```r
# Load required libraries
library(shiny)
library(shinydashboard)
```

```
##
## Attaching package: 'shinydashboard'
```

```
## The following object is masked from 'package:graphics':
##
##     box
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
library(DT)
```

```
##
## Attaching package: 'DT'
```

```
## The following objects are masked from 'package:shiny':
##
##     dataTableOutput, renderDataTable
```

```r
library(readr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(reshape2)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
# Read data
sales_data <- read_csv("Sales_Data_No_Outliers.csv")
```

```
## Rows: 1500 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (6): Order ID, Category, Sub-Category, CustomerName, State, City
## dbl  (3): Amount, Profit, Quantity
## date (1): Order Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
sales_targets <- read_csv("Sales target.csv")
```

```
## Rows: 36 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Month of Order Date, Category
## dbl (1): Target
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Pre-process data
# Convert date
sales_data$`Order Date` <- as.Date(sales_data$`Order Date`)
sales_data$Month <- month(sales_data$`Order Date`)
sales_data$MonthName <- month(sales_data$`Order Date`, label = TRUE)
sales_data$Year <- year(sales_data$`Order Date`)
```

```r
# Pre-calculate some summaries for faster loading
state_summary <- sales_data %>%
  group_by(State) %>%
  summarise(
    TotalSales = sum(Amount),
    TotalProfit = sum(Profit),
    OrderCount = n_distinct(`Order ID`),
    ProfitMargin = (TotalProfit / TotalSales) * 100
  ) %>%
  arrange(desc(TotalSales))

category_summary <- sales_data %>%
  group_by(Category) %>%
  summarise(
    TotalSales = sum(Amount),
    TotalProfit = sum(Profit),
```

```
    OrderCount = n_distinct(`Order ID`),
    ProfitMargin = (TotalProfit / TotalSales) * 100
  ) %>%
  arrange(desc(TotalSales))
```

```
# Create Month mapping for targets
month_mapping <- data.frame(
  MonthName = month.name,
  Month = 1:12,
  stringsAsFactors = FALSE
)
```

```
# Prepare target data
sales_targets <- sales_targets %>%
  mutate(Month = match(`Month of Order Date`, month.name))
```

```
# Install required packages if needed
# install.packages(c("shiny", "shinydashboard", "dplyr", "ggplot2", "plotly", "DT", "readr", "lubridate

# Load libraries
library(shiny)
library(shinydashboard)
library(dplyr)
library(ggplot2)
library(plotly)
library(DT)
library(readr)
library(lubridate)
library(scales)

# UI Definition
ui <- dashboardPage(
  dashboardHeader(title = "Sales Dashboard"),

  dashboardSidebar(
    sidebarMenu(
      menuItem("Overview", tabName = "overview", icon = icon("dashboard")),
      menuItem("Regional", tabName = "regional", icon = icon("map-marker")),
      menuItem("Categories", tabName = "categories", icon = icon("tags")),
      menuItem("Targets", tabName = "targets", icon = icon("bullseye"))
    ),

    # Filters
    br(),
    h4("Filters", style = "padding-left: 15px;"),

    selectInput("stateFilter", "State:", c("All" = "All")),
    selectInput("categoryFilter", "Category:", c("All" = "All")),
    dateRangeInput("dateRange", "Date Range:",
                   start = "2018-01-04",
                   end = "2019-12-03")
  ),

  dashboardBody(
```

```
    # Custom CSS for better appearance and responsiveness
    tags$head(
      tags$style(HTML("
        /* Responsive styles */
        .content-wrapper, .right-side {
          overflow-x: hidden;
        }
        .box {
          overflow: visible !important;
        }
        /* Make plots responsive */
