```
# Interactive Web Frameworks Workshop
#
# Part 1 - Plotly
#
# 1) Generate Plotly scatterplot using 'iris' dataset with '~Sepal.Length' as x
variable and '~Petal.Length' as y variable.
#   a) Add color variable to depend on '~Species'.
#   b) Add size variable to depend on '~Petal.Width'.
#
# Answer:
  plot_ly(iris, x = ~Sepal.Length, y = ~Petal.Length, color=~Species,
size=~Petal.Width, type='scatter', mode='markers')


# 2) Email a copy of the Figure from 1) to yourself. Verify that the plot can be
opened.
# hint: You will be sending a folder.
#
# Answer:
#   Figure can be shared by sending entire folder where the 'index.html' resides
instead of just the file alone.
#
# 3) Create bar-chart using the 'iris' dataset with '~Sepal.Width' as y
variable.
#
# Answer:
  plot_ly(iris, y = ~Sepal.Width, type='bar')

# 4) Combine figures from 1) and 3) into one figure. Hint: subplot(p1,p2)
#
# Answer:
  p1<- plot_ly(iris, x = ~Sepal.Length, y = ~Petal.Length, color=~Species,
size=~Petal.Width, type='scatter', mode='markers')
  p2 <- plot_ly(iris, y = ~Sepal.Width, type='bar')
  subplot(p1,p2)

# 5) Change the dots from Figure 1) to 'cross' symbols. Hint1:
marker=list(symbol= ) Hint2: https://plot.ly/r/reference/#scatter-marker-symbol
#
# Answer:
  plot_ly(iris, x = ~Sepal.Length, y = ~Petal.Length, color=~Species,
size=~Petal.Width, type='scatter', mode='markers',marker=list(symbol='cross'))
```

Part 2 - Shiny

1) Create Shiny App that take number of rows as input and displays n rows of the ΓÇÿirisΓÇÖ table.

Answer:

```
  library(shiny)
  # Define UI for application that draws a histogram
  ui <- shinyUI(fluidPage(
    sidebarPanel(
      numericInput('nrows', 'Number of Rows', 3, min = 1, max = 150)
    ),
    mainPanel(
      tableOutput("table")
    )
  ))

  server <- shinyServer(function(input, output) {
    output$table <- renderTable({
      head(iris, n = input$nrows)
    })
  })

  shinyApp(ui = ui, server = server)
```

```
# 2) Create the following Shiny App(example below):
#   a) Takes in number value using a sliderInput with (min = 1, max = 50, value
= 30).
#   b) Plot histogram using the ΓÇÿfaithful$waitingΓÇÖ data.
```

Answer:

```
  library(shiny)

  # Define UI for application that draws a histogram
  ui <- shinyUI(fluidPage(

    # Application title
    titlePanel("Hello Shiny!"),

    # Sidebar with a slider input for the number of bins
    sidebarLayout(
      sidebarPanel(
        sliderInput("bins",
                    "Number of bins:",
                    min = 1,
                    max = 50,
                    value = 30)
      ),

      # Show a plot of the generated distribution
      mainPanel(
        plotOutput("distPlot")
      )
    )
  ))

  server <- shinyServer(function(input, output) {

    # Expression that generates a histogram. The expression is
    # wrapped in a call to renderPlot to indicate that:
    #
    #  1) It is "reactive" and therefore should be automatically
    #     re-executed when inputs change
    #  2) Its output type is a plot

    output$distPlot <- renderPlot({
      x    <- faithful[, 2]  # Old Faithful Geyser data
      bins <- seq(min(x), max(x), length.out = input$bins + 1)

      # draw the histogram with the specified number of bins
      hist(x, breaks = bins, col = 'darkgray', border = 'white')
    })

  })


  shinyApp(ui = ui, server = server)
```