Alcala_STA445_HW4

Angelica Alcala

2023-10-26

Question 1

a.

```
Survey <- read.csv('https://www.lock5stat.com/datasets3e/StudentSurvey.csv',
                   na.strings=c('',' '))
Coll.stats <- Survey %>%
  arrange(Year)

head(Coll.stats)
```

```
##        Year Sex Smoke   Award HigherSAT Exercise TV Height Weight Siblings
## 1 FirstYear   M    No   Nobel      Math       14  5     72    208        2
## 2 FirstYear   F    No Olympic      Math       10 10     66    128        1
## 3 FirstYear   F    No   Nobel      Math       12  1     60    115        7
## 4 FirstYear   M    No Olympic      Math       10  5     63    200        2
## 5 FirstYear   M    No   Nobel    Verbal        9  5     68    193        1
## 6 FirstYear   F    No Olympic      Math       10  2     63    110        1
##   BirthOrder VerbalSAT MathSAT  SAT  GPA Pulse Piercings
## 1          1       550     560 1110 2.55   130         0
## 2          1       640     680 1320 2.77    94         8
## 3          8       670     700 1370 3.70    94         2
## 4          2       580     600 1180   NA    72         0
## 5          1       700     650 1350   NA    72         0
## 6          2       590     610 1200 3.86    59         4
```

b.

Using some combination of dplyr functions, produce a data set with eight rows that contains the number of responses for each gender:year combination. Make sure your table orders the Year variable in the correct order of First Year, Sophmore, Junior, and then Senior.

```
Comb.stats <- Coll.stats %>%
  count(Year, Sex, name = "Responses") %>%
  mutate(Year = factor(Year, levels = c("FirstYear","Sophomore", "Junior", "Senior"))) %>%
  drop_na()
Comb.stats
```

```
##        Year Sex Responses
## 1 FirstYear   F        43
## 2 FirstYear   M        51
## 3    Junior   F        18
## 4    Junior   M        17
## 5    Senior   F        10
```

```
## 6    Senior   M        26
## 7 Sophomore   F        96
## 8 Sophomore   M        99
```

c.

```
Pivot.stats <- Comb.stats %>%
  pivot_wider(Sex, names_from = 'Year',
              values_from = 'Responses')
```

```
## Warning: Specifying the `id_cols` argument by position was deprecated in tidyr 1.3.0.
## i Please explicitly name `id_cols`, like `id_cols = Sex`.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
Pivot.stats
```

```
## # A tibble: 2 x 5
##   Sex   FirstYear Junior Senior Sophomore
##   <chr>     <int>  <int>  <int>     <int>
## 1 F            43     18     10        96
## 2 M            51     17     26        99
```

Question 2

```
Flagtemp <- read.csv('https://raw.githubusercontent.com/dereksonderegger/444/master/data-raw/FlagMaxTem
                     na.strings = c('NA',' '))
head(Flagtemp)
```

```
##   X Year Month    X1    X2    X3    X4    X5    X6    X7    X8    X9   X10
## 1 1 1985     5 71.06 71.06 68.00 68.00 64.94 64.04 64.04 64.94 69.08 66.02
## 2 2 1985     6 62.96 62.96 64.94 60.08 69.08 75.92 82.04 86.00 84.92 84.02
## 3 3 1985     7 80.96 86.00 89.96 87.98 91.94 91.94 89.06 87.98 89.96 87.08
## 4 4 1985     8 77.00 68.00 78.08 80.06 82.04 80.96 82.94 82.94 80.06 80.06
## 5 5 1985     9 82.94 75.02 73.94 71.96 66.92 62.96 62.96 64.04 68.00 64.94
## 6 6 1985    10 64.04 60.08 64.04 71.06 71.06 75.02 69.08 53.96 51.08 55.04
##     X11   X12   X13   X14   X15   X16   X17   X18   X19   X20   X21   X22   X23
## 1 51.08 55.94 59.00 57.92 66.02 66.92 66.02 66.02 68.00 66.92 66.92 62.96    NA
## 2 82.04 82.94 84.92 82.94 82.94 86.00 84.92 87.08 87.08 84.02 80.96 82.04 84.92
## 3 84.02 84.02 84.92 87.08 84.02 84.92 78.98 80.96 75.92 73.94 64.94 71.96 73.04
## 4 80.06 75.92 78.98 78.98 80.96 80.96 78.98 75.02 82.04 82.04 69.98 80.06 84.02
## 5 68.00 66.02 66.92 75.02 73.94 73.04 73.04 69.98 51.98 59.00 55.04 68.00 68.00
## 6 50.00 55.04 57.02 53.96 51.08 55.94 57.02 60.08 62.96 62.96 62.06 55.04 53.96
##     X24   X25   X26   X27   X28   X29   X30   X31
## 1 69.98 73.94 71.06 71.06 69.08 73.04 69.08 62.06
## 2 80.96 73.94 71.96 73.04 80.96 84.02 82.04    NA
## 3 80.06 80.06 77.00 80.06 82.04 77.00 73.94 73.04
## 4 87.98 91.04 84.92 84.02 80.96 78.98 82.04 82.94
## 5 71.06 71.96 73.04 71.96 64.94 60.98 64.04    NA
## 6 68.00 66.92 64.94 66.92 68.00 64.04 64.94 62.06
```

