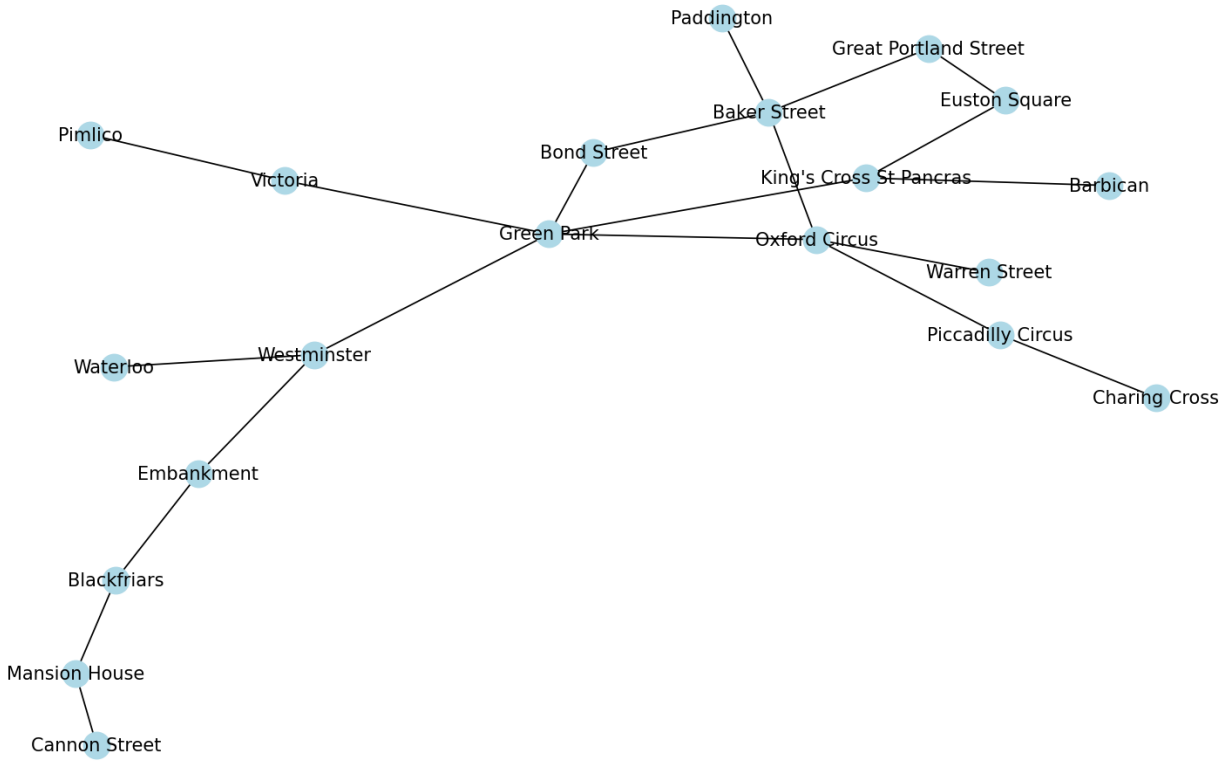
Part 1: Solution to Python task

Question 1:

```
import networkx as nx

import matplotlib.pyplot as plt


# Create a graph

G = nx.Graph()


# Outlines and stations for updated lines

lines = {

   "Victoria": ["Warren Street", "Oxford Circus", "Green Park", "Victoria", "Pimlico"],

   "Jubilee": ["Baker Street", "Bond Street", "Green Park", "Westminster", "Waterloo"],

   "Bakerloo": ["Paddington", "Baker Street", "Oxford Circus", "Piccadilly Circus", "Charing Cross"],

   "District": ["Westminster", "Embankment", "Blackfriars", "Mansion House", "Cannon Street"],

   "Metropolitan": ["Baker Street", "Great Portland Street", "Euston Square", "King's Cross St Pancras",
"Barbican"],

}


# For every line, add edges and nodes

for line, stations in lines.items():

   for i in range(len(stations) - 1):

      G.add_edge(stations[i], stations[i + 1], line=line)


# Identify transfer stations (nodes serve multiple lines)

transfer_stations = {station for station in G.nodes if sum(station in line_stations for line_stations in
lines.values()) > 1}


# Adding extra transfer links for graph connectivity

additional_transfers = [

   ("King's Cross St Pancras", "Green Park"),

   ("Baker Street", "Paddington"),
```

```
]
for station1, station2 in additional_transfers:
    G.add_edge(station1, station2, line="Transfer")

# Check if graph is connected
is_connected = nx.is_connected(G)
print("Graph is connected:", is_connected)

plt.figure(figsize=(12, 12))
pos = nx.spring_layout(G, seed=42)  # Set seed for consistent layout
nx.draw(G, pos, with_labels=True, node_size=300, node_color="lightblue")
plt.show()
```

