Toronto Metropolitan University

LEVERAGING DATA ANALYTICS FOR SALES OPTIMIZATION AND REVENUE GROWTH

Project Documentation

Prepared by
Rabia Danish

Contents

Problem Definition:

Businesses like Amazon need to know which features of their products determine the success of a sale in the highly competitive realm of online shopping. Price, customer rating, product category, and discount are all important considerations when evaluating a product's sales performance. The challenge lies in correctly judging these factors to make more accurate decisions that enhance sales and please clients. Finding out which product attributes—such as category, price, rating, and discount have the most impact on sales success is the aim of this project. We also want to Know how to maximize customer ratings and sales volume across a wide range of product categories by optimizing the discount percentages.

Data Description:

The dataset chosen for the project is the Amazon Sales Dataset. This dataset is collected from Kaggle. The link to this dataset is given below.

Link: https://www.kaggle.com/datasets/ahmedsayed564/amazon-sales-dataset

i. For a range of products sold on Amazon, it contains important details like prices, ratings, and discounts in addition to sales success. The attributes of the dataset are listed below.

Attributes	**Description**
product_id	Product ID
product_name	Name of the Product
category	Category of the Product
discounted_price	Discounted Price of the Product
actual_price	Actual Price of the Product
discount_percentage	Percentage of Discount for the Product
rating	Rating of the Product
rating_count	Number of people who voted for the Amazon rating

about_product	Description about the Product
user_id	ID of the user who wrote review for the Product
user_name	Name of the user who wrote review for the Product
review_id	ID of the user review
review_title	Short review
review_content	Long review
img_link	Image Link of the Product
product_link	Official Website Link of the Product

b. Statistics of the Data

The dataset contains several raw attributes, such as Actual Price, Discount Percentage, Rating, and Rating Count. To have insight into the analysis of the performance of products, several more derived attributes were computed based on pre-defined formulae. The derived columns drawn out give a better view of sales dynamics, generation of revenue, and how inventories perform at the site. A glimpse of the derived attributes is shown below, with the statistical analysis performed on the raw and derived data.

1. Derived Attributes

The following attributes were calculated in order to extend the dataset and allow further analysis:

Sales Volume: This quantifies demand and helps in identifying products that show high customer engagement and popularity.
Revenue: It gives the amount of money that each product brings in, hence helping in determining the most profitable categories.
Normalized Metrics: These metrics allow for fair comparison across products of different scales. It, therefore, does not rule out smaller-scale products in its analysis.

These derived columns provided a basis for some analysis that was not directly achievable from raw data.

2. Descriptive Statistics

Descriptive statistics for key raw and derived attributes were computed using PySpark. A sample of deriving data statistics is shown below.

```
>>> from pyspark.sql import SQLContext
>>> sqlContext=SQLContext(sc)
>>> df_amazon=sqlContext.read.format('com.databricks.spark.csv').options(header='true', inferSchema='true').load('/user/root/project/amazonv2_normalised.csv')
```

```
>>> df_amazon.select("discounted_price").summary("count","mean","stddev","min","max").show()
+-------+------------------+
|summary|  discounted_price|
+-------+------------------+
|  count|              1465|
|   mean|3125.3108737201364|
| stddev| 6944.304394400707|
|    min|              39.0|
|    max|           77990.0|
+-------+------------------+
```

With a similar approach, data statistics of all the columns are mentioned in the below table.

Attributes	Count	Mean	Standard Deviation	Min	Max
discounted_price	1465	3125.311	6944.304	39.0	77990.0
actual_price	1465	5444.990	10874.827	39.0	139900.0
discount_percentage	1465	0.477	0.216	0.0	0.940
rating	1465	4.096	0.292	2.0	5.0
rating_count	1465	18270.564	42729.995	0.0	426973.0
sales_volume	1465	76160.819	100107.493	0.0	1878681.2
revenue	1465	2.046	6.962	0.0	1.093
NormalizedSalesVolume	1465	0.041	0.096	0.0	1.0
NormalizedRevenue	1465	0.019	0.064	0.0	1.0

Work Distribution:

Name	Contribution
Rabia Danish	Apache Nifi, Kibana
Ahnaf Shahriyar Chowdhury	Pyspark
Syed Ali Javed	Hive
Sameer Ul Haq	Apache Nifi, Kibana

Solution Description:
Tools Used

1. Apache NiFi: Data Ingestion

Apache NiFi is a major tool for data integration and processing; it offers a user-friendly interface based on drag-and-drop and allows real-time monitoring. It helps in smooth data transfer between systems. It's very handy and can be used for transferring huge amounts of data to distributed storage systems, like Hadoop HDFS.

How It Was Used

The steps to use Apache NiFi are mentioned below.

Installing and Configuring Apache NiFi:

- Version 1.28.0 of Apache NiFi was downloaded and installed.
- The default port to access the web interface of NiFi was changed to 8444 to avoid conflicts and to secure the setup.
- The NiFi instance was started and the GUI accessed via https://localhost:8444/nifi

NiFi Flow Setup:

- A GetFile Processor was configured to read the dataset, an Amazon CSV file stored in a local directory.
- A PutHDFS Processor was configured to write the file into Hadoop's HDFS.

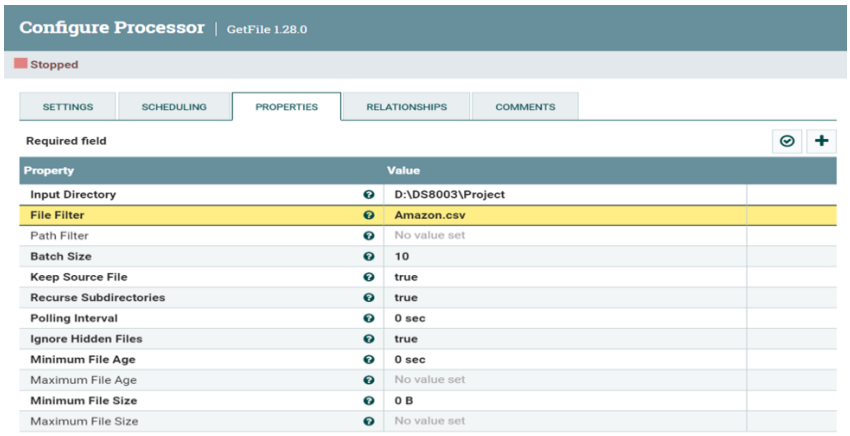
- Both processors were wired together so that the dataset could be routed from the local system into HDFS.

Why Apache NiFi Was Chosen

Apache NiFi was chosen for its ease of use, with a graphical interface that simplifies workflow design without extensive coding. Its pre-built processors, like GetFile and PutHDFS, streamline data ingestion tasks. Additionally, NiFi's scalability supports growing data volumes and varied data formats, while real-time monitoring ensures efficient tracking of file operations, performance metrics and errors.

