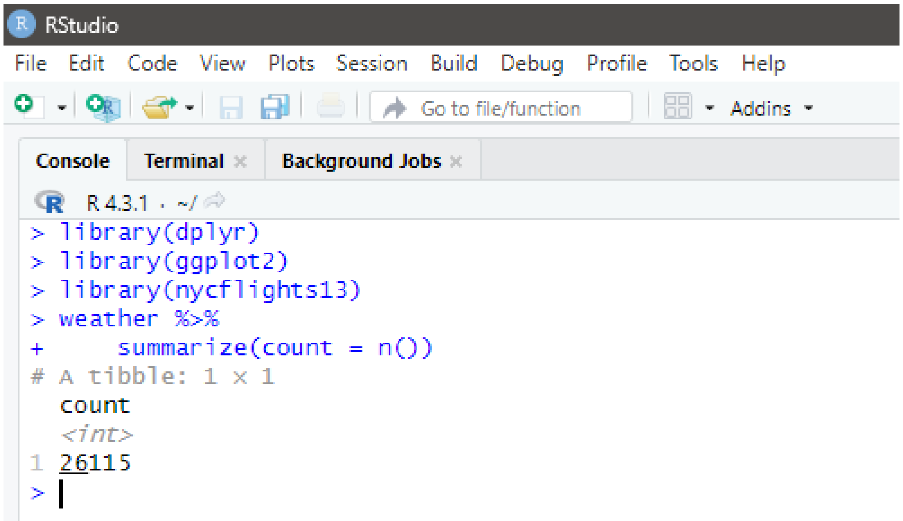
(LC3.1) What's another way using the "not" operator! to filter only the rows that are not going to Burlington, VT nor Seattle, WA in the flights data frame? Test this out using the code above.

```
not_BTV_SEA <- flights %>%
  filter(!(dest == "BTV" | dest == "SEA"))
not_BTV_SEA <- flights %>%
  filter(!dest == "BTV" & !dest == "SEA")
not_BTV_SEA <- flights %>%
  filter(dest != "BTV" & dest != "SEA")
```

(LC3.2) Say a doctor is studying the effect of smoking on lung cancer for a large number of patients who have records measured at five-year intervals. She notices that a large number of patients have missing data points because the patient has died, so she chooses to ignore these patients in her analysis. What is wrong with this doctor's approach?

Lung cancer may have killed the missing patients! So ignoring them might severely skew our results! It is critical to consider the implications of omitting missing data for our study! Considering the following:
Is there a reason why some values are missing? If this is the case, our findings may be skewed!
If there isn't, it may be acceptable to "sweep missing values under the rug."

(LC3.3) Modify the above summarize function to create summary_temp to also use the n() summary function: summarize(count = n()). What does the returned value correspond to?

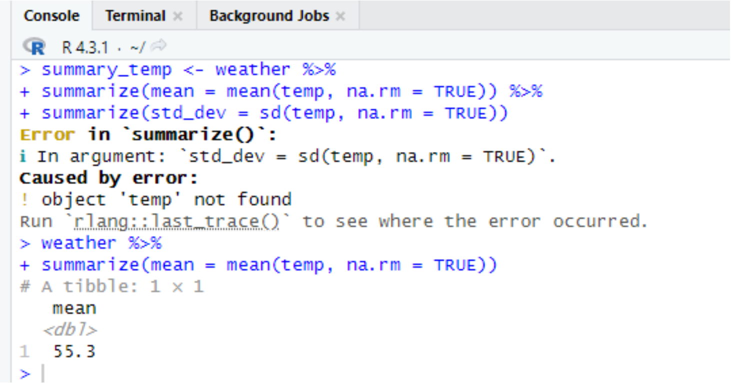


```
> library(dplyr)
> library(ggplot2)
> library(nycflights13)
> weather %>%
+     summarize(count = n())
# A tibble: 1 × 1
  count
  <int>
1 26115
> 
```

```
weather %>%
  summarize(count = n())
```

```
# A tibble: 1 × 1
  count
  <int>
1 26115
```

(LC3.4) Why doesn't the following code work? Run the code line by line instead of all at once, and then look at the data. In other words, run summary_temp <- weather %>% summarize(mean = mean(temp, na.rm = TRUE)) first.



