Design and Implementation of a Data warehouse for a Retail Store with Store-level Data

Report - 4

TEAM MEMBERS:

- Pavan Santosh
- Vaishnavi Peddada
- Sohith Somasani

Introduction

Dominick's Finer Foods (DFF) has a very rich place in Chicago's retail history. It was established by Dominick Di Matteo in 1918. The chain grew from a small neighborhood grocery shop into a huge supermarket brand very rapidly. By 1950, DFF introduced a lot of innovations, involving frozen food and delis within their stores, providing well to the changing needs of consumers and grocery store goers. Over the following many years, DFF made its position stone set as a leader within Chicago's grocery stores, competing very strongly with other chains like Jewel. However, the grocery market became extremely competitive after 1990 and DFF struggled very hard to maintain its market share after the owners changed multiple times.

A critical thing for DFF happened in late 1980s when they partnered with the University of Chicago Booth School of Business. This collaborative effort made sure that some randomized experiments were happening in the stores. These were made for analyzing pricing strategies for each store, shelf/inventory management, and customer buying analysis. This happened mostly between years 1989 and 1994, and the research covered more than 100 Dominick's stores in the Chicago metropolitan area. The data generated from those experiments has more than 3500 UPCs and is data of nine years, it is a very good resource for understanding retail operations nowadays.

The primary objective of this project is for building a data warehouse using this huge dataset to understand and provide best solutions for the business challenges of DFF. A data warehouse will facilitate the analysis of historical sales data while being like a central repository. It can also analyze the demographics of customers, the movement of products, and pricing strategies. This knowledge can be helpful for answering many important business questions about customer preferences, sales patterns across seasons, and effectiveness of promotions. The size, complexity, and varying formats of the dataset though can prove to be big challenges. Data cleaning, integration, and establishment of proper normalized relationships between customer traffic, sales data, and product movement are the most important tasks to ensure that we are able to derive usable knowledge from the data.

This project's main aim is for assisting DFF to make decisions using the data they have so that they can optimize inventory, make better marketing, and increase overall profits. By working on a proper understanding of the factors that influence sales such as demographics, store location, and promotions DFF can make its competitiveness better in the retail market and respond quicker to customer needs.

Understanding the Data

In this project, we are going to build data warehouse using dataset of from Dominick's Finer Foods (DFF) when it was in partnership with University of Chicago's Booth School of Business. This dataset has store-level data about shelf management, pricing, customer traffic, and demographic information collected from a period of more than seven years (from 1989 to 1994) across over 100 stores in the Chicago area. This section aims to provide a nice and proper understanding of the raw data we have, which will act as a foundation for building meaningful understandings and insights to help DFF improve its business strategy.

The Data Structure:

The Dominick's dataset is split into two things: general files and category-specific files. These files are in different different formats: CSV, SAS, manual files, and cover key variables related to sales, products, and customer demographics.

General Files:

Customer Count File: This file has info on store traffic, like number of customers visiting each store and using the coupons. It has daily in-store traffic and includes sales values by department

Demographics File: This file has store-level census data for Chicago metropolitan area, which helps in understanding the demographic profiles of customers from different stores. Variables like age, income, people in house, education, and employment are important to analyze how demographics influence sales performance (Thompson, 1994. "Data Warehousing for Retail," vol. 5, 1994).

Category-Specific Files:

UPC Files: The Universal Product Code (UPC) files provide detailed product level information like product name, size, and commodity code. Each UPC is linked to specific product categories which makes it easier to track the performance of different items in the stores.

Movement Files: The movement files keep a track of weekly sales data for each UPC in each store. These files contain variables like price, units sold, profit margin, and promotional details ("Changing Market Dynamics").

Key Variables in the Dataset:

The dataset has a lot of variables, all of them play important roles in analyzing sales by store, customer behavior, and performance of each product:

Date and Week Number: These variables show the time-period in which sales were recorded. They can be useful for time-series analysis to track trends like seasonal fluctuations and promotional effectiveness.

Store Code: It's a unique identifier for every DFF store, this code is used for differentiation between different locations, for comparative analysis across stores.

UPC and Product Information: UPC codes identify each product sold in the dff stores uniquely, and the related variables like product size and commodity code are for additional detailing.

Price and Sales Data: The dataset includes weekly sales figures, retail prices, and gross margins. This data is vital for understanding product profitability and how pricing strategies affect consumer purchasing decisions.

Demographics: Demographic data is for specific stores and includes age distribution, income levels, education, and household size. This data is important for identifying how different customer segments influence sales patterns (Thompson, 1994. "Retail Analytics and Decision-Making," vol. 2, 1994).

Data Cleaning and Challenges:

Before the data can be used properly, it has to undergo a lot of cleaning. Too many challenges need to be addressed:

Data Quality Issues: This dataset contains a lot of inconsistencies, such as missing values and discrepancies between product codes. These issues must be corrected to for proper analysis.

Data Integration: Since data is stored in different files, one of the primary tasks is to integrate these files into an unified data warehouse.Primary keys such as UPC codes and store codes will be used to link data amongst the different files and categories.

Once cleaned and integrated, this dataset will form the basis for answering business questions and developing strategies to improve DFF's operations. The data warehouse will provide insights about the pricing strategies, customer behavior, and store performance, for finally helping DFF make informed, data-driven decisions to optimize sales and profitability.

Metadata Description

The metadata in the Dominick's Finer Foods (DFF) dataset contains several files whose metadata describe the structure, meaning, and type of data stored in each field, to make it easy to work with the dataset. Here's a breakdown of the key files and their associated metadata:

Customer Count File:

The Customer Count file has info on the no. of customers who visit each store regularly. This file also has data on total sales per day and the number of coupons redeemed. The fields in this file are:

- **DATE:** Date of the observation (formatted as YYYY-MM-DD).
- **WEEK:** The week number corresponding to the sales data.
- **STORE:** Unique store code.
- **CUSTCOUN:** The count of customers visiting the store.
- **COUPONS:** The number of coupons redeemed by customers at the store.

Demographics File:

The Demographics file has store related demographic information based on U.S. government census data. This file includes key demographic variables that can be linked to store-level data to analyze customer behavior patterns based on factors such as income, age, and household size. The fields are:

- **AGE9:** Percentage of the population under age 9.
- **AGE60:** Percentage of the population over age 60.
- **INCOME:** Logarithm of the median household income in the store's area.
- **HHSIZE:** Average household size.
- **EDUCATION:** Percentage of college graduates in the area.

UPC Files:

The Universal Product Code (UPC) files has detailed product-level information for each category of goods sold at DFF stores. This data is for tracking product sales, price changes, and profitability. Fields are:

- **UPC:** The unique identifier for each product, consisting of a series of numbers.
- **COM_CODE:** The commodity code assigned to each product category.
- **NITEM:** Dominick's internal item code for tracking products.
- **DESCRIP:** A brief description of the product.

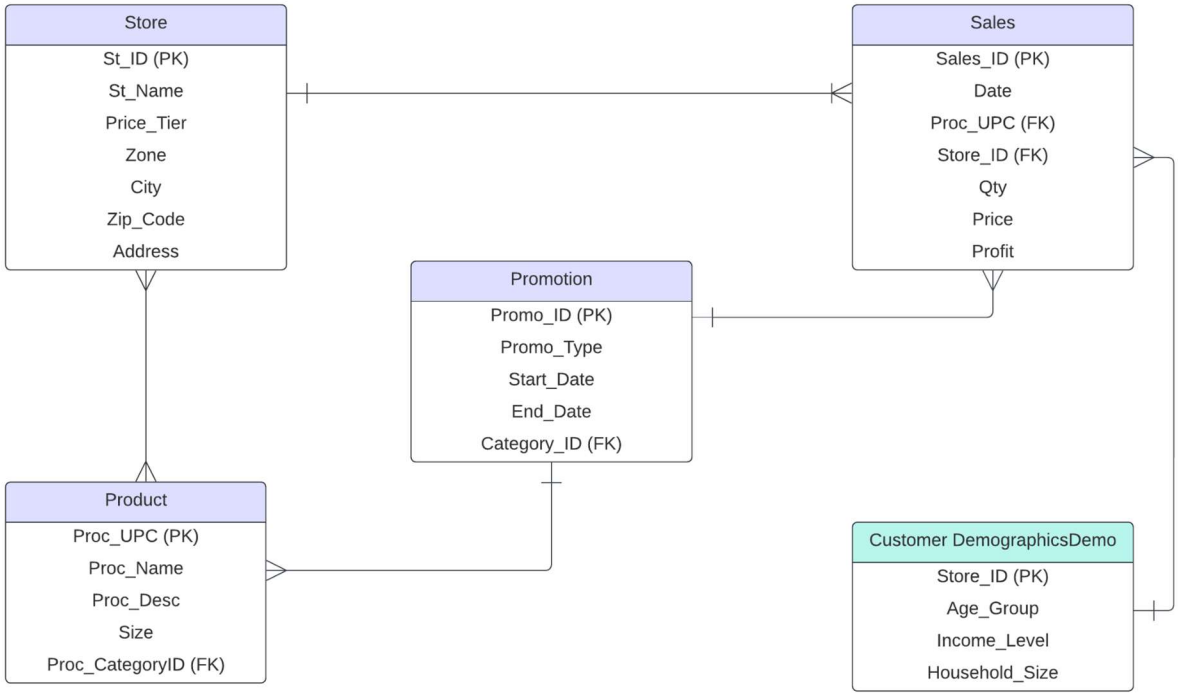
Movement Files:

The Movement files track weekly sales for each UPC in all DFF stores. These files contain variables that are for doing detailed analysis about product movement, pricing, and promotions. The fields include:

- **UPC:** The UPC code for the product.
- **STORE:** The unique identifier for each store.
- **WEEK:** The week number for the sales data.

- **MOVE:** The number of units sold during the week.
- **PRICE:** The retail price of the product.
- **PROFIT:** The gross margin percentage.
- **SALE_CODE:** A code indicating whether the product was on sale (B = Bonus Buy, C = Coupon, S = Price Reduction) (Thompson, 1994. “Retail Analytics and Decision-Making,” vol. 2, 1994).

Entity Relationship Diagram (ERD)



Subject Area Understanding

To address the business challenges faced by Dominick's Finer Foods (DFF) effectively, it's essential to deeply understand the retail grocery industry. By examining this sector thoroughly, we can gain insights into DFF's business environment and identify relevant issues and solutions. To make the most of the dataset collected through DFF's collaboration with the University of Chicago Booth School of Business, we also need to have a strong grasp of both the retail landscape and customer behavior. This will help us leverage the data's full potential to develop meaningful business strategies.

Seasonal Trends and Pricing Decisions:

Seasonal trends are a crucial part of retail strategy, especially in the grocery business, where customer demand can vary significantly throughout the year. For Dominick's Finer Foods (DFF), managing pricing during high-demand seasons like Thanksgiving and Christmas is key to maximizing revenue. Traditionally, sales of holiday-related products such as baked goods, meats, and beverages surge during these times. DFF's challenge is to find the right pricing balance that attracts cost-conscious customers while still ensuring healthy profit margins during these peak periods (Sen, 1993. Retail Analytics, vol. 9, 1993).

Promotional Effectiveness:

In the supermarket industry, promotions play a major role in driving sales. Dominick's Finer Foods (DFF) uses various promotional tactics such as price cuts, bonus buys, and coupons to encourage customers to make purchases. However, not all promotions have the same impact, so it's crucial for DFF to understand which strategies are most effective to get the best return on their marketing efforts.

DFF's dataset contains detailed weekly sales records, including information on whether a product was part of a promotion, like a price discount. By analyzing this data, DFF can identify which promotional methods generate the highest sales boost for different product categories. For example, discounts might work better for high-margin items like premium cuts of meat, while price reductions could be more effective for fast-selling items like drinks or snacks. This insight allows DFF to optimize their promotional strategies and maximize sales.

Store Location and Customer Traffic:

Store location is one of the key factors influencing customer traffic and sales performance in the grocery business. Dominick's Finer Foods (DFF) operates stores in both suburban and urban areas, each serving a distinct customer base with different shopping habits. Urban stores, for example, tend to have higher foot traffic due to their proximity to densely populated areas, while suburban stores often see fewer visits but larger basket sizes, as customers tend to shop less frequently and buy more per visit (Thompson, 1994. *Retail Analytics and Decision-Making*, vol. 2, 1994).

DFF can use demographic data from their dataset to explore how customer behavior varies by store location. By analyzing factors like income levels, employment status, and household size, DFF can gain insights into how these demographics influence shopping patterns at different stores. For instance, urban stores might focus on smaller, more affordable products suited for single-person households or young professionals, while suburban stores in wealthier neighborhoods may see higher demand for premium or organic items (Sen, 1993. *Retail Analytics*, vol. 9, 1993).

Impact of Demographics on Sales:

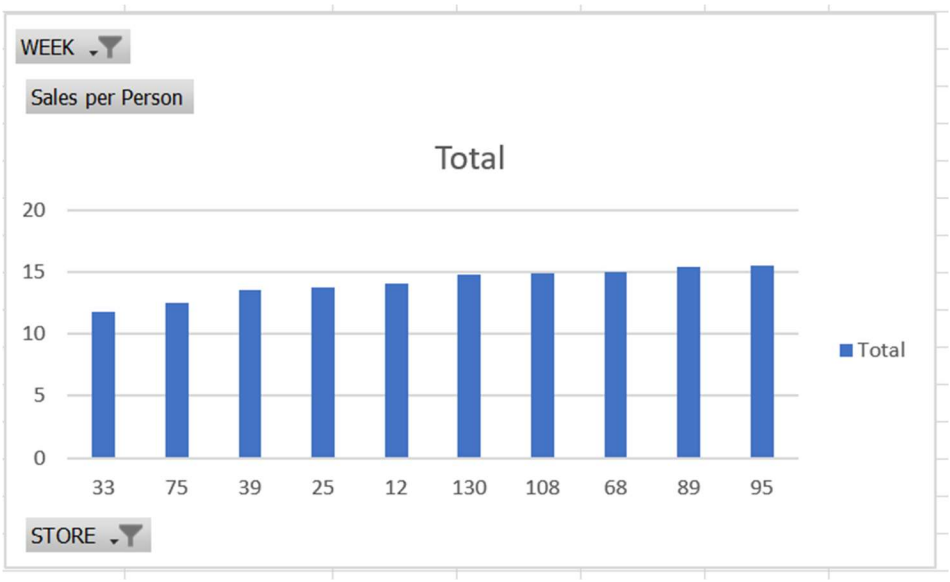
Customer demographics play a major role in shaping purchasing behavior. The demographic data in DFF's dataset, including details like age, household size, income, and education level, provides valuable insights into the preferences and buying habits of different customer groups. By understanding these factors, DFF can tailor its product offerings to better meet the needs of its customers. For example, stores located in wealthier areas might focus on premium or organic products, while those in neighborhoods with many young families may prioritize bulk items and family-sized products (Sen, 1993. Changing Market Dynamics, vol. 4, 1993).

Business Questions

1. Which stores have the highest customer traffic, and how does customer traffic correlate with sales?

- **Importance:** Identifying stores with the highest customer traffic and analyzing how it correlates with sales is crucial for DFF to assess store performance and improve conversion rates. High foot traffic stores that aren't translating traffic into sales may signal issues such as poor product assortment, ineffective promotions, or suboptimal customer service. By understanding these dynamics, DFF can implement targeted marketing campaigns, refine store layouts, or adjust product offerings to convert more visitors into buyers. This analysis helps allocate resources more effectively, ensuring that stores maximize their revenue potential by capitalizing on high customer traffic.

Row Labels	Sales per Person
33	11.8094888
75	12.51191902
39	13.5907608
25	13.71604597
12	14.10689771
130	14.75867211
108	14.9182158
68	14.96516726
89	15.41106498
95	15.54063643
Grand Total	**14.0390323**

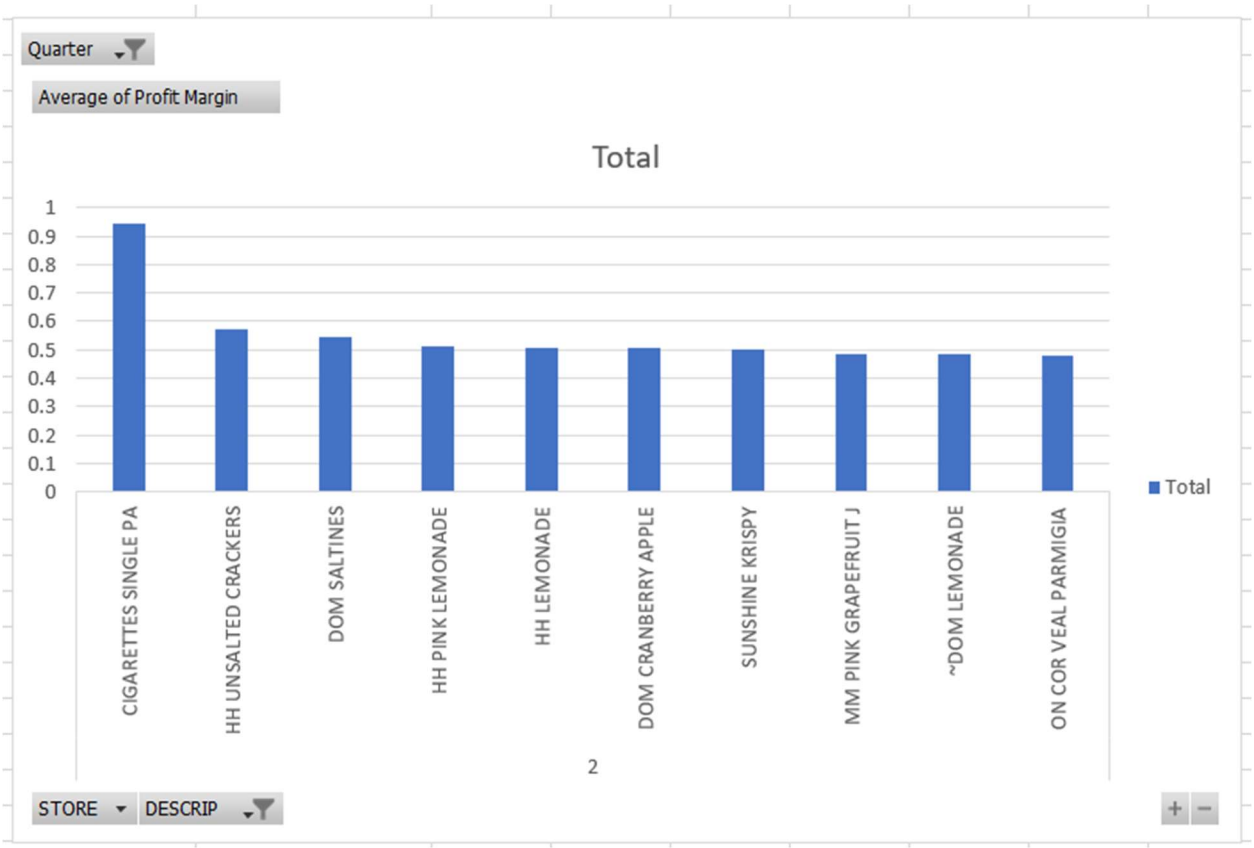


- **Result Analysis:** Store 33 is the worst performance store in terms of sales per customer followed by 75 and 39
- **Sample Considerations:** The table only considers a few products and only the top 10 worst performing stores are displayed.

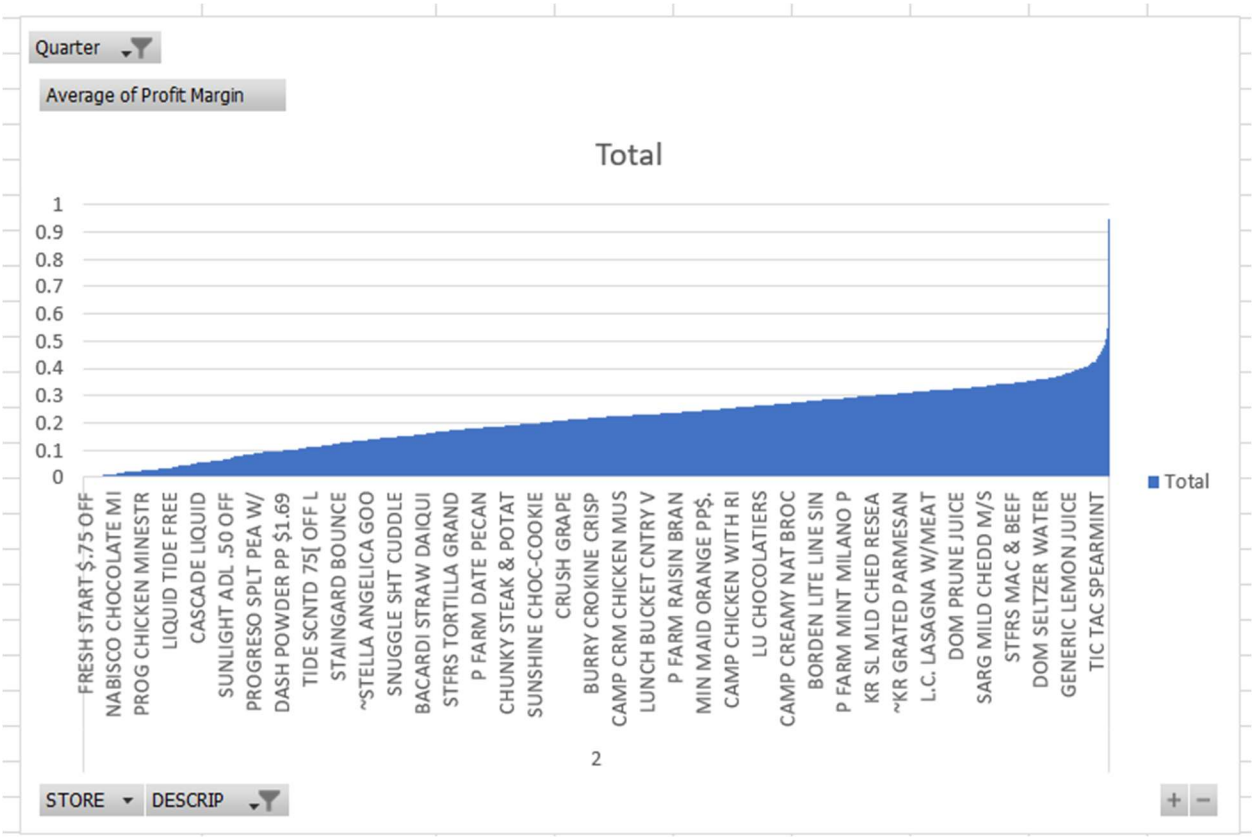
2. Which products generate the highest profit margins, and how consistent are these margins across different stores and time periods?

- **Importance:** Identifying products that generate the highest profit margins is crucial for Dominick's Finer Foods (DFF) as it enables the company to prioritize and strategically promote these highly profitable items, thereby enhancing overall profitability. Focusing on high-margin products allows DFF to allocate marketing resources more effectively, optimize shelf space, and tailor inventory management to ensure consistent availability of these lucrative offerings across different stores and time periods. Conversely, analyzing products with low profit margins is equally important, as it helps DFF identify items that may be underperforming or contributing minimally to the bottom line. By understanding the factors driving low margins—such as high costs, low sales volume, or intense competition—DFF can make informed decisions to either improve the profitability of these products through cost reduction or strategic pricing or consider discontinuing them to free up resources for more profitable alternatives. This comprehensive approach ensures that DFF not only maximizes revenue from high-margin products but also maintains a balanced and efficient product portfolio, ultimately driving sustained financial success and competitive advantage in the retail market.

Row Labels	Average of Profit Margin
2	**0.554463462**
CIGARETTES SINGLE PA	0.947032692
HH UNSALTED CRACKERS	0.569857692
DOM SALTINES	0.54675
HH PINK LEMONADE	0.514330769
HH LEMONADE	0.506832692
DOM CRANBERRY APPLE	0.505534615
SUNSHINE KRISPY	0.500155769
MM PINK GRAPEFRUIT J	0.486596154
~DOM LEMONADE	0.485315385
ON COR VEAL PARMIGIA	0.482228846
Grand Total	**0.554463462**



Quarter	(Multiple Items)
Row Labels	**Average of Profit Margin**
2	**0.204261233**
FRESH START $.75 OFF	0.001015385
DADS ROOT BEER	0.001476923
BRAND SPRITE	0.003198077
OLD LONDON RYE TOAST	0.003790385
DIET COKE N/R	0.003821154
WESTON VARIETY PACK	0.003846154
~STELLA DORO DIETETI	0.004473077
BUTTERFLY CRISPS REG	0.004540385
ORIG N.Y.EXPRESS ICE	0.004548077
OLD LNDON UNSLT SESA	0.004642308

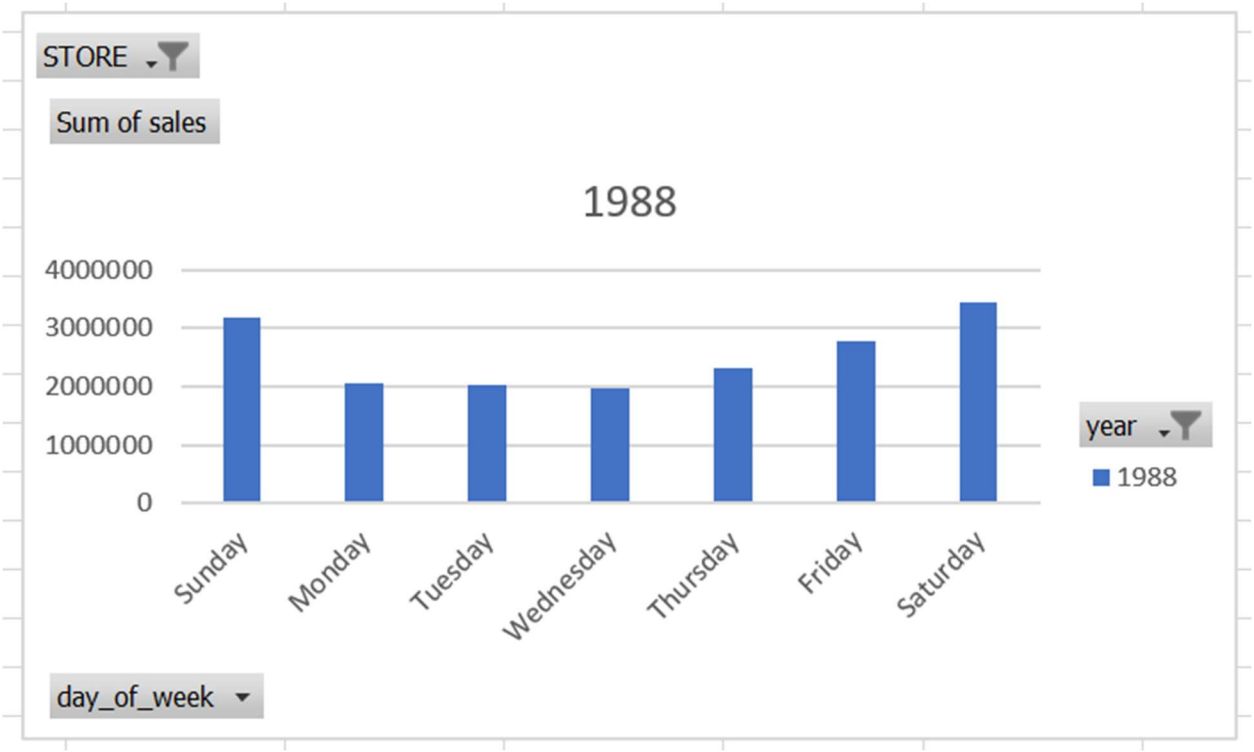


- **Result Analysis:** cigarette single PA is by far the best performing product followed by unsalted crackers and salties. Fresh start, root beer and brand sprite are some of the worse performing products for store 2.
- **Sample Considerations:** The table above contains the product sales data for only Store 2

3. How do customer counts and sales vary by day of the week and by store, and what patterns emerge over time?

- **Importance:** Understanding how customer counts and sales vary by day of the week and by store, along with the emerging patterns over time, is essential for Dominick's Finer Foods (DFF) to optimize its operations and enhance overall efficiency. By analyzing these variations, DFF can accurately forecast peak and off-peak periods, allowing for strategic staffing adjustments that ensure adequate customer service without incurring unnecessary labor costs. Additionally, insights into daily and store-specific sales trends enable more effective inventory management, reducing instances of stockouts or overstocking and ensuring that popular products are readily available when demand is highest. Recognizing temporal patterns also aids in the planning of targeted marketing campaigns and promotional activities, tailored to specific days or seasons that drive higher traffic and sales. Furthermore, understanding store-by-store performance variations facilitates the identification of best practices and areas needing improvement, fostering a data-driven approach to store operations. Ultimately, this comprehensive understanding of customer and sales dynamics empowers DFF to meet customer demand more efficiently, enhance profitability, and maintain a competitive edge in the retail market.

STORE	2
Sum of sales	**Column Labels**
Row Labels	**1988**
Sunday	3183136.82
Monday	2055482.72
Tuesday	2014843.28
Wednesday	1974283.69
Thursday	2328145.65
Friday	2766581.71
Saturday	3441718.95
Grand Total	**17764192.82**

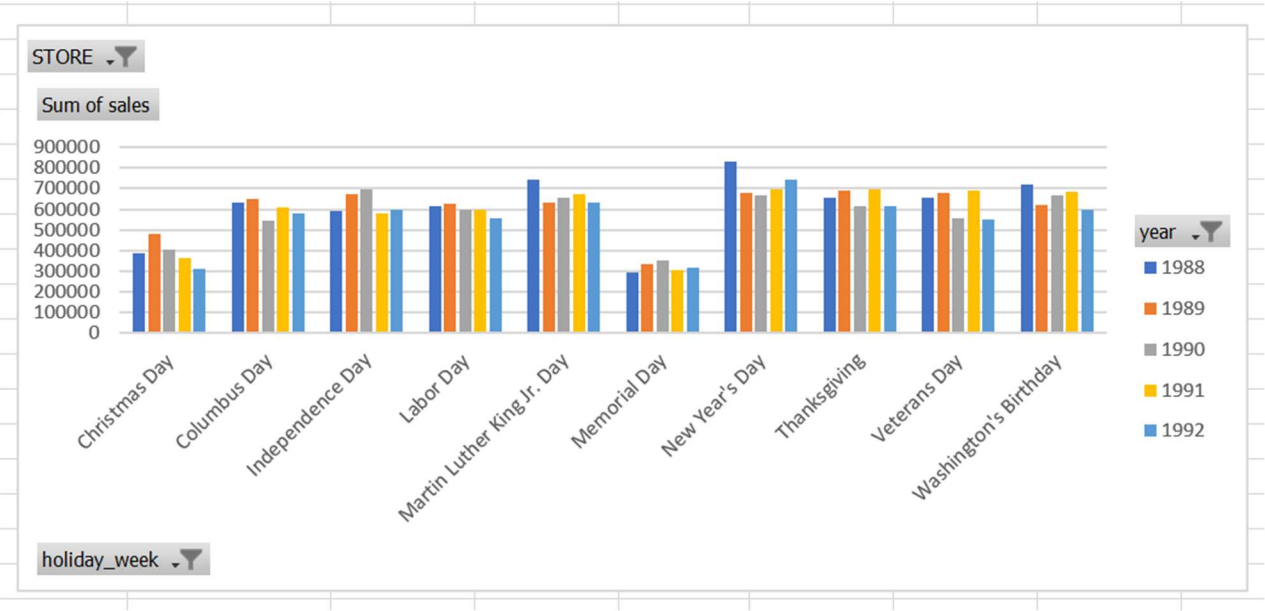


- **Result Analysis:** Sales are high on Saturday and Sunday, the weekend and slows down as the week starts.
- **Sample Considerations:** The table above contains the product sales data for only Store 2

4. How do sales and customer traffic vary across different holiday weeks, and which holidays contribute the most to overall sales performance in each store and department?

- **Importance:** Analyzing how sales and customer traffic fluctuate during different holiday weeks is pivotal for Dominick's Finer Foods (DFF) to strategically enhance its operational and financial performance. By identifying which holidays significantly boost sales and attract more customers, DFF can effectively allocate resources, ensuring optimal staffing levels and adequate inventory to meet heightened demand without incurring unnecessary costs. Understanding these seasonal spikes allows for the tailoring of marketing campaigns and promotional activities that resonate with customers during peak periods, thereby maximizing revenue opportunities. Additionally, recognizing the unique impact of specific holidays on various stores and departments enables DFF to implement localized strategies that cater to distinct market dynamics and customer preferences. This comprehensive insight not only drives efficient store operations and inventory management but also fosters a proactive approach to capitalizing on high-impact holidays, ultimately strengthening DFF's competitive edge and enhancing overall profitability in the retail landscape.

Sum of sales	Column Labels				
Row Labels	1988	1989	1990	1991	1992
Christmas Day	384649.45	479272.97	406880.62	365020.94	311708.49
Columbus Day	630323.29	648351.78	544656.84	611601.61	578070.57
Independence Day	592817.76	670605.67	699199.06	579133	599820.8
Labor Day	612896.66	624616.68	597618.59	598385.76	555997.25
Martin Luther King Jr. Day	740353.37	629753.17	657311.57	670464.84	634931.6
Memorial Day	294913.76	332151.18	351252.68	306952.48	315622.23
New Year's Day	830761.66	679012.45	667541.76	696215.12	744650.44
Thanksgiving	656408.71	690072.63	616169.49	695008	614978.21
Veterans Day	654728.47	676276.04	555417.93	687684.88	553297.61
Washington's Birthday	721200.82	619461.84	669134.21	683020.97	599323.62
Grand Total	**6119053.95**	**6049574.41**	**5765182.75**	**5893487.6**	**5508400.82**

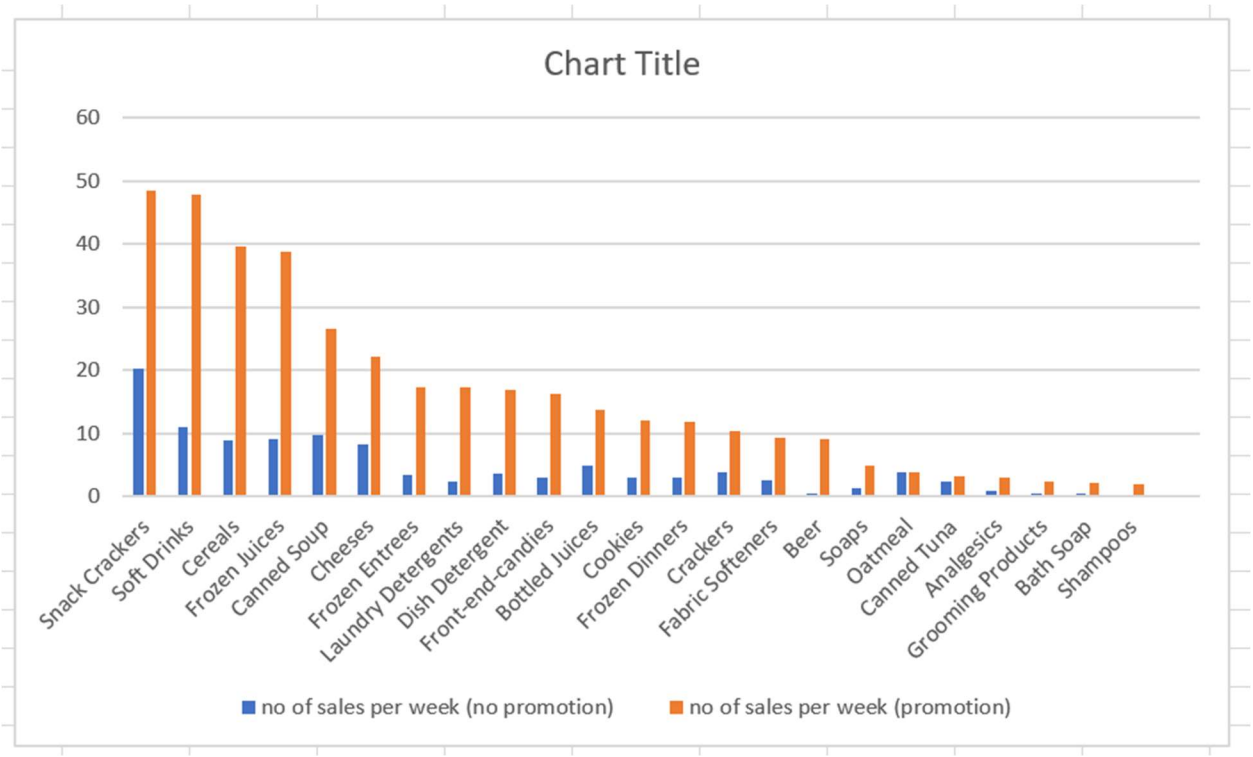


- **Result Analysis:** For Store 2, New Year's Week seems to be the most profitable week and memorial day week the least profitable.
- **Sample Considerations:** Data only considered for Store 2 and for years 1988-1992

5. How do sales and the number of products sold differ between promotion periods and non-promotion periods, and which types of promotions drive the most significant increase in customer purchases?

- **Importance:** Understanding the impact of promotions on sales and the number of products sold is crucial for Dominick's Finer Foods (DFF) to optimize its promotional strategies. By analyzing how sales and customer purchases change during promotional periods compared to non-promotion times, DFF can identify the most effective promotion types that lead to the highest sales growth and increased customer engagement. This insight helps DFF refine its marketing tactics, allocate promotional budgets more efficiently, and tailor discount offerings to maximize sales and profitability. Analyzing the effectiveness of promotions also ensures that DFF can create targeted campaigns that resonate with customers, drive higher traffic to the stores, and ultimately boost overall revenue.

A	B	C	D	E	F	G
Row Labels	Sum of MOVE	Count of WEEK	Sum of MOVE	Count of WEEK	no of sales per week (no promotion)	no of sales per week (promotion)
Snack Crackers	22040	1093	6990	144	20.16468435	48.54166667
Soft Drinks	340483	30895	423368	8859	11.02065059	47.7895925
Cereals	611928	69245	174517	4406	8.837143476	39.60894235
Frozen Juices	284815	30961	216934	5607	9.199153774	38.68985197
Canned Soup	696415	72161	232901	8766	9.650850182	26.56867442
Cheeses	763843	93635	340085	15300	8.157665403	22.22777778
Frozen Entrees	361012	108321	253960	14704	3.332797888	17.27149075
Laundry Detergents	169419	73035	111308	6465	2.319696036	17.21701469
Dish Detergent	149270	40305	66484	3929	3.703510731	16.92135403
Front-end-candies	21412	7405	9422	581	2.891559757	16.21686747
Bottled Juices	292015	59141	146052	10623	4.937606736	13.74865857
Cookies	423912	139299	196141	16296	3.043180497	12.03614384
Frozen Dinners	22484	7296	24240	2034	3.081688596	11.91740413
Crackers	106737	27354	52785	5122	3.902061856	10.30554471
Fabric Softeners	113557	43391	38502	4146	2.617063446	9.286541245
Beer	77	145	18	2	0.531034483	9
Soaps	562	407	93	19	1.380835381	4.894736842
Oatmeal	297	79	50	13	3.759493671	3.846153846



- **Result Analysis:** Snack Crackers tend to sell more when there are promotion offers on it and for products like tuna it doesn't matter if there are any promotional offers.
- **Sample Considerations:** Considered few stores and all the UPCs

Overview of Kimball's Methodology

Kimball's methodology for data warehousing is a widely used approach focused on creating dimensional models that support efficient querying and reporting. This methodology advocates for building the data warehouse in a bottom-up manner, starting with the creation of independent data marts that are later integrated into an enterprise data warehouse. The key steps in Kimball's methodology are as follows:

1. **Business Requirements Definition:**
    - The first step involves understanding the business needs and identifying key questions that the data warehouse must answer. For DFF, the primary focus is on analyzing store-level data related to sales, shelf management, and pricing across multiple branches.
2. **Data Modeling and Design:**
    - Once the business requirements are clear, the design of the data warehouse schema begins. This involves the creation of star schemas where a central fact table (e.g., FactSales in our case) is surrounded by multiple dimension tables (e.g., DimStore and

DimDate). The schema ensures that all relevant data points can be captured and analyzed, such as sales performance, customer traffic, and temporal analysis.

3. **ETL Process Design:**
    - Kimball emphasizes a strong ETL (Extract, Transform, Load) process to ensure data consistency and integrity. For DFF, the data flow is defined by mapping tables that outline the process of loading data from source files to staging tables and then from staging tables to data marts. These mappings involve copying and transforming data to fit the dimensional model.
4. **Data Mart Development:**
    - Independent data marts are designed to support specific areas of analysis, each of which is built as a self-contained structure. For DFF, each data mart focuses on different aspects such as store performance, product sales, and time-based analysis, following the dimensional modeling technique.
5. **Conformable Dimensions:**
    - Kimball stresses the importance of creating conformed dimensions, meaning that dimensions like DimDate and DimStore are shared across different data marts to ensure consistency. This allows for integrated querying across the data warehouse while maintaining the modularity of independent data marts.
6. **Iterative Development:**
    - The methodology supports iterative development, where the data warehouse can grow and evolve over time as new business questions emerge or additional data sources are integrated. This is especially important for DFF, as their data warehouse needs to scale with their growing data from multiple branches.

Following Kimball's methodology to create independent conformable data marts is crucial for several reasons:

- **Efficiency in Querying:** The star schema design ensures optimized querying for reporting and analysis by reducing the number of joins required to fetch relevant data. Each data mart is tailored to answer specific business questions, leading to faster performance.
- **Scalability:** As DFF grows and more stores and products are added, the modularity of independent data marts ensures that the data warehouse can expand without having to overhaul the existing architecture. Conformed dimensions make it easy to integrate new marts without data duplication or inconsistency.
- **Flexibility:** With separate data marts, DFF can run focused analyses on specific areas such as store performance or time-based trends, while still allowing for broader analysis across different marts. This flexibility is key in adapting to changing business needs.
- **Consistency:** The use of conformable dimensions ensures that dimensions such as date and store location are consistent across all data marts, maintaining data integrity and accuracy in analysis.

By adhering to Kimball's methodology, the logical design of DFF's data warehouse is both efficient and scalable, enabling comprehensive analysis and reporting to meet current and future business needs.

To efficiently handle all five business questions, the data warehouse design is split into **two separate data warehouses**. **Data Warehouse 1** will address **Questions 3, 5, and 6**, focusing on store performance, sales trends, and customer traffic by leveraging the **FactSales**, **DimStore**, and **DimDate** tables. It enables analysis of metrics like **TotalSalesAmount** and **CustomerCount** to identify traffic patterns and sales correlations over time. **Data Warehouse 2** will handle **Questions 4 and 10**, focusing on product performance and profitability using metrics such as **UnitsSold**, **ProfitMargin**, and **TotalSales**, with the integration of **DimProduct**, **DimStore**, and **DimDate** to provide insights on pricing and promotions. This structure ensures efficient, targeted analysis for each set of business questions.

	Dimensions				
BQs	**Data Marts**	**Product**	**Store**	**Category**	**Time**
3,5,6	Category Sales			x	x
4,10	Product Sales	x	x		x

Category Sales DataMart

❖ Star Schema Design

➢ Dimension Tables :-

- **DimStore -** This table holds information about each store's characteristics, including its location, pricing category, and classification. The primary key, StoreID, uniquely identifies each store, and additional details like City, Price, Tier, and Zone help provide context for the store's operations and demographics.

