Homework 6

[Ra-Zakee Muhammad]

Due THURSDAY 10/21/2021

Classmates/other resources consulted: [type answer here]

```
library(tidyverse)
library(nycflights13)
```

Question 1 (12 points)

> **This question deals with the Lahman package, which has several tibbles related to baseball. Install it and load it before beginning this question (you're likely going to have to comment out your installation command to make this file knit):**

```
#install.packages("Lahman")
library(Lahman)
```

a. **What column makes a primary key in the People table? Explain how you know this is a valid key.**

```
Lahman::People %>% count(playerID) %>% filter(n>1)
```

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

```
Lahman::People %>% distinct(playerID) %>% nrow()
```

```
## [1] 20093
```

```
Lahman::People %>% nrow()
```

```
## [1] 20093
```

I know that playerID makes a primary key because no two players have the same iD, the number of distinct ids is equal to the number of players in total. So this column alone makes up the primary key.

b. **Explain why the pair of columns { nameFirst, nameLast } aren't a key for the People table. Give an example of specific entries in the table that support your explanation.**

{ nameFirst, nameLast } isnt unique for each player which means that referencing those two columns alone wont give us access to the complete table. For instance there are 5 players named Bob Smith but they all have different player ids meaning we can reference them seperatly.

c. **Is the column you identified in part (a) a primary key in the Batting table? Explain why or why not.**

It isnt a primary key because when we examine groupings of player id in batting we get significantly fewer groups than rows meaning that some player ids are used for multiple rows. this means referencing playerID wont give us access to all distinct rows therefore its not a primary key.

d. **Is the column you identified in part (a) a foreign key in the Batting table? Explain why or why not.**

it is a foreign key because it is a factor that makes up the primary key of another table but isnt the primary key of this table.

e. **Are there any players that appear in the Batting table but not in the People table? Show how you know.**

No, and I know because the following table is empty.

```
anti_join(distinct(select(Batting,playerID)), distinct(select(People,playerID)))
```

```
## Joining, by = "playerID"
```

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

Anti join examines all playerIDs which references individual players in the batting table and asks which are present that arnt also present in people table and there are no such players as indicated by the empty tibble

f. **Are there any players that appear in the People table but not in the Batting table? Show how you know.**

yes there are 195 players and I know because of the number of rows created by the following command.

```
anti_join( distinct(select(People,playerID)), distinct(select(Batting,playerID))) %>% nrow()
```

```
## Joining, by = "playerID"
```

```
## [1] 195
```

Anti join examines all playerIDs which references individual players in the people table and asks which are present that arnt also present in batting table and there are 195 such players as indicated by the above output

Question 2 (3 points)

> **Import the atmos data set, which is attached to this assignment in the atmos.csv file. What is the best set of columns to choose to serve as a primary key for this table? Explain how you know it is a valid key.**

the best combination is : {cloudmid, lat, ozone, surftemp, long, cloudhigh, cloudlow, month} because when you find all distinct combinations of these factors present in the data you get the exact number of observations in the original table.

Question 3 (5 points)

> **Explain why the diamonds data set doesn't meet the three assumptions we discussed in class on 10-12; be specific about which assumption(s) it violates. Then, modify the data set so that it meets all three assumptions.**

```
diamonds %>% distinct() %>% nrow()
```

```
## [1] 53794
```

but

```
diamonds %>% nrow()
```

```
## [1] 53940
```

We see that some of the rows are identical which is why the number of distinct rows isn't equal to the number of total rows which mean diamonds violates assumption 3 of distinct rows. Inorder to keep our data while making the rows distinct we will create a serrogate key

```
diamonds %>% mutate(id = row_number()) %>% select(id, everything())
```

```
## # A tibble: 53,940 x 11
##       id carat cut       color clarity depth table price     x     y     z
##    <int> <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1     1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2     2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3     3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4     4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5     5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6     6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7     7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8     8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9     9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10    10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # ... with 53,930 more rows
```

and this table satisfies all assumptions we discussed in class.