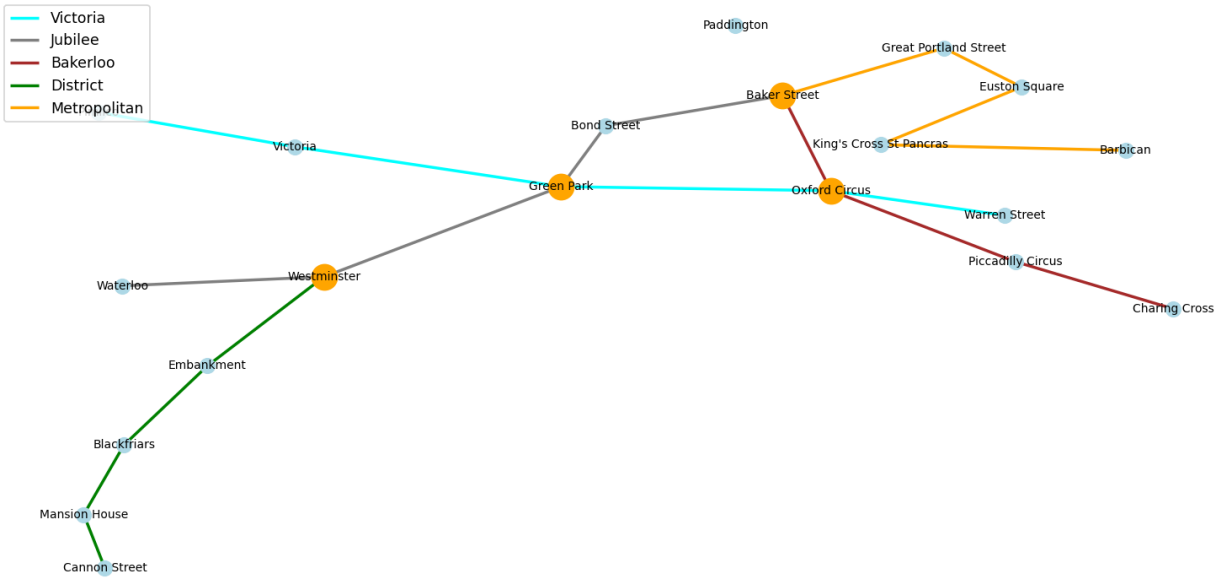
Plot after question 1:



Question 2:

```
# Color map for new lines
color_map = {
   "Victoria": "cyan",
   "Jubilee": "gray",
   "Bakerloo": "brown",
   "District": "green",
   "Metropolitan": "orange",
}


# Draw network
plt.figure(figsize=(12, 12))
pos = nx.spring_layout(G, seed=42)  # Set seed for consistent layout
# Draw each line with colors
for line, color in color_map.items():
   edges = [(u, v) for u, v, d in G.edges(data=True) if d["line"] == line]
   nx.draw_networkx_edges(G, pos, edgelist=edges, edge_color=color, width=2, label=line)


# With a larger size and different color, draw transfer stations.
nx.draw_networkx_nodes(G, pos, nodelist=transfer_stations, node_color="orange", node_size=300,
label="Transfer Stations")


# Draw other stations
nx.draw_networkx_nodes(G, pos, nodelist=set(G.nodes) - transfer_stations, node_color="lightblue",
node_size=100)


# Draw labels
nx.draw_networkx_labels(G, pos, font_size=8)
plt.legend(color_map.keys(), loc='upper left', fontsize=10)
plt.show()
```

Plot after question 2:



Question 3:

Pair some with edges weights (distances in meters)

distances = {

("Warren Street", "Oxford Circus"): 900,

("Oxford Circus", "Green Park"): 700,

("Green Park", "Victoria"): 1000,

("Victoria", "Pimlico"): 900,

("Baker Street", "Bond Street"): 1200,

("Bond Street", "Green Park"): 800,

("Green Park", "Westminster"): 1200,

("Westminster", "Waterloo"): 800,

("Paddington", "Baker Street"): 1600,

("Baker Street", "Oxford Circus"): 1500,

("Oxford Circus", "Piccadilly Circus"): 500,

("Piccadilly Circus", "Charing Cross"): 400,

("Westminster", "Embankment"): 400,

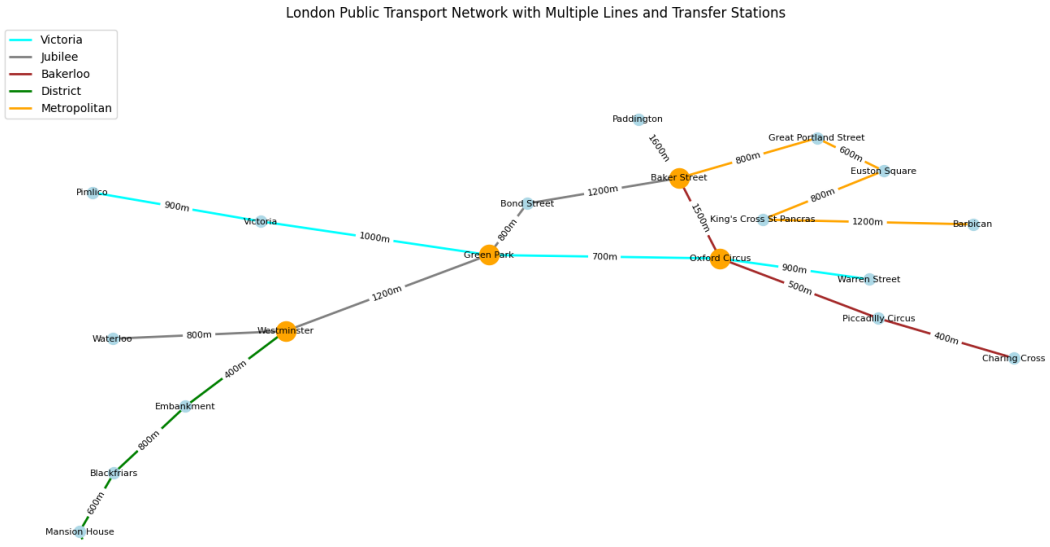
("Embankment", "Blackfriars"): 800,

("Blackfriars", "Mansion House"): 600,

("Mansion House", "Cannon Street"): 300,

```
    ("Baker Street", "Great Portland Street"): 800,
    ("Great Portland Street", "Euston Square"): 600,
    ("Euston Square", "King's Cross St Pancras"): 800,
    ("King's Cross St Pancras", "Barbican"): 1200,
}
# Update graph with distances
for (u, v), distance in distances.items():
    if G.has_edge(u, v):
        G[u][v]['distance'] = distance
# Connect the dots with edges and labels (distances)
edge_labels = {(u, v): f"{d['distance']}m" for u, v, d in G.edges(data=True) if 'distance' in d}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8)


# Add legend and title
plt.legend(color_map.keys(), loc='upper left', fontsize=10)
plt.title("London Public Transport Network with Multiple Lines and Transfer Stations")
plt.axis("off")
plt.show()
```

Plot after question 3:



Question 4:

Improvements:

1. **Color Coding and Legend Enhancement**: This means that every line must have a different and strong color and the legend should be displayed as clearly as possible with understandable labels. This will make it easy for commuters to distinguish between the lines and what routes they take.
2. **Station Name Clarity:** Enhancing the font-size of any Station names, and making sure a Station name is not colliding with another graphic elements on the map. It improves the readability and makes it easier for users to find out which station they are at immediately.

