ENHANCING THE PERFORMANCE OF E-COMMERCE BUSINESSES

This report explores key strategies for e-commerce success: improving customer retention, increasing product revenues, and optimizing inventory management. It leverages extensive datasets from platforms like Shiprocket and INCREFF, covering financial metrics, SKU codes, inventory levels, and product details from major stores such as Ajio, Amazon, Flipkart, and others. Analyses focus on customer purchases, transaction rates, categories, fulfilment methods, and gross amounts.

Three main datasets—"International_sales_Report.csv," "Amazon_Sale_Report.csv," and "Sales_Report.csv"—provide insights into sales quantity, pricing, order details, and product performance. Identified areas for further investigation are accompanied by concise problem statements for targeted analysis.

Problem Statement 1 - Comparing the profitability of one-time VS repeat customers

Analysing customer retention rates is crucial for businesses aiming to improve loyalty which is vital for sustained profitability and long-term success. It measures the percentage of customers who are consistent with purchases. Strategies to boost customer satisfaction and loyalty can be developed by studying past purchases history and consequently, company discern patterns for the future. This involves examining preferences, purchase frequency, service interactions, and other factors influencing repeat business.

Data Cleaning & Manipulation

I had to thoroughly clean the dataset before analysing its structure and contents. I noticed that rows 18637 to 19676 were empty, and the columns were in the wrong order. To tackle this, I split the dataset and planned to rejoin it later during cleaning. Additionally, I used the slice() function to remove rows containing irrelevant information, as shown in Figure 1.

```
> correct_part <- int_sale_rep_data %>% slice(1:18635)

> wrong_part <- int_sale_rep_data %>% slice(19678:nrow(int_sale_rep_data)-1)
```

Figure 1

I rearranged the columns correctly using the data.frame() function, creating a new data frame to hold the required information. The size column was filled by extracting the last digits from the SKU codes using regular expressions (RegEx) and the grepl() function to search for matches in the string vector. Then, we rejoined the data frame with the previously sliced dataset, as illustrated in Figure 2.

```
> wrong_part_corrected <- data.frame(index = wrong_part$index, DATE = wrong_part$Months, Mon
ths = wrong_part$CUSTOMER, CUSTOMER = wrong_part$DATE, Style = wrong_part$Style, SKU = wrong
_part$SKU, Size = NA, PCS = wrong_part$Size, RATE = wrong_part$PCS, GROSS.AMT = wrong_part$R
ATE)

> wrong_part_corrected <- wrong_part_corrected %>% mutate(Size = ifelse(grepl("-", SKU), sub
(".*-([A-Za-z]+)$", "\\1", SKU), ""))
>
> # Rejoin the dataframe
> int_sale_rep_data <- bind_rows(correct_part, wrong_part_corrected)
```

Figure 2

After data cleaning, the total observations in this dataset is 34598 and the total number of variables is 10. The international sales data contains 9 categorical variables and 1 numerical variable based on the datatypes shown in Figure 3.

```
> str(int_sale_rep_data)
'data.frame':	34598 obs. of  10 variables:
 $ index    : int  0 1 2 3 4 5 6 7 8 9 ...
 $ DATE     : chr  "06-05-21" "06-05-21" "06-05-21" "06-05-21" ...
 $ Months   : chr  "Jun-21" "Jun-21" "Jun-21" "Jun-21" ...
 $ CUSTOMER : chr  "REVATHY LOGANATHAN" "REVATHY LOGANATHAN" "REVATHY LOGANATHAN" "REVATHY L
OGANATHAN" ...
 $ Style    : chr  "MEN5004" "MEN5004" "MEN5004" "MEN5009" ...
 $ SKU      : chr  "MEN5004-KR-L" "MEN5004-KR-XL" "MEN5004-KR-XXL" "MEN5009-KR-L" ...
 $ Size     : chr  "L" "XL" "XXL" "L" ...
 $ PCS      : chr  "1.00" "1.00" "1.00" "1.00" ...
 $ RATE     : chr  "616.56" "616.56" "616.56" "616.56" ...
 $ GROSS.AMT: chr  "617.00" "617.00" "617.00" "617.00" ...
```

Figure 3

1 variable contains missing values (NAs), which is the SKU variable. The missing value count is shown in Figure 4 below. Empty spaces were also replaced with NA to ensure it was calculated and removed. The missing values in the dataset were removed using the na.omit() function. I eliminate any missing values initially identified in the dataset to prevent errors, anomalies, or biases that may arise from their absence. This process also helps reduce the datasets complexity.

```
> # Check for NA values and remove them
> int_sale_rep_data[int_sale_rep_data == ""] <- NA
> colSums(is.na(int_sale_rep_data))
    index      DATE    Months  CUSTOMER     Style       SKU      Size       PCS 
        0         0         0         0         0      1320         0         0 
     RATE GROSS.AMT 
        0         0 
>
> int_sale_rep_data <- na.omit(int_sale_rep_data)
```

Figure 4

The summary and descriptive statistics of the variables in cleaned data are shown in Figure 5 below. There are now 33278 observations after removing the missing values.

```
> # Summary statistics of cleaned data
> summary(international_cleaned)
      DATE               Months            CUSTOMER             Style          
 Min.   :2021-06-05   Length:33278       Length:33278       Length:33278      
 1st Qu.:2021-09-18   Class :character   Class :character   Class :character  
 Median :2021-11-16   Mode  :character   Mode  :character   Mode  :character  
 Mean   :2021-11-21                                                           
 3rd Qu.:2022-02-21                                                           
 Max.   :2022-03-31                                                           
     SKU                Size                PCS             RATE          
 Length:33278       Length:33278       Min.   : 1.000   Length:33278      
 Class :character   Class :character   1st Qu.: 1.000   Class :character  
 Mode  :character   Mode  :character   Median : 1.000   Mode  :character  
                                       Mean   : 1.272                     
                                       3rd Qu.: 1.000                     
                                       Max.   :15.000                     
   GROSS.AMT     
 Min.   : 227.0  
 1st Qu.: 462.0  
 Median : 653.0  
 Mean   : 829.4  
 3rd Qu.: 950.0  
 Max.   :9735.0  
```

Figure 5

I performed a series of steps to manipulate the data for us to be able to analyse and compare the profitability of one – time and repeating customers. Firstly, I determined the date range of the sales in the dataset using the max() and min() functions – sales ranged from 5th June 2021 to 31st Match 2022 as seen below.

```
> # Problem Statement 1: Comparing the profitability of one-time vs repeat customers
> 
> # International sale report consist of sales from 5 June 2021 to 31 March 2022
> print(max(international_cleaned$DATE))
[1] "2022-03-31"
> print(min(international_cleaned$DATE))
[1] "2021-06-05"
```

Figure 6

As seen in Figure 7, the group_by() function was then used to group customers and their purchase histories as I specify new columns through the use of the summarise() function.

```
> # Identify unique customers and their purchase histories
> customer_purchases <- international_cleaned %>%
+   group_by(CUSTOMER) %>%
+   summarise(
+     First_Purchase = min(DATE),
+     Last_Purchase = max(DATE),
+     Days_Since_Last_Purchase = max(DATE) - min(DATE),
+     Purchase_Count = n_distinct(DATE), # How many purchases made on different dates
+     Total_purchase_frequency = n(), # Total number of items purchased
+     Total_amount_spent = sum(GROSS.AMT) # Total amount spent (INR) per customer
+   )
```

