Analyzing Fashion Trends and Customer Preferences Using Big Data Technologies

Avirit Singh, Jay Joshi ,Tanu Datt ,Vaibhavi Rao,Varun Patil, Pragya Priyadarshini

San Jose State University
1 Washington Sq, San Jose, CA 95192

avirit.singh@sjsu.edu
jaynarendrabhai.joshi@sjsu.edu
tanu.datt@sjsu.edu
vaibhavi.rao@sjsu.edu
varunpruthviraj.patil@sjsu.edu
pragya.priyadarshini@sjsu.edu

GitHub link: https://github.com/JayJoshi4520/DATA-228-Project.git

Abstract— **In this project, we aim to leverage the Fashion Product Images dataset from Kaggle to analyze and uncover hidden insights into fashion trends and customer preferences. The dataset contains over 44,000 images of fashion products along with metadata such as product categories, gender, and usage occasions. Using big data technologies such as Apache Spark and Hadoop, we will process and analyze the dataset to identify patterns in fashion trends across different demographics and seasons. Additionally, we will employ machine learning algorithms to predict product popularity and customer preferences based on product attributes such as color, material, and style. This analysis can provide insights for fashion retailers to optimize inventory and marketing strategies. Our goal is to provide actionable insights using a scalable and robust approach, offering value to fashion retailers and brands in the competitive fashion industry.**

***Keywords*——HDFS, YARN, Apache Spark, Spark SQL, Spark MLlib, JSON.**

I. INTRODUCTION

The fashion business serves as a symbol of creativity and economic activities, driving major trends in the world economy. In this digital era, consumer behavior and preferences are rapidly changing, the success of the industry depends on correctly anticipating and responding to these changes. The emergence of online retail, social media, and digital marketing produces massive amounts of data. Everything from high-resolution fashion product images to voluminous logs of user interaction is apparently capable of uncovering new information about ever-changing fashion trends and customer preference. This potential can be tapped only if it is not just complex but big. The analytics tools used, in general, fall short while processing and extracting actionable insights out of this multivariate-multi-volume data. The project at hand deals with reviewing a Kaggle dataset on Fashion Product Images using advanced big data technologies, namely Apache Spark and HDFS. This paper, therefore, integrates machine learning algorithms to interpret current fashion trends and make full-scale forecasts of future demand with the aim of revolutionizing inventory control, marketing campaigns, and enhancing the overall customer experience in the fashion retail sector.

II. PROBLEM STATEMENT

The most important challenge for fashion retailers is having to process efficiently and analyze large, varied data in order to arrive at actionable insights for strategic decisions. This project addresses the requirement for advanced analytic models capable of processing such huge quantities of fashion data to predict future trends and consumer preferences with accuracy. Big data technology coupled with machine learning is applied herein to enhance inventory optimization and marketing effectiveness with a view toward improvement in profitability and customer satisfaction for apparel retailers.

III. PROPOSED METHOD

The proposed approach, represented in Figure 1, performs a big data pipeline that was intended for the processing and analysis of the Kaggle Fashion Product Images Dataset. First, it goes into storage, which Hadoop is able to provide with its HDFS for distributed and fault-tolerant storage of the large dataset of fashion images with metadata. Resource management is accomplished through Hadoop YARN, which distributes the computation tasks efficiently across the cluster.

Data processing and scalable computation are done through Apache Spark integrated with Hadoop. Cleaning metadata, transformation of image data, or the running of queries on structured data is managed through Spark SQL. Besides that, application of machine learning-for example, in product categorization or recommendations based on similarities-can be realized by using Spark MLlib. Extract final insights, such as popular fashion trends or customer segmentation, for business decisions. This figure will be included in the report for the purpose of graphically explaining the flow of data through this pipeline, hence giving a better understanding of the methodology.

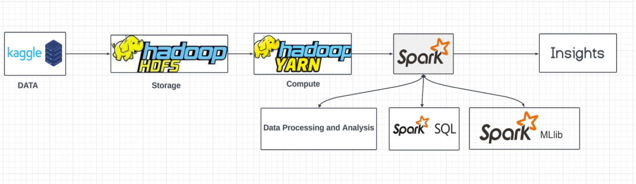


Figure no 1: Workflow

IV. DATA OVERVIEW

Structure

The dataset is organized into two main directories, complemented by two CSV files for streamlined data access:

Directories:

images/: Contains 44,400 high-resolution product images in JPG format, each associated with a unique product ID.

styles/: Includes metadata files in JSON format, offering detailed product descriptions and attributes.

Files:

styles.csv: A CSV file with 12 columns that maps product IDs to key attributes such as categories, display names, and additional labels.

images.csv: Provides supplementary information about the image data.

Content:

Each product has a unique product ID that links the image, metadata, and records in the CSVs. The styles.csv contains product categories, attributes, and display names, and it'll be the main mapping file. Metadata in JSON files provides further context of the products beyond the visual information on the directory named images/.

Summary:

Total Files: ~88,900
Directories: 2 (images, styles)
File Formats: JPG, JSON, CSV
Size: ~23.1 GB

It is cumulative in structure in this way, allowing access and analysis to be much easier, while bringing the various visual data, metadata, and categorical information together in support of deep exploration and research in the e-commerce domain.

V. DATA STORAGE AND COMPUTATION

Handling a large dataset of approximately 15 GB of fashion product images in a big data environment will be managed using Hadoop's ecosystem; the storage and computation processes, respectively. Each component will play an important role in the aforementioned respect, which is explained as follows:

HDFS - Hadoop Distributed File System

Data Storage: HDFS is the backbone of storing data in the Hadoop ecosystem. It works its best when storing a large amount of data on multiple machines in a distributed manner. It offers high fault tolerance and provides increased access speeds for data, which is very important for efficient handling of large volumes of data.

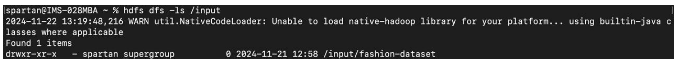
YARN- Yet Another Resource Negotiator.

Computation Management: YARN handles computing resources in the Hadoop ecosystem. It offers general resource management to support various models of distributed computing for processing computational tasks carried out among nodes in a cluster. YARN further extends scalability with better cluster utilization efficiency, which is rather essential for performing extensive data processing duties, like those required for the large-scale fashion dataset.

Command Used for uploading data from local machine to HDFS:
$hdfs dfs -put /local/path/to/file /hdfs/destination/path

/local/path/to/file: The full path to the file on your local machine.

/hdfs/destination/path: The target directory or path in HDFS.



```
spartan@IMS-028MBA ~ % hdfs dfs -ls /input
2024-11-22 13:19:48,216 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
Found 1 items
drwxr-xr-x   - spartan supergroup          0 2024-11-21 12:58 /input/fashion-dataset
```

Figure no 2. Upload data to HDFS from local machine

Spark on YARN:
It integrates Spark with YARN. YARN provides the compute used to carry out data processing and analysis more effectively. Spark provides extremely high-speed in-memory processing, which is very useful, especially in iterative algorithms for data analysis and machine learning. Data Processing: It would also provide a distributed way of handling data using Spark on YARN, whereby complex transformation and aggregation of data would be done along with much-needed machine learning workflows to derive meaningful insights from the fashion dataset.

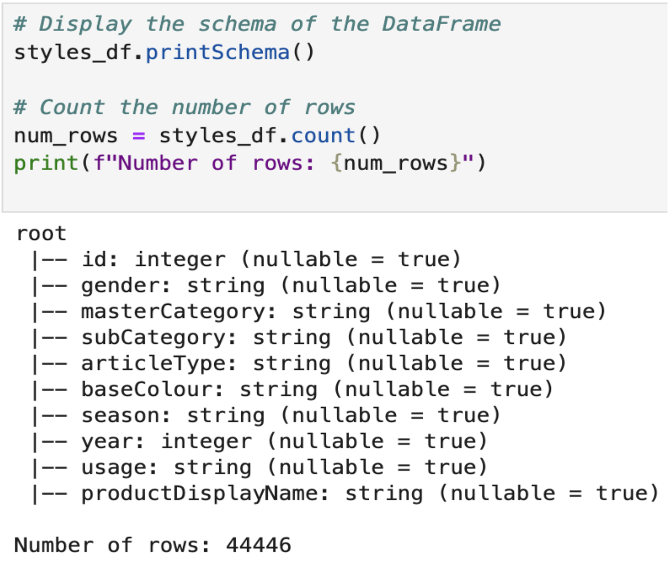
VI. DATA PRE-PROCESSING

The pre-processing script for the fashion dataset has handled the JSON files efficiently by defining a schema with all the details required to handle nested structures and assure coherence in data types. That exact schema is used to read multiple JSON files into a Spark DataFrame. Such complex fields as `meta` and `data` have been flattened into separate fields, making querying easier, for instance: `meta.code`, `meta.requestId`, `data.id`, `data.price`, `data.discountedPrice`, `data.styleType`, `data.productTypeId`, and `data.articleNumber`. Key transformations include the explosion of arrays such as the `data_crossLinks` into individual rows with columns such as `crossLink_key` and `crossLink_value`. It also involves dropping columns that are either irrelevant or redundant, including `notification`, `data_styleImages`, `data_visualTag`, `data_articleAttributes_Pattern`, and `data_brandUserProfile`. The process further caters for cleaning of data by treating missing or invalid values in critical columns.

