



CONSULTING REPORT

Integrated Report 4: Integrated Consulting report with BI Report Design

Submitted By:

Group 5

Karan Gupta

Namrta Singh

Shreya Apte

RESOURCES:

1. **PC - labs-184-15.mbs.tamu.edu**
2. **Staging Database Name:** 601-group5-staging-area
3. **Warehouse Database Name:** 601-group5-dw-area
4. **SSAS Database Name:** Section601-Group5-SSAS
5. **SSRS Deployed Report URL:**
<http://infodata16.mbs.tamu.edu/ReportServer/Pages/ReportViewer.aspx?%2fSection601Group5-SSRS%2fBusiness+Question+1&rs:Command=Render>
6. **SSRS+SSAS Deployed Report URL:**
<http://infodata16.mbs.tamu.edu/ReportServer/Pages/ReportViewer.aspx?%2fSection601-Group5-SSAS-SSRS%2fBusiness+Question+3&rs:Command=Render>
7. **PowerBI Deployed Report URL:**
<https://app.powerbi.com/view?r=eyJrIjoiNzliNDg0OWQtNzJmNy00M2Y2LWFmNGItODE2NjVjMjI3MjBjIiwidCI6IjY4ZjM4MWUzLTQ2ZGEtNDdiOS1iYTU3LTZmMzIyYjhMGRhMSIsImMiOjN9>

TABLE OF CONTENTS

1.	INTRODUCTION.....	6
2.	DETAILS ABOUT THE DATA	7
2.1	Data Understanding	7
2.2	Metadata	8
2.3	ERD.....	13
3.	DOMAIN UNDERSTANDING.....	14
4.	BUSINESS QUESTIONS.....	16
5.	JUSTIFICATION OF SELECTED QUESTIONS.....	25
	Q1. How are average sales of a product changing by store from 1995 - 1997?	25
	Q2. What are the total sales of various food items on New Year's Eve over the years?	25
	Q3. What is the Sale of different kinds of alcoholic beverages over HOLIDAY weekS?.....	25
	Q4. Which is the most profitable detergent brand at a particular store?	25
	Q5. Which are the best performing stores for Jewelry sales where more than 10% of visitors are below the poverty line?	26
6.	INDEPENDENT DATA MARTS DESIGN USING KIMBALL'S APPROACH.....	27
a.	Information Package DIAGRAMS	27
7.	HIERARCHICAL DIMENSION AND FACT TABLES	29
7.1	DIMENSION TABLES.....	29
7.2	FACT TABLES	30
7.3	FACT AND DIMENSIONS	32
8.	DIMENSIONAL MODELLING (STAR SCHEMA).....	33
8.1.	Product Sales Data Mart	33
8.2.	PRODUCT Category sales Data Mart.....	34
8.3.	STOREWISE AGGREGATED Category Sales Data Mart.....	35
9.	MAPPING TABLES.....	36
10.	PHYSICAL DESIGN PLAN	39
	10.1 Standardization	39

10.2. Aggregation	40
10.3. Partitioning	40
10.4. Clustering.....	40
10.5. Indexing	41
10.6. Data Structure for storage	41
11. DATA QUALITY ISSUES	42
12. ETL DEVELOPMENT PLAN	43
12.1 Extraction plan.....	44
12.2 Data Transformation	44
12.3 Loading.....	45
13. DATA SOURCES AND TARGET DATA.....	45
13.1 Target Data Needed in the warehouse	45
13.2 Determining Data Sources	46
13.3 Data Mappings for data elements from Sources to Staging and Staging to Data warehouse	47
14. DATA EXTRACTION RULES	52
15. DATA TRANSFORMATION AND CLEANSING RULES	53
15.1 Removal of unwanted data	53
15.2 Removal of inconsistent data/special characters.....	53
15.3 Removal of NULL values	53
15.4 Data conversion	53
15.5 Creation of Surrogate Keys.....	54
15.6 Creation of Derived Columns.....	54
15.7 SSIS Functions	55
16. PLAN FOR AGGREGATE TABLES	56
17. ORGANISATION OF DATA STAGING AREA	56
18. PROCEDURES FOR EXTRACTION and LOADING	58
18.1 EXTRACTION PROCEDURES.....	58
18.2 LOADING PROCEDURES	59
19. ETL FOR DIMENSIONS and FACT TABLES.....	61

19.1 ETL for Dimension Tables.....	61
19.2 ETL for Fact Tables	62
20 IMPLEMENTATION OF ETL.....	64
20.1 EXTRACTION OF DATA	64
20.1.1 Temporary Table Creation	64
20.2 Staging Area before cleaning and Transformation	77
20.3 Transformation and Cleaning of dATA.....	78
Store.....	78
Ccount.....	79
Movement	84
WeekDecode	85
UPC.....	86
20.4 STAGING AREA AFTER CLEANING and TRANSFORMATION.....	87
20.5 LOADING OF DATA.....	88
20.5.1 CREATION OF DIMENSION TABLE	88
20.5.2 Populating Dimension Tables	89
20.5.3 CREATION OF FACT TABLES.....	91
20.5.4 POPULATING FACT TABLES	92
21 GRANULARITY OF DATA MARTS	101
22 TABLE CONTENT BEFORE AND AFTER ETL IN STAGING	101
22.1 Before ETL.....	101
22.2 After ETL.....	105
23 DELETION OF TEMP TABLES.....	110
24. REPORTING PLAN.....	111
24.1 Determining target reports that satisfy business questions.....	112
24.2 Mappings from independent data marts to report attributes	114
25. REPORT BUILDING.....	119
References.....	138

1. INTRODUCTION

Dominick's was a chain of retail stores founded in 1918 and located in Illinois. It used to be Chicago's second-largest retailer chain following Jewel which combined food stores and drug stores facilities with one. In 2013 Dominick's completely ceased operation due to low store performance in turn causing low sales over a period of time.

The dataset being analyzed in this study is from the Dominick's Fine Food (DFF) database obtained from Chicago Booth after both the parties developed collaboration for store-level research on shelf management and pricing. The data includes 25 product categories sold between 1989 and 1997 at DFF's 100 discount chain stores located in the Chicago metropolitan area.

This project will involve analysis of the data and through the construction of a data warehouse on top of the data to assist Dominick's Fine Food (DFF) to systematically evaluate and identify the factors affecting product sales to enhance revenue and business. This project report aims at formulating business questions that target these factors to help in solving key business problems pertaining to sales, profit, customer base, products etc.

The main challenge before identifying the key factors to make business decisions would be to understand the data. Due to the sheer volume of data, the task of identifying a logical relationship between the data is complex and requires sufficient domain knowledge. But it is also a very unique dataset, as it contains extensive information covering various aspects of a retail store in detail.

2. DETAILS ABOUT THE DATA

2.1 DATA UNDERSTANDING

As a part of the project, our team would analyze the data collected from James M. Klits Center, University of Chicago Booth School of Business. This data spans over 100 stores of Dominic's Fine Food and over 1989 to 1994. The size of the dataset collected is 5.6 GB. This data is highly comprehensive and has details about several aspects of the store like store-specific demographic customer details, UPC information, etc. It stores information about more than 3500 UPCs which were distributed by around 100 stores of DFF throughout the United States of America over 9 years. Most of these stores were located in Chicago.

In order to do any kind of analysis, the data has to be transformed from its current dirty state to a state in which useful insights can be generated.

File Details:

The data consists of five files which could be classified into two categories: General Files and Categorical Files. All of these files are present in .csv format.

General Files:

a. Customer Count Files

This file contains the information about traffic within the stores for all the stores i.e the number of customers visiting the store in a week, which was collected by the scanners present in the store. It also has details on coupons redeemed and sales in dollars.

b. Store-Specific Demographics

This file contains store specific information of all the customers based on their demography like age, ethnicity, household income etc. This customer profile has been collected by referring to the U.S. government (1990) census data for the Chicago metropolitan area. This kind of data can be important for a store for market segmentation and targeting customers.

Categorical Files:

c. UPC Files

UPC is used to identify the products and the manufacturer of the product. This file contains information about various products in the form their UPC, the commodity category it belongs to and the amount of product.

d. Movement Files by UPC

This file contains information about the movement of each product in terms of number of units sold and selling price information. This data is aggregated on a weekly basis. This file can be used to analyze the profit margin and focus on products which aren't selling well.

2.2 METADATA

2.2.1 Customer Count Files

The data in the customer count files deals with the number of customers the stores receives on daily basis, the total sales by each department, the datatype, the redeemed coupons etc.

The file contains variables as follows:

Variable	Description	Type	Length
STORE	Store Code	Numeric	3
DATE	Date of the Observation	Character	6
WEEK	Week Number	Numeric	6
MEAT	Meat Sales in Dollars	Numeric	8
BEER	Beer Sales in Dollars	Numeric	8
GROCERY	Grocery Sales in Dollars	Numeric	8
DELI	Deli Sales in Dollars	Numeric	8
DELI_COUP	Deli Coupons Redeemed	Numeric	8
FROZEN	Frozen Sales in Dollars	Numeric	8
DAIRY	Dairy Sales in Dollars	Numeric	8

2.2.2 Store Specific Demographics Metadata

The demographic data pertaining to the specific Dominick's Finer Foods stores is stored in this file. The information is very detailed and is based on the U.S. Government Census Data. The below table provides a description related to each attribute in **DEMO** file:

Variable	Description	Type	Length
MMID	Mobile Money Identification Number	Numeric	8
NAME	Name of Store	Character	16
CITY	Name of City	Character	16
ZIP	Zip Code of Area	Character	8
INCOME	Log of Median Income	Numeric	8
STORE	Store Number	Numeric	3
AGE60	% of population over 60	Numeric	8
POVERTY	% of population n with income under \$15,000	Numeric	8
AGE9	% of population under 9	Numeric	8

2.2.3 UPC

This file contains a record for each of the UPC in a particular category which is denoted by the last three letter acronym. The data is about product, the commodity code, and item code, description of the product, the number of units of the product, product size and the number of the units in the case. All the data is sorted according to the week, date and the UPC.

The table below provides a description of each of the attributes in UPC file.

Variable	Description	Type	Length
UPC	UPC Number	Numeric	8
COM_CODE	Dominick's commodity code	Numeric	8
NITEM	Dominick's item code	Numeric	8
DESCRIP	Product Name	Character	20
SIZE	Product Size	Character	6
CASE	Number of items in a case	Numeric	8

2.2.4 Movement

The data is about each UPC in a category at store level. The table below provides a description of each of the attributes in the **MOVEMENT** file.

Variable	Description	Type	Length
UPC	UPC Number	Numeric	8
STORE	Store Number	Numeric	3
WEEK	Week Number	Numeric	3
MOVE	Number of Units Sold	Numeric	8
PRICE	Retail Price	Numeric	8
QTY	Number of items bundled together	Numeric	3

PROFIT	Gross Margin	Numeric	8
SALE	Sale Code (B, C, S)	Numeric	8
OK	1 for valid data, 0 for trash	Numeric	3

2.2.5 Store Data

The table below provides detailed information on the Dominick's stores like city, zone, address, price tier, and zip code

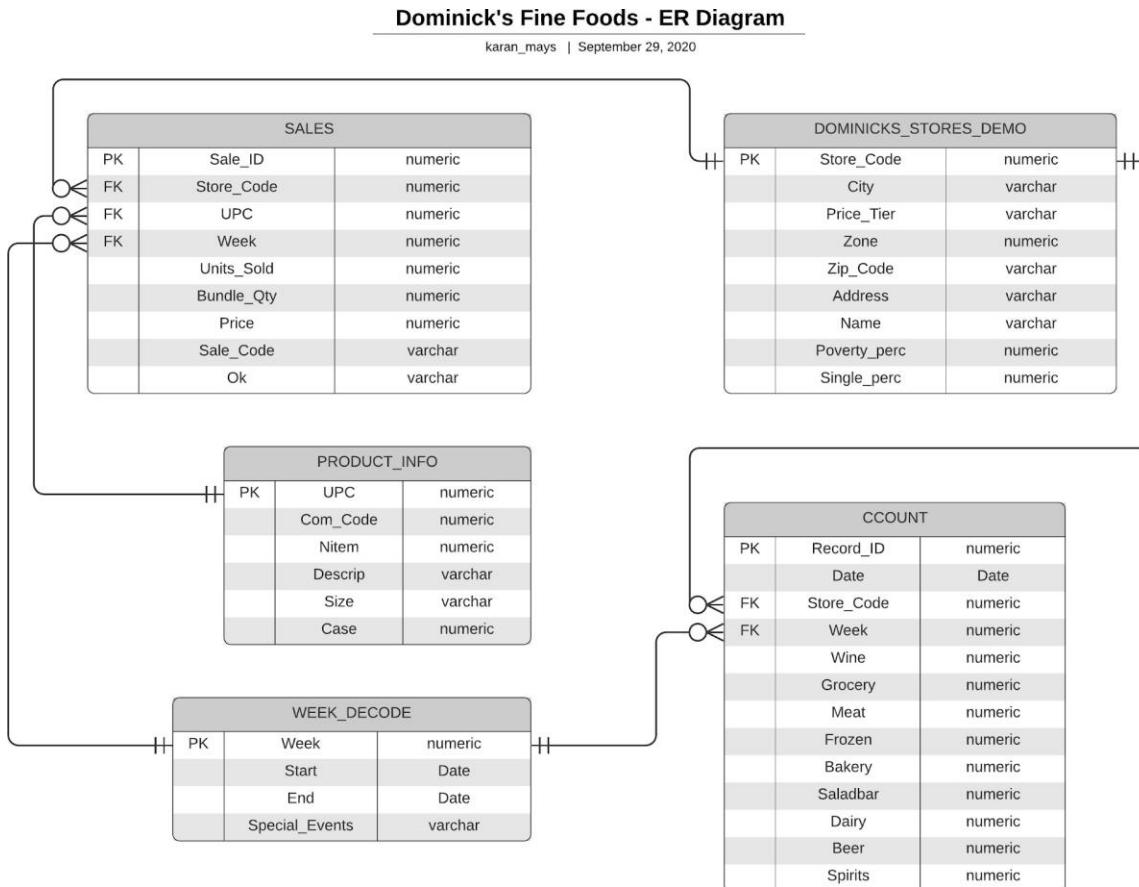
Store	City	Price Tier	Zone	Zip Code	Address
2	River Forest	High	1	60305	7501 W. North Ave.
4	Park Ridge	Medium	2	60068	Closed
5	Palatine	Medium	2	60067	223 Northwest HWY.
8	Oak Lawn	Low	5	60435	8700 S. Cicero Ave.
9	Morton Grove	Medium	2	60053	6931 Dempster
12	Chicago	High	7	60660	6009 N. Broadway Ave.

2.2.6 Week Decode Table

The information on week, its start and end date and holiday events is stated in this file.
The table below provides a description of each of the attributes.

Week #	Start	End	Special Events
1	09/14/89	09/20/89	
2	09/21/89	09/27/89	
3	09/28/89	10/04/89	
4	10/05/89	10/11/89	
5	10/12/89	10/18/89	
6	10/19/89	10/25/89	
7	10/26/89	11/01/89	Halloween
8	11/02/89	11/08/89	
9	11/09/89	11/15/89	
10	11/16/89	11/22/89	
11	11/23/89	11/29/89	Thanksgiving
12	11/30/89	12/06/89	
13	12/07/89	12/13/89	
14	12/14/89	12/20/89	
15	12/21/89	12/27/89	Christmas
16	12/28/89	01/03/90	New-Year
17	01/04/90	01/10/90	
18	01/11/90	01/17/90	
19	01/18/90	01/24/90	
20	01/25/90	01/31/90	
21	02/01/90	02/07/90	
22	02/08/90	02/14/90	
23	02/15/90	02/21/90	Presidents Day
24	02/22/90	02/28/90	

2.3 ERD



1. Sales Records needed for UPC of: Cheese, Bathroom Tissues, Detergent

2. DOMINICKS_STORES_DEMO Table formed by merging DFF Store Price Tier Table (the Price_Tier and Address fields merged on basis of Store_Code) inside the Demographics Table.

3. DOMAIN UNDERSTANDING

In order to understand how to extract meaningful and relevant data from a retail stores dataset, we went through several research papers to develop an understanding of what factors affect the decision-making process and strategies in these retail stores and what kind of business goals are targeted while choosing these factors. In this report we are summarizing our understanding of these strategies from three such domain-specific research papers.

In their research Daniel, et al [1] summarize the effect of Holiday in the increase or decrease in sales of any retail store or chain. Two of the biggest holidays in the United States are the Thanksgiving and Christmas during which every retail chain faces huge sales. In order to maintain customer satisfaction, these stores have to be prepared beforehand to tackle huge customer traffic by stocking inventory and shelves. Along with this we also find that price rigidity is highest during holiday which means the store can't gain profit or loss by changing prices in the holiday season. A hypothesis test using a dataset is performed in the research to prove that demand and supply do not affect the cost of the products during the holiday season and vice versa. Hence a store must analyze the cost of price adjustment based on its resources when there is a high chance of no change in demand.

According to Nevo, et al [2] Coupons have a significant effect in the increase or decrease of sales of any retail chain. This paper discusses the scenarios in which coupons are introduced by a Manufacturer and the effect of it in detail. The paper also posits that giving coupons is better than spending a substantial amount in advertisements. It goes on to explain how the price of commodities whose quality is not comparable to its competitors will decrease in long-term. This is very important in formulating our strategies in later stages of the project for improving sales based on the insights derived from the data aggregated inside the data warehouse.

Nevo, et al also discusses different reasons behind a manufacturer introducing a discount or coupon on a particular product. Firstly, it can be for the inception of a new product in the market. Secondly, coupons are also introduced to boost up sales of a particular product that might be lagging behind in its sales. In addition, the paper also discusses how the perception of customers about a product can change if the coupons are made available continuously. Effects and correlation of price and coupons in different situations are derived and explained mathematically in the entire paper.

Retail industries need to consider the global chain-level impact when they are planning and implementing the price-promotions strategy. In [3], Wagner and Wooseong discussed this impact of price promotions on a specific product across the organization. The authors conducted an analysis to study the effect of price promotions globally and proposed a model to analyze both store-level and cross-category impact of the price promotion on the product.

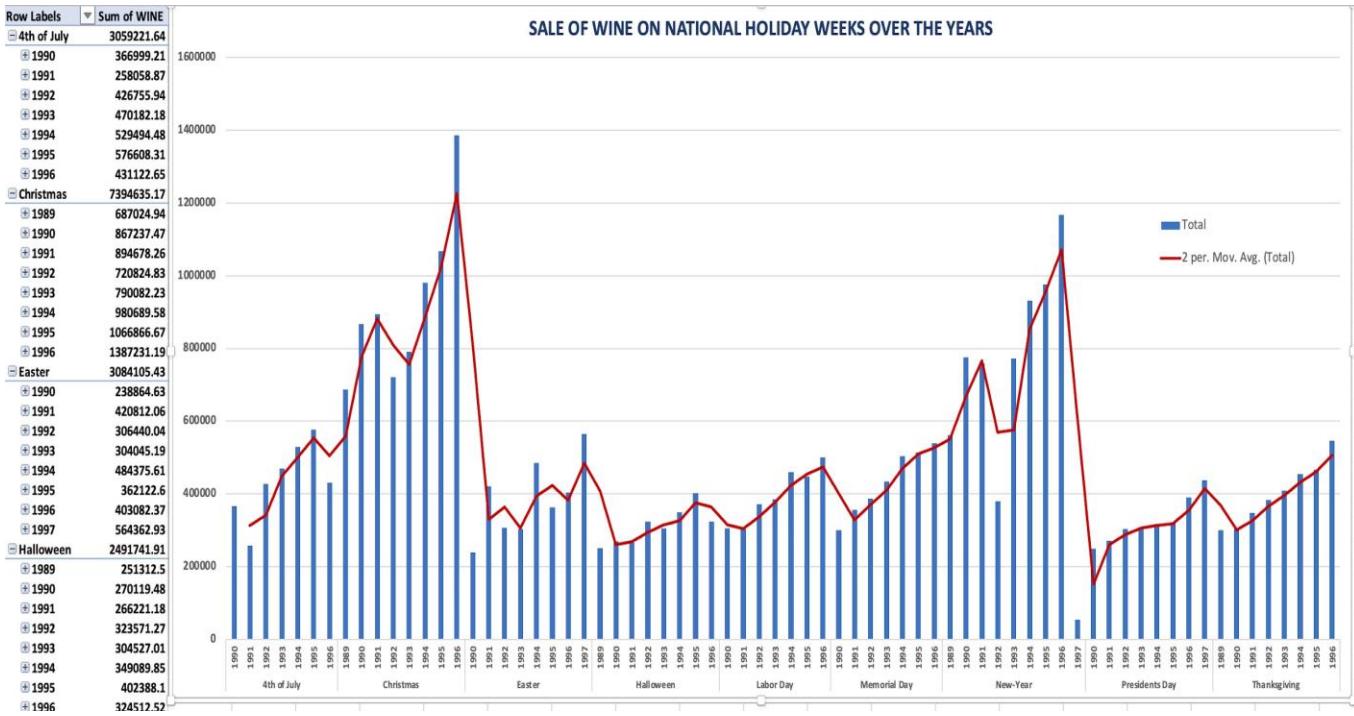
This model was then applied to Dominick's retail chain and compared with several other competing models, concluding to be a better model offering higher flexibility. This research is conducted on the sales and prices of toothbrush and toothpaste across various brands and stores. It was inferred that the discount on toothpastes can have both positive and negative effects on sales of various toothpaste brands. This research illustrates the effect of bonus buy, price reduction and discounts across brands and stores of Dominick's.

In [4], the authors assess the demand for beer treating it as a distinguished product. Various metrics like the own-price, cross-price and income elasticities for different types of beer like craft beer, imported and mass-produced beer are estimated. The analysis is performed using the detailed market scanner database from Dominick's supermarkets in Chicago, consisting of nearly nine years of store level data of more than 700 beer products and consumer attributes. The results showed that beer is treated as a normal commodity and did not significantly impact the consumer choice depending upon the price or type, the least responsive category being the mass-produced beer.

This research has some limitations. As the analysis was based on Chicago area, we cannot generalize and apply the results across all geography, especially nationally. Secondly, the period of this study is limited to a certain timeline following which the craft beer industry continued to expand in the United States. A proper comparison across different time periods is needed to understand how the preferences vary across different types of beer developed over time[4].

4. BUSINESS QUESTIONS

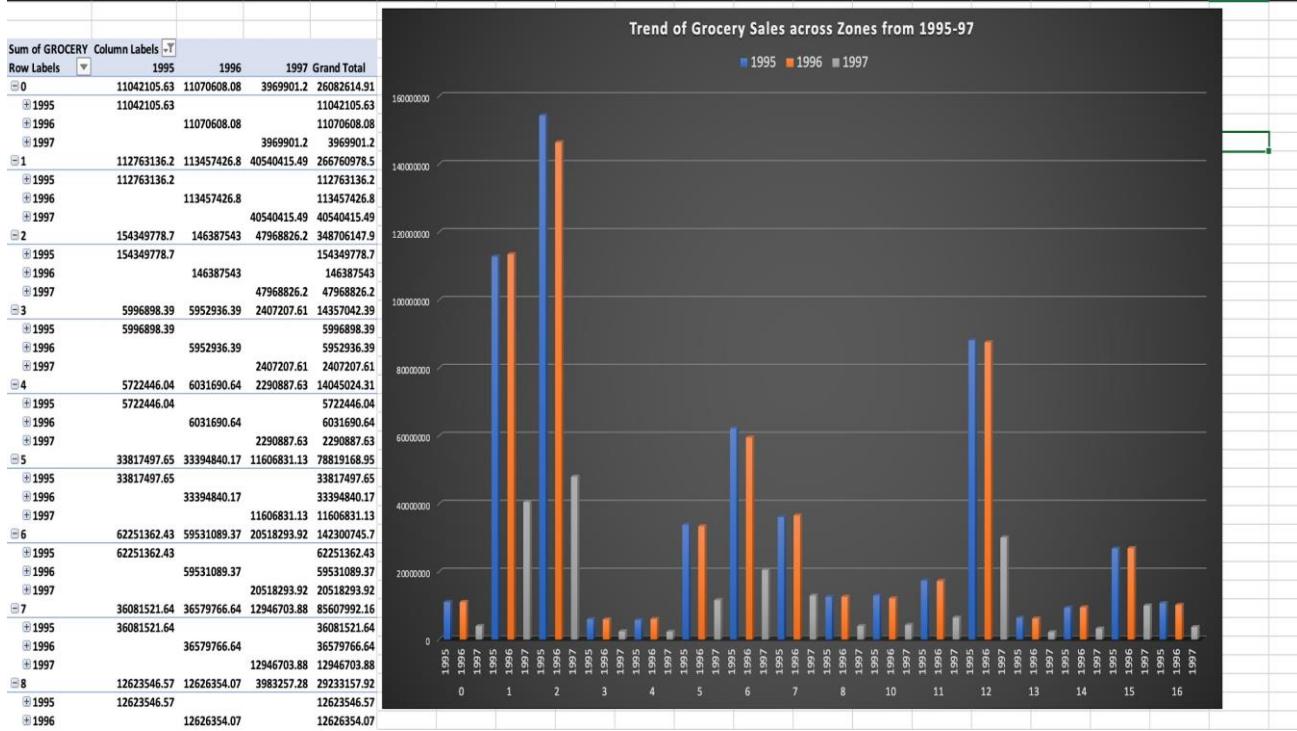
Q1. What is the sale of Wine on National Holiday Weeks over the years?



Justification:

Customers are likely to shop extensively during the holiday season. As a result, certain products are likely to see a significant boost in sales. One such product is Wine, which is often bought for celebrating special occasions, and when spending time with family and friends. Here we can see that the sale of Wine is particularly higher during Christmas and New Year's Eve, followed by Thanksgiving. Analysis of such product sales during special occasions will help DFF identify products that show seasonal growth. Using this information, DFF can plan for these products effectively during the holiday season to cater to the demand. Proper shelf management can also be done ahead of time, with a good estimate of demand. This analysis is primarily important for retail industries to know what the best time is, to market their products and gain popularity among customers. By knowing which products sell the most during holidays, DFF can invest in early marketing to have an edge over their competitors and come up with exciting promotional offers to attract more customers and consequently, maximize profitability.

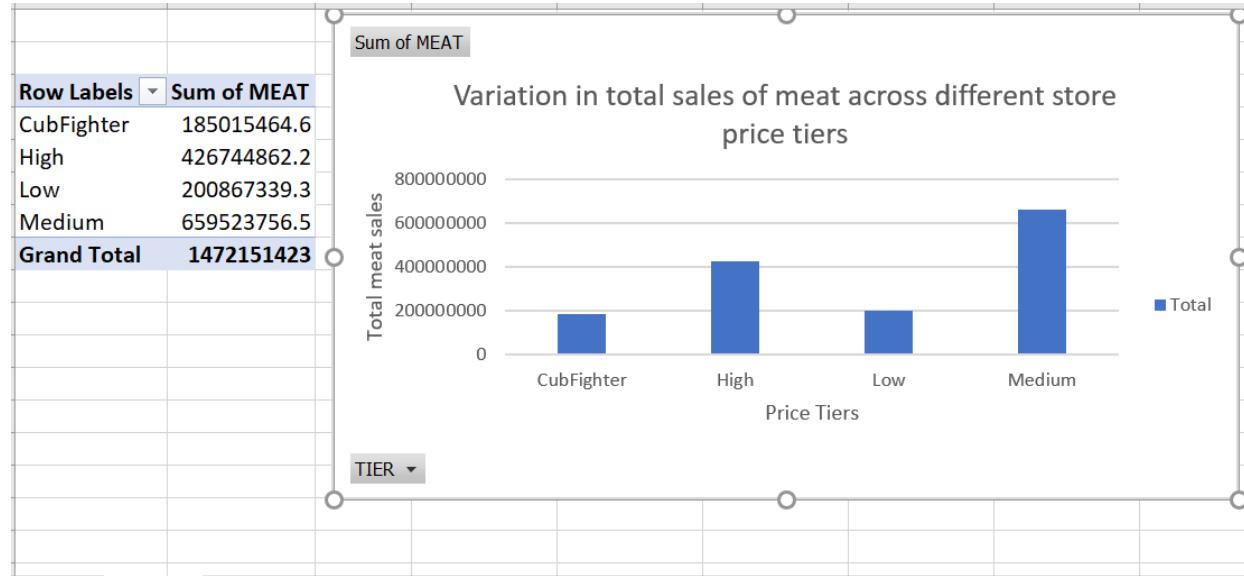
Q2. How are average sales of a product changing by store from 1995 - 1997?



