TECHNICAL REPORT

GROCERY SALES

Sri Sandeep Sakthivel – 20065749

Chittalummoodu Biju Bimal – 20059265

Harsha Choudhary – 20044005

Guided By

Bernie Lydon

TABLE OF CONTENTS

1. INTRODUCTION 3

1.2 Reasons for Selecting the Subject Area and Data 3
1.3 Vision and Goals 3
1.4 Key Stakeholders 4
1.5 Business and Technical Requirements 4

2. Data Source and Profile 5

3. SCHEMA 6

3.1 Star Schema 6
3.2 Description of the Grocery Sales Star Schema 7

3. ETL (Extract, Transform, Load) Process 10

3.1 Populating the Product Dimension (DimProduct) 10
3.2 Populating the Customer Dimension (DimCustomer) 11
3.3 Populating the Employee Dimension (DimEmployee) 12
3.4 Populating the Sales Fact Table (FactSales) 13
3.5 ETL Data Quality and Performance Considerations 15-

4. VISUALISATION AND REPORTS 15

4.1 Dashboard Summary: U.S. Sales Performance & Discount Strategy 15

Appendix A: Data Warehouse Development Scripts 17
A.1 Date Dimension (DimDate) Scripts 17

1. INTRODUCTION

In today's competitive retail landscape, the ability to transform vast volumes of transactional data into actionable intelligence is critical. This project undertakes the end-to-end development of a comprehensive business intelligence solution for a grocery sales environment. The core of this project is the design and implementation of a robust data warehouse using a dimensional model, which serves as a single source of truth.

This report documents the **complete project lifecycle**, beginning with the architectural design of the star schema and the development of an automated Extract, Transform, and Load (ETL) pipeline using SQL Server Integration Services (**SSIS**). To cater to diverse business needs, the solution delivers insights through two distinct platforms: **static**, operational reports built in SQL Server Reporting Services (**SSRS**) and a **dynamic**, interactive dashboard created in **Tableau** for exploratory analysis. Furthermore, this project extends into an evaluative analysis, comparing the traditional relational database model with a modern graph database by implementing and contrasting queries in **SQL** and **Neo4j**.

1.2 Reasons for Selecting the Subject Area and Data

The grocery retail domain was chosen for its practical relevance and analytical complexity. The industry generates high-volume, multi-faceted data daily, making it an ideal candidate for demonstrating a full-stack data warehousing solution. The selected datasets provide a comprehensive view of business operations, including sales, products, customers, and employees. This richness allows for a multi-dimensional analysis that mirrors real-world challenges and provides a robust foundation for showcasing key technical skills in data integration, schema design, ETL development, and multi-platform reporting. The complexity of the inter-related data also makes it a compelling use case for comparing the query performance and structural advantages of relational versus graph database technologies.

1.3 Vision and Goals

The vision for this project is to architect and deploy a complete business intelligence ecosystem that transforms raw operational data into strategic assets. The goal is to demonstrate proficiency across the entire data value chain, from backend data processing to **frontend analytics and technology evaluation**.

Key project goals include:

- **Design a Star Schema Data Warehouse:** To model and implement a robust, scalable dimensional model that integrates disparate data sources into a unified repository.
- **Develop an Automated ETL Pipeline:** To implement reliable data integration using SQL Server Integration Services (SSIS) for cleansing, transforming, and loading data into the warehouse.

- **Deliver Multi-Platform Reports and Visualizations:** To create four distinct operational reports in SSRS for detailed analysis and four interactive visualizations within a Tableau dashboard for exploratory insights.
- **Conduct a Comparative Database Analysis:** To implement a parallel data model in Neo4j and perform a comparative analysis of seven distinct queries in SQL and Cypher (CQL) to evaluate the strengths of relational and graph databases.

1.4 Key Stakeholders

The stakeholders for this comprehensive BI solution encompass both business users who consume the analytics and technical teams responsible for the system's implementation and maintenance.

- **Sales Manager:** The primary business user, who relies on the Tableau dashboard and SSRS reports to monitor team performance, analyse **discount effectiveness**, and **track regional sales**.
- **Marketing Analyst:** Uses the analytical outputs to understand **product category performance**, **assess promotional ROI**, and identify **key customer segments**.
- **IT and Data Engineering Team:** The technical stakeholders responsible for designing the schema, developing and maintaining the SSIS packages, ensuring data quality, and managing the underlying database infrastructure.

1.5 Business and Technical Requirements

The following requirements were established to guide the development of the end-to-end solution, ensuring it is technically sound and delivers business value:

1. **Integrated Single Source of Truth:** The data warehouse must centralize data from product, sales, employee, and customer systems into a unified star schema.
2. **Differentiated Reporting Capabilities:** The solution must provide both paginated, operational reports via SSRS for detailed, static views and a dynamic, interactive Tableau dashboard for high-level, exploratory analysis.
3. **Automated and Reliable Data Processing:** An ETL pipeline using SSIS must be developed to automate the extraction, cleansing, transformation, and loading of data, ensuring high data integrity and timeliness.
4. **Multi-Dimensional Analysis:** The system must support the analysis of sales across key dimensions, including product category, time, customer location, and discount level, across both reporting platforms.
5. **Comparative Technology Evaluation:** The project must include an implementation of the dataset in a Neo4j graph database to critically compare its data retrieval

capabilities against the traditional SQL-based relational model for at least seven distinct queries.