        .plotly {
          width: 100% !important;
          height: 100% !important;
        }
        /* Mobile optimization */
        @media (max-width: 768px) {
          .box {
            margin-bottom: 10px;
          }
        }
        /* Fix value boxes */
        .info-box {min-height: 100px;}
        .info-box-icon {height: 100px; line-height: 100px;}
        .info-box-content {padding-top: 10px; padding-bottom: 10px;}
        .box-title {font-size: 18px; font-weight: 500;}
      "))
    ),

    tabItems(
      # Overview Tab
      tabItem(tabName = "overview",
              fluidRow(
                valueBoxOutput("totalSalesBox", width = 4),
                valueBoxOutput("totalProfitBox", width = 4),
                valueBoxOutput("profitMarginBox", width = 4)
              ),

              fluidRow(
                box(plotlyOutput("salesTrendPlot", height = "300px"), width = 8, title = "Sales Trend",
                box(plotlyOutput("categoryPieChart", height = "300px"), width = 4, title = "Category Br
              )
      ),

      # Regional Tab
      tabItem(tabName = "regional",
              fluidRow(
                box(plotlyOutput("topStatesPlot", height = "350px"), width = 6, title = "Top States", s
                box(plotlyOutput("stateMarginPlot", height = "350px"), width = 6, title = "State Profit
              ),
              fluidRow(
                box(plotlyOutput("topCitiesPlot", height = "350px"), width = 12, title = "Top Cities",
              )
```

```
      ),

      # Categories Tab
      tabItem(tabName = "categories",
              fluidRow(
                box(plotlyOutput("categoryPerformancePlot", height = "350px"), width = 12, title = "Cate
              ),
              fluidRow(
                box(plotlyOutput("subCategoryPlot", height = "350px"), width = 12, title = "Sub-Category
              )
      ),

      # Targets Tab
      tabItem(tabName = "targets",
              fluidRow(
                box(plotlyOutput("achievementGauge", height = "350px"), width = 6, title = "Overall Achi
                box(plotlyOutput("achievementByCategoryPlot", height = "350px"), width = 6, title = "Ach
              ),
              fluidRow(
                box(plotlyOutput("monthlyAchievementPlot", height = "350px"), width = 12, title = "Month
              )
      )
    )
  )
)

# Server logic
server <- function(input, output, session) {
  # Load data
  sales_data <- reactive({
    # Read sales data
    df <- read_csv("Sales_Data_No_Outliers.csv")
    df$`Order Date` <- as.Date(df$`Order Date`)
    df$Month <- month(df$`Order Date`)
    df$MonthName <- month(df$`Order Date`, label = TRUE)
    df$Year <- year(df$`Order Date`)
    return(df)
  })

  sales_targets <- reactive({
    # Read target data
    df <- read_csv("Sales target.csv")
    df$Month <- match(df$`Month of Order Date`, month.name)
    return(df)
  })

  # Update filters
  observe({
    req(sales_data())
    updateSelectInput(session, "stateFilter",
                      choices = c("All" = "All", unique(sales_data()$State)))

    updateSelectInput(session, "categoryFilter",
```

```
                    choices = c("All" = "All", unique(sales_data()$Category)))
})

# Filtered data
filtered_data <- reactive({
  data <- sales_data()

  # Apply filters
  if(input$stateFilter != "All") {
    data <- data %>% filter(State == input$stateFilter)
  }

  if(input$categoryFilter != "All") {
    data <- data %>% filter(Category == input$categoryFilter)
  }

  data <- data %>% filter(`Order Date` >= input$dateRange[1] &
                            `Order Date` <= input$dateRange[2])

