Title: Coffee Shop Chain Database

(Katerina Shvetsova 1637126, Ziyu Li 1052776)

Introduction

A well-structured database plays a vital role in supporting the daily operations and long-term success of a company. It serves as the main storage system for all business information, allowing data to be organized, updated, and retrieved when needed. An efficient database not only helps the owner to track and analyze company activities but also provides a reliable foundation for decision-making. To achieve this, the system must be clearly structured, manageable, easy to integrate, and without unnecessary redundancies.

In this project, we will create a database that meets the needs of a client coffee shop chain and fulfills the requirements of a Hi-Fi database. The company already stores sales, product, and employee information, but their current system is fragmented and lacks advanced analytical capabilities. For example, sales performance, inventory tracking, and supplier management are not fully integrated, making it difficult to extract meaningful insights. As a result, it is difficult to analyze relationships between sales performance, inventory levels, and supplier activities. The absence of unified data management also limits the company’s ability to make data-driven decisions.

To improve this system, we designed a more complete database model that includes not only sales and employees, but also inventory management, suppliers, promotions, and store operations. The new system integrates sales, employee, and supplier data with modules for inventory management, promotional campaigns, and store operations. It will also support the visualization of sales trends, stock movements, and performance indicators across different stores, helping the management team monitor progress and plan future strategies more effectively. Furthermore, management is interested in comparing sales data across time periods and categories, which makes data visualization an important aspect of this project.

The main topics we aim to research and address are formulated into the following questions:

Sales Analysis

- What is the total revenue per store and per product category?
- How does sales growth compare across different periods?
- Who are the top-performing employees?

Inventory and Supplier Management

- What is the current stock level and valuation per store?

- How are supply orders distributed across suppliers and products?

Promotions and Marketing

- Which promotions generate the most sales impact?
- How are discounts applied across products and stores?

Data Visualization with Power BI

- How can Power BI be used to create engaging and informative dashboards from the CoffeeShop database?

Entity-Relationship Diagram

The ER diagram illustrates the entities, attributes, and relationships that form the structure of the CoffeeShop database. It serves as a key element in the database design process, providing a clear overview of all components and how they are interconnected. In our model, the main entities include **Employee**, **Inventory**, **Product**, **Promotion**, **Sale**, **Sale_Item**, **Store**, **Supplier**, and **Supply_Order**. Each entity contains a distinct set of attributes, which correspond to the columns in their respective database tables.