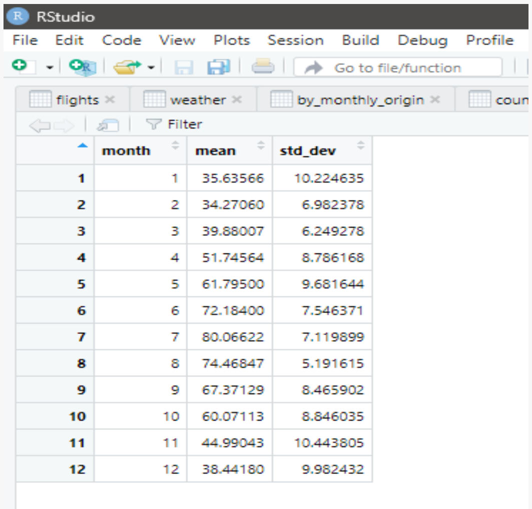
```
> summary_temp <- weather %>%
+ summarize(mean = mean(temp, na.rm = TRUE)) %>%
+ summarize(std_dev = sd(temp, na.rm = TRUE))
Error in `summarize()`:
i In argument: `std_dev = sd(temp, na.rm = TRUE)`.
Caused by error:
! object 'temp' not found
Run `rlang::last_trace()` to see where the error occurred.
> weather %>%
+ summarize(mean = mean(temp, na.rm = TRUE))
# A tibble: 1 × 1
   mean
  <dbl>
1  55.3
> 
```

```
weather %>%
  summarize(mean = mean(temp, na.rm = TRUE))
```

```
# A tibble: 1 × 1
    mean
   <dbl>
1 55.3
```

As the variable temp has been compressed to the value mean after the first summarise(). When we try to perform the second summarise(), it is unable to locate the variable temp on which to compute the standard deviation.

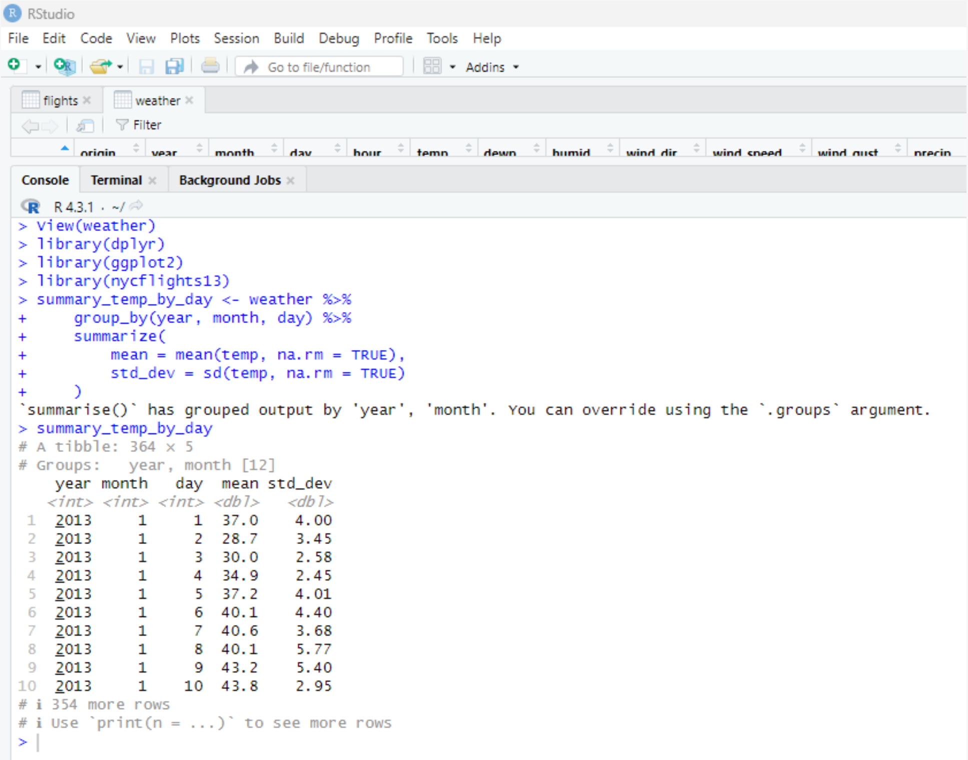
(LC3.5) Recall from Chapter 2 when we looked at plots of temperatures by months in NYC. What does the standard deviation column in the summary_monthly_temp data frame tell us about temperatures in New York City throughout the year?



	month	mean	std_dev
1	1	35.63566	10.224635
2	2	34.27060	6.982378
3	3	39.88007	6.249278
4	4	51.74564	8.786168
5	5	61.79500	9.681644
6	6	72.18400	7.546371
7	7	80.06622	7.119899
8	8	74.46847	5.191615
9	9	67.37129	8.465902
10	10	60.07113	8.846035
11	11	44.99043	10.443805
12	12	38.44180	9.982432

(LC3.6) What code would be required to get the mean and standard deviation temperature for each day in 2013 for NYC?

