INFO 607-001

Dr. Il-Yeol Song

Final Project Report

Spring 2023

**Data Lake and Data Lakehouse Technology**

**Using Databricks**

| Patrick Artac | pa423 | PA |
| --- | --- | --- |
| Allie Schneider | as5664 | AS |
| Francis Villamater | fv48 | FV |
| Zoe Weiner | zrw29 | ZW |

**Table of Contents**

[Project Goals 3](#_ffq1t1osxebc)

[Business Questions 3](#_cq8mxilpir82)

[Data 4](#_qnqs384np7bd)

[Dataset: 4](#_e37kia42empn)

[Installation Guidelines: 4](#_1t8yrsd4fe1u)

[System Architecture: 4](#_wnc1ybkc1sbt)

[Implementation Aspects: 5](#_bzsq27z432he)

[Test Guidelines: 6](#_l8bsdygua5j)

[Issues: 7](#_oiecxrv0614a)

[Data Quality & Pre-processing Issues: 8](#_ilup8ixwit8d)

[Main Methods Employed: 9](#_qp77cxmg82op)

[Input / Results: 9](#_wwd73zc68aiv)

[Interpretation of Results: 12](#_ydkq2m52betj)

[Significance of Project 15](#_7nq9nbz200mv)

[Lessons Learned 15](#_hvfhtkn59ddk)

[Appendix 16](#_v67redfdu88y)

[Diagrams: 16](#_ykh77o88zacy)

[Code / Input: 17](#_6utf7kk44vm9)

[Output: 24](#_kz25b36qv44q)

[References 27](#_pfaup72afeu9)

## 

## **Project Goals**

Our project aims to achieve multiple goals that revolve around implementing a data lake and harnessing the capabilities of Databricks. Despite working with the Databricks Community Edition, we are determined to showcase the full potential of the platform and its impact on data analysis. One of our key objectives is to thoroughly analyze the Store Transaction dataset by leveraging Databricks' powerful tools and functionalities. By utilizing Databricks, we can employ advanced data processing techniques and run complex queries to uncover valuable insights hidden within the dataset. We intend to explore various aspects of the data, including sales performance, customer behavior, and market trends, to gain a comprehensive understanding of the retail environment.

Furthermore, our project seeks to demonstrate the significance and benefits of data lakes and data lake houses, both conceptually and practically. Through the implementation of a data lake using Databricks, we aim to showcase the advantages of a centralized and scalable data storage solution. We will highlight how data lakes enable organizations to store, process, and analyze vast amounts of data, facilitating flexible data exploration and supporting advanced analytics. By presenting the outcomes of our analysis, we aim to emphasize how Databricks and data lakes can drive data-driven decision-making, improve operational efficiency, and fuel innovation. Although we are utilizing the Databricks Community Edition, which has certain limitations in terms of data size and resources, we intend to maximize its capabilities to derive meaningful insights. Overall our project aims to showcase the versatility and power of Databricks, and that even with limited resources, organizations can leverage its functionalities to conduct in-depth data analysis and gain valuable business insights.

## **Business Questions**

1. **Average Value per Unit by Brand:** We determined the average value per unit for each brand, providing insights into pricing strategies and market positioning.
2. **Total Quantity Sold and Total Revenue for Each Product by Brand:** We calculated the total quantity sold and revenue earned for each product by brand, enabling businesses to assess sales performance.
3. **Counts of Different Columns:** We obtained counts of different columns, helping organizations understand data distribution and information availability.
4. **Counts of Goods Sold per Store:** We analyzed the number of goods sold per store, providing insights into sales volume and performance.
5. **Counts of Brands by Place:** We examined brand counts by location, allowing businesses to identify market trends and expansion opportunities.
6. **Count of Categories by Location/Store/Overall**: We determined category counts by location, store, and overall, providing an understanding of product category demand.
7. **Value Sum by Item:** We calculated the total revenue generated by each item, aiding in the identification of high-value products.
8. **Average Product Value:** We derived the average value of a product, enabling businesses to evaluate pricing effectiveness and market acceptance.

## **Data**

### ***Dataset:***

This file contains transaction data across 10 stores for the previous 3 months. Data containing brands, categories, sale total, and unit sales are included.

* 1. Month - Month ID (M1, M2, M3)
  2. Storecode - Store Code (P1, P2, … P10)
  3. Qty - Sales Unit
  4. Value - Sales Value
  5. Grp - Category
  6. Sgrp - Subcategory
  7. Ssgrp - Sub Subcategory
  8. CMP - Company/Manufacturer
  9. Mbrd - Mother Brand
  10. Brd - Brand

### ***Installation Guidelines:***

For our project, we utilized the Databricks Community Edition, a web-based application that eliminated the need for installation. We simply signed up for an account on the Databricks website, granting us access to the Databricks workspace.

Within the workspace, we could write and execute Python and PySpark code, run SQL queries, and perform data analysis tasks. The platform's built-in capabilities for Python, PySpark, and SQL allowed us to work seamlessly without requiring any additional installations. This streamlined process enabled us to focus on building a data lake, conducting analytics, and deriving insights from the Store Transaction dataset.

### ***System Architecture:***

The Databricks architecture enabled us to collaborate securely and efficiently with backend services, freeing our group of the load of system management. The architecture is made up of two fundamental parts: the control plane and the data plane. Databricks manages backend services such as notebooks, jobs, and clusters in the control plane, using its own cloud service account to assure data at rest encryption. The data plane, on the other hand, provides a dedicated space for hosting notebooks, tasks, and data processing driven by Apache Spark clusters. Within our cloud service account, we were able to securely store our data lake and easily access its outcomes. This complete design not only streamlines our operations, but also leverages the capabilities of Databricks and Apache Spark for efficient data processing and advanced analytics, allowing us to gain insights from our data assets.