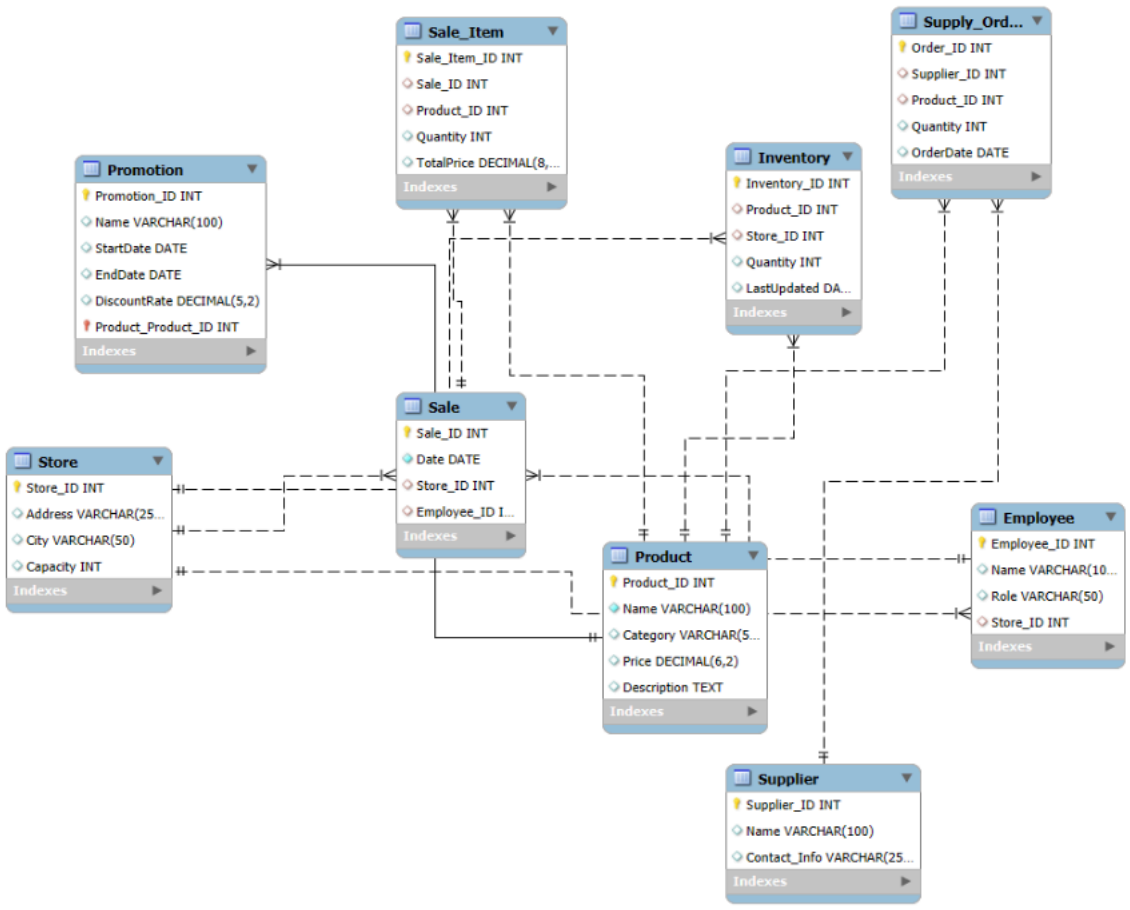


Figure 1. Entity-Relationship diagram for the database for the Coffeeshop company. The relationships between the entities are illustrated with the connection line.

Entities of Coffee Shop DB

1. Entity Employee

Employee stores information about each staff member working in the coffee shop chain. Each employee has a unique Employee_ID, a name, a role (e.g., Barista, Manager), and is assigned to a specific store via Store_ID. This table is central for tracking sales performance and shift management.

2. Entity Store

Store contains details of all coffee shop locations. Attributes include Store_ID, address, city, and store capacity. This table is used to analyze sales and inventory per location.

3. Entity Product

Product lists all items sold in the coffee shop, including beverages and bakery products. Each product has a unique Product_ID, name, category (e.g., Beverage, Bakery), price, and description. This table connects sales and inventory data.

4. Entity Inventory

Inventory tracks stock levels of products in each store. Attributes include Inventory_ID, Product_ID, Store_ID, quantity, and the date of last update (LastUpdated). It is essential for monitoring stock and supply needs.

5. Entity Sale

Sale records each transaction in the coffee shop. Key attributes are Sale_ID, sale date, the store where it occurred (Store_ID), and the employee responsible (Employee_ID). This table links directly to sale items for revenue analysis.

6. Entity Sale_Item

Sale_Item stores the details of products sold in each sale. Each record includes Sale_Item_ID, the Sale_ID it belongs to, Product_ID, quantity sold, and the total price. This table enables detailed sales reporting per product.

7. Entity Supplier

Supplier contains information about the vendors providing products and raw materials to the coffee shops. Key attributes include Supplier_ID, name, and contact information. It is used to track orders and supply sources.

8. Entity Supply_Order

Supply_Order stores all orders made to suppliers. Attributes include Order_ID, Supplier_ID, Product_ID, quantity ordered, and order date (OrderDate). This table is critical for managing supply chains and stock replenishment.

9. Entity Promotion

Promotion tracks discounts and campaigns offered in the coffee shop. Key attributes are Promotion_ID, name, start and end dates, and discount rate. This entity helps evaluate the effect of promotions on sales.

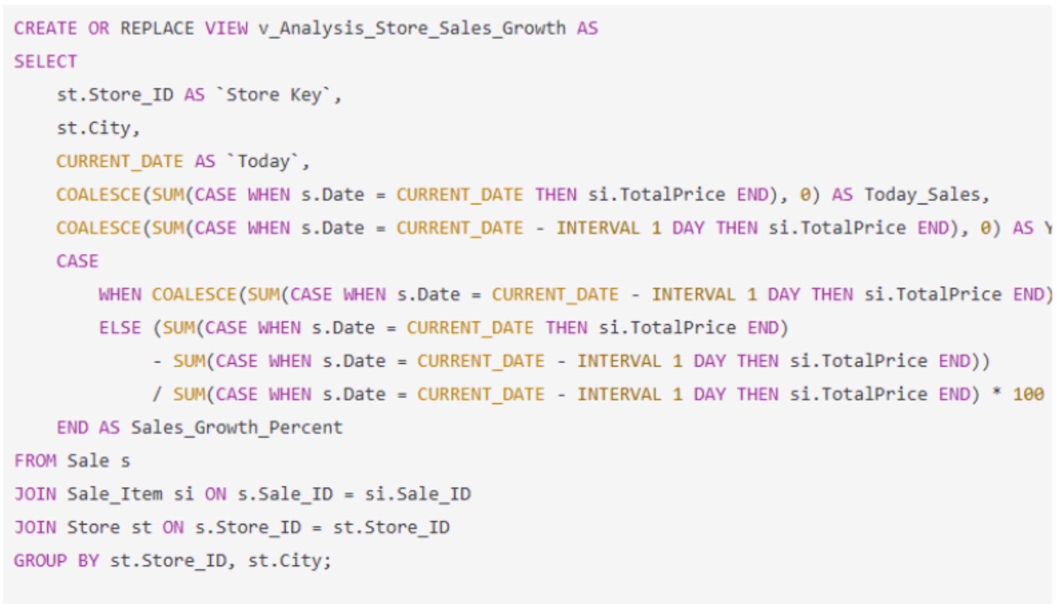
SQL Queries

1.**View: v_Analysis_Store_Sales_Growth** (sales growth per store (compared to previous day)

Purpose: Track daily sales growth per store.

