1. Problem Statements

Problem Statement 1:

Our first problem statement revolves around analysing inventory management on e-commerce platforms by discerning the top products that significantly contribute to revenue. We can distinguish top-performing products by delving into sales data and stock levels across different product attributes and sizes. Through this, we can determine the optimal stocking levels for various product types to avoid overstocking or understocking, enhancing inventory management effectiveness. Key objectives include optimising stock levels, reducing inventory costs, and ensuring product availability. Through insightful analysis, e-commerce platforms aim to streamline inventory processes, improve stock turnover rates, and reduce costs associated with excess or insufficient inventory.

Problem Statement 2:

The problem statement concentrates on maximising revenue generation by strategically leveraging insights derived from sales data and product performance metrics within the e-commerce platform. The company endeavours to identify opportunities for total sales optimisation for each product through meticulous analysis of sales trends, product attributes, and customer preferences. By diving deeper into the nuances of product sales data and understanding the relationship between various product attributes and customer demand, the company seeks to formulate strategies to amplify total sales. This includes identifying product features, pricing strategies, and customer segmentation approaches that can be leveraged to drive revenue growth.

Problem Statement 3:

The problem statement focuses on analysing the time series data to study customer purchasing patterns, enabling businesses to make informed decisions about marketing strategies and overall business planning. By delving into the trends and patterns revealed by analysing purchase amounts and total sales over various time periods, companies can gain valuable insights into customer preferences, seasonal

fluctuations, and potential market trends. This comprehensive understanding allows businesses to anticipate changes in demand, identify growth opportunities, and tailor their marketing efforts more effectively to align promotions and advertisements with anticipated customer needs and preferences.

2. Data Cleaning, Manipulation, and Joining

```
> print(paste("Number of rows are:", nrow(amazon_sale_report)))
[1] "Number of rows are: 128975"
> print(paste("Number of columns are:", ncol(amazon_sale_report)))
[1] "Number of columns are: 24"
```

Figure 1: Dimensions of Amazon Sale Report.csv

In the Amazon Sale Report dataset, based on Figure 1, the total number of observations is 128,975, and the total number of variables is 24.

```
> colSums(sapply(amazon_sale_report, function(x) x == ""))
            index          Order.ID              Date            Status       Fulfilment     Sales.Channel
                0                 0                 0                 0                 0                 0
ship.service.level             Style               SKU          Category              Size              ASIN
                0                 0                 0                 0                 0                 0
    Courier.Status               Qty          currency            Amount         ship.city        ship.state
             6872                 0              7795                 0                33                33
  ship.postal.code     ship.country     promotion.ids               B2B      fulfilled.by     Unnamed..22
                NA                33             49153                 0             89698             49050

> colSums(is.na(amazon_sale_report))
            index          Order.ID              Date            Status       Fulfilment     Sales.Channel
                0                 0                 0                 0                 0                 0
ship.service.level             Style               SKU          Category              Size              ASIN
                0                 0                 0                 0                 0                 0
    Courier.Status               Qty          currency            Amount         ship.city        ship.state
                0                 0                 0              7795                 0                 0
  ship.postal.code     ship.country     promotion.ids               B2B      fulfilled.by     Unnamed..22
               33                 0                 0                 0                 0                 0
```

Figure 2: Missing value counts in all variables from Amazon Sale Report

We used a code that applies ‘function’ to take in the values to check for empty strings. Next, we used the ‘is.na’ function to determine the number of null values in the Amazon Sale Report dataset. As shown in Figure 2 above, there are 9 variables with empty strings in the dataset and 2 variables with null values.

```
> #Replace null values in amount with 0
> amazon_sale_report$Amount <- ifelse(is.na(amazon_sale_report$Amount), 0, amazon_sale_report$Amount)
> colSums(is.na(amazon_sale_report))
             index           Order.ID               Date             Status         Fulfilment      Sales.Channel
                 0                  0                  0                  0                  0                  0
ship.service.level              Style                SKU           Category               Size               ASIN
                 0                  0                  0                  0                  0                  0
    Courier.Status                Qty           currency             Amount          ship.city         ship.state
                 0                  0                  0                  0                  0                  0
  ship.postal.code       ship.country      promotion.ids                B2B       fulfilled.by       Unnamed..22
                33                  0                  0                  0                  0                  0
```

Figure 3: Cleaned "Amount" column

The first data cleaning we did was replacing the null values in the "Amount" column with 0 because it is the column we will work with. Figure 3 above shows the new number of null values in each column.

```
>
> #Format the Date column to Date format
> amazon_sale_report$Date = as.Date(amazon_sale_report$Date, format = "%m-%d-%y")
>
> #Format the Month.Year column to Date format
> amazon_sale_report$Month.Year <- format(amazon_sale_report$Date, "%B %Y")
>
$ Order.ID          : chr  "405-8078784-5731545" "171-9198151-1101146" "404-0687676-7273146" "403
$ Date              : Date, format: "2022-04-30" "2022-04-30" "2022-04-30" "2022-04-30" ...
$ Status            : chr  "Cancelled" "Shipped - Delivered to Buyer" "Shipped" "Cancelled" ...
```

