Part 1. Python tasks solutions

Q1 Solution:

```
import networkx as nx
import matplotlib.pyplot as plt

def create_delhi_metro_map():
            # Create an undirected graph

            G = nx.Graph()

          # Add nodes representing locations in the transportation system
            locations = ["Rithala", "Netaji Subhash Place", "Pratap Nagar", "Kashmere Gate", "Welcome", "Dilshad
            Garden", "Samaypur Badli", "Azadpur", "Vidhan Sabh" ...]
            G.add_nodes_from(locations)

            # Add edges representing connections between locations
            connections = [("Rithala", "Netaji Subhash Place", {'weight': 2.5}), ("Netaji Subhash Place", "Pratap Nagar",
            {'weight': 2.3}), ("Pratap Nagar", "Kashmere Gate", {'weight':  2.0}), ("Kashmere Gate", "Welcome",
            {'weight':  2.1}) ...]

            G.add_edges_from(connections)
```

Q2 Solution:

```
egde_list1 = [('Rithala', 'Netaji Subhash Place'), ('Netaji Subhash Place', 'Pratap Nagar'), ('Pratap Nagar', 'Kashmere
Gate'), ('Kashmere Gate', 'Welcome'), ('Welcome', 'Dilshad Garden')]
# other edges are added in similar ways.
 edges = [egde_list1, egde_list2, egde_list3, egde_list4, egde_list5]
```

Q3 Solution:

```
# Specify colors for nodes
    node_colors = {'Rithala': 'red', 'Netaji Subhash Place': 'red', 'Pratap Nagar': 'blue', 'Kashmere Gate': 'blue', 'Wel-
come': 'green', 'Dilshad Garden': 'black'}
...
```

Q4 Solution:

```
station_points = { # Manually adds the coordinates for the station names
        'Rithala': [0, 8.4],
        'Netaji Subhash Place': [1.6, 6],
        'Pratap Nagar': [2.8, 4.8],
        'Kashmere Gate': [4.7, 4.8],
        'Welcome': [5.8, 6.6],
        'Dilshad Garden': [7.5, 7.4],
...
```

```
def draw_graph(graph):
   # Draw the graph
   # explicitly set positions for the stations
   pos = {"Rithala": (0, 8), "Netaji Subhash Place": (1.3, 6), "Pratap Nagar": (3, 4), "Kashmere Gate": (4.5, 5),
        "Welcome": (6, 6), "Dilshad Garden": (7.5, 7), ...}
    plt.figure(figsize=(8, 8)) #adjust the figure size for our map.
    nx.draw(graph, pos, with_labels=False, node_size=300, font_size=10, font_color='black', linewidths=1,
node_color= node_color)

  for i in range(len(path_colors)): # draws the edges with specific colors and weights
           nx.draw_networkx_edges(graph, pos, edgelist=edges[i], edge_color=path_colors[i], width=3.0)
           nx.draw_networkx_edge_labels(graph, pos, edge_labels=edge_weights)
```

Q5 solution:

```
# plot the stations:
   for stations, value in station_points.items():
      plt.text(value[0], value[1], stations)

   plt.legend(handles = [l1, l2, l3, l4, l5], title= "Delhi Metro Map") # plots the legends
   # Save the plot as a PNG file
   plt.savefig('delhi_metro.png')
   plt.show()
```

Q6 Solution:

Here, two improvements that could be applied on the generated map that would improve the commute of the passenger are:

1. Better connectivity between the stations: We can see that although the map has quite good connections, there are still some areas where the connections could be added to improve the commute of the passengers (e.g., stations Botanical and NHPC are seen to be very close but passengers have to take a long route to reach there)
2. Add more stations on certain route: In the **Najafagarh – Raja Nahar** route, the distance between the stations are quite long. Here more stations could be added on short intervals to improve the commute experience.