Column Name	Data Type	Description
StoreID	INT (PK)	Store number (store)
City	VARCHAR(100)	City where the store is located
Price	VARCHAR(50)	Price level/category of the store

Tier	INT	Tier classification of the store
Zone	VARCHAR(50)	Zone designation for the store
ZipCode	INT	Zip code of the store location
Address	VARCHAR(200)	Physical address of the store

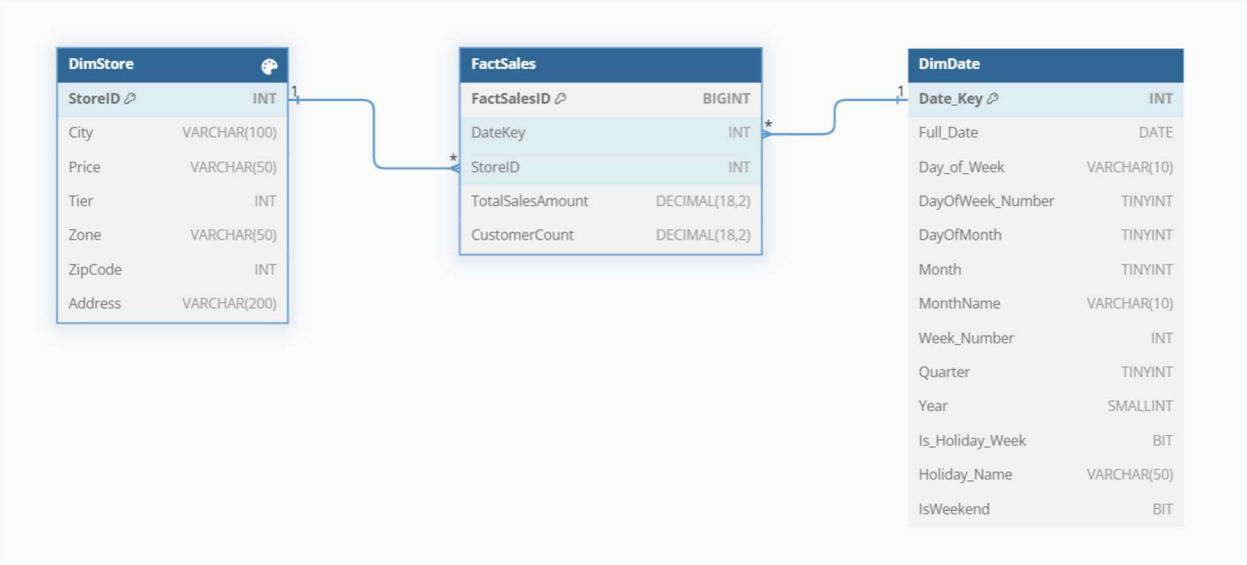
- **DimDate -** This table provides details on calendar-related attributes that allow analysis by various date components. It includes a Date_Key as a surrogate primary key, along with fields like Full_Date, Day_of_Week, Month, Year, and holiday indicators, which help in breaking down sales and customer data based on time dimensions such as holidays, weekends, and specific days of the week.

Column Name	Data Type	Description
Date_Key	INT (PK)	Surrogate key for the date dimension or in YYYYMMDD format
Full_Date	DATE	Full date value
Day_of_Week	VARCHAR(10)	Name of the day (e.g., Monday)
DayOfWeek_Number	TINYINT	Numeric day of the week (1=Sunday, 7=Saturday)
DayOfMonth	TINYINT	Day of the month (1-31)
Month	TINYINT	Month number (1-12)
MonthName	VARCHAR(10)	Name of the month (e.g., January)
Week_Number	INT	Week number of the year (1-53)
Quarter	TINYINT	Quarter of the year (1-4)
Year	SMALLINT	Four-digit year
Is_Holiday_Week	BIT	Indicator if the week is a holiday week (0=False, 1=True)
Holiday_Name	VARCHAR(50)	Name of the holiday, if applicable
IsWeekend	BIT	Indicator if the day is a weekend (0=False, 1=True)

- **FactSales -** This table contains transactional data related to sales at each store, linking back to the DimStore and DimDate dimension tables via StoreID and DateKey, respectively. The FactSalesID serves as the primary key, while the TotalSalesAmount and CustomerCount fields capture the core metrics of sales performance and customer footfall on a given day for each store.

Column Name	Data Type	Description
FactSalesID	BIGINT	Primary Key (surrogate key)
DateKey	INT	Foreign Key to DimDate (DateKey)
StoreID	INT	Foreign Key to DimStore (StoreKey)
TotalSalesAmount	DECIMAL(18,2)	Total sales amount for the date and store
CustomerCount	DECIMAL(18,2)	Number of customers on the specific date

❖ Data Warehouse Schema Diagram



❖ Mapping Table #1 - Source Files to Staging Tables

Source File Name	Source File Attribute	Mapping Function	Staging Table Type	Staging Table Name	Staging Table Attribute
CCOUNTs.csv	STORE	Copy	Relation	Stage_Sales	StoreID
CCOUNTs.csv	DATE	Date Format Transformation	Relation	Stage_Sales	Full_Date
CCOUNTs.csv	GROCERY to SSDELICP	Sum (Total_Sales)	Relation	Stage_Sales	Total_Sales
CCOUNTs.csv	CUSTCOUN	Copy	Relation	Stage_Sales	Customer_Counts
Store.csv	StoreID	Copy	Relation	Stage_Store	Store_ID
Store.csv	City	Copy	Relation	Stage_Store	City
Store.csv	Price	Copy	Relation	Stage_Store	Price_Tier
Store.csv	Zone	Copy	Relation	Stage_Store	Zone
Store.csv	Zip	Copy	Relation	Stage_Store	Zip_Code
Store.csv	Tier	Copy	Relation	Stage_Store	Tier
Store.csv	Address	Copy	Relation	Stage_Store	Address

**Date Transformation: Convert DATE from '900806' to '1990-08-06'.*

❖ Mapping Table #2 - Staging Tables to Presentation Server Tables

Staging Table	Staging Table Attribute	Mapping Function	Data Mart Table Type	Data Mart Table Name	Data Mart Table Attribute
Stage_Sales	Store_ID	Copy	Fact Table	Fact_Sales	StoreID
Stage_Sales	Full_Date	Copy	Fact Table	Fact_Sales	Date_Key
Stage_Sales	Total_Sales	Copy	Fact Table	Fact_Sales	Total_Sales
Stage_Sales	Customer_Counts	Copy	Fact Table	Fact_Sales	Customer_Counts
Stage_Store	Store_ID	Copy	Dimension Table	Dim_Store	Store_Key
Stage_Store	City	Copy	Dimension Table	Dim_Store	City
Stage_Store	Price_Tier	Copy	Dimension Table	Dim_Store	Price_Tier
Stage_Store	Zone	Copy	Dimension Table	Dim_Store	Zone
Stage_Store	Zip_Code	Copy	Dimension Table	Dim_Store	Zip_Code
Stage_Store	Address	Copy	Dimension Table	Dim_Store	Address
Stage_Store	Address	Copy	Dimension Table	Dim_Store	Address
Stage_Sales	Full_Date	Extract Date Components	Dimension Table	Dim_Date	Full_Date, Day_of_Week, Month, etc.

**Extract Date Parts: From the date, DayOfWeek, Month, etc., for DimDate.are extracted*

❖ Schema Justification and Support for Business Questions 3, 5 & 6

The schema supports the business question by enabling comprehensive analysis of sales and customer counts over time and across stores.

- Time Analysis (DimDate):
    - Day of the Week: Attributes like DayOfWeek and DayNameOfWeek allow analysis of sales patterns on different weekdays.
    - Seasonality: Attributes like Month, Quarter, and WeekOfYear enable identification of seasonal trends.
- Store Analysis (DimStore):
    - Location-Based Patterns: Attributes like City, Zone, and ZipCode facilitate geographic analysis.
    - Store Characteristics: Attributes like Price, Tier allow segmentation based on store types.
- Fact Data (FactSales):
    - Sales and Customer Metrics: TotalSalesAmount and CustomerCount provide quantitative measures.
    - Granularity: The grain is daily per store, allowing detailed analysis.

By joining the SalesFact table with the StoreDimension and DateDimension, we can run queries to identify stores with the highest traffic and analyze the correlation between traffic and sales over time for BQ-3. For BQ-5 TotalSalesAmount and CustomerCount can be used to analyze sales and customer traffic within different periods of time.

❖ Physical Design Plan

The data warehouse will be implemented using Microsoft SQL Server 2016 and SQL Server Integration Services (SSIS). This combination provides robust tools for database management and efficient ETL processes, enhancing performance and scalability.

Data Aggregate Plan

- Granularity: Will store data at a daily grain in the FactSales table to enable detailed analysis of sales and customer traffic.
- Aggregations: We will utilize SQL Server's capabilities to create summary tables or indexed views for common aggregations (e.g., weekly, monthly) to improve query performance.

Indexing Plan

- Dimension Tables: Clustered indexes on primary keys (StoreID, DateID) in DimStore and DimDate.
- Fact Table: Non-clustered indexes on foreign keys (StoreID, DateID) in FactSales to enhance join efficiency.

Data Standardization Plan

- Consistent Data Types: INT for identifiers and DECIMAL(18,2) for monetary values to ensure data consistency.
- Naming Conventions: Clear and consistent naming for all database objects to improve maintainability.

Product Sales DataMart

❖ Star Schema Design

➢ **Dimension Tables :-**

- **DimProduct -** Contains product-related information such as UPC, commodity code, item code, product description, size, and the number of items in a case, enabling detailed product-level analysis.

Column Name	Data Type	Description
UPC	BIGINT (PK)	Universal Product Code
CommodityCode	BIGINT	Dominick's Commodity Code
ItemCode	BIGINT	Dominick's Item Code
Description	VARCHAR(50)	Product Name
Size	VARCHAR(10)	Product Size
CasePack	INT	Number of items in a case

- **DimStore -** Stores store-specific details like StoreID, city, price category, tier classification, zone, postal code, and address, providing geographic and store-specific context for sales analysis.

Column Name	Data Type	Description
StoreID	INT (PK)	Unique Store Identifier
City	VARCHAR(50)	City where the store is located
Price	DECIMAL(10,2)	Price category or index
Tier	VARCHAR(20)	Store tier classification

Zone	VARCHAR(20)	Zone designation
ZipCode	VARCHAR(10)	Postal code
Address	VARCHAR(100)	Store address

- **DimDate -** Provides date-related information including TimeKey, exact date, week number, month, quarter, and year, allowing for time-based trend analysis across various temporal dimensions.

Column Name	Data Type	Description
TimeKey	INT (PK)	Surrogate key for time
Date	DATE	Exact date (calculated from week)
WeekNumber	INT	Week number (1-400)
Month	INT	Month (calculated from week)
Quarter	INT	Quarter (calculated from week)
Year	INT	Year (calculated from week)

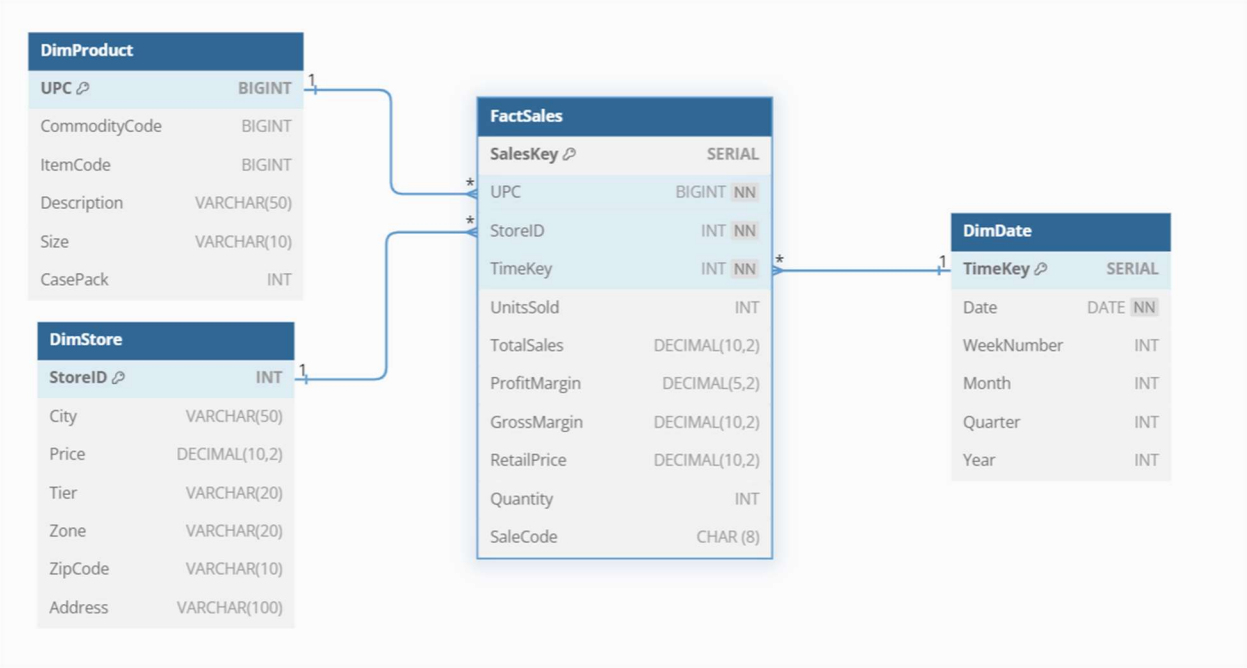
➢ **Fact Tables :-**

- **FactSales -** Holds sales transaction data, including UnitsSold, TotalSales, ProfitMargin, GrossMargin, and RetailPrice, with foreign keys linking to product, store, and time dimensions, enabling comprehensive analysis of sales performance, profitability, and product trends.

Column Name	Data Type	Description
SalesKey	INT (PK)	Surrogate key for sales
UPC	BIGINT (FK)	Foreign key to ProductDimension
StoreID	INT (FK)	Foreign key to StoreDimension
TimeKey	INT (FK)	Foreign key to TimeDimension
UnitsSold	INT	Number of units sold
TotalSales	DECIMAL(10,2)	Total sales value

ProfitMargin	DECIMAL(5,2)	Profit margin percentage
GrossMargin	DECIMAL(10,2)	Gross margin amount
RetailPrice	DECIMAL(10,2)	Retail price of the product
Quantity	INT	Quantity of items bundled together
SaleCode	CHAR(8)	Sale code (B, C, S)

❖ Data Warehouse Schema Diagram



❖ Mapping Table #1: Source Files to Staging Tables

Source File Name	Source File Attribute	Mapping	Staging Table Type	Staging Table Name	Staging Table Attribute
movement*.csv	upc	Copy	Table	stg_movement	upc
movement*.csv	store	Copy	Table	stg_movement	store
movement*.csv	week	Copy	Table	stg_movement	week
movement*.csv	move	Copy	Table	stg_movement	move
movement*.csv	price	Copy	Table	stg_movement	price
movement*.csv	qty	Copy	Table	stg_movement	qty

movement*.csv	profit	Copy	Table	stg_movement	profit
movement*.csv	sale	Copy	Table	stg_movement	sale
movement*.csv	ok	Copy	Table	stg_movement	ok
upc*.csv	upc	Copy	Table	stg_upc	upc
upc*.csv	com_code	Copy	Table	stg_upc	com_code
upc*.csv	nitem	Copy	Table	stg_upc	nitem
upc*.csv	descrip	Copy	Table	stg_upc	descrip
upc*.csv	size	Copy	Table	stg_upc	size
upc*.csv	case	Copy	Table	stg_upc	case
StoreDetails.csv	StoreID	Copy	Table	stg_store	StoreID
StoreDetails.csv	City	Copy	Table	stg_store	City
StoreDetails.csv	Price	Copy	Table	stg_store	Price
StoreDetails.csv	Tier	Copy	Table	stg_store	Tier
StoreDetails.csv	Zone	Copy	Table	stg_store	Zone
StoreDetails.csv	Zip	Copy	Table	stg_store	Zip
StoreDetails.csv	Address	Copy	Table	stg_store	Address

❖ Mapping Table #2: Staging Tables to Presentation Server Tables

Staging Table	Staging Table Attribute	Mapping	Data Mart Table Type	Data Mart Table Name	Data Mart Table Attribute
stg_movement	upc	Copy	Fact Table	SalesFact	UPC
stg_movement	store	Copy	Fact Table	SalesFact	StoreID
stg_movement	week	Copy	Fact Table	SalesFact	TimeKey

stg_movement	move	Copy	Fact Table	SalesFact	UnitsSold
stg_movement	price	Copy	Fact Table	SalesFact	RetailPrice
stg_movement	qty	Copy	Fact Table	SalesFact	Quantity
stg_movement	profit	Calculate ProfitMargin = profit * 100	Fact Table	SalesFact	ProfitMargin
stg_movement	profit	Copy	Fact Table	SalesFact	GrossMargin
stg_movement	sale	Copy	Fact Table	SalesFact	SaleCode
stg_upc	upc	Copy	Dimension Table	ProductDimension	UPC
stg_upc	com_code	Copy	Dimension Table	ProductDimension	CommodityCode
stg_upc	nitem	Copy	Dimension Table	ProductDimension	ItemCode
stg_upc	descrip	Copy	Dimension Table	ProductDimension	Description
stg_upc	size	Copy	Dimension Table	ProductDimension	Size
stg_upc	case	Copy	Dimension Table	ProductDimension	CasePack
stg_store	StoreID	Copy	Dimension Table	StoreDimension	StoreID
stg_store	City	Copy	Dimension Table	StoreDimension	City
stg_store	Price	Copy	Dimension Table	StoreDimension	Price
stg_store	Tier	Copy	Dimension Table	StoreDimension	Tier
stg_store	Zone	Copy	Dimension Table	StoreDimension	Zone
stg_store	Zip	Copy	Dimension Table	StoreDimension	ZipCode
stg_store	Address	Copy	Dimension Table	StoreDimension	Address

stg_movement	week	Calculate Date, Month, Quarter, Year from WeekNumber	Dimension Table	TimeDimension	Time Attributes

❖ Schema Justification and Support for Business Questions 4 & 10

The Star schema effectively supports the business question by using natural keys (UPC and StoreID) as the primary keys in the product and store dimensions, respectively.

- **Product Dimension**:
  - Using **UPC** as the primary key ensures that each product is uniquely identified by its universally recognized code, facilitating easy integration and lookup.
- **Store Dimension**:
  - Using **StoreID** as the primary key aligns with the data provided, ensuring a unique identifier for each store across all CSV files.
- **Time Dimension**:
  - A surrogate key **TimeKey** is used for the time dimension, as is standard practice due to the complex nature of time attributes.
- **Sales Fact Table**:
  - Contains foreign keys to the dimensions using **UPC**, **StoreID**, and **TimeKey**.
  - Measures such as **UnitsSold**, **TotalSales**, **ProfitMargin** etc., are included to enable analysis of product profit margins across stores and time periods.
  - **SaleCode** captures the type of sale which can be used identify the effect of coupon on sales across different products, time ranges and stores.

By integrating these dimensions with the fact table using the natural keys, we can generate comprehensive reports and insights, such as:

- Identifying products with the highest profit margins.
- Analyzing coupon sales patterns
- Assessing the consistency of profit margins across different stores identified by StoreID.
- Analyzing trends over time using the Time Dimension.

❖ Physical Design Plan

The data warehouse will be implemented using Microsoft SQL Server 2016 and SQL Server Integration Services (SSIS). This combination provides robust tools for database management and efficient ETL processes, enhancing performance and scalability.

Data Aggregate Plan

- Granularity: Will store data at a daily grain in the FactSales table to enable detailed analysis of sales and customer traffic.

- Aggregations: Will utilize SQL Server's capabilities to create summary tables or indexed views for common aggregations (e.g., weekly, monthly) to improve query performance.

Indexing Plan

- **Fact Table Indexes**:
  - Create indexes on foreign key columns (**UPC**, **StoreID**, **TimeKey**) to optimize joins.
- **Dimension Table Indexes**:
  - Since the primary keys are natural keys (UPC and StoreID), these are inherently indexed.
  - Additional indexes can be created on attributes frequently used in queries, such as **CommodityCode** in the ProductDimension or **City** in the StoreDimension.

Data Standardization Plan

- Consistent Data Types: Will use INT for identifiers and DECIMAL(18,2) for monetary values to ensure data consistency.
- Naming Conventions: Will apply clear and consistent naming for all database objects to improve maintainability.

ETL Pipeline Development and Implementation Report for Category Sales DataMart

- ## Target Data Needed in the Data Warehouse :-

  The data warehouse will consist of the following target tables:
  - **Dimension Tables**:
    - **DimStore**: Contains store-related information.
      - Attributes: StoreID, City, Price_Tier, Zone, ZipCode, Address
    - **DimDate**: Contains date-related information.
      - Attributes: Date_Key, Full_Date, Day_of_Week, DayOfWeek_Number, DayOfMonth, Month, MonthName, Week_Number, Quarter, Year, Is_Holiday_Week, Holiday_Name, IsWeekend
  - **Fact Table**:
    - **FactSales**: Contains transactional sales data.
      - Attributes: FactSalesID, DateKey, StoreID, TotalSalesAmount, CustomerCount

- ## Data Sources (Available from DFF) :-

  The following data sources are available from the Data Feed Files (DFF):

  - **CCOUNTs.csv**: Contains daily sales and customer counts per store.
    - Attributes: STORE, DATE, GROCERY, SSDELICP, CUSTCOUN
  - **Store.csv**: Contains master data for each store.
    - Attributes: StoreID, City, Price, Zone, Zip, Tier, Address

- ## Data Mappings :-

  Mapping from Source Files to Staging Tables :-

  - **Store.csv to Stage_Store**

Source File	Source Attribute	Transformation	Staging Attribute
Store.csv	StoreID	Copy	StoreID
Store.csv	City	Copy	City
Store.csv	Price	Copy	Price_Tier
Store.csv	Zone	Copy	Zone
Store.csv	Zip	Copy	Zip_Code
Store.csv	Address	Copy	Address

  - **Stage_Date to DimDate**

Source File	Staging Table Attribute	Transformation	Dimension Table	Dim Table Attribute
Stage_Date	DateKey	Copy	DimDate	DateKey
Stage_Date	Full_Date	Copy	DimDate	Full_Date
Stage_Date	Day_Of_Week	Copy	DimDate	Day_Of_Week
Stage_Date	DayOfWeek_Number	Copy	DimDate	DayOfWeek_Number
Stage_Date	DayOfMonth	Copy	DimDate	DayOfMonth
Stage_Date	Month	Copy	DimDate	Month
Stage_Date	MonthName	Copy	DimDate	MonthName
Stage_Date	Week_Number	Copy	DimDate	Week_Number
Stage_Date	Quarter	Copy	DimDate	Quarter
Stage_Date	Year	Copy	DimDate	Year
Stage_Date	Holiday_Name	Copy	DimDate	Holiday_Name
Stage_Date	IsWeekend	Copy	DimDate	IsWeekend
Stage_Date	Holiday_Week	Copy	DimDate	Holiday_Week

  - **Ccount.csv to Stage_Sales_v1**

Source File	Source Attribute	Transformation	Staging Attribute
Ccount.csv	STORE	Copy	STORE
Ccount.csv	DATE	Copy	DATE

Ccount.csv	GROCERY	Copy	GROCERY
Ccount.csv	DAIRY	Copy	DAIRY
Ccount.csv	FROZEN	Copy	FROZEN
Ccount.csv	BOTTLE	Copy	BOTTLE
Ccount.csv	MVPCLUB	Copy	MVPCLUB
Ccount.csv	GROCCOUP	Copy	GROCCOUP
Ccount.csv	MEAT	Copy	MEAT
Ccount.csv	MEATFROZ	Copy	MEATFROZ
Ccount.csv	MEATCOUP	Copy	MEATCOUP
Ccount.csv	FISH	Copy	FISH
Ccount.csv	FISHCOUP	Copy	FISHCOUP
Ccount.csv	PROMO	Copy	PROMO
Ccount.csv	PROMCOUP	Copy	PROMCOUP
Ccount.csv	PRODUCE	Copy	PRODUCE
Ccount.csv	BULK	Copy	BULK
Ccount.csv	SALADBAR	Copy	SALADBAR
Ccount.csv	PRODCOUP	Copy	PRODCOUP
Ccount.csv	BULKCOUP	Copy	BULKCOUP
Ccount.csv	SALCOUP	Copy	SALCOUP
Ccount.csv	FLORAL	Copy	FLORAL
Ccount.csv	FLORCOUP	Copy	FLORCOUP
Ccount.csv	DELI	Copy	DELI
Ccount.csv	DELISELF	Copy	DELISELF
Ccount.csv	DELIEXPR	Copy	DELIEXPR
Ccount.csv	CONVFOOD	Copy	CONVFOOD
Ccount.csv	CHEESE	Copy	CHEESE
Ccount.csv	DELICOUP	Copy	DELICOUP
Ccount.csv	BAKERY	Copy	BAKERY
Ccount.csv	PHARMACY	Copy	PHARMACY
Ccount.csv	PHARCOUP	Copy	PHARCOUP
Ccount.csv	GM	Copy	GM
Ccount.csv	JEWELRY	Copy	JEWELRY
Ccount.csv	COSMETIC	Copy	COSMETIC
Ccount.csv	HABA	Copy	HABA
Ccount.csv	GMCOUP	Copy	GMCOUP
Ccount.csv	CAMERA	Copy	CAMERA
Ccount.csv	PHOTOFIN	Copy	PHOTOFIN
Ccount.csv	VIDEO	Copy	VIDEO
Ccount.csv	VIDEOREN	Copy	VIDEOREN
Ccount.csv	VIDCOUP	Copy	VIDCOUP
Ccount.csv	BEER	Copy	BEER
Ccount.csv	WINE	Copy	WINE
Ccount.csv	SPIRITS	Copy	SPIRITS

Ccount.csv	MISCSCP	Copy	MISCSCP
Ccount.csv	MANCOUP	Copy	MANCOUP
Ccount.csv	CUSTCOUN	Copy	CUSTCOUN
Ccount.csv	FTGCHIN	Copy	FTGCHIN
Ccount.csv	FTGCCOUP	Copy	FTGCCOUP
Ccount.csv	FTGITAL	Copy	FTGITAL
Ccount.csv	FTGICOUP	Copy	FTGICOUP
Ccount.csv	DAIRCOUP	Copy	DAIRCOUP
Ccount.csv	FROZCOUP	Copy	FROZCOUP
Ccount.csv	HABACOUP	Copy	HABACOUP
Ccount.csv	PHOTCOUP	Copy	PHOTCOUP
Ccount.csv	COSMCOUP	Copy	COSMCOUP
Ccount.csv	SSDELICP	Copy	SSDELICP
Ccount.csv	BAKCOUP	Copy	BAKCOUP
Ccount.csv	LIQCOUP	Copy	LIQCOUP
Ccount.csv	WEEK	Copy	WEEK

- **Stage_Sales_v1 to Stage_Sales_v2**

Source File	Source Attribute	Transformation	Staging Attribute
Stage_Sales_v1	STORE	Copy	STORE
Stage_Sales_v1	DATE	Copy	DATE
Stage_Sales_v1	GROCERY	Transform from String to Decimal	GROCERY
Stage_Sales_v1	DAIRY	Transform from String to Decimal	DAIRY
Stage_Sales_v1	FROZEN	Transform from String to Decimal	FROZEN
Stage_Sales_v1	BOTTLE	Transform from String to Decimal	BOTTLE
Stage_Sales_v1	MVPCLUB	Transform from String to Decimal	MVPCLUB
Stage_Sales_v1	GROCCOUP	Transform from String to Decimal	GROCCOUP
Stage_Sales_v1	MEAT	Transform from String to Decimal	MEAT
Stage_Sales_v1	MEATFROZ	Transform from String to Decimal	MEATFROZ
Stage_Sales_v1	MEATCOUP	Transform from String to Decimal	MEATCOUP
Stage_Sales_v1	FISH	Transform from String to Decimal	FISH
Stage_Sales_v1	FISHCOUP	Transform from String to Decimal	FISHCOUP
Stage_Sales_v1	PROMO	Transform from	PROMO

		String to Decimal	
Stage_Sales_v1	PROMCOUP	Transform from String to Decimal	PROMCOUP
Stage_Sales_v1	PRODUCE	Transform from String to Decimal	PRODUCE
Stage_Sales_v1	BULK	Transform from String to Decimal	BULK
Stage_Sales_v1	SALADBAR	Transform from String to Decimal	SALADBAR
Stage_Sales_v1	PRODCOUP	Transform from String to Decimal	PRODCOUP
Stage_Sales_v1	BULKCOUP	Transform from String to Decimal	BULKCOUP
Stage_Sales_v1	SALCOUP	Transform from String to Decimal	SALCOUP
Stage_Sales_v1	FLORAL	Transform from String to Decimal	FLORAL
Stage_Sales_v1	FLORCOUP	Transform from String to Decimal	FLORCOUP
Stage_Sales_v1	DELI	Transform from String to Decimal	DELI
Stage_Sales_v1	DELISELF	Transform from String to Decimal	DELISELF
Stage_Sales_v1	DELIEXPR	Transform from String to Decimal	DELIEXPR
Stage_Sales_v1	CONVFOOD	Transform from String to Decimal	CONVFOOD
Stage_Sales_v1	CHEESE	Transform from String to Decimal	CHEESE
Stage_Sales_v1	DELICOUP	Transform from String to Decimal	DELICOUP
Stage_Sales_v1	BAKERY	Transform from String to Decimal	BAKERY
Stage_Sales_v1	PHARMACY	Transform from String to Decimal	PHARMACY
Stage_Sales_v1	PHARCOUP	Transform from String to Decimal	PHARCOUP
Stage_Sales_v1	GM	Transform from String to Decimal	GM
Stage_Sales_v1	JEWELRY	Transform from String to Decimal	JEWELRY
Stage_Sales_v1	COSMETIC	Transform from String to Decimal	COSMETIC
Stage_Sales_v1	HABA	Transform from String to Decimal	HABA
Stage_Sales_v1	GMCOUP	Transform from String to Decimal	GMCOUP

Stage_Sales_v1	CAMERA	Transform from String to Decimal	CAMERA
Stage_Sales_v1	PHOTOFIN	Transform from String to Decimal	PHOTOFIN
Stage_Sales_v1	VIDEO	Transform from String to Decimal	VIDEO
Stage_Sales_v1	VIDEOREN	Transform from String to Decimal	VIDEOREN
Stage_Sales_v1	VIDCOUP	Transform from String to Decimal	VIDCOUP
Stage_Sales_v1	BEER	Transform from String to Decimal	BEER
Stage_Sales_v1	WINE	Transform from String to Decimal	WINE
Stage_Sales_v1	SPIRITS	Transform from String to Decimal	SPIRITS
Stage_Sales_v1	MISCSCP	Transform from String to Decimal	MISCSCP
Stage_Sales_v1	MANCOUP	Transform from String to Decimal	MANCOUP
Stage_Sales_v1	CUSTCOUN	Transform from String to Decimal	CUSTCOUN
Stage_Sales_v1	FTGCHIN	Transform from String to Decimal	FTGCHIN
Stage_Sales_v1	FTGCCOUP	Transform from String to Decimal	FTGCCOUP
Stage_Sales_v1	FTGITAL	Transform from String to Decimal	FTGITAL
Stage_Sales_v1	FTGICOUP	Transform from String to Decimal	FTGICOUP
Stage_Sales_v1	DAIRCOUP	Transform from String to Decimal	DAIRCOUP
Stage_Sales_v1	FROZCOUP	Transform from String to Decimal	FROZCOUP
Stage_Sales_v1	HABACOUP	Transform from String to Decimal	HABACOUP
Stage_Sales_v1	PHOTCOUP	Transform from String to Decimal	PHOTCOUP
Stage_Sales_v1	COSMCOUP	Transform from String to Decimal	COSMCOUP
Stage_Sales_v1	SSDELICP	Transform from String to Decimal	SSDELICP
Stage_Sales_v1	BAKCOUP	Transform from String to Decimal	BAKCOUP
Stage_Sales_v1	LIQCOUP	Transform from String to Decimal	LIQCOUP
Stage_Sales_v1	WEEK	Copy	WEEK

- **Stage_Sales_v2 to Stage_Sales_v3**

Source File	Source Attribute	Transformation	Staging Attribute
Stage_Sales_v3	STORE	Copy	STORE
Stage_Sales_v3	DATE	Date format conversion	DATE
Stage_Sales_v3	GROCERY, SSDELICP	Sum (GROCERY + ….. + SSDELICP)	TOTAL_SALES
Stage_Sales_v3	CUSTCOUN	Copy	CUST_COUNT

Mapping from Staging to Data Marts:

- **Stage_Store to DimStore**

Source File	Staging Table Attribute	Transformation	Dimension Table	Dim Table Attribute
Stage_Store	StoreID	Copy	DimStore	StoreID
Stage_Store	City	Copy	DimStore	City
Stage_Store	Price	Copy	DimStore	Price_Tier
Stage_Store	Zone	Copy	DimStore	Zone
Stage_Store	Zip	Copy	DimStore	Zip_Code
Stage_Store	Address	Copy	DimStore	Address

- **Stage_Sales_v3 to FactSales**

Source File	Staging Table Attribute	Transformation	Fact Table	Fact Table Attribute
Stage_Date	STORE	Copy	FactSales	StoreID
Stage_Date	DATE	Copy	FactSales	DateKey
Stage_Date	TOTAL_SALES	Copy	FactSales	Total_Sales
Stage_Date	CUST_COUNT	Copy	FactSales	Cust_Count

- ## Data Extraction Rules :-
  - **For DimStore :-**
    1. Store Details City, Price_Tier, Zone, Zip, Address are present in the DFF Data Manual, it is copied to csv Store.csv
    2. Store.csv data is stored in staging table Stage_Store in [601_grp4_CategorySalesStaging] DB by copying all the columns from csv
  - **For FactSales :-**

1. Ccount.csv has all the data about sales and customer count metrics for each day for all the stores is downloaded from the Zip provided.
2. Data from the csv is copied to table Stage_Sales_v1 in [601_grp4_CategorySalesStaging] DB by copying all the columns.

- ## Data Transformation and Cleansing Rules :-
  - **For DimStore :-**
    1. Remove " character from Store ID and Address columns using derived Column in Data Flow

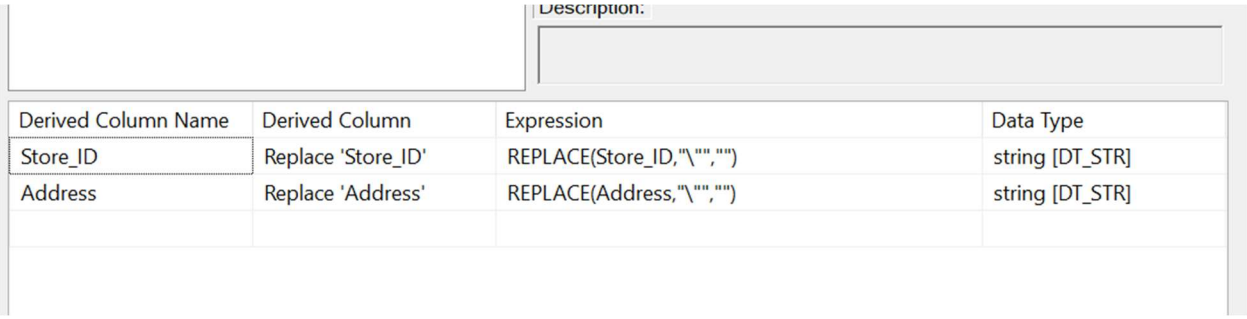


Fig1. Derived Column Block to remove "

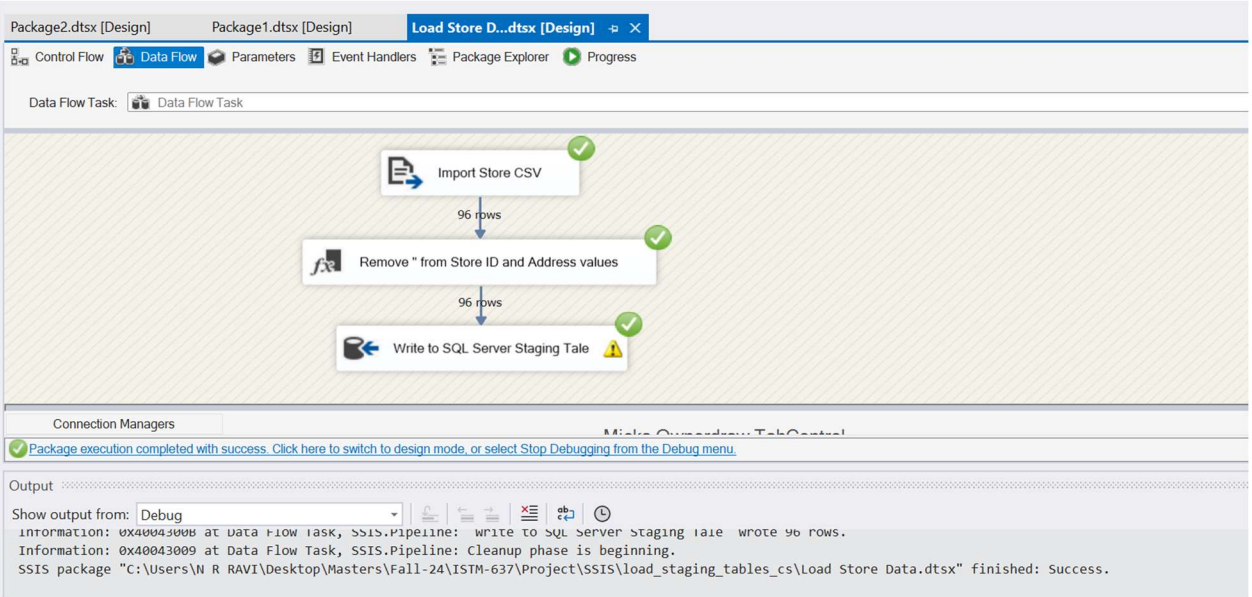
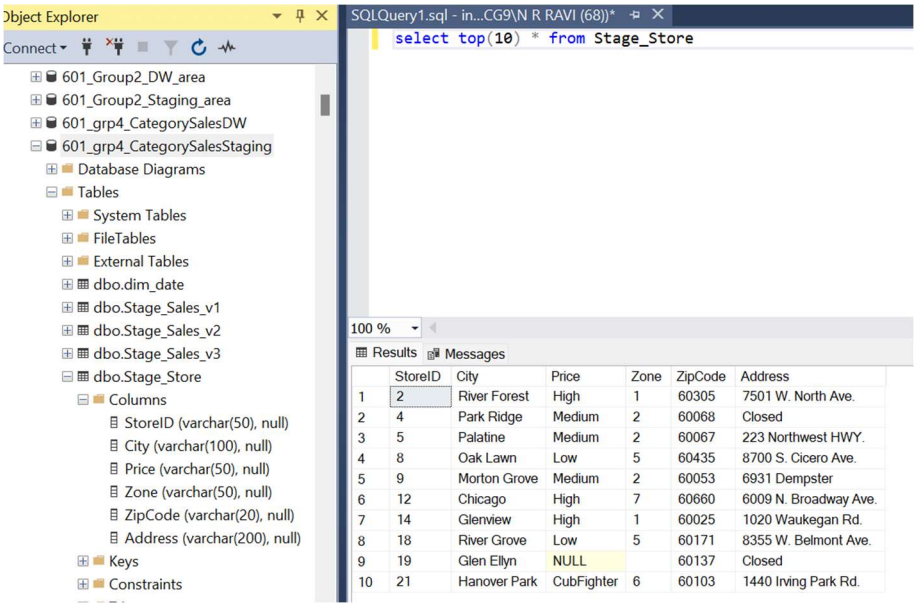


Fig 2. Data copied from Store.csv to Stage_Store



	StoreID	City	Price	Zone	ZipCode	Address
1	2	River Forest	High	1	60305	7501 W. North Ave.
2	4	Park Ridge	Medium	2	60068	Closed
3	5	Palatine	Medium	2	60067	223 Northwest HWY.
4	8	Oak Lawn	Low	5	60435	8700 S. Cicero Ave.
5	9	Morton Grove	Medium	2	60053	6931 Dempster
6	12	Chicago	High	7	60660	6009 N. Broadway Ave.
7	14	Glenview	High	1	60025	1020 Waukegan Rd.
8	18	River Grove	Low	5	60171	8355 W. Belmont Ave.
9	19	Glen Ellyn	NULL		60137	Closed
10	21	Hanover Park	CubFighter	6	60103	1440 Irving Park Rd.

