The Essence of Luxury: Analyzing Perception, Appearance, Relevant Entities, and Fashion Preferences for Louis Vuitton, Chanel, Dior, and Versace according to Top Fashion Magazines

1. Purpose, Objectives, and Background

1.1 Introduction

Fashion has evolved into an indispensable aspect of modern life. However, the "State of Fashion 2023" report by McKinsey [1] underscores a challenging landscape for the industry this year. Macro-economic and social factors, such as slowing growth, rising inflation, and geopolitical tensions, have significantly impacted the sector.

The year 2024 are fraught with uncertainties for fashion industry leaders who foresee moderate economic growth, persistent inflation, and weakened consumer confidence. McKinsey forecasts a revenue growth of 2 to 4 percent in the global fashion industry in 2024 [1], with the luxury segment being the primary profit generator. However, even this sector will encounter challenges within the complex economic environment.

Amidst this uncertainty, many fashion companies seek valuable insights to enhance innovation, marketing strategies and unlocking new performance drivers. [2] Many current of theses approaches are focus on research to understand consumer behaviour, critics and industry trends.

Traditionally, surveys were the norm, but proved limiting due to time constraints and potential deviations from the intended focus. Consequently, computational methods such as Natural Language Processing (NLP) and text mining have gained prominence. Researchers like Chunmin Lang, Muzhen Li, and Li Zhao (2020) [3] utilized text mining to analyze online fashion rental experiences, discovering that words like "easy," "use," "special," and "highly" appeared more frequently in positive reviews. Similarly, Heo and Lee (2019) [4] employed sentiment analysis to understand Gucci brand trends, revealing an increased brand appreciation following Alexandre Michele's assumption of the creative director role in 2015.

Given this scenario, the initiative arises to apply these computational methodologies to four prominent luxury brands in 2023: Louis Vuitton, Chanel, Dior, and Versace. To achieve this, a variety of fashion articles from the current year have been compiled from influential and prestigious sources such as VOGUE, ELLE, HARPERSBAZAAR, and GLAMOUR. This approach will provide us with valuable insights exclusively focused on these brands, encompassing aspects such as their reputation, trends in the public discourse within the fashion expert community, and, ultimately, granting us a deeper understanding of emotional critiques and shared perceptions by this specialized audience in relation to these brands.

This analysis will provide essential insights to explore new ways of forecasting trends and assess critical points within the fashion industry.

1.2 Objectives and Research Questions

1.2.1 Objectives

- **Reputation analysis:**
  Explore and evaluate the reputation of each brand individually based on the collected fashion articles.
- **Trend identification:**
  Independently identify and analyze emerging trends in the public discourse within the fashion expert community for each brand.
- **Understanding emotional critiques:**
  Obtain a deeper understanding of emotional critiques expressed in the articles and the perceptions shared by the specialized audience for each brand.

1.2.2 Research Questions

> A Note to the Reader:
> In crafting our research questions, we have employed plain language without computational jargon. This is intended to facilitate understanding for those responsible for disseminating information, such as communicators, journalists, fashion influencers, among others. Technical details on how each question will be addressed are provided after each question, maintaining simplicity in presentation.

- **For reputation analysis**:
  Research Question 1: What are the key factors contributing to the reputation of Louis Vuitton, Chanel, Dior, and Versace in 2023 fashion articles? *Focus: Identify the most frequent and positive words or terms associated with each brand, providing insights into highlighted aspects contributing to their reputation.*
  Research Question 2: How do influential fashion sources perceive and portray the reputation of each brand? *Focus: Evaluate mentions and adjectives used by influential sources when referring to each brand, considering the frequency of positive and negative adjectives.*

- **For trend identification:**
  Research Question 3: What emerging trends are present in discussions about Louis Vuitton, Chanel, Dior, and Versace within the fashion expert community? *Focus: Identify keywords related to trends and analyze their frequency in articles.*
  Research Question 4: Are there specific themes or styles dominating discussions about each brand? *Focus: Conduct a thematic analysis and identify specific terms associated with each brand, revealing standout styles or characteristics.*

- **For understanding emotional critique:**
  Research Question 5: How do fashion articles express emotional critiques towards Louis Vuitton, Chanel, Dior, and Versace? *Focus: Utilize sentiment analysis to categorize critiques as positive, negative, or neutral, providing insights into emotional perception.*
  Research Question 6: What emotions are commonly associated with each brand, and how do these evolve throughout 2023? *Focus: Conduct sentiment analysis over time to observe changes in emotions associated with each brand.*

- **Comparative analysis between luxury brands:** Following individual analyses, the project will conclude with a comprehensive comparative analysis between brands.
  Research Question 7: What are the similarities and differences in reputation, trends, and emotional responses between Louis Vuitton, Chanel, Dior, and Versace when collectively analyzed? *Focus: Conduct direct comparisons of keywords, themes, and sentiments associated with each brand to identify overall patterns and specific distinctions.*

1.2.3 Personal objectives for successful analysis

(1) Clear understanding of constraints: It is essential to have a clear understanding of temporal and resource limitations. This will enable the establishment of realistic expectations and determine the maximum number of articles that can be collected for each brand.
(2) Data collection through Web Scraping: Data collection will be executed through web scraping techniques to extract publicly available posts containing the keywords "Louis Vuitton," "Chanel," "Versace," and "Dior."
(3) Crucial data cleaning operations: Data cleaning is crucial and involves various operations, such as the removal of invalid data, addition of missing data, and decomposition and reorganization of information. This process will ensure data quality and consistency by eliminating redundancies, noise, and errors.
(4) Focus on processing and analysis: After data collection and cleaning, the focus will shift to processing and analysis. Techniques such as text mining and sentiment analysis will be employed to address the previously established research questions, providing valuable insights.
(5) Contextualization for non-technical audiences: It is crucial not only to present data but also to contextualize the results in understandable terms for those without technical expertise. This approach facilitates the interpretation and utilization of findings by communicators, journalists, and other professionals not specialized in data analysis.

1.3 Data

1.3.1 A brief history of the digital Fashion Revolution

In the centuries of absolute monarchy, fashion was intrinsically linked to the power hierarchy, reserved for the nobility and dictated within the court [5]. However, the first sixty years of the 18th century witnessed the advent of a "consumer revolution", where an increasing number of individuals, with purchasing power, immersed themselves in the variety of buying and selling products, giving rise to large-scale production [6].

The democratization of luxury became both a cause and an effect of this production. Fashion transformed into a guiding device, disciplining consumers and creating markets for products [7].

The commercialization of women's fashion played a crucial role as well. The imitation of aristocratic attire became accessible to a broad market thanks to fashion magazines, which multiplied in the 18th century. By the end of the century, dozens of fashion almanacs and yearbooks existed in England, showcasing the growing influence of these publications [8].

Over time, fashion magazines played a pivotal role in diversifying knowledge about trends and styles. As technology immersed itself in our lives, these magazines transitioned from physical to digital. The shift to the digital era allowed faster and broader access to fashion articles, magazines, and content, becoming an omnipresent influence in contemporary society. Fashion, now just a click away, has solidified its position as a pervasive element in everyone's daily life. For instance, many luxury brands often expand and gain more recognition through articles or features in influential fashion websites [9].

1.3.2 Selection of Magazines and Articles

Recognizing the profound influence that fashion magazines wield in disseminating trends, styles, and constructing the reputation of luxury brands, four of the most influential platforms in today's fashion scene were selected for article compilation:

- **VOGUE**, founded in 1892, stands as an influential fashion and lifestyle magazine that has transcended borders to become a global icon. A part of the Conde Nast publishing empire, VOGUE boasts an extensive history, a presence in multiple international editions, and collaborations with prominent figures in fashion and photography [10]. Its ability to dictate trends, elevate emerging designers, and maintain high-quality standards gives it a distinctive status in the fashion industry, making it an irreplaceable reference for enthusiasts and professionals worldwide.
- **ELLE**, a women's fashion magazine founded in France in 1945, has gained global recognition. As part of the Lagardère group, ELLE is distinguished for its fresh and contemporary approach to covering fashion, beauty, culture, and celebrities [11]. With editions in numerous countries, ELLE positions itself as an influential voice addressing both global and local trends.
- **GLAMOUR**, founded in 1939 as "Glamour of Hollywood," part of the Condé Nast family, has evolved into a magazine that highlights fashion, beauty, and celebrities. With over 1.4 million readers in the United States and more than 988 thousand readers worldwide [n], GLAMOUR has adapted its content to reflect changing trends and contemporary culture in the realm of women's fashion [12].
- **Harper's Bazaar**, founded in 1862 by Harper & Brothers, has been a fashion magazine for over a century, serving as the first fashion magazine in the United States. Considered one of the world's most important fashion magazines, with over 35 editions and available in 14 different languages across 100 countries, including Vietnam [13], it has significantly influenced the development of the fashion magazine industry and continues to set trends in modern fashion.

The selected articles for this analysis are those that address, critique, and comment on four of the most prestigious and renowned brands in the world of fashion:

- **Louis Vuitton**, founded in Paris, France, in 1854, is renowned for its luxury products, especially its iconic bags and leather goods. Active participation in fashion events and the implementation of global marketing strategies further contribute to its influential position in the industry. The brand distinguishes itself by maintaining an aura of exclusivity, releasing limited collections that generate fervent demand and solidify its status as a trendsetter in the contemporary fashion world. Recently, Louis Vuitton became the first European company to surpass $500 billion in market value [14], thanks to a significant increase in sales and stock prices.
- **Chanel**, the iconic fashion house founded by Gabrielle "Coco" Chanel in 1910, stands as a significant influence in the industry. Recognized for its timeless elegance, Chanel has set standards for how women dress and perceive themselves. Gabrielle 'Coco' Chanel introduced an unexplored concept of femininity and confidence in her time, where fashion became an ally to liberate women from conventional dress norms [15]. Currently, her legacy endures as a beacon of style and elegance, consolidating Chanel as one of the most influential brands in the fashion world with over 300 locations worldwide.
- **Dior**, a French fashion house founded by designer Christian Dior in 1946, is known for its elegant designs and refined femininity. Dior has significantly influenced women's fashion, with its historic "New Look" introduced in 1947 marking a transcendental turn, reintroducing femininity and opulence in the post-war era [16]. Its iconic haute couture dresses, coveted bags, and constant presence in international fashion consolidate Dior's lasting influence, remaining a symbol of elegance and style.
- **Versace**, the iconic Italian fashion house founded by Gianni Versace in 1978, is recognized for its bold and luxurious aesthetic. Celebrating extravagance and individuality over the years, Versace has stood out for its vibrant designs, striking prints, and creative use of high-quality materials [17]. Today, the brand has maintained its relevance in the fashion industry, with collections that continue to make noise and captivate attention.

1.3.3 Criteria for Data

As mentioned earlier, only four fashion magazines were selected, acknowledging the inherent limitations of time and resources. From each of these magazines, four articles per luxury brand were gathered. This limited selection, as articles from these magazines tend to be both concise and comprehensive, provides a balanced approach for the research. The evaluation of essential parameters in the compilation of each article was conducted to ensure its suitability and alignment with the analysis objectives. This meticulous approach will facilitate subsequent analysis using advanced Natural Language Processing (NLP) techniques.

Method: A temporality criterion was established, focusing on articles published from January 2023 onwards that incorporate the keywords ("Louis Vuitton," "Chanel," "Dior," and "Versace"). From the entirety of the collected articles for each brand in each magazine, a careful selection of the top 4 most relevant pieces was made. To ensure efficiency in data compilation, specific criteria were employed using web scraping methods. These methods involved identifying and extracting articles that contained the word of the selected luxury brand ("Louis Vuitton", "Chanel", "Dior" and "Versace") at least three times in each articles (i.e. if the article collected was from Louis Vuitton for example, it had to have the word "Louis Vuitton" at least three times). This systematic approach guarantees an in-depth exploration of each brand, aligning seamlessly with the analysis objectives. Furthermore, article length is considered an indicator of substantive content, with a focus on prioritizing those offering extensive and detailed coverage.

1.3.4 Limitations of the data

(1) The reputation of a brand can be significantly influenced by specific events or significant changes in the industry. The decision to limit the analysis to the actual year may not capture far-reaching events that potentially have a lasting impact on brand perception. However, it is crucial to acknowledge the inherent limitations related to the time and resources available.

(2) Although the plan is to analyze 16 articles per brand from the last 11 months, it is essential to note that these selected 16 articles may not necessarily share the same publication date, as the choice of the most relevant ones involves a random selection of dates.

(3) Despite the professional rigor in crafting the selected articles, it is imperative to recognize the possibility of biases that may reflect the individual perspectives of experts in the field, including potential political influences and other subjective factors.

(4) The focused selection of English-language journals implies an intrinsic limitation by excluding potential sources of information in other languages. This could result in a partial and biased representation of the diversity of perspectives globally.

(5) The analysis exclusively focuses on the interpretation of articles, deliberately excluding the consideration of ratings and comments present on social media platforms or other external sources.

1.4 Ethical considerations

1.4.1 License to Access and Use

Web scraping was used for data collection, including a detailed analysis of the robots.txt files of each of the selected websites. The robots.txt file serves as a tool allowing website owners to communicate guidelines to web crawlers regarding which areas of the site can be crawled and which should be avoided. In this context, a thorough review of each clause in the robots.txt files of the selected sites was carried out, ensuring strict compliance with specified restrictions and rules. This approach was adopted to guarantee that the web scraping process was ethical, legal, and respectful of the policies set by each platform, avoiding any infringement or misunderstanding in accessing information.

VOGUE
Webscraping: Not mentioned. Content: Prohibited without permissions. Attempts were made to contact VOGUE during the development of the project, but no response was obtained. While the site's terms and conditions expressly forbid certain types of access and automated use, it is crucial to highlight that the collected content will be used exclusively for personal, educational, and non-commercial purposes. Moreover, the analysis focuses on understanding and evaluating the perception of luxury brands, with no intention to violate the site's terms and conditions.

ELLE
Webscraping: Not mentioned. Content: Personal use allowed. Accessing, viewing, printing individual pages, and storing them electronically is permitted for personal, non-commercial use only. Furthermore, the use must be ethical, respecting the commitment not to employ this content accompanied by any obscene, indecent, or offensive language, defamatory, abusive, harassing, or hateful.

GLAMOUR
Webscraping: Not mentioned. Content: Access and storage of the service or any of its content are allowed for personal, non-commercial use. Additionally, the user is solely responsible for all activities conducted on the service by oneself or those authorized or permitted to use it.

HARPER'S BAZAAR
Webscraping: Not mentioned. Content: Prohibits the use of content to develop any software program, including training machine learning or artificial intelligence (AI) systems, without prior permission. HYPER'S BAZAR was therefore contacted and permission was obtained to use the content for educational and personal purposes only.

1.4.2 Subsequent Use of Extracted Data

In the context of the subsequent use or reuse of the collected data, careful consideration of the established conditions and restrictions detailed in section *1.4.3 of Ethical Considerations* is emphasized.

(1) Any text extracted from the websites present in this project intended for reuse, must undergo a thorough analysis of these ethical policies to ensure conformity and coherence with the established principles.

(2) In situations where prior considerations specify that access or use of content requires specific permissions, the imperative nature of obtaining the corresponding authorization before proceeding with any additional application or reuse of the data is highlighted.

(3) In cases where access to content is subject to particular conditions, such as limited use for personal and non-commercial purposes, the importance of respecting and strictly adhering to these restrictions is underscored.

It is crucial to note that the conditions and restrictions set by websites may be continuously modified and updated. In this regard, it is warned that the ethical considerations mentioned above might become outdated over time, emphasizing the need for constant review and updating of relevant policies and regulations.

1.4.3 Post-Analysis Impacts

This project focuses on presenting a comprehensive analysis using various text mining and natural language processing techniques. Therefore, it is crucial to recognize that this analysis does not aim to definitively determine the exact trends, forecasts, or reputations of the fashion brands in question. Instead, its purpose is to offer a detailed perspective based on the collected and analyzed data. In this regard, it is vital to consider the intrinsic limitations of the study mentioned in Section *1.3.4 Limitations of the Data*, which may introduce biases in representing perceived realities in the fashion and luxury brand domain.

Hence, this analysis should be interpreted with caution, acknowledging its ability to provide valuable insights while maintaining an awareness of its restrictions and limitations in representing the entirety of the landscape.

In [1]:
```
#Import necessary libraries and modules
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
import re
import nltk
from nltk import pos_tag
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk import ne_chunk
from collections import Counter
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
import matplotlib.pyplot as plt
import PIL.Image
import plotly.express as px
import plotly.graph_objects as go
```

2. Data Verification

Subsequent to the extraction and organization of the selected articles for each brand (Louis Vuitton, Chanel, Dior, Versacce) from chosen websites (Vogue, Glamour, Elle, Harper's Bazar), a process was employed to structure this information into a CSV file. This CSV file encompasses a detailed overview of the collected articles, systematically cataloging essential attributes such as the source website from which each article originated, the corresponding luxury brand associated with the article, along with details such as the article's title and content. The creation of this file adhered rigorously to the parameters outlined in Section "1.3.3 Criteria for Data." However, ensuring the integrity and adherence of the collected data to the established parameters is paramount.

In [2]:
```
path = "selectedFinalArticles.csv"
df_articles = pd.read_csv(path)
```

In [3]:
```
#Veryfing no missing data
missingData = False
for col in df_articles.columns:
    missing_count = df_articles[col].isna().sum()
```

```
    if missing_count > 0:
        print(f"Colum '{col}' contains missing data")
        missingData = True
        break
print("No missing data found") if missingData == False else None
```

```
No missing data found
```

The first step in the data verification process involves ensuring that there are between 3 to 4 articles available for each luxury brand (Louis Vuitton, Chanel, Dior, Versace) from each selected website (Vogue, Glamour, Elle, Harper's Bazaar).

In [4]:
```
# Verified that there are only four distinct websites.
df_articles["website"].unique().tolist()
```

Out[4]:
```
['vogue', 'glamour', 'elle', "harper's bazar"]
```

In [5]:
```
# Verified that there are only four distinct brands.
df_articles["brand"].unique().tolist()
```

Out[5]:
```
['louis vuitton', 'chanel', 'dior', 'versace']
```

In [6]:
```
# Group by website and brand, and count the articles
df_group = df_articles.groupby(
    ["website", "brand"]).size().reset_index(name='article_count')

# Check if the number of articles any website and brand combination is 4.
# Assume initially that all articles meet the condition
minimum_articles = True
for _, row in df_group.iterrows():
    if row["article_count"] < 3:
        print("Website '{}' has only {} article(s) for the brand '{}', \
        but it should have 4 articles.".format(row['website'], row['article_count'], row['brand']))
        minimum_articles = False
        break
print("4 articles for each brand for each website") if minimum_articles else None
```

```
4 articles for each brand for each website
```

2.2 Verification of Brand Keyword Frequency

The second stage of data verification entails confirming that each article contains the brand's name attributed to it at least four times. This scrutiny aims to ensure that the selected articles indeed focus substantially on the specified luxury brands (Louis Vuitton, Chanel, Dior, Versace).

In [7]:
```
df_articles_frequency = df_articles.copy()
```

In [8]:
```
# Assume initially that all articles meet the condition
all_articles_contain_brand = True

for index, row in df_articles_frequency.iterrows():
    brand_count = len(re.findall("|".join(["louis vuitton", "LV", "LVMH"]) if row["brand"] =="louis vuitton" else row["brand"],
                                 row["content"],
                                 re.IGNORECASE))

    if brand_count <= 2:
        print("Article {}: Brand '{}' appears less than 2 times.".format(index, row['brand']))
        # Update the flag if any article doesn't meet the condition
        all_articles_contain_brand = False
        break

# Print the message after the loop based on the flag
print("All articles contain the brand more than 2 times.") if all_articles_contain_brand else None
```

```
All articles contain the brand more than 2 times.
```

2.3 Verification of Article Length

The third stage of data verification involves confirming that the length of each article surpasses 200 words. This criterion ensures that the selected articles provide substantive content for subsequent analysis.

In [9]:
```
# Add a new column to the DataFrame to store the length (number of words) of each article content.
df_articles["Content length"] = df_articles["content"].apply(
    lambda x: len(x.split()))
```

In [10]:
```
df_articles["Content length"].describe()
```

Out[10]:
```
count      64.000000
mean      434.875000
std       224.935934
min       168.000000
25%       280.750000
50%       345.500000
75%       542.000000
max      1097.000000
Name: Content length, dtype: float64
```

In [11]:
```
# Assume initially that all articles meet the condition
all_articles_length = True

for index, row in df_articles.iterrows():
    if row["Content length"] < 150:
        print("Article {}: contains less than 100 words.".format(index))
        all_articles_length = False
    elif row["Content length"] > 1100:
        print("Article {}: contains more than 1500 words.".format(index))
        all_articles_length = False

# Print the message after the loop based on the flag
print("All articles contain between 100 and 1100 words.") if all_articles_length else None
```

```
All articles contain between 100 and 1100 words.
```

3. Organizing, Cleaning, and Processing Data

The current state of the extracted data lacks the necessary organization for a precise and insightful analysis. Additionally, inherent anomalies within the dataset require meticulous cleansing to ensure the derivation of high-quality and reliable results. Data quality is paramount for deriving meaningful insights, and the initial raw data must undergo a thorough cleansing process to rectify any inconsistencies or inaccuracies.

- **Data organization** involves the strategic structuring and categorization of information to optimize efficiency and usability. This strategic arrangement is crucial for maximizing the utility of the dataset in subsequent analyses.
- **Data cleansing and processing** represent critical operations aimed at eliminating anomalies and deriving an accurate and unique representation of the underlying information. These processes are fundamental to ensuring the integrity of the data, especially given that subsequent stages involve advanced Natural Language Processing (NLP) analyses and graphical comparisons between brands.

This phase assumes paramount importance, as the subsequent analytical stages heavily rely on the organized and cleaned dataset.

3.1 Data organization

(1) Creation of Corpora:
To enhance the analytical precision of our data, corpora will be established for each brand associated with every website. Each corpus will comprehensively encompass all selected articles and their respective titles. This process will yield four distinct corpora for each website, each specifically dedicated to a brand.

In [12]:
```
def createListBrand(brand,list_of_websites):
    """
    Create a list of dictionaries, where each dictionary represents a corpus associated with the specified brand
    for a different website.


    Parameters:
    - brand (str): The brand for which corpora are created.
    - list_of_websites (list): List of websites for which corpora are created.

    Returns:
    - completeList (list): List of 4 dictionaries, each containing brand (the same for the 4 dicts), website,
    title corpus, and article corpus.
    """
    completeList= []

    for website in list_of_websites:
        # Filter to include only articles related to the specified brand and website
        con = (df_articles["website"] == website) & (
            df_articles["brand"] == brand)
        df_articles_per_brand = df_articles[con]
        # print(df_articles_per_brand)

        # Initialize a dictionary to store the corpus information
        brandDict = {}

        # Concatenate all titles and articles into respective corpora
        all_titles = " ".join(df_articles_per_brand["title"])
        all_articles = " ".join(df_articles_per_brand["content"])

        # print(len(all_titles.split()))
        # print(len(all_articles.split()))

        # Populate the dictionary with brand, website, title corpus, and article corpus
        brandDict["brand"] = brand
        brandDict["website"] = website
        brandDict["title_corpus"] = all_titles
        brandDict["article_corpus"] = all_articles

        # Append the dictionary to the complete list
        completeList.append(brandDict)

    return completeList
```

(2) DataFrame Construction for Each Brand:
Now, dedicated DataFrames will be formulated for each brand. As a result, each DataFrame will consolidate four corpora, each associated with the same brand but sourced from distinct websites.

In [13]:
```
def createDataframeBrand(brand):
    list_of_websites = ["vogue", "glamour", "elle", "harper's bazar"]
    # call to "createListBrand()" to retrieve the list of dictionaries
    final_list_of_corpus = createListBrand(brand, list_of_websites)
    # Create a DataFrame from the list of dictionaries
    df_brand = pd.DataFrame(final_list_of_corpus)
    return df_brand


df_louisVuitton = createDataframeBrand("louis vuitton")  # for louis vuitton
df_chanel = createDataframeBrand("chanel")  # for chanel
df_dior = createDataframeBrand("dior")  # for dior
df_versace = createDataframeBrand("versace")  # for versace
```

> Unit testing was applied to ensure that the dataframes meet the expected standards. The set of unit tests provided checks crucial aspects such as column names, the uniqueness of brand values, and the presence of specific website values in each dataframe. Running these tests allows us to confidently ascertain that the dataframes are correctly structured.

In [14]:
```
# Importing the unittest module for conducting unit tests
import unittest
class MyTestDF(unittest.TestCase):
    # List of dataframes for different brands
        list_df_brands = [df_louisVuitton, df_chanel, df_dior, df_versace]

        def test_column_names(self):
            # Test to ensure that the column names of each dataframe match the expected names
            for db in self.list_df_brands:
                self.assertEqual(
                    [col for col in db.columns],
                    ['brand','website','title_corpus','article_corpus']
                )

        def test_unique_brand(self):
            # Test to ensure that the "brand" column contains only one unique value in each dataframe
            for db in self.list_df_brands:
                self.assertEqual(len(db["brand"].unique()),1)

        def test_dif_websites(self):
            # Test to ensure that the "website" column contains the expected unique values in each dataframe
            for db in self.list_df_brands:
                self.assertEqual(
                    [w for w in db["website"].unique()],
```

```
                ["vogue", "glamour", "elle", "harper's bazar"]
            )


unittest.main(argv=["ignored", "-v"], exit=False)
```

```
test_column_names (__main__.MyTestDF.test_column_names) ... ok
test_dif_websites (__main__.MyTestDF.test_dif_websites) ... ok
test_unique_brand (__main__.MyTestDF.test_unique_brand) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.006s

OK
```

Out[14]: <unittest.main.TestProgram at 0x26e8dbeab50>

In [15]:
```
#Some information about each DataFrame

# List of dataframes for different brands
list_df_brands = [df_louisVuitton, df_chanel, df_dior, df_versace]
# Names corresponding to each dataframe
name_df_brands = ["df_louisVuitton", "df_chanel", "df_dior", "df_versace"]

# Iterate through each dataframe and its corresponding name
for db, nameDF in zip(list_df_brands, name_df_brands):
    # Display the name of the dataframe
    nameDF = f"Dataframe: {nameDF}"
    # Display the unique brand associated with the dataframe
    brand = f"Brand: {db['brand'].unique()[0]}"
    print(nameDF)
    print(brand)

    # Iterate through each row in the dataframe
    for index_row in range(0, len(db)):
        # Display information about the corpus on each website
        mes = "Website {} has a total corpus of {} words".format(db['website'][index_row],
                                                                 len(db['article_corpus'][index_row].split()))
        print(mes)
    print()
```

```
Dataframe: df_louisVuitton
Brand: louis vuitton
Website vogue has a total corpus of 2061 words
Website glamour has a total corpus of 1405 words
Website elle has a total corpus of 1980 words
Website harper's bazar has a total corpus of 2567 words

Dataframe: df_chanel
Brand: chanel
Website vogue has a total corpus of 1395 words
Website glamour has a total corpus of 1195 words
Website elle has a total corpus of 2178 words
Website harper's bazar has a total corpus of 1509 words

Dataframe: df_dior
Brand: dior
Website vogue has a total corpus of 1700 words
Website glamour has a total corpus of 1474 words
Website elle has a total corpus of 1802 words
Website harper's bazar has a total corpus of 2904 words

Dataframe: df_versace
Brand: versace
Website vogue has a total corpus of 1617 words
Website glamour has a total corpus of 1062 words
Website elle has a total corpus of 1421 words
Website harper's bazar has a total corpus of 1562 words
```

3.2 Data Cleaning and Processing

(3) Identification and Cleansing of anomalies: This phase involves a comprehensive examination of the data to detect and rectify anomalies. Anomalies may encompass special characters, unnecessary spaces, or any irregularities that could potentially compromise the quality of the data.

In [16]:
```
def cleanCOR(corpus, lower):
    """
    Cleans the given corpus by removing special characters, unnecessary spaces,
    and converting all words to lowercase.

    Parameters:
    - corpus (str): The input text corpus to be cleaned.
    - lower(bool): all words in the returned corpus in lowercase or not

    Returns:
    str: The cleaned corpus.
    """

    # Define a regular expression pattern to match non-alphanumeric characters
    pattern = re.compile('[^a-zA-Z\']')
    # Remove non-alphanumeric characters, extra spaces, and convert to lowercase
    cleaned_corpus = re.sub(pattern, ' ', corpus)
    cleaned_corpus = ' '.join(cleaned_corpus.split())
    cleaned_corpus = cleaned_corpus.lower() if lower else cleaned_corpus
    return cleaned_corpus
```

(4) Identification and Elimination of Stopwords: Another critical aspect of our data processing strategy involves the identification and removal of stopwords. Stopwords, common words devoid of significant meaning, can introduce noise and hinder the effectiveness of our subsequent analyses.

In [17]:
```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\aloza\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\aloza\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\aloza\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[17]:
```
True
```

In [18]:
```python
# 1 identyfing stop words not in NLTK stopwords

#type of word
def get_word_pos(word):
    """
    Retrieves the part-of-speech tag for a given word using the NLTK library.

    Parameters:
    - word (str): The input word.

    Returns:
    str: The part-of-speech tag.
    """
    # Use NLTK pos_tag to get the part-of-speech tag
    tagged_word = pos_tag(word_tokenize(word))
    return tagged_word[0][1] if tagged_word else None

def get_freq_words(df, columnName,lower=True):
    """
    Extracts and analyzes the frequency of non-stopwords from a DataFrame's specific column.