Figure 7

Using the nrow() function, we know that there are 123 unique customers in the dataset. The filter() function is used to determine repeating customers where their purchase count was greater than 1 like seen below.

```
> total_customers <- nrow(customer_purchases) # 123 unqiue customers
> print(paste("Total unique customers:", total_customers))
[1] "Total unique customers: 123"
>
> repeat_customers <- customer_purchases %>% filter(Purchase_Count > 1)

> total_repeat_customers <- nrow(repeat_customers)
> print(paste("Total repeat customers:", total_repeat_customers))
[1] "Total repeat customers: 41"
```

Figure 8

Therefore, there are 41 repeating customers as seen in the output of the nrow() function used on the repeat_customers data frame. Subsequently, there are a total of 82 one-time customers as shown in the figure below.

```
> # 82 one time customers
> total_one_time_customers <- nrow(one_time_customers)
> print(paste("Total one-time customers:", total_one_time_customers))
[1] "Total one-time customers: 82"
```

Figure 9

Furthermore, I analysed the dataset by computing the average quantity per purchase for both one – time and repeating customers. As in Figure 10, this was performed through the use of filter() function for the specific type of customers as well as the mean() function within the summarise() function to output the required average value.

```
> # Print the average quantity per purchase for one time customers
> one_time_customers_avg_qty <- international_cleaned %>%
+   filter(CUSTOMER %in% one_time_customers$CUSTOMER) %>%
+   summarise(
+     Avg_Qty_Per_Purchase = mean(PCS)
+   )
> print("One-time Customers Average Quantity Per Purchase")
[1] "One-time Customers Average Quantity Per Purchase"
> print(one_time_customers_avg_qty) # 1.24
  Avg_Qty_Per_Purchase
1             1.238345

> # Calculate average quantity per purchase for repeat customers
> repeat_customers_avg_qty <- international_cleaned %>%
+   filter(CUSTOMER %in% repeat_customers$CUSTOMER) %>%
+   group_by(CUSTOMER) %>%
+   summarise(
+     Avg_Qty_Per_Purchase = mean(PCS)
+   )
> print("Repeat Customers Average Quantity Per Purchase")
[1] "Repeat Customers Average Quantity Per Purchase"
> print(repeat_customers_avg_qty %>% summarise(Avg_Qty_Per_Purchase = mean(Avg_Qty_Per_Purch
ase))) # 1.26
# A tibble: 1 × 1
  Avg_Qty_Per_Purchase
                 <dbl>
1                 1.26
```

Figure 10

Thus, it is clear that the repeating customers have a higher average quantity per purchase which denotes that more products are bought by repeating customers. To ensure the analysis is impactful, I further computed the summary of the total amount spent for both one – time and repeating customers, shown in the figure below. This was done using the total amount spent column in the dataset.

```
> # Do repeat customer really spend more than one-time customers?
> # Summary statistics of the total amount spent by one-time customers and repeat cu
tomers
> one_time_summary <- one_time_customers %>% select(Total_amount_spent) %>%
+     summary(Total_amount_spent)
> print("One-time Customers Spending Summary")
[1] "One-time Customers Spending Summary"
> print(one_time_summary)
 Total_amount_spent
 Min.   : 10116
 1st Qu.: 34722
 Median : 60145
 Mean   :145298
 3rd Qu.:133384
 Max.   :885096

> repeat_summary <- repeat_customers %>% select(Total_amount_spent) %>%
+     summary(Total_amount_spent)
> print("Repeat Customers Spending Summary")
[1] "Repeat Customers Spending Summary"
> print(repeat_summary)
 Total_amount_spent
 Min.   :  35370
 1st Qu.: 119200
 Median : 216686
 Mean   : 382569
 3rd Qu.: 353076
 Max.   :3325150
```

Figure 11

Hence, it is evident that repeat customers spend more money than the one–time customers as outputted in the summary of both customer categories. I notice a major difference in mean values of both spending summaries for one–time and repeat customers. I make this analysis clearer by creating a data frame which only holds information on the percentage of the total amount spent for both customer categories, seen below.

```
> total_spent <- data.frame(Group = c("One Time Customer", "Repeat Customer"),
+                           count= c(sum(one_time_customers$Total_amount_spent),
+                                    sum(repeat_customers$Total_amount_spent)))
> total_spent <- total_spent %>%
+   mutate(percent = paste(100*round(count/sum(count), 3), "%"))
> head(total_spent)
              Group    count percent
1 One Time Customer 11914423  43.2 %
2   Repeat Customer 15685339  56.8 %
```

Figure 12

The repeating customer group spends a higher percentage compared to one-time customers, as seen in previous code. Additionally, I noticed that customers tend to return during specific times of the year. Using the group_by() function to group the data by sale date, it is evident that most customers return in January and around October, as depicted in Figure 13.

```
> # Analyze the distribution of repeat customers over time
> repeat_customer_trends <- repeat_customers %>%
+   mutate(YearMonth = floor_date(Last_Purchase, "month")) %>%
+   group_by(YearMonth) %>%
+   summarise(Repeat_Customer_Count = n()) %>%
+   arrange(desc(Repeat_Customer_Count))
>
> # Most of the customers repeated come back during January, a huge amount of customer also com
es back around October
> # January for new year? October for Deepavali? These spike in increase repeat customer count
 may be caused by special occasions
```

Figure 13

Peak sales align with New Year and Deepavali celebrations, especially for Indian clothing like sarees and kurtas. Deepavali sales spiking in October confirms this trend. With 66.7% of customers being non-repeat purchasers, theres ample room for improvement. Repeat customers, averaging 1.26 purchases, present an opportunity for bulk sales. To encourage repeat business, e-commerce firms should focus on improving customer satisfaction, potentially through loyalty programs and future purchase discounts.

Data Visualisations

```r
repeat_customers$metric <- repeat_customers$Total_amount_spent / repeat_customers$Total_purchase_frequency
one_time_customers$metric <- one_time_customers$Total_amount_spent / one_time_customers$Total_purchase_frequency
# Step 3: Calculate the average of the metrics
average_repeat <- mean(repeat_customers$metric, na.rm = TRUE)
average_one_time <- mean(one_time_customers$metric, na.rm = TRUE)

# Step 4: Create a data frame for plotting
average_data <- data.frame(
  Customer_Type = c("Repeat Customers", "One-time Customers"),
  Average_Metric = c(average_repeat, average_one_time)
```

Figure 14

The metric for repeat and one-time customers is calculated by dividing their total spent by their purchase frequency, stored in new columns. The average metric for each customer category is then computed using the mean() function with na.rm = TRUE to ensure accurate results by ignoring any missing values, enhancing analysis accuracy.

```r
ggplot(average_data, aes(x = Customer_Type, y = Average_Metric, fill = Customer_Type)) +
  geom_bar(stat = "identity", width = 0.5) +
  labs(title = "Average Spending Per Purchase Based on Customer Types",
       x = " ",
       y = "Average Spending Per Purchase") +
  theme_minimal() +
  scale_fill_manual(values = c("Repeat Customers" = "lightblue", "One-time Customers" = "pink")) +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5)) +
  geom_text(aes(label=round(Average_Metric, 2)),
            position=position_dodge(width=0.9), vjust=-0.25, size = 3)
```

Figure 15

A bar chart effectively illustrates the comparison of average spending between two customer categories by visually representing the difference in bar lengths. In Figure 15, I utilized the ggplot library to create a bar chart using average_data as the data source. Aesthetic mappings were set with x representing customer types, y for average metrics, and fill to differentiate bars by customer type. Adding bars to the plot with geom_bar(), I used the identity statistic to directly plot metric values and specified a width of 0.5. Additionally, I displayed average metric values above the bars using geom_text(), formatted to two decimal places and positioned appropriately.