Fig 3. Stage_Store Populated

- **For DimDate :-**
  1. Customer count has data from all the dates from January 1st 1988 to December 31st 1996 with a granularity of 1 day.
  2. SQL query is written to populate this table.

```
- - CTE to select the dat2 range from 1st Jan 1988 to 31st dec 1966
WITH DateRange AS (
   SELECT CAST('1988-01-01' AS DATE) AS Full_Date
   UNION ALL
   SELECT DATEADD(DAY, 1, Full_Date)
   FROM DateRange
   WHERE Full_Date < '1996-12-31'
)
-- Insert generated dates into dim_date with calculated fields
INSERT INTO dim_date (
   Date_Key,
   Full_Date,
   Day_of_Week,
   DayOfWeek_Number,
   DayOfMonth,
   Month,
   MonthName,
   Week_Number,
   Quarter,
   Year,
   Holiday_Week,
   Holiday_Name,
```

```
    IsWeekend
)
SELECT
    CONVERT(INT, FORMAT(Full_Date, 'yyyyMMdd')) AS Date_Key,
    Full_Date,
    DATENAME(WEEKDAY, Full_Date) AS Day_of_Week,
    DATEPART(WEEKDAY, Full_Date) AS DayOfWeek_Number,
    DAY(Full_Date) AS DayOfMonth,
    MONTH(Full_Date) AS Month,
    DATENAME(MONTH, Full_Date) AS MonthName,
    DATEPART(WEEK, Full_Date) AS Week_Number,
    DATEPART(QUARTER, Full_Date) AS Quarter,
    YEAR(Full_Date) AS Year,
    NULL AS Holiday_Week,
    NULL AS Holiday_Name,
    CASE WHEN DATEPART(WEEKDAY, Full_Date) IN (1, 7) THEN 1 ELSE 0 END AS
IsWeekend
FROM DateRange
OPTION (MAXRECURSION 0);

-- CTE for years
WITH Years AS (
    SELECT 1988 AS Year
    UNION ALL
    SELECT Year + 1 FROM Years WHERE Year + 1 <= 1996
),
-- CTE for Holidays
Holidays AS (
    -- New Year's Day
    SELECT Year, 'New Year''s Day' AS Holiday_Name, DATEFROMPARTS(Year, 1, 1) AS
Holiday_Date FROM Years
    UNION ALL
    -- Martin Luther King Jr. Day (Third Monday in January)
    SELECT Year, 'Martin Luther King Jr. Day',
        DATEADD(WEEK, 2, DATEADD(DAY, (1 - DATEPART(WEEKDAY, DATEFROMPARTS(Year,
1, 1)) + 7) % 7, DATEFROMPARTS(Year, 1, 1)))
    FROM Years
    UNION ALL
    -- Presidents' Day (Third Monday in February)
    SELECT Year, 'Presidents'' Day',
        DATEADD(WEEK, 2, DATEADD(DAY, (1 - DATEPART(WEEKDAY, DATEFROMPARTS(Year,
2, 1)) + 7) % 7, DATEFROMPARTS(Year, 2, 1)))
    FROM Years
    UNION ALL
```

```
    -- Memorial Day (Last Monday in May)
    SELECT Year, 'Memorial Day',
        DATEADD(DAY, -((DATEPART(WEEKDAY, EOMONTH(DATEFROMPARTS(Year, 5, 1))) + 6)
% 7), EOMONTH(DATEFROMPARTS(Year, 5, 1)))
    FROM Years
    UNION ALL
    -- Juneteenth National Independence Day
    SELECT Year, 'Juneteenth National Independence Day', DATEFROMPARTS(Year, 6, 19)
FROM Years
    UNION ALL
    -- Independence Day
    SELECT Year, 'Independence Day', DATEFROMPARTS(Year, 7, 4) FROM Years
    UNION ALL
    -- Labor Day (First Monday in September)
    SELECT Year, 'Labor Day',
        DATEADD(DAY, (1 - DATEPART(WEEKDAY, DATEFROMPARTS(Year, 9, 1)) + 7) % 7,
DATEFROMPARTS(Year, 9, 1))
    FROM Years
    UNION ALL
    -- Columbus Day (Second Monday in October)
    SELECT Year, 'Columbus Day',
        DATEADD(WEEK, 1, DATEADD(DAY, (1 - DATEPART(WEEKDAY, DATEFROMPARTS(Year,
10, 1)) + 7) % 7, DATEFROMPARTS(Year, 10, 1)))
    FROM Years
    UNION ALL
    -- Veterans Day
    SELECT Year, 'Veterans Day', DATEFROMPARTS(Year, 11, 11) FROM Years
    UNION ALL
    -- Thanksgiving Day (Fourth Thursday in November)
    SELECT Year, 'Thanksgiving Day',
        DATEADD(WEEK, 3, DATEADD(DAY, (5 - DATEPART(WEEKDAY, DATEFROMPARTS(Year,
11, 1)) + 7) % 7, DATEFROMPARTS(Year, 11, 1)))
    FROM Years
    UNION ALL
    -- Christmas Day
    SELECT Year, 'Christmas Day', DATEFROMPARTS(Year, 12, 25) FROM Years
)
-- Update Holiday_Name in dim_date
UPDATE d
SET d.Holiday_Name = h.Holiday_Name
FROM dim_date d
JOIN Holidays h ON d.Full_Date = h.Holiday_Date;

-- To retrieve all holidays with their week numbers
```

```
WITH HolidayWeeks AS (
   SELECT
      d.Year,
      d.Week_Number,
      d.Holiday_Name
   FROM dim_date d
   WHERE d.Holiday_Name IS NOT NULL
)
-- To Update the Holiday_Week column for all days in the same week as any holiday
UPDATE d
SET d.Holiday_Week = hw.Holiday_Name
FROM dim_date d
INNER JOIN HolidayWeeks hw
   ON d.Year = hw.Year
   AND d.Week_Number = hw.Week_Number;
```

3. All the dates in the range is generated and SQL method DATEPART and DATENAME is use to extract the values for Date_Key, Full_Date, Day_of_Week, DayOfWeek_Number, DayOfMonth, Month, Month number, MonthName, Week_Number , Quarter, Year, IsWeekend
4. For getting the holiday CTEs are written to identify one of the following holidays and column Holiday_Name column is populated
    - **New Year's Day** - January 1
    - **Martin Luther King Jr. Day** - Third Monday in January
    - **Presidents' Day** - Third Monday in February
    - **Memorial Day** - Last Monday in May
    - **Juneteenth National Independence Day** - June 19
    - **Independence Day** - July 4
    - **Labor Day** - First Monday in September
    - **Columbus Day** (or Indigenous Peoples' Day in some states) - Second Monday in October
    - **Veterans Day** - November 11
    - **Thanksgiving Day** - Fourth Thursday in November
    - **Christmas Day** - December 25
5. After the holiday dates are identified, the week to which the holiday belongs is identified and the column Holiday_Week is filled with holiday name for the whole week

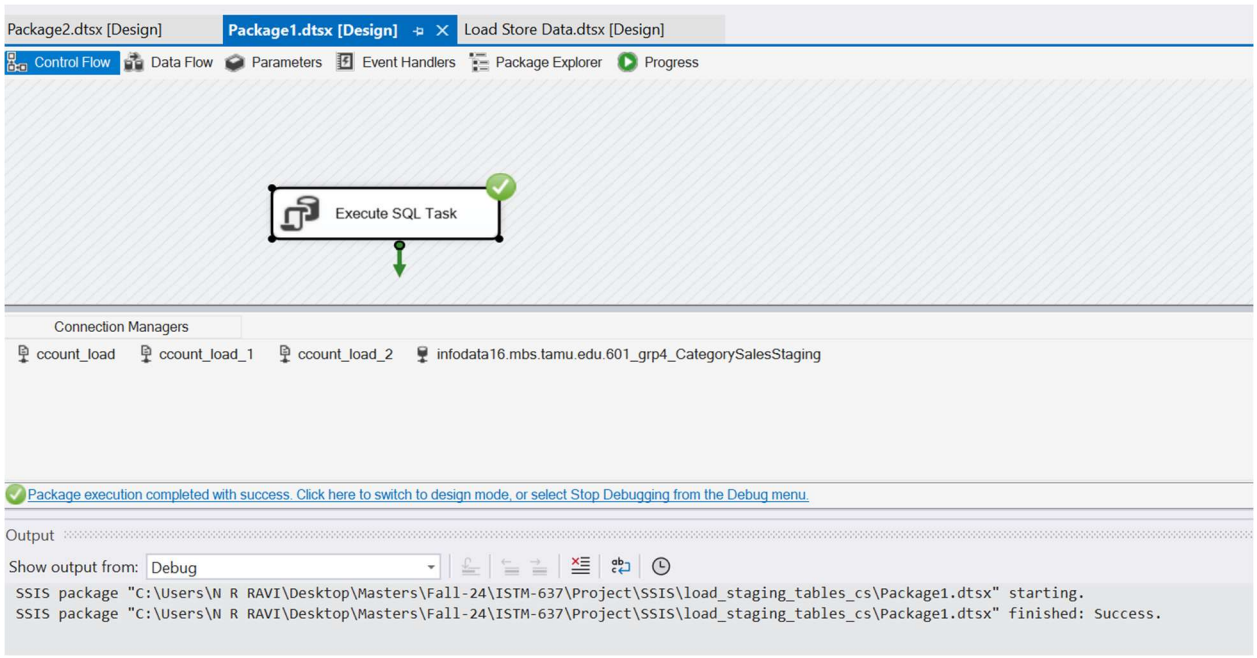
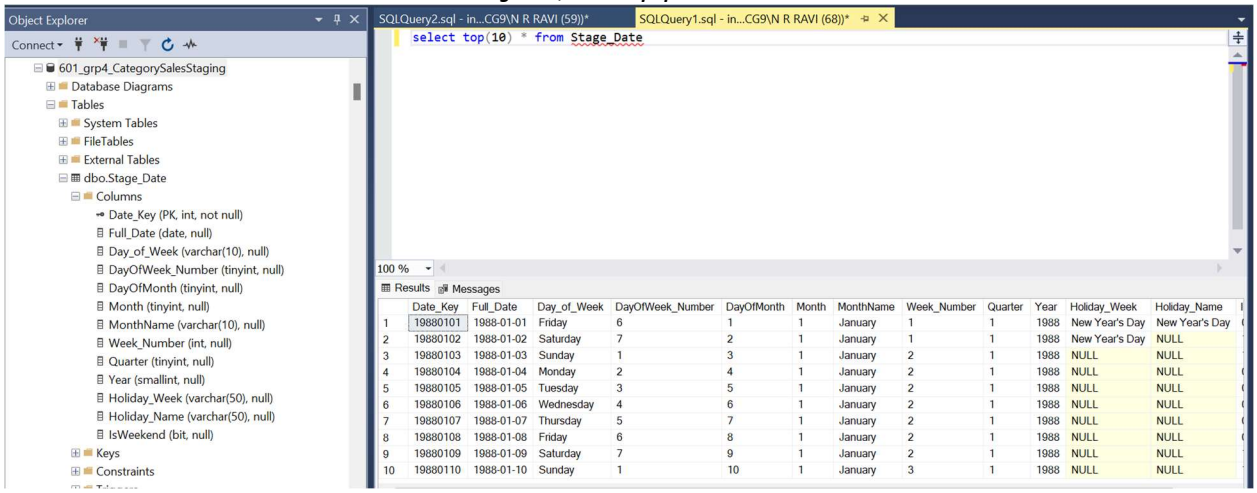


Fig 4. SQL Task to populate the dates



	Date_Key	Full_Date	Day_of_Week	DayOfWeek_Number	DayOfMonth	Month	MonthName	Week_Number	Quarter	Year	Holiday_Week	Holiday_Name
1	19880101	1988-01-01	Friday	6	1	1	January	1	1	1988	New Year's Day	New Year's Day
2	19880102	1988-01-02	Saturday	7	2	1	January	1	1	1988	New Year's Day	NULL
3	19880103	1988-01-03	Sunday	1	3	1	January	2	1	1988	NULL	NULL
4	19880104	1988-01-04	Monday	2	4	1	January	2	1	1988	NULL	NULL
5	19880105	1988-01-05	Tuesday	3	5	1	January	2	1	1988	NULL	NULL
6	19880106	1988-01-06	Wednesday	4	6	1	January	2	1	1988	NULL	NULL
7	19880107	1988-01-07	Thursday	5	7	1	January	2	1	1988	NULL	NULL
8	19880108	1988-01-08	Friday	6	8	1	January	2	1	1988	NULL	NULL
9	19880109	1988-01-09	Saturday	7	9	1	January	2	1	1988	NULL	NULL
10	19880110	1988-01-10	Sunday	1	10	1	January	3	1	1988	NULL	NULL

Fig 5. Stage_Date Table Populated

- **For FactSales :-**
  1. SQL query is used to transform all the sales data per category from STR in Stage_Sales_v1 to DECIMAL data type. If the column has non numerical data, it is considered as 0 else it is converted and stored in Stage_Sales_v2

```
INSERT INTO Stage_Sales_v2 (
   STORE, DATE, "WEEK", GROCERY, DAIRY, FROZEN, BOTTLE, MVPCLUB, GROCCOUP,
MEAT, MEATFROZ, MEATCOUP,
   FISH, FISHCOUP, PROMO, PROMCOUP, PRODUCE, "BULK", SALADBAR, PRODCOUP,
BULKCOUP,
```

```
   SALCOUP, FLORAL, FLORCOUP, DELI, DELISELF, DELIEXPR, CONVFOOD, CHEESE,
DELICOUP,
   BAKERY, PHARMACY, PHARCOUP, GM, JEWELRY, COSMETIC, HABA, GMCOUP,
CAMERA, PHOTOFIN,
   VIDEO, VIDEOREN, VIDCOUP, BEER, WINE, SPIRITS, MISCSCP, MANCOUP,
CUSTCOUN, FTGCHIN,
   FTGCCOUP, FTGITAL, FTGICOUP, DAIRCOUP, FROZCOUP, HABACOUP, PHOTCOUP,
COSMCOUP,
   SSDELICP, BAKCOUP, LIQCOUP
)
SELECT
   STORE,
   DATE,
   "WEEK",
   COALESCE(TRY_CAST(GROCERY AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(DAIRY AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FROZEN AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(BOTTLE AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(MVPCLUB AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(GROCCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(MEAT AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(MEATFROZ AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(MEATCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FISH AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FISHCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PROMO AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PROMCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PRODUCE AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST("BULK" AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(SALADBAR AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PRODCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(BULKCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(SALCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FLORAL AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FLORCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(DELI AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(DELISELF AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(DELIEXPR AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(CONVFOOD AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(CHEESE AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(DELICOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(BAKERY AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PHARMACY AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PHARCOUP AS NUMERIC(18,2)), 0),
```

```
   COALESCE(TRY_CAST(GM AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(JEWELRY AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(COSMETIC AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(HABA AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(GMCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(CAMERA AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PHOTOFIN AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(VIDEO AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(VIDEOREN AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(VIDCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(BEER AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(WINE AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(SPIRITS AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(MISCSCP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(MANCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(CUSTCOUN AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FTGCHIN AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FTGCCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FTGITAL AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FTGICOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(DAIRCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(FROZCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(HABACOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(PHOTCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(COSMCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(SSDELICP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(BAKCOUP AS NUMERIC(18,2)), 0),
   COALESCE(TRY_CAST(LIQCOUP AS NUMERIC(18,2)), 0)
FROM Stage_Sales_v1;
```

2. All the columns from Grocery to LIQCOUP are added from Stage_Sales_v2 to generated the total sales for each row in table Stage_Sales_v3. Date column is copied with character " which is remove and dates which do not fall in the range and the rows which have garbage date values is discarded.

```
INSERT INTO Stage_Sales_v3 (STORE, TOTAL_SALES, CUST_COUNT, DATE)

SELECT

   -- Store copied as is

   STORE AS STORE,

   -- Sum of all specified columns

   (GROCERY + DAIRY + FROZEN + BOTTLE + MVPCLUB + GROCCOUP + MEAT +
MEATFROZ + MEATCOUP +
```

```
   FISH + FISHCOUP + PROMO + PROMCOUP + PRODUCE + "BULK" + SALADBAR +
PRODCOUP + BULKCOUP +

   SALCOUP + FLORAL + FLORCOUP + DELI + DELISELF + DELIEXPR + CONVFOOD +
CHEESE + DELICOUP +

   BAKERY + PHARMACY + PHARCOUP + GM + JEWELRY + COSMETIC + HABA +
GMCOUP + CAMERA + PHOTOFIN +

   VIDEO + VIDEOREN + VIDCOUP + BEER + WINE + SPIRITS + MISCSCP + MANCOUP +
FTGCHIN + FTGCCOUP +

   FTGITAL + FTGICOUP + DAIRCOUP + FROZCOUP + HABACOUP + PHOTCOUP +
COSMCOUP + SSDELICP +

   BAKCOUP + LIQCOUP) AS TOTAL_SALES,


   -- Copying CUSTCOUN as is

   CUSTCOUN AS CUST_COUNT,

-- Check if date is of size 6 and fall within range, if yes add it as an INT column

   CASE

      WHEN TRY_CAST(CAST((TRY_CAST(SUBSTRING(REPLACE(DATE, '"', ''), 1, 2) AS
INT) + 1900) AS CHAR(4))

         + SUBSTRING(REPLACE(DATE, '"', ''), 3, 2) + SUBSTRING(REPLACE(DATE, '"', ''),
5, 2) AS DATE)

         BETWEEN '1988-01-01' AND '1996-12-31'

      THEN TRY_CAST(CAST((TRY_CAST(SUBSTRING(REPLACE(DATE, '"', ''), 1, 2) AS
INT) + 1900) AS CHAR(4))

         + SUBSTRING(REPLACE(DATE, '"', ''), 3, 2) + SUBSTRING(REPLACE(DATE, '"', ''),
5, 2) AS DATE)

   END AS DATE

FROM

   Stage_Store_v3

WHERE

   LEN(REPLACE(DATE, '"', '')) = 6

   AND TRY_CAST(SUBSTRING(REPLACE(DATE, '"', ''), 1, 2) AS INT) IS NOT NULL

   AND TRY_CAST(SUBSTRING(REPLACE(DATE, '"', ''), 3, 2) AS INT) IS NOT NULL
```

```
AND TRY_CAST(SUBSTRING(REPLACE(DATE, '"', ''), 5, 2) AS INT) IS NOT NULL;
```

3. Store ID has values 306, 307 which are out of range in the Store IDs range given in the data manual and there is no information about these stores, so this SQL Task removes Stores which are not in the specified list of stores from the manual

```
DELETE FROM Stage_Sales_v3
WHERE (STORE NOT IN ('2', '4', '5', '8', '9', '12', '14', '18', '19', '21', '25', '28', '32',
'33', '39', '40', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '59',
'60', '62', '64', '65', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76', '77', '78', '80',
'81', '83', '84', '86', '88', '89', '90', '91', '92', '93', '94', '95', '97', '98', '100',
        '101', '102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '112',
'113', '114', '115', '116', '117', '118', '119', '121', '122', '123', '124', '126', '128',
'129', '130', '131', '132', '133', '134', '136', '137', '139'))
```

4. Few DATE columns are populated as NULL which are out of range of the date specified, i.e, 1st January 1988 to 31st December 1996. Those dates have to be removed.

```
DELETE FROM Stage_Sales_v3
WHERE ([DATE] IS NULL)
```

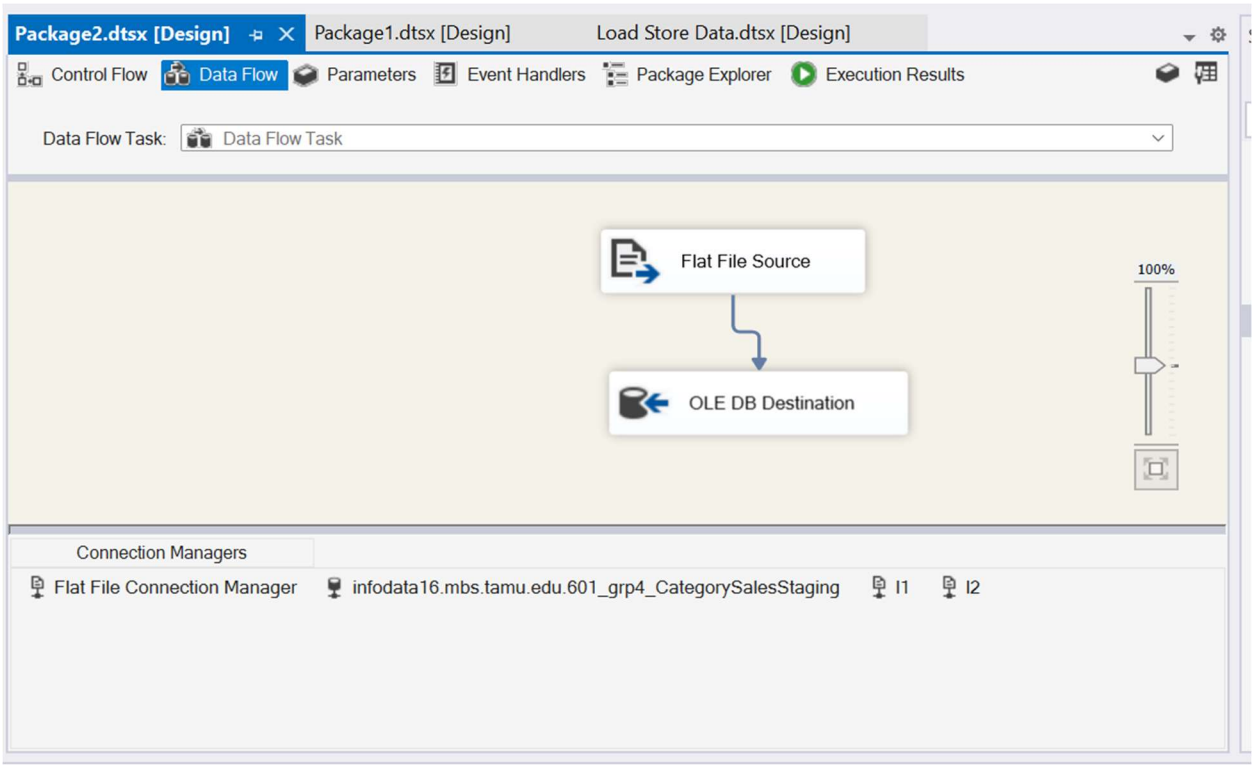


Fig 6. Data Flow Task to import data

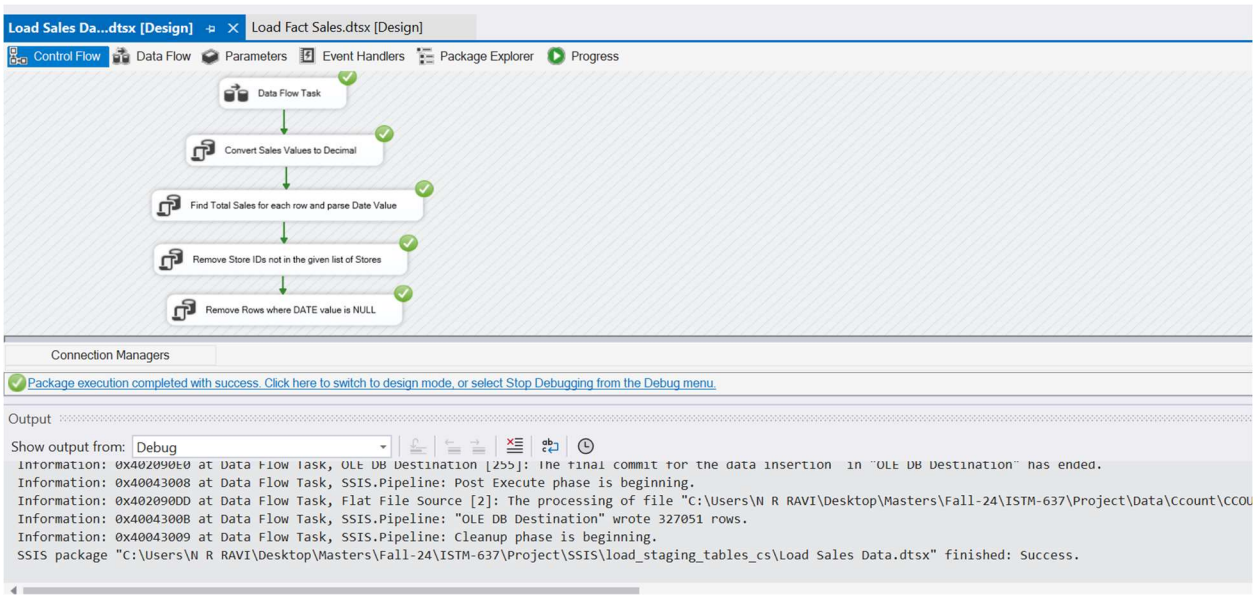


Fig 7. Execution of Data Flow task and the 4 SQL tasks

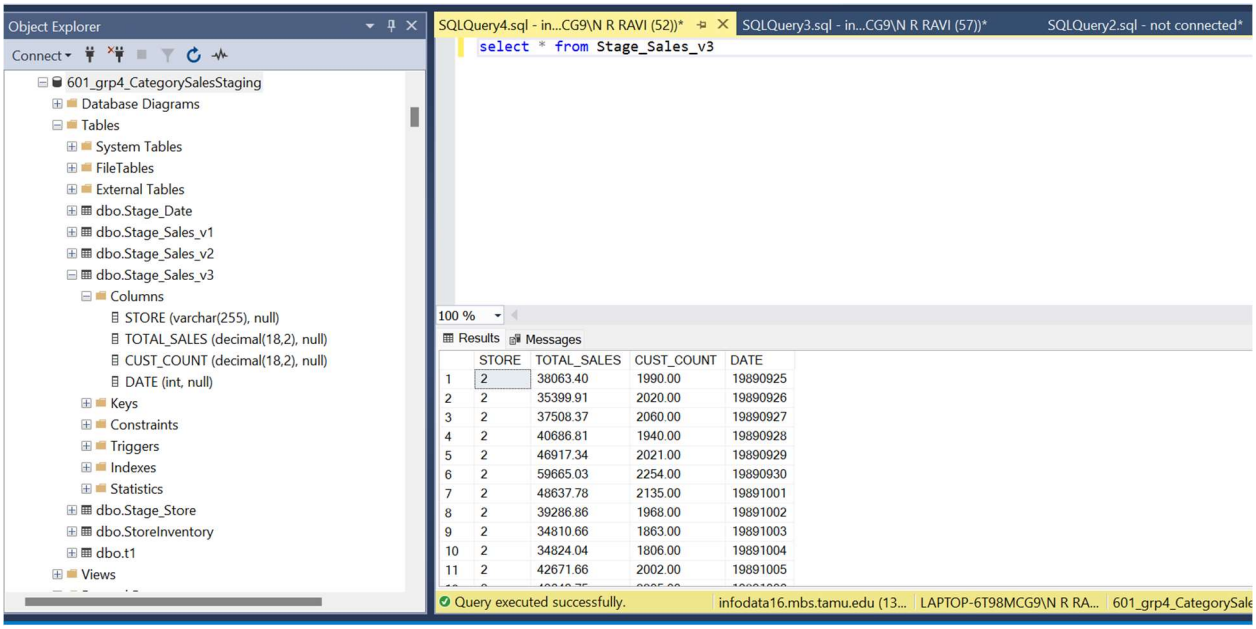


Fig 8. Fact table populated in Staging DB

- ## Organization of Data Staging Area :-

  Database: A separate database named [601_grp4_CategorySalesStaging].

  Staging Tables:

  - Stage_Store
  - Stage_Sales_v3
  - Stage_Date

  Temporary Tables:

- Stage_Sales_v1
- Stage_Sales_v2

- **Procedures for Data Extractions and Loadings :-**
  - Run the SSIS packages to load the data into Staging tables
  - Apply transformations during the control and data flow.

- **ETL for Dim Tables :-**
  - **DimStore :-**

Loading data into DimStore from Stage_Store using SSIS

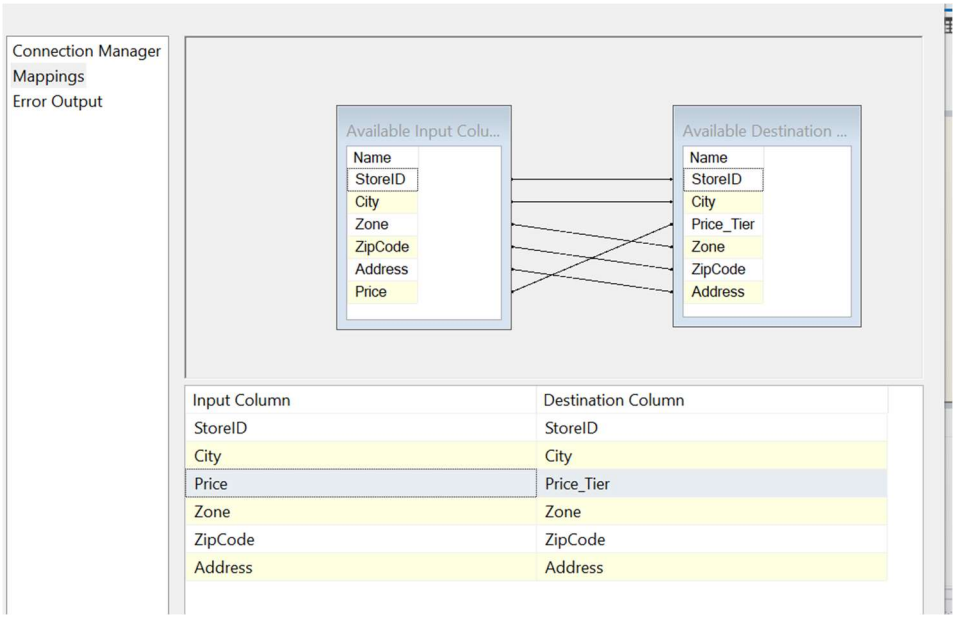


Fig 9. Mapping from Staging to Dim Store table

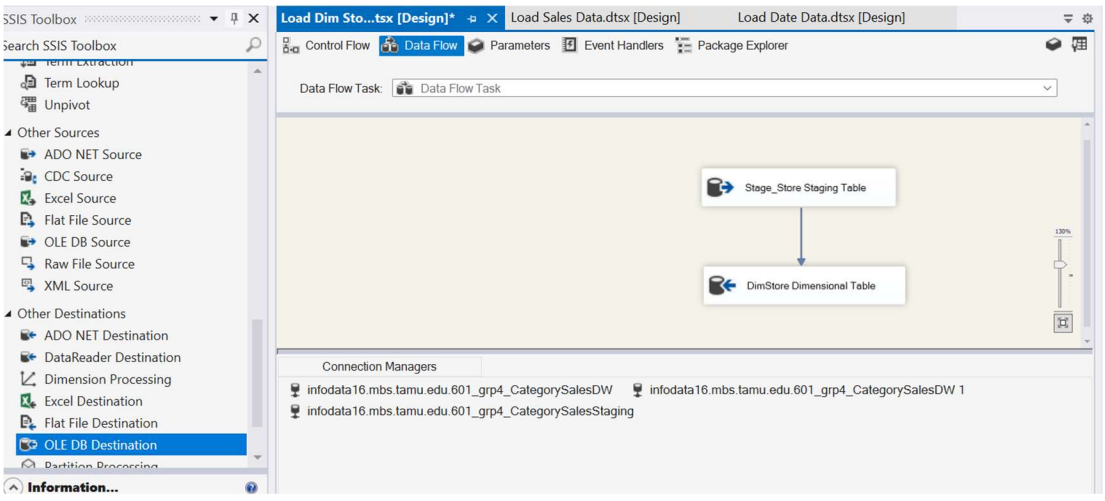


Fig 10. Data Flow task to copy from Staging to Dim Store table

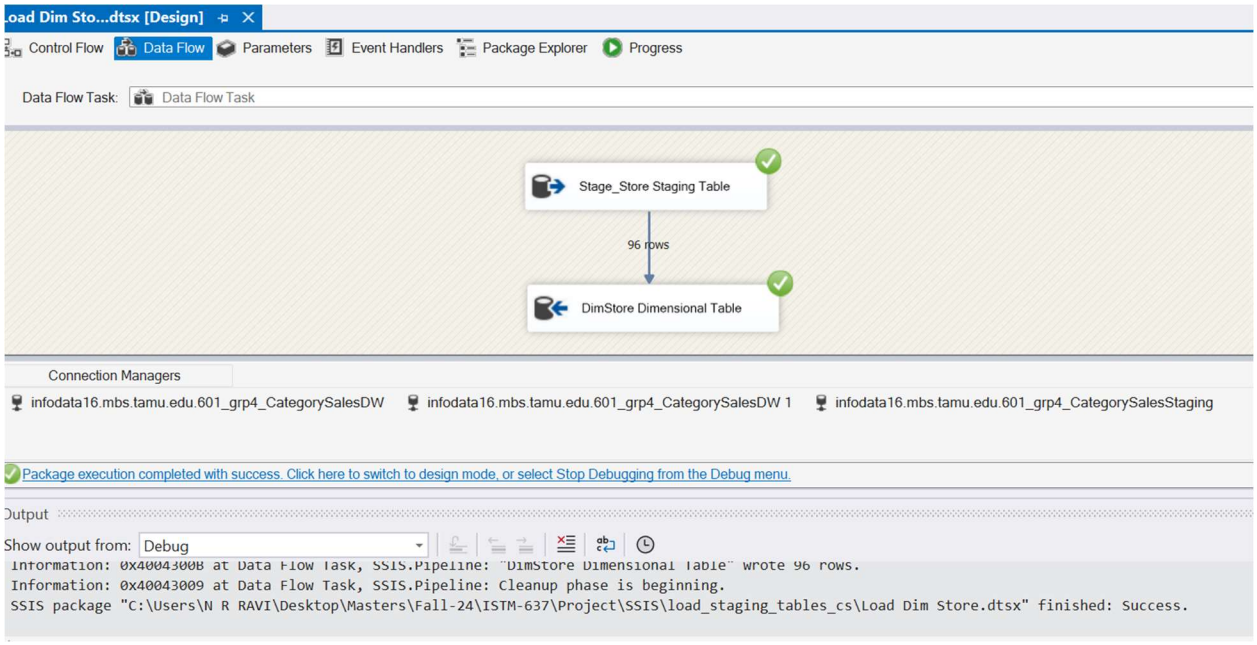
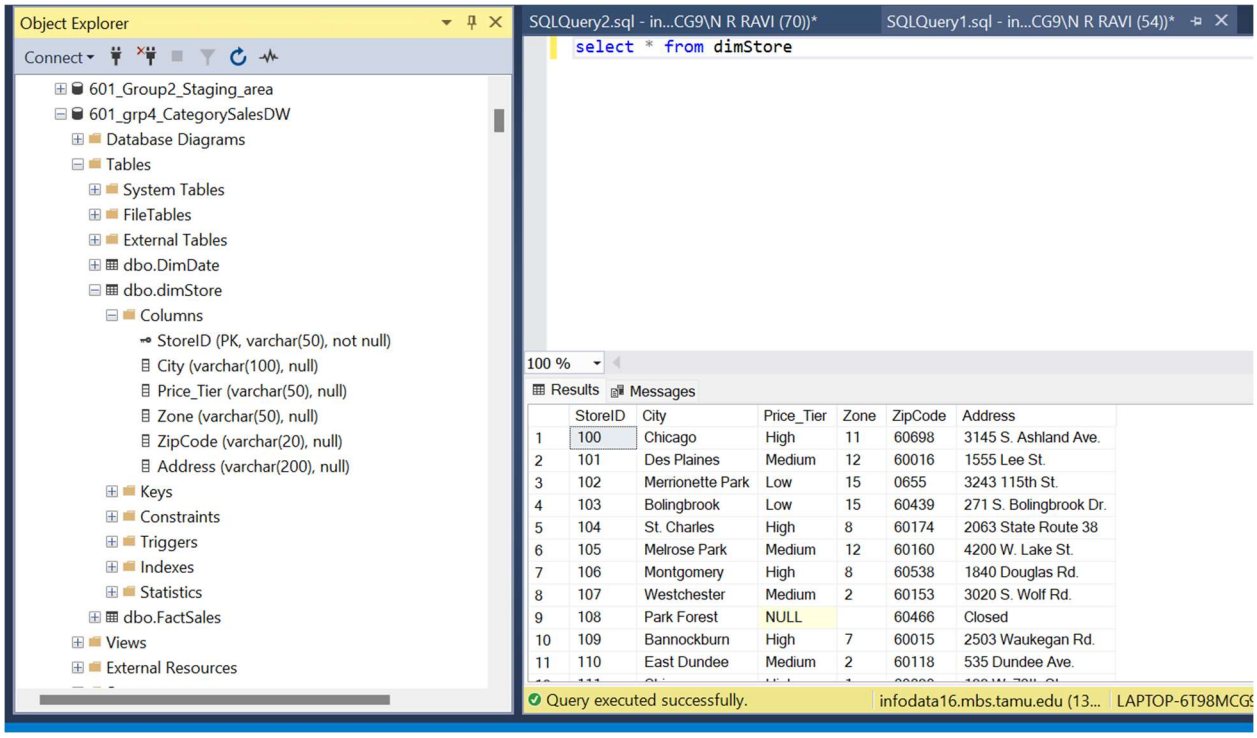


Fig 11. Copy task for Store successfully executed



	StoreID	City	Price_Tier	Zone	ZipCode	Address
1	100	Chicago	High	11	60698	3145 S. Ashland Ave.
2	101	Des Plaines	Medium	12	60016	1555 Lee St.
3	102	Merrionette Park	Low	15	0655	3243 115th St.
4	103	Bolingbrook	Low	15	60439	271 S. Bolingbrook Dr.
5	104	St. Charles	High	8	60174	2063 State Route 38
6	105	Melrose Park	Medium	12	60160	4200 W. Lake St.
7	106	Montgomery	High	8	60538	1840 Douglas Rd.
8	107	Westchester	Medium	2	60153	3020 S. Wolf Rd.
9	108	Park Forest	NULL		60466	Closed
10	109	Bannockburn	High	7	60015	2503 Waukegan Rd.
11	110	East Dundee	Medium	2	60118	535 Dundee Ave.

