DAT7015

BIG DATA TECHNOLOGIES

ASSESSMENT 2

BIG DATA ANALYTICS

IN

THE FASHION RETAIL INDUSTRY

Ikenna Joseph Chukwudum (ijc1crt@bolton.ac.uk)

Student No: 2215323

University of Bolton

Lecturer: Dr. Pradeep Hewage

BIG DATA ANALYTICS

IN

THE FASHION RETAIL INDUSTRY

AN EXPLORATORY ANALYSIS

A Report by

Ikenna Chukwudum

January 2023

Abstract

Over the past two decades, especially with the evolution of big data, AI and machine learning, there has been significant growth in data collection and analysis in the fashion industry. In fact, the term “fashion data” has been coined to envelope this concept. Growing competition among fashion brands, big and small, has led to more value being placed on data collection in customer interests, trends and purchasing power (Jain et al., 2017). Season, climate, geographical location, gender, age et cetera are also among the factors that influence purchase and choice of fashion; thus, brands are constantly growing their fashion databases based on these. Big data with all its advantages, however, faces teeming issues as it relates to ethics, manipulation, privacy, transparency and paternalism, and fashion data is not left out. In this report, fashion datasets will be analysed and visualized, the challenges of fashion big data analytics will be discussed and AI solutions will be proffered.

Keywords*: Consumer Experience; Fashion Retail; Big Data; AI; Internet of Things, Trend Forecasting; Machine Learning; Fashion Industry; Data Analytics*

CONTENTS

1. INTRODUCTION 1
1.1 Background 1
2. DATA ANALYSIS AND VISUALIZATION 3
2.1 Wrangling, Cleaning and Merging 3
2.1.1 Data description 3
2.1.2 Data wrangling 3
2.1.3 Merging the dataframes with SQL 6
2.2 Questions 8
2.2.1 Question 1: What were the products with an average rating of 3.0 and below? 8
2.2.2 Question 2: How much could the fashion retail companies make from sale of black coloured products? 9
2.3 MongoDB Application to Product Attributes Field, “p_attributes” 11
2.4 Working with Spark(PySpark) 14
2.5 Machine Learning 17
2.5.1 Supervised machine learning 18
2.5.2 Unsupervised machine learning: KMeans clustering 21
2.6 Visualization 24
3. Issues in the Use of Big Data Analytics in the Fashion Retail Industry 27
4. Methodology 28
5. Issues Identified and Discussed 29
5.1 Are There Sufficient, Accurate and Ethical Data Collection Pipelines in the Industry? 29
I. Discussion 29
II. Solution 29
5.2 Are Environmental, Climate Change, Societal and Cultural Factors Considered in Fashion Data Analytics? 30
I. Discussion 30
II. Solution 31
6. Conclusion 33
References 34
Appendix 36

1. INTRODUCTION

The fashion retail industry includes those companies, individuals and processes that facilitate fashion products getting to the final consumer from the manufacturers. They are an integral part of the fashion industry and are largely synonymous with it as many manufacturers and fashion brands actively participate in directly getting their products to their customers. Artificial intelligence has spurned a new relationship among fashion brands, suppliers and consumers in the fashion industry. Huge inflows of data, facilitated by the explosive growth of social media and online marketplaces like Amazon, has led to fashion companies faced with the challenge of managing the different data with their correlations and complexities (Thomassey, 2018). Most global fashion houses now have dedicated data analysis teams to streamline these vast data in order to keep them competitive and maximise profits.

The processes and techniques involved in the wrangling, cleaning, filtering, sorting, interpreting and visualization of large and diverse , structured or unstructured datasets is called big data analytics. The main reasons underlying big data application to the fashion retail industry include customer experience and engagement, marketing, waste reduction, trend forecasting, improved supply chain, quality control and counterfeits minimization (Silva et al., 1970). Textile quality control, intelligent tracking systems, design support systems and sensory evaluation are some of the other ways artificial intelligence systems has guided the growth in the fashion industry in the twenty first century and this emphasizes how crucial it is for fashion companies to master these data systems.

1.1 Background

There is an alarming ignorance to the negative implications derived from the rapid marriage of fashion to big data. Competition and profit margins have largely blurred the vision to the breach of customer privacy, data security, price inflation, artificial scarcity and paternalism that major fashion companies have now embraced. This is on the one hand.

On the other hand, many believe the fashion retail industry has not done enough in adopting artificial intelligence and machine learning technologies, especially with regards the smaller, local brands. This raises some questions as to the issues with big data analytics in the fashion industry: Is there sufficient and accurate data collection for the industry? Is data coverage broad enough (to include the aging demographic and less advanced countries)? Are the right parameters considered when design choices are made or are AI tilted for profits? Are environmental and climate factors in production factored into useful data? Are customer comfort and need prioritized? Are culture and religious choices factored into design useful data?

In this report, samples of global fashion brands data have been collected, analysed and visualized—the processes of which are detailed here. Questions relating to how various parameters and attributes affect customer interest are investigated; and machine learning techniques are compared and adopted for predictive pricing analysis. The challenges facing big data analytics and technologies in the fashion industry are also discussed. Finally, various big data technologies will be recommended to ameliorate the teeming issues facing big data analytics in the fashion retail industry.

2. DATA ANALYSIS AND VISUALIZATION

Here, big data tools were adopted to show practically the impact of big data and AI on samples of fashion data. Hence, work was done on two datasets of contemporary fashion brands using several of these tools from pandas to Mongo DB. These datasets were cleaned, wrangled, merged and filtered to answer questions like if product colour impacted on product price or average rating by customers. Unstructured data was also analysed to determine specific attributes in brand products. Apache Spark was used for further filtering to answer questions in relation to the available parameters. Machine learning was adopted to predict product price in relationship to the other independent parameters of the merged datasets. Finally, visualizations were done using matplotlib, seaborn and Power BI to determine the average rating and average price per brand.

* The complete step-by-step processes involved in the data analysis via Jupyter Notebook can be accessed in the Appendix section of this report.

2.1 Wrangling, Cleaning and Merging

2.1.1 Data description

Python was used for the bulk of the analysis because of its flexibility, as well as its ability to install a number of other useful programs and import libraries. The integrated development environment IDE used was Jupyter Notebook installed with anaconda distribution.

To begin the analysis, the two datasets were visually inspected and it was observed that dataset 1, “fashion_dataset.csv”, was more detailed offering over 8 columns with product ID, price, colour and average rating among the attributes listed. The second dataset, “fashion_brand_details.xlsx” was much simpler with only 2 columns: brand ID and brand name.

The two datasets “fashion_dataset.csv” and “fashion_brand_details.xlsx” were read into dataframes design1_df and design2_df respectively using pandas. The dataframes created contained 14,329 rows and 9 fields (design1_df) and 1020 rows and 2 fields (design2_df). Work was carried out on cleaning each dataframe in order for them to be merged in a master fashion data frame, fashion_master_df.

2.1.2 Data wrangling

To start the process, the required libraries were imported: NumPy and pandas. The os library was also imported to track where the datasets were in the drive. Thereafter, panda’s “read” method was used and the dataframes created. Dataframes are flexible forms of data which can be cleaned, wrangled and visualized more easily with python libraries.

Preliminary Wrangling

In [1]:
```
# Imported numpy and pandas packages for cleaning

import numpy as np
import pandas as pd
```

> The analysis was started by reading the datasets "fashion_dataset.csv" and "fashion_brand_details.xlsx" to the dataframes design1_df and design2_df respectively, and thereafter investigating them

In [2]:
```
# The os library was imported to ensure datasets could be located in the directory
import os
wd = os.getcwd()
files = os.listdir(wd)  # Get all the files in that directory
list(files)
```

Out[2]:
```
['.ipynb_checkpoints',
 'ALX PROJECTS',
 'bigdata digram.pbix',
 'bigdataassignment_2.ipynb',
 'bigdataassignment_2.pdf',
 'blackaprdts.csv',
 'blackbprdts.csv',
 'blackcprdts.csv',
```

Slide 1: Relevant packages were imported for wrangling

> The first data frame design1_df

In [3]:
```
design1_df = pd.read_csv("fashion_dataset.csv")
```

In [4]:
```
design1_df.head(3)
```

Out[4]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes
0	1518329.0	Dupatta Bazaar White Embroidered Chiffon Dupatta	899.0	White	Dupatta Bazaar	1321.0	4.548827	White embroidered dupattaChiffon Hand-...	{'Occasion': 'Daily', 'Pattern': 'Embroidered'...
1	5829334.0	Roadster Women Mustard Yellow Solid Hooded Swe...	1199.0	Mustard	Roadster	5462.0	4.313255	Mustard yellow solid sweatshirt, has a hood, t...	{'Body Shape ID': '443,424,324', 'Body or Garm...
2	10340119.0	Inddus Peach-Coloured & Beige Unstitched Dress...	5799.0	Peach	Inddus	145.0	4.068966	Peach-Coloured and beige woven design unstitch...	{'Bottom Fabric': 'Cotton Blend', 'Bottom Patt...

Slide 2: The first dataframe design1_df was created

The design1_df dataframe was investigated with .head() and .info() methods and it was apparent early on that some cleaning would be required as it was a relatively larger dataframe with more columns, compared to design2_df. What stood out in this dataframe were the null values present in every column especially in the “ratingCount” and “avg_rating” fields, which both had over half of their entries as empty values. This prompted the need to fill the empty values as dropping that many rows could skew the investigations eventually. The “p_id” (product ID) column was filled with zeros using the .fillna() method. The “name”, “colour”, “brand”, “description” and “p_attributes” (product attributes) fields were filled with “NA” for their null values. The “avg_rating” (average rating), “ratingCount” and “price” columns were each filled with their means; and since “avg_rating” and “price” were floats, they were each rounded up to 2 decimal places.

```
In [6]:
# The .info() method showed a number of empty entries especially in ratingCount and avg_rating columns
# There were also missing entries in every other column
# The missing values in product ID column were filled with zeros; the price, rating count and average rating
#columns were filled with their means and all missing string values were filled with 'NA'

design1_df['p_id'] = design1_df['p_id'].fillna(0).astype(int)

In [7]:
# These float data type columns filled with the mean of the entries

design1_df['avg_rating'].fillna(design1_df['avg_rating'].mean(), inplace=True)
design1_df['ratingCount'].fillna(design1_df['ratingCount'].mean(), inplace=True)
design1_df['price'].fillna(design1_df['price'].mean(), inplace=True)

In [8]:
# String columns filled with 'NA'

design1_df[['name','colour','brand','description','p_attributes']] = design1_df[['name','colour','brand','description','p_att
```

Slide 3: .fillna() method was used on the columns of design1_df

These processes left the design1_df dataframe without null values, however, using the .duplicated() method, 106 duplicates were discovered and subsequently dropped. To conclude our cleaning, the product ID "p_id" was set as index, with the number of rows at 14,223.

```
In [17]:
design1_df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 14223 entries, 1518329 to 19361058
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          14223 non-null  object
 1   price         14223 non-null  float64
 2   colour        14223 non-null  object
 3   brand         14223 non-null  object
 4   ratingCount   14223 non-null  int32
 5   avg_rating    14223 non-null  float64
 6   description   14223 non-null  object
 7   p_attributes  14223 non-null  object
dtypes: float64(2), int32(1), object(5)
memory usage: 944.5+ KB
```

Slide 4: design1_df was inspected with .info()

```
In [16]:
# Cleaned data frame

design1_df.head(3)
```

Out[16]:

p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes
1518329	Dupatta Bazaar White Embroidered Chiffon Dupatta	899.0	White	Dupatta Bazaar	1321	4.55	White embroidered dupattaChiffon Hand-...	{'Occasion': 'Daily', 'Pattern': 'Embroidered'...
5829334	Roadster Women Mustard Yellow Solid Hooded Swe...	1199.0	Mustard	Roadster	5462	4.31	Mustard yellow solid sweatshirt, has a hood, t...	{'Body Shape ID': '443,424,324', 'Body or Garm...
10340119	Inddus Peach-Coloured & Beige Unstitched Dress...	5799.0	Peach	Inddus	145	4.07	Peach-Coloured and beige woven design unstitch...	{'Bottom Fabric': 'Cotton Blend', 'Bottom Patt...

Slide 5: design1_df was inspected with .head()

The design2_df dataframe was investigated with Python's .head() method to explore the first few data entries. The index of the two-column dataframe was set to the "brand_id" column, with the "brand name" column containing the names of the brands. Further investigations occurred with .info(), .describe() and .isnull() methods, the latter of which was used to determine if there were null values or empty cells. The .info() and .describe() methods were used to output basic information like data type and simple calculations and averages on the dataframe. Next, duplicate values were checked for in this dataframe and with the absence of any, the cleaning was wrapped up after index was set to "brand_id".

The second data frame design2_df

```
In [18]: design2_df = pd.read_excel("fashion_brand_details.xlsx")

In [19]: # The second data frame was investigated using .describe(), .info(), .isnull() as well as duplicates checked
         design2_df.head()
```

Out[19]:

	brand_id	brand_name
0	1	513
1	2	109F
2	3	20Dresses
3	4	250 Designs
4	5	3Pin

Slide 6: The second dataframe design2_df was created and inspected with .head()

```
In [24]: # Index was set as brand_id as there were similarities between brand_id and the newly created index
         design2_df.set_index('brand_id', inplace=True)

In [25]: design2_df.head()
```

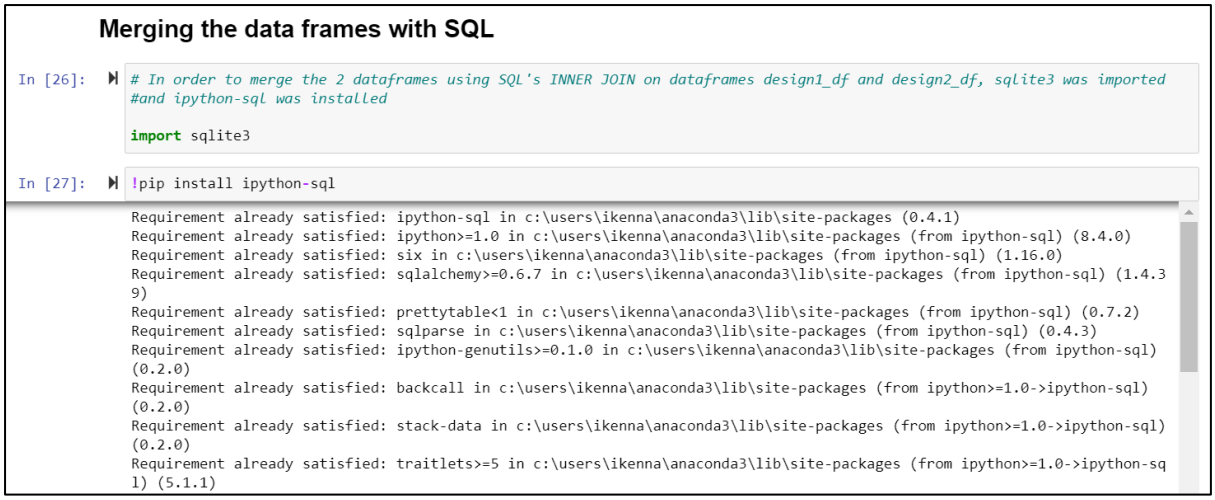
Out[25]:

	brand_name
brand_id	
1	513
2	109F
3	20Dresses
4	250 Designs
5	3Pin

Slide 7: design2_df's index was set and dataframe displayed with .head()

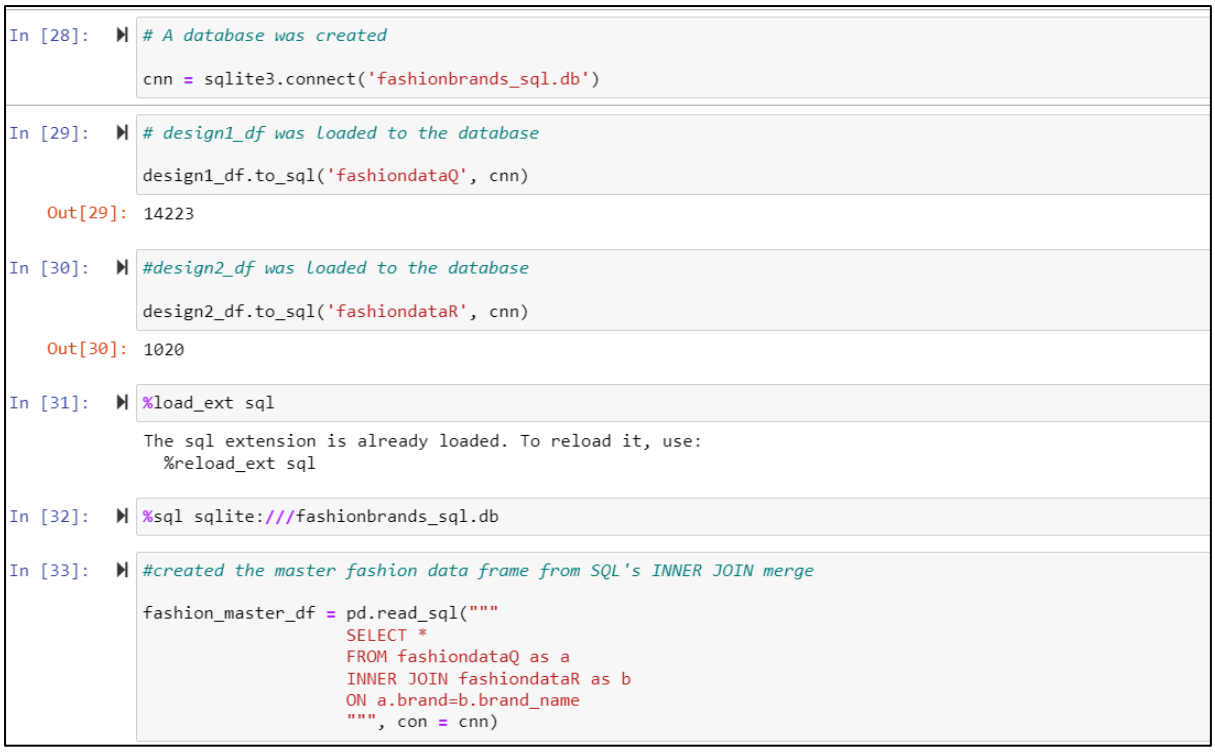
2.1.3 Merging the dataframes with SQL

The flexibility of python to import libraries to perform functions was implied here. Hence, sqlite3 was imported and ipython-sql was installed thereafter. The goal of the installation was to use SQL's inner join function to merge the already cleaned design1_df and design2_df dataframes.



Slide 8: Sqlite3 was imported and ipython-sql package installed

Subsequently, a database was created and connection made to sqlite3. Design1_df was connected to this database as "fashiondataQ" and design2_df was connected as "fashiondataR". The number of rows in each dataframe was displayed. The fashiondataQ and fashiondataR data were joined using inner join on "brand" and "brand_name" respectively. The master dataframe from the merged data, fashion_master_df was created using panda's .read_sql() method. After the merge, .info() method showed the number of rows reduced to 8052 while the number of columns increased to 11. The master dataframe was moved to csv file as "fashion1_master.csv".



Slide 9: The dataframes were loaded to their respective databases and the INNER JOIN command initiated

```
In [35]: fashion_master_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8052 entries, 0 to 8051
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   p_id          8052 non-null   object
 1   name          8052 non-null   object
 2   price         8052 non-null   float64
 3   colour        8052 non-null   object
 4   brand         8052 non-null   object
 5   ratingCount   8052 non-null   int64
 6   avg_rating    8052 non-null   float64
 7   description   8052 non-null   object
 8   p_attributes  8052 non-null   object
 9   brand_id      8052 non-null   int64
 10  brand_name    8052 non-null   object
dtypes: float64(2), int64(2), object(7)
memory usage: 692.1+ KB

In [36]: # Dataframe moved to csv file

fashion_master_df.to_csv('fashion1_master.csv', index=False)
```

Slide 10: The merged master dataframe, fashion_master_df, was inspected and converted to .csv file "fashion1_master.csv"

2.2 Questions

With the master dataframe, fashion_master_df created, a number of questions could then be answered.

2.2.1 Question 1: What were the products with an average rating of 3.0 and below?

The purpose of this question was to enable the fashion retailers make speedy data-driven decisions as to whether these poorly rated products of 3.0 and below should be completely discarded or worked on and improved. This would hence be based on a comparison of the minimum price of the poorly rated products to that of all the products in general.

The master dataframe was filtered for the required information to create a new dataframe, review_prdts_df. This new dataframe was further filtered for "avg_rating" entries not greater than 3.0 and then sorted by "price".

Questions

> Q1: What were the products with an average rating of 3.0 and below? Information on this would determine whether product will be discarded or improved.

```
In [37]: # Data frame was created to review products of rating 3.0 and lower

         review_prdts_df = fashion_master_df[fashion_master_df['avg_rating'] <= 3].sort_values('price', ascending=True)

In [38]: review_prdts_df.reset_index(drop=True, inplace=True)

In [39]: review_prdts_df
```

Out[39]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes	brand_id	brand_name
0	1411010	Dupatta Bazaar Beige Striped Chanderi Cotton S...	449.0	Beige	Dupatta Bazaar	6	2.83	Beige striped dupatta, has a zari border ...	{'Ornamentation': 'Zari', 'Print or Pattern Ty...	242	Dupatta Bazaar
1	14745736	Baby Lakshmi Girls Orange & Off White Ready to...	698.0	Orange	Baby Lakshmi	1	2.00	Orange and off white solid lehenga choli, O...	{'Blouse Closure': 'Button', 'Blouse Fabric': ...	108	Baby Lakshmi

Slide 11: The review_prdts_df dataframe was created and sorted for price

```
In [40]: # The least price in the master dataset

         fashion_master_df['price'].min()

Out[40]: 295.0
```

> Answer 1: The fashion retail companies are advised to not discard the average rated products but rather improve them. This is because these products still generate revenue from per unit sale well above the least sale price of 295 dollars. Hence, products being discarded could reduce profit margins

Slide 12: The least product price of the master dataframe was determined

The results showed that the least price of what was termed the “poorly reviewed products” was £449 while the overall minimum price was £295. It thus implied that the fashion retail brands should not discard these poorly rated products but should rather improve them as they have a relatively higher per unit sale value. Hence, discarding these products could impact on profit margins.

2.2.2 Question 2: How much could the fashion retail companies make from sale of black coloured products?

This question explored the brand name, description, average rating and price of the black products. As black is usually a fashionable colour with diverse use in both work and casual outing, it was imperative that this question was investigated.

A new dataframe black_prdts_df was created from filtering the master dataframe for ‘colour==”black” ’ and sorting by “price”. Further filtering was done for the data required to give filtered_black_prdts_df.

Q2: This question explored brand name, description, average rating and price of products that were of colour black. How much did fashion companies actually make from black products?

```
In [41]: # Creating the black products dataframe

         black_prdts_df = fashion_master_df[fashion_master_df['colour'] == 'Black'].sort_values('price', ascending=False)

In [42]: # Dropping any new index created

         black_prdts_df.reset_index(drop=True, inplace=True)

In [43]: black_prdts_df.head()
```

Out[43]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes	brand_id	brand_name
0	16920516	Masaba Woman Black Tulip Cape Set	30000.0	Black	Masaba	184	4.10	V-neckline Full sleeves Embr...	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...	553	Masaba
1	18585472	Mitera Black & Pink Floral Pure Linen Saree	22300.0	Black	Mitera	184	4.10	Design Details Black and pi...	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...	583	Mitera
2	13168738	Justanned Plus Women Plus Size Black Solid Lea...	19999.0	Black	Justanned	184	4.10	Black solid jacket, has a spread collar, 6 poc...	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...	442	Justanned

Slide 13: black_prdts_df was created and displayed

```
In [44]: # Selecting required fields

         filtered_black_prdts_df = black_prdts_df.loc[:,['brand_name','avg_rating','price', 'description']]

In [45]: filtered_black_prdts_df.head()
```

Out[45]:

	brand_name	avg_rating	price	description
0	Masaba	4.10	30000.0	V-neckline Full sleeves Embr...
1	Mitera	4.10	22300.0	Design Details Black and pi...
2	Justanned	4.10	19999.0	Black solid jacket, has a spread collar, 6 poc...
3	Justanned	4.10	19998.0	Black solid lightweight biker jacket with zip...
4	Justanned	4.67	15998.0	Black solid striped biker jacket with zip det...

Slide 14: filtered_black_prdts_df was created for the required information

It was discovered that 268 brands made black clothes. The percentage of clothes that were black was approximately 13%. The average price of black clothes was £2,623.68 which was below, but close to, the general average of £2,876.96; and the mean average rating of black clothes was relatively high at 4.10. The processes used to generate these results are displayed in the slide below:

```
In [48]: # 268 brands made black clothes

         len(filtered_black_prdts_df['brand_name'].unique())

Out[48]: 268

In [49]: # Percentage of black products was approximately 13%

         per_blck = (len(black_prdts_df)/len(fashion_master_df))*100
         per_blck

Out[49]: 13.02781917536016

In [50]: # Dataframe moved to csv file

         filtered_black_prdts_df.to_csv('upfilteredblackprdts.csv', index=False)

In [51]: # Average price of black clothes

         filtered_black_prdts_df['price'].mean()

Out[51]: 2623.6787416587226

In [52]: # Average price of all clothes

         fashion_master_df['price'].mean()

Out[52]: 2876.9600099354197

In [53]: # Mean of average rating

         filtered_black_prdts_df['avg_rating'].mean()

Out[53]: 4.103365109628194
```

> Answer 2: The findings showed that the average pricing of black clothes was £2623.68. This was lower than the general average of £2876.96 but close enough to it. The average rating though remained high with a mean of over 4.10. Hence, black clothes generally had above average rating despite their average sale price.