6. **Interactive Exploration:** The Tableau dashboard component must be fully interactive, allowing users to filter the entire view by key dimensions and use cross-filtering between visuals to conduct guided analysis.

2. Data Source and Profile

Source of Data

The data for this data warehouse project was sourced from the **"Grocery Sales Dataset"** available on Kaggle. This is a synthetic dataset designed to emulate real-world transactional data from a retail grocery environment. It is comprised of a collection of interconnected CSV files, each representing a different business entity.

dataset link : https://www.kaggle.com/datasets/andrexibiza/grocery-sales-dataset/data

Description of the Dataset

The dataset is structured as a relational model, distributed across several key files:

- **sales.csv:** The central transactional file, containing individual sales records with details such as SalesID, CustomerID, ProductID, EmployeeID, Quantity, and Discount. This file serves as the primary source for the FactSales table.
- **products.csv & categories.csv:** These files contain descriptive information about the products, including names, prices, and their respective categories. They are the sources for the DimProduct dimension.
- **customers.csv:** Contains demographic and address information for each customer. This file is the source for the DimCustomer dimension.
- **employees.csv:** Contains details about the sales employees, including their names, personal details, and location. This file is the source for the DimEmployee dimension.
- **cities.csv & countries.csv:** These files provide geographic information, which is joined with customer and employee data to enrich the location-based attributes in the dimension tables.

  The synthetic nature of this dataset makes it an ideal resource for an academic project. It provides a clean, well-structured, and sufficiently complex set of interconnected tables necessary to demonstrate the entire data warehousing lifecycle, from ETL and schema implementation to multi-platform reporting and advanced analytics, without the data quality issues often present in real-world operational data.