### ***Implementation Aspects:***

***Building a data lake in Databricks:*** Below you will find the necessary steps required to build a data lake within the Databricks Community edition:

1. ***Define the Use Case:***Understand the business problem or use case that needs to be solved. This will help guide the collection, storage, and analysis processes of building the datalake.
2. ***Data Ingestion:***The collection, processing, cleaning, and importing of data into the data lake takes place in this step. Databricks can ingest structured, semi-structured, and unstructured data from various sources, including real-time data streams or cloud data storage. Databricks, based on Apache Spark, enables the ingestion of structured, semi-structured, and unstructured data. Utilizing Sparks’s built in libraries affords Databricks the ability to read these different data formats. For semi-structured or unstructured data, Spark converts this data into DataFrames for analysis, so structure is imposed on the once unstructured data. For our project, we have imported our data via .csv file, noting that the Databricks Community Edition allows for only 1GB of data to be ingested, before a more advanced version of Databricks must be purchased.

***Data Storage:***This step focuses on data ingestion using the Databricks File System (DBFS) that stores data in cloud storage like AWS S3 or Azure Blob Storage.

***To upload the data to Databricks, adhere to the following procedure:***

1. ***Create a notebook:***In the Workspace area in Databricks, select *“Create”* and then *“Notebook”*. Name the notebook and select the default language which can be either Python, Scala, or SQL.
   1. ***Create a data cell:***In the first cell of the notebook use the command *“%fs”* to interact with the Databricks FIle System (DBFS). Press SHIFT + ENTER to create a new cell.
   2. ***Upload the file:***In the new cell, type *“%fs upload”*. Click the now visible button *“Upload to DBFS”*. Now select a file from your computer to upload. Click *“Open”* after selecting the correct file. Once this file is uploaded, the data will be accessible to use. Another way to upload a data file will be to designate a variable called *“file\_location”* and set that to the location of the file on the system being worked on. Designate what the file type is by creating a variable *“file\_type”* and setting that to the type of file being uploaded (in the case of this project, the file is a CSV). Delegate CSV options for the code by setting up the following variables in the Databricks code: *“infer\_schema”* set to *false*, *“first\_row\_is\_header”* set to *true*, and “delimiter” set to *“,”.* Once these components are set, create a dataframe following the following framework, Once this df is run, the data will be uploaded into Databricks.  
       *df = spark.read.format(file\_type) \a  
      .option("inferSchema", infer\_schema) \  
       .option("header", first\_row\_is\_header) \  
      .option("sep", delimiter) \  
      .load(file\_location)*
   3. ***Data Processing and Transformation:***Once the data is uploaded to Databricks, processing and transformation can happen using Databricks Spark integration. This step includes cleaning, structuring, and transforming raw data into a format that is easier to analyze.
   4. ***Data Analysis and Visualization:***Once the data is uploaded, stored, processed, and cleaned, users can create different notebooks to analyze the data using languages like Python, SQL, Scala, and R. There are also data visualization capabilities as Databricks can integrate with Tableau, PowerBI, and more. For this project, no data visualization components will be integrated.

### ***Test Guidelines:***

We adopted test policies that were closely linked with our research goals and the nature of our analysis for our present project, which uses the Store Transaction dataset from the Databricks Community Edition. These rules were very important in ensuring the validity, accuracy, and reliability of our research findings.

We started our project by employing core testing concepts and a systematic strategy. This required careful test planning, during which we established the goals, parameters, and strategy of our implementation procedure. We made sure that our study efforts were concentrated on and directed toward reaching our anticipated outcomes by creating defined objectives.

Configuring a cluster on the Databricks platform was one of the crucial tasks in our testing procedure. To maximize the speed and scalability of our inquiry, this entailed carefully picking the right cluster size, efficiently allocating memory, and selecting the compatible Spark version. After setting up the cluster settings, we performed our workbooks within the Databricks environment without any issues. We performed an in-depth analytical investigation on the Store Transaction dataset by utilizing the extensive features provided by Databricks, including Python, PySpark, and SQL.

The Store Transaction dataset included transactional data from a retail environment. It covered a wide range of factors, including brand names, categories, sales values, and store codes. We have many possibilities to examine correlations, trends, and insights relating to consumer behavior, sales performance, and market dynamics thanks to this large and diversified amount of information.

We paid close attention to data preparation in our testing strategy. The dataset was meticulously selected to ensure its accuracy and excellence. As a result of this process, we were able to work with representative and pertinent data, which was essential for maintaining the accuracy and dependability of our study. We created particular business issues that functioned as our test cases using an organized process akin to test case design and execution. We were able to evaluate the effectiveness and performance of our implementation thanks to the questions' definition of the evaluation's scenarios and objectives.

We did a thorough analysis of the outcomes following our analysis, similar to the analysis of test results. We carefully examined the results, seeking out trends, correlations, and patterns, and deriving important inferences from the information. This analysis was essential in determining the accuracy of our implementation and the degree to which our research goals were met.

We also adopted the tracking and reporting of defects throughout the project. This involves finding any problems, contradictions, or weaknesses in our analysis. We improved our models, methodology, and data interpretation by addressing and overcoming these problems, ensuring the correctness and validity of our findings.

In conclusion, we built a comprehensive testing strategy for our project by following the test standards. By using this strategy, we were able to evaluate our implementation, guarantee the accuracy of our research results, and offer insightful contributions to our area of study. We had a strong analytical foundation thanks to the Store Transaction dataset and Databricks' potent capabilities, and the test instructions directed us toward an exhaustive and complete examination.

