CONSULTING REPORT

Integrated Report 3: Requirements Analysis, Data Design and ETL Design using SSIS

Submitted By:

Group 5

Karan Gupta

Namrta Singh

Shreya Apte

TABLE OF CONTENTS

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 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

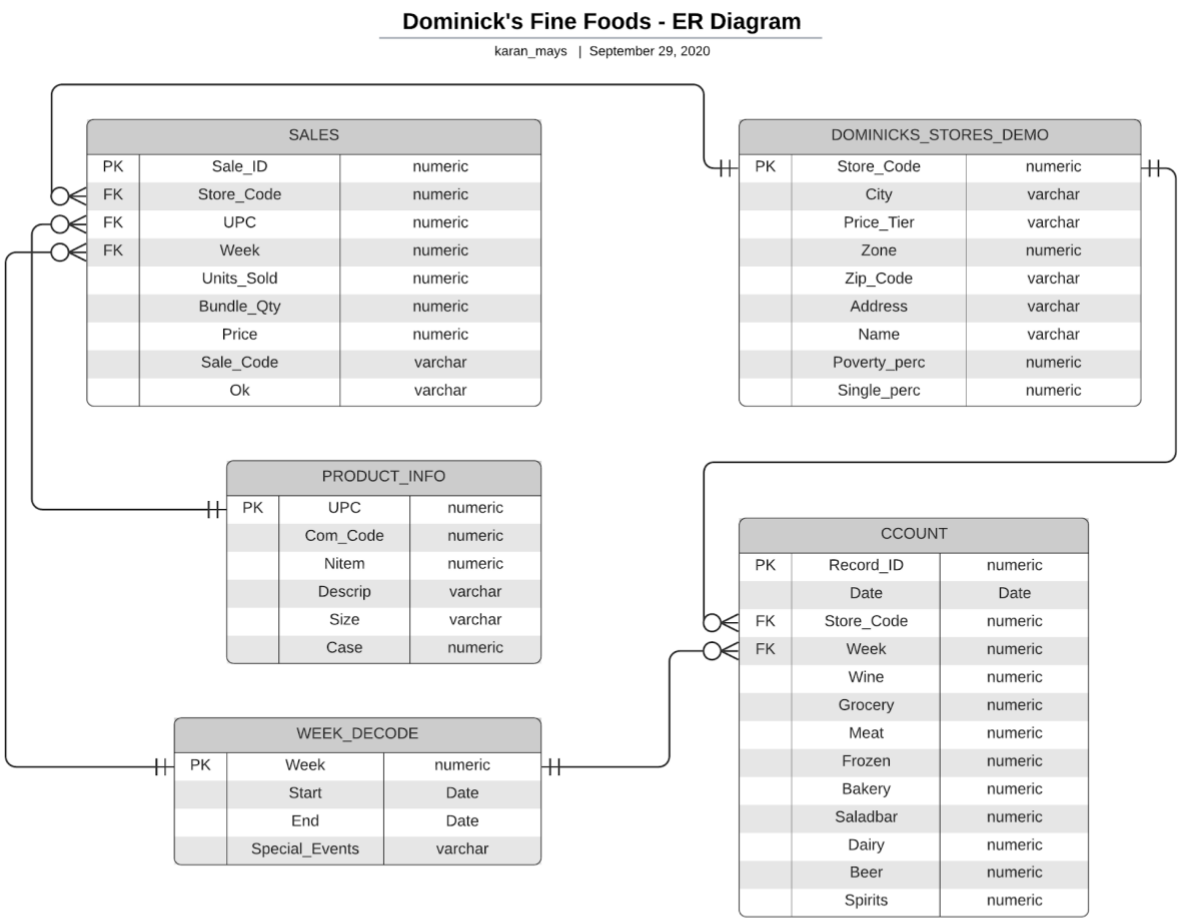
Store	City	Price Tier	Zone	Zip Code	Address
2	River Forest	High	1	60305	7501 W. North Ave.
4	Park Ridge	Medium	2	60068	Closed
5	Palatine	Medium	2	60067	223 Northwest HWY.
8	Oak Lawn	Low	5	60435	8700 S. Cicero Ave.
9	Morton Grove	Medium	2	60053	6931 Dempster
12	Chicago	High	7	60660	6009 N. Broadway Ave.

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.

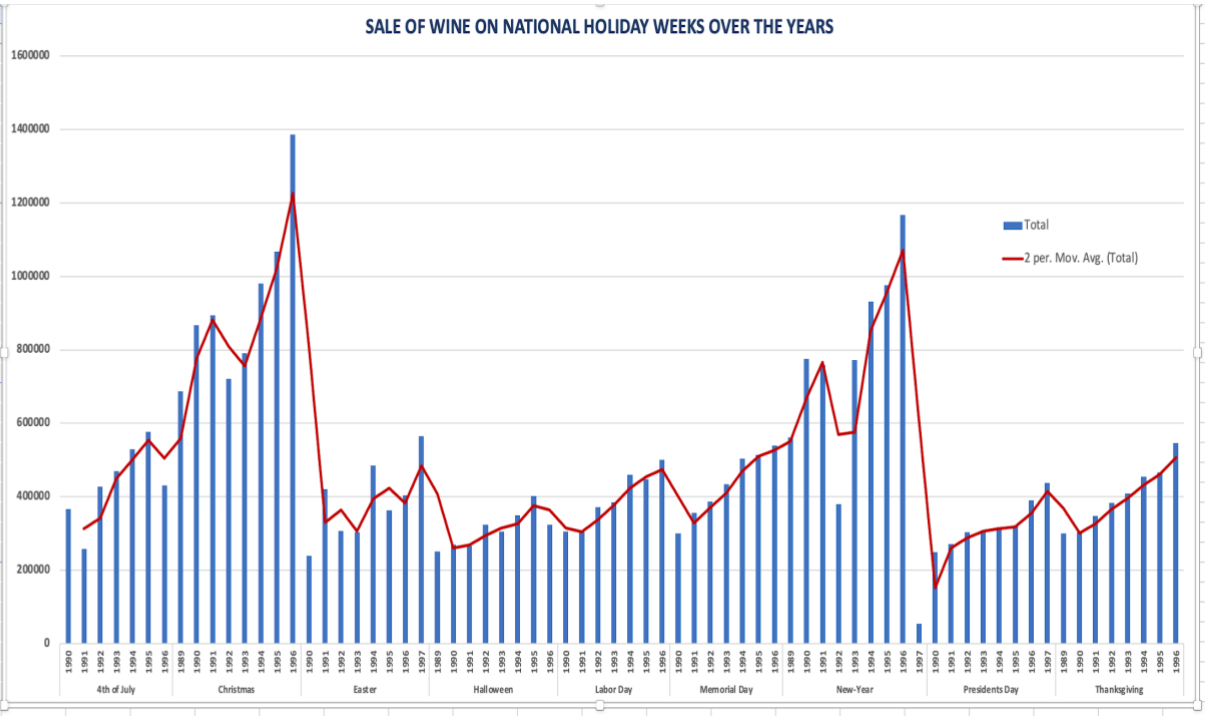
In [4], the authors assess the demand for beer treating it as a distinguished product. Various metrics like the own-price, cross-price and income elasticities for different types of beer like craft beer, imported and mass-produced beer are estimated. The analysis is performed using the detailed market 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?

Row Labels	Sum of WINE
4th of July	3059221.64
1990	366999.21
1991	258058.87
1992	426755.94
1993	470182.18
1994	529494.48
1995	576608.31
1996	431122.65
Christmas	7394635.17
1989	687024.94
1990	867237.47
1991	894678.26
1992	720824.83
1993	790082.23
1994	980689.58
1995	1066866.67
1996	1387231.19
Easter	3084105.43
1990	238864.63
1991	420812.06
1992	306440.04
1993	304045.19
1994	484375.61
1995	362122.6
1996	403082.37
1997	564362.93
Halloween	2491741.91
1989	251312.5
1990	270119.48
1991	266221.18
1992	323571.27
1993	304527.01
1994	349089.85
1995	402388.1
1996	324512.52

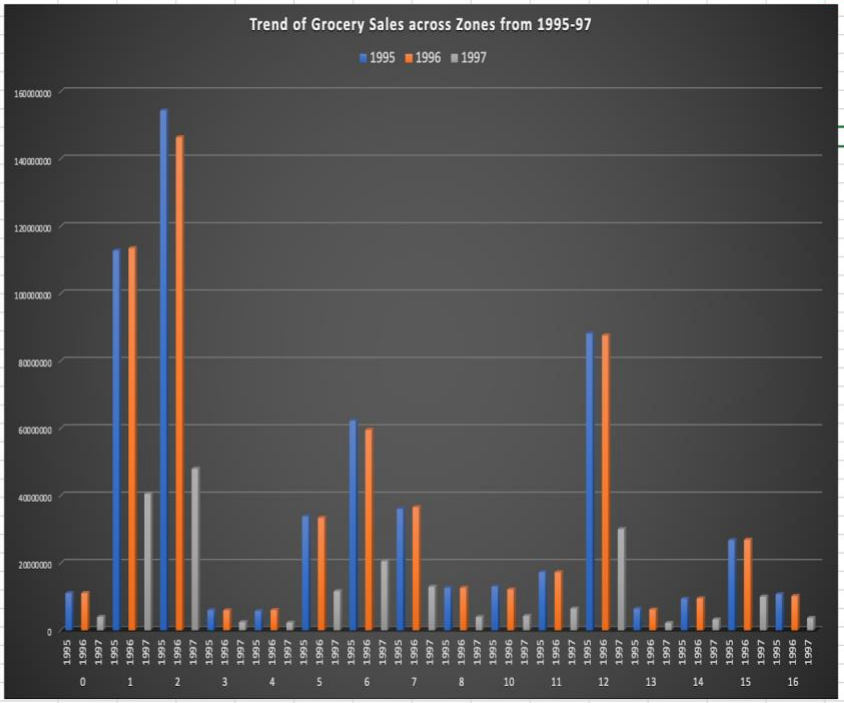


Justification:

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?

Sum of GROCERY	Column Labels			
Row Labels	1995	1996	1997	Grand Total
0	11042105.63	11070608.08	3969901.2	26082614.91
1995	11042105.63			11042105.63
1996		11070608.08		11070608.08
1997			3969901.2	3969901.2
1	112763136.2	113457426.8	40540415.49	266760978.5
1995	112763136.2			112763136.2
1996		113457426.8		113457426.8
1997			40540415.49	40540415.49
2	154349778.7	146387543	47968826.2	348706147.9
1995	154349778.7			154349778.7
1996		146387543		146387543
1997			47968826.2	47968826.2
3	5996898.39	5952936.39	2407207.61	14357042.39
1995	5996898.39			5996898.39
1996		5952936.39		5952936.39
1997			2407207.61	2407207.61
4	5722446.04	6031690.64	2290887.63	14045024.31
1995	5722446.04			5722446.04
1996		6031690.64		6031690.64
1997			2290887.63	2290887.63
5	33817497.65	33394840.17	11606831.13	78819168.95
1995	33817497.65			33817497.65
1996		33394840.17		33394840.17
1997			11606831.13	11606831.13
6	62251362.43	59531089.37	20518293.92	142300745.7
1995	62251362.43			62251362.43
1996		59531089.37		59531089.37
1997			20518293.92	20518293.92
7	36081521.64	36579766.64	12946703.88	85607992.16
1995	36081521.64			36081521.64
1996		36579766.64		36579766.64
1997			12946703.88	12946703.88
8	12623546.57	12626354.07	3983257.28	29233157.92
1995	12623546.57			12623546.57
1996		12626354.07		12626354.07

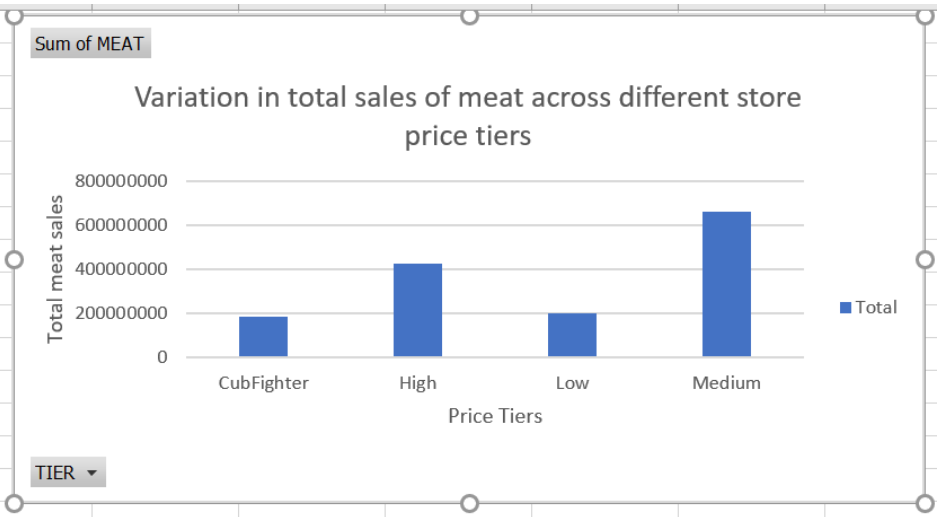


Justification:

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?

Row Labels	Sum of MEAT
CubFighter	185015464.6
High	426744862.2
Low	200867339.3
Medium	659523756.5
Grand Total	**1472151423**

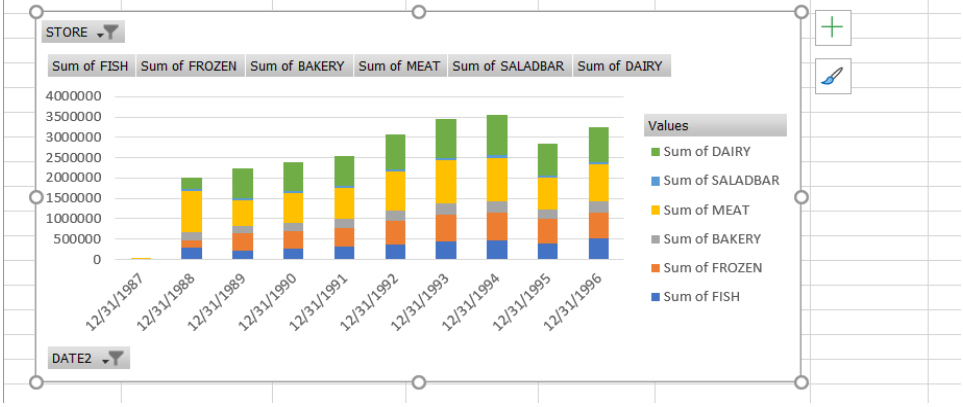


Justification:

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?

Row Labels	Sum of FISH	Sum of FROZEN	Sum of BAKERY	Sum of MEAT	Sum of SALADBAR	Sum of DAIRY
12/31/1987	1676.84	0	1661.68	9544.26	0	0
12/31/1988	294245.54	173082.41	208913.79	1011612.77	50458.21	270149.81
12/31/1989	223243.73	425954.99	172052.66	641952.38	47002.45	719691.9
12/31/1990	270253.46	425965.99	197741.86	728326.42	49106.19	726991.22
12/31/1991	323719.82	458733.39	212867.53	768160.12	53230.33	734069.3
12/31/1992	368970.33	578903.69	259257.1	945334.59	51969.22	877194.72
12/31/1993	438374.12	656000.91	277270.75	1073254.29	54835.81	959588.65
12/31/1994	462905.87	681847.03	286992.11	1069146.06	53454.28	989677.05
12/31/1995	401987.71	595006.34	229853.62	779168.82	45900.45	802292.31
12/31/1996	531904.68	613055.28	290026.48	904648.02	51964.56	854939.11
Grand Total	**3317282.1**	**4608550.03**	**2136637.58**	**7931147.73**	**457921.5**	**6934594.07**

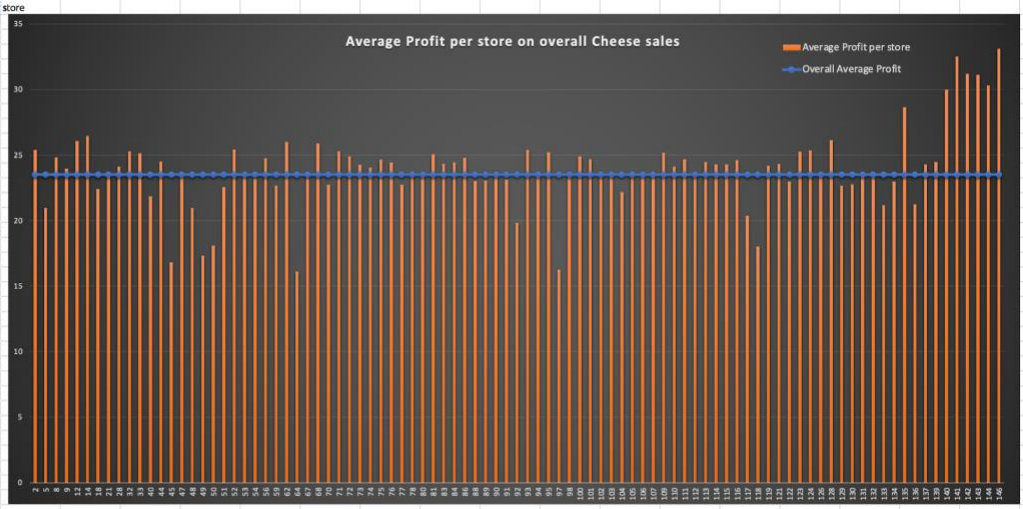


Justification:

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?

Row Labels	Overall Averag	Average Profit per store
2	23.5336646	25.40996419
5	23.5336646	20.95357182
8	23.5336646	24.84041881
9	23.5336646	23.97718267
12	23.5336646	26.08683442
14	23.5336646	26.47578989
18	23.5336646	22.42260785
21	23.5336646	23.57559239
28	23.5336646	24.10605116
32	23.5336646	25.28037869
33	23.5336646	25.15312674
40	23.5336646	21.84553619
44	23.5336646	24.49933956
45	23.5336646	16.82136863
47	23.5336646	23.31170598
48	23.5336646	20.96614961
49	23.5336646	17.32430679
50	23.5336646	18.1035268
51	23.5336646	22.5676622
52	23.5336646	25.43936591
53	23.5336646	23.59423299
54	23.5336646	23.22377441
56	23.5336646	24.76311349
59	23.5336646	22.66110866
62	23.5336646	25.99564105
64	23.5336646	16.12747344
67	23.5336646	23.65636319
68	23.5336646	25.89767606
70	23.5336646	22.73351551
71	23.5336646	25.27642771
72	23.5336646	24.91269318
73	23.5336646	24.25049335
74	23.5336646	24.0580721
75	23.5336646	24.64515906
76	23.5336646	24.44555291

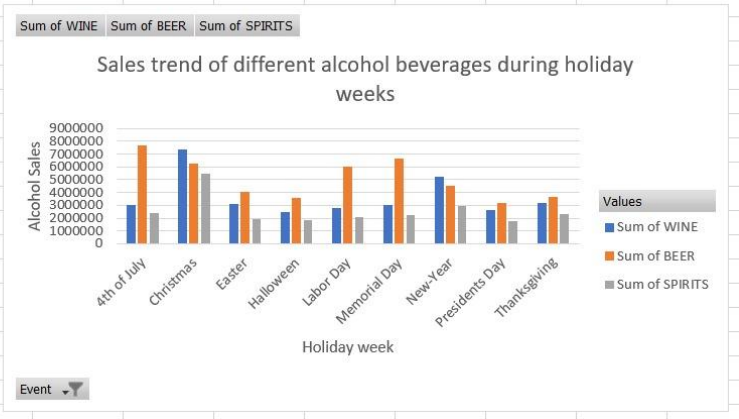


Justification:

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?

Holiday Wee	Sum of WINE	Sum of BEER	Sum of SPIRITS
4th of July	3059221.64	7680806.33	2416076.19
Christmas	7394635.17	6255876.38	5450273.23
Easter	3084105.43	4017635	1901863.7
Halloween	2491741.91	3549681.83	1872672.59
Labor Day	2773667.7	6014038.27	2121624.65
Memorial Day	3032160.82	6663225.79	2233103.04
New-Year	5255273.03	4548389.22	2933999.5
Presidents Day	2596464.73	3214189.59	1764741.39
Thanksgiving	3212501.54	3625757.46	2341215.85
Grand Total	**32899771.97**	**45569599.87**	**23035570.14**



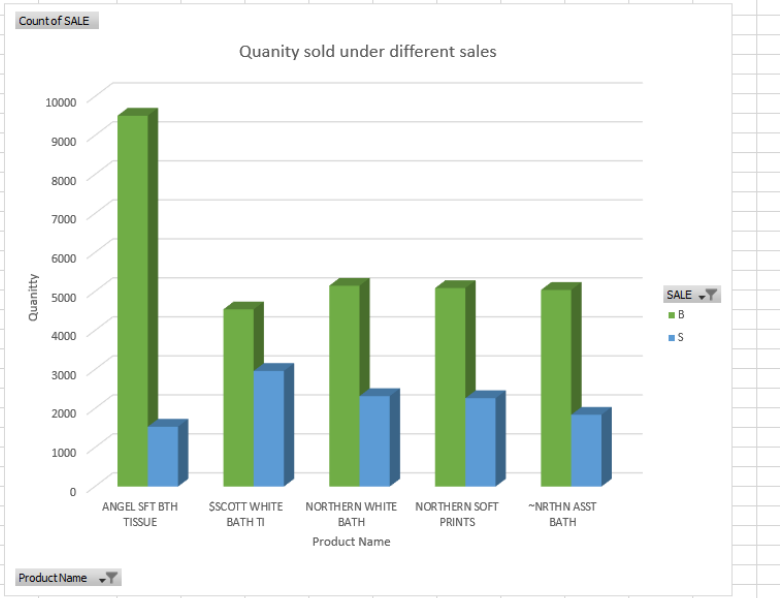
Justification:

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)?

Count of SALE	Column Labels		
Row Labels	B	S	Grand Total
ANGEL SFT BTH TISSUE	9509	1535	11044
SSCOTT WHITE BATH TI	4543	2964	7507
NORTHERN WHITE BATH	5149	2319	7468
NORTHERN SOFT PRINTS	5089	2265	7354
~NRTHN ASST BATH	5040	1841	6881
Grand Total	29330	10924	40254

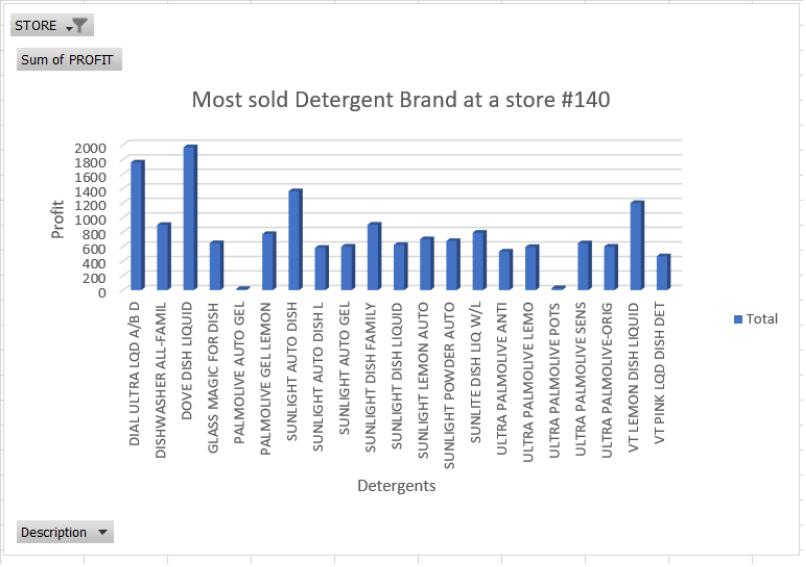


Justification:

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?

STORE	140
Row Labels	**Sum of PROFIT**
DIAL ULTRA LQD A/B D	1758.88
DISHWASHER ALL-FAMIL	899.47
DOVE DISH LIQUID	1968.79
GLASS MAGIC FOR DISH	648.04
PALMOLIVE AUTO GEL	15.15
PALMOLIVE GEL LEMON	773.64
SUNLIGHT AUTO DISH	1364.07
SUNLIGHT AUTO DISH L	583.17
SUNLIGHT AUTO GEL	599.4
SUNLIGHT DISH FAMILY	902.17
SUNLIGHT DISH LIQUID	622.09
SUNLIGHT LEMON AUTO	702.58
SUNLIGHT POWDER AUTO	677.55
SUNLITE DISH LIQ W/L	791.32
ULTRA PALMOLIVE ANTI	531.2
ULTRA PALMOLIVE LEMO	593.78
ULTRA PALMOLIVE POTS	27.09
ULTRA PALMOLIVE SENS	645.82
ULTRA PALMOLIVE-ORIG	600.28
VT LEMON DISH LIQUID	1200.65
VT PINK LQD DISH DET	467.6
Grand Total	**16372.74**

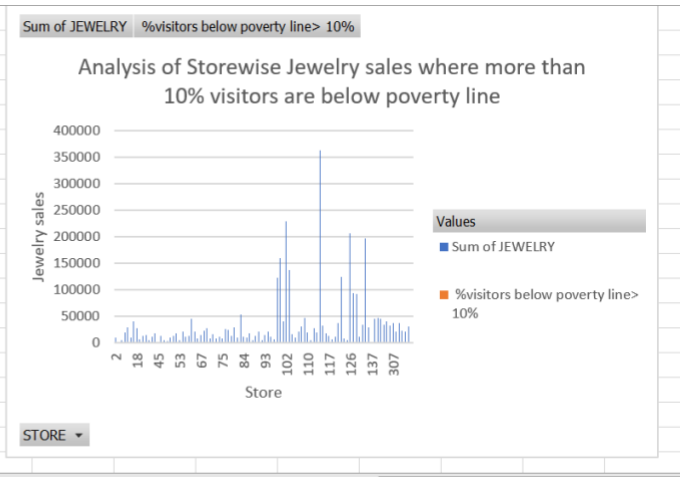


Justification:

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?

Store	Sum of JEWELRY	%visitors below poverty line> 10%
2	10575.4	0.0619594
4	0	0.037592397
5	4805.83	0.025104603
8	19108.58	0.05135254
9	28763.28	0.028060483
12	10166.82	0.168864097
14	41004.25	0.019350496
18	27376.59	0.075727381
21	7549.31	0.024785242
28	13332.94	0.035497431
32	15255.25	0.037700865
33	5095.99	0.087538074
40	10945.91	0.055184805
44	17981.43	0.030154849
45	921.41	0.022827393
47	13293.96	0.032623559
48	5288.57	0.018204313
49	3709.28	0.028320971



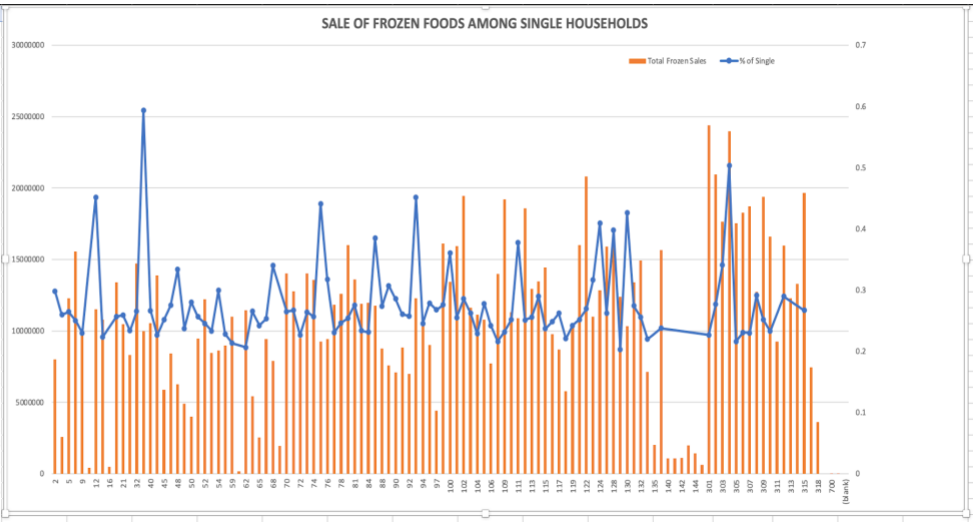
Justification:

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?