Description: Compares today’s sales with yesterday’s and calculates percentage growth.

Use: Monitor short-term performance, identify trends, guide operational adjustments.



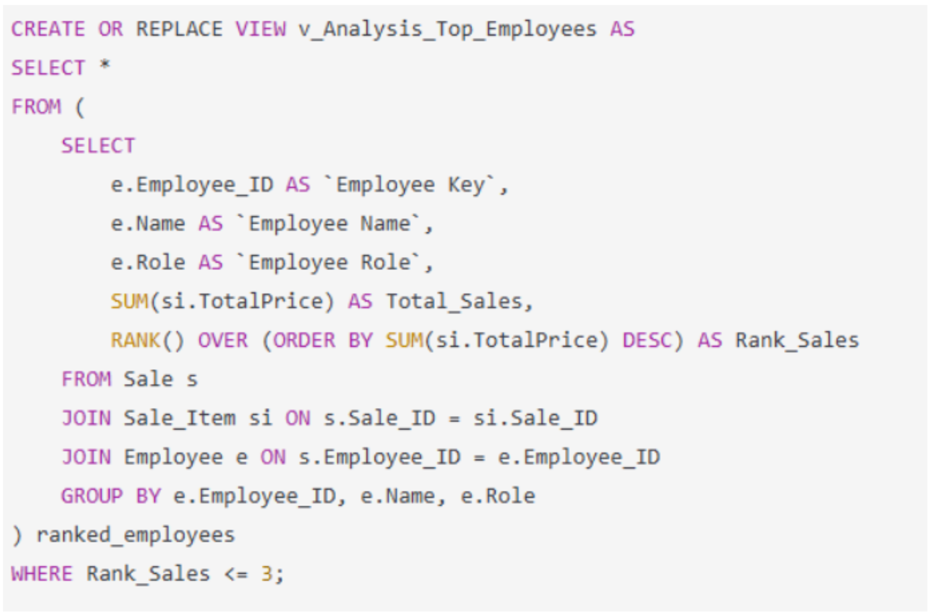
```sql
CREATE OR REPLACE VIEW v_Analysis_Store_Sales_Growth AS
SELECT
    st.Store_ID AS `Store Key`,
    st.City,
    CURRENT_DATE AS `Today`,
    COALESCE(SUM(CASE WHEN s.Date = CURRENT_DATE THEN si.TotalPrice END), 0) AS Today_Sales,
    COALESCE(SUM(CASE WHEN s.Date = CURRENT_DATE - INTERVAL 1 DAY THEN si.TotalPrice END), 0) AS Y
    CASE
        WHEN COALESCE(SUM(CASE WHEN s.Date = CURRENT_DATE - INTERVAL 1 DAY THEN si.TotalPrice END)
        ELSE (SUM(CASE WHEN s.Date = CURRENT_DATE THEN si.TotalPrice END)
             - SUM(CASE WHEN s.Date = CURRENT_DATE - INTERVAL 1 DAY THEN si.TotalPrice END))
             / SUM(CASE WHEN s.Date = CURRENT_DATE - INTERVAL 1 DAY THEN si.TotalPrice END) * 100
    END AS Sales_Growth_Percent
FROM Sale s
JOIN Sale_Item si ON s.Sale_ID = si.Sale_ID
JOIN Store st ON s.Store_ID = st.Store_ID
GROUP BY st.Store_ID, st.City;
```

2. **View: v_Analysis_Top_Employees** (sales per employee)

Purpose: Find the top 3 employees with the highest sales.

Description: Rank the employee sales from all the shops.

Use: Track the employee performance, establish a reward system.



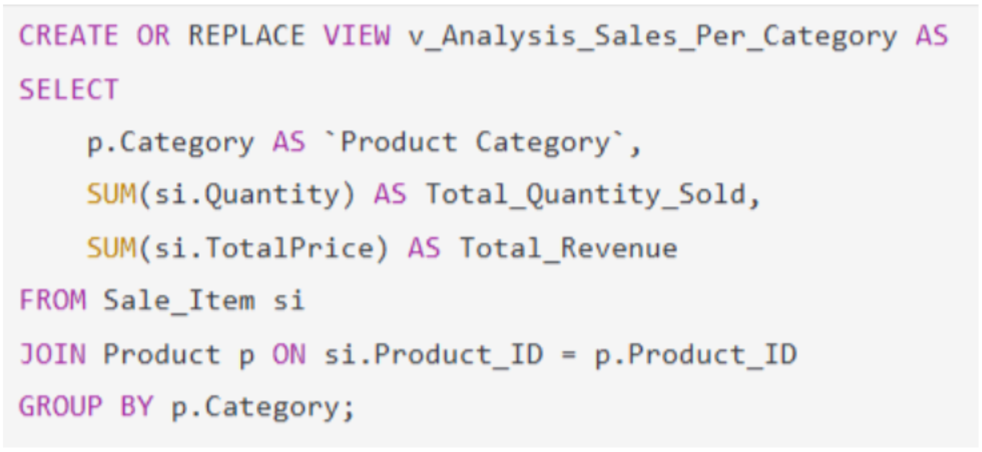
```sql
CREATE OR REPLACE VIEW v_Analysis_Top_Employees AS
SELECT *
FROM (
    SELECT
        e.Employee_ID AS `Employee Key`,
        e.Name AS `Employee Name`,
        e.Role AS `Employee Role`,
        SUM(si.TotalPrice) AS Total_Sales,
        RANK() OVER (ORDER BY SUM(si.TotalPrice) DESC) AS Rank_Sales
    FROM Sale s
    JOIN Sale_Item si ON s.Sale_ID = si.Sale_ID
    JOIN Employee e ON s.Employee_ID = e.Employee_ID
    GROUP BY e.Employee_ID, e.Name, e.Role
) ranked_employees
WHERE Rank_Sales <= 3;
```

