Marketing Data Optimization: Cleaning, Standardization, and SEO-Driven Title Enhancement

Table of Contents

Introduction

In the realm of digital marketing, data quality plays a crucial role in driving informed decisions, enhancing customer engagement, and optimizing sales performance. This project focuses on preparing raw product data for effective marketing analysis.

The main objectives of this project are:

- Clean the dataset by addressing data quality issues such as missing values, duplicates, and inconsistent formats to ensure data accuracy and reliability.
- Standardize data formats and resolve encoding issues to improve data integrity for seamless analysis.
- Introduce a new feature, `short_title`, to create concise, SEO-friendly product titles that enhance readability and search visibility.
- Optimize data quality and product titles to support deeper marketing insights and boost product discoverability.

Data Source

The dataset, `productdata.xlsx`, contains product-related information, including product IDs, titles, bullet points, descriptions, product types, and dimensions. This dataset was provided in Excel format and served as the primary source for the data cleaning process. It was provided by **HNG Tech** as part of the internship program, serving as a practical case study for applying data cleaning, standardization, and optimization techniques.

Using the `data.shape` function, it was discovered that the dataset contains **3,847 rows** and **6 columns**, providing a comprehensive set of product data for analysis.

Tools

- **Google Sheet:** Used for data gathering and univariate analysis
- **Python**: The primary language for data manipulation.
    - **Pandas**: For data loading, exploration, and cleaning.
    - **Regular Expressions (re)**: To clean text data and remove unwanted patterns.
    - **Matplotlib (plt):** For data visualization, helping to analyze and display patterns.
- **Google Colab:** Provided a cloud-based platform for executing Python code efficiently.

Data Loading

The following Python code was used to load the dataset into Google Colab and verify successful import:

```
# Import libraries
import pandas as pd
import re
import matplotlib.pyplot as plt

# Load the dataset
file_path = '/content/productdata.xlsx'
data = pd.read_excel(file_path)
print('File uploaded successfully')
```

Data Cleaning and Preparation

Raw data often contains inconsistencies, errors, and missing values. This phase involved data cleansing, including handling missing data and correcting inconsistencies to ensure data accuracy and reliability for subsequent analysis.

The following Python code was used to explore the dataset and gain an initial understanding of the data:

```
# Display the first few rows of the dataframe to understand its structure
data.head()

# This function provided a quick overview of the data
data.info
```

- `data.head()`: Displays the first five rows of the dataset, providing a quick overview of the data structure, column names, and sample entries.
- `data.info()`: Provides a concise summary of the dataset, including the number of entries and data types, offering insights into the presence of null values.

The DESCRIPTION column had the highest number of missing data at 52.4%, followed by BULLET_POINTS at 38.9% with ProductLength and PRODUCTTYPEID both at 4.4%