Question 4 (12 points)

Consider the following tibbles (do not modify these tibbles in any way)

a. **Join these tibbles according to species using an inner join. Which animal(s) appear in two different rows, which animal(s) appear only in one row, and which animal(s) don't appear in this tibble? Explain why this is.**

```r
October_Pets
```

```
## # A tibble: 10 x 4
##    name    species age_months arrival_day
##    <chr>   <chr>        <dbl>       <dbl>
##  1 Sparky  Dog             31           0
##  2 Fido    Dog             29           0
##  3 Fluffy  Cat             78           4
##  4 Lassie  Dog             98           0
##  5 Patches Cat            115           0
##  6 Spot    Dog              7          12
##  7 Socks   Cat              4          17
##  8 Buddy   Dog             15           0
##  9 Lizzie  Lizard           2           0
## 10 Tweety  Bird             6           2
```

```r
Pet_Locations
```

```
## # A tibble: 5 x 3
##   Location Species occupancy_limit
##   <chr>    <chr>             <dbl>
## 1 Room 1   Dog                  10
## 2 Room 2   Dog                   8
## 3 Room 3   Cat                  15
## 4 Room 4   Reptile              20
## 5 Room 5   Bird                 12
```

```r
inner_join(October_Pets, Pet_Locations, by = c( "species"= "Species"))
```

```
## # A tibble: 14 x 6
##    name    species age_months arrival_day Location occupancy_limit
##    <chr>   <chr>        <dbl>       <dbl> <chr>              <dbl>
##  1 Sparky  Dog             31           0 Room 1                10
##  2 Sparky  Dog             31           0 Room 2                 8
##  3 Fido    Dog             29           0 Room 1                10
##  4 Fido    Dog             29           0 Room 2                 8
##  5 Fluffy  Cat             78           4 Room 3                15
##  6 Lassie  Dog             98           0 Room 1                10
##  7 Lassie  Dog             98           0 Room 2                 8
##  8 Patches Cat            115           0 Room 3                15
##  9 Spot    Dog              7          12 Room 1                10
## 10 Spot    Dog              7          12 Room 2                 8
## 11 Socks   Cat              4          17 Room 3                15
## 12 Buddy   Dog             15           0 Room 1                10
## 13 Buddy   Dog             15           0 Room 2                 8
## 14 Tweety  Bird             6           2 Room 5                12
```

twice: Sparky, Fido, Lassie, Spots, Buddy Once: Tweety, Socks, Patches, Fluffy none: Lizzy

This is the case because of the way the inner join function takes in names from the two tibbles' columns. In the pets_location column there are 2 dogs listed, which means that when the inner join function asks if each dog in the october pets species factor is present in the pets location species factor the matching will happen twice resulting in doubles for dogs, singles for cats and birds. Lizards isnt a sting in the pets location species factor, so lizzie is not in the inner join.

b. **Joining these tables with a left_join rather than an inner_join results in a tibble with one more row than in part (a). Which additional row is present here and why?**

```
anti_join(left_join(October_Pets, Pet_Locations, by = c( "species"= "Species" )), inner_join(October_Pe
```

```
## Joining, by = c("name", "species", "age_months", "arrival_day", "Location", "occupancy_limit")

## # A tibble: 1 x 6
##   name   species age_months arrival_day Location occupancy_limit
##   <chr>  <chr>        <dbl>       <dbl> <chr>              <dbl>
## 1 Lizzie Lizard           2           0 <NA>                  NA
```

because of how to the left join works what is on the left is kept and what is on the right is either NA, matches, with whats on the left or is discarded. Lizzie is from the left and whats joined to that row are NA values since no "Lizards" are in the right tibble.

c. **Joining these tables with a right_join rather than an inner_join results in a tibble with one more row than in part (a). Which additional row is present here and why?**

```
anti_join(right_join(October_Pets, Pet_Locations, by = c( "species"= "Species" )), inner_join(October_P
```

```
## Joining, by = c("name", "species", "age_months", "arrival_day", "Location", "occupancy_limit")

## # A tibble: 1 x 6
##   name  species age_months arrival_day Location occupancy_limit
##   <chr> <chr>        <dbl>       <dbl> <chr>              <dbl>
## 1 <NA>  Reptile         NA          NA Room 4                20
```

because of how to the right join works what is on the right is kept and what is on the left is either NA, matches with whats on the left, or is discarded. reptile is from the right and whats joined to that row are NA values since no "reptiles" are in the left tibble.

