Analyzing Fashion Trends and Customer Preferences Using Big Data Technologies

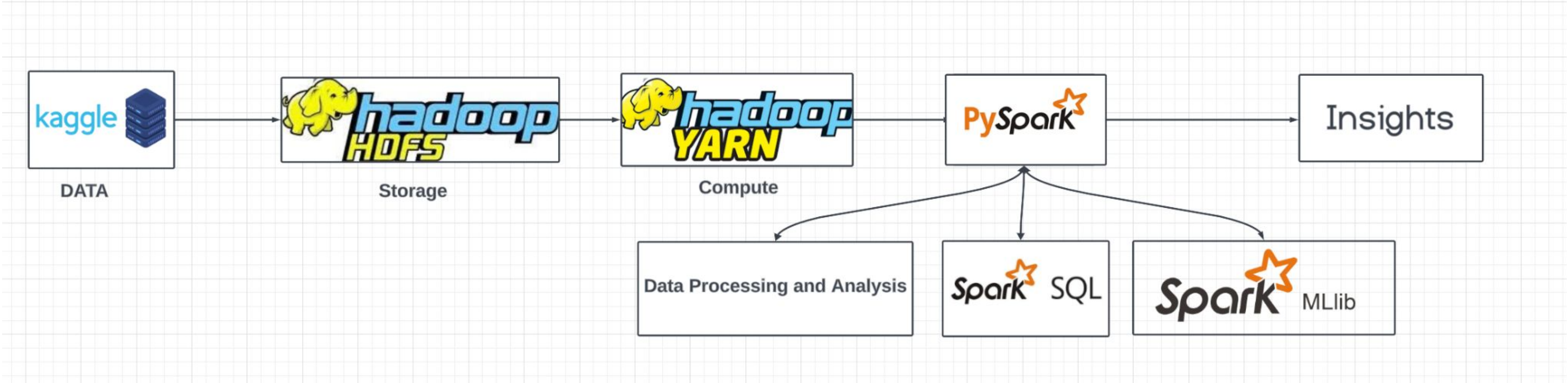
Group 6

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

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.

Proposed Method



Dataset Overview

Structure

- **Directories**
  - **images/: Contains 44,400 high-resolution product images (JPG format), each mapped to a unique product ID.**
  - **styles/: Includes metadata files (JSON format) with detailed product descriptions and attributes.**
- **Files**
  - **styles.csv: Maps product IDs to 12 key attributes (e.g., categories, display names, labels).**
  - **images.csv: Provides supplementary information about the image data.**

Content

- **Each product is linked through a unique product ID, connecting images, metadata, and CSV records.**
- **styles.csv serves as the primary mapping file, detailing product attributes.**
- **JSON metadata complements visual data with rich product context.**

Summary

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

Dataset Pre-Processing

Efficient handling of JSON

Defined a schema to process nested structures, ensuring coherence in data types.

Flattened complex fields like meta and data to query them more easily- for example, meta.code, data.price.

Exploded the arrays in individual rows to further develop the relational analysis that could be done with data_crossLinks.

Data Cleaning and Validation

Addressed null/NaN values in critical columns, such as label, via checks, filters, and replacements.

Dropped irrelevant or redundant columns: notification, data_styleImages etc.

Ensured completeness and consistency in the final data for analysis.

Feature Selection and Transformations

Focused on main attributes such as data_id, data_price, data_brandName, and meta_code. Extracted new features, for example discount calculation: data.price - data.discountedPrice. Prepared data for EDA, feature engineering, and machine learning workflows.

Data Analysis[Insights]

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

Top 10 Brands by Total Revenue

```
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
```

Data Analysis[Insights]

Finding the Highest Priced Nike (any brand)Product

```python
from pyspark.sql import functions as F
import matplotlib.pyplot as plt
from PIL import Image
import os

brand = input(str("Enter your Brand: "))
# Step 1: Filter rows where brand is "Nike"
nike_data = cleaned_df.filter(cleaned_df['data_brandName'] == brand)

# 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]

# Extract values from the row
highest_price_id = max_price_row['data_id']
data_price = max_price_row['data_price']
data_productDisplayName = max_price_row['data_productDisplayName']
data_brandName = max_price_row['data_brandName']
data_ageGroup = max_price_row['data_ageGroup']
data_gender = max_price_row['data_gender']
data_displayCategories = max_price_row['data_displayCategories']

# Print the respective values
print(f"The data_id with the highest data_price for {brand} is: INR {highest_price_id}")
print(f"Details of the product with the highest price:")
print(f"Price: INR {data_price}")
print(f"Product Display Name: {data_productDisplayName}")
print(f"Brand Name: {data_brandName}")
print(f"Age Group: {data_ageGroup}")
print(f"Gender: {data_gender}")
print(f"Display Categories: {data_displayCategories}")
```

```
Enter your Brand:  Nike
[Stage 188:==================================================> (1350 + 8) / 1389]
The data_id with the highest data_price for Nike is: INR 44235
Details of the product with the highest price:
Price: INR 12995
Product Display Name: Nike Men Air Max+ 2012 Blue Sports Shoes
Brand Name: Nike
Age Group: Adults-Men
Gender: Men
Display Categories: Footwear
```