3. SCHEMA

3.1 Star Schema

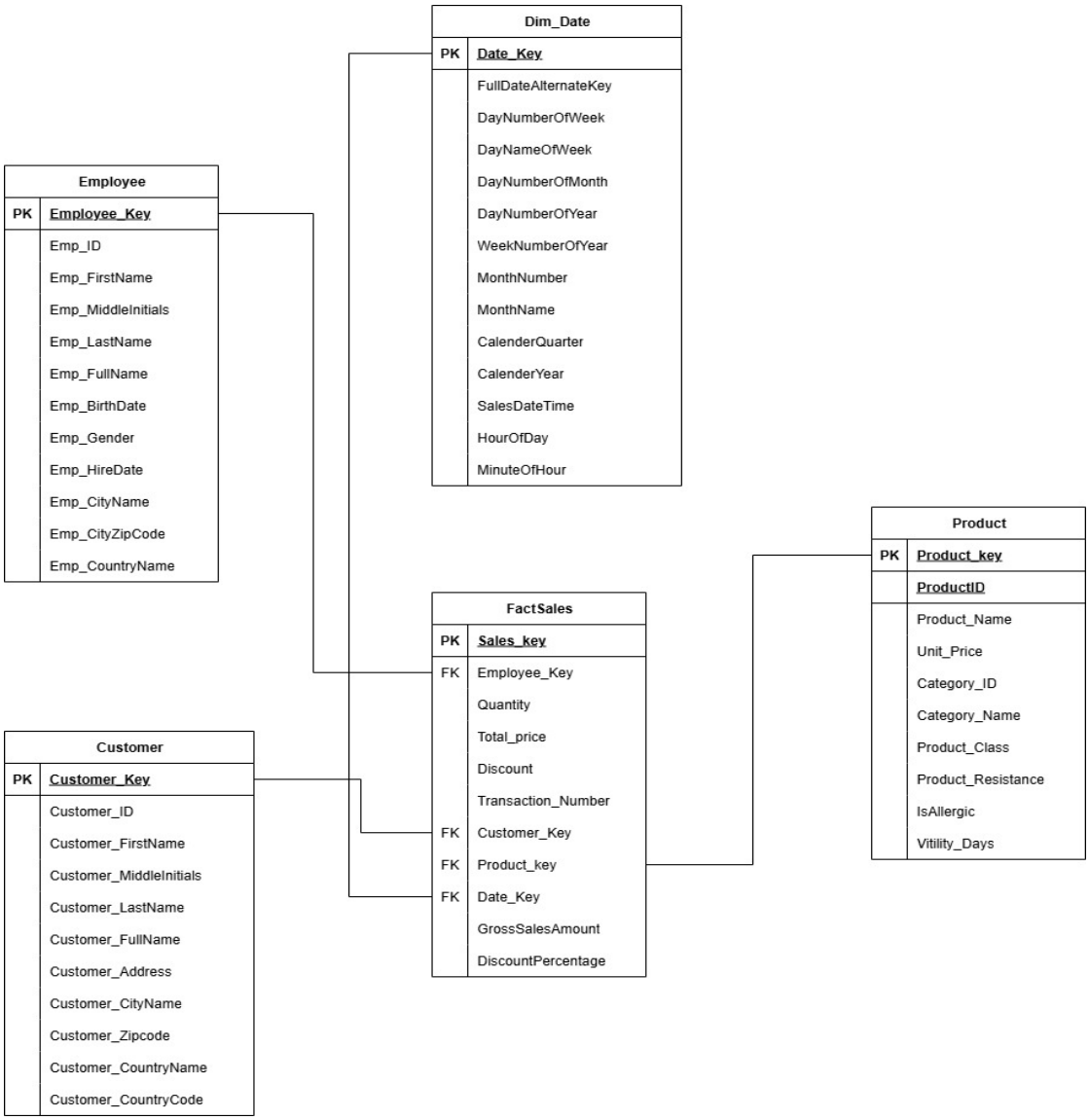

Fig 2.1. Star Schema for Grocery sales

3.2 Description of the Grocery Sales Star Schema

Architectural Overview

The data warehouse is architected using a classic star schema, a dimensional modelling paradigm optimized for high-performance querying and business intelligence. This design is centred around a single, granular fact table, **FactSales**, which is linked to a suite of descriptive, conformed **dimension tables**. The rationale for selecting a star schema is its inherent simplicity and efficiency; by denormalizing descriptive attributes into wide dimension tables, it minimizes complex joins at query time, which is critical for the responsive performance required by analytical tools like Tableau and SSRS.

Core Components

1. FactSales (Transactional Fact Table)

The FactSales table represents the core of the data warehouse, capturing business events at the most atomic level : a single line item within a sales transaction.

- **Grain:** One row per product line item per sale.
- **Purpose:** To store the **quantitative, numeric measures** of the business process. These measures are the primary focus of analysis and aggregation.
- **Key Attributes:**

  Surrogate Keys (FKs):

  Employee_Key, Customer_Key, Product_key, Date_Key are integer-based foreign keys that provide high-performance joins to their respective dimension tables.

  Measures:

  The table includes both base measures (Quantity, Discount) and derived measures (TotalPrice, GrossSalesAmount). These numeric fields are designed for aggregation (e.g., SUM, AVG) to answer key business questions.

  Degenerate Dimension:

  Transaction_Number is included directly within the fact table. It provides contextual grouping for all line items belonging to a single purchase without requiring its own dimension table, thereby **streamlining the model**.

2. Dimension Tables

The dimension tables provide the descriptive context for the facts. They contain the textual attributes that are used for filtering, grouping, and labelling analytical queries, answering the "who, what, where, when, and why" of the business. Each dimension is built around a single primary surrogate key to decouple it from the operational source systems, ensuring historical accuracy and data integrity.

- **DimDate (Date Dimension)**
    - **Purpose:** Provides a rich temporal context for sales analysis. It is a pre-generated, static table that enables sophisticated time-based slicing and dicing (e.g., Month-over-Month, Same-Day-Last-Year comparisons).
    - **Hierarchy:** Year > Quarter > Month > Day.
    - Date identifiers for day, week, month, quarter, and year.
    - Attributes for day names, month names, and time-of-day, to support various aggregation levels (e.g., monthly sales, sales by weekday, hourly patterns).

- **DimProduct (Product Dimension)**
    - **Purpose:** Contains all descriptive attributes related to the products sold. This dimension allows for analysis of sales by product, category, class, and other specific attributes.
    - **Hierarchy:**
        - CategoryName > ProductName.
    - Product_key (PK), ProductID: Identifiers for the product records.
    - Product_Name, Unit_Price, Category_ID, Category_Name: Basic product metadata and association to product categories.

- **DimCustomer (Customer Dimension)**
    - **Purpose:** Stores all relevant attributes of the customer. This dimension is critical for geographic analysis, market segmentation, and understanding customer purchasing behaviour.
    - **Hierarchy:**

      CustomerCountryName > CustomerCityName > CustomerFullName.
    - Customer_Key (PK), Customer_ID: Identifiers for the customers.
    - Full details including names, addresses, city, postcode, country, and country code for market segmentation and geographic analysis.

- **DimEmployee (Employee Dimension)**
    - Purpose: Contains all descriptive attributes for the sales employees. This dimension enables the analysis of sales performance by an **individual** or team.
    - Employee_Key (PK), Emp_ID: Employee identifiers.
    - Employee names (first, middle, last, full), demographics (Emp_Gender, Emp_BirthDate), employment details (Emp_HireDate).
    - Location references such as city and country, supporting geographic drill-down on sales by employee location.

Architectural Rationale and Analytical Capabilities

The design of this star schema is deliberate, intended to provide a robust and scalable foundation for enterprise-level business intelligence.

- **Performance Optimization:** The schema is optimized for read-heavy analytical workloads. It's simple star structure with minimal join paths allows the database query optimizer to execute aggregation queries far more efficiently than it could on a normalized, OLTP-style database.
- **Business Intelligence Enablement:** The model directly supports standard **OLAP** (Online Analytical Processing) operations. Business users can intuitively "slice" the data (e.g., view sales for a single product category), "dice" it (e.g., view sales for a specific category in a specific city), and "drill down" through hierarchies (e.g., from year to month to day).
- **Scalability and Extensibility:** The modular nature of the design allows for straightforward extension. New business processes can be integrated by adding new fact tables, and new analytical perspectives can be added by creating new conformed dimensions (e.g., DimStore, DimPromotion) with minimal disruption to the existing structure.
- **Data Integrity and Governance:** By centralizing all descriptive attributes into conformed dimensions, the warehouse establishes a "**single source of the truth**." For instance, a product's category is defined once in DimProduct and used consistently across all reports and analyses, ensuring data quality and reliable reporting.