    Parameters:
    - df (DataFrame): The input DataFrame containing the "article_corpus" column.
    - columnName (str): name of the column for which the frequencies are obtained
    -lower (optional)(bool): If True, the words in the returned dictionary will be in lowercase.
    Defaults to True.

    Returns:
    dict: A dictionary of non-adjective words sorted by frequency across all corpora.
    """

    all_final_words = {}

    # Iterate over each corpus in the DataFrame
    for corpus in df[columnName]:
        # Clean the corpus and split it into a list of words
        words_corpus_list = cleanCOR(corpus, lower).split()
        # Extract unique words and filter out stopwords
        unique_words = set(words_corpus_list)
        selected_nonStopW = [word for word in unique_words if not word in stopwords.words('english')]

        # Count the frequency of each unique word
        countUniqueWords = [
            {"Word": word, "Frequency": words_corpus_list.count(word)} for word in selected_nonStopW]

        for i in range(len(countUniqueWords)):
            # Filter out non-adjective words since they're often more relevant for identifying stopwords
            if get_word_pos(countUniqueWords[i]["Word"]) != 'JJ':
                word = countUniqueWords[i]["Word"]
                all_final_words[word] = all_final_words.get(
                    word, 0) + words_corpus_list.count(word)

        # Sort words by frequency in descending order
        sorted_countUniqueW = dict(
            sorted(all_final_words.items(), key=lambda item: item[1], reverse=True))
    return sorted_countUniqueW

for dataFrame, nameDF in zip(list_df_brands,name_df_brands):
    print(f"Length of final selected words for {nameDF}:", len(get_freq_words(dataFrame,"article_corpus")))
```

```
Length of final selected words for df_louisVuitton: 2027
Length of final selected words for df_chanel: 1679
Length of final selected words for df_dior: 2282
Length of final selected words for df_versace: 1500
```

Upon initial exploration, it became evident that the sheer volume of returned words from the corpus analysis is extensive. To streamline the subsequent analysis, a meticulous approach is warranted. The strategy involves a manual examination of the *top 100 most frequent words* extracted from `get_freq_words()` for each DataFrame. This deliberate selection will enable the identification of words that may not contribute significantly. Subsequently, the chosen words from this manual analysis will be designated as **additional stopwords**.

In [19]:
```python
freq_words_LV = list(get_freq_words(df_louisVuitton, "article_corpus").items())[:100] # <- print()
additionalStopWords_LV = ["de", "one", "also", "that's", "i'm", "done", "show", "jacobs", "jacobs's",
                          "dillane", "dillane's", "us", "globes", "john", "armas's", "ceo", "abloh's", "a", "like"]
```

In [20]:
```python
freq_words_chanel = list(get_freq_words(df_chanel, "article_corpus").items())[:100]  # <- print()
additionalStopWords_chanel = ["first", "time", "even", "viard", "waves", "los", "we'll","tha's",
                              "there's", "vanessa", "may", "way", "user", "paradis", "show", "a", "like", "d'art"]
```

In [21]:
```python
freq_words_dior = list(get_freq_words(df_dior, "article_corpus").items())[:100]  # <- print()
additionalStopWords_dior = ["jones", "eye", "time", "chiuri", "spring", "shahidi's", "that's",
                            "there's", "back", "front", "baker", "josephine", "eliot's",
                            "pattinson", "christie", "show", "a","like"]
```

In [22]:
```python
freq_words_versace = list(get_freq_words(df_versace, "article_corpus").items())[
    :100]  # <- print()
additionalStopWords_versace = ["los", "angeles", "day", "night", "still", "center", "front",
                               "also", "since", "month", "row", "that's", "there's", "show", "a", "like"]
```

Now that we have meticulously curated the list of stopwords, we are poised to embark on the process of creating filtered corpora. The resulting filtered corpora will exclusively contain the words deemed relevant after the curation process.

In [23]:
```python
# 2 created a filter corpora
def remove_StopWords(corpus,customWords,lower=True):
    """
    Removes stopwords and additional "stopwords" from a given corpus.

    Args:
    - corpus (str): The input corpus to be processed.
    - customWords (list): A list of additional custom words to be removed.
```

```
    -lower (optional)(bool): If True, the words in the returned corpus will be in lowercase.
    Defaults to True.

    Returns:
    - str: The filtered corpus without stopwords and custom words.
    """

    cleaned_corpus = cleanCOR(corpus, lower)
    tokens = word_tokenize(cleaned_corpus)

    # Define a list of common suffixes and additional custom words to remove
    selected_suffixes = ["'s", "n't", "'ve", "'ll", "'re", "'d", "'m", "'t", "th", "'"]
    remove_words = selected_suffixes + customWords + stopwords.words('english')

    # Filter tokens to remove stopwords and custom words
    filtered_tokens = [word for word in tokens if word not in remove_words]
    filtered_corpus = " ".join(filtered_tokens)

    return filtered_corpus
```

In [24]:
```
def addFilterCorporaToBrand(dfName,brand):
    """
    Adds filtered corpus columns to a DataFrame.

    Args:
    - dfName (DataFrame): The input DataFrame.
    - brand (str): The brand associated to the DataFrame.
    """

    # Initialize new columns for filtered corpus and title corpus
    dfName["filtered_title_corpus"] = ""
    dfName["filtered_corpus"] = ""

    brand = ["additionalStopWords_{brand}"]

    # Loop through each row in the DataFrame
    for i in range(len(dfName)):
        # Apply remove_StopWords function to filter title corpus
        dfName.loc[i, "filtered_title_corpus"] = remove_StopWords(
            dfName["title_corpus"][i], brand)

        # Apply remove_StopWords function to filter article corpus
        dfName.loc[i, "filtered_corpus"] = remove_StopWords(
            dfName["article_corpus"][i], brand)


addFilterCorporaToBrand(df_louisVuitton, "LV")
addFilterCorporaToBrand(df_chanel, "Chanel")
addFilterCorporaToBrand(df_dior, "Dior")
addFilterCorporaToBrand(df_versace, "Versace")
```

In [25]:
```
# Add columns of filtered content and title for articles in main DataFrame df_articles
def addFilterCorporaToArticles(dfName):
    dfName["Filtered_title"] = ""
    dfName["Filtered_content"]=""


    for index,row in dfName.iterrows():
        brand, title, content = row["brand"], row["title"], row["content"]
        brand = "LV" if brand == "louis vuitton" else brand.capitalize()
        final_brand = ["additionalStopWords_{brand}"]

        dfName.loc[index, "Filtered_title"] = remove_StopWords(
            title, final_brand)
        dfName.loc[index,"Filtered_content"] = remove_StopWords(
            content, final_brand)


addFilterCorporaToArticles(df_articles)
```

(5) Lemmatizing: Lemmatization is a linguistic technique that allows us to derive a word to its base or lemma form. This approach is crucial for extracting fundamental meanings from the words used in the context. By eliminating redundancies and superficial variations, we concentrate on the semantic essence, providing valuable information for further analysis, such as perceptions of reputation among brands.

In [26]:
```
lemmatizer = WordNetLemmatizer()
def lemma_corpus(corpus):
    """
    Lemmatize each word in the given corpus.

    Parameters:
    - corpus (str): The input text corpus.

    Returns:
    - str: A string containing all the lemmatized words.
    """
    tokens = word_tokenize(corpus)
    lemmatizedWords = " ".join([lemmatizer.lemmatize(word) for word in tokens])
    return lemmatizedWords
```

In [27]:
```
#Add columns of lemmatized title and content for each corpus for each DataFrame:
# df_louisVuitton, df_chanel, df_dior, df_versace
for dfName in list_df_brands:
    dfName["lemmatized_title_corpus"] = dfName["filtered_title_corpus"].apply(
        lemma_corpus)
    dfName["lemmatized_corpus"] = dfName["filtered_corpus"].apply(lemma_corpus)
```

In [28]:
```
# Add columns of lemmatized title and content for articles in main DataFrame df_articles
df_articles["lemmatized_title"] = df_articles["Filtered_title"].apply(
        lemma_corpus)
df_articles["lemmatized_content"] = df_articles["Filtered_content"].apply(
    lemma_corpus)
```

(6) Construction of brand-specific corpora: In order to conduct a more expansive analysis, we will initiate the creation of four final corpora, each dedicated to a specific brand: Louis Vuitton, Chanel, Dior, and Versace. The consolidation of these corpora will be executed by merging text data from individual datasets associated with each brand, such as `df_louisVuitton, df_chanel, df_dior and df_versace`. This approach will enable a more holistic and brand-specific perspective. The amalgamation of content from diverse sources into a single corpus will facilitate deeper and comparative analyses. This could includes examinations of reputation concepts among brands or the frequency of positive and negative adjectives.

In [29]:
```
def globalDataFrame(DFbrands):
    """
```

```
    Creates a global DataFrame containing consolidated corpora for each brand.

    Args:
        DFbrands (list): List of DataFrames.

    Returns:
        pd.DataFrame: DataFrame with columns for brand, titles_corpus_final,
                      articles_corpus_final, filtered_titles_corpus, filtered_articles_corpus,
                      lemmatized_titles_corpus, and lemmatized_articles_corpus.
    """


    # Initialize columns names for the DataFrame
    nameColumns = ["brand","titles_corpus_final",
                   "articles_corpus_final", "filtered_titles_corpus", "filtered_articles_corpus",
                   "lemmatized_titles_corpus", "lemmatized_articles_corpus"]
    df_globalBrands = pd.DataFrame(columns=nameColumns)

    for i in range(len(DFbrands)):

        # Create a dictionary to store corpus information for the current brand
        corpusBrand_dict = {}

        # Consolidate article and title corpora for the current brand
        articles_corpus_final = " ".join(DFbrands[i]["article_corpus"])
        titles_corpus_final = " ".join(DFbrands[i]["title_corpus"])

        # Generate custom stop words specific to each brand
        customWords = additionalStopWords_LV + additionalStopWords_chanel + \
        additionalStopWords_dior + additionalStopWords_versace

        customFinalWords = list(set(customWords))

        # Apply stop word removal to filter out unnecessary words
        filtered_titles_corpus = remove_StopWords(titles_corpus_final,customFinalWords)
        filtered_articles_corpus = remove_StopWords( articles_corpus_final, customFinalWords)

        # Apply lemmatization for further refinement
        lemmatized_titles_corpus = lemma_corpus(filtered_titles_corpus)
        lemmatized_articles_corpus = lemma_corpus(filtered_articles_corpus)

        # Populate the dictionary with corpus information
        corpusBrand_dict["brand"] = DFbrands[i]["brand"].unique()[0]
        corpusBrand_dict["titles_corpus_final"] = titles_corpus_final
        corpusBrand_dict["articles_corpus_final"] = articles_corpus_final
        corpusBrand_dict["filtered_titles_corpus"] = filtered_titles_corpus
        corpusBrand_dict["filtered_articles_corpus"] = filtered_articles_corpus
        corpusBrand_dict["lemmatized_titles_corpus"] = lemmatized_titles_corpus
        corpusBrand_dict["lemmatized_articles_corpus"] = lemmatized_articles_corpus

        # Append the dictionary as a new row to the DataFrame
        df_globalBrands.loc[len(df_globalBrands)] = corpusBrand_dict


    return df_globalBrands

df_globalBrands = globalDataFrame(list_df_brands)
```

4. NLP Analysis of Brand Perception

According to Lakoff (1973) [18], *use of language embodies attitudes as well as referential meanings.*" (1973: 45), and these attitudes become evident through communication analysis.

Fashion, intricately tied to appearance, embodies prestige, value, power, influence, art, feelings, and emotions. As Virginia Woolf illustrates [19]: "*Vain trifles as they seem, clothes have, they say, more important offices than to merely keep us warm. They change our view of the world and the world's view of us.*"

In this light, we assert that fashion, as a cultural phenomenon, many times could be manifested itself through language in various ways, from the careful selection of words in articles to the subtle construction of meanings in narratives denoting specific brands.

This section will delve into various Natural Language Processing (NLP) techniques applied to the fashion articles using NLTK. The focus is to evaluate the perception of each selected brand (Luois Vuitton, Chanel, Dior, Versace), unravel the narrative surrounding it, analyze the prestige it represents, and comprehend the overall impression it conveys.

4.1 Identifying key adjectives and entities

(1) Exploring adjectives associated with the verb 'Look': a pivotal avenue lies in the meticulous exploration of adjectives tied to the verb "look." This verb, representing the very essence of "appearance," has been strategically chosen due to its presence in fashion discourse, especially when articulating nuanced descriptions for fashion items.

For example, in this paragraph, extracted from the Glamour magazine,
`However, the standout feature of Kardashian's look was her pearlescent Chanel clutch bag, a rare item designed by the late, legendary designer Karl Lagerfeld.` Kim Kardashian Dipped Into the Chanel Archives Ahead of the Met Gala ,the use of the verb "look" is related to the adjectives "pearlescent" and "rare" when describing the Chanel bag.

Therefore, the overarching objective is to obtain the specific adjectives employed by the selected fashion websites when characterizing items de moda associated with the selected brands—Louis Vuitton, Chanel, Dior, and Versace.

adj_relatedToLook()
The `adj_relatedToLook` function is designed to identify and select adjectives associated with the word "look" within a given text corpus. The function utilizes the NLTK library, specifically the pos_tag() function, to determine the part-of-speech (POS) tags of each word in the corpus. The primary objective is to capture adjectives (identified by the POS tag "JJ") occurring after instances of the word "look."

In [30]:
```
porter_stemmer = PorterStemmer()

"""
    1. The function starts by initializing a position pointer (pos) to iterate through the
    list of tokens (words) in the corpus.
    2. The function iterates through the tokens, and when it encounters the word "look", it
    advances the pointer (pos) to the next word.
    3. Subsequently, the function continues to iterate through the tokens, selecting adjectives
    identified by the POS tag "JJ" and adding them to the list of selected adjectives (listOfAdj).
    This process continues until a period (".") or word "look" are encountered, indicating the
    end of a sentence or idea.
    4. The selected adjectives are then joined into a string, separated by commas. This string
    is converted to lowercase and returned as the output of the function.
"""
```

```
def adj_relatedToLook(corpus):
    """
    This function selects adjectives related to the word "look" from a given corpus.

    Args:
    - corpus (str): The input text corpus.

    Returns:
    - str: A comma-separated string of selected adjectives in lowercase.
    """

    tokens = word_tokenize(corpus) # Tokenize the input text.
    pos=0 # Initialize position variable.
    listOfAdj = [] # Initialize a list to store selected adjectives.

    while pos < len(tokens):
        # Check if the current token (stemmed) is "look."
        if porter_stemmer.stem(tokens[pos]) == "look":
            pos+=1 # Move to the next token.
            while pos < len(tokens) and tokens[pos] != "look" and tokens[pos] != ".":
                # Check if the current token is an adjective ("JJ").
                if get_word_pos(tokens[pos]) == "JJ":
                    listOfAdj.append(tokens[pos]) # Add the adjective to the list.
                pos += 1
        pos+=1
     # Return a comma-separated string of selected adjectives in lowercase.
    return ",".join(set(listOfAdj)).lower()
```

Now, a function will be created to streamline the retrieval of adjectives associated with the term *"look"* for each corpus within the **"article_corpus"** column of the DataFrames designed for the selected brands (i.e., df_louisVuitton, df_chanel, df_dior, df_versace).

In [31]:
```
def get_adj_relatedToLookDF(nameDF, columName, selectedNames=[f"row_{i}" for i in range(0, len(nameDF))]):
    """
    This function returns a list of dictionaries containing adjectives related to the word "look" for each
    corpus in the specified DataFrame column.

    Args:
    - nameDF (pd.DataFrame): The DataFrame containing text corpora.
    - columName (str): The column name in the DataFrame containing text corpora.
    - selectedNames (list): A list of selected(websites) names  to be the keys of the dictionaries created
    for each row (optional), default is row_1,row_2...row_len(nameDF), None only return a list of strings,
    each string represents the selected adjectives.

    Returns:
    - list: A list of dictionaries containing as keys selected(websites) names and as values corresponding
    selected adjectives.
    """

    if selectedNames != None:
        # Create a list of dictionaries with selectedNamesand associated selected adjectives.
        allSets = [{name: list(adj_relatedToLook(corpus).split(","))}
                   for corpus, name in zip(nameDF[columName], selectedNames)]
    else:
        # Create a list(strings) of  selected adjectives for each corpus.
        allSets = [adj_relatedToLook(corpus) for corpus in nameDF[columName]]
    return allSets
```

Example of different usages of `get_adj_relatedToLookDF`

In [32]:
```
# design: [{website}:[selected adjectives]]
lsWebs = df_dior["website"].tolist()
get_adj_relatedToLookDF(df_dior, "article_corpus", lsWebs)
```

Out[32]:
```
[{'vogue': ['natural', 'normal', 'iconic', 'upper']},
 {'glamour': ['natural',
   'nice',
   'knee-high',
   'popsicle-stained',
   'clear',
   'white',
   'classic',
   'black',
   'single-breasted']},
 {'elle': ['low',
   'same',
   'chic',
   'strong',
   'little',
   'black',
   'sensual',
   'soft']},
 {"harper's bazar": ['enormous',
   'few',
   'second',
   'complex',
   'big',
   'previous',
   'white',
   'victorian',
   'black',
   'technical',
   'silver-striped',
   'oscar-worthy',
   'curvaceous']}]
```

In [33]:
```
name_of_Brands = ["Louis Vuitton", "Chanel", "Dior", "Versace"]
for dfName, nameDFBrand, nameBrand in zip(list_df_brands, name_df_brands, name_of_Brands):
    lsWebs = dfName["website"].tolist()
    print(nameDFBrand)
    print(f"Products appearance for {nameBrand} per each website: ")
    # design: [{website_1}:[selected adjectives],{website_2}:[selected adjectives]]
    print(get_adj_relatedToLookDF(dfName, "article_corpus", lsWebs))
    print()
```

```
df_louisVuitton
Products appearance for Louis Vuitton per each website:
[{'vogue': ['classic', 'modern', 'fresh', 'white']}, {'glamour': ['unbuttoned', 'black']}, {'elle': ['venetian']}, {"harper's bazar": ['creative', 'red', 'obvious', 'former']}]

df_chanel
Products appearance for Chanel per each website:
[{'vogue': ['chic', 'strong', 'other', 'white', 'black', 'short', 'direct']}, {'glamour': ['red', 'such', 'legendary', 'la-inspired', 'able', 'white', 'daily', 'black', 'particular']}, {'elle': ['invisible']}, {"harper's bazar": ['red', 'many', 'chic', 'different', 'full', 'hot']}]

df_dior
Products appearance for Dior per each website:
[{'vogue': ['natural', 'normal', 'iconic', 'upper']}, {'glamour': ['natural', 'nice', 'knee-high', 'popsicle-stained', 'clear', 'white', 'classic', 'black', 'single-breasted']}, {'elle': ['low', 'same', 'chic', 'strong', 'little', 'black', 'sensual', 'soft']}, {"harper's bazar": ['enormous', 'few', 'second', 'complex', 'big', 'previous', 'white', 'victorian', 'black', 'technical', 'silver-striped', 'oscar-worthy', 'curvaceous']}]

df_versace
Products appearance for Versace per each website:
[{'vogue': ['subversive', 'red', 'last', 'same', 'kardashian', 'last', 'non-stop', 'chic', 'weekly', 'glamorous', 'black', 'well-heeled', 'new', 'run-through', 'hot']}, {'glamour': ['complete', 'red', 'low', 'last', 'strappy', 'modest', 'chanel-inspired']}, {'elle': ['own', 'gorgeous', 'white', 'black', 'real', 'good']}, {"harper's bazar": ['black']}]
```

In [34]:
```python
# design: [{row_1}:[selected adjectives],{row_2}:[selected adjectives]...]
for dfName, nameDFBrand in zip(list_df_brands, name_df_brands):
    print(f"For {nameDFBrand}:", get_adj_relatedToLookDF(dfName, "article_corpus"))
    print()
```

```
For df_louisVuitton: [{'row_0': ['classic', 'modern', 'fresh', 'white']}, {'row_1': ['unbuttoned', 'black']}, {'row_2': ['venetian']}, {'row_3': ['creative', 'red', 'obvious', 'former']}]

For df_chanel: [{'row_0': ['chic', 'strong', 'other', 'white', 'black', 'short', 'direct']}, {'row_1': ['red', 'such', 'legendary', 'la-inspired', 'able', 'white', 'daily', 'black', 'particular']}, {'row_2': ['invisible']}, {'row_3': ['red', 'many', 'chic', 'different', 'full', 'hot']}]

For df_dior: [{'row_0': ['natural', 'normal', 'iconic', 'upper']}, {'row_1': ['natural', 'nice', 'knee-high', 'popsicle-stained', 'clear', 'white', 'classic', 'black', 'single-breasted']}, {'row_2': ['low', 'same', 'chic', 'strong', 'little', 'black', 'sensual', 'soft']}, {'row_3': ['enormous', 'few', 'second', 'complex', 'big', 'previous', 'white', 'victorian', 'black', 'technical', 'silver-striped', 'oscar-worthy', 'curvaceous']}]

For df_versace: [{'row_0': ['subversive', 'red', 'last', 'same', 'kardashian', 'last', 'non-stop', 'chic', 'weekly', 'glamorous', 'black', 'well-heeled', 'new', 'run-through', 'hot']}, {'row_1': ['complete', 'red', 'low', 'last', 'strappy', 'modest', 'chanel-inspired']}, {'row_2': ['own', 'gorgeous', 'white', 'black', 'real', 'good']}, {'row_3': ['black']}]
```

In [35]:
```python
# design: ['adjA1,adjA2..' , 'adjB1,adB2..', 'adjC1,adjC2..', ...]
for dfName,nameDFBrand in zip(list_df_brands,name_df_brands):
    print(f"For {nameDFBrand}:",get_adj_relatedToLookDF(dfName, "article_corpus", None))
    print()
```

```
For df_louisVuitton: ['classic,modern,fresh,white', 'unbuttoned,black', 'venetian', 'creative,red,obvious,former']

For df_chanel: ['chic,strong,other,white,black,short,direct', 'red,such,legendary,la-inspired,able,white,daily,black,particular', 'invisible', 'red,many,chic,different,full,hot']

For df_dior: ['natural,normal,iconic,upper', 'natural,nice,knee-high,popsicle-stained,clear,white,classic,black,single-breasted', 'low,same,chic,strong,little,black,sensual,soft', 'enormous,few,second,complex,big,previous,white,victorian,black,technical,silver-striped,oscar-worthy,curvaceous']

For df_versace: ['subversive,red,last,same,kardashian,last,non-stop,chic,weekly,glamorous,black,well-heeled,new,run-through,hot', 'complete,red,low,last,strappy,modest,chanel-inspired', 'own,gorgeous,white,black,real,good', 'black']
```

Finally, several enhancements will be implemented in the existing DataFrames. **(a) Firstly**, new columns will be introduced in each brand-specific DataFrame `(df_louisvuitton, df_chanel, df_dior, df_versace)` to catalog the adjectives identified in connection with each corpus. **(b) Secondly**, in the global DataFrame, `df_globalBrands`, which consolidates the four individual brand corpora, a new column will be incorporated to house the adjectives associated with each corpus using the `adj_relatedToLook` function for each brand-specific corpus. **(c) Lastly**, an additional column will be appended to the primary DataFrame, `df_articles`, capturing the five most common adjectives associated with each article.

In [36]:
```python
# (a)
# Iterate through the list of DataFrames for each brand
for dfName in list_df_brands:
    # Create a new column "products_appearance" to store the identified adjectives for each corpus
    dfName["products_appearance"] = dfName["article_corpus"].apply(adj_relatedToLook)
```

In [37]:
```python
#(b)
# Applying the adj_relatedToLook function to extract adjectives associated with the term "look"
#for each brand-specific corpus in the df_globalBrands DataFrame.
df_globalBrands["products_appearance"] = df_globalBrands["articles_corpus_final"].apply(
    adj_relatedToLook)
```

In [38]:
```python
#(c)
def get_freq_adjs(text, lower=True, sorted_frec=True):
    """
    This function takes a text as input and returns a dictionary containing
    the frequency of each adjective present in the text.

    Parameters:
    - text (str): The text for which adjective frequencies are calculated.
    - lower (bool): If True, converts all words to lowercase in the returned dictionary.

    Returns:
    - dict: A dictionary where keys are adjectives, and values are their respective frequencies.
    """

    # Tokenize the text into words
    tokens = word_tokenize(text)
    # Create a set of unique tokens to avoid redundant counting
    setTokens = set(tokens)
    # Create a dictionary to store the frequency of each adjective
    freq_tokens = {token.lower() if lower else token:tokens.count(token) \
                   for token in setTokens if get_word_pos(token)=="JJ"}
    # Sort the adjectives by frequency in descending order
    sortedAdjectivesOrnot = dict(
        sorted(freq_tokens.items(), key=lambda item: item[1], reverse=True)) if sorted_frec else freq_tokens

    return sortedAdjectivesOrnot

df_articles["Most_commom_Adjectives"] = \
df_articles["Filtered_content"].apply(lambda x: ",".join(list(get_freq_adjs(x).keys())[:5]))
```

To record and analyze the frequencies of adjectives associated with each brand, the DataFrame `df_globalBrands` will be employed, containing a full corpora for each brand. The `get_freq_adjs` function will extract groups of adjectives along with their respective frequencies from each corpus in the `"articles_corpus_final"` column of `df_globalBrands`. These sets of adjectives and frequencies will be channeled into individual DataFrames, each corresponding to the associated brand.

The decision to store adjectives and frequencies in separate DataFrames for each brand is grounded in the subsequent utility of individually evaluating the frequency of each adjective. Furthermore, it allows us to identify the most and least frequent adjectives for each brand. This information is essential, as these adjectives uniquely singularize and characterize each brand, providing valuable insights into how they are perceived in the analyzed texts.

In [39]:
```
def createDF_frequency(freq_dict,columName):
   """
   Creates a DataFrame from a frequency dictionary.

   Parameters:
   - freq_dict (dict): A dictionary containing words as keys and their frequencies as values.
   - column_name (str): The name to be assigned to the column containing words in the DataFrame.

   Returns:
   - pd.DataFrame: A DataFrame with two columns - one for words and the other for frequencies.

   Example:
   >>> word_freq = {'happy': 15, 'sad': 8, 'excited': 12, 'calm': 10}
   >>> df = createDF_frequency(word_freq, 'Emotion')
   >>> print(df)
      Emotion  Frequency
   0   happy        15
   1     sad         8
   2 excited        12
   3    calm        10
   """
   return pd.DataFrame(list(freq_dict.items()), columns=[f"{columName}", "Frequency"])
```

In [40]:
```
def get_freqAdj_dict(index):
    """
    Get the frequency of adjectives for a given index in the "articles_corpus_final"
    column of df_globalBrands.

    Parameters:
    - index (int): The index representing one of the selected luxury brands in df_globalBrands.

    Returns:
    - dict: A dictionary containing adjectives as keys and their frequencies in the
    corresponding corpus as values.

    Example:
    >>> # Assuming df_globalBrands["articles_corpus_final"][0] corresponds to the
        # corpus of Louis Vuitton
    >>> lv_adj_freq = get_freqAdj_dict(0)
    >>> print(lv_adj_freq)
    {'luxurious': 25, 'elegant': 18, 'iconic': 12, 'exclusive': 20, ...}
    """
    return get_freq_adjs(
    df_globalBrands["articles_corpus_final"][index], sorted_frec=False)
```

In [41]:
```
# Create DataFrames for adjective frequencies for each luxury brand
df_adjectives_LV = createDF_frequency(get_freqAdj_dict(0), "Adjective")
df_adjectives_Chanel = createDF_frequency(get_freqAdj_dict(1), "Adjective")
df_adjectives_Dior = createDF_frequency(get_freqAdj_dict(2), "Adjective")
df_adjectives_Versace = createDF_frequency(get_freqAdj_dict(3), "Adjective")

# Example:
print(df_adjectives_LV.head())
# Output:
#      Adjective  Frequency
# 0       formal          2
# 1        green          4
# 2  comfortable          1
# 3   historical          1
# 4     gorgeous          2
# ...
```

```
    Adjective  Frequency
0        true          2
1    artistic          1
2     overall          1
3  functional          1
4   lug-soled          1
```

(2) Reconnaissance of Proper Nouns and Entities: This phase involves the recognition of proper nouns and substantive entities within the fashion articles. This strategic step is paramount, considering that these entities predominantly encompass organizations , luxury products, designers, and artists. By identifying and extracting such entities, we aim to enrich our understanding of the intricate network of relationships and influences woven into the narrative of each brand.

For example, in this paragraph, extracted from the Glamour magazine,
However, the standout feature of Kardashian's look was her pearlescent Chanel clutch bag, a rare item designed by the late, legendary designer Karl Lagerfeld.
Kim Kardashian Dipped Into the Chanel Archives Ahead of the Met Gala,
the proper noun "Karl Lagerfeld" represents a legendary German fashion designer who left an indelible mark in the industry during the latter half of the 20th century.

In [42]:
```
nltk.download("maxent_ne_chunker")
nltk.download("words")
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\aloza\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\aloza\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Out[42]:
```
True
```

In order to address the limitations inherent in the NLTK library's ability to accurately recognize certain nouns, a supplementary list of entities was manually curated.