Output:

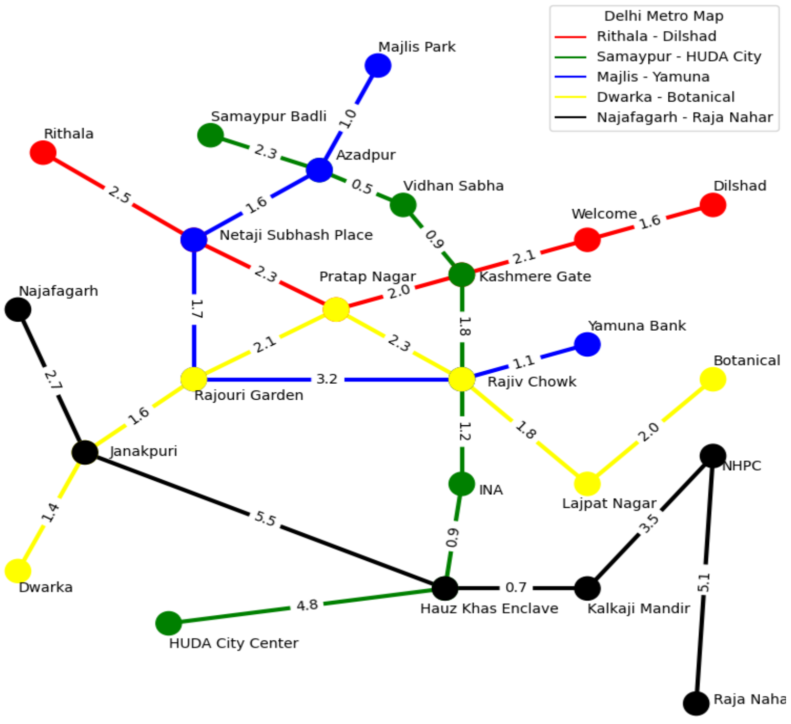

Figure 1: Delhi metro map

Part 2: R Task Solutions

Question 1:

1. Solution 1

```
# provide the full path where our file is located.
file_path = "C:/Users/pranjalp/Desktop/University of Greenwich/Fundamental_DS/labs/coursework/cw_r.xlsx"
sheets <- excel_sheets(file_path)
print(sheets)
```

The code above prints all the available sheets within our excel file.

2. Solution 2

The Excel file used for this coursework mainly contains fraud and computer misuse data. The main sources of the data are the National Fraud Intelligence Bureau (NFIB) and the Office for National Statistics (ONS). There are 4 major (5 including the table of contents) sheets in our file. Each sheet represents different types of data related to fraud. The sheets named “Table 3d” and “Table 5” contain the Fraud and computer misuse offenses referred to by the NFIB and the remaining sheets, “Table” and “Table 8” contain Fraud data that affected adults either by household and area characteristics or by personal characteristics.

We can observe many links between these datasets, the first being that all of these fraud data were recorded in England and Wales. Another link is that all of the recorded data were until March 2022. Different statistical methods are used in this coursework to visualize better and understand the patterns exhibited by these data.

Question 2:

1. Solution 1

```
# here the file path refers to the excel sheet path which we are using. The skip argument is used to
# skip the unnecessary rows and helps reduce the dataframe to only relevent data which is being used later.
fraud_and_comp_missues_by_type <- read_excel(file_path, sheet = "Table 3d", skip = 8)
fraud_and_comp_missues_by_area <- read_excel(file_path, sheet = "Table 5", skip = 9)
adult_victims_by_personal_chars <- read_excel(file_path, sheet = "Table 7", skip = 7)
adult_victims_by_household_and_area_chars <- read_excel(file_path, sheet = "Table 8", skip = 8)
```

2. The variables are defined above with each dataset being unique. Hyphen is used to divide the words resulting in a meaningful declaration of the variables.

Question 3:

Solutions to all the subtasks:

1. Two new dataframe named “banking_and_credit_fraud” and “consumer_and_retail_fraud” is created using the package dplyr. The slice, subset and pivot longer functions are used to preprocess our data before being plotted. The head() and str() methods are used to examine the structure and basic information about the dataset.