Data Analysis[Insights]

Profitability Metrics by Category

```python
# 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

Data Analysis[Insights]

Top Brands by Usage Segment and Gender

```
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
```

Data Analysis[Insights]

Best-Selling Categories by 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
```

Data Analysis[Insights]

Total Revenue by Base Color [popularity]

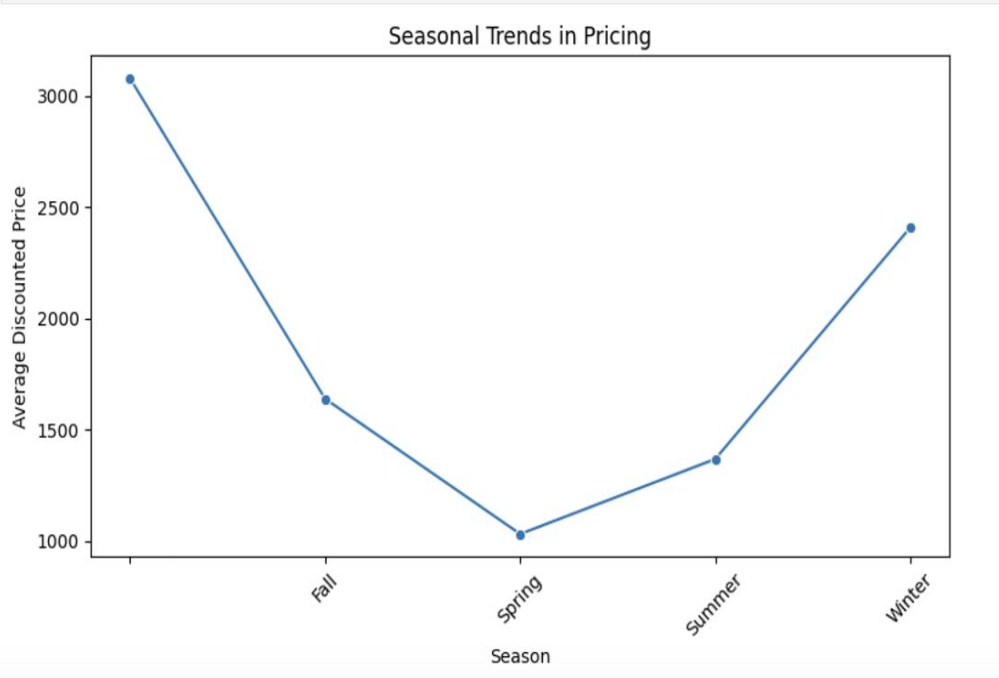
```
# Group by base color and calculate total revenue
color_revenue = final_df.groupBy("data_baseColour").agg(
    sum("data_discountedPrice").alias("total_revenue")
).orderBy("total_revenue", ascending=False)

# 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
```

Data Analysis[Insights]

B.] Seasonal Trends in Discounted Pricing

Seasonal Trends in Average Discounted Pricing




```python
from pyspark.sql import functions as F
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate the average discounted price by season
seasonal_price = final_df.groupBy("data_season") \
    .agg(F.avg("data_discountedPrice").alias("avg_discounted_price")) \
    .orderBy("data_season")

# Convert to Pandas for visualization
seasonal_price_pd = seasonal_price.toPandas()

# Visualization: Line graph
plt.figure(figsize=(8, 5))
sns.lineplot(data=seasonal_price_pd, x="data_season", y="avg_discounted_price", marker="o")
plt.title("Seasonal Trends in Pricing")
plt.ylabel("Average Discounted Price")
plt.xlabel("Season")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Data Analysis[Insights]

Usage Analysis by Season and Category

```
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
```

Data Analysis[Insights]

Top Products by Revenue and Season

