H&M Personalized Fashion Recommendations

In this Exploratory Data Analysis Notebook, we will examine the data, check for missing information, understand the data distribution, and explore the relationships between data in various files.

We will also delve into the image data to understand how images are indexed in the CSV files and determine if there are articles in the dataset without images. Additionally, we will explore additional information related to the images, such as image width and height.

We will also investigate a very simple baseline model.

In [1]:
```
import numpy as np
import pandas as pd
import os
from tqdm import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from datetime import datetime
from PIL import Image
```

In [6]:
```
articles_df = pd.read_csv("/input/h-and-m-personalized-fashion-recommendations/articles.csv")
customers_df = pd.read_csv("/input/h-and-m-personalized-fashion-recommendations/customers.csv")
```

In [7]:
```
transactions_train_df = pd.read_csv("/input/h-and-m-personalized-fashion-recommendations/transactions_train.csv")
```

In [8]:
```
articles_df.head()
```

Out[8]:

	article_id	product_code	prod_name	product_type_no	product_type_name	product_group_name	graphical_appearance_no	graphical_appe
0	108775015	108775	Strap top	253	Vest top	Garment Upper body	1010016	
1	108775044	108775	Strap top	253	Vest top	Garment Upper body	1010016	
2	108775051	108775	Strap top (1)	253	Vest top	Garment Upper body	1010017	
3	110065001	110065	OP T-shirt (Idro)	306	Bra	Underwear	1010016	
4	110065002	110065	OP T-shirt (Idro)	306	Bra	Underwear	1010016	

5 rows × 25 columns

In [9]:
```
customers_df.head()
```

Out[9]:

	customer_id	FN	Active	club_member_status	fashion_news_frequency	age	
0	00000dbacae5abe5e23885899a1fa44253a17956c6d1c3...	NaN	NaN	ACTIVE	NONE	49.0	52043ee2162cf5aa7ee7
1	0000423b00ade91418cceaf3b26c6af3dd342b51fd051e...	NaN	NaN	ACTIVE	NONE	25.0	2973abc54daa8a5f8cc
2	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	NaN	NaN	ACTIVE	NONE	24.0	64f17e6a330a85798e49
3	00005ca1c9ed5f5146b52ac8639a40ca9d57aeff4d1bd2...	NaN	NaN	ACTIVE	NONE	54.0	5d36574f52495e81f019
4	00006413d8573cd20ed7128e53b7b13819fe5cfc2d801f...	1.0	1.0	ACTIVE	Regularly	52.0	25fa5ddee9aac01b35208

In [11]:
```
transactions_train_df.head()
```

Out[11]:

	t_dat	customer_id	article_id	price	sales_channel_id
0	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	663713001	0.050831	2
1	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	541518023	0.030492	2
2	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	505221004	0.015237	2
3	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687003	0.016932	2
4	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687004	0.016932	2

There are three main tables in the dataset:

Articles: This table contains information about each article, such as product code, name, product group code, and more.

Customers: In this table, you can find information about each customer, including details like fidelity card membership, age, and postal code.

Transactions (train): The 'Transactions' table contains data related to transactions and includes two important foreign keys – customer_id and article_id, which link to the 'Customers' and 'Articles' tables. Additionally, the 'Transactions' table includes the field `sales_channel_id.

The 'Transaction train' data includes entries for the date of each transaction, the corresponding customer ID, article ID, transaction price, and a sales channel ID.

In [12]:
```
def missing_data(data):
    total = data.isnull().sum().sort_values(ascending = False)
    percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(ascending = False)
    return pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
```

In [13]:
```
def unique_values(data):
    total = data.count()
    tt = pd.DataFrame(total)
    tt.columns = ['Total']
    uniques = []
    for col in data.columns:
        unique = data[col].nunique()
        uniques.append(unique)
    tt['Uniques'] = uniques
    return tt
```

In [14]:
```
articles_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105542 entries, 0 to 105541
Data columns (total 25 columns):
 #   Column                        Non-Null Count   Dtype 
---  ------                        --------------   ----- 
 0   article_id                    105542 non-null  int64 
 1   product_code                  105542 non-null  int64 
 2   prod_name                     105542 non-null  object
 3   product_type_no               105542 non-null  int64 
 4   product_type_name             105542 non-null  object
 5   product_group_name            105542 non-null  object
 6   graphical_appearance_no       105542 non-null  int64 
 7   graphical_appearance_name     105542 non-null  object
 8   colour_group_code             105542 non-null  int64 
 9   colour_group_name             105542 non-null  object
 10  perceived_colour_value_id     105542 non-null  int64 
 11  perceived_colour_value_name   105542 non-null  object
 12  perceived_colour_master_id    105542 non-null  int64 
 13  perceived_colour_master_name  105542 non-null  object
 14  department_no                 105542 non-null  int64 
 15  department_name               105542 non-null  object
 16  index_code                    105542 non-null  object
 17  index_name                    105542 non-null  object
 18  index_group_no                105542 non-null  int64 
 19  index_group_name              105542 non-null  object
 20  section_no                    105542 non-null  int64 
 21  section_name                  105542 non-null  object
 22  garment_group_no              105542 non-null  int64 
 23  garment_group_name            105542 non-null  object
 24  detail_desc                   105126 non-null  object
dtypes: int64(11), object(14)
memory usage: 20.1+ MB
```

In [15]:
```
missing_data(articles_df)
```

Out[15]:

	Total	Percent
detail_desc	416	0.394156
perceived_colour_master_name	0	0.000000
garment_group_name	0	0.000000
garment_group_no	0	0.000000
section_name	0	0.000000
section_no	0	0.000000
index_group_name	0	0.000000
index_group_no	0	0.000000
index_name	0	0.000000
index_code	0	0.000000
department_name	0	0.000000
department_no	0	0.000000
article_id	0	0.000000
product_code	0	0.000000
perceived_colour_value_name	0	0.000000
perceived_colour_value_id	0	0.000000
colour_group_name	0	0.000000
colour_group_code	0	0.000000
graphical_appearance_name	0	0.000000
graphical_appearance_no	0	0.000000
product_group_name	0	0.000000
product_type_name	0	0.000000
product_type_no	0	0.000000
prod_name	0	0.000000
perceived_colour_master_id	0	0.000000

In the article data, there is only missing data for the detailed article description, which accounts for 0.4% of the total data.

In [16]:
```
customers_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1371980 entries, 0 to 1371979
Data columns (total 7 columns):
 #   Column                  Non-Null Count    Dtype  
---  ------                  --------------    -----  
 0   customer_id             1371980 non-null  object 
 1   FN                      476930 non-null   float64
 2   Active                  464404 non-null   float64
 3   club_member_status      1365918 non-null  object 
 4   fashion_news_frequency  1355971 non-null  object 
 5   age                     1356119 non-null  float64
 6   postal_code             1371980 non-null  object 
dtypes: float64(3), object(4)
memory usage: 73.3+ MB
```

In [17]:
```
missing_data(customers_df)
```

Out[17]:

	Total	Percent
Active	907576	66.150819
FN	895050	65.237831
fashion_news_frequency	16009	1.166854
age	15861	1.156066
club_member_status	6062	0.441843
customer_id	0	0.000000
postal_code	0	0.000000

Only the 'customer_id' and 'postal code' fields are completely filled. 'Age' and 'fashion news frequency' have approximately 1% missing data, while 'FN' and 'Active' have 65% and 66% missing data, respectively.

In [19]:
```
transactions_train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31788324 entries, 0 to 31788323
Data columns (total 5 columns):
 #   Column            Dtype
---  ------            -----
 0   t_dat             object
 1   customer_id       object
 2   article_id        int64
 3   price             float64
 4   sales_channel_id  int64
dtypes: float64(1), int64(2), object(2)
memory usage: 1.2+ GB
```

In [20]:
```
missing_data(transactions_train_df)
```

Out[20]:

	Total	Percent
t_dat	0	0.0
customer_id	0	0.0
article_id	0	0.0
price	0	0.0
sales_channel_id	0	0.0

No missing data from the transactions train data source.

Unique values

In [21]:
```
unique_values(articles_df)
```

Out[21]:

	Total	Uniques
article_id	105542	105542
product_code	105542	47224
prod_name	105542	45875
product_type_no	105542	132
product_type_name	105542	131
product_group_name	105542	19
graphical_appearance_no	105542	30
graphical_appearance_name	105542	30
colour_group_code	105542	50
colour_group_name	105542	50
perceived_colour_value_id	105542	8
perceived_colour_value_name	105542	8
perceived_colour_master_id	105542	20
perceived_colour_master_name	105542	20
department_no	105542	299
department_name	105542	250
index_code	105542	10
index_name	105542	10
index_group_no	105542	5
index_group_name	105542	5
section_no	105542	57
section_name	105542	56
garment_group_no	105542	21
garment_group_name	105542	21
detail_desc	105126	43404

We've noticed discrepancies in the number of unique values between related features. For instance:

'product_type_no' and 'product_type_name' 'department_no' and 'department_name' 'section_no' and 'section_name'

These pairs have different numbers of unique values, indicating potential issues with categories having the same name.

On the other hand:

'index_code' and 'index_name' 'garment_group_no' and 'garment_group_name'

In [22]:
```
unique_values(customers_df)
```

Out[22]:

	Total	Uniques
customer_id	1371980	1371980
FN	476930	1
Active	464404	1
club_member_status	1365918	3
fashion_news_frequency	1355971	4
age	1356119	84
postal_code	1371980	352899

In [23]:
```
unique_values(transactions_train_df)
```

Out[23]:

	Total	Uniques
t_dat	31788324	734
customer_id	31788324	1362281
article_id	31788324	104547
price	31788324	9857
sales_channel_id	31788324	2

We've observed that not all customers in the customer data have corresponding transactions in the transaction train data. Additionally, not all articles are represented in the transaction data.

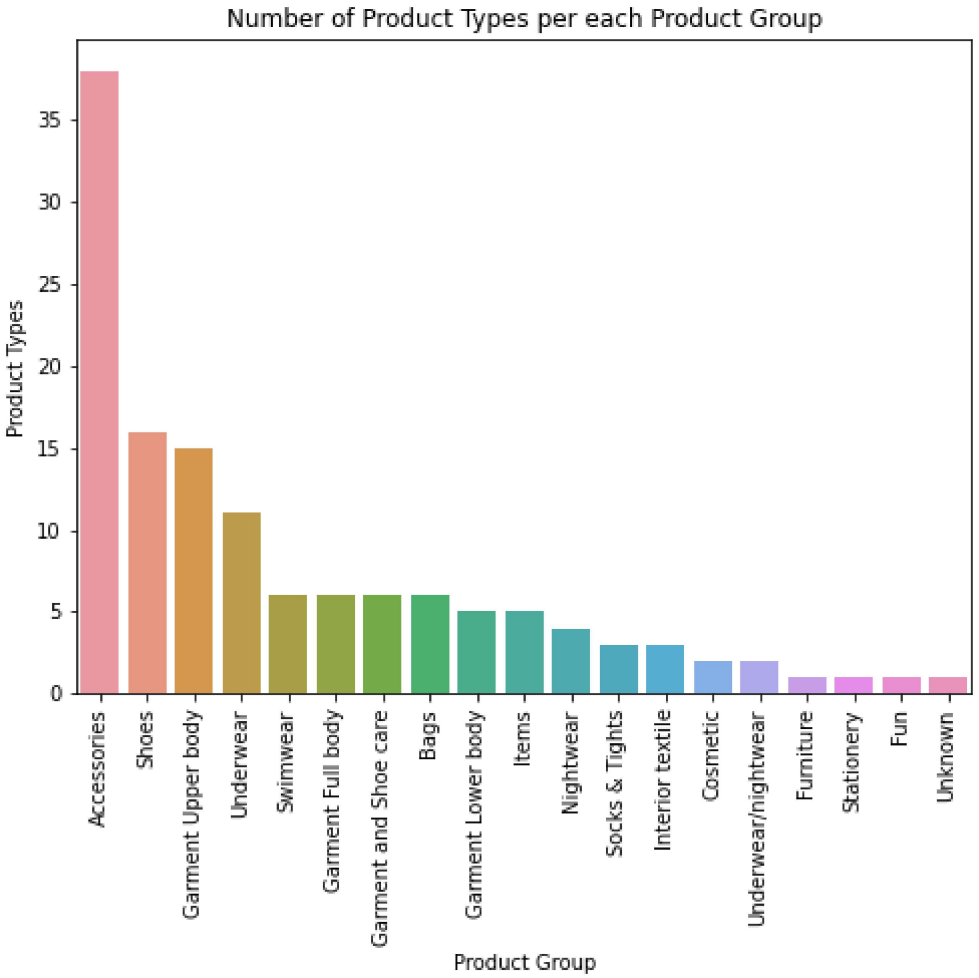
Interestingly, there are a relatively small number of distinct prices, considering the 31.7 million transactions. Similarly, out of 1.3 million customers, they purchase 104,000 different articles. Furthermore, there are only 734 unique dates. Let's examine some statistics related to this.

