Ch5: group_by and summarize

YOUR NAME HERE

2/28/2022

Notes about summarize

- The other "verbs" we used (mutate, filter, select) kept the meaning of the rows of the dataset the same (e.g. each row is a flight, each row is a diamond). Summarize will NOT. Now each row represents a GROUP.
- summary statistics summarize information about a group with a single value. Examples are mean, median, minimum, maximum, standard deviation and IQR. Those are all useful summary statistics for numeric variables. For categorical variables, we might summarize them with a proportion, or a count. (Yes, the count, or number, is also a summary statistic because it describes a group!)

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

In this markdown we will practice group_by and summarize.

To make things simpler, we will at first use a subset of all the NOT CANCELLED flights

```
not_cancelled <- flights %>%
  filter(!is.na(dep_delay), !is.na(arr_delay)) %>%
  select(year, month, day, origin, dest, distance, carrier, sched_dep_time, hour, dep_time, dep_delay, a

not_cancelled
```

```
## # A tibble: 327,346 x 13
##     year month   day origin dest  distance carrier sched~1  hour dep_t~2 dep_d~3
##    <int> <int> <int> <chr>  <chr>    <dbl> <chr>     <int> <dbl>   <int>   <dbl>
##  1  2013     1     1 EWR    IAH       1400 UA          515     5     517       2
##  2  2013     1     1 LGA    IAH       1416 UA          529     5     533       4
##  3  2013     1     1 JFK    MIA       1089 AA          540     5     542       2
##  4  2013     1     1 JFK    BQN       1576 B6          545     5     544      -1
##  5  2013     1     1 LGA    ATL        762 DL          600     6     554      -6
##  6  2013     1     1 EWR    ORD        719 UA          558     5     554      -4
##  7  2013     1     1 EWR    FLL       1065 B6          600     6     555      -5
##  8  2013     1     1 LGA    IAD        229 EV          600     6     557      -3
##  9  2013     1     1 JFK    MCO        944 B6          600     6     557      -3
## 10  2013     1     1 LGA    ORD        733 AA          600     6     558      -2
## # ... with 327,336 more rows, 2 more variables: arr_time <int>,
## #   arr_delay <dbl>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_time, 3: dep_delay
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

1. What does the code above do?

Ans: The code above filters out the records that does not have any arrival or departure delay. That is because either those flights got cancelled or they went to a different destination and

did not arriave in the intended destination.

For all of the following, use the dataset "not_cancelled"!

Summarize

We can use summarize to find summary statistics, such as average, median, standard deviation....

```
not_cancelled %>% summarize(MeanArrTime = mean(arr_time))
```

```
## # A tibble: 1 x 1
##   MeanArrTime
##         <dbl>
## 1       1502.
```

```
not_cancelled %>% summarize(MedianArrTime = median(arr_time))
```

```
## # A tibble: 1 x 1
##   MedianArrTime
##           <dbl>
## 1          1535
```

```
not_cancelled %>% summarize(MedianArrTime = median(arr_time),
                            MedianDepTime = median(dep_time))
```

```
## # A tibble: 1 x 2
##   MedianArrTime MedianDepTime
##           <dbl>         <dbl>
## 1          1535          1400
```

2. Summarize can be particularly powerful in combination with other verbs! Descrie what the code below does.

```
not_cancelled %>%
  filter(origin == "JFK") %>%
  summarize(mean_dep_delay = mean(dep_delay))
```

```
## # A tibble: 1 x 1
##   mean_dep_delay
##            <dbl>
## 1           12.0
```

Ans: The code chunk calculates the mean departure delay of flights that have origin destination as JFK airport.

3. Find the average arrival delay for flights flying into MSP.

```
not_cancelled %>%
  filter(dest == "MSP") %>%
  summarize(mean_arr_delay = mean(arr_delay))
```

```
## # A tibble: 1 x 1
##   mean_arr_delay
##            <dbl>
## 1           7.27
```

group_by()

Often times we are interested in finding means, proportions, or other statistics BY GROUPS.

For example, if we want to find the average arrival delay *for each origin airport*:

```
not_cancelled %>%
  group_by(origin) %>%
  summarize(mean_arrival_delay = mean(arr_delay))
```

```
## # A tibble: 3 x 2
##   origin mean_arrival_delay
##   <chr>               <dbl>
## 1 EWR                  9.11
## 2 JFK                  5.55
## 3 LGA                  5.78
```

Notice that this produces a new, smaller dataset!

4. Read the code below carefully. Before running it, predict how many rows and columns the resulting dataset should have. Were you right?

Ans: The resulting dataset should have 3 columns and 12 rows.

