Optimizing Amazon Apparel Sales

MSDA 3999: Capstone Practicum

Field Project

Author: Pranjal Pramod Bhagat

Advisor: Prof. Khald Aboalayon

Master's in Data Analytics

Clark University
December 15, 2023

TABLE OF CONTENTS

TABLE OF FIGURES

DECLARATION

I hereby declare that this project entitled Optimizing Amazon Apparel Sales is entirely my own work and it has never been submitted nor is it currently being submitted for any other degree.

Signed: PRANJAL PRAMOD BHAGAT

Date: December 15, 2023

ACKNOWLEDGEMENTS

I extend my sincere gratitude to Professor Khald Aboalayon for his invaluable guidance and mentorship throughout the project, which greatly contributed to its success. I would like to express my appreciation to Professor Kelly Giardullo for imparting valuable skills in the art of storytelling with data during the course. A special thanks to Professor Shasha Yu, whose expertise in Tableau dashboarding significantly aided me in creating dynamic and impactful visualizations, enhancing the overall presentation of my work. My gratitude also extends to my peers for their collaborative spirit, providing solutions to some of my queries and enriching the learning experience. Lastly, I acknowledge and thank Sharma A. (2022) for sharing the E-commerce sales dataset on Kaggle, which served as the foundation for my research.

ABSTRACT

Project Objectives:

The project, "Optimizing Amazon Apparel Sales: Data Analysis for Business Growth and Efficiency," aims to analyse e-commerce data related to apparel sales on Amazon and other platforms. The main goal is to understand the relationship between apparel item price and different apparel features. Followed by comparing Amazon's performance with other e-commerce sales channels and investigating business trends for apparel type in the international market.

Project Significance:

The research is important in filling gaps in our understanding of e-commerce, especially apparel sales.

The research conducted is crucial for business because of the following reasons:

- It guides business in smart inventory decision
- It offers broad market insights not just limited to Amazon
- It sets the stage for future innovation in the dynamic world of e-commerce

Project Methods:

The project utilizes various datasets, including Amazon Sale Report, Sale Report, March 2021, May 2022, and International Sale Report, collected from Kaggle.

The methodology involves the following steps:

Data Exploration: Checked dataset characteristics, information, and descriptions.

Data Cleaning: Performed missing value check, outlier analysis, and data validation.

Data Analysis: Created dashboards on Amazon India Apparel Sales Overview, Inventory and Sales Channel Analysis and International Apparel Sales.

Feature Engineering: Transformed numeric data types to date and categorical type based on the feature significance.

Variable Transformation: Converted categorical columns to numeric data type using Binary Mapping, One-Hot Encoding and Label Encoding.

Multi-collinearity check: Checked the multi-collinearity between independent variables and dropped each variable having high collinearity separately.

Feature Scaling: Adjusted the scale of the features to ensure they contribute equally to the analysis.

Model Building: Built models as below for predicting the target variable “Amount” using the predictor variables.

1. Linear Regression
2. Ridge Regression
3. Lasso Regression
4. Decision Tree Regressor
5. Random Forest Regressor

Model Evaluation: Models are evaluated based on the three metrics namely Mean Squared Error (MSE), Mean Absolute Error (MAE) and R squared.

Tools used for the project include MS Excel, Visual Studio Code, Tableau, and Jupyter notebook.

Programming Language: Python

Key Findings: Important details about Amazon India's clothing sales are revealed by the study. Sales are largely driven by top-selling categories including "Kurta," "Top Bottom Set," and "Western Dress." The complexities of consumer behavior are highlighted by preferences for Small to Extra-Large sizes, expedited shipment, and significant order patterns in multiple categories. By identifying important variables influencing the target variable, the Random Forest Model offers predictive significance.

Conclusion: To conclude, this research explores the dynamics of Amazon India's clothing sales and provides important insights for tactical improvement. Making well-informed decisions starts with identifying sales trends, evaluating performance against rivals, and comprehending how factors and pricing impact. These observations, together with useful suggestions, support Amazon India's e-commerce strategy by highlighting its applicability in the dynamic environment.

Keywords: E-commerce; Sales Optimization; Data Analysis; Amazon; Inventory Optimization

CHAPTER ONE INTRODUCTION

1.1 Background

The project emphasis on Optimizing the apparel sales of Amazon India by improving efficiency, increasing revenue, and making informed decisions based on data-driven insights. In the ever-evolving landscape of E-commerce, the goal is to decipher the complexities of Amazon India, apparel sales data. Focused on enhancing business growth and efficiency, this project delves into the complexities of apparel item prices and their dependencies on various influencing factors.

Statement of the Problem

In the Amazon Sales Report, we have spotted price fluctuations in apparel item due to various factors. The challenge is to decipher relationship between these factors and apparel price. The goal is to build a machine learning model that predicts the item amount.

1.2 Purpose and Significance

The project, "Optimizing Amazon Apparel Sales: Data Analysis for Business Growth and Efficiency," aims to analyse e-commerce data related to apparel sales on Amazon and other platforms.

The project's main objectives are as follows:

- Investigate the relationship between apparel item price and features like type, fulfilment, shipping, and size
- Identify sales trend based on month, category, shipping method, customer segment and region
- Compare Amazon's performance with other e-commerce sales channels – Ajio, Flipkart and Myntra
- Understand business trends for apparel type in the international market

The research is important in filling gaps in our understanding of e-commerce, especially apparel sales. We are driven by a commitment to make decision backed by data with our goal to give business a competitive edge with strategic insights. Ultimately, we are not just analyzing for ourselves we are contributing to the entire e-commerce industry.

Additionally, the research can contribute to the following:

- It guides business in smart inventory decision
- It offers broad market insights not just limited to Amazon
- It sets the stage for future innovation in the dynamic world of e-commerce

CHAPTER TWO LITERATURE REVIEW

(Khanna P. & Sampat B., 2015) Factors Influencing Online Shopping During Diwali Festival 2014: Case Study of Flipkart and Amazon.

With a special emphasis on the Diwali celebration, this study illuminates the variables influencing online shopping behaviour throughout celebratory seasons. Their research highlights the shifting attitudes of urban consumers and the growing possibility of online purchasing in India during holiday seasons, which aligns with our findings. We can highlight the critical role that festivals play in influencing customer behaviour and contributing to the growth of e-commerce sales in the Indian retail industry by examining their findings and drawing comparisons with our research.

(Nasabi A. & Sujaya H., 2022) Consumer Buying Behaviour Trends of ECommerce in India - A Case Study.

The findings of this study, which examines changes in consumer purchasing behaviour in the Indian e-commerce industry, are consistent with our conclusion, which highlights the benefits of e-commerce in influencing corporate operations. Their case study observations may be connected to our research on regional preferences and top-selling stock keeping units (SKUs). Through an examination of their study, we can make comparisons between the simplicity of use and scalability provided by e-commerce, demonstrating how the flexibility of

advertising spending and ease of purchase correspond with the trends in consumer buying behaviour found in the Indian market.

(Yadav N. & Sagar M., 2018) Amazon India’s 'Apni Dukaan': Branding Strategy.

The analysis of Amazon India's branding, as seen in advertisements like "Try to kar"(Try it) and "Apni Dukaan"(Our store) is consistent with our findings on popular categories and their impact on overall sales. Their thorough analysis of Amazon's branding, which places a strong emphasis on understanding consumer behaviour and cultivating a "trusted, reliable, and local" brand identity, aligns with our research on the most popular Style Ids/ SKUs and the value of customer trust. Through an examination of their findings, we can make connections between Amazon's unique branding tactics and the noted performance of product categories as well as their influence on the sales environment.

(Chauhan R. & Shah S., 2020) A Research Study on Logistic and E-Commerce of Amazon and Flipkart.

This study which examined the logistics and e-commerce operations of Amazon and Flipkart, is consistent with our findings on inventory control and order quantities. Their focus on the difficulties and prospects facing the logistics sector is in line with our analysis of the sales statistics from Amazon India. We can make links between their insights into how service providers might meet the needs of manufacturers and end users and the tactics that Amazon uses to effectively manage its supply chain. Examining their results offers insightful background for comprehending the subtleties of e-commerce and logistics

operations in the Indian market, illuminating the ways in which these factors affect customer happiness and sales.

CHAPTER THREE METHODS

3.1 Research approach

The main objective is to monitor the fluctuating trend of apparel item amount with respect to different features of the apparel item. In order to acknowledge the research objective, we follow a research approach which first analyses the key metrics followed by building a machine learning model to predict the apparel item amount using the predictor variables.

3.2 Techniques and Procedure

3.2.1 Dataset Description

- Primary dataset:
    - Amazon Sales Report with 128,851 records from Amazon India, offering insights into SKU code, apparel types, order details shipping information from March 31, 2022, to June 29, 2022.
- Additional datasets:
    - Sales Report which focuses on SKU code, apparel types and stock details
    - March 2021 and May 2022 with price of each SKU code across various sales channel

- International Sales Report which provides information on SKU codes, sales data, customers, and gross amount from June 2021 to March 2022

3.2.2 Data Collection

The data collection process involves acquiring datasets from Kaggle, ensuring data integrity and completeness. Since the datasets are already available, there is no need for additional data collection efforts.

3.2.3 Data Processing

To process the data, I have utilized several libraries of Python, including Pandas, NumPy, Seaborn, Matplotlib and Scikit-Learn.

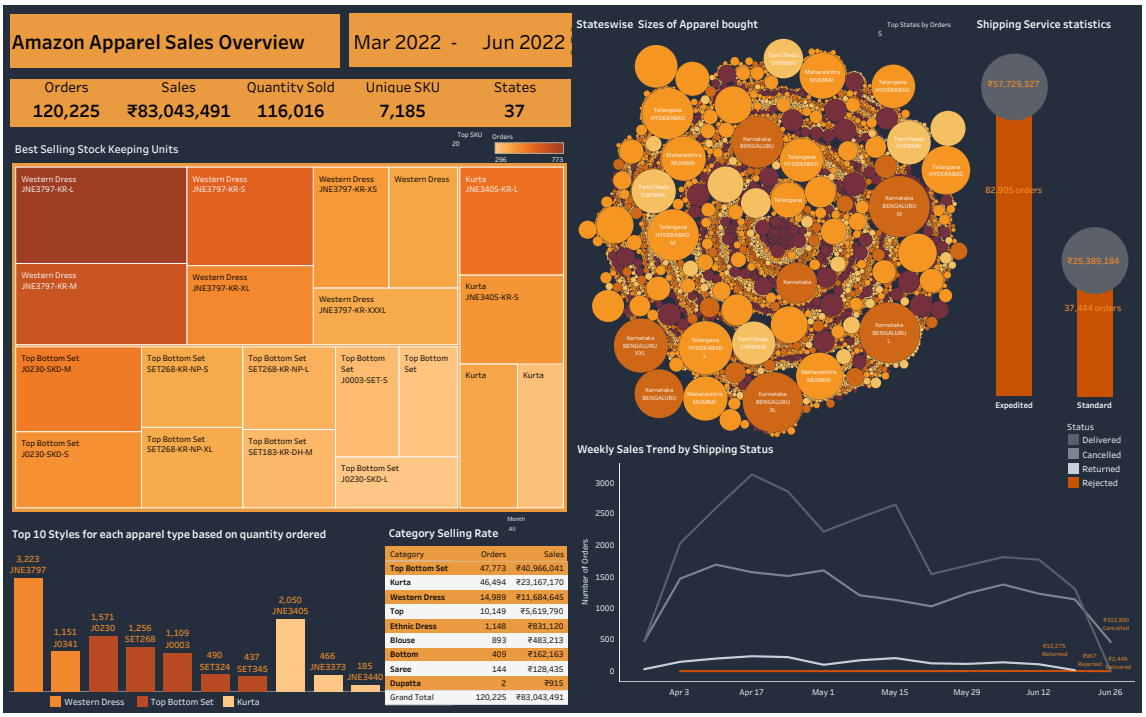
Data Exploration: Checked dataset characteristics, information, and descriptions.

Data Cleaning: Performed missing value check, outlier analysis, and data validation.

- Missing values treatment: Dropped columns with more than 60% null values, removed rows for column with less than 1% null values, and imputed all other null values
- Data Manipulation: Changed the data type to date and object for categorical features
- Checked for unique categories in each object type column to understand the column significance

- Outlier Analysis: Truncated the outliers and included data till 99th percentile

3.2.4 Exploratory Data Analysis

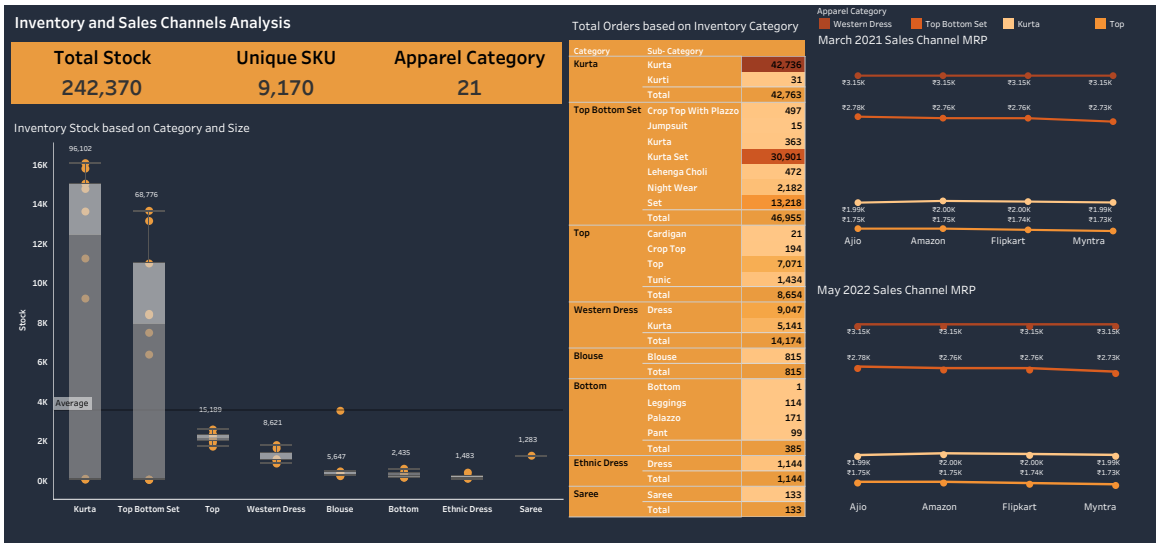


https://public.tableau.com/app/profile/pranjal.pramod.bhagat4432/viz/AmazonIndiaApparelSalesDashboard/AmazonApparelSales

- Best Selling Stock Keeping Units are from apparel category “Western Dress”, “Top Bottom Set” and “Kurta” and these categories contribute to maximum sales made by Amazon India

- In top five states based on orders placed, size “Small”, “Medium”, “Large” and “Extra Large” are in demand

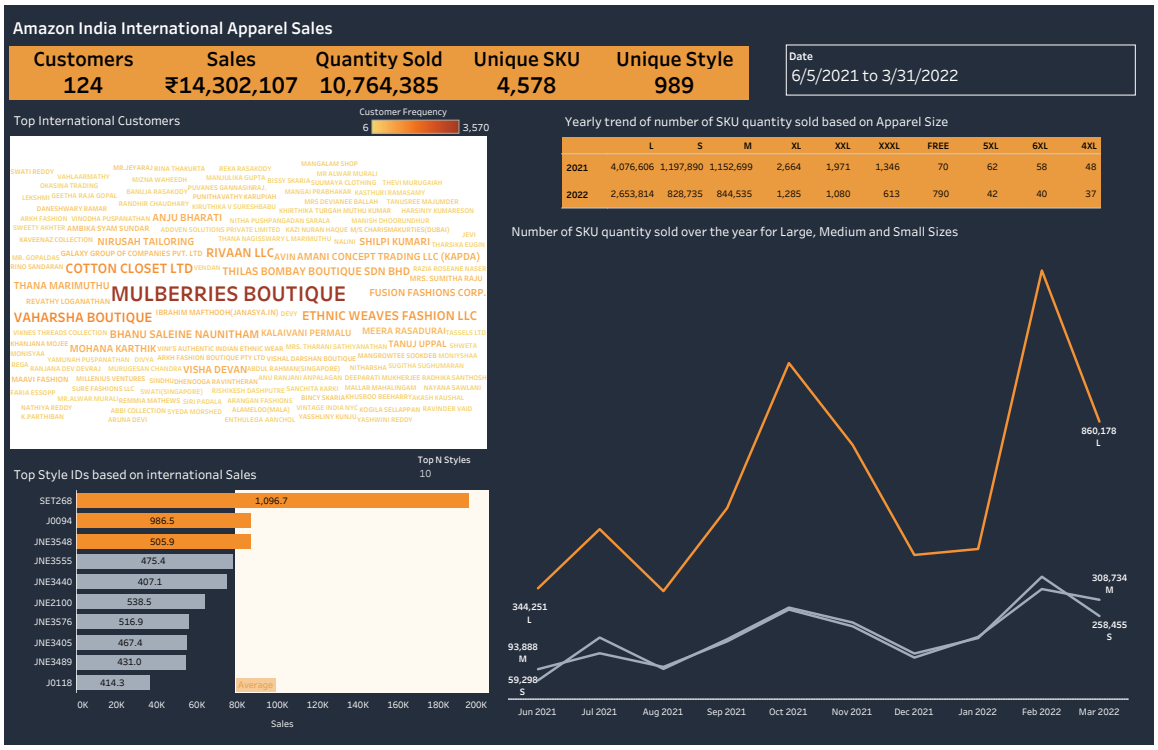
- Even though shipping service “Expedited” leads customers to spend more, number of customers preferring faster shipping is 37% more than customer preferring standard shipping
- If we look at the category wise orders, 10K orders were placed for “Top” category, however only 409 orders were placed for “Bottom” category
- Even though number of orders with “Returned” and “Rejected” status are lower than “Shipped”, we can see reasonable number of orders were “Cancelled”



https://public.tableau.com/app/profile/pranjal.pramod.bhagat4432/viz/InventorySalesChannelAnalysisDashboard/InventorySalesChannels

- For apparel category “Kurta” and “Top Bottom Set” there were 42K and 46K orders respectively, and the inventory had 96K and 68K items in stock respectively

- However, apparel category “Western Dress” there were 14K orders, and the inventory had 8K items in stock
- If we look at the average Maximum Retail Price (MRP) for different categories across various Sales channels we can see there is not much difference
- If we analyze the MRP trend for March 2021 and May 2022 it seems to be similar across all the sales channel



https://public.tableau.com/app/profile/pranjal.pramod.bhagat4432/viz/AmazonInternationalDashboard/International

- The top international customer “Mulberries Boutique” has placed 3570 orders and it seems to be a private business rather than individual customer
- The other top customers include “Vaharsha Boutique”, “Cotton Closet LTD” and “Ethnic Weaves Fashion LLC” with more than 1K orders
- If we observe for both the years 2021 and 2022 orders, size “Small”, “Medium”, “Large” and “Extra Large” are in demand
- International Orders for the in-demand size categories are at peak before the festive seasons in India

3.2.5 Data Pre-processing

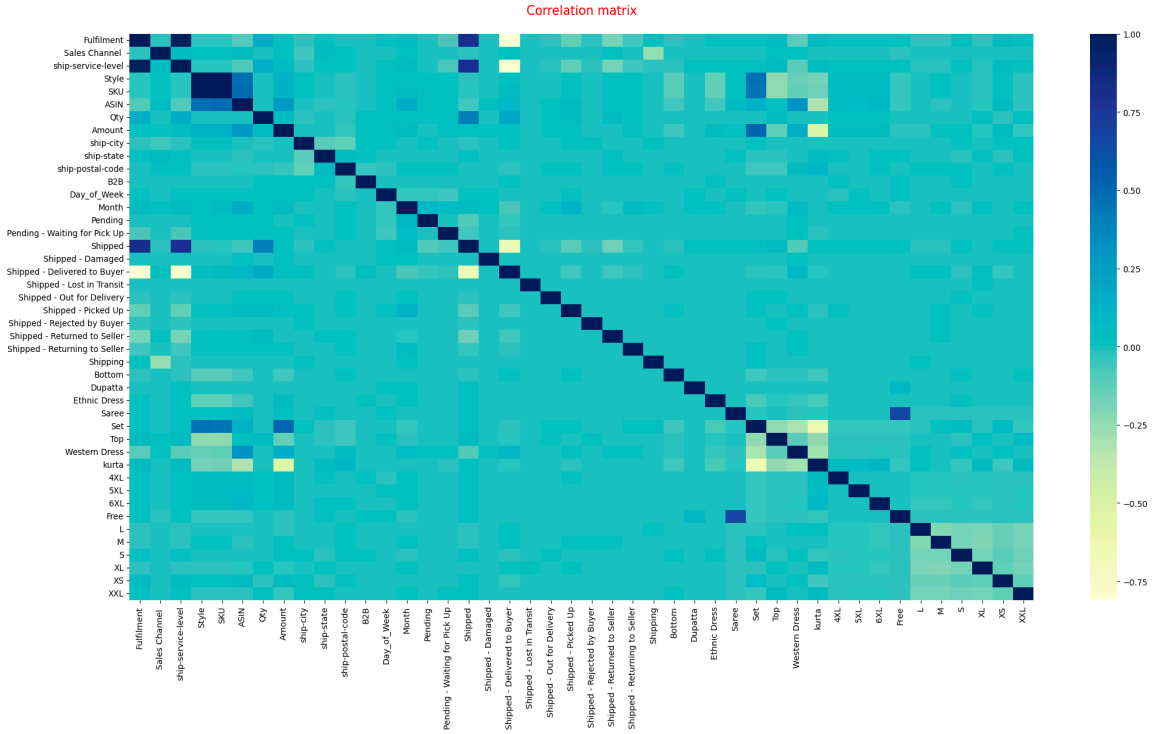
Feature Engineering: Transformed Date field to Day of Week and Month of Week

Variable Transformation: Converted categorical columns to numeric data type using:

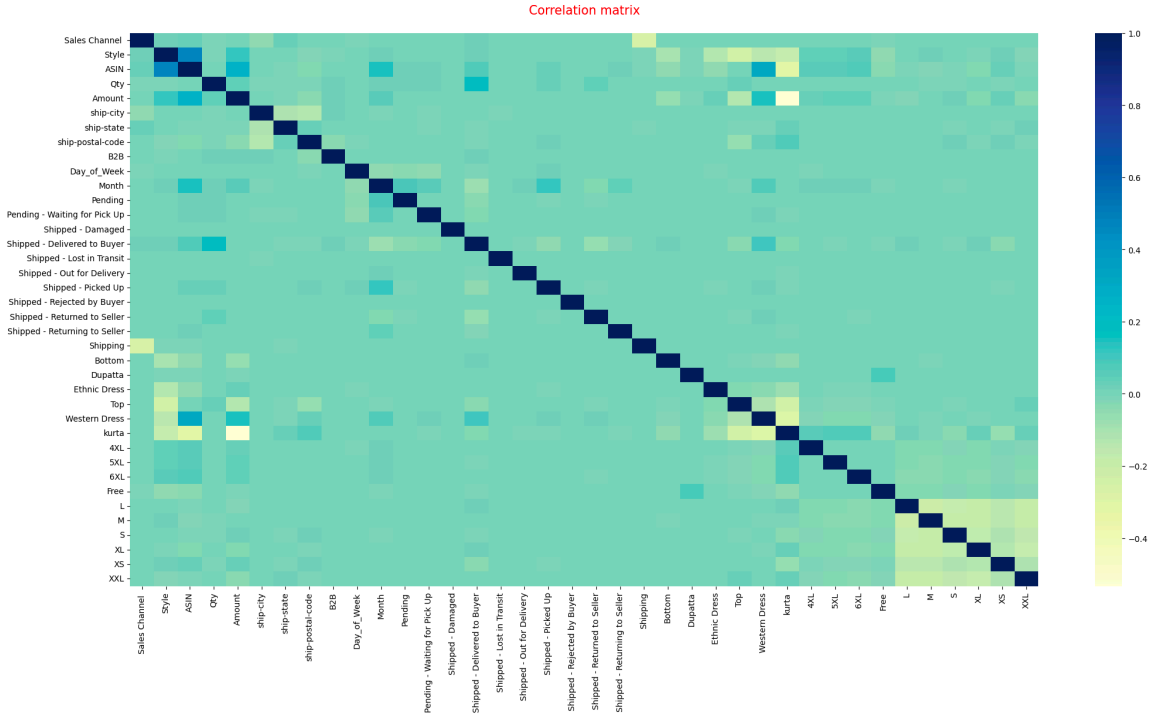
- Binary Mapping: It represents two categories as 0 and 1.
- One-Hot Encoding: It creates binary columns for each category, assigning 1 to presence of category and 0 otherwise.
- Label Encoding: It assigns unique numerical label to each category.

Multi-collinearity check: Where two or more predictor variables are highly correlated, making it challenging to discern the effect of each variable on dependent variable. Checked the multi-collinearity between independent variables and dropped each variable having high collinearity with other predictor variables separately.

Before Multi-collinearity check:



After Multi-collinearity check:



Feature Scaling: Adjusted the scale of the features to ensure they contribute equally to the analysis.

Train Test Split: Data was split into training and testing sets (70% training and 30% testing).

3.2.6 Implementation

The final aim to build a machine learning model that learns the patterns of predictor variable which are the apparel features and predicts apparel item amount. I started by assuming a linear relationship between target and predictor variables.

Linear Regression models the relationship between variables by fitting a straight line to the observed data points, allowing prediction of the dependent variables based on the values of independent variables.

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Linear Regression with Standard Scalar	40967.33	144.56	0.42	41004.91	144.49	0.43
Linear Regression with MinMax Scalar	40967.33	144.56	0.42	41004.91	144.49	0.43
Linear Regression with Cross Validation	40961.55	144.52	0.426	41215.99	145.22	0.421

Implementing more robust models which can overcome overfitting issue:

Ridge Regression: Regularization technique to prevent overfitting by adding a penalty term based on the sum of squared coefficient in addition to minimizing the least squares loss.

Lasso Regression: Regularization technique that introduces penalty term based on the absolute values of the coefficient encouraging sparsity in the model by driving some coefficients to exactly zero.

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Ridge Regression	40967.33	144.56	0.42	41004.91	144.49	0.43
Linear Regression	41014.64	144.68	0.424	41039.12	144.59	0.429

Implementing Decision Tree and Random Forest Models:

Decision Tree Regressor: A Decision Tree Regressor is a non-linear regression algorithm that recursively splits the dataset based on features to predict the target variable. It constructs a tree-like structure where each internal node represents a decision based on a feature, and each leaf node holds a predicted value.

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Decision Tree	76.54	0.31	0.99	30393.50	56.93	0.57

Random Forest Regressor: An ensemble learning method that builds multiple decision trees during training and outputs the average prediction of the individual trees for regression tasks. It enhances accuracy and generalization by combining predictions from diverse trees.

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Random Forest	2311.55	19.32	0.967	15974.35	51.34	0.78
Hyper-parameter Tuning	8840.23	36.60	0.87	14214.27	48.67	0.81

Model Evaluation: Models are evaluated based on the three major metrics namely Mean Squared Error (MSE), Mean Absolute Error (MAE) and R squared.

Mean Squared Error: The MSE represents the average of the squared differences between the actual target values "Amount" and the predicted values. Whereas MAE is the absolute difference between predicted and actual values. For the Linear Regression model, the MSE is approximately 41004.90. Considering the scale of the "Amount" variable, which has a maximum value of 1442 and a mean of 644.62, an MSE of 41004.90 is relatively high. It indicates a substantial level of error in your model's predictions, especially when compared to the range and mean of the target variable. However, in Random Forest model using hyper-parameter tuning the MSE is lowest compared to all other models.

R-squared: The R2 score measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating a better fit. The R2 score for Random Forest model using hyper-parameter tuning is 0.87 for training set and 0.81 for testing set. This is a good R2 score with a difference between train and test of less than 7.

CHAPTER FOUR RESULTS

Results

Best Selling Categories: The top-performing clothing categories, "Kurta," "Top Bottom Set," and "Western Dress," make up a sizable portion of Amazon India's maximum sales.

Size Preferences: Based on orders placed, the top 5 states are dominated by the Small, Medium, Large, and Extra-Large sizes, suggesting a demand for a variety of sizes.

Shipping Preferences: A significant 37% of customers choose faster shipping over regular choices, even though using the expedited shipping service results in higher customer expenditure.

Orders by Category: The "Top" category has more orders (10K) than the "Bottom" category (409 orders), indicating a possible area for development or marketing emphasis.

Order Status: Although the percentage of rejected and returned orders is lower than that of shipped orders, a sizable portion of cancelled orders call for more research into the causes of cancellations.

Inventory insights: Products in categories such as "Top Bottom Set" and "Kurta" have high inventory and order numbers, indicating their popularity. Conversely, "Western Dress" has less inventory and fewer orders.

MRP Analysis: The average Maximum Retail Price (MRP) for distinct categories varies very little across different sales channels.

MRP Trends: There aren't many changes between the MRP trends examined in March 2021 and May 2022, indicating consistent pricing practices.

International Sales: Prominent foreign clients with a strong global market presence include "Mulberries Boutique" and others.

Size Trends in International Orders: "Small," "Medium," "Large," and "Extra Large" sizes are continually in demand, with demand peaked prior to India's festival seasons, much like domestic trends.

Key variables including "Style," "Kurta," "ASIN," "Top," and "Qty" are identified by the Random Forest Model as major contributors to predicting the target variable “Amount”.

Recommendations:

Marketing Focus: To capitalize on the popularity with best-selling categories including "Western Dress," "Top Bottom Set," and "Kurta," increase marketing efforts in these areas.

Inventory Management: Consider increasing the stock for western dress apparel category to meet the demand, and optimizing the excess stocks for categories which are highly in demand using forecasting techniques.

Shipping Strategy: Consider enhancing standard shipping options to meet the 37% demand for faster shipping, potentially improving customer satisfaction.

Order Cancellation Analysis: To lower this part of customer attrition, investigate and solve the issues that lead to order cancellations.

Global Expansion: By adjusting strategies according to customer preferences, take use of knowledge from overseas sales to increase the global market presence.

CHAPTER FIVE CONCLUSION

To conclude, the primary goals of this research were to examine the dynamics of Amazon India's apparel sales.

Relationship Analysis: This research addressed the complex interactions between the pricing of clothing items and several aspects such size, type, fulfilment, and shipping. The knowledge acquired offers an in-depth understanding of the way these variables interact to shape consumer preferences and purchasing patterns.

Sales Trend Identification: The research effectively discovered patterns based on month, category, delivery method, client segment, and location through a thorough analysis of sales trends. This thorough analysis offers insightful information for planning sales and marketing campaigns.

