USCOTS 2025 breakout session: Explore the airlines data using SQL and parquet

Nicholas Horton (nhorton@amherst.edu) and Jo Hardin (jo.hardin@pomona.edu)

2025-07-11

Table of contents

Introduction

This file analyzes airline flight data from the American Statistical Association's Data Expo 2024. Once the `1_download_data.qmd` Quarto file has been successfully rendered, you should be able to render this file (`2_explore_sql.qmd`) which shows off SQL syntax while analyzing the downloaded data.

See https://community.amstat.org/dataexpo/home for more information on the data.

See https://beanumber.github.io/abdwr3e/12-large.html and https://mdsr-book.github.io/mdsr3e/15-sqlI.html for resources on databases in R.

See https://hardin47.netlify.app/courses/sds261-sql/ for an accessible overview of SQL and databases.

Check for files

First we check that the files are where we expect. If you run the code below with no errors, you are ready to go! (If you run into problems, try rendering the file or “Change Working Directory” to “File Location” under the “Session” Menu in RStudio.

```
folder_name <- "data_airlines"
stopifnot(file.exists(folder_name))
stopifnot(file.exists(paste0(folder_name, "/Year=2024/data_0.parquet")))
```

Check reading via DuckDb

We begin by creating an in-memory database using DuckDb. This is just a placeholder that we can reference.

```
con_duckdb <- DBI::dbConnect(duckdb::duckdb())
```

Writing SQL code

The function `dbGetQuery()` (from the **DBI** package) allows us to run SQL code on the data which are linked to the `con_duckdb` connection. Here, the `con_duckdb` connection sets up an empty sandbox (using the duckDB SQL dialect) that can point to the parquet files you downloaded previously. The results are saved as a data frame in the local R environment.

```
LAX_ATL_flights <- DBI::dbGetQuery(
  con_duckdb,
  "SELECT
    COUNT(*) as N,
    AVG(ArrDelay) as Avg_Delay,
    YEAR,
    MONTH,
    DAYOFMONTH,
    DEST,
    FROM read_parquet('data_airlines/Year*/*.parquet')
    WHERE DEST = 'LAX' OR DEST = 'ATL'
    GROUP BY MONTH, YEAR, DAYOFMONTH, DEST;"
)
class(LAX_ATL_flights)
```

```
[1] "data.frame"
```

```r
dim(LAX_ATL_flights)
```

```
[1] 1094    6
```

```r
LAX_ATL_flights |> head()
```

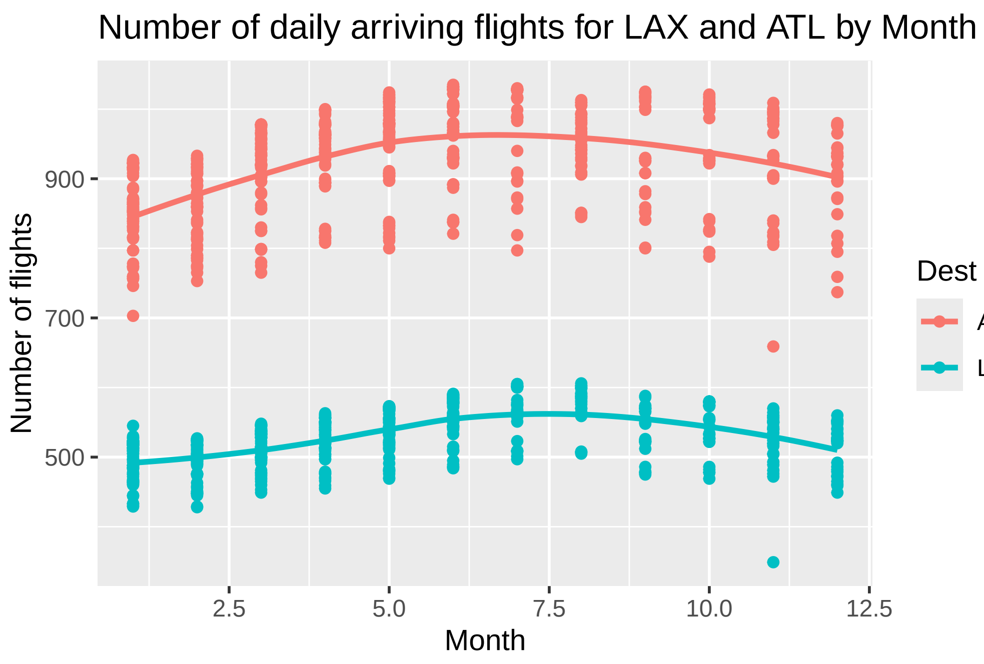
```
    N Avg_Delay Year Month DayofMonth Dest
1 984 75.307792 2023     8          6  ATL
2 994 10.604251 2023     8         14  ATL
3 851 -6.167059 2023     8         19  ATL
4 972 -1.638918 2023     8         20  ATL
5 560 26.932358 2023     8          6  LAX
6 599 13.952703 2023     8         14  LAX
```