Code Snippets and Logic Explanation

GetFile Processor:



Purpose: This processor is responsible for reading the CSV dataset from a designated local directory on the system.

Configuration:

· Directory: The absolute path of the folder containing the dataset that is D:\DS803\Project

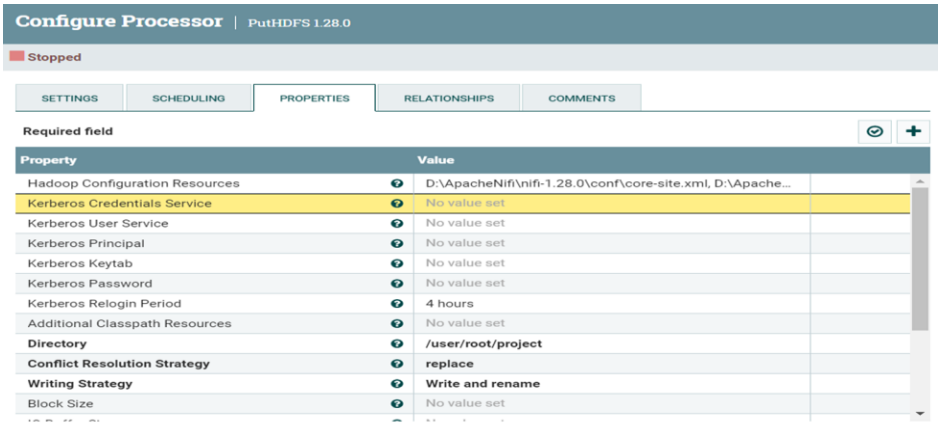
· File Filter: Specifies the name of the file to be ingested that is Amazon.csv

Logic: Once activated, the processor continuously monitors the specified folder. When the desired file is detected, it reads the file and places it in the flow queue for subsequent processing.

PutHDFS Processor:

Purpose: This processor transfers the file from the NiFi queue to a specified location in Hadoop HDFS.

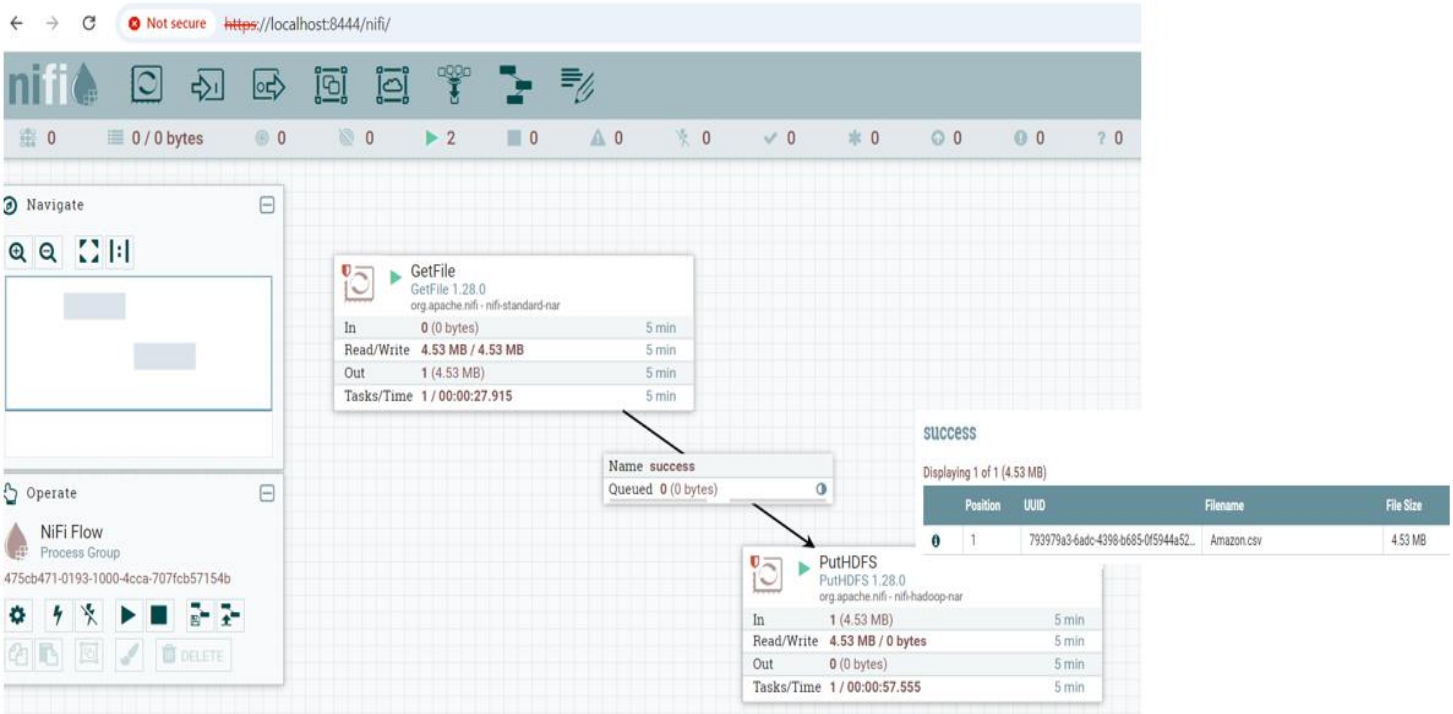
Configuration:



· HDFS Directory: The target location in HDFS where the file will be stored.

· Hadoop Configuration Resources: Configured to reference Hadoop's core-site.xml and hdfs-site.xml files to establish a connection with the HDFS cluster.

Logic: On receipt of the file from the queue, the processor writes into a specified HDFS directory thus integrating it completely with the Hadoop ecosystem. The flow has been started in Ni-Fi, which successfully transferred the 4.53 MB Amazon.csv dataset to HDFS. It could be tracked on the dashboard of NiFi itself to show that it has run successfully.



Toronto Metropolitan University

2. HDFS

How it was used:

HDFS (Hadoop Distributed File System) was used as the storage system for the ingested data. After ingestion through NiFi, the CSV file was stored in HDFS, ensuring scalability, fault tolerance, and efficient data retrieval in a distributed environment. The usage of HDFS is given as an example screenshot below.

Why it was chosen:

HDFS is used because it can handle big data in a distributed manner, hence scalable and redundant. It is well-suited for big data applications, hence the data can be securely stored and easily accessed by downstream tools like PySpark for further analysis.

