Introduction

This task is a consultancy job for JC Penney, a prominent American retail company. It involves the processing of 5 datasets which contains information on products sold by JC Penney, alongside synthetic customer reviews. The data are stored in both JSON and CSV file formats, and various data manipulation and analysis skills were used to address the task.

The task was broken down into 7 sections/objectives as follows:

- Choosing the Python Libraries
- Data Exploration
- Data Validation
- Data Visualisation
- Data Analysis
- Data Augmentation
- Recommendations

To achieve the aforementioned objectives, a number of python libraries were imported and used for manipulating the datasets.

1.0 CHOOSING THE PYTHON LIBRARIES

Based on the file formats, and the objectives of this task, the following python libraries are required:

In [136…

```
# Needed for working with the JSON data; jcpenney_products.json and jcpenney_reviewers.json
import json

# Needed to provide DataFrame structures for efficient data analysis
import pandas as pd

# Needed to for mathematical functions
import numpy as np

# To create plots and visualizations
import matplotlib.pyplot as plt

# Just like plt, it is needed for informative and attractive statistical graphics
import seaborn as sns

# for working with human language data such as the 'reviews' in the dataset
import nltk
```

```
# allows for interaction with the file system and executing commands
import os

# Required for sentiment analysis
nltk.download('vader_lexicon', quiet=True)

# vital for determining the sentiment polarity of sentences
from nltk.sentiment import SentimentIntensityAnalyzer

# Required for graphical representations of words that appear more frequently
from wordcloud import WordCloud

# provides tools for working with fonts and text in image processing
from PIL import ImageFont

# Needed to filter out common words when executing natural language processing tasks
from nltk.corpus import stopwords
```

2.0 DATA EXPLORATION

In [137…

```
# 2.1 Exploration of the two JSON files

# Load jcpenney_products.json file
# Load jcpenney_reviewers.json file
# Specify the file path

jcpenney_products_file_path = "jcpenney_products.json"

with open(jcpenney_products_file_path, 'r') as file:
    products_data = [json.loads(line) for line in file]


jcpenney_reviewers_file_path = "jcpenney_reviewers.json"

with open(jcpenney_reviewers_file_path, 'r') as file:
    reviewers_data = [json.loads(line) for line in file]

# The lines of code read the contents of jcpenney_products.json and jcpenney_reviewers.json line by line
# Converts each line into a Python object, and stores these objects in products_data and reviewers_data lists
```

In [138…

```
# 2.1.1 Display the first 2 elements of products_data

# Use list slicing operation to extract the elements from index 0 to index 1 (inclusive)
# Output the specified content

print(products_data[:2])
```

```
# The purpose is to quickly inspect and understand the structure of the data in the list
```

```
[{'uniq_id': 'b6c0b6bea69c722939585baeac73c13d', 'sku': 'pp5006380337', 'name_title': 'Alfred Dunner® Essential Pull On Capri Pant', 'descrip
tion': 'You\'ll return to our Alfred Dunner pull-on capris again and again when you want an updated, casual look and all the comfort you lov
e. \xa0 elastic waistband approx. 19-21" inseam slash pockets polyester washable imported \xa0 \xa0 \xa0', 'list_price': '41.09', 'sale_pric
e': '24.16', 'category': 'alfred dunner', 'category_tree': 'jcpenney|women|alfred dunner', 'average_product_rating': 2.625, 'product_url': 'h
ttp://www.jcpenney.com/alfred-dunner-essential-pull-on-capri-pant/prod.jump?ppId=pp5006380337&catId=cat100210079&&_dyncharset=UTF-8&urlState
=/women/shop-brands/alfred-dunner/yellow/_/N-gkmp33Z132/cat.jump', 'product_image_urls': 'http://s7d9.scene7.com/is/image/JCPenney/DP12282015
17142050M.tif?hei=380&amp;wid=380&op_usm=.4,.8,0,0&resmode=sharp2&op_usm=1.5,.8,0,0&resmode=sharp', 'brand': 'Alfred Dunner', 'total_number_r
eviews': 8, 'Reviews': [{'User': 'fsdv4141', 'Review': 'You never have to worry about the fit...Alfred Dunner clothing sizes are true to size
and fits perfectly. Great value for the money.', 'Score': 2}, {'User': 'krpz1113', 'Review': 'Good quality fabric. Perfect fit. Washed very w
ell no iron.', 'Score': 4}, {'User': 'mbmg3241', 'Review': 'I do not normally wear pants or capris that have an elastic waist, but I decided
to try these since they were on sale and I loved the color. I was very surprised at how comfortable they are and wear really well even wearin
g all day. I will buy this style again!', 'Score': 4}, {'User': 'zeqg1222', 'Review': 'I love these capris! They fit true to size and are so
comfortable to wear. I am planning to order more of them.', 'Score': 1}, {'User': 'nvfn3212', 'Review': 'This product is very comfortable and
the fabric launders very well', 'Score': 1}, {'User': 'aajh3423', 'Review': 'I did not like the fabric. It is 100% polyester I thought it was
different.I bought one at the store apprx two monts ago, and I thought it was just like it', 'Score': 5}, {'User': 'usvp2142', 'Review': 'Wha
t a great deal. Beautiful Pants. Its more than I expected.', 'Score': 3}, {'User': 'yemw3321', 'Review': 'Alfred Dunner has great pants, good
fit and very comfortable', 'Score': 1}], 'Bought With': ['898e42fe937a33e8ce5e900ca7a4d924', '8c02c262567a2267cd207e35637feb1c', 'b62dd54545c
dc1a05d8aaa2d25aed996', '0da4c2dcc8cfa0e71200883b00d22b30', '90c46b841e2eeece992c57071387899c']}, {'uniq_id': '93e5272c51d8cce02597e3ce67b7ad
0a', 'sku': 'pp5006380337', 'name_title': 'Alfred Dunner® Essential Pull On Capri Pant', 'description': 'You\'ll return to our Alfred Dunner
pull-on capris again and again when you want an updated, casual look and all the comfort you love. \xa0 elastic waistband approx. 19-21" inse
am slash pockets polyester washable imported \xa0 \xa0 \xa0', 'list_price': '41.09', 'sale_price': '24.16', 'category': 'alfred dunner', 'cat
egory_tree': 'jcpenney|women|alfred dunner', 'average_product_rating': 3.0, 'product_url': 'http://www.jcpenney.com/alfred-dunner-essential-p
ull-on-capri-pant/prod.jump?ppId=pp5006380337&catId=cat1002310017&&_dyncharset=UTF-8&urlState=/women/specialty-sizing/petites/shop-brands/alf
red-dunner/10-petite/_/N-gkqzcxZ1z1407s/cat.jump', 'product_image_urls': 'http://s7d9.scene7.com/is/image/JCPenney/DP1228201517142050M.tif?he
i=380&amp;wid=380&op_usm=.4,.8,0,0&resmode=sharp2&op_usm=1.5,.8,0,0&resmode=sharp', 'brand': 'Alfred Dunner', 'total_number_reviews': 8, 'Rev
iews': [{'User': 'tpcu2211', 'Review': 'You never have to worry about the fit...Alfred Dunner clothing sizes are true to size and fits perfec
tly. Great value for the money.', 'Score': 1}, {'User': 'vutl2421', 'Review': 'Good quality fabric. Perfect fit. Washed very well no iron.',
'Score': 3}, {'User': 'ixlo1324', 'Review': 'I do not normally wear pants or capris that have an elastic waist, but I decided to try these si
nce they were on sale and I loved the color. I was very surprised at how comfortable they are and wear really well even wearing all day. I wi
ll buy this style again!', 'Score': 3}, {'User': 'dued2313', 'Review': 'I love these capris! They fit true to size and are so comfortable to
wear. I am planning to order more of them.', 'Score': 3}, {'User': 'nkmn4113', 'Review': 'This product is very comfortable and the fabric lau
nders very well', 'Score': 5}, {'User': 'tyfr4414', 'Review': 'I did not like the fabric. It is 100% polyester I thought it was different.I b
ought one at the store apprx two monts ago, and I thought it was just like it', 'Score': 2}, {'User': 'hyhy1222', 'Review': 'What a great dea
l. Beautiful Pants. Its more than I expected.', 'Score': 4}, {'User': 'oofb2342', 'Review': 'Alfred Dunner has great pants, good fit and very
comfortable', 'Score': 3}], 'Bought With': ['bc9ab3406dcaa84a123b9da862e6367d', '18eb69e8fc27e9c79b3209b672317f5b', '52c6e0ee4e9f1f2389510d46
d0405c1f', 'a610cd02e529a837b54e83cef5064146', 'a1cba5df721a1b35f2c706ddb6b889a0']}]
```

In [139…

```
# 2.1.2 Display the first 5 elements of reviewers_data

print(reviewers_data[:5])

# As shown, they are both semi-structured. For easy exploration, both datasets would be converted to a dataFrame via Pandas
```

```
[{'Username': 'bkpn1412', 'DOB': '31.07.1983', 'State': 'Oregon', 'Reviewed': ['cea76118f6a9110a893de2b7654319c0']}, {'Username': 'gqjs4414',
'DOB': '27.07.1998', 'State': 'Massachusetts', 'Reviewed': ['fa04fe6c0dd5189f54fe600838da43d3']}, {'Username': 'eehe1434', 'DOB': '08.08.195
0', 'State': 'Idaho', 'Reviewed': []}, {'Username': 'hkxj1334', 'DOB': '03.08.1969', 'State': 'Florida', 'Reviewed': ['f129b1803f447c2b1ce435
08fb822810', '3b0c9bc0be65a3461893488314236116']}, {'Username': 'jjbd1412', 'DOB': '26.07.2001', 'State': 'Georgia', 'Reviewed': []}]
```

In [140…

```
# 2.1.3 Convert both products_json and reviewers_json to pandas dataFrames

# Create dataframe objects from the loaded json data

products_json = pd.DataFrame(products_data)
reviewers_json = pd.DataFrame(reviewers_data)

# The code converts the semi-structured JSON data into a structured tabular data using pandas DataFrames
# Each row in these DataFrames represents a record
# And the columns correspond to the attributes or fields present in the data
# This structured representation makes it easier to analyze and manipulate the data using pandas functionalities
```

In [141…

```
# 2.1.4 Displaying the content of products_json

# Reveal the first few rows of the products_json DataFrame

display(products_json.head())

# The code displays the first 5 rows of the products_json DataFrame
# This is useful for quickly inspecting the structure & content of the DataFrame,
# including the column names and some sample data
# This helps to understand the dataset before performing further analysis or processing
```

	uniq_id	sku	name_title	description	list_price	sale_price	category	category_tree	average_product_rating	
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	alfred dunner	jcpenney|women|alfred dunner	2.625	http://www.jc
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	alfred dunner	jcpenney|women|alfred dunner	3.000	http://www.jc
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner	41.09	24.16	view all	jcpenney|women|view all	2.625	http://www.jc

	uniq_id	sku	name_title	description	list_price	sale_price	category	category_tree	average_product_rating	
				pull-on cap...						
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	3.500	http://www.jc
4	d969a8542122e1331e304b09f81a83f6	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	3.125	http://www.jc

In [142…

```
# 2.1.5 Displaying the content of reviewers_json

display(reviewers_json.head())

# Same as the cell above
```

	Username	DOB	State	Reviewed
0	bkpn1412	31.07.1983	Oregon	[cea76118f6a9110a893de2b7654319c0]
1	gqjs4414	27.07.1998	Massachusetts	[fa04fe6c0dd5189f54fe600838da43d3]
2	eehe1434	08.08.1950	Idaho	[]
3	hkxj1334	03.08.1969	Florida	[f129b1803f447c2b1ce43508fb822810, 3b0c9bc0be6...
4	jjbd1412	26.07.2001	Georgia	[]

In [143…

```
# 2.2 Exploration of the three CSV files

# Load csv files
# Use Pandas to read data from the CSV files and return a Dataframe
# Store the DataFrame created from the CSV files in the assigned variables
# Display the first few rows of the DataFrames created from that CSV files

products_csv = pd.read_csv('products.csv')
reviews_csv = pd.read_csv('reviews.csv')
```

```
users_csv = pd.read_csv('users.csv')

print("Displaying the content of products.csv file:")
display(products_csv.head())

print("Displaying the content of reviews.csv file:")
display(reviews_csv.head())

print("Displaying the content of users.csv file:")
display(users_csv.head())

# These line of codes aim to provide a quick overview of the contents of the three csv files
```

```
Displaying the content of products.csv file:
```

	Uniq_id	SKU	Name	Description	Price	Av_Score
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...	41.09	2.625
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...	41.09	3.000
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...	41.09	2.625
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...	41.09	3.500
4	d969a8542122e1331e304b09f81a83f6	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	Youll return to our Alfred Dunner pull-on capr...	41.09	3.125

```
Displaying the content of reviews.csv file:
```

	Uniq_id	Username	Score	Review
0	b6c0b6bea69c722939585baeac73c13d	fsdv4141	2	You never have to worry about the fit...Alfred...
1	b6c0b6bea69c722939585baeac73c13d	krpz1113	1	Good quality fabric. Perfect fit. Washed very ...
2	b6c0b6bea69c722939585baeac73c13d	mbmg3241	2	I do not normally wear pants or capris that ha...
3	b6c0b6bea69c722939585baeac73c13d	zeqg1222	0	I love these capris! They fit true to size and...
4	b6c0b6bea69c722939585baeac73c13d	nvfn3212	3	This product is very comfortable and the fabri...

```
Displaying the content of users.csv file:
```

	Username	DOB	State
0	bkpn1412	31.07.1983	Oregon
1	gqjs4414	27.07.1998	Massachusetts
2	eehe1434	08.08.1950	Idaho
3	hkxj1334	03.08.1969	Florida
4	jjbd1412	26.07.2001	Georgia

In [144…

```python
# 2.3 Shape of the JSON datasets

# Determine and print the dimensions of products_json and reviewers_json
# Assigns the tuple to a variable

products_json_shape = products_json.shape
print("Shape of the products_json DataFrame is:", products_json_shape)

reviewers_json_shape = reviewers_json.shape
print("Shape of the reviewers_json DataFrame is:", reviewers_json_shape)

# The code helps reveal the number of rows and columns each DataFrame has
# This is useful for understanding the size and structure of the datasets
```

```
Shape of the products_json DataFrame is: (7982, 15)
Shape of the reviewers_json DataFrame is: (5000, 4)
```

