Author : Eunice Lee Wen Jing
Date : 22/4/2024

SECTION A Data joining

Q1

Code:

```
##Q1
#Perform inner join order & order items
order = read.csv("olist_orders_dataset.csv")
order_items = read.csv("olist_order_items_dataset.csv")

innerJoin = inner_join(order,order_items,by = "order_id")
head(innerJoin)

glimpse(order)
glimpse(order_items)
glimpse(innerJoin)
```

Output:

```
> glimpse(order)
Rows: 99,441
Columns: 8
$ order_id                      <chr> "e481f51cbdc54678b7cc49136f2d6af7", "53cdb2f…
$ customer_id                   <chr> "9ef432eb6251297304e76186b10a928d", "b0830fb…
$ order_status                  <chr> "delivered", "delivered", "delivered", "deli…
$ order_purchase_timestamp      <chr> "2017-10-02 10:56:33", "2018-07-24 20:41:37"…
$ order_approved_at             <chr> "2017-10-02 11:07:15", "2018-07-26 03:24:27"…
$ order_delivered_carrier_date  <chr> "2017-10-04 19:55:00", "2018-07-26 14:31:00"…
$ order_delivered_customer_date <chr> "2017-10-10 21:25:13", "2018-08-07 15:27:45"…
$ order_estimated_delivery_date <chr> "2017-10-18 00:00:00", "2018-08-13 00:00:00"…
> glimpse(order_items)
Rows: 112,650
Columns: 7
$ order_id            <chr> "00010242fe8c5a6d1ba2dd792cb16214", "00018f77f2f0320c5…
$ order_item_id       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, …
$ product_id          <chr> "4244733e06e7ecb4970a6e2683c13e61", "e5f2d52b802189ee6…
$ seller_id           <chr> "48436dade18ac8b2bce089ec2a041202", "dd7ddc04e1b6c2c61…
$ shipping_limit_date <chr> "2017-09-19 09:45:35", "2017-05-03 11:05:13", "2018-01…
$ price               <dbl> 58.90, 239.90, 199.00, 12.99, 199.90, 21.90, 19.90, 81…
$ freight_value       <dbl> 13.29, 19.93, 17.87, 12.79, 18.14, 12.69, 11.85, 70.75…

> glimpse(innerJoin)
Rows: 112,650
Columns: 14
$ order_id                      <chr> "e481f51cbdc54678b7cc49136f2d6af7", "53cdb2f…
$ customer_id                   <chr> "9ef432eb6251297304e76186b10a928d", "b0830fb…
$ order_status                  <chr> "delivered", "delivered", "delivered", "deli…
$ order_purchase_timestamp      <chr> "2017-10-02 10:56:33", "2018-07-24 20:41:37"…
$ order_approved_at             <chr> "2017-10-02 11:07:15", "2018-07-26 03:24:27"…
$ order_delivered_carrier_date  <chr> "2017-10-04 19:55:00", "2018-07-26 14:31:00"…
$ order_delivered_customer_date <chr> "2017-10-10 21:25:13", "2018-08-07 15:27:45"…
$ order_estimated_delivery_date <chr> "2017-10-18 00:00:00", "2018-08-13 00:00:00"…
$ order_item_id                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,…
$ product_id                    <chr> "87285b34884572647811a353c7ac498a", "595fac2…
$ seller_id                     <chr> "3504c0cb71d7fa48d967e0e4c94d59d9", "289cdb3…
$ shipping_limit_date           <chr> "2017-10-06 11:07:15", "2018-07-30 03:24:27"…
$ price                         <dbl> 29.99, 118.70, 159.90, 45.00, 19.90, 147.90,…
$ freight_value                 <dbl> 8.72, 22.76, 19.22, 27.20, 8.72, 27.36, 16.0…
```

Explanation:

Initially, in the olist_orders_dataset, there were 99441 observations and 8 variables, while in the olist_order_items dataset, there were 112650 observations and 7 variables. After performing an inner join for both datasets, the merged dataset contained 112650 observations and 14 variables. The inner join was conducted using the key column "order_id". The results of the merged data have proven that every order in the "order" dataset has one corresponding entry in the "order_items" dataset. This is because an inner join only returns the rows with matching values in both datasets based on the key column "order_id", and the merged dataset returns the same number of observations as the olist_order_items_dataset, 112650 rows.

Q2

Code:

```
##Q2
#Perform left join orders & order reviews
review = read.csv("olist_order_reviews_dataset.csv")

leftJoin = left_join(order, review, by = "order_id")
head(leftJoin)

glimpse(order)
glimpse(review)
glimpse(leftJoin)

# Order with reviews
With_reviews <- leftJoin %>% filter(!is.na(review_id)) %>% nrow()
With_reviews # 99224

# Check how many orders do not have corresponding reviews
without_reviews <- leftJoin %>%
  filter(is.na(review_id)) %>%  # Filter rows where review_id is NA
  nrow() #Show no of rows without reviews

# Display the result
cat("Number of orders without corresponding reviews:", without_reviews, "\n") #768
```