This code snippet checks for null or NaN values by using a combination of isNull and isnan on the column named label and prints out the counts for validation. It filters out rows containing null or NaN in the label column by setting a condition: col("label").isNotNull() & ~isnan(col("label")); any remaining null or NaN values in label are replaced with 0 using the fillna() method. These steps ensure that there is no invalid or incomplete data within the final dataset. The cleaned DataFrame, that this would yield, focuses on only some critical attributes of product details-`data_id`, `data_price`, `data_brandName`, `data_baseColour`, `data_fashionType`, metadata -`meta_code`, `meta_requestId`, and relationships-`data_crossLinks` that will help in easy analysis and modeling.

This process ensures that the data is organized, cleaned, and prepared for further EDA, feature engineering, and machine learning applications. This can be further enriched by adding more validation and feature derivations such as calculation of discount, data.price - data.discountedPrice, or brand trends analysis, data.brandName, data.season, data.year. The output is really well prepared for storage in the analytics workflow.

VII. EXPLORATORY DATA ANALYSIS

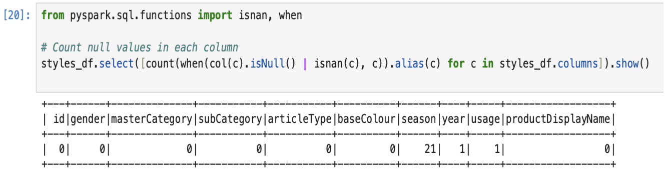


```
# Display the schema of the DataFrame
styles_df.printSchema()

# Count the number of rows
num_rows = styles_df.count()
print(f"Number of rows: {num_rows}")
```

```
root
 |-- id: integer (nullable = true)
 |-- gender: string (nullable = true)
 |-- masterCategory: string (nullable = true)
 |-- subCategory: string (nullable = true)
 |-- articleType: string (nullable = true)
 |-- baseColour: string (nullable = true)
 |-- season: string (nullable = true)
 |-- year: integer (nullable = true)
 |-- usage: string (nullable = true)
 |-- productDisplayName: string (nullable = true)

Number of rows: 44446
```

Figure no 3. Schema of style.csv

In figure no 2 we observe that
styles_df.printSchema(): Displays the schema of the DataFrame, detailing each column's name, data type, and nullability. The schema reveals that the dataset contains 10 columns, with data types including integer for id and year, and string for the remaining columns.
styles_df.count(): Calculates the total number of rows in the dataset. The result shows that the dataset contains 44,446 rows.

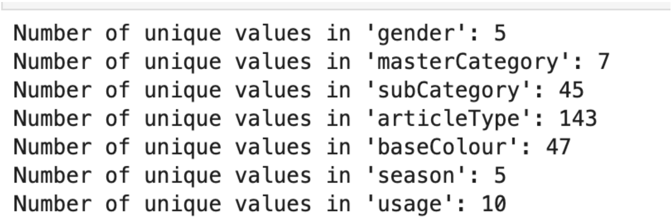


```
[20]: from pyspark.sql.functions import isnan, when

      # Count null values in each column
      styles_df.select([count(when(col(c).isNull() | isnan(c), c)).alias(c) for c in styles_df.columns]).show()
```

```
+---+------+--------------+-----------+-----------+----------+------+----+-----+------------------+
| id|gender|masterCategory|subCategory|articleType|baseColour|season|year|usage|productDisplayName|
+---+------+--------------+-----------+-----------+----------+------+----+-----+------------------+
|  0|     0|             0|          0|          0|         0|    21|   1|    1|                 0|
+---+------+--------------+-----------+-----------+----------+------+----+-----+------------------+
```

Figure no 4. Null values in style.csv

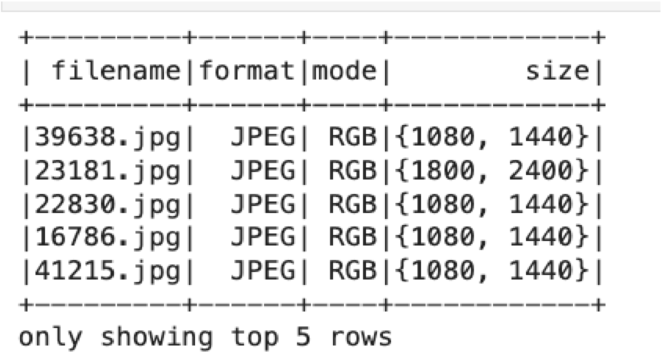
In figure no 3 we observe that
from pyspark.sql.functions import isnan, when: Imports functions to handle null and NaN values.
styles_df.select([...]).show(): Iterates over all columns and checks for null (isNull) or NaN (isnan) values using conditional logic.
The results show that:season has 21 null values, usage and year each have 1 null value. All other columns contain 0 null values.



```
Number of unique values in 'gender': 5
Number of unique values in 'masterCategory': 7
Number of unique values in 'subCategory': 45
Number of unique values in 'articleType': 143
Number of unique values in 'baseColour': 47
Number of unique values in 'season': 5
Number of unique values in 'usage': 10
```

Figure no 5. unique values in style.csv

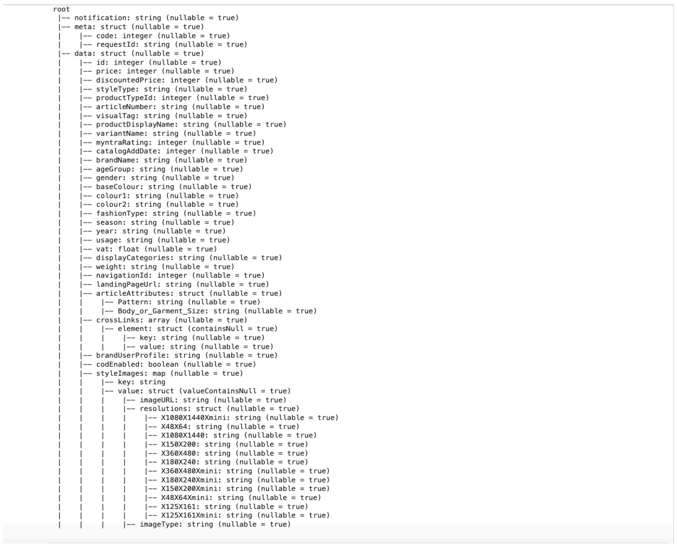
The figure no 4 counts the unique values in categorical columns (`gender`, `masterCategory`, `subCategory`, etc.) to understand their diversity. It uses PySpark's `select`, `distinct`, and `count` functions to calculate the unique values for each column. Key results include 5 unique `gender` values, 7 in `masterCategory`, 45 in `subCategory`, and 143 in `articleType`. This step helps identify category granularity and prepares the data for encoding or grouping in further analysis.



```
+---------+------+----+------------+
| filename|format|mode|        size|
+---------+------+----+------------+
|39638.jpg|  JPEG| RGB|{1080, 1440}|
|23181.jpg|  JPEG| RGB|{1800, 2400}|
|22830.jpg|  JPEG| RGB|{1080, 1440}|
|16786.jpg|  JPEG| RGB|{1080, 1440}|
|41215.jpg|  JPEG| RGB|{1080, 1440}|
+---------+------+----+------------+
only showing top 5 rows
```