d. **Joining these tables with a full_join rather than an inner_join results in a tibble with two more rows than in part (a). Which additional rows are present here and why?**

```
anti_join(full_join(October_Pets, Pet_Locations, by = c( "species"= "Species" )), inner_join(October_Pe
```

```
## Joining, by = c("name", "species", "age_months", "arrival_day", "Location", "occupancy_limit")
```

```
## # A tibble: 2 x 6
##   name   species age_months arrival_day Location occupancy_limit
##   <chr>  <chr>        <dbl>       <dbl> <chr>              <dbl>
## 1 Lizzie Lizard           2           0 <NA>                  NA
## 2 <NA>   Reptile         NA          NA Room 4                20
```

Full join is simply a combination of the left join and the right join which means that the rows that are created in both will be present together in the full join resulting in 2 extra rows.

Question 5 (12 points)

> **Consider the following two tibbles.**
>
> a. **Join these tibbles by the college column using a full_join. Explain why doing this join is probably a bad idea.**

```
full_join(campus_majors, campus_observations1, by = "college")
```

```
## # A tibble: 14 x 5
##    college major          num student year
##    <chr>   <chr>        <dbl> <chr>   <chr>
##  1 CMC     math            21 A       Freshman
##  2 CMC     math            21 B       Freshman
##  3 CMC     math            21 C       Junior
##  4 CMC     math            21 D       Junior
##  5 CMC     data science    14 A       Freshman
##  6 CMC     data science    14 B       Freshman
##  7 CMC     data science    14 C       Junior
##  8 CMC     data science    14 D       Junior
##  9 Scripps math             6 E       Sophomore
## 10 Scripps math             6 G       Senior
## 11 Scripps math             6 H       Senior
## 12 Scripps data science     8 E       Sophomore
## 13 Scripps data science     8 G       Senior
## 14 Scripps data science     8 H       Senior
```

This join isnt a good idea because it creates new data that isnt actually observed as well as removes the significance of the student factor by associating mutiple rows to each student. > b. **Explain why you have the number of rows that you do in your join in the previous part.**

The way that full join matches information it takes one row from the left and matches it with everything on the right that shares the specified factor value. There are 4 cmc entries so 4 new entries will be made in the joined table for each cmc in the left. 4x2 is 8 cmc entries. there are 3 scipps so 2x3 is 6 scripps entries.

> c. **Explain why you get the exact same tibble as in the previous parts whether you use full_join, right_join, left_join, or inner_join.**

```
left_join(campus_majors, campus_observations1, by = "college")
```

```
## # A tibble: 14 x 5
##    college major          num student year
##    <chr>   <chr>        <dbl> <chr>   <chr>
##  1 CMC     math            21 A       Freshman
##  2 CMC     math            21 B       Freshman
##  3 CMC     math            21 C       Junior
##  4 CMC     math            21 D       Junior
##  5 CMC     data science    14 A       Freshman
##  6 CMC     data science    14 B       Freshman
##  7 CMC     data science    14 C       Junior
##  8 CMC     data science    14 D       Junior
##  9 Scripps math             6 E       Sophomore
## 10 Scripps math             6 G       Senior
## 11 Scripps math             6 H       Senior
## 12 Scripps data science     8 E       Sophomore
## 13 Scripps data science     8 G       Senior
## 14 Scripps data science     8 H       Senior
```

