ETW2001 Assignment 2

Lee Zhen Xuan 31860532

Section A

Question 1

```
> #Question 1
> olist_orders_dataset = read.csv("olist_orders_dataset.csv")
> olist_order_items_dataset = read.csv("olist_order_items_dataset.csv")
>
> dim(olist_orders_dataset)
[1] 99441     8
> ls(olist_orders_dataset)
[1] "customer_id"                   "order_approved_at"             "order_delivered_carrier_date"  "order_delivered_customer_date"
[5] "order_estimated_delivery_date" "order_id"                      "order_purchase_timestamp"      "order_status"
>
> dim(olist_order_items_dataset)
[1] 112650      7
> ls(olist_order_items_dataset)
[1] "freight_value"       "order_id"            "order_item_id"       "price"               "product_id"
[6] "seller_id"           "shipping_limit_date"
>
> orders_itemsIJ = inner_join(olist_orders_dataset,olist_order_items_dataset, by = "order_id")
> dim(orders_itemsIJ)
[1] 112650     14
> glimpse(orders_itemsIJ)
Rows: 112,650
Columns: 14
$ order_id                      <chr> "e481f51cbdc54678b7cc49136f2d6af7", "53cdb2fc8bc7dce0b6741e2150273451", "47770eb9100c2d0c449…
$ customer_id                   <chr> "9ef432eb6251297304e76186b10a928d", "b0830fb4747a6c6d20dea0b8c802d7ef", "41ce2a54c0b03bf3443…
$ order_status                  <chr> "delivered", "delivered", "delivered", "delivered", "delivered", "delivered", "invoiced", "d…
$ order_purchase_timestamp      <chr> "2017-10-02 10:56:33", "2018-07-24 20:41:37", "2018-08-08 08:38:49", "2017-11-18 19:28:06", …
$ order_approved_at             <chr> "2017-10-02 11:07:15", "2018-07-26 03:24:27", "2018-08-08 08:55:23", "2017-11-18 19:45:59", …
$ order_delivered_carrier_date  <chr> "2017-10-04 19:55:00", "2018-07-26 14:31:00", "2018-08-08 13:50:00", "2017-11-22 13:39:59", …
$ order_delivered_customer_date <chr> "2017-10-10 21:25:13", "2018-08-07 15:27:45", "2018-08-17 18:06:29", "2017-12-02 00:28:42", …
$ order_estimated_delivery_date <chr> "2017-10-18 00:00:00", "2018-08-13 00:00:00", "2018-09-04 00:00:00", "2017-12-15 00:00:00", …
$ order_item_id                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,…
$ product_id                    <chr> "87285b34884572647811a353c7ac498a", "595fac2a385ac33a80bd5114aec74eb8", "aa4383b373c6aca5d87…
$ seller_id                     <chr> "3504c0cb71d7fa48d967e0e4c94d59d9", "289cdb325fb7e7f891c38608bf9e0962", "4869f7a5dfa277a7dca…
$ shipping_limit_date           <chr> "2017-10-06 11:07:15", "2018-07-30 03:24:27", "2018-08-13 08:55:23", "2017-11-23 19:45:59", …
$ price                         <dbl> 29.99, 118.70, 159.90, 45.00, 19.90, 147.90, 49.90, 59.99, 19.90, 149.99, 99.00, 99.00, 98.0…
$ freight_value                 <dbl> 8.72, 22.76, 19.22, 27.20, 8.72, 27.36, 16.05, 15.17, 16.05, 19.77, 30.53, 30.53, 16.13, 18.…
```

An inner join is performed between olist_orders_dataset and olist_order_items_dataset to produce a new resulting data frame named "orders_itemsIJ". The inner join includes observations that match in both olist_orders_dataset and olist_items_dataset and includes all the variables from both the datasets.

The new resulting data frame consists of 112,650 rows and 14 variables. From the merged dataset, we can learn more information for each order item, and we can learn more details about each order item such as shipping details, delivery date, status of the order and more.

