```
# Assignment: ASSIGNMENT 2
# Name: Koby-Hercsky, Theodore
# Date: 2021-03-25

## Check your current working directory using `getwd()`
getwd()

## List the contents of the working directory with the `dir()` function
dir()
#[1] "dsc520" "GitHub"
## If the current directory does not contain the `data` directory, set the
## working directory to project root folder (the folder should contain the `data` directory
## Use `setwd()` if needed
etwd("~/Downloads/DSC520/dsc520/data")

## Load the file `data/tidynomicon/person.csv` to `person_df1` using `read.csv`
## Examine the structure of `person_df1` using `str()`
person_df1 <-read.csv('person.csv', header = TRUE, sep = ",")
str(person_df1)
## R interpreted names as factors, which is not the behavior we want
## Load the same file to person_df2 using `read.csv` and setting `stringsAsFactors` to `FALSE`
## Examine the structure of `person_df2` using `str()`
person_df2 <-read.csv('person.csv', stringsAsFactors = FALSE, sep = ",")
str(person_df2)
## Read the file `data/scores.csv` to `scores_df`
## Display summary statistics using the `summary()` function
scores_df <-read.csv('scores.csv', stringsAsFactors = FALSE, sep = ",")
summary(scores_df)
## Load the `readxl` library
library(`readxl`)

## Using the excel_sheets() function from the `readxl` package,
## list the worksheets from the file `data/G04ResultsDetail2004-11-02.xls`
excel_sheets('G04ResultsDetail2004-11-02.xls')

## Using the `read_excel` function, read the Voter Turnout sheet
## from the `data/G04ResultsDetail2004-11-02.xls`
## Assign the data to the `voter_turnout_df1`
## The header is in the second row, so make sure to skip the first row
## Examine the structure of `voter_turnout_df1` using `str()`
read_excel('G04ResultsDetail2004-11-02.xls', sheet = "Voter Turnout", skip = 1)
# created a data frame and excluded the first row from the voter turnout data sheet
voter_turnout_df1 <- data.frame(read_excel('G04ResultsDetail2004-11-02.xls', sheet = "Voter Turnout", skip = 1))

## Using the `read_excel()` function, read the Voter Turnout sheet
## from `data/G04ResultsDetail2004-11-02.xls`
## Skip the first two rows and manually assign the columns using `col_names`
## Use the names "ward_precint", "ballots_cast", "registered_voters", "voter_turnout"
## Assign the data to the `voter_turnout_df2`
## Examine the structure of `voter_turnout_df2` using `str()`
# Created a vector titled Voter_Vector
voter_vector <- c("ward_precint", "ballots_cast", "registered_voters", "voter_turnout")
# Read the voter turnout and removed the first two rows and inserted the voter_vector
read_excel('G04ResultsDetail2004-11-02.xls', sheet = "Voter Turnout", skip = 2, col_names = voter_vector)
# Created the datat frame from the voter turnout sheet and removed the first two rows and inserted the Voter_vector
voter_turnout_df2 <- data.frame(read_excel('G04ResultsDetail2004-11-02.xls', sheet = "Voter Turnout", skip = 2,
col_names = voter_vector))

## Load the `DBI` library
library(DBI)

## Create a database connection to `data/tidynomicon/example.db` using the dbConnect() function
## The first argument is the database driver which in this case is `RSQLite::SQLite()`
## The second argument is the path to the database file
## Assign the connection to `db` variable
# Use the library to connect to RSQLite
library(RSQLite)
# Use dbdrive to specify the driver SQLite
drv <- dbDriver('SQLite')
class(drv)
# Establish connection using dbconnect to connect to the example database
db <- dbConnect(drv, 'example.db')

## Query the Person table using the `dbGetQuery` function and the
## `SELECT * FROM PERSON;` SQL statement
## Assign the result to the `person_df` variable
## Use `head()` to look at the first few rows of the `person_df` dataframe
person_df <- dbGetQuery(db, "SELECT * Person", stringsAsFactors=FALSE)
head(person_df)

## List the tables using the `dbListTables()` function
## Assign the result to the `table_names` variable
dbListTables(db)
table_names <- dbListTables(db)

## Read all of the tables at once using the `lapply` function and assign the result to the `tables` variable
## Use `table_names`, `dbReadTable`, and `conn = db` as arguments
## Print out the tables
lapply(X = db, FUN = table_names)
print(tables)
## Use the `dbDisconnect` function to disconnect from the database
dbDisconnect(db)

## Import the `jsonlite` library
library(jsonlite)

## Convert the scores_df dataframe to JSON using the `toJSON()` function
scores_df <-toJSON(scores_df)

## Convert the scores dataframe to JSON using the `toJSON()` function with the `pretty=TRUE` option
scores_df <-toJSON(scores_df, pretty = TRUE)
```