Figure no 6. Schema for image files

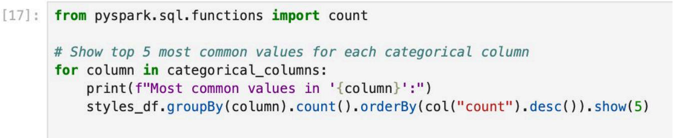
The code will draw metadata from image files from a given folder and then organize that information into a Spark DataFrame for analysis. This code uses the PIL library to loop through all the .jpg files in the directory and collect details regarding filename, file format, image size, and color mode. Its output will be a Spark DataFrame in which each image is mapped onto a single row. For instance, the metadata includes that all images are in the format of JPEG, color mode RGB, with sizes such as {1080, 1440} or {1800, 2400} pixels. The output is quite a good overview of the dataset for further analysis or preprocessing of images with respect to their attributes. This shows the first five rows as a sample and provides assurance that metadata has been extracted and structured.



```
root
 |-- notification: string (nullable = true)
 |-- meta: struct (nullable = true)
 |    |-- code: integer (nullable = true)
 |    |-- requestId: string (nullable = true)
 |-- data: struct (nullable = true)
 |    |-- id: integer (nullable = true)
 |    |-- price: integer (nullable = true)
 |    |-- discountedPrice: integer (nullable = true)
 |    |-- styleType: string (nullable = true)
 |    |-- productTypeId: integer (nullable = true)
 |    |-- articleNumber: string (nullable = true)
 |    |-- visualTag: string (nullable = true)
 |    |-- productDisplayName: string (nullable = true)
 |    |-- variantName: string (nullable = true)
 |    |-- myntraRating: integer (nullable = true)
 |    |-- catalogAddDate: integer (nullable = true)
 |    |-- brandName: string (nullable = true)
 |    |-- ageGroup: string (nullable = true)
 |    |-- gender: string (nullable = true)
 |    |-- baseColour: string (nullable = true)
 |    |-- colour1: string (nullable = true)
 |    |-- colour2: string (nullable = true)
 |    |-- fashionType: string (nullable = true)
 |    |-- season: string (nullable = true)
 |    |-- year: string (nullable = true)
 |    |-- usage: string (nullable = true)
 |    |-- vat: float (nullable = true)
 |    |-- displayCategories: string (nullable = true)
 |    |-- weight: string (nullable = true)
 |    |-- navigationId: integer (nullable = true)
 |    |-- landingPageUrl: string (nullable = true)
 |    |-- articleAttributes: struct (nullable = true)
 |    |    |-- Pattern: string (nullable = true)
 |    |    |-- Body_or_Garment_Size: string (nullable = true)
 |    |-- crossLinks: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- key: string (nullable = true)
 |    |    |    |-- value: string (nullable = true)
 |    |-- brandUserProfile: string (nullable = true)
 |    |-- codEnabled: boolean (nullable = true)
 |    |-- styleImages: map (nullable = true)
 |    |    |-- key: string
 |    |    |-- value: struct (valueContainsNull = true)
 |    |    |    |-- imageURL: string (nullable = true)
 |    |    |    |-- resolutions: struct (nullable = true)
 |    |    |    |    |-- X1080X1440Xmini: string (nullable = true)
 |    |    |    |    |-- X48X64: string (nullable = true)
 |    |    |    |    |-- X1080X1440: string (nullable = true)
 |    |    |    |    |-- X150X200: string (nullable = true)
 |    |    |    |    |-- X360X480: string (nullable = true)
 |    |    |    |    |-- X180X240: string (nullable = true)
 |    |    |    |    |-- X360X480Xmini: string (nullable = true)
 |    |    |    |    |-- X180X240Xmini: string (nullable = true)
 |    |    |    |    |-- X150X200Xmini: string (nullable = true)
 |    |    |    |    |-- X48X64Xmini: string (nullable = true)
 |    |    |    |    |-- X125X161: string (nullable = true)
 |    |    |    |    |-- X125X161Xmini: string (nullable = true)
 |    |    |    |-- imageType: string (nullable = true)
```

Figure no 7. Schema generated from Json file by flattening

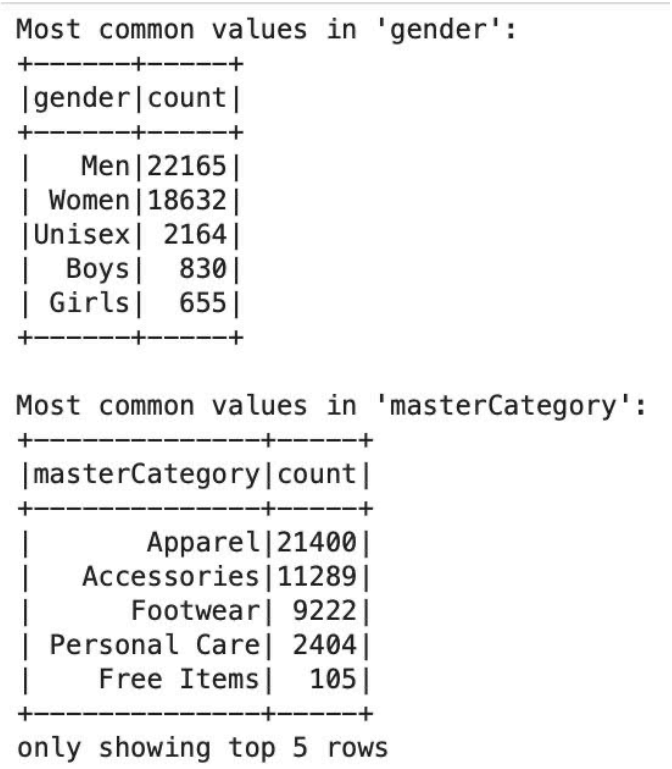
The process of flattening a JSON involves extracting the schema from this hierarchical, nested structure into a simple tabular form for further analysis. This includes accessing and denormalizing the nested objects to dot-separated keys and arrays by indexing or exploding values into rows. In this process, it ultimately generates a schema to represent or describe the structure of that JSON, including field names, data type specification, like string and integer, or even boolean; and nullability. At this stage, Python with pandas or PySpark are one of the popular combinations when it comes to ETL as it makes use of entities like `StructType` at the time of defining schema structure. This approach has served as the key towards going through the complex JSON objects and enabling analysis seamlessly down the pipeline with the utmost ease and interpretability needed for schema validation and data governance.



```
[17]:  from pyspark.sql.functions import count

       # Show top 5 most common values for each categorical column
       for column in categorical_columns:
           print(f"Most common values in '{column}':")
           styles_df.groupBy(column).count().orderBy(col("count").desc()).show(5)
```

Figure no 8. Query for counting common values

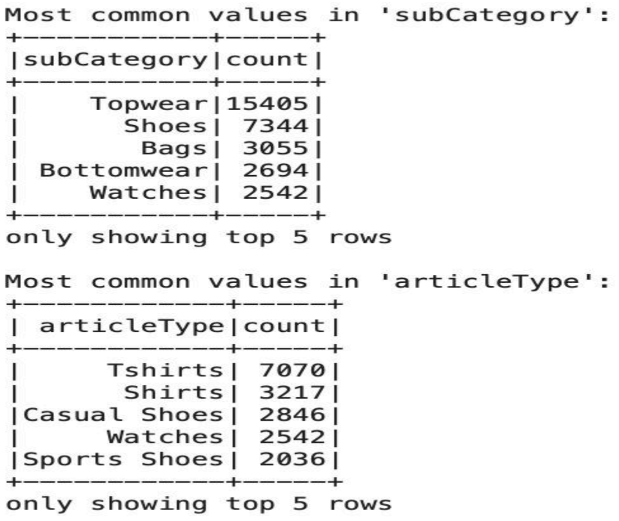
The figure no 8 finds the top 5 most frequent values for each categorical column present in the dataset. This program groups data by each column, counts the frequency of each unique value appearing in the column, and sorts the result in descending manner to find the most frequent entries. Output: The output will give a good understanding of the distribution of data in a particular dataset to analyze popular categories or trends.



```
Most common values in 'gender':
+------+-----+
|gender|count|
+------+-----+
|   Men|22165|
| Women|18632|
|Unisex| 2164|
|  Boys|  830|
| Girls|  655|
+------+-----+

Most common values in 'masterCategory':
+--------------+-----+
|masterCategory|count|
+--------------+-----+
|       Apparel|21400|
|   Accessories|11289|
|      Footwear| 9222|
| Personal Care| 2404|
|    Free Items|  105|
+--------------+-----+
only showing top 5 rows
```

Figure no 9. Common Values count of gender and masterCategory

Figure no 9 explains about:
Gender: The dataset has the highest count for Men, 22,165, followed by Women, 18,632, while categories like Unisex, Boys, and Girls have considerably lower counts. This would suggest a strong focus on adult men's and women's products.
MasterCategory: Apparel tops in 21,400; Accessories, 11,289; and Footwear with 9,222. Thus, many categories, including personal care and free items, do not represent as many entries in this data set.



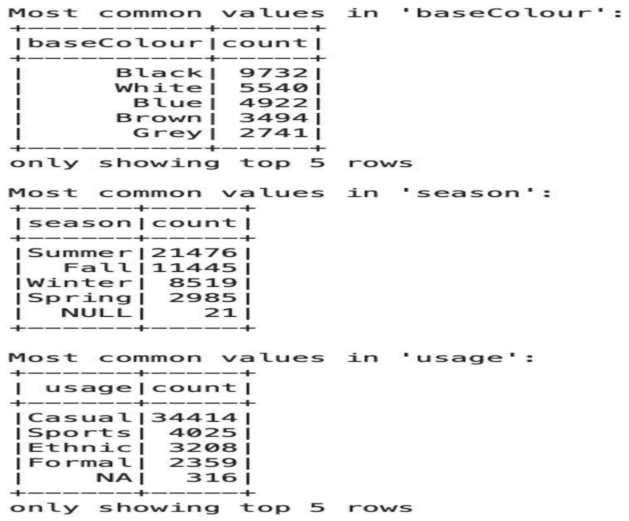
```
Most common values in 'subCategory':
+------------+-----+
|subCategory|count|
+------------+-----+
|     Topwear|15405|
|       Shoes| 7344|
|        Bags| 3055|
|  Bottomwear| 2694|
|     Watches| 2542|
+------------+-----+
only showing top 5 rows

Most common values in 'articleType':
+-------------+-----+
| articleType|count|
+-------------+-----+
|      Tshirts| 7070|
|       Shirts| 3217|
|Casual Shoes| 2846|
|      Watches| 2542|
|Sports Shoes| 2036|
+-------------+-----+
only showing top 5 rows
```

Figure no 10. Common Values count of subcategory and articletype

Figure no 10 explains about:

SubCategory: Topwear is the most common subcategory with 15,405 entries, followed by Shoes (7,344). Other popular categories include Bags, Bottomwear, and Watches, reflecting a focus on clothing and accessories.

