ALY6040_Week1_EDA_Group6

July 20, 2022

1 H&M Recommender System - Exploratory Data Analysis

1.1 ALY6040 - Data Mining

1.2 Group 6 - Harkirat Singh, Harshit Gaur, Puneet Madan, Akash Raj

1.3 Introduction

The H&M Group is a collection of brands and companies with about 4,850 physical locations and 53 online marketplaces. Customers can browse a wide assortment of products in our online store. However, if there are too many options, clients could not find what they are looking for or what intrigues them right away, which could prevent them from making a purchase. Product recommendations are essential for improving the buying experience. More importantly, assisting consumers in making sound decisions benefits sustainability since it lowers returns and, as a result, lowers transportation-related emissions. We are going to develop product recommendations based on data from previous transactions, as well as from customer and product meta data.

1.4 About the dataset

The dataset contains the following files.

- images/ - a folder of images corresponding to each article_id. The images are placed in subfolders starting with the first three digits of the article_id.
- articles.csv - CSV file containing the detailed metadata for each article_id available for purchase.
- customers.csv - CSV file containing the metadata for each customer_id in dataset.
- transactions_train.csv - CSV file containing the transactions data. It consists of the purchases of each customer for each date, as well as additional information. Duplicate rows correspond to multiple purchases of the same item.

We are going to start off our analysis by exploring the data, understanding the meaning and significance of the attributes involved properly, and perform necessary actions to streamline our process for the road to our goal.

1.4.1 Importing data from kaggle

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import json
```

```python
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', 40)
```

[ ]:
```python
# Label Encoding and One-Hot Encoding Libraries
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

# Standardization and Normalization Libraries
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

# SciPy.Stats for Plotting
import scipy.stats as stats
import pylab
```

[ ]:
```
!mkdir ~/.kaggle
```

[ ]:
```python
json_string = {"username":"mrnerd","key":"e90faefa3b2a5f6183c87004e6f7dd56"}
with open('/root/.kaggle/kaggle.json', 'w', encoding='utf-8') as f:
    json.dump(json_string, f, ensure_ascii=False, indent=4)
```

[ ]:
```
!chmod 600 /root/.kaggle/kaggle.json
```

1.4.2 Downloading and Unzipping the whole dataset (if not using images, then use another script after this)

[ ]:
```
#! kaggle competitions download -c h-and-m-personalized-fashion-recommendations
```

```
Downloading h-and-m-personalized-fashion-recommendations.zip to /content
100% 28.7G/28.7G [03:56<00:00, 149MB/s]
100% 28.7G/28.7G [03:56<00:00, 130MB/s]
```

[ ]:
```
#! unzip /content/h-and-m-personalized-fashion-recommendations.zip
```

1.4.3 Downloading and Unzipping specific files from Kaggle Dataset

[ ]:
```
! kaggle competitions download -c h-and-m-personalized-fashion-recommendations -f articles.csv
! kaggle competitions download -c h-and-m-personalized-fashion-recommendations -f customers.csv
! kaggle competitions download -c h-and-m-personalized-fashion-recommendations -f sample_submission.csv
! kaggle competitions download -c h-and-m-personalized-fashion-recommendations -f transactions_train.csv
```

```
Downloading articles.csv.zip to /content
  0% 0.00/4.26M [00:00<?, ?B/s]
100% 4.26M/4.26M [00:00<00:00, 97.2MB/s]
```

```
Downloading customers.csv.zip to /content
 79% 77.0M/97.9M [00:00<00:00, 90.9MB/s]
100% 97.9M/97.9M [00:00<00:00, 117MB/s]
Downloading sample_submission.csv.zip to /content
 87% 44.0M/50.3M [00:00<00:00, 169MB/s]
100% 50.3M/50.3M [00:00<00:00, 154MB/s]
Downloading transactions_train.csv.zip to /content
 96% 562M/584M [00:05<00:00, 98.0MB/s]
100% 584M/584M [00:05<00:00, 118MB/s]
```

[ ]:
```
! unzip /content/articles.csv.zip
! unzip /content/customers.csv.zip
! unzip /content/sample_submission.csv.zip
! unzip /content/transactions_train.csv.zip
```

```
Archive:  /content/articles.csv.zip
  inflating: articles.csv
Archive:  /content/customers.csv.zip
  inflating: customers.csv
Archive:  /content/sample_submission.csv.zip
  inflating: sample_submission.csv
Archive:  /content/transactions_train.csv.zip
  inflating: transactions_train.csv
```

[ ]:
```
articles = pd.read_csv('/content/articles.csv')
```

[ ]:
```
articles.head(3)
```

[ ]:
```
   article_id  product_code      prod_name  product_type_no product_type_name  \
0   108775015        108775      Strap top              253          Vest top
1   108775044        108775      Strap top              253          Vest top
2   108775051        108775  Strap top (1)              253          Vest top

   product_group_name  graphical_appearance_no graphical_appearance_name  \
0  Garment Upper body                  1010016                     Solid
1  Garment Upper body                  1010016                     Solid
2  Garment Upper body                  1010017                    Stripe

   colour_group_code colour_group_name  perceived_colour_value_id  \
0                  9             Black                          4
1                 10             White                          3
2                 11         Off White                          1

  perceived_colour_value_name  perceived_colour_master_id  \
0                        Dark                           5
1                       Light                           9
2                Dusty Light                           9
```

```
  perceived_colour_master_name  department_no department_name index_code  \
0                        Black           1676    Jersey Basic          A
1                        White           1676    Jersey Basic          A
2                        White           1676    Jersey Basic          A

   index_name  index_group_no index_group_name  section_no  \
0  Ladieswear               1       Ladieswear          16
1  Ladieswear               1       Ladieswear          16
2  Ladieswear               1       Ladieswear          16

            section_name  garment_group_no garment_group_name  \
0  Womens Everyday Basics              1002       Jersey Basic
1  Womens Everyday Basics              1002       Jersey Basic
2  Womens Everyday Basics              1002       Jersey Basic

                             detail_desc
0  Jersey top with narrow shoulder straps.
1  Jersey top with narrow shoulder straps.
2  Jersey top with narrow shoulder straps.
```

```
Warning: Total number of columns (25) exceeds max_columns (20) limiting to first
(20) columns.
```

[ ]:
```python
customers = pd.read_csv('/content/customers.csv')
```

[ ]:
```python
customers.head(3)
```

[ ]:
```
                                         customer_id  FN  Active  \
0  00000dbacae5abe5e23885899a1fa44253a17956c6d1c3…  NaN     NaN
1  0000423b00ade91418cceaf3b26c6af3dd342b51fd051e…  NaN     NaN
2  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…  NaN     NaN

  club_member_status fashion_news_frequency   age  \
0             ACTIVE                   NONE  49.0
1             ACTIVE                   NONE  25.0
2             ACTIVE                   NONE  24.0

                                         postal_code
0  52043ee2162cf5aa7ee79974281641c6f11a68d276429a…
1  2973abc54daa8a5f8ccfe9362140c63247c5eee03f1d93…
2  64f17e6a330a85798e4998f62d0930d14db8db1c054af6…
```

[ ]:
```python
transactions = pd.read_csv('/content/transactions_train.csv')
```

[ ]:
```python
transactions.head(3)
```

[ ]:
```
        t_dat                                        customer_id  article_id  \
0  2018-09-20  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…   663713001
```

```
1  2018-09-20  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…   541518023
2  2018-09-20  00007d2de826758b65a93dd24ce629ed66842531df6699…   505221004

      price  sales_channel_id
0  0.050831                 2
1  0.030492                 2
2  0.015237                 2
```

1.5 Data Analysis and Visualization

1.5.1 Columns and Shape of the dataset

- The articles file contains 105,542 data points with 25 features.
- The customers file contains 1,371,980 data points with 17 features.
- The transactions file contains 1,371,980 data points with 6 features.

[ ]:
```
print(f"Number of articles are {articles.shape[0]}")
print(f"Number of customers are {customers.shape[0]}")
print(f"Number of transaction are {transactions.shape[0]}")
```

```
Number of articles are 105542
Number of customers are 1371980
Number of transaction are 31788324
```

[ ]:
```
%ls
```

