Zeyd Khalil HW9, October 22, 2020

```
library(tidyverse)
```

```
## -- Attaching packages -------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ----------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(nycflights13)
d <- data(package = "nycflights13")
library(Lahman)
library(babynames)
library(nasaweather)
```

```
##
## Attaching package: 'nasaweather'

## The following object is masked from 'package:dplyr':
##
##     storms
```

Exercise 1 - Identify the primary keys in the following datasets (1.5 points: 1/2 pt. for each dataset). Be sure to show that you have the primary key by showing there are no duplicate entries.

- Lahman::Batting
- babynames::babynames
- nasaweather::atmos

```
BattingKey <- Batting %>%
    mutate(PrimaryKey = row_number()) %>%
    select(PrimaryKey, everything())

BattingKey %>% count(PrimaryKey) %>% filter(n>1)
```

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

> I found out that the Batting table does not have a Primary Key, because all the variables repeat themselves more than once. What I did was I added a surrogate key to Batting using mutate and row_number().

```
babynames %>% count(year, sex, name, n) %>% filter(nn>1)
```

```
## Storing counts in ‘nn‘, as ‘n‘ already present in input
## i Use ‘name = "new_name"‘ to pick a new name.
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: year <dbl>, sex <chr>, name <chr>, n <int>, nn <int>
```

> The Primary Key in babynames is the combination of year, sex, name and n. When there's more than one Primary Key, we usually call this a Composite Key.

```
atmos %>% count(lat, long, year, month) %>% filter(n>1)
```

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

> The Primary Key in atmos is the Composite Key that has lat, long, year, and month.

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

> The relationship between the Batting, Master, and Salaries tables is that they all have a playerID variable.

Exercise 3 - (2 points) 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 so we can see them. Save the result as flights2.

```
flights2 <- full_join(airlines, flights)
```

```
## Joining, by = "carrier"
```

```
flights2
```

```
## # A tibble: 336,776 x 20
##    carrier name   year month   day dep_time sched_dep_time dep_delay arr_time
##    <chr>   <chr> <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1 9E      Ende~  2013     1     1      810            810         0     1048
##  2 9E      Ende~  2013     1     1     1451           1500        -9     1634
##  3 9E      Ende~  2013     1     1     1452           1455        -3     1637
##  4 9E      Ende~  2013     1     1     1454           1500        -6     1635
##  5 9E      Ende~  2013     1     1     1507           1515        -8     1651
##  6 9E      Ende~  2013     1     1     1530           1530         0     1650
##  7 9E      Ende~  2013     1     1     1546           1540         6     1753
##  8 9E      Ende~  2013     1     1     1550           1550         0     1844
##  9 9E      Ende~  2013     1     1     1552           1600        -8     1749
## 10 9E      Ende~  2013     1     1     1554           1600        -6     1701
## # ... with 336,766 more rows, and 11 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Exercise 4 - Use an appropriate join to add the airport name to the flights2 dataset you got in exercise 3. The codes and names of the airports are in the airports dataset of the nycflights13 package. Put the carrier and carrier name first followed by the destination and destination name, then everything else.

```r
flights3 <- flights2 %>% left_join(airports, c("dest" = "faa"))