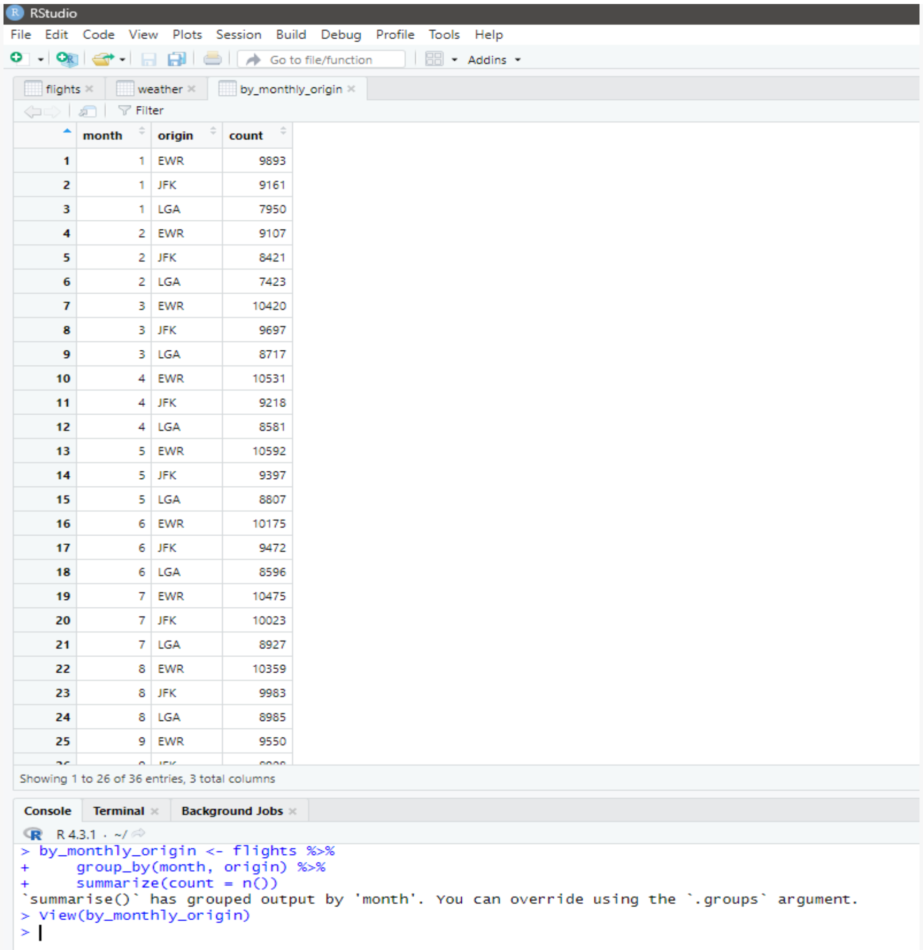
```
summary_temp_by_day <- weather %>%
  group_by(year, month, day) %>%
  summarize(
    mean = mean(temp, na.rm = TRUE),
    std_dev = sd(temp, na.rm = TRUE)
  )
summary_temp_by_day
```



```
> View(weather)
> library(dplyr)
> library(ggplot2)
> library(nycflights13)
> summary_temp_by_day <- weather %>%
+     group_by(year, month, day) %>%
+     summarize(
+         mean = mean(temp, na.rm = TRUE),
+         std_dev = sd(temp, na.rm = TRUE)
+     )
`summarise()` has grouped output by 'year', 'month'. You can override using the `.groups` argument.
> summary_temp_by_day
# A tibble: 364 × 5
# Groups:   year, month [12]
    year month   day  mean std_dev
   <int> <int> <int> <dbl>   <dbl>
 1  2013     1     1  37.0    4.00
 2  2013     1     2  28.7    3.45
 3  2013     1     3  30.0    2.58
 4  2013     1     4  34.9    2.45
 5  2013     1     5  37.2    4.01
 6  2013     1     6  40.1    4.40
 7  2013     1     7  40.6    3.68
 8  2013     1     8  40.1    5.77
 9  2013     1     9  43.2    5.40
10  2013     1    10  43.8    2.95
# ℹ 354 more rows
# ℹ Use `print(n = ...)` to see more rows
> |
```