In [24]:
```
print(f"Percent of articles present in transactions: {round(104547/105542,3)*100}%")
print(f"Percent of articles present in transactions: {round(1362281/1371980,3)*100}%")
```

```
Percent of articles present in transactions: 99.1%
Percent of articles present in transactions: 99.3%
```

Articles data

In [25]:
```
temp = articles_df.groupby(["product_group_name"])["product_type_name"].nunique()
df = pd.DataFrame({'Product Group': temp.index,
                   'Product Types': temp.values
                  })
df = df.sort_values(['Product Types'], ascending=False)
plt.figure(figsize = (8,6))
plt.title('Number of Product Types per each Product Group')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Product Group', y="Product Types", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



In [26]:

```
stopwords = set(STOPWORDS)

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=5,
        random_state=1
    ).generate(str(data))

    fig = plt.figure(1, figsize=(10,10))
```

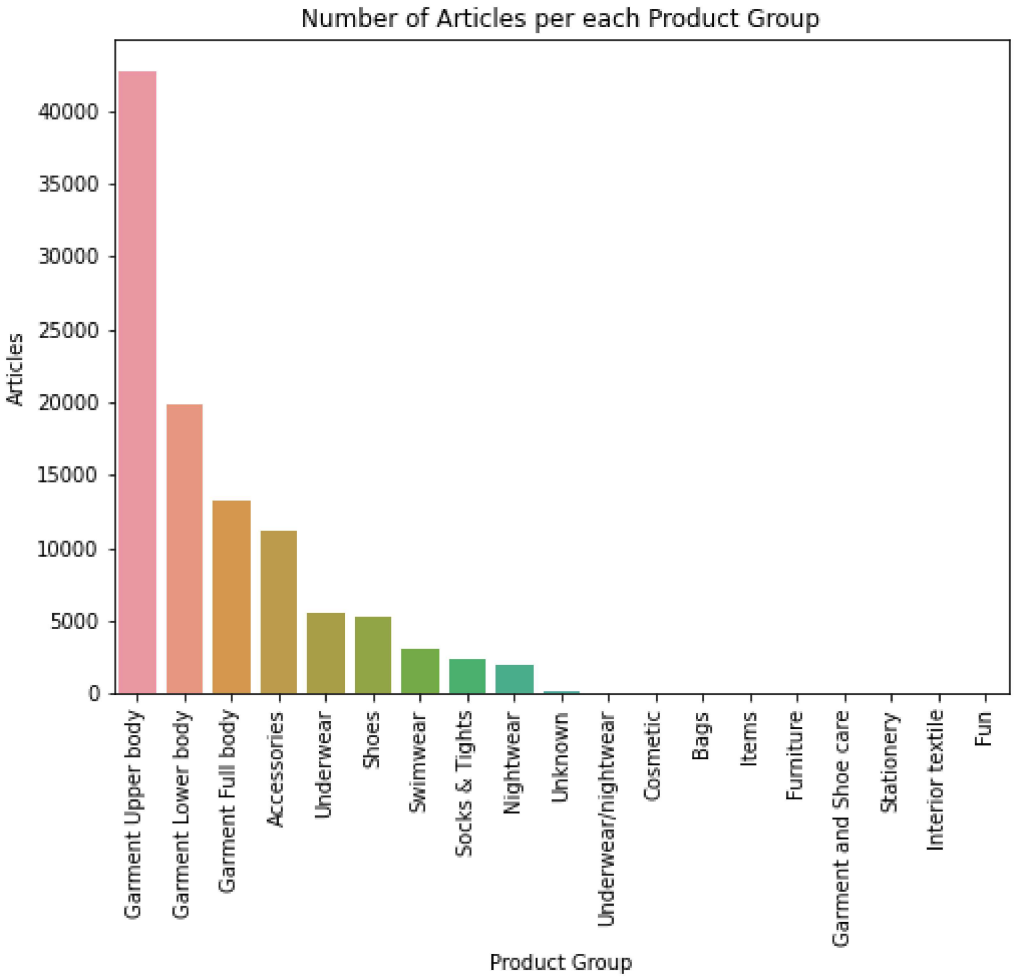
```
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=14)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()
```

In [27]:
```
show_wordcloud(articles_df["prod_name"], "Wordcloud from product name")
```

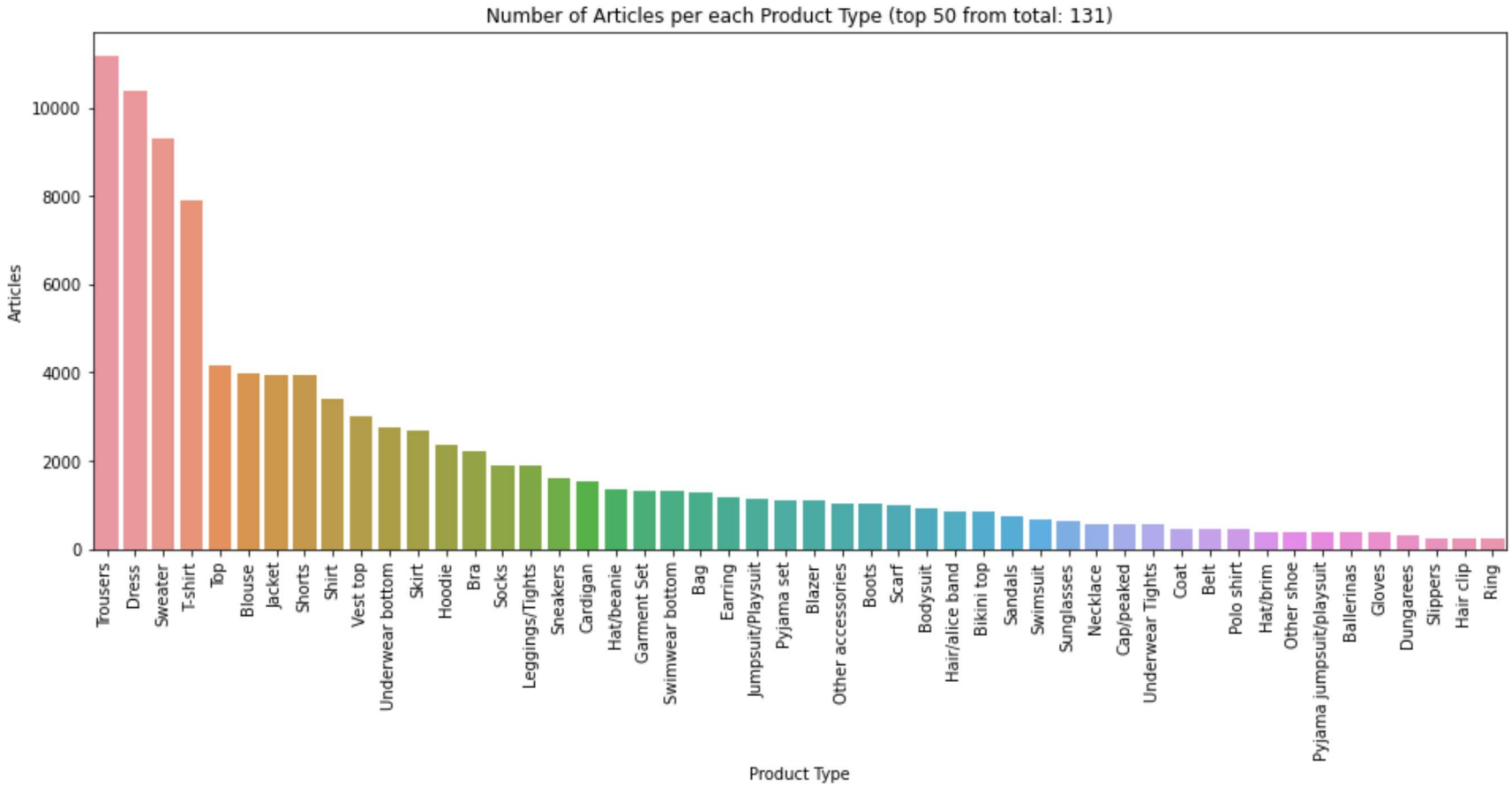
Wordcloud from product name

In [28]:
```
temp = articles_df.groupby(["product_group_name"])["article_id"].nunique()
df = pd.DataFrame({'Product Group': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (8,6))
plt.title('Number of Articles per each Product Group')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Product Group', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

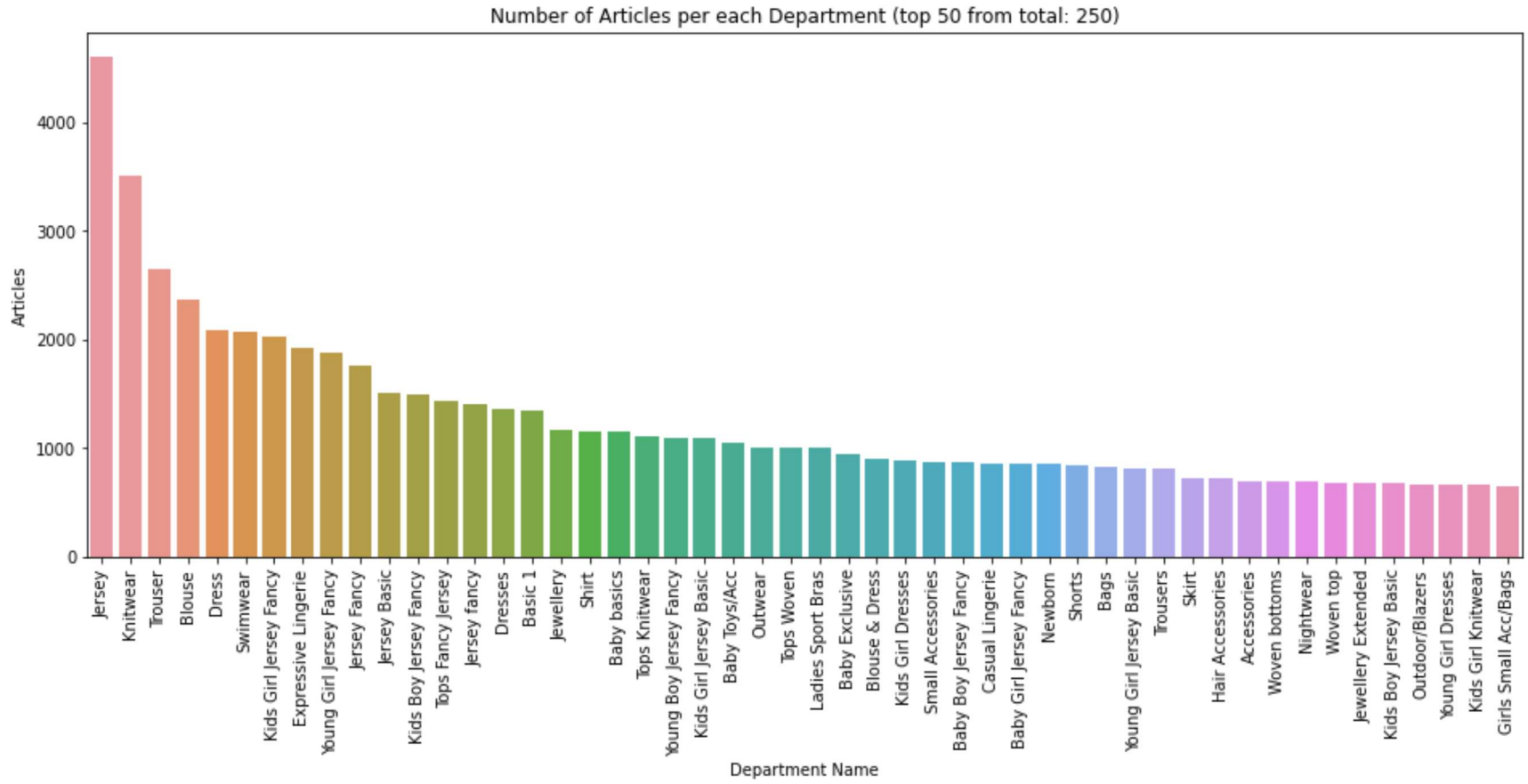


In [29]:

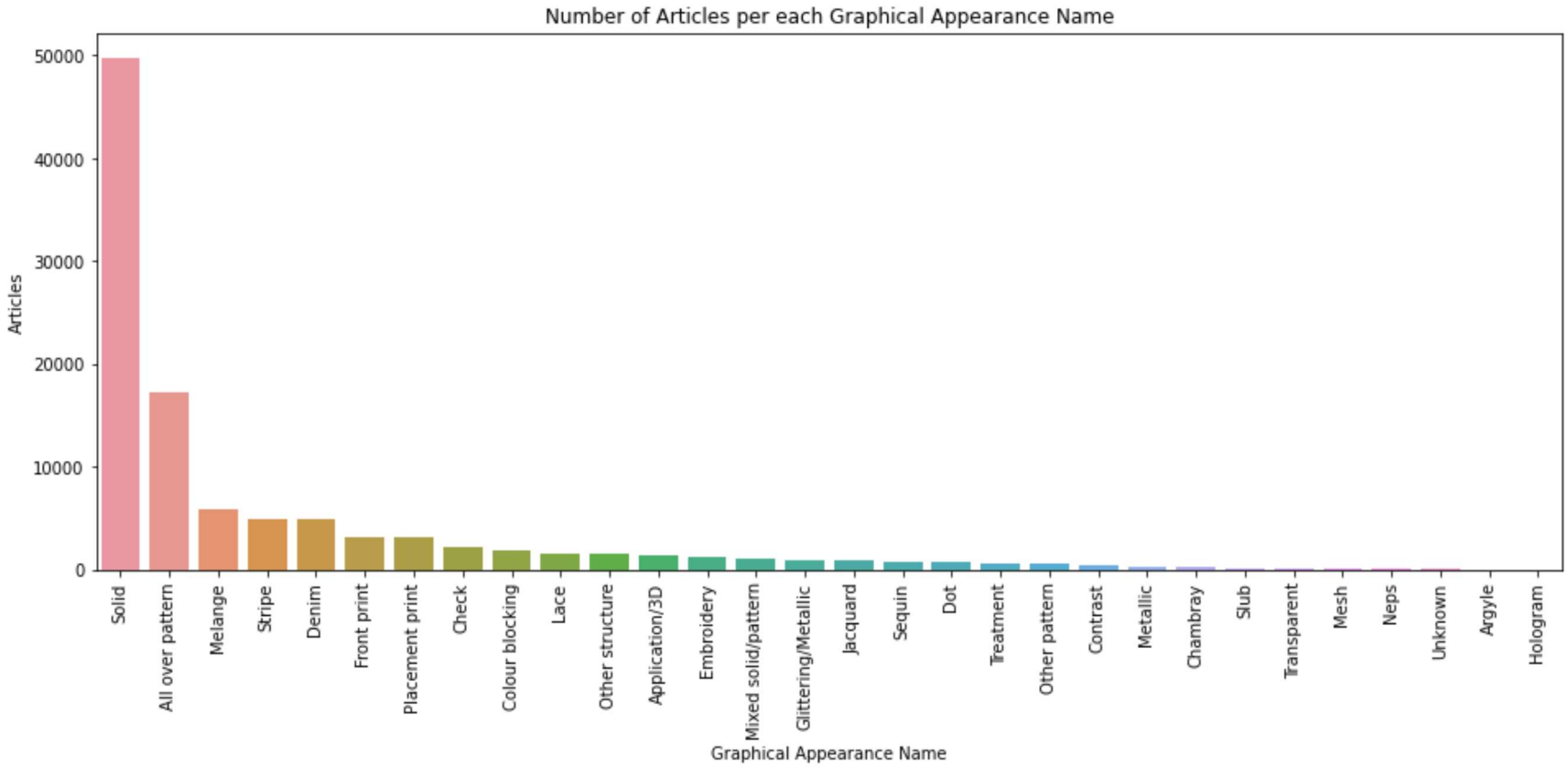
```
temp = articles_df.groupby(["product_type_name"])["article_id"].nunique()
df = pd.DataFrame({'Product Type': temp.index,
                   'Articles': temp.values
                  })
total_types = len(df['Product Type'].unique())
df = df.sort_values(['Articles'], ascending=False)[0:50]
plt.figure(figsize = (16,6))
plt.title(f'Number of Articles per each Product Type (top 50 from total: {total_types})')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Product Type', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



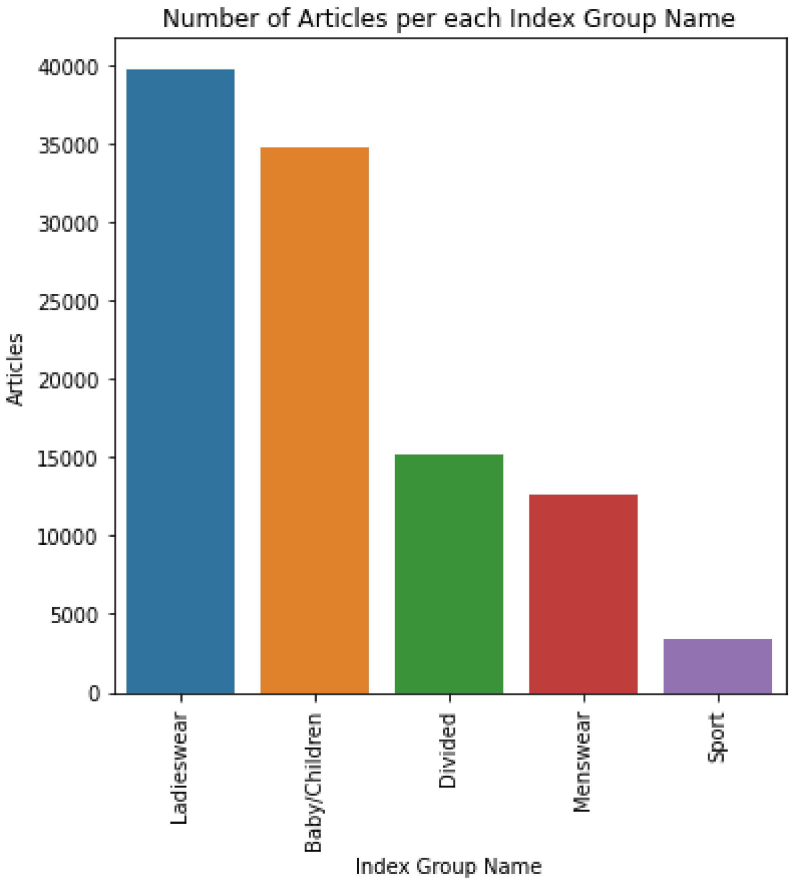
In [30]:
```
temp = articles_df.groupby(["department_name"])["article_id"].nunique()
df = pd.DataFrame({'Department Name': temp.index,
                   'Articles': temp.values
                  })
total_depts = len(df['Department Name'].unique())
df = df.sort_values(['Articles'], ascending=False).head(50)
plt.figure(figsize = (16,6))
plt.title(f'Number of Articles per each Department (top 50 from total: {total_depts})')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Department Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



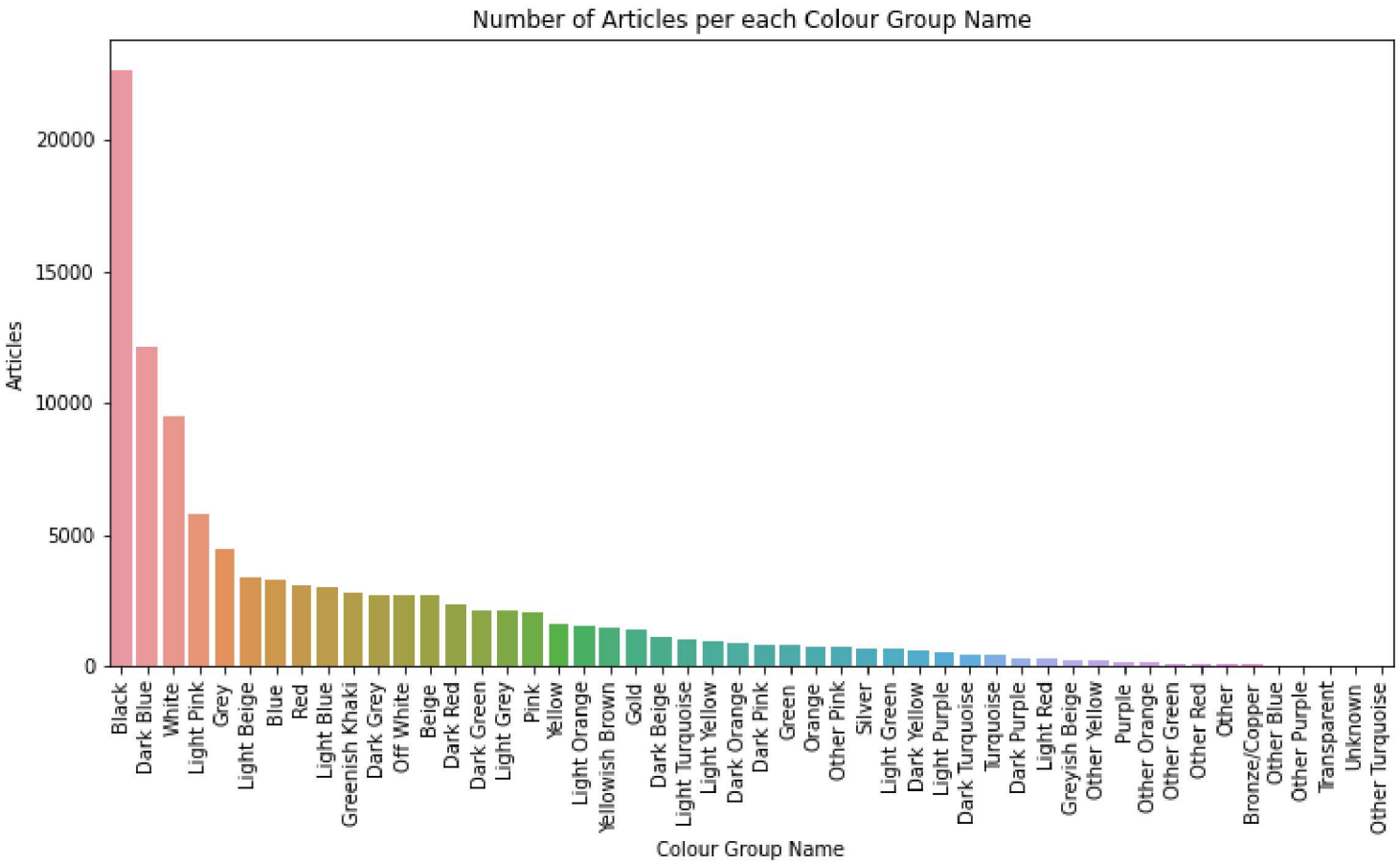
In [31]:
```
temp = articles_df.groupby(["graphical_appearance_name"])["article_id"].nunique()
df = pd.DataFrame({'Graphical Appearance Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False).head(50)
plt.figure(figsize = (16,6))
plt.title(f'Number of Articles per each Graphical Appearance Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Graphical Appearance Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



In [32]:
```
temp = articles_df.groupby(["index_group_name"])["article_id"].nunique()
df = pd.DataFrame({'Index Group Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (6,6))
plt.title(f'Number of Articles per each Index Group Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Index Group Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