Output:

```
> glimpse(review)
Rows: 99,224
Columns: 7
$ review_id               <chr> "7bc2406110b926393aa56f80a40eba40", "80e641a11e56f04c…
$ order_id                <chr> "73fc7af87114b39712e6da79b0a377eb", "a548910a1c614779…
$ review_score            <int> 4, 5, 5, 5, 5, 1, 5, 5, 5, 4, 5, 5, 4, 4, 3, 5, 2, 5,…
$ review_comment_title    <chr> "", "", "", "", "", "", "", "", "", "recomendo", "", …
$ review_comment_message  <chr> "", "", "", "Recebi bem antes do prazo estipulado.", …
$ review_creation_date    <chr> "2018-01-18 00:00:00", "2018-03-10 00:00:00", "2018-0…
$ review_answer_timestamp <chr> "2018-01-18 21:46:59", "2018-03-11 03:05:13", "2018-0…
> glimpse(leftJoin)
Rows: 99,992
Columns: 14
$ order_id                      <chr> "e481f51cbdc54678b7cc49136f2d6af7", "53cdb2fc8b…
$ customer_id                   <chr> "9ef432eb6251297304e76186b10a928d", "b0830fb474…
$ order_status                  <chr> "delivered", "delivered", "delivered", "deliver…
$ order_purchase_timestamp      <chr> "2017-10-02 10:56:33", "2018-07-24 20:41:37", "…
$ order_approved_at             <chr> "2017-10-02 11:07:15", "2018-07-26 03:24:27", "…
$ order_delivered_carrier_date  <chr> "2017-10-04 19:55:00", "2018-07-26 14:31:00", "…
$ order_delivered_customer_date <chr> "2017-10-10 21:25:13", "2018-08-07 15:27:45", "…
$ order_estimated_delivery_date <chr> "2017-10-18 00:00:00", "2018-08-13 00:00:00", "…
$ review_id                     <chr> "a54f0611adc9ed256b57ede6b6eb5114", "8d52660420…
$ review_score                  <int> 4, 4, 5, 5, 5, 4, 2, 5, 1, 5, 1, 4, 5, 5, 4, 5,…
$ review_comment_title          <chr> "", "Muito boa a loja", "", "", "", "", "", "",…
$ review_comment_message        <chr> "Não testei o produto ainda, mas ele veio corre…
$ review_creation_date          <chr> "2017-10-11 00:00:00", "2018-08-08 00:00:00", "…
$ review_answer_timestamp       <chr> "2017-10-12 03:43:48", "2018-08-08 18:37:50", "…

> # Display the result
> cat("Number of orders without corresponding reviews:", num_without_reviews, "\n")
Number of orders without corresponding reviews: 768
```

Explanation:

We observed that in the olist_orders_dataset, there were 99441 observations and 8 variables, while in the olist_reviews_items dataset, there were 99224 observations and 7 variables. After performing a left join for both datasets, the merged dataset contained 99992 observations and 14 variables. The left join was executed using the key column "order_id". The merged dataset was used to identify orders without reviews. A total of 768 orders were found without reviews.

Q3

Code:

```r
##Q3
#Perform right join items & products
product = read.csv("olist_products_dataset.csv") #32951obv ,9var

rightJoin = right_join(order_items,product,by="product_id")
glimpse(rightJoin) #112650obv,15 var

# Identify products that have not been sold yet
unsold <- rightJoin %>%
  filter(is.na(order_id))%>%  # Filter rows where order_id is NA
  nrow()

glimpse(unsold) #All product is sold

cat("The number of unsold product in product dataset:",unsold,"\n")
```

Output:

```
> glimpse(rightJoin)
Rows: 112,650
Columns: 15
$ order_id                   <chr> "00010242fe8c5a6d1ba2dd792cb16214", "00018f77f2f03…
$ order_item_id              <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,…
$ product_id                 <chr> "4244733e06e7ecb4970a6e2683c13e61", "e5f2d52b80218…
$ seller_id                  <chr> "48436dade18ac8b2bce089ec2a041202", "dd7ddc04e1b6c…
$ shipping_limit_date        <chr> "2017-09-19 09:45:35", "2017-05-03 11:05:13", "201…
$ price                      <dbl> 58.90, 239.90, 199.00, 12.99, 199.90, 21.90, 19.90…
$ freight_value              <dbl> 13.29, 19.93, 17.87, 12.79, 18.14, 12.69, 11.85, 7…
$ product_category_name      <chr> "cool_stuff", "pet_shop", "moveis_decoracao", "per…
$ product_name_lenght        <int> 58, 56, 59, 42, 59, 36, 52, 39, 59, 52, 57, 35, 54…
$ product_description_lenght <int> 598, 239, 695, 480, 409, 558, 815, 1310, 493, 1192…
$ product_photos_qty         <int> 4, 2, 2, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 2, 2, 4, 2,…
$ product_weight_g           <int> 650, 30000, 3050, 200, 3750, 450, 200, 13805, 2000…
$ product_length_cm          <int> 28, 50, 33, 16, 35, 24, 27, 35, 30, 29, 35, 16, 50…
$ product_height_cm          <int> 9, 30, 13, 10, 40, 8, 5, 75, 12, 3, 25, 16, 10, 22…
$ product_width_cm           <int> 14, 40, 33, 15, 30, 15, 20, 45, 16, 21, 20, 11, 40…
```

```
> glimpse(unsold) #All product is sold
 int 0
> cat("The number of unsold product in product dataset:",unsold,"\n")
The number of unsold product in product dataset: 0
```

We observed that in the olist_order_items_dataset, there were 112650 observations and 7 variables, while in the olist_products_dataset, there were 32951 observations and 9 variables. After performing a right join for both datasets, the merged dataset contained 112650 observations and 15 variables. In a right join, all data from the right dataset (in this case, the olist_product_dataset) are retained, with matching records from the olist_order_items_dataset included where available. Null values are filled in for the variables from the olist_order_items_dataset in cases where there are no matching records. This join type ensures that all records from the olist_product_dataset are included, while also incorporating relevant information from the olist_order_items_dataset. The merged dataset is used to retrieve the rows where order_id is NA to identify products that have not been sold. It shows that there are 0 rows left after filtering, indicating that all products listed in the product dataset have been sold. In conclusion, there are 0 products in the dataset that have not been sold yet.

