Cleaning and Preprocessing

This is the data cleaning and preprocessing for the Recommendation task

```
import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd

# Loading the dataset
file_path = '/content/drive/MyDrive/FDS project/Recommendations/Whole Data Set.csv'
data = pd.read_csv(file_path)

data.head()
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6
0	Brand Name	COUNTRY	1. POLICY & COMMITMENTS	1.1 What are the company's human rights and en...	Total Section 1.1	1.2 What are the company's vendor/supplier pol...	Total Section 1.2
1	Abercrombie & Fitch	USA	NaN	NaN	3.5	Final scoring \nFR TEAM ONLY	7.25
2	Adidas	Germany	NaN	NaN	4.5	Final scoring \nFR TEAM ONLY	8.5
3	Aeropostale	USA	NaN	NaN	0.5	Final scoring \nFR TEAM ONLY	3.5
4	AJIO	India	NaN	NaN	3	Final scoring \nFR TEAM ONLY	2

5 rows × 65 columns

Columns needed cleaning and editing. They all had huge titles and needed somehting to make it easier for the user to see and work with.

```
# Setting row 1 as the column headers and dropping the first two rows (now redundant)
df = data.rename(columns=data.iloc[0]).drop(data.index[0])

df.head()
```

	Brand Name	COUNTRY	1. POLICY & COMMITMENTS	1.1 What are the company's human rights and environmental policies?	Total Section 1.1	1.2 What are the company's vendor/supplier policies covering human rights and environmental standards across the supply chain? (e.g. Code of Conduct, Terms of Engagement, Supplier Guidebook)	To Sect
1	Abercrombie & Fitch	USA	NaN	NaN	3.5	Final scoring \nFR TEAM ONLY	
2	Adidas	Germany	NaN	NaN	4.5	Final scoring \nFR TEAM ONLY	
3	Aeropostale	USA	NaN	NaN	0.5	Final scoring \nFR TEAM ONLY	

```
# Creating a DataFrame with column names from the original DataFrame
column_names = df.columns.to_numpy()

# Alternatively, you can store them in a new DataFrame
column_names_df = pd.DataFrame({'Column Names': df.columns})

# Print the results
print("Column Names Array:")
print(column_names)
```

```
Column Names Array:
['Brand Name' 'COUNTRY' '1. POLICY & COMMITMENTS'
 '1.1 What are the company’s human rights and environmental policies? '
 'Total Section 1.1'
 '1.2 What are the company’s vendor/supplier policies covering human rights and environmental standards across the su
 'Total Section 1.2'
 '1.3 Publishes the company’s human rights and environmental management procedures (i.e. how the company is putting i
 'Total Section 1.3'
 '1.4 Publishes a strategic plan towards progressively improving human rights and environmental impacts (i.e. roadmap
 'Total Section 1.4'
 '1.5 Publishes an annual sustainability or corporate social responsibility report that is audited or verified by an
 'Total Section 1.5' 'Total Score Section 1' '2. GOVERNANCE'
 '2.1 Publicly discloses contact details for the department the company that has responsibility for human rights and
 'Total Section 2.1'
 '2.2 Publicly discloses the company board member or board committee accountable for human rights and environmental i
 'Total Section 2.2'
 '2.3 Publicly acknowledges how the company prioritises money spent on managing and implementing CSR and sustainabil
 'Total Section 2.3'
 '2.4 Publishes how the company incorporates human rights and environmental performance into purchasing practices'
 'Total Section 2.4' 'Total Score Section 2' '3. TRACEABILITY'
 '3.1 Publishes tier one factories (direct relationship with buyer e.g. production units, Cut Make Trim (CMT) facili
 'Total Section 3.1'
 '3.2 Publishes processing facilities (e.g. ginning and spinning, knitting, weaving, sub-contractors, dyeing and wet
 'Total Section 3.2'
 '3.3 Publishes suppliers of raw materials such as fibres, hides, rubber, dyes, metals, etc. (e.g. raw material provi
 'Total Section 3.3' 'Total Score Section 3' '4. KNOW, SHOW & FIX'
 '4.1 Know, Show & Fix: Publicly discloses human rights and environmental due diligence processes, outcomes and what
 'Total Section 4.1'
 '4.1b Know, Show & Fix: Publicly discloses environmental due diligence processes, outcomes and what brand is doing
 'Total Section 4.1'
 '4.2 Know: Publicly discloses how the company assesses implementation of its supply chain policies (as described in
 'Total Section 4.2'
 '4.3 Show: Publicly discloses findings from its facility-level assessments (e.g. at factories, processing facilitie
 'Total Section 4.3'
```

