**#Get Started with Shiny**

**-make hello world :**

**library(shiny)**

**# Define UI for application that draws a histogram**

**ui <- fluidPage("Hello World !!!")**

**server <- function(input, output, session){**

**}**

**# Run the application**

**shinyApp(ui = ui, server = server)**

**-Membuat kotak input :**

**library(shiny)**

**ui <- fluidPage(**

**# CODE BELOW: Add a text input "name"**

**textInput("name", "Enter a name :")**

**)**

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

**}**

**shinyApp(ui = ui, server = server)**

**-Hello World App output :**

**library(shiny)**

**ui <- fluidPage(**

**textInput("name", "What is your name?"),**

**# CODE BELOW: Display the text output, greeting**

**# Make sure to add a comma after textInput()**

**textOutput("greeting")**

**)**

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

**# CODE BELOW: Render a text output, greeting**

**output$greeting <- renderText({**

**paste("Hello,", input$name)**

**})**

**}**

**shinyApp(ui = ui, server = server)**

**-Input nama otomatis :**

**library(shiny)**

**ui <- fluidPage(**

**# CODE BELOW: Add a text input "name"**

**textInput('name', 'Enter Name', 'David') -> (input nama, text yg ada di atasnya, nama otomatis)**

**)**

**server <- function(input, output, session) {**

**}**

**shinyApp(ui = ui, server = server)**

**-Add Output ggplot() :**

**library(shiny)**

**library(ggplot2)**

**ui <- fluidPage(**

**textInput('name', 'Enter Name', 'David'),**

**# CODE BELOW: Display the plot output named 'trend'**

**plotOutput('trend')**

**)**

**server <- function(input, output, session) {**

**# CODE BELOW: Render an empty plot and assign to output named 'trend'**

**output$trend <- renderPlot({**

**ggplot()**

**})**

**}**

**shinyApp(ui = ui, server = server)**

**-Mengatur letak plot :**

**library(shiny)**

**library(ggplot2)**

**ui <- fluidPage(**

**titlePanel("Baby Name Explorer"),**

**# CODE BELOW: Add a sidebarLayout, sidebarPanel, and mainPanel**

**sidebarLayout(**

**sidebarPanel(**

**textInput('name', 'Enter Name', 'David')**

**),**

**mainPanel(**

**plotOutput('trend')**

**)))**

**server <- function(input, output, session) {**

**output$trend <- renderPlot({**

**ggplot()**

**})**

**}**

**shinyApp(ui = ui, server = server)**

**-Update Output, and plot the graph :**

**library(shiny)**

**library(ggplot2)**

**library(babynames)**

**ui <- fluidPage(**

**titlePanel("Baby Name Explorer"),**

**sidebarLayout(**

**sidebarPanel(textInput('name', 'Enter Name', 'David')),**

**mainPanel(plotOutput('trend'))**

**)**

**)**

**server <- function(input, output, session) {**

**output$trend <- renderPlot({**

**# CODE BELOW: Update to display a line plot of the input name**

**data\_name <- subset(babynames, name==input$name**

**)**

**ggplot(data\_name) + geom\_line(aes(year, n, color = sex))**

**})**

**}**

**shinyApp(ui = ui, server = server)**

**#INPUT, OUTPUT AND LAYOUT**

-Input Function :

**\*selectInput(“inputId”, “label”, choices = c(“A”, “B”, “C”)**

-Slider Input :

**\*sliderInput(“inputId”, “label”, value = 1925, min = 1900, max = 2000)**

-put them all :

**\*ui <- fluidPage(**

**textInput(“name”, “Enter a Name:”),**

**selectInput(“animal”,”dog or cats?”,choices=c(“dogs”,”cats”)),**

**textOutput(“greeting”),**

**textOutput(“answer”))**

**\*server<- function(input, output, session){**

**output$greeting <- renderText({**

**paste(“Do you prefer dogs or cats,”,input$name, “?”)**

**})**

**output$answer <- renderText({**

**Paste(“I Prefer”,input$animal, “!”)**

**})**

**}**

**Selecting an input**

Shiny provides a wide variety of inputs that allows users to provide **text** (textInput, selectInput), **numbers** (numericInput, sliderInput), **booleans** (checkBoxInput, radioInput), and **dates** (dateInput, dateRangeInput).