Q4

Code:

```
##Q4
#Perform full outer join customers & order
customer = read.csv("olist_customers_dataset.csv")

fullJoin = full_join(customer, order, by = "customer_id")
glimpse(fullJoin)

# Any customers without orders or orders without customer details
without_orders <- fullJoin %>%
  filter(is.na(order_id)) %>%  # Filter rows where order_id is NA
  nrow()

without_customers <- fullJoin %>%
  filter(is.na(customer_id))%>%  # Filter rows where customer_id is NA
  nrow()

cat("Number of customers without orders:", without_orders, "\n") #0
cat("Number of orders without customer details:", without_customers, "\n") #0
```

Output:

```
> glimpse(fullJoin)
Rows: 99,441
Columns: 12
$ customer_id                   <chr> "06b8999e2fba1a1fbc88172c00ba8bc7", "18955e83d3…
$ customer_unique_id            <chr> "861eff4711a542e4b93843c6dd7febb0", "290c77bc52…
$ customer_zip_code_prefix      <int> 14409, 9790, 1151, 8775, 13056, 89254, 4534, 35…
$ customer_city                 <chr> "franca", "sao bernardo do campo", "sao paulo",…
$ customer_state                <chr> "SP", "SP", "SP", "SP", "SP", "SC", "SP", "MG",…
$ order_id                      <chr> "00e7ee1b050b8499577073aeb2a297a1", "29150127e6…
$ order_status                  <chr> "delivered", "delivered", "delivered", "deliver…
$ order_purchase_timestamp      <chr> "2017-05-16 15:05:35", "2018-01-12 20:48:24", "…
$ order_approved_at             <chr> "2017-05-16 15:22:12", "2018-01-12 20:58:32", "…
$ order_delivered_carrier_date  <chr> "2017-05-23 10:47:57", "2018-01-15 17:14:59", "…
$ order_delivered_customer_date <chr> "2017-05-25 10:35:35", "2018-01-29 12:41:19", "…
$ order_estimated_delivery_date <chr> "2017-06-05 00:00:00", "2018-02-06 00:00:00", "…
> cat("Number of customers without orders:", without_orders, "\n") #0
Number of customers without orders: 0
> cat("Number of orders without customer details:", without_customers, "\n") #0
Number of orders without customer details: 0
```

Explanation:

After performing a full outer join for both datasets, the merged dataset contained 99441 observations and 12 variables. A full outer join ensures that all records from both datasets are retained, aligning customer details with corresponding orders where possible. This comprehensive merging approach facilitated the investigation into the presence of customers without orders and orders without customer details. As a result, there are no customers without orders and no orders without customer details, as indicated by 0 counts for both cases. This implies that all customers have placed orders, and all orders have corresponding customer information.

Q5

Code:

```
##Q5
#Perform semiJoin sellers & items
seller = read.csv("olist_sellers_dataset.csv")

#details of customers who have placed orders
semiJoin = semi_join(seller, order_items, by = "seller_id")
glimpse(semiJoin)

# Characteristics of active sellers VS the complete seller list
cat("Number of active sellers:", nrow(semiJoin), "\n") #3095
cat("Total number of sellers in the complete list:", nrow(seller), "\n") #3095
```

Output:

```
> glimpse(semiJoin)
Rows: 3,095
Columns: 4
$ seller_id              <chr> "3442f8959a84dea7ee197c632cb2df15", "d1b65fc7debc3361e…
$ seller_zip_code_prefix <int> 13023, 13844, 20031, 4195, 12914, 20920, 55325, 16304,…
$ seller_city            <chr> "campinas", "mogi guacu", "rio de janeiro", "sao paulo…
$ seller_state           <chr> "SP", "SP", "RJ", "SP", "SP", "RJ", "PE", "SP", "SP", …
```

```
> # Characteristics of active sellers VS the complete seller list
> cat("Number of active sellers:", nrow(semiJoin), "\n") #3095
Number of active sellers: 3095
> cat("Total number of sellers in the complete list:", nrow(seller), "\n") #3095
Total number of sellers in the complete list: 3095
```

Explanation:
I used a semi join to filter the olist_sellers_dataset for sellers who have made sales, based on the olist_order_items_dataset. After merging, the merged dataset contains information on active sellers who have recorded sales. From the merged dataset, we know that there are 3095 active sellers. The total number of sellers in the complete seller list is also 3095. Thus, this indicates that all sellers in the dataset have made sales.

Q6
Code:

```
##Q6
#Perform antijoin customer & order
customer = read.csv("olist_customers_dataset.csv")

antiJoin = anti_join(customer,order, by = "customer_id")
glimpse(antiJoin) #0 rows
```

Output:

```
> glimpse(antiJoin) #0 rows
Rows: 0
Columns: 5
$ customer_id              <chr>
$ customer_unique_id       <chr>
$ customer_zip_code_prefix <int>
$ customer_city            <chr>
$ customer_state           <chr>
```

Explanation:
In the context of the olist_customer_dataset and olist_order_dataset, the anti join help identify customers who have never placed an order. This is achieved by comparing the customer IDs in the olist_customer_dataset with those in the olist_order_dataset and selecting only the records from the olist_customer_dataset that do not have a matching customer ID in the olist_order_dataset. Since there are zero entries in the combined dataset, it is evident that every client on the list has made purchases. Therefore, as none of the consumers have never placed an order, no profile can be summarised for those who have not. The hypothesis suggests that the dataset only captures active customers, potentially skewing the representation of the customer base towards those who have already made a purchase.