```
articles.csv      customers.csv.zip          transactions_train.csv
articles.csv.zip  sample_submission.csv      transactions_train.csv.zip
customers.csv     sample_submission.csv.zip
```

[ ]:
```
## Features in the articles data
articles.columns
```

[ ]:
```
Index(['article_id', 'product_code', 'prod_name', 'product_type_no',
       'product_type_name', 'product_group_name', 'graphical_appearance_no',
       'graphical_appearance_name', 'colour_group_code', 'colour_group_name',
       'perceived_colour_value_id', 'perceived_colour_value_name',
       'perceived_colour_master_id', 'perceived_colour_master_name',
       'department_no', 'department_name', 'index_code', 'index_name',
       'index_group_no', 'index_group_name', 'section_no', 'section_name',
       'garment_group_no', 'garment_group_name', 'detail_desc'],
      dtype='object')
```

[ ]:
```
## Features in the customers data
customers.columns
```

[ ]:
```
Index(['customer_id', 'FN', 'Active', 'club_member_status',
       'fashion_news_frequency', 'age', 'postal_code'],
```

```
      dtype='object')
```

```
[ ]: ## Features in the transaction data
transactions.columns
```

```
[ ]: Index(['t_dat', 'customer_id', 'article_id', 'price', 'sales_channel_id',
       'year', 'month', 'day', 'product_type_name', 'age', 'age_bucket',
       'Season'],
      dtype='object')
```

1.5.2 Identifying missing values

It's important to indentify the missing values in the dataset, so that appropriate missing value imputation can be applied. The following function identifies the total number of missing values in all the columns of the dataset.?

```
[ ]: # Function to identify the missing values

def find_missing_features(df):
  return dict([(feature,df[feature].isnull().sum()) for feature in df.columns␣
 ↪if df[feature].isnull().sum() >=1])
```

```
[ ]: find_missing_features(articles)
```

```
[ ]: {'detail_desc': 416}
```

```
[ ]: find_missing_features(customers)
```

```
[ ]: {'Active': 907576,
 'FN': 895050,
 'age': 15861,
 'club_member_status': 6062,
 'fashion_news_frequency': 16009}
```

```
[ ]: find_missing_features(transactions)
```

```
[ ]: {}
```

- The 'detail_desc' column in the articles dataset contain 416 missing values.
- There are some missing values in the 'Active', 'FN', 'age', 'club_member_status', 'fashion_news_frequency' columns in the customers dataset contain 416 missing records.
- There are no missing values in the transaction table.

1.5.3 Outlier Detection

For outlier detection, we used box plot to visually explore the feature and created a table with different quantile values of the attribute. As we can see in the box plot and the quantile values in the table, there are several outliers in the dataset for '*price*' feature which is right skewed in distribution as well.

Indetifying the numeric and non-numeric columns

```
[ ]: t_cols = transactions.dtypes[transactions.dtypes.isin([int, float])]
     t_num_cols = list(t_cols.index)
     t_num_cols
```

```
[ ]: ['article_id', 'price', 'sales_channel_id', 'year', 'month', 'day', 'age']
```

```
[ ]: a_cols = articles.dtypes[articles.dtypes.isin([int, float])]
     a_num_cols = list(a_cols.index)
     a_num_cols
```

```
[ ]: ['article_id',
      'product_code',
      'product_type_no',
      'graphical_appearance_no',
      'colour_group_code',
      'perceived_colour_value_id',
      'perceived_colour_master_id',
      'department_no',
      'index_group_no',
      'section_no',
      'garment_group_no']
```

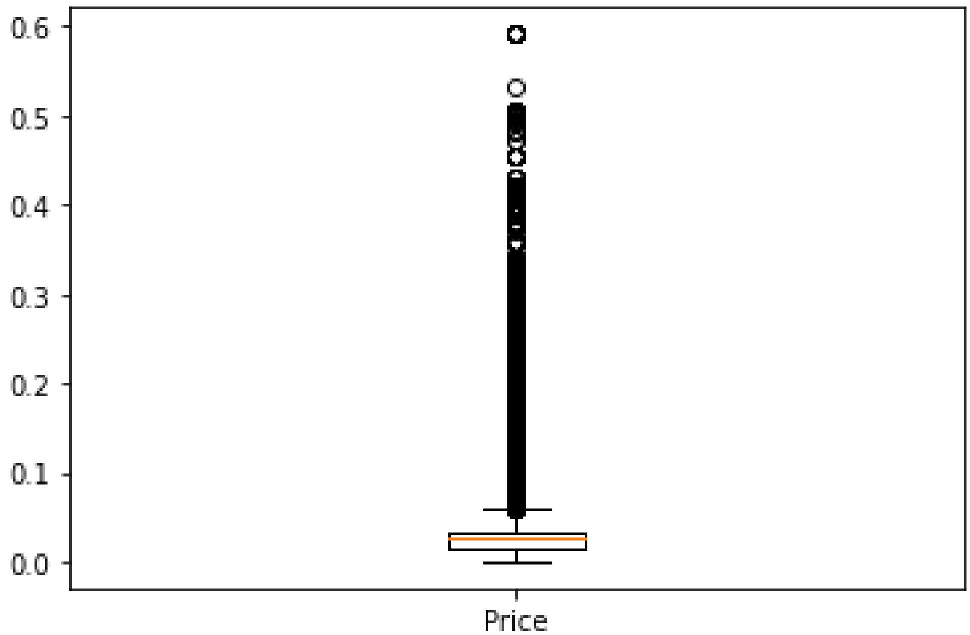
```
[ ]: c_cols = customers.dtypes[customers.dtypes.isin([int, float])]
     c_num_cols = list(c_cols.index)
     c_num_cols
```

```
[ ]: ['FN', 'Active', 'age']
```

Boxplot to visualize outliers

```
[ ]: plt.boxplot(transactions['price'])
     plt.xticks([1], ["Price"])
```

```
[ ]: ([<matplotlib.axis.XTick at 0x7f48b2494710>], [Text(0, 0, 'Price')])
```



Obtaining the percentile values

```
[ ]: def quantile_out(df, cols):
       perc = pd.DataFrame()
       perc = perc.append({'column' : cols, 'min val': min(df[cols]), 'perc 5th' :␣
      ↪float(df[cols].quantile(0.05)), 'perc 95th' : float(df[cols].quantile(0.
      ↪95)), 'max val': max(df[cols])},
             ignore_index = True)
       return perc
```

```
[ ]: quantile_out(transactions, 'price')
```

[ ]:

	column	min val	perc 5th	perc 95th	max val
0	price	0.000017	0.00761	0.059305	0.591525

Observation: The 95th percentile value of 'Price' is 0.059, whereas the max value is 0.59. This indicates presence of some outliers, or high price items.

```
[ ]: quantile_out(transactions, 'age')
```

[ ]:

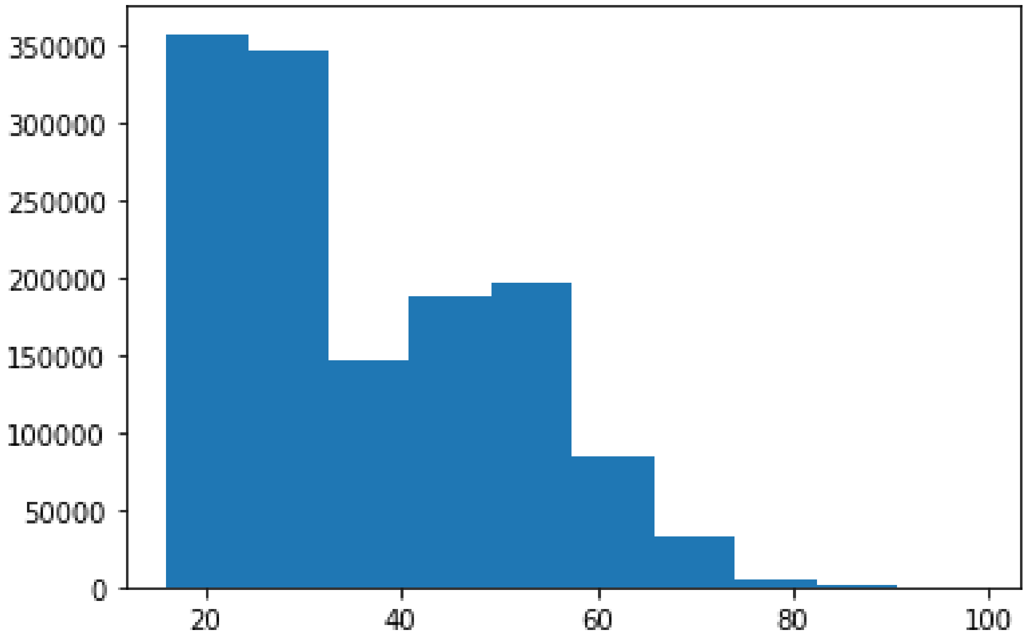
	column	min val	perc 5th	perc 95th	max val
0	age	16.0	21.0	59.0	99.0

Observation: There are no outliers in the 'age' variable.

Histogram of age variable