-Add a select input exercise :

ui <- fluidPage(

  titlePanel("What's in a Name?"),

  # CODE BELOW: Add select input named "sex" to choose between "M" and "F"

  selectInput('sex', 'Select Sex', choices = c("F", "M")),

  # Add plot output to display top 10 most popular names

  plotOutput('plot\_top\_10\_names')

)

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

  # Render plot of top 10 most popular names

  output$plot\_top\_10\_names <- renderPlot({

    # Get top 10 names by sex and year

    top\_10\_names <- babynames %>%

      # MODIFY CODE BELOW: Filter for the selected sex

      filter(sex == input$sex) %>%

      filter(year == 1900) %>%

      top\_n(10, prop)

    # Plot top 10 names by sex and year

    ggplot(top\_10\_names, aes(x = name, y = prop)) +

      geom\_col(fill = "#263e63")

  })

}

shinyApp(ui = ui, server = server)

-Add table ouput

**library(shiny)**

**library(babynames)**

**library(dplyr)**

**ui <- fluidPage(**

**titlePanel("What's in a Name?"),**

**# Add select input named "sex" to choose between "M" and "F"**

**selectInput('sex', 'Select Sex', choices = c("F", "M")),**

**# Add slider input named "year" to select year between 1900 and 2010**

**sliderInput('year', 'Select Year', min = 1900, max = 2010, value = 1900),**

**# CODE BELOW: Add table output named "table\_top\_10\_names"**

**tableOutput("table\_top\_10\_names")**

**)**

**server <- function(input, output, session){**

**# Function to create a data frame of top 10 names by sex and year**

**top\_10\_names <- function(){**

**top\_10\_names <- babynames %>%**

**filter(sex == input$sex) %>%**

**filter(year == input$year) %>%**

**top\_n(10, prop)**

**}**

**# CODE BELOW: Render a table output named "table\_top\_10\_names"**

**output$table\_top\_10\_names <- renderTable({**

**top\_10\_names()**

**})**

**}**

**shinyApp(ui = ui, server = server)**

-Add an interactive table output :

**library(shiny)**

**library(DT)**

**library(dplyr)**

**library(babynames)**

**ui <- fluidPage(**

**titlePanel("What's in a Name?"),**

**# Add select input named "sex" to choose between "M" and "F"**

**selectInput('sex', 'Select Sex', choices = c("M", "F")),**

**# Add slider input named "year" to select year between 1900 and 2010**

**sliderInput('year', 'Select Year', min = 1900, max = 2010, value = 1900),**

**# MODIFY CODE BELOW: Add a DT output named "table\_top\_10\_names"**

**DT::DTOutput('table\_top\_10\_names')**

**)**

**server <- function(input, output, session){**

**top\_10\_names <- function(){**

**babynames %>%**

**filter(sex == input$sex) %>%**

**filter(year == input$year) %>%**

**top\_n(10, prop)**

**}**

**# MODIFY CODE BELOW: Render a DT output named "table\_top\_10\_names"**

**output$table\_top\_10\_names <- DT::renderDT({**

**top\_10\_names()**

**})**

**}**

**shinyApp(ui = ui, server = server)**

-Add interactive plotly :

ui <- fluidPage(

  selectInput('name', 'Select Name', top\_trendy\_names$name),

  # CODE BELOW: Add a plotly output named 'plot\_trendy\_names'

  plotly::plotlyOutput("plot\_trendy\_names")

)

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

  # Function to plot trends in a name

  plot\_trends <- function(){

     babynames %>%

      filter(name == input$name) %>%

      ggplot(aes(x = year, y = n)) +

      geom\_col()