Question 2

```
> #Question 2
> olist_order_reviews_dataset = read.csv("olist_order_reviews_dataset.csv")
>
> orders_reviewsLJ = left_join(olist_orders_dataset, olist_order_reviews_dataset, by = "order_id")
> dim(orders_reviewsLJ)
[1] 99992    14
>
> #finding orders with no reviews by orders with no reviews_id
> orders_no_reviews = orders_reviewsLJ %>% filter(is.na(review_id))
>
> paste("There are",nrow(orders_no_reviews) ,"orders with no reviews")
[1] "There are 768 orders with no reviews"
```

A left join is performed with olist_orders_dataset and olist_order_reviews_dataset to produce a new data frame named “orders_reviewsLJ”. The left join operation includes all observations in olist_orders_dataset, regardless of whether they match or not. This means that it includes all orders regardless of if they have no reviews. The new data frame consists of 99992 observations and 14 variables.

We can find orders with no reviews in the resulting data frame by finding the orders with no review ids. There are 768 orders with no reviews.

Question 3

```
> #Question 3
> olist_products_dataset = read.csv("olist_products_dataset.csv")
>
> items_productsRJ = right_join(olist_order_items_dataset, olist_products_dataset, by = "product_id")
> dim(items_productsRJ)
[1] 112650     15
>
> #finding products not sold with products with no order_id
> products_not_sold = items_productsRJ %>% filter(is.na(order_id))
>
> paste("There are",nrow(products_not_sold) ,"products that have not been sold")
[1] "There are 0 products that have not been sold"
> unique(products_not_sold$product_id)
character(0)
```

A right join is performed with olist_order_items_dataset and olist_products_dataset to produce a new data frame named items_productsRJ. The right join operation includes all observations in olist_products_dataset, regardless of whether they match or not. This means that it includes all products regardless of if the products have no order id. The resulting data frame consists of 112650 observations and 15 variables.

We can find products with no orders in the resulting data frame by finding the products with no order id. There are no products that have not been sold.

Question 4

```
> #Question 4
> olist_customers_dataset = read.csv("olist_customers_dataset.csv")
>
> customers_ordersFJ = full_join(olist_customers_dataset, olist_orders_dataset, by = "customer_id")
> dim(customers_ordersFJ)
[1] 99441    12
>
> #finding customers without orders by customers with no order_id
> customers_without_orders = customers_ordersFJ %>% filter(is.na(order_id))
>
> #fidning orders without customers by orders with no customer_id
> orders_without_customers = customers_ordersFJ %>% filter(is.na(customer_id))
>
> paste("There are", nrow(customers_without_orders), "customers without orders")
[1] "There are 0 customers without orders"
> paste("There are", nrow(orders_without_customers), "orders without customers")
[1] "There are 0 orders without customers"
```

A full join is performed between olist_customers_dataset and olist_orders_dataset to combine both the datasets. The full join operation includes all observations from olist_customers_dataset and olist_orders_dataset, therefore every row in both the dataset in the resulting data frame is included. There are 99441 observations and 12 variables in the resulting data frame.

We can find the customers without orders by finding the rows with no order id. There are no customers without orders.

We can find the orders without customers by finding the rows with no customer id. There are no orders without customers.

Question 5

```
> #Question 5
> olist_sellers_dataset = read.csv("olist_sellers_dataset.csv")
>
> customers_sellersSJ = semi_join(olist_sellers_dataset, olist_order_items_dataset, by = "seller_i
d")
>
> summary(olist_sellers_dataset)
  seller_id          seller_zip_code_prefix seller_city        seller_state
 Length:3095        Min.   : 1001          Length:3095        Length:3095
 Class :character   1st Qu.: 7094          Class :character   Class :character
 Mode  :character   Median :14940          Mode  :character   Mode  :character
                    Mean   :32291
                    3rd Qu.:64553
                    Max.   :99730
> summary(customers_sellersSJ)
  seller_id          seller_zip_code_prefix seller_city        seller_state
 Length:3095        Min.   : 1001          Length:3095        Length:3095
 Class :character   1st Qu.: 7094          Class :character   Class :character
 Mode  :character   Median :14940          Mode  :character   Mode  :character
                    Mean   :32291
                    3rd Qu.:64553
                    Max.   :99730
> dim(customers_sellersSJ)
[1] 3095    4
>
> length(unique(olist_sellers_dataset$seller_id))
[1] 3095
> length(unique(customers_sellersSJ$seller_id))
[1] 3095
>
> inactive_sellers = anti_join(olist_sellers_dataset, olist_order_items_dataset)
Joining with `by = join_by(seller_id)`
> paste("There are", nrow(inactive_sellers), "inactive sellers")
[1] "There are 0 inactive sellers"
```

```
> length(unique(customers_sellersSJ$seller_zip_code_prefix))
[1] 2246
> length(unique(customers_sellersSJ$seller_city))
[1] 611
> length(unique(customers_sellersSJ$seller_state))
[1] 23
```