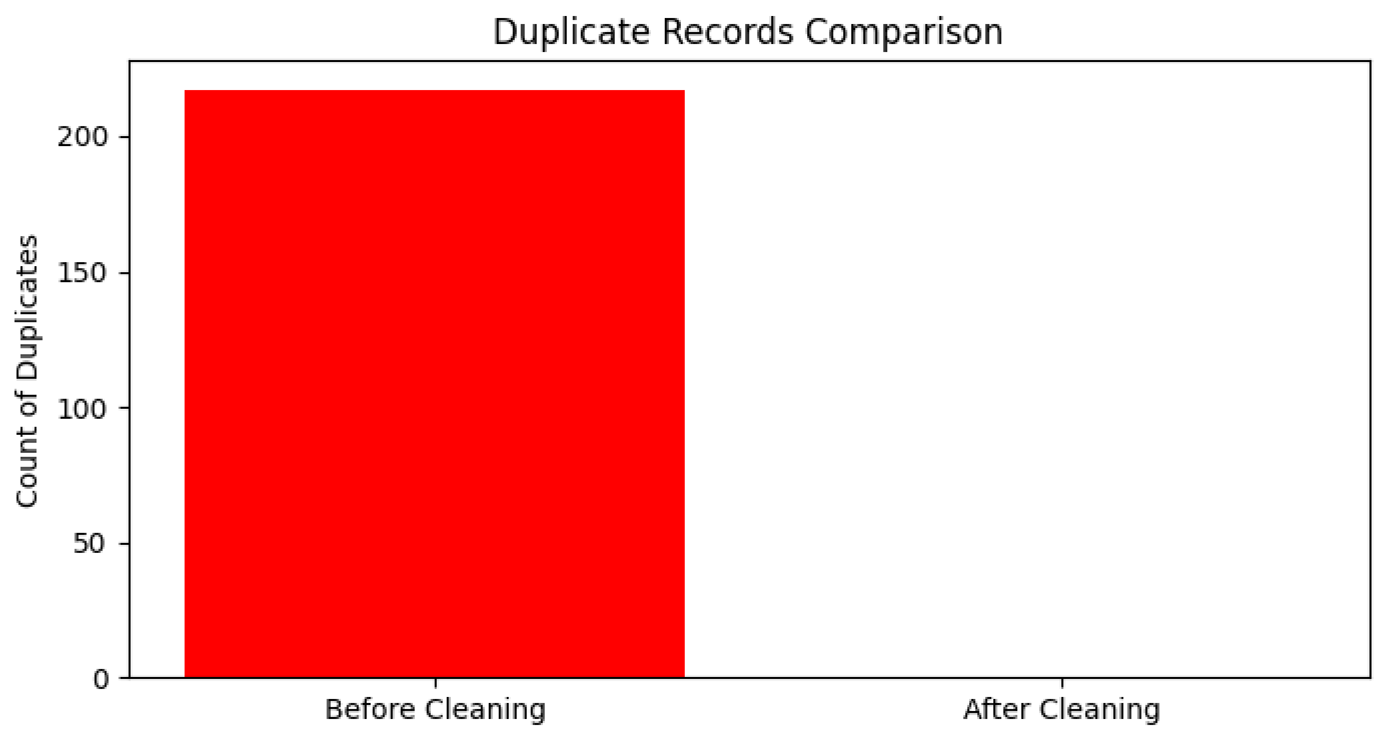
Duplicate Values

The code below was used to identify and remove duplicate records from the dataset:

```
# Check for duplicate records
data.duplicated().sum()

# Remove duplicate records
data = data.drop_duplicates()

# Verify that duplicates have been removed
data.duplicated().sum()
```



A total of **217 duplicate records** were found and successfully removed, ensuring data accuracy. The final check confirmed that no duplicates remained in the dataset.