```
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
```

Data Analysis[Insights]

Seasonal and Yearly Distribution of Base Colors in Products

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

Data Analysis[Insights]

Multi-Level Revenue and Sales Count by Brand, Style, and Season

```
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
```

Data Analysis[Insights]

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

Customer Segmentation by Spending Levels

```
# 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	1338.469607404418

Data Analysis[Insights]

Potential Revenue Loss

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

# Filter for unsold winter products from previous years
unsold_products = final_df.filter(
    (col("data_year") < 2023) & (col("data_season") == "Winter")
).groupBy("data_productDisplayName", "data_brandName").agg(
    count("*").alias("unsold_count"),
    sum("label").alias("potential_revenue_loss")  # Assuming 'label' is the original price
).orderBy(col("potential_revenue_loss").desc())

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

data_productDisplayName	data_brandName	unsold_count	potential_revenue_loss
Nautica Men Black Dial Chronograph Watch	Nautica	13	156435
Timex Men Black Dial Watch	Timex	22	120140
Giordano Men Black Dial Watch	GIORDANO	19	110450
Titan Men White Dial Watch	Titan	26	105200
Morellato Men Silver Dial Watch	Morellato	9	83250
Fastrack Men Black Dial Watch	Fastrack	41	78745
Giordano Men White Dial Watch	GIORDANO	14	76400
Ray-Ban Men Aviator Sunglasses	Ray-Ban	16	76240
Ed Hardy Men Black Dial Watch	Ed Hardy	8	74980
Citizen Men Black Dial Chronograph Watch	Citizen	7	73200
Titan Men Black Dial Watch	Titan	15	72930
Catwalk Women Black Heels	Catwalk	43	70485
Miss Sixty Silver Dial Watch	MISS SIXTY	8	68660
Titan Men Black Watch	Titan	16	68110
Ray-Ban Men Aviator Gold Sunglasses	Ray-Ban	13	67470
United Colors of Benetton Men Black Sunglasses	United Colors of Benetton	18	64450
Red Tape Men Brown Shoes	Red Tape	26	64370
Polaroid Men Sunglasses	Polaroid	18	64291
Citizen Men Black Dial Eco-Drive Watch	Citizen	6	63600
Maxima Men White Dial Watch	maxima	37	63574

only showing top 20 rows

Data Analysis[Insights]

Distinct Product purchased by each customer

```
from pyspark.sql.functions import countDistinct

# Count distinct products purchased by each customer
customer_loyalty = final_df.groupBy("data_id", "data_gender", "data_ageGroup").agg(
    countDistinct("data_productDisplayName").alias("unique_purchases"),
    sum("data_discountedPrice").alias("total_spent")
).orderBy("total_spent", ascending=False)

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

```
+-------+-----------+-------------+----------------+-----------+
|data_id|data_gender|data_ageGroup|unique_purchases|total_spent|
+-------+-----------+-------------+----------------+-----------+
|35288  |Unisex     |Adults-Unisex|1               |28950      |
|35282  |Unisex     |Adults-Unisex|1               |21950      |
|52686  |Men        |Adults-Men   |1               |21220      |
|59253  |Men        |Adults-Men   |1               |18995      |
|5062   |Men        |Adults-Men   |1               |18900      |
|28438  |Women      |Adults-Women |1               |17995      |
|53014  |Women      |Adults-Women |1               |17500      |
|52691  |Women      |Adults-Women |1               |17150      |
|52690  |Women      |Adults-Women |1               |17000      |
|29945  |Men        |Adults-Men   |1               |16450      |
|29923  |Men        |Adults-Men   |1               |15550      |
|51639  |Men        |Adults-Men   |1               |15495      |
|53020  |Women      |Adults-Women |1               |15426      |
|29950  |Men        |Adults-Men   |1               |15350      |
|29951  |Men        |Adults-Men   |1               |15350      |
|29937  |Men        |Adults-Men   |1               |15350      |
|29934  |Women      |Adults-Women |1               |15350      |
|29944  |Men        |Adults-Men   |1               |15350      |
|53016  |Women      |Adults-Women |1               |15150      |
|52688  |Women      |Adults-Women |1               |15050      |
+-------+-----------+-------------+----------------+-----------+
only showing top 20 rows
```

Data Analysis[Insights]

Pricing Analysis by Gender, Season, and Price Range

```
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

Data Analysis[Insights]

Discount Analysis by Range, Sales Count, and Total Revenue

```
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      |
+--------------+-----------+-------------+
```

