```
# Assignment: 5.2 Exercise Housing dataset
# Name: Koby-Hercsky, Theodore
# Date: 2021-04-13

# Using the dplyr package, use the 6 different operations to analyze/transform the data - GroupBy, Summarize, Mutate, Filter, Select, and Arrange – Remember this isn't just modifying data, you are learning about your data also – so play around and start to understand your dataset in more detail
# 1. GroupBy
Updated_Housing_Dataset %>% group_by(zip5) %>% tally()
# 2. Summarize
Updated_Housing_Dataset %>% group_by(zip5) %>% summarize(Mean_Price = mean(Sale.Price, na.rm = TRUE))
# 3. Mutate
Price_Per_square_feet_housingdata <- mutate(Updated_Housing_Dataset, price_per_square_foot = Sale.Price / square_feet_total_living)
# 4. Filter
zipcode_Housing_Data <- filter(Updated_Housing_Dataset, zip5 == 98053)
# 5. Select
Updated_Housing_Dataset <- select(Housing_dataset, Sale.Price, zip5, square_feet_total_living, bedrooms, bath_full_count, bath_half_count, sq_ft_lot)
# 6. Arrange
Housing_Data_By_Price <- Updated_Housing_Dataset %>% arrange(desc(Sale.Price), .by_group = FALSE)

# Using the purrr package – perform 2 functions on your dataset.  You could use zip_n, keep, discard, compact, etc.
# 1. Keep
Mean_Keep_Housing_data_set <- Updated_Housing_Dataset %>% keep(~ mean(.x) > 600)
# 2. Discard
Discard_Housingdata <- Updated_Housing_Dataset %>% discard(~ sum(.x) > 1000000)
# Use the cbind and rbind function on your dataset
# cbind
Cbind_Housing <- cbind(zipcode_house, Price_Housing_Dataset, Discard_Housingdata)
# rbind
rbind_ZipHousing <- rbind(zipcode_98052_Housing_Data, zipcode_98053_Housing_Data)

# Split a string, then concatenate the results back together
Year_sold <- str_split(string = Housing_data$Sale.Date, pattern = "-")
```

Assignment: 5.2 Exercise Housing dataset explanation and screenshots
Name: Koby-Hercsky, Theodore
Date: 2021-04-13

1. Using the dplyr package, use the 6 different operations to analyze/transform the data - GroupBy, Summarize, Mutate, Filter, Select, and Arrange – Remember this isn't just modifying data, you are learning about your data also – so play around and start to understand your dataset in more detail
    a. **# 1. GroupBy** – I used the groupby function below to group each zip code and tally the number of houses that are in each. As seen below the largest number of houses in our data set is coming from 98052 with 7,452 houses while zip code 98059 only has one house in our data set.
        i. **Updated_Housing_Dataset %>% group_by(zip5) %>% tally()**
        ii.

```
# A tibble: 4 x 2
   zip5     n
  <dbl> <int>
1 98052  7452
2 98053  5339
3 98059     1
4 98074    73
```

    b. **# 2. Summarize** – I used the summarize function to find the mean price of each zip code in our housing data set. The one with the highest mean price is zipcode 98074 as it is for $ 951,544.00 while it only has 73 houses in this data set as seen in our group by function.

```
# A tibble: 4 x 2
   zip5 Mean_Price
  <dbl>      <dbl>
1 98052    649375.
2 98053    672624.
3 98059    645000
4 98074    951544.
```

        i. **Updated_Housing_Dataset %>% group_by(zip5) %>% summarize(Mean_Price = mean(Sale.Price, na.rm = TRUE))**
    c. **# 3. Mutate** – Used the mutate function seen below to add another column that is for the price per square foot of living space per house in our data set. This gives an idea of how much each house is in comparison to how big the actual house is. So, if the house is small but the price per square foot is high it must be in a nice area and or be a little smaller but very nice.

i. **Price_Per_square_feet_housingdata <- mutate(Updated_Housing_Dataset, price_per_square_foot = Sale.Price / square_feet_total_living)**

	Sale.Price	zip5	square_feet_total_living	bedrooms	bath_full_count	bath_half_count	sq_ft_lot	price_per_square_foot
1	698000	98052	2810	4	2	1	6635	248.39858
2	649990	98052	2880	4	2	0	5570	225.69097
3	572500	98052	2770	4	1	1	8444	206.67870
4	420000	98052	1620	3	1	0	9600	259.25926
5	369900	98052	1440	3	1	0	7526	256.87500
6	184667	98053	4160	4	2	1	7280	44.39111
7	1050000	98053	3960	5	3	0	97574	265.15152
8	875000	98053	3720	4	2	1	30649	235.21505
9	660000	98053	4160	4	2	1	42688	158.65385
10	650000	98052	2760	4	1	0	94889	235.50725
11	599950	98052	2180	3	2	1	7949	275.20642
12	526787	98052	2480	3	2	1	2647	212.41411
13	470000	98052	2230	4	1	0	12070	210.76233
14	165000	98053	1850	3	2	0	278891	89.18919