A semi join is performed between olist_sellers_dataset and olist_order_items_dataset to produce the resulting data frame named “customers_sellersSJ”. The semi join operation keeps rows in olist_sellers_dataset that have one or more matches in olist_order_items_dataset and discard other values. This means that it only includes sellers who have made sales. There are 3095 observations and 4 variables in the resulting data frame.

All of the 3095 sellers are active sellers with order items. Based on the output, there are 2246 unique zip code prefixes for active sellers, 611 unique cities where sellers live in, and 23 unique states where sellers live in.

Question 6

```
> #Question 6
> customers_ordersAJ = anti_join(olist_customers_dataset, olist_orders_dataset, by = "customer_id")
> dim(customers_ordersAJ)
[1] 0 5
```

Anti join is performed between olist_customers_dataset and olist_orders_dataset. The anti_join keeps rows in olist_customers_dataset that match zero rows in olist_orders_dataset. As there are 0 observations in the resulting data frame, there are no customers who have never placed an order.

Based on the dataset, we can hypothesize that good marketing strategies have been carried out for each product to reach their sales to customers.

Question 7

```
> #Question 7
> orders_items = inner_join(olist_orders_dataset, olist_order_items_dataset, by= "order_id")
> order_items_products = left_join(orders_items, olist_products_dataset, by = "product_id")
> order_items_products_sellers = left_join(order_items_products, olist_sellers_dataset, by = "seller_id")
>
> order_values = order_items_products_sellers %>% group_by(order_id) %>% summarise(total_price = sum(price))
>
> summary(order_values$total_price)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    0.85    45.90    86.90   137.75   149.90 13440.00
```

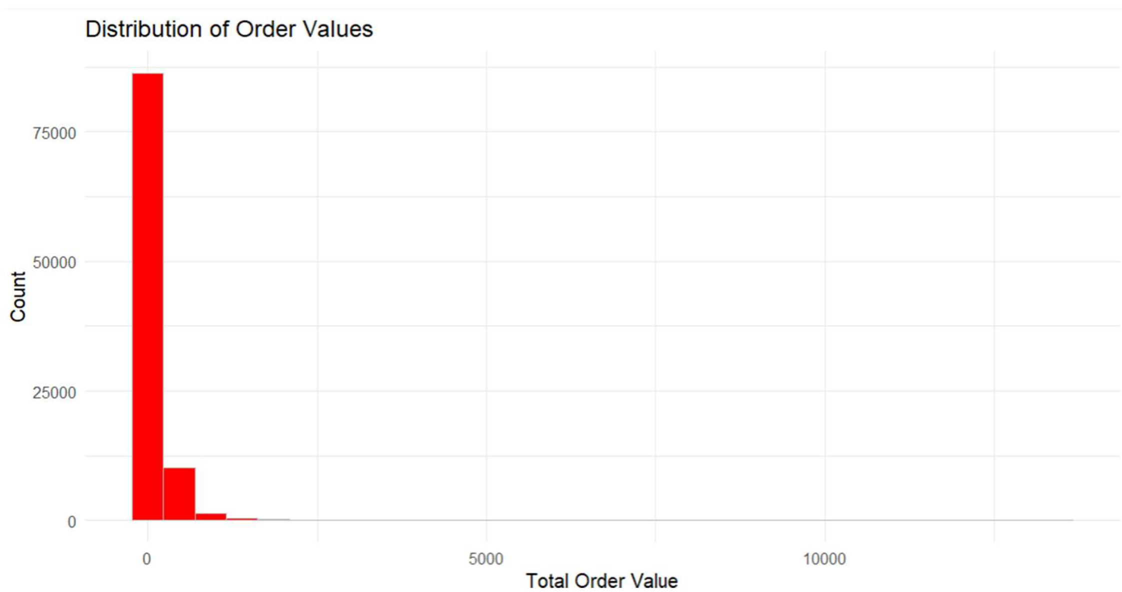
Firstly, inner join is performed between olist_orders_dataset and olist_order_items_dataset to produce a resulting dataframe named “orders_items”. From the merged dataset, we can learn more information for each order item, and we can learn more details about each order item such as shipping details, delivery date, status of the order and more.