### ***Issues:***

While implementing a data lake using Databricks for this project, we encountered a couple of issues. Firstly, we were unable to incorporate different data structures. Our Store Transaction dataset consists of structured data in a tabular form. It would have been beneficial to explore how Databricks handles, manages, and integrates unstructured or semi-structured data into our data lake. This would have provided insights into the versatility of Databricks in handling diverse data structures.

Another challenge we faced during the project was the absence of live data for real-time updates to our data lake. One of the major advantages of storing transactional data, like ours, in a data lake is the ability to access and manage real-time data. However, as we are not an actual store with daily transactions, we were unable to fully experience the functionality of a Databricks data lake in handling and processing regularly updated real-time data.

Lastly, due to the use of the community edition for this project, we were unable to utilize the collaboration feature of Databricks. This feature enables different users to collaborate and work together on a project, enhancing productivity and knowledge sharing. While the community edition restricted our ability to collaborate within Databricks, we still managed to collaborate effectively using other communication and collaboration tools outside of the platform.

Despite these limitations, our experience with Databricks in implementing a data lake was overall positive, and we were able to gain valuable insights and perform meaningful analysis using the available tools and functionalities.

### ***Data Quality & Pre-processing Issues:***

In our data quality and pre-processing analysis, we encountered a few small issues that required attention. Firstly, we observed that some columns in our dataset, specifically the transaction results of each store location and product, contained both “*0*” and “*NULL*” values. To ensure the accuracy of our analysis, we decided to exclude these 0 values from our calculations. We utilized SQL queries within our analysis to pre-process and filter out these values to create a more seamless result for our analysis.

Additionally, we assessed the correctness of the data types within the dataset. We verified that each attribute was assigned the appropriate data type to ensure accurate computations and analysis. While the usability score of the dataset obtained from Kaggle.com was initially high, we found that the recorded time frames in the dataset did not meet our desired level of granularity. The dataset only provided a monthly overview of the stores' transaction history, without specific timestamps or dates for individual orders. The limited availability of transaction time frames hindered our ability to perform more detailed time-series analysis or identify patterns at a finer temporal scale.

Overall, despite the need for some pre-processing steps to address the presence of “*0*” and “*NULL*” values and ensure data type correctness, the Store Transaction dataset exhibited high usability and completeness. We conducted a thorough review of the dataset prior to its ingestion into Databricks to ensure the quality and reliability of the data for our analysis.

### *Main Methods Employed:*

In our analysis, we utilized several key methods to extract insights and derive our conclusions from the Store Transaction dataset. First, we leveraged the Parquet format for effective storage and retrieval of the data. By grouping data into columns, Parquet, a columned storage file format, helped to speed up query performance for our analysis.

To load and manipulate the dataset, we utilized Python, which provided us a versatile environment for data manipulation, allowing us to load the dataset into a dataframe and perform various data transformations, filtering, and computations. Additionally we leveraged various SQL queries to conduct our analysis on our dataframes. We were able to formulate specific queries and draw relevant transactional data from the dataset such as (Product x Store), (Product x Store x Sales) (Product x Store x Cost) etc. SQL's straightforward syntax and flexibility allowed us to collect, filter, join, and analyze the data in a structured and effective manner.

By combining the capabilities of Parquet, Python, and SQL, we were able to do a full analysis of the Store Transaction dataset. The Parquet format optimized our data storage, Python facilitated data manipulation and transformation, and SQL queries enabled us to extract/filter the desired insights from our dataset. We were able to identify patterns, trends, and relationships in the data using these techniques, which helped us gain a better understanding of the dynamics of store transactions and support defensible decision-making.

### *Input / Results****:***

**Data Ingestion (Python):**

The data ingestion involved us loading the Sales Transaction Table dataset from a CSV file into a DataFrame. We chose this approach to leverage the capabilities of Databricks, which supports reading and manipulating large datasets. By using the PySpark API, we were able to take advantage of the distributed computing to handle the dataset effectively. We specified the file type, delimiter, and other options to ensure proper parsing of the CSV file.

**Data Pre-Processing:**

The data pre-processing phase focused on preparing our dataset for further analysis by addressing data quality issues and to ensure consistency. We performed an initial SQL query to form a descriptive analysis of the dataset so we could examine the data types of each column (*Pre-Pro 1)*. This step was crucial for identifying any inconsistencies in data types, if we did not perform this, we found that it can lead to errors in subsequent analysis.

We then performed a SQL query to check the dataset for ‘0’ values in the 'qty' and 'value' columns. Filtering out rows with ‘0’ values was essential to exclude insignificant data points from future analyses. By removing these rows we ensured that further analysis was based on meaningful and relevant data. (*Pre-Pro 2)*

We then created a new table, "sales\_data\_3," that excluded rows with ‘0’ values in 'qty' or 'value'. This filtered dataset provided us a cleaner and more focused subset *(Pre-Pro 3)*. By eliminating rows with ‘0’ values, we found that we could avoid skewing calculations and obtain more accurate insights into the sales transactions. We had to modify the data types of the 'qty' and 'value' columns to decimal (10,0) in the "sales\_data\_3" table for more accurate numerical calculations. We also dropped the original 'qty' and 'value' columns from the "sales\_data\_3" table to remove columns with incorrect data types. Doing so reduced clutter in the dataset by retaining only the columns with the correct data types.

