CPSC 260
Spring 2022

Instructions: This assignment will give you a little more practice using what you have learned about data manipulation using the dplyr package, *specifically joining data sets*. It will also give you a chance to review previous concepts and pick up some new code!

For each problem copy and paste your R code from the R script file window into this Word document. Please use a color other than black for your R code. I do not need the code's output, unless otherwise specified, but don't forget to answer any additional questions presented in the problems.

You will upload this Word document to Bb when finished.

Begin by downloading the *fueleconomy* and *maps* packages. Then load in the *tidyverse*, *nycflights13, Lahman, babynames, fueleconomy,* and *maps* libraries.

Next, read 13.1 through 13.4 from Chapter 13: Relational Data in the R for Data Science book. As you read you will do the exercises specified below. I have provided notes for some problems to help you or to clarify the problem. Please read these carefully.

Section 13.2.1

- **#1** ~Hint: Drawing the routes requires the latitude and longitude of the origin and the destination airports of each flight.

The 'flights' and 'airports' tables. 'flights' has the origin and destination airport of each flight. The 'airports' table has the longitude and latitude of each airport. You would need to combine 'flights' and 'airports'

- **#3**

#If weather was included for all airports in the US, then it would provide

#the weather for the destination of each flight

Section 13.3.1

- **#2 parts 1, 2, and 4**
  ~Note: You are looking for a variable or set of variables that allow(s) you to uniquely identify observations. For each part verify that you have found the correct answer using *count()*. See the book's examples for this.

  ~BUT keep this in mind while you work – In section 13.6, the book states that when you join tables you should "start by identifying the primary key in each table. ***You should usually do this based on your understanding of the data, not empirically by looking for a combination of variables that give a unique identifier.*** If you just look for variables without thinking about what they mean, you might get (un)lucky and find a combination that's unique in your current data but the relationship might not be true in general."

  #1) Each player is assigned a unique code (`playerID`). All of the information in different tables relating to that player is tagged with his `playerID`. playerID and yearID are not a primary key because a player can play on more than 1 team per year if they were traded or something.

```
Lahman::Batting %>%
  count(playerID, yearID, stint) %>%
  filter(n > 1) %>%
  nrow()
```

#2) Year, sex, and name are keys.

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

The 'year' and 'name' columns are not a primary key.

#3) the 'id' column is the primary key.

```
fueleconomy::vehicles %>%
  count(id) %>%
  filter(n > 1) %>%
  nrow()
```

- Refer to **#2 part 5** – Explain why there is no primary key. Then add a surrogate key to the data set using the code below. Explain the purpose of adding the surrogate key.

  Diamonds2 <- mutate(diamonds, id = row_number())
  Diamonds2

There is not a combination of variables that identifies each observation. The surrogate key is a unique identifier

- **#3** ~Just answer the first part: Draw a diagram illustrating the connections between the *Batting*, *People*, and *Salaries* tables in the *Lahman* package.

Batting	People	Salaries
playerID →	PlayerID →	PlayerID
yearID		yearID
Stint (order of plate appearances in a season)		teamID

CPSC 260
Spring 2022

Section 13.4.6

- **#1** ~ Start by using the correct *dplyr* commands to compute the average delay by destination. Then join the result to the *airports* data set using *inner_join()*. Call this new joined data set *avg_dest_delays*. You can then modify the book's code to create your map! [I've provided the correct code below.]

```
avg_dest_delays %>%
    ggplot(aes(lon, lat, colour = delay)) +
    borders("state") +
    geom_point() +
    coord_quickmap()
```

```
avg_dest_delays =
  flights %>%
  group_by(dest) %>%
  summarise(delay = mean(arr_delay, na.rm = T)) %>%
  inner_join(airports, by = c(dest = 'faa'))
```

- **#3** ~ In the *planes* data set, year represents the year that the plane was manufactured. In the *flights* data set, year represents the year of the flight. You can join the two data sets and then use these two pieces of information to figure out the age of the plane at the time of the flight. After you have this, you can create a graph (using ggplot2) to explore the relationship between delays and year of manufacture. Make sure to interpret the graph and answer the original question.

```
planes_flights_joined = inner_join(flights, select(planes, tailnum,
                                                   plane_year = year), #plane_year is the year plane was built
                                   by = 'tailnum') %>%
  mutate(age = year - plane_year) %>%
  filter(!is.na(age)) %>%
  group_by(age) %>%
  summarise(dep_delay_avg = mean(dep_delay, na.rm = T)) #NA values remove is True

ggplot(planes_flights_joined, aes(x=age, y = dep_delay_avg))+
  geom_point()+
  labs(x = 'Age of plane', y = 'Average departure delay')
```

The average delay goes up for about the first 9 ish years of the plane age then drops off the next 10 years and then is all over the place after that

- **#4** ~ Challenge! See if you can figure this one out. Maybe you can treat it similarly to #3? Be thorough in your data exploration and answer.

```
flights_weather_delay = flights %>%
  inner_join(weather, by = c(
    'origin' = 'origin',
    'year' = 'year',
    'month' = 'month',
    'day' = 'day',
    'hour' = 'hour'

flights_weather_delay %>%
    group_by(precip) %>%
    summarise(delay=mean(dep_delay, na.rm = T)) %>%
    ggplot(aes(x=precip, y = delay))+
    geom_point()
```

Between about 0 - .17 of precipitation there seems to be a positive correlation, as the precipitation goes up so does the delay. After that it is scattered

Extra Credit: Read section 13.5 and then answer #2 from 13.5.1.
To receive the full amount of extra credit you must answer this question in two different ways:

A. Using a *semi_join()*. See their example!

B. Using *group_by()* with *mutate()* instead. No joins!
[Note: We've always used *group_by()* with *summarise()*, so this is new!]
~Hint: You'll need to filter out flights that are missing a tail number, otherwise they will all be treated as a single plane.