Slide 15: Analysis was carried out on filtered_black_prdts_df

Hence, black clothes generally had an above average rating despite their average sale price. This analysis showed that fashion brands could increase their profit margins by further increasing the number of black clothes produced as well as perhaps being even more flexible with the pricing of the products.

2.3 MongoDB Application to Product Attributes Field, "p_attributes"

Still using the black products dataframe black_prdts_df, filtering was done for brand name and product attributes to create the mongofiltered_black_prdts_df dataframe. Library "ast" was imported with class "literal_eval" which was used to change the data type of the "p_attributes" entries from string to dictionary. The .csv file "blackcprdts.csv" was created from the dataframe to be exported to MongoDB Compass for further analysis.

```
mongofiltered_black_prdts_df['p_attributes']

Out[57]: 0       {'Add-Ons': 'NA', 'Better Cotton Initiative': ...
         1       {'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...
         2       {'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...
         3       {'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
         4       {'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
                                    ...
         1044    {'Border': 'Woven Design', 'Fabric': 'Art Silk...
         1045    {'Add-Ons': 'NA', 'Body Shape ID': '443,324,42...
         1046    {'Body Shape ID': '333,424', 'Body or Garment ...
         1047    {'Body or Garment Size': 'To-Fit Denotes Body ...
         1048    {'Body Shape ID': '333,424', 'Body or Garment ...
         Name: p_attributes, Length: 1049, dtype: object

In [58]: # ast library imported to change the attributes data type from string to dictionary

         from ast import literal_eval

In [59]: mongofiltered_black_prdts_df['Attributes'] = mongofiltered_black_prdts_df['p_attributes'].apply(lambda x: literal_eval(x))
```

Slide 16: mongofiltered_black_prdts_df "p_attributes" column was inspected and ast imported

```
In [61]: type(mongofiltered_black_prdts_df['Attributes'][0])

Out[61]: dict

In [62]: mongofiltered_black_prdts_df.drop('p_attributes', axis=1, inplace=True)

         C:\Users\IKENNA\AppData\Local\Temp\ipykernel_16708\1633810029.py:1: SettingWithCopyWarning:
         A value is trying to be set on a copy of a slice from a DataFrame

         See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
           mongofiltered_black_prdts_df.drop('p_attributes', axis=1, inplace=True)

In [63]: mongofiltered_black_prdts_df.head()

Out[63]:
```

	brand_name	Attributes
0	Masaba	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...
1	Mitera	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...
2	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...
3	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
4	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...

```
In [64]: # The filtered dataframe is moved to csv to be imported to mongoDB Compass for further analysis

         mongofiltered_black_prdts_df.to_csv('blackcprdts.csv', index=False)
```

Slide 17: mongofiltered_black_prdts_df dict type "Attributes" column was created and "p_attributes" dropped; output was moved to csv

However, to work within Jupyter Notebook, pymongo was imported and installed. A client was created at localhost 27017 server and database "fashiondb" was initiated. A collection with 1049 documents of product attributes was instantiated. Thereafter, Pymongo filtering tools were used to select "Columbia" brands.

```
In [65]: # To work in the notebook, pymongo is installed and imported

         !pip install pymongo

         Requirement already satisfied: pymongo in c:\users\ikenna\anaconda3\lib\site-packages (4.3.3)
         Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in c:\users\ikenna\anaconda3\lib\site-packages (from pymongo) (2.2.
         1)

In [66]: import pymongo

In [67]: # Server is set at localhost:27017

         client = pymongo.MongoClient("localhost", 27017)

In [68]: # The database is created and named

         db = client.fashiondb

In [69]: db.name

Out[69]: 'fashiondb'

In [70]: # The list of databases at localhost 27017

         client.list_database_names()

Out[70]: ['admin', 'config', 'fashiondb', 'local']

In [71]: # A collection is created

         coll = db.products_attributes

In [72]: # Details on the product attributes collection

         db.products_attributes

Out[72]: Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'fashiondb'),
         'products_attributes')
```

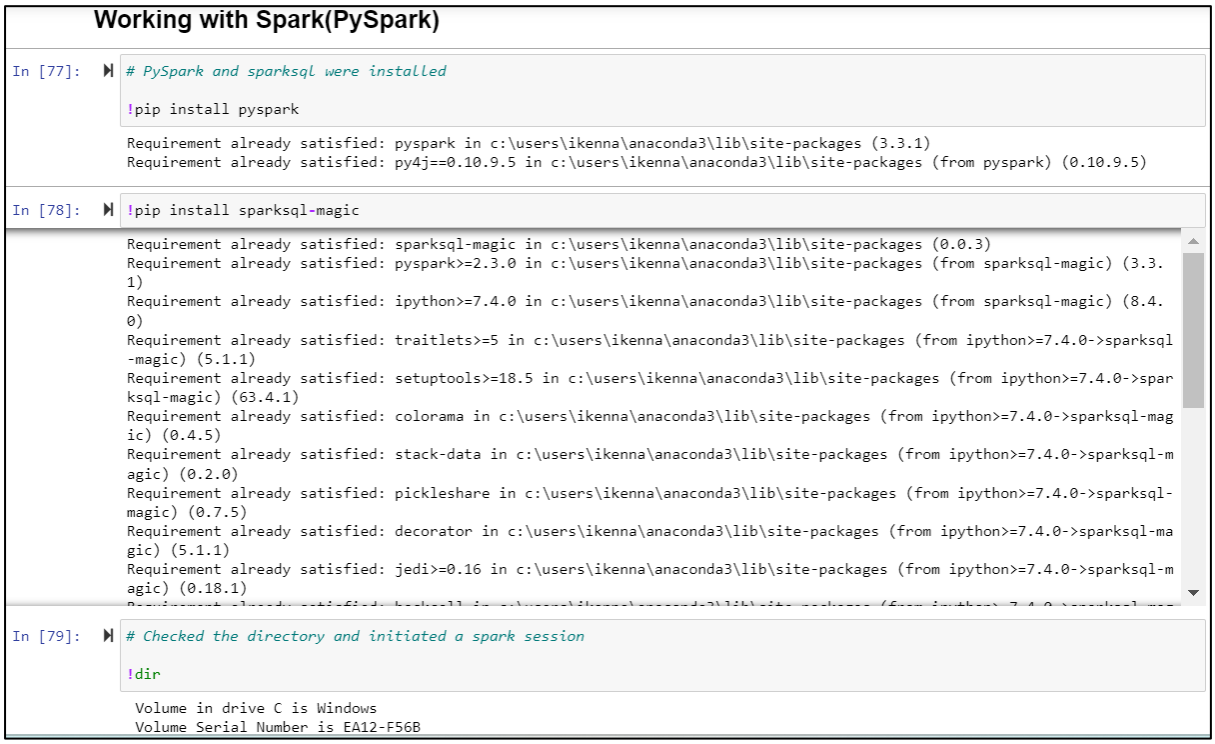
Slide 18: pymongo was installed and imported; the database and collection were created

```
In [73]: db.list_collection_names()

Out[73]: ['products_attributes']

In [74]: # Collection has 1049 documents

         coll.count_documents({})

Out[74]: 1049

In [75]: # Selected a document

         coll.find_one()

Out[75]: {'_id': ObjectId('63872b29c557eb483de5b023'),
          'brand_name': 'Masaba',
          'p_attributes': "{'Add-Ons': 'NA', 'Better Cotton Initiative': 'Regular', 'Bottom Closure': 'Zip', 'Bottom Fabric': 'Silk B
         lend', 'Bottom Pattern': 'Printed', 'Bottom Type': 'Trousers', 'Character': 'NA', 'Lining': 'NA', 'Neck': 'Round Neck', 'Num
         ber of Pockets': 'NA', 'Occasion': 'Casual', 'Sleeve Length': 'Long Sleeves', 'Sustainable': 'Regular', 'Top Fabric': 'Silk
         Blend', 'Top Pattern': 'Solid', 'Top Type': 'Top', 'Trends': 'NA', 'Wash Care': 'Dry Clean'}"}

In [76]: # Filtered and selected several documents

         cur = coll.find({'brand_name': "Columbia"}, {'p_attributes': 1, '_id': 1})
         for doc in cur:
             print(doc)

         {'_id': ObjectId('63872b29c557eb483de5b029'), 'p_attributes': "{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Body or Garment Si
         ze': 'Garment Measurements in', 'Character': 'NA', 'Closure': 'Zip', 'Collar': 'Hooded', 'Fabric': 'Nylon', 'Features': 'Ref
         lective Strip', 'Hemline': 'Straight', 'Length': 'Longline', 'Lining Fabric': 'Polyester', 'Main Trend': 'NA', 'Number of Po
         ckets': '3', 'Occasion': 'Sports', 'Pattern': 'Solid', 'Print or Pattern Type': 'Solid', 'Sleeve Length': 'Long Sleeves', 'S
         port': 'Outdoor', 'Surface Styling': 'NA', 'Sustainable': 'Regular', 'Technology': 'NA', 'Technology Present': 'Yes', 'Typ
         e': 'Padded Jacket', 'Wash Care': 'Machine Wash', 'Weave Type': 'Woven', 'Where-to-wear': ''}"}
         {'_id': ObjectId('63872b29c557eb483de5b02d'), 'p_attributes': "{'Add-Ons': 'NA', 'Body Shape ID': '333,424', 'Body or Garmen
         t Size': 'Garment Measurements in', 'Character': 'NA', 'Closure': 'Zip', 'Collar': 'Hooded', 'Contact Brand or Retailer for
         pre-sales product queries': 'chogori india retail ltd. , a-16, rear-side, mohan co-operative industrial estate, main mathura
         road, new delhi-110044', 'Fabric': 'Nylon', 'Features': 'Insulator', 'Hemline': 'Straight', 'Length': 'Regular', 'Lining Fab
         ric': 'Nylon', 'Main Trend': 'NA', 'Number of Pockets': '2', 'Occasion': 'Casual', 'Pattern': 'Solid', 'Print or Pattern Typ
         e': 'Solid', 'Sleeve Length': 'Long Sleeves', 'Sport': 'NA', 'Surface Styling': 'Zip Detail', 'Sustainable': 'Regular', 'Tec
         hnology': 'NA', 'Technology Present': 'No', 'Type': 'Padded Jacket', 'Wash Care': 'Machine Wash', 'Weave Type': 'Woven'}"}
```

Slide 19: collection was inspected and documents selected

2.4 Working with Spark(PySpark)

To start with, pyspark and sparksql-magic were installed, the directory was looked into and a spark session was initiated. Using the .read.csv() method, spark dataframe fashionspark_df was created.



Slide 20: pyspark and sparksql-magic were installed

Thereafter, the columns were explored and the schema also in order to see column data types. It was observed that all the column data types were strings, hence explicitly applying a schema with the help of StructType method, the data types were correctly appropriated. Other pyspark methods were explored to filter, rename, drop and group the data fields.

```
In [81]:
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # Property used to format output tables better
spark
```

Out[81]:

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.2.2

Master
local[*]

AppName
pyspark-shell

```
In [82]:
# Loaded the master data from csv to a dataframe using spark

fashionspark_df = spark.read.csv('fashion1_master.csv', header=True, sep=",")
fashionspark_df.show(5, truncate=True)
```

```
+--------+--------------------+------+-------+--------------+-----------+---------+--------------------+--------------------
-+--------+--------------+
|    p_id|                name| price| colour|         brand|ratingCount|avg_rating|         description|        p_attribute
s|brand_id|    brand_name|
+--------+--------------------+------+-------+--------------+-----------+---------+--------------------+--------------------
-+--------+--------------+
| 1518329|Dupatta Bazaar Wh...| 899.0|  White|Dupatta Bazaar|       1321|     4.55|White embroidered...|{'Occasion': 'Da
i...|     242|Dupatta Bazaar|
| 5829334|Roadster Women Mu...|1199.0|Mustard|      Roadster|       5462|     4.31|"Mustard yellow s...|{'Body Shape I
D':...|     750|      Roadster|
|10340119|Inddus Peach-Colo...|5799.0|  Peach|        Inddus|        145|     4.07|Peach-Coloured an...|{'Bottom Fabri
c':...|     389|        Inddus|
```

Slide 21: A spark session was started and fashionspark_df dataframe created from "fashion1_master.csv"

```
In [83]:
# Explored the columns

fashionspark_df.columns
```

```
In [84]:
# Explored the schema to see column data types

fashionspark_df.printSchema()
```

```
In [85]:
from pyspark.sql.types import *
```

```
In [86]:
# Created a list of the schema in the format column, data type
labels = [
     ('p_id',StringType()),
     ('name',StringType()),
     ('price',DoubleType()),
     ('colour',StringType()),
     ('brand',StringType()),
     ('ratingCount',IntegerType()),
     ('avg_rating',DoubleType()),
     ('description',StringType()),
     ('p_attributes',StringType()),
     ('brand_id',IntegerType()),
     ('brand_name',StringType())
]
```

```
In [87]:
# Used StructType method to examine the schema

schema = StructType([StructField (x[0], x[1], True) for x in labels])
schema
```

Out[87]:
```
StructType(List(StructField(p_id,StringType,true),StructField(name,StringType,true),StructField(price,DoubleType,true),Struc
tField(colour,StringType,true),StructField(brand,StringType,true),StructField(ratingCount,IntegerType,true),StructField(avg_
rating,DoubleType,true),StructField(description,StringType,true),StructField(p_attributes,StringType,true),StructField(brand
_id,IntegerType,true),StructField(brand_name,StringType,true)))
```

Slide 22: fashionspark_df columns were inspected

Some cleaning of the fashionspark_df dataframe was done by renaming the columns using pyspark's .withColumnRenamed() method. The columns "p_id", "ratingCount", "avg_rating" and "p_attributes" were renamed to "product_id", "rating_count", "average_rating" and "product_attributes" respectively.

```
In [91]: # Renamed the columns

fashionspark_df = fashionspark_df.withColumnRenamed('p_id', 'product_id') \
        .withColumnRenamed('ratingCount', 'rating_count') \
        .withColumnRenamed('avg_rating', 'average_rating') \
        .withColumnRenamed('p_attributes', 'product_attributes')
fashionspark_df.show(truncate=True)
```

```
+----------+--------------------+------+-------+----------------+------------+--------------+--------------------+------
--------------+--------+--------------------+
|product_id|                name| price| colour|           brand|rating_count|average_rating|         description|  prod
uct_attributes|brand_id|          brand_name|
+----------+--------------------+------+-------+----------------+------------+--------------+--------------------+------
--------------+--------+--------------------+
|   1518329|Dupatta Bazaar Wh...| 899.0|  White|   Dupatta Bazaar|        1321|          4.55|White embroidered...|{'Occa
sion': 'Dai...|     242|      Dupatta Bazaar|
|   5829334|Roadster Women Mu...|1199.0|Mustard|        Roadster|        5462|          4.31|"Mustard yellow s...|{'Body
Shape ID':...|     750|            Roadster|
|  10340119|Inddus Peach-Colo...|5799.0|  Peach|          Inddus|         145|          4.07|Peach-Coloured an...|{'Bott
om Fabric':...|     389|              Inddus|
|  12384822|Kotty Women Black...|1999.0|  Black|           Kotty|       12260|          4.08|"Black dark wash ...|{'Add-
Ons': 'NA',...|     482|               Kotty|
|  14021452|Sera Women Multic...|1494.0|  Multi|            Sera|         750|          4.29|Brown and blue pr...|{'Body
or Garment...|     793|                Sera|
|  14063026|Tokyo Talkies Wom...| 699.0|  Black|   Tokyo Talkies|        1856|          4.53|"Black solid mid-...|{'Body
or Garment...|     903|       Tokyo Talkies|
|  14324806|Anouk Stylish Bla...|4699.0|  Black|           Anouk|          84|          3.81|Stay fashionable ...|{'Blou
```

Slide 23: fashionspark_df columns were renamed

The column “brand” was dropped as it was a similar column to “brand_name” and “product_attributes” was dropped as it was not required. Multiple filtering for white coloured clothes with an average rating above 2.50 was carried out and the output shown.

```
In [93]: # Dropped brand and product attributes columns

fashionspark_df = fashionspark_df.drop('brand') \
       .drop('product_attributes')
fashionspark_df.show(5,truncate=True)
```

```
+----------+--------------------+------+-------+------------+--------------+--------------------+--------+--------------+
|product_id|                name| price| colour|rating_count|average_rating|         description|brand_id|    brand_name|
+----------+--------------------+------+-------+------------+--------------+--------------------+--------+--------------+
|   1518329|Dupatta Bazaar Wh...| 899.0|  White|        1321|          4.55|White embroidered...|     242|Dupatta Bazaar|
|   5829334|Roadster Women Mu...|1199.0|Mustard|        5462|          4.31|"Mustard yellow s...|     750|      Roadster|
|  10340119|Inddus Peach-Colo...|5799.0|  Peach|         145|          4.07|Peach-Coloured an...|     389|        Inddus|
|  12384822|Kotty Women Black...|1999.0|  Black|       12260|          4.08|"Black dark wash ...|     482|         Kotty|
|  14021452|Sera Women Multic...|1494.0|  Multi|         750|          4.29|Brown and blue pr...|     793|          Sera|
+----------+--------------------+------+-------+------------+--------------+--------------------+--------+--------------+
only showing top 5 rows
```

```
In [94]: from pyspark.sql.functions import *
```

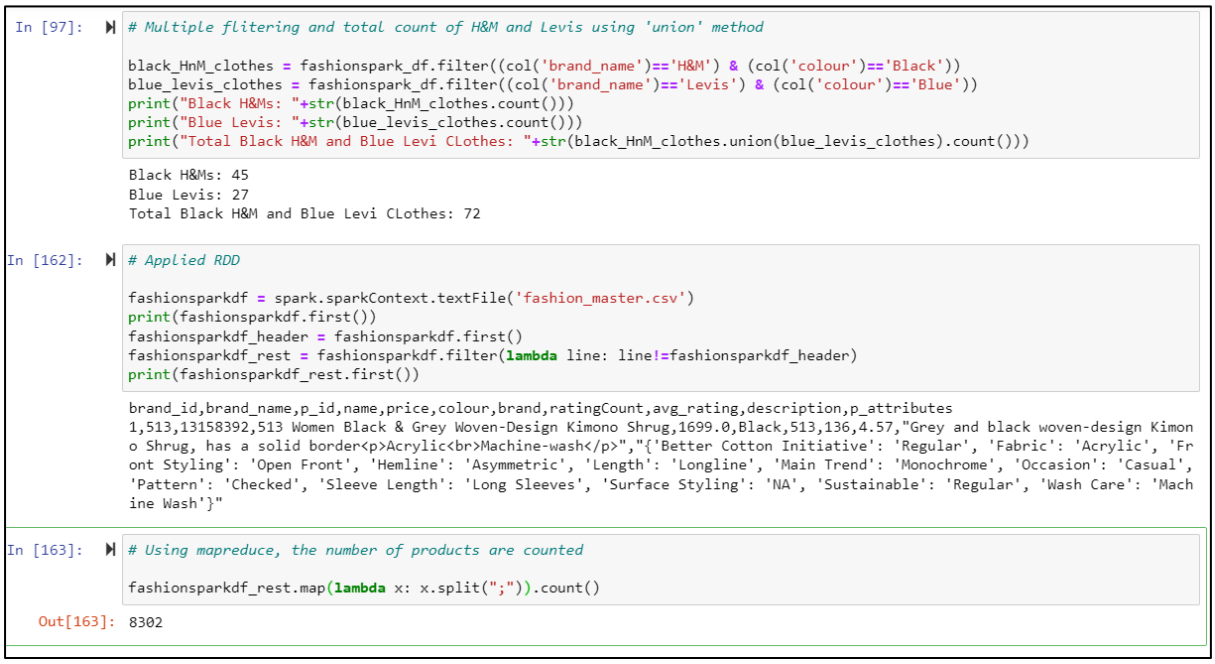
```
In [95]: # Filtered rows based on colour and avergae rating conditions

fashionspark_df.filter((col('colour')=='White') & (col('average_rating')>=2.50)).show(truncate=True)
```

```
+----------+--------------------+------+------+------------+--------------+--------------------+--------+--------------+
|product_id|                name| price|colour|rating_count|average_rating|         description|brand_id|    brand_name|
+----------+--------------------+------+------+------------+--------------+--------------------+--------+--------------+
|   1518329|Dupatta Bazaar Wh...| 899.0| White|        1321|          4.55|White embroidered...|     242|Dupatta Bazaar|
|  11697268|Roadster Women Wh...| 499.0| White|        3106|           4.3|"White printed de...|     750|      Roadster|
|  10711448|Dupatta Bazaar Wo...| 599.0| White|        1531|          4.54|White solid dupat...|     242|Dupatta Bazaar|
|  19147754|H&M Women White S...|1499.0| White|         184|           4.1|"White solid a-li...|     353|           H&M|
|  13736998|Athena White Tie-...|1699.0| White|        2200|          4.19|<ul><li>Colour: w...|      95|        Athena|
|  10182385|Style Quotient Wo...|1599.0| White|        1559|          4.43|White self design...|     845|Style Quotient|
|  11054006|Jaipur Kurti Wome...|1149.0| White|        6590|          4.06|"White solid mid-...|     418|  Jaipur Kurti|
|  17124538|H&M Women White R...|1499.0| White|          86|          4.66|"<p>Cardigan in a...|     353|           H&M|
|  17819832|Mast & Harbour Wh...|1599.0| White|         184|           4.1|<ul> <li> White a...|     554|Mast & Harbour|
```

Slide 24: fashionspark_df was filtered for white coloured products with an average rating not less than 2.50

Investigating the total number of black H&M clothes and blue Levis clothes was done with more multiple filtering and the count() function. This showed how fashion enthusiasts can, with the help of pyspark, select products across multiple brands and designs to make their selections and analyses.

Finally, on pyspark, the map reduce RDD concept was applied to inspect the headers and first row of the dataframe. Count was also done using this concept.



```
In [97]:
# Multiple flitering and total count of H&M and Levis using 'union' method

black_HnM_clothes = fashionspark_df.filter((col('brand_name')=='H&M') & (col('colour')=='Black'))
blue_levis_clothes = fashionspark_df.filter((col('brand_name')=='Levis') & (col('colour')=='Blue'))
print("Black H&Ms: "+str(black_HnM_clothes.count()))
print("Blue Levis: "+str(blue_levis_clothes.count()))
print("Total Black H&M and Blue Levi CLothes: "+str(black_HnM_clothes.union(blue_levis_clothes).count()))
```

```
Black H&Ms: 45
Blue Levis: 27
Total Black H&M and Blue Levi CLothes: 72
```

```
In [162]:
# Applied RDD

fashionsparkdf = spark.sparkContext.textFile('fashion_master.csv')
print(fashionsparkdf.first())
fashionsparkdf_header = fashionsparkdf.first()
fashionsparkdf_rest = fashionsparkdf.filter(lambda line: line!=fashionsparkdf_header)
print(fashionsparkdf_rest.first())
```

```
brand_id,brand_name,p_id,name,price,colour,brand,ratingCount,avg_rating,description,p_attributes
1,513,13158392,513 Women Black & Grey Woven-Design Kimono Shrug,1699.0,Black,513,136,4.57,"Grey and black woven-design Kimon
o Shrug, has a solid border<p>Acrylic<br>Machine-wash</p>","{'Better Cotton Initiative': 'Regular', 'Fabric': 'Acrylic', 'Fr
ont Styling': 'Open Front', 'Hemline': 'Asymmetric', 'Length': 'Longline', 'Main Trend': 'Monochrome', 'Occasion': 'Casual',
'Pattern': 'Checked', 'Sleeve Length': 'Long Sleeves', 'Surface Styling': 'NA', 'Sustainable': 'Regular', 'Wash Care': 'Mach
ine Wash'}"
```

```
In [163]:
# Using mapreduce, the number of products are counted

fashionsparkdf_rest.map(lambda x: x.split(";")).count()
```

```
Out[163]: 8302
```

Slide 25: fashionspark_df was filtered for multiple brands and colours, then counted; mapreduce RDD concept was applied and used to count

2.5 Machine Learning

Key to machine learning in python was importing the sklearn library and all its relevant functions. The “black_prdts_df” dataframe was further filtered to drop all “ratingCount” entries with zero as entries. The resulting dataframe focus_1 was used for the machine learning focusing on the “avg_rating”, “ratingCount” and “price” fields.