In [145…

```python
# 2.3.1 Shape of the CSV datasets

print("Shape of products_csv is:", products_csv.shape)
print("Shape of reviews_csv is:", reviews_csv.shape)
print("Shape of users_csv is:", users_csv.shape)

# Same as the cell above
# Printing the shapes of these dataframes gives an understanding of the number of rows and columns in each dataframe
# This provides insights into the size and structure of the data
```

```
Shape of products_csv is: (7982, 6)
Shape of reviews_csv is: (39063, 4)
Shape of users_csv is: (5000, 3)
```

In [146…

```python
# 2.4 General information about the products_json DataFrame

# Provide concise summary about the DataFrame

products_json.info()

# This code helps to quickly assess the data quality and to understand the characteristics of the DataFrame
# It identifies elements such as; missing values, data types, and an estimate of memory usage
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7982 entries, 0 to 7981
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
```

```
 0   uniq_id                 7982 non-null   object 
 1   sku                     7982 non-null   object 
 2   name_title              7982 non-null   object 
 3   description             7982 non-null   object 
 4   list_price              7982 non-null   object 
 5   sale_price              7982 non-null   object 
 6   category                7982 non-null   object 
 7   category_tree           7982 non-null   object 
 8   average_product_rating  7982 non-null   float64
 9   product_url             7982 non-null   object 
 10  product_image_urls      7982 non-null   object 
 11  brand                   7982 non-null   object 
 12  total_number_reviews    7982 non-null   int64  
 13  Reviews                 7982 non-null   object 
 14  Bought With             7982 non-null   object 
dtypes: float64(1), int64(1), object(13)
memory usage: 935.5+ KB
```

In [147…

```
# 2.4.1 General information about the reviewers_json DataFrame

reviewers_json.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Username  5000 non-null   object
 1   DOB       5000 non-null   object
 2   State     5000 non-null   object
 3   Reviewed  5000 non-null   object
dtypes: object(4)
memory usage: 156.4+ KB
```

In [148…

```
# 2.4.2 General information about products_csv Dataframe

products_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7982 entries, 0 to 7981
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Uniq_id      7982 non-null   object
 1   SKU          7915 non-null   object
 2   Name         7982 non-null   object
 3   Description  7439 non-null   object
 4   Price         5816 non-null   float64
 5   Av_Score      7982 non-null   float64
dtypes: float64(2), object(4)
memory usage: 374.3+ KB
```

In [149…

```
# 2.4.3 General information about reviews_csv Dataframe

reviews_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39063 entries, 0 to 39062
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Uniq_id   39063 non-null  object
 1   Username  39063 non-null  object
 2   Score     39063 non-null  int64 
 3   Review    39063 non-null  object
dtypes: int64(1), object(3)
memory usage: 1.2+ MB
```

In [150…

```
# 2.4.4 General information about users_csv Dataframe

users_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Username  5000 non-null   object
 1   DOB       5000 non-null   object
 2   State     5000 non-null   object
dtypes: object(3)
memory usage: 117.3+ KB
```

2.5 Data Understanding: Observations from the General Information of the 5 Datasets

Based on the data exploration of the 5 datasets by looking into their shapes, and other general information:

(a) products_json & products_csv have the same entries as seen in 2.4 and 2.4.2. They are identical, but the former has more variables. Because of this, we shall adopt products_json for further analysis henceforth.

(b) reviewers.json contain variables that can be found in reviews_csv & users_csv. As seen in 2.1.5, 'Reviewed' column contains alpha-numeric strings that doesn't offer much for our analysis. Because of this, reviewers_json shall be dropped.

(c) As seen in 2.4.3 & 2.4.4 respectively, users_csv and reviews_csv both have 'username' as a common variable. Because of this, both files shall be merged through an outer join.

(d) Looking at products_csv in 2.4.2, it reveals an entry of 7982 for all variables. The 'price' column however has an entry of 5816. This suggests there are missing values in the price column. Furthermore, when this is compared with the identical products_json, it reveals that the 'list_price' doesn't have any missing value. This is a problem, and it would be resolved by changing the data type of 'list_price' (sales_price inclusive) from an object to a float. The empty strings would also be replaced with NaN missing values for python to know that the column is numeric.

Summarily, we will be working with just 2 dataframes as follows;

(1) products_json

(2) A merged file of both: users_csv and reviews_csv

In [151…

```
# 2.6 Check for Empty Strings

# Assigning products_json to a new variable
# Create a new reference to the same dataframe
# Checking for the presence of empty strings in 'list_price'
# Checking for the presence of empty strings in 'sale_price'

jcpenney_products = products_json

empty_strings = jcpenney_products['list_price'].str.strip() == ''
empty_string_count = empty_strings.sum()
print(f"Number of empty strings in 'list_price' is: {empty_string_count}")

empty_strings1 = jcpenney_products['sale_price'].str.strip() == ''
empty_string_count = empty_strings1.sum()
print(f"Number of empty strings in 'sale_price' is: {empty_string_count}")

# The codes are performing data quality checks by counting the number of empty strings in 'list_price' and 'sale_price' columns
# This helps identify and handle missing or improperly formatted data in these columns
```

```
Number of empty strings in 'list_price' is: 2166
Number of empty strings in 'sale_price' is: 18
```

In [152…

```
# 2.6.1 Check for Non-convertible values or a range of values

# Identify non-convertible values
# Print count of non-convertible values in 'sale_price'
# Print count of non-convertible values in 'list_price'

def print_non_convertible_counts(column_name):
    non_convertible_counts = pd.to_numeric(jcpenney_products[column_name], errors='coerce').isnull().sum()

    print(f"Count of non-convertible values in '{column_name}': {non_convertible_counts}")

print_non_convertible_counts('sale_price')
print_non_convertible_counts('list_price')

# The code prints the count of non-convertible values in a specified column: 'sale_price' and'list_price
# Helpful for understanding the data quality in numeric columns
# Interestingly, sale_price contains other non-convertible values aside the empty strings revealed in previous cell
```

```
Count of non-convertible values in 'sale_price': 263
Count of non-convertible values in 'list_price': 2166
```

In [153…

```
# 2.6.2 Display non-convertible values in sale_price

# Extract non-convertible values
# Prints a sample of those values

def print_non_convertible_samples(column_name, sample_size=5):
    non_convertible_values = jcpenney_products.loc[pd.to_numeric(jcpenney_products[column_name], errors='coerce').isnull(), column_name]
    unique_non_convertible_values = non_convertible_values.unique()

    print(f"Sample of non-convertible values in '{column_name}':")
    print(unique_non_convertible_values[:sample_size])

# Call the function with the desired column name and sample size
print_non_convertible_samples('sale_price', sample_size=5)
```

```
Sample of non-convertible values in 'sale_price':
['12.07-60.42' '30.20-181.28' '34.10-227.40' '30.20-48.33' '14.49-15.70']
```

In [129…

```
# 2.6.3 Replacing the empty strings and non-convertible values

# Replace empty strings in 'list-price' with missing values - NaN
# Replace empty strings in 'sale_price' with missing values - NaN
# Coerce any values in sale_price that cannot be converted to numeric to NaN

jcpenney_products['list_price'] = jcpenney_products['list_price'].replace('', np.nan)
jcpenney_products['sale_price'] = jcpenney_products['sale_price'].replace('', np.nan)

jcpenney_products['sale_price'] = pd.to_numeric(jcpenney_products['sale_price'], errors='coerce')

# These steps are part of data preparation and cleaning to ensure the data is suitable for analysis and modeling.
```

In [130…

```python
# 2.6.4 Convert data type to a float

# Convert 'list_price' and 'sale_price' columns to float
# # Print the updated dataframe information

jcpenney_products['list_price'] = jcpenney_products['list_price'].astype(float)
jcpenney_products['sale_price'] = jcpenney_products['sale_price'].astype(float)

print(jcpenney_products.info())

# This is done to ensure that these columns are represented as numeric values
# The info() printout allows us to verify the changes in data types and check for any missing values in the DataFrame
# As against 2.4, the data type of list_price and sale_price has now been changed to a float
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7982 entries, 0 to 7981
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   uniq_id                 7982 non-null   object 
 1   sku                     7982 non-null   object 
 2   name_title              7982 non-null   object 
 3   description             7982 non-null   object 
 4   list_price              5816 non-null   float64
 5   sale_price              7719 non-null   float64
 6   category                7982 non-null   object 
 7   category_tree           7982 non-null   object 
 8   average_product_rating  7982 non-null   float64
 9   product_url             7982 non-null   object 
 10  product_image_urls      7982 non-null   object 
 11  brand                   7982 non-null   object 
 12  total_number_reviews    7982 non-null   int64  
 13  Reviews                 7982 non-null   object 
 14  Bought With             7982 non-null   object 
dtypes: float64(3), int64(1), object(11)
memory usage: 935.5+ KB
None
```

In [131…

```python
# 2.7 Merge users_csv and reviews_csv

# Assign merged dataset to a new variable
# Perform an outer join on the 'Username' column
# Display the shape of the dataset

jcpenney_reviews = pd.merge(reviews_csv, users_csv, on='Username', how='outer')
display(jcpenney_reviews.head())
print("Shape of jcpenney_reviews is:", jcpenney_reviews.shape)
```

```
# The code merges reviews_csv and users_csv based on the 'Username' column using an outer join
# The new DataFrame; jcpenney_reviews, contains information from both DataFrames
# The output displays the first few rows and the shape of the merged DataFrame
```

	Uniq_id	Username	Score	Review	DOB	State
0	b6c0b6bea69c722939585baeac73c13d	fsdv4141	2.0	You never have to worry about the fit...Alfred...	31.07.1980	American Samoa
1	cbe8d131628ec67e803c47d3dd6f2529	fsdv4141	2.0	These are great shirts, looks great all day. W...	31.07.1980	American Samoa
2	5ea5f53bbb750106865a044634404dd7	fsdv4141	5.0	I purchase three rugs to replace twenty-year-o...	31.07.1980	American Samoa
3	0144d2094668b42ae7c674915806f5f3	fsdv4141	1.0	I am a huge user of BE original fromula. I hap...	31.07.1980	American Samoa
4	99141a2b164cf257c96bcb4593915b50	fsdv4141	1.0	Very soft and stretchy! They arent as dressy a...	31.07.1980	American Samoa

```
Shape of jcpenney_reviews is: (39086, 6)
```

In [132…

```
# 2.8 Numerical Summary Statistics of the jcpenney_products and jcpenney_reviews Datasets

# Having changed the data type of the 'list_price' column of jcpenny_products to a float
# We can now perform a statistical summary of both datasets
# print a horizontal line to separate content in the output

display(jcpenney_products.describe())

print("\n" + "="*80 + "\n")

display(jcpenney_reviews.describe())

# This code provides a brief numerical summary of both Dataframes
# To display the summary of other non-numerical columns, the function 'include=all' would be required
```

	list_price	sale_price	average_product_rating	total_number_reviews
count	5816.000000	7719.000000	7982.000000	7982.000000
mean	144.776618	101.605623	2.988683	4.893886
std	499.223719	360.965606	0.911673	3.314284
min	-65.270000	3.610000	1.000000	1.000000
25%	40.700000	22.950000	2.500000	2.000000
50%	58.010000	35.460000	3.000000	4.000000
75%	87.020000	60.420000	3.500000	8.000000

	list_price	sale_price	average_product_rating	total_number_reviews
max	17122.170000	10273.300000	5.000000	23.000000

```
================================================================================
```

	Score
count	39080.000000
mean	1.487769
std	1.400414
min	0.000000
25%	0.000000
50%	1.000000
75%	2.000000
max	5.000000

2.9 Interpretation of summary statistics

As seen in 2.8, the minimum price of list_price is a negative value. Unfortunately, price can't be a negative number. This is probably an error. Also, the maximum price of the list_price and sale_price is very high, indicating a high degree of variability in both columns. This suggests the presence of outliers in both columns.

3.0 DATA VALIDATION

The issues mentioned in 2.9 above would be handled in this section, amongst others.