In [43]:
```
additional_entities = ["Kristen", "Martin","Pharrell"]
```

get_entities()
The `get_entities` function has been designed with the purpose of extracting named entities from text. In doing so, it recognizes both proper nouns (NNP in NLTK terminology) and nouns (NN) that start with a capital letter. This last adaptation arises from the observation that, despite the power of NLTK, some entities are not correctly identified as NNP but as NN. Therefore, to recognize nouns that could be entities, the function verifies whether they start with a capital letter.

Recognizing that entities often consist of multiple words, the function has been tailored to iterate over each word. When it encounters a word recognized as NNP or NN (initial capital letter), it examines consecutive words until they no longer meet this condition, ensuring the capture of the complete entity name.

It is crucial to note that not all words recognized as NNP or NN (initial capital letter) are valid entities. To address this ambiguity, the function uses NLTK's entity analyzer, ne_chunk, to determine if the recognized word(s) is of type PERSON. This validation process ensures that only true entities are considered. The identified entities are added to a list and returned as a set to avoid duplicates.

In [44]:
```
def is_valid_entity(tokens, start_pos, end_pos, exclude_words):
    """
    Helper function to check if a sequence of tokens forms a valid entity.

    Args:
    - tokens (list): List of tokens representing the entire corpus.
    - start_pos (int): Starting position of the sequence of tokens.
    - end_pos (int): Exclusive ending position of the sequence of tokens.
    - exclude_words (list): A list of words to be excluded from the entity extraction process.

    Returns:
    - bool: True if the sequence forms a valid entity, False otherwise.
    """

    # Combine tokens to form the entity
    nnp = " ".join(tokens[start_pos:end_pos])

    # Perform part-of-speech tagging and named entity chunking
    tagged_nnp = pos_tag(word_tokenize(nnp))
    chunked_nnp = ne_chunk(tagged_nnp)

    # Check if entity should be excluded based on specified words
    if exclude_words is not None:
        exclude_pattern = "|".join(exclude_words)
        if re.findall(exclude_pattern, nnp):
            return False

    # Check if the chunked entity is of type PERSON
    return chunked_nnp and hasattr(chunked_nnp[0], 'label') and chunked_nnp[0].label() == "PERSON"


def get_entities(corpus, exclude_words=None, additional_entities=None, unique=True):
    """
    Extracts named entities from a given corpus, considering both proper
    nouns (NNP) and capitalized nouns (NN).

    Args:
    - corpus (str): The input text corpus.
    - exclude_words (list): A list of words to be excluded from the entity
    extraction process. (optional)
    - additional_entities (lits): A list of manually added entities. (optional)

    Returns:
    - set: A set containing identified named entities in the corpus.
    """

    # Tokenize the input corpus
    tokens = word_tokenize(corpus)
    pos = 0
    final_entities = []

    while pos < len(tokens):
        # Check if the part of speech of the token is a proper noun (NNP) or a
        # common noun (NN), or if the token is in the list of additional_entities
        # (manually added entities), and if the first character of the token is uppercase.
        if (get_word_pos(tokens[pos]) in {"NNP", "NN"} or
                (additional_entities is not None and tokens[pos] in additional_entities)) \
        and tokens[pos][0].isupper():
            start_pos = pos
            pos += 1

            # Continue checking consecutive tokens forming a potential entity
            while pos < len(tokens) and get_word_pos(tokens[pos]) in {"NNP", "NN"} \
            and tokens[pos][0].isupper():
                pos += 1

            # Check if the identified sequence forms a valid entity
            if is_valid_entity(tokens, start_pos, pos, exclude_words):
                final_entities.append(" ".join(tokens[start_pos:pos]))
        else:
            pos += 1

    return set(final_entities) if unique else final_entities
```

Several enhancements are planned for the existing DataFrames. **(a) First**, an additional column will be introduced for each brand-specific DataFrame `(df_louisvuitton, df_chanel, df_dior, df_versace)` that will contain the set of entities identified in connection with each corpus of the `article_corpus` column. **(b) Subsequently**, in the global DataFrame, `df_globalBrands`, which consolidates the four individual brand corpora, a new column will also be added to store the named entities associated with each corpus. **(c) Finally**, the same will be done for primary `df_articles` DataFrame: an additional column will be appended capturing the unique named entities present in each article. The `get_entities()` function will be employed for these processes.

In [45]:
```
# (a)
# Iterate through the list of DataFrames for each brand
# Exclude the brand names as entities, as we want to focus on other entities in the corpus
exclusiveWords = [["Vuitton"],["Chanel"],["Dior"],["Versace"]]
for dfName,excluW in zip(list_df_brands,exclusiveWords):
    # Create a new column 'entities_involved' for each brand-specific DataFrame
    dfName["entities_involved"] = \
    dfName["article_corpus"].apply(lambda x: ",".join(get_entities(x, excluW,additional_entities=additional_entities)))

#Summary
for dfName,nameDF in zip(list_df_brands,name_df_brands):
    print(f"{nameDF} count of entities")
    # Print the count of entities in the 'entities_involved' column for each brand-specific DataFrame
    print(dfName["entities_involved"].apply(lambda x: len(x.split())))
    print()
```

```
df_louisVuitton count of entities
0    30
1     8
2    11
3    21
Name: entities_involved, dtype: int64

df_chanel count of entities
0    17
1    16
2    22
3    25
Name: entities_involved, dtype: int64

df_dior count of entities
0    20
1    21
2    23
3    32
Name: entities_involved, dtype: int64

df_versace count of entities
0    40
1    33
2    21
3    10
Name: entities_involved, dtype: int64
```

In [46]:
```python
# (b) Extract named entities for each brand in df_globalBrands
df_globalBrands["entities_involved"] = ""
for index, row in df_globalBrands.iterrows():
    # Use the corresponding exclusion words for the brand
    excluW = exclusiveWords[index]
    # Use get_entities to extract named entities
    df_globalBrands.at[index, "entities_involved"] = ",".join(
        get_entities(row["articles_corpus_final"], excluW, additional_entities=additional_entities))

# Summary
print(f"df_globalBrands column 'entities_involved'")
# Iterate over entities and brand names to print a summary
for entities, brandName in zip(df_globalBrands["entities_involved"],name_of_Brands):
    print(f"{brandName}: {len(entities.split())} entities involved")
```

```
df_globalBrands column 'entities_involved'
Louis Vuitton: 60 entities involved
Chanel: 54 entities involved
Dior: 82 entities involved
Versace: 75 entities involved
```

It is crucial to have a count of all entities closely related to each brand. To achieve this, we will apply the `get_entities` function to each corpus in the `"articles_corpus_final"` column of `df_globalBrands`. Subsequently, using the `Counter()` tool, we will obtain a dictionary containing the entities and their respective frequencies. Then, we will store this data in individual DataFrames for each brand.

This approach will provide us with a detailed insight into the entities that have a higher or lower association with each specific brand. The decision to maintain the information in individual DataFrames is based on the convenience of visualizing all entities related to each brand clearly and efficiently.

In [47]:
```python
def counter_df_entities(index, excludeWord):
    """
    Count entities in the articles corpus for a specific brand.

    Parameters:
    - index (int): The index representing the brand in the DataFrame df_globalBrands.
    - excludeWord (str): The brand name to exclude from entity identification.

    Returns:
    collections.Counter: A Counter object with entities and their frequencies.

    Example:
    >>> counter_df_entities(0, "Louis Vuitton")
    Counter({'fashion': 20, 'luxury': 15, 'style': 10, ...})

    >>> # Assuming df_globalBrands["articles_corpus_final"][0] corresponds to the
    corpus of Louis Vuitton
    >>> lv_ent_freq = counter_df_entities(0,"Vuitton")
    >>> print(lv_ent_freq)
    Counter({'Taylor Swift': 25, 'Marc Jacobs': 18, 'Alexander McQueen': 12, ...})
    """
    return Counter(get_entities(df_globalBrands["articles_corpus_final"][index],
                                exclude_words=[excludeWord],
                                additional_entities=additional_entities,
                                unique=False))


# Create DataFrames for entity frequencies for each brand
df_entities_LV = createDF_frequency(
    counter_df_entities(0, "Vuitton"), "Entity")
df_entities_Chanel = createDF_frequency(
    counter_df_entities(1, "Chanel"), "Entity")
df_entities_Dior = createDF_frequency(
    counter_df_entities(2,"Dior"), "Entity")
df_entities_Versace = createDF_frequency(
    counter_df_entities(3, "Versace"), "Entity")

# Example:
print(df_entities_LV.head())
# Output:
#      Entity           Frequency
# 0      John Galliano          1
# 1  Alexander McQueen          1
# 2        Marc Jacobs          5
# 3    Bernard Arnault          2
# 4    Martin Margiela          1
# ...
```

```
              Entity  Frequency
0      John Galliano          1
1  Alexander McQueen          1
2        Marc Jacobs          5
3    Bernard Arnault          2
4    Martin Margiela          1
```

In [48]:
```
#(c)
df_articles["entities_involved"] = ""
for index, row in df_articles.iterrows():
    excluW = [row["brand"].capitalize()] if excluW != "Louis vuitton" else ["Vuitton"]
    df_articles.at[index, "entities_involved"] = ",".join(get_entities(
        row["content"], excluW, additional_entities=additional_entities))

# Summary
# Calculate the average number of entities per article
average_entity_length = round(df_articles["entities_involved"].apply(
    lambda x: len(x.split())).mean())
print("The average number of entities per article is: {}".format(average_entity_length))
```

```
The average number of entities per article is: 8
```

4.2 Density of Historical terms in the world of Fashion

(3) Glamour : In ancient eras, the term "Glamour" held its roots in the mystical realm, denoting a magical spell or illusion, often associated with the enchantments cast by witches. As the 19th century drew to a close, the term underwent a transformation, referring to a non-magical object employed to enhance one's attractiveness, becoming known as "a glamour" [20]. In the modern history, the inception of glamorous aesthetics in fashion design can be traced back to the Blooming Age Glamour Look [21], spanning from the late 19th century to the late 1920s. This period was characterized by the opulent high fashion styles embraced by courtesans. The narrative unfolds further with the advent of the Pop Age Glamour Look [21], marked by fashion styles inspired by pop stars. In the 1980s and 1990s, the Glamour Renaissance Look [21] materialized, representing glamorous aesthetics that transcended social classes and were distinguished by luxury. This evolution of the term "glamour" encapsulates a narrative that extends to the contemporary era. Today, when referring to brands characterized as luxurious, enchanting, and exuding an aura of magic, the term "glamour" serves as a succinct descriptor. Recognizing its importance in fashion discourse becomes pivotal in characterizing the level of allure and magic attributed to a brand.

Recognizing the pivotal role this word plays in shaping the narrative of each brand is instrumental in discerning the differences in perceived glamour across brands. To quantify it, the frequency of the term "glamour" within the articles will be measured.

In [49]:
```
def get_freq_ofWord(text, target_word):
    """
    Calculate the frequency of a target word in the given text.

    Args:
    - text (str): The input text.
    - target_word (str): The word to calculate the frequency for.

    Returns:
    - int: The frequency of the target word in the text.
    """
    tokens = word_tokenize(text.lower())
    return tokens.count(target_word.lower())
```

A new column will be added to the global DataFrame `df_globalBrands`, representing the frequency of the word "glamour" for each lemmatized corpus in the `"lemmatized_articles_corpus"` column. This decision is driven by the interest in assessing the presence and prominence of the keyword at the level of each specific brand.

In [50]:
```
df_globalBrands["glamour_frequency"] = \
df_globalBrands["lemmatized_articles_corpus"].apply(lambda x: get_freq_ofWord(x,"glamour"))
```

(4) Iconic: In the realm of fashion, the term "iconic" denotes something highly recognizable, influential, and distinctive [22]. A design, garment, or style earns the label of iconic when it has made a significant impact on fashion and continues to be relevant and admired over time. An exemplary instance is the iconic Chanel 2.55 handbag [23], which defied conventional notions of femininity and stereotypes upon its launch in February 1955. It marked the first luxury handbag for women with a shoulder strap, challenging the previous norm of handheld purses. Iconic elements in fashion often define eras, shape future trends, and can even reflect social and cultural shifts [24].

Recognizing the vitality of this term, we will assess the frequency of the word "iconic" in fashion articles. This analysis will provide valuable insights into how the concept of "iconic" is conceptualized and communicated in relation to brands.

A new column will be added to the global DataFrame `df_globalBrands`, representing the frequency of the word "iconic" for each lemmatized corpus in the `"lemmatized_articles_corpus"` column. This decision is driven by the interest in assessing the presence and prominence of the keyword at the level of each specific brand.

In [51]:
```
df_globalBrands["iconic_frequency"] = \
df_globalBrands["lemmatized_articles_corpus"].apply(lambda x: get_freq_ofWord(x, "iconic"))
```

(5) Beauty: The word "beauty" encapsulates not only aesthetic attributes but also the unique expression of identity through clothing and style [25]. The influence of certain fashion elements, such as Audrey Hepburn's iconic black dress in "Breakfast at Tiffany's," has transcended generations, becoming synonymous with timeless elegance and grace [26]. "Beauty" in fashion extends beyond the physical, serving as a means to tell stories, challenge norms, and reflect the evolution of society.

Recognizing the intrinsic importance of the word "beauty," we will explore its frequency of use in fashion articles.

A new column will be added to the global DataFrame `df_globalBrands`, representing the frequency of the word "beauty" for each lemmatized corpus in the `"lemmatized_articles_corpus"` column. This decision is driven by the interest in assessing the presence and prominence of the keyword at the level of each specific brand.

In [52]:
```
df_globalBrands["beauty_frequency"] = df_globalBrands["lemmatized_articles_corpus"].apply(
    lambda x: get_freq_ofWord(x, "beauty"))
```

4.3 Apparel Trends

Exploring the most popular garments and dominant fabric types.

(6) Garment: In this phase, the primary objective is to identify the most relevant garment in each article. To accomplish this, we will extract words recognized as clothing items from the article or corpus. To ensure accurate identification, an extensive list covering over 200 different clothing items relevant in the world of fashion will be employed. Subsequently, we will assess the frequency of each of these selected clothing items in the corresponding article or corpus. This approach will help us discern which garment stands out in comparison to others, offering a nuanced understanding of clothing preferences expressed in fashion articles.

The list of clothing items was created from a TXT document. This document comprises over 200 distinct apparel keywords, each separated by commas. By extracting theses keywords, a comprehensive list of various fabrics is formed

In [53]:
```
#file path for the TXT document containing clothing keywords
file_path = 'clothing_keywords.txt'
with open(file_path, 'r') as file:
    content = file.read()
    # Split the content based on commas and create a list of clothing keywords
    clothing_keywords = list(content.split(', '))
```

In [54]:
```
def get_word_frequencies_singleText(text, order, clean=True):
    """
    Extracts word frequencies from a given text.
```

```
    Parameters:
    - text (str): Input text.
    - clean (bool): If True, clean the text before processing.

    Returns:
    - dict: A dictionary containing word frequencies in descending order.
    """
    # Clean the text if specified
    text = cleanCOR(text,lower=True).split() if clean else text.split()
    # Count the occurrences of each word
    frequencyOFwords = Counter(text)
    # Sort the word frequencies in descending order
    sorted_countUniqueW = dict(
        sorted(frequencyOFwords.items(), key=lambda item: item[1], reverse=True)) if order else frequencyOFwords
    return sorted_countUniqueW
```

In [55]:
```
def freq_based_in_single_keywords(text, keywords,order=True):
    """
    Extracts the frequency of single words in the given text that match the provided keywords.

    Parameters:
    - text (str): The input text for frequency analysis.
    - keywords (list): A list of keywords representing words to find a match.

    Returns:
    - frequency_based_keywords (dict): A dictionary containing the frequency of each selected word.
      The dictionary is sorted in descending order based on the frequency.
    """
    # Get the frequencies of all words in the text
    words_freq = get_word_frequencies_singleText(text, order)
    # Filter frequencies based on specified keywords
    frequency_based_keywords = {item: words_freq[item]
                    for item in words_freq if item in keywords}
    return frequency_based_keywords
```

(a) Initially, an additional column will be introduced in each brand-specific DataFrame (`df_louisvuitton` , `df_chanel` , `df_dior` , `df_versace`) to store the set of two most relevant clothing items for each entry in the `filtered_corpus` column. **(b) Following that**, in the global DataFrame `df_globalBrands` , which consolidates the four individual brand corpora, another column will be added to capture the set of the top 5 most relevant clothing items associated with each corpus. **(c) Finally**, the same procedure will be applied to the primary `df_articles` DataFrame: an extra column will be appended to record the most significant clothing items present in each article. The `freq_based_in_single_keywords()` function will be utilized for these processes.

In [56]:
```
#(a)
# Iterate through the list of DataFrames for each brand
for dfName in list_df_brands:
    # Create a new column to save top two relevant apparel for each corpus
    dfName["relevant_apparel"] = \
    dfName["filtered_corpus"].apply(lambda x: ",".join(list(freq_based_in_single_keywords(x, clothing_keywords))[:2]))
```

relevant_column_view()
This function creates a multi-level indexed DataFrame where the index comprises the 'brand' and 'website' columns, enabling a hierarchical view of the information. By specifying the relevant column of interest, such as 'relevant_apparel', users can easily examine which apparel items dominate in each unique combination of brand and website.

The `relevant_column` parameter in the function plays a crucial role by allowing the user to specifically choose the column of interest when displaying multi-level information between 'brand' and 'website'. This modularity provides flexibility to tailor the function for future scenarios.

In [57]:
```
def relevant_column_view(dfName, relevant_column, name=None):
    """
    Displays a multi-level indexed DataFrame with information about each brand, website,
    and a specified relevant column.

    Parameters:
    - dfName (pd.DataFrame): The DataFrame containing the relevant data, with columns
    'brand', 'website', and relevant_column.
    - name (str, optional): The optional name of the DataFrame to be displayed at the
    top of the result for context.

    Returns:
    pd.DataFrame: A sorted DataFrame based on the multi-index of 'brand' and 'website',
    containing only the relevant_column column.

    Example when relevant_column = relevant_apparel:
    >>> df = pd.DataFrame({
    ...     'brand': ['Brand1', 'Brand2', 'Brand1', 'Brand2'],
    ...     'website': ['website1', 'website2', 'website1', 'website2'],
    ...     'relevant_apparel': ['Apparel1', 'Apparel2', 'Apparel3', 'Apparel4']
    ... })
    >>> relevant_apparel_view(df, name='Sample Data')
    Displaying relevant_column view for DataFrame: Sample Data

    brand   website
    Brand1  website1    Apparel1
            website1    Apparel3
    Brand2  website2    Apparel2
            website2    Apparel4
    Name: relevant_apparel, dtype: object
    """
    # Display the optional name at the top of the result for context
    if name is not None:
        print(f"Displaying {relevant_column} view for DataFrame: {name}\n")
    # Set the index to multi-index on 'brand' and 'website', then sort the DataFrame
    result_df = dfName.set_index(['brand', 'website']).sort_index()[
        [relevant_column]]
    return result_df
```

Primary apparel for Luis Vuitton across different websites:

In [58]:
```
relevant_column_view(df_louisVuitton, "relevant_apparel", "df_louisVuitton")
```

```
Displaying relevant_apparel view for DataFrame: df_louisVuitton
```

Out[58]:

		relevant_apparel
brand	**website**	
louis vuitton	**elle**	hat,ring
	glamour	dress,beanie
	harper's bazar	jacket,shorts
	vogue	dress,gown

Primary apparel for Chanel across different websites:

In [59]:
```
relevant_column_view(df_chanel, "relevant_apparel", "df_chanel")
```

```
Displaying relevant_apparel view for DataFrame: df_chanel
```

Out[59]:

		relevant_apparel
brand	**website**	
chanel	**elle**	heels,jumpsuit
	glamour	dress,shorts
	harper's bazar	dress,gown
	vogue	dress,loafers

Primary apparel for Dior across different websites:

In [60]:
```
relevant_column_view(df_dior, "relevant_apparel", "df_dior")
```

```
Displaying relevant_apparel view for DataFrame: df_dior
```

Out[60]:

		relevant_apparel
brand	**website**	
dior	**elle**	sneakers,boots
	glamour	dress,gown
	harper's bazar	dress,skirt
	vogue	jacket,dress

Primary apparel for Versace across different websites:

In [61]:
```
relevant_column_view(df_versace, "relevant_apparel", "df_versace")
```

```
Displaying relevant_apparel view for DataFrame: df_versace
```

Out[61]:

		relevant_apparel
brand	**website**	
versace	**elle**	dress,gown
	glamour	dress,gown
	harper's bazar	jacket,boots
	vogue	dress,skirt

In [62]:
```
#(b)
# Update the 'relevant_apparel' column based on the top 5 dominant apparel items in each luxury brand's corpus
df_globalBrands["relevant_apparel"] = df_globalBrands["filtered_articles_corpus"].apply(
    lambda x: ",".join(list(freq_based_in_single_keywords(x, clothing_keywords).keys())[:5])
)
```

Essential clothing items for each brand:

In [63]:
```
df_globalBrands[["brand","relevant_apparel"]]
```

Out[63]:

	brand	relevant_apparel
0	louis vuitton	dress,jacket,gown,shirt,hat
1	chanel	dress,heels,loafers,gown,shorts
2	dior	dress,gown,skirt,jacket,coat
3	versace	dress,gown,skirt,boots,suit

Recognizing the importance of maintaining a detailed record of identified clothing items in each corpus within the `"articles_corpus_final"` column of `df_globalBrands`, a process has been implemented to store these garments and their respective frequencies in four separate dictionaries, one for each brand.

In [64]:
```
def freq_basedOnKeywords_byIndex(index, list_of_keywords):
    """
    Calculate the frequency of specified keywords for a given index in df_globalBrands.

    Parameters:
    - index (int): The index corresponding to a brand in the "articles_corpus_final"
    column of df_globalBrands.
    - list_of_keywords (list): A list of keywords for which a match is sought in the text.

    Returns:
    - dict: A dictionary containing the frequencies of keywords that matched the text
    at the specified index.

    Example:
    ```python
    # Sample usage
    index = 0  # Assuming index 0 corresponds to "Louis Vuitton" in df_globalBrands
    keywords = ['dress', 'jacket']

    # Calculate the frequency of specified keywords for the given index
    result_dict = freq_apparel_byIndex(index, keywords)
```
```

```
    # Expected Result:
    # {'dress': 5, 'jacket': 2}
    ```
    """
    return freq_based_in_single_keywords(
        df_globalBrands["articles_corpus_final"][index], list_of_keywords)

freq_apparel_LV = freq_basedOnKeywords_byIndex(0,clothing_keywords)
freq_apparel_Chanel = freq_basedOnKeywords_byIndex(1, clothing_keywords)
freq_apparel_Dior = freq_basedOnKeywords_byIndex(2, clothing_keywords)
freq_apparel_Versace = freq_basedOnKeywords_byIndex(3, clothing_keywords)
```

---

Recognizing the differences in frequencies of the same clothing item among the selected brands is crucial. To achieve this, we will design a DataFrame with three columns: `"Brand,"` `"Apparel,"` and `"Frequency."` This DataFrame will encompass all frequencies associated with identified garments for each brand. It is noteworthy that in the `"Apparel"` column, the same garment may appear more than once, each instance linked to different brands. This approach enables us to analyze and comprehend variations in the frequencies of a specific garment across different brands.

In [65]:
```
def createList_freq_items_by_brand(dfName, colName, brands_list, itemName, list_of_keywords):
    """
    Creates a list of dictionaries, each containing information about the frequency
    of specified items associated with brands.

    Parameters:
    - dfName (pd.DataFrame): The DataFrame from which data will be extracted.
    - colName (str): The column containing the text data.
    - brands_list (list): A list of brand names associated with each text in the
    specified column.
    - itemName (str): The name associated with the identified item (e.g., "clothing").
    - list_of_keywords (list): A list of keywords for which a match is sought in the
    text, along with their frequencies.

    Returns:
    - list: A list of dictionaries, each containing {Brand: brand_name,
    itemName: identified_item, Frequency: frequency}.

    Example:
    ```python
    import pandas as pd

    # Sample DataFrame
    data = {'Brand': ['Brand1', 'Brand2'],
            'TextData': ['Text containing dress and t-shirt.', 'Brand2 text with dress.']}
    df = pd.DataFrame(data)

    # List of keywords
    keywords = ['dress', 't-shirt']

    # Create the list of frequencies for items by brand
    result_list = createList_freq_items_by_brand(df, 'TextData', df['Brand'], 'clothing', keywords)

    # Expected Result:
    # [{'Brand': 'Brand1', 'clothing': 'dress', 'Frequency': 1},
    #  {'Brand': 'Brand1', 'clothing': 't-shirt', 'Frequency': 1},
    #  {'Brand': 'Brand2', 'clothing': 'dress', 'Frequency': 1}]
    ```

    Note:
    - This function extracts text data from the specified column and
    associates it with brand names.
    - It then identifies specified items based on the provided list of
    keywords and their frequencies.
    - The result is a list of dictionaries representing the frequency of
    identified items associated with brands.
    """
    # Initialize an empty list to store the frequencies of items associated with brands
    total_frequencies_brands_apparel = []
    # Iterate through each text and its corresponding brand
    for text, brand in zip(dfName[colName], brands_list):
        # Calculate the frequencies of items based on the list of keywords
        freq_text_keywords = freq_based_in_single_keywords(
            text, list_of_keywords)
        # Extend the list with dictionaries containing brand, identified item, and frequency
        total_frequencies_brands_apparel.extend(
            {"Brand": brand, itemName: item, 'Frequency': frequency} for item, frequency in freq_text_keywords.items())
    # Return the final list of frequencies for items associated with brands
    return total_frequencies_brands_apparel
```

In [66]:
```
# Use createList_freq_items_by_brand to obtain a list of dictionaries containing apparel frequencies associated with each brand
dict_freq_apparel_global = createList_freq_items_by_brand(
    df_globalBrands, "filtered_articles_corpus", name_of_Brands, "Apparel", clothing_keywords)
# Create a DataFrame with columns "Brand", "Apparel", and "Frequency" from the generated dictionary
df_frequency_apparel_by_brand = pd.DataFrame(dict_freq_apparel_global)

# Example Result:
#   Brand   Apparel Frequency
# 0 Louis Vuitton dress 5
# 1 Louis Vuitton jacket 2
# 2 Chanel dress 4
df_frequency_apparel_by_brand.head()
```

Out[66]:

| | Brand | Apparel | Frequency |
|---|---|---|---|
| **0** | Louis Vuitton | dress | 8 |
| **1** | Louis Vuitton | jacket | 5 |
| **2** | Louis Vuitton | gown | 4 |
| **3** | Louis Vuitton | shirt | 3 |
| **4** | Louis Vuitton | hat | 3 |

In [67]:
```
#(c)
Update the 'relevant_apparel' column with the most dominant apparel item for each article
df_articles["relevant_apparel"] = df_articles["Filtered_content"].apply(lambda x : ",".join(list(freq_based_in_single_keywords(x,clothing_keywords).keys())[:1]))
```

In [68]:
```
def get_titles_by_parameter(dfName, relevant_column,item, selectDFCol=None):
    """
```
```

```
    Retrieves titles from a DataFrame where the specified relevant_column
    matches the given item.

    Parameters:
    - dfName (pd.DataFrame): The DataFrame to search for rows, containing a
    column specified by 'relevant_column'.
    - relevant_column (str): The name of the column to be considered during the search.
    - item (str): The item to search for in the specified 'relevant_column' column.
    - selectDFCol (list, optional): A list of column names to include in the
    returned DataFrame. Defaults to None.

    Returns:
    - pd.DataFrame or list: If 'selectDFCol' is provided, returns a DataFrame
    with specified columns.
      Otherwise, returns a list of titles for articles where the specified
      'relevant_column' matches 'item'.

    Example:
    >>> get_titles_by_parameter(df_articles, "relevant_apparel", "gown", selectDFCol=["title", "relevant_apparel"])
    Returns a DataFrame with columns 'title' and 'relevant_apparel' for articles with 'gown' in "relevant_apparel".

    >>> get_titles_by_apparel_inDF(df_articles, "relevant_apparel", "shirt")
    Returns a list of titles for articles with 'shirt' in the specified 'relevant_column'.
    """
    # Filter the DataFrame based on the specified relevant_column and item
    titles = df_articles.loc[dfName[relevant_column] == f"{item}"]
    # Return either a DataFrame with specified columns or a list of titles
    return titles[selectDFCol] if selectDFCol != None else titles["title"].tolist()
```

In [69]:
```
# Example usage
print("What are the titles of the articles where the dominant apparel is 'gown'?")
get_titles_by_parameter(df_articles,"relevant_apparel", "gown")
```

```
What are the titles of the articles where the dominant apparel is 'gown'?
['Kristen Stewart Goes Braless in Sheer Chanel',
 'Kristen Stewart Looked Like a Doll in a White Ruffled Chanel Gown in Berlin']
```

Out[69]:

(7)The most dominant clothing item in the World of Fashion according to Louis Vuitton, Chanel, Dior, and Versace: To determine the most dominant apparel throughout the year in the world of fashion, we will focus on the selected luxury fashion brands (Louis Vuitton, Chanel, Dior, and Versace) based on the available articles. To achieve this, we will consolidate all these articles using the `filtered_articles_corpus` column from df_globalBrands, which encompasses all the articles associated with each brand.

