Data warehouses and business intelligence teamwork

Table of contents

Introduction 2
Team members 2
Description 2
Use Cases 2
The task 2
Source dataset 2
Stage database 3
Each table and its main columns 3
Data Warehouse Database 5
Dimensional model 7
Fact table 7
Dimension tables 8
ETL 8
Extract 8
Transform 11
Load 15
Reports 25
1) Number of sales per countries/states 25
2) Number of sales for promotions 25
3) What type of credit card is paid by state/province 26
4) Number of online and in-store purchases 26
5) Tax types by country and province 27
6) Reasons for purchases by area 27

INTRODUCTION

Team members

- Bodrogi Lili
- Elek Kitti Laura
- Kalmár Nóra Kata
- Lukács Eszter
- Naszály Noémi

Description

The goal of the project is to build a data warehouse that will enable us to track and analyze the company's sales from various perspectives.

Use Cases

The customer wants to analyze the following:

1) How many sales were made per state/province?
2) In the case of promotion, what changes can be observed in the number of the sales?
3) What type of bank card are people mostly used within each state/province?
4) How many online and non-online purchases have been made in the provinces of the defined countries?
5) What types of taxes are there per country and per province (and how can we calculate our prices accordingly)?
6) What were the motivations for purchases by people per region, and within that, per province?

THE TASK

Source dataset

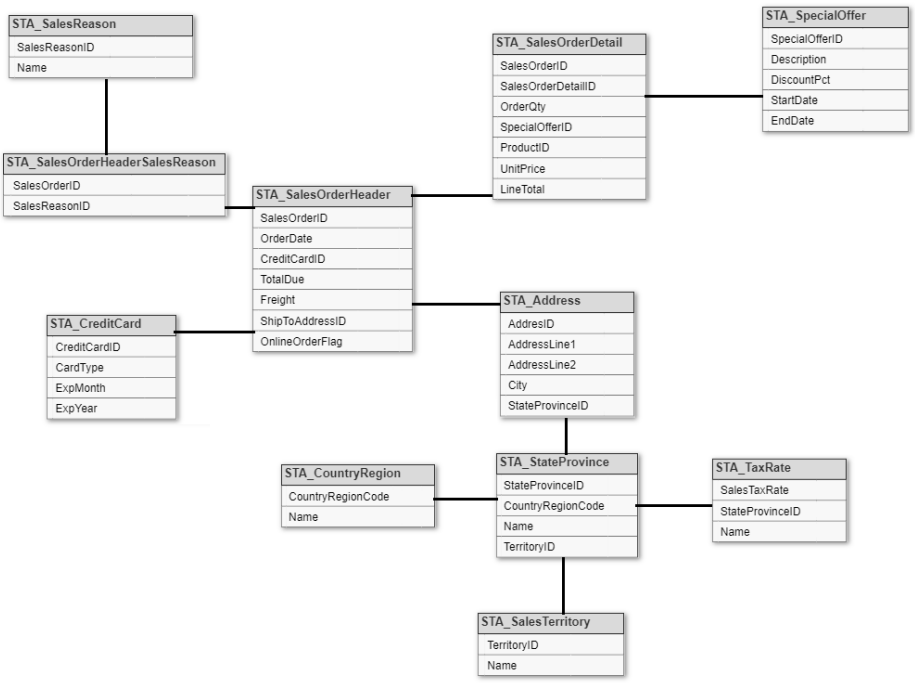
The AdventureWorks database was our choice as a base dataset.

The structure of the source database is shown in the following link:
https://txtrainingstore.blob.core.windows.net/db-backup-aw2019/AdventureWorksERD

The tables used from the source database are illustrated by the database structure of the Stage and Data Warehouse layers of the ETL process.

Stage database

The tables from the source database are loaded into the Stage database with their structure unchanged. The only difference is that the column types are converted to VARCHAR. The tables used and their relationships are shown in the figure below:



Each table and its main columns

STA_SalesOrderHeader: Contains header information for each sale.

SalesOrderID: Unique identifier of the sale.
OrderDate: Date of sale.
CreditCardID: Unique identifier of the credit card used, reference to the CreditCard table.
TotalDue: The total value of the product sold in dollars.
Freight: The shipping cost.
ShipToAddressID: Ship-to Address. Link to address table.
OnlineOrderFlag: Was it an online order. (0 = no, 1 = yes)

STA_SalesTerritory: Data from the area.

TerritoryID: Unique identifier of the area.
Name: Area name.

STA_CountryRegion: Table containing countries.

CountryRegionCode: Unique identifier of the country.
Name: The name of the country.

STA_Address: Table containing geographical addresses.

AddressID: Unique identifier of the address.
AddressLine1: First line of address (street, house number).
AddressLine2: Second line of address (if the street and house number is not precise enough).
City: City name.
StateProvinceID: Unique identifier of states or domains. Reference to the StateProvince table.

STA_StateProvince: A table containing states / provinces.

StateProvinceID: Unique identifier of the state or domain.
CountryRegionCode: The official code of the country.
Name: The name of the state or province.
TerritoryID: Unique identifier of the area. Reference to the SalesTerritory table.

STA_SalesOrderHeaderSalesReason: This represents the relationship between product purchases and the reasons for buying them. Essentially, it's a linking table between SalesOrderHeader and SalesReason, which is necessary due to the many-to-many relationship.

SalesOrderID: Unique identifier of the purchase. Reference to the SalesOrderHeader table.
SalesReasonID: Identification of the reasons for purchase. Reference to the SalesReason table.

STA_SalesReason: What are the reasons for buying.

SalesReasonID: Unique identification code for the reasons.
Name: Name of the reason for the purchase.

STA_SalesOrderDetail: This is where the products sold in individual sales are recorded. In other words, it's the invoice detail, listing products item by item.

SalesOrderID: Unique codes for sales. Reference to the SalesOrderHeader table.
SalesOrderDetailID: Unique identifier of the drill-down.
OrderQty: How many of the products were sold at the time of sale.
SpecialOfferID: At the time of sale, the promotion was applied to the product. This is a reference to the SpecialOffer table.
ProductID: Unique identifier of the products sold. Reference to the Product table.
UnitPrice: The price of the product sold.
LineTotal: The total amount to be paid (the price of products+shipping)

STA_SpecialOffer: A table containing the actions and their details.

SpecialOfferID: Unique identifier of the action.
Description: Description of the action.
DiscountPct: Discount rate as a percentage.
StartDate: When the discount is valid.
EndDate: How long the discount is valid.

STA_CreditCard: A table containing information about the credit cards used by customers when paying.

CreditCardID: Unique identifier of the credit card.
CardType: Type of card.
ExpMonth: Expiration Month.
ExpYear: Expiration Year.

STA_TaxRate: Table with different taxes. It contains data by state/province.

SalesTaxRate: Unique identifier of the tax.
StateProvinceID: Unique identifier of the state or domain. Reference to the StateProvince table.
Name: The name of the tax.

Data Warehouse Database

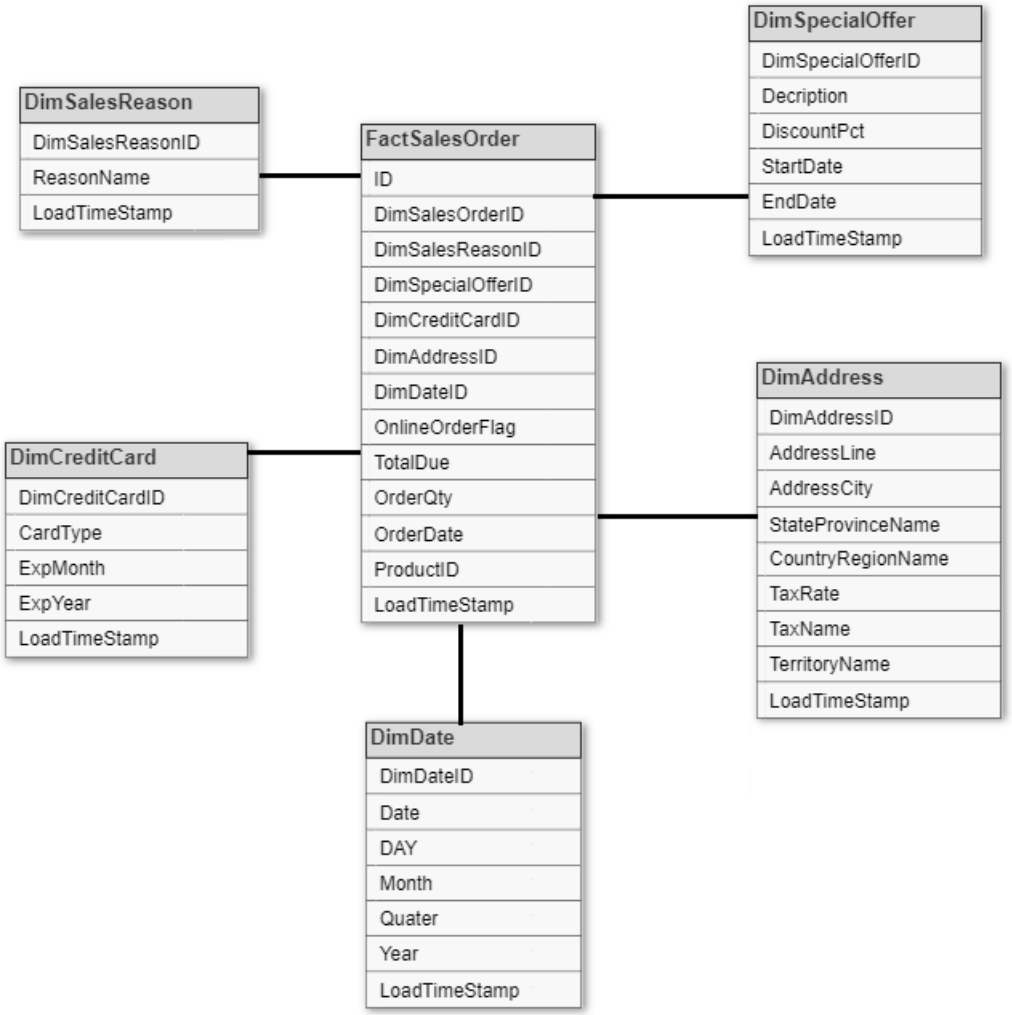
The structure of the 2nd layer database is identical to the 1st Stage layer database, with two exceptions:

- Instead of VARCHAR data types, the columns now have data types appropriate for the specific data. (Refer to the SQL script in the appendix for details.)
- The tables include ValidFrom and ValidTo columns. These date-type columns were created for historization purposes. If a row is modified in the source database, its new version must also be stored in the DW layer. ValidFrom indicates from when, and ValidTo indicates until when, the given values were valid for that specific row ID. The ValidTo column of the latest version contains a NULL value until any of its values are modified.

  Contains:
  - SalesTaxRate
  - SalesReason
  - SalesOrderHeader
  - SalesOrderDetail
  - SpecialOffer
  - CountryRegion

Dimensional model

The planned Star Scheme model is illustrated in the following illustration:



The mapping of individual columns to their corresponding source database columns is detailed in the mapping.xlsx Excel table, which is included in the appendix.

Fact table

FactSalesOrder: Contains sales data, with item-level granularity.

- Its sources are the SalesOrderHeader, SalesOrderDetail, and SalesOrderHeaderSalesReason tables.
- If someone indicated multiple reasons (SalesReason) for a purchase, that purchase will appear with multiple rows in the fact table. (For example, someone buys 3 different products and states the reason as seeing it on TV, but also came in because of a promotion.
- In this case, the invoice ID (SalesOrderHeader) gets sequence number 1, and the 3 items in SalesOrderDetail get sequence numbers 1, 2, and 3 as details of invoice number 1.

