SQL in Quarto: three ways

USCOTS 2025 breakout session

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

July 18, 2025

Today's example

The `airlines` database

Consider a database of all US flights between 2013 and 2015 The flight information was collected from the Bureau of Transportation Statistics, US Department of Transportation.

The database is a superset of the data in the `nycflights13` **R** package that tracks only flights departing airports serving New York City in 2013.

SQL connection

To set up a **SQL** connection, you need the location of the server (`host`) as well as a `user`name and `password`. For example, you may want to use the subset of of data from 2013 to 2015 which exists in a **SQL** database hosted by Ben Baumer to connect to the text *Modern Data Science in R*.

~~~
```{r}
con_mysql <- DBI::dbConnect(
  RMariaDB::MariaDB(),
  dbname = "airlines",
  host = Sys.getenv("MDSR_HOST"),
  user = Sys.getenv("MDSR_USER"),
  password = Sys.getenv("MDSR_PWD")
)
```
~~~

Using SQL in Quarto

We can write **SQL** code in three distinct ways:

1. Using the **DBI** package, we can send **SQL** queries through an `r` chunk (method used in the introductory presentation).
2. Using the package **dbplyr**, **R** will directly translate **dplyr** code into **SQL**.
3. Using a `sql` chunk, we can write actual **SQL** code inside a Quarto document.

1. SQL queries via the DBI package

- Look at the first few rows of the `flights` data.

~~~
```{r}
DBI::dbGetQuery(con_mysql,
                "SELECT * FROM flights LIMIT 8;")
```
~~~

```
  year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
1 2013    10   1        2             10        -8      453            505
2 2013    10   1        4           2359         5      730            729
3 2013    10   1       11             15        -4      528            530
4 2013    10   1       14           2355        19      544            540
5 2013    10   1       16             17        -1      515            525
6 2013    10   1       22             20         2      552            554
7 2013    10   1       29             35        -6      808            816
8 2013    10   1       29             35        -6      449            458
  arr_delay carrier tailnum flight origin dest air_time distance cancelled
1       -12      AA  N201AA   2400    LAX  DFW      149     1235         0
2         1      FL  N344AT    710    SFO  ATL      247     2139         0
```

```
3        -2      AA  N3KMAA   1052    SFO  DFW      182     1464          0
4         4      AA  N3ENAA   2392    SEA  ORD      191     1721          0
5       -10      UA  N38473   1614    LAX  IAH      157     1379          0
6        -2      UA  N458UA    291    SFO  IAH      188     1635          0
7        -8      US  N551UW    436    LAX  CLT      256     2125          0
8        -9      AS  N402AS    108    ANC  SEA      181     1448          0
  diverted hour minute           time_hour
1        0    0     10 2013-10-01 00:10:00
2        0   23     59 2013-10-01 23:59:00
3        0    0     15 2013-10-01 00:15:00
4        0   23     55 2013-10-01 23:55:00
5        0    0     17 2013-10-01 00:17:00
6        0    0     20 2013-10-01 00:20:00
7        0    0     35 2013-10-01 00:35:00
8        0    0     35 2013-10-01 00:35:00
```

1. SQL queries via the DBI package

- How many flights per year are in the `flights` table?

~~~
```{r}
#| cache: true

DBI::dbGetQuery(con_mysql,
  "SELECT year, COUNT(*) AS num_flights
  FROM flights
  GROUP BY year
  ORDER BY num_flights;")
```
~~~

```
  year num_flights
1 2015     5819079
2 2014     5819811
3 2013     6369482
```

1. SQL queries via the DBI package

- What is the average Departure Delay (and number of flights) for each destination?

~~~
```{r}
#| cache: true

DBI::dbGetQuery(con_mysql,
  "SELECT dest,
          AVG(dep_delay) AS mean_delay,
          COUNT(*) AS num_flights
   FROM flights
   GROUP BY dest
   LIMIT 8;")
```
~~~

```
  dest mean_delay num_flights
1  ABE     9.9056        7253
2  ABI     8.9290        8114
3  ABQ    11.7639       74189
4  ABR     2.4078        2239
5  ABY     9.5748        3001
6  ACK     5.5521        1295
7  ACT    10.2526        5225
8  ACV    12.5367        7785
```