Logic Explanation:

The ingested data from NiFi was stored in the HDFS directory /user/root/project. Implementation of HDFS usage is given as a screenshot below.

```
[root@sandbox-hdp ~]# cd /root/project
[root@sandbox-hdp project]# hadoop fs -put Amazon.csv /user/root/project
[root@sandbox-hdp project]# hadoop fs -ls /user/root/project
Found 1 items
-rw-r--r--   1 root root    4745677 2024-11-22 21:22 /user/root/project/Amazon.csv
[root@sandbox-hdp project]#
```

This directory works as the location from which the data is retrieved for preprocessing with PySpark.

3. PySpark

How it was used:

This data in HDFS was then pre-processed using PySpark. The PySpark DataFrame reads all the data from the analysis and manipulations into the HDFS directory. Thanks to the power of distributed computing through Spark, the processing tasks on large chunks of data-cleaning, transformation, derivation of new attributes about activities-run seamlessly without hiccups. This preprocessing was essential because most analyses or developments require a serious cleaning of the data before any real work may begin.

Why it was chosen:

PySpark was preferred since it will efficiently process large-scale data with the power of distributed computing. It handles big data in terms of its scale. Besides, PySpark has a really powerful set of built-in functions and is well-integrated with HDFS; which makes it perfect for this pre-processing task.

Code Snippets and Logic Explanation

Data cleaning:

```
>>> spark.sql("""
... CREATE OR REPLACE TEMP VIEW amazon_new AS
... SELECT
...     product_id,
...     product_name,
...     category,
...     CAST(REGEXP_REPLACE(discounted_price, '[₹,]', '') AS FLOAT) AS discounted_price,
...     CAST(REGEXP_REPLACE(actual_price, '[₹,]', '') AS FLOAT)AS actual_price,
...     CAST(REGEXP_REPLACE(discount_percentage, '[%]', '') AS FLOAT) AS discount_percentage,
...     CAST(REGEXP_REPLACE(rating_count, '[,]', '') AS FLOAT) AS rating_count,
...     rating
... FROM Amazon
... """).show()
++
||
++
++
```

Explanation: Following the preparation for analysis, this will make use of Spark SQL to filter the desired data from the created table Amazon and then create an original table, amazon_new, such that the cleaning, on various columns such as discount, actual price, DiscountPercentage, and Rating count removes things like %, ₹,, and \", the REGEXP_REPLACE shall come first by removing these characters while applying a string type variable which would then be eventually translated into the FLOAT form while carrying out the conversion using this code.

Deriving new columns:

```
>>> spark.sql("""
... CREATE OR REPLACE TEMP VIEW amazonv2_derived_columns AS
... SELECT *,
... (rating * rating_count) AS Sales_volume,
... (discounted_price * (rating * rating_count)) AS revenue
... FROM amazon_sales_update
... """).show()
++
||
++
++

>>> spark.sql("DESCRIBE amazonv2_derived_columns").show()
+--------------------+---------+-------+
|            col_name|data_type|comment|
+--------------------+---------+-------+
|          product_id|   string|   null|
|            category|   string|   null|
|    discounted_price|   double|   null|
|        actual_price|   double|   null|
|discount_percentage|   double|   null|
|              rating|   double|   null|
|        rating_count|   double|   null|
|        Sales_volume|   double|   null|
|             revenue|   double|   null|
+--------------------+---------+-------+
```

Explanation: This code creates, from the amazon_sales_udpated dataset, a Spark SQL Temporary view named amazonv2_derived_columns and includes four calculated metrics:inventory_performance-Weighted Turnover using rating_count*rating and Sales_volume Sales_volume - popularity of the product, rating*rating_count. Revenue - estimated sales. discounted_price*Sales_volume Discount_effectiveness - discounted efficiency. (actual_price-discounted_price) / actual_price. These calculations, when combined with the original columns, provide insights into sales, discounts, and product performance while ensuring that the analysis is scalable and modular throughout the session.

Data Normalization:

```
>>> spark.sql("""
... CREATE OR REPLACE TEMP VIEW amazon_normalised_data AS
... SELECT
...     product_id,
...     category,
...     discounted_price,
...     actual_price,
...     discount_percentage,
...     rating,
...     rating_count,
...     Sales_volume,
...     revenue,
...     discount_effectiveness,
...     inventory_performance,
...     (Sales_volume - MIN(Sales_volume) OVER ()) /
...     (MAX(Sales_volume) OVER () - MIN(Sales_volume) OVER ()) AS NormalizedSalesVolume,
...     (revenue - MIN(revenue) OVER ()) /
...     (MAX(revenue) OVER () - MIN(revenue) OVER ()) AS NormalizedRevenue,
...     (inventory_performance - MIN(inventory_performance) OVER ()) /
...     (MAX(inventory_performance) OVER () - MIN(inventory_performance) OVER ()) AS NormalizedInventoryPerformance
... FROM amazon_sales
... """).show()
++
||
++
++
```

Explanation: This SQL would create a temporary view called amazon_normalized_data from the amazon_sales dataset with all the columns of interest-product description, price, and feedback metrics- along with computed values of sale quantity, revenue, and computed value for the performance of inventories. Window functions such as, (sales_volume - MIN(sales_volume)) / (MAX(sales_volume) - MIN(sales_volume)), would then be used in normalizing the three metrics that will come in namely sales_volume, revenue, and inventory_performance onto a scale from 0 to 1; hence, they allow comparing items from a large number of different scales uniformly. This normalization makes it easier to find trends, outliers, and patterns, therefore enabling flexible and insightful data analysis for useful decision-making.

Analysis:

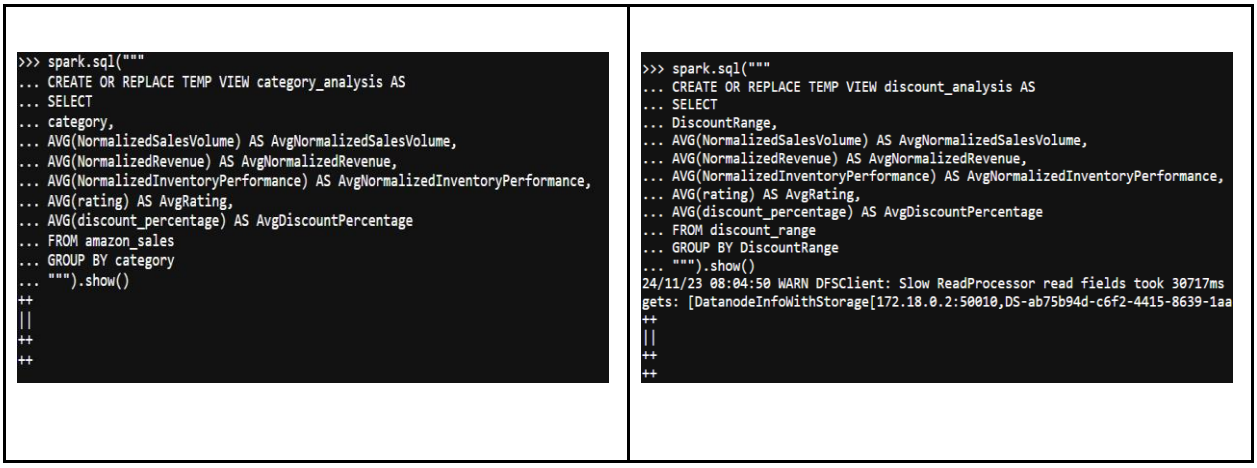


Fig 1: SparkSQL commands for category analysis and discount analysis

Explanation for category analysis: This code will create a Spark SQL temporary view called category_analysis that aggregates and analyzes the important metrics at the category level. It provides the performance of the category in terms of sales, revenue, and inventory efficiency by computing averages of normalized measures such as Normalized Sales Volume, Normalized Revenue, and Normalized Inventory Performance. It also calculates the average rating and discount_percentage to understand consumer feedback and pricing plans, respectively. This view groups data by category, underlining broad patterns that will inform strategic choices on everything from identifying top-performing categories, price optimization, and improving inventory management.

Explanation for discount analysis: This code creates a Spark SQL temporary view, discount_analysis_new, which can be used to analyze discount ranges against some performance metrics across classes of products. It generates averages of variables such as sales volume, inventory performance, and rating, and revenue, and it also ranges the discount_percentage using a case statement, like a range of 0-10%, 10-20%, etc. The grouped analysis shows such trends as the effect of discounts on sales, inventory optimization, and customer evaluations. The temporary view allows for exploratory analysis without changing the

source dataset. It provides a valuable insight into discount tactics and their impact on performance at the category level.

```
>>> spark.sql("""
... CREATE OR REPLACE TEMP VIEW rating_analysis AS
... SELECT
... category,
... CASE
... WHEN rating <= 2 THEN 'Low Ratings (1.0-2.0)'
... WHEN rating > 2 AND rating <= 4 THEN 'Medium Ratings (2.1-4.0)'
... ELSE 'High Ratings (4.1-5.0)'
... END AS RatingRange,
... COUNT(*) AS ProductCount,
... AVG(NormalizedSalesVolume) AS AvgNormalizedSalesVolume,
... AVG(NormalizedRevenue) AS AvgNormalizedRevenue,
... AVG(NormalizedInventoryPerformance) AS AvgNormalizedInventoryPerformance,
... AVG(discount_percentage) AS AvgDiscountPercentage
... FROM amazon_sales
... GROUP BY category, RatingRange
... """).show()
++
||
++
++
```

Explanation for rating analysis: This would create a temporary view named rating analysis, showing how product rating affects performance across categories. Rating ranges are classified into 'Low' ratings between the range of 1-2, 'Medium Ratings' between 2.1 and 4, whereas 'High Ratings' vary between 4.1 and 5 using a standard CASE statement. Group the view by Category and Rating Range - compute Revenue, Inventory Performance, average Normalized Sales Volume, Total Count of the Products, and Discount Percentage:. This segmentation shows the trend of customer satisfaction and its consequences in terms of sales, revenue, and inventory efficiency. This could enable actionable insights toward improvement in product performance based on customer input.

4. Hive

How it was used:

Hive was used to query the data stored in HDFS. After ingesting the data via Apache NiFi and storing it in HDFS, we created a Hive table pointing to that data. This allowed us to run SQL-like queries to analyze the data, such as calculating the average rating per category or sales volume per category.

Why it was chosen:

Hive was chosen because it allows SQL queries on large datasets stored in HDFS. It simplifies

the analysis process, offering an efficient way to work with big data without needing complex programming.

Code Snippets and Logic Explanation

```
hive> select category, count(product_id) AS count_of_products_sold from amazon_norm_data_2024 group by category order by count_of_products_sold DESC limit 10;
Query ID = root_20241102015129_d780107a-a8fb-441b-8cb3-fe5a62bb275e
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1730245551650_0043)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 19.68 s
--------------------------------------------------------------------------------
OK
category        count_of_products_sold
Computers&Accessories|Accessories&Peripherals|Cables&Accessories|Cables|USBCables       233
Electronics|WearableTechnology|SmartWatches     76
Electronics|Mobiles&Accessories|Smartphones&BasicMobiles|Smartphones    68
Electronics|HomeTheater,TV&Video|Televisions|SmartTelevisions   63
Electronics|Headphones,Earbuds&Accessories|Headphones|In-Ear    52
Electronics|HomeTheater,TV&Video|Accessories|RemoteControls     49
Home&Kitchen|Kitchen&HomeAppliances|SmallKitchenAppliances|MixerGrinders       27
Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cleaning&Ironing|Irons,Steamers&Accessories|Irons|DryIrons   24
Electronics|HomeTheater,TV&Video|Accessories|Cables|HDMICables  24
Computers&Accessories|Accessories&Peripherals|Keyboards,Mice&InputDevices|Mice  24
```

Explanation: This HiveQL will calculate how many kinds of goods are sold in each category from the amazon_norm_data_2024 dataset. The COUNT(product_id) function in the query counts the products by category, and the results are grouped by category in descending order of count. LIMIT 10 ensures that only the top ten categories with the highest counts are displayed. This query gives quick category-level insights into the best selling product categories.

```
    > select category, cast(avg(revenue) as bigint) AS avg_revenue from amazon_norm_data_2024 group by category order by avg_revenue DESC limit 10;
Query ID = root_20241102063820_377fe87d-8fda-4465-818d-dcb169a20f01
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1730245551650_0051)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 10.12 s
--------------------------------------------------------------------------------
OK
category        avg_revenue
Computers&Accessories|ExternalDevices&DataStorage|ExternalSolidStateDrives      1890000000
Electronics|Mobiles&Accessories|Smartphones&BasicMobiles|Smartphones    1776130108
Electronics|HomeTheater,TV&Video|Televisions|SmartTelevisions    1171550292
Computers&Accessories|ExternalDevices&DataStorage|ExternalHardDisks     762759796
Home&Kitchen|Heating,Cooling&AirQuality|AirConditioners|Split-SystemAirConditioners     597000000
Computers&Accessories|NetworkingDevices|Repeaters&Extenders     489333333
Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cleaning&Ironing|Vacuums&FloorCare|Vacuums|RoboticVacuums    424000000
Electronics|Cameras&Photography|SecurityCameras|DomeCameras     389100000
Computers&Accessories|Tablets   358000000
Electronics|Mobiles&Accessories|MobileAccessories|Chargers|PowerBanks   354508333
```