```
[ ]: plt.hist(customers['age'])
```

```
[ ]: (array([3.57169e+05, 3.46715e+05, 1.46283e+05, 1.87960e+05, 1.96469e+05,
            8.39560e+04, 3.15830e+04, 5.38800e+03, 5.19000e+02, 7.70000e+01]),
      array([16. , 24.3, 32.6, 40.9, 49.2, 57.5, 65.8, 74.1, 82.4, 90.7, 99. ]),
      <a list of 10 Patch objects>)
```



1.5.4 Exploratory Data Analysis of Columns

Most of the columns in our dataset are categorical variables. Therefore, we will need to identify the cardinality and the frequency of each categories in the column.

The following function creates a table, barplot, and pie chart to infer the cardinality and frequency.

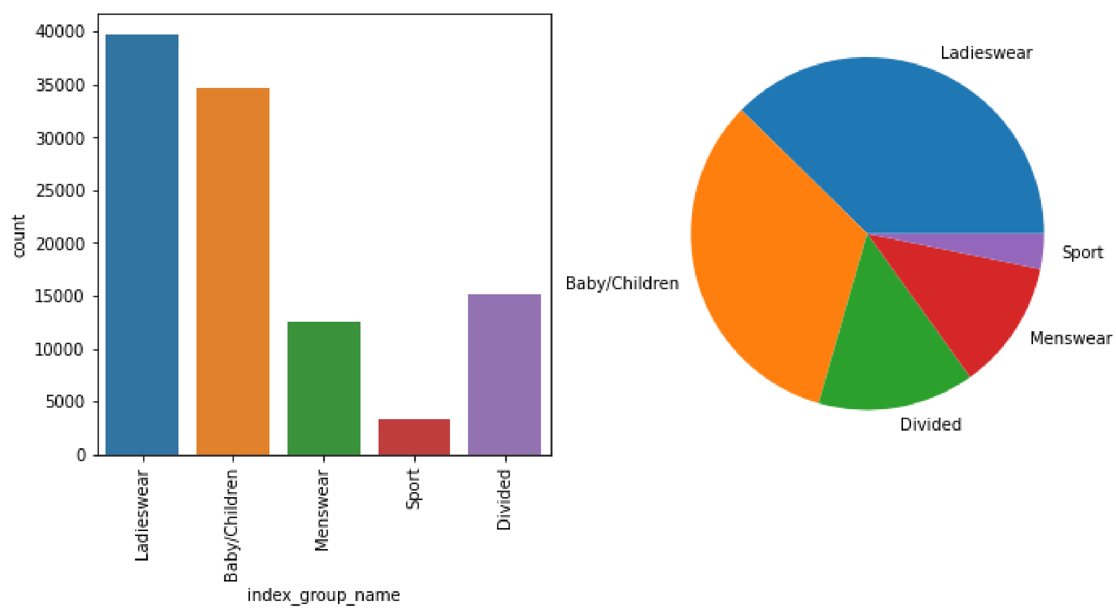
```
[ ]: def plot_data(data, column):
         """
         Function that takes in dataset and column as input and plots
         frequency bar chart and pie chart for the same
         """
         counts = dict(data[column].value_counts())
         print(data[column].value_counts())
         print()

         figure = plt.figure(figsize=(11,5))

         # bar chart
         plt.subplot(1,2,1)
         ax = sns.countplot(x = data[column])
```

```
    ax.set_xticklabels(ax.get_xticklabels(),rotation = 90)

    # pie chart
    plt.subplot(1,2,2)
    plt.pie(x=counts.values(), labels=counts.keys())

    plt.show()
```

Exploring different columns of articles

[ ]:
```
plot_data(data=articles, column='index_group_name')
```

```
Ladieswear       39737
Baby/Children    34711
Divided          15149
Menswear         12553
Sport             3392
Name: index_group_name, dtype: int64
```

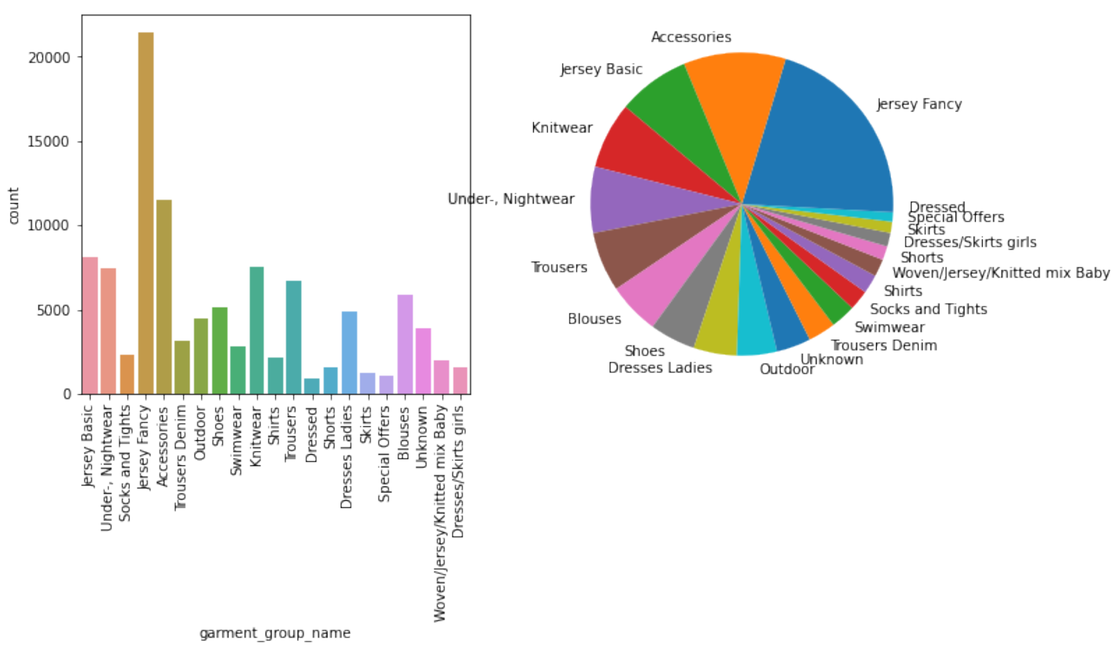


Observation: We see that the 'Ladieswear' and 'Baby/Children' dominate in the index groups.

[ ]:
```
plot_data(data=articles, column='garment_group_name')
```

```
Jersey Fancy                   21445
Accessories                    11519
Jersey Basic                    8126
Knitwear                        7490
Under-, Nightwear               7441
```