Comparative Performance: The research conducted a comparative analysis of Amazon's performance against other prominent e-commerce sales channels, including Ajio, Flipkart, and Myntra. These kinds of comparisons provide a standard by which to evaluate the competitive landscape's advantages, disadvantages, and potential improvement areas.

International Business Insights: An in-depth analysis of business trends for various clothing categories in the global market provided insight into Amazon's position and growth prospects. Developing successful plans for international expansion requires an understanding of these patterns.

The results acknowledge the limits of the research, such as data constraints and external market issues, and offer practical recommendations for the e-commerce business. Future studies might concentrate on the dynamics of client segments, how technology affects sales patterns, and how effective focused marketing is. These paths offer a more profound comprehension of the dynamics of e-commerce.

In conclusion, the research adds value to Amazon India's clothing sales approach by highlighting areas that should be improved. Because of its broad scope, it remains relevant for researchers, practitioners, and policymakers negotiating the dynamic world of e-commerce.

REFERENCES

Sharma A. (2022) E-commerce sales dataset. Kaggle. https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commercesales-data

Khanna P. & Sampat B. (October 7, 2015) Factors Influencing Online Shopping During Diwali Festival 2014: Case Study of Flipkart and Amazon. https://scholarworks.lib.csusb.edu/jitim/vol24/iss2/5/

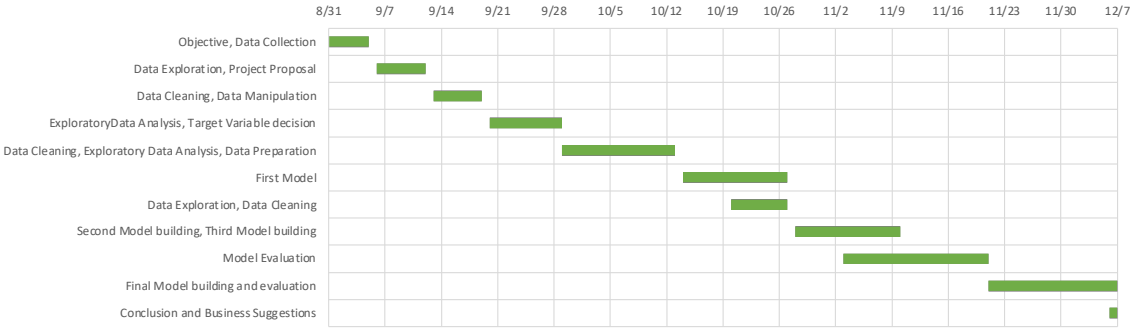
Nasabi A. & Sujaya H. (December 31, 2022) Consumer Buying Behaviour Trends of E-Commerce in India - A Case Study. International Journal of Management Technology and Social Sciences. https://supublication.com/index.php/ijmts/article/view/584

Yadav N. & Sagar M. (August 20, 2018) Amazon India's 'Apni Dukaan': Branding Strategy. Emerald Insights. https://www.emerald.com/insight/content/doi/10.1108/EEMCS-09-2017-0230/full/html

Chauhan R. & Shah S. (December 1, 2020) A Research Study on Logistic and ECommerce of Amazon and Flipkart. Pal. Arch. https://archives.palarch.nl/index.php/jae/article/view/2159

Lipson, C. (2011). *Cite right: a quick guide to citation styles--MLA, APA, Chicago, the sciences, professions, and more*. University of Chicago Press.

APPENDIX A: PROJECT PLAN AND SCHEDULE



APPENDIX B: SLIDE PRESENTATION

OPTIMIZING amazon APPAREL SALES

AUTHOR: PRANJAL PRAMOD BHAGAT
ADVISOR: DR. KHALID ABOALAYON
COURSE: MSDA3999 Fall 2023

CLARK UNIVERSITY

Table of Contents

2

Introduction

In the ever-evolving landscape of E-commerce, the goal is to decipher the complexities of Amazon India, apparel sales data. Focused on enhancing business growth and efficiency, this project delves into the complexities of apparel item prices and their dependencies on various influencing factors.

Objectives

- Investigate the relationship between apparel item price and features like type, fulfillment, shipping and size
- Identify sales trend based on month, category, shipping method, customer segment and region
- Compare Amazon's performance with other e-commerce sales channels – Ajio, Flipkart and Myntra
- Understand business trends for apparel type in the international market

Research Motivation

ADDRESSING THE KNOWLEDGE GAP

DATA-DRIVEN DECISION-MAKING

ENHANCING COMPETITIVENESS

CONTRIBUTING TO INDUSTRY GROWTH

5

Research Significance

- Strategic Inventory Optimization
- Comprehensive Market Insights
- Foundation of Future Innovation

6

Prospective Stakeholders

Retail Businesses

Optimize operations based on insights, enhancing inventory management, marketing, and pricing strategies for e-commerce retailers

Marketing Team

Tailor campaigns using customer behavior insights to boost engagement and conversion rate within the organization

Supply Chain Managers

Streamline operations with project-driven inventory optimization, reducing cost and enhancing overall efficiency

Problem Statement

Within our Amazon Sales Report, we have identified notable fluctuations in apparel item prices, driven by a multitude of influencing factors.

The challenge lies in understanding the intricate dependencies between apparel item price and these influencing factors.

The aim is to develop a machine learning model to predict apparel item amount based on the current data dependency, unraveling the nuanced between price and key variables.

Methodology:
Data Collection

The datasets can be downloaded from Kaggle and author of the data is Sharma A.

- Primary dataset:
  - Amazon Sales Report with 128,851 records from Amazon India, offering insights into SKU code, apparel types, order details shipping information from March 31, 2022, to June 29, 2022.
- Additional datasets:
  - Sales Report which focuses on SKU code, apparel types and stock details
  - March 2021 and May 2022 with price of each SKU code across various sales channel
  - International Sales Report which provides information on SKU codes, sales data, customers and gross amount from June 2021 to March 2022

https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data

Methodology:
Data Cleaning and Pre-processing

Data: Amazon Sale Report csv file

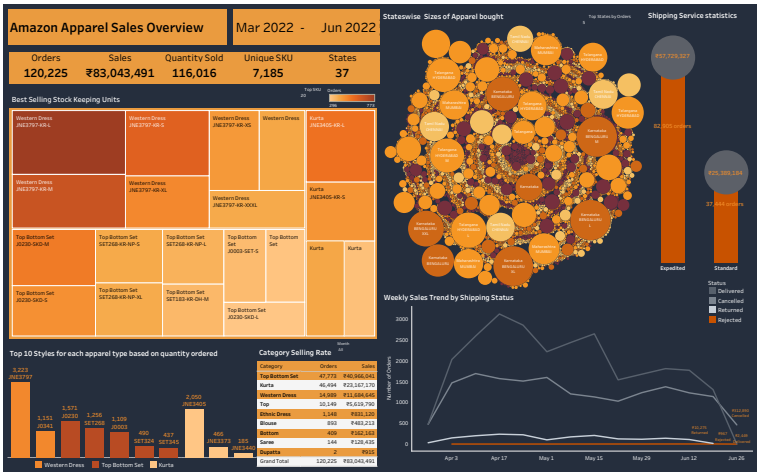
- Missing values treatment: Drop columns with more than 60% null values, remove rows for column with less than 1% null values, and imputed all other null values
- Data Manipulation: Changed the data type to date and object for categorical features
- Checking for unique categories in each object type column
- Outlier Analysis: Truncated the outliers and included data till 99th percentile

Additional datasets:

- Missing value treatment: remove rows for column with less than 1% null values
- Data Manipulation: Capitalize the categories to group different case values together, remove junk values, created a function that re-arranges the misplaced values across columns

Methodology:
Exploratory Data Analysis (EDA)

https://public.tableau.com/app/profile/pranjal.pramod.bhagat4432/viz/AmazonIndiaApparelSalesDashboard/AmazonApparelSales



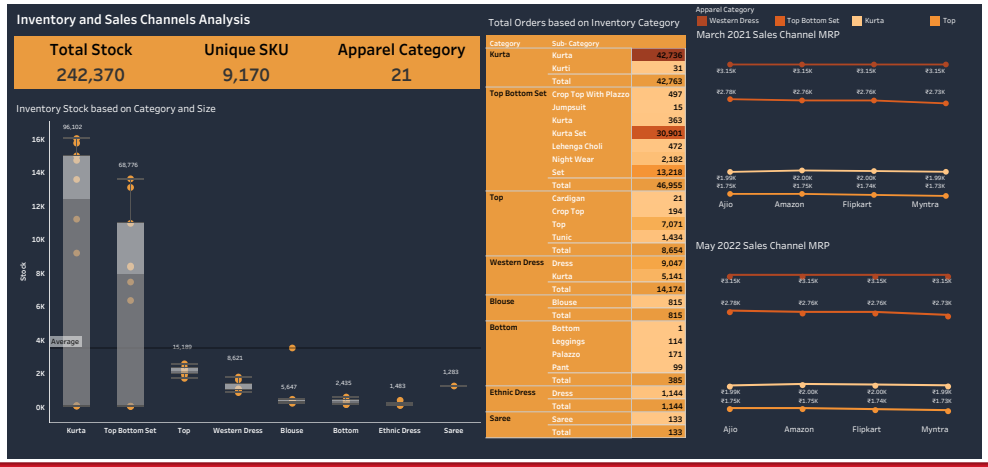
Methodology:
Exploratory Data Analysis (EDA)

Data: Amazon Sale Report csv file

- Best Selling Stock Keeping Units are from apparel category “Western Dress”, “Top Bottom Set” and “Kurta” and these categories contribute to maximum sales made by Amazon India
- If we observe top 5 states based on orders placed, size “Small”, “Medium”, “Large” and “Extra Large” are in demand
- Even though Shipping service “Expedited” leads customers to spend more, number of customers preferring faster shipping over standard is 37%
- If we look at the category wise orders, 10K orders were placed for “Top” category, however only 409 orders were placed for “Bottom” category
- Even though number of orders with “Returned” and “Rejected” status are lower than “Shipped”, we can see reasonable number of order were “Cancelled”

Methodology:
Exploratory Data Analysis (EDA)

https://public.tableau.com/app/profile/pranjal.pramod.bhagat4432/viz/InventorySalesChannelAnalysisDashboard/InventorySalesChannels



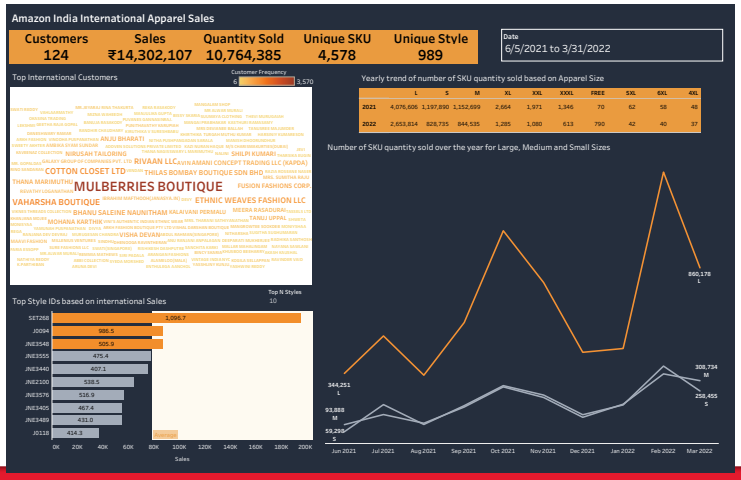
Methodology:
Exploratory Data Analysis (EDA)

Data: Inventory and March 2021/ May 2021 Report

- For apparel category “Kurta” and “Top Bottom Set” there were 42K and 46K orders respectively and the inventory had 96K and 68K stocks respectively
- However, apparel category “Western Dress” there were 14K orders and the inventory had 8K stock
- If we look at the average Maximum Retail Price (MRP) for different categories across various Sales channels we can see there is not much difference
- If we analyze the MRP trend for March 2021 and May 2022 there is not much difference

Methodology:
Exploratory Data Analysis (EDA)

https://public.tableau.com/app/profile/pranjal.pramod.bhagat4432/viz/AmazonInternationalDashboard/International



Methodology:
Exploratory Data Analysis (EDA)

Data: International Sales Report

- The top international customer “Mulberries Boutique” has placed 3570 orders
- The other top customers include “Vaharsha Boutique” , “Cotton Closet LTD” and “Ethnic Weaves Fashion LLC” with more than 1K orders
- If we observe for both the years 2021 and 2022 orders, size “Small”, “Medium”, “Large” and “Extra Large” are in demand
- International Orders for the in-demand size categories are at peak before the festive seasons in India

Methodology: Statistical

Feature Engineering: Transformed Date field to Day of Week and Month of Week

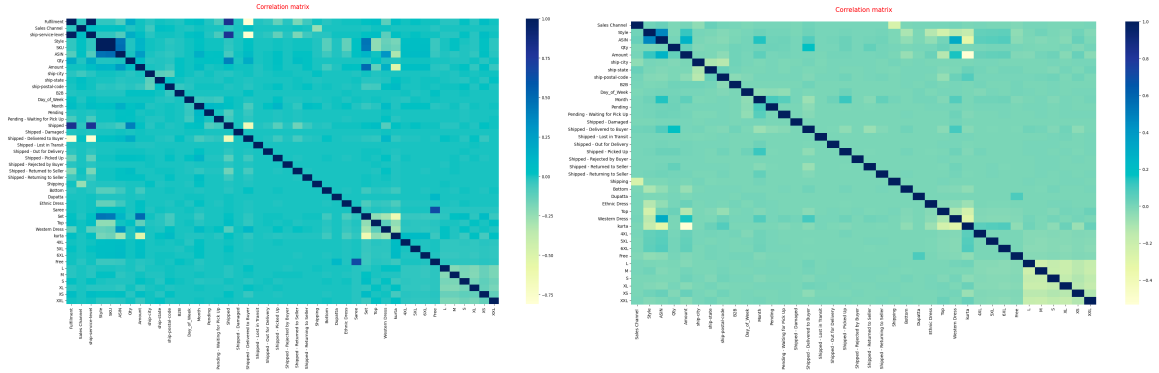
Variable Transformation: Converted categorical columns to numeric data type using

- Binary Mapping: Representing two categories as 0 and 1
- One-Hot Encoding: Creating binary columns for each category, assigning 1 to presence of category and 0 otherwise
- Label Encoding: Assigning unique numerical label to each category

17

Methodology: Statistical

Multi-collinearity : Where two or more predictor variables are highly correlated, making it challenging to discern the effect of each variables on dependent variable.



18

Methodology: Statistical

- Feature Scaling: Adjusting the scale of the features to ensure they contribute equally to the analysis
- Standard Scaler: Adjust features to have mean of 0 and standard deviation of 1
- Min Max Scaler: Scales features to a specific range between 0 and 1
- Data was split into training and testing sets (70% training and 30% testing)

Methodology: Machine Learning

Linear Regression models the relationship between variables by fitting a straight line to the observed data points, allowing prediction of the dependent variables based on the values of independent variables.

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Linear Regression with Standard Scalar	40967.33	144.56	0.42	41004.91	144.49	0.43
Linear Regression with MinMax Scalar	40967.33	144.56	0.42	41004.91	144.49	0.43
Linear Regression with Cross Validation	40961.55	144.52	0.426	41215.99	145.22	0.421

Methodology:
Machine Learning

- Ridge Regression: Regularization technique to prevent overfitting by adding a penalty term based on the sum of squared coefficient in addition to minimizing the least squares loss
- Lasso Regression: Regularization technique that introduces penalty term based on the absolute values of the coefficient encouraging sparsity in the model by driving some coefficients to exactly zero

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Ridge Regression	40967.33	144.56	0.42	41004.91	144.49	0.43
Linear Regression	41014.64	144.68	0.424	41039.12	144.59	0.429

21

Methodology:
Machine Learning

Decision Tree Regressor: A Decision Tree Regressor is a non-linear regression algorithm that recursively splits the dataset based on features to predict the target variable. It constructs a tree-like structure where each internal node represents a decision based on a feature, and each leaf node holds a predicted value.

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Decision Tree	76.54	0.31	0.99	30393.50	56.93	0.57

22

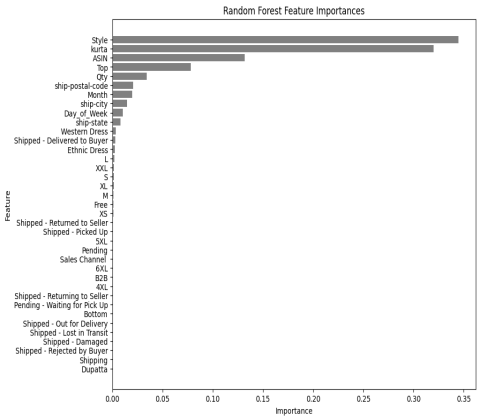
Methodology:
Machine Learning

Random Forest Regressor: An ensemble learning method that builds multiple decision trees during training and outputs the average prediction of the individual trees for regression tasks. It enhances accuracy and generalization by combining predictions from diverse trees.

Machine Learning Model	Training data score			Testing data score		
Metrics	MSE	MAE	R Squared	MSE	MAE	R Squared
Random Forest	2311.55	19.32	0.967	15974.35	51.34	0.78
Hyper-parameter Tuning	8840.23	36.60	0.87	14214.27	48.67	0.81

Results

- Feature Importance Plot: Visual representation of each feature's impact on the prediction of the target variable, "Amount."
- Top Features: "Style", "Kurta", "ASIN," "Top," and "Qty contribute significantly to the model.
- Moderate Impact: Features like " Postal Code," "Month" , "City", "Day of week" and "State" play a substantial role.



Challenges

Data Cleaning:

As the categorical columns values were misspelled or inconsistent, cleaned them using string and replace methods

The values of a dataset were shifted under incorrect column header, created a function that re-arranges the values back to its original column name

Lessons Learned & Future Work

- Acquired proficiency in interpreting and leveraging feature importance in a Random Forest model
- Uncovered the pivotal impact of "Style" and "kurta" on predicting sales amount
- Uncovered that customers prefer expedited delivery type over standard
- Propose exploring customer segmentation based on identified influential features for targeted marketing
- Consider extending the analysis to include external factors such as seasonality or marketing promotions
- Data scraping for collecting and analyzing the geographical data for International customers

Conclusion

Strategic Recommendations:

- Leverage insights from top features for targeted inventory management.
- Tailor marketing strategies based on influential features like "Style" and "kurta."
- Enhance customer engagement considering the identified impactful features.

Acknowledgments

I extend my sincere gratitude to Professor Khald Aboalayon for his invaluable guidance and mentorship throughout the project, which greatly contributed to its success. I would like to express my appreciation to Professor Kelly Giardullo for imparting valuable skills in the art of storytelling with data during the course. A special thanks to Professor Shasha Yu, whose expertise in Tableau dashboarding significantly aided me in creating dynamic and impactful visualizations, enhancing the overall presentation of my work. My gratitude also extends to my peers for their collaborative spirit, providing solutions to some of my queries and enriching the learning experience. Lastly, I acknowledge and thank Sharma A. (2022) for sharing the E-commerce sales dataset on Kaggle, which served as the foundation for my research.

References

Sharma A. (2022) E-commerce sales dataset. Kaggle.

https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data

Khanna P. & Sampat B. (October 7, 2015) Factors Influencing Online Shopping During Diwali Festival 2014: Case Study of Flipkart and Amazon.

https://scholarworks.lib.csusb.edu/jitim/vol24/iss2/5/

Nasabi A. & Sujaya H. (December 31, 2022) Consumer Buying Behaviour Trends of E-Commerce in India - A Case Study. International Journal of Management Technology and Social Sciences.

https://supublication.com/index.php/ijmts/article/view/584

Yadav N. & Sagar M. (August 20, 2018) Amazon India's 'Apni Dukaan': Branding Strategy. Emerald Insights.

https://www.emerald.com/insight/content/doi/10.1108/EEMCS-09-2017-0230/full/html

Chauhan R. & Shah S. (December 1, 2020) A Research Study on Logistic and E-Commerce of Amazon and Flipkart. Pal.Arch.

https://archives.palarch.nl/index.php/jae/article/view/2159

THANK YOU

CLARK UNIVERSITY

CHALLENGE CONVENTION. CHANGE OUR WORLD.

APPENDIX C: SOURCE CODE

CapstoneProject

December 16, 2023

1 Optimizing Amazon Apparel Sales: Data Analysis for Business Growth and Efficiency

[1]:
```
# Importing libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re

import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import cross_val_score,cross_val_predict
from sklearn import metrics
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso


# Ignore warnings

import warnings
warnings.filterwarnings("ignore")
```

```
# Dsiaplay all rows and columns

pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)
```

[2]:
```
font_dict = {
    'family': 'monospace',      # Font family (e.g., 'serif', 'sans-serif',␣
 ↪'monospace')
    'color': '#FF9900',     # Amazon bright orage
    'weight': 'bold',     # Font weight ('normal', 'bold', 'light', 'heavy')
    'size': 12,             # Font size
}

# Amazon blue theme color
amazon_blue = '#146eb4'
```

1.1 Data Exploration

1.1.1 Amazon Sale Report data

[3]:
```
# Read Amazon Sale Report csv file

amazon_df = pd.read_csv("Amazon Sale Report.csv")

# Display first 5 rows

amazon_df.head()
```

[3]:

	index	Order ID	Date	Status	\
0	0	405-8078784-5731545	04-30-22	Cancelled	
1	1	171-9198151-1101146	04-30-22	Shipped - Delivered to Buyer	
2	2	404-0687676-7273146	04-30-22	Shipped	
3	3	403-9615377-8133951	04-30-22	Cancelled	
4	4	407-1069790-7240320	04-30-22	Shipped	

	Fulfilment	Sales Channel	ship-service-level	Style	SKU	\
0	Merchant	Amazon.in	Standard	SET389	SET389-KR-NP-S	
1	Merchant	Amazon.in	Standard	JNE3781	JNE3781-KR-XXXL	
2	Amazon	Amazon.in	Expedited	JNE3371	JNE3371-KR-XL	
3	Merchant	Amazon.in	Standard	J0341	J0341-DR-L	
4	Amazon	Amazon.in	Expedited	JNE3671	JNE3671-TU-XXXL	

	Category	Size	ASIN	Courier Status	Qty	currency	Amount	\
0	Set	S	B09KXVBD7Z	NaN	0	INR	647.62	
1	kurta	3XL	B09K3WFS32	Shipped	1	INR	406.00	
2	kurta	XL	B07WV4JV4D	Shipped	1	INR	329.00	
3	Western Dress	L	B099NRCT7B	NaN	0	INR	753.33	

```
4            Top  3XL  B098714BZP        Shipped    1      INR  574.00

     ship-city   ship-state  ship-postal-code ship-country  \
0       MUMBAI  MAHARASHTRA          400081.0           IN
1    BENGALURU    KARNATAKA          560085.0           IN
2  NAVI MUMBAI  MAHARASHTRA          410210.0           IN
3   PUDUCHERRY   PUDUCHERRY          605008.0           IN
4      CHENNAI   TAMIL NADU          600073.0           IN

                                        promotion-ids    B2B fulfilled-by  \
0                                                 NaN  False    Easy Ship
1  Amazon PLCC Free-Financing Universal Merchant …  False    Easy Ship
2       IN Core Free Shipping 2015/04/08 23-48-5-108   True          NaN
3                                                 NaN  False    Easy Ship
4                                                 NaN  False          NaN

  Unnamed: 22
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
```

[4]:
```
# Dispaly the size and dimension

amazon_df.shape

print("Size of data : {} records".format(amazon_df.shape[0]))
print("Dimension of data: {} columns".format(amazon_df.shape[1]))
```

```
Size of data : 128975 records
Dimension of data: 24 columns
```

[5]:
```
# Display the information of data

amazon_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128975 entries, 0 to 128974
Data columns (total 24 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   index               128975 non-null  int64
 1   Order ID            128975 non-null  object
 2   Date                128975 non-null  object
 3   Status              128975 non-null  object
 4   Fulfilment          128975 non-null  object
 5   Sales Channel       128975 non-null  object
```

```
 6   ship-service-level  128975 non-null  object
 7   Style               128975 non-null  object
 8   SKU                 128975 non-null  object
 9   Category            128975 non-null  object
 10  Size                128975 non-null  object
 11  ASIN                128975 non-null  object
 12  Courier Status      122103 non-null  object
 13  Qty                 128975 non-null  int64
 14  currency            121180 non-null  object
 15  Amount              121180 non-null  float64
 16  ship-city           128942 non-null  object
 17  ship-state          128942 non-null  object
 18  ship-postal-code    128942 non-null  float64
 19  ship-country        128942 non-null  object
 20  promotion-ids       79822 non-null   object
 21  B2B                 128975 non-null  bool
 22  fulfilled-by        39277 non-null   object
 23  Unnamed: 22         79925 non-null   object
dtypes: bool(1), float64(2), int64(2), object(19)
memory usage: 22.8+ MB
```

[6]:
```
# Describe the data features

amazon_df.describe()
```

[6]:

	index	Qty	Amount	ship-postal-code
count	128975.000000	128975.000000	121180.000000	128942.000000
mean	64487.000000	0.904431	648.561465	463966.236509
std	37232.019822	0.313354	281.211687	191476.764941
min	0.000000	0.000000	0.000000	110001.000000
25%	32243.500000	1.000000	449.000000	382421.000000
50%	64487.000000	1.000000	605.000000	500033.000000
75%	96730.500000	1.000000	788.000000	600024.000000
max	128974.000000	15.000000	5584.000000	989898.000000

As we can see from the above data exploration there are 128,975 rows and 24 columns. We need to conduct data cleaning in order to use the data for predictive modeling, the date column needs to be converted to date type. The ship-postal-code columns is numerical data type but is not calculative so it needs to be converted to categorical data type. Aditionally, the quantity column needs to be analyzed further for verfying the data distribution.

Missing value check

[7]:
```
# Checking null values

amazon_df.isnull().sum()
```

[7]:
```
index                   0
Order ID                0
Date                    0
Status                  0
Fulfilment              0
Sales Channel           0
ship-service-level      0
Style                   0
SKU                     0
Category                0
Size                    0
ASIN                    0
Courier Status       6872
Qty                     0
currency             7795
Amount               7795
ship-city              33
ship-state             33
ship-postal-code       33
ship-country           33
promotion-ids       49153
B2B                     0
fulfilled-by        89698
Unnamed: 22         49050
dtype: int64
```

[8]:
```
# Dispaly columns with null values and their respetive percenatge
i = 1
for (col,percentage) in (amazon_df.isnull().sum()/ len(amazon_df) * 100).
 ↪items():
    if percentage > 0:
        print(f"{i}. {col} has {percentage:.2f}% of missing values")
        i+=1
```

1. Courier Status has 5.33% of missing values
2. currency has 6.04% of missing values
3. Amount has 6.04% of missing values
4. ship-city has 0.03% of missing values
5. ship-state has 0.03% of missing values
6. ship-postal-code has 0.03% of missing values
7. ship-country has 0.03% of missing values
8. promotion-ids has 38.11% of missing values
9. fulfilled-by has 69.55% of missing values
10. Unnamed: 22 has 38.03% of missing values

As we can see there are 10 columns having missing valuee. We will analyse columns based on their missing value percentage.

[9]:
```
# Looking at the column with most missing values

i = 1
for (col,percentage) in (amazon_df.isnull().sum()/ len(amazon_df) * 100).
 ↪items():
    if percentage > 50 :
        print(f"{i}. {col} has {percentage:.2f}% of missing values\n")
        print(amazon_df[col].value_counts())
        i+=1
```

```
1. fulfilled-by has 69.55% of missing values

fulfilled-by
Easy Ship    39277
Name: count, dtype: int64
```

We can see that column fulfilled-by has 69.55% of missing values and as a categorical column it contains only one category so we will drop this column as it won't contribute towards our analysis.

[10]:
```
# Drop fulfilled-by

amazon_df.drop(columns="fulfilled-by", inplace=True)
```

[11]:
```
# Verify by checking the columns

amazon_df.columns
```

[11]:
```
Index(['index', 'Order ID', 'Date', 'Status', 'Fulfilment', 'Sales Channel ',
       'ship-service-level', 'Style', 'SKU', 'Category', 'Size', 'ASIN',
       'Courier Status', 'Qty', 'currency', 'Amount', 'ship-city',
       'ship-state', 'ship-postal-code', 'ship-country', 'promotion-ids',
       'B2B', 'Unnamed: 22'],
      dtype='object')
```

[12]:
```
# Looking at the column with missing values btween 50% to 5%

i = 1
for (col,percentage) in (amazon_df.isnull().sum()/ len(amazon_df) * 100).
 ↪items():
    if percentage <= 50 and percentage >= 5 :
        print(f"{i}. {col} has {percentage:.2f}% of missing values\n")
        #print(amazon_df[col].value_counts())
        i+=1
```

```
1. Courier Status has 5.33% of missing values

2. currency has 6.04% of missing values
```

3. Amount has 6.04% of missing values

4. promotion-ids has 38.11% of missing values

5. Unnamed: 22 has 38.03% of missing values

[13]:
```
# Analysing column Courier Status

amazon_df["Courier Status"].value_counts()
```

[13]:
```
Courier Status
Shipped      109487
Unshipped      6681
Cancelled      5935
Name: count, dtype: int64
```

The column Courier status has similar categories as column Status so we will drop this column.