Fig 12. DimStore Populated with data

- **DimDate :-**

  Loading Data from Stage_Date to Dim_Date using SSIS

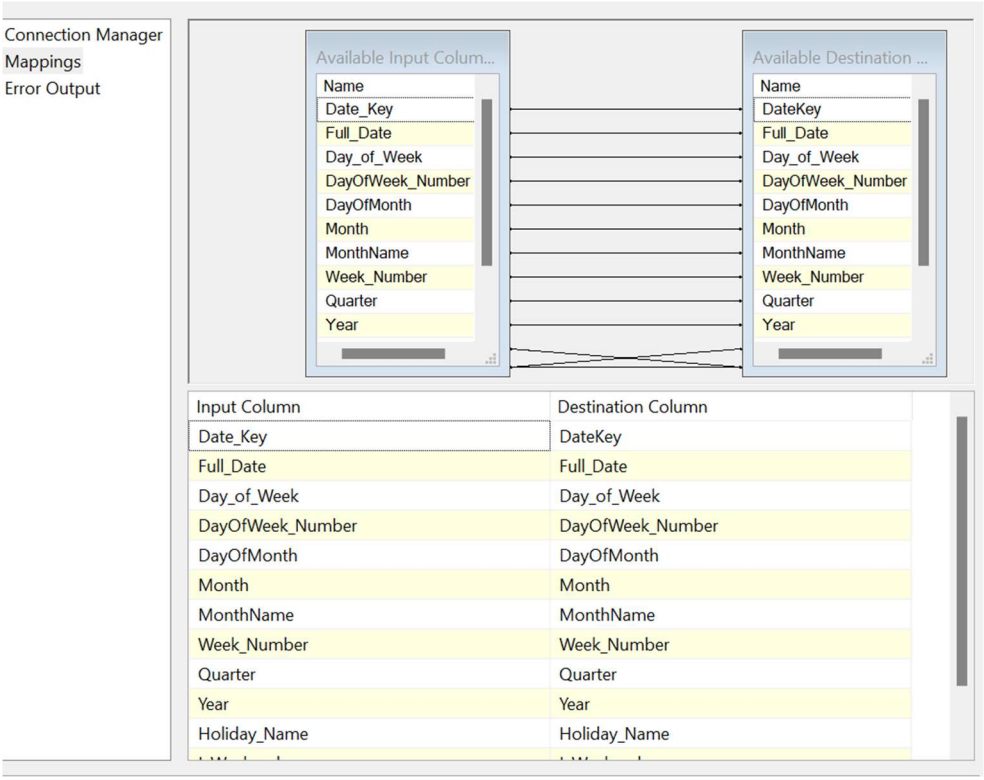


Fig 13. Mapping from Stage Date to Dim Date

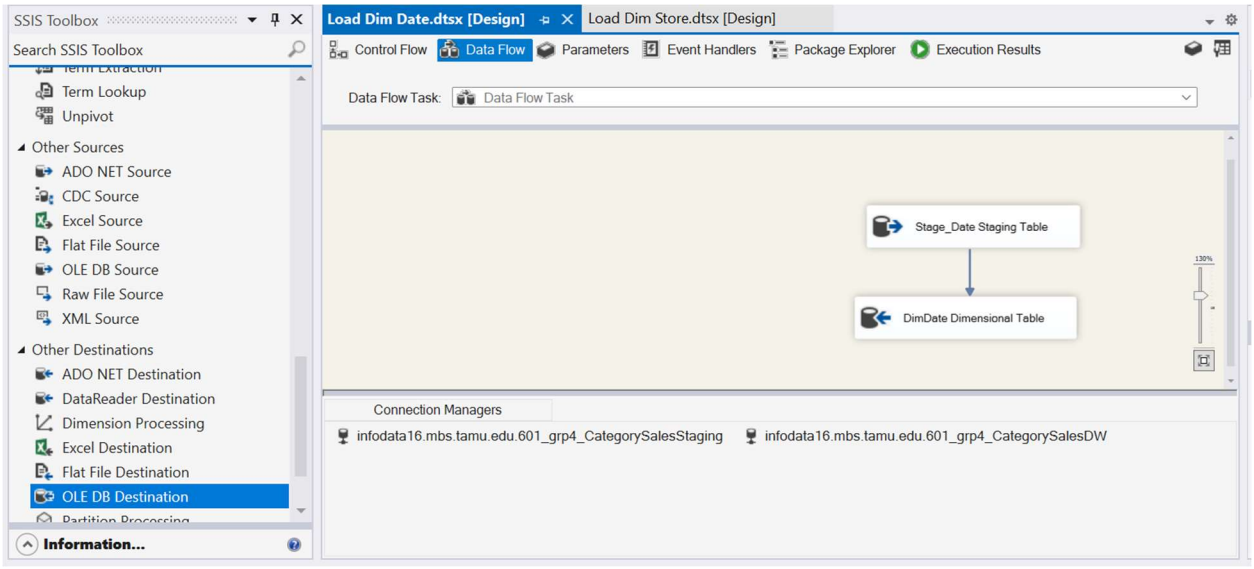


Fig 14. Data Flow Task to copy data

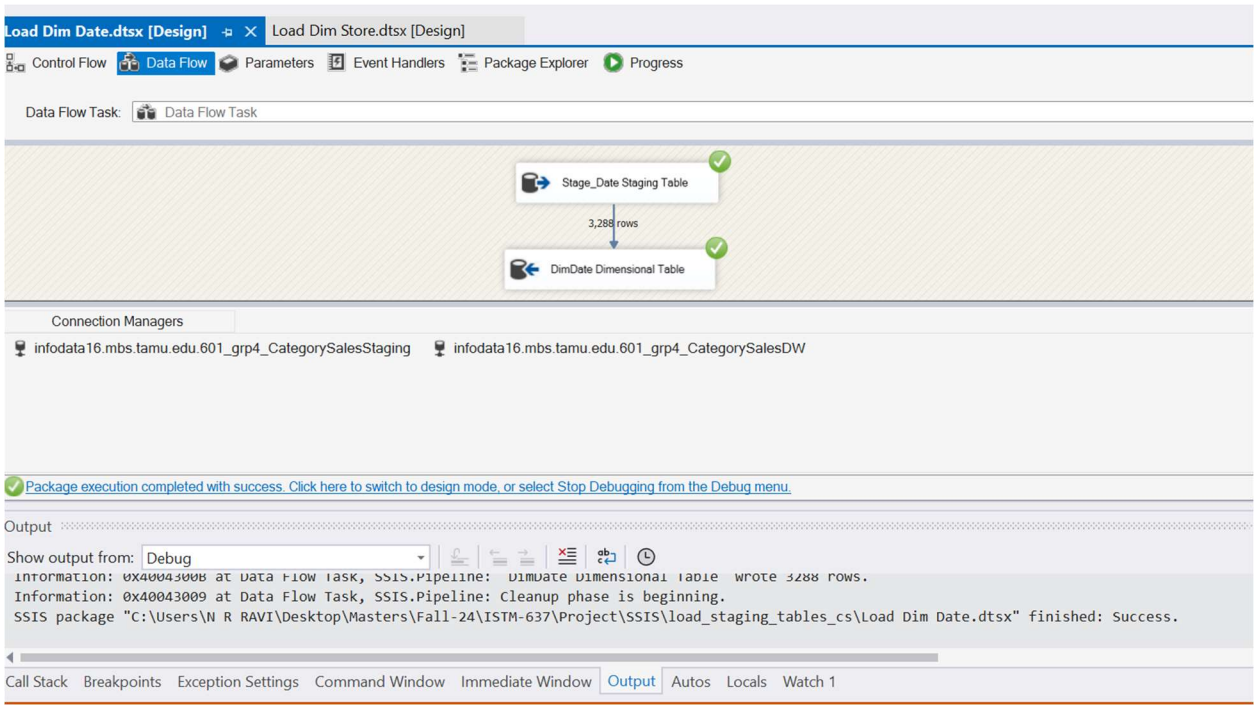


Fig 15. Execution of Task to Copy

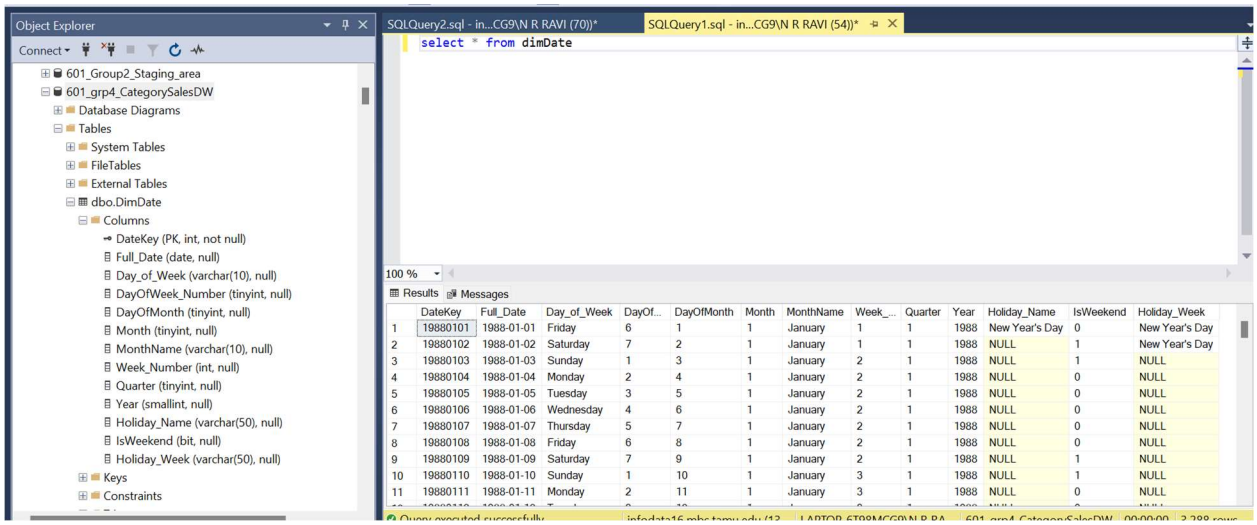


Fig 16. DimDate table populated

- **ETL for Fact Tables :-**
  - **FactSales :-**

    Load Stage_Sales Data to FactSales using SSIS

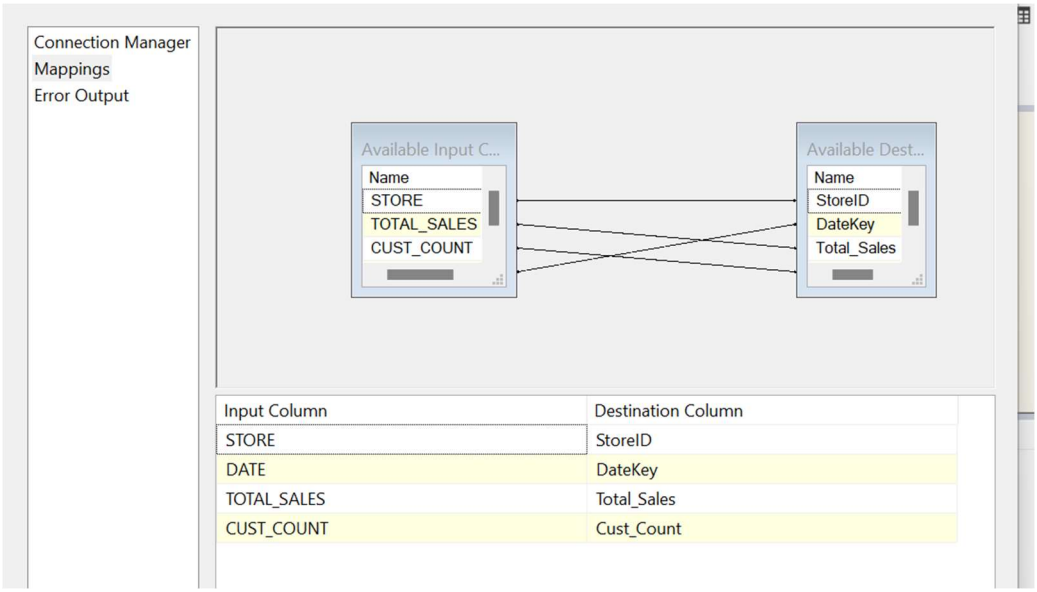


Fig 17. Stage Sales to Fact Sales Mapping

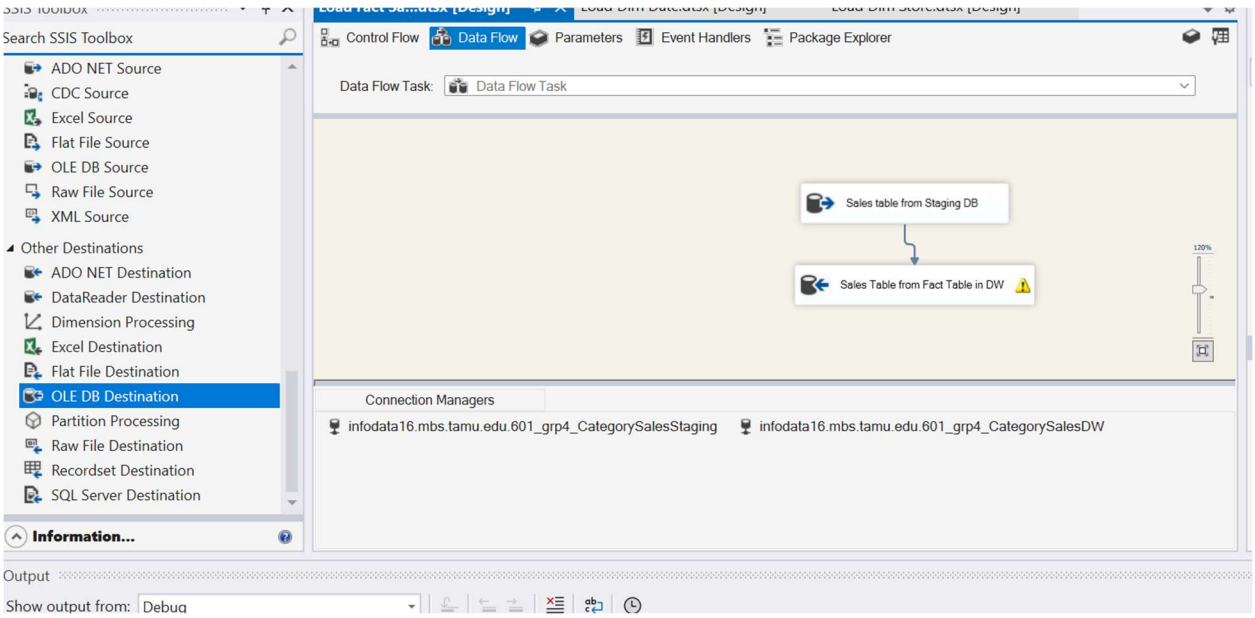


Fig 18. Data Flow tasks to copy data

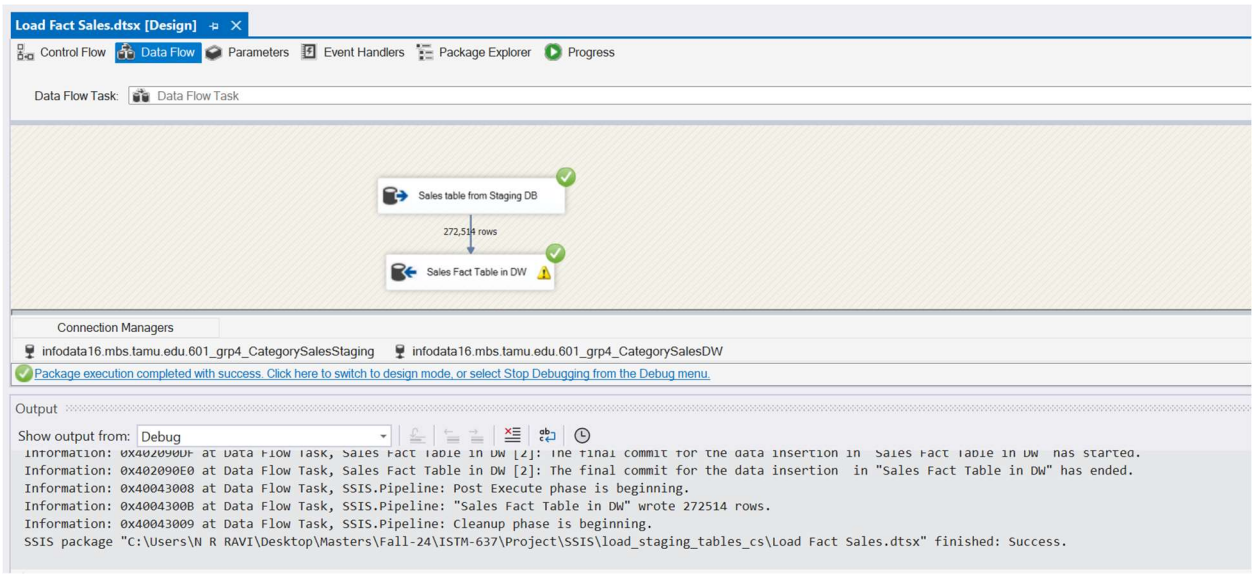


Fig 19. Execution of Load from Staging DB to DW DB for Fact table

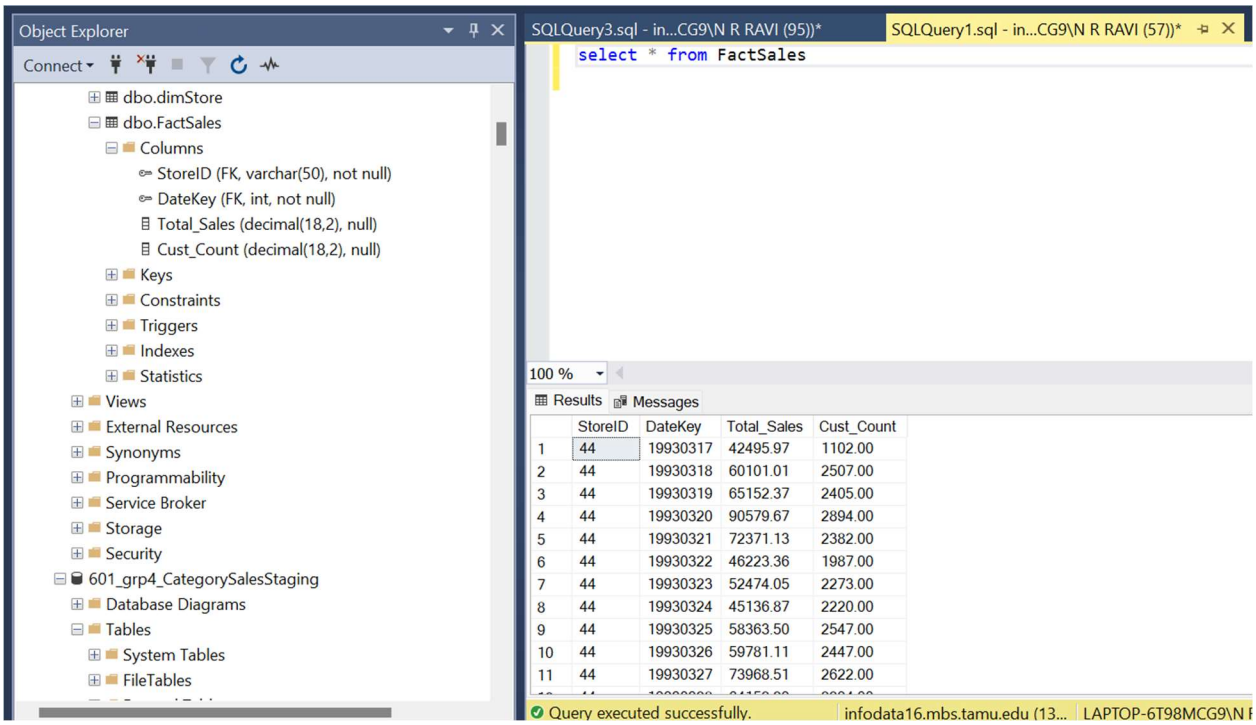


Fig 20. Fact Table Populated in the Data Warehouse

ETL Pipeline Development and Implementation Report for Product Sales DataMart

- ## Target Data Needed in the Data Warehouse :-

  The data warehouse will consist of the following target tables:

  - **Dimension Tables**:
    - **DimProduct**: Contains product information.
      - Attributes: UPC, Description, Size, CasePack
    - **DimStore**: Contains store-related information.
      - Attributes: StoreID, City, Price_Tier, Zone, ZipCode, Address
    - **DimDate**: Contains date-related information.
      - Attributes: TimeKey, Date, WeekNumber, Month, Quarter, Year
  - **Fact Table**:
    - **FactSales**: Sales and movement data
      - SalesKey, UPC, StoreID, TimeKey, UnitsSold, TotalSales, ProfitMargin, GrossMargin, RetailPrice, Quantity

- ## Data Sources (Available from DFF) :-

  The following data sources are available from the Data Feed Files (DFF):

  - **movement*.csv:** Contains sales transactions data with attributes like upc, store, week, move, price, qty, profit, sale, ok.
  - **upc*.csv:** Contains product details with attributes like upc, com_code, nitem, descrip, size, case.
  - **StoreDetails.csv:** Contains store information with attributes like StoreID, City, Price, Tier, Zone, Zip, Address.

- ## Data Mappings :-

  Mapping from Source Files to Staging Tables :-

  - **Source Files to Stage_UPC_v1**

Source File	Source Attribute	Transformation	Staging Attribute
upc*.csv	Upc	Copy	upc
upc*.csv	com_code	Copy	com_code
upc*.csv	Nitem	Copy	nitem
upc*.csv	descrip	Copy	descrip

upc*.csv	Size	Copy	size
upc*.csv	Case	Copy	case

- **Source Files to Stage_Store**

Source File	Source Attribute	Transformation	Staging Attribute
upc*.csv	upc	Copy	upc
upc*.csv	com_code	Copy	com_code
upc*.csv	nitem	Copy	nitem
upc*.csv	descrip	Copy	descrip
upc*.csv	size	Copy	size
upc*.csv	case	Copy	case

- **Source Files to Stage_Movement_v1**

Source File	Source Attribute	Transformation	Staging Attribute
movement*.csv	upc	Copy	upc
movement*.csv	store	Copy	store
movement*.csv	week	Copy	week
movement*.csv	move	Copy	move
movement*.csv	price	Copy	price
movement*.csv	qty	Copy	qty
movement*.csv	profit	Copy	profit
movement*.csv	sale	Copy	sale

- **Stage_UPC_v1 to Stage_UPC_v2**

Source File	Source Attribute	Transformation	Staging Attribute
Stage_UPC_v1	upc	Copy	upc
Stage_UPC_v1	descrip	Copy	description
Stage_UPC_v1	size	Copy	size
Stage_UPC_v1	case	Copy	case

- **Stage_Movement_v1 to Stage_Movement_v2**

Source File	Source Attribute	Transformation	Staging Attribute
Stage_Movement_v1	upc	Copy	upc
Stage_Movement_v1	store	Copy	store
Stage_Movement_v1	week	Convert to INT	week
Stage_Movement_v1	move	Convert to Decimal	move
Stage_Movement_v1	price	Convert to Decimal	price

Stage_Movement_v1	qty	Convert to INT	qty
Stage_Movement_v1	profit	Convert to Decimal	Profit
Stage_Movement_v1	sale	Copy	Sale
Stage_Movement_v1	sale	profit * 100	profit_margin
Stage_Movement_v1	sale	price * move / qty	Sales

- ## Data Extraction Rules :-
  - **For DimStore :-**

    1. Store Details City, Price_Tier, Zone, Zip, Address are present in the DFF Data Manual, it is copied to csv Store.csv

    2. Store.csv data is stored in staging table Stage_Store in [601_grp4_ProductSalesStaging] DB by copying all the columns from csv

  - **For DimProduct :-**
    5. upc*.csv has all the data about each product, its description size etc.
    6. Data from the csv is copied to table Stage_UPC_v1 in [601_grp4_ProductSalesStaging] DB by copying all the columns.
  - **For FactSales :-**
    7. movement*.csv has all the data about each product, its description size etc.
    8. Data from the csv is copied to table Stage_Movement_v1 in [601_grp4_ProductSalesStaging] DB by copying all the columns.

- ## Data Transformation and Cleansing Rules :-
  - **For DimProduct :-**
    1. Used MultiFlatFile Connection to import all the UPC files at once

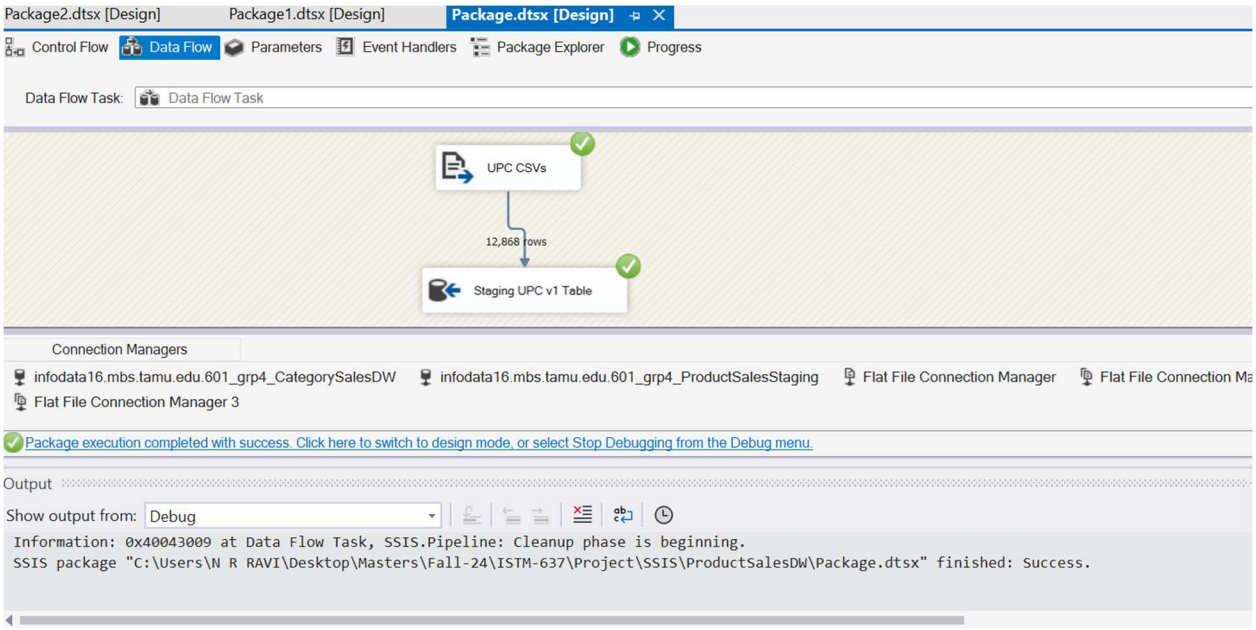


Fig 21. Import UPC data from source files to Stage_UPC_v1

2. Drop columns nitems and com_code and copy all the other columns.

```
INSERT INTO Stage_UPC_v2 (UPC, DESCRIPTION, SIZE, [CASE])
SELECT
   UPC,
   CASE
      WHEN DESCRIP = '"""' THEN NULL
      ELSE DESCRIP
   END AS DESCRIPTION,
   CASE
      WHEN SIZE = '"""' THEN NULL
      ELSE SIZE
   END AS SIZE,
   CASE
      WHEN [CASE] = '"""' THEN NULL
      ELSE [CASE]
   END AS [CASE]
FROM
   Stage_UPC_v1
-- If UPC is NULL ignore those rows of data
WHERE
   UPC <> '"""';
```

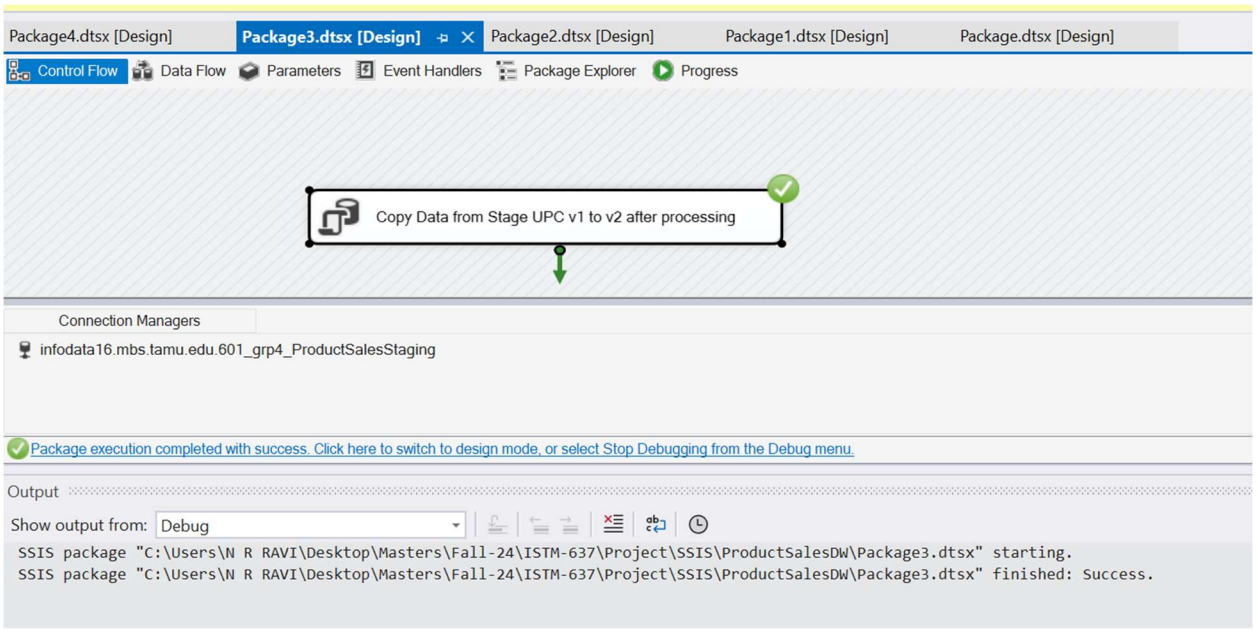


Fig 22. Copy data from v1 staging to v2 with transformation

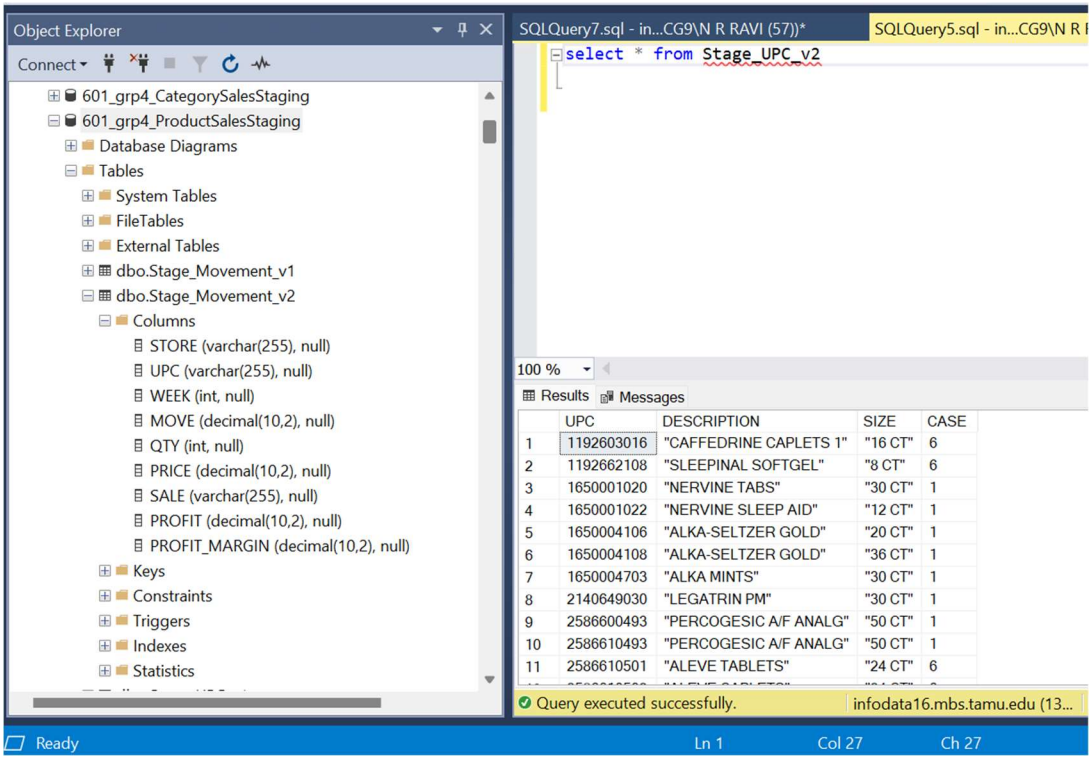


Fig 23. Stage UPC v2 table populated

- **For DimStore :-**
  1. Remove “ character from Store ID and Address columns using derived Column in Data Flow

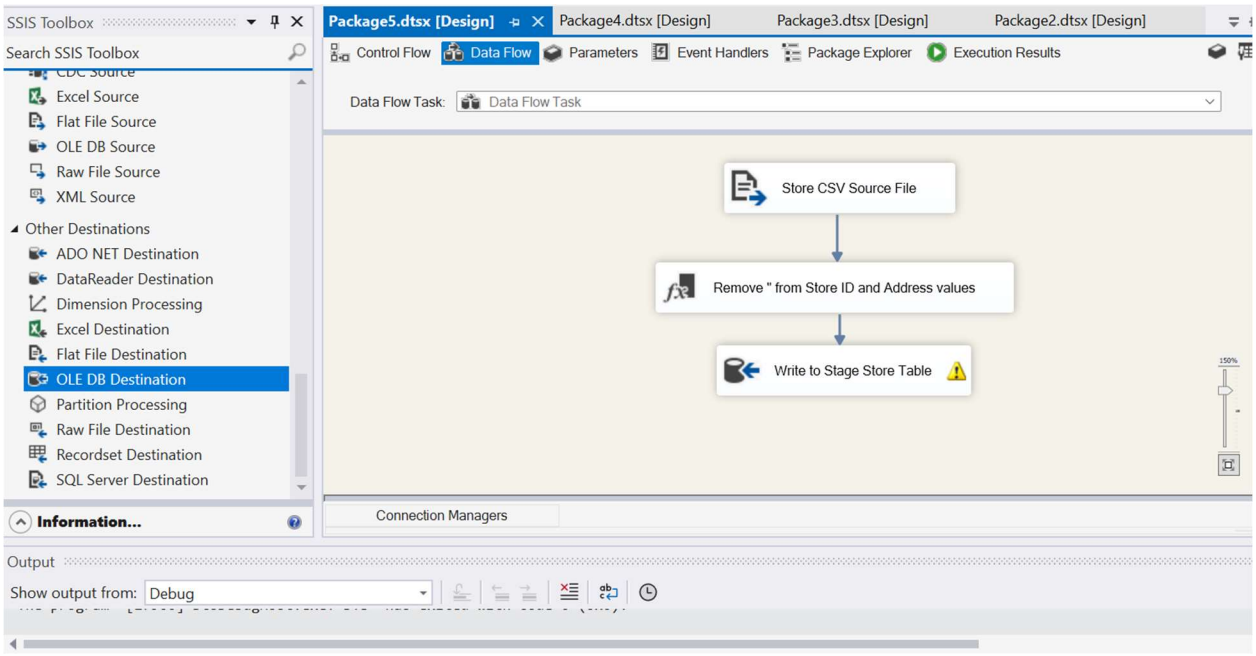


Fig 24. Data Flow to write data to Stage Store Table

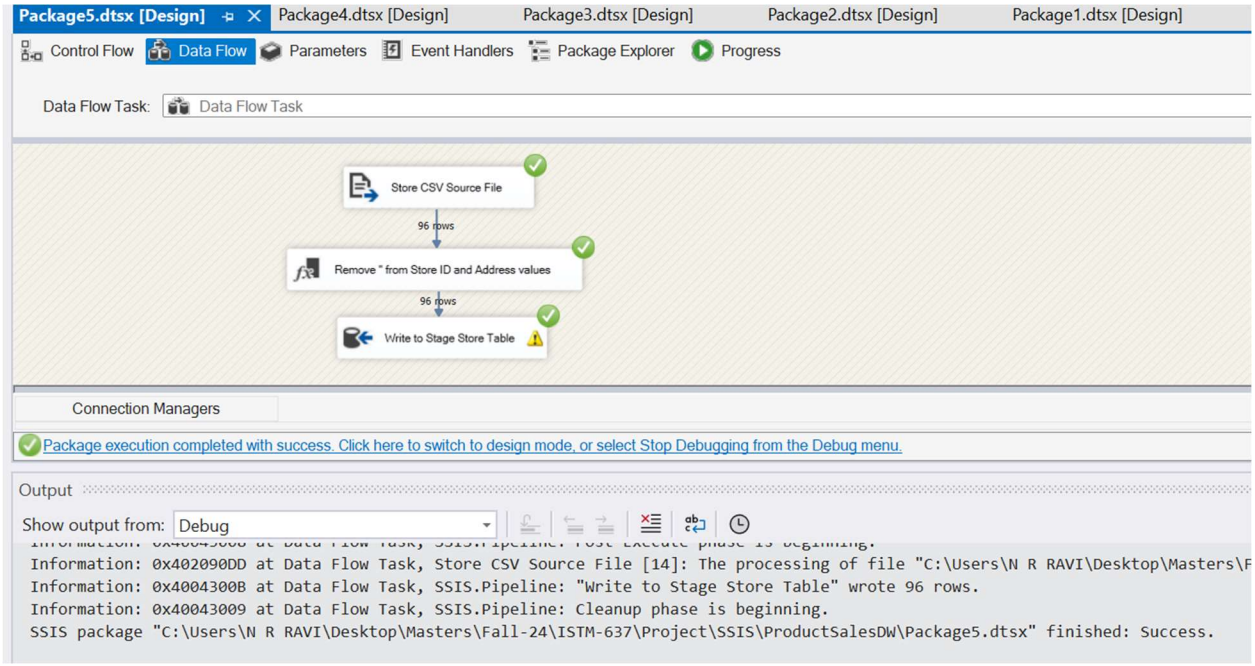


Fig 25. Execution of pipeline

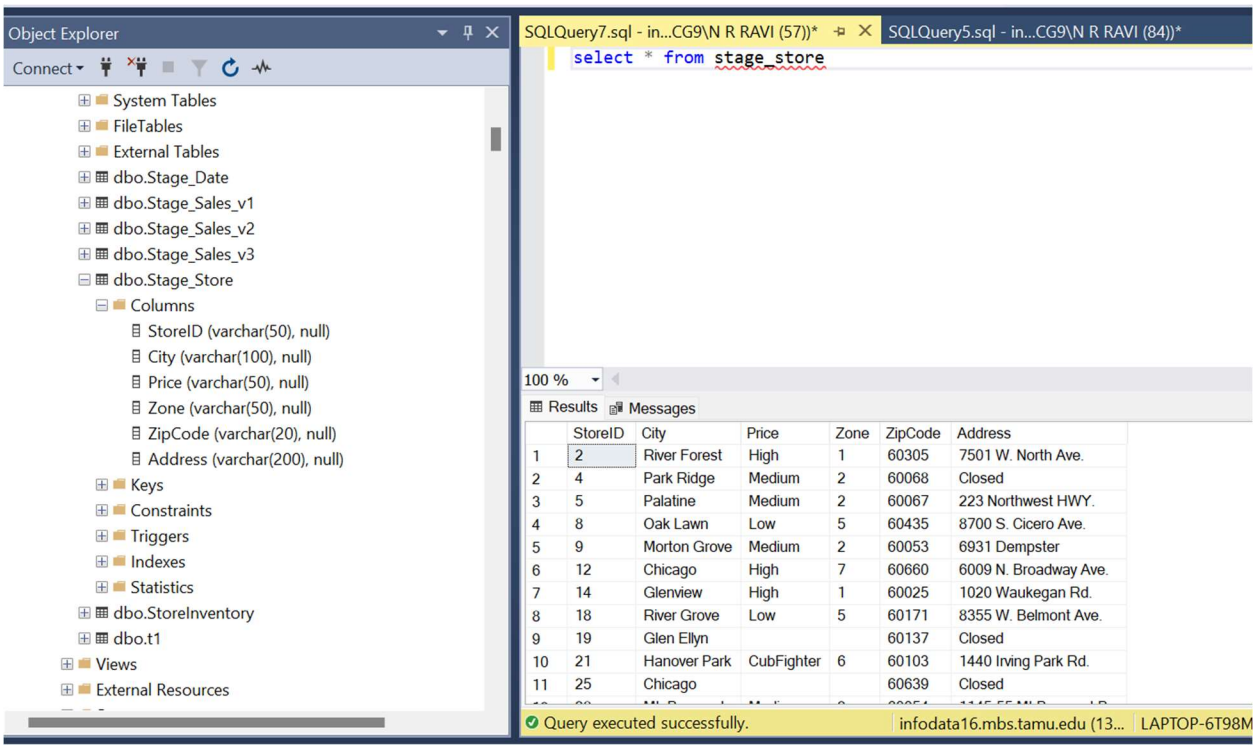


Fig 26. Stage Store table populated

- **For DimDate :-**
  1. Stage_Date is populated from week 1-400 (that is the range in movement files). Week 1 is considered as 1st week for year 1988 and so on

```
-- Consider the 1st week of 1988 as week 1
DECLARE @StartDate DATE = '1988-01-04';

INSERT INTO Stage_Date (Date, WeekNumber, Month, Quarter, Year)
SELECT
   -- 1st day of the week as DATE
   DATEADD(WEEK, WeekNumber - 1, @StartDate) AS Date,
   WeekNumber,
   -- Month and Year derived from week number
   MONTH(DATEADD(WEEK, WeekNumber - 1, @StartDate)) AS Month,
   DATEPART(QUARTER, DATEADD(WEEK, WeekNumber - 1, @StartDate)) AS Quarter,
   YEAR(DATEADD(WEEK, WeekNumber - 1, @StartDate)) AS Year
FROM
   (SELECT ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS WeekNumber
    FROM master..spt_values) AS Weeks
WHERE
   -- Range of weeks present in the data
   WeekNumber BETWEEN 1 AND 400;
```

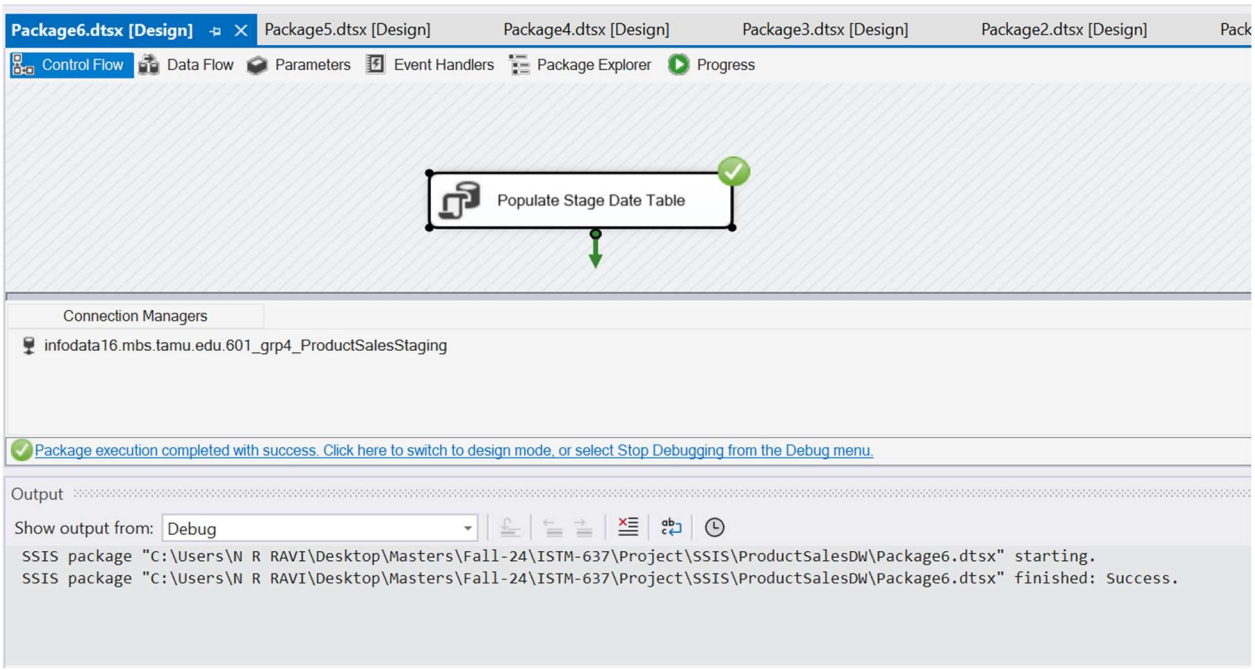


Fig 27. Control Flow Task to create date data

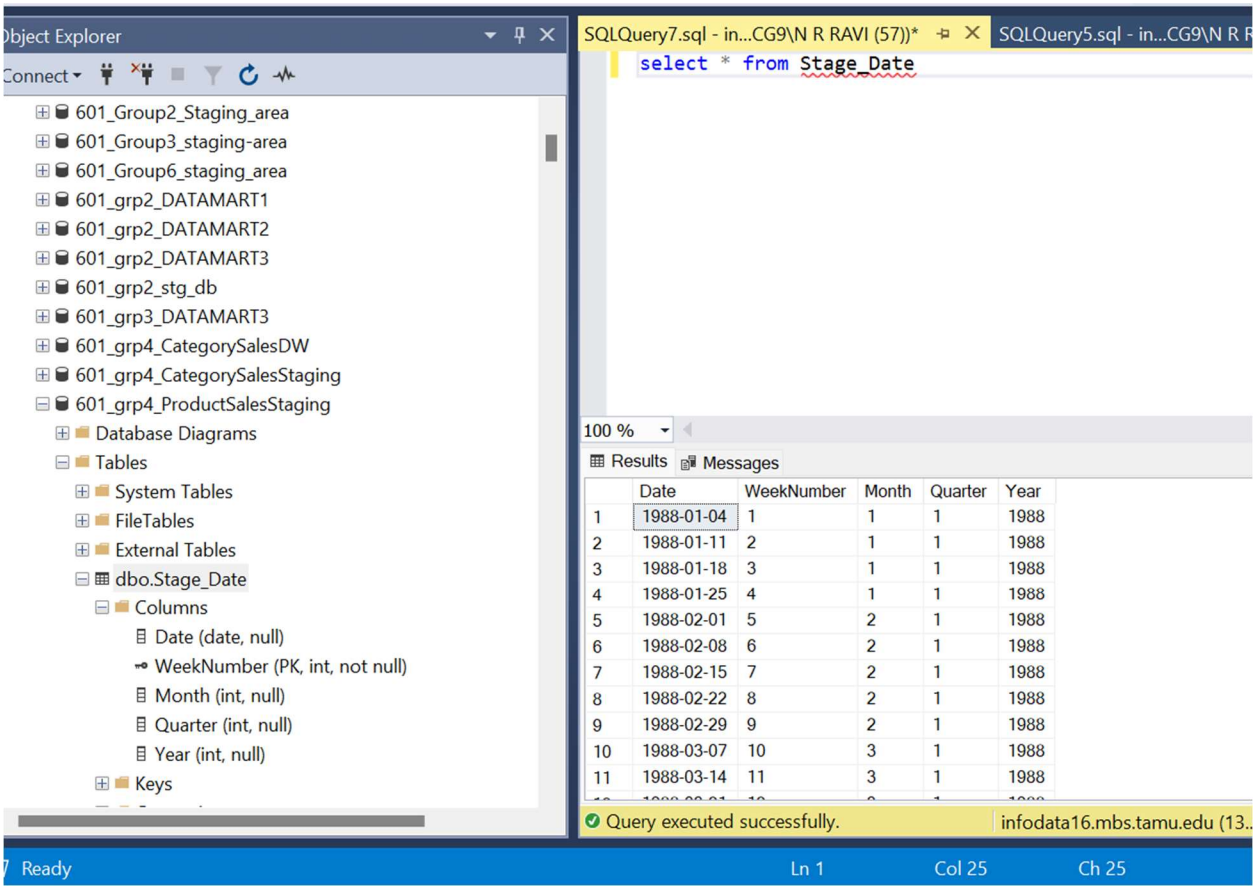


Fig 28. Stage Date table populated

- **For FactSales :-**
    1. Used MultiFlatFile connection to import all the movement files at once and copied to Stage_Movement_v1 table
    2. Movement Files have different schemas, 2 types of schemas were observed, 2 separate multiflatfile load was done

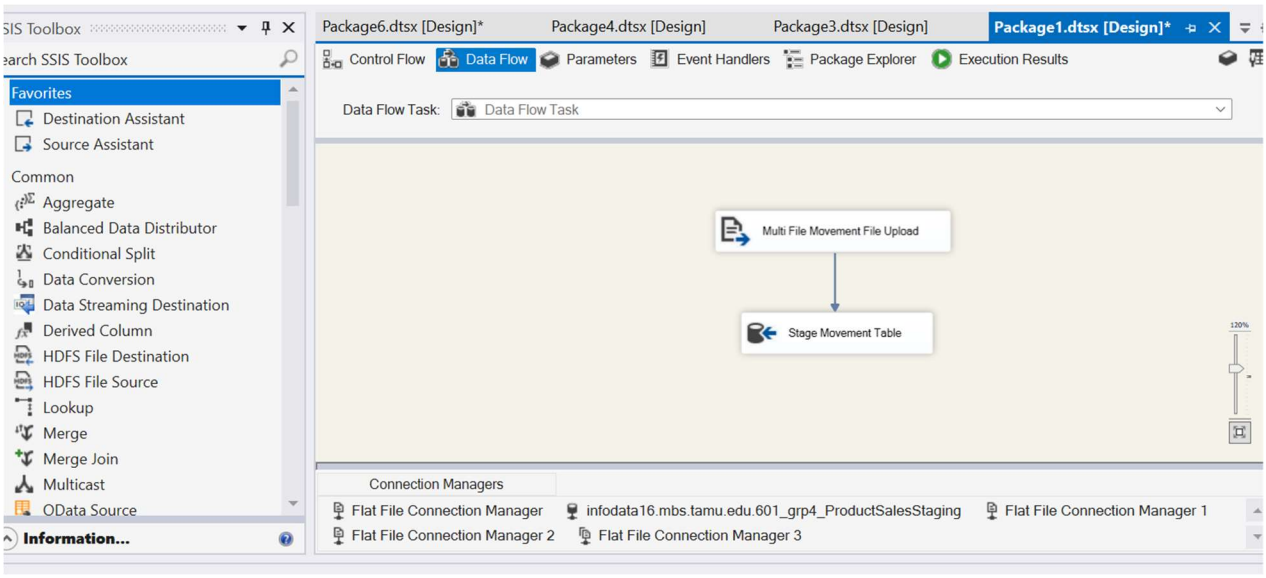


Fig 29. Multi Flat File upload of movement data to Stage Movement v1

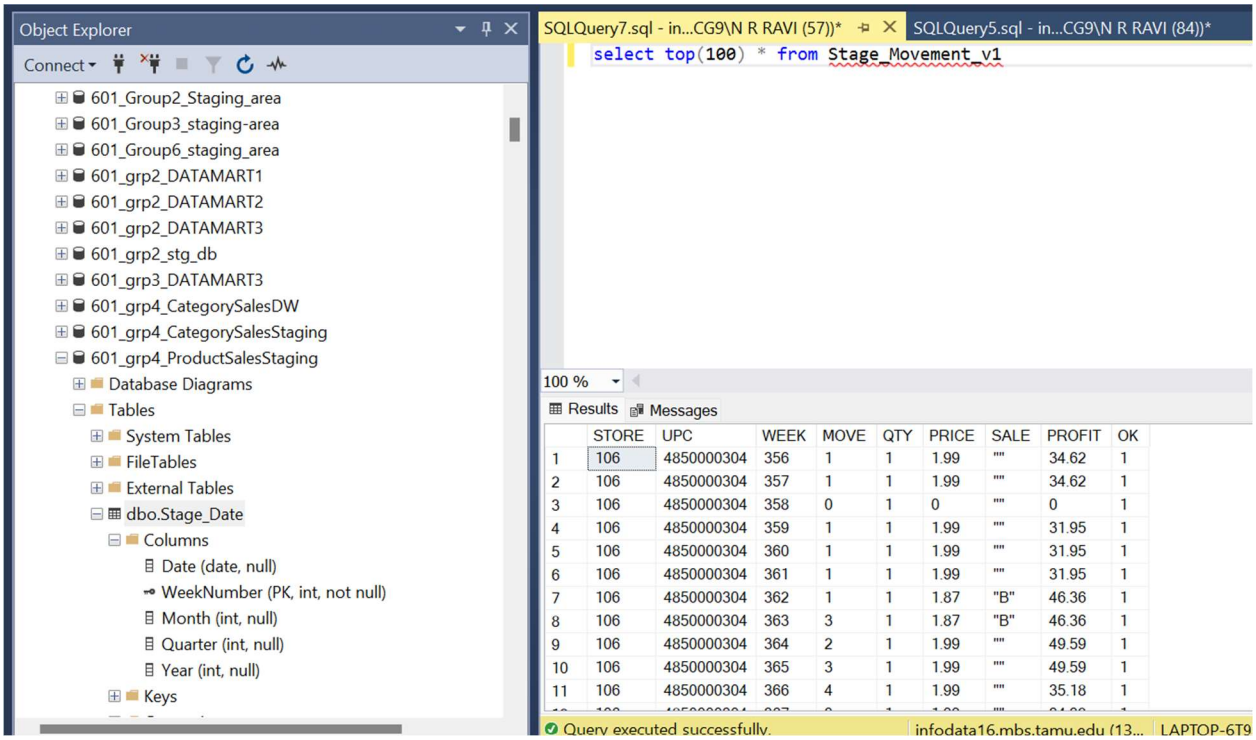


Fig 30. Stage Movement v1 populated

3. Checking columns STORE, UPC, WEEK, if the values within them are null or empty string, these columns are dropped. Also Store column with value "." And UPC column with value 2 is dropped which observed in the v1 dataset but not present in the dim Tables.

```
DELETE FROM Stage_Movement_v1
WHERE (STORE = '.') OR
(STORE IS NULL) OR
(STORE = '') OR
(UPC IS NULL) OR
(UPC = '2') OR
(UPC = '') OR
(WEEK = '') OR
(WEEK IS NULL)
```