Row Labels	% of Single	Total Frozen Sales
2	0.297620974	8002733.99
4	0.259745235	2563399.64
5	0.26458527	12286261.56
8	0.250372149	15558883.26
9	0.230005194	9706555.45
10	#N/A	425105
12	0.451637338	11524995.1
14	0.223351772	10768335.22
16	#N/A	478829
18	0.256403079	13390134.97
21	0.258991055	10481303.89
28	0.233601718	8301303.41
32	0.265329599	14719502.32
33	0.593475533	9976360.42
40	0.266478451	10546652.53
44	0.226447947	13890861.58
45	0.2516	5879364.35
47	0.275212381	8433552.45
48	0.334087413	6261364.68
49	0.237270004	4899158.01
50	0.280109682	3991966.89
51	0.256408379	9471654.51
52	0.245364813	12227982
53	0.233270677	8436034.23
54	0.299425563	8612661.97
56	0.228409181	8973656.32
59	0.213364348	11009567.18
61	#N/A	169751
62	0.206418161	11455825.67
64	0.265272875	5414491.56
65	0.242001641	2536048.53



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.

1. 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	

6. HIERARCHICAL DIMENSION AND FACT TABLES

6.1 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:

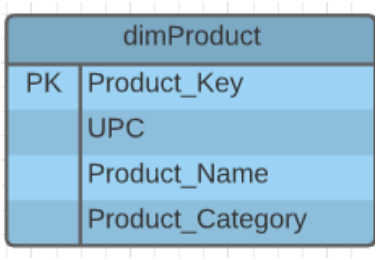
1. dimStore

- **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

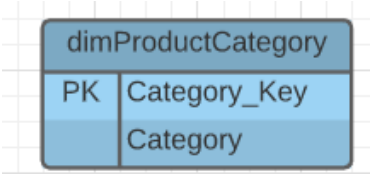
- **Date_Key-** A unique identifier of the time dimension table
- **Year-** The year in which the sales were recorded
- **Month -** The month in which the sales were recorded
- **Week_No-** The week in the year when sales have occurred
- **Special_Event-** The identifier to indicate the occurrence of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

3. dimProduct



- **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



- **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.

6.2 FACT TABLES IN HIERARCHY

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.

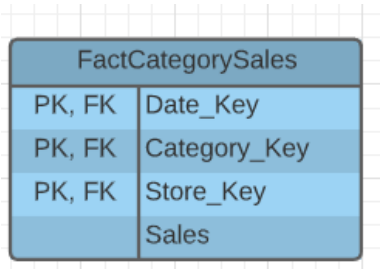
Fact Table Keys:

- **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.



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

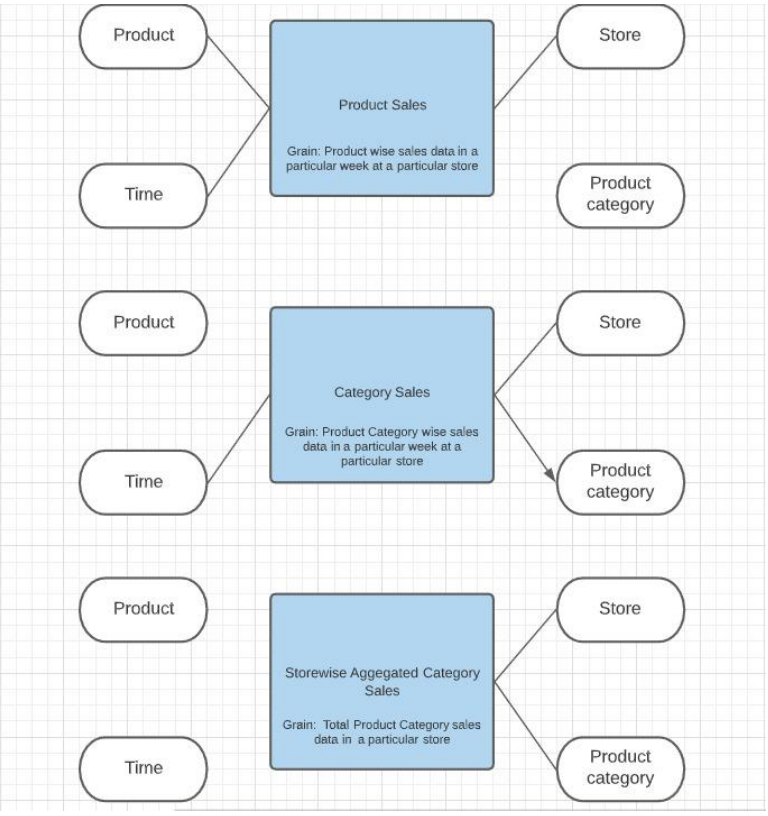
Fact Table Keys:

- **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.

1. Fact Table Diagrams

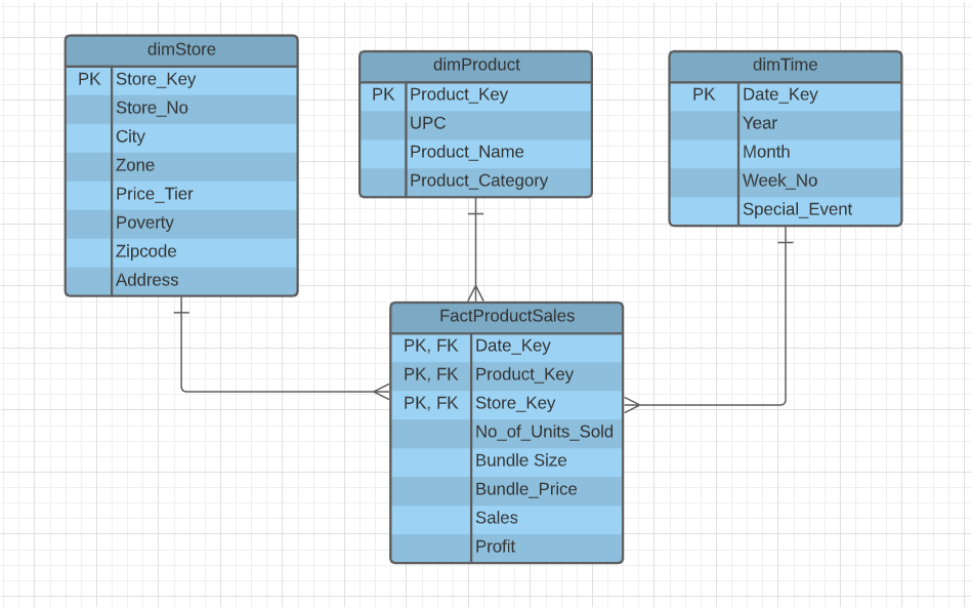


7. DIMENSIONAL MODELLING (STAR SCHEMA)

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.

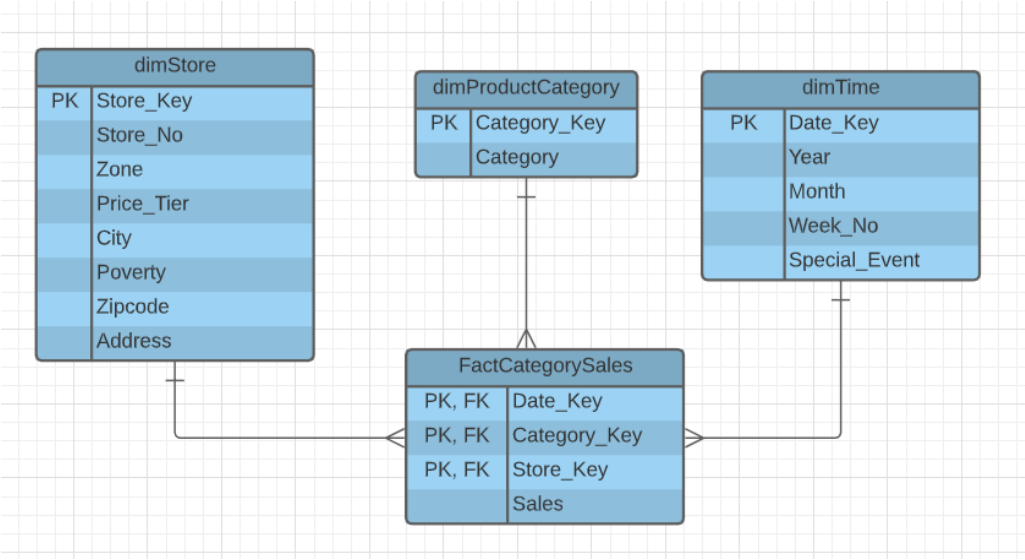
7.1. PRODUCT SALES DATA MART

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



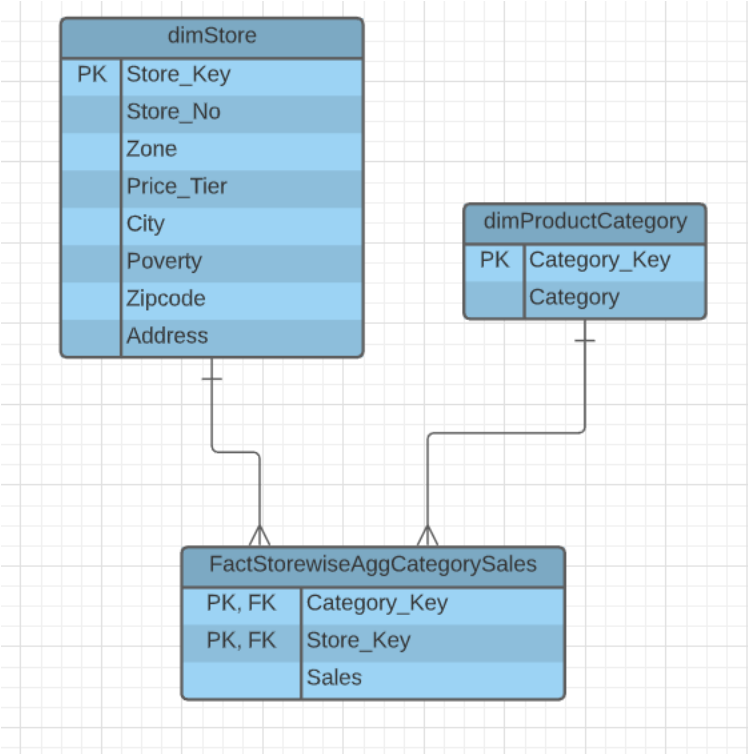
7.2. PRODUCT CATEGORY SALES DATA MART

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.



7.3. 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.



8. 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		**FactStorewiseAggCategorySales**		
			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	

9. 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.

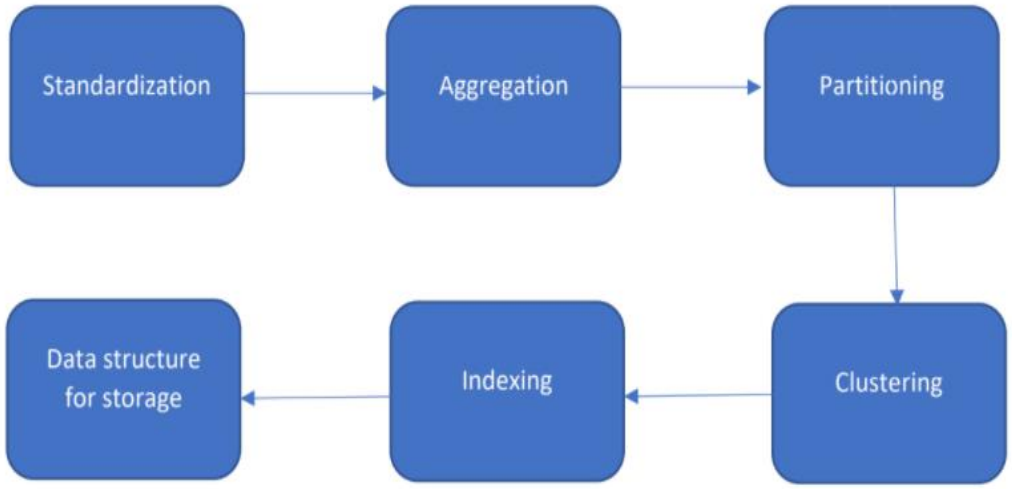
Q5. WHICH ARE THE BEST PERFORMING STORES FOR JEWELRY SALES WHERE MORE THAN 10% OF VISITORS ARE BELOW THE POVERTY LINE?

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.

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

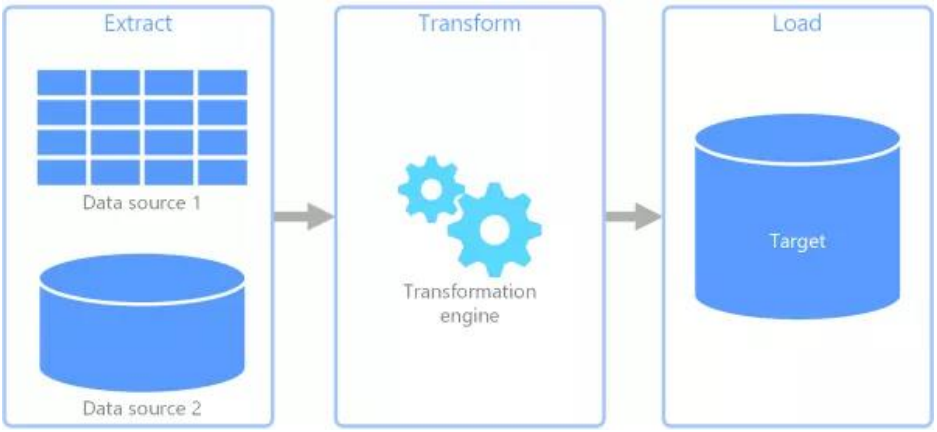
After deciding the strategies of aggregating, partitioning, clustering, and indexing, we will concentrate on how to physically store the data on the storage unit, how to break the files into different blocks to ensure maximum performance. To store the data for successful data retrieval, consideration 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. 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.



11.1 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.

11.2 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.

11.3 DATA LOADING

The following steps were taken for Data Loading:

12 DATA SOURCES AND TARGET DATA

12.1 TARGET DATA NEEDED IN THE WAREHOUSE

Data Source	Source File Name	Staging Area Tables	Warehouse Tables
Demographics	Demographics.csv	Demographics_tmp, Store	dimStore
Price Tier	DFF Store Price Tier File.xls	PriceTier_tmp, Store	
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

12.2 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

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

12.3.1 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

12.3.2 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_Catetgory	

Category_Count		**dimProductCategory**	Category_Key	An auto-incremental Surrogate Key generated to be used as the Primary Key
	Category		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_Count		**FactStorewiseAggCategorySales**	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	(Sales recorded by grouping the sales from Category_Count by Store, Catetgory)

13 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'

14 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:

14.1 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.

14.2 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.

14.3 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.

14.4 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.

14.5 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
- Catetgory_Key - Unique Identifier for the Product Category (those categories for which UPC level data is unavailable) Table

14.6 CREATION OF DERIVED COLUMNS

For meeting the requirements of the business questions a few columns were required to be derived from existing attributes. Sch 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 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

14.7 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

15 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.

16 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

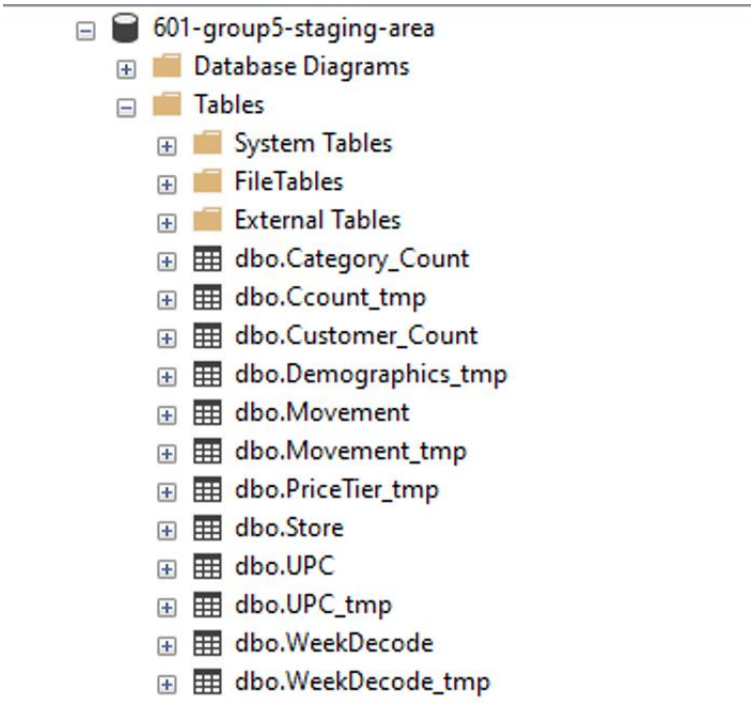
which we have extracted the data, as is from the provided source files, using SSIS Import Export tasks. After extraction, data in these temp tables is cleansed, transformed and then the clean data is stored inside the other type of tables, which are our final staging tables.

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'.



17 PROCEDURES FOR EXTRACTION AND LOADING

17.1 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.

17.2 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

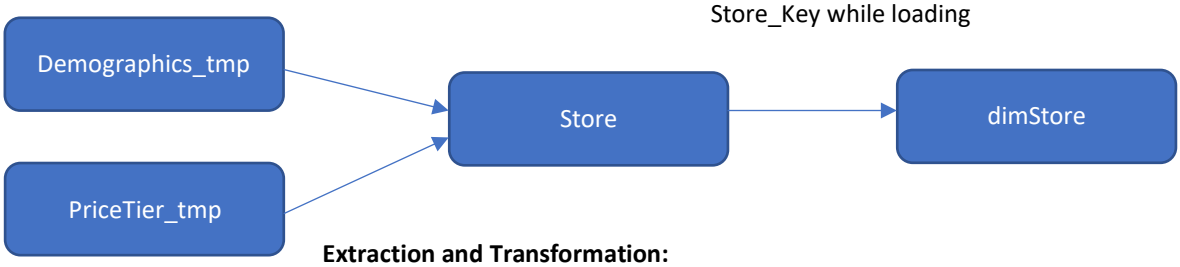
- FactProductSales
  i. Created the Product level Sales Fact table – FactProductSales in the database – 601-group5-dw-area
  ii. This table stores sales for a Product (UPC), on a particular week, in a particular store
  iii. Data from Movement Staging table is loaded in this fact table
  iv. While loading, 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

18 ETL FOR DIMENSION AND FACT TABLES

18.1 ETL FOR DIMENSION TABLES

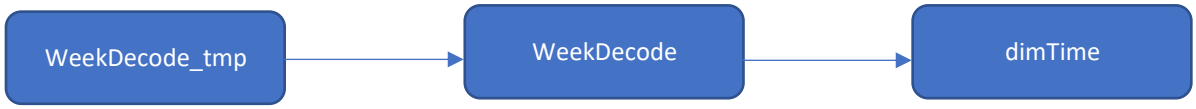
1. Store Dimension Table

Loading: Added a surrogate key column - Store_Key while loading



Extraction and Transformation: Left join between Demographics_tmp and PriceTier_tmp; conversion of fields to appropriate data types

2. Date Dimension Table



Extraction and Transformation: Loaded data from csv in temp, created derived columns Month and Year, converted data types

Loading: Added a surrogate key column – Date_Key while loading

3. Product Dimension Table

Extraction and Transformation: Loaded data from UPCLND.csv to UPC_tmp, cleaned Product Description data

Loading: Added a surrogate key column – Product_Key while loading

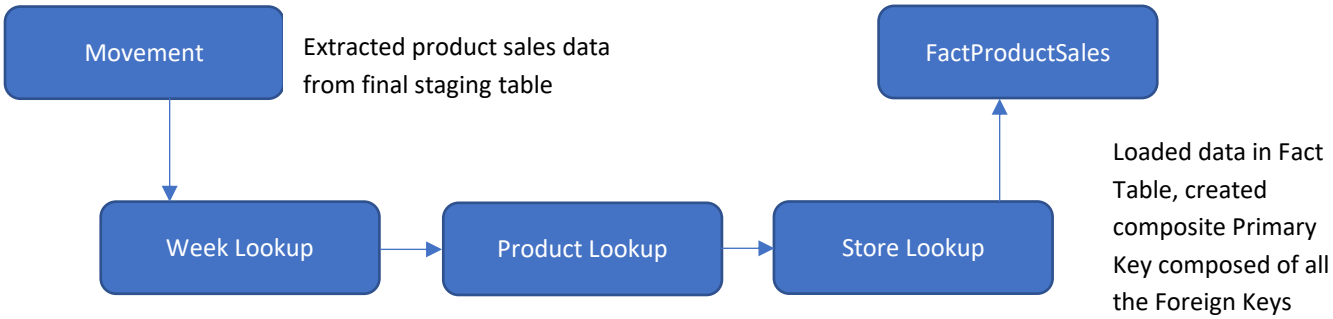
4. Product Category Dimension Table

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

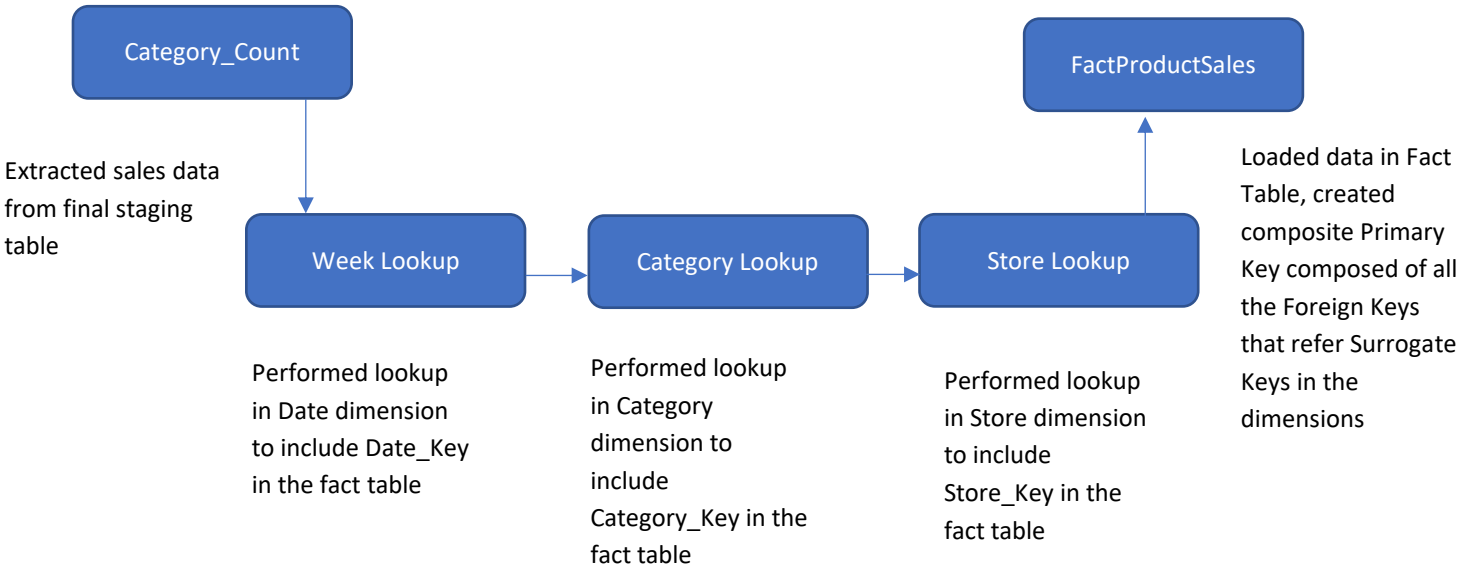
Loading: Loaded data for Categories, added a surrogate key column – Category_Key while loading