[14]:
```
# Drop Courier Status

amazon_df.drop(columns="Courier Status", inplace=True)
```

[15]:
```
# Analysing column Courier Status

amazon_df["currency"].value_counts()
```

[15]:
```
currency
INR    121180
Name: count, dtype: int64
```

[16]:
```
amazon_df[amazon_df["currency"].isnull()].head()
```

[16]:

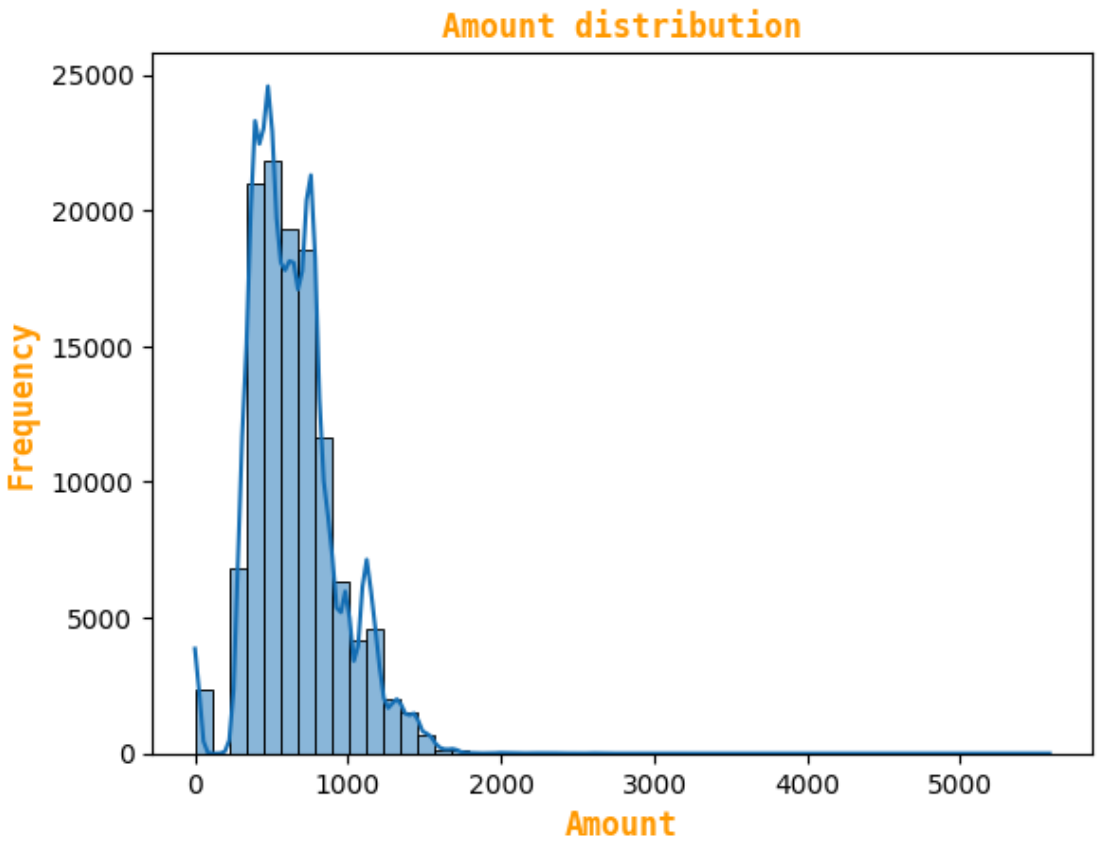
	index	Order ID	Date	Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	Category	Size	ASIN
8	8	407-5443024-5233168	04-30-22	Cancelled	Amazon	Amazon.in	Expedited	SET200	SET200-KR-NP-A-XXXL	Set	3XL	B08L91ZZXN
29	29	404-5933402-8801952	04-30-22	Cancelled	Merchant	Amazon.in	Standard	JNE2132	JNE2132-KR-398-XXXL	kurta	3XL	B07JG3CND8
65	65	171-4137548-0481151	04-30-22	Cancelled	Amazon	Amazon.in	Expedited	JNE3373	JNE3373-KR-XXL	kurta	XXL	B082W8RWN1
84	84	403-9950518-0349133	04-30-22	Cancelled	Amazon	Amazon.in	Expedited	JNE3510	JNE3510-KR-M	kurta	M	B08WPR5MCB
95	95	405-9112089-3379536	04-30-22	Cancelled	Amazon	Amazon.in	Expedited	JNE3405	JNE3405-KR-L	kurta	L	B081WSCKPQ

```
    Qty currency  Amount  ship-city   ship-state  ship-postal-code  \
8     0      NaN     NaN  HYDERABAD    TELANGANA          500008.0
29    0      NaN     NaN   GUWAHATI        ASSAM          781003.0
65    0      NaN     NaN      Dahod      Gujarat          389151.0
84    0      NaN     NaN  HYDERABAD    TELANGANA          500072.0
95    0      NaN     NaN       PUNE  MAHARASHTRA          411046.0

   ship-country                                promotion-ids    B2B  \
8            IN  IN Core Free Shipping 2015/04/08 23-48-5-108  False
29           IN                                          NaN  False
65           IN                                          NaN  False
84           IN                                          NaN  False
95           IN                                          NaN  False

   Unnamed: 22
8          NaN
29         NaN
65         NaN
84         NaN
95         NaN
```

Currency column has missing values as the amount is missing as well. Now that we already know the currency is INR for whole data we will drop currency column and analyze amount column.

[17]:
```
# Drop currency

amazon_df.drop(columns="currency", inplace=True)
```

[18]:
```
# Analysing amount column

# Create a histogram using Seaborn
sns.histplot(amazon_df["Amount"], bins=50, kde=True, color=amazon_blue)

# Add labels and a title
plt.xlabel("Amount", fontdict=font_dict)
plt.ylabel("Frequency", fontdict=font_dict)
plt.title("Amount distribution", fontdict=font_dict)

# Show the plot
plt.show()
```



As the amount column appears to be right skewed we will replace the missing values with the median value.

[19]:
```
# Imputing missing values of column amount using median

amazon_df["Amount"].fillna(amazon_df["Amount"].median(), inplace=True)
```

[21]:
```
# Saving promotion ids for future use case
promotion_ids = amazon_df["promotion-ids"]

# Dropping promotion-id

amazon_df.drop(columns="promotion-ids", inplace=True)
```

[22]:
```
# Checking the Unnamed: 22 column

amazon_df["Unnamed: 22"].value_counts()
```

[22]:
```
Unnamed: 22
False    79925
```

```
Name: count, dtype: int64
```

Unnamed: 22 has only one category False which is not useful for the analysis so we will drop the column.

[23]:
```
# Dropping Unnamed: 22

amazon_df.drop(columns="Unnamed: 22", inplace=True)
```

[24]:
```
# Looking at the column with missing values but less than 5%

i = 1
for (col,percentage) in (amazon_df.isnull().sum()/ len(amazon_df) * 100).
 ↪items():
    if percentage <= 5 and percentage > 0:
        print(f"{i}. {col} has {percentage:.2f}% of missing values\n")
        i+=1
```

1. ship-city has 0.03% of missing values

2. ship-state has 0.03% of missing values

3. ship-postal-code has 0.03% of missing values

4. ship-country has 0.03% of missing values

As the missing values are less than 1% we will be dropping this records.

[25]:
```
# Dropping the records having missing values

amazon_df = amazon_df.dropna()
```

[26]:
```
# Checking null values

amazon_df.isnull().sum()
```

[26]:
```
index                 0
Order ID              0
Date                  0
Status                0
Fulfilment            0
Sales Channel         0
ship-service-level    0
Style                 0
SKU                   0
Category              0
Size                  0
```

```
ASIN                  0
Qty                   0
Amount                0
ship-city             0
ship-state            0
ship-postal-code      0
ship-country          0
B2B                   0
dtype: int64
```

[27]:
```
# Re-checking the shape

amazon_df.shape
```

[27]:
```
(128942, 19)
```

Now the data doesn't have missing values.

Data Manipulation

[28]:
```
# Checking the data type of columns

amazon_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 128942 entries, 0 to 128974
Data columns (total 19 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   index               128942 non-null  int64
 1   Order ID            128942 non-null  object
 2   Date                128942 non-null  object
 3   Status              128942 non-null  object
 4   Fulfilment          128942 non-null  object
 5   Sales Channel       128942 non-null  object
 6   ship-service-level  128942 non-null  object
 7   Style               128942 non-null  object
 8   SKU                 128942 non-null  object
 9   Category            128942 non-null  object
 10  Size                128942 non-null  object
 11  ASIN                128942 non-null  object
 12  Qty                 128942 non-null  int64
 13  Amount              128942 non-null  float64
 14  ship-city           128942 non-null  object
 15  ship-state          128942 non-null  object
 16  ship-postal-code    128942 non-null  float64
 17  ship-country        128942 non-null  object
 18  B2B                 128942 non-null  bool
```

```
dtypes: bool(1), float64(2), int64(2), object(14)
memory usage: 18.8+ MB
```

From the above information we can see that date column is object type so we will convert it to date type. Additionally, ship-postal-code and B2B are required to be converted to object type.

[29]:
```
# Converting column Date to date type

amazon_df["Date"] = pd.to_datetime(amazon_df["Date"])
```

[30]:
```
# Print the time period of data

print("Start date: ", amazon_df["Date"].min())
print("End date: ", amazon_df["Date"].max())
```

```
Start date:  2022-03-31 00:00:00
End date:  2022-06-29 00:00:00
```

As we can see the data is from March 2022 till June 2022.

[31]:
```
# Converting column B2B and ship-postal-code to object type

amazon_df["B2B"] = amazon_df["B2B"].astype('object')
amazon_df["ship-postal-code"] = amazon_df["ship-postal-code"].astype('int')
amazon_df["ship-postal-code"] = amazon_df["ship-postal-code"].astype('object')
```

[32]:
```
# Save categorical columns (object or category data types)
categorical_columns = amazon_df.select_dtypes(include=["object"]).columns.
 ↪tolist()

# Save numeric columns (int and float data types)
numeric_columns = amazon_df.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()

print("Categorical Columns:", categorical_columns)
print("Numeric Columns:", numeric_columns)
```

```
Categorical Columns: ['Order ID', 'Status', 'Fulfilment', 'Sales Channel ',
'ship-service-level', 'Style', 'SKU', 'Category', 'Size', 'ASIN', 'ship-city',
'ship-state', 'ship-postal-code', 'ship-country', 'B2B']
Numeric Columns: ['index', 'Qty', 'Amount']
```

[33]:
```
# Analysing Quantity column

amazon_df['Qty'].value_counts().sort_values()
```

[33]:
```
Qty
15         1
```

```
9          1
13         1
8          1
5          2
4          9
3         32
2        341
0      12802
1     115752
Name: count, dtype: int64
```

[34]:
```
# Check the categorical columns having categories greater than 20
i=1
for column in categorical_columns:
    if len(amazon_df[column].value_counts()) > 15:
        print("{0} {1} contains {2} unique categories".
 ↪format(i,column,len(amazon_df[column].unique())))
        i+=1
```

```
1 Order ID contains 120350 unique categories
2 Style contains 1377 unique categories
3 SKU contains 7195 unique categories
4 ASIN contains 7190 unique categories
5 ship-city contains 8955 unique categories
6 ship-state contains 69 unique categories
7 ship-postal-code contains 9459 unique categories
```

[35]:
```
# Analysing ship-state column

amazon_df["ship-state"].value_counts().sort_index()
```

[35]:
```
ship-state
ANDAMAN & NICOBAR          257
ANDHRA PRADESH            5430
APO                          1
AR                           1
ARUNACHAL PRADESH          141
ASSAM                     1663
Arunachal Pradesh            3
Arunachal pradesh            2
BIHAR                     2086
Bihar                       27
CHANDIGARH                 322
CHHATTISGARH               909
Chandigarh                  11
DADRA AND NAGAR             70
DELHI                     6782
```

Delhi	164
GOA	1102
Goa	30
Gujarat	4489
HARYANA	4415
HIMACHAL PRADESH	788
JAMMU & KASHMIR	702
JHARKHAND	1456
KARNATAKA	17326
KERALA	6585
LADAKH	43
LAKSHADWEEP	4
MADHYA PRADESH	2529
MAHARASHTRA	22260
MANIPUR	311
MEGHALAYA	204
MIZORAM	75
Manipur	5
Meghalaya	3
Mizoram	1
NAGALAND	184
NL	2
Nagaland	1
New Delhi	81
ODISHA	2115
Odisha	21
Orissa	2
PB	1
PUDUCHERRY	349
PUNJAB	1869
Pondicherry	1
Puducherry	1
Punjab	34
Punjab/Mohali/Zirakpur	1
RAJASTHAN	2650
RJ	2
Rajasthan	55
Rajshthan	3
Rajsthan	1
SIKKIM	202
Sikkim	3
TAMIL NADU	11483
TELANGANA	11330
TRIPURA	151
UTTAR PRADESH	10638
UTTARAKHAND	1553
WEST BENGAL	5963

```
bihar                          1
delhi                         21
goa                            5
orissa                         1
punjab                        14
rajasthan                      6
rajsthan                       1
Name: count, dtype: int64
```

We can see that there are some states with same spelling but different case.

[36]:
```
# Change the ship-state to title case

amazon_df["ship-state"] = amazon_df["ship-state"].str.title()
```

[37]:
```
# Check the value count of ship-state

amazon_df["ship-state"].value_counts().sort_index()
```

[37]:
```
ship-state
Andaman & Nicobar            257
Andhra Pradesh              5430
Apo                            1
Ar                             1
Arunachal Pradesh            146
Assam                       1663
Bihar                       2114
Chandigarh                   333
Chhattisgarh                 909
Dadra And Nagar               70
Delhi                       6967
Goa                         1137
Gujarat                     4489
Haryana                     4415
Himachal Pradesh             788
Jammu & Kashmir              702
Jharkhand                   1456
Karnataka                  17326
Kerala                      6585
Ladakh                        43
Lakshadweep                    4
Madhya Pradesh              2529
Maharashtra                22260
Manipur                      316
Meghalaya                    207
Mizoram                       76
Nagaland                     185
```

```
New Delhi                     81
Nl                             2
Odisha                      2136
Orissa                         3
Pb                             1
Pondicherry                    1
Puducherry                   350
Punjab                      1917
Punjab/Mohali/Zirakpur         1
Rajasthan                   2711
Rajshthan                      3
Rajsthan                       2
Rj                             2
Sikkim                       205
Tamil Nadu                 11483
Telangana                  11330
Tripura                      151
Uttar Pradesh              10638
Uttarakhand                 1553
West Bengal                 5963
Name: count, dtype: int64
```

[38]:
```
# Analyse the city of ship-state

amazon_df[amazon_df["ship-state"].isin(("Apo","Ar","Nl","Pb","Rj"))]
```

[38]:

	index	Order ID	Date	Status	\
3413	3413	171-2247193-1690717	2022-04-28	Shipped	
25093	25093	407-8361131-8501960	2022-04-15	Shipped - Delivered to Buyer	
39782	39782	404-6611202-9794717	2022-04-06	Shipped	
45187	45187	405-0034289-0259545	2022-04-03	Cancelled	
49144	49144	405-8584178-5101966	2022-05-31	Shipped - Delivered to Buyer	
51420	51420	407-1375529-7104311	2022-05-30	Shipped	
94844	94844	171-4171855-7000362	2022-06-26	Shipped	

	Fulfilment	Sales Channel	ship-service-level	Style	SKU	\
3413	Amazon	Amazon.in	Expedited	JNE3640	JNE3640-TP-N-M	
25093	Merchant	Amazon.in	Standard	JNE3160	JNE3160-KR-G-XL	
39782	Amazon	Amazon.in	Expedited	JNE3458	JNE3458-KR-XXXL	
45187	Amazon	Amazon.in	Expedited	J0077	J0077-SKD-S	
49144	Merchant	Amazon.in	Standard	JNE3793	JNE3793-KR-XL	
51420	Amazon	Amazon.in	Expedited	JNE3659	JNE3659-TP-N-L	
94844	Amazon	Amazon.in	Expedited	JNE3654	JNE3654-TP-M	

	Category	Size	ASIN	Qty	Amount	ship-city	ship-state	\
3413	Top	M	B08ZHM3373	1	518.0	DIMAPUR	Nl	
25093	kurta	XL	B07K4C1KTB	1	685.0	JAIPUR	Rj	

```
39782    kurta  3XL  B08HK3XSKF    1   399.0      ZIRA         Pb
45187      Set    S  B08YNTF3X3    0   605.0       APO        Apo
49144    kurta   XL  B09NQ6824N    1   355.0   JODHPUR         Rj
51420      Top    L  B08ZHT1PKK    1   493.0  ITANAGAR         Ar
94844      Top    M  B09B3HRDLP    1   443.0   DIMAPUR         Nl

      ship-postal-code ship-country    B2B
3413            797116           IN  False
25093           302029           IN   True
39782           142044           IN  False
45187           959121           IN  False
49144           342009           IN  False
51420           791113           IN  False
94844           797116           IN  False
```

We can see that there are some abbrevated states so we will look at the city and replace the abbrevated state with the state name.

[39]:
```
# Replace abbrevated state with the state name

amazon_df["ship-state"] = amazon_df["ship-state"].replace({"Nl":"Nagaland","Rj":
 ↪"Rajasthan","Pb":"Punjab","Ar":"Arunachal Pradesh","Orissa":
 ↪"Odisha","Puducherry":"Pondicherry","Punjab/Mohali/Zirakpur":
 ↪"Punjab","Rajshthan":"Rajasthan","Rajsthan":"Rajasthan"})
```

[40]:
```
# Drop the row with state as Apo

amazon_df = amazon_df[~(amazon_df["ship-state"] == "Apo")]
```

[41]:
```
amazon_df["ship-state"].value_counts().sort_index()
```

[41]:
```
ship-state
Andaman & Nicobar       257
Andhra Pradesh         5430
Arunachal Pradesh       147
Assam                  1663
Bihar                  2114
Chandigarh              333
Chhattisgarh            909
Dadra And Nagar          70
Delhi                  6967
Goa                    1137
Gujarat                4489
Haryana                4415
Himachal Pradesh        788
Jammu & Kashmir         702
Jharkhand              1456
```

```
Karnataka             17326
Kerala                 6585
Ladakh                   43
Lakshadweep               4
Madhya Pradesh         2529
Maharashtra           22260
Manipur                 316
Meghalaya               207
Mizoram                  76
Nagaland                187
New Delhi                81
Odisha                 2139
Pondicherry             351
Punjab                 1919
Rajasthan              2718
Sikkim                  205
Tamil Nadu            11483
Telangana             11330
Tripura                 151
Uttar Pradesh         10638
Uttarakhand            1553
West Bengal            5963
Name: count, dtype: int64
```

[42]:
```
# Check the value counts of categorical columns
i=1
for column in categorical_columns:
    if len(amazon_df[column].value_counts()) < 15:
        print(i,amazon_df[column].value_counts())
        print()
        i+=1
```

```
1 Status
Shipped                          77788
Shipped - Delivered to Buyer     28762
Cancelled                        18324
Shipped - Returned to Seller      1950
Shipped - Picked Up                973
Pending                            658
Pending - Waiting for Pick Up      281
Shipped - Returning to Seller      145
Shipped - Out for Delivery          35
Shipped - Rejected by Buyer         11
Shipping                             8
Shipped - Lost in Transit            5
Shipped - Damaged                    1
Name: count, dtype: int64
```

```
2 Fulfilment
Amazon      89677
Merchant    39264
Name: count, dtype: int64

3 Sales Channel
Amazon.in     128817
Non-Amazon       124
Name: count, dtype: int64

4 ship-service-level
Expedited    88594
Standard     40347
Name: count, dtype: int64

5 Category
Set              50271
kurta            49859
Western Dress    15499
Top              10620
Ethnic Dress      1159
Blouse             926
Bottom             440
Saree              164
Dupatta              3
Name: count, dtype: int64

6 Size
M       22704
L       22123
XL      20872
XXL     18093
S       17083
3XL     14815
XS      11160
6XL       738
5XL       550
4XL       425
Free      378
Name: count, dtype: int64

7 ship-country
IN    128941
Name: count, dtype: int64

8 B2B
False    128070
```

```
True        871
Name: count, dtype: int64
```

As ship-country column has one country we will drop this column.

[43]:
```
# Dropping ship-country

amazon_df.drop(columns="ship-country", inplace=True)
```

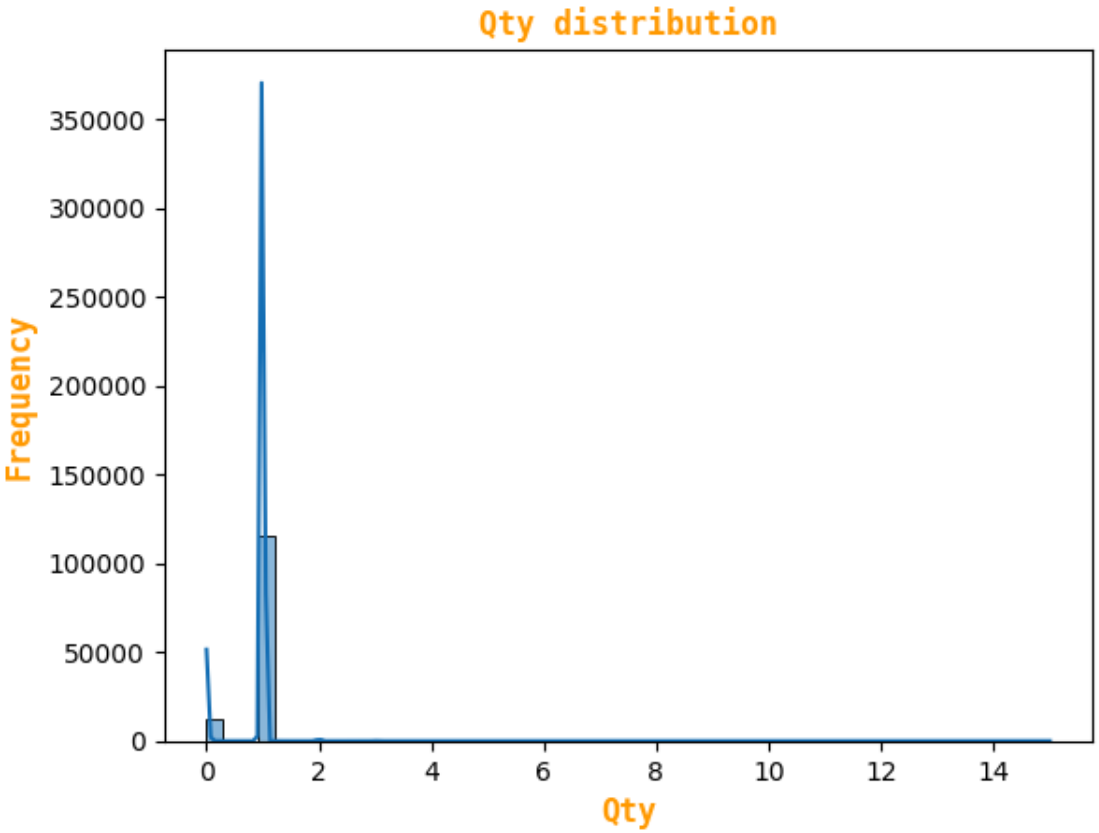
[44]:
```
def clean_and_uppercase(df, column_name):
    df[column_name] = df[column_name].str.replace(r'[^a-zA-Z ]', '',␣
 ↪regex=True)  # Remove special characters
    df[column_name] = df[column_name].str.upper()  # Convert to uppercase

# Cleaning ship-city
clean_and_uppercase(amazon_df, "ship-city")
```

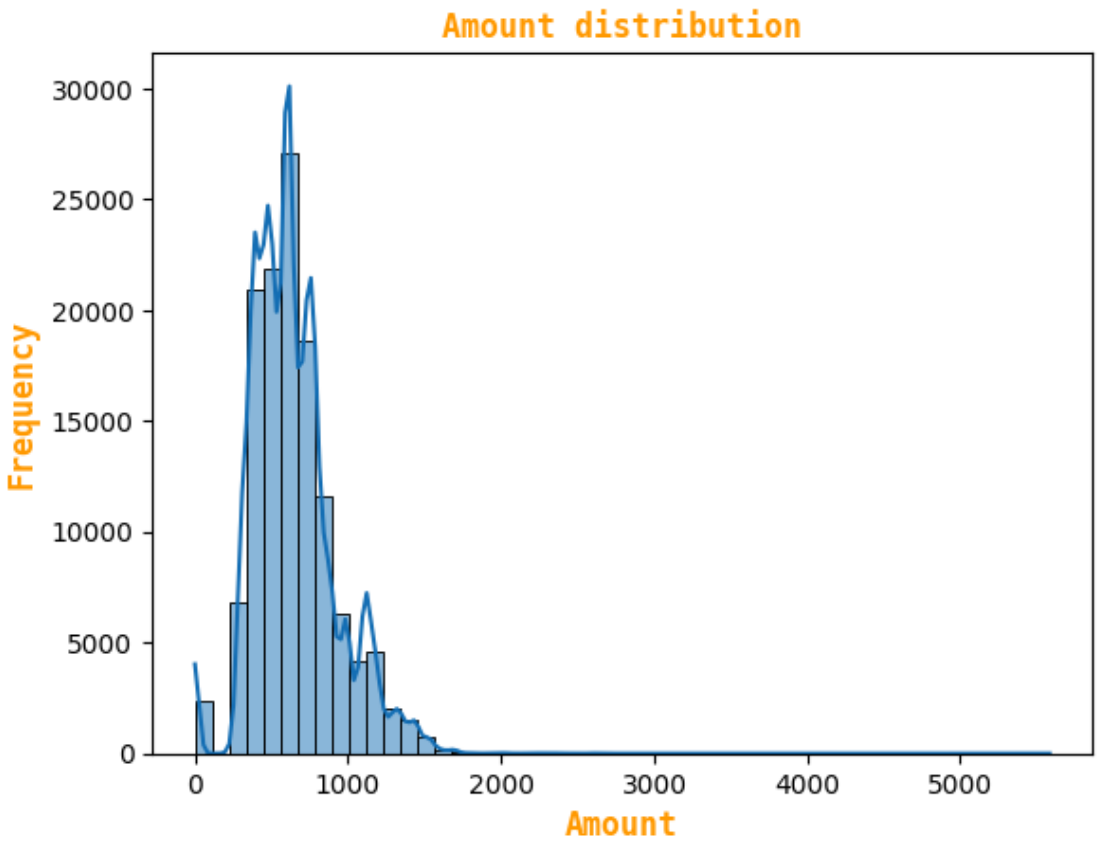
[45]:
```
amazon_df.shape
```

[45]:
```
(128941, 18)
```

Outlier Analysis

[46]:
```
# Checking for distribution of numeric columns

for column in numeric_columns[1:]:

    # Create a histogram using Seaborn
    sns.histplot(amazon_df[column], bins=50, kde=True, color=amazon_blue)

    # Add labels and a title
    plt.xlabel(column, fontdict=font_dict)
    plt.ylabel("Frequency", fontdict=font_dict)
    plt.title(column+" distribution", fontdict=font_dict)

    # Show the plot
    plt.show()
```


Qty distribution
Frequency
350000
300000
250000
200000
150000
100000
50000
0
0
2
4
6
8
10
12
14
Qty



From the above histogram we can see that both quantity and amont seems to be right skewed so we need to handle the outliers.

[47]:
```
# Describing Qty and Amount columns

amazon_df[["Qty","Amount"]].describe(percentiles=[0.25,0.5,0.75,0.8,0.9,0.95,0.
↪99])
```

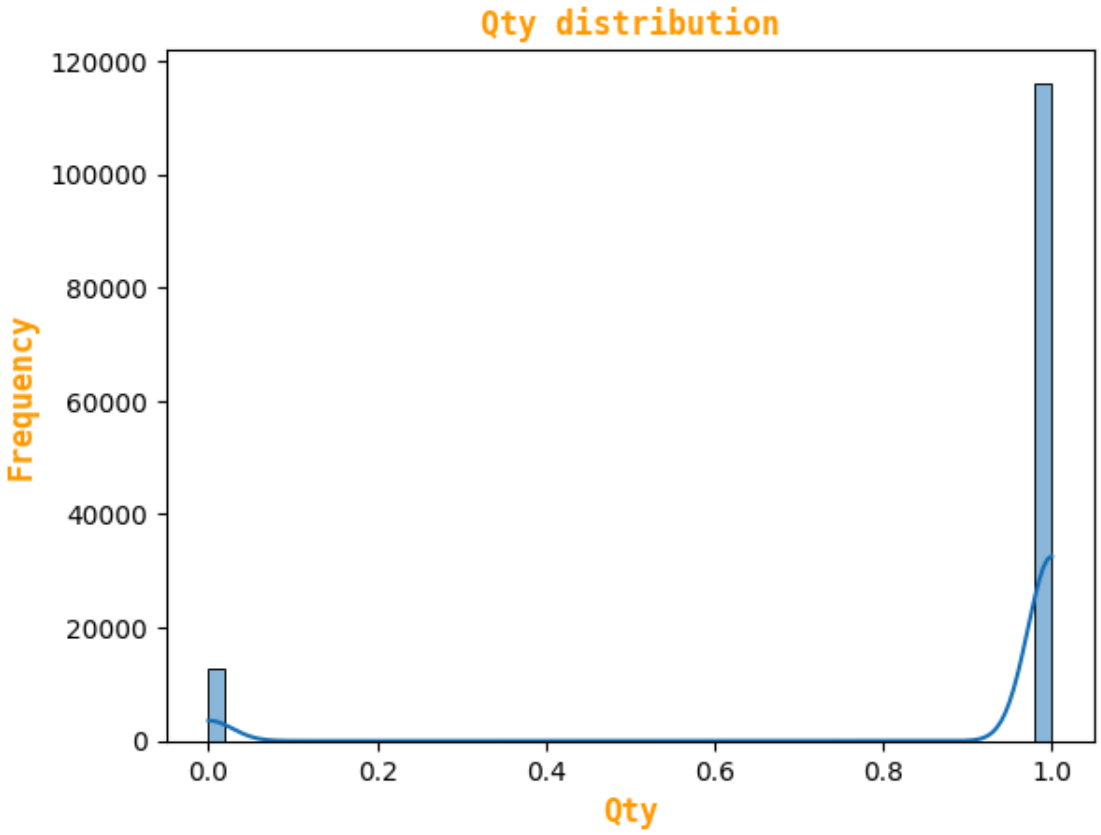
[47]:

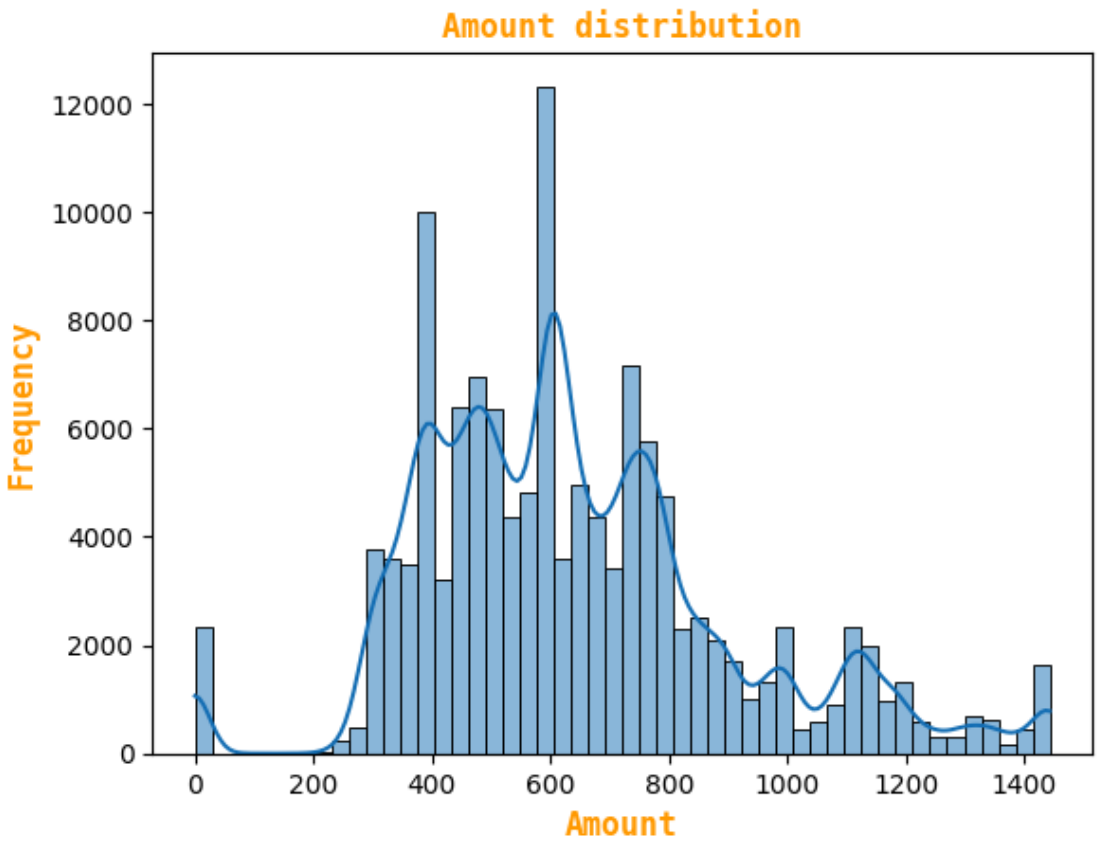
	Qty	Amount
count	128941.000000	128941.000000
mean	0.904452	645.940138
std	0.313331	272.790076
min	0.000000	0.000000
25%	1.000000	459.000000
50%	1.000000	605.000000
75%	1.000000	771.000000
80%	1.000000	824.000000
90%	1.000000	1033.000000
95%	1.000000	1166.000000
99%	1.000000	1442.000000

```
max         15.000000    5584.000000
```