In [27]:

```
# 3.1 Identify missing values in jcpenney_products and jcpenney_reviews

# List out the number of missing values in the columns of each dataset
# isnull() method checks for missing values in each cell of the DataFrames

jcpenney_products_missing_values = jcpenney_products.isnull().sum()
jcpenney_reviews_missing_values = jcpenney_reviews.isnull().sum()

print("Number of missing values in each column of jcpenney_products are as follows:")
display(jcpenney_products_missing_values)

print("\n" + "="*80 + "\n")

print("Number of missing values in each column of jcpenney_reviews are as follows:")
```

```
display(jcpenney_reviews_missing_values)

# The code provides a summary of missing values in the columns of the jcpenney_products DataFrame
# It returns a DataFrame of the same shape with True where a value is missing and False otherwise
# The sum() method is then applied, it sums the True values along each column, giving the total count of missing values
for each
# This makes it easy to identify columns with missing data and assess the extent of missingness in the dataset
```

```
Number of missing values in each column of jcpenney_products are as follows:
uniq_id                      0
sku                          0
name_title                   0
description                  0
list_price                2166
sale_price                 263
category                     0
category_tree                0
average_product_rating       0
product_url                  0
product_image_urls           0
brand                        0
total_number_reviews         0
Reviews                      0
Bought With                  0
dtype: int64
================================================================================

Number of missing values in each column of jcpenney_reviews are as follows:
Uniq_id     6
Username    0
Score       6
Review      6
DOB         0
State       0
dtype: int64
```

In [28]:

```
# 3.1.1 Handle the missing values of jcpenney_products 'list_price' and 'sale_price' via imputation method

# use replace() method to remove any non-numeric and non-decimal characters from the 'list_price' column
# fills any remaining missing values in the 'list_price' & 'sale_price' with the mean of the column using the fillna()
method

jcpenney_products['list_price'] = jcpenney_products['list_price'].replace('[^\d.]', '', regex=True)
# Fill missing values with the mean
jcpenney_products['list_price'] = jcpenney_products['list_price'].fillna(jcpenney_products['list_price'].mean())
jcpenney_products.isnull().sum()

jcpenney_products['sale_price'] = jcpenney_products['sale_price'].replace('[^\d.]', '', regex=True)
```

```
jcpenney_products['sale_price'] = jcpenney_products['sale_price'].fillna(jcpenney_products['sale_price'].mean())
jcpenney_products.isnull().sum()

# The code aims to clean the columns by removing non-numeric characters
# And then imputing any missing values with the mean of the respective columns
```

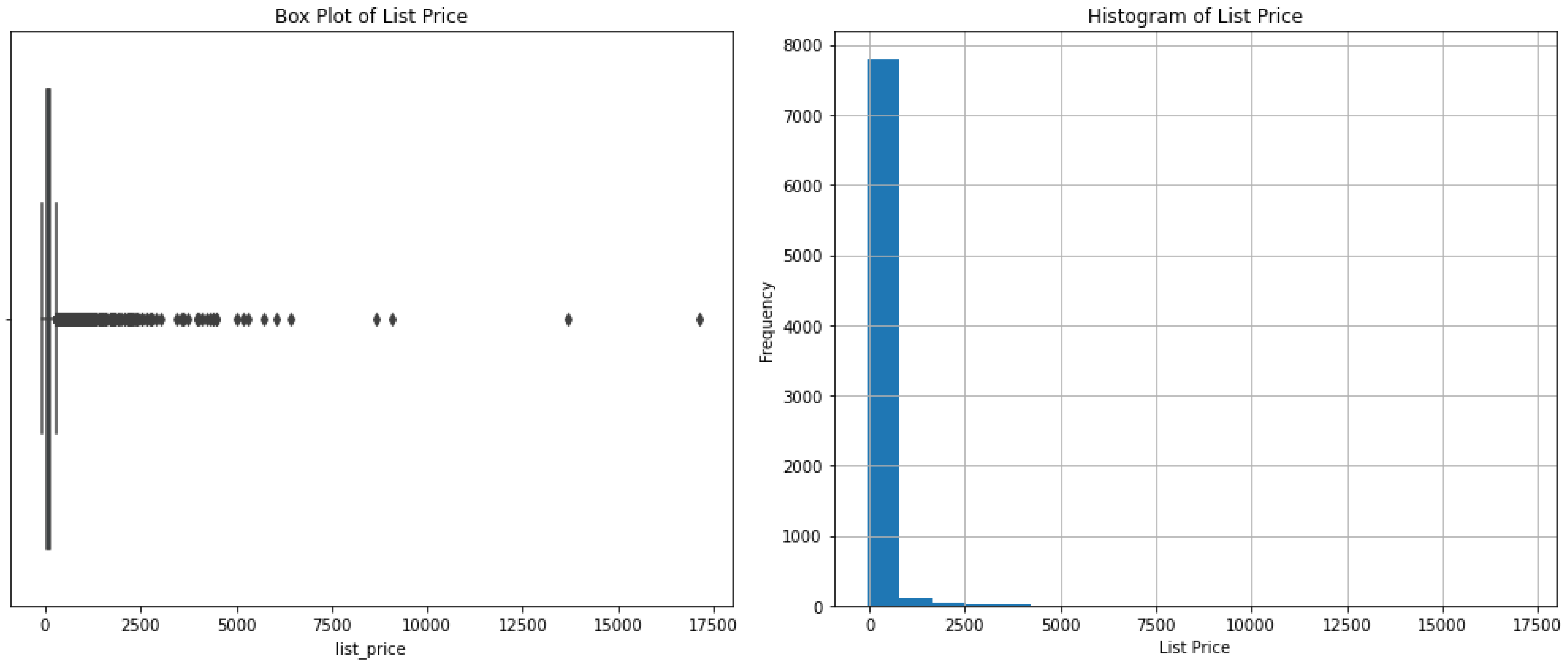
Out[28]:
```
uniq_id                  0
sku                      0
name_title               0
description              0
list_price               0
sale_price               0
category                 0
category_tree            0
average_product_rating   0
product_url              0
product_image_urls       0
brand                    0
total_number_reviews     0
Reviews                  0
Bought With              0
dtype: int64
```

In [29]:
```
# 3.1.2 Handling the missing values of jcpenney_reviews by using drop.na

# drop.na is used because the missing values aren't significant, as revealed in the downward part of 3.1
# remove rows containing any missing values
# The inplace=True parameter is set to modify the DataFrame in place

jcpenney_reviews.dropna(inplace=True)

# Display the modified DataFrame
jcpenney_reviews.isnull().sum()

# The code removes rows and then checks to confirm that there are no more missing values by showing the count of missing
values
# As shown in the output, the missing values have been dropped
```

Out[29]:
```
Uniq_id     0
Username    0
Score       0
Review      0
DOB         0
State       0
dtype: int64
```

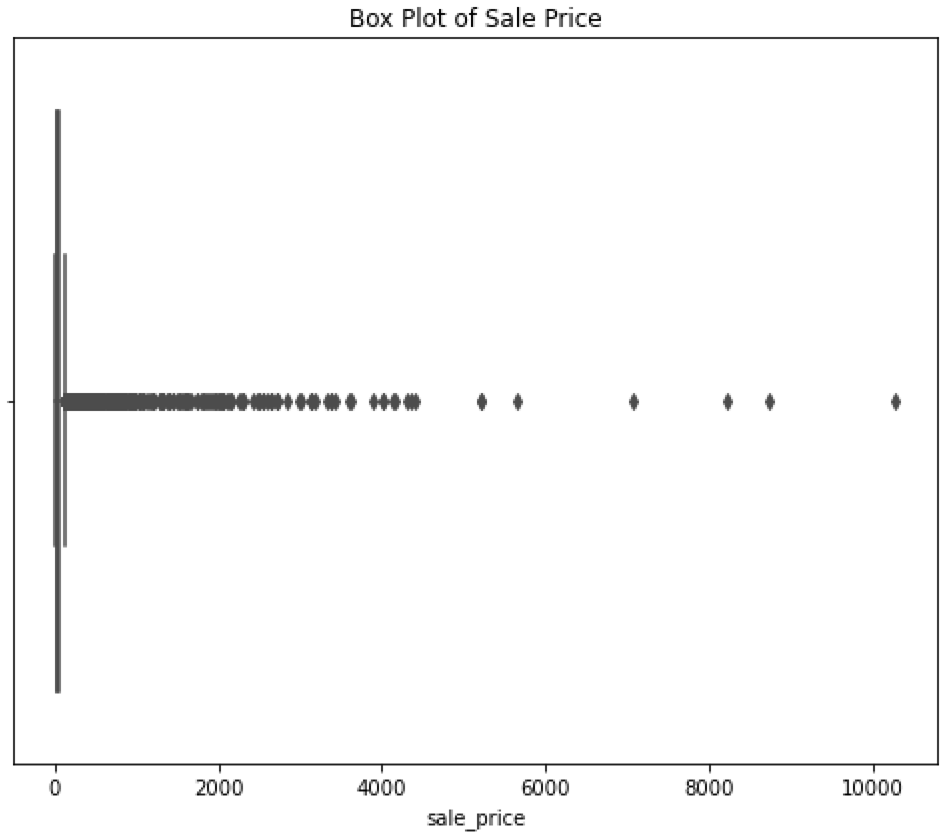
In [30]:
```
# 3.2 Identify outliers in 'list_price' via box plot and histogram
```

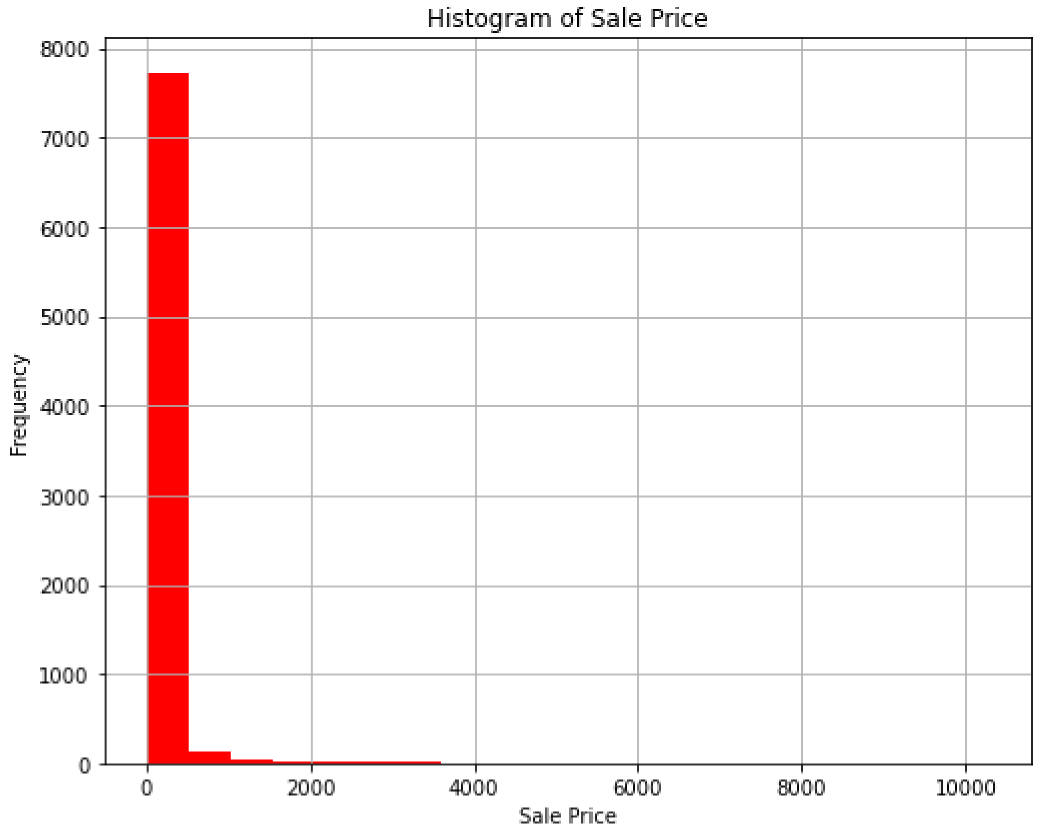
```python
# Generate a side-by-side visualization of the distribution of 'list_price'
# Displays the Matplotlib figure

# Set up the figure with a 1x2 grid
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Box plot on the left
sns.boxplot(x='list_price', data=jcpenney_products, ax=axes[0])
axes[0].set_title('Box Plot of List Price')

# Histogram on the right
jcpenney_products['list_price'].hist(bins=20, ax=axes[1])
axes[1].set_title('Histogram of List Price')
axes[1].set_xlabel('List Price')
axes[1].set_ylabel('Frequency')

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()

# The code provides a visual representation of the distribution for better insights into the data's statistical
characteristics
# The plots show the presence of outliers in both distribution
# As seen by the stand-alone dots and tall bar of the box plot and histogram respectively
```



In [31]:
```
# 3.2.1 Identify outliers in 'sale_price' via box plot or histogram

# Set up the figure with a 1x2 grid
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Box plot on the left
sns.boxplot(x='sale_price', data=jcpenney_products, ax=axes[0], color='red')
axes[0].set_title('Box Plot of Sale Price')

# Histogram on the right
jcpenney_products['sale_price'].hist(bins=20, ax=axes[1], color='red')
axes[1].set_title('Histogram of Sale Price')
axes[1].set_xlabel('Sale Price')
axes[1].set_ylabel('Frequency')

# Adjust layout to prevent overlap
plt.tight_layout()

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

```
# Just as in 3.2, both distributions in 3.2.1 also reveal the presence of outliers in the columns
```





In [32]:
```
# 3.2.2 Handling the Outliers through Inter-Quartile Range (IQR) Method

# Calculate the IQR for list_price in jcpenney_products
Q1_list_price = jcpenney_products['list_price'].quantile(0.25)
Q3_list_price = jcpenney_products['list_price'].quantile(0.75)
IQR_list_price = Q3_list_price - Q1_list_price

# Define the lower and upper bounds for outliers removal in list_price
lower_bound_list_price = Q1_list_price - 1.5 * IQR_list_price
upper_bound_list_price = Q3_list_price + 1.5 * IQR_list_price

# Calculate the IQR for sale_price in jcpenney_products
Q1_sale_price = jcpenney_products['sale_price'].quantile(0.25)
Q3_sale_price = jcpenney_products['sale_price'].quantile(0.75)
IQR_sale_price = Q3_sale_price - Q1_sale_price

# Define the lower and upper bounds for outliers removal in sale_price
lower_bound_sale_price = Q1_sale_price - 1.5 * IQR_sale_price
```

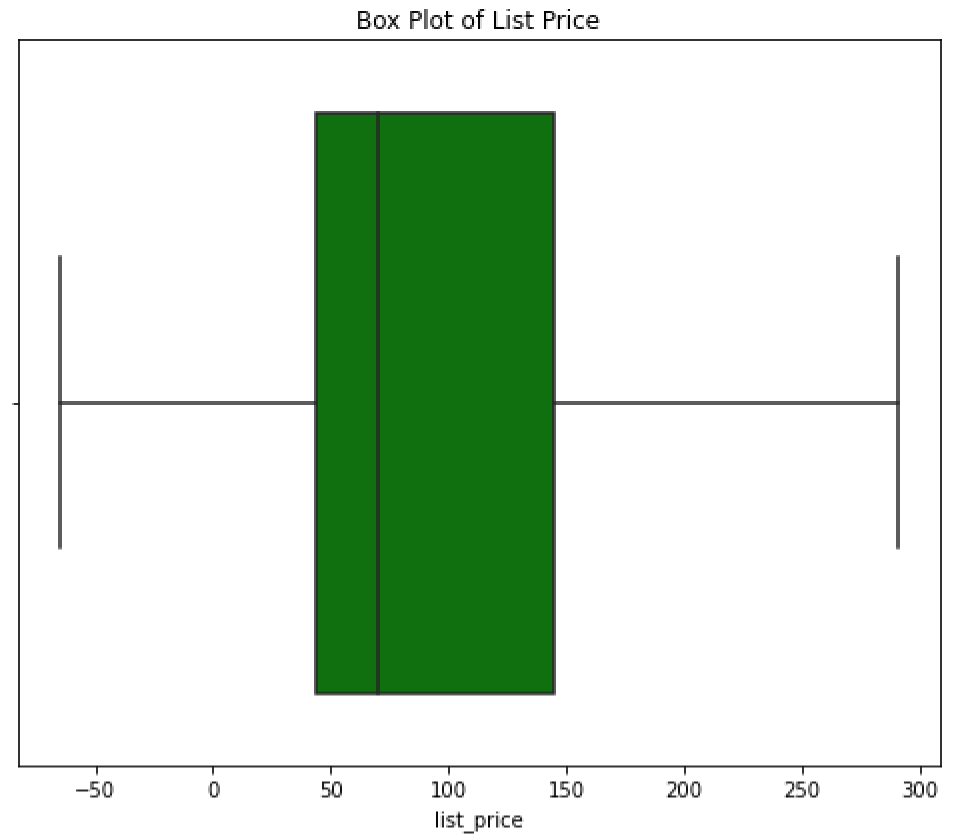
```
upper_bound_sale_price = Q3_sale_price + 1.5 * IQR_sale_price