Figure 4: Converted dates data

Next, we converted the dates of the Amazon Sale Report dataset into the date form with the 'as.Date' function, as seen in Figure 4 above.

```
> amazon_sale_report_LJ = left_join(amazon_sale_report,sale_report,by = c("SKU" = "SKU.Code"))
```

index.y	Design.No.	Stock	Category.y	Size.y	Color
8978	SET389	32	SET	S	White
5971	JNE3781	96	KURTA	XXXL	Green
3805	JNE3371	4	KURTA	XL	Light Green
2530	J0341	193	DRESS	L	Blue
5336	JNE3671	6	TUNIC	XXXL	Pink
8242	SET264	81	KURTA SET	XL	Gold
1096	J0095	1	KURTA SET	L	Blue
3945	JNE3405	423	KURTA	S	Pink
NA	*NA*	*NA*	*NA*	*NA*	*NA*
4261	JNE3461	2	KURTA	XXL	Teal
NA	*NA*	*NA*	*NA*	*NA*	*NA*
4463	JNE3500	16	KURTA	XS	Grey

Figure 5: Joined data sets

In order to analyse the colour for the orders placed, we performed the 'left_join' function to join the Sale Report to the Amazon Sale Report by the "SKU.code" and "SKU" columns (these are the common key, just under different column names).

```
> amazon_sale_report_updated = amazon_sale_report %>% filter(Qty != 0)
```

Figure 6: Filtering Orders with 0 quantity

In the Amazon Sale Report, using the filter function, we filtered out the orders with "0" in quantity because we discovered that they were cancelled and unshipped orders, which could indicate that the orders and shipping service did not meet customers' expectations. Thus, we decided to use the order data that are shipped as it would derive more insights into our analysis.

3. Visualizations, Interpretation, and Problem-solving

Problem Statement 1

```
> #Top 20 Styles by Total Sales
> amazon_products_count = amazon_sale_report_updated %>% group_by(Style,Category) %>% summarise(Count =
n()) # Grouping the sales by style and category and counting how many there are
`summarise()` has grouped output by 'Style'. You can override using the `.groups` argument.
> top_products_count <- amazon_products_count %>%
+   top_n(20, Count) # Generating the top 20 styles by count
>
> top_products_count = top_products_count %>%
+   arrange(desc(Count)) # Arranging the top 20 styles in descending order
>
> top_products_style = head(top_products_count, 20) # Taking the top 20 styles by highest order count
>
> top20_styles_category_barplot = ggplot(data = top_products_style, aes(x = Count, y = reorder(Style, C
ount), fill = Category) ) + # Creating a bar plot for the top 20 styles order
+   geom_bar(stat = "identity") +
+   labs(x = "Orders Placed", y = "Style", title = "Top 20 Styles Ordered") + # Labeling and titling th
e bar plot
+   theme_minimal() + # Removing the background
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold")  # Centering and bolding the title
+   )
>
> top20_styles_category_barplot # Printing the results
```

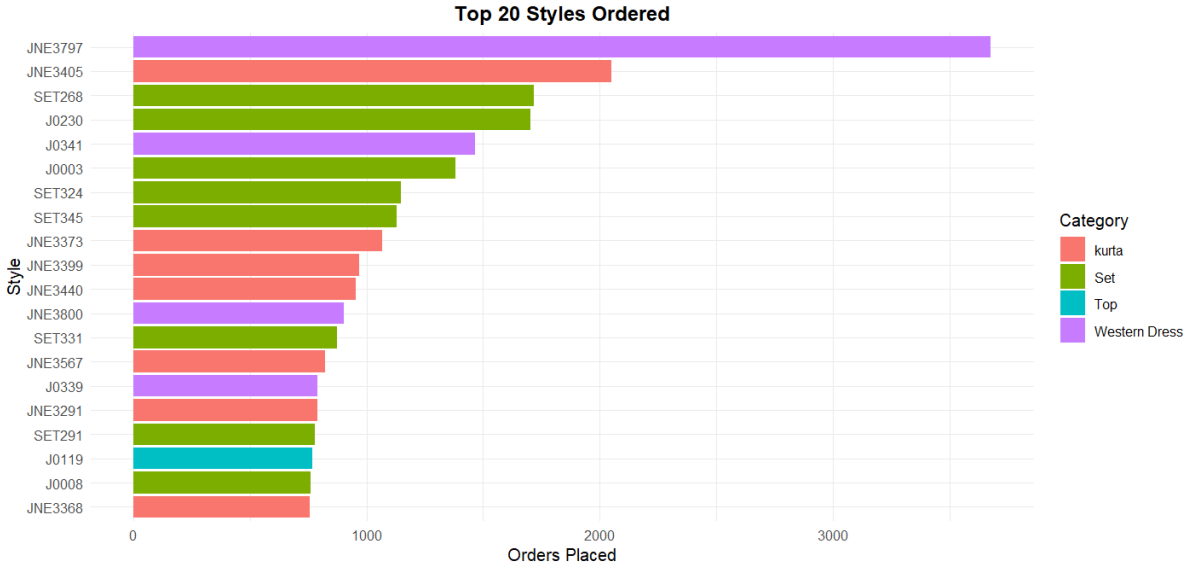

Figure 7: Bar Plot of Top 20 Styles Ordered

Figure 7 is a bar plot displaying the top 20 most ordered styles by their categories. The colours represent the apparel categories. To create the desired data frame for the visualisation above, we first grouped the data set by “Style” and “Category”, and counted the number of observations to get the number of orders placed for each “Style” and “Category”. Next, we take the top 20 observations based on the count (number of orders placed) by arranging the data frame in descending order and taking the first 20 observations from the top. A bar plot is then plotted using ggplot2’s ‘geom_bar’ function.