ArticleType: Tshirts (7,070) and Shirts (3,217) are the most frequent, showing that the dataset emphasizes casual wear. Categories like Casual Shoes, Watches, and Sports Shoes are also well-represented.



```
Most common values in 'baseColour':
+----------+-----+
|baseColour|count|
+----------+-----+
|     Black| 9732|
|     White| 5540|
|      Blue| 4922|
|     Brown| 3494|
|      Grey| 2741|
+----------+-----+
only showing top 5 rows
Most common values in 'season':
+------+-----+
|season|count|
+------+-----+
|Summer|21476|
|  Fall|11445|
|Winter| 8519|
|Spring| 2985|
|  NULL|   21|
+------+-----+

Most common values in 'usage':
+------+-----+
| usage|count|
+------+-----+
|Casual|34414|
|Sports| 4025|
|Ethnic| 3208|
|Formal| 2359|
|    NA|  316|
+------+-----+
only showing top 5 rows
```

Figure no 11. Common Values count of baseColour, Season and usage

Figure no explaines about:

BaseColour: Black is the most frequent colour, 9,732, followed by White, Blue, and Brown, suggesting a preference for neutral and classic tones.

Season: There are most representations for Summer- 21,476, followed by Fall with 11,445, Winter at 8,519; whereas Spring and null are pretty low, which again confirms this data set should discuss yearly/seasonal trends.

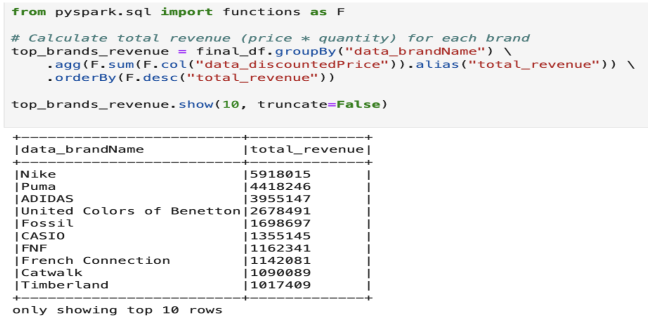
Usage: The Casual usage is highly represented at 34,414 entries, while other categories are Sports, Ethnic, and Formal. This shows that the clothes worn on a daily basis are more in number in this dataset.

To gain a comprehensive understanding of the dataset, our Exploratory Data Analysis (EDA) began with assessing its structure, row count, and completeness by identifying null values. We then extracted key metadata, including image attributes like size, format, and color mode, for further exploration. The categorical columns were checked to show unique values and the most frequent entries to give an essence of men's and women's products with more bias toward those categories, such as 'Apparel' and 'Accessories', preferring 'Neutral' color and mostly 'Casual' in their usage. This EDA thus outlines important features that a dataset will be able to provide and, secondly, indicates key patterns from which a basis can be taken for further cleaning and modeling.

VIII. ANALYTICAL INSIGHTS

In this section, we have implemented queries that result in valuable insights and convey the necessary information about the question or topic detailed below. These queries are designed in a way to analyze the data effectively and deliver actionable and relevant findings to address the subject matter.

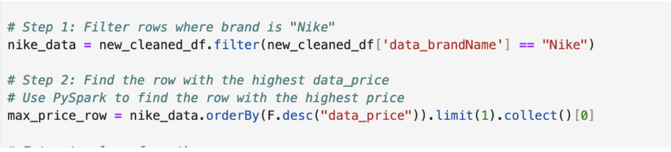
A.] Decoding Sales Performance: A Multifaceted Data Analysis Approach



```
from pyspark.sql import functions as F

# Calculate total revenue (price * quantity) for each brand
top_brands_revenue = final_df.groupBy("data_brandName") \
    .agg(F.sum(F.col("data_discountedPrice")).alias("total_revenue")) \
    .orderBy(F.desc("total_revenue"))

top_brands_revenue.show(10, truncate=False)
```

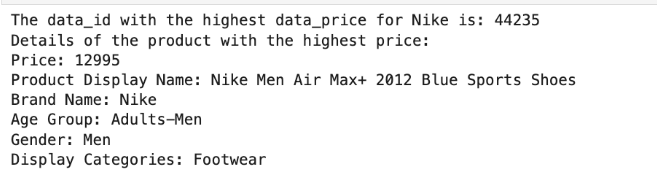
```
+-------------------------+-------------+
|data_brandName           |total_revenue|
+-------------------------+-------------+
|Nike                     |5918015      |
|Puma                     |4418246      |
|ADIDAS                   |3955147      |
|United Colors of Benetton|2678491      |
|Fossil                   |1698697      |
|CASIO                    |1355145      |
|FNF                      |1162341      |
|French Connection        |1142081      |
|Catwalk                  |1090089      |
|Timberland               |1017409      |
+-------------------------+-------------+
only showing top 10 rows
```

Figure no 12. Top 10 Brands by Total Revenue



```
# Step 1: Filter rows where brand is "Nike"
nike_data = new_cleaned_df.filter(new_cleaned_df['data_brandName'] == "Nike")

# Step 2: Find the row with the highest data_price
# Use PySpark to find the row with the highest price
max_price_row = nike_data.orderBy(F.desc("data_price")).limit(1).collect()[0]
```

Figure no 13. Finding the Highest Priced Nike (any brand)Product (a)



```
The data_id with the highest data_price for Nike is: 44235
Details of the product with the highest price:
Price: 12995
Product Display Name: Nike Men Air Max+ 2012 Blue Sports Shoes
Brand Name: Nike
Age Group: Adults-Men
Gender: Men
Display Categories: Footwear
```

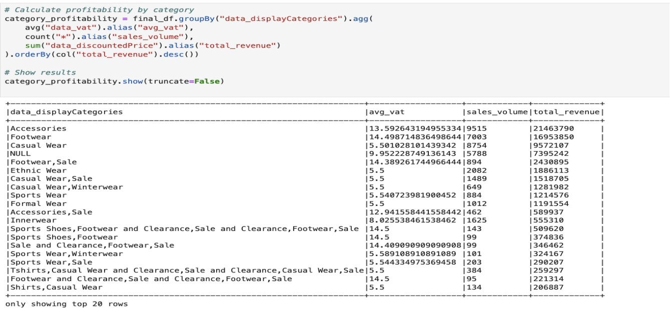
Nike Men Air Max+ 2012 Blue Sports Shoes - Nike
Price: 12995

Figure no 14. Finding the Highest Priced Nike (any brand)Product (b)

Revenue and Pricing Analysis:

Revenue Generation: Brand analysis by total revenue will help identify top performers.

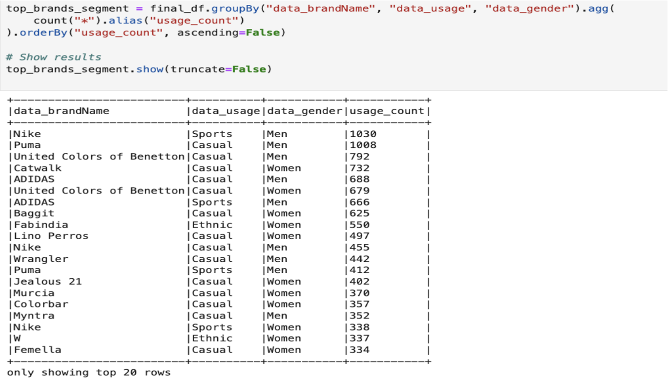
Pricing Strategy: The revenue analysis is complemented by the identification of the most expensive products per brand, which provided insight into the product range and pricing tactics of each brand.



```
# Calculate profitability by category
category_profitability = final_df.groupBy("data_displayCategories").agg(
    avg("data_vat").alias("avg_vat"),
    count("*").alias("sales_volume"),
    sum("data_discountedPrice").alias("total_revenue")
).orderBy(col("total_revenue").desc())

# Show results
category_profitability.show(truncate=False)
```

```
+------------------------------------------------------------------------+------------------+------------+-------------+
|data_displayCategories                                                  |avg_vat           |sales_volume|total_revenue|
+------------------------------------------------------------------------+------------------+------------+-------------+
|Accessories                                                             |13.592643194955334|9515        |21463790     |
|Footwear                                                                |14.498714836498644|7003        |16953850     |
|Casual Wear                                                             |5.501028101439342 |8754        |9572107      |
|NULL                                                                    |9.952228749136143 |5788        |7395242      |
|Footwear,Sale                                                           |14.389261744966444|894         |2430895      |
|Ethnic Wear                                                             |5.5               |2082        |1886113      |
|Casual Wear,Sale                                                        |5.5               |1489        |1518705      |
|Casual Wear,Winterwear                                                  |5.5               |649         |1281982      |
|Sports Wear                                                             |5.540723981900452 |884         |1214576      |
|Formal Wear                                                             |5.5               |1012        |1191554      |
|Accessories,Sale                                                        |12.941558441558442|462         |589937       |
|Innerwear                                                               |8.025538461538462 |1625        |555310       |
|Sports Shoes,Footwear and Clearance,Sale and Clearance,Footwear,Sale    |14.5              |143         |509620       |
|Sports Shoes,Footwear                                                   |14.5              |99          |374836       |
|Sale and Clearance,Footwear,Sale                                        |14.409090909090908|99          |346462       |
|Sports Wear,Winterwear                                                  |5.589108910891089 |101         |324167       |
|Sports Wear,Sale                                                        |5.544334975369458 |203         |290207       |
|Tshirts,Casual Wear and Clearance,Sale and Clearance,Casual Wear,Sale   |5.5               |384         |259297       |
|Footwear and Clearance,Sale and Clearance,Footwear,Sale                 |14.5              |95          |221314       |
|Shirts,Casual Wear                                                      |5.5               |134         |206887       |
+------------------------------------------------------------------------+------------------+------------+-------------+
only showing top 20 rows
```