Justification:

The chart above shows the trend of grocery sales across DFF stores in different zones over three years, from 1995 – 1997. Retail industries, like DFF that have multiple stores, require this analysis to understand the buying trend of their customers across multiple regions. Customer demographics is diverse geographically and may greatly impact the buying patterns of the customer. For instance, a store in Florida or Arizona are not likely to see a large number of sales of snow overcoats, however, the situation would be completely opposite in colder areas like Chicago or Michigan. By understanding the trend of product sales across stores, DFF can efficiently perform targeted marketing. Looking at the analysis over the years helps to find a pattern in the product sale. If a product is making the highest sale in a store over a few years, there's a high probability that the trend may continue in future. DFF can then devise customized marketing strategies for the most popular products in every store, zone-wise, to maximize revenues.

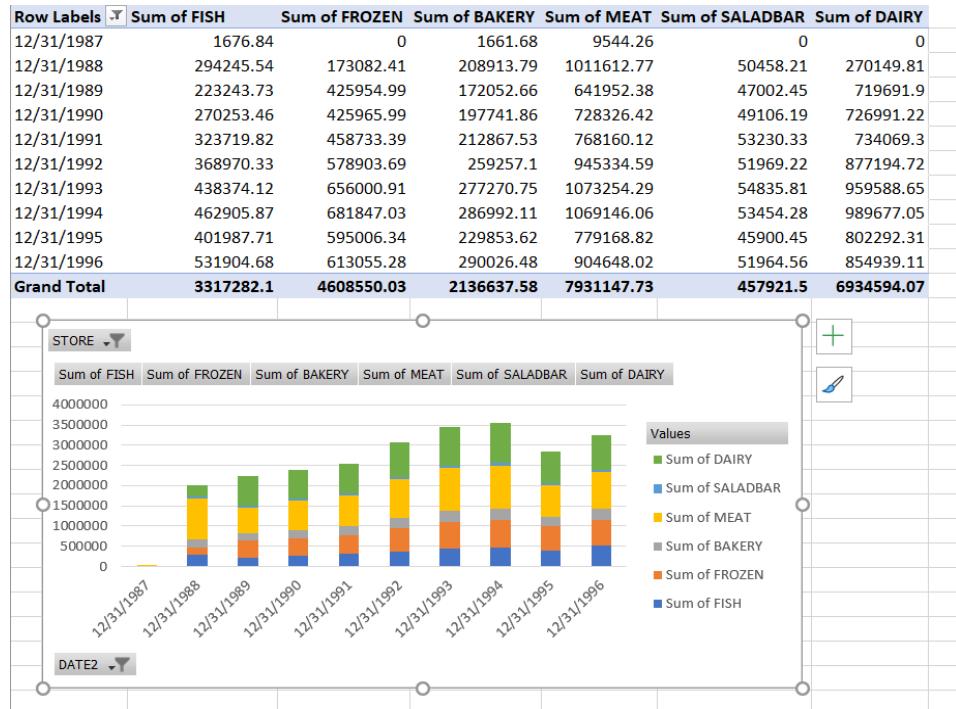
Q3. What is the variation in total meat sales across the different store price tiers?



Justification:

The above plot compares total meat sales across stores falling in the four different price tiers – Low, Medium, High and CubFighter. This analysis provides an insight into the customer demographics and their buying patterns. High sales in 'High' price tier indicates that the stores in this tier might be in rich neighborhoods where people can afford to spend lavishly. Retail industries can utilize this information for understanding demand trends in different stores/tiers and plan their promotional schemes or distribute merchandize based on the price tiers. They can also plan the product distribution based on the demand trend.

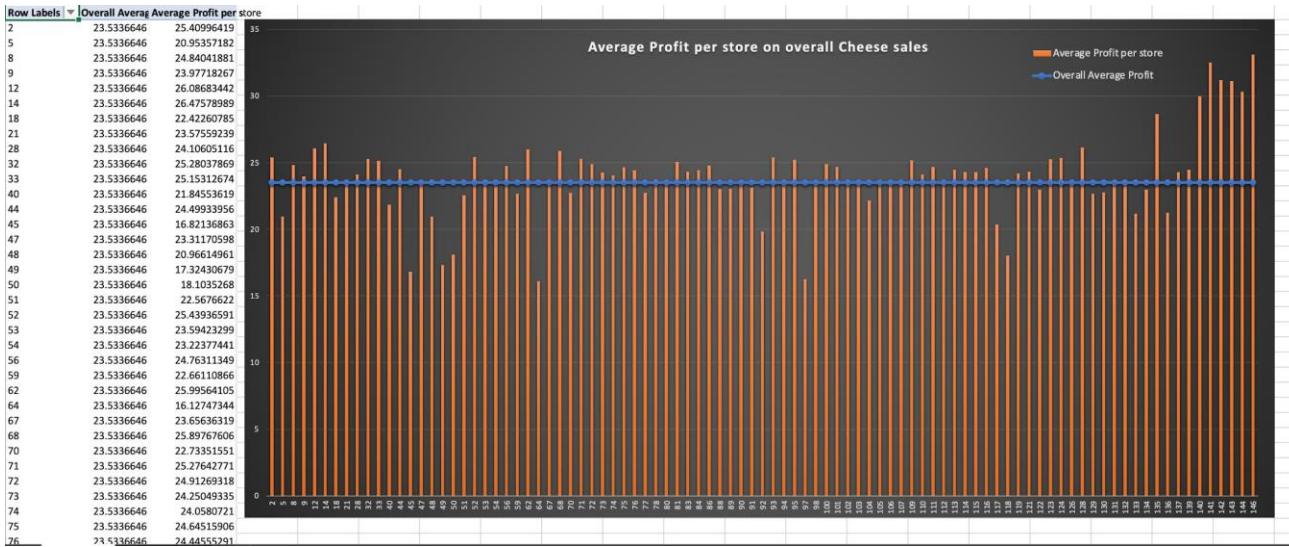
Q4. What are the total sales of various food items on New Year's Eve over the years?



Justification:

During festive occasions like Thanksgiving and Christmas, some products seem to have a boost in sales over others. It is important to analyze this data in a drill-down format to have a detailed view of sales during these seasons. So, the first step would be to have a general idea of product type sold the most in each category like Food, Clothes, Appliances etc. Food is an important category during the holiday season. Such an analysis will help DFF identify all the product categories that see a growth in sales in holiday to manage its supplies efficiently and establish shelf management strategies accordingly during these seasons.

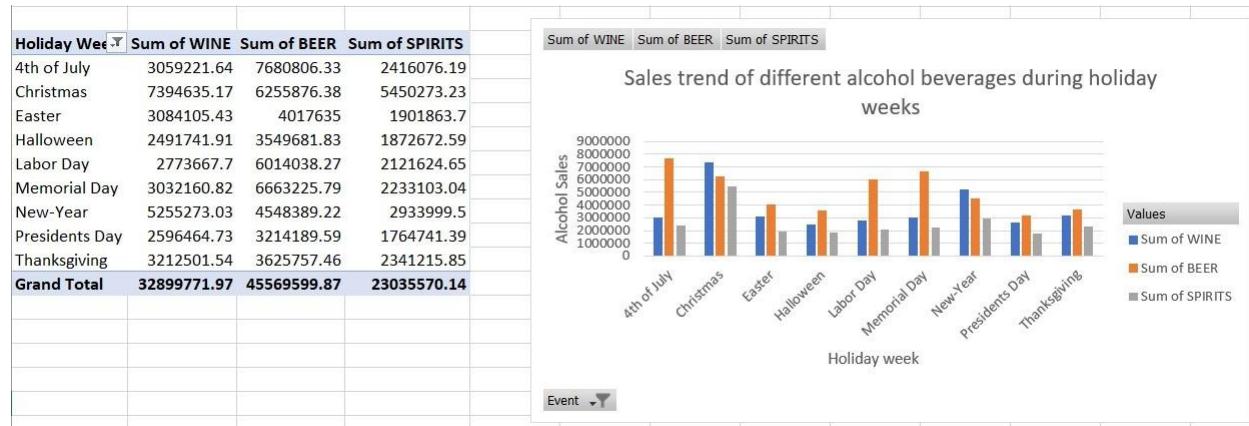
Q5. Which store's average profit, is below the overall average profit in terms of sales of cheese?



Justification:

The above chart shows the average profit per store obtained on the sale of cheese and compares it with the overall average profit obtained from the cheese sales. Analyzing the sale of a product and the product type which sells out the most on a weekly basis can help the retail store enhance its inventory and ensure it is adequately stocked on the days the demand is high. It will also help in identifying which store's sale is below average. The retail industries can then develop better marketing strategies for low profit stores through promotions or introduction of new products.

Q6. What is the Sale of different kinds of alcoholic beverages during holiday weeks?

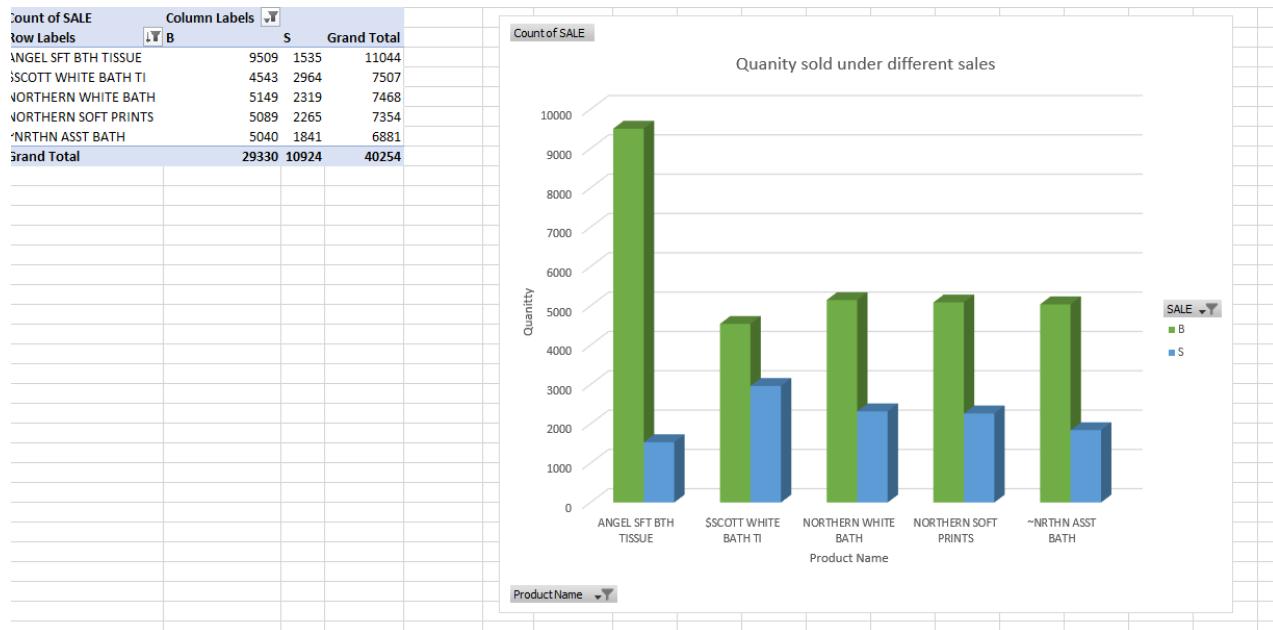


Justification:

The above chart shows the sale of different kinds of alcoholic beverages over holiday weeks. As we can see the highest sales for all three types of alcohol are around Christmas. Such analysis can help the retail industries plan for the products in high demand on special occasions, a little ahead of time to ensure they are able to cater to all their customers' needs and maximize profitability during the time of the year when the sales have good chances of soaring high. Weekly planning and management of products is desired both at store level and shelf level to ensure that all the consumer requirements are met, resulting in higher sales and better profitability.

Note: This question has been changed from – What is the sale of different alcoholic beverages over a week to this, since in the original question, data was required to be available in a daily format, which wasn't available as the date dimension is defined at the grain of the week.

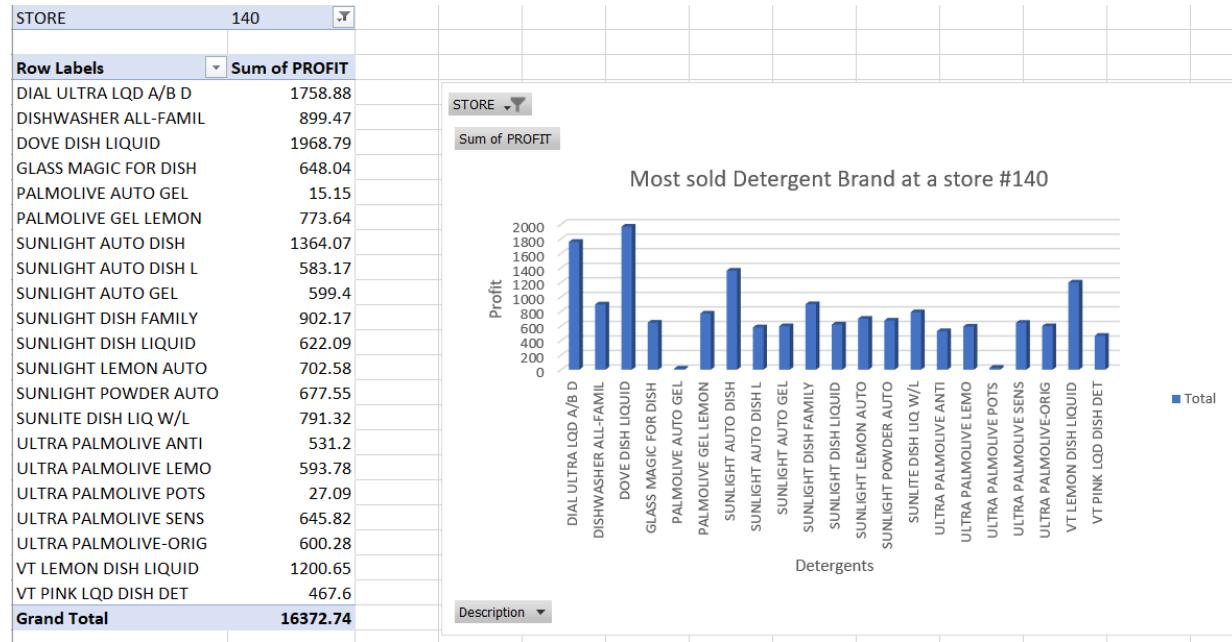
Q7. What is the effect of different cost reduction methods (Bonus Buy/Coupon/Simple price reduction) on the profit from a product (Bathroom Tissues)?



Justification:

Promotion strategies on pricing is one of the strongest marketing strategies in the retail industry. It is a quick way to attract a large consumer base for making high product sales. In the chart above, we have showed the effect of three different cross-reduction methods – bonus buy, coupons and price reduction on the profit of Bathroom tissues. This type of analysis is crucial for testing the impact of the promotional strategy. Retail industries can compare the profits made by each of the strategy to identify which one results in maximum profit or which one gives the lowest results. It also helps in understanding the consumer behavior.

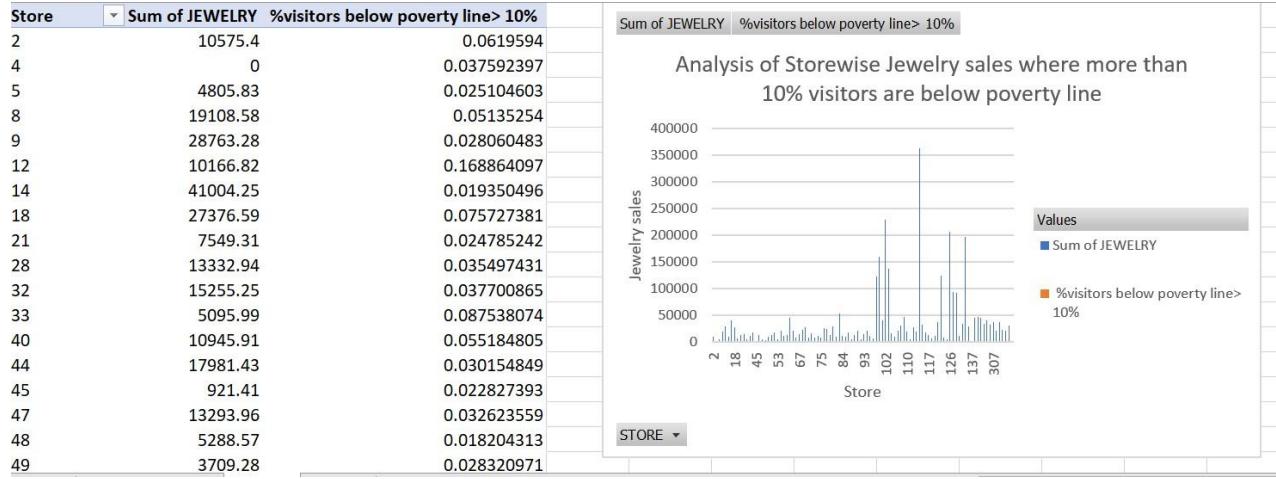
Q8. Which is the most profitable detergent brand at a particular store?



Justification:

There is a fierce competition in the retail industry today. Almost every product is launched by multiple brands, creating a plethora of options for consumers to choose from while adding pressure of accommodating every customer's needs on the retail stores at the same time. In order to sustain the competition, it is very crucial for the retail industry to understand the brand preferences of the consumers. The above chart compares the profits made by multiple detergent brands to identify which one is the most profitable. This analysis can help in ensuring that the demand for the most popular brand is always met by the retail store, resulting in higher customer base.

Q9. Which are the best performing stores for Jewelry sales where more than 10% of visitors are below the poverty line?

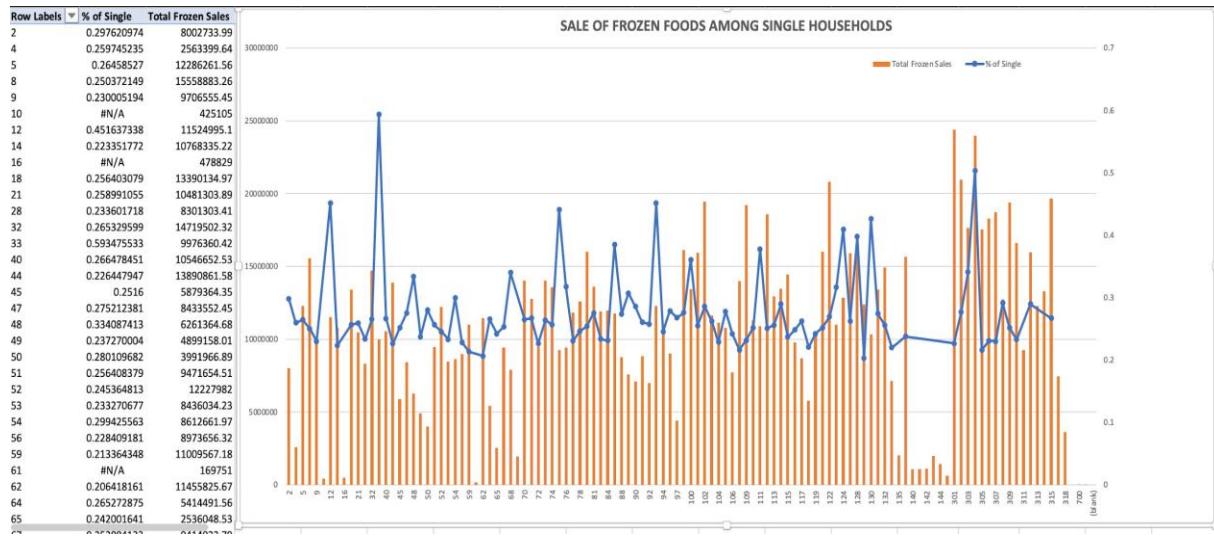


Justification:

The plot above shows the stores that see the best sale of jewelry when the percentage of people below poverty line is above 10% . Through this analysis, retail industries can identify which stores are more popular among low income households for a certain product. This analysis will help in appropriate stock distribution, depending on the purchase patterns of the people coming from varied backgrounds. The marketing strategies can also be customized to attract this class of people by providing them with larger discounts or offers.

Note: This question has been changed from – Which store is most visited by Customer below poverty line in Chicago to this, since in the original question, Fact Table measurement was not required to be tracked.

Q10. What are the sales of frozen food among single households across all the stores?



Justification:

The above graph shows the trend of total sales of frozen food among single households. An analysis of this type helps in understanding peculiarities in the consumer behavior. A common trend observed in day-to-day life can be reflected in the consumer buying pattern.

5. JUSTIFICATION OF SELECTED QUESTIONS

Q1. HOW ARE AVERAGE SALES OF A PRODUCT CHANGING BY STORE FROM 1995 - 1997?

The amount of sales made by a product at a specific store can be obtained from the Product_Sales fact table. The average sales of a product can be plotted across multiple stores for the given time frame, that is from the year 1995 to 1997 using information from the dimProduct, dimStore and dimTime tables. The dimProduct table will contain the product information, dimStore can be used to gather store information and dimTime can be used to analyze the sales at different points of time, yearly in this case.

Q2. WHAT ARE THE TOTAL SALES OF VARIOUS FOOD ITEMS ON NEW YEAR'S EVE OVER THE YEARS?

The amount of sales made by various food items can be obtained using the Product_Sales fact table. As we are interested in analyzing the sales on a special event, like New Year's eve in this case, dimProduct and dimTime are two dimensional tables that will be used for obtaining the product information and record of special events respectively.

Q3. WHAT IS THE SALE OF DIFFERENT KINDS OF ALCOHOLIC BEVERAGES OVER HOLIDAY WEEKS?

Sales of Alcoholic Beverages is available for 3 different Product Categories - Beer, Wine and Spirits. These categories are stored in the dimension - dimProductCategory. We have created this separate dimension for Product Categories, to include those categories for which product (UPC) level sales data is unavailable. In the Ccount dataset we observed many such categories like Grocery, Wine, Spirits etc. for which product level data was unavailable. So, we will be storing such categories in the dimProductCategory and record the sales of these categories in each of DFF's stores, every day from 1987 to 1996 in the fact table - FactCategorySale. This Fact table will use 3 dimensions - dimStore, dimProductCategory and dimTime to record sales at the level of detail of total daily sales for a product category at each of the stores.

Q4. WHICH IS THE MOST PROFITABLE DETERGENT BRAND AT A PARTICULAR STORE?

The most profitable detergent brand can be from product Sales Data Mart. This data mart consists of Fact Table-FactProductSales connected to the dimensions dimStore and dimProduct. The fact table stores data about the stores and different products within these stores along with the profit they make which can help in analyzing profits of different brands of the same product in a given store.

Q5. WHICH ARE THE BEST PERFORMING STORES FOR JEWELRY SALES WHERE MORE THAN 10% OF VISITORS ARE BELOW THE POVERTY LINE?

Most visited stores by customers below poverty line for a certain product can be obtained from Storewise Aggregated Category Sales Data Mart. As we are interested in analysing the customer visit for a demographic type, this data mart consists of dimensions-dimStore and dimProdCategory and the fact table factStorewiseAggCategorySales.

6. INDEPENDENT DATA MARTS DESIGN USING KIMBALL'S APPROACH

We have used Dimensional Modeling technique to create STAR schema for designing the data warehouse for Dominick's Finer Foods retail store chain. Below section of this report, provides details for the various Business dimensions and fact tables, created to answer the business questions pertaining to Dominick's Finer Foods.

A. INFORMATION PACKAGE DIAGRAMS

Product	Store	Time
Product_Key	Store_Key	Date_Key
UPC	Store_No	Year
Product_Name	City	Month
Product_Category	Price_Tier	Week_No
	Zone	Day
		Special_Event
Fact: Bundle_Price, Profit_per_Dollar, No_of_Units_Sold, Bundle_Size, Sales		

Store	ProdCategory	Time
Store_Key	Category_Key	Date_Key
Store_ID	Category	Year
City		Month
Price_Tier		Week_No
Zone		Day
		Special_Event
Fact: Category_Sales		

Store	ProdCategory
Store_Key	Category_Key
Store_ID	Category
City	
Price_Tier	
Zone	
Fact: Storewise Aggregated Category Sales	

7. HIERARCHICAL DIMENSION AND FACT TABLES

7.1 DIMENSION TABLES IN HIERARCHY

A total of five-dimension tables have been created viz. Store dimension, Product dimension, Demographics dimension, ProductCategory dimension and Time dimension. The detailed descriptions for each table are as follows:

1. dimStore

dimStore	
PK	Store_Key
	Store_No
	City
	Zone
	Price_Tier
	Poverty
	Zipcode
	Address

- **Store_Key**- A unique identifier of the store dimension table
- **Store_No**- The assigned number to a particular store
- **Zone**- The zone to which a store belongs
- **City**- The city where the store is located
- **Price_Tier**- The price tier of a store in the DFF retail store chain
- **Poverty**- Percentage of population that falls below the poverty line
- **Address**- Address of the store

2. dimTime

dimTime	
PK	Date_Key
	Year
	Month
	Week_No
	Special_Event

- **Date_Key**- A unique identifier of the time dimension table
- **Year**- The year in which the sales were recorded
- **Month** - The month in which the sales were recorded
- **Week_No**- The week in the year when sales have occurred
- **Special_Event**- The identifier to indicate the occurrence of a special event in the year

3. dimProduct

dimProduct	
PK	Product_Key
	UPC
	Product_Name
	Product_Category

- **Product_Key** – Unique identifier of the product dimension table
- **UPC** - Denotes the UPC code for each product
- **Product_Name** – It is mapped to Product Description in the UPC table
- **Product_Category** - It denotes the category under which a set of products fall

4. dimProductCategory

dimProductCategory	
PK	Category_Key
	Category

- **Category_Key** – Unique identifier of the product category
- **Category**– Various product categories present within Dominick's data (includes those categories for which, Product (UPC) level data is unavailable.

7.2 FACT TABLES

1. **FactProductSales:** This table contains the quantitative and aggregated information for the sales and profits of the various products available in the Dominick's Finer Foods retail store chain.

FactProductSales	
PK, FK	Date_Key
PK, FK	Product_Key
PK, FK	Store_Key
	No_of_Units_Sold
	Bundle Size
	Bundle_Price
	Sales
	Profit

Fact Table Keys:

- **Store_Key** - A unique identifier for time dimension which has been generated as a surrogate key and implemented here as a Foreign Key.
- **Date_Key** – A unique identifier for time dimension which has been generated as a surrogate key.
- **Product_Key** – A unique identifier for products, which has been generated as a surrogate key.

Fact Table Measures:

- **Bundle_Price** - This attribute denotes the bundle price of a product.
 - **Profit_per_Dollar** - This attribute denotes the profit per dollar sale of the product.
 - **Bundle_size** - This attribute denotes the bundle size.
 - **Sales** – This attribute denotes the dollar value of sales for a product.
2. **FactCategorySales:** This table contains the quantitative and aggregated information for the sales pertaining to various Product Categories in the Dominick's Finer Foods retail store chain. This Fact table only records sales for those product categories that don't have an associated UPC level sales data.

FactCategorySales	
PK, FK	Date_Key
PK, FK	Category_Key
PK, FK	Store_Key
	Sales

Fact Table Keys:

- **Store_Key** – A unique identifier for stores, which has been generated as a surrogate key and helps in dividing data by store.
- **Category_Key** – Unique identifier of the product category.
- **Date_Key** – A unique identifier for time dimension which has been generated as a surrogate key.

Fact Table Measures:

Category_Sales - This attribute denotes the dollar value of sales for a particular product category.

3. **FactStorewiseAggCategorySales:** This table contains the aggregated information for the sales pertaining to various product brands in a particular product category at a particular store in the Dominick's Finer Foods retail store chain.

