Lecture 17: Overview of SQL

Data stored in multiple tables

The `nycflights13` package contains information on flights from NYC airports in 2013. The data is stored across several data frames:

- `airlines`: information on each airline
- `airports`: information on each airport
- `flights`: information on each flight
- `planes`: information on each plane
- `weather`: hourly weather data

Limitations

```
1 nycflights13::flights |>
2   object.size() |>
3   print(units = "Mb")
```

```
38.8 Mb
```

- R stores objects in memory (RAM), which can be easily accessed
- The amount of RAM on your computer is a limit on the possible size of objects
- Objects larger than a few Gb are generally too big to load

Full airlines data

The `nycflights13` package contains a small subset of a database on 48 million flights. The `airlines` database includes the following tables:

- `airports`
- `carriers`
- `flights`
- `planes`

This data is too big to store locally, but can be on servers which we can access remotely.

Connecting to an SQL server

```
1  library(tidyverse)
2  library(mdsr)
3  library(DBI)
4
5  db <- dbConnect_scidb("airlines")
6
7  query <- "
8    SHOW TABLES;
9  "
10 dbGetQuery(db, query)
```

connecting to remote server

query we want to run (represented as a string in R)

run that query remotely on server, get results back from server

```
  Tables_in_airlines
1           airports
2           carriers
3            flights
4             planes
```

An example query

```
SELECT
  name,
  SUM(1) AS N,
  SUM(arr_delay <= 15) / SUM(1) AS pct_ontime
FROM flights
JOIN carriers ON flights.carrier = carriers.carrier
WHERE year = 2015 AND month = 9
  AND dest = 'JFK'
GROUP BY name
HAVING N >= 100
ORDER BY pct_ontime DESC
LIMIT 0,4;
```

Warm-up

What do you think each part of this query is doing?

```
1  SELECT
2    name,
3    SUM(1) AS N,
4    SUM(arr_delay <= 15) / SUM(1) AS pct_ontime
5  FROM flights
6  JOIN carriers ON flights.carrier = carriers.carrier
7  WHERE year = 2015 AND month = 9
8    AND dest = 'JFK'
9  GROUP BY name
10 HAVING N >= 100
11 ORDER BY pct_ontime DESC
12 LIMIT 0,4;
```

	name	N	pct_ontime
1	Virgin America	322	0.8789
2	United Air Lines Inc.	356	0.8736
3	Delta Air Lines Inc.	2100	0.8505
4	American Airlines Inc.	1500	0.8113

General structure of an SQL query

```
SELECT ...
FROM ...
JOIN ...
WHERE ...
GROUP BY ...
HAVING ...
ORDER BY ...
LIMIT ...
```

- The SELECT and FROM clauses are *required*
- Clauses must be written in this order

SELECT ... FROM

choose all columns

```
1 SELECT * FROM carriers LIMIT 0, 10;
```

	carrier	name
1	02Q	Titan Airways
2	04Q	Tradewind Aviation
3	05Q	Comlux Aviation, AG
4	06Q	Master Top Linhas Aereas Ltd.
5	07Q	Flair Airlines Ltd.
6	09Q	Swift Air, LLC
7	0BQ	DCA
8	0CQ	ACM AIR CHARTER GmbH
9	0GQ	Inter Island Airways, d/b/a Inter Island Air
10	0HQ	Polar Airlines de Mexico d/b/a Nova Air

- SELECT: the columns to be retrieved
- FROM: the table containing the data
- LIMIT: limit the rows to return

carriers |>
select (....)

SELECT ... FROM

```
1 SELECT ... FROM ... LIMIT 0, 10;
```

What if I want the `year`, `origin`, `dest`, `dep_delay`, and `arr_delay` columns from the `flights` table?