4. Columns MOVE, SALE, PRICE, QTY, PROFIT are processed. Every column if updated to remove character " from it. Sale is observed to have values 0 and 1, which are dropped to null, MOVE, PROFT, QTY and PRICE are updated if they have null string values it is updated to 0.

```
UPDATE Stage_Movement_v1

SET

SALE = CASE WHEN SALE = '' OR

SALE = '0' OR

SALE = '1' THEN NULL ELSE SALE END,

MOVE = CASE WHEN [MOVE] = '' THEN '0' ELSE [MOVE] END,

QTY = CASE WHEN QTY = '' THEN '0' ELSE QTY END,

PRICE = CASE WHEN PRICE = '' THEN '0' ELSE PRICE END,

PROFIT = CASE WHEN PROFIT = '' THEN '0' ELSE PROFIT END,

OK = CASE WHEN OK = '' THEN NULL ELSE OK END
```

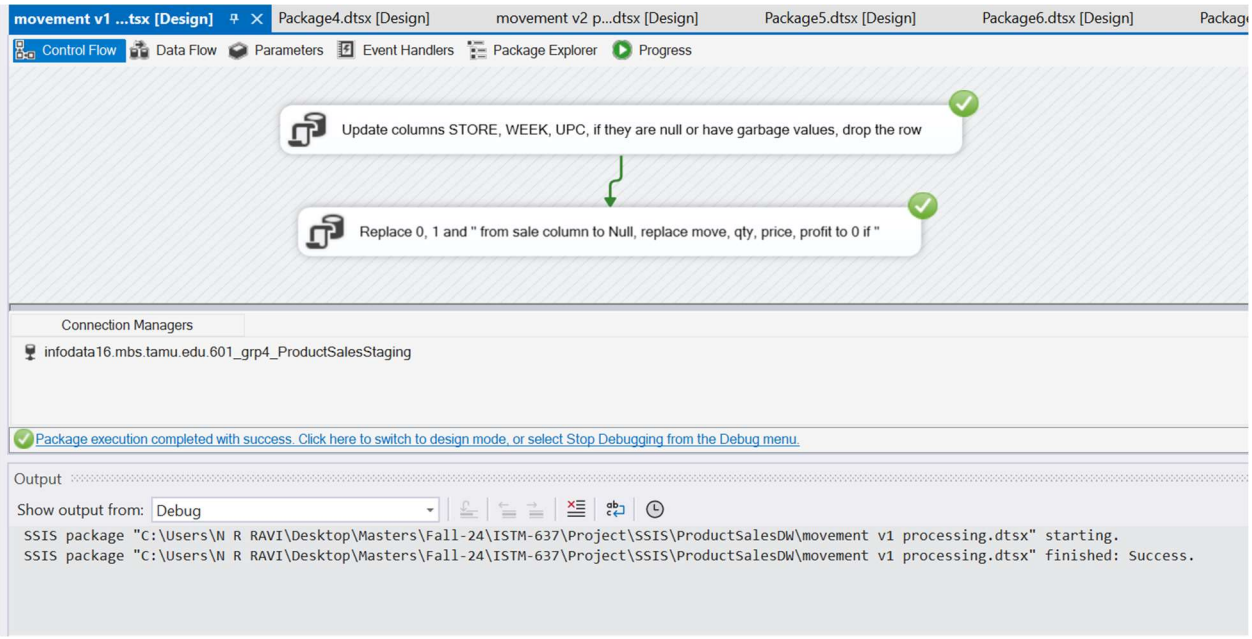


Fig 31. Execution of Stage Movement v1 transformation

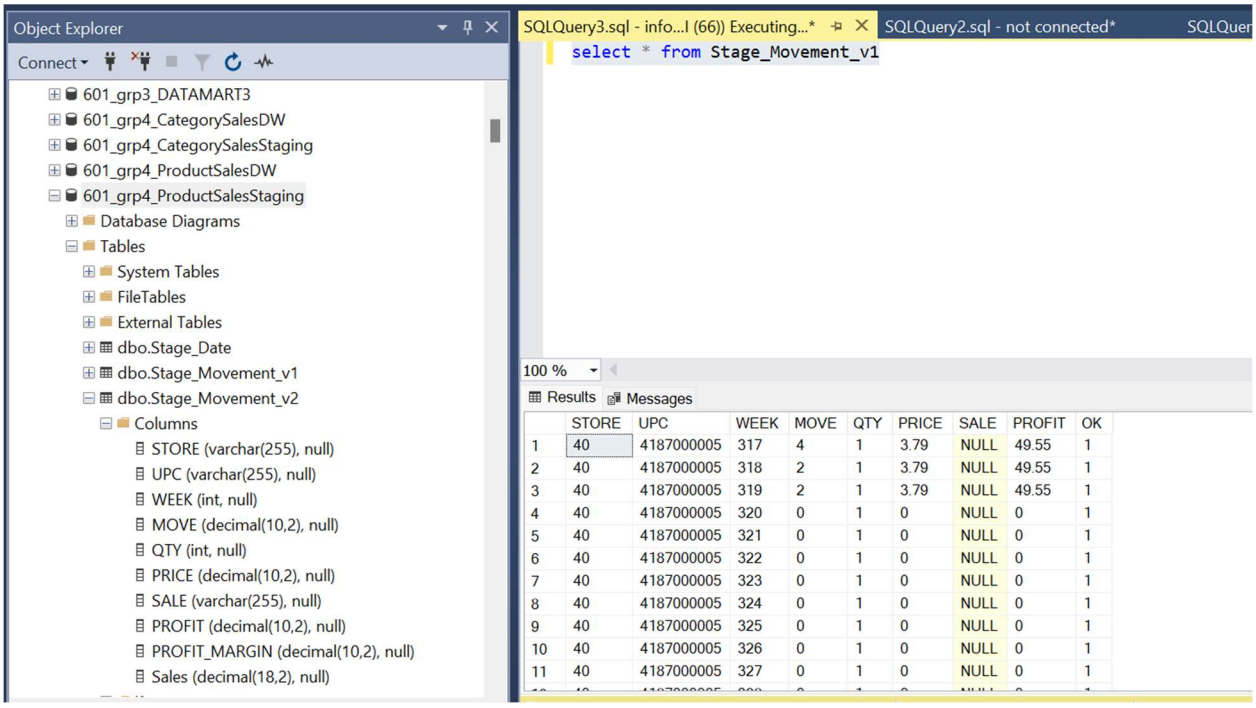


Fig 32. Stage Movement v1 after processing

5. Copy data from Movement v1 to Movement v2, cast WEEK, QTY to INT, PRICE, MOVE, add columns PROFIT_MARGIN and SALES

```
INSERT INTO Stage_Movement_v2 (STORE, UPC, WEEK, [MOVE], QTY, PRICE, SALE, PROFIT,
PROFIT_MARGIN, Sales)

SELECT
```

```
        STORE,

        UPC,

        CASE WHEN WEEK IS NOT NULL AND TRY_CAST(WEEK AS INT) IS NOT NULL THEN
CAST(WEEK AS INT) ELSE NULL END AS WEEK,

        CASE WHEN [MOVE] IS NOT NULL AND TRY_CAST([MOVE] AS DECIMAL(10, 2)) IS NOT
NULL THEN CAST([MOVE] AS DECIMAL(10, 2))  ELSE NULL END AS [MOVE],

        CASE WHEN QTY IS NOT NULL AND TRY_CAST(QTY AS INT) IS NOT NULL THEN CAST(QTY
AS INT) ELSE NULL END AS QTY,

        CASE WHEN PRICE IS NOT NULL AND TRY_CAST(PRICE AS DECIMAL(10, 2)) IS NOT NULL
THEN CAST(PRICE AS DECIMAL(10, 2)) ELSE NULL END AS PRICE,

        SALE,

        CASE WHEN PROFIT IS NOT NULL AND TRY_CAST(PROFIT AS DECIMAL(10, 2)) IS NOT
NULL THEN CAST(PROFIT AS DECIMAL(10, 2)) ELSE NULL END AS PROFIT,

        CASE WHEN PROFIT IS NOT NULL AND TRY_CAST(PROFIT AS DECIMAL(10, 2))

      IS NOT NULL THEN CAST(PROFIT AS DECIMAL(10, 2)) * 100 ELSE NULL END AS
PROFIT_MARGIN,

        CASE WHEN QTY IS NOT NULL AND TRY_CAST(QTY AS INT) = 0 THEN 0 WHEN PRICE IS
NOT NULL AND [MOVE] IS NOT NULL AND QTY IS NOT NULL AND TRY_CAST(PRICE AS
DECIMAL(10, 2)) IS NOT NULL AND TRY_CAST([MOVE] AS DECIMAL(10, 2)) IS NOT NULL AND
TRY_CAST(QTY AS INT) IS NOT NULL THEN CAST(PRICE AS DECIMAL(10, 2)) * CAST([MOVE] AS
DECIMAL(10, 2)) / NULLIF (CAST(QTY AS INT), 0) ELSE NULL END AS Sales

FROM   Stage_Movement_v1;
```

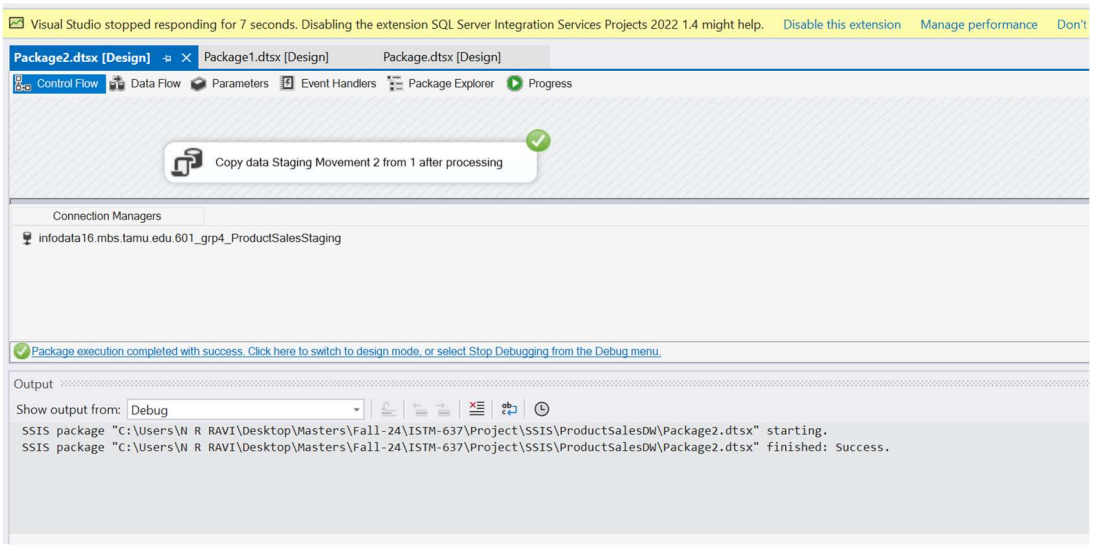


Fig 33. Execution of copying from v1 to v2 staging table

6. Profit Margin and Sales are calculated. Profit margin is calculated as profit *100 and sales is calculated as move * price / qty. If qty is 0 sales is considered as 0. ** Profit Margin is profit / 100 not profit *100, this is fixed in the end by dividing the profit margin column by 10000

```
UPDATE Stage_Movement_v2

SET

    Sales = CASE WHEN QTY = 0 THEN 0 ELSE PRICE * MOVE / QTY END
```

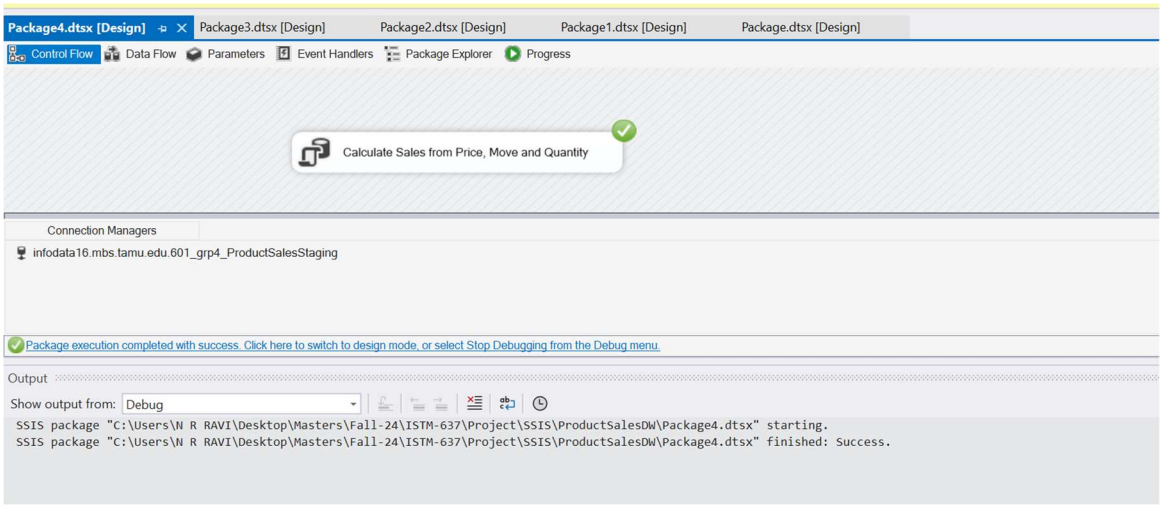


Fig 34. V2 table processed to calculate new columns

7. Remove UPC values from the Movement table which are not present in DimProduct table, UPC will be a foreign key and if there are UPC values in Fact table and not present in DimProduct table it cannot be added to DataMart

```
DELETE FROM Stage_Movement_v2

WHERE (UPC NOT IN (SELECT UPC FROM    Stage_UPC_v2))
```

8. Remove STORE values from the Movement table which are not present in DimStore table, STORE will be a foreign key and if there are STORE values in Fact table and not present in DimStore table it cannot be added to DataMart

```
DELETE FROM Stage_Movement_v2

WHERE (STORE NOT IN ('2', '4', '5', '8', '9', '12', '14', '18', '19', '21', '25', '28', '32', '33',
'39', '40', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '59', '60', '62',
'64', '65', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76', '77', '78', '80', '81', '83', '84',
'86', '88', '89', '90', '91', '92', '93', '94', '95', '97', '98', '100',
```

'101', '102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '112', '113', '114', '115', '116', '117', '118', '119', '121', '122', '123', '124', '126', '128', '129', '130', '131', '132', '133', '134', '136', '137', '139'))

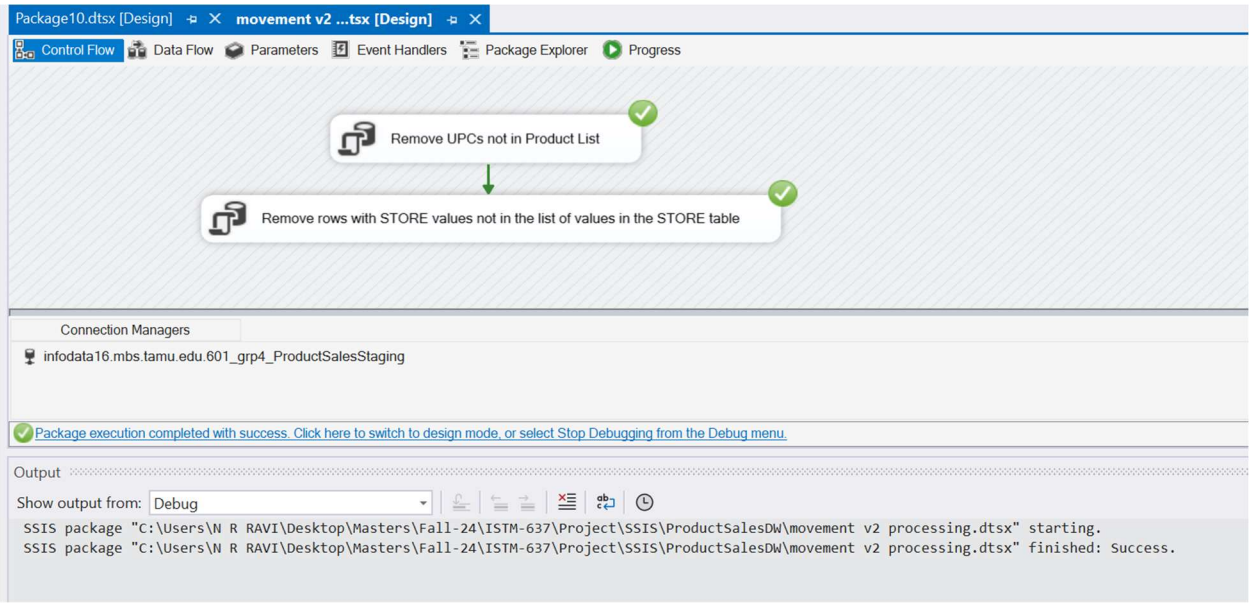
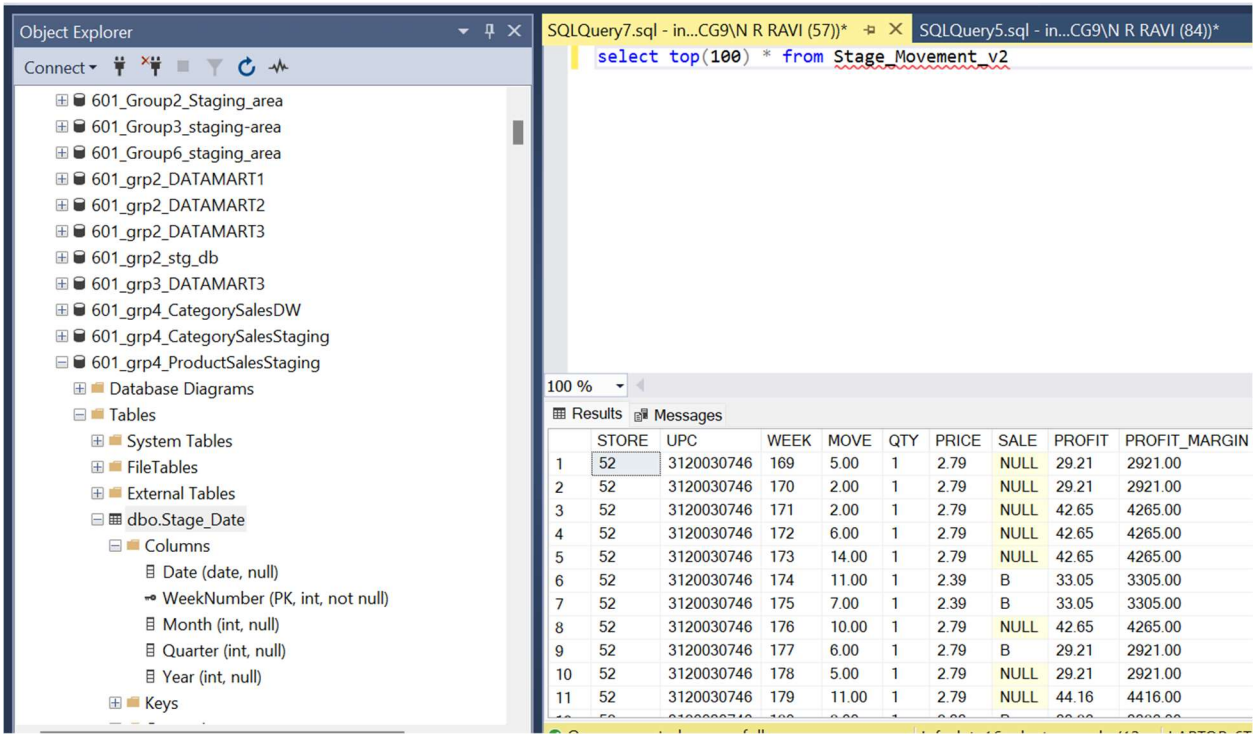


Fig 35. Task to remove UPCs and Store rows not in Dim Tables



	STORE	UPC	WEEK	MOVE	QTY	PRICE	SALE	PROFIT	PROFIT_MARGIN
1	52	3120030746	169	5.00	1	2.79	NULL	29.21	2921.00
2	52	3120030746	170	2.00	1	2.79	NULL	29.21	2921.00
3	52	3120030746	171	2.00	1	2.79	NULL	42.65	4265.00
4	52	3120030746	172	6.00	1	2.79	NULL	42.65	4265.00
5	52	3120030746	173	14.00	1	2.79	NULL	42.65	4265.00
6	52	3120030746	174	11.00	1	2.39	B	33.05	3305.00
7	52	3120030746	175	7.00	1	2.39	B	33.05	3305.00
8	52	3120030746	176	10.00	1	2.79	NULL	42.65	4265.00
9	52	3120030746	177	6.00	1	2.79	B	29.21	2921.00
10	52	3120030746	178	5.00	1	2.79	NULL	29.21	2921.00
11	52	3120030746	179	11.00	1	2.79	NULL	44.16	4416.00

Fig 36. Final version of v2 table, to be copied to Fact Table

- ## Organization of Data Staging Area :-

  Database: A separate database named [601_grp4_ProductSalesStaging].

  Staging Tables:

  - Stage_Store
  - Stage_Movement_v2
  - Stage_UPC_v2
  - Stage_Date

  Temporary Tables:

  - Stage_Movement_v1
  - Stage_UPC_v1

- ## Procedures for Data Extractions and Loadings :-

  - Run the SSIS packages to load the data into Staging tables
  - Apply transformations during the data and control flow.

- ## ETL for Dim Tables :-

  - **DimProduct :-**

    Remove Duplicate UPCs in Stage table using SQL Task

    Load Data into DimProduct by copying from Stage_UPC_v2

```
WITH CTE AS (
  SELECT
    UPC,
    DESCRIPTION,
    SIZE,
    [CASE],
    ROW_NUMBER() OVER (PARTITION BY UPC, DESCRIPTION, SIZE, [CASE] ORDER BY (SELECT NULL)) AS RowNum
  FROM
    Stage_UPC_v2
)
DELETE FROM Stage_UPC_v2
WHERE UPC IN (
  SELECT UPC
  FROM CTE
```

```
        WHERE RowNum > 1
    );
```

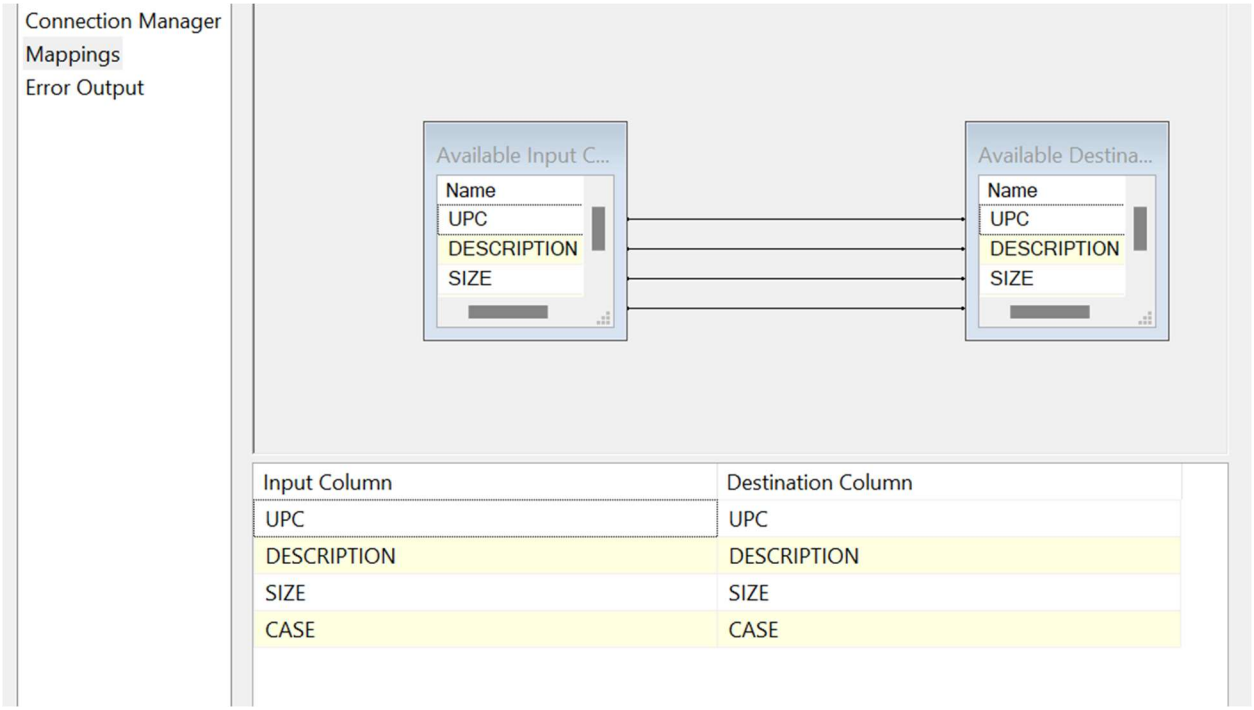


Fig 37. Mapping from Stage UPC to DimProduct

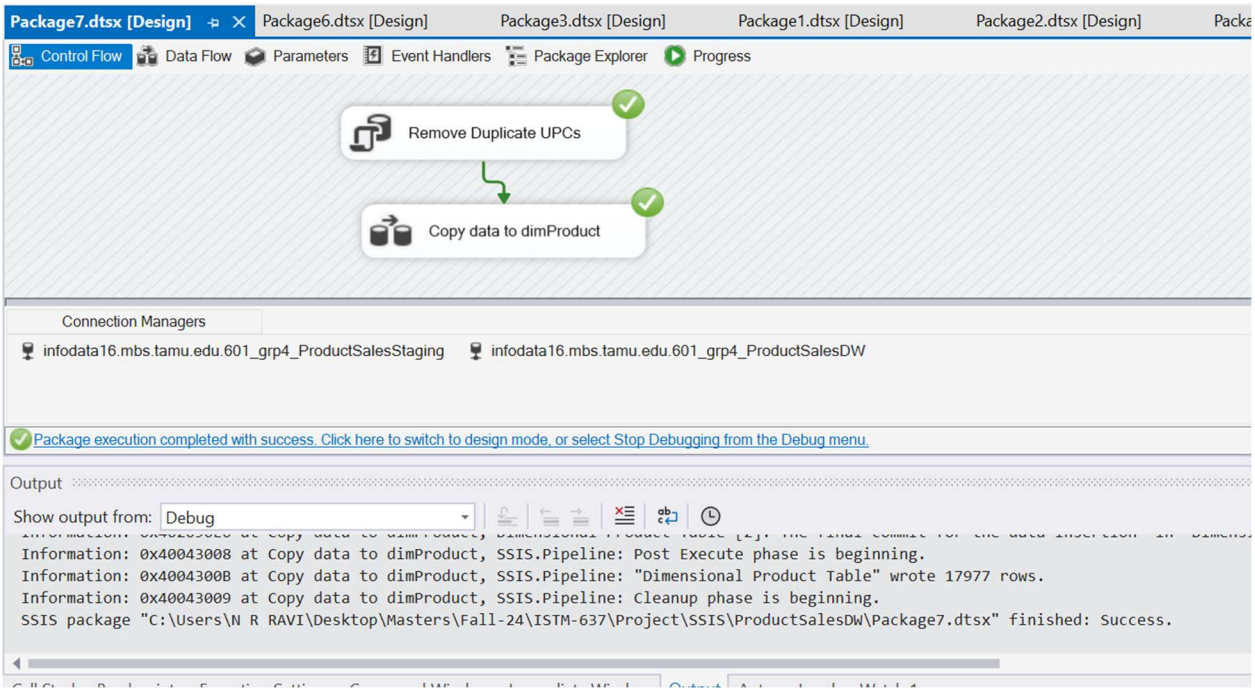


Fig 38. Execution of Product Load

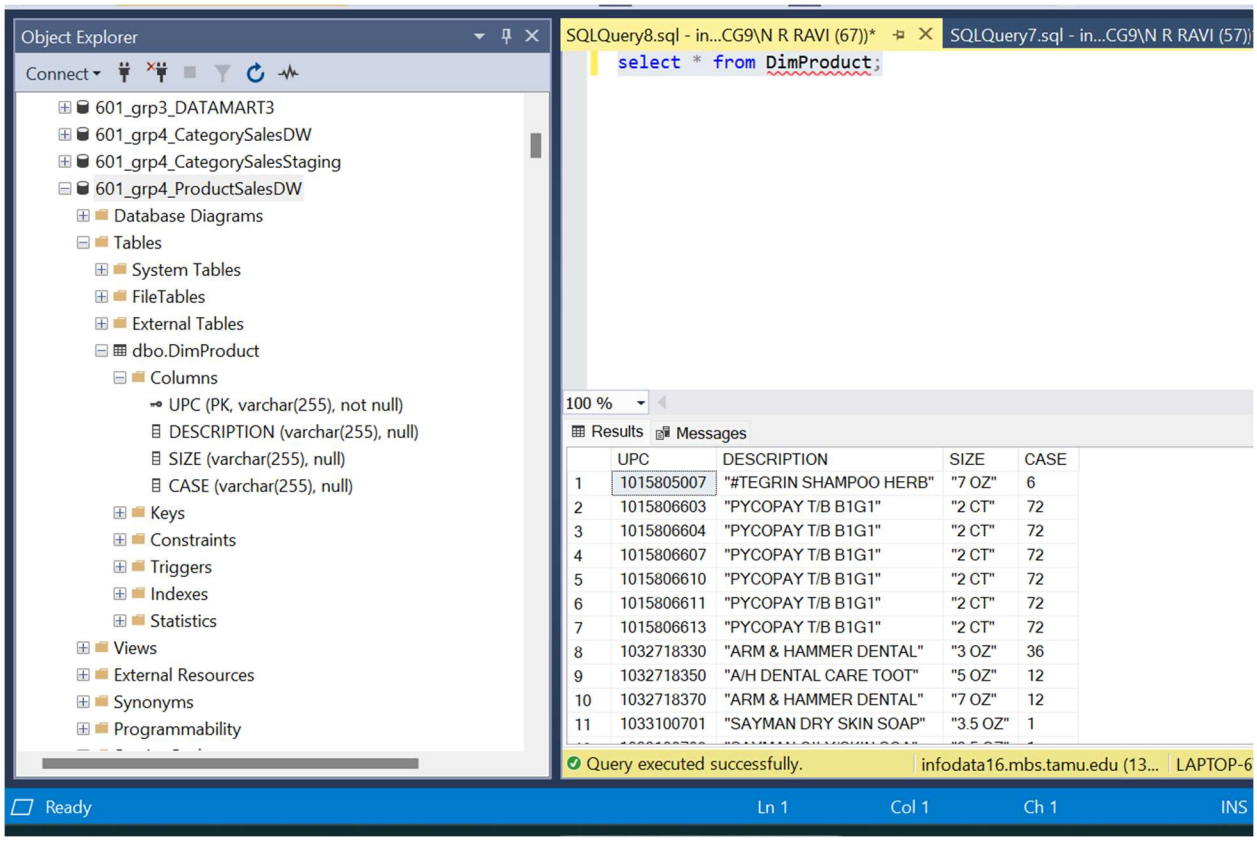


Fig 39. Product data populated

- **DimStore :-**

Load data into DimStore by loading from Stage_Store

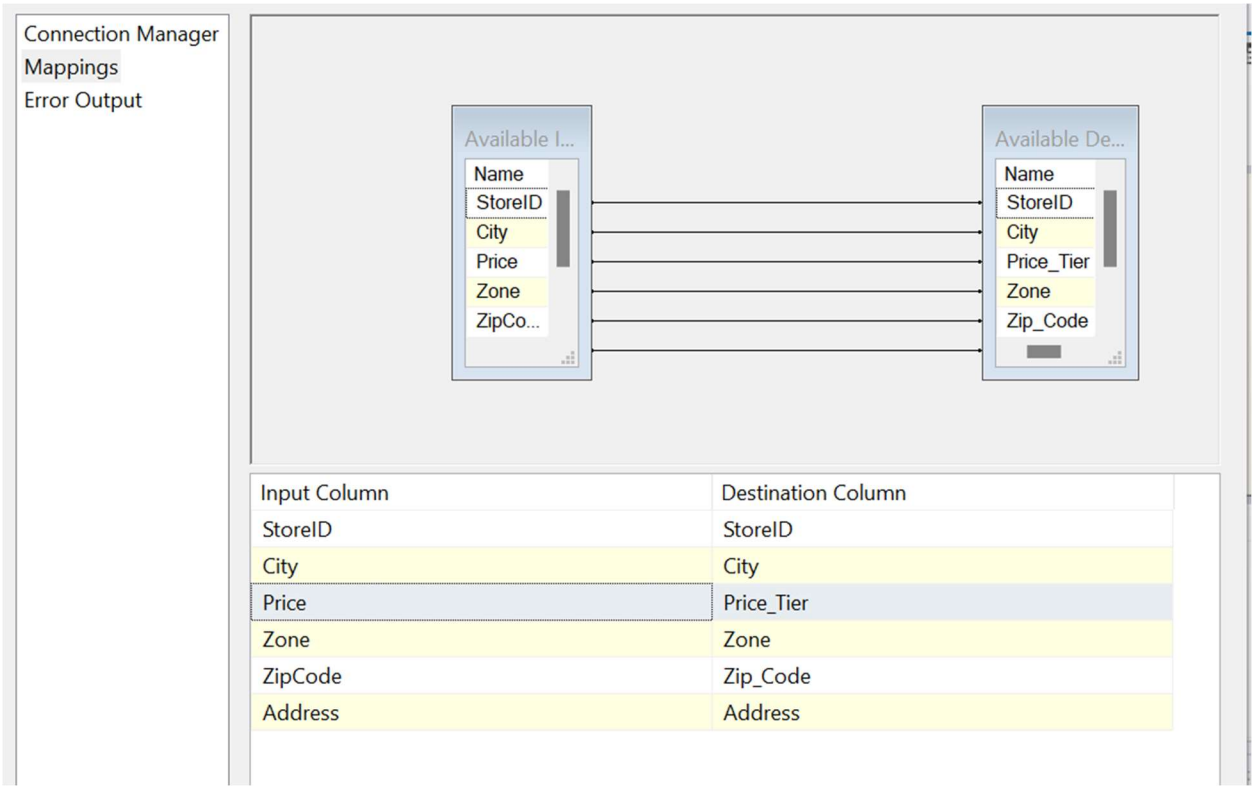


Fig 40. Store data mapping

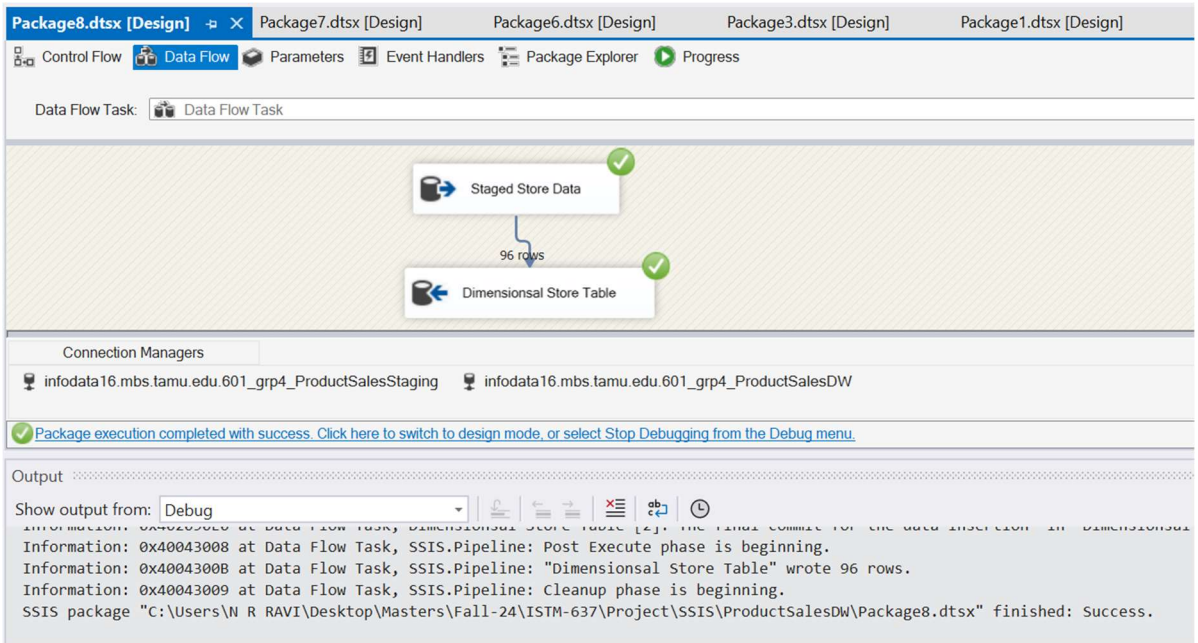
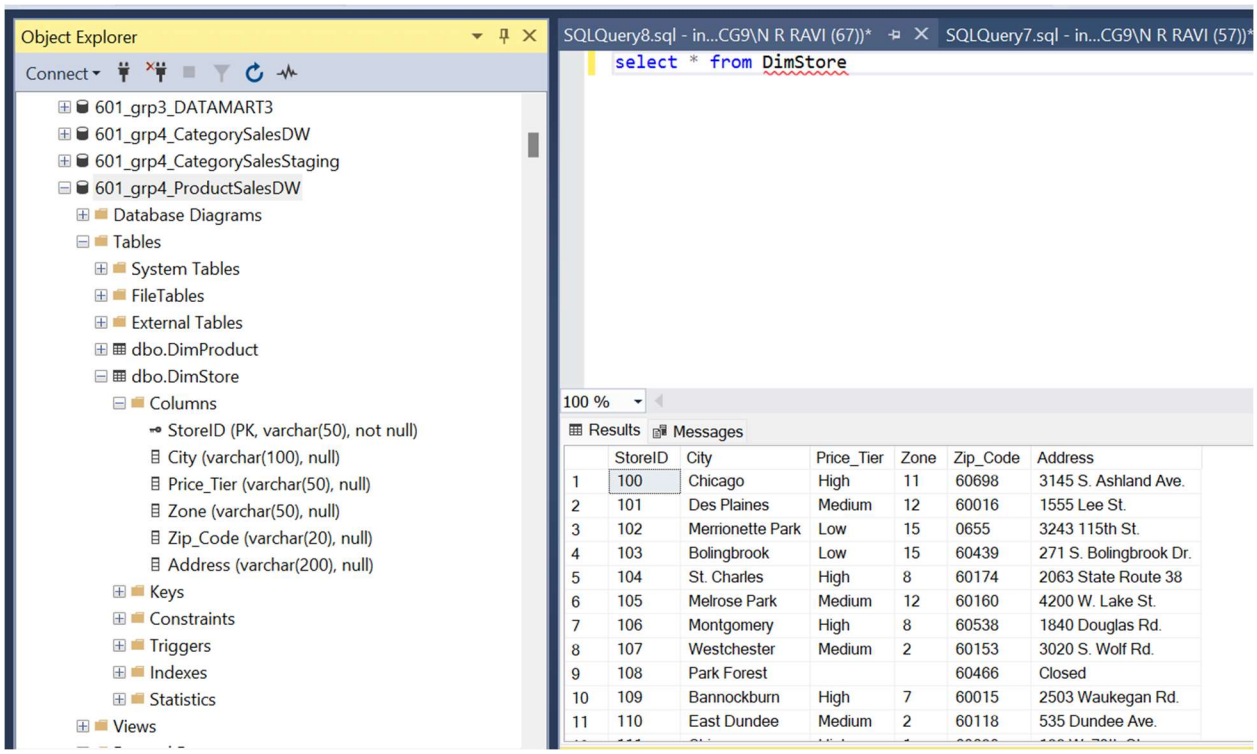


Fig 41. Execution of loading to DimStore table



	StoreID	City	Price_Tier	Zone	Zip_Code	Address
1	100	Chicago	High	11	60698	3145 S. Ashland Ave.
2	101	Des Plaines	Medium	12	60016	1555 Lee St.
3	102	Merrionette Park	Low	15	0655	3243 115th St.
4	103	Bolingbrook	Low	15	60439	271 S. Bolingbrook Dr.
5	104	St. Charles	High	8	60174	2063 State Route 38
6	105	Melrose Park	Medium	12	60160	4200 W. Lake St.
7	106	Montgomery	High	8	60538	1840 Douglas Rd.
8	107	Westchester	Medium	2	60153	3020 S. Wolf Rd.
9	108	Park Forest			60466	Closed
10	109	Bannockburn	High	7	60015	2503 Waukegan Rd.
11	110	East Dundee	Medium	2	60118	535 Dundee Ave.