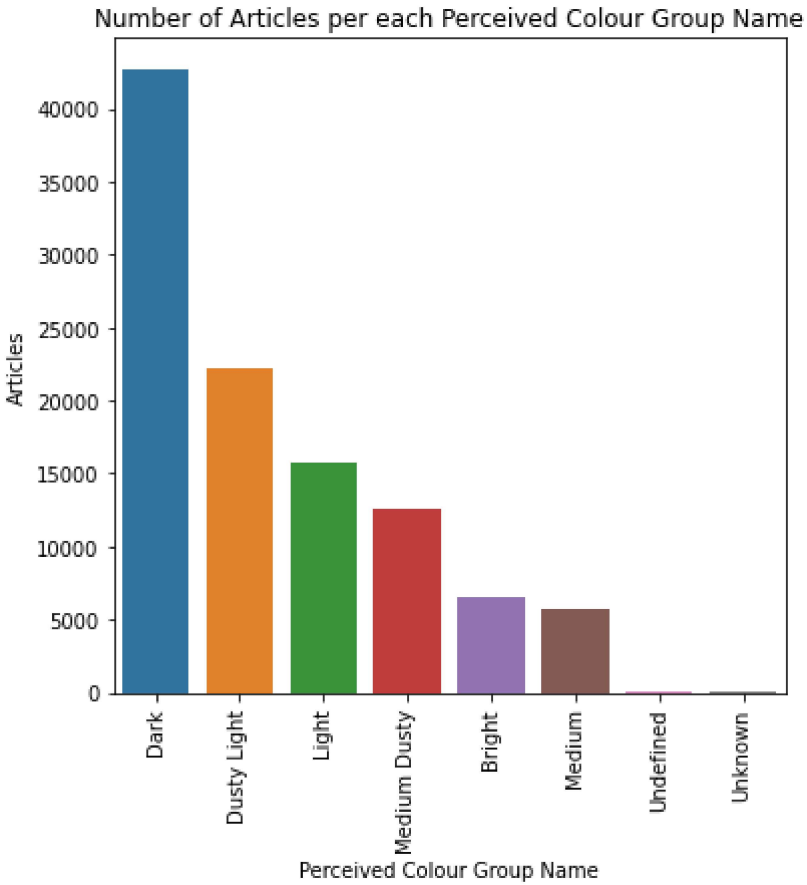


In [33]:
```
temp = articles_df.groupby(["colour_group_name"])["article_id"].nunique()
df = pd.DataFrame({'Colour Group Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (12,6))
plt.title(f'Number of Articles per each Colour Group Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Colour Group Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

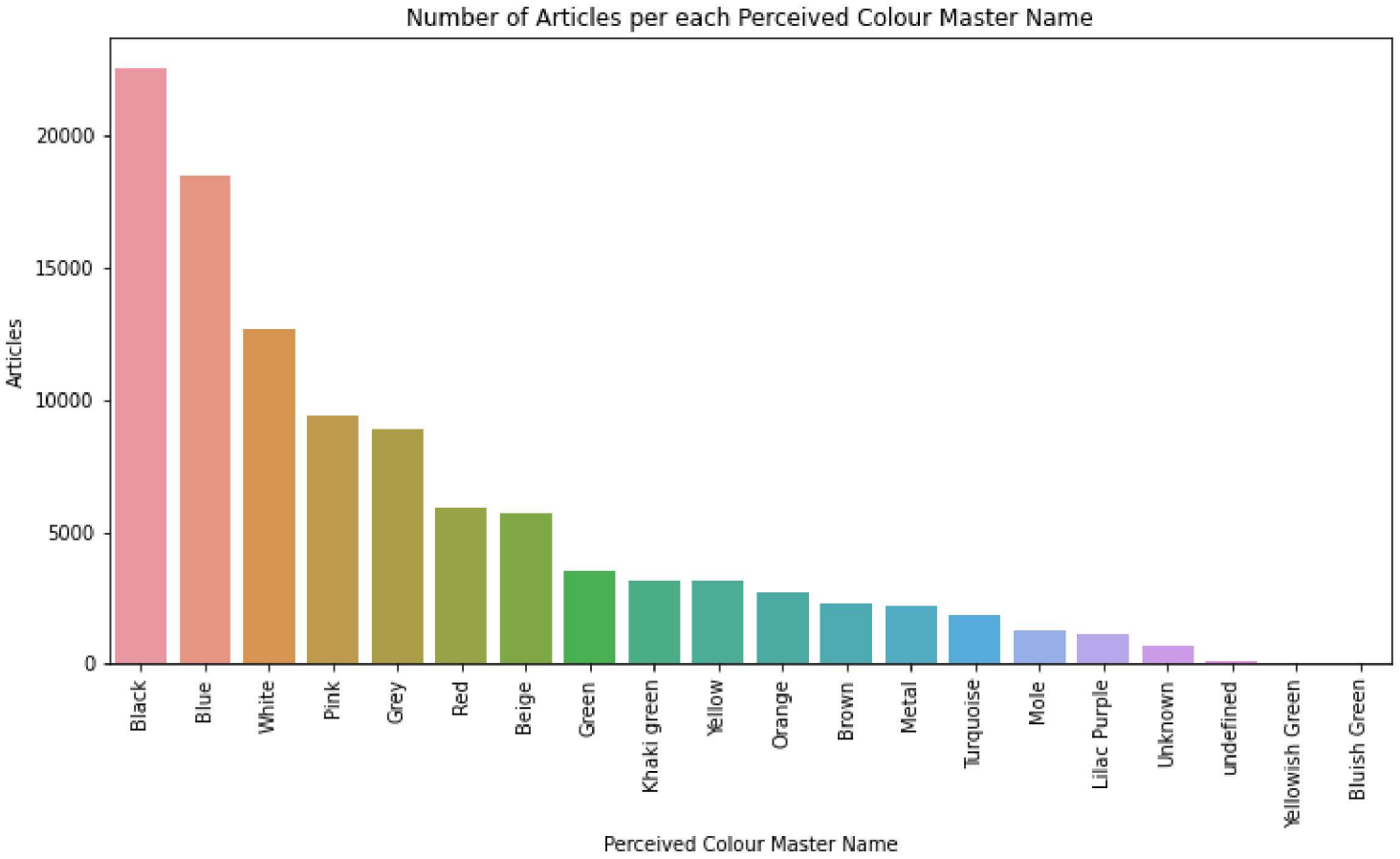


In [34]:

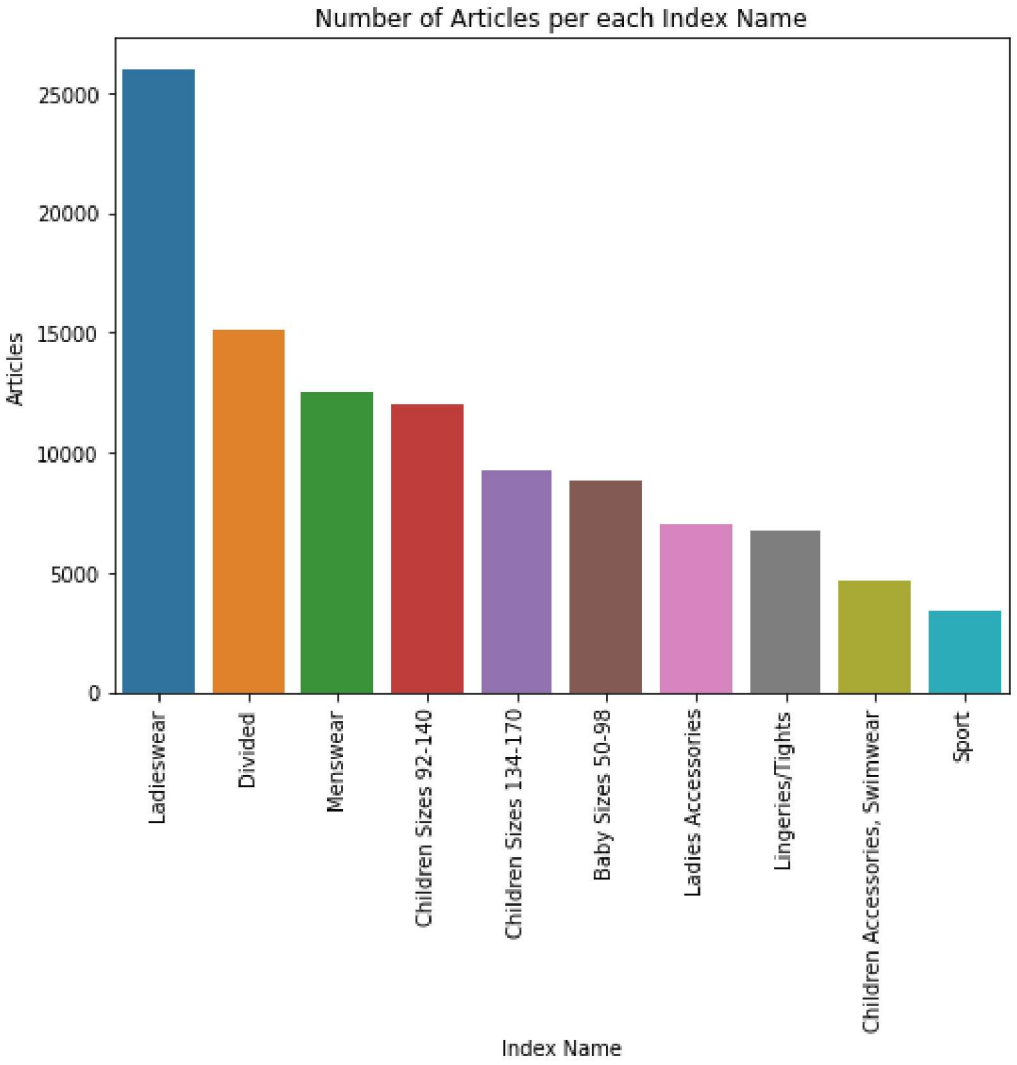
```
temp = articles_df.groupby(["perceived_colour_value_name"])["article_id"].nunique()
df = pd.DataFrame({'Perceived Colour Group Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (6,6))
plt.title(f'Number of Articles per each Perceived Colour Group Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Perceived Colour Group Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



In [35]:
```
temp = articles_df.groupby(["perceived_colour_master_name"])["article_id"].nunique()
df = pd.DataFrame({'Perceived Colour Master Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (12,6))
plt.title(f'Number of Articles per each Perceived Colour Master Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Perceived Colour Master Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

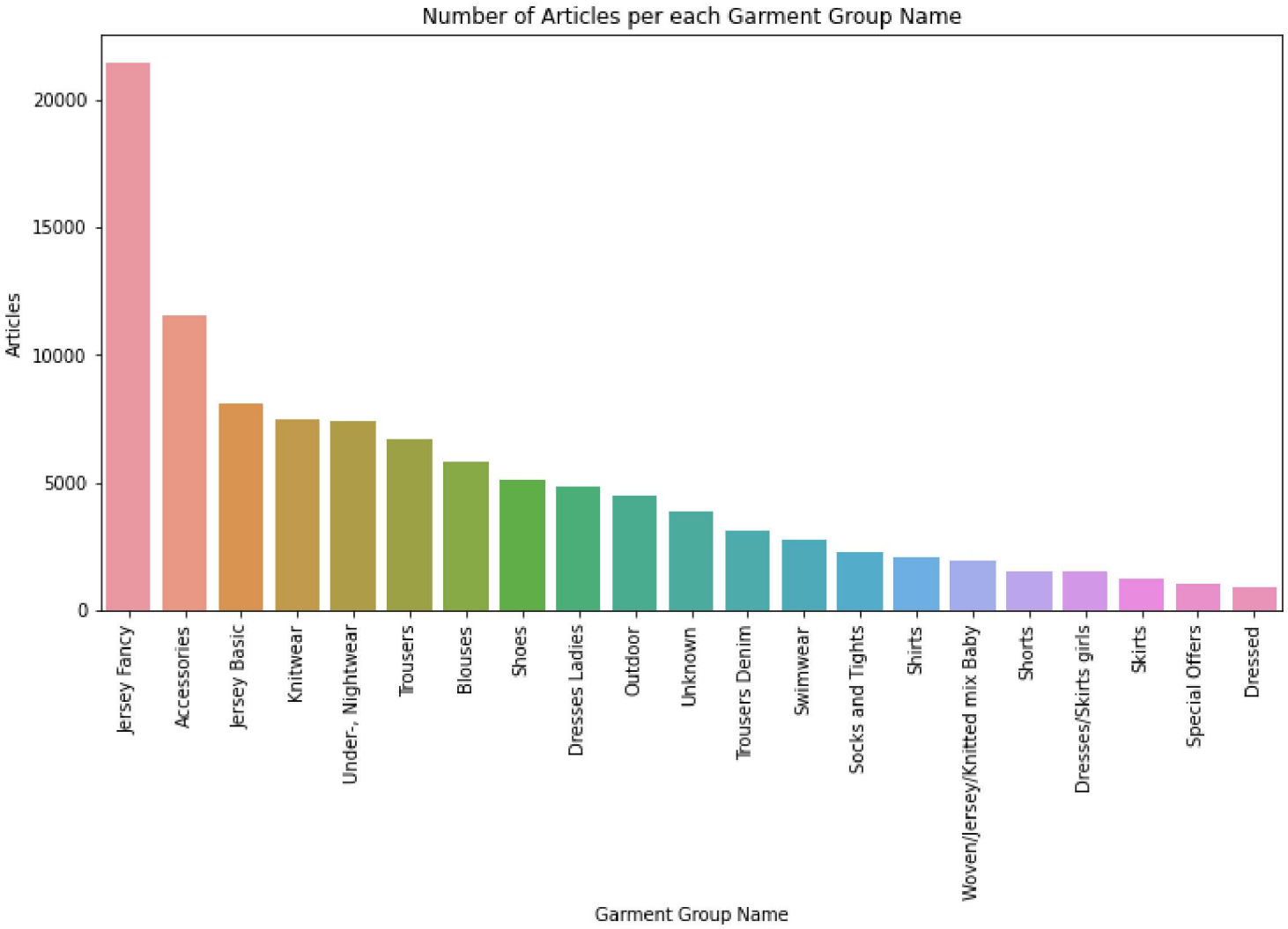


In [36]:
```
temp = articles_df.groupby(["index_name"])["article_id"].nunique()
df = pd.DataFrame({'Index Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (8,6))
plt.title(f'Number of Articles per each Index Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Index Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