  return(data)
})

# Calculate target achievement
target_achievement <- reactive({
  # Get filtered data
  filtered <- filtered_data()
  filtered_months <- unique(month(filtered$`Order Date`))

  # Get relevant targets
  relevant_targets <- sales_targets() %>%
    filter(Month %in% filtered_months)

  if(input$categoryFilter != "All") {
    relevant_targets <- relevant_targets %>%
      filter(Category == input$categoryFilter)
  }

  # Calculate sum of targets and actuals
  total_target <- sum(relevant_targets$Target)
  total_actual <- sum(filtered$Amount)

  # Calculate achievement
  achievement <- if(total_target > 0) (total_actual / total_target) * 100 else 0

  return(list(
    target = total_target,
    actual = total_actual,
    achievement = achievement
  ))
})

# KPI Boxes
output$totalSalesBox <- renderValueBox({
```

```
  sales <- sum(filtered_data()$Amount)
  valueBox(
    formatC(sales, format="f", big.mark=",", digits=0),
    "Total Sales ($)",
    icon = icon("dollar-sign"),
    color = "blue"
  )
})

output$totalProfitBox <- renderValueBox({
  profit <- sum(filtered_data()$Profit)
  valueBox(
    formatC(profit, format="f", big.mark=",", digits=0),
    "Total Profit ($)",
    icon = icon("chart-line"),
    color = "green"
  )
})

output$profitMarginBox <- renderValueBox({
  margin <- (sum(filtered_data()$Profit) / sum(filtered_data()$Amount)) * 100
  valueBox(
    paste0(round(margin, 2), "%"),
    "Profit Margin",
    icon = icon("percentage"),
    color = "purple"
  )
})

# Sales Trend Plot
output$salesTrendPlot <- renderPlotly({
  # Group by month and sum
  monthly_sales <- filtered_data() %>%
    group_by(Month, MonthName) %>%
    summarise(Sales = sum(Amount), .groups = "drop") %>%
    arrange(Month)

  # Add all 12 months if some are missing
  all_months <- data.frame(
    Month = 1:12,
    MonthName = month(1:12, label = TRUE)
  )

  monthly_sales <- monthly_sales %>%
    right_join(all_months, by = c("Month", "MonthName")) %>%
    mutate(Sales = ifelse(is.na(Sales), 0, Sales))

  p <- ggplot(monthly_sales, aes(x = MonthName, y = Sales, group = 1)) +
    geom_line(color = "#1976D2", size = 1.2) +
    geom_point(color = "#1976D2", size = 3) +
    theme_minimal() +
    labs(x = "Month", y = "Sales ($)") +
    scale_y_continuous(labels = dollar_format()) +
```

```
    theme(axis.text.x = element_text(angle = 45, hjust = 1))

  ggplotly(p) %>%
    layout(autosize = TRUE,
           margin = list(l = 50, r = 50, b = 100, t = 50, pad = 4))
})

# Category Pie Chart
output$categoryPieChart <- renderPlotly({
  cat_data <- filtered_data() %>%
    group_by(Category) %>%
    summarise(Sales = sum(Amount), .groups = "drop")

  # Calculate percentages
  cat_data <- cat_data %>%
    mutate(Percentage = Sales / sum(Sales) * 100,
           Label = paste0(Category, " (", round(Percentage, 1), "%)"))

  # Standard colors
  colors <- c("#4CAF50", "#2196F3", "#FF9800")

  plot_ly(cat_data, labels = ~Category, values = ~Sales, type = 'pie',
          textinfo = 'label+percent',
          insidetextorientation = 'radial',
          marker = list(colors = colors,
                        line = list(color = '#FFFFFF', width = 1)),
          textposition = 'inside',
          hoverinfo = 'text',
          text = ~paste0(Category, ": $", format(Sales, big.mark = ","), " (", round(Percentage, 1),
    layout(showlegend = TRUE,
           autosize = TRUE,
           margin = list(l = 20, r = 20, b = 20, t = 30, pad = 0),
           legend = list(orientation = "v", x = 1, y = 0.5))
})

# Top States Plot
output$topStatesPlot <- renderPlotly({