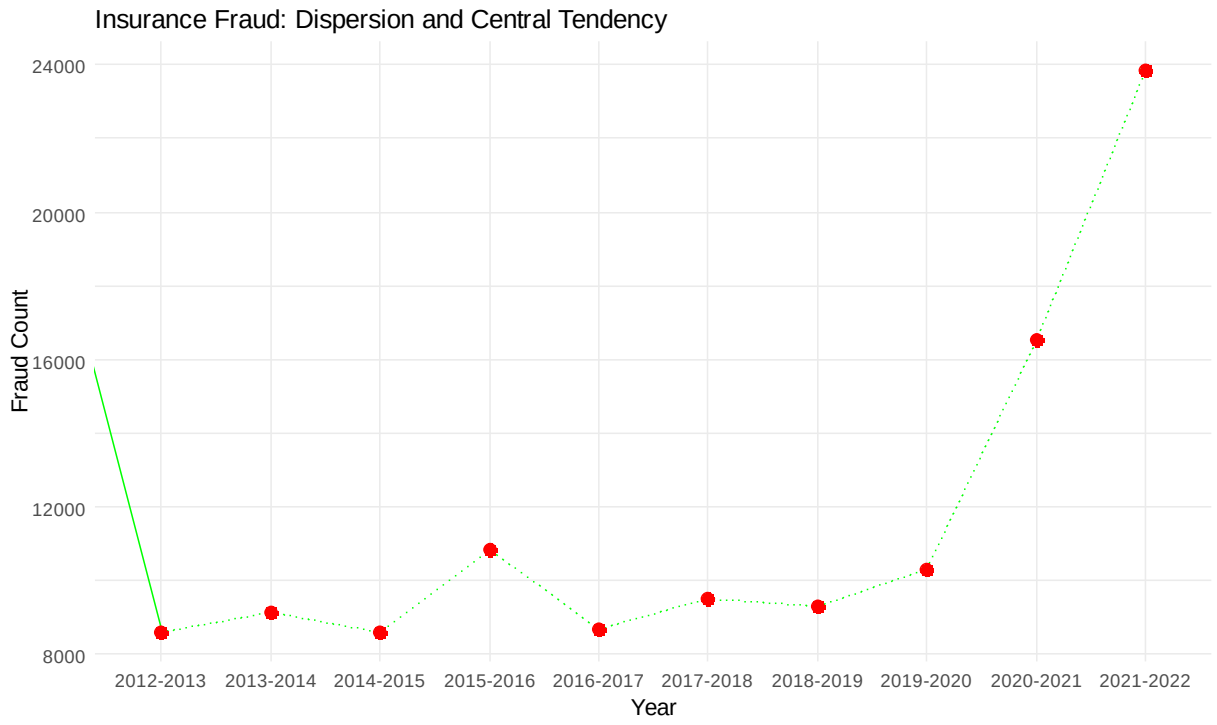
Part 2: Solution to R task

Question 1:

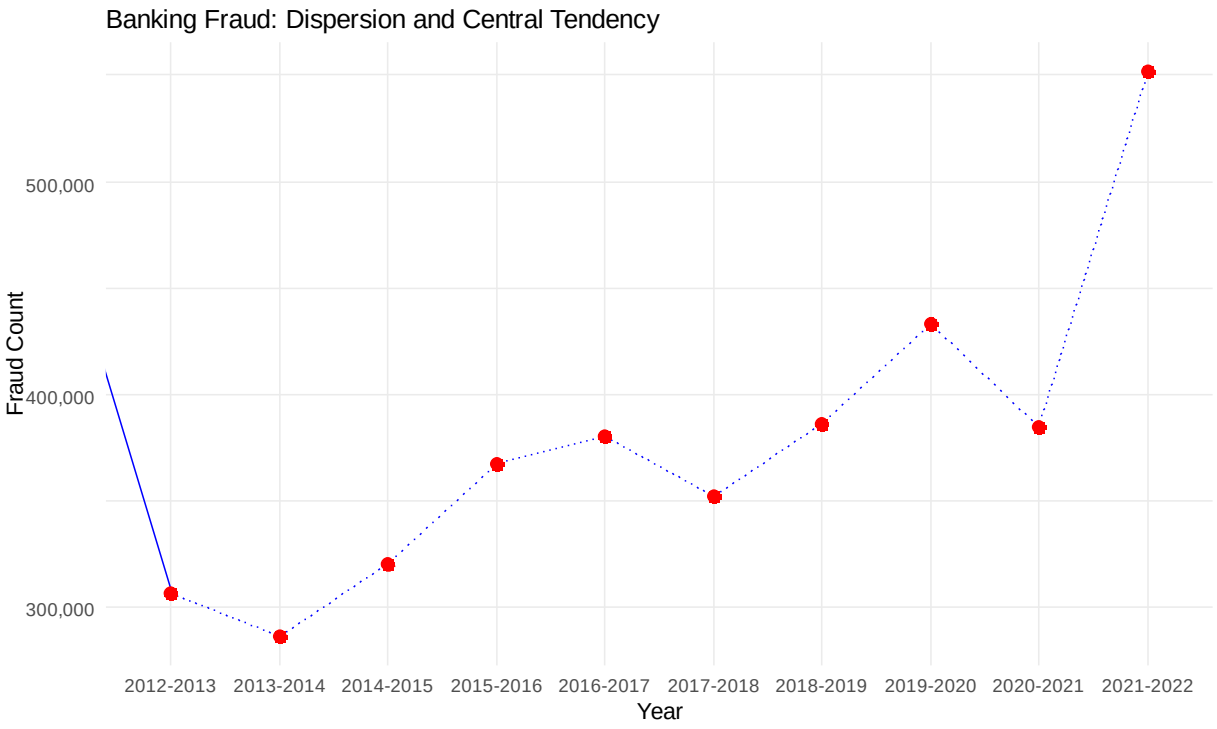
```
# Load necessary libraries

library(readxl)

library(dplyr)

library(ggplot2)

library(tidyr)

library(scales)


# Set the path to Excel file

file_path <- "filePath/cw_r.xlsx"


# We have to read the data from relevant sheet by skipping rows as need for these problem statements.

data <- read_excel(file_path, sheet = "Table 3d", skip = 8)


# But please make sure that the data is structured correctly so to read it well.

str(data)


# Changing column names to get easy access

colnames(data) <- c("Category", "2012-2013", "2013-2014", "2014-2015", "2015-2016",

                    "2016-2017", "2017-2018", "2018-2019", "2019-2020", "2020-2021",

                    "2021-2022", "Percentage_Change")


# Generate a data frame for each of the fraud types
```

```
# Slicing and dicing data for Data Science For All, 'Banking and Credit Industry Fraud'
banking_fraud <- data %>%
 filter(Category == "Banking and credit industry fraud") %>%
 select(-Category, -Percentage_Change) %>%
 pivot_longer(cols = starts_with("20"), names_to = "Year", values_to = "Count")

# Filter and reshape data for "Insurance Fraud"
insurance_fraud <- data %>%
 filter(Category == "Insurance fraud") %>%
 select(-Category, -Percentage_Change) %>%
 pivot_longer(cols = starts_with("20"), names_to = "Year", values_to = "Count")

# Displaying basic information about the data frames
str(banking_fraud)
str(cheque_fraud)

# Displaying the first few rows of the data frames to verify contents
head(banking_fraud)
head(cheque_fraud)

# Ensure the data is sorted by Year for proper line plotting
banking_fraud <- banking_fraud[order(banking_fraud$Year), ]
insurance_fraud <- insurance_fraud[order(insurance_fraud$Year), ]

# Showing basic information about data frames
ggplot(insurance_fraud, aes(x = Year, y = Count)) +
 geom_point(color = "green", size = 2) +          # Scatter plot for dispersion
 geom_line(aes(group = 1), color = "green", linetype = "dotted") +   # Line for trend
 stat_summary(fun = mean, geom = "point", color = "red", size = 3) + # Mean as central tendency
 labs(title = "Insurance Fraud: Dispersion and Central Tendency",
     x = "Year",
```

```
    y = "Fraud Count") +
  scale_y_continuous(labels = comma) +            # Format y-axis with commas
  theme_minimal()
```




```
# Exploring dispersions and central tendencies for Banking Fraud
ggplot(banking_fraud, aes(x = Year, y = Count)) +
  geom_point(color = "blue", size = 2) +            # Scatter plot for dispersion
  geom_line(aes(group = 1), color = "blue", linetype = "dotted") +    # Line for trend
  stat_summary(fun = mean, geom = "point", color = "red", size = 3) + # Mean as central tendency
  labs(title = "Banking Fraud: Dispersion and Central Tendency",
       x = "Year",
       y = "Fraud Count") +
  scale_y_continuous(labels = comma) +            # Format y-axis with commas
  theme_minimal()
```



Workflow:

- **Data Preparation:** I first filtered the base data (main dataset) into banking fraud and insurance fraud wise counts. Next, I used head() and str() to inspect the data frames were structured correctly for plotting.
- **Visualization:** I chose scatter plots with fitted lines (combined trend line and central tendency points) This option gives a clear picture of how much fraud has happened each year (dispersion) as well as the overall tendency, which is important to compare both types with respect to time fluctuation.

Justification for Plotting Choice:

A scatter plot with trend lines and central tendency points effectively communicates both individual data points and overall trends.

Given the annual nature of the data, this approach accurately captures variation, outliers, and central tendency.

The added average points make it easier to identify deviations from the mean, providing a deeper understanding of the statistical distribution of each fraudulent activity over time.

This method is visually informative and satisfies the requirement of understanding scatter and central tendency well.