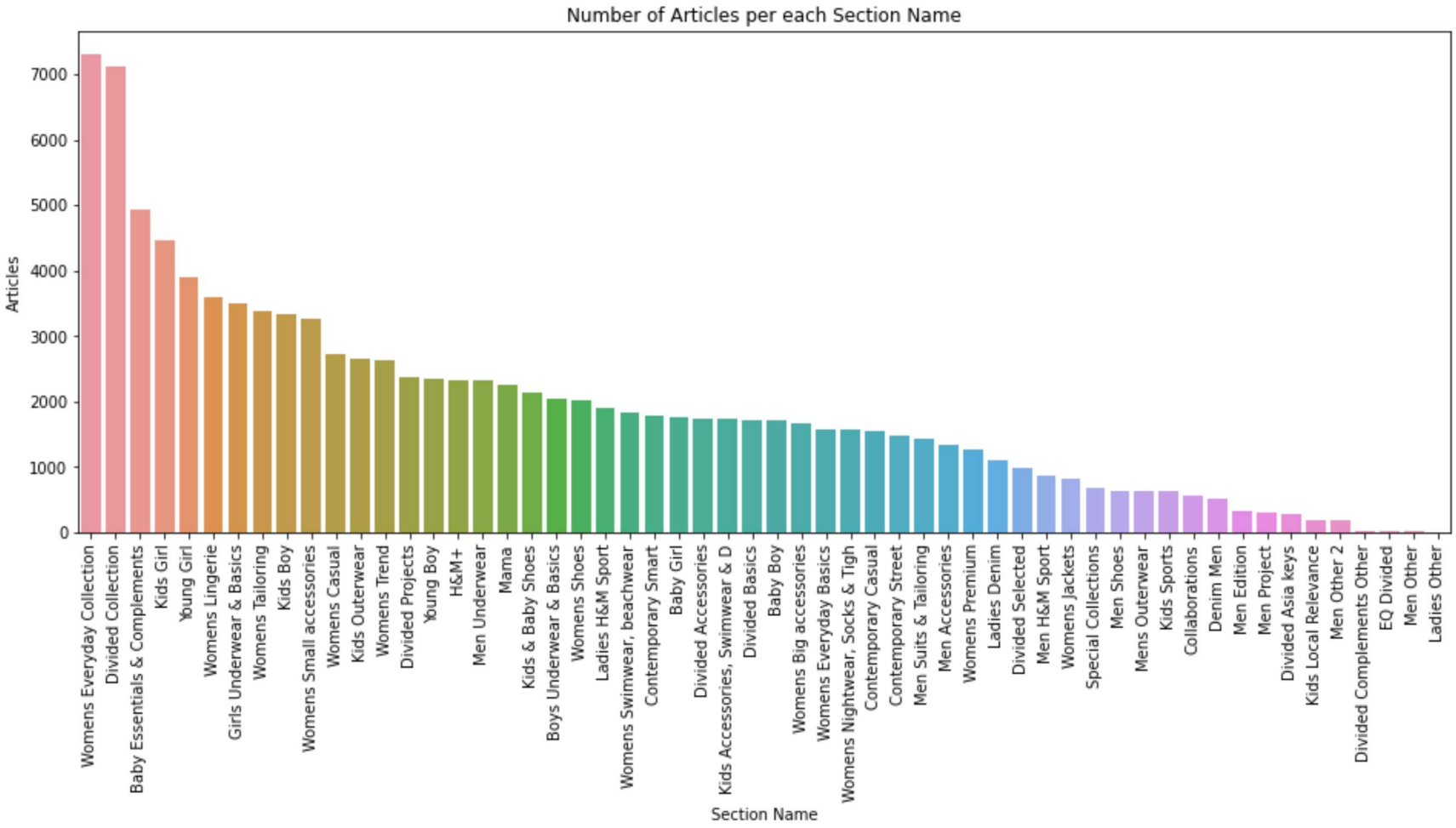
In [37]:
```
temp = articles_df.groupby(["garment_group_name"])["article_id"].nunique()
df = pd.DataFrame({'Garment Group Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (12,6))
plt.title(f'Number of Articles per each Garment Group Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Garment Group Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
```

```
locs, labels = plt.xticks()
plt.show()
```



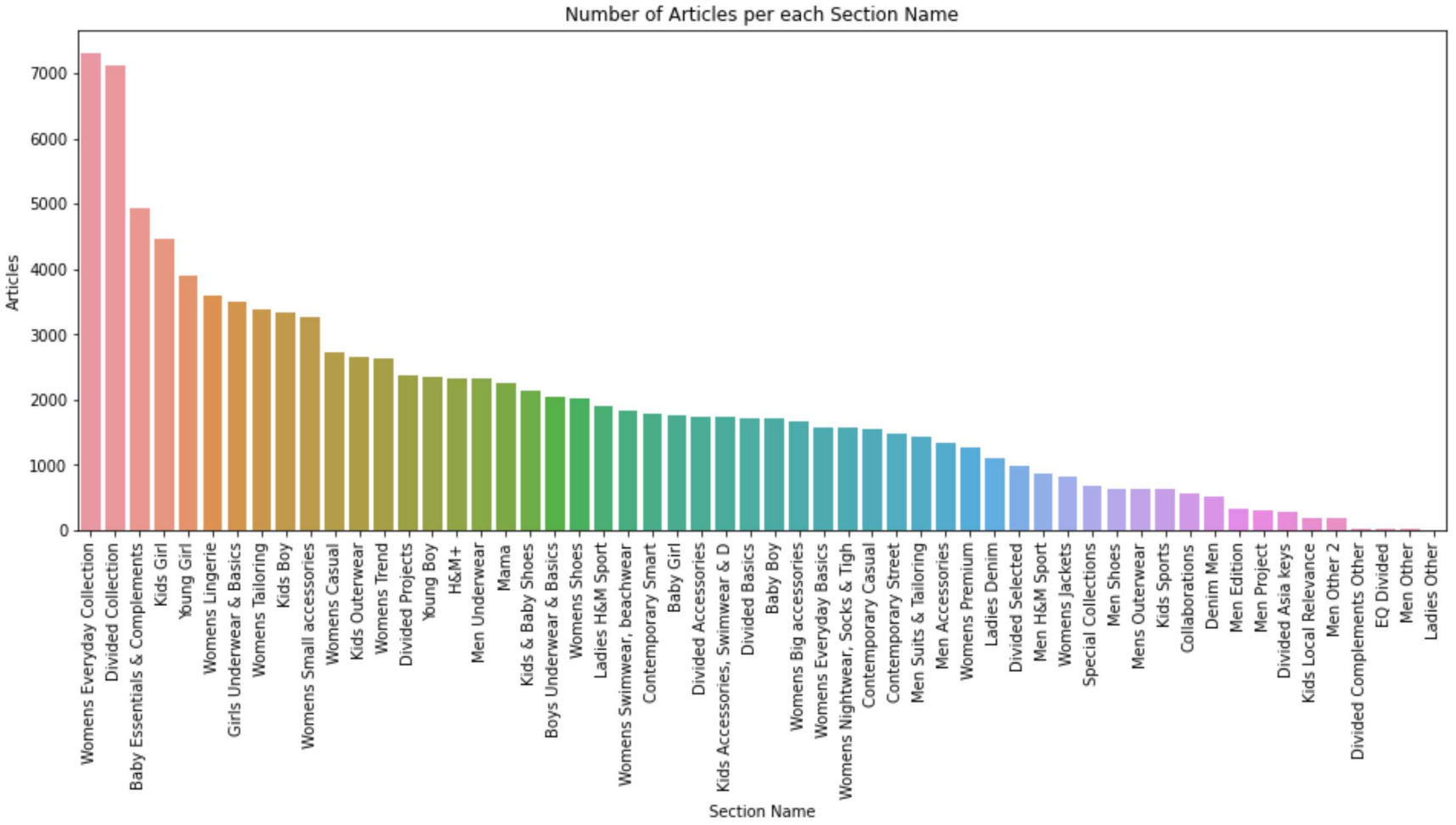
In [38]:
```
temp = articles_df.groupby(["section_name"])["article_id"].nunique()
df = pd.DataFrame({'Section Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (16,6))
plt.title(f'Number of Articles per each Section Name')
sns.set_color_codes("pastel")
```

```
s = sns.barplot(x = 'Section Name', y="Articles", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



In [39]:
```
temp = articles_df.groupby(["section_name"])["article_id"].nunique()
df = pd.DataFrame({'Section Name': temp.index,
                   'Articles': temp.values
                  })
df = df.sort_values(['Articles'], ascending=False)
plt.figure(figsize = (16,6))
plt.title(f'Number of Articles per each Section Name')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Section Name', y="Articles", data=df)
```

```
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

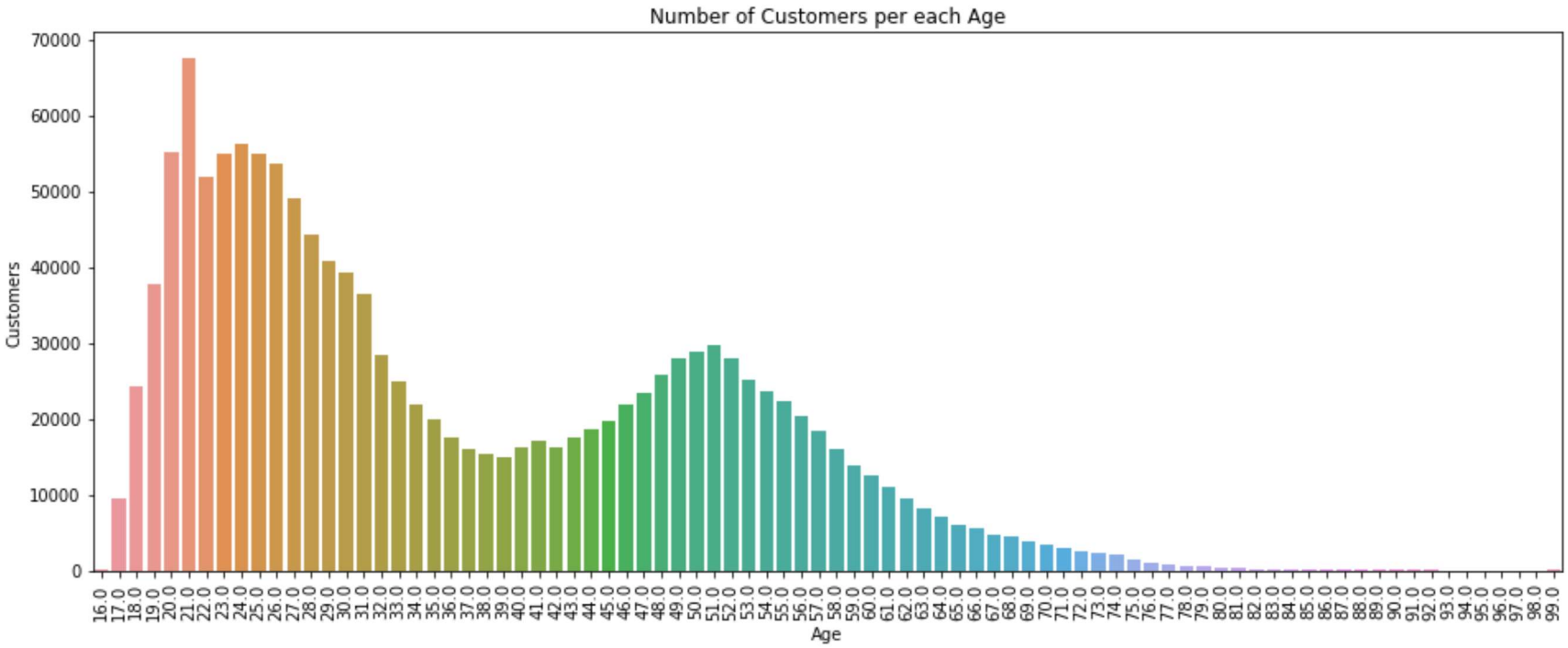


In [40]:
```
show_wordcloud(articles_df["detail_desc"], "Wordcloud from detailed description of articles")
```

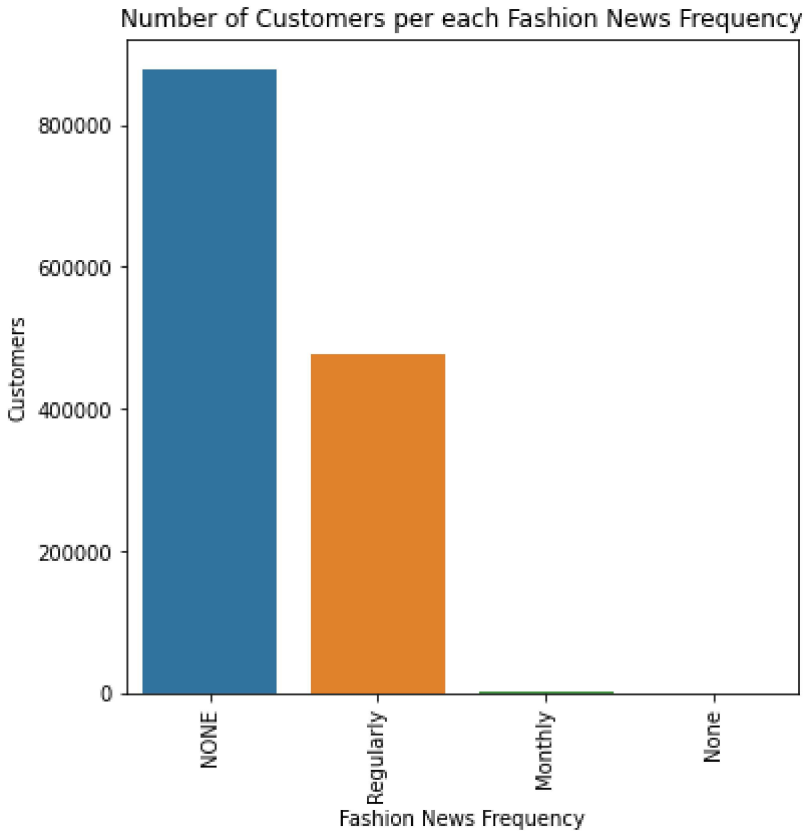
Wordcloud from detailed description of articles