   The head() function returns the top 6 (default) rows of our dataset. It is mostly used to have a sneak peak to our data, especially if we have a large dataset without loading the whole dataset. The str() method provides the structure of our dataset. We can understand the structure of our dataframe and the datatype our data is composed of. The code implementation of the task is given below.

```
library(dplyr)

fraud_and_comp_missues_by_type_df = data.frame(fraud_and_comp_missues_by_type)

# Renaming the first column to "Fraud Types" for our convenience
colnames(fraud_and_comp_missues_by_type_df)[colnames(fraud_and_comp_missues_by_type_df) ==
"Reporting.body.and.fraud.and.computer.misuse.type..note.3..4."] <- "Fraud_Types"

banking_and_credit_fraud <- fraud_and_comp_missues_by_type_df %>%
slice(2:6)
consumer_and_retail_fraud <- fraud_and_comp_missues_by_type_df %>%
slice(32:38)

# Remove the last column as we do not use it in our evaluation.
banking_and_credit_fraud <- subset(banking_and_credit_fraud, select = -ncol(banking_and_credit_fraud))
consumer_and_retail_fraud <- subset(consumer_and_retail_fraud, select = -ncol(consumer_and_retail_fraud))

# Pivot the dataframe before plotting them, the pivot_longer function is used to pivot the dataframe.
banking_and_credit_fraud_new <- banking_and_credit_fraud %>%
pivot_longer(cols = -Fraud_Types, names_to = "fraud_date", values_to = "fraud_count")
consumer_and_retail_fraud_new <- consumer_and_retail_fraud %>%
pivot_longer(cols = -Fraud_Types, names_to = "fraud_date", values_to = "fraud_count")
head (banking_and_credit_fraud_new)
str(banking_and_credit_fraud_new)
head(consumer_and_retail_fraud_new)
str(consumer_and_retail_fraud_new)
```

2. Code for the visualization of dispersion and central tendency:

```
# Box plot for evaluation of the dispersion and central tendency.
# Here we have used the ggplot library for both the dataset to plot the data.
ggplot(banking_and_credit_fraud_new, aes(x = fraud_count, y = Fraud_Types)) + geom_boxplot() +
ggtitle("Banking and Credit Industry Fraud (Apr 2012 to Mar 2022)") + xlab("Fraud Cases") + ylab("Fraud Types")
theme_minimal() + stat_summary(fun = mean, geom = "point", shape = 16, size = 2, color = "orange")

ggplot(consumer_and_retail_fraud_new, aes(x = fraud_count, y = Fraud_Types, fill = fraud_count))+geom_boxplot()
+ ggtitle("Consumer and Retail Fraud (Apr 2012 to Mar 2022)") + xlab("Fraud Cases") + ylab("Fraud Types")
+theme_minimal() + theme(axis.text.x = element_text(angle = 45, hjust = 1)) + stat_summary(fun = mean, geom =
"point", shape = 16, size = 2, color = "orange")
```

Outputs:

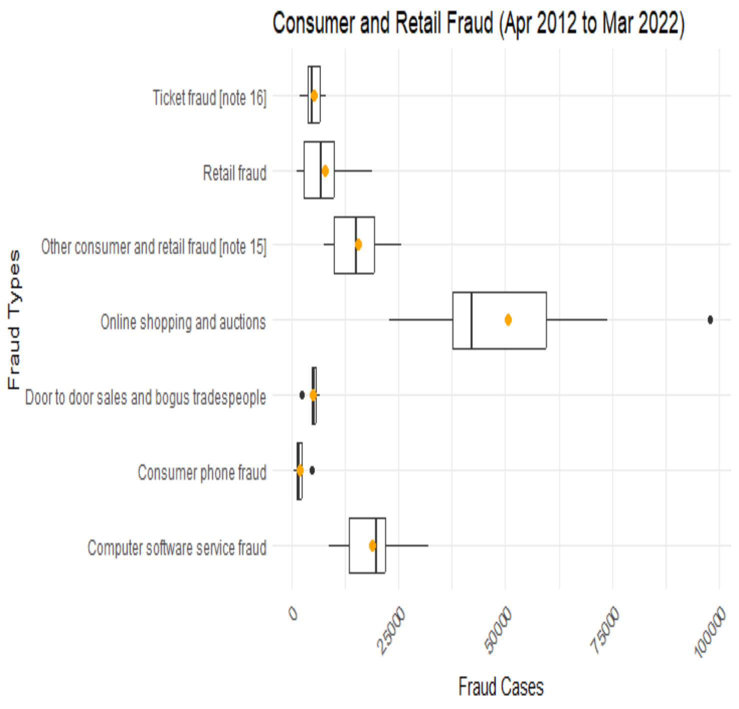

Figure 2: Dispersion and Central tendency for consumer and retail fraud

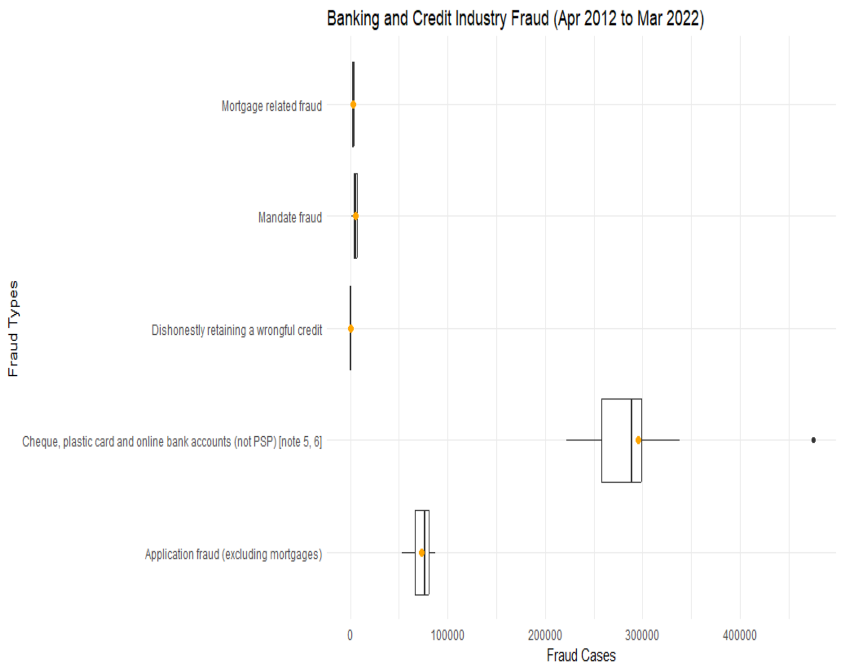

Figure 3: Dispersion and central tendency for banking and credit fraud

3. Plot Interpretation for dispersion and central tendency.

   To interpret the box plot we must know that, in our chart, the box represents the interquartile range that contains the middle 50% of our data. The line inside the box is the median value, which indicates the middle value of the dataset. The data points beyond the whiskers, shown as a black dot, represents any outlier present in our dataset. The whiskers represent the range of our data. Finally, the orange dot is used to visualize the mean value for each dataset.

   For the consumer and Retail Fraud, looking at the box plot, we can conclude that the most common type (with the highest number of fraud cases) of the fraud was "Online shopping and auctions" in the given date range whereas the least common fraud type was "Door to door sales and bogus trades people". We can see the presence of outliers in some of the fraud types (Online shopping, Door to door, Consumer phone).

   For the Banking and credit industry fraud, in the given date range, we can see the fraud type with the greatest number of cases reported was "cheques, plastic card and online bank account" whereas the least reported fraud cases were "dishonestly retaining a wrongful credit". We can see relatively a smaller number of outliers in this category of the dataset. The only visible outlier could be found on the most common type of fraud. While comparing the total number of fraud cases for each dataset, we can see that, in the same date range, the Banking and Credit Industry frauds were much higher in number (exceeding 4 million total cases) as compared to Consumer and retail fraud (approx. 1 million total cases).

4. To solve this task of visualizing and interpreting the dispersion and central tendency, we first analyzed our dataset. We manipulated our data, removed the non-relevant columns, renamed the longer columns, pivoted the data and finally sliced only the necessary rows before plotting them. We were able to achieve all those preprocessing steps using the R libraries.

   For our visualization part, we decided to use the box plot from the ggplot library. Box plot was the most suitable visualization tool for understanding both the dispersion and central tendency in our dataset. There are many advantages of using box plot over other tools for visualization. They are visually simple to understand both the central tendencies and dispersions. Box plots allows for very easy comparison of medians between different groups or datasets. They are less affected by skewness in the data compared to other methods. We can easily identify the presence of outliers in our data, as they are explicitly highlighted in the charts. Finally, box plots also provide a compact representation of major statistics which is quite important given the nature of our datasets.

Question 4:

Solution 1:

```
# slicing the necessary rows and removing the unnecessary columns (last)

fraud_and_comp_offences <- sheet_5 %>%
slice(3:50)
# removing yoy changes
fraud_and_comp_offences <- subset(fraud_and_comp_offences, select = -ncol(fraud_and_comp_offences))
# unique_area_codes   are used to differentiate between regions and counties.
area_codes <- c('E12000001', 'E12000002', 'E12000003', 'E12000004',  'E12000005', 'E12000006', 'E12000007',
'E12000008', 'E12000009')
region_level_data <- filter(fraud_and_comp_offences, `Area Code` %in% area_codes)
county_level_data <- fraud_and_comp_offences %>%
filter(!`Area Code` %in% area_codes)
```

The code above rearranges the data frame so that only the relevant data for our region and county are filtered and made ready to be used in plots below.