Showing 1 to 14 of 12,865 entries, 8 total columns

d. **# 4. Filter** – Seen below is my code using the Filter function that has filter houses that are located in the zip code 98053. As seen in the screen shot below, we have narrowed down our rows to 5,339 from 12,865 in our original table. This shows that the number of houses in in the 98053 zip code is almost half the houses in this data set.

i. **zipcode_Housing_Data <- filter(Updated_Housing_Dataset, zip5 == 98053)**

	Sale.Price	zip5	square_feet_total_living	bedrooms	bath_full_count	bath_half_count	sq_ft_lot
1	184667	98053	4160	4	2	1	7280
2	1050000	98053	3960	5	3	0	97574
3	875000	98053	3720	4	2	1	30649
4	660000	98053	4160	4	2	1	42688
5	165000	98053	1850	3	2	0	278891
6	803000	98053	3180	3	2	1	95013
7	765000	98053	4000	4	2	1	7611
8	372500	98053	1620	3	1	0	47480
9	513262	98053	1930	2	2	0	4958
10	482000	98053	2360	3	2	1	4080
11	765000	98053	3520	4	3	0	35348
12	265000	98053	4920	4	4	1	112650
13	523935	98053	1680	2	2	0	4764
14	399900	98053	1350	2	2	0	4781

Showing 1 to 14 of 5,339 entries, 7 total columns

e. **# 5. Select** – Seen below is my code for the select function from dplyr. I selected columns that I believe is needed to do further research on the housing data such as the price of sales, the zip code the house is located in, how large the living area is, and also the amount of bed and baths that are in the house.

i. Updated_Housing_Dataset -> select(Housing_dataset, Sale.Price, zip5, square_feet_total_living, bedrooms, bath_full_count, bath_half_count, sq_ft_lot)

	Sale.Price	zip5	square_feet_total_living	bedrooms	bath_full_count	bath_half_count	sq_ft_lot
1	698000	98052	2810	4	2	1	6635
2	649990	98052	2880	4	2	0	5570
3	572500	98052	2770	4	1	1	8444
4	420000	98052	1620	3	1	0	9600
5	369900	98052	1440	3	1	0	7526
6	184667	98053	4160	4	2	1	7280
7	1050000	98053	3960	5	3	0	97574
8	875000	98053	3720	4	2	1	30649
9	660000	98053	4160	4	2	1	42688
10	650000	98052	2760	4	1	0	94889
11	599950	98052	2180	3	2	1	7949
12	526787	98052	2480	3	2	1	2647
13	470000	98052	2230	4	1	0	12070
14	165000	98053	1850	3	2	0	278891

Showing 1 to 14 of 12,865 entries, 7 total columns

f. **# 6. Arrange** – I used the Arrange function to sort the price of each house from highest to lowest. This is useful to see houses in order of how much they cost. As seen below from our data set, we see the highest costing house in our data set to be $ 4,400,000.00.

i. **Housing_Data_By_Price <- Updated_Housing_Dataset %>% arrange(desc(Sale.Price), .by_group = FALSE)**

	Sale.Price	zip5	square_feet_total_living	bedrooms	bath_full_count	bath_half_count	sq_ft_lot
1	4400000	98052	5790	3	2	1	657816
2	4400000	98052	2410	3	1	0	1327090
3	4380542	98052	3290	4	2	1	6712
4	4380542	98052	2450	4	2	1	4749
5	4380542	98052	2750	4	2	1	5816
6	4380542	98052	3010	4	2	0	8908
7	4380542	98052	3200	5	2	1	4584
8	4380542	98052	3200	5	2	1	4681
9	4380542	98052	3620	5	3	0	9901
10	4380542	98052	2810	4	2	1	13289
11	4380542	98052	2550	4	2	1	4368
12	4380542	98052	2440	3	2	1	4244
13	4380542	98052	3160	4	2	1	5778
14	4380542	98052	3400	4	3	0	6740

Showing 1 to 14 of 12,865 entries, 7 total columns

2. Using the purrr package – perform 2 functions on your dataset. You could use zip_n, keep, discard, compact, etc. This will be used when I use my Cbind in the following tasks.