Customers data

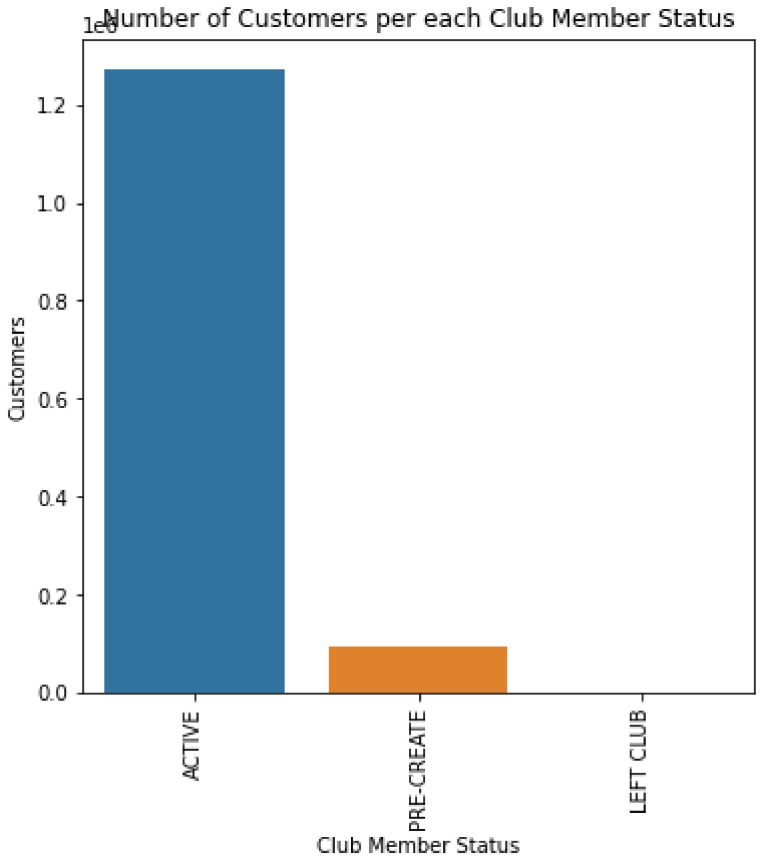
In [41]:
```
temp = customers_df.groupby(["age"])["customer_id"].count()
df = pd.DataFrame({'Age': temp.index,
                   'Customers': temp.values
                  })
df = df.sort_values(['Age'], ascending=False)
plt.figure(figsize = (16,6))
plt.title(f'Number of Customers per each Age')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Age', y="Customers", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```



In [42]:
```
temp = customers_df.groupby(["fashion_news_frequency"])["customer_id"].count()
df = pd.DataFrame({'Fashion News Frequency': temp.index,
                   'Customers': temp.values
                  })
df = df.sort_values(['Customers'], ascending=False)
plt.figure(figsize = (6,6))
plt.title(f'Number of Customers per each Fashion News Frequency')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Fashion News Frequency', y="Customers", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

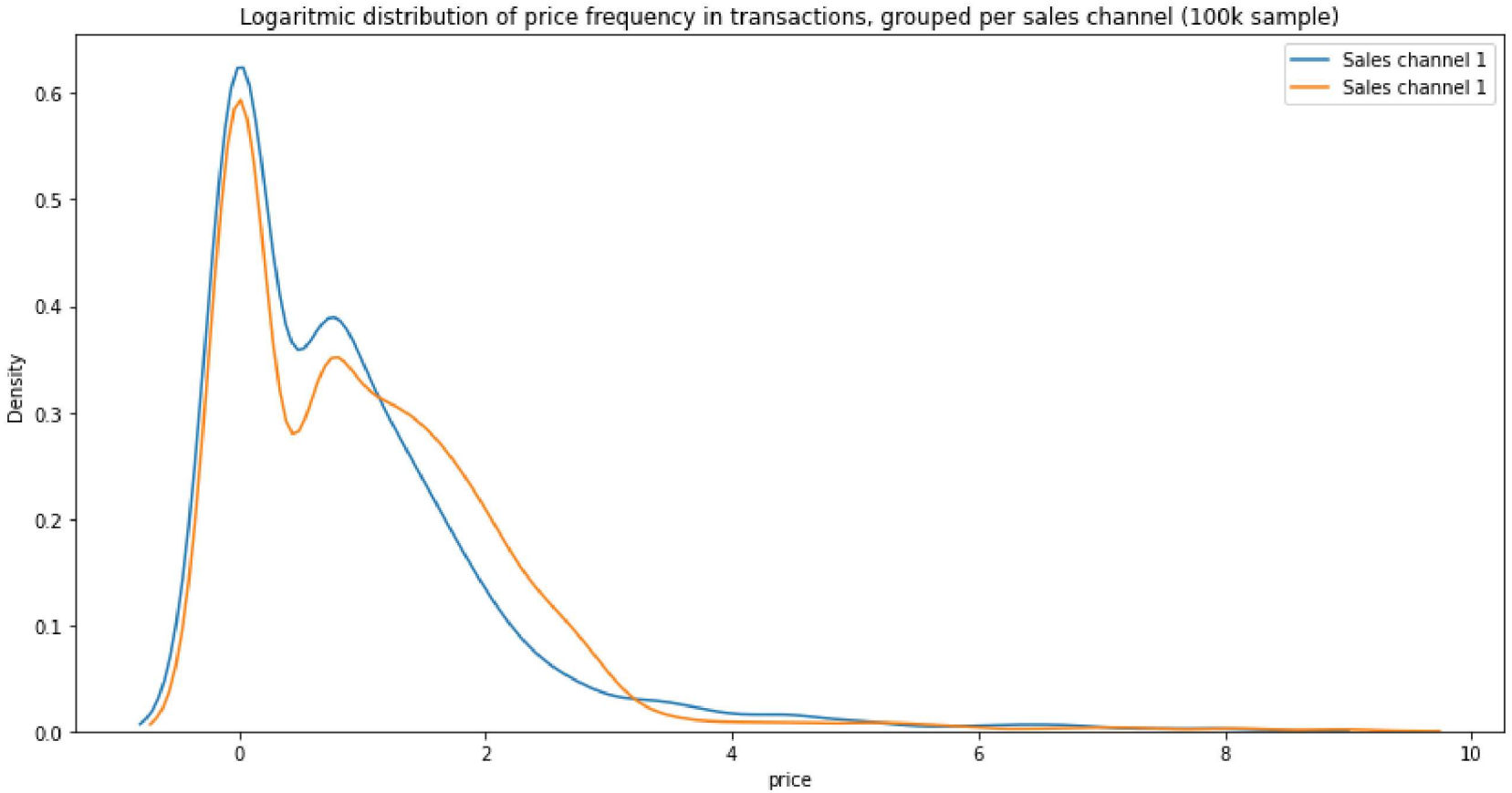


In [43]:
```
temp = customers_df.groupby(["club_member_status"])["customer_id"].count()
df = pd.DataFrame({'Club Member Status': temp.index,
                   'Customers': temp.values
                  })
df = df.sort_values(['Customers'], ascending=False)
plt.figure(figsize = (6,6))
plt.title(f'Number of Customers per each Club Member Status')
sns.set_color_codes("pastel")
s = sns.barplot(x = 'Club Member Status', y="Customers", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

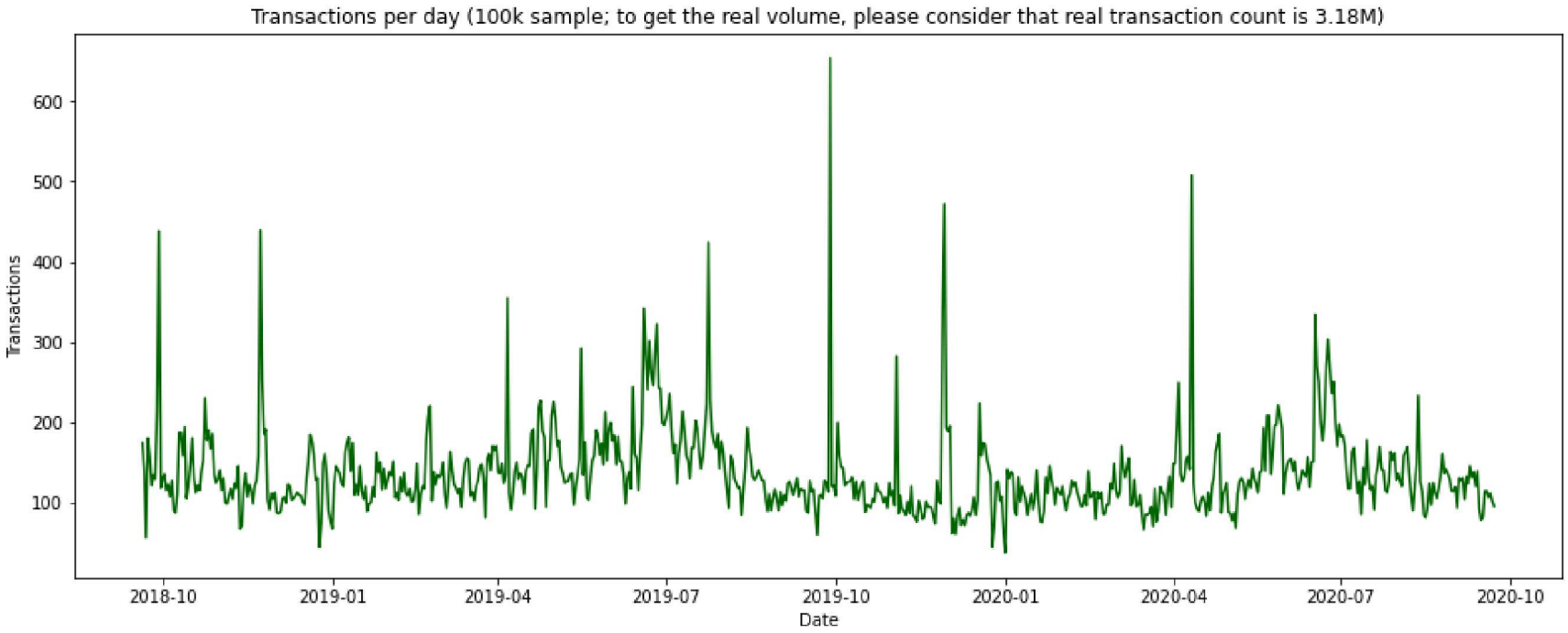


Transactions data

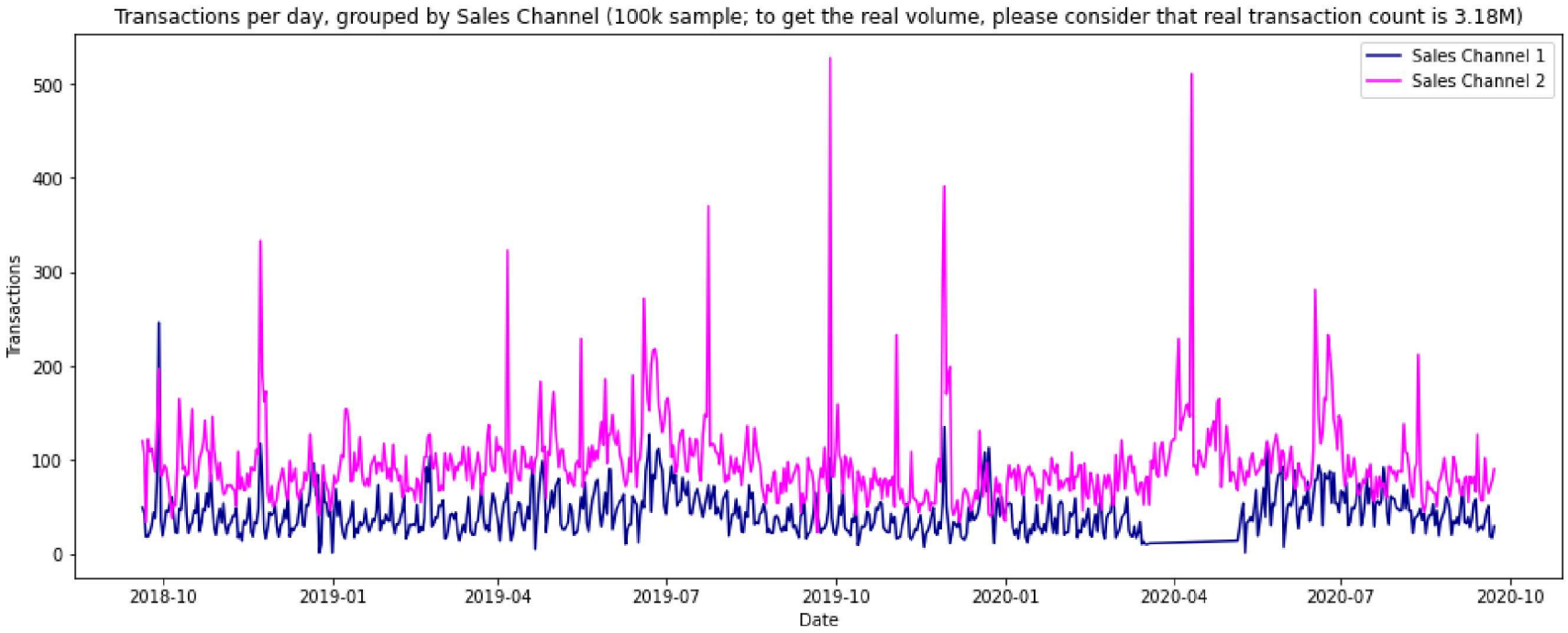
In [44]:
```
df = transactions_train_df.sample(100_000)
fig, ax = plt.subplots(1, 1, figsize=(14, 7))
sns.kdeplot(np.log(df.loc[df["sales_channel_id"]==1].price.value_counts()))
sns.kdeplot(np.log(df.loc[df["sales_channel_id"]==2].price.value_counts()))
ax.legend(labels=['Sales channel 1', 'Sales channel 1'])
plt.title("Logaritmic distribution of price frequency in transactions, grouped per sales channel (100k sample)")
plt.show()
```



In [45]:
```
df = transactions_train_df.sample(100_000).groupby(["t_dat"])["article_id"].count().reset_index()
df["t_dat"] = df["t_dat"].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
df.columns = ["Date", "Transactions"]
fig, ax = plt.subplots(1, 1, figsize=(16,6))
plt.plot(df["Date"], df["Transactions"], color="Darkgreen")
plt.xlabel("Date")
plt.ylabel("Transactions")
plt.title(f"Transactions per day (100k sample; to get the real volume, please consider that real transaction count is {
plt.show()
```



In [46]:
```
df = transactions_train_df.sample(100_000).groupby(["t_dat", "sales_channel_id"])["article_id"].count().reset_index()
df["t_dat"] = df["t_dat"].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
df.columns = ["Date", "Sales Channel Id", "Transactions"]
fig, ax = plt.subplots(1, 1, figsize=(16,6))
g1 = ax.plot(df.loc[df["Sales Channel Id"]==1, "Date"], df.loc[df["Sales Channel Id"]==1, "Transactions"], label="Sales
g2 = ax.plot(df.loc[df["Sales Channel Id"]==2, "Date"], df.loc[df["Sales Channel Id"]==2, "Transactions"], label="Sales
plt.xlabel("Date")
plt.ylabel("Transactions")
ax.legend()
plt.title(f"Transactions per day, grouped by Sales Channel (100k sample; to get the real volume, please consider that re
plt.show()
```



In [47]:
```
df = transactions_train_df.groupby(["t_dat", "sales_channel_id"])["article_id"].nunique().reset_index()
df["t_dat"] = df["t_dat"].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
df.columns = ["Date", "Sales Channel Id", "Unique Articles"]
fig, ax = plt.subplots(1, 1, figsize=(16,6))
g1 = ax.plot(df.loc[df["Sales Channel Id"]==1, "Date"], df.loc[df["Sales Channel Id"]==1, "Unique Articles"], label="Sa
g2 = ax.plot(df.loc[df["Sales Channel Id"]==2, "Date"], df.loc[df["Sales Channel Id"]==2, "Unique Articles"], label="Sa
plt.xlabel("Date")
plt.ylabel("Unique Articles / Day")
ax.legend()
plt.title(f"Unique articles per day, grouped by Sales Channel")
plt.show()
```

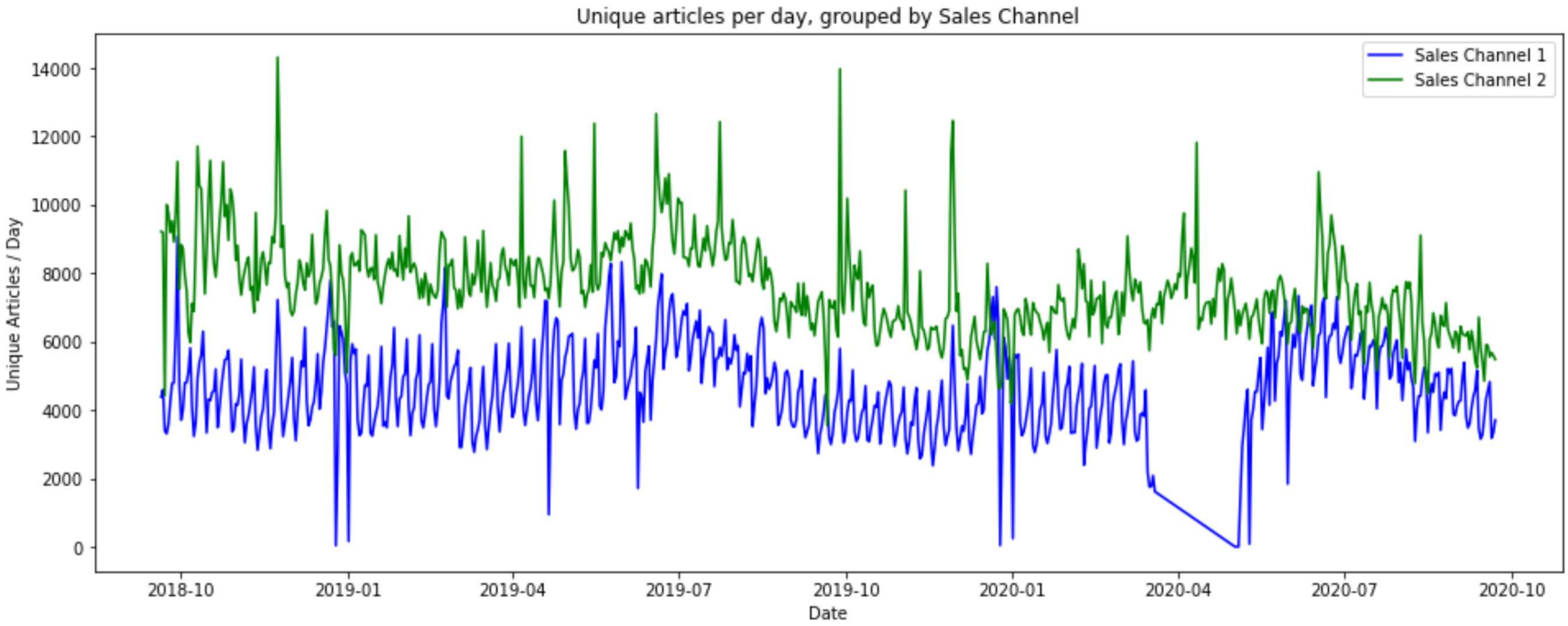


Image data

There are 105542 articles and 105100 different images. Let's check first which articles does not have corresponding images.

The `article_id` corresponds to digits from 2nd to the last of the image name. The digits from 2nd to 7th of image name correspond to product code (`product_code`).

In [48]:
```
image_name_df = pd.DataFrame(images_names, columns = ["image_name"])
image_name_df["article_id"] = image_name_df["image_name"].apply(lambda x: int(x[1:]))
```

In [49]:
```
image_name_df.head()
```

Out[49]:

	image_name	article_id
0	0570177001	570177001
1	0575944002	575944002
2	0576129001	576129001
3	0570806001	570806001
4	0572155001	572155001

In [50]:
```
image_article_df = articles_df[["article_id", "product_code", "product_group_name", "product_type_name"]].merge(image_n
print(image_article_df.shape)
image_article_df.head()
```