Based on the plot, the most popular style is “JNE3797”, which is in the category “Western Dress”. It exceeds the other styles by a large order number. The second most popular style is JNE3405, which is a “kurta”. The third most popular style is “SET268”, a “Kurta Set”.

When observing Figure 7, we can see the color green appearing on most of the bars, which indicates that “Set” (kurta set) is the most popular choice among customers for order count. This allows the e-platforms to consider stocking most kurta set as inventory.

```
> #Stock
> #-------------------------------------------------------------------------------------------------------------------
-
> amazon_sale_stock = amazon_sale_report %>% group_by(SKU) %>% summarise(Quantity = sum(Qty)) # Grouping the dataset
by SKU (Product) and finding the total quantities sold
> sale_report_stock = sale_report %>% rename(SKU = SKU.Code) # Renaming the column name for joining purposes
> amazon_sale_stock_LJ = left_join(amazon_sale_stock, sale_report_stock, by = "SKU") # Left join to include the Stoc
k count for each product
>
> top_stocks <- amazon_sale_stock_LJ %>%
+   top_n(20, Quantity) # Finding the top 20 products based on quantities sold
>
> top_stocks_longer = pivot_longer( # Performing pivot longer for stacked bar chart plotting purposes
+   top_stocks,
+   cols = c("Quantity", "Stock"),
+   names_to = "Variable",
+   values_to = "Count"
+ )
> top_stocks_longer$Variable <- factor(top_stocks_longer$Variable, levels = c("Stock", "Quantity")) #Factoring the l
evels to visualize comparison clearer
> top20_stock_quantity = ggplot(top_stocks_longer, aes(x = Count,y = reorder(SKU,Count), fill = Variable)) + # Plott
ing the stacked bar chart to compare product quantities sold and their stock count
+   geom_bar(stat = "identity") +
+   labs(x = "Count", y = "Style", title = "Orders Placed and Stock Count for Top 20 Products")+
+   theme_minimal() + # Removing the background
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold")  # Centering and bolding the title
+   )
>
> top20_stock_quantity
```

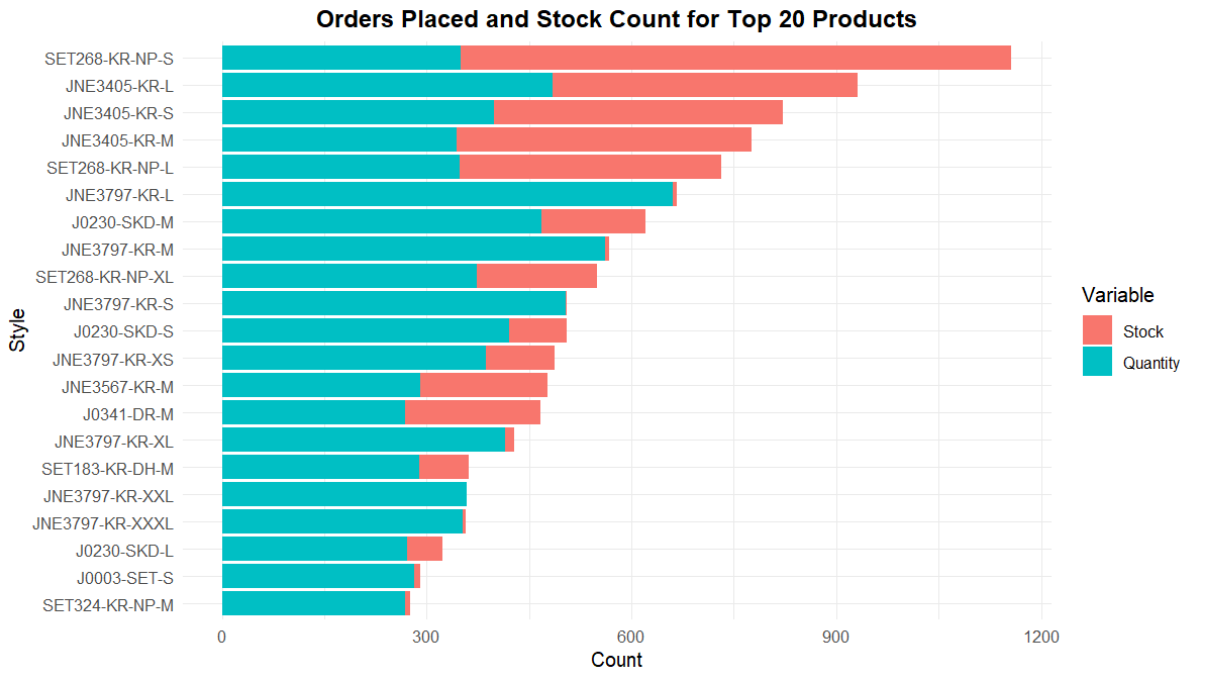

Figure 8: Proportion of Stocks and Orders

Figure 8 showcases the top 20 purchased products compared to the number of stocks the distributors keep. To create the data frame required for the visualization described above, we introduced two new columns: "Quantity," representing the total quantity purchased, and "Stock," indicating the stock count for each product (style, category, and size). Subsequently, we applied the ‘pivot_longer’ function to facilitate the grouping of quantity and stock data for each style, resulting in the desired data frame conducive to plotting the visualisation.

Based on the top 20 most purchased products, some of the stocks are minuscule in proportion to the amount that was purchased. There is also a product that does not have any stocks. This indicates poor inventory management, understocking of inventory. Overstocking leads to higher inventory costs, tying up funds that can be used for other productive purposes. Understocking means not adhering to customers’ demands, reducing customer satisfaction and missing out on potential revenue.

