**Technical Report for Grocery Sales Datawarehouse and Visualization**



**Piyush Aughad (20040022)**

**Damodharan Arokyam (20017736)**

**Rutuja Bhatankar(20035261)**

Group Project (CA\_ONE): DATA STORAGE SOLUTIONS

Supervisor - Bernie Lydon

submitted for the degree of

M.Sc. in Data Analytics

at Dublin Business School

April 2025

Contents

[1. Introduction 2](#_Toc37316268)

[1.2. Reasons for selecting the subject area AND DATA 2](#_Toc37316269)

[1.3. Vision and Goals 3](#_Toc37316270)

[1.4. Key StakeHolders` 3](#_Toc37316271)

[1.5. Business requirements 3](#_Toc37316272)

[1.6. Any other sections 4](#_Toc37316273)

[2. SCHEMA 4](#_Toc37316274)

[3. ETL 7](#_Toc37316275)

[4. VISUALIZATIONS AND REPORTS 10](#_Toc37316276)

[4.1. Visualizations 11](#_Toc37316277)

[4.2. Reports 14](#_Toc37316278)

[5. Graph Databases 18](#_Toc37316279)

[5.1. COMAPRISON to realtional databases 22](#_Toc37316280)

[6. Conclusions 25](#_Toc37316281)

[7. Bibliography 26](#_Toc37316282)

[Appendix B – Neo 4J code 26](#_Toc37316284)

# 1. Introduction

In an increasingly competitive retail landscape, leveraging data to drive business decisions is essential for sustained growth and market leadership. This report presents an analytical framework centered around integrating and analyzing diverse datasets, including products, categories, customers, employees, and geographic locations. The primary objective is to build a centralized data warehouse that consolidates key business data, enabling stakeholders to gain valuable insights into performance, trends, and customer behavior.

The AdventureWorks dataset is a comprehensive and realistic sample database developed by Microsoft to simulate the operations of a fictitious bicycle manufacturing company. It includes detailed business data across various departments such as sales, purchasing, manufacturing, human resources, and finance. This dataset is widely used in academic, professional, and training environments to practice and demonstrate relational database design, querying, and reporting.

In this project, the AdventureWorks dataset is utilized to explore and compare two different data models and query languages: SQL used in relational database systems like Microsoft SQL Server, and CQL used in distributed NoSQL databases. While SQL supports normalized schemas and complex relationships, CQL emphasizes scalability and high availability through denormalized, query-optimized designs.

By working with the AdventureWorks dataset in both SQL and CQL environments, this project aims to provide hands-on experience in multi-model database design and demonstrate the trade-offs between traditional and modern data management approaches.

**Dataset link for Warehouse:**  <https://www.kaggle.com/datasets/andrexibiza/grocery-sales-dataset?select=cities.csv>

**Dataset link for SQL and CQL:** https://www.kaggle.com/datasets/dorcasdavid/adventureworks-sales

## 1.2. Reasons for selecting the subject area AND DATA

The selection of this subject area stems from the growing need for data-driven decision-making in the retail industry, particularly within the grocery sector where product variety, pricing, customer preferences, and regional demand fluctuate frequently. By focusing on sales, product performance, customer demographics, and employee efficiency, the project addresses core aspects that directly impact business profitability and customer satisfaction.

* Product Performance: The products.csv and categories.csv provide rich data on product attributes, prices, and categories, enabling analysis of profitability, pricing trends, and inventory management.
* Customer Demographics: The customers.csv, cities.csv, and countries.csv allow geographic analysis of customer distribution, which is critical for targeted marketing and logistics planning.

## 1.3. Vision and Goals

Build a centralized data warehouse that integrates sales, product, customer, employee, location, and calendar data. This will empower stakeholders with the insights needed to help the company become a leading grocery retailer by offering the right products based on sales trends, customer preferences, and regional demand. It will also support competitive pricing through strategic discounts and ensure a diverse range of dietary options, including choices for allergen-sensitive customers.

## 1.4. Key StakeHolders

* **Sales Managers:** Track product performance and sales trends.
* **Marketing Teams:** Segment customers by location and demographics.
* **Supply Chain Managers:** Optimize inventory based on product categories.
* **HR Departments:** Analyse employee distribution and hiring patterns.

## 1.5. Business requirements

**Product Category Performance:**

* Which product categories have the highest and lowest growth rates (0% to 20%), and what factors contribute to these trends?
* Are there seasonal or regional patterns affecting sales in categories like Beverages, Dairy, or Seafood?

**Discount & Pricing Strategy:**

* How do discounts (10% vs. 20%) impact sales volume and profitability in different product categories?
* How do these monthly trends compare to previous years?

**Geographic Sales Distribution:**

* Which countries or areas are making the most sales, and are there any new markets we could expand into?
* Do certain product categories perform better in specific locations (e.g., seafood in coastal regions)?

**Allergen-Related Products:**

* What is the revenue contribution of allergen labelled products, and is there demand for allergen-free alternatives?

**Employee Performance:**

* Which employees have the highest and lowest average sales, and what strategies do top performers use (e.g., upselling, customer retention)?

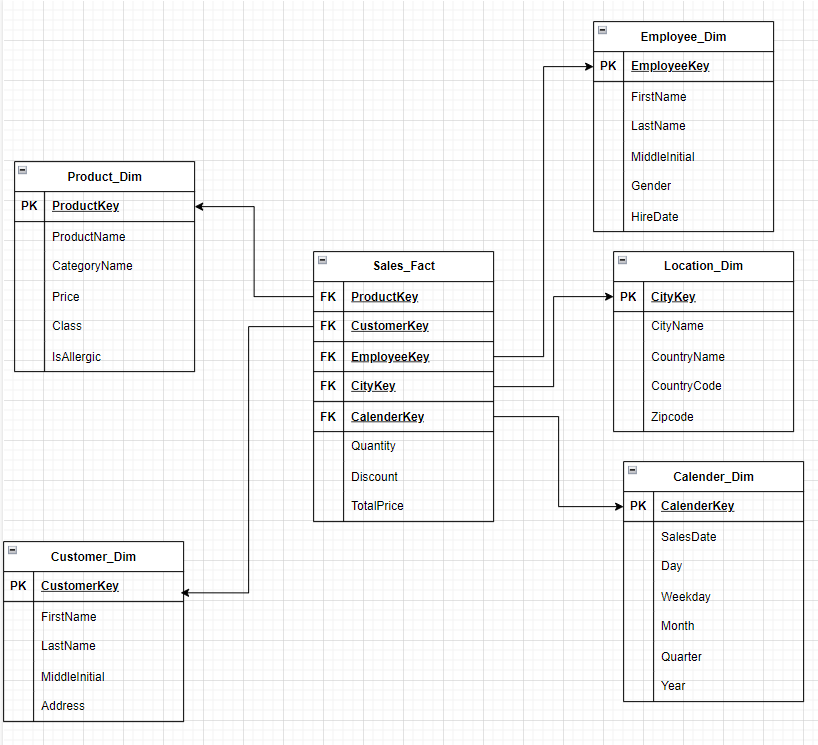
## 1.6. Any Other Sections

# 2. SCHEMA

Normalization helps us find a database schema at design time that can process the most frequent updates efficiently at run time. Unfortunately, relational normalization only works for idealized database instances in which duplicates and null markers are not present. On one hand, these features occur frequently in real-world data compliant with the industry standard SQL, and especially in modern application domains. (Köhler and Link, 2016)

Traditional methods of analyzing customer behaviour are proving inadequate in the face of these challenges, often failing to process the vast volumes of transaction data efficiently or identify complex product relationships accurately. Association rule mining and market basket analysis have proven to be essential tools for retailers seeking deeper insights into customer purchasing patterns and product relationships.(Farheen and Dharani, 2024)

* For easy to fetch the valuable data for business insight we have design a Data Warehouse using technique a star schema method and, in the warehouse, we have 5 dim (Product, Employee, Location, Customer and Calander) and have 3 attributes (Quantity, Discount and Total price) in fact tables.



*Fig – 1: Star Schema*

**Star Schema Code:**

CREATE DATABASE Sales\_DW;

GO

USE Sales\_DW;

GO

CREATE TABLE Product\_Dim (

ProductKey INT NOT NULL IDENTITY,

ProductName VARCHAR(50) NOT NULL,

CategoryName VARCHAR(50),

Price DECIMAL(20,2) NOT NULL,

Class VARCHAR(50),

IsAllergic VARCHAR(50),

PRIMARY KEY (ProductKey));

GO

CREATE TABLE Customer\_Dim (

CustomerKey INT NOT NULL IDENTITY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

MiddleInitial CHAR(5),

Address VARCHAR(50)

PRIMARY KEY (CustomerKey));

GO

CREATE TABLE Employee\_Dim (

EmployeeKey INT NOT NULL IDENTITY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

MiddleInitial VARCHAR(5),

Gender VARCHAR(10),

HireDate DATETIME

PRIMARY KEY (EmployeeKey));

GO

CREATE TABLE Location\_Dim (

CityKey INT NOT NULL IDENTITY,

CityName VARCHAR(50) NOT NULL,

CountryName VARCHAR(50),

CountryCode VARCHAR(10),

Zipcode VARCHAR(20)

PRIMARY KEY (CityKey));

GO

CREATE TABLE Calender\_Dim (

CalenderKey INT NOT NULL IDENTITY,

SalesDate DATETIME NOT NULL,

Day INT NOT NULL,

Weekday VARCHAR(10),

Month char(10) NOT NULL,

Quarter char(2) NOT NULL,

Year INT NOT NULL

PRIMARY KEY (CalenderKey));