```
not_cancelled %>%
  group_by(month) %>%
  summarize(mean_dep_delay = mean(dep_delay),
            median_dep_delay = median(dep_delay),
            max_dep_delay = max(dep_delay))
```

```
## # A tibble: 12 x 4
##    month mean_dep_delay median_dep_delay max_dep_delay
##    <int>          <dbl>            <dbl>         <dbl>
##  1     1           9.99               -2          1301
##  2     2          10.8                -2           853
##  3     3          13.2                -1           911
##  4     4          13.8                -2           960
##  5     5          12.9                -1           878
##  6     6          20.7                 0          1137
##  7     7          21.5                 0          1005
##  8     8          12.6                -1           520
##  9     9           6.63               -3          1014
## 10    10           6.23               -3           702
## 11    11           5.42               -3           798
## 12    12          16.5                 0           896
```

Ans: Unfortunately I forgot about the month column.

5. The variable "hour" indicates the hour of the day the flight is scheduled to depart (e.g. departing at 5:35Am is hour=5; departing at 1:32pm, or 13:32, is hour=13). Find the mean departure delay for each hour.

```
not_cancelled %>%
  group_by(hour) %>%
  summarize(mean_dep_delay = mean(dep_delay))
```

```
## # A tibble: 19 x 2
##     hour mean_dep_delay
##    <dbl>          <dbl>
```

```
##  1     5          0.689
##  2     6          1.60
##  3     7          1.91
##  4     8          4.11
##  5     9          4.54
##  6    10          6.45
##  7    11          7.15
##  8    12          8.52
##  9    13         11.3
## 10    14         13.7
## 11    15         16.8
## 12    16         18.6
## 13    17         21.0
## 14    18         21.0
## 15    19         24.7
## 16    20         24.2
## 17    21         24.2
## 18    22         18.7
## 19    23         14.0
```

n() : counting the number of rows.

The function n() just counts how many rows. We always use n() within a summarize(), never on its own!

```
not_cancelled %>%
  summarize(number_of_flights = n())
```

```
## # A tibble: 1 x 1
##   number_of_flights
##               <int>
## 1            327346
```

It can be useful in combination with group_by! For example, predict what the following will do before you run the code!

Ans: The code chunk will produce a dataset with mean departure delay of flights in each month and how many flights were there in each of those months.