```
(105542, 5)
```

Out[50]:

	article_id	product_code	product_group_name	product_type_name	image_name
0	108775015	108775	Garment Upper body	Vest top	0108775015
1	108775044	108775	Garment Upper body	Vest top	0108775044
2	108775051	108775	Garment Upper body	Vest top	0108775051
3	110065001	110065	Underwear	Bra	0110065001
4	110065002	110065	Underwear	Bra	0110065002

Products without images.

In [51]:
```
article_no_image_df = image_article_df.loc[image_article_df.image_name.isna()]
print(article_no_image_df.shape)
article_no_image_df.head()
```

```
(442, 5)
```

Out[51]:

	article_id	product_code	product_group_name	product_type_name	image_name
88	174057028	174057	Nightwear	Pyjama jumpsuit/playsuit	NaN
117	179208001	179208	Garment Lower body	Leggings/Tights	NaN
258	212042043	212042	Shoes	Sneakers	NaN
259	212042066	212042	Shoes	Sneakers	NaN
261	212629004	212629	Garment Full body	Dress	NaN

In [52]:
```
print("Product codes with some missing images: ", article_no_image_df.product_code.nunique())
print("Product groups with some missing images: ", list(article_no_image_df.product_group_name.unique()))
```

```
Product codes with some missing images:  372
Product groups with some missing images:  ['Nightwear', 'Garment Lower body', 'Shoes', 'Garment Full body', 'Accessorie
s', 'Garment Upper body', 'Underwear', 'Socks & Tights', 'Swimwear', 'Cosmetic']
```

Let's visualize few images.

In [53]:
```
def plot_image_samples(image_article_df, product_group_name, cols=1, rows=-1):
    image_path = "/input/h-and-m-personalized-fashion-recommendations/images/"
    _df = image_article_df.loc[image_article_df.product_group_name==product_group_name]
    article_ids = _df.article_id.values[0:cols*rows]
    plt.figure(figsize=(2 + 3 * cols, 2 + 4 * rows))
    for i in range(cols * rows):
        article_id = ("0" + str(article_ids[i]))[-10:]
        plt.subplot(rows, cols, i + 1)
        plt.axis('off')
        plt.title(f"{product_group_name} {article_id[:3]}\n{article_id}.jpg")
        image = Image.open(f"{image_path}{article_id[:3]}/{article_id}.jpg")
        plt.imshow(image)
```

Let's choose from some product group name.

In [54]:
```
print(image_article_df.product_group_name.unique())
```