GO

CREATE TABLE Sales\_Fact (

ProductKey INT NOT NULL,

CustomerKey INT NOT NULL,

EmployeeKey INT NOT NULL,

CityKey INT NOT NULL,

CalenderKey INT NOT NULL,

Quantity INT NOT NULL,

Discount DECIMAL(20,1) DEFAULT 0,

TotalPrice DECIMAL(20,2) NOT NULL,

PRIMARY KEY (ProductKey,CalenderKey),

FOREIGN KEY (ProductKey) REFERENCES Product\_Dim(ProductKey),

FOREIGN KEY (CustomerKey) REFERENCES Customer\_Dim(CustomerKey),

FOREIGN KEY (EmployeeKey) REFERENCES Employee\_Dim(EmployeeKey),

FOREIGN KEY (CityKey) REFERENCES Location\_Dim(CityKey),

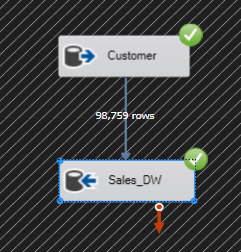
FOREIGN KEY (CalenderKey) REFERENCES Calender\_Dim(CalenderKey)

);

# 3. ETL

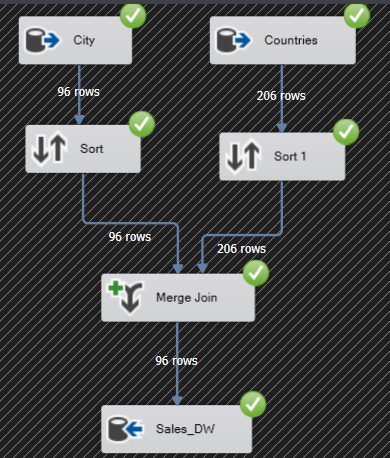
We have started ETL process and dataflow for each dimension and fact tables with SSIS using the help of visual studio 2022 tool.

* **Customer\_Dim:** Load the 98,759 customer records into the Customer\_Dim table in the data warehouse.

****

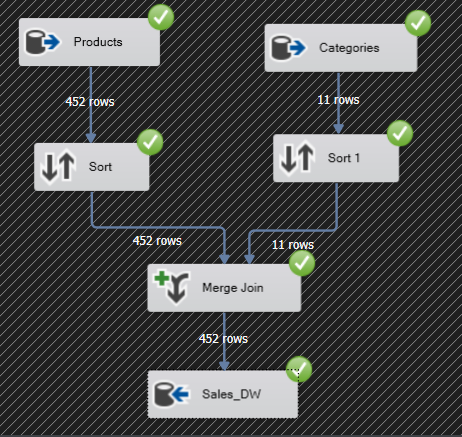
*Fig – 2: Customer Dimension*

* **Location\_Dim:** We have 96 city records and 206 county records, all originating from the USA. The ETL process has been completed for both city and county data, and all 96 city records, along with the associated county information, have been loaded into the **Location\_Dim** table in the data warehouse.

****

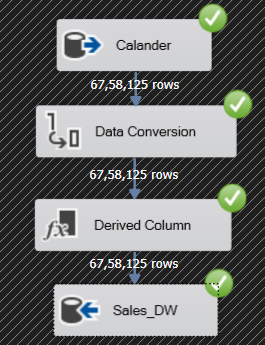
*Fig – 3: Location Dimension*

* **Product\_Dim:** We have 452 product records and 11 category records. The ETL process for both datasets has been completed, and all product and category data has been successfully loaded into the Product\_Dim table in the data warehouse.

****

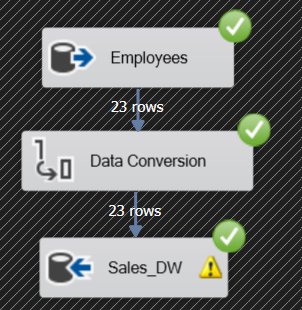
*Fig – 4: Product Dimension*

* **Calander\_Dim:** We extracted the sales dates into a calendar structure, creating separate fields for date, weekday, month, year, and quarter. The ETL process has been completed, and the dataset containing 6,758,125 records has been successfully loaded into the data warehouse.

****

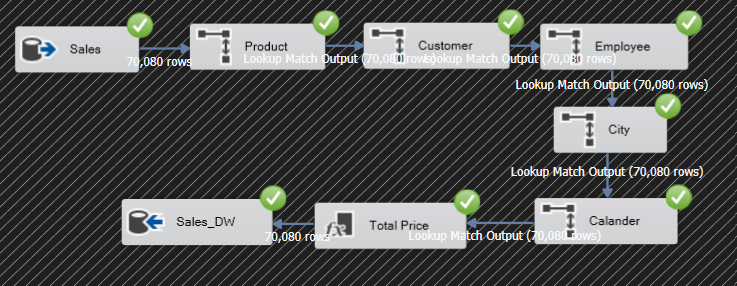
*Fig – 5: Calender Dimension*

* **Employee\_Dim:** We have a total of 23 employee records. The ETL process has been completed, and fields such as first name, middle name, last name, gender, and hire date have been extracted and loaded into the data warehouse.

****

*Fig – 6: Employee Dimension*

* **Sales\_Fact:** The structure below represents the ETL process for the fact table. Keys were retrieved using IDs, and additional fields such as Quantity, Discount, and Total Price were included. The primary keys for the Sales\_Fact table are ProductKey and CalendarKey. A total of 70,080 records have been loaded into the Sales\_Fact table in the data warehouse.

****

*Fig – 7: Sales Fact*

# 4. VISUALIZATIONS AND REPORTS

In today’s data-centric retail landscape, having immediate access to visualized insights enables faster decision-making and cross-functional alignment. This dashboard bridges the gap between raw data and strategic business thinking by offering flexible, user-friendly access to critical sales metrics. By reducing the time spent on manual data aggregation and static reporting, the dashboard supports a more agile and proactive approach to performance management.

This report presents a strategic analysis of the Grocery Sales Report Dashboard built using Tableau. The dashboard provides a powerful platform for visualizing and interacting with sales data across products, employees, time, and geography. With the ability to apply dynamic parameters and filters, users gain granular insights tailored to specific business questions. The following analysis highlights core findings and demonstrates how interactivity enhances decision-making.

The design of this dashboard also aims to reduce cognitive overload for users. Instead of sifting through complex spreadsheets or disconnected reports, users can intuitively navigate a visual environment. Decision-making becomes less reactive and more deliberate, grounded in evidence and patterns that emerge from interactive exploration.

The objective behind developing this dashboard is to equip decision-makers with actionable insights into sales performance. By integrating data from multiple sources and dimensions, the dashboard delivers:

- Evaluation of top-selling products and categories

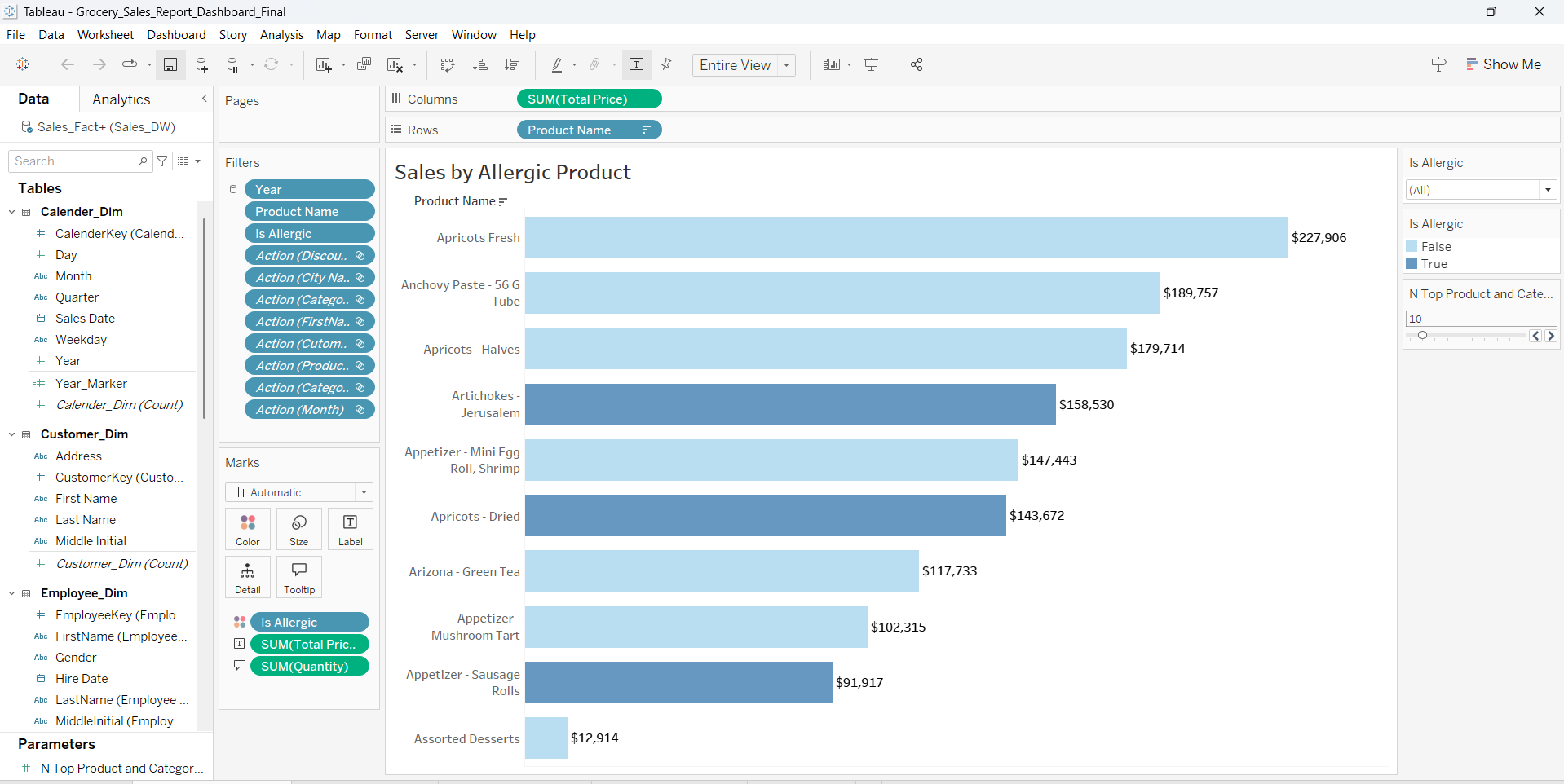
- Monthly and seasonal sales patterns

- Performance metrics across employees and locations

- Analytical segmentation using customer behavior and product attributes

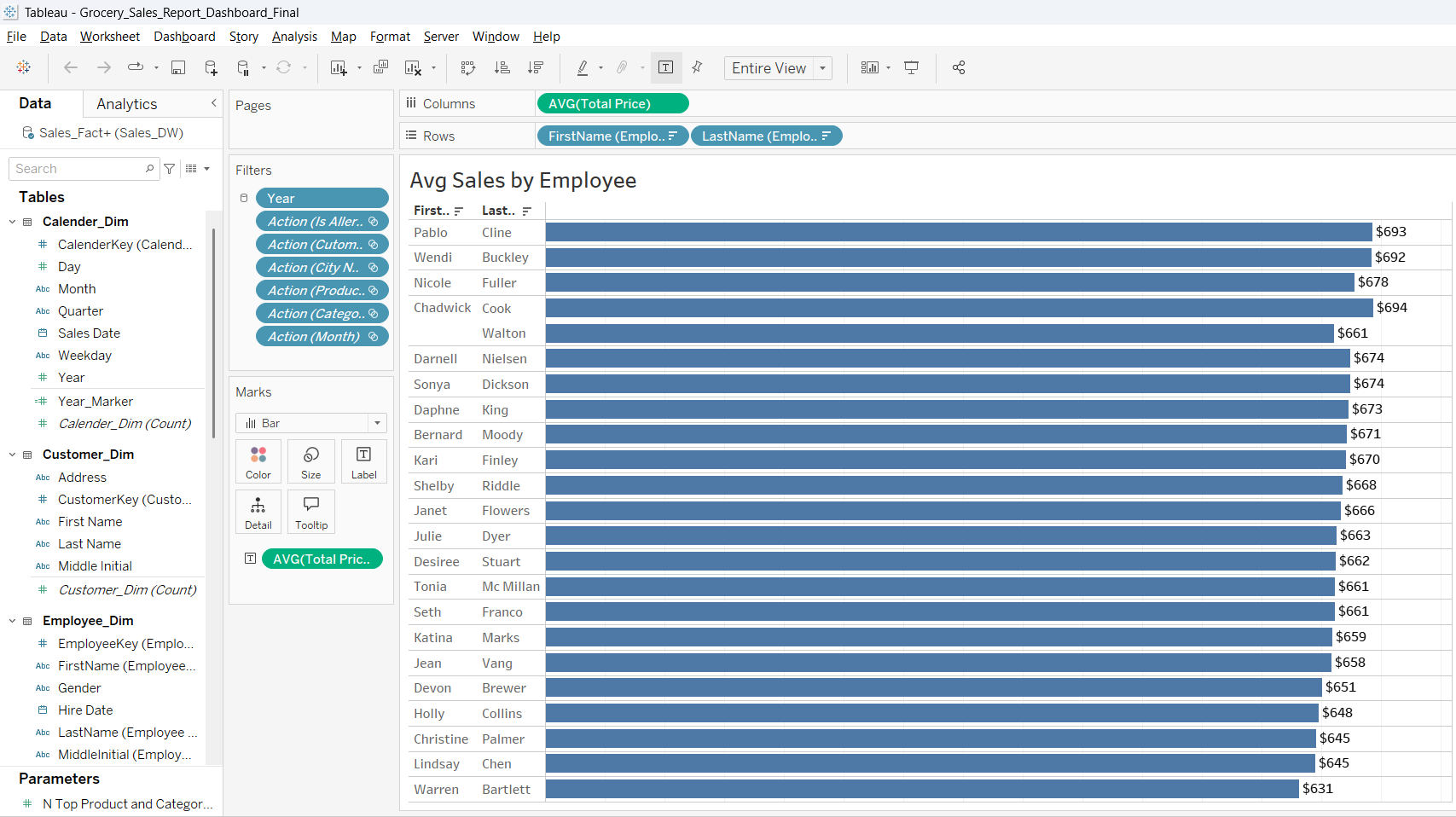
## 4.1. Visualizations

1. **Sales by Allergic Product:** Allergic items such as 'Apricots Fresh' and 'Anchovy Paste' top the chart, indicating the importance of allergen labeling and shelf placement strategy. The parameter 'Top N Categories' enables zooming into these products specifically, while filters like 'Is Allergic' refine the analysis further.



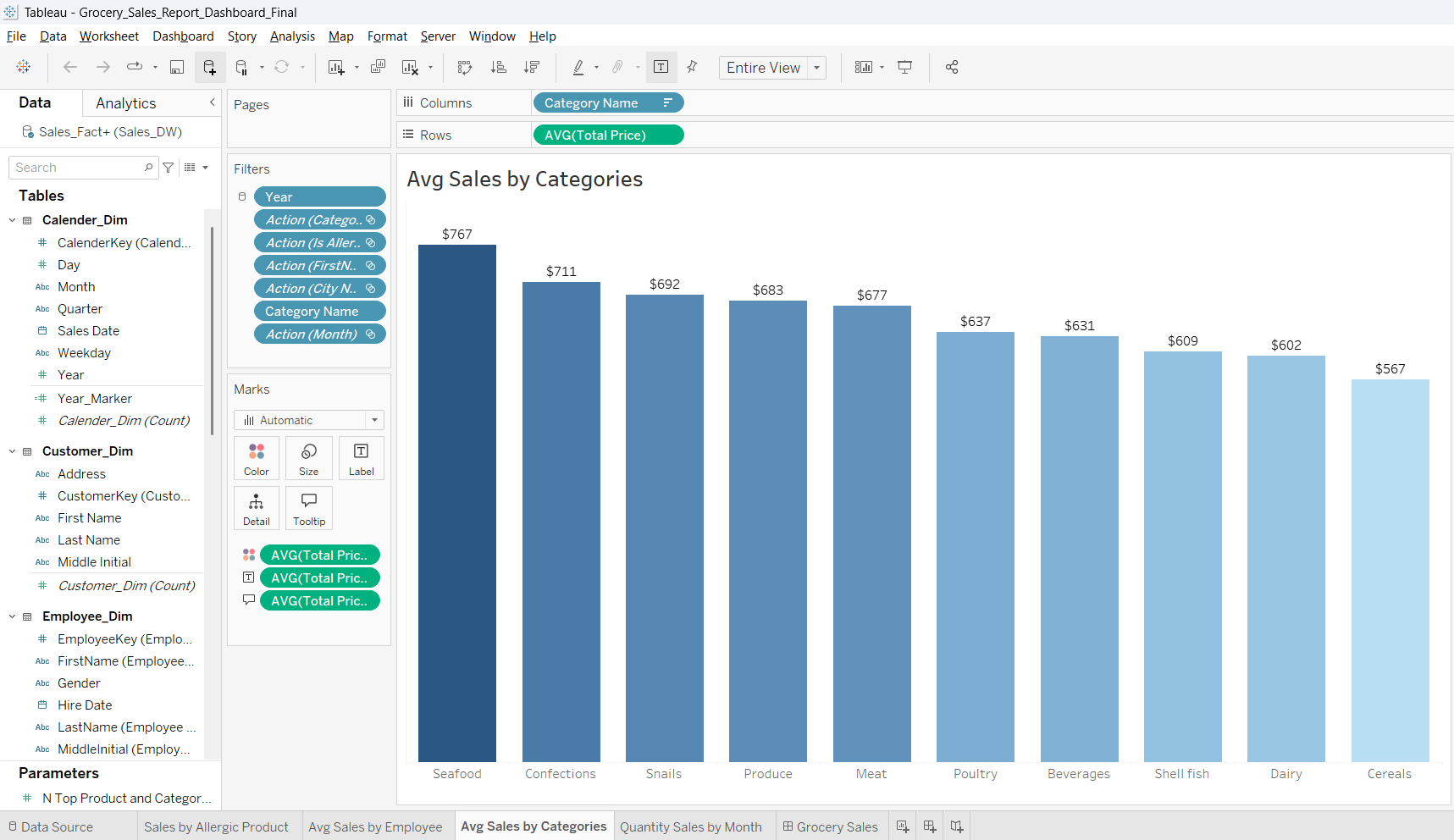
*Fig – 8: Sales by Allergic Product*

1. **Avg Sales by Employee:** With names filtered using 'FirstName, LastName' fields, the chart highlights standout performers such as Pablo and Wendi. This metric could inform training plans or bonus incentives, especially when filtered by time to understand consistency.