```
not_cancelled %>%
  group_by(month) %>%
  summarize(mean_dep_delay = mean(dep_delay),
            n = n())
```

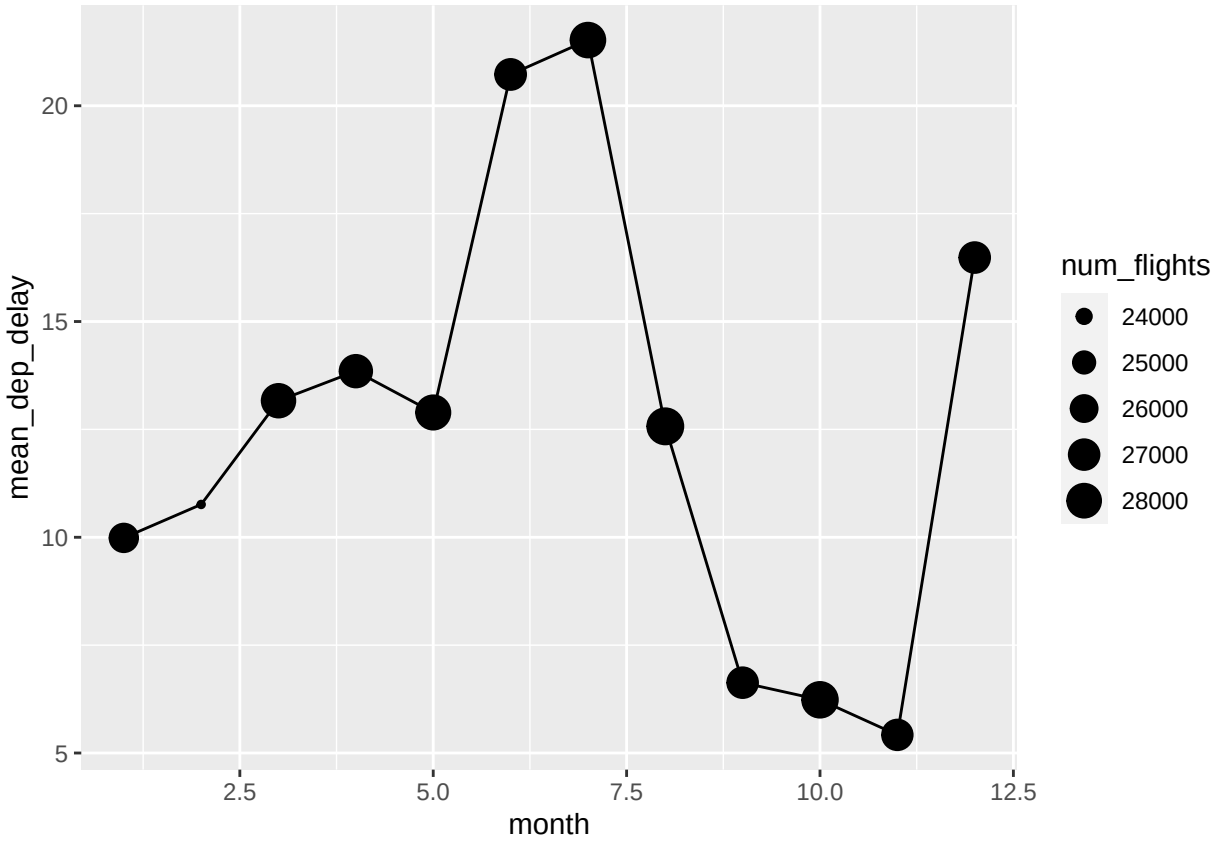
```
## # A tibble: 12 x 3
##    month mean_dep_delay     n
##    <int>          <dbl> <int>
##  1     1           9.99 26398
##  2     2          10.8  23611
##  3     3          13.2  27902
##  4     4          13.8  27564
##  5     5          12.9  28128
##  6     6          20.7  27075
##  7     7          21.5  28293
##  8     8          12.6  28756
##  9     9           6.63 27010
```