18.2 ETL FOR FACT TABLES

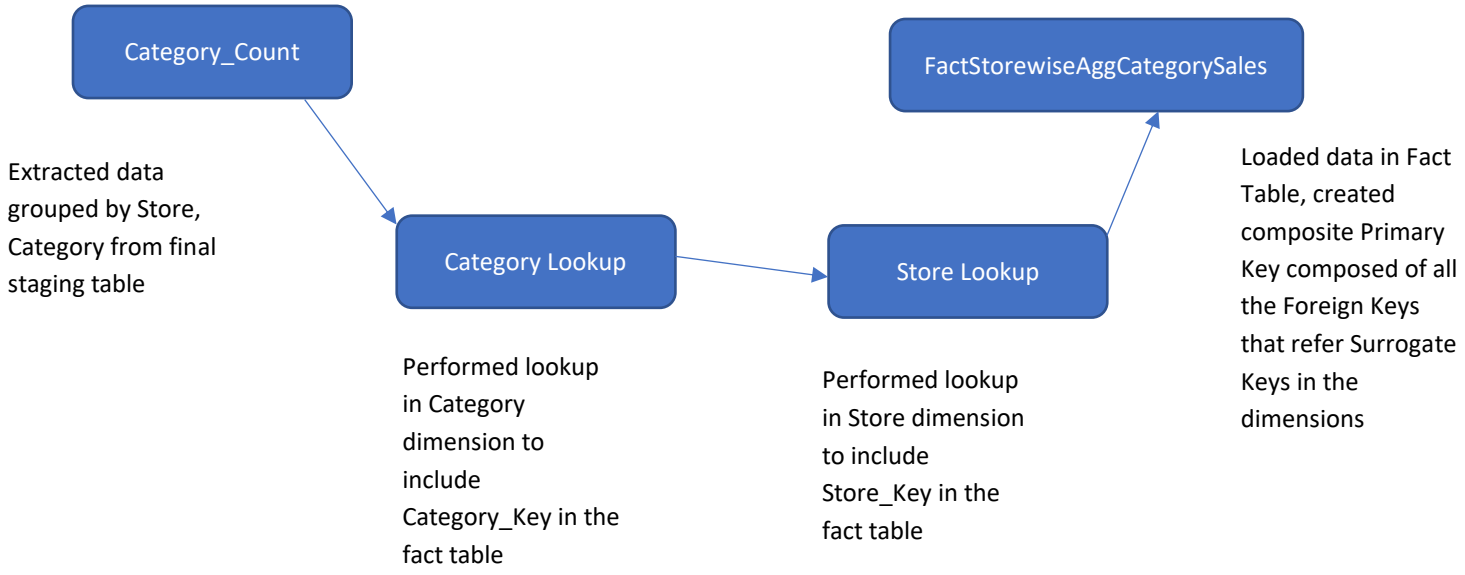
1. Product Sales Fact Table



2. Category Sales Fact Table



3. Storewise Aggregated Category Sales Fact Table



19 IMPLEMENTATION OF ETL

19.1 EXTRACTION OF DATA

19.1.1 TEMPORARY TABLE CREATION

Demographics Temp Staging Table

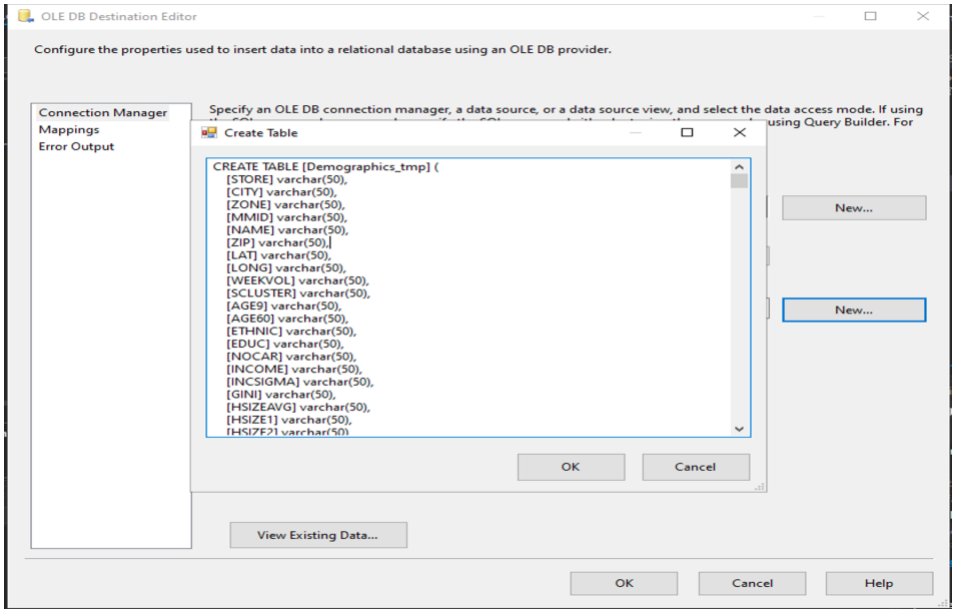


Figure 1:Demographic Temporary Table Creation

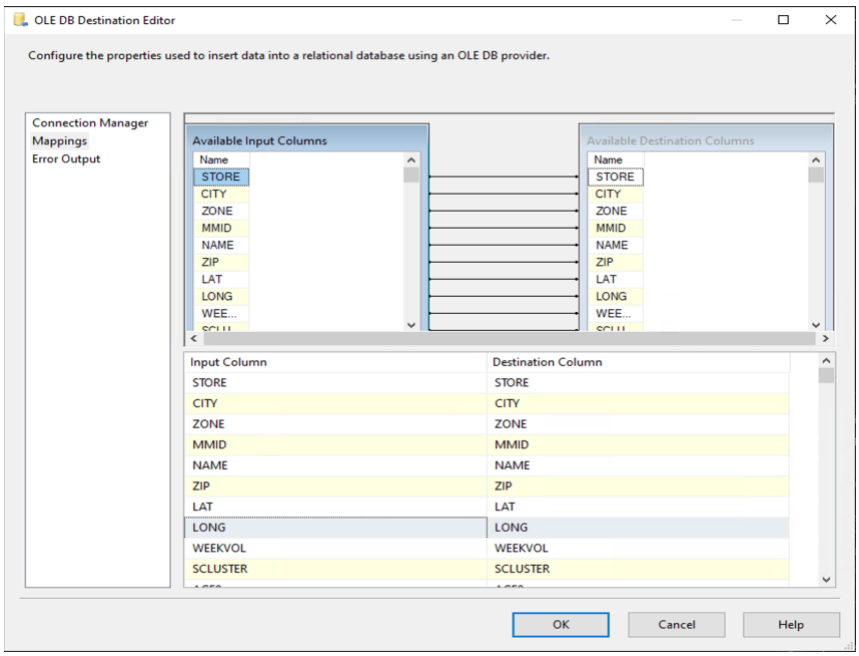


Figure 2: Source to Staging Mapping

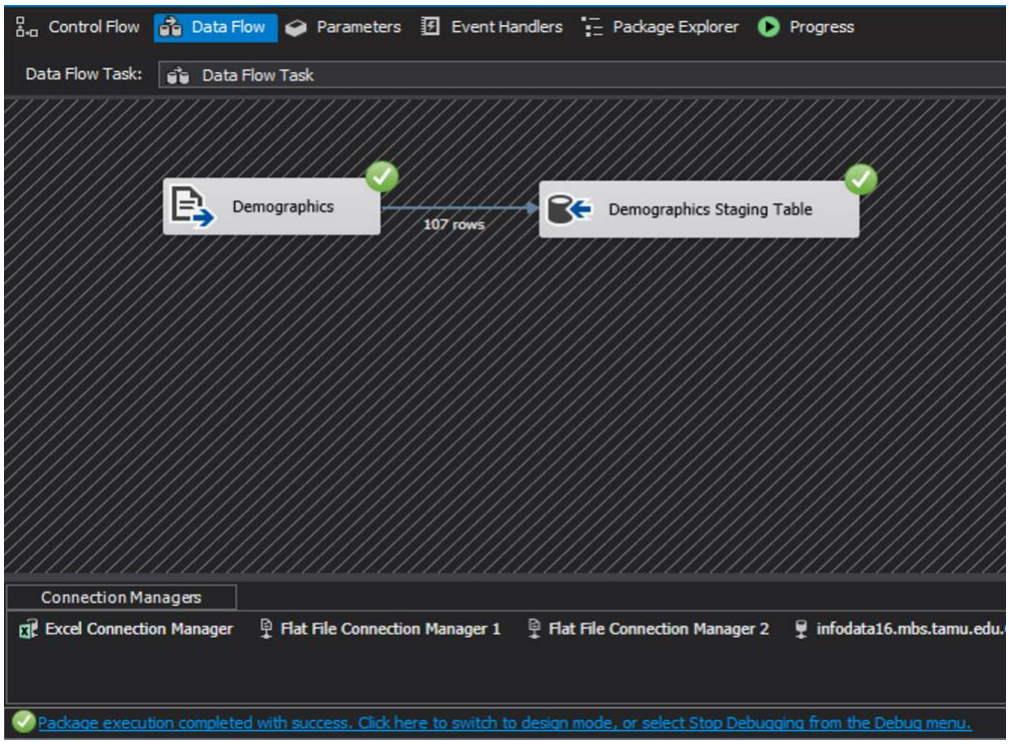
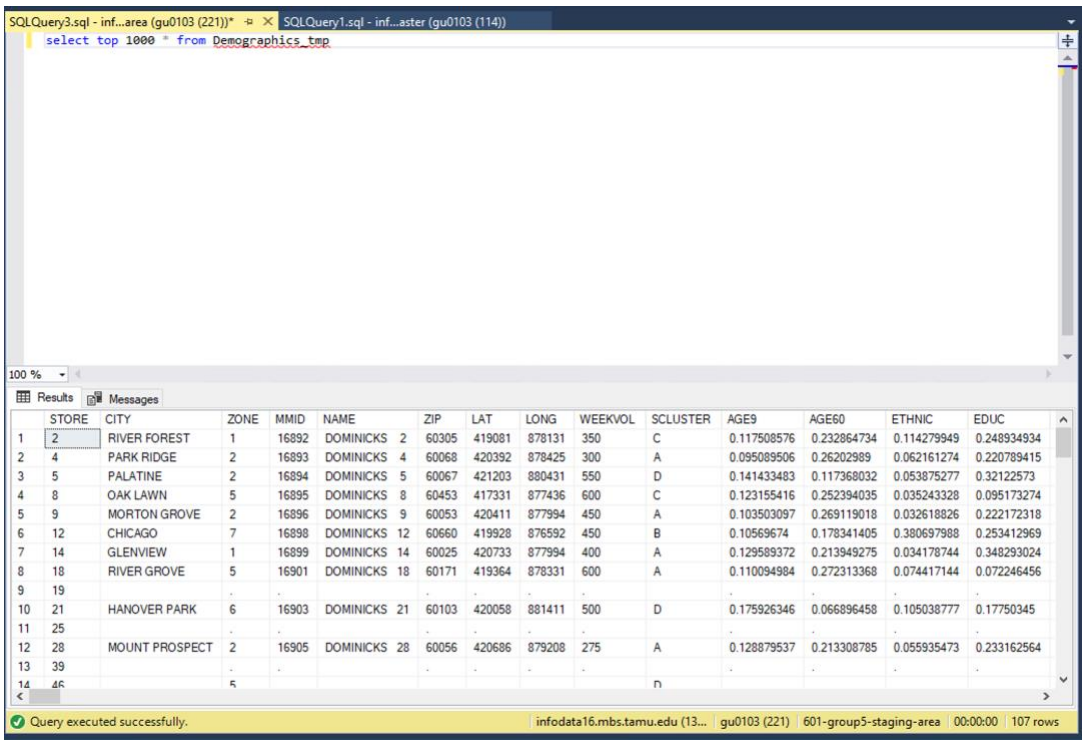


Figure 3: Data Flow



	STORE	CITY	ZONE	MMID	NAME	ZIP	LAT	LONG	WEEKVOL	SCLUSTER	AGE9	AGE60	ETHNIC	EDUC
1	2	RIVER FOREST	1	16892	DOMINICKS	2	60305	419081	878131	350	C	0.117508576	0.232864734	0.114279949	0.248934934
2	4	PARK RIDGE	2	16893	DOMINICKS	4	60068	420392	878425	300	A	0.095089506	0.26202989	0.062161274	0.220789415
3	5	PALATINE	2	16894	DOMINICKS	5	60067	421203	880431	550	D	0.141433483	0.117368032	0.053875277	0.32122573
4	8	OAK LAWN	5	16895	DOMINICKS	8	60453	417331	877436	600	C	0.123155416	0.252394035	0.035243328	0.095173274
5	9	MORTON GROVE	2	16896	DOMINICKS	9	60053	420411	877994	450	A	0.103503097	0.269119018	0.032618826	0.222172318
6	12	CHICAGO	7	16898	DOMINICKS	12	60660	419928	876592	450	B	0.10569674	0.178341405	0.380697988	0.253412969
7	14	GLENVIEW	1	16899	DOMINICKS	14	60025	420733	877994	400	A	0.129589372	0.213949275	0.034178744	0.348293024
8	18	RIVER GROVE	5	16901	DOMINICKS	18	60171	419364	878331	600	A	0.110094984	0.272313368	0.074417144	0.072246456
9	19		.	.		.	.	.	.		.	.	.	.
10	21	HANOVER PARK	6	16903	DOMINICKS	21	60103	420058	881411	500	D	0.175926346	0.066896458	0.105038777	0.17750345
11	25		.	.		.	.	.	.		.	.	.	.
12	28	MOUNT PROSPECT	2	16905	DOMINICKS	28	60056	420686	879208	275	A	0.128879537	0.213308785	0.055935473	0.233162564
13	39		.	.		.	.	.	.		.	.	.	.
14	46		5							D				

Figure 4: Demographics Temp Table Data

Price Tier temp staging Table

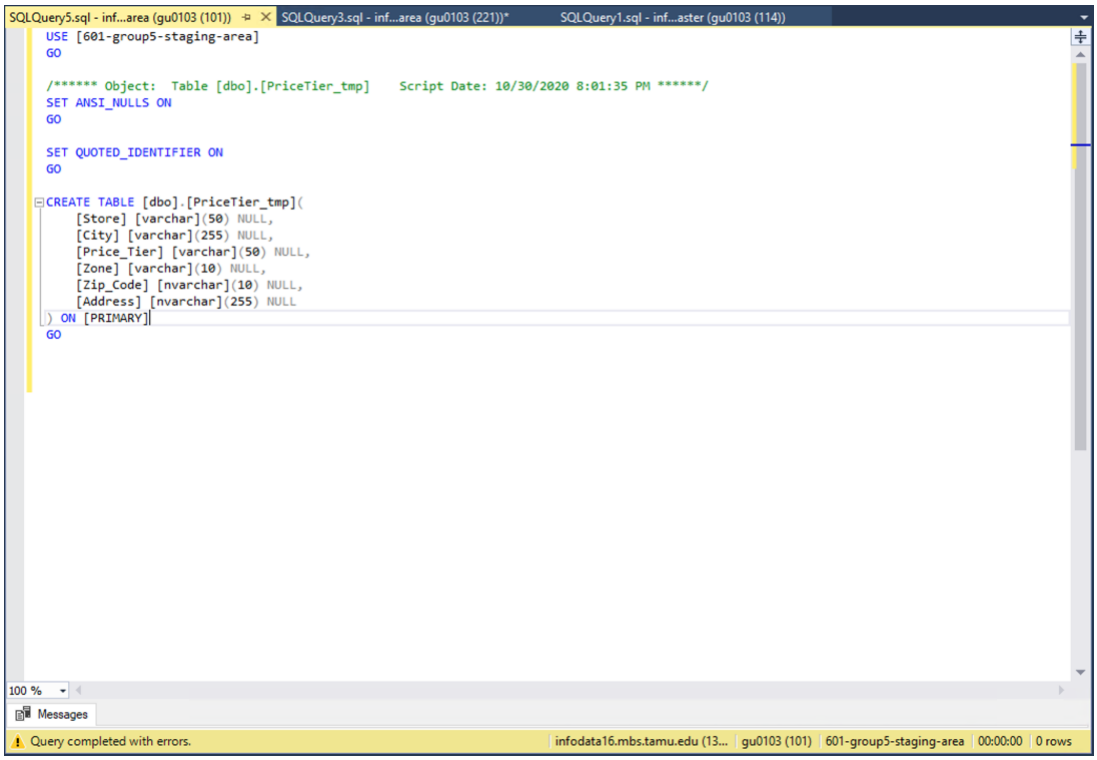


Figure 5: Price Tier Temporary Table Creation

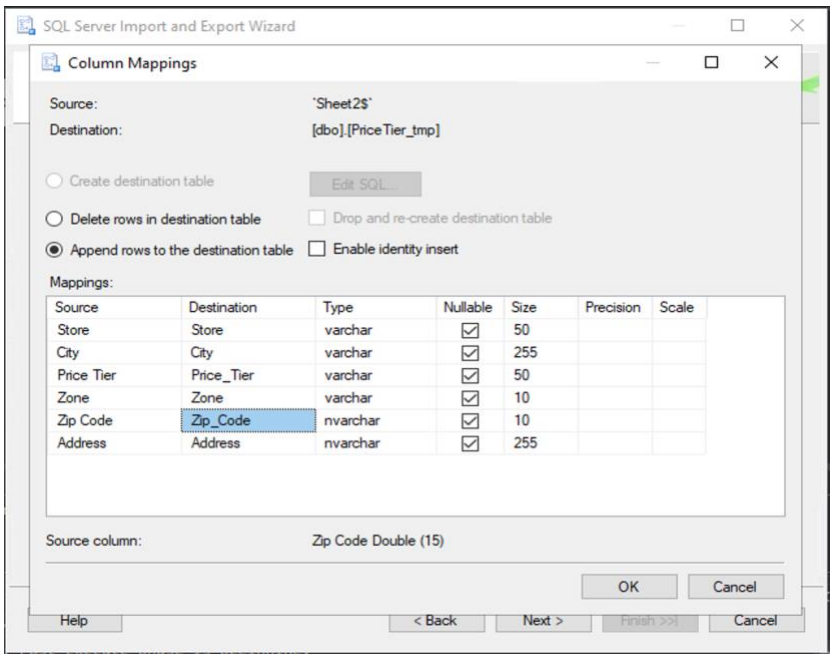


Figure 6: Source to Staging Mapping

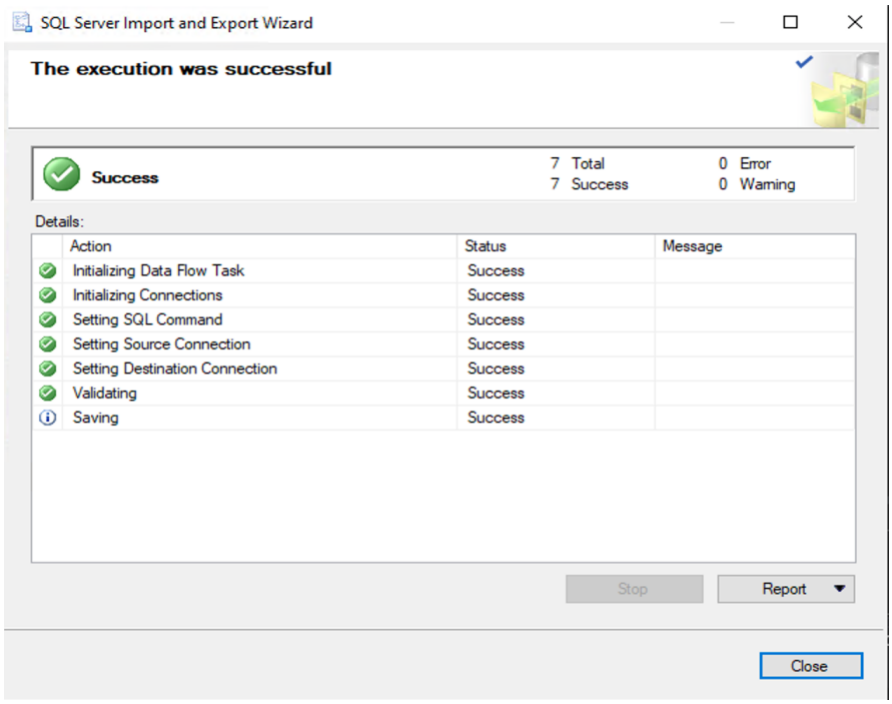


Figure 7: Data Flow

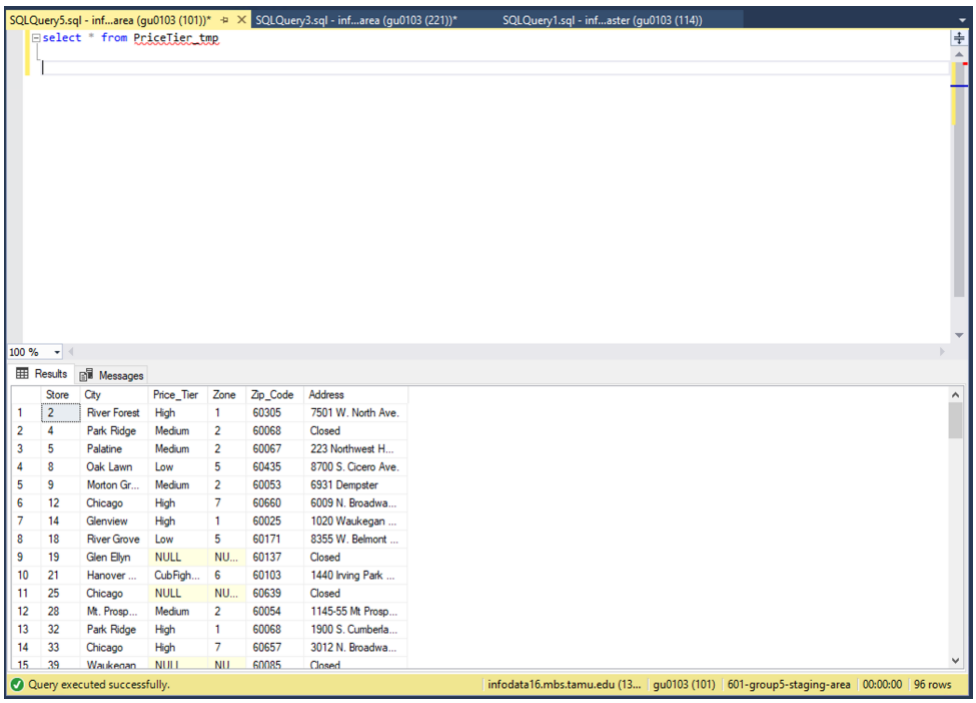


Figure 8: Price Tier Temp Table Data

Ccount Tier temp staging Table

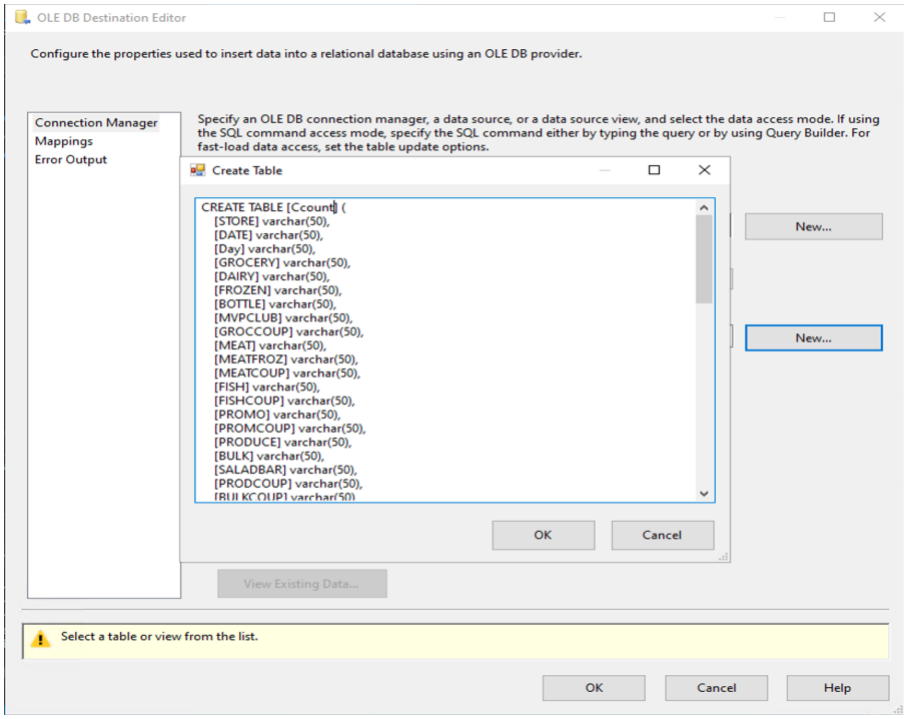


Figure 9: CCount Temp Table Creation

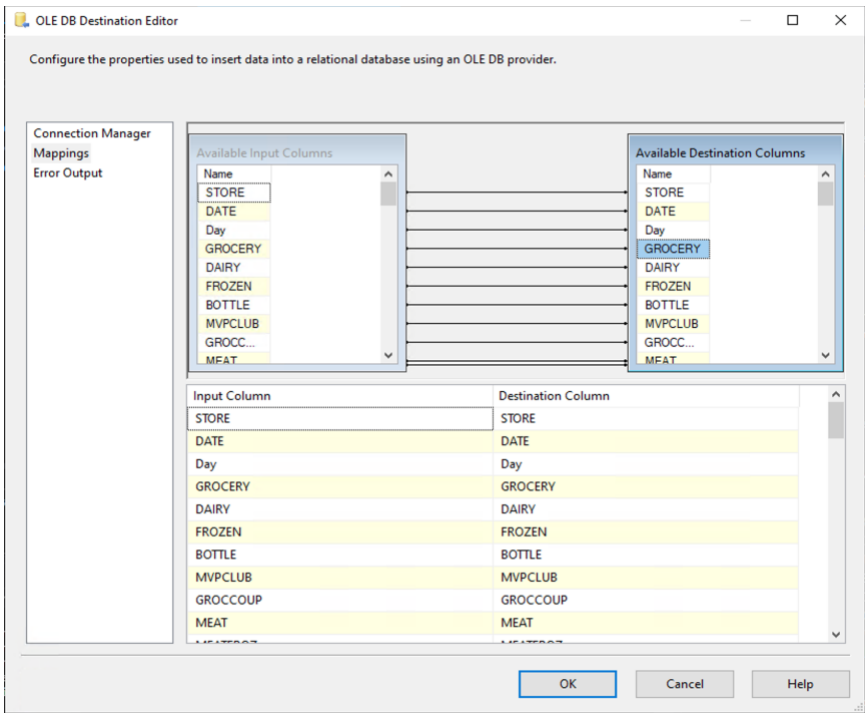


Figure 10: Source to Staging Mapping

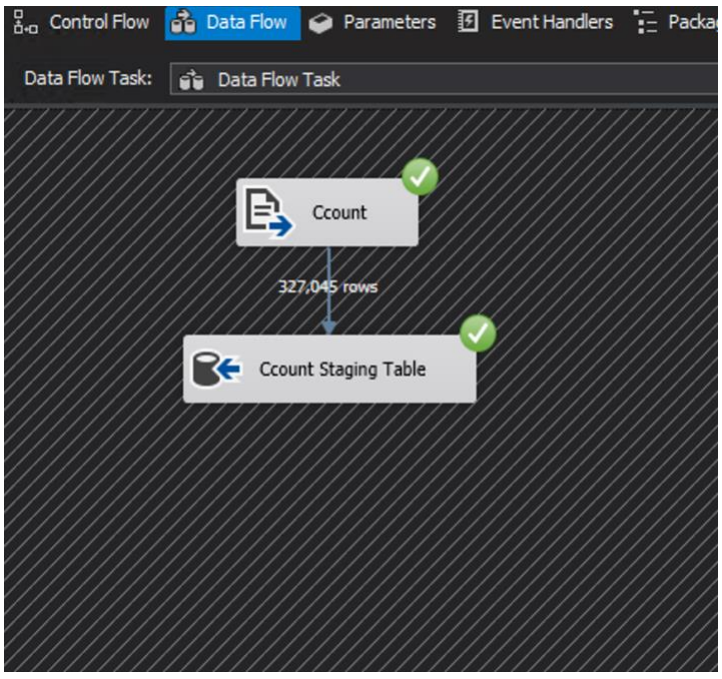
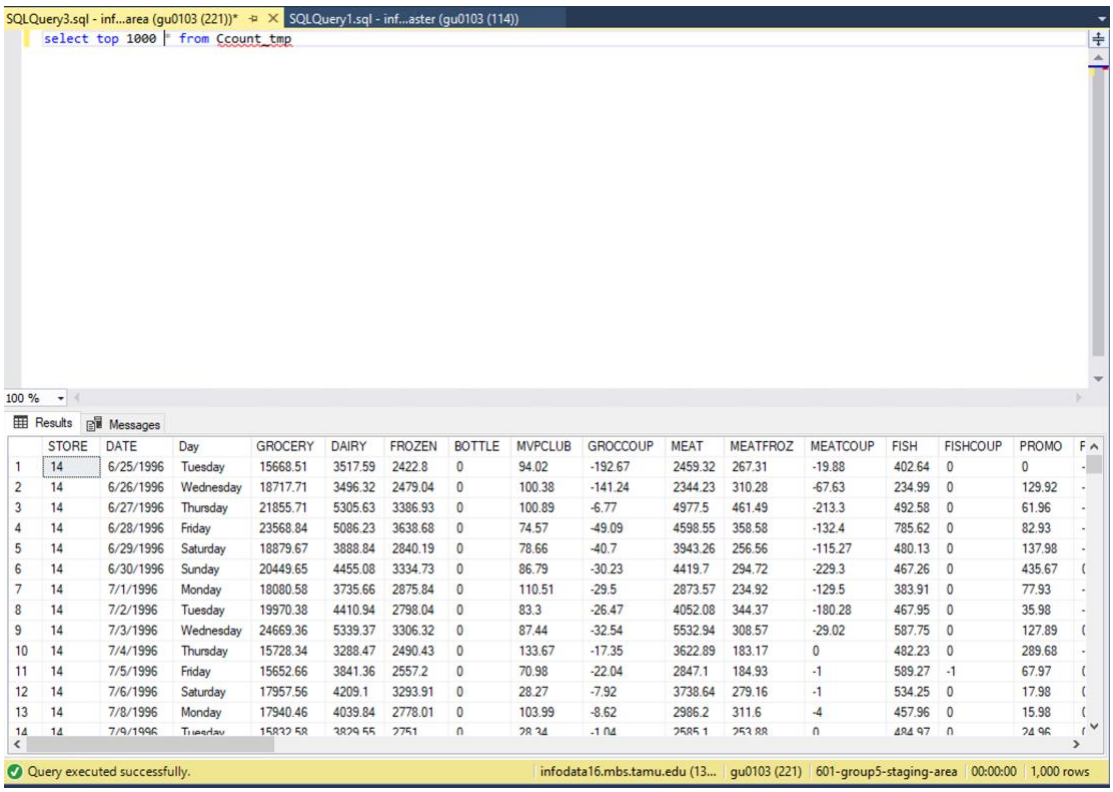


Figure 11: Data Flow



	STORE	DATE	Day	GROCERY	DAIRY	FROZEN	BOTTLE	MVPCLUB	GROCCOUP	MEAT	MEATFROZ	MEATCOUP	FISH	FISHCOUP	PROMO
1	14	6/25/1996	Tuesday	15668.51	3517.59	2422.8	0	94.02	-192.67	2459.32	267.31	-19.88	402.64	0	0
2	14	6/26/1996	Wednesday	18717.71	3496.32	2479.04	0	100.38	-141.24	2344.23	310.28	-67.63	234.99	0	129.92
3	14	6/27/1996	Thursday	21855.71	5305.63	3386.93	0	100.89	-6.77	4977.5	461.49	-213.3	492.58	0	61.96
4	14	6/28/1996	Friday	23568.84	5086.23	3638.68	0	74.57	-49.09	4598.55	358.58	-132.4	785.62	0	82.93
5	14	6/29/1996	Saturday	18879.67	3888.84	2840.19	0	78.66	-40.7	3943.26	256.56	-115.27	480.13	0	137.98
6	14	6/30/1996	Sunday	20449.65	4455.08	3334.73	0	86.79	-30.23	4419.7	294.72	-229.3	467.26	0	435.67
7	14	7/1/1996	Monday	18080.58	3735.66	2875.84	0	110.51	-29.5	2873.57	234.92	-129.5	383.91	0	77.93
8	14	7/2/1996	Tuesday	19970.38	4410.94	2798.04	0	83.3	-26.47	4052.08	344.37	-180.28	467.95	0	35.98
9	14	7/3/1996	Wednesday	24669.36	5339.37	3306.32	0	87.44	-32.54	5532.94	308.57	-29.02	587.75	0	127.89
10	14	7/4/1996	Thursday	15728.34	3288.47	2490.43	0	133.67	-17.35	3622.89	183.17	0	482.23	0	289.68
11	14	7/5/1996	Friday	15652.66	3841.36	2557.2	0	70.98	-22.04	2847.1	184.93	-1	589.27	-1	67.97
12	14	7/6/1996	Saturday	17957.56	4209.1	3293.91	0	28.27	-7.92	3738.64	279.16	-1	534.25	0	17.98
13	14	7/8/1996	Monday	17940.46	4039.84	2778.01	0	103.99	-8.62	2986.2	311.6	-4	457.96	0	15.98
14	14	7/9/1996	Tuesday	15832.58	3829.55	2751	0	28.34	-1.04	2585.1	253.88	0	484.97	0	24.96

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.