**Data Analysis:**

To begin our data analysis we first focused on calculating descriptive statistics for the quantity (qty\_num) column to view the distribution of the quantity sold, including the count, minimum value, maximum value, mean, sum, standard deviation, and percentiles (25th, 50th, and 75th) *(Analysis 1)*. Similarly, we obtained descriptive statistics for the value (value\_num) column. This afforded us a comprehensive understanding of the sales value's performance and variability across different transactions. *(Analysis 2*)

To gain insights into the pricing dynamics and variability of individual sales units, we calculated descriptive statistics for the (unit\_value) column. Similar to the previous queries analyzing the percentiles (25th, 50th, and 75th). These statistics offered valuable insights into the pricing and profitability associated with individual sales units. *(Analysis 3*)

Additionally we executed queries to count the number of unique months and unique values in various columns of the “sales\_data\_3” table. The count of unique months provided us an understanding of the temporal coverage of the dataset and the frequency of data collection. By assessing the number of unique months, the granularity and time span of the data could be determined and also the count of unique values in different columns, including “MONTH”, “STORECODE”, “GRP”, “SGRP”, “SSGRP”, “CMP”, “MBRD”, “BRD”, and a concatenated column. These counts helped us assess the breadth of categories, brands, and store locations, and the overall average value per item present in the data. *(Analysis 4*)

Furthermore, we calculated the total sales value for each store (STORECODE) in our table table. We did this by summing the value column (value\_num) for each store and the results of the (TotalRevenue) in descending order. Additionally we conducted a time series analysis on the dataset where we focused on identifying the best-selling categories at each store based on the total quantity sold and total revenue generated. For each store-category combination, we determined the total quantity sold and total revenue by grouping the data by (STORECODE) and ‘grp’ (product category). Within each (STORECODE), the results were ranked, and we were able to identify the categories with the greatest amount of sales. Through this analysis we were able to learn more about the sales trends of various categories and how they affected total revenue by tying store-level revenue data to the best-selling categories.

We also evaluated the average quantity sold and quantity standard deviation for each store. By grouping the data by store (STORECODE) and ‘BRD’ (brand), we calculated the total quantity sold for each store-brand combination. From this information, we derived the average quantity sold and the standard deviation of quantity. These metrics provided insights into the sales performance and variability of different brands at each store. Understanding the average quantity sold and its variation helped identify stores with consistent sales patterns and those with more fluctuating sales.

Further queries provided views of various aspects of store performance and sales dynamics. For example, we analyzed the number of unique products sold by each store throughout the three-month period, offering insights into the product variety offered by different stores. Additionally, we calculated the revenue per unit sold and revenue per sale for each store, enabling us to assess the profitability of sales transactions. These metrics helped identify stores with high revenue per unit or per sale, indicating potential areas for store improvement.

By exploring the variation in sales performance across months, we identified the stores with the greatest (number of brands sold) each month. This analysis highlighted stores that consistently offered a diverse range of products, indicating their ability to cater to customer preferences. Additionally, we identified the largest order values by store and by month, providing views of high-value transactions.*(Analysis 5*)

Finally, we examined the revenue per unit sold and revenue per sale by month, allowing us to identify the months with higher revenue per unit or per sale. These findings helped uncover potential seasonality or trends in sales revenue, which in theory could help these organizations in their marketing efforts, and inventory management. *(Analysis 6*)

In conclusion, our standard analysis of the data set, and time series analysis provided comprehensive insight into the various aspects of each store's performance, sales dynamics, and profitability. These findings contribute to a deeper understanding of the dataset, empowering data-driven decision-making to promote effective strategies to enhance each store's performance.

### *Interpretation of Results:*

We first aimed to analyze how well certain stores are doing in terms of sales. With data on total sales, average quantity, and standard deviation of quantity, each row of our results represented a particular store. We found that the overall sales throughout the stores range from 10,432 to 34,041 with all differing significantly. We found that Store P4 held the highest overall sales, highlighting its high revenue generation.

When we look at the average quantity, we can see that the stores differ from one another. With an average quantity of 91.75 units sold. Store P4 once more leads the pack, indicating that customers here tend to make larger or more frequent purchases each transaction. Store P5 has the greatest value of the quantity standard deviation, which calculates the variability in the number of things sold, at 295.06. This implies that the store suffers more variations in sales amounts compared to other stores because it reflects a greater variety of quantities sold.

With this information we were able to gather important insights into sales performance, typical transaction sizes, and sales fluctuation across various stores. In terms of total sales and average quantity per transaction, Store P4 shows strong performance while Store P5 displayed more variation in sales volume. If applied, these conclusions could help the store management in making decisions about resource allocation, inventory management, and sales tactics *(Output 1)*.

We then examined the distribution of sales across various categories in our analysis of "Sales Distribution by store" using a pie chart *(Output 2)*. The pie chart consisted of six slices, each representing the percentage contribution of a specific category to the overall sales.

The largest slice in the pie chart represented the "Other" category, which accounted for 57.1% of the sales distribution. All other products were lumped into “Other” for this pie chart for visual purposes only to highlight the top 5 categories’ contributions to the entire sales distribution. If this was not applied, considering that there are 1,904 products in total, several other slices would be crowding the figure and make the overall pie chart incomprehensible.

The third-largest slice represented "Biscuits - Core & NON Core," accounting for 11.2% of the sales distribution. This implies that both core and non-core varieties of biscuits have a significant market presence and contribute significantly to the overall sales. "Toilet Soaps" constituted 8.6% of the sales distribution, which showed a moderate but noteworthy contribution to the total sales, suggesting that toilet soaps are a popular product category among consumers. While Salty Snacks accounted for 7.2% of the sales distribution, representing a moderate share in the overall sales. We were able to derive that there is a consistent demand for salty snacks but not enough to drive sales upward. Lastly, the "Shampoo" category occupied the smallest slice of the pie chart, representing 3.6% of the sales distribution. Although shampoo has a smaller share compared to other categories, it still was shown to hold significance in the overall sales mix. The pie chart afforded us a visual representation of the sales distribution by category, allowing us to quickly assess the relative importance of each category.