```
## 10    10           6.23 28618
## 11    11           5.42 26971
## 12    12          16.5  27020
```

Ans: It will create a geom plot with two layers. One layer will create a line graph of month number on the x axis and mean departure delay on the y axis. And the next layer will add points to each value and the size of the points will depend on the number of flights in each of those months.

Predict what this code will do before you run it:

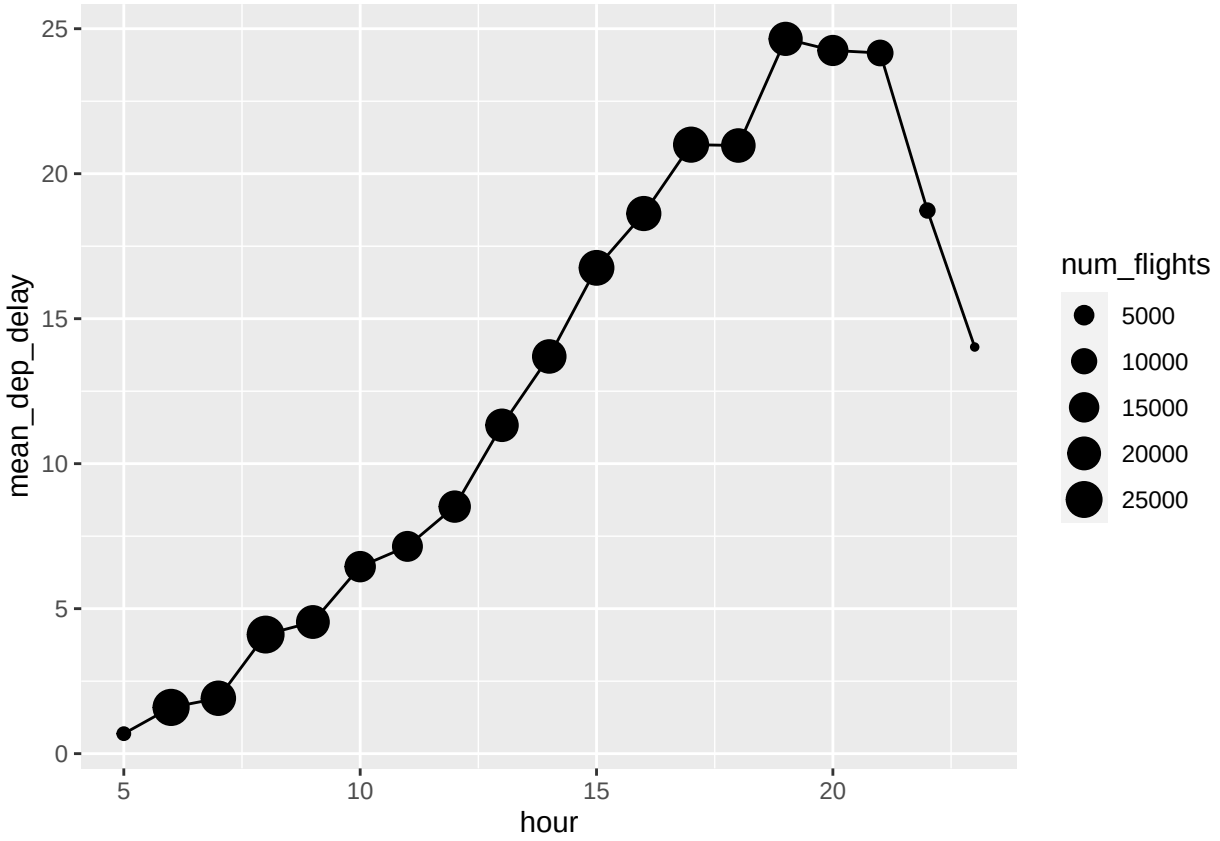
```
not_cancelled %>%
  group_by(month) %>%
  summarize(mean_dep_delay = mean(dep_delay),
            num_flights = n()) %>%
  ggplot(aes(x = month, y = mean_dep_delay)) +
  geom_line() +
  geom_point(aes(size = num_flights))
```



6. Copy and modify your code from #5 to create a graph with hour on x axis, mean_dep_delay on y-axis, with points sized by number of flights. Describe the pattern you see in the graph.

```
not_cancelled %>%
  group_by(hour) %>%
  summarize(mean_dep_delay = mean(dep_delay),
            num_flights = n()) %>%
  ggplot(aes(x = hour, y = mean_dep_delay)) +
  geom_line() +
```