[48]:
```
# Truncate outliers at the 99th percentile

for column in numeric_columns[1:]:
    threshold = amazon_df[column].quantile(0.99)
    amazon_df[column] = amazon_df[column].apply(lambda x: threshold if x >␣
 ↪threshold else x)
```

[49]:
```
# Checking for distribution of numeric columns

for column in numeric_columns[1:]:

    # Create a histogram using Seaborn
    sns.histplot(amazon_df[column], bins=50, kde=True, color=amazon_blue)

    # Add labels and a title
    plt.xlabel(column, fontdict=font_dict)
    plt.ylabel("Frequency", fontdict=font_dict)
    plt.title(column+" distribution", fontdict=font_dict)

    # Show the plot
    plt.show()
```


Qty distribution
120000
100000
80000
60000
40000
20000
0
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
Qty



[50]:
```
amazon_df.shape
```

[50]:
```
(128941, 18)
```

[51]:
```
amazon_df.to_csv("Amazon_cleaned_dataset.csv")
```

[52]:
```
amazon_df.columns
```

[52]:
```
Index(['index', 'Order ID', 'Date', 'Status', 'Fulfilment', 'Sales Channel ',
       'ship-service-level', 'Style', 'SKU', 'Category', 'Size', 'ASIN', 'Qty',
       'Amount', 'ship-city', 'ship-state', 'ship-postal-code', 'B2B'],
      dtype='object')
```

1.1.2 Sale Report dataset

[53]:
```
# Read Sale Report csv file

inventory_df = pd.read_csv("Sale Report.csv")

# Display first 5 rows
```

```
inventory_df.head()
```

[53]:

	index	SKU Code	Design No.	Stock	Category	Size	Color
0	0	AN201-RED-L	AN201	5.0	AN : LEGGINGS	L	Red
1	1	AN201-RED-M	AN201	5.0	AN : LEGGINGS	M	Red
2	2	AN201-RED-S	AN201	3.0	AN : LEGGINGS	S	Red
3	3	AN201-RED-XL	AN201	6.0	AN : LEGGINGS	XL	Red
4	4	AN201-RED-XXL	AN201	3.0	AN : LEGGINGS	XXL	Red

[54]:
```
# Dispaly the size and dimension

inventory_df.shape

print("Size of Sale report data : {} records".format(inventory_df.shape[0]))
print("Dimension of Sale report data: {} columns".format(inventory_df.shape[1]))
```

```
Size of Sale report data : 9271 records
Dimension of Sale report data: 7 columns
```

[55]:
```
# Display the information of data

inventory_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9271 entries, 0 to 9270
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   index       9271 non-null   int64
 1   SKU Code    9188 non-null   object
 2   Design No.  9235 non-null   object
 3   Stock       9235 non-null   float64
 4   Category    9226 non-null   object
 5   Size        9235 non-null   object
 6   Color       9226 non-null   object
dtypes: float64(1), int64(1), object(5)
memory usage: 507.1+ KB
```

[56]:
```
# Drop index column

inventory_df.drop(columns="index", inplace=True)
```

[57]:
```
# Checking null values

inventory_df.isnull().sum()
```

[57]:
```
SKU Code      83
Design No.    36
Stock         36
Category      45
Size          36
Color         45
dtype: int64
```

[58]:
```
# Dispaly columns with null values and their respetive percenatge
i = 1
for (col,percentage) in (inventory_df.isnull().sum()/ len(inventory_df) * 100).
 ↪items():
    if percentage > 0:
        print(f"{i}. {col} has {percentage:.2f}% of missing values")
        i+=1
```

1. SKU Code has 0.90% of missing values
2. Design No. has 0.39% of missing values
3. Stock has 0.39% of missing values
4. Category has 0.49% of missing values
5. Size has 0.39% of missing values
6. Color has 0.49% of missing values

[59]:
```
# Dropping the records having missing values

inventory_df = inventory_df.dropna()
```

[60]:
```
# Checking null values

inventory_df.isnull().sum()
```

[60]:
```
SKU Code      0
Design No.    0
Stock         0
Category      0
Size          0
Color         0
dtype: int64
```

[61]:
```
# Save categorical columns (object or category data types)

inventory_cat_cols = inventory_df.select_dtypes(include=["object"]).columns.
 ↪tolist()
```

[62]:
```
# Check the value counts of categorical columns
i=1
for column in inventory_cat_cols:
```

```
    if len(inventory_df[column].value_counts()) < 100:
        print(i,inventory_df[column].value_counts())
        print()
        i+=1
```

```
1 Category
KURTA                   3704
KURTA SET               1596
SET                     1049
TOP                      861
DRESS                    700
BLOUSE                   234
NIGHT WEAR               217
TUNIC                    154
SAREE                    147
AN : LEGGINGS            130
PALAZZO                   91
PANT                      91
CROP TOP                  42
SHARARA                   40
LEHENGA CHOLI             35
KURTI                     28
SKIRT                     20
BOTTOM                    19
CARDIGAN                  16
JUMPSUIT                   7
CROP TOP WITH PLAZZO       7
Name: count, dtype: int64

2 Size
S       1351
M       1341
XL      1339
XXL     1337
L       1335
XS      1110
XXXL    1066
FREE     216
5XL       32
4XL       31
6XL       30
Name: count, dtype: int64

3 Color
Blue               777
Pink               777
Black              654
```

Green	640
Maroon	481
Grey	454
White	409
Teal	383
Mustard	376
Yellow	349
Peach	347
Red	343
Navy Blue	341
Light Green	281
Cream	228
Beige	208
OFF WHITE	206
Multicolor	203
Brown	195
Orange	178
Sea Green	133
Sky Blue	102
Turquoise Blue	100
Wine	87
Turquoise	83
Magenta	70
Light Pink	68
Rust	60
Dark Green	58
Olive	56
Olive Green	52
Purple	50
Dark Blue	47
Turquoise Green	45
Gold	41
TEAL GREEN	35
Mauve	27
Light Blue	21
LIGHT YELLOW	17
TEAL BLUE	17
Indigo	14
Powder Blue	14
Light Brown	14
NAVY	14
CORAL	14
Teal Green	12
BURGUNDY	12
MINT GREEN	11
Lemon Yellow	9
CORAL PINK	7
CORAL ORANGE	7

```
LIME GREEN           7
LEMON                7
LEMON                7
Charcoal             7
Khaki                7
MINT                 7
CORAL                7
AQUA GREEN           5
Taupe                5
Chiku                1
NO REFERENCE         1
Name: count, dtype: int64
```

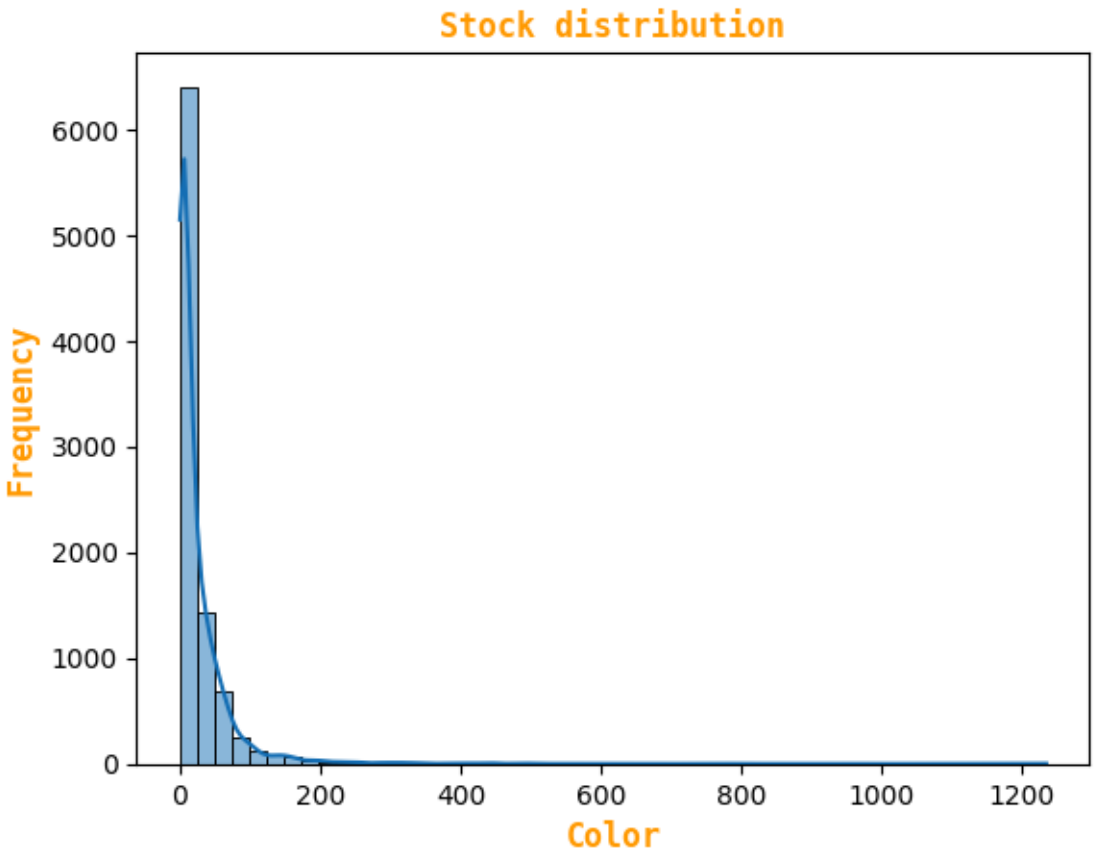
[63]:
```
# Replace category value

inventory_df["Category"] = inventory_df["Category"].replace({"AN : LEGGINGS":
 ↪"LEGGINGS"})
```

[64]:
```
# Change the font to title case

inventory_df["Category"] = inventory_df["Category"].str.title()
inventory_df["Color"] = inventory_df["Color"].str.title()
```

[65]:
```
# Print the number of unique Design no and SKU code

print("Number of unique Design no: ", len(inventory_df["Design No."].unique()))
print("Number of unique SKU: ", len(inventory_df["SKU Code"].unique()))
```

```
Number of unique Design no:  1591
Number of unique SKU:  9170
```

[66]:
```
# Create a histogram using Seaborn

sns.histplot(inventory_df["Stock"], bins=50, kde=True, color=amazon_blue)

# Add labels and a title
plt.xlabel(column, fontdict=font_dict)
plt.ylabel("Frequency", fontdict=font_dict)
plt.title("Stock distribution", fontdict=font_dict)

# Show the plot
plt.show()
```



[67]:
```
# Describing Qty and Amount columns

inventory_df[["Stock"]].describe(percentiles=[0.25,0.5,0.75,0.8,0.9,0.95,0.99])
```

[67]:

	Stock
count	9188.000000
mean	26.378973
std	58.582786
min	0.000000
25%	3.000000
50%	8.000000
75%	31.000000
80%	39.000000
90%	62.000000
95%	94.000000
99%	237.260000
max	1234.000000

[68]:
```
# Checking the records having stock quantity greater than 500

inventory_df[inventory_df["Stock"] > 250].sort_values(by="Stock",␣
 ↪ascending=False)
```

[68]:

	SKU Code	Design No.	Stock	Category	Size
3948	JNE3405-KR-XXL	JNE3405	1234.0	Kurta	XXL
3947	JNE3405-KR-XS	JNE3405	1230.0	Kurta	XS
3300	JNE1525-KR-UDF19BLACK-M	JNE1525	1082.0	Kurta	M
3789	JNE3368-KR-XXXL	JNE3368	985.0	Kurta	XXXL
3495	JNE2270-KR-487-XXL	JNE2270	949.0	Kurta	XXL
8301	SET273-KR-NP-M	SET273	880.0	Kurta Set	M
3299	JNE1525-KR-UDF19BLACK-L	JNE1525	874.0	Kurta	L
8302	SET273-KR-NP-S	SET273	871.0	Kurta Set	S
8304	SET273-KR-NP-XS	SET273	862.0	Kurta Set	XS
3302	JNE1525-KR-UDF19BLACK-XL	JNE1525	844.0	Kurta	XL
8269	SET268-KR-NP-S	SET268	807.0	Kurta Set	S
4972	JNE3613-KR-L	JNE3613	779.0	Kurta	L
3301	JNE1525-KR-UDF19BLACK-S	JNE1525	775.0	Kurta	S
3788	JNE3368-KR-XXL	JNE3368	757.0	Kurta	XXL
7487	SET058-KR-NP-XS	SET058	696.0	Kurta Set	XS
3949	JNE3405-KR-XXXL	JNE3405	677.0	Kurta	XXXL
8300	SET273-KR-NP-L	SET273	637.0	Kurta Set	L
82	BL021-71BLACK	BL021	628.0	Blouse	FREE
4978	JNE3613-KR-XXXL	JNE3613	609.0	Kurta	XXXL
4975	JNE3613-KR-XL	JNE3613	593.0	Kurta	XL
69	BL009-61BLACK	BL009	575.0	Blouse	FREE
3303	JNE1525-KR-UDF19BLACK-XS	JNE1525	563.0	Kurta	XS
8268	SET268-KR-NP-M	SET268	542.0	Kurta Set	M
3946	JNE3405-KR-XL	JNE3405	515.0	Kurta	XL
4977	JNE3613-KR-XXL	JNE3613	504.0	Kurta	XXL
4093	JNE3435-KR-S	JNE3435	490.0	Kurta	S
115	BL061-75BLACK	BL061	488.0	Blouse	FREE
4973	JNE3613-KR-M	JNE3613	485.0	Kurta	M
3786	JNE3368-KR-XL	JNE3368	481.0	Kurta	XL
3491	JNE2270-KR-487-M	JNE2270	478.0	Kurta	M
99	BL035-161GOLD	BL035	448.0	Blouse	FREE
3304	JNE1525-KR-UDF19BLACK-XXL	JNE1525	448.0	Kurta	XXL
3943	JNE3405-KR-L	JNE3405	446.0	Kurta	L
8303	SET273-KR-NP-XL	SET273	445.0	Kurta Set	XL
2532	J0341-DR-S	J0341	441.0	Dress	S
3944	JNE3405-KR-M	JNE3405	432.0	Kurta	M
3783	JNE3368-KR-L	JNE3368	429.0	Kurta	L
3945	JNE3405-KR-S	JNE3405	423.0	Kurta	S
3490	JNE2270-KR-487-L	JNE2270	414.0	Kurta	L
3294	JNE1408-GREY-KR-UDF19-S	JNE1408	404.0	Kurta	S
8271	SET268-KR-NP-XS	SET268	403.0	Kurta Set	XS

72	BL013-62BLACK	BL013	399.0	Blouse	FREE
295	BTM005-L	BTM005	398.0	Pant	L
8267	SET268-KR-NP-L	SET268	384.0	Kurta Set	L
4797	JNE3567-KR-L	JNE3567	377.0	Kurta	L
7852	SET197-KR-NP-L	SET197	358.0	Kurta Set	L
4930	JNE3607-KR-L	JNE3607	350.0	Kurta	L
297	BTM005-S	BTM005	347.0	Pant	S
4132	JNE3440-KR-S	JNE3440	345.0	Kurta	S
4800	JNE3567-KR-XL	JNE3567	336.0	Kurta	XL
3493	JNE2270-KR-487-XL	JNE2270	333.0	Kurta	XL
8241	SET264-KR-NP-S	SET264	330.0	Kurta Set	S
2536	J0341-DR-XXXL	J0341	329.0	Dress	XXXL
5600	JNE3720-KR-M	JNE3720	326.0	Kurta	M
7485	SET058-KR-NP-S	SET058	326.0	Kurta Set	S
7797	SET187-KR-DH-M	SET187	318.0	Kurta Set	M
654	J0008-SKD-XL	J0008	318.0	Kurta Set	XL
4933	JNE3607-KR-XL	JNE3607	317.0	Kurta	XL
5183	JNE3645-TP-S	JNE3645	315.0	Top	S
5599	JNE3720-KR-L	JNE3720	312.0	Kurta	L
8275	SET269-KR-NP-M	SET269	310.0	Set	M
3762	JNE3364-KR-1051-M	JNE3364	306.0	Kurta	M
4960	JNE3611-KR-S	JNE3611	304.0	Kurta	S
3785	JNE3368-KR-S	JNE3368	296.0	Kurta	S
8240	SET264-KR-NP-M	SET264	295.0	Kurta Set	M
4802	JNE3567-KR-XXL	JNE3567	295.0	Kurta	XXL
7770	SET183-KR-DH-S	SET183	293.0	Kurta Set	S
8630	SET329-KR-NP-M	SET329	292.0	Kurta Set	M
7510	SET073-KR-SHA-XS	SET073	289.0	Kurta Set	XS
5185	JNE3645-TP-XS	JNE3645	288.0	Top	XS
3519	JNE2305-KR-533-XL	JNE2305	284.0	Kurta	XL
3516	JNE2305-KR-533-L	JNE2305	284.0	Kurta	L
4935	JNE3607-KR-XXL	JNE3607	281.0	Kurta	XXL
4092	JNE3435-KR-M	JNE3435	268.0	Kurta	M
4958	JNE3611-KR-L	JNE3611	267.0	Kurta	L
8740	SET345-KR-NP-S	SET345	266.0	Kurta Set	S
3296	JNE1408-GREY-KR-UDF19-XS	JNE1408	262.0	Kurta	XS
4962	JNE3611-KR-XS	JNE3611	260.0	Kurta	XS
4974	JNE3613-KR-S	JNE3613	259.0	Kurta	S
4931	JNE3607-KR-M	JNE3607	257.0	Kurta	M
5602	JNE3720-KR-XL	JNE3720	255.0	Kurta	XL
8631	SET329-KR-NP-S	SET329	252.0	Kurta Set	S

	Color
3948	Pink
3947	Pink
3300	Black
3789	Light Green

```
3495        Beige
8301        Brown
3299        Black
8302        Brown
8304        Brown
3302        Black
8269    Off White
4972         Teal
3301        Black
3788  Light Green
7487         Teal
3949         Pink
8300        Brown
82          Black
4978         Teal
4975         Teal
69          Black
3303        Black
8268    Off White
3946         Pink
4977         Teal
4093         Pink
115         Black
4973         Teal
3786  Light Green
3491        Beige
99           Gold
3304        Black
3943         Pink
8303        Brown
2532         Blue
3944         Pink
3783  Light Green
3945         Pink
3490        Beige
3294         Grey
8271    Off White
72          Black
295         Black
8267    Off White
4797         Blue
7852      Mustard
4930       Maroon
297         Black
4132        White
4800         Blue
3493        Beige
```

```
8241         Gold
2536         Blue
5600    Navy Blue
7485         Teal
7797        Peach
654       Mustard
4933       Maroon
5183          Red
5599    Navy Blue
8275        Green
3762  Light Green
4960        Black
3785  Light Green
8240         Gold
4802         Blue
7770  Olive Green
8630        Peach
7510   Dark Green
5185          Red
3519  Light Brown
3516  Light Brown
4935       Maroon
4092         Pink
4958        Black
8740         Pink
3296         Grey
4962        Black
4974         Teal
4931       Maroon
5602    Navy Blue
8631        Peach
```

[69]:
```
# Output cleaned dataset

inventory_df.to_csv("Inventory_cleaned_dataset.csv")
```

1.1.3 P L March 2021 dataset

[70]:
```
# Read P L March 2021 csv file

March2021_df = pd.read_csv("P  L March 2021.csv")

# Display first 5 rows

March2021_df.head()
```

[70]:

	index	Sku	Style Id	Catalog	Category	Weight	TP 1	TP 2
0	0	Os206_3141_S	Os206_3141	Moments	Kurta	0.3	538	435.78
1	1	Os206_3141_M	Os206_3141	Moments	Kurta	0.3	538	435.78
2	2	Os206_3141_L	Os206_3141	Moments	Kurta	0.3	538	435.78
3	3	Os206_3141_XL	Os206_3141	Moments	Kurta	0.3	538	435.78
4	4	Os206_3141_2XL	Os206_3141	Moments	Kurta	0.3	538	435.78

	MRP Old Final	MRP Old Ajio	MRP Amazon	MRP Amazon FBA	MRP Flipkart	MRP
0	2178	2295	2295	2295	2295	2295
1	2178	2295	2295	2295	2295	2295
2	2178	2295	2295	2295	2295	2295
3	2178	2295	2295	2295	2295	2295
4	2178	2295	2295	2295	2295	2295

	Limeroad MRP	Myntra MRP	Paytm MRP	Snapdeal MRP
0	2295	2295	2295	2295
1	2295	2295	2295	2295
2	2295	2295	2295	2295
3	2295	2295	2295	2295
4	2295	2295	2295	2295

[71]:
```
# Dispaly the size and dimension

March2021_df.shape

print("Size of March 2021 data : {} records".format(March2021_df.shape[0]))
print("Dimension of March 2021 data: {} columns".format(March2021_df.shape[1]))
```

```
Size of March 2021 data : 1330 records
Dimension of March 2021 data: 18 columns
```

[72]:
```
# Display the information of data

March2021_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1330 entries, 0 to 1329
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           1330 non-null   int64
 1   Sku             1330 non-null   object
 2   Style Id        1330 non-null   object
 3   Catalog         1330 non-null   object
 4   Category        1330 non-null   object
 5   Weight          1330 non-null   object
 6   TP 1            1330 non-null   object
 7   TP 2            1330 non-null   object
```

```
 8   MRP Old          1330 non-null   object
 9   Final MRP Old    1330 non-null   object
 10  Ajio MRP         1330 non-null   object
 11  Amazon MRP       1330 non-null   object
 12  Amazon FBA MRP   1330 non-null   object
 13  Flipkart MRP     1330 non-null   object
 14  Limeroad MRP     1330 non-null   object
 15  Myntra MRP       1330 non-null   object
 16  Paytm MRP        1330 non-null   object
 17  Snapdeal MRP     1330 non-null   object
dtypes: int64(1), object(17)
memory usage: 187.2+ KB
```

[73]:
```
# Drop index column

March2021_df.drop(columns="index", inplace=True)
```

[74]:
```
# Re-name column TP1 and TP2

March2021_df.rename(columns={"TP 1": "First_third_party_platform_price", "TP 2":
 ↪ "Second_third_party_platform_price"}, inplace=True)
```

[75]:
```
# Checking null values

March2021_df.isnull().sum()
```

[75]:
```
Sku                                0
Style Id                           0
Catalog                            0
Category                           0
Weight                             0
First_third_party_platform_price   0
Second_third_party_platform_price  0
MRP Old                            0
Final MRP Old                      0
Ajio MRP                           0
Amazon MRP                         0
Amazon FBA MRP                     0
Flipkart MRP                       0
Limeroad MRP                       0
Myntra MRP                         0
Paytm MRP                          0
Snapdeal MRP                       0
dtype: int64
```

[76]:
```
March2021_df[March2021_df["First_third_party_platform_price"] == "#VALUE!"]
```

[76]:
```
              Sku Style Id Catalog   Category Weight  \
1230    Os1041_S   Os1041     Mix  Kurta Set    0.4
1231    Os1041_M   Os1041     Mix  Kurta Set    0.4
1232    Os1041_L   Os1041     Mix  Kurta Set    0.4
1233   Os1041_XL   Os1041     Mix  Kurta Set    0.4
1234  Os1041_2XL   Os1041     Mix  Kurta Set    0.4
1235  Os1041_3XL   Os1041     Mix  Kurta Set    0.4

     First_third_party_platform_price Second_third_party_platform_price  \
1230                          #VALUE!                           #VALUE!
1231                          #VALUE!                           #VALUE!
1232                          #VALUE!                           #VALUE!
1233                          #VALUE!                           #VALUE!
1234                          #VALUE!                           #VALUE!
1235                          #VALUE!                           #VALUE!

     MRP Old Final MRP Old Ajio MRP Amazon MRP Amazon FBA MRP Flipkart MRP  \
1230    Nill          Nill     Nill       Nill           Nill         Nill
1231    Nill          Nill     Nill       Nill           Nill         Nill
1232    Nill          Nill     Nill       Nill           Nill         Nill
1233    Nill          Nill     Nill       Nill           Nill         Nill
1234    Nill          Nill     Nill       Nill           Nill         Nill
1235    Nill          Nill     Nill       Nill           Nill         Nill

     Limeroad MRP Myntra MRP Paytm MRP Snapdeal MRP
1230         Nill       2895      Nill         Nill
1231         Nill       2895      Nill         Nill
1232         Nill       2895      Nill         Nill
1233         Nill       2895      Nill         Nill
1234         Nill       2895      Nill         Nill
1235         Nill       2895      Nill         Nill
```

[77]:
```
# Converting the column type to integer
columns_to_convert = ["Weight",
    "First_third_party_platform_price",
    "Second_third_party_platform_price",
    "MRP Old",
    "Final MRP Old",
    "Ajio MRP",
    "Amazon MRP",
    "Amazon FBA MRP",
    "Flipkart MRP",
    "Limeroad MRP",
    "Myntra MRP",
    "Paytm MRP",
    "Snapdeal MRP"
]
```

```
for col in columns_to_convert:
    March2021_df[col] = pd.to_numeric(March2021_df[col], errors='coerce')
```

[78]:
```
# Checking null values

March2021_df.isnull().sum()
```

[78]:
```
Sku                                  0
Style Id                             0
Catalog                              0
Category                             0
Weight                              73
First_third_party_platform_price     6
Second_third_party_platform_price    6
MRP Old                             37
Final MRP Old                       37
Ajio MRP                            37
Amazon MRP                          37
Amazon FBA MRP                      37
Flipkart MRP                        37
Limeroad MRP                        37
Myntra MRP                          31
Paytm MRP                           37
Snapdeal MRP                        37
dtype: int64
```

[79]:
```
# Dispaly columns with null values and their respetive percenatge
i = 1
for (col,percentage) in (March2021_df.isnull().sum()/ len(March2021_df) * 100).
 ↪items():
    if percentage > 0:
        print(f"{i}. {col} has {percentage:.2f}% of missing values")
        i+=1
```

1. Weight has 5.49% of missing values
2. First_third_party_platform_price has 0.45% of missing values
3. Second_third_party_platform_price has 0.45% of missing values
4. MRP Old has 2.78% of missing values
5. Final MRP Old has 2.78% of missing values
6. Ajio MRP has 2.78% of missing values
7. Amazon MRP has 2.78% of missing values
8. Amazon FBA MRP has 2.78% of missing values
9. Flipkart MRP has 2.78% of missing values
10. Limeroad MRP has 2.78% of missing values
11. Myntra MRP has 2.33% of missing values
12. Paytm MRP has 2.78% of missing values

13. Snapdeal MRP has 2.78% of missing values

[80]:
```
# Dropping the records having missing values

March2021_df = March2021_df.dropna()
```

[81]:
```
# Check the value counts of categorical columns

March_2021_cat_cols = ["Style Id", "Catalog", "Category"]
i=1
for column in March_2021_cat_cols:
    if len(March2021_df[column].value_counts()) < 100:
        print(i,March2021_df[column].value_counts())
        print()
        i+=1
```

```
1 Catalog
Mix            794
Surmaya        192
Colors-8        48
Rozana          48
Colors-7        43
Moments         36
Breeze-4        36
Four Gems 2     24
Name: count, dtype: int64

2 Category
Kurta        802
Kurta Set    342
Tops          45
Gown          32
Name: count, dtype: int64
```

[82]:
```
March2021_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1221 entries, 0 to 1329
Data columns (total 17 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Sku                                 1221 non-null   object
 1   Style Id                            1221 non-null   object
 2   Catalog                             1221 non-null   object
 3   Category                            1221 non-null   object
 4   Weight                              1221 non-null   float64
 5   First_third_party_platform_price    1221 non-null   float64
```

```
 6   Second_third_party_platform_price  1221 non-null   float64
 7   MRP Old                            1221 non-null   float64
 8   Final MRP Old                      1221 non-null   float64
 9   Ajio MRP                           1221 non-null   float64
 10  Amazon MRP                         1221 non-null   float64
 11  Amazon FBA MRP                     1221 non-null   float64
 12  Flipkart MRP                       1221 non-null   float64
 13  Limeroad MRP                       1221 non-null   float64
 14  Myntra MRP                         1221 non-null   float64
 15  Paytm MRP                          1221 non-null   float64
 16  Snapdeal MRP                       1221 non-null   float64
dtypes: float64(13), object(4)
memory usage: 171.7+ KB
```

[83]:
```
# Output cleaned dataset

March2021_df.to_csv("March2021_cleaned_dataset.csv")
```

1.1.4 May 2022 dataset

[84]:
```
# Read P L March csv file

May2022_df = pd.read_csv("May-2022.csv")

# Display first 5 rows

May2022_df.head()
```

[84]:

	index	Sku	Style Id	Catalog	Category	Weight	TP	MRP Old	Final MRP Old	Ajio MRP	Amazon MRP	Amazon FBA MRP	Flipkart MRP	Limeroad MRP	Myntra MRP	Paytm MRP	Snapdeal MRP
0	0	Os206_3141_S	Os206_3141	Moments	Kurta	0.3	538	2178	2295	2295	2295	2295	2295	2295	2295	2295	2295
1	1	Os206_3141_M	Os206_3141	Moments	Kurta	0.3	538	2178	2295	2295	2295	2295	2295	2295	2295	2295	2295
2	2	Os206_3141_L	Os206_3141	Moments	Kurta	0.3	538	2178	2295	2295	2295	2295	2295	2295	2295	2295	2295
3	3	Os206_3141_XL	Os206_3141	Moments	Kurta	0.3	538	2178	2295	2295	2295	2295	2295	2295	2295	2295	2295
4	4	Os206_3141_2XL	Os206_3141	Moments	Kurta	0.3	538	2178	2295	2295	2295	2295	2295	2295	2295	2295	2295

```
4        2295        2295           2295
```

[85]:
```
# Dispaly the size and dimension

May2022_df.shape

print("Size of May 2020 data : {} records".format(May2022_df.shape[0]))
print("Dimension of May 2020 data: {} columns".format(May2022_df.shape[1]))
```

```
Size of May 2020 data : 1330 records
Dimension of May 2020 data: 17 columns
```

[86]:
```
# Display the information of data

May2022_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1330 entries, 0 to 1329
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   index           1330 non-null   int64
 1   Sku             1330 non-null   object
 2   Style Id        1330 non-null   object
 3   Catalog         1330 non-null   object
 4   Category        1330 non-null   object
 5   Weight          1330 non-null   object
 6   TP              1330 non-null   object
 7   MRP Old         1330 non-null   object
 8   Final MRP Old   1330 non-null   object
 9   Ajio MRP        1330 non-null   object
 10  Amazon MRP      1330 non-null   object
 11  Amazon FBA MRP  1330 non-null   object
 12  Flipkart MRP    1330 non-null   object
 13  Limeroad MRP    1330 non-null   object
 14  Myntra MRP      1330 non-null   object
 15  Paytm MRP       1330 non-null   object
 16  Snapdeal MRP    1330 non-null   object
dtypes: int64(1), object(16)
memory usage: 176.8+ KB
```

[87]:
```
# Drop index column

May2022_df.drop(columns="index", inplace=True)
```

[88]:
```
# Re-name column TP
```

```
May2022_df.rename(columns={"TP": "original_prod_price"}, inplace=True)
```

[89]:
```
# Checking null values

May2022_df.isnull().sum()
```

[89]:
```
Sku                   0
Style Id              0
Catalog               0
Category              0
Weight                0
original_prod_price   0
MRP Old               0
Final MRP Old         0
Ajio MRP              0
Amazon MRP            0
Amazon FBA MRP        0
Flipkart MRP          0
Limeroad MRP          0
Myntra MRP            0
Paytm MRP             0
Snapdeal MRP          0
dtype: int64
```

[90]:
```
# Removing records with junk values in price columns

May2022_df = May2022_df[~(May2022_df["original_prod_price"] == "#VALUE!")]
```

[91]:
```
# Converting the column type to integer
columns_to_convert = ["Weight",
    "original_prod_price",
    "MRP Old",
    "Final MRP Old",
    "Ajio MRP",
    "Amazon MRP",
    "Amazon FBA MRP",
    "Flipkart MRP",
    "Limeroad MRP",
    "Myntra MRP",
    "Paytm MRP",
    "Snapdeal MRP"
]

for col in columns_to_convert:
    May2022_df[col] = pd.to_numeric(May2022_df[col], errors='coerce')
```

[92]:
```
# Dispaly columns with null values and their respetive percenatge
i = 1
for (col,percentage) in (May2022_df.isnull().sum()/ len(May2022_df) * 100).
 ↪items():
    if percentage > 0:
        print(f"{i}. {col} has {percentage:.2f}% of missing values")
        i+=1
```

```
1. Weight has 5.51% of missing values
2. MRP Old has 2.34% of missing values
3. Final MRP Old has 2.34% of missing values
4. Ajio MRP has 2.34% of missing values
5. Amazon MRP has 2.34% of missing values
6. Amazon FBA MRP has 2.34% of missing values
7. Flipkart MRP has 2.34% of missing values
8. Limeroad MRP has 2.34% of missing values
9. Myntra MRP has 2.34% of missing values
10. Paytm MRP has 2.34% of missing values
11. Snapdeal MRP has 2.34% of missing values
```

[93]:
```
# Dropping the records having missing values

May2022_df = May2022_df.dropna()
```

[94]:
```
May2022_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1221 entries, 0 to 1329
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sku                  1221 non-null   object
 1   Style Id             1221 non-null   object
 2   Catalog              1221 non-null   object
 3   Category             1221 non-null   object
 4   Weight               1221 non-null   float64
 5   original_prod_price  1221 non-null   float64
 6   MRP Old              1221 non-null   float64
 7   Final MRP Old        1221 non-null   float64
 8   Ajio MRP             1221 non-null   float64
 9   Amazon MRP           1221 non-null   float64
 10  Amazon FBA MRP       1221 non-null   float64
 11  Flipkart MRP         1221 non-null   float64
 12  Limeroad MRP         1221 non-null   float64
 13  Myntra MRP           1221 non-null   float64
 14  Paytm MRP            1221 non-null   float64
 15  Snapdeal MRP         1221 non-null   float64
```

```
dtypes: float64(12), object(4)
memory usage: 162.2+ KB
```

[95]:
```
# Check the value counts of categorical columns

May_2022_cat_cols = ["Style Id", "Catalog", "Category"]
i=1
for column in May_2022_cat_cols:
    if len(May2022_df[column].value_counts()) < 100:
        print(i,May2022_df[column].value_counts())
        print()
        i+=1
```

```
1 Catalog
Mix            794
Surmaya        192
Colors-8        48
Rozana          48
Colors-7        43
Moments         36
Breeze-4        36
Four Gems 2     24
Name: count, dtype: int64

2 Category
Kurta        802
Kurta Set    342
Tops          45
Gown          32
Name: count, dtype: int64
```

[96]:
```
# Output cleaned dataset

May2022_df.to_csv("May2022_cleaned_dataset.csv")
```

1.1.5 International Sale Report dataset

[144]:
```
# Read International Sale Report csv file

international_df = pd.read_csv("International Sale Report.csv")

# Display first 5 rows

international_df.head()
```

[144]:
	index	DATE	Months	CUSTOMER	Style	SKU	Size	\
0	0	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5004	MEN5004-KR-L	L	
1	1	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5004	MEN5004-KR-XL	XL	
2	2	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5004	MEN5004-KR-XXL	XXL	
3	3	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5009	MEN5009-KR-L	L	
4	4	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5011	MEN5011-KR-L	L	

	PCS	RATE	GROSS AMT
0	1.00	616.56	617.00
1	1.00	616.56	617.00
2	1.00	616.56	617.00
3	1.00	616.56	617.00
4	1.00	616.56	617.00

[145]:
```
# Dispaly the size and dimension

international_df.shape

print("Size of May 2020 data : {} records".format(international_df.shape[0]))
print("Dimension of May 2020 data: {} columns".format(international_df.
 ↪shape[1]))
```

```
Size of May 2020 data : 37432 records
Dimension of May 2020 data: 10 columns
```

[146]:
```
# Display the information of data

international_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37432 entries, 0 to 37431
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   index      37432 non-null  int64
 1   DATE       37431 non-null  object
 2   Months     37407 non-null  object
 3   CUSTOMER   36392 non-null  object
 4   Style      36392 non-null  object
 5   SKU        34958 non-null  object
 6   Size       36392 non-null  object
 7   PCS        36392 non-null  object
 8   RATE       36392 non-null  object
 9   GROSS AMT  36392 non-null  object
dtypes: int64(1), object(9)
memory usage: 2.9+ MB
```

[147]:
```
# Drop index column

international_df.drop(columns=["index"], inplace=True)
```

[148]:
```
international_df[international_df['Style'] == 'SHIPPING'].head(10)
```

[148]:

	DATE	Months	CUSTOMER	Style	SKU	Size	PCS	RATE	GROSS AMT
19696	REVATHY LOGANATHAN	06-05-21	Jun-21	SHIPPING	SHIPPING	1.00	2250.00	2250.00	0.00
19751	FARIA ESSOPP	06-08-21	Jun-21	SHIPPING	SHIPPING	1.00	11000.00	11000.00	0.00
19816	MANGALAM SHOP	06-11-21	Jun-21	SHIPPING	SHIPPING	1.00	8600.00	8600.00	0.00
19845	THANA NAGISSWARY L MARIMUTHU	06-15-21	Jun-21	SHIPPING	SHIPPING	1.00	5800.00	5800.00	0.00
19865	REVATHY LOGANATHAN	06-17-21	Jun-21	SHIPPING	SHIPPING	1.00	2200.00	2200.00	0.00
19877	MR.ALWAR MURALI	06-18-21	Jun-21	SHIPPING	SHIPPING	1.00	2100.00	2100.00	0.00
19898	RAZIA ROSEANE NASER	06-18-21	Jun-21	SHIPPING	SHIPPING	1.00	5100.00	5100.00	0.00
19913	SIRI PADALA	06-19-21	Jun-21	SHIPPING	SHIPPING	1.00	2500.00	2500.00	0.00
19943	FUSION FASHIONS CORP.	06-23-21	Jun-21	SHIPPING	SHIPPING	1.00	10560.00	10560.00	0.00
19969	MIZNA WAHEEDH	06-23-21	Jun-21	SHIPPING	SHIPPING	1.00	6020.00	6020.00	0.00

[149]:
```
def modify_dataframe(df):
    error_list = []

    # Step 1: Check if Style is "Shipping" and store corresponding DATE values in error_list
    error_list = df[df['Style'] == 'SHIPPING']['DATE'].tolist()

    #print(df[df["DATE"].isin(error_list)].shape)
    # Print the shape of the DataFrame with matching 'DATE' values
    matching_df_shape = df[df['DATE'].isin(error_list)].shape
    print(f"Shape of DataFrame with matching 'DATE' values: {matching_df_shape}")

    # Step 2: Replace values as described
    df.loc[df['DATE'].isin(error_list), 'CUSTOMER'], df.loc[df['DATE'].isin(error_list), 'DATE'] = df['DATE'], df['Months']
```

```
    df.loc[df['DATE'].isin(error_list), 'Months'], df.loc[df['DATE'].
 ↪isin(error_list), 'CUSTOMER'] = df['CUSTOMER'], df['Months']
    df.loc[df['DATE'].isin(error_list), 'PCS'], df.loc[df['DATE'].
 ↪isin(error_list), 'RATE'] = df['GROSS AMT'], df['RATE']
    df.loc[df['DATE'].isin(error_list), 'GROSS AMT'], df.loc[df['DATE'].
 ↪isin(error_list), 'RATE'] = df['RATE'], df['PCS']

    # Step 3: Replace the initial values of CUSTOMER and PCS in Months and PCS␣
 ↪columns
    df.loc[df['DATE'].isin(error_list), 'Months'], df.loc[df['DATE'].
 ↪isin(error_list), 'RATE'] = df['CUSTOMER'], df['PCS']
    df.loc[df['DATE'].isin(error_list), 'CUSTOMER'], df.loc[df['DATE'].
 ↪isin(error_list), 'PCS'] = '', ''

    return df
```

[150]:
```
error_list = international_df[international_df['Style'] == 'SHIPPING']['DATE'].
 ↪tolist()

#print(df[df["DATE"].isin(error_list)].shape)
# Print the shape of the DataFrame with matching 'DATE' values
matching_df_shape = international_df[international_df['DATE'].isin(error_list)].
 ↪shape
print(f"Shape of DataFrame with matching 'DATE' values: {matching_df_shape}")
```

```
Shape of DataFrame with matching 'DATE' values: (16554, 9)
```

[151]:
```
# Call the function to modify the DataFrame
international_df = modify_dataframe(international_df)
```

```
Shape of DataFrame with matching 'DATE' values: (16554, 9)
```

[152]:
```
international_df.head()
```

[152]:

	DATE	Months	CUSTOMER	Style	SKU	Size	PCS	RATE	GROSS AMT
0	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5004	MEN5004-KR-L	L	1.00	616.56	617.00
1	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5004	MEN5004-KR-XL	XL	1.00	616.56	617.00
2	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5004	MEN5004-KR-XXL	XXL	1.00	616.56	617.00
3	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5009	MEN5009-KR-L	L	1.00	616.56	617.00
4	06-05-21	Jun-21	REVATHY LOGANATHAN	MEN5011	MEN5011-KR-L	L	1.00	616.56	617.00

[153]:
```
international_df[international_df["CUSTOMER"] == "SIRI PADALA"].tail(10)
```

[153]:

	DATE	Months	CUSTOMER	Style	SKU	Size	PCS	RATE	GROSS AMT
19904	06-19-21	06-19-21	SIRI PADALA	J0138	J0138-KR-S	1.00	649.35	649.00	4.00
19905	06-19-21	06-19-21	SIRI PADALA	J0148	J0148-SET-S	1.00	876.85	877.00	4.00
19906	06-19-21	06-19-21	SIRI PADALA	J0207	J0207-DR-S	1.00	811.85	812.00	1.00
19907	06-19-21	06-19-21	SIRI PADALA	MEN5029	MEN5029-KR-M	1.00	616.56	617.00	41.00
19908	06-19-21	06-19-21	SIRI PADALA	MEN5008	MEN5008-KR-M	1.00	616.56	617.00	9.00
19909	06-19-21	06-19-21	SIRI PADALA	MEN5003	MEN5003-KR-M	1.00	616.56	617.00	35.00
19910	06-19-21	06-19-21	SIRI PADALA	MEN5013	MEN5013-KR-M	1.00	649.03	649.00	7.00
19911	06-19-21	06-19-21	SIRI PADALA	J0202	J0202-TP-S	1.00	649.35	649.00	4.00
19912	06-19-21	06-19-21	SIRI PADALA	J0213	J0213-TP-S	1.00	584.35	584.00	31.00
19913	06-19-21	06-19-21	SIRI PADALA	SHIPPING	SHIPPING	1.00	2500.00	2500.00	0.00

[154]:
```
international_df["Size"] = international_df["Size"].apply(lambda x: 'Missing'␣
↪if str(x).replace(".", "", 1).isdigit() else x)
```

[155]:
```
international_df[international_df["CUSTOMER"] == "SIRI PADALA"].tail(10)
```

[155]:

	DATE	Months	CUSTOMER	Style	SKU	Size	PCS	RATE	GROSS AMT
19904	06-19-21	06-19-21	SIRI PADALA	J0138	J0138-KR-S	Missing	649.35	649.00	4.00
19905	06-19-21	06-19-21	SIRI PADALA	J0148	J0148-SET-S	Missing	876.85	877.00	4.00
19906	06-19-21	06-19-21	SIRI PADALA	J0207	J0207-DR-S	Missing	811.85	812.00	1.00
19907	06-19-21	06-19-21	SIRI PADALA	MEN5029	MEN5029-KR-M	Missing	616.56	617.00	41.00
19908	06-19-21	06-19-21	SIRI PADALA	MEN5008	MEN5008-KR-M	Missing	616.56	617.00	9.00
19909	06-19-21	06-19-21	SIRI PADALA	MEN5003	MEN5003-KR-M	Missing	616.56	617.00	35.00
19910	06-19-21	06-19-21	SIRI PADALA	MEN5013	MEN5013-KR-M	Missing	649.03	649.00	7.00
19911	06-19-21	06-19-21	SIRI PADALA	J0202	J0202-TP-S	Missing	649.35	649.00	4.00
19912	06-19-21	06-19-21	SIRI PADALA	J0213	J0213-TP-S	Missing	584.35	584.00	31.00
19913	06-19-21	06-19-21	SIRI PADALA	SHIPPING	SHIPPING	Missing	2500.00	2500.00	0.00

```
19907   616.56   617.00     41.00
19908   616.56   617.00      9.00
19909   616.56   617.00     35.00
19910   649.03   649.00      7.00
19911   649.35   649.00      4.00
19912   584.35   584.00     31.00
19913  2500.00  2500.00      0.00
```

[156]:
```
# Drop rows where the 'SKU' column is "SHIPPING"
international_df = international_df[international_df["SKU"] != 'SHIPPING']
```

[157]:
```
# Apply the lambda function to the DataFrame
international_df["Size"] = international_df.apply(lambda row:␣
 ↪str(row["SKU"])[-1] if row["Size"] == "Missing" else row["Size"], axis=1)
```

[158]:
```
international_df[international_df["CUSTOMER"] == "SIRI PADALA"].tail(10)
```

[158]:
```
           DATE    Months     CUSTOMER    Style           SKU Size      PCS  \
19903  06-19-21  06-19-21  SIRI PADALA    J0279   J0279-SET-S    S  1039.35
19904  06-19-21  06-19-21  SIRI PADALA    J0138    J0138-KR-S    S   649.35
19905  06-19-21  06-19-21  SIRI PADALA    J0148   J0148-SET-S    S   876.85
19906  06-19-21  06-19-21  SIRI PADALA    J0207    J0207-DR-S    S   811.85
19907  06-19-21  06-19-21  SIRI PADALA  MEN5029  MEN5029-KR-M    M   616.56
19908  06-19-21  06-19-21  SIRI PADALA  MEN5008  MEN5008-KR-M    M   616.56
19909  06-19-21  06-19-21  SIRI PADALA  MEN5003  MEN5003-KR-M    M   616.56
19910  06-19-21  06-19-21  SIRI PADALA  MEN5013  MEN5013-KR-M    M   649.03
19911  06-19-21  06-19-21  SIRI PADALA    J0202    J0202-TP-S    S   649.35
19912  06-19-21  06-19-21  SIRI PADALA    J0213    J0213-TP-S    S   584.35

          RATE GROSS AMT
19903  1039.00     48.00
19904   649.00      4.00
19905   877.00      4.00
19906   812.00      1.00
19907   617.00     41.00
19908   617.00      9.00
19909   617.00     35.00
19910   649.00      7.00
19911   649.00      4.00
19912   584.00     31.00
```

[159]:
```
international_df[international_df["Size"] == "9"]
```

[159]:
```
           DATE    Months                    CUSTOMER   Style  \
25910  11-01-21  11-01-21  RANDHIR CHAUDHARY  SAR079
27786  01-28-22  01-28-22    THANA MARIMUTHU    CMB5
27788  01-28-22  01-28-22    THANA MARIMUTHU    CMB5
```

28001	02-05-22	02-05-22	THILAS BOMBAY BOUTIQUE SDN BHD	CMB5
28367	02-12-22	02-12-22	VAHARSHA BOUTIQUE	CMB5
28374	02-12-22	02-12-22	VAHARSHA BOUTIQUE	CMB5
28455	02-14-22	02-14-22	SINDHU	SAR049
28905	02-22-22	02-22-22	MR.JEYARAJ	CMB5
29916	03-03-22	03-03-22	IBRAHIM MAFTHOOH(JANASYA.IN)	CMB5
29923	03-03-22	03-03-22	IBRAHIM MAFTHOOH(JANASYA.IN)	CMB5
31775	11-01-21	11-01-21	RANDHIR CHAUDHARY	SAR079
33651	01-28-22	01-28-22	THANA MARIMUTHU	CMB5
33653	01-28-22	01-28-22	THANA MARIMUTHU	CMB5
33866	02-05-22	02-05-22	THILAS BOMBAY BOUTIQUE SDN BHD	CMB5
34232	02-12-22	02-12-22	VAHARSHA BOUTIQUE	CMB5
34239	02-12-22	02-12-22	VAHARSHA BOUTIQUE	CMB5
34320	02-14-22	02-14-22	SINDHU	SAR049
34770	02-22-22	02-22-22	MR.JEYARAJ	CMB5
35781	03-03-22	03-03-22	IBRAHIM MAFTHOOH(JANASYA.IN)	CMB5
35788	03-03-22	03-03-22	IBRAHIM MAFTHOOH(JANASYA.IN)	CMB5
36254	03-03-22	03-03-22	IBRAHIM MAFTHOOH(JANASYA.IN)	CMB5
36261	03-03-22	03-03-22	IBRAHIM MAFTHOOH(JANASYA.IN)	CMB5
36614	03-10-22	03-10-22	FUSION FASHIONS CORP.	SAR029
36621	03-12-22	03-12-22	MRS. SUMITHA RAJU	CMB5

	SKU	Size	PCS	RATE	GROSS AMT
25910	SAR079	9	1903.75	3693.00	0.00
27786	CMB5-JNE3829	9	299.00	1495.00	0.00
27788	CMB5-JNE3849	9	299.00	1495.00	0.00
28001	CMB5-JNE3849	9	299.00	1495.00	0.00
28367	CMB5-JNE3849	9	299.00	4485.00	0.00
28374	CMB5-JNE3839	9	374.00	3740.00	0.00
28455	SAR049	9	598.50	598.00	92.00
28905	CMB5-JNE3839	9	374.00	1870.00	0.00
29916	CMB5-JNE3839	9	374.00	1776.00	0.00
29923	CMB5-JNE3849	9	299.00	1420.00	0.00
31775	SAR079	9	1903.75	3693.00	0.00
33651	CMB5-JNE3829	9	299.00	1495.00	0.00
33653	CMB5-JNE3849	9	299.00	1495.00	0.00
33866	CMB5-JNE3849	9	299.00	1495.00	0.00
34232	CMB5-JNE3849	9	299.00	4485.00	0.00
34239	CMB5-JNE3839	9	374.00	3740.00	0.00
34320	SAR049	9	598.50	598.00	92.00
34770	CMB5-JNE3839	9	374.00	1870.00	0.00
35781	CMB5-JNE3839	9	374.00	1776.00	0.00
35788	CMB5-JNE3849	9	299.00	1420.00	0.00
36254	CMB5-JNE3839	9	374.00	1776.00	0.00
36261	CMB5-JNE3849	9	299.00	1420.00	0.00
36614	SAR029	9	661.25	661.00	0.00
36621	CMB5-JNE3849	9	299.00	1495.00	0.00

[160]:
```
international_df["Size"].value_counts()
```

[160]:
```
Size
L           14349
M            6420
S            5643
XL           3256
XXL          2684
XXXL         1653
XS            879
Free          241
.             132
5XL           117
6XL           109
4XL           100
9              24
8              23
E              21
0              19
I              19
5              17
3              17
n              17
6              16
7              15
C              14
4              12
D              11
K              10
2              10
S TO XXL        9
1               9
FREE            8
s               6
xxxl            4
xxl             4
xl              4
l               4
m               4
)               2
G               1
PCS             1
Name: count, dtype: int64
```

[163]:
```
# Capitalize the values in the 'Size' column
international_df["Size"] = international_df["Size"].str.upper()
```

[164]:
```
international_df["Size"].value_counts()
```

[164]:
```
Size
L           14353
M            6424
S            5649
XL           3260
XXL          2688
XXXL         1657
XS            879
FREE          249
.             132
5XL           117
6XL           109
4XL           100
9              24
8              23
E              21
I              19
0              19
5              17
3              17
N              17
6              16
7              15
C              14
4              12
D              11
K              10
2              10
S TO XXL        9
1               9
)               2
G               1
PCS             1
Name: count, dtype: int64
```

[165]:
```
# Filter the DataFrame to keep rows with allowed 'Size' values
international_df = international_df[international_df["Size"].isin(["S", "M",␣
 ↪"L", "XL", "XXL", "XXXL", "4XL", "5XL", "6XL", "FREE"])]
```

[166]:
```
international_df["Size"].value_counts()
```

[166]:
```
Size
L       14353
M        6424
S        5649
```

```
XL       3260
XXL      2688
XXXL     1657
FREE      249
5XL       117
6XL       109
4XL       100
Name: count, dtype: int64
```

[167]:
```
# Converting column Date to date type

international_df["DATE"] = pd.to_datetime(international_df["DATE"],␣
 ↪errors="coerce")
```

[168]:
```
print(international_df["DATE"].max())

print(international_df["DATE"].min())
```

```
2022-05-11 00:00:00
2021-06-05 00:00:00
```

[169]:
```
# Checking null values

international_df.isnull().sum()
```

[169]:
```
DATE         1194
Months          0
CUSTOMER        0
Style           0
SKU          1315
Size            0
PCS             0
RATE            0
GROSS AMT       0
dtype: int64
```

[170]:
```
# Dispaly columns with null values and their respetive percenatge
i = 1
for (col,percentage) in (international_df.isnull().sum()/ len(international_df)␣
 ↪* 100).items():
    if percentage > 0:
        print(f"{i}. {col} has {percentage:.2f}% of missing values")
        i+=1
```

1. DATE has 3.45% of missing values
2. SKU has 3.80% of missing values

[171]:
```
international_df[international_df.isna()].head()
```

[171]:

	DATE	Months	CUSTOMER	Style	SKU	Size	PCS	RATE	GROSS AMT
0	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

[172]:
```
# Dropping the records having missing values

international_df = international_df.dropna()
```

[173]:
```
# Drop Months column

international_df.drop(columns=["Months"], inplace=True)
```

[174]:
```
international_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 32097 entries, 0 to 37430
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   DATE        32097 non-null  datetime64[ns]
 1   CUSTOMER    32097 non-null  object
 2   Style       32097 non-null  object
 3   SKU         32097 non-null  object
 4   Size        32097 non-null  object
 5   PCS         32097 non-null  object
 6   RATE        32097 non-null  object
 7   GROSS AMT   32097 non-null  object
dtypes: datetime64[ns](1), object(7)
memory usage: 2.2+ MB
```

[175]:
```
# Converting the column type to integer
columns_to_convert = ["PCS",
    "RATE",
    "GROSS AMT",
]

for col in columns_to_convert:
    international_df[col] = pd.to_numeric(international_df[col],␣
 ↪errors='coerce')
```

[176]:
```
# Output cleaned dataset
```

```
international_df.to_csv("International_cleaned_dataset.csv")
```

1.1.6 Feature Engineering

Note: Starting here we will only be using amazon_df dataframe as it is the primary dataset used for prediction of Apparel amount.

[53]:
```
amazon_df["Date"].head()
```

[53]:
```
0   2022-04-30
1   2022-04-30
2   2022-04-30
3   2022-04-30
4   2022-04-30
Name: Date, dtype: datetime64[ns]
```

[54]:
```
# Derive the day of the week and month and create new columns

amazon_df['Day_of_Week'] = pd.to_datetime(amazon_df['Date']).dt.dayofweek
amazon_df['Month'] = pd.to_datetime(amazon_df['Date']).dt.month
```

[55]:
```
amazon_df.head()
```

[55]:

	index	Order ID	Date	Status	\
0	0	405-8078784-5731545	2022-04-30	Cancelled	
1	1	171-9198151-1101146	2022-04-30	Shipped - Delivered to Buyer	
2	2	404-0687676-7273146	2022-04-30	Shipped	
3	3	403-9615377-8133951	2022-04-30	Cancelled	
4	4	407-1069790-7240320	2022-04-30	Shipped	

	Fulfilment	Sales Channel	ship-service-level	Style	SKU	\
0	Merchant	Amazon.in	Standard	SET389	SET389-KR-NP-S	
1	Merchant	Amazon.in	Standard	JNE3781	JNE3781-KR-XXXL	
2	Amazon	Amazon.in	Expedited	JNE3371	JNE3371-KR-XL	
3	Merchant	Amazon.in	Standard	J0341	J0341-DR-L	
4	Amazon	Amazon.in	Expedited	JNE3671	JNE3671-TU-XXXL	

	Category	Size	ASIN	Qty	Amount	ship-city	ship-state	\
0	Set	S	B09KXVBD7Z	0.0	647.62	MUMBAI	Maharashtra	
1	kurta	3XL	B09K3WFS32	1.0	406.00	BENGALURU	Karnataka	
2	kurta	XL	B07WV4JV4D	1.0	329.00	NAVI MUMBAI	Maharashtra	
3	Western Dress	L	B099NRCT7B	0.0	753.33	PUDUCHERRY	Pondicherry	
4	Top	3XL	B098714BZP	1.0	574.00	CHENNAI	Tamil Nadu	

	ship-postal-code	B2B	Day_of_Week	Month
0	400081	False	5	4
1	560085	False	5	4

```
2          410210   True            5      4
3          605008  False            5      4
4          600073  False            5      4
```

[56]:
```
# Dropping columns which are not relevant for model building

amazon_df.drop(columns=["index", "Order ID", "Date"], inplace=True)
```

[57]:
```
amazon_df.head()
```

[57]:
```
                       Status Fulfilment Sales Channel  ship-service-level  \
0                   Cancelled   Merchant      Amazon.in            Standard
1  Shipped - Delivered to Buyer   Merchant      Amazon.in            Standard
2                     Shipped     Amazon      Amazon.in           Expedited
3                   Cancelled   Merchant      Amazon.in            Standard
4                     Shipped     Amazon      Amazon.in           Expedited

     Style              SKU       Category Size        ASIN  Qty  Amount  \
0   SET389   SET389-KR-NP-S            Set    S  B09KXVBD7Z  0.0  647.62
1  JNE3781  JNE3781-KR-XXXL          kurta  3XL  B09K3WFS32  1.0  406.00
2  JNE3371    JNE3371-KR-XL          kurta   XL  B07WV4JV4D  1.0  329.00
3    J0341       J0341-DR-L  Western Dress    L  B099NRCT7B  0.0  753.33
4  JNE3671  JNE3671-TU-XXXL            Top  3XL  B098714BZP  1.0  574.00

     ship-city   ship-state ship-postal-code    B2B  Day_of_Week  Month
0       MUMBAI  Maharashtra           400081  False            5      4
1    BENGALURU    Karnataka           560085  False            5      4
2  NAVI MUMBAI  Maharashtra           410210   True            5      4
3   PUDUCHERRY  Pondicherry           605008  False            5      4
4      CHENNAI   Tamil Nadu           600073  False            5      4
```

1.1.7 Variable Transformation

[58]:
```
# Save categorical columns (object or category data types)
categorical_cols = amazon_df.select_dtypes(include=["object"]).columns.tolist()

# Save numeric columns (int and float data types)
numeric_cols = amazon_df.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()

print("Categorical Columns:", categorical_cols)
print("Numeric Columns:", numeric_cols)
```

```
Categorical Columns: ['Status', 'Fulfilment', 'Sales Channel ', 'ship-service-
level', 'Style', 'SKU', 'Category', 'Size', 'ASIN', 'ship-city', 'ship-state',
'ship-postal-code', 'B2B']
Numeric Columns: ['Qty', 'Amount']
```

As we can we see there are categoric columns and we need to convert it to numeric before feeding into the model.