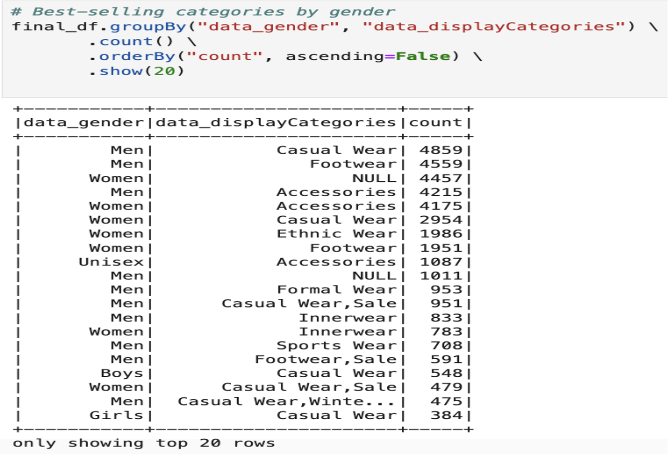
Figure no 15. Profitability Metrics by Category



```
top_brands_segment = final_df.groupBy("data_brandName", "data_usage", "data_gender").agg(
    count("*").alias("usage_count")
).orderBy("usage_count", ascending=False)

# Show results
top_brands_segment.show(truncate=False)
```

```
+-------------------------+----------+-----------+-----------+
|data_brandName           |data_usage|data_gender|usage_count|
+-------------------------+----------+-----------+-----------+
|Nike                     |Sports    |Men        |1030       |
|Puma                     |Casual    |Men        |1008       |
|United Colors of Benetton|Casual    |Men        |792        |
|Catwalk                  |Casual    |Women      |732        |
|ADIDAS                   |Casual    |Men        |688        |
|United Colors of Benetton|Casual    |Women      |679        |
|ADIDAS                   |Sports    |Men        |666        |
|Baggit                   |Casual    |Women      |625        |
|Fabindia                 |Ethnic    |Women      |550        |
|Lino Perros              |Casual    |Women      |497        |
|Nike                     |Casual    |Men        |455        |
|Wrangler                 |Casual    |Men        |442        |
|Puma                     |Sports    |Men        |412        |
|Jealous 21               |Casual    |Women      |402        |
|Murcia                   |Casual    |Women      |370        |
|Colorbar                 |Casual    |Women      |357        |
|Myntra                   |Casual    |Men        |352        |
|Nike                     |Sports    |Women      |338        |
|W                        |Ethnic    |Women      |337        |
|Femella                  |Casual    |Women      |334        |
+-------------------------+----------+-----------+-----------+
only showing top 20 rows
```

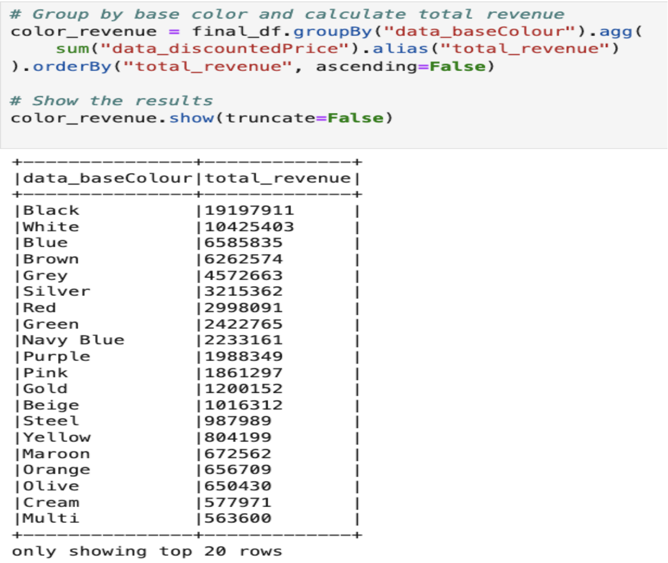
Figure no 16. Top Brands by Usage Segment and Gender



```
# Best-selling categories by gender
final_df.groupBy("data_gender", "data_displayCategories") \
        .count() \
        .orderBy("count", ascending=False) \
        .show(20)
```

```
+-----------+----------------------+-----+
|data_gender|data_displayCategories|count|
+-----------+----------------------+-----+
|        Men|            Casual Wear| 4859|
|        Men|               Footwear| 4559|
|      Women|                   NULL| 4457|
|        Men|            Accessories| 4215|
|      Women|            Accessories| 4175|
|      Women|            Casual Wear| 2954|
|      Women|            Ethnic Wear| 1986|
|      Women|               Footwear| 1951|
|     Unisex|            Accessories| 1087|
|        Men|                   NULL| 1011|
|        Men|            Formal Wear|  953|
|        Men|       Casual Wear,Sale|  951|
|        Men|              Innerwear|  833|
|      Women|              Innerwear|  783|
|        Men|            Sports Wear|  708|
|        Men|          Footwear,Sale|  591|
|       Boys|            Casual Wear|  548|
|      Women|       Casual Wear,Sale|  479|
|        Men|   Casual Wear,Winte...|  475|
|      Girls|            Casual Wear|  384|
+-----------+----------------------+-----+
only showing top 20 rows
```

Figure no 17. Best-Selling Categories by Gender



```
# Group by base color and calculate total revenue
color_revenue = final_df.groupBy("data_baseColour").agg(
    sum("data_discountedPrice").alias("total_revenue")
).orderBy("total_revenue", ascending=False)

# Show the results
color_revenue.show(truncate=False)
```

```
+---------------+-------------+
|data_baseColour|total_revenue|
+---------------+-------------+
|Black          |19197911     |
|White          |10425403     |
|Blue           |6585835      |
|Brown          |6262574      |
|Grey           |4572663      |
|Silver         |3215362      |
|Red            |2998091      |
|Green          |2422765      |
|Navy Blue      |2233161      |
|Purple         |1988349      |
|Pink           |1861297      |
|Gold           |1200152      |
|Beige          |1016312      |
|Steel          |987989       |
|Yellow         |804199       |
|Maroon         |672562       |
|Orange         |656709       |
|Olive          |650430       |
|Cream          |577971       |
|Multi          |563600       |
+---------------+-------------+
only showing top 20 rows
```

Figure no 18. Total Revenue by Base Color [popularity]

From Figure no 12 to 18 we infer that

Brand Performance: The top-selling categories by gender can give an inclination of which brands can perform well with certain demographics.

Pricing Insights: Understanding gender differences in categories helps companies avoid some of the pricing mistakes.

Revenue Optimization: Knowing the popular categories by gender can help in focusing marketing efforts for maximum revenue. also we get to know about the popular colour by observing the total revenue incurred by that product based on colour.

B.] Seasonal Trends in Discounted Pricing

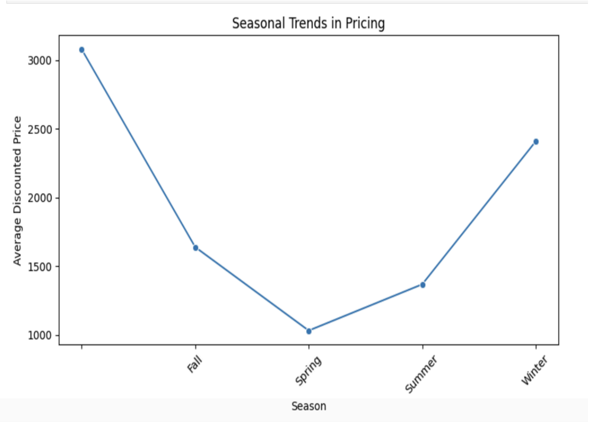
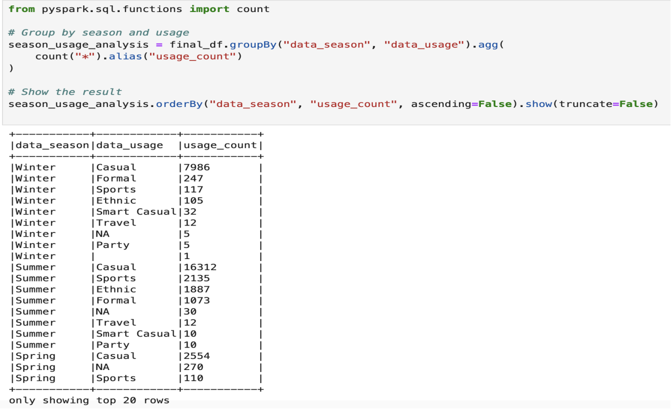