```
  geom_point(aes(size = num_flights))
```

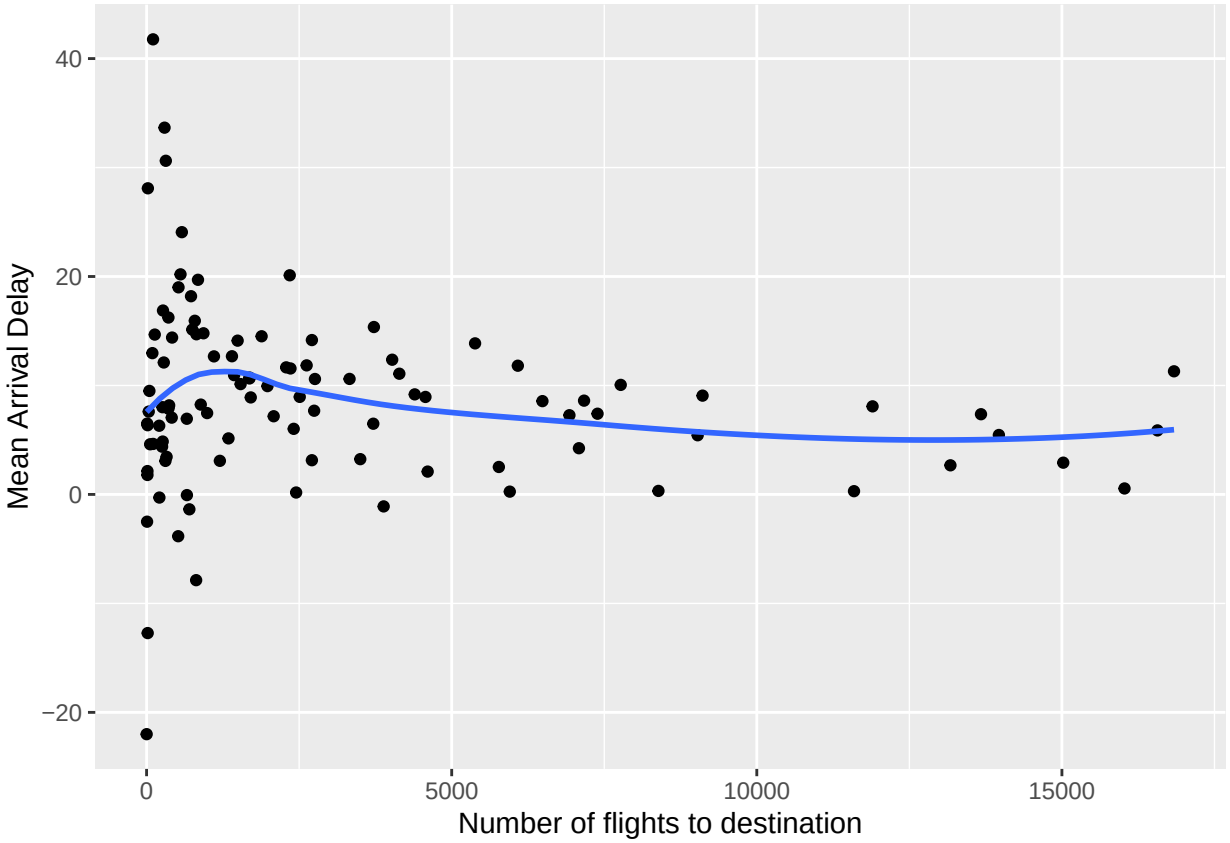


Ans: The departure delay increases as the hour proceeds from day to night and few hour before mifnight the delay drops drastically as the number of flights drop as well.

7. Look in the Class > Code > images folder. Create a graph that looks like ch5_mean_arr_delay_by_dest.png. Note that each dot on the graph represents one destination airport (dest). Before starting, think about what variables are on the x and y axis. What should your dataset look like? (what rows? columns?) You might even sketch it out!

```
not_cancelled %>%
  group_by(dest) %>%
  summarize(mean_arr_delay = mean(arr_delay),
            num_flights = n()) %>%
  ggplot(aes(x = num_flights , y = mean_arr_delay)) +
  geom_point()+
  geom_smooth(se = FALSE)+
  xlab("Number of flights to destination")+
  ylab("Mean Arrival Delay")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