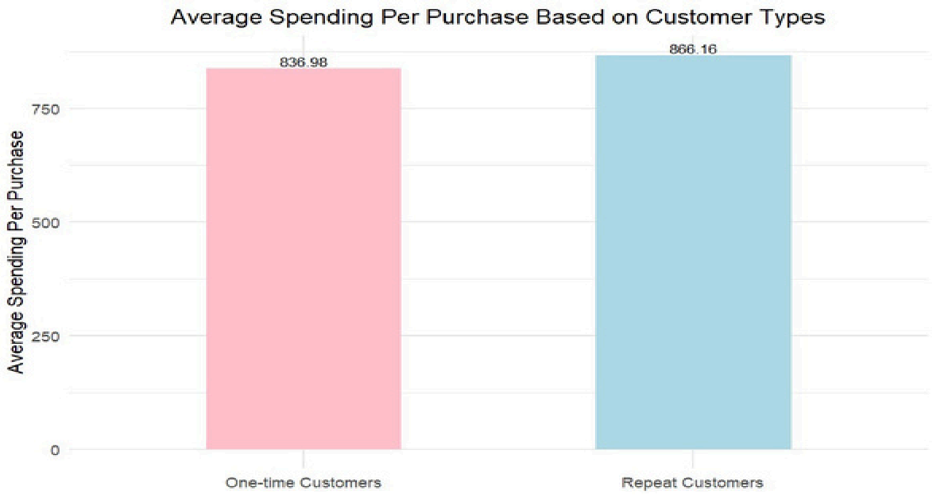

Figure 16

Figure 16 illustrates the difference in average spending between one-time and repeat customers, showing that repeat customers tend to spend more on average per purchase; thus, repeat customers are likely more profitable for the company. Consequently, the companys priority should be increasing repeat customers, using strategies such as offering customer loyalty program which could efficiently boost overall revenue, enhance profitability and build a more stable customer base.

A data frame is created like in Figure 17, where it contains two groups representing each customer type. Each group has it records of the sum for total_amount_spent from the initial dataset. The mutate function is used to add a new column percent to the total_spent data frame. This column represents the percentage each group contributes to the total amount spent, calculated by dividing the count of each group by the total count and then formatted as a percentage.

```
total_spent <- data.frame(Group = c("One Time Customer", "Repeat Customer"),
                          count= c(sum(one_time_customers$Total_amount_spent),
                           sum(repeat_customers$Total_amount_spent)))
total_spent <- total_spent %>%
  mutate(percent = paste(100*round(count/sum(count), 3), "%"))
```

Figure 17

A pie chart clearly shows the relative spending between the two. With only two categories, pie chart with two slices is very simple and easy to understand at a glance. There is no risk of clutter, and the viewer can quickly grasp the information being presented. Hence, it effectively highlights the difference in spending between the two categories, making it visually apparent how the expenditures compare.

```
total_spent <- data.frame(Group = c("One Time Customer", "Repeat Customer"),
                          count= c(sum(one_time_customers$Total_amount_spent),
                           sum(repeat_customers$Total_amount_spent)))
total_spent <- total_spent %>%
  mutate(percent = paste(100*round(count/sum(count), 3), "%"))
View(total_spent)

ggplot(total_spent, aes(x="", y=count, fill=Group)) +
  geom_col(color = "black") +
  geom_text(aes(label = percent),
            position = position_stack(vjust = 0.5),
            size = 8,
            color = c("white","black")) +
  coord_polar("y", start=0) +
  labs(title = "Total Spent by both Groups") +
  scale_fill_manual(values = c("pink","lightblue")) +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        panel.border = element_blank(),
        axis.text = element_blank(),
        panel.grid = element_blank(),
        axis.ticks = element_blank(),
        panel.background = element_rect(fill = "#ebf2ff"),
        plot.background = element_rect(fill = "#ebf2ff"),
        legend.background = element_rect(fill = "#ebf2ff"),
        axis.title.y = element_blank(),
        axis.title.x = element_blank())
```

Figure 18

I utilized the ggplot library to generate a pie chart, as depicted in Figure 18. Initializing the plot with the total_spent data frame, I defined aesthetic mappings where y=count represented bar heights and fill=Group distinguished colours. Adding coloured bars with black borders to the plot using geom_col(), I included percentage labels on each bar positioned at the middle with geom_text(). Transforming the plot into a circular polar coordinate system with coord_polar("y", start=0), Ilabelled the plot with a title "Total Spent by both Groups" using labs(). Manually setting colours for the bars representing each group with scale_fill_manual(), I then adjusted various theme elements with theme(), such as removing the background grid and axis titles, and setting the plot background colour, to enhance appearance.

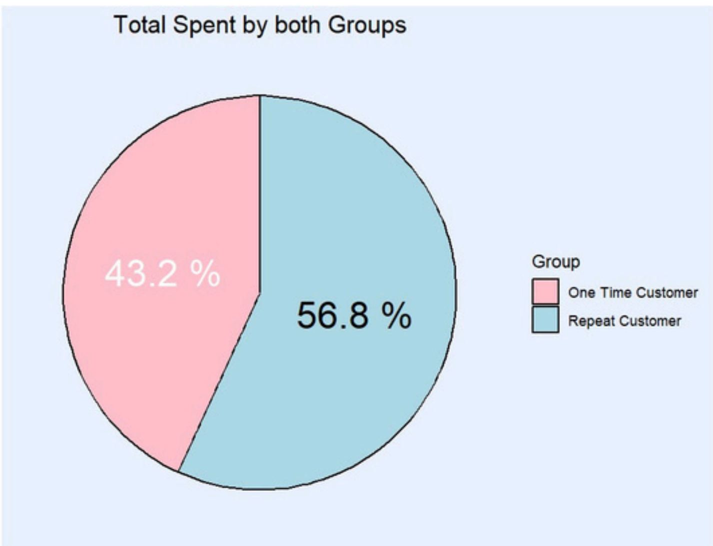

Figure 19

The pie chart illustrates total expenditures from two customer demographics: one-time and repeat customers. Repeat customers contribute a larger portion to the total expenditure; which is a significant portion of the business's revenue. Companies are able to analysis this data to comprehend its revenue sources and enhancing customer retention to foster repeat business. However, it's worth noting that while repeat customers spend more, the disparity in total spending between the two groups is relatively small, suggesting that their contribution is not significant compared to one-time customers.

