delta Air Lines Inc.              900     800375   -1.01
 6 UA      United Air Lines Inc.             621     493528   -0.982
 7 MQ      Envoy Air                         442     392701   -0.455
 8 AA      American Airlines Inc.            390     537697   -0.0340
 9 HA      Hawaiian Airlines Inc.             56      74732    0.272
10 OO      SkyWest Airlines Inc.            1250     613030    0.358
```

```
11 B6      JetBlue Airways                  316     249693    0.767
12 EV      ExpressJet Airlines Inc.        1534     686021    0.845
13 WN      Southwest Airlines Co.          1284    1174633    1.13
14 F9      Frontier Airlines Inc.           326      85474    2.29
```

On Your Own - Adapting 164 Code

These problems are based on class exercises from SDS 164, so you've already solved them in R! Now we're going to try to duplicate those solutions in SQL (but with 2023 data instead of 2013).

```
# Read in 2013 NYC flights data
library(nycflights13)
flights_nyc13 <- nycflights13::flights
planes_nyc13 <- nycflights13::planes
```

1. Summarize carriers flying to MSP by number of flights and proportion that are cancelled (assuming that a missing arrival time indicates a cancelled flight). [This was #4 in 17_longer_pipelines.Rmd.]

First duplicate the output above, then check trends in 2023 across all origins. Here are a few hints:

- use flightdata instead of flights_nyc13
- remember that flights_nyc13 only contained 2013 and 3 NYC origin airports (EWR, JFK, LGA)
- is.na can be replaced with CASE WHEN ArrTime IS NULL THEN 1 ELSE 0 END or with CASE WHEN cancelled = 1 THEN 1 ELSE 0 END
- CASE WHEN can also be used replace fct_collapse

```
# Original solution from SDS 164
flights_nyc13 |>
  mutate(carrier = fct_collapse(carrier, "Delta +" = c("DL", "9E"),
                                         "American +"= c("AA", "MQ"),
                                        "United +" = c("EV", "OO", "UA"))) |>
  filter(dest == "MSP") |>
  group_by(origin, carrier) |>
  summarize(n_flights = n(),
            num_cancelled = sum(is.na(arr_time)),
            prop_cancelled = mean(is.na(arr_time)))
```

```
# A tibble: 5 x 5
# Groups:   origin [3]
  origin carrier    n_flights num_cancelled prop_cancelled
  <chr>  <fct>          <int>         <int>          <dbl>
1 EWR    Delta +          598            10         0.0167
2 EWR    United +        1779           105         0.0590
3 JFK    Delta +         1095            41         0.0374
4 LGA    Delta +         2420            25         0.0103
5 LGA    American +      1293            62         0.0480
```