  state_data <- filtered_data() %>%
    group_by(State) %>%
    summarise(Sales = sum(Amount), .groups = "drop") %>%
    arrange(desc(Sales)) %>%
    head(10)

  p <- ggplot(state_data, aes(x = reorder(State, Sales), y = Sales)) +
    geom_bar(stat = "identity", fill = "#1976D2") +
    coord_flip() +
    theme_minimal() +
    labs(x = "", y = "Sales ($)") +
    scale_y_continuous(labels = dollar_format())

  ggplotly(p) %>%
    layout(autosize = TRUE,
           margin = list(l = 100, r = 20, b = 50, t = 30, pad = 4))
```

```
})

# State Profit Margins Plot
output$stateMarginPlot <- renderPlotly({
  state_margin <- filtered_data() %>%
    group_by(State) %>%
    summarise(
      TotalSales = sum(Amount),
      TotalProfit = sum(Profit),
      ProfitMargin = (TotalProfit / TotalSales) * 100,
      .groups = "drop"
    ) %>%
    arrange(desc(ProfitMargin)) %>%
    head(10)

  p <- ggplot(state_margin, aes(x = reorder(State, ProfitMargin), y = ProfitMargin)) +
    geom_bar(stat = "identity", fill = "#4CAF50") +
    coord_flip() +
    theme_minimal() +
    labs(x = "", y = "Profit Margin (%)")

  ggplotly(p) %>%
    layout(autosize = TRUE,
           margin = list(l = 100, r = 20, b = 50, t = 30, pad = 4))
})

# Top Cities Plot
output$topCitiesPlot <- renderPlotly({
  # Get filtered data for cities
  city_data <- filtered_data()

  # Apply additional state filter if no specific state is selected
  if (input$stateFilter == "All") {
    # Get top state
    top_state <- city_data %>%
      group_by(State) %>%
      summarise(TotalSales = sum(Amount), .groups = "drop") %>%
      arrange(desc(TotalSales)) %>%
      slice(1) %>%
      pull(State)

    city_data <- city_data %>% filter(State == top_state)
    city_title <- paste("Top Cities in", top_state)
  } else {
    city_title <- paste("Top Cities in", input$stateFilter)
  }

  # Get top cities
  top_cities <- city_data %>%
    group_by(City) %>%
    summarise(Sales = sum(Amount), .groups = "drop") %>%
    arrange(desc(Sales)) %>%
    head(10)
```

```
  # Handle case with no cities
  if(nrow(top_cities) == 0) {
    return(plotly_empty(type = "scatter", mode = "markers") %>%
             layout(title = "No city data available for the selected filters"))
  }

  p <- ggplot(top_cities, aes(x = reorder(City, Sales), y = Sales)) +
    geom_bar(stat = "identity", fill = "#9C27B0") +
    coord_flip() +
    theme_minimal() +
    labs(title = city_title, x = "", y = "Sales ($)") +
    scale_y_continuous(labels = dollar_format())

  ggplotly(p) %>%
    layout(title = list(text = city_title),
           autosize = TRUE,
           margin = list(l = 100, r = 20, b = 50, t = 50, pad = 4))
})

# Category Performance Plot
output$categoryPerformancePlot <- renderPlotly({
  cat_perf <- filtered_data() %>%
    group_by(Category) %>%
    summarise(
      Sales = sum(Amount),
      Profit = sum(Profit),
      Margin = (Profit / Sales) * 100,
      .groups = "drop"
    )

  # Create a plot with two y-axes
  plot_ly() %>%
    add_trace(
      data = cat_perf,
      x = ~Category,
      y = ~Sales,
      type = 'bar',
      name = 'Sales',
      marker = list(color = '#1976D2')
    ) %>%
    add_trace(
      data = cat_perf,
      x = ~Category,
      y = ~Profit,
      type = 'scatter',
      mode = 'lines+markers',
      name = 'Profit',
      yaxis = 'y2',
      line = list(color = '#4CAF50', width = 3),
      marker = list(color = '#4CAF50', size = 10)
    ) %>%
    layout(
      title = "Category Performance: Sales vs Profit",
```

```
      autosize = TRUE,
      margin = list(l = 60, r = 60, b = 60, t = 60, pad = 4),