In summary, this star schema provides a technically sound and business-aligned foundation, transforming raw transactional data into a powerful asset for strategic analysis.

3. ETL (Extract, Transform, Load) Process

The data warehousing solution employs a series of Extract, Transform, and Load (ETL) processes developed using SQL Server Integration Services (SSIS). These packages are responsible for extracting data from the source CSV files, performing necessary transformations to ensure data quality and integrity, and loading the cleansed data into the target dimensional model in the GROCERYSALES_DW database.

The ETL strategy is designed to be robust and repeatable, populating the dimension tables first to ensure referential integrity, followed by the population of the **central fact table**. Each major table load is encapsulated within its own Data Flow Task for modularity and ease of maintenance.

3.1 Populating the Product Dimension (DimProduct)

The DimProduct package is responsible for integrating product and category information into a single, denormalized dimension table.

Data Flow Breakdown:

1. **Extract**: Data is extracted in parallel from two separate source files: Products Table (products.csv) and Categories Table (categories.csv).
2. **Transform**:
    - Sort: Both data streams are sorted by CategoryID. This is a prerequisite for the Merge Join transformation, which requires sorted inputs to operate efficiently.
    - Merge Join: The two sorted streams are combined using a Merge Join transformation on CategoryID. This enriches the product data with its corresponding category name.
3. **Load**: The final, unified dataset, now containing all product and category attributes, is loaded into the OLE DB Destination which points to the **DimProduct** table in the data warehouse.

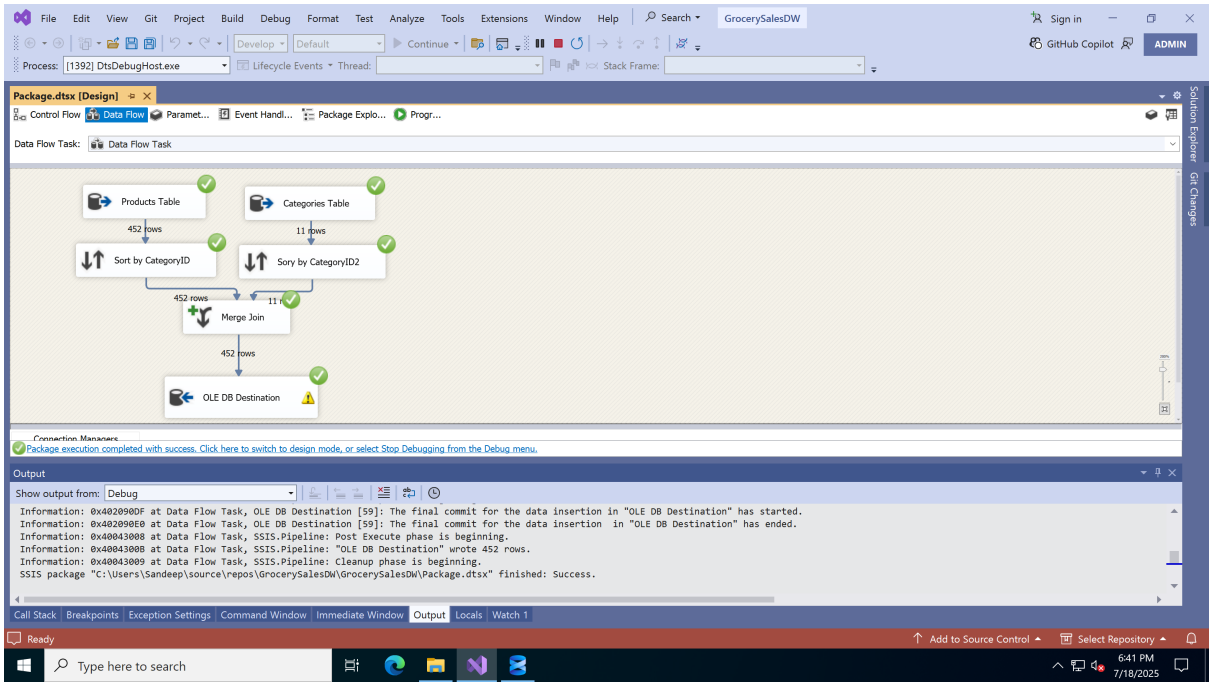

Fig 3.1: SSIS Data Flow for DimProduct

3.2 Populating the Customer Dimension (DimCustomer)

This package enriches customer data with detailed geographic information.

Data Flow Breakdown:

1. **Extract**: The process begins by extracting data from three source files: Customer Table, Cities Table, and Countries Table.
2. **Transform**:
    - Sort: Each of the three data streams is sorted by its respective join key (CityID or CountryID).
    - Merge Join (Customers and Cities): The Customer and Cities data streams are joined on CityID to append city-level details to each customer record.
    - Merge Join (Enriched Customers and Countries): The resulting stream is then joined with the Countries data on CountryID to add the final layer of geographic information.
    - Derived Column: A Derived Column transformation is used to concatenate CustomerFirstName and CustomerLastName to create a CustomerFullName attribute for easier reporting.
3. **Load**: The fully enriched customer data is loaded into the **DimCustomer** destination table.