  }

  # CODE BELOW: Render a plotly output named 'plot\_trendy\_names'

output$plot\_trendy\_names <- plotly::renderPlotly({

    plot\_trends()

})

}

shinyApp(ui = ui, server = server)

-Layout and Themes :

Membuat 2 tab layout

**ui <- fluidPage(**

**titlePanel(“Histogram”),**

**sidebarLayout(**

**sidebarPanel(sliderInput(‘nb\_bins’, ‘# Bins’, 5,10,5)),**

**mainPanel(**

**tabsetPanel(**

**tabPanel(‘Waiting,**

**plotOutput(‘hist\_waiting’)),**

**tabPanel(‘Eruption’,**

**plotOutput(‘hist\_eruption’))**

**))))**

**server <- function(input, output, session) {**

**output$hist\_Waiting <- renderPlot({**

**hist(faithful$waiting, breaks = input$nb\_bins, col = ‘steelblue’)**

**})**

**output$hist\_Eruption <- renderPlot({**

**hist(faithful$eruption, breaks = inpu$nb\_bins, col = ‘steelblue’)**

**})**

**}**

**shinyApp(ui = ui, server = server)**

Membuat dropdown theme selector

**ui <- fluidPage(**

**titlePanel(“Histogram”),**

**shinythemes :: themeSelector(),**

**sidebarLayout(**

**sidebarPanel(sliderInput(‘nb\_bins’, ‘# Bins’, 5,10,5)),**

**mainPanel(**

**tabsetPanel(**

**tabPanel(‘Waiting,**

**plotOutput(‘hist\_waiting’)),**

**tabPanel(‘Eruption’,**

**plotOutput(‘hist\_eruption’))**

**))))**

**server <- function(input, output, session) {**

**output$hist\_Waiting <- renderPlot({**

**hist(faithful$waiting, breaks = input$nb\_bins, col = ‘steelblue’)**

**})**

**output$hist\_Eruption <- renderPlot({**

**hist(faithful$eruption, breaks = inpu$nb\_bins, col = ‘steelblue’)**

**})**

**}**

**shinyApp(ui = ui, server = server)**

-Applied superhero theme

**ui <- fluidPage(**

**titlePanel(“Histogram”),**

**theme = shinythemes ::shinytheme(‘superhero’)**

**sidebarLayout(**

**sidebarPanel(sliderInput(‘nb\_bins’, ‘# Bins’, 5,10,5)),**

**mainPanel(**

**tabsetPanel(**

**tabPanel(‘Waiting,**

**plotOutput(‘hist\_waiting’)),**

**tabPanel(‘Eruption’,**

**plotOutput(‘hist\_eruption’))**

**))))**

**server <- function(input, output, session) {**

**output$hist\_Waiting <- renderPlot({**

**hist(faithful$waiting, breaks = input$nb\_bins, col = ‘steelblue’)**

**})**

**output$hist\_Eruption <- renderPlot({**

**hist(faithful$eruption, breaks = inpu$nb\_bins, col = ‘steelblue’)**

**})**

**}**

**shinyApp(ui = ui, server = server)**

-Tab Layout Exercise

ui <- fluidPage(

  sidebarLayout(

    sidebarPanel(

      selectInput('name', 'Select Name', top\_trendy\_names$name)

    ),

    mainPanel(

      # MODIFY CODE BLOCK BELOW: Wrap in a tabsetPanel

        # MODIFY CODE BELOW: Wrap in a tabPanel providing an appropriate label

        tabsetPanel(

          tabPanel('plot\_trendy\_names',plotly::plotlyOutput('plot\_trendy\_names')),

        # MODIFY CODE BELOW: Wrap in a tabPanel providing an appropriate label

        tabPanel('table\_trendy\_names',DT::DTOutput('table\_trendy\_names')

    )

  )

)))

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

  # Function to plot trends in a name

  plot\_trends <- function(){

     babynames %>%

      filter(name == input$name) %>%

      ggplot(aes(x = year, y = n)) +

      geom\_col()

