R CODE: -

###Cleaning data

LIBRARY USE
```{r}
library(readxl)
library(dplyr)
library(writexl)
```
IMPORT & CHECK DATA
```{r}
#file.choose()
dt <- read_xlsx("C:\\Users\\Asus\\Documents\\R CODING\\Group Project\\Suicide
Data.xlsx")
head(dt);tail(dt)
str(dt)
names(dt)
dim(dt)
```
DATA CLEANING
```{r}
anyNA(dt)
colSums(is.na(dt))
dt_clean <- na.omit(dt)
anyNA(dt_clean)

remove_duplicate_rows <- dt_clean %>% distinct()
dt_clean_2 <- dt_clean %>% select(-`country-year`)

str(dt_clean_2)
head(dt_clean_2)
dim(dt_clean_2)

# Function to calculate mean for age ranges
mean_age <- function(age_range) {
  if (age_range == "75+ years") {
    return(87.5) #assuming the upper bound is 100 years old
  } else {
    range <- as.numeric(unlist(strsplit(gsub(" years", "", age_range), "-")))
    return(mean(range))
```

~~~~
  }
}

# Apply the function to the age column and create a new column for median age
dt_clean_2 <- dt_clean_2 %>%
  mutate(age_mean = sapply(age, mean_age)) %>%
  relocate(age_mean, .after = age)

str(dt_clean_2)
dim(dt_clean_2)
names(dt_clean_2)

write_xlsx(dt_clean_2, "Cleaned_Suicide_Data.xlsx")
print("Cleaned data has been saved to 'Cleaned_Suicide_Data.xlsx'")
```



###Start to visualize data
```{r}
##file.choose()

library(readxl)
dt <- read_excel("C:/Users/local pc/OneDrive/Cleaned_Suicide_DataFinal.xlsx")### TUKAR
IKUT NAMA FILE SENDIRI
head(dt)
tail(dt)
```
```{r}
## LOAD R PACKAGES

library(tidyverse) #load dplyr ggplot2, string
library(sf) #working with geographic simple features in r
library(RColorBrewer)
library(rnaturalearth) #world map data from natural earth
library(countrycode) #get ISO  code from country name
library(ggrepel) #ggplot2 extension for overlapping text labels
library(tmap)    # for static and interactive maps
```
```{r}
### Map
### Get world Data
library(rnaturalearth)
~~~~

```
# Load and prepare data
world <- ne_countries(scale = "small", returnclass = "sf")

data <- dt %>%
  select(suicides_no, year, country, gdp_per_capita) %>%
  separate_rows(country, sep = ", ") %>%
  mutate(suicides_case = TRUE)

data_with_iso <- data %>%
  mutate(Iso3 = countrycode::countrycode(
    sourcevar = country,
    origin = "country.name",
    destination = "iso3c")
  )

countries_suicides_case <- world %>%
  select(geometry, name, iso_a3) %>%
  left_join(data_with_iso, by = c("iso_a3" = "Iso3")) %>%
  filter(suicides_case == TRUE)

Total_gdp_per_capita <- data %>%
  filter(year >= 1985, year <= 2016) %>%
  group_by(country) %>%
  summarise(gdp_per_capita = sum(gdp_per_capita, na.rm = TRUE)) %>%
  arrange(desc(gdp_per_capita))

world_gdp <- world %>%
  left_join(Total_gdp_per_capita, by = c("admin" = "country"))

Total_suicides_no <- data %>%
  filter(year >= 1985, year <= 2016) %>%
  group_by(country) %>%
  summarise(suicides_no = sum(suicides_no, na.rm = TRUE)) %>%
  arrange(desc(suicides_no))

world_filtered <- world %>%
  filter(admin != "Antarctica")

merged_data <- merge(world_filtered, Total_suicides_no, by.x = "admin", by.y = "country", all.x =
TRUE)
merged_data <- st_transform(merged_data, crs = "+proj=robin")

# Define UI
ui <- fluidPage(
  titlePanel("Global Analysis of GDP per Capita and Suicide Rates"),
  sidebarLayout(
    sidebarPanel(
      selectInput("mapType", "Select Map to Display:",
                  choices = c("GDP per Capita" = "gdp", "Suicide Rates" = "suicide"))
    ),
    mainPanel(
      plotOutput("mapPlot")
    )
  )
)

# Define server logic
server <- function(input, output) {
  output$mapPlot <- renderPlot({
    if (input$mapType == "gdp") {
      world_gdp %>%
        filter(admin != "Antarctica") %>%
        st_transform(crs = "+proj=robin") %>%
        ggplot() +
        geom_sf(color = "black") +
        geom_sf(aes(fill = gdp_per_capita)) +
        scale_fill_viridis_c(option = "C", direction = -1) +
        theme_minimal() +
        theme(plot.title = element_text(face = "bold"),
              legend.position = "right") +
        labs(title = "Global GDP Per Capita by Country",
             subtitle = "Analyzing GDP Per Capita Trends Across Nations (1985-2016)",
             x = "", y = "",
             caption = "group2project.com")
    } else {
      ggplot() +
        geom_sf(data = merged_data, aes(fill = suicides_no), color = "black") +
        scale_fill_gradient(low = "lightblue", high = "darkblue") +
        theme_minimal() +
        theme(plot.title = element_text(face = "bold"),
              axis.text.x = element_blank()) +
        labs(title = "Total Suicides by Country",
             subtitle = "Analyzing Suicide Rates and Trends Across Nations",
             x = NULL, y = NULL,
             caption = "group2project.com")
    }
  })
}
```

```
# Run the application
shinyApp(ui = ui, server = server)
```
```{r}
### Boxplot

library(shiny)
library(readxl)
library(ggplot2)
library(dplyr)

# Load the data
data <- read_excel("C:/Users/local pc/OneDrive/Cleaned_Suicide_DataFinal.xlsx")

# Identify top 3 countries by total suicides
top_countries <- data %>%
  group_by(country) %>%
  summarise(total_suicides = sum(suicides_no, na.rm = TRUE)) %>%
  top_n(3, total_suicides) %>%
  pull(country)

# Filter data for the top 3 countries
top_countries_data <- data %>% filter(country %in% top_countries)

# Define UI
ui <- fluidPage(
  titlePanel("Boxplot of Suicides and Population for Top 3 Countries"),
  sidebarLayout(
    sidebarPanel(
      selectInput("variable", "Select Variable:",
                  choices = c("suicides_no", "population"))
    ),
    mainPanel(
      plotOutput("boxplot")
    )
  )
)

# Define server logic
server <- function(input, output) {
  output$boxplot <- renderPlot({
    ggplot(top_countries_data, aes_string(x = 'country', y = input$variable, fill = 'country')) +
      geom_boxplot() +
```

~~~
      labs(y = input$variable, title = paste("Boxplot of", input$variable, "for Top 3 Countries")) +
      theme_minimal() +
      scale_fill_brewer(palette = "Set3")
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

