Step-by-Step Report

Step 1:

```
if(!require(shiny)) install.packages("shiny")
if(!require(dplyr)) install.packages("dplyr")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(readr)) install.packages("readr")
if(!require(scales)) install.packages("scales")

library(shiny)
library(dplyr)
library(ggplot2)
library(readr)
library(scales)
```

These steps ensure that all required R packages are installed and loaded before running the application. Shiny for building an interactive web application, Dplyr for data filtering, grouping and summarizing, Ggplot2 for data visualization, Readr for CSV files reading, and Scales to improve readability of numeric datas.

Step 2:

```
dengue_final <- read_csv("data/cleaned_dengue.csv")
```

The cleaned dataset is imported from a CSV file stored in the project directory. This dataset contains dengue case counts by year, month, and region. Loading the data once at the beginning allows it to bre reused throughout the application.

Step 3:

```
colnames_lower <- tolower(colnames(dengue_final))
```

The column names are converted to lowercase to avoid inconsistencies due to capitalization differences, which can cause errors in data processing.

```
year_col <- colnames(dengue_final)[grepl("year", colnames_lower)]
month_col <- colnames(dengue_final)[grepl("month", colnames_lower)]
region_col <- colnames(dengue_final)[grepl("region|province", colnames_lower)]
cases_col <- colnames(dengue_final)[grepl("case|total|confirmed", colnames_lower)]
```

This step programmatically identifies the relevant columns even if the dataset uses slightly different naming conventions. It increases robustness and ensures the app can adapt to minor variations.

```
dengue_final <- dengue_final %>%
  rename(
    year = all_of(year_col),
    month = all_of(month_col),
    region = all_of(region_col),
    cases = all_of(cases_col)
  )
```

Columns are categorized into year, month, region, and cases, so they can be referenced consistently throughout the app.

```
mutate(
  region = trimws(region),
  year = as.numeric(year),
  cases = as.numeric(cases),
  month = case_when(
    ...
  )
)
```

Region names are trimmed to remove extra spaces, year and case counts are converted to numeric values to allow aggregation, and month values are converted from text to numeric values, enabling proper time-based ordering in plots.

```
filter(!is.na(year) & !is.na(month) & !is.na(cases) & !is.na(region))
```

This removes incomplete records, ensuring that only valid observations are used in the analysis and visualization.

Step 4:

```
ui <- fluidPage(
  titlePanel("Monthly Dengue Cases Stacked by Region (Philippines)"),
```

The UI defines what users see when they open the app. The title panel clearly communicates the purpose of the visualization.

```
sidebarLayout(
  sidebarPanel(
    selectInput("year", "Select Year:", ...),
    selectInput("region_filter", "Select Region(s):", ..., multiple = TRUE)
  ),
```

The two interactive filters are provided: the Year filter allows users to focus on a specific year and Region filter allows selection of one or multiple regions for comparison.

```
mainPanel(
  plotOutput("stackedBarPlot")
)
```

This main panel displays the stacked bar chart output.

Step 5:

```
server <- function(input, output) {
filtered_data <- reactive({
```

```
  dengue_final %>%
    filter(year == input$year,
           region %in% input$region_filter)
})
```

The server function controls how the app reacts to user input. The reactive expression dynamically filters the dataset based on the selected year and regions.

```
output$stackedBarPlot <- renderPlot({
  data <- filtered_data()
  req(nrow(data) > 0)
data_monthly <- data %>%
  group_by(month, region) %>%
  summarise(cases = sum(cases, na.rm = TRUE))
```

Monthly dengue cases are aggregated by region to prepare the data for the stacked bar chart.

Step 6:

```
ggplot(data_monthly, aes(x = factor(month, levels=1:12),
                         y = cases, fill = region)) +
  geom_bar(stat = "identity")
```

A stacked bar chart is created where the x-axis represents months, the y-axis represents the number of dengue cases, and each bar is segmented by region. The visualization allows simultaneous comparison of seasonal trends and regional contributions.

```
scale_x_discrete(labels = month.abb)
scale_y_continuous(labels = scales::comma)
```

These formatting steps enhance readability by labeling months clearly and formatting large numbers.

Step 7:

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

This command launches the Shiny app, linking the user interface and server logic. Users can now interactively explore dengue case trends by year and region.