```
'4.4A Fix: Publicly discloses description and status of the remediation process'
'Total Section 4.4A'
'4.4B Fix: Publicly discloses how the company ensures human rights and environmental grievances from employees and v
'Total Section 4.4B' 'Total Score Section 4'
'5. SPOTLIGHT ISSUES\n \n5.1 – DECENT WORK & PURCHASING PRACTICES\n5.2 – GENDER & RACIAL EQUALITY\n5.3 – SUSTAINABLI
nan
' 5.1 – DECENT WORK & PURCHASING PRACTICES\n\nAs related to SDG 8: Decent Work & Economic Growth – particularly tar
'Total Section 5.1' nan
' 5.2 – GENDER & RACIAL EQUALITY\n\nAs related to SDG 10: Reducing Inequalities – particularly targets 10.2, 10.3 a
'Total Section 5.2'
'5.3 – SUSTAINABLE SOURCING & MATERIALS\n\nAs related to SDG 12: Responsible Consumption and Production – particula
'Total Section 5.3'
'5.4 – OVERCONSUMPTION, WASTE & CIRCULARITY\n\nAs related to SDG 12: Responsible Consumption and Production – parti
'Total Section 5.4'
'5.5 – WATER & CHEMICALS\n\nAs related to SDG 6: Clean Water – particularly targets 6.3 and 6.4 '
'Total Section 5.5'
'5.6 – CLIMATE CHANGE & BIODIVERSITY\n\nAs related to SDG 13: Climate Action target 13.3 and SDG 15: Life on Land –
'Total Section 5.6' 'Total Score Section 5' 'TOTAL POINTS' nan nan]
```

Above, we got a new dictionary that would help us later to map our section to what particular metric they belong to.

```
# Converting the array of column names to a DataFrame
column_names_array = df.columns.to_numpy()
column_names_df = pd.DataFrame(column_names_array, columns=['Column Names'])

column_names_df
```

	Column Names
0	Brand Name
1	COUNTRY
2	1. POLICY & COMMITMENTS
3	1.1 What are the company's human rights and en...
4	Total Section 1.1
...	...
60	Total Section 5.6
61	Total Score Section 5
62	TOTAL POINTS
63	NaN
64	NaN

65 rows × 1 columns

```
# Finding the row number where the brand name is 'Zeeman'
zeeman_row = df[df['Brand Name'].str.contains('Zeeman', na=False)].index.tolist()
zeeman_row
```

```
[250]
```

```
# Removing all rows after row 250 (keeping rows up to and including row 250)
df_trimmed = df.iloc[:250]

# Displaying the last few rows of the trimmed dataframe to confirm the operation
df_trimmed.tail()
```

	Brand Name	COUNTRY	1. POLICY & COMMITMENTS	1.1 What are the company's human rights and environmental policies?	Total Section 1.1	1.2 What are the company's vendor/supplier policies covering human rights and environmental standards across the supply chain? (e.g. Code of Conduct, Terms of Engagement, Supplier Guidebook)	Total Section 1.2	1.3 Publishes the company's human rights and environmental management procedures (i.e. how the company is putting its policies 1.1 and 1.2 into action; simply auditing for compliance is not enough to score points) Reports published after January 2020	Total Section 1.3	1.4 Publ a stra plan to progress impr human r environm impacts roadm v docu Must dates i t
246	Wrangler	USA	NaN	NaN	3.5	Final scoring \nFR TEAM ONLY	7.75	Final scoring \nFR TEAM ONLY	11	Final \nFF
247	Youngor	China	NaN	NaN	0	Final scoring \nFR TEAM ONLY	0	Final scoring \nFR TEAM ONLY	0	Final \nFF
248	Zalando	Germany	NaN	NaN	2.75	Final scoring \nFR TEAM ONLY	6	Final scoring \nFR TEAM ONLY	8	Final \nFF
249	Zara	Spain	NaN	NaN	4.5	Final scoring \nFR TEAM ONLY	7.5	Final scoring \nFR TEAM ONLY	13.5	Final \nFF