Fig 42. Store Table Populated

- **DimDate :-**

Load data into DimDate by loading from Stage_Date

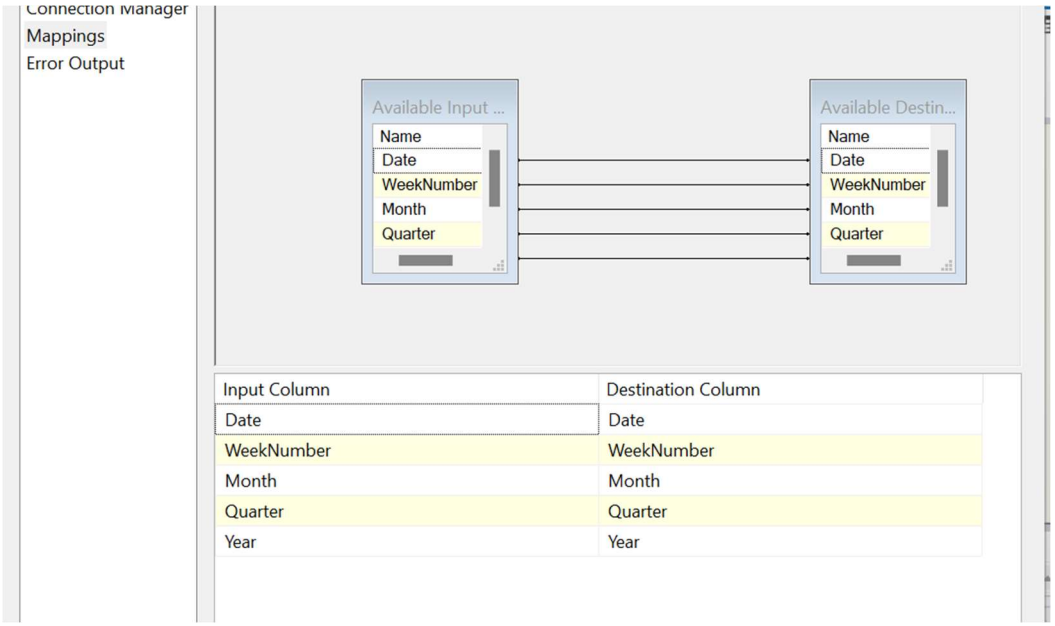


Fig 43. Mapping Date Data

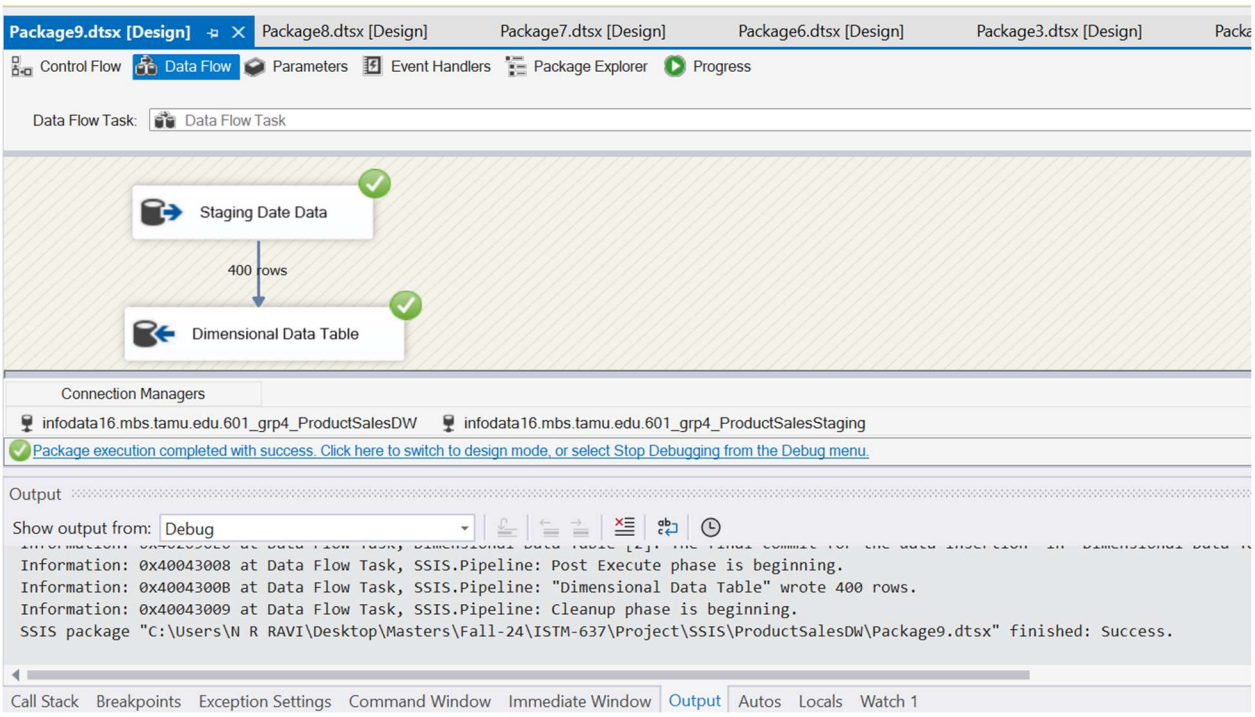


Fig 44. Execution of Date Load

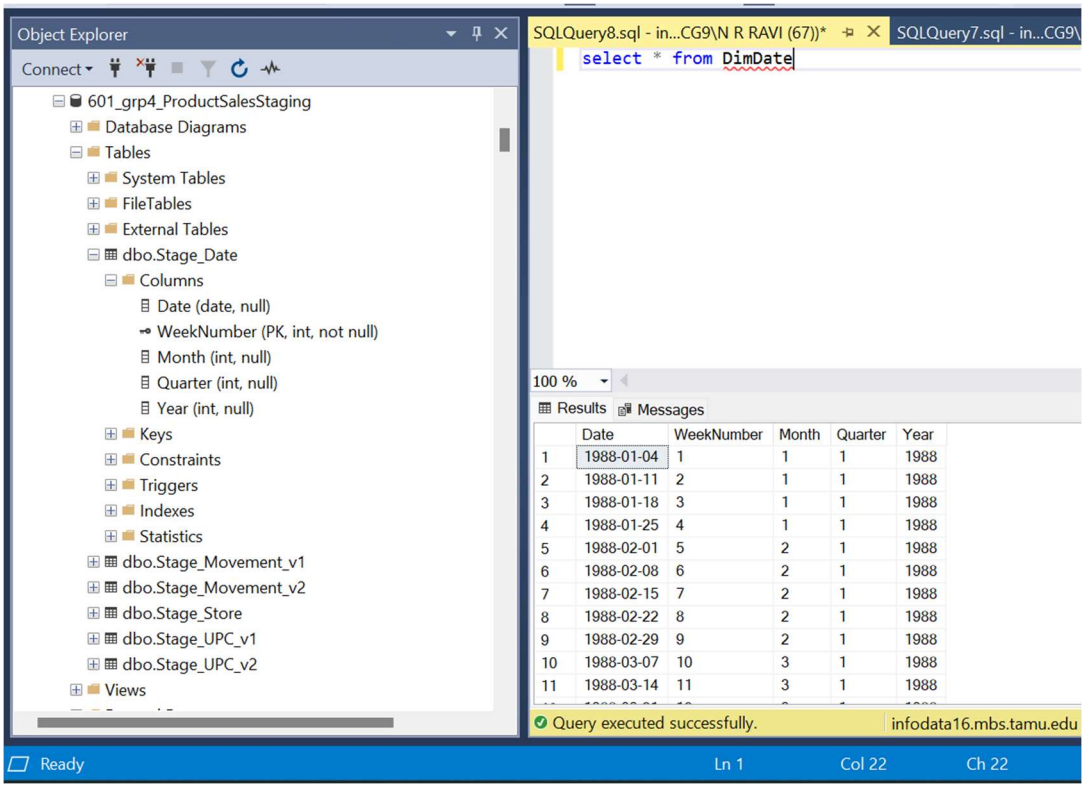


Fig 45. Date data populated

- **ETL for Fact Tables :-**
    - **FactSales :-**

Load data into FactSales by loading from Stage_Movement_v2

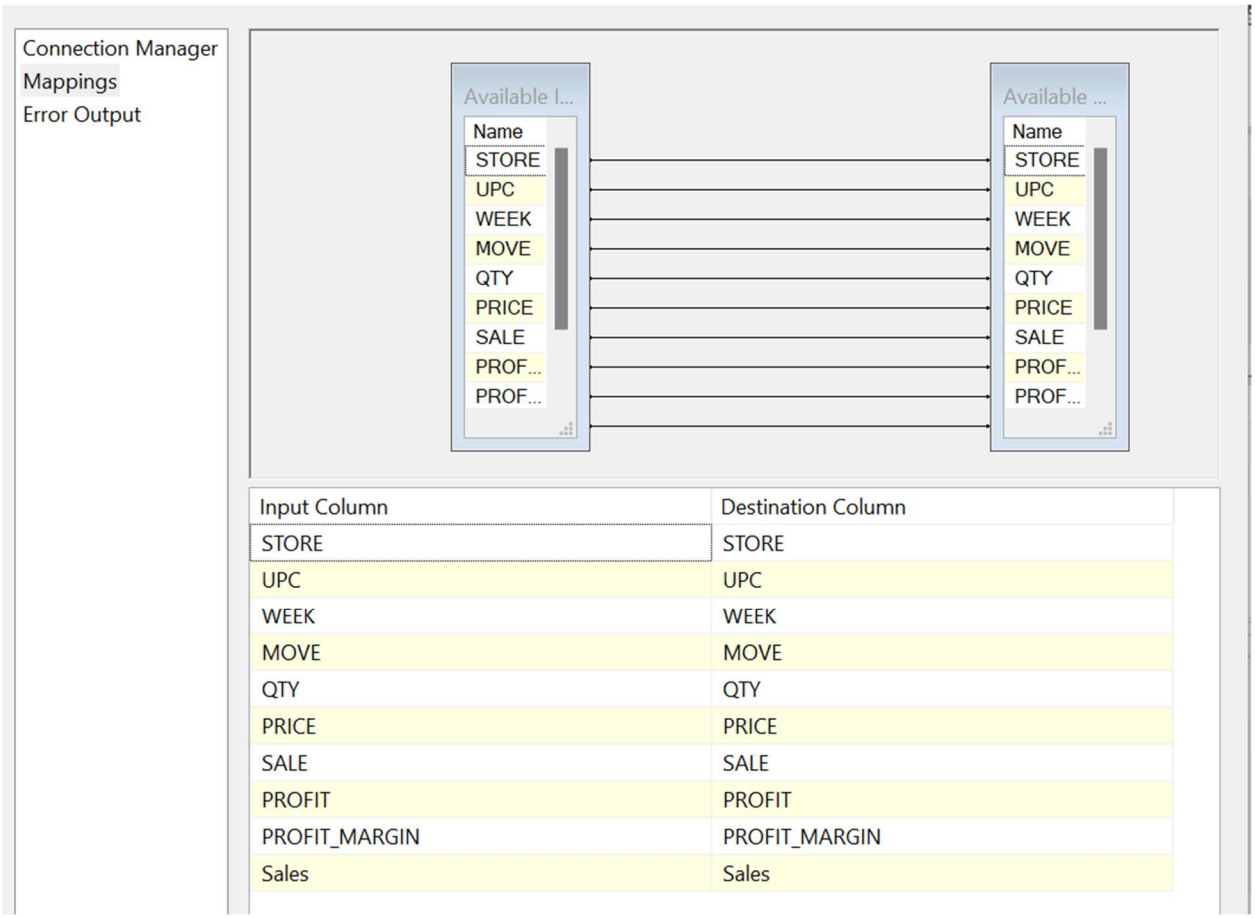


Fig 46. Stage Movement to FactSales Mapping

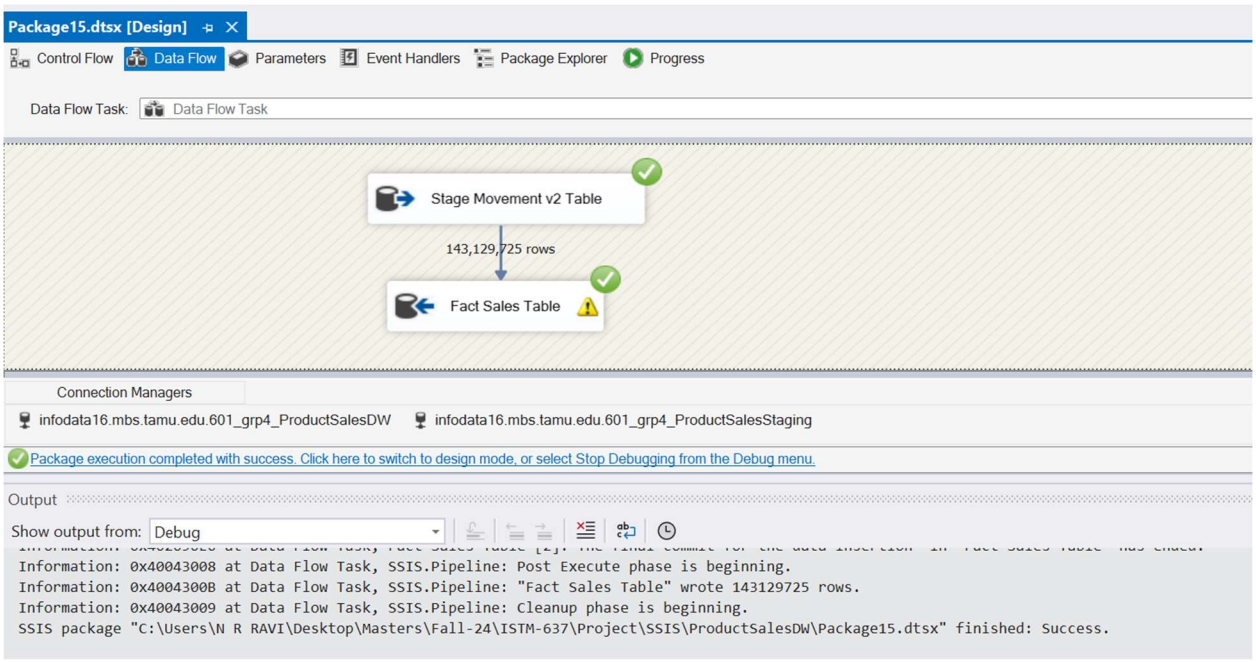


Fig 47. Execution of Data Copy task from Staging to Fact Table

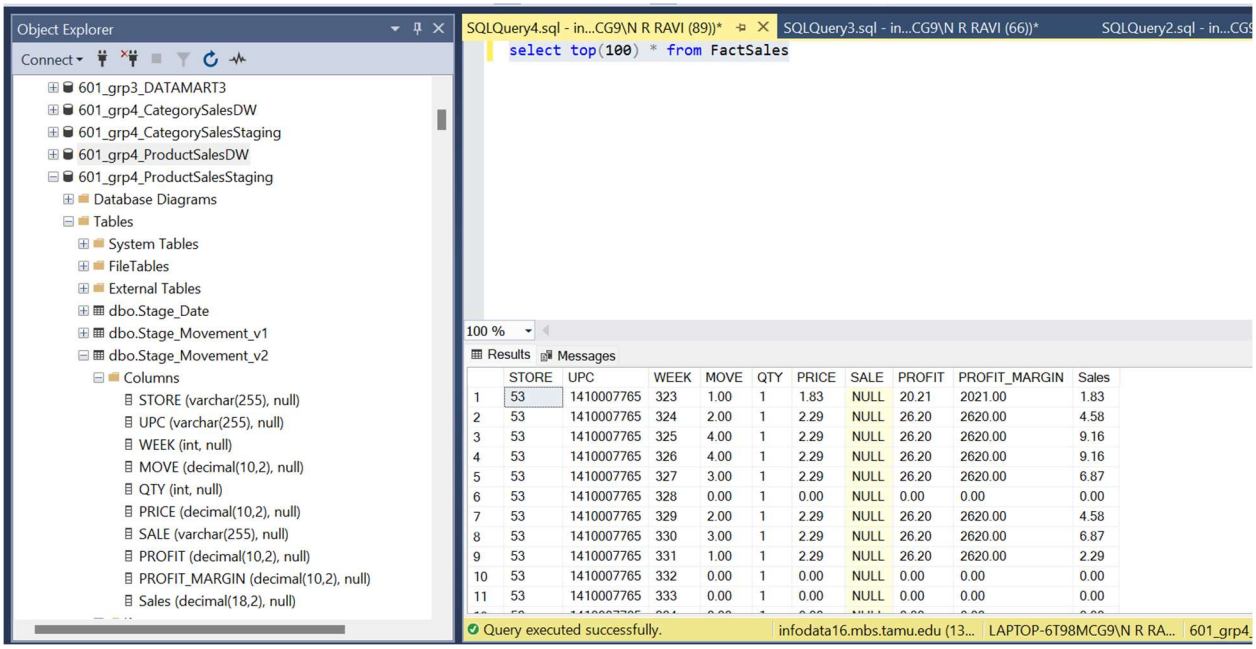


Fig 48. FactSales Table Populated

Fix Profit Margin:-

```
UPDATE FactSales

SET

        PROFIT_MARGIN = PROFIT_MARGIN / 10000
```

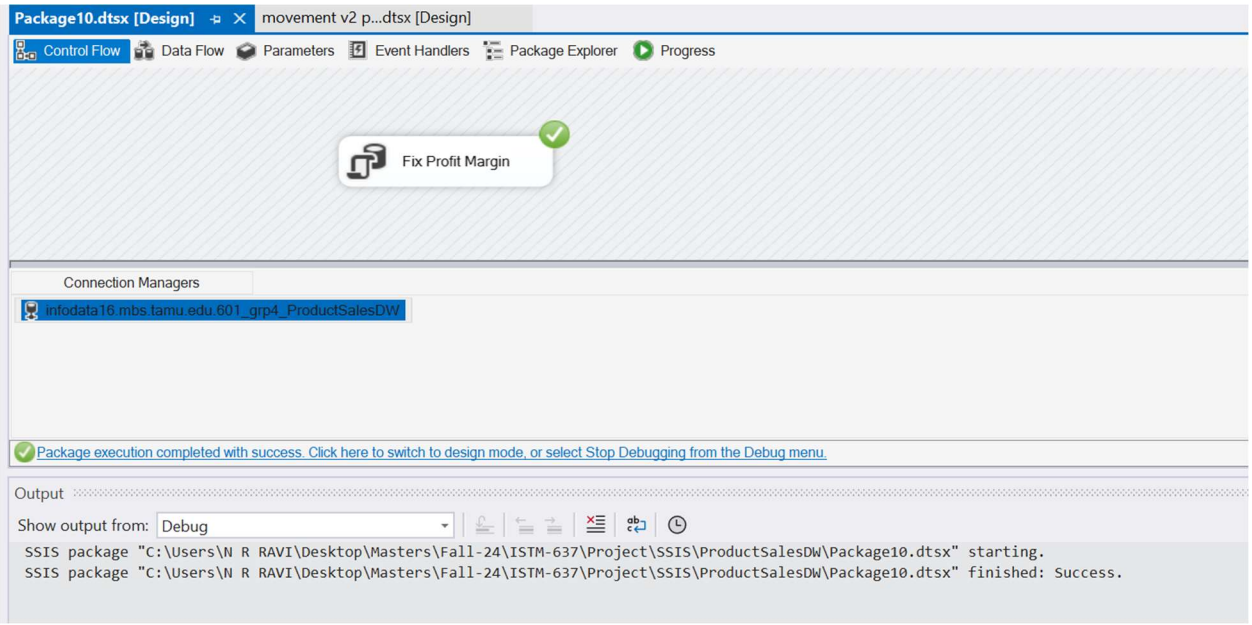


Fig 49. SQL Task to fix profit Margin

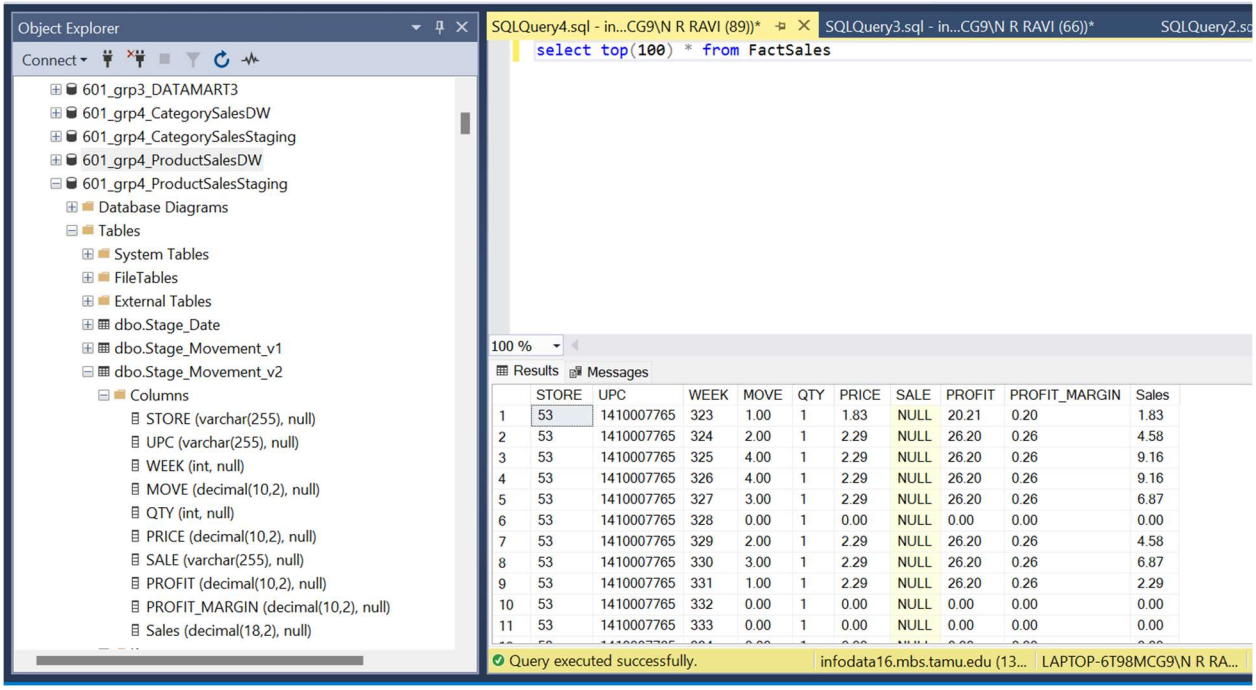


Fig 50. Final version of FactSales

Drop Temporary Tables

- **For Category Sales Staging DB**
  USE DATABASE 601_grp4_CategorySalesStaging;
  DROP TABLE Stage_Movement_v1;
  DROP TABLE Stage_UPC_v1;
- **For Product Sales Staging DB**
  USE DATABASE 601_grp4_ProductSalesStaging;
  DROP TABLE Stage_Sales_v1;
  DROP TABLE Stage_Sales_v2;

Report - 5

➢ Target reports that satisfy the business questions

❖ BQ1: Create a report using only the SSAS cube

Target Report:

- A report analyzing aggregated sales and customer data using SSAS cube's multidimensional capabilities.

Why SSAS Cube Works:

- **Efficient Aggregation**: SSAS cubes are designed to handle multidimensional data analysis, making it easy to query aggregated metrics like sales and customer counts.
- **Interactive Analysis**: SSAS provides drill-down and slicing/dicing capabilities, enabling dynamic exploration of data hierarchies.

❖ BQ2: Create a report using SSRS only

Target Report:

- A tabular report summarizing store-level sales and customer traffic using SSRS.

Why SSRS Works:

- **Detailed Layouts**: SSRS excels in creating paginated, pixel-perfect reports ideal for operational reporting.
- **Direct Database Connectivity**: SSRS can query independent data marts directly, allowing detailed data retrieval without relying on cubes.

❖ BQ3: Create a report using Power BI

Target Report:

- A visualization of customer traffic patterns and sales trends by day of the week, created in Power BI.

Why Power BI Works:

- **Rich Visualization**: These tools are ideal for creating interactive dashboards with charts, graphs, and heatmaps, which provide better insights into data patterns.
- **User-Friendly Interface**: Intuitive drag-and-drop interfaces enable quick report development and customization.

❖ BQ4: Create a report using Power BI with SSAS cube

Target Report:

- A comparative analysis of holiday vs. non-holiday sales using Power BI with SSAS cube integration.

Why Power BI with SSAS Cube Works:

- **Enhanced Analytical Power**: Combining SSAS cubes with advanced visualization/reporting tools like Power BI or Tableau allows for in-depth exploration of data.
- **Power BI with SSAS**: Perfect for creating interactive and intuitive dashboards to communicate insights effectively.

❖ **BQ5: Create a report using Power BI**

Target Report:

- A flexible choice depending on the business context. Using Power BI for visualizing promotion effectiveness, generating a precise operational report.

Why Power BI Works:

- This question allows experimentation to use the best-suited tool for the dataset and business question, Power BI is a modern tool with many interactive visualization options.

➢ Business Question - 1

▪ Mappings from Independent Data Marts to Report Attributes

Independent Data Mart	Attribute	Report Attribute	Mapping Logic
FactSales	Total Sales	Average Sales Per Person	Computed by dividing Total Sales by Customer Count.
DimStore	StoreID	StoreID	Identifies individual stores for traffic and sales analysis.
DimDate	Year	Year	Used to filter or group data by specific years.

▪ Description of Report Templates

Graph Type

- Type: Tabular Report (SSAS Cube Browser)
- Purpose:
    - The report displays Average Sales Per Person across stores and time periods, offering insights into customer traffic and its relationship to store revenue.

Columns and Filters

- **Columns**:
    - **StoreID**: Lists each store analyzed in the report.
    - **Average Sales Per Person**: Displays the computed metric for each store or time period.
- **Filters**:
    - **Year**: Filters data by specific years (e.g., 1990 in Image 2).
    - **StoreID**: Filters data by a specific store (e.g., Store 100 in Image 3).

- **Building the Report – PowerBI**

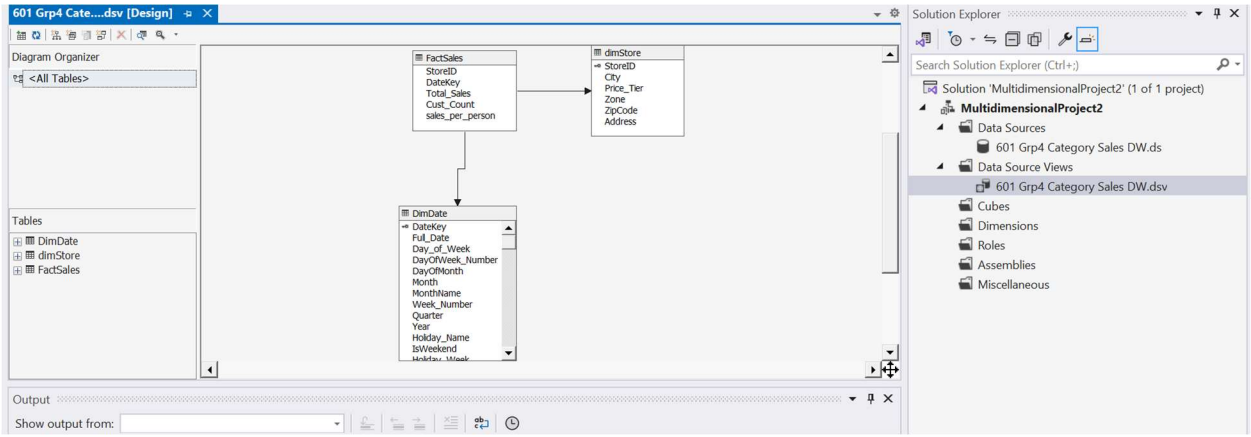


Fig 51. Data Source view from SSAS cube

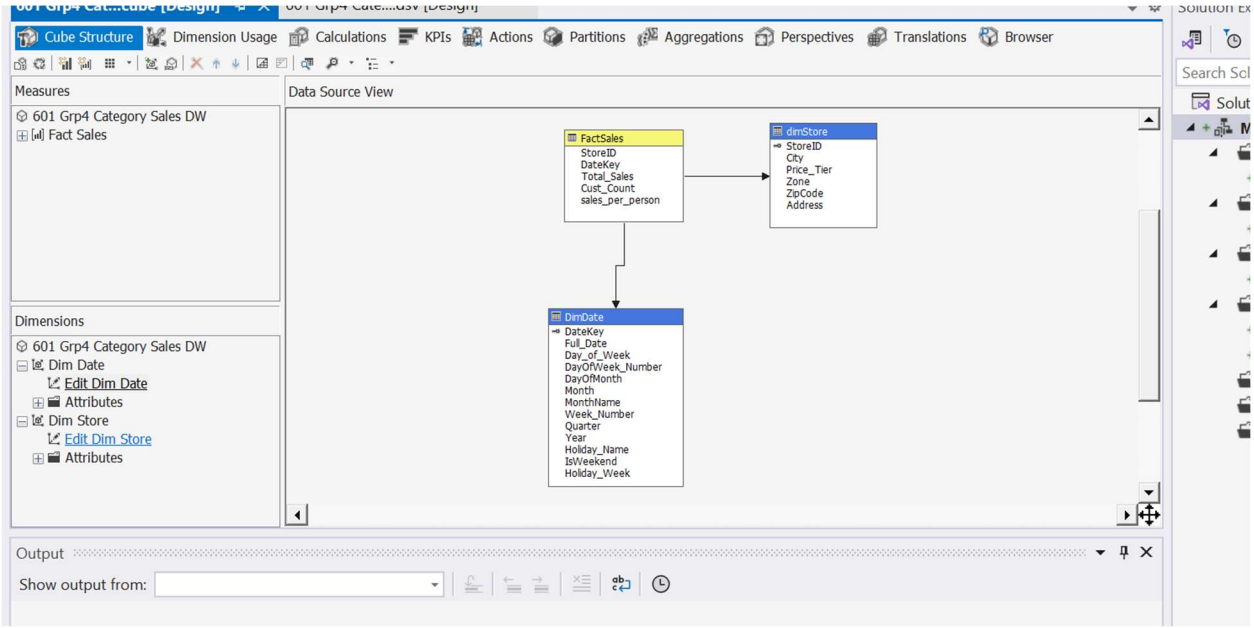


Fig 52. Structure of the SSAS Cube

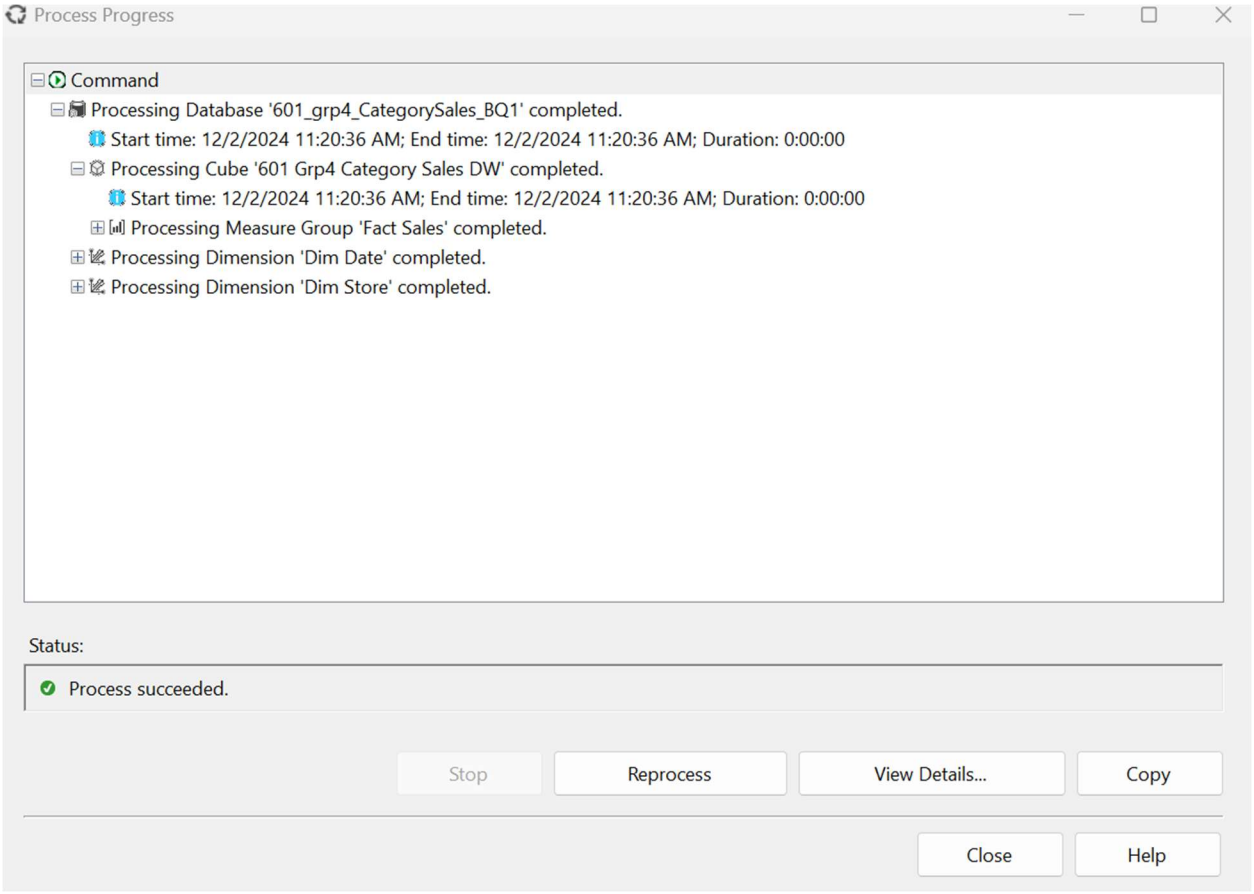


Fig 53. Successful deployment of SSAS cube

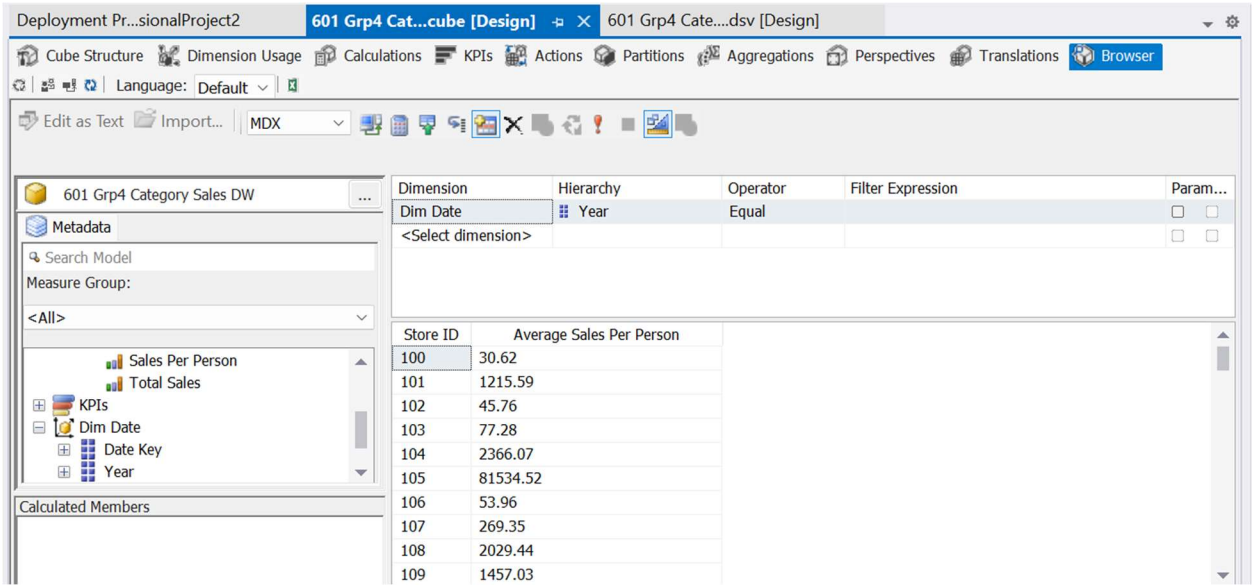


Fig 54. Average Sales Per Person For all the stores for all the years

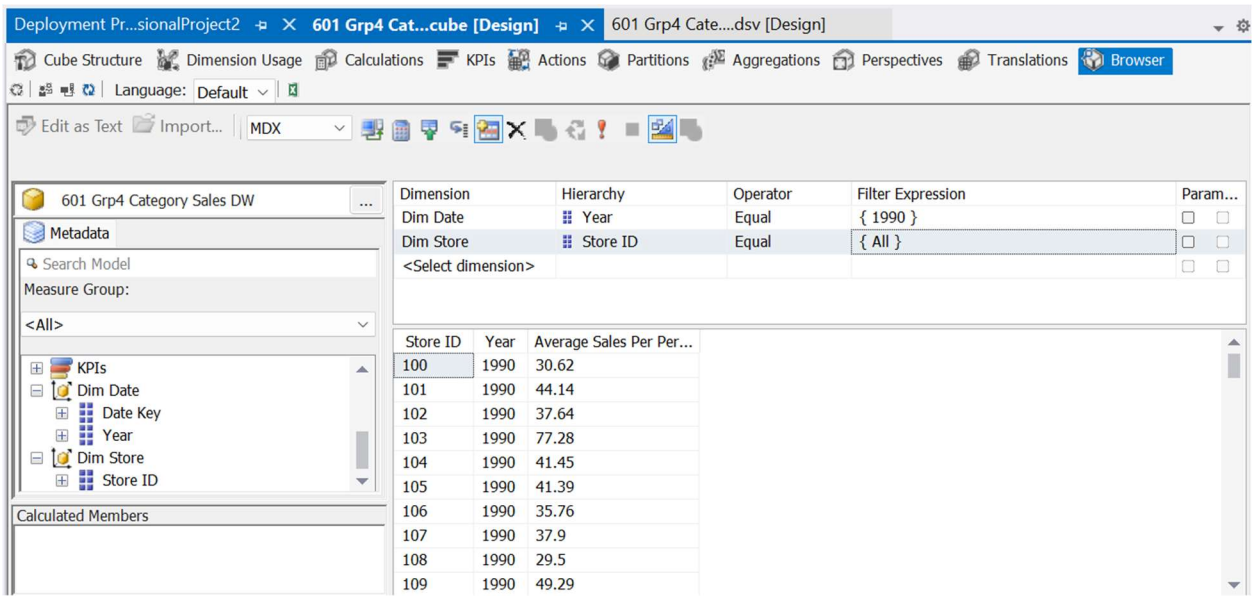


Fig 55. Average Sales Per Person For all the stores for year 1990

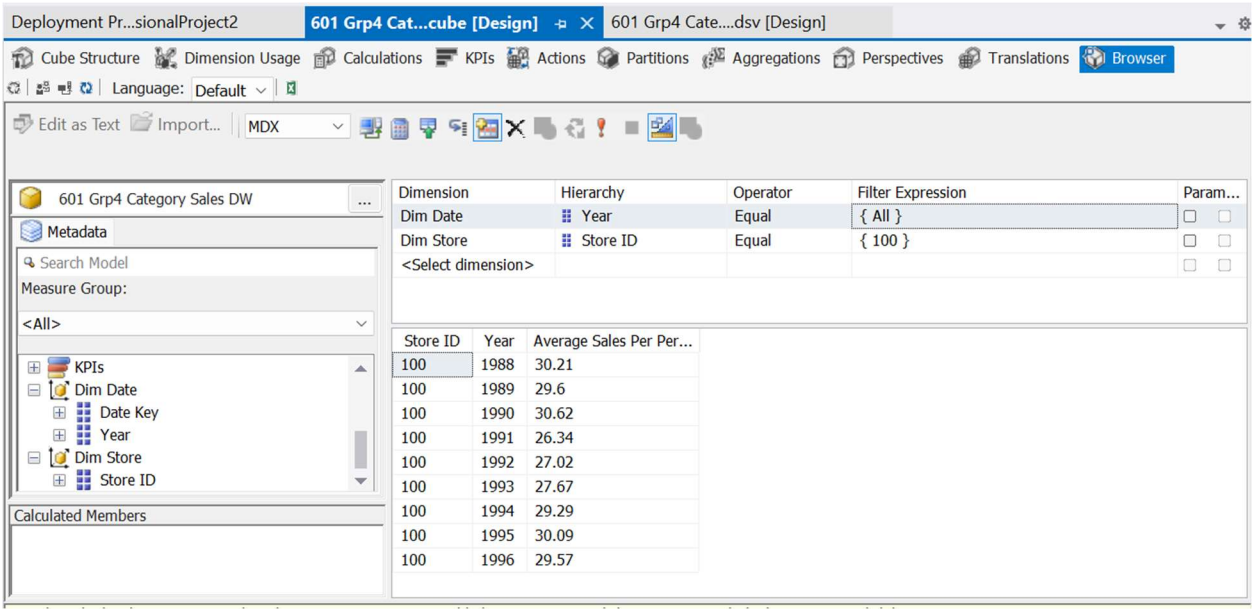


Fig 56. Average Sales Per Person For store 100 for all the years

- ## Results and Analysis

1. **Customer Traffic vs. Sales Correlation**:
    - **High Traffic, Low Sales**:
        - Stores with high customer traffic but lower average sales per person might indicate inefficiencies, such as poor product assortment or ineffective promotions.

        - Example: Store 101 in Image 1 has high customer traffic but relatively low sales, signaling potential improvement areas.
    - **High Traffic, High Sales**:
        - Stores with both high traffic and high sales indicate efficient conversion of customers into buyers.
        - Example: Store 105 in Image 1 demonstrates strong performance.
2. **Year-Specific Performance**:
    - Image 2 focuses on the year 1990, allowing analysis of historical trends or specific economic impacts that influenced customer behavior during that time.
3. **Store-Specific Insights**:
    - Image 3 provides granular analysis for **Store 100** across all years, highlighting its performance trends and consistency over time.

Operational and Strategic Implications

1. **Optimization Opportunities**:
    - Stores with low **Average Sales Per Person** should focus on improving product mix, promotions, and customer engagement strategies.
    - Example: Stores like 101 and 106 in Image 1 may need targeted interventions to increase conversion rates.
2. **Resource Allocation**:
    - High-performing stores with strong sales per person justify increased inventory and staffing during peak periods.
    - Example: Store 105 demonstrates sustained performance, indicating an opportunity for scaling resources.
3. **Promotion Effectiveness**:
    - Cross-referencing customer traffic and sales during promotional periods can identify whether campaigns effectively drive revenue.
4. **Strategic Planning**:
    - Insights from year-on-year and store-specific trends help refine marketing strategies, adjust inventory, and optimize store layouts for better performance.

Conclusion

This SSAS cube browser report provides actionable insights into customer traffic and sales correlation, allowing Dominick's Finer Foods to optimize store operations, target low-performing stores, and enhance

customer engagement strategies. The metrics serve as a foundation for driving profitability and operational efficiency across all stores.

➢ Business Question – 2

▪ Mappings from Independent Data Marts to Report Attributes

Independent Data Mart	Attribute	Report Attribute	Mapping Logic
FactSales	Profit Margin	Profit Margin	Calculated from Profit
DimProduct	Product_Description	Description	Lists product descriptions for sales and margin analysis.
DimStore	StoreID	Store Filters	Filters used to focus on specific stores.

▪ Description of Report Templates

Graph Type

- **Type**: Tabular Report (SSRS)
- **Purpose**:
    - The report compares the **Profit Margins** of different products for **specific stores** (Store 2 and Store 4).
    - It highlights high-margin products, low-margin products, and consistency across product categories.

Columns and Filters

- **Columns**:
    - **Description**: Lists the products under analysis.
    - **Profit Margin**: Displays the calculated profit margin for each product.
- **Filters**:
    - **StoreID**: Focuses the report on specific stores (e.g., Store 2 in the provided images).
    - **Time Period**: The analysis can be extended or refined by filtering to specific time periods to capture seasonal or periodic trends.