****

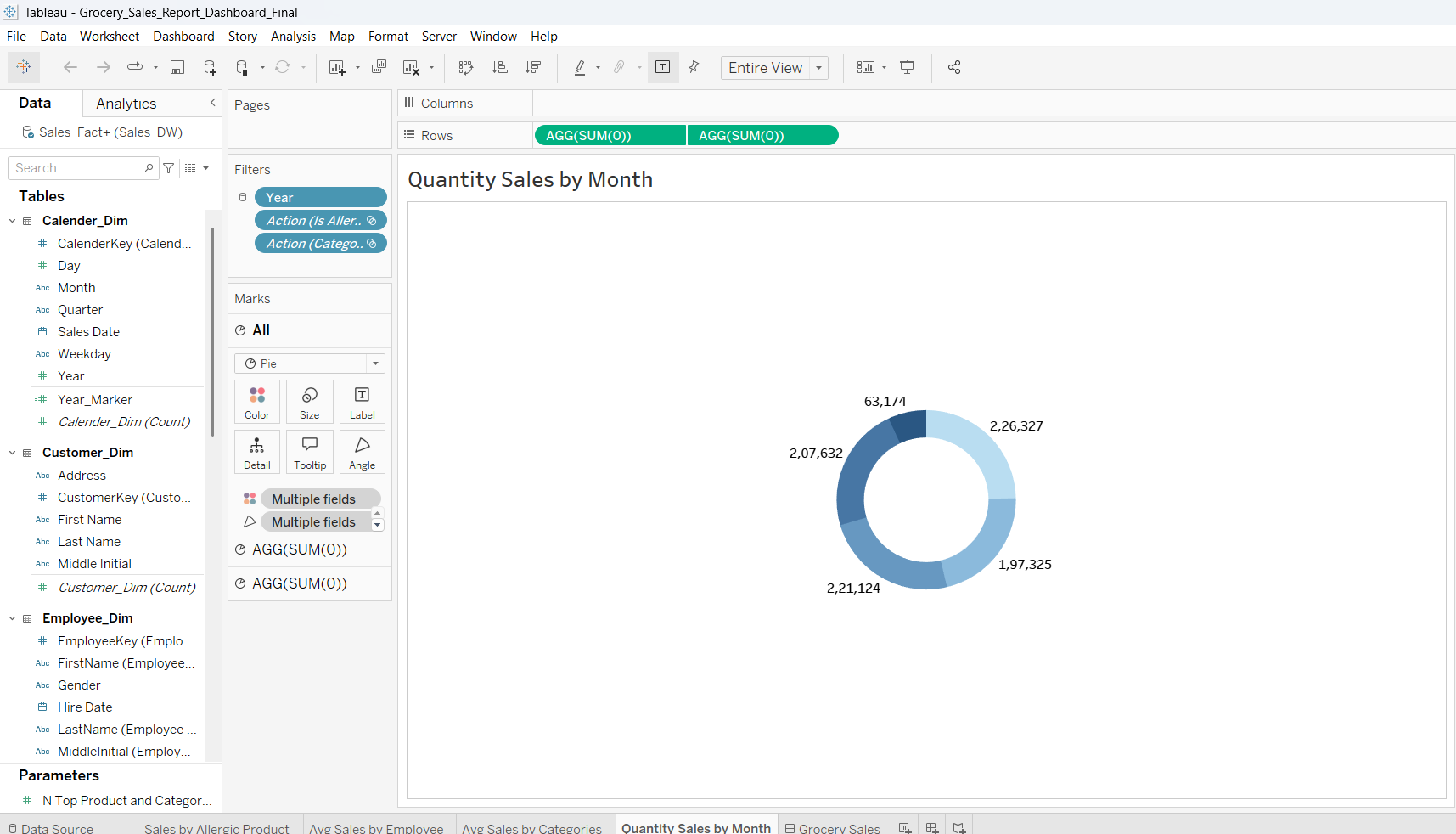
*Fig – 9: Avg Sales by Employee*

1. **Avg Sales by Categories:** Seafood and Confections lead in average sale value, while cereals lag behind. Category filters allow managers to break this down by geography or promotion type. This helps in identifying which product lines deliver the highest returns per sale. Importantly, these recommendations are not prescriptive but adaptive. Each can be revisited as the business environment evolves. For instance, a product that currently performs poorly might gain momentum due to seasonality or a marketing campaign. In that case, real-time filter usage and parameter updates will ensure insights stay aligned with live conditions.

****

*Fig – 10: Avg Sales by Categories*

1. **Quantity Sales by Month:** This donut chart displays monthly variations in total quantity retailed Ultimately, the value of a dashboard lies not only in what it displays but in how it empowers people. This tool democratizes data access, allowing users across roles from sales associates to regional managers to become insight-driven. The stronger the alignment between data strategy and user behavior, the greater the competitive edge.

****

*Fig – 11: Quantity Sales by Month*

**Final Integrated Dashboard Overview:** The following visualization presents the final integrated dashboard, combining key analytical elements from across the project. This dashboard is a comprehensive interface designed to allow dynamic exploration of sales data using the implemented filters and parameters

A screenshot of a computer

AI-generated content may be incorrect.

*Fig – 12: Dashboard Overview*

## 4.2. Reports

1. **Sales for Allergic product**



*Fig – 13: Sales for Allergic product*

The provided data outlines the sales performance of products categorized by their allergic status: False (non-allergic), True (allergic), and Unknown (allergic status not specified). The analysis reveals significant differences in both sales volume and revenue across these categories.

* **Non-Allergic Products (False):**

Non-allergic products demonstrated the highest performance in both sales volume and revenue. A total of 334,758 units were sold, generating $18,288,515.62 in revenue. This category's strong performance suggests that non-allergic products are the most popular or widely available, contributing significantly to the overall sales.

* **Allergic Products (True):**

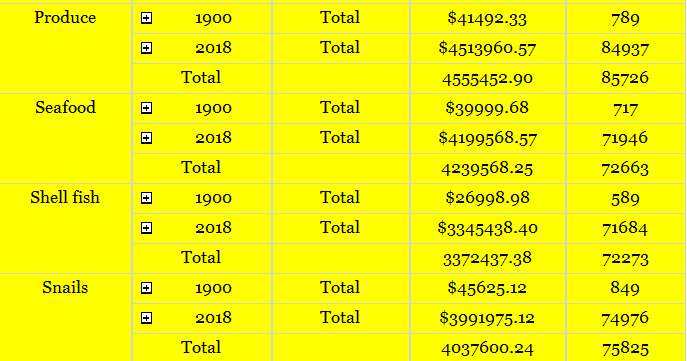
Allergic products followed closely in sales volume, with 320,078 units sold, but generated lower revenue at $15,866,361.53. The slightly lower revenue compared to non-allergic products may indicate pricing differences, such as allergic products being sold at a discount or having a lower average price per unit.

* **Products with Unknown Allergic Status (Unknown):**

Products with an unspecified allergic status had the lowest sales volume and revenue, with 260,746 units sold and $12,441,582.00 in revenue. This could reflect limited consumer interest or a lack of clear information about these products, potentially deterring purchases.

1. **Sales Performance Report by Product Category**





*Fig – 14: Sales by Product Category*

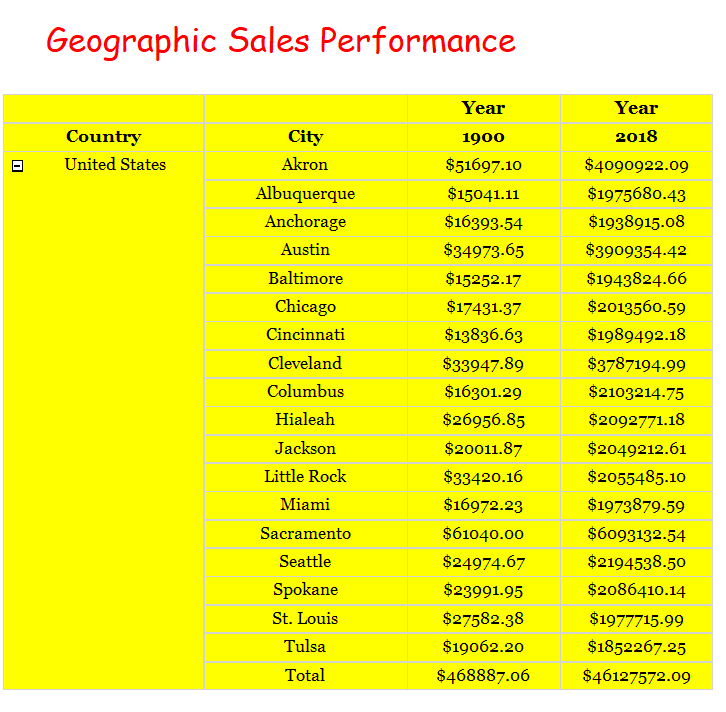
The provided data compares sales performance across various product categories for the years 1900 and 2018, highlighting significant growth in revenue and units sold over this period. The analysis reveals key trends and disparities between categories.