```
# Counting the number of NaN values in each column
nan_count = df_trimmed.isna().sum()
nan_count
```

```
Brand Name                                                          0
COUNTRY                                                             0
1. POLICY & COMMITMENTS                                           250
1.1 What are the company's human rights and environmental policies?  250
Total Section 1.1                                                   0
                                                                   ...
Total Section 5.6                                                   0
Total Score Section 5                                               0
TOTAL POINTS                                                        0
NaN                                                               250
NaN                                                               248
Length: 65, dtype: int64
```

```
# Removing all columns that contain any NaN values
df = df_trimmed.dropna(axis=1)

df.head()
```

	Brand Name	COUNTRY	Total Section 1.1	1.2 What are the company's vendor/supplier policies covering human rights and environmental standards across the supply chain? (e.g. Code of Conduct, Terms of Engagement, Supplier Guidebook)	Total Section 1.2	1.3 Publishes the company's human rights and environmental management procedures (i.e. how the company is putting its policies 1.1 and 1.2 into action; simply auditing for compliance is not enough to score points) Reports published after January 2020	Total Section 1.3	1.4 Publishes a strategic plan towards progressively improving human rights and environmental impacts (i.e. roadmap or vision document) Must cover dates in the future	Total Section 1.4	Total Section 1.5	...
1	Abercrombie & Fitch	USA	3.5	Final scoring \nFR TEAM ONLY	7.25	Final scoring \nFR TEAM ONLY	7.5	Final scoring \nFR TEAM ONLY	3	0	...
2	Adidas	Germany	4.5	Final scoring \nFR TEAM ONLY	8.5	Final scoring \nFR TEAM	14	Final scoring \nFR TEAM	3	1	...

```
# Removing columns that contain "Final scoring \nFR TEAM ONLY" in any of its rows
columns_to_remove = df.columns[df.apply(lambda x: (x == "Final scoring \nFR TEAM ONLY").any())]
df_cleaned = df.drop(columns=columns_to_remove)