- Finally, these 3 rows enter the fact table twice due to the indicated reasons, so we can describe the purchase with 6 rows in our dimensional model.)

Dimension tables

DimSalesReason: A table with reasons to purchase.
DimSpecialOffer: The action table.
DimAddress: A table with detailed addresses in the form of: address (street, house number, city), name of state or province, name of the area, country name. It also includes the rate and name of taxes, which are usually levied by state/province.
DimCreditCard: Credit card details.
DimDate: Dimension table with dates.

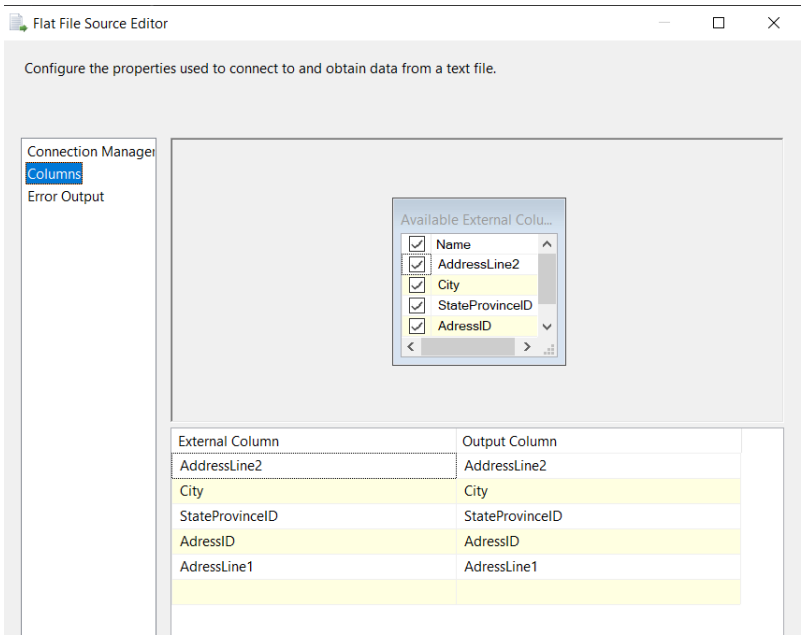
ETL

Extract

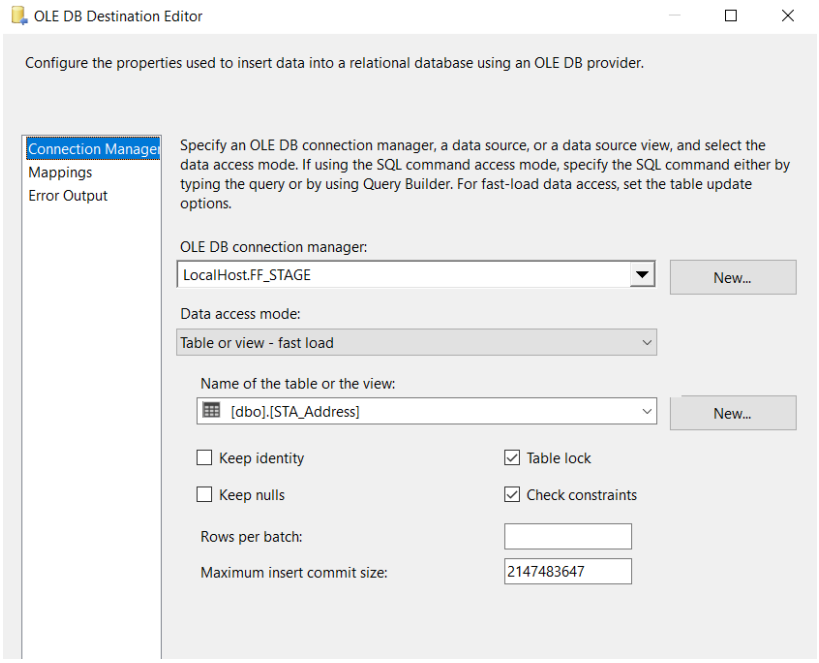
During the Extract process, all data and columns from every table were transferred to the STAGE database. No transformations or conversions occur; all data is converted to VARCHAR type. The primary goal here is rapid data transfer.

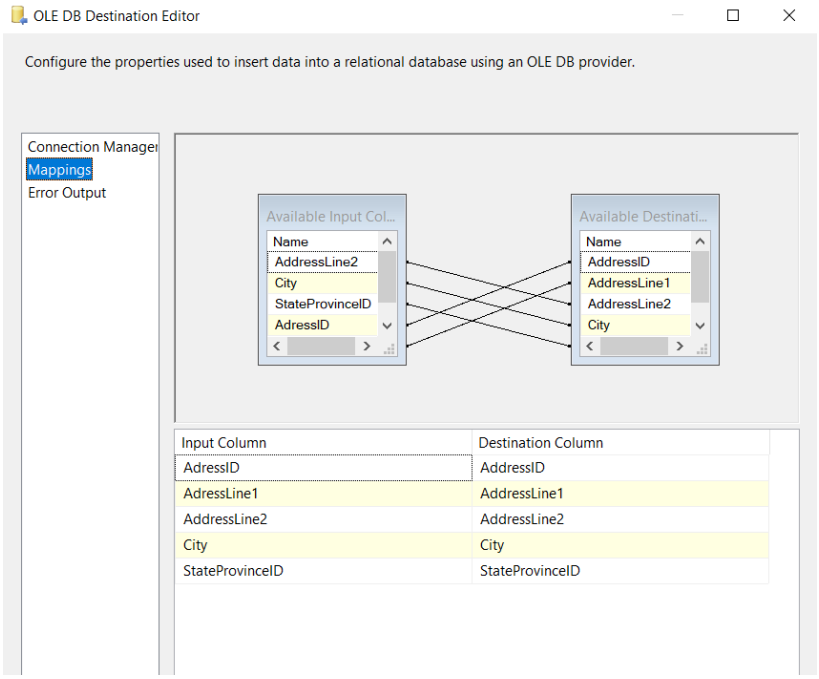
To achieve this, after creating a new project in SSIS, we set up an Execute SQL Task. This task is responsible for deleting all tables from the STAGE database, which prevents data from being appended to the tables repeatedly.

The data needed to populate the database was extracted from CSV files. To load our 11 tables, we created a Data Flow Task. Within this task, we individually configured a Flat File Source element for each CSV file to read the data. To avoid conversion issues, we set the locale to USA.



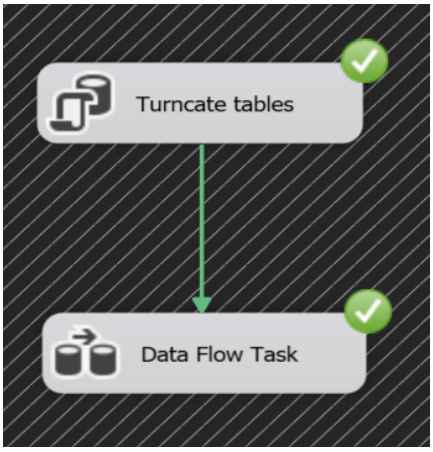
Next, we used an OLE DB Destination element to specify which table to load. Before loading, we checked the Mappings tab for any errors.



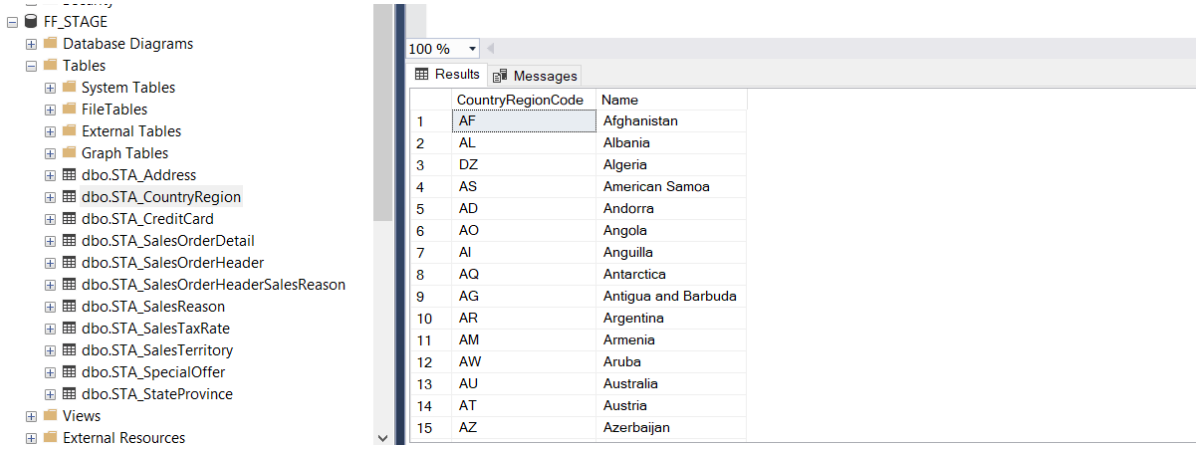


Here's what it looked like when executed on all 11 boards:

Now the entire package could be run:



In SQL Server Management Studio, we checked to see if the load was successful:



	CountryRegionCode	Name
1	AF	Afghanistan
2	AL	Albania
3	DZ	Algeria
4	AS	American Samoa
5	AD	Andorra
6	AO	Angola
7	AI	Anguilla
8	AQ	Antarctica
9	AG	Antigua and Barbuda
10	AR	Argentina
11	AM	Armenia
12	AW	Aruba
13	AU	Australia
14	AT	Austria
15	AZ	Azerbaijan

This confirmed the Extract process was complete.

We encountered no issues during the procedure; we successfully executed everything using the methods learned in class.

Transform

During the Transform process, we performed transformations on the tables in the STAGE layer. After these modifications, the data was loaded into the DW layer. This layer is expected to be in a complete and perfect state, as the subsequent process for the star schema will utilize these tables.

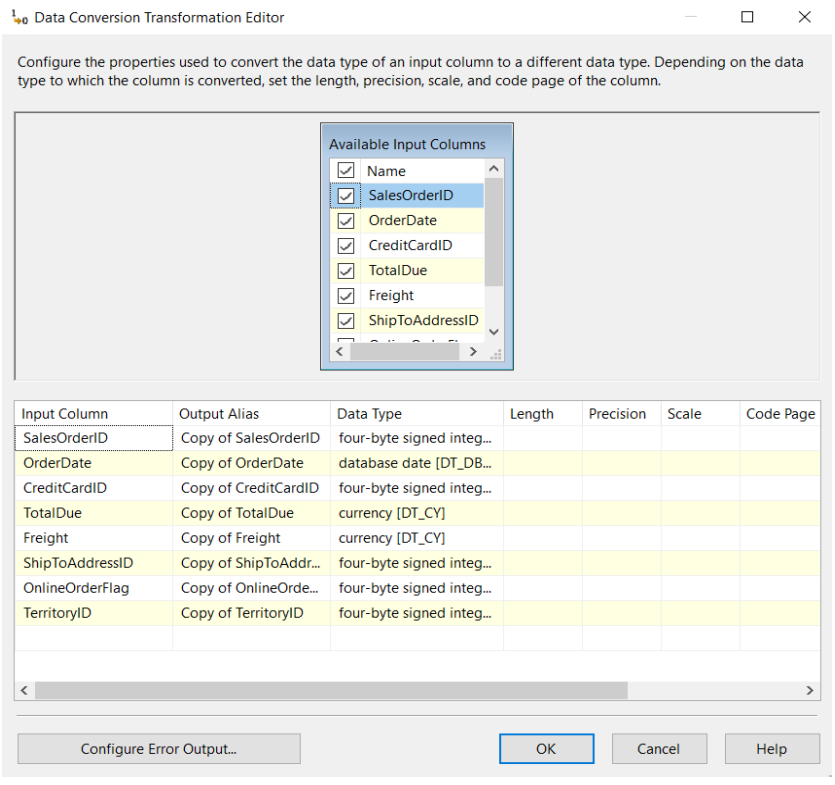
As a first step, we handled the NULL values, which were present as strings within the data. Using the method taught in class, we iterated through the tables with two nested loops. Wherever a "NULL" string was found, we replaced it with an actual NULL value using an UPDATE statement. We followed the provided supplementary material for these steps.