Null Values

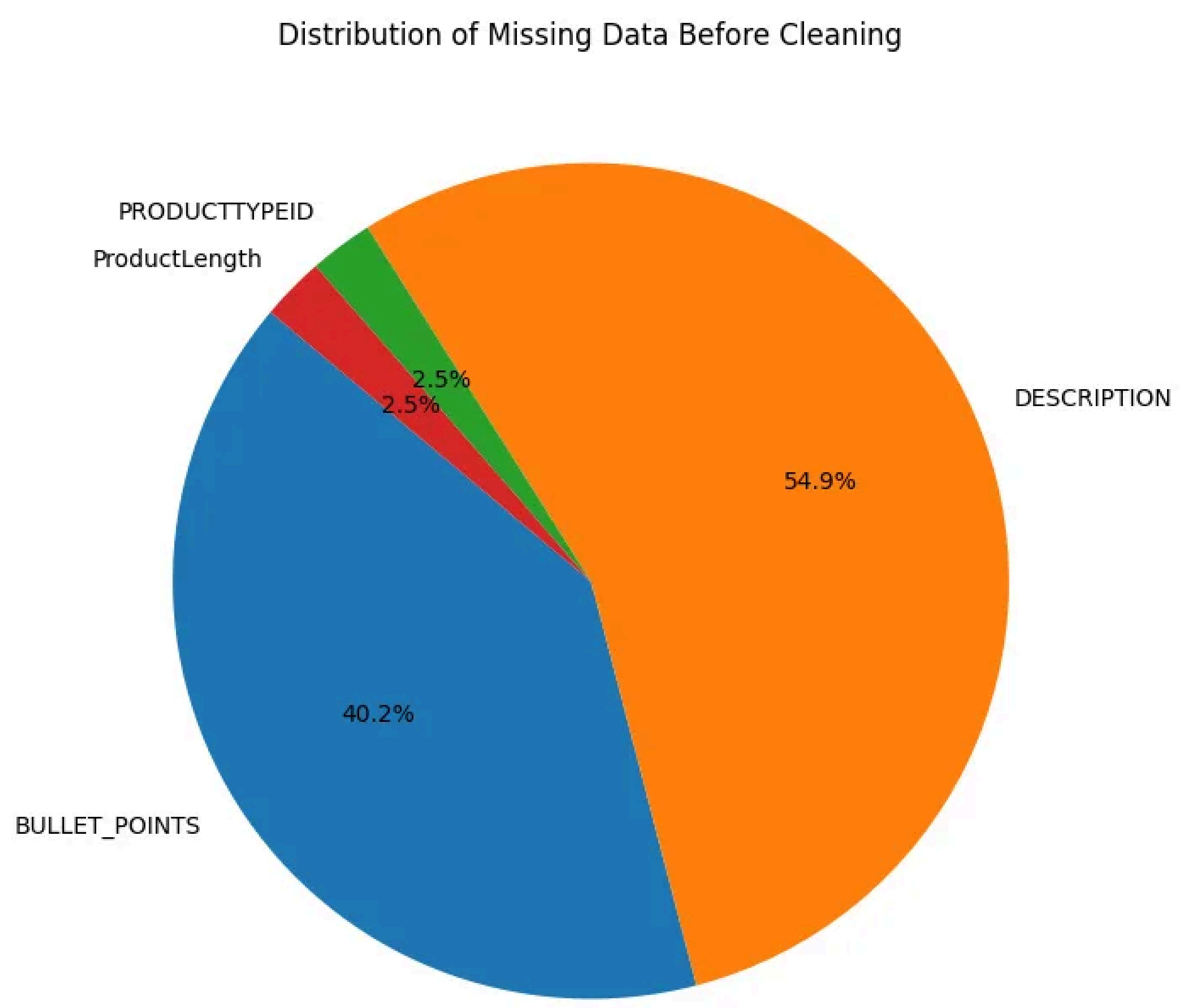
The code below was used to identify and handle null values in the dataset:

```
# Check for null values
data.isnull().sum()

# Plot for distribution of null values
missing_data = data.isnull().sum()
missing_data = missing_data[missing_data > 0]
plt.figure(figsize=(8, 8))
plt.pie(missing_data, labels=missing_data.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Missing Data Before Cleaning')
plt.show()

# Drop rows with null values in PRODUCTTYPEID and ProductLength
data = data.dropna(subset=['PRODUCTTYPEID', 'ProductLength'])

# Handle missing values in BULLET_POINTS and DESCRIPTION
data['BULLET_POINTS'].fillna('No information available', inplace=True)
data['DESCRIPTION'].fillna('No information available', inplace=True)
```



A total of **1,985 (54.9%) null values** were found in the **DESCRIPTION** column and **1,452 (40.2%) null values** in the **BULLET_POINTS** column. Additionally, **89 (2.5%) null values** were identified in both **PRODUCTTYPEID** and **ProductLength**. Rows with missing values in **PRODUCTTYPEID** and **ProductLength** were removed, while missing values in **BULLET_POINTS** and **DESCRIPTION** were filled with the placeholder **"No information available"** to maintain data completeness.

Standardizing Column Names

To ensure consistency and improve readability, column names were standardized using the code below:

```
# Standardize column names
data.rename(columns={
    'PRODUCTID': 'product_id',
    'TITLE': 'title',
    'BULLET_POINTS': 'bullet_points',
    'DESCRIPTION': 'description',
    'PRODUCTTYPEID': 'product_type_id',
    'ProductLength': 'product_length'
}, inplace=True)

# Confirm changes
data.columns
```

This process converted column names to a consistent lowercase format with underscores, enhancing clarity and making the dataset easier to work with. The changes were verified using the `data.columns` function.

Cleaning Text Data

During univariate analysis, several text-related issues were identified in the **title**, **bullet_points**, and **description** columns. These issues included:

- **Foreign Characters:** Non-ASCII characters and corrupted symbols (e.g. `�`) that compromised text readability.
- **HTML Tags:** Embedded HTML tags such as `<br>`, `<p>`, and HTML entities like `&nbsp;` that cluttered the text.

- **Un spaced Words:** Missing spaces after punctuation, especially commas, making text harder to read.
- **Encoding Issues:** Inconsistent text encoding leading to display errors and garbled characters.
- **Irregular Formatting:** Fully uppercase words and inconsistent sentence capitalization.

Encoding Fixes

- Converted text from `latin1` to `UTF-8` to standardize encoding across the dataset, eliminating garbled characters.
- Removed **non-ASCII characters** and problematic symbols like `�` that can occur due to encoding mismatches.

