**##### EXTRA R EXERCISES – R FROM SCRATCH – R – LADIES #####**

R contains thousands of packages, each of them offering functions that do something useful. The package that is most commonly used (by far!) for data manipulation is called **`dplyr`** (pronounced *di-pl-a-yer*). Every package first has to be installed to your local machine (only once) and then loaded to your workspace (every time you open R). Let's install and load dplyr.

**### DPLYR PACKAGE ###**

```
install.packages("dplyr")
library(dplyr)
```

**### LOADING DATA ###**

Now, before we practice using dplyr, let's load the data we're going to work with:

1. Read about one of the most popular R's in-built datasets mtcars, using `help()` function or ?
2. Explore dataset's structure and summary statistics. Pay attention to data types.

```
?mtcars
data("mtcars")
str(mtcars)
summary(mtcars)
```

**### DATA MANIPULATION ###**

Let's learn about some of the most useful dplyr's functions:

1. Find out about the following dplyr's functions: `mutate()` and `filter()` (try looking up e.g. `?dplyr::mutate`, etc.). What do they do? How do they work? What's the syntax?

2. Create a new data.frame (`new_mtcars`) that contains only observations where number of cylinders is greater or equal 6.

3. We're going to create a new variable, `cyl_desc`, that will put cyl in three buckets dependent on the value of `cyl`: low, medium and high (greater or equal 2, between 3 and 5 and greater or equal 6, respectively). For this reason let's learn a bit about `ifelse()` statement (use R's available help tools).

4. Create `cyl_desc` using `mutate()`.

5. Check the structure of `new_mtcars` – what's the class of the new variable? Turn it to factor.

```
?dplyr::mutate

new_mtcars <- filter(mtcars, cyl >=6)

new_mtcars <- mutate(new_mtcars,
                     cyl_desc = ifelse(cyl <=2, "low",
                                       ifelse(cyl >= 6, "high",
"medium"))
                     )

str(new_mtcars)
new_mtcars$cyl_desc <- as.factor(new_mtcars$cyl_desc)
```

**### JOINING DATASETS ###**

It's time to join the new dataset to the original mtcars. However, there are many ways to do it!

1. Read about available joins in dplyr (e.g. try `?dplyr::join`)
2. Create a new dataset inner_df where you use `inner_join()` to join `mtcars` and `new_mtcars` together.
3. Check the structure of `inner_df` – how many variables and observations are there? What variables can you see? Are there any missing values?
4. Create a new dataset `left_df` where you use `left_join()` to join `mtcars` and `new_mtcars` together.
5. Check the structure of `left_df` – how many variables and observations are there? What variables can you see? Are there any missing values?
6. Create a new dataset `right_df` where you use `right_join()` to join `mtcars` and `new_mtcars` together.
7. Check the structure of `right_df` – how many variables and observations are there? What variables can you see? Are there any missing values?
8. Create a new dataset `anti_df` where you use `anti_join()` to join `mtcars` and `new_mtcars` together.
9. Check the structure of `anti_df` – how many variables and observations are there? What variables can you see? Are there any missing values?

```
inner_df <- inner_join(mtcars, new_mtcars)
str(inner_df)
```

```r
left_df <- left_join(mtcars, new_mtcars)
str(left_df)
summary(left_df)
head(left_df)

right_df <- right_join(mtcars, new_mtcars)
str(right_df)

?dplyr::anti_join
anti_df <- anti_join(mtcars, new_mtcars)
str(anti_df)
```