Explanation: This HiveQL will return categories in amazon_norm_data_2024 by the highest average revenue. To simplify matters, this uses AVG(revenue) to reduce the average income for each category to an integer value BIGINT. The results will be ordered in descending order of average revenue once the categorization by type is done. Only top ten

categories having the highest average income will be displayed thanks to the LIMIT 10 clause. This aids in determining which product categories yield the most profits.

```
hive> select category, avg(rating) AS average_rating from amazon_norm_data_2024 group by category order by average_rating DESC limit 15;
Query ID = root_20241102044117_c3e400e3-dc3d-44b6-99ad-4a8fd06569b5
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1730245551650_0048)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 5.73 s
--------------------------------------------------------------------------------
OK
category        average_rating
Computers&Accessories|Tablets   4.6
OfficeProducts|OfficeElectronics|Calculators|Basic      4.5
Computers&Accessories|Components|Memory 4.5
Computers&Accessories|NetworkingDevices|NetworkAdapters|PowerLANAdapters        4.5
Home&Kitchen|Kitchen&HomeAppliances|Coffee,Tea&Espresso|CoffeePresses   4.5
Electronics|Cameras&Photography|Accessories|Film        4.5
Electronics|HomeAudio|MediaStreamingDevices|StreamingClients    4.5
Home&Kitchen|CraftMaterials|PaintingMaterials   4.5
Electronics|PowerAccessories|SurgeProtectors    4.5
Home&Kitchen|Kitchen&HomeAppliances|SmallKitchenAppliances|SmallApplianceParts&Accessories      4.5
HomeImprovement|Electrical|CordManagement       4.5
Electronics|Mobiles&Accessories|MobileAccessories|Maintenance,Upkeep&Repairs|ScreenProtectors   4.4714285714285715
Home&Kitchen|Kitchen&HomeAppliances|SmallKitchenAppliances|DeepFatFryers|AirFryers      4.46
OfficeProducts|OfficeElectronics|Calculators|Scientific 4.45
Home&Kitchen|CraftMaterials|PaintingMaterials|Paints    4.433333333333334
```

Explanation: This HiveQL query finds the top categories in the amazon_norm_data_2024 dataset by average rating. It groups the results by category, and the average rating for each category is computed using AVG(rating). The results are sorted in descending order based on the average rating. The LIMIT 15 clause ensures that only the top 15 categories with the highest average ratings are displayed. This query thus gives information on which categories the customers value most when considering ratings.

```
    > select category, avg(discount_percentage) AS avg_discount_percentage from amazon_norm_data_2024 group by category order by avg_discount_percentage DESC limit 15;
Query ID = root_20241102050107_ceea4580-33da-4cb1-aa0e-f62db0dd937a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1730245551650_0048)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 14.71 s
--------------------------------------------------------------------------------
OK
category        avg_discount_percentage
Computers&Accessories|AccessoriesPeripherals|Cables&Accessories|CableConnectionProtectors      0.9
Electronics|Headphones,Earbuds&Accessories|Earpads      0.9
Electronics|Mobiles&Accessories|MobileAccessories|Décor|PhoneCharms     0.9
Electronics|Headphones,Earbuds&Accessories|Adapters      0.88
Computers&Accessories|AccessoriesPeripherals|Keyboards,Mice&InputDevices|Keyboard&MiceAccessories|DustCovers   0.875
Electronics|Mobiles&Accessories|MobileAccessories|Mounts|Shower&WallMounts      0.82
Computers&Accessories|Components|InternalHardDrives      0.8
Computers&Accessories|AccessoriesPeripherals|Adapters|USBtoUSBAdapters 0.7849999999999999
Electronics|Mobiles&Accessories|MobileAccessories|Stands        0.7639999999999999
Electronics|Headphones,Earbuds&Accessories|Cases         0.76
Computers&Accessories|AccessoriesPeripherals|LaptopAccessories|NotebookComputerStands  0.7566666666666667
Computers&Accessories|AccessoriesPeripherals|HardDriveAccessories|Caddies       0.75
Electronics|Cameras&Photography|VideoCameras     0.75
Electronics|Mobiles&Accessories|MobileAccessories|Mounts|Bedstand&DeskMounts     0.745
Electronics|Mobiles&Accessories|MobileAccessories|Décor 0.732
```

Explanation: This HiveQL query will be utilized in finding the categories in amazon_norm_data_2024 with the highest average discount percentages. The query groups the results by category, uses AVG(discount_percentage) to determine the average discount % for each category, and ranks the results in decreasing order by that average discount %. The LIMIT 15 clause ensures only the top 15 categories by average discount are shown. This question helps in a determination of which categories, averagely, provide what amount of savings.

```
    > select category, avg(discounted_price) AS avg_discounted_price from amazon_norm_data_2024 group by category order by avg_discounted_price DESC limit 15;
Query ID = root_20241102061335_a68b8381-7a11-4b3b-9c65-074e4c6da891
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1730245551650_0050)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 4.91 s
--------------------------------------------------------------------------------
OK
category        avg_discounted_price
Home&Kitchen|Heating,Cooling&AirQuality|AirConditioners|Split-SystemAirConditioners     42990.0
Computers&Accessories|Laptops|TraditionalLaptops        37247.0
Computers&Accessories|Tablets   26999.0
Electronics|HomeTheater,TV&Video|Televisions|SmartTelevisions   24840.190476190477
Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cleaning&Ironing|Vacuums&FloorCare|Vacuums|RoboticVacuums    23449.5
Electronics|Mobiles&Accessories|Smartphones|BasicMobiles|Smartphones    15754.441176470587
Home&Kitchen|Kitchen&HomeAppliances|SmallKitchenAppliances|Juicers|ColdPressJuicers     12609.0
Home&Kitchen|Heating,Cooling&AirQuality|AirPurifiers|HEPAAirPurifiers   11917.0
Computers&Accessories|ExternalDevices&DataStorage|ExternalSolidStateDrives      10389.0
Electronics|HomeTheater,TV&Video|Projectors     9990.0
Computers&Accessories|Monitors  8199.0
Electronics|HomeTheater,TV&Video|Televisions|StandardTelevisions        7180.833333333333
Home&Kitchen|Kitchen&HomeAppliances|WaterPurifiers&Accessories|WaterFilters&Purifiers   7015.25
Home&Kitchen|Heating,Cooling&AirQuality|WaterHeaters&Geysers|StorageWaterHeaters        6323.333333333333
Home&Kitchen|Kitchen&HomeAppliances|SmallKitchenAppliances|DeepFatFryers|AirFryers      6276.4
```

Explanation: This HiveQL will determine the most costly categories by average discounted price in a dataset named amazon_norm_data_2024. This gets the average discounted price according to each category using the AVG(discounted_price) and groups by each category. It then orders based on the average discounted_price in descending order. Finally, the LIMIT 15 clause returns only the top 15 categories with the largest average reduced prices. This query shows details about product categories where prices are very high.