2. Translating dplyr code into SQL

```
flights_tbl <- dplyr::tbl(con_mysql, "flights")

flights_tbl |>
  head()
```

```
# Source:   SQL [?? x 21]
# Database: mysql  [mdsr_public@mdsr.crcbo51tmesf.us-east-2.rds.amazonaws.com:3306/airlines]
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <int>    <int>
1  2013    10     1        2             10        -8      453
2  2013    10     1        4           2359         5      730
3  2013    10     1       11             15        -4      528
4  2013    10     1       14           2355        19      544
5  2013    10     1       16             17        -1      515
6  2013    10     1       22             20         2      552
# i 14 more variables: sched_arr_time <int>, arr_delay <int>,
```

```
#   carrier <chr>, tailnum <chr>, flight <int>, origin <chr>,
#   dest <chr>, air_time <int>, distance <int>, cancelled <int>,
#   diverted <int>, hour <int>, minute <int>, time_hour <dttm>
```

2. Translating dplyr code into SQL

- Over what years is the `flights` data taken?

```
yrs <- flights_tbl |>
  summarize(min_year = min(year, na.rm = TRUE),
            max_year = max(year, na.rm = TRUE))

yrs
```

```
# Source:   SQL [?? x 2]
# Database: mysql  [mdsr_public@mdsr.crcbo51tmesf.us-east-2.rds.amazonaws.com:3306/airlines]
  min_year max_year
     <int>    <int>
1     2013     2015
```

2. Translating dplyr code into SQL

Because `flights_tbl` is not actually a `data.frame` in **R** (but instead a `tbl` in **SQL**), the work that was done above was actually performed in **SQL**. To see the **SQL** code, we can use the function `show_query`.

~~~
```{r}
dplyr::show_query(yrs)
```
~~~

```
<SQL>
SELECT MIN(`year`) AS `min_year`, MAX(`year`) AS `max_year`
FROM `flights`
```

2. Translating dplyr code into SQL

- Create a data set containing only flights between `LAX` and `BOS` in 2015.

~~~
```{r}
#| cache: true

la_bos <- flights_tbl |>
  filter(year == 2015 & ((origin == "LAX" & dest == "BOS") |
           (origin == "BOS" & dest == "LAX")))

dplyr::show_query(la_bos)
```
~~~

```
<SQL>
SELECT `flights`.*
FROM `flights`
WHERE (`year` = 2015.0 AND ((`origin` = 'LAX' AND `dest` = 'BOS') OR (`origin` = 'BOS' AND `
```

2. Translating dplyr code into SQL

- What is the average Departure Delay (and number of flights) for each destination?

~~~
```{r}
#| cache: true

aveDepDel <- flights_tbl |>
  group_by(dest) |>
  summarize(mean_delay = mean(dep_delay), num_flights = n())

aveDepDel

dplyr::show_query(aveDepDel)
```
~~~

```
# Source:   SQL [?? x 3]
# Database: mysql  [mdsr_public@mdsr.crcbo51tmesf.us-east-2.rds.amazonaws.com:3306/airlines]
   dest  mean_delay num_flights
   <chr>      <dbl>     <int64>
 1 ABE         9.91        7253
 2 ABI         8.93        8114
 3 ABQ        11.8        74189
 4 ABR         2.41        2239
 5 ABY         9.57        3001
 6 ACK         5.55        1295
```

```
 7 ACT        10.3         5225
 8 ACV        12.5         7785
 9 ACY        12.0         4311
10 ADK        -1.14         313
# i more rows
<SQL>
SELECT `dest`, AVG(`dep_delay`) AS `mean_delay`, COUNT(*) AS `num_flights`
FROM `flights`
GROUP BY `dest`
```

2. Translating dplyr code into SQL

- **dbplyr** doesn't translate every **R** command into **SQL**.
- **SQL** is not a statistical software and doesn't, for example, have a mechanism for creating data visualizations.
- track which **R** commands are connected to **SQL** at the **dbplyr** reference sheet.