Removing Special Characters and Tags

- **Hexadecimal Encodings (e.g., `_x0000_`):** Removed to clean up hidden or system-generated codes.
- **HTML Tags & Entities:** Tags like `<br>`, `<p>`, and entities like `&nbsp;` were stripped out to ensure plain, readable text.

Bullet Point-Specific Cleaning

- **Square Brackets:** Removed unnecessary square brackets (`[]`) that often clutter product details.
- **Comma Spacing:** Added spaces after commas where missing.

Formatting Improvements

- **Capitalization:** Fully uppercase words were properly capitalized for better readability.
- **Sentence Structure:** Ensured each sentence starts with a capital letter, improving text flow.
- **Whitespace Management:** Removed leading/trailing spaces to maintain clean formatting.

Code

To address these issues, the following Python code was implemented:

```
def clean_text(text, is_bullet_points=False):
    try:
        text = str(text)

        # Encoding and special character cleaning
        text = text.encode('latin1', errors='ignore').decode('utf-8', errors='ignore')
        text = re.sub(r'[^\x00-\x7F]+', '', text)              # Remove non-ASCII characters
        text = re.sub(r'[�]+', '', text)                       # Remove problematic characters
        text = re.sub(r'_x[0-9A-Fa-f]{4}_', '', text)          # Remove hex-like encoding
        text = re.sub(r'&nbsp;', ' ', text)                    # Replace HTML space entities
        text = re.sub(r'<.*?>', '', text)                      # Remove HTML tags

        # Bullet point specific cleaning
        if is_bullet_points:
            text = re.sub(r'[\[\]]', '', text)                 # Remove square brackets
            text = re.sub(r',(?=[^\s])', ', ', text)           # Add space after commas if missing

        # Formatting improvements
        text = ' '.join([word.capitalize() if word.isupper() else word for word in text.split()])
        text = '. '.join([sentence.strip().capitalize() for sentence in text.split('. ')])

    except Exception as e:
        print(f"Error processing text: {e}")
        return text.strip()  # Return cleaned text even if there's an error

    return text.strip()

# Apply the updated cleaning function
data['title'] = data['title'].apply(clean_text)
data['bullet_points'] = data['bullet_points'].apply(lambda x: clean_text(x, is_bullet_points=True))
```

```
data['description'] = data['description'].apply(clean_text))

data.shape
```

The provided code is designed to clean and standardize text data within the dataset, specifically targeting the `title`, `bullet_points`, and `description` columns. The core function, `clean_text()`, applies a series of text preprocessing steps to enhance the readability and consistency of the data. The code then handles encoding issues by removing non-ASCII characters, problematic symbols (e.g., `�`), and hexadecimal-like encodings (e.g., `_x0000_`). It also strips out HTML tags and replaces HTML space entities (e.g., `&nbsp;`) with regular spaces to clean up any web-sourced text.

When cleaning bullet points specifically (indicated by `is_bullet_points=True`), the function removes square brackets and ensures proper spacing after commas. Additionally, the function improves text formatting by capitalizing words that are entirely in uppercase, ensuring proper sentence case, and applying consistent punctuation rules. To prevent any processing errors, the function includes an exception handling mechanism that logs issues without breaking the data processing workflow.

Finally, the cleaning function is applied to the dataset using the `apply()` method, ensuring each record in the `title`, `bullet_points`, and `description` columns undergoes the cleaning process. This results in a dataset with well-formatted, clean, and user-friendly text, suitable for further analysis, reporting, or display.

After completing the data cleaning process, the dataset's structure was re-evaluated using `data.shape`, revealing the following:

- **Initial Dataset: 3,847 rows** and **6 columns**.
- **Post-Cleaning Dataset: 3,541 rows** and **6 columns**.
- **Rows Removed: 306 rows** were removed due to:
  - **Duplicate Entries:** Ensuring data accuracy.
  - **Missing Critical Values:** Dropping records with nulls in essential fields like `product_type_id` and `product_length`.

Short Title Feature Methodology

The objective of creating the `short_title` feature was to develop concise, SEO-friendly versions of product titles that retain essential information while enhancing readability and optimization. This report details the methodology used to achieve this goal.

Data Exploration and Pattern Identification

Using `data.head(100)`, a sample of the data was extracted to understand the structure and content of the product titles. Through univariate analysis, several patterns were identified that informed the logic for generating optimized short titles. Key observations included:

1. **Significance of Colors:**

   Colors mentioned in the titles, such as "Red," "Black," and "Navy," played a vital role in describing product variants. Recognizing this, colors were explicitly included in the short titles to preserve meaningful product distinctions.