5. Kibana - Data Visualization

How it was used:

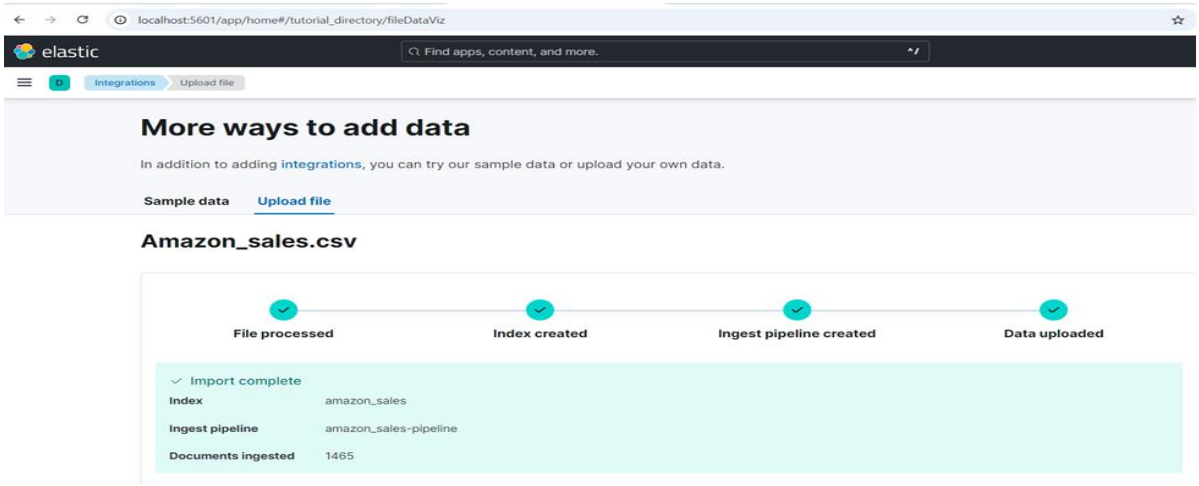
Kibana was used to visualize the data imported into Elasticsearch. After setting up Kibana on my Windows system, I accessed it using the URL http://localhost:5601. To import the data, I navigated to the Search option in the Overview section of Kibana's left-hand menu. From there, I selected the "Import CSV file" option to upload the CSV data, it imports the

CSV file into Elasticsearch for storage and indexing, making the data searchable and available for visualization through Kibana

Why it was chosen:

Kibana was chosen for its robust data visualization capabilities. It allows for seamless integration with Elasticsearch, making it easy to explore and visualize large datasets. The ability to import CSV files directly into Kibana provided a straightforward way to bring the cleaned dataset into the system for visualization, enabling quick insights from the data.

Code Snippets and Logic Explanation



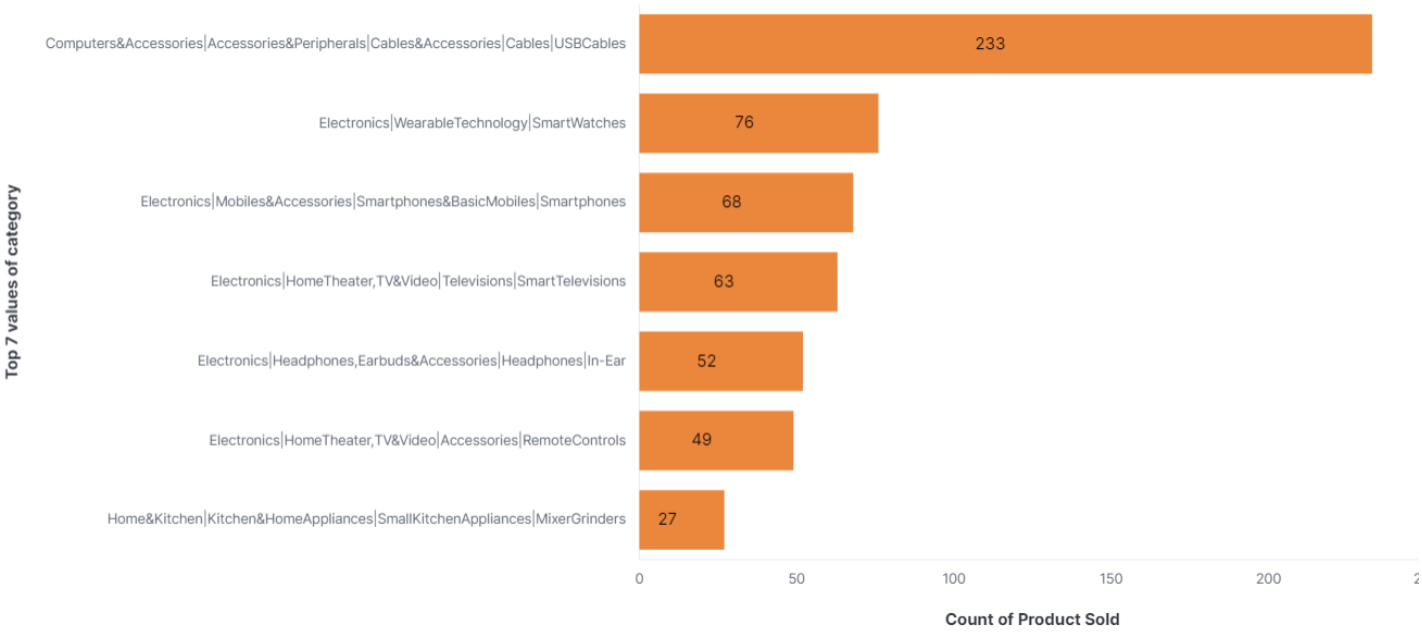
Logic: The image demonstrates the data import process in Kibana for the "Amazon_sales.csv" file. It shows how the data was ingested with steps including file processing, index creation, ingest pipeline setup, and data upload. Each step ensures the CSV data is correctly integrated into Kibana for further visualization.