3. Direct SQL queries via `sql` chunks

SQL queries can be written directly inside a `sql` chunk in Quarto.

~~~
```{sql}
#| connection: con_mysql

SELECT * FROM flights LIMIT 8;
```
~~~

Table 1: 8 records

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	cancelled	diverted	hour	minute	time_hour
2013	10	1	2	10	-8	453	505	-12	AA	N201AA	2400	LAX	DFW	149	1235	0	0	0	10	2013-10-01 00:10:00
2013	10	1	4	2359	5	730	729	1	FL	N344AT	710	SFO	ATL	247	2139	0	0	23	59	2013-10-01 23:59:00

year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	cancelled	diverted	hour	minute	time_hour
2013	10	1	11	15	-4	528	530	-2	AA	N3KMAA	1528	SFO	DFW	182	1464	0	0	0	15	2013-10-01 00:15:00
2013	10	1	14	2355	19	544	540	4	AA	N3ENAA	2392	SEA	ORD	191	1721	0	0	23	55	2013-10-01 23:55:00
2013	10	1	16	17	-1	515	525	-10	UA	N38417	1614	LAX	IAH	157	1379	0	0	0	17	2013-10-01 00:17:00
2013	10	1	22	20	2	552	554	-2	UA	N458UA	291	SFO	IAH	188	1635	0	0	0	20	2013-10-01 00:20:00
2013	10	1	29	35	-6	808	816	-8	US	N551UW	436	LAX	CLT	256	2125	0	0	0	35	2013-10-01 00:35:00
2013	10	1	29	35	-6	449	458	-9	AS	N402AS	108	ANC	SEA	181	1448	0	0	0	35	2013-10-01 00:35:00

3. Direct SQL queries via `sql` chunks

SQL queries can be written directly inside a `sql` chunk in Quarto.

~~~
```{sql}
#| connection: con_mysql
#| cache: true

SELECT year, count(*) AS num_flights
       FROM flights
       GROUP BY year
       ORDER BY num_flights;
```
~~~

Table 2: 3 records

year	num_flights
2015	5819079
2014	5819811
2013	6369482

3. Direct SQL queries via `sql` chunks

- What is the average Departure Delay (and number of flights) for each destination?

~~~
```{sql}
#| connection: con_mysql
#| cache: true

SELECT dest,
          AVG(dep_delay) AS mean_delay,
          COUNT(*) AS num_flights
   FROM flights
   GROUP BY dest
   LIMIT 8;
```
~~~

Table 3: 8 records

dest	mean_delay	num_flights
ABE	9.9056	7253
ABI	8.9290	8114
ABQ	11.7639	74189
ABR	2.4078	2239
ABY	9.5748	3001
ACK	5.5521	1295
ACT	10.2526	5225
ACV	12.5367	7785

Good practice

Always a good idea to terminate the **SQL** connection when you are done with it.

```r
RMariaDB::dbDisconnect(con_mysql, shutdown = TRUE)
```

Parquet

- Connecting to the local parquet files is very similar to connecting to a remote **SQL** database.
- Note that the datasets themselves have slightly diffent variable names (not ideal for teaching!!!)

Connecting to the parquet files via DuckDB

Start an in-memory database using DuckDB. The function `duckdb()` comes from the **duckdb** package.

```r
con_parq <- DBI::dbConnect(duckdb())
```