```
Trousers                         6727
Blouses                          5838
Shoes                            5145
Dresses Ladies                   4874
Outdoor                          4501
Unknown                          3873
Trousers Denim                   3100
Swimwear                         2787
Socks and Tights                 2272
Shirts                           2116
Woven/Jersey/Knitted mix Baby    1965
Shorts                           1559
Dresses/Skirts girls             1541
Skirts                           1254
Special Offers                   1061
Dressed                           908
Name: garment_group_name, dtype: int64
```

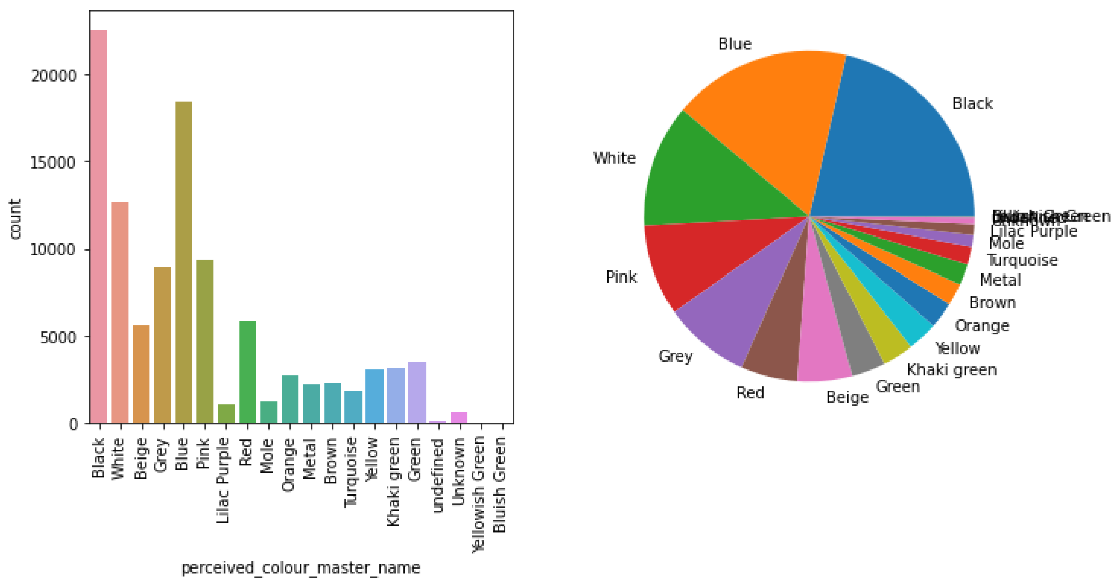


Observation: We see that maximum number of article belong to 'Jersy Fancy' garment group.

[ ]:
```
plot_data(data=articles, column='perceived_colour_master_name')
```

```
Black            22585
Blue             18469
White            12665
Pink              9403
```

```
Grey                    8924
Red                     5878
Beige                   5657
Green                   3526
Khaki green             3181
Yellow                  3121
Orange                  2734
Brown                   2269
Metal                   2180
Turquoise               1829
Mole                    1223
Lilac Purple            1100
Unknown                  685
undefined                105
Yellowish Green            5
Bluish Green               3
Name: perceived_colour_master_name, dtype: int64
```

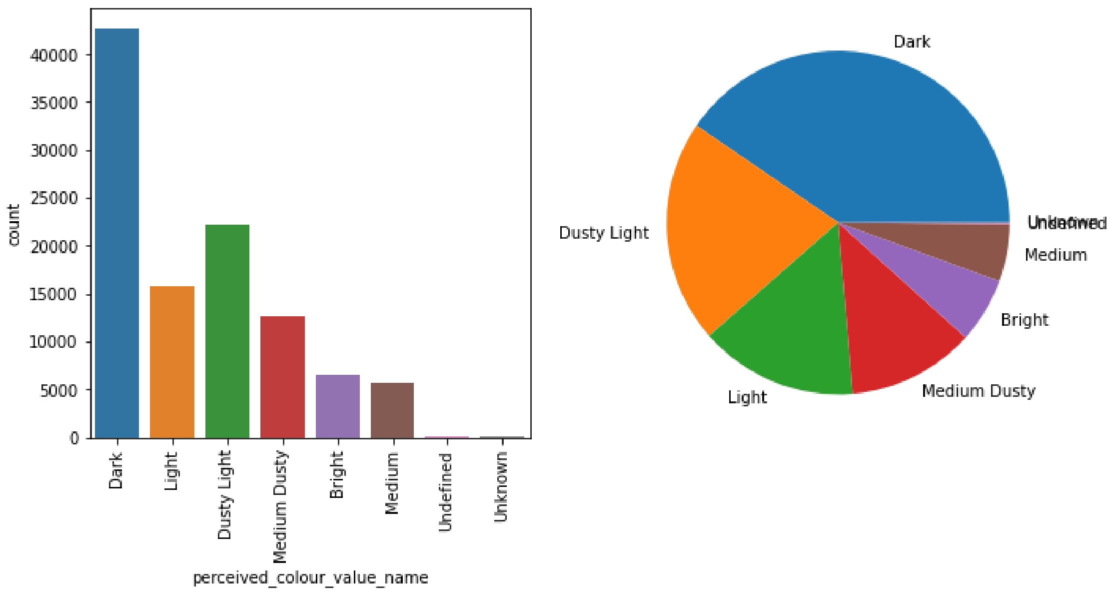


Observation: The majority of articles are in 'White', 'Blue' or 'Black' colour.

[ ]:
```
plot_data(data=articles, column='perceived_colour_value_name')
```

```
Dark             42706
Dusty Light      22152
Light            15739
Medium Dusty     12630
Bright            6471
Medium            5711
```

```
Undefined          105
Unknown             28
Name: perceived_colour_value_name, dtype: int64
```

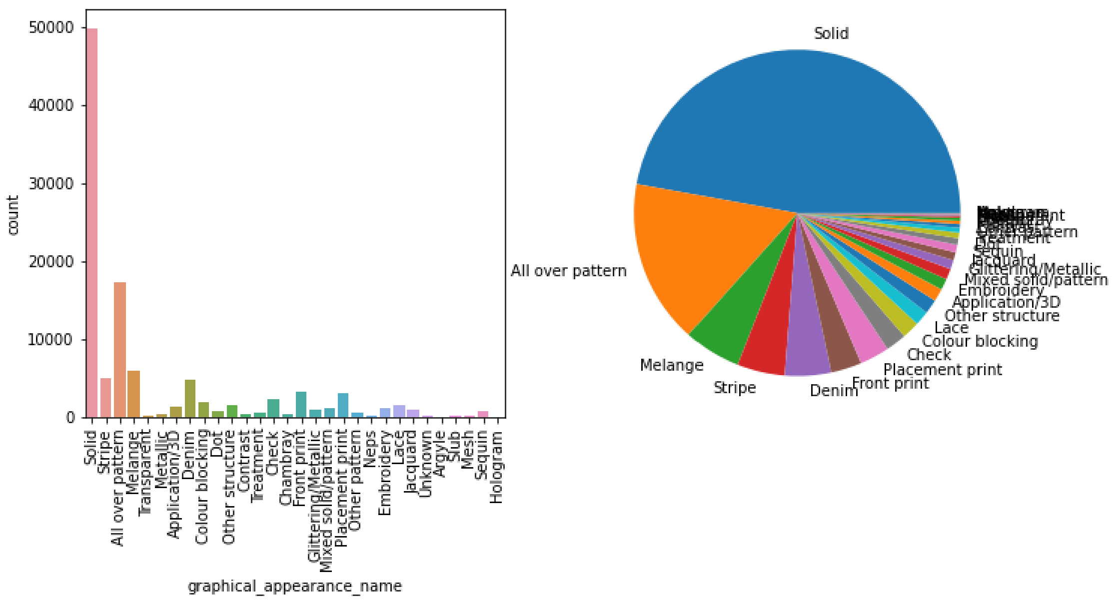


Observation: Majority of articles are 'Dusty Light' or 'Dark' in shade.

[ ]:
```
plot_data(data=articles, column='graphical_appearance_name')
```

```
Solid                    49747
All over pattern         17165
Melange                   5938
Stripe                    4990
Denim                     4842
Front print               3215
Placement print           3098
Check                     2178
Colour blocking           1830
Lace                      1513
Other structure           1502
Application/3D            1341
Embroidery                1165
Mixed solid/pattern       1132
Glittering/Metallic        958
Jacquard                   830
Sequin                     806
Dot                        681
Treatment                  586
Other pattern              515
```