Q7

Code:

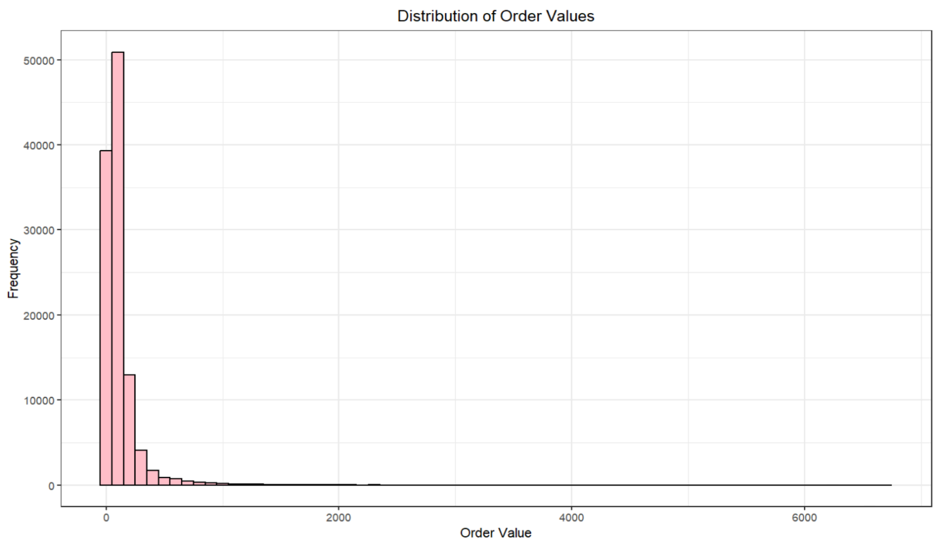
```
merged_data <- order %>%
  inner_join(order_items, by = "order_id") %>%
  inner_join(product, by = "product_id") %>%
  inner_join(seller, by = "seller_id")
glimpse(merged_data)


# Sellers to product
# Number of products sold by each seller
products_per_seller <- merged_data %>%
  group_by(seller_id) %>%
  mutate(total_products_sold = n_distinct(product_id))%>%
  select(seller_id,total_products_sold)

glimpse(products_per_seller)
products_per_seller
summary(products_per_seller)

# Visualize the distribution of order values
# Histogram
ggplot(merged_data, aes(x = price)) +
  geom_histogram(binwidth = 100, fill = "pink", color = "black") +
  labs(title = "Distribution of Order Values", x = "Order Value", y = "Frequency")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))  # Center title
```