Null Conversion Successful:

	AddressID	AddressLine1	AddressLine2	City	StateProvinceID
1	11476	5957 Pacifica Ave.	NULL	East Brisbane	64
2	26126	5957 Pacifica Ave.	NULL	Walla Walla	79
3	13849	5958 Cross Road	NULL	Lake Oswego	58
4	13954	5958 Meadow Glen Way	NULL	Spokane	79
5	16856	5959 C Olivera Rd	NULL	London	14
6	26795	596 Marfargoa Drive	NULL	Imperial Beach	9
7	23662	596 Marfargoa Drive	NULL	Perth	66
8	26948	596 Springwood Way	NULL	National City	9

Afterward, the data was loaded into the DW layer. First, we created a Data Flow to perform this operation. We specified the source table (from STAGE) using an OLE DB Source and loaded the data into the DW layer using an OLE DB Destination.

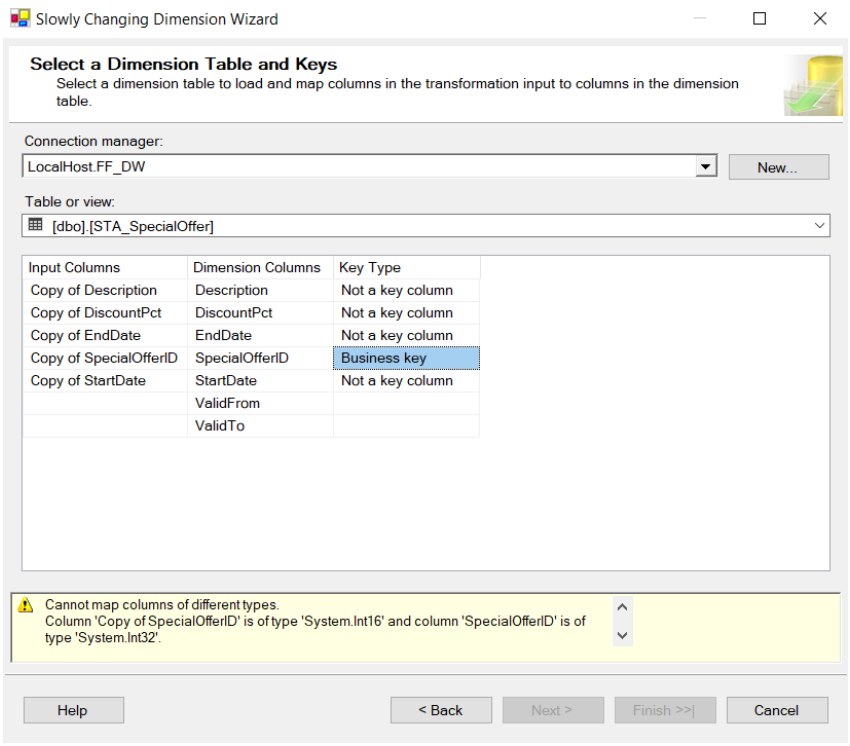
Between these two elements, the data was converted from VARCHAR to its actual data type using the Data Conversion element. We reviewed the Mappings file and adjusted the necessary data types:



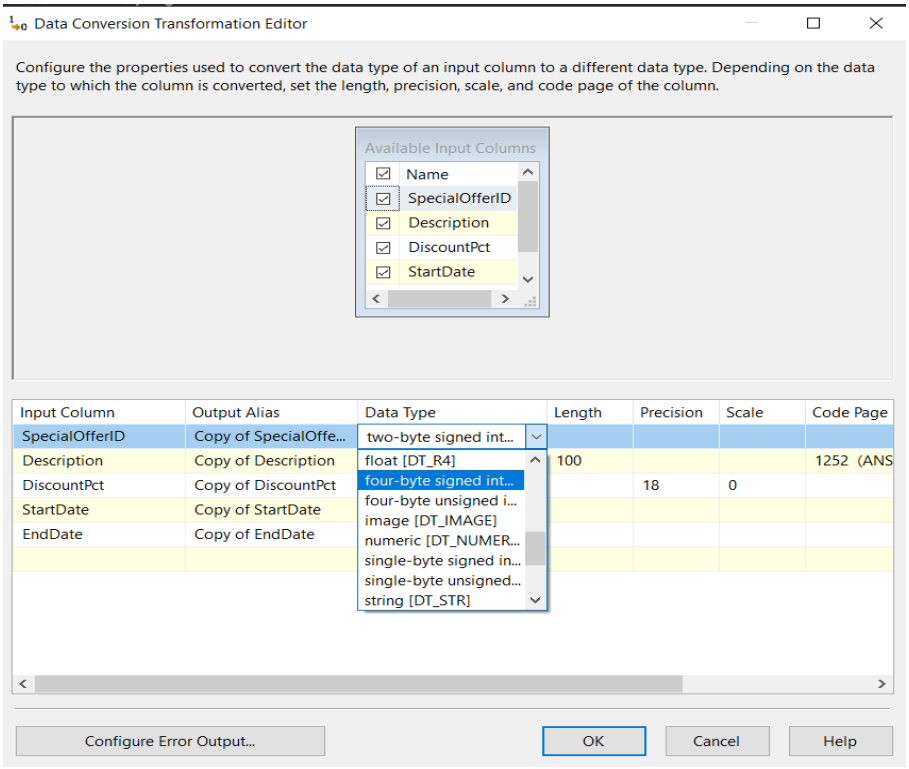
Next, using a Slowly Changing Dimension (SCD), we configured the Business Keys and defined both the fixed and historical data (in our case, there was no 'changing' type). Wherever the historical option was set, the ValidFrom and ValidTo date columns were updated. The ValidFrom column was populated, allowing us to easily track changes using these two columns.

We encountered a minor issue during this process. When we was adjusting the data types in the Data Conversion element according to the Mappings file, we set the INT types to "two-byte signed integer" as shown in class. At that point, it didn't flag this as an error. However, when we needed to specify the Business Key and configure the "Copy of" versions of the columns in the Slowly Changing Dimension, an error indicated that certain data types were INT16 instead of INT32 or INT64. We noted these down and went back a step to modify them in the Data Conversion. After resolving this error, the elements ran perfectly, and the correctly typed data was loaded into the DW layer tables.

The error message:

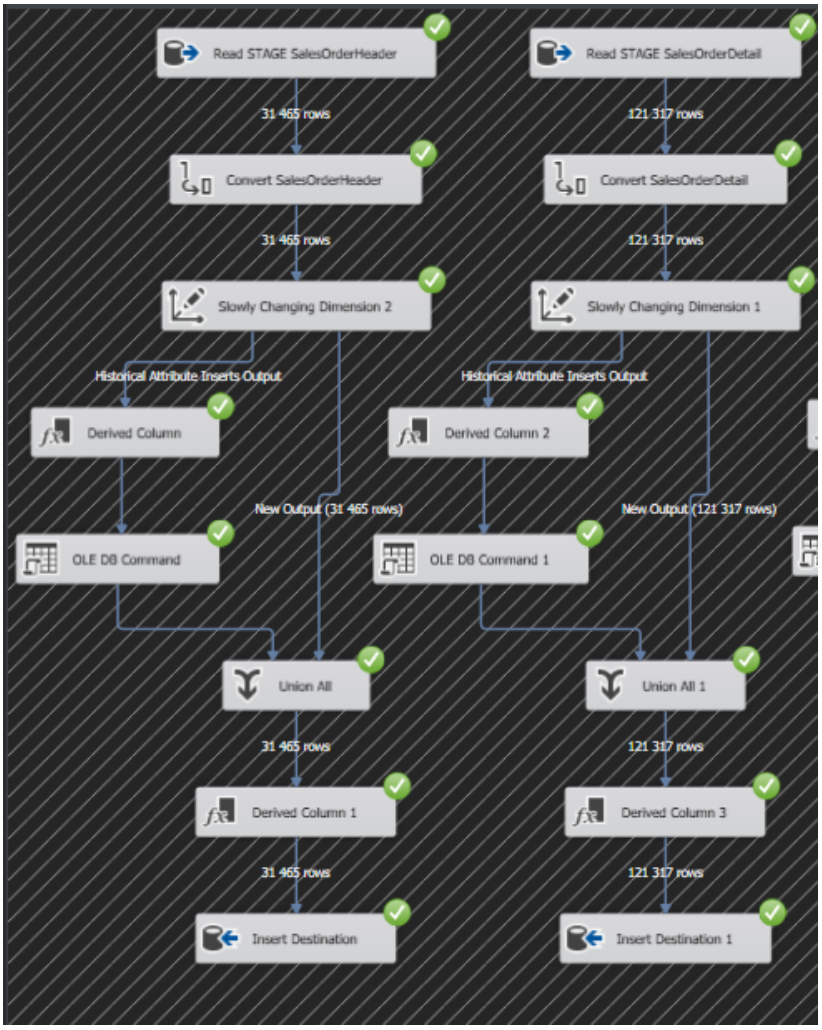


The solution:

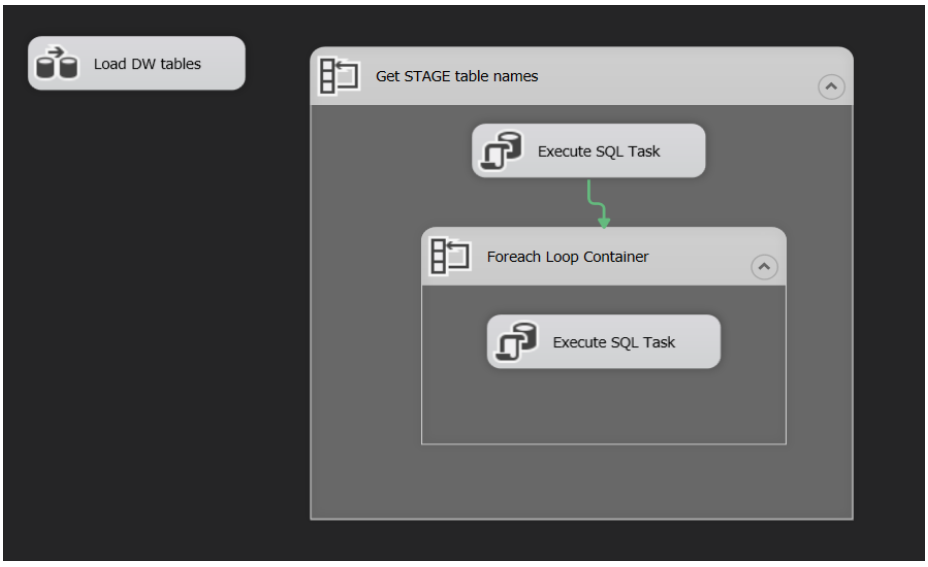


In the end, everything ran successfully:

Zoomed in on a small detail about loading two boards into a DW layer:



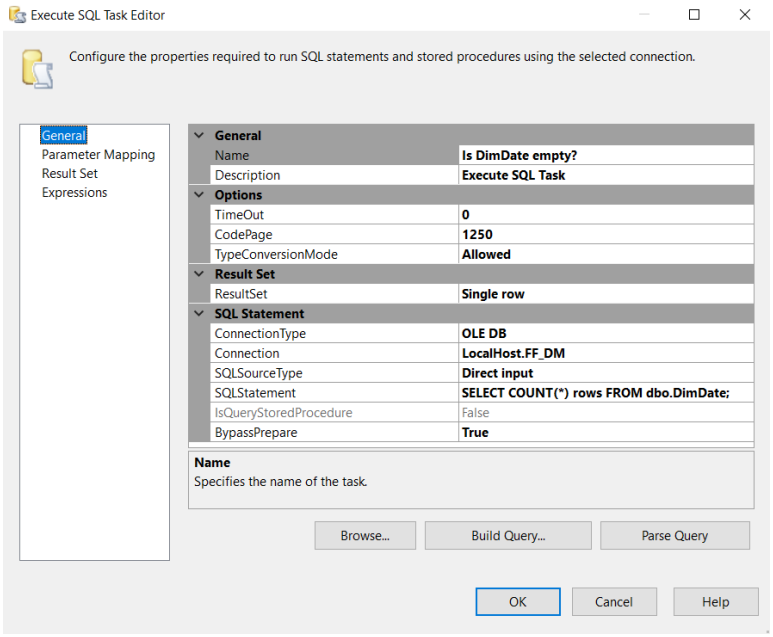
Finished status in Control Flow:



Load

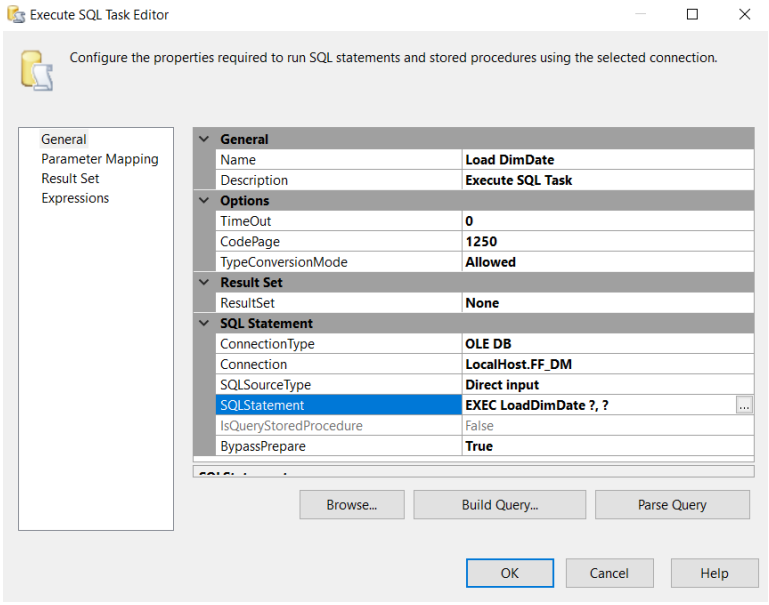
Creating the Load layer proved to be the most challenging and time-consuming part. In this process, we had to populate the dimension tables and the fact table according to the star schema, using pre-prepared views.

We started populating the dimension tables with the date dimension table, which is always a mandatory creation. Used the same template that was utilized in class for its creation. First, we created the initial Execute SQL Task to check if the DimDate table was empty.



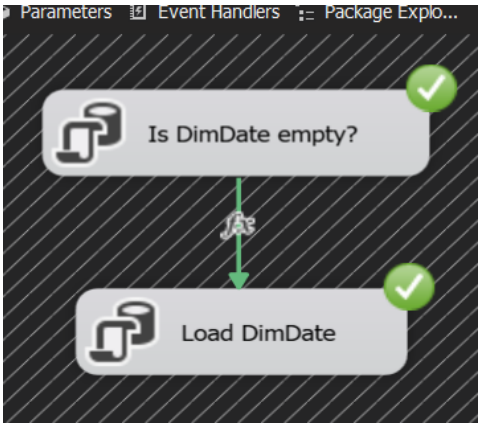
Next, we set the date to 01.01.2008 since this is the date of the first data in our tables.

Executed this SQL statement using Execute SQL Task:



Finally, by linking the two tasks, we specified the condition that it can populate the DimDate table only if it was previously empty.

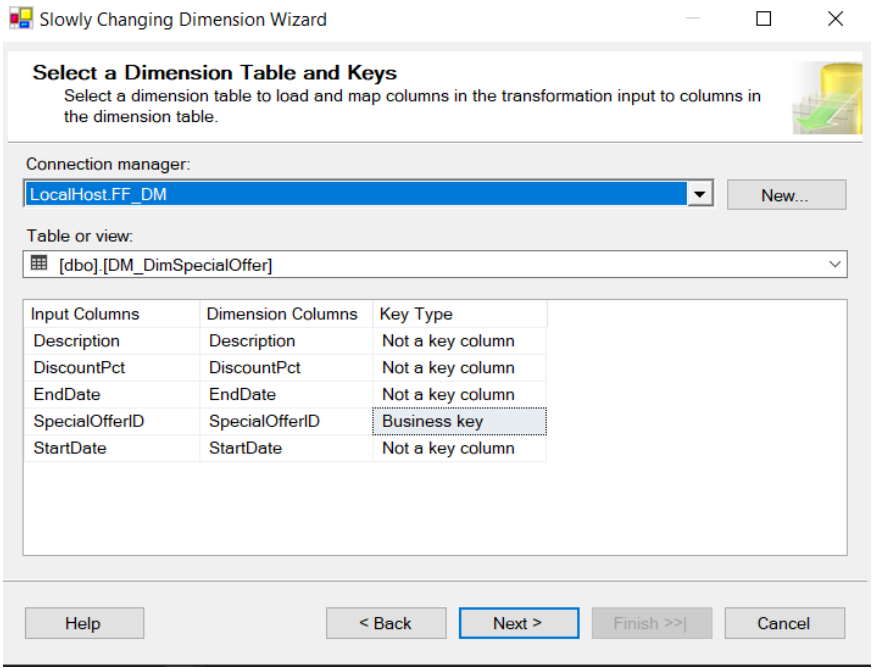
The result:



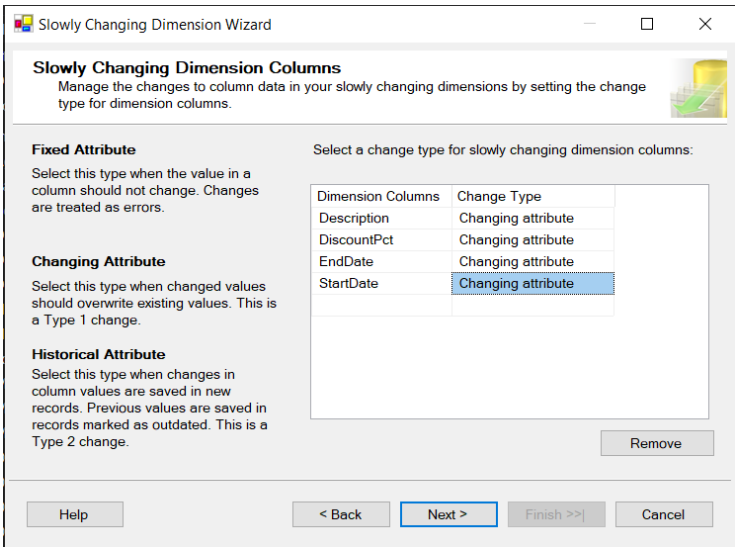
Afterward, we began populating the other dimension tables.

To do this, we created views for the dimension tables in Microsoft SQL Server Management Studio, utilizing tables from previous layers and joining the necessary ones. Within the Data Flow, these views were used in the OLE DB Sources corresponding to each dimension table. We used the Slowly Changing Dimension (SCD) element to set all table columns to "Changing Attribute". This ensures that if a row's values change, they are updated within the existing row rather than creating a new one.

Here's the SCD wizard for our SpecialOffer dimension table:



Changing attributes:



The other elements are generated by SCD (OLE DB Command, Insert).

Everything ran successfully:

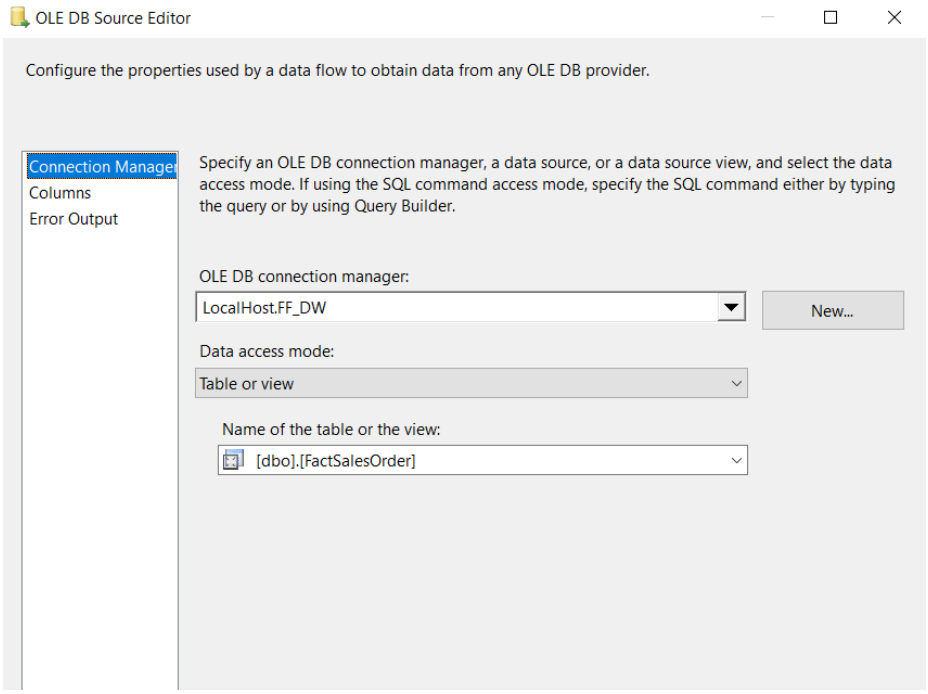
In Microsoft SQL Server Management Studio, we checked the uploaded dimension tables.