Machine Learning

```
In [100]:
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
In [101]:
# Used the black products dataframe

black_prdts_df.head()
```

Out[101]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes	brand_id	brand_name
0	16920516	Masaba Woman Black Tulip Cape Set	30000.0	Black	Masaba	184	4.10	V-neckline Full sleeves Embr...	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...	553	Masaba
1	18585472	Mitera Black & Pink Floral Pure Linen Saree	22300.0	Black	Mitera	184	4.10	Design Details Black and pi...	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...	583	Mitera
2	13168738	Justanned Plus Women Plus Size Black Solid Lea...	19999.0	Black	Justanned	184	4.10	Black solid jacket, has a spread collar, 6 poc...	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...	442	Justanned
3	17981980	Justanned Women Black Leather Lightweight Crop...	19998.0	Black	Justanned	184	4.10	Black solid lightweight biker jacket with zip...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	442	Justanned
4	15880878	Justanned Women Black Striped Leather Crop Out...	15998.0	Black	Justanned	6	4.67	Black solid striped biker jacket with zip det...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	442	Justanned

```
In [102]:
# Filtered further by dropping all rows with rating count equalling zero

black_prdts_mldf = black_prdts_df[black_prdts_df['ratingCount'] != 0]
```

```
In [103]:
black_prdts_mldf.info()
```

Slide 26: All relevant sklearn imports were made and black_prdts_df filtered to black_prdts_mldf

```
In [104]: # Chose to focus the model on average rating, rating count and price

          focus_1 = black_prdts_mldf.loc[:,['avg_rating','ratingCount','price']]

In [105]: focus_1.head()
```

Out[105]:

	avg_rating	ratingCount	price
0	4.10	184	30000.0
1	4.10	184	22300.0
2	4.10	184	19999.0
3	4.10	184	19998.0
4	4.67	6	15998.0

```
In [106]: # Imported Normalizer in order to preprocess the focused data and bring them to scale

          from sklearn.preprocessing import Normalizer

          normalizer = Normalizer(norm='max')

In [107]: focus_1[['avg_rating', 'ratingCount', 'price']]  = normalizer.fit_transform(focus_1[['avg_rating', 'ratingCount', 'price']])

In [108]: focus_1.head()
```

Out[108]:

	avg_rating	ratingCount	price
0	0.000137	0.006133	1.0
1	0.000184	0.008251	1.0
2	0.000205	0.009200	1.0
3	0.000205	0.009201	1.0

Slide 27: focus_1 dataframe was created from black_prdts_mldf and Normalizer imported to pre-process the data

It must be noted, at this point, that a pre-processing tool Normalizer was imported to normalize the focus_1 df attributes to scale in order to get a more balanced prediction or output.

2.5.1 Supervised machine learning

I. Simple and multiple linear regression

Simple linear and multiple linear regressions were carried out with training a portion of the data with the train_test_split library. For the simple linear regression, the values were trained on X for “avg_rating” and y for “price”— with X indicating the independent variable and y the dependent variable or output. The test size parameter was 20% implying that 80% of the data would be used for training while 20% would be the test data. Prediction was done thereafter and the error values outputted. The model scores were also investigated.

The multiple linear regression method followed a similar pattern of training and output prediction, with the difference being that the input variable X had more than 1 parameter. X in this case referred to the “avg_rating” and “ratingCount” attributes.

Using linear regression

In [109]:
```
# Created the X and y dataframes

X = pd.DataFrame(focus_1['avg_rating'])
y = pd.DataFrame(focus_1['price'])
```

In [110]:
```
#Trained the model on X for avg_rating and y for price

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [111]:
```
X_train
```

...

In [112]:
```
# Assigned the model

model_LR = LinearRegression()
```

In [113]:
```
# Fitted the model to the training set

model_LR.fit(X_train, y_train)
```

Out[113]:
```
LinearRegression()
```

In [114]:
```
# Predicted using the test set

y_pred = model_LR.predict(X_test)
```

Slide 28: Simple linear regression model fitted to the trained data and output predicted

In [116]:
```
# Prediction of first 10 results

y_pred[0:10]
```

Out[116]:
```
array([[0.99252353],
       [0.99271392],
       [0.99264764],
       [0.99332   ],
       [0.99288113],
       [0.9921499 ],
       [0.99241951],
       [0.99256186],
       [0.99214501],
       [0.9928168 ]])
```

In [117]:
```
# Checked accuracy of the model

model_LR.score(X_test, y_test)
```

Out[117]:
```
0.0005899455605113957
```

In [118]:
```
# Explored the errors

print('Mean Absolute Error: ', mean_absolute_error(y_test, y_pred))
print('Mean Squared Error: ', mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error: ', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error:  0.014983112465987432
Mean Squared Error:  0.0037374819138062923
Root Mean Squared Error:  0.06113494838311628
```

Slide 29: Simple linear regression model score and error values

Using multiple regression

```
In [119]: # Used more independent variables for a multiple regression

          X1 = pd.DataFrame(focus_1[['avg_rating', 'ratingCount']])
          X1
```

```
In [120]: X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y, test_size=0.2, random_state=2)
```

```
In [121]: model_LR.fit(X1_train, y1_train)

Out[121]: LinearRegression()
```

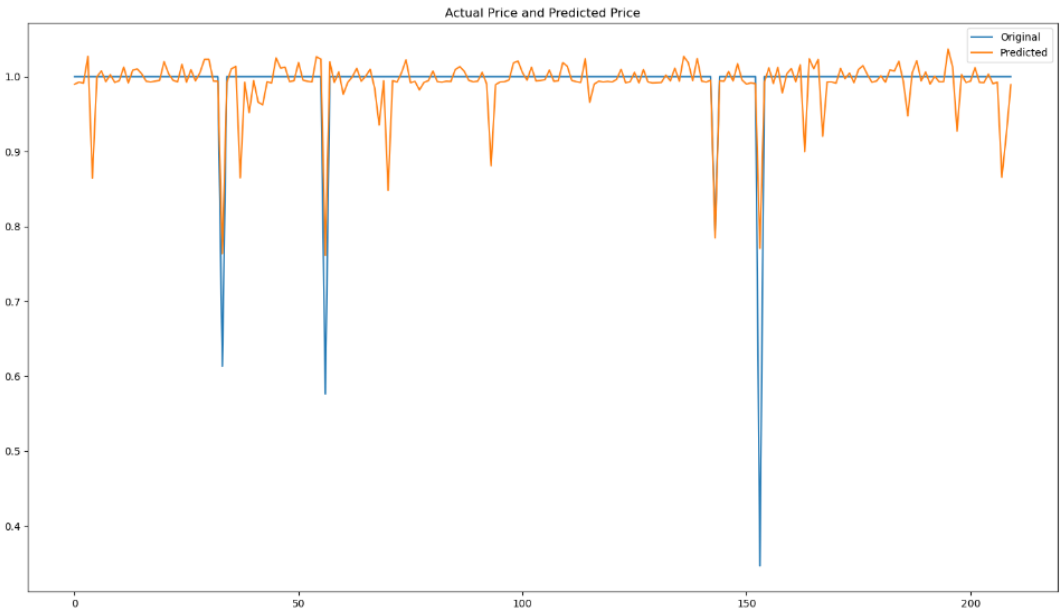
```
In [122]: y1_pred = model_LR.predict(X1_test)
```

```
In [123]: # Results using multiple regression

          y1_pred[0:10]

Out[123]: array([[0.98984365],
                 [0.99269703],
                 [0.99142294],
                 [1.02715041],
                 [0.86457251],
                 [0.9999197 ],
                 [1.0075999 ],
                 [0.99334251],
                 [1.00254201],
                 [0.99226031]])
```

```
In [124]: # In order to see the graphical corretation between actual and predicted outputs,set plots imported to be embedded inline

          import matplotlib.pyplot as plt
          import seaborn as sb

          %matplotlib inline
```

Slide 30: Multiple linear regression model fitted to the trained data and output predicted

```
In [125]: plt.figure(figsize=(18,10))
          x_axis=range(len(y1_test))
          plt.plot(x_axis, y1_test, label='Original')
          plt.plot(x_axis, y1_pred, label='Predicted')
          plt.title('Actual Price and Predicted Price')
          plt.legend()
          plt.show()
```



Slide 31: Multiple linear regression predicted and actual values plot

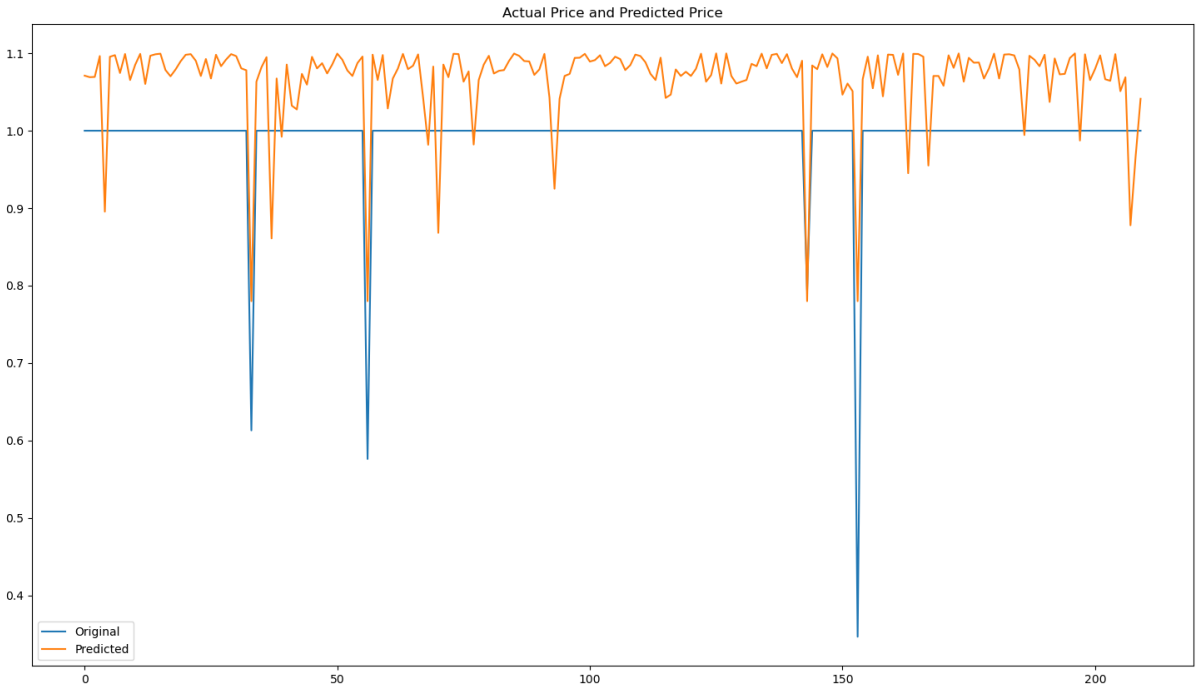
A graph depicting the correlation between the actual and predicted outcomes was plotted in order to depict more clearly the accuracy of the model deployed and the machine learning process adopted.

II. Support vector regression

Similar machine learning processes were carried out using support vector regression SVR. SVR is designed to find the hyperplane in a high dimensional space that maximally separates data points into their respective classes; and hence it is robust against outliers.

Its application procedures were similar to that of the earlier mentioned regression methods. SVR however displayed a lower model accuracy score.

The graph plotted showed a similarity with that of multiple regression but on closer inspection, the divergence of the actual output from the predicted output between the two was apparent.



Slide 32: Support vector regression predicted and actual values plot

2.5.2 Unsupervised machine learning: KMeans clustering

Due to the poor model scores of the earlier machine learning techniques, further machine learning was done to explore the grouping of the data based on similar characteristics in pricing and average rating; as such, K-Means clustering machine learning method was adopted.

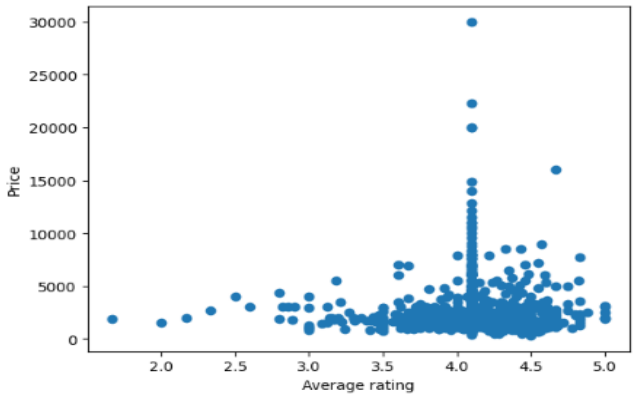
Using KMeans Clustering

```
In [137]: # Imported KMeans and MinMaxScaler from the sklearn library

from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
```

```
In [138]: # Filtered for the attributes to be analysed using KMeans

focus_2 = black_prdts_mldf.loc[:,['avg_rating','price']]
```

```
In [139]: # Reasserted attributes in focus

rating = focus_2['avg_rating']
pricing = focus_2['price']
```

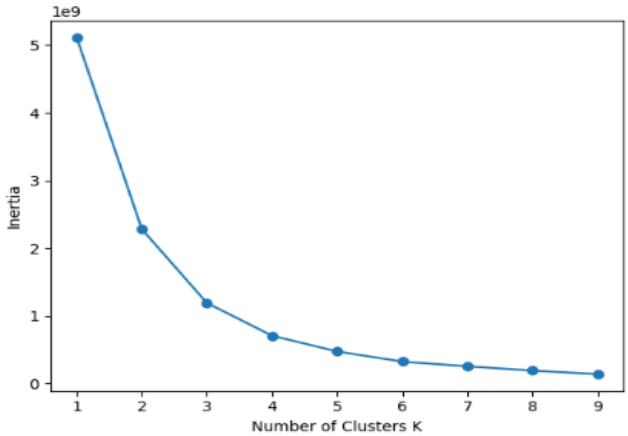
```
In [140]: # Examined the data points

plt.scatter(rating, pricing)
plt.xlabel('Average rating')
plt.ylabel('Price');
```



Slide 33: Relevant libraries were imported and price plotted against average rating from black products dataframe

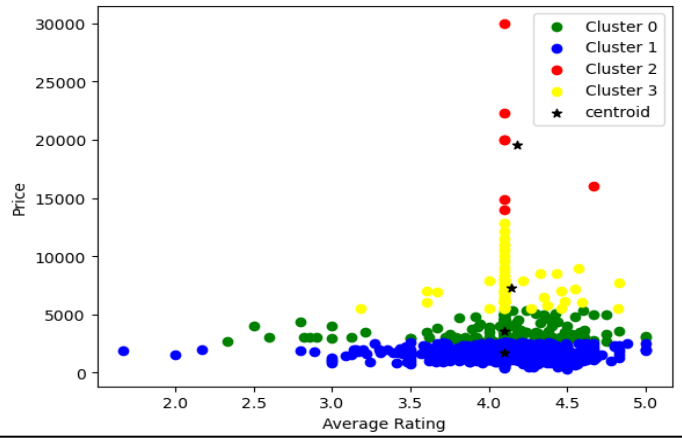
The elbow point was first calculated by plotting a graph of inertia sse vs number of clusters k to determine the point from which inertia decreased more slowly as k increased. 4 was decided as the number of clusters to be used(elbow point) and hence KMeans was instantiated and fitted to the dataframe. The output and centroids were determined and the datapoints plotted to depict their cluster.

```
In [142]: # Plotted inertia vs number of clusters to better determine the elbow point

plt.xlabel('Number of Clusters K')
plt.ylabel('Inertia')
plt.plot(k_range, sse, '-o');
```



Slide 34: Inertia plotted against number of clusters k

In [148]:
```
# Dataframes were created and plotted with regards to their cluster grouping of 0,1,2 or 3

focus_2_c0= focus_2[focus_2.cluster==0]
focus_2_c1= focus_2[focus_2.cluster==1]
focus_2_c2= focus_2[focus_2.cluster==2]
focus_2_c3= focus_2[focus_2.cluster==3]

plt.scatter(focus_2_c0['avg_rating'], focus_2_c0['price'], label='Cluster 0', color='green')
plt.scatter(focus_2_c1['avg_rating'], focus_2_c1['price'], label='Cluster 1',color='blue')
plt.scatter(focus_2_c2['avg_rating'], focus_2_c2['price'], label='Cluster 2',color='red')
plt.scatter(focus_2_c3['avg_rating'], focus_2_c3['price'], label='Cluster 3',color='yellow')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1],color='black', marker='*', label ='centroid')


plt.xlabel('Average Rating')
plt.ylabel('Price')
plt.legend();
```



Slide 35: Cluster data points and centroids plotted

For more accurate clustering, MinMaxScaler was imported and applied on the average rating and price entries. Updated outputs, centroids and cluster data points were determined after the range scaling and plotted.

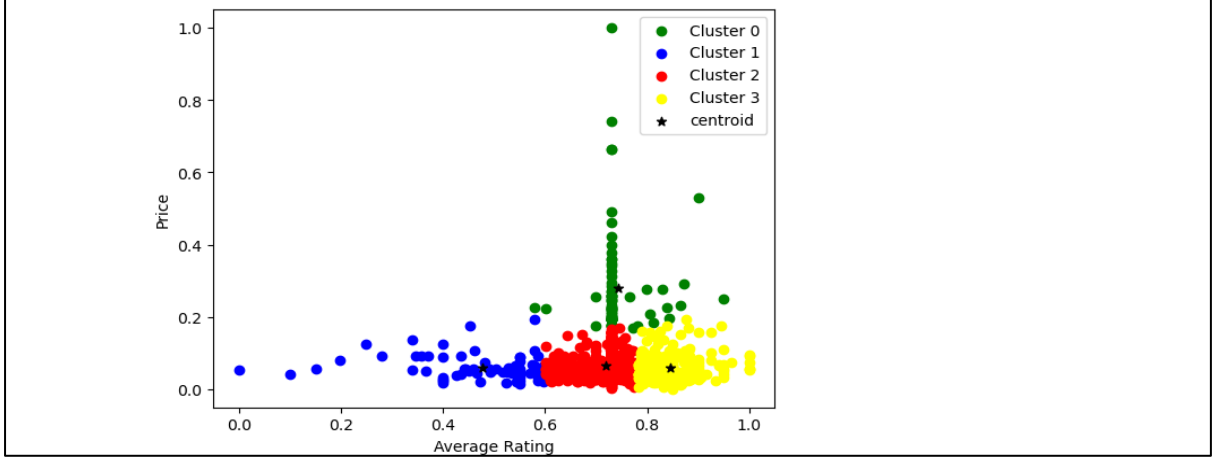
In [149]:
```
# MinMaxScaler was used to preprocess the price and average rating entries to a range scale for more accurate output

scaler=MinMaxScaler()
scaler.fit(focus_2[['price']])
focus_2[['price']]= scaler.transform(focus_2[['price']])

scaler.fit(focus_2[['avg_rating']])
focus_2[['avg_rating']]= scaler.transform(focus_2[['avg_rating']])

focus_2
```

Out[149]:

	avg_rating	price	cluster
0	0.729730	1.000000	2
1	0.729730	0.740784	2
2	0.729730	0.663323	2
3	0.729730	0.663289	2
4	0.900901	0.528632	2
...	...	...	...

Slide 36: MinMaxScaler() applied to the data

```
In [154]:
# Dataframe was sliced and plotted with regards to the updated cluster groupings

focus_2_c0= focus_2[focus_2.New_cluster==0]
focus_2_c1= focus_2[focus_2.New_cluster==1]
focus_2_c2= focus_2[focus_2.New_cluster==2]
focus_2_c3= focus_2[focus_2.New_cluster==3]

plt.scatter(focus_2_c0['avg_rating'], focus_2_c0['price'], label='Cluster 0', color='green')
plt.scatter(focus_2_c1['avg_rating'], focus_2_c1['price'], label='Cluster 1',color='blue')
plt.scatter(focus_2_c2['avg_rating'], focus_2_c2['price'], label='Cluster 2',color='red')
plt.scatter(focus_2_c3['avg_rating'], focus_2_c3['price'], label='Cluster 3',color='yellow')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1],color='black', marker='*', label ='centroid')


plt.xlabel('Average Rating')
plt.ylabel('Price')
plt.legend();
```



Slide 37: Updated cluster data points and centroids plotted

Based on the KMeans cluster analysis, it was observed that most products were above average and had an affordable price, both of which are good for the ever-growing fashion industry. The clusters were largely divided into those with medium average rating and low price, above medium average rating and low price, high average rating and low price and above medium to high average rating and a higher price. Based on the cluster a product fell into, fashion brands could determine the changes required to meet the needs of their shareholders and consumers.

2.6 Visualization

The review_prdts_df where the poorly rated products were examined was used here to determine, visually, the relationship between price and average rating. The colours of the products were taken into consideration using the hue parameter of seaborn's scatterplot.

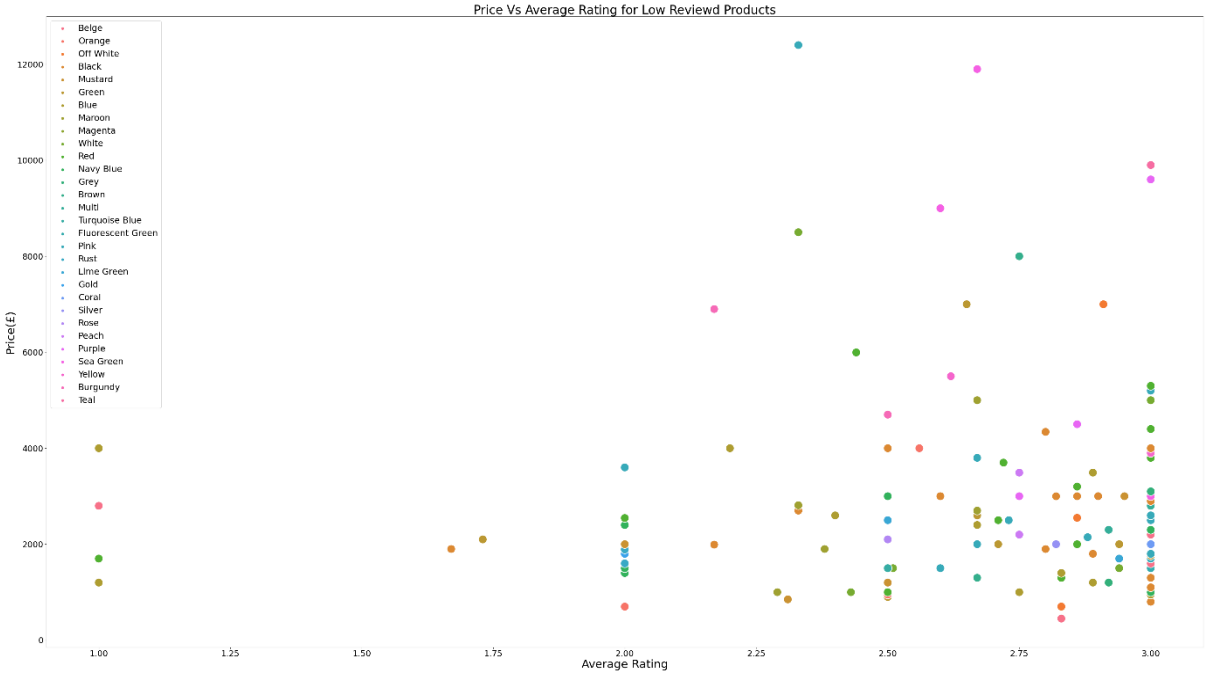
Seaborn and its scatterplot function were used for the visualization. X and y parameters were set, matplotlib was used to set labels, titles and legend, as well as the figure size. Finally, the scatterplot graph was plotted.