Using the repeat_customers data frame, a new column called YearMonth is created with mutate(), rounding down Last_Purchase dates to the nearest month using floor_date() from the lubridate package. This effectively groups all dates within the same month. The data is then grouped by YearMonth with group_by(), and summarise() calculates the number of repeat customers for each month. arrange() sorts the data frame in descending order based on repeat_customer_count, showcasing months with the highest repeat customers first.

```
repeat_customer_trends <- repeat_customers %>%
  mutate(YearMonth = floor_date(Last_Purchase, "month")) %>%
  group_by(YearMonth) %>%
  summarise(Repeat_Customer_Count = n()) %>%
  arrange(desc(Repeat_Customer_Count))
```

Figure 20

Line charts are particularly effective for illustrating trends over a period, making it easy to observe how data points change over intervals. They help in recognizing patterns or behaviours in data, such as seasonal fluctuations or long-term growth.

```
ggplot(repeat_customer_trends, aes(x = YearMonth, y = Repeat_Customer_Count)) +
  geom_line() +
  labs(title = "Trends of Repeat Customers Over Time", x = "Months", y = "Number of Repeat Customers") +
  scale_x_date(date_breaks = "1 month", date_labels = "%Y %b") +
  theme(plot.title = element_text(hjust = 0.5, size = 15),
        panel.background = element_rect(fill = "#ebf2ff")) +
  geom_smooth(method=lm, se=FALSE, col='lightblue', size=1)
```

Figure 21

I created a line chart using ggplot, with repeat_customer_trends data frame plotted against YearMonth on the x-axis and Repeat_Customer_Count on the y-axis. Adding a line plot layer with Geom_line(), Idepicted trends over time. Customizing plot labels with labs(), I labelled the title as "Trends of Repeat Customers Over Time", x-axis as "Months", and y-axis as "Number of Repeat Customers". Formatting the x-axis with scale_x_date(), I set breaks to occur every month and labels to display year and month. Adjusting plot appearance, I centred the title, set its size, and modified the background colour of the plotting area with theme(). Finally, I smoothed the line using linear regression without displaying standard error bands and customized the line colour and thickness with geom_smooth().

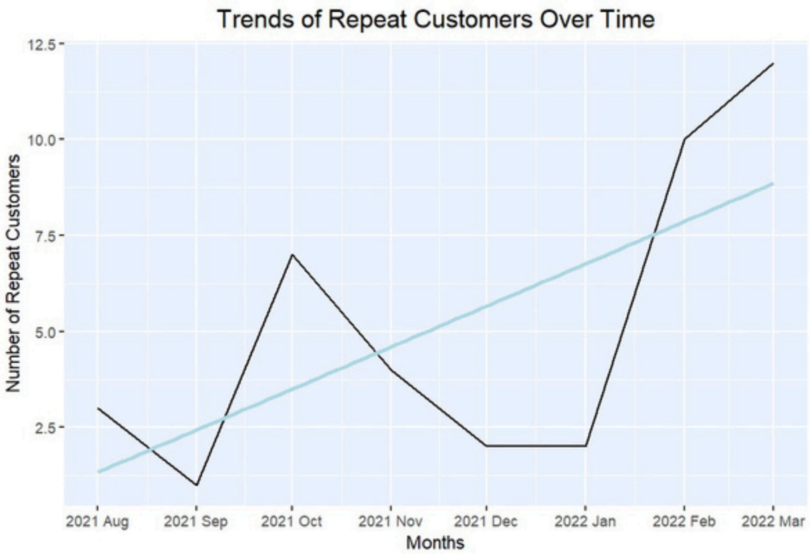

Figure 22

The line graph illustrates the frequency of repeat customers over time, with notable peaks observed in October and February. These spikes coincide with significant holidays such as Deepavali in October and Lunar New Year in February. Leveraging these seasonal fluctuations could offer a strategic opportunity for the business to enhance both customer retention and satisfaction. By prioritizing efforts during these high-traffic periods, the business can capitalize on increased customer engagement and potentially further solidify its market presence.

Problem Statement 2 - Analysis of revenue made in each product category

Discrepancies between shipment volume and revenue can signal inefficiencies or untapped opportunities. Understanding these gaps is vital for optimizing resource allocation and maximizing profitability. Factors like demand fluctuations, customer preferences, and average order values contribute to these differences. Comprehensive shipping data, including shipment numbers, product categories, destinations, and segmented revenue data, are essential for analysis. This problem statement utilizes the amazon_sale_report.csv dataset.

Data Cleaning & Manipulation

Before I could delve into analysing the dataset, thorough data cleaning was imperative. Initially, I used the is.na() function to determine the number of missing values in the columns of the dataset

However, upon further checking through the dataset, I noticed that many data were depicted through empty strings thus not showcased in the count of missing values. Hence, it is required to replace the empty strings with NA values to be detected using the is.na() function.

```
> # Replace empty string values with NA
> amazon_sale_rep_data[amazon_sale_rep_data == ''] <- NA
>
> # Now we can obtain the real missing values
> colSums(is.na(amazon_sale_rep_data))
            index          Order.ID              Date
                0                 0                 0
           Status        Fulfilment     Sales.Channel
                0                 0                 0
ship.service.level            Style               SKU
                0                 0                 0
         Category              Size              ASIN
                0                 0                 0
   Courier.Status               Qty          currency
             6872                 0              7795
           Amount         ship.city        ship.state
             7795                33                33
 ship.postal.code      ship.country     promotion.ids
               33                33             49153
              B2B      fulfilled.by       Unnamed..22
                0             89698             49050
```

Figure 23

After analysing the dataset, I found 9 variables with numerous missing values. To further clean the dataset, I removed redundant columns like ship country and currency details. These columns didnt contribute to the analysis as information like ship_city and ship_state already implied the dataset was for India, making other columns irrelevant.

```
> # Remove redundant columns that are idea just for references or already implied
> # (ship country and currency we already know it's a dataset for india)
> amazon_sale_rep_data <- select(amazon_sale_rep_data, -index, -promotion.ids, -Unnamed..22, -fulfi
led.by, -ship.country, -currency)
```

Figure 24

The dataset included information on the date however it was not in the accurate data type. Thus, conversion was done to the Date column using the mutate() and as.Date() function.

```
> # Convert Date to Date datatype
> amazon_sale_rep_data <- amazon_sale_rep_data %>%
+   mutate(Date_ = as.Date(Date, format="%m-%d-%y"))
```