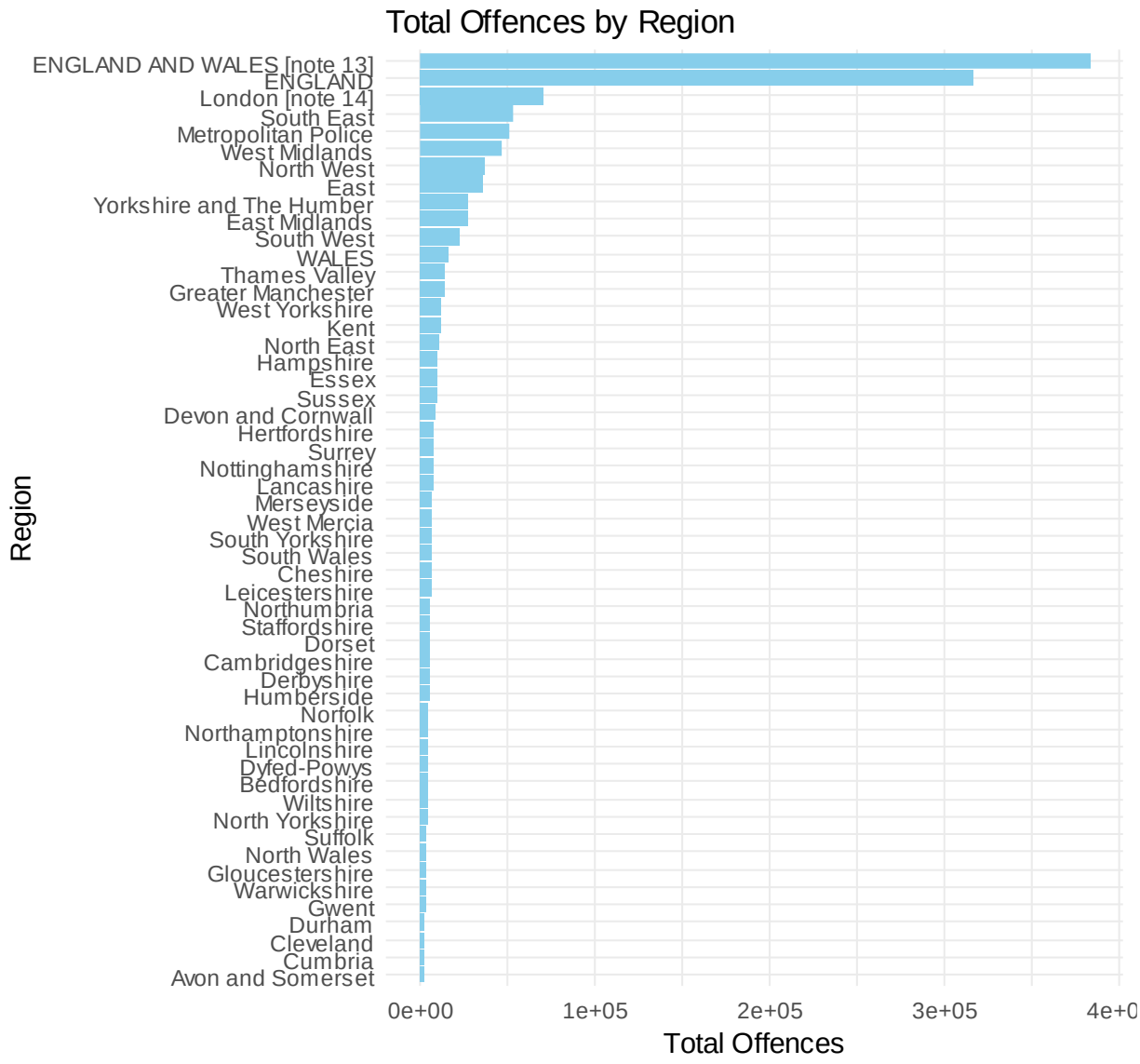
Question 2:

```
# Load necessary libraries
```

```
library(dplyr)

library(readxl)

library(ggplot2)

# outline the file path to the Excel data set .

file_path <- "FilePath/cw_r.xlsx"


# Load the data set from the specified Excel file.

# Skip the first 9 rows and access the data in the Table 5 sheet directly.

crime_data <- read_excel(file_path, sheet = "Table 5", skip = 9)


# Review the data set structure  to understand the columns and data types.

str(crime_data)


# Change the name of  the columns for easier access.

colnames(crime_data) <- c("Area_Code", "Area_Name", "Number_of_offences", "Rate_per_1000_population",
"Percent_change")


# Ensure that Rate_per_1000_population is numeric to allow accurate calculations.

crime_data$Rate_per_1000_population <- as.numeric(crime_data$Rate_per_1000_population)



# Filter out rows with NA values for "number_of_crimes" and "rate_per_1000_population".

crime_data <- crime_data %>%

  filter(!is.na(Number_of_offences) & !is.na(Rate_per_1000_population))


# Aggregate the data by area (Area_Name) to determine the total number of crimes and the average rate per 1000
population.

# Group  by "Area_Name" to provide an overview of crime and rates created per area

region_data <- crime_data %>%

  group_by(Area_Name) %>%

  summarize(
```

```
    total_offences = sum(Number_of_offences, na.rm = TRUE),
    avg_rate_per_1000 = mean(Rate_per_1000_population, na.rm = TRUE)
  )

# View aggregated area data to check accuracy
print(region_data)

# Visualize total crimes per area
ggplot(region_data, aes(x = reorder(`Area_Name`, total_offences), y = total_offences)) +
 geom_bar(stat = "identity", fill = "skyblue") +
 coord_flip() +
 labs(title = "Total Offences by Region", x = "Region", y = "Total Offences") +
 theme_minimal()
```