These articles will be amalgamated into a unified corpus. The subsequent step involves applying the `freq_based_in_single_keywords` method and utilizing the `clothing_keywords` list. This process will enable us to pinpoint the most prominent apparel among all the articles linked to these prestigious fashion brands.

In [70]:
```
# Concatenate all articles in the "filtered_articles_corpus" column to create a unified corpus
final_corpus = " ".join(df_globalBrands["filtered_articles_corpus"])
# Use the 'freq_based_in_single_keywords' function to calculate the frequency of apparel in the unified corpus
frequency_apparel_global = freq_based_in_single_keywords(final_corpus,clothing_keywords,order=False)
# Create a DataFrame from frequency_apparel_global
df_frequency_apparel_global = pd.DataFrame(
    list(frequency_apparel_global.items()), columns=['Apparel', 'Frequency'])
```

THE WINNER

In [71]:
```
df_frequency_apparel_global.loc[df_frequency_apparel_global["Frequency"].idxmax()]
```

```
Apparel      dress
Frequency       64
Name: 1, dtype: object
```

Out[71]:

The winning clothing item in the data exploration was 'dress' with a frequency of 64. This result indicates that dresses were the most dominant garment throughout the year according to Louis Vuitton, Chanel, Dior, and Versace.

(8) Fabric: For this second phase, the objective is to identify the most relevant fabric in each article or corpus. To achieve this, words recognized as fabrics will be extracted. To ensure accurate identification, an extensive list covering over 150 different fabrics will be employed. Subsequently, we will assess the frequency of each of these selected fabrics in the corresponding article or corpus. This approach will help us discern which fabric stands out in comparison to others.

The list of fabrics was created from a TXT document. This document comprises over 150 distinct fabric keywords, each separated by commas. By extracting theses keywords, a comprehensive list of various fabrics is formed

In [72]:
```
file_path = 'fabric_keywords.txt'
with open(file_path, 'r') as file:
    content = file.read()
    fabric_keywords = list(content.split(','))
```

Recognizing that fabric names are often compound, either to specify a particular style or for other reasons, we are introducing the `freq_based_in_compound_keywords` function. This function will enable the identification and frequency analysis of fabrics, which can be both singular and compound names, present in a given text.

In [73]:
```
def freq_based_in_singlecompound_keywords(text, keywords, sort=True):
    """
    Analyzes the frequency of single/compound keywords in a given text.

    Parameters:
    - text (str): The text to analyze.
    - keywords (list): A list of compound keywords to search for in the text.

    Returns:
    - dict: A dictionary containing the compound keywords found in the text
            and their respective frequencies.

    Example:
    >>> text = "This fabric is made of cotton and polyester. Cotton-poly blends are popular."
    >>> keywords = ["cotton", "polyester", "cotton-poly"]
    >>> result = freq_based_in_compound_keywords(text, keywords)
    >>> print(result)
    Output: {'cotton': 2, 'polyester': 1, 'cotton-poly': 1}
    """

    frequency_based_keywords = {}
    for keyword in keywords:
        # Create a regular expression pattern for each keyword, ignoring case and matching whole words
        pattern = re.compile(fr'\b{keyword}\b', flags=re.IGNORECASE)
        # Find all occurrences of the keyword in the text
        occurrences = pattern.findall(text)
        # Store the frequency of the keyword in the dictionary
        frequency_based_keywords[keyword] = len(occurrences)

    # Sort the dictionary based on keyword frequency in descending order
    sorted_frequency = dict(sorted(
```

```
        frequency_based_keywords.items(), key=lambda item: item[1], reverse=True)) if sort else frequency_based_keywords

    # Create a final dictionary containing only keywords with non-zero frequency
    final_frequency_based_keywords = {
        key: value for key, value in sorted_frequency.items() if value != 0}

    return final_frequency_based_keywords
```

(a) Initially, an additional column will be introduced in each brand-specific DataFrame (`df_louisvuitton` , `df_chanel` , `df_dior` , `df_versace`) to store the set of two most relevant fabrics items for each entry in the `filtered_corpus` column. **(b) Following that**, in the global DataFrame `df_globalBrands` , which consolidates the four individual brand corpora, another column will be added to capture the set of the top 5 most relevant fabrics associated with each corpus. **(c) Finally**, the same procedure will be applied to the primary `df_articles` DataFrame: an extra column will be appended to record the most significant fabric present in each article.
The `freq_based_in_singlecompound_keywords()` function will be utilized.

In [74]:
```
#(a)
# Iterate over a list of DataFrames containing brand-related data
for dfName in list_df_brands:
    # Add a new column 'relevant_fabric' to each DataFrame
    # The column will contain the top two relevant fabric keywords based on the filtered_corpus
    dfName["relevant_fabric"] = dfName["filtered_corpus"].apply(
        lambda x: ",".join(list(freq_based_in_singlecompound_keywords(x, fabric_keywords).keys())[:2])
    )
```

Primary apparel for Louis Vuitton across different websites:

In [75]:
```
relevant_column_view(df_louisVuitton,"relevant_fabric","df_louisVuitton")
```

```
Displaying relevant_fabric view for DataFrame: df_louisVuitton
```

Out[75]:

		relevant_fabric
brand	**website**	
louis vuitton	**elle**	linen,tulle
	glamour	denim,leather
	harper's bazar	leather,wool
	vogue	leather,cotton

Primary apparel for Chanel across different websites:

In [76]:
```
relevant_column_view(df_chanel, "relevant_fabric","df_chanel")
```

```
Displaying relevant_fabric view for DataFrame: df_chanel
```

Out[76]:

		relevant_fabric
brand	**website**	
chanel	**elle**	tweed,felt
	glamour	denim,lace
	harper's bazar	tweed,felt
	vogue	tweed,felt

Primary apparel for Dior across different websites:

In [77]:
```
relevant_column_view(df_dior, "relevant_fabric","df_dior")
```

```
Displaying relevant_fabric view for DataFrame: df_dior
```

Out[77]:

		relevant_fabric
brand	**website**	
dior	**elle**	tweed,floral
	glamour	velvet
	harper's bazar	silk,velvet
	vogue	velvet,wool

Primary apparel for Versace across different websites:

In [78]:
```
relevant_column_view(df_versace, "relevant_fabric","df_versace")
```

```
Displaying relevant_fabric view for DataFrame: df_versace
```

Out[78]:

		relevant_fabric
brand	**website**	
versace	**elle**	tweed,leather
	glamour	leather,flocked
	harper's bazar	leather,floral
	vogue	denim,satin

In [79]:
```
#(b)
# Add a new column 'relevant_fabric' to the DataFrame df_globalBrands
# The column will contain the top five relevant fabric keywords based on the filtered_articles_corpus
df_globalBrands["relevant_fabric"] = df_globalBrands["filtered_articles_corpus"].apply(
    lambda x: ",".join(list(freq_based_in_singlecompound_keywords(x, fabric_keywords).keys())[:5]))
```

Essential fabrics for each brand:

In [80]:
```
df_globalBrands[["brand", "relevant_fabric"]]
```

Out[80]:

	brand	relevant_fabric
0	louis vuitton	leather,denim,felt,cotton,silk
1	chanel	tweed,felt,leather,lace,denim
2	dior	silk,velvet,wool,satin,tulle
3	versace	leather,denim,floral,satin,tweed

Recognizing the importance of maintaining a detailed record of identified fabric items in each corpus within the `"articles_corpus_final"` column of `df_globalBrands`, a process has been implemented to store these fabrics and their respective frequencies in four separate dictionaries, one for each brand.

In [81]:
```
# Calculate fabric frequencies for each brand using freq_basedOnKeywords_byIndex
freq_fabric_LV = freq_basedOnKeywords_byIndex(0, fabric_keywords)
freq_fabric_Chanel = freq_basedOnKeywords_byIndex(1, fabric_keywords)
freq_fabric_Dior = freq_basedOnKeywords_byIndex(2, fabric_keywords)
freq_fabric_Versace = freq_basedOnKeywords_byIndex(3, fabric_keywords)

# Example Result:
# freq_fabric_LV: {'silk': 3, 'cotton': 1}
# freq_fabric_Dior: {'silk': 2, 'cotton': 1,'leather':1}
#...
```

Recognizing the differences in frequencies of the fabric among the selected brands is crucial. To achieve this, we will design a DataFrame with three columns: `"Brand,"` `"Fabric,"` and `"Frequency."` This DataFrame will encompass all frequencies associated with identified fabrics for each brand. It is noteworthy that in the `"Fabric"` column, the same fabric may appear more than once, each instance linked to different brands. This approach enables us to analyze and comprehend variations in the frequencies of a specific fabric across different brands.

In [82]:
```
# Use createList_freq_items_by_brand to obtain a list of dictionaries
#containing fabric frequencies associated with each brand
dict_freq_apparel_global = createList_freq_items_by_brand(
    df_globalBrands, "articles_corpus_final", name_of_Brands, "Fabric", fabric_keywords)
# Create a DataFrame with columns "Brand", "Fabric", and "Frequency" from the generated dictionary
df_frequency_fabric_by_brand = pd.DataFrame(dict_freq_apparel_global)
```

In [83]:
```
#(c)
df_articles["relevant_fabric"] = df_articles["Filtered_content"].apply(
    lambda x : "".join(list(freq_based_in_singlecompound_keywords(x, fabric_keywords).keys())[:1])
)

print("What are the titles of the articles where the dominant fabric is 'cotton'?")
get_titles_by_parameter(df_articles, "relevant_fabric", "silk", selectDFCol=["title","brand","relevant_fabric"])
```

```
What are the titles of the articles where the dominant fabric is 'cotton'?
```

Out[83]:

	title	brand	relevant_fabric
1	How Ana de Armas's Golden Globes Dress Channel...	louis vuitton	silk
57	Dior Collaborates with Artist Mickalene Thomas	dior	silk
58	Paris Couture: Should Designers Roar or Whisper?	dior	silk

(9)The most dominant fabric in the World of Fashion according to Louis Vuitton, Chanel, Dior, and Versace: To determine the most dominant fabric throughout the year in the world of fashion, we will focus on the selected luxury fashion brands (Louis Vuitton, Chanel, Dior, and Versace) based on the available articles. To achieve this, we will consolidate all these articles using the `filtered_articles_corpus` column from df_globalBrands, which encompasses all the articles associated with each brand.

These articles will be amalgamated into a unified corpus. The subsequent step involves applying the `def freq_based_in_singlecompound_keywords` method and utilizing the `fabric_keywords` list. This process will enable us to pinpoint the most prominent fabric among all the articles linked to these prestigious fashion brands.

In [84]:
```
# Use the 'freq_based_in_singlecompound_keywords' function to calculate the
#frequency of fabrics in the unified corpus
frequency_fabrics_global = freq_based_in_singlecompound_keywords(
    final_corpus, fabric_keywords,sort=False)
# Create a DataFrame from frequency_apparel_global
df_frequency_fabric_global = pd.DataFrame(
    list(frequency_fabrics_global.items()), columns=['Fabric', 'Frequency'])
```

THE WINNER

In [85]:
```
df_frequency_fabric_global.loc[df_frequency_fabric_global["Frequency"].idxmax()]
```

Out[85]:
```
Fabric       leather
Frequency         17
Name: 12, dtype: object
```

The winning fabric in the data exploration was 'leather' with a frequency of 17. This result indicates that leather were the most dominant fabric throughout the year according to Louis Vuitton, Chanel, Dior, and Versace.

4.4 Sentiment Analysis

(10) To conclude this section, we will delve into the sentiment analysis phase using the NLTK sentiment analyzer with the VADER lexicon. This process will enable us to quantify the emotions underlying the narratives in the articles, representing them across three measures: negative, neutral, and positive. These measures will be expressed on a scale ranging from **-1 (indicating negative sentiment) to 1 (indicating positive sentiment)**.
This step is particularly significant as it allows for the identification of the emotional tone, both positive and negative, associated with a brand through the content of an article.

In [86]:
```
nltk.download('vader_lexicon')
analyser = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\aloza\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [87]:
```
# Applying sentiment analysis to the "content" column and updating sentiment-related columns
df_articles[["Neg_Sentiment", "Neu_Sentiment", "Pos_Sentiment", "Compound_Sentiment"]] = \
df_articles["content"].apply(
    lambda x: pd.Series(list(analyser.polarity_scores(x).values()))
)
```

To streamline this analysis across all selected fashion brands, these sentiment scores have been incorporated into the DataFrame `df_global_Brands`.

In [88]:
```
# Create new columns to represent sentiment tones
df_globalBrands[["Neg_Sentiment", "Neu_Sentiment", "Pos_Sentiment", "Compound_Sentiment"]] = \
```

```
df_globalBrands["articles_corpus_final"].apply(
    lambda x: pd.Series(list(analyser.polarity_scores(x).values()))
)
```

Sentiment Trends in Luxury Brand Narratives (Compound Analysis): It is crucial to identify the luxury brand for which the narratives in the articles tend to be more positive or negative. To achieve this, we will evaluate the compound sentiments associated with each brand. This process will allow us to calculate the average of the compound sentiment scores for each brand. Subsequently, we will determine which brand has the highest or lowest average, indicating a greater positive or negative inclination, respectively.

In [89]:
```
# Calculate the average sentiment scores for each brand
average_sentiments = df_articles.groupby('brand')['Compound_Sentiment'].mean()

# Find the brand with the highest average sentiment score (most positive)
most_positive_brand = average_sentiments.idxmax()
# Find the brand with the lowest average sentiment score (most negative)
most_negative_brand = average_sentiments.idxmin()

print("Most positive brand:", most_positive_brand.title())
print("Most negative brand:", most_negative_brand.title())
```

```
Most positive brand: Chanel
Most negative brand: Louis Vuitton
```

Later on, we will formally visualize the distributions of these compound sentiment measures to provide a detailed overview of these results.

5. Visual Interpretation of Results

In this section, various graphs will be employed to effectively and attractively present the results obtained throughout the analysis. These visual representations will range from the analysis of characteristic adjectives for each brand to the exploration of entities closely related to them. Additionally, the relevance of terms such as "*glamour*," "*iconic*" and "*beauty*" for each brand will be unveiled, providing a detailed insight into how these brands are perceived in the fashion industry. Then, the relevance that the most dominant clothing items and fabrics had will be visualized. We will conclude this section with a graphical presentation that simplifies and enhances the understanding of the sentiments expressed in the articles. The use of visual representations is justified as a strategy to facilitate the interpretation of complex results, enabling a quick and profound understanding of the data analysis outcomes.

5.1 WordClouds

(1) Adjectives: This phase will visually present distinctive adjectives that independently characterize each luxury brand. We will utilize DataFrames such as `"df_adjectives_LV"`, `"df_adjectives_Chanel"`, `"df_adjectives_Dior"`, and `"df_adjectives_Versace"`, which contain previously identified adjectives for each luxury brand. The central strategy will involve creating an individual `Word Cloud` for each brand. To enhance visual comprehension and ensure an appealing presentation, we will incorporate a distinctive mask for each `Word Cloud`, using the corresponding brand's logo.

In [90]:
```
def create_Masked_Wordcloud(text, font_path,
                            maskimage_path=None, backgroundColor="white", contourCol="black",
                            colormap="cool_r", collocations=True):

    """
    Creates a Word Cloud with a mask using a specific image.

    Parameters:
    - text (str): The text to be visualized in the Word Cloud.
    - font_path (str): The path to the typography font file to be used.
    - maskimage_path (str): The path to the image that will be used as a mask to shape the Word Cloud.
    - backgroundColor (str, optional): The background color of the Word Cloud. Default is "white".
    - contourCol (str, optional): The contour color of the Word Cloud. Default is "black".
    - colormap (str, optional): The colormap to be used for visualization. Default is "cool_r".

    Returns:
    None: Displays the Word Cloud with the mask in the graphical interface.

    Example:
    >>> create_Masked_Wordcloud("Sample text", "fonts_path/font.ttf", "mask_path/mask.png")
    Creates and displays a Word Cloud using the specified mask and provided text.
    """

    # Opens the mask image
    mask = np.array(PIL.Image.open(maskimage_path)) if maskimage_path else None
    # Creates the Word Cloud with the mask and specified parameters
    wordcloud = WordCloud(mask=mask,
                          background_color=backgroundColor,
                          contour_color=contourCol,
                          contour_width=1,
                          min_font_size=9,
                          colormap=colormap,
                          font_path=font_path,
                          collocations=collocations,
    ).generate(text)
    # Displays the Word Cloud in the graphical interface
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()
```

Masked Word Cloud for Louis Vuitton

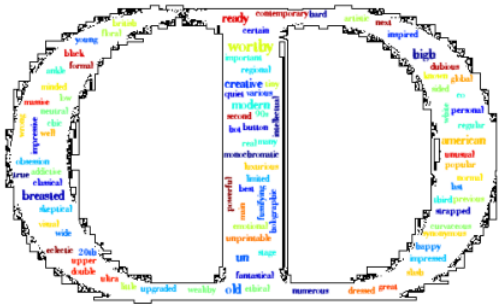
In [91]:
```
create_Masked_Wordcloud(",".join(
    df_adjectives_LV["Adjective"]), "font/SIXTY.TTF", backgroundColor="Black", contourCol="White")

create_Masked_Wordcloud(",".join(df_adjectives_LV["Adjective"]), "font/SIXTY.TTF",
                        maskimage_path="Images/brands/lv.png",backgroundColor="Black", contourCol="White")
```

Masked Word Cloud for Chanel

In [92]:
```
create_Masked_Wordcloud(",".join(df_adjectives_Chanel["Adjective"]), "font/SIXTY.TTF",
                        colormap="PiYG_r", contourCol="White",backgroundColor="Black")

create_Masked_Wordcloud(",".join(df_adjectives_Chanel["Adjective"]), "font/SIXTY.TTF",
                        maskimage_path="Images/brands/chanel.jpg", colormap="PiYG_r")
```

Masked Word Cloud for Dior

In [93]:
```
create_Masked_Wordcloud(",".join(
    df_adjectives_Dior["Adjective"]), "font/SIXTY.TTF", colormap="jet",
                        backgroundColor="Black", contourCol="White")

create_Masked_Wordcloud(",".join(
    df_adjectives_Dior["Adjective"]), "font/SIXTY.TTF", maskimage_path="Images/brands/dior.jpg",
                        colormap="jet")
```



Masked Word Cloud for Versace

In [94]:
```
create_Masked_Wordcloud(",".join(
    df_adjectives_Versace["Adjective"]), "font/SIXTY.TTF" ,colormap="autumn_r",
                        backgroundColor="black", contourCol="white")

create_Masked_Wordcloud(",".join(
    df_adjectives_Versace["Adjective"]), "font/SIXTY.TTF", maskimage_path="Images/brands/versace.png",
                        colormap="autumn_r",backgroundColor="black", contourCol="white")
```

(2) Entities: This section will visually present entities that were closely related this year to each of the fashion brands. For this purpose, the DataFrame `df_globalBrands` will be utilized, as it encompasses the corpora of the four selected brands, containing previously identified entities for each luxury brand. The central strategy will involve creating an individual `Word Cloud` for each brand. To enhance visual comprehension and ensure an appealing presentation, we will incorporate a distinctive mask for each `Word Cloud`, using the corresponding brand's logo. Once again, the function `create_Masked_Wordcloud`, will be employed for this task.

Masked Word Cloud for Louis Vuitton

In [95]:
```
create_Masked_Wordcloud(df_globalBrands["entities_involved"][0],
                        font_path="font/Icecreamypersonaluse Bold-3zgoy.otf")
create_Masked_Wordcloud(df_globalBrands["entities_involved"][0],
                        font_path="font/Icecreamypersonaluse Bold-3zgoy.otf",
                        maskimage_path="Images./brands/lv.png")
```

Through this word cloud, the prominent presence of the term "Marc Jacobs" is discernible. **Marc Jacobs**, born on April 9, 1963, in New York, USA, is an influential American fashion designer renowned for his distinctive interpretations of popular culture trends. Notably, he gained widespread recognition for his groundbreaking "grunge" collection [27], credited with initiating the grunge style movement in the 1990s. The designer is the creative force behind his eponymous brand, Marc Jacobs, as well as a secondary line called Marc by Marc Jacobs, boasting a global presence with 200 stores across 80 countries. Marc Jacobs achieved a remarkable milestone in 1987 when he became the youngest designer ever to receive the Perry Ellis Award for New Fashion Talent from the Council of Fashion Designers of America (CFDA) [28]. Additionally, Marc Jacobs served as the creative director of the luxury fashion brand **Louis Vuitton** from 1997 to 2014 [29].

Marc Jacobs at Louis Vuitton: Most Iconic Catwalk Moments


Marc Jacobs LV


Marc Jacobs LV


Marc Jacobs LV

Masked Word Cloud for Chanel

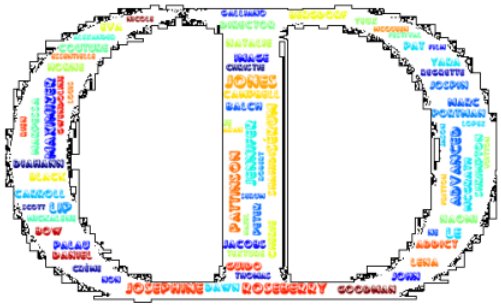
In [96]:
```
create_Masked_Wordcloud(
    df_globalBrands["entities_involved"][1],
    font_path="font/IcecreamypersonaluseBold-3zgoy.otf",
    colormap="PiYG_r")
create_Masked_Wordcloud(df_globalBrands["entities_involved"][1],
                        font_path="font/IcecreamypersonaluseBold-3zgoy.otf",
                        maskimage_path="Images./brands/chanel.jpg",
                        colormap="PiYG_r")
```

*Upon exploring the word cloud, the name **Gwyneth Kate Paltrow** emerges—a distinguished American actress and singer. Paltrow has garnered numerous accolades, including an Oscar, a Golden Globe, and two Screen Actors Guild Awards, all bestowed for her outstanding portrayal of Viola de Lesseps in the film 'Shakespeare in Love.' Additionally, in 2010, she was honored with a star on the Hollywood Walk of Fame. [30]*


Marc Jacobs LV
Gwyneth Paltrow at the 2011 Venice Film Festival

Masked Word Cloud for Dior

In [97]:
```
create_Masked_Wordcloud(df_globalBrands["entities_involved"][2],
                        font_path="font/Icecreamypersonaluse Bold-3zgoy.otf", colormap="jet")
create_Masked_Wordcloud(df_globalBrands["entities_involved"][2],
                        font_path="font/Icecreamypersonaluse Bold-3zgoy.otf",
                        maskimage_path="Images./brands/dior.jpg", colormap="jet")
```



*Through the word cloud analysis, the presence of **Lee Alexander McQueen**, a notable English fashion designer, was observed. McQueen gained recognition for his role as the chief designer at Givenchy from 1996 to 2001 and for founding his own eponymous label, Alexander McQueen. [31]*

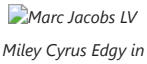

Marc Jacobs LV

Alexander McQueen, SS24 IK ALDAMA

Masked Word Cloud for Versace

In [98]:
```
create_Masked_Wordcloud(
    df_globalBrands["entities_involved"][3],
    font_path="font/Icecreamypersonaluse Bold-3zgoy.otf",
    colormap="autumn_r")
create_Masked_Wordcloud(df_globalBrands["entities_involved"][3],
                        font_path="font/Icecreamypersonaluse Bold-3zgoy.otf",
                        maskimage_path="Images./brands/versace.png",
                        colormap="autumn_r",
                        backgroundColor="Black",contourCol="White")
```

*In the Versace word cloud, the name **Miley Ray Cyrus** emerged. **Miley Ray Cyrus** is an American singer, songwriter, record producer, and actress known for her distinctive deep-toned voice. Her musical repertoire spans a wide range of genres, including pop, country, hip hop, experimental, and rock. [32]*


Marc Jacobs LV

Miley Cyrus Edgy in Ombre to Versace Fashion Show 2023 in Los Angeles

5.2 Bar Charts

(3) Adjective Frequency Bar Charts: *In this phase, bar charts detailing the identified adjectives for each brand and their respective frequencies will be presented. We will utilize the DataFrames (`df_adjectives_LV`, `df_adjectives_Chanel`, `df_adjectives_Dior`, and `df_adjectives_Versace`) generated during the `"(1) Exploring adjectives associated with the verb 'Look'"` phase in the `"4. NLP Analysis of Brand Perception"` section. This visualization aims to provide a deeper understanding of brand perceptions in terms of commonly used adjectives.*

To achieve this, `Plotly` will be employed, a library chosen for its versatility in generating visually appealing and informative bar charts. Plotly's set of style and color options enhances the clarity and attractiveness of the graphics, contributing to a more engaging interpretation of the data.

In [99]:
```
def create_bar_chart(dfName, x, y, color, color_discrete_sequence,
                     title_text,textBar=None, marker_line_color = None,
                     marker_line_width = None,color_text="Black",
                     size_text=7,showBackGrid=True):
    """
    Creates a bar chart using Plotly Express.

    Parameters:
    - dfName (pd.DataFrame): The DataFrame containing the data.
    - x (str): The column name for the x-axis.
    - y (str): The column name for the y-axis.
    - color (str): The column name for color differentiation.
    - color_discrete_sequence (list): The color palette for the bars.
    - title_text (str): The title of the chart.
    - marker_line_color (str, optional): The color of the marker lines for the bars.
    - marker_line_width (float, optional): The width of the marker lines for the bars.
    - color_text (str, optional): The color of the text on the chart.

    Example:
    # Sample DataFrame
    df_adjectives_LV
    # Create a bar chart
    create_bar_chart(df_adjectives_LV, x='Adjective', y='Frequency', color='Adjective',
                     color_discrete_sequence=['#FF0000', '#00FF00', '#0000FF', '#FFFF00'...],
                     title_text='Sample Bar Chart',
                     marker_line_color='Black', marker_line_width=1.5, color_text='White')

    Note:
    - The function utilizes Plotly Express (`px.bar`) for creating the bar chart.
    - `color_discrete_sequence` should be a list of color codes corresponding to the unique values in the 'color' column.
    - `marker_line_color` and `marker_line_width` are optional parameters for customizing bar borders.
    - `color_text` sets the color of the text on the chart.
    """

    # Create a bar chart using Plotly Express
    figure_bar = px.bar(dfName, x=x, y=y,
                        color=color,
                        color_discrete_sequence=color_discrete_sequence,
                        text=textBar)

    # Update the layout of the chart, including title and font settings
    figure_bar.update_layout(title_text= title_text,
                     title_x=0.5,  # Center the title horizontally
                     title_y=0.94,  # Adjust the vertical position
                              font=dict(family='Old Standard TT, serif', size=size_text, color=color_text),
                              yaxis=dict(showgrid=showBackGrid))

    # Update the traces of the chart, including marker line settings
    figure_bar.update_traces(marker_line_color=marker_line_color,
                              marker_line_width=marker_line_width)
    # Display the chart
    figure_bar.show()
```

In [100…
```
def moreThanN(dfName,n):
    """
    Filters a DataFrame to include only rows where the 'Frequency' column has values greater than n.

    Parameters:
    - dfName (pd.DataFrame): The DataFrame containing the data.
    - n (int): The threshold value for filtering.