We also utilized a pie chart *(Output 3)* to graphically display the sales distribution across distinct stores present in our data set where we assigned a percentage that corresponded to each store's share of total sales. When we examined the statistics, we saw that the distribution of sales between the locations varied. With a large share of 17.1% in the sales distribution, Store 8 emerged as the best performer suggesting that Store 8 has a significant sales presence and provides a sizable share of the total revenue across all store locations. Store 6 was not far behind, having the second-largest portion of the pie with 14.0% of the total sales distribution. This implies that Store 6 contributes significantly to the overall revenue and also has a big impact on overall sales performance. In the distribution of sales, stores 3 and 10 each have shares of 10.2% and 10.3%, respectively showing that the sales performance of these stores is generally stable and accounts for a sizable share of total revenue.

The proportions of other stores {1, 2, 4, 5, 7, and 9}, range from 5.6% to 9.9%. Even though these stores may not have the biggest slices, they have a significant impact on how sales are distributed and help drive overall revenue. Our analysis of sales distribution uncovers variances in the ways that various stores contribute to total revenue. Store 8 stands out as the highest performer with Store 6 is closely behind. Although at various rates, the remaining stores also contribute significantly to the sales distribution.

Furthering our analysis we concentrated on the overall quantity sold for each brand while analyzing the "Top quantities by brand" using a bar graph *(Output 4)*. The "X" axis lists the brand names, including Everest, Balaji, RIN, Parle-G, and Yellow Diamond, while the "Y" axis shows the overall number sold. The "Y" axis of the bar graph, which ranges from 0 to 12000, is divided into increments of 2,000 to represent the quantities sold for each brand. Everest was the best-selling brand of the brands examined, with a total of 11,497.0 units sold. This large sales volume implies that the Everest brand is in high demand and is well-liked amongst the customer base.

The bar graph also shows Balaji as the next best-selling brand, with a total number of units sold of 10,704.0. This suggests that Balaji has a large consumer base and a strong market position. RIN is the third-placed brand, selling a total of 5,716.0 units. Despite selling fewer units than the top two brands, RIN still shows a substantial market share and strong consumer demand. Fourth place is held by Parle-G, which sold 4,949.0 units in total. Even though Parle-G sells less than Everest, Balaji, and RIN, it still has a sizable market share. Lastly, with a total of 4,813.0 units sold, Yellow Diamond ranks fifth in terms of quantity. Despite having the lowest sales volume among the brands we analyzed, it nonetheless has a client base and makes a contribution to the total market share.

The sales success of each brand can be quickly and visually understood thanks to the bar graph, we were able to see that Everest and Balaji were the top-selling brands in the dataset. These results offer useful information for business decision-making, including recognizing the successful brands, and can also answer business related questions such as allocating resource allocation, and how to strategically place brands among the top performing stores.

We continued our analysis with a box plot study, where we found some interesting patterns and trends *(Output 5)* of the quantities sold in months 1-3 (M1, M2, M3) across all storecodes. For the analysis, the amounts were divided into intervals of 100 (0, 100, 200, 300, 400, 500, 600). The box plot for month 1 (M1) showed that the biggest concentration of quantities sold fell between 1 and 250. The quantity sold at a select few establishments, however, exceeded 600, indicating extraordinarily high sales in those particular locations. This indicates that while the majority of retailers reported sales in the range of 1 to 250, certain stores reported noticeably higher demand, which was shown outside of the typical range.

Moving on to month 2 (M2), the box plot revealed a spaced-out distribution of sales volume past the 250-unit threshold. Similar to M1, the majority of stores had sales quantities of 1 to 250. However, there was a decline in the number of shops with more than $250 worth of sales. We found that there was a reduction in overall sales compared to M1 with the largest quantity sold in M2 being around 550.

In contrast to the first two months, month three (M3) showed a clear pattern. The box plot revealed a considerable rise in sales, with more stores selling greater than 600 units. Through the box plot analysis we were able to see a noticeable increase in product demand this month. We then derived that there is potential for improved income and profitability during this time period by the fact that Month 3 saw the highest quantities sold among the three months available in the dataset. Although if a larger time frame was available, it is possible that other months could disprove our analysis.

In conclusion, our box plot study showed that while most retailers sold quantities between 1 and 250 over the first two months, several locations stood out with remarkably high sales numbers. However, during the third month, sales volumes significantly increased and more stores had exceeded the 600-item threshold. These results show a variety of sales patterns and emphasize the significance of comprehending the dynamics of sales quantities over time. If applied, this information could be used to improve inventory control, resource allocation, and guide businesses on how to take advantage of spikes in demand to increase total sales and profitability.

## Significance of Project

The decision-making and data analysis fields greatly benefit from our project. We were able to use Databricks, a robust data analytics platform, to take use of its cutting-edge tools and capabilities in order to mine the Store Transaction dataset for insightful information. In order to do extensive data manipulation, processing, and analysis, Databricks supplied us with a comprehensive platform that allowed us to smoothly combine several programming languages including Python, PySpark, and SQL.

The importance of Databricks resides in its capacity to manage massive amounts of data and efficiently carry out complex queries. By processing and analyzing massive datasets in parallel using Databricks, we were able to dramatically cut the amount of time it took to process data. Apache Spark is a powerful distributed computing platform. We were able to analyze the dataset, extract critical indicators, and acquire insightful information about sales performance, consumer behavior, and market trends thanks to the SQL queries we conducted on Databricks. Our study was further strengthened by Databricks' interface with other visualization tools like Tableau and PowerBI, which allowed us to produce powerful visual representations of our results.

For businesses aiming to acquire a competitive edge through data-driven decision-making, Databricks' capabilities are essential. The platform's efficiency and scalability make it ideal for managing big data analytics, allowing businesses to find hidden trends, make wise decisions, and spur innovation. Organizations can process and analyze massive, complex information more quickly and effectively using Databricks, which improves operational effectiveness, customer insights, and strategic decision-making. The project demonstrated how Databricks can enable businesses to gain useful insights from their data, allowing them to remain competitive in today's data-driven environment.