a.

Create a line graph that gives the daily maximum temperature for 2005. Make sure the x-axis is a date and covers the whole year.

```
Flagtemp1 <- Flagtemp %>%
  filter(Year == '2005') %>%
  pivot_longer(X1:X31, names_to = 'Day', values_to = 'Temperature') %>%
  drop_na()
head(Flagtemp1)
```

```
## # A tibble: 6 x 5
##       X  Year Month Day   Temperature
##   <int> <int> <int> <chr>       <dbl>
## 1   235  2005     1 X1           37.9
## 2   235  2005     1 X2           37.9
## 3   235  2005     1 X3           39.0
## 4   235  2005     1 X4           34.0
## 5   235  2005     1 X5           35.1
## 6   235  2005     1 X6           27.0
```

```
Flagtemp2 <- subset(Flagtemp1, select = -X)
head(Flagtemp2)
```

```
## # A tibble: 6 x 4
##    Year Month Day   Temperature
##   <int> <int> <chr>       <dbl>
## 1  2005     1 X1           37.9
## 2  2005     1 X2           37.9
## 3  2005     1 X3           39.0
## 4  2005     1 X4           34.0
## 5  2005     1 X5           35.1
## 6  2005     1 X6           27.0
```

```
Flagtemp3 <- Flagtemp2 %>%
  mutate( Day = str_remove_all(Day, "\\D"))
head(Flagtemp3)
```

```
## # A tibble: 6 x 4
##    Year Month Day   Temperature
##   <int> <int> <chr>       <dbl>
## 1  2005     1 1            37.9
## 2  2005     1 2            37.9
## 3  2005     1 3            39.0
## 4  2005     1 4            34.0
## 5  2005     1 5            35.1
## 6  2005     1 6            27.0
```