Secondly, we perform the left_join between the resulting dataframe “orders_items” and olist_products_dataset to produce a new resulting dataframe “order_items_products” to list all the product details for every order, for example the dimensions of the products, category of the products and more.

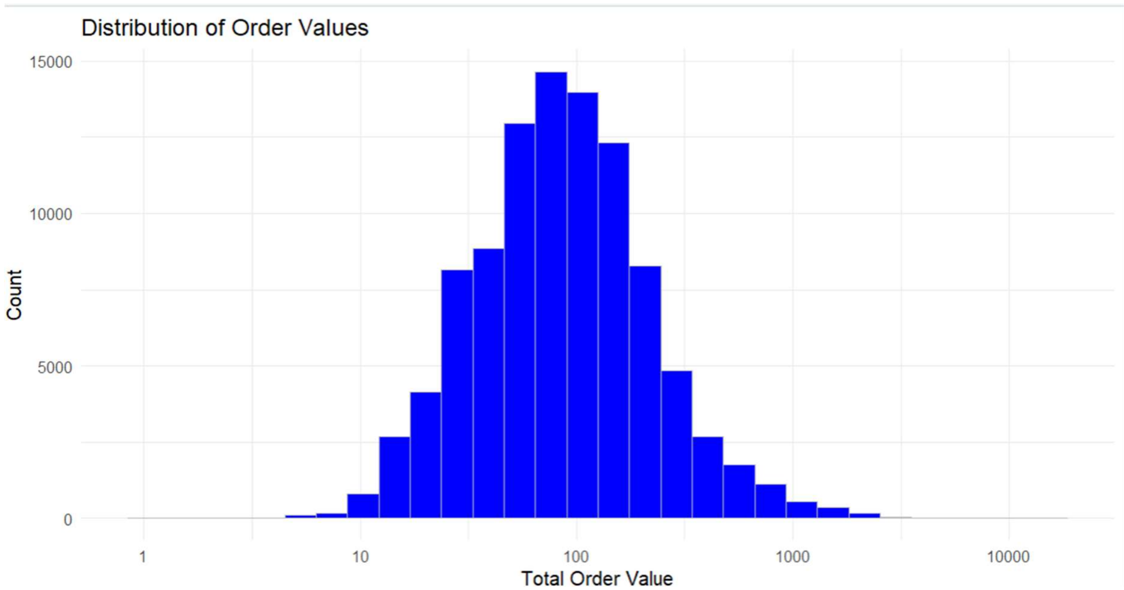
Lastly, we perform the left_join operation between the resulting dataframe “order_items_products” and olist_sellers_dataset to produce a new dataframe called “order_items_products_sellers”. This resulting dataframe includes all the seller details for each item purchased.

```
> order_values = order_items_products_sellers %>% group_by(order_id) %>% summarise(total_price =
sum(price))
>
> summary(order_values$total_price)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    0.85    45.90    86.90   137.75   149.90 13440.00
> View(order_items_products)
> View(order_items_products)
> order_values = order_items_products_sellers %>% group_by(order_id) %>% summarise(total_price =
sum(price))
>
> summary(order_values$total_price)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    0.85    45.90    86.90   137.75   149.90 13440.00
>
> ggplot(order_values, aes(x = total_price)) +
+   geom_histogram(bins = 30, fill = "red", color = "grey") +
+   labs(x = "Total Order Value", y = "Count", title = "Distribution of Order Values") +
+   theme_minimal()
```

We can find the total price for each order by grouping all the order items placed by their order id, then summing the price for every order item. The plot for the order values is shown below:



From the plot above, we can see that the total order value is positively skewed, which means that there are more customers who places their order at a lower price.