FactStorewiseAggCategorySales	
PK, FK	Category_Key
PK, FK	Store_Key
	Sales

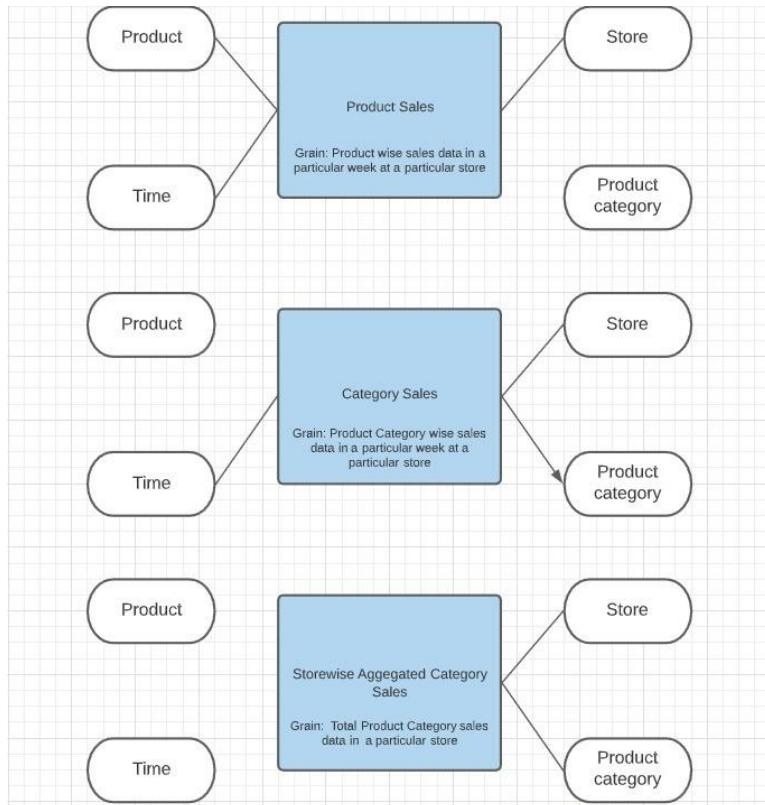
Fact Table Keys:

- **Category_Key** – Unique identifier of the product category.
- **Store_Key** – A unique identifier for stores, which has been generated as a surrogate key and helps in dividing data by store.

Fact Table Measures:

Sales - This attribute denotes the dollar value of sales for a particular product brand.

7.3 FACT AND DIMENSIONS

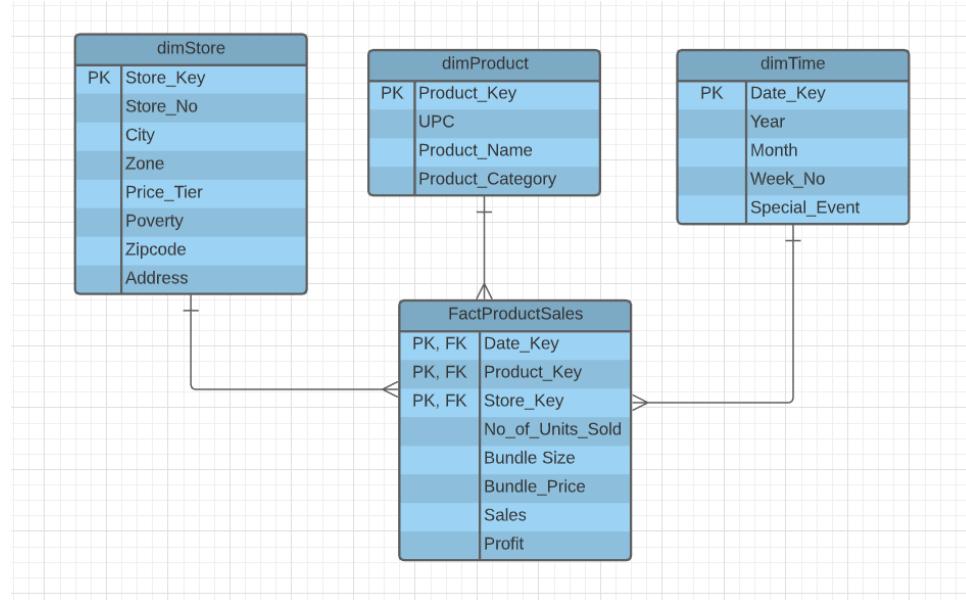


8. DIMENSIONAL MODELLING (STAR SCHEMA)

Using the above fact tables and dimension tables, we have integrated them to create three Star Schemas. These are Product Sales, Customer Demographics and Product Category Sales. The data marts created using these tables are sufficient to answer the selected business questions.

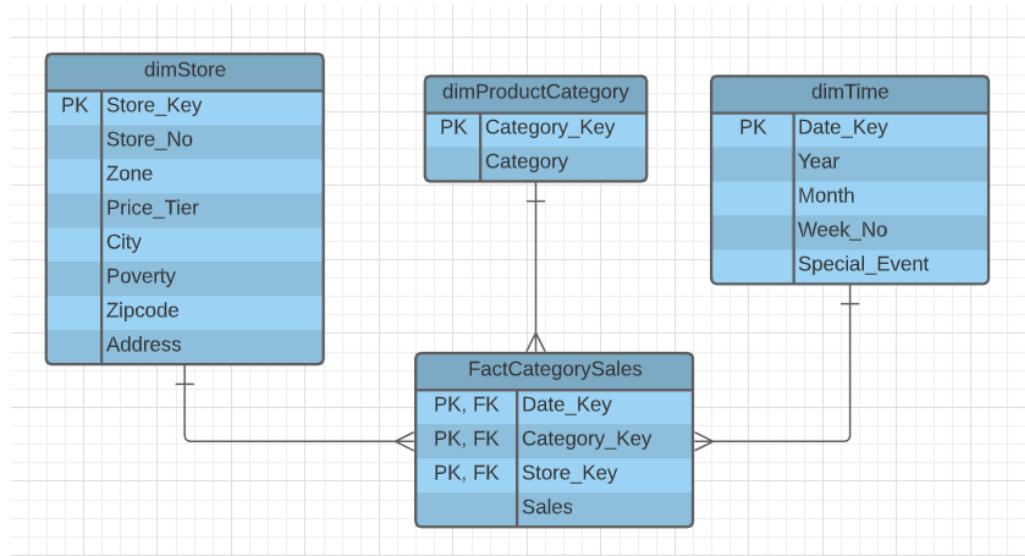
8.1. PRODUCT SALES DATA MART

The Product Sales data mart comprises the Product Sales fact table i.e. FactProductSales and three-dimension tables viz. dimProduct, dimTime and dimStore.



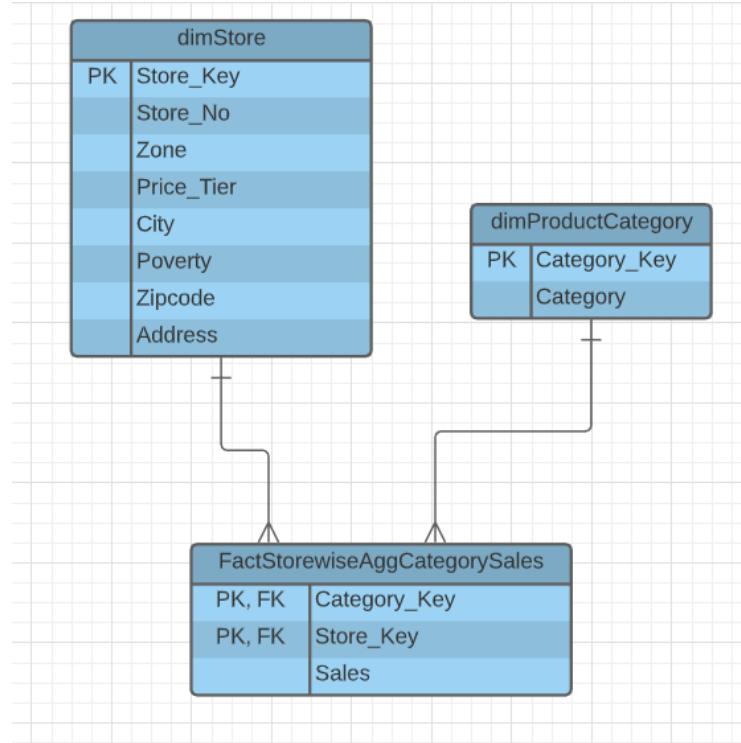
8.2. PRODUCT CATEGORY SALES DATA MART

The Product Sales data mart comprises of the Customer Demographics fact table i.e. FactDemographics and two dimension tables viz. dimDemographics and dimStore. This data mart stores sales data for those categories of DFF for which UPC level sales data is unavailable.



8.3. STOREWISE AGGREGATED CATEGORY SALES DATA MART

The Storewise Aggregated Category Sales data mart, is a data mart that comprises Storewise Aggregated Category fact table, and two dimension tables viz. dimProductCategory, and dimStore. In this, Sales data for a particular category for a particular store is saved. The fact table in this, FactStorewiseAggCategorySales is an aggregated Fact Table in which sales data, aggregated across all the dates is stored.



9. MAPPING TABLES

Data_Source	Source_Attribute	Warehouse Table	Warehouse Attribute	Mapping Function
UPC		dimProduct		
			Product_Key	Surrogate Key for linking each Product in the fact table; Auto-generated
	UPC		UPC	Source: Product_Description => Warehouse_Attribute: Product_Name
	Product_Description		Product_Name	Source: UPC => Warehouse_Attribute: UPC
	Product_Category		Product_Category	
Ccount		dimProdCategory		
			Category_Key	
	Category		Category	
DFF Store Price Tier and Demographics		dimStore		
			Store_Key	Surrogate Key for linking each Store in the fact table; Auto-generated
	Store		Store_No	
	City		City	
	PriceTier		Price_Tier	
	Zone		Zone	
	Poverty		Poverty	
	Zip_Code		Zip_Code	
	Address		Address	

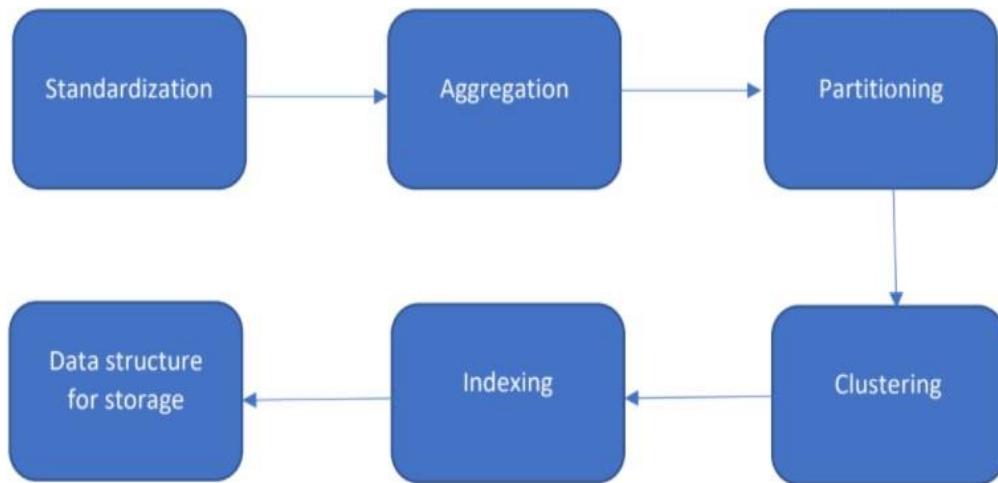
Week Decode		dimTime		
			Date_Key	Surrogate Key for linking each Date Value in the fact table; Auto-generated
	Year		Year	
	Month		Month	
	Week_No		Week_No	
	Special_Event		Special_Event	
Movement		FactProductSales		
			Date_Key	Foreign Key referring to the Surrogate Key - Date_Key in the Date Dimension
			Product_Key	Foreign Key referring to the Surrogate Key - Product_Key in the Product Dimension
			Store_Key	Foreign Key referring to the Surrogate Key - Store_Key in the Store Dimension
	Price		Bundle_Price	
	Move		No_of_Units_Sold	
	Qty		Bundle_Size	
			Sales	Derived Column; Sales = Bundle_Price*No_of_Units_Sold/ Bundle_Size
			Profit	Derived Column; Profit = Profit_Per_Dollar * Sales/100
Ccount		FactStorewiseAgg CategorySales		
			Store_Key	Foreign Key referring to the Surrogate Key - Store_Key in the Store Dimension

			Category_Key	Foreign Key referring to the Surrogate Key – Category_Key in the ProductCategory Dimension
			Sales	
Ccount		FactCategorySales		
			Date_Key	Foreign Key referring to the Surrogate Key - Date_Key in the Date Dimension
			Store_Key	Foreign Key referring to the Surrogate Key - Store_Key in the Store Dimension
			Category_Key	Foreign Key referring to the Surrogate Key – Category_Key in the ProductCategory Dimension
	Sales		Sales	

10. PHYSICAL DESIGN PLAN

The key objective of building the data warehouse physical design plan is to ensure that the physical model fits well with the system-integrated underlying computing platform (hardware + operating system), database, and third-party software. This seems simple, but it can be overwhelming, particularly because when building the physical design plan, there are several perspectives to be considered. Therefore, for the data warehouse to be successful both in terms of physical storage and efficiency, it is extremely necessary to consider the various perspectives that impact the physical design.

Steps of Physical Design Plan



10.1 STANDARDIZATION

In the data warehouse, standardization is important not only to overcome uncertainty in naming database objects but also to ensure consistency across the warehouse in terms of the results returned from the queries against it. In particular, when the same data warehouse is queried for results by various departments within the organization, this becomes more of a requirement.

For instance, to formulate and run their queries in a way similar to the Sales Department, the IT Department can use the same set of conventions. Our team has collectively come up with naming conventions for all databases, tables, and attributes to ensure standardization in the data warehouse. To ensure that the data naming is clear, we have agreed to name the tables as 'Dim' concatenated with the table name, e.g. DimProduct, and attributes as 'tableName attributeName'.

10.2. AGGREGATION

Most of the queries obtained from the data warehouse is summarized information. Every time a query is performed, if we store data at the lowest atomic level, it must aggregate data from multiple records, which contributes to efficiency and performance depravity. Therefore, for quicker query execution and retrieval of data, it would be a good choice to store some summarized information in tables.

For e.g. In our data warehouse, we need to know the product in a product category, which made the highest sales. If we have detailed records keeping sales by quantity and price for a product, in order to retrieve the results, it has to process a large number of records from different categories and fetch the data. To avoid this, we have created summary attributes in the fact table e.g. Sales attributes in the fact table FactProductSales keeps total sales of different products in a category.

10.3. PARTITIONING

Partitioning involves dividing large tables into smaller and more manageable parts to ensure faster data retrieval. If the data warehouse is considerably large, we might have to process millions of rows before getting the result. Also loading of large database tables, backing them up and recovering the data is difficult because of the huge volume of data involved. To ensure faster data retrieval, partitioning requires splitting large tables into smaller and more manageable pieces. If the data warehouse is substantially huge, we can have to process millions of rows before the result is obtained. It is also difficult to load massive database tables, back them up, and retrieve the data due to the vast amount of data involved. An easy and more efficient way to solve this is to divide the tables into smaller tables and then search for the appropriate tables to get the response you need.

For e.g. In our data warehouse, we have FactProductSales, FactDemo, FactCategory fact table and DimProduct, DimStore, DimTime and DimDemographic dimension tables. The DimTime table can be partitioned vertically to store the partitions as the number of queries retrieved by week would be frequently run against the database.

10.4. CLUSTERING

Often, in the data warehouse, putting similar data together in the same block or same file in the storage greatly improves the data warehouse's efficiency. Because of the handling of relevant data, this is attributable to the reduction in the number of accesses and faster processing attributable to simpler recovery. This process of grouping similar data together in a single operation for fast retrieval is called clustering. For the data to be clustered, we first need to find the related data and the corresponding tables.

10.5. INDEXING

Since the data warehouse relies entirely on how effectively the queries are conducted and the results are collected, it is extremely necessary to concentrate on minimizing the amount of time spent on extracting data from millions of database records. To enhance performance, one way to ensure this is to incorporate good indexing techniques in physical design. This is one of the physical design's most significant moves. The indexing method includes deciding on the columns to be indexed, the order in which they need to be indexed, and also whether the bitmapped indexes qualify for any of the columns. For data warehouses, B-Tree indexes and bitmapped indexes are most fitting.

In our data warehouse, we can create unique B-Tree indexes on the Product_Key, Store_No, Time_Key, etc. because they are single-column primary keys. We can also create a B-Tree index for the full primary key in the fact table.

10.6. DATA STRUCTURE FOR STORAGE

After deciding the strategies of aggregating, partitioning, clustering, and indexing, we will concentrate on how to physically store the data on the storage unit, how to break the files into different blocks to ensure maximum performance. To store the data for successful data retrieval, consideration must be taken when selecting the data structures. In the physical design process, preparing and implementing these various techniques not only helps to boost the data warehouse performance, but also decreases the number of accesses and record traversal, thereby increasing the productivity of the entire system.

11. DATA QUALITY ISSUES

In the data provided by DFF, there are myriad quality issues. Issues in Data Quality lead to inconsistent data design, which then fails to meet the requirements posited by users. It is one of the biggest reasons behind failure of a Data Warehouse project. Hence, we must enumerate these issues in a great detail and prepare a plan to fix/circumvent through these issues. Major issues identified in the DFF data are listed below:

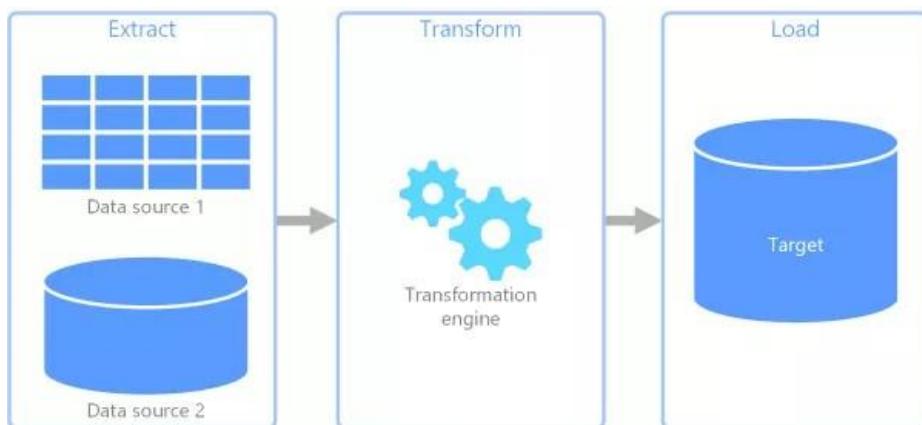
1. Dummy values in fields – In many of the Ccount records, entries for a large proportion of dates was blank or filled with a dummy value. Such values if added in the warehouse can cause grave confusion amongst stakeholders.
2. Absence of data values – A very large number of records in Ccount as well as UPC have blank values. Such intermediate blanks in continuous records can cause issues in aggregation and interpretative analysis.
3. Cryptic values – Many Product Names in UPC, contain cryptic, in-intuitive values such as specific special characters. These must be normalized before they are loaded in the data warehouse.
4. Unofficial use of fields – Some fields like Profit, Sales, Bundle Price, Bundle Size are named in a very unconventional way in the DFF Data. These fields must be appropriately named so that their meaning is understandable at even a glance.

12. ETL DEVELOPMENT PLAN

During data warehouse building, ETL is one of the most essential processes. ETL stands for Extract, Transform and Load stands. Each of these processes has its very own definition. This includes large operations from the extraction of data followed by the conversion of data to the desired format. Finally, in dimensional and fact tables, the data is loaded into the data warehouse. To carry out ETL operations, there are no fixed predefined steps. All activities are carried out based on the available data structure and the data warehouse's end purpose, depending on business requirements. Extraction is the method by which data is extracted from one or more sources. The data is obtained via the extraction, which can also be of various formats. For the final loading point, further transformation is applied to the data to make it more meaningful. Depending on the needs of the data warehouse, data is cleaned and transferred from one format to another. The generation of surrogate keys and various measured columns can be examples of transformation. The data is eventually loaded into the dimensional and fact tables.

Our ETL Plan consists of first extracting the data from the myriad files provided by DFF, into corresponding temp tables in the Staging Area. During this extraction process, we are not cleaning the data. The only rule during Extraction is to move the data as is from the source files to the temp tables. We will only be extracting the required fields from the source files. The second step in our ETL Plan is to clean and transform the data extracted into the temp tables, and then store the clean, transformed data into a final set of Staging Tables. This way, we ensure that data is available in a ready format in the Staging Area itself so that while loading the data into the warehouse, we would only require to transform the data to suit the model used in the warehouse.

The third step is where we define packages in SSIS to load the data from the final staging tables to the corresponding Dimension and Fact tables in the Data Warehouse. Here we would aggregate the data, create surrogate keys and store the various fields separately into dimensions and fact tables depending upon whether the field is a descriptive property of some dimension, or a measure.



12.1 EXTRACTION PLAN

The process of data extraction from the source files is the initial step involved in the ETL process of designing and implementing a data warehouse. Data extraction is the process of retrieving data out of data sources for further data processing. Proper methodology to perform extraction is essential to build an efficient data warehouse. The extraction of data involves combining all the data sources to single format tables in Microsoft SQL Server Studio, that can be later used for transformation and loading. The data from the sources are in different formats - Comma Separated Value (.csv), Excel Files (.xls) as well as Text Files (.txt). The data for Week and Store details was extracted from the data manual for Dominick Finer Foods, and .xls files were created for such tabular data.

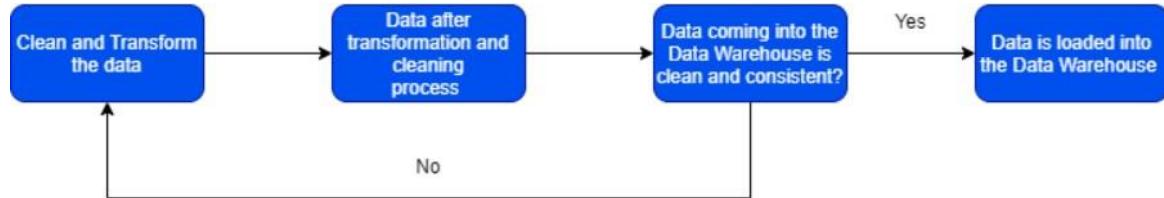
12.2 DATA TRANSFORMATION

Data extraction is followed by data transformation and cleaning process. Extracted data from the staging area is the input to the transformation process. Strict data cleansing rules have been applied to this data to create a data that is clean and consistent throughout. Clean data is then loaded into the DW area so as to create data marts and get the correct results. The transformation and cleaning rules applied to the data are the following

- **NULL value removal:** Null values existing in the data and those which were created during extraction process will be deleted
- **Data conversion:** The data extracted from source files are stored as attributes of type varchar, nvarchar etc. in the staging area. These have been converted into their respective data types such as int, decimal and date. The date was available in a format which wasn't understandable. Using functionalities in SSIS, we converted the date into standard format for user understanding.
- **Dirty data removal:** Attributes unrelated to the business questions are removed. For example, in the Ccount file apart from Frozen, Meat, Bakery, Cheese, Beer, Spirits, Wine, Dairy, Jewelry, Custcount, Week, Store Number everything else is removed. Blank records, rows with just dots and other meaningless values as negatives are eliminated.
- **Derived attributes:** Derived attributes were executed through the year. We derived month, year and day from date attributes respectively. We have also derived Sales and Profit columns for using the existing columns of the UPC table.
- **Creation of surrogate keys:** We have created surrogate keys for all dimension and fact tables before data is loaded.

12.3 DATA LOADING

The following steps were taken for Data Loading:



13. DATA SOURCES AND TARGET DATA

13.1 TARGET DATA NEEDED IN THE WAREHOUSE

Data Source	Source File Name	Staging Area Tables	Warehouse Tables
Demographics	Demographics.csv	Demographics_tmp, Store	
Price Tier	DFF Store Price Tier File.xls	PriceTier_tmp, Store	dimStore
Ccount	CCOUNT - COPY.csv	Ccount_tmp, Customer_Count, Category_Count	dimProductCategory, FactCategorySales, FactStorewiseAggCategorySales
UPC	UPCLND.csv	UPC_tmp, UPC	dimProduct
Movement	wlnd.csv	Movement_tmp, Movement	FactProductSales
Week Decode	Week Decode.txt	WeekDecode_tmp, WeekDecode	dimTime

13.2 DETERMINING DATA SOURCES

The data sources for the data warehouse is provided by the University of Chicago Booth School of Business. The files which are required for the business questions for this project are as follows:

DATA	SOURCE FILE
MOVEMENT	WLND.csv
CUSTOMER COUNT	Ccount.csv
DEMOGRAPHICS	Demo.csv
TIME	WeekDecode.xlsx
UPC	UPCLND.csv
PRICE TIER	DFFStorePriceTire.xlsx

13.3 DATA MAPPINGS FOR DATA ELEMENTS FROM SOURCES TO STAGING AND STAGING TO DATA WAREHOUSE

13.3.1 Source Data to Staging Temp Tables

Source	Source Attributes	Staging Temp Table	Staging Temp Table Attributes
DEMO.csv	STORE	Demographics_tmp	Store_No
	CITY		City
	ZONE		Zone
	POVERTY		Poverty
DFF Store Price Tier File.xls	Store	PriceTier_tmp	Store
	City		City
	Price Tier		Price Tier
	Zone		Zone
	Zip Code		Zip Code
	Address		Address
CCOUNT - COPY.csv	STORE	Ccount_tmp	STORE
	DATE		DATE
	WEEK		WEEK
	DAIRY		DAIRY
	FROZEN		FROZEN
	MEAT		MEAT
	CHEESE		CHEESE
	BEER		BEER
	WINE		WINE
	SPIRITS		SPIRITS
	BAKERY		BAKERY
	JEWELRY		JEWELRY
UPCLND.csv	UPC	UPC_tmp	UPC
	COM_CODE		COM_CODE

	DESCRIP		DESCRIP
	SIZE		SIZE
	CASE		CASE
	NITEM		NITEM
WLND.csv	STORE	Movement_tmp	STORE
	UPC		UPC
	WEEK		WEEK
	MOVE		MOVE
	CITY		CITY
	PRICE		PRICE
	SALE		SALE
	PROFIT		PROFIT
	OK		OK
Week Decode.txt	Week_number	WeekDecode_tmp	Week_Number
	Start		Start
	End		End
	Event_name		Event_name

13.3.2 Staging Final Tables to Warehouse Tables

Staging Area Table	Staging Area Table Attributes	Warehouse Table	Warehouse Table Attributes	Mapping Function
Store		dimStore	Store_Key	An auto-incremental Surrogate Key generated to be used as the Primary Key
	Store_No		Store_No	
	City		City	
	Zone		Zone	
	Price_Tier		Price_Tier	
	Poverty		Poverty	
	Zip_Code		Zip_Code	
	Address		Address	
WeekDecode		dimTime	Date_Key	An auto-incremental Surrogate Key generated to be used as the Primary Key
	Year		Year	
	Month		Month	
	Week_Number		Week_No	
	Special_Event		Special_Event	
UPC		dimProduct	Product_Key	An auto-incremental Surrogate Key generated to be used as the Primary Key
	UPC		UPC	
	Product_Name		Product_Name	
	Product_Category		Product_Category	

Category_Count		dimProductCategory	Category_Key	An auto-incremental Surrogate Key generated to be used as the Primary Key
			Category	
Movement		FactProductSales	Date_Key	Foreign Key referring the Surrogate Key Date_Key in the dimension - dimDate
			Product_Key	Foreign Key referring the Surrogate Key Product_Key in the dimension - dimProduct
			Store_Key	Foreign Key referring the Surrogate Key Store_Key in the dimension - dimStore
	No_of_Units_Sold		No_of_Units_Sold	
	Bundle_Size		Bundle_Size	
	Bundle_Price		Bundle_Price	
	Sales		Sales	
	Profit		Profit	
Category_Count		FactCategorySales	Date_Key	Foreign Key referring the Surrogate Key Date_Key in the dimension - dimDate
			Category_Key	Foreign Key referring the Surrogate Key Category_Key in the dimension -

				dimProductCategory
			Store_Key	Foreign Key referring the Surrogate Key Store_Key in the dimension - dimStore
	Sales		Sales	
			Category_Key	Foreign Key referring the Surrogate Key Category_Key in the dimension - dimProductCategory
Category_Count		FactStorewiseAggCategorySales	Store_Key	Foreign Key referring the Surrogate Key Store_Key in the dimension - dimStore
			Sales	(Sales recorded by grouping the sales from Category_Count by Store, Category)
	Sales			

14 DATA EXTRACTION RULES

Data extraction is the first and foremost step in ETL and is crucial for efficiently integrating data from multiple sources or applications. Ideally, when a data warehouse is built, the data is gathered from numerous data sources but for the scope of this project, we have collected entire data from Chicago booth data set. Source data may be present in different formats on different source machines. The source systems are generally OLTP systems that monitor the day-to-day transactions of a business. The extraction of data should be performed with caution to ensure that the source systems are not impacted during the data extraction process.