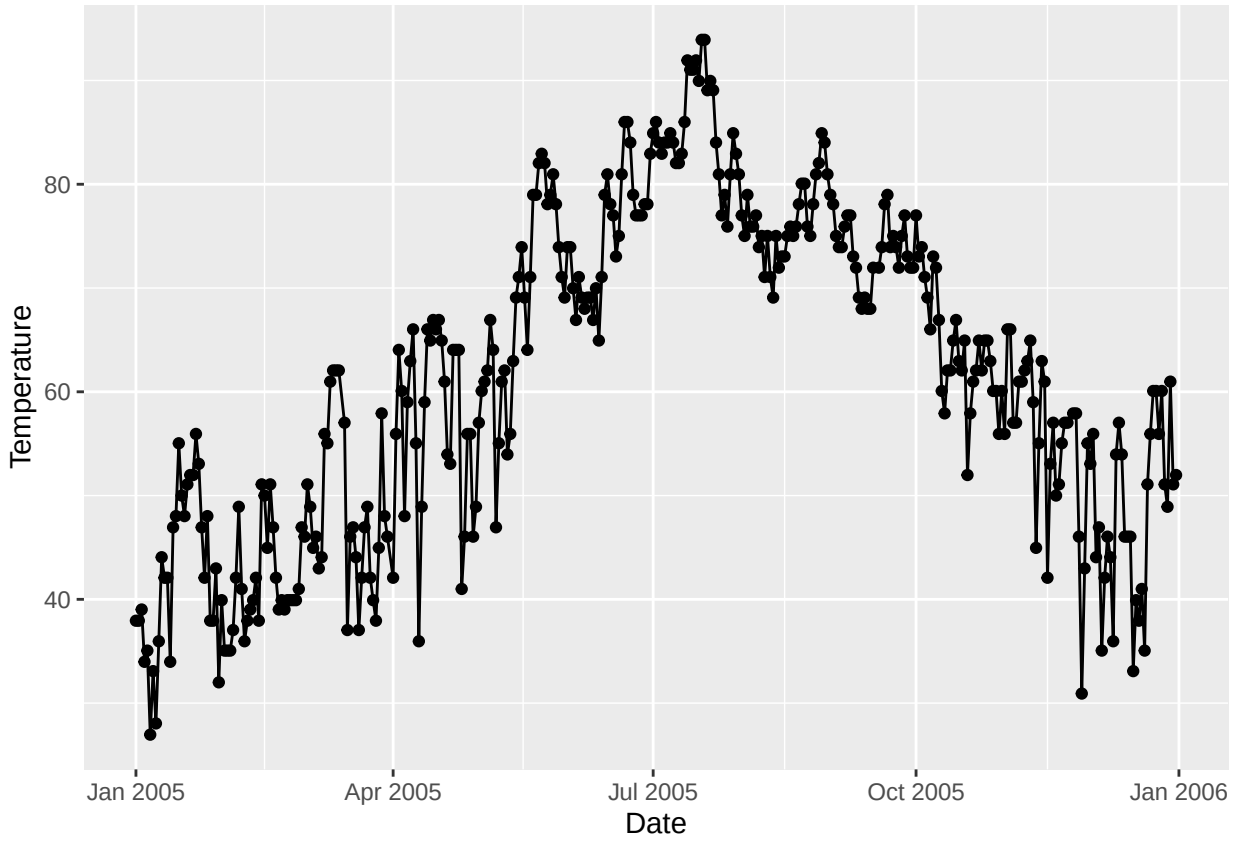
```
Flagtemp4 <- Flagtemp3 %>%
  mutate(Date = make_date(year = Year, month = Month, day = Day))
Flagtemp5 <- subset(Flagtemp4, select = c('Date', 'Temperature')) %>%
  drop_na()

Flagtemp5$Temperature <- as.numeric(Flagtemp5$Temperature)
head(Flagtemp5)
```

```
## # A tibble: 6 x 2
##   Date       Temperature
##   <date>           <dbl>
## 1 2005-01-01        37.9
## 2 2005-01-02        37.9
```

```
## 3 2005-01-03        39.0
## 4 2005-01-04        34.0
## 5 2005-01-05        35.1
## 6 2005-01-06        27.0
```

```
ggplot(data=Flagtemp5, aes(x=Date, y=Temperature) ) +
    geom_point(  ) +
  geom_line()
```




```
  labs(title = 'Daily Maximum Temperature for 2005')
```

```
## $title
## [1] "Daily Maximum Temperature for 2005"
##
## attr(,"class")
## [1] "labels"
```

b.

Create a line graph that gives the monthly average maximum temperature for 2013 - 2015.

```
FlagT <- Flagtemp %>%
  filter(Year %in% c('2013','2014','2015')) %>%
  pivot_longer(X1:X31, names_to = 'Day', values_to = 'Temperature') %>%
  drop_na()
head(FlagT)
```

```
## # A tibble: 6 x 5
##       X  Year Month Day   Temperature
```

```
##   <int> <int> <int> <chr>       <dbl>
## 1   330  2013     1 X1           30.0
## 2   330  2013     1 X2           28.9
## 3   330  2013     1 X3           36.0
## 4   330  2013     1 X4           48.0
## 5   330  2013     1 X5           43.0
## 6   330  2013     1 X6           41
```

```
FlagTmean <- FlagT %>%
  group_by(Year, Month) %>%
  summarise(MonthlyMean = mean(Temperature)) %>%
  drop_na()
```