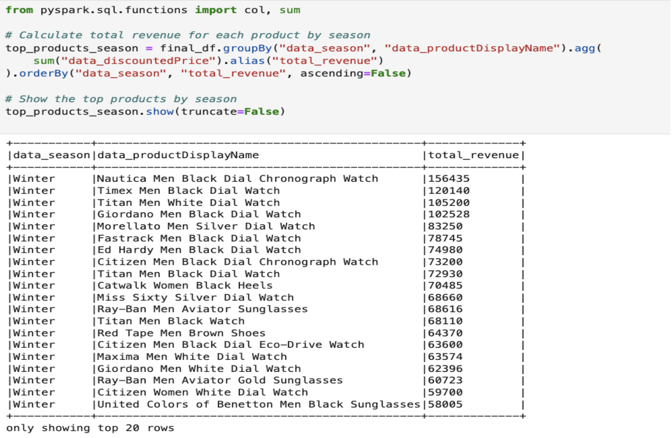
Figure no 19.Seasonal Trends in Average Discounted Pricing



```
from pyspark.sql.functions import count

# Group by season and usage
season_usage_analysis = final_df.groupBy("data_season", "data_usage").agg(
    count("*").alias("usage_count")
)

# Show the result
season_usage_analysis.orderBy("data_season", "usage_count", ascending=False).show(truncate=False)
```

```
+-----------+------------+-----------+
|data_season|data_usage  |usage_count|
+-----------+------------+-----------+
|Winter     |Casual      |7986       |
|Winter     |Formal      |247        |
|Winter     |Sports      |117        |
|Winter     |Ethnic      |105        |
|Winter     |Smart Casual|32         |
|Winter     |Travel      |12         |
|Winter     |NA          |5          |
|Winter     |Party       |5          |
|Winter     |            |1          |
|Summer     |Casual      |16312      |
|Summer     |Sports      |2135       |
|Summer     |Ethnic      |1887       |
|Summer     |Formal      |1073       |
|Summer     |NA          |30         |
|Summer     |Travel      |12         |
|Summer     |Smart Casual|10         |
|Summer     |Party       |10         |
|Spring     |Casual      |2554       |
|Spring     |NA          |270        |
|Spring     |Sports      |110        |
+-----------+------------+-----------+
only showing top 20 rows
```

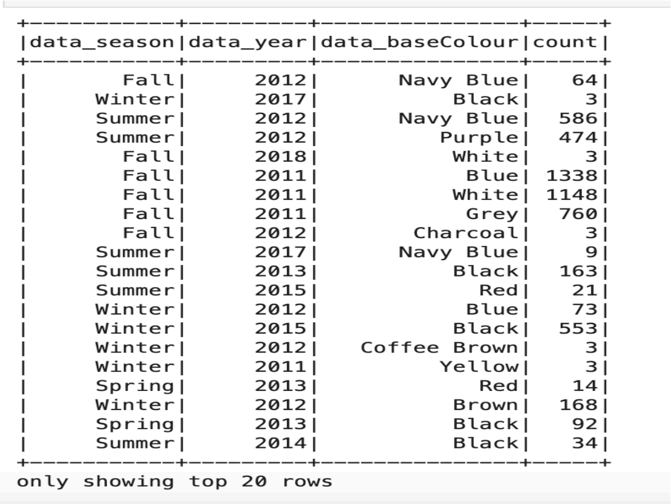
Figure no 20.Usage Analysis by Season and Category



```
from pyspark.sql.functions import col, sum

# Calculate total revenue for each product by season
top_products_season = final_df.groupBy("data_season", "data_productDisplayName").agg(
    sum("data_discountedPrice").alias("total_revenue")
).orderBy("data_season", "total_revenue", ascending=False)

# Show the top products by season
top_products_season.show(truncate=False)
```

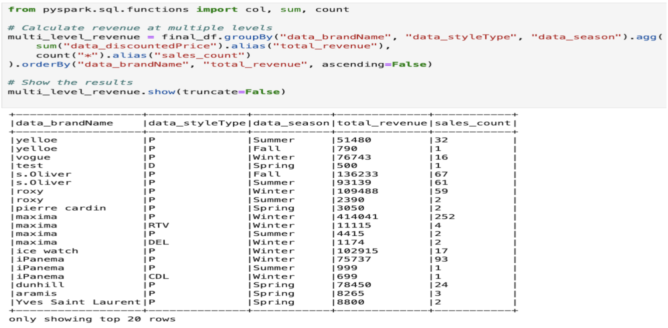
```
+-----------+-----------------------------------------------+-------------+
|data_season|data_productDisplayName                        |total_revenue|
+-----------+-----------------------------------------------+-------------+
|Winter     |Nautica Men Black Dial Chronograph Watch       |156435       |
|Winter     |Timex Men Black Dial Watch                     |120140       |
|Winter     |Titan Men White Dial Watch                     |105200       |
|Winter     |Giordano Men Black Dial Watch                  |102528       |
|Winter     |Morellato Men Silver Dial Watch                |83250        |
|Winter     |Fastrack Men Black Dial Watch                  |78745        |
|Winter     |Ed Hardy Men Black Dial Watch                  |74980        |
|Winter     |Citizen Men Black Dial Chronograph Watch       |73200        |
|Winter     |Titan Men Black Dial Watch                     |72930        |
|Winter     |Catwalk Women Black Heels                      |70485        |
|Winter     |Miss Sixty Silver Dial Watch                   |68660        |
|Winter     |Ray-Ban Men Aviator Sunglasses                 |68616        |
|Winter     |Titan Men Black Watch                          |68110        |
|Winter     |Red Tape Men Brown Shoes                       |64370        |
|Winter     |Citizen Men Black Dial Eco-Drive Watch         |63600        |
|Winter     |Maxima Men White Dial Watch                    |63574        |
|Winter     |Giordano Men White Dial Watch                  |62396        |
|Winter     |Ray-Ban Men Aviator Gold Sunglasses            |60723        |
|Winter     |Citizen Women White Dial Watch                 |59700        |
|Winter     |United Colors of Benetton Men Black Sunglasses|58005        |
+-----------+-----------------------------------------------+-------------+
only showing top 20 rows
```

Figure no 21.Top Products by Revenue and Season



```
+-----------+---------+--------------+-----+
|data_season|data_year|data_baseColour|count|
+-----------+---------+--------------+-----+
|       Fall|     2012|     Navy Blue|   64|
|     Winter|     2017|         Black|    3|
|     Summer|     2012|     Navy Blue|  586|
|     Summer|     2012|        Purple|  474|
|       Fall|     2018|         White|    3|
|       Fall|     2011|          Blue| 1338|
|       Fall|     2011|         White| 1148|
|       Fall|     2011|          Grey|  760|
|       Fall|     2012|      Charcoal|    3|
|     Summer|     2017|     Navy Blue|    9|
|     Summer|     2013|         Black|  163|
|     Summer|     2015|           Red|   21|
|     Winter|     2012|          Blue|   73|
|     Winter|     2015|         Black|  553|
|     Winter|     2012|  Coffee Brown|    3|
|     Winter|     2011|        Yellow|    3|
|     Spring|     2013|           Red|   14|
|     Winter|     2012|         Brown|  168|
|     Spring|     2013|         Black|   92|
|     Summer|     2014|         Black|   34|
+-----------+---------+--------------+-----+
only showing top 20 rows
```

Figure no 22.Seasonal and Yearly Distribution of Base Colors in Products



```python
from pyspark.sql.functions import col, sum, count

# Calculate revenue at multiple levels
multi_level_revenue = final_df.groupBy("data_brandName", "data_styleType", "data_season").agg(
    sum("data_discountedPrice").alias("total_revenue"),
    count("*").alias("sales_count")
).orderBy("data_brandName", "total_revenue", ascending=False)

# Show the results
multi_level_revenue.show(truncate=False)
```

```
+-------------------+-------------+-----------+-------------+-----------+
|data_brandName     |data_styleType|data_season|total_revenue|sales_count|
+-------------------+-------------+-----------+-------------+-----------+
|yelloe             |P            |Summer     |51480        |32         |
|yelloe             |P            |Fall       |790          |1          |
|vogue              |P            |Winter     |76743        |16         |
|test               |D            |Spring     |500          |1          |
|s.Oliver           |P            |Fall       |136233       |67         |
|s.Oliver           |P            |Summer     |93139        |61         |
|roxy               |P            |Winter     |109488       |59         |
|roxy               |P            |Summer     |2390         |2          |
|pierre cardin      |P            |Spring     |3050         |2          |
|maxima             |P            |Winter     |414041       |252        |
|maxima             |RTV          |Winter     |11115        |4          |
|maxima             |P            |Summer     |4415         |2          |
|maxima             |DEL          |Winter     |1174         |2          |
|ice watch          |P            |Winter     |102915       |17         |
|iPanema            |P            |Winter     |75737        |93         |
|iPanema            |P            |Summer     |999          |1          |
|iPanema            |CDL          |Winter     |699          |1          |
|dunhill            |P            |Spring     |78450        |24         |
|aramis             |P            |Spring     |8265         |3          |
|Yves Saint Laurent|P            |Spring     |8800         |2          |
+-------------------+-------------+-----------+-------------+-----------+
only showing top 20 rows
```

Figure no 23. Multi-Level Revenue and Sales Count by Brand, Style, and Season

From figure no 19 to 23 we infer that

Revenue Impact:

Seasonal variation in prices can have significant influences on overall revenue performance:

-High-revenue brands, selected in sections, will illustrate different seasonal pricing patterns.