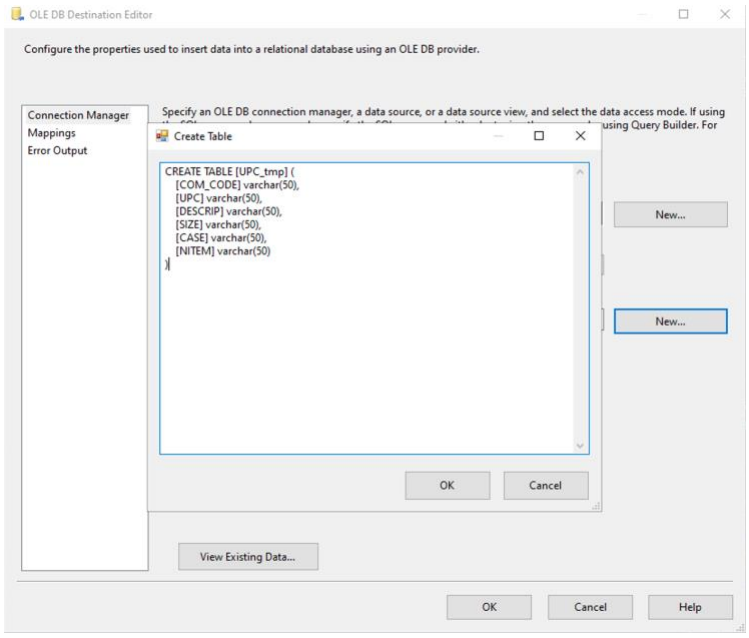


Figure 13: UPC Temporary Table Creation

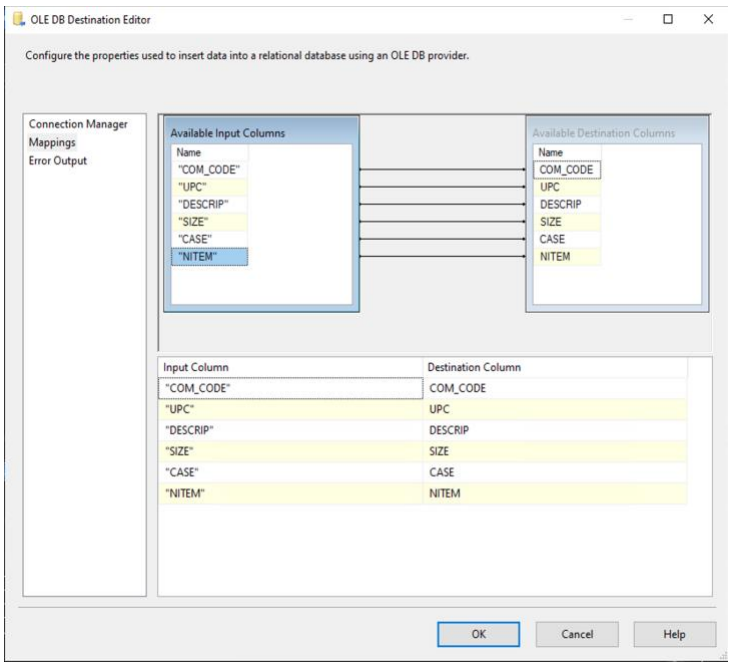


Figure 14: Source to Staging Mapping

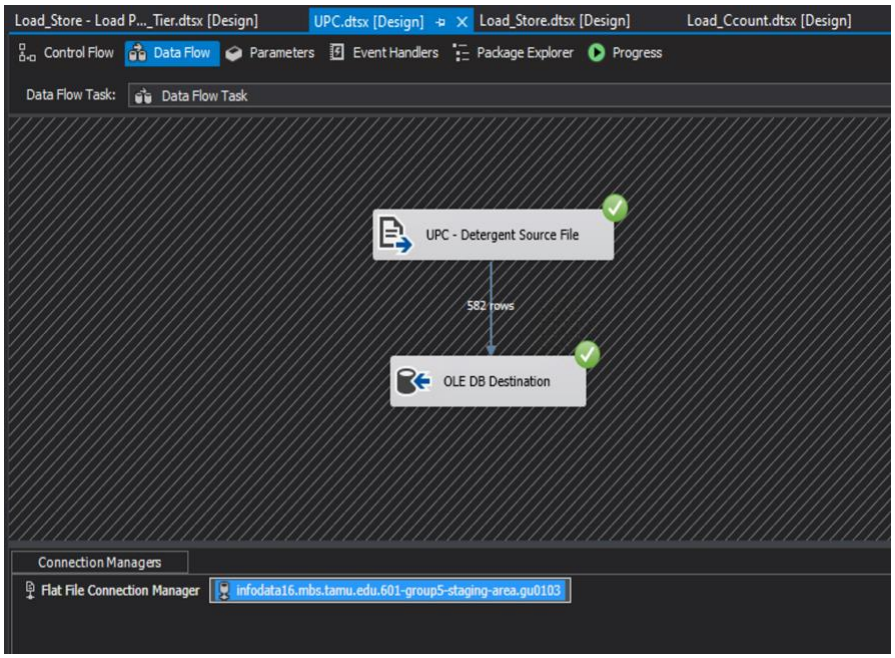


Figure 15: Data Flow

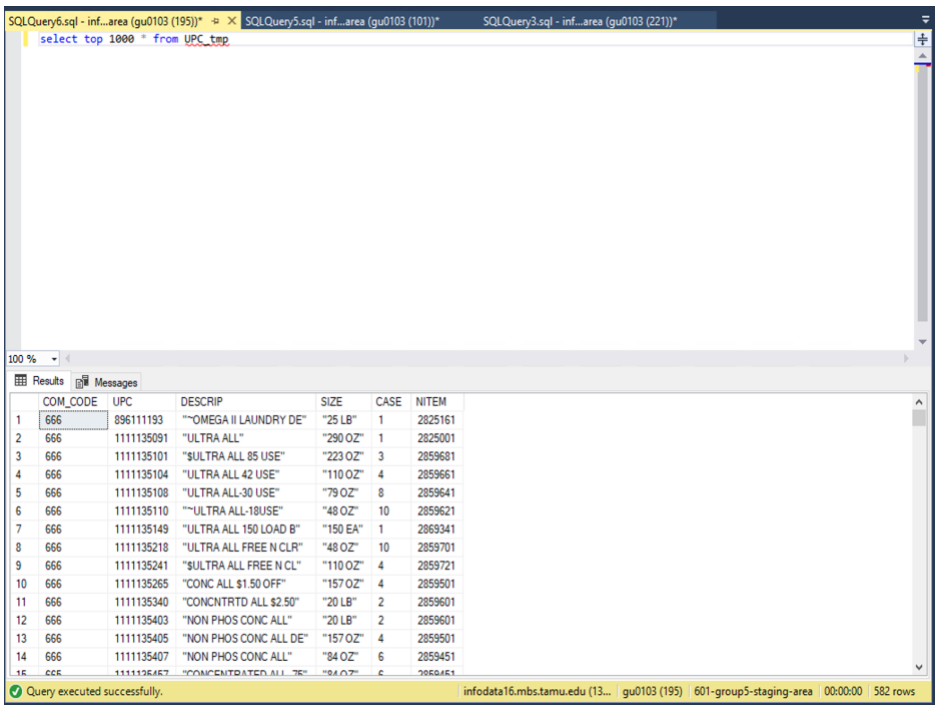


Figure 16: UPC Temporary Table Data

Movement temp staging Table

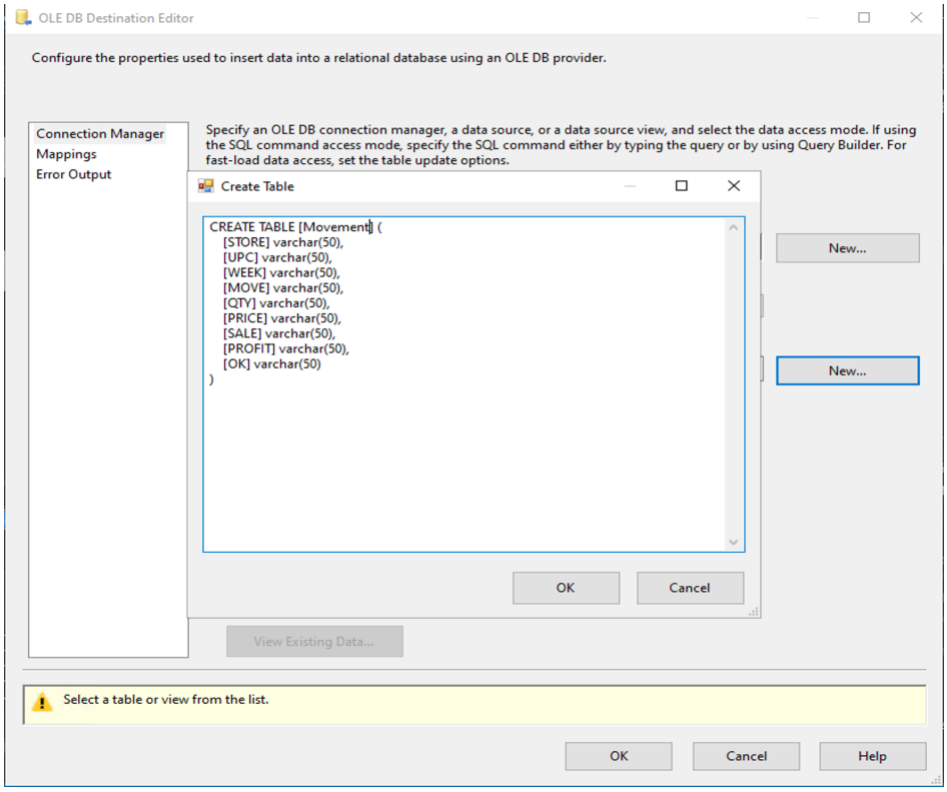


Figure 17: Movement Temporary Table Creation

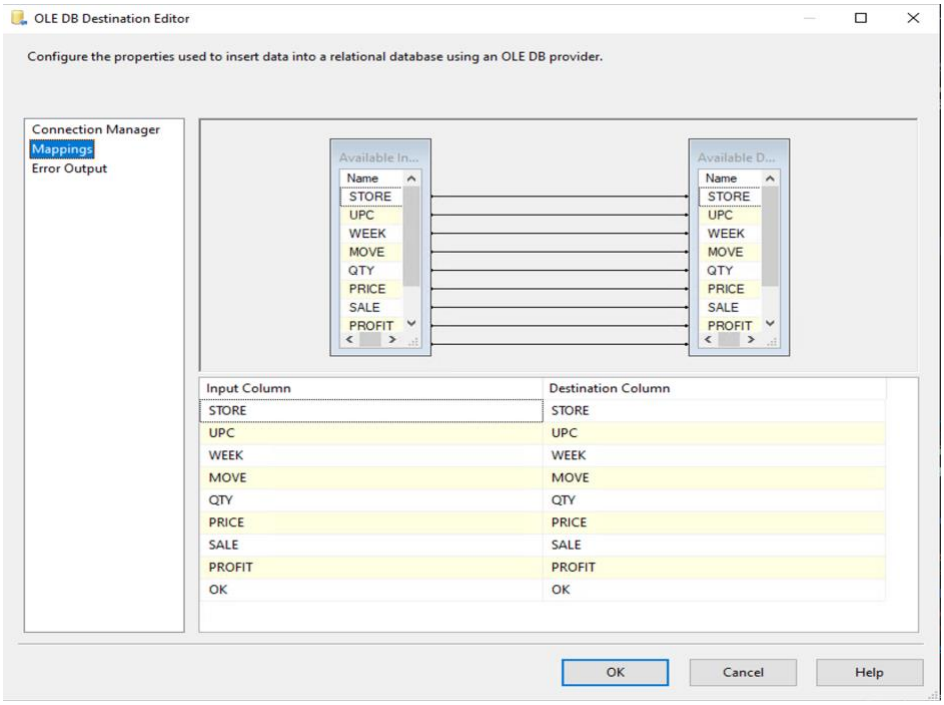


Figure 18: Source and Staging Mapping

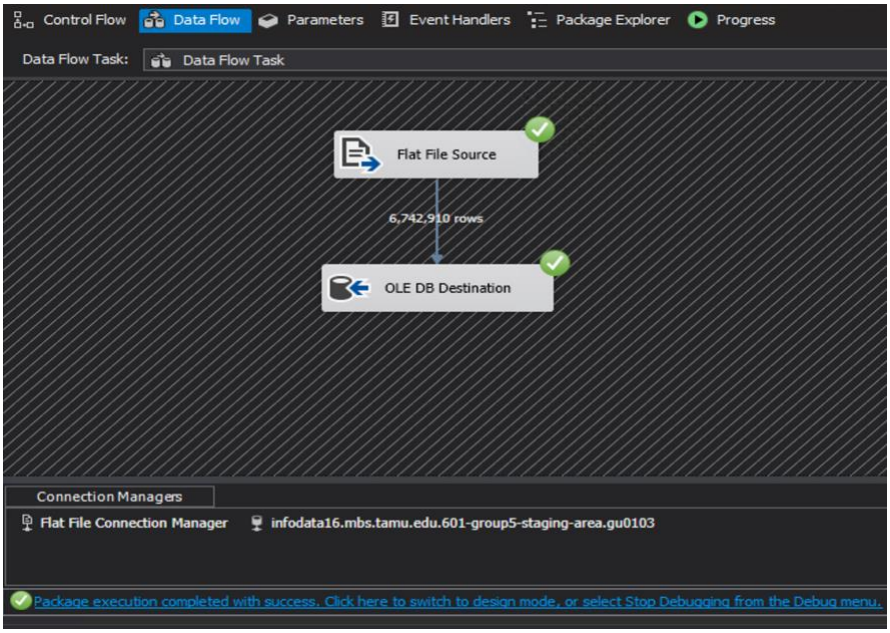
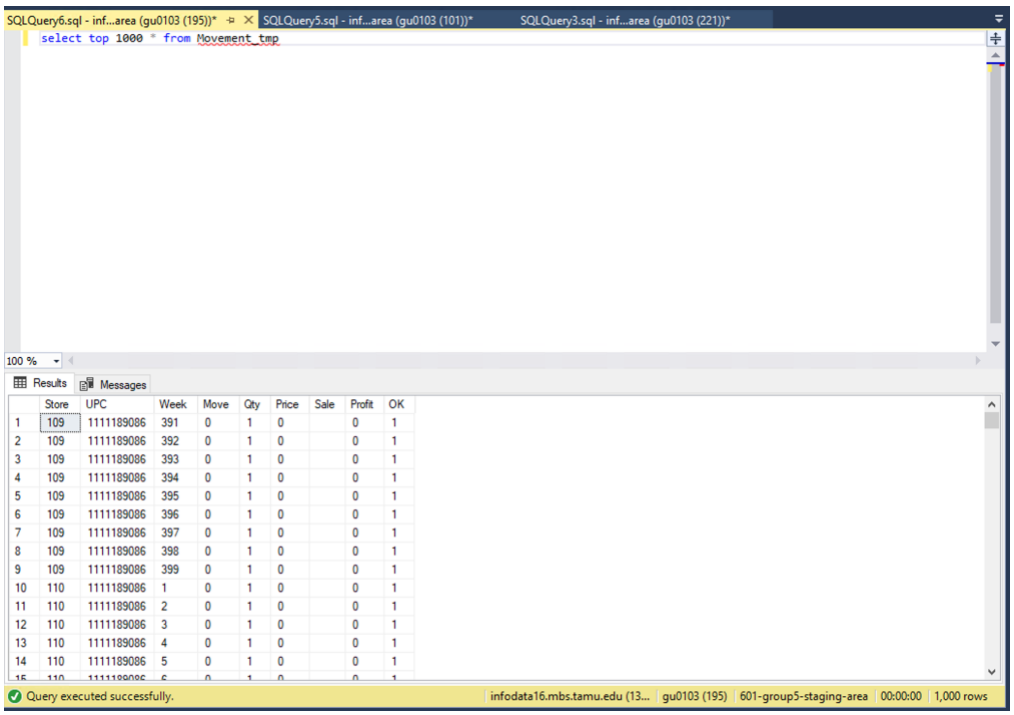


Figure 19: Data Flow



	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

Figure 20: Movement Temporary Data Table

Week_Decode temp staging Table

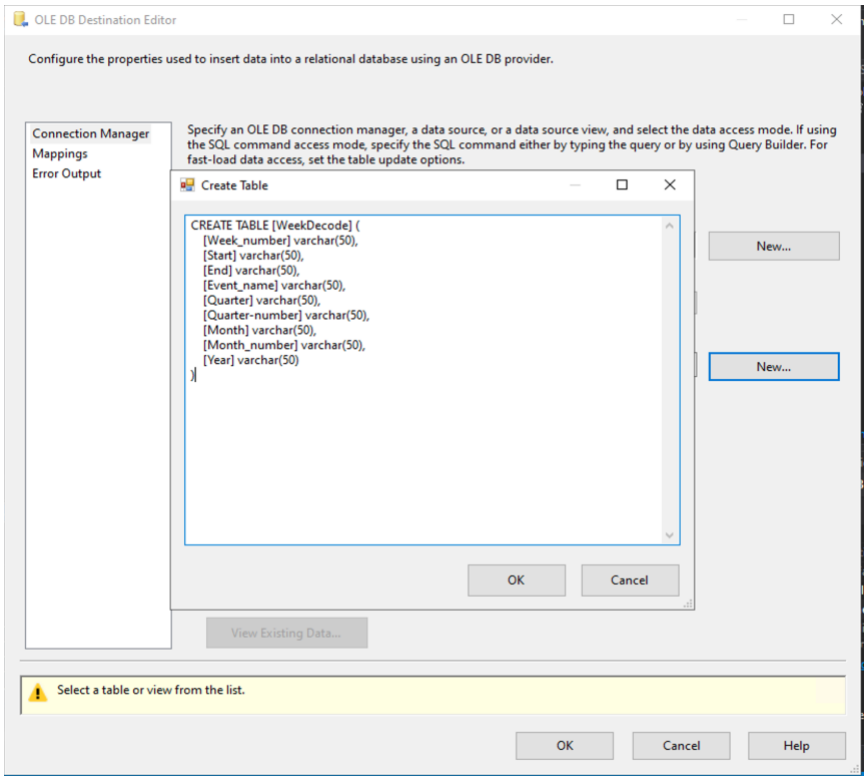


Figure 21: Week Decode Temporary Table Creation

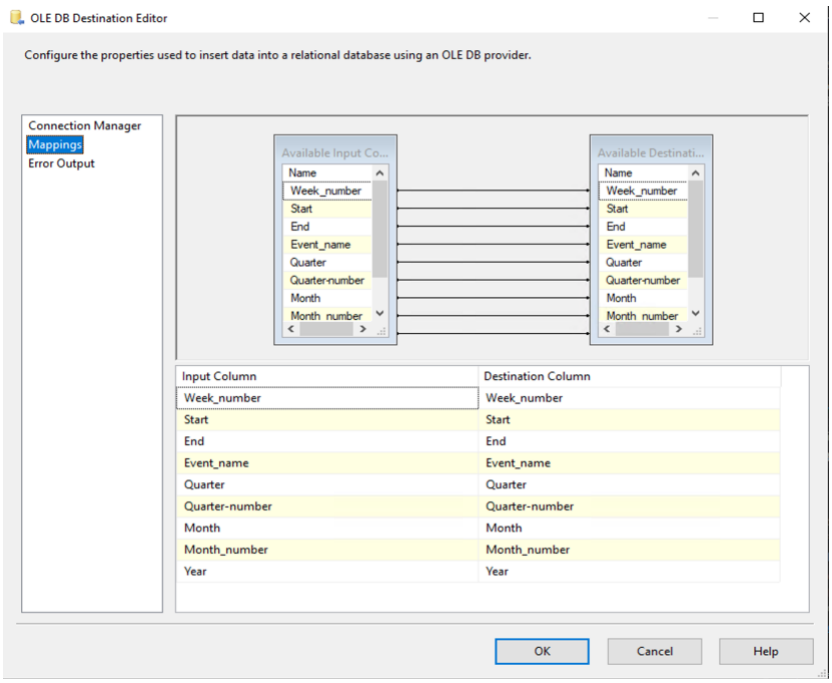


Figure 22: Source to Staging Mapping

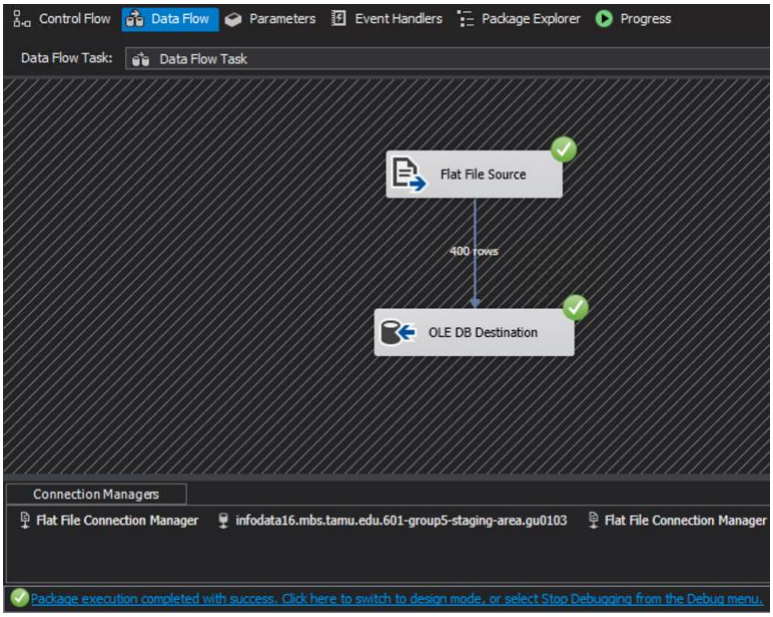
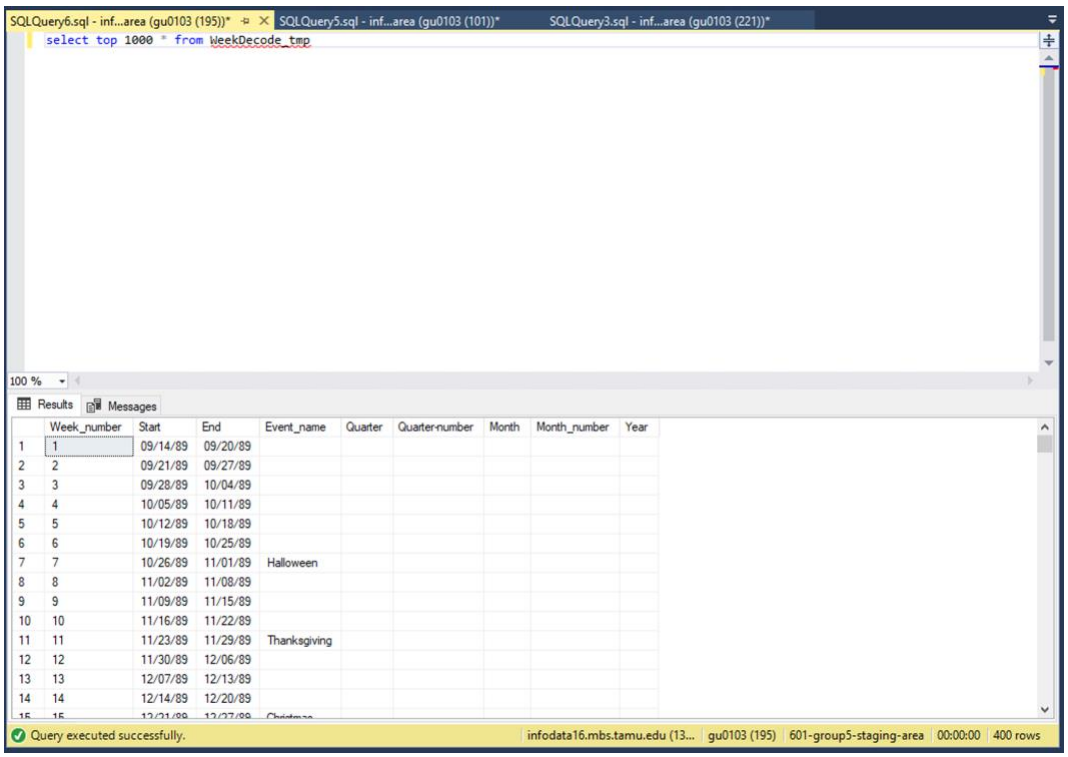