(LC3.7) Recreate by_monthly_origin, but instead of grouping via group_by(origin, month), group variables in a different order group_by(month, origin). What differs in the resulting dataset?

```
count_flights_by_airport <- flights %>%
  group_by(origin, carrier) %>%
  summarize(count = n())
```



	month	origin	count
1	1	EWR	9893
2	1	JFK	9161
3	1	LGA	7950
4	2	EWR	9107
5	2	JFK	8421
6	2	LGA	7423
7	3	EWR	10420
8	3	JFK	9697
9	3	LGA	8717
10	4	EWR	10531
11	4	JFK	9218
12	4	LGA	8581
13	5	EWR	10592
14	5	JFK	9397
15	5	LGA	8807
16	6	EWR	10175
17	6	JFK	9472
18	6	LGA	8596
19	7	EWR	10475
20	7	JFK	10023
21	7	LGA	8927
22	8	EWR	10359
23	8	JFK	9983
24	8	LGA	8985
25	9	EWR	9550

Showing 1 to 26 of 36 entries, 3 total columns

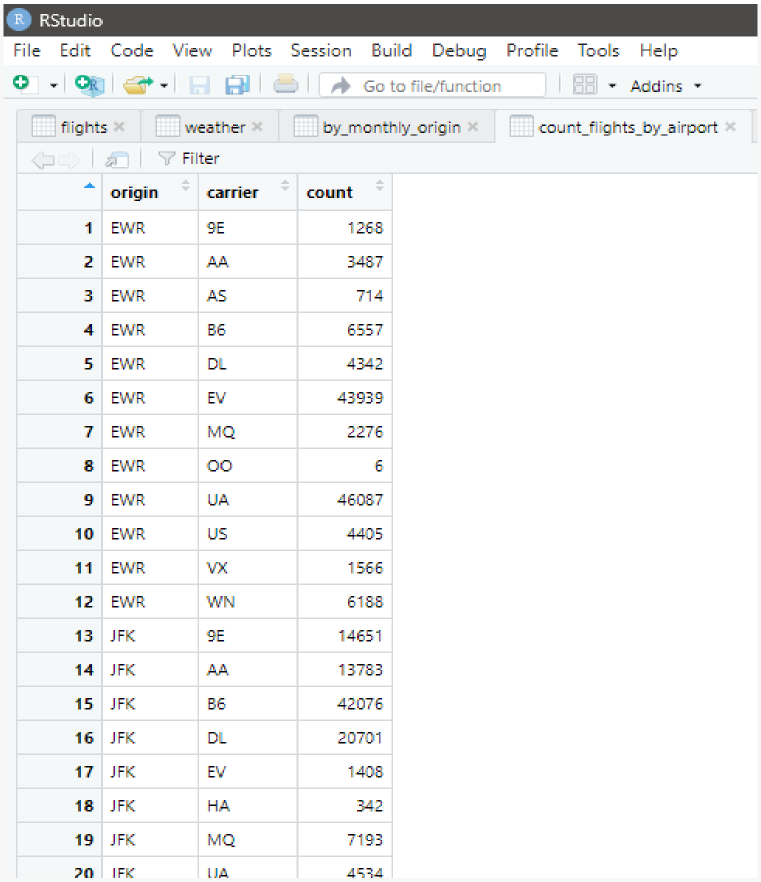
```
> by_monthly_origin <- flights %>%
+     group_by(month, origin) %>%
+     summarize(count = n())
`summarise()` has grouped output by 'month'. You can override using the `.groups` argument.
> view(by_monthly_origin)
> 
```

The month column is now first in by_monthly_origin, and the rows are ordered by month rather than origin. When we use the View() method to compare the values of count in by_origin_monthly and by_monthly_origin, we'll see that the data are the same, only displayed in a different order.

(LC3.8) How could we identify how many flights left each of the three airports for each carrier?

