

# Explore the airlines data from parquet files

USCOTS 2025 breakout session

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

July 18, 2025

## Table of contents

Introduction . . . . .	1
Check for files . . . . .	2
Check reading via DuckDb . . . . .	2
<b>Writing SQL code</b>	<b>2</b>
SQL example 1 . . . . .	2
Using the output from the SQL query . . . . .	3
SQL example 2 . . . . .	5
SQL example 3 . . . . .	6
Stretch activity . . . . .	6
Closing the SQL connection . . . . .	6

## Introduction

The current Quarto 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 the current 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-sql.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, which is just a placeholder that we can reference.

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

## Writing SQL code

### SQL example 1

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"
```

```
dim(LAX_ATL_flights)
```

```
[1] 1094    6
```

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

	N	Avg_Delay	Year	Month	DayofMonth	Dest
1	578	1.1840278	2023	10	9	LAX
2	580	6.3719723	2023	10	19	LAX
3	478	-0.7953586	2023	10	21	LAX
4	575	0.5662021	2023	10	23	LAX
5	573	4.8743455	2023	10	27	LAX
6	478	-4.4444444	2023	10	31	LAX

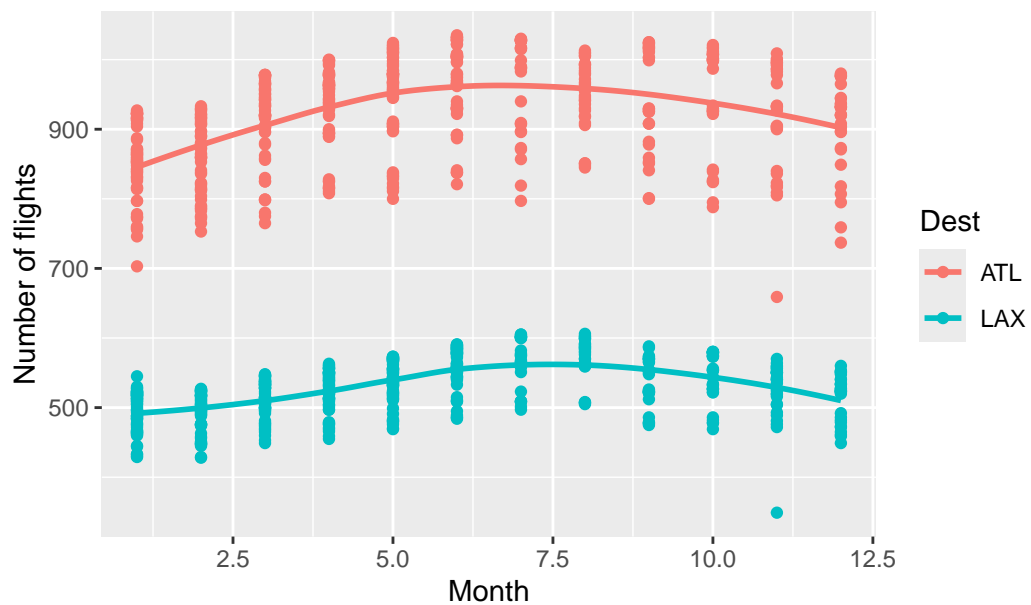
## Using the output from the SQL query

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.

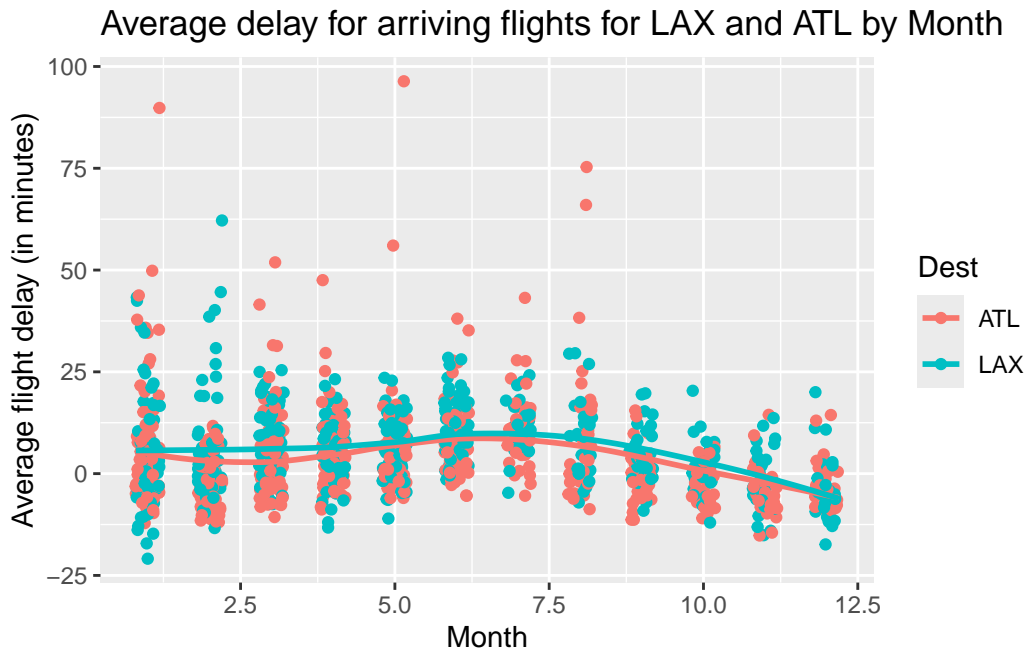
Let's make some plots!

```
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"  
  )
```

Number of daily arriving flights for LAX and ATL by Month



```
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)"
  )
```



## SQL example 2

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 example 3

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 (particularly important if you are accessing a remote database).

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