Figure 23: Data Flow



	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						

Figure 24: Week_Decode Temporary Data

19.2 STAGING AREA BEFORE CLEANING AND TRANSFORMATION

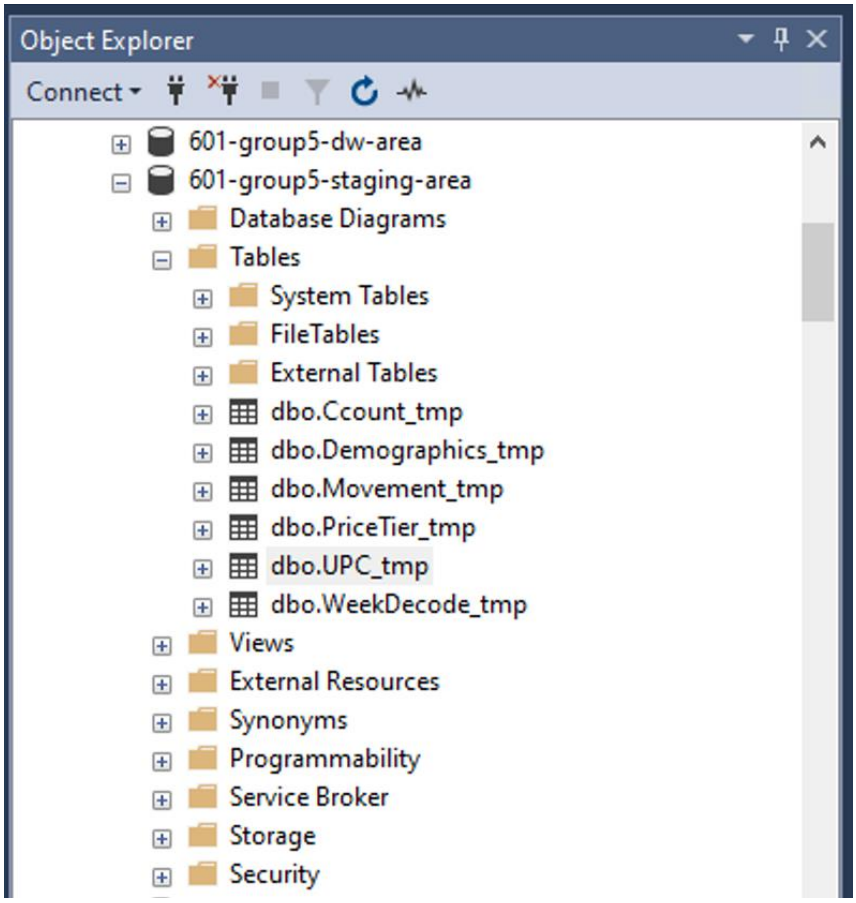
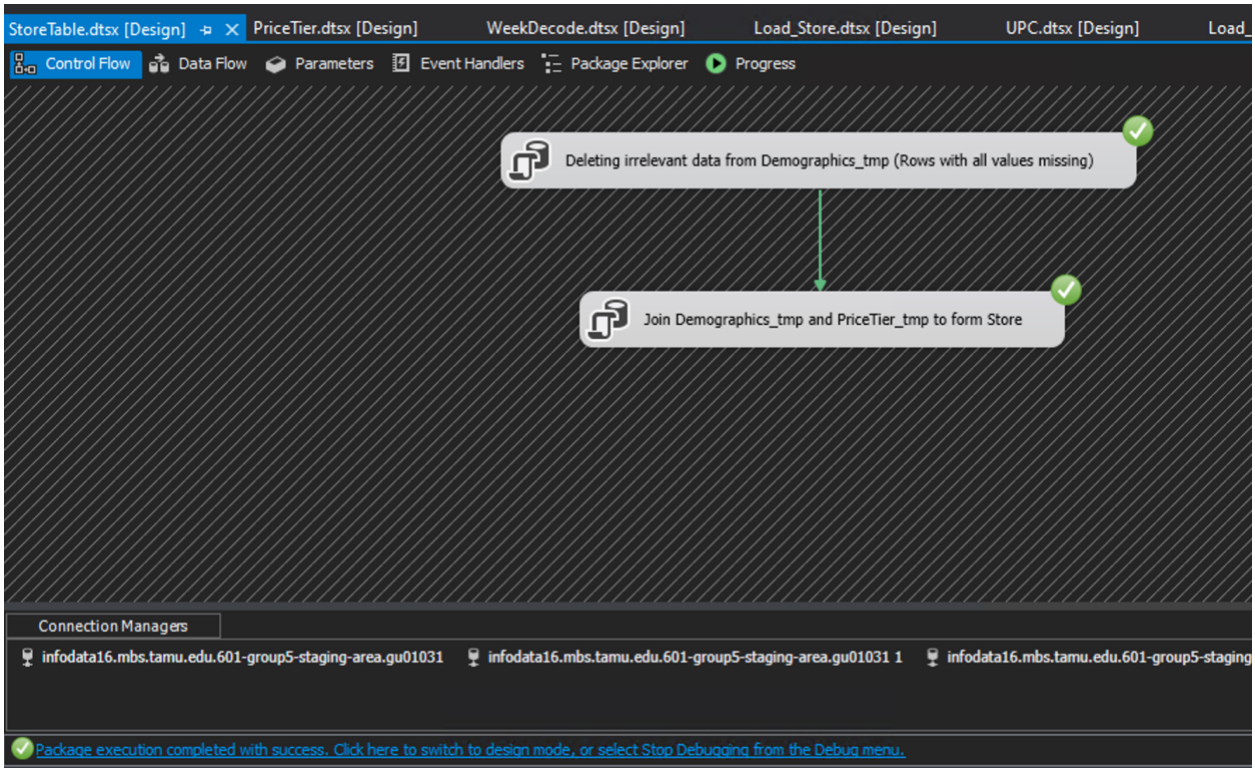


Figure 25: SSMS Database Snapshot containing Temporary Files

19.3 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='.') and (ZONE='' or ZONE = '.')
```

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
```

CCOUNT

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

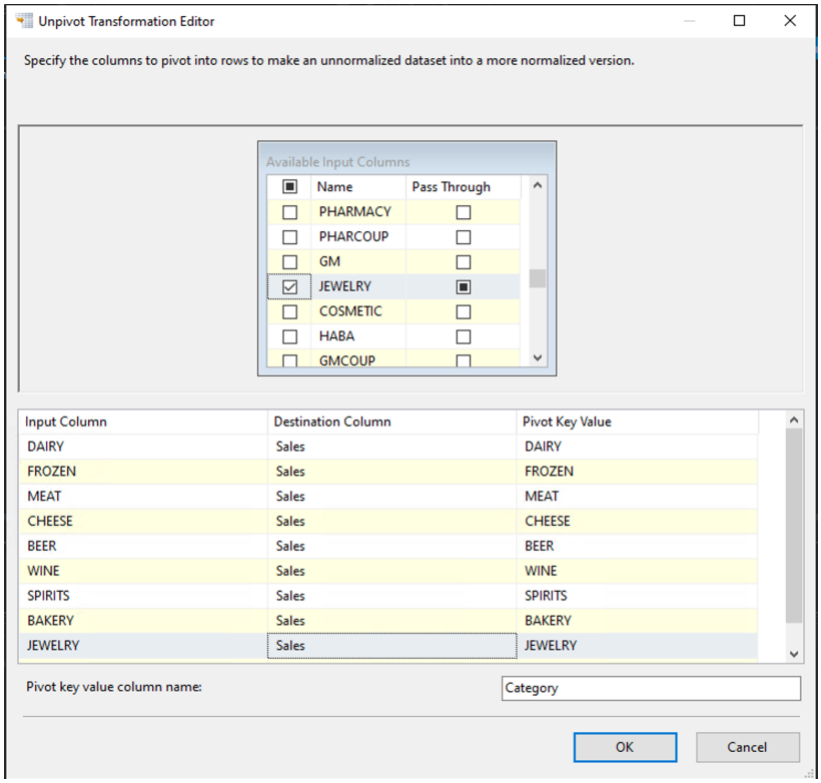
Removing inconsistent records

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’

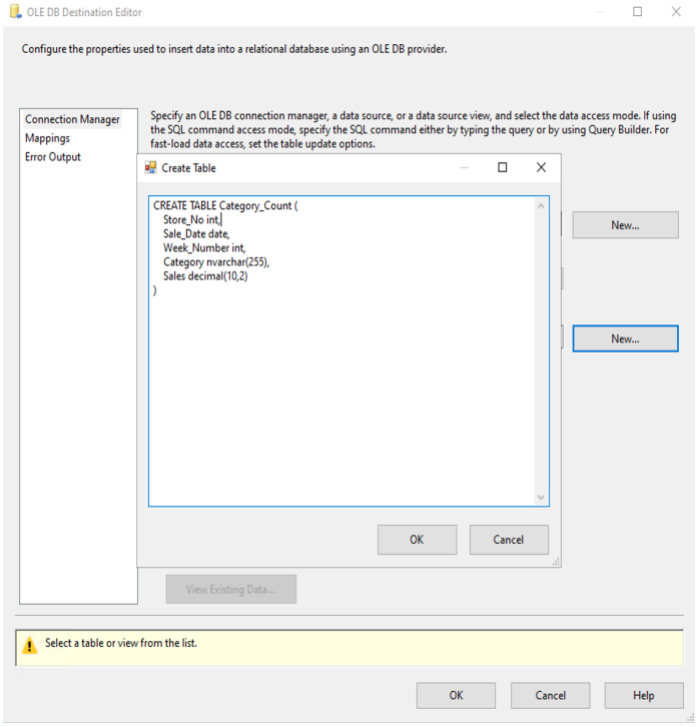
Data from Ccount_tmp will be moved to 2 different final staging tables.

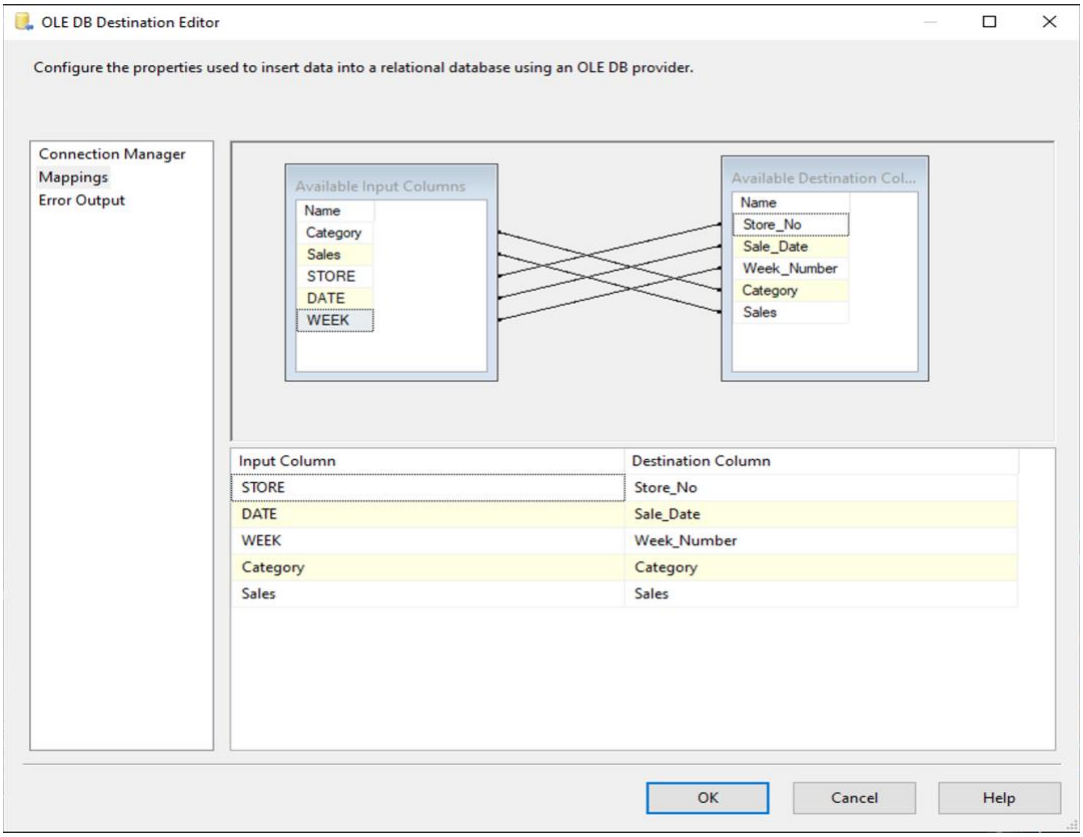
1. Moving categorical sales data to Category_Count staging table

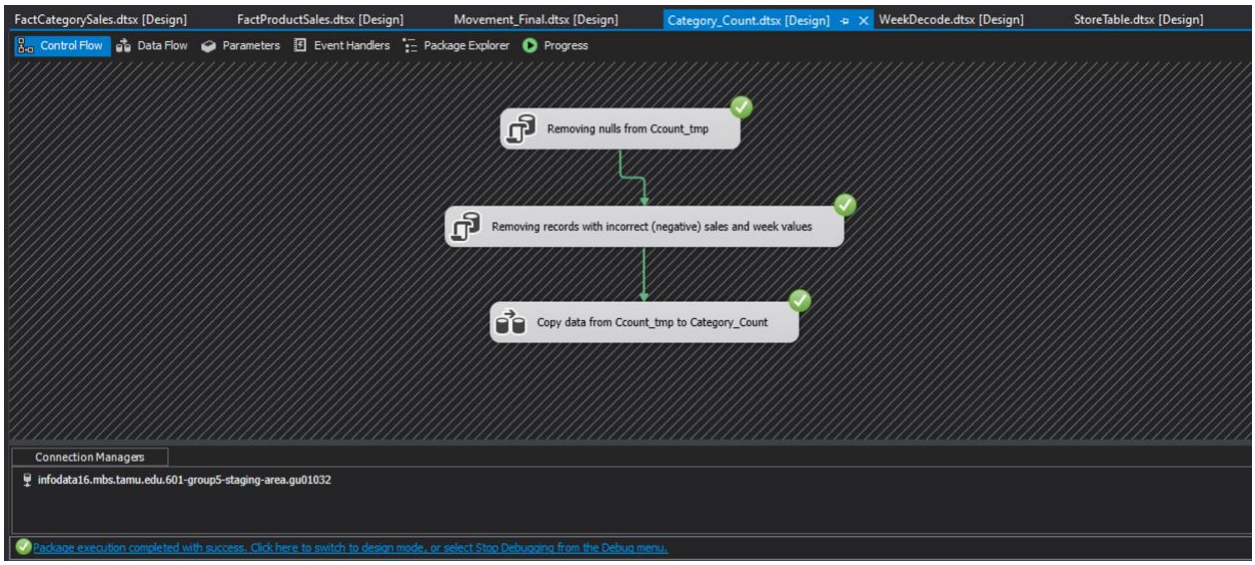
Unpivoting temp data

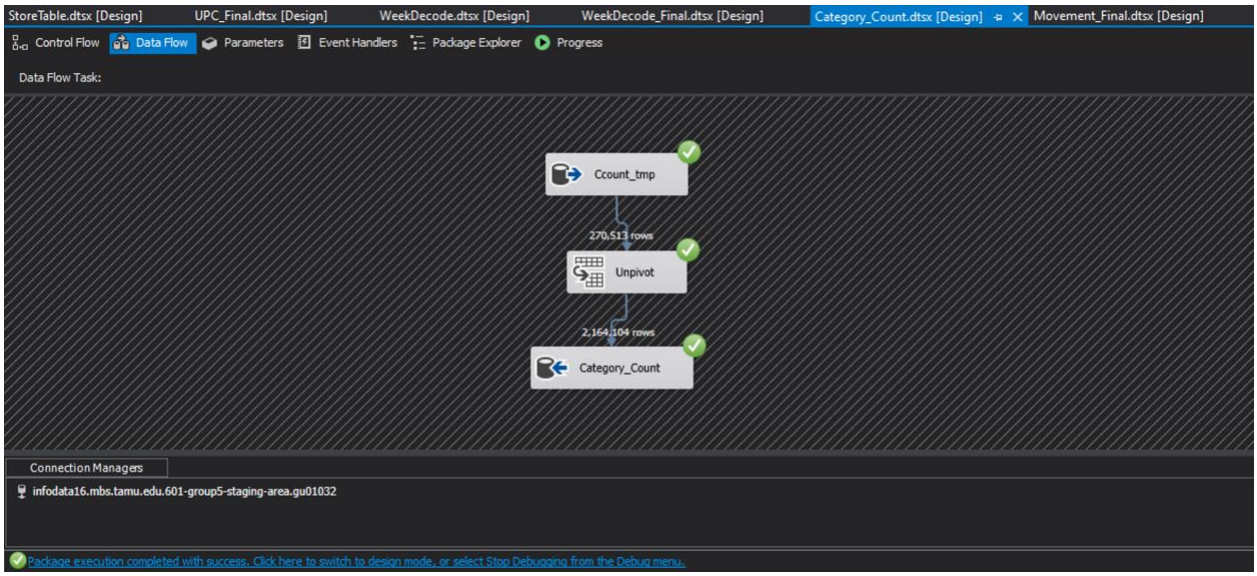


Moving Unpivoted data to final staging table

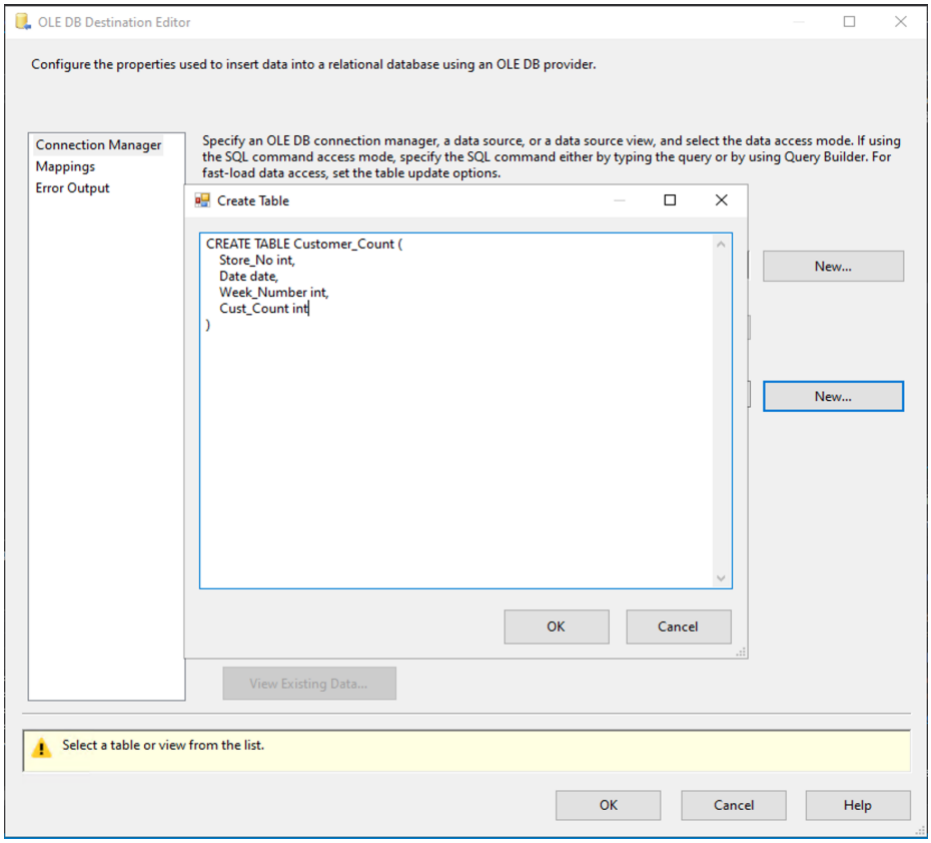


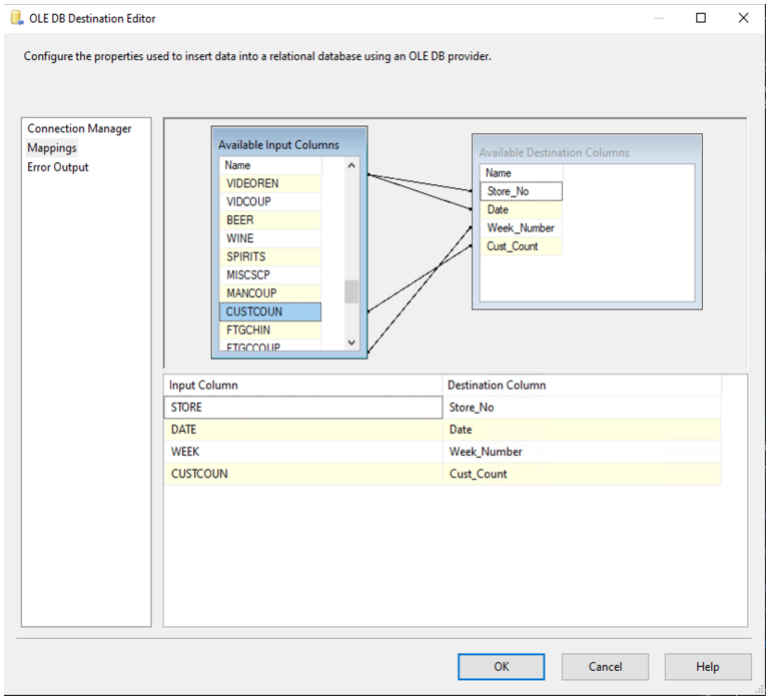


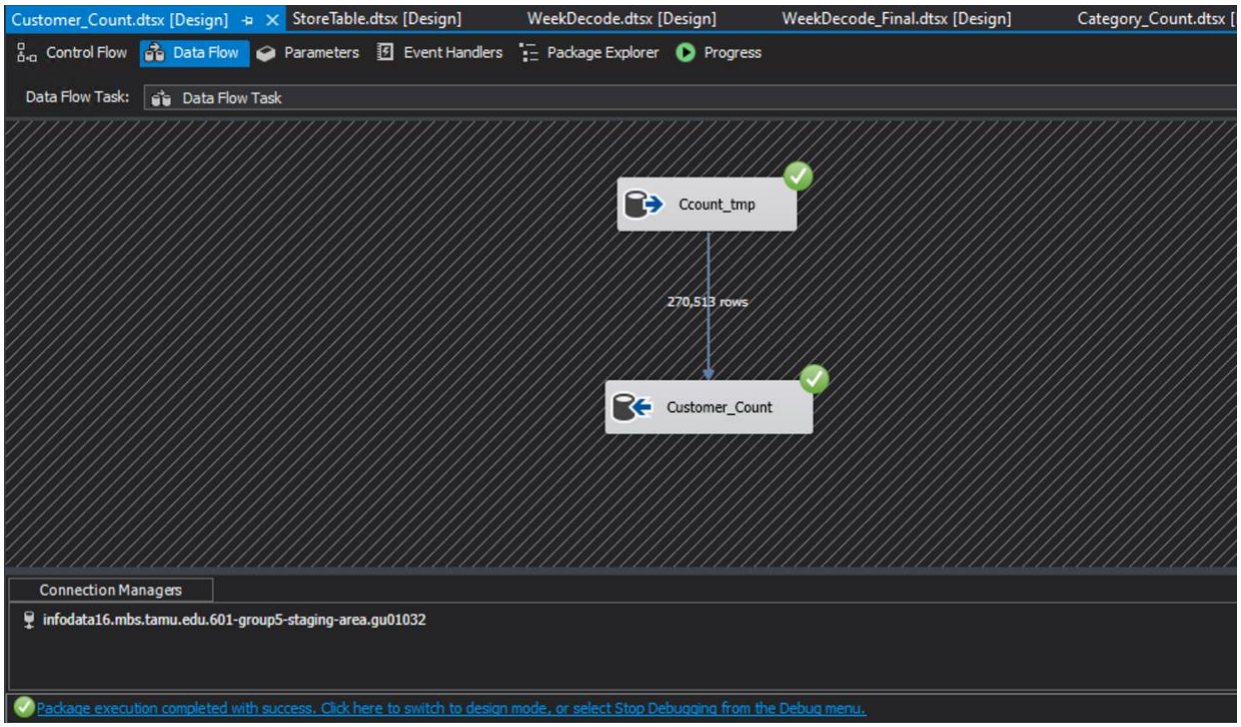

FactCategorySales.dtsx [Design]
FactProductSales.dtsx [Design]
Movement_Final.dtsx [Design]
Category_Count.dtsx [Design]
WeekDecode.dtsx [Design]
StoreTable.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Removing nulls from Ccount_tmp
Removing records with incorrect (negative) sales and week values
Copy data from Ccount_tmp to Category_Count
Connection Managers
infodata16.mbs.tamu.edu.601-group5-staging-area.gu01032
Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.