The way that left join matches information it takes one row from the left and matches it with everything on the right that shares the specified factor value. There are 4 cmc entries on the right so 4 new entries will be made in the joined table for each cmc in the left. 4x2 is 8 cmc entries. there are 3 scipps entries on the right so 3x2 is 6 scripps entries.

```r
right_join(campus_majors, campus_observations1, by = "college")
```

```
## # A tibble: 14 x 5
##    college major          num student year
##    <chr>   <chr>        <dbl> <chr>   <chr>
##  1 CMC     math            21 A       Freshman
##  2 CMC     math            21 B       Freshman
##  3 CMC     math            21 C       Junior
##  4 CMC     math            21 D       Junior
##  5 CMC     data science    14 A       Freshman
##  6 CMC     data science    14 B       Freshman
##  7 CMC     data science    14 C       Junior
##  8 CMC     data science    14 D       Junior
##  9 Scripps math             6 E       Sophomore
## 10 Scripps math             6 G       Senior
## 11 Scripps math             6 H       Senior
## 12 Scripps data science     8 E       Sophomore
## 13 Scripps data science     8 G       Senior
## 14 Scripps data science     8 H       Senior
```

The way that right join matches information it takes one row from the right and matches it with everything on the left that shares the specified factor value. There are 2 cmc entries on the left so 2 new entries will be made in the joined table for each cmc in the right. 2x4 is 8 cmc entries. there are 3 scipps so 2x3 is 6 scripps entries.

d. **Suppose you also have the following tibble. Combine it with the campus_observations1 tibble in an appropriate way.**

```
rbind(campus_observations1,campus_observations2)
```

```
## # A tibble: 12 x 3
##    student college year
##    <chr>   <chr>   <chr>
##  1 A       CMC     Freshman
##  2 B       CMC     Freshman
##  3 C       CMC     Junior
##  4 D       CMC     Junior
##  5 E       Scripps Sophomore
##  6 G       Scripps Senior
##  7 H       Scripps Senior
##  8 V       CMC     Junior
##  9 W       CMC     Sophomore
## 10 X       Scripps Senior
## 11 Y       Scripps Freshman
## 12 Z       Scripps Senior
```

Question 6 (6 points)

> **Add to the flights data set the latitude and longitude of the origin airports, and the latitude and longitude of the destination airports. That is, each row should now have 4 more additional columns. Move your columns for origin, destination, and their latitudes and longitudes to the front of your data set, with the remaining columns displayed after them.**

```
left_join(left_join(flights,
    select(airports,
           origin_lat=lat,
           origin_lon = lon, faa),
    by = c("origin" = "faa")),
  select(airports, dest_lat=lat, dest_lon = lon, faa), by = c("dest"="faa")) %>%
  select(origin, dest, origin_lat, origin_lon, dest_lat, dest_lon, everything())
```

```
## # A tibble: 336,776 x 23
##    origin dest  origin_lat origin_lon dest_lat dest_lon  year month   day
##    <chr>  <chr>      <dbl>      <dbl>    <dbl>    <dbl> <int> <int> <int>
##  1 EWR    IAH         40.7      -74.2     30.0    -95.3  2013     1     1
##  2 LGA    IAH         40.8      -73.9     30.0    -95.3  2013     1     1
##  3 JFK    MIA         40.6      -73.8     25.8    -80.3  2013     1     1
##  4 JFK    BQN         40.6      -73.8     NA       NA    2013     1     1
##  5 LGA    ATL         40.8      -73.9     33.6    -84.4  2013     1     1
##  6 EWR    ORD         40.7      -74.2     42.0    -87.9  2013     1     1
##  7 EWR    FLL         40.7      -74.2     26.1    -80.2  2013     1     1
##  8 LGA    IAD         40.8      -73.9     38.9    -77.5  2013     1     1
##  9 JFK    MCO         40.6      -73.8     28.4    -81.3  2013     1     1
## 10 LGA    ORD         40.8      -73.9     42.0    -87.9  2013     1     1
## # ... with 336,766 more rows, and 14 more variables: dep_time <int>,
## #   sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
```

```
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

Question 7 (18 points)

The following command attaches plane information to the flights tibble, for all flights where the tail number appears in the planes tibble. There's over 284,000 such flights:

```
inner_join(flights, planes, by = "tailnum")
```

```
## # A tibble: 284,170 x 27
##    year.x month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##     <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1   2013     1     1      517            515         2      830            819
##  2   2013     1     1      533            529         4      850            830
##  3   2013     1     1      542            540         2      923            850
##  4   2013     1     1      544            545        -1     1004           1022
##  5   2013     1     1      554            600        -6      812            837
##  6   2013     1     1      554            558        -4      740            728
##  7   2013     1     1      555            600        -5      913            854
##  8   2013     1     1      557            600        -3      709            723
##  9   2013     1     1      557            600        -3      838            846
## 10   2013     1     1      558            600        -2      849            851
## # ... with 284,160 more rows, and 19 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>,
## #   year.y <int>, type <chr>, manufacturer <chr>, model <chr>, engines <int>,
## #   seats <int>, speed <int>, engine <chr>
```

a. (3 points) **When we remove the "by" argument, we get a tibble with fewer than 5000 rows. Explain what's happening here, and why these particular rows have been included in this tibble.**

```
inner_join(flights, planes)
```

```
## Joining, by = c("year", "tailnum")
```

```
## # A tibble: 4,630 x 26
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1    18     1846           1810        36     2156           2120
##  2  2013    10     1      647            655        -8      744            809
##  3  2013    10     1      652            652         0      921            954
##  4  2013    10     1      755            800        -5      954           1013
##  5  2013    10     1      813            820        -7     1050           1110
##  6  2013    10     1      925            930        -5     1025           1038
##  7  2013    10     1     1113           1120        -7     1215           1230
##  8  2013    10     1     1426           1429        -3     1535           1548
```

```
## 9  2013    10     1     1446           1450        -4     1635           1652
## 10  2013    10     1     1454           1455        -1     1751           1718
## # ... with 4,620 more rows, and 18 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>,
## #   type <chr>, manufacturer <chr>, model <chr>, engines <int>, seats <int>,
## #   speed <int>, engine <chr>
```

whats happening is that when the by parameter is absent the function joins by the only factors that the two tibbles share and these are Year and Tailnum. Year means two seperate things depending on which table you are looking at, for planes it is manufacture data and for flights it is flight date. Through the join we are only examining flights with planes manufactured in 2013 that also take flight in 2013. These rows are flights in 2013 using planes that are manufactured in 2013.

b. (3 points) **Are there any planes in the planes tibble that did not do any flights out of NYC in 2013? Explain how you know.**

```
anti_join(planes, flights, "tailnum")
```

```
## # A tibble: 0 x 9
## # ... with 9 variables: tailnum <chr>, year <int>, type <chr>,
## #   manufacturer <chr>, model <chr>, engines <int>, seats <int>, speed <int>,
## #   engine <chr>
```

Anti join examines the planes data frame and questions which tailnumbers are present that are not also present in the flights data frame. There are now planes in the planes data frame that are not also in the flights data frame.

c. (3 points) **What are the three most popular manufacturers of planes in the planes tibble?**

```
planes %>% count(manufacturer) %>% arrange(desc(n)) %>% head(3)%>% select(manufacturer)
```

```
## # A tibble: 3 x 1
##   manufacturer
##   <chr>
## 1 BOEING
## 2 AIRBUS INDUSTRIE
## 3 BOMBARDIER INC
```

d. (3 points) **Of the flights whose tailnum appears in the planes tibble, what are the three most popular manufacturers? (Hint: The answer will be different from the previous part)**

```
inner_join(flights, planes, "tailnum") %>% count(manufacturer) %>% arrange(desc(n)) %>% head(3) %>% sel
```

```
## # A tibble: 3 x 1
##   manufacturer
##   <chr>
## 1 BOEING
## 2 EMBRAER
## 3 AIRBUS
```

e. (6 points) **Does the manufacturer of a plane affect the average departure delay of a flight? Group your flights (that have tailnums appearing in planes) by the manufacturer of the plane, and compute the average departure delay for each manufacturer. Only consider manufacturers with at least 1000 flights. Explain your conclusions about whether there is a relationship between a plane's manufacturer and its average departure delay by referencing the tibble produced.**

```
inner_join(flights, planes, "tailnum") %>% filter(!is.na(dep_delay)) %>% group_by(manufacturer) %>% summ
```