Output:

```
> glimpse(merged_data)
Rows: 112,650
Columns: 25
$ order_id                      <chr> "e481f51cbdc54678b7cc49136f2d6af7", "53cdb2fc8b…
$ customer_id                   <chr> "9ef432eb6251297304e76186b10a928d", "b0830fb474…
$ order_status                  <chr> "delivered", "delivered", "delivered", "deliver…
$ order_purchase_timestamp      <chr> "2017-10-02 10:56:33", "2018-07-24 20:41:37", "…
$ order_approved_at             <chr> "2017-10-02 11:07:15", "2018-07-26 03:24:27", "…
$ order_delivered_carrier_date  <chr> "2017-10-04 19:55:00", "2018-07-26 14:31:00", "…
$ order_delivered_customer_date <chr> "2017-10-10 21:25:13", "2018-08-07 15:27:45", "…
$ order_estimated_delivery_date <chr> "2017-10-18 00:00:00", "2018-08-13 00:00:00", "…
$ order_item_id                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,…
$ product_id                    <chr> "87285b34884572647811a353c7ac498a", "595fac2a38…
$ seller_id                     <chr> "3504c0cb71d7fa48d967e0e4c94d59d9", "289cdb325f…
$ shipping_limit_date           <chr> "2017-10-06 11:07:15", "2018-07-30 03:24:27", "…
$ price                         <dbl> 29.99, 118.70, 159.90, 45.00, 19.90, 147.90, 49…
$ freight_value                 <dbl> 8.72, 22.76, 19.22, 27.20, 8.72, 27.36, 16.05, …
$ product_category_name         <chr> "utilidades_domesticas", "perfumaria", "automot…
$ product_name_lenght           <int> 40, 29, 46, 59, 38, 49, NA, 59, 41, 45, 36, 36,…
$ product_description_lenght    <int> 268, 178, 232, 468, 316, 608, NA, 956, 432, 527…
$ product_photos_qty            <int> 4, 1, 1, 3, 4, 1, NA, 1, 2, 1, 1, 1, 1, 1, 1, 2…
$ product_weight_g              <int> 500, 400, 420, 450, 250, 7150, 600, 50, 300, 97…
$ product_length_cm             <int> 19, 19, 24, 30, 51, 65, 35, 16, 35, 42, 42, 42,…
$ product_height_cm             <int> 8, 13, 19, 10, 15, 10, 35, 16, 35, 41, 12, 12, …
$ product_width_cm              <int> 13, 19, 21, 20, 15, 65, 15, 17, 15, 42, 39, 39,…
$ seller_zip_code_prefix        <int> 9350, 31570, 14840, 31842, 8752, 7112, 5455, 12…
$ seller_city                   <chr> "maua", "belo horizonte", "guariba", "belo hori…
$ seller_state                  <chr> "SP", "SP", "SP", "MG", "SP", "SP", "SP", "SP",…

> glimpse(products_per_seller)
Rows: 112,650
Columns: 2
Groups: seller_id [3,095]
$ seller_id           <chr> "3504c0cb71d7fa48d967e0e4c94d59d9", "289cdb325fb7e…
$ total_products_sold <int> 14, 25, 95, 20, 15, 32, 37, 37, 6, 198, 11, 11, 56…
> products_per_seller
# A tibble: 112,650 × 2
# Groups:   seller_id [3,095]
   seller_id                        total_products_sold
   <chr>                                          <int>
 1 3504c0cb71d7fa48d967e0e4c94d59d9                  14
 2 289cdb325fb7e7f891c38608bf9e0962                  25
 3 4869f7a5dfa277a7dca6462dcf3b52b2                  95
 4 66922902710d126a0e7d26b0e3805106                  20
 5 2c9e548be18521d1c43cde1c582c6de8                  15
 6 8581055ce74af1daba164fdbd55a40de                  32
 7 dc8798cbf453b7e0f98745e396cc5616                  37
 8 16090f2ca825584b5a147ab24aa30c86                  37
 9 63b9ae557efed31d1f7687917d248a8d                   6
10 7c67e1448b00f6e969d365cea6b010ab                 198
# ℹ 112,640 more rows
# ℹ Use `print(n = ...)` to see more rows

> summary(products_per_seller)
  seller_id         total_products_sold
 Length:112650      Min.   :  1.00    
 Class :character   1st Qu.: 14.00    
 Mode  :character   Median : 37.00    
                    Mean   : 70.93    
                    3rd Qu.:104.00    
                    Max.   :399.00    
> 
```

```
> highest_products_sold <- products_per_seller %>%
+   filter(total_products_sold == max(products_per_seller$total_products_sold))
> highest_products_sold
# A tibble: 1,987 × 2
# Groups:   seller_id [1]
   seller_id                        total_products_sold
   <chr>                                          <int>
 1 4a3ca9315b744ce9f8e9374361493884                 399
 2 4a3ca9315b744ce9f8e9374361493884                 399
 3 4a3ca9315b744ce9f8e9374361493884                 399
 4 4a3ca9315b744ce9f8e9374361493884                 399
 5 4a3ca9315b744ce9f8e9374361493884                 399
 6 4a3ca9315b744ce9f8e9374361493884                 399
 7 4a3ca9315b744ce9f8e9374361493884                 399
 8 4a3ca9315b744ce9f8e9374361493884                 399
 9 4a3ca9315b744ce9f8e9374361493884                 399
10 4a3ca9315b744ce9f8e9374361493884                 399
# ℹ 1,977 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```
> lowest_products_sold <- products_per_seller %>%
+   filter(total_products_sold == min(products_per_seller$total_products_sold))
> lowest_products_sold
# A tibble: 1,438 × 2
# Groups:   seller_id [746]
   seller_id                        total_products_sold
   <chr>                                          <int>
 1 c4af86330efa7a2620772227d2d670c9                   1
 2 28405831a29823802aa22c084cfd0649                   1
 3 57df9869a600bd6b7c405f2a862eccfb                   1
 4 a56a8043ebf66e42119618fb8cf232c6                   1
 5 4be2e7f96b4fd749d52dff41f80e39dd                   1
 6 9bade61a92bed55a25d2b67b9f4ed739                   1
 7 9bade61a92bed55a25d2b67b9f4ed739                   1
 8 0015a82c2db000af6aaaf3ae2ecb0532                   1
 9 3a3c180dd702a725bd0ba4117689239e                   1
10 c87abc38c8ed3240861729e1aeadf221                   1
# ℹ 1,428 more rows
# ℹ Use `print(n = ...)` to see more rows
```



Explanation:

I chose to use an inner join to merge all four datasets based on their respective key columns (order_id, product_id, seller_id) because an inner join only returns the dataset with matching values, ensuring that the merged dataset retains complete and consistent information. This approach helps maintain data integrity by excluding any records without corresponding entries in all datasets.

Once the datasets are merged, the focus shifts to understanding the flow from sellers to products. To achieve this, I first grouped the merged dataset by seller_id and then counted the distinct product_ids associated with each seller, assigning the results to the products_per_seller dataframe. This dataframe contains 112650 rows and 2 columns (seller_id, total_product_sold).

Next, I used the summary function to gain insights into the sales performance of individual sellers. The summary revealed that the minimum number of products sold by a seller is 1, the maximum is 399, and the average is 71. Furthermore, I found out that 1987 sellers have sold the maximum of 399 products, while 1438 sellers have sold the minimum of 1 product.

I created a histogram to visualise the distribution of order values. The histogram revealed a positively skewed distribution, indicated by a long tail on the right side. The highest frequency of orders falls above 50000, suggesting that a significant number of orders have relatively high values.

SECTION B Data Visualisation

Q1

Code:

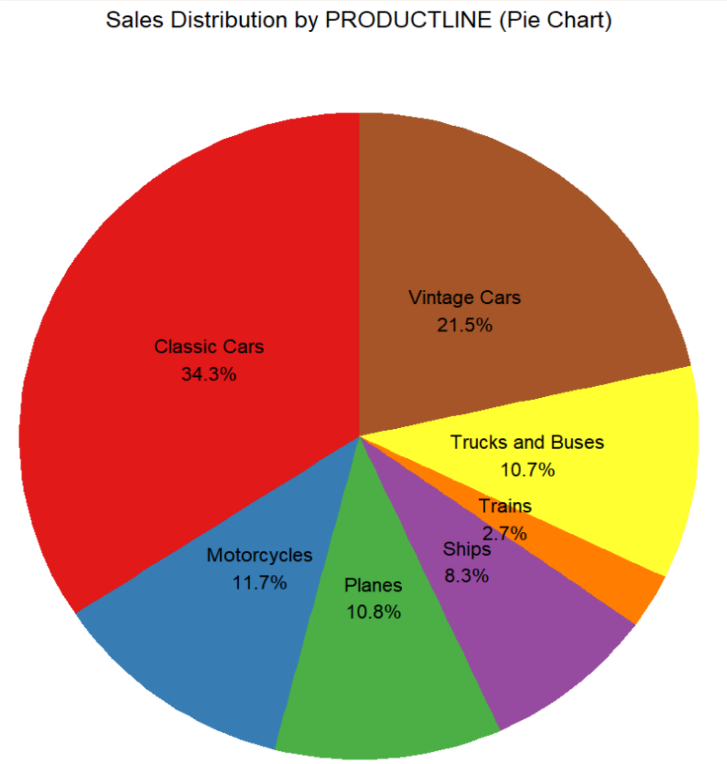
```r
#Generate Fig3 Pie Chart
colnames(sales_data)
str(sales_data)

table_product <- as.data.frame(table(sales_data$PRODUCTLINE))
colnames(table_product) <- c("ProductLine", "Count")
table_product

# Plotting the pie chart
ggplot(table_product, aes(x = "", y = Count, fill = ProductLine)) +
  geom_bar(stat = "identity") +
  # Converts the Cartesian coordinates to polar coordinates, turn bar plot into a pi
  # start = 0 specifies that the first slice should start at the top.
  coord_polar("y", start = 0)+

  labs(title = "Sales Distribution by PRODUCTLINE (Pie Chart)") +
  scale_fill_brewer(palette = "Set1") +
  #Calculate the percentage, round percentage of the count to 1 digit
  geom_text(aes(label = paste0(ProductLine, "\n", round(Count / sum(Count) * 100,1),
            position = position_stack(vjust = 0.5)) +
  theme_void()+
  #Remove legend, labels in the pie chart already
  theme(legend.position = "none",plot.title = element_text(hjust = 0.5))
```

Output:

The column names of the sales data dataset.

```
> colnames(sales_data)
 [1] "ORDERNUMBER"      "QUANTITYORDERED"  "PRICEEACH"        "ORDERLINENUMBER"
 [5] "SALES"            "ORDERDATE"        "STATUS"           "QTR_ID"
 [9] "MONTH_ID"         "YEAR_ID"          "PRODUCTLINE"      "MSRP"
[13] "PRODUCTCODE"      "CUSTOMERNAME"     "PHONE"            "ADDRESSLINE1"
[17] "ADDRESSLINE2"     "CITY"             "STATE"            "POSTALCODE"
[21] "COUNTRY"          "TERRITORY"        "CONTACTLASTNAME"  "CONTACTFIRSTNAME"
[25] "DEALSIZE"
```

Fig 3 Pie Chart



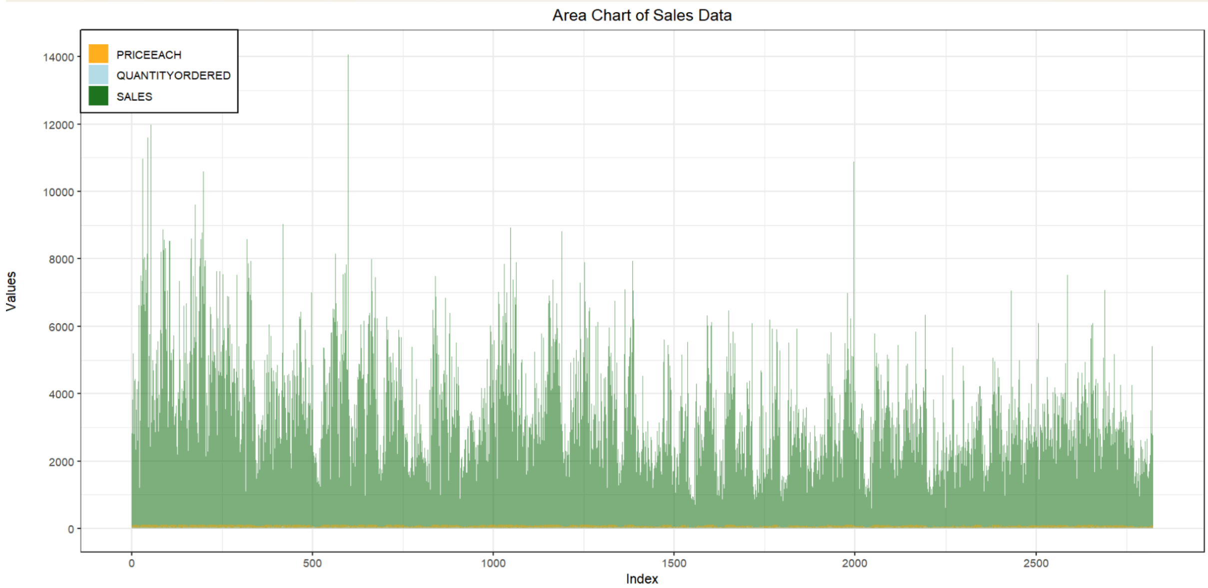
Q2 :