Figure 25

After significant amount of data cleaning, the dataset consists of 128975 observations with 18 variables. There are 14 categorical variables, 3 numerical variables and 1 Date variable; as seen in the figure 26.

```
> # Now has 128975 observations 18 columns
> # Datatype seems appropriate as well
> str(amazon_sale_rep_data)
'data.frame':	128975 obs. of  18 variables:
 $ Order.ID          : chr  "405-8078784-5731545" "171-9198151-1101146" "404-0687676-7273146" "403-9615377-8133951" ...
 $ Date              : Date, format: "2022-04-30" "2022-04-30" "2022-04-30" ...
 $ Status            : chr  "Cancelled" "Shipped - Delivered to Buyer" "Shipped" "Cancelled" ...
 $ Fulfilment        : chr  "Merchant" "Merchant" "Amazon" "Merchant" ...
 $ Sales.Channel     : chr  "Amazon.in" "Amazon.in" "Amazon.in" "Amazon.in" ...
 $ ship.service.level: chr  "Standard" "Standard" "Expedited" "Standard" ...
 $ Style             : chr  "SET389" "JNE3781" "JNE3371" "J0341" ...
 $ SKU               : chr  "SET389-KR-NP-S" "JNE3781-KR-XXXL" "JNE3371-KR-XL" "J0341-DR-L" ...
 $ Category          : chr  "Set" "kurta" "kurta" "Western Dress" ...
 $ Size              : chr  "S" "3XL" "XL" "L" ...
 $ ASIN              : chr  "B09KXVBD7Z" "B09K3WFS32" "B07WV4JV4D" "B099NRCT7B" ...
 $ Courier.Status    : chr  NA "Shipped" "Shipped" NA ...
 $ Qty               : int  0 1 1 0 1 1 1 1 0 1 ...
 $ Amount            : num  648 406 329 753 574 ...
 $ ship.city         : chr  "MUMBAI" "BENGALURU" "NAVI MUMBAI" "PUDUCHERRY" ...
 $ ship.state        : chr  "MAHARASHTRA" "KARNATAKA" "MAHARASHTRA" "PUDUCHERRY" ...
 $ ship.postal.code  : num  400081 560085 410210 605008 600073 ...
 $ B2B               : chr  "False" "False" "True" "False" ...
```

Figure 26

However, several duplicate values were found while performing analysis. Thus, I use the distinct() function to produce a dataset with only unique values. Therefore, now I have a dataset with 128969 observations, where 6 were duplicate values.

```
> # Remove duplicate values (6 of them)
> amazon_sale_rep_data <- amazon_sale_rep_data %>% distinct()
> str(amazon_sale_rep_data)
'data.frame':	128969 obs. of  18 variables:
 $ Order.ID          : chr  "405-8078784-5731545" "171-9198151-1101146" "404-0687676-7273146" "403-9615377-8133951" ...
 $ Date              : Date, format: "2022-04-30" "2022-04-30" "2022-04-30" ...
 $ Status            : chr  "Cancelled" "Shipped - Delivered to Buyer" "Shipped" "Cancelled" ...
 $ Fulfilment        : chr  "Merchant" "Merchant" "Amazon" "Merchant" ...
 $ Sales.Channel     : chr  "Amazon.in" "Amazon.in" "Amazon.in" "Amazon.in" ...
 $ ship.service.level: chr  "Standard" "Standard" "Expedited" "Standard" ...
 $ Style             : chr  "SET389" "JNE3781" "JNE3371" "J0341" ...
 $ SKU               : chr  "SET389-KR-NP-S" "JNE3781-KR-XXXL" "JNE3371-KR-XL" "J0341-DR-L" ...
 $ Category          : chr  "Set" "kurta" "kurta" "Western Dress" ...
 $ Size              : chr  "S" "3XL" "XL" "L" ...
 $ ASIN              : chr  "B09KXVBD7Z" "B09K3WFS32" "B07WV4JV4D" "B099NRCT7B" ...
 $ Courier.Status    : chr  NA "Shipped" "Shipped" NA ...
 $ Qty               : int  0 1 1 0 1 1 1 1 0 1 ...
 $ Amount            : num  648 406 329 753 574 ...
 $ ship.city         : chr  "MUMBAI" "BENGALURU" "NAVI MUMBAI" "PUDUCHERRY" ...
 $ ship.state        : chr  "MAHARASHTRA" "KARNATAKA" "MAHARASHTRA" "PUDUCHERRY" ...
 $ ship.postal.code  : num  400081 560085 410210 605008 600073 ...
 $ B2B               : chr  "False" "False" "True" "False" ...
```

Figure 27

I remove the missing values that were found from the dataset initially to avoid errors, anomalies or biases formed from missing values. Removing missing values also ensures that the complexity of the dataset is reduced.

```
> amazon_sale_rep_data <- na.omit(amazon_sale_rep_data)
> nrow(amazon_sale_rep_data)
[1] 116013
```

Figure 28

Hence, the summary and descriptive statistics of the variables in cleaned data are shown in Figure 29 below.

```
> # Summary statistics of cleaned data
> summary(amazon_cleaned)
   Order.ID              Date                Status           Fulfilment
 Length:116013      Min.   :2022-03-31   Length:116013      Length:116013
 Class :character   1st Qu.:2022-04-20   Class :character   Class :character
 Mode  :character   Median :2022-05-10   Mode  :character   Mode  :character
                    Mean   :2022-05-12
                    3rd Qu.:2022-06-05
                    Max.   :2022-06-29
 Sales.Channel      ship.service.level    Style              SKU                Category
 Length:116013      Length:116013      Length:116013      Length:116013      Length:116013
 Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character


     Size               ASIN            Courier.Status          Qty             Amount
 Length:116013      Length:116013      Length:116013      Min.   :1.000   Min.   :   0.0
 Class :character   Class :character   Class :character   1st Qu.:1.000   1st Qu.: 449.0
 Mode  :character   Mode  :character   Mode  :character   Median :1.000   Median : 606.0
                                                          Mean   :1.004   Mean   : 649.8
                                                          3rd Qu.:1.000   3rd Qu.: 788.0
                                                          Max.   :8.000   Max.   :5584.0

  ship.city          ship.state         ship.postal.code      B2B
 Length:116013      Length:116013      Min.   :110001   Length:116013
 Class :character   Class :character   1st Qu.:382424   Class :character
 Mode  :character   Mode  :character   Median :500032   Mode  :character
                                       Mean   :463320
                                       3rd Qu.:600017
                                       Max.   :855117
```

Figure 29

Continuing the process of analysing our dataset, I manipulate the data by creating a revenue column in the dataset using the mutate() function. As mentioned in the introduction of this problem statement, the revenue is calculated by obtaining the product quantity and the amount of the item as shown in Figure 30.

```
> #creating revenue column
> amazon_cleaned <- amazon_cleaned %>%
+   mutate(Revenue = Qty * Amount)
```

Figure 30

Adding to that, I group the dataset based on the different categories of the product using the group_by function and output the total revenue using the sum() function on the newly created revenue column. Since I was unsure of the exact different categories and their product count in the dataset, I performed the count() function to output the number of products for each category. Hence, I realised that there are 9 different product categories as in Figure 31.

```
> #total revenue for each category
> total_revenue <- amazon_cleaned %>%
+   group_by(Category) %>%
+   summarise(Total_Revenue = sum(Revenue))
> head(total_revenue)
# A tibble: 6 x 2
  Category       Total_Revenue
  <chr>                  <dbl>
1 Blouse                441259
2 Bottom                142870
3 Dupatta                  915
4 Ethnic Dress          762949
5 Saree                 125767
6 Set                 37923384
```

```
> #counts in Category
> counts <- amazon_cleaned%>%
+   count(Category)
> print(counts)
        Category     n
1         Blouse   837
2         Bottom   393
3        Dupatta     3
4   Ethnic Dress  1050
5          Saree   148
6            Set 45077
7            Top  9864
8  Western Dress 13893
9          kurta 44748
```

Figure 31

To compare the revenue based on the number of products based on each category, we create a data frame while merging the total revenue and the counts using the Category column.

```
> #combine total revenue and counts
> merged_df <- merge(total_revenue, counts, by = "Category")
> head(merged_df)
     Category Total_Revenue     n
1      Blouse        441259   837
2      Bottom        142870   393
3     Dupatta           915     3
4 Ethnic Dress       762949  1050
5       kurta      20667948 44748
6       Saree        125767   148
```