* **Beverages:** Back in 1900, Beverages brought in just $45,443.76 from 969 units sold. Fast forward to 2018, and sales skyrocketed to over $3752021.30 with nearly 77984 units sold showing massive growth in popularity and market reach.
* **Cereals:** Cereal sales followed a similar path. In 1900, revenue was about $36399.65 from 914 units, but by 2018, it had soared to $3882389.42 with over 90,000 units sold reflecting strong, ongoing demand for cereal products.
* **Grain and Meat:** Grain sales grew from just over $33610.85 (744 units) in 1900 to $3330712.89 million (57,868 units) in 2018.
* Meat saw an even more dramatic jump from around $51475.67 (1,009 units) to over $5188079.87 (98,995 units), making it one of the top-selling categories thanks to its staple status in many diets.
* **Poultry and Seafood:** Poultry revenue in 2018 hit a whopping $4492720.21 million from over 93338 units, a huge leap from its humble beginnings in 1900.
* Seafood also experienced significant growth, climbing from just $39,999.68 (717 units) to over $4199568.57 (71,946 units), highlighting increased interest in protein-rich, health-conscious foods.
* **Shellfish and Snails:** Although more niche, both categories saw impressive growth. Shellfish rose from about $26998.98 (589 units) to $3345438.40 (71,684 units).

Snail sales jumped from $45,625.12 (849 units) to nearly $3991975.12 (74,976 units), suggesting growing interest in more diverse food choices.

* **Produce:** Produce sales also took off starting at $41,492.33 (789 units) in 1900 and growing to over $4.513960.57 (84,937 units) in 2018, reflecting the modern consumer’s growing preference for fresh and healthy food options.

1. **Geographical Sales Performance:**



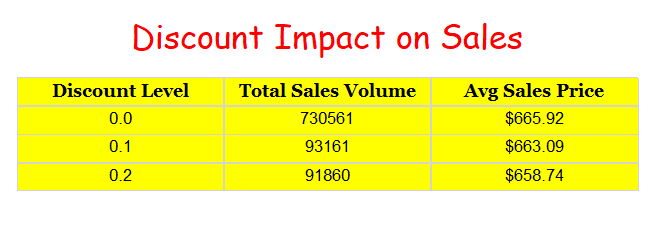
*Fig – 15: Geographical Sales Performance*

Looking at sales data from 1900 to 2018, it’s clear just how much things have changed across U.S. cities. Back in 1900, total sales across the listed cities added up to just $468,887.06. Sacramento led the way with about $61,000 in sales, followed by Akron at $51,697.10, and Cleveland at $33,947.89. Smaller cities like Albuquerque and Baltimore had sales under $20,000, showing that economic activity was limited in certain areas at the time.

Fast forward to 2018, and the numbers are on a completely different level. Total sales exploded to over $46127572.09, showing how much the economy and consumer demand grew over more than a century. Sacramento kept its top spot with sales hitting $6093132.54, while Akron and Cleveland also saw big growth rising to over $4090922.09 and $3787194.99 million, respectively.

Cities like Seattle and Columbus also saw impressive jumps, with each reaching over $2 million in sales. This shows how urban development and rising populations helped fuel economic growth. On the flip side, Tulsa lagged a bit behind, with sales just under $2 million, suggesting some cities didn’t grow quite as fast as others.

1. **Discount Impact on Sales Report**



*Fig – 16: Discount Impact on Sales*

The data provided examines the relationship between discount levels and sales performance, focusing on total sales volume and average sales price. At a discount level of 20% (0.2), the total sales volume reached 93,161 units, with an average sales price of $663.09. This suggests that the 20% discount significantly influenced customer purchasing behavior, driving higher sales volume while reducing the average price per unit. The inverse relationship between discount levels and average sales price is expected, as price reductions typically make products more accessible, encouraging bulk purchases or attracting price-sensitive customers. However, the data does not provide comparative figures for other discount levels or baseline sales (0% discount), making it difficult to assess the full impact or optimal discount strategy. Further analysis with additional discount tiers (e.g., 10%, 30%) would help determine the balance between volume growth and profit margins.

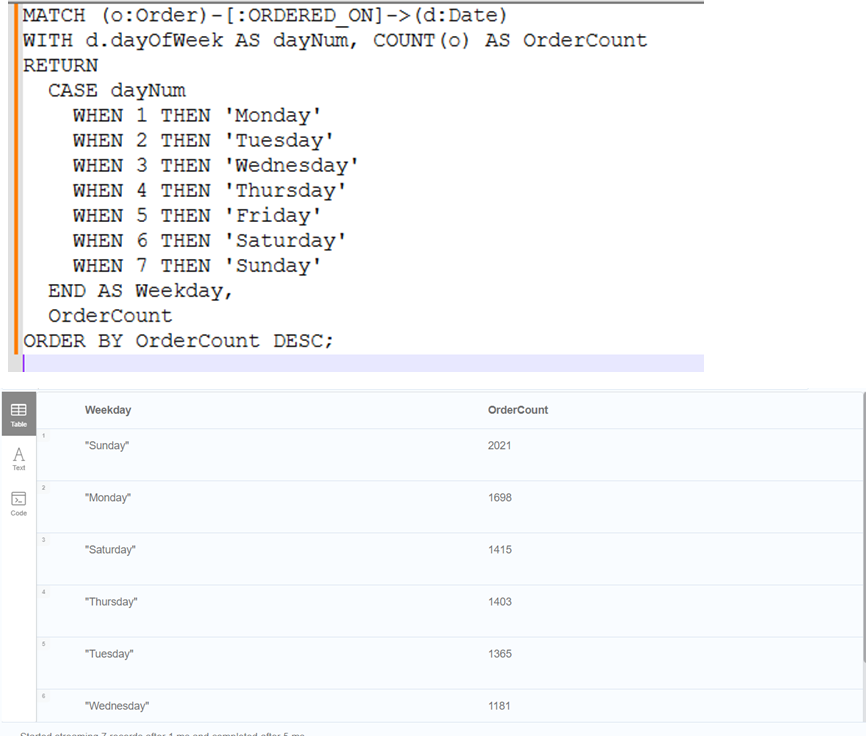
# 5. Graph Databases

Graph data analysis is one of the upcoming methodologies in various niches of computer science. Traditionally for storing, retrieving and experimenting test data, researchers start with mysql database which is more approachable and easier to build their test experimentation platform. These test bed mysql databases will store data in the form of rows and columns, over which various SQL queries are performed.(Johnpaul and Mathew, 2017)

Graph databases are a type of NoSQL database in which data is represented as graphs with nodes and edges. Nodes represent entities in a database and can have associated properties and labels. A graph database is based on a graph data model of properties or labels that provide internal structures with nodes and edges. This type of graph provides additional features to make the graph easier to understand, where nodes can have one or more labels, and relationships between nodes can contain properties.(Rusu and Huang, 2019)

1. **Peak Order Day of the Week:**

This query returns the peak order day of the week, showing the day with the highest number of orders. It provides the number of orders for each weekday and sorts the results by the order count in descending order, displaying the day with the most orders at the top.



*Fig – 17: Peak Order Day of the Week*

1. **Top 5 popular products:**

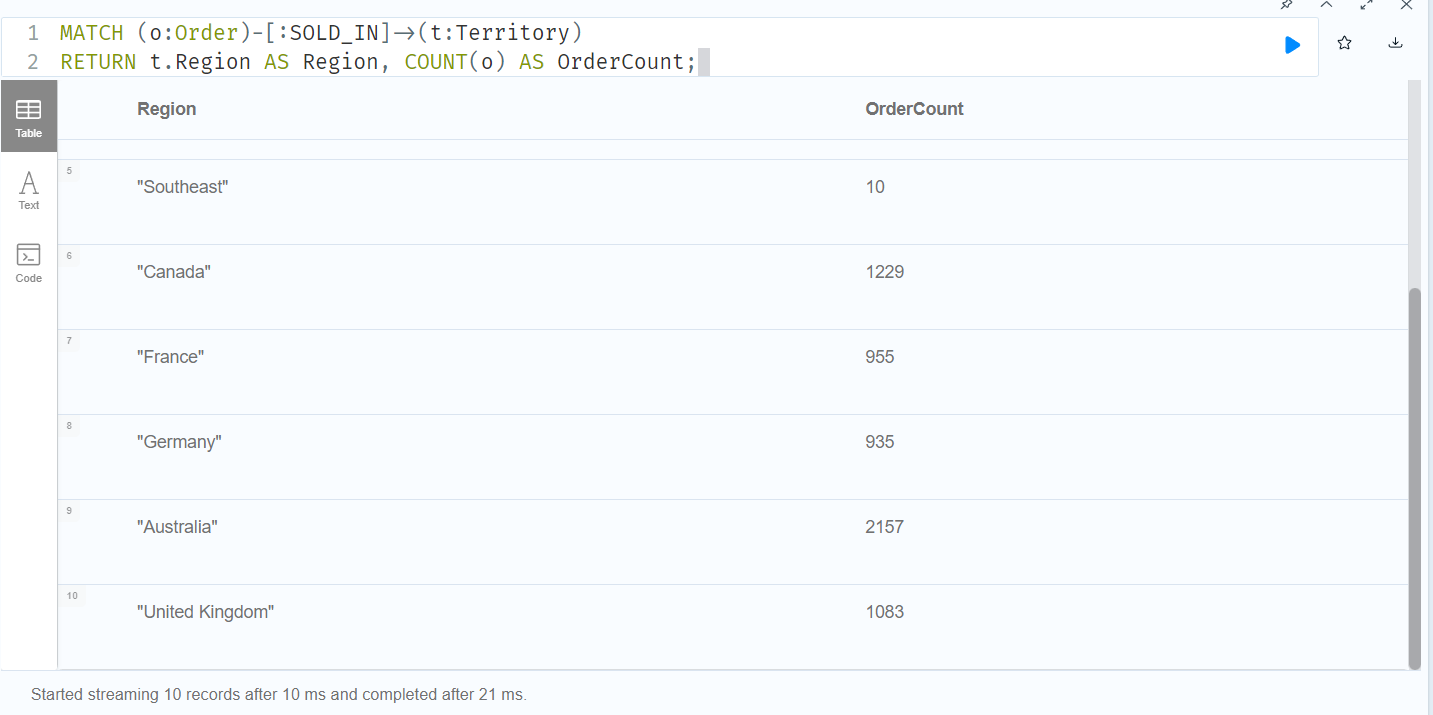
This query returns the top 5 popular products based on the highest order count. It retrieves the product name and the total number of orders for each product, and sorts the results in descending order by the order count. The result is limited to the top 5 products with the most orders.