    Returns:
    - pd.DataFrame: A filtered DataFrame containing rows with values greater than N in the specified column.
    """
    return dfName.loc[dfName["Frequency"]>n]
```

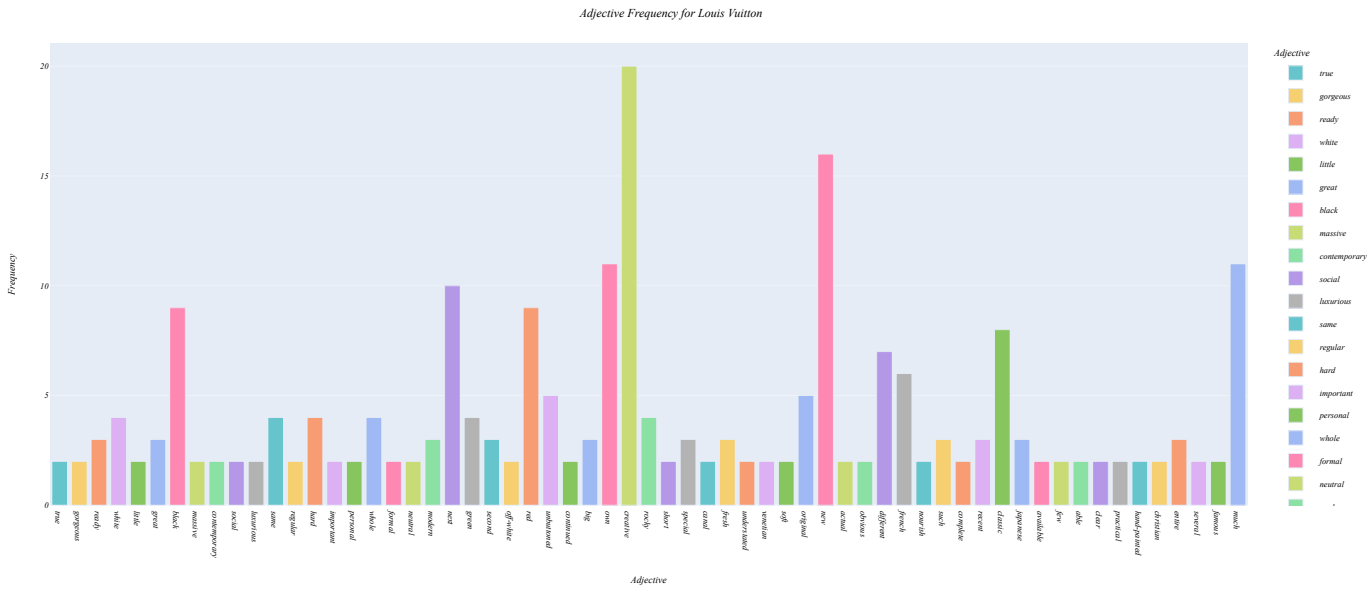
The `moreThan1` function is designed to filter a DataFrame and return only the rows that meet the condition of having a value greater than N in the `"Frequency"` column. This approach is adopted with the purpose of emphasizing the relevance of data that occurs more than N times. This way, a clearer and more meaningful representation of the data is achieved, eliminating redundancy associated with lower occurrences.

In [101…
```
# Generate color swatches using Plotly Express
# use for color_discrete_sequence
fig = px.colors.qualitative.swatches()
fig.show()
```

Adjective Frequency Bar Chart for Louis Vuitton

In [102…

```
df_adjectives_LV_morethan1 = moreThanN(df_adjectives_LV,1)

create_bar_chart(df_adjectives_LV_morethan1, x="Adjective",
                 y="Frequency",
                 color="Adjective",
                 color_discrete_sequence=px.colors.qualitative.Pastel,
                 title_text="Adjective Frequency for Louis Vuitton")
```

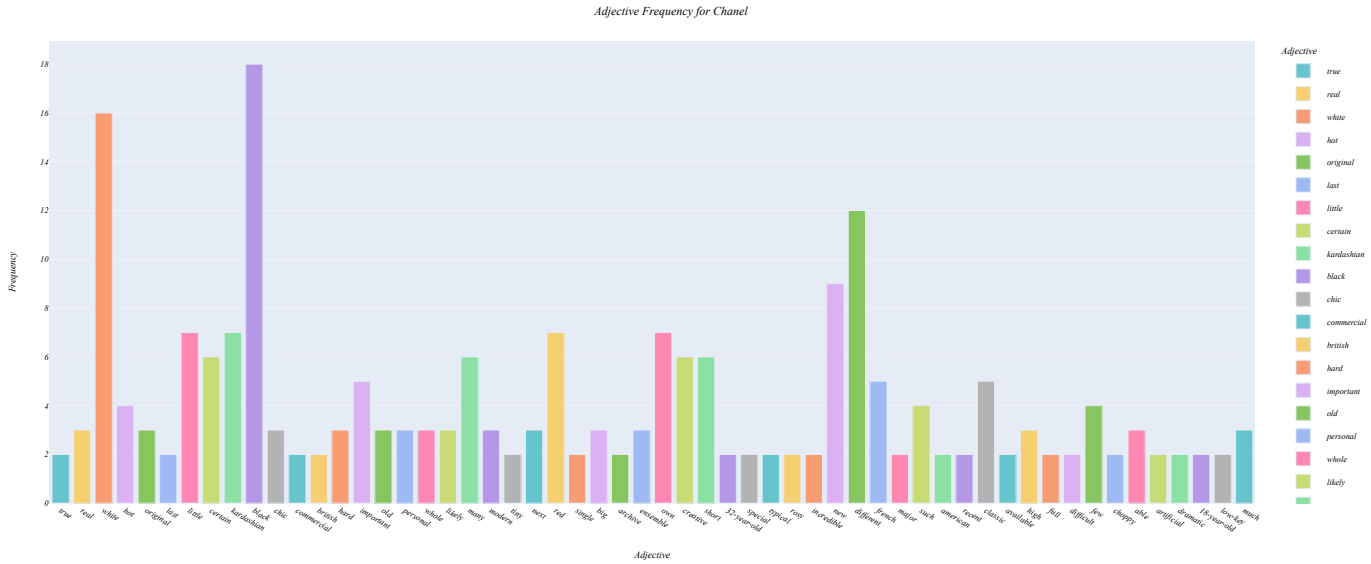


Let's observe that the most frequent adjectives were "creative" and "new."
Louis Vuitton was a brand recognized for its creativity this year.

Adjective Frequency Bar Chart for Chanel

In [103…

```
df_adjectives_Chanel_morethan1 = moreThanN(df_adjectives_Chanel,1)
create_bar_chart(df_adjectives_Chanel_morethan1,
                 x="Adjective", y="Frequency", color="Adjective",
                 color_discrete_sequence=px.colors.qualitative.Pastel,
                  title_text="Adjective Frequency for Chanel")
```

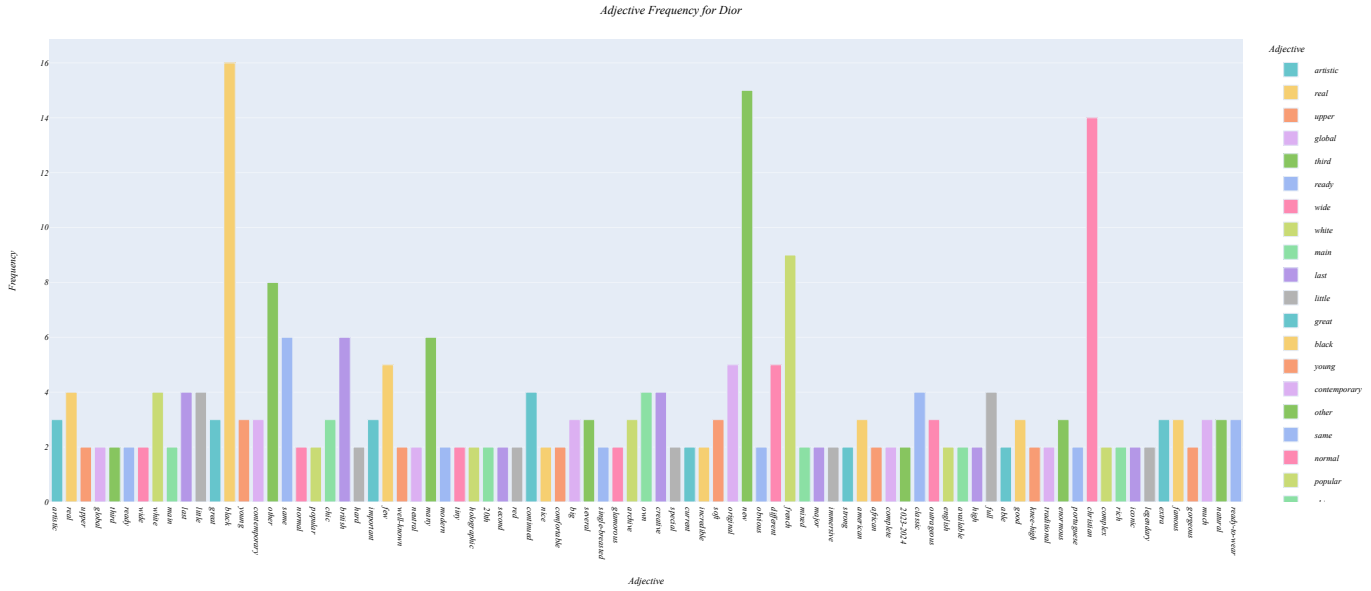


Let's observe that the most frequent adjectives were "black" and "white." Undoubtedly, many of the luxury items designed for Chanel this year were characterized by being "white" or "black." A minimalist yet interesting trend.

Adjective Frequency Bar Chart for Dior

In [104…

```
df_adjectives_Dior_morethan1 = moreThanN(df_adjectives_Dior,1)
create_bar_chart(df_adjectives_Dior_morethan1,
                 x="Adjective", y="Frequency", color="Adjective",
                 color_discrete_sequence=px.colors.qualitative.Pastel,
                  title_text="Adjective Frequency for Dior")
```

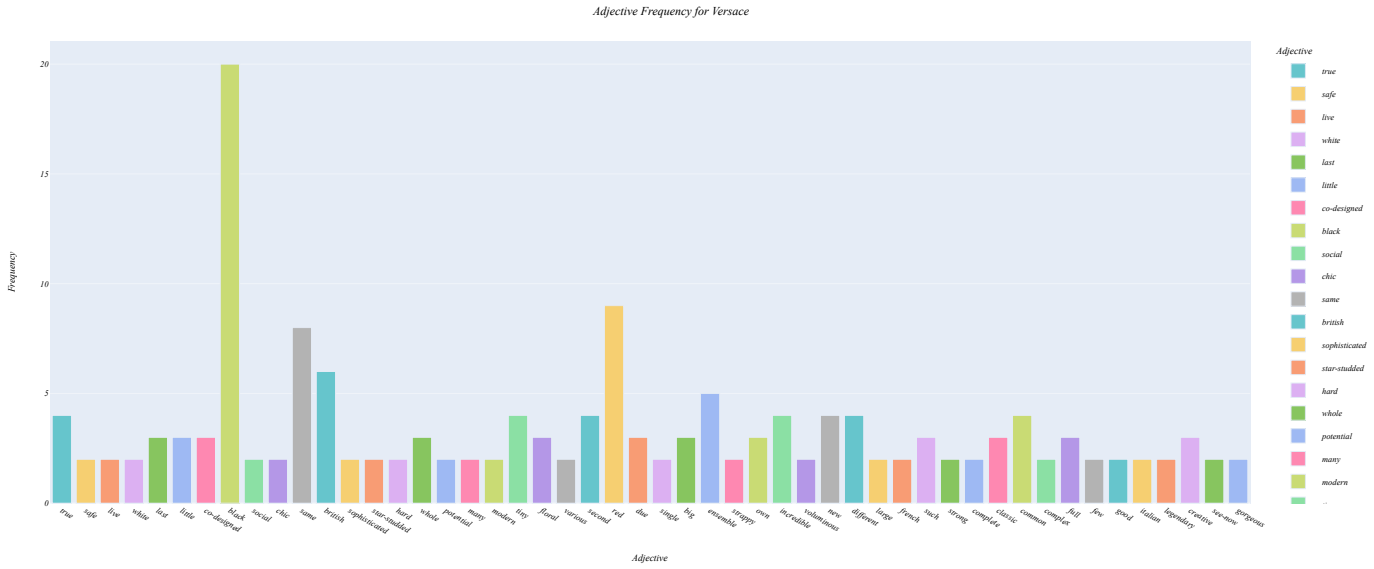


Let's observe that the most frequent adjectives were "black" and "new" for Dior. This trend could be attributed to the brand's commitment to modernity and a sophisticated aesthetic. The use of the color "black" may signify a timeless elegance, while the emphasis on "new" reflects Dior's dedication to staying at the forefront of fashion, introducing fresh and innovative designs to captivate the audience.

Adjective Frequency Bar Chart for Versace

In [105…

```
df_adjectives_LV_morethan1 = moreThanN(df_adjectives_Versace,1)
create_bar_chart(df_adjectives_LV_morethan1,
                 x="Adjective", y="Frequency",
                  color="Adjective",
                 color_discrete_sequence=px.colors.qualitative.Pastel,
                  title_text="Adjective Frequency for Versace")
```



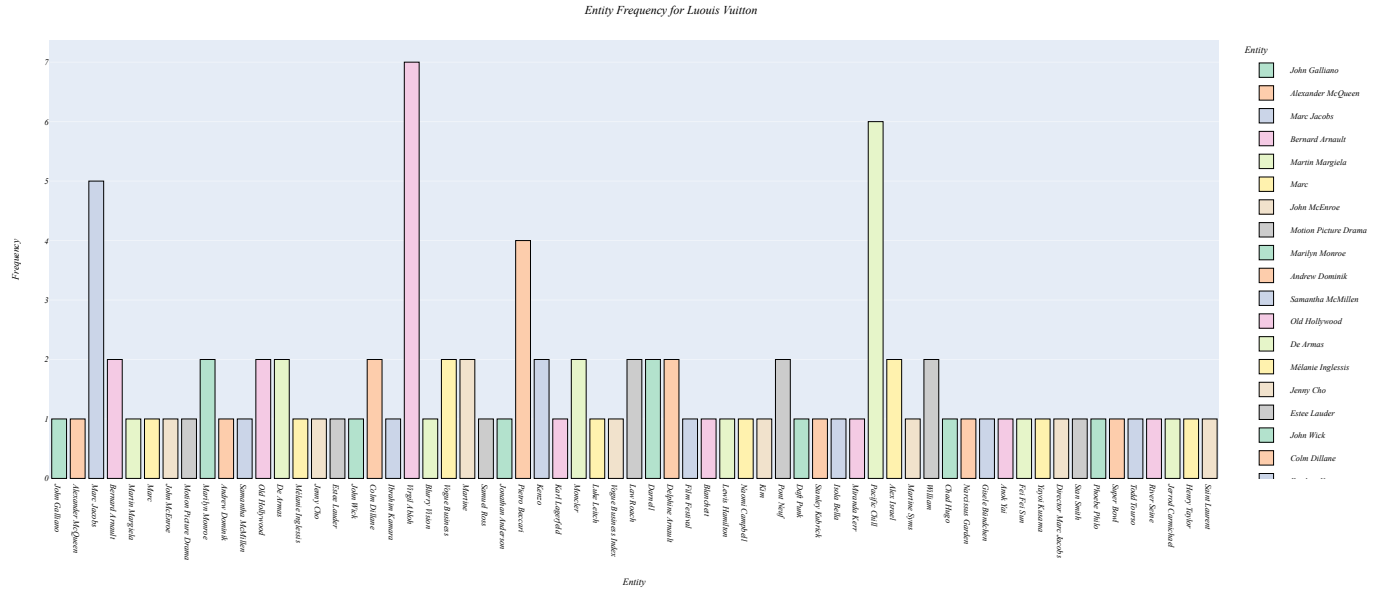
Let's observe that the most frequent adjectives were "black" and "last". This trend might be indicative of Versace's focus on enduring style and timeless sophistication. The use of the color "black" could symbolize a classic and versatile approach to fashion, while the emphasis on "last" suggests a commitment to creating pieces that transcend fleeting trends and possess a lasting appeal.

***(4) Frequency of entities related to each brand**: In a prior analysis, we observed that Marc Jacobs stood out among all entities associated with Louis Vuitton. However, a compelling question arises: Is there another entity that has been noticeably dominant this year for Louis Vuitton? Additionally, we noted that Miley Cyrus took center stage among entities related to Versace, but is she the sole protagonist? These inquiries will be addressed through the graphical representation of distinct bar charts, showcasing entities and their respective frequencies for each selected brand. We will leverage the previously created DataFrames (`df_entities_LV` , `df_entities_Chanel` , `df_entities_Dior` , `df_entities_Versace`) from the `(2) Reconnaissance of Proper Nouns and Entities` phase in Section `4. NLP Analysis of Brand Perception` . These visualizations will offer a comprehensive view of the entities that have taken the spotlight in the fashion world this year for Louis Vuitton, Chanel, Dior, and Versace.*

Entity Frecuency Bar Chart for Louis Vuitton

In [106…

```
create_bar_chart(df_entities_LV,
                 x="Entity", y="Frequency", color="Entity",
                 color_discrete_sequence=px.colors.qualitative.Pastel2,
                 title_text="Entity Frequency for Luouis Vuitton",
                  marker_line_color="Black",
                  marker_line_width=.05)
```

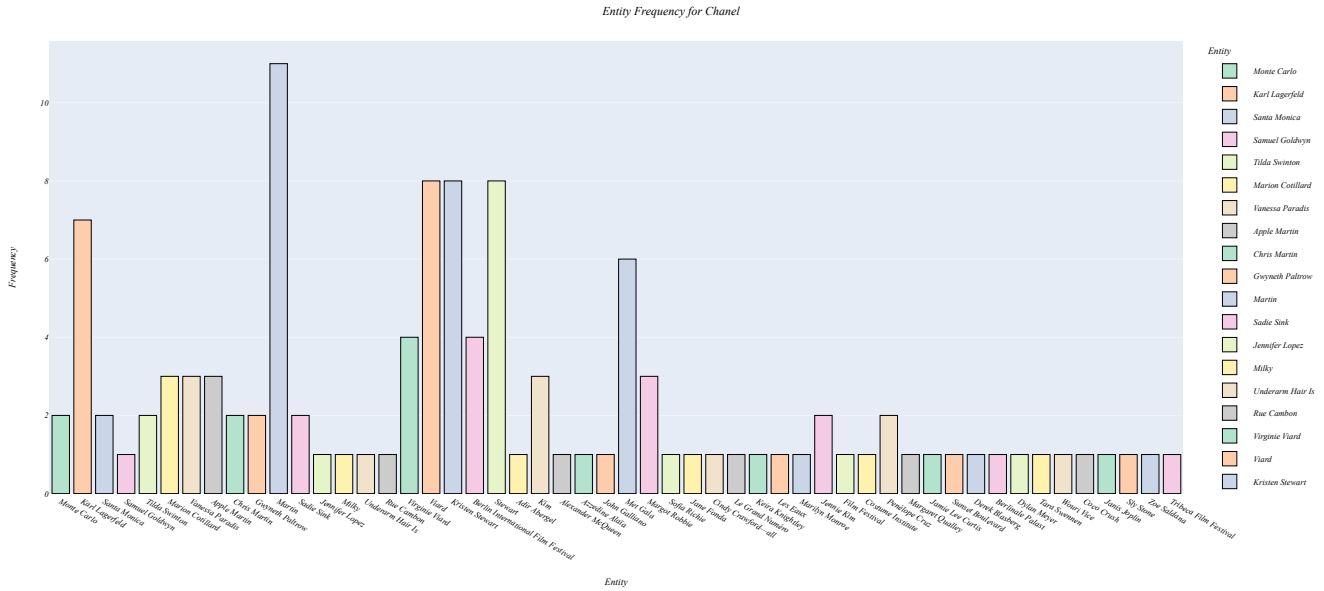


Let's observe that the most frequent entity was Virgil Abloh. Virgil Abloh was an American fashion designer, DJ, and producer, serving as the creative director for Louis Vuitton's men's collection from March 2015 until his passing. He also held the position of CEO for the Off-White brand until his death. Super interesting! Despite his passing, his designs persist to this day, serving as a meaningful, representative, and admirable memory in the fashion industry.

Entity Frecuency Bar Chart for Chanel

In [107…

```
create_bar_chart(df_entities_Chanel,
                 x="Entity", y="Frequency", color="Entity",
                 color_discrete_sequence=px.colors.qualitative.Pastel2,
                 title_text="Entity Frequency for Chanel",
                 marker_line_color="Black", marker_line_width=.05)
```

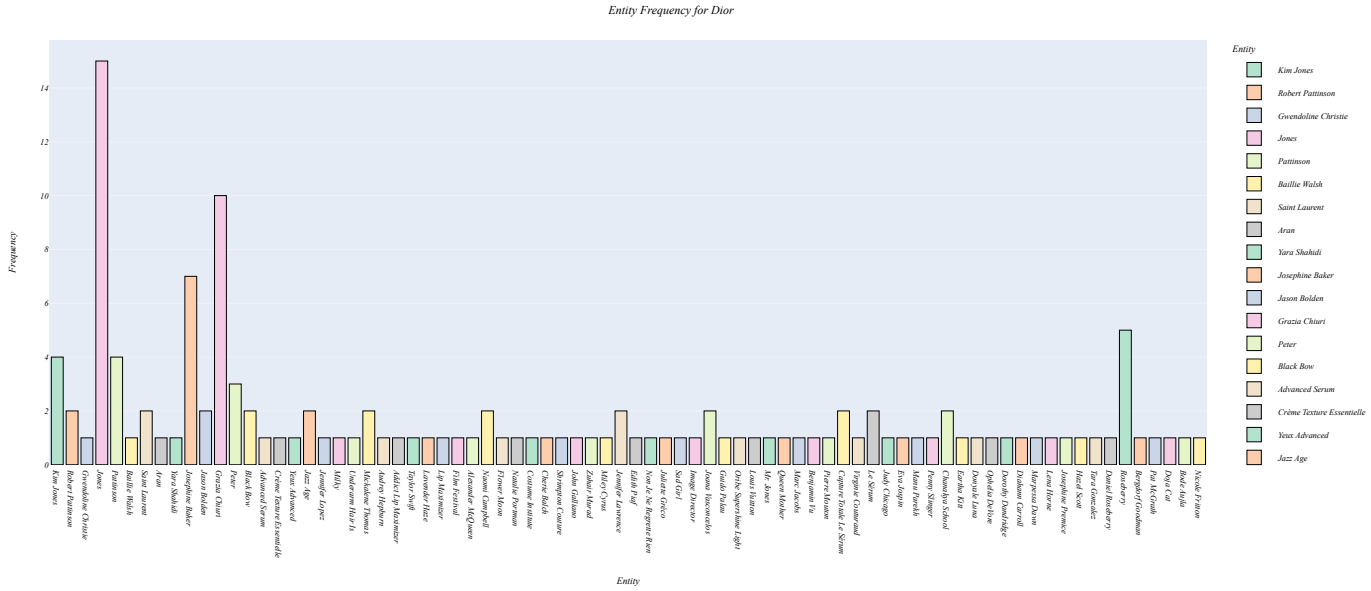


Let's observe that the most frequent entity was "Martin," the last name belonging to Gwyneth Paltrow's eldest daughter, Apple Martin. Gwyneth is an American actress and singer, having won an Oscar, a Golden Globe, and two Screen Actors Guild Awards. Her daughter is undoubtedly as iconic as she is, being recognized by Chanel as a prominent figure.

Entity Frecuency Bar Chart for Dior

In [108…

```
create_bar_chart(df_entities_Dior,
                 x="Entity", y="Frequency", color="Entity",
                 color_discrete_sequence=px.colors.qualitative.Pastel2,
                 title_text="Entity Frequency for Dior",
                 marker_line_color="Black", marker_line_width=.05)
```

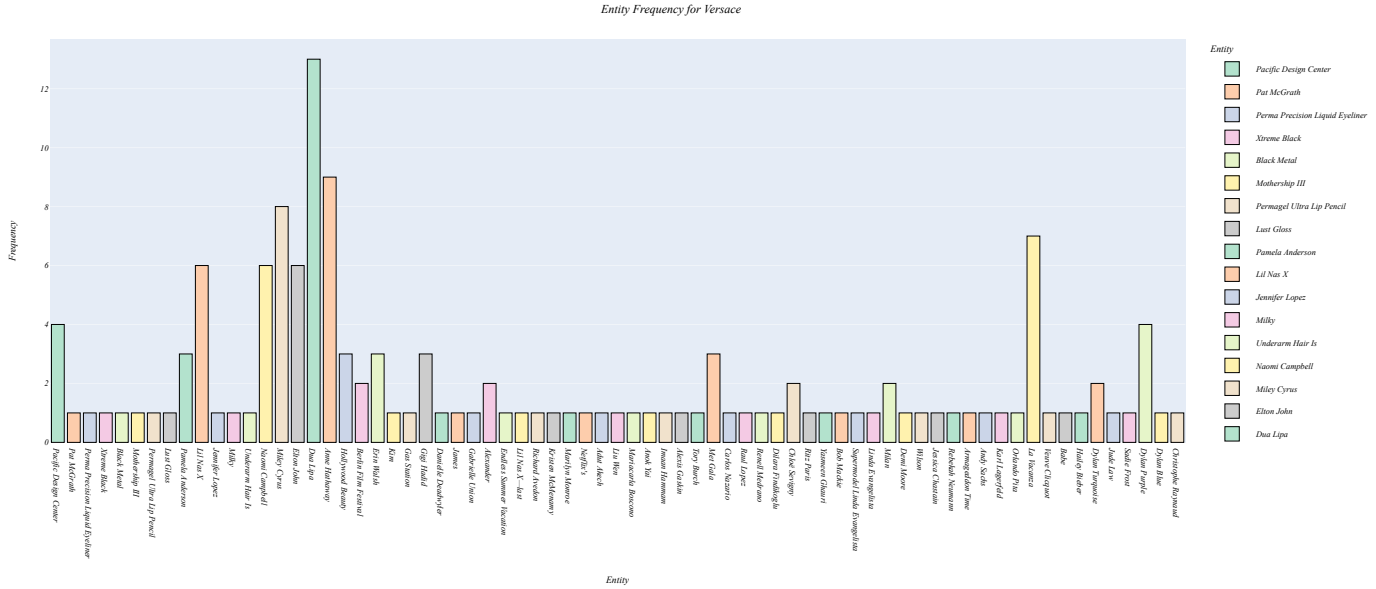


Let's observe that the most frequent entity was "Jones," a renowned surname in the world of fashion, belonging to Kim Niklas Jones OBE. He is an English fashion designer and a graduate of Central St Martins College of Art and Design.

Entity Frecuency Bar Chart for Versace

In [109…

```
create_bar_chart(df_entities_Versace,
                 x="Entity", y="Frequency", color="Entity",
                 color_discrete_sequence=px.colors.qualitative.Pastel2,
                 title_text="Entity Frequency for Versace",
                 marker_line_color="Black", marker_line_width=.05)
```



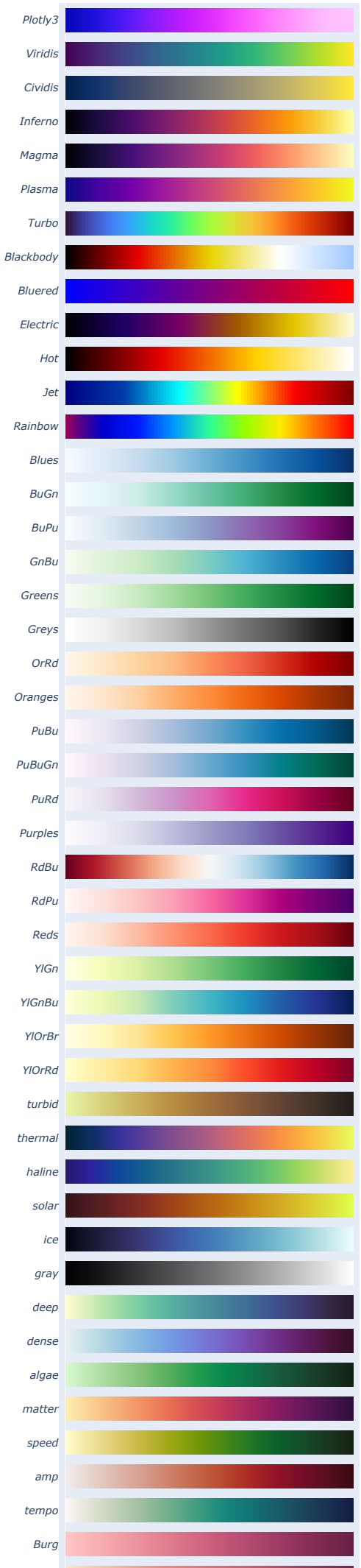
Let's observe that the most frequent entity for Versace was "Dua Lipa." Dua Lipa, an internationally acclaimed singer and songwriter, has not only been a prominent figure in the music industry but has also become a notable presence in the fashion world.

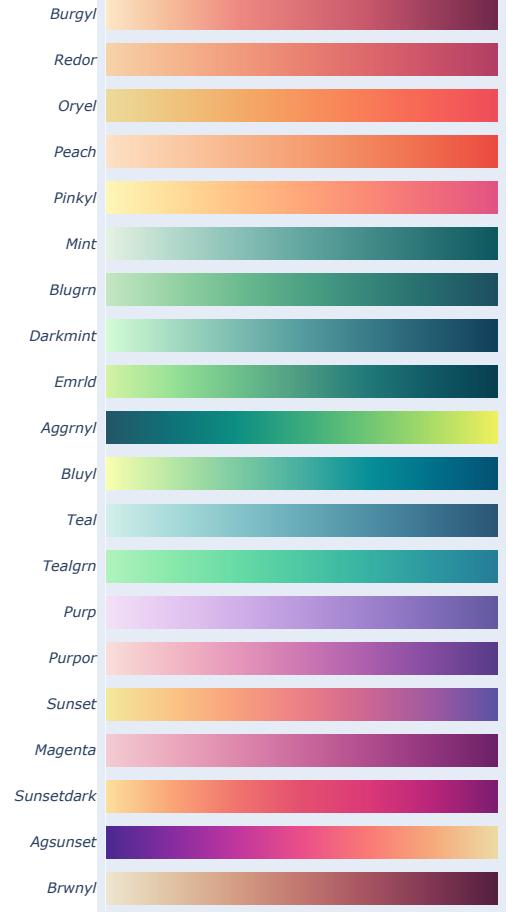
***(5) Relevance of Glamour, Iconic & Beauty by Brand**: In this phase, our focus will be on visualizing the significance of the words 'glamour,' 'iconic,' and 'beauty' in the articles when referring to each of the selected luxury brands.To conduct this analysis, we will utilize the DataFrame `df_globalBrands`, which contains the frequencies of these words for each brand in the columns `'glamour_frequency,'` `'iconic_frequency,'` and `'beauty_frequency.'` These columns were implemented during the `'(3) Glamour'` phase in Section `4. NLP Analysis of Brand Perception`.*

In [110…

```
# Generate continuous color swatches using Plotly Express
fig = px.colors.sequential.swatches_continuous()
fig.show()
```

plotly.colors.sequential





In [111…

```python
def create_bar_chart_for_wordFreq(dfName, x, y, list_of_hover_data, color, labelsXY, color_continuous_scale, title_text):
    """
    Creates a bar chart to visualize word frequencies.