1. SQL queries via the DBI package

~~~
```{r}
DBI::dbGetQuery(con_parq,
                "SELECT *
                FROM read_parquet('activity/data_airlines/Year*/*.parquet')
                LIMIT 8;")
```
~~~

```
  Year Quarter Month DayofMonth DayOfWeek FlightDate Reporting_Airline
1 2023       1     1          2         1 2023-01-02                9E
2 2023       1     1          3         2 2023-01-03                9E
3 2023       1     1          4         3 2023-01-04                9E
4 2023       1     1          5         4 2023-01-05                9E
5 2023       1     1          6         5 2023-01-06                9E
6 2023       1     1          7         6 2023-01-07                9E
7 2023       1     1         14         6 2023-01-14                9E
8 2023       1     1         21         6 2023-01-21                9E
  DOT_ID_Reporting_Airline IATA_CODE_Reporting_Airline Tail_Number
1                    20363                          9E      N605LR
2                    20363                          9E      N605LR
```

```
3                  20363                          9E      N331PQ
4                  20363                          9E      N906XJ
5                  20363                          9E      N337PQ
6                  20363                          9E      N336PQ
7                  20363                          9E      N311PQ
8                  20363                          9E      N917XJ
  Flight_Number_Reporting_Airline OriginAirportID OriginAirportSeqID
1                            4628           10529            1052907
2                            4628           10529            1052907
3                            4628           10529            1052907
4                            4628           10529            1052907
5                            4628           10529            1052907
6                            4628           10529            1052907
7                            4628           12953            1295304
8                            4628           12953            1295304
  OriginCityMarketID Origin OriginCityName OriginState OriginStateFips
1              30529    BDL   Hartford, CT          CT              09
2              30529    BDL   Hartford, CT          CT              09
3              30529    BDL   Hartford, CT          CT              09
4              30529    BDL   Hartford, CT          CT              09
5              30529    BDL   Hartford, CT          CT              09
6              30529    BDL   Hartford, CT          CT              09
7              31703    LGA   New York, NY          NY              36
8              31703    LGA   New York, NY          NY              36
  OriginStateName OriginWac DestAirportID DestAirportSeqID DestCityMarketID
1     Connecticut        11         12953          1295304            31703
2     Connecticut        11         12953          1295304            31703
3     Connecticut        11         12953          1295304            31703
4     Connecticut        11         12953          1295304            31703
5     Connecticut        11         12953          1295304            31703
6     Connecticut        11         12953          1295304            31703
7        New York        22         11193          1119302            33105
8        New York        22         11193          1119302            33105
  Dest   DestCityName DestState DestStateFips DestStateName DestWac CRSDepTime
1  LGA   New York, NY        NY            36      New York      22       0800
2  LGA   New York, NY        NY            36      New York      22       0800
3  LGA   New York, NY        NY            36      New York      22       0800
4  LGA   New York, NY        NY            36      New York      22       0800
5  LGA   New York, NY        NY            36      New York      22       0800
6  LGA   New York, NY        NY            36      New York      22       0800
7  CVG Cincinnati, OH        KY            21      Kentucky      52       1500
8  CVG Cincinnati, OH        KY            21      Kentucky      52       1500
  DepTime DepDelay DepDelayMinutes DepDel15 DepartureDelayGroups DepTimeBlk
```

```
1    0757       -3              0        0                   -1  0800-0859
2    0755       -5              0        0                   -1  0800-0859
3    0755       -5              0        0                   -1  0800-0859
4    0754       -6              0        0                   -1  0800-0859
5    0759       -1              0        0                   -1  0800-0859
6    0750      -10              0        0                   -1  0800-0859