a. **Keep** – Used the Keep function to keep each line with a mean over 600 which got rid of my bedroom and bathroom columns
    i. Mean_Keep_Housing_data_set <- Updated_Housing_Dataset %>% keep(~ mean(.x) > 600)

	Sale.Price	zip5	square_feet_total_living	sq_ft_lot
1	698000	98052	2810	6635
2	649990	98052	2880	5570
3	572500	98052	2770	8444
4	420000	98052	1620	9600
5	369900	98052	1440	7526
6	184667	98053	4160	7280
7	1050000	98053	3960	97574
8	875000	98053	3720	30649
9	660000	98053	4160	42688
10	650000	98052	2760	94889
11	599950	98052	2180	7949
12	526787	98052	2480	2647
13	470000	98052	2230	12070
14	165000	98053	1850	278891

Showing 1 to 14 of 12,865 entries, 4 total columns

b. **Discard** – I used the discard function to get rid of all the columns except the ones that pertained to the bedrooms and bathrooms. This will be used when I use my Cbind in the following tasks.
    i. Discard_Housingdata <- Updated_Housing_Dataset %>% discard(~ sum(.x) > 100000

ii.

	bedrooms	bath_full_count	bath_half_count
1	4	2	1
2	4	2	0
3	4	1	1
4	3	1	0
5	3	1	0
6	4	2	1
7	5	3	0
8	4	2	1
9	4	2	1
10	4	1	0
11	3	2	1
12	3	2	1

3. Use the cbind and rbind function on your dataset
    a. Cbind – I used Cbind to combine what I kept and discarded when I used the purrr package with keep and discard. As seen below I combined the zipcode, price, bed, and bath to bring my data set back together.
        i. Cbind_Housing <- cbind(zipcode_house, Price_Housing_Dataset, Discard_Housingdata)

	zip5	Sale.Price	bedrooms	bath_full_count	bath_half_count
1	98052	698000	4	2	1
2	98052	649990	4	2	0
3	98052	572500	4	1	1
4	98052	420000	3	1	0
5	98052	369900	3	1	0
6	98053	184667	4	2	1
7	98053	1050000	5	3	0
8	98053	875000	4	2	1
9	98053	660000	4	2	1
10	98052	650000	4	1	0
11	98052	599950	3	2	1
12	98052	526787	3	2	1

Showing 1 to 12 of 12,865 entries, 5 total columns

b. Rbind – I have used Rbind to combine two data sets I created from the filter function that I created for zip code 98053 and 98052. This combined the rows for both zipcodes back together.
    i. rbind_ZipHousing <- rbind(zipcode_98052_Housing_Data, zipcode_98053_Housing_Data)

	Sale.Price	zip5	square_feet_total_living	bedrooms	bath_full_count	bath_half_count	sq_ft_lot
1	698000	98052	2810	4	2	1	6635
2	649990	98052	2880	4	2	0	5570
3	572500	98052	2770	4	1	1	8444
4	420000	98052	1620	3	1	0	9600
5	369900	98052	1440	3	1	0	7526
6	650000	98052	2760	4	1	0	94889
7	599950	98052	2180	3	2	1	7949
8	526787	98052	2480	3	2	1	2647
9	470000	98052	2230	4	1	0	12070
10	507950	98052	2480	3	2	1	3099
11	589950	98052	2570	4	2	1	4737
12	501000	98052	2620	3	1	0	9649
13	372500	98052	1640	2	1	1	3279
14	1392000	98052	3740	4	3	2	17291

Showing 1 to 14 of 12,791 entries, 7 total columns

4. Split a string, then concatenate the results back together
    a. Split the date the house was sold by using the str_split by using the code below and seen the output in the screen shot.

b. Year_sold <- str_split(string = Housing_data$Sale.Date, pattern = "-")

Name	Type	Value
Year_sold	list [12865]	List of length 12865
[[1]]	character [3]	'2006' '01' '03'
[[2]]	character [3]	'2006' '01' '03'
[[3]]	character [3]	'2006' '01' '03'
[[4]]	character [3]	'2006' '01' '03'
[[5]]	character [3]	'2006' '01' '03'
[[6]]	character [3]	'2006' '01' '03'
[[7]]	character [3]	'2006' '01' '04'
[[8]]	character [3]	'2006' '01' '04'
[[9]]	character [3]	'2006' '01' '04'
[[10]]	character [3]	'2006' '01' '04'
[[11]]	character [3]	'2006' '01' '04'

c. Year_sold[[8]]