Figure 22

The average revenue per product was calculated by dividing the total_revenue by the number of products sold. A data frame was created with a new column for the average revenue per product using the mutate() function.

```
> #finding the average revenue
> merged_df <- merged_df %>%
+   mutate(Average_Revenue_Per_Product = Total_Revenue / `Number of products sold`)
> head(merged_df)
     Category Total_Revenue Number of products sold Average_Revenue_Per_Product
1      Blouse        441259                     837                    527.1912
2      Bottom        142870                     393                    363.5369
3     Dupatta           915                       3                    305.0000
4 Ethnic Dress       762949                    1050                    726.6181
5       kurta      20667948                   44748                    461.8742
6       Saree        125767                     148                    849.7770
```

Figure 33

Data Visualisations

```
melt_bar <- melt(merged_df[,c('Category','Total_Revenue',
                              'Number of products sold')],id.vars = 1)
View(melt_bar)
merged_df2 <- merged_df %>%
  select(Category, Average_Revenue_Per_Product)

melt_bar2 <- left_join(melt_bar, merged_df2, by = "Category")
View(melt_bar2)

ggplot(data=melt_bar2, aes(x= reorder(Category, value), y=value, fill=variable, group=variable)) +
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label=value), position=position_dodge(width=0.9), vjust=-0.25, size = 2.5) +
  scale_fill_manual(values = c("pink","lightblue")) +
  scale_color_manual(values=c("blue")) +
  scale_y_log10() +
  geom_line(aes(x = reorder(Category, value), y = Average_Revenue_Per_Product,
                color = "Average_Revenue_Per_Product"), group = 1) +
  theme_bw() +
  labs(title = "Total Revenue vs Quantity Sold for each Product",
       x = "Products",
       y = "Frequency") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=0.5),
        axis.text.y = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 15),
        panel.grid = element_line(color = "white"),
        panel.background = element_rect(fill = "#ebf2ff"),
        legend.position = c(0.16,0.88),
        legend.background = element_rect(fill = "#ebf2ff"),
        legend.text = element_text(size = 8),
        legend.title = element_blank(),
        legend.spacing.y = unit(-0.2, "cm"))
```

Figure 34

I reshaped the dataset using melt_bar() to facilitate visualization, focusing on specific columns like Category, Total Revenue, and Number of products sold. Calculating the average revenue per product, I merged both datasets with left_join(). The resulting side-by-side bar chart enabled straightforward category comparison within each group.

Employing ggplot, I constructed the chart: setting up initial parameters with ggplot(), adding adjacent bars for different variables with geom_bar(), and labelling values atop bars using geom_text(). I defined fill colours for bars with scale_fill_manual(), applied a logarithmic scale to the y-axis with scale_y_log10(), and plotted a line representing Average_Revenue_Per_Product with geom_line(). Customizations to plot appearance were made with theme_bw(), labs(), and Theme().

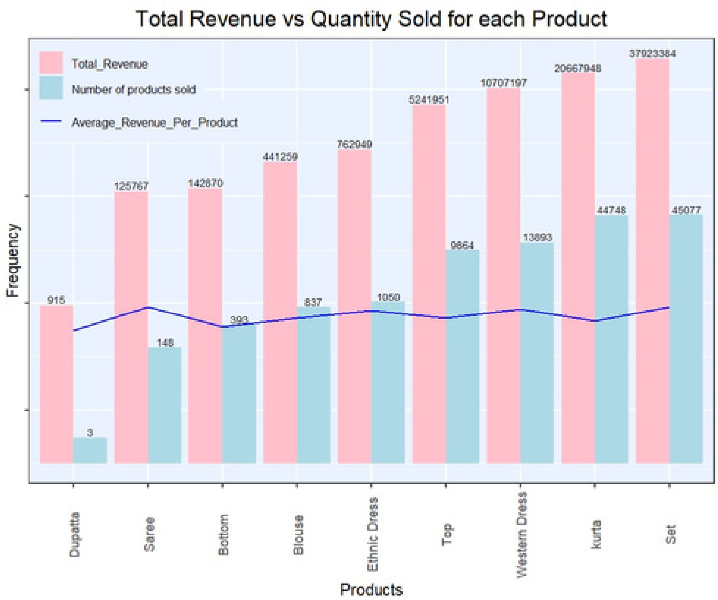

Figure 35

This side-by-side bar chart illustrates both the total revenue and total quantity for each product sold, aiding the business in gauging demand and product popularity. Consequently, the business can implement tailored strategies such as price adjustments and targeted advertising campaigns. Additionally, the accompanying graph displays the average revenue per product, the contribution of each product to the overall revenue, for each category, calculated by dividing total revenue by quantity sold; this is important for informed decision- making.

Problem Statement 3 - Analysing the popularity of different colours across all products

Analysing colour popularity across products is vital for inventory optimization and sales growth. This entails assessing colour popularity with stock levels and sales data to uncover trends and customer preferences. Detailed data on SKU codes, stock levels, sales quantities, and demand trends is essential. By identifying colours with high sales and consistent demand, businesses can optimize inventory, reduce holding costs, and improve sales. This analysis utilises the amazon_sale_report.csv and sales_report.csv datasets to enhance inventory turnover and meet customer demand more effectively.

Data Cleaning & Manipulation

The total observations in this dataset is 9271 and the total number of variables is 6. The sales report data contains 5 categorical variables and 1 numerical variable based on the datatypes shown in Figure 36.

```
> # Remove index (redundant)
> sale_rep_data <- select(sale_rep_data, -index)
> # Lots of empty rows at the bottom, datatype looks fine
> # 9271 observations
> str(sale_rep_data)
'data.frame':	9271 obs. of  6 variables:
 $ SKU.Code  : chr  "AN201-RED-L" "AN201-RED-M" "AN201-RED-S" "AN201-RED-XL" ...
 $ Design.No.: chr  "AN201" "AN201" "AN201" "AN201" ...
 $ Stock     : num  5 5 3 6 3 11 3 16 8 14 ...
 $ Category  : chr  "AN : LEGGINGS" "AN : LEGGINGS" "AN : LEGGINGS" "AN : LEGGINGS" ...
 $ Size      : chr  "L" "M" "S" "XL" ...
 $ Color     : chr  "Red" "Red" "Red" "Red" ...
```

Figure 36

There are no variables containing missing values (NAs). However, this also means that there are empty strings so there is a need to convert them to NA. After conversion and removing them using the na.omit function, the total number of observations after removing the missing values is 9188 as shown in Figure 37. I eliminate any missing values identified in the dataset initially to prevent errors, anomalies, or biases that may arise from them. This process also serves to simplify the dataset, reducing its complexity.

```
> # Not much NA detected, means they're empty spaces so convert them to NA
> sale_rep_data[sale_rep_data == ""] <- NA
> #remove NA values
> sale_rep_data <- na.omit(sale_rep_data)
> # 9188 observations now
> str(sale_rep_data)
'data.frame':	9188 obs. of  6 variables:
 $ SKU.Code  : chr  "AN201-RED-L" "AN201-RED-M" "AN201-RED-S" "AN201-RED-XL" ...
 $ Design.No.: chr  "AN201" "AN201" "AN201" "AN201" ...
 $ Stock     : num  5 5 3 6 3 11 3 16 8 14 ...
 $ Category  : chr  "AN : LEGGINGS" "AN : LEGGINGS" "AN : LEGGINGS" "AN : LEGGINGS" ...
 $ Size      : chr  "L" "M" "S" "XL" ...
 $ Color     : chr  "Red" "Red" "Red" "Red" ...
 - attr(*, "na.action")= 'omit' Named int [1:83] 136 143 198 199 224 249 274 547 902 903 ...
  ..- attr(*, "names")= chr [1:83] "136" "143" "198" "199" ...
```