7    1452       -8              0        0                   -1  1500-1559
8    1450      -10              0        0                   -1  1500-1559
  TaxiOut WheelsOff WheelsOn TaxiIn CRSArrTime ArrTime ArrDelay ArrDelayMinutes
1   11.00      0808     0833  20.00       0905    0853      -12               0
2   19.00      0814     0851   6.00       0905    0857       -8               0
3   14.00      0809     0837   7.00       0905    0844      -21               0
4   13.00      0807     0845   3.00       0905    0848      -17               0
5   17.00      0816     0844   5.00       0905    0849      -16               0
6   17.00      0807     0845   7.00       0905    0852      -13               0
7   26.00      1518     1643   6.00       1720    1649      -31               0
8   16.00      1506     1650   5.00       1720    1655      -25               0
  ArrDel15 ArrivalDelayGroups ArrTimeBlk Cancelled CancellationCode Diverted
1        0                 -1  0900-0959         0             <NA>        0
2        0                 -1  0900-0959         0             <NA>        0
3        0                 -2  0900-0959         0             <NA>        0
4        0                 -2  0900-0959         0             <NA>        0
5        0                 -2  0900-0959         0             <NA>        0
6        0                 -1  0900-0959         0             <NA>        0
7        0                 -2  1700-1759         0             <NA>        0
8        0                 -2  1700-1759         0             <NA>        0
  CRSElapsedTime ActualElapsedTime AirTime Flights Distance DistanceGroup
1             65                56   25.00       1      101             1
2             65                62   37.00       1      101             1
3             65                49   28.00       1      101             1
4             65                54   38.00       1      101             1
5             65                50   28.00       1      101             1
6             65                62   38.00       1      101             1
7            140               117   85.00       1      585             3
8            140               125  104.00       1      585             3
  CarrierDelay WeatherDelay NASDelay SecurityDelay LateAircraftDelay
1         <NA>         <NA>     <NA>          <NA>              <NA>
2         <NA>         <NA>     <NA>          <NA>              <NA>
3         <NA>         <NA>     <NA>          <NA>              <NA>
4         <NA>         <NA>     <NA>          <NA>              <NA>
5         <NA>         <NA>     <NA>          <NA>              <NA>
6         <NA>         <NA>     <NA>          <NA>              <NA>
7         <NA>         <NA>     <NA>          <NA>              <NA>
```

```
8         <NA>         <NA>     <NA>          <NA>              <NA>
  FirstDepTime TotalAddGTime LongestAddGTime DivAirportLandings DivReachedDest
1         <NA>          <NA>            <NA>                  0           <NA>
2         <NA>          <NA>            <NA>                  0           <NA>
3         <NA>          <NA>            <NA>                  0           <NA>
4         <NA>          <NA>            <NA>                  0           <NA>
5         <NA>          <NA>            <NA>                  0           <NA>
6         <NA>          <NA>            <NA>                  0           <NA>
7         <NA>          <NA>            <NA>                  0           <NA>
8         <NA>          <NA>            <NA>                  0           <NA>
  DivActualElapsedTime DivArrDelay DivDistance Div1Airport Div1AirportID
1                 <NA>        <NA>        <NA>        <NA>          <NA>
2                 <NA>        <NA>        <NA>        <NA>          <NA>
3                 <NA>        <NA>        <NA>        <NA>          <NA>
4                 <NA>        <NA>        <NA>        <NA>          <NA>
5                 <NA>        <NA>        <NA>        <NA>          <NA>
6                 <NA>        <NA>        <NA>        <NA>          <NA>
7                 <NA>        <NA>        <NA>        <NA>          <NA>
8                 <NA>        <NA>        <NA>        <NA>          <NA>
  Div1AirportSeqID Div1WheelsOn Div1TotalGTime Div1LongestGTime Div1WheelsOff
1             <NA>         <NA>           <NA>             <NA>          <NA>
2             <NA>         <NA>           <NA>             <NA>          <NA>
3             <NA>         <NA>           <NA>             <NA>          <NA>
4             <NA>         <NA>           <NA>             <NA>          <NA>
5             <NA>         <NA>           <NA>             <NA>          <NA>