```
## # A tibble: 9 x 3
##   manufacturer                  group_size avg_dep_del
##   <chr>                              <int>       <dbl>
## 1 AIRBUS                             47009       11.4
## 2 AIRBUS INDUSTRIE                   40753       10.2
## 3 BOEING                             82524       11.7
## 4 BOMBARDIER INC                     27588       17.5
## 5 CANADAIR                            1492       18.3
## 6 EMBRAER                            63783       16.8
## 7 MCDONNELL DOUGLAS                   3865        8.34
## 8 MCDONNELL DOUGLAS AIRCRAFT CO       8864       12.3
## 9 MCDONNELL DOUGLAS CORPORATION       1251       12.6
```

From the table produced we can strongly infer that manufacturer influences departure delay time because if we examine group sizes we see that boing has 82542 flights and airbus has 47009 flights (and very similar departure delay times around 11.5), but embraer has 63783 flights which falls right in between the previous two airlines, but its departure time average is almost 17 minutes. we see that at least in the case of embraer, it has significantly higher departure delay times.

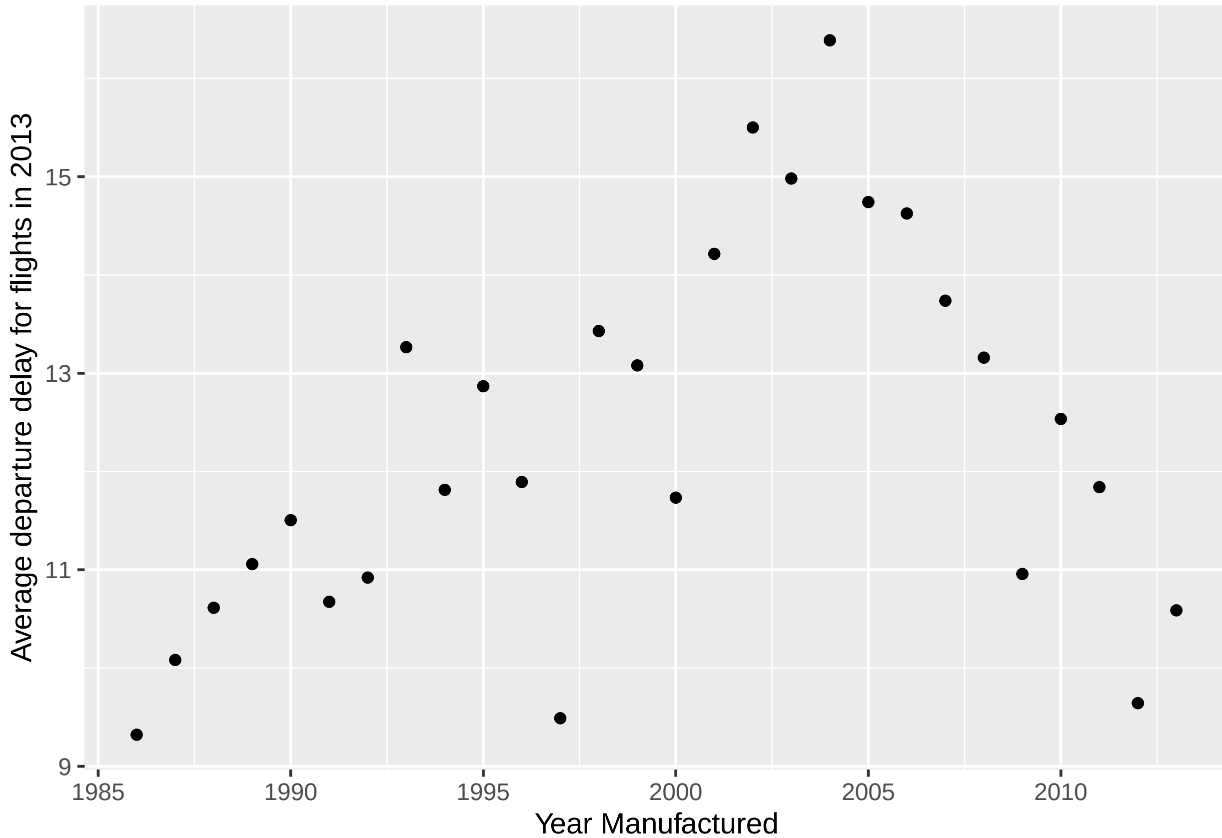
Question 8 (8 points)

Does the year a plane was built affect the average departure delay of a flight? Follow similar steps as in the previous question. You can restrict your attention to flights whose tailnum appears in planes, and years of manufacture for which there were at least 1000 flights. Make a plot of year of manufacture vs. average departure delay, and explain your conclusion by referencing this plot (and, if you'd like, referencing any tibbles produced.)

Hint: What is the name of the year of manufacture column in your joined tibble? It may not be what you think.