- **Building the Report – SSRS**

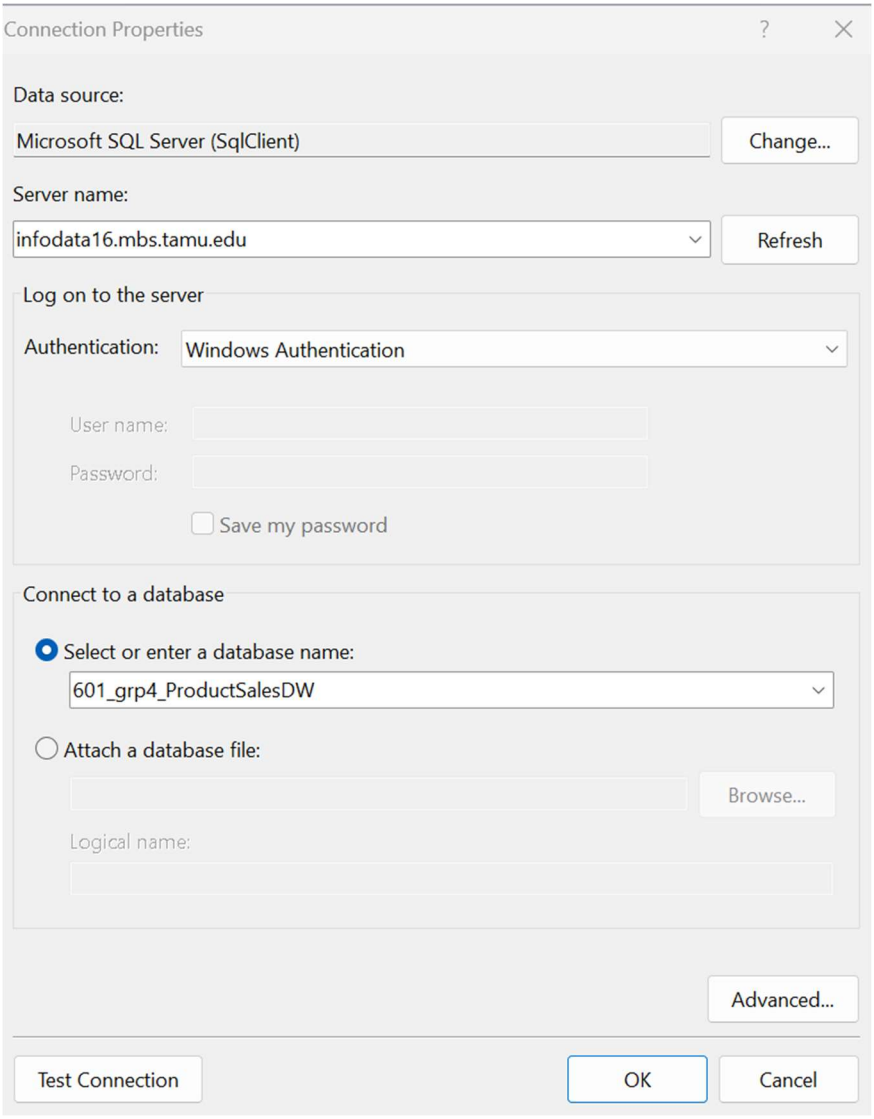


Fig 57. Connecting to Data Mart from SSRS

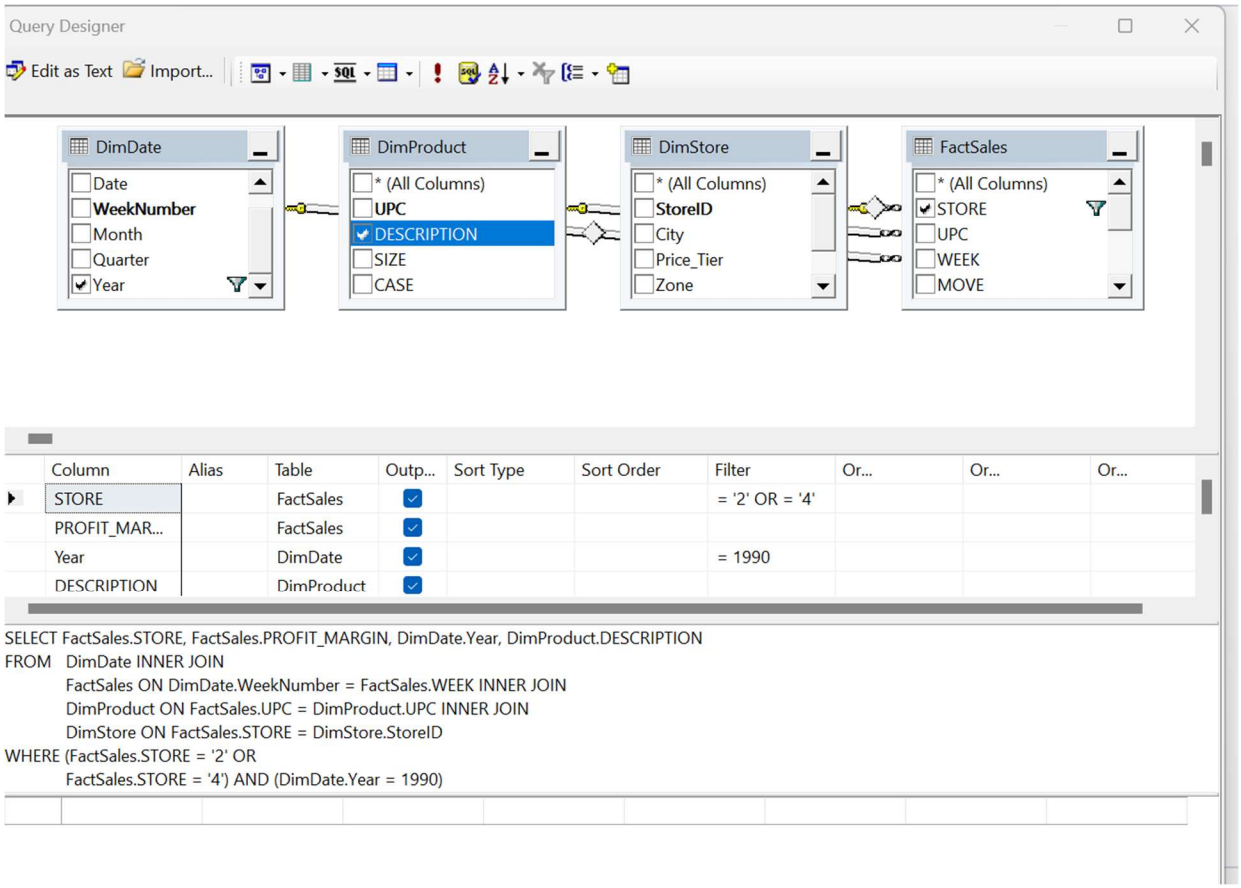


Fig 58. SSRS Import Query

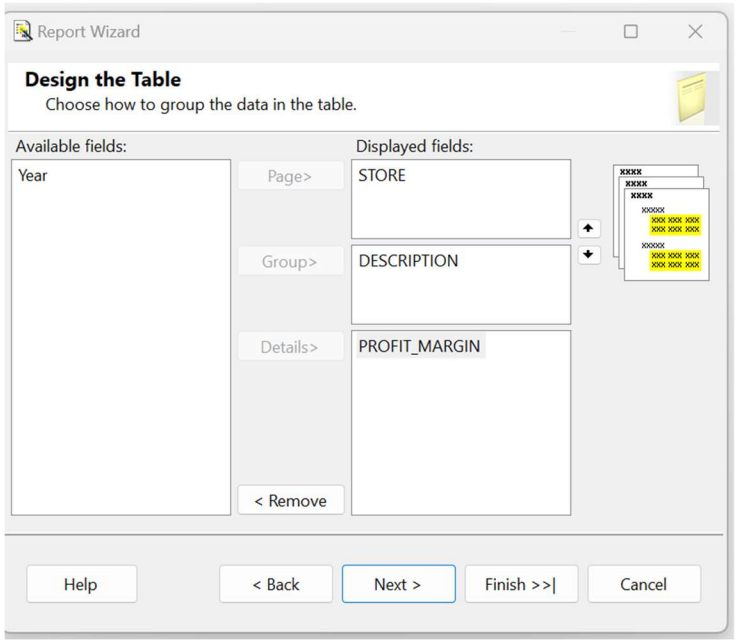


Fig 59. Report Mapping for Imported Columns

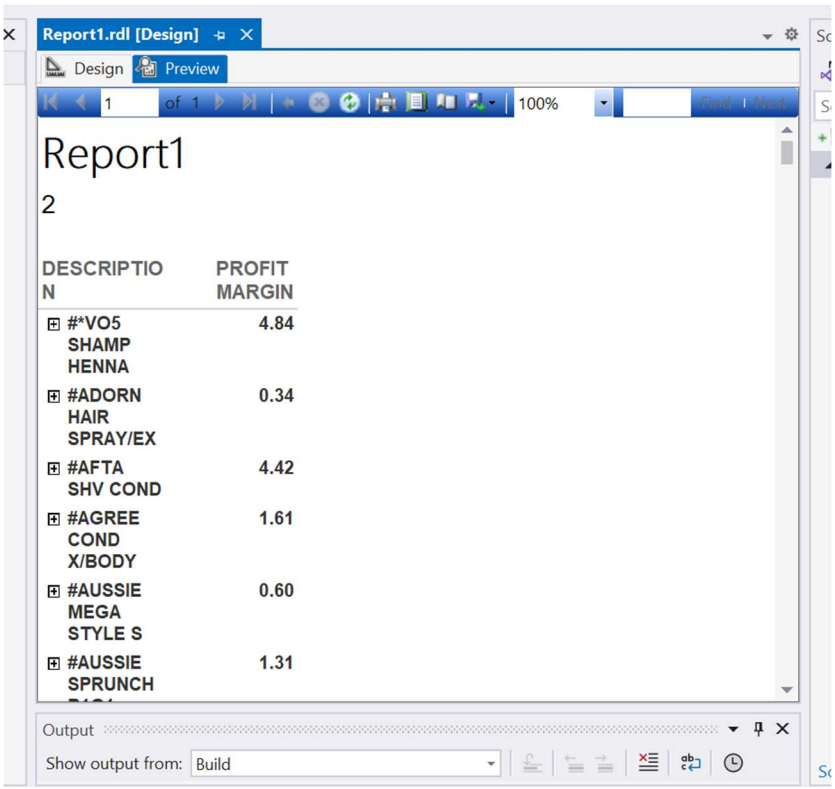


Fig 60. SSRS Report Preview

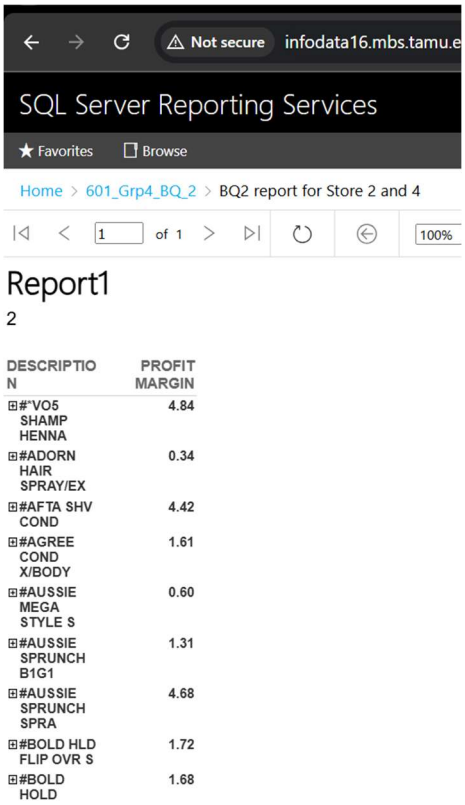


Fig 61. SSRS Deployed Report

- **Results and Analysis**

Link -
http://infodata16.mbs.tamu.edu/reports/report/601_Grp4_BQ_2/BQ2%20report%20for%20Store%202%20and%204

1. **High-Margin Products**:
    - Products with high profit margins, such as **Personal Touch Disposable Razors (11.05)** and **Gen Mills Triples (10.01)**, demonstrate significant profitability and should be prioritized for marketing campaigns, shelf space, and inventory availability.
2. **Low-Margin Products**:
    - Items with low or negligible profit margins, such as **Studio Gelling Curl (0.00)** and **Ultra Swim Moist Shampoo (0.00)**, may need reevaluation.
    - Potential actions include:
        - Renegotiating supplier costs.
        - Adjusting pricing strategies.
        - Discontinuing underperforming items.
3. **Product Categories**:
    - Categories with consistent high margins across multiple items (e.g., shampoos like **VO5 Hair Dressing (8.19)** and **Schick Razors (9.75)**) suggest reliable profit drivers.
    - Conversely, low-margin categories may indicate the need for strategic improvements.
4. **Store-Specific Trends**:
    - By focusing on Store 2, this report identifies the unique profitability patterns and customer preferences specific to this location.
    - Comparing results with other stores (e.g., Store 4) can provide insights into regional differences and tailored strategies.

Operational and Strategic Implications

1. **Marketing Focus**:
    - High-margin products should be highlighted in promotional campaigns to maximize profitability.
    - Bundling or upselling these items can enhance sales volume while maintaining high profit margins.
2. **Inventory Management**:

- Prioritizing high-margin items in stock planning ensures optimal use of shelf space and avoids overstocking low-margin products.

3. **Profitability Optimization**:
    - For low-margin products, explore opportunities to reduce costs, increase sales volume, or adjust pricing strategies.
4. **Cross-Store Comparisons**:
    - Analyzing profit margins across different stores and time periods helps identify consistent performers and outliers, enabling better resource allocation and strategy development.

Conclusion

This report provides actionable insights into product profitability, helping Dominick's Finer Foods to enhance revenue by focusing on high-margin items while addressing underperforming products. The analysis supports data-driven decisions for marketing, inventory, and strategic planning, ensuring long-term profitability and operational efficiency.

** The Fact Table had 140M+ rows and it was not loading into SSRS so only Data from Store 2 and 4 for year 1990 was imported to the SSRS*

➢ Business Question - 3

▪ Mappings from Independent Data Marts to Report Attributes

Independent Data Mart	Attribute	Report Attribute	Mapping Logic
Category Sales DataMart	Total_Sales	Total_Sales	Aggregated sales amount per day.
DimDate	Day_of_Week	Day_of_Week	Extracted from the date attribute.
FactSales	StoreID	Store Filters	Used for filtering sales by specific stores.
DimDate	Year	Year Filters	Used for filtering sales by specific years.

▪ Description of Report Templates

Graph Type: The reports use **Stacked Bar Charts** to visualize sales data, making it easy to compare total sales across categories (days of the week) in a clear and concise format.

X-Axis: The **Sum of Total Sales** (numeric) represents the aggregated sales values for each day of the week, providing quantitative insights into sales performance.

Y-Axis: The **Day_of_Week** (categorical) is displayed as labels for the bar chart, showing the sales distribution across days like Monday, Tuesday, and so on.

Filters:

- **Store ID**: Filters are applied to focus on specific stores (e.g., Store ID 2 or Store ID 4) to identify their unique sales trends.
- **Year**: Filters are used to analyze data for a specific year (e.g., 1989) to identify temporal patterns and trends.
- **Default View**: Some reports display aggregated sales data for all years and all stores, serving as a baseline for comparison.

Purpose:

- The visualization provides insights into sales trends across days of the week, with the ability to drill down into specific stores or years for more detailed analysis.
- The filters enhance the flexibility of the report, allowing users to focus on subsets of data for actionable insights.

This combined description captures the essence of the report templates, highlighting their structure, attributes, and functionality.

▪ Building the Report – PowerBI

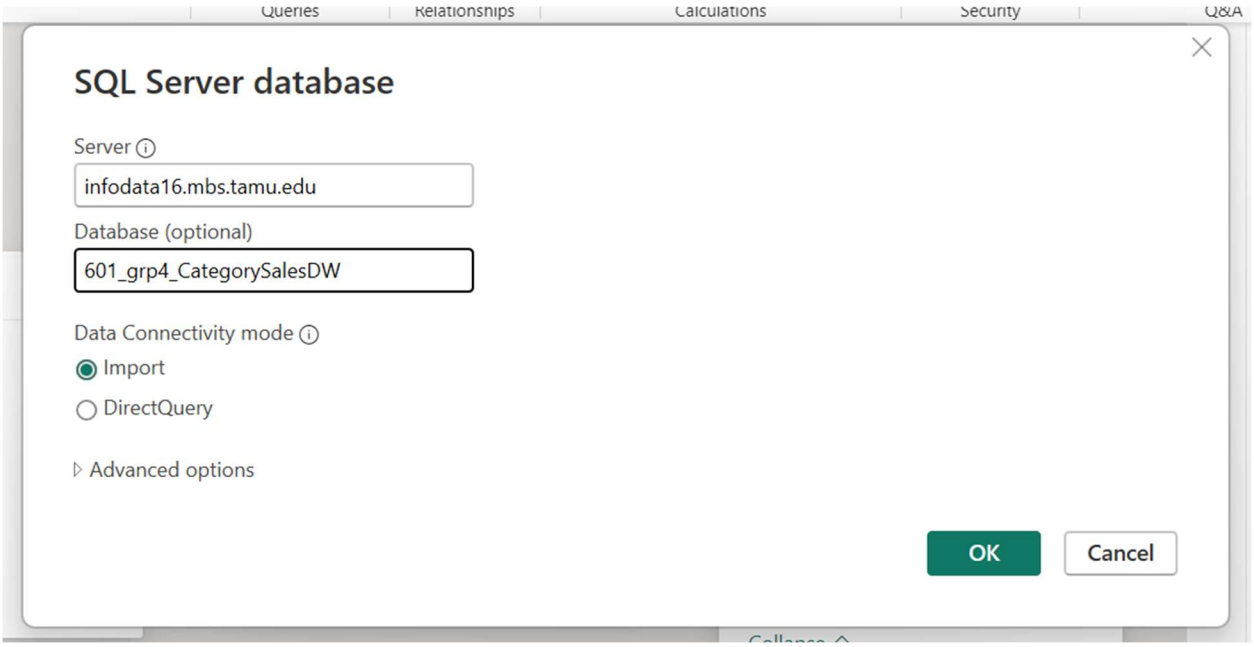


Fig 62. Selecting the server and Database in PBI

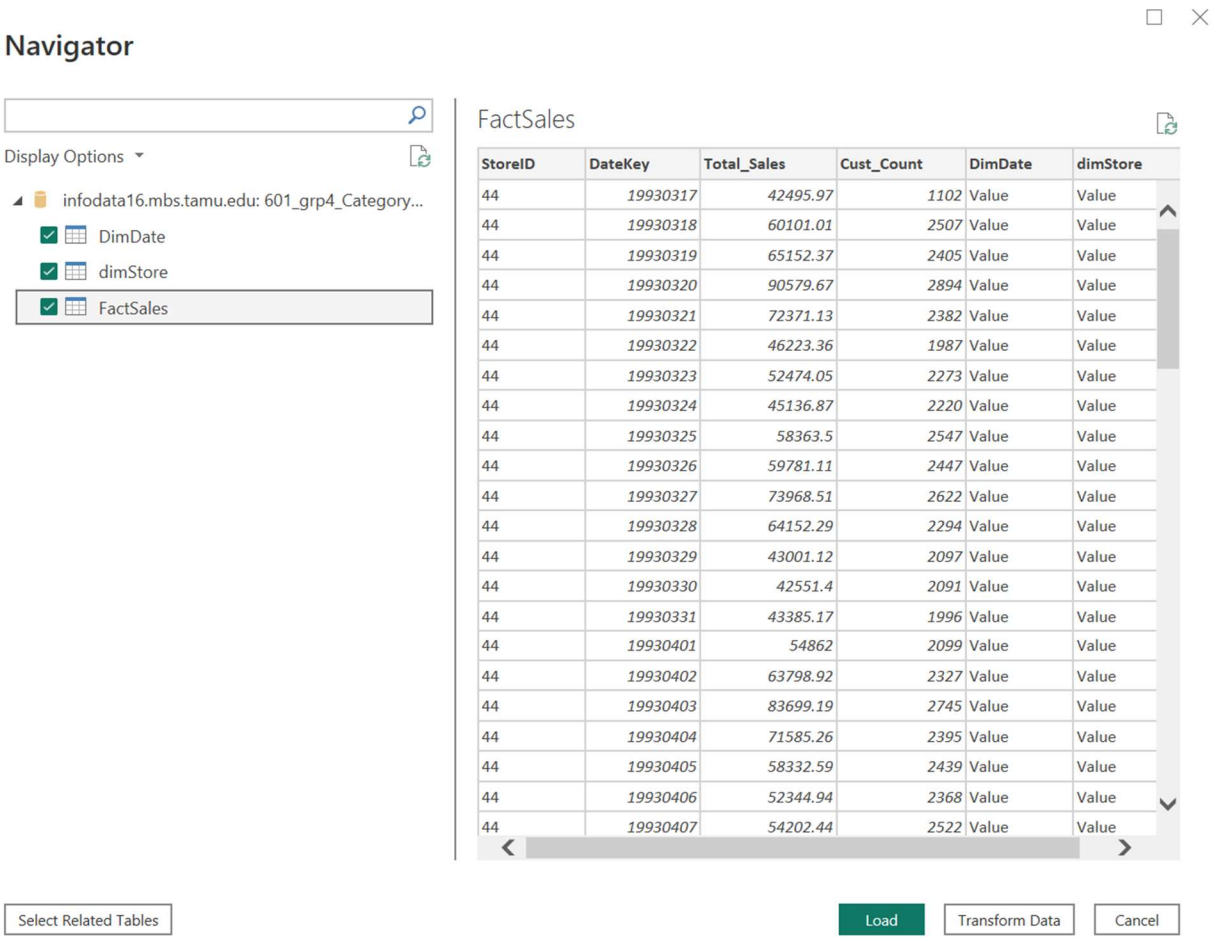


Fig 63. Tables selected from CategorySalesDW DataMart into PBI

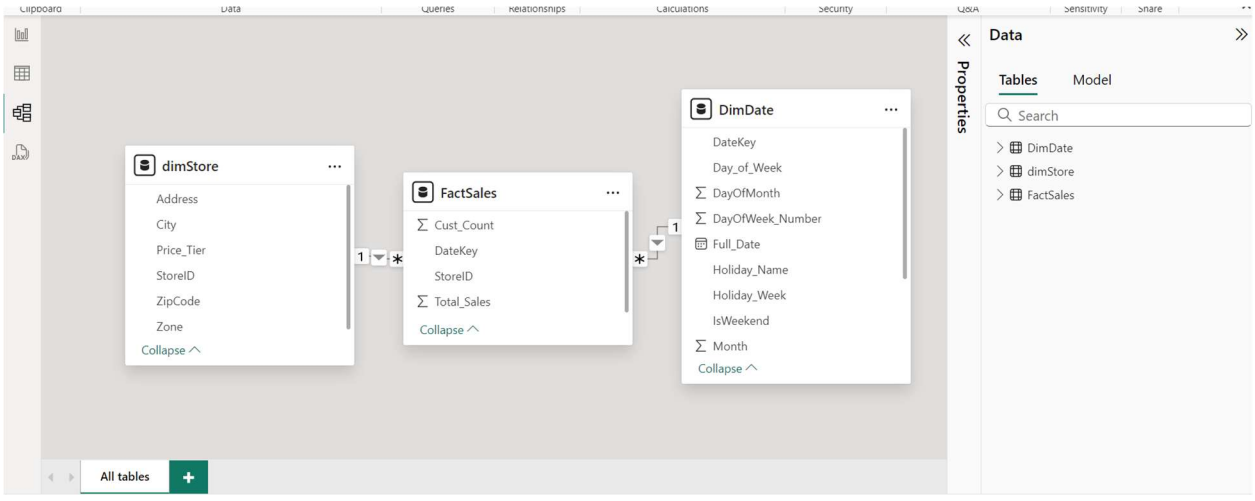


Fig 64. Model View in PBI

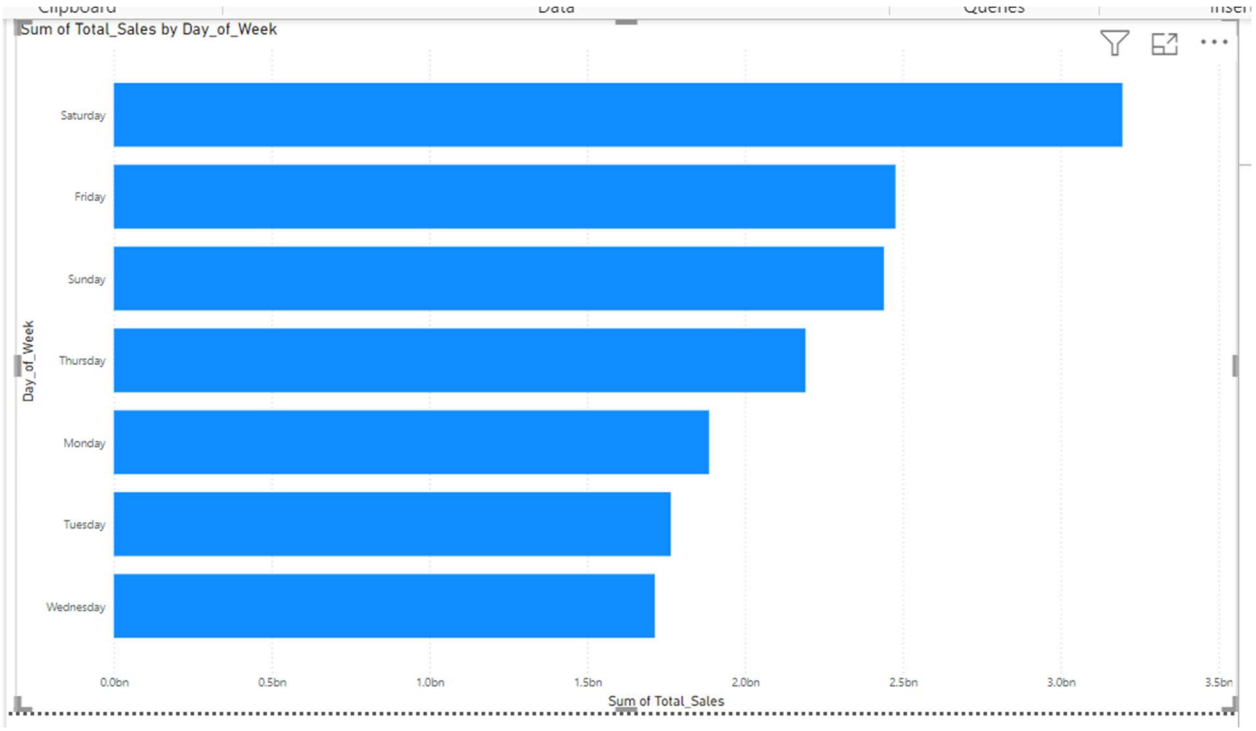


Fig 65. Day of the Week vs Total Sales for all the stores and all the years

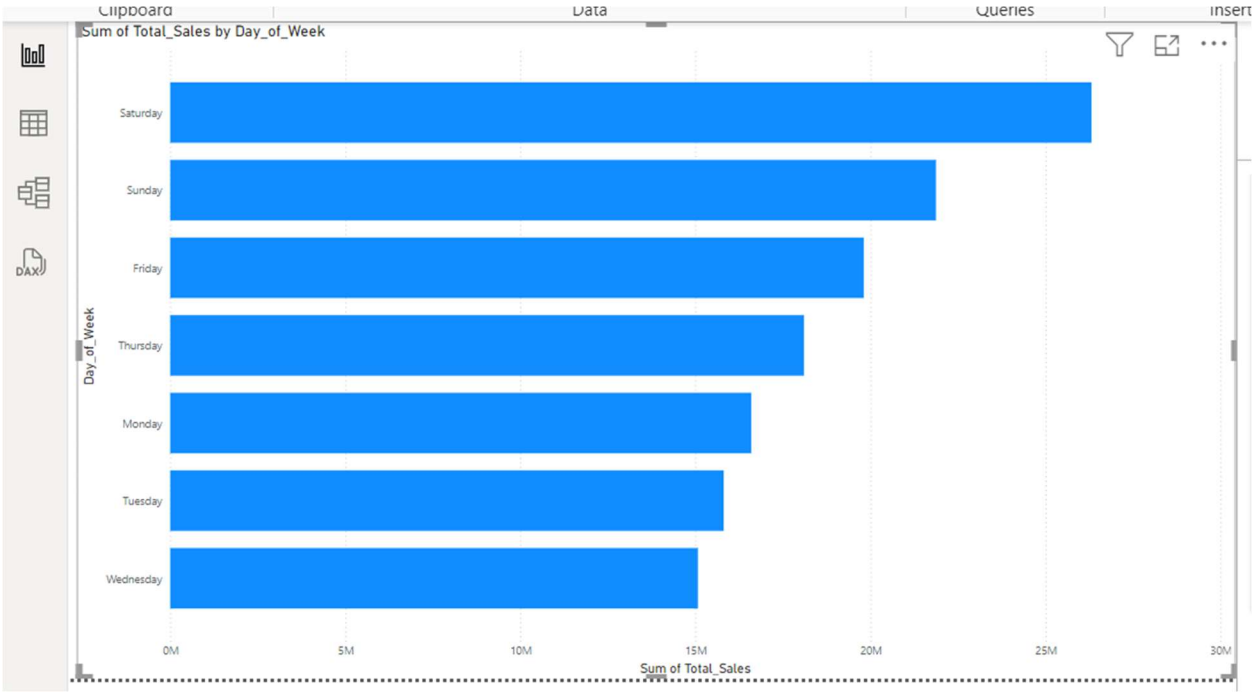


Fig 66. Day of the Week vs Total Sales for Store 2 and all the years

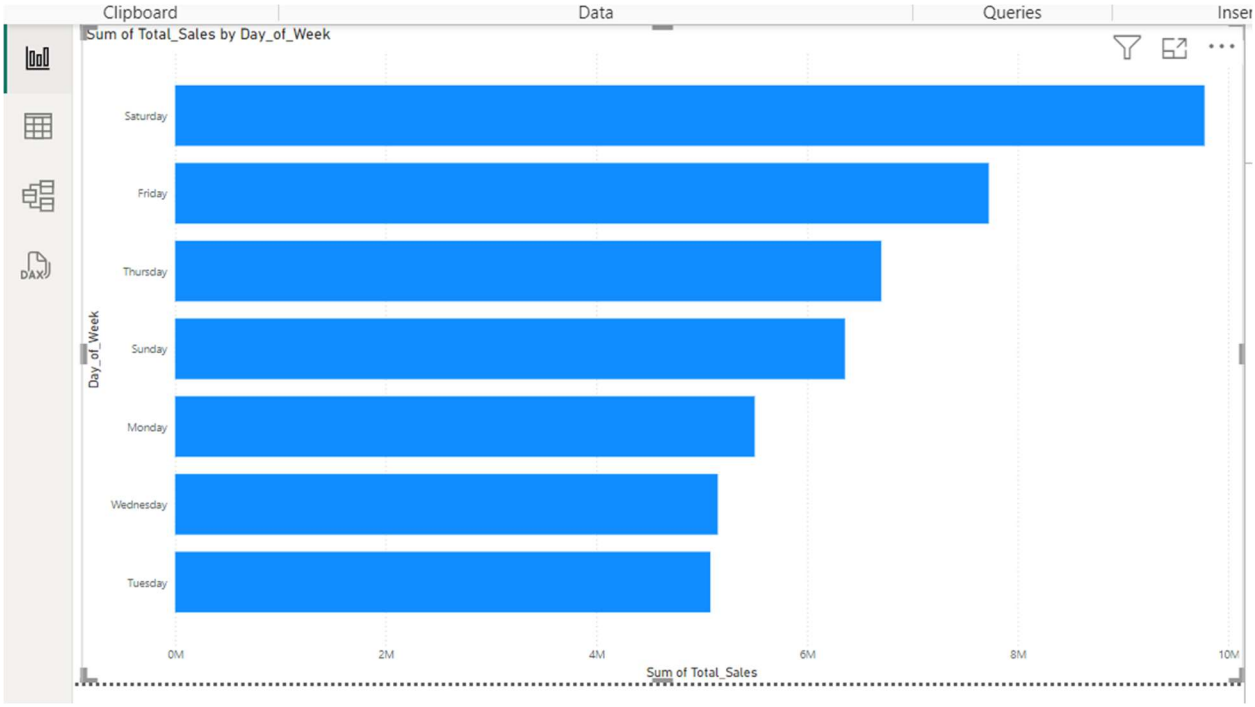


Fig 67. Day of the Week vs Total Sales for Store 4 and all the years

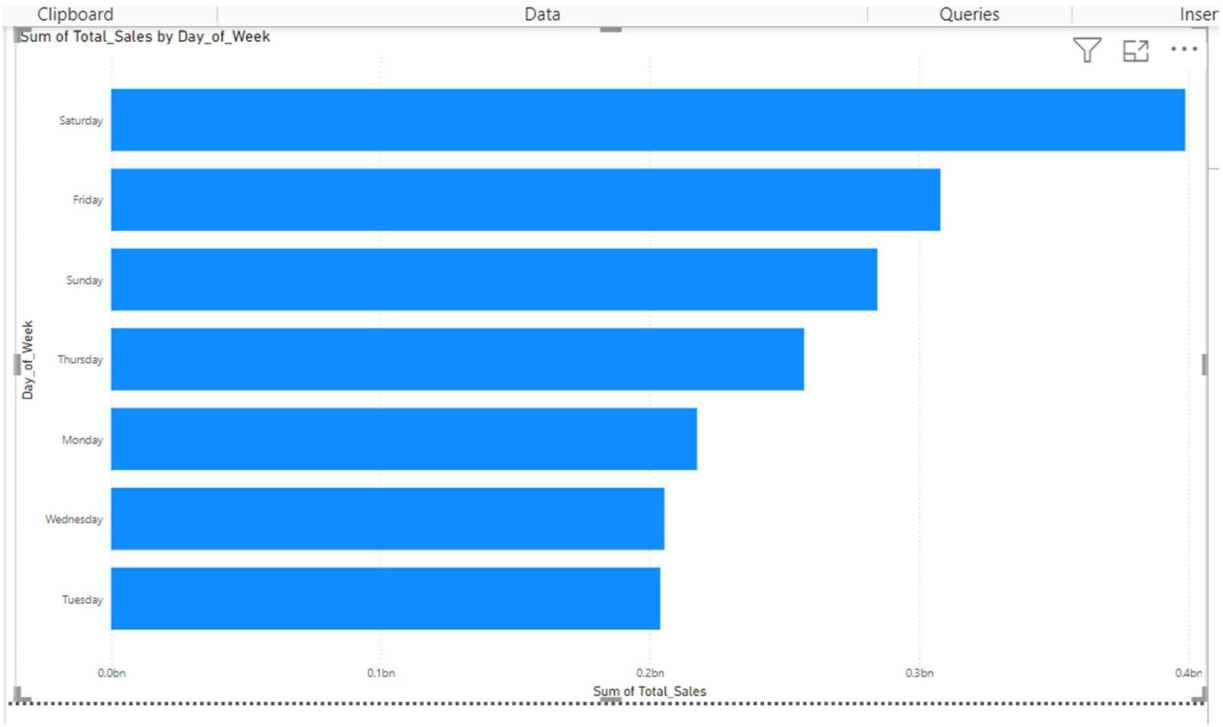


Fig 68. Day of the Week vs Total Sales for all the stores and year 1989

- ## Results and Analysis

1. **Weekend Dominance**:
    - Across all filters, Saturdays consistently generate the highest sales, emphasizing their importance for revenue generation.
    - Fridays and Sundays are also strong performers, suggesting that promotions during these days can maximize impact.
2. **Store-Specific Insights**:
    - While the overall trend is stable, individual stores like ID 2 and ID 4 show slight variations, such as Thursday being a stronger day for Store ID 4.
    - Tailored strategies for each store based on its unique traffic and sales patterns can lead to better outcomes.
3. **Midweek Sales**:
    - Midweek days like Tuesday and Wednesday generally underperform, presenting opportunities for businesses to introduce incentives like "midweek discounts" to drive traffic during these slower periods.
4. **Temporal Consistency**:
    - The year-specific data (1989) aligns closely with overall trends, confirming that these patterns are stable over time and can guide future sales strategies.

By analyzing sales trends under these different filters, businesses can identify high-performing days, store-specific patterns, and areas where improvements can be made to optimize sales and inventory management.

➢ Business Question - 4

- ## Mappings from Independent Data Marts to Report Attributes

Independent Data Mart	Attribute	Report Attribute	Mapping Logic
FactSales	Total_Sales	Total Sales	Aggregated sales for each holiday week.
DimDate	Holiday_Week	Holiday Week	Extracted to categorize sales by holidays.
DimStore	StoreID	Store Filters	Used to filter by specific stores.
DimDate	Year	Year Filters	Used to analyze sales across years.

- ## Description of Report Templates

Graph Type

Type: Clustered Bar Chart

Purpose:

- The report visualizes **total sales across various holiday weeks**, enabling easy comparison of performance between holidays.

- It supports the analysis of sales trends at both **aggregate and granular levels**, such as across multiple years, specific stores, and holiday weeks.
- By identifying high-performing holidays and year-over-year patterns, the chart provides insights to optimize resource allocation, inventory management, and marketing strategies.
- The inclusion of **filters for stores and years** allows for customized analysis, aiding in tailored decision-making for localized and time-specific strategies.

Axes

- **X-Axis**:
  - Total Sales (Numeric) – Aggregated total sales across different combinations of filters.
- **Y-Axis**:
  - Holiday Week (Categorical) – Lists holidays such as Thanksgiving, Christmas, etc.

Filters

- **Year**: Analyzes sales across specific time periods (e.g., 1989-1996).
- **StoreID**: Focuses on performance for individual stores or groups of stores.