```
In [160]:
# Scatter plot done with price against average rating

sb.scatterplot(data=review_prdts_df, x="avg_rating", y="price", hue=review_prdts_df['colour'], s=500)

plt.ylabel('Price(£)', fontsize=30)
plt.xlabel('Average Rating', fontsize=30)
plt.legend(fontsize=23);

plt.xticks(fontsize=22, rotation = 0);
plt.yticks(fontsize=22);

plt.title('Price Vs Average Rating for Low Reviewd Products', fontsize=32)

from matplotlib import rcParams
rcParams['figure.figsize'] = 55,30
```

Slide 38: Seaborn scatterplot was used for price against average rating plot with data as review_prdts_df

The visualization showed that most poorly reviewed products were actually above rating 2.0 and had pricing around £2,000. However, there were noticeable outliers throughout.

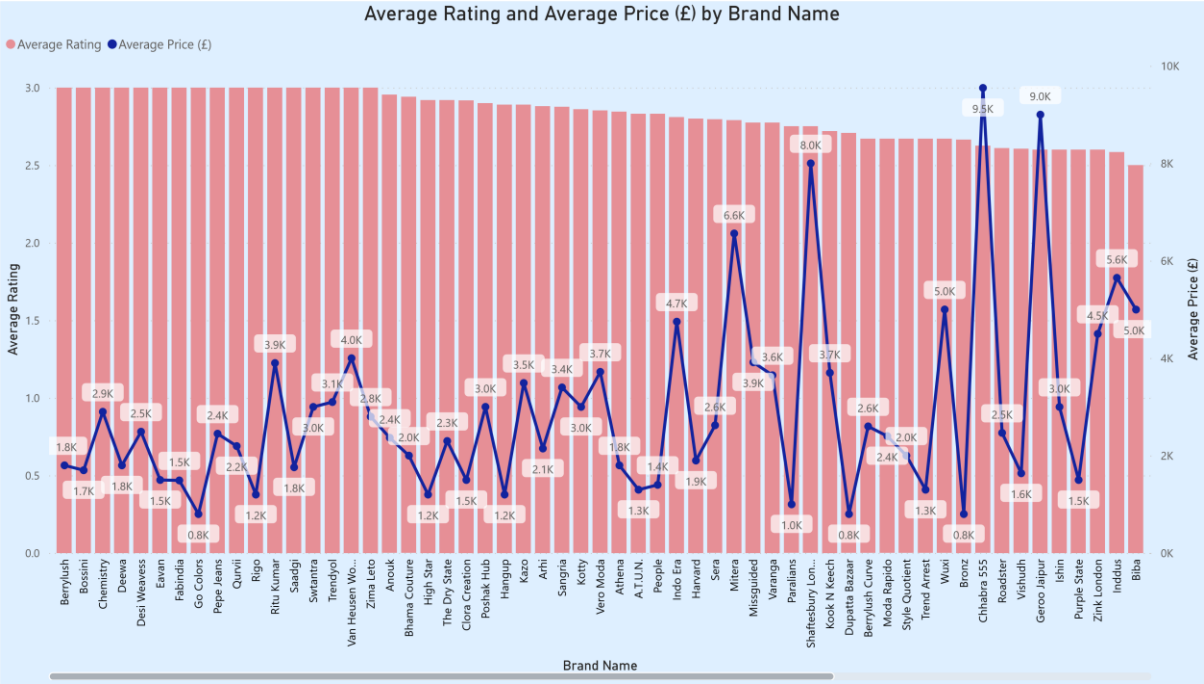
a:Python Visualization



Plot 1: Poorly reviewed products review_prdts_df plot of Price(£) Vs Average Rating

Hence, fashion brands can use these visuals to easily determine average ratings per colour and the relationship between average rating and price of products. They could also determine the products in most need of improvement and see where discounts can be given in order to improve the appeal of select products.

b:Power BI Visualization



Plot 2: Poorly reviewed products plot of Average Rating and Average Price(£) by Brand Name

Finally, on visualization, the review_prdts_df dataframe was exported to a .csv file "reviewprdts.csv" which was analysed and displayed via Power BI. The Power BI display shows a line and stacked column chart with the average rating by brand displayed with columns while the average prices were displayed with the line graph.

This comprehensive visual could provide the framework for quick decision making and trend analysis for fashion enthusiasts and the larger fashion retail brands.

3. ISSUES IN THE USE OF BIG DATA ANALYTICS IN THE FASHION RETAIL INDUSTRY

The issues facing the fashion industry in their application of big data analytics can be summarized in the following questions:

a) Are there sufficient, accurate and ethical data collection pipelines in the industry?
b) Are environmental, climate change, societal and cultural factors considered in fashion data analytics?

4. METHODOLOGY

The study used the qualitative literature review methodology for preparing this report. This involved analyses and reviews of 15 papers referenced in this report using relevant keywords in the selection process. Data was extracted from Google Scholar databases amongst other sources with particular attention being paid to refereed academic journals and academic books. In addition, exploratory analyses into today's fashion brands, their prices, customer ratings and product attributes were carried out, using real life datasets, in order to learn more about the industry, filter for on-demand products and carry out machine learning tactics for predictive analysis. Jupyter Notebook was the principal notebook applied during this process, wherein MongoDB was imported as PyMongo and Spark as PySpark. Sqlite3 was imported for database filtering and management; and Sklearn for machine learning. While attempting to use Seaborn and Matplotlib, the number of data entries made it more convenient to adopt Power BI for our visualization.

It must be noted, at this point, the limited resources that were available regarding the issues with fashion data analytics. Results from both Scopus and Google Scholar mostly pointed to the challenges of big data analytics in general. The application of big data to fashion retail was broadly seen as a net good with issues discovered at any step of the research largely minimised. The search strings included keywords "fashion retail", "consumer experience", "trend forecasting", "big data", "AI", "Internet of Things", "machine learning". Hence, an inductive reasoning approach was often used to spot the issues with fashion big data, in addition to the qualitative review of academic journals and books.

Furthermore, the study was largely approached from the basis of the 4th industrial revolution and big data explosion; and how the fashion industry has woken up to big data application to its core business. The issues with this broad application, however, cannot remain ignored.

5. ISSUES IDENTIFIED AND DISCUSSED

5.1 Are There Sufficient, Accurate and Ethical Data Collection Pipelines in the Industry?

I. Discussion

Research has shown that only about 23% of medical data registered in hospitals in the US align with what patients actually put down in registration. While this is a different sector than fashion, it raises fears on the accuracy of the information getting through from varying sources. Online shopping platforms use cookies to gather information on consumers and recommend products based on these data. However, many factors here can lead to inaccurate data collection. For instance, a person might regularly choose to shop for a relative or friend through his profile, and data gathered on given customer would largely be that of the relative or friend and not the consumer himself. Moreover, consumers largely are not interested in giving feedback or participating in surveys and simply want a transaction to end once purchase has been made. This leads most consumers to hurriedly, and sometimes inaccurately, fill surveys when prompted to. All these promote the trend of inaccurate data collection; and due to the speed of modern commerce, little time is taken to sift through to determine levels of accuracy.

A segue from here into ethics is proper as large amounts of consumer data are often collected with minimal consent or understanding by the customer. Most customers simply wish to make a purchase and do not take the time to go through the "Terms and Conditions" required during registration to online shopping sites. Hence, do online marketers and fashion brands have a right to gather customers' personal information just because they want to purchase a product? Do customers want to be guided to making purchases? There are forms of unwarranted paternalism displayed in brands taking the responsibility and guiding customers on products to purchase and trends to follow.

On the other hand, large consumer demographics that are less internet savvy occasionally are left out of the fashion big data boom. While the Internet of Things has made goods and services more accessible, no account has been taken of the older generation who are not inclined to be active on the internet (Ahmed et al., 2017). Third world countries have also been largely ignored by top fashion brands due to poor infrastructure, amongst other issues. Consequently, there is insufficient data coverage of this demographic of individuals, as well as potential customers of certain geographical locations.

II. Solution

Fashion companies have, not only an ethical, but also a fiduciary responsibility to gather accurate and sufficient data, while keeping in mind the customer's need for privacy and transparency. Irrespective of several countries' adoption of General Data Protection Regulations broadly, responsibilities lie with fashion houses to implement big data technologies that are congruent with handling and managing big data. Technologies

such as SQL and NoSQL for structured and unstructured database management can be adopted to ensure data collected is appropriately filtered to meet customer need and give a more personalized service, while maintaining privacy. Also, Apache Kafka can be adopted for video streaming advertisement of products to consumers (Chandrakant, 2022).

In a bid to mitigate the ethical risk of exclusion and ensure accurate and encompassing data analysis (Drosou et al., 1970), certain big data technologies can be employed. Cloud computing services such as AWS, Azure and the Internet of Things can be broadened to include all demographics by reducing the requirements for its access e.g., online forms can be shortened to gather only essential information. A result of this could be the Internet of Things being extended to the Internet of Fashion, which would permit for already filled online forms to be moved easily across the different fashion brands' websites. This would imply a network of fashion where customers' information and trends are linked resulting in customer needs more easily met.

Informatica Data Integration software can be used to Extract, Transform and Load raw data from data sources into data warehouses which can be sifted to the appropriate data marts for further analysis. This is in a bid to encourage personalized customer products (Banik, 2021) and promote predictive analysis of fashion trends.

As fashion retail companies embark on their big data journey, adopting the appropriate technology and methods will reduce cases of returned items, promote trend prediction, reduce resources wastage, boost personalized customer experience and overall grow the profit bottom line of these companies (Aloysius et al., 2016).

5.2 Are Environmental, Climate Change, Societal and Cultural Factors Considered in Fashion Data Analytics?

I. Discussion

The world of fashion is currently worth over $2.5 trillion globally and employs over 75 million people in its value chain. This implies that there are vast production and value chain costs to get the products from the manufacturer to the final consumer (Kharde, 2022). Big fashion brands being able to partake in predictive trend analysis with the aid of big data tools, engage in mass production of these products implying the use of heavy machinery and fuel. The cost to the environment is inevitable and far-reaching as, besides production, logistics and transportation of these products contribute to carbon emissions and pollution (Barroca & Andrade, 2021). According to the World Bank, "the fashion industry is responsible for 10% of annual global carbon emissions, more than all international flights and maritime shipping combined" (World Bank Group, 2019). The fashion industry has also been touted to be the second most polluting industry in the world. Discouragingly, top brands often try to hide these emission numbers and pollution stats (Donald, 2022).

From climate change to cultural change, fashion and the industry are core to this aspect of society. It is said that fashion captures the zeitgeist of the times, be it in decades or even centuries (Kristy, 2021). Summarily, fashion is downstream from culture. The fashion trends of an era often reflect the prominent school of thought of that era. However, there has been a shift in the influence of fashion as the industry has embraced big data. Predictive analysis and machine learning have moved the goal post of fashion as a reflection of the times to fashion as a director of culture. Global brands backed by big data now have the power to influence culture. Certain trends can be promoted and fed to the customer to subconsciously direct customer choice and influence culture. Big global brands can also leverage on social media, advertisements, "fashion icons" and celebrities to promote these trends (Assoune, 2021). The 21st century for instance has seen the promotion of less conservative clothing trends; and has even led to these fashion trends being exported to the more conservative developing countries. This implies the paternalism that can ensue with the marriage of big global brands with huge amounts of customer data.

II. Solution

Data pipelines have to include pollution and climate change costs to their parameters. With how large the fashion industry has become in terms of revenue, companies must incur the costs required for limiting their carbon emissions and reducing pollution (Cernansky, 2019). Standardization of data pipeline to ensure these costs are considered should be looked into by countries and governing bodies where these fashion companies operate. Furthermore, the fashion industry should be wary of negatively impacting the culture and moral inclinations of its customers. They should make efforts to remain a reflection of the customers that patronize them; and ensure they pay attention to customer needs in lieu of their own goals and benefits. In line with this, Natural Language Processing (NLP) technology should be broadly adopted by the fashion retail industry to extricate customer feedback and reviews from large blocks of unstructured data. Predictive and machine learning algorithms can also be adopted in line with customer and cultural wants, as well as data visualization tools like Power BI to show clear visualizations of customer reviews and sociocultural impacts of fashion products.

Table 1: Select Fashion Big Data Technologies and The Solutions Offered

Big Data Technology	**Solution**
SQL and NoSQL	Manages large and diverse, structured and unstructured fashion data to better ensure customer feedback is extracted
Apache Kafka	Used to stream product information and adverts
Cloud Computing	Provides scalable and more expansive reach to customers
ETL (Extract, Transform, Load) Technology like	Extracts, transforms and loads raw fashion data to data warehouses for further analysis

Informatica Data Integration	
Natural Language Processing (NLP)	Used to extract customer reviews from large datasets
Machine Learning	Learns, evolves and predicts customer trends and facilitates personalized services
Data Visualization (Power BI, Tableau, QlikView)	Presents clear trends, patterns, analysis and insights through visuals and aids fashion retailers in problem solving.

6. CONCLUSION

The results of the study show the enormous advantages and huge influence of big data analytics to the fashion retail industry. With the large revenues generated from fashion, it has become imperative for fashion companies to remain competitive and leverage on the assets big data analytics provide. These companies can now see the bigger picture on how their products are faring based on customer rating, pricing and even product colour. Predictive analysis using machine learning has led to improved customer service, quicker decision making and better resources management, all greatly improving profit margins.

These fashion houses, however, should not ignore the impact of pollution and climate change that can come from such a large industry. Efforts should be taken for environment improvement and reinvesting revenues to stem pollution. Also, customer privacy must be respected and countries should come in at this point to ensure well-supervised privacy policies. The companies could also massively grow their profits by making their online platforms easier to use and access across all demographics and geographical locations. Finally, large industries like the fashion industry must be consistently and ethically aware of the culture and beliefs of their various customers and must take steps to ensure they reflect, and not dictate, these cultures.

This study adopted a qualitative approach where previous literature on big data and artificial intelligence were analysed. Two fashion datasets, which were wrangled with different libraries and imported tools were also used. The research was thus based on literature review where various artificial intelligence and big data papers were critically evaluated in the context of the fashion retail industry.

References

Ahmed, E. *et al.* (2017) "The role of Big Data Analytics in internet of things," *Computer Networks*, 129, pp. 459–471. Available at: https://doi.org/10.1016/j.comnet.2017.06.013.

Aloysius, J.A. *et al.* (2016) "Big Data Initiatives in retail environments: Linking service process perceptions to shopping outcomes," *Annals of Operations Research*, 270(1-2), pp. 25–51. Available at: https://doi.org/10.1007/s10479-016-2276-3.

Assoune, A. (2021) *Fast fashion social impacts and how it affects society*, *Panaprium*. Panaprium. Available at: https://www.panaprium.com/blogs/i/fast-fashion-society.

Banik, A. (2021) *Application of big data in the Fashion Industry*, *Analytics Insight*. Available at: https://www.analyticsinsight.net/application-of-big-data-in-the-fashion-industry.

Barroca, G. and Andrade, M. (2021) *Fast fashion: What we are not seeing*, *Increasing awareness and knowledge with our economics newspaper*. Available at: https://theawarenessnews.com/2021/03/25/fast-fashion-what-we-are-not-seeing.

Cernansky, R. (2019) *Only two big brands do enough to fight climate change, report claims*, *Vogue Business*. Available at: https://www.voguebusiness.com/sustainability/fashion-climate-change-sustainability-standearth-paris-agreement (Accessed: January 5, 2023).

Chandrakant, K. (2022) *Building a data pipeline with Kafka, Spark Streaming and cassandra*, *Baeldung*. Available at: https://www.baeldung.com/kafka-spark-data-pipeline.

Donald, R. (2022) *Why 'eco-conscious' fashion brands can continue to increase emissions*, *The Guardian*. Guardian News and Media. Available at: https://www.theguardian.com/environment/2022/apr/09/why-eco-conscious-fashion-brands-can-continue-to-increase-emissions (Accessed: January 5, 2023).

Drosou, M. *et al.* (1970) *[PDF] diversity in big data: A review: Semantic scholar*, *Big data*. Available at: https://www.semanticscholar.org/paper/Diversity-in-Big-Data%3A-A-Review-Drosou-Jagadish.

Jain, S. *et al.* (2017) *IOPscience*, *IOP Conference Series: Materials Science and Engineering*. IOP Publishing. Available at: https://iopscience.iop.org/article/10.1088/1757-899X/254/15/152005.

Kharde, A. (2022) *Global clothing market size to reach USD 2,230.07 billion in 2028*, *Textile Focus*. Available at: https://textilefocus.com/global-clothing-market-size-to-reach-usd-2230-07-billion-in-2028.

Kristy (2021) *How the Fashion Industry contributes to Society & Societal Change*, *TPS Blog*. Available at: https://www.thepearlsource.com/blog/how-fashion-industry-contributes-to-society-change.

Silva, E.S., Hassani, H. and Madsen, D.Ø. (2019) "Big Data In Fashion: Transforming the retail sector," *Journal of Business Strategy*, 41(4), pp. 21–27. Available at: https://doi.org/10.1108/jbs-04-2019-0062.

Thomassey, S. (2018) *Artificial Intelligence for fashion industry in the Big Data Era*, *Google Books*. Springer. Available at: https://books.google.com/books/about/Artificial_Intelligence_for_Fashion_Indu.html?id=9QNbDwAAQBAJ.

World Bank Group (2019) *How much do our wardrobes cost to the environment? World Bank*. World Bank Group. Available at: https://www.worldbank.org/en/news/feature/2019/09/23/costo-moda-medio-ambiente.

Appendix

Below is the complete data analysis process, with respect to the provided datasets, which was done on Jupyter Notebook, downloaded, saved as a pdf file and added to this report

The raw python notebook document DAT7015_Project2_complete.ipynb is accessible on GitHub at https://github.com/IkenChuks.

Global Fashion Brands: An Exploratory Analysis

by Ikenna Chukwudum

Introduction

> For the analysis of the fashion brands, we used 2 datasets: "fashion_brand_details.xlsx" with 1020 entries and 2 fields; and "fashion_dataset.csv" with 14329 entries and 9 columns. The analysis was started by merging these 2 datasets after importing them as dataframes into the notebook.
>
> The new dataframe, after cleaning with python and merging with SQL, was explored to answer the questions on if poorly rated products should be discarded and whether black clothes are more highly rated and pricier than clothes of other colours. Also, attributes of the different products were examined and predictive analysis was carried out on the price of the commodities using machine learning.
>
> Further analysis and filtering was done with Spark, including map reduce. Thereafter, visualization to show the correlation between price and average rating was carried out with both seaborn and Power BI.

Preliminary Wrangling

In [1]:
```
# Imported numpy and pandas packages for cleaning

import numpy as np
import pandas as pd
```

> The analysis was started by reading the datasets "fashion_dataset.csv" and "fashion_brand_details.xlsx" to the dataframes design1_df and design2_df respectively, and thereafter investigating them

In [2]:
```
# The os library was imported to ensure datasets could be located in the directory
import os
wd = os.getcwd()
files = os.listdir(wd)  # Get all the files in that directory
list(files)
```

Out[2]:
```
['.ipynb_checkpoints',
 'ALX PROJECTS',
 'bigdata digram.pbix',
 'bigdataassignment_2.ipynb',
 'bigdataassignment_2.pdf',
 'blackaprdts.csv',
 'blackbprdts.csv',
 'blackcprdts.csv',
 'blackcprdtsattr.csv',
 'blackprdts.csv',
 'blackxprdts.csv',
 'cars.csv',
 'cars.csv.1',
```

```
'cars.csv.2',
'cars.csv.3',
'classwork 02-12.ipynb',
'copy1.ipynb',
'Data_Analysis_Lecture_DAT (1).ipynb',
'Decision Tree classification.ipynb',
'done_assign_2.ipynb',
'fashion1_master.csv',
'fashionbrands_sql.db',
'fashionbrandx_sql.db',
'fashionbrandy_sql.db',
'fashionbrandz_sql.db',
'fashionbrand_sql.db',
'fashionbrnd_sql.db',
'fashion_brand_details.xlsx',
'fashion_dataset.csv',
'fashion_master.csv',
'fashion_sql.db',
'filteredblackprdts.csv',
'i-Copy1.ipynb',
'i.ipynb',
'jupyter_sql.db',
'LowertRatedMovieSpark.txt',
'Multiple Regression-2 Complete.ipynb',
'PySpark (1).ipynb',
'real_estate.csv',
'review prdts visual.pbix',
'review prdts visual.pdf',
'reviewprdts.csv',
'spark-3.1.1-bin-hadoop3.2',
'spark-3.1.1-bin-hadoop3.2.tgz',
'spark-3.1.1-bin-hadoop3.2.tgz.1',
'spark-3.1.1-bin-hadoop3.2.tgz.2',
'Untitled.ipynb',
'Untitled1.ipynb',
'upfilteredblackprdts.csv']
```

> The first data frame design1_df

In [3]:
```
design1_df = pd.read_csv("fashion_dataset.csv")
```

In [4]:
```
design1_df.head(3)
```

Out[4]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	d
0	1518329.0	Dupatta Bazaar White Embroidered Chiffon Dupatta	899.0	White	Dupatta Bazaar	1321.0	4.548827	embroidered dupattaChiffon<b
1	5829334.0	Roadster Women Mustard Yellow Solid Hooded Swe...	1199.0	Mustard	Roadster	5462.0	4.313255	Mustard yellow solid sweatshirt, has
2	10340119.0	Inddus Peach-Coloured & Beige	5799.0	Peach	Inddus	145.0	4.068966	Peach-Coloured and beige wo

Unstitched Dress...

In [5]:
```
design1_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14329 entries, 0 to 14328
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   p_id          14311 non-null  float64
 1   name          14310 non-null  object 
 2   price         14310 non-null  float64
 3   colour        14307 non-null  object 
 4   brand         14305 non-null  object 
 5   ratingCount   6581 non-null   float64
 6   avg_rating    6581 non-null   float64
 7   description   14310 non-null  object 
 8   p_attributes  14310 non-null  object 
dtypes: float64(4), object(5)
memory usage: 1007.6+ KB
```

In [6]:
```
# The .info() method showed a number of empty entries especially in ratingCount and avg_
# There were also missing entries in every other column
# The missing values in product ID column were filled with zeros; the price, rating coun
#columns were filled with their means and all missing string values were filled with 'NA

design1_df['p_id'] = design1_df['p_id'].fillna(0).astype(int)
```

In [7]:
```
# These float data type columns filled with the mean of the entries

design1_df['avg_rating'].fillna(design1_df['avg_rating'].mean(), inplace=True)
design1_df['ratingCount'].fillna(design1_df['ratingCount'].mean(), inplace=True)
design1_df['price'].fillna(design1_df['price'].mean(), inplace=True)
```

In [8]:
```
# String columns filled with 'NA'

design1_df[['name','colour','brand','description','p_attributes']] = design1_df[['name',
```

In [9]:
```
# Results of initial wrangling investigated

design1_df.head()
```

Out[9]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	
0	1518329	Dupatta Bazaar White Embroidered Chiffon Dupatta	899.0	White	Dupatta Bazaar	1321.0	4.548827	embroidered dupattaChiffon<
1	5829334	Roadster Women Mustard Yellow Solid Hooded Swe...	1199.0	Mustard	Roadster	5462.0	4.313255	Mustard yellow solid sweatshirt, ha
2	10340119	Inddus Peach-Coloured & Beige	5799.0	Peach	Inddus	145.0	4.068966	Peach-Coloured and beige w

		Unstitched Dress...						
3	10856380	SASSAFRAS Women Black Parallel Trousers	1499.0	Black	SASSAFRAS	9124.0	4.147523	Black solid woven high-rise paralle
4	12384822	Kotty Women Black Wide Leg High-Rise Clean Loo...	1999.0	Black	Kotty	12260.0	4.078467	Black dark wash 4-pocket high-rise

In [10]:
```
design1_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14329 entries, 0 to 14328
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   p_id          14329 non-null  int32  
 1   name          14329 non-null  object 
 2   price         14329 non-null  float64
 3   colour        14329 non-null  object 
 4   brand         14329 non-null  object 
 5   ratingCount   14329 non-null  float64
 6   avg_rating    14329 non-null  float64
 7   description   14329 non-null  object 
 8   p_attributes  14329 non-null  object 
dtypes: float64(3), int32(1), object(5)
memory usage: 951.7+ KB
```

In [11]:
```
# Avg_rating and price columns were rounded up to 2 decimal places and rating count chan
# Since p_id is immutable, it is set as string data type

design1_df[['avg_rating','price']] = round(design1_df[['avg_rating','price']], 2)
design1_df['ratingCount'] = design1_df['ratingCount'].astype(int)
design1_df['p_id'] = design1_df['p_id'].astype(str)
```

In [12]:
```
design1_df.tail()
```

Out[12]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes
14324	17029604	The Chennai Silks Pink & Silver-Toned Floral Z...	3999.0	Pink	The Chennai Silks	184	4.1	Design Details Pink and silver-...	{'Better Cotton Initiative': 'Regular', 'Blous...
14325	17600212	Kinder Kids Girls Blue & Green Printed Foil Pr...	2050.0	Blue	Kinder Kids	184	4.1	Blue and green printed lehenga choli, has foi...	{'Blouse Closure': 'NA', 'Blouse Fabric': 'Cot...
14326	18159266	KLOTTHE Women Green & Black Floral Printed Pal...	1659.0	Green	KLOTTHE	184	4.1	Green and black woven palazzos ...	{'Body or Garment Size': 'To-Fit Denotes Body ...
14327	18921114	InWeave	2399.0	Red	InWeave	184	4.1	<p>Red	{'Add-Ons':

		Women Red Printed A-Line Skirt							printed A-line skirt, has drawstring cl...	'NA', 'Body Shape ID': '324,333,42...
14328	19361058	BoStreet Women Navy Blue Tapered Fit Trousers	2599.0	Navy Blue	BoStreet		184	4.1	Navy blue knitted trousers <li...	{'Add-Ons': 'NA', 'Body Shape ID': '443,333,42...

In [13]:
```
# Checked for duplicates

design1_df['p_id'].duplicated().sum()
```

Out[13]:
```
106
```

In [14]:
```
# Dropped duplicates

design1_df.drop_duplicates(['p_id'], inplace=True)
```

In [15]:
```
# Product id 'p_id' set as index

design1_df.set_index('p_id', inplace=True)
```

In [16]:
```
# Cleaned data frame

design1_df.head(3)
```

Out[16]:

p_id	name	price	colour	brand	ratingCount	avg_rating	descr
1518329	Dupatta Bazaar White Embroidered Chiffon Dupatta	899.0	White	Dupatta Bazaar	1321	4.55	embroidered dupattaChiffon H
5829334	Roadster Women Mustard Yellow Solid Hooded Swe...	1199.0	Mustard	Roadster	5462	4.31	Mustard yellow solid sweatshirt, has a ho
10340119	Inddus Peach-Coloured & Beige Unstitched Dress...	5799.0	Peach	Inddus	145	4.07	Peach-Coloured and beige woven uns

In [17]:
```
design1_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 14223 entries, 1518329 to 19361058
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   name           14223 non-null  object
```