In this project, we extracted data from 3 types of file sources:

- a. Comma separated value (csv) files: These include - DEMO.csv, CCOUNT-copy.csv, UPCLND.csv, wlnd.csv
- b. Excel Files - DFF Store Price Tier File.xls
- c. Text Files - Week Decode.txt. This data was scraped from DFF's manual in a text (tab-delimited) format.

The goal of data extraction is to get all of the data from various sources and combine it in a single format. Here, we are gathering data from csv files into different tables in the Microsoft SQL Server Management Studio, where the data will be further transformed. The data from these files will be initially stored in the temp tables of the Staging Area.

Below rules are followed during the extraction step:

- a. Data from source files will initially be stored in corresponding temp tables in the Staging area. During this initial extraction, data would be stored as is so as to have a copy of the original source data handy in the Database for ease of transformation
- b. Surrogate Keys created after extracting data in the warehouse, will follow the naming convention - '<dimension_name>_Key'

15 DATA TRANSFORMATION AND CLEANSING RULES

The data extraction process is followed by the data transformation process, which is an equally important step in the ETL process. The way the data will be transformed in this step will determine its ability to answer the business questions, hence standardizing and converting it into a suitable format is very crucial for the efficient implementation of the data warehouse.

Data transformation essentially deals with converting the data in a suitable format. Majority of data transformation tasks include data cleansing. Data cleansing is the process of improving the data quality by performing certain cleaning tasks like removing the blank values, null values, correcting spelling errors, data entry errors and duplicates. These tasks are important for ensuring that the data is consistent and ready for modeling of the data warehouse.

Data transformation also includes generating or calculating new columns. These are the calculated columns and are not present in the source or raw data. An example of a new column could be the surrogate key column. A surrogate key which is an auto-incremented key is used as a primary key for dimensional tables in the data warehouse. Apart from these, there are also derived columns. Example of derived attributes is Profit or Sales columns in the FactProductSales fact table.

The general transformation and cleansing rules that are applied are as follows:

15.1 REMOVAL OF UNWANTED DATA

Attributes that are not part of the answers to the business questions are removed. For example, in the Ccount file apart from Frozen, Meat, Bakery, Cheese, Beer, Spirits, Wine, Dairy, Jewelry, Custcount, Week, Store Number everything else is removed.

15.2 REMOVAL OF INCONSISTENT DATA/SPECIAL CHARACTERS

Some of the values in some of the fields like DATE in Ccount and Product_Name in the UPC files contain inconsistent values like special characters or illegible expressions. Such fields are removed during cleansing and transformation.

15.3 REMOVAL OF NULL VALUES

Null values as well as blank records that were existing in the data and that were created while extracting data into tables will be deleted.

15.4 DATA CONVERSION

The data type of all fields extracted from the source files are in varchar() or nvarchar(). Thus, the temp tables also contain the fields in those formats. While/before loading these fields in the final staging tables, they have been converted into their respective data types such as int, float and date to maintain data integrity.

15.5 CREATION OF SURROGATE KEYS

Surrogate keys have been created for all the dimension tables before loading the data in the data warehouse. These dimensional surrogate keys, which are used as Primary Keys in the Dimension tables, are created as Foreign Keys in the Fact tables that need to use these dimensions. We have created and used the below Surrogate Keys in our warehouse -

- Store_Key - Unique Identifier for the Store Dimension Table
- Product_Key - Unique Identifier for the Product Dimension Table
- Date_Key - Unique Identifier for the Date Dimension Table
- Category_Key - Unique Identifier for the Product Category (those categories for which UPC level data is unavailable) Table

15.6 CREATION OF DERIVED COLUMNS

For meeting the requirements of the business questions a few columns were required to be derived from existing attributes. Such columns which were derived as part of the transformation process in the staging area are listed below. These columns were created in the final staging area tables while loading data from temp tables to the final staging tables.

Staging Temp Table	Staging Temp Table Attributes	Final Staging Table	Final Staging Table Attributes	Transformations done
Movement_tmp	Sales	Movement	Sales	Derived the column Sales in temp table, using formula: $Sales = Price * Move/Qty$ i.e. $Sales = Bundle_Price * No_of_Units_Sold/Bundle_Size$
	Total_Profit		Profit	Derived the column Profit in temp table, using formula: $Total_Profit = Profit * Sales/100$ (since the temp table, Profit is available as cents per dollar)
WeekDecode_tmp	Year	WeekDecode	Year	Extracted the datepart - Year as int from the Start Date Available in WeekDecode_tmp after converting the Start Date to date

			Month	Month	Extracted the datepart - Month as int from the Start Date Available in WeekDecode_tmp after converting the Start Date to date
--	--	--	-------	-------	---

15.7 SSIS FUNCTIONS

The SSIS functions that aided during the transformation and cleaning process include:

- 1) LOOKUP:** It joins additional columns to the data flow by looking up values in a table.
- 2) DATA CONVERSION:** It converts data from one data type to another.
- 3) DERIVED COLUMNS:** It creates new column values by applying expressions to input columns.
- 4) AGGREGATE:** It aggregates data with functions such as count and sum.
- 5) UNPIVOT:** It makes an un-normalized dataset into a more normalized version by expanding values from multiple columns in a single record into multiple records with the same values in a single column.
- 6) MERGE JOIN:** The Merge Join Transformation in SSIS is used to perform SQL Joins such as Inner Join, Left Outer Join, Full Outer Join, and Right Outer Join in SSIS

16 PLAN FOR AGGREGATE TABLES

Aggregation is presumably the absolute most dominant element for improving execution in data warehouses. The presence of aggregates can speed up the querying, in view of this sensational impact on execution, building aggregates ought to be considered as a piece of act tuning the warehouse. In this manner, the current aggregates ought to be reconsidered occasionally as the business necessity changes. We note that the advantages of aggregation accompany the overhead of extra storage and support overheads. Those regions that will be every now and again gotten to or routinely announced become contenders for aggregation.

Aggregates have less rows than the base tables. Accordingly, when the vast majority of the queries are kept running against the aggregate tables rather than the base fact table, you see a colossal lift to execution in the data warehouse. Arrangement of aggregation tables is positively an extremely powerful strategy to improve query execution.

In our Warehouse Model, considering the business questions we identified the need to create one such Aggregate Table - FactStorewiseAggCategorySales. This Fact table has been created to record sales of product categories (those categories from Ccount for which UPC level data is unavailable), grouped by Store. This table records sales for a product category, in a particular store aggregated for all available dates. The Fact table - FactCategorySales records the same sales but date-wise. FactStorewiseAggCategorySales table contains data aggregated for all dates. We have included this third Fact table for optimal query performance because as a Data Warehouse is intended to store humongous historical data as well, the FactCategorySales table would grow with time and thus, fetching such aggregated results from that table can be time consuming. Storing the pre-computed aggregate data in another fact table saves the processing time utilized in real-time aggregation.

17 ORGANISATION OF DATA STAGING AREA

The data staging area is a temporary data storage area where data extracted from multiple heterogeneous data sources is stored before loading it into the data warehouse. The extracted data is stored separately in the staging area on a centralized or decentralized server, for performing a series of cleansing as well as transformation operations before the data is populated in the data warehouse.

ETL is one of the most critical processes when it comes to building a data warehouse and takes place in the data staging area where data is combined from multiple sources, cleaned, transformed and converted into the desired format and validated to ensure that clean and consistent data is loaded in the data warehouse for fast and optimized query processing.

In our project, we have created a database that denotes our data staging area, namely '601-group5-staging-area'. In this database, we have 2 types of tables - Tables appended by the suffix '_tmp' and others. The ones appended by '_tmp' are our temp tables, and these are the tables in

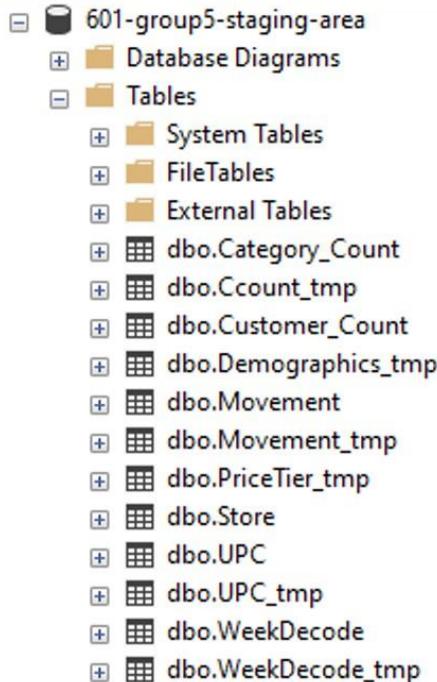
which we have extracted the data, as is from the provided source files, using SSIS Import Export tasks. After extraction, data in these temp tables is cleansed, transformed and then the clean data is stored inside the other type of tables, which are our final staging tables.

The sets of the two types of tables that compose our Staging area are listed below:

Temp tables: Ccount_tmp, Demographics_tmp, PriceTier_tmp, UPC_tmp, Movement_tmp and WeekDecode_tmp.

Final Staging tables: Category_Count, Customer_Count, Store, UPC, Movement and WeekDecode.

- The data in ‘Ccount_tmp’ is cleaned and broken into two tables- ‘Category_Count’ (that shows category wise sales per store on a particular date) and ‘Customer_Count’ (that shows the number of customers visiting a particular store on a particular date).
- The data in ‘PriceTier_tmp’ and ‘Demo_tmp’ was joined to create a new table called ‘Store’. This contains all the data that describes a DFF Store.
- ‘UPC_tmp’ was cleaned and loaded in ‘UPC’
- ‘WeekDecode_tmp’ was cleaned, transformed and loaded in ‘WeekDecode’
- ‘Movement_tmp’ was cleaned, transformed and loaded into a separate table called ‘Movement’.



18 PROCEDURES FOR EXTRACTION AND LOADING

18.1 EXTRACTION PROCEDURES

▪ Category_Count –

- The source file CCOUNT-Copy.csv is first extracted into the temp table – Ccount_tmp
- Irrelevant columns are removed from Ccount_tmp
- The fields for the Categories – DAIRY, BAKERY, MEAT, FROZEN, CHEESE, JEWELRY, BEER, WINE and SPIRITS are unpivoted and stored into rows, in the table – Category_Count. Pivot Key was stored under the column Category and Pivot value was stored under Sales
- Store_No, Sale_Date, Week_Number and Sales columns are assigned the appropriate data types in the Category_Count table

▪ Customer_Count

- Store_No, Date, Week_Number and Cust_Count fields are stored into Customer_Count from Ccount_tmp
- All fields are converted into their appropriate formats before copying

▪ WeekDecode

- Data from DFF Manual is scraped into Week Decode.txt
- This data is then extracted from the text file and loaded in the temp table WeekDecode_tmp
- Data from the temp table is transformed, in that the data type of Week_Number is changed to int, Start date is changed to date, End date is changed to date and Year, Month derived attributes are created while moving the data from WeekDecode_tmp to the final staging table - WeekDecode

▪ UPC

- Data from the csv file – UPCLND.csv is extracted and stored in the temp table – UPC_tmp
- Data in the temp table is cleaned, in that illegible values in the column – Product_Name are removed through regex expressions, and the clean data is moved to the final staging table - UPC

▪ Movement

- Data from the Movement file – wlnd.csv is extracted and stored in the temp table – Movement_tmp
- Data from this temp table is then converted to the appropriate data type – (int for Store_No, int for Week, int for Move, int for Qty, float for Price)
- 2 Derived columns are created – one for Sales which is calculated as – Price * Move/Qty; and another for Profit as – Profit = Profit_per_dollar_sale * Sales/100

- Then the clean, transformed data from Movement_tmp is moved to the final staging table – Movement after renaming the columns to more intuitive ones (Move -> No_of_Units_Sold, Qty -> Bundle_Size, Price -> Bundle_Price)
- Store
 - Data from the source files – DEMO.csv and DFF Store Price Tier.xls is moved to the temp tables- Demographics_tmp and PriceTier_tmp
 - Blank and NULL records are deleted
 - After selecting only relevant attributes from these 2 tables, the data is joined on Store_No and the merged data is moved to the final staging table – Store after conversion of data fields to their appropriate formats.

18.2 LOADING PROCEDURES

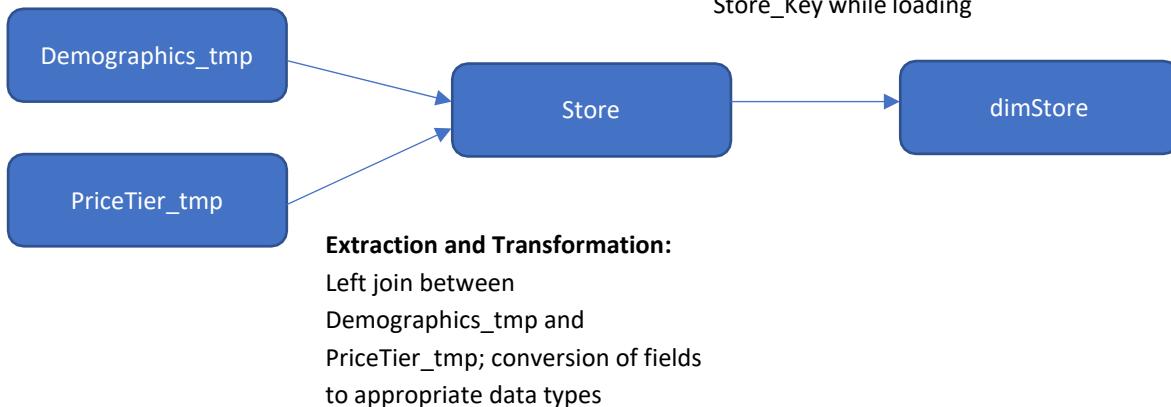
- dimStore
 - i. Created the Store dimension table – dimStore in the database – 601-group5-dw-area
 - ii. Data from the final staging table – Store is loaded in this dimension table
 - iii. A new column – Store_Key is created in the table during definition, which is an autoincrement, not null Surrogate Key which will be utilized as the Primary Key
- dimProduct
 - i. Created the Product dimension table – dimProduct in the database – 601-group5-dw-area
 - ii. Data from the final staging table – UPC is loaded in this dimension table
 - iii. A new column – Product_Key is created in the table during definition, which is an autoincrement, not null Surrogate Key which will be utilized as the Primary Key
- dimProductCategory
 - i. Created the Product Category dimension table – dimProductCategory in the database – 601-group5-dw-area
 - ii. This dimension table has been created to store data for Product Categories in Ccount, for which UPC level data is not available, thereby making it necessary to store sales for these categories at the categorical level
 - iii. The value for different available categories, from the final staging table – Category_Count is loaded in this dimension table
 - iv. A new column – Category_Key is created in the table during definition, which is an autoincrement, not null Surrogate Key which will be utilized as the Primary Key
- dimTime
 - i. Created the Date/Time dimension table – dimTime in the database – 601-group5-dw-area
 - ii. Data from the final staging table – WeekDecode is loaded in this dimension table

- iii. A new column – Date_Key is created in the table during definition, which is an autoincrement, not null Surrogate Key which will be utilized as the Primary Key
- FactProductSales
 - i. Created the Product level Sales Fact table – FactProductSales in the database – 601-group5-dw-area
 - ii. This table stores sales for a Product (UPC), on a particular week, in a particular store
 - iii. Data from Movement Staging table is loaded in this fact table
 - iv. While loading, lookup is performed on dimDate, dimProduct and dimStore to fetch the corresponding Surrogate Keys, which will be compositely used as primary key for this fact table
- FactCategorySales
 - i. Created the Category level Sales Fact table – FactCategorySales in the database – 601-group5-dw-area
 - ii. This table stores sales for a Category (those categories that don't have UPC level sales record), on a particular week, in a particular store
 - iii. Data from Category_Count Staging table is loaded in this fact table
 - iv. While loading, lookup is performed on dimDate, dimProductCategory and dimStore to fetch the corresponding Surrogate Keys, which will be compositely used as primary key for this fact table
- FactStorewiseAggCategorySales
 - i. This is an aggregate fact table created to store category sales records, aggregated by store number. It has been created to accelerate the performance of queries that need details of category sales in a store for all the available dates
 - ii. Data from Category_Count staging table is grouped by Store_No and Category and then loaded in this fact table
 - iii. While loading, lookup is performed on dimStore and dimProductCategory to fetch the corresponding surrogate keys which will be compositely used as a primary key for this fact table

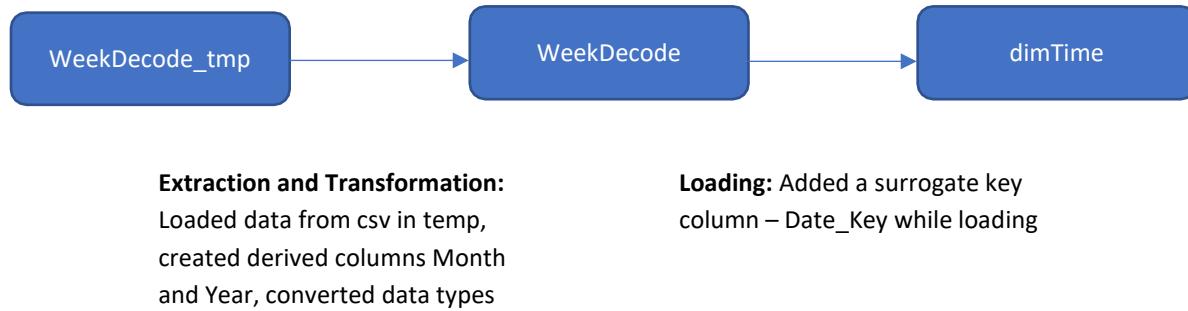
19 ETL FOR DIMENSION AND FACT TABLES

19.1 ETL FOR DIMENSION TABLES

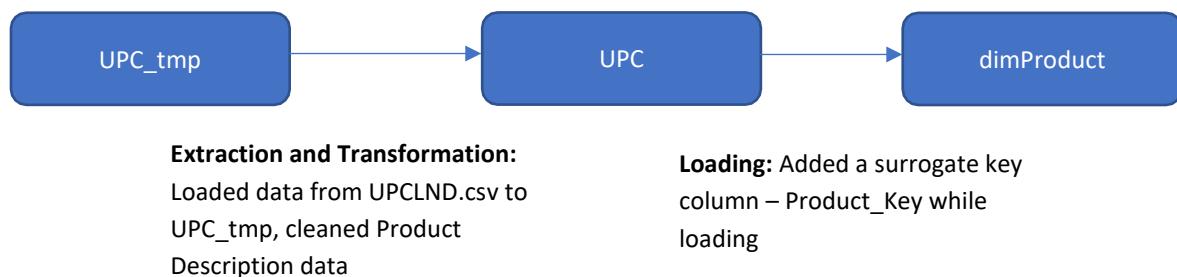
1. Store Dimension Table



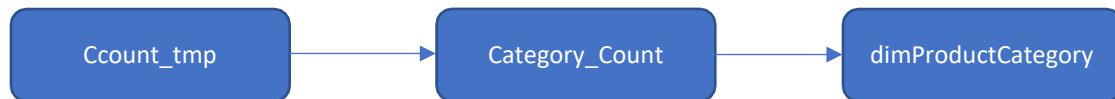
2. Date Dimension Table



3. Product Dimension Table



4. Product Category Dimension Table

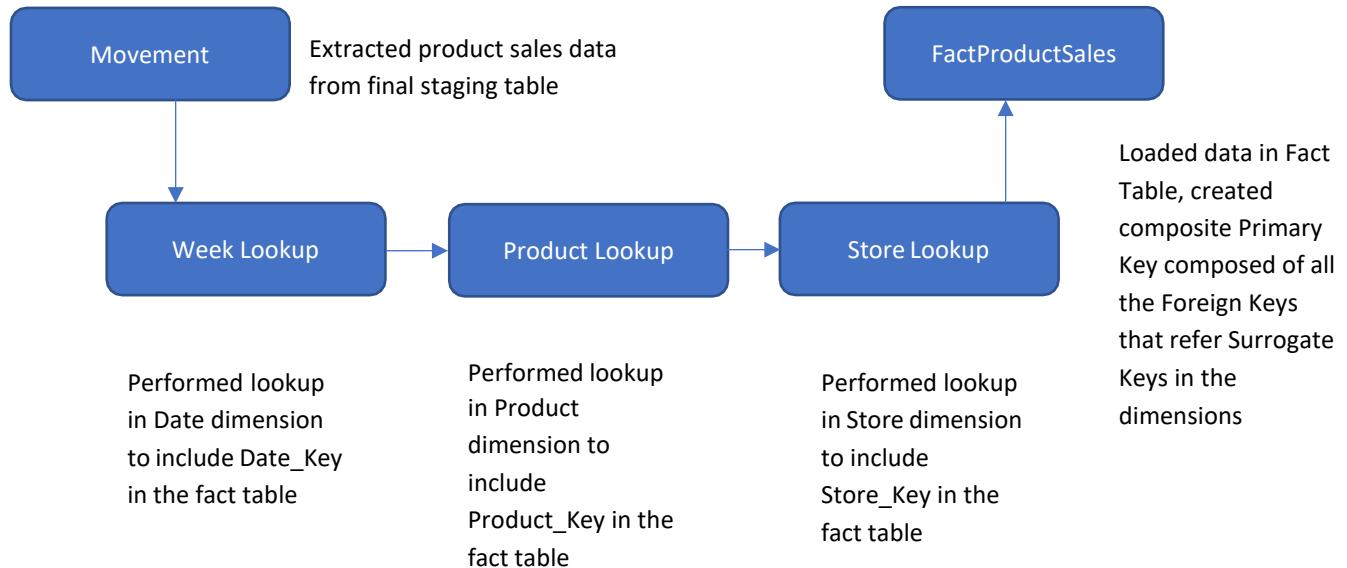


Extraction and Transformation:
Extracted data from Ccount-Copy,
unpivoted data for Categories from
Ccount and converted data types

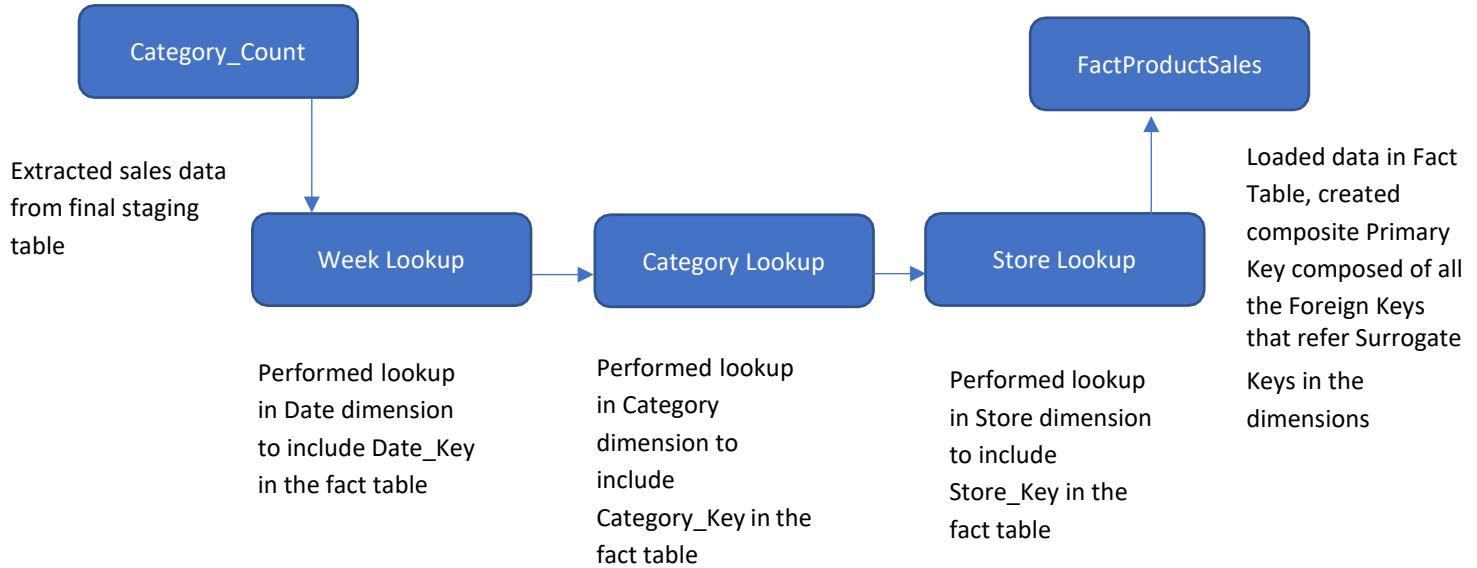
Loading: Loaded data for
Categories, added a surrogate key
column – Category_Key while
loading