df_cleaned.columns.tolist()
```

```
['Brand Name',
 'COUNTRY',
 'Total Section 1.1',
 'Total Section 1.2',
 'Total Section 1.3',
 'Total Section 1.4',
 'Total Section 1.5',
 'Total Score Section 1',
 'Total Section 2.1',
 'Total Section 2.2',
 'Total Section 2.3',
 'Total Section 2.4',
 'Total Score Section 2',
 'Total Section 3.1',
 'Total Section 3.2',
 'Total Section 3.3',
 'Total Score Section 3',
 'Total Section 4.1',
 'Total Section 4.1',
 'Total Section 4.2',
 'Total Section 4.3',
 'Total Section 4.4A',
 'Total Section 4.4B',
 'Total Score Section 4',
 'Total Section 5.1',
 'Total Section 5.2',
 'Total Section 5.3',
 'Total Section 5.4',
 'Total Section 5.5',
 'Total Section 5.6',
 'Total Score Section 5',
 'TOTAL POINTS']
```

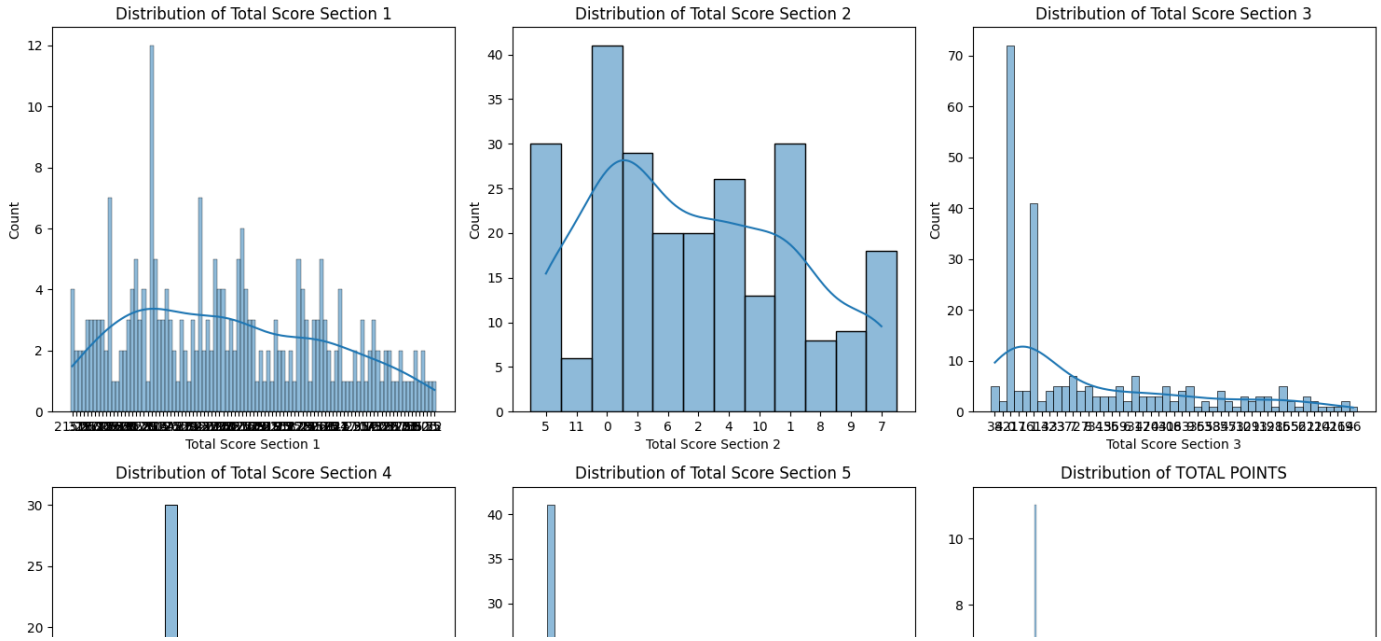
```
df_cleaned.head()
```

	Brand Name	COUNTRY	Total Section 1.1	Total Section 1.2	Total Section 1.3	Total Section 1.4	Total Section 1.5	Total Score Section 1	Total Section 2.1	Total Section 2.2	...	Total Section 4.4B	Total Score Section 4	Se
1	Abercrombie & Fitch	USA	3.5	7.25	7.5	3	0	21.25	1	2	...	2	9	
2	Adidas	Germany	4.5	8.5	14	3	1	31	1	3	...	5	21	
3	Aeropostale	USA	0.5	3.5	1.5	0	0	5.5	0	0	...	0	2	
4	AJIO	India	3	2	6	0	0	11	1	2		1	5	

Visualising the data by scores

Carried out some data visualisation to better understand the distribution of the data points in the dataset.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Basic statistical summary of the numeric columns
stats_summary = df_cleaned.describe()

# Plotting distributions of some key numeric columns (Total Scores of different sections)
plt.figure(figsize=(15, 10))

# List of score columns to plot
score_columns = ['Total Score Section 1', 'Total Score Section 2', 'Total Score Section 3',
                 'Total Score Section 4', 'Total Score Section 5', 'TOTAL POINTS']

for i, column in enumerate(score_columns, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df_cleaned[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()

stats_summary
```




```
# Counting the types of data present in each column of the dataset
data_types_count = df_cleaned.dtypes.value_counts()
data_types_count
```

```
object    32
dtype: int64
```

```
# Identifying the columns that are of object (string) type
object_columns = df_cleaned.select_dtypes(include=['object']).columns.tolist()
object_columns
```

```
['Brand Name',
 'COUNTRY',
 'Total Section 1.1',
 'Total Section 1.2',
 'Total Section 1.3',
 'Total Section 1.4',
 'Total Section 1.5',
 'Total Score Section 1',
 'Total Section 2.1',
 'Total Section 2.2',
 'Total Section 2.3',
 'Total Section 2.4',
 'Total Score Section 2',
 'Total Section 3.1',
 'Total Section 3.2',
 'Total Section 3.3',
 'Total Score Section 3',
 'Total Section 4.1',
 'Total Section 4.1',
 'Total Section 4.2',
 'Total Section 4.3',
 'Total Section 4.4A',
 'Total Section 4.4B',
 'Total Score Section 4',
 'Total Section 5.1',
 'Total Section 5.2',
 'Total Section 5.3',
 'Total Section 5.4',
 'Total Section 5.5',
 'Total Section 5.6',
 'Total Score Section 5',
 'TOTAL POINTS']
```

```
# Identifying the columns that are of float type
float_columns = df_cleaned.select_dtypes(include=['float64']).columns.tolist()
float_columns
```