A log function is applied to visualise the trend of the total order value and the count of orders.

Section B

Question 1

```
> library(ggplot2)
>
> #Question 1.
> sales_data <- data.frame(
+   Product_Line = c("Classic Cars", "Trains", "Ships", "Trucks and Buses", "Planes", "Motorcycl
es", "Vintage Cars"),
+   Count = c(34.3, 2.7, 8.3, 10.7, 10.8, 11.7, 21.5)
+ )
> print(sum(sales_data$Count))
[1] 100
>
> sales_data$Percentage <- sales_data$Count / sum(sales_data$Count) * 100
> #create a new column to find the percentage for labelling purposes
>
> ggplot(sales_data, aes(x = "", y = Count, fill = Product_Line)) +
+   geom_col() +
+   geom_text(aes(label = paste0(round(Percentage), "%")), position = position_stack(vjust = 0.
5)) +  # adding the percentage numbers inside the pie chart
+   coord_polar(theta = "y") + scale_y_continuous(breaks=cumsum(sales_data$Count) - sales_data$C
ount / 2, labels= sales_data$Product_Line) +
+   ggtitle("Sales Distribution by PRODUCTLINE (Pie Chart)") +theme_minimal() #theme_minimal to
change the background white
```

Firstly, a sample dataframe called sales_data is created to provide the data to plot the pie chart with two columns “Product_Line” and “Count” with their corresponding values. A new column called “Percentage” is calculated by taking the percentage of each of the value of the Product Line.

A pie chart is plotted, by creating a stacked barchart first using the geom_bar() function, then marking it circular with the coord_polar() function. The texts are shown in the pie chart using the geom_text function, and by pasting a “%” percentage value behind every value in the pie chart. The labels are also shown by taking the values from the Product_Line column and is positioned by the outer frame of each of the space in the pie chart.

Overall, the pie chart is used to visualise the categorical data of the product lines, and to help visualise the proportion of the sales of each product line better.

Question 2

```
> #Question 2.
> set.seed(31860532)
> sales_data <- data.frame(
+   Index = 1:100,
+   QUANTITYORDERED = sample(1:200, 100, replace = TRUE),
+   PRICEEACH = runif(100, min = 50, max = 550),
+   SALES = runif(100, min = 2000, max = 12000)
+ )
>
> ggplot(sales_data, aes(x = Index)) +
+   geom_area(aes(y = QUANTITYORDERED, fill = "QUANTITYORDERED"), alpha = 0.6) +
+   geom_area(aes(y = SALES, fill = "SALES"), alpha = 0.6) +
+   geom_area(aes(y = PRICEEACH, fill = "PRICEEACH"), alpha = 0.6) +
+   scale_fill_manual(values = c("QUANTITYORDERED" = "blue", "PRICEEACH" = "orange", "SALES" = "green
4")) +
+   labs(title = "Area Chart of Sales Data",
+        y = "Values",
+        fill = "Variable") +
+   theme_minimal() +
+   labs(title = "Area Chart of Sales Data") + ylim(0,14000)
```

Firstly, a new dataframe “sales_data” is created with 3 columns “Quantity Ordered”, “Price Each” and “Sales”. These values of each column are sampled randomly by with a specific seed “31860532” for reproducibility. A stacked area chart is plotted by using the “geom_area” function for each of the variables from the dataframe, specifying their opacity with “alpha”, and coloured manually with 3 different colours by using the “scale_fill_manual” function. To remove the grey background of the plot, “theme_minimal” is used. The title is also shown using the “labs” function, and the y limit is set from 0 to 14000 for visibility purposes.

Overall, the stacked area chart is a good choice for visualising the trend of the sales over a period, and to highlight and determine the trends of each variable.