Binary Mapping

[59]:
```
# Replace values in the 'Fulfilment' column

amazon_df["Fulfilment"].replace({"Amazon": 1, "Merchant": 0}, inplace=True)
```

[60]:
```
# Replace values in the 'Sales Channel' column

amazon_df["Sales Channel "].replace({'Amazon.in': 1, "Non-Amazon": 0},␣
 ↪inplace=True)
```

[61]:
```
# Replace values in the 'ship-service-level' column

amazon_df["ship-service-level"].replace({"Expedited": 1, "Standard": 0},␣
 ↪inplace=True)
```

[62]:
```
# Replace values in the 'B2B' column

amazon_df["B2B"].replace({True: 1, False: 0}, inplace=True)
```

One hot encoding

[63]:
```
# Creating dummy variables

Status_dummy = pd.get_dummies(amazon_df["Status"], drop_first=True)
Status_dummy = Status_dummy.astype(int)
Status_dummy.head()
```

[63]:

	Pending	Pending - Waiting for Pick Up	Shipped	Shipped - Damaged	Shipped - Delivered to Buyer	Shipped - Lost in Transit	Shipped - Out for Delivery	Shipped - Picked Up
0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0

```
2                              0                    0
3                              0                    0
4                              0                    0

   Shipped - Rejected by Buyer  Shipped - Returned to Seller  \
0                            0                             0
1                            0                             0
2                            0                             0
3                            0                             0
4                            0                             0

   Shipped - Returning to Seller  Shipping
0                              0         0
1                              0         0
2                              0         0
3                              0         0
4                              0         0
```

[64]:
```
# Creating dummy variable

Category_dummy = pd.get_dummies(amazon_df["Category"], drop_first= True)
Category_dummy = Category_dummy.astype(int)
```

[65]:
```
# Creating dummy variable

Size_dummy = pd.get_dummies(amazon_df["Size"], drop_first= True)
Size_dummy = Size_dummy.astype(int)
```

[66]:
```
# Concatenate the dummy datasets with the amazon_df dataset

amazon_transformed_df = pd.concat([amazon_df, Status_dummy, Category_dummy,␣
 ↪Size_dummy], axis = 1)
amazon_transformed_df.head()
```

[66]:
```
                        Status  Fulfilment  Sales Channel   \
0                    Cancelled           0               1
1  Shipped - Delivered to Buyer           0               1
2                      Shipped           1               1
3                    Cancelled           0               1
4                      Shipped           1               1

   ship-service-level    Style              SKU       Category Size  \
0                   0   SET389   SET389-KR-NP-S            Set    S
1                   0  JNE3781  JNE3781-KR-XXXL          kurta  3XL
2                   1  JNE3371    JNE3371-KR-XL          kurta   XL
3                   0    J0341       J0341-DR-L  Western Dress    L
4                   1  JNE3671  JNE3671-TU-XXXL            Top  3XL
```

```
         ASIN  Qty  Amount    ship-city   ship-state ship-postal-code  B2B  \
0  B09KXVBD7Z  0.0  647.62       MUMBAI  Maharashtra           400081    0
1  B09K3WFS32  1.0  406.00    BENGALURU    Karnataka           560085    0
2  B07WV4JV4D  1.0  329.00  NAVI MUMBAI  Maharashtra           410210    1
3  B099NRCT7B  0.0  753.33   PUDUCHERRY  Pondicherry           605008    0
4  B098714BZP  1.0  574.00      CHENNAI   Tamil Nadu           600073    0

   Day_of_Week  Month  Pending  Pending - Waiting for Pick Up  Shipped  \
0            5      4        0                              0        0
1            5      4        0                              0        0
2            5      4        0                              0        1
3            5      4        0                              0        0
4            5      4        0                              0        1

   Shipped - Damaged  Shipped - Delivered to Buyer  Shipped - Lost in Transit  \
0                  0                             0                          0
1                  0                             1                          0
2                  0                             0                          0
3                  0                             0                          0
4                  0                             0                          0

   Shipped - Out for Delivery  Shipped - Picked Up  \
0                           0                    0
1                           0                    0
2                           0                    0
3                           0                    0
4                           0                    0

   Shipped - Rejected by Buyer  Shipped - Returned to Seller  \
0                            0                             0
1                            0                             0
2                            0                             0
3                            0                             0
4                            0                             0

   Shipped - Returning to Seller  Shipping  Bottom  Dupatta  Ethnic Dress  \
0                              0         0       0        0             0
1                              0         0       0        0             0
2                              0         0       0        0             0
3                              0         0       0        0             0
4                              0         0       0        0             0

   Saree  Set  Top  Western Dress  kurta  4XL  5XL  6XL  Free  L  M  S  XL  \
0      0    1    0              0      0    0    0    0     0  0  0  1   0
1      0    0    0              0      1    0    0    0     0  0  0  0   0
2      0    0    0              0      1    0    0    0     0  0  0  0   1
```

```
3      0    0    0              1      0    0    0    0     0  1  0  0   0
4      0    0    1              0      0    0    0    0     0  0  0  0   0

   XS  XXL
0   0    0
1   0    0
2   0    0
3   0    0
4   0    0
```

[67]:
```
# Drop the categorical columns

amazon_transformed_df.drop(columns= ["Status", "Category", "Size"], axis = 1,␣
 ↪inplace = True)
amazon_transformed_df.head()
```

[67]:
```
   Fulfilment  Sales Channel   ship-service-level    Style              SKU  \
0           0               1                   0   SET389   SET389-KR-NP-S
1           0               1                   0  JNE3781  JNE3781-KR-XXXL
2           1               1                   1  JNE3371    JNE3371-KR-XL
3           0               1                   0    J0341       J0341-DR-L
4           1               1                   1  JNE3671  JNE3671-TU-XXXL

         ASIN  Qty  Amount    ship-city   ship-state ship-postal-code  B2B  \
0  B09KXVBD7Z  0.0  647.62       MUMBAI  Maharashtra           400081    0
1  B09K3WFS32  1.0  406.00    BENGALURU    Karnataka           560085    0
2  B07WV4JV4D  1.0  329.00  NAVI MUMBAI  Maharashtra           410210    1
3  B099NRCT7B  0.0  753.33   PUDUCHERRY  Pondicherry           605008    0
4  B098714BZP  1.0  574.00      CHENNAI   Tamil Nadu           600073    0

   Day_of_Week  Month  Pending  Pending - Waiting for Pick Up  Shipped  \
0            5      4        0                              0        0
1            5      4        0                              0        0
2            5      4        0                              0        1
3            5      4        0                              0        0
4            5      4        0                              0        1

   Shipped - Damaged  Shipped - Delivered to Buyer  Shipped - Lost in Transit  \
0                  0                             0                          0
1                  0                             1                          0
2                  0                             0                          0
3                  0                             0                          0
4                  0                             0                          0

   Shipped - Out for Delivery  Shipped - Picked Up  \
0                           0                    0
1                           0                    0
```

```
2                             0                    0
3                             0                    0
4                             0                    0

   Shipped - Rejected by Buyer  Shipped - Returned to Seller  \
0                            0                             0
1                            0                             0
2                            0                             0
3                            0                             0
4                            0                             0

   Shipped - Returning to Seller  Shipping  Bottom  Dupatta  Ethnic Dress  \
0                              0         0       0        0             0
1                              0         0       0        0             0
2                              0         0       0        0             0
3                              0         0       0        0             0
4                              0         0       0        0             0

   Saree  Set  Top  Western Dress  kurta  4XL  5XL  6XL  Free  L  M  S  XL  \
0      0    1    0              0      0    0    0    0     0  0  0  1   0
1      0    0    0              0      1    0    0    0     0  0  0  0   0
2      0    0    0              0      1    0    0    0     0  0  0  0   1
3      0    0    0              1      0    0    0    0     0  1  0  0   0
4      0    0    1              0      0    0    0    0     0  0  0  0   0

   XS  XXL
0   0    0
1   0    0
2   0    0
3   0    0
4   0    0
```

Label Encoding

[68]:
```
# Label encoding the variables

# Create a list of the column names you want to label encode
columns_to_encode = ["Style", "SKU", "ASIN", "ship-city", "ship-state"]

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Use apply to label encode the specified columns
amazon_transformed_df[columns_to_encode] =␣
 ↪amazon_transformed_df[columns_to_encode].apply(label_encoder.fit_transform)
```

[69]:
```
amazon_transformed_df.head()
```

[69]:

	Fulfilment	Sales Channel	ship-service-level	Style	SKU	ASIN	Qty
0	0	1	0	1342	6992	5563	0.0
1	0	1	0	847	4436	5277	1.0
2	1	1	1	533	2633	675	1.0
3	0	1	0	373	1767	4638	0.0
4	1	1	1	754	3837	4351	1.0

	Amount	ship-city	ship-state	ship-postal-code	B2B	Day_of_Week	Month
0	647.62	4184	20	400081	0	5	4
1	406.00	774	15	560085	0	5	4
2	329.00	4483	20	410210	1	5	4
3	753.33	5241	27	605008	0	5	4
4	574.00	1251	31	600073	0	5	4

	Pending	Pending - Waiting for Pick Up	Shipped	Shipped - Damaged
0	0	0	0	0
1	0	0	0	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0

	Shipped - Delivered to Buyer	Shipped - Lost in Transit
0	0	0
1	1	0
2	0	0
3	0	0
4	0	0

	Shipped - Out for Delivery	Shipped - Picked Up
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0

	Shipped - Rejected by Buyer	Shipped - Returned to Seller
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0

	Shipped - Returning to Seller	Shipping	Bottom	Dupatta	Ethnic Dress
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

```
4                                0         0       0        0              0

   Saree  Set  Top  Western Dress  kurta  4XL  5XL  6XL  Free  L  M  S  XL  \
0      0    1    0              0      0    0    0    0     0  0  0  1   0
1      0    0    0              0      1    0    0    0     0  0  0  0   0
2      0    0    0              0      1    0    0    0     0  0  0  0   1
3      0    0    0              1      0    0    0    0     0  1  0  0   0
4      0    0    1              0      0    0    0    0     0  0  0  0   0

   XS  XXL
0   0    0
1   0    0
2   0    0
3   0    0
4   0    0
```

1.1.8 Multicollinearity

Multicollinearity is a phenomenon in multiple regression analysis where two or more independent variables in a regression model are highly correlated, making it difficult to separate their individual effects on the dependent variable.

[70]:
```
amazon_transformed_df.corr()
```

[70]:
```
                                Fulfilment  Sales Channel   ship-service-level  \
Fulfilment                        1.000000       -0.020530            0.980513
Sales Channel                    -0.020530        1.000000            0.045975
ship-service-level                0.980513        0.045975            1.000000
Style                            -0.027330        0.012575           -0.026986
SKU                              -0.031690        0.013025           -0.031260
ASIN                             -0.090293        0.030211           -0.087107
Qty                               0.167431       -0.010300            0.162566
Amount                            0.003742        0.004599            0.038761
ship-city                        -0.013660       -0.047311           -0.014815
ship-state                        0.006070        0.032346            0.007776
ship-postal-code                  0.001803        0.011303            0.003833
B2B                              -0.005096        0.002559           -0.003564
Day_of_Week                       0.012801       -0.010435            0.011549
Month                             0.046237        0.003590            0.045352
Pending                          -0.010083        0.002222           -0.009179
Pending - Waiting for Pick Up    -0.070628        0.001450           -0.069251
Shipped                           0.815976       -0.020557            0.797149
Shipped - Damaged                -0.004209        0.000086           -0.004127
Shipped - Delivered to Buyer     -0.809775        0.016624           -0.793995
Shipped - Lost in Transit        -0.009411        0.000193           -0.009228
Shipped - Out for Delivery       -0.024902        0.000511           -0.024417
Shipped - Picked Up              -0.131780        0.002705           -0.129212
```

Shipped - Rejected by Buyer	-0.013959	0.000287	-0.013687
Shipped - Returned to Seller	-0.187273	0.003845	-0.183623
Shipped - Returning to Seller	-0.050708	0.001041	-0.049720
Shipping	0.005212	-0.253886	-0.011672
Bottom	-0.016765	-0.002475	-0.017014
Dupatta	0.003192	0.000150	0.003255
Ethnic Dress	0.007487	0.002955	0.006852
Saree	0.008483	0.001107	0.008596
Set	0.019975	-0.005979	0.018187
Top	0.041924	0.005655	0.043082
Western Dress	-0.116516	0.008390	-0.112906
kurta	0.032378	-0.001568	0.031406
4XL	0.007179	0.001784	0.007583
5XL	0.012019	-0.001809	0.012345
6XL	0.007088	-0.000963	0.006857
Free	0.008137	-0.016830	0.007512
L	-0.014764	0.000846	-0.015815
M	-0.014440	0.001205	-0.014653
S	0.014812	-0.000422	0.014182
XL	-0.014743	-0.000630	-0.015164
XS	0.038859	-0.002018	0.039860
XXL	0.007061	0.002449	0.007488

	Style	SKU	ASIN	Qty	\
Fulfilment	-0.027330	-0.031690	-0.090293	0.167431	
Sales Channel	0.012575	0.013025	0.030211	-0.010300	
ship-service-level	-0.026986	-0.031260	-0.087107	0.162566	
Style	1.000000	0.998830	0.469413	-0.007270	
SKU	0.998830	1.000000	0.486358	-0.007133	
ASIN	0.469413	0.486358	1.000000	0.002649	
Qty	-0.007270	-0.007133	0.002649	1.000000	
Amount	0.121496	0.114020	0.268393	0.041719	
ship-city	0.005577	0.005404	-0.003083	-0.014561	
ship-state	-0.002204	-0.001409	-0.016568	-0.006315	
ship-postal-code	-0.018779	-0.017977	-0.025527	-0.011348	
B2B	-0.007788	-0.007641	-0.005577	0.012814	
Day_of_Week	0.004024	0.004267	0.006880	0.007538	
Month	0.023462	0.029279	0.146594	0.012513	
Pending	0.006618	0.007306	0.016669	0.023049	
Pending - Waiting for Pick Up	0.007406	0.008036	0.021991	0.015515	
Shipped	-0.023834	-0.027052	-0.066031	0.404473	
Shipped - Damaged	-0.002579	-0.002548	0.003798	0.000925	
Shipped - Delivered to Buyer	0.020638	0.024435	0.074277	0.177392	
Shipped - Lost in Transit	0.001416	0.001368	0.000808	0.002067	
Shipped - Out for Delivery	0.001240	0.001389	0.006460	0.005471	
Shipped - Picked Up	0.007017	0.007885	0.024457	0.028949	
Shipped - Rejected by Buyer	0.001317	0.001133	0.002412	0.003067	

Shipped - Returned to Seller	0.004983	0.005590	0.010966	0.040502
Shipped - Returning to Seller	0.009798	0.010147	0.014391	0.011139
Shipping	-0.010860	-0.010894	-0.007996	0.002615
Bottom	-0.106585	-0.105209	-0.048882	-0.001031
Dupatta	-0.008307	-0.008377	-0.007841	0.001601
Ethnic Dress	-0.131415	-0.133181	-0.051816	0.001666
Saree	0.019656	0.020432	0.031471	0.000205
Set	0.458682	0.435804	0.115195	-0.007508
Top	-0.234096	-0.236535	0.008431	0.028521
Western Dress	-0.139670	-0.128253	0.309161	-0.005048
kurta	-0.174229	-0.157477	-0.311446	-0.005810
4XL	0.037629	0.044064	0.056670	0.005065
5XL	0.042879	0.050212	0.064762	0.006607
6XL	0.049590	0.058179	0.074865	0.007655
Free	-0.046136	-0.043762	-0.033250	0.000253
L	-0.002917	-0.003394	-0.009791	-0.000803
M	0.013913	0.012206	-0.020725	-0.006196
S	-0.003048	-0.005272	-0.006567	-0.008724
XL	-0.015687	-0.015658	-0.029603	0.004023
XS	0.013746	0.011231	0.031475	-0.014025
XXL	-0.020517	-0.019459	-0.011025	0.009800

	Amount	ship-city	ship-state	\
Fulfilment	0.003742	-0.013660	0.006070	
Sales Channel	0.004599	-0.047311	0.032346	
ship-service-level	0.038761	-0.014815	0.007776	
Style	0.121496	0.005577	-0.002204	
SKU	0.114020	0.005404	-0.001409	
ASIN	0.268393	-0.003083	-0.016568	
Qty	0.041719	-0.014561	-0.006315	
Amount	1.000000	-0.000526	-0.007053	
ship-city	-0.000526	1.000000	-0.112176	
ship-state	-0.007053	-0.112176	1.000000	
ship-postal-code	-0.037711	-0.137136	0.032550	
B2B	0.013776	0.014413	0.009240	
Day_of_Week	0.003596	0.005331	-0.004858	
Month	0.053404	-0.010881	-0.001490	
Pending	0.002725	0.003278	0.002716	
Pending - Waiting for Pick Up	0.006275	-0.006109	-0.008115	
Shipped	0.011616	-0.016865	0.005623	
Shipped - Damaged	0.005119	-0.002808	0.002824	
Shipped - Delivered to Buyer	0.004341	0.007868	-0.007731	
Shipped - Lost in Transit	-0.005713	-0.007731	-0.001735	
Shipped - Out for Delivery	0.007765	0.000348	0.002464	
Shipped - Picked Up	0.011091	0.003619	-0.003066	
Shipped - Rejected by Buyer	0.000641	0.004939	-0.002064	
Shipped - Returned to Seller	0.001877	0.002302	0.003627	

Shipped - Returning to Seller	0.011526	-0.002348	-0.001620
Shipping	-0.001168	0.012012	-0.008212
Bottom	-0.060131	-0.003009	0.001359
Dupatta	-0.006128	0.000362	0.002687
Ethnic Dress	0.025821	-0.003583	0.001552
Saree	0.018492	-0.005280	0.005035
Set	0.511363	0.010031	-0.016681
Top	-0.129139	-0.004023	-0.012933
Western Dress	0.151311	-0.003266	-0.007725
kurta	-0.532737	-0.001715	0.028203
4XL	0.035301	0.000598	0.009184
5XL	0.037738	0.000198	-0.001487
6XL	0.045926	-0.001001	0.007554
Free	-0.014011	-0.005738	0.008209
L	-0.018899	-0.003764	-0.002714
M	0.006899	-0.004536	-0.004627
S	0.023645	0.004267	-0.012712
XL	-0.025101	-0.003569	0.008310
XS	0.033704	0.006096	-0.011632
XXL	-0.035445	0.006101	0.012204

	ship-postal-code	B2B	Day_of_Week	\
Fulfilment	0.001803	-0.005096	0.012801	
Sales Channel	0.011303	0.002559	-0.010435	
ship-service-level	0.003833	-0.003564	0.011549	
Style	-0.018779	-0.007788	0.004024	
SKU	-0.017977	-0.007641	0.004267	
ASIN	-0.025527	-0.005577	0.006880	
Qty	-0.011348	0.012814	0.007538	
Amount	-0.037711	0.013776	0.003596	
ship-city	-0.137136	0.014413	0.005331	
ship-state	0.032550	0.009240	-0.004858	
ship-postal-code	1.000000	-0.035647	-0.013804	
B2B	-0.035647	1.000000	-0.001187	
Day_of_Week	-0.013804	-0.001187	1.000000	
Month	0.005221	-0.003462	-0.043181	
Pending	0.005724	-0.000591	-0.038825	
Pending - Waiting for Pick Up	0.006072	-0.003854	-0.048779	
Shipped	-0.006713	0.001072	0.016501	
Shipped - Damaged	0.001979	-0.000230	0.004106	
Shipped - Delivered to Buyer	-0.012284	0.012443	-0.005973	
Shipped - Lost in Transit	0.001715	-0.000514	-0.001880	
Shipped - Out for Delivery	0.004419	-0.001359	-0.000562	
Shipped - Picked Up	0.014466	-0.001721	0.019784	
Shipped - Rejected by Buyer	-0.002525	-0.000762	-0.004612	
Shipped - Returned to Seller	0.004533	-0.003237	-0.005309	
Shipped - Returning to Seller	0.010400	-0.002767	-0.000997	

Shipping	-0.002870	-0.000650	-0.003933
Bottom	0.005535	0.001669	-0.000279
Dupatta	0.001468	-0.000398	0.007112
Ethnic Dress	-0.004596	0.007194	-0.006401
Saree	0.003152	0.002370	0.006662
Set	-0.064910	0.000081	0.002277
Top	-0.054929	-0.003698	0.004567
Western Dress	0.029172	0.002418	0.000621
kurta	0.075743	-0.001711	-0.004423
4XL	0.002484	0.006821	-0.014294
5XL	0.002915	-0.003945	0.004605
6XL	0.005688	-0.002492	-0.003238
Free	0.000176	0.011290	0.007254
L	0.003649	-0.000111	0.002930
M	0.006190	-0.003323	-0.003299
S	0.012674	0.004079	0.000608
XL	-0.006067	-0.003339	-0.002436
XS	0.016307	-0.002824	0.002159
XXL	-0.009545	-0.000877	0.000714

	Month	Pending	\
Fulfilment	0.046237	-0.010083	
Sales Channel	0.003590	0.002222	
ship-service-level	0.045352	-0.009179	
Style	0.023462	0.006618	
SKU	0.029279	0.007306	
ASIN	0.146594	0.016669	
Qty	0.012513	0.023049	
Amount	0.053404	0.002725	
ship-city	-0.010881	0.003278	
ship-state	-0.001490	0.002716	
ship-postal-code	0.005221	0.005724	
B2B	-0.003462	-0.000591	
Day_of_Week	-0.043181	-0.038825	
Month	1.000000	0.094125	
Pending	0.094125	1.000000	
Pending - Waiting for Pick Up	0.062288	-0.003347	
Shipped	0.030064	-0.088318	
Shipped - Damaged	0.003712	-0.000199	
Shipped - Delivered to Buyer	-0.073521	-0.038375	
Shipped - Lost in Transit	-0.002353	-0.000446	
Shipped - Out for Delivery	0.020236	-0.001180	
Shipped - Picked Up	0.114795	-0.006245	
Shipped - Rejected by Buyer	-0.000001	-0.000662	
Shipped - Returned to Seller	-0.028339	-0.008875	
Shipped - Returning to Seller	0.039631	-0.002403	
Shipping	0.010499	-0.000564	

Bottom	-0.004884	-0.002324
Dupatta	0.006429	-0.000345
Ethnic Dress	0.010866	0.001252
Saree	-0.006411	-0.002556
Set	-0.034956	-0.001013
Top	-0.012094	0.000319
Western Dress	0.072620	0.004320
kurta	-0.004816	-0.001438
4XL	0.016795	0.001578
5XL	0.018020	0.000323
6XL	0.023742	0.001780
Free	-0.016771	-0.001870
L	-0.004312	0.002628
M	0.005155	0.002612
S	-0.012777	-0.003909
XL	0.003933	0.003099
XS	-0.003735	-0.001142
XXL	0.009484	-0.001044

	Pending - Waiting for Pick Up	Shipped	\
Fulfilment	-0.070628	0.815976	
Sales Channel	0.001450	-0.020557	
ship-service-level	-0.069251	0.797149	
Style	0.007406	-0.023834	
SKU	0.008036	-0.027052	
ASIN	0.021991	-0.066031	
Qty	0.015515	0.404473	
Amount	0.006275	0.011616	
ship-city	-0.006109	-0.016865	
ship-state	-0.008115	0.005623	
ship-postal-code	0.006072	-0.006713	
B2B	-0.003854	0.001072	
Day_of_Week	-0.048779	0.016501	
Month	0.062288	0.030064	
Pending	-0.003347	-0.088318	
Pending - Waiting for Pick Up	1.000000	-0.057630	
Shipped	-0.057630	1.000000	
Shipped - Damaged	-0.000130	-0.003434	
Shipped - Delivered to Buyer	-0.025041	-0.660757	
Shipped - Lost in Transit	-0.000291	-0.007679	
Shipped - Out for Delivery	-0.000770	-0.020320	
Shipped - Picked Up	-0.004075	-0.107529	
Shipped - Rejected by Buyer	-0.000432	-0.011390	
Shipped - Returned to Seller	-0.005791	-0.152810	
Shipped - Returning to Seller	-0.001568	-0.041376	
Shipping	-0.000368	-0.009714	
Bottom	0.002969	-0.011810	

Dupatta	-0.000225	0.003912
Ethnic Dress	-0.004451	0.009372
Saree	-0.001668	0.008923
Set	-0.000530	0.010802
Top	-0.002507	0.042333
Western Dress	0.023128	-0.091174
kurta	-0.012519	0.022978
4XL	-0.002687	0.007635
5XL	-0.000507	0.013669
6XL	0.000864	0.009830
Free	-0.002534	0.009078
L	-0.003182	-0.012339
M	0.000664	-0.016521
S	-0.004037	0.008328
XL	0.002490	-0.006617
XS	0.000402	0.018738
XXL	0.004583	0.008480

	Shipped - Damaged	\
Fulfilment	-0.004209	
Sales Channel	0.000086	
ship-service-level	-0.004127	
Style	-0.002579	
SKU	-0.002548	
ASIN	0.003798	
Qty	0.000925	
Amount	0.005119	
ship-city	-0.002808	
ship-state	0.002824	
ship-postal-code	0.001979	
B2B	-0.000230	
Day_of_Week	0.004106	
Month	0.003712	
Pending	-0.000199	
Pending - Waiting for Pick Up	-0.000130	
Shipped	-0.003434	
Shipped - Damaged	1.000000	
Shipped - Delivered to Buyer	-0.001492	
Shipped - Lost in Transit	-0.000017	
Shipped - Out for Delivery	-0.000046	
Shipped - Picked Up	-0.000243	
Shipped - Rejected by Buyer	-0.000026	
Shipped - Returned to Seller	-0.000345	
Shipped - Returning to Seller	-0.000093	
Shipping	-0.000022	
Bottom	-0.000163	
Dupatta	-0.000013	

Ethnic Dress	-0.000265
Saree	-0.000099
Set	-0.002226
Top	-0.000834
Western Dress	0.007534
kurta	-0.002211
4XL	-0.000160
5XL	-0.000182
6XL	-0.000211
Free	-0.000151
L	-0.001267
M	-0.001287
S	-0.001088
XL	-0.001224
XS	-0.000857
XXL	-0.001125

	Shipped - Delivered to Buyer \
Fulfilment	-0.809775
Sales Channel	0.016624
ship-service-level	-0.793995
Style	0.020638
SKU	0.024435
ASIN	0.074277
Qty	0.177392
Amount	0.004341
ship-city	0.007868
ship-state	-0.007731
ship-postal-code	-0.012284
B2B	0.012443
Day_of_Week	-0.005973
Month	-0.073521
Pending	-0.038375
Pending - Waiting for Pick Up	-0.025041
Shipped	-0.660757
Shipped - Damaged	-0.001492
Shipped - Delivered to Buyer	1.000000
Shipped - Lost in Transit	-0.003337
Shipped - Out for Delivery	-0.008829
Shipped - Picked Up	-0.046723
Shipped - Rejected by Buyer	-0.004949
Shipped - Returned to Seller	-0.066398
Shipped - Returning to Seller	-0.017979
Shipping	-0.004221
Bottom	0.013051
Dupatta	-0.002585
Ethnic Dress	-0.004842

Saree	-0.007622
Set	-0.021910
Top	-0.030421
Western Dress	0.099206
kurta	-0.025734
4XL	-0.005786
5XL	-0.009629
6XL	-0.001635
Free	-0.007346
L	0.014289
M	0.008979
S	-0.017232
XL	0.012150
XS	-0.037791
XXL	-0.001388

	Shipped - Lost in Transit \
Fulfilment	-0.009411
Sales Channel	0.000193
ship-service-level	-0.009228
Style	0.001416
SKU	0.001368
ASIN	0.000808
Qty	0.002067
Amount	-0.005713
ship-city	-0.007731
ship-state	-0.001735
ship-postal-code	0.001715
B2B	-0.000514
Day_of_Week	-0.001880
Month	-0.002353
Pending	-0.000446
Pending - Waiting for Pick Up	-0.000291
Shipped	-0.007679
Shipped - Damaged	-0.000017
Shipped - Delivered to Buyer	-0.003337
Shipped - Lost in Transit	1.000000
Shipped - Out for Delivery	-0.000103
Shipped - Picked Up	-0.000543
Shipped - Rejected by Buyer	-0.000058
Shipped - Returned to Seller	-0.000772
Shipped - Returning to Seller	-0.000209
Shipping	-0.000049
Bottom	-0.000364
Dupatta	-0.000030
Ethnic Dress	-0.000593
Saree	-0.000222

Set	0.000129
Top	-0.001866
Western Dress	0.001528
kurta	0.000170
4XL	-0.000358
5XL	-0.000408
6XL	-0.000472
Free	-0.000338
L	-0.002834
M	-0.002879
S	0.001240
XL	0.007407
XS	-0.001917
XXL	0.001070

	Shipped - Out for Delivery \
Fulfilment	-0.024902
Sales Channel	0.000511
ship-service-level	-0.024417
Style	0.001240
SKU	0.001389
ASIN	0.006460
Qty	0.005471
Amount	0.007765
ship-city	0.000348
ship-state	0.002464
ship-postal-code	0.004419
B2B	-0.001359
Day_of_Week	-0.000562
Month	0.020236
Pending	-0.001180
Pending - Waiting for Pick Up	-0.000770
Shipped	-0.020320
Shipped - Damaged	-0.000046
Shipped - Delivered to Buyer	-0.008829
Shipped - Lost in Transit	-0.000103
Shipped - Out for Delivery	1.000000
Shipped - Picked Up	-0.001437
Shipped - Rejected by Buyer	-0.000152
Shipped - Returned to Seller	-0.002042
Shipped - Returning to Seller	-0.000553
Shipping	-0.000130
Bottom	-0.000964
Dupatta	-0.000079
Ethnic Dress	-0.001569
Saree	-0.000588
Set	0.005168

Top	0.003626
Western Dress	0.002596
kurta	-0.008250
4XL	-0.000948
5XL	-0.001078
6XL	-0.001250
Free	-0.000893
L	0.001242
M	0.001035
S	0.004670
XL	-0.003407
XS	-0.001723
XXL	0.002831

	Shipped - Picked Up \
Fulfilment	-0.131780
Sales Channel	0.002705
ship-service-level	-0.129212
Style	0.007017
SKU	0.007885
ASIN	0.024457
Qty	0.028949
Amount	0.011091
ship-city	0.003619
ship-state	-0.003066
ship-postal-code	0.014466
B2B	-0.001721
Day_of_Week	0.019784
Month	0.114795
Pending	-0.006245
Pending - Waiting for Pick Up	-0.004075
Shipped	-0.107529
Shipped - Damaged	-0.000243
Shipped - Delivered to Buyer	-0.046723
Shipped - Lost in Transit	-0.000543
Shipped - Out for Delivery	-0.001437
Shipped - Picked Up	1.000000
Shipped - Rejected by Buyer	-0.000805
Shipped - Returned to Seller	-0.010805
Shipped - Returning to Seller	-0.002926
Shipping	-0.000687
Bottom	0.010265
Dupatta	-0.000421
Ethnic Dress	-0.006405
Saree	-0.000597
Set	0.004897
Top	-0.003631

Western Dress	0.019027
kurta	-0.014582
4XL	0.009057
5XL	-0.001582
6XL	-0.000676
Free	-0.001413
L	0.001440
M	0.003923
S	-0.004999
XL	0.004014
XS	-0.007718
XXL	0.001153