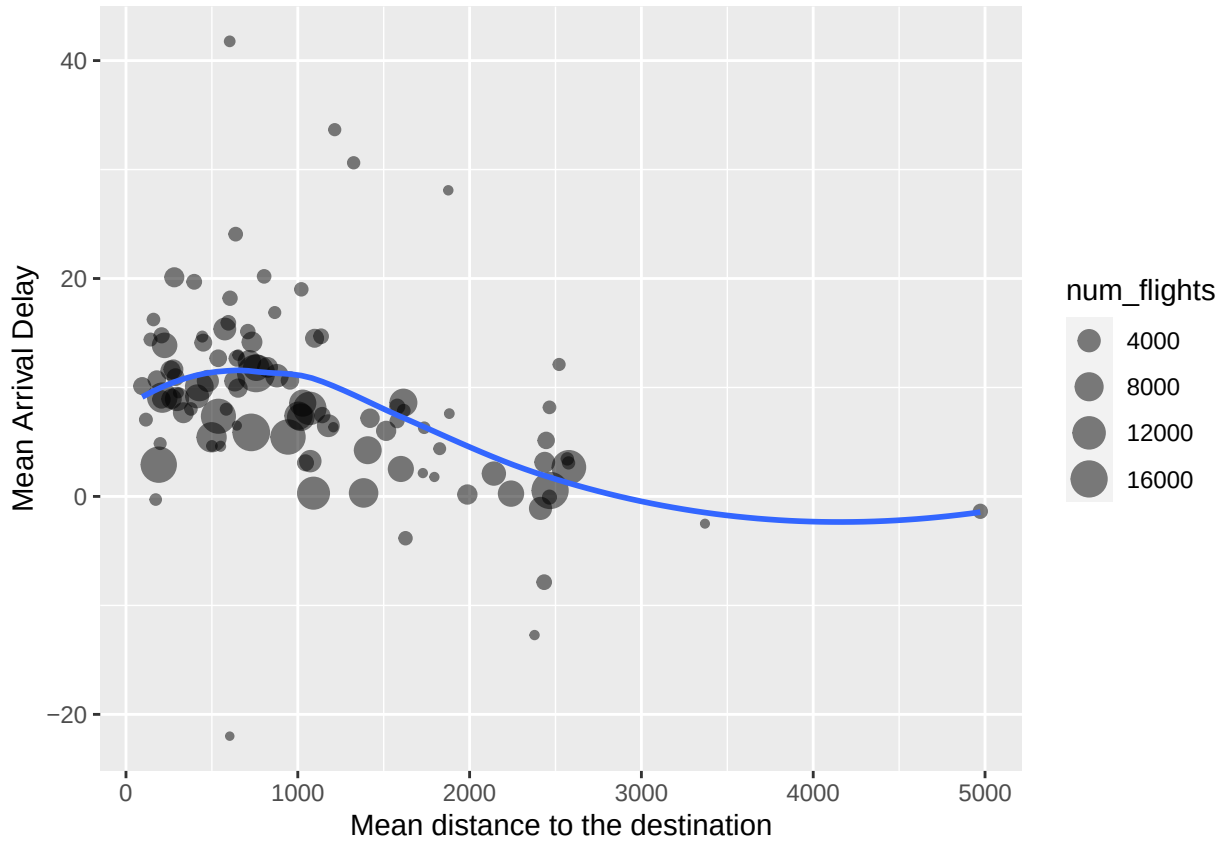
8. Describe the pattern you see in the graph. Do busier airports have longer average arrival delays?

Ans: There is a bit of noise in the mean arrival delay when the number of flights to destination is very low but as the number of flights increases the mean tends to remain whithin a short standard deviation.

9. Optional bonus problem: Copy and modify your code from 7 so that your graph has Distance on the x axis and each point is sized according to the number of flights. (each point still represents one destination; consider changing alpha level to make it easier to read!).

```
not_cancelled %>%
  group_by(dest) %>%
  summarize(mean_arr_delay = mean(arr_delay),
            mean_distance = mean(distance),
            num_flights = n()) %>%
  ggplot(aes(x = mean_distance , y = mean_arr_delay)) +
  geom_point(aes(size = num_flights), alpha = .5)+
  geom_smooth(se = FALSE)+
  xlab("Mean distance to the destination")+
  ylab("Mean Arrival Delay")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



10. Use slice_max to find the 5 destinations with the shortest mean arrival delay.

```
not_cancelled %>%
  group_by(dest) %>%
  summarize(mean_arr_delay = mean(arr_delay)) %>%
  arrange(mean_arr_delay)%>%
  slice_max(-(mean_arr_delay), n=5)
```

```
## # A tibble: 5 x 2
##   dest  mean_arr_delay
##   <chr>          <dbl>
## 1 LEX            -22
## 2 PSP            -12.7
## 3 SNA             -7.87
## 4 STT             -3.84
## 5 ANC             -2.5
```

11. Use slice_max to find the 5 destinations with the most flights.