SELECT ... FROM

What if I want the `year`, `origin`, `dest`, `dep_delay`, and `arr_delay` columns from the `flights` table?

```
1 SELECT
2   year, origin, dest,
3   dep_delay, arr_delay
4 FROM flights
5 LIMIT 0, 5;
```

	year	origin	dest	dep_delay	arr_delay
1	2013	LAX	DFW	-8	-12
2	2013	SFO	ATL	5	1
3	2013	SFO	DFW	-4	-2
4	2013	SEA	ORD	19	4
5	2013	LAX	IAH	-1	-10

SELECT ... FROM

```
1 SELECT
2   year, origin, dest,
3   dep_delay, arr_delay
4 FROM flights
5 LIMIT 0, 5;
```

What if I also want to calculate the difference between arrival delay and departure delay?

arr_delay − dep_delay AS delay_diff

SELECT ... FROM

What if I also want to calculate the difference between arrival delay and departure delay?

```
1 SELECT ← mutating, summarizing, selecting columns
2   year, origin, dest, dep_delay, arr_delay,
3   arr_delay - dep_delay AS delay_diff
4 FROM flights
5 LIMIT 0, 3;
```

```
  year origin dest dep_delay arr_delay delay_diff
1 2013    LAX  DFW        -8       -12         -4
2 2013    SFO  ATL         5         1         -4
3 2013    SFO  DFW        -4        -2          2
```

What are the equivalent `dplyr` functions?

mutate & select

Converting from R to SQL

```r
flights <- tbl(db, "flights")

flights |>
  select(year, origin, dest, dep_delay, arr_delay) |>
  mutate(delay_diff = arr_delay - dep_delay) |>
  head() |>
  show_query()
```

```
<SQL>
SELECT
  `year`,
  `origin`,
  `dest`,
  `dep_delay`,
  `arr_delay`,
  `arr_delay` - `dep_delay` AS `delay_diff`
FROM `flights`
LIMIT 6
```

Calculating summary statistics

Back to our original SQL query:

```
1  SELECT
2    SUM(1) AS N,
3    SUM(arr_delay <= 15) / SUM(1) AS pct_ontime
4  FROM flights
5  LIMIT 0, 10;
```

```
         N pct_ontime
1 18008372     0.8091
```

return summary statistics (# of flights & fraction of on-time flights) for each group

Calculating summary statistics

SELECT can also be used to calculate summary statistics. For example, if we want the average departure delay:

```
1  SELECT
2    AVG(dep_delay) AS mean_dep_delay
3  FROM flights
4  LIMIT 0, 10;
```

```
  mean_dep_delay
1         9.7471
```

WHERE

Now suppose that I only want the mean departure delay for flights from EWR in 2013:

```sql
SELECT
  AVG(dep_delay) AS mean_dep_delay
FROM flights
WHERE year = 2013 AND origin = 'EWR'
LIMIT 0, 10;
```

```
  mean_dep_delay
1         14.703
```

What do you think should I do if I want the mean delay for each airport in November 2013?

flights |>
  filter(year = , origin =) |>
  summarize (...)

WHERE year = 2013 AND month = 11
GROUP BY origin

GROUP BY

```sql
SELECT
  AVG(dep_delay) AS mean_dep_delay
FROM flights
WHERE year = 2013 AND month = 11
GROUP BY origin
LIMIT 0, 10;
```

	mean_dep_delay
1	3.7766
2	2.2070
3	8.0122
4	-0.2985
5	5.2750
6	3.6619
7	8.2222
8	18.8750
9	-2.1042
10	8.0443

Do you notice anything about this output?

GROUP BY

```sql
SELECT
  origin,
  AVG(dep_delay) AS mean_dep_delay
FROM flights
WHERE year = 2013 AND month = 11
GROUP BY origin
LIMIT 0, 10;
```

	origin	mean_dep_delay
1	ABE	3.7766
2	ABI	2.2070
3	ABQ	8.0122
4	ABR	-0.2985
5	ABY	5.2750
6	ACT	3.6619
7	ACV	8.2222
8	ADK	18.8750
9	ADQ	-2.1042
10	AEX	8.0443

Class activity

https://sta279-s24.github.io/class_activities/ca_lecture_17.html