*Fig – 18: Top 5 popular products*

1. **Number of orders per region:**

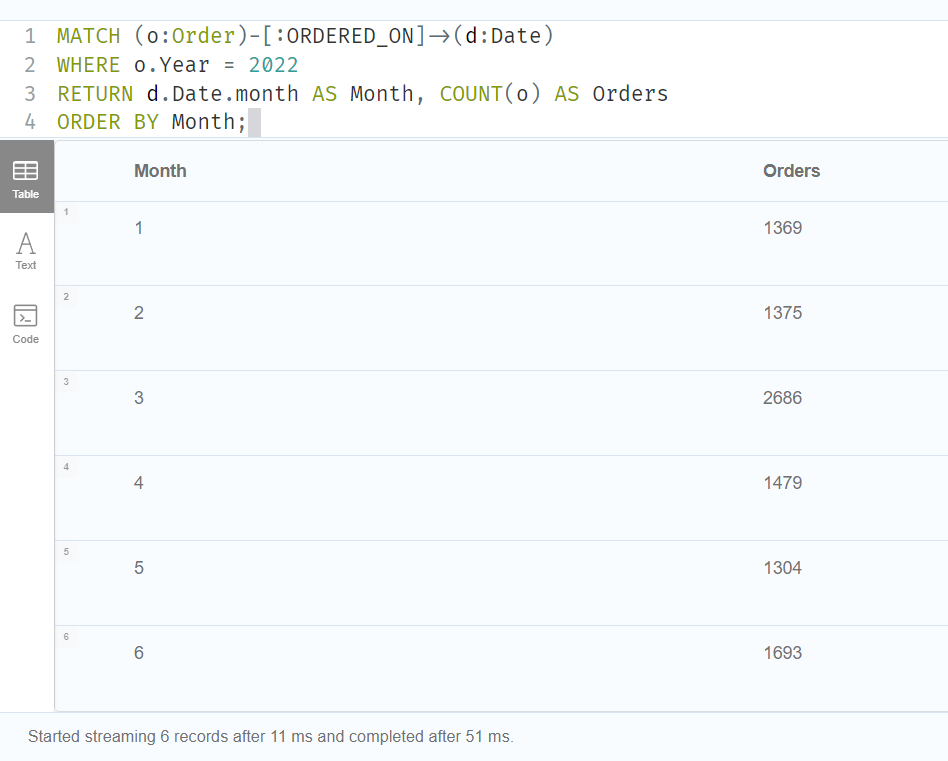
This query retrieves the total number of orders placed in each region. It joins the Sales\_2022 table with the Territory table to match orders with their corresponding regions. The result is grouped by region, and the number of orders is counted for each region.



*Fig – 19: Number of orders per region*

1. **Order count per month in 2022:**

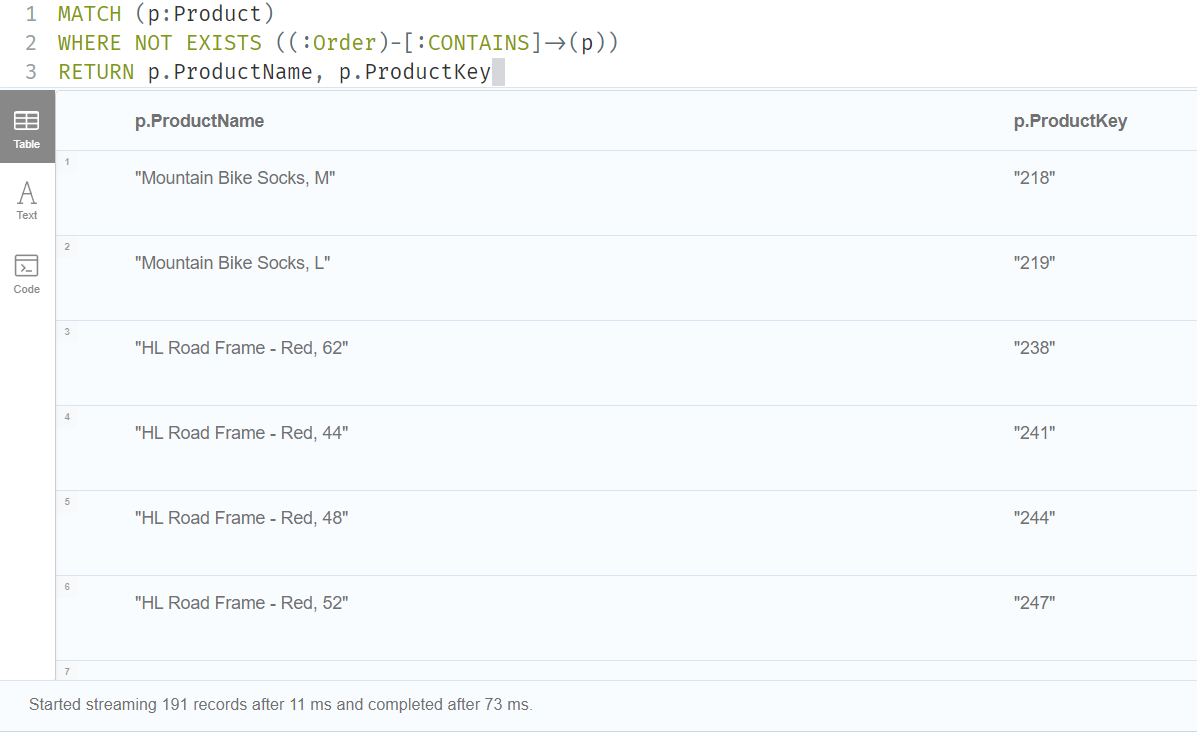
This query retrieves the total number of orders placed for each month in the year 2022. It groups the data by the month extracted from the ***OrderDate*** and counts the number of orders for each month. The result is sorted by the month in ascending order.

****

*Fig – 20: Order count per month in 2022*

1. **Products never ordered:**

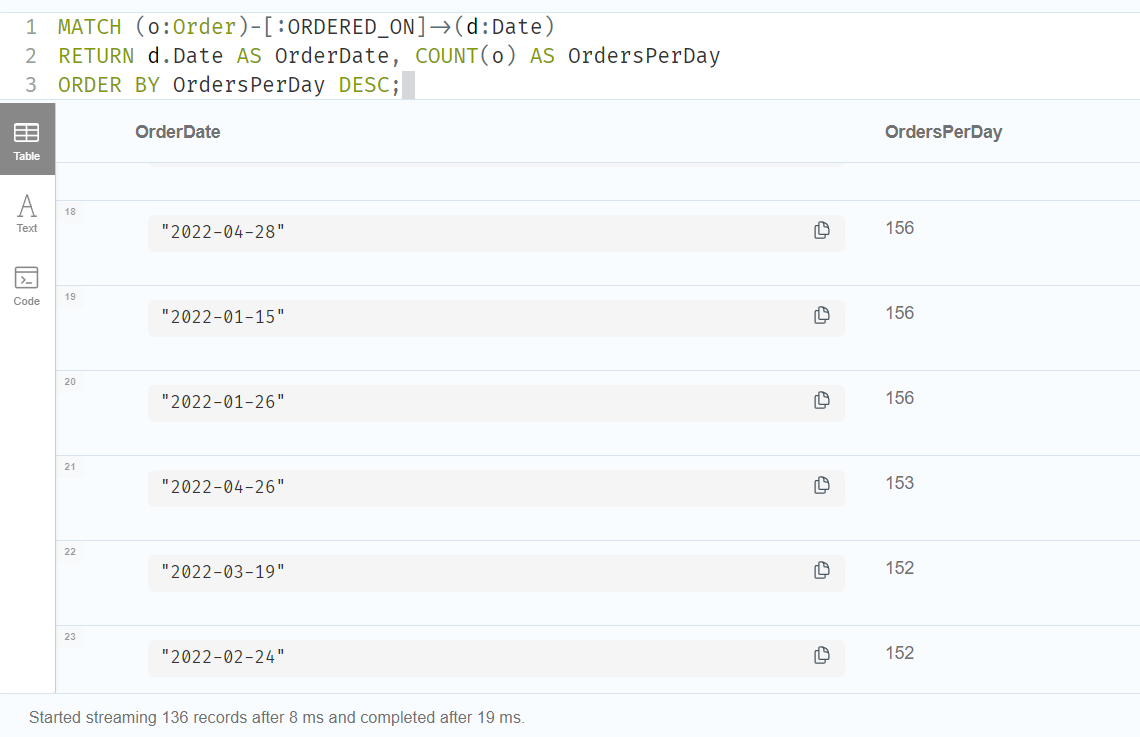
This query retrieves a list of products that have never been ordered. It uses a LEFT JOIN between the Product and Sales\_2022 tables, matching the ProductKey. The query filters for products that do not have corresponding records in the Sales\_2022 table, indicated by a NULL value in the OrderNumber column.