```sql
#checking the variable names
SELECT *
FROM flightdata
LIMIT 1, 10;
```

id	Year	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	2023	1	3	2	2023-01-03	[illegible]	Hartford, CT	[illegible]	New York, NY	New York	[illegible]	-5	0	0	-1	0800-0859	[illegible]	905	-8
3	2023	1	4	3	2023-01-04	[illegible]	Hartford, CT	[illegible]	New York, NY	New York	[illegible]	-5	0	0	-1	0800-0859	[illegible]	905	21
4	2023	1	5	4	2023-01-05	[illegible]	Hartford, CT	[illegible]	New York, NY	New York	[illegible]	-6	0	0	-1	0800-0859	[illegible]	905	-17
5	2023	1	6	5	2023-01-06	[illegible]	Hartford, CT	[illegible]	New York, NY	New York	[illegible]	-1	0	0	-1	0800-0859	[illegible]	905	-16
6	2023	1	7	6	2023-01-07	[illegible]	Hartford, CT	[illegible]	New York, NY	New York	[illegible]	-10	0	0	-1	0800-0859	[illegible]	905	-13
7	2023	1	14	6	2023-01-14	[illegible]	New York, NY	New York	[illegible]	Cincinnati, OH	[illegible]	-8	0	0	-1	1500-1559	[illegible]	1720	31
8	2023	1	21	6	2023-01-21	[illegible]	New York, NY	New York	[illegible]	Cincinnati, OH	[illegible]	-10	0	0	-1	1500-1559	[illegible]	1720	-25

id	Year	Q	M	D	D	[illegible]
9	2023	1	1	28	6	2023-01-28 [illegible] New York, NY 36 New York [illegible] Cincinnati, OH [illegible] 0 0 -1 1500-1559 [illegible] 15
10	2023	1	1	9	1	2023-01-09 [illegible] New York, NY 36 New York [illegible] NY New York [illegible] 0 0 -1 2100-2159 [illegible] 3
11	2023	1	1	10	2	2023-01-10 [illegible] New York, NY 36 New York [illegible] NY New York [illegible] 15 0 0 -1 2100-2159 [illegible]

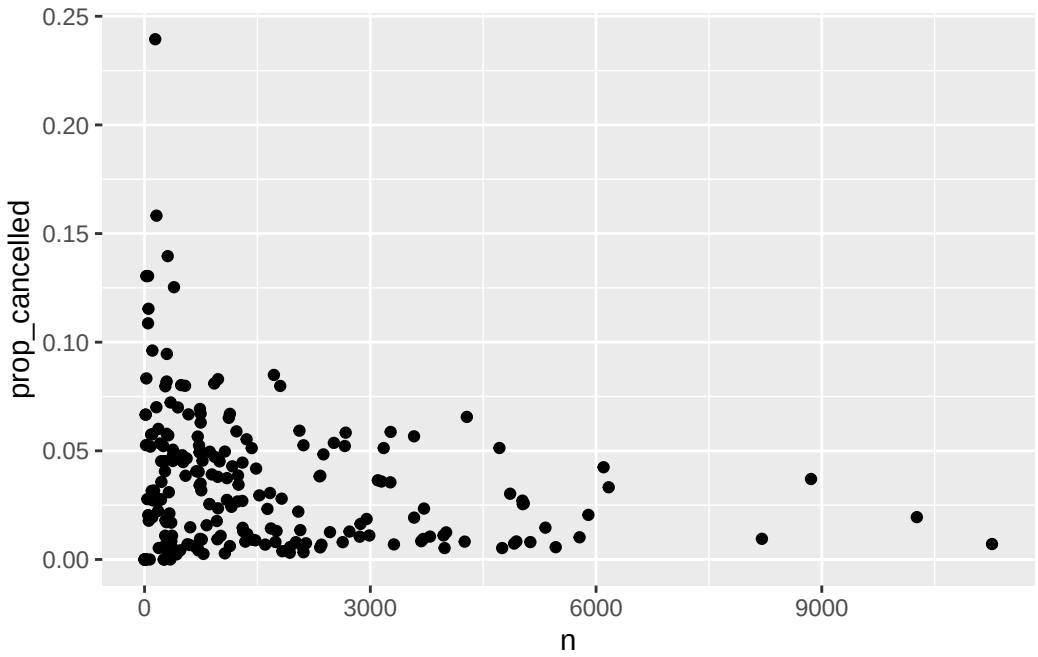
```sql
SELECT origin,
  CASE WHEN Reporting_Airline IN ("DL", "9E") THEN 'Delta +'
    WHEN Reporting_Airline IN ("AA", "MQ") THEN 'American +'
    WHEN Reporting_Airline IN ("EV", "OO", "UA") THEN 'United+'
    ELSE "Other" END AS new_carrier,
  SUM(1) AS n_flights,
  SUM(cancelled) AS num_cancelled,
  AVG(cancelled) AS prop_cancelled
FROM flightdata
WHERE dest = 'MSP'
GROUP BY origin, new_carrier
LIMIT 1, 10;
```

Table 2: Displaying records 1 - 10

origin	new_carrier	n_flights	num_cancelled	prop_cancelled
ABR	United+	4007	84	0.0210
ALB	Delta +	118	0	0.0000
ALB	United+	156	3	0.0192
ANC	Delta +	2818	13	0.0046
ANC	Other	161	2	0.0124
ASE	United+	34	3	0.0882
ATL	Delta +	18167	93	0.0051
ATL	Other	2409	43	0.0178
ATW	Delta +	2555	15	0.0059
ATW	United+	3009	71	0.0236

2. Plot number of flights vs. proportion cancelled for every origin-destination pair (assuming that a missing arrival time indicates a cancelled flight). [This was #7 in 17_longer_pipelines.Rmd.]