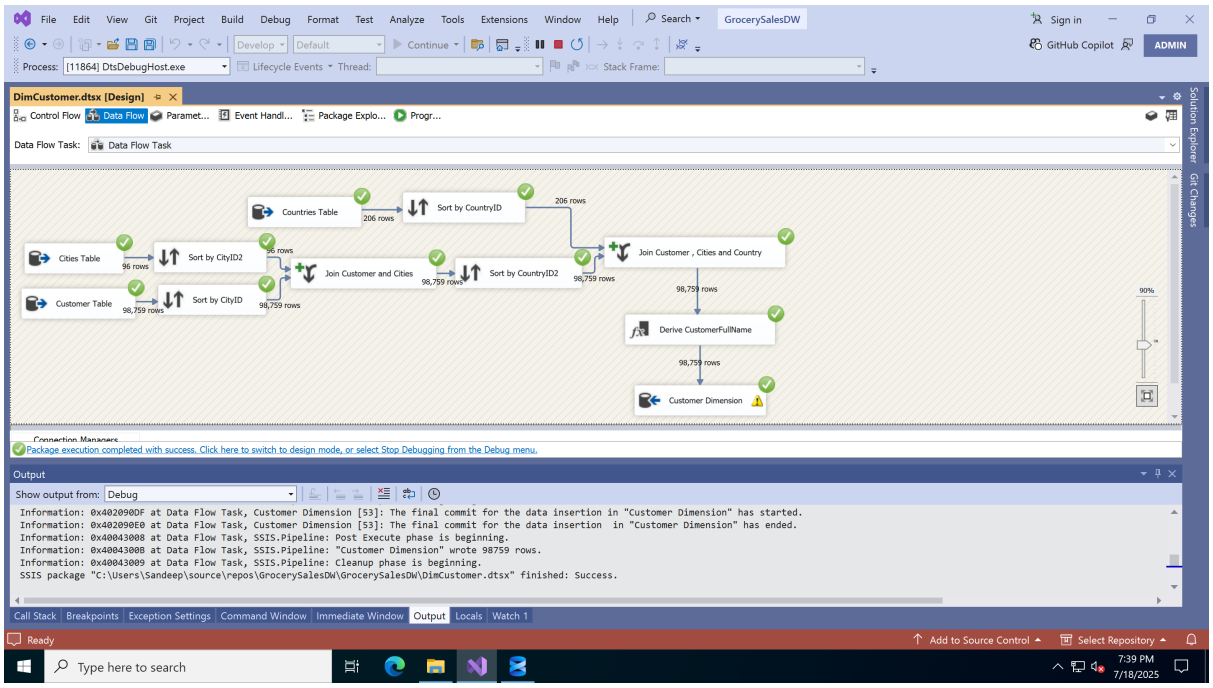

Fig 3.2: SSIS Data Flow for DimCustomer

3.3 Populating the Employee Dimension (DimEmployee)

Similar to the customer dimension, this package integrates employee and geographic data.

Data Flow Breakdown:

1. **Extract**: Data is sourced from the Employee Table, Cities Table, and Countries Table.
2. **Transform**: The process mirrors the customer dimension's logic:
    - Sort & Merge Join: The Employee and Cities data are sorted and joined on CityID.
    - Sort & Merge Join: The resulting stream is sorted and joined with the Countries data on CountryID.
    - Derived Column: A Derived Column transformation creates the EmployeeFullName attribute.
3. **Load**: The final, complete employee records are loaded into the **DimEmployee** destination table.

