Module 6 Lab

Project set up

1. Create a new project for this lab.
2. Create a new .Rmd report. You can copy the text from this file into your .Rmd file and add your own notes and code as you see fit.
3. Download the `Month-XX.csv` files from the class Canvas module page to this project location.
4. Load the **tidyverse** and **repurrrsive** packages. The **repurrrsive** package provides some data sets we'll use later for the last few questions of the lab.

Analysis questions

Exercise 1

Import the `Month-XX.csv` files into your current R session. Do so using a `for` loop or ***purrr***. Rather than have 11 separate data frames (one for each month), combine these so that you have one data frame containing all the data. Your final data frame should have 698,159 rows and 10 columns.

```
# import all data files into one data frame titled "df"
# add your code here

# after your done, run this line and you should have same
# results as you see in the code chunk below
glimpse(df)
```

```
## Rows: 698,159
## Columns: 10
## $ Account_ID            <dbl> 5, 16, 28, 40, 62, 64, 69, 69, 70, 79, 88, 90, 9~
## $ Transaction_Timestamp <dttm> 2009-01-08 00:16:41, 2009-01-20 22:40:08, 2009-~
## $ Factor_A              <dbl> 2, 2, 2, 2, 2, 7, 2, 2, 2, 7, 8, 10, 10, 2, 2, 2~
## $ Factor_B              <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 18, 6, 6, 6, 6, 6,~
## $ Factor_C              <chr> "VI", "VI", "VI", "VI", "VI", "MC", "VI", "VI", ~
## $ Factor_D              <dbl> 20, 20, 21, 20, 20, 20, 20, 20, 20, 20, 20, 20, ~
## $ Factor_E              <chr> "A", "H", "NULL", "H", "B", "NULL", "H", "H", "B~
## $ Response              <dbl> 1020, 1020, 1020, 1020, 1020, 1020, 1020, 1020, ~
## $ Transaction_Status    <chr> "Approved", "Approved", "Approved", "Approved", ~
## $ Month                 <chr> "Jan", "Jan", "Jan", "Jan", "Jan", "Jan", "Jan",~
```

Exercise 2

Use one of the `map()` function variants to check the current class of each column i.e. (`class(df$Account_ID)`).

Exercise 3

Use one of the `map()` function variants to assess how many unique values exists in each column?

Exercise 4

The "Factor_D" variable contains 15 unique values (i.e. 10, 15, 20, 21, ..., 85, 90). There is at least one single observation where `Factor_D = 26` (possibly more). Assume these observations were improperly recorded and, in fact, the value should be 25.

Using `ifelse()` (or **dplyr**'s `if_else()`) inside `mutate()`, recode any values where `Factor_D == 26` to be 25. After completing this, how many unique values exist in this column? How many observations are there for each level of `Factor_D`?

Exercise 5

Unfortunately, some of the "Factor_" variables have observations that contain the value "NULL" (they are recorded as a character string, not the actual `NULL` value.
Use `filter_at()` to filter out any of these observations. We have not spent much time using `filter_at()` so you may need to do some research on it!

How many rows does your data now have (hint: it should be less than 500,000)?

Exercise 6

Using `mutate_at()` , convert all variables except for "Transaction_Timestamp" to factors. However, make sure the "Month" variable is an ordered factor. This may require you to do two separate `mutate()` statements.

```
df <- df %>%
  mutate_at(_____, _____) %>%
  mutate(Month = factor(_____, levels = _____))

# make sure all features are changed to factors
glimpse(df)

# make sure the Month variable is an ordered factor
levels(df$Month)
```

Exercise 7

Use the `summarize_if()` function to assess how many unique values there are for all the other variables in our data set? Hint: use `n_distinct()`.

```
df %>% summarize_if(_____, _____)
```

Now, if you group by "Transaction_Status", are there approximately equal distributions of unique values across all the variables? In other words, for transactions that are approved versus declined, do we have near equal representation of observations across each variable?

```
df %>%
  group_by(_____) %>%
  summarize_if(_____, _____)
```

Exercise 8

Create a function `convert_to_qtr()` that converts monthly values to quarters. This function should take a vector of character month values ("Jan", "Feb", ..., "Dec") and convert to "Q1", "Q2", "Q3", or "Q4". Do it such that:

- If the month input is Jan-Mar, then the function returns "Q1"
- If the month input is Apr-Jun, then the function returns "Q2"
- If the month input is Jul-Sep, then the function returns "Q3"
- If the month input is Oct-Dec, then the function returns "Q4"

Note, there is a function in the **lubridate** package (`quarter()`) that can accomplish this but I want you to use `case_when()` within the function instead.

Once you've created this function you should be able to test is on the following vector and get the same results:

```
example_months <- c("Jan", "Mar", "May", "May", "Aug", "Nov", "Nov", "Dec")
convert_to_qtr(example_months)
## [1] "Q1" "Q1" "Q2" "Q2" "Q3" "Q4" "Q4" "Q4"
```

Now, use this function you created above in a `mutate()` statement to create a new variable called "Qtr" in the data frame. How many observations do you have in each quarter?

Exercise 9

Take some time to understand the `sw_people` data set provided by the the **repurrrsive** package (`?sw_people`). Using a `map_xxx()` variant, extract the name of each Star Wars character in the `sw_people` data set. Hint: the first element of each list item is the characters name.

```
sw_people %>% map_xxx(________)
```

Exercise 10

Using the `sw_people` data set, find the number of films each Star Wars characters appears in. Be sure to use the most appropriate `map_xxx()` variant.

```
sw_people %>% map_xxx(________)
```

Exercise 11

Create a plot that shows the number of films each Star Wars character has been in. To do so, fill in the blanks below and your plot should look similar to one below the following code chunk.

```
sw_people %>%
  map_chr(_____) %>%                          # extract the names
  set_names(sw_people, nm = .) %>%            # set the list names as character names
  map_df(_____) %>%                           # extract number of films and make a data frame
  pivot_longer(                               # tidy data frame
     cols = everything(),
```

```
    names_to = "Character",
    values_to = "Films"
    ) %>%
ggplot(aes(Films, reorder(Character, Films))) +  # plot results
geom_point()
```