5 Insights gathered from data

Insight 1: Product Popularity by Category

The bar chart reveals the count of products sold across different categories. The "Computers & Accessories | Accessories & Peripherals | Cables & Accessories | Cables | USB Cables" category is the most popular with 233 units sold, indicating high demand for USB

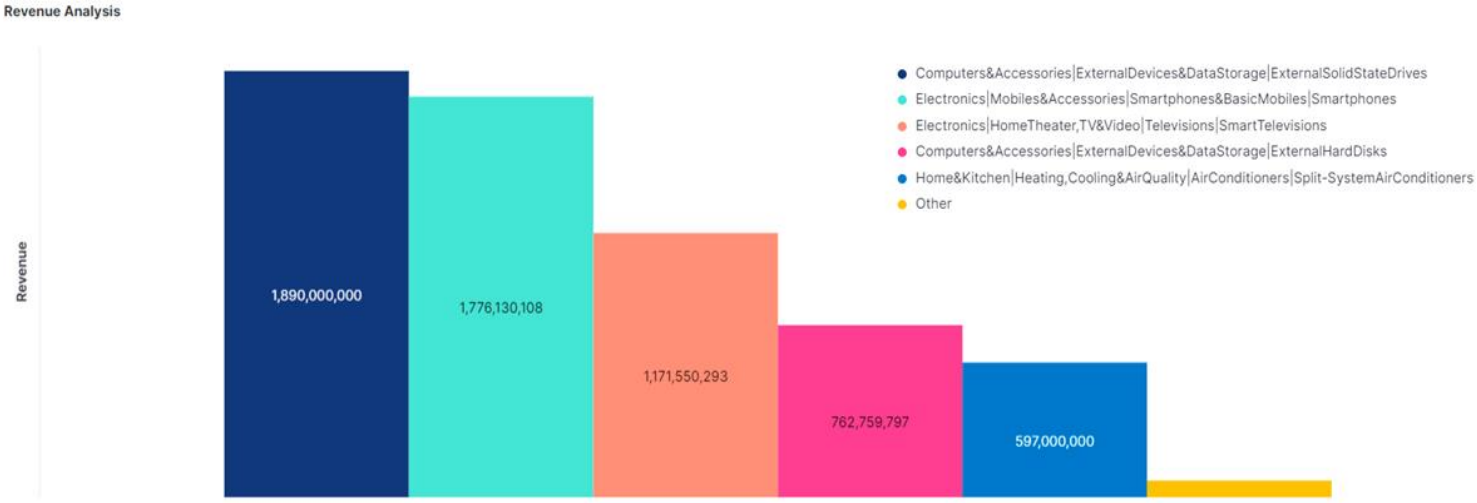
cables. Following this, the "Electronics | Wearable Technology | Smart Watches" category has 76 units sold, suggesting a growing interest in smart watches. Additionally, "Electronics | Mobiles & Accessories | Smartphones & Basic Mobiles | Smartphones" with 68 units sold, and "Electronics | Home Theater, TV & Video | Televisions | Smart Televisions" with 63 units sold, show a steady demand for these products. "Electronics | Headphones, Earbuds & Accessories | Headphones | In-Ear" with 52 units sold, and "Home & Kitchen | Kitchen & Home Appliances | Small Kitchen Appliances | Mixer Grinders" with 27 units sold, are also popular categories. This insight helps identify the most sought-after product categories, enabling retailers to focus on stocking and promoting these high-demand items.



Insight 2: Revenue Distribution Across Categories

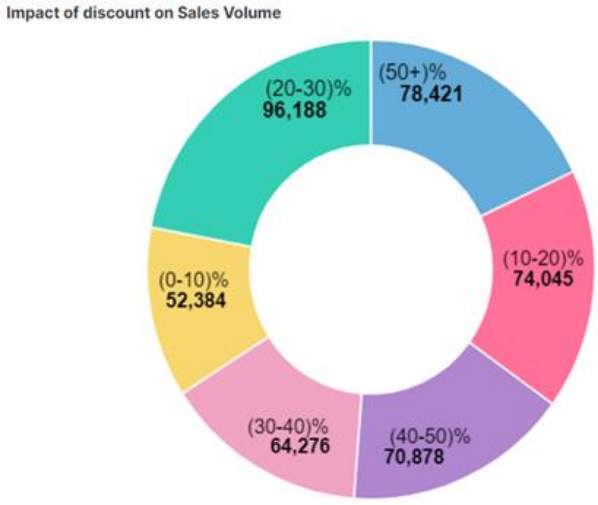
The "Computers & Accessories | External Devices & Data Storage | External Solid State Drives" category generates the highest revenue at approximately 1.89 billion. This indicates a significant demand for external solid-state drives among consumers. Other high-revenue categories include "Electronics | Mobiles & Accessories | Smartphones | Basic Mobiles |

Smartphones" and "Electronics | Home Theater, TV & Video | Televisions | Smart Televisions," demonstrating the popularity and high sales volume of smartphones and smart televisions.



Insight 3: Impact of Discounts on Sales Volume and Revenue

The visualizations highlight the significant influence of discounts on sales and revenue:



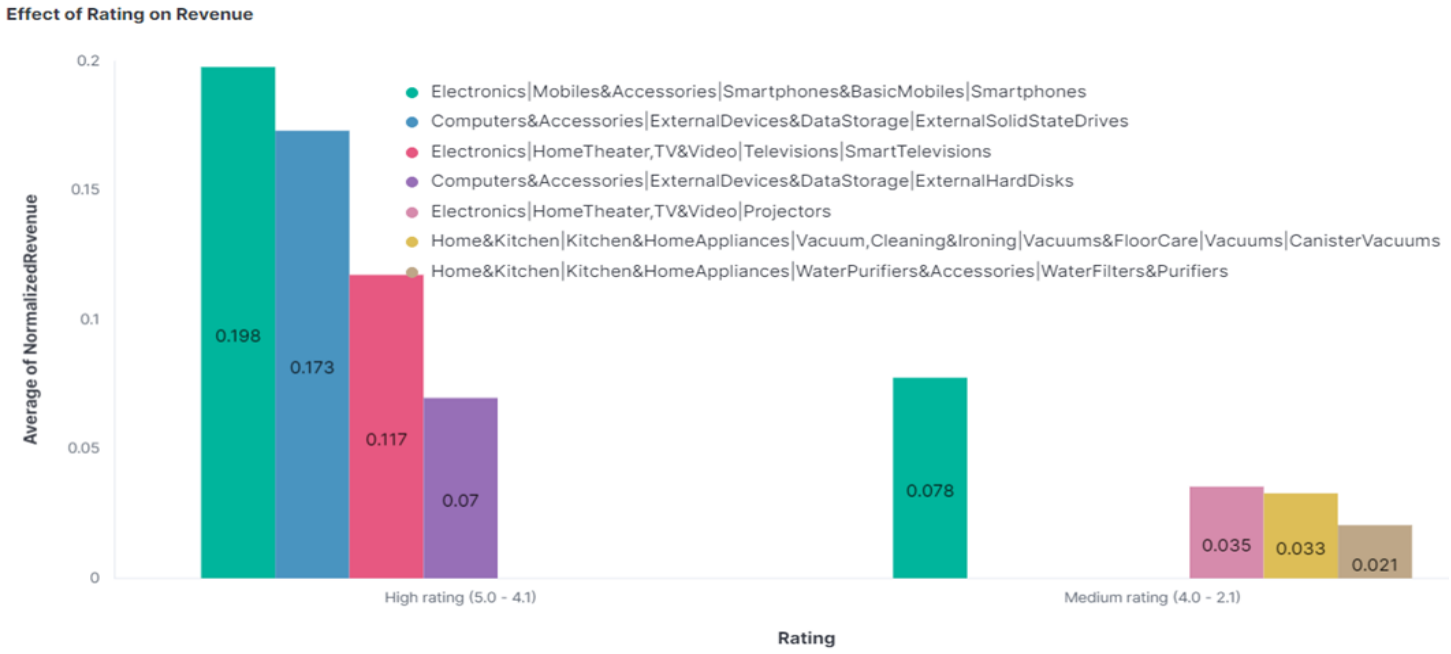
- Sales Volume: The highest sales volume, 96,188 units, is observed within the 20-30% discount range. This suggests that moderate discounts are the most effective in driving sales.
- Revenue: The highest revenue is observed in the 20-30% discount range. This indicates that moderate discounts not only drive the most sales but also generate the highest revenue, making them the optimal discount range for maximizing financial performance.