Using the n() method, which counts rows, we may summarise the count from each airport.

```
count_flights_by_airport <- flights %>%
  group_by(origin, carrier) %>%
  summarize(count = n())
```



	origin	carrier	count
1	EWR	9E	1268
2	EWR	AA	3487
3	EWR	AS	714
4	EWR	B6	6557
5	EWR	DL	4342
6	EWR	EV	43939
7	EWR	MQ	2276
8	EWR	OO	6
9	EWR	UA	46087
10	EWR	US	4405
11	EWR	VX	1566
12	EWR	WN	6188
13	JFK	9E	14651
14	JFK	AA	13783
15	JFK	B6	42076
16	JFK	DL	20701
17	JFK	EV	1408
18	JFK	HA	342
19	JFK	MQ	7193
20	JFK	UA	4534

(LC3.9) How does the filter operation differ from a group by followed by a summarize?

The filter extracts rows from the original dataset while leaving the others alone, although the group by%>% summarise generates new values by computing summaries of numerical variables.

(LC3.10) What do positive values of the gain variable in flights correspond to? What about negative values? And what about a zero value?

Let's assume that a flight was delayed by 30 minutes, dep_delay = 30.
came after that 20 minutes late, or arr_delay = 20.
As a result, gain = dep_delay - arr_delay = 30 - 20 = 10 is positive, indicating that it "made up/gained time in the air."

If both the departure and arrival times were 0, no extra time was added. We see that the increase is typically close to zero minutes.

(LC3.11) Could we create the dep_delay and arr_delay columns by simply subtracting dep_time from sched_dep_time and similarly for arrivals? Try the code out and explain any differences between the result and what actually appears in flights

No, because times cannot be directly arithmetic. There are 4 minutes between 12:03 and 11:59, yet 12:03 and 11:59 equal 44.

(LC3.12) What can we say about the distribution of gain? Describe it in a few sentences using the plot and the gain_summary data frame values.

The gain is often between -50 and 50 minutes and slightly over zero (the median is 7, implying gain is above 0 at least 50% of the time). Although, there are some extreme instances!

(LC3.13) Looking at Figure 3.7, when joining flights and weather (or, in other words, matching the hourly weather values with each flight), why do we need to join by all of year, month, day, hour, and origin, and not just hour?

Hour is only a number between 0 and 23, thus we need to know the year, month, day, and airport in order to identify a certain hour.

(LC3.14) What surprises you about the top 10 destinations from NYC in 2013?

The large number of flights to Boston surprised me.

(LC3.15) What are some advantages of data in normal forms? What are some disadvantages?

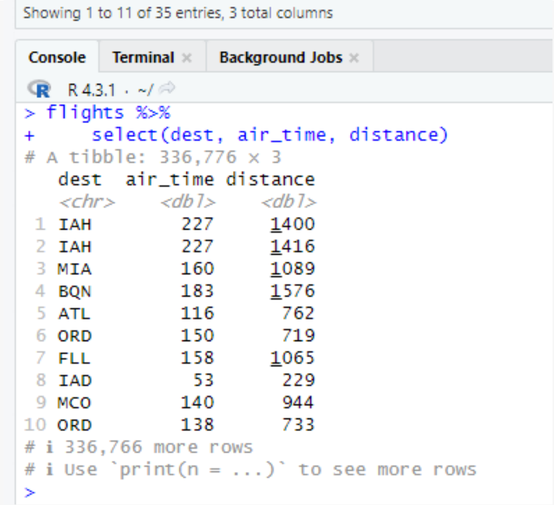
We can simply join datasets with other datasets when they are in normal form! We could, for instance, combine flight and aircraft data.

(LC3.16) What are some ways to select all three of the dest, air_time, and distance variables from flights? Give the code showing how to do this in at least three different ways.

```
flights %>%
  select(dest, air_time, distance)
```

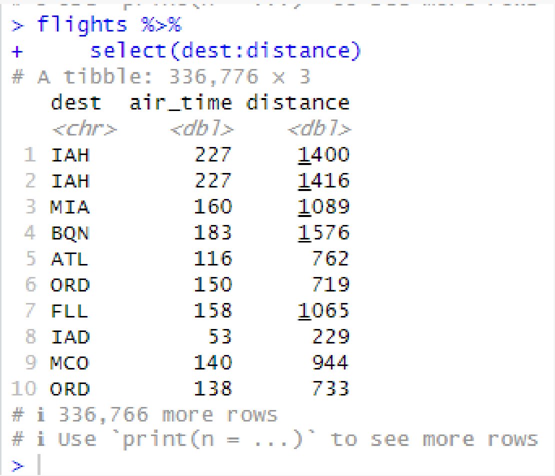
```
flights %>%
+   select(dest:distance)
```

```
flights %>%
  select(
    -year, -month, -day, -dep_time, -sched_dep_time, -dep_delay, -arr_time,
    -sched_arr_time, -arr_delay, -carrier, -flight, -tailnum, -origin,
    -hour, -minute, -time_hour
  )
```