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

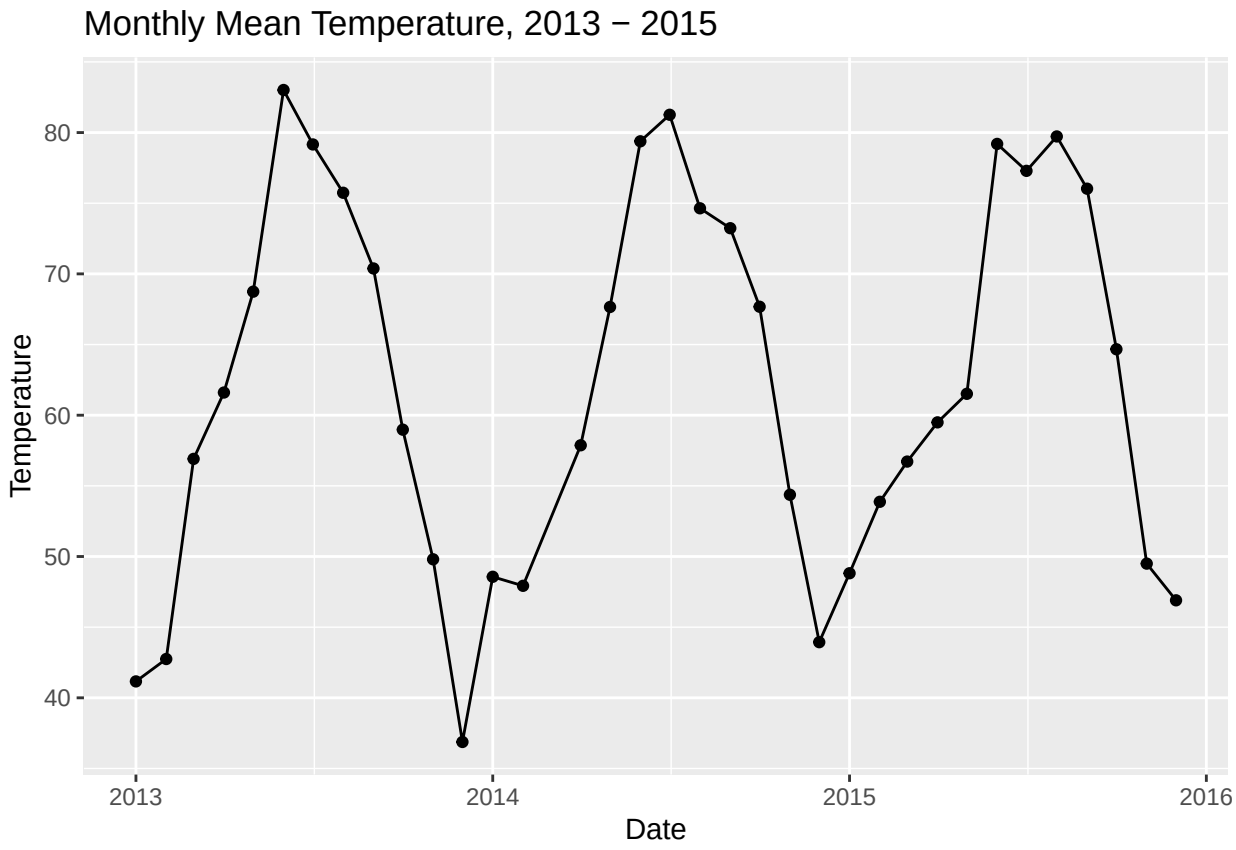
```
head(FlagTmean)
```

```
## # A tibble: 6 x 3
## # Groups:   Year [1]
##    Year Month MonthlyMean
##   <int> <int>       <dbl>
## 1  2013     1        41.2
## 2  2013     2        42.7
## 3  2013     3        56.9
## 4  2013     4        61.6
## 5  2013     5        68.7
## 6  2013     6        83.0
```

```
FlagTmean2 <- FlagTmean %>%
  mutate(date = make_date(year = Year, month = Month) )
head(FlagTmean2)
```

```
## # A tibble: 6 x 4
## # Groups:   Year [1]
##    Year Month MonthlyMean date
##   <int> <int>       <dbl> <date>
## 1  2013     1        41.2 2013-01-01
## 2  2013     2        42.7 2013-02-01
## 3  2013     3        56.9 2013-03-01
## 4  2013     4        61.6 2013-04-01
## 5  2013     5        68.7 2013-05-01
## 6  2013     6        83.0 2013-06-01
```

```
ggplot(data=FlagTmean2, aes(x=date, y=MonthlyMean) ) +
    geom_point(  ) +
  geom_line() +
  labs(title = 'Monthly Mean Temperature, 2013 - 2015',
       x = 'Date', y = 'Temperature')
```