StoreTable.dtsx [Design]
UPC_Final.dtsx [Design]
WeekDecode.dtsx [Design]
WeekDecode_Final.dtsx [Design]
Category_Count.dtsx [Design]
Movement_Final.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Ccount_tmp
270,513 rows
Unpivot
2,164,104 rows
Category_Count
Connection Managers
infodata16.mbs.tamu.edu.601-group5-staging-area.gu01032
Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

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

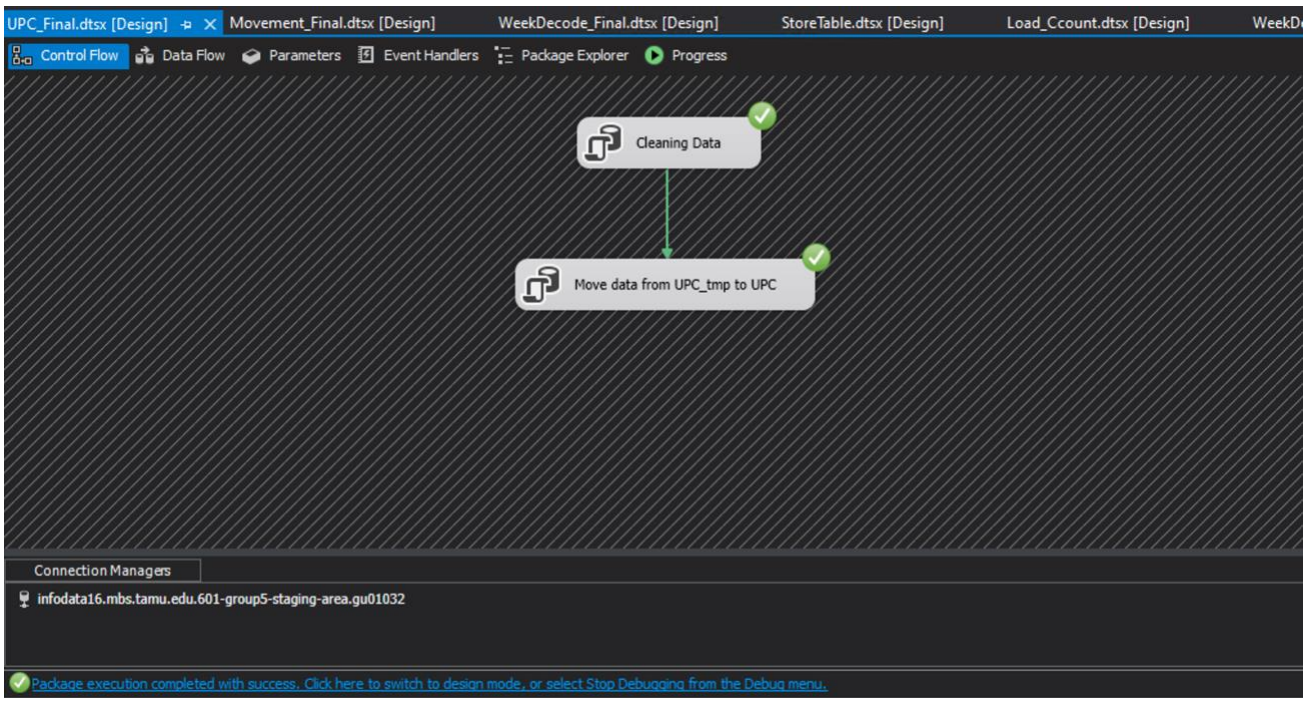
Cleaning Data

```
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, DESCRIP from UPC_tmp
```



19.4 STAGING AREA AFTER CLEANING AND TRANSFORMATION

- 601-group5-staging-area
  - Database Diagrams
  - Tables
    - System Tables
    - FileTables
    - External Tables
    - dbo.Category_Count
    - dbo.Ccount_tmp
    - dbo.Customer_Count
    - dbo.Demographics_tmp
    - dbo.Movement
    - dbo.Movement_tmp
    - dbo.PriceTier_tmp
    - dbo.Store
    - dbo.UPC
    - dbo.UPC_tmp
    - dbo.WeekDecode
    - dbo.WeekDecode_tmp
  - Views
  - External Resources
  - Synonyms
  - Programmability
  - Service Broker
  - Storage
  - Security

19.5 LOADING OF DATA

19.5.1 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**

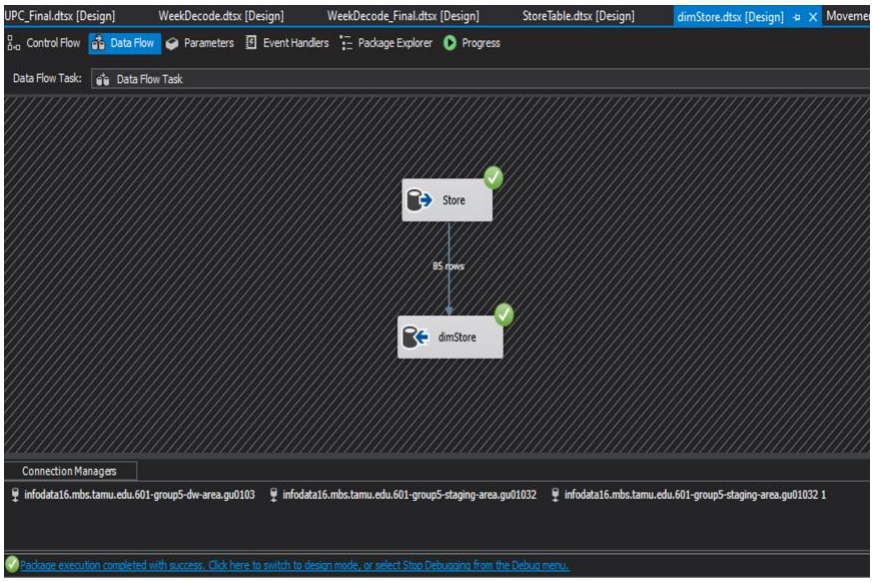
```
CREATE TABLE dimProduct(
        Product_Key int identity(1,1) PRIMARY KEY,
        UPC varchar(50) not null,
        Product_Name varchar(255),
        Product_Category varchar(255))
```

4. **dimProductCategory**

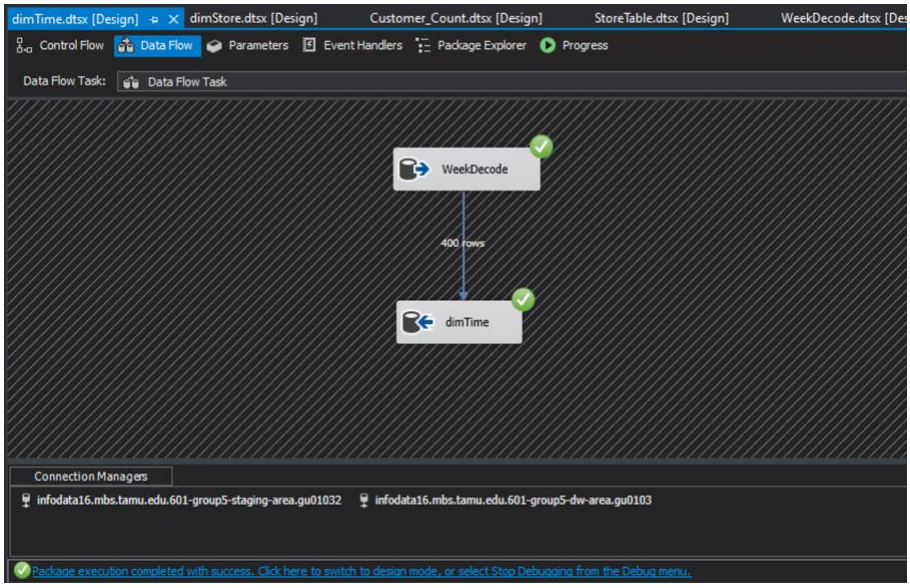
```
CREATE TABLE dimProductCategory(
        Category_Key int identity(1,1) PRIMARY KEY,
        Category nvarchar(255)
)
```

19.5.2 POPULATING DIMENSION TABLES

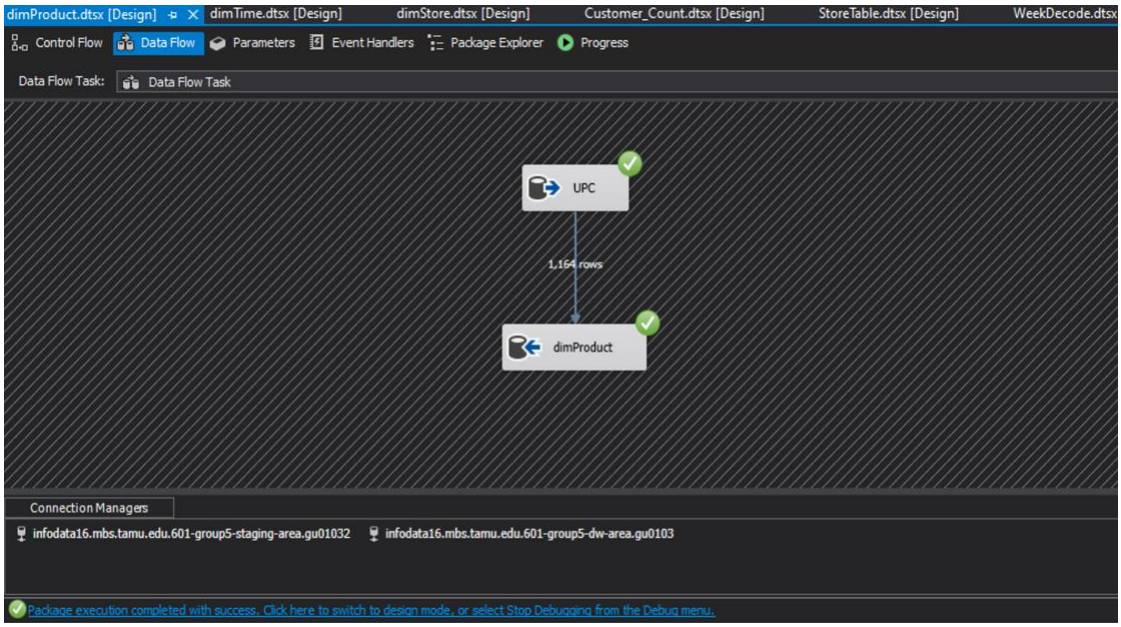
1. **dimStore:**



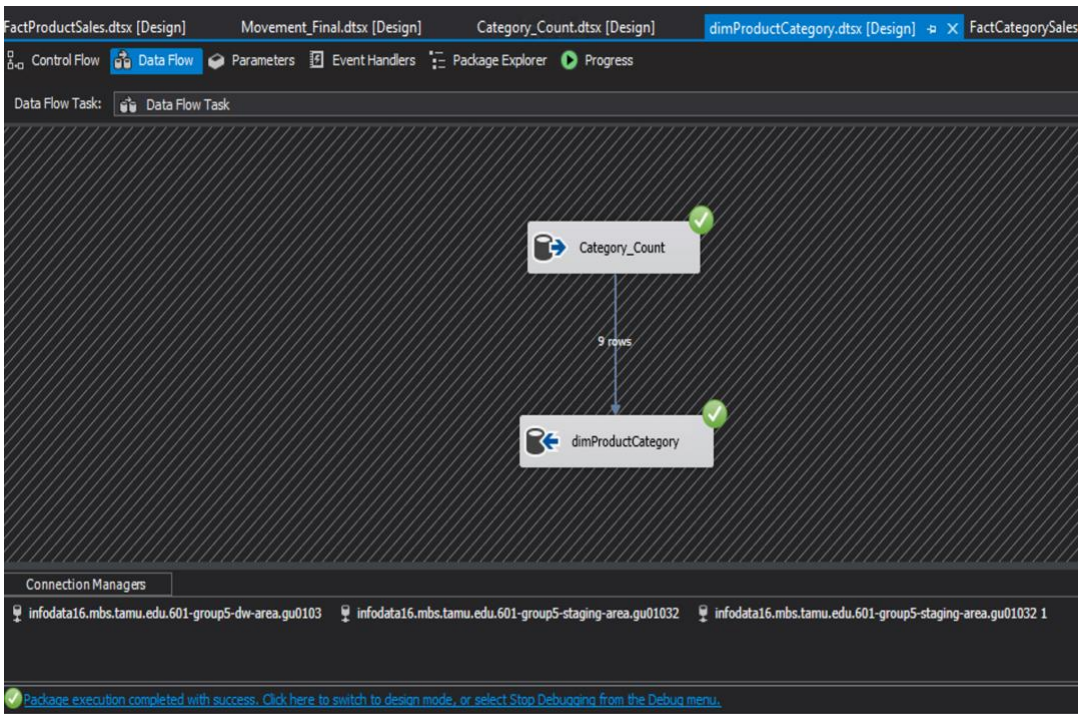
2. **dimTime:**



3. dimProduct:



4. dimProductCategory



19.5.3 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**

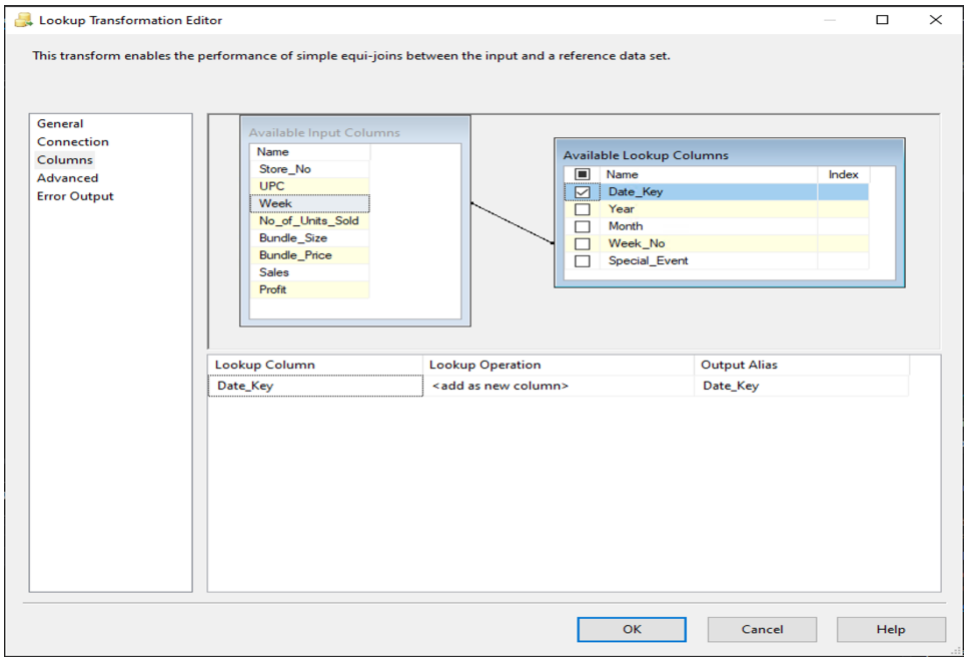
```
CREATE TABLE FactProductSales(
Date_Key int not null,
Product_Key int not null,
Store_Key int not null,
Bundle_Price decimal(10,2),
Profit decimal(10,2),
No_Of_Units_Sold int,
Bundle_Size int,
Sales decimal (10,2),
PRIMARY KEY (Date_Key, Store_Key, Product_Key),
FOREIGN KEY (Date_Key) REFERENCES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

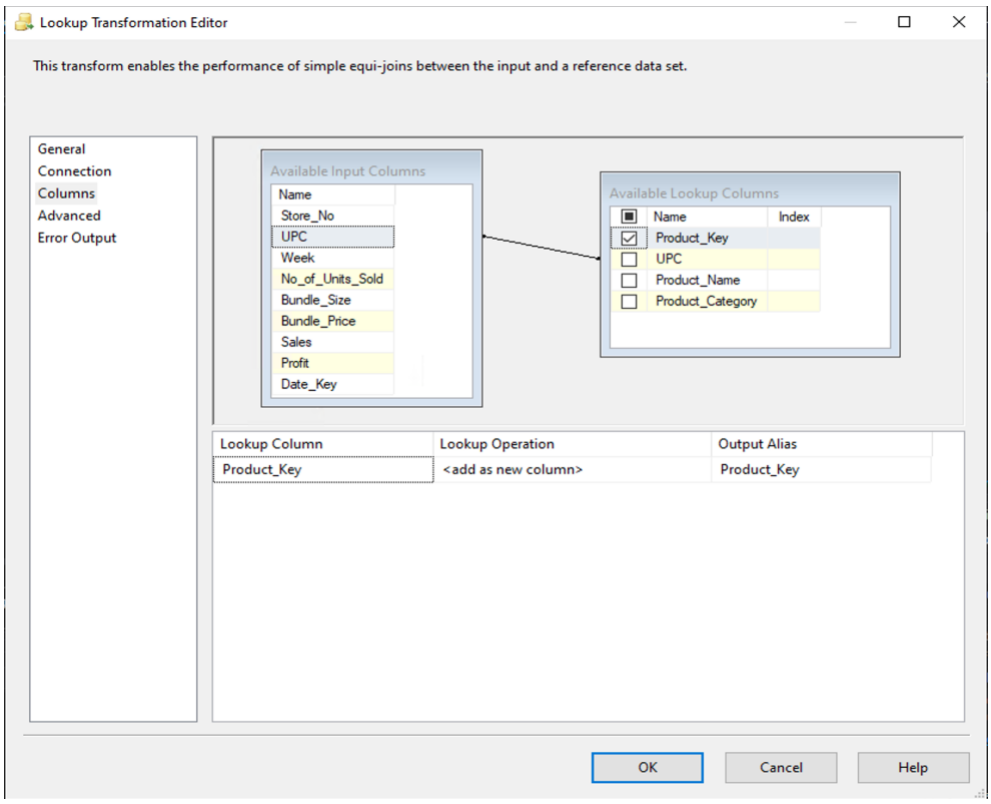
3. **FactCategorySales**

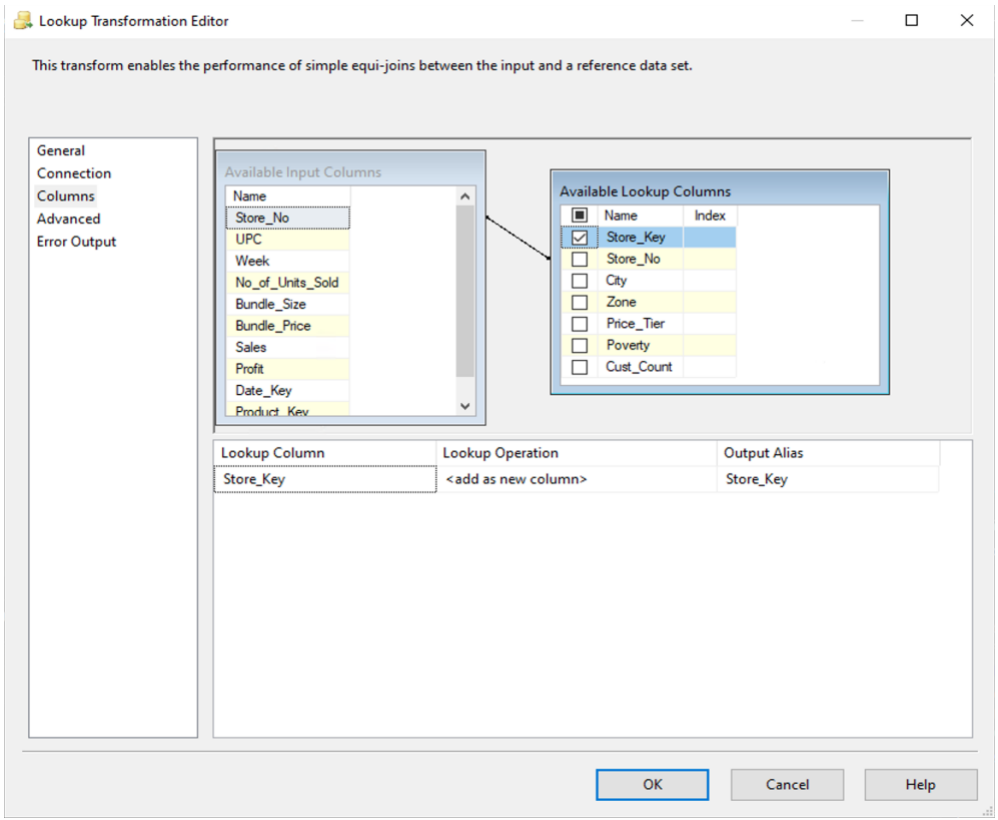
```
CREATE TABLE FactCategorySales(
Store_Key int not null,
Date_Key int not null,
Category_Key int not null,
Sales decimal (10,2),
PRIMARY KEY (Date_Key, Category_Key, Store_Key),
FOREIGN KEY (Date_Key) REFERENCES dimTime(Date_Key),
FOREIGN KEY (Category_Key) REFERENCES dimProductCategory(Category_Key),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

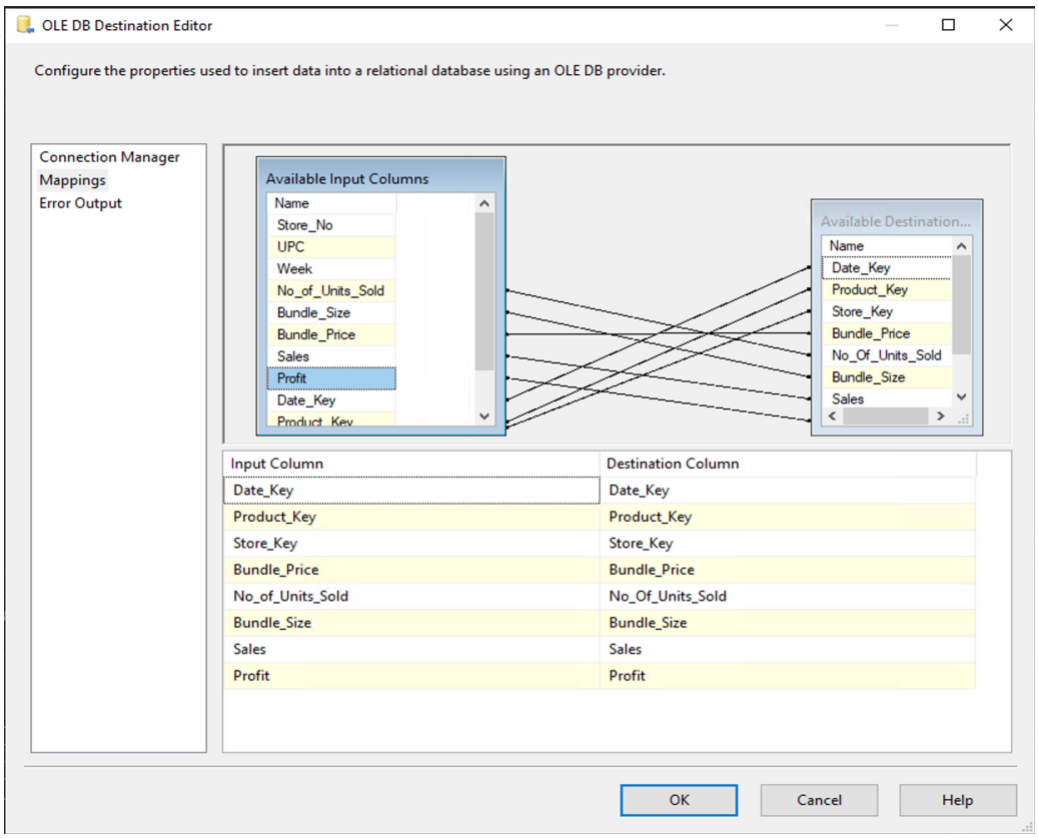
19.5.4 POPULATING FACT TABLES

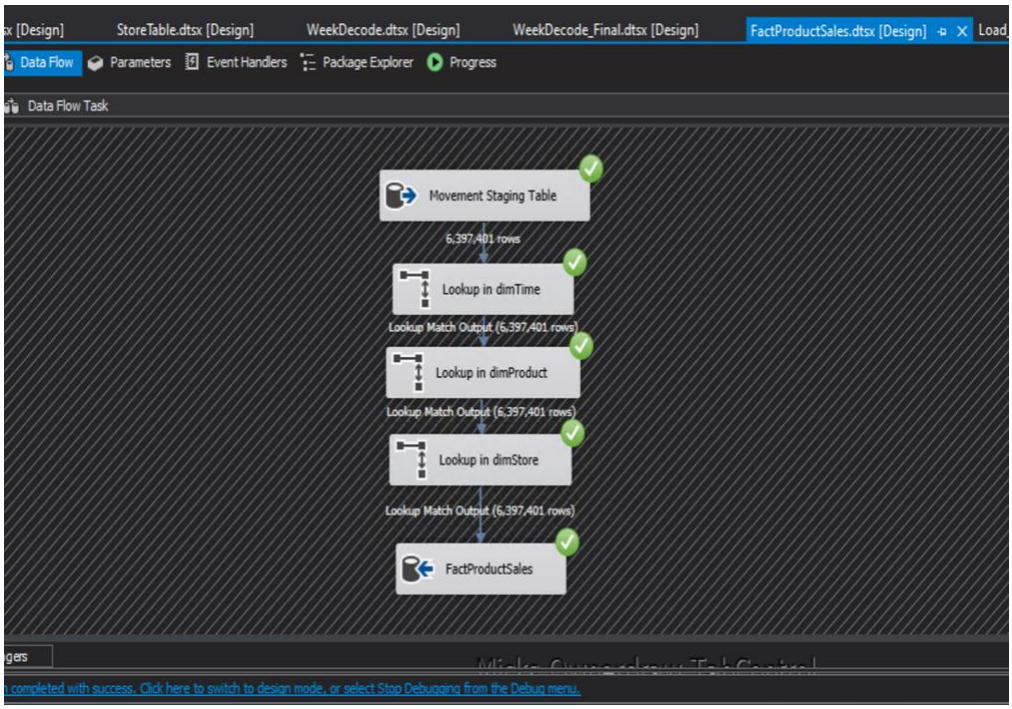
FactProductSales:



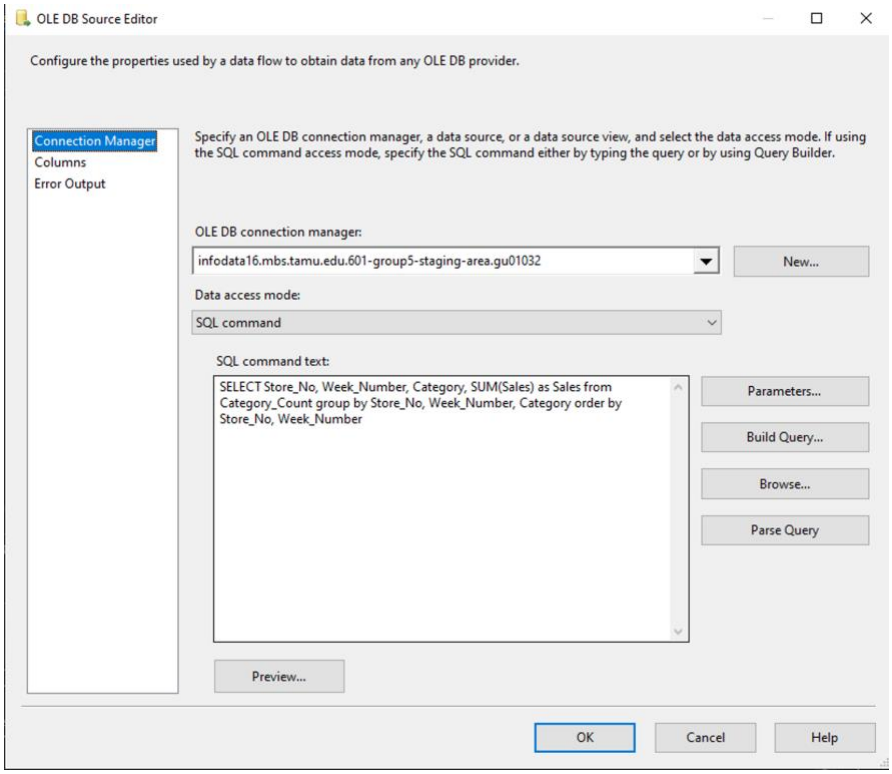

Lookup Transformation Editor
This transform enables the performance of simple equi-joins between the input and a reference data set.
General
Connection
Columns
Advanced
Error Output
Available Input Columns
Name
Store_No
UPC
Week
No_of_Units_Sold
Bundle_Size
Bundle_Price
Sales
Profit
Date_Key
Available Lookup Columns
Name
Index
Product_Key
UPC
Product_Name
Product_Category
Lookup Column
Lookup Operation
Output Alias
Product_Key
<add as new column>
Product_Key
OK
Cancel
Help