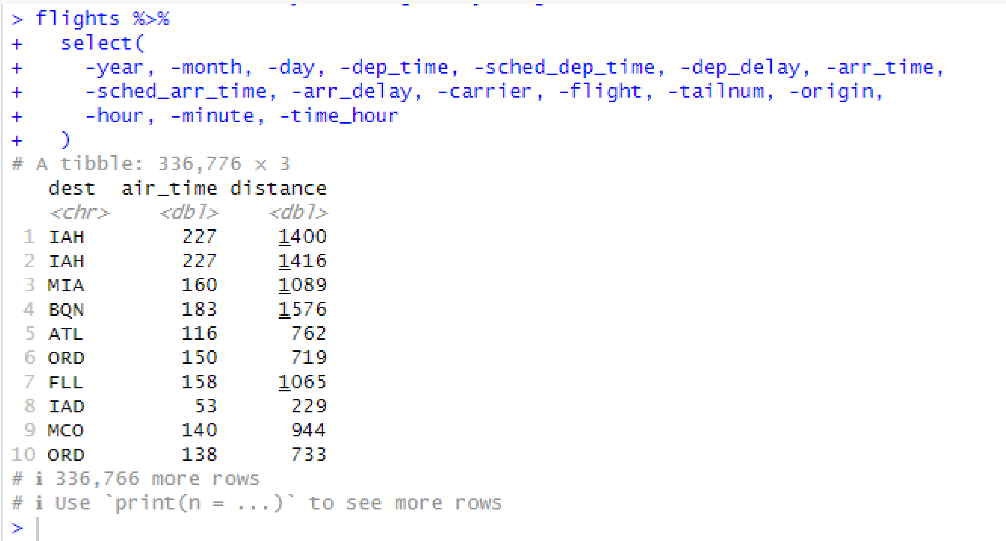


Showing 1 to 11 of 35 entries, 3 total columns

```
> flights %>%
+     select(dest, air_time, distance)
# A tibble: 336,776 × 3
   dest  air_time distance
   <chr>    <dbl>    <dbl>
 1 IAH        227     1400
 2 IAH        227     1416
 3 MIA        160     1089
 4 BQN        183     1576
 5 ATL        116      762
 6 ORD        150      719
 7 FLL        158     1065
 8 IAD         53      229
 9 MCO        140      944
10 ORD        138      733
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
>
```



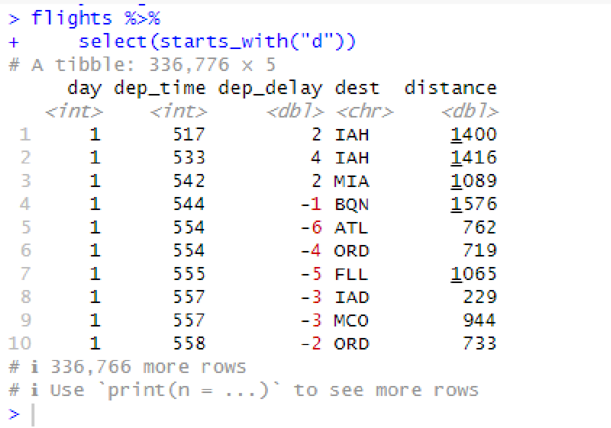
```
> flights %>%
+     select(dest:distance)
# A tibble: 336,776 × 3
   dest  air_time distance
   <chr>    <dbl>    <dbl>
 1 IAH        227     1400
 2 IAH        227     1416
 3 MIA        160     1089
 4 BQN        183     1576
 5 ATL        116      762
 6 ORD        150      719
 7 FLL        158     1065
 8 IAD         53      229
 9 MCO        140      944
10 ORD        138      733
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
> |
```