# Remove outliers from list_price and sale_price in jcpenney_products
new_jcpenney_products = jcpenney_products[
    (jcpenney_products['list_price'] >= lower_bound_list_price) & (jcpenney_products['list_price'] <=
                                                                   upper_bound_list_price) &
    (jcpenney_products['sale_price'] >= lower_bound_sale_price) & (jcpenney_products['sale_price'] <=
                                                                   upper_bound_sale_price)
]

# Display information about outliers
print(f"List Price - Q1: {Q1_list_price}, Q3: {Q3_list_price}, IQR: {IQR_list_price}")
print(f"List Price - Lower Bound: {lower_bound_list_price}, Upper Bound: {upper_bound_list_price}")

print(f"\nSale Price - Q1: {Q1_sale_price}, Q3: {Q3_sale_price}, IQR: {IQR_sale_price}")
print(f"Sale Price - Lower Bound: {lower_bound_sale_price}, Upper Bound: {upper_bound_sale_price}")

print(f"\nNumber of Outliers removed: {len(jcpenney_products) - len(new_jcpenney_products)}")

# The code aims to filter out data points that are considered outliers based on the distribution of 'list_price' &
'sale_price'
# The lower and upper bounds are calculated as a range within which most of the data is expected to lie
# Values outside this range are considered outliers
# We have got a new dataframe called 'new_jcpenney_products'
```

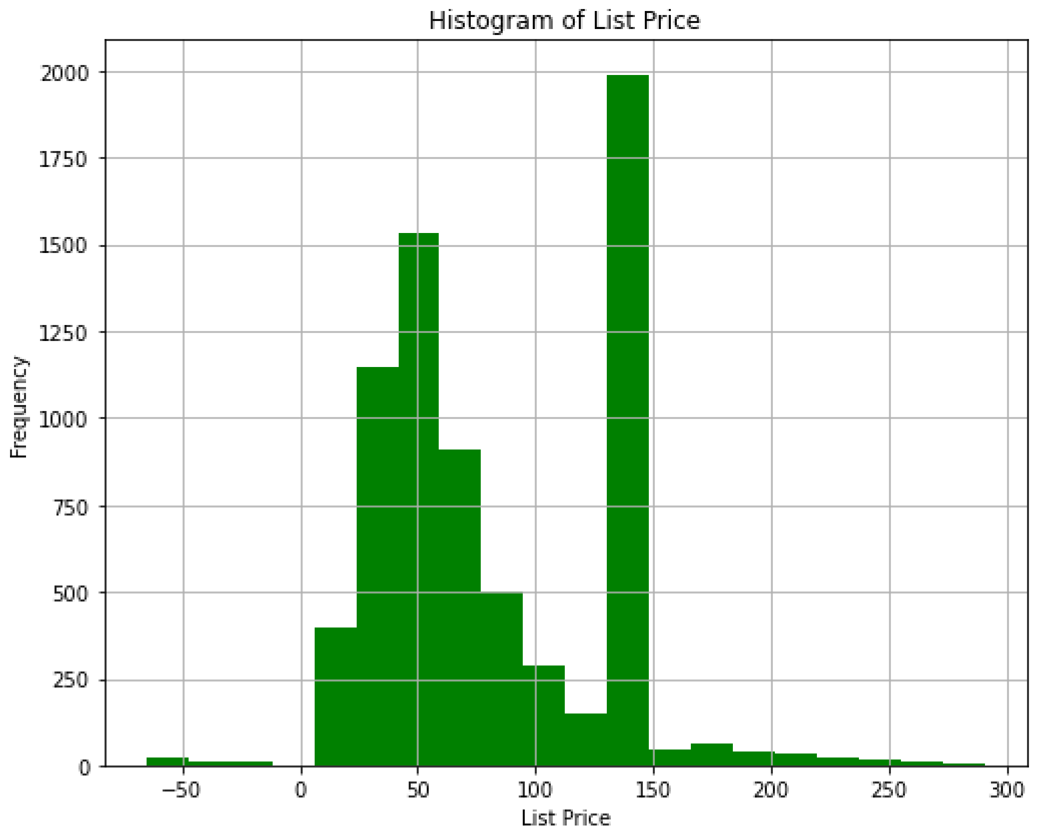
```
List Price - Q1: 47.3, Q3: 144.7766179504829, IQR: 97.47661795048289
List Price - Lower Bound: -98.91492692572434, Upper Bound: 290.99154487620723

Sale Price - Q1: 23.262500000000003, Q3: 66.4680725, IQR: 43.2055725
Sale Price - Lower Bound: -41.545858749999994, Upper Bound: 131.27643125

Number of Outliers removed: 796
```

In [33]:

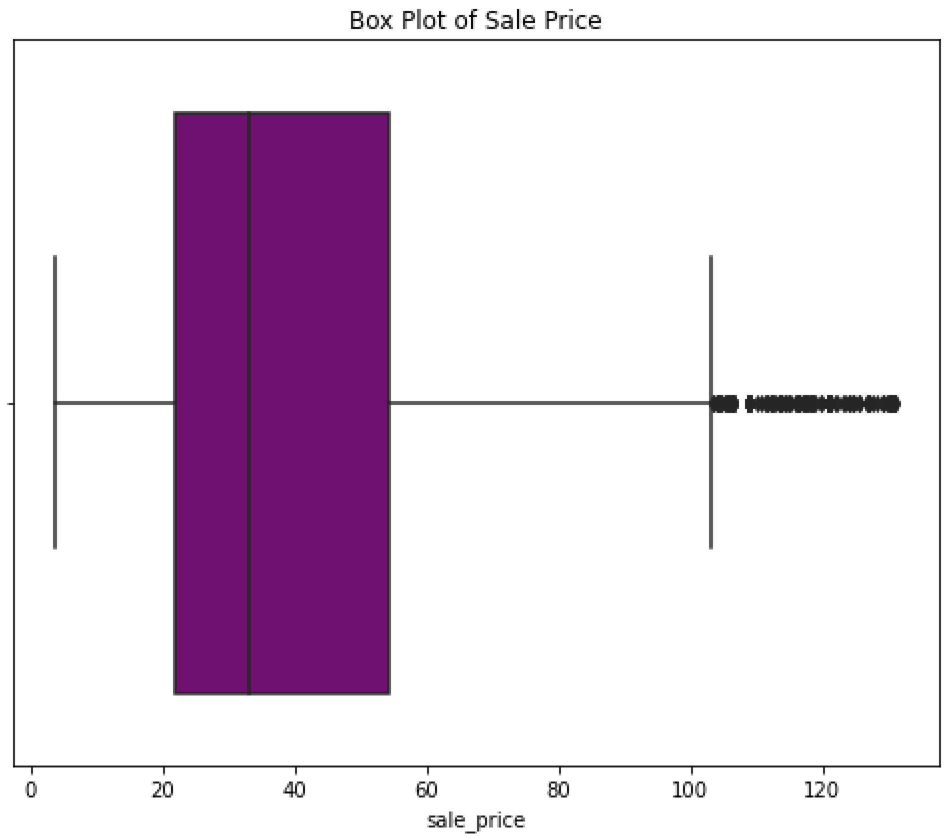
```
# 3.2.3 Check if outliers have been handled in both 'list_price' and 'sale_price'

# Make use of the new dataset 'new_jcpenney_products' created from 3.2.2 above
# Create histogram and box plot to see if there are changes

# Set up the figure with a 1x2 grid
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Box plot on the left
sns.boxplot(x='list_price', data=new_jcpenney_products, ax=axes[0], color='green')
axes[0].set_title('Box Plot of List Price')

# Histogram on the right
new_jcpenney_products['list_price'].hist(bins=20, ax=axes[1], color='green')
axes[1].set_title('Histogram of List Price')
```

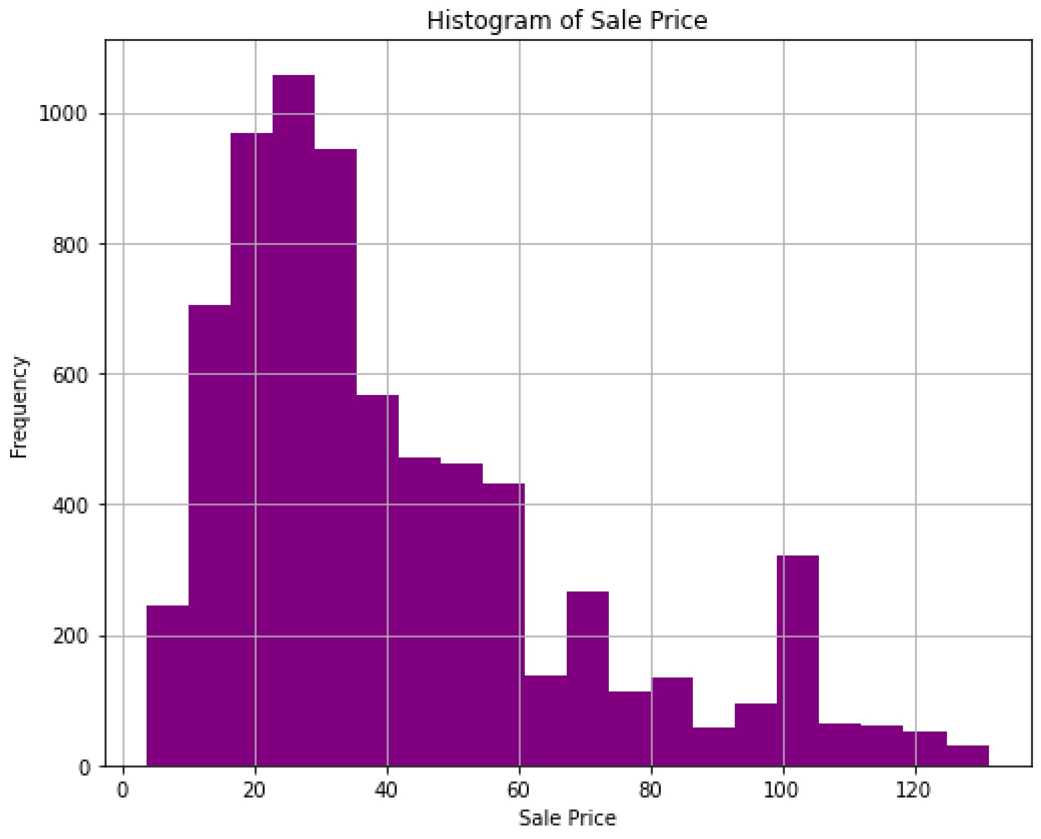
```python
axes[1].set_xlabel('List Price')
axes[1].set_ylabel('Frequency')

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()


print("\n" + "="*125 + "\n")


# Identify outliers in 'sale_price' via box plot or histogram
# Set up the figure with a 1x2 grid
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))

# Box plot on the left
sns.boxplot(x='sale_price', data=new_jcpenney_products, ax=axes[0], color='purple')
axes[0].set_title('Box Plot of Sale Price')

# Histogram on the right
new_jcpenney_products['sale_price'].hist(bins=20, ax=axes[1], color='purple')
axes[1].set_title('Histogram of Sale Price')
axes[1].set_xlabel('Sale Price')
axes[1].set_ylabel('Frequency')

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()

# The output of the code, as revealed in the plots below, shows that the outliers have been handled in 'list_price'
# As for the box plot of 'sale_price', the outliers are beyond the whiskers, and this means the distribution is okay
# This is regarded as acceptable outliers, and it is allowed
# Interestingly, the Histogram of sale_price shows an abscence of outliers in the distribution
```


Box Plot of List Price
−50
0
50
100
150
200
250
300
list_price


Histogram of List Price
Frequency
0
250
500
750
1000
1250
1500
1750
2000
−50
0
50
100
150
200
250
300
List Price

==



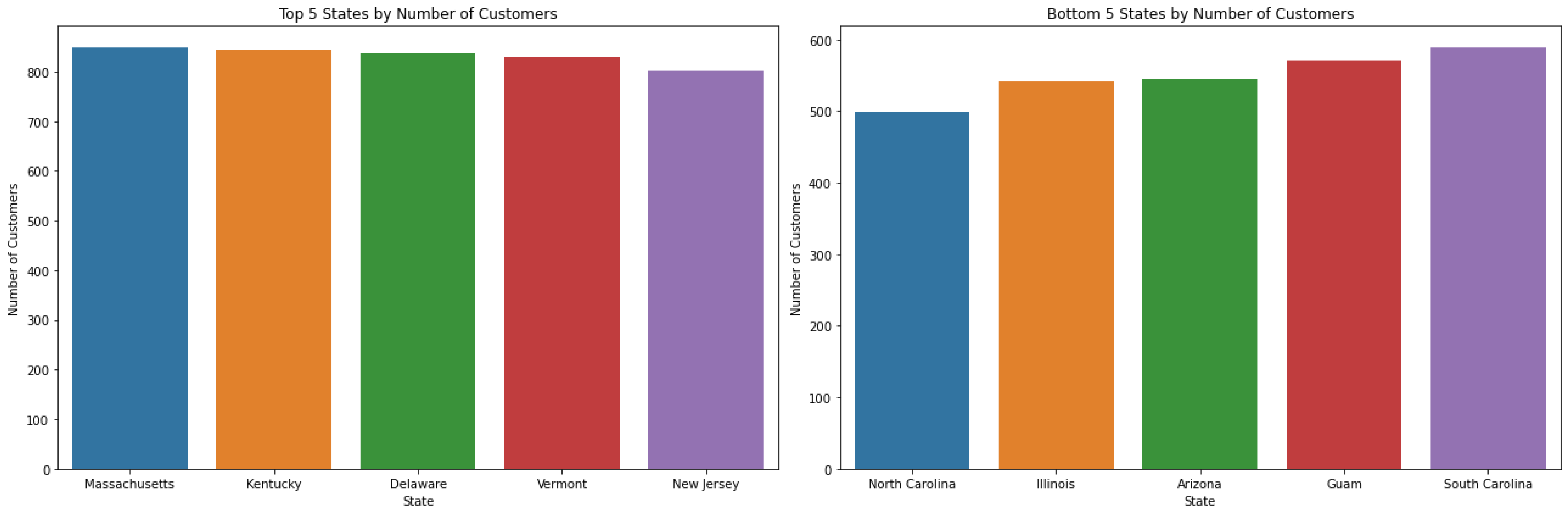


In [34]:
```
# 3.3 Remove irrelevant columns from new_jcpenney_products & jcpenney_reviews