19.2 ETL FOR FACT TABLES

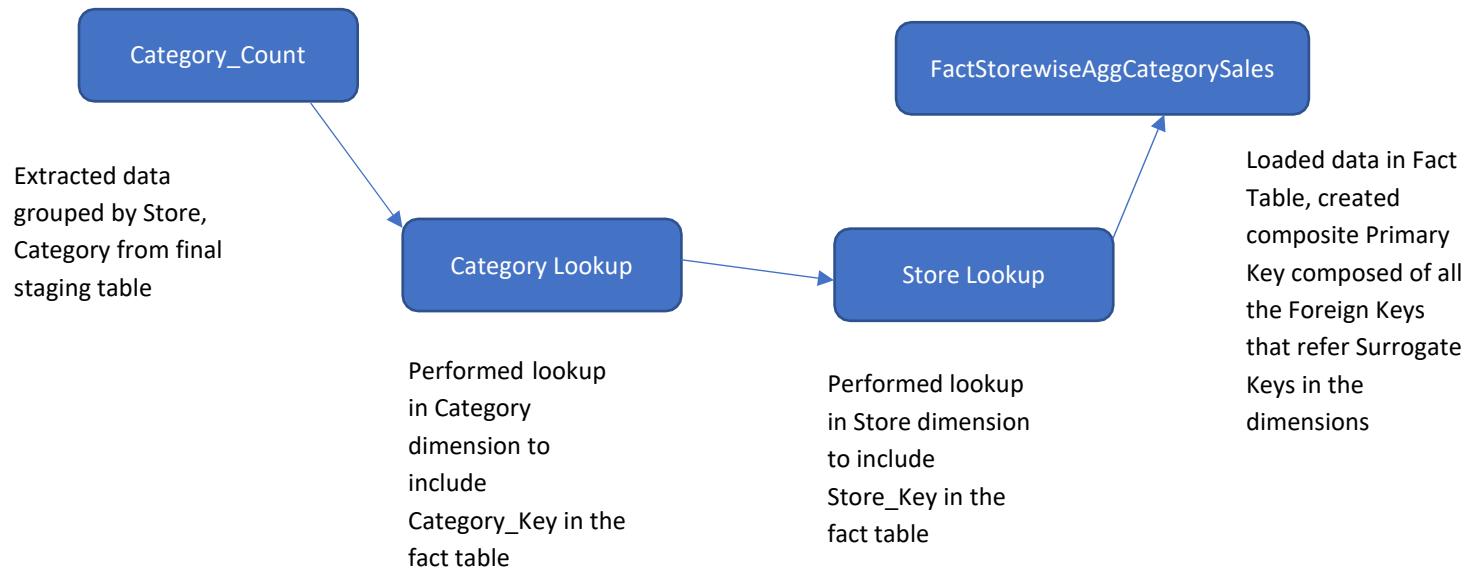
1. Product Sales Fact Table



2. Category Sales Fact Table



3. Storewise Aggregated Category Sales Fact Table



20 IMPLEMENTATION OF ETL

20.1 EXTRACTION OF DATA

20.1.1 TEMPORARY TABLE CREATION

Demographics Temp Staging Table

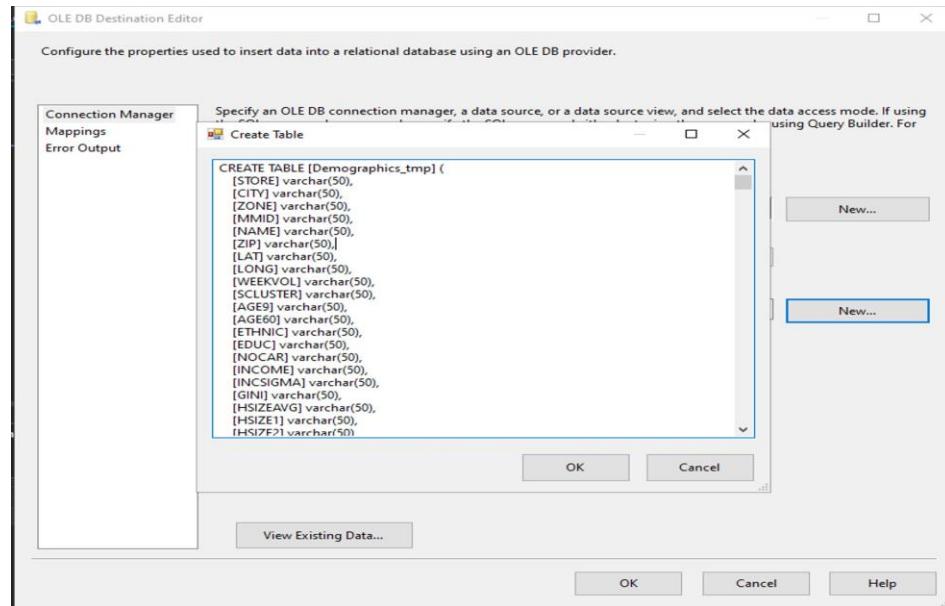


Figure 1: Demographic Temporary Table Creation

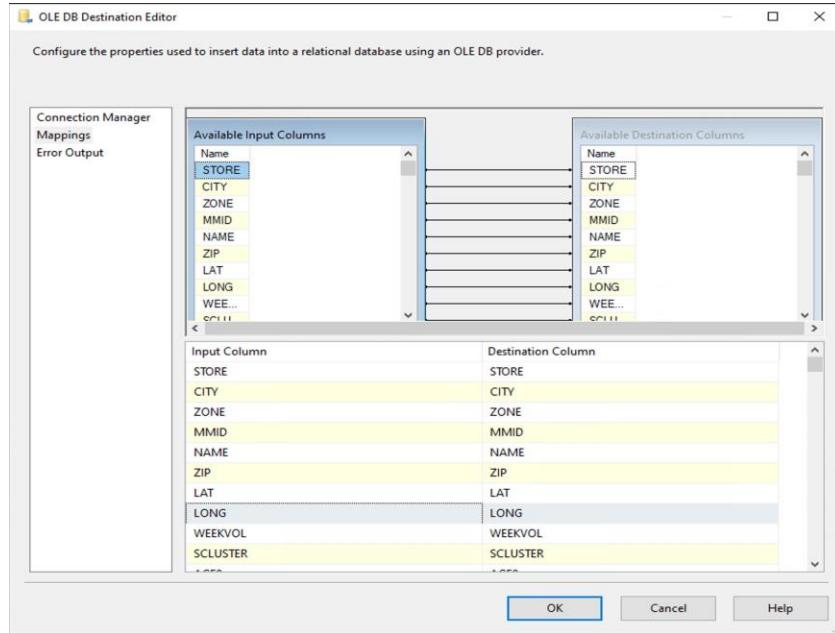


Figure 2: Source to Staging Mapping

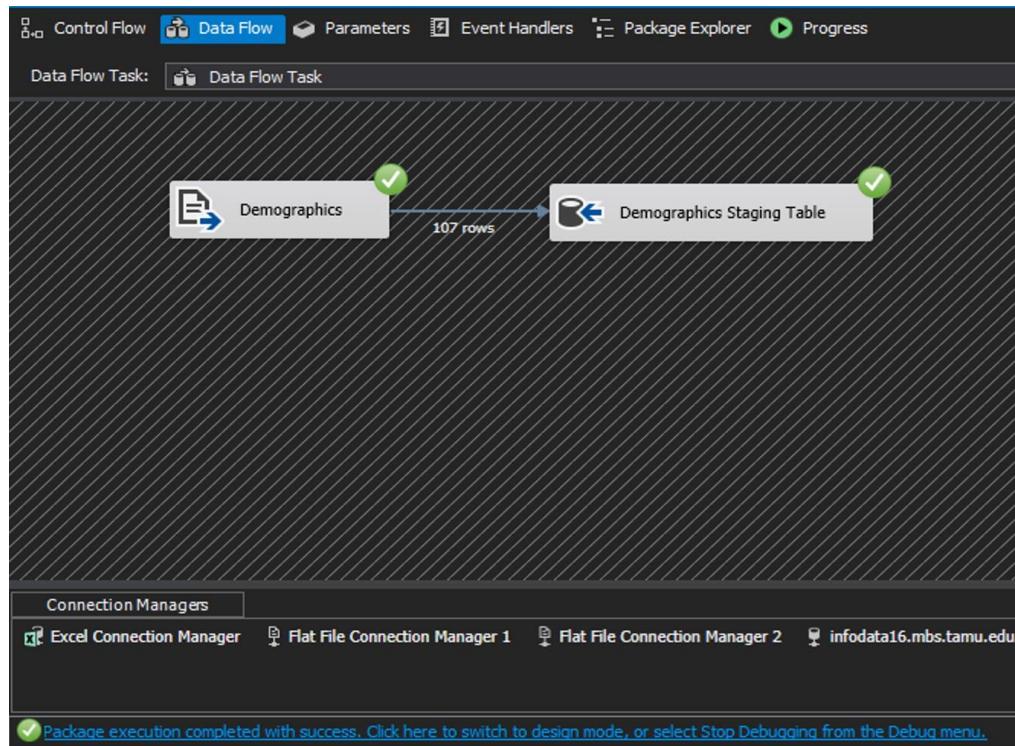


Figure 3: Data Flow

The screenshot shows a SQL Server Management Studio window with two tabs: 'SQLQuery3.sql - inf...area (gu0103 (221))' and 'SQLQuery1.sql - inf...aster (gu0103 (114))'. The results tab displays the output of the following query:

```
select top 1000 * from Demographics_temp
```

The results table has the following columns: STORE, CITY, ZONE, MMID, NAME, ZIP, LAT, LONG, WEEKVOL, SCLUSTER, AGE9, AGE60, ETHNIC, and EDUC. The data shows various store locations and their demographic details. The last row of the results table is highlighted in yellow.

At the bottom of the results pane, it says 'Query executed successfully.' and shows the status bar with 'infodata16.mbs.tamu.edu (13... | gu0103 (221) | 601-group5-staging-area | 00:00:00 | 107 rows'.

Figure 4: Demographics Temp Table Data

Price Tier temp staging Table

The screenshot shows a SQL Server Management Studio window with three tabs: 'SQLQuery5.sql - inf...area (gu0103 (101))', 'SQLQuery3.sql - inf...area (gu0103 (221))', and 'SQLQuery1.sql - inf...aster (gu0103 (114))'. The code pane contains the following T-SQL script:

```
USE [601-group5-staging-area]
GO

/****** Object: Table [dbo].[PriceTier_tmp] Script Date: 10/30/2020 8:01:35 PM *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[PriceTier_tmp](
    [Store] [varchar](50) NULL,
    [City] [varchar](255) NULL,
    [Price_Tier] [varchar](50) NULL,
    [Zone] [varchar](10) NULL,
    [Zip_Code] [nvarchar](10) NULL,
    [Address] [nvarchar](255) NULL
) ON [PRIMARY]
GO
```

The status bar at the bottom indicates 'Query completed with errors.' and shows the status bar with 'infodata16.mbs.tamu.edu (13... | gu0103 (101) | 601-group5-staging-area | 00:00:00 | 0 rows'.

Figure 5: Price Tier Temporary Table Creation

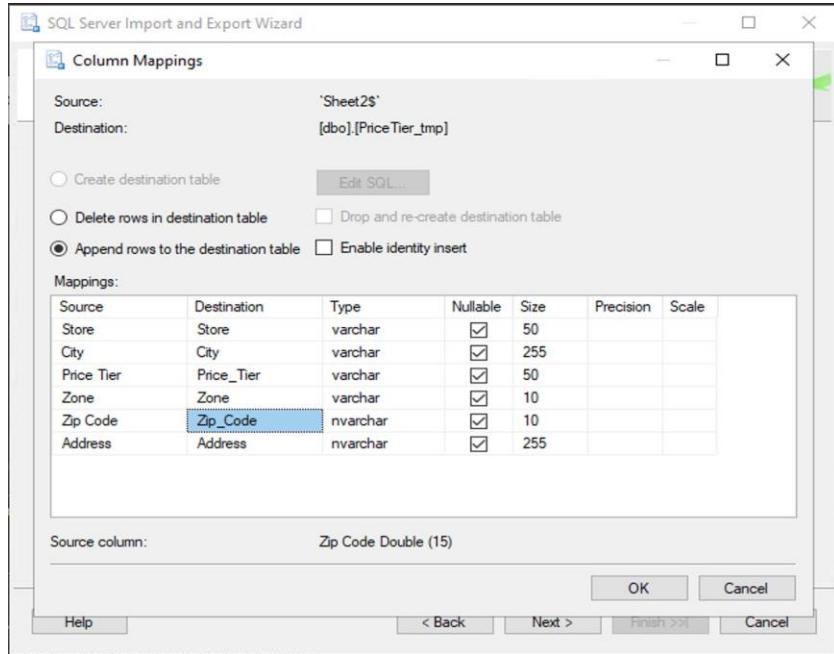


Figure 6: Source to Staging Mapping

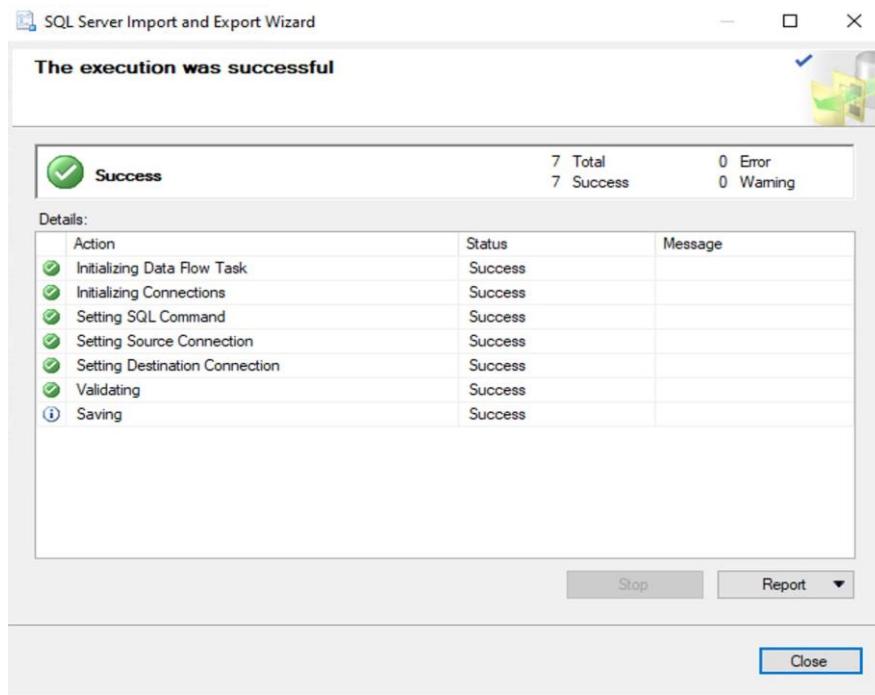


Figure 7: Data Flow

The screenshot shows a multi-tab SSMS interface. The active tab is 'SQLQuery5.sql - inf...area (gu0103 (101))'. The query window contains:

```
select * from PriceTier_tmp
```

The results pane shows the following data:

	Store	City	Price_Tier	Zone	Zip_Code	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 H...
4	8	Oak Lawn	Low	5	60435	8700 S. Cicero Ave.
5	9	Morton Gr...	Medium	2	60053	6931 Dempster
6	12	Chicago	High	7	60660	6009 N. Broadway...
7	14	Glenview	High	1	60025	1020 Waukegan ...
8	18	River Grove	Low	5	60171	8355 W. Belmont ...
9	19	Glen Elyn	NULL	NU...	60137	Closed
10	21	Hanover ...	CubFigh...	6	60103	1440 Irving Park ...
11	25	Chicago	NULL	NU...	60639	Closed
12	28	Mt. Prop...	Medium	2	60054	1145-55 Mt Prop...
13	32	Park Ridge	High	1	60068	1900 S. Cumberla...
14	33	Chicago	High	7	60657	3012 N. Broadwa...
15	39	Waukegan	NULL	NU...	60085	Closed

At the bottom, a message bar indicates: 'Query executed successfully.'

Figure 8: Price Tier Temp Table Data

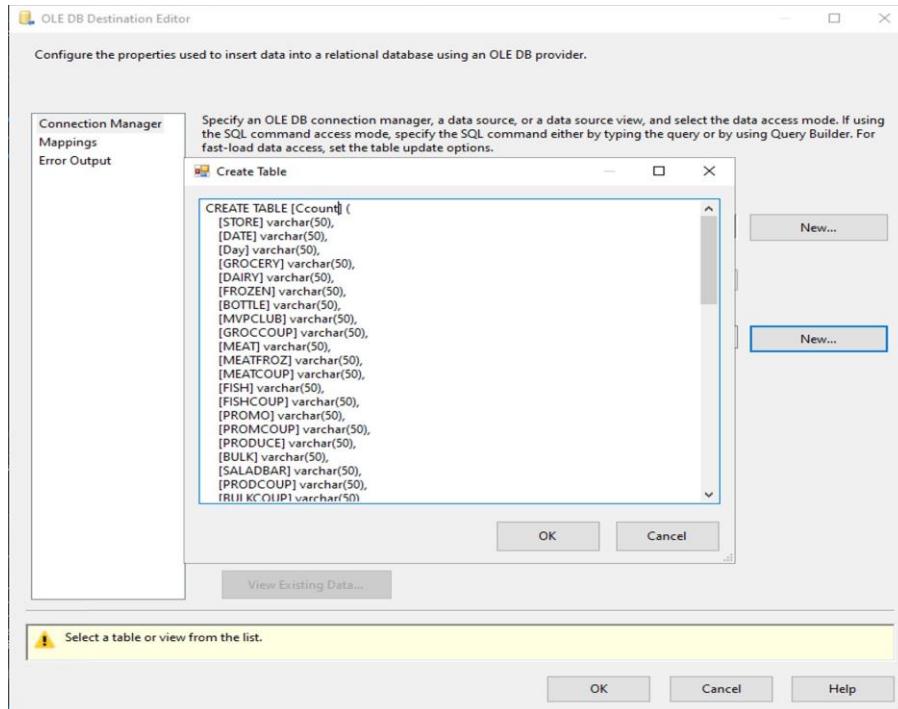
Ccount Tier temp staging Table

Figure 9: CCount Temp Table Creation

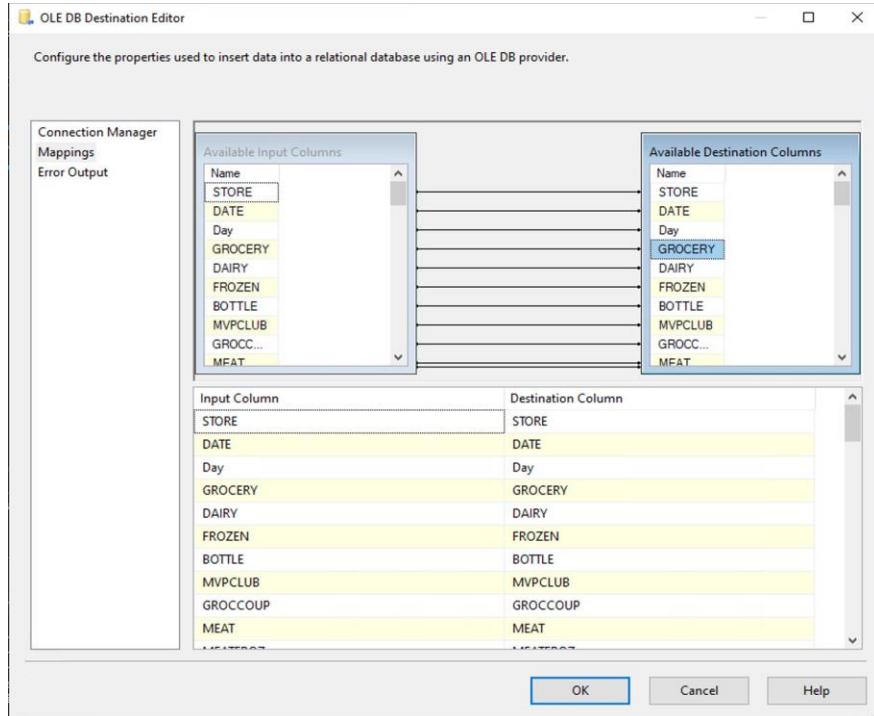


Figure 10: Source to Staging Mapping

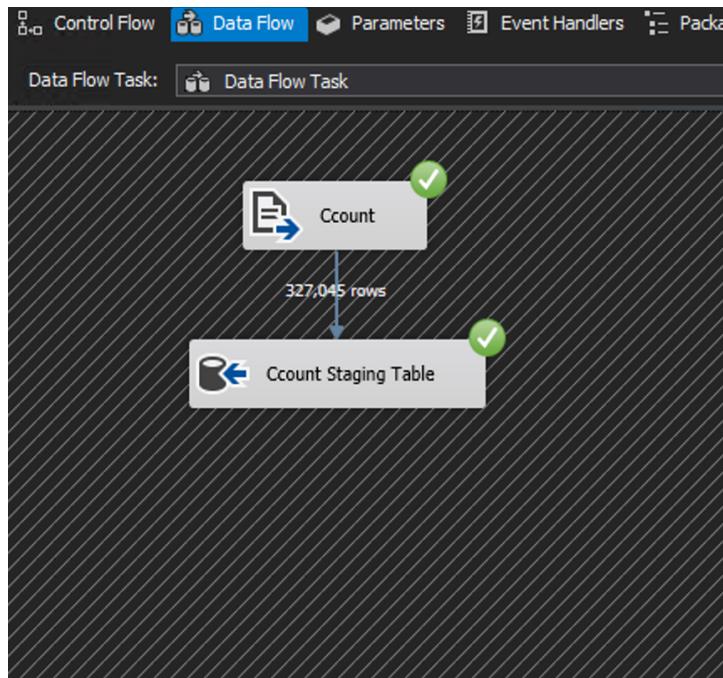


Figure 11: Data Flow

The screenshot shows the SSMS interface with two tabs open: 'SQLQuery3.sql - info...area (gu0103 (221))' and 'SQLQuery1.sql - inf...aster (gu0103 (114))'. The results tab displays the output of the following query:

```
select top 1000 * from CCount_tmp
```

The results grid contains 14 rows of data, each representing a different day from June 1996. The columns include STORE, DATE, Day, GROCERY, DAIRY, FROZEN, BOTTLE, MVPCLUB, GROCCOUP, MEAT, MEATFROZ, MEATCOUP, FISH, FISHCOUP, PROMO, and F. The data shows various sales figures for different product categories across different stores and days.

Figure 12: CCount Temp Table Data

UPC temp staging Table

Since we are tracking Detergent brand sales, we will be loading the UPC file of Detergent.

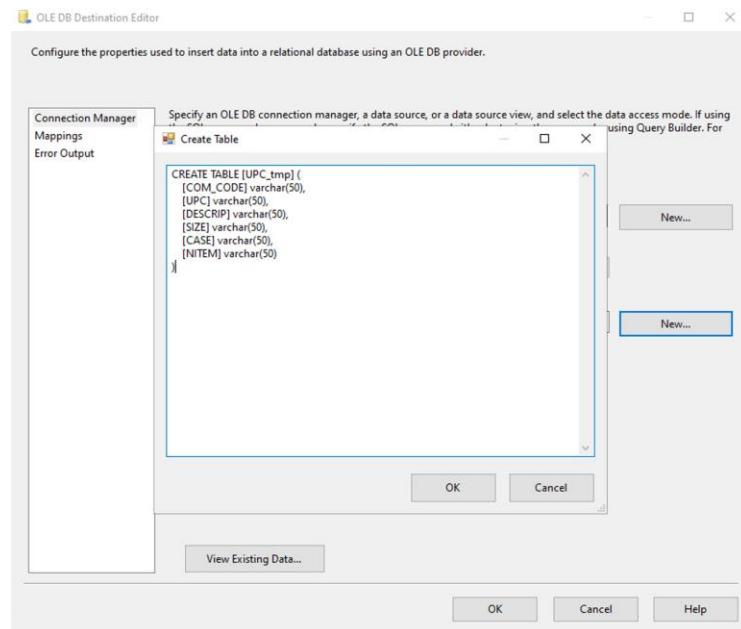


Figure 13: UPC Temporary Table Creation

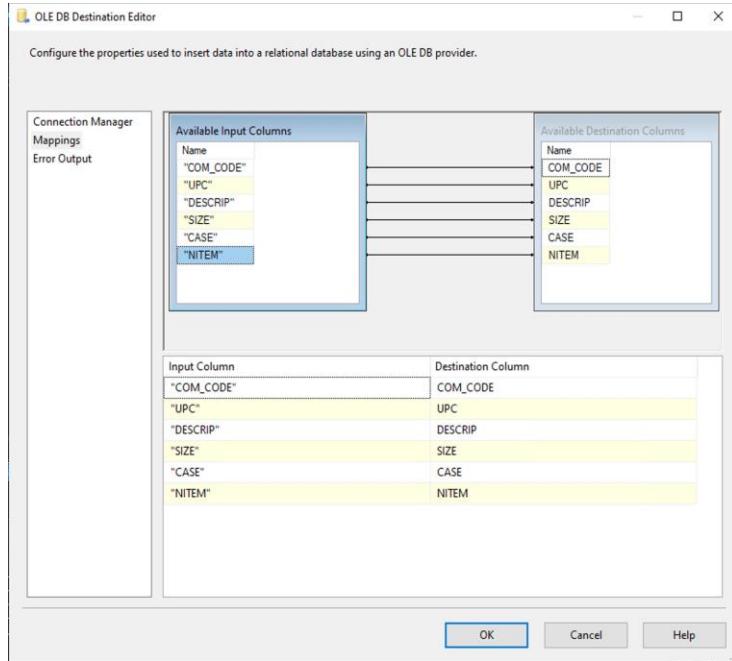


Figure 14: Source to Staging Mapping

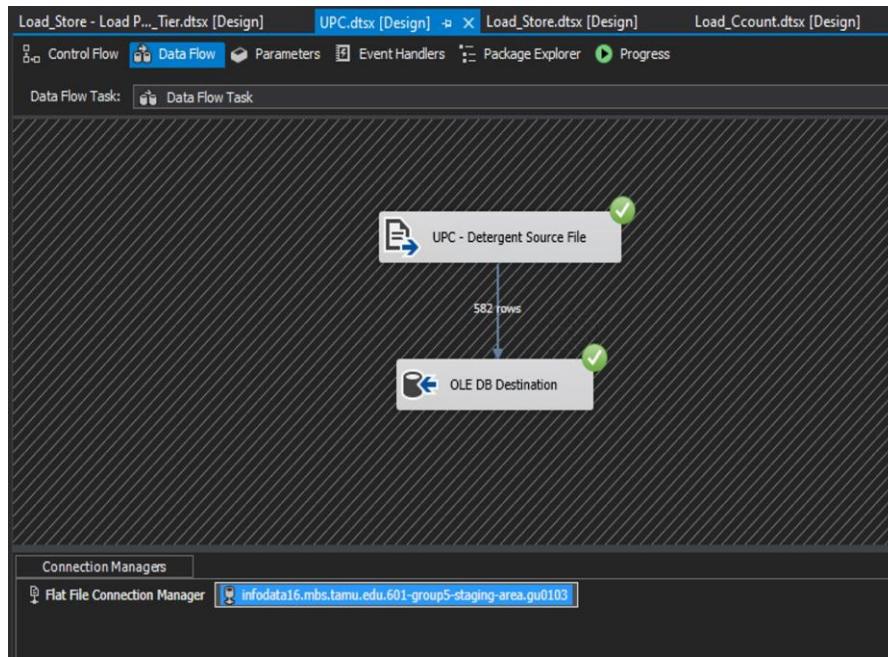


Figure 15: Data Flow

The screenshot shows a SQL Server Management Studio window with three tabs at the top: 'SQLQuery6.sql - inf...area (gu0103 (195))'*, 'SQLQuery5.sql - inf...area (gu0103 (101))'* (highlighted in yellow), and 'SQLQuery3.sql - inf...area (gu0103 (221))'*.

The 'Results' tab is selected, displaying the output of the following query:

```
select top 1000 * from UPC_tmp
```

The results show a table with columns: COM_CODE, UPC, DESCRIPT, SIZE, CASE, and NITEM. The data consists of 14 rows, each representing a different product entry. The last row is a total record.

	COM_CODE	UPC	DESCRIPT	SIZE	CASE	NITEM
1	666	8961111193	"OMEGA II LAUNDRY DE"	"25 LB"	1	2825161
2	666	1111135091	"ULTRA ALL"	"290 OZ"	1	2825001
3	666	1111135101	"ULTRA ALL 85 USE"	"223 OZ"	3	2859681
4	666	1111135104	"ULTRA ALL-42 USE"	"110 OZ"	4	2859661
5	666	1111135106	"ULTRA ALL-30 USE"	"79 OZ"	8	2859641
6	666	1111135110	"ULTRA ALL-18 USE"	"48 OZ"	10	2859621
7	666	1111135149	"ULTRA ALL 150 LOAD B"	"150 EA"	1	2869341
8	666	1111135218	"ULTRA ALL FREE N CLR"	"48 OZ"	10	2859701
9	666	1111135241	"ULTRA ALL FREE N CL"	"110 OZ"	4	2859721
10	666	1111135265	"CONC ALL \$1.50 OFF"	"157 OZ"	4	2859501
11	666	1111135340	"CONCTRND ALL \$2.50"	"20 LB"	2	2859601
12	666	1111135403	"NON PHOS CONC ALL"	"20 LB"	2	2859601
13	666	1111135405	"NON PHOS CONC ALL DE"	"157 OZ"	4	2859501
14	666	1111135407	"NON PHOS CONC ALL"	"84 OZ"	6	2859451
	ccc	1111135427	"CONCENTRATED ALL 75%"	"84 OZ"	6	2859451

At the bottom of the results pane, it says 'Query executed successfully.' and shows the connection details: infodata16.mbs.tamu.edu (13...) | gu0103 (195) | 601-group5-staging-area | 00:00:00 | 582 rows.