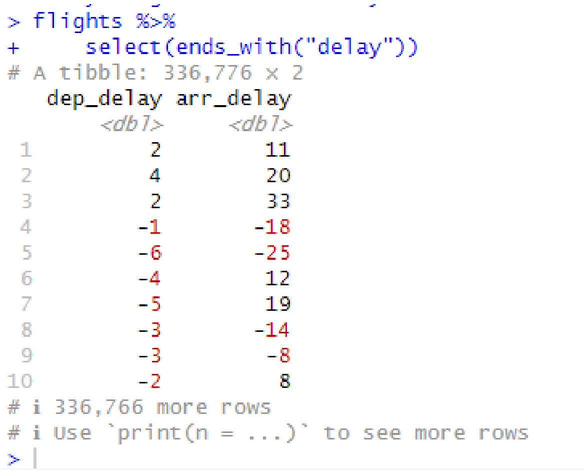
```
> flights %>%
+   select(
+     -year, -month, -day, -dep_time, -sched_dep_time, -dep_delay, -arr_time,
+     -sched_arr_time, -arr_delay, -carrier, -flight, -tailnum, -origin,
+     -hour, -minute, -time_hour
+   )
# A tibble: 336,776 × 3
   dest  air_time distance
   <chr>    <dbl>    <dbl>
 1 IAH        227     1400
 2 IAH        227     1416
 3 MIA        160     1089
 4 BQN        183     1576
 5 ATL        116      762
 6 ORD        150      719
 7 FLL        158     1065
 8 IAD         53      229
 9 MCO        140      944
10 ORD        138      733
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
> |
```

(LC3.17) How could one use starts_with, ends_with, and contains to select columns from the flights data frame? Provide three different examples in total: one for starts_with, one for ends_with, and one for contains

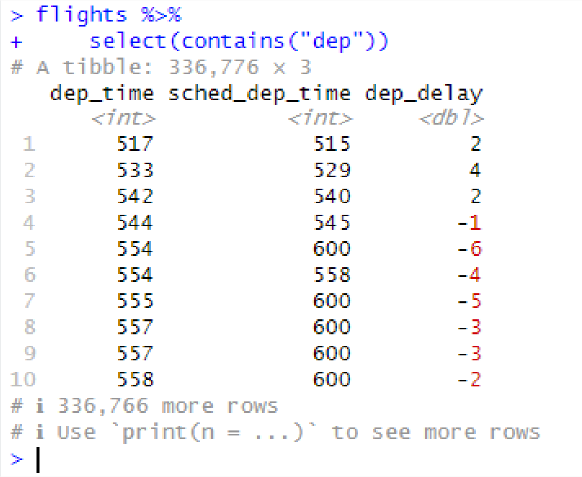
```
flights %>%
  select(starts_with("d"))

flights %>%
  select(ends_with("delay"))

flights %>%
  select(contains("dep"))
```



```
> flights %>%
+     select(starts_with("d"))
# A tibble: 336,776 × 5
     day dep_time dep_delay dest  distance
   <int>    <int>     <dbl> <chr>    <dbl>
 1     1      517         2 IAH       1400
 2     1      533         4 IAH       1416
 3     1      542         2 MIA       1089
 4     1      544        -1 BQN       1576
 5     1      554        -6 ATL        762
 6     1      554        -4 ORD        719
 7     1      555        -5 FLL       1065
 8     1      557        -3 IAD        229
 9     1      557        -3 MCO        944
10     1      558        -2 ORD        733
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
> 
```



```
> flights %>%
+     select(ends_with("delay"))
# A tibble: 336,776 × 2
   dep_delay arr_delay
       <dbl>     <dbl>
 1         2        11
 2         4        20
 3         2        33
 4        -1       -18
 5        -6       -25
 6        -4        12
 7        -5        19
 8        -3       -14
 9        -3        -8
10        -2         8
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
> 
```

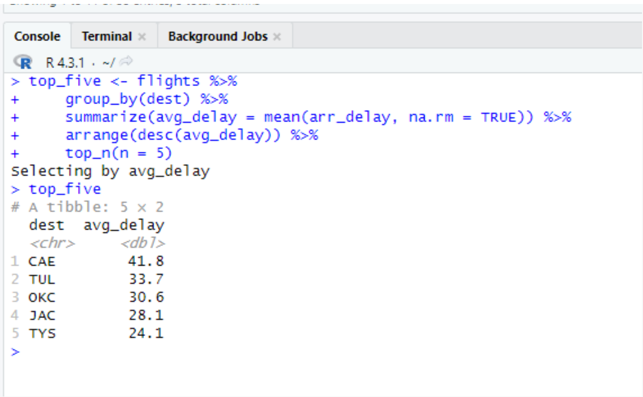


```
> flights %>%
+     select(contains("dep"))
# A tibble: 336,776 × 3
   dep_time sched_dep_time dep_delay
      <int>          <int>     <dbl>
 1      517            515         2
 2      533            529         4
 3      542            540         2
 4      544            545        -1
 5      554            600        -6
 6      554            558        -4
 7      555            600        -5
 8      557            600        -3
 9      557            600        -3
10      558            600        -2
# ℹ 336,766 more rows
# ℹ Use `print(n = ...)` to see more rows
> 
```