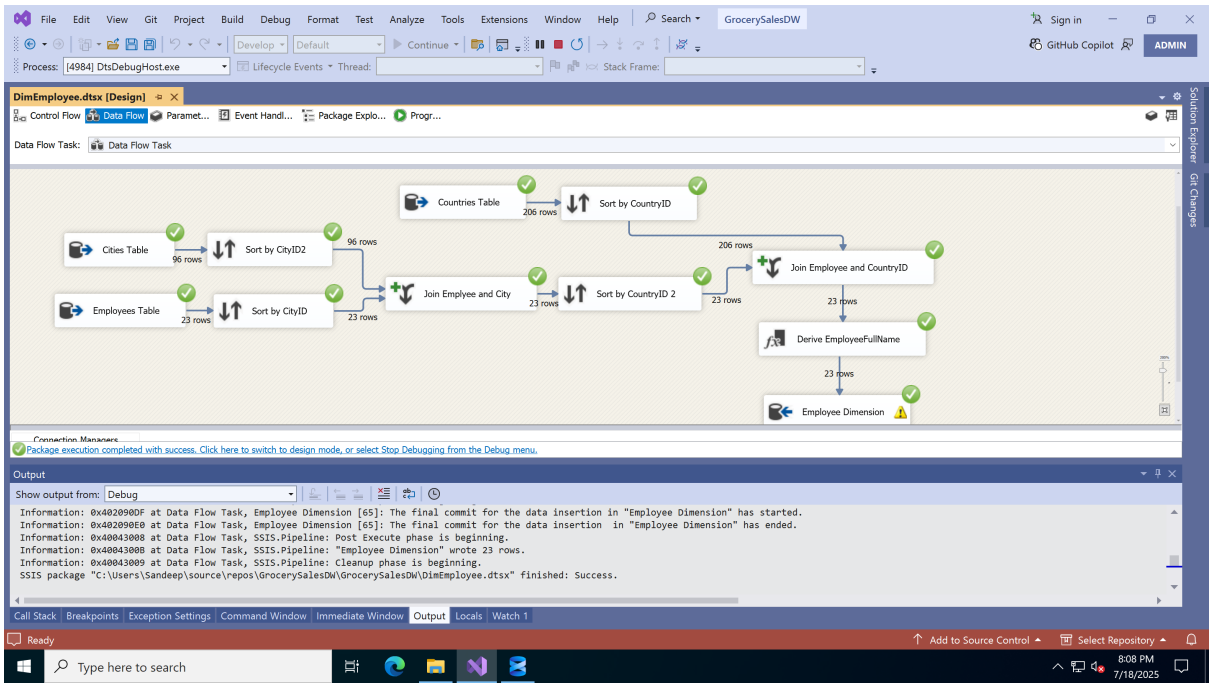

Fig 3.3: SSIS Data Flow for DimEmployee

3.4 Populating the Sales Fact Table (FactSales)

This is the most complex ETL process, as it involves integrating data from the primary sales source file and looking up surrogate keys from the already-populated dimension tables.

Data Flow Breakdown:

1. **Extract**: The process starts by extracting all records from the Sales Table (sales.csv).
2. **Transform**:
    - **Data Cleansing: An initial step (Remove Invalid Errors) could be a conditional split or error handling task to filter out rows with invalid or incomplete data before processing.**
    - Surrogate Key Lookups: This is the critical step. The data flows through a series of Lookup transformations. Each lookup takes a business key from the sales data (ProductID, CustomerID) and finds the corresponding integer-based surrogate key (e.g., ProductKey, CustomerKey) from its respective dimension table (DimProduct, DimCustomer, etc.). This

replaces business keys with the more efficient surrogate keys required by the star schema.

- Derived Columns: Several Derived Column transformations are used to perform calculations and create new measures on-the-fly, such as GrossSalesAmount, TotalPrice, and DiscountPercentage, based on the source data.

3. **Load**: The final, fully transformed data stream, now containing the correct surrogate keys and all calculated measures is loaded into the **FactTableDW** destination.

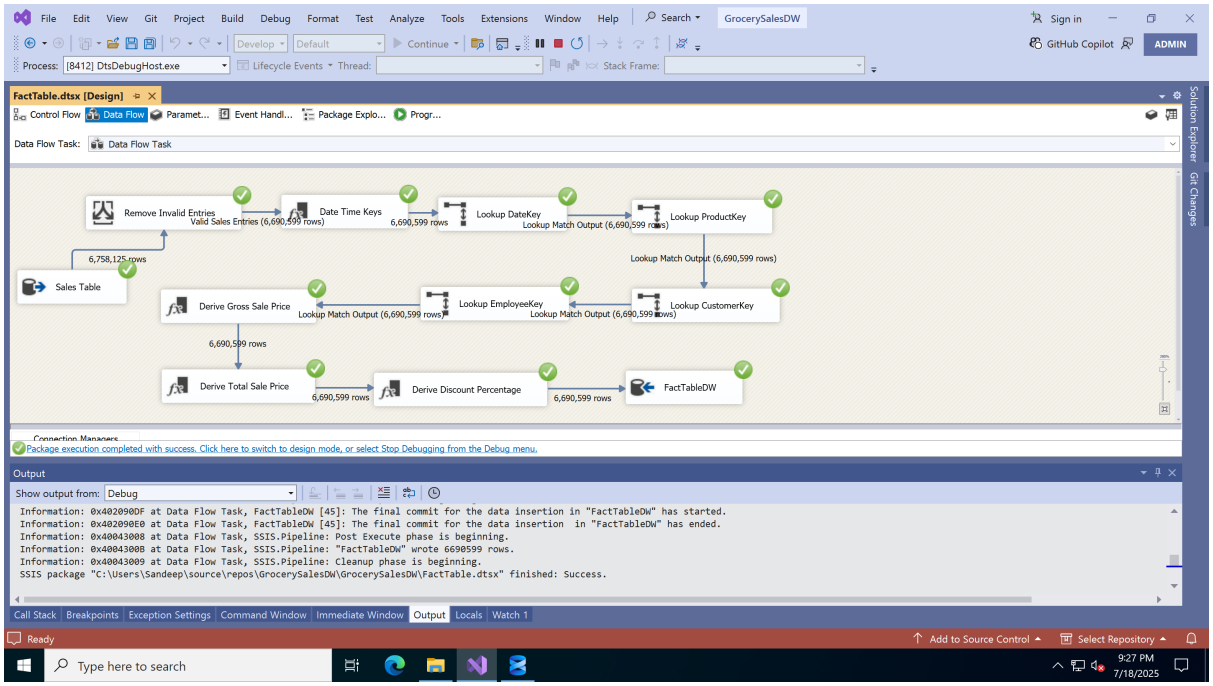

Fig 3.4: SSIS Data Flow for FactSales

3.5 ETL Data Quality and Performance Considerations

During the development of the SSIS packages, several data quality and performance challenges were identified in the source dataset. Addressing these issues was critical to ensure the integrity, accuracy, and usability of the final data warehouse. The following section details the key considerations and the solutions implemented within the ETL data flows.