```
['Garment Upper body' 'Underwear' 'Socks & Tights' 'Garment Lower body'
 'Accessories' 'Items' 'Nightwear' 'Unknown' 'Underwear/nightwear' 'Shoes'
 'Swimwear' 'Garment Full body' 'Cosmetic' 'Interior textile' 'Bags'
 'Furniture' 'Garment and Shoe care' 'Fun' 'Stationery']
```

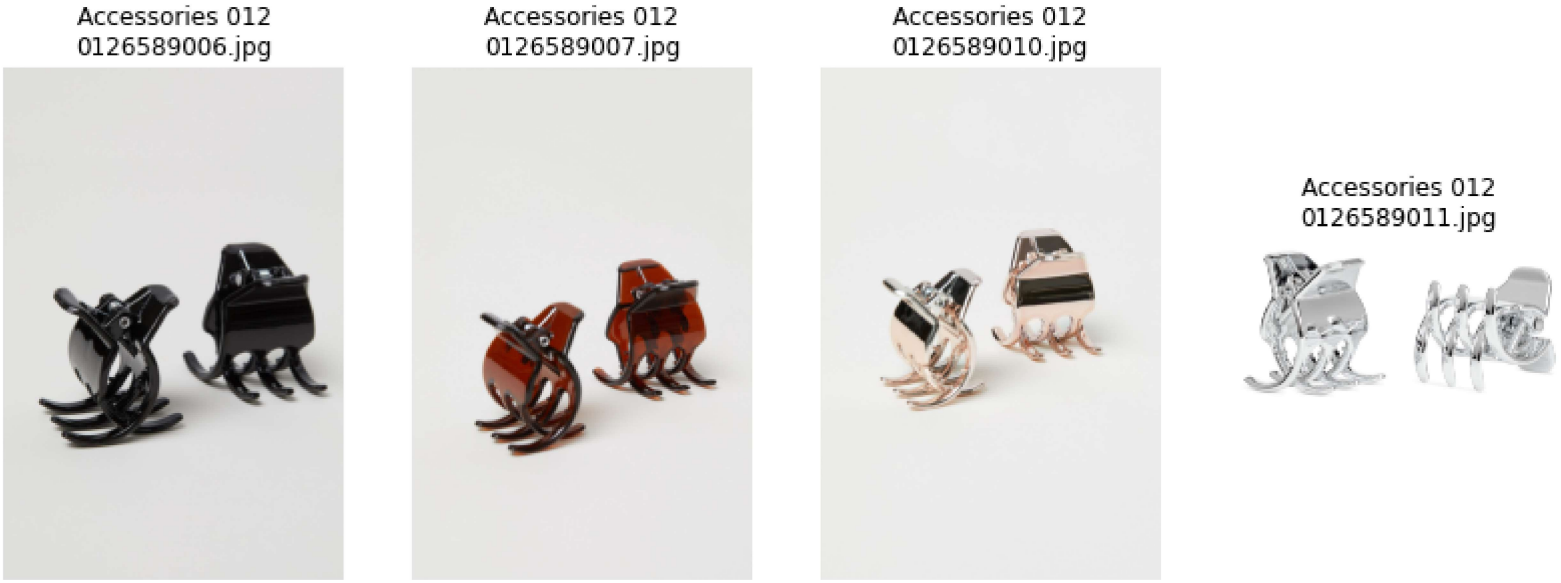
We will represent images grouped on product group name.

In [55]:
```
plot_image_samples(image_article_df, "Garment Lower body", 4, 2)
```

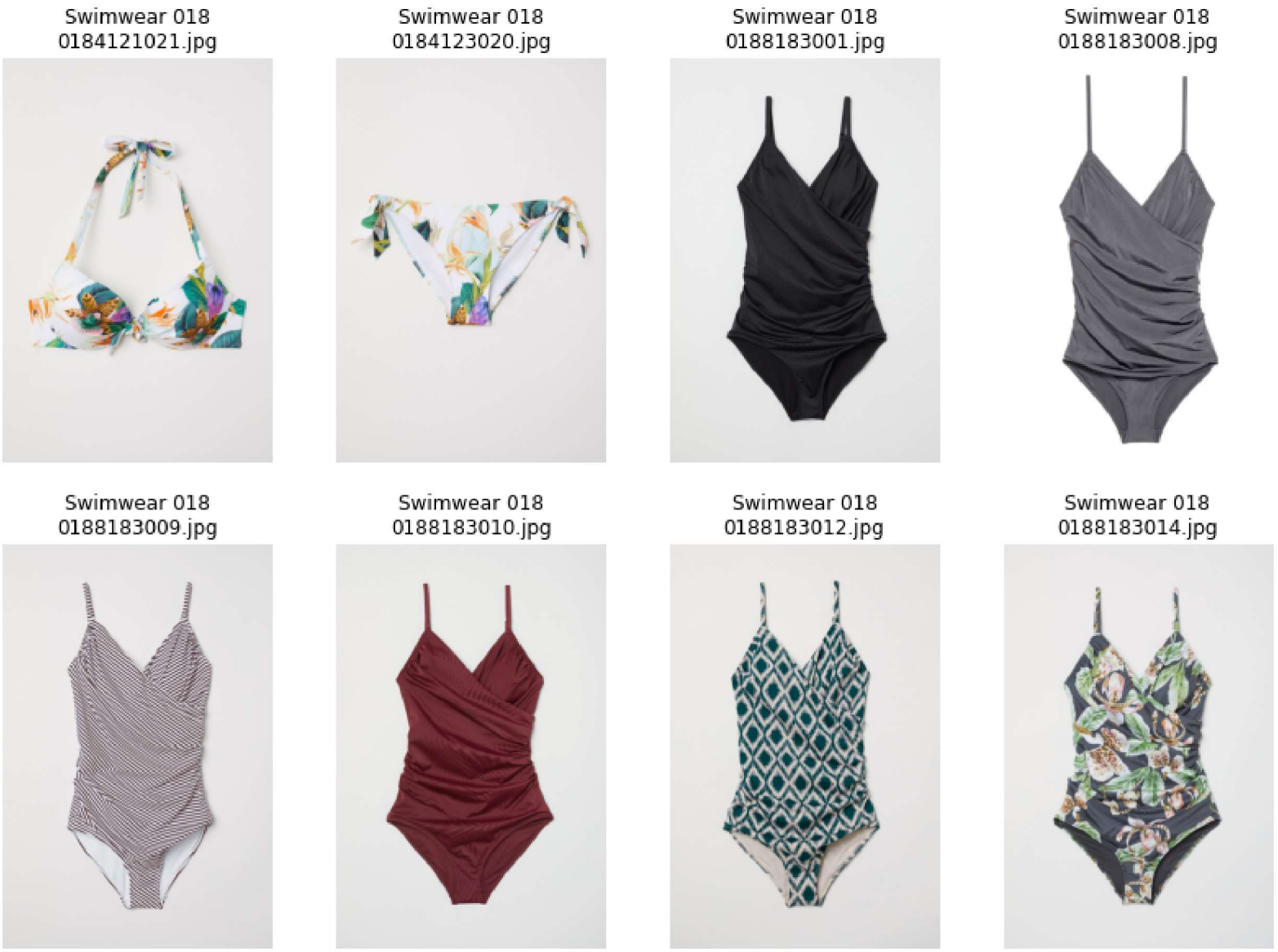
In [56]:
```
plot_image_samples(image_article_df, "Stationery", 4, 1)
```

In [57]:
```
plot_image_samples(image_article_df, "Fun", 2, 1)
```

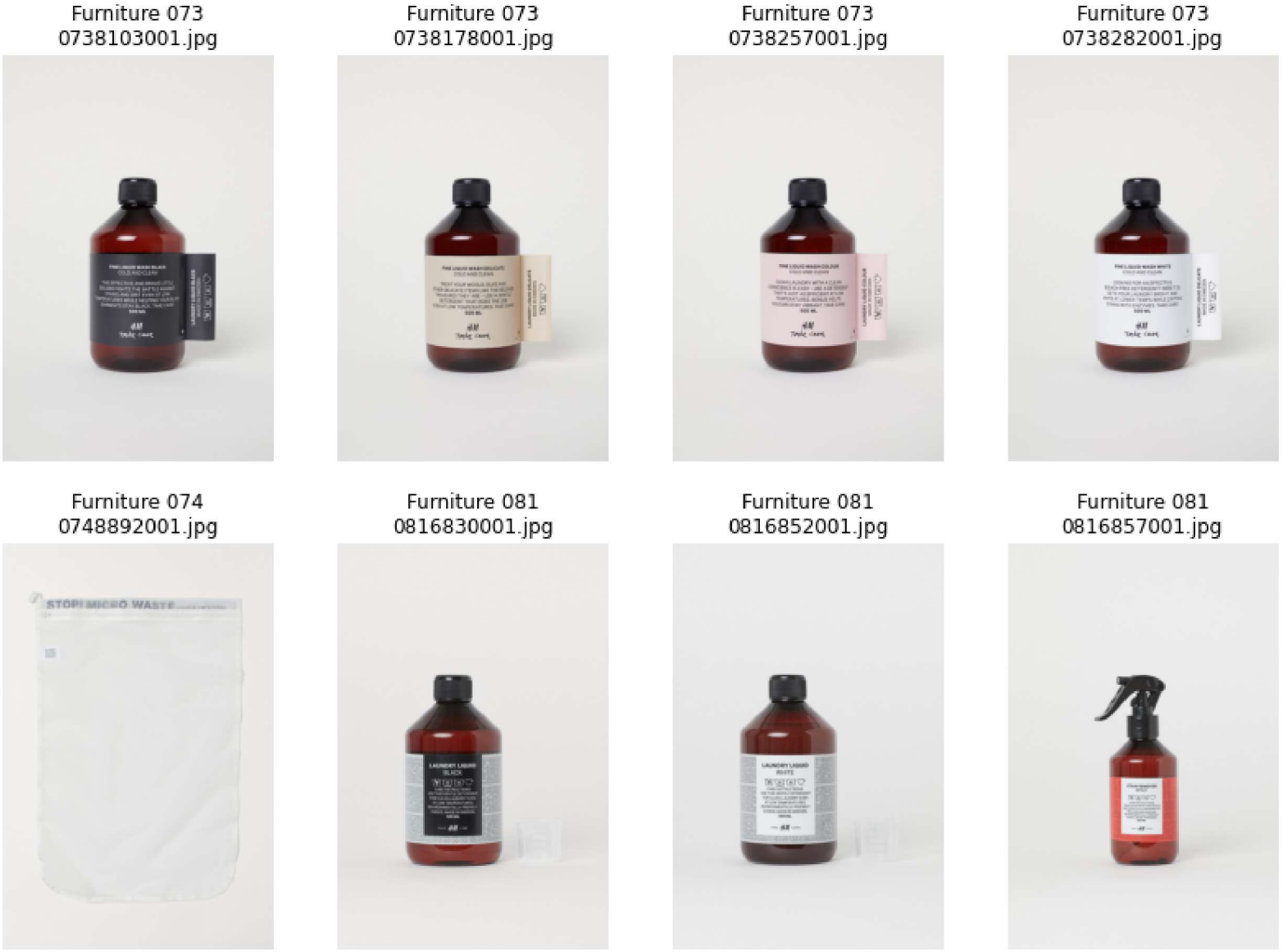
In [58]:
```
plot_image_samples(image_article_df, "Accessories", 4, 1)
```



In [59]:
```
plot_image_samples(image_article_df, "Swimwear", 4, 2)
```



In [60]:
```
plot_image_samples(image_article_df, "Furniture", 4, 2)
```



In [61]:
```
plot_image_samples(image_article_df, "Cosmetic", 4, 1)
```

In [62]:
```
plot_image_samples(image_article_df, "Bags", 4, 3)
```

Bags 063
0633152012.jpg

Bags 065
0650534001.jpg

Bags 065
0650534002.jpg

Bags 068
0682238003.jpg

Bags 068
0682238013.jpg

Bags 068
0682238030.jpg

Bags 072
0721805001.jpg

Bags 072
0721805010.jpg

Bags 073
0736631005.jpg

Bags 075
0753475001.jpg

Bags 075
0753475008.jpg

Bags 075
0753475022.jpg

For the result, we apply the following simplified logic:

- if there are articles for a certain client, pick the most recent buys;
- if there are not articles for a certain client, just pick the most frequently buyed articles.

In [63]:
```
transactions_train_df = transactions_train_df.sort_values(["customer_id", "t_dat"], ascending=False)
```

In [64]:
```
transactions_train_df.head()
```

Out[64]:

	t_dat	customer_id	article_id	price	sales_channel_id
19867243	2019-12-04	ffffd9ac14e89946416d80e791d064701994755c3ab686...	806050001	0.084729	2
27806865	2020-06-22	ffffd7744cebcf3aca44ae7049d2a94b87074c3d4ffe38...	882810001	0.016932	1
25077914	2020-04-25	ffffd7744cebcf3aca44ae7049d2a94b87074c3d4ffe38...	866755002	0.050831	2
24375394	2020-04-09	ffffd7744cebcf3aca44ae7049d2a94b87074c3d4ffe38...	866755002	0.043203	2
24375395	2020-04-09	ffffd7744cebcf3aca44ae7049d2a94b87074c3d4ffe38...	840360003	0.013542	2

First, let's identify the most frequently purchased articles in recent transactions.

In [65]:
```
last_date = transactions_train_df.t_dat.max()
print(last_date)
print(transactions_train_df.loc[transactions_train_df.t_dat==last_date].shape)
```

```
2020-09-22
(32866, 5)
```

In [66]:
```
most_frequent_articles = list(transactions_train_df.loc[transactions_train_df.t_dat==last_date].article_id.value_counts
art_list = []
for art in most_frequent_articles:
    art = "0"+str(art)
    art_list.append(art)
art_str = " ".join(art_list)
print("Frequent articles bought recently: ", art_str)
```

```
Frequent articles bought recently:  0924243002 0751471001 0448509014 0918522001 0866731001 0714790020 0788575004 091552
9005 0573085028 0918292001 0850917001 0928206001
```

In [67]:
```
agg_df = transactions_train_df.groupby(["customer_id"])["article_id"].agg(lambda x: str(x.values[0:12])[1:-1]).reset_in
```

In [68]:
```
def padding_articles(x):
    if x:
        xl = x.split()
        x = []
        for xi in xl:
            x.append("0"+xi)
        dimm_x = len(x)
        if dimm_x < 12:
            x.extend(art_list[:12-dimm_x])
        return(" ".join(x))
```

In [69]:
```
agg_df["article_id"] = agg_df["article_id"].apply(lambda x: padding_articles(x))
```

In [70]:
```
print("Aggregated transaction history: ", agg_df.customer_id.nunique())
print("Submission sample: ", sample_submission_df.customer_id.nunique())
```

```
Aggregated transaction history:  1362281
Submission sample:  1371980
```

In [71]:
```
print(sample_result_df.shape)
sample_result_df.head()
```

```
(1371980, 2)
```

Out[71]:

	customer_id	prediction
0	00000dbacae5abe5e23885899a1fa44253a17956c6d1c3...	0706016001 0706016002 0372860001 0610776002 07...
1	0000423b00ade91418cceaf3b26c6af3dd342b51fd051e...	0706016001 0706016002 0372860001 0610776002 07...
2	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	0706016001 0706016002 0372860001 0610776002 07...
3	00005ca1c9ed5f5146b52ac8639a40ca9d57aeff4d1bd2...	0706016001 0706016002 0372860001 0610776002 07...
4	00006413d8573cd20ed7128e53b7b13819fe5cfc2d801f...	0706016001 0706016002 0372860001 0610776002 07...

For the customers with missing articles, we simply replace with most frequent buyed articles in most recent day(s).

In [72]:
```
submission_df = agg_df.merge(sample_submission_df[["customer_id"]], how="right")
submission_df.columns = ["customer_id", "prediction"]
```

```
print(submission_df.shape)
submission_df.head()
```

```
(1371980, 2)
```

Out[72]:

	customer_id	prediction
0	00000dbacae5abe5e23885899a1fa44253a17956c6d1c3...	0568601043 0841260003 0887593002 0890498002 07...
1	0000423b00ade91418cceaf3b26c6af3dd342b51fd051e...	0826211002 0599580055 0599580055 0811835004 08...
2	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	0794321007 0858883002 0851400006 0750424014 07...
3	00005ca1c9ed5f5146b52ac8639a40ca9d57aeff4d1bd2...	0742079001 0732413001 0924243002 0751471001 04...
4	00006413d8573cd20ed7128e53b7b13819fe5cfc2d801f...	0896152002 0730683050 0927530004 0791587015 05...

In [73]:
```
print("Rows with missing data in submission: ", submission_df.loc[submission_df.prediction.isna()].shape[0])
```

```
Rows with missing data in submission:  9699
```

We replace the missing data with the most frequently bought articles, from recent days.

In [74]:
```
submission_df.loc[submission_df.prediction.isna(), ["prediction"]] = art_str
```

In [75]:
```
print("Rows with missing data in submission: ", submission_df.loc[submission_df.prediction.isna()].shape[0])
```

```
Rows with missing data in submission:  0
```

In [76]:
```
submission_df.to_csv("submission.csv", index=False)
```