    Parameters:
    - dfName (pd.DataFrame): The DataFrame containing the data.
    - x (str): The column name for the x-axis.
    - y (str): The column name for the y-axis.
    - list_of_hover_data (list): List of additional data to be displayed on hover.
    - color (str): The column name for color differentiation.
    - labelsXY (list): List containing two strings for x and y-axis labels.
    - color_continuous_scale (str): The color scale for continuous color data.
    - title_text (str): The title of the chart.

    Example:
    ```python
    import pandas as pd

    # Sample DataFrame
    data = {'Brand': ['Louis Vuitton', 'Chanel', 'Dior'],
            'glamour_frequency': [20, 15, 25],
            'otherDetails_1': [a, b, c],
            'otherDetails_2': [d, e, f]}
    df = pd.DataFrame(data)

    # Create a word frequency bar chart
    create_bar_chart_for_wordFreq(df, x='Brand', y='glamour_frequency',
                                  list_of_hover_data=['otherDetails_1', 'otherDetails_2'],
                                  color='glamour_frequency',
                                  labelsXY=['Luxury Brands', 'Word Frequency'],
                                  color_continuous_scale='Plotly3',
                                  title_text='"Glamour" Frequency Analysis')
    ```

    Note:
    - The function utilizes Plotly Express (`px.bar`) for creating the bar chart.
    - `list_of_hover_data` is a list of additional data columns to be displayed on hover.
    - `labelsXY` provides labels for the x and y axes.
    - `color_continuous_scale` determines the color scale for continuous color data.
    """
    wordFreq_barChart = px.bar(
        dfName, x=x, y=y, hover_data=list_of_hover_data, color=color,
        labels={x: labelsXY[0], y: labelsXY[1]}, height=400,
        color_continuous_scale=color_continuous_scale)

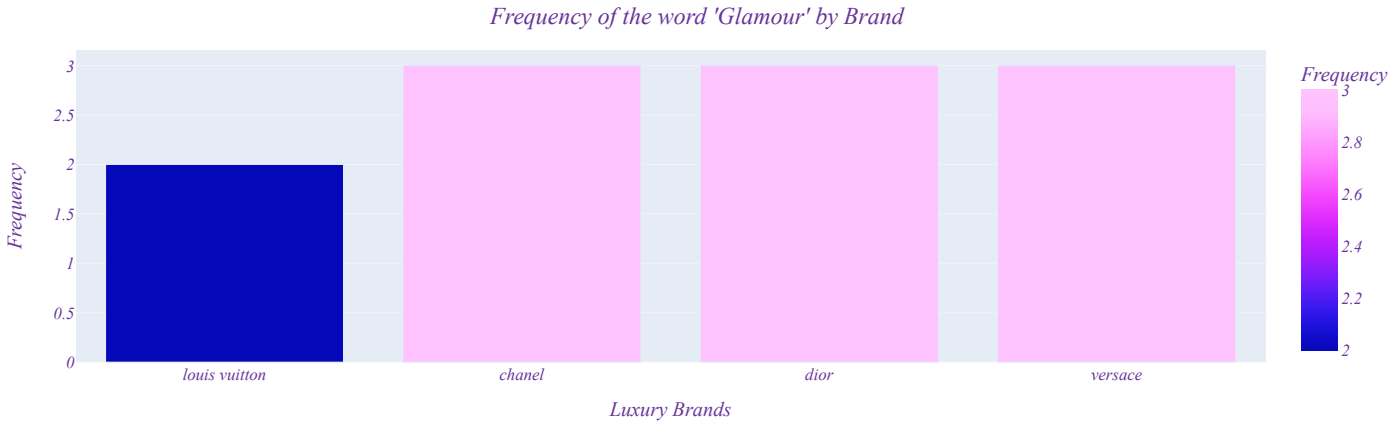
    wordFreq_barChart.update_layout(title_text=title_text,
                    title_x=0.5, # Center the title horizontally
                    title_y=0.94, # Adjust the vertical position if needed
                    font=dict(family='Old Standard TT, serif', size=14, color="RebeccaPurple"))

    wordFreq_barChart.show()
```

Word 'Glamour' Frequency by Luxury Brand

In [112…

```python
create_bar_chart_for_wordFreq(df_globalBrands, x='brand', y='glamour_frequency',
                              list_of_hover_data=['brand', 'Pos_Sentiment', 'Neg_Sentiment'], color='glamour_frequency',
                              labelsXY=['Luxury Brands', 'Frequency'],
                              color_continuous_scale='Plotly3',
                              title_text="Frequency of the word 'Glamour' by Brand")
```

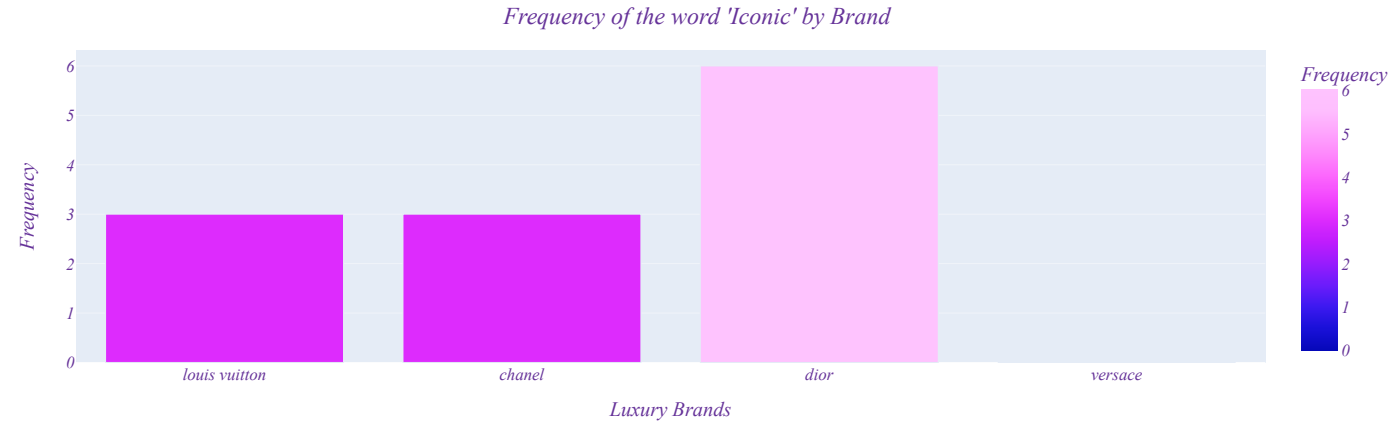


The frequency of the word "Glamour" for Chanel, Dior, and Versace is the same, suggesting that they all exuded and shared glamour in a similar fashion. This consistency in usage implies a common thread of sophistication and elegance across these iconic fashion houses, emphasizing their commitment to a luxurious and glamorous aesthetic that resonates with their respective audiences.

Word 'Iconic' Frequency by Luxury Brand

In [113…

```
create_bar_chart_for_wordFreq(df_globalBrands, x='brand', y='iconic_frequency',
                              list_of_hover_data=['brand', 'Pos_Sentiment', 'Neg_Sentiment'],
                              color='iconic_frequency',
                              labelsXY=['Luxury Brands', 'Frequency'],
                              color_continuous_scale='Plotly3',
                              title_text="Frequency of the word 'Iconic' by Brand")
```



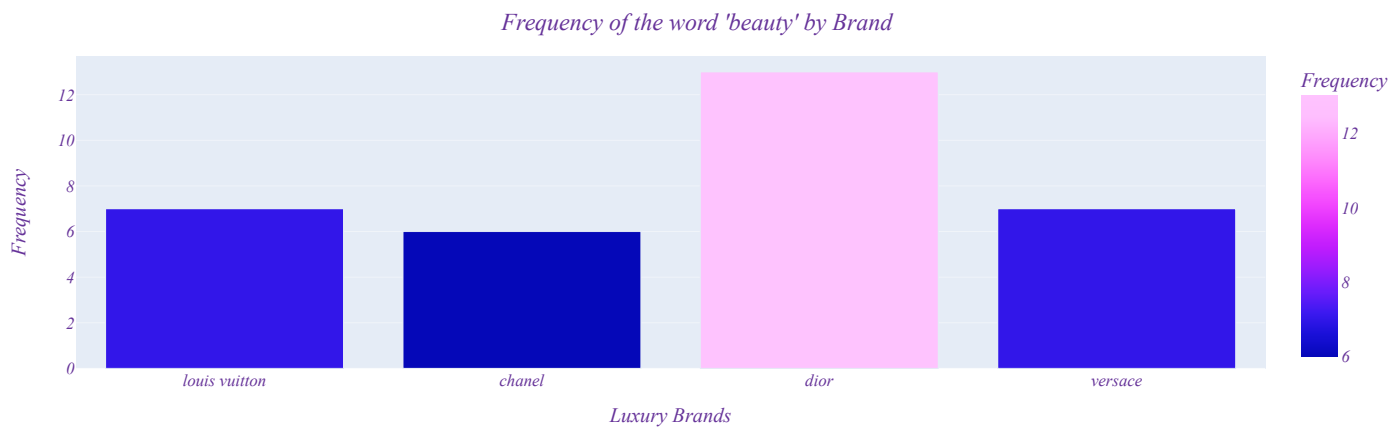
The significance of the word "Iconic" for Dior was indispensable, solidifying its position as the standout in terms of iconicity in this analysis. This consistent association suggests that Dior has successfully created a brand image and designs that are not only recognized but celebrated as iconic.

Note
The detailed examination of the bar charts reveals a notable observation concerning Versace in the current year: the absence of the term 'iconic' in any of the analyzed articles. This finding, though unexpected, offers valuable insights into the thematic and linguistic approaches employed by the various analyzed websites when referencing Versace.

Word 'Beauty' Frequency by Luxury Brand

In [114…

```
create_bar_chart_for_wordFreq(df_globalBrands, x='brand', y='beauty_frequency',
                              list_of_hover_data=['brand', 'Pos_Sentiment', 'Neg_Sentiment'],
                              color='beauty_frequency',
                              labelsXY=['Luxury Brands', 'Frequency'],
                              color_continuous_scale='Plotly3',
                              title_text="Frequency of the word 'beauty' by Brand")
```



A noteworthy aspect of this visualization is the inclusion of details regarding measurements of positive and negative sentiments when hovering over each bar.

Once again, Dior has earned acclaim for its image, characterized by sweetness and charm when discussed in the context of beauty. The term "beauty" emerged as indispensable in fashion magazines' references to Dior. This association underscores Dior's ability to consistently evoke a sense of beauty and allure.

5.3 Grouped Bar Chart

Summary : RELEVANCE OF LEGENDARY WORDS BY BRAND

Now that we have individually visualized the relevance represented by the frequencies of the words "Glamour," "Iconic," and "Beauty," we can summarize this information in a consolidated graphic. In a single visualization, we will arrange and compare the frequencies of these three keywords for each selected brand.

In [115…
```
# Define individual RGB color codes for each brand
color_LV = 'rgb(253, 197, 245)'
color_LV_border = 'rgb(255, 153, 200)'

color_Chanel = 'rgb(247, 174, 248)'
color_Chanel_border ='rgb(179, 136, 235)'

color_Dior = 'rgb(111, 255, 233)'
color_Dior_border= 'rgb(91, 192, 190)'

color_Versace = 'rgb(114, 221, 247)'
color_Versace_border = 'rgb(128, 147, 241)'

# Create a palette with the defined colors for each brand
palette_for_brands_fill = [color_LV, color_Chanel, color_Dior, color_Versace]
palette_for_brands_border = [color_LV_border, color_Chanel_border, color_Dior_border , color_Versace_border]
```

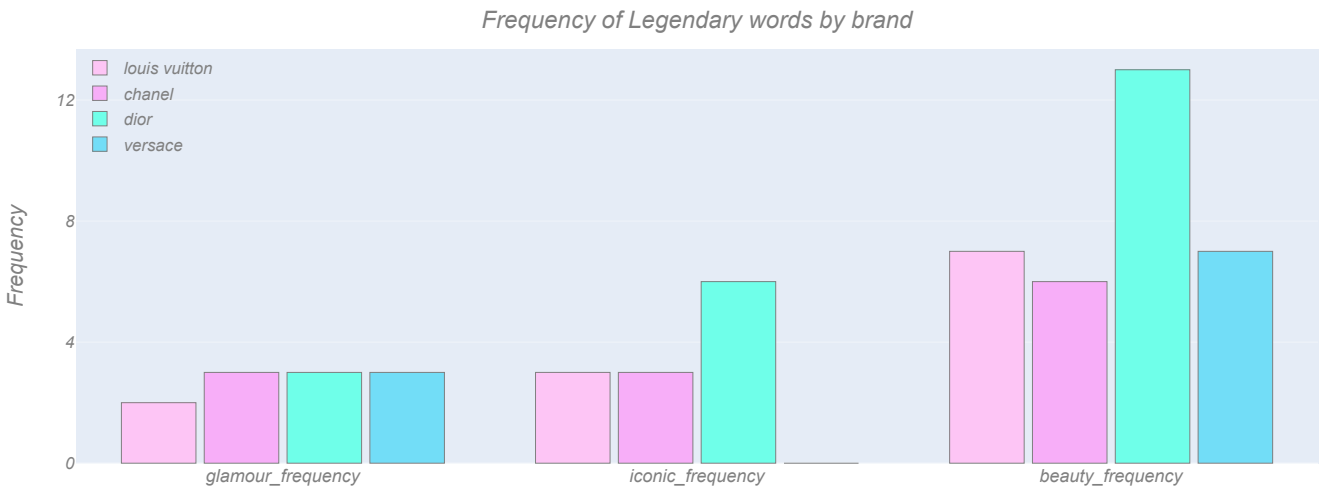
To enhance the organization and analysis of our data, we employ the `pd.melt()` function on the `df_globalBrands` DataFrame. This transformation results in the creation of a new DataFrame, named `df_wordsLegendary`. This DataFrame is structured with three key columns: `"brand,"` representing the brand names; `"variable,"` containing indicators such as `"glamour_frequency,"` `"iconic_frequency,"` and `"beauty_frequency,"` specifying the corresponding legendary word; and `"value,"` which stores the frequencies associated with each specified legendary word.
This reconfiguration streamlines the subsequent analytical process.

In [116…
```
df_wordsLegendary = pd.melt(df_globalBrands,id_vars=["brand"],
                            value_vars=["glamour_frequency", "iconic_frequency","beauty_frequency"])
# Example of the resulting format:
#    brand          variable  value
# 0  louis vuitton  glamour_frequency  2
# 1  chanel         glamour_frequency  3
# 2  dior           glamour_frequency  3
# ...
df_wordsLegendary.head()
```

Out[116]:

	brand	variable	value
0	louis vuitton	glamour_frequency	2
1	chanel	glamour_frequency	3
2	dior	glamour_frequency	3
3	versace	glamour_frequency	3
4	louis vuitton	iconic_frequency	3

In [117…
```
# Create a new figure for summarizing legendary word frequencies by brand
fig_summary_words = go.Figure()

# Iterate over unique brands to add traces for each brand
for brand, colorBrand_fill in zip(df_wordsLegendary['brand'].unique(),palette_for_brands_fill):

    brand_df = df_wordsLegendary[df_wordsLegendary['brand'] == brand]
    # Add a bar trace for each brand with specified color and marker attributes
    fig_summary_words.add_trace(go.Bar(
        name=brand,
        x=brand_df['variable'], # X-axis: Legendary word (e.g., Glamour, Iconic, Beauty)
        y=brand_df['value'],  # Y-axis: Frequency of legendary words
        width=0.18,  # Width of the bars
        marker_color=colorBrand_fill,  # Fill color of the bars
        marker_line_color='Gray',  # Border color of the bars
        marker_line_width=0.2))  # Border width of the bars

# Update the layout of the bar chart
fig_summary_words.update_layout(barmode='group',  # Bar mode: Grouped bars
                                xaxis_tickfont_size=15,  # Font size of X-axis ticks
                                yaxis_title='Frequency',  # Y-axis title
                                yaxis=dict(
                                    titlefont_size=18,  # Font size of Y-axis title
                                    tickfont_size=14,  # Font size of Y-axis ticks
                                    dtick=4, # Tick step size
                                ),
                                legend=dict(
                                    x=0,  # Legend x-coordinate
                                    y=1.0,  # Legend y-coordinate
                                    bgcolor='rgba(255, 255, 255, 0)',
                                    bordercolor='rgba(255, 255, 255, 0)'
                                ),  # Chart title
                                title='Frequency of Legendary words by brand',
                                # Font size of chart title
                                title_font=dict(size=20),
                                title_x=0.5,  # Center the title horizontally
                                title_y=0.87,  # Adjust the vertical position of the title
                                # Font style for the chart
                                font=dict(family="Arial, sans-serif", size=14, color="Gray"))

# Show the bar chart
fig_summary_words.show()
```



5.4 Stacked Bar Chart

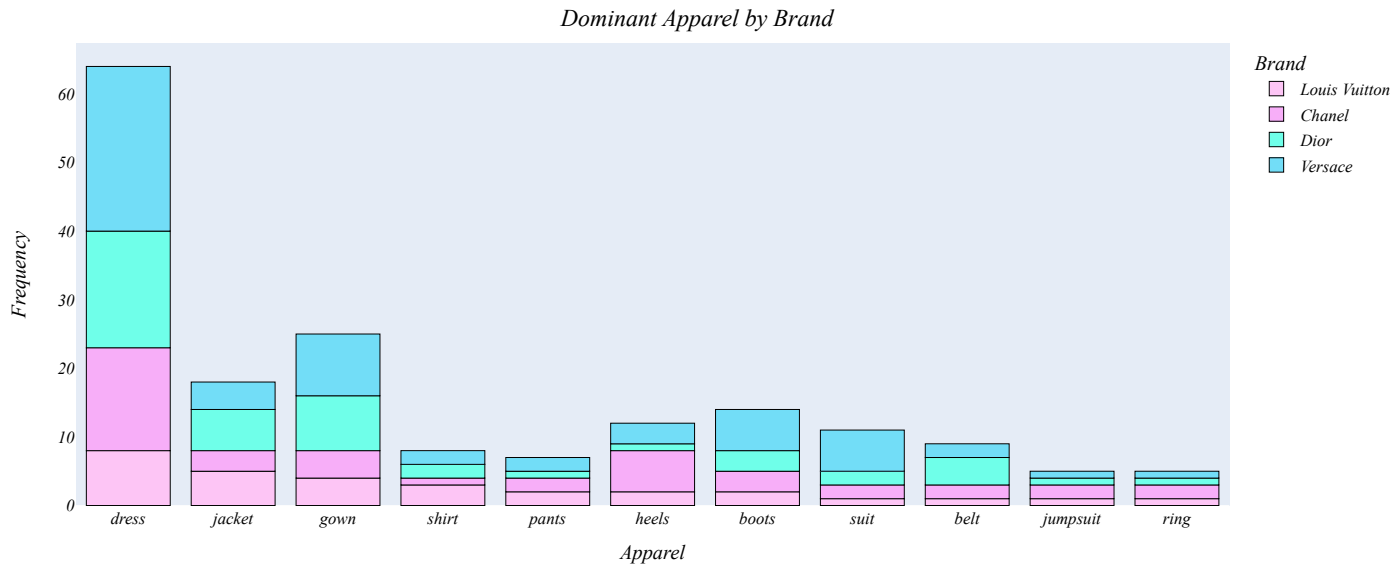
(6) In-Depth Exploration of Garment Relevance: *Exploring the relevance of garments in phase* `"(7)The most dominant clothing item in the World of Fashion according to Louis Vuitton, Chanel, Dior, and Versace:"` *in section* `"4. NLP Analysis of Brand Perception"` *revealed that dresses stood out as the dominant attire of the year, attracting the most attention in the selected articles. However, the question persists: were dresses the exclusive standout attire of the year? To delve deeper into this inquiry, we will generate a bar chart illustrating the relevance of various garments, sorted by their frequency across each selected brand. We will utilize the DataFrame df_frequency_fabric_by_brand, created in phase* `"(6) Garment (b)"` *of section* `"4. NLP Analysis of Brand Perception,"` *which stores data on identified garments, their associations with each brand, and their respective frequencies.*

Dominant Apparel by Brand

df_filtered_garments
In order to analyze the dominant garments common among the selected brands (Louis Vuitton, Chanel, Dior, and Versace), a filtering of information from the original DataFrame (df_frequency_apparel_by_brand) has been performed. This new DataFrame, named df_filtered_garments, will only include the frequencies of garments associated with the four brands, enabling a more focused and revealing visualization of shared dominant garments.

In [118…

```
#conventional way
# Identify the garments associated with all four brands
garments_in4Brands = [garment for garment in df_frequency_apparel_by_brand["Apparel"].unique() \
                      if df_frequency_apparel_by_brand["Apparel"].tolist().count(garment) == 4]
# Filter the DataFrame to include only the garments associated with all four brands
df_filtered_garments = df_frequency_apparel_by_brand[df_frequency_apparel_by_brand['Apparel'].isin(garments_in4Brands)]

# Create a bar chart to visualize the frequencies of dominant garments by brand
create_bar_chart(df_filtered_garments, x='Apparel', y='Frequency',
                 color='Brand',
                 color_discrete_sequence=[color_LV, color_Chanel, color_Dior, color_Versace],
                 title_text="Dominant Apparel by Brand", marker_line_color="Black",
                 marker_line_width=.07, size_text=14,showBackGrid=False)
```



Definitely, the predominant garments of this year according to Louis Vuitton, Chanel, Dior and Versace were dresses, and this can be attributed to their versatile and timeless appeal. Dresses are a versatile fashion choice suitable for various occasions, from casual to formal events, making them a staple in many wardrobes. The comfort and elegance offered by dresses likely contributed to their dominance in the fashion scene. However, gowns also had their moment in the spotlight, securing the second position. Gowns, with their luxurious and often elaborate designs, are typically associated with more formal and grand events. The prominence of gowns could be linked to special occasions, red carpet events, or a desire for elevated and opulent fashion choices.

Four consolidated bar charts will be created in a single diagram. This approach will enable a clear appreciation of the relative frequencies of each garment for each brand, providing an organized and meticulous perspective on the most prominent garment within the repertoire of each brand.

To achieve this, the functions `go.Figure()` *and* `add_trace()` *will be employed.* `go.Figure()` *takes care of setting up a blank canvas, onto which various traces can be added using* `add_trace()`*, where each trace represents an individual chart. This approach provides the necessary flexibility to combine different bar charts into a single diagram.*

In [119…

```python
def create_multiple_bar(dfName, x, y, name, palette_colors, xaxis_title=None, yaxis_title=None, title=None):
    """
    Create a grouped bar chart with multiple bars for each unique item in the specified column.

    Parameters:
    - dfName: DataFrame containing the data.
    - x: Column representing the x-axis data.
    - y: Column representing the y-axis data.
    - name: Column representing the color by which bars will be differentiated.
    - palette_colors: List of colors to be assigned to each unique item in the 'name' column.
    - xaxis_title: Title for the x-axis (optional).
    - yaxis_title: Title for the y-axis (optional).
    - title: Title for the entire chart (optional).

    Returns:
    - None: The function displays the generated chart.

    Example:
    ```python
    # Example DataFrame
    df = pd.DataFrame({
        'Category': ['A', 'A', 'B', 'B', 'C', 'C'],
        'Value': [10, 15, 20, 25, 12, 18],
        'Type': ['X', 'Y', 'X', 'Y', 'X', 'Y']
    })

    # Creating a grouped bar chart
    create_multiple_bar(df, 'Category', 'Value', 'Type', ['red', 'blue', 'green'],
                        xaxis_title='Categories', yaxis_title='Values', title='Comparison of Values by Type')
    ```

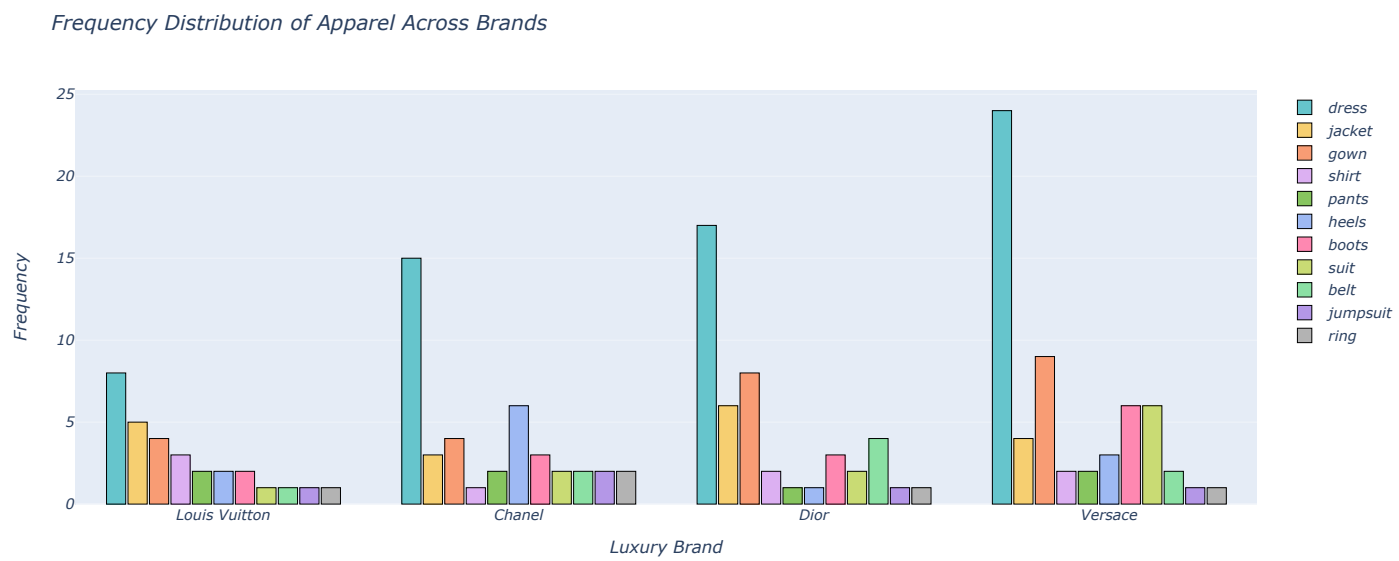
    Notes:
    - Ensure that the 'dfName' DataFrame includes the specified columns ('x', 'y', 'name').
    - 'palette_colors' should have a color for each unique item in the 'name' column.
    """

    fig_multiple = go.Figure()
    # Iterating over unique items and corresponding colors
    for item,color in zip(dfName[name].unique(),palette_colors):
        # Filtering the DataFrame for the current item
        brand_df = dfName[dfName[name] == item]
        # Adding a bar trace for the current item
        fig_multiple.add_trace(go.Bar(name=item,
                                x=brand_df[x],
                                y=brand_df[y],
                                marker_color=color,
                                marker_line_color='black',
                                marker_line_width=0.05,
                                width=0.065))
    # Updating the layout of the figure
    fig_multiple.update_layout(barmode='group',
                               xaxis_title=xaxis_title,
                               yaxis_title=yaxis_title,
                                title=title,
                                bargap=0.2)

    # Displaying the figure
    fig_multiple.show()
```

In [120…

```python
create_multiple_bar(df_filtered_garments, "Brand",
                    "Frequency","Apparel",
                    px.colors.qualitative.Pastel,xaxis_title="Luxury Brand",
                    yaxis_title="Frequency",title='Frequency Distribution of Apparel Across Brands')
```

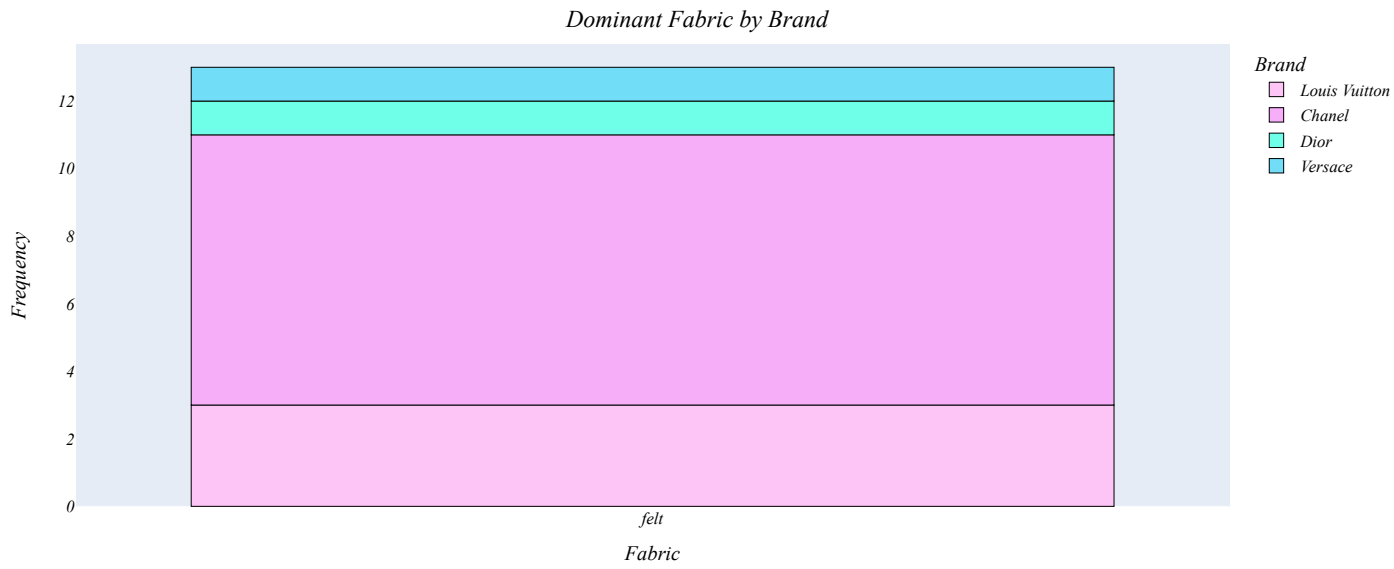


(7)In-Depth Exploration of Fabric Relevance: *In the exploration of fabric relevance during phase* `"(9)The most dominant fabric in the World of Fashion according to Louis Vuitton, Chanel, Dior, and Versace:"` *in section* `"4. NLP Analysis of Brand Perception,"` *it was revealed that leather emerged as the predominant material of the year, garnering the most attention in the selected articles. However, the question remains: was leather the exclusive standout fabric of the year? To further investigate, we will create a bar chart depicting the relevance of various fabrics, sorted by their frequency across each selected brand. We will leverage the DataFrame* `df_frequency_fabric_by_brand` *, established in phase* `"(8) Fabric (b)"` *of section* `"4. NLP Analysis of Brand Perception,"` *housing information on identified fabrics, their associations with each brand, and their respective frequencies.*

Dominant Fabric by Brand

df_filtered_fabrics In order to analyze the dominant fabrics common among the selected brands (Louis Vuitton, Chanel, Dior, and Versace), a filtering of information from the original DataFrame (df_frequency_fabric_by_brand) has been performed. This new DataFrame, named df_filtered_garments, will only include the frequencies of fabrics associated with the four brands, enabling a more focused and revealing visualization of shared dominant garments.

In [121…
```
# Identifying fabrics present in all four brands
fabrics_in4Brands = [fabric for fabric in df_frequency_fabric_by_brand["Fabric"].unique() \
                     if df_frequency_fabric_by_brand["Fabric"].tolist().count(fabric) == 4]