[ ]

```
# Adjusting the conversion process for each column individually
# Converting all columns (except the first two) to float data type
columns_to_convert = df_cleaned.columns[2:]  # Excluding the first two columns (Brand Name, COUNTRY)

for col in columns_to_convert:
    df_cleaned[col] = df_cleaned[col].astype('float', errors='ignore')

# Checking the data types after conversion
converted_data_types = df_cleaned.dtypes
converted_data_types
```

```
Brand Name                object
COUNTRY                   object
Total Section 1.1        float64
Total Section 1.2        float64
Total Section 1.3        float64
Total Section 1.4        float64
Total Section 1.5        float64
Total Score Section 1    float64
Total Section 2.1        float64
Total Section 2.2        float64
Total Section 2.3        float64
Total Section 2.4        float64
Total Score Section 2    float64
Total Section 3.1        float64
Total Section 3.2        float64
Total Section 3.3        float64
Total Score Section 3    float64
Total Section 4.1        float64
Total Section 4.1        float64
Total Section 4.2        float64
Total Section 4.3        float64
Total Section 4.4A       float64
Total Section 4.4B       float64
Total Score Section 4    float64
Total Section 5.1        float64
Total Section 5.2        float64
Total Section 5.3        float64
Total Section 5.4        float64
Total Section 5.5        float64
Total Section 5.6        float64
Total Score Section 5    float64
TOTAL POINTS             float64
dtype: object
```

```
# Removing the 'COUNTRY' column from the dataset, realised I didnt need this info for the metric decisions for now
df = df_cleaned.drop(columns=['COUNTRY'])
df.head()
```

	Brand Name	Total Section 1.1	Total Section 1.2	Total Section 1.3	Total Section 1.4	Total Section 1.5	Total Score Section 1	Total Section 2.1	Total Section 2.2	Total Section 2.3	...	Total Section 4.4B	Total Score Section 4	Se
1	Abercrombie & Fitch	3.5	7.25	7.5	3.0	0.0	21.25	1.0	2.0	0.0	...	2.0	9.0	
2	Adidas	4.5	8.50	14.0	3.0	1.0	31.00	1.0	3.0	1.0	...	5.0	21.0	
3	Aeropostale	0.5	3.50	1.5	0.0	0.0	5.50	0.0	0.0	0.0	...	0.0	2.0	
4	AJIO	3.0	2.00	6.0	0.0	0.0	11.00	1.0	2.0	0.0	...	1.0	5.0	
5	ALDI Nord	4.0	8.00	10.0	3.0	1.0	26.00	1.0	2.0	0.0	...	4.0	16.0	

5 rows × 31 columns

Normalising the data in the dataset

From the graphs and the data outputed above, it became evident that the data might not be normalised which may cause issues later on, so we looked at normalising the data

```
# Checking to see if the data is normalised
score_columns = [col for col in df.columns if 'Total' in col]
normalization_stats = pd.DataFrame(index=['min', 'max', 'mean', 'std'])
for col in score_columns:
    normalization_stats[col] = [df[col].min(), df[col].max(), df[col].mean(), df[col].std()]
print(normalization_stats)
```

```
      Total Section 1.1  Total Section 1.2  Total Section 1.3  \
min            0.000000           0.000000           0.000000
max            5.000000           9.000000          14.500000
mean           2.922000           5.398000           7.256000
std            1.468688           2.615619           4.375505

      Total Section 1.4  Total Section 1.5  Total Score Section 1  \
min            0.000000           0.000000               0.000000
max            3.000000           1.000000              32.250000
mean           1.604000           0.384000              17.564000
std            1.250375           0.487334               9.306711

      Total Section 2.1  Total Section 2.2  Total Section 2.3  \
min            0.000000           0.000000           0.000000
max            1.000000           3.000000           1.000000
mean           0.712000           1.348000           0.452000
std            0.453739           1.019339           0.498689

      Total Section 2.4  ...  Total Section 4.4A  Total Section 4.4B  \
min            0.000000  ...             0.00000            0.000000
max            6.000000  ...             3.00000            6.000000
mean           1.480000  ...             1.01600            2.712000
std            1.903706  ...             1.08279            2.079991

      Total Score Section 4  Total Section 5.1  Total Section 5.2  \
min                0.000000           0.000000           0.000000
max               37.000000          21.000000           7.000000
mean              10.444000           2.384000           1.772000
std                8.578884           3.614527           1.674285

     Total Section 5.3  Total Section 5.4  Total Section 5.5  \
min           0.000000           0.000000           0.000000
max           9.000000          11.000000           8.000000
mean          3.300000           2.176000           1.672000
std           3.200464           2.364612           2.218129

      Total Section 5.6  Total Score Section 5
min            0.000000               0.000000
max           18.000000              67.000000
mean           4.960000              16.260000
std            4.781826              15.306258

[4 rows x 28 columns]
```

The data is not normalised. So we normalised the dataset.