```
Contrast                 376
Metallic                 346
Chambray                 322
Slub                     153
Transparent               86
Mesh                      86
Neps                      66
Unknown                   52
Argyle                    15
Hologram                   8
Name: graphical_appearance_name, dtype: int64
```

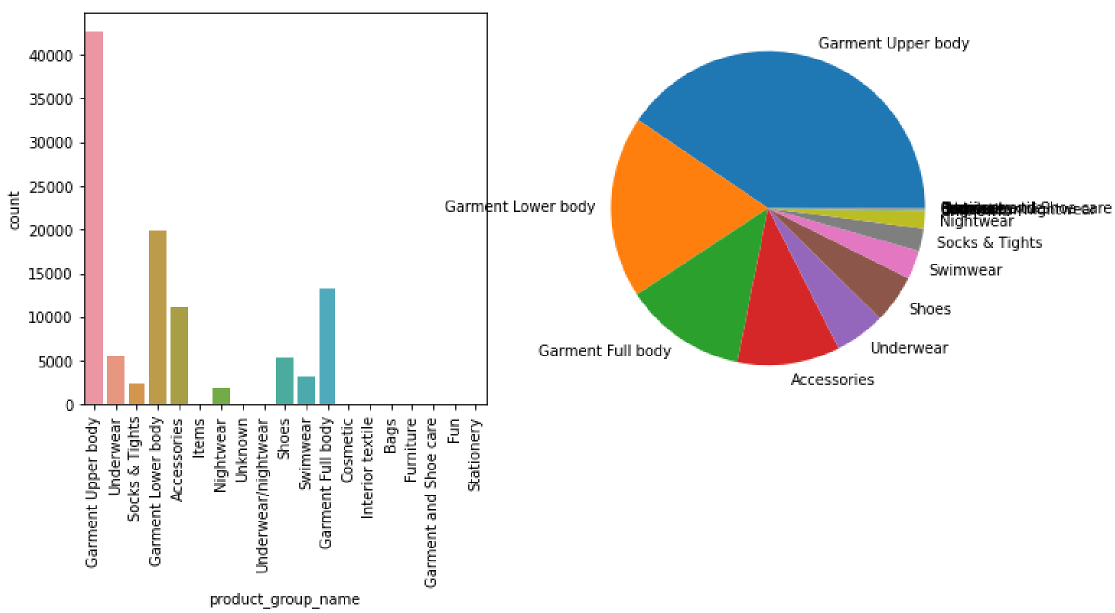


Observation: Majority of the articles have a 'Solid' graphical apperance.

[ ]:
```
plot_data(data=articles, column='product_group_name')
```

```
Garment Upper body       42741
Garment Lower body       19812
Garment Full body        13292
Accessories              11158
Underwear                 5490
Shoes                     5283
Swimwear                  3127
Socks & Tights            2442
Nightwear                 1899
Unknown                    121
Underwear/nightwear         54
```

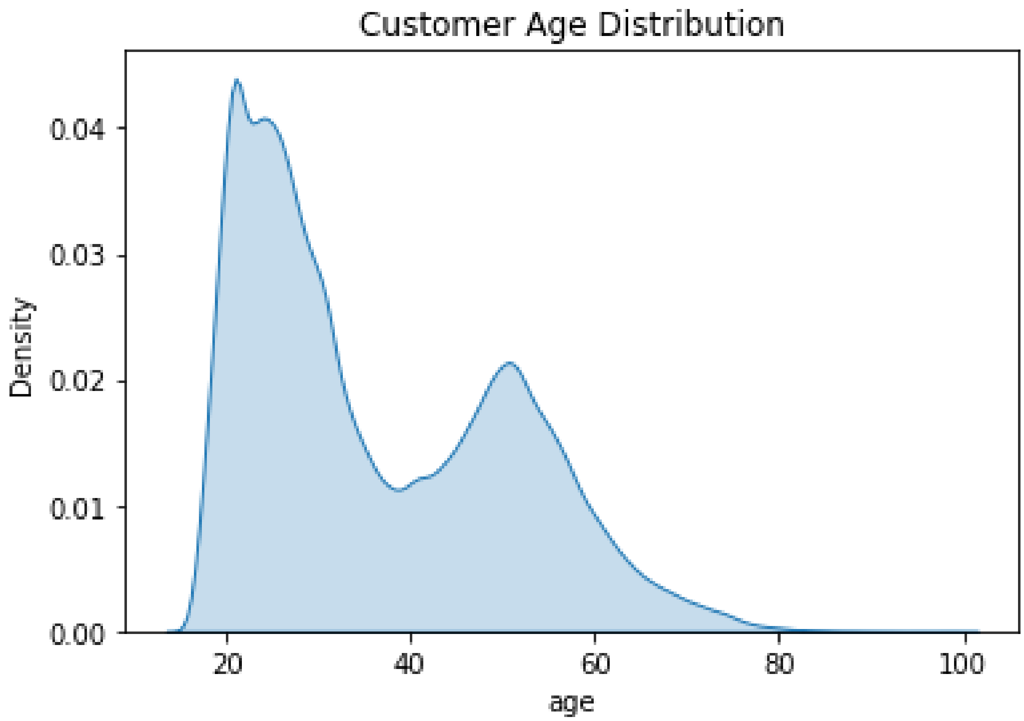
```
Cosmetic                    49
Bags                        25
Items                       17
Furniture                   13
Garment and Shoe care        9
Stationery                   5
Interior textile             3
Fun                          2
Name: product_group_name, dtype: int64
```



Observation: Most of the articles are of 'Garment' product group.

Exploring different columns of customer

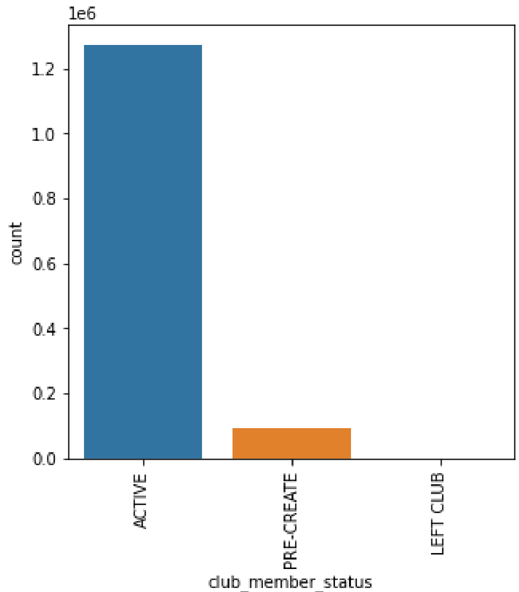
```
[ ]: sns.kdeplot(customers['age'], shade=True)
     plt.title('Customer Age Distribution')
     plt.show()
```

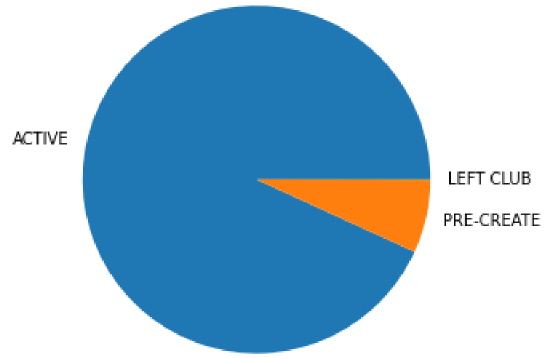


Observation: The Age of the customers is right skewed and the majority of customers are of the age 18-30 years old.

```
[ ]: plot_data(data=customers, column='club_member_status')
```

```
ACTIVE        1272491
PRE-CREATE      92960
LEFT CLUB         467
Name: club_member_status, dtype: int64
```



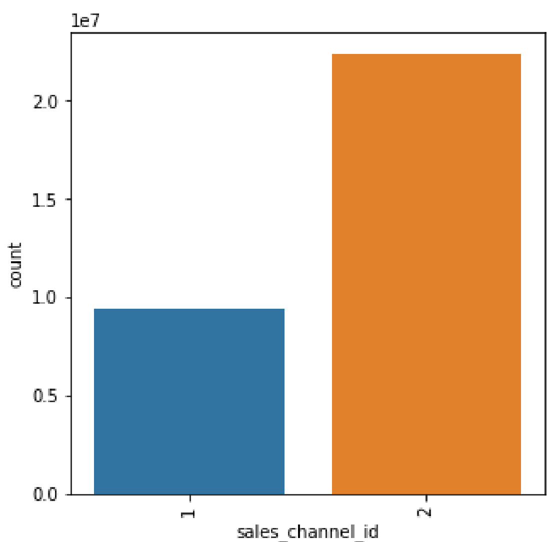


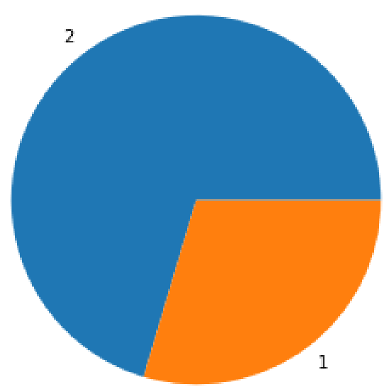
Observation: The majority of customers are an 'ACTIVE' club member

Exploring different columns of transaction

[ ]:
```
plot_data(data=transactions, column='sales_channel_id')
```