Conclusion:

In conclusion, these analyses provide valuable insights into inventory management. By examining customer preferences regarding the different attributes of our product category, we can identify top-performing products. We can then optimise stocking levels accordingly to improve inventory management effectiveness. Ultimately, through insightful analysis of top-performing products and inventory management practices, these visualizations enable e-commerce platforms to reduce costs associated with excess inventory, thereby addressing the objectives outlined in our first problem statement.

Problem Statement 2

```
> #Problem Statement 2: Revenue
> #-----------------------------------------------------------------------------------------------------------
-----------------------------------------------
> amazon_category_amount = amazon_sale_report_updated %>% group_by(Category) %>% summarise(Total.Sales =
sum(Amount)) # Grouping the data by category and generating the total sales for each category
> category_amount_barplot = ggplot(data = amazon_category_amount, aes(x = Total.Sales, y = reorder(Catego
ry, Total.Sales), fill = Category)) + # Creating a bar plot
+   geom_bar(stat = "identity") +
+   scale_x_continuous(labels = scales::comma) + # Changing the x-axis numerical labels into millions ins
tead of scientific numbers
+   labs(x = "Total Sales", y = "Category", title = "Total Sales by Order Categories") + # Titling and la
beling the plot
+   theme_minimal() + # Removing the background
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold")  # Centering and bolding the title
+   )
>
> category_amount_barplot # Printing the results
```

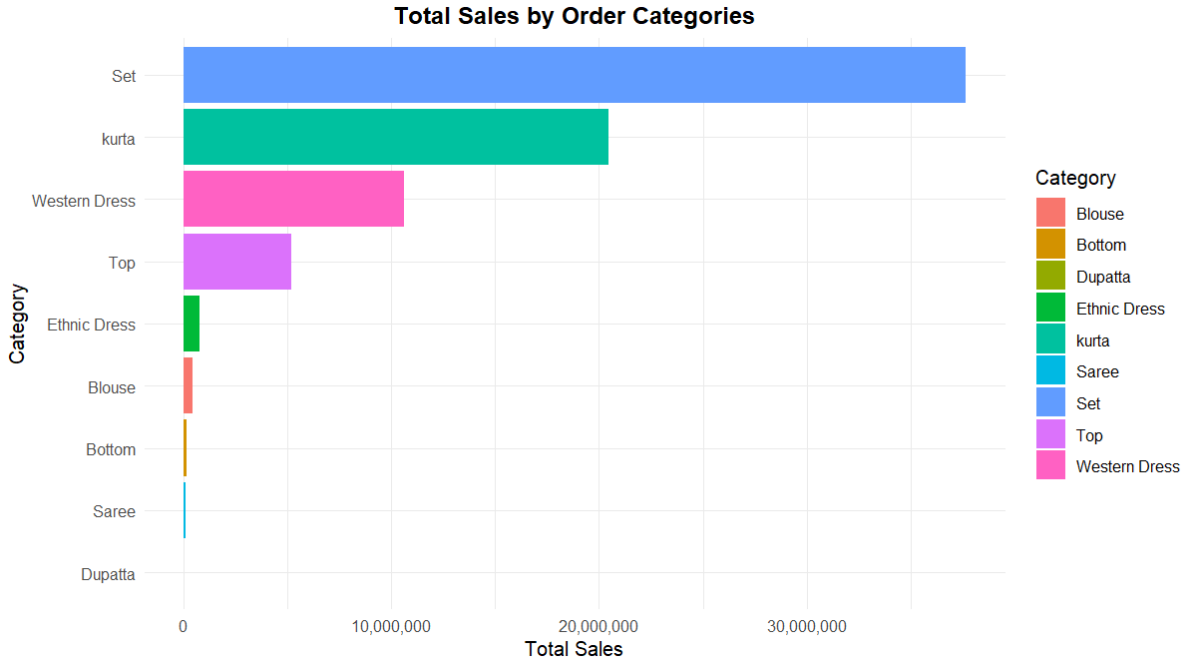

Figure 9: Total Sales by Order Categories

Figure 9 shows the total sales for each category. As seen above, the “Set” (Kurta set) category has the highest total sales, followed by “Kurta” and “Western Dress”. To produce the data frame used to visualize the plot, the original data frame is grouped by “Category”, and the total sales for each category are calculated by summing the “Amount” column. The ggplot2 package is then used to create the bar plot with the ‘geom_bar’ function.

The “Kurta set” has the highest sales as it is a garment that best reflects the Indian culture. It is also a unisex garment that can be worn on any occasion due to its light and breathable material, which is perfect for India’s tropical climate. The garment is also worn during special occasions such as weddings and Diwali. “Kurta sets” are

bought more than the “Kurta” itself due to the sets’ convenience, colour coordination ease, and style. “Western dresses” are bought more often than, say, a “saree” as they offer more comfort and convenience that the working class requires nowadays. For example, ease of walking, in which a “saree” is too restrictive.