Figure 16: UPC Temporary Table Data

Movement temp staging Table

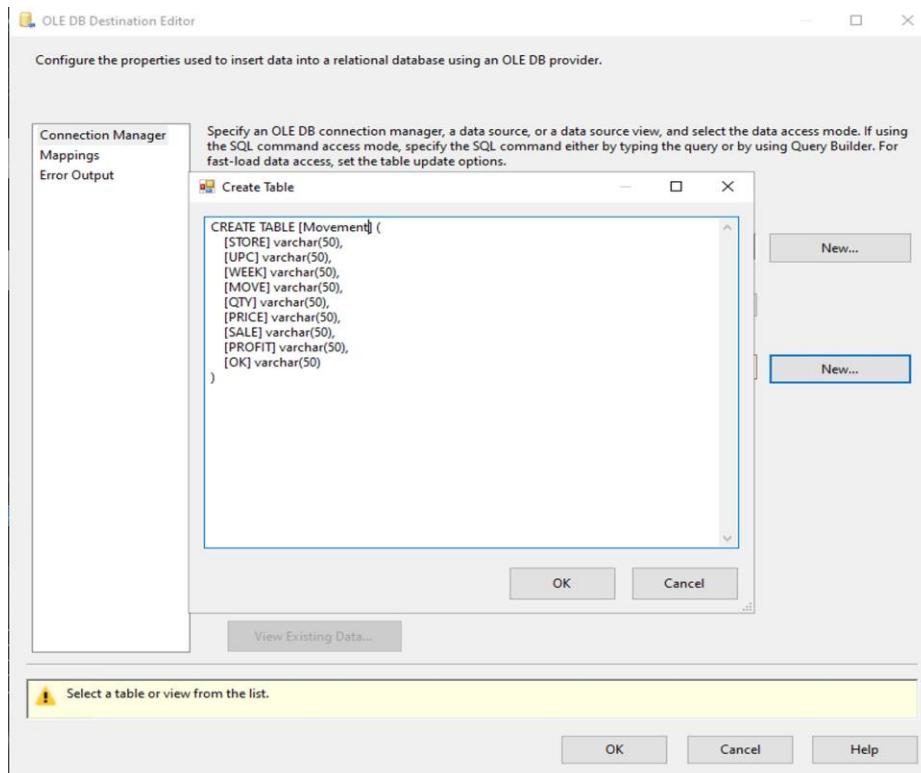


Figure 17: Movement Temporary Table Creation

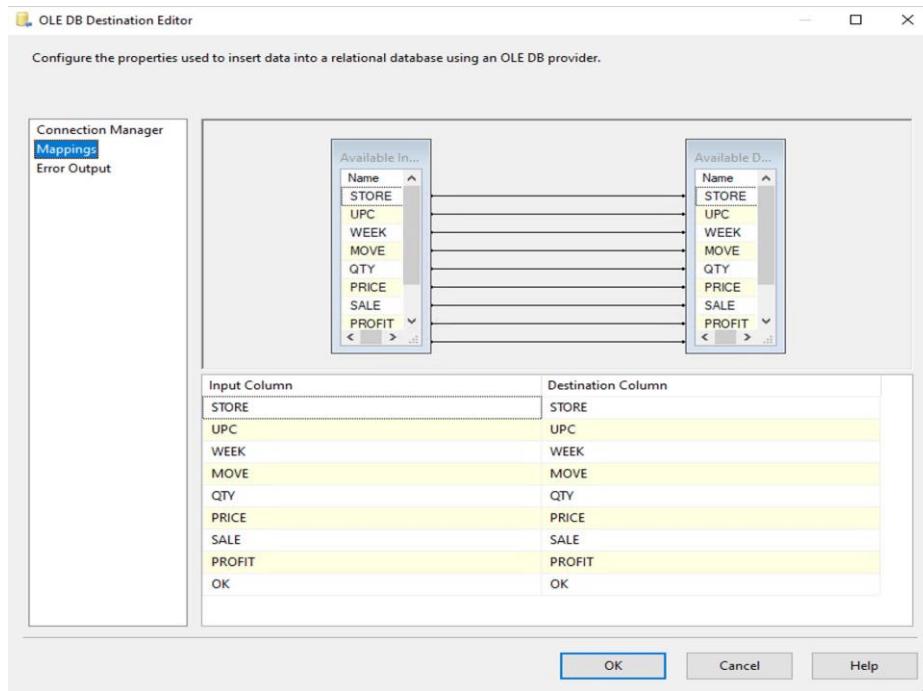


Figure 18: Source and Staging Mapping

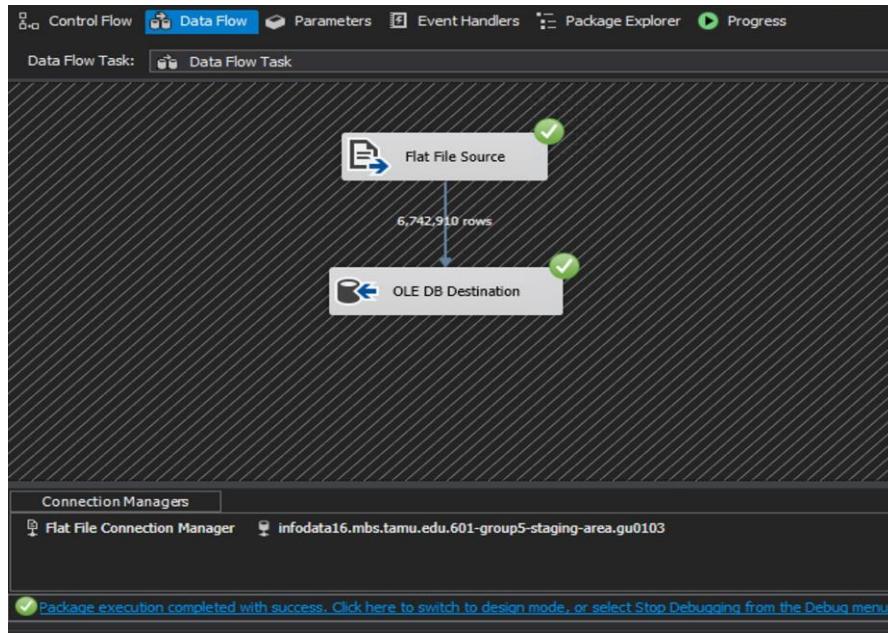


Figure 19: Data Flow

```
SQLQuery6.sql - inf...area (gu0103 (195)) * - X SQLQuery5.sql - inf...area (gu0103 (101)) * - SQLQuery3.sql - inf...area (gu0103 (221)) *
select top 1000 * from Movement_tmp
```

Store	UPC	Week	Move	Qty	Price	Sale	Profit	OK
1	109	1111189086	391	0	1	0	0	1
2	109	1111189086	392	0	1	0	0	1
3	109	1111189086	393	0	1	0	0	1
4	109	1111189086	394	0	1	0	0	1
5	109	1111189086	395	0	1	0	0	1
6	109	1111189086	396	0	1	0	0	1
7	109	1111189086	397	0	1	0	0	1
8	109	1111189086	398	0	1	0	0	1
9	109	1111189086	399	0	1	0	0	1
10	110	1111189086	1	0	1	0	0	1
11	110	1111189086	2	0	1	0	0	1
12	110	1111189086	3	0	1	0	0	1
13	110	1111189086	4	0	1	0	0	1
14	110	1111189086	5	0	1	0	0	1
15	110	1111189086	c	n	1	n	n	1

Query executed successfully. infodata16.mbs.tamu.edu (13...) gu0103 (195) | 601-group5-staging-area | 00:00:00 | 1,000 rows

Figure 20: Movement Temporary Data Table

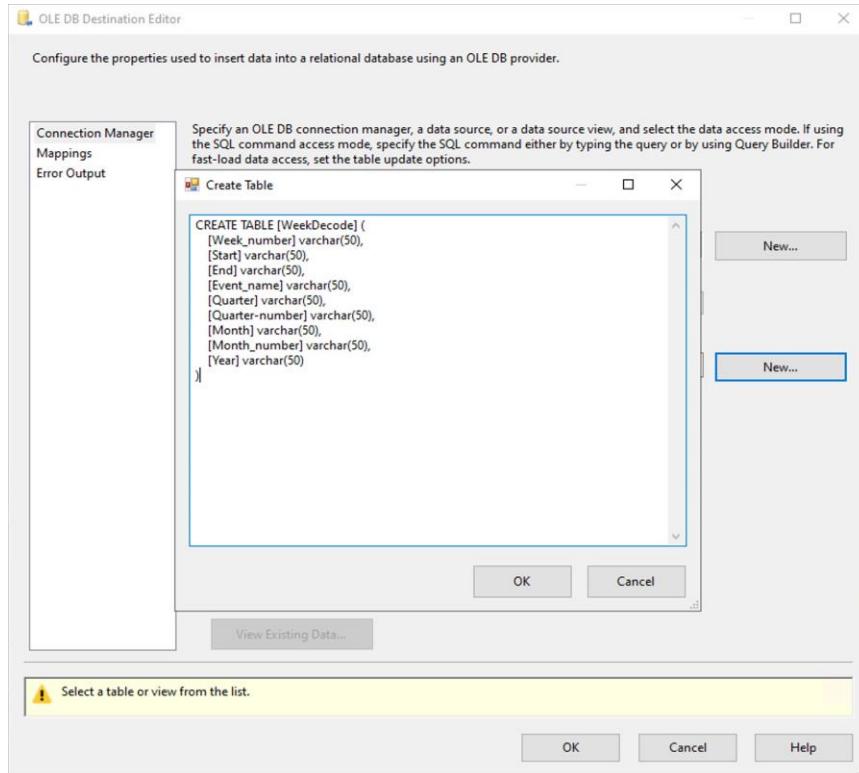
Week Decode temp staging Table

Figure 21: Week Decode Temporary Table Creation

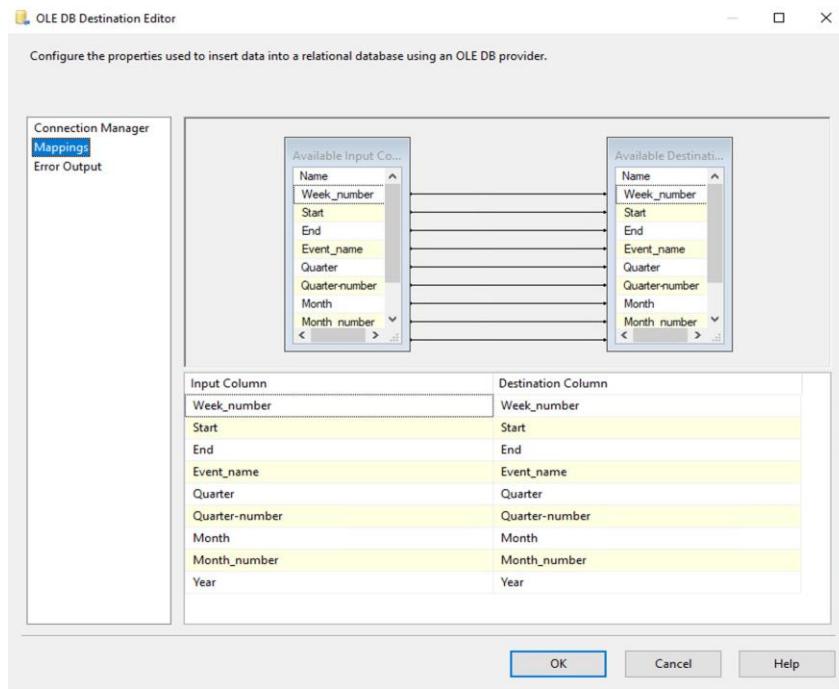


Figure 22: Source to Staging Mapping

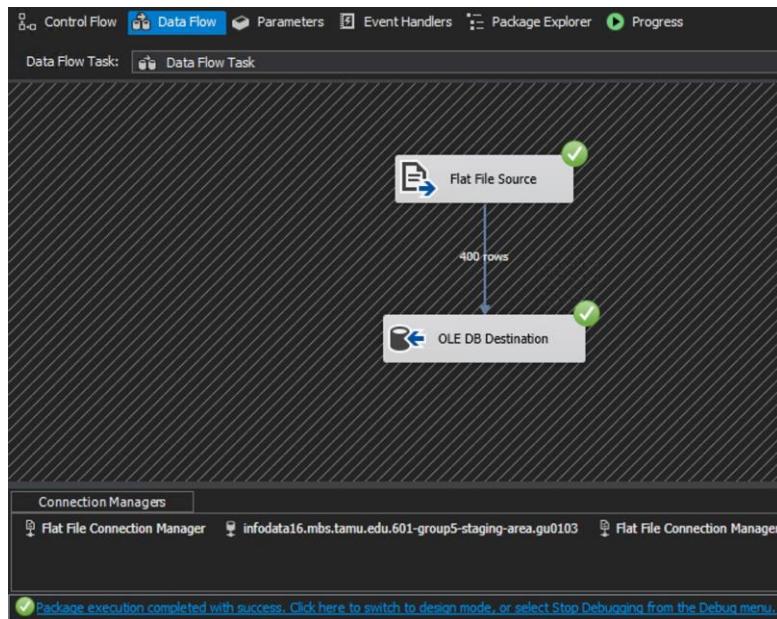


Figure 23: Data Flow

The screenshot shows a SQL Server Management Studio (SSMS) interface. At the top, there are three tabs: 'SQLQuery6.sql - inf...area (gu0103 (195))' (selected), 'SQLQuery5.sql - inf...area (gu0103 (101))', and 'SQLQuery3.sql - inf...area (gu0103 (221))'. Below the tabs, a query window displays the following T-SQL code:

```
select top 1000 * from WeekDecode_tmp
```

The results pane shows a table with the following data:

	Week_number	Start	End	Event_name	Quarter	Quarter-number	Month	Month_number	Year
1	1	09/14/89	09/20/89						
2	2	09/21/89	09/27/89						
3	3	09/28/89	10/04/89						
4	4	10/05/89	10/11/89						
5	5	10/12/89	10/18/89						
6	6	10/19/89	10/25/89						
7	7	10/26/89	11/01/89	Halloween					
8	8	11/02/89	11/08/89						
9	9	11/09/89	11/15/89						
10	10	11/16/89	11/22/89						
11	11	11/23/89	11/29/89	Thanksgiving					
12	12	11/30/89	12/06/89						
13	13	12/07/89	12/13/89						
14	14	12/14/89	12/20/89						
15	15	12/21/89	12/27/89	Christmas					

At the bottom of the results pane, a message indicates: 'Query executed successfully.' Below the results pane, the status bar shows: 'infodata16.mbs.tamu.edu (13...) | gu0103 (195) | 601-group5-staging-area | 00:00:00 | 400 rows'.

Figure 24: Week_Decode Temporary Data

20.2 STAGING AREA BEFORE CLEANING AND TRANSFORMATION

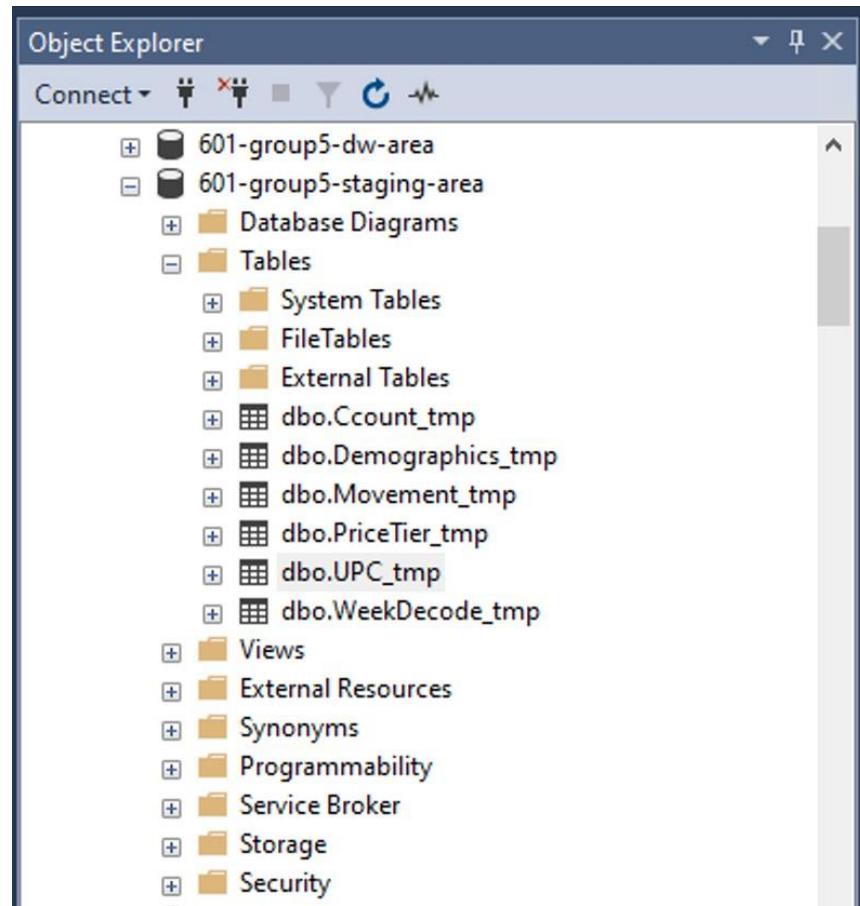
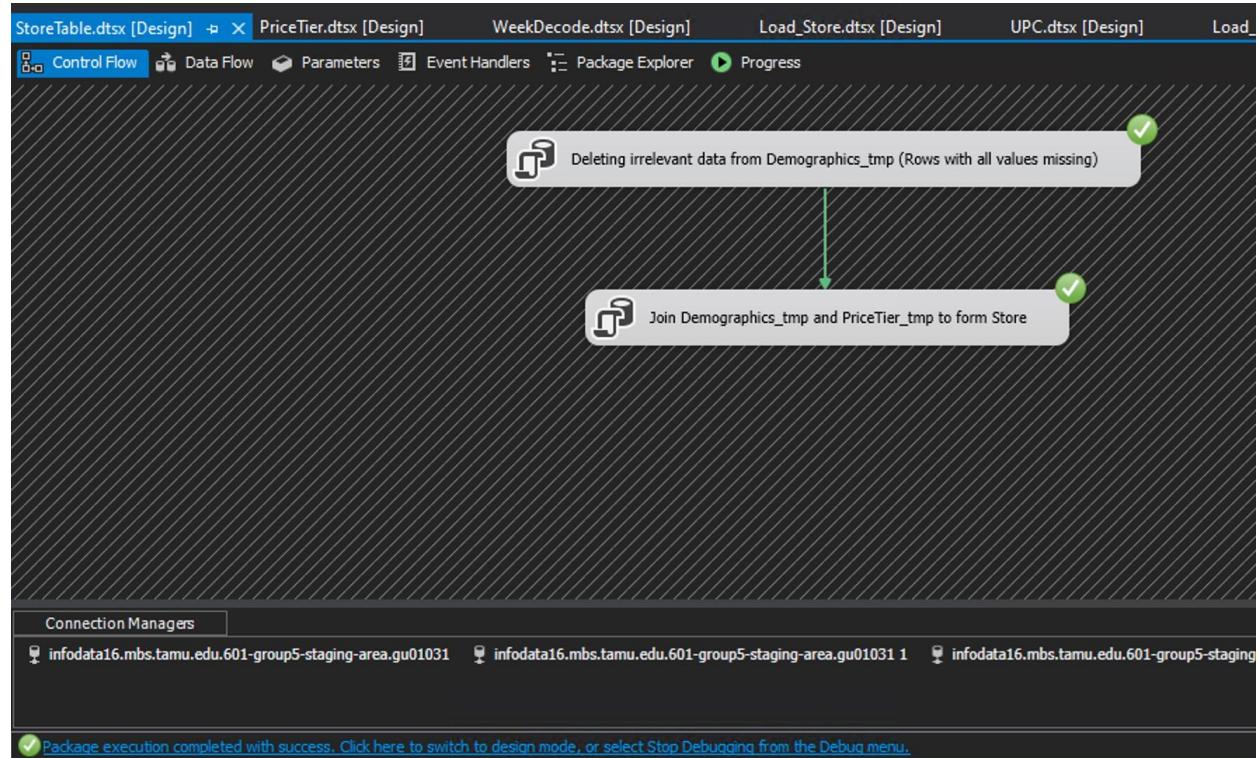


Figure 25: SSMS Database Snapshot containing Temporary Files

20.3 TRANSFORMATION AND CLEANING OF DATA

STORE

Store table is composed by joining cleaned data from Demographics Temp Table (StoreNo, City, Zone) and PriceTier Temp Table (PriceTier)



Removing Nulls/Missing Records

`DELETE FROM Demographics_tmp where (CITY=' ' or CITY=NULL) and (ZONE=' ' or ZONE = NULL)`

Moving temp data to final staging table

IF not exists (select * from sysobjects where name='Store' and xtype='U')

create table Store(

 Store_No int not null,

 City varchar(255),

 Zone varchar(10),

 Price_Tier varchar(50),

 Poverty decimal(10,4),

 Zip_Code varchar(50),

```
Address varchar(255),  
PRIMARY KEY (Store_No)  
)
```

INSERT INTO Store

```
SELECT convert(int, a.STORE), a.CITY as City, a.ZONE as Zone, Price_Tier,  
convert(decimal(10,4), a.POVERTY), Zip_Code, Address from Demographics_tmp as a join  
PriceTier_tmp as b on a.STORE = b.Store
```

COUNT

The Ccount temp Table will be stored in 2 different final tables - Category_Count, which will store the Unpivoted Categorical Sales information, for a particular store on a particular date and Customer_Count which will store the data for Customer Count in a particular store on a particular date.

Removing Nulls/Missing Records

```
DELETE FROM Ccount_tmp where DATE = '#NAME?' or DATE = '#VALUE!' or STORE = '0'  
or STORE = '1' or STORE is null
```

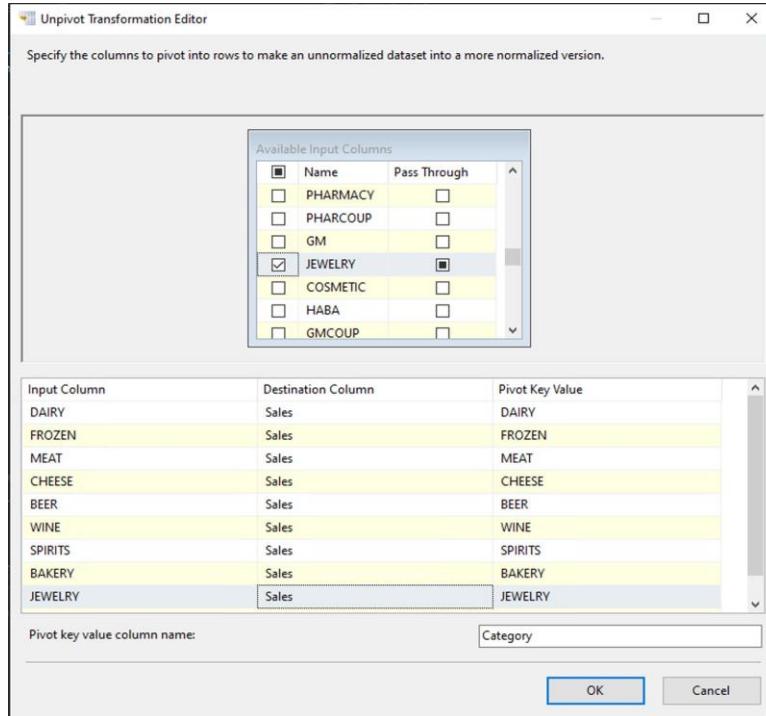
Removing inconsistent records

```
DELETE FROM Ccount_tmp where DAIRY like '-%' or FROZEN like '-%' or BAKERY like '-%'  
or MEAT like '-%' or BEER like '-%' or SPIRITS like '-%' or WINE like '-%' or CHEESE like '-%'  
or WEEK like '%-' or WEEK = '0'
```

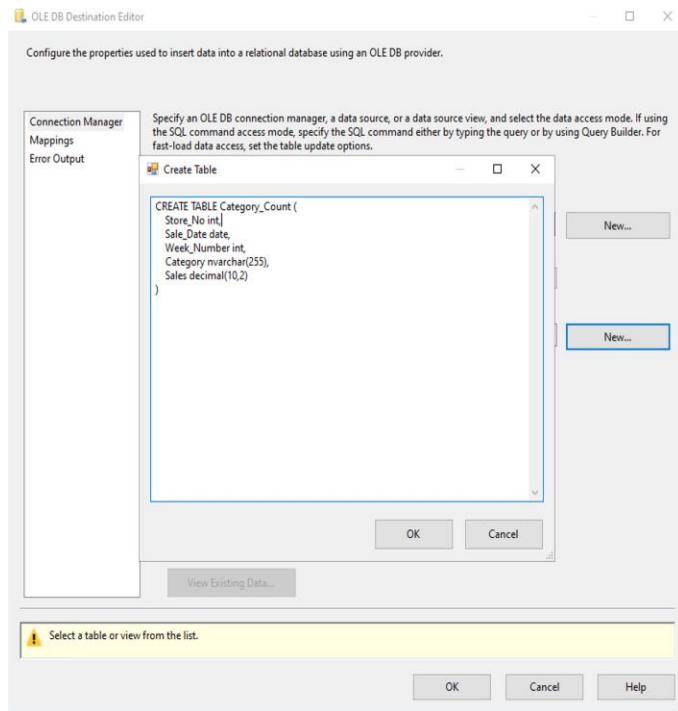
Data from Ccount_tmp will be moved to 2 different final staging tables.

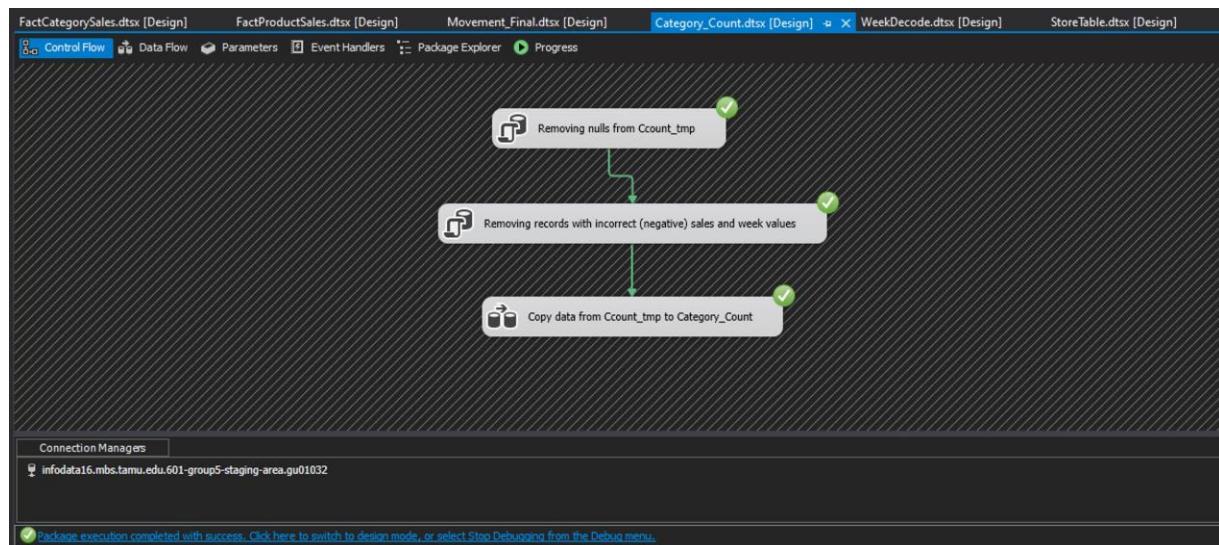
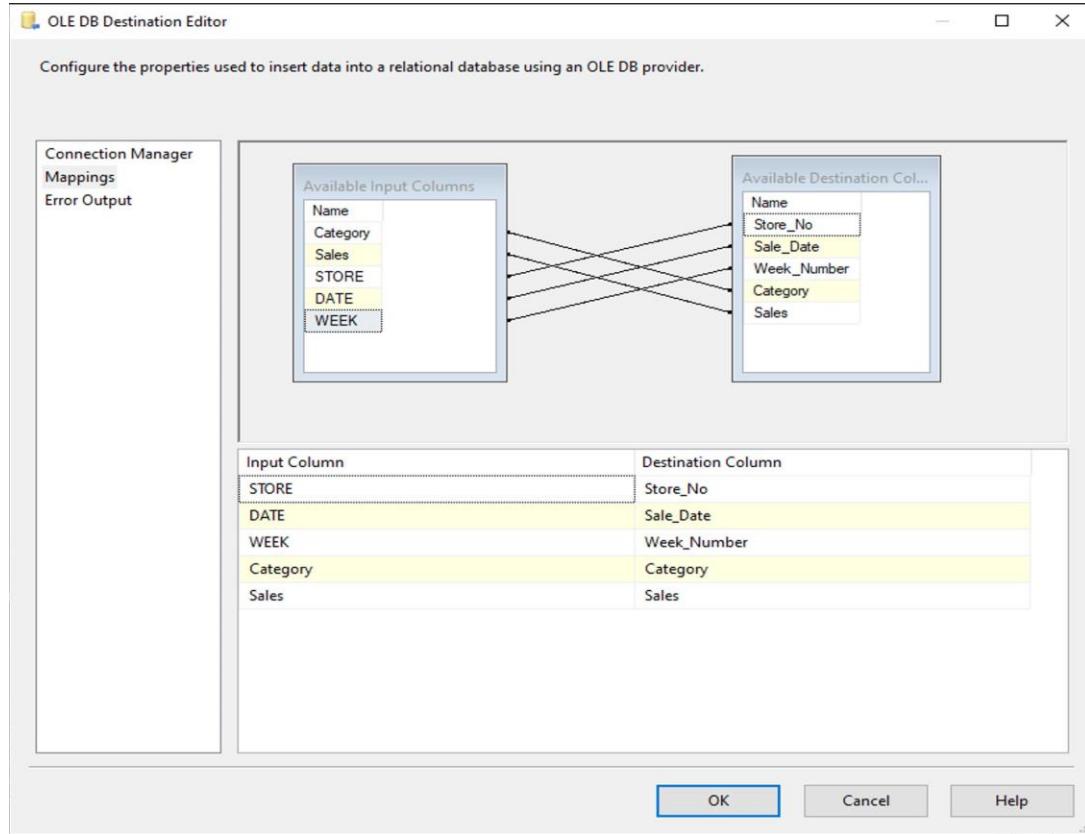
1. Moving categorical sales data to Category_Count staging table