- Handling Missing Values: The SalesDate column in the source sales.csv file was found to contain a number of NULL values. To prevent these records from being discarded and causing data loss, a conditional split or error handling transformation was implemented. This step isolates rows with null dates, allowing them to be either filtered out or redirected to a logging table for review, ensuring that only valid, date-stamped transactions populate the fact table.

- Data Type Alignment: A common challenge encountered was the mismatch between data types in the flat-file sources and the destination SQL Server tables. For instance, there were discrepancies in integer sizes and string types. Careful configuration of the data conversion transformations within the SSIS Data Flow tasks was necessary to explicitly manage these mappings, preventing data truncation errors and ensuring seamless data loading.
- Performance Optimization and Data Scaling for SSRS: The primary source file, sales.csv, contained approximately 6.6 million rows. While the full dataset was loaded into the data warehouse for comprehensive analysis in Tableau, this volume proved too large for performant report generation within SQL Server Reporting Services (SSRS) for this project's scope. To ensure the SSRS reports were functional and responsive, a pragmatic data scaling strategy was implemented. A separate, smaller data source was created for SSRS by taking a representative subset of 5,000 rows from the sales data. This allowed for the successful development and demonstration of the SSRS reporting requirements without being hindered by processing delays.

4. VISUALISATION AND REPORTS

4.1 Dashboard Summary: U.S. Sales Performance & Discount Strategy

The U.S. Sales Performance & Discount Strategy Dashboard is an interactive analytical tool developed in Tableau, designed to provide a comprehensive overview of grocery sales performance for management stakeholders. It leverages a dimensional model from the **GROCERYSALES_DW** data warehouse to move beyond static reporting and enable **dynamic**, data-driven decision-making.

The dashboard is comprised of four key visualizations that work together to tell a cohesive story:

1. Executive KPI Scorecards: At the top of the dashboard, a series of Key Performance Indicators (KPIs) provides an immediate, at-a-glance summary of core business metrics, including Total Sales, Quantity Sold, and Total Transactions.
2. Discount Strategy Effectiveness Chart: This combination chart directly analyses the financial impact of promotional activities. It correlates the total gross sales (bars) with the average order value (line) for each discount category (No Discount, 10%, 20%), allowing managers to assess if higher discounts are effectively driving revenue or simply eroding margins.
3. Geographic Sales Map: A proportional symbol map visualizes sales performance across U.S. cities. The size of each bubble represents the total sales volume, while

the colour indicates the average order value. This provides immediate insight into which markets are not only large but also have high-value customers.

4. Employee Performance Leaderboard: This bar chart ranks employees by total sales and visually compares their performance against a dynamic sales target.

The true analytical power of the dashboard lies in its interactivity. Using a Product Category filter and a Sales Target parameter, a manager can **customize** the view to answer specific questions. Furthermore, Dashboard **Action Filters** create a guided analytical path; for example, clicking on the "20% Discount" category instantly filters the map and employee leaderboard to reveal which regions and sales representatives are most successful with that specific promotion.

In essence, this dashboard empowers stakeholders to seamlessly transition from what is happening in their business to why, enabling them to optimize marketing strategy, manage team performance, and ultimately drive profitability.