In terms of total sales, “Kurta” made only about half the sales the “Set” did. This phenomenon can be attributed to the “Kurta set” being priced higher as it is a full set of clothing, containing more garments, including the “kurta” and bottoms.

```
> #Finding colors based on Total Sales Orders
> amazon_sale_report_LJ = left_join(amazon_sale_report_updated,sale_report,by = c("SKU" = "SKU.Code")) # Joining the
data sets by common key
>
> amazon_sale_report_color = amazon_sale_report_LJ %>% group_by(SKU, Color) %>% summarise(Total.Sales = sum(Amount))
`summarise()` has grouped output by 'SKU'. You can override using the `.groups` argument.
> top_colors =
+   amazon_sale_report_color  %>%
+   top_n(20, Total.Sales) %>% # Picking the top 20 colors by sales
+   arrange(desc(Total.Sales)) # Arranging the data in descending order
>
> top_colors_20 = head(top_colors,20)
>
> custom_colors <- c(
+   "Blue" = "lightblue",
+   "Green" = "lightgreen",
+   "OFF WHITE" = "grey",
+   "Olive Green" = "#BAB86C",
+   "Orange" = "orange"
+   # Add more colors as needed
+ )
> top_colors_barplot <- ggplot(data = top_colors_20, aes(x = Total.Sales, y = reorder(SKU, Total.Sales), fill = Colo
r)) +
+   geom_bar(stat = "identity") +
+   labs(x = "Total Sales",
+        y = "SKU",
+        title = "Top 20 Products and Colors by Total Sales") +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold")  # Centering and bolding the title
+   ) +
+   scale_fill_manual(values = custom_colors) # Manually assigning colors
>
> # Printing the plot
> print(top_colors_barplot)
```

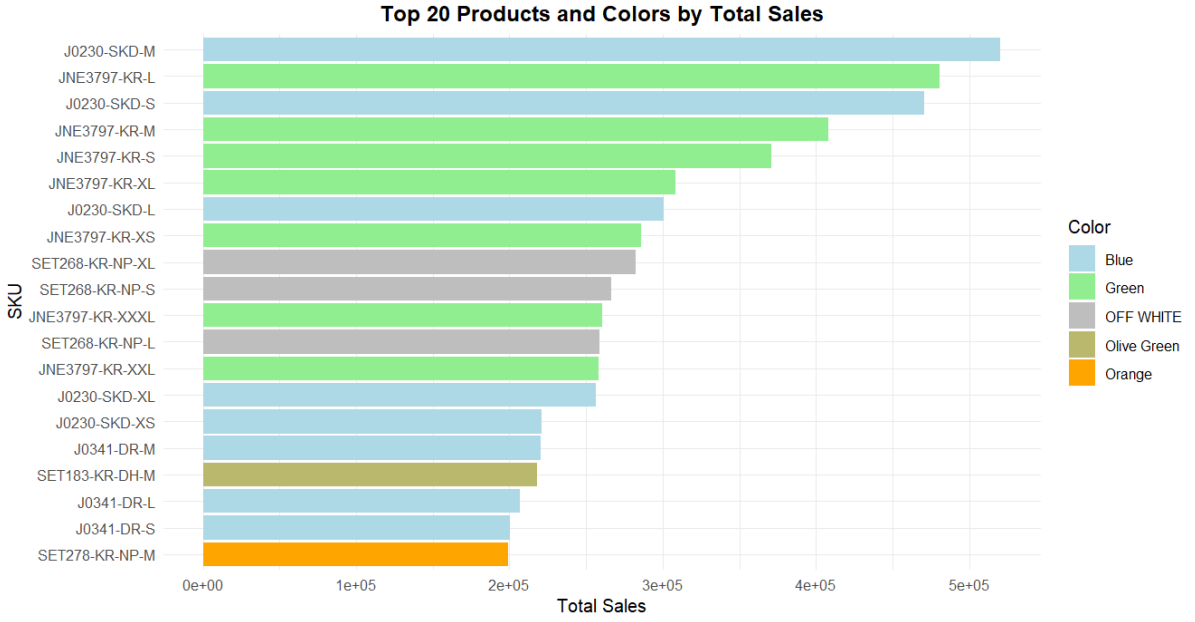

Figure 10: Top 20 Products and Colours by Total Sales

Figure 10 is a bar plot that shows the top 20 products (by style, size, and colour) sold with the highest sales. The bars are colour-coded according to the most popular colours purchased, as indicated in the legend. To produce the data frame for this visualization, the 'Amazon Sale Dataset' is left-joined with the “sales_report” dataset to obtain the “color” column for all ordered products. The resulting data frame is then grouped by “SKU” (product) and “Color”. Next, the total sales for each product and colour are calculated by summing the “Amount”. The top 20 products are identified by first arranging the data frame in descending order of total sales and selecting the first 20 observations. The ggplot2 package is used to create the bar plot with the ‘geom_bar’ function, and the bar colours are manually specified using the ‘scale_fill_manual’ function.

