Homework 7

Carmen Canedo

27 November 2020

Loading Libraries

Keys

Primary keys uniquely identify an observation within its own table. *Foreign keys* uniquely identify an observation in another table.

To be sure the identification is unique, count the primary keys and see if any are greater than 1.

There is not a primary key in the flights tibble because the values repeat. We need to add a *surrogate key* to flights.

Exercise 1 Identify the primary keys in three datasets.

Be sure to show that you have the primary key by showing there are no duplicate entries.

a. Lahman Batting Primary Key

```
## [1] playerID stint    yearID   n
## <0 rows> (or 0-length row.names)
```

The primary key for `Lahman::Batting` is `playerID`, `stint`, and `yearID`.

b. Babynames Primary key

The primary key for `babynames` is `name`, `year`, and `sex`.

c. Atmos Primary Key

```
## # A tibble: 0 x 5
## # ... with 5 variables: lat <dbl>, long <dbl>, year <int>, month <int>, n <int>
```

the primary key for `atmos` is `lat`, `long`, `year`, `month`.

Exercise 2 What is the relationship between the Batting, Master, and Salaries tables in the Lahman package? What are the keys for ech dataset and how do they relate to each other?

```
## [1] playerID n
## <0 rows> (or 0-length row.names)
```

```
## [1] playerID teamID   yearID   n
## <0 rows> (or 0-length row.names)
```

As we know from above, the primary key for `Batting` is `playerID`, `stint`, and `yearID`. The primary key for `Master` is `playerID`, and for `Salaries` it is `playerID`, `teamID`, and `yearID`.

`playerID` in `Master` has a 1 to many relationship in Salaries and Batting.

Mutating Joins

Inner Joins connect pairs of observations when their keys are equal. Unmatched rows will not be included in the result.

Outer Joins keep observations that appear in at least one of the tables:

- `left_join(x, y)` keeps all observations in `x`
- `right_join(x, y)` keeps all observations in `y`
- `full_join(x, y)` keeps all observations

Natural join is the default and uses all variables with the same name in both tables.

Using `by = "variable_name` joins by specified variables only.

You can also use `by = c("variable_a" = "variable_b")` to use a key with a different name in each table

Exercise 3 Use an appropriate join to add a column containing the airline name to the flights dataset.

Be sure to put the carrier code and name in the first two columns of the result. **Save as `flights2`**

Exercise 4 Use an appropriate join to add the airport name to the `flights2` dataset.

The codes and names of the airports are in the airports dataset of `nycflights13` package.

```
## # A tibble: 336,776 x 22
##    carrier name  origin airport_origin dest  airport_dest  year month   day
##    <chr>   <chr> <chr>  <chr>          <chr> <chr>        <int> <int> <int>
##  1 UA      Unit~ EWR    Newark Libert~ IAH   George Bush~  2013     1     1
##  2 UA      Unit~ LGA    La Guardia     IAH   George Bush~  2013     1     1
##  3 AA      Amer~ JFK    John F Kenned~ MIA   Miami Intl    2013     1     1
##  4 B6      JetB~ JFK    John F Kenned~ BQN   <NA>          2013     1     1
##  5 DL      Delt~ LGA    La Guardia     ATL   Hartsfield ~  2013     1     1
##  6 UA      Unit~ EWR    Newark Libert~ ORD   Chicago Oha~  2013     1     1
```

```
##  7 B6      JetB~ EWR    Newark Libert~ FLL   Fort Lauder~  2013     1     1
##  8 EV      Expr~ LGA    La Guardia     IAD   Washington ~  2013     1     1
##  9 B6      JetB~ JFK    John F Kenned~ MCO   Orlando Intl  2013     1     1
## 10 AA      Amer~ LGA    La Guardia     ORD   Chicago Oha~  2013     1     1
## # ... with 336,766 more rows, and 13 more variables: dep_time <int>,
## #   sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
## #   sched_arr_time <int>, arr_delay <dbl>, flight <int>, tailnum <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Filtering Joins

You can use *filter joins* to filter observations from one tibble based on whether or not they match an observation in another tibble.

A *semi join* between `x` and `y`:

- Keeps all observations in `x` that have a match in `y`
- Is useful for matchinf filtered summary back to original row

An *anti join* between `x` and `y` is the inverse of the semi join:

- *Drops* all observations in `x` that have a match in `y`
- Useful for diagnosing join mismatches

Exercise 5 Compute the average delay by destination, then join the airports data frame so you can show the spatial distribution of delays.

- Use the size or color of the points to display the average delay for each airport.
- Add the location of the origin and destination (lat and lon) to flights
- Compute the average delay by destination

Set Operations

`dplyr` provides functions for performing standard set operations:

- `intersect(x, y)` returns the intersection of datasets `x` and `y`
- `union(x, y)` returns the union of datasets `x` and `y`
- `setdiff(x, y)` returns set members in `x` that are not in `y`.

Exercise 6 Use a set operation function to find which airport codes from flights are not in the airports datset