```r
# Original solution from SDS 164
flights_nyc13 |>
  group_by(origin, dest) |>
  summarize(n = n(),
            prop_cancelled = mean(is.na(arr_time))) |>
  filter(prop_cancelled < 1) |>
  ggplot(aes(n, prop_cancelled)) +
  geom_point()
```



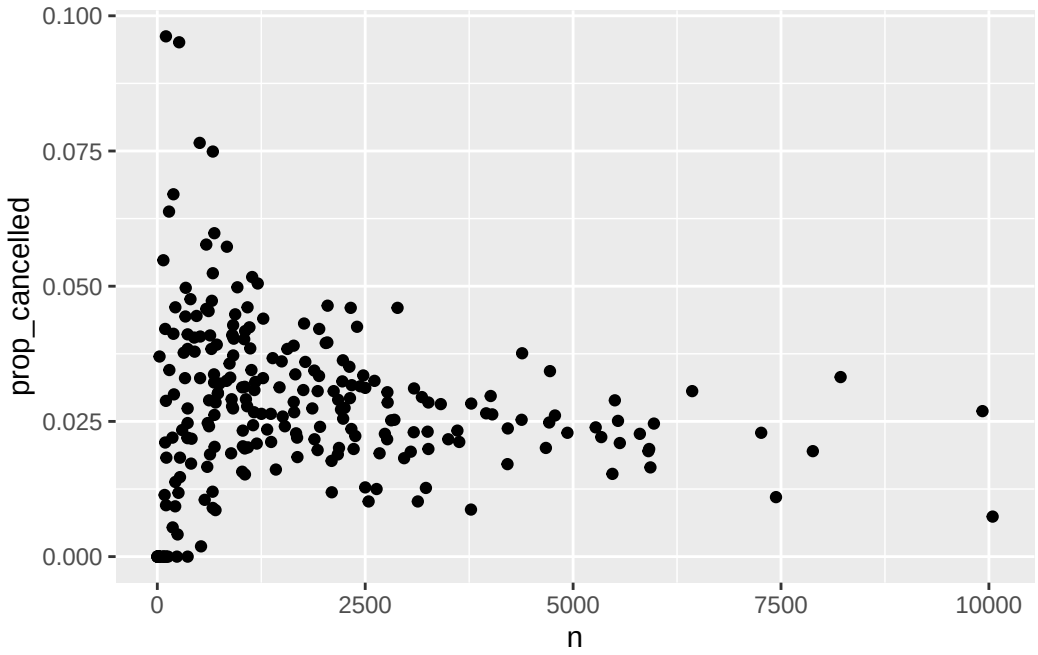
First duplicate the plot above for 2023 data, then check trends across all origins. Do all of the data wrangling in SQL. Here are a few hints:

- use flightdata instead of flights_nyc13
- remember that flights_nyc13 only contained 2013 and 3 NYC origin airports (EWR, JFK, LGA)
- use an `sql` chunk and an `r` chunk
- include `connection =` and `output.var =` in your sql chunk header (this doesn't seem to work with dbGetQuery()…)

```sql
SELECT
  origin, dest, cancelled,
```

```
  SUM(1) AS n,
  AVG(cancelled) AS prop_cancelled
FROM flightdata
WHERE Year = 2023 AND origin IN ('EWR', 'JFK', 'LGA')
GROUP BY origin, dest
HAVING prop_cancelled < 1;
```

```
flightsframe |>
  ggplot(aes(n, prop_cancelled)) +
  geom_point()
```



3. [SKIP until the planes dataset becomes available] Produce a table of weighted plane age by carrier, where weights are based on number of flights per plane. [This was #6 in 26_more_joins.Rmd.]