# Use the drop method

new_jcpenney_products = new_jcpenney_products.drop(['sku', 'description', 'category_tree', 'product_url',
                                                     'product_image_urls'], axis=1)
jcpenney_reviews = jcpenney_reviews.drop(['Username'], axis=1)

# The code excludes unnecessary or redundant columns from the DataFrames
# This simplifies the data structure and potentially improves the efficiency of subsequent analyses
```

4.0 DATA VISUALIZATION

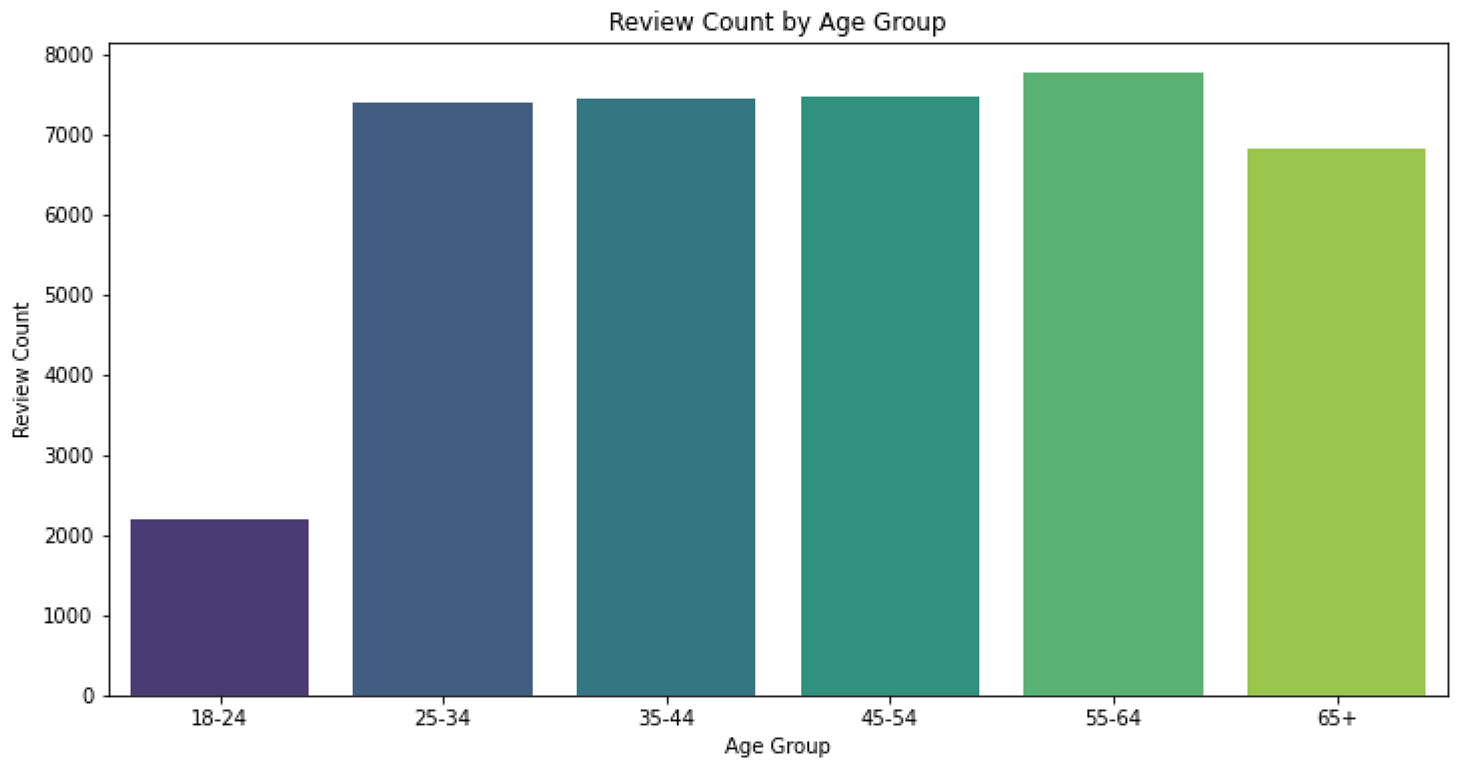
In [39]:
```
# 4.1 What are the top 5 states and bottom 5 states that buys JC Penney products?

# Get the top 5 and bottom 5 states
top_states = jcpenney_reviews['State'].value_counts().nlargest(5).index
bottom_states = jcpenney_reviews['State'].value_counts().nsmallest(5).index
```

```
# Filter the DataFrame for the top 5 and bottom 5 states
selected_states_df = jcpenney_reviews[jcpenney_reviews['State'].isin(top_states.union(bottom_states))]

# Create a side-by-side count plot for the top 5 and bottom 5 states
plt.figure(figsize=(18, 6))
plt.subplot(1, 2, 1)
sns.countplot(x='State', data=selected_states_df[selected_states_df['State'].isin(top_states)], order=top_states)
plt.title('Top 5 States by Number of Customers')
plt.xlabel('State')
plt.ylabel('Number of Customers')

plt.subplot(1, 2, 2)
sns.countplot(x='State', data=selected_states_df[selected_states_df['State'].isin(bottom_states)], order=bottom_states)
plt.title('Bottom 5 States by Number of Customers')
plt.xlabel('State')
plt.ylabel('Number of Customers')

plt.tight_layout()
plt.show()
```



In [82]:
```
# 4.2 Are there any age groups that are more active in providing reviews?

# Create age groups
age_bins = [18, 25, 35, 45, 55, 65, 100]
age_labels = ['18-24', '25-34', '35-44', '45-54', '55-64', '65+']
jcpenney_reviews['Age Group'] = pd.cut(jcpenney_reviews['Age'], bins=age_bins, labels=age_labels, right=False)
```

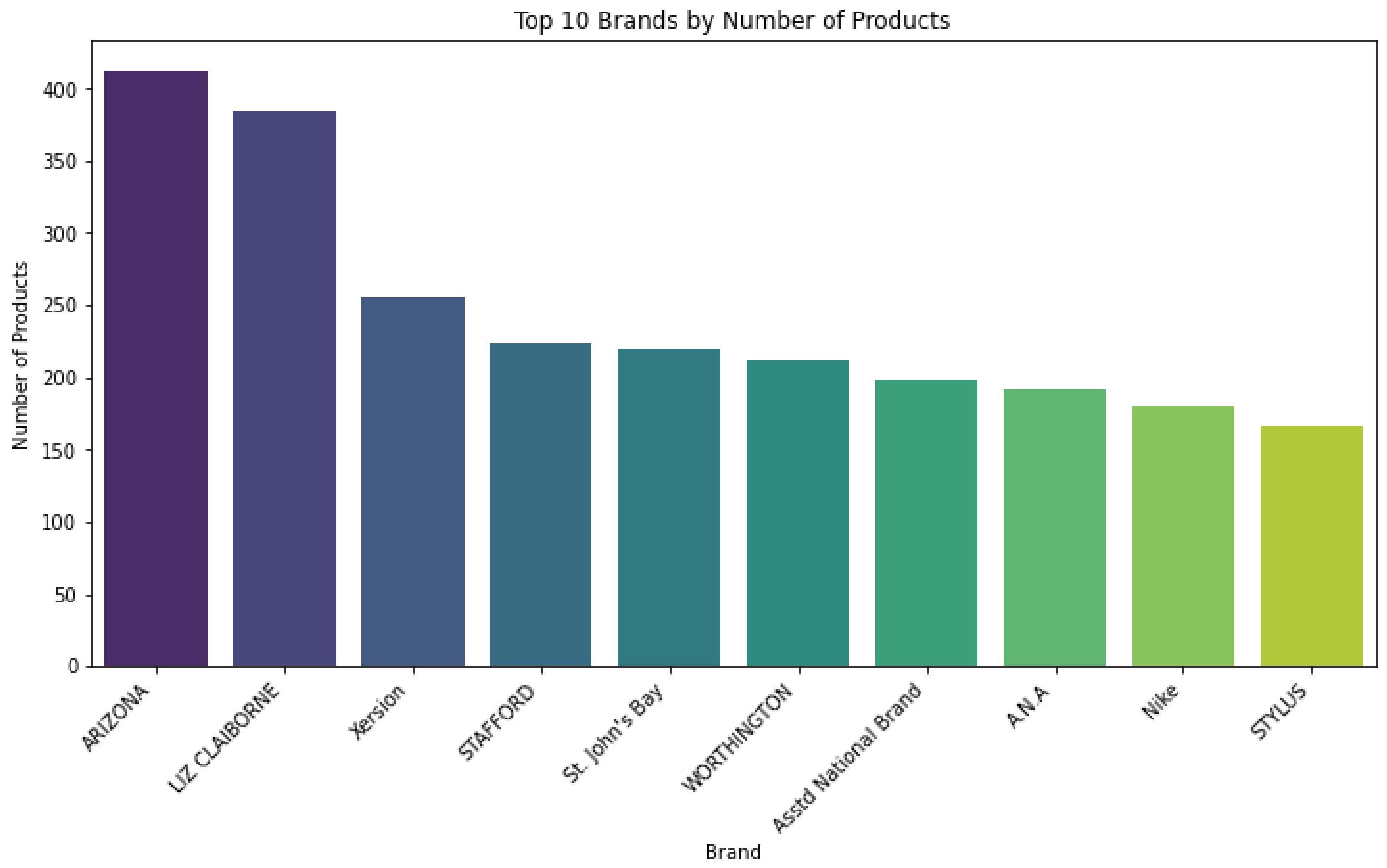
```
# Create a bar plot for review count by age group
plt.figure(figsize=(12, 6))
sns.countplot(x='Age Group', data=jcpenney_reviews, order=age_labels, palette='viridis')
plt.title('Review Count by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Review Count')
plt.show()
```



In [83]:

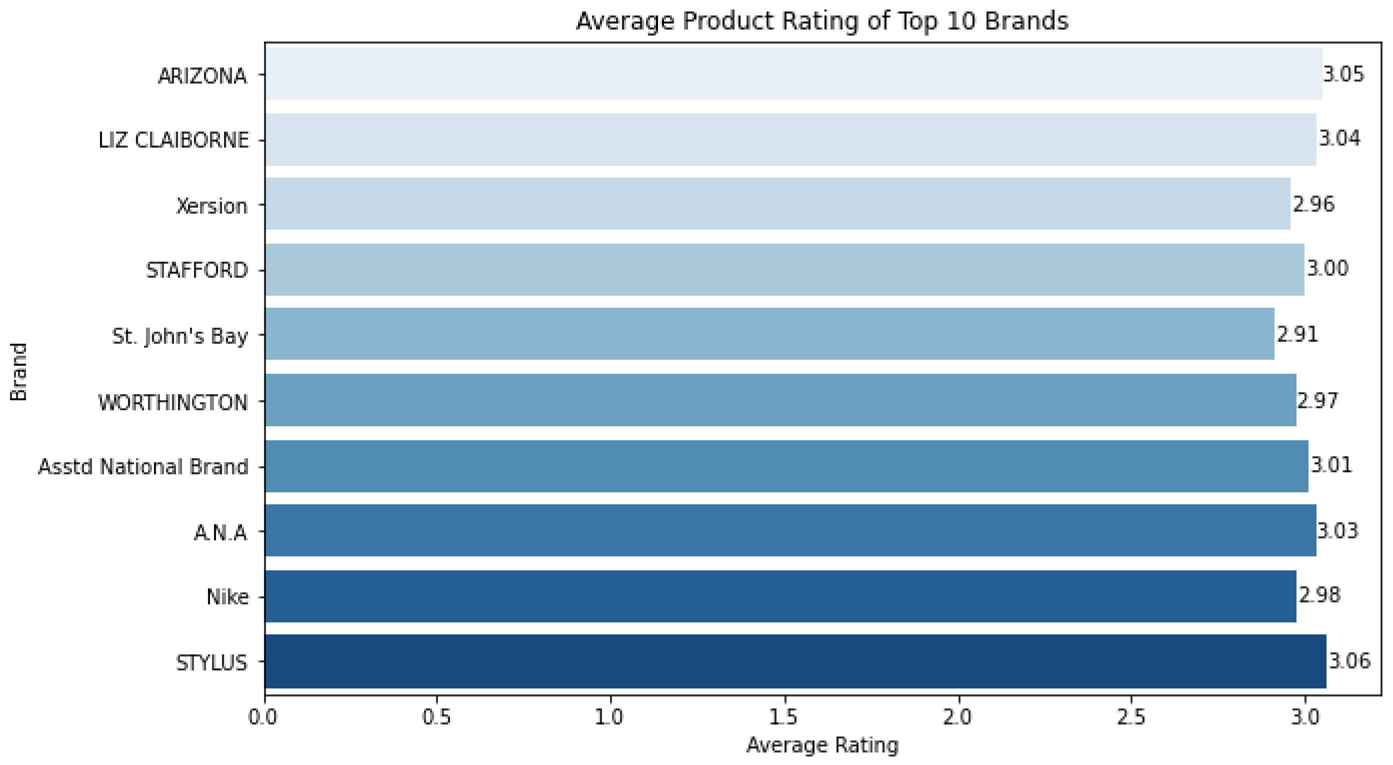
```
# 4.3 What are the top brands based on the number of products?