Solution 2:

```
# pie  chart
region_level_data$Percentage <- region_level_data$`Number of offences` / sum(region_level_data$`Number of
offences`) * 100
pie(region_level_data$Percentage, labels = paste(region_level_data$`Area Name`, ": ",
round(region_level_data$Percentage, 1), "%"), main = "Percentage of Total Offences by Region")

# converting our 'Rate per 1,000 population to numeric type.
region_level_data$`Rate per 1,000 population` <- as.numeric(region_level_data$`Rate per 1,000 population`)

# Create a horizontal bar chart for total count of offences by region
ggplot(region_level_data, aes(x = `Area Name`, y = `Rate per 1,000 population`, fill = `Area Name`)) + geom_bar(stat
= "identity") + labs(title = "Offence rate per 1000 by Region", x = "Region", y = "Rates") + theme_minimal() +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Output:

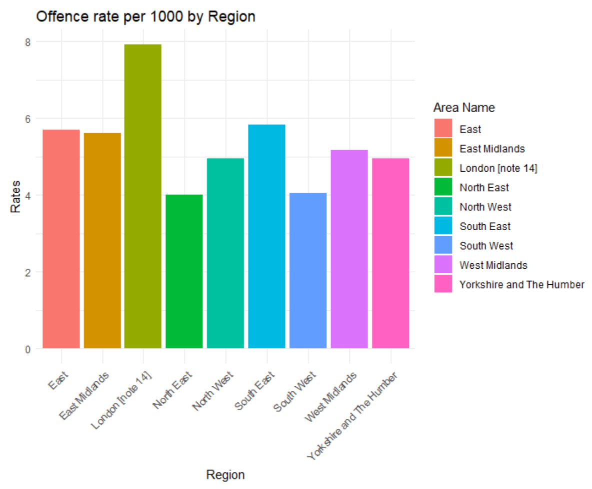

Figure 4: Bar chart representing the offence rate per 1000

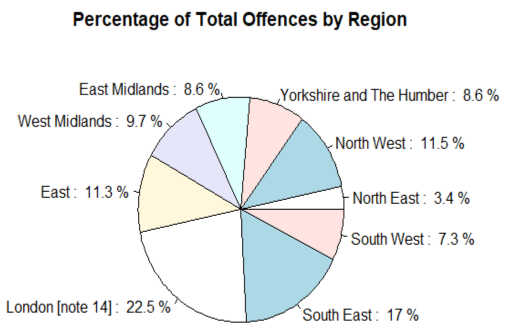

Figure 5: Pie Chart representing count of total offences

As per our visualization using the bar chart and pie chart, we can conclude that the region with the highest total count of offences as well as offence rate per 1000 was found to be **London.** From the pie chart, we can see that London 22.5 % of the total offences by region and the offence rate per 1000 there is nearly 8 making it the region with the highest values in both the conditions.

We have made the use of bar chart and pie chart for our visualization as both the tools are a simple and powerful way of representing the comparison data. They are easy to read and summarize. Pie charts makes it very easy to visualize the percentages and fractions. They also work very well when we have relatively lesser number of categories (regions) in our case. To conclude, looking at the nature of our dataset, their complexity and the result to derive, those two methods were the best ones.

Solution 3:

```
# Sort the data by Total_Offences in ascending order, easy to visualize the top 3 counties.
county_level_data <- county_level_data[order(county_level_data$`Number of offences`), ]

# Specify the order of counties
county_level_data$`Area Name` <- factor(county_level_data$`Area Name`, levels = county_level_data$`Area
Name`)