  }

  output$plot\_trendy\_names <- plotly::renderPlotly({

    plot\_trends()

  })

  output$table\_trendy\_names <- DT::renderDT({

    babynames %>%

      filter(name == input$name)

  })

}

shinyApp(ui = ui, server = server)

-SideBar Layout Exercise :

ui <- fluidPage(

  # MODIFY CODE BELOW: Wrap in a sidebarLayout

    # MODIFY CODE BELOW: Wrap in a sidebarPanel

    sidebarLayout(

      sidebarPanel(

    selectInput('name', 'Select Name', top\_trendy\_names$name)),

    # MODIFY CODE BELOW: Wrap in a mainPanel

    mainPanel(plotly::plotlyOutput('plot\_trendy\_names'),

    DT::DTOutput('table\_trendy\_names'))

))

# DO NOT MODIFY

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

  # Function to plot trends in a name

  plot\_trends <- function(){

     babynames %>%

      filter(name == input$name) %>%

      ggplot(aes(x = year, y = n)) +

      geom\_col()

  }

  output$plot\_trendy\_names <- plotly::renderPlotly({

    plot\_trends()

  })

  output$table\_trendy\_names <- DT::renderDT({

    babynames %>%

      filter(name == input$name)

  })

}

shinyApp(ui = ui, server = server)

-Themes Exercise :

ui <- fluidPage(

  # CODE BELOW: Add a titlePanel with an appropriate title

  titlePanel('Top Trendy Names'),

  # REPLACE CODE BELOW: with theme = shinythemes::shinytheme("<your theme>")

  theme = shinythemes::shinytheme('superhero'),

  sidebarLayout(

    sidebarPanel(

      selectInput('name', 'Select Name', top\_trendy\_names$name)

    ),

    mainPanel(

      tabsetPanel(

        tabPanel('Plot', plotly::plotlyOutput('plot\_trendy\_names')),

        tabPanel('Table', DT::DTOutput('table\_trendy\_names'))

      )

    )

  )

)

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

  # Function to plot trends in a name

  plot\_trends <- function(){

     babynames %>%

      filter(name == input$name) %>%

      ggplot(aes(x = year, y = n)) +

      geom\_col()

  }

  output$plot\_trendy\_names <- plotly::renderPlotly({

    plot\_trends()

  })

  output$table\_trendy\_names <- DT::renderDT({

    babynames %>%

      filter(name == input$name)

  })

}

shinyApp(ui = ui, server = server)

-Add Input tambahan :

**\*selectInput(‘continent’, ‘Select Continent’, unique(gapminder$continent))**

**\*sliderInput(‘year’, ‘Select Year’, 1952, 2007, 1992, step =5)** -> step year every 5

-Multilingual Greeting Exercise :

ui <- fluidPage(

  selectInput('greeting', 'Select greeting', choices = c('Hello', 'Bonjour'), 'Bonjour'),

  textInput('name', 'Enter your name', 'Kaelen'),

textOutput('greeting'))

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

  output$greeting <- renderText({

      paste(input$greeting, input$name)

  })

}

shinyApp(ui = ui, server = server)

-Popular Baby Names Exercise :

ui <- fluidPage(

    titlePanel('Most Popular Names'),

    sidebarLayout(

        sidebarPanel(

  selectInput('sex','Select Sex', choices = c("M","F")),

  sliderInput('year', 'Select Year', 1880, 2017,1900)),

  mainPanel(plotOutput('plot')))

)

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

  output$plot <- renderPlot({

      top\_names\_by\_sex\_year <- get\_top\_names(input$sex, input$name)

      ggplot(top\_names\_by\_sex\_year, aes(year, prop)) +geom\_col()

  })

}

shinyApp(ui = ui, server = server)

-Popular babynames redux :

ui <- fluidPage(

  titlePanel("Most Popular Names"),

  sidebarLayout(

    sidebarPanel(

      selectInput('sex', 'Select Sex', c("M", "F")),

      sliderInput('year', 'Select Year', 1880, 2013, 1900)