# Get the top brands based on the number of products
top_brands = new_jcpenney_products['brand'].value_counts().head(10)

# Set up the figure
plt.figure(figsize=(12, 6))

# Bar plot for top brands
sns.barplot(x=top_brands.index, y=top_brands.values, palette='viridis')
plt.title('Top 10 Brands by Number of Products')
plt.xlabel('Brand')
plt.ylabel('Number of Products')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
```

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



In [84]:

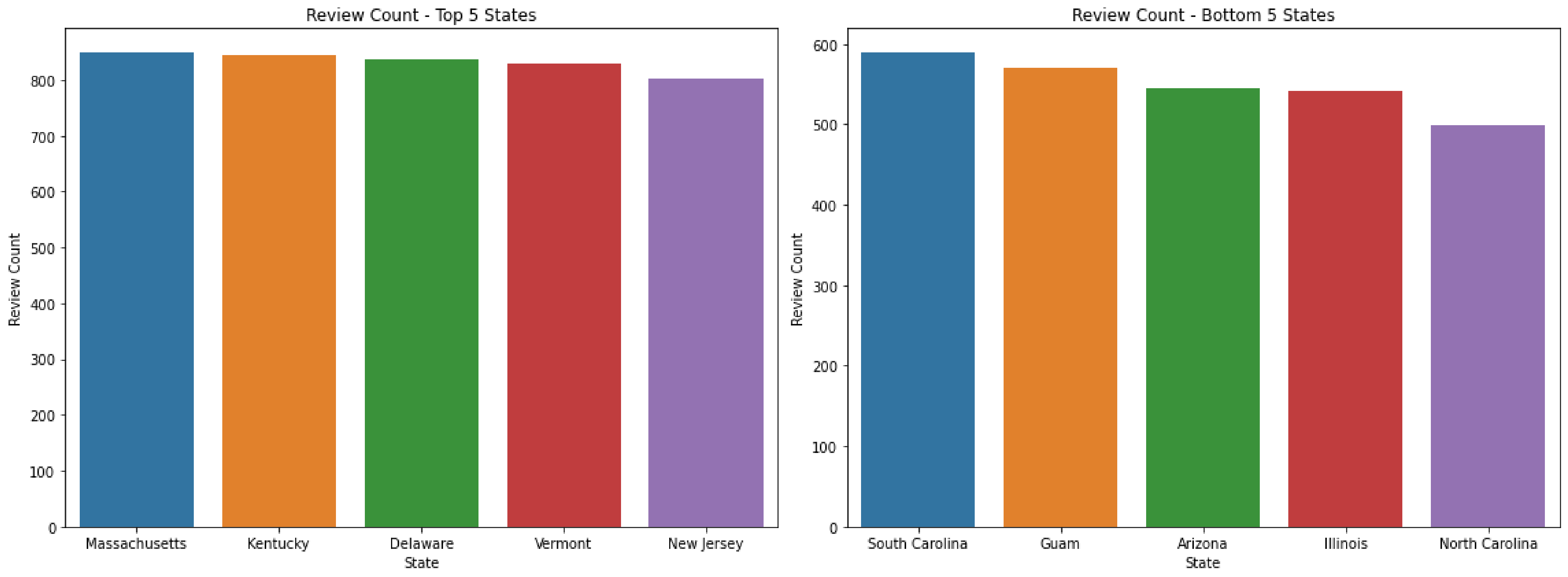
```python
# 4.4 What is the Average product rating of the top 10 brands?

#Specify the top 10 brands
top_brands = new_jcpenney_products['brand'].value_counts().head(10).index

# Create an empty dictionary to store the average product rating for each brand
average_rating_dict = {}

# Iterate through each top brand
for brand in top_brands:
    # Filter dataframe for the current brand
    brand_df = new_jcpenney_products[new_jcpenney_products['brand'] == brand]

    # Calculate the average product rating
    average_rating = brand_df['average_product_rating'].mean()
    # Store the information in the dictionary
    average_rating_dict[brand] = average_rating

# Plotting
plt.figure(figsize=(10, 6))
plot = sns.barplot(x=list(average_rating_dict.values()), y=list(average_rating_dict.keys()), palette='Blues')

# Add average rating text on each bar
for index, value in enumerate(list(average_rating_dict.values())):
    plot.text(value, index, f'{value:.2f}', va='center', fontsize=10, color='black')

plt.title('Average Product Rating of Top 10 Brands')
plt.xlabel('Average Rating')
plt.ylabel('Brand')
plt.show()
```



In [85]:

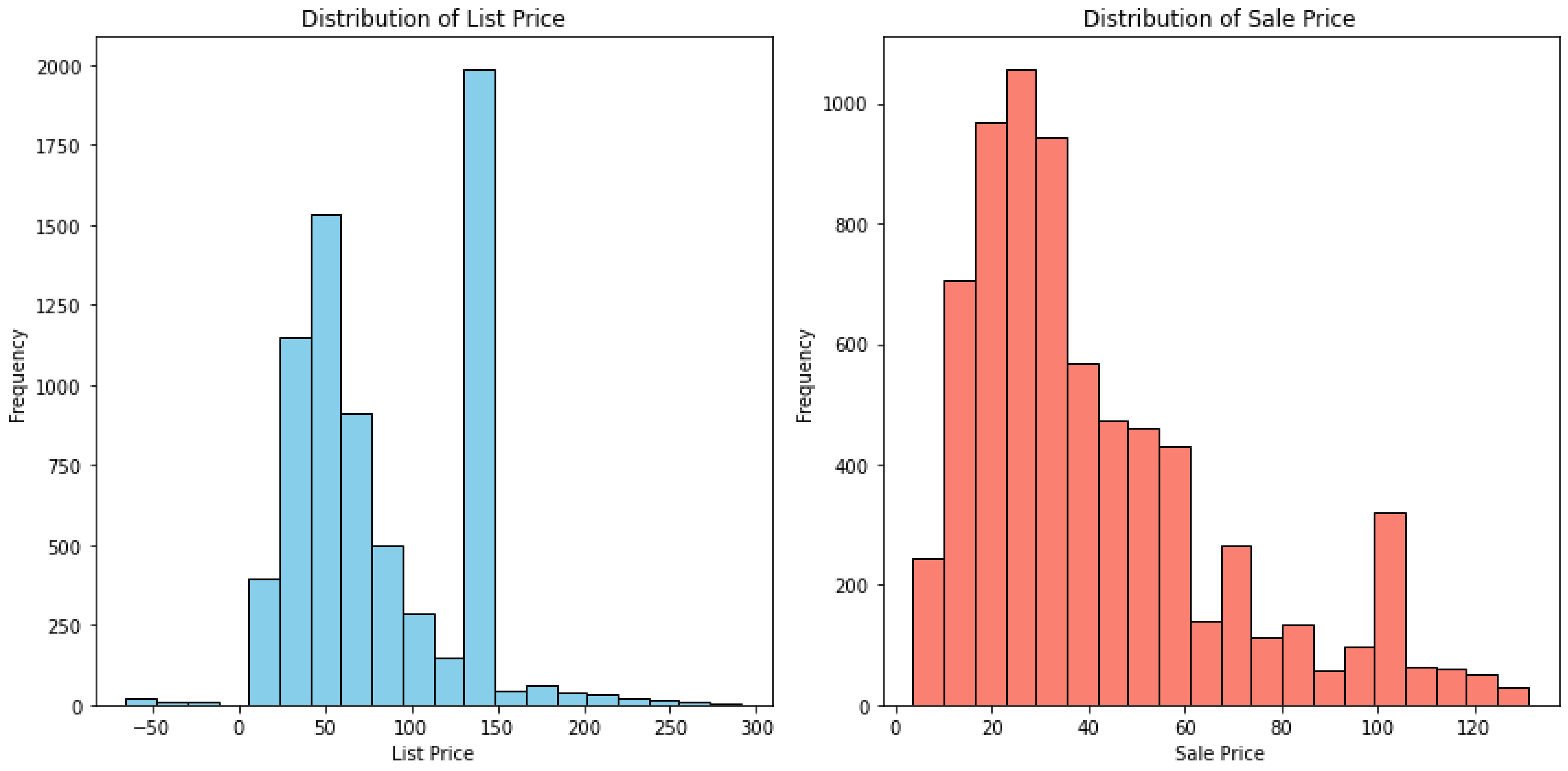
```
# 4.5 What is the review count based on the top 5 state and the bottom 5 state?

# Extract the top 5 and bottom 5 states
top_states = jcpenney_reviews['State'].value_counts().head(5).index
bottom_states = jcpenney_reviews['State'].value_counts().tail(5).index
```

```python
# Filter dataframe to include only the top and bottom states
top_states_df = jcpenney_reviews[jcpenney_reviews['State'].isin(top_states)]
bottom_states_df = jcpenney_reviews[jcpenney_reviews['State'].isin(bottom_states)]

# Create subplots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(16, 6))

# Plot for top 5 states
sns.countplot(x='State', data=top_states_df, order=top_states, ax=axes[0])
axes[0].set_title('Review Count - Top 5 States')
axes[0].set_xlabel('State')
axes[0].set_ylabel('Review Count')

# Plot for bottom 5 states
sns.countplot(x='State', data=bottom_states_df, order=bottom_states, ax=axes[1])
axes[1].set_title('Review Count - Bottom 5 States')
axes[1].set_xlabel('State')
axes[1].set_ylabel('Review Count')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



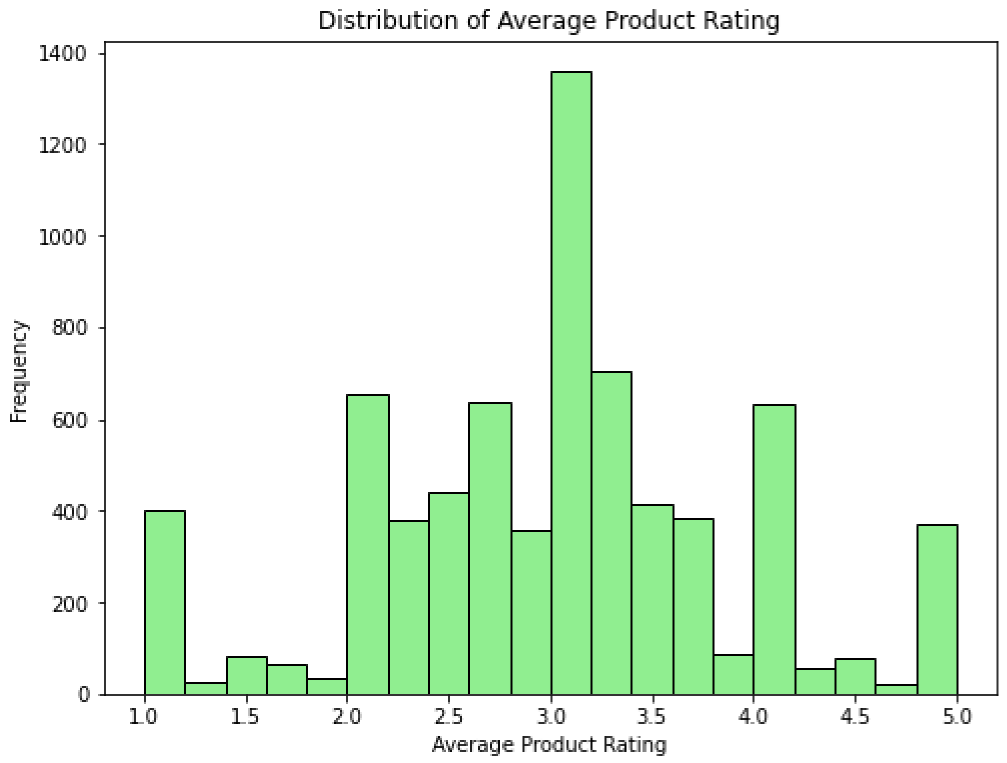
In [86]:

```python
# 4.6 What is the distribution of list_price and sale_price?

# Set up the figure with a 1x2 grid
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

# Histogram for list_price
axes[0].hist(new_jcpenney_products['list_price'], bins=20, color='skyblue', edgecolor='black')
axes[0].set_title('Distribution of List Price')
axes[0].set_xlabel('List Price')
axes[0].set_ylabel('Frequency')

# Histogram for sale_price
axes[1].hist(new_jcpenney_products['sale_price'], bins=20, color='salmon', edgecolor='black')
axes[1].set_title('Distribution of Sale Price')
axes[1].set_xlabel('Sale Price')
axes[1].set_ylabel('Frequency')

# Adjust layout to prevent overlap
plt.tight_layout()

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

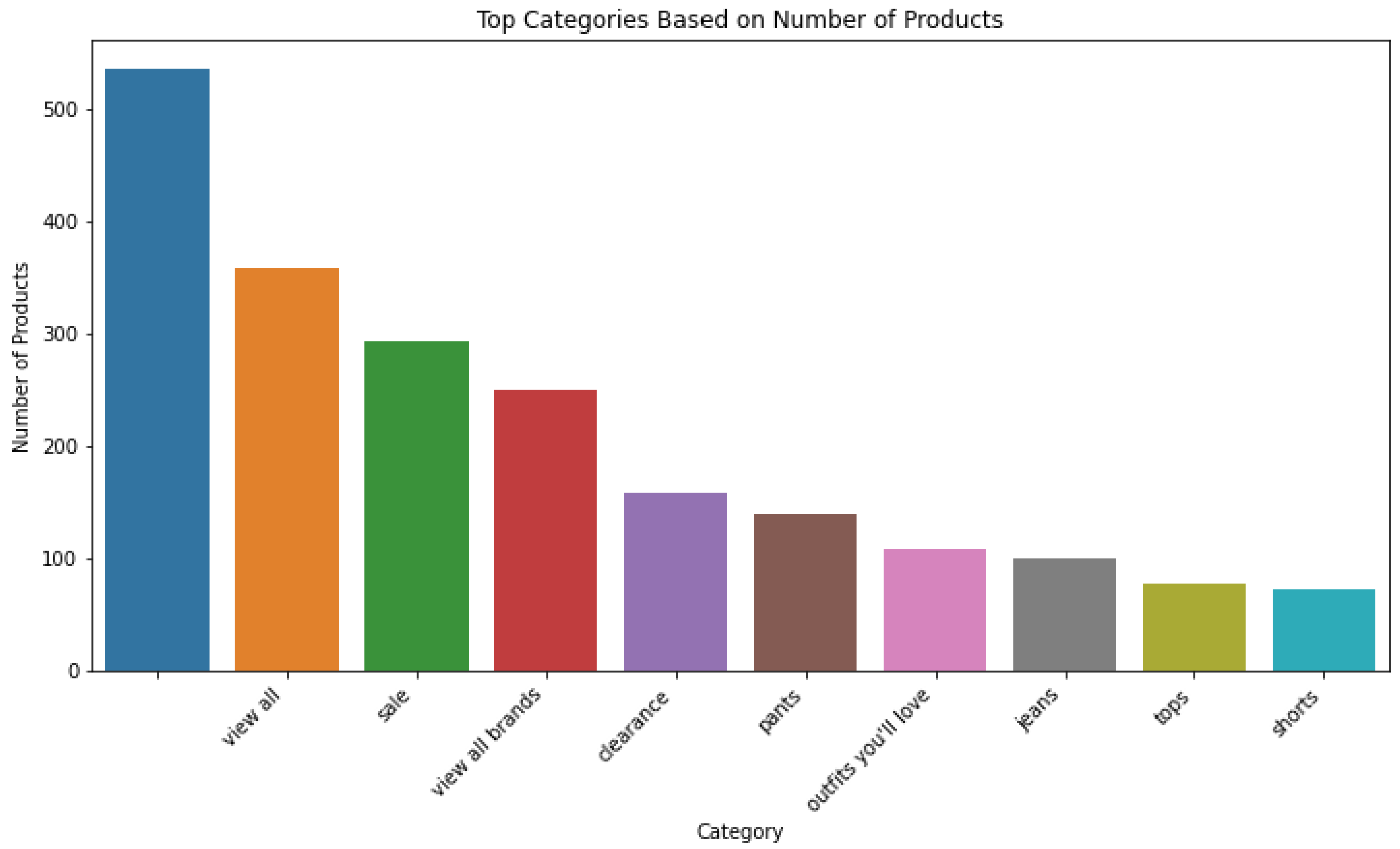


In [87]:

```
# 4.7 How does the distribution of average_product_rating look like?

# Set up the figure
plt.figure(figsize=(8, 6))

# Histogram for average_product_rating
plt.hist(new_jcpenney_products['average_product_rating'], bins=20, color='lightgreen', edgecolor='black')
plt.title('Distribution of Average Product Rating')
plt.xlabel('Average Product Rating')
plt.ylabel('Frequency')

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



In [88]:
```
# 4.8 What are the top categories based on the number of products?

# Count plot of top categories
plt.figure(figsize=(12, 6))
sns.countplot(x='category', data=new_jcpenney_products, order=new_jcpenney_products['category'].value_counts().index[:10])
plt.title('Top Categories Based on Number of Products')
plt.xlabel('Category')
plt.ylabel('Number of Products')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.show()
```



5.0 DATA ANALYSIS

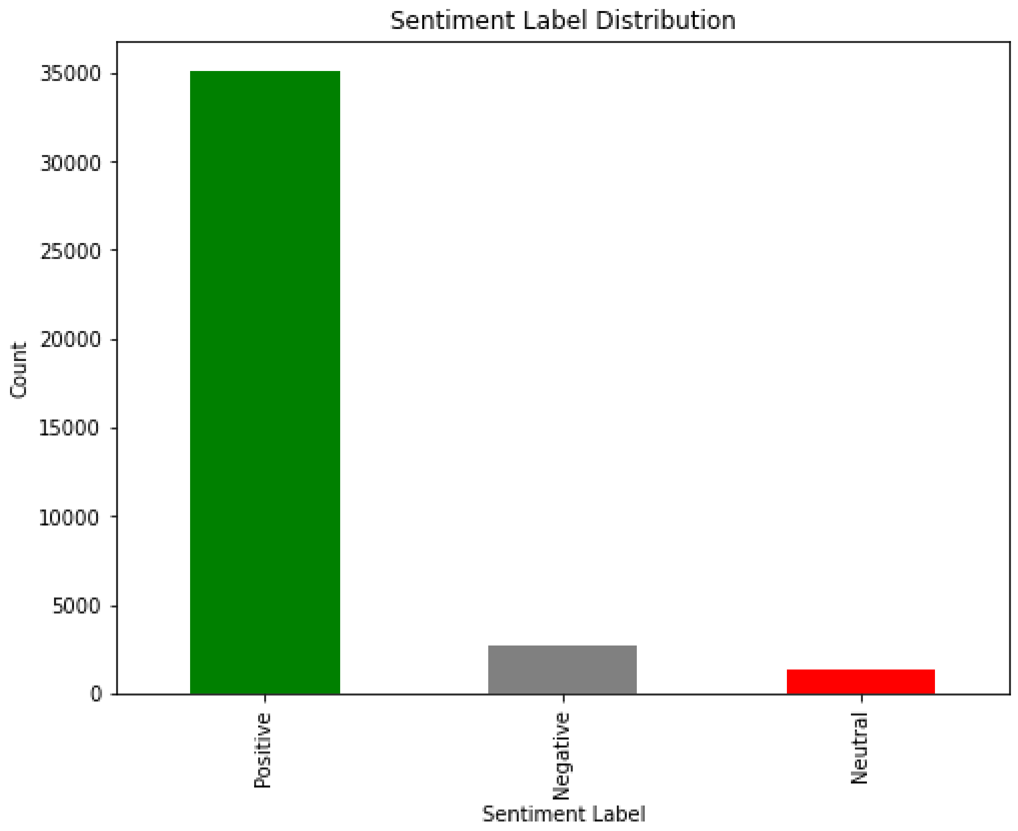
In [89]:
```
# 5.1 Sentiment Analysis

# Sentiment analysis is applied to each review in the 'Review' column of jcpenney_reviews
# The apply function is used to calculate the sentiment score for each review
# Two new columns are created in the DataFrame; 'sentiment_score' and 'sentiment_label'
# If the sentiment score is > 0, it is labeled as "Positive"; if < 0, it is labeled as "Negative"; and if = 0, it is
labeled as "Neutral
sid = SentimentIntensityAnalyzer()

# Apply sentiment analysis to each review and create a new column 'sentiment'
jcpenney_reviews['sentiment_score'] = jcpenney_reviews['Review'].apply(lambda x: sid.polarity_scores(x)['compound'])

# Classify sentiment based on polarity threshold and create a new column 'sentiment_label'
jcpenney_reviews['sentiment_label'] = jcpenney_reviews['sentiment_score'].apply(lambda x: "Positive" if x > 0 else
                                                                                        "Negative" if x < 0 else "Neutral")

# Print the DataFrame with sentiment analysis results
```

```
display(jcpenney_reviews[['Review', 'sentiment_score', 'sentiment_label']].head())

# This code essentially performs sentiment analysis on the reviews in the DataFrame
# It provides a numerical sentiment score and a labeled sentiment category for each review
```

	Review	sentiment_score	sentiment_label
0	You never have to worry about the fit...Alfred...	0.9423	Positive
1	These are great shirts, looks great all day. W...	0.8834	Positive
2	I purchase three rugs to replace twenty-year-o...	0.7959	Positive
3	I am a huge user of BE original fromula. I hap...	0.8345	Positive
4	Very soft and stretchy! They arent as dressy a...	0.0000	Neutral

In [90]:

```
# 5.1.1 Check distribution of Sentiment Label

# Plot the distribution of sentiment label
# count the occurrences of each sentiment label
# create a bar plot with sentiment labels on the x-axis and their respective counts on the y-axis

plt.figure(figsize=(8, 6))
jcpenney_reviews['sentiment_label'].value_counts().plot(kind='bar', color=['green', 'gray', 'red'])
plt.title('Sentiment Label Distribution')
plt.xlabel('Sentiment Label')
plt.ylabel('Count')
plt.show()

# This code visually represents the distribution of sentiment labels in the 'Review' column
# It provides insights into the overall sentiment tendencies, whether they are predominantly positive, neutral, or negative
# The plot shows majority of the reviews are positive
```



In [91]:

```
# 5.1.2 Summary Statistics

print("Mean Sentiment Score:", jcpenney_reviews['sentiment_score'].mean())
print("Median Sentiment Score:", jcpenney_reviews['sentiment_score'].median())
print("Standard Deviation of Sentiment Score:", jcpenney_reviews['sentiment_score'].std())
```

```
Mean Sentiment Score: 0.666142200614134
Median Sentiment Score: 0.8221
Standard Deviation of Sentiment Score: 0.38643068365704425
```

5.1.3 Interpretation of Sentiment Score Summary Statistics

As revealed in 5.1.2 above, the mean sentiment score is approximately 0.6661. This indicates that, on average, the sentiment expressed in the reviews tends to be positive, as the score is closer to 1 (positive) on the scale. The median sentiment score is 0.8221. The median is higher than the mean, suggesting that the majority of reviews have high positive sentiment, and the presence of some reviews with lower sentiment scores is pulling the mean down. The standard deviation is approximately 0.3864. This indicates a moderate level of variability in sentiment scores. Some reviews may have significantly lower sentiment scores, contributing to the spread around the mean.

The positive mean and median sentiment scores suggest that, overall, the reviews tend to express positive sentiments. The standard deviation suggests some variability, implying that there is a range of sentiment scores in the reviews, including both high and lower sentiment expressions.

In [92]:
```
# 5.1.4 Explore Specific Sentiments

# Separate the reviews into three categories based on their sentiment labels
# Display sample positive reviews

positive_reviews = jcpenney_reviews[jcpenney_reviews['sentiment_label'] == 'Positive']['Review']
print("Positive Reviews:")
print(positive_reviews.head())
print()

# Display sample negative reviews

negative_reviews = jcpenney_reviews[jcpenney_reviews['sentiment_label'] == 'Negative']['Review']
print("Negative Reviews:")
print(negative_reviews.head())
print()

# Display sample neutral reviews

neutral_reviews = jcpenney_reviews[jcpenney_reviews['sentiment_label'] == 'Neutral']['Review']
print("Neutral Reviews:")
print(neutral_reviews.head())

# The output provides a quick look at the content of reviews in each sentiment category
# This allows for a qualitative understanding of the sentiment expressed in the dataset
```

```
Positive Reviews:
0    You never have to worry about the fit...Alfred...
1    These are great shirts, looks great all day. W...
2    I purchase three rugs to replace twenty-year-o...
3    I am a huge user of BE original fromula. I hap...
6    I love these! Perfect find for my SouthWest ba...
Name: Review, dtype: object

Negative Reviews:
13    So before I begin this review I should start b...
23    I received this item today and it immediately ...
45    I bought this 2 months ago and it is nowhere a...
50    I did not like the fabric. It is 100% polyeste...
55    I wasnt sure about these shoe at first, but on...
Name: Review, dtype: object

Neutral Reviews:
4     Very soft and stretchy! They arent as dressy a...
```

```
5      Just what I expected and delivered on time as ...
37     Ive been using this cookware every day for the...
78     The quilt stitching is very poorly done. Two c...
114    As soon as the new Hairsetter Set came in my w...
Name: Review, dtype: object
```

In [93]:

```python
# 5.1.5 Find the most negative sentence and the most positive sentence in the Review column

# Find the most negative and most positive sentences
most_negative_sentence = jcpenney_reviews.loc[jcpenney_reviews['sentiment_score'].idxmin(), 'Review']
most_positive_sentence = jcpenney_reviews.loc[jcpenney_reviews['sentiment_score'].idxmax(), 'Review']

# Get their sentiment scores and locations (index)
most_negative_score = jcpenney_reviews['sentiment_score'].min()
most_positive_score = jcpenney_reviews['sentiment_score'].max()
most_negative_location = jcpenney_reviews['sentiment_score'].idxmin()
most_positive_location = jcpenney_reviews['sentiment_score'].idxmax()

# Display/print the results
print("Most Negative Sentence:")
print("Sentence:", most_negative_sentence)
print("Sentiment Score:", most_negative_score)
print("Location (Index):", most_negative_location)
print("\n")

print("Most Positive Sentence:")
print("Sentence:", most_positive_sentence)
print("Sentiment Score:", most_positive_score)
print("Location (Index):", most_positive_location)
```

```
Most Negative Sentence:
Sentence: The rug that I ordered, based on the picture, is NOT the rug that I received. The rug that I received is a simplified, crude versio
n of what is pictured. In the photo, the rug has what is often called a ring of wheat leaves, immediately outside the central blank area, and
then just beyond that are parallel ropes, with crosses above the smaller four outside medallions, and small accent lines crossing the ropes a
t the corners above the larger four outside medallions. The rug that I received has no ropes, just parallel lines with random messy spots of
another color in them, the four smaller outside medallions have been eliminated (this may be needed in the 5 size, but should be described an
d/or pictured), and there are no crosses and no lines across what should have been ropes. The pattern of the wool has random spots of other c
olors where they should not be. At the edges, the fabric sewn to the bottom shows badly. Not only is this design a crude version of what was
pictured, it is a poorly executed example of the crude version of the design. The rug in the photo is a well made rug. The rug I received is
not well made. Its a mess. In addition, it is about 3/4 thick, not the 1/2 specified in the description. What a disappointment and inconvenie
nce to receive something of much lower quality than what was shown in the photo, and have to return it to a JCP store. Not good.
Sentiment Score: -0.9892
Location (Index): 3374



Most Positive Sentence:
Sentence: I was pleasantly surprised with this bedding. I havent shopped at Pennys in a long while but have recently noticed them popping up
when doing searches online for certain household items. I have been looking for a long time for a nice comforter in these colors which are ve
```

```
ry pretty in person and true to what is seen online. I dont usually want all the extra things that come with bedding sets but in this case I
actually like the shams and pillows. Sort of a miracle because I am usually so disappointed in the accompanying items that I dont use them an
d am annoyed at having to pay for things I dont even want or use. But I will this time! The accent pillows are especially nice and pretty. Th
ey are flattened from shipping but they plump up very well after a while. I ironed/steamed all which greatly improves the look of everything.
The shams are just as nice and the comforter is fluffy and just the thickness and fluff I was hoping for. The bed skirt is too short for my o
lder Drexel Heritage 4 poster bed but Im typically not a fan of bedskirts anyway. I knew it would be too short when I ordered but since I was
nt planning on using it that was OK. I could add trim to the bottom and make it work but I am leaning toward lining it with heavier lining, b
oard mounting it and using it as a pleated valance for the window. It would be perfect for that and solve my window treatment need. My only c
omplaint about this set would be that the solid fabric is thin, almost like a suit lining, maybe thinner, but its still OK, well see how it h
olds up over time. The patterned face fabric of the comforter is thicker but the solid fabric is very thin. The stitching seems ok and stron
g. Because of the thin fabric I gave the quality an average rating but rated everything else as excellent. Im very pleased with my purchase,
the beautiful look, the great price and quick shipping I received from JC Penny.
Sentiment Score: 0.9986
Location (Index): 18162
```