```
not_cancelled %>%
  group_by(dest) %>%
  summarize(num_flights = n()) %>%
  slice_max(num_flights, n=5)
```

```
## # A tibble: 5 x 2
##   dest  num_flights
##   <chr>       <int>
## 1 ATL         16837
```

```
## 2 ORD         16566
## 3 LAX         16026
## 4 BOS         15022
## 5 MCO         13967
```

count()

Count() is an alternative to n(). It is basically a shortcut for group_by() and summarize(... = n()).

For example, we can count the total number of rows in not_cancelled:

```
not_cancelled %>% count()
```

```
## # A tibble: 1 x 1
##        n
##    <int>
## 1 327346
```

```
not_cancelled %>% summarize(n = n())
```

```
## # A tibble: 1 x 1
##        n
##    <int>
## 1 327346
```

We can count how many rows for each destination:

```
not_cancelled %>%
  group_by(dest) %>%
  summarize(n = n())
```

```
## # A tibble: 104 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
##  3 ALB     418
##  4 ANC       8
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # ... with 94 more rows
## # i Use `print(n = ...)` to see more rows
```

```
not_cancelled %>%
  count(dest)
```

```
## # A tibble: 104 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
##  3 ALB     418
##  4 ANC       8
```

```
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # ... with 94 more rows
## # i Use 'print(n = ...)' to see more rows
```

By observing that our resulting table has 104 rows, we can also use this to figure out that there are 104 unique destinations in the not_cancelled dataset!

12. How many destinations does United (UA) fly to? (hint: first filter!)

```
not_cancelled %>%
  filter(carrier == "UA") %>%
  count(dest)
```

```
## # A tibble: 47 x 2
##    dest      n
##    <chr> <int>
##  1 ANC       8
##  2 ATL     102
##  3 AUS     664
##  4 BDL       7
##  5 BOS    3297
##  6 BQN     295
##  7 BZN      35
##  8 CHS       1
##  9 CLE    1863
## 10 CLT       2
## # ... with 37 more rows
## # i Use 'print(n = ...)' to see more rows
```

Ans: 47 destinations.

13. How many destinations are flown to from Newark? (origin == "EWR")

```
not_cancelled %>%
  filter(origin == "EWR") %>%
  count(dest)
```

```
## # A tibble: 85 x 2
##    dest      n
##    <chr> <int>
##  1 ALB     418
##  2 ANC       8
##  3 ATL    4876
##  4 AUS     957
##  5 AVL     251
##  6 BDL     412
##  7 BNA    2241
##  8 BOS    5247
##  9 BQN     295
## 10 BTV     886
## # ... with 75 more rows
## # i Use 'print(n = ...)' to see more rows
```

Ans: 85 destinations.

14. Which are the top 5 destinations flown to from Newark? (hint slice_max!)

```
not_cancelled %>%
  filter(origin == "EWR") %>%
  count(dest)%>%
  slice_max(n , n = 5)
```

```
## # A tibble: 5 x 2
##   dest      n
##   <chr> <int>
## 1 ORD    5828
## 2 BOS    5247
## 3 SFO    5064
## 4 CLT    4893
## 5 MCO    4893
```

For some real crazy, try using group_by and count:

```
not_cancelled %>%
  group_by(origin) %>%
  count(dest)
```

```
## # A tibble: 223 x 3
## # Groups:   origin [3]
##    origin dest      n
##    <chr>  <chr> <int>
##  1 EWR    ALB     418
##  2 EWR    ANC       8
##  3 EWR    ATL    4876
##  4 EWR    AUS     957
##  5 EWR    AVL     251
##  6 EWR    BDL     412
##  7 EWR    BNA    2241
##  8 EWR    BOS    5247
##  9 EWR    BQN     295
## 10 EWR    BTV     886
## # ... with 213 more rows
## # i Use `print(n = ...)` to see more rows
```

15. Can you add another line to the code above to count how many destinations each origin airport flies to? (e.g. resulting dataset should have 3 rows). Remember we can read `count( XXXX )` as "count the number of rows for each category in XXX" For example, count(dest) counts the number of rows (flights) to each destination airport.

```
not_cancelled %>%
  group_by(origin) %>%
  count(dest)%>%
  count(origin)
```

```
## # A tibble: 3 x 2
## # Groups:   origin [3]
##   origin     n
##   <chr>  <int>
## 1 EWR       85
## 2 JFK       70
## 3 LGA       68
```