Question 4

```
A <- tribble(
  ~Name, ~Car,
  'Alice', 'Ford F150',
  'Bob',   'Tesla Model III',
  'Charlie', 'VW Bug')

B <- tribble(
  ~First.Name, ~Pet,
  'Bob',  'Cat',
  'Charlie', 'Dog',
  'Alice', 'Rabbit') %>%
  arrange(First.Name)
A
```

```
## # A tibble: 3 x 2
##   Name    Car
##   <chr>   <chr>
## 1 Alice   Ford F150
## 2 Bob     Tesla Model III
## 3 Charlie VW Bug
```

```
B
```

```
## # A tibble: 3 x 2
##   First.Name Pet
##   <chr>      <chr>
```

```
## 1 Alice      Rabbit
## 2 Bob        Cat
## 3 Charlie    Dog
```

a.

Squish the data frames together to generate a data set with three rows and three columns. Do two ways: first using cbind and then using one of the dplyr join commands.

```
#B2 <- tibble(Pet = c('Rabbit','Cat','Dog'))
```

Using cbind:

```
pet.car <- cbind(A,B$Pet)
pet.car
```

```
##      Name             Car  B$Pet
## 1   Alice       Ford F150 Rabbit
## 2     Bob Tesla Model III    Cat
## 3 Charlie          VW Bug    Dog
```

Using join:

```
pet.car2 <- full_join(A,B,by = c("Name" = "First.Name"))
pet.car2
```

```
## # A tibble: 3 x 3
##   Name    Car             Pet
##   <chr>   <chr>           <chr>
## 1 Alice   Ford F150       Rabbit
## 2 Bob     Tesla Model III Cat
## 3 Charlie VW Bug          Dog
```

b.

It turns out that Alice also has a pet guinea pig. Add another row to the B data set. Do this using either the base function rbind, or either of the dplyr functions add_row or bind_rows.

```
B2 <- B %>%
  add_row(First.Name = 'Alice', Pet = 'Guinea Pig')

B2
```

```
## # A tibble: 4 x 2
##   First.Name Pet
##   <chr>      <chr>
## 1 Alice      Rabbit
## 2 Bob        Cat
## 3 Charlie    Dog
## 4 Alice      Guinea Pig
```

c.

Squish the A and B data sets together to generate a data set with four rows and three columns. Do this two ways: first using cbind and then using one of the dplyr join commands. Which was easier to program? Which is more likely to have an error.

Using cbind

Cbind gets an error because we are joining data frames that have a different number of rows.

Using join

```
pet.car4 <- full_join(A,B2,by = c("Name" = "First.Name"))
pet.car4
```

```
## # A tibble: 4 x 3
##   Name    Car             Pet
##   <chr>   <chr>           <chr>
## 1 Alice   Ford F150       Rabbit
## 2 Alice   Ford F150       Guinea Pig
## 3 Bob     Tesla Model III Cat
## 4 Charlie VW Bug          Dog
```

I think the join command was much easier, and it has the option for recognizing that the name columns were representing the same thing. cbind is more likely to get an error because it doesn't contain the ignore NA that join does, so if the rows mismatch it will return an error.