****

*Fig – 21: Products never ordered*

1. **Number of Orders Per Day:**

This query retrieves the total number of orders placed on each day. It groups the data by ***OrderDate*** and counts the number of orders for each specific day. The result is sorted in descending order by the number of orders, displaying the days with the highest order count at the top.

*****Fig – 22: Number of Orders Per Day*

1. **Sales order by color:**

This query returns the number of orders for each product color in the graph database. It matches the ***Order*** and ***Product*** nodes based on the ***CONTAINS*** relationship and counts the number of orders for each product color. The results are sorted in descending order by the number of orders, with the most ordered product color at the top.

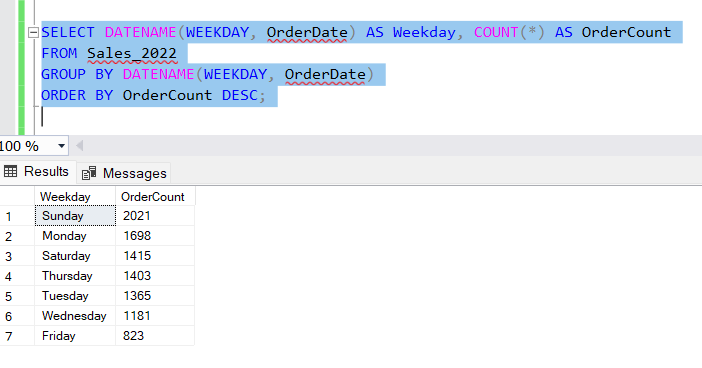
****

*Fig – 23: Sales order by color*

## 5.1. COMAPRISON to realtional databases

1. **Peak Order Day of the Week:**

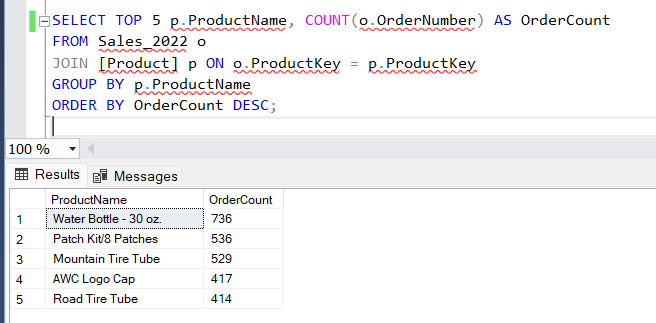
This query returns the peak order day of the week, showing the day with the highest number of orders. It provides the number of orders for each weekday and sorts the results by the order count in descending order, displaying the day with the most orders at the top.

****

*Fig – 24: Peak Order Day of the Week*

1. **Top 5 popular products:**

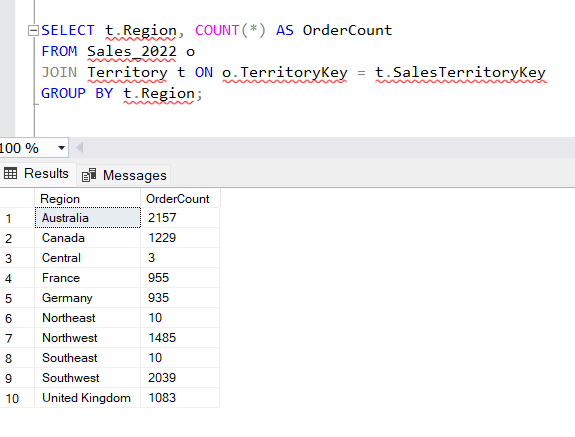
This query returns the top 5 popular products based on the highest order count. It retrieves the product name and the total number of orders for each product, and sorts the results in descending order by the order count. The result is limited to the top 5 products with the most orders.

****

*Fig – 25: Top 5 popular products*

1. **Number of orders per region:**

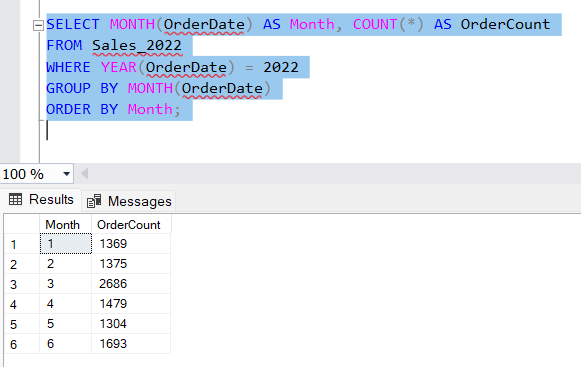
This query retrieves the total number of orders placed in each region. It joins the Sales\_2022 table with the Territory table to match orders with their corresponding regions. The result is grouped by region, and the number of orders is counted for each region.



*Fig – 26: Number of orders per region*

1. **Order count per month in 2022:**

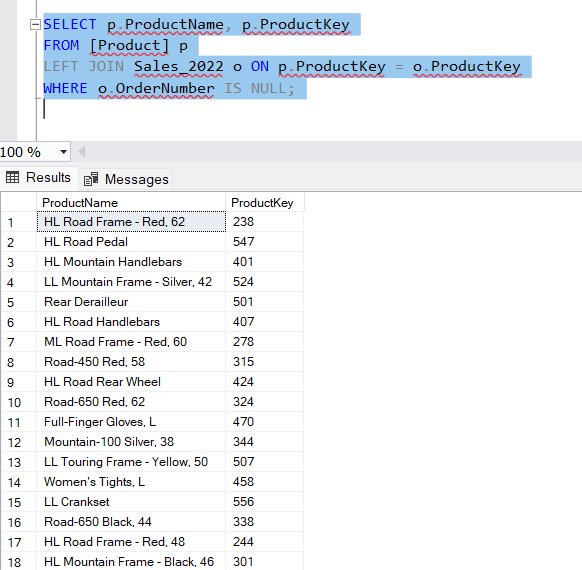
This query retrieves the total number of orders placed for each month in the year 2022. It groups the data by the month extracted from the ***OrderDate*** and counts the number of orders for each month. The result is sorted by the month in ascending order.

****

*Fig – 27: Order count per month in 2022*

1. **Products never ordered:**

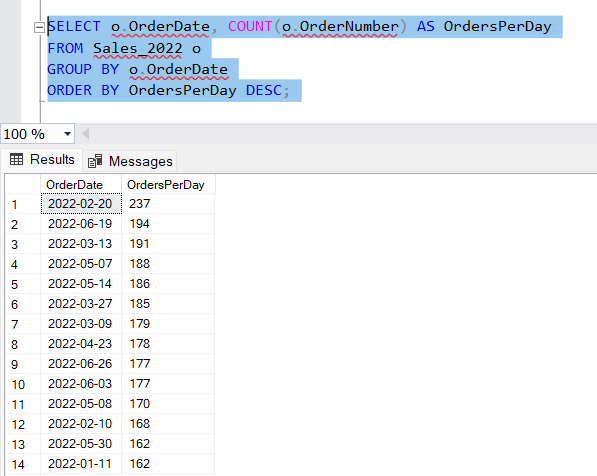
This query retrieves a list of products that have never been ordered. It uses a LEFT JOIN between the ***Product*** and ***Sales\_2022*** tables, matching the ***ProductKey***. The query filters for products that do not have corresponding records in the ***Sales\_2022*** table, indicated by a ***NULL*** value in the ***OrderNumber*** column.

****

*Fig – 28: Products never ordered*

1. **Number of Orders Per Day:**

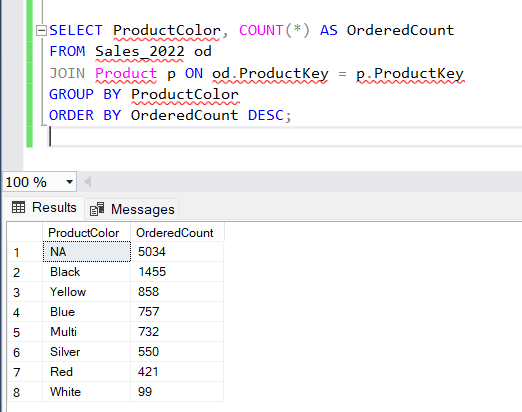
This query retrieves the total number of orders placed on each day. It groups the data by OrderDate and counts the number of orders for each specific day. The result is sorted in descending order by the number of orders, displaying the days with the highest order count at the top.

****

*Fig – 29: Number of Orders Per Day*

1. **Sales order by color:**

This query retrieves the number of orders placed for each product colour. It joins the ***Sales\_2022*** table with the ***Product*** table based on the ***ProductKey*** and counts the number of orders for each product colour. The result is sorted in descending order by the number of orders, displaying the most ordered product colour at the top.

****

*Fig – 30: Sales order by color*

# 6. Conclusions

In this project, we created a data warehouse and interactive dashboard using grocery sales data. We used a star schema to store data from customers, products, employees, and locations. We used SSIS for the ETL process and built visual dashboards in Tableau. These dashboards help users understand sales trends, product performance, and employee contributions. We also used Neo4j graph database to explore advanced queries and compared it with SQL.