	Shipped - Rejected by Buyer \
Fulfilment	-0.013959
Sales Channel	0.000287
ship-service-level	-0.013687
Style	0.001317
SKU	0.001133
ASIN	0.002412
Qty	0.003067
Amount	0.000641
ship-city	0.004939
ship-state	-0.002064
ship-postal-code	-0.002525
B2B	-0.000762
Day_of_Week	-0.004612
Month	-0.000001
Pending	-0.000662
Pending - Waiting for Pick Up	-0.000432
Shipped	-0.011390
Shipped - Damaged	-0.000026
Shipped - Delivered to Buyer	-0.004949
Shipped - Lost in Transit	-0.000058
Shipped - Out for Delivery	-0.000152
Shipped - Picked Up	-0.000805
Shipped - Rejected by Buyer	1.000000
Shipped - Returned to Seller	-0.001145
Shipped - Returning to Seller	-0.000310
Shipping	-0.000073
Bottom	-0.000540
Dupatta	-0.000045
Ethnic Dress	-0.000880
Saree	-0.000330
Set	0.002946
Top	0.003342
Western Dress	-0.000832

kurta	-0.003886
4XL	-0.000531
5XL	-0.000605
6XL	-0.000701
Free	-0.000501
L	-0.004204
M	0.004548
S	0.001344
XL	-0.001780
XS	0.003130
XXL	-0.001314

	Shipped - Returned to Seller \
Fulfilment	-0.187273
Sales Channel	0.003845
ship-service-level	-0.183623
Style	0.004983
SKU	0.005590
ASIN	0.010966
Qty	0.040502
Amount	0.001877
ship-city	0.002302
ship-state	0.003627
ship-postal-code	0.004533
B2B	-0.003237
Day_of_Week	-0.005309
Month	-0.028339
Pending	-0.008875
Pending - Waiting for Pick Up	-0.005791
Shipped	-0.152810
Shipped - Damaged	-0.000345
Shipped - Delivered to Buyer	-0.066398
Shipped - Lost in Transit	-0.000772
Shipped - Out for Delivery	-0.002042
Shipped - Picked Up	-0.010805
Shipped - Rejected by Buyer	-0.001145
Shipped - Returned to Seller	1.000000
Shipped - Returning to Seller	-0.004158
Shipping	-0.000976
Bottom	-0.001803
Dupatta	-0.000598
Ethnic Dress	-0.001029
Saree	-0.002639
Set	0.000488
Top	-0.008462
Western Dress	0.015556
kurta	-0.005223

4XL	-0.002691
5XL	0.001640
6XL	-0.006032
Free	-0.002018
L	-0.000265
M	0.005613
S	0.001996
XL	-0.000802
XS	-0.001983
XXL	-0.003042

	Shipped - Returning to Seller	Shipping	\
Fulfilment	-0.050708	0.005212	
Sales Channel	0.001041	-0.253886	
ship-service-level	-0.049720	-0.011672	
Style	0.009798	-0.010860	
SKU	0.010147	-0.010894	
ASIN	0.014391	-0.007996	
Qty	0.011139	0.002615	
Amount	0.011526	-0.001168	
ship-city	-0.002348	0.012012	
ship-state	-0.001620	-0.008212	
ship-postal-code	0.010400	-0.002870	
B2B	-0.002767	-0.000650	
Day_of_Week	-0.000997	-0.003933	
Month	0.039631	0.010499	
Pending	-0.002403	-0.000564	
Pending - Waiting for Pick Up	-0.001568	-0.000368	
Shipped	-0.041376	-0.009714	
Shipped - Damaged	-0.000093	-0.000022	
Shipped - Delivered to Buyer	-0.017979	-0.004221	
Shipped - Lost in Transit	-0.000209	-0.000049	
Shipped - Out for Delivery	-0.000553	-0.000130	
Shipped - Picked Up	-0.002926	-0.000687	
Shipped - Rejected by Buyer	-0.000310	-0.000073	
Shipped - Returned to Seller	-0.004158	-0.000976	
Shipped - Returning to Seller	1.000000	-0.000264	
Shipping	-0.000264	1.000000	
Bottom	0.002005	-0.000461	
Dupatta	-0.000162	-0.000038	
Ethnic Dress	-0.003196	-0.000750	
Saree	-0.001197	-0.000281	
Set	0.007813	0.005816	
Top	-0.003319	0.001222	
Western Dress	0.007522	-0.002912	
kurta	-0.010011	-0.004233	
4XL	-0.001930	-0.000453	

5XL	0.001355	-0.000516
6XL	-0.002546	-0.000598
Free	-0.001819	-0.000427
L	-0.001153	0.004250
M	0.002715	-0.001056
S	-0.000144	-0.003078
XL	0.002217	0.001885
XS	-0.003744	0.001077
XXL	0.000435	-0.003182

	Bottom	Dupatta	Ethnic Dress	Saree	\
Fulfilment	-0.016765	0.003192	0.007487	0.008483	
Sales Channel	-0.002475	0.000150	0.002955	0.001107	
ship-service-level	-0.017014	0.003255	0.006852	0.008596	
Style	-0.106585	-0.008307	-0.131415	0.019656	
SKU	-0.105209	-0.008377	-0.133181	0.020432	
ASIN	-0.048882	-0.007841	-0.051816	0.031471	
Qty	-0.001031	0.001601	0.001666	0.000205	
Amount	-0.060131	-0.006128	0.025821	0.018492	
ship-city	-0.003009	0.000362	-0.003583	-0.005280	
ship-state	0.001359	0.002687	0.001552	0.005035	
ship-postal-code	0.005535	0.001468	-0.004596	0.003152	
B2B	0.001669	-0.000398	0.007194	0.002370	
Day_of_Week	-0.000279	0.007112	-0.006401	0.006662	
Month	-0.004884	0.006429	0.010866	-0.006411	
Pending	-0.002324	-0.000345	0.001252	-0.002556	
Pending - Waiting for Pick Up	0.002969	-0.000225	-0.004451	-0.001668	
Shipped	-0.011810	0.003912	0.009372	0.008923	
Shipped - Damaged	-0.000163	-0.000013	-0.000265	-0.000099	
Shipped - Delivered to Buyer	0.013051	-0.002585	-0.004842	-0.007622	
Shipped - Lost in Transit	-0.000364	-0.000030	-0.000593	-0.000222	
Shipped - Out for Delivery	-0.000964	-0.000079	-0.001569	-0.000588	
Shipped - Picked Up	0.010265	-0.000421	-0.006405	-0.000597	
Shipped - Rejected by Buyer	-0.000540	-0.000045	-0.000880	-0.000330	
Shipped - Returned to Seller	-0.001803	-0.000598	-0.001029	-0.002639	
Shipped - Returning to Seller	0.002005	-0.000162	-0.003196	-0.001197	
Shipping	-0.000461	-0.000038	-0.000750	-0.000281	
Bottom	1.000000	-0.000282	-0.005573	-0.002088	
Dupatta	-0.000282	1.000000	-0.000459	-0.000172	
Ethnic Dress	-0.005573	-0.000459	1.000000	-0.003399	
Saree	-0.002088	-0.000172	-0.003399	1.000000	
Set	-0.046776	-0.003856	-0.076131	-0.028527	
Top	-0.017531	-0.001445	-0.028532	-0.010691	
Western Dress	-0.021629	-0.001783	-0.035202	-0.013191	
kurta	-0.046463	-0.003830	-0.075621	-0.028336	
4XL	-0.003365	-0.000277	-0.005477	-0.002052	
5XL	-0.003830	-0.000316	-0.006233	-0.002336	

6XL	-0.004440	-0.000366	-0.007226	-0.002708
Free	-0.003173	0.088957	-0.005164	0.658135
L	-0.001585	-0.002195	-0.001494	-0.016241
M	-0.006800	-0.002230	-0.005194	-0.016497
S	0.006161	-0.001885	0.005441	-0.013946
XL	-0.000081	-0.002120	0.001426	-0.015683
XS	-0.004769	-0.001485	-0.003598	-0.010985
XXL	0.004694	-0.001949	-0.001332	-0.014418

	Set	Top	Western Dress	kurta	\
Fulfilment	0.019975	0.041924	-0.116516	0.032378	
Sales Channel	-0.005979	0.005655	0.008390	-0.001568	
ship-service-level	0.018187	0.043082	-0.112906	0.031406	
Style	0.458682	-0.234096	-0.139670	-0.174229	
SKU	0.435804	-0.236535	-0.128253	-0.157477	
ASIN	0.115195	0.008431	0.309161	-0.311446	
Qty	-0.007508	0.028521	-0.005048	-0.005810	
Amount	0.511363	-0.129139	0.151311	-0.532737	
ship-city	0.010031	-0.004023	-0.003266	-0.001715	
ship-state	-0.016681	-0.012933	-0.007725	0.028203	
ship-postal-code	-0.064910	-0.054929	0.029172	0.075743	
B2B	0.000081	-0.003698	0.002418	-0.001711	
Day_of_Week	0.002277	0.004567	0.000621	-0.004423	
Month	-0.034956	-0.012094	0.072620	-0.004816	
Pending	-0.001013	0.000319	0.004320	-0.001438	
Pending - Waiting for Pick Up	-0.000530	-0.002507	0.023128	-0.012519	
Shipped	0.010802	0.042333	-0.091174	0.022978	
Shipped - Damaged	-0.002226	-0.000834	0.007534	-0.002211	
Shipped - Delivered to Buyer	-0.021910	-0.030421	0.099206	-0.025734	
Shipped - Lost in Transit	0.000129	-0.001866	0.001528	0.000170	
Shipped - Out for Delivery	0.005168	0.003626	0.002596	-0.008250	
Shipped - Picked Up	0.004897	-0.003631	0.019027	-0.014582	
Shipped - Rejected by Buyer	0.002946	0.003342	-0.000832	-0.003886	
Shipped - Returned to Seller	0.000488	-0.008462	0.015556	-0.005223	
Shipped - Returning to Seller	0.007813	-0.003319	0.007522	-0.010011	
Shipping	0.005816	0.001222	-0.002912	-0.004233	
Bottom	-0.046776	-0.017531	-0.021629	-0.046463	
Dupatta	-0.003856	-0.001445	-0.001783	-0.003830	
Ethnic Dress	-0.076131	-0.028532	-0.035202	-0.075621	
Saree	-0.028527	-0.010691	-0.013191	-0.028336	
Set	1.000000	-0.239489	-0.295474	-0.634727	
Top	-0.239489	1.000000	-0.110738	-0.237883	
Western Dress	-0.295474	-0.110738	1.000000	-0.293493	
kurta	-0.634727	-0.237883	-0.293493	1.000000	
4XL	-0.032653	-0.017228	-0.021256	0.059087	
5XL	-0.037436	-0.019609	-0.024192	0.067523	
6XL	-0.043576	-0.022731	-0.028044	0.078454	

```
Free                          -0.043345 -0.016245      -0.020043 -0.043055
L                             -0.018315 -0.003676       0.011751  0.015226
M                              0.020551 -0.003331      -0.008958 -0.009290
S                              0.039763 -0.014064       0.012070 -0.041510
XL                            -0.024499  0.010333      -0.008408  0.025950
XS                             0.065262  0.002992       0.007595 -0.067080
XXL                           -0.040565  0.024919      -0.003901  0.030207

                                    4XL       5XL       6XL      Free  \
Fulfilment                     0.007179  0.012019  0.007088  0.008137
Sales Channel                  0.001784 -0.001809 -0.000963 -0.016830
ship-service-level             0.007583  0.012345  0.006857  0.007512
Style                          0.037629  0.042879  0.049590 -0.046136
SKU                            0.044064  0.050212  0.058179 -0.043762
ASIN                           0.056670  0.064762  0.074865 -0.033250
Qty                            0.005065  0.006607  0.007655  0.000253
Amount                         0.035301  0.037738  0.045926 -0.014011
ship-city                      0.000598  0.000198 -0.001001 -0.005738
ship-state                     0.009184 -0.001487  0.007554  0.008209
ship-postal-code               0.002484  0.002915  0.005688  0.000176
B2B                            0.006821 -0.003945 -0.002492  0.011290
Day_of_Week                   -0.014294  0.004605 -0.003238  0.007254
Month                          0.016795  0.018020  0.023742 -0.016771
Pending                        0.001578  0.000323  0.001780 -0.001870
Pending - Waiting for Pick Up -0.002687 -0.000507  0.000864 -0.002534
Shipped                        0.007635  0.013669  0.009830  0.009078
Shipped - Damaged             -0.000160 -0.000182 -0.000211 -0.000151
Shipped - Delivered to Buyer  -0.005786 -0.009629 -0.001635 -0.007346
Shipped - Lost in Transit     -0.000358 -0.000408 -0.000472 -0.000338
Shipped - Out for Delivery    -0.000948 -0.001078 -0.001250 -0.000893
Shipped - Picked Up            0.009057 -0.001582 -0.000676 -0.001413
Shipped - Rejected by Buyer   -0.000531 -0.000605 -0.000701 -0.000501
Shipped - Returned to Seller  -0.002691  0.001640 -0.006032 -0.002018
Shipped - Returning to Seller -0.001930  0.001355 -0.002546 -0.001819
Shipping                      -0.000453 -0.000516 -0.000598 -0.000427
Bottom                        -0.003365 -0.003830 -0.004440 -0.003173
Dupatta                       -0.000277 -0.000316 -0.000366  0.088957
Ethnic Dress                  -0.005477 -0.006233 -0.007226 -0.005164
Saree                         -0.002052 -0.002336 -0.002708  0.658135
Set                           -0.032653 -0.037436 -0.043576 -0.043345
Top                           -0.017228 -0.019609 -0.022731 -0.016245
Western Dress                 -0.021256 -0.024192 -0.028044 -0.020043
kurta                          0.059087  0.067523  0.078454 -0.043055
4XL                            1.000000 -0.003764 -0.004363 -0.003118
5XL                           -0.003764  1.000000 -0.004966 -0.003549
6XL                           -0.004363 -0.004966  1.000000 -0.004114
Free                          -0.003118 -0.003549 -0.004114  1.000000
```

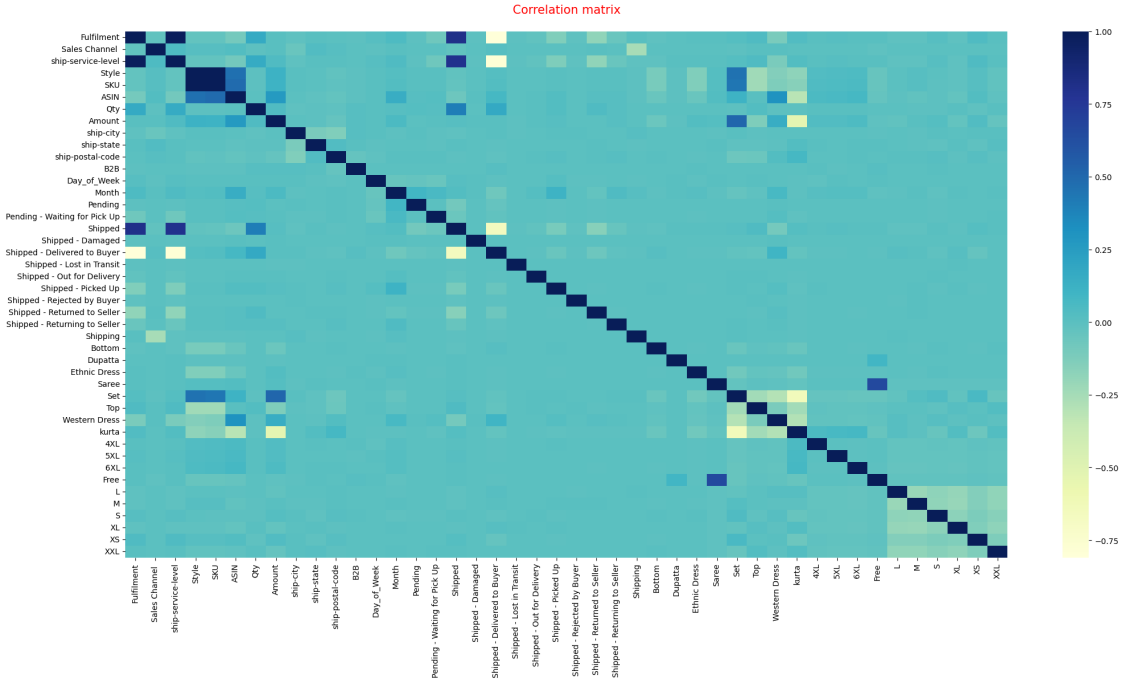
```
L                              -0.026171 -0.029786 -0.034529 -0.024677
M                              -0.026585 -0.030257 -0.035075 -0.025067
S                              -0.022473 -0.025578 -0.029650 -0.021190
XL                             -0.025272 -0.028764 -0.033343 -0.023830
XS                             -0.017702 -0.020147 -0.023355 -0.016691
XXL                            -0.023233 -0.026443 -0.030653 -0.021907

                                       L         M         S        XL  \
Fulfilment                     -0.014764 -0.014440  0.014812 -0.014743
Sales Channel                   0.000846  0.001205 -0.000422 -0.000630
ship-service-level             -0.015815 -0.014653  0.014182 -0.015164
Style                          -0.002917  0.013913 -0.003048 -0.015687
SKU                            -0.003394  0.012206 -0.005272 -0.015658
ASIN                           -0.009791 -0.020725 -0.006567 -0.029603
Qty                            -0.000803 -0.006196 -0.008724  0.004023
Amount                         -0.018899  0.006899  0.023645 -0.025101
ship-city                      -0.003764 -0.004536  0.004267 -0.003569
ship-state                     -0.002714 -0.004627 -0.012712  0.008310
ship-postal-code                0.003649  0.006190  0.012674 -0.006067
B2B                            -0.000111 -0.003323  0.004079 -0.003339
Day_of_Week                     0.002930 -0.003299  0.000608 -0.002436
Month                          -0.004312  0.005155 -0.012777  0.003933
Pending                         0.002628  0.002612 -0.003909  0.003099
Pending - Waiting for Pick Up -0.003182  0.000664 -0.004037  0.002490
Shipped                        -0.012339 -0.016521  0.008328 -0.006617
Shipped - Damaged              -0.001267 -0.001287 -0.001088 -0.001224
Shipped - Delivered to Buyer    0.014289  0.008979 -0.017232  0.012150
Shipped - Lost in Transit      -0.002834 -0.002879  0.001240  0.007407
Shipped - Out for Delivery      0.001242  0.001035  0.004670 -0.003407
Shipped - Picked Up             0.001440  0.003923 -0.004999  0.004014
Shipped - Rejected by Buyer    -0.004204  0.004548  0.001344 -0.001780
Shipped - Returned to Seller   -0.000265  0.005613  0.001996 -0.000802
Shipped - Returning to Seller -0.001153  0.002715 -0.000144  0.002217
Shipping                        0.004250 -0.001056 -0.003078  0.001885
Bottom                         -0.001585 -0.006800  0.006161 -0.000081
Dupatta                        -0.002195 -0.002230 -0.001885 -0.002120
Ethnic Dress                   -0.001494 -0.005194  0.005441  0.001426
Saree                          -0.016241 -0.016497 -0.013946 -0.015683
Set                            -0.018315  0.020551  0.039763 -0.024499
Top                            -0.003676 -0.003331 -0.014064  0.010333
Western Dress                   0.011751 -0.008958  0.012070 -0.008408
kurta                           0.015226 -0.009290 -0.041510  0.025950
4XL                            -0.026171 -0.026585 -0.022473 -0.025272
5XL                            -0.029786 -0.030257 -0.025578 -0.028764
6XL                            -0.034529 -0.035075 -0.029650 -0.033343
Free                           -0.024677 -0.025067 -0.021190 -0.023830
L                               1.000000 -0.210384 -0.177848 -0.200001
```

M	-0.210384	1.000000	-0.180660	-0.203163
S	-0.177848	-0.180660	1.000000	-0.171743
XL	-0.200001	-0.203163	-0.171743	1.000000
XS	-0.140086	-0.142301	-0.120294	-0.135278
XXL	-0.183862	-0.186769	-0.157885	-0.177551

	XS	XXL
Fulfilment	0.038859	0.007061
Sales Channel	-0.002018	0.002449
ship-service-level	0.039860	0.007488
Style	0.013746	-0.020517
SKU	0.011231	-0.019459
ASIN	0.031475	-0.011025
Qty	-0.014025	0.009800
Amount	0.033704	-0.035445
ship-city	0.006096	0.006101
ship-state	-0.011632	0.012204
ship-postal-code	0.016307	-0.009545
B2B	-0.002824	-0.000877
Day_of_Week	0.002159	0.000714
Month	-0.003735	0.009484
Pending	-0.001142	-0.001044
Pending - Waiting for Pick Up	0.000402	0.004583
Shipped	0.018738	0.008480
Shipped - Damaged	-0.000857	-0.001125
Shipped - Delivered to Buyer	-0.037791	-0.001388
Shipped - Lost in Transit	-0.001917	0.001070
Shipped - Out for Delivery	-0.001723	0.002831
Shipped - Picked Up	-0.007718	0.001153
Shipped - Rejected by Buyer	0.003130	-0.001314
Shipped - Returned to Seller	-0.001983	-0.003042
Shipped - Returning to Seller	-0.003744	0.000435
Shipping	0.001077	-0.003182
Bottom	-0.004769	0.004694
Dupatta	-0.001485	-0.001949
Ethnic Dress	-0.003598	-0.001332
Saree	-0.010985	-0.014418
Set	0.065262	-0.040565
Top	0.002992	0.024919
Western Dress	0.007595	-0.003901
kurta	-0.067080	0.030207
4XL	-0.017702	-0.023233
5XL	-0.020147	-0.026443
6XL	-0.023355	-0.030653
Free	-0.016691	-0.021907
L	-0.140086	-0.183862
M	-0.142301	-0.186769

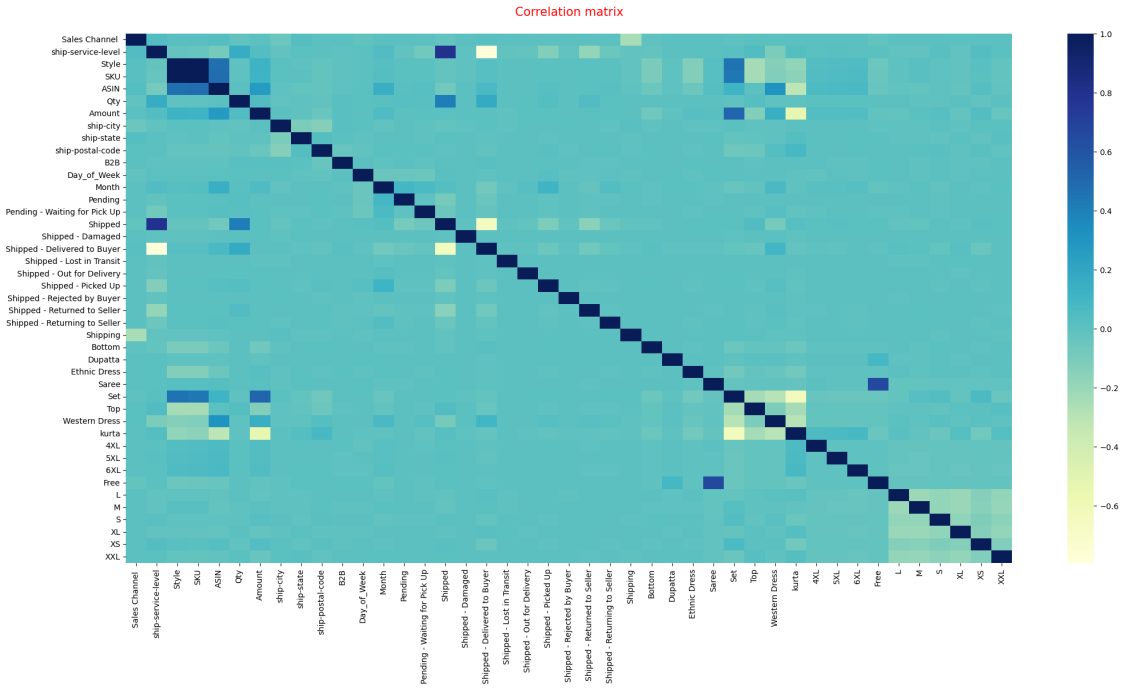
```
S                          -0.120294 -0.157885
XL                         -0.135278 -0.177551
XS                          1.000000 -0.124361
XXL                        -0.124361  1.000000
```

[71]:
```
# Plotting a heatmap for ispecting correlation between the numeric variables

plt.figure(figsize=[25,12])
sns.heatmap(amazon_transformed_df.corr(), cmap="YlGnBu")
plt.title("Correlation matrix\n", fontdict={"fontsize":15, "fontweight":5,␣
 ↪"color":"Red"})
plt.show()
```



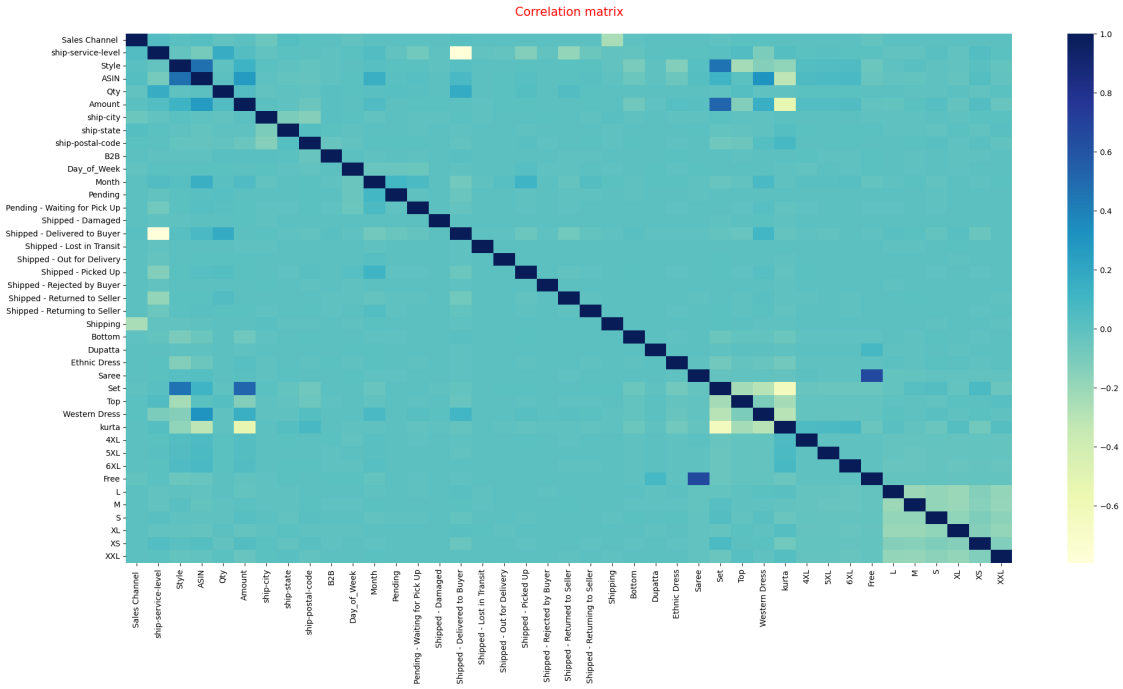
[72]:
```
# Dropping column Fulfilment

amazon_transformed_df.drop(columns="Fulfilment", inplace=True)
```

[73]:
```
# Plotting a heatmap for ispecting correlation between the numeric variables

plt.figure(figsize=[25,12])
sns.heatmap(amazon_transformed_df.corr(), cmap="YlGnBu")
plt.title("Correlation matrix\n", fontdict={"fontsize":15, "fontweight":5,␣
 ↪"color":"Red"})
plt.show()
```



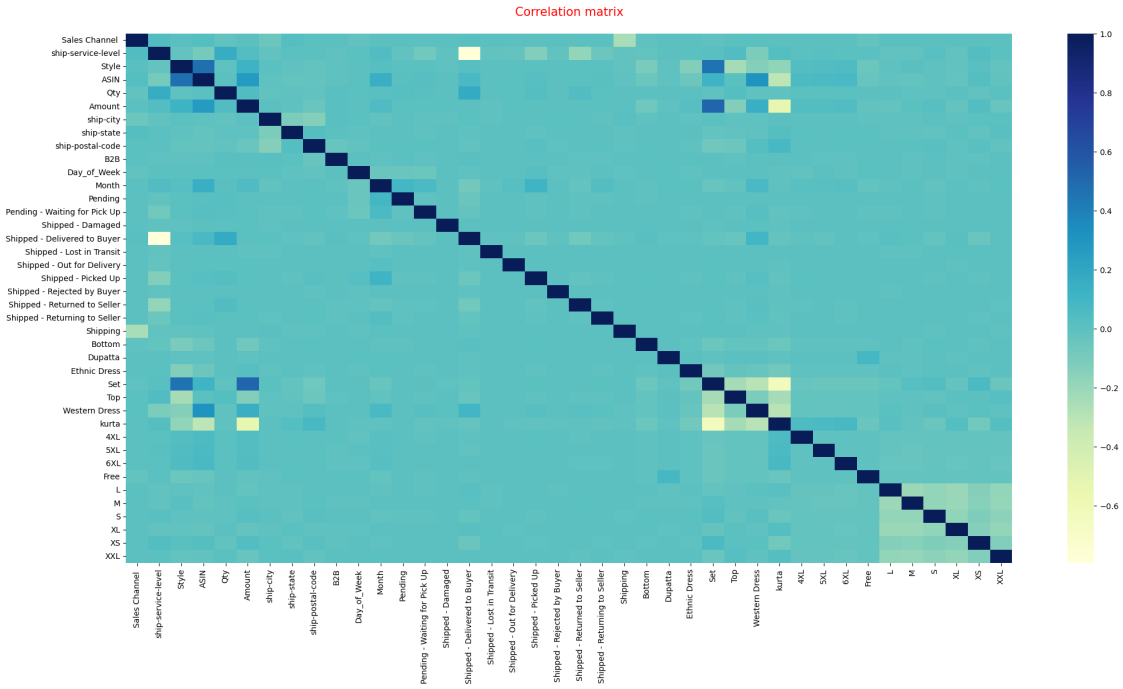
[74]:
```
# Dropping column Fulfilment

amazon_transformed_df.drop(columns=["SKU", "Shipped"], inplace=True)
```

[75]:
```
# Plotting a heatmap for ispecting correlation between the numeric variables

plt.figure(figsize=[25,12])
sns.heatmap(amazon_transformed_df.corr(), cmap="YlGnBu")
plt.title("Correlation matrix\n", fontdict={"fontsize":15, "fontweight":5,␣
  ↪"color":"Red"})
plt.show()
```