Code:

```r
##Q2
#Generate Fig10 Area Chart
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 = "PRICEEACH"), alpha = 0.5) +
  geom_area(aes(y = QUANTITYORDERED, fill = "QUANTITYORDERED"), alpha = 0.5) +

  labs(title = "Area Chart of Sales Data",
       x = "Index", y = "Values") +
  theme_bw()+

  scale_fill_manual(values = c("QUANTITYORDERED" = "lightblue", "PRICEEACH" = "orang
  theme(
    legend.position = c(0, 1),  # Place legend at top left
    legend.justification = c(0, 1),  # Anchor legend to top left
    legend.title = element_blank(),  # Remove legend title
    plot.title = element_text(hjust = 0.5),  # Center title
    legend.background = element_rect(fill = "white", color = "black", size = 0.5),
    axis.title.y = element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),  # Ac
  ) +
  scale_y_continuous(breaks = seq(0, 14000, by = 2000)) +  # Adjust y-axis ticks
  scale_x_continuous(breaks = seq(0, 2500, by = 500))  # Adjust x-axis ticks
```

Output:

Fig 10 Area Chart



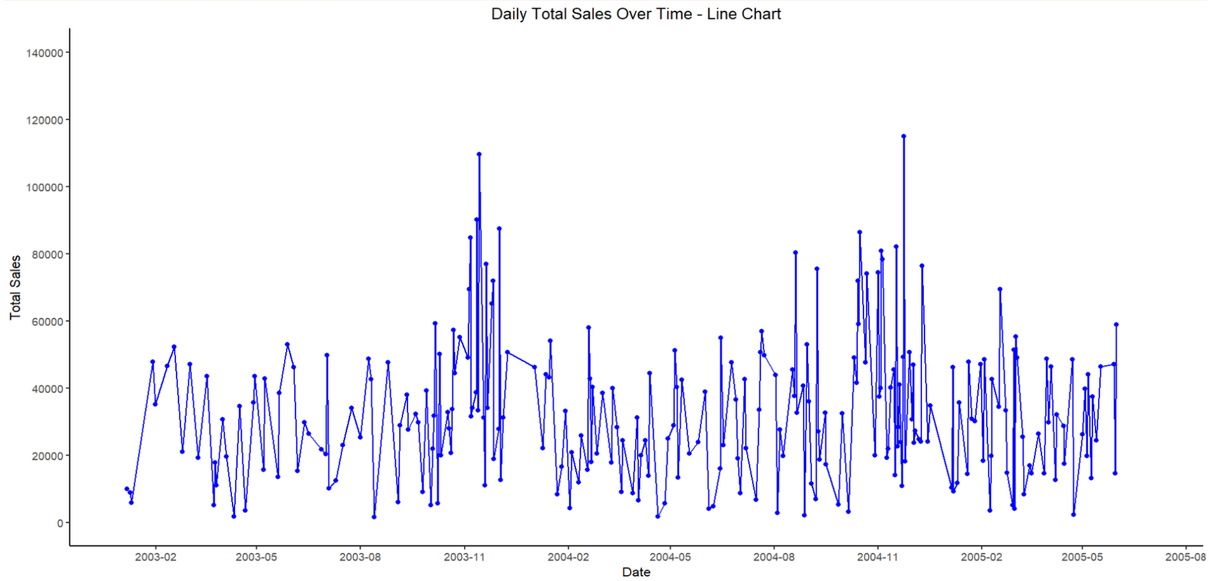
Q3 :

Code:

```r
#Generate Fig17 Bar Chart
library(lubridate)
# Convert to Date format "YYYY-MM-DD"
sales_data <- sales_data %>%
  mutate(ORDERDATE = as.Date(ORDERDATE, format = "%m/%d/%Y %H:%M"))
# Generate Total_Sales column
sales_data <- sales_data %>%
  group_by(ORDERDATE) %>%
  mutate(Total_Sales = sum(QUANTITYORDERED * PRICEEACH)) |
# Extract columns "ORDERDATE" and "Total_Sales"
sales_date <- sales_data %>%select(ORDERDATE, Total_Sales) %>% arrange(ORDERDATE)
sales_date

# Plot line chart
ggplot(sales_date, aes(x = ORDERDATE, y = Total_Sales)) +
  geom_line(color = "blue") +
  geom_point(color = "blue") +
  labs(title = "Daily Total Sales Over Time - Line Chart",
       x = "Date",
       y = "Total Sales") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5)) +  # Center title
  scale_x_date(date_breaks = "3 months", date_labels = "%Y-%m",
               limits = as.Date(c("2003-01-01", "2005-07-01"))) +
  scale_y_continuous(limits = c(0, 140000), breaks = seq(0, 140000, by = 20000))
```

Output:

Fig 16 Line Chart



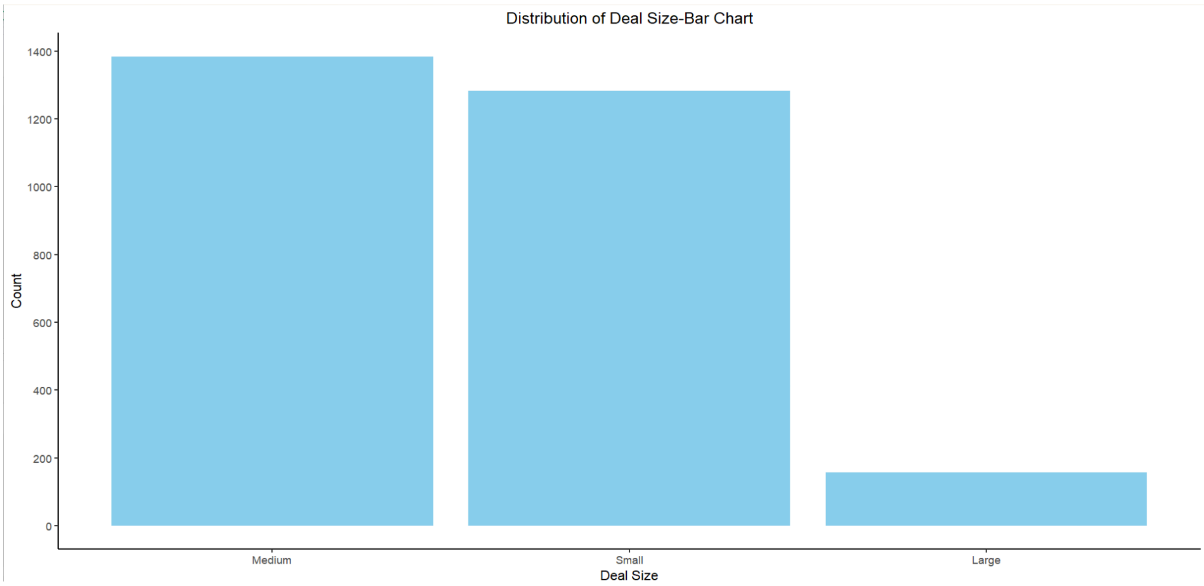
Q4 :