Based on the plot, we can deduce that the product “J0230-SKD” of size “M” with the colour "Blue" generates the most sales. The second highest sales-generating product is “JNE3797-KR-L” in “Green”, followed by another “Blue” product. By analysing the graph, we can infer customers’ top preferences in product colour is “Blue” and “Green”.

```
> #Promotion
> amazon_promotion = amazon_sale_report_updated %>%
+   mutate(Promotion = ifelse(promotion.ids == "", "No Promotion", "Promotion Applied")) # Creating a new co
lumn for sales with and without promotion using the if else function
> amazon_promotion_sales = amazon_promotion %>% group_by(Style, Category, Promotion) %>% summarise(Total.Sal
es = sum(Amount)) # Grouping the sales by Style, Category, Promotion and counting the total sales
`summarise()` has grouped output by 'Style', 'Category'. You can
override using the `.groups` argument.
>
> top_style_sales_promotion <- amazon_promotion_sales  %>%
+   top_n(20, Total.Sales) #Picking the top 20 styles according to sales
>
> top_style_sales_promotion = top_style_sales_promotion %>%
+   arrange(desc(Total.Sales)) #Arranging the data in descending order
>
> top_products_style_promotion = head(top_style_sales_promotion, 20) #Extracting the first 20 values from th
e dataframe
>
> top20_styles_category_barplot_promotion = ggplot(data = top_products_style_promotion, aes(x = Total.Sales,
y = reorder(Style, Total.Sales), fill = Promotion) ) + # Creating a bar plot of the total sales for styles,
where the color represent if the promotion is applied
+   geom_bar(stat = "identity") + # Length of bars represent the total sales
+   labs(x = "Total Sales", y ="Styles", title = "Top 20 Styles from Total Sales with Promotion") + # Titlin
g and labeling the plot
+   theme_minimal() + # Removing the background
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold")  # Centering and bolding the title
+   )
>
> top20_styles_category_barplot_promotion # Printing the results
```

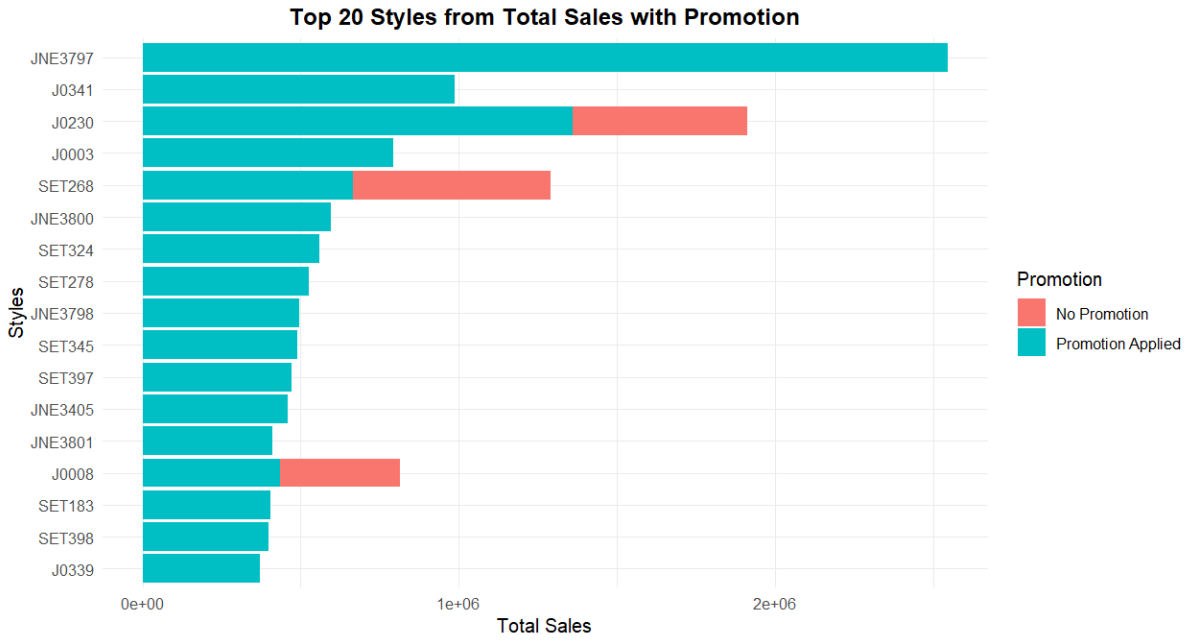

Figure 11: Top 20 Products with Promotion

Figure 11 is a bar plot that shows the top 20 products based on their total sales generated with promotions. The data frame used to visualize the plot is created by adding a new column with an ‘ifelse’ statement to indicate whether or not a product has a promotion applied. Then, the data frame is grouped by “Style”, “Category”, and “Promotion”, and the total sales are calculated by summing the “Amount”. The top 20 styles are obtained using the ‘top_n’ function on the “Total.Sales” column, extracting the first 20 observations from the resulting data frame. The ggplot2 package is used

to plot the resulting stacked bar chart with the ‘geom_bar’ function. The colour on the bars indicates the proportion of sales attributed to promotions for each product.