Question 5

Data table joins are extremely common because effective database design almost always involves having multiple tables for different types of objects. To illustrate both the table joins and the usefulness of multiple tables we will develop a set of data frames that will represent a credit card company's customer data base. We will have tables for Customers, Retailers, Cards, and Transactions. Below is code that will create and populate these tables.

```
Customers <- tribble(
     ~PersonID, ~Name, ~Street, ~City, ~State,
     1, 'Derek Sonderegger',  '231 River Run', 'Flagstaff', 'AZ',
     2, 'Aubrey Sonderegger', '231 River Run', 'Flagstaff', 'AZ',
     3, 'Robert Buscaglia', '754 Forest Heights', 'Flagstaff', 'AZ',
     4, 'Roy St Laurent', '845 Elk View', 'Flagstaff', 'AZ')

   Retailers <- tribble(
     ~RetailID, ~Name, ~Street, ~City, ~State,
     1, 'Kickstand Kafe', '719 N Humphreys St', 'Flagstaff', 'AZ',
     2, 'MartAnnes', '112 E Route 66', 'Flagstaff', 'AZ',
     3, 'REI', '323 S Windsor Ln', 'Flagstaff', 'AZ' )

   Cards <- tribble(
     ~CardID, ~PersonID, ~Issue_DateTime, ~Exp_DateTime,
     '9876768717278723',  1,  '2019-9-20 0:00:00', '2022-9-20 0:00:00',
     '5628927579821287',  2,  '2019-9-20 0:00:00', '2022-9-20 0:00:00',
     '7295825498122734',  3,  '2019-9-28 0:00:00', '2022-9-28 0:00:00',
     '8723768965231926',  4,  '2019-9-30 0:00:00', '2022-9-30 0:00:00' )

   Transactions <- tribble(
     ~CardID, ~RetailID, ~DateTime, ~Amount,
     '9876768717278723', 1, '2019-10-1 8:31:23',    5.68,
     '7295825498122734', 2, '2019-10-1 12:45:45',  25.67,
     '9876768717278723', 1, '2019-10-2 8:26:31',    5.68,
     '9876768717278723', 1, '2019-10-2 8:30:09',    9.23,
     '5628927579821287', 3, '2019-10-5 18:58:57',  68.54,
     '7295825498122734', 2, '2019-10-5 12:39:26',  31.84,
     '8723768965231926', 2, '2019-10-10 19:02:20', 42.83)
```

```
    Cards <- Cards %>%
      mutate( Issue_DateTime = lubridate::ymd_hms(Issue_DateTime),
              Exp_DateTime   = lubridate::ymd_hms(Exp_DateTime) )
    Transactions <- Transactions %>%
      mutate( DateTime = lubridate::ymd_hms(DateTime))
```

a.

Create a table that gives the credit card statement for Derek. It should give all the transactions, the amounts, and the store name. Write your code as if the only initial information you have is the customer's name.

```
cust2 <- Customers %>%
  filter(Name == 'Derek Sonderegger')
cust2
```

```
## # A tibble: 1 x 5
##   PersonID Name              Street        City      State
##      <dbl> <chr>             <chr>         <chr>     <chr>
## 1        1 Derek Sonderegger 231 River Run Flagstaff AZ
```

```
Cards2 <- Cards %>%
  filter(PersonID == '1')
Cards2
```

```
## # A tibble: 1 x 4
##   CardID           PersonID Issue_DateTime      Exp_DateTime
##   <chr>               <dbl> <dttm>              <dttm>
## 1 9876768717278723        1 2019-09-20 00:00:00 2022-09-20 00:00:00
```

```
Trans2 <- Transactions %>%
  filter(CardID == '9876768717278723')
Trans2
```

```
## # A tibble: 3 x 4
##   CardID           RetailID DateTime            Amount
##   <chr>               <dbl> <dttm>               <dbl>
## 1 9876768717278723        1 2019-10-01 08:31:23   5.68
## 2 9876768717278723        1 2019-10-02 08:26:31   5.68
## 3 9876768717278723        1 2019-10-02 08:30:09   9.23
```

```
card.data <- full_join(cust2, Cards2)
```

```
## Joining with `by = join_by(PersonID)`
```

```
card.data2 <- subset(card.data, select = - PersonID)
card.data2
```

```
## # A tibble: 1 x 7
##   Name         Street City  State CardID Issue_DateTime      Exp_DateTime
##   <chr>        <chr>  <chr> <chr> <chr>  <dttm>              <dttm>
## 1 Derek Sonde~ 231 R~ Flag~ AZ    98767~ 2019-09-20 00:00:00 2022-09-20 00:00:00
```

```
card.data3 <- full_join(Trans2, Retailers) %>%
  drop_na() %>%
  rename(Retailer = Name) %>%
  rename(Address = Street) %>%
  rename(RetCity = City) %>%
  rename(RetState = State)
```

```
## Joining with `by = join_by(RetailID)`
```

```
card.data4 <- subset(card.data3, select = - RetailID)
card.data4
```

```
## # A tibble: 3 x 7
##   CardID           DateTime            Amount Retailer  Address RetCity RetState
##   <chr>            <dttm>               <dbl> <chr>     <chr>   <chr>   <chr>
## 1 9876768717278723 2019-10-01 08:31:23   5.68 Kickstan~ 719 N ~ Flagst~ AZ
## 2 9876768717278723 2019-10-02 08:26:31   5.68 Kickstan~ 719 N ~ Flagst~ AZ
## 3 9876768717278723 2019-10-02 08:30:09   9.23 Kickstan~ 719 N ~ Flagst~ AZ
```

```
card.data5 <- full_join(card.data2,card.data4)
```

```
## Joining with `by = join_by(CardID)`
```

```
card.data5
```

```
## # A tibble: 3 x 13
##   Name         Street City  State CardID Issue_DateTime      Exp_DateTime
##   <chr>        <chr>  <chr> <chr> <chr>  <dttm>              <dttm>
## 1 Derek Sonde~ 231 R~ Flag~ AZ    98767~ 2019-09-20 00:00:00 2022-09-20 00:00:00
## 2 Derek Sonde~ 231 R~ Flag~ AZ    98767~ 2019-09-20 00:00:00 2022-09-20 00:00:00
## 3 Derek Sonde~ 231 R~ Flag~ AZ    98767~ 2019-09-20 00:00:00 2022-09-20 00:00:00
## # i 6 more variables: DateTime <dttm>, Amount <dbl>, Retailer <chr>,
## #   Address <chr>, RetCity <chr>, RetState <chr>
```