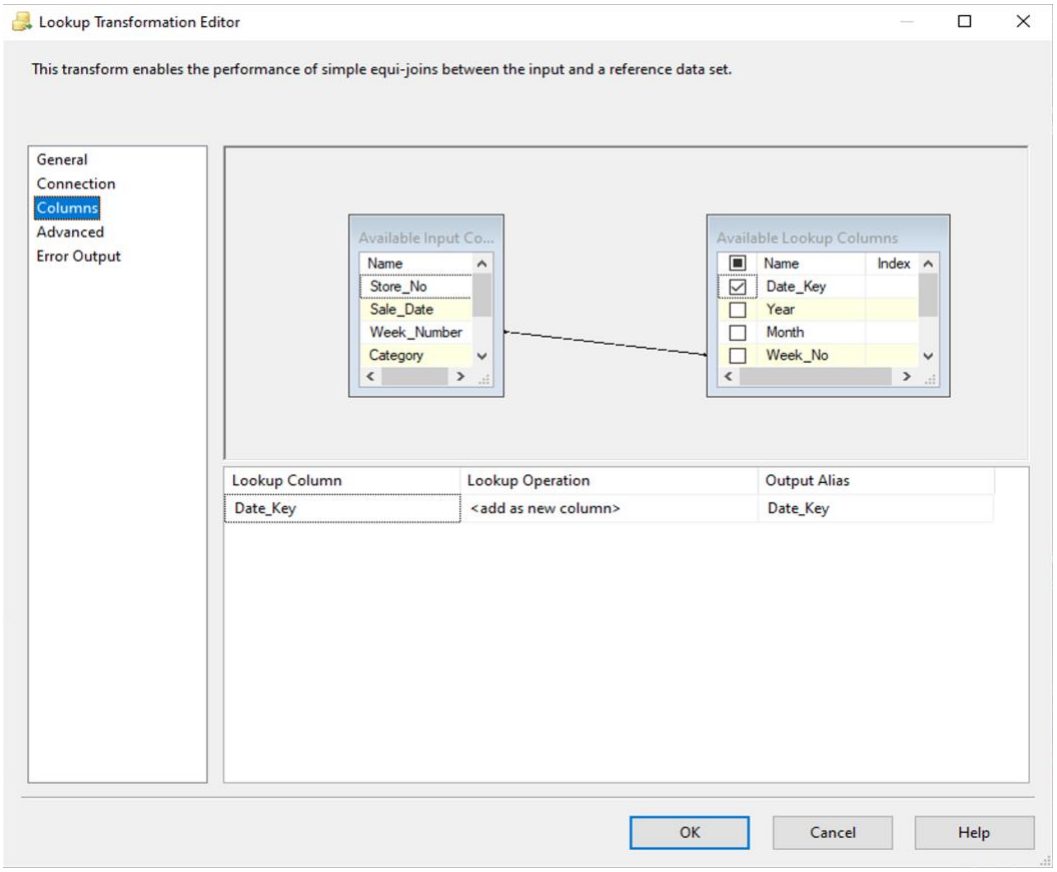

Lookup Transformation Editor
This transform enables the performance of simple equi-joins between the input and a reference data set.
General
Connection
Columns
Advanced
Error Output
Available Input Columns
Name
Store_No
UPC
Week
No_of_Units_Sold
Bundle_Size
Bundle_Price
Sales
Profit
Date_Key
Available Lookup Columns
Name
Index
Store_Key
Store_No
City
Zone
Price_Tier
Poverty
Cust_Count
Lookup Column
Lookup Operation
Output Alias
Store_Key
<add as new column>
Store_Key
OK
Cancel
Help

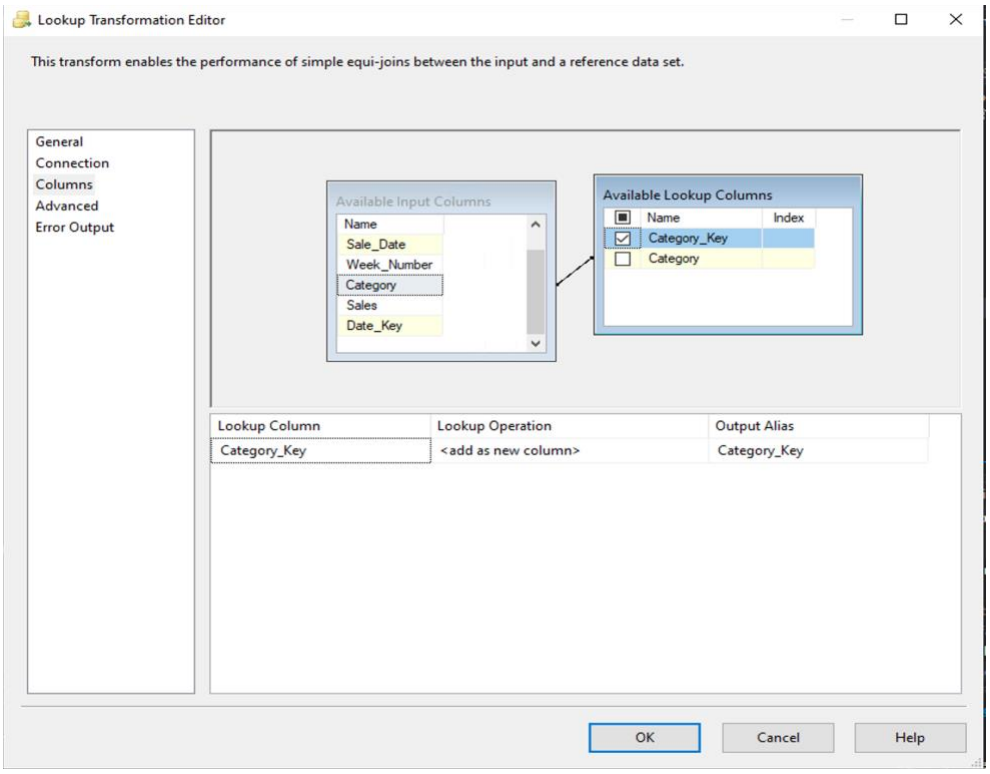

OLE DB Destination Editor
Configure the properties used to insert data into a relational database using an OLE DB provider.
Connection Manager
Mappings
Error Output
Available Input Columns
Name
Store_No
UPC
Week
No_of_Units_Sold
Bundle_Size
Bundle_Price
Sales
Profit
Date_Key
Product_Key
Available Destination...
Name
Date_Key
Product_Key
Store_Key
Bundle_Price
No_Of_Units_Sold
Bundle_Size
Sales
Input Column
Destination Column
Date_Key
Product_Key
Store_Key
Bundle_Price
No_of_Units_Sold
No_Of_Units_Sold
Bundle_Size
Sales
Profit
OK
Cancel
Help

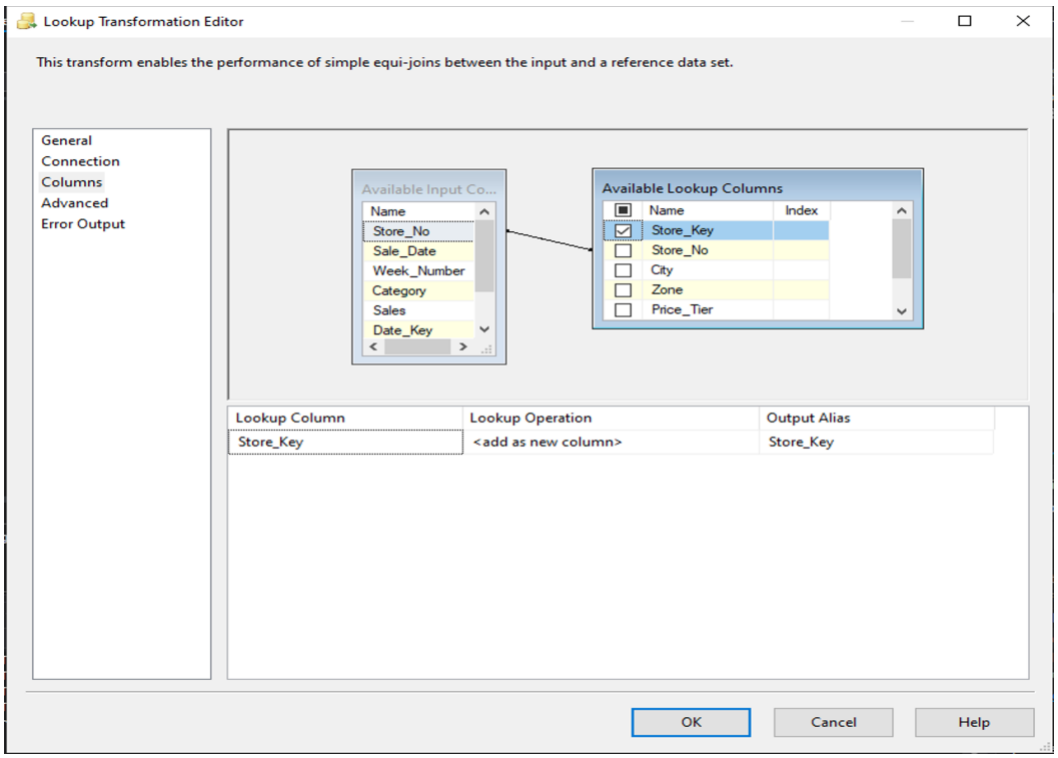

StoreTable.dtsx [Design]
WeekDecode.dtsx [Design]
WeekDecode_Final.dtsx [Design]
FactProductSales.dtsx [Design]
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task
Movement Staging Table
6,397,401 rows
Lookup in dimTime
Lookup Match Output (6,397,401 rows)
Lookup in dimProduct
Lookup Match Output (6,397,401 rows)
Lookup in dimStore
Lookup Match Output (6,397,401 rows)
FactProductSales

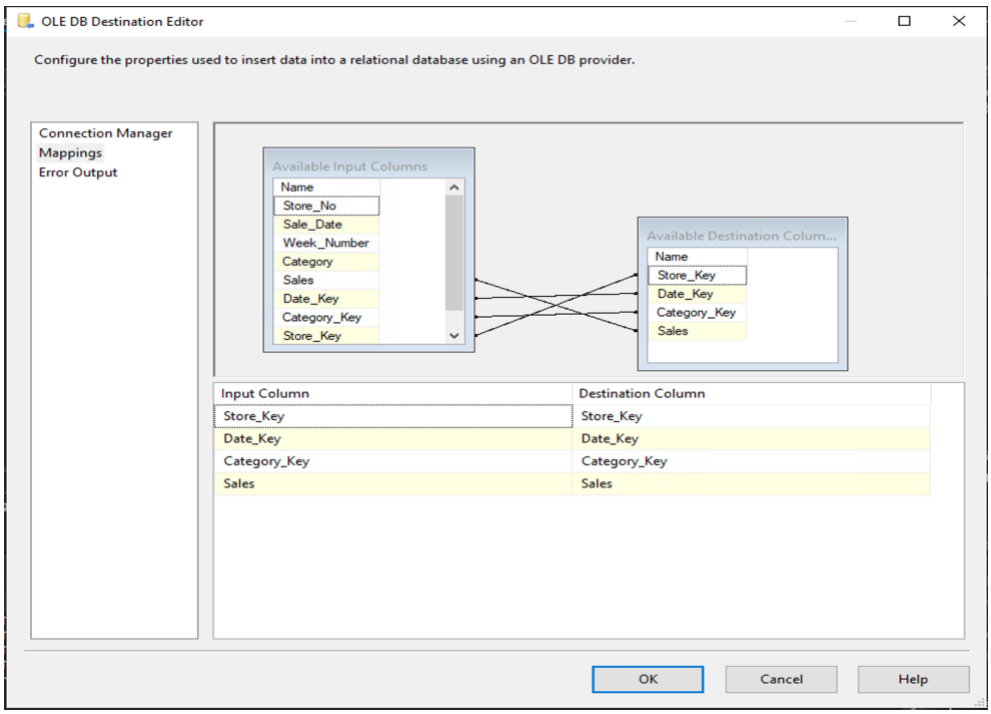
FactCategorySales:

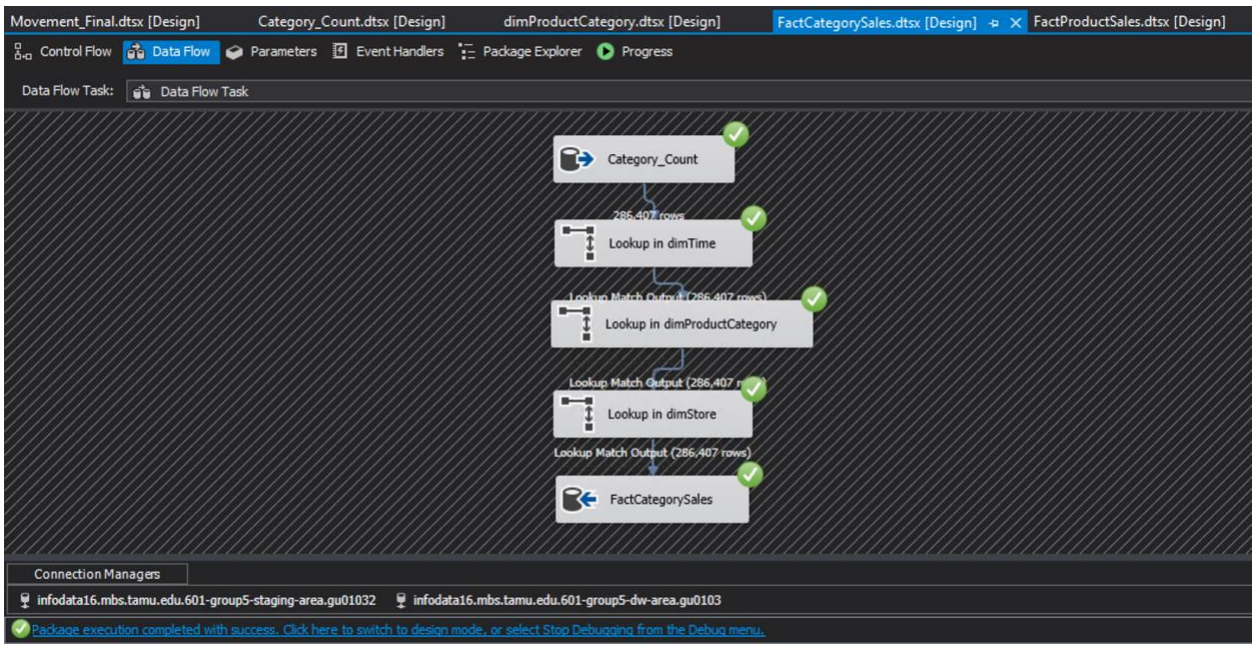



Lookup Transformation Editor
This transform enables the performance of simple equi-joins between the input and a reference data set.
General
Connection
Columns
Advanced
Error Output
Available Input Co...
Name
Store_No
Sale_Date
Week_Number
Category
Available Lookup Columns
Name
Index
Date_Key
Year
Month
Week_No
Lookup Column
Lookup Operation
Output Alias
Date_Key
<add as new column>
Date_Key
OK
Cancel
Help

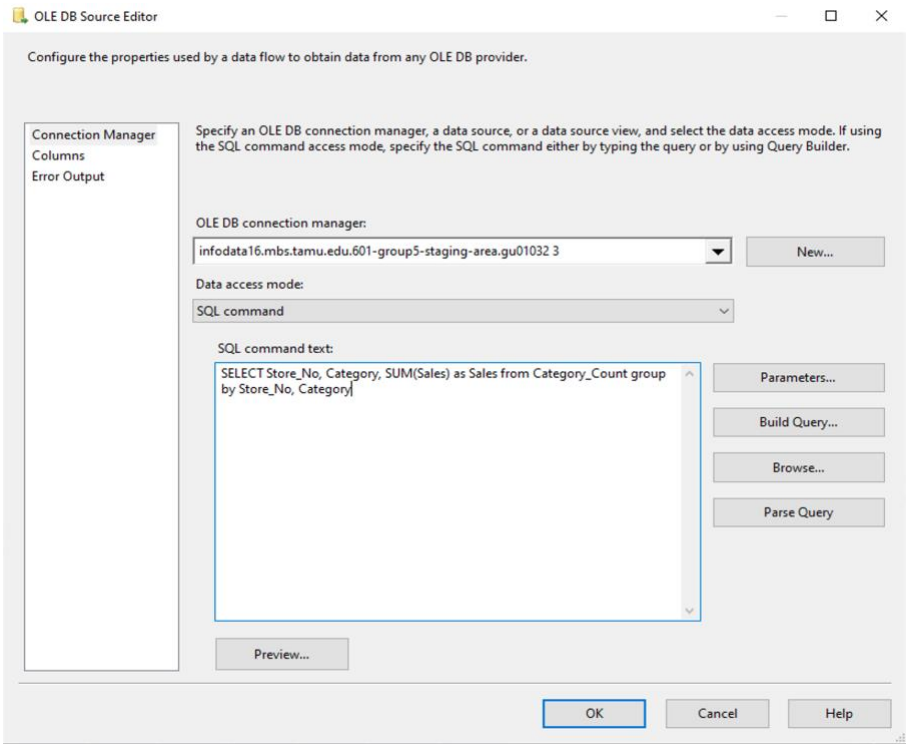

Lookup Transformation Editor
This transform enables the performance of simple equi-joins between the input and a reference data set.
General
Connection
Columns
Advanced
Error Output
Available Input Columns
Name
Sale_Date
Week_Number
Category
Sales
Date_Key
Available Lookup Columns
Name
Index
Category_Key
Category
Lookup Column
Lookup Operation
Output Alias
Category_Key
<add as new column>
Category_Key
OK
Cancel
Help

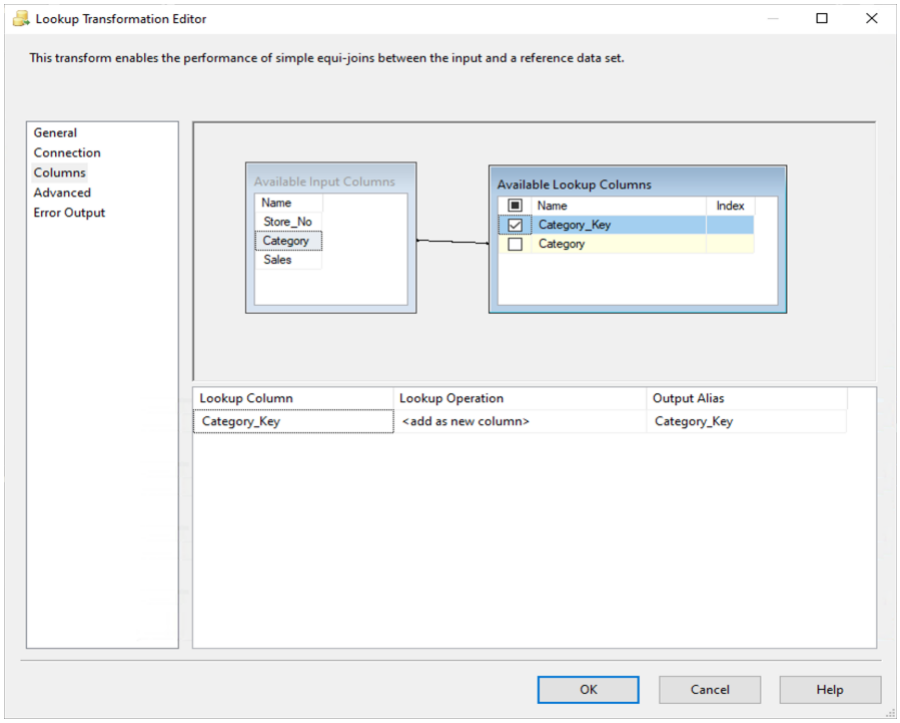

Lookup Transformation Editor
This transform enables the performance of simple equi-joins between the input and a reference data set.
General
Connection
Columns
Advanced
Error Output
Available Input Co...
Name
Store_No
Sale_Date
Week_Number
Category
Sales
Date_Key
Available Lookup Columns
Name
Index
Store_Key
Store_No
City
Zone
Price_Tier
Lookup Column
Lookup Operation
Output Alias
Store_Key
<add as new column>
Store_Key
OK
Cancel
Help

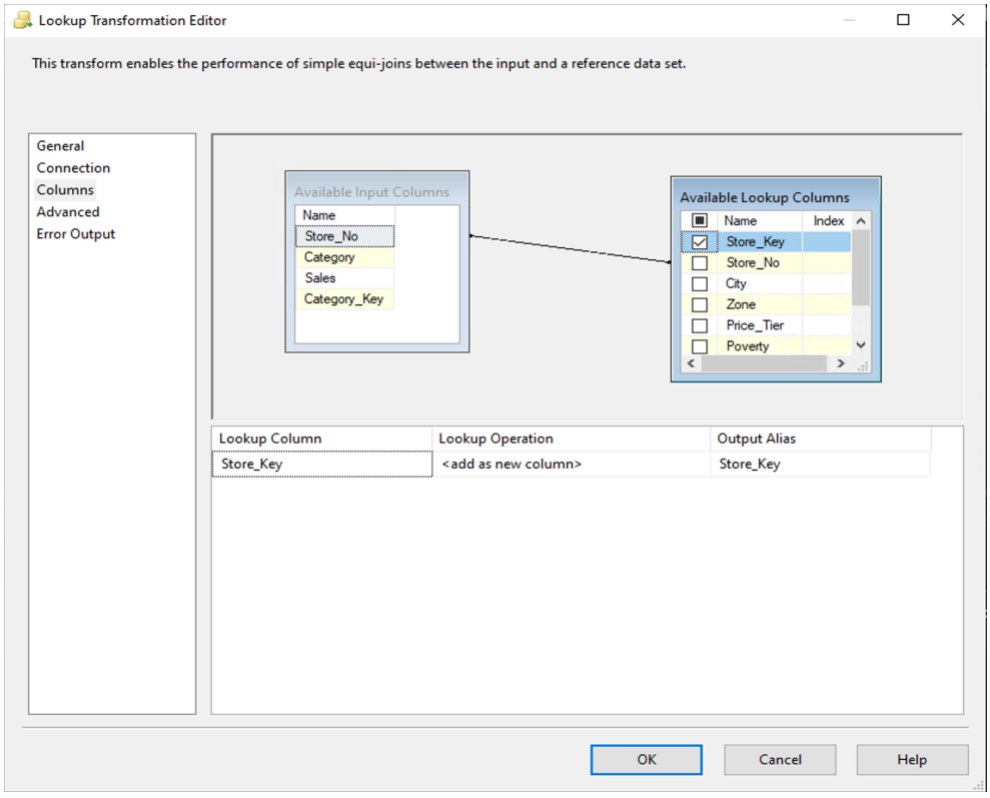

OLE DB Destination Editor
Configure the properties used to insert data into a relational database using an OLE DB provider.
Connection Manager
Mappings
Error Output
Available Input Columns
Name
Store_No
Sale_Date
Week_Number
Category
Sales
Date_Key
Category_Key
Store_Key
Available Destination Colum...
Name
Store_Key
Date_Key
Category_Key
Sales
Input Column
Destination Column
Store_Key
Store_Key
Date_Key
Date_Key
Category_Key
Category_Key
Sales
Sales
OK
Cancel
Help

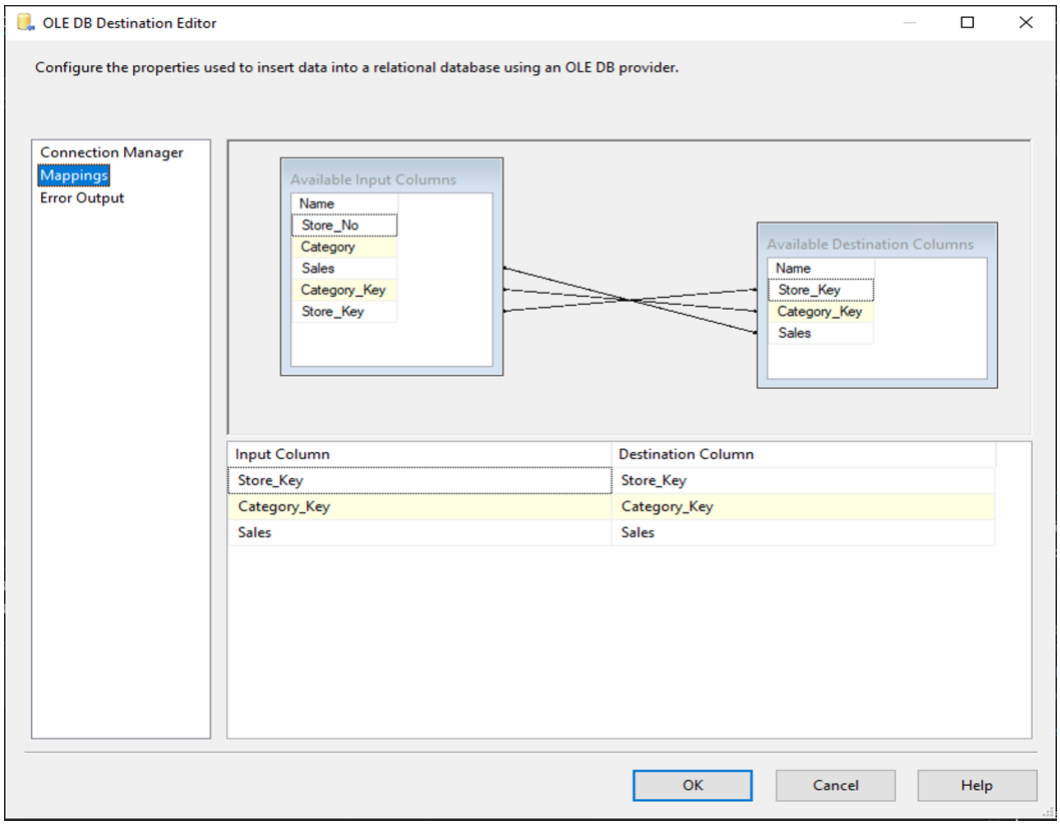


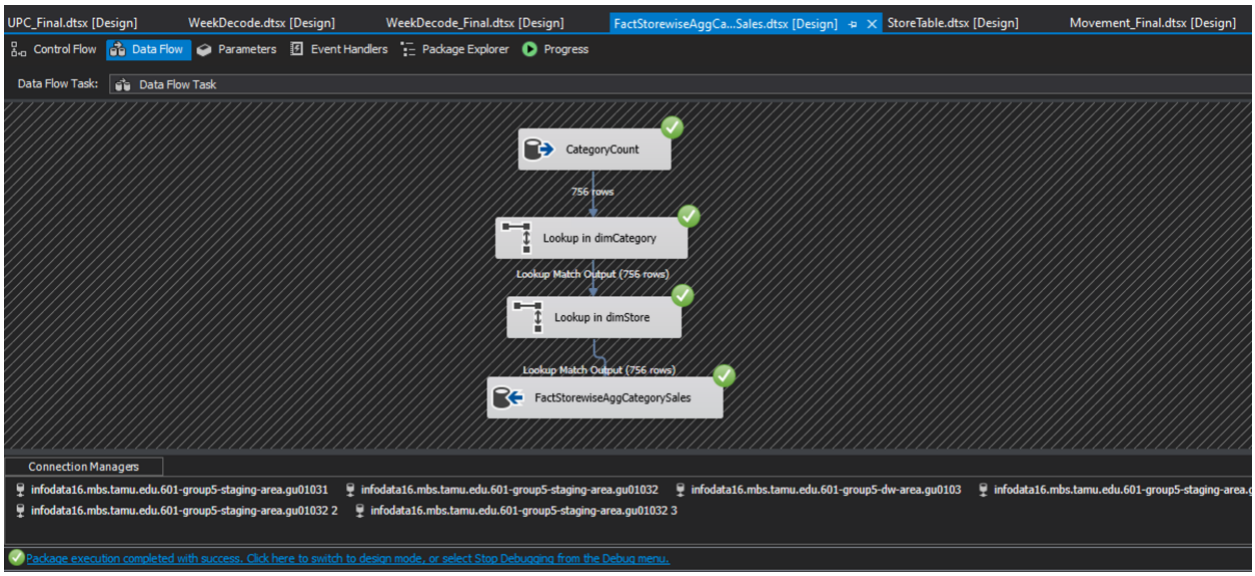
FactStorewiseAggCategorySales:




Lookup Transformation Editor
This transform enables the performance of simple equi-joins between the input and a reference data set.
General
Connection
Columns
Advanced
Error Output
Available Input Columns
Name
Store_No
Category
Sales
Available Lookup Columns
Name
Index
Category_Key
Category
Lookup Column
Lookup Operation
Output Alias
Category_Key
<add as new column>
Category_Key
OK
Cancel
Help


Lookup Transformation Editor
This transform enables the performance of simple equi-joins between the input and a reference data set.
General
Connection
Columns
Advanced
Error Output
Available Input Columns
Name
Store_No
Category
Sales
Category_Key
Available Lookup Columns
Name
Index
Store_Key
Store_No
City
Zone
Price_Tier
Poverty
Lookup Column
Lookup Operation
Output Alias
Store_Key
<add as new column>
Store_Key
OK
Cancel
Help


OLE DB Destination Editor
Configure the properties used to insert data into a relational database using an OLE DB provider.
Connection Manager
Mappings
Error Output
Available Input Columns
Name
Store_No
Category
Sales
Category_Key
Store_Key
Available Destination Columns
Name
Store_Key
Category_Key
Sales
Input Column
Destination Column
Store_Key
Store_Key
Category_Key
Category_Key
Sales
Sales
OK
Cancel
Help


UPC_Final.dtsx [Design]
WeekDecode.dtsx [Design]
WeekDecode_Final.dtsx [Design]
FactStorewiseAggCa...Sales.dtsx [Design]
StoreTable.dtsx [Design]
Movement_Final.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Data Flow Task
CategoryCount
756 rows
Lookup in dimCategory
Lookup Match Output (756 rows)
Lookup in dimStore
Lookup Match Output (756 rows)
FactStorewiseAggCategorySales
Connection Managers
infodata16.mbs.tamu.edu.601-group5-staging-area.gu01031
infodata16.mbs.tamu.edu.601-group5-staging-area.gu01032
infodata16.mbs.tamu.edu.601-group5-dw-area.gu0103
infodata16.mbs.tamu.edu.601-group5-staging-area.gu01032 2
infodata16.mbs.tamu.edu.601-group5-staging-area.gu01032 3
Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

20 GRANULARITY OF DATA MARTS

We have 3 different Data Marts in our Warehouse. The Granularity in each of those is explained below:

1. Product Sales Data Mart – This data mart records sales at the level of a product, on a particular date, at a particular store.

