M55-Query & Report

learningSpoonsR@gmail.com

I. query_report.Rmd

Header

```
# ---
# title: "M55 - Query & Report"
# output:
#   flexdashboard::flex_dashboard:
#     source_code: embed
# runtime: shiny
# ---
```

```
# ```{r global, include=FALSE}
# library(readxl)
# library(dplyr)
# library(ggplot2)
# library(DT)
# library(tidyr)
# library(data.table)
# library(rmarkdown)
# library(stringr)
# dataset <- read_excel("input_data/retail.xlsx")
# dataset <- dataset %>%
#   select(`Order ID`, `Order Date`, Category, `Sub-Category`,
#          `Product Name`, Quantity, Profit)
# dataset$`Order Date` <- as.Date(dataset$`Order Date`)
# ```
```

- **shinyApp**는 **ui**와 **server** 두 개의 input을 받습니다.
- **ui**는 'User Interface'에 관한 객체로서 input panel들과 다른 디자인 요소가 포함됩니다.
- **server**는 **ui**에서 받아들인 **input**을 이용해서 **output**을 만들어 냅니다.

Page 정의하고 ui만들기

```
Main
==========================================================================

###

# Part 1. create `ui`
ui <- fillPage(
  fillCol(flex = c(NA, 1),
    inputPanel(
      # selectInput("region", "Region:", choices = colnames(WorldPhones)),
      dateRangeInput(inputId = "dt_rng", label = "Date Range",
                     min = min(dataset$`Order Date`), start = max(dataset$`Order Date`)-14,
                     max = max(dataset$`Order Date`), end = max(dataset$`Order Date`)),
      radioButtons(inputId = "class", label = "Category",
                   choices = c("Furniture", "Office Supplies", "Technology")),
      downloadButton("downloadData", label = "Save to CSV"),
      actionButton("render_docx", "Report using csv")
    ),
    DT::dataTableOutput("dataset")
  )
)
```

server만들기 그리고 ui와 server합하기

```
# Part 2. create `server`
server <- function(input, output) {
  # Data Filtering using input values
  datasetFiltered <- reactive({
    dataset %>%
      filter(`Order Date` >= input$dt_rng[1] & `Order Date` <= input$dt_rng[2]) %>%
      filter(Category == input$class)
  })
  # Display the filtered table
  output$dataset <- DT::renderDataTable({
    datasetFiltered()
  })
  # Download filtered table to .csv
  output$downloadData <- downloadHandler(
    filename = function() {
      "dataset_filtered.csv"
    },
    content = function(file) {
      write.csv(datasetFiltered(), file)
    }
  )
```

(이어짐)

```
  # CONTINUED
  # Use the saved .csv to render a docx
  observeEvent(input$render_docx, {
    # library(beepr); beep() # simple way to check if this block is responding
    rmarkdown::render(
      input = "M55-summary_docx.Rmd",
      output_file =
        paste0("summary_report_",
               Sys.time() %>%
                 strptime("%Y-%m-%d %H:%M:%OS") %>%
                 str_replace_all(":", "-") %>%
                 str_replace_all(" ", "-"),
               ".docx"),
      encoding = "UTF-8")
  })
}
```

```
# Part 3. use `shinyApp()` to combine `ui` and `server`
shinyApp(ui, server, options = list(height = 600))
```

II. References

1. `flexdashboard`에서 `shiny`를 사용하는 법
   - https://rmarkdown.rstudio.com/flexdashboard/shiny.html
   - 예제
     - https://beta.rstudioconnect.com/jjallaire/shiny-embedding
     - `jjallaire`는 Rstudio의 CEO입니다!
2. `shiny`에서 `DataTables`을 사용하는 법
   - `M12` appendix 참조
   - https://shiny.rstudio.com/articles/datatables.html
3. ~~`DataTables`에서 객체를 `csv`로 저장하는 법~~
   - 아래 링크에서 '2. Buttons'
   - https://rstudio.github.io/DT/extensions.html
   - `shiny`환경에서는 잘 안되기 때문에 `shiny`의 `downloadHandler()`를 이용하자.
4. `shiny`의 `downloadHandler()` 사용하는 법
   - https://shiny.rstudio.com/reference/shiny/0.14/downloadHandler.html
   - 예제: https://shiny.rstudio.com/gallery/download-file.html
   - ~~Rstudio 나랑 장난하니?~~
     - 'Note that the download button does not work well in the RStudio viewer.'
     - 'Open in Browser'를 눌러주세요.
     - https://stackoverflow.com/questions/25984138/shiny-app-downloadhandler-does-not-produce-a-file
5. `actionButton`을 이용한 docx 렌더링
   - https://shiny.rstudio.com/articles/action-buttons.html
   - 바탕화면 폴더 이름 인식하기
     - https://stackoverflow.com/questions/25886987/find-the-desktop-path-in-r
   - `M28` 참조