```
 1   price          14223 non-null  float64
 2   colour         14223 non-null  object
 3   brand          14223 non-null  object
 4   ratingCount    14223 non-null  int32
 5   avg_rating     14223 non-null  float64
 6   description    14223 non-null  object
 7   p_attributes   14223 non-null  object
dtypes: float64(2), int32(1), object(5)
memory usage: 944.5+ KB
```

> The second data frame design2_df

In [18]:
```
design2_df = pd.read_excel("fashion_brand_details.xlsx")
```

In [19]:
```
# The second data frame was investigated using .describe(), .info(), .isnull() as well a

design2_df.head()
```

Out[19]:

	brand_id	brand_name
0	1	513
1	2	109F
2	3	20Dresses
3	4	250 Designs
4	5	3Pin

In [20]:
```
design2_df.describe()
```

Out[20]:

	brand_id
count	1020.000000
mean	510.500000
std	294.592939
min	1.000000
25%	255.750000
50%	510.500000
75%	765.250000
max	1020.000000

In [21]:
```
design2_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1020 entries, 0 to 1019
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   brand_id    1020 non-null   int64
 1   brand_name  1020 non-null   object
dtypes: int64(1), object(1)
memory usage: 16.1+ KB
```

In [22]:
```
design2_df.isnull().sum()
```

```
brand_id      0
```

Out[22]:
```
brand_name    0
dtype: int64
```

In [23]:
```
design2_df.duplicated().sum()
```

Out[23]:
```
0
```

In [24]:
```
# Index was set as brand_id as there were similarities between brand_id and the newly cr

design2_df.set_index('brand_id', inplace=True)
```

In [25]:
```
design2_df.head()
```

Out[25]:

brand_id	brand_name
1	513
2	109F
3	20Dresses
4	250 Designs
5	3Pin

In [ ]:
```

```

Merging the data frames with SQL

In [26]:
```
# In order to merge the 2 dataframes using SQL's INNER JOIN on dataframes design1_df and
#and ipython-sql was installed

import sqlite3
```

In [27]:
```
!pip install ipython-sql
```

```
Requirement already satisfied: ipython-sql in c:\users\ikenna\anaconda3\lib\site-package
s (0.4.1)
Requirement already satisfied: ipython>=1.0 in c:\users\ikenna\anaconda3\lib\site-packag
es (from ipython-sql) (8.4.0)
Requirement already satisfied: six in c:\users\ikenna\anaconda3\lib\site-packages (from
ipython-sql) (1.16.0)
Requirement already satisfied: ipython-genutils>=0.1.0 in c:\users\ikenna\anaconda3\lib
\site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: prettytable<1 in c:\users\ikenna\anaconda3\lib\site-packa
ges (from ipython-sql) (0.7.2)
Requirement already satisfied: sqlalchemy>=0.6.7 in c:\users\ikenna\anaconda3\lib\site-p
ackages (from ipython-sql) (1.4.39)
Requirement already satisfied: sqlparse in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython-sql) (0.4.3)
Requirement already satisfied: colorama in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=1.0->ipython-sql) (0.4.5)
Requirement already satisfied: pygments>=2.4.0 in c:\users\ikenna\anaconda3\lib\site-pac
kages (from ipython>=1.0->ipython-sql) (2.11.2)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in c:\users
\ikenna\anaconda3\lib\site-packages (from ipython>=1.0->ipython-sql) (3.0.20)
Requirement already satisfied: decorator in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=1.0->ipython-sql) (5.1.1)
Requirement already satisfied: stack-data in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=1.0->ipython-sql) (0.2.0)
Requirement already satisfied: matplotlib-inline in c:\users\ikenna\anaconda3\lib\site-p
```

```
ackages (from ipython>=1.0->ipython-sql) (0.1.6)
Requirement already satisfied: traitlets>=5 in c:\users\ikenna\anaconda3\lib\site-packag
es (from ipython>=1.0->ipython-sql) (5.1.1)
Requirement already satisfied: pickleshare in c:\users\ikenna\anaconda3\lib\site-package
s (from ipython>=1.0->ipython-sql) (0.7.5)
Requirement already satisfied: backcall in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=1.0->ipython-sql) (0.2.0)
Requirement already satisfied: jedi>=0.16 in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=1.0->ipython-sql) (0.18.1)
Requirement already satisfied: setuptools>=18.5 in c:\users\ikenna\anaconda3\lib\site-pa
ckages (from ipython>=1.0->ipython-sql) (63.4.1)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\ikenna\anaconda3\lib\site-pa
ckages (from sqlalchemy>=0.6.7->ipython-sql) (1.1.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\ikenna\anaconda3\lib\site
-packages (from jedi>=0.16->ipython>=1.0->ipython-sql) (0.8.3)
Requirement already satisfied: wcwidth in c:\users\ikenna\anaconda3\lib\site-packages (f
rom prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=1.0->ipython-sql) (0.2.5)
Requirement already satisfied: asttokens in c:\users\ikenna\anaconda3\lib\site-packages
(from stack-data->ipython>=1.0->ipython-sql) (2.0.5)
Requirement already satisfied: executing in c:\users\ikenna\anaconda3\lib\site-packages
(from stack-data->ipython>=1.0->ipython-sql) (0.8.3)
Requirement already satisfied: pure-eval in c:\users\ikenna\anaconda3\lib\site-packages
(from stack-data->ipython>=1.0->ipython-sql) (0.2.2)
```

In [28]:
```
# A database was created

cnn = sqlite3.connect('fashionbrands_sql.db')
```

In [29]:
```
# design1_df was loaded to the database

design1_df.to_sql('fashiondataQ', cnn)
```

Out[29]:
```
14223
```

In [30]:
```
#design2_df was loaded to the database

design2_df.to_sql('fashiondataR', cnn)
```

Out[30]:
```
1020
```

In [31]:
```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [32]:
```
%sql sqlite:///fashionbrands_sql.db
```

In [33]:
```
#created the master fashion data frame from SQL's INNER JOIN merge

fashion_master_df = pd.read_sql("""
                    SELECT *
                    FROM fashiondataQ as a
                    INNER JOIN fashiondataR as b
                    ON a.brand=b.brand_name
                    """, con = cnn)
```

In [34]:
```
fashion_master_df.head()
```

Out[34]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	d
0	1518329	Dupatta Bazaar White Embroidered	899.0	White	Dupatta Bazaar	1321	4.55	embroidered dupattaChiffon<b

		Chiffon Dupatta						
1	5829334	Roadster Women Mustard Yellow Solid Hooded Swe...	1199.0	Mustard	Roadster	5462	4.31	Mustard yellow solid sweatshirt, has
2	10340119	Inddus Peach-Coloured & Beige Unstitched Dress...	5799.0	Peach	Inddus	145	4.07	Peach-Coloured and beige wov
3	12384822	Kotty Women Black Wide Leg High-Rise Clean Loo...	1999.0	Black	Kotty	12260	4.08	Black dark wash 4-pocket high-rise je
4	14021452	Sera Women Multicoloured Printed Tie-Up Shrug	1494.0	Multi	Sera	750	4.29	Brown and blue printed tie-up longlin

In [35]:
```
fashion_master_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8052 entries, 0 to 8051
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   p_id           8052 non-null   object 
 1   name           8052 non-null   object 
 2   price          8052 non-null   float64
 3   colour         8052 non-null   object 
 4   brand          8052 non-null   object 
 5   ratingCount    8052 non-null   int64  
 6   avg_rating     8052 non-null   float64
 7   description    8052 non-null   object 
 8   p_attributes   8052 non-null   object 
 9   brand_id       8052 non-null   int64  
 10  brand_name     8052 non-null   object 
dtypes: float64(2), int64(2), object(7)
memory usage: 692.1+ KB
```

In [36]:
```
# Dataframe moved to csv file

fashion_master_df.to_csv('fashion1_master.csv', index=False)
```

In [ ]:
```

```

Questions

> Q1: What were the products with an average rating of 3.0 and below? Information on this would determine whether product will be discarded or improved.

In [37]:
```
# Data frame was created to review products of rating 3.0 and lower
```

```
review_prdts_df = fashion_master_df[fashion_master_df['avg_rating'] <= 3].sort_values('p
```

In [38]:
```
review_prdts_df.reset_index(drop=True, inplace=True)
```

In [39]:
```
review_prdts_df
```

Out[39]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attribu
0	1411010	Dupatta Bazaar Beige Striped Chanderi Cotton S...	449.0	Beige	Dupatta Bazaar	6	2.83	Beige striped dupatta, has a zari border ...	{'Ornamentati 'Zari', 'Prin Pattern
1	14745736	Baby Lakshmi Girls Orange & Off White Ready to...	698.0	Orange	Baby Lakshmi	1	2.00	Orange and off white solid lehenga choli, O...	{'Blouse Closu 'Button', 'Blc Fabri
2	13278842	Bronz Women Off-White Printed Tunic	699.0	Off White	Bronz	12	2.83	Off-White printed Tunic, has a round neck, and...	{'Bod Garment S 'To-Fit Denc Boc
3	18290008	Go Colors Women Black Tapered Fit Trousers	799.0	Black	Go Colors	2	3.00	Black woven trousers Tape...	{'Add-Ons': 'I 'Body Shape '443,333,
4	11524932	Vishudh Women's Mustard & Pink Striped Tunic	849.0	Mustard	Vishudh	13	2.31	Mustard and Pink striped Tunic, has a mandarin...	{'Bod Garment S 'Garm Measuremer
...	...	...	...	...	...	...	...	...	
129	12418562	Geroo Jaipur Hand Dyed Bandhani Sea Green Silk...	8999.0	Sea Green	Geroo Jaipur	5	2.60	Sea Green lehenga choli Pink and Golden wov...	{'Blouse Closu 'NA', 'Blc Fabric': 'A
130	16877276	Mitera Women Purple & Silver Embroidered Lehen...	9600.0	Purple	Mitera	3	3.00	Purple and silver embroidered lehenga choli wi...	{'Blouse Closu 'Zip', 'Blc Fabric': '
131	15344604	Chhabra 555 Teal Green Embroidered Zardozi Mad...	9900.0	Teal	Chhabra 555	2	3.00	Bring your dream look to life with this fashio...	{'Blouse Closu 'NA', 'Blc Fabric': 'F
132	14539442	Chhabra 555 Sea Green & Gold-Toned Embellished...	11900.0	Sea Green	Chhabra 555	3	2.67	Sea green and gold-toned embellished lehenga c...	{'Blouse Closu 'NA', 'Blc Fabric': 'N
133	15552632	Chhabra 555 Pretty Pink Made To Measure Leheng...	12400.0	Pink	Chhabra 555	3	2.33	Own the party by wearing this stylishly design...	{'Blouse Closu 'Zip', 'Blc Fabric': '

134 rows × 11 columns

In [40]:
```
# The least price in the master dataset

fashion_master_df['price'].min()
```

Out[40]:
```
295.0
```

> Answer 1: The fashion retail companies are advised to not discard the average rated products but rather improve them. This is because these products still generate revenue from per unit sale well above the least sale price of 295 dollars. Hence, products being discarded could reduce profit margins

In [ ]:
```

```

> Q2: This question explored brand name, description, average rating and price of products that were of colour black. How much did fashion companies actually make from black products?

In [41]:
```
# Creating the black products dataframe

black_prdts_df = fashion_master_df[fashion_master_df['colour'] == 'Black'].sort_values('
```

In [42]:
```
# Dropping any new index created

black_prdts_df.reset_index(drop=True, inplace=True)
```

In [43]:
```
black_prdts_df.head()
```

Out[43]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes	brand_
0	16920516	Masaba Woman Black Tulip Cape Set	30000.0	Black	Masaba	184	4.10	V-neckline Full sleeves Embr...	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...	5
1	18585472	Mitera Black & Pink Floral Pure Linen Saree	22300.0	Black	Mitera	184	4.10	Design Details Black and pi...	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...	5
2	13168738	Justanned Plus Women Plus Size Black Solid Lea...	19999.0	Black	Justanned	184	4.10	Black solid jacket, has a spread collar, 6 poc...	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...	4
3	17981980	Justanned Women Black Leather Lightweight Crop...	19998.0	Black	Justanned	184	4.10	Black solid lightweight biker jacket with zip...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	4
4	15880878	Justanned Women Black Striped	15998.0	Black	Justanned	6	4.67	Black solid striped biker jacket with zip det...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	4

Leather
Crop Out...

In [44]:
```
# Selecting required fields

filtered_black_prdts_df = black_prdts_df.loc[:,['brand_name','avg_rating','price', 'desc
```

In [45]:
```
filtered_black_prdts_df.head()
```

Out[45]:

	brand_name	avg_rating	price	description
0	Masaba	4.10	30000.0	V-neckline Full sleeves Embr...
1	Mitera	4.10	22300.0	Design Details Black and pi...
2	Justanned	4.10	19999.0	Black solid jacket, has a spread collar, 6 poc...
3	Justanned	4.10	19998.0	Black solid lightweight biker jacket with zip...
4	Justanned	4.67	15998.0	Black solid striped biker jacket with zip det...

In [46]:
```
filtered_black_prdts_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1049 entries, 0 to 1048
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   brand_name   1049 non-null   object 
 1   avg_rating   1049 non-null   float64
 2   price        1049 non-null   float64
 3   description  1049 non-null   object 
dtypes: float64(2), object(2)
memory usage: 32.9+ KB
```

In [47]:
```
# There are 1,049 black clothes

filtered_black_prdts_df.describe()
```

Out[47]:

	avg_rating	price
count	1049.000000	1049.000000
mean	4.103365	2623.678742
std	0.327403	2208.596538
min	1.670000	295.000000
25%	4.100000	1499.000000
50%	4.100000	1999.000000
75%	4.210000	2990.000000
max	5.000000	30000.000000

In [48]:
```
# 268 brands made black clothes

len(filtered_black_prdts_df['brand_name'].unique())
```

Out[48]:
```
268
```

In [49]:
```
# Percentage of black products was approximately 13%
```

```
per_blck = (len(black_prdts_df)/len(fashion_master_df))*100
per_blck
```

Out[49]:
```
13.02781917536016
```

In [50]:
```
# Dataframe moved to csv file

filtered_black_prdts_df.to_csv('upfilteredblackprdts.csv', index=False)
```

In [51]:
```
# Average price of black clothes

filtered_black_prdts_df['price'].mean()
```

Out[51]:
```
2623.6787416587226
```

In [52]:
```
# Average price of all clothes

fashion_master_df['price'].mean()
```

Out[52]:
```
2876.9600099354197
```

In [53]:
```
# Mean of average rating

filtered_black_prdts_df['avg_rating'].mean()
```

Out[53]:
```
4.103365109628194
```

> Answer 2: The findings showed that the average pricing of black clothes was £2623.68. This was lower than the general average of £2876.96 but close enough to it. The average rating though remained high with a mean of over 4.10. Hence, black clothes generally had above average rating despite their average sale price.

In [ ]:
```

```

MongoDB application to product attributes field, p_attributes

> Looking further into black coloured products and their attributes breakdown using MongoDB

In [54]:
```
# Filtered for product attributes and brand name

mongofiltered_black_prdts_df = black_prdts_df[['brand_name','p_attributes']]
```

In [55]:
```
mongofiltered_black_prdts_df
```

Out[55]:

	brand_name	p_attributes
0	Masaba	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...
1	Mitera	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...
2	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...
3	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...

4	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
...	...	...
1044	Soch	{'Border': 'Woven Design', 'Fabric': 'Art Silk...
1045	Zivame	{'Add-Ons': 'NA', 'Body Shape ID': '443,324,42...
1046	Friskers	{'Body Shape ID': '333,424', 'Body or Garment ...
1047	Bewakoof	{'Body or Garment Size': 'To-Fit Denotes Body ...
1048	Jockey	{'Body Shape ID': '333,424', 'Body or Garment ...

1049 rows × 2 columns

In [56]:
```
mongofiltered_black_prdts_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1049 entries, 0 to 1048
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   brand_name     1049 non-null   object
 1   p_attributes   1049 non-null   object
dtypes: object(2)
memory usage: 16.5+ KB
```

In [57]:
```
# Exploring the attributes

mongofiltered_black_prdts_df['p_attributes']
```

Out[57]:
```
0       {'Add-Ons': 'NA', 'Better Cotton Initiative': ...
1       {'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...
2       {'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...
3       {'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
4       {'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
                              ...
1044    {'Border': 'Woven Design', 'Fabric': 'Art Silk...
1045    {'Add-Ons': 'NA', 'Body Shape ID': '443,324,42...
1046    {'Body Shape ID': '333,424', 'Body or Garment ...
1047    {'Body or Garment Size': 'To-Fit Denotes Body ...
1048    {'Body Shape ID': '333,424', 'Body or Garment ...
Name: p_attributes, Length: 1049, dtype: object
```

In [58]:
```
# ast library imported to change the attributes data type from string to dictionary

from ast import literal_eval
```

In [59]:
```
mongofiltered_black_prdts_df['Attributes'] = mongofiltered_black_prdts_df['p_attributes'
```

```
C:\Users\IKENNA\AppData\Local\Temp\ipykernel_41756\297140986.py:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  mongofiltered_black_prdts_df['Attributes'] = mongofiltered_black_prdts_df['p_attribute
s'].apply(lambda x: literal_eval(x))
```

In [60]:
```
mongofiltered_black_prdts_df.head()
```

Out[60]:

	brand_name	p_attributes	Attributes
0	Masaba	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...

1	Mitera	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...
2	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...
3	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
4	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...

In [61]:
```
type(mongofiltered_black_prdts_df['Attributes'][0])
```

Out[61]:
```
dict
```

In [62]:
```
mongofiltered_black_prdts_df.drop('p_attributes', axis=1, inplace=True)
```

```
C:\Users\IKENNA\AppData\Local\Temp\ipykernel_41756\1633810029.py:1: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  mongofiltered_black_prdts_df.drop('p_attributes', axis=1, inplace=True)
```

In [63]:
```
mongofiltered_black_prdts_df.head()
```

Out[63]:

	brand_name	Attributes
0	Masaba	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...
1	Mitera	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...
2	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...
3	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...
4	Justanned	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...

In [64]:
```
# The filtered dataframe is moved to csv to be imported to mongoDB Compass for further a

mongofiltered_black_prdts_df.to_csv('blackcprdts.csv', index=False)
```

In [65]:
```
# To work in the notebook, pymongo is installed and imported

!pip install pymongo
```

```
Requirement already satisfied: pymongo in c:\users\ikenna\anaconda3\lib\site-packages
(4.3.3)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in c:\users\ikenna\anaconda3\lib
\site-packages (from pymongo) (2.2.1)
```

In [66]:
```
import pymongo
```

In [67]:
```
# Server is set at localhost:27017

client = pymongo.MongoClient("localhost", 27017)
```

In [68]:
```
# The database is created and named

db = client.fashiondb
```

In [69]:
```
db.name
```

Out[69]:
```
'fashiondb'
```

In [70]:
```
# The list of databases at localhost 27017

client.list_database_names()
```

Out[70]:
```
['admin', 'config', 'fashiondb', 'local']
```

In [71]:
```
# A collection is created

coll = db.products_attributes
```

In [72]:
```
# Details on the product attributes collection

db.products_attributes
```

Out[72]:
```
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=
False, connect=True), 'fashiondb'), 'products_attributes')
```

In [73]:
```
db.list_collection_names()
```

Out[73]:
```
['products_attributes']
```

In [74]:
```
# Collection has 1049 documents

coll.count_documents({})
```

Out[74]:
```
1049
```

In [75]:
```
# Selected a document

coll.find_one()
```

Out[75]:
```
{'_id': ObjectId('63872b29c557eb483de5b023'),
 'brand_name': 'Masaba',
 'p_attributes': "{'Add-Ons': 'NA', 'Better Cotton Initiative': 'Regular', 'Bottom Closu
re': 'Zip', 'Bottom Fabric': 'Silk Blend', 'Bottom Pattern': 'Printed', 'Bottom Type':
'Trousers', 'Character': 'NA', 'Lining': 'NA', 'Neck': 'Round Neck', 'Number of Pocket
s': 'NA', 'Occasion': 'Casual', 'Sleeve Length': 'Long Sleeves', 'Sustainable': 'Regula
r', 'Top Fabric': 'Silk Blend', 'Top Pattern': 'Solid', 'Top Type': 'Top', 'Trends': 'N
A', 'Wash Care': 'Dry Clean'}"}
```

In [76]:
```
# Filtered and selected several documents

cur = coll.find({'brand_name': "Columbia"}, {'p_attributes': 1, '_id': 1})
for doc in cur:
    print(doc)
```

```
{'_id': ObjectId('63872b29c557eb483de5b029'), 'p_attributes': "{'Add-Ons': 'NA', 'Body S
hape ID': '424', 'Body or Garment Size': 'Garment Measurements in', 'Character': 'NA',
'Closure': 'Zip', 'Collar': 'Hooded', 'Fabric': 'Nylon', 'Features': 'Reflective Strip',
'Hemline': 'Straight', 'Length': 'Longline', 'Lining Fabric': 'Polyester', 'Main Trend':
'NA', 'Number of Pockets': '3', 'Occasion': 'Sports', 'Pattern': 'Solid', 'Print or Patt
ern Type': 'Solid', 'Sleeve Length': 'Long Sleeves', 'Sport': 'Outdoor', 'Surface Stylin
g': 'NA', 'Sustainable': 'Regular', 'Technology': 'NA', 'Technology Present': 'Yes', 'Ty
pe': 'Padded Jacket', 'Wash Care': 'Machine Wash', 'Weave Type': 'Woven', 'Where-to-wea
r': ''}"}
{'_id': ObjectId('63872b29c557eb483de5b02d'), 'p_attributes': "{'Add-Ons': 'NA', 'Body S
hape ID': '333,424', 'Body or Garment Size': 'Garment Measurements in', 'Character': 'N
A', 'Closure': 'Zip', 'Collar': 'Hooded', 'Contact Brand or Retailer for pre-sales produ
ct queries': 'chogori india retail ltd. , a-16, rear-side, mohan co-operative industrial
estate, main mathura road, new delhi-110044', 'Fabric': 'Nylon', 'Features': 'Insulato
r', 'Hemline': 'Straight', 'Length': 'Regular', 'Lining Fabric': 'Nylon', 'Main Trend':
'NA', 'Number of Pockets': '2', 'Occasion': 'Casual', 'Pattern': 'Solid', 'Print or Patt
ern Type': 'Solid', 'Sleeve Length': 'Long Sleeves', 'Sport': 'NA', 'Surface Styling':
```

'Zip Detail', 'Sustainable': 'Regular', 'Technology': 'NA', 'Technology Present': 'No',
'Type': 'Padded Jacket', 'Wash Care': 'Machine Wash', 'Weave Type': 'Woven'}"}
{'_id': ObjectId('63872b29c557eb483de5b02e'), 'p_attributes': "{'Add-Ons': 'NA', 'Body S
hape ID': '333,424', 'Body or Garment Size': 'Garment Measurements in', 'Character': 'N
A', 'Closure': 'Zip', 'Collar': 'Hooded', 'Fabric': 'Polyester', 'Features': 'Insulato
r', 'Hemline': 'Straight', 'Length': 'Regular', 'Lining Fabric': 'Polyester', 'Main Tren
d': 'NA', 'Number of Pockets': '2', 'Occasion': 'Sports', 'Pattern': 'Solid', 'Print or
Pattern Type': 'Solid', 'Sleeve Length': 'Long Sleeves', 'Sport': 'Outdoor', 'Surface St
yling': 'Zip Detail', 'Sustainable': 'Regular', 'Technology': 'NA', 'Technology Presen
t': 'Yes', 'Type': 'Quilted Jacket', 'Wash Care': 'Machine Wash', 'Weave Type': 'Woven',
'Where-to-wear': ''}"}
{'_id': ObjectId('63872b29c557eb483de5b03d'), 'p_attributes': "{'Add-Ons': 'NA', 'Body S
hape ID': '333,424', 'Body or Garment Size': 'Garment Measurements in', 'Character': 'N
A', 'Closure': 'Zip', 'Collar': 'Mock Collar', 'Fabric': 'Polyester', 'Features': 'Refle
ctive Strip', 'Hemline': 'Straight', 'Length': 'Regular', 'Lining Fabric': 'Polyester',
'Main Trend': 'NA', 'Number of Pockets': '2', 'Occasion': 'Sports', 'Pattern': 'Solid',
'Print or Pattern Type': 'Solid', 'Sleeve Length': 'Sleeveless', 'Sport': 'Outdoor', 'Su
rface Styling': 'NA', 'Sustainable': 'Regular', 'Technology': 'NA', 'Technology Presen
t': 'Yes', 'Type': 'Quilted Jacket', 'Wash Care': 'Machine Wash', 'Weave Type': 'Woven',
'Where-to-wear': 'hiking, ski, snow sports, casual lifestyle'}"}
{'_id': ObjectId('63872b29c557eb483de5b05d'), 'p_attributes': "{'Add-Ons': 'NA', 'Body S
hape ID': '443,424', 'Body or Garment Size': 'To-Fit Denotes Body Measurements in', 'Bra
nd Fit Name': 'NA', 'Care for me': 'Follow the garment wash care label
\\r\\nCan be t
umble dried in a cool setting if machine washed
\\r\\nPrevent static build-up by dryi
ng while damp
\\r\\nUse low heat to iron', 'Character': 'Mickey & Donald', 'Closure':
'Slip-On', 'Contact Brand or Retailer for pre-sales product queries': 'Chogori India Ret
ail Ltd. , A-16, Rear-side, Mohan Co-operative Industrial Estate, Main Mathura Road, New
Delhi-110044', 'Fabric': 'Polyester', 'Fabric 2': 'NA', 'Features': 'Plain', 'Fit': 'Sli
m Fit', 'Fly Type': 'No Fly', 'Length': 'Regular', 'Main Trend': 'New Basics', 'Multipac
k Set': 'NA', 'Number of Pockets': 'NA', 'Occasion': 'Casual', 'Pattern': 'Solid', 'Prin
t or Pattern Type': 'Solid', 'Sustainable': 'Regular', 'Type': 'Regular Trousers', 'Type
of Pleat': 'Flat-Front', 'Waist Rise': 'Mid-Rise', 'Wash Care': 'Machine Wash', 'Weave T
ype': 'Woven'}"}
{'_id': ObjectId('63872b29c557eb483de5b05e'), 'p_attributes': "{'Add-Ons': 'NA', 'Body S
hape ID': '443,324,333,424', 'Body or Garment Size': 'Garment Measurements in', 'Charact
er': 'NA', 'Closure': 'Zip', 'Collar': 'Mock Collar', 'Fabric': 'Polyester', 'Features':
'NA', 'Hemline': 'Straight', 'Length': 'Regular', 'Lining Fabric': 'Polyester', 'Main Tr
end': 'NA', 'Number of Pockets': '2', 'Occasion': 'Sports', 'Pattern': 'Solid', 'Print o
r Pattern Type': 'Solid', 'Sleeve Length': 'Long Sleeves', 'Sport': 'Outdoor', 'Surface
Styling': 'NA', 'Sustainable': 'Regular', 'Technology': 'NA', 'Technology Present': 'Ye
s', 'Type': 'Sporty Jacket', 'Wash Care': 'Machine Wash', 'Weave Type': 'Woven', 'Where-
to-wear': ''}"}
{'_id': ObjectId('63872b29c557eb483de5b0e4'), 'p_attributes': "{'Add-Ons': 'NA', 'Body S
hape ID': '443,333,424', 'Body or Garment Size': 'To-Fit Denotes Body Measurements in',
'Brand Fit Name': 'NA', 'Character': 'NA', 'Closure': 'Slip-On', 'Fabric': 'Polyester',
'Fabric 2': 'Elastane', 'Features': 'Stain Resistant', 'Fit': 'Regular Fit', 'Fly Type':
'No Fly', 'Length': 'Regular', 'Main Trend': 'New Basics', 'Multipack Set': 'NA', 'Numbe
r of Pockets': '2', 'Occasion': 'Casual', 'Pattern': 'Solid', 'Print or Pattern Type':
'Solid', 'Sustainable': 'Regular', 'Type': 'Regular Trousers', 'Type of Pleat': 'Flat-Fr
ont', 'Waist Rise': 'Mid-Rise', 'Wash Care': 'Machine Wash', 'Weave Type': 'Knitted'}"}