Question 3

```
> #Question 3.
> daily_total_sales <- data.frame(
+   Date = seq(as.Date("2022-01-01"), by = "day", length.out = 365),
+   Total_Sales = sample(0:140000, 365, replace = TRUE)
+ )
>
> ggplot(daily_total_sales, aes(x = Date, y = Total_Sales)) +
+   geom_line(color = "blue") +
+   geom_point(color = "blue") +
+   scale_x_date(date_labels = "%Y-%m") +  #use the dates in the format of year-month
+   labs(title = "Daily Total Sales Over Time - Line Chart",
+        x = "Date",
+        y = "Total Sales") +
+   theme_minimal() +
+   theme(plot.title = element_text(hjust = 0.5)) #to center the title
>
```

Firstly, a sample dataframe “daily_total_sales” is created with 2 columns “Date” and “Total_Sales” with the sales count for each date. The Total_Sales count is sampled randomly using the “sample” function ranging from values 0 to 140000 with 365 values.

The line chart is plotted with the dataframe “daily_total_sales” with the geom_line() function, and the lines are coloured blue. The points on the line are also plotted with the geom_point() function with the points coloured blue as well. The dates values for the x-axis are scaled with the format “Year-Month”, and the y-axis values represents the total sales for each day. The function theme_minimal() is used to remove the grey background, and the plot.title = element_text(hjust = 0.5) is used to center the title text.

Overall, the line chart effectively illustrates the total sales of a product over time, facilitating the identification and analysis of trends crucial for forecasting future sales patterns.

Question 4

```
> #Question 4.
>
> distribution_of_deal_sizes = data.frame(
+   Deal_Size = c("Small", "Medium", "Large"),
+   Count = c(1300, 1400, 200)
+ )
>
> ggplot(distribution_of_deal_sizes, aes(x= reorder(Deal_Size, - Count), y = C
ount)) + geom_bar(stat = "identity", fill= "lightblue") +labs( x = "Deal Siz
e", y = "Count", title="Distribution of Deal Sizes - Bar Chart") +
+   theme_minimal() + #to change the background to white
+   theme(plot.title = element_text(hjust = 0.5)) #to center the title
```

A sample dataframe named "distribution_of_deal_sizes" is created with the columns "Deal_Size", and "Count" with the categories of "Deal_Size" and the values of "Count" manually specified. The bar chart is plotted using the geom_bar() function with the "Deal_Size" as its x-axis, and "Count" as the y-axis. The bars in the barchart are coloured with the colour "lightblue". The background of the plot is removed using theme_minimal(), and the plot title is centred with theme(plot.title = element_text(hjust = 0.5)).

Overall, the bar chart provides a clear and simple means to compare the distribution of deal sizes.

Question 5

```
> #Question 5.
>
> #an example dataset
> country  = data.frame(
+   Country = c("Australia", "Austria", "Belgium", "Canada", "Denmark", "Finla
nd", "France", "Germany", "Ireland", "Italy", "Japan", "Norway", "Philippine
s", "Singapore", "Spain", "Sweden", "Switzerland", "UK", "USA"),
+   number_of_sales = c(200, 75, 50,100,90,120,370,95,25,130,125,150,200,150,2
00,100,120,130,120)
+ )
>
> ggplot(country, aes(x = Country, y = number_of_sales)) + geom_bar(stat = "id
entity", fill = "steelblue") +
+   labs(x = "Country", y = "Number of Sales", title = "Number of Sales by Cou
ntry") +
+   theme_minimal() + #to change the background to white
+   theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1),plot.titl
e = element_text(hjust = 0.5)) #to make the x values rotate by 45 degrees to v
isualise each value more clearly
```

A sample dataset “country” is created with the columns “Country” and “number_of_sales” as the total number of sales.

The barchart is plotted with “Country” as the x-axis, and “number_of_sales” as the y-axis using the function geom_bar, coloured with “steelblue”. The grey background is removed using the function theme_minimal(). The values on the x-axis are rotated to 45 degrees to visualize each value clearly, and it’s positioned to its corresponding position under the bars. The plot title is centred with plot.title = element_text(hjust = 0.5)

The bar chart serves as a tool to compare sales numbers across different countries, aiding in the identification of countries with strong sales performance.