-Understanding such trends can help optimize pricing to capture the best revenues across seasons.

Brand-Specific Trends

Seasonal pricing analysis complements information about brand performance:

-Different brands may have different discounting strategies for different seasons.

- The examination shall show which brands are more sensitive to seasonal price variability.

Product Category Insights

Seasonal pricing trends will most probably differ from one product category to another.

- Best-selling categories by gender may have different seasonal pricing behaviors.

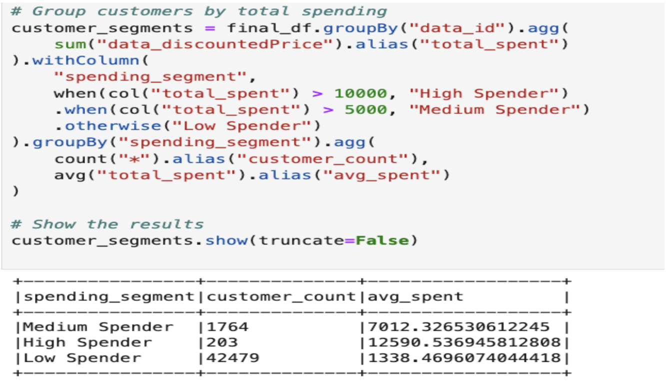
- It provides information for category-specific pricing strategies throughout the year.

Data-Driven Strategy : The exploration of seasonal pricing feeds into an overall approach: data-driven.

It offers yet another dimension to informed pricing and inventory control decisions.

Combined with other insights, it allows for a far better understanding of the dynamics of the market. Further analysis of seasonal variation in price will help complete a pattern that shows pricing strategies and the development of the consumer's response behavior toward seasonal products, and ways that provide revenue optimization within periods throughout a year.

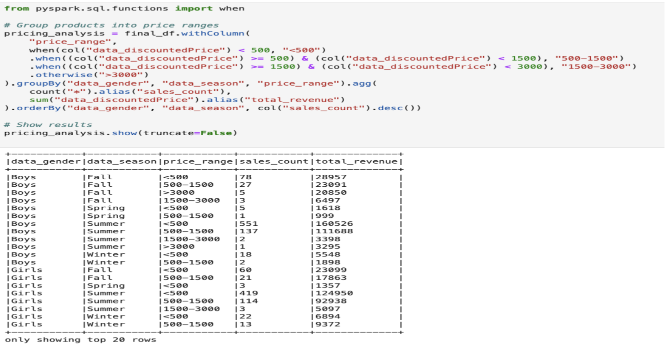
C.] Retail Insights: Discount, Customer, and Pricing Analysis



```python
# Group customers by total spending
customer_segments = final_df.groupBy("data_id").agg(
    sum("data_discountedPrice").alias("total_spent")
).withColumn(
    "spending_segment",
    when(col("total_spent") > 10000, "High Spender")
    .when(col("total_spent") > 5000, "Medium Spender")
    .otherwise("Low Spender")
).groupBy("spending_segment").agg(
    count("*").alias("customer_count"),
    avg("total_spent").alias("avg_spent")
)

# Show the results
customer_segments.show(truncate=False)
```

```
+----------------+--------------+------------------+
|spending_segment|customer_count|avg_spent         |
+----------------+--------------+------------------+
|Medium Spender  |1764          |7012.326530612245 |
|High Spender    |203           |12590.536945812808|
|Low Spender     |42479         |1338.4696074044418|
+----------------+--------------+------------------+
```

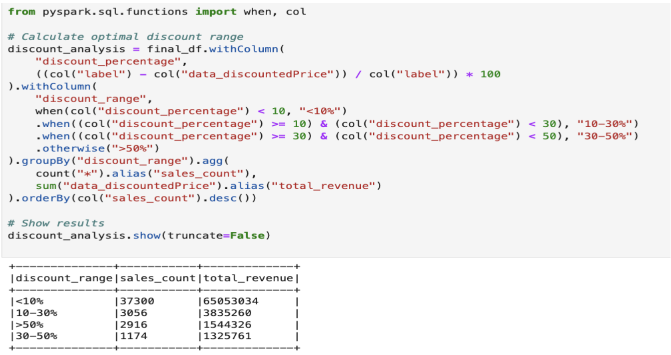
Figure no 24. Customer Segmentation by Spending Levels



```python
from pyspark.sql.functions import when

# Group products into price ranges
pricing_analysis = final_df.withColumn(
    "price_range",
    when(col("data_discountedPrice") < 500, "<500")
    .when((col("data_discountedPrice") >= 500) & (col("data_discountedPrice") < 1500), "500-1500")
    .when((col("data_discountedPrice") >= 1500) & (col("data_discountedPrice") < 3000), "1500-3000")
    .otherwise(">3000")
).groupBy("data_gender", "data_season", "price_range").agg(
    count("*").alias("sales_count"),
    sum("data_discountedPrice").alias("total_revenue")
).orderBy("data_gender", "data_season", col("sales_count").desc())

# Show results
pricing_analysis.show(truncate=False)
```

```
+-----------+-----------+-----------+-----------+-------------+
|data_gender|data_season|price_range|sales_count|total_revenue|
+-----------+-----------+-----------+-----------+-------------+
|Boys       |Fall       |<500       |78         |28957        |
|Boys       |Fall       |500-1500   |27         |23091        |
|Boys       |Fall       |>3000      |5          |20850        |
|Boys       |Fall       |1500-3000  |3          |6497         |
|Boys       |Spring     |<500       |5          |1618         |
|Boys       |Spring     |500-1500   |1          |999          |
|Boys       |Summer     |<500       |551        |160526       |
|Boys       |Summer     |500-1500   |137        |111688       |
|Boys       |Summer     |1500-3000  |2          |3398         |
|Boys       |Summer     |>3000      |1          |3295         |
|Boys       |Winter     |<500       |18         |5548         |
|Boys       |Winter     |500-1500   |2          |1898         |
|Girls      |Fall       |<500       |60         |23099        |
|Girls      |Fall       |500-1500   |21         |17863        |
|Girls      |Spring     |<500       |3          |1357         |
|Girls      |Summer     |<500       |419        |124950       |
|Girls      |Summer     |500-1500   |114        |92938        |
|Girls      |Summer     |1500-3000  |3          |5097         |
|Girls      |Winter     |<500       |22         |6894         |
|Girls      |Winter     |500-1500   |13         |9372         |
+-----------+-----------+-----------+-----------+-------------+
only showing top 20 rows
```

Figure no 25. Pricing Analysis by Gender, Season, and Price Range



```python
from pyspark.sql.functions import when, col

# Calculate optimal discount range
discount_analysis = final_df.withColumn(
    "discount_percentage",
    ((col("label") - col("data_discountedPrice")) / col("label")) * 100
).withColumn(
    "discount_range",
    when(col("discount_percentage") < 10, "<10%")
    .when((col("discount_percentage") >= 10) & (col("discount_percentage") < 30), "10-30%")
    .when((col("discount_percentage") >= 30) & (col("discount_percentage") < 50), "30-50%")
    .otherwise(">50%")
).groupBy("discount_range").agg(
    count("*").alias("sales_count"),
    sum("data_discountedPrice").alias("total_revenue")
).orderBy(col("sales_count").desc())

# Show results
discount_analysis.show(truncate=False)
```

```
+--------------+-----------+-------------+
|discount_range|sales_count|total_revenue|
+--------------+-----------+-------------+
|<10%          |37300      |65053034     |
|10-30%        |3056       |3835260      |
|>50%          |2916       |1544326      |
|30-50%        |1174       |1325761      |
+--------------+-----------+-------------+
```

Figure no 26. Discount Analysis by Range, Sales Count, and Total Revenue

From figure no 24,25 and 26 we infer

1. Discount Analysis

Sales and revenue have been analyzed based on four ranges of discount.

Less than 10%: The maximum number of sales-37,300 and revenue ₹ 65 crore accrued here, which shows that low discounts are very effective.
10–30%: Moderate discounts generated ₹3.8 crore revenue with 3,056 sales.
The least effective one is >50%, reaping only ₹1.3 crore revenue and 2,311 sales.
Key Insight: Larger discounts maximize sales and revenue.

2. Customer Segmentation

Customers with higher total spending were separated into groups:
High Spenders: Fewest customers, highest average spent-203, ₹ 12,590.
Medium Spenders: Medium size, 1,764; average spend, ₹7,012.
Low Spenders: Big group 42,479, but low average spend ₹1,338.
Key Takeaway: The strategy of moving medium spenders to high spenders will drive profitability higher.