```
2    22379862
1     9408462
Name: sales_channel_id, dtype: int64
```





Observation: Majority of of sales are from channel 2.

1.6 Feature Engineering

[ ]:
```
transactions["t_dat"] = pd.to_datetime(transactions["t_dat"])

# add year, month, day
transactions["year"] = transactions["t_dat"].dt.year
transactions["month"] = transactions["t_dat"].dt.month
transactions["day"] = transactions["t_dat"].dt.day
```

[ ]:
```
transactions = pd.merge(transactions, articles[["article_id",␣
↪"product_type_name"]],
                        on="article_id")
```

[ ]:
```
transactions = pd.merge(transactions, customers[["customer_id", "age"]],
                        on="customer_id")
display(transactions.head())
```

```
       t_dat                                       customer_id  article_id  \
0 2018-09-20  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…   663713001
1 2018-09-24  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…   663713001
2 2018-09-20  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…   541518023
3 2019-03-01  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…   578020002
4 2020-02-03  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca…   351484002

      price  sales_channel_id  year  month  day product_type_name   age
0  0.050831                 2  2018      9   20     Underwear body  24.0
1  0.050831                 2  2018      9   24     Underwear body  24.0
2  0.030492                 2  2018      9   20                Bra  24.0
3  0.013542                 2  2019      3    1             Blouse  24.0
4  0.022017                 2  2020      2    3    Swimwear bottom  24.0
```

[ ]:
```
bins = [i for i in range(10, 101, 10)]
labels = [i for i in range(1, len(bins))]

transactions["age_bucket"] = pd.cut(transactions["age"], bins=bins,␣
↪labels=labels)
```

[ ]:
```
customer_prod_count = transactions.groupby(["age_bucket"])["product_type_name"].
↪value_counts()
```

[ ]:
```
customer_prod = pd.DataFrame(
    index=np.sort(np.array(transactions["age_bucket"].unique().dropna())),
    columns=articles["product_type_name"].unique()
)

customer_prod = customer_prod.fillna(0)
```

```python
# count product data
for age_bucket in customer_prod_count.index.get_level_values("age_bucket").
↪unique():
    for prod in customer_prod_count.loc[age_bucket].index.
↪get_level_values("product_type_name"):
        customer_prod.loc[age_bucket, prod] = customer_prod_count.
↪loc[age_bucket].loc[prod]
```

[ ]:
```python
customer_prod
```

[ ]:

	Vest top	Bra	Underwear Tights	Socks	Leggings/Tights	Sweater	\
1	69450	82721	5316	21906	24994	124372	
2	621400	644575	90789	196438	306898	1127013	
3	280552	240054	48179	85684	157863	455679	
4	219879	185956	28739	88739	125154	472188	
5	177485	152281	22331	74527	94291	452598	
6	34406	21292	4648	11809	18344	113950	
7	4357	2194	585	1349	2730	22522	
8	343	191	30	110	210	1612	
9	49	43	9	43	23	131	

	Top	Trousers	Hair clip	Umbrella	Pyjama jumpsuit/playsuit	Bodysuit	\
1	71595	167941	1924	205	210	5510	
2	657560	1620163	17561	2158	4326	71332	
3	283205	740297	8151	759	4208	33146	
4	247712	762682	6052	773	2226	13621	
5	244616	704037	5120	873	1157	11345	
6	60289	171140	918	220	407	2471	
7	10683	30180	137	43	51	325	
8	634	1818	7	2	3	17	
9	77	172	0	0	0	3	

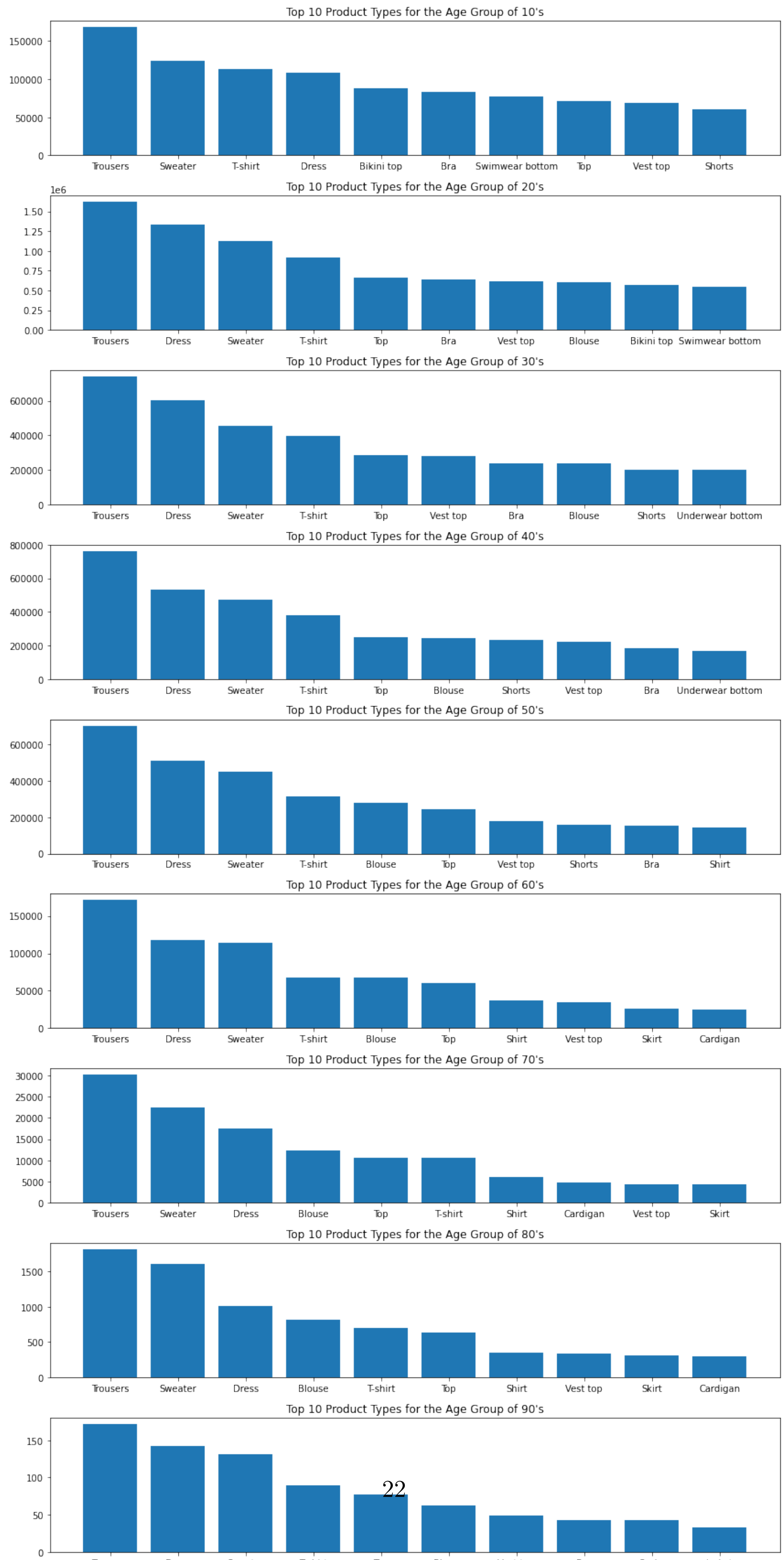
	Hair string	Unknown	Hoodie	…	Cross-body bag	Moccasins	Towel	\
1	2613	4780	37037	…	110	0	1	
2	19958	41032	185979	…	406	15	7	
3	7678	18603	68591	…	112	16	10	
4	7576	16496	101364	…	141	6	1	
5	5185	13431	74196	…	109	2	1	
6	754	2061	11588	…	21	2	0	
7	121	171	1739	…	6	0	0	
8	9	12	178	…	0	0	0	
9	2	3	30	…	0	0	0	

	Wood balls	Zipper head	Mobile case	Pre-walkers	Toy	Marker pen	Bumbag	\
1	0	0	236	0	0	35	1	
2	5	12	918	0	2	91	8	