    ),

    mainPanel(

      tabsetPanel(

        tabPanel('Plot', plotOutput('plot')),

        tabPanel('Table', tableOutput('table'))

      )

    )

  )

)

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

  output$plot <- renderPlot({

    d <- get\_top\_names(input$year, input$sex)

    qplot(name, prop, data = d, geom = 'col')

  })

  output$table <- renderTable({

    get\_top\_names(input$year, input$sex)

  })

}

shinyApp(ui = ui, server = server)

**#REACTIVE PROGRAMMING**

-Add reactive expression exercise

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

  # CODE BELOW: Add a reactive expression rval\_bmi to calculate BMI

  rval\_bmi <- reactive({

    input$weight/(input$height^2)

  })

  output$bmi <- renderText({

    # MODIFY CODE BELOW: Replace right-hand-side with reactive expression

    bmi <- rval\_bmi()

    paste("Your BMI is", round(bmi, 1))

  })

  output$bmi\_range <- renderText({

    # MODIFY CODE BELOW: Replace right-hand-side with reactive expression

    bmi <- rval\_bmi()

    bmi\_status <- cut(bmi,

      breaks = c(0, 18.5, 24.9, 29.9, 40),

      labels = c('underweight', 'healthy', 'overweight', 'obese')

    )

    paste("You are", bmi\_status)

  })

}

ui <- fluidPage(

  titlePanel('BMI Calculator'),

  sidebarLayout(

    sidebarPanel(

      numericInput('height', 'Enter your height in meters', 1.5, 1, 2),

      numericInput('weight', 'Enter your weight in Kilograms', 60, 45, 120)

    ),

    mainPanel(

      textOutput("bmi"),

      textOutput("bmi\_range")

    )

  )

)

shinyApp(ui = ui, server = server)

-Add another reactive expression exercise :

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

  rval\_bmi <- reactive({

    input$weight/(input$height^2)

  })

  # CODE BELOW: Add a reactive expression rval\_bmi\_status to

  # return health status as underweight etc. based on inputs

  rval\_bmi\_status <- reactive({

    cut(rval\_bmi(),

      breaks = c(0, 18.5, 24.9, 29.9, 40),

      labels = c('underweight', 'healthy', 'overweight', 'obese')

    )

  })

  output$bmi <- renderText({

    bmi <- rval\_bmi()

    paste("Your BMI is", round(bmi, 1))

  })

  output$bmi\_status <- renderText({

    # MODIFY CODE BELOW: Replace right-hand-side with

    # reactive expression rval\_bmi\_status

    bmi\_status <- rval\_bmi\_status()

    paste("You are", bmi\_status)

  })

}

ui <- fluidPage(

  titlePanel('BMI Calculator'),

  sidebarLayout(

    sidebarPanel(

      numericInput('height', 'Enter your height in meters', 1.5, 1, 2),

      numericInput('weight', 'Enter your weight in Kilograms', 60, 45, 120)

    ),

    mainPanel(

      textOutput("bmi"),

      textOutput("bmi\_status")

    )

  )

)

shinyApp(ui = ui, server = server)

-Add an observer to display notification exercise :

ui <- fluidPage(

  textInput('name', 'Enter your name')

)

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

  # CODE BELOW: Add an observer to display a notification

  # 'You have entered the name xxxx' where xxxx is the name

observe({

  showNotification(

    paste("You have entered the name", input$name)

  )

})

}

shinyApp(ui = ui, server = server)

observe hanya tampilan previewnya saja tidak tampil di browser

-Stop Reaction with isolate

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

  rval\_bmi <- reactive({

    input$weight/(input$height^2)

  })

  output$bmi <- renderText({

    bmi <- rval\_bmi()

    # MODIFY CODE BELOW:

    # Use isolate to stop output from updating when name changes.

    paste("Hi", isolate({input$name}), ". Your BMI is", round(bmi, 1))