## Lessons Learned

Over the course of this project, we have learned a lot about Databricks. We have familiarized ourselves with the capabilities of Databricks, what it can do, and how to use it. We discovered collaboration features offered by Databricks, which allows multiple users to collaborate and work together on a specific project. However, as we discussed in the issues section, we were not able to fully explore these features due to the use of the community version.

Additionally, we noticed some interesting security features that Databricks offers its users including pop ups when adding or removing sources from a project. We found this to be a nice feature as it stands as a protection from the user making an accidental change in sources. The user interface of Databricks is well thought out and easy for the user to interact with.

Another feature of Databricks that we were not able to explore due to the limited scope of our data and the use of the community edition is the ability to incorporate mixed data structures. A data lake on Databricks allows the user to store, manage, and access data of all structures in one data lake. This can be extremely valuable to a user as it allows for all of their data to be stored in one place, which makes access and processing data simpler.

Additionally, we have learned that Databricks has the ability to use multiple languages in one notebook. This can be very beneficial to the user. In our project, we uploaded and processed our dataset using the language Python, and performed our exploratory analysis using queries written in SQL.

Finally, we have learned about a Parquet format, which provides for a more efficient compute process as it is less expensive in memory and resources than a csv format on a cloud platform.

## Appendix

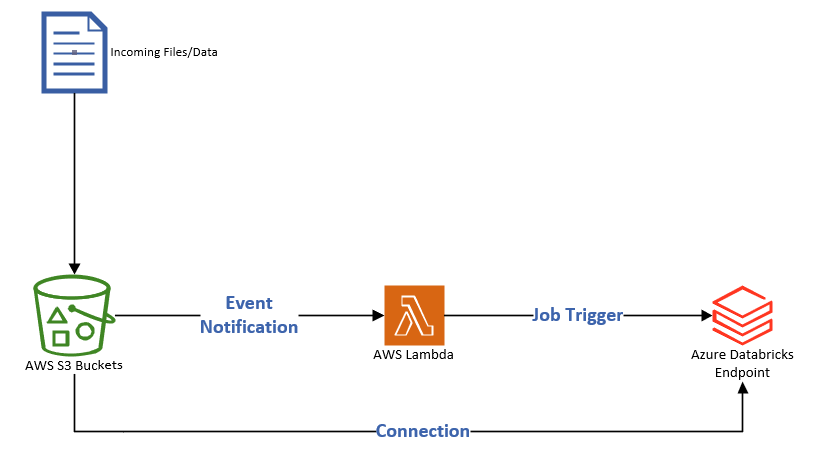
### *Diagrams:*

When Databricks is integrated with AWS S3 and a listener such as AWS Lambda, Databricks offers a potential solution for managing a steady stream of data. Through this integration, scalable and effective data ingestion, processing, and analysis are made possible. Data sources or files are first uploaded to AWS S3 which is configured for multiple incoming data sources, acting as the main location for storing incoming data. AWS Lambda, a serverless compute service, operates as the listener after the files have been uploaded and keeps an eye out for any new data arrivals in the S3 bucket.

When implemented, AWS Lambda creates an event and pushes the newly discovered file to Databricks. By using a constant connection, this promotes a steady flow of data between S3 and Databricks. Databricks is capable of connecting to these services via API to manage the various data streams. Through a seamless AWS to Databricks connection, businesses are able to take advantage of Spark-based processing and Databricks' distributed computing capabilities to quickly process and analyze continuous amounts of data.

The architecture of the data pipeline (*Diagram 1*) is made simpler by Databricks' seamless integration with AWS S3 and Lambda, which facilitates effective data management. It gives businesses a reliable method of handling data streams, scale, data ingestion and processing to meet an organization's demands. Although we could not implement these features with the community edition of Databricks, further research into Databricks capabilities has shown how combining AWS S3, and Lambda provides a complete and scalable solution for managing an ongoing flow of data.

*(Diagram 1 / S3 Integration)*



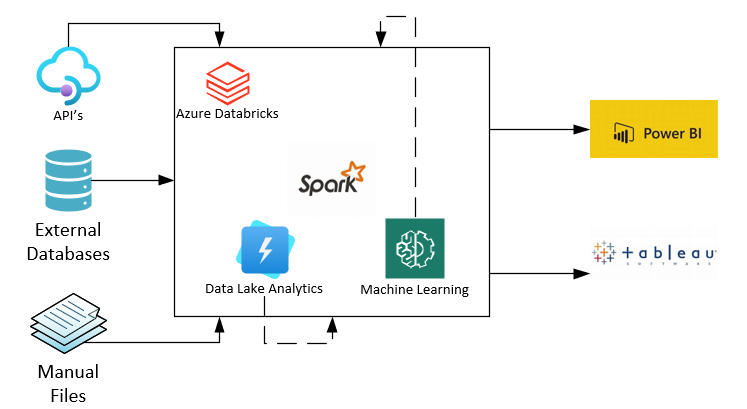
### *Enterprise Edition (Diagram 2):*

**Ingestion:** Implementing different sources in Databricks' enterprise edition is a simple procedure. The platform's data ingesting features can be used by users to easily ingest data from a variety of sources, including APIs, external databases, and manual files. With the help of Databricks' connectors and interfaces with well-known data sources, it is simple to extract data and add it to the platform. This makes it possible for businesses to efficiently gather data from many sources and get it ready for processing and analysis.