b.

Aubrey has lost her credit card on Oct 15, 2019. Close her credit card at 4:28:21 PM and issue her a new credit card in the Cards table.

```
aubexp <- make_datetime(year = 2019, month = 10, day = 15,
                        hour = 16, min = 28, sec = 21)
aubexp
```

```
## [1] "2019-10-15 16:28:21 UTC"
```

```
aubexp2 <- Cards %>%
  filter(PersonID == 2) %>%
  mutate(Exp_DateTime = aubexp)
aubexp2
```

```
## # A tibble: 1 x 4
##   CardID           PersonID Issue_DateTime      Exp_DateTime
##   <chr>               <dbl> <dttm>              <dttm>
## 1 5628927579821287        2 2019-09-20 00:00:00 2019-10-15 16:28:21
```

```
aubnew <- aubexp2 %>%
  mutate(Issue_DateTime = aubexp,
         Exp_DateTime = Issue_DateTime + years(6),
         CardID = '5628927533618892',
         PersonID = 2)

aubnew
```

```
## # A tibble: 1 x 4
##   CardID           PersonID Issue_DateTime      Exp_DateTime
##   <chr>               <dbl> <dttm>              <dttm>
```

```
## 1 5628927533618892        2 2019-10-15 16:28:21 2025-10-15 16:28:21
```

```
Cards3 <- full_join(Cards,aubnew)
```

```
## Joining with `by = join_by(CardID, PersonID, Issue_DateTime, Exp_DateTime)`
```

```
Cards3
```

```
## # A tibble: 5 x 4
##   CardID           PersonID Issue_DateTime      Exp_DateTime
##   <chr>               <dbl> <dttm>              <dttm>
## 1 9876768717278723        1 2019-09-20 00:00:00 2022-09-20 00:00:00
## 2 5628927579821287        2 2019-09-20 00:00:00 2022-09-20 00:00:00
## 3 7295825498122734        3 2019-09-28 00:00:00 2022-09-28 00:00:00
## 4 8723768965231926        4 2019-09-30 00:00:00 2022-09-30 00:00:00
## 5 5628927533618892        2 2019-10-15 16:28:21 2025-10-15 16:28:21
```

c.

Aubrey is using her new card at Kickstand Kafe on Oct 16, 2019 at 2:30:21 PM for coffee with a charge of $4.98. Generate a new transaction for this action.

```
Aubtrans <- tribble(
      ~CardID, ~RetailID, ~DateTime, ~Amount,
      '5628927533618892', 1, '2019-10-16 14:30:21',  4.98)
Aubtrans <- Aubtrans %>%
  mutate( DateTime = lubridate::ymd_hms(DateTime))
Aubtrans
```

```
## # A tibble: 1 x 4
##   CardID           RetailID DateTime            Amount
##   <chr>               <dbl> <dttm>               <dbl>
## 1 5628927533618892        1 2019-10-16 14:30:21   4.98
```

```
NewTrans <- full_join(Transactions,Aubtrans)
```

```
## Joining with `by = join_by(CardID, RetailID, DateTime, Amount)`
```

```
NewTrans
```

```
## # A tibble: 8 x 4
##   CardID           RetailID DateTime            Amount
##   <chr>               <dbl> <dttm>               <dbl>
## 1 9876768717278723        1 2019-10-01 08:31:23   5.68
## 2 7295825498122734        2 2019-10-01 12:45:45  25.7
## 3 9876768717278723        1 2019-10-02 08:26:31   5.68
## 4 9876768717278723        1 2019-10-02 08:30:09   9.23
## 5 5628927579821287        3 2019-10-05 18:58:57  68.5
## 6 7295825498122734        2 2019-10-05 12:39:26  31.8
## 7 8723768965231926        2 2019-10-10 19:02:20  42.8
## 8 5628927533618892        1 2019-10-16 14:30:21   4.98
```

d.