Code:

```
#Q4
#Count the categorical variables in DEALSIZE
table_dealsize <- as.data.frame(table(sales_data$DEALSIZE))
colnames(table_dealsize) <- c("DealSize", "Count")
table_dealsize <- table_dealsize %>% arrange(desc(DealSize))

#Plot Bar Chart
ggplot(table_dealsize, aes(x = reorder(DealSize, -Count), y = Count)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Distribution of Deal Size-Bar Chart",
       x = "Deal Size",
       y = "Count") +
  scale_y_continuous(breaks = seq(0, 1400, by = 200))+
  theme_classic()+
  theme(plot.title = element_text(hjust = 0.5))    # Center title
```

Output:

Fig 17 Bar Chart



Distribution of Deal Size-Bar Chart
Count
1400
1200
1000
800
600
400
200
0
Medium
Small
Large
Deal Size

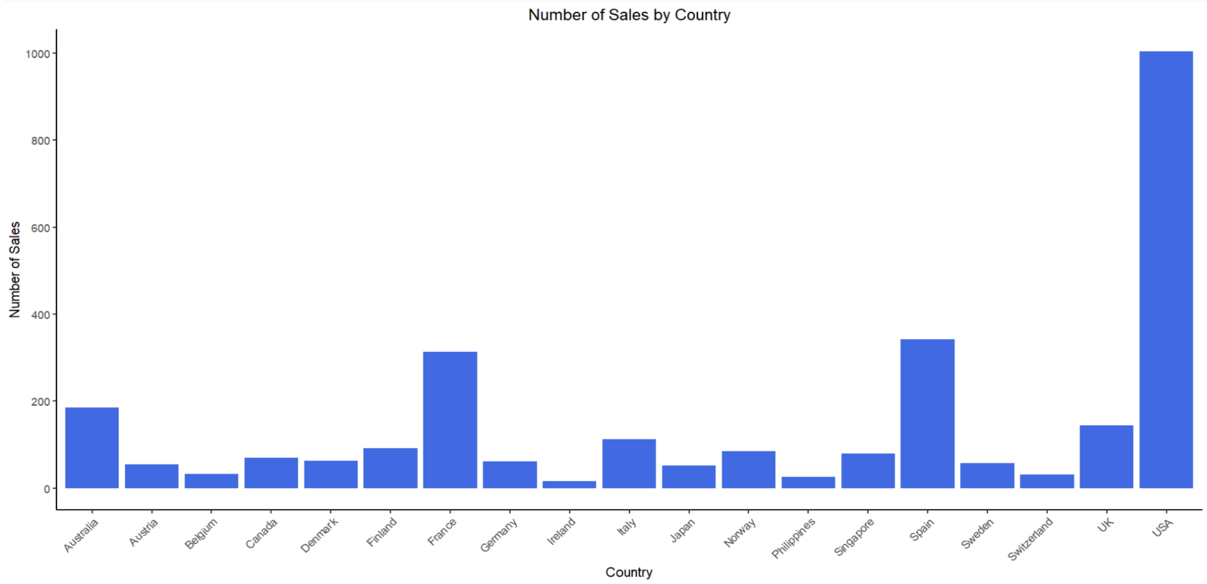
Q5 :

Code:

```
#Q5
#Count number of sales for each country
country_sales <- sales_data %>%
  group_by(COUNTRY) %>%
  mutate(Number_of_Sales = n()) %>%
  distinct(COUNTRY, Number_of_Sales) %>%
  arrange(COUNTRY)

#Plot bar plot
# Plot bar plot
ggplot(country_sales, aes(x = COUNTRY, y = Number_of_Sales)) +
  geom_bar(stat = "identity", fill = "royalblue") +
  labs(title = "Number of Sales by Country",
       x = "Country",
       y = "Number of Sales") +
  scale_y_continuous(breaks = seq(0, 1000, by = 200)) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))  # Rotate x-axis labels
```

Output:

Fig 23 Bar Plot



Explanation:

The bar plot above shows the number of sales by country. The X-axis represents countries in alphabetical order, encompassing a total of 19 countries, while the Y-axis indicates the number of sales ranging from 0 to 1000.

The bar plot clearly shows that the USA has the highest number of sales, reaching 1000. On the other hand, Ireland has the lowest number of sales. Spain is surprisingly in second place with around 400 purchases, showing that it has a huge gap between the top and second-highest sales.Additionally, the majority of countries fall within the range of 0 to 200 sales, suggesting a common trend of moderate sales volumes across most regions.