**12.2.1**

1. In all the tables the variables are being displayed in the columns and each of the observations are shown on the rows.
   1. In table2 the variables are country, year, type and count.
   2. In table3 the variables are country, year and rate
   3. In table4a and table4b it is a (3 x 3) tibble and the variables are country, the year ‘1999’ and ‘2000’. These tables are not tidy because the year variable should be stacked together not as separate variables.
   4. In table5 the variables are country, century, year and rate
2. Compute the rate for table2, table4a and table4b
   1. Table2 🡪 finding the number of TB cases per country per year  
      table2\_cases <- filter(table2, type == "cases")[["count"]]

table2\_country <- filter(table2, type == "cases")[["country"]]

table2\_year <- filter(table2, type == "cases")[["year"]]

table2\_population <- filter(table2, type == "population")[["count"]]

table2\_fixed <- tibble(country = table2\_country,

year = table2\_year,

rate = table2\_cases / table2\_population)

* 1. Table4a and Table4b: use a tibble function to find the cases per year in each country. It looks like one table is cases per year in each country and one table is population per year per country  
     tibble(country = table4a[["country"]],

`1999` = table4a[["1999"]] / table4b[["1999"]],

`2000` = table4b[["2000"]] / table4b[["2000"]])

* 1. I would say that they both have advantages. Creating a tibble function for the two separate tables with a small dataset is easy, but could get out a hand with thousands of observations. Overall, the strategy in table2 is easier and can be applied across larger datasets with more easy.

1. First we would need to separate the population and cases from the observations/rows and instead create two separate variable columns by filtering to cases.  
    filter(type == "cases") %>%

ggplot(aes(year, count)) +

geom\_line(aes(group = country), colour = "grey50") +

geom\_point(aes(colour = country))

**12.3.3**

1. They are different because the column type is not the same. In one version year is an integer and in the second example it’s a character vector. The convert argument is used to change character to the appropriate type
2. For this code to run you need to include ` ` around the year because it is numeric
3. The tibble doesn’t work because there are two observations that are both Phillip Woods and age. They are not going to be classed as separate observations.
4. Using a gather function will be most helpful and we can use the following function: gather(preg, sex, count, male, female)

**12.4.3**

1. **Extra** controls what happens when there are too many pieces. “warn” = emit a warning and drop extra values. “drop” = drop any extra values without a warning. “merge” = only splits at most length (into) times. **Fill** controls what happens when there are not enough pieces. “warn” = emit a warning and fill from the right. “right” = fill with missing values on the right. “left” = fill with missing values on the left.
   1. **Merge** would probably work best for the first dataset because then it would show the two letters on one observation without any errors
2. The remove argument, if TRUE will remove the input column from the output data frame. Setting it to FALSE will create a new variable, but keep the old one.
3. Extract aligns data using a regular expression to find groups. Unite is simple because it is many columns to one using the sep. Separate there are one to many as well.

**12.5.1**

1. In **spread** the fill argument sets the value to replace NA. In **complete** the fill argument is named list. Both cases replace implicit and explicit missing values
2. The direction argument will tell which way to replace all the NA values with the unique proximity value.

**12.6.1**

1. I think in terms of this dataset using na.rm =TRUE will be fine, but I think it’s important to distinguish between explicit and implicit missing values.
2. When rows begin with newrel the sexage isn’t there so we end up shifting all our data to the right in the wrong columns.
3. We can select the 3 rows that we want: country, iso2 and iso3, then we can filter to distinct values and group on the country. Therefore, we see that there are no misaligned observations. They all have unique country, iso2 and iso3 relationships  
    select (who3, country, iso2, iso3) %>%  
    distinct() %>%  
    group\_by(country)
4. It’s going to be nearly impossible to graph this since there are too many countries to graph on one plane. This equation leads to a vert messy output but it satisfies the problem.  
     
   who5 %>%

group\_by(country, year, sex) %>%

filter(year > 1995) %>%

summarise(cases = sum(cases)) %>%

unite(country\_sex, country, sex, remove = FALSE) %>%

ggplot(aes(x = year, y = cases, group = country\_sex, color = sex)) +

geom\_line()

**13.2.1**

1. You would need origin and dest for the flights table. And then the longitude and latitude from the airports table would help determine the distance between start and finish
2. Origin could be matched with the faa or maybe the name in the airport table
3. You can match all the time functions to each other, including: year, month, day, hour.
4. You could potentially add a table with all the holidays and special dates through the year and join them on the original tables with year, month and day.