DimAddress table:

	DimAddressID	AddressID	AddressLine1	AddressLine2	City	StateProvinceName	TaxRate	TaxRateName	TerritoryName	CountryRegionName	LoadTimeStar
1	1	11476	5957 Pacifica Ave.	NULL	East Brisbane	Queensland	NULL	NULL	Australia	Australia	0x000000000
2	2	26126	5957 Pacifica Ave.	NULL	Walla Walla	Washington	8	Washington State Sales Tax	Northwest	United States	0x000000000
3	3	13849	5958 Cross Road	NULL	Lake Oswego	Oregon	NULL	NULL	Northwest	United States	0x000000000
4	4	13954	5958 Meadow Glen Way	NULL	Spokane	Washington	8	Washington State Sales Tax	Northwest	United States	0x000000000
5	5	16856	5959 C Olivera Rd	NULL	London	England	17	United Kingdom Output Tax	United Kingdom	United Kingdom	0x000000000
6	6	26795	596 Marfargoa Drive	NULL	Imperial Beach	California	8	California State Sales Tax	Southwest	United States	0x000000000
7	7	23662	596 Marfargoa Drive	NULL	Perth	South Australia	NULL	NULL	Australia	Australia	0x000000000
8	8	26948	596 Springwood Way	NULL	National City	California	8	California State Sales Tax	Southwest	United States	0x000000000
9	9	15560	5960 Cross Road	# 20	Newton	British Columbia	7	Canadian GST	Canada	Canada	0x000000000
10	10	23553	5960 Springfield Drive	NULL	Berkeley	California	8	California State Sales Tax	Southwest	United States	0x000000000
11	11	21077	5960 Springfield Drive	NULL	La Jolla	California	8	California State Sales Tax	Southwest	United States	0x000000000
12	12	20914	5961 Crowe Pl.	NULL	Lakewood	California	8	California State Sales Tax	Southwest	United States	0x000000000
13	13	14038	5963 Meadow Lane	NULL	Rockhampton	Queensland	NULL	NULL	Australia	Australia	0x000000000
14	14	21197	5964 Sepulveda Ct.	NULL	Darlinghurst	New South Wales	NULL	NULL	Australia	Australia	0x000000000

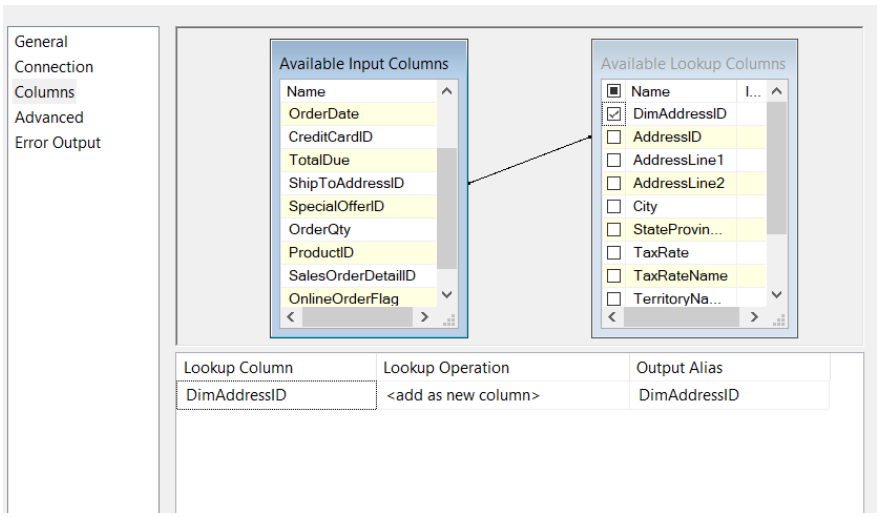
We continued with the fact table, and once completed, it was immediately ready for generating reports. We set up an auto-incrementing ID as the primary key, just as before. Additionally, we included the primary keys of the dimension tables and other fact columns.

The pre-created fact table view is read using an OLE DB Source once again::

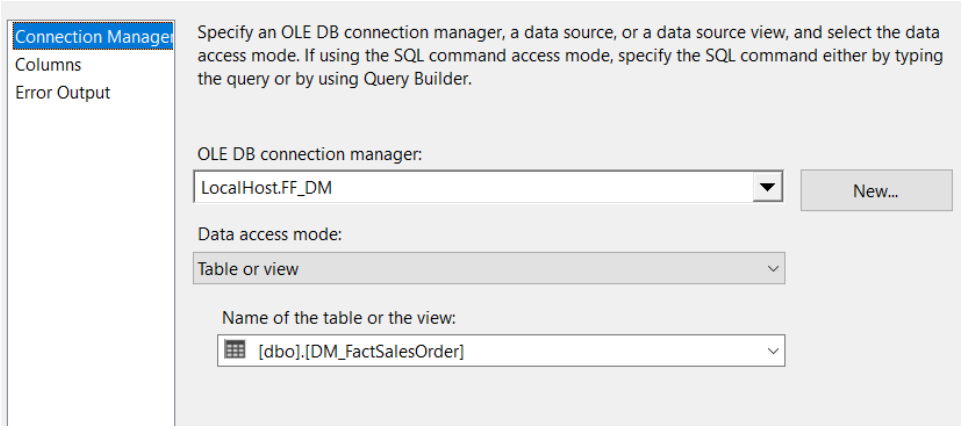


The primary keys of the dimension tables are retrieved using Lookup elements, which then link them to the corresponding row in the fact table and fetch the DimID for that row.

An example of this, is DimAddress's Lookup:

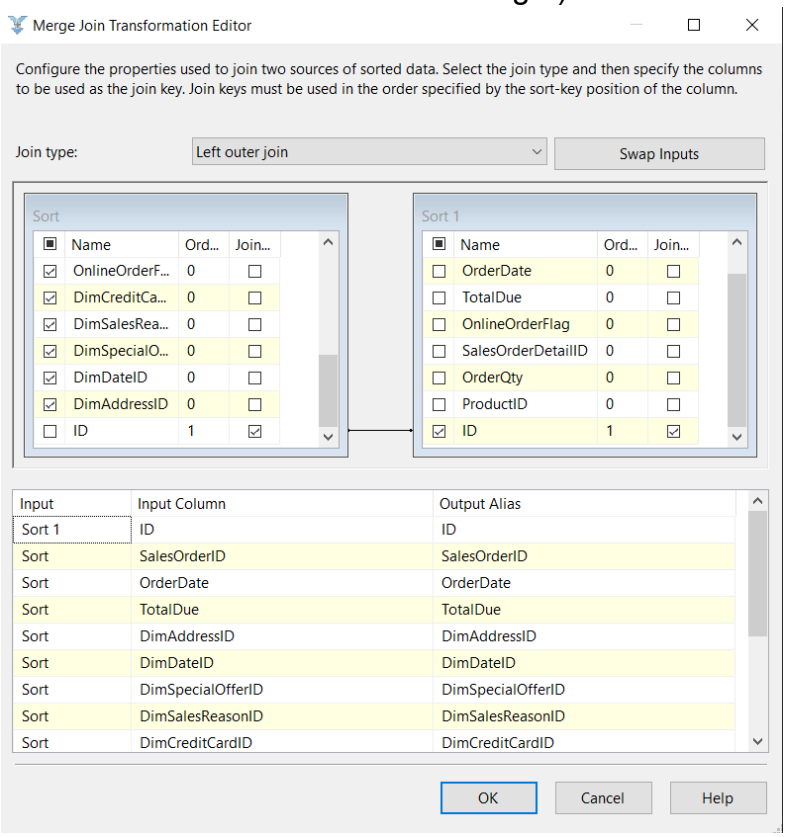


After performing all the Lookups for every dimension table, we started creating the other branch. This branch is responsible for examining the data in our actual fact table. If it finds new data, it loads it; if it finds modified data, it updates it. To do this, we needed to create another OLE DB Source, where we specified the DM layer's FactSalesOrder table as the source:

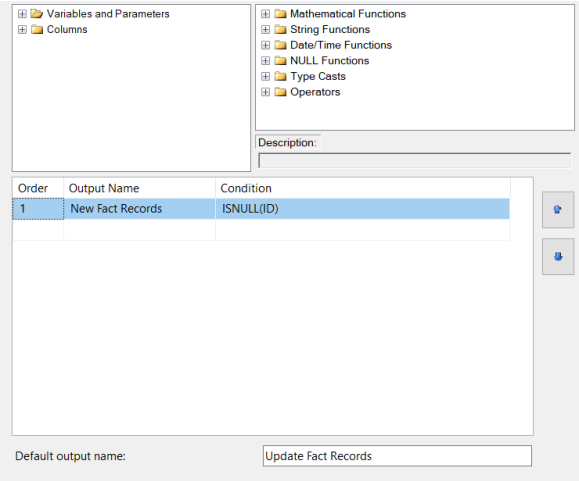


To enable these two branches to work together, we connected them using a Merge Join element with a Left outer join. Sorting is a prerequisite for the Merge Join, so we sorted both branches by ID using the Sort transformation.

Here's the content of the Merge Join element (the actual fact table is on the left, and the view-based fact table is on the right):

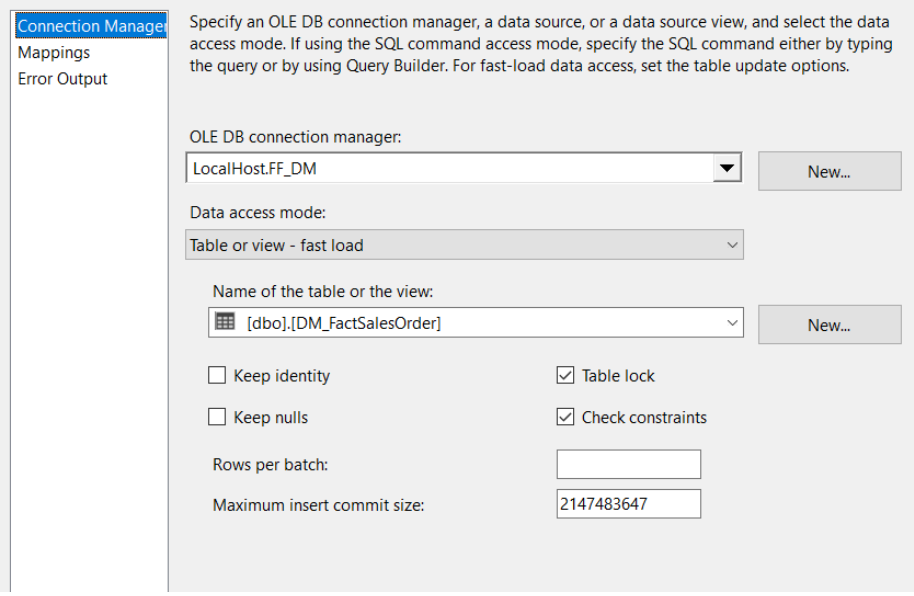


To ensure that even those elements of the view-based fact table that are not yet present in the actual fact table are loaded/modified (i.e., where the given ID is NULL in the actual fact table, it should be loaded, and where it is not NULL, it should be modified), we needed to create a Conditional Split transformation. Our condition is ISNULL(ID), as taught in class, which determines the direction of the branch.