# Filtering the DataFrame to include only fabrics present in all four brands
df_filtered_fabrics = df_frequency_fabric_by_brand[df_frequency_fabric_by_brand["Fabric"].isin(fabrics_in4Brands)]

# Creating a bar chart for the dominant fabric by brand
create_bar_chart(df_filtered_fabrics,
                 x='Fabric',
                 y='Frequency',
                 color='Brand',
                 color_discrete_sequence=[color_LV, color_Chanel, color_Dior, color_Versace],
                 title_text="Dominant Fabric by Brand",
                 marker_line_color="Black",
                 marker_line_width=.07,
                 size_text=14,
                 showBackGrid=False)
```



It is noteworthy that according to Louis Vuitton, Chanel, Dior, and Versace, the fabric they shared as the main protagonist this year was "felt." This choice could be attributed to several reasons. Firstly, felt is a versatile and durable material, making it a popular option for various garments and accessories. Its soft and pleasant texture can impart a sense of luxury and comfort.

5.5 Bubble Scatter Plots

(8) Bubble Scatterplots of Garments and Fabrics: *During Stages* ***(6) In-Depth Exploration of Garment Relevance*** *and* ***(7) In-Depth Exploration of Fabric Relevance****, the focus was on visualizing prominent clothing items and fabrics throughout the year. However, these approaches exclusively centered on garments and fabrics shared among the selected luxury brands. To attain a comprehensive perspective of the relevance of all identified clothing items and fabrics throughout the articles analysis, two separate bubble scatterplots will be crafted. These graphs will depict garments and fabrics, along with their corresponding frequencies, integrating bubble size as an indicator of garment and fabric frequency.*

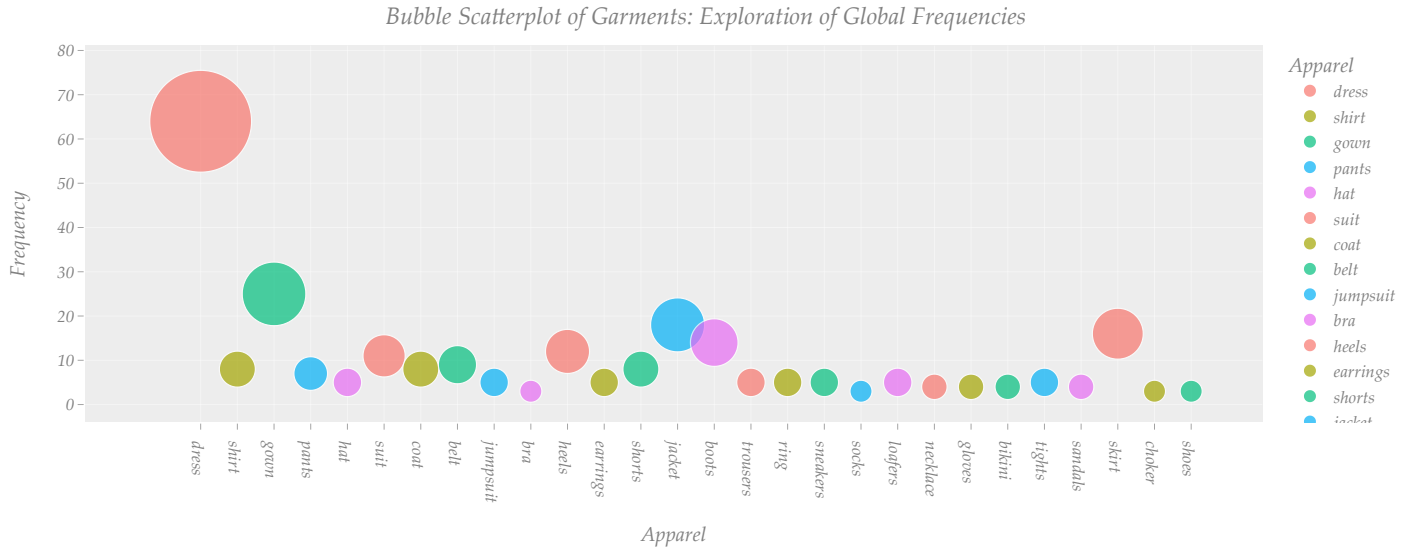
In [122…
```
# Get the available Plotly templates
import plotly.io as pio
pio.templates.keys()
```

Out[122]:
```
dict_keys(['ggplot2', 'seaborn', 'simple_white', 'plotly', 'plotly_white', 'plotly_dark', 'presentation', 'xgridoff', 'ygridoff', 'gridon', 'none'])
```

In [123…
```
def create_scatter(dfName,x,y,size,color,template,title):
    """
    Create and display a scatterplot using Plotly Express.

    Parameters:
    - dfName: DataFrame containing the data.
    - x: Column representing the x-axis data.
    - y: Column representing the y-axis data.
    - size: Column representing the size of bubbles in the scatterplot.
    - color: Column representing the color of bubbles in the scatterplot.
    - template: Plotly template to use for the chart.
    - title: Title for the scatterplot.

    Returns:
    - None: The function displays the generated scatterplot.

    Example:
    >>> import pandas as pd
    >>> df = pd.DataFrame({
    ...     'X': [1, 2, 3, 4],
    ...     'Y': [10, 11, 12, 13],
    ...     'Frequency': [2, 1, 3, 5],
    ...     'Brand': ['a', 'b', 'c', 'd']
    ... })
    >>> create_scatter(df, 'X', 'Y', 'Frequency', 'Brand', 'plotly', 'Example Scatterplot')
    """
    # Create the scatterplot according specific parameters
    figure_scatter = px.scatter(dfName,x=x, y=y, size=size, color=color,
                                size_max=60, template=template,title=title)
    # Customize the layout of the scatterplot
    figure_scatter.update_layout(
        title_x=0.50,  # Center the title horizontally
        title_y=0.87,  # Adjust the vertical position of the title
        font=dict(family="Palatino Linotype, Book Antiqua, Palatino, serif",
                  size=14, color="Gray")
    )
    # Display the scatterplot
    figure_scatter.show()
```

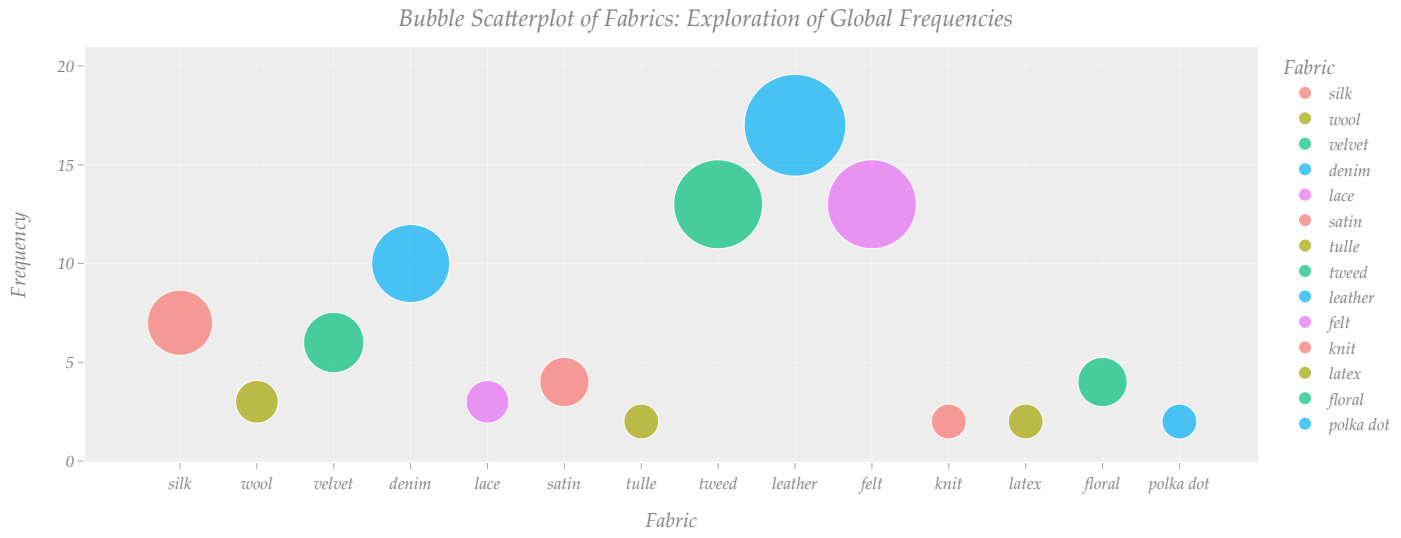
Bubble Scatterplot of Garments

In [124…
```
df_frequency_apparel_global_1 = moreThanN(df_frequency_apparel_global, 2)
create_scatter(df_frequency_apparel_global_1, x="Apparel", y="Frequency", size="Frequency",
               color="Apparel", template="ggplot2",
               title="Bubble Scatterplot of Garments: Exploration of Global Frequencies")
```



Bubble Scatterplot of Fabrics

In [125…
```
df_frequency_fabric_global_1 = moreThanN(df_frequency_fabric_global,1)
create_scatter(df_frequency_fabric_global_1, x="Fabric", y="Frequency",
               size="Frequency", color="Fabric", template="ggplot2",
               title="Bubble Scatterplot of Fabrics: Exploration of Global Frequencies")
```



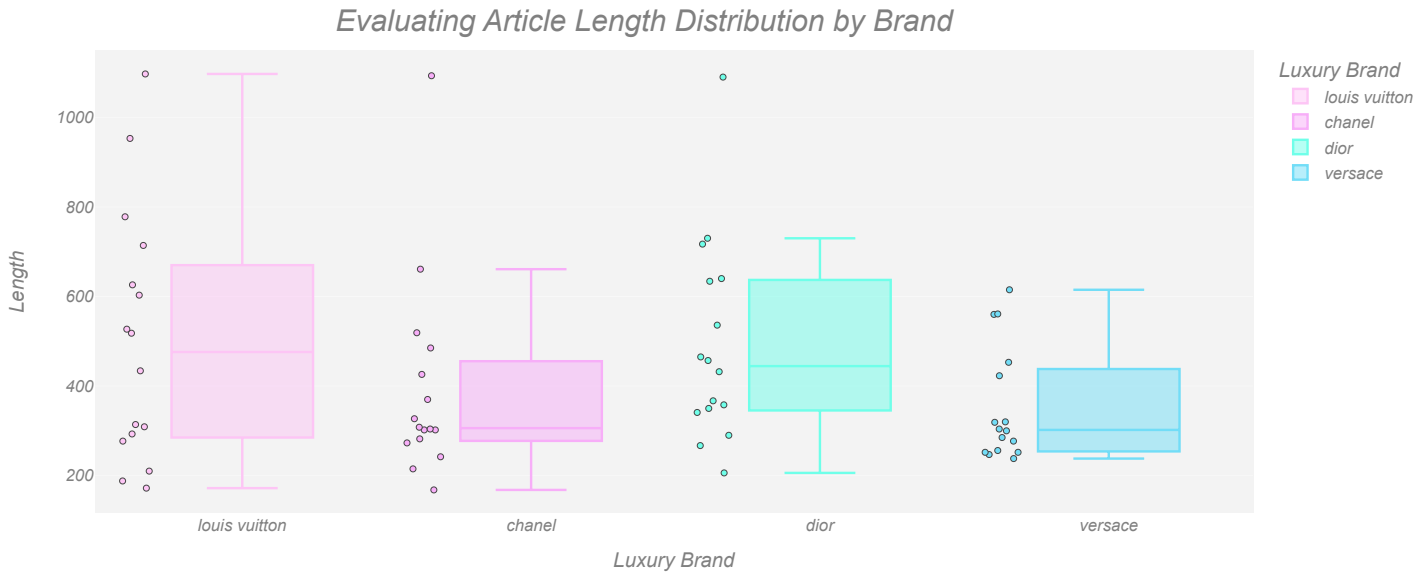
Previously, we observed that "felt" was the shared protagonist fabric among our selected luxury brands. However, throughout the analysis of all the articles, one fabric that was indispensable for many of these brands was "leather." Although it didn't claim the shared spotlight among the four brands, it proved to be essential for many of them.

5.6 Box Plots

(9) Evaluating Article Length Distribution: *In this phase, our focus shifts to exploring the distributions of article lengths within the selected luxury brands. To achieve this, we employ a boxplot, where the y-axis serves as the conduit for length data distribution. This approach is esencial for gaining deeper insights into the variations and balance in article lengths.*

In [126…
```
# Create a boxplot to analyze article length distribution by brand
figure_length_articles = px.box(
    df_articles,  # DataFrame containing the data
    x='brand',  # Column for the x-axis (Luxury Brand)
    y='Content length',  # Column for the y-axis (Length)
    color="brand",  # Column for color differentiation by brand
    boxmode="overlay",  # Overlay mode for multiple boxes per brand
    points="all",  # Display individual data points
    color_discrete_sequence=palette_for_brands_fill,  # Color palette for brands
    labels={'brand': "Luxury Brand", 'Content length': "Length"})

figure_length_articles.update_traces(marker_size=5, marker_line_width=0.1)

# Customize the layout of the boxplot
figure_length_articles.update_layout(
    plot_bgcolor='rgb(243, 243, 243)',
    title_text="Evaluating Article Length Distribution by Brand",
    title_font=dict(size=25),
    title_x=0.45,   # Center the horizant position of the title
    title_y=0.95,  # Adjust the vertical position of the title
    font=dict(family="Arial, sans-serif", size=14, color="Gray")
)
```

```
# Display the boxplot
figure_length_articles.show()
```



Through this graph, we observe that articles about the Louis Vuitton brand are significantly longer than those about Versace, Chanel, and Dior. This difference can be attributed to the longevity of Louis Vuitton, founded in 1854, making it a more established brand with a more extensive history. As one of the most recognized luxury brands, the diversity of Louis Vuitton's products and business lines, spanning from leather goods to fashion and travel items, may generate more content for media coverage.

5.7 Grouped Box Plots

***(10) Sentiment Analysis of Luxury Brands across Fashion magazines**: In this phase, we will visualize the distributions of sentiment measures surrounding the selected luxury brands, as portrayed by the four fashion magazines: Vogue, Elle, Glamour, and Harper's Bazaar. The selected luxury brands—Louis Vuitton, Chanel, Dior, and Versace—serve as focal points, each receiving a dedicated box plot. These graphics act as enlightening windows into the different sentiments (positive, neutral, and negative) associated with each brand in the selected magazines.*

The decision to collectively analyze negative, neutral, and positive sentiments is rooted in their interrelation, summing up to one and capturing the overall emotional responses. Conversely, the compound sentiment is addressed separately due to its distinct scale, allowing an evaluation of the general polarity without being influenced by the interplay of individual sentiments.

In [127…
```
# Define RGB color codes for Vogue
color_Vogue = 'rgb(171, 225, 136)'
color_Vogue_border = 'rgb(147, 159, 92)'

# Define RGB color codes for Elle
color_Elle = 'rgb(247, 239, 153)'
color_Elle_border = 'rgb(187, 148, 87)'

# Define RGB color codes for Glamour
color_Glamour = 'rgb(241, 187, 135)'
color_Glamour_border = 'rgb(226, 109, 90)'

# Define RGB color codes for Harper's Bazar
color_HarpersBazar = 'rgb(247, 142, 105)'
color_HarpersBazar_border = 'rgb(209, 96, 20)'

# Create palettes for fill and border colors
palette_for_websites_fill = [color_Vogue,color_Elle,color_Glamour,color_HarpersBazar]
palette_for_websites_border = [color_Vogue_border, color_Elle_border,color_Glamour_border,color_HarpersBazar_border]
```

The next steps involve leveraging the data within the `df_articles` DataFrame, encompassing all selected articles, their associated luxury brands, and corresponding websites. `melt()` function will be employed to generate four new DataFrames (`df_LV_sentiments` , `df_Chanel_sentiments` , `df_Dior_sentiments` , and `df_Versace_sentiments`), each linked to a specific brand. This process ensures that each resulting DataFrame is tailored to exclusively contain the sentiment measures (unpivoted) identified for the respective brand, along with the associated website.

In [128…
```
# melt_for_brand is a lambda function that takes a brand as an argument
#and creates a melted DataFrame for sentiment measures.The resulting DataFrame
#is unpivoted, containing sentiment measures for the specified brand and associated website.
melt_for_brand = lambda brand : pd.melt(
    df_articles[df_articles["brand"] == brand],
    id_vars="website",
    value_vars=["Neg_Sentiment", "Neu_Sentiment", "Pos_Sentiment"]
).rename(columns={"variable": "Sentiment_Category"})

df_LV_sentiments = melt_for_brand("Louis vuitton")
df_Chanel_sentiments = melt_for_brand("chanel")
df_Dior_sentiments = melt_for_brand("dior")
df_Versace_sentiments = melt_for_brand("versace")

df_LV_sentiments.head(5)
```

Out[128]:

	website	Sentiment_Category	value
0	vogue	Neg_Sentiment	0.050
1	vogue	Neg_Sentiment	0.015
2	vogue	Neg_Sentiment	0.032
3	vogue	Neg_Sentiment	0.026
4	glamour	Neg_Sentiment	0.078

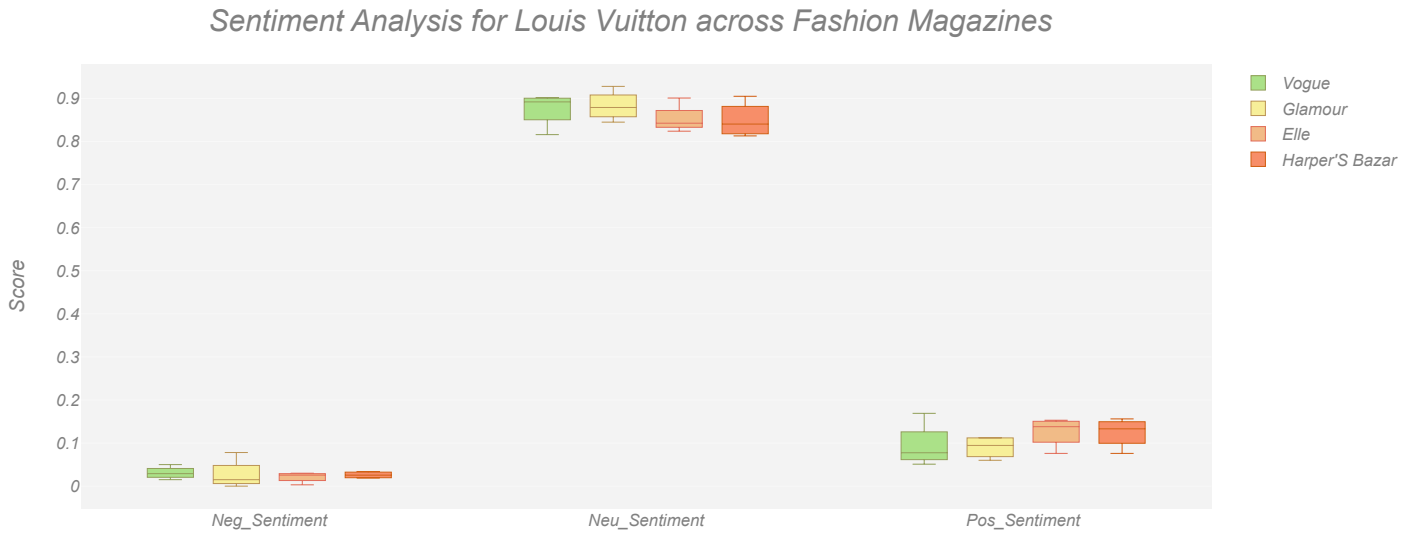
In [129…
```
def create_grouped_boxplot(dfName, x_col, y_col, hue_col, palette_for_fill,
                           palette_for_border, boxpoints=False,
                           title=None, height=550):
    """
    Generate a grouped boxplot based on the specified DataFrame.

    Parameters:
```

```
    - dfName (pd.DataFrame): DataFrame containing sentiment scores.
    - x_col (str): The column representing the x-axis values.
    - y_col (str): The column representing the y-axis values.
    - hue_col (str): The column to categorize and group the boxplots.
    - palette_for_fill (list): List of colors for filling the boxplots.
    - palette_for_border (list): List of colors for the borders of the boxplots.
    - title (str): Title for the plot.

    Returns:
    - None: Displays the generated boxplot.

    # Example:

    >>> create_grouped_boxplot_byWebsite(df_LV_sentiments, 'website', 'value', 'variable',
                                         palette_for_fill, palette_for_border, 'Sentiments measures by Brands')
    """
    # Initialize a Plotly Figure
    figure = go.Figure()

    # Iterate through unique categories (websites)
    for category, color_fill, color_border in zip(dfName[hue_col].unique(), palette_for_fill, palette_for_border):
        # Filter the DataFrame for the current category (website)
        category_df = dfName[dfName[hue_col] == category]
        # Add a trace for the boxplot
        figure.add_trace(go.Box(
            y=category_df[y_col],
            x=category_df[x_col],
            name=category.title(),
            marker_color=color_border,
            fillcolor=color_fill,
            marker_size=9,
            line_width=0.5,
            boxpoints=boxpoints
        ))

    # Update layout settings for the figure
    figure.update_layout(
        yaxis_title='Score',
        boxmode='group',
        height=height,
        yaxis=dict(
            autorange=True,
            showgrid=True,
            zeroline=True,
            dtick=0.1,
            gridcolor='rgb(255, 255, 255)',
            gridwidth=1,
            zerolinecolor='rgb(255, 255, 255)',
            zerolinewidth=1,
        ),
        plot_bgcolor='rgb(243, 243, 243)',
        title_text=title,
        title_font=dict(size=25),
        title_x=0.45,  # Center the title horizontally
        title_y=0.90,  # Adjust the vertical position of the title
        font=dict(family="Arial, sans-serif", size=14, color="Gray")
    )

    # Display the created figure
    figure.show()
```

Sentiment Analysis for Louis Vuitton across Fashion Magazines

In [130…

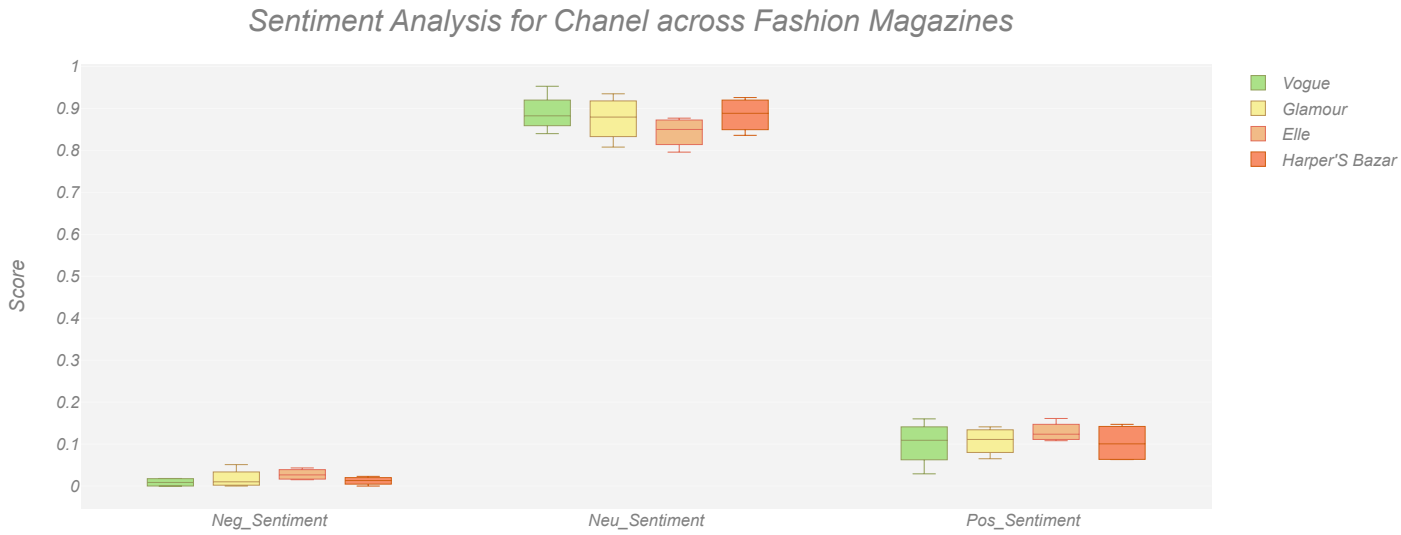
```
create_grouped_boxplot(df_LV_sentiments, x_col="Sentiment_Category", y_col="value",
                       hue_col="website", palette_for_fill=palette_for_websites_fill,
                       palette_for_border=palette_for_websites_border,
                       title="Sentiment Analysis for Louis Vuitton across Fashion Magazines")
```



Sentiment Analysis for Chanel across Fashion Magazines

In [131…

```
create_grouped_boxplot(df_Chanel_sentiments, x_col="Sentiment_Category", y_col="value",
                       hue_col="website", palette_for_fill=palette_for_websites_fill,
                       palette_for_border=palette_for_websites_border,
                       title="Sentiment Analysis for Chanel across Fashion Magazines")
```

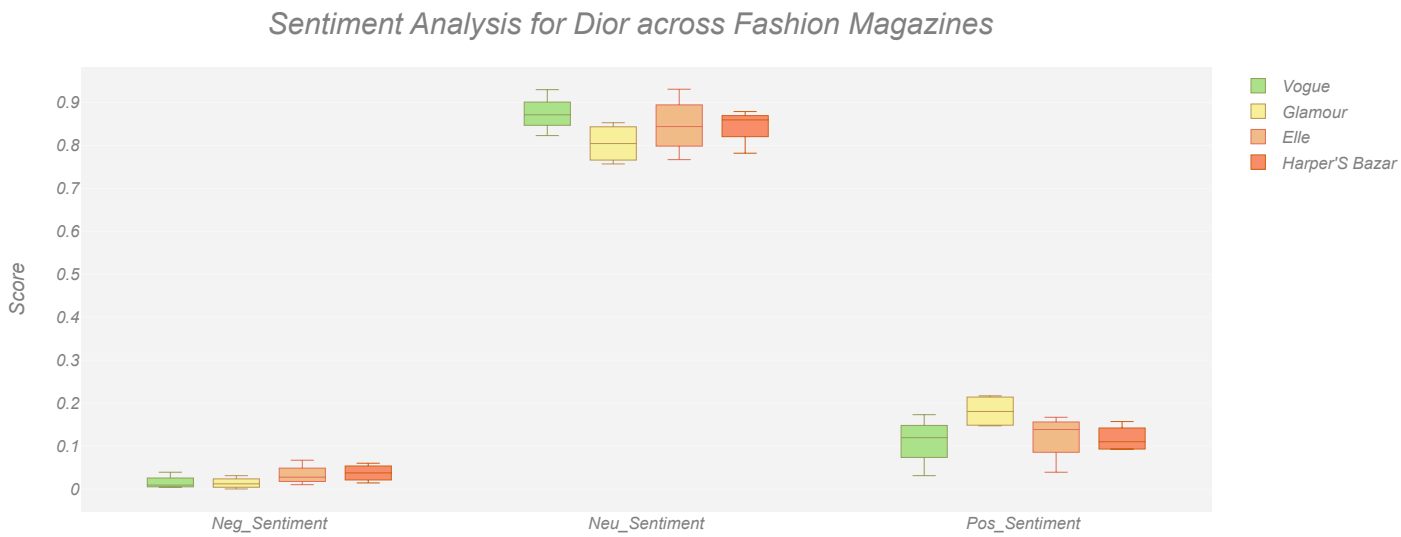


Based on the graph, we recognize that negative sentiments expressed in articles about Chanel were minimal. However, one magazine, Elle, stood out in this regard. The prevalence of sentiments identified as neutral is very close to 1, indicating that the majority of these fashion magazines wrote about Louis Vuitton in a neutral tone. Finally, the impact of positive sentiments recorded was not surprising, but one magazine slightly outshone the others, and that was Harper's Bazaar.

Sentiment Analysis for Dior across Fashion Magazines

In [132…