In [ ]:

In [ ]:

Working with Spark(PySpark)

In [77]:
```
# PySpark and sparksql were installed

!pip install pyspark
```

Requirement already satisfied: pyspark in c:\users\ikenna\anaconda3\lib\site-packages
(3.3.1)

```
Requirement already satisfied: py4j==0.10.9.5 in c:\users\ikenna\anaconda3\lib\site-pack
ages (from pyspark) (0.10.9.5)
```

In [78]:
```
!pip install sparksql-magic
```

```
Requirement already satisfied: sparksql-magic in c:\users\ikenna\anaconda3\lib\site-pack
ages (0.0.3)
Requirement already satisfied: pyspark>=2.3.0 in c:\users\ikenna\anaconda3\lib\site-pack
ages (from sparksql-magic) (3.3.1)
Requirement already satisfied: ipython>=7.4.0 in c:\users\ikenna\anaconda3\lib\site-pack
ages (from sparksql-magic) (8.4.0)
Requirement already satisfied: traitlets>=5 in c:\users\ikenna\anaconda3\lib\site-packag
es (from ipython>=7.4.0->sparksql-magic) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=7.4.0->sparksql-magic) (0.18.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in c:\users
\ikenna\anaconda3\lib\site-packages (from ipython>=7.4.0->sparksql-magic) (3.0.20)
Requirement already satisfied: decorator in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=7.4.0->sparksql-magic) (5.1.1)
Requirement already satisfied: matplotlib-inline in c:\users\ikenna\anaconda3\lib\site-p
ackages (from ipython>=7.4.0->sparksql-magic) (0.1.6)
Requirement already satisfied: backcall in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=7.4.0->sparksql-magic) (0.2.0)
Requirement already satisfied: setuptools>=18.5 in c:\users\ikenna\anaconda3\lib\site-pa
ckages (from ipython>=7.4.0->sparksql-magic) (63.4.1)
Requirement already satisfied: colorama in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=7.4.0->sparksql-magic) (0.4.5)
Requirement already satisfied: pygments>=2.4.0 in c:\users\ikenna\anaconda3\lib\site-pac
kages (from ipython>=7.4.0->sparksql-magic) (2.11.2)
Requirement already satisfied: stack-data in c:\users\ikenna\anaconda3\lib\site-packages
(from ipython>=7.4.0->sparksql-magic) (0.2.0)
Requirement already satisfied: pickleshare in c:\users\ikenna\anaconda3\lib\site-package
s (from ipython>=7.4.0->sparksql-magic) (0.7.5)
Requirement already satisfied: py4j==0.10.9.5 in c:\users\ikenna\anaconda3\lib\site-pack
ages (from pyspark>=2.3.0->sparksql-magic) (0.10.9.5)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\ikenna\anaconda3\lib\site
-packages (from jedi>=0.16->ipython>=7.4.0->sparksql-magic) (0.8.3)
Requirement already satisfied: wcwidth in c:\users\ikenna\anaconda3\lib\site-packages (f
rom prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=7.4.0->sparksql-magic) (0.2.
5)
Requirement already satisfied: asttokens in c:\users\ikenna\anaconda3\lib\site-packages
(from stack-data->ipython>=7.4.0->sparksql-magic) (2.0.5)
Requirement already satisfied: pure-eval in c:\users\ikenna\anaconda3\lib\site-packages
(from stack-data->ipython>=7.4.0->sparksql-magic) (0.2.2)
Requirement already satisfied: executing in c:\users\ikenna\anaconda3\lib\site-packages
(from stack-data->ipython>=7.4.0->sparksql-magic) (0.8.3)
Requirement already satisfied: six in c:\users\ikenna\anaconda3\lib\site-packages (from
asttokens->stack-data->ipython>=7.4.0->sparksql-magic) (1.16.0)
```

In [79]:
```
# Checked the directory and initiated a spark session

!dir
```

```
 Volume in drive C is Windows
 Volume Serial Number is EA12-F56B

 Directory of C:\Users\IKENNA\bigdatatechassignments

01/01/2023  22:48    <DIR>          .
14/12/2022  01:29    <DIR>          ..
01/01/2023  06:58    <DIR>          .ipynb_checkpoints
19/12/2022  06:40    <DIR>          ALX PROJECTS
07/12/2022  19:47            36,803 bigdata digram.pbix
27/12/2022  06:52           701,009 bigdataassignment_2.ipynb
08/12/2022  18:53         3,265,308 bigdataassignment_2.pdf
06/12/2022  17:38           652,203 blackaprdts.csv
```

```
06/12/2022  17:44               478 blackbprdts.csv
01/01/2023  22:47           607,034 blackcprdts.csv
06/12/2022  16:36           596,797 blackcprdtsattr.csv
30/11/2022  09:44           607,036 blackprdts.csv
06/12/2022  14:41           607,036 blackxprdts.csv
12/10/2022  19:59            22,608 cars.csv
12/10/2022  19:59            22,608 cars.csv.1
13/12/2022  21:21            22,608 cars.csv.2
13/12/2022  21:21            22,608 cars.csv.3
07/12/2022  19:47           569,338 classwork 02-12.ipynb
09/12/2022  11:31           733,827 copy1.ipynb
01/12/2022  23:44           619,814 Data_Analysis_Lecture_DAT (1).ipynb
08/12/2022  21:13            91,288 Decision Tree classification.ipynb
01/01/2023  22:48           934,362 done_assign_2.ipynb
01/01/2023  22:41         7,911,318 fashion1_master.csv
01/01/2023  22:35        49,352,704 fashionbrands_sql.db
06/12/2022  05:54        16,289,792 fashionbrandx_sql.db
07/12/2022  15:48        16,289,792 fashionbrandy_sql.db
08/12/2022  18:21        16,289,792 fashionbrandz_sql.db
29/11/2022  13:33        16,289,792 fashionbrand_sql.db
06/12/2022  05:50        16,289,792 fashionbrnd_sql.db
27/11/2022  22:49            36,780 fashion_brand_details.xlsx
27/11/2022  22:49        13,963,294 fashion_dataset.csv
27/12/2022  06:46         7,903,092 fashion_master.csv
29/11/2022  13:28        16,314,368 fashion_sql.db
01/12/2022  17:14           309,925 filteredblackprdts.csv
09/12/2022  11:31           733,827 i-Copy1.ipynb
27/12/2022  05:37            37,163 i.ipynb
28/11/2022  04:15        32,587,776 jupyter_sql.db
01/12/2022  23:45             1,888 LowertRatedMovieSpark.txt
07/12/2022  19:44            86,061 Multiple Regression-2 Complete.ipynb
01/01/2023  08:43           176,355 PySpark (1).ipynb
11/11/2022  00:12            21,968 real_estate.csv
01/01/2023  15:26            32,769 review prdts visual.pbix
01/01/2023  15:28            85,963 review prdts visual.pdf
01/01/2023  14:10           125,141 reviewprdts.csv
01/01/2023  06:36    <DIR>          spark-3.1.1-bin-hadoop3.2
22/02/2021  02:45       228,721,937 spark-3.1.1-bin-hadoop3.2.tgz
22/02/2021  02:45       228,721,937 spark-3.1.1-bin-hadoop3.2.tgz.1
22/02/2021  02:45       228,721,937 spark-3.1.1-bin-hadoop3.2.tgz.2
08/12/2022  21:16               589 Untitled.ipynb
01/01/2023  06:55             2,332 Untitled1.ipynb
01/01/2023  22:46           309,925 upfilteredblackprdts.csv
              46 File(s)    907,720,774 bytes
               5 Dir(s)  74,766,614,528 bytes free
```

In [80]:
```
pwd
```

Out[80]:
```
'C:\\Users\\IKENNA\\bigdatatechassignments'
```

In [81]:
```
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # Property used to format outpu
spark
```

Out[81]:

SparkSession - in-memory

SparkContext

Spark UI

Version `v3.2.2`

Master	`local[*]`
AppName	`pyspark-shell`

In [82]:
```
# Loaded the master data from csv to a dataframe using spark

fashionspark_df = spark.read.csv('fashion1_master.csv', header=True, sep=",")
fashionspark_df.show(5, truncate=True)
```

```
+--------+--------------------+------+-------+--------------+----------+----------+----
----------------+--------------------+--------+--------------+
|    p_id|                name| price| colour|         brand|ratingCount|avg_rating|
    description|        p_attributes|brand_id|    brand_name|
+--------+--------------------+------+-------+--------------+----------+----------+----
----------------+--------------------+--------+--------------+
| 1518329|Dupatta Bazaar Wh...| 899.0|  White|Dupatta Bazaar|      1321|      4.55|Whit
e embroidered...|{'Occasion': 'Dai...|     242|Dupatta Bazaar|
| 5829334|Roadster Women Mu...|1199.0|Mustard|      Roadster|      5462|      4.31|"Mus
tard yellow s...|{'Body Shape ID':...|     750|      Roadster|
|10340119|Inddus Peach-Colo...|5799.0|  Peach|        Inddus|       145|      4.07|Peac
h-Coloured an...|{'Bottom Fabric':...|     389|        Inddus|
|12384822|Kotty Women Black...|1999.0|  Black|         Kotty|     12260|      4.08|"Bla
ck dark wash ...|{'Add-Ons': 'NA',...|     482|         Kotty|
|14021452|Sera Women Multic...|1494.0|  Multi|          Sera|       750|      4.29|Brow
n and blue pr...|{'Body or Garment...|     793|          Sera|
+--------+--------------------+------+-------+--------------+----------+----------+----
----------------+--------------------+--------+--------------+
only showing top 5 rows
```

In [83]:
```
# Explored the columns

fashionspark_df.columns
```

Out[83]:
```
['p_id',
 'name',
 'price',
 'colour',
 'brand',
 'ratingCount',
 'avg_rating',
 'description',
 'p_attributes',
 'brand_id',
 'brand_name']
```

In [84]:
```
# Explored the schema to see column data types

fashionspark_df.printSchema()
```

```
root
 |-- p_id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- price: string (nullable = true)
 |-- colour: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- ratingCount: string (nullable = true)
 |-- avg_rating: string (nullable = true)
 |-- description: string (nullable = true)
 |-- p_attributes: string (nullable = true)
 |-- brand_id: string (nullable = true)
 |-- brand_name: string (nullable = true)
```

In [85]:
```
from pyspark.sql.types import *
```

In [86]:
```
# Created a list of the schema in the format column, data type
labels = [
     ('p_id',StringType()),
     ('name',StringType()),
     ('price',DoubleType()),
     ('colour',StringType()),
     ('brand',StringType()),
     ('ratingCount',IntegerType()),
     ('avg_rating',DoubleType()),
     ('description',StringType()),
     ('p_attributes',StringType()),
     ('brand_id',IntegerType()),
     ('brand_name',StringType())
]
```

In [87]:
```
# Used StructType method to examine the schema

schema = StructType([StructField (x[0], x[1], True) for x in labels])
schema
```

Out[87]:
```
StructType(List(StructField(p_id,StringType,true),StructField(name,StringType,true),Stru
ctField(price,DoubleType,true),StructField(colour,StringType,true),StructField(brand,Str
ingType,true),StructField(ratingCount,IntegerType,true),StructField(avg_rating,DoubleTyp
e,true),StructField(description,StringType,true),StructField(p_attributes,StringType,tru
e),StructField(brand_id,IntegerType,true),StructField(brand_name,StringType,true)))
```

In [88]:
```
# Updated column data types

fashionspark_df = spark.read.csv('fashion1_master.csv', header=True, sep=",", schema=sch
fashionspark_df.printSchema()
```

```
root
 |-- p_id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- price: double (nullable = true)
 |-- colour: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- ratingCount: integer (nullable = true)
 |-- avg_rating: double (nullable = true)
 |-- description: string (nullable = true)
 |-- p_attributes: string (nullable = true)
 |-- brand_id: integer (nullable = true)
 |-- brand_name: string (nullable = true)
```

In [89]:
```
# Selected required columns

fashionspark_df.select(fashionspark_df['brand_name'],fashionspark_df['p_id'],fashionspar
                                          fashionspark_df['avg_rating']).show(truncate=Tr
```

```
+--------------------+--------+------+-------+----------+
|          brand_name|    p_id| price| colour|avg_rating|
+--------------------+--------+------+-------+----------+
|      Dupatta Bazaar| 1518329| 899.0|  White|      4.55|
|             Roadster| 5829334|1199.0|Mustard|      4.31|
|               Inddus|10340119|5799.0|  Peach|      4.07|
|                Kotty|12384822|1999.0|  Black|      4.08|
|                 Sera|14021452|1494.0|  Multi|      4.29|
|       Tokyo Talkies|14063026| 699.0|  Black|      4.53|
|               Anouk|14324806|4699.0|  Black|      3.81|
|             Roadster|14955068|2599.0|  Mauve|      4.21|
|       Tokyo Talkies|16827132|1399.0| Maroon|      3.91|
|           Khushal K|17048614|5099.0|  Black|      4.42|
|           Anubhutee|17413232|1739.0|   Pink|      4.25|
|               Styli|18841352|1949.0|   Gold|       4.1|
|                 171| 8317561|1699.0|   Blue|      4.31|
```

```
|              Inddus| 8340727|6549.0|   Pink|      3.99|
|       Tokyo Talkies|13401782|1699.0|  Black|      4.46|
|              Athena|13719116|1299.0|Fuchsia|      4.53|
|              Inddus|13843398|4399.0|   Teal|      4.45|
|{'Body Shape ID':...|14024558|3799.0|   Grey|      4.55|
|               Azira|14120438|1199.0|  Multi|      4.25|
|               Anouk|16897384|3399.0|  Mauve|      4.32|
+--------------------+--------+------+-------+----------+
only showing top 20 rows
```

In [90]:
```
fashionspark_df.columns
```

Out[90]:
```
['p_id',
 'name',
 'price',
 'colour',
 'brand',
 'ratingCount',
 'avg_rating',
 'description',
 'p_attributes',
 'brand_id',
 'brand_name']
```

In [91]:
```
# Renamed the columns

fashionspark_df = fashionspark_df.withColumnRenamed('p_id', 'product_id') \
        .withColumnRenamed('ratingCount', 'rating_count') \
        .withColumnRenamed('avg_rating', 'average_rating') \
        .withColumnRenamed('p_attributes', 'product_attributes')
fashionspark_df.show(truncate=True)
```

```
+----------+--------------------+------+-------+----------------+------------+---------
-----+--------------------+--------------------+--------+--------------------+
|product_id|                name| price| colour|           brand|rating_count|average_r
ating|         description|  product_attributes|brand_id|          brand_name|
+----------+--------------------+------+-------+----------------+------------+---------
-----+--------------------+--------------------+--------+--------------------+
|   1518329|Dupatta Bazaar Wh...| 899.0|  White|   Dupatta Bazaar|        1321|
4.55|White embroidered...|{'Occasion': 'Dai...|     242|      Dupatta Bazaar|
|   5829334|Roadster Women Mu...|1199.0|Mustard|        Roadster|        5462|
4.31|"Mustard yellow s...|{'Body Shape ID':...|     750|            Roadster|
|  10340119|Inddus Peach-Colo...|5799.0|  Peach|          Inddus|         145|
4.07|Peach-Coloured an...|{'Bottom Fabric':...|     389|              Inddus|
|  12384822|Kotty Women Black...|1999.0|  Black|           Kotty|       12260|
4.08|"Black dark wash ...|{'Add-Ons': 'NA',...|     482|               Kotty|
|  14021452|Sera Women Multic...|1494.0|  Multi|            Sera|         750|
4.29|Brown and blue pr...|{'Body or Garment...|     793|                Sera|
|  14063026|Tokyo Talkies Wom...| 699.0|  Black|    Tokyo Talkies|        1856|
4.53|"Black solid mid-...|{'Body or Garment...|     903|       Tokyo Talkies|
|  14324806|Anouk Stylish Bla...|4699.0|  Black|           Anouk|          84|
3.81|Stay fashionable ...|{'Blouse Closure'...|      75|               Anouk|
|  14955068|Roadster Women El...|2599.0|  Mauve|        Roadster|         752|
4.21|Add a hint of dra...|{'Add-Ons': 'NA',...|     750|            Roadster|
|  16827132|Tokyo Talkies Wom...|1399.0| Maroon|    Tokyo Talkies|         125|
3.91|<p>Maroon solid r...|{'Body Shape ID':...|     903|       Tokyo Talkies|
|  17048614|Khushal K Women B...|5099.0|  Black|       Khushal K|        4522|
4.42|Black printed Kur...|{'Add-Ons': 'NA',...|     467|           Khushal K|
|  17413232|Anubhutee Pink Et...|1739.0|   Pink|       Anubhutee|         273|
4.25|"<ul> <li> Pink s...|{'Body or Garment...|      79|           Anubhutee|
|  18841352|Styli Women Golde...|1949.0|   Gold|           Styli|         184|
  4.1|"<p>Golden sequin...|{'Add-Ons': 'NA',...|     848|               Styli|
|   8317561|Chemistry Blue De...|1699.0|   Blue|       Chemistry|         900|
4.31|"Blue&amp;nbsp;solid ...| 2% elastane<br>M...|    null|                 171|
|   8340727|Inddus Women Dust...|6549.0|   Pink|          Inddus|         636|
```

```
3.99|"Dusty Pink semi-...|{'Blouse Closure'...|     389|              Inddus|
|  13401782|Tokyo Talkies Wom...|1699.0|  Black|    Tokyo Talkies|         347|
4.46|"Black solid jack...|{'Add-Ons': 'NA',...|     903|       Tokyo Talkies|
|  13719116|Athena Chic Fuchs...|1299.0|Fuchsia|           Athena|       11553|
4.53|Jazz up your coll...|{'Body Shape ID':...|      95|              Athena|
|  13843398|Inddus Teal Blue ...|4399.0|   Teal|           Inddus|         580|
4.45|<b>Design Details...|{'Blouse': 'Blous...|     389|              Inddus|
|  14024558|Allen Solly Woman...|3799.0|   Grey|Allen Solly Woman|         133|
4.55|"<p>Grey solid re...|         33% Viscose|    null|{'Body Shape ID':...|
|  14120438|Azira Multicolour...|1199.0|  Multi|            Azira|        2300|
4.25|<ul><li>Multi-col...|{'Body Shape ID':...|     106|               Azira|
|  16897384|Anouk Women Mauve...|3399.0|  Mauve|            Anouk|          75|
4.32|"Mauve and white ...|{'Body or Garment...|      75|               Anouk|
+----------+--------------------+------+-------+-----------------+------------+---------
-----+--------------------+--------------------+--------+--------------------+
only showing top 20 rows
```

In [92]:
```
# Grouped dataframe by brand name and colour

fashionspark_df.groupBy('brand_name', 'colour').count().orderBy('count', ascending=False
```

```
+--------------+------+-----+
|    brand_name|colour|count|
+--------------+------+-----+
|      Roadster|  Blue|   83|
| Tokyo Talkies|  Blue|   52|
|           H&M| Black|   45|
| Tokyo Talkies| Black|   42|
| Tokyo Talkies| Green|   42|
|      Roadster| Black|   36|
|          null|Mitera|   33|
|          null|   170|   29|
|           H&M| Beige|   27|
|Clora Creation| Black|   27|
+--------------+------+-----+
only showing top 10 rows
```

In [93]:
```
# Dropped brand and product attributes columns

fashionspark_df = fashionspark_df.drop('brand') \
       .drop('product_attributes')
fashionspark_df.show(5,truncate=True)
```

```
+----------+--------------------+------+-------+------------+--------------+------------
--------+--------+--------------+
|product_id|                name| price| colour|rating_count|average_rating|         des
cription|brand_id|    brand_name|
+----------+--------------------+------+-------+------------+--------------+------------
--------+--------+--------------+
|   1518329|Dupatta Bazaar Wh...| 899.0|  White|        1321|          4.55|White embroi
dered...|     242|Dupatta Bazaar|
|   5829334|Roadster Women Mu...|1199.0|Mustard|        5462|          4.31|"Mustard yel
low s...|     750|      Roadster|
|  10340119|Inddus Peach-Colo...|5799.0|  Peach|         145|          4.07|Peach-Colour
ed an...|     389|        Inddus|
|  12384822|Kotty Women Black...|1999.0|  Black|       12260|          4.08|"Black dark
wash ...|     482|         Kotty|
|  14021452|Sera Women Multic...|1494.0|  Multi|         750|          4.29|Brown and bl
ue pr...|     793|          Sera|
+----------+--------------------+------+-------+------------+--------------+------------
--------+--------+--------------+
only showing top 5 rows
```

In [94]:
```
from pyspark.sql.functions import *
```

In [95]:
```
# Filtered rows based on colour and avergae rating conditions

fashionspark_df.filter((col('colour')=='White') & (col('average_rating')>=2.50)).show(tr
```

```
+----------+--------------------+------+------+------------+--------------+------------
-------+--------+--------------+
|product_id|                name| price|colour|rating_count|average_rating|         desc
ription|brand_id|    brand_name|
+----------+--------------------+------+------+------------+--------------+------------
-------+--------+--------------+
|   1518329|Dupatta Bazaar Wh...| 899.0| White|        1321|          4.55|White embroid
ered...|     242|Dupatta Bazaar|
|  11697268|Roadster Women Wh...| 499.0| White|        3106|           4.3|"White printe
d de...|     750|      Roadster|
|  10711448|Dupatta Bazaar Wo...| 599.0| White|        1531|          4.54|White solid d
upat...|     242|Dupatta Bazaar|
|  19147754|H&M Women White S...|1499.0| White|         184|           4.1|"White solid
a-li...|     353|           H&M|
|  13736998|Athena White Tie-...|1699.0| White|        2200|          4.19|<ul><li>Colou
r: w...|      95|        Athena|
|  10182385|Style Quotient Wo...|1599.0| White|        1559|          4.43|White self de
sign...|     845|Style Quotient|
|  11054006|Jaipur Kurti Wome...|1149.0| White|        6590|          4.06|"White solid
mid-...|     418|  Jaipur Kurti|
|  17124538|H&M Women White R...|1499.0| White|          86|          4.66|"<p>Cardigan
in a...|     353|           H&M|
|  17819832|Mast & Harbour Wh...|1599.0| White|         184|           4.1|<ul> <li> Whi
te a...|     554|Mast & Harbour|
|  18437892|Vero Moda Women W...|2499.0| White|         184|           4.1|<p>White and
blac...|     958|     Vero Moda|
|  17449504|Sangria White & G...|2199.0| White|         184|           4.1|<b>ABOUT THE
BRAN...|     776|       Sangria|
|  11672988|Bhama Couture Wom...|1799.0| White|         315|          4.16|"White, green
and...|     131| Bhama Couture|
|   5362057|Bhama Couture Whi...|1799.0| White|         344|          4.02|"White and bl
ue p...|     131| Bhama Couture|
|  18363414|Forever New Women...|4400.0| White|         184|           4.1|White and gre
en p...|     314|   Forever New|
|  18376730|H&M Women White C...|1299.0| White|          30|           4.2|<p>Cropped zi
p-th...|     353|           H&M|
|  17497950|Bewakoof Women Wh...|2549.0| White|           7|          4.86|This clothing
set...|     130|      Bewakoof|
|   9039205|Varanga White Shi...|1399.0| White|         712|          3.75|"White solid
wove...|     945|       Varanga|
|  17988970|Flying Machine Wo...|1999.0| White|          24|          4.29|"<ul> <li> Li
ght ...|     309|Flying Machine|
|  13769098|Style Quotient Wo...|1399.0| White|         105|           4.1|White self de
sign...|     845|Style Quotient|
|  17923962|La Aimee Women Wh...|2294.0| White|         184|           4.1|<ul> <li> Whi
te w...|     489|      La Aimee|
+----------+--------------------+------+------+------------+--------------+------------
-------+--------+--------------+
only showing top 20 rows
```