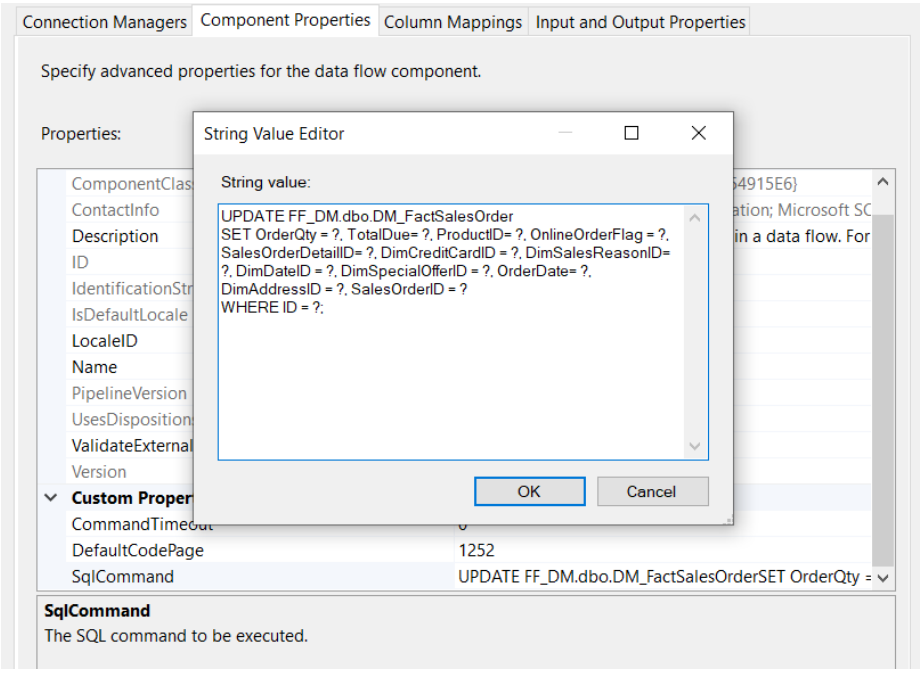


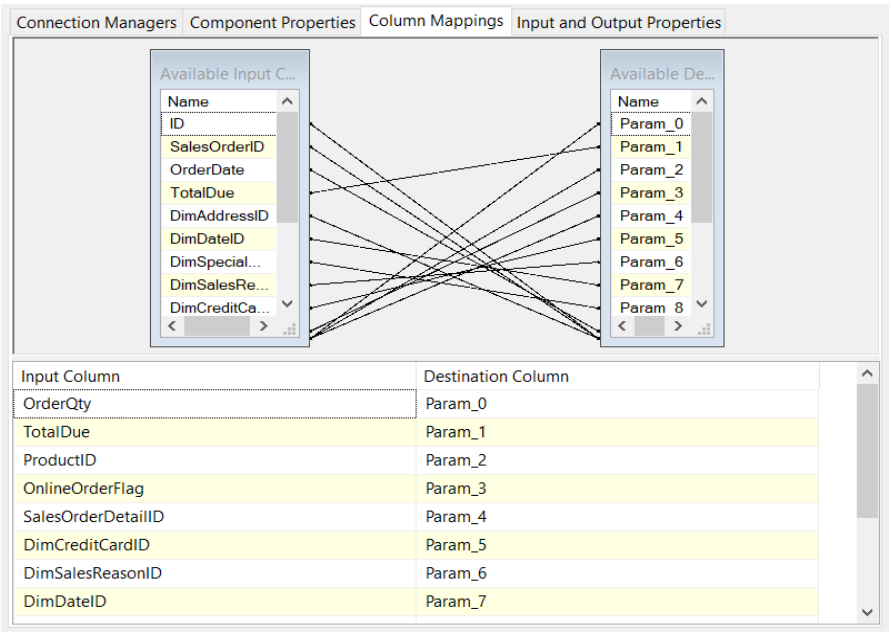
Depending on the direction, new data (OLE DB Destination) is loaded or modified (OLE DB Command).

OLE DB Destination:

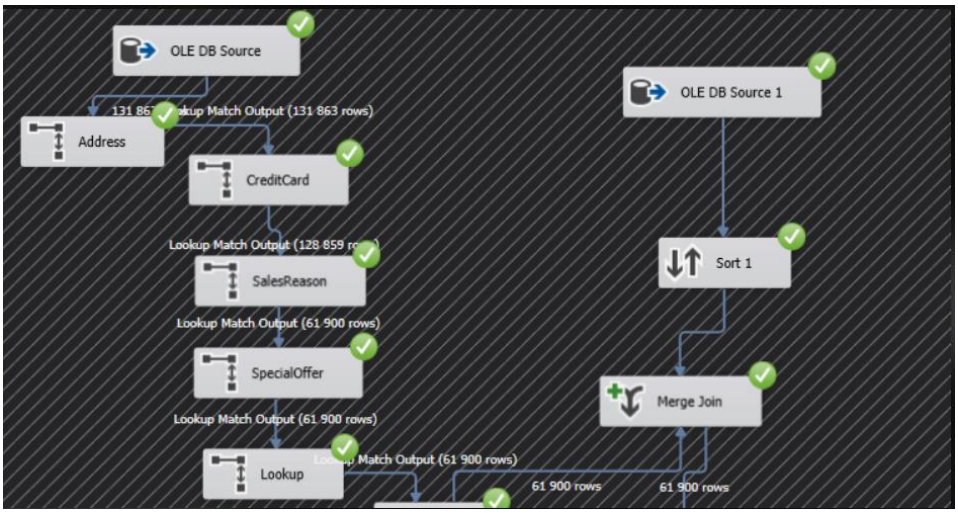


OLE DB Command with the appropriate UPDATE statement and parameterization:



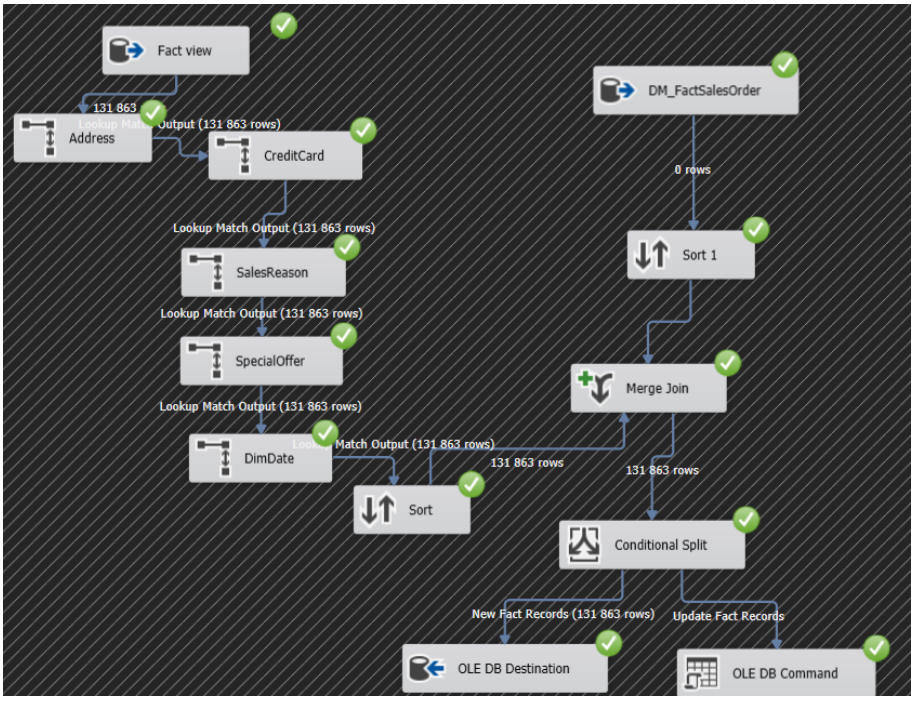


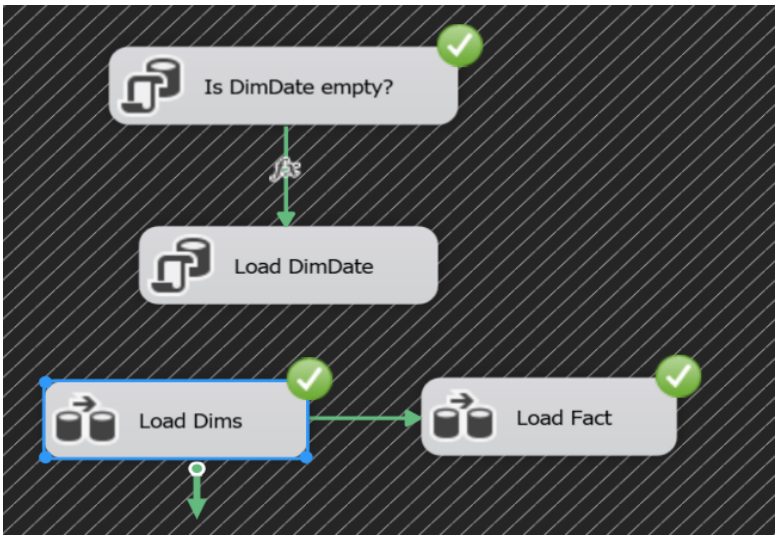
On the first run, we ran into this error:



At certain Lookups, the data volume decreased, and ultimately, only about half the data from the view-based version was loaded into the actual fact table. This resulted in the loss of a lot of important information. After many hours of debugging and re-attempting, we realized that we needed to change the "Redirect rows to no match output" setting on the first page of the Lookup element to "Ignore failure" to get all the data. This way, even if a match isn't found in the dimension table for a given fact entry, we still include it in the fact table. In such cases, since the data isn't present in the dimension table, we insert NULL values into the foreign key column.

Then we run:





Finally, the first 20 lines of the fact table:

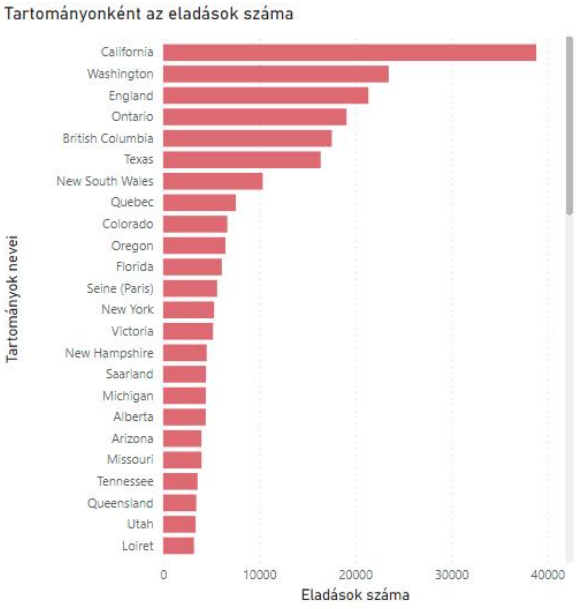
	ID	SalesOrderID	OrderDate	TotalDue	OnlineOrderFlag	SalesOrderDetailID	OrderQty	ProductID	DimDateID	DimSpecialOfferID	DimAddressID	DimCreditCardID	DimSalesReasonID	LoadTimeStamp
1	1	43659	2011-05-31	23153,2339	0	1	1	776	20110531	1	15656	5080	NULL	0x00000000001C314B
2	2	43659	2011-05-31	23153,2339	0	2	3	777	20110531	1	15656	5080	NULL	0x00000000001C314C
3	3	43659	2011-05-31	23153,2339	0	3	1	778	20110531	1	15656	5080	NULL	0x00000000001C314D
4	4	43659	2011-05-31	23153,2339	0	4	1	771	20110531	1	15656	5080	NULL	0x00000000001C314E
5	5	43659	2011-05-31	23153,2339	0	5	1	772	20110531	1	15656	5080	NULL	0x00000000001C314F
6	6	43659	2011-05-31	23153,2339	0	6	2	773	20110531	1	15656	5080	NULL	0x00000000001C3150
7	7	43659	2011-05-31	23153,2339	0	7	1	774	20110531	1	15656	5080	NULL	0x00000000001C3151
8	8	43659	2011-05-31	23153,2339	0	8	3	714	20110531	1	15656	5080	NULL	0x00000000001C3152
9	9	43659	2011-05-31	23153,2339	0	9	1	716	20110531	1	15656	5080	NULL	0x00000000001C3153
10	10	43659	2011-05-31	23153,2339	0	10	6	709	20110531	1	15656	5080	NULL	0x00000000001C3154
11	11	43659	2011-05-31	23153,2339	0	11	2	712	20110531	1	15656	5080	NULL	0x00000000001C3155
12	12	43659	2011-05-31	23153,2339	0	12	4	711	20110531	1	15656	5080	NULL	0x00000000001C3156
13	13	43660	2011-05-31	1457,3288	0	13	1	762	20110531	1	243	13535	NULL	0x00000000001C3157
14	14	43660	2011-05-31	1457,3288	0	14	1	758	20110531	1	243	13535	NULL	0x00000000001C3158
15	15	43661	2011-05-31	36865,8012	0	15	1	745	20110531	1	12356	9263	NULL	0x00000000001C3159
16	16	43661	2011-05-31	36865,8012	0	16	1	743	20110531	1	12356	9263	NULL	0x00000000001C315A
17	17	43661	2011-05-31	36865,8012	0	17	2	747	20110531	1	12356	9263	NULL	0x00000000001C315B
18	18	43661	2011-05-31	36865,8012	0	18	4	712	20110531	1	12356	9263	NULL	0x00000000001C315C
19	19	43661	2011-05-31	36865,8012	0	19	4	715	20110531	1	12356	9263	NULL	0x00000000001C315D
20	20	43661	2011-05-31	36865,8012	0	20	2	742	20110531	1	12356	9263	NULL	0x00000000001C315E