```{r}
##PIE CHART
# Define UI
ui <- fluidPage(
  titlePanel("Suicides by Gender in Top Country"),
  mainPanel(
    plotlyOutput("pie_chart")
  )
)

# Define server logic
server <- function(input, output) {
  # Load the dataset
  file_path <- "C:/Users/local pc/OneDrive/Cleaned_Suicide_DataFinal.xlsx"  # Replace with the
actual path to your Excel file
  data <- read_excel(file_path)

  # Find the top country with the most suicide numbers
  top_country <- aggregate(suicides_no ~ country, data = data, sum)
  top_country <- top_country[which.max(top_country$suicides_no), "country"]

  # Filter data for the top country
  top_country_data <- subset(data, country == top_country)

  # Summarize the number of suicides by sex in the top country
  suicides_by_sex <- aggregate(suicides_no ~ sex, data = top_country_data, sum)

  # Calculate percentages
  suicides_by_sex$percentage <- round((suicides_by_sex$suicides_no /
sum(suicides_by_sex$suicides_no)) * 100, 1)

  # Create labels for the pie chart
  labels <- paste(suicides_by_sex$sex, suicides_by_sex$percentage, "%")
~~~

```
  # Create tooltips
  tooltip_values <- paste("Gender: ", suicides_by_sex$sex, "<br>Percentage: ",
suicides_by_sex$percentage, "%<br>Suicides: ", suicides_by_sex$suicides_no)