3.**VIEW v_Analysis_Sales_Per_Category**

Purpose: Understand sales performance by product category.

Description: Calculates total quantity sold and revenue per category.

Use: Inform marketing strategies, guide product assortment decisions.



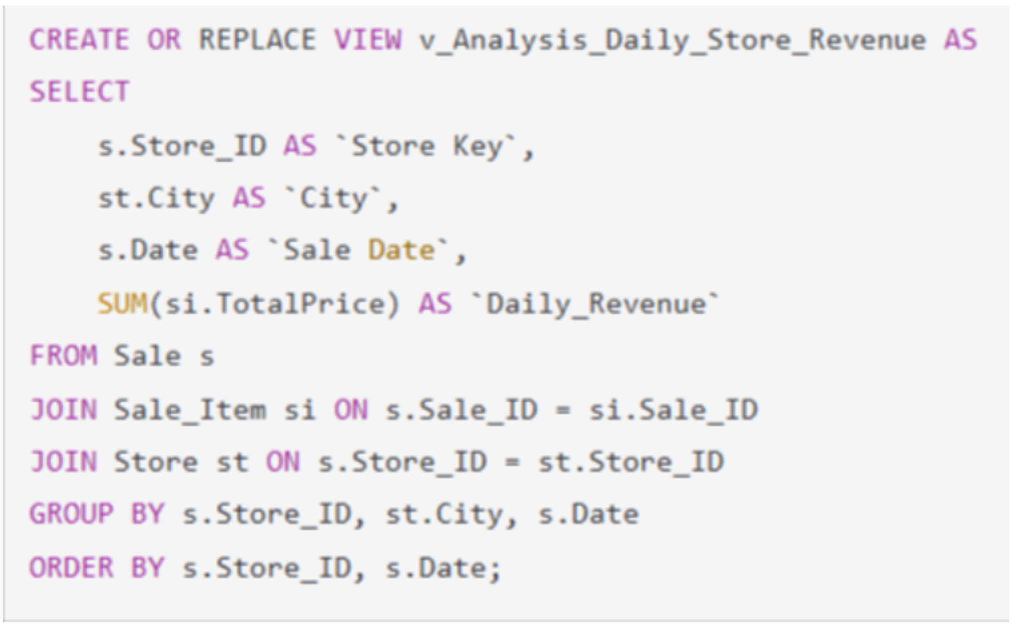
```sql
CREATE OR REPLACE VIEW v_Analysis_Sales_Per_Category AS
SELECT
    p.Category AS `Product Category`,
    SUM(si.Quantity) AS Total_Quantity_Sold,
    SUM(si.TotalPrice) AS Total_Revenue
FROM Sale_Item si
JOIN Product p ON si.Product_ID = p.Product_ID
GROUP BY p.Category;
```

4. **View: v_Analysis_Daily_Store_Revenue**

Purpose: Analyze daily revenue performance per store.

Description: Calculates total revenue per store per day by summing the TotalPrice of all sale items.

Use: Track daily sales trends, compare store performance, identify peak sales days.



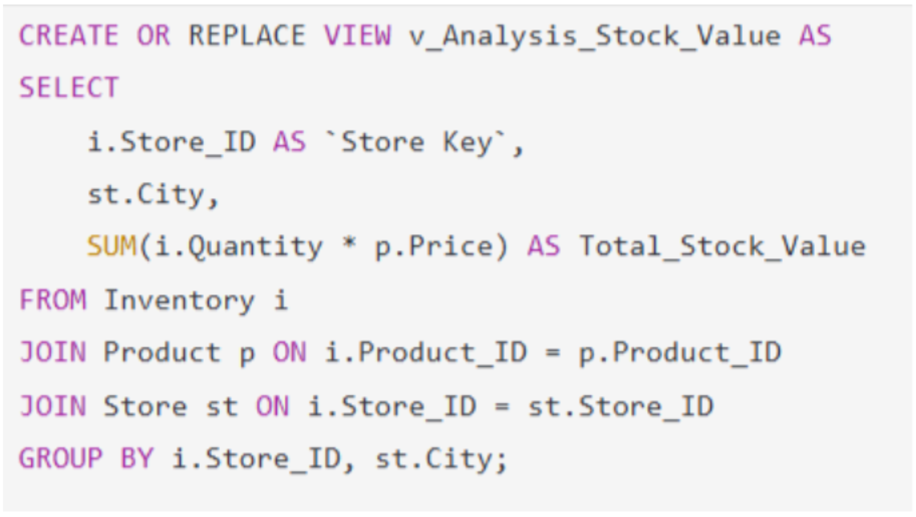
```sql
CREATE OR REPLACE VIEW v_Analysis_Daily_Store_Revenue AS
SELECT
    s.Store_ID AS `Store Key`,
    st.City AS `City`,
    s.Date AS `Sale Date`,
    SUM(si.TotalPrice) AS `Daily_Revenue`
FROM Sale s
JOIN Sale_Item si ON s.Sale_ID = si.Sale_ID
JOIN Store st ON s.Store_ID = st.Store_ID
GROUP BY s.Store_ID, st.City, s.Date
ORDER BY s.Store_ID, s.Date;
```