```
inner_join(flights, planes, "tailnum") %>%
  filter(!is.na(dep_delay)) %>%
  group_by(year.y) %>%
  summarise(group_size =n(), avg_dep_del = mean(dep_delay)) %>%
  filter(group_size >= 1000) %>%
  ggplot(mapping = (aes(x = year.y, y = avg_dep_del))) +
  geom_point() + xlab("Year Manufactured") +ylab("Average departure delay for flights in 2013")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Planes manufactured from 1985 up to 2005 seem to have an increasing departure delay and this comes to a peak around 2004 where after this years the average departure delay falls again. The year a plane was built definitely impacts in some way the departure delay of that plane because of the trends present in this graph. There are few outliers but for the most part before 2004 the older the plane the lower the average departure delay and after 2004 the younger the plane the lower the average departure delay.

Question 9 (9 points)

a. **In Homework 3, we filtered the flights data set to only contains flights with tailnums that made at least 100 non-canceled flights out of a NYC airport. Here's the code from the Homework 3 solutions:**

```
not_canceled <- flights %>% filter(!is.na(dep_time))
not_canceled %>% group_by(tailnum) %>% filter(n() >= 100)
```

```
## # A tibble: 223,197 x 19
## # Groups:   tailnum [1,210]
##      year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##     <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##   1  2013     1     1      517            515         2      830            819
##   2  2013     1     1      533            529         4      850            830
##   3  2013     1     1      544            545        -1     1004           1022
##   4  2013     1     1      554            558        -4      740            728
##   5  2013     1     1      555            600        -5      913            854
```

```
## 6 2013 1 1 557 600 -3 709 723
## 7 2013 1 1 557 600 -3 838 846
## 8 2013 1 1 558 600 -2 849 851
## 9 2013 1 1 558 600 -2 853 856
## 10 2013 1 1 558 600 -2 923 937
## # ... with 223,187 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Create the same tibble, but using a join that we've learned this week and any extra tibbles you may need to make. Be sure you're only working with the not_cancelled flights.

```
tail_numbers <- not_canceled %>% count(tailnum) %>% filter(n >= 100) %>% select(tailnum)
```

```
not_canceled %>% semi_join(tail_numbers, by = "tailnum")
```

```
## # A tibble: 223,197 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      544            545        -1     1004           1022
##  4  2013     1     1      554            558        -4      740            728
##  5  2013     1     1      555            600        -5      913            854
##  6  2013     1     1      557            600        -3      709            723
##  7  2013     1     1      557            600        -3      838            846
##  8  2013     1     1      558            600        -2      849            851
##  9  2013     1     1      558            600        -2      853            856
## 10  2013     1     1      558            600        -2      923            937
## # ... with 223,187 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

b. **Filter the non-canceled flights to only include flights along the 50 most popular routes, where a route consists of both the origin airport and the destination airport. Use a join we've learned this week.**

```
routes_tibble <- not_canceled %>% count(origin, dest) %>% arrange(desc(n)) %>% head(50) %>% select(orig
```

```
 not_canceled %>% semi_join(routes_tibble, by= c("origin", "dest"))
```

```
## # A tibble: 211,651 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      554            600        -6      812            837
##  5  2013     1     1      554            558        -4      740            728
##  6  2013     1     1      555            600        -5      913            854
```

```
## 7 2013 1 1 557 600 -3 838 846
## 8 2013 1 1 558 600 -2 753 745
## 9 2013 1 1 558 600 -2 853 856
## 10 2013 1 1 558 600 -2 924 917
## # ... with 211,641 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

c. **Filter the non-canceled flights to only include flights along the 50 routes with the largest average arrival delays.**