  # Output the pie chart with tooltips
  output$pie_chart <- renderPlotly({
    plot_ly(labels = ~labels, values = ~suicides_by_sex$suicides_no, type = "pie",
            text = tooltip_values, hoverinfo = "text", marker = list(colors = c("skyblue", "lightcoral")))
%>%
      layout(title = paste("Percentage of Suicides by Sex in", top_country))
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```
```{r}
###BAR CHART

library(shiny)
library(readxl)
library(dplyr)
library(ggplot2)
library(plotly)

# Load the dataset
file_path <- "C:/Users/local pc/OneDrive/Cleaned_Suicide_DataFinal.xlsx" # Replace with the
actual path to your Excel file
data <- read_excel(file_path)

# Define UI
ui <- fluidPage(
  titlePanel("Suicide Data Analysis"),
  sidebarLayout(
    sidebarPanel(
      selectInput("visualization", "Choose Visualization:",
                  choices = c("Median Age of Suicides", "Suicides per 100k Population by Sex")
      )
    ),
    mainPanel(
      plotlyOutput("plot")
    )
  )
)
```

```
# Define server logic
server <- function(input, output) {
  observe({
    if (input$visualization == "Median Age of Suicides") {
      # Calculate the median age for each age group
      age_median <- aggregate(age_median ~ age, data = data, median)

      # Create the bar chart for median age
      output$plot <- renderPlotly({
        p <- ggplot(age_median, aes(x = age, y = age_median)) +
          geom_bar(stat = "identity", fill = "skyblue") +
          labs(title = paste("Median Age of Suicides"), x = "Age Group", y = "Median Age") +
          theme_minimal()

        ggplotly(p, tooltip = c("y"))
      })
    } else if (input$visualization == "Suicides per 100k Population by Sex") {
      # Find the top country with the most suicide numbers
      top_country <- aggregate(suicides_no ~ country, data = data, sum)
      top_country <- top_country[which.max(top_country$suicides_no), "country"]

      # Filter data for the top country
      top_country_data <- subset(data, country == top_country)

      # Calculate suicides per 100k population
      top_country_data$suicides_per_100k <- (top_country_data$suicides_no /
top_country_data$population) * 100000

      # Summarize suicides per 100k population by sex
      suicides_per_100k_by_sex <- aggregate(suicides_per_100k ~ sex, data =
top_country_data, sum)

      # Create the bar chart
      output$plot <- renderPlotly({
        p <- ggplot(suicides_per_100k_by_sex, aes(x = sex, y = suicides_per_100k, fill = sex)) +
          geom_bar(stat = "identity") +
          labs(title = paste("Suicides per 100k Population by Sex in", top_country),
               x = "Sex", y = "Suicides per 100k Population") +
          theme_minimal() +
          scale_fill_manual(values = c("skyblue", "lightcoral"))

        ggplotly(p, tooltip = c("y"))
      })
```

~~~
    }
  })
}

# Run the application
shinyApp(ui = ui, server = server)