**Within Databricks:** An all inclusive SASS for data processing and analysis is provided by the Databricks environment. Through Apache Spark, it offers a scalable and distributed computing solution enabling customers to carry out intricate computations and dataset changes. Databricks speeds up data processing and supports a variety of data analytics jobs with Spark's in-memory processing capabilities. Graph processing, machine learning, and SQL querying are just a few of the analytics features offered by Datalake Analytics, which is built on top of Spark.

**Visualization:** Databricks provides users with powerful visualization tools to aid in understanding and conveying their findings. Users can explore and evaluate data with interactive visualizations' charts, graphs, and plots. Users can build dynamic visual representations of their data using the platform's support for a number of visualization tools, such as Matplotlib, Seaborn, and Plotly. Users of Databricks can also build interactive dashboards that mix many visualizations into a single view for thorough data display. Databricks' also supports geographic visualization and enables users to plot and examine data on maps, allowing for spatial analysis and insights from location-based data. Additionally, Databricks integrates with well-known business intelligence (BI) tools like Power BI and Tableau, allowing for the integration of advanced visualization features and reports.

*(Diagram 2 / High Level Architecture Enterprise Edition)*



### *Code / Input:*

**Data Ingestion (Python):**

*(Data-In 1):*

# File location and type

file\_location = "/FileStore/tables/Hackathon\_Ideal\_Data.csv"

file\_type = "csv"

# CSV options

infer\_schema = "false"

first\_row\_is\_header = "true"

delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.

df = spark.read.format(file\_type) \

.option("inferSchema", infer\_schema) \

.option("header", first\_row\_is\_header) \

.option("sep", delimiter) \

.load(file\_location)

display(df)

**Data Pre-Processing:**

(*Pre-Pro 1):*

%sql

--Checking all datatypes are correct

describe sales\_data\_csv

(*Pre-Pro 2):*

%sql

/\* Note that 'qty' and 'value' are of the wrong data type. This will be addressed.

Now checking for 0 values in 'qty' and 'value'\*/

SELECT count(\*)

FROM sales\_data\_csv

WHERE qty <> 0 or value <> 0;

*(Pre-Pro 3):*

%sql

--Creating a new table with all rows except where there was a value of 0 in 'qty' or 'value'

CREATE OR REPLACE TABLE sales\_data\_3 AS

SELECT \*

FROM sales\_data\_csv

WHERE qty <> 0 or value <> 0;

**Data Analysis:**

*(Analysis 1)*:

%sql

SELECT

COUNT(qty\_num) AS count,

MIN(qty\_num) AS min\_value,

MAX(qty\_num) AS max\_value,

AVG(qty\_num) AS mean,

SUM(qty\_num) AS sum,

STDDEV(qty\_num) AS standard\_deviation,

PERCENTILE\_CONT(0.25) WITHIN GROUP (ORDER BY qty\_num) AS percentile\_25,

PERCENTILE\_CONT(0.50) WITHIN GROUP (ORDER BY qty\_num) AS median,

PERCENTILE\_CONT(0.75) WITHIN GROUP (ORDER BY qty\_num) AS percentile\_75

FROM

sales\_data\_3;

*(Analysis 2*):

%sql

SELECT

COUNT(value\_num) AS count,

MIN(value\_num) AS min\_value,

MAX(value\_num) AS max\_value,

AVG(value\_num) AS mean,

SUM(value\_num) AS sum,

STDDEV(value\_num) AS standard\_deviation,

PERCENTILE\_CONT(0.25) WITHIN GROUP (ORDER BY value\_num) AS percentile\_25,

PERCENTILE\_CONT(0.50) WITHIN GROUP (ORDER BY value\_num) AS median,

PERCENTILE\_CONT(0.75) WITHIN GROUP (ORDER BY value\_num) AS percentile\_75

FROM

sales\_data\_3;

*(Analysis 3*):

%sql

SELECT

COUNT(unit\_value) AS count,

MIN(unit\_value) AS min\_value,

MAX(unit\_value) AS max\_value,

AVG(unit\_value) AS mean,

SUM(unit\_value) AS sum,

STDDEV(unit\_value) AS standard\_deviation,

PERCENTILE\_CONT(0.25) WITHIN GROUP (ORDER BY unit\_value) AS percentile\_25,

PERCENTILE\_CONT(0.50) WITHIN GROUP (ORDER BY unit\_value) AS median,

PERCENTILE\_CONT(0.75) WITHIN GROUP (ORDER BY unit\_value) AS percentile\_75

FROM

sales\_data\_3;

*(Analysis 4*):

%sql

--Obtaining number of Unique Values and Unique Items in Total using a combination of columns as an identifier.

SELECT

a.Unique\_Months,

b.Unique\_Stores,

c.Unique\_Grp,

d.Unique\_Sgrp,

e.Unique\_Ssgrp,

f.Unique\_Cmp,

g.Unique\_Mbrd,

h.Unique\_Brd,

i.Unique\_Items

FROM

(SELECT COUNT(DISTINCT MONTH) as Unique\_Months FROM sales\_data\_3 as Unique\_Months) a,

(SELECT COUNT(DISTINCT STORECODE) as Unique\_Stores FROM sales\_data\_3 as Unique\_Stores) b,

(SELECT COUNT(DISTINCT GRP) as Unique\_Grp FROM sales\_data\_3 as Unique\_Grp) c,

(SELECT COUNT(DISTINCT SGRP) as Unique\_Sgrp FROM sales\_data\_3 as Unique\_Sgrp) d,

(SELECT COUNT(DISTINCT SSGRP) as Unique\_Ssgrp FROM sales\_data\_3 as Unique\_Ssgrp) e,

(SELECT COUNT(DISTINCT CMP) as Unique\_Cmp FROM sales\_data\_3 as Unique\_Cmp) f,

(SELECT COUNT(DISTINCT MBRD) as Unique\_Mbrd FROM sales\_data\_3 as Unique\_Mbrd) g,

(SELECT COUNT(DISTINCT BRD) as Unique\_Brd FROM sales\_data\_3 as Unique\_Brd) h,

(select count(DISTINCT concat(grp,'/',sgrp,'/',ssgrp,'/',cmp,'/',mbrd,'/',brd)) as Unique\_Items from sales\_data\_3 as Unique\_Items) i

*(Analysis 5*):

%sql

-- calculate the total quantities and revenue sold for each product at each store in the product\_sales table.