```
routes_tibble_c <- not_canceled %>%
  group_by(origin, dest) %>%
  summarise(avg_arr_del = mean(arr_delay), how_many =  n()) %>%
  arrange(desc(avg_arr_del)) %>%
  head(50)
```

```
## `summarise()` has grouped output by 'origin'. You can override using the `.groups` argument.
```

```
not_canceled %>% semi_join(routes_tibble_c, by= c("origin", "dest"))
```

```
## # A tibble: 9,374 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      629            630        -1      721            740
##  2  2013     1     1      743            749        -6     1043           1054
##  3  2013     1     1      831            835        -4     1021           1039
##  4  2013     1     1      857            900        -3     1516           1530
##  5  2013     1     1      909            810        59     1331           1315
##  6  2013     1     1      913            918        -5     1346           1416
##  7  2013     1     1     1059           1100        -1     1201           1215
##  8  2013     1     1     1150           1156        -6     1302           1314
##  9  2013     1     1     1208           1158        10     1540           1502
## 10  2013     1     1     1315           1317        -2     1413           1423
## # ... with 9,364 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Question 10 (6 points)

You run an animal shelter, and have the following data about the pets that were in your shelter in the month of October. Arrival_day is the day of the month the animal arrived at the shelter, and 0 means the animal was already in the shelter at the start of the month.

a. **You want to make a tibble consisting of two columns, the first containing all the names of the pets in your shelter that were adopted in October, and the second containing their species. First, do this using a mutating join (inner_join, left_join, right_join, or full_join) and whatever other transformations are necessary.**

```r
inner_join(October_Adoptions, October_Pets, by = "name") %>% select(name, species)
```

```
## # A tibble: 4 x 2
##   name    species
##   <chr>   <chr>
## 1 Sparky  Dog
## 2 Patches Cat
## 3 Lassie  Dog
## 4 Tweety  Bird
```

b. **Make the same tibble as in the previous part (two columns, the first containing all the names of the pets in your shelter that were adopted in October, and the second containing their species). But instead, use a filtering join (semi_join or anti_join) as well as whatever other transformations are necessary. Don't use a mutating join here.**

```r
semi_join(October_Pets, October_Adoptions, by = "name") %>% select(name, species)
```

```
## # A tibble: 4 x 2
##   name    species
##   <chr>   <chr>
## 1 Sparky  Dog
## 2 Lassie  Dog
## 3 Patches Cat
## 4 Tweety  Bird
```

Question 11 (9 points)

You visited the animal shelter yesterday and today, visited several pets:

a. **What pets did you visit both days? Give a simple command that produces a tibble with the answer.**

```r
semi_join(pet_visits_yesterday, pet_visits_today, by = "pet")
```

```
## # A tibble: 2 x 2
##   pet    species
##   <chr>  <chr>
## 1 Fluffy Cat
## 2 Sparky Dog
```

b. **What pets did you visit today but not yesterday? Give a simple command that produces a tibble with the answer.**

```r
anti_join( pet_visits_today,pet_visits_yesterday, by = "pet")
```

```
## # A tibble: 2 x 2
##   pet    species
##   <chr>  <chr>
## 1 Lassie Dog
## 2 Spot   Lizard
```

c. **You also made a more complete data set, where you also noted each pet's mood during your visit. Now how would you make a tibble containing the pets that you visited both days?**

```
inner_join(pet_visits_yesterday, pet_visits_today, by = c("pet", "species")) %>%
  mutate(mood_yesterday = mood.x, mood_today = mood.y) %>%
  select(pet, species, mood_yesterday, mood_today)
```

```
## # A tibble: 2 x 4
##   pet    species mood_yesterday mood_today
##   <chr>  <chr>   <chr>          <chr>
## 1 Fluffy Cat     Sleepy         Playful
## 2 Sparky Dog     Playful        Sleepy
```