6             <NA>         <NA>           <NA>             <NA>          <NA>
7             <NA>         <NA>           <NA>             <NA>          <NA>
8             <NA>         <NA>           <NA>             <NA>          <NA>
  Div1TailNum Div2Airport Div2AirportID Div2AirportSeqID Div2WheelsOn
1        <NA>        <NA>          <NA>             <NA>         <NA>
2        <NA>        <NA>          <NA>             <NA>         <NA>
3        <NA>        <NA>          <NA>             <NA>         <NA>
4        <NA>        <NA>          <NA>             <NA>         <NA>
5        <NA>        <NA>          <NA>             <NA>         <NA>
6        <NA>        <NA>          <NA>             <NA>         <NA>
7        <NA>        <NA>          <NA>             <NA>         <NA>
8        <NA>        <NA>          <NA>             <NA>         <NA>
  Div2TotalGTime Div2LongestGTime Div2WheelsOff Div2TailNum Div3Airport
1           <NA>             <NA>          <NA>        <NA>        <NA>
2           <NA>             <NA>          <NA>        <NA>        <NA>
3           <NA>             <NA>          <NA>        <NA>        <NA>
4           <NA>             <NA>          <NA>        <NA>        <NA>
5           <NA>             <NA>          <NA>        <NA>        <NA>
```

```
6           <NA>             <NA>          <NA>        <NA>        <NA>
7           <NA>             <NA>          <NA>        <NA>        <NA>
8           <NA>             <NA>          <NA>        <NA>        <NA>
  Div3AirportID Div3AirportSeqID Div3WheelsOn Div3TotalGTime Div3LongestGTime
1          <NA>             <NA>         <NA>           <NA>             <NA>
2          <NA>             <NA>         <NA>           <NA>             <NA>
3          <NA>             <NA>         <NA>           <NA>             <NA>
4          <NA>             <NA>         <NA>           <NA>             <NA>
5          <NA>             <NA>         <NA>           <NA>             <NA>
6          <NA>             <NA>         <NA>           <NA>             <NA>
7          <NA>             <NA>         <NA>           <NA>             <NA>
8          <NA>             <NA>         <NA>           <NA>             <NA>
  Div3WheelsOff Div3TailNum Div4Airport Div4AirportID Div4AirportSeqID
1          <NA>        <NA>        <NA>          <NA>             <NA>
2          <NA>        <NA>        <NA>          <NA>             <NA>
3          <NA>        <NA>        <NA>          <NA>             <NA>
4          <NA>        <NA>        <NA>          <NA>             <NA>
5          <NA>        <NA>        <NA>          <NA>             <NA>
6          <NA>        <NA>        <NA>          <NA>             <NA>
7          <NA>        <NA>        <NA>          <NA>             <NA>
8          <NA>        <NA>        <NA>          <NA>             <NA>
  Div4WheelsOn Div4TotalGTime Div4LongestGTime Div4WheelsOff Div4TailNum
1         <NA>           <NA>             <NA>          <NA>        <NA>
2         <NA>           <NA>             <NA>          <NA>        <NA>
3         <NA>           <NA>             <NA>          <NA>        <NA>
4         <NA>           <NA>             <NA>          <NA>        <NA>
5         <NA>           <NA>             <NA>          <NA>        <NA>
6         <NA>           <NA>             <NA>          <NA>        <NA>
7         <NA>           <NA>             <NA>          <NA>        <NA>
8         <NA>           <NA>             <NA>          <NA>        <NA>
  Div5Airport Div5AirportID Div5AirportSeqID Div5WheelsOn Div5TotalGTime
1        <NA>          <NA>             <NA>         <NA>           <NA>
2        <NA>          <NA>             <NA>         <NA>           <NA>
3        <NA>          <NA>             <NA>         <NA>           <NA>
4        <NA>          <NA>             <NA>         <NA>           <NA>
5        <NA>          <NA>             <NA>         <NA>           <NA>
6        <NA>          <NA>             <NA>         <NA>           <NA>
7        <NA>          <NA>             <NA>         <NA>           <NA>
8        <NA>          <NA>             <NA>         <NA>           <NA>
  Div5LongestGTime Div5WheelsOff Div5TailNum column109
1             <NA>          <NA>        <NA>      <NA>
2             <NA>          <NA>        <NA>      <NA>
3             <NA>          <NA>        <NA>      <NA>
```

```
4              <NA>          <NA>         <NA>      <NA>
5              <NA>          <NA>         <NA>      <NA>
6              <NA>          <NA>         <NA>      <NA>
7              <NA>          <NA>         <NA>      <NA>
8              <NA>          <NA>         <NA>      <NA>
```