On Oct 17, 2019, some nefarious person is trying to use her OLD credit card at REI. Make sure your code in part (c) first checks to see if the credit card is active before creating a new transaction. Using the same code, verify that the nefarious transaction at REI is denied.

e.

Generate a table that gives the credit card statement for Aubrey. It should give all the transactions, amounts, and retailer name for both credit cards she had during this period.

```
AubCards <- Customers %>%
  filter(PersonID == '2')
AubCards
```

```
## # A tibble: 1 x 5
##   PersonID Name               Street        City      State
##      <dbl> <chr>              <chr>         <chr>     <chr>
## 1        2 Aubrey Sonderegger 231 River Run Flagstaff AZ
```

```
AubCards2 <- Cards %>%
  filter(PersonID == '2')
AubCards2
```

```
## # A tibble: 1 x 4
##   CardID           PersonID Issue_DateTime      Exp_DateTime
##   <chr>               <dbl> <dttm>              <dttm>
## 1 5628927579821287        2 2019-09-20 00:00:00 2022-09-20 00:00:00
```

```
AubCards3 <- full_join(AubCards,AubCards2) %>%
  subset(select = -PersonID)
```

```
## Joining with `by = join_by(PersonID)`
```

```
AubCards3
```

```
## # A tibble: 1 x 7
##   Name         Street City  State CardID Issue_DateTime      Exp_DateTime
##   <chr>        <chr>  <chr> <chr> <chr>  <dttm>              <dttm>
## 1 Aubrey Sond~ 231 R~ Flag~ AZ    56289~ 2019-09-20 00:00:00 2022-09-20 00:00:00
```

```
AubCards4 <- NewTrans %>%
  filter(CardID == c('5628927579821287','5628927533618892'))
AubCards4
```

```
## # A tibble: 2 x 4
##   CardID           RetailID DateTime            Amount
##   <chr>               <dbl> <dttm>               <dbl>
## 1 5628927579821287        3 2019-10-05 18:58:57  68.5
## 2 5628927533618892        1 2019-10-16 14:30:21   4.98
```

```
AubCards5 <- full_join(AubCards4, Retailers) %>%
  drop_na() %>%
  rename(Retailer = Name) %>%
  rename(Address = Street) %>%
  rename(RetCity = City) %>%
  rename(RetState = State) %>%
  subset(select = -RetailID)
```

```
## Joining with `by = join_by(RetailID)`
```

```
AubCards5
```

```
## # A tibble: 2 x 7
##   CardID           DateTime            Amount Retailer  Address RetCity RetState
##   <chr>            <dttm>               <dbl> <chr>     <chr>   <chr>   <chr>
## 1 5628927579821287 2019-10-05 18:58:57  68.5  REI       323 S ~ Flagst~ AZ
```

```
## 2 5628927533618892 2019-10-16 14:30:21   4.98 Kickstan~ 719 N ~ Flagst~ AZ
```

```
AubreySTMT <- full_join(AubCards3,AubCards5)
```

```
## Joining with `by = join_by(CardID)`
```

```
AubreySTMT2 <- full_join(aubnew,AubreySTMT)
```

```
## Joining with `by = join_by(CardID, Issue_DateTime, Exp_DateTime)`
```

```
AubreySTMT2
```

```
## # A tibble: 3 x 14
##   CardID     PersonID Issue_DateTime      Exp_DateTime        Name  Street City
##   <chr>         <dbl> <dttm>              <dttm>              <chr> <chr>  <chr>
## 1 562892753~        2 2019-10-15 16:28:21 2025-10-15 16:28:21 <NA>  <NA>   <NA>
## 2 562892757~       NA 2019-09-20 00:00:00 2022-09-20 00:00:00 Aubr~ 231 R~ Flag~
## 3 562892753~       NA NA                  NA                  <NA>  <NA>   <NA>
## # i 7 more variables: State <chr>, DateTime <dttm>, Amount <dbl>,
## #   Retailer <chr>, Address <chr>, RetCity <chr>, RetState <chr>
```