```
# Original solution from SDS 164
flights_nyc13 |>
  left_join(planes_nyc13, join_by(tailnum)) |>
  mutate(plane_age = 2013 - year.y) |>
  group_by(carrier) |>
  summarize(unique_planes = n_distinct(tailnum),
```

```
            mean_weighted_age = mean(plane_age, na.rm =TRUE),
            sd_weighted_age = sd(plane_age, na.rm =TRUE)) |>
  arrange(mean_weighted_age)
```

```
# A tibble: 16 x 4
   carrier unique_planes mean_weighted_age sd_weighted_age
   <chr>           <int>             <dbl>           <dbl>
 1 HA                 14              1.55            1.14
 2 AS                 84              3.34            3.07
 3 VX                 53              4.47            2.14
 4 F9                 26              4.88            3.67
 5 B6                193              6.69            3.29
 6 OO                 28              6.84            2.41
 7 9E                204              7.10            2.67
 8 US                290              9.10            4.88
 9 WN                583              9.15            4.63
10 YV                 58              9.31            1.93
11 EV                316             11.3             2.29
12 FL                129             11.4             2.16
13 UA                621             13.2             5.83
14 DL                629             16.4             5.49
15 AA                601             25.9             5.42
16 MQ                238             35.3             3.13
```

First duplicate the output above for 2023, then check trends across all origins. Do all of the data wrangling in SQL. Here are a few hints:

- use flightdata instead of flights_nyc13
- remember that flights_nyc13 only contained 2013 and 3 NYC origin airports (EWR, JFK, LGA)
- you'll have to merge the flights dataset with the planes dataset *when it becomes available*
- you can use DISTINCT inside a COUNT()
- investigate SQL clauses for calculating a standard deviation
- you cannot use a derived variable inside a summary clause in SELECT

For bonus points, also merge the airlines dataset and include the name of each carrier and not just the abbreviation!

```
SELECT *
FROM planes
LIMIT 1, 6;
```

Table 3: 6 records

tailnum	year	type	manufacturer	model	engines	seats	speed	engine
N101DU	2018	Fixed wing multi engine	C SERIES AIRCRAFT LTD PTNRSP	BD-500-1A10	2	133	0	Turbo-fan
N101HQ	2007	Fixed wing multi engine	EMBRAER-EMPRESA BRASILEIRA DE	ERJ 170-200 LR	2	80	0	Turbo-fan
N101NN	2013	Fixed wing multi engine	AIRBUS INDUSTRIE	A321-231	2	379	0	Turbo-fan
N102DN	2020	Fixed wing multi engine	AIRBUS	A321-211	2	199	0	Turbo-fan
N102DU	NA	Fixed wing multi engine	C SERIES AIRCRAFT LTD PTNRSP	BD-500-1A10	2	133	0	Turbo-fan
N102HQ	2007	Fixed wing multi engine	EMBRAER-EMPRESA BRASILEIRA DE	ERJ 170-200 LR	2	80	0	Turbo-fan

```
SELECT *
FROM airlines
LIMIT 1, 6;
```

Table 4: 6 records

carrier	name
04Q	Tradewind Aviation
06Q	Master Top Linhas Aereas Ltd.
07Q	Flair Airlines Ltd.
09Q	Swift Air, LLC d/b/a Eastern Air Lines d/b/a Eastern
0BQ	DCA
0CQ	ACM AIR CHARTER GmbH