Retailers might consider focusing on these discount ranges to maximize their overall sales performance and revenue generation.

Insight 4: Effect of Rating on Revenue

From the visualization, higher-rated products tend to generate more revenue:

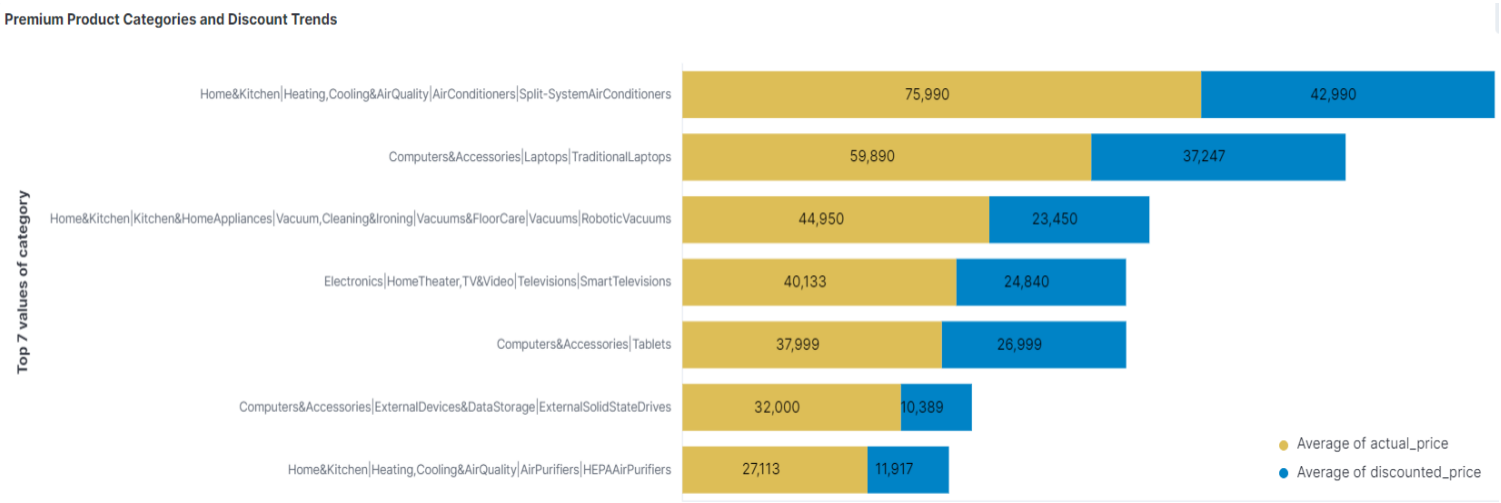
- High Rating (5.0 - 4.1): Categories like "Electronics | Mobiles & Accessories | Smartphones & Basic Mobiles | Smartphones" and "Computers & Accessories | External Devices & Data Storage | External Solid State Drives" have the highest average normalized revenue, suggesting that products with higher ratings are more likely to attract buyers and generate significant revenue.
- Medium Rating (4.0 - 2.1): Categories such as "Electronics | Home Theater, TV & Video | Projectors" and "Home & Kitchen | Kitchen & Home Appliances | Vacuum, Cleaning & Ironing | Vacuums & Floor Care | Vacuums | Canister Vacuums" show comparatively



lower average normalized revenue, indicating that lower-rated products are less likely to generate high revenue.

Insight 5: Premium Product Categories and Discount Trends

This insight reveals that high-priced items often see substantial discounts, particularly in categories like air conditioners, laptops, and robotic vacuums. Retailers can leverage this data to better understand pricing strategies and consumer buying behavior.



Future Work:

Enhancing real-time insights, consumer sentiment analysis, and predictive analytics can be the main goals of future work on this project. Future analytics can forecast sales patterns, improve inventory control, and pinpoint the best discount tactics by utilizing machine learning models. Elasticsearch indexes and Kibana dashboards may be dynamically updated with real-time data pipelines utilizing technologies like Apache Kafka, allowing for real-time insights. Furthermore, a better comprehension of preferences and areas for development may be obtained by integrating consumer sentiment analysis using Natural Language Processing (NLP) of reviews. Geospatial visualizations may reveal geographical patterns, and integrating multi-channel data from physical stores and e-commerce platforms can provide a cohesive picture of performance.

References:

https://nifi.apache.org/docs/nifi-docs/html/getting-started.html#for-windows-users

https://medium.com/turknettech/installation-of-apache-nifi-2856fca5bbdc

https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-hadoop-nar/1.5.0/org.apache.nifi.processors.hadoop.PutHDFS/index.html

https://stackoverflow.com/questions/38898878/how-to-configure-puthdfs-processor-in-apache-nifi-such-that-i-could-transfer-fil

https://community.cloudera.com/t5/Support-Questions/Standard-way-of-giving-Hadoop-Configuration-Resources-in/m-p/216805

https://community.cloudera.com/t5/Support-Questions/SCP-file-transfer-between-Hortonworks-Sandbox-to-Windows/m-p/315727

https://www.elastic.co/guide/en/kibana/current/windows.html

https://www.elastic.co/guide/en/kibana/current/get-started.html

https://medium.com/vedity/python-data-preprocessing-using-pyspark-cc3f709c3c23