```
```{r}
###TREND LINE PART

library(shiny)
library(ggplot2)
library(dplyr)
library(readxl)
library(plotly)

# Load the data
data <- read_excel("C:/Users/local pc/OneDrive/Cleaned_Suicide_DataFinal.xlsx")

# Identify the top country by total suicides
top_country <- data %>%
  group_by(country) %>%
  summarise(total_suicides = sum(suicides_no, na.rm = TRUE)) %>%
  top_n(1, total_suicides) %>%
  pull(country)

# Filter data for the top country
filtered_data <- data %>% filter(country == top_country)

# Shiny app UI
ui <- fluidPage(
  titlePanel("Trend Line of Year vs HDI for Top Country by Total Suicides"),
  sidebarLayout(
    sidebarPanel(
      selectInput("country", "Select Country", choices = unique(data$country), selected =
top_country),
      checkboxInput("show_trendline", "Show Trendline", value = TRUE)
    ),
    mainPanel(
      plotlyOutput("trendPlot"),
      textOutput("error")
    )
  )
~~~

```
)

# Shiny app server
server <- function(input, output) {
  filteredData <- reactive({
    req(input$country)
    data %>%
      filter(country == input$country)
  })

  output$trendPlot <- renderPlotly({
    plot_data <- filteredData()
    if(nrow(plot_data) == 0) {
      output$error <- renderText("No data available for the selected country.")
      return(NULL)
    }

    p <- ggplot(plot_data, aes(x = year, y = `HDI for year`)) +
      geom_point(aes(text = paste("Year:", year, "<br>HDI:", `HDI for year`))) +
      labs(title = paste("Trend Line for", input$country),
           x = "Year",
           y = "HDI for year") +
      theme_minimal()

    gg <- ggplotly(p, tooltip = "text") %>%
      config(displayModeBar = FALSE) %>%
      layout(showlegend = FALSE)

    if (input$show_trendline) {
      lm_model <- lm(`HDI for year` ~ year, data = plot_data)
      trendline <- data.frame(year = plot_data$year, HDI = predict(lm_model))

      gg <- gg %>%
        add_lines(data = trendline, x = ~year, y = ~HDI, line = list(color = 'tomato'))
    }

    gg
  })

  output$error <- renderText(NULL)
}

# Run the Shiny app
shinyApp(ui = ui, server = server)
```

```
```
```{r}
### DASHBOARD
# Define UI
ui <- dashboardPage(
  dashboardHeader(
    title = "A JOURNEY THROUGH DECADES OF GLOBAL SUICIDE RATES (1985 - 2016)",
    titleWidth = 650
  ),
  dashboardSidebar(
    sidebarMenu(
      id = "sidebar",
      menuItem("Dataset", tabName = "data", icon = icon("database")),
      menuItem("Overview", tabName = "Overview", icon = icon("globe")),
      menuItem("Visualization 1", tabName = "viz1", icon = icon("chart-bar")),
      menuItem("Visualization 2", tabName = "viz2", icon = icon("chart-pie")),
      menuItem("Visualization 3", tabName = "viz3", icon = icon("chart-line"))
    )
  ),
  dashboardBody(
    tags$head(
      tags$style(
        HTML("
          .skin-blue .main-header .logo {
            background-color: darkred;
          }
          .skin-blue .main-header .logo:hover {
            background-color: darkred;
          }
          .skin-blue .main-header .navbar {
            background-color: darkred;
          }
          .skin-blue .main-header .navbar:hover {
            background-color: darkred;
          }
        ")
      )
    ),
    tabItems(
      tabItem(tabName = "data",
        fluidRow(
          box(
            title = "Dataset",
            width = 12,
            DT::dataTableOutput("datasetTable")
          )
        )
      ),
      tabItem(tabName = "Overview",
        fluidRow(
          box(
            title = "Global Analysis of GDP per Capita and Suicide Rates",
            width = 12,
            sidebarLayout(
              sidebarPanel(
                selectInput("mapType", "Select Map to Display:",
                            choices = c("GDP per Capita" = "gdp", "Suicide Rates" = "suicide"))
              ),
              mainPanel(
                plotOutput("mapPlot")
              )
            )
          )
        )
      ),
      tabItem(tabName = "viz1",
        fluidRow(
          box(
            title = "Boxplot of Suicides and Population for Top 3 Countries",
            width = 12,
            sidebarLayout(
              sidebarPanel(
                selectInput("variable", "Select Variable:", choices = c("suicides_no", "population"))
              ),
              mainPanel(
                plotOutput("boxplot")
              )
            )
          )
        )
      ),
      tabItem(tabName = "viz2",
        fluidRow(
          box(
            title = "Suicides by Gender in Top Country",
            width = 12,
            plotlyOutput("pie_chart")
```

```
          ),
          box(
            title = "Select Visualization",
            width = 12,
            sidebarLayout(
              sidebarPanel(
                selectInput("visualization", "Choose Visualization:",
                            choices = c("Median Age of Suicides", "Suicides per 100k Population by Sex")
                )
              ),
              mainPanel(
                plotlyOutput("plot")
              )
            )
          )
        )
      ),
      tabItem(tabName = "viz3",
        fluidRow(
          box(
            title = "Trend Line of Year vs HDI for Top Country by Total Suicides",
            width = 12,
            sidebarLayout(
              sidebarPanel(
                selectInput("country", "Select Country", choices = unique(data$country), selected =
top_country),
                checkboxInput("show_trendline", "Show Trendline", value = TRUE)
              ),
              mainPanel(
                plotlyOutput("trendPlot"),
                textOutput("error")
              )
            )