2. **Importance of the Last Two Words:**

   The last two words in the titles often contained critical product details, such as size, quantity, or model number. This observation prompted the inclusion of these words in every short title to maintain product specificity.

3. **Use of Hyphen:**

   A hyphen was introduced before the last two words to create a clear separation between the main product description and these important details, improving both readability and structure.

Text Cleaning and Formatting

To ensure the short titles were clean, concise, and optimized, the following text preprocessing techniques were applied:

- **Proper Formatting:** Each word in the short title was capitalized to maintain a professional and uniform appearance.
- **Introduction of Stop Words:** Common redundant phrases and stop words (e.g., "includes," "set of," "features") were removed to reduce clutter and focus on essential information.

- **Character Removal:** Unnecessary characters such as colons, brackets, and underscores were stripped from the titles to improve clarity.

Logic Implementation

The core logic for generating the `short_title` feature involved the following steps:

1. **Data Cleaning:** Removing special characters and formatting the text.
2. **Color Identification:** Detecting color names in the titles and ensuring their inclusion.
3. **Keyword Extraction:** Retaining the first few significant words to summarize the product.
4. **Retention of Critical Details:** Including the last two words from the original title, preceded by a hyphen.
5. **Length Management:** Ensuring the title remained between 30-50 characters without cutting words midway. If a word exceeded the character limit, it was omitted entirely rather than partially displayed.

Short Title Examples

- **Original Title:** Delavala self adhesive kitchen backsplash wallpaper, oil proof aluminum foil kitchen sticker (sliver 5(mtr))

  Short Title: Delavala Self Adhesive Kitchen - Sliver 5Mtr
- **Original Title:** Hexwell essential oil for home fragrance oil aroma diffuser oil set of 2 rajnigandha oil & teatree oil -10ml each

  Short Title: Hexwell Essential Oil Home Fragrance - 10Ml Each
- **Original Title:** Hot wheels 2004 first editions crooze wail tale 77/100 white 077 1:64 scale

  Short Title: Hot Wheels 2004 First Editions White - 1:64 Scale
- **Original Title:** Whey protein isolate 24g protein per serve,990g - 30 servings (5 flavors) (cafe mocha, 2 kg)

  Short Title: Whey Protein Isolate 24G Protein - 2 Kg
- **Original Title:** Cqlek flexible laptop stand adjustable for desk, portable ipad stand computer stand, aluminum laptop riser holder, foldable laptop cooling stand with 6 levels height adjustment

  Short Title: Cqlek Flexible Laptop Stand - Height Adjustment

Code

The code and logic implemented to generate the `short_title` are presented below.

```
data.head(100)

def create_formatted_short_title(title):
    if len(title) < 50:
        return title.title()  # Return title-cased original title if it's less than 50 characters

    # Identify colors commonly used in product descriptions
    colors = ['black', 'white', 'red', 'blue', 'green', 'yellow', 'pink', 'purple',
              'orange', 'brown', 'grey', 'gray', 'navy', 'beige', 'gold', 'silver',
              'cream', 'maroon']

    # Remove brackets, underscores, and replace with space
    cleaned_title = re.sub(r'[\(\)]', '', title)  # Removing brackets
    cleaned_title = cleaned_title.replace('_', ' ')  # Replacing underscores with spaces

    # Extract color if present
    color_found = next((color for color in colors if color in cleaned_title.lower()), '')

    # Remove redundant words/phrases
    redundant_phrases = ['includes', 'set of', 'features', 'for', 'with', 'and', '|', '-', 't86']
    cleaned_title = re.sub(r'\b(?:' + '|'.join(redundant_phrases) + r')\b', '', cleaned_title, flags=re.IGNORECASE)
    # Extract key components (first 5 significant words + color if found)
    words = cleaned_title.split()
    key_components = words[:5]  # First 5 significant words for more content
    if color_found and color_found.capitalize() not in key_components:
        key_components.append(color_found.capitalize())  # Add color if not already included

    # Include the last 2 words from the original title, separated by a hyphen
    last_two_words = title.strip().replace('_', ' ').replace('(', '').replace(')', '').split()[-2:]
    mandatory_parts = []
    if color_found:
        mandatory_parts.append(color_found.capitalize())
    mandatory_parts.extend(['-'] + last_two_words)

    # Capitalize first letter of each word
    short_title = ' '.join(key_components + mandatory_parts).title()

    # Remove any duplicate hyphens
    short_title = re.sub(r'-\s*-', '-', short_title)

        # Ensure it's within 30-50 characters
    # Ensure colors and last two words are retained
    if len(short_title) > 50:
        # Calculate the remaining length after mandatory parts
        mandatory_length = sum(len(part) for part in mandatory_parts) + len(mandatory_parts) - 1
        remaining_length = 50 - mandatory_length

        # Build the trimmed title
        trimmed_title = ''
        for word in key_components:
            if len(trimmed_title) + len(word) + 1 <= remaining_length:
                trimmed_title += (word + ' ')
            else:
                break

        # Combine the trimmed title with mandatory parts
        short_title = (trimmed_title.strip() + ' ' + ' '.join(mandatory_parts)).strip().title()

    return short_title

# Apply the updated function to create the 'short_title' column
data['short_title'] = data['title'].apply(create_formatted_short_title)
```