In [96]:
```
# Showed unique brand names

fashionspark_df.select('brand_name','name').distinct().show()
```

```
+--------------+--------------------+
|    brand_name|                name|
+--------------+--------------------+
|           750|Roadster Women Pe...|
|      Roadster|Roadster Women Bl...|
```

```
|          null|{'Better Cotton I...|
|      Indo Era|Indo Era Women Bl...|
|      Roadster|The Roadster Life...|
|          Biba|Biba Red & Beige ...|
|          null|Wuxi Orange & Gol...|
|       Emprall|Emprall Multicolo...|
| Tokyo Talkies|Tokyo Talkies Bla...|
|          Biba|Biba Women Navy B...|
| Tokyo Talkies|Tokyo Talkies Wom...|
|         Libas|Libas Women Burgu...|
|         Bronz|Bronz Women Green...|
|   Belle Fille|Belle Fille Women...|
|         Anouk|Anouk Women Rust ...|
|          null| Do not bleach</p...|
|     Indibelle|Indibelle Women Y...|
|Clora Creation|Clora Creation Cr...|
|         Styli|Styli Green Geome...|
| Tokyo Talkies|Tokyo Talkies Wom...|
+--------------+--------------------+
only showing top 20 rows
```

In [97]:
```
# Multiple flitering and total count of H&M and Levis using 'union' method

black_HnM_clothes = fashionspark_df.filter((col('brand_name')=='H&M') & (col('colour')==
blue_levis_clothes = fashionspark_df.filter((col('brand_name')=='Levis') & (col('colour'
print("Black H&Ms: "+str(black_HnM_clothes.count()))
print("Blue Levis: "+str(blue_levis_clothes.count()))
print("Total Black H&M and Blue Levi CLothes: "+str(black_HnM_clothes.union(blue_levis_c
```

```
Black H&Ms: 45
Blue Levis: 27
Total Black H&M and Blue Levi CLothes: 72
```

In [162…
```
# Applied RDD

fashionsparkdf = spark.sparkContext.textFile('fashion_master.csv')
print(fashionsparkdf.first())
fashionsparkdf_header = fashionsparkdf.first()
fashionsparkdf_rest = fashionsparkdf.filter(lambda line: line!=fashionsparkdf_header)
print(fashionsparkdf_rest.first())
```

```
brand_id,brand_name,p_id,name,price,colour,brand,ratingCount,avg_rating,description,p_at
tributes
1,513,13158392,513 Women Black & Grey Woven-Design Kimono Shrug,1699.0,Black,513,136,4.5
7,"Grey and black woven-design Kimono Shrug, has a solid border<p>Acrylic<br>Machine-was
h</p>","{'Better Cotton Initiative': 'Regular', 'Fabric': 'Acrylic', 'Front Styling': 'O
pen Front', 'Hemline': 'Asymmetric', 'Length': 'Longline', 'Main Trend': 'Monochrome',
'Occasion': 'Casual', 'Pattern': 'Checked', 'Sleeve Length': 'Long Sleeves', 'Surface St
yling': 'NA', 'Sustainable': 'Regular', 'Wash Care': 'Machine Wash'}"
```

In [163…
```
# Using mapreduce, the number of products are counted

fashionsparkdf_rest.map(lambda x: x.split(";")).count()
```

Out[163]:
```
8302
```

Machine Learning

In [100…
```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
from sklearn import preprocessing
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [101…

```
# Used the black products dataframe

black_prdts_df.head()
```

Out[101]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes	brand
0	16920516	Masaba Woman Black Tulip Cape Set	30000.0	Black	Masaba	184	4.10	V-neckline Full sleeves Embr...	{'Add-Ons': 'NA', 'Better Cotton Initiative': ...	5
1	18585472	Mitera Black & Pink Floral Pure Linen Saree	22300.0	Black	Mitera	184	4.10	Design Details Black and pi...	{'Blouse': 'Blouse Piece', 'Blouse Fabric': 'P...	5
2	13168738	Justanned Plus Women Plus Size Black Solid Lea...	19999.0	Black	Justanned	184	4.10	Black solid jacket, has a spread collar, 6 poc...	{'Add-Ons': 'NA', 'Body Shape ID': '424', 'Bod...	4
3	17981980	Justanned Women Black Leather Lightweight Crop...	19998.0	Black	Justanned	184	4.10	Black solid lightweight biker jacket with zip...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	4
4	15880878	Justanned Women Black Striped Leather Crop Out...	15998.0	Black	Justanned	6	4.67	Black solid striped biker jacket with zip det...	{'Add-Ons': 'NA', 'Body Shape ID': '333,424', ...	4

In [102…

```
# Filtered further by dropping all rows with rating count equalling zero

black_prdts_mldf = black_prdts_df[black_prdts_df['ratingCount'] != 0]
```

In [103…

```
black_prdts_mldf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1049 entries, 0 to 1048
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   p_id          1049 non-null   object 
 1   name          1049 non-null   object 
 2   price         1049 non-null   float64
 3   colour        1049 non-null   object 
 4   brand         1049 non-null   object 
 5   ratingCount   1049 non-null   int64  
 6   avg_rating    1049 non-null   float64
 7   description   1049 non-null   object 
 8   p_attributes  1049 non-null   object 
 9   brand_id      1049 non-null   int64  
 10  brand_name    1049 non-null   object 
```

```
dtypes: float64(2), int64(2), object(7)
memory usage: 98.3+ KB
```

In [104…
```
# Chose to focus the model on average rating, rating count and price

focus_1 = black_prdts_mldf.loc[:,['avg_rating','ratingCount','price']]
```

In [105…
```
focus_1.head()
```

Out[105]:

	avg_rating	ratingCount	price
0	4.10	184	30000.0
1	4.10	184	22300.0
2	4.10	184	19999.0
3	4.10	184	19998.0
4	4.67	6	15998.0

In [106…
```
# Imported Normalizer in order to preprocess the focused data and bring them to scale

from sklearn.preprocessing import Normalizer

normalizer = Normalizer(norm='max')
```

In [107…
```
focus_1[['avg_rating', 'ratingCount', 'price']]  = normalizer.fit_transform(focus_1[['av
```

In [108…
```
focus_1.head()
```

Out[108]:

	avg_rating	ratingCount	price
0	0.000137	0.006133	1.0
1	0.000184	0.008251	1.0
2	0.000205	0.009200	1.0
3	0.000205	0.009201	1.0
4	0.000292	0.000375	1.0

> Using linear regression

In [109…
```
# Created the X and y dataframes

X = pd.DataFrame(focus_1['avg_rating'])
y = pd.DataFrame(focus_1['price'])
```

In [110…
```
#Trained the model on X for avg_rating and y for price

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [111…
```
X_train
```

Out[111]:

	avg_rating
796	0.002735
142	0.000750

550	0.002051
451	0.001822
209	0.001243
...	...
360	0.001641
466	0.001864
299	0.001465
493	0.001787
527	0.002051

839 rows × 1 columns

In [112…
```
# Assigned the model

model_LR = LinearRegression()
```

In [113…
```
# Fitted the model to the training set

model_LR.fit(X_train, y_train)
```

Out[113]:
```
LinearRegression()
```

In [114…
```
# Predicted using the test set

y_pred = model_LR.predict(X_test)
```

In [115…
```
#Taking a look into X_test with 20 percent test set and 80 percent train set

X_test
```

Out[115]:

	avg_rating
347	0.001689
614	0.002159
700	0.001995
947	0.003656
705	0.002572
...	...
903	0.003420
620	0.002169
1005	0.004127
924	0.003162
964	0.004104

210 rows × 1 columns

In [116…
```
# Prediction of first 10 results
```

```
y_pred[0:10]
```

Out[116]:
```
array([[0.99252353],
       [0.99271392],
       [0.99264764],
       [0.99332   ],
       [0.99288113],
       [0.9921499 ],
       [0.99241951],
       [0.99256186],
       [0.99214501],
       [0.9928168 ]])
```

In [117…
```
# Checked accuracy of the model

model_LR.score(X_test, y_test)
```

Out[117]:
```
0.0005899455605113957
```

In [118…
```
# Explored the errors

print('Mean Absolute Error: ', mean_absolute_error(y_test, y_pred))
print('Mean Squared Error: ', mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error: ', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error:  0.014983112465987432
Mean Squared Error:  0.0037374819138062923
Root Mean Squared Error:  0.06113494838311628
```

In [ ]:
```

```

> Using multiple regression

In [119…
```
# Used more independent variables for a multiple regression

X1 = pd.DataFrame(focus_1[['avg_rating', 'ratingCount']])
X1
```

Out[119]:

	avg_rating	ratingCount
0	0.000137	0.006133
1	0.000184	0.008251
2	0.000205	0.009200
3	0.000205	0.009201
4	0.000292	0.000375
...	...	...
1044	0.008233	0.369478
1045	0.008970	0.214141
1046	0.008577	1.000000
1047	0.010276	0.461153
1048	0.015254	0.396610

1049 rows × 2 columns

In [120…
```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y, test_size=0.2, random_sta
```

In [121…
```
model_LR.fit(X1_train, y1_train)
```

Out[121]:
```
LinearRegression()
```

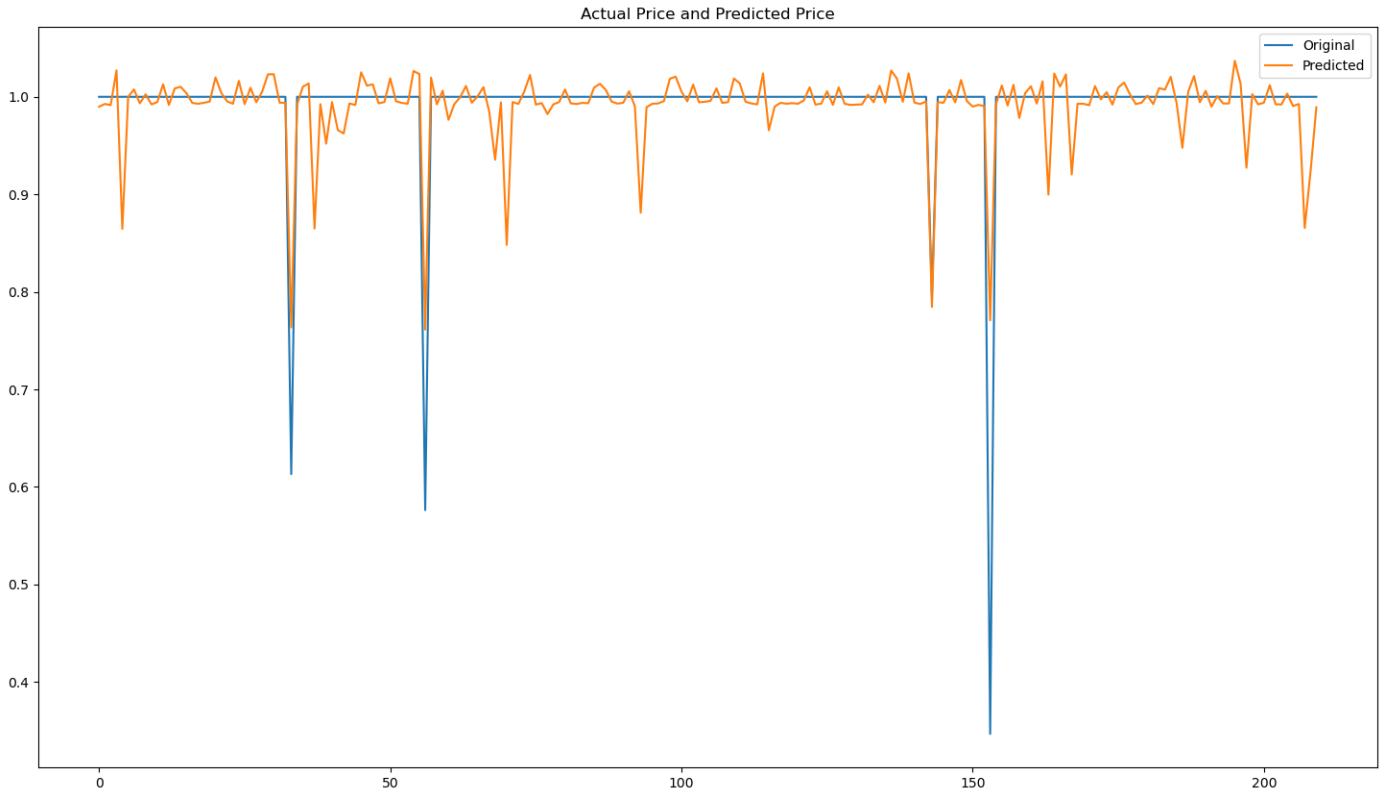
In [122…
```
y1_pred = model_LR.predict(X1_test)
```

In [123…
```
# Results using multiple regression

y1_pred[0:10]
```

Out[123]:
```
array([[0.98984365],
       [0.99269703],
       [0.99142294],
       [1.02715041],
       [0.86457251],
       [0.9999197 ],
       [1.0075999 ],
       [0.99334251],
       [1.00254201],
       [0.99226031]])
```

In [124…
```
# In order to see the graphical corretation between actual and predicted outputs,set plo

import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
```

In [125…
```
plt.figure(figsize=(18,10))
x_axis=range(len(y1_test))
plt.plot(x_axis, y1_test, label='Original')
plt.plot(x_axis, y1_pred, label='Predicted')
plt.title('Actual Price and Predicted Price')
plt.legend()
plt.show()
```



In [126…
```
model_LR.score(X1_test, y1_test)
```

Out[126]:
```
0.4991957115238791
```

In [127…
```
# Multiple regression errors

print('Mean Absolute Error: ', mean_absolute_error(y1_test, y1_pred))
print('Mean Squared Error: ', mean_squared_error(y1_test, y1_pred))
print('Root Mean Squared Error: ', np.sqrt(mean_squared_error(y1_test, y1_pred)))
```

```
Mean Absolute Error:  0.018288417068626876
Mean Squared Error:  0.0018728518511712251
Root Mean Squared Error:  0.04327645839450388
```

In [ ]:
```

```

> Using support vector regression

In [128…
```
# Used support vector regression to back up results

from sklearn.svm import SVR
```

In [129…
```
# Modelled using linear SVR

model_SV = SVR(kernel = 'linear')
```

In [130…
```
X2_train, X2_test, y2_train, y2_test = train_test_split(X1, y, test_size=0.2, random_sta
```

In [131…
```
model_SV.fit(X2_train, y2_train)
```

```
C:\Users\IKENNA\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[131]:
```
SVR(kernel='linear')
```

In [132…
```
y2_pred = model_SV.predict(X2_test)
```

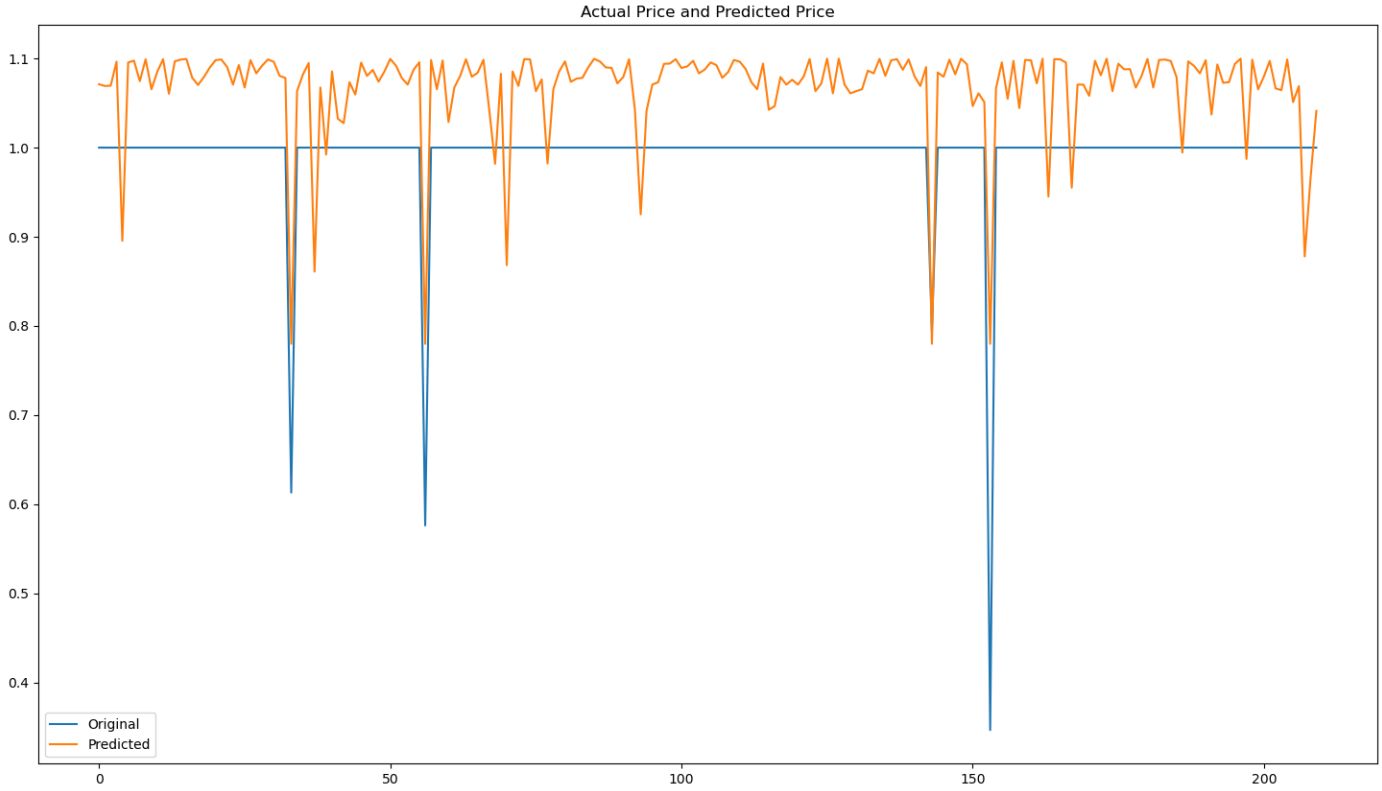
In [133…
```
# Results using linear SVR

y2_pred
```

Out[133]:
```
array([1.07108304, 1.06915003, 1.06945332, 1.09639587, 0.89548159,
       1.09551287, 1.09756698, 1.07454359, 1.09911093, 1.06550088,
       1.08505128, 1.0992009 , 1.06034692, 1.09679001, 1.0987164 ,
       1.09944984, 1.07833846, 1.07028243, 1.07911771, 1.08962797,
       1.09805191, 1.09888117, 1.09033654, 1.07070078, 1.09263501,
       1.06742687, 1.09808921, 1.08332527, 1.09193376, 1.09890985,
       1.09629324, 1.0805203 , 1.0780602 , 0.7799143 , 1.06333399,
       1.08194874, 1.09497493, 0.86094055, 1.06744505, 0.99221978,
       1.08542883, 1.03238778, 1.02747983, 1.07337915, 1.05955087,
       1.0953917 , 1.0804729 , 1.0871578 , 1.07397431, 1.08542515,
       1.0995442 , 1.0917386 , 1.07785102, 1.07070078, 1.08738744,
       1.09575862, 0.77991085, 1.09820949, 1.06550088, 1.09769577,
       1.02874378, 1.06742687, 1.08031213, 1.09918767, 1.07949407,
       1.08394167, 1.09857018, 1.04217575, 0.98185657, 1.08283539,
       0.86815753, 1.08542883, 1.06915003, 1.09931028, 1.09895835,
       1.06333399, 1.07659288, 0.9821775 , 1.06550088, 1.08542883,
       1.0967297 , 1.0738339 , 1.0773577 , 1.07830609, 1.09013359,
       1.09963641, 1.09665531, 1.08996371, 1.08944429, 1.07210377,
       1.07911771, 1.09919941, 1.04282513, 0.92507641, 1.04122256,
       1.07070078, 1.07337915, 1.09401743, 1.09427361, 1.09924066,
       1.08929743, 1.09116203, 1.09741066, 1.08332527, 1.08762174,
```

```
1.09557902, 1.09248762, 1.07833846, 1.08465385, 1.09830133,
1.0964237 , 1.08836413, 1.07337915, 1.06550088, 1.09432283,
1.04239859, 1.04658443, 1.07911771, 1.07070078, 1.07612175,
1.07070078, 1.07998703, 1.09944558, 1.06333399, 1.07183396,
1.0998328 , 1.06087798, 1.09978591, 1.07070078, 1.06087798,
1.06333399, 1.06550088, 1.08633412, 1.08332527, 1.09948694,
1.08052684, 1.09798051, 1.09922378, 1.08735212, 1.09891188,
1.08018749, 1.06915003, 1.09033654, 0.77994674, 1.08423494,
1.07948682, 1.09866001, 1.08230531, 1.09968922, 1.09342942,
1.04658443, 1.06087798, 1.05105191, 0.77992556, 1.06649141,
1.09562257, 1.05483155, 1.09729379, 1.04439265, 1.09835253,
1.09796065, 1.07210377, 1.09969964, 0.94512987, 1.09920708,
1.0990619 , 1.09551186, 0.95504319, 1.07070078, 1.07070078,
1.05807086, 1.09732745, 1.08121715, 1.0997784 , 1.06333399,
1.09410583, 1.08783075, 1.08808544, 1.06735396, 1.0805203 ,
1.09951722, 1.06742687, 1.09835559, 1.09875786, 1.09719913,
1.07911771, 0.99442736, 1.09674997, 1.09165078, 1.08328189,
1.09806185, 1.03716066, 1.09329963, 1.0727563 , 1.07337915,
1.09380746, 1.09979159, 0.98734195, 1.09864684, 1.06541911,
1.07984319, 1.09725156, 1.06649141, 1.06445028, 1.09889278,
1.05105191, 1.06900241, 0.87792025, 0.96525093, 1.04122256])
```

In [134…

```
plt.figure(figsize=(18,10))
x_axis=range(len(y2_test))
plt.plot(x_axis, y2_test, label='Original')
plt.plot(x_axis, y2_pred, label='Predicted')
plt.title('Actual Price and Predicted Price')
plt.legend()
plt.show()
```



In [135…

```
model_SV.score(X2_test, y2_test)
```

Out[135]:

```
-1.118273164926109
```

In [136…

```
# Errors using linear SVR

print('Mean Absolute Error: ', mean_absolute_error(y2_test, y2_pred))
print('Mean Squared Error: ', mean_squared_error(y2_test, y2_pred))
print('Root Mean Squared Error: ', np.sqrt(mean_squared_error(y2_test, y2_pred)))
```