  })

}

ui <- fluidPage(

  titlePanel('BMI Calculator'),

  sidebarLayout(

    sidebarPanel(

      textInput('name', 'Enter your name'),

      numericInput('height', 'Enter your height (in m)', 1.5, 1, 2, step = 0.1),

      numericInput('weight', 'Enter your weight (in Kg)', 60, 45, 120)

    ),

    mainPanel(

      textOutput("bmi")

    )

  )

)

shinyApp(ui = ui, server = server)

ketika input nama, hasil tidak akan langsung keluar krn telah diisolate. Hasil akan keluar setelah diisi angka bmi nya.

-Delay reactions with eventReactive :

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

  # MODIFY CODE BELOW: Use eventReactive to delay the execution of the

  # calculation until the user clicks on the show\_bmi button (Show BMI)

  rval\_bmi <- eventReactive(input$show\_bmi,{

    input$weight/(input$height^2)

  })

  output$bmi <- renderText({

    bmi <- rval\_bmi()

    paste("Hi", input$name, ". Your BMI is", round(bmi, 1))

  })

}

ui <- fluidPage(

  titlePanel('BMI Calculator'),

  sidebarLayout(

    sidebarPanel(

      textInput('name', 'Enter your name'),

      numericInput('height', 'Enter height (in m)', 1.5, 1, 2, step = 0.1),

      numericInput('weight', 'Enter weight (in Kg)', 60, 45, 120),

      actionButton("show\_bmi", "Show BMI")

    ),

    mainPanel(

      textOutput("bmi")

    )

  )

)

shinyApp(ui = ui, server = server)

eventReactive akan mendelay hasil, nnt akan muncu jika kita mengklik button

-Trigger Reactions with observeEvent()

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

  # MODIFY CODE BELOW: Wrap in observeEvent() so the help text

  # is displayed when a user clicks on the Help button.

  observeEvent(input$show\_help, {

     # Display a modal dialog with bmi\_help\_text

     # MODIFY CODE BELOW: Uncomment code

     showModal(modalDialog(bmi\_help\_text))

  })

  rv\_bmi <- eventReactive(input$show\_bmi, {

    input$weight/(input$height^2)

  })

  output$bmi <- renderText({

    bmi <- rv\_bmi()

    paste("Hi", input$name, ". Your BMI is", round(bmi, 1))

  })

}

ui <- fluidPage(

  titlePanel('BMI Calculator'),

  sidebarLayout(

    sidebarPanel(

      textInput('name', 'Enter your name'),

      numericInput('height', 'Enter your height in meters', 1.5, 1, 2),

      numericInput('weight', 'Enter your weight in Kilograms', 60, 45, 120),

      actionButton("show\_bmi", "Show BMI"),

      # CODE BELOW: Add an action button named "show\_help"

      actionButton("show\_help", "Help")

    ),

    mainPanel(

      textOutput("bmi")

    )

  )

)

shinyApp(ui = ui, server = server)

mentrigger perintah dengan klik tombol

-Convert height fro inch to cm

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

  # MODIFY CODE BELOW: Delay the height calculation until

  # the show button is pressed

  rval\_height\_cm <- eventReactive({

    input$height \* 2.54

  })

  output$height\_cm <- renderText({

    height\_cm <- rval\_height\_cm()

      paste("Your height in centimeters is", height\_cm, "cm")

    })

}

ui <- fluidPage(

  titlePanel("Inches to Centimeters Conversion"),

  sidebarLayout(

    sidebarPanel(

      numericInput("height", "Height (in)", 60),

      actionButton("show\_height\_cm", "Show height in cm")

    ),

    mainPanel(

      textOutput("height\_cm")

    )

  )

)

shinyApp(ui = ui, server = server)