5. **View: v_Analysis_Stock_Value**

Purpose: Determine total monetary value of stock per store.

Description: Computes inventory value per store by multiplying quantity by price.

Use: Supports financial reporting, inventory valuation, and cost management



```sql
CREATE OR REPLACE VIEW v_Analysis_Stock_Value AS
SELECT
    i.Store_ID AS `Store Key`,
    st.City,
    SUM(i.Quantity * p.Price) AS Total_Stock_Value
FROM Inventory i
JOIN Product p ON i.Product_ID = p.Product_ID
JOIN Store st ON i.Store_ID = st.Store_ID
GROUP BY i.Store_ID, st.City;
```

6.**View: v_Analysis_Employee_Sales**

Purpose: Evaluate sales performance of employees.

Description: Aggregates total sales per employee, including name and role.

Use: Identify top-performing employees and assess individual contribution.

```
CREATE OR REPLACE VIEW v_Analysis_Employee_Sales AS
SELECT
    e.Employee_ID AS `Employee Key`,
    e.Name AS `Employee Name`,
    e.Role AS `Employee Role`,
    SUM(si.TotalPrice) AS Total_Sales
FROM Sale s
JOIN Sale_Item si ON s.Sale_ID = si.Sale_ID
JOIN Employee e ON s.Employee_ID = e.Employee_ID
GROUP BY e.Employee_ID, e.Name, e.Role;
```

7.View: v_Analysis_Inventory_Stock

Purpose: Monitor stock levels per product in each store.

Description: Summarizes quantity of each product per store and last update date.

Use: Support inventory management, prevent stockouts, plan restocking.

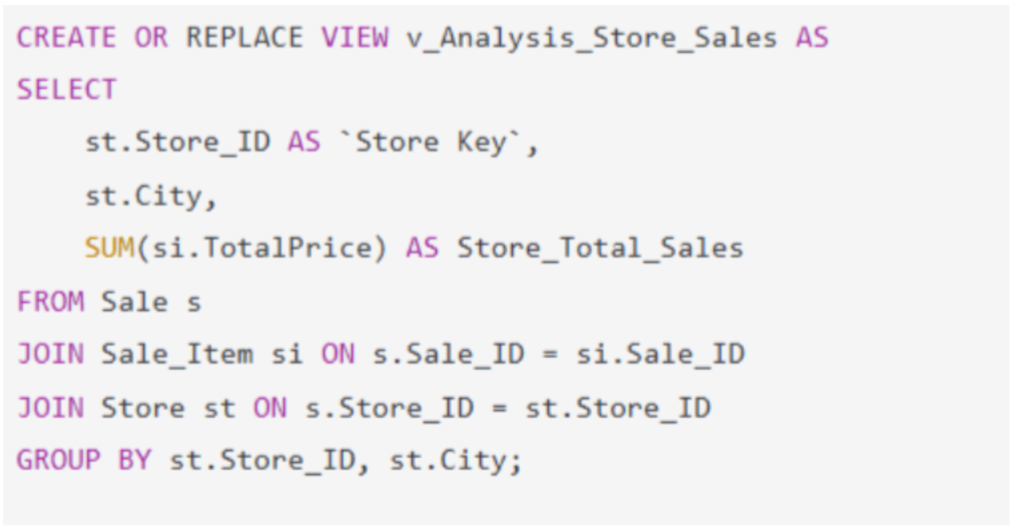
```
CREATE OR REPLACE VIEW v_Analysis_Inventory_Stock AS
SELECT
    i.Store_ID AS `Store Key`,
    i.Product_ID AS `Product Key`,
    SUM(i.Quantity) AS Total_Stock,
    MAX(i.LastUpdated) AS Last_Updated
FROM Inventory i
GROUP BY i.Store_ID, i.Product_ID;
```

8. **View: v_Analysis_Store_Sales (Total sales performance per store)**

Purpose: Evaluate total sales performance per store.

