Coffee Shop Sales Analysis (Power BI)

Usama Bin Hafeez Abbasi

Production Analyst, Afiniti

Atom Camp

usamahafeez.abbasi1234@gmail.com

Introduction

This report presents a comprehensive analysis of Coffee Shop Sales data, performed as part of a task assigned by Atom Camp. The objective of this analysis was to transform, normalize, and analyze the dataset to derive actionable insights. The tasks included data cleaning, normalization, relationship building, and advanced calculations using Power BI and Power Query.

Key Objectives

- Normalize the dataset into structured tables (Transactions, Stores, Products).
- Perform data transformations, including creating calculated columns and parameters.
- Analyze sales performance, product distribution, and transaction trends.
- Visualize insights to support data-driven decision-making.

Tools Used

- Power BI: For data modeling, analysis, and visualization.
- Power Query: For data transformation and cleaning.

Report Overview

This report includes the following sections:

- Data Preparation: Normalization and cleaning of the dataset.
- Advanced Analysis: Calculations, filtering, and aggregation.
- Insights and Visualizations: Key findings and visual representations of the data.

LOAD COFFEE SHOP DATA INTO POWER BI

The first step involved loading the provided Coffee Shop Sales dataset into Power BI. The data was sourced from an Excel file, which was seamlessly uploaded into Power BI Desktop. Upon loading, the dataset was opened in Power Query Editor to initiate the data preparation process. This step ensures that the raw data is accessible and ready for further transformation and analysis within the Power BI environment. The process was straightforward, adhering to best practices for data ingestion, and sets the foundation for subsequent data modeling and visualization tasks.

DATA TRANSFORMATION

The second step involved a thorough validation of the dataset to ensure data quality and consistency. This included reviewing column names to confirm their accuracy, verifying data types for each column to ensure they align with the nature of the data (e.g., numeric, text, date), and checking for any missing or error values. After a manual inspection, no issues were identified. All column names were clear and relevant, data types were correctly assigned, and there were no missing or erroneous entries. This validation confirms the dataset is clean and ready for further transformation and analysis.

NORMALIZATION OF DATA AND DUPLICATE REMOVAL

The third step involved normalizing the dataset by splitting it into smaller, more focused tables. The main table was duplicated in Power Query Editor, and irrelevant columns were removed to create three distinct tables: **Transactions**, **Stores**, and **Products**.

- The **Transactions** table includes: transaction_id, transaction_date, transaction_time, transaction_qty, store_id, and product_id.
- The **Stores** table includes: store_id and store_location.
- The **Products** table includes: product_id, unit_price, product_category, product_type, and product_detail.

Additionally, duplicates were checked and removed to ensure data integrity. No duplicates were found in the **Transactions** table, while duplicates were identified and removed from the **Products** and **Stores** tables. This normalization process ensures a clean, efficient, and well-structured dataset, ready for analysis and relationship building in Power BI.

Data Modeling

The **Transactions** table was identified as the **Fact Table**, containing measurable transactional data, while the **Stores** and **Products** tables were identified as **Dimension Tables**, providing descriptive context for the analysis.

Relationships were established between the tables to enable seamless data analysis. A **one-to-many** relationship was created between the **Products** table and the **Transactions** table, as well as

between the **Stores** table and the **Transactions** table. The cross-filter direction was set to **Both** to ensure bidirectional filtering, allowing for comprehensive analysis across related tables.

The data model was identified as a **Star Schema**, with the **Transactions** table serving as the central fact table and the **Products** and **Stores** tables acting as dimension tables.

POWER QUERY ANALYSIS

CREATE A SALES COLUMN

To calculate sales, the **Unit Price** column from the **Products** table was merged into the **Transactions** table using the **Product ID** as the join key. After merging, the **Unit Price** column was extracted from the nested table using the filter option. A new custom column, **Sales**, was then created using the formula:

Sales = [Unit Price] * [transaction_qty]

This step ensures accurate sales calculations by combining transactional data with product pricing, enabling detailed revenue analysis in Power BI.

Create a Conditional Column

A conditional column, **Is High Quantity**, was added to the **Transactions** table. The logic applied was:

if transaction_qty > 4, return "Yes" otherwise "No"

This was achieved using the **Conditional Column** option in Power Query, enabling quick categorization of high-quantity transactions.

Calculate and Store Parameters

Two key parameters were calculated and stored

- **Total Sales**: Sum of the **Sales** column.
- **Average Transaction Quantity**: Average of the transaction_qty column.

These calculations were performed using the **Statistics** option in Power BI and saved as parameters for further analysis.

Filter Based on Parameters

A duplicate of the **Transactions** table was created, and transactions with a transaction_qty greater than the **Average Transaction Quantity** parameter were filtered. This step helps identify above-average transactions for deeper analysis.

Sales Based on Location

The **Sales** column from the **Transactions** table was merged into the **Stores** table using **Store ID** as the join key. The aggregated value, **Sum of Sales**, was calculated to analyze sales performance by store location.

Count of Products in Each Category

A duplicate of the **Products** table was created, and the **Group By** operation was applied to count products in each product_category. The resulting table was renamed as **Product Summary**, providing a clear overview of product distribution by category.