1. SQL queries via the DBI package

- What is the average Departure Delay (and number of flights) for each destination?
- Can now apply the `MEDIAN()` function!

~~~
```{r}
DBI::dbGetQuery(con_parq,
                "SELECT MEDIAN(DepDelay) AS med_delay,
                        AVG(DepDelay) AS mean_delay, Dest
                FROM read_parquet('activity/data_airlines/Year*/*.parquet')
                GROUP BY Dest
                LIMIT 8;")
```
~~~

```
  med_delay mean_delay Dest
1        -1  11.344920  STL
2        -2  11.778499  CHS
3        -3  11.035951  GRB
4        -2  12.511458  SYR
5        -2  11.553647  ROC
6        -2  11.161244  CMH
7        -3  10.076172  MDT
8        -4   5.933875  MLU
```

2. Translating dplyr code into SQL

- Creating a `tbl` requires a pointer to the folder where the parquet files live.

~~~
```{r}
flights_parq <- tbl(
  con_parq,
  "read_parquet('activity/data_airlines/Year*/*.parquet')")

flights_parq |>
~~~

```
  head()
```
```

Source: SQL [?? x 110]
Database: DuckDB v1.1.2 [root@Darwin 24.5.0:R 4.4.2/:memory:]
   Year Quarter Month DayofMonth DayOfWeek FlightDate
  <dbl>   <dbl> <dbl>   <dbl>   <dbl> <date>
1 2023   1   1   2   1 2023-01-02
2 2023   1   1   3   2 2023-01-03
3 2023   1   1   4   3 2023-01-04
4 2023   1   1   5   4 2023-01-05
5 2023   1   1   6   5 2023-01-06
6 2023   1   1   7   6 2023-01-07
i 104 more variables: Reporting_Airline <chr>,
DOT_ID_Reporting_Airline <dbl>,
IATA_CODE_Reporting_Airline <chr>, Tail_Number <chr>,
Flight_Number_Reporting_Airline <dbl>, OriginAirportID <dbl>,
OriginAirportSeqID <dbl>, OriginCityMarketID <dbl>, Origin <chr>,
OriginCityName <chr>, OriginState <chr>, OriginStateFips <chr>,
OriginStateName <chr>, OriginWac <dbl>, DestAirportID <dbl>, ...
```

## 2. Translating dplyr code into SQL

- Over what years is the `flights` data taken?

~~~
```{r}
yrs <- flights_parq |>
  summarize(min_year = min(Year, na.rm = TRUE),
            max_year = max(Year, na.rm = TRUE))

yrs

dplyr::show_query(yrs)
```
~~~

```
# Source:   SQL [?? x 2]
# Database: DuckDB v1.1.2 [root@Darwin 24.5.0:R 4.4.2/:memory:]
  min_year max_year
     <dbl>    <dbl>
1     2023     2024
<SQL>
```

```
SELECT MIN("Year") AS min_year, MAX("Year") AS max_year
FROM (FROM read_parquet('activity/data_airlines/Year*/*.parquet')) q01
```

2. Translating dplyr code into SQL

- Create a data set containing only flights between `LAX` and `BOS` in 2024.

~~~
```{r}
#| cache: true

la_bos_parquet <- flights_parq |>
  filter(Year == 2024 & ((Origin == "LAX" & Dest == "BOS") |
           (Origin == "BOS" & Dest == "LAX")))

dplyr::show_query(la_bos_parquet)
```
~~~

```
<SQL>
SELECT q01.*
FROM (FROM read_parquet('activity/data_airlines/Year*/*.parquet')) q01
WHERE ("Year" = 2024.0 AND ((Origin = 'LAX' AND Dest = 'BOS') OR (Origin = 'BOS' AND Dest =
```

2. Translating dplyr code into SQL

- What is the average Departure Delay (and number of flights) for each destination?
- Can now apply the `MEDIAN()` function!

~~~
```{r}
#| cache: true

aveDepDel_parquet <- flights_parq |>
  group_by(Dest) |>
  summarize(mean_delay = mean(DepDelay),
            med_delay = median(DepDelay),
            num_flights = n())

aveDepDel_parquet

dplyr::show_query(aveDepDel_parquet)
```
~~~