```
Mean Absolute Error:  0.08230146899683417
Mean Squared Error:  0.007921681002952026
Root Mean Squared Error:  0.08900382577705312
```

> It was observed that the multiple linear regression errors and SVR errors had similar values

In [ ]:

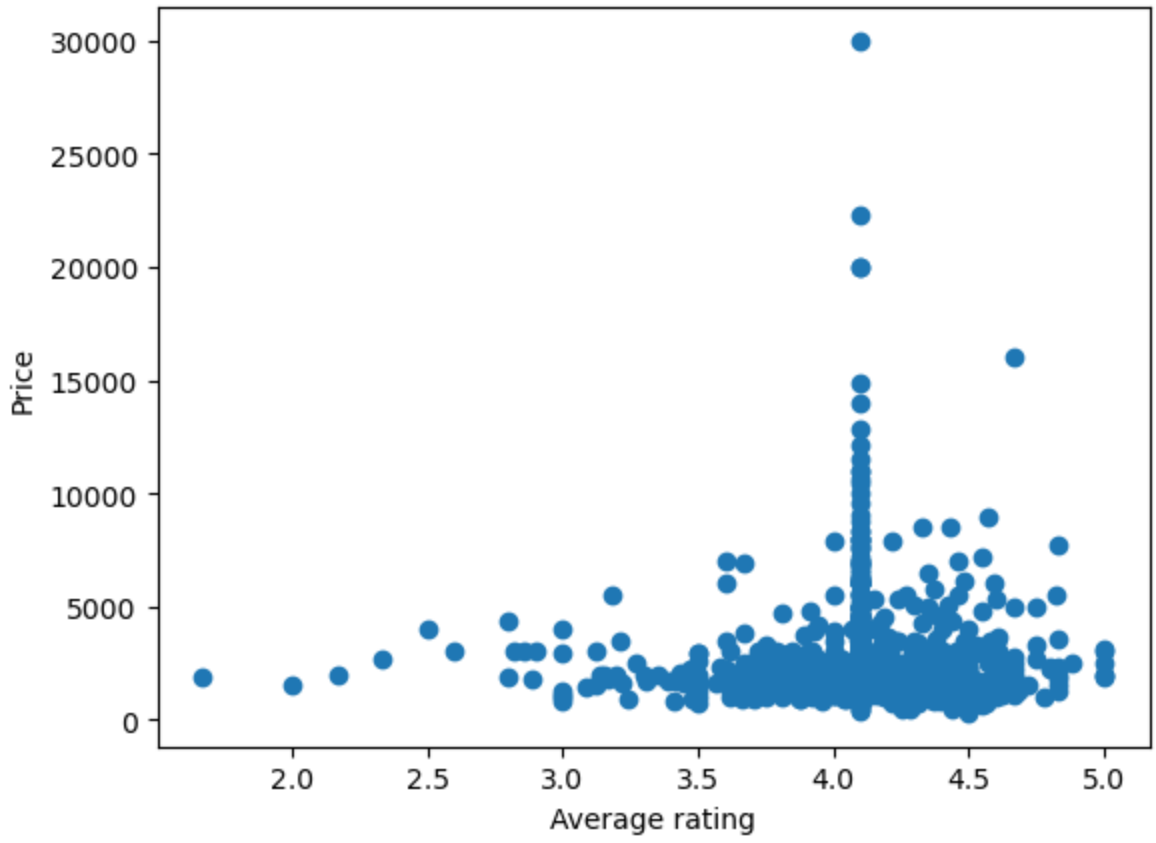
> Using KMeans Clustering

In [137…
```
# Imported KMeans and MinMaxScaler from the sklearn library

from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
```

In [138…
```
# Filtered for the attributes to be analysed using KMeans

focus_2 = black_prdts_mldf.loc[:,['avg_rating','price']]
```

In [139…
```
# Reasserted attributes in focus

rating = focus_2['avg_rating']
pricing = focus_2['price']
```

In [140…
```
# Examined the data points

plt.scatter(rating, pricing)
plt.xlabel('Average rating')
plt.ylabel('Price');
```



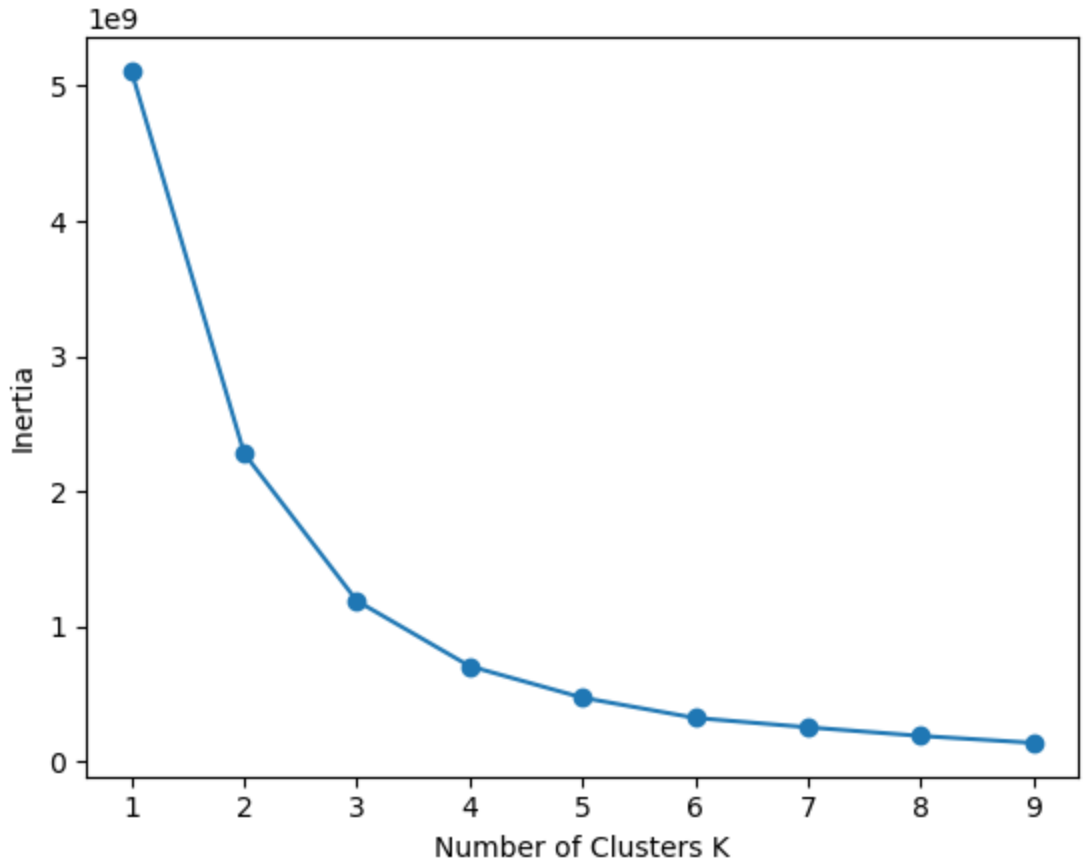
In [141…
```
# Calculated the elbow point for best number of clusters k using sum of squares error ss
```

```
k_range=range(1,10)
sse=[]
for k in k_range:
    km = KMeans(n_clusters=k)
    km.fit(focus_2[['avg_rating','price']])
    sse.append(km.inertia_)

sse
```

```
C:\Users\IKENNA\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarnin
g: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks
than available threads. You can avoid it by setting the environment variable OMP_NUM_THR
EADS=5.
  warnings.warn(
```

Out[141]:
```
[5112037915.07396,
 2281052220.373087,
 1189361837.4235616,
 706125223.3310348,
 472795094.5967941,
 323034080.88589936,
 253512698.77917242,
 189728891.35611627,
 138311786.7314675]
```

In [142…
```
# Plotted inertia vs number of clusters to better determine the elbow point

plt.xlabel('Number of Clusters K')
plt.ylabel('Inertia')
plt.plot(k_range, sse, '-o');
```



In [143…
```
# Set an instance of KMeans with number of clusters = 4
# 4 was chosen as the elbow point as there was no significant decrease in inertia from t

km= KMeans(n_clusters=4, random_state=0)
```

In [144…
```
# Fitted the KMeans to focus_2 df to determine the cluster groupings
```

```
y3_pred= km.fit_predict(focus_2)

y3_pred
```

Out[144]:
```
array([2, 2, 2, ..., 1, 1, 1])
```

In [145…
```
set(y3_pred) # To see the unnique values of y3_pred, "set" function was used
```

Out[145]:
```
{0, 1, 2, 3}
```

In [146…
```
km.cluster_centers_  # .cluster_centers_ attribute was used to determine the co-ordinate
```

Out[146]:
```
array([[4.09540650e+00, 3.52556098e+03],
       [4.10180690e+00, 1.69437379e+03],
       [4.18142857e+00, 1.95991429e+04],
       [4.13915493e+00, 7.31459155e+03]])
```

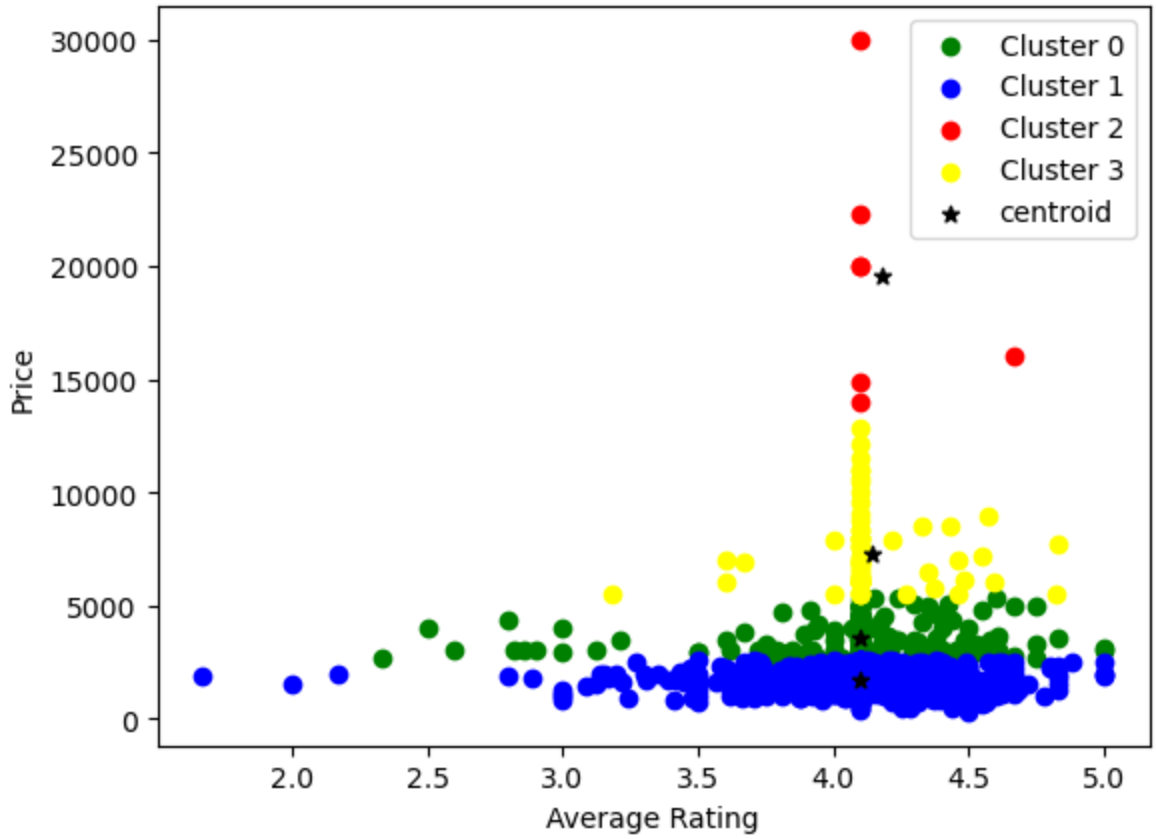
In [147…
```
# Cluster column was created

focus_2['cluster']=y3_pred
focus_2
```

Out[147]:

	avg_rating	price	cluster
0	4.10	30000.0	2
1	4.10	22300.0	2
2	4.10	19999.0	2
3	4.10	19998.0	2
4	4.67	15998.0	2
...	...	...	...
1044	4.10	498.0	1
1045	4.44	495.0	1
1046	4.28	489.0	1
1047	4.10	399.0	1
1048	4.50	295.0	1

1049 rows × 3 columns

In [148…
```
# Dataframes were created and plotted with regards to their cluster grouping of 0,1,2 or

focus_2_c0= focus_2[focus_2.cluster==0]
focus_2_c1= focus_2[focus_2.cluster==1]
focus_2_c2= focus_2[focus_2.cluster==2]
focus_2_c3= focus_2[focus_2.cluster==3]

plt.scatter(focus_2_c0['avg_rating'], focus_2_c0['price'], label='Cluster 0', color='gre
plt.scatter(focus_2_c1['avg_rating'], focus_2_c1['price'], label='Cluster 1',color='blue
plt.scatter(focus_2_c2['avg_rating'], focus_2_c2['price'], label='Cluster 2',color='red'
plt.scatter(focus_2_c3['avg_rating'], focus_2_c3['price'], label='Cluster 3',color='yell
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1],color='black', marker='*'


plt.xlabel('Average Rating')
plt.ylabel('Price')
plt.legend();
```



In [149…

```
# MinMaxScaler was used to preprocess the price and average rating entries to a range sc

scaler=MinMaxScaler()
scaler.fit(focus_2[['price']])
focus_2[['price']]= scaler.transform(focus_2[['price']])

scaler.fit(focus_2[['avg_rating']])
focus_2[['avg_rating']]= scaler.transform(focus_2[['avg_rating']])

focus_2
```

Out[149]:

	avg_rating	price	cluster
0	0.729730	1.000000	2
1	0.729730	0.740784	2
2	0.729730	0.663323	2
3	0.729730	0.663289	2
4	0.900901	0.528632	2
...	...	...	...
1044	0.729730	0.006834	1
1045	0.831832	0.006733	1
1046	0.783784	0.006531	1
1047	0.729730	0.003501	1
1048	0.849850	0.000000	1

1049 rows × 3 columns

In [150…

```
# focus_2 df was sliced for needed columns and instanced as kfocus_2
```

```
kfocus_2 = focus_2[['avg_rating','price']]
kfocus_2
```

Out[150]:

	avg_rating	price
0	0.729730	1.000000
1	0.729730	0.740784
2	0.729730	0.663323
3	0.729730	0.663289
4	0.900901	0.528632
...	...	...
1044	0.729730	0.006834
1045	0.831832	0.006733
1046	0.783784	0.006531
1047	0.729730	0.003501
1048	0.849850	0.000000

1049 rows × 2 columns

In [151…

```
# kfocus_2 was fitted to KMeans

y3_pred = km.fit_predict(kfocus_2)
y3_pred
```

Out[151]:

```
array([0, 0, 0, ..., 3, 2, 3])
```

In [152…

```
# The updated centroids after scaling are generated

km.cluster_centers_
```

Out[152]:

```
array([[0.74328382, 0.2781682 ],
       [0.47637779, 0.06030948],
       [0.71959569, 0.06452473],
       [0.84439485, 0.0601119 ]])
```

In [153…

```
# New_cluster column is generated to show the updated cluster groupings after scaling

focus_2['New_cluster']=y3_pred
focus_2
```

Out[153]:

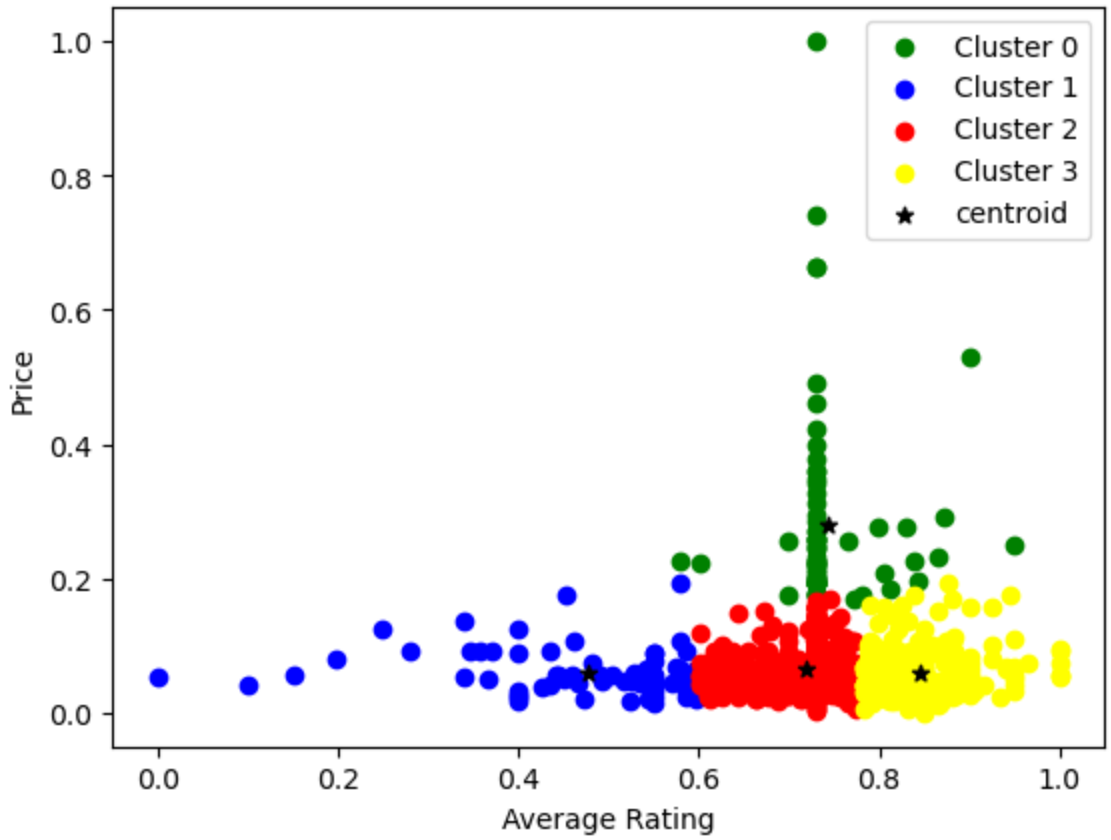
	avg_rating	price	cluster	New_cluster
0	0.729730	1.000000	2	0
1	0.729730	0.740784	2	0
2	0.729730	0.663323	2	0
3	0.729730	0.663289	2	0
4	0.900901	0.528632	2	0
...	...	...	...	...
1044	0.729730	0.006834	1	2
1045	0.831832	0.006733	1	3

1046	0.783784	0.006531	1	3
1047	0.729730	0.003501	1	2
1048	0.849850	0.000000	1	3

1049 rows × 4 columns

In [154…

```
# Dataframe was sliced and plotted with regards to the updated cluster groupings

focus_2_c0= focus_2[focus_2.New_cluster==0]
focus_2_c1= focus_2[focus_2.New_cluster==1]
focus_2_c2= focus_2[focus_2.New_cluster==2]
focus_2_c3= focus_2[focus_2.New_cluster==3]

plt.scatter(focus_2_c0['avg_rating'], focus_2_c0['price'], label='Cluster 0', color='gre
plt.scatter(focus_2_c1['avg_rating'], focus_2_c1['price'], label='Cluster 1',color='blue
plt.scatter(focus_2_c2['avg_rating'], focus_2_c2['price'], label='Cluster 2',color='red'
plt.scatter(focus_2_c3['avg_rating'], focus_2_c3['price'], label='Cluster 3',color='yell
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1],color='black', marker='*'


plt.xlabel('Average Rating')
plt.ylabel('Price')
plt.legend();
```



> Based on the KMeans cluster analysis, it was observed that most products were above average and had an affordable price, both of which are good for the ever-growing fashion industry. The clusters were largely divided into those with medium average rating and low price, above medium average rating and low price, high average rating and low price and above medium to high average rating and a higher price. Based on the cluster a product fell into, fashion brands could determine the changes required to meet the needs of their shareholders and consumers.

In [ ]:

Visualizations

In [155…
```
# Imported set plots to be embedded inline

import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
```

In [156…
```
# Adopted the products reviewed with low ratings for the visualizations

review_prdts_df.head()
```

Out[156]:

	p_id	name	price	colour	brand	ratingCount	avg_rating	description	p_attributes	bra
0	1411010	Dupatta Bazaar Beige Striped Chanderi Cotton S...	449.0	Beige	Dupatta Bazaar	6	2.83	Beige striped dupatta, has a zari border ...	{'Ornamentation': 'Zari', 'Print or Pattern Ty...	
1	14745736	Baby Lakshmi Girls Orange & Off White Ready to...	698.0	Orange	Baby Lakshmi	1	2.00	Orange and off white solid lehenga choli, O...	{'Blouse Closure': 'Button', 'Blouse Fabric': ...	
2	13278842	Bronz Women Off-White Printed Tunic	699.0	Off White	Bronz	12	2.83	Off-White printed Tunic, has a round neck, and...	{'Body or Garment Size': 'To-Fit Denotes Body ...	
3	18290008	Go Colors Women Black Tapered Fit Trousers	799.0	Black	Go Colors	2	3.00	Black woven trousers Tape...	{'Add-Ons': 'NA', 'Body Shape ID': '443,333,42...	
4	11524932	Vishudh Women's Mustard & Pink Striped Tunic	849.0	Mustard	Vishudh	13	2.31	Mustard and Pink striped Tunic, has a mandarin...	{'Body or Garment Size': 'Garment Measurements...	

In [157…
```
review_prdts_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134 entries, 0 to 133
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
```

```
---  ------         --------------  -----
 0   p_id           134 non-null    object
 1   name           134 non-null    object
 2   price          134 non-null    float64
 3   colour         134 non-null    object
 4   brand          134 non-null    object
 5   ratingCount    134 non-null    int64
 6   avg_rating     134 non-null    float64
 7   description    134 non-null    object
 8   p_attributes   134 non-null    object
 9   brand_id       134 non-null    int64
 10  brand_name     134 non-null    object
dtypes: float64(2), int64(2), object(7)
memory usage: 11.6+ KB
```

In [158…

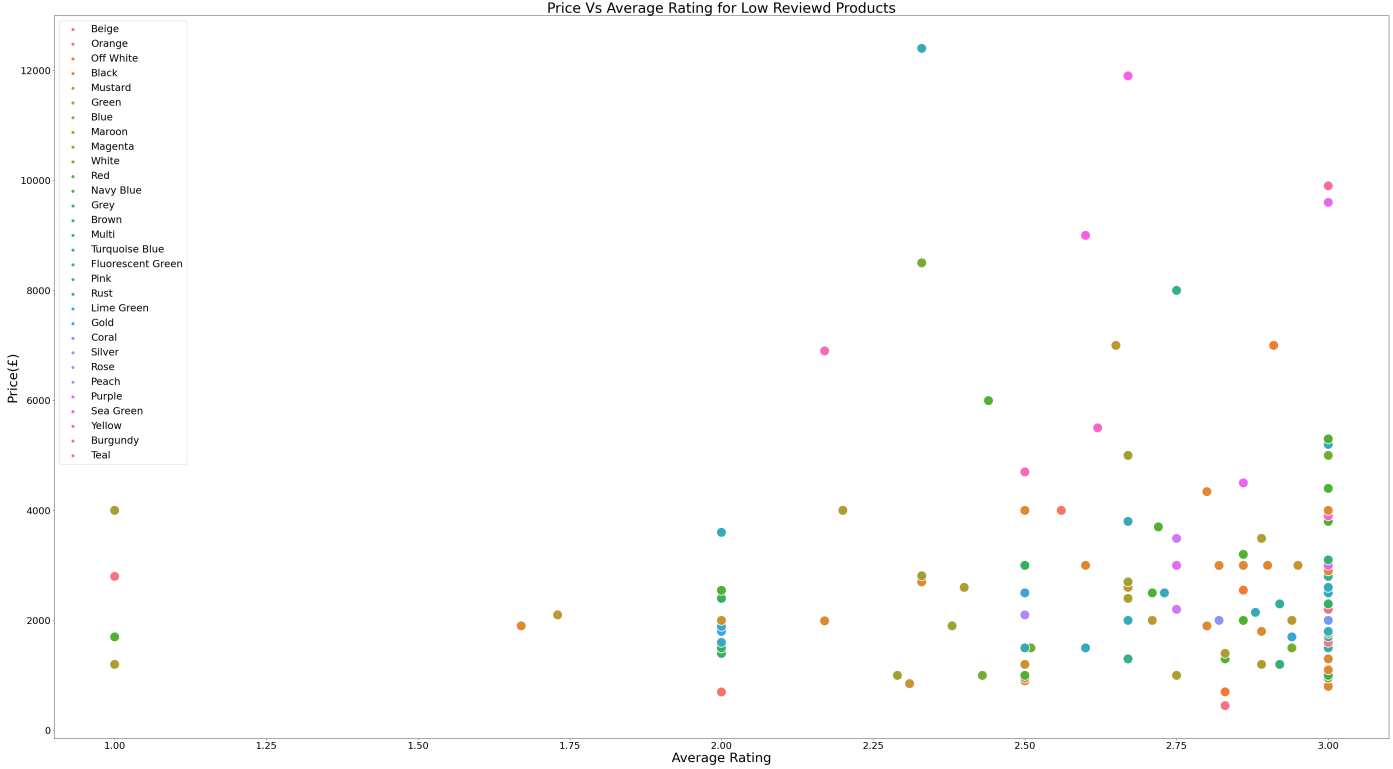
```
# The colour column was adopted for the hue parameter for the seaborn scatterplot

list(review_prdts_df['colour'].unique())
```

Out[158]:

```
['Beige',
 'Orange',
 'Off White',
 'Black',
 'Mustard',
 'Green',
 'Blue',
 'Maroon',
 'Magenta',
 'White',
 'Red',
 'Navy Blue',
 'Grey',
 'Brown',
 'Multi',
 'Turquoise Blue',
 'Fluorescent Green',
 'Pink',
 'Rust',
 'Lime Green',
 'Gold',
 'Coral',
 'Silver',
 'Rose',
 'Peach',
 'Purple',
 'Sea Green',
 'Yellow',
 'Burgundy',
 'Teal']
```

In [160…

```
# Scatter plot done with price against average rating

sb.scatterplot(data=review_prdts_df, x="avg_rating", y="price", hue=review_prdts_df['col

plt.ylabel('Price(£)', fontsize=30)
plt.xlabel('Average Rating', fontsize=30)
plt.legend(fontsize=23);

plt.xticks(fontsize=22, rotation = 0);
plt.yticks(fontsize=22);

plt.title('Price Vs Average Rating for Low Reviewd Products', fontsize=32)

from matplotlib import rcParams
rcParams['figure.figsize'] = 55,30
```



> Fashion brands could use this visual presentation to easily determine which colours have poor ratings and the relationship between average rating and price of the poorly reviewed products. They could also determine the products in most need of improvement and see where discounts can be given in order to improve the appeal of given products.

In [161…

```
# Created the reviewprdts csv file for further analysis on Power BI

review_prdts_df.to_csv('reviewprdts.csv', index=False)
```