      xaxis = list(title = ""),
      yaxis = list(
        title = "Sales Amount ($)",
        tickformat = "$,.0f",
        side = 'left'
      ),
      yaxis2 = list(
        title = "Profit Amount ($)",
        tickformat = "$,.0f",
        overlaying = "y",
        side = "right"
      ),
      legend = list(orientation = 'h')
    )
})

# Sub-Category Analysis Plot
output$subCategoryPlot <- renderPlotly({
  # Determine which category to analyze
  if(input$categoryFilter != "All") {
    # If specific category is selected, use it
    selected_category <- input$categoryFilter
  } else {
    # Otherwise, use the top category by sales
    selected_category <- filtered_data() %>%
      group_by(Category) %>%
      summarise(Sales = sum(Amount), .groups = "drop") %>%
      arrange(desc(Sales)) %>%
      slice(1) %>%
      pull(Category)
  }

  # Filter for selected category
  subcat_data <- filtered_data() %>%
    filter(Category == selected_category) %>%
    group_by(`Sub-Category`) %>%
    summarise(
      Sales = sum(Amount),
      Profit = sum(Profit),
      Margin = (Profit / Sales) * 100,
      .groups = "drop"
    ) %>%
    arrange(desc(Sales))

  # Handle case with no subcategories
  if(nrow(subcat_data) == 0) {
    return(plotly_empty(type = "scatter", mode = "markers") %>%
             layout(title = "No subcategory data available for the selected filters"))
  }

  p <- ggplot(subcat_data, aes(x = reorder(`Sub-Category`, Sales), y = Sales)) +
```

```
    geom_bar(stat = "identity", fill = "#FF9800") +
    coord_flip() +
    theme_minimal() +
    labs(title = paste("Sub-Categories in", selected_category), x = "", y = "Sales ($)") +
    scale_y_continuous(labels = dollar_format())

  ggplotly(p) %>%
    layout(title = paste("Sub-Categories in", selected_category),
           autosize = TRUE,
           margin = list(l = 150, r = 20, b = 50, t = 50, pad = 4))
})

# Achievement Gauge
output$achievementGauge <- renderPlotly({
  achievement <- target_achievement()$achievement
  achievement <- min(200, max(0, achievement))  # Ensure it's between 0-200

  # Create gauge chart with plotly
  plot_ly(
    type = "indicator",
    mode = "gauge+number",
    value = achievement,
    title = list(text = "Target Achievement"),
    gauge = list(
      axis = list(range = list(0, 200), tickwidth = 1),
      bar = list(color = "darkblue"),
      steps = list(
        list(range = c(0, 70), color = "firebrick"),
        list(range = c(70, 90), color = "gold"),
        list(range = c(90, 110), color = "forestgreen"),
        list(range = c(110, 200), color = "royalblue")
      ),
      threshold = list(
        line = list(color = "red", width = 4),
        thickness = 0.75,
        value = 100
      )
    ),
    number = list(
      font = list(size = 24)
    )
  ) %>%
  layout(
    autosize = TRUE,
    margin = list(l = 20, r = 20, b = 20, t = 50, pad = 4)
  )
})

# Achievement by Category Plot
output$achievementByCategoryPlot <- renderPlotly({
  # Get filtered data
  filtered <- filtered_data()
  filtered_months <- unique(month(filtered$`Order Date`))
```

```
  # Get actual sales by category
  actual_by_category <- filtered %>%
    group_by(Category) %>%
    summarise(ActualSales = sum(Amount), .groups = "drop")

  # Get targets by category
  targets_by_category <- sales_targets() %>%
    filter(Month %in% filtered_months) %>%
    group_by(Category) %>%
    summarise(TargetSales = sum(Target), .groups = "drop")

  # Join and calculate achievement
  achievement_data <- left_join(actual_by_category, targets_by_category, by = "Category") %>%
    mutate(