Range (Min-Max): The minimum values of the scores start at 0, which is common for both normalized and non-normalized data. The maximum values vary significantly across different sections, suggesting they are not normalized to a common scale. Mean: The mean values are not centered around 0, which would be expected if the data were standardized (z-score normalization). Standard Deviation: The standard deviations are not equal to 1, which would also be expected in standardized data.

```
df_normalized = df.copy()
for col in score_columns:
    df_normalized[col] = (df[col] - df[col].min()) / (df[col].max() - df[col].min())
```

```
# Normalising the dataset here
normalized_stats = pd.DataFrame(index=['min', 'max', 'mean', 'std'])
for col in score_columns:
    normalized_stats[col] = [df_normalized[col].min(), df_normalized[col].max(),
                             df_normalized[col].mean(), df_normalized[col].std()]
print(normalized_stats)
```

```
      Total Section 1.1  Total Section 1.2  Total Section 1.3  \
min            0.000000           0.000000           0.000000
max            1.000000           1.000000           1.000000
mean           0.584400           0.599778           0.500414
std            0.293738           0.290624           0.301759

      Total Section 1.4  Total Section 1.5  Total Score Section 1  \
min            0.000000           0.000000                0.00000
max            1.000000           1.000000                1.00000
mean           0.534667           0.384000                0.54462
std            0.416792           0.487334                0.28858

      Total Section 2.1  Total Section 2.2  Total Section 2.3  \
min            0.000000           0.000000           0.000000
max            1.000000           1.000000           1.000000
mean           0.712000           0.449333           0.452000
std            0.453739           0.339780           0.498689

      Total Section 2.4  ...  Total Section 4.4A  Total Section 4.4B  \
min            0.000000  ...            0.000000            0.000000
max            1.000000  ...            1.000000            1.000000
mean           0.246667  ...            0.338667            0.452000
std            0.317284  ...            0.360930            0.346665

      Total Score Section 4  Total Section 5.1  Total Section 5.2  \
min                0.000000           0.000000           0.000000
max                1.000000           1.000000           1.000000
mean               0.282270           0.113524           0.253143
std                0.231862           0.172120           0.239184

     Total Section 5.3  Total Section 5.4  Total Section 5.5  \
min           0.000000           0.000000           0.000000
max           1.000000           1.000000           1.000000
mean          0.366667           0.197818           0.209000
std           0.355607           0.214965           0.277266

      Total Section 5.6  Total Score Section 5
min            0.000000                0.000000
max            1.000000                1.000000
mean           0.275556                0.242687
std            0.265657                0.228452

[4 rows x 28 columns]
```

```
df.head()
```

	Brand Name	Total Section 1.1	Total Section 1.2	Total Section 1.3	Total Section 1.4	Total Section 1.5	Total Score Section 1	Total Section 2.1	Total Section 2.2	Total Section 2.3	...	Total Score Section 4	Total Section 5.1	Se
1	Abercrombie & Fitch	3.5	7.25	7.5	3.0	0.0	21.25	1.0	2.0	0.0	...	9.0	2.0	
2	Adidas	4.5	8.50	14.0	3.0	1.0	31.00	1.0	3.0	1.0	...	21.0	6.0	
3	Aeropostale	0.5	3.50	1.5	0.0	0.0	5.50	0.0	0.0	0.0	...	2.0	0.0	
4	AJIO	3.0	2.00	6.0	0.0	0.0	11.00	1.0	2.0	0.0	...	5.0	0.0	
5	ALDI Nord	4.0	8.00	10.0	3.0	1.0	26.00	1.0	2.0	0.0	...	16.0	2.0	

5 rows × 32 columns

Saving the dataset to drive

Saved the dataset into drive so as to use it for the Recommendation

```
df.to_csv('/content/drive/MyDrive/FDS project/Recommendations/final.csv', index=False)
```