2. Product Category Sales Data Mart – This data mart records sales at the level of a product category (those categories that don't have a sales record at the UPC level), on a particular date, at a particular store

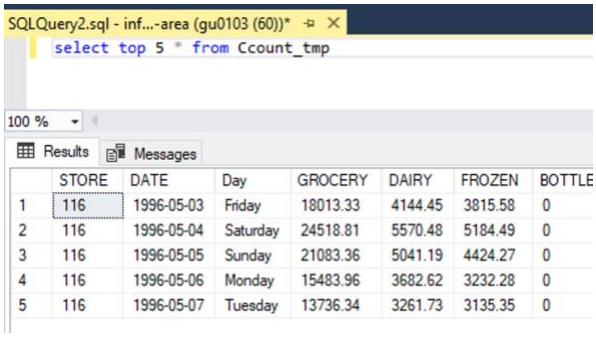
3. Store-wise Aggregated Category Sales Data Mart – This data mart records aggregated sales at the level of a product category, at a store. The fact table in this Data Mart is an aggregated fact table.

21 TABLE CONTENT BEFORE AND AFTER ETL IN STAGING

11.1 BEFORE ETL

Before ETL, data in our staging table resides in temp tables. Below screenshots show the data before cleaning in the temp tables.

Ccount_tmp:



	STORE	DATE	Day	GROCERY	DAIRY	FROZEN	BOTTLE
1	116	1996-05-03	Friday	18013.33	4144.45	3815.58	0
2	116	1996-05-04	Saturday	24518.81	5570.48	5184.49	0
3	116	1996-05-05	Sunday	21083.36	5041.19	4424.27	0
4	116	1996-05-06	Monday	15483.96	3682.62	3232.28	0
5	116	1996-05-07	Tuesday	13736.34	3261.73	3135.35	0

INFODATA16.601-gr...- dbo.Ccount_tmp | SQLQuery2.sql - inf...-area (gu010

Column Name	Data Type	Allow Nulls
STORE	varchar(50)	☑
DATE	date	☑
Day	varchar(50)	☑
GROCERY	varchar(50)	☑
DAIRY	varchar(50)	☑
FROZEN	varchar(50)	☑
BOTTLE	varchar(50)	☑
MVPCLUB	varchar(50)	☑
GROCCOUP	varchar(50)	☑
MEAT	varchar(50)	☑
MEATFROZ	varchar(50)	☑
MEATCOUP	varchar(50)	☑
FISH	varchar(50)	☑
FISHCOUP	varchar(50)	☑
PROMO	varchar(50)	☑
PROMCOUP	varchar(50)	☑
PRODUCE	varchar(50)	☑
[BULK]	varchar(50)	☑

Movement_tmp:

SQLQuery2.sql - inf...-area (gu0103 (60))*

```
select top 5 * from Movement_tmp
```

100 %

Results | Messages

	Store	UPC	Week	Move	Qty	Price	Sale	Profit	OK	Sales	Total_Profit
1	109	1111189086	391	0	1	0.00		0	1	0.0000000000000	0.0000000000
2	109	1111189086	392	0	1	0.00		0	1	0.0000000000000	0.0000000000
3	109	1111189086	393	0	1	0.00		0	1	0.0000000000000	0.0000000000
4	109	1111189086	394	0	1	0.00		0	1	0.0000000000000	0.0000000000
5	109	1111189086	395	0	1	0.00		0	1	0.0000000000000	0.0000000000

INFODATA16.601-...dbo.Movement_tmp | SQLQuery2.sql - inf...-area (g

Column Name	Data Type	Allow Nulls
Store	int	☑
UPC	varchar(50)	☑
Week	int	☑
Move	int	☑
Qty	int	☑
Price	decimal(10, 2)	☑
Sale	varchar(50)	☑
Profit	varchar(50)	☑
OK	varchar(50)	☑
Sales		☑
Total_Profit		☑
		☐

WeekDecode_tmp:

SQLQuery2.sql - inf...-area (gu0103 (60))*

```
select top 5 * from WeekDecode_tmp
```

100 %

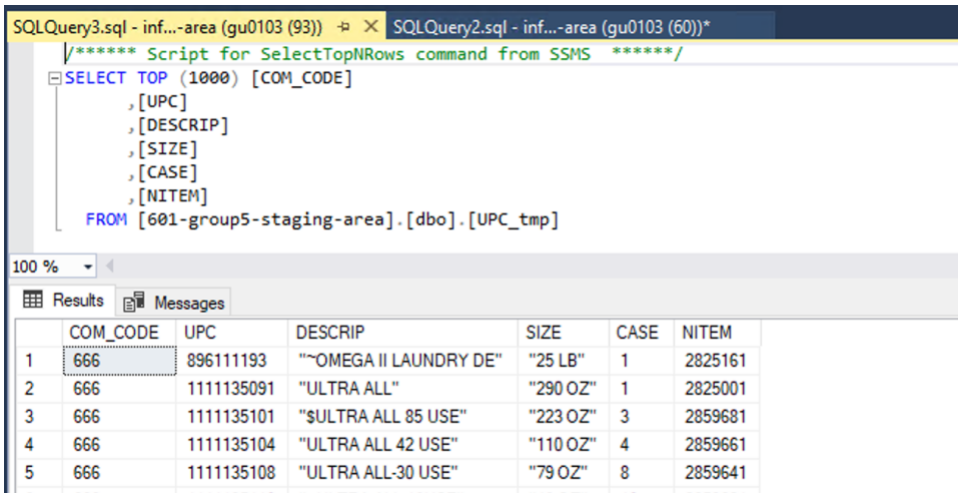
Results | Messages

	Week_number	Start	End	Event_name	Quarter	Quarter-number	Month	Month_number	Year	Month_Part	Year_Part
1	1	1989-09-14	1989-09-20				9		1989	9	1989
2	2	1989-09-21	1989-09-27				9		1989	9	1989
3	3	1989-09-28	1989-10-04				9		1989	9	1989
4	4	1989-10-05	1989-10-11				10		1989	10	1989
5	5	1989-10-12	1989-10-18				10		1989	10	1989

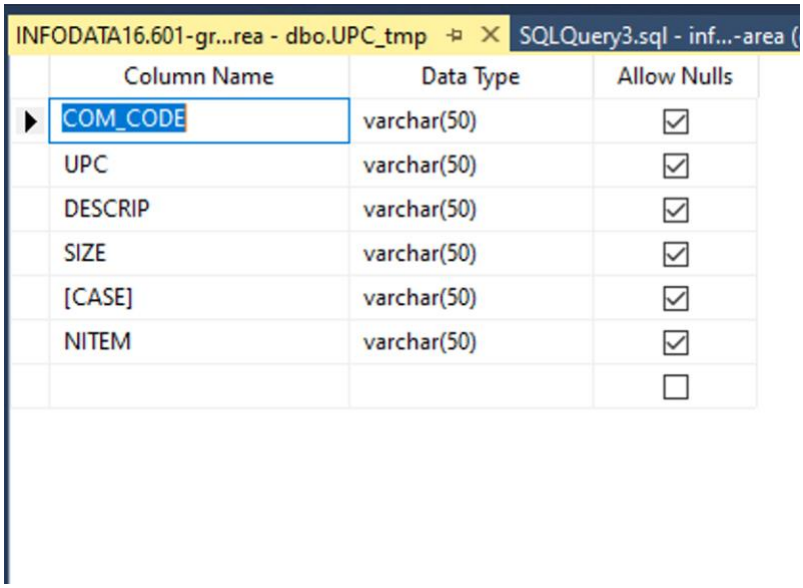
INFODATA16.601-g...o.WeekDecode_tmp | SQLQuery2.sql - inf...-area (g

Column Name	Data Type	Allow Nulls
Week_number	int	☑
Start	date	☑
[End]	date	☑
Event_name	varchar(50)	☑
Quarter	varchar(50)	☑
[Quarter-number]	varchar(50)	☑
Month	varchar(50)	☑
Month_number	varchar(50)	☑
Year	varchar(50)	☑
Month_Part		☑
Year_Part		☑
		☐

UPC_tmp



	COM_CODE	UPC	DESCRIP	SIZE	CASE	NITEM
1	666	896111193	"~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	1111135108	"ULTRA ALL-30 USE"	"79 OZ"	8	2859641

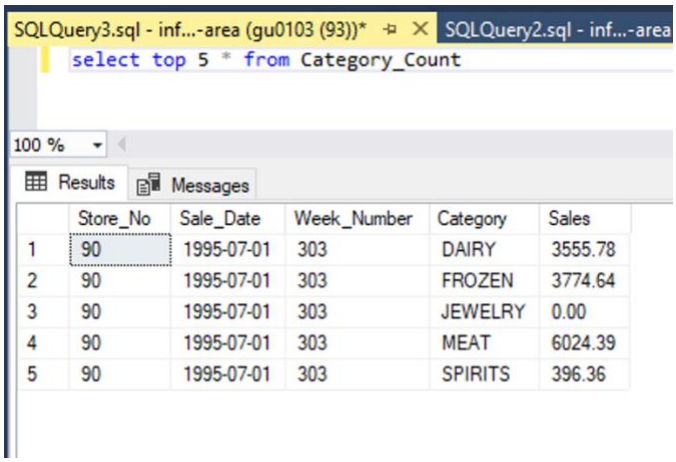


Column Name	Data Type	Allow Nulls
COM_CODE	varchar(50)	☑
UPC	varchar(50)	☑
DESCRIP	varchar(50)	☑
SIZE	varchar(50)	☑
[CASE]	varchar(50)	☑
NITEM	varchar(50)	☑
		☐

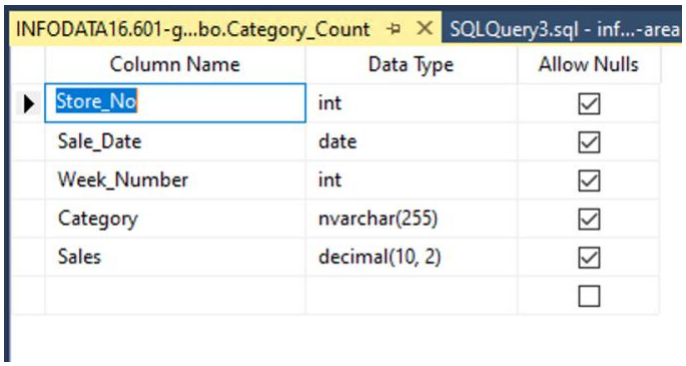
11.2 AFTER ETL

After ETL data is stored in the final staging tables. Below screenshots show the sample data and the data description in these tables.

Category_Count

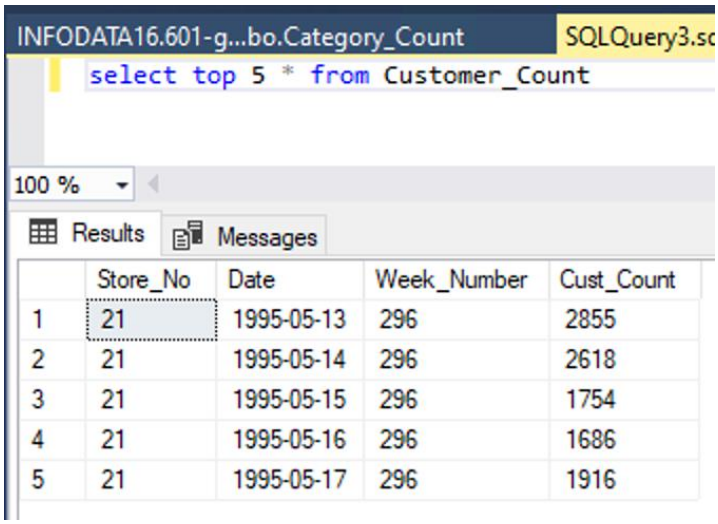


	Store_No	Sale_Date	Week_Number	Category	Sales
1	90	1995-07-01	303	DAIRY	3555.78
2	90	1995-07-01	303	FROZEN	3774.64
3	90	1995-07-01	303	JEWELRY	0.00
4	90	1995-07-01	303	MEAT	6024.39
5	90	1995-07-01	303	SPIRITS	396.36

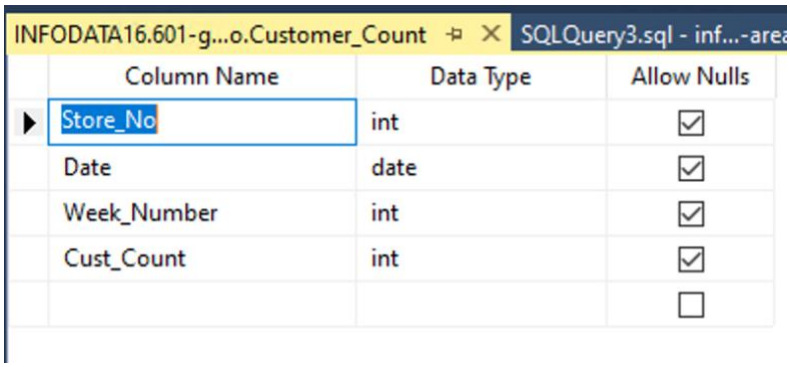


Column Name	Data Type	Allow Nulls
Store_No	int	☑
Sale_Date	date	☑
Week_Number	int	☑
Category	nvarchar(255)	☑
Sales	decimal(10, 2)	☑
		☐

Customer_Count

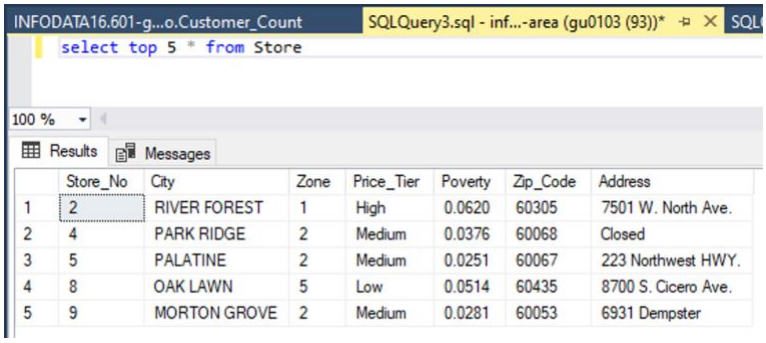


	Store_No	Date	Week_Number	Cust_Count
1	21	1995-05-13	296	2855
2	21	1995-05-14	296	2618
3	21	1995-05-15	296	1754
4	21	1995-05-16	296	1686
5	21	1995-05-17	296	1916



Column Name	Data Type	Allow Nulls
Store_No	int	☑
Date	date	☑
Week_Number	int	☑
Cust_Count	int	☑
		☐

Store



	Store_No	City	Zone	Price_Tier	Poverty	Zip_Code	Address
1	2	RIVER FOREST	1	High	0.0620	60305	7501 W. North Ave.
2	4	PARK RIDGE	2	Medium	0.0376	60068	Closed
3	5	PALATINE	2	Medium	0.0251	60067	223 Northwest HWY.
4	8	OAK LAWN	5	Low	0.0514	60435	8700 S. Cicero Ave.
5	9	MORTON GROVE	2	Medium	0.0281	60053	6931 Dempster

INFODATA16.601-gr...g-area - dbo.Store | INFODATA16.601-g...o.Cu

Column Name	Data Type	Allow Nulls
Store_No	int	☐
City	varchar(255)	☑
Zone	varchar(10)	☑
Price_Tier	varchar(50)	☑
Poverty	decimal(10, 4)	☑
Zip_Code	varchar(50)	☑
Address	varchar(255)	☑
		☐

WeekDecode

INFODATA16.601-gr...g-area - dbo.Store | SQLQue

```
select top 5 * from WeekDecode where S
```

100 %

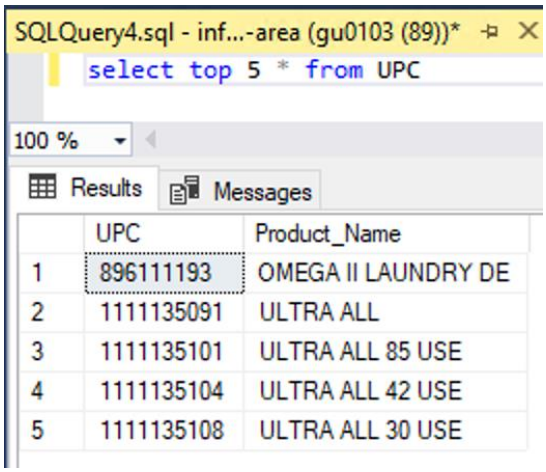
Results | Messages

	Year	Month	Week_Number	Special_Event
1	1989	10	7	Halloween
2	1989	11	11	Thanksgiving
3	1989	12	15	Christmas
4	1989	12	16	New-Year
5	1990	2	23	Presidents Day

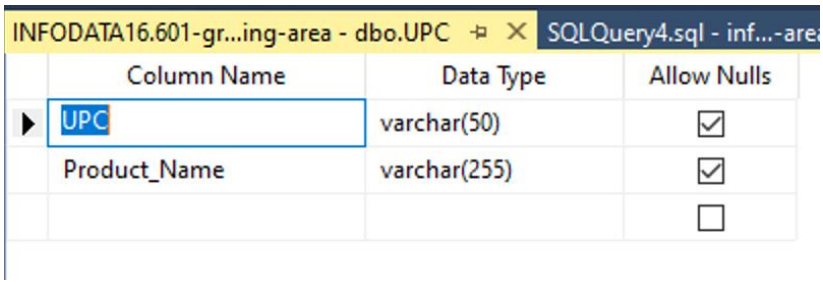
INFODATA16.601-gr...- dbo.WeekDecode

Column Name	Data Type	Allow Nulls
Year	int	☑
Month	int	☑
Week_Number	int	☑
Special_Event	varchar(255)	☑
		☐

UPC

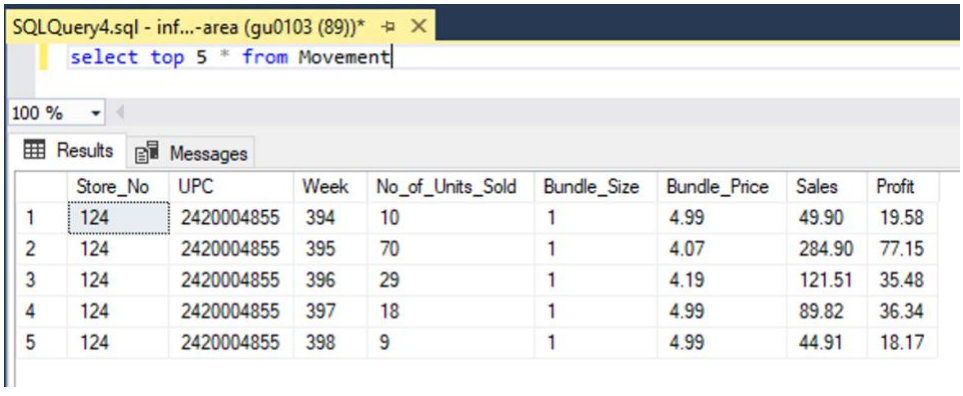


	UPC	Product_Name
1	896111193	OMEGA II LAUNDRY DE
2	1111135091	ULTRA ALL
3	1111135101	ULTRA ALL 85 USE
4	1111135104	ULTRA ALL 42 USE
5	1111135108	ULTRA ALL 30 USE

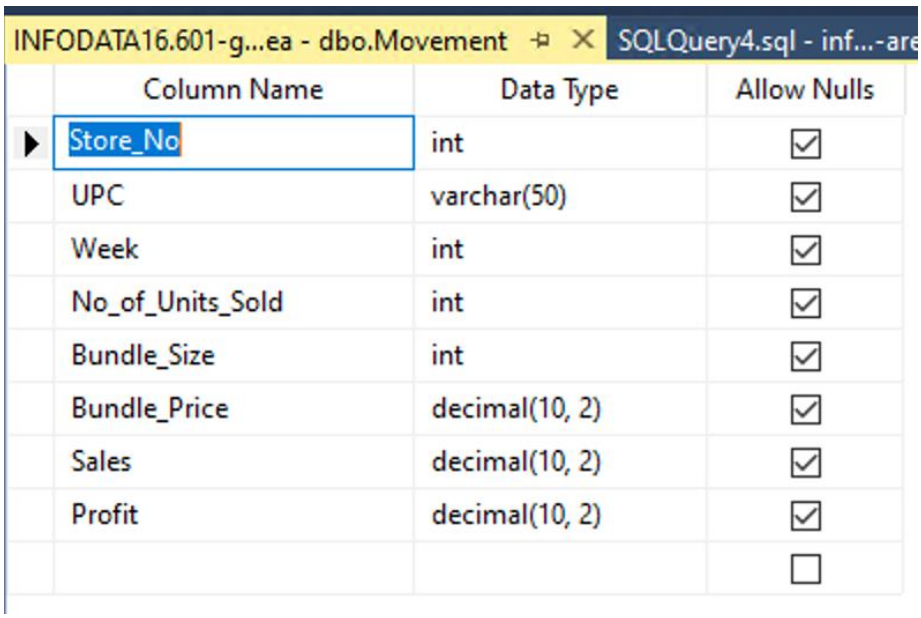


Column Name	Data Type	Allow Nulls
UPC	varchar(50)	☑
Product_Name	varchar(255)	☑
		☐

Movement

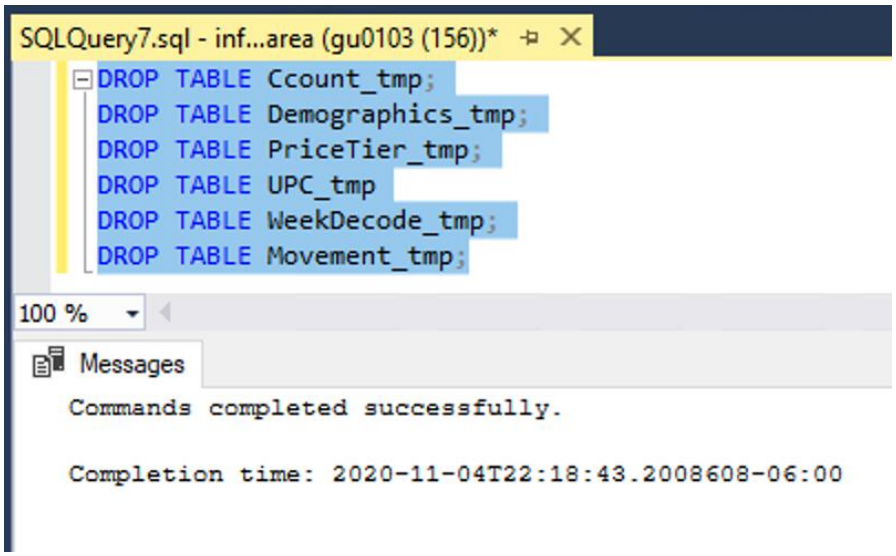


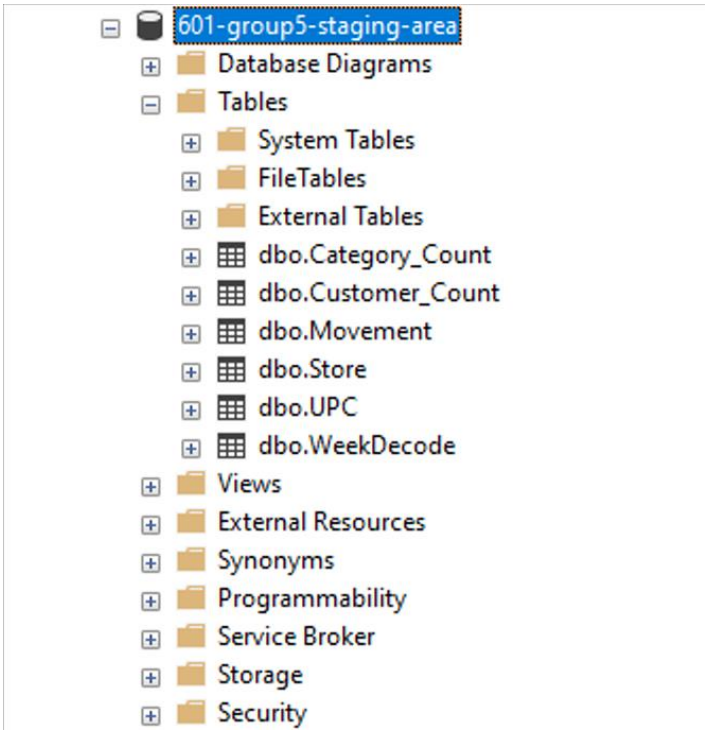
	Store_No	UPC	Week	No_of_Units_Sold	Bundle_Size	Bundle_Price	Sales	Profit
1	124	2420004855	394	10	1	4.99	49.90	19.58
2	124	2420004855	395	70	1	4.07	284.90	77.15
3	124	2420004855	396	29	1	4.19	121.51	35.48
4	124	2420004855	397	18	1	4.99	89.82	36.34
5	124	2420004855	398	9	1	4.99	44.91	18.17



Column Name	Data Type	Allow Nulls
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)	☑
		☐

22 DELETION OF TEMP TABLES





REFERENCES

1. Daniel Levy, Georg Müller, Haipeng Allan Chen, Mark Bergen, Shantanu Dutta. “Holiday Price Rigidity and Cost of Price Adjustment”. Economica, Wiley, 2010, 77 (305), pp.172-198
2. Nevo, Aviv and Catherine Wolfram, “Why do Manufacturers Issue Coupons? An Empirical Analysis of Breakfast Cereals," The RAND Journal of Economics, 33 (Summer 2002): 319-339.
3. Kamakura, Wagner A. and Wooseong Kang, "Chain-wide and Store-level Analysis for Cross-category Management," Journal of Retailing, 83 (February 2007): 159- 170.
4. Toro-González, Daniel, Jill J. McCluskey, and Ron C. Mittelhammer. "Beer Snobs Do Exist: Estimation of Beer Demand by Type." Journal of Agricultural and Resource Economics (JARE) (2014).
5. https://en.wikipedia.org/wiki/Dominick's
6. https://research.chicagobooth.edu/marketing/databases/dominicks/docs/1994-shelfmanagement.pdf