(LC3.18) Why might we want to use the select() function on a data frame?
To reduce the data frame's size and make it easier to examine. use View() as an illustration.

(LC3.19) Create a new data frame that shows the top 5 airports with the largest arrival delays from NYC in 2013.

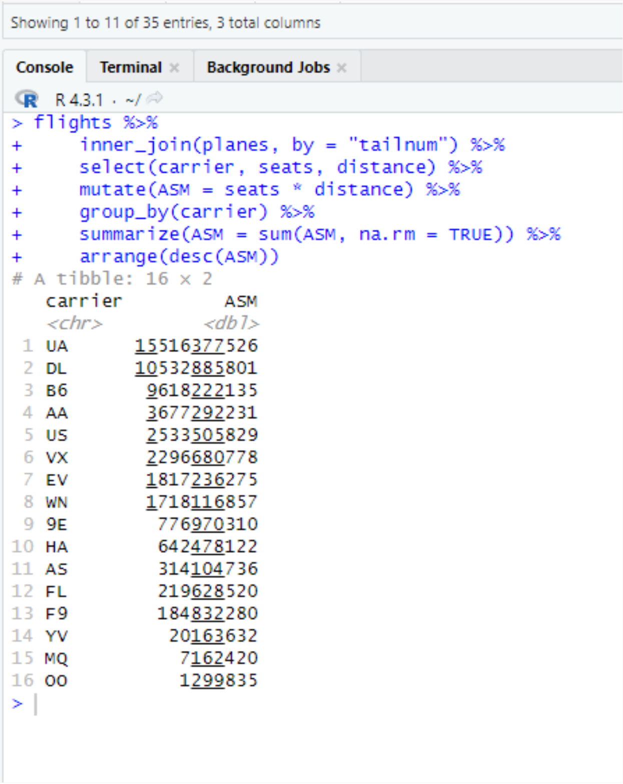
```
top_five <- flights %>%
  group_by(dest) %>%
  summarize(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(avg_delay)) %>%
  top_n(n = 5)
top_five
```



```
> top_five <- flights %>%
+     group_by(dest) %>%
+     summarize(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
+     arrange(desc(avg_delay)) %>%
+     top_n(n = 5)
Selecting by avg_delay
> top_five
# A tibble: 5 × 2
  dest  avg_delay
  <chr>     <dbl>
1 CAE        41.8
2 TUL        33.7
3 OKC        30.6
4 JAC        28.1
5 TYS        24.1
>
```

(LC3.20) Using the datasets included in the nycflights13 package, compute the available seat miles for each airline sorted in descending order. After completing all the necessary data wrangling steps, the resulting data frame should have 16 rows (one for each airline) and 2 columns (airline name and available seat miles). Here are some hints:

```
flights %>%
  inner_join(planes, by = "tailnum") %>%
  select(carrier, seats, distance) %>%
  mutate(ASM = seats * distance) %>%
  group_by(carrier) %>%
  summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
  arrange(desc(ASM))
```


Showing 1 to 11 of 35 entries, 3 total columns

```
> flights %>%
+     inner_join(planes, by = "tailnum") %>%
+     select(carrier, seats, distance) %>%
+     mutate(ASM = seats * distance) %>%
+     group_by(carrier) %>%
+     summarize(ASM = sum(ASM, na.rm = TRUE)) %>%
+     arrange(desc(ASM))
# A tibble: 16 × 2
   carrier         ASM
   <chr>         <dbl>
 1 UA      15516377526
 2 DL      10532885801
 3 B6       9618222135
 4 AA       3677292231
 5 US       2533505829
 6 VX       2296680778
 7 EV       1817236275
 8 WN       1718116857
 9 9E        776970310
10 HA        642478122
11 AS        314104736
12 FL        219628520
13 F9        184832280
14 YV         20163632
15 MQ          7162420
16 OO          1299835
> |
```