Figure 37

The summary and descriptive statistics of the variables in cleaned data are shown in Figure 38.

```
> sale_cleaned <- sale_rep_data
> # Summary statistics of cleaned data
> summary(sale_cleaned)
   SKU.Code          Design.No.            Stock           Category             Size              Color          
 Length:9185        Length:9185        Min.   :   0.00   Length:9185        Length:9185        Length:9185       
 Class :character   Class :character   1st Qu.:   3.00   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Median :   8.00   Mode  :character   Mode  :character   Mode  :character  
                                       Mean   :  26.39                                                           
                                       3rd Qu.:  31.00                                                           
                                       Max.   :1234.00                                                           
```

Figure 38

In view of analysis based on the problem statement, it was required to join the Amazon dataset together with the sales dataset to comment on the popularity of sales based on the product colours. I used both cleaned versions of the dataset to perform the left_join function as seen in Figure 39. This function allows us to analyse only the products that have made sales.

```
> #PS3
> #rename SKU
> sale_cleaned <- sale_cleaned %>%
+   rename(SKU = SKU.Code)
> #join on amazon dataset and sales report
> combined_dataset <- left_join(amazon_cleaned, sale_cleaned, by = "SKU")
```

Figure 39

Since the analysis was on the sales made based on the colours of the product, I removed the unwanted columns from the combined datasets. Hence, this outputs the Colour and Category columns alone as seen below.

```
> #remove unwanted columns
> cleared_dataset <- combined_dataset[,c("Color", "Category.x")]
> cleaned_dataset <- cleared_dataset %>%
+   filter(!is.na(Color))
> head(cleaned_dataset)
        Color Category.x
1       Green      kurta
2        Pink      kurta
3        Pink      kurta
4       Cream        Set
5 Light Green      kurta
6       Mauve      kurta
```

Figure 40

As seen in the categories, there are different types of products available in the dataset and it was necessary to find the number of products in each category with a specific colour. Thus, I performed a count function on each category based on each colour. The different categories include Kurta, Blouse, Bottom, Ethnic Dress, Saree, Set, Top and Western Dress. The figure below demonstrates the process of categorising the Kurta products based on the colour and their count. The same process was done for the other product categories.

```
> #most popular colors for kurta in descending order
> kurta_counts <- cleaned_dataset %>%
+   filter(Category.x == "kurta") %>%
+   count(Color) %>%
+   arrange(desc(n)) %>%
+   rename(colour_count = n)
> head(kurta_counts)
   Color colour_count
1   Blue         1892
2   Pink          964
3  Black          901
4  Green          747
5  Peach          629
6 Maroon          578
```

Figure 41

<u>Data Visualisations</u>

A heatmap visually represents colour distribution across product categories, aiding in spotting trends and patterns. It uses a color-coded matrix to easily identify popular and less popular colours within each category. This helps determine if certain colours are consistently popular across multiple categories or if preferences vary by category.

```r
# Create a list of all category counts with their respective category names
category_counts_list <- list(
  Kurta = kurta_counts,
  Blouse = blouse_counts,
  Bottom = bottom_counts,
  `Ethnic Dress` = ethnic_dress_counts,
  Saree = saree_counts,
  Set = set_counts,
  Top = top_counts,
  `Western Dress` = western_dress_counts
)

# Filter out NULL entries and combine the remaining data frames
all_counts <- lapply(names(category_counts_list), function(category) {
  df <- category_counts_list[[category]]
  if (!is.null(df)) {
    df <- mutate(df, Category = category)
  }
  df
}) %>% bind_rows()

 # Define a mapping from specific colors to broader color categories
 color_mapping <- c(
   "Blue" = "Blue", "Teal" = "Blue", "Turquoise Blue" = "Blue", "Navy Blue" = "Blue",
   "Powder Blue" = "Blue", "Sky Blue" = "Blue", "Dark Blue" = "Blue", "Light Blue" = "Blue",
   "Navy" = "Blue", "Turquoise" = "Blue", "TEAL BLUE" = "Blue", "NAVY" = "Blue",
   "TEAL BLUE " = "Blue",
   "Pink" = "Pink", "Light Pink" = "Pink", "Magenta" = "Pink", "Hot Pink" = "Pink",
   "Coral Pink" = "Pink", "CORAL PINK" = "Pink",
   "Black" = "Black",
   "Green" = "Green", "Light Green" = "Green", "Dark Green" = "Green",
   "Olive Green" = "Green", "Sea Green" = "Green", "Olive" = "Green",
   "Turquoise Green" = "Green", "Teal Green" = "Green", "TEAL GREEN " = "Green",
   "Yellow" = "Yellow", "Gold" = "Yellow", "Light Yellow" = "Yellow",
   "LIGHT YELLOW" = "Yellow", "Mustard" = "Yellow", "Lemon" = "Yellow", "LEMON" = "Yellow",
   "LEMON " = "Yellow",
   "Red" = "Red", "Dark Red" = "Red", "Maroon" = "Red", "Wine" = "Red",
   "White" = "White", "OFF WHITE" = "White",
   "Grey" = "Grey", "Charcoal" = "Grey",
   "Peach" = "Peach", "Coral" = "Peach", "CORAL" = "Peach", "Coral Orange" = "Peach",
   "CORAL " = "Peach",
   "Orange" = "Orange", "ORANGE" = "Orange", "Rust" = "Orange", "CORAL ORANGE" = "Orange",
   "Purple" = "Purple", "Violet" = "Purple", "Mauve" = "Purple", "Indigo" = "Purple",
   "Brown" = "Brown", "Beige" = "Brown", "Tan" = "Brown", "Light Brown" = "Brown",
   "Multicolor" = "Multicolor",
   "Cream" = "Cream", "Taupe" = "Brown", "Lemon Yellow" = "Yellow", "Khaki" = "Brown",
   "LIME GREEN" = "Green", "BURGUNDY" = "Red", "MINT" = "Green", "MINT GREEN" = "Green"
 )
```

Figure 42

After manipulating the dataset by creating a list of all category counts with their respective category names while making sure that the null data have been filtered out, I define a map from specific colours to broader categories.

```r
# Map specific colors to broader color categories
all_counts$BroadColor <- color_mapping[all_counts$Color]

# Aggregate the data by BroadColor and Category
broad_color_counts <- all_counts %>%
  group_by(Category, BroadColor) %>%
  summarise(Count = sum(n, na.rm = TRUE)) %>%
  ungroup()

# Create the heatmap OF COLOUR CATEGORY
ggplot(broad_color_counts, aes(x = BroadColor, y = Category, fill = Count, label = Count)) +
  geom_tile() +
  geom_text(color = "black", size = 3) +
  scale_fill_gradient(low = "pink", high = "blue") +
  labs(title = "Heatmap of Color Popularity by Category",
       x = "Color",
       y = "Category") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))
```

Figure 43

I used ggplot to create a heatmap showing colour popularity by category. The x-axis represents broad colours, the y-axis represents categories, and tile fill represents counts. I added text labels for counts and set a pink-to-blue gradient for fill colour. The plots title is "Heatmap of Colour Popularity by Category", with x-axis labelled "Colour" and y-axis labelled "Category". I applied a minimal theme and customized it by rotating x-axis labels by 45 degrees and centring the plot title.