From the plot, we can ascertain that all the top-selling products have a significant portion of their sales driven by promotions, highlighting the effectiveness of promotional strategies in boosting sales.

```
> mrp_columns <- c("Ajio.MRP", "Amazon.MRP", "Amazon.FBA.MRP", "Flipkart.MRP",
+                  "Limeroad.MRP", "Myntra.MRP", "Paytm.MRP", "Snapdeal.MRP")
> may_2022[mrp_columns] <- lapply(may_2022[mrp_columns], as.integer)
> may_2022$TP = as.numeric(may_2022$TP)
> # Loop through each MRP column and calculate the markup
> for (mrp_col in mrp_columns) {
+   # Calculate markup and create a new column
+   may_2022[paste0(mrp_col, "_Markup")] <- ((may_2022[[mrp_col]] - may_2022$TP) / may_2022$TP) * 100
+ }
>
>
> markup_columns <- c("Ajio.MRP_Markup", "Amazon.MRP_Markup", "Amazon.FBA.MRP_Markup",
+                     "Flipkart.MRP_Markup", "Limeroad.MRP_Markup", "Myntra.MRP_Markup",
+                     "Paytm.MRP_Markup", "Snapdeal.MRP_Markup")
>
> # Create the Platform_Markup dataframe
> Platform_Markup <- may_2022[, c("Sku", "Style.Id",markup_columns)]
>
> # Assuming Platform_Markup is your dataframe
> platform_markup_long <- pivot_longer(
+   data = Platform_Markup,
+   cols= c("Ajio.MRP_Markup", "Amazon.MRP_Markup", "Amazon.FBA.MRP_Markup",
+           "Flipkart.MRP_Markup", "Limeroad.MRP_Markup", "Myntra.MRP_Markup",
+           "Paytm.MRP_Markup", "Snapdeal.MRP_Markup"),
+   names_to = "Platform",
+   values_to = "Markup"
+ )
> # Convert Markup column to numeric
> platform_markup_long$Markup <- as.numeric(platform_markup_long$Markup)
>
> # Remove rows with NA values in Markup column
> platform_markup_long <- platform_markup_long[!is.na(platform_markup_long$Markup), ]
>
> # Calculate average markup for each platform
> platform_markup_average <- platform_markup_long %>%
+   group_by(Platform) %>%
+   summarise(Average = mean(Markup, na.rm = TRUE))
> platform_markup_average <- platform_markup_average %>%
+   arrange(desc(Average))
> # View the resulting dataframe
```

```
> # View the resulting dataframe
> platform_markup_average
# A tibble: 8 × 2
  Platform              Average
  <chr>                   <dbl>
1 Amazon.FBA.MRP_Markup    313.
2 Amazon.MRP_Markup        313.
3 Limeroad.MRP_Markup      312.
4 Flipkart.MRP_Markup      312.
5 Ajio.MRP_Markup          311.
6 Snapdeal.MRP_Markup      311.
7 Paytm.MRP_Markup         311.
8 Myntra.MRP_Markup        309.
```

Figure 12: Average Mark-up Percentage Across Platforms

To prepare the data, we first converted the maximum retail price (“MRP”) values from strings into integers. Afterwards, we created new columns for each platform’s “MRP” so we could calculate the mark-up percentage. We used the formula “MRP” minus the trade price (“TP”), then divided it by “TP” and multiplied it by 100 to get the results in percentage form.

To derive insights from the mark-up percentage, we created a new data frame containing the “SKU”, “Style.Id”, and the mark-up percentage we calculated. We converted the wide data frame into a long one using the “pivot_longer” function. Next, by applying the ‘group_by’ function, we grouped the mark-up percentages by the different platforms and generated the average mark-up for each platform.

The dataset provides insights into the average MRP Markup values across various e-commerce platforms. The output displays a narrow distribution, with the average MRP mark-up ranging from 309% to 313%. This suggests relatively consistent pricing strategies across platforms. Amazon stands out with the highest average MRP Markup, possibly attributed to its fulfilment services, while Myntra lags slightly behind with the lowest percentage.

Based on the insights, the platforms can tailor their pricing strategies in terms of their mark-up margin to leverage platform-specific dynamics, potentially enhancing competitiveness and profitability. The higher the mark-up percentage from the original price (“TP”), in which the products are priced greater than the original manufacturers’ price, the higher the profit potential is. However, there are risks of customers choosing competitors’ products instead.

Conclusion:

As a whole, the analyses conducted on sales trends, product attributes, and customer preferences provided valuable insights into maximizing revenue generation on the e-commerce platform, catering to our second problem statement. From the insights of the products that generate the most sales (category, product, and colour), and having proof that promotions are effective in boosting sales, we can formulate promotional strategies according to customer preferences to allow the businesses to capitalize on opportunities for total sales optimization for each product category.

Furthermore, using the mark-up percentage information, the platforms can derive whether the pricing strategies are effective in yielding sales and compare it with their competitors (other platforms). As a result, they can achieve the objective of maximizing revenue generation on the e-commerce platform.

Problem Statement 3

```
> amazon_date_amount = amazon_sale_report_updated %>% group_by(Date) %>% summarise(Total.Sales = sum(Amount))
# Grouping by date, and finding the total amount of sales
>
> time_series_total_sales = ggplot(data = amazon_date_amount, aes(x = Date, y = Total.Sales)) + # Creating a
line plot for the total amount of sales
+   geom_line(color = "red") +
+   geom_point(color = "red") +
+   labs(x= "Month", y = "Total Sales", title = "Time Series of Order Revenue in Amazon")+
+   theme_minimal() + # Removing the background
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold")  # Centering and bolding the title
+   )
>
> time_series_total_sales
```

Figure 13: Time Series of Order Revenue in Amazon

Figure 13 is a line plot that illustrates the trajectory of Amazon's total sales from April to July 2022. First of all, the sales data set was grouped according to the dates, and then the total sales of each date was generated. This new data frame is used to create the plot with the ggplot2 package. Using 'geom_point', the points are plotted based on the dates the total sales were generated (x-axis) and the sales amount (y-

axis). From there on, the ‘geom_line’ function was then used to connect the points to create a line chart.

This line plot offers valuable insights into the fluctuations in total sales generated over this period. It enables businesses to dive deeper into the dynamics of revenue growth and identify potential patterns or trends influencing sales performance. By leveraging these insights, businesses can also strategically adjust their operations and marketing efforts to maximize revenue, capitalize on peak periods of consumer purchasing activities, and adapt to changing market dynamics.