REPORTS

1) Number of sales per countries/states

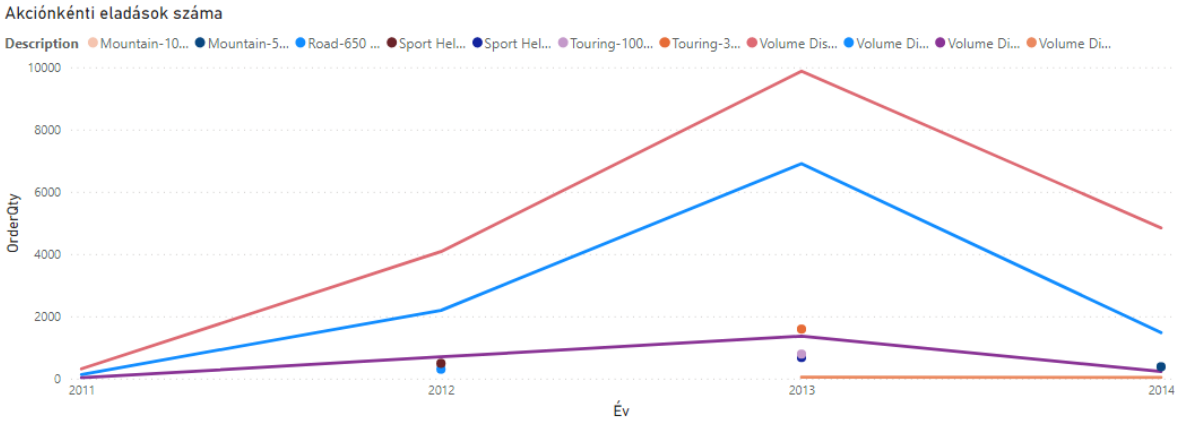
285460

OrderQty



Gain insights into sales distribution by region. See sales quantities and locations at a glance on a map and bar chart. For precise figures, simply filter the data down to individual regions.

2) Number of sales for promotions

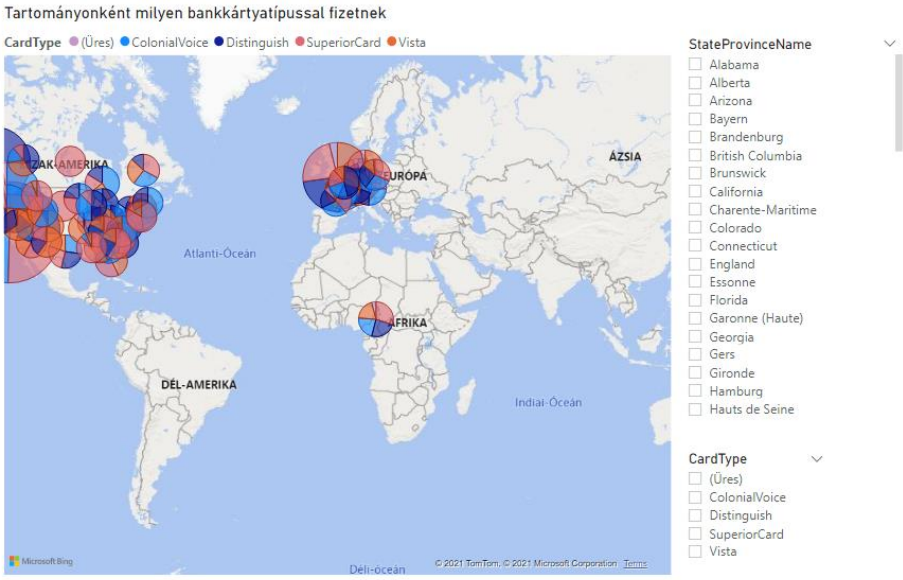


Description	OrderQty	ProductID
Mountain-100 Clearance Sale	456	707
Mountain-500 Silver Clearance Sale	382	708
Road-650 Overstock	304	709
Sport Helmet Discount-2002	492	711
Sport Helmet Discount-2003	680	712
Touring-1000 Promotion	788	714
Touring-3000 Promotion	1592	715
Volume Discount 11 to 14	19121	716
Volume Discount 15 to 24	10713	725
Volume Discount 25 to 40	2321	730
Volume Discount 41 to 60	85	738
Összesen	**36934**	748

Discover how specific promotions impact sales numbers. You can easily search for promotions or products using the filters to reveal precise sales figures.

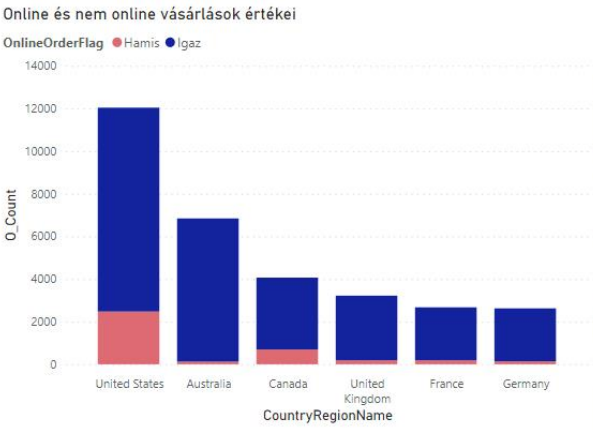
3) What type of credit card is paid by state/province

StateProvinceName	CardType	Count
Alabama	ColonialVoice	9
Alabama	SuperiorCard	5
Alabama	Vista	8
Alberta	ColonialVoice	15
Alberta	Distinguish	37
Alberta	SuperiorCard	10
Alberta	Vista	28
Arizona	ColonialVoice	4
Arizona	Distinguish	9
Arizona	SuperiorCard	39
Arizona	Vista	15
Bayern		12
Bayern	ColonialVoice	85
Bayern	Distinguish	83
Bayern	SuperiorCard	88
Bayern	Vista	76
Brandenburg		1
Brandenburg	ColonialVoice	14
Brandenburg	Distinguish	13
Brandenburg	SuperiorCard	16
Brandenburg	Vista	6
British Columbia		107
British Columbia	ColonialVoice	840
British Columbia	Distinguish	864
British Columbia	SuperiorCard	848
British Columbia	Vista	813
Összesen		**31465**



Discover which credit card types are favored in each province. The visualization allows you to filter by domain and card type, or to see the exact number of users paying with certain card types within each domain.

4) Number of online and in-store purchases



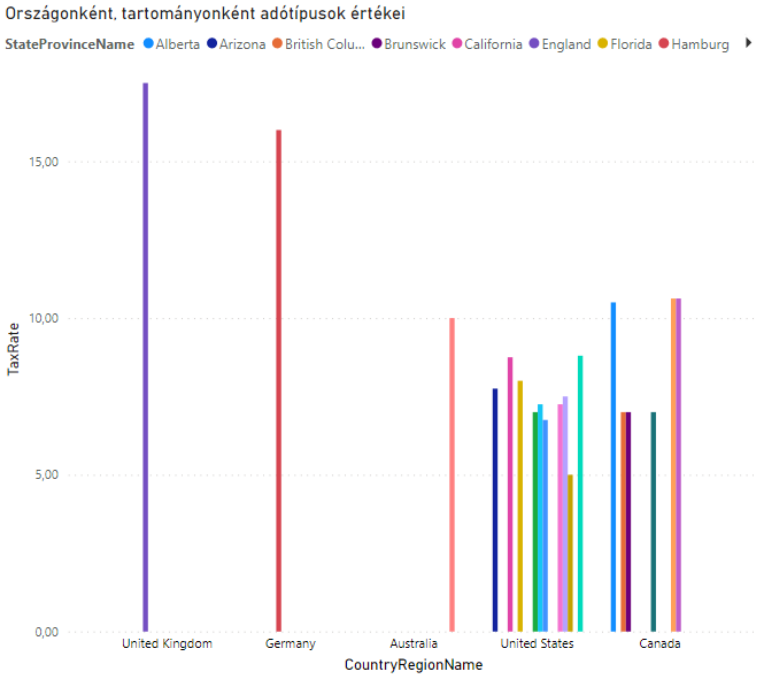
CountryRegionName	StateProvinceName	OnlineOrderFlag	O_Count
Australia	New South Wales	Hamis	86
Australia	New South Wales	Igaz	2923
Australia	Queensland	Hamis	4
Australia	Queensland	Igaz	1457
Australia	South Australia	Hamis	11
Australia	South Australia	Igaz	455
Australia	Tasmania	Igaz	197
Australia	Victoria	Hamis	24
Australia	Victoria	Igaz	1686
Canada	Alberta	Hamis	75
Összesen			**31465**

Discover the distribution of online versus in-store purchases across each country and its provinces. You can find the precise purchase counts in the provided table. (Online = True (Igaz), In-store = False (Hamis))

5) Tax types by country and province

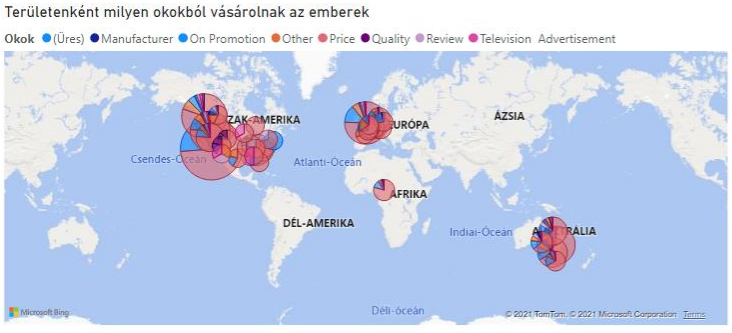
CountryRegionName	StateProvinceName	TaxRate
United Kingdom	England	17,50
Germany	Hamburg	16,00
Canada	Ontario	14,25
Canada	Quebec	14,25
Canada	Alberta	14,00
Australia	Victoria	10,00
United States	Washington	8,80
United States	California	8,75
United States	Florida	8,00
United States	Arizona	7,75
United States	Texas	7,50
United States	Michigan	7,25
United States	Tennessee	7,25
Canada	Alberta	7,00
Canada	British Columbia	7,00
Canada	Brunswick	7,00
Canada	Manitoba	7,00
Canada	Ontario	7,00
Canada	Quebec	7,00
United States	Massachusetts	7,00
United States	Minnesota	6,75
United States	Utah	5,00

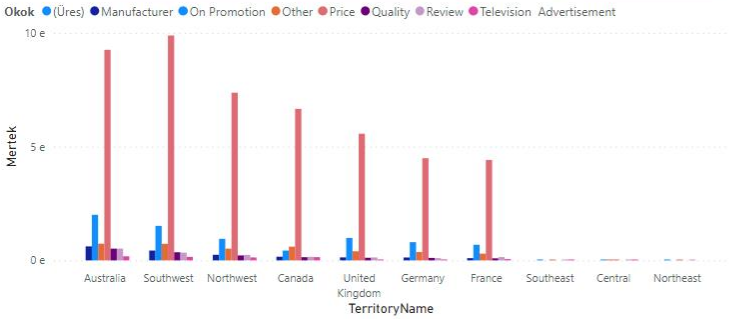
CountryRegionName
- Australia
- Canada
- France
- Germany
- United Kingdom
- United States



The visualization shows the tax rates by country or domain.