flights3
```

```
## # A tibble: 336,776 x 27
##    carrier name.x  year month   day dep_time sched_dep_time dep_delay arr_time
##    <chr>   <chr>  <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1 9E      Endea~  2013     1     1      810            810         0     1048
##  2 9E      Endea~  2013     1     1     1451           1500        -9     1634
##  3 9E      Endea~  2013     1     1     1452           1455        -3     1637
##  4 9E      Endea~  2013     1     1     1454           1500        -6     1635
##  5 9E      Endea~  2013     1     1     1507           1515        -8     1651
##  6 9E      Endea~  2013     1     1     1530           1530         0     1650
##  7 9E      Endea~  2013     1     1     1546           1540         6     1753
##  8 9E      Endea~  2013     1     1     1550           1550         0     1844
##  9 9E      Endea~  2013     1     1     1552           1600        -8     1749
## 10 9E      Endea~  2013     1     1     1554           1600        -6     1701
## # ... with 336,766 more rows, and 18 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>,
## #   name.y <chr>, lat <dbl>, lon <dbl>, alt <dbl>, tz <dbl>, dst <chr>,
## #   tzone <chr>
```

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

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

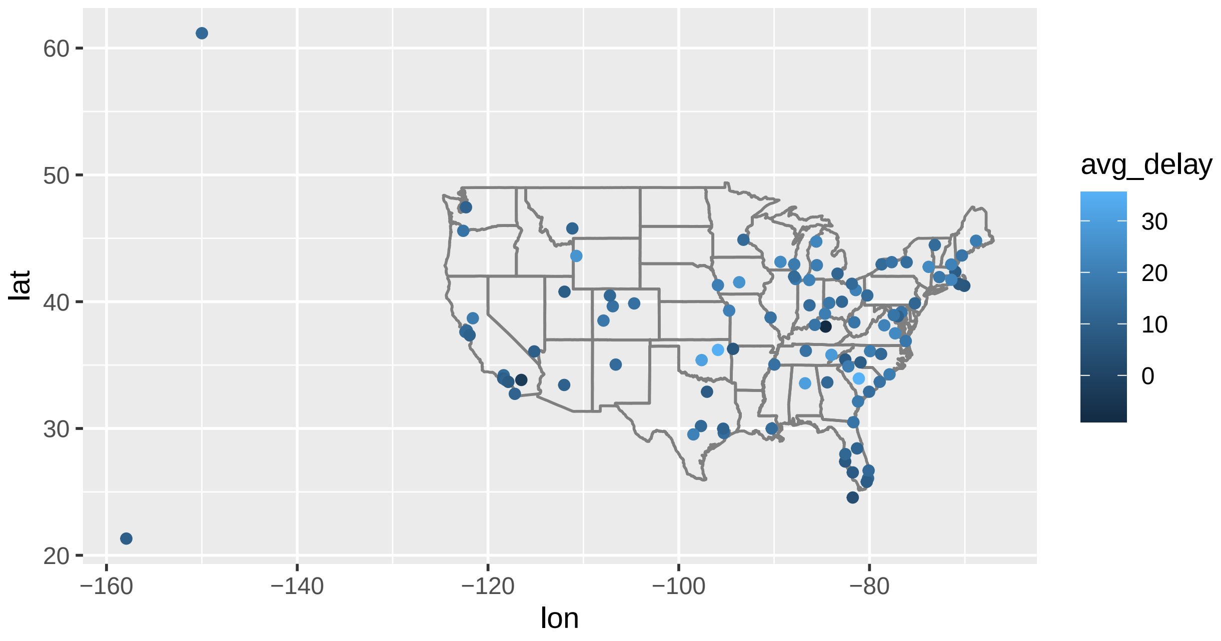
```
flights3 <- flights3 %>% group_by(dest) %>% summarize(avg_delay = mean(dep_delay, na.rm = TRUE)) %>% sel
```

```
## ‘summarise()‘ ungrouping output (override with ‘.groups‘ argument)
```

```
flights4 <- flights3 %>% full_join(airports, c("dest" = "faa"))

flights4 <- flights4 %>% na.omit()

flights4 %>% ggplot(aes(lon, lat, color = avg_delay)) + borders("state") + geom_point() + coord_quickmap
```



Exercise 6 -

```
flightsOrigin <- flights %>% select(origin)

airportsFaa <- airports %>% select(faa) %>% rename(c("origin" = "faa"))

setdiff(flightsOrigin, airportsFaa)
```

```
## # A tibble: 0 x 1
## # ... with 1 variable: origin <chr>
```

This shows that there are no airport codes from the flights dataset that are not in the airports dataset

```r
setdiff(airportsFaa, flightsOrigin)
```

```
## # A tibble: 1,455 x 1
##    origin
##    <chr>
##  1 04G
##  2 06A
##  3 06C
##  4 06N
##  5 09J
##  6 0A9
##  7 0G6
##  8 0G7
##  9 0P2
## 10 0S9
## # ... with 1,445 more rows
```

This, however, shows that there are 1445 airport codes in the airports dataset that are not in the flights dataset.