```
SELECT Reporting_Airline, name,
  (2024 - planes.year) AS plane_age,
  COUNT(DISTINCT tail_number) AS unique_planes,
  AVG(2024 - planes.year) AS mean_weighted_age,
  STDDEV_SAMP(2024 - planes.Year) AS mean_weighted_age
```

```
FROM flightdata
LEFT JOIN planes ON flightdata.tail_number = planes.tailnum
JOIN airlines ON flightdata.Reporting_Airline = airlines.carrier
WHERE flightdata.Year = 2023 AND origin IN ('EWR', 'JFK', 'LGA')
GROUP BY Reporting_Airline
ORDER BY mean_weighted_age ASC
LIMIT 1, 16;
```

Table 5: Displaying records 1 - 10

Reporting_Airline	name	plane_age	unique_planes	mean_weighted_age	mean_weighted_age
F9	Frontier Airlines Inc.	3	136	5.135947	2.365734
OO	SkyWest Airlines Inc.	16	188	6.850972	4.212619
AS	Alaska Airlines Inc.	2	215	7.404048	4.900640
NK	Spirit Air Lines	NA	218	7.738930	3.613592
HA	Hawaiian Airlines Inc.	10	24	10.728571	1.938314
MQ	Envoy Air	8	102	11.690840	5.861740
WN	Southwest Airlines Co.	12	841	11.848831	6.883556
YX	Republic Airline	11	229	12.993201	4.826594
9E	Endeavor Air Inc.	10	127	13.218725	3.968461
AA	American Airlines Inc.	10	890	13.943699	6.999990

On Your Own - Noninvasive Auditory Diagnostic Tools

You will use SQL to query the Wideband Acoustic Immittance (WAI) Database hosted by Smith College. WAI measurements are being developed as noninvasive auditory diagnostic tools for people of all ages, and the WAI Database hosts WAI ear measurements that have been published in peer-review articles. The goal of the database is to "enable auditory researchers to share WAI measurements and combine analyses over multiple datasets."

You have two primary goals:

1) duplicate Figure 1 from a 2019 manuscript by Susan Voss. You will need to query the WAI Database to build a dataset which you can pipe into `ggplot()` to recreate Figure 1 *as closely as possible*.

2) Find a study where subjects of different sex, race, ethnicity, or age groups were enrolled, and produce plots of frequency vs. mean absorption by group.

You should be using JOINs in both (1) and (2).

Hints:

- Parse the caption from Figure 1 carefully to determine how mean absorbances are calculated: "Mean absorbances for the 12 studies within the WAI database as of July 1, 2019. Noted in the legend are the peer-reviewed publications associated with the datasets, the number of individual ears, and the equipment used in the study. When multiple measurements were made on the same ear, the average from those measurements was used in the calculation across subjects for a given study. Some subjects have measurements on both a right and a left ear, and some subjects have measurements from only one ear; this figure includes every ear in the database and does not control for the effect of number of ears from each subject."
- filter for only the 12 studies shown in Figure 1 (and also for frequencies shown in Figure 1)
- study the patterns of frequencies. It seems that most researchers used the same set of frequencies for each subject, ear, and session.
- note the scale of the x-axis
- the key labels contain AuthorsShortList, Year, and Instrument, in addition to the number of unique ears (I think Werner's N may be incorrect?)

Starter Code:

```
library(tidyverse)
library(mdsr)
library(dbplyr)
library(DBI)

library(RMariaDB)
con <- dbConnect(
  MariaDB(), host = "scidb.smith.edu",
  user = "waiuser", password = "smith_waiDB",
  dbname = "wai"
)
Measurements <- tbl(con, "Measurements")
PI_Info <- tbl(con, "PI_Info")
Subjects <- tbl(con, "Subjects")

# collect(Measurements)
```

Run the following queries in a chunk with {sql, connection = con}:

- SHOW TABLES;
- DESCRIBE Measurements;
- SELECT * FROM PI_Info LIMIT 0,1;