In [94]:

```python
# 5.2 What are the most frequent words used by customers when giving reviews?

# Combine all reviews into a single string
all_reviews_text = ' '.join(jcpenney_reviews['Review'])

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_reviews_text)

# Display the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title(' Most Frequent Words in Product Reviews')
plt.show()

# The code provides a visualization representing the most frequent words used in the product reviews
# Word size is proportional to frequency in the combined text
# The output shows words like; love, fit, an comfortable are the most frequently used, signifying customers product
satisfaction
```

Most Frequent Words in Product Reviews

6.0 Data Augmentation

- New data (JC Penney) was obtained from Kaggle website to bring new insights to the existing data
- Data source: https://www.kaggle.com/datasets/PromptCloudHQ/all-jc-penny-products/

In [95]:
```
# 6.1 Exploration of new CSV file

# Load csv file
# Assign data to a new variable

jcpenney_ecommerce = pd.read_csv(r"C:\Users\oob00011\OneDrive - University of Stirling\Semester Courses -
Autumn\Representing & Manipulating Data\Assignment 2\JCPenneyFiles\jcpenney_ecommerce.csv")
print("Displaying the content of jcpenney_ecommerce.csv file:")
display(jcpenney_ecommerce.head())
```

```
Displaying the content of jcpenney_ecommerce.csv file:
```

	uniq_id	sku	name_title	description	list_price	sale_price	category	category_tree	average_product_rating	
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	alfred dunner	jcpenney|women|alfred dunner	4.7 out of 5	http://www.jc

	uniq_id	sku	name_title	description	list_price	sale_price	category	category_tree	average_product_rating	
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	alfred dunner	jcpenney|women|alfred dunner	4.7 out of 5	http://www.jc
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	4.7 out of 5	http://www.jc
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	4.7 out of 5	http://www.jc
4	d969a8542122e1331e304b09f81a83f6	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	4.7 out of 5	http://www.jc

In [96]:
```
# 6.2 Shape of the jcpenney_ecommerce data set

print("Shape of jcpenney_ecommerce is:", jcpenney_ecommerce.shape)
```

```
Shape of jcpenney_ecommerce is: (20000, 14)
```

In [97]:
```
# 6.3 General information about jcpenney_ecommerce Dataframe

jcpenney_ecommerce.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   uniq_id                 20000 non-null  object
 1   sku                     18768 non-null  object
 2   name_title              13921 non-null  object
 3   description             13235 non-null  object
 4   list_price              10335 non-null  object
 5   sale_price              13898 non-null  object
 6   category                12840 non-null  object
 7   category_tree           12840 non-null  object
 8   average_product_rating  7982 non-null   object
 9   product_url             20000 non-null  object
 10  product_image_urls      13711 non-null  object
 11  brand                   13921 non-null  object
 12  total_number_reviews    7149 non-null   object
 13  Reviews                 7982 non-null   object
dtypes: object(14)
memory usage: 2.1+ MB
```

In [98]:

```python
# 6.4 Rename the 'Reviews' column to 'Review'

jcpenney_ecommerce.rename(columns={'Reviews': 'Review'}, inplace=True)

# Display the DataFrame after renaming
print("\nDataFrame after renaming:")
display(jcpenney_ecommerce)
```

```
DataFrame after renaming:
```

	uniq_id	sku	name_title	description	list_price	sale_price	category	category_tree	average_product_rating	
0	b6c0b6bea69c722939585baeac73c13d	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	alfred dunner	jcpenney|women|alfred dunner	4.7 out of 5	http://w
1	93e5272c51d8cce02597e3ce67b7ad0a	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	alfred dunner	jcpenney|women|alfred dunner	4.7 out of 5	http://w
2	013e320f2f2ec0cf5b3ff5418d688528	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	4.7 out of 5	http://w

	uniq_id	sku	name_title	description	list_price	sale_price	category	category_tree	average_product_rating	
3	505e6633d81f2cb7400c0cfa0394c427	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	4.7 out of 5	http://w
4	d969a8542122e1331e304b09f81a83f6	pp5006380337	Alfred Dunner® Essential Pull On Capri Pant	You'll return to our Alfred Dunner pull-on cap...	41.09	24.16	view all	jcpenney|women|view all	4.7 out of 5	http://w
...	...	...	...	...	...	...	...	...	...	
19995	f8701d2f3eb9d7cc035e223d9f75e433	pp5004221198	NaN	NaN	NaN	NaN	NaN	NaN	NaN	http://
19996	7f00f6b0752560e8ddcdb327bed8ca11	pp5006020298	NaN	NaN	NaN	NaN	NaN	NaN	NaN	htt
19997	9fa7a25e5aef95c1f35e32cb7a4cfcc2	pp5007270180	NaN	NaN	NaN	NaN	NaN	NaN	NaN	http:/
19998	4275ea350e678feec08d21490e9ac517	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	http://w
19999	083d8e18b9ea3d1c756d5173fcc3ba84	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	http://\

20000 rows × 14 columns

In [99]:
```
# 6.5 Concatenate the dataframes of 'reviews_csv' and 'jcpenney_ecommerce' vertically

# Creates a new DataFrame that consolidates the 'Review' column from both original DataFrames

concatenated_df = pd.concat([reviews_csv, jcpenney_ecommerce], ignore_index=True)

final_df = concatenated_df[['Review']]
display(final_df.head())

# The code ensured final_df now contains only the 'Review' column from both Dataframes
# The first few rows are displayed
```

	Review
0	You never have to worry about the fit...Alfred...

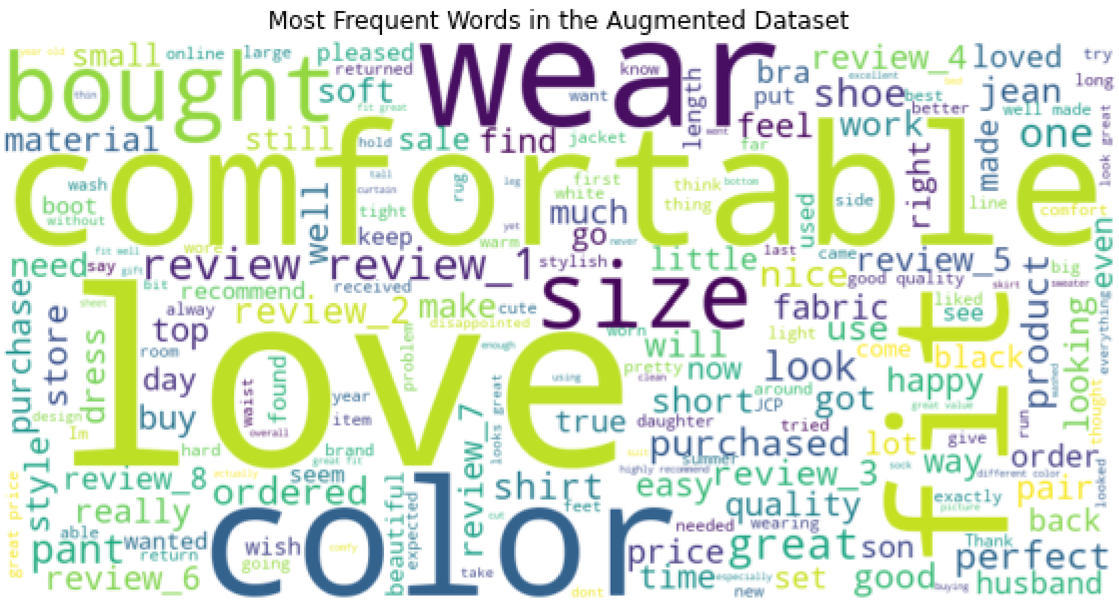
	Review
1	Good quality fabric. Perfect fit. Washed very ...
2	I do not normally wear pants or capris that ha...
3	I love these capris! They fit true to size and...
4	This product is very comfortable and the fabri...

In [101…

```
# 6.6 Replace NaN with empty strings

final_df['Review'].fillna('', inplace=True)

# As highlighted in 6.4, the dataframe coontains missing values - NaN
# For the purpose of analysing the text (Review column), missing values are converted from numeric to strings
```

In [102…

```
# 6.7 Based on new dataset, are there new insights in terms of 'most frequently used words' by customers when
giving reviews?

# Combine all reviews into a single string

reviews_text = ' '.join(final_df['Review'])

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(reviews_text)

# Display the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Most Frequent Words in the Augmented Dataset')
plt.show()

# There seem to be no significant change in the new dataset when compared to the former based on Word Cloud analysis
# This suggests the customers enjoy buying and using JC Penney's products
```



In [103…

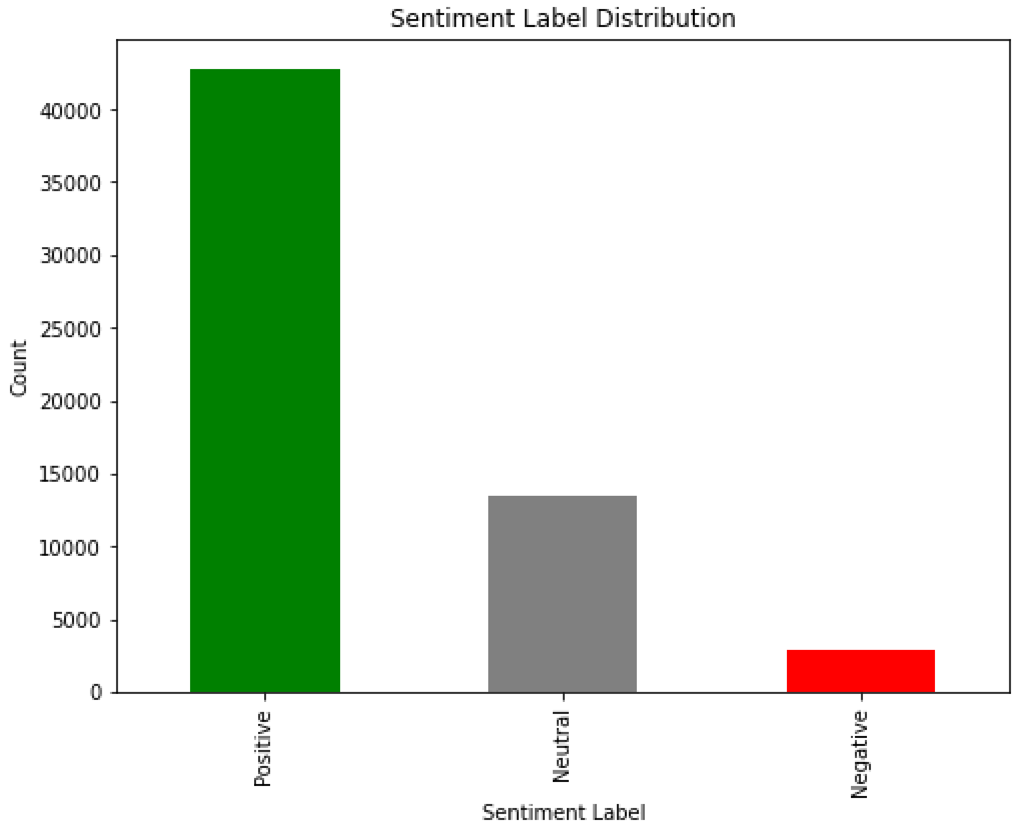
```python
# 6.8 Creating a Sentiment Analysis on the Review column of the new dataset

# Create a copy of the DataFrame
final_df = final_df.copy()

final_df['sentiment_score'] = final_df['Review'].apply(lambda x: sid.polarity_scores(x)['compound'])
final_df['sentiment_label'] = final_df['sentiment_score'].apply(lambda x: "Positive" if x > 0 else "Negative" if x
                                                                 < 0 else "Neutral")
display(final_df[['Review', 'sentiment_score', 'sentiment_label']].head())
```

	Review	sentiment_score	sentiment_label
0	You never have to worry about the fit...Alfred...	0.9423	Positive
1	Good quality fabric. Perfect fit. Washed very ...	0.8408	Positive
2	I do not normally wear pants or capris that ha...	0.9514	Positive
3	I love these capris! They fit true to size and...	0.9329	Positive
4	This product is very comfortable and the fabri...	0.7172	Positive

In [104…

```python
# 6.9 Check distribution of Sentiment Label

# Plot the distribution of sentiments
plt.figure(figsize=(8, 6))
final_df['sentiment_label'].value_counts().plot(kind='bar', color=['green', 'gray', 'red'])
```

```
plt.title('Sentiment Label Distribution')
plt.xlabel('Sentiment Label')
plt.ylabel('Count')
plt.show()

# When compared with 5.1.1, the number of customers that are neutral are higher
# In both datasets though, the positive reviews from customer seem to tower above negative reviews or those that are
neutral
```



7.0 RECOMMENDATIONS

(a) Based on the review count by age group in 4.2, JC Penney seem to have customers scattered across the age range 25-65+. However, the buyers within the age range 18-24 seem low. In this modern world of social media, the company would do well to target such customers via social media marketing. Perhaps, lack of funds is also a factor for this group. The company can make use of sales pricing (discounts, bonuses etc)to attract customers in this age bracket.

(b) The company would also do well to look into the bottom 5 states where they are experiencing low customers/sales as revealed in 4.1, and see how to penetrate these areas with aggressive marketing and sales promotion.

(c) Having a single customer database is also vital for targeted marketing of the company's cold and warm customers.

```
In [ ]:
```