# ETW2001 Foundations of Data Analysis

Foo Kai Yan 33085625

kfoo0012@student.monash.edu

## Section A

After setting the appropriate working directory, the necessary libraries are loaded into the working directory. The environment is thoroughly cleaned to prevent any possible conflicts in the code and to guarantee that only the necessary files are correctly loaded and processed in the environment.

```
R 4.3.3 · C:/Monash/ETW2001/
> ################################################################################
> #            Student Name: Foo Kai Yan                                          #
> #            Student ID: 33085625                                               #
> #            Student Email: kfoo0012@student.monash.edu                         #
> ################################################################################
> # Set Working Directory
> setwd("C:/Monash/ETW2001")
>
> # Load required libraries
> library(tidyverse)
> library(ggplot2)
> library(tidyr)
> library(dplyr)
>
> ################################################################################
> #                                  SECTION A                                    #
> ################################################################################
> # Remove/Clean the environment
> rm(list=ls())
> # Load the provided datasets
> cust_dataset = read.csv("olist_customers_dataset.csv", header = TRUE)
> geoloc_dataset = read.csv("olist_geolocation_dataset.csv", header = TRUE)
> ord_itm_dataset = read.csv("olist_order_items_dataset.csv ", header = TRUE)
> ord_pay_dataset = read.csv("olist_order_payments_dataset.csv ", header = TRUE)
> ord_rvw_dataset = read.csv("olist_order_reviews_dataset.csv ", header = TRUE)
> ord_dataset = read.csv("olist_orders_dataset.csv", header = TRUE)
> prd_dataset = read.csv("olist_products_dataset.csv", header = TRUE)
> sel_dataset = read.csv("olist_sellers_dataset.csv ", header = TRUE)
> prd_cat_name_dataset = read.csv("product_category_name_translation.csv ", header = TRUE)
```

1. After inner join is performed, there are 14 columns in total in the new dataset which includes the columns: order_id, customer_id, order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date, order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value.

   One of the insights that can be derived from the merged dataset includes getting the number of items within each orders made which is stored in the variable order_item_count. Another insight that can be derived from the merged dataset includes the minimum and maximum of items present in the orders recorded which the result is stored in order_item_stats. The highest quantity of items in one order is 21 and the lowest quantity recorded is 1. One of the last insights obtained from the merged dataset is finding the product with the maximum and minimum number of orders. The maximum number of orders a product has received is 527 which has the product_id of aca2eb7d00ea1a7b8ebd4e68314663af. The minimum number of orders a product has received is 1 which has been shown in the variable min_orders. There are more than 1 products that have only 1 order recorded.

Inner join:

```
> # Inner join between ord_itm_dataset and ord_dataset
> inner_joined_dataset <- inner_join(ord_dataset, ord_itm_dataset)
Joining with `by = join_by(order_id)`
> names(inner_joined_dataset)
 [1] "order_id"                      "customer_id"                   "order_status"
 [4] "order_purchase_timestamp"      "order_approved_at"             "order_delivered_carrier_date"
 [7] "order_delivered_customer_date" "order_estimated_delivery_date" "order_item_id"
[10] "product_id"                    "seller_id"                     "shipping_limit_date"
[13] "price"                         "freight_value"
```

Insight 1: Number of items within each order made

```
> order_item_count <- inner_joined_dataset %>%
+   group_by(order_id) %>%
+   summarise(number_of_items = n())
> order_item_count
# A tibble: 98,666 × 2
   order_id                         number_of_items
   <chr>                                      <int>
 1 00010242fe8c5a6d1ba2dd792cb16214               1
 2 00018f77f2f0320c557190d7a144bdd3               1
 3 000229ec398224ef6ca0657da4fc703e               1
 4 00024acbcdf0a6daa1e931b038114c75               1
 5 00042b26cf59d7ce69dfabb4e55b4fd9               1
 6 00048cc3ae777c65dbb7d2a0634bc1ea               1
 7 00054e8431b9d7675808bcb819fb4a32               1
 8 000576fe39319847cbb9d288c5617fa6               1
 9 0005a1a1728c9d785b8e2b08b904576c               1
10 0005f50442cb953dcd1d21e1fb923495               1
# ℹ 98,656 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Insight 2: Minimum and maximum of items present in the orders