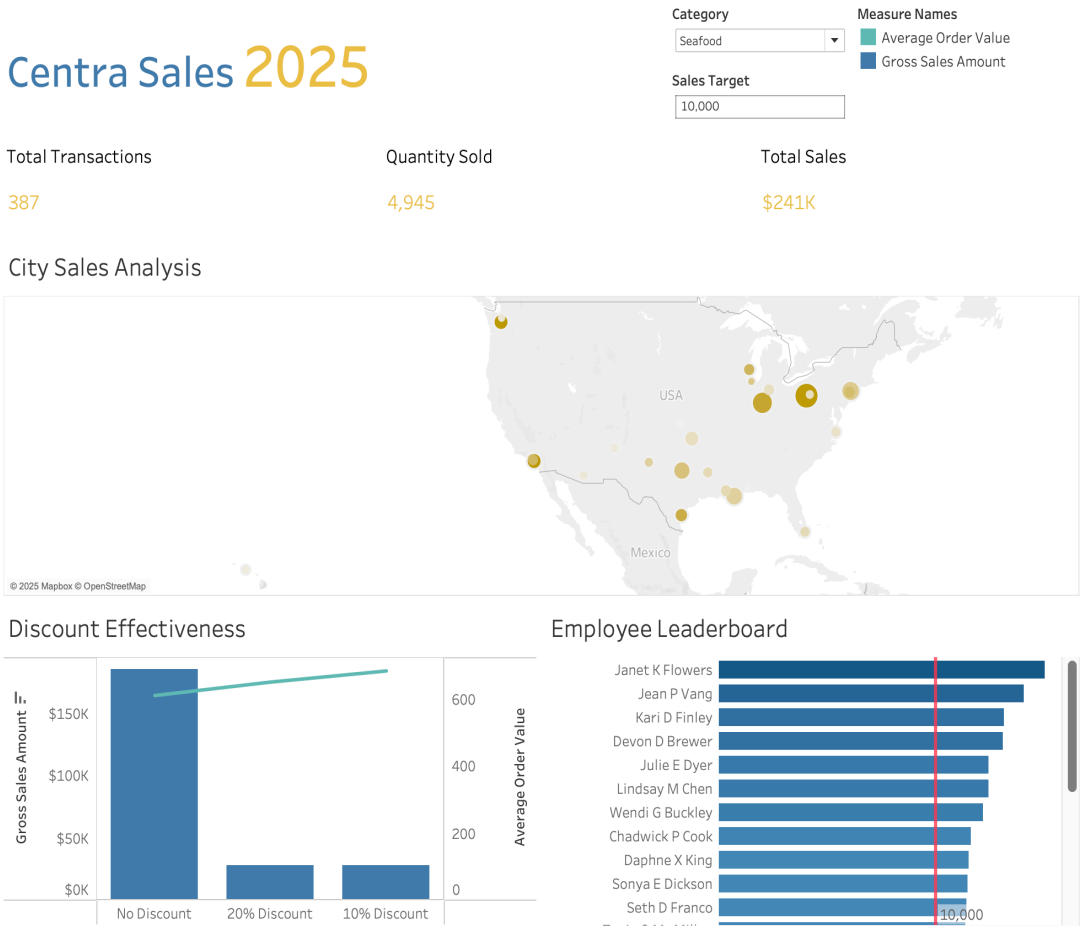

Fig 4.1. Dashboard on Grocery sales

Appendix A: Data Warehouse Development Scripts

A.1 Date Dimension (DimDate) Scripts

The DimDate table is a critical component of the data warehouse, enabling rich time-based analysis. Unlike other dimensions populated via SSIS packages, the date dimension is typically generated and loaded once using a T-SQL script, as its values are static and predictable.

1. CREATE TABLE Script for DimDate

This Data Definition Language (DDL) script defines the structure for the DimDate table. It includes a **primary key (DateKey)** and a comprehensive set of denormalized attributes such as day, week, month, quarter, and year, which are designed to support efficient slicing and dicing of fact data without requiring complex date calculations at query time.

```
USE GrocerySalesDW;
GO
CREATE TABLE DimDate (
    DateKey INT PRIMARY KEY,
    FullDateAlternateKey DATE NOT NULL,
    DayNumberOfWeek INT NOT NULL,
    DayNameOfWeek VARCHAR(10) NOT NULL,
    DayNumberOfMonth INT NOT NULL,
    DayNumberOfYear INT NOT NULL,
    WeekNumberOfYear INT NOT NULL,
    MonthNumber INT NOT NULL,
    MonthName VARCHAR(10) NOT NULL,
    CalendarQuarter INT NOT NULL,
    CalendarYear INT NOT NULL,
    SalesDateTime DATETIME NULL,
    HourOfDay INT NULL,
    MinuteOfHour INT NULL
);
```

2. INSERT Script for Populating DimDate

This script populates the DimDate table with records for a specified date range. It uses a recursive Common Table Expression (CTE) named DateSeries to generate a continuous sequence of dates. For each date generated, it calculates and inserts the various date-part attributes (e.g., DayNameOfWeek, MonthNumber) into the DimDate table, ensuring a **complete** and accurate temporal dimension for analysis**.**

```
USE GrocerySalesDW;
GO

TRUNCATE TABLE DimDate;
GO

DECLARE @StartDate DATE = '2018-01-01';
DECLARE @EndDate DATE = '2018-05-10';

WITH DateSeries AS (
  SELECT @StartDate AS DateValue
  UNION ALL
  SELECT DATEADD(day, 1, DateValue)
  FROM DateSeries
  WHERE DATEADD(day, 1, DateValue) <= @EndDate
)
INSERT INTO DimDate (
  DateKey,
  FullDateAlternateKey,
  DayNumberOfWeek,
  DayNameOfWeek,
  DayNumberOfMonth,
  DayNumberOfYear,
  WeekNumberOfYear,
  MonthNumber,
  MonthName,
  CalendarQuarter,
```

```
    CalendarYear,
    SalesDateTime,
    HourOfDay,
    MinuteOfHour
)
SELECT
    CAST(FORMAT(DateValue, 'yyyyMMdd') AS INT) AS DateKey,
    DateValue AS FullDateAlternateKey,
    DATEPART(weekday, DateValue) AS DayNumberOfWeek,
    DATENAME(weekday, DateValue) AS DayNameOfWeek,
    DAY(DateValue) AS DayNumberOfMonth,
    DATEPART(dayofyear, DateValue) AS DayNumberOfYear,
    DATEPART(wk, DateValue) AS WeekNumberOfYear,
    MONTH(DateValue) AS MonthNumber,
    DATENAME(month, DateValue) AS MonthName,
    DATEPART(quarter, DateValue) AS CalendarQuarter,
    YEAR(DateValue) AS CalendarYear,
    NULL AS SalesDateTime,
    NULL AS HourOfDay,
    NULL AS MinuteOfHour
FROM DateSeries
SELECT * FROM DimDate ORDER BY DateKey;
```

♦♠♥♣