```
3           10            16            201              0     1           50        2
4           11            13            273              0     2           21        4
5            8            17            184              0     0           28        0
6            1             6             15              0     0            0        0
7            0             0              3              0     0            0        1
8            0             0              0              0     0            0        0
9            0             0              0              0     0            0        0

   Dog wear  Eyeglasses  Wireless earphone case  Stain remover spray  \
1         5           4                      62                    0
2       105           9                     116                   18
3        64           5                      19                    5
4        39           5                      45                    5
5        42           1                      22                    5
6         9           1                       0                    3
7         0           0                       0                    0
8         0           0                       0                    0
9         0           0                       0                    0

   Clothing mist
1              0
2              0
3              1
4              3
5              0
6              0
7              0
8              0
9              0

[9 rows x 131 columns]
```

[ ]:
```python
f, ax = plt.subplots(nrows=9, ncols=1, figsize=(12, 24))
ax = ax.flatten()

top_products = []

for i, bin_age in enumerate(customer_prod.index):
    tmp_df = customer_prod.loc[bin_age]

    # descending sort
    indices = tmp_df.values.argsort()[::-1]

    # extract top 10
    columns = customer_prod.columns[indices][:10]
    top_products += columns.tolist()
    ax[i].bar(columns, tmp_df[columns])
```

```
    ax[i].set_xticklabels(columns)
    ax[i].set_title(f"Top 10 Product Types for the Age Group of␣
↪{int(bin_age)*10}'s")

top_products = set(top_products)
plt.tight_layout()
plt.show()
```


Top 10 Product Types for the Age Group of 10's
Trousers Sweater T-shirt Dress Bikini top Bra Swimwear bottom Top Vest top Shorts
Top 10 Product Types for the Age Group of 20's
1e6
Trousers Dress Sweater T-shirt Top Bra Vest top Blouse Bikini top Swimwear bottom
Top 10 Product Types for the Age Group of 30's
Trousers Dress Sweater T-shirt Top Vest top Bra Blouse Shorts Underwear bottom
Top 10 Product Types for the Age Group of 40's
Trousers Dress Sweater T-shirt Top Blouse Shorts Vest top Bra Underwear bottom
Top 10 Product Types for the Age Group of 50's
Trousers Dress Sweater T-shirt Blouse Top Vest top Shorts Bra Shirt
Top 10 Product Types for the Age Group of 60's
Trousers Dress Sweater T-shirt Blouse Top Shirt Vest top Skirt Cardigan
Top 10 Product Types for the Age Group of 70's
Trousers Sweater Dress Blouse Top T-shirt Shirt Cardigan Vest top Skirt
Top 10 Product Types for the Age Group of 80's
Trousers Sweater Dress Blouse T-shirt Top Shirt Vest top Skirt Cardigan
Top 10 Product Types for the Age Group of 90's
Trousers Dress Sweater T-shirt Top Blouse Vest top Bra Socks Jacket

```
[ ]: transactions['Season'] = np.where(transactions['month'].isin([3,4,5]),␣
     ↪"Summer", "Others")
```

1.7 Encoding and Scaling

Exploring the 'Transactions' table

```
[ ]: transactions.info()
     transactions['sales_channel_id'].value_counts()
```

```
<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
```

```
[ ]: 2    22379862
     1     9408462
     Name: sales_channel_id, dtype: int64
```

Exploring 'Customers' To explore the distribution of the features grouped together in order to grasp any tangible information about these features.

```
[ ]: customers[['Active', 'club_member_status', 'fashion_news_frequency']].
     ↪value_counts()
```

```
[ ]: Active  club_member_status  fashion_news_frequency
     1.0     ACTIVE              Regularly                 457229
             PRE-CREATE          Regularly                   5567
             ACTIVE              Monthly                      735
                                 NONE                         488
             PRE-CREATE          Monthly                       58
                                 NONE                           6
             LEFT CLUB           Regularly                      3
     dtype: int64
```

1.7.1 Replacing 'None' with 'NONE' in the *fashion_news_frequency* attribute.

We found a data point with value 'None' in the '*fashion_news_frequency*' feature. To congregate this data point with other 'NONE' values of the same feature, we are replacing it with 'NONE'.

```
[ ]: customers.loc[customers['fashion_news_frequency'] == 'None',␣
      ↪'fashion_news_frequency'] = 'NONE'
```

To explore the distribution of the features grouped together in order to grasp any tangible information about these features.

```
[ ]: customers[['club_member_status', 'fashion_news_frequency']].value_counts()
```

```
[ ]: club_member_status  fashion_news_frequency
     ACTIVE              NONE                      788484
                         Regularly                 471304
     PRE-CREATE          NONE                       85065
                         Regularly                   5787
     ACTIVE              Monthly                      778
     LEFT CLUB           NONE                         459
     PRE-CREATE          Monthly                       59
     LEFT CLUB           Regularly                      8
     dtype: int64
```

1.7.2 One Hot Encoding

Encoding '*Index Code*' in Articles table There are no ordinal variables in the data set since all the categorical variables in this file do not have any ranking/ordering in between themselves respectively.

Therefore, we are going to employ One Hot Encoding to encode the '*Index Code*' feature containing around 10 index codes.

```
[ ]: # Defining one-hot encoder object
     encoder = OneHotEncoder(sparse = True)

     # Performing the encoding
     articles_new = pd.DataFrame(encoder.fit_transform(articles[['index_code']]).
      ↪toarray())
     articles_new.columns = encoder.get_feature_names_out(['index_code'])

     articles = articles.join(articles_new)
```

To check the successful encoding action on the feature in the data set.

```
[ ]: articles.head(3)
```

```
[ ]:    article_id  product_code      prod_name  product_type_no product_type_name  \
     0   108775015        108775      Strap top              253          Vest top
     1   108775044        108775      Strap top              253          Vest top
     2   108775051        108775  Strap top (1)              253          Vest top

        product_group_name  graphical_appearance_no graphical_appearance_name  \
     0  Garment Upper body                  1010016                     Solid
```

```
1  Garment Upper body                    1010016                    Solid
2  Garment Upper body                    1010017                   Stripe

   colour_group_code colour_group_name  perceived_colour_value_id  \
0                  9             Black                          4
1                 10             White                          3
2                 11         Off White                          1

  perceived_colour_value_name  perceived_colour_master_id  \
0                        Dark                           5
1                       Light                           9
2                Dusty Light                           9

  perceived_colour_master_name  department_no  … section_no  \
0                        Black           1676  …         16
1                        White           1676  …         16
2                        White           1676  …         16

             section_name garment_group_no  garment_group_name  \
0  Womens Everyday Basics             1002       Jersey Basic
1  Womens Everyday Basics             1002       Jersey Basic
2  Womens Everyday Basics             1002       Jersey Basic

                              detail_desc  index_code_A index_code_B  \
0  Jersey top with narrow shoulder straps.           1.0          0.0
1  Jersey top with narrow shoulder straps.           1.0          0.0
2  Jersey top with narrow shoulder straps.           1.0          0.0

   index_code_C index_code_D index_code_F  index_code_G  index_code_H  \
0           0.0          0.0          0.0           0.0           0.0
1           0.0          0.0          0.0           0.0           0.0
2           0.0          0.0          0.0           0.0           0.0

   index_code_I  index_code_J  index_code_S
0           0.0           0.0           0.0
1           0.0           0.0           0.0
2           0.0           0.0           0.0

[3 rows x 35 columns]
```

Encoding '*Club Member Status*' and '*Fashion News Frequency*' variables in Customers table For the features mentioned above, we are going to employ One Hot Encoding to encode their nominal categorical features.

[ ]:
```python
# Defining one-hot encoder object
encoder = OneHotEncoder(sparse = True)
```

```
# Performing the encoding
customers_new = pd.DataFrame(encoder.
↪fit_transform(customers[['club_member_status', 'fashion_news_frequency']]).
↪toarray())
customers_new.columns = encoder.get_feature_names_out(['club_member_status',␣
↪'fashion_news_frequency'])

customers = customers.join(customers_new)
```

To check the successful encoding action on the feature in the data set.

[ ]:
```
customers.head(3)
```

[ ]:
```
                                         customer_id  FN  Active  \
0  00000dbacae5abe5e23885899a1fa44253a17956c6d1c3… NaN     NaN
1  0000423b00ade91418cceaf3b26c6af3dd342b51fd051e… NaN     NaN
2  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca… NaN     NaN

  club_member_status fashion_news_frequency   age  \
0             ACTIVE                   NONE  49.0
1             ACTIVE                   NONE  25.0
2             ACTIVE                   NONE  24.0