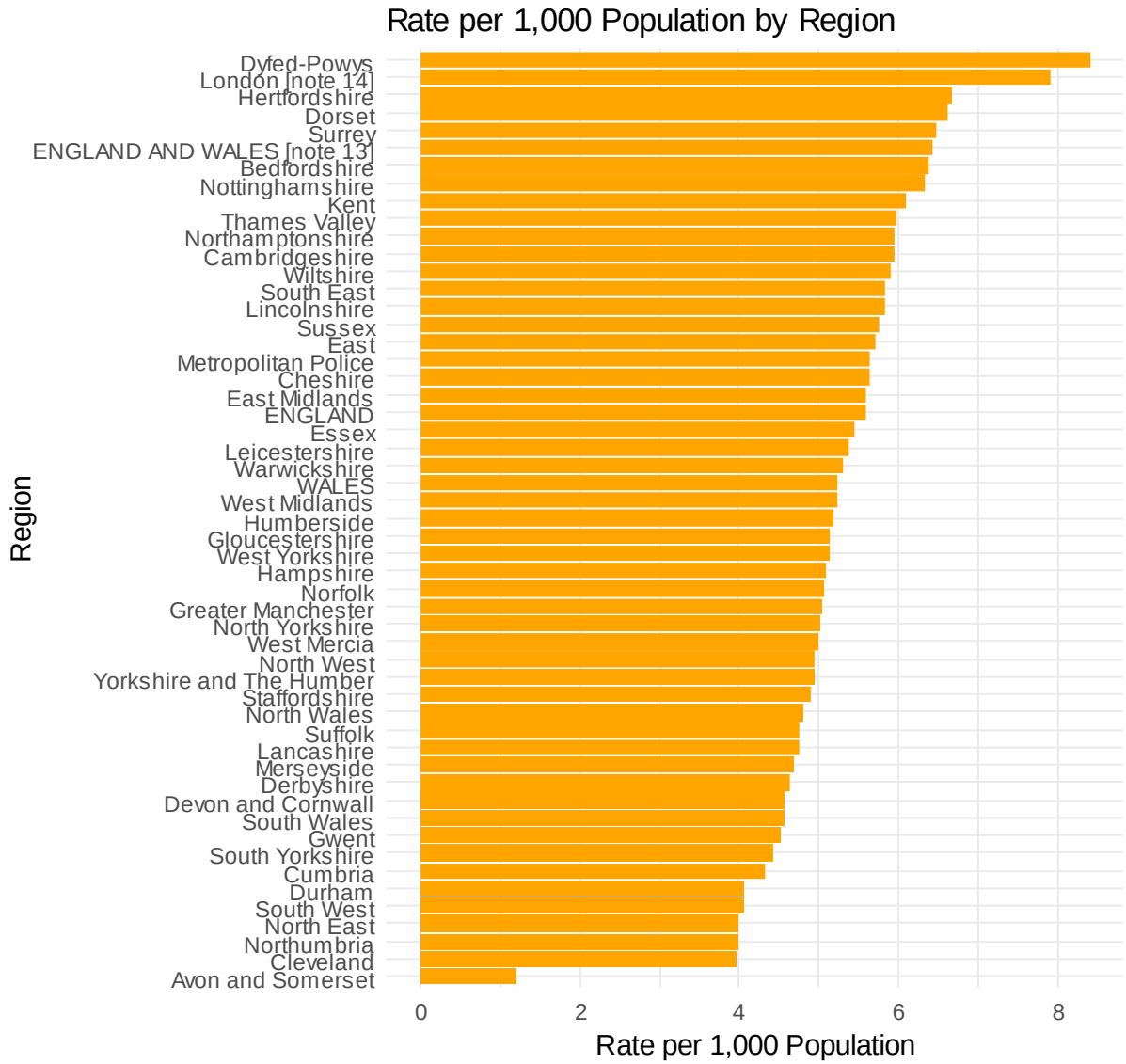
Explanation: A horizontal bar chart was chosen to facilitate comparison between regions.