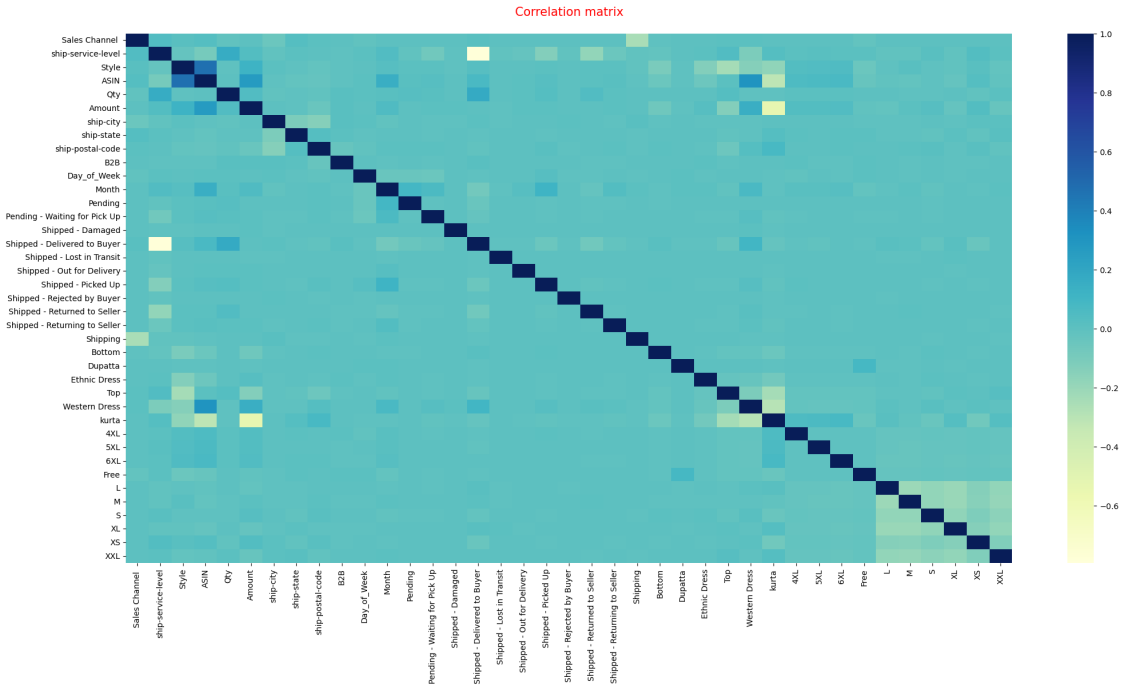
[76]:
```
# Dropping column Fulfilment

amazon_transformed_df.drop(columns="Saree", inplace=True)
```

[77]:
```
# Plotting a heatmap for ispecting correlation between the numeric variables

plt.figure(figsize=[25,12])
sns.heatmap(amazon_transformed_df.corr(), cmap="YlGnBu")
plt.title("Correlation matrix\n", fontdict={"fontsize":15, "fontweight":5,␣
 ↪"color":"Red"})
plt.show()
```



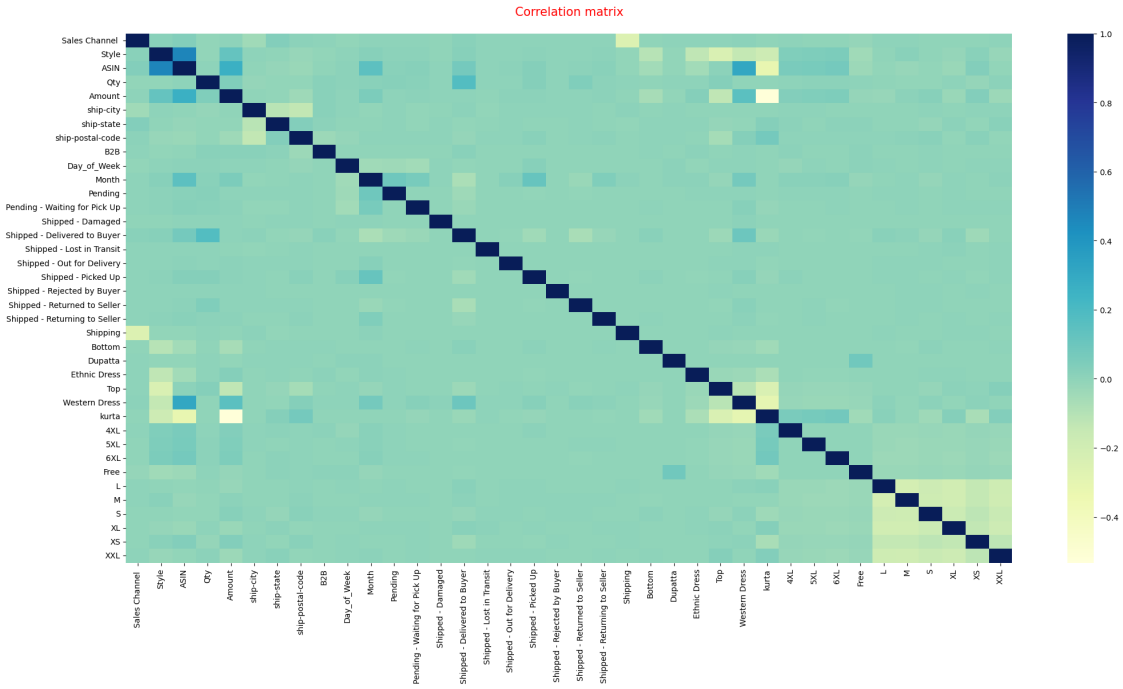
[78]:
```
# Dropping column Fulfilment

amazon_transformed_df.drop(columns=["Set"], inplace=True)
```

[79]:
```
# Plotting a heatmap for ispecting correlation between the numeric variables

plt.figure(figsize=[25,12])
sns.heatmap(amazon_transformed_df.corr(), cmap="YlGnBu")
plt.title("Correlation matrix\n", fontdict={"fontsize":15, "fontweight":5,␣
 ↪"color":"Red"})
plt.show()
```



[80]:
```
# Dropping column Fulfilment

amazon_transformed_df.drop(columns=["ship-service-level"], inplace=True)
```

[81]:
```
# Plotting a heatmap for ispecting correlation between the numeric variables

plt.figure(figsize=[25,12])
sns.heatmap(amazon_transformed_df.corr(), cmap="YlGnBu")
plt.title("Correlation matrix\n", fontdict={"fontsize":15, "fontweight":5,␣
 ↪"color":"Red"})
plt.show()
```



1.1.9 Train Test Split

[82]:
```
amazon_transformed_df.columns
```

[82]:
```
Index(['Sales Channel ', 'Style', 'ASIN', 'Qty', 'Amount', 'ship-city',
       'ship-state', 'ship-postal-code', 'B2B', 'Day_of_Week', 'Month',
       'Pending', 'Pending - Waiting for Pick Up', 'Shipped - Damaged',
       'Shipped - Delivered to Buyer', 'Shipped - Lost in Transit',
       'Shipped - Out for Delivery', 'Shipped - Picked Up',
       'Shipped - Rejected by Buyer', 'Shipped - Returned to Seller',
       'Shipped - Returning to Seller', 'Shipping', 'Bottom', 'Dupatta',
       'Ethnic Dress', 'Top', 'Western Dress', 'kurta', '4XL', '5XL', '6XL',
       'Free', 'L', 'M', 'S', 'XL', 'XS', 'XXL'],
      dtype='object')
```

[83]:
```
# Separating target variable and independent variables

y = amazon_transformed_df.pop("Amount")
X = amazon_transformed_df
```

[84]:
```
# Split the data into training and testing sets (70% training and 30% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

```
# The 'test_size' parameter determines the proportion of data to be allocated␣
 ↪to the test set.
# 'random_state' ensures reproducibility by fixing the random seed.
```

[85]:
```
X.describe()
```

[85]:
```
       Sales Channel            Style            ASIN             Qty  \
count   128941.000000  128941.000000  128941.000000  128941.000000
mean         0.999038     762.466361    3717.440403       0.900722
std          0.030996     380.593634    2048.858166       0.299036
min          0.000000       0.000000       0.000000       0.000000
25%          1.000000     438.000000    1843.000000       1.000000
50%          1.000000     777.000000    3887.000000       1.000000
75%          1.000000    1143.000000    5612.000000       1.000000
max          1.000000    1376.000000    7189.000000       1.000000

           ship-city     ship-state            B2B    Day_of_Week  \
count  128941.000000  128941.000000  128941.000000  128941.000000
mean     3209.553920      20.644706       0.006755       3.011951
std      1942.727944      10.210218       0.081911       2.026497
min         0.000000       0.000000       0.000000       0.000000
25%      1363.000000      15.000000       0.000000       1.000000
50%      2947.000000      20.000000       0.000000       3.000000
75%      4595.000000      31.000000       0.000000       5.000000
max      7032.000000      36.000000       1.000000       6.000000

               Month        Pending  Pending - Waiting for Pick Up  \
count  128941.000000  128941.000000                  128941.000000
mean        4.909214       0.005103                       0.002179
std         0.818410       0.071254                       0.046632
min         3.000000       0.000000                       0.000000
25%         4.000000       0.000000                       0.000000
50%         5.000000       0.000000                       0.000000
75%         6.000000       0.000000                       0.000000
max         6.000000       1.000000                       1.000000

       Shipped - Damaged  Shipped - Delivered to Buyer  \
count      128941.000000                 128941.000000
mean            0.000008                      0.223063
std             0.002785                      0.416302
min             0.000000                      0.000000
25%             0.000000                      0.000000
50%             0.000000                      0.000000
75%             0.000000                      0.000000
max             1.000000                      1.000000

       Shipped - Lost in Transit  Shipped - Out for Delivery  \
```

```
              128941.000000              128941.000000
mean               0.000039                   0.000271
std                0.006227                   0.016473
min                0.000000                   0.000000
25%                0.000000                   0.000000
50%                0.000000                   0.000000
75%                0.000000                   0.000000
max                1.000000                   1.000000

       Shipped - Picked Up  Shipped - Rejected by Buyer  \
count       128941.000000                128941.000000
mean             0.007546                     0.000085
std              0.086540                     0.009236
min              0.000000                     0.000000
25%              0.000000                     0.000000
50%              0.000000                     0.000000
75%              0.000000                     0.000000
max              1.000000                     1.000000

       Shipped - Returned to Seller  Shipped - Returning to Seller  \
count                 128941.000000                  128941.000000
mean                      0.015123                       0.001125
std                       0.122043                       0.033516
min                       0.000000                       0.000000
25%                       0.000000                       0.000000
50%                       0.000000                       0.000000
75%                       0.000000                       0.000000
max                       1.000000                       1.000000

           Shipping         Bottom        Dupatta   Ethnic Dress  \
count  128941.000000  128941.000000  128941.000000  128941.000000
mean        0.000062       0.003412       0.000023       0.008989
std         0.007877       0.058316       0.004823       0.094382
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

                Top  Western Dress          kurta            4XL  \
count  128941.000000  128941.000000  128941.000000  128941.000000
mean        0.082363       0.120202       0.386681       0.003296
std         0.274918       0.325199       0.486991       0.057317
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       1.000000       0.000000
```

```
max         1.000000       1.000000       1.000000       1.000000

                5XL            6XL            Free              L  \
count  128941.000000  128941.000000  128941.000000  128941.000000
mean        0.004266       0.005724       0.002932       0.171575
std         0.065172       0.075438       0.054065       0.377012
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

                  M              S             XL             XS  \
count  128941.000000  128941.000000  128941.000000  128941.000000
mean        0.176081       0.132487       0.161872       0.086551
std         0.380890       0.339021       0.368335       0.281177
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

                XXL
count  128941.00000
mean        0.14032
std         0.34732
min         0.00000
25%         0.00000
50%         0.00000
75%         0.00000
max         1.00000
```

1.1.10 Scaling

[86]:
```
# Create a StandardScaler instance
scaler = StandardScaler()

# Fit and transform the scaler on the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data using the same scaler
X_test_scaled = scaler.transform(X_test)
```

1.2 Model Building

1.2.1 Linear Regression Model

[87]:
```
# Create a Linear Regression model
regression_model = LinearRegression()

# Fit the model on the training data
regression_model.fit(X_train_scaled, y_train)
```

[87]:
```
LinearRegression()
```

[88]:
```
# Make predictions on the train and test data

y_train_pred = regression_model.predict(X_train_scaled)
y_test_pred = regression_model.predict(X_test_scaled)
```

[89]:
```
# Function to evaluate train and test scores

def train_test_score(y_train, y_test, y_train_pred, y_test_pred):

    # Calculate training metrics
    mse_train = mean_squared_error(y_train, y_train_pred)
    mae_train = mean_absolute_error(y_train, y_train_pred)
    r2_train = r2_score(y_train, y_train_pred)

    # Calculate test metrics
    mse_test = mean_squared_error(y_test, y_test_pred)
    mae_test = mean_absolute_error(y_test, y_test_pred)
    r2_test = r2_score(y_test, y_test_pred)

    # Return the scores in a dictionary
    scores = {
        'Mean Squared Error (MSE) train': mse_train,
        'Mean Absolute Error (MAE) train': mae_train,
        'R-squared (R2) Score train': r2_train,
        'Mean Squared Error (MSE) test': mse_test,
        'Mean Absolute Error (MAE) test': mae_test,
        'R-squared (R2) Score test': r2_test
    }
    return scores
```

[90]:
```
# Evaluating the linear regression model

train_test_score(y_train, y_test, y_train_pred, y_test_pred)
```

[90]:
```
{'Mean Squared Error (MSE) train': 40967.32276625967,
 'Mean Absolute Error (MAE) train': 144.55014641546012,
```

```
 'R-squared (R2) Score train': 0.42492240504267076,
 'Mean Squared Error (MSE) test': 41004.90057075955,
 'Mean Absolute Error (MAE) test': 144.4904289914778,
 'R-squared (R2) Score test': 0.43025759785226403}
```

Mean Squared Error (MSE): The MSE represents the average of the squared differences between the actual target values "Amount" and the predicted values. As we can see, the MSE is approximately 41004.90. Considering the scale of the "Amount" variable, which has a maximum value of 1442 and a mean of 644.62, an MSE of 41004.90 is relatively high. It indicates a substantial level of error in our model's predictions, especially when compared to the range and mean of the target variable.

R-squared (R2) Score: The R2 score measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating a better fit. In our result, the R2 score is approximately 0.43, which means that our model explains about 43% of the variance in the target variable. An R2 score of 1 would mean a perfect fit, while a score of 0 indicates that the model doesn't explain any of the variance.

These metrics provide an initial assessment of our linear regression model's performance. We might want to explore more complex regression models or fine-tune hyperparameters to improve predictive accuracy based on our specific modeling goals and the nature of our data. Additionally, we can use other regression evaluation metrics and techniques for model validation and improvement.

[91]:
```
y.describe()
```

[91]:
```
count    128941.000000
mean        644.624374
std         267.317377
min           0.000000
25%         459.000000
50%         605.000000
75%         771.000000
max        1442.000000
Name: Amount, dtype: float64
```

1.2.2 Scaling

[92]:
```
#Create a MinMaxScaler instance
mscaler = MinMaxScaler()

# Fit and transform the scaler on the training data
X_train_mscaled = mscaler.fit_transform(X_train)

# Transform the test data using the same scaler
X_test_mscaled = mscaler.transform(X_test)
```

[93]:
```
# Create a Linear Regression model
regression_model_2 = LinearRegression()
```

```
# Fit the model on the training data
regression_model_2.fit(X_train_mscaled, y_train)
```

[93]:
```
LinearRegression()
```

[94]:
```
# Make predictions on the train and test data

y_train_pred_m = regression_model_2.predict(X_train_mscaled)
y_test_pred_m = regression_model_2.predict(X_test_mscaled)
```

[95]:
```
# Evaluating the linear regression model

train_test_score(y_train, y_test, y_train_pred_m, y_test_pred_m)
```

[95]:
```
{'Mean Squared Error (MSE) train': 40967.32276625968,
 'Mean Absolute Error (MAE) train': 144.55014641546012,
 'R-squared (R2) Score train': 0.42492240504267076,
 'Mean Squared Error (MSE) test': 41004.90057075955,
 'Mean Absolute Error (MAE) test': 144.49042899147784,
 'R-squared (R2) Score test': 0.43025759785226403}
```

1.2.3 Building Linear Regression Model using K fold Cross validation

GridSearchCV with 5 folds

[96]:
```
# Building cross validation model

model_cv = LinearRegression().fit(X, y)
```

[98]:
```
# Evaluating the cross validation
predictions_cv = cross_val_predict(model_cv, X, y, cv=10)
accuracy = metrics.r2_score(y, predictions_cv)
print("Cross-Predicted Accuracy:", accuracy)
print('Mean Squared Error: {}'.format(mean_squared_error(y, predictions_cv)))
```

```
Cross-Predicted Accuracy: 0.4232143247947172
Mean Squared Error: 41215.96571486316
```

[99]:
```
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(model_cv, X, y, cv=10)
print("Cross-Validation Scores:", cv_scores)
```

```
Cross-Validation Scores: [0.43612034 0.47852224 0.46827063 0.42888998 0.39048217
0.38038634
 0.41980786 0.42248381 0.40385577 0.39109206]
```

[101]:
```
from sklearn.metrics import make_scorer
```

[106]:
```
# Define scoring metrics
scoring = {'neg_mean_squared_error', 'neg_mean_absolute_error', 'r2'}
```

[109]:
```
# Perform 10-fold cross-validation


scoring_metrics = ['neg_mean_squared_error', 'neg_mean_absolute_error', 'r2']

for metric in scoring_metrics:
    cv_results = cross_val_score(model_cv, X, y, cv=10, scoring=metric)
    print(f"{metric}: {cv_results.mean()}")
```

```
neg_mean_squared_error: -41215.98785659733
neg_mean_absolute_error: -145.2297278133783
r2: 0.42199112020896495
```

[111]:
```
from sklearn.model_selection import cross_validate
# Perform 10-fold cross-validation with multiple scoring metrics
cv_results = cross_validate(model_cv, X, y, cv=10, scoring=scoring_metrics,␣
 ↪return_train_score=True)

# Display results for each metric
for metric in scoring_metrics:
    train_score_mean = - cv_results[f'train_{metric}'].mean()
    test_score_mean = - cv_results[f'test_{metric}'].mean()
    print(f"{metric} - Train: {train_score_mean}, Test: {test_score_mean}")
```

```
neg_mean_squared_error - Train: 40961.54654728642, Test: 41215.98785659733
neg_mean_absolute_error - Train: 144.52352680564147, Test: 145.2297278133783
r2 - Train: -0.42675936148280735, Test: -0.42199112020896495
```

1.2.4 Decision Tree

[193]:
```
# Define the Decision Tree model
decisiontree_model = DecisionTreeRegressor(random_state=42)
```

[194]:
```
# Fit the Decision Tree model on the scaled training data
decisiontree_model.fit(X_train_scaled, y_train)
```

[194]:
```
DecisionTreeRegressor(random_state=42)
```

[195]:
```
# Make predictions on the train and test data

y_train_pred_dt = decisiontree_model.predict(X_train_scaled)
y_test_pred_dt = decisiontree_model.predict(X_test_scaled)
```

[196]:
```
# Evaluating the linear regression model

train_test_score(y_train, y_test, y_train_pred_dt, y_test_pred_dt)
```

[196]:
```
{'Mean Squared Error (MSE) train': 76.54694920874974,
 'Mean Absolute Error (MAE) train': 0.303295774335793,
 'R-squared (R2) Score train': 0.9989254744396296,
 'Mean Squared Error (MSE) test': 30393.500635383116,
 'Mean Absolute Error (MAE) test': 56.93440438952513,
 'R-squared (R2) Score test': 0.5776976453875313}
```

1.2.5 Random Forest Model

[197]:
```
# Define the Random Forest model
randomforest_model = RandomForestRegressor(random_state=42)
```

[198]:
```
# Fit the Random Forest model on the scaled training data
randomforest_model.fit(X_train_scaled, y_train)
```

[198]:
```
RandomForestRegressor(random_state=42)
```

[199]:
```
# Make predictions on the train and test data

y_train_pred_rf = randomforest_model.predict(X_train_scaled)
y_test_pred_rf = randomforest_model.predict(X_test_scaled)
```

[200]:
```
# Evaluating the linear regression model

train_test_score(y_train, y_test, y_train_pred_rf, y_test_pred_rf)
```

[200]:
```
{'Mean Squared Error (MSE) train': 2311.5566163961607,
 'Mean Absolute Error (MAE) train': 19.32976353534386,
 'R-squared (R2) Score train': 0.9675515916148754,
 'Mean Squared Error (MSE) test': 15974.356113372882,
 'Mean Absolute Error (MAE) test': 51.34388551592598,
 'R-squared (R2) Score test': 0.7780443825466432}
```

Best Model

[201]:
```
# Define the hyperparameters to tune
parameters = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

[202]:
```
# Use GridSearchCV for hyperparameter tuning
gridsearch = GridSearchCV(estimator=randomforest_model, param_grid=parameters,␣
↪cv=5, scoring='neg_mean_squared_error', n_jobs=-1)
gridsearch.fit(X_train_scaled, y_train)
```

[202]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(random_state=42), n_jobs=-1,
             param_grid={'max_depth': [None, 10, 20],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [50, 100, 150]},
             scoring='neg_mean_squared_error')
```

[203]:
```
# Get the best hyperparameters
best_params = gridsearch.best_params_
print("Best Hyperparameters:", best_params)
```

```
Best Hyperparameters: {'max_depth': None, 'min_samples_leaf': 4,
'min_samples_split': 10, 'n_estimators': 150}
```

[204]:
```
# Make predictions on the test set using the best model
best_rf_model = gridsearch.best_estimator_
y_pred_rfb = best_rf_model.predict(X_test_scaled)
```

[205]:
```
# Evaluate the model
mse_rfb = mean_squared_error(y_test, y_pred_rfb)
r2_rfb = r2_score(y_test, y_pred_rfb)

print(f"Mean Squared Error (MSE): {mse_rfb}")
print(f"R-squared (R2) Score: {r2_rfb}")
```

```
Mean Squared Error (MSE): 14214.278477916143
R-squared (R2) Score: 0.8024997731471198
```

[206]:
```
# Fit the model to your training data
best_rf_model.fit(X_train_scaled, y_train)
```

[206]:
```
RandomForestRegressor(min_samples_leaf=4, min_samples_split=10,
                      n_estimators=150, random_state=42)
```

[207]:
```
# Make predictions on the train and test data

y_train_pred_rfb = best_rf_model.predict(X_train_scaled)
y_test_pred_rfb = best_rf_model.predict(X_test_scaled)
```

[208]:
```
# Evaluating the linear regression model

train_test_score(y_train, y_test, y_train_pred_rfb, y_test_pred_rfb)
```

[208]:
```
{'Mean Squared Error (MSE) train': 8840.233046487932,
 'Mean Absolute Error (MAE) train': 36.608268752345225,
 'R-squared (R2) Score train': 0.8759054872039729,
 'Mean Squared Error (MSE) test': 14214.278477916143,
 'Mean Absolute Error (MAE) test': 48.67132288151028,
 'R-squared (R2) Score test': 0.8024997731471198}
```

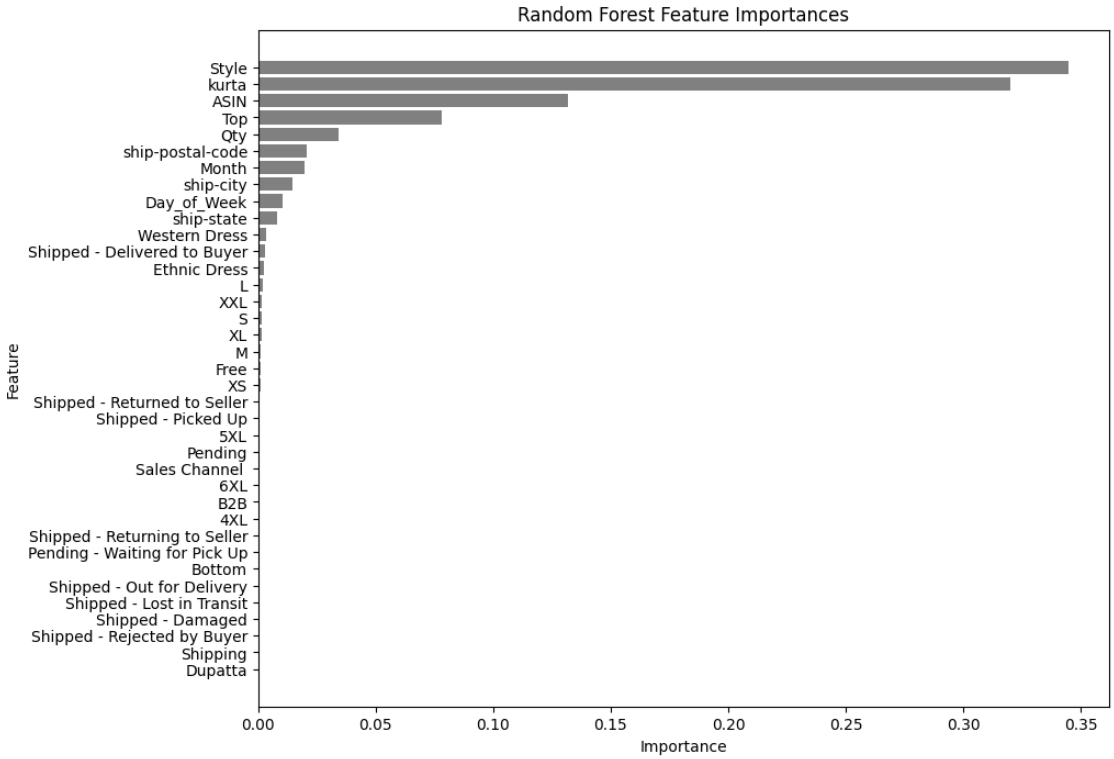
1.2.6 Feature Importance using Random Forest

[114]:
```
# Get feature importances
feature_importances = best_rf_model.feature_importances_
```

[115]:
```
# Get the names of the features
featurenames = X_train.columns
```

[116]:
```
# Sort the features based on their importance
sortedindices = feature_importances.argsort()[::-1]
```

[125]:
```
plt.figure(figsize=(10, 8))
plt.barh(range(X_train.shape[1]), feature_importances[sortedindices[::-1]],␣
 ↪color='grey')
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.yticks(range(X_train.shape[1]), featurenames[sortedindices][::-1])
plt.title("Random Forest Feature Importance")
plt.show()
```



1.2.7 Principal Component Analysis

[223]:
```
# Apply PCA to the scaled training data
pca = PCA()
X_train_pca = pca.fit_transform(X_train_scaled)
```

[224]:
```
# Explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_
```

[225]:
```
# Cumulative explained variance
cuml_explained_variance = explained_variance_ratio.cumsum()
print("Cumulative Explained Variance: ",cuml_explained_variance)
```

```
Cumulative Explained Variance:  [0.04947894 0.08745264 0.12273156 0.15638795
0.18954002 0.22210412
 0.25454777 0.28662045 0.31827375 0.34912089 0.37912646 0.4083249
 0.43726192 0.46583872 0.49360746 0.52132523 0.54866262 0.57584485
 0.60291797 0.62996305 0.65699435 0.68402061 0.71101561 0.73796603
 0.76489158 0.79149919 0.81761427 0.84287597 0.86802494 0.8923185
 0.91480115 0.93652858 0.95645352 0.97558263 0.98941075 0.99636836
 1.        ]
```

[226]:
```
# Determine the optimal number of components to retain based on the explained␣
↪variance
optimalcomponents = len(cuml_explained_variance[cuml_explained_variance <= 0.
↪95]) + 1
print(f"Optimal Number of Components: {optimalcomponents}")
```

```
Optimal Number of Components: 33
```

[227]:
```
# Apply PCA to the test data using the same components
X_test_pca = pca.transform(X_test_scaled)[:, :optimalcomponents]
```

[228]:
```
# Fit a linear regression model on the PCA-transformed training data

pca_rf_model = gridsearch.best_estimator_
```

[229]:
```
y_pred_rf_pca = pca_rf_model.fit(X_train_pca[:, :optimalcomponents], y_train)
```

[230]:
```
# Make predictions on the train and test data

y_train_pred_pca = pca_rf_model.predict(X_train_pca[:, :optimalcomponents])
y_test_pred_pca = pca_rf_model.predict(X_test_pca)
```

[231]:
```
# Make predictions on the test set
y_pred_rf_pca = pca_rf_model.predict(X_test_pca)

# Evaluate the model
mse_rf_pca = mean_squared_error(y_test, y_pred_rf_pca)
r2_rf_pca = r2_score(y_test, y_pred_rf_pca)

print(f"Mean Squared Error (MSE): {mse_rf_pca}")
print(f"R-squared (R2) Score: {r2_rf_pca}")
```

```
Mean Squared Error (MSE): 31221.499635193613
R-squared (R2) Score: 0.5661930170977031
```

1.2.8 Ridge Regression

[87]:
```
# Create a Ridge regression model

ridge_mod = Ridge(alpha=1.0)
```

[88]:
```
# Fit the Ridge model on the scaled training data

ridge_mod.fit(X_train_scaled, y_train)
```

[88]:
```
Ridge()
```

[93]:
```
# Make predictions on the train test set

y_train_pred_ridge = ridge_mod.predict(X_train_scaled)
y_test_pred_ridge = ridge_mod.predict(X_test_scaled)
```

[94]:
```
# Evaluating the linear regression model

train_test_score(y_train, y_test, y_train_pred_ridge, y_test_pred_ridge)
```

[94]:
```
{'Mean Squared Error (MSE) train': 40967.322785332755,
 'Mean Absolute Error (MAE) train': 144.5501975153157,
 'R-squared (R2) Score train': 0.4249224047749329,
 'Mean Squared Error (MSE) test': 41004.907625439664,
 'Mean Absolute Error (MAE) test': 144.49048827114913,
 'R-squared (R2) Score test': 0.43025749983104367}
```

1.2.9 Lasso Regression

[95]:
```
# Create a Lasso regression model

lasso_mod = Lasso(alpha=1.0)
```

[96]:
```
# Fit the Lasso model on the scaled training data

lasso_mod.fit(X_train_scaled, y_train)
```

[96]:
```
Lasso()
```

[97]:
```
# Make predictions on the train test set



y_train_pred_lasso = lasso_mod.predict(X_train_scaled)
y_test_pred_lasso = lasso_mod.predict(X_test_scaled)
```

[99]:
```
# Evaluating the linear regression model

train_test_score(y_train, y_test, y_train_pred_lasso, y_test_pred_lasso)
```

[99]:
```
{'Mean Squared Error (MSE) train': 41014.645337689275,
 'Mean Absolute Error (MAE) train': 144.68117839041915,
 'R-squared (R2) Score train': 0.4242581158305052,
 'Mean Squared Error (MSE) test': 41039.120732409756,
 'Mean Absolute Error (MAE) test': 144.59486876030252,
 'R-squared (R2) Score test': 0.4297821259738033}
```

Thank you!