As seen in Figure 13, sales were high from April to early May. However, sales dropped thereafter, only increasing slightly in early June before dropping off again. This negative trend can be explained by clothes collections. When clothes collections, in this case a summer collection, are released, consumers will likely purchase those garment pieces during the first few months of the season. However, as time passes, interest in those collections diminishes.

```
> #Total Sales over time for each category
> amazon_category_sales = amazon_sale_report_updated %>% group_by(Date, Category) %>% summarise(Total.Sal
es = sum(Amount)) # Grouping the sales dataset by Date and Category, and finding the total sales
`summarise()` has grouped output by 'Date'. You can override using the `.groups` argument.
>
> category_amount_areaplot = ggplot(data = amazon_category_sales, aes(x = Date, y = Total.Sales, color =
Category)) +  # Creating an area plot for total sales over time for each categories
+   geom_line() +
+   labs(x = "Month", y="Total Sales", title = "Total Sales over time for each Category") +
+   theme_minimal() + # Removing the background
+   theme(
+     plot.title = element_text(hjust = 0.5, face = "bold")  # Centering and bolding the title
+   ) +
+   scale_y_continuous(labels = scales::comma)
>
> category_amount_areaplot # Printing the resulting dataframe
```

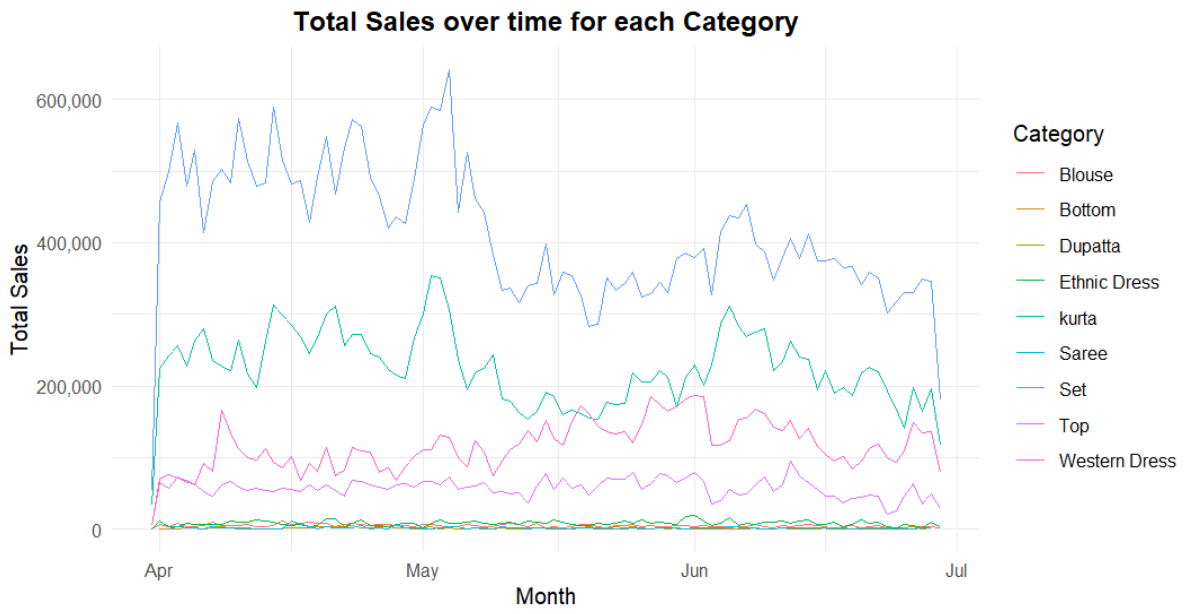

Figure 14: Total Sales of Each Category over time

Figure 14 is a line plot that illustrates the sales trends of various categories over time based on their total sales. The data frame used to visualize the plot is created by grouping the Amazon Sale Report by the “Date” and “Category” columns, and the total sales are calculated by summing the “Amount”. The ggplot2 package is used to plot the line plot with the ‘geom_line’ function, with the lines representing the sales trend of each category. The plot utilizes different coloured lines to represent each category, providing a clearer view of the progression of each category’s sales over time, enabling easy comparison of their total sales performance.

From the plot, we can observe the relative growth or decline of each category’s total sales over a specific month, providing valuable insights into time-specific market trends and consumer purchase behaviour.

Based on the line chart, the “Set” (Kurta Set) and “Kurta” sales peaked in April to May and in early June. Although most of the Indian population wears kurtas on a daily basis, these three apparel categories are commonly used during special events in India, such as weddings and Diwali. The peak in April and May can be explained by the trend in India, where the most popular wedding months include April and May. Since weddings require guests to fit a certain colour scheme, Kurta and Kurta sets are bought to fulfil that wish.

Another possible reason Kurta and Kurta Set have high sales during this period is due to it being the summer season. India is at its hottest during this season, so the Kurta set and Kurta are popular as it is light and breathable.

As for the “Top” and “Western Dress”, even though the total sales proportion is low compared to the Indian apparels, the data peaked in June, which can be attributed to the summer months when these apparels are most suitable for use.

Conclusion:

In conclusion, the analyses conducted on time series data have provided valuable insights into analysing trends of customer purchasing patterns, resolving the problem statement. By delving into trends and patterns of purchase amounts and total sales over various time periods, the platforms can gain a comprehensive understanding of customer preferences, potential seasonal fluctuations, and market trends. This enables businesses to gain valuable insights into customer purchasing patterns, allowing them to respond to demand shifts and strategize their approach in a time-specific manner.

(Total: 2,544 words)