Project 1 (Exploratory Data Analysis)

Part 1 - There are 20 files with .dat extension. You have to read all the files in to single data frame.

Ans -

```
setwd("~/Desktop/iris.dat")
file_names = list.files(getwd())
file_names = file_names[grepl(".dat",file_names)]
file_names
dataframe = read.csv("~/Desktop/iris.dat/001.dat", header =F)
str(dataframe)
dataframe = lapply(file_names, read.csv, header=F, stringsAsFactors = F, skip =9)
dataframe = do.call(rbind,dataframe)
str(dataframe)
```

output -

```
> setwd("~/Desktop/iris.dat")
> file_names = list.files(getwd())
> file_names = file_names[grepl(".dat",file_names)]
> file_names
 [1] "001.dat" "002.dat" "003.dat" "004.dat" "005.dat" "006.dat" "007.dat" "008.dat"
 [9] "009.dat" "010.dat" "011.dat" "012.dat" "013.dat" "014.dat" "015.dat" "016.dat"
[17] "017.dat" "018.dat" "019.dat" "020.dat"
> dataframe = read.csv("~/Desktop/iris.dat/001.dat", header =F)
> str(dataframe)
'data.frame':   416 obs. of  2 variables:
 $ V1: Factor w/ 64 levels " Iris-virginica}",..: 11 6 7 4 5 3 1 9 2 10 ...
 $ V2: Factor w/ 47 levels "","," 2.5]"," 4.4]",..: 1 5 3 4 2 6 1 8 7 1 ...
> dataframe = lapply(file_names, read.csv, header=F, stringsAsFactors = F, skip =9)
> dataframe = do.call(rbind,dataframe)
> str(dataframe)
'data.frame':   1500 obs. of  5 variables:
 $ V1: chr  "4.7" "4.6" "5.0" "5.4" ...
 $ V2: chr  "3.2" "3.1" "<null>" "3.9" ...
 $ V3: chr  "1.3" "1.5" "1.4" "1.7" ...
 $ V4: chr  "0.2" "0.2" "0.2" "0.4" ...
 $ V5: chr  "Iris-setosa" "Iris-setosa" "Iris-setosa" "Iris-setosa" ...
>
```

Part II

The data is present in xml format, with file name, iris.Xml. Your task is to read the XML data and store it in the data frame df.

Ans -

```
library(XML)
xml.url <- "~/Desktop/iris.xml"
xmlfile <- xmlTreeParse(xml.url)
class(xmlfile)
xmltop = xmlRoot(xmlfile)
df <- xmlToDataFrame("~/Desktop/iris.xml")
print(df)
```

Output -

```
> library(XML)
> xml.url <- "~/Desktop/iris.xml"
> xmlfile <- xmlTreeParse(xml.url)
> class(xmlfile)
[1] "XMLDocument"         "XMLAbstractDocument"
> xmltop = xmlRoot(xmlfile)
>
> df <- xmlToDataFrame("~/Desktop/iris.xml")
> print(df)
  Sepal.Length Sepal.Width Petal.Length Petal.Width           Species
1          4.4         3.5          1.5         0.6            setosa
2          5.0         3.2          1.7         0.3        versicolor
3          4.8         3.0          1.1         0.4         virginica
>
```

<u>Part III)</u>

Convert the iris data into the JSON format and read the data in JSON format and convert it into data frame "iris_data"

Ans -

```
library("rjson")
 write_json <- function(df,"~/Desktop/iris.dat" , df_type = "rows", raw_type = "mongo"){
require(readr)
require(jsonlite)
df <- toJSON(dataframe = df_type, raw = raw_type)
write_lines("~/Desktop/iris.dat")
}
json_data <- fromJSON(file = "df")
iris_data <- as.data.frame(json_data)
print(iris_data)
```

Output -

json_data have class as "json".

```
> class(json_data)
[1] "json"
```

After converting json_data into iris_data, it will print 416 objects in iris_data with class as data frame.

```
> class(iris_data)
[1] "data.frame"
```

Part IV)

Use dplyr function on the data iris_data. Implement select, match, filter, arrange, rename, and mutate function on the iris_data

Ans :- Deplyr-

By constraining your options, deplyr simplifies how you can think about common data manipulation tasks.

It provides simple “verbs”, functions that correspond to the most common data manipulation tasks, to help you translate those thoughts into code.

It uses efficient data storage backends, so you spend less time waiting for the computer**.**

Filter-

```
virginica <- filter(iris_data, species == "virginica")
head(virginica)
```

Select -

```
select(iris, starts_with("Petal"))
select(iris, ends_with("Width"))
select(iris, contains("etal"))
select(iris, matches(".t."))
select(iris, Petal.Length, Petal.Width)
selected <- select(iris_data, sepal.length, sepal.width, petal.length)
# select all columns from sepal.length to petal.length
selected2 <- select(iris_data, sepal.length:petal.length)
head(selected, 3)
```

Match -

```
newCol <- match(petal.width, sepal.width, no match =0)
```

Arrange-

```
newCol <- arrange(newCol, petal.width)
head(newCol)
```

Rename -

```
rename(iris_data, petal_length = Petal.Length)
```

Mutate -

```
newCol <- mutate(iris_data, greater.half = sepal.width > 0.5 * sepal.length)
tail(newCol)
```

Part V) Print the summary of iris_data

Ans: -

```
> summary(iris_data)
                V1             V2
 Iris-setosa    : 45          :139
 Iris-versicolor: 45   0.2    : 26
 Iris-virginica : 45   <null> : 21
 <null>         : 33   3.0    : 20
 5.1            : 13   3.2    : 12
 5.6            : 11   1.3    : 11
 (Other)        :224   (Other):187
```