                                         postal_code  \
0  52043ee2162cf5aa7ee79974281641c6f11a68d276429a…
1  2973abc54daa8a5f8ccfe9362140c63247c5eee03f1d93…
2  64f17e6a330a85798e4998f62d0930d14db8db1c054af6…

   club_member_status_ACTIVE  club_member_status_LEFT CLUB  \
0                        1.0                           0.0
1                        1.0                           0.0
2                        1.0                           0.0

   club_member_status_PRE-CREATE  club_member_status_nan  \
0                            0.0                     0.0
1                            0.0                     0.0
2                            0.0                     0.0

   fashion_news_frequency_Monthly  fashion_news_frequency_NONE  \
0                             0.0                          1.0
1                             0.0                          1.0
2                             0.0                          1.0

   fashion_news_frequency_Regularly  fashion_news_frequency_nan
0                               0.0                         0.0
1                               0.0                         0.0
2                               0.0                         0.0
```

1.7.3 Scaling - Normalization or Standardization?

Before proceeding further with the scaling process, we wanted to explore the data set for features which require scaling applied to them.

We also wanted to figure out which scaling techniques (Normlization or Standardization) to apply to the features based on their distribution and nature.

[ ]:
```
transactions.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
```

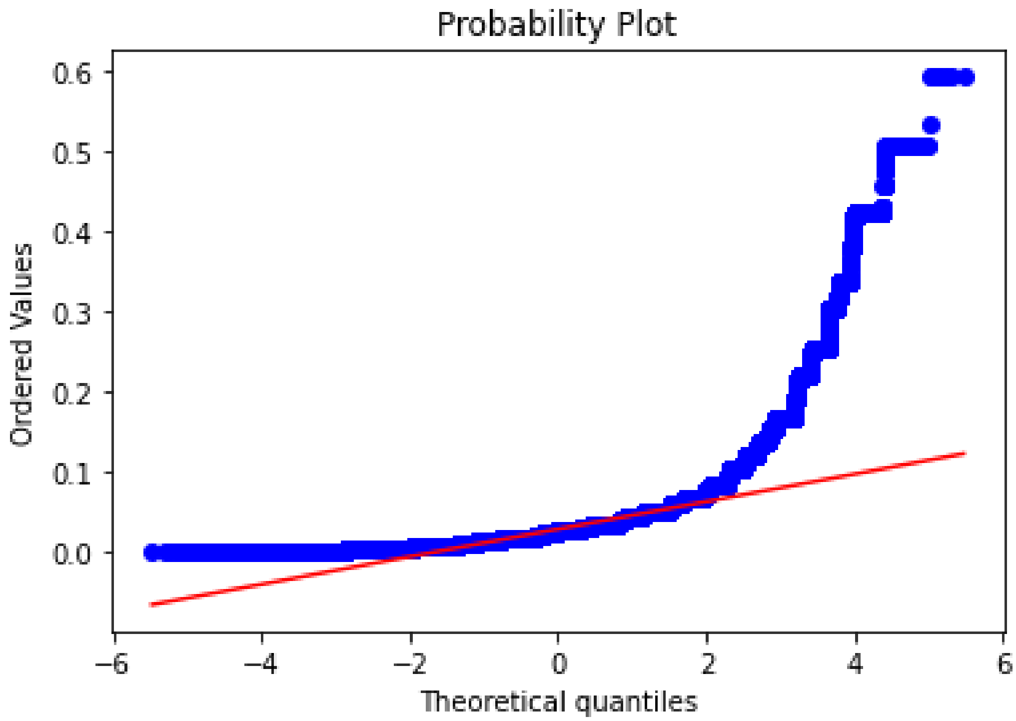
[ ]:
```
transactions.describe()
```

[ ]:

	article_id	price	sales_channel_id
count	3.178832e+07	3.178832e+07	3.178832e+07
mean	6.962272e+08	2.782927e-02	1.704028e+00
std	1.334480e+08	1.918113e-02	4.564786e-01
min	1.087750e+08	1.694915e-05	1.000000e+00
25%	6.328030e+08	1.581356e-02	1.000000e+00
50%	7.145820e+08	2.540678e-02	2.000000e+00
75%	7.865240e+08	3.388136e-02	2.000000e+00
max	9.562170e+08	5.915254e-01	2.000000e+00

To check if '*Price*' feature is normally distributed or not

[ ]:
```
stats.probplot(transactions['price'], dist="norm", plot=pylab)
pylab.show()
```



Since the data points are deviating significantly from the straight red-line, the '*price*' feature does not seem to be normally distributed

1.7.4 Normalization

We're going to apply Normalization scaling technique to the '*price*' feature of the transactions table in order to normalize it and scale up the data points within a range.

[ ]:
```
# Defining MinMax scaler object
scaler = MinMaxScaler()

# Performing the scaling
transactions['price_scaled'] = pd.DataFrame(scaler.
 ↪fit_transform(transactions[['price']]))
```

[ ]:
```
transactions.head(5)
```

[ ]:
```
        t_dat                                       customer_id  article_id  \
0  2018-09-20  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...   663713001
1  2018-09-20  000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...   541518023
2  2018-09-20  00007d2de826758b65a93dd24ce629ed66842531df6699...   505221004
3  2018-09-20  00007d2de826758b65a93dd24ce629ed66842531df6699...   685687003
4  2018-09-20  00007d2de826758b65a93dd24ce629ed66842531df6699...   685687004
```

	price	sales_channel_id	price_scaled
0	0.050831	2	0.085905
1	0.030492	2	0.051520
2	0.015237	2	0.025731
3	0.016932	2	0.028597
4	0.016932	2	0.028597

[ ]:
```
transactions.describe()
```

[ ]:

	article_id	price	sales_channel_id	price_scaled
count	3.178832e+07	3.178832e+07	3.178832e+07	3.178832e+07
mean	6.962272e+08	2.782927e-02	1.704028e+00	4.701932e-02
std	1.334480e+08	1.918113e-02	4.564786e-01	3.242748e-02
min	1.087750e+08	1.694915e-05	1.000000e+00	0.000000e+00
25%	6.328030e+08	1.581356e-02	1.000000e+00	2.670564e-02
50%	7.145820e+08	2.540678e-02	2.000000e+00	4.292387e-02
75%	7.865240e+08	3.388136e-02	2.000000e+00	5.725092e-02
max	9.562170e+08	5.915254e-01	2.000000e+00	1.000000e+00

1.8 What intrigued you about the data? Why does that matter?

The data appeared in a kaggle competition to recommend the customers clothing basis their demographics and past purchase data.

Recommendor systems are a norm now in various industries like OTT and FMCG. However, it is a little tricky for the fashion industry to predict customer future purchase as the trend keeps on changing rapidly and unlike the Groceries in FMCG. Customers are less likely to buy the same clothing again. Further, we have the image data of the products and the purchase behaviour is also affected by the images of the products available on the website.

We took this data to get familiar with the data and customer behaviour in the clothing industry.

1.9 What would your proposed next steps be?

Our next steps would be exploring the image data and how it can add value to the recommendor system we will be creating. We have to decide on the what model we will chose for this use case basis this EDA and our further exploration of the image data.

Further, the data size is large (>30 GB). Hence we need to decide on how to get enough processing power to create model with this data.

1.10 Observations and Findings

In our initial analysis, we started off by checking for any missing data. We found several data points in the '*Articles*' and '*Customers*' data set missing information which were dealt by imputing them by either their central tendencies or were removed, if insignificant.

We moved on to check the cardinality of the features and found out that some of them have very high cardinality. We performed some necessary actions to deal with or remove them accordingly.

Next, we used techniques to check for outliers in all the 3 data sets and found several outliers present in the '*transactions*' data set. Necessary steps were performed to remove outliers from the data set.

Later, we implemented feature engineering techniques to extract and modify date-time information from '*transactions*' table and performed exploratory data analysis (EDA) to find insights about the pattern and behaviour of customers with their past purchases.