```
create_grouped_boxplot(df_Dior_sentiments, x_col="Sentiment_Category", y_col="value",
                       hue_col="website", palette_for_fill=palette_for_websites_fill,
palette_for_border=palette_for_websites_border,
                       title="Sentiment Analysis for Dior across Fashion Magazines")
```

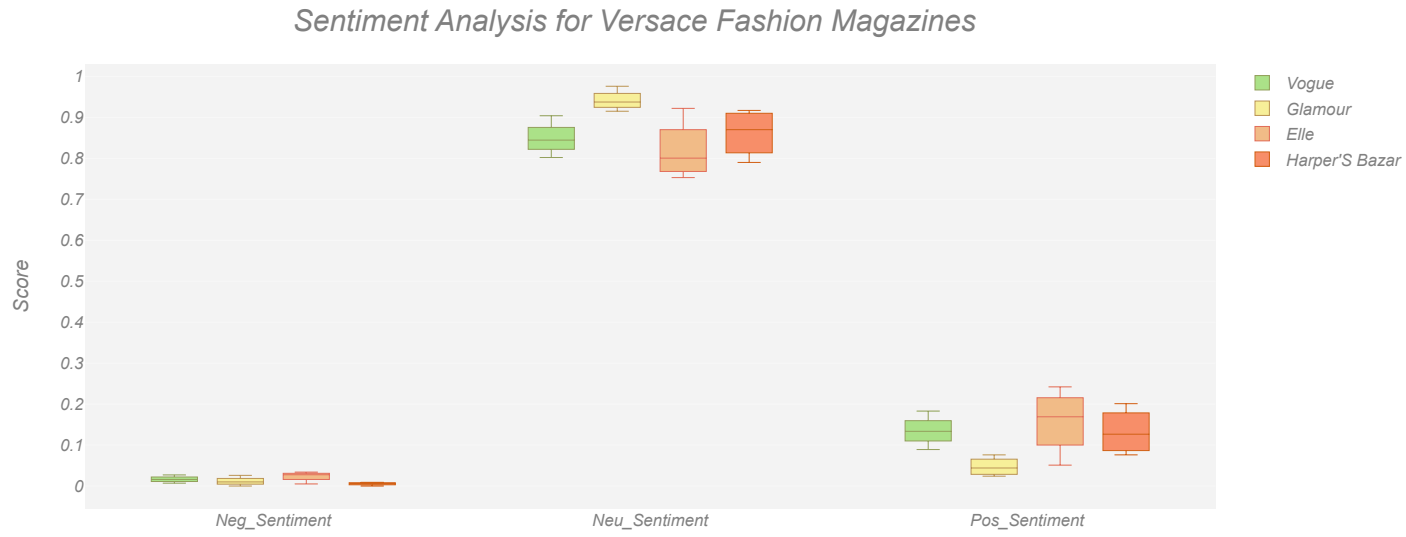


Indeed, from this graph, it's clear that Glamour strongly favors Dior, as evidenced by articles with a less neutral and more positive narrative towards the brand. Furthermore, concerning the measure of negative sentiments, Glamour exhibited the fewest negative sentiments in the narrative about Dior.

Sentiment Analysis for Versace across Fashion Magazines

In [133…

```
create_grouped_boxplot(df_Versace_sentiments, x_col="Sentiment_Category", y_col="value",
                       hue_col="website", palette_for_fill=palette_for_websites_fill,
                       palette_for_border=palette_for_websites_border,
                       title="Sentiment Analysis for Versace Fashion Magazines")
```

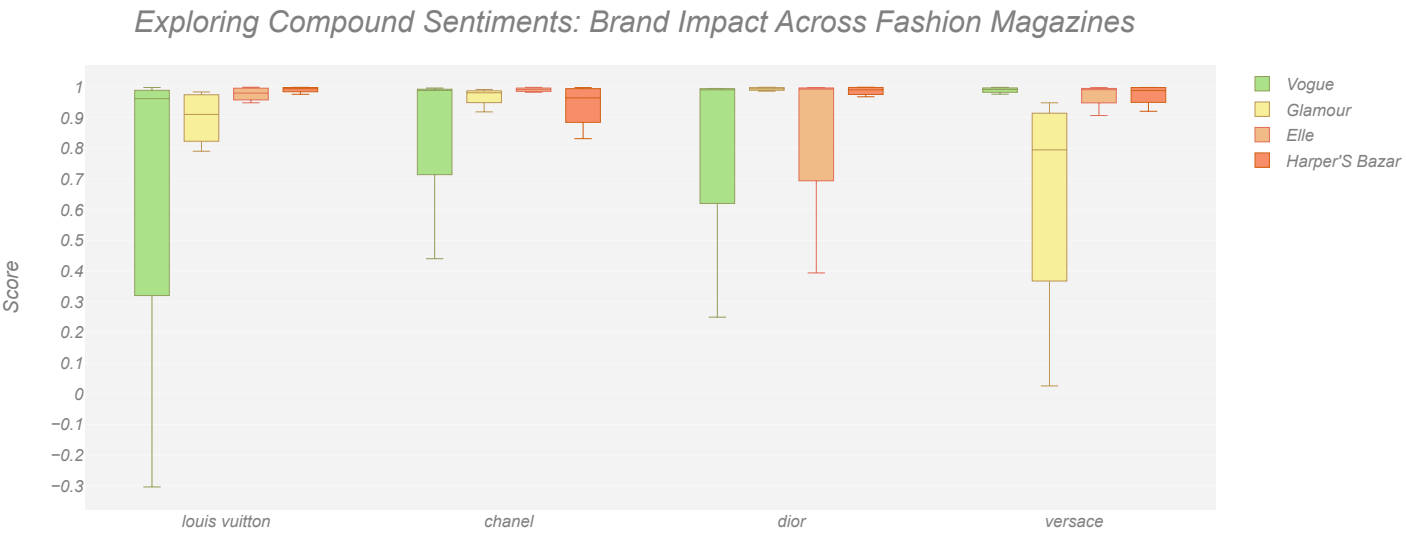


Through this graph, we observe a notable difference in the positive sentiments identified in the articles written by Glamour about Versace; they were definitely less positive than those of other fashion magazines. The reason behind this variation could be related to editorial preferences, writing approaches, or individual perspectives of Glamour towards the Versace brand. As we saw earlier, Glamour clearly demonstrates strong support for Dior, which could influence the tone of their articles about other fashion brands.

Compound Sentiments

In [134…

```
create_grouped_boxplot(df_articles, x_col="brand", y_col="Compound_Sentiment",
                       hue_col="website", palette_for_fill=palette_for_websites_fill,
                       palette_for_border=palette_for_websites_border,
                       title="Exploring Compound Sentiments: Brand Impact Across Fashion Magazines")
```



In general, all the magazines rated these luxury brands positively or neutrally, steering clear of expressing negative sentiments. This is understandable given that the world of fashion largely revolves around aesthetic appreciation and perception of the articles. Vogue exhibited a tendency to be more neutral in its narrative towards Louis Vuitton, Chanel, and Dior. However, it was exceptionally positive when discussing Versace, suggesting a clear favoritism towards this brand by Vogue. Glamour, on the other hand, revealed a strong appreciation for Dior, as the overall positive sentiments were notably higher for Dior compared to other luxury brands. Elle maintained a more neutral stance regarding Dior, while Harper's Bazaar showcased a clear affection for Louis Vuitton.

(11) Unveiling Luxury brands' emotional impact in selected articles: *In this phase, our aim is to identify the luxury brand that has generated the most positive, neutral, and negative perceptions in the selected articles. To achieve this, we employ a visual approach using boxplots. Each boxplot represents the frequency distribution of three sentiment categories:* ***Positive****,* ***Neutral****, and* ***Negative****, associated with a specific brand.*

This visual analysis provides valuable insights into the overall perception of each brand in terms of sentiments, enabling us to pinpoint the brand that has created the most favorable impression in the selected articles.

The decision to collectively analyze negative, neutral, and positive sentiments is rooted in their interrelation, summing up to one and capturing the overall emotional responses. Conversely, the compound sentiment is addressed separately due to its distinct scale, allowing an evaluation of the general polarity without being influenced by the interplay of individual sentiments.

The Pandas function `pd.melt()` is employed to reorganize the DataFrame `df_articles`, which contains information about selected articles, luxury brands, and their associated sentiments (Negative, Neutral, and Positive). The purpose of this operation is to reshape the sentiment columns into a more structured format. The `"brand"` column is retained as a unique identifier for each article, while the sentiment columns are "melted" or combined into two new columns: `"variable"` (indicating the sentiment type) and `"value"` (storing the corresponding numerical values). The resulting DataFrame, named `df_sentiments`, exhibits a more suitable structure for effectively exploring and analyzing sentiment categories associated with each brand in the selected articles.

In [135…

```
df_sentiments = pd.melt(df_articles,
                        id_vars=["brand"],
                        value_vars=["Neg_Sentiment", "Neu_Sentiment", "Pos_Sentiment"])
```

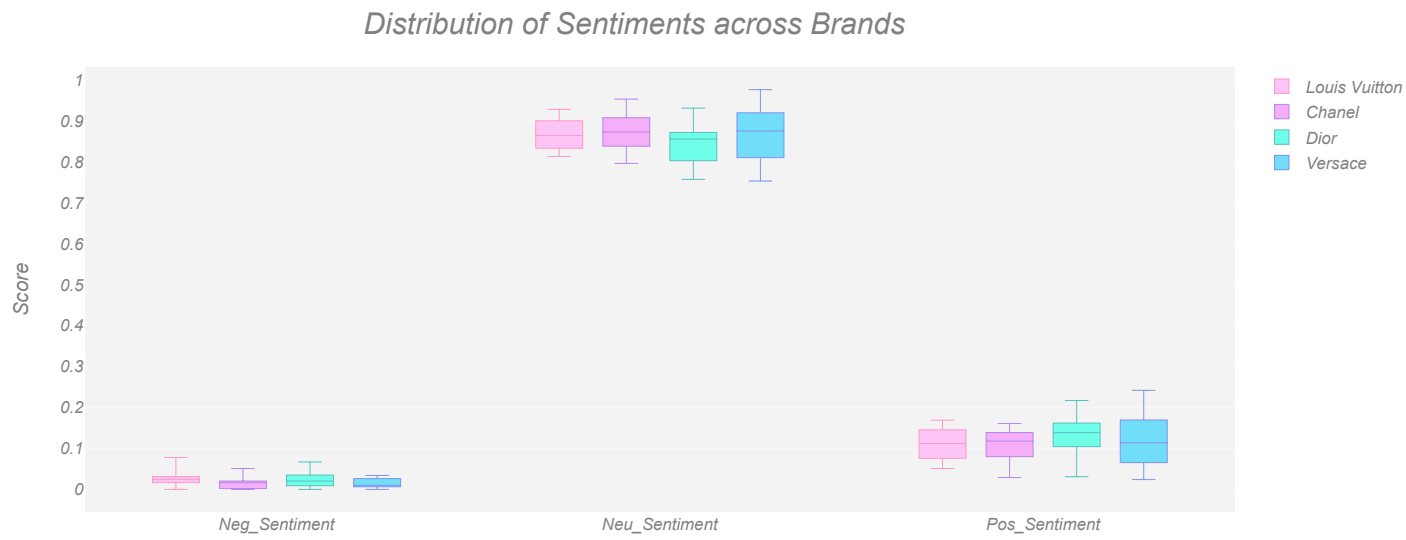
```
# Example of the resulting format:
#    brand               variable    value
# 0  louis vuitton  Neg_Sentiment  0.050
# 1  chanel         Neg_Sentiment  0.022
# 2  dior           Neg_Sentiment  0.010
# ...
df_sentiments.head()
```

Out[135]:

	brand	variable	value
0	louis vuitton	Neg_Sentiment	0.050
1	louis vuitton	Neg_Sentiment	0.015
2	louis vuitton	Neg_Sentiment	0.032
3	louis vuitton	Neg_Sentiment	0.026
4	chanel	Neg_Sentiment	0.018

In [136…

```
create_grouped_boxplot(df_sentiments, x_col="variable", y_col="value",
                       hue_col="brand",palette_for_fill=palette_for_brands_fill,
                       palette_for_border=palette_for_brands_border,
                       title="Distribution of Sentiments across Brands")
```

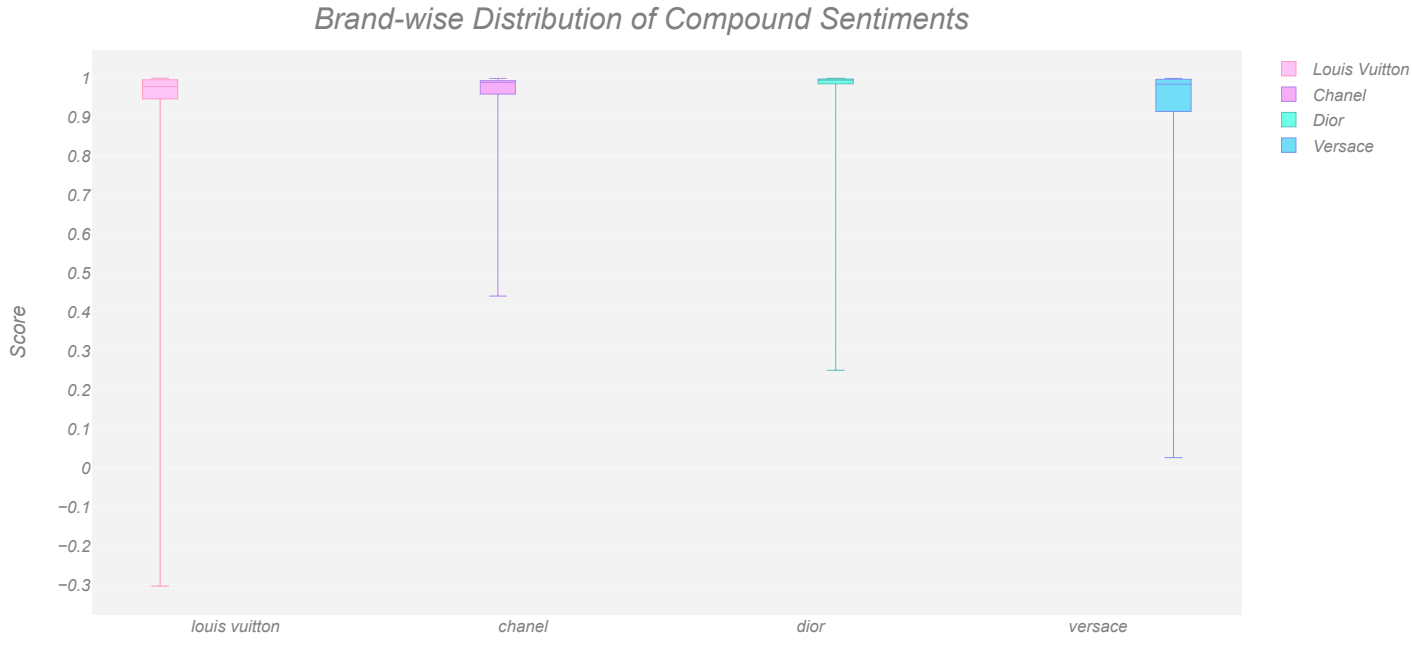


All luxury brands exhibited a similar distribution, remaining relatively neutral in their evaluations. This approach makes sense in the fashion world, where aesthetic appreciation, subjective interpretation, and positive perception are essential to the industry.

Compound Sentiments by brand

In [137…

```
create_grouped_boxplot(df_articles, x_col="brand", y_col="Compound_Sentiment",
                       hue_col="brand", palette_for_border=palette_for_brands_border,
                       palette_for_fill=palette_for_brands_fill,boxpoints=False,height=650,
                       title="Brand-wise Distribution of Compound Sentiments")
```



The brand Louis Vuitton exhibited some more pronounced negative evaluations compared to other luxury brands analyzed. This could be attributed to various reasons, such as the diversity of products and the variety of lines offered by Louis Vuitton, which may generate more polarized opinions. In contrast, Chanel received mainly neutral to positive evaluations. This could be due to the strong identity of the brand and its focus on classic elegance, which tends to resonate positively in assessments.

6. Summary

6.1 Conclusions

Dominant Adjectives

Through a comprehensive analysis of fashion articles, we identified the adjectives that predominantly characterize each luxury brand. Leveraging natural language processing techniques, we unveiled the qualities and attributes most associated with Louis Vuitton, Chanel, Dior, and Versace, providing a detailed insight into the linguistic perception of these brands.

Entities Involved

By identifying and analyzing entities related to each brand, we mapped a complex network of connections and influences. From iconic designers to collaborations and events, these entities provide a comprehensive understanding of the elements shaping the perception and position of each brand in the industry.

Dominant Garments in each brand

We examined the most mentioned garments in fashion articles associated with each brand. This analysis provides valuable insights into style preferences and the distinctive pieces that define the identity of Louis Vuitton, Chanel, Dior, and Versace in the fashion world.

Dominant Fabrics in each brand

The detailed exploration of fabrics mentioned in articles reveals the textures and materials intrinsically associated with each luxury brand. This analysis adds an additional layer of information about the sensory and aesthetic aspects characterizing these brands.

Dominant Garment and Fabric

Identifying common ground across all four luxury brands, our analysis unraveled a shared protagonist garment and fabric in the fashion discourse. By examining the narratives surrounding Louis Vuitton, Chanel, Dior, and Versace, we pinpointed the single most frequently highlighted clothing item and fabric across all brands. This shared protagonist not only signifies a unifying element in the fashion landscape but also underscores the collective impact of specific garments and fabrics that transcend individual brand identities.

Positive and Negative Perceptions in Fashion Magazine Narratives for Luxury Brands

By analyzing sentiments in fashion magazine narratives, we identified the emotional tones associated with each brand. These perceptions contribute to a holistic understanding of how luxury brands are perceived in the media space.

Luxury Brand Preferences According to Narrative in Fashion Magazines

We conclude by revealing brand preferences outlined by fashion magazine narratives. This analysis provides valuable insights into which brands emerge more strongly in the media landscape, influencing the collective perception in the fashion industry.

6.2 Final Data

Main DataFrames

Final DataFrame of All Selected Articles.

In [267…
```
df_articles.columns
```

Out[267]:
```
Index(['website', 'brand', 'title', 'content', 'Content length',
       'Filtered_title', 'Filtered_content', 'Lemmatized_title',
       'Lemmatized_content', 'Most_commom_Adjectives', 'entities_involved',
       'relevant_apparel', 'relevant_fabric', 'Neg_Sentiment', 'Neu_Sentiment',
       'Pos_Sentiment', 'Compound_Sentiment'],
      dtype='object')
```

Finalized DataFrames for each selected brand

Comprehensive data structures for Louis Vuitton, Chanel, Dior, and Versace articles, encompassing brand details, website sources, title and article corpus representations, filtered and lemmatized versions, product and entity mentions, as well as relevant details about apparel and fabric appearance.

In [268…
```
for dfName,name in zip(list_df_brands,name_df_brands):
    print(name, dfName.columns,"\n")
```

```
df_louisVuitton Index(['brand', 'website', 'title_corpus', 'article_corpus',
       'filtered_title_corpus', 'filtered_corpus', 'lemmatized_title_corpus',
       'lemmatized_corpus', 'products_appearance', 'entities_involved',
       'relevant_apparel', 'relevant_fabric'],
      dtype='object')

df_chanel Index(['brand', 'website', 'title_corpus', 'article_corpus',
       'filtered_title_corpus', 'filtered_corpus', 'lemmatized_title_corpus',
       'lemmatized_corpus', 'products_appearance', 'entities_involved',
       'relevant_apparel', 'relevant_fabric'],
      dtype='object')

df_dior Index(['brand', 'website', 'title_corpus', 'article_corpus',
       'filtered_title_corpus', 'filtered_corpus', 'lemmatized_title_corpus',
       'lemmatized_corpus', 'products_appearance', 'entities_involved',
       'relevant_apparel', 'relevant_fabric'],
      dtype='object')

df_versace Index(['brand', 'website', 'title_corpus', 'article_corpus',
       'filtered_title_corpus', 'filtered_corpus', 'lemmatized_title_corpus',
       'lemmatized_corpus', 'products_appearance', 'entities_involved',
       'relevant_apparel', 'relevant_fabric'],
      dtype='object')
```

Final Brand-Specific DataFrame

In [269…
```
df_globalBrands.columns
```

Out[269]:
```
Index(['brand', 'titles_corpus_final', 'articles_corpus_final',
       'filtered_titles_corpus', 'filtered_articles_corpus',
       'lemmatized_titles_corpus', 'lemmatized_articles_corpus',
       'products_appearance', 'entities_involved', 'glamour_frequency',
       'iconic_frequency', 'beauty_frequency', 'relevant_apparel',
       'relevant_fabric', 'Neg_Sentiment', 'Neu_Sentiment', 'Pos_Sentiment',
       'Compound_Sentiment'],
      dtype='object')
```

Secondary DataFrames

DataFrame containing identified adjectives and their frequencies for each brand

`df_adjectives_LV` `df_adjectives_Chanel` `df_adjectives_Dior` `df_adjectives_Versace`

In [270…
```
for df in [df_adjectives_LV,df_adjectives_Chanel,df_adjectives_Dior, df_adjectives_Versace]:
    print(df.columns)
```

```
Index(['Adjective', 'Frequency'], dtype='object')
Index(['Adjective', 'Frequency'], dtype='object')
Index(['Adjective', 'Frequency'], dtype='object')
Index(['Adjective', 'Frequency'], dtype='object')
```

DataFrames for Apparel and Fabric Frequency by Brand

`df_frequency_apparel_by_brand` `df_frequency_fabric_by_brand`

In [271…
```
df_frequency_apparel_by_brand.columns
```

Out[271]:
```
Index(['Brand', 'Apparel', 'Frequency'], dtype='object')
```

In [272…
```
df_frequency_fabric_by_brand.columns
```

Out[272]:
```
Index(['Brand', 'Fabric', 'Frequency'], dtype='object')
```

DataFrames for Sentiment Analysis by Brand

`df_LV_sentiments` `df_Chanel_sentiments` `df_Dior_sentiments` `df_Versace_sentiments`

In [273…
```
for df in [df_LV_sentiments,df_Chanel_sentiments,df_Dior_sentiments,df_Versace_sentiments]:
    print(df.columns, "\n")
```

```
Index(['website', 'Sentiment_Category', 'value'], dtype='object')

Index(['website', 'Sentiment_Category', 'value'], dtype='object')

Index(['website', 'Sentiment_Category', 'value'], dtype='object')

Index(['website', 'Sentiment_Category', 'value'], dtype='object')
```

7. References and Resources

7.1 References

1. Amed, I., & Berg, A. (2023, November 29). The state of fashion 2024. https://www.businessoffashion.com/reports/news-analysis/the-state-of-fashion-2024-report-bof-mckinsey
2. Editorial Team. (2023). Big data drives luxury brands growth beyond digital. Luxe Digital. https://luxe.digital/business/digital-luxury-reports/big-data-drives-luxury-brands-growth-beyond-digital/.
3. Chunmin Lang, Muzhen Li, Li Zhao. (2020). Understanding consumers' online fashion renting experiences: A text-mining approach, Sustainable Production and Consumption, 21, 132-144. https://doi.org/10.1016/j.spc.2019.12.003.
4. Heo, J. S., & Lee, E. J. (2019). Trend analysis of fashion brand evaluation using big data. Journal of the Korean Society of Costume, 69(6), 38-51. https://doi.org/10.7233/jksc.2019.69.6.038
5. Yonnet, P. (1985, p. 240). Juegos, modas y masas.
6. Bocock, D.R., & Bocock, R. (1993). Consumption (1st ed.). Routledge. https://doi.org/10.4324/9780203131091
7. Sassatelli, R. (2004). Consumo, cultura e società. https://books.google.com.pe/books?id=tt-rAAAACAAJ
8. Fontana, J. (1999). Introducción al estudio de la historia. https://books.google.com.pe/books?id=QL69AAAACAAJ
9. Pérez-Curiel, Concha & Luque, Sergio & Villena-Alarcón, Eduardo. (2017). Influencia de las revistas especializadas en el consumo de moda Estudio de casos: Smoda Elle y Harper's Bazaar. 39. https://idus.us.es/handle/11441/68929
10. Britannica, T. Editors of Encyclopaedia (2023, November 9). Vogue. Encyclopedia Britannica. https://www.britannica.com/topic/Vogue-American-magazine
11. Britannica, T. Editors of Encyclopaedia (2023, October 8). Ll. Encyclopedia Britannica. https://www.britannica.com/topic/Elle-French-fashion-magazine
12. WorthPoint. (n.d.). Glamour magazine. WorthPoint Dictionary. Retrieved December 1, 2023, from https://www.worthpoint.com/dictionary/p/books-paper-magazines/titles/glamour-magazine
13. Stephen Mooallem. (2021). 150 Years of Harper's Bazaar. https://www.harpersbazaar.com/culture/features/a18658/history-of-harpers-bazaar/
14. Smith, M. (2023, April 24). LVMH becomes the first European company to surpass $500 billion in value. CNBC. https://www.cnbc.com/2023/04/24/lvmh-becomes-the-first-european-company-surpass-500-billion-in-value.html.
15. Vogue México. (2021, November 19). Coco Chanel: biografía, quién es y frases. Vogue México. https://www.vogue.mx/estilo-de-vida/articulo/coco-chanel-biografia-quien-es-frases
16. Vogue México. (2021, January 21). Christian Dior: biografía del diseñador que cambió la moda. https://www.vogue.mx/estilo-de-vida/articulo/christian-dior-biografia
17. Vogue México. (2020, 2 de diciembre). Gianni Versace: biografía, cómo murió y legado del diseñador de modas. Vogue México. https://www.vogue.mx/estilo-de-vida/articulo/gianni-versace-biografia-como-murio-y-legado-del-disenador-de-modas
18. Lakoff, R. (1973). Language and Woman's Place. Language in Society, 2(1), 45–80. http://www.jstor.org/stable/4166707
19. Woolf, V. (1928). Orlando. Rosetta Books(pp. 111-112).
20. Professor Mary Lynn Stewart, review of Glamour. A History, (review no. 713) https://reviews.history.ac.uk/review/713 Date accessed: 20 December, 2023
21. Hahn, S., & Yang, S.H. (2006). A Study on Glamour Look Expressed in Fashion (Part I). Journal of the Korean Society of Clothing and Textiles, 30, 1288-1300.
22. Merriam-Webster. "Iconic." Merriam-Webster.com Dictionary, Merriam-Webster, https://www.merriam-webster.com/dictionary/iconic. Accessed 18 Dec. 2023.
23. CNN Style. (2021, January 9). Coco Chanel: How the fashion designer's legacy lives on. CNN. https://www.cnn.com/style/article/coco-chanel-fashion-50-years
24. Trotter, K. (2020, April 20). 10 most important fashion moments in history. Vogue Arabia. https://en.vogue.me/fashion/10-historical-fashion-moments/
25. Ines, A. (2020). Beauty and fashion: A powerful alliance. VOCAST. https://vocast.com/beauty-and-fashion-a-powerful-alliance

26. *L'OFFICIEL USA. (2021, May 4). The history and rise to fame of Audrey Hepburn's Givenchy little black dress. L'OFFICIEL. https://www.lofficielusa.com/fashion/audrey-hepburn-givenchy-little-black-dress-history*

27. *Marc Jacobs. About Marc. Marc Jacobs. https://www.marcjacobs.com/default/about-marc/*

28. *Marc Jacobs. LVMH. https://www.lvmh.com/houses/fashion-leather-goods/marc-jacobs/*

29. *Andjelic, A. How Marc Jacobs Shaped Louis Vuitton's Future. Highsnobiety. https://www.highsnobiety.com/p/marc-jacobs-louis-vuitton-history/*

30. *Gwyneth Paltrow: Official Website https://gwynethpaltrow.com/*

31. *The house of Alexander McQueen. Kering, https://www.kering.com/en/houses/couture-and-leather-goods/alexander-mcqueen/history/*

32. *APA: Levine, N. (2023, March 9). Why Miley Cyrus is the ultimate 21st-century pop star. BBC Culture. https://www.bbc.com/culture/article/20230309-why-miley-cyrus-is-the-ultimate-21st-century-pop-star*

7.2 Resources

NLP Analysis

1. *Python Software Foundation. (2021). collections — Container datatypes. Python 3.10.0 Documentation. https://docs.python.org/3/library/collections.html#collections.Counter*
2. *Pandas. pandas.melt.*
   https://pandas.pydata.org/docs/reference/api/pandas.melt.html

Visual Interpretation of Results

1. *Color Palettes. Coolors. Trending palettes.*
   https://coolors.co/palettes/trending

2. *Plotly. (2021). Built-in continuous color scales. Plotly.*
   https://plotly.com/python/builtin-colorscales/

3. *Plotly. (2021). How to make Bar Charts in Python with Plotly.*
   https://plotly.com/python/bar-charts/

4. *Plotly. (2023). How to make bubble charts in Python with Plotly.*
   https://plotly.com/python/bubble-charts/

5. *Plotly. (2021). How to make box plots in Python with Plotly.*
   https://plotly.com/python/box-plots/