From our analysis, we found that non-allergic products had the highest sales, and categories like seafood and poultry showed strong growth. Discounts increased sales volume, and some cities like Sacramento showed the highest sales. Graph databases helped us find useful patterns, such as peak order days and most ordered products.

Overall, this project shows how data warehouses and visual dashboards help businesses make better decisions using data.

# 7. Bibliography

Farheen, Z., Dharani, A., 2024. Prediction of Customer Purchasing Patterns for Retail Optimization Using Market Basket Techniques, in: 8th IEEE International Conference on Computational System and Information Technology for Sustainable Solutions, CSITSS 2024. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/CSITSS64042.2024.10816740

Johnpaul, C.I., Mathew, T., 2017. A Cypher query based NoSQL data mining on protein datasets using Neo4j graph database, in: 2017 4th International Conference on Advanced Computing and Communication Systems, ICACCS 2017. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICACCS.2017.8014558

Köhler, H., Link, S., 2016. SQL schema design: Foundations, normal forms, and normalization, in: Proceedings of the ACM SIGMOD International Conference on Management of Data. Association for Computing Machinery, pp. 267–279. https://doi.org/10.1145/2882903.2915239

Rusu, F., Huang, Z., 2019. In-Depth Benchmarking of Graph Database Systems with the Linked Data Benchmark Council (LDBC) Social Network Benchmark (SNB).

# Appendix B – Neo 4J code

LOAD CSV WITH HEADERS FROM 'file:///Product\_Categories.csv' AS row

CREATE (:ProductCategory {

ProductCategoryKey: row.ProductCategoryKey,

CategoryName: row.CategoryName

});

LOAD CSV WITH HEADERS FROM 'file:///Product\_Subcategories.csv' AS row

CREATE (:ProductSubcategory {

ProductSubcategoryKey: row.ProductSubcategoryKey,

SubcategoryName: row.SubcategoryName,

ProductCategoryKey: row.ProductCategoryKey

});

LOAD CSV WITH HEADERS FROM 'file:///Product.csv' AS row

CREATE (:Product {

ProductKey: row.ProductKey,

ProductSubcategoryKey: row.ProductSubcategoryKey,

ProductName: row.ProductName,

ModelName: row.ModelName,

ProductDescription: row.ProductDescription,

ProductColor: row.ProductColor,

ProductSize: row.ProductSize,

ProductStyle: row.ProductStyle,

ProductCost: toFloat(row.ProductCost),

ProductPrice: toFloat(row.ProductPrice)

});

LOAD CSV WITH HEADERS FROM 'file:///Customer.csv' AS row

WITH row, split(row.BirthDate, "-") AS dateParts

WHERE size(dateParts) = 3

AND toInteger(dateParts[0]) >= 1 AND toInteger(dateParts[0]) <= 12

AND toInteger(dateParts[1]) >= 1 AND toInteger(dateParts[1]) <= 31

AND toInteger(dateParts[2]) >= 1900 AND toInteger(dateParts[2]) <= 2100

WITH row, date(dateParts[2] + "-" + dateParts[0] + "-" + dateParts[1]) AS parsedBirthDate

CREATE (:Customer {

CustomerKey: row.CustomerKey,

Prefix: row.Prefix,

FirstName: row.FirstName,

LastName: row.LastName,

BirthDate: parsedBirthDate,

MaritalStatus: row.MaritalStatus,

Gender: row.Gender,

EmailAddress: row.EmailAddress,

AnnualIncome: toFloat(row.AnnualIncome),

TotalChildren: toInteger(row.TotalChildren),

EducationLevel: row.EducationLevel,

Occupation: row.Occupation,

HomeOwner: toBoolean(row.HomeOwner)

});

LOAD CSV WITH HEADERS FROM 'file:///Territory.csv' AS row

CREATE (:Territory {

SalesTerritoryKey: row.SalesTerritoryKey,

Region: row.Region,

Country: row.Country,

Continent: row.Continent

});

LOAD CSV WITH HEADERS FROM 'file:///Sales\_2022.csv' AS row

WITH row,

split(row.OrderDate, "/") AS OrderParts,

split(row.StockDate, "/") AS StockParts

WHERE size(OrderParts) = 3 AND size(StockParts) = 3

AND toInteger(OrderParts[0]) >= 1 AND toInteger(OrderParts[0]) <= 31

AND toInteger(OrderParts[1]) >= 1 AND toInteger(OrderParts[1]) <= 12

AND toInteger(OrderParts[2]) >= 1900 AND toInteger(OrderParts[2]) <= 2100

AND toInteger(StockParts[0]) >= 1 AND toInteger(StockParts[0]) <= 31

AND toInteger(StockParts[1]) >= 1 AND toInteger(StockParts[1]) <= 12

AND toInteger(StockParts[2]) >= 1900 AND toInteger(StockParts[2]) <= 2100

WITH row,

date(OrderParts[2] + "-" + OrderParts[1] + "-" + OrderParts[0]) AS parsedOrderDate,

date(StockParts[2] + "-" + StockParts[1] + "-" + StockParts[0]) AS parsedStockDate

CREATE (:Order {

OrderDate: parsedOrderDate,

StockDate: parsedStockDate,

OrderNumber: toString(row.OrderNumber),

ProductKey: row.ProductKey,

CustomerKey: row.CustomerKey,

TerritoryKey: row.TerritoryKey,

OrderLineItem: toInteger(row.OrderLineItem),

OrderQuantity: toInteger(row.OrderQuantity),

Year: 2022

});

CREATE CONSTRAINT FOR (p:Product) REQUIRE p.ProductKey IS UNIQUE;

CREATE CONSTRAINT FOR (c:Customer) REQUIRE c.CustomerKey IS UNIQUE;

CREATE CONSTRAINT FOR (t:Territory) REQUIRE t.SalesTerritoryKey IS UNIQUE;

CREATE INDEX FOR (o:Order) ON (o.OrderNumber);

CREATE INDEX FOR (d:Date) ON (d.Date);

**// Create relationships between Orders and Products (optimized)**

MATCH (o:Order)

MATCH (p:Product {ProductKey: o.ProductKey})

CREATE (o)-[:CONTAINS]->(p);

**// Create relationships between Orders and Customers (optimized)**

MATCH (o:Order)

MATCH (c:Customer {CustomerKey: o.CustomerKey})

CREATE (o)-[:PLACED\_BY]->(c);

**// Create relationships between Orders and Territories (optimized)**

MATCH (o:Order)

MATCH (t:Territory {SalesTerritoryKey: o.TerritoryKey})

CREATE (o)-[:SOLD\_IN]->(t);

**// Create relationships between Products and Subcategories (optimized)**

MATCH (p:Product)

MATCH (s:ProductSubcategory {ProductSubcategoryKey: p.ProductSubcategoryKey})

CREATE (p)-[:BELONGS\_TO\_SUBCATEGORY]->(s);

**// Create relationships between Subcategories and Categories (optimized)**

MATCH (s:ProductSubcategory)

MATCH (c:ProductCategory {ProductCategoryKey: s.ProductCategoryKey})

CREATE (s)-[:BELONGS\_TO\_CATEGORY]->(c);

**// Create date relationships (optimized)**

MATCH (o:Order)

WITH o, o.OrderDate AS orderDate

MERGE (d:Date {Date: orderDate})

CREATE (o)-[:ORDERED\_ON]->(d);

**// Create relationships between Order and stockdate**

MATCH (o:Order)

WITH o, o.StockDate AS stockDate

MERGE (d:Date {Date: stockDate})

CREATE (o)-[:STOCKED\_ON]->(d);

**//1 Peak Order Day of the Week**

MATCH (o:Order)-[:ORDERED\_ON]->(d:Date)

WITH date(d.Date).dayOfWeek AS dayNum, COUNT(o) AS OrderCount

RETURN

CASE dayNum

WHEN 1 THEN 'Monday'

WHEN 2 THEN 'Tuesday'

WHEN 3 THEN 'Wednesday'

WHEN 4 THEN 'Thursday'

WHEN 5 THEN 'Friday'

WHEN 6 THEN 'Saturday'

WHEN 7 THEN 'Sunday'

END AS Weekday,

OrderCount

ORDER BY OrderCount DESC;

**//2 Find the most popular products (top 5 by order count)**

MATCH (o:Order)-[:CONTAINS]->(p:Product)

RETURN p.ProductName, COUNT(o) AS OrderCount

ORDER BY OrderCount DESC

LIMIT 5;

**//3 Number of orders per region**

MATCH (o:Order)-[:SOLD\_IN]->(t:Territory)

RETURN t.Region AS Region, COUNT(o) AS OrderCount;

**//4 Order count per month in 2022**

MATCH (o:Order)-[:ORDERED\_ON]->(d:Date)

WHERE o.Year = 2022

RETURN d.Date.month AS Month, COUNT(o) AS Orders

ORDER BY Month;

**//5 Products never ordered:**

MATCH (p:Product)

WHERE NOT EXISTS ((:Order)-[:CONTAINS]->(p))

RETURN p.ProductName, p.ProductKey

**//6 Number of Orders Per Day (Based on Order Date):**

MATCH (o:Order)-[:ORDERED\_ON]->(d:Date)

RETURN d.Date AS OrderDate, COUNT(o) AS OrdersPerDay

ORDER BY OrdersPerDay DESC;

**//7 Sales order by color**

MATCH (o:Order)-[:CONTAINS]->(p:Product)

RETURN p.ProductColor, Count(\*) AS OrderCount

ORDER BY OrderCount DESC;