Using the output from the SQL query

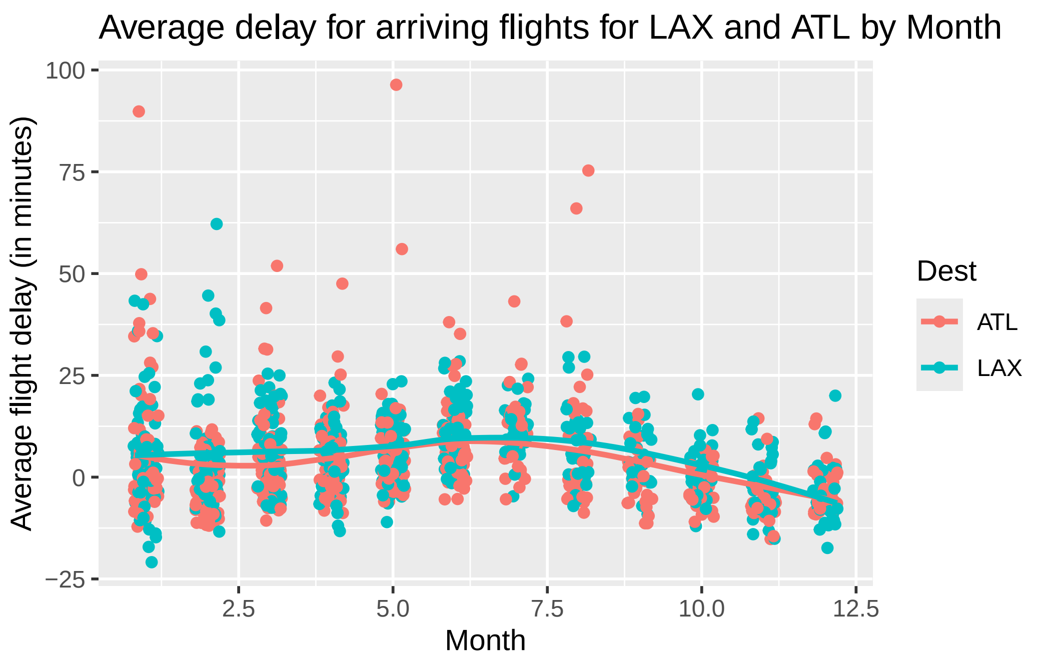
In the previous section, we used SQL to query the parquet files using a DuckDB connection. We created an object called `LAX_ATL_flights` which has 1094 rows and 6 columns and now lives in the local R environment.

Let's make some plots!

```r
ggplot(LAX_ATL_flights, aes(x = Month, y = N, color = Dest)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  labs(
    title = "Number of daily arriving flights for LAX and ATL by Month",
    x = "Month",
    y = "Number of flights"
  )
```



```
ggplot(LAX_ATL_flights, aes(x = jitter(Month), y = Avg_Delay, color = Dest)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  labs(
    title = "Average delay for arriving flights for LAX and ATL by Month",
    x = "Month",
    y = "Average flight delay (in minutes)"
  )
```



Another SQL example

With the SQL connection to the parquet files, we can run any SQL statements which result in a queried dataset. The query below returns the top ten rows of the 2023 airlines data frame for a handful of selected columns.

```
dbGetQuery(
  con_duckdb,
  "SELECT Tail_Number, Origin, OriginCityName, OriginState, Dest, DestCityName, DestState
  FROM 'data_airlines/Year=2023/data_0.parquet' LIMIT 10;"
)
```

```
   Tail_Number Origin OriginCityName OriginState Dest   DestCityName DestState
1       N605LR    BDL    Hartford, CT          CT  LGA   New York, NY        NY
2       N605LR    BDL    Hartford, CT          CT  LGA   New York, NY        NY
3       N331PQ    BDL    Hartford, CT          CT  LGA   New York, NY        NY
4       N906XJ    BDL    Hartford, CT          CT  LGA   New York, NY        NY
5       N337PQ    BDL    Hartford, CT          CT  LGA   New York, NY        NY
6       N336PQ    BDL    Hartford, CT          CT  LGA   New York, NY        NY
7       N311PQ    LGA    New York, NY          NY  CVG Cincinnati, OH        KY
8       N917XJ    LGA    New York, NY          NY  CVG Cincinnati, OH        KY
9       N336PQ    LGA    New York, NY          NY  CVG Cincinnati, OH        KY
10      N491PX    LGA    New York, NY          NY  BGM Binghamton, NY        NY
```

SQL in R two ways

Let's say we are interested in comparing the average delay time for the months January, February, and March by year. We could write SQL code to calculate the relevant averages:

```
dbGetQuery(
  con_duckdb,
  "SELECT
    COUNT(*) as N,
    AVG(ArrDelay) as AVG_DELAY,
    YEAR,
    MONTH
    FROM read_parquet('data_airlines/Year*/*.parquet')
    WHERE Month IN (1,2,3)
    GROUP BY MONTH, YEAR
    ORDER BY YEAR, MONTH;"
)
```

```
       N  AVG_DELAY Year Month
1 538837  7.7763929 2023     1
2 502749  4.1419877 2023     2
3 580322  9.0699067 2023     3
4 547271 10.3522984 2024     1
5 519221  0.5934551 2024     2
6 591767  6.5044763 2024     3
```

Stretch activity

Use the following chunk to write your own SQL code to answer the question: "What is the number of flights and average flight delay (in minutes) for flights arriving at DSM (Des Moines) each month we have data in 2023 and 2024?"

Closing the SQL connection

It is always good practice to close your connection when you are through with it (this is particularly important if you are accessing a remote database).

```
DBI::dbDisconnect(con_duckdb)
```