**13.3.1**

1. Adding a unique flight identifier is helpful to tie everything together. “unique\_id”. Sorting the data with arrange first will make it more palatable when using a glimpse function. Glimpse is helpful because it flips the rows and columns and makes it more digestable:  
   flights %>%  
    arrange(year, month, day, sched\_dep\_time, carrier, flight) %>%  
    mutate(unique\_id = row\_number()) %>%  
    glimpse()
2. Identifying the primary keys from the various datasets:
   1. Lahman::Batting 🡪 playerID, yearID and stint because each unique player had a unique year and unique stint. All the other variables are numeric values
   2. babynames::babynames 🡪 year, sex, name
   3. nasaweather:: atmos 🡪 lat, long, year, month 🡪 you can also filter down to a more unique primary id if there is day or hour in the dataset. There is not though
   4. fueleconomy:: vehicles 🡪 the variable id appears to be the primary id.
   5. ggplot2:: diamonds 🡪 no primary key
3. It seems that you can tie together the tables with the playerID, lgID and teamID because they all appear in the different tables.

**13.4.6**

1. This calculates the average delay by destination. I joined the table flights and airports on the dest and faa to know the airport rather than the abbreviation:  
     
   average\_delay <-

flights %>%

group\_by(dest) %>%

summarise(delay = mean(arr\_delay, na.rm = TRUE)) %>%

inner\_join(airports, by = c(dest = "faa"))  
I used this equation to show the delay by destination. I figured using size of point would be the best view of average delay  
  
average\_delay %>%

ggplot(aes(lon, lat, size = delay)) +

borders("state") +

geom\_point()+

coord\_quickmap()

1. I left joined the flights table with the airport table which houses all the lat and lon info:  
    flights %>%

left\_join(airports, by = c(dest = "faa")) %>%

left\_join(airports, by = c(origin = "faa")) %>%

head()

1. No there isn’t a change at all. If anything, the time of delay trends downward with the increasing age of the plane.   
   age <-

planes %>%

mutate(age = 2013 - year) %>%

select(tailnum, age)  
**Visualize**flights %>%

inner\_join(age, by ="tailnum") %>%

group\_by(age) %>%

summarise(delay = mean(dep\_delay)) %>%

ggplot(aes(x = age, y = delay)) +

geom\_point() +

geom\_line()

1. Code I used:  
     
   glimspse(weather)

weather\_fly <-

flights %>%

inner\_join(weather, by = c("origin" = "origin",

"year" = "year",

"month" = "month",

"day" = "day",

"hour" ="hour"))

weather\_fly %>%

group\_by(wind\_speed) %>%

summarise(delay = mean(dep\_delay, na.rm = TRUE)) %>%

ggplot(aes(x = wind\_speed, y = delay)) +

geom\_line() +

geom\_point()  
  
This came to the conclusion that higher wind speeds correlate with higher average delays. For some reason there was an outlier in the data where wind speed was over 1000, but I wasn’t able to remove that from the dataset

1. It appears that there were high levels of delays in the mid-southwest region on July 13th, 2013. Per Wikipedia, there were “derechos” occurring at that time in the region. A derecho is a widespread, long-lived, straight-line wind storm that is associate with a fast-moving group of severe thunderstorms. This corroborates my story in Question #4 where I said that higher wind speeds lead to greater average delays.

**13.5.1**

1. There is no talinum data from AA and MQ. I’m assuming this data isn’t collected
2. Filter flights to only show flights with planes that flew <100 flights  
     
   planes2 <-

filter(flights) %>%

group\_by(tailnum) %>%

count() %>%

filter(n > 100)

flights %>%

semi\_join(planes2, by = "tailnum")

1. Joining vehicles with common in order to get the id, make, model, year and class all together   
     
   fueleconomy::vehicles %>%

semi\_join(fueleconomy::common, by = c("make", "model"))

1. This question has been giving me a lot of trouble. I hope to go over it in the breakout session on Thursday morning.
2. That first function shows the flights that go to airports not regulated or listed by the FAA. Maybe they are international flights or flights to not popularly known flights. Second equation string is showing airports with no data for that string
3. There doesn’t appear to be any relationship between plane and airline because they are constantly sold or a bigger airline buys out a smaller one (for example, Alaska Airlines and Virgin America becoming Alaska Airlines)