# Create a horizontal bar chart
ggplot(county_level_data, aes(x = county_level_data$`Number of offences`, y = `Area Name`)) + geom_bar(stat =
"identity", fill = "skyblue") + labs(title = "Total Count of Offences by County", x = "Total Offences", y = "County") +
theme_minimal()
```

Output:

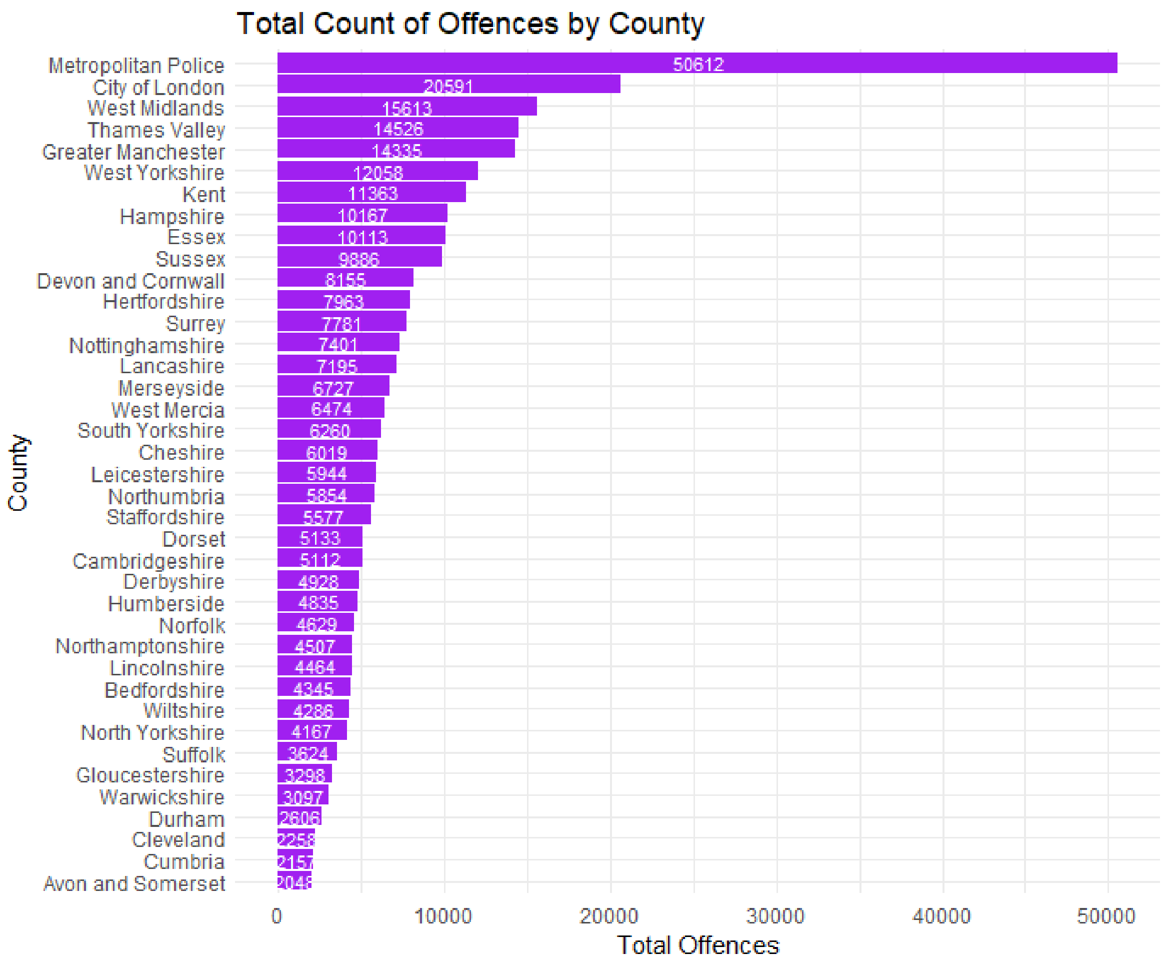

Figure 6: Horizontal bar chart for top 3 counties.

Explanation of the Visualization:

As per our visualization, the top three counties that have the lowest total count of offences were “**Avon and Somerset (2048)**”, “**Cumbria (2157)**” and “**Cleveland (2258)”.**

For the solution of this task, a horizontal bar chart was used. Before plotting the data to the charts, they were sorted in arranging order so that it would be easy to identity the counties based on offences. The bar chart was chosen because our requirement of the task was comparing the dataset between different counties and reach to a conclusion. Since bar charts are mostly suitable for the comparison tasks and they also provided a better visualization approach to accommodate all of our counties horizontally, bar chart was the preferred visualization technique.