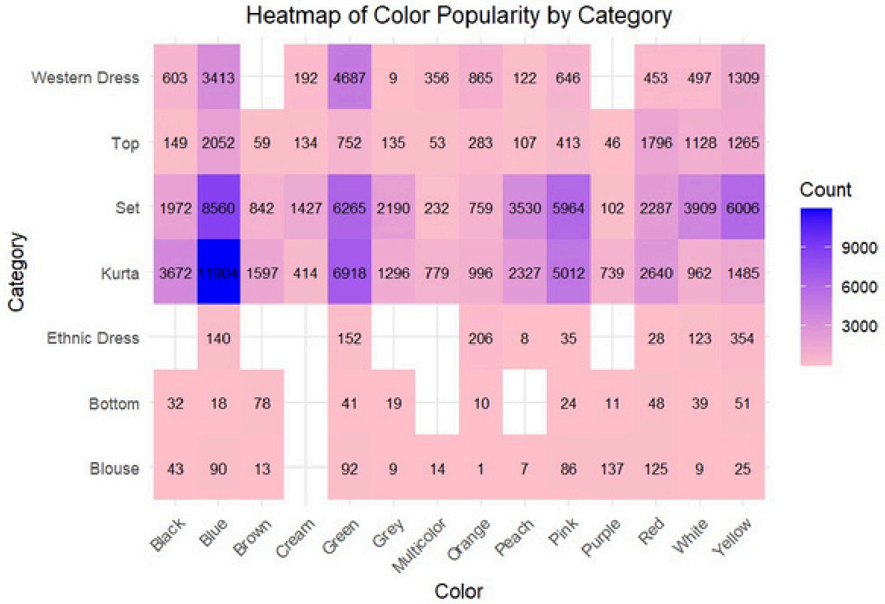

Figure 44

The heatmap depicts colour frequency across clothing types, illustrating Kurta as the most popular choice, indicated by the darkest shade of blue; which emerges as the top choice for tops and sets and the second most preferred for Western dresses, suggesting its overall popularity therefore the prevalence of blue might be skewed due to its various shades, with 13 categorized as different blues out of 71 total colours. This extensive categorization could inflate the perception of blues dominance compared to a more balanced categorization.

Bubble charts are visually appealing and simplify complex data by representing product categories as bubbles, with each bubbles size and colour indicating differences across categories and colours.

```r
# Total counts for each broad color
total_color_counts <- all_counts %>%
  group_by(BroadColor) %>%
  summarise(TotalCount = sum(n, na.rm = TRUE)) %>%
  ungroup()

# Define the sizes corresponding to each broad color category
size_mapping <- c(1, 13, 6, 1, 9, 2, 1, 4, 5, 6, 4, 5, 2, 8)

# Add a new column with the corresponding sizes
total_color_counts$BubbleSize <- size_mapping

# Create the bubble plot with customized aesthetics
ggplot(total_color_counts, aes(x = BroadColor, y = TotalCount, size = BubbleSize)) +
  geom_point(aes(color = BroadColor)) +
  geom_text(aes(label = BubbleSize, size = 0.8)) +
  labs(title = "Total Counts for Each Broad Colour",
       x = "Broad Colour",
       y = "Total Count",
       size = "Colour Categorisation Count") +  # Add size legend label
  scale_size_continuous(range = c(3, 15), guide = guide_legend(title = "Colour Categorisation Count")) +
  scale_color_manual(values = c("darkgrey","lightblue", "brown","beige", "green", "lightgrey", "lavender"
                              ,"orange","MistyRose","pink", "purple", "red", "Ivory", "yellow")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "none")
```

Figure 45

I began by aggregating the dataset using group_by() based on the "BroadColor" column, calculating the total count of each colour with summarise() while disregarding missing values. After ungrouping, each row represented a colour and its count. For the bubble chart, I customized bubble sizes by creating a vector and assigning them to a new column, "Bubble size", in the data frame. Using ggplot, I plotted the chart: points represented colours, sized by "Bubble size", coloured by "BroadColor", with text overlays showing bubble sizes. I labelled the plot using labs(), defined the size range with scale_size_continuous(), and assigned colours with scale_color_manual(). Applying a minimal theme with theme_minimal(), I further adjusted the appearance, including formatting x-axis text and centring the plot title. Lastly, I removed the legend with theme(legend.position = "none").

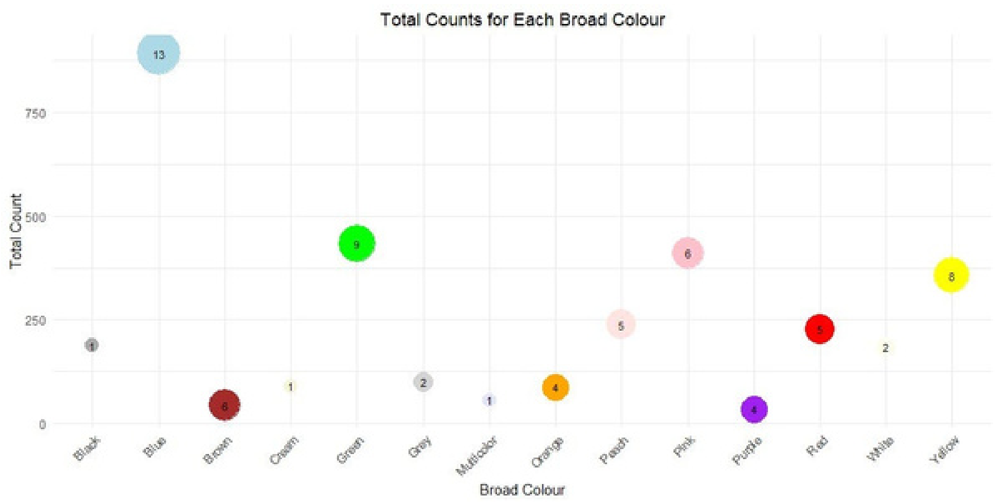

Figure 46

This bubble chart illustrates the most frequent colours, with bubble height representing the frequency and bubble size indicating the number of items in each colour category (e.g., both navy blue and light blue fall under the blue category). From the analysis, a trend emerges: the higher the bubble, the larger the number within it that suggests that the way colours are categorized significantly influences the frequency shown; alternatively, it could be indicative of higher production due to more popularity amongst customers. Thus, a higher frequency for a colour group means there is more variation within that group. For instance, there are more types of blue shades because blue is a popular colour overall.

In conclusion, the company should prioritize improving customer retention rates to boost profits. This includes better inventory management and increased sales during peak months like October and February. Understanding product popularity informs pricing and advertising strategies. Focusing on producing clothing in popular colours like blue, with a variety of shades, can enhance sales.