ML Model

```
from pyspark.ml.evaluation import RegressionEvaluator

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


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

```
Root Mean Square Error (RMSE) Linear Regression: 0.0999982838823679
[Stage 44:======================================>                  (24 + 8) / 36]
R² Linear Regression: 0.999999996604294
```

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

```
Coefficients LR: [0.0,0.0,0.0,0.9999417273126453,0.0]
Intercept LR: 0.09854435853073856
```

ML Model

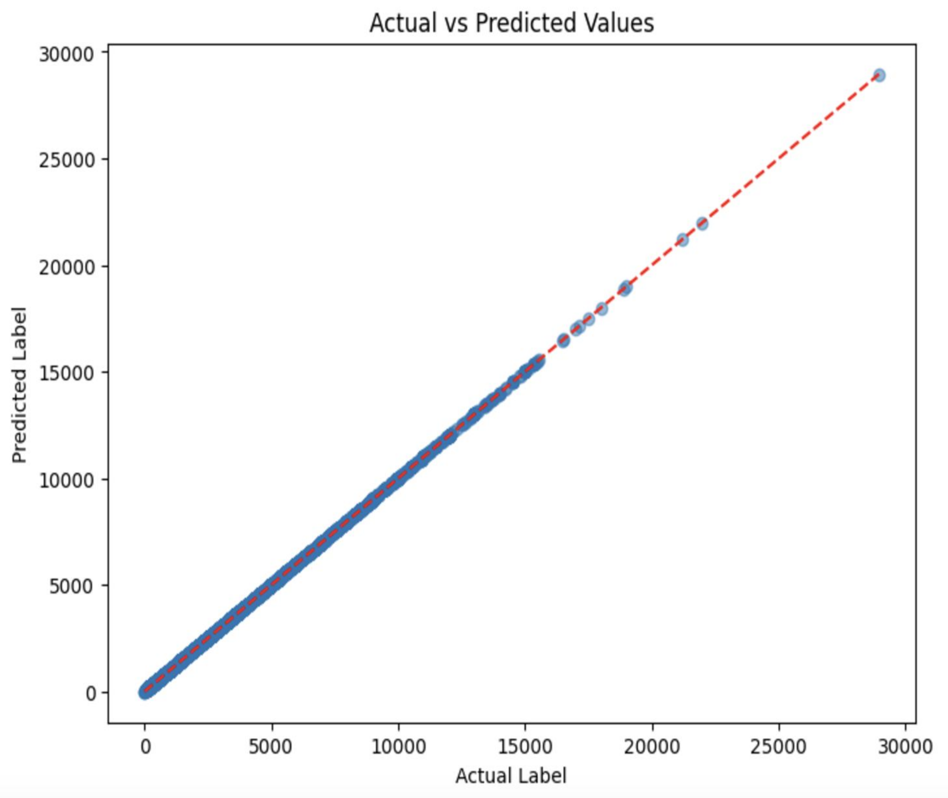
Descriptive Features:

- Categorical Features: Indexed (data_fashionType, data_brandName, data_baseColour)
- Numerical Features: Used directly (data_price, data_year)

Target Feature: data_price

Model Implemented: linear regression
Model Performance:

- **R² Value**: 0.09 - Indicates perfect prediction accuracy.
- **RMSE**: Remarkably low - Signifies the model's predictions precisely match the actual values.
- **Visual**: Small plot showing Actual vs. Predicted Values illustrating the direct correspondence.



Conclusion

1. **Big Data & Machine Learning Integration: Leveraged Apache Spark and Hadoop to process over 44,000 images and metadata, enabling scalable data storage and computation.**

2. **Actionable Insights: Identified patterns in pricing, brand popularity, and demographic trends, aiding retailers in strategic decision-making.**

3. **Machine Learning Success: Applied Linear Regression with high predictive accuracy for inventory optimization and marketing enhancement.**

4. **Industry Transformation: Demonstrated the potential of big data and ML to enhance decision-making, customer experiences, and competitiveness in fashion retail.**

5. **Future Scope: Suggests incorporating real-time data streams and advanced predictive models to refine trend forecasting and customer segmentation.**

Contribution

Avirit Singh- Insights, Report , ML model, Data Pre-Processing

Jay Joshi- Insights, Demo, Data Handling

Tanu Datt- Insights, Report , Data Pre-Processing

Vaibhavi Rao- EDA , Insights, PPT

Varun Patil- Insights, Report, PPT, ML Model

Pragya Priyadarshini- Insights , PPT, Data Pre-Processing