-- We then rank these sales within each store in the ranked\_sales table. The product with the most quantity sold at a store will have a rank of 1.

-- We also calculate the total revenue for each store in the store\_revenue table.

-- In the final query, we join the store total revenue with the best-selling categories and their total revenue from the ranked\_sales table. We then calculate the percentage of total revenue that the best selling category accounts for.

WITH product\_sales AS (

SELECT

STORECODE,

grp,

SUM(QTY\_num) as TotalQuantitySold,

SUM(VALUE\_num) as TotalCategoryRevenue

FROM sales\_data\_3

GROUP BY STORECODE, grp

),

ranked\_sales AS (

SELECT

STORECODE,

grp,

TotalQuantitySold,

TotalCategoryRevenue,

ROW\_NUMBER() OVER (PARTITION BY STORECODE ORDER BY TotalQuantitySold DESC) as rank

FROM product\_sales

),

store\_revenue AS (

SELECT

STORECODE,

SUM(VALUE\_num) as TotalRevenue

FROM sales\_data\_3

GROUP BY STORECODE

)

SELECT

store\_revenue.STORECODE,

store\_revenue.TotalRevenue,

best\_selling\_product.grp as BestSellingCategory,

best\_selling\_product.TotalCategoryRevenue,

round((best\_selling\_product.TotalCategoryRevenue / store\_revenue.TotalRevenue) \* 100, 2) as PercentageRevenueBestSellingCategory

FROM store\_revenue

LEFT JOIN ranked\_sales as best\_selling\_product

ON store\_revenue.STORECODE = best\_selling\_product.STORECODE AND best\_selling\_product.rank = 1

ORDER BY TotalRevenue DESC

*(Analysis 6*):

%sql

-- average quantity and quantity standard deviation rounded to two decimal places.

WITH store\_sales AS (

SELECT

STORECODE,

BRD,

SUM(qty\_num) AS TotalQuantity

FROM sales\_data\_3

GROUP BY STORECODE, BRD

),

store\_sales\_stats AS (

SELECT

STORECODE,

ROUND(AVG(TotalQuantity), 2) AS AverageQuantity,

ROUND(STDDEV(TotalQuantity), 2) AS QuantityStdDev

FROM store\_sales

GROUP BY STORECODE

)

SELECT

sales\_data\_3.STORECODE,

sales\_data\_3.TotalSales,

stats.AverageQuantity,

stats.QuantityStdDev

FROM (

SELECT

STORECODE,

SUM(qty\_num) AS TotalSales

FROM sales\_data\_3

GROUP BY STORECODE

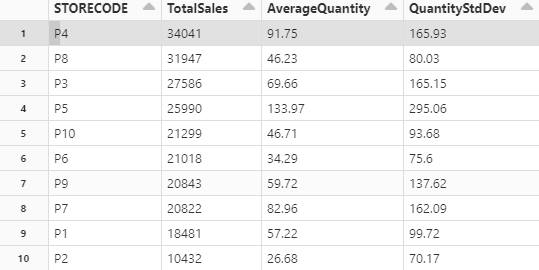
) sales\_data\_3

LEFT JOIN store\_sales\_stats stats ON sales\_data\_3.STORECODE = stats.STORECODE

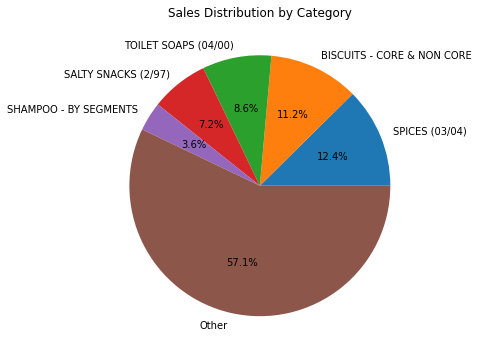
ORDER BY sales\_data\_3.TotalSales DESC

### *Output*:

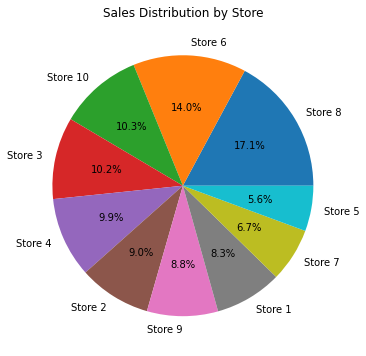
*(Output 1):*



*(Output 2):*



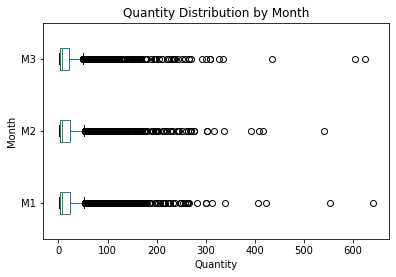
*(Output 3):*



*(Output 4):*

### 

*(Output 5):*



### 

## ***References***

1. *What is Apache Parquet?* (n.d.). Databricks. <https://www.databricks.com/glossary/what-is-parquet#:~:text=What%20is%20Parquet%3F>
2. *Data Lakehouse Architecture and AI Company*. Databricks. (n.d.). https://www.databricks.com/
3. *Databricks architecture overview*. (n.d.). Databricks Architecture Overview | Databricks on AWS. <https://docs.databricks.com>