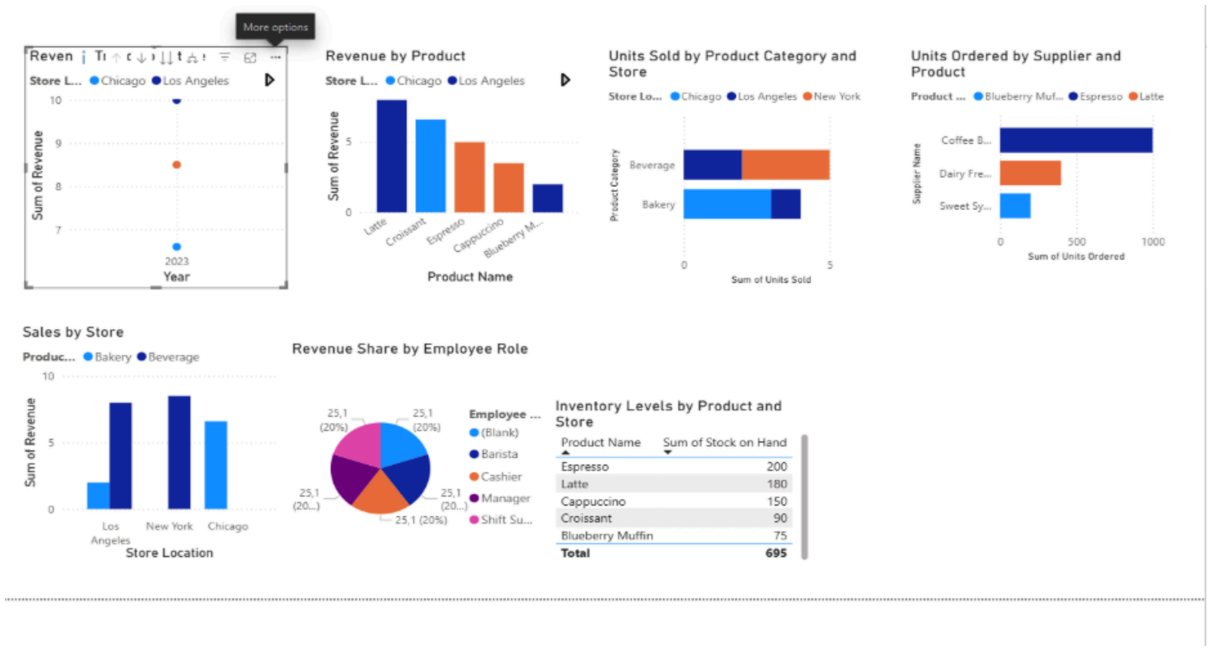
Description: Aggregates total revenue per store including city.

Use: Compare store performance, identify high- and low-performing locations.



```sql
CREATE OR REPLACE VIEW v_Analysis_Store_Sales AS
SELECT
    st.Store_ID AS `Store Key`,
    st.City,
    SUM(si.TotalPrice) AS Store_Total_Sales
FROM Sale s
JOIN Sale_Item si ON s.Sale_ID = si.Sale_ID
JOIN Store st ON s.Store_ID = st.Store_ID
GROUP BY st.Store_ID, st.City;
```

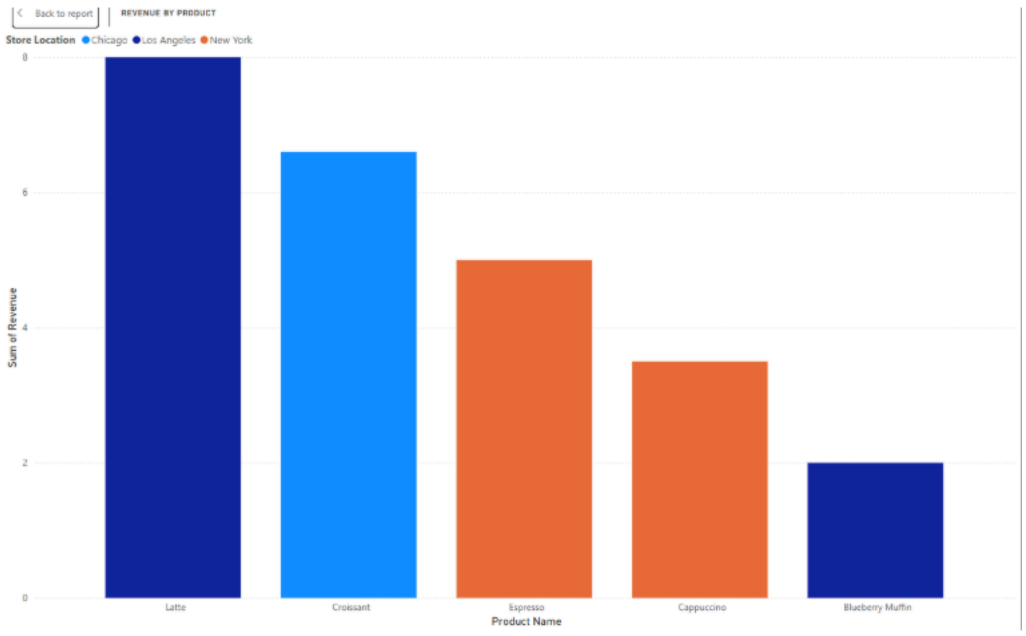
Advanced Functions



Product Name	Sum of Stock on Hand
Espresso	200
Latte	180
Cappuccino	150
Croissant	90
Blueberry Muffin	75
Total	**695**