          )
        )
      )
    )
  )
)

# Define the server
server <- function(input, output) {
  # Render the dataset table
  output$datasetTable <- DT::renderDataTable({
```

```
  DT::datatable(data)
})

# Boxplot for top 3 countries
output$boxplot <- renderPlot({
  ggplot(top_countries_data, aes_string(x = 'country', y = input$variable, fill = 'country')) +
    geom_boxplot() +
    labs(y = input$variable, title = paste("Boxplot of", input$variable, "for Top 3 Countries")) +
    theme_minimal() +
    scale_fill_brewer(palette = "Set3")
})

# Filter data for selected country
filteredData <- reactive({
  req(input$country)
  data %>%
    filter(country == input$country)
})

# Trend line plot
output$trendPlot <- renderPlotly({
  plot_data <- filteredData()
  if (nrow(plot_data) == 0) {
    output$error <- renderText("No data available for the selected country.")
    return(NULL)
  }

  p <- ggplot(plot_data, aes(x = year, y = `HDI for year`)) +
    geom_point(aes(text = paste("Year:", year, "<br>HDI:", `HDI for year`))) +
    labs(title = paste("Trend Line for", input$country),
         x = "Year",
         y = "HDI for year") +
    theme_minimal()

  gg <- ggplotly(p, tooltip = "text") %>%
    config(displayModeBar = FALSE) %>%
    layout(showlegend = FALSE)

  if (input$show_trendline) {
    lm_model <- lm(`HDI for year` ~ year, data = plot_data)
    trendline <- data.frame(year = plot_data$year, HDI = predict(lm_model))

    gg <- gg %>%
      add_lines(data = trendline, x = ~year, y = ~HDI, line = list(color = 'tomato'))
    }

    gg
  })

  output$error <- renderText(NULL)

  # Pie chart for suicides by gender in the top country
  output$pie_chart <- renderPlotly({
    top_country_data <- data %>% filter(country == top_country)
    suicides_by_sex <- top_country_data %>%
      group_by(sex) %>%
      summarise(suicides_no = sum(suicides_no, na.rm = TRUE))
    suicides_by_sex$percentage <- round((suicides_by_sex$suicides_no /
sum(suicides_by_sex$suicides_no)) * 100, 1)
    labels <- paste(suicides_by_sex$sex, suicides_by_sex$percentage, "%")
    tooltip_values <- paste("Gender: ", suicides_by_sex$sex, "<br>Percentage: ",
suicides_by_sex$percentage, "%<br>Suicides: ", suicides_by_sex$suicides_no)

    plot_ly(labels = ~labels, values = ~suicides_by_sex$suicides_no, type = "pie",
            text = tooltip_values, hoverinfo = "text", marker = list(colors = c("skyblue", "lightcoral")))
%>%
      layout(title = paste("Percentage of Suicides by Sex in", top_country))
  })

  # Visualization for "Median Age of Suicides" and "Suicides per 100k Population by Sex"
  observe({
    if (input$visualization == "Median Age of Suicides") {
      # Calculate the median age for each age group
      age_median <- aggregate(age_median ~ age, data = data, median)

      # Create the bar chart for median age
      output$plot <- renderPlotly({
        p <- ggplot(age_median, aes(x = age, y = age_median)) +
          geom_bar(stat = "identity", fill = "skyblue") +
          labs(title = "Median Age of Suicides", x = "Age Group", y = "Median Age") +
          theme_minimal()

        ggplotly(p, tooltip = c("y"))
      })
    } else if (input$visualization == "Suicides per 100k Population by Sex") {
      # Find the top country with the most suicide numbers
      top_country <- aggregate(suicides_no ~ country, data = data, sum)
      top_country <- top_country[which.max(top_country$suicides_no), "country"]
      # Filter data for the top country
      top_country_data <- subset(data, country == top_country)

      # Calculate suicides per 100k population
      top_country_data$suicides_per_100k <- (top_country_data$suicides_no /
top_country_data$population) * 100000

      # Summarize suicides per 100k population by sex
      suicides_per_100k_by_sex <- aggregate(suicides_per_100k ~ sex, data =
top_country_data, sum)

      # Create the bar chart
      output$plot <- renderPlotly({
        p <- ggplot(suicides_per_100k_by_sex, aes(x = sex, y = suicides_per_100k, fill = sex)) +
          geom_bar(stat = "identity") +
          labs(title = paste("Suicides per 100k Population by Sex in", top_country),
               x = "Sex", y = "Suicides per 100k Population") +
          theme_minimal() +
          scale_fill_manual(values = c("skyblue", "lightcoral"))

        ggplotly(p, tooltip = c("y"))
      })
    }
  })

  # Map plot
  output$mapPlot <- renderPlot({
    if (input$mapType == "gdp") {
      world_gdp %>%
        filter(admin != "Antarctica") %>%
        st_transform(crs = "+proj=robin") %>%
        ggplot() +
        geom_sf(color = "black") +
        geom_sf(aes(fill = gdp_per_capita)) +
        scale_fill_viridis_c(option = "C", direction = -1) +
        theme_minimal() +
        theme(plot.title = element_text(face = "bold"),
              legend.position = "right") +
        labs(title = "Global GDP Per Capita by Country",
             subtitle = "Analyzing GDP Per Capita Trends Across Nations (1985-2016)",
             x = "", y = "",
             caption = "group2project.com")
    } else {
```

~~~
      ggplot() +
        geom_sf(data = merged_data, aes(fill = suicides_no), color = "black") +
        scale_fill_gradient(low = "lightblue", high = "darkblue") +
        theme_minimal() +
        theme(plot.title = element_text(face = "bold"),
              axis.text.x = element_blank()) +
        labs(title = "Total Suicides by Country",
             subtitle = "Analyzing Suicide Rates and Trends Across Nations",
             x = NULL, y = NULL,
             caption = "group2project.com")
    }
  })
}
# Run the Shiny app
shinyApp(ui = ui, server = server)
```
~~~