```
> # Calculate max & min number of items in the order
> order_item_stats <- inner_joined_dataset %>%
+   group_by(order_id) %>%
+   summarise(number_of_items = n()) %>%
+   summarise(max_items = max(number_of_items), min_items = min(number_of_items))
> order_item_stats
# A tibble: 1 × 2
  max_items min_items
      <int>     <int>
1        21         1
```

Insight 3: Product(s) with the maximum and minimum number of orders

```
> # Count number of order for each item
> number_of_order_each_item <- inner_joined_dataset %>%
+   group_by(product_id) %>%
+   summarise(number_of_orders = n())
> # Find the product_id with the maximum number of orders
> max_orders <- number_of_order_each_item %>%
+   filter(number_of_orders == max(number_of_orders))
> max_orders
# A tibble: 1 × 2
  product_id                       number_of_orders
  <chr>                                       <int>
1 aca2eb7d00ea1a7b8ebd4e68314663af              527
> # Find the product_id with the minimum number of orders
> min_orders <- number_of_order_each_item %>%
+   filter(number_of_orders == min(number_of_orders))
> min_orders
# A tibble: 18,117 × 2
   product_id                       number_of_orders
   <chr>                                       <int>
 1 00066f42aeeb9f3007548bb9d3f33c38                1
 2 00088930e925c41fd95ebfe695fd2655                1
 3 0009406fd7479715e4bef61dd91f2462                1
 4 000d9be29b5207b54e86aa1b1ac54872                1
 5 0011c512eb256aa0dbbb544d8dffcf6e                1
 6 001b237c0e9bb435f2e54071129237e9                1
 7 001c5d71ac6ad696d22315953758fa04                1
 8 0021a87d4997a48b6cef1665602be0f5                1
 9 002552c0663708129c0019cc97552d7d                1
10 002959d7a0b0990fe2d69988affcbc80                1
# ℹ 18,107 more rows
# ℹ Use `print(n = ...)` to see more rows
```

1. After left join is performed between ord_dataset and ord_rvw_dataset, it was found out that there are 768 orders present in the dataset that does not have any reviews given.

Left join:

```
> left_joined_dataset <- left_join(ord_dataset, ord_rvw_dataset)
Joining with `by = join_by(order_id)`
> names(left_joined_dataset)
 [1] "order_id"
 [2] "customer_id"
 [3] "order_status"
 [4] "order_purchase_timestamp"
 [5] "order_approved_at"
 [6] "order_delivered_carrier_date"
 [7] "order_delivered_customer_date"
 [8] "order_estimated_delivery_date"
 [9] "review_id"
[10] "review_score"
[11] "review_comment_title"
[12] "review_comment_message"
[13] "review_creation_date"
[14] "review_answer_timestamp"
> # Count the number of orders without reviews by using all columns from ord_rvw_dataset
> orders_without_reviews <- sum(rowSums(is.na(left_joined_dataset[, c("review_id", "review_score", "review_commen
t_title", "review_comment_message", "review_creation_date", "review_answer_timestamp")])) > 0)
> orders_without_reviews
[1] 768
```

2. After right join has been performed between ord_itm_dataset and prd_dataset, it has been discovered that there are 0 products that has not been sold which implies that all products recorded in the dataset have been sold at least once as order_id is the unique identifier for each order which means that each order have only 1 unique order_id and order_item_id is the identifier for each item within an order which means each product have its own unique order_item_id.

Right join:

```
> # Right join between ord_itm_dataset and prd_dataset
> right_joined_dataset <- right_join(ord_itm_dataset, prd_dataset)
Joining with `by = join_by(product_id)`
> not_sold <- sum(rowSums(is.na(right_joined_dataset[, c("order_id", "order_item_id")])) > 0)
> not_sold
[1] 0
```

3. After full outer join has been performed between cust_dataset and ord_dataset, findings have indicated that all orders have customer details, and all customers have order details. The variable customers_without_orders contain the data on customers without orders whereas the variable orders_without_customers contain the data of orders without customer details. customer_id is the unique identifier for each customer which means they can't be repeated.

Full outer join:

```
> # Full outer join between cust_dataset and ord_dataset
> full_out_dataset <- full_join(cust_dataset, ord_dataset)
Joining with `by = join_by(customer_id)`
> # Check for customers without orders
> customers_without_orders <- sum(is.na(full_out_dataset$order_id))
> customers_without_orders
[1] 0
> # Check for orders without customer details
> orders_without_customers <- sum(is.na(full_out_dataset$customer_id))
> orders_without_customers
[1] 0
```

4. After semi join has been done, the data output has showed that there are 3095 sellers within the seller dataset and there is also a total of 3095 active sellers within the dataset, which suggested that all sellers in the dataset are active sellers.

Semi join:

```
> semi_join_dataset <- semi_join(sel_dataset, ord_itm_dataset)
Joining with `by = join_by(seller_id)`
> names(semi_join_dataset)
[1] "seller_id"              "seller_zip_code_prefix" "seller_city"            "seller_state"
> total_sellers <- nrow(sel_dataset)
> total_sellers
[1] 3095
> active_sellers_count <- nrow(semi_join_dataset)
> active_sellers_count
[1] 3095
> dim(sel_dataset) # There is a total of 3095 rows
[1] 3095    4
```

5. After anti join has been executed, the investigations have uncovered that all customers within the dataset have made orders before as the result of the anti join shows that the anti_dataset which is the result of the anti join have 0 rows and 5 columns.

Anti join:

```
> anti_dataset <- anti_join(cust_dataset, ord_dataset)
Joining with `by = join_by(customer_id)`
> names(anti_dataset)
[1] "customer_id"              "customer_unique_id"       "customer_zip_code_prefix"
[4] "customer_city"            "customer_state"
> head(anti_dataset)
[1] customer_id              customer_unique_id       customer_zip_code_prefix customer_city
[5] customer_state
<0 rows> (or 0-length row.names)
> dim(anti_dataset)
[1] 0 5
```

6. Left join has been used to merge ord_itm_dataset with ord_dataset based on a common key, which is order_id. This join will ensure that all records from ord_itm_dataset are retained in the resulting order_left_join dataset. If there are any order_ids in ord_itm_dataset that don't have a matching order_id in ord_dataset, those records will still be included in the order_left_join with NA filled in for the missing columns from ord_dataset. Left join was used again to preserve the records from order_left_join when merging with prd_dataset and sel_dataset so records from order_left_join is maintained when more information from prd_dataset and sel_dataset was added. The final dataset used after all the left join is stored in order_prd_sell. It can be seen that all unique columns from prd_dataset, sel_dataset, ord_itm_dataset and ord_dataset has been included in order_prd_sell which has 112650 rows of data and 25 columns in total.

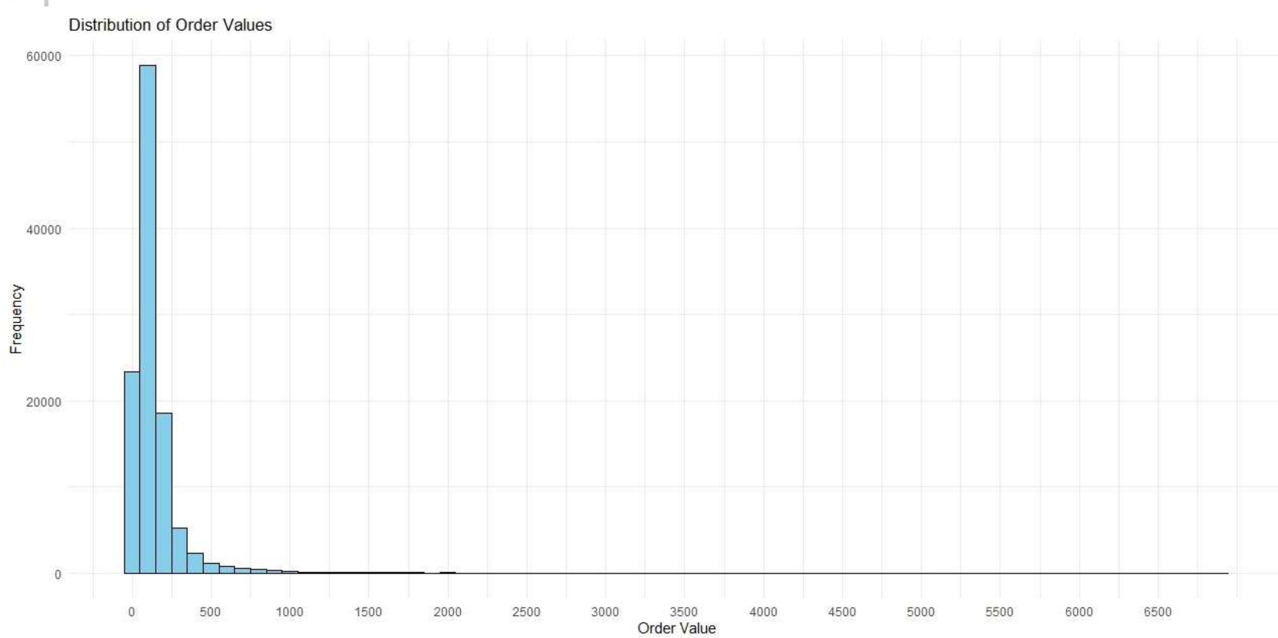
   Findings has shown that the flow from sellers to products and items within orders goes like sellers sell products to consumers and each product has specific characteristics such as category, weight, dimensions, and price. And when a customer places an order, they select one or more products where each product becomes an item within the order. So, when the consumer place an order on their purchases of products from multiple sellers, there would be multiple order items quantity listed under a single order. This can result in the consumers' order to have more than one freight value being calculated as there might be more than one seller the consumer bought the product item from.

Left join:

```
> # Left join between ord_itm_dataset and ord_dataset
> order_left_join <- left_join(ord_itm_dataset, ord_dataset)
Joining with `by = join_by(order_id)`
> # Left join between order_left_join and prd_dataset
> order_left_join_prd <- left_join(order_left_join, prd_dataset)
Joining with `by = join_by(product_id)`
> # Left join between order_left_join_prd and sel_dataset
> order_prd_sell <- left_join(order_left_join_prd, sel_dataset)
Joining with `by = join_by(seller_id)`
> # Column names
> names(order_prd_sell)
 [1] "order_id"                      "order_item_id"                 "product_id"
 [4] "seller_id"                     "shipping_limit_date"           "price"
 [7] "freight_value"                 "customer_id"                   "order_status"
[10] "order_purchase_timestamp"      "order_approved_at"             "order_delivered_carrier_date"
[13] "order_delivered_customer_date" "order_estimated_delivery_date" "product_category_name"
[16] "product_name_lenght"           "product_description_lenght"    "product_photos_qty"
[19] "product_weight_g"              "product_length_cm"             "product_height_cm"
[22] "product_width_cm"              "seller_zip_code_prefix"        "seller_city"
[25] "seller_state"
> # Dimension
> dim(order_prd_sell)
[1] 112650     25
> # Basic Summary
> summary(order_prd_sell)
```

Visualization:

```
> # Visualizing Distribution of Order Value: (using ggplot)
> # Histogram on order value
> order_values <- order_prd_sell$price + order_prd_sell$freight_value
> ggplot(data = data.frame(OrderValue = order_values), aes(x = OrderValue)) +
+   geom_histogram(binwidth = 100, fill = "skyblue", color = "black") +
+   labs(title = "Distribution of Order Values",
+        x = "Order Value",
+        y = "Frequency") +
+   theme_minimal() +
+   scale_x_continuous(breaks = seq(0, max(order_values), by = 500))
>
```




Order values refer to the total amount of the products' price and freight value combined. The distribution of order values is shown in the histogram, indicating that many orders fall within the range of 0-500, with a noticeable decrease in frequency as the order values increase. To sum up, the histogram highlighted that most orders have lower values, with higher-value orders being less frequent.

## Section B

For this section of the assignment, the sales dataset was downloaded from the Kaggle with using the link attached within the article “customer sales analysis.pdf”. The dataset was named sales_data and was loaded into the environment.

This is the Kaggle link: https://www.kaggle.com/datasets/kyanyoga/sample-sales-data/data

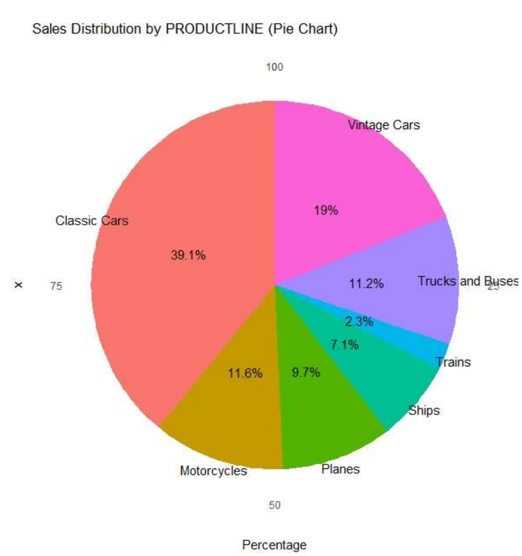
```
> ##############################################################################
> #                                  SECTION B                                 #
> #----------------------------------------------------------------------------#
> ##############################################################################
> # Load sales_data obtain from Kaggle
> # Source: https://www.kaggle.com/datasets/kyanyoga/sample-sales-data/data
> sales_data = read.csv("sales_data_sample.csv", header = TRUE)
> names(sales_data)
 [1] "ORDERNUMBER"      "QUANTITYORDERED"  "PRICEEACH"        "ORDERLINENUMBER"  "SALES"
 [6] "ORDERDATE"        "STATUS"           "QTR_ID"           "MONTH_ID"         "YEAR_ID"
[11] "PRODUCTLINE"      "MSRP"             "PRODUCTCODE"      "CUSTOMERNAME"     "PHONE"
[16] "ADDRESSLINE1"     "ADDRESSLINE2"     "CITY"             "STATE"            "POSTALCODE"
[21] "COUNTRY"          "TERRITORY"        "CONTACTLASTNAME"  "CONTACTFIRSTNAME" "DEALSIZE"
>
```

In this section, other than question 5 that include an intepretation of bar plot, question 1 to 4 will only contain the R code and the graph output.

R code:

```
> #----------------------------------------------------------------------------#
> #|                                QUESTION 1                                |#
> #----------------------------------------------------------------------------#
> # Calculate the total sales for each product line
> count <- aggregate(SALES ~ PRODUCTLINE, data = sales_data, sum)
> # Calculate the percentage of sales for each product line
> count$Percentage <- 100 * count$SALES / sum(count$SALES)
> # Create a pie chart using ggplot
> ggplot(data = count, aes(x = "", y = Percentage, fill = PRODUCTLINE)) +
+   geom_bar(stat = "identity") +
+   coord_polar(theta = "y") +
+   labs(title = "Sales Distribution by PRODUCTLINE (Pie Chart)",
+        fill = "Product Line") +
+   theme_minimal() +
+   theme(legend.position = "none") +
+   geom_text(aes(label = paste0(round(Percentage, 1), "%")),
+             position = position_stack(vjust = 0.5)) +
+   geom_text(aes(x = 1.5, label = PRODUCTLINE),
+             position = position_stack(vjust = 0.5)) +
+   theme(panel.background = element_blank(),
+         panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank())
```

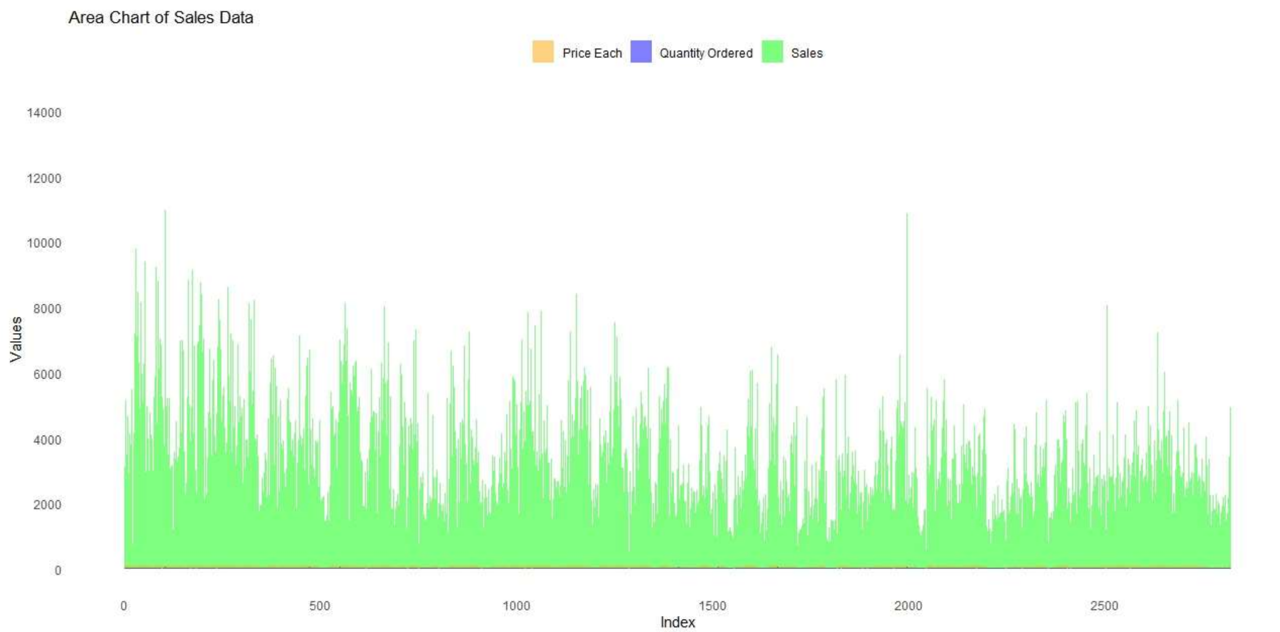
Pie Chart:



R code:

```
> #----------------------------------------------------------------------------#
> #|                                 QUESTION 2                               |#
> #----------------------------------------------------------------------------#
> ggplot(sales_data, aes(x = 1:nrow(sales_data))) +
+   geom_area(aes(y = SALES, fill = "Sales"), alpha = 0.5) +
+   geom_area(aes(y = PRICEEACH, fill = "Price Each"), alpha = 0.5) +
+   geom_area(aes(y = QUANTITYORDERED, fill = "Quantity Ordered"), alpha = 0.5) +
+   scale_fill_manual(values = c("Quantity Ordered" = "blue", "Price Each" = "orange", "Sales" = "green")) +
+   labs(title = "Area Chart of Sales Data",
+        x = "Index",
+        y = "Values") +
+   theme_minimal() +
+   scale_x_continuous(breaks = seq(0, nrow(sales_data), by = 500)) +
+   scale_y_continuous(breaks = seq(0, max(sales_data$SALES), by = 2000)) +
+   theme(legend.position = "top")+
+   theme(legend.title=element_blank()) +
+   theme(panel.background = element_blank(),
+         panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank())
```

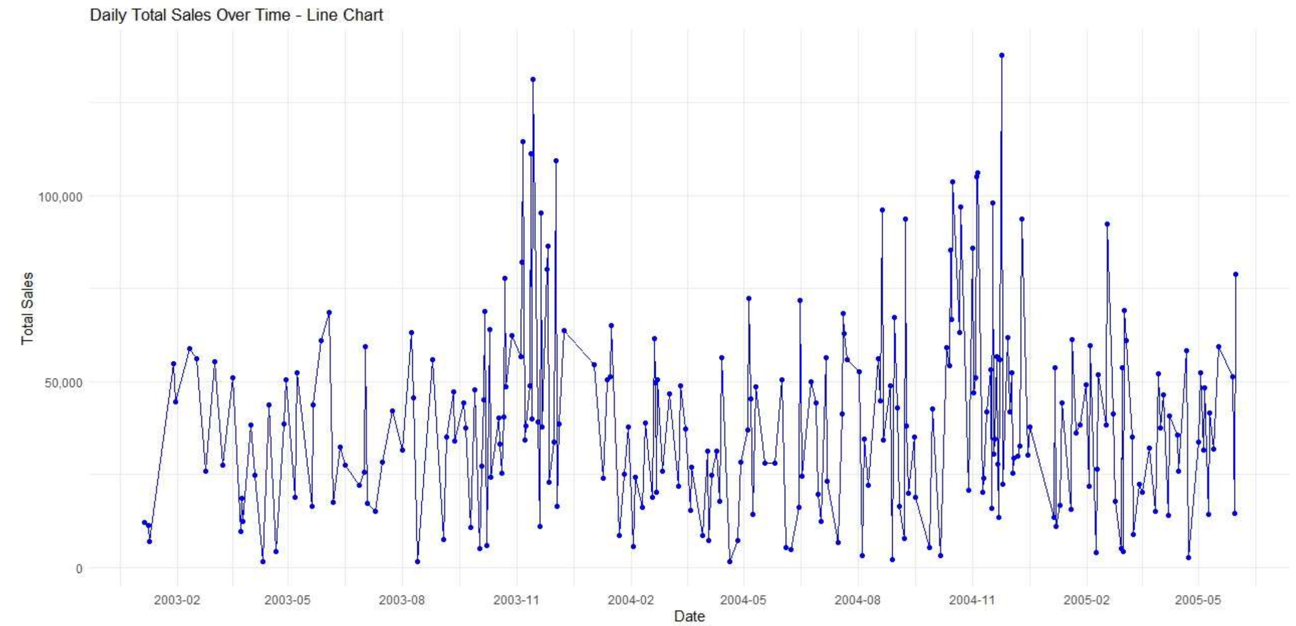
Area Chart:




R code:

```
> #----------------------------------------------------------------------------#
> #|                                 QUESTION 3                               |#
> #----------------------------------------------------------------------------#
> sales_data$ORDERDATE <- as.Date(sales_data$ORDERDATE, format = "%m/%d/%Y %H:%M")
> daily_total_sales <- sales_data %>%
+   group_by(ORDERDATE) %>%
+   summarise(Total_Sales = sum(SALES))
> # Create the line chart
> ggplot(daily_total_sales, aes(x = ORDERDATE, y = Total_Sales)) +
+   geom_line(color = "blue") +
+   geom_point(color = "blue") +
+   scale_x_date(date_labels = "%Y-%m", date_breaks = "3 months") +
+   scale_y_continuous(labels = scales::comma) +
+   labs(title = "Daily Total Sales Over Time - Line Chart",
+        x = "Date",
+        y = "Total Sales") +
+   theme_minimal()
```

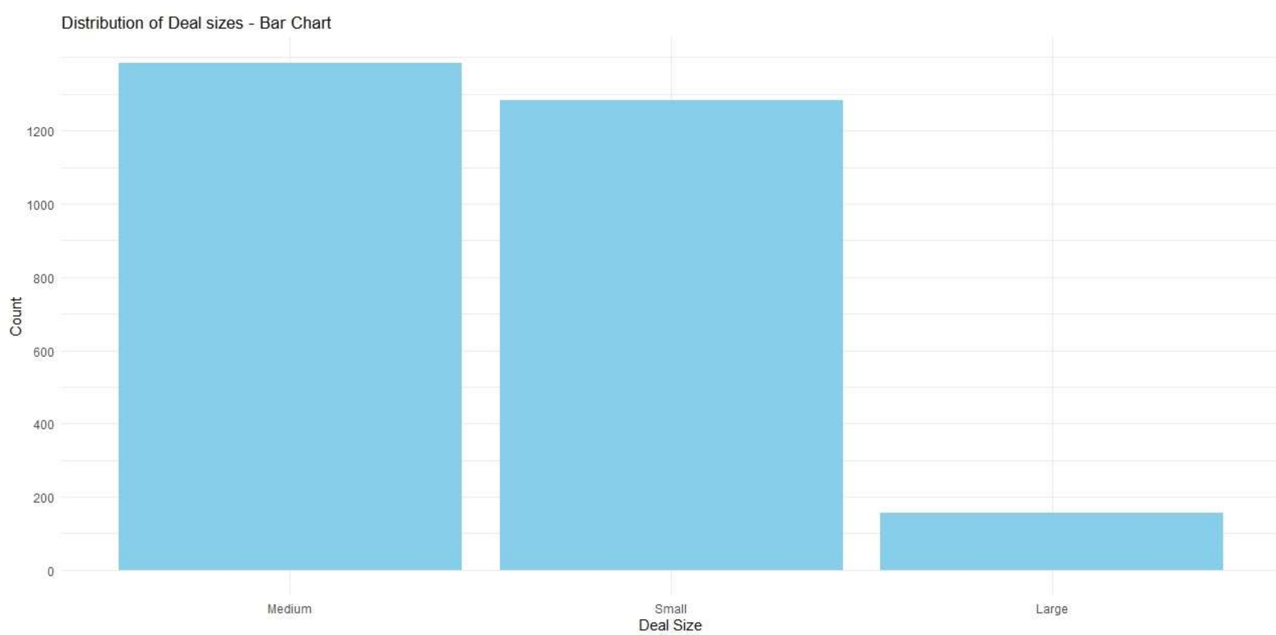
Line Chart:




R code:

```
> #----------------------------------------------------------------------------#
> #|                              QUESTION 4                                  |#
> #----------------------------------------------------------------------------#
> deal_sizes <- sales_data %>%
+   group_by(DEALSIZE) %>%
+   summarise(Count = n())
> ggplot(deal_sizes, aes(x = reorder(DEALSIZE, -Count), y = Count)) +
+   geom_bar(stat = "identity", fill = "skyblue") +
+   labs(
+     title = "Distribution of Deal sizes - Bar Chart",
+     x = "Deal Size",
+     y = "Count"
+   ) +
+   scale_y_continuous(breaks = seq(0, max(deal_sizes$Count), by = 200)) +
+   theme_minimal()
```

Bar Chart:



R code:

```
> #--------------------------------------------------------------------------#
> #|                                 QUESTION 5                              |#
> #--------------------------------------------------------------------------#
> sales_by_country <- sales_data %>%
+   group_by(COUNTRY) %>%
+   summarise(Number_of_Sales = n())
> ggplot(sales_by_country, aes(x = reorder(COUNTRY, -Number_of_Sales), y = Number_of_Sales)) +
+   geom_bar(stat = "identity", fill = "skyblue") +
+   labs(
+     title = "Number of Sales by Country",
+     x = "Country",
+     y = "Number of Sales"
+   ) +
+   scale_y_continuous(breaks = seq(0, max(sales_by_country$Number_of_Sales), by = 200)) +
+   theme_minimal() +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
>
```

Bar Plot:

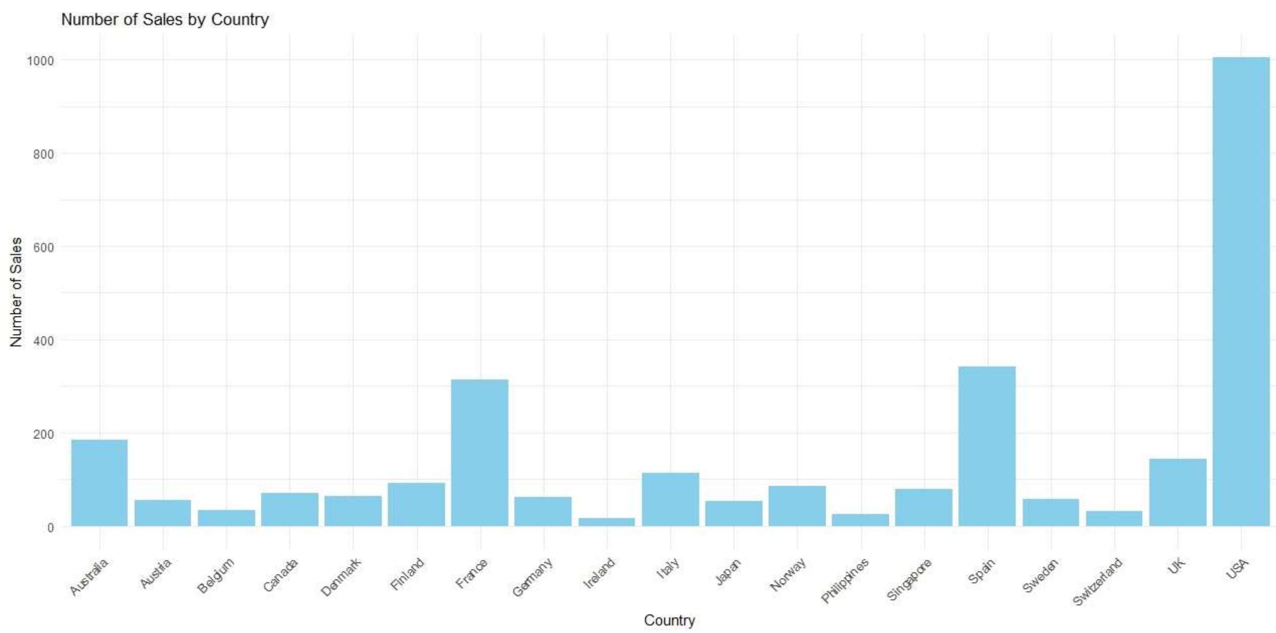



Fig.23 shows the breakdown of sales in various countries, with the USA having the highest sales and Ireland having the lowest sales. The bar graph indicates that the company branch in USA leads in sales among the other company branches located in other countries, indicating that it is a key market for the products or services sold. Ireland, at the opposite end, has the smallest bar indicating the company located there has the least number of sales compared to the other countries. This could suggest that Ireland as a whole, has a smaller population which leads to a smaller market size, or there might be lower demand or awareness for the products/services within the people that resides in that region.

The different heights of the bars representing other countries would indicate a variety of sales volumes, offering understanding on the impact of various markets on total sales. For instances, Spain and France have similar number of sales. Countries with higher bars like USA, Spain and France make greater contributions to the company, whereas those with lower bars like Ireland, Phillipines and Belguim make smaller contributions to the company.