6) Reasons for purchases by area





StateProvinceName
Alabama
Alberta
Arizona
Bayern
Brandenburg
British Columbia
Brunswick
California
Charente-Maritime
Colorado
Connecticut
England
Essonne
Florida
Garonne (Haute)
Georgia
Gers
Gironde
Hamburg
Hauts de Seine
Hessen
Idaho
Illinois
Indiana
Kentucky
Loir et Cher
Loiret
Maine
Manitoba
Maryland
Massachusetts
Michigan

Name
Demo Event
Magazine Advertisement
Manufacturer
On Promotion
Other
Price
Quality
Review
Sponsorship
Television Advertisement

TerritoryName
Australia
Canada
Central
France
Germany
Northeast
Northwest
Southeast
Southwest
United Kingdom

64515
Mertek

This feature allows us to understand what drives purchases within specific areas and their respective provinces.

Source				Target			
Database Name	Table Name	Column Name	Data Type	Database Name	Table Name	Column name	Data Type
FF_STAGE	STA_SalesOrderHeaderSalesReason	SalesOrderID	VARCHAR(256)	FF_DW	DW_SalesOrderHeaderSalesReason	SalesOrderID	INT
FF_STAGE	STA_SalesOrderHeaderSalesReason	SalesReasonID	VARCHAR(256)	FF_DW	DW_SalesOrderHeaderSalesReason	SalesReasonID	INT
FF_STAGE	STA_SalesReason	SalesReasonID	VARCHAR(256)	FF_DW	DW_SalesReason	SalesReasonID	INT
FF_STAGE	STA_SalesReason	Name	VARCHAR(256)	FF_DW	DW_SalesReason	Name	VARCHAR(30)
FF_STAGE	STA_SalesOrderHeader	SalesOrderID	VARCHAR(256)	FF_DW	DW_SalesOrderHeader	SalesOrderID	INT
FF_STAGE	STA_SalesOrderHeader	OrderDate	VARCHAR(256)	FF_DW	DW_SalesOrderHeader	OrderDate	DATE
FF_STAGE	STA_SalesOrderHeader	CreditCardID	VARCHAR(256)	FF_DW	DW_SalesOrderHeader	CreditCardID	INT
FF_STAGE	STA_SalesOrderHeader	TotalDue	VARCHAR(256)	FF_DW	DW_SalesOrderHeader	TotalDue	MONEY
FF_STAGE	STA_SalesOrderHeader	Freight	VARCHAR(256)	FF_DW	DW_SalesOrderHeader	Freight	MONEY
FF_STAGE	STA_SalesOrderHeader	ShipToAddressID	VARCHAR(256)	FF_DW	DW_SalesOrderHeader	ShipToAddressID	INT
FF_STAGE	STA_SalesOrderHeader	OnlineOrderFlag	VARCHAR(256)	FF_DW	DW_SalesOrderHeader	OnlineOrderFlag	INT
FF_STAGE	STA_CreditCard	CreditCardID	VARCHAR(256)	FF_DW	DW_CreditCard	CreditCardID	INT
FF_STAGE	STA_CreditCard	CardType	VARCHAR(256)	FF_DW	DW_CreditCard	CardType	VARCHAR(20)
FF_STAGE	STA_CreditCard	ExpMonth	VARCHAR(256)	FF_DW	DW_CreditCard	ExpMonth	INT
FF_STAGE	STA_CreditCard	ExpYear	VARCHAR(256)	FF_DW	DW_CreditCard	ExpYear	INT
FF_STAGE	STA_SalesOrderDetail	SalesOrderID	VARCHAR(256)	FF_DW	DW_SalesOrderDetail	SalesOrderID	INT
FF_STAGE	STA_SalesOrderDetail	SalesOrderDetailID	VARCHAR(256)	FF_DW	DW_SalesOrderDetail	SalesOrderDetailID	INT
FF_STAGE	STA_SalesOrderDetail	OrderQty	VARCHAR(256)	FF_DW	DW_SalesOrderDetail	OrderQty	INT
FF_STAGE	STA_SalesOrderDetail	SpecialOfferID	VARCHAR(256)	FF_DW	DW_SalesOrderDetail	SpecialOfferID	INT
FF_STAGE	STA_SalesOrderDetail	ProductID	VARCHAR(256)	FF_DW	DW_SalesOrderDetail	ProductID	INT
FF_STAGE	STA_SalesOrderDetail	UnitPrice	VARCHAR(256)	FF_DW	DW_SalesOrderDetail	UnitPrice	MONEY
FF_STAGE	STA_SalesOrderDetail	LineTotal	VARCHAR(256)	FF_DW	DW_SalesOrderDetail	LineTotal	MONEY
FF_STAGE	STA_Address	AddressID	VARCHAR(256)	FF_DW	DW_Address	AddressID	INT
FF_STAGE	STA_Address	AddressLine1	VARCHAR(256)	FF_DW	DW_Address	AddressLine1	VARCHAR(256)
FF_STAGE	STA_Address	AddressLine2	VARCHAR(256)	FF_DW	DW_Address	AddressLine2	VARCHAR(256)
FF_STAGE	STA_Address	City	VARCHAR(256)	FF_DW	DW_Address	City	VARCHAR(50)
FF_STAGE	STA_Address	StateProvinceID	VARCHAR(256)	FF_DW	DW_Address	StateProvinceID	INT
FF_STAGE	STA_SpecialOffer	SpecialOfferID	VARCHAR(256)	FF_DW	DW_SpecialOffer	SpecialOfferID	INT
FF_STAGE	STA_SpecialOffer	Description	VARCHAR(256)	FF_DW	DW_SpecialOffer	Description	VARCHAR(100)
FF_STAGE	STA_SpecialOffer	DiscountPct	VARCHAR(256)	FF_DW	DW_SpecialOffer	DiscountPct	DECIMAL
FF_STAGE	STA_SpecialOffer	StartDate	VARCHAR(256)	FF_DW	DW_SpecialOffer	StartDate	DATE
FF_STAGE	STA_SpecialOffer	EndDate	VARCHAR(256)	FF_DW	DW_SpecialOffer	EndDate	DATE
FF_STAGE	STA_StateProvince	StateProvinceID	VARCHAR(256)	FF_DW	DW_StateProvince	StateProvinceID	INT
FF_STAGE	STA_StateProvince	CountryRegionCode	VARCHAR(256)	FF_DW	DW_StateProvince	CountryRegionCode	CHAR(2)
FF_STAGE	STA_StateProvince	Name	VARCHAR(256)	FF_DW	DW_StateProvince	Name	VARCHAR(100)
FF_STAGE	STA_StateProvince	TerritoryID	VARCHAR(256)	FF_DW	DW_StateProvince	TerritoryID	INT
FF_STAGE	STA_SalesTaxRate	TaxRate	VARCHAR(256)	FF_DW	DW_SalesTaxRate	TaxRate	DECIMAL
FF_STAGE	STA_SalesTaxRate	StateProvinceID	VARCHAR(256)	FF_DW	DW_SalesTaxRate	StateProvinceID	INT
FF_STAGE	STA_SalesTaxRate	Name	VARCHAR(256)	FF_DW	DW_SalesTaxRate	Name	VARCHAR(100)
FF_STAGE	STA_SalesTerritory	TerritoryID	VARCHAR(256)	FF_DW	DW_SalesTerritory	TerritoryID	INT
FF_STAGE	STA_SalesTerritory	Name	VARCHAR(256)	FF_DW	DW_SalesTerritory	Name	VARCHAR(30)
FF_STAGE	STA_CountryRegion	CountryRegionCode	VARCHAR(256)	FF_DW	DW_CountryRegion	CountryRegionCode	CHAR(2)
FF_STAGE	STA_CountryRegion	Name	VARCHAR(256)	FF_DW	DW_CountryRegion	Name	VARCHAR(100)

Source				Target			
Database Name	Table Name	Column Name	Data Type	Database Name	Table Name	Column name	Data Type
FF_DW	DW_SalesOrderHeaderSalesReason	SalesReasonID	INT	FF_DM	FactSalesOrder	DimSalesReasonID	INT
FF_DW	DW_SalesReason	SalesReasonID	INT	FF_DM	DimSalesReason	DimSalesReasonID	INT
FF_DW	DW_SalesReason	Name	VARCHAR(30)	FF_DM	DimSalesReason	SalesReasonName	VARCHAR(30)
FF_DW	DW_SalesOrderHeader	SalesOrderID	INT	FF_DM	FactSalesOrder	SalesOrderID	INT
FF_DW	DW_SalesOrderHeader	OrderDate	DATE	FF_DM	FactSalesOrder	OrderDate	DATE
FF_DW	DW_SalesOrderHeader	CreditCardID	INT	FF_DM	FactSalesOrder	DimCreditCardID	INT
FF_DW	DW_SalesOrderHeader	TotalDue	MONEY	FF_DM	FactSalesOrder	TotalDue	MONEY
FF_DW	DW_SalesOrderHeader	ShipToAddressID	INT	FF_DM	FactSalesOrder	DimAddressID	INT
FF_DW	DW_SalesOrderHeader	OnlineOrderFlag	INT	FF_DM	FactSalesOrder	OnlineOrderFlag	INT
FF_DW	DW_CreditCard	CreditCardID	INT	FF_DM	DimCreditCard	DimCreditCardID	INT
FF_DW	DW_CreditCard	CardType	VARCHAR(20)	FF_DM	DimCreditCard	CardType	VARCHAR(20)
FF_DW	DW_CreditCard	ExpMonth	INT	FF_DM	DimCreditCard	ExpMonth	INT
FF_DW	DW_CreditCard	ExpYear	INT	FF_DM	DimCreditCard	ExpYear	INT
FF_DW	DW_SalesOrderDetail	SalesOrderDetailID	INT	FF_DM	FactSalesOrder	SalesOrderDetailID	INT
FF_DW	DW_SalesOrderDetail	OrderQty	INT	FF_DM	FactSalesOrder	OrderQty	INT
FF_DW	DW_SalesOrderDetail	SpecialOfferID	INT	FF_DM	FactSalesOrder	DimSpecialOfferID	INT
FF_DW	DW_SalesOrderDetail	ProductID	INT	FF_DM	FactSalesOrder	ProductID	INT
FF_DW	DW_Address	AddressID	INT	FF_DM	DimAddress	DimAddressID	INT
FF_DW	DW_Address	AddressLine1	VARCHAR(256)	FF_DM	DimAddress	AddressLine1	VARCHAR(256)
FF_DW	DW_Address	AddressLine2	VARCHAR(256)	FF_DM	DimAddress	AddressLine2	VARCHAR(256)
FF_DW	DW_Address	City	VARCHAR(50)	FF_DM	DimAddress	City	VARCHAR(50)
FF_DW	DW_SpecialOffer	SpecialOfferID	INT	FF_DM	DimSpecialOffer	DimSpecialOfferID	INT
FF_DW	DW_SpecialOffer	Description	VARCHAR(100)	FF_DM	DimSpecialOffer	Description	VARCHAR(100)
FF_DW	DW_SpecialOffer	DiscountPct	DECIMAL	FF_DM	DimSpecialOffer	DiscountPct	DECIMAL
FF_DW	DW_SpecialOffer	StartDate	DATE	FF_DM	DimSpecialOffer	StartDate	DATE
FF_DW	DW_SpecialOffer	EndDate	DATE	FF_DM	DimSpecialOffer	EndDate	DATE
FF_DW	DW_StateProvince	Name	VARCHAR(100)	FF_DM	DimAddress	StateProvinceName	VARCHAR(100)
FF_DW	DW_SalesTaxRate	TaxRate	DECIMAL	FF_DM	DimAddress	TaxRate	DECIMAL
FF_DW	DW_SalesTaxRate	Name	VARCHAR(100)	FF_DM	DimAddress	TaxRateName	VARCHAR(100)
FF_DW	DW_SalesTerritory	Name	VARCHAR(30)	FF_DM	DimAddress	TerritoryName	VARCHAR(30)
FF_DW	DW_CountryRegion	Name	VARCHAR(100)	FF_DM	DimAddress	CountryRegionName	VARCHAR(100)