```
# Source:   SQL [?? x 4]
# Database: DuckDB v1.1.2 [root@Darwin 24.5.0:R 4.4.2/:memory:]
   Dest  mean_delay med_delay num_flights
   <chr>      <dbl>     <dbl>       <dbl>
 1 FAT         11.5        -2       16727
 2 FLL         18.9         0      139111
 3 STT         10.7        -2        7959
 4 CHS         11.8        -2       37139
 5 MDT         10.1        -3        8176
 6 CMH         11.2        -2       61620
 7 RSW         14.2        -1       53403
 8 OKC         12.8        -1       32305
 9 LIT         12.1        -1       17742
10 OMA         12.2        -1       34355
# i more rows
<SQL>
SELECT
  Dest,
  AVG(DepDelay) AS mean_delay,
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY DepDelay) AS med_delay,
  COUNT(*) AS num_flights
FROM (FROM read_parquet('activity/data_airlines/Year*/*.parquet')) q01
GROUP BY Dest
```

3. Direct SQL queries via `sql` chunks

~~~
```{sql}
#| connection: con_parq

SELECT *
FROM read_parquet('activity/data_airlines/Year*/*.parquet')
LIMIT 8;
```
~~~

Year	Q[illegible]	M[illegible]	D[illegible]	D[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2023	1	1	2	1	2023-01-02	[illegible] Hartford, CT [illegible]	[illegible] New York, NY	36	New York	22	0800 0757 -3	0	0	-1 0800-0859 [illegible] 12 0 [illegible]

2023	1	1	3	2	2023-01-03	9E-20363N605LR5032987	Hartford, CT	09 Connecticut 11 129 231 703	New York, NY	36	New York	22	0800	0755 -5	0	0	-1	0800-0859	0810 0845 1100 0905 -57	8	0
2023	1	1	4	3	2023-01-04	9E-20363N346PQ5032987	Hartford, CT	09 Connecticut 11 129 231 703	New York, NY	36	New York	22	0800	0755 -5	0	0	-1	0800-0859	0814 0837 1100 0905 -44	21	0
2023	1	1	5	4	2023-01-05	9E-20363N906XJ5032987	Hartford, CT	09 Connecticut 11 129 231 703	New York, NY	36	New York	22	0800	0754 -6	0	0	-1	0800-0859	0813 0830 1100 0905 -48	17	0
2023	1	1	6	5	2023-01-06	9E-20363N347PQ5032987	Hartford, CT	09 Connecticut 11 129 231 703	New York, NY	36	New York	22	0800	0759 -1	0	0	-1	0800-0859	0817 0816 1100 0905 -49	16	0
2023	1	1	7	6	2023-01-07	9E-20363N346PQ5032987	Hartford, CT	09 Connecticut 11 129 231 703	New York, NY	36	New York	22	0800	0750 -10	0	0	-1	0800-0859	0807 0845 1100 0905 -52	13	0
2023	1	1	14	6	2023-01-14	9E-20363N461PQ5301	New York, NY	36 New York 22 1119 3930	Cincinnati, OH	CVG KY 21 Kentucky 52 1559			1500	1452 -8	0	0	-1	1500-1559	1526 1518 1630 1726 -49	31	0
2023	1	1	21	6	2023-01-21	9E-20363N467PQ5301	New York, NY	36 New York 22 1119 3930	Cincinnati, OH	CVG KY 21 Kentucky 52 1559			1500	1450 -10	0	0	-1	1500-1559	1516 1506 1650 1726 -55	25	0

3. Direct SQL queries via `sql` chunks

- What is the average Departure Delay (and number of flights) for each destination?
- Can now apply the `MEDIAN()` function!

~~~
```{sql}
#| connection: con_parq
#| cache: true

SELECT MEDIAN(DepDelay) AS med_delay,
                        AVG(DepDelay) AS mean_delay, Dest
                FROM read_parquet('activity/data_airlines/Year*/*.parquet')
                GROUP BY Dest
                LIMIT 8;
```
~~~

Table 5: 8 records

med_delay	mean_delay	Dest
-2	9.750851	PDX
-1	13.586533	MSY
-2	16.059348	BOS
-2	8.899172	SNA
-1	13.173580	ALB
-1	13.369200	PVD
-1	10.840029	SAN
-4	3.948148	KTN

Good practice

Always a good idea to terminate the **SQL** connection when you are done with it.

```r
RMariaDB::dbDisconnect(con_parq, shutdown = TRUE)
```