**#BUILD SHINY APPS**

-choices from the column directly

**\*ui <- fluidPage(**

**\*selectInput(“shape”, “Choose a Shape:”, choices = unique(usa\_ufo\_sightings$shape)**

**))**

-Alien Sightings : add input

ui <- fluidPage(

  # CODE BELOW: Add a title

  titlePanel("UFO Sightings"),

  sidebarLayout(

    sidebarPanel(

      # CODE BELOW: One input to select a U.S. state

      # And one input to select a range of dates

      selectInput("state",

        "Choose a U.S. state:",

        choices = unique(usa\_ufo\_sightings$state)),

      dateRangeInput("dates", "Choose a date range:",

        start = "1920-01-01",

        end = "1950-01-01"

      )

    ),

    mainPanel()

  )

)

server <- function(input, output) {

}

shinyApp(ui, server)

-Alien Sightings : add output

server <- function(input, output) {

  # CODE BELOW: Create a plot output name 'shapes', of sightings by shape,

  # For the selected inputs

  output$shapes <- renderPlot({

    usa\_ufo\_sightings %>%

      filter(state == input$state,

             date\_sighted >= input$dates[1],

             date\_sighted <= input$dates[2]) %>%

      ggplot(aes(shape)) +

      geom\_bar() +

      labs(x = "Shape", y = "# Sighted")

  })

  # CODE BELOW: Create a table output named 'duration\_table', by shape,

  # of # sighted, plus mean, median, max, and min duration of sightings

  # for the selected inputs

  output$duration\_table <- renderTable({

    usa\_ufo\_sightings %>%

      filter(

        state == input$state,

        date\_sighted >= input$dates[1],

        date\_sighted <= input$dates[2]

      ) %>%

      group\_by(shape) %>%

      summarize(

        nb\_sighted = n(),

        avg\_duration = mean(duration\_sec),

        median\_duration = median(duration\_sec),

        min\_duration = min(duration\_sec),

        max\_duration = max(duration\_sec)

      )

  })

}

ui <- fluidPage(

  titlePanel("UFO Sightings"),

  sidebarLayout(

    sidebarPanel(

      selectInput("state", "Choose a U.S. state:", choices = unique(usa\_ufo\_sightings$state)),

      dateRangeInput("dates", "Choose a date range:",

                     start = "1920-01-01",

                     end = "1950-01-01")

    ),

    mainPanel(

      # Add plot output named 'shapes'

      plotOutput("shapes"),

      # Add table output named 'duration\_table'

      tableOutput("duration\_table")

    )

  )

)

shinyApp(ui, server)

-Add Sightings : Tab Layout

ui <- fluidPage(

  titlePanel("UFO Sightings"),

  sidebarPanel(

    selectInput("state", "Choose a U.S. state:", choices = unique(usa\_ufo\_sightings$state)),

    dateRangeInput("dates", "Choose a date range:",

      start = "1920-01-01",

      end = "1950-01-01"

    )

  ),

  # MODIFY CODE BELOW: Create a tab layout for the dashboard

  mainPanel(

    tabsetPanel(

      tabPanel("Number sighted", plotOutput("shapes")),

      tabPanel("Duration table", tableOutput("duration\_table"))

    )

  )

)

server <- function(input, output) {

  output$shapes <- renderPlot({

    usa\_ufo\_sightings %>%

      filter(

        state == input$state,

        date\_sighted >= input$dates[1],

        date\_sighted <= input$dates[2]

      ) %>%

      ggplot(aes(shape)) +

      geom\_bar() +

      labs(

        x = "Shape",

        y = "# Sighted"

      )

  })

  output$duration\_table <- renderTable({

    usa\_ufo\_sightings %>%

      filter(

        state == input$state,

        date\_sighted >= input$dates[1],

        date\_sighted <= input$dates[2]

      ) %>%

      group\_by(shape) %>%

      summarize(

        nb\_sighted = n(),

        avg\_duration\_min = mean(duration\_sec) / 60,

        median\_duration\_min = median(duration\_sec) / 60,

        min\_duration\_min = min(duration\_sec) / 60,

        max\_duration\_min = max(duration\_sec) / 60

      )

  })

}

shinyApp(ui, server)