Ch 15 from Modern Data Science with R

Database querying using SQL

Section 15.4: The SQL data manipulation language

We will use `sql` chunks to write and execute SQL queries. Unfortunately, these chunks will only run (in both Rmd and qmd files) when you render (knit) the entire document. In newer versions of RStudio you can run current `sql` chunks in a quarto document using the right arrow.

```
DESCRIBE airports;
```

Table 1: 9 records

Field	Type	Null	Key	Default	Extra
faa	varchar(3)	NO	PRI		
name	varchar(255)	YES		NA	
lat	decimal(10,7)	YES		NA	
lon	decimal(10,7)	YES		NA	
alt	int(11)	YES		NA	
tz	smallint(4)	YES		NA	
dst	char(1)	YES		NA	
city	varchar(255)	YES		NA	
country	varchar(255)	YES		NA	

Here's the query we were exploring using `dplyr` in Part 1. It's kind of clunky, but to run it in this older version of RStudio, we have to output and then print out an R data frame that we call `mydataframe`.

```
SELECT
  c.name,
  SUM(1) AS N,
  SUM(arr_delay <= 15) / SUM(1) AS pct_ontime
FROM flights AS f
JOIN carriers AS c ON f.carrier = c.carrier
WHERE year = 2016 AND month = 9
  AND dest = 'JFK'
GROUP BY name
HAVING N >= 100
ORDER BY pct_ontime DESC
LIMIT 0,4;
```

```
mydataframe
```

```
##                     name    N pct_ontime
## 1   Delta Air Lines Inc. 2396     0.8689
## 2         Virgin America  347     0.8329
## 3        JetBlue Airways 3463     0.8169
## 4 American Airlines Inc. 1397     0.7817
```

Section 15.4.1: SELECT...FROM Note that every SQL query must have a SELECT and FROM. Here we also create a new column that's a function of other columns, combining latitude and longitude into a single column of geographic coordinates named `coordinates`. Note that we also rename the airport name column using the AS keyword.

```
SELECT
  name AS airport_name,
  CONCAT('(', lat, ', ', lon, ')') AS coordinates
FROM airports
LIMIT 0, 6;
```

```
mydataframe
```

```
##                     airport_name               coordinates
## 1              Lansdowne Airport (41.1304722, -80.6195833)
## 2  Moton Field Municipal Airport (32.4605722, -85.6800278)
## 3            Schaumburg Regional (41.9893408, -88.1012428)
## 4               Randall Airport (41.4319120, -74.3915611)
## 5          Jekyll Island Airport (31.0744722, -81.4277778)
## 6 Elizabethton Municipal Airport (36.3712222, -82.1734167)
```

Section 15.4.2: WHERE The WHERE clause is analogous to the `filter()` command in `dplyr` – it allows you to restrict the set of rows that are retrieved to only those rows that match a certain condition. Here we pick off the few flights that left Bradley International Airport on June 26th, 2013.

```
SELECT
  year, month, day, origin, dest,
  flight, carrier
FROM flights
WHERE year = 2013 AND month = 6 AND day = 26
AND origin = 'BDL'
LIMIT 0, 6;
```

```
mydataframe
```

```
##   year month day origin dest flight carrier
## 1 2013     6  26    BDL  EWR   4714      EV
## 2 2013     6  26    BDL  MIA   2015      AA
## 3 2013     6  26    BDL  DTW   1644      DL
## 4 2013     6  26    BDL  BWI   2584      WN
## 5 2013     6  26    BDL  ATL   1065      DL
## 6 2013     6  26    BDL  DCA   1077      US
```

Here we use STR_TO_DATE to convert 3 columns to a date field. Notice how `year`, `month`, and `day` are not part of the final data frame, but we can still use these columns in the WHERE clause (in fact, WHERE only operates on columns from the original table).

```
SELECT
  STR_TO_DATE(CONCAT(year, '-', month, '-', day), '%Y-%m-%d') AS theDate,
  origin,
  flight, carrier
```

```
FROM flights
WHERE year = 2013 AND month = 6 AND day = 26
  AND origin = 'BDL'
LIMIT 0, 6;
```

```
mydataframe
```

```
##      theDate origin flight carrier
## 1 2013-06-26    BDL   4714      EV
## 2 2013-06-26    BDL   2015      AA
## 3 2013-06-26    BDL   1644      DL
## 4 2013-06-26    BDL   2584      WN
## 5 2013-06-26    BDL   1065      DL
## 6 2013-06-26    BDL   1077      US
```

Therefore, this query produces an error, while the following query works since WHERE can operate on functions of original columns. However, the second query is very slow because it doesn't make use of indices of the original variables (see Section 16.1.4).

```
SELECT
  STR_TO_DATE(CONCAT(year, '-', month, '-', day), '%Y-%m-%d') AS theDate,
  origin, flight, carrier
FROM flights
WHERE theDate = '2013-06-26'
  AND origin = 'BDL'
LIMIT 0, 6;
```

```
SELECT
  STR_TO_DATE(CONCAT(year, '-', month, '-', day), '%Y-%m-%d') AS theDate,
  origin, flight, carrier
FROM flights
WHERE STR_TO_DATE(CONCAT(year, '-', month, '-', day), '%Y-%m-%d') =
  '2013-06-26'
  AND origin = 'BDL'
LIMIT 0, 6;
```

```
mydataframe
```

```
##      theDate origin flight carrier
## 1 2013-06-26    BDL   4714      EV
## 2 2013-06-26    BDL   2015      AA
## 3 2013-06-26    BDL   1644      DL
## 4 2013-06-26    BDL   2584      WN
## 5 2013-06-26    BDL   1065      DL
## 6 2013-06-26    BDL   1077      US
```

There are several keywords that can help with filtering using WHERE. These include BETWEEN and IN. Explain in the queries below why:

- the first block produces 5 rows while the second block produces 2 rows
- both blocks produce only a single column

```
SELECT
  DISTINCT STR_TO_DATE(CONCAT(year, '-', month, '-', day), '%Y-%m-%d')
    AS theDate
FROM flights
WHERE year = 2013 AND month = 6 AND day BETWEEN 26 and 30
  AND origin = 'BDL'
LIMIT 0, 6;
```

```
SELECT
  DISTINCT STR_TO_DATE(CONCAT(year, '-', month, '-', day), '%Y-%m-%d')
    AS theDate
FROM flights
WHERE year = 2013 AND month = 6 AND day IN (26, 30)
  AND origin = 'BDL'
LIMIT 0, 6;
```

In the following two blocks, explain why the first one contains 557,874 rows while the second only contains 2,542.

```
SELECT
  COUNT(*) AS N
FROM flights
WHERE year = 2013 AND month = 6 OR day = 26
  AND origin = 'BDL';
```

```
SELECT
  COUNT(*) AS N
FROM flights
WHERE year = 2013 AND (month = 6 OR day = 26)
  AND origin = 'BDL';
```

Section 15.4.2: GROUP BY The GROUP BY clause allows one to aggregate multiple rows according to some criteria. The challenge when using GROUP BY is specifying how multiple rows of data should be reduced into a single value. Aggregate functions (e.g., COUNT(), SUM(), MAX(), and AVG()) are necessary.

For example, we know that there were 65 flights that left Bradley Airport on June 26th, 2013, but how many belonged to each airline carrier? To get this information we need to aggregate the individual flights, based on who the carrier was.