Building the Report – Power BI with SSAS cube

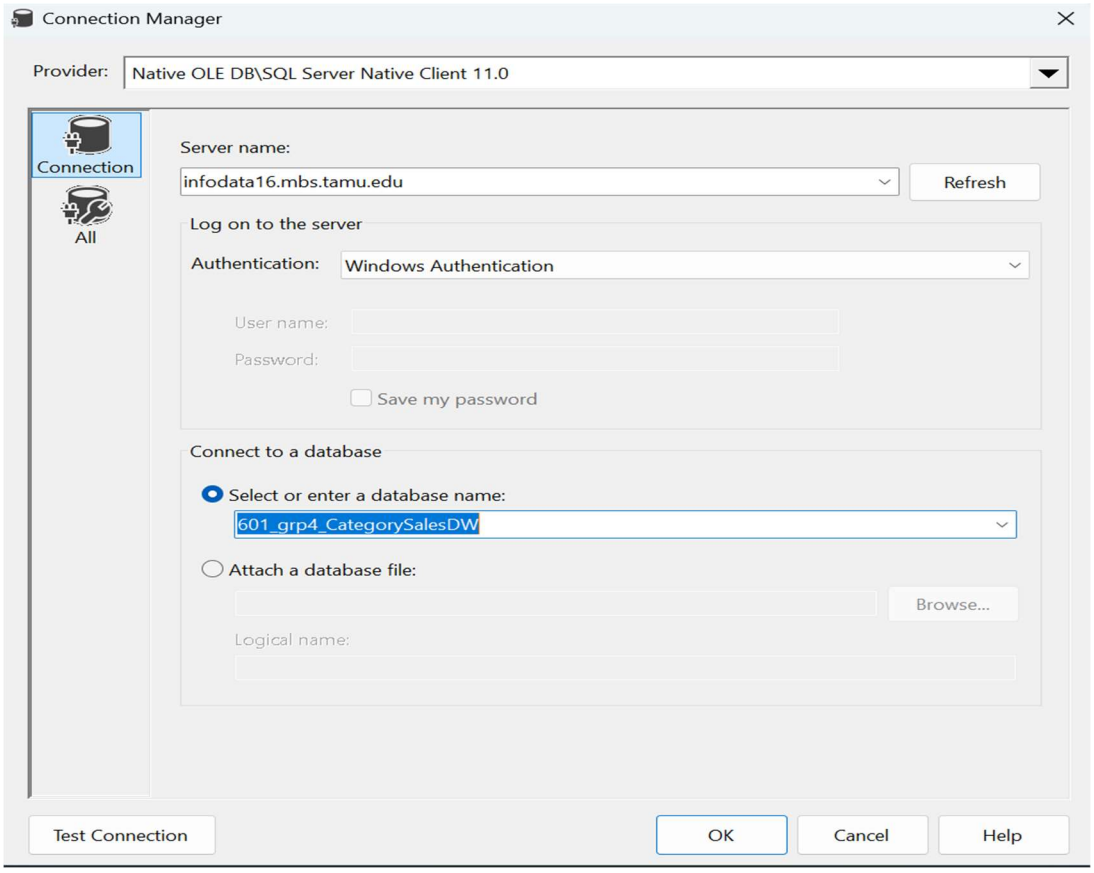


Fig 69. Data Source Creation

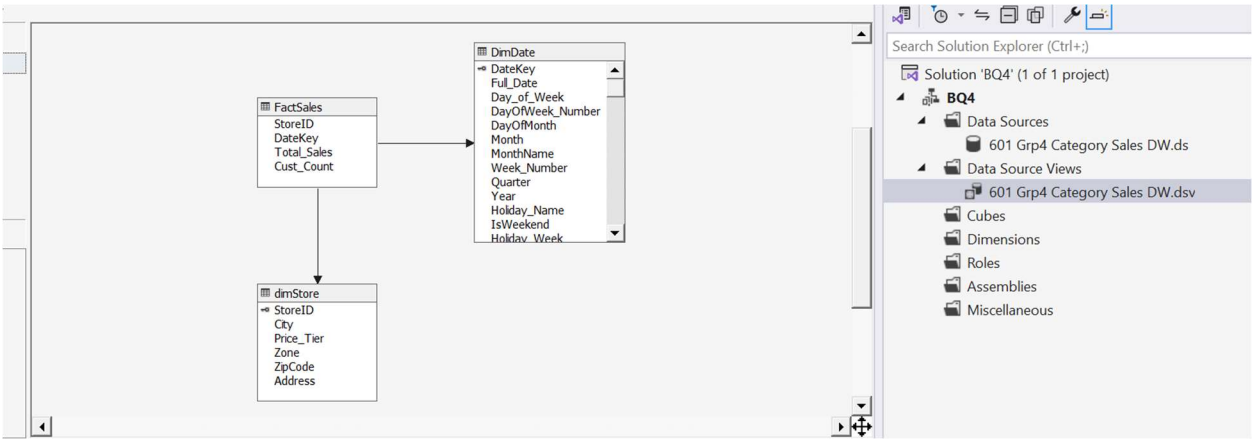


Fig 70. Data Source View Creation

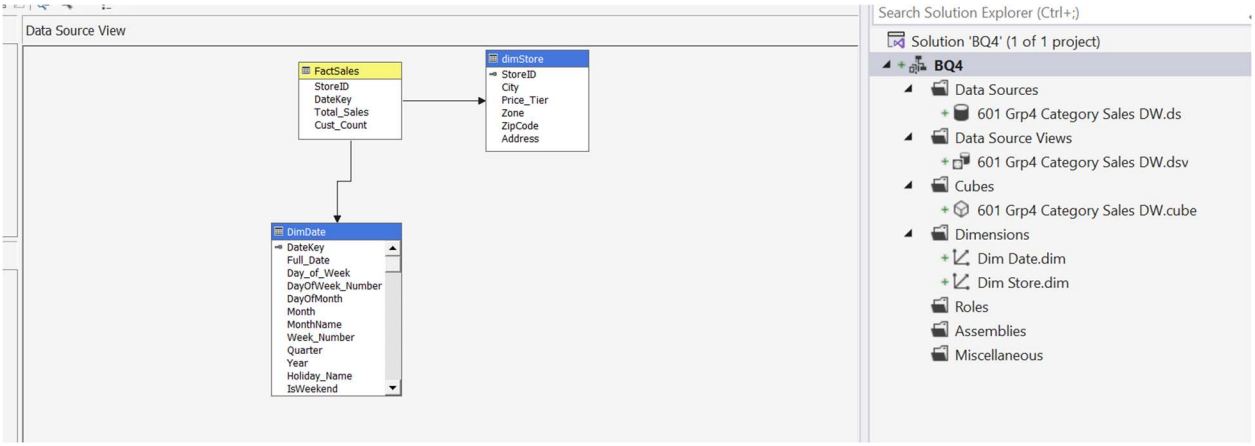


Fig 71. Data Cube Creation

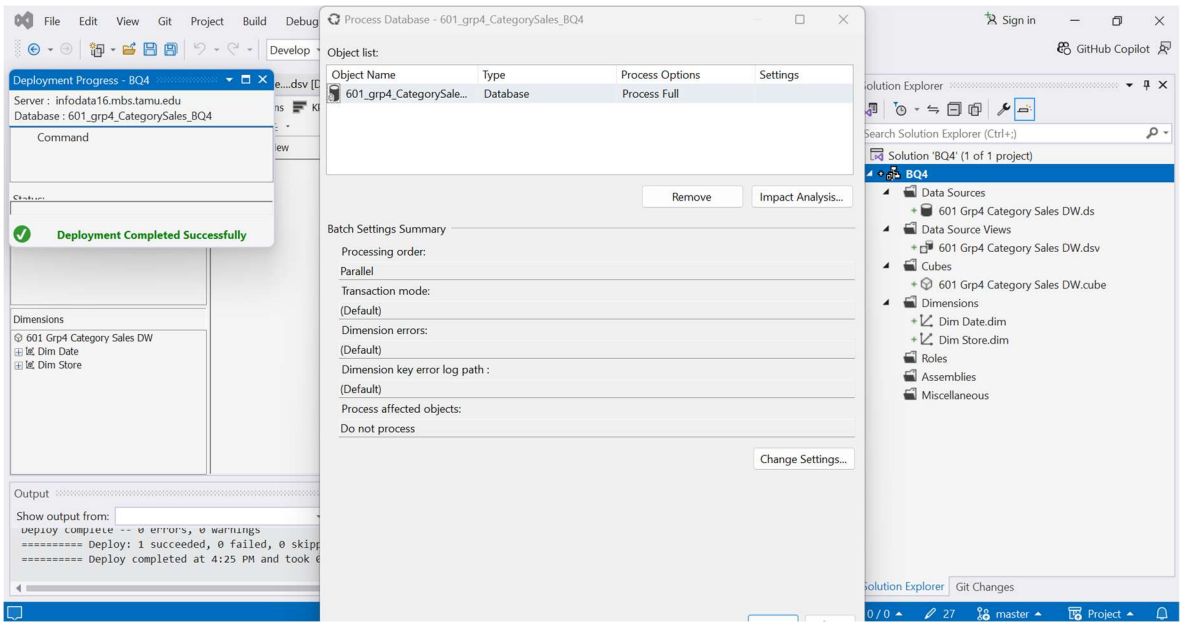


Fig 72. Deploying SSAS cube to infodata server

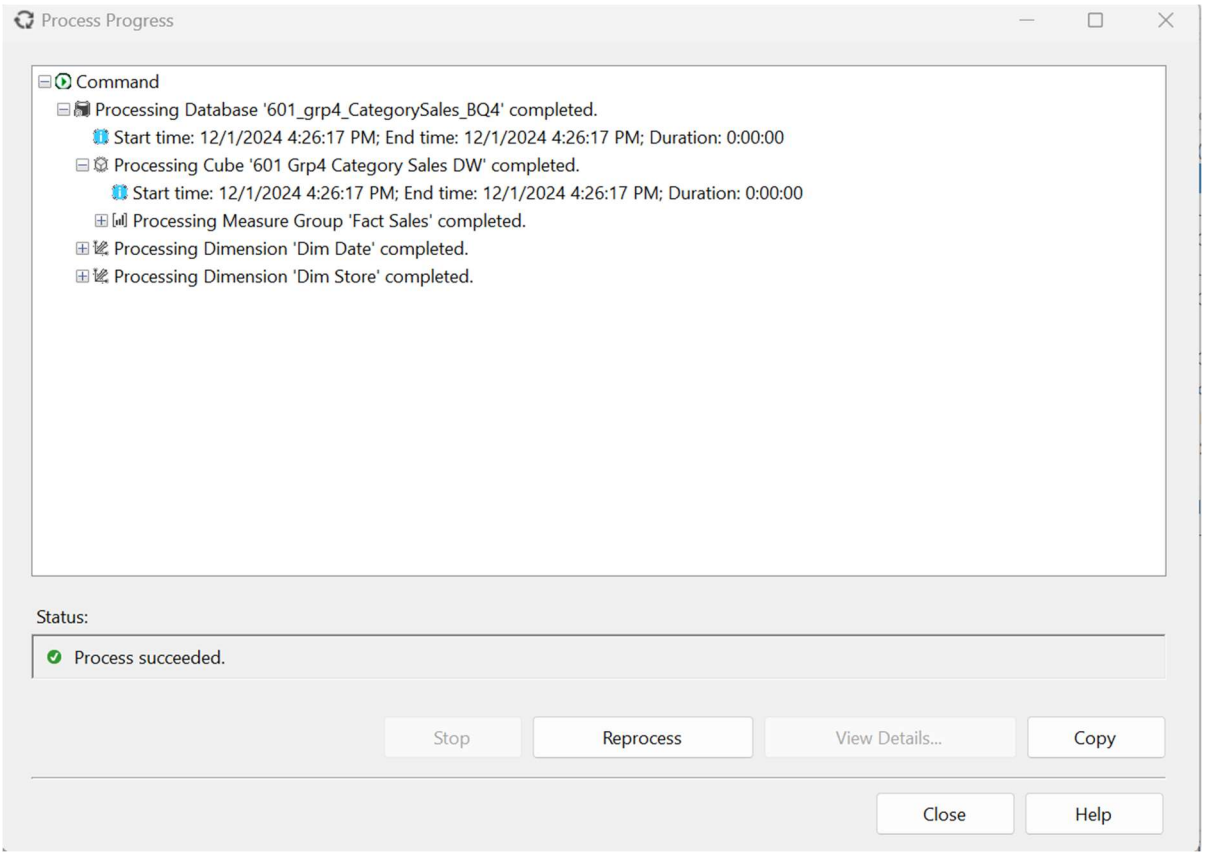


Fig 73. Successful Deployment

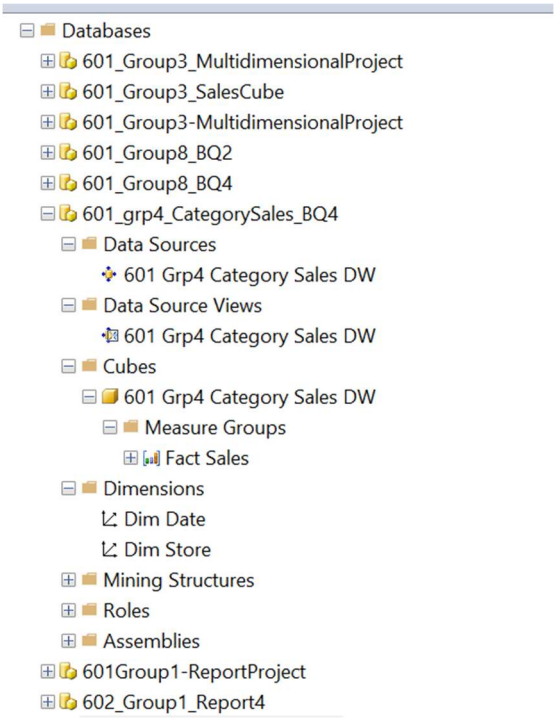


Fig 74. Cube part of infodata server

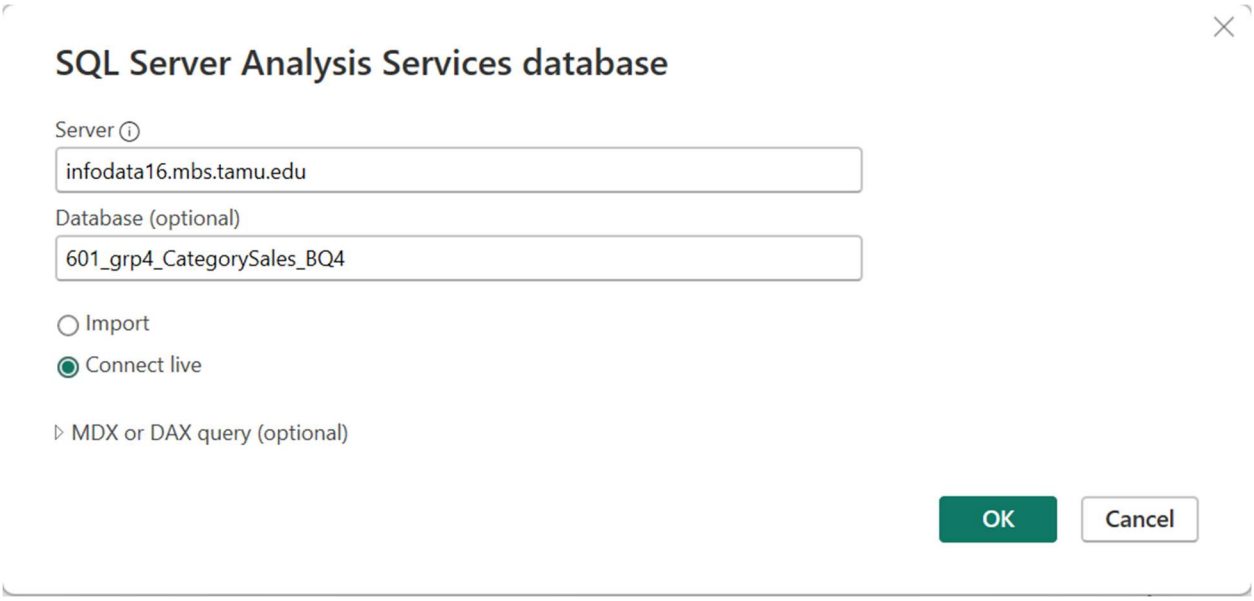


Fig 75. Connect the SSAS DB to Power BI

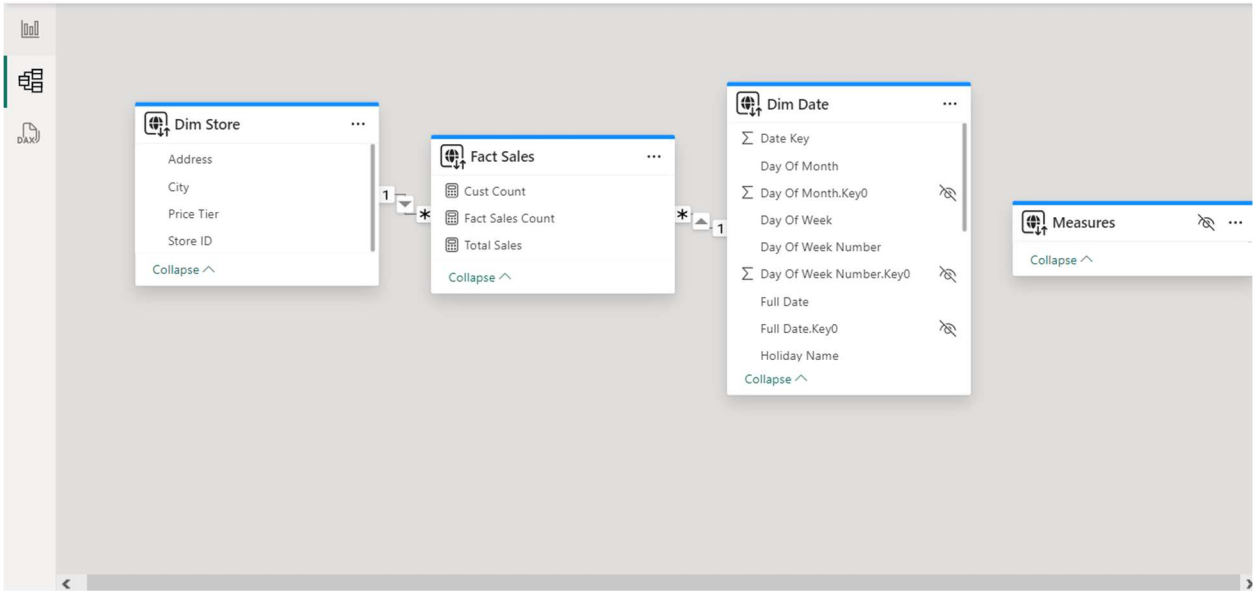


Fig 76. Model View of the imported DB

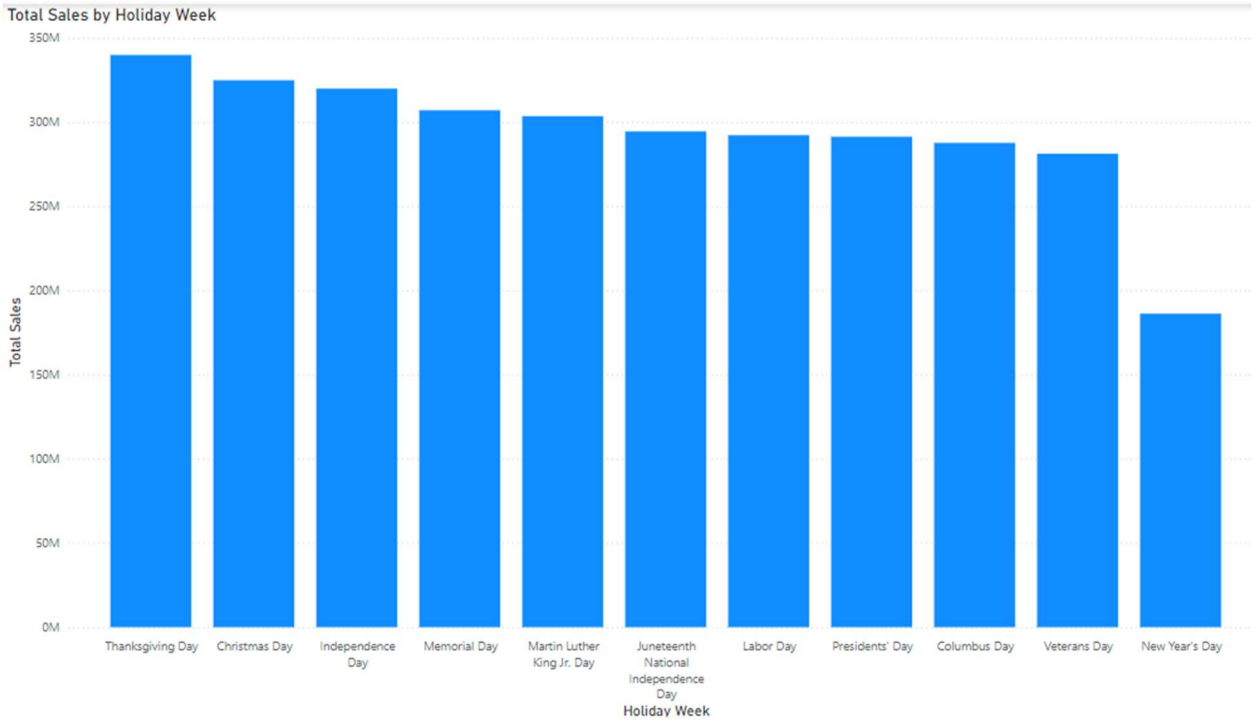


Fig 77. Sales on different Holiday weeks for all the years and for all the store

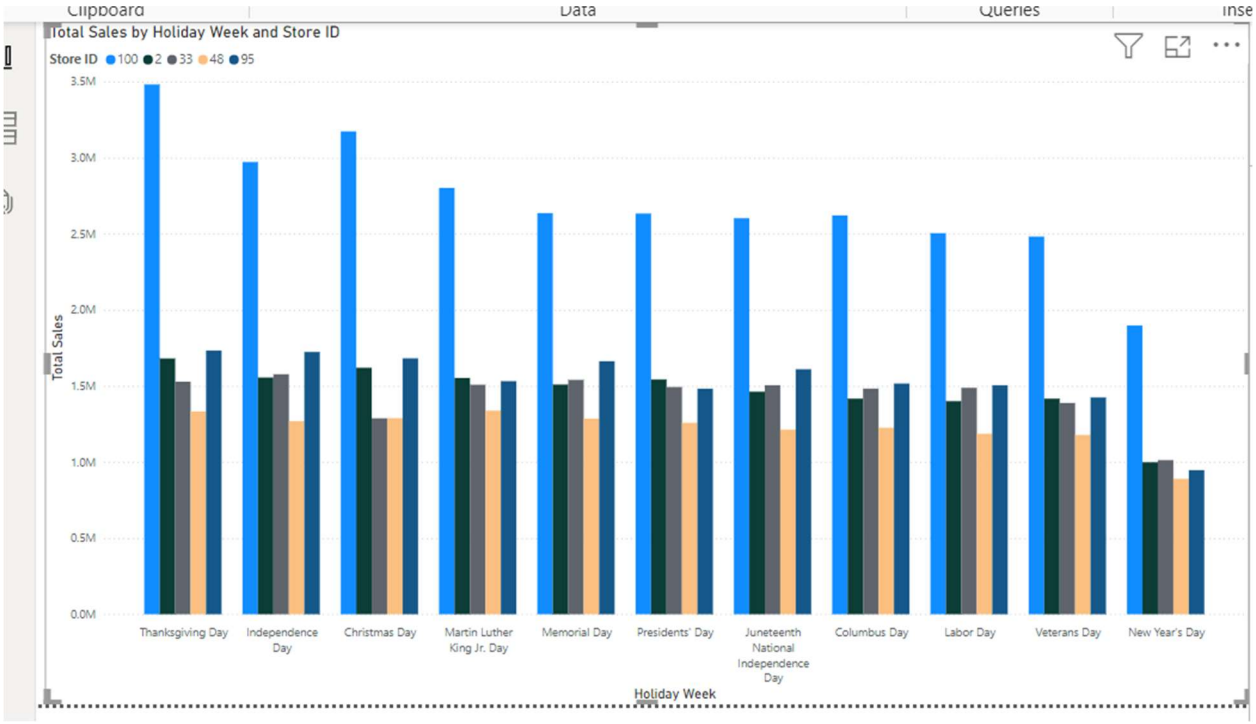


Fig 78. Sales on different Holiday weeks for all the years and for 5 stores

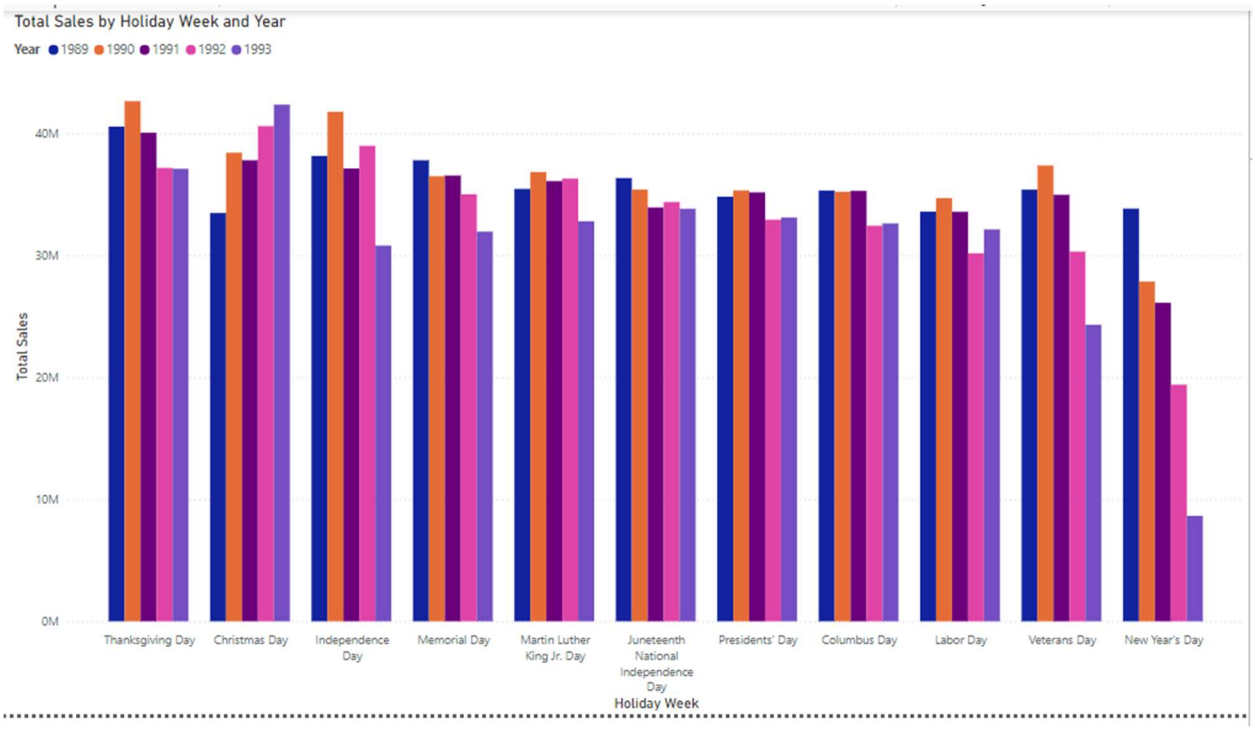


Fig 79. Sales on different Holiday weeks for all the store for 5 years

- ## Results and Analysis

1. Peak Sales Holidays:

- Thanksgiving and Christmas consistently show the highest sales, likely due to increased consumer spending on food, gifts, and celebrations during these periods.
- These holidays are pivotal for revenue generation and require focused marketing, inventory management, and staffing to handle the heightened demand.

2. Moderately High Sales Holidays:

- Independence Day, Labor Day, and Memorial Day also contribute significantly to sales, driven by outdoor gatherings and holiday-specific purchases.
- These holidays are less intensive than Thanksgiving or C. hristmas but still offer opportunities for tailored promotions.

3. Relatively Low Sales Holidays:

- Holidays like Juneteenth and Martin Luther King Jr. Day tend to show lower sales, possibly due to lower consumer spending patterns compared to major holidays.
- These periods may benefit from targeted discounts or localized campaigns to boost sales.

4. Year-over-Year Growth and Fluctuations:

- Some holidays, like Thanksgiving, demonstrate steady growth in sales over the years, reflecting increasing customer participation or economic factors.
- Other holidays, such as New Year's Day, show a decline in sales, which might suggest changing consumer behavior or reduced importance in holiday shopping.

5. Store-Level Variations:

- Sales performance during holiday weeks varies significantly across stores. Larger stores or those in high-traffic locations (e.g., Store ID 100) tend to dominate sales during all holidays.
- Smaller stores show less consistent performance, indicating the need for localized strategies tailored to their specific market dynamics.

6. Operational Implications:

- Major holidays like Thanksgiving and Christmas require careful planning for staffing and inventory to meet high demand.
- For moderate or low-sales holidays, businesses can focus on cost-effective promotions to drive traffic without over-committing resources.

7. Seasonal Campaign Opportunities:

- Leveraging holidays with high consumer spending (e.g., Christmas) for themed promotions can boost sales significantly.
- Similarly, utilizing moderate-sales holidays (e.g., Labor Day) to clear out older inventory can enhance operational efficiency.

8. Insights for Marketing:

- Holidays with peak sales should have marketing campaigns emphasizing early planning and engagement to capture demand early.
- Identifying holidays with lower sales provides opportunities for improvement through strategic promotions, partnerships, or localized events.

Conclusion

By analyzing sales trends across holiday weeks, Dominick's Finer Foods can identify high-priority holidays, adapt to year-over-year trends, and optimize resources effectively. This comprehensive understanding enables the business to capitalize on peak sales periods while improving performance during lower-traffic holidays.

➢ Business Question - 5

▪ Mappings from Independent Data Marts to Report Attributes

Independent Data Mart	Attribute	Report Attribute	Mapping Logic
FactSales	Total_Sales	Sum of Sales	Aggregated sales amount for each product.
FactSales	Promotion_Status	Coupon Status	Categorized as **With Coupon** or **Without Coupon**.
DimProduct	Product_Description	Description	Product names used to analyze sales trends.
DimStore	StoreID	Store Filters	Filters used to analyze specific stores.

- Description of Report Templates

Graph Type

- **Type**: 100% Stacked Bar Chart
- **Purpose**:
    - This chart compares sales made **with coupons** and **without coupons** for various products.
    - It shows the relative contribution of promotional vs. non-promotional sales to the overall sales for each product, aiding in understanding the effectiveness of coupons.

Axes

- **X-Axis**:
    - **Percent of Sales** (Numeric) – Represents the proportion of sales with and without coupons, normalized to 100% for each product.
- **Y-Axis**:
    - **Description** (Categorical) – Lists product descriptions to identify sales distribution by item.

Filters

StoreID: Filters are applied to analyze sales performance in specific stores.

- **Building the Report – Power BI**

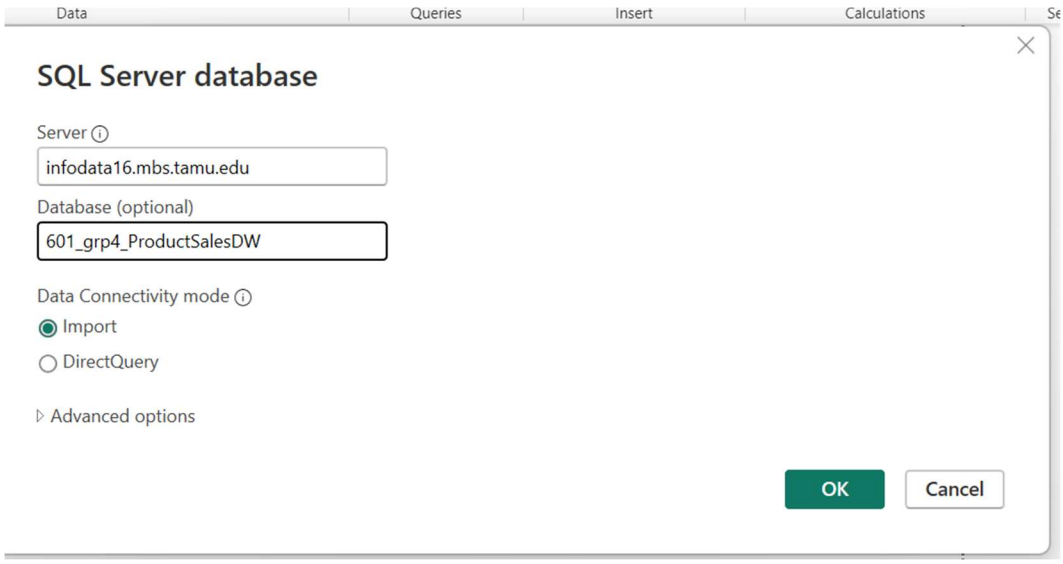


Fig 80. Import Data from data mart to power BI

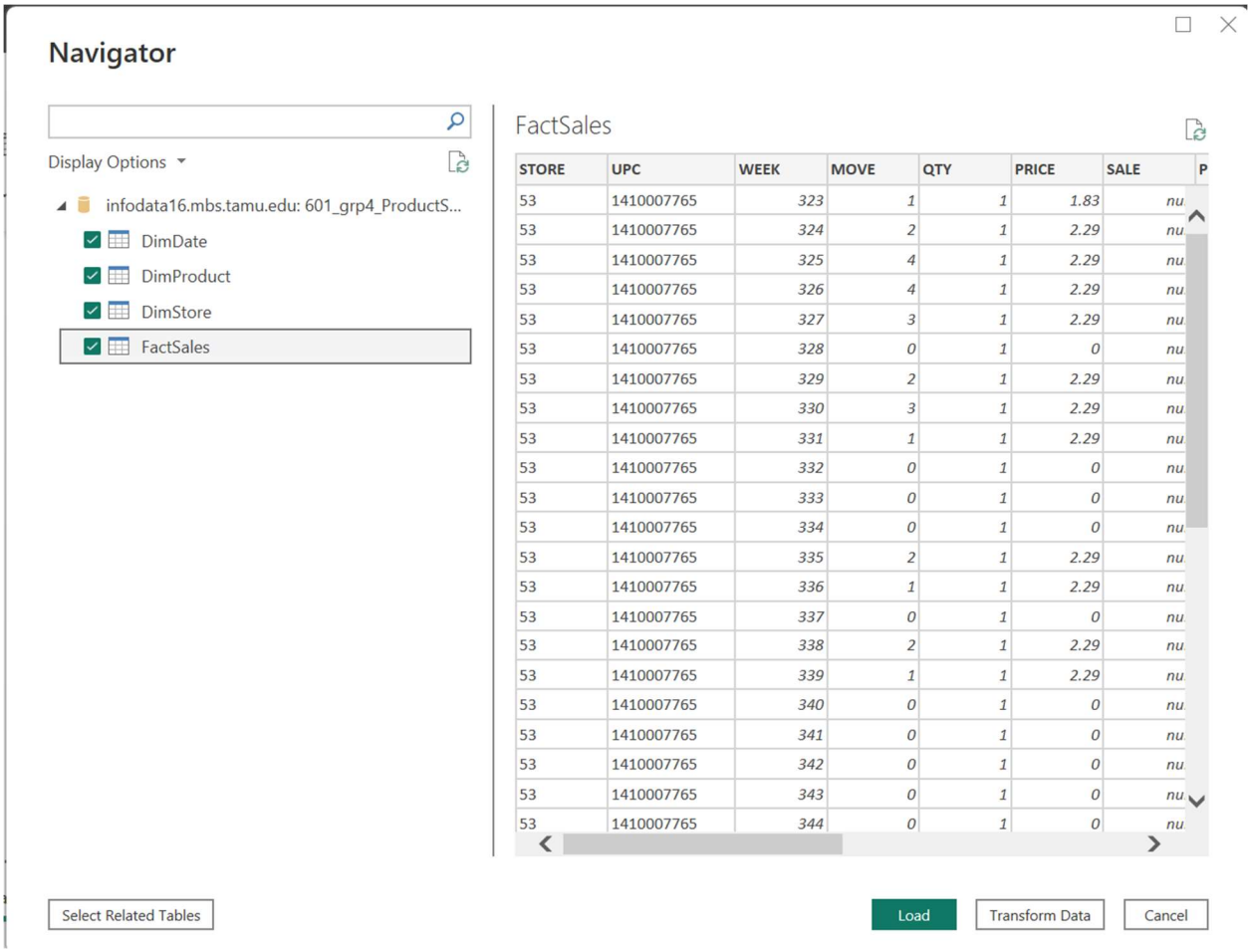


Fig 81. Preview of Imported Data

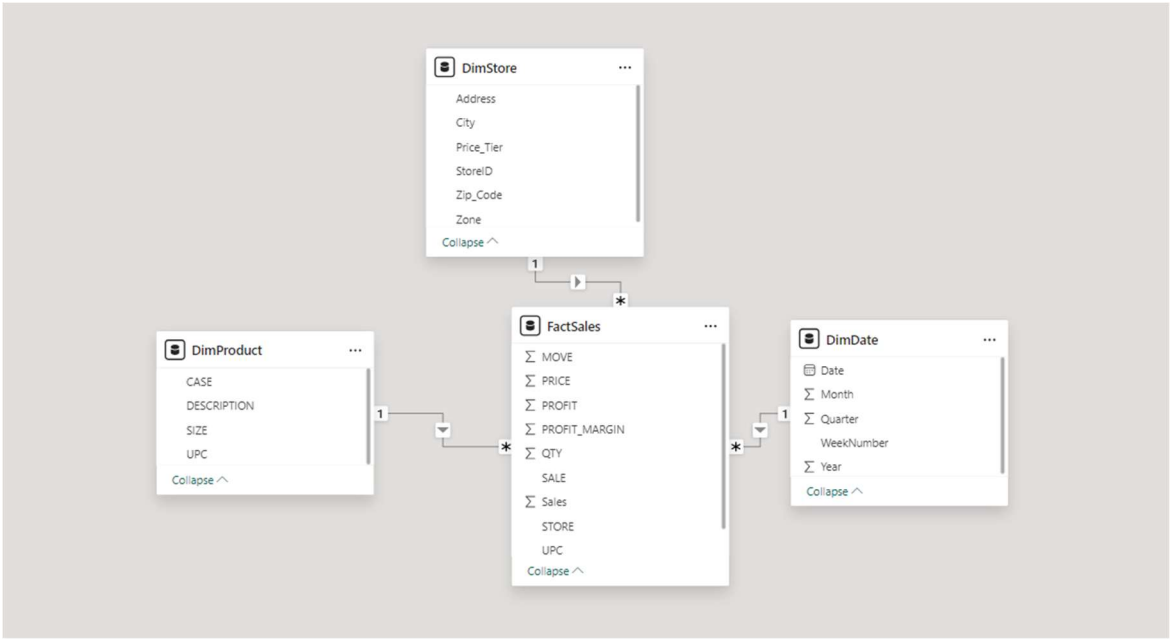


Fig 82. Model View

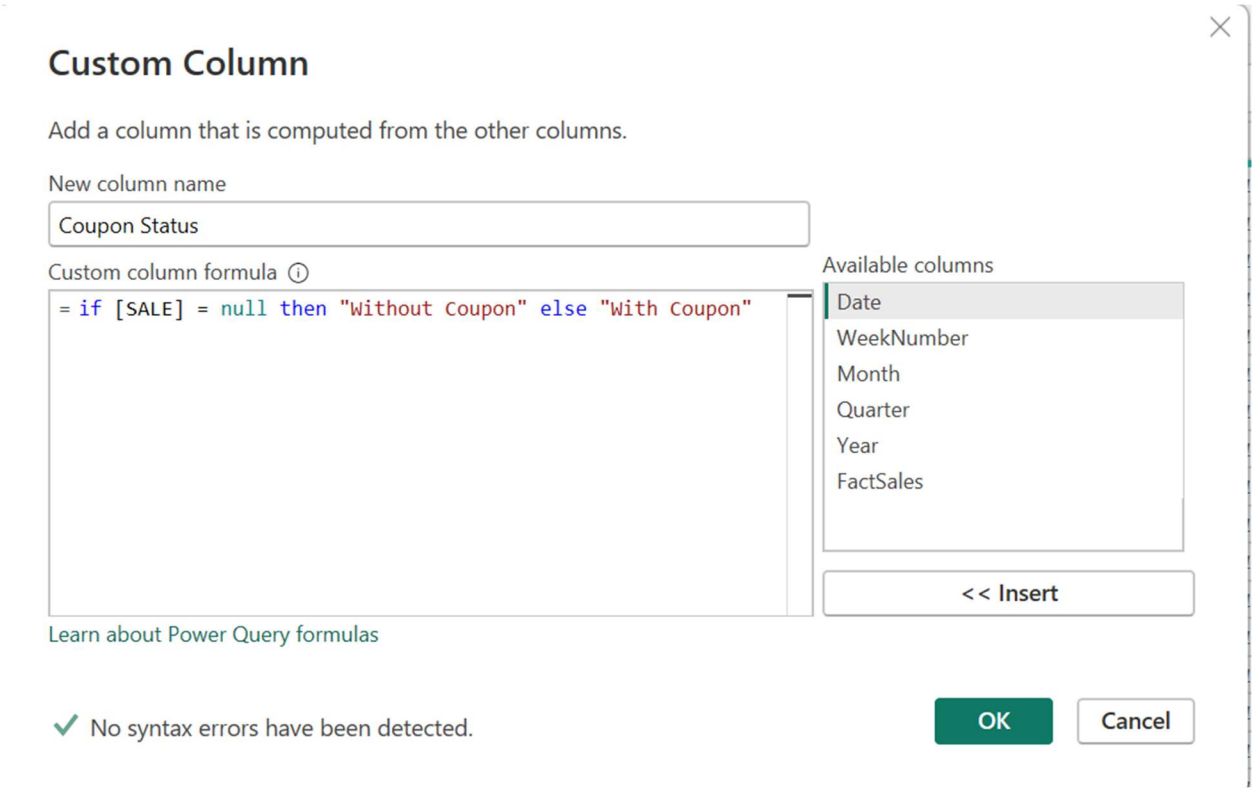


Fig 83. Add a column to identify if that sale was made with a promotion

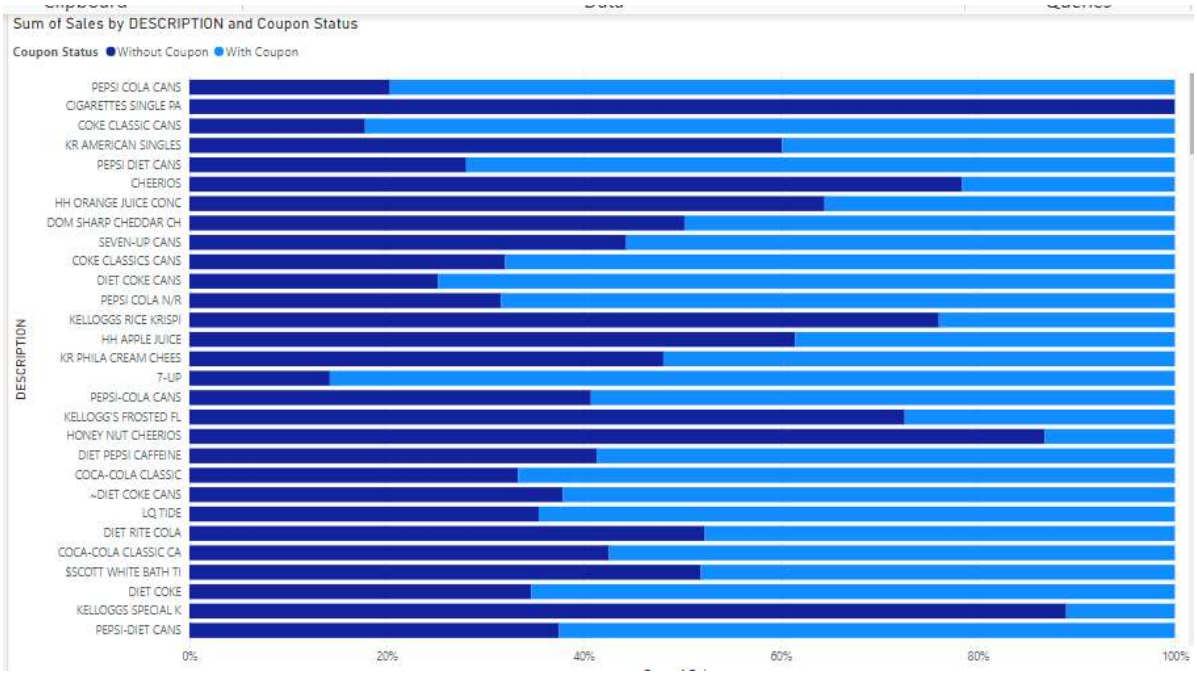


Fig 84. Product Sales with and without coupon for Store ID 2

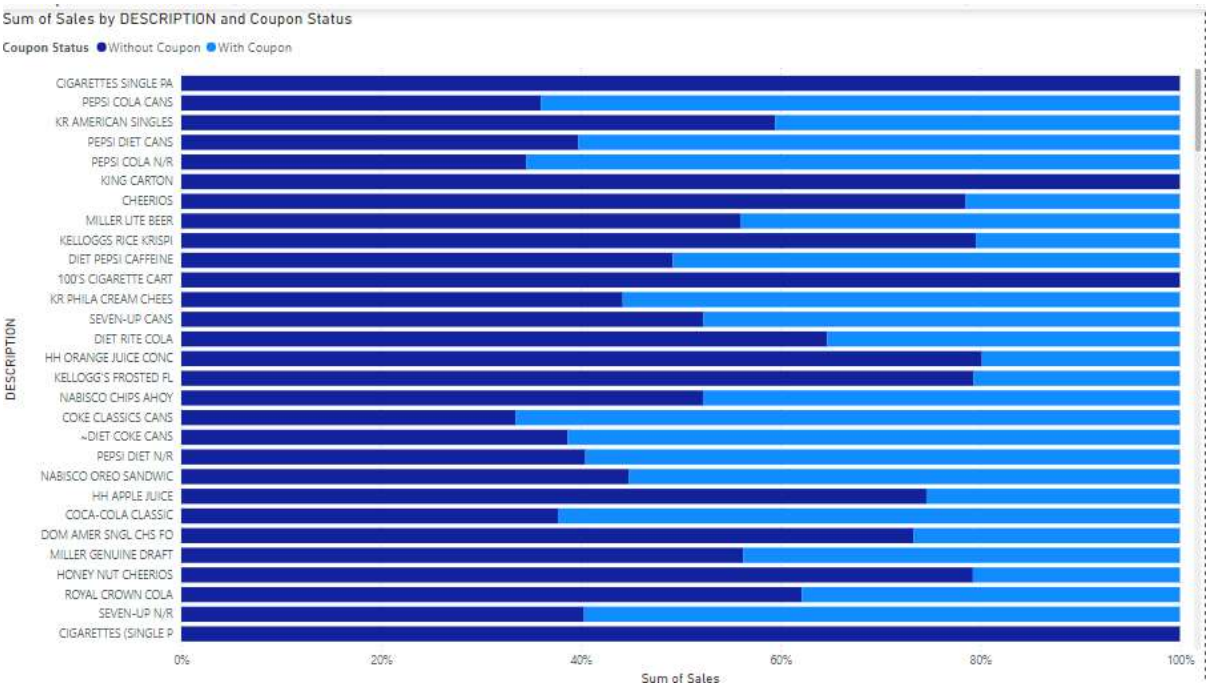


Fig 85. Product Sales with and without coupon for Store ID 50

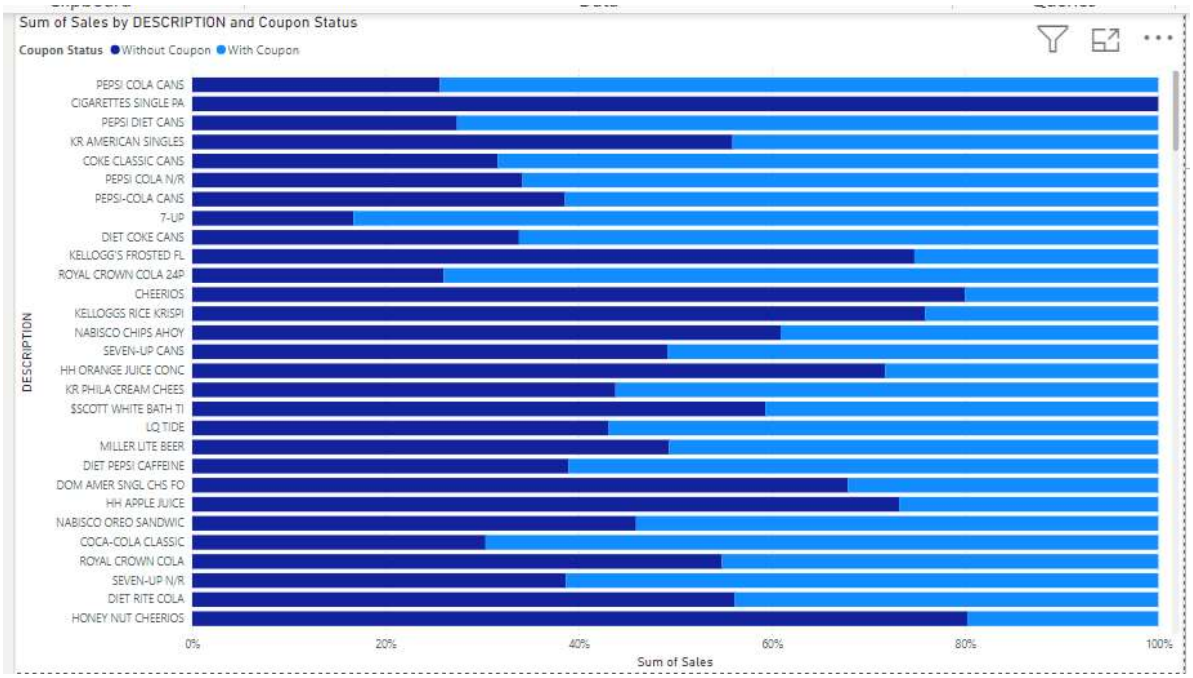


Fig 86. Product Sales with and without coupon for Store ID 8

▪ Results and Analysis

Store ID 2

- **Products That Sell Well With Coupons**:

    - **Pepsi Cola Cans**: The majority of sales for this product come from promotions, showing a high dependency on coupons to drive customer purchases.
    - **Diet Pepsi Cans**: Similar to Pepsi Cola, this product benefits significantly from promotional offers.
    - **Cigarettes Single Pack**: Strong sales during coupon usage indicate customer sensitivity to price discounts on this item.
- **Products That Do Not Depend on Promotions**:
    - **Honey Nut Cheerios**: Sales are largely consistent regardless of promotions, showing this product's inherent demand.
    - **Coca-Cola Classic**: Non-coupon sales dominate, suggesting strong brand loyalty and regular purchasing behavior without the need for discounts.

Store ID 50

- **Products That Sell Well With Coupons**:
    - **Coke Classic Cans**: High coupon-driven sales demonstrate the product's success in leveraging promotions to boost purchases.
    - **Pepsi Diet Cans**: This product sees significant sales during coupon periods, making it a good candidate for frequent promotions.
    - **Kellogg's Rice Krispies**: Strong dependency on coupons suggests promotional campaigns drive its sales.
- **Products That Do Not Depend on Promotions**:
    - **Royal Crown Cola**: Sales remain steady without coupons, showing low dependency on promotions.
    - **Kellogg's Frosted Flakes**: Regular purchases regardless of discounts indicate a loyal customer base for this product.

Store ID 8

- **Products That Sell Well With Coupons**:
    - **7-Up**: A significant portion of sales comes from promotions, highlighting its reliance on discounts to attract customers.
    - **Diet Coke Cans**: Similar to 7-Up, this product performs well during promotional periods.
    - **Scott White Bath Tissue**: A household essential that sees increased sales with coupons, making it a key promotional product.
- **Products That Do Not Depend on Promotions**:

- **Diet Rite Cola**: High non-coupon sales show that customers purchase this product regardless of promotions.
- **Honey Nut Cheerios**: Consistent sales with minimal promotional impact demonstrate its steady demand.

Why These Metrics Are Useful

1. **Targeted Promotions**:
   - **Products That Sell Well With Coupons**: These items are highly responsive to discounts and should be included in frequent promotions to maximize customer traffic and sales during promotional periods.
   - Example: Including popular beverages like **Pepsi Cola** or **Diet Coke** in promotions can attract more customers to the store.
2. **Efficient Resource Allocation**:
   - **Products That Do Not Depend on Promotions**: These items generate stable revenue without the need for discounts, allowing stores to allocate promotional budgets more effectively toward items that need a boost.
   - Example: Avoid over-promoting **Honey Nut Cheerios** and **Diet Rite Cola**, which already sell well without discounts.
3. **Inventory Planning**:
   - Understanding the products that sell well with coupons helps stores plan inventory more effectively, ensuring sufficient stock is available during promotional periods to meet demand.
4. **Customer Behavior Insights**:
   - Products with consistent non-coupon sales reflect brand loyalty or essential needs, providing insights into customer preferences that can guide product placement and marketing strategies.
5. **Profitability Optimization**:
   - By focusing promotions on items that yield significant sales increases and minimizing discounts on products with stable demand, stores can optimize profitability while maintaining customer satisfaction.

Conclusion

These metrics allow Dominick's Finer Foods to strategically plan their promotions, tailor marketing campaigns to customer preferences, and allocate resources more effectively. Understanding which products depend on promotions and which do not enables the store to strike a balance between driving sales and maintaining profitability, ensuring long-term business success.