Justification: A bar chart is ideal here as it clearly shows the difference in total crimes per region and makes it easy to identify the regions with the highest number of crimes.

```
# Show crime rates per 1,000 population per region

ggplot(region_data, aes(x = reorder(`Area_Name`, avg_rate_per_1000), y = avg_rate_per_1000)) +

 geom_bar(stat = "identity", fill = "orange") +

 coord_flip() +

 labs(title = "Rate per 1,000 Population by Region", x = "Region", y = "Rate per 1,000 Population") +

 theme_minimal()
```

Explanation: A bar chart helps to effectively compare crime rates between regions.

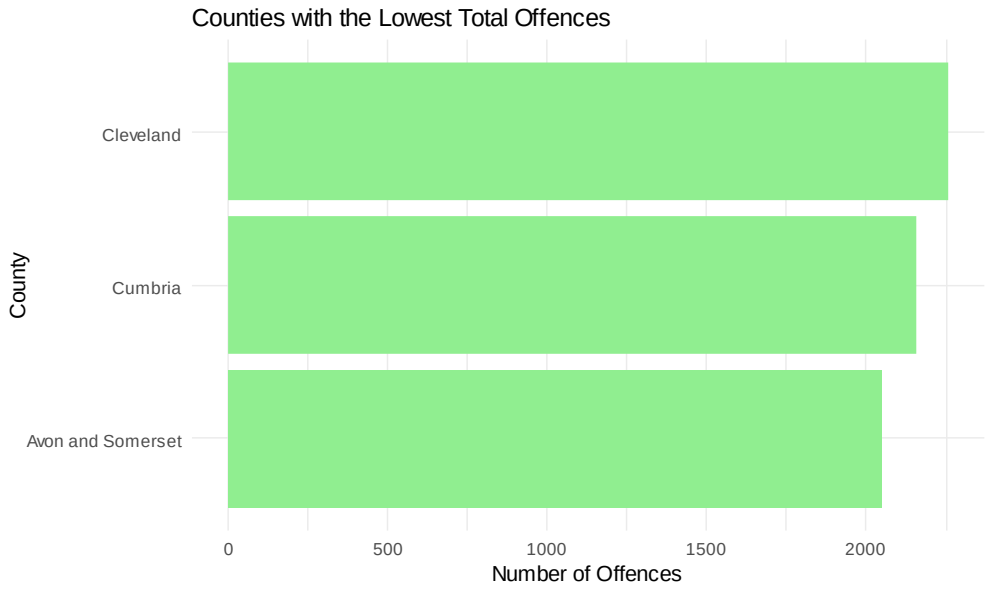


Justification: This bar chart was chosen for clarity and comparability as it makes it easy to visualize which regions have the highest crime rates per 1,000 population.

```
# Identify the three counties with the lowest total crimes.

# Sort the data by number_of_crimes to find the counties with the lowest number of crimes.

lowest_counties <- crime_data %>%

 arrange(Number_of_offences) %>%

 slice(1:3)
```

```
# Graph the Counties with the Lowest Total Crimes
ggplot(lowest_counties, aes(x = reorder(Area_Name, Number_of_offences), y = Number_of_offences)) +
 geom_bar(stat = "identity", fill = "lightgreen") +
 coord_flip() +
 labs(title = "Counties with the Lowest Total Offences", x = "County", y = "Number of Offences") +
 theme_minimal()
```

Explanation: A bar graph of the counties with the lowest total crimes provides a clear overview of this subset.



Justification: The bar chart clearly highlights the counties with the least crime, making them easy to identify and compare.