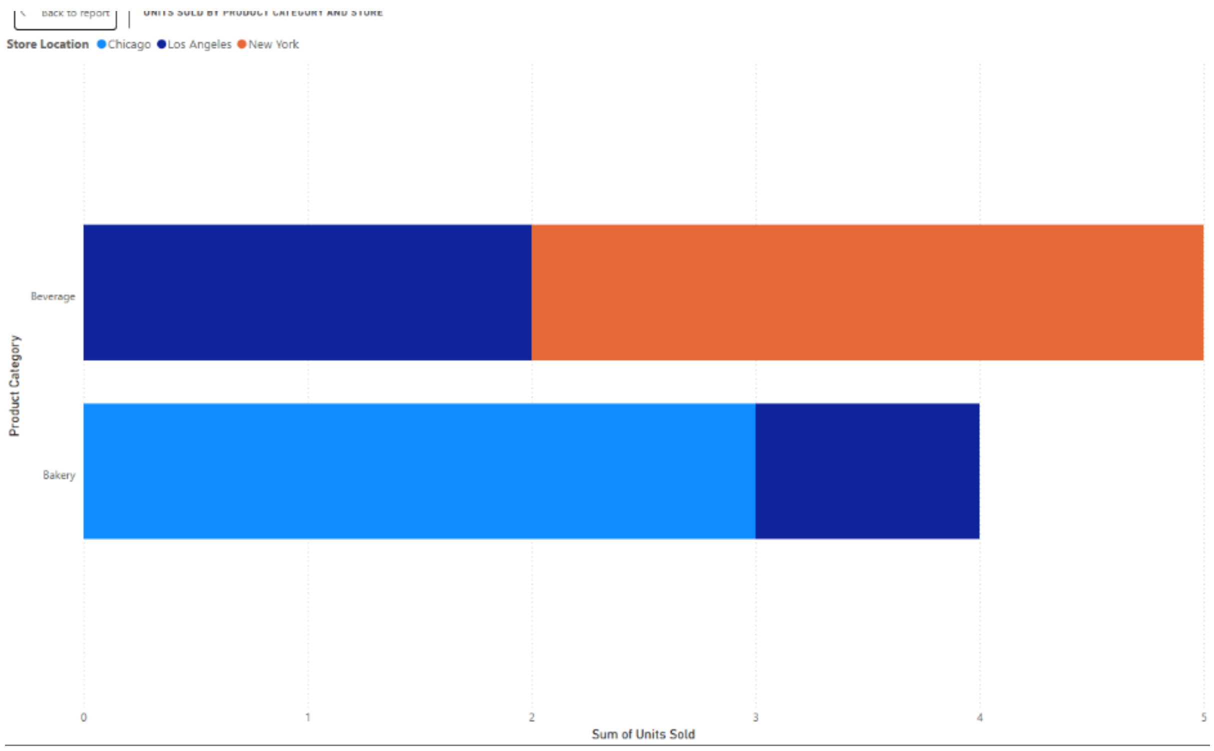
1. Revenue by Product

Goal: Show total revenue per product



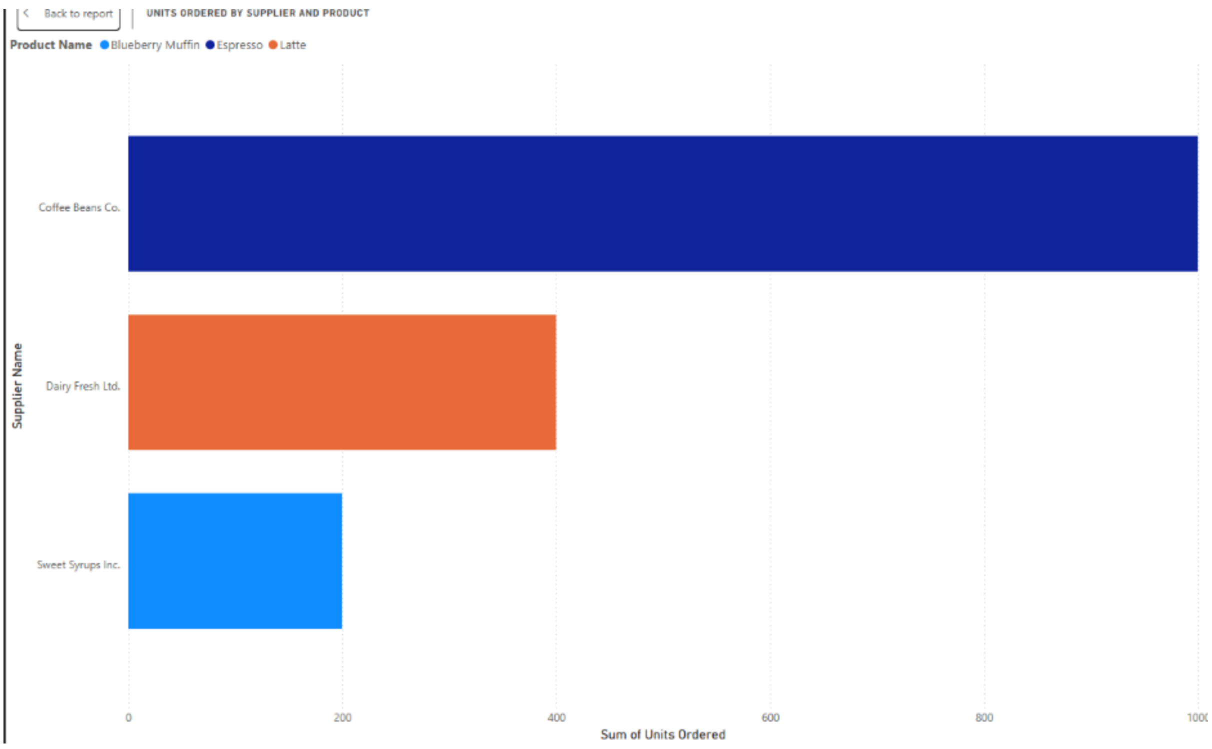
2. Units Sold by Product Category and Store