3. Price Analysis

The products were categorized on the basis of price, gender, and season:
< ₹500: Dominates sales across seasons.
₹500–1,500: Pertinent to Girls during Summer (₹1.25 lakh revenue, 419 sales).
Key Insight: Price below ₹500 for volume; use ₹500+ ranges strategically for high-revenue niches.

IX. MODEL IMPLEMENTATION

In this project, the Linear Regression model is implemented using MLlib of Apache Spark to estimate a quantitative response variable. Linear Regression was chosen because of its simplicity, interpretability, and efficiency that are fitting when Linear Regression presumes a linear dependency or relationship between the input features and the target variable. The clearest and most straightforward interpretations that this model can give regarding how various input features relate to the target variable-when their relationships among the latter set are assumed linear-make it an ideal model for execution with this dataset.

We used Spark MLlib, which provided scalable and powerful algorithms to train and test the model on the same dataset; now, the aim is to find how effective the Linear Regression model will be in making good predictions with this model, so that further model selection and data analytics strategy will be done with the help of results obtained from this research.

A. Linear Regression

In the data preparation phase for the model, the focus is on indexing categorical features and assembling both indexed and numerical features into a comprehensive feature vector. This process is crucial for models that depend on a unified representation of data inputs to effectively learn and make predictions.

Descriptive Features:

Categorical Features:

data_fashionType, **data_brandName**, and **data_baseColour** are examples of categorical data within the dataset. These features are typically text-based and represent various attributes of the fashion products, such as the type of fashion item, brand name, and base color. To make these features usable in machine learning models, they are transformed into numerical indices. This transformation is accomplished using indexing techniques which map each unique categorical value to a unique integer.

Numerical Features:

data_price and **data_year** are examples of numerical features. These features are used directly in the model without the need for transformation. data_price represents the price of the fashion items, which can be an important variable for predictions related to sales or customer preferences. data_year might indicate the year of the product's release or the year data was collected, providing temporal context that could affect modeling outcomes.

Target Features:

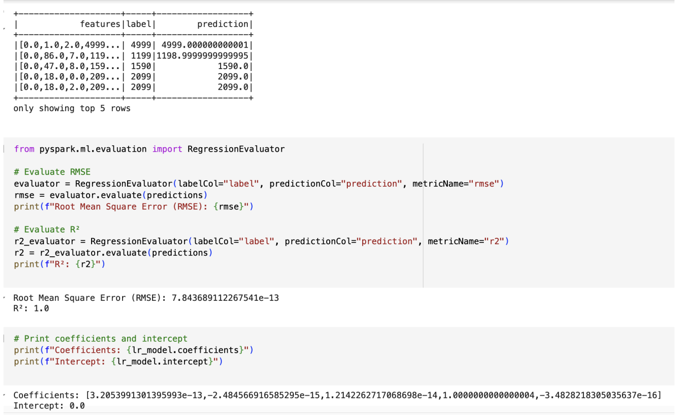
data_price: This is used as the target variable for the model. In machine learning terms, the target variable is what the model is attempting to predict. In your case, "data_price" seems to be serving dual roles—both as part of the input features and as the target variable. This can be typical in scenarios where the model might need to understand the influence of the price itself in conjunction with other features to predict future prices or related financial metrics.

In the above Linear Regression model, using the LinearRegression class from pyspark.ml.regression, featuresCol is configured to "features" for the input features, and labelCol is set to "label" for the target variable. This means the configuration of this model should predict the values in the column named "label" of our DataFrame final_df. First, it fits the model using the fit method on this DataFrame, which includes learning in the relationship between the input features and the target variable. Predictions will be made on this very same dataset, and results showing the predicted outcome include actual values of the target variable for comparison in output. This model's evaluation involves an understanding of how good the predicted values correspond to the actual values, which is facilitated by the provision of results directly.

B. Evaluation Results

Model Performance: The Linear Regression model demonstrates exceptional performance with an R^2 value of 1.0 and a remarkably low RMSE, signifying perfect predictions where the predicted values precisely match the actual values. The plot of Actual vs. Predicted Values illustrates this accuracy, showing a direct correspondence between predicted and actual labels.

Interpretation of Coefficients: The nearly zero coefficients for some features indicate minimal influence on the model's predictions, whereas a coefficient close to one suggests a direct linear influence of that feature. The zero intercept further supports the model's accuracy, indicating no bias when all features are zero.



```
+--------------------+-----+-----------------+
|            features|label|       prediction|
+--------------------+-----+-----------------+
|[0.0,1.0,2.0,4999...| 4999| 4999.000000000001|
|[0.0,86.0,7.0,119...| 1199|1198.9999999999995|
|[0.0,47.0,8.0,159...| 1590|           1590.0|
|[0.0,18.0,0.0,209...| 2099|           2099.0|
|[0.0,18.0,2.0,209...| 2099|           2099.0|
+--------------------+-----+-----------------+
only showing top 5 rows
```

```
from pyspark.ml.evaluation import RegressionEvaluator

# Evaluate RMSE
evaluator = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print(f"Root Mean Square Error (RMSE): {rmse}")

# Evaluate R²
r2_evaluator = RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="r2")
r2 = r2_evaluator.evaluate(predictions)
print(f"R²: {r2}")
```

```
Root Mean Square Error (RMSE): 7.843689112267541e-13
R²: 1.0
```

```
# Print coefficients and intercept
print(f"Coefficients: {lr_model.coefficients}")
print(f"Intercept: {lr_model.intercept}")
```

```
Coefficients: [3.2053991301395993e-13,-2.484566916585295e-15,1.2142262717068698e-14,1.0000000000000004,-3.4828218305035637e-16]
Intercept: 0.0
```

Figure no 27. Results For Linear regression

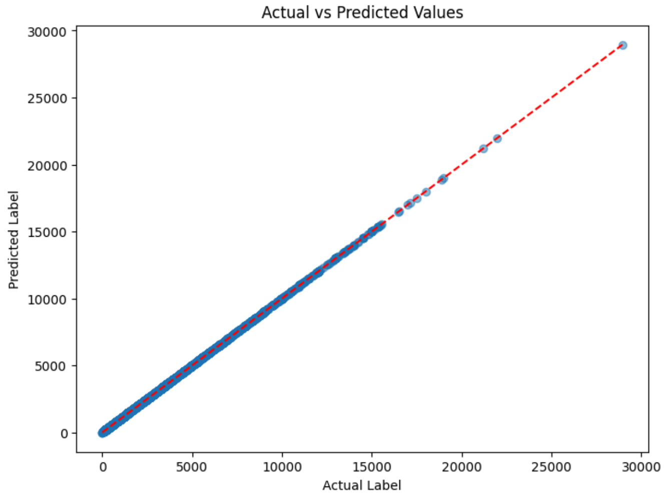


Figure no 28. Actual VS Predicted values

X. CONCLUSION

In this project, we used big data technologies and machine learning algorithms to gain insight into fashion trends and customer preferences from the Kaggle Fashion Product Images dataset. We applied data parallelism principles using Apache Spark and Hadoop for distributed processing of over 44,000 high-resolution images along with their metadata. Spark MLlib allowed scalability in the machine learning applications, such as product categorization and trend prediction.

Our exploratory data analysis showed trends in pricing, brand popularity, and demographic-specific trends that provided actionable insights into the field of fashion retailers. The application of a Linear Regression model has shown high predictive accuracy; thus, it shows the potentiality of using data-driven strategies with regard to optimizing inventory management and marketing campaigns.

This research underlines big data and machine learning for their transforming roles within the fashion retail industry. A correct marriage of scalable technologies with strong analytics will, however, enable businesses to make better decisions, experience more customers, and keep the pace in a fast-moving market. Further work could extend this analysis into real-time data streams and advanced predictive models to further refine the trend forecasting and customer segmentation strategies.

XI. REFERENCES

[1] Y. Li, H. Wang and Y. Li, "Research on query analysis and optimization based on spark," 2017 6th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2017, pp. 251-255,

[2] doi:10.1109/ICCSNT.2017.8343697.keywords: {Sparks;Optimization;BigData;Syntactics;Data bases;Computerarchitecture;Histograms;Big data;SparkSQL;Catalyst;cost optimization;hash join},

[3] Bandi, Raswitha & Jayavel, Amudhavel & Karthik, R.. (2018). Machine Learning with PySpark - Review. Indonesian Journal of Electrical Engineering and Computer Science. 12.102-106. 10.11591/ijeecs.v12.i1.pp102-106.

[4] Agrawal, Saroj & Gupta, Yogesh. (2023). Optimization and Performance analysis on PySpark techniques for lungs X-Ray Images. 10.21203/rs.3.rs-3299056/v1.

[5] H. T. Almansouri and Y. Masmoudi, "Hadoop Distributed File System for Big data analysis," 2019 4th World Conference on Complex Systems (WCCS), Ouarzazate, Morocco, 2019, pp. 1-5, doi: 10.1109/ICoCS.2019.8930804. keywords: {Training;Data collection;Software;Software algorithms;Task analysis;Industrial engineering;Virtual reality;Hadoop;MapReduce;HDFS;DataNode; NameNode;Big Data Analysis},