      TargetSales = ifelse(is.na(TargetSales), 0, TargetSales),
      Achievement = ifelse(TargetSales > 0, (ActualSales / TargetSales) * 100, 0),
      Achievement = pmin(Achievement, 200)  # Cap at 200% for readability
    ) %>%
    arrange(desc(Achievement))

  # Handle case with no data
  if(nrow(achievement_data) == 0) {
    return(plotly_empty(type = "scatter", mode = "markers") %>%
             layout(title = "No achievement data available for the selected filters"))
  }

  # Create color scale manually
  achievement_data$Color <- sapply(achievement_data$Achievement, function(x) {
    if(x < 70) return("firebrick")
    else if(x < 90) return("gold")
    else if(x < 110) return("forestgreen")
    else return("royalblue")
  })

  p <- ggplot(achievement_data, aes(x = reorder(Category, Achievement), y = Achievement, fill = Color
    geom_bar(stat = "identity") +
    geom_hline(yintercept = 100, linetype = "dashed", color = "red") +
    coord_flip() +
    scale_fill_identity() +
    theme_minimal() +
    labs(x = "", y = "Achievement (%)")

  ggplotly(p) %>%
    layout(autosize = TRUE,
           margin = list(l = 110, r = 20, b = 50, t = 30, pad = 4))
})

# Monthly Achievement Plot
output$monthlyAchievementPlot <- renderPlotly({
  # Get filtered data
  filtered <- filtered_data()

  # Prepare monthly actual data
```

```
  monthly_actual <- filtered %>%
    group_by(Month, MonthName) %>%
    summarise(
      ActualSales = sum(Amount),
      .groups = "drop"
    )

  # Prepare monthly target data
  monthly_target <- sales_targets() %>%
    filter(Month %in% unique(filtered$Month))

  if (input$categoryFilter != "All") {
    monthly_target <- monthly_target %>%
      filter(Category == input$categoryFilter)
  }

  monthly_target <- monthly_target %>%
    group_by(Month) %>%
    summarise(
      TargetSales = sum(Target),
      .groups = "drop"
    )

  # Join data
  monthly_achievement <- full_join(monthly_actual, monthly_target, by = "Month") %>%
    mutate(
      ActualSales = ifelse(is.na(ActualSales), 0, ActualSales),
      TargetSales = ifelse(is.na(TargetSales), 0, TargetSales),
      Achievement = ifelse(TargetSales > 0, (ActualSales / TargetSales) * 100, 0)
    ) %>%
    arrange(Month)

  # Add a column for status coloring
  monthly_achievement$Status <- ifelse(monthly_achievement$Achievement >= 100, "Met", "Not Met")

  # Create plot
  plot_ly(monthly_achievement) %>%
    add_trace(
      x = ~MonthName,
      y = ~Achievement,
      type = 'scatter',
      mode = 'lines+markers',
      name = 'Achievement',
      line = list(color = '#1976D2', width = 3),
      marker = list(color = ~ifelse(Status == "Met", "#4CAF50", "#F44336"),
                    size = 10,
                    line = list(color = 'black', width = 1))
    ) %>%
    add_trace(
      x = ~MonthName,
      y = rep(100, nrow(monthly_achievement)),
      type = 'scatter',
      mode = 'lines',
```

```
        name = 'Target (100%)',
        line = list(color = 'red', dash = 'dash', width = 2)
      ) %>%
      layout(
        title = "Monthly Target Achievement",
        autosize = TRUE,
        margin = list(l = 50, r = 20, b = 80, t = 50, pad = 4),
        xaxis = list(title = "Month",
                     categoryorder = "array",
                     categoryarray = month.abb,
                     tickangle = -45),
        yaxis = list(title = "Achievement (%)",
                     range = c(0, max(200, max(monthly_achievement$Achievement, na.rm = TRUE) * 1.1))),
        legend = list(orientation = 'h', y = -0.2),
        hovermode = "closest"
      )
  })
}

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