Goal: Show total units sold by category and store.



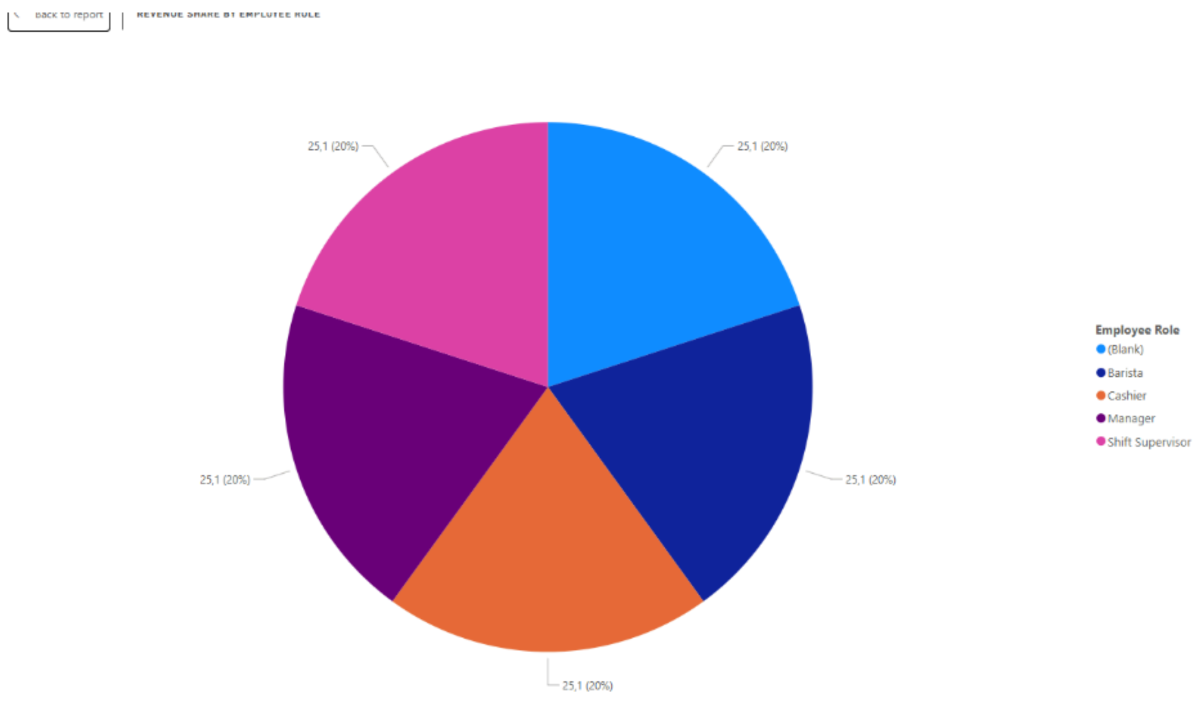
3. Units Ordered by Supplier and Product

Goal: Show units supplied by each supplier per product.



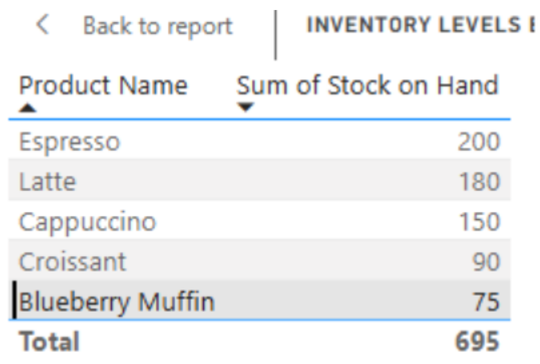
4. Revenue Share by Employee Role

Goal: Show revenue contribution per employee role.



5. Inventory Levels by Product and Store

Goal: Show current stock per product and store.



Product Name	Sum of Stock on Hand
Espresso	200
Latte	180
Cappuccino	150
Croissant	90
Blueberry Muffin	75
Total	**695**