The provided code is designed to generate a concise and SEO-friendly version of product titles, referred to as the `short_title`, while retaining essential product information. It begins by previewing the first 100 rows of the dataset using `data.head(100)` to understand the structure of the titles. The core function, `create_formatted_short_title()`, processes each product title with specific logic. If the original title is less than 50 characters, it simply formats it in title case. For longer titles, the code performs text cleaning by removing unnecessary characters such as brackets and underscores and replacing them with spaces. It identifies and retains color names from a predefined list, as colors often hold key product information.

Additionally, redundant phrases (e.g., "includes," "set of," "features") are removed to declutter the title. The function then extracts the first five significant words, ensures the inclusion of identified colors, and appends the last two words of the original title, separated by a hyphen, as these typically contain important product details like size or model. To maintain readability, duplicate hyphens are eliminated. Lastly, if the short title exceeds 50 characters, it trims the title without cutting words midway, ensuring the color and last two words are always preserved. The function is applied to the dataset using `apply()`, creating a new column, `short_title`, with the optimized product titles.

Visualization

To gain further insights into the distribution of title lengths before and after optimization, the following Python code was used:

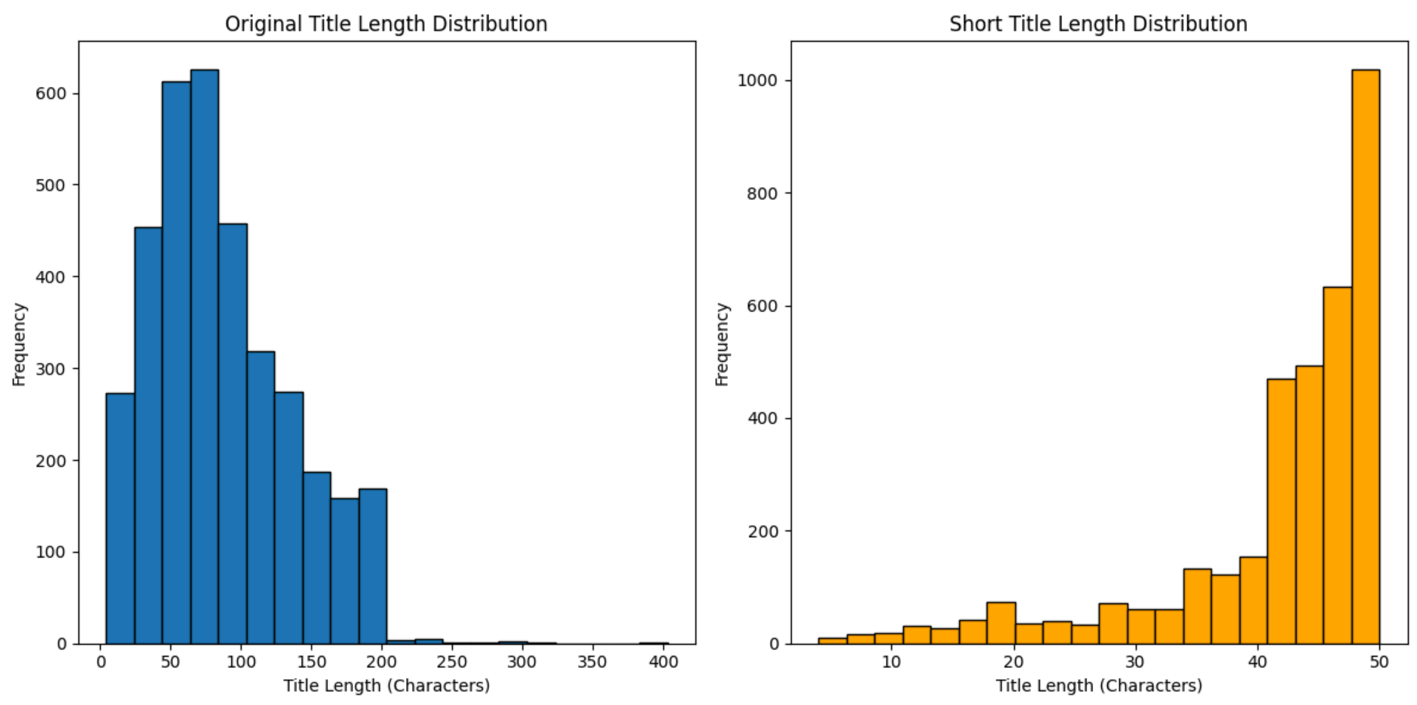
```
# Title Length Distribution Visualization
    plt.figure(figsize=(12, 6))

    # Original Title Length
    plt.subplot(1, 2, 1)
    data['original_title_length'] = data['title'].apply(len)
    plt.hist(data['original_title_length'], bins=20, edgecolor='black')
    plt.title('Original Title Length Distribution')
    plt.xlabel('Title Length (Characters)')
    plt.ylabel('Frequency')

    # Short Title Length
    plt.subplot(1, 2, 2)
    data['short_title_length'] = data['short_title'].apply(len)
    plt.hist(data['short_title_length'], bins=20, color='orange', edgecolor='black')
    plt.title('Short Title Length Distribution')
    plt.xlabel('Title Length (Characters)')
    plt.ylabel('Frequency')

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

This code provides a side-by-side histogram comparison of the original and short title lengths. It helps visualize how the optimization process impacts title length distribution, offering insights into the consistency and effectiveness of the `short_title` generation logic.



Exporting the Data

After generating the short titles, the following code was used to export the dataset to an Excel file:

```
output_file = 'Product Data with Short Title.xlsx'
data.to_excel(output_file, index=False)
```

The methodology effectively condensed long product titles into concise, informative short titles optimized for SEO and user readability. The inclusion of colors, critical product details, and a clean, structured format ensured that the short titles retained essential information while enhancing their marketing effectiveness.

Recommendations

Based on the data cleaning process, the following recommendations are suggested to maintain data quality in the future:

- **Automate Data Cleaning Processes:**

  Implement automated scripts to handle common data quality issues such as duplicates, missing values, and encoding inconsistencies. Automation reduces manual effort, minimizes human error, and ensures data remains clean and reliable in real-time.

- **Standardized Data Entry Practices:**

  Enforce standardized data entry protocols, particularly for critical fields like `product_type_id` and `product_length`. This includes predefined formats, mandatory fields, and drop-down selections to reduce inconsistencies at the source.

- **SEO Performance Monitoring for Short Titles:**

  Regularly track the performance of the `short_title` feature in marketing campaigns. Utilize A/B testing to evaluate the effectiveness of different title formats, optimizing for higher search engine rankings and improved click-through rates.

- **Data Validation Pipelines:**

  Establish robust data validation pipelines that automatically check for data anomalies, missing values, and format discrepancies before data is integrated into the system. This proactive approach helps catch errors early, maintaining data integrity.

- **Regular Data Audits and Feedback Loops:**

  Conduct periodic data audits to identify and rectify emerging data quality issues. Additionally, implement feedback loops from marketing analytics to continuously refine data cleaning strategies based on real-world performance insights.

Conclusion

The data cleaning and preparation process significantly improved the dataset's quality, ensuring consistency, accuracy, and reliability. By addressing critical issues such as missing values, duplicate entries, encoding inconsistencies, and poorly formatted text, the dataset is now well-structured and optimized for analysis. Additionally, the introduction of the `short_title` feature has enhanced the dataset's marketing value, making product titles more SEO-friendly and user-centric.

This refined dataset not only supports data-driven decision-making but also lays a strong foundation for advanced analytical tasks, such as predictive modeling, customer segmentation, and trend analysis. Going forward, implementing the recommended data governance practices will help maintain high data quality standards, ensuring that the organization continues to derive meaningful insights and achieve better marketing outcomes.