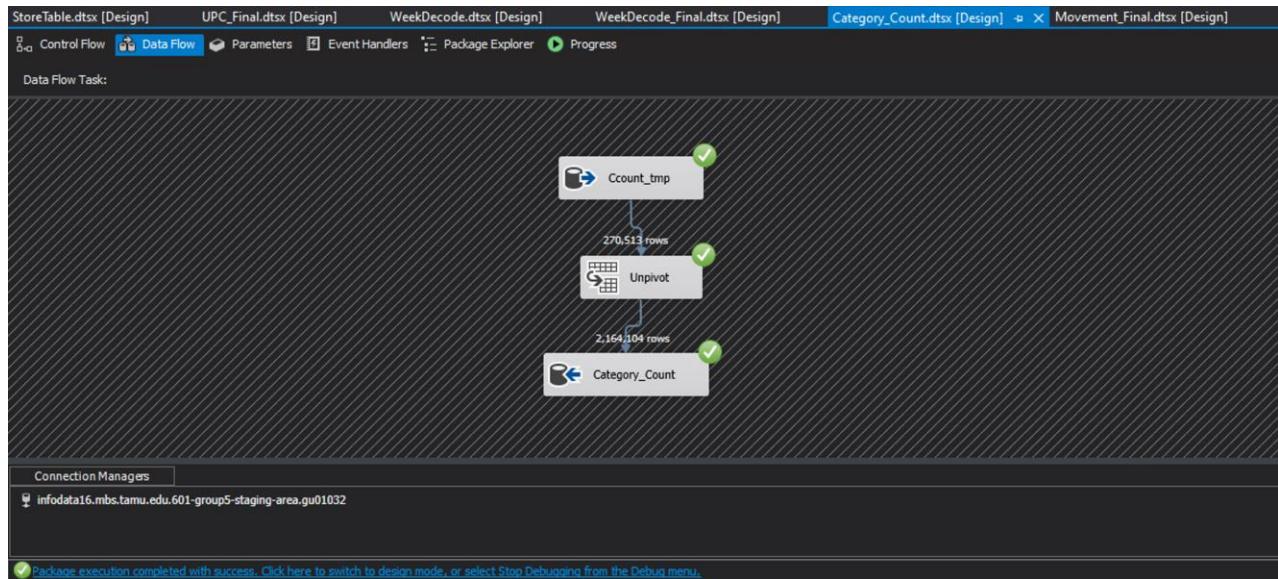
Unpivoting temp data



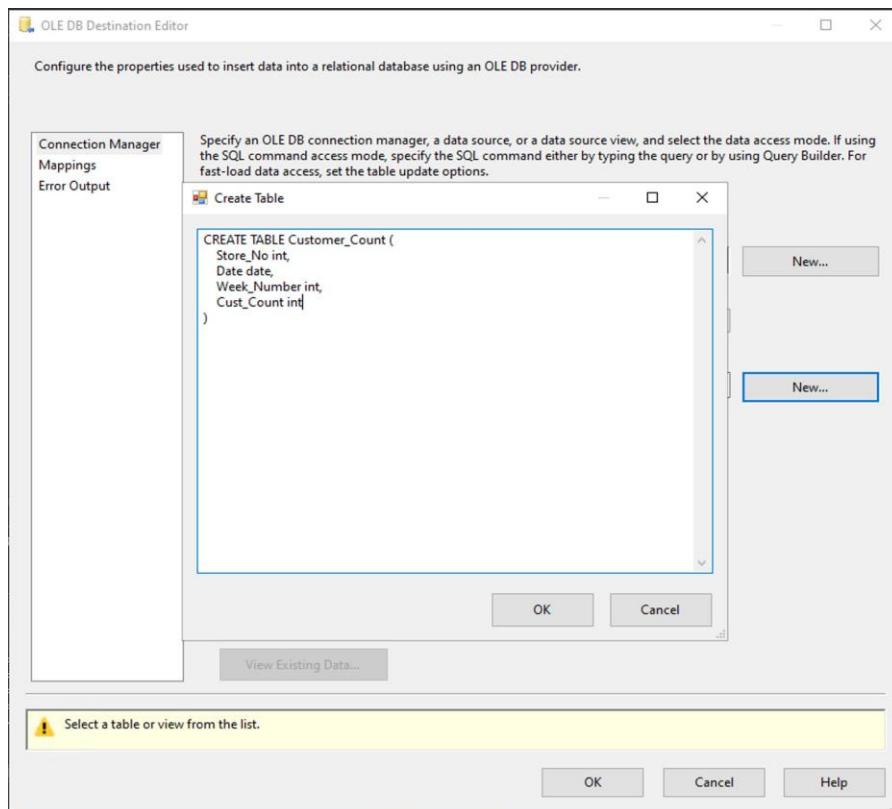
Moving Unpivoted data to final staging table

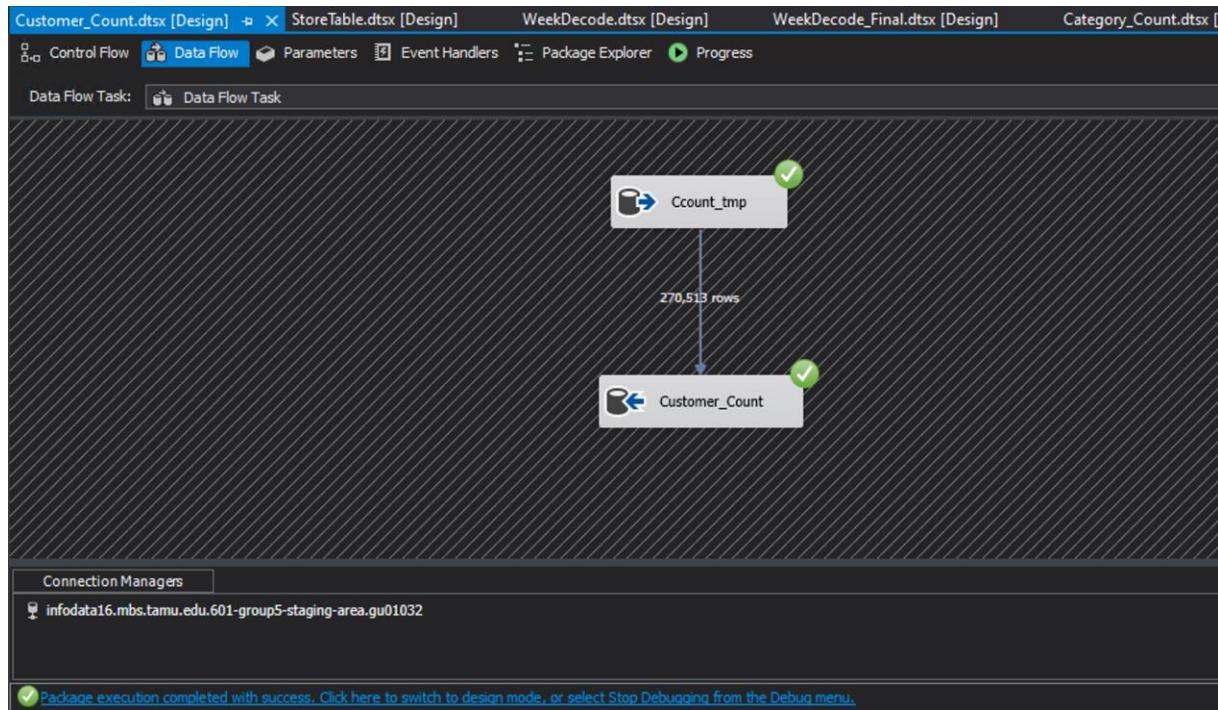
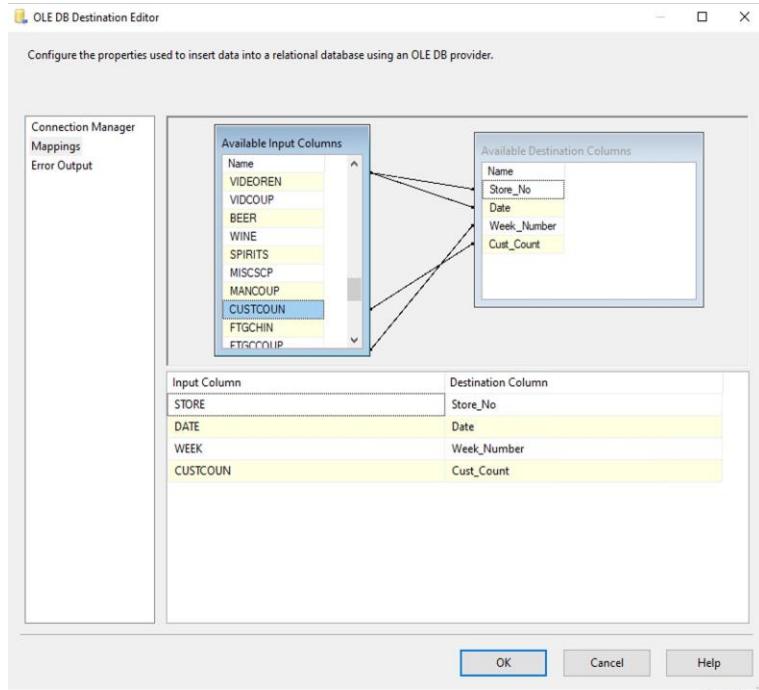






2. Moving customer count data to Customer_Count staging table





MOVEMENT

Data type conversion

```
ALTER TABLE Movement_tmp  
ALTER COLUMN Store int;
```

```
ALTER TABLE Movement_tmp  
ALTER COLUMN Week int;
```

```
ALTER TABLE Movement_tmp  
ALTER COLUMN Move int;
```

```
ALTER TABLE Movement_tmp  
ALTER COLUMN Qty int;
```

```
ALTER TABLE Movement_tmp  
ALTER COLUMN Price decimal(10,2);
```

Creating Derived Columns

#Created Derived column for Product Sales as: Sales = Price*Move/Qty

```
ALTER TABLE Movement_tmp  
ADD Sales AS (convert(decimal(10,2), Price)*convert(int, Move)/convert(int, Qty)) PERSISTED
```

#Created Derived column for Profit as: Total_Profit = Profit*Sales/100 (Since profit is in cents)

```
ALTER TABLE Movement_tmp  
ADD Total_Profit AS (convert(decimal(10,2), Profit)*convert(decimal(10,2), Price)*convert(int, Move)/(convert(int, Qty)*100)) PERSISTED;
```

Moving temp data to final staging table

```
IF not exists (select * from sysobjects where name='Movement' and xtype='U')  
create table Movement(  
    Store_No int,  
    UPC varchar(50),  
    Week int,
```

```
No_of_Units_Sold int,  
Bundle_Size int,  
Bundle_Price decimal(10,2),  
Sales decimal(10,2),  
Profit decimal(10,2)  
)  
INSERT INTO Movement(Store_No, UPC, Week, No_of_Units_Sold, Bundle_Size,  
Bundle_Price, Sales, Profit)  
SELECT Store, UPC, Week, Move, Qty, Price, Sales, Total_Profit from Movement_tmp
```

WEEKDECODE

Data Type Conversion

```
ALTER TABLE WeekDecode_tmp  
ALTER COLUMN Week_Number int;  
  
ALTER TABLE WeekDecode_tmp  
ALTER COLUMN Start date;  
  
ALTER TABLE WeekDecode_tmp  
ALTER COLUMN [End] date;
```

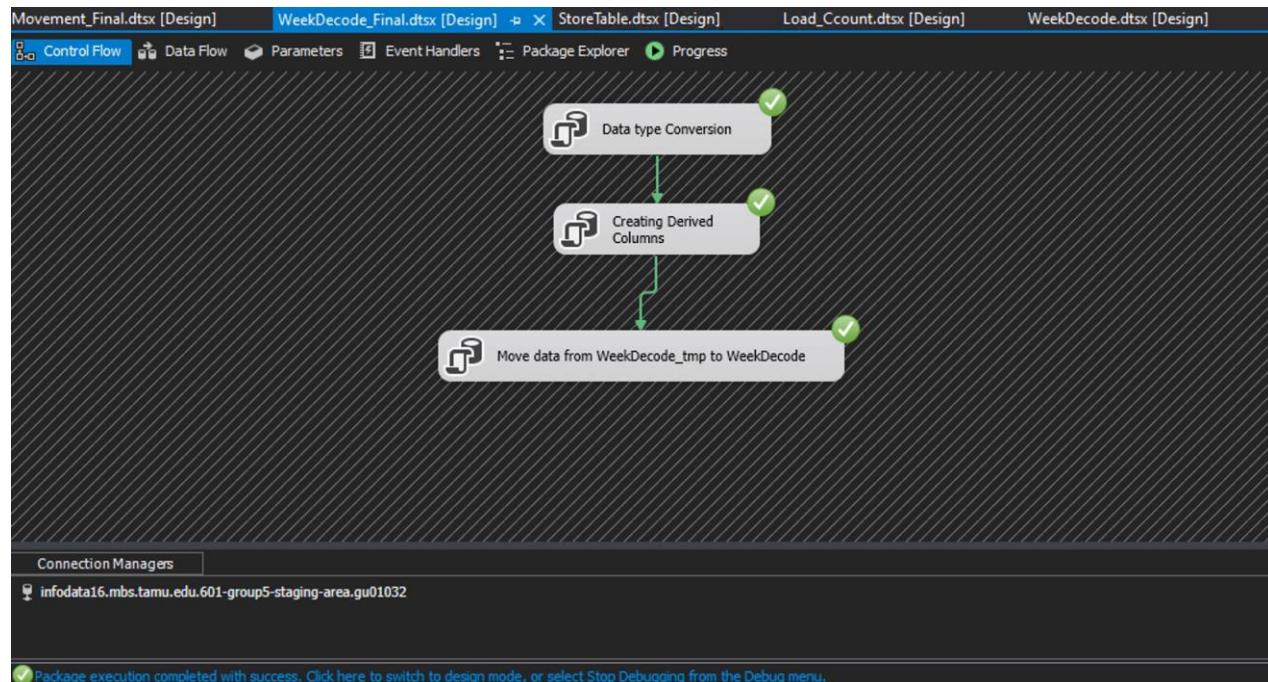
Creating Derived Columns

```
ALTER TABLE WeekDecode_tmp  
ADD Month_Part AS MONTH(Start)  
  
ALTER TABLE WeekDecode_tmp  
ADD Year_Part AS YEAR(Start)
```

Moving temp data to final staging table

```
IF not exists (select * from sysobjects where name='WeekDecode' and xtype='U')  
create table WeekDecode(  
    Year int,  
    Month int,  
    Week_Number int,  
    Special_Event varchar(255)  
)  
INSERT INTO WeekDecode(Year, Month, Week_Number, Special_Event)
```

SELECT convert(int,Year_Part), convert(int,Month_Part), Week_number, Event_name from WeekDecode_tmp



UPC

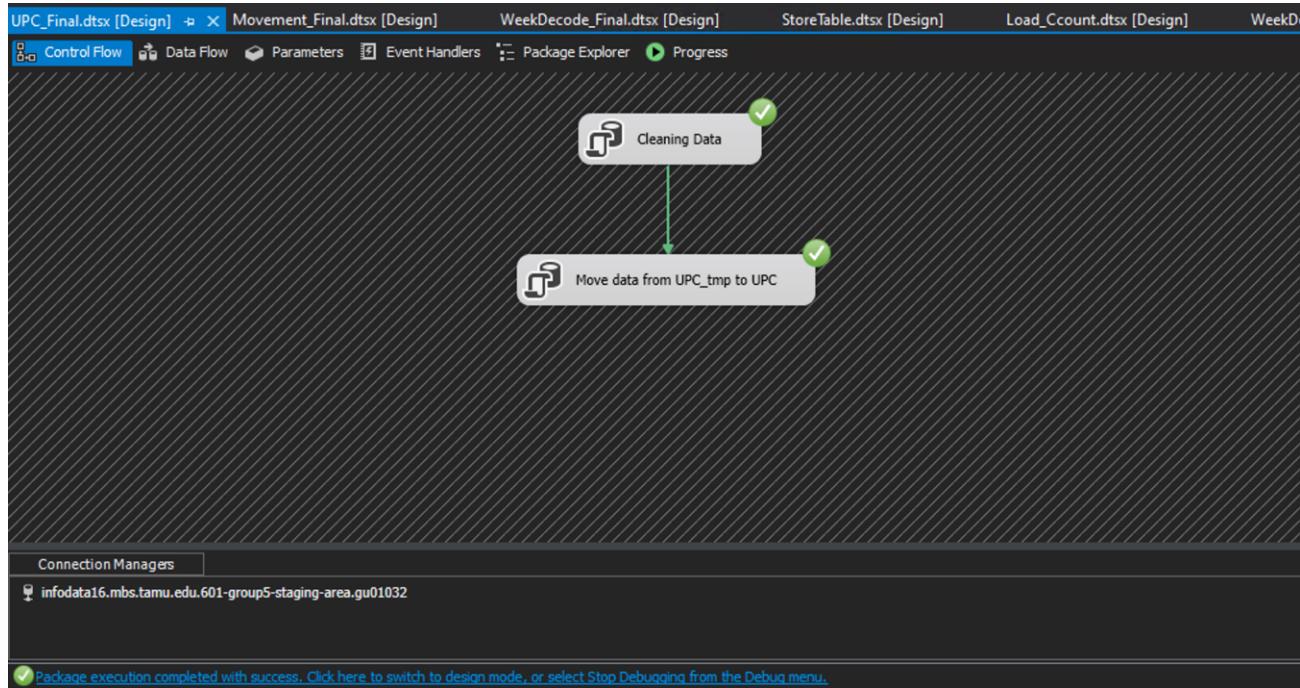
Cleaning Data

```
UPDATE UPC set Product_Name = Replace(REPLACE(Product_Name,  
SUBSTRING(Product_Name, PATINDEX('%[~,@,#,$,%,&,*^,&,%,*,(,)"]%', Product_Name),  
1),"')-'-'')
```

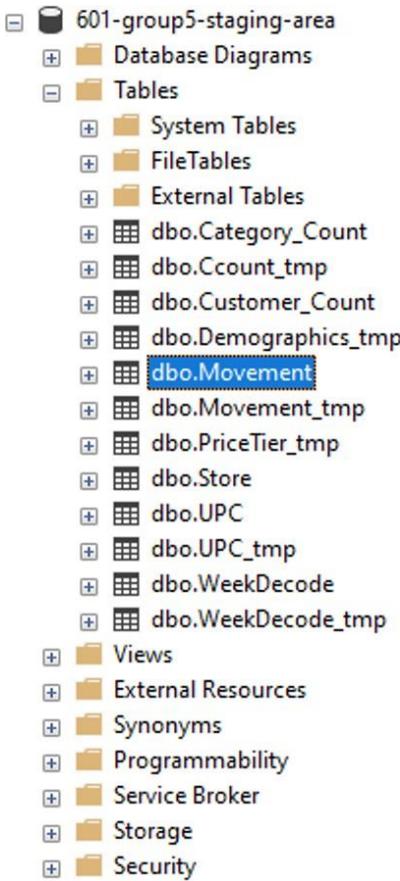
Moving temp data to final staging table

```
IF not exists (select * from sysobjects where name='UPC' and xtype='U')  
create table UPC(  
    UPC varchar(50),  
    Product_Name varchar(255)  
)
```

```
INSERT INTO UPC(UPC, Product_Name)  
SELECT UPC, DESCRIPT from UPC_tmp
```



20.4 STAGING AREA AFTER CLEANING AND TRANSFORMATION



20.5 LOADING OF DATA

20.5.1 CREATION OF DIMENSION TABLE

Sql queries for creation of dimension tables:

1. dimStore:

```
CREATE TABLE dimStore(  
    Store_Key int identity(1,1) primary key,  
    Store_No int,  
    City varchar(255),  
    Zone varchar(10),  
    Price_Tier varchar(50),  
    Poverty decimal(10,4),  
    Zip_Code varchar(50),  
    Address varchar(255))
```

2. dimTime

```
CREATE TABLE dimTime(  
    Date_Key int identity(1,1) primary key,  
    Year int,  
    Month int,  
    Week_No int,  
    Special_Event varchar(255)  
)
```

3. dimProduct

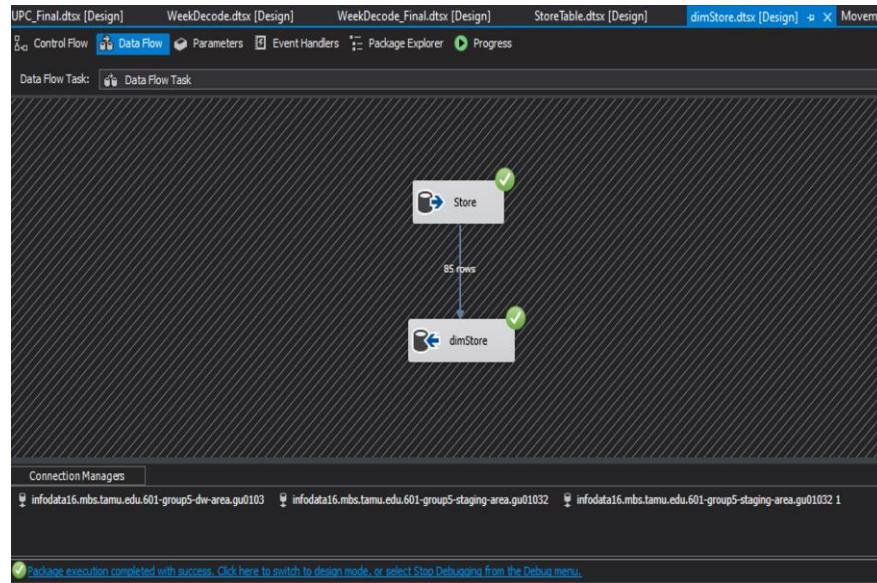
```
CREATE TABLE dimProduct(  
    Product_Key int identity(1,1) PRIMARY KEY,  
    UPC varchar(50) not null,  
    Product_Name varchar(255),  
    Product_Category varchar(255))
```

4. dimProductCategory

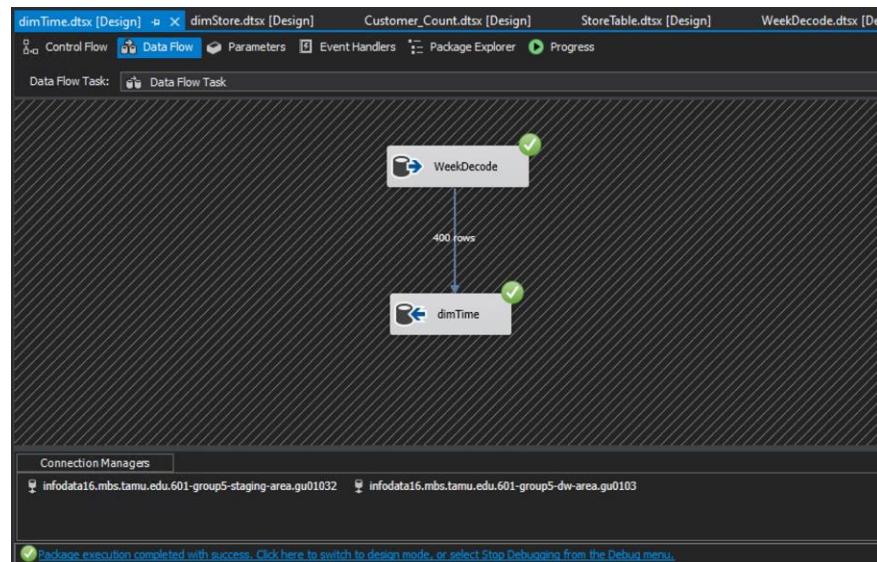
```
CREATE TABLE dimProductCategory(  
    Category_Key int identity(1,1) PRIMARY KEY,  
    Category nvarchar(255)  
)
```

20.5.2 POPULATING DIMENSION TABLES

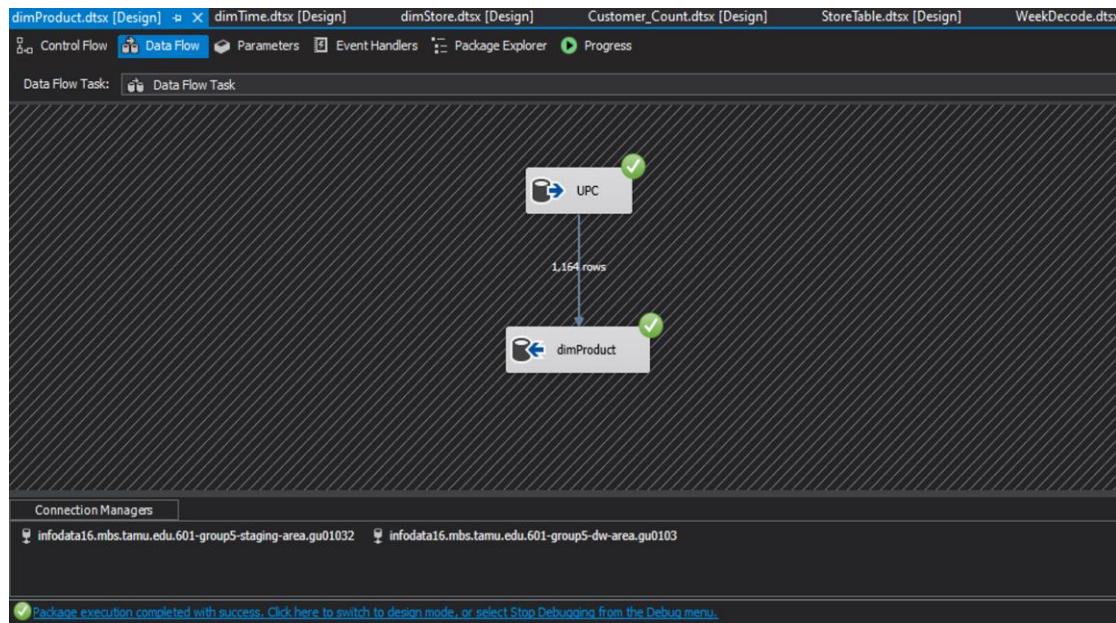
1. dimStore:



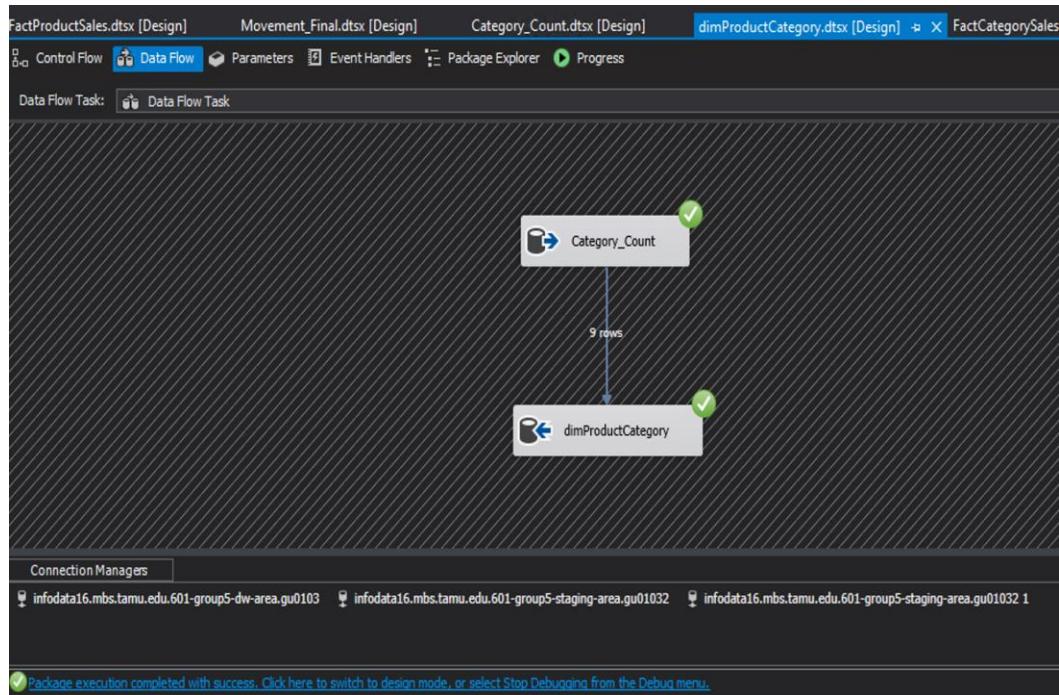
2. dimTime:



3. dimProduct:



4. dimProductCategory



20.5.3 CREATION OF FACT TABLES

Sql queries for creation of fact tables:

1. FactStorewiseAggCategorySales

```
CREATE TABLE FactStorewiseAggCategorySales(
    Store_Key int not null,
    Category_Key int not null,
    Sales decimal (10,2),
    PRIMARY KEY (Category_Key, Store_Key),
    FOREIGN KEY (Category_Key) REFERENCES dimProductCategory(Category_Key),
    FOREIGN KEY (Store_Key) REFERENCES dimStore(Store_Key)
);
```

2. FactProductSales

```
CREATE TABLE FactProductSales(
    Date_Key int not null,
    Product_Key int not null,
    Store_Key int not null,
    Bundle_Price decimal(10,2),
    Profit decimal(10,2),
    No_Of_Units_Sold int,
    Bundle_Size int,
    Sales decimal (10,2),
    PRIMARY KEY (Date_Key, Store_Key, Product_Key),
    FOREIGN KEY (Date_Key) REFERENCES dimTime(Date_Key),
    FOREIGN KEY (Store_Key) REFERENCES dimStore(Store_Key),
    FOREIGN KEY (Product_Key) REFERENCES dimProduct(Product_Key)
);
```

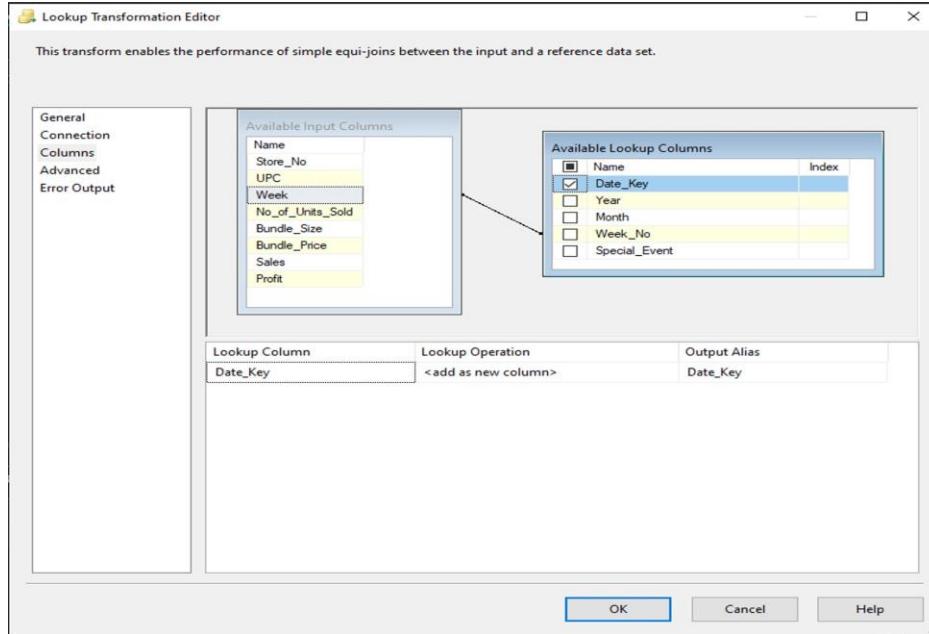
3. FactCategorySales

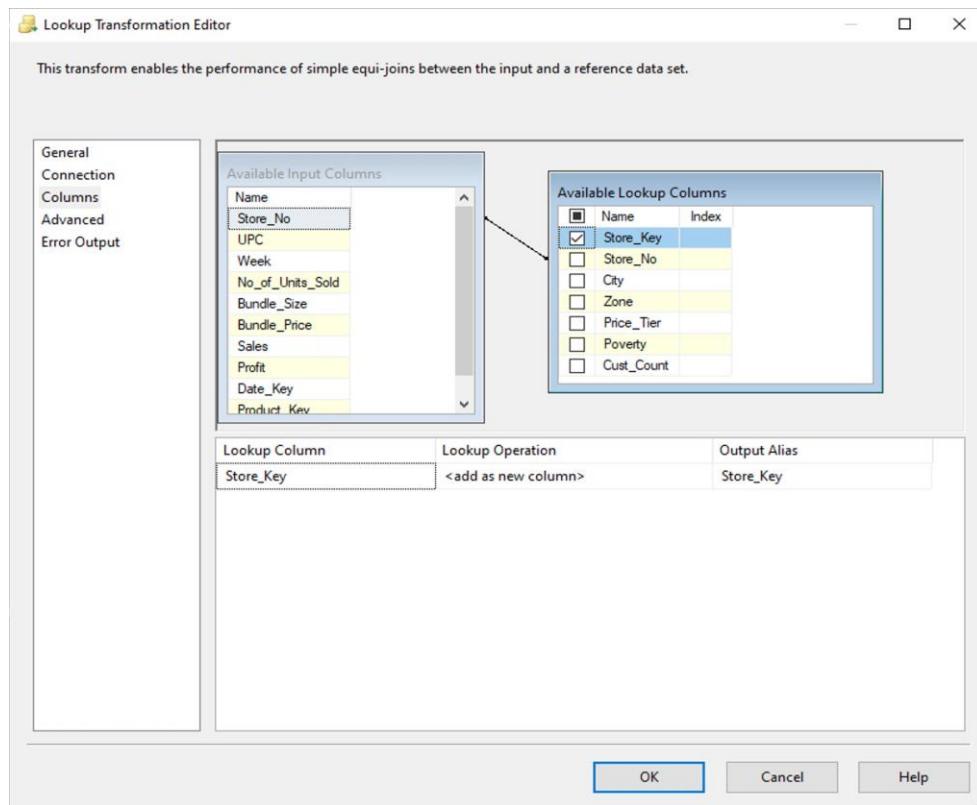
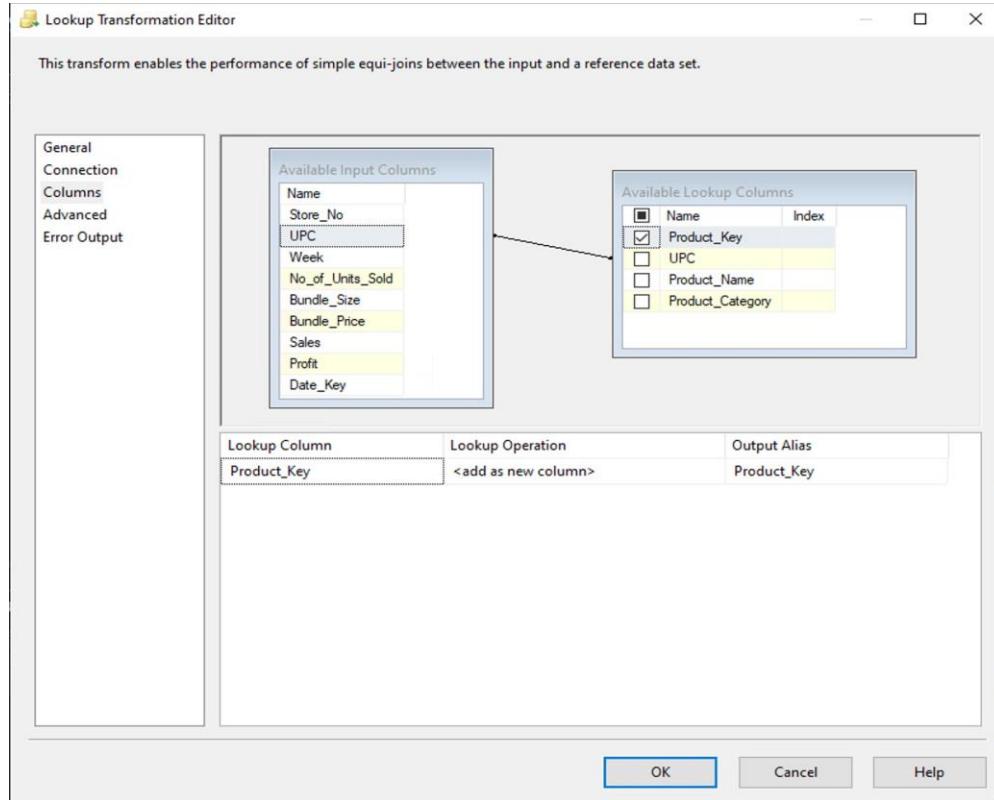
```
CREATE TABLE FactCategorySales(
    Store_Key int not null,
    Date_Key int not null,
    Category_Key int not null,
    Sales decimal (10,2),
    PRIMARY KEY (Date_Key, Category_Key, Store_Key),
    FOREIGN KEY (Date_Key) REFERENCES dimTime(Date_Key),
    FOREIGN KEY (Category_Key) REFERENCES dimProductCategory(Category_Key),
    FOREIGN KEY (Store_Key) REFERENCES dimStore(Store_Key));

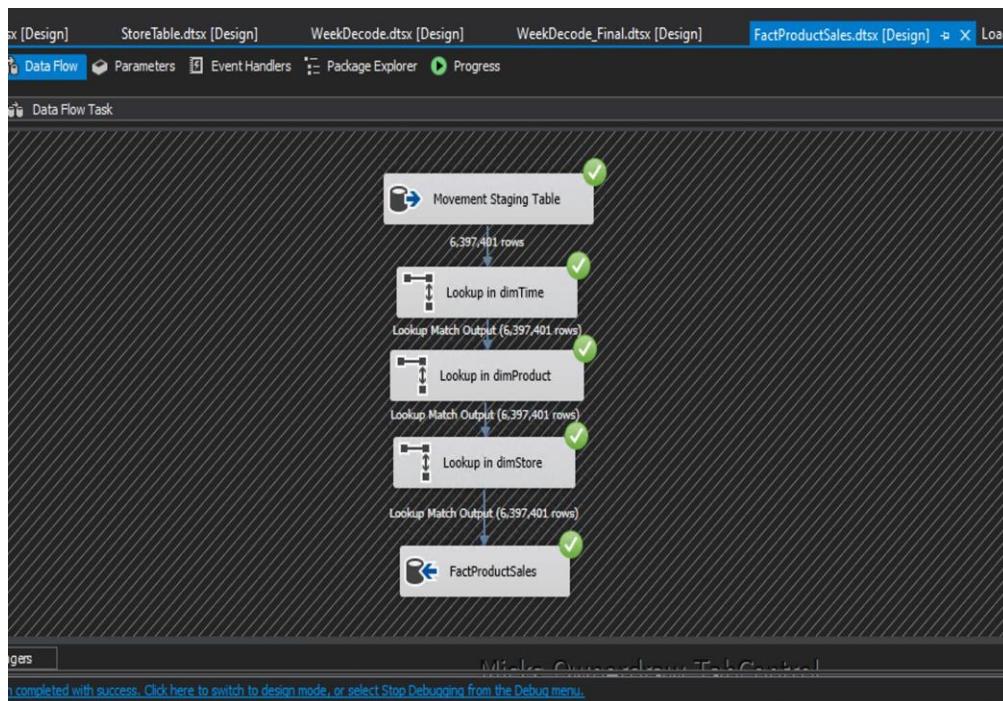
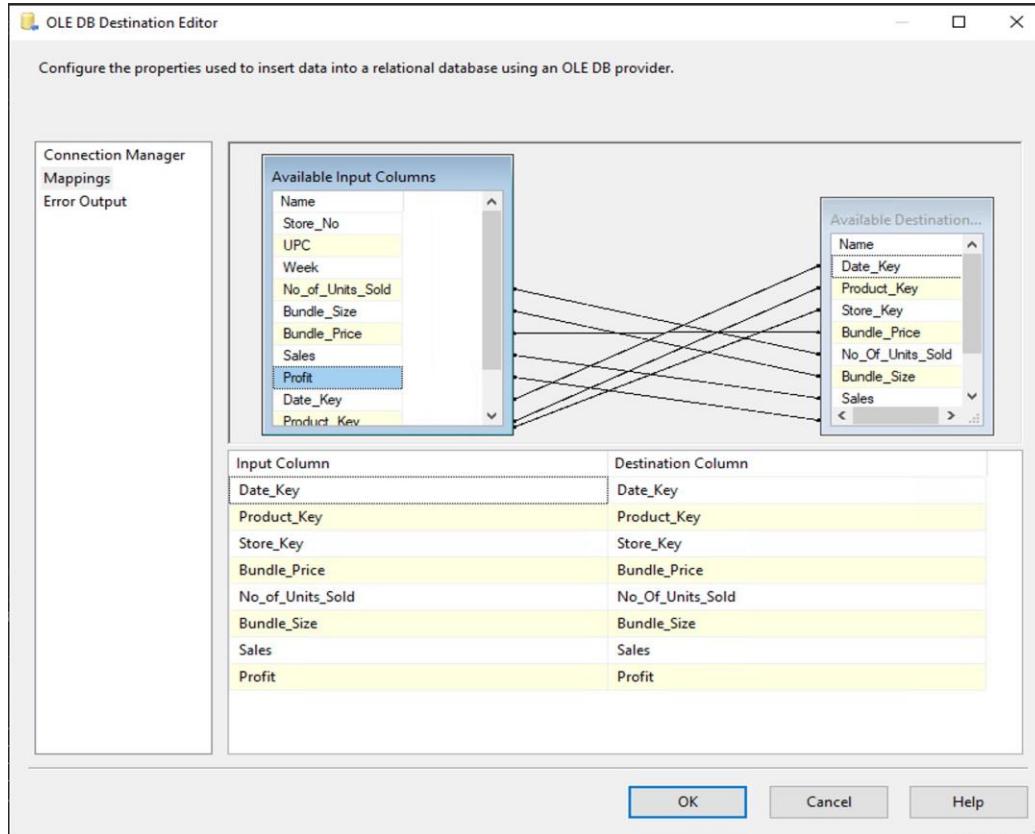
```

20.5.4 POPULATING FACT TABLES

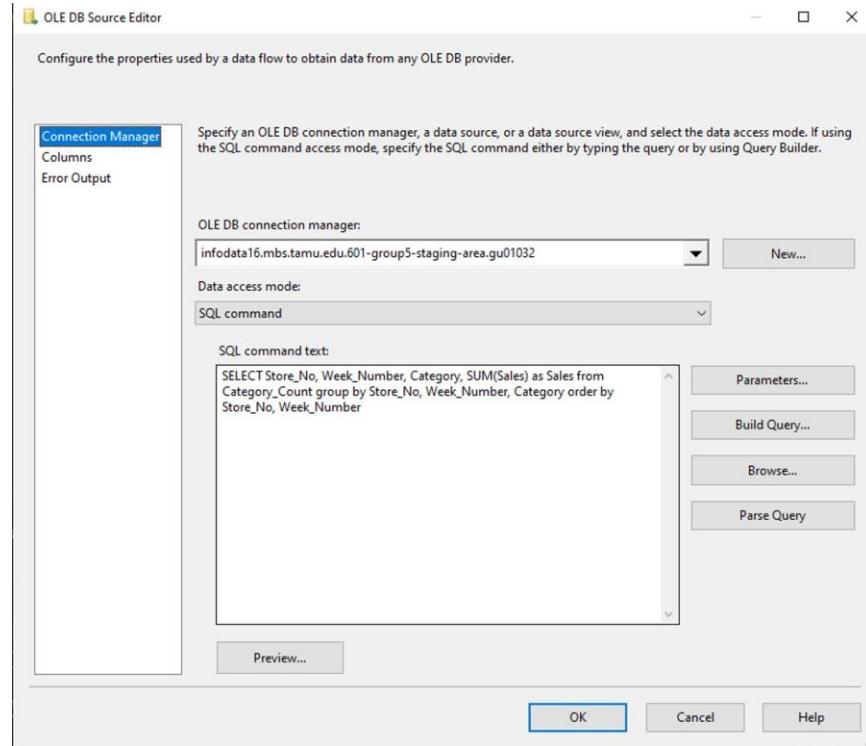
FactProductSales:

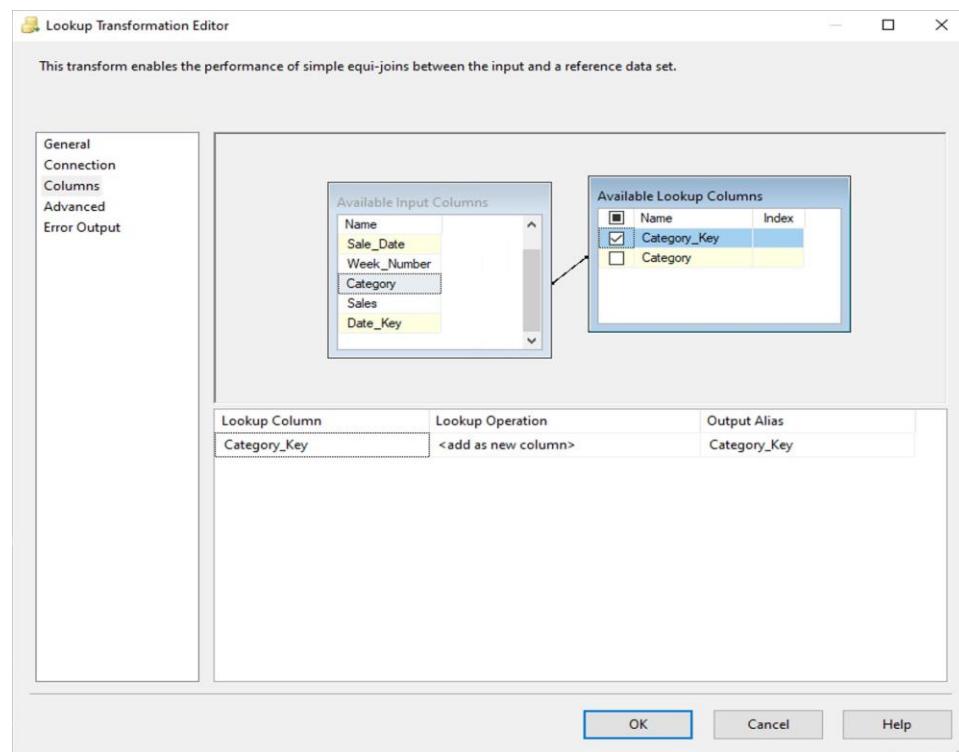
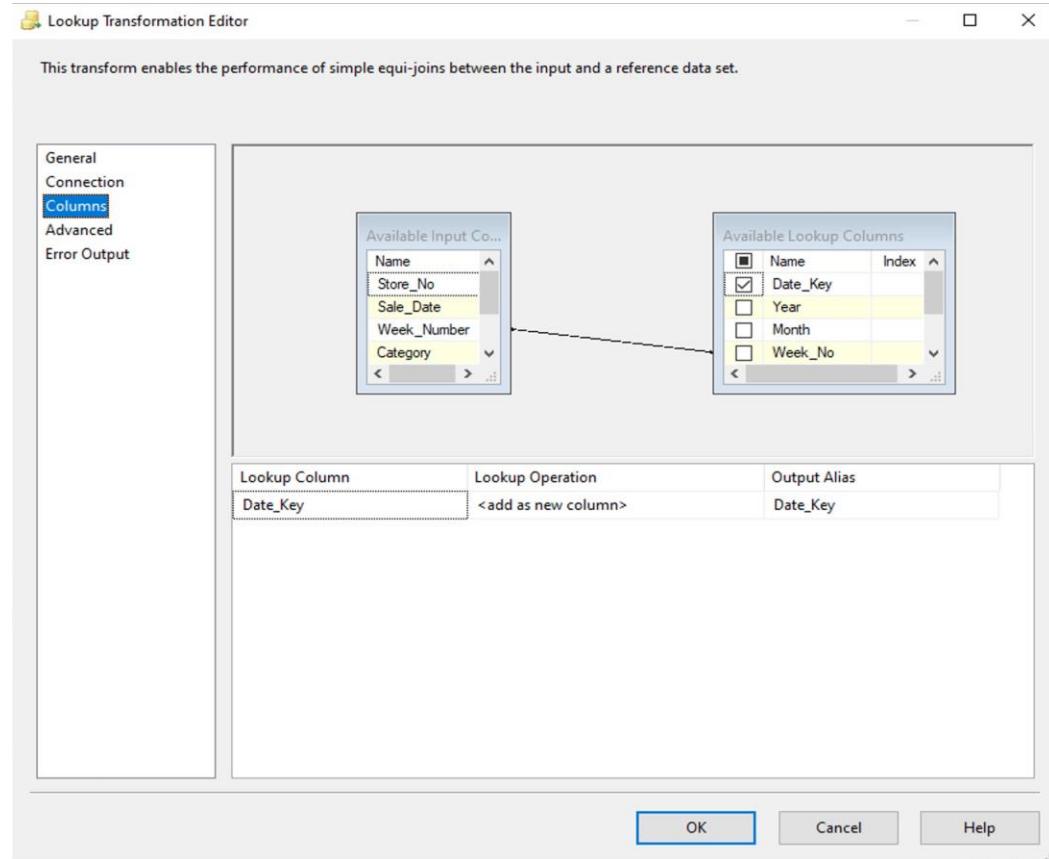


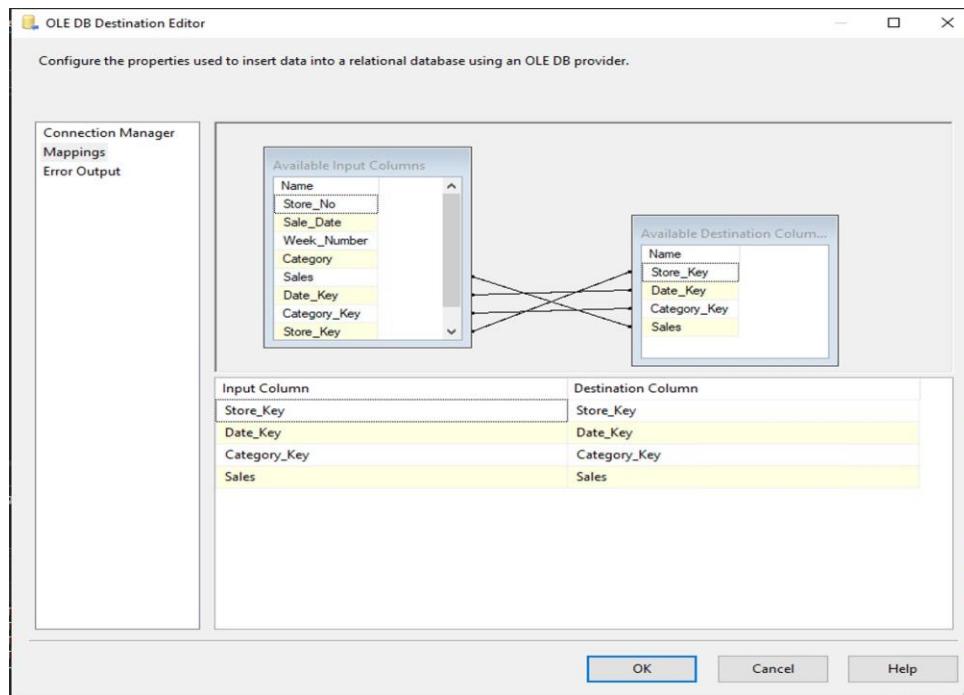
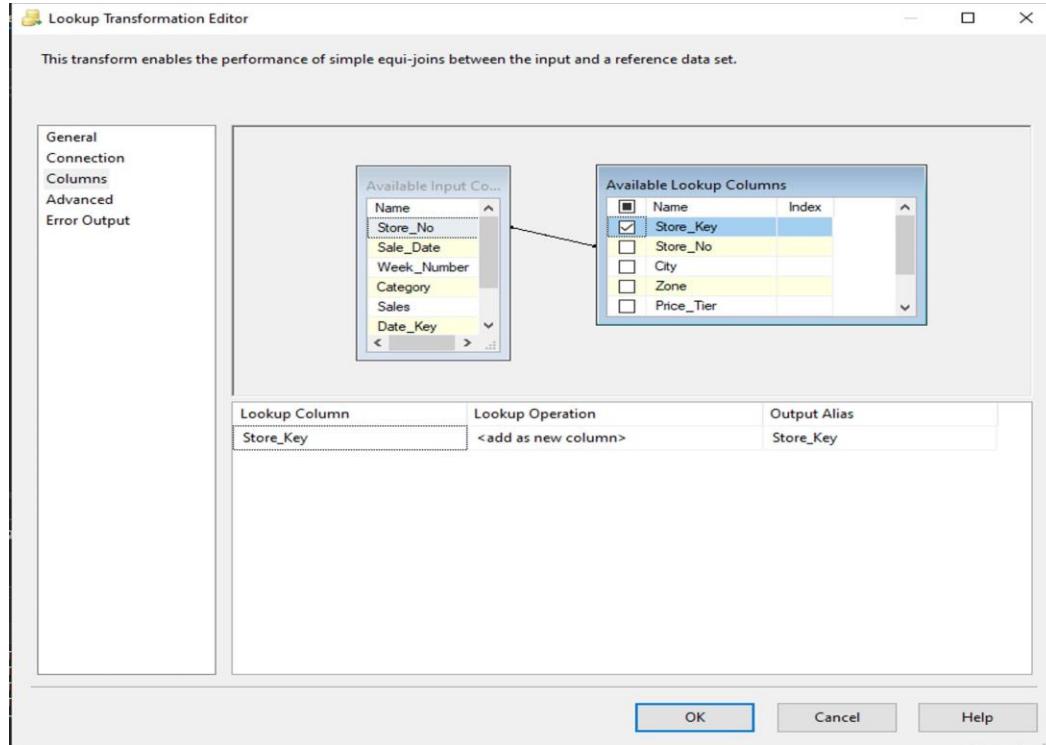


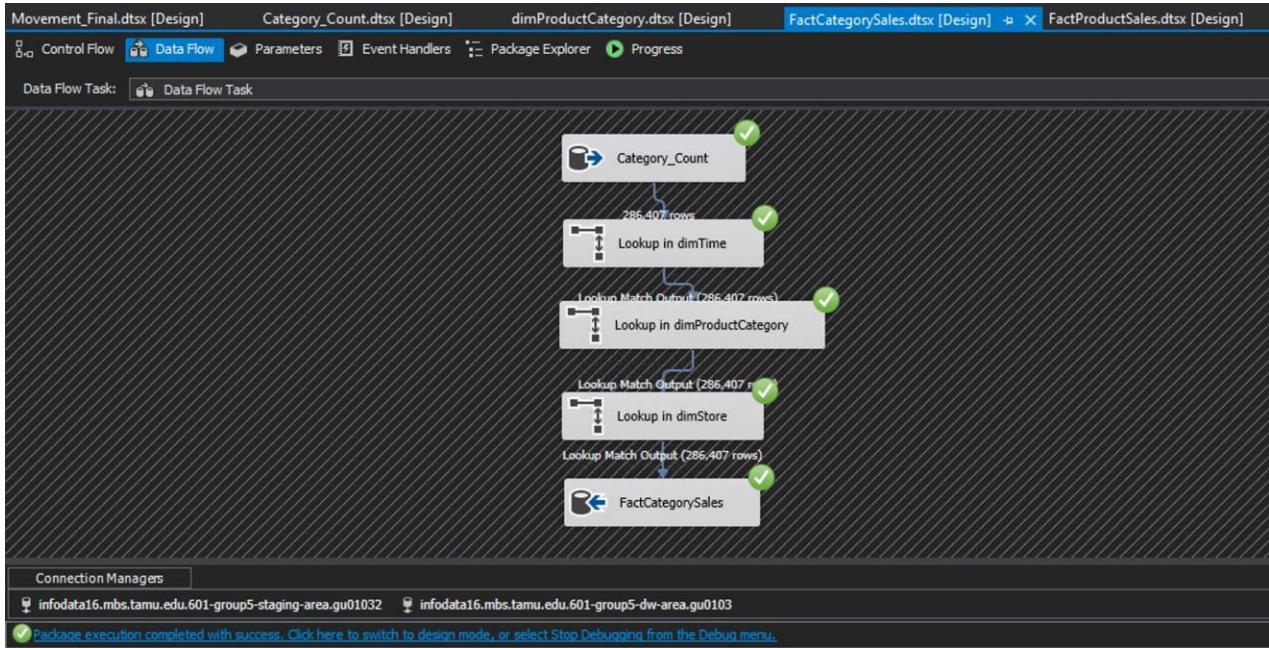


FactCategorySales:









FactStorewiseAggCategorySales:

