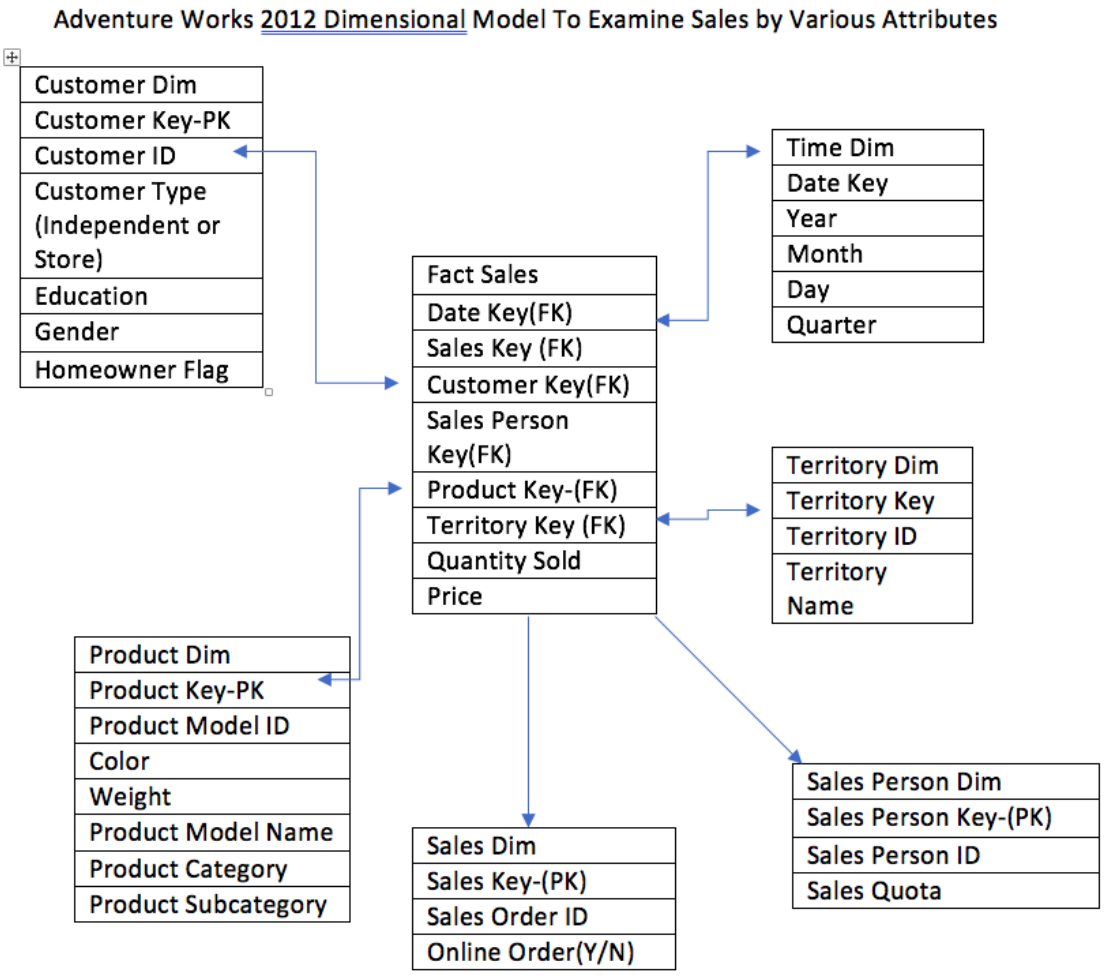
IS 549-Data Warehousing Final Project

Author: Jonathan Phelan

Winter Quarter 2018

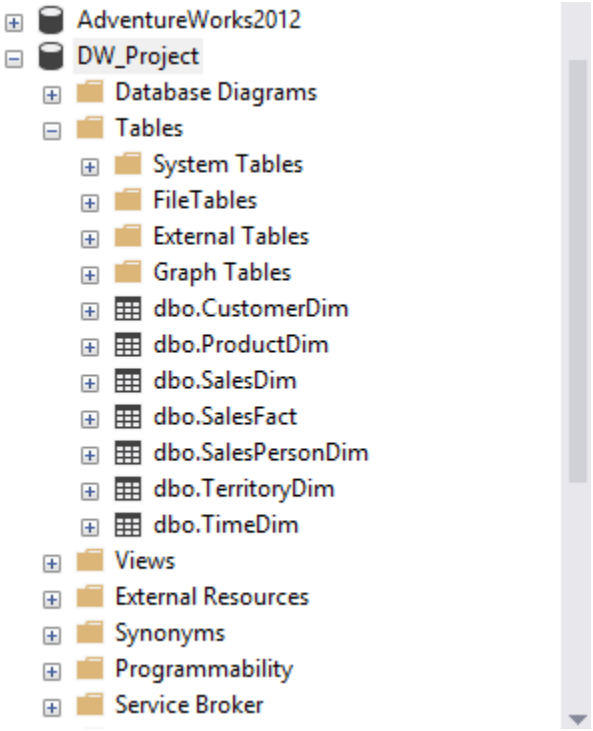
Summary: *For my project, I chose to utilize the fictitious Adventure Works 2012 OLTP database. The focus of my project was on sales as I wanted to create a data warehouse whereby I could look at sales by various attributes. For instance, what types of customers are buying Product X? Where are sales strongest/weakest? These are the types of questions I wanted to gain a better understanding of and why I chose to look at sales. These are questions that I felt most business would want to know the answers to.*

Creating the Dimensional Model



The above image is my final dimensional model for this project. There are 6 dimensions including the time dimension. Each dimension was considered to give me the information about a sale that was important for making strategic business decisions about who was buying certain products.

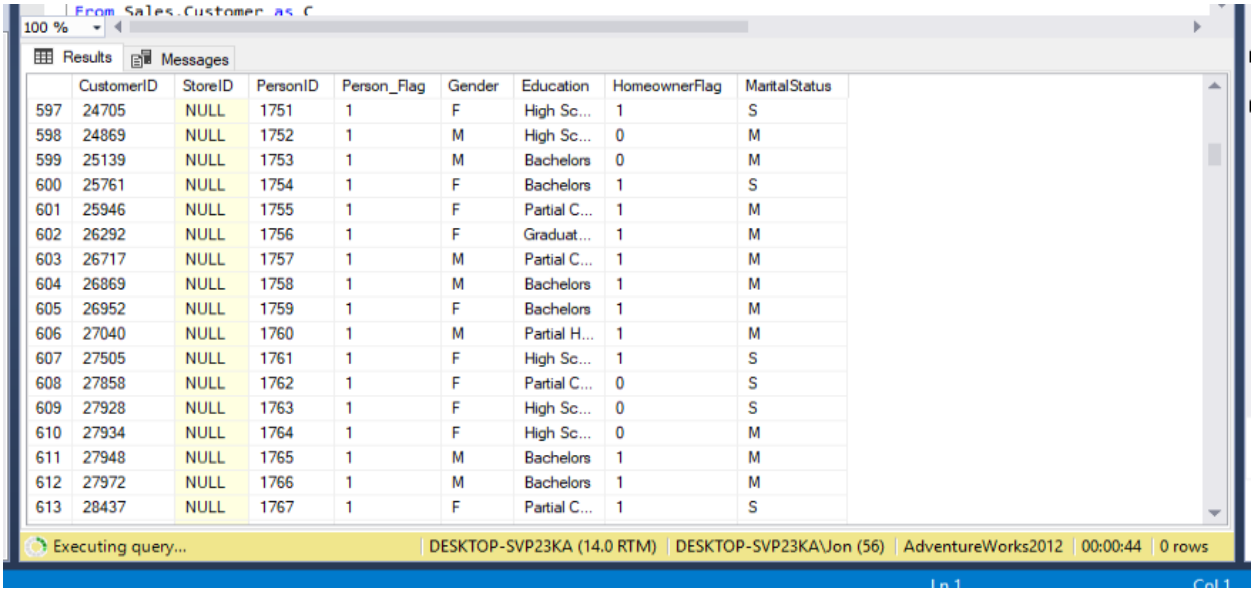
Examining database tables (dimensions)



Here, we have our dimension tables created in the target database. Here, the target is just called 'DW_Project' but you could also create a new instance. We can also see that we have our sales fact table as well. For each of these dimensions, views were created in SQL. The views extracted the data using SQL commands. Using SSIS, the data within those views was loaded into our dimension tables. An example of the customer view can be seen below and the resulting table is also shown.

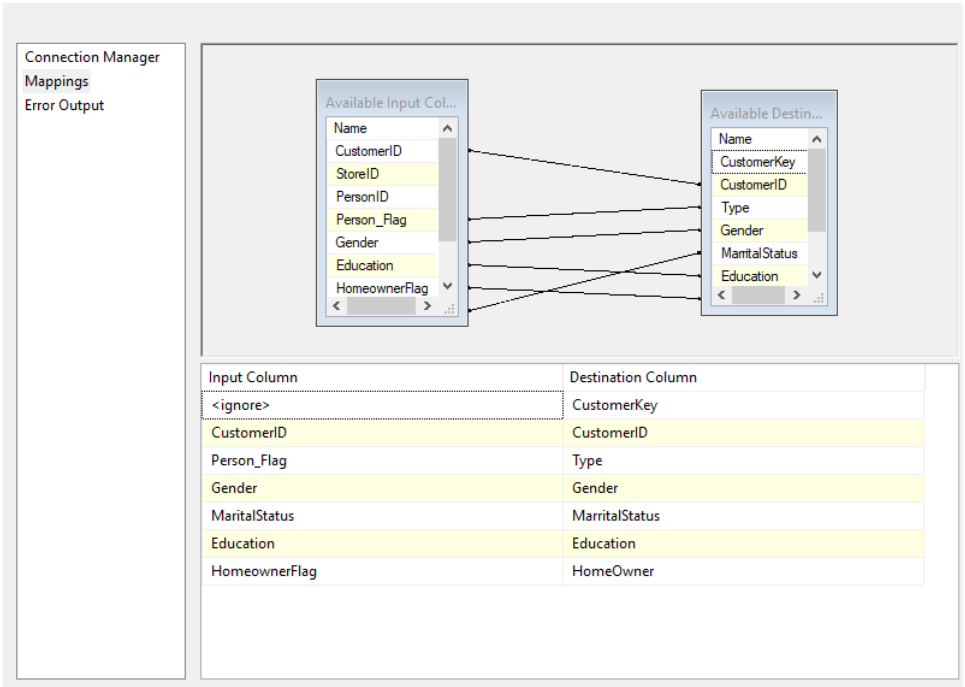
```
CREATE VIEW [CustomerDim] AS
Select CustomerID, StoreID, PersonID, Person_Flag, D.Gender, D.Education, D.HomeownerFlag, D.MaritalStatus
From Sales.Customer as C
FULL OUTER JOIN Person.Person AS P on C.PersonID=P.BusinessEntityID
FULL OUTER JOIN Sales.vPersonDemographics AS D on D.BusinessEntityID=P.BusinessEntityID
WHERE CustomerID IS NOT NULL
```

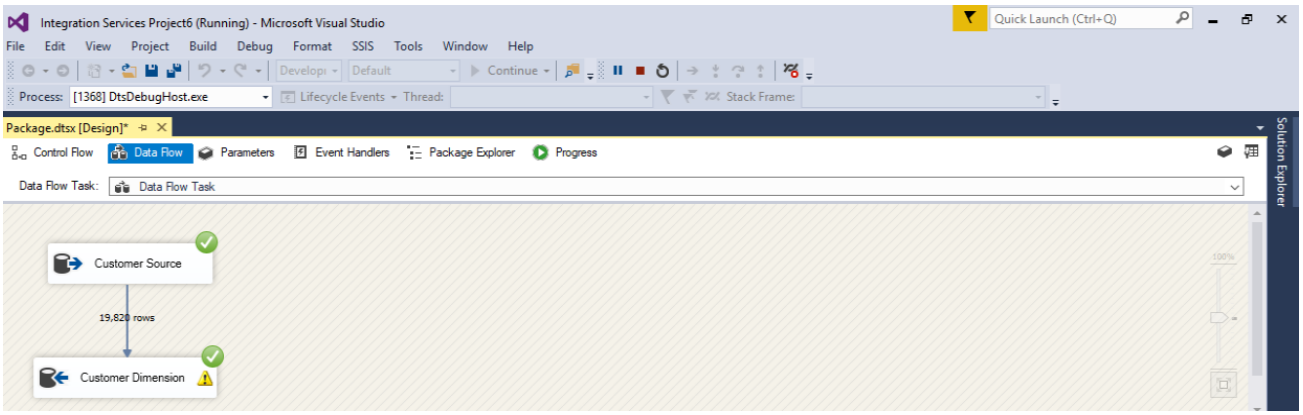
And the resulting table looks like this:



Views were creates for each dimension and then imported into SSIS to be loaded into the appropriate dimensions. This process can be seen via screenshots below.

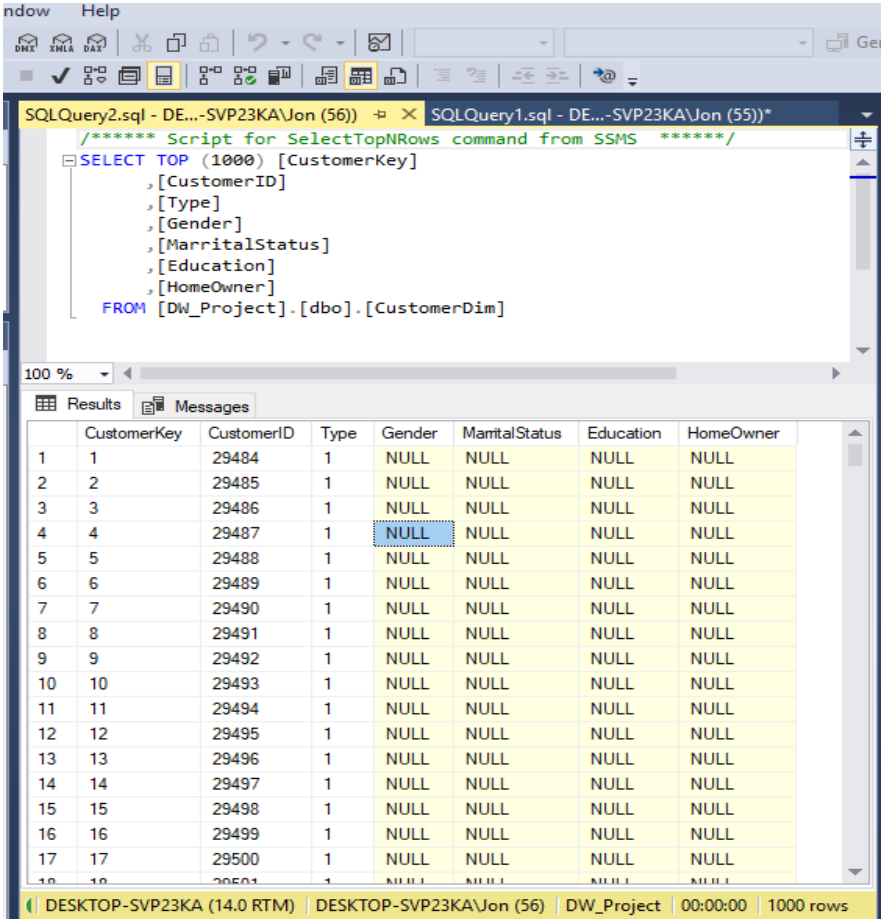
Mappings of Customer Dimension Example





-The yellow triangle appeared just to warn about labels for the 'Education' level column.

Result of Loading the Customer data into Customer Dimension



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [CustomerKey]
      ,[CustomerID]
      ,[Type]
      ,[Gender]
      ,[MarritalStatus]
      ,[Education]
      ,[HomeOwner]
  FROM [DW_Project].[dbo].[CustomerDim]
```

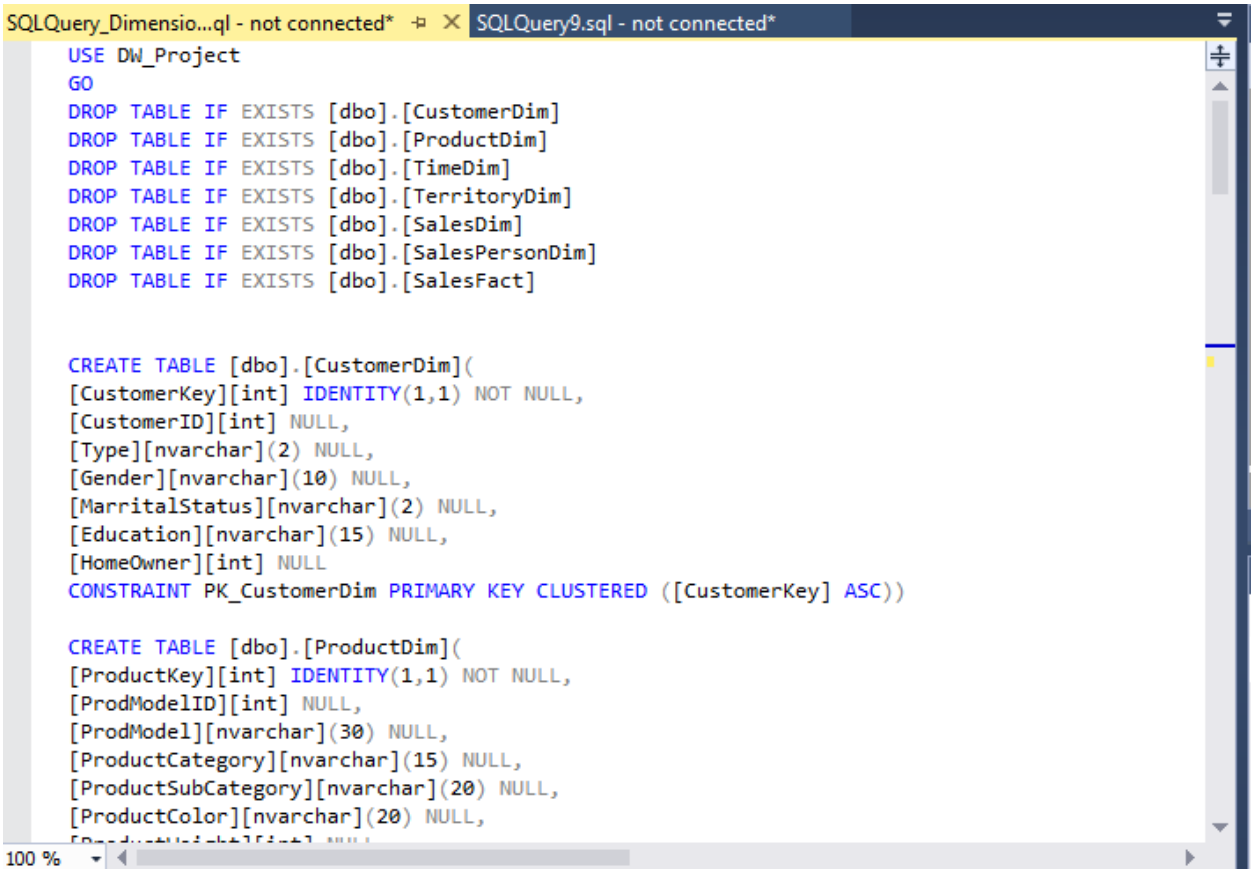
Results | Messages

	CustomerKey	CustomerID	Type	Gender	MarritalStatus	Education	HomeOwner
1	1	29484	1	NULL	NULL	NULL	NULL
2	2	29485	1	NULL	NULL	NULL	NULL
3	3	29486	1	NULL	NULL	NULL	NULL
4	4	29487	1	NULL	NULL	NULL	NULL
5	5	29488	1	NULL	NULL	NULL	NULL
6	6	29489	1	NULL	NULL	NULL	NULL
7	7	29490	1	NULL	NULL	NULL	NULL
8	8	29491	1	NULL	NULL	NULL	NULL
9	9	29492	1	NULL	NULL	NULL	NULL
10	10	29493	1	NULL	NULL	NULL	NULL
11	11	29494	1	NULL	NULL	NULL	NULL
12	12	29495	1	NULL	NULL	NULL	NULL
13	13	29496	1	NULL	NULL	NULL	NULL
14	14	29497	1	NULL	NULL	NULL	NULL
15	15	29498	1	NULL	NULL	NULL	NULL
16	16	29499	1	NULL	NULL	NULL	NULL
17	17	29500	1	NULL	NULL	NULL	NULL

DESKTOP-SVP23KA (14.0 RTM) | DESKTOP-SVP23KA\Jon (56) | DW_Project | 00:00:00 | 1000 rows

Here, we see many NULL values because those are stores, not individuals, thus columns like Gender, Marrital Status, and Education do not apply to those specific rows. We know this because our 'Type' indicator is 1, indicating that row corresponds to a store, not an individual.

Below is the code used to create my dimensions. While this was not required, I thought it would be important to see how I created the tables and how they were populated.



SQLQuery_Dimensio...ql - not connected* | SQLQuery9.sql - not connected*

```
CREATE TABLE [dbo].[ProductDim](
[ProductKey][int] IDENTITY(1,1) NOT NULL,
[ProdModelID][int] NULL,
[ProdModel][nvarchar](30) NULL,
[ProductCategory][nvarchar](15) NULL,
[ProductSubCategory][nvarchar](20) NULL,
[ProductColor][nvarchar](20) NULL,
[ProductWeight][int] NULL,
CONSTRAINT PK_ProductDim PRIMARY KEY CLUSTERED ([ProductKey] ASC))

CREATE TABLE [dbo].[SalesPersonDim] (
[SalesPersonKey][int] IDENTITY(1,1) NOT NULL,
[SalesPersonID][int] NULL,
[SalesQuota][int] NULL,
CONSTRAINT PK_SalesPersonDim PRIMARY KEY CLUSTERED ([SalesPersonKey] ASC))


CREATE TABLE [dbo].[SalesDim](
[SalesKey][int] IDENTITY(1,1) NOT NULL,
[SalesOrderID][int] NULL,
[Online][nvarchar](5) NULL,
CONSTRAINT PK_SalesDim PRIMARY KEY CLUSTERED ([SalesKey] ASC))

CREATE TABLE [dbo].[TerritoryDim](
[TerritoryKey][int] IDENTITY(1,1) NOT NULL,
[TerritoryID][int] NULL,
[TerritoryName][nvarchar](30) NULL,
CONSTRAINT PK_TerritoryDim PRIMARY KEY CLUSTERED ([TerritoryKey] ASC))
```

SQLQuery_Dimensio...ql - not connected* | SQLQuery9.sql - not connected*

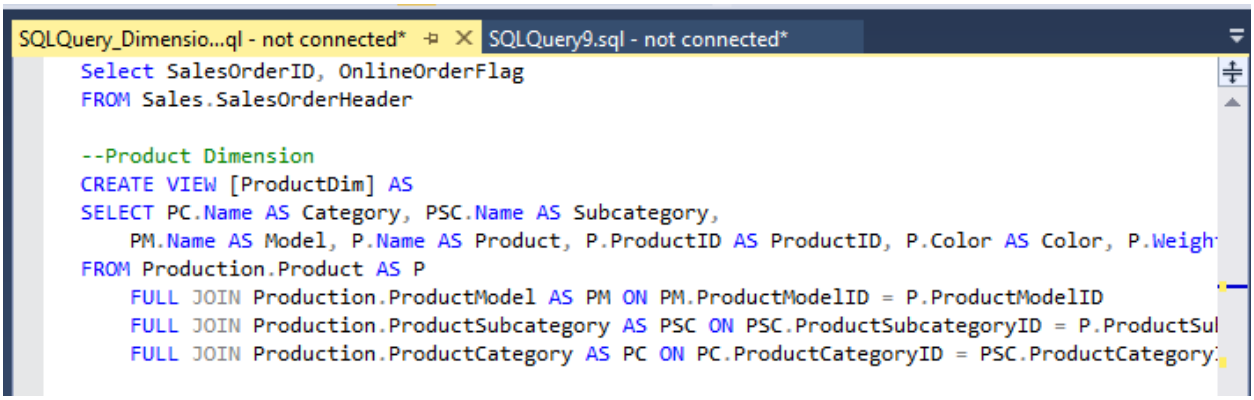
```
CONSTRAINT PK_SalesDim PRIMARY KEY CLUSTERED ([SalesKey] ASC))

CREATE TABLE [dbo].[TerritoryDim](
[TerritoryKey][int] IDENTITY(1,1) NOT NULL,
[TerritoryID][int] NULL,
[TerritoryName][nvarchar](30) NULL,
CONSTRAINT PK_TerritoryDim PRIMARY KEY CLUSTERED ([TerritoryKey] ASC))

/* Create Time Dimension */
CREATE TABLE [dbo].[TimeDim] (
[DateID] int NOT NULL IDENTITY(1,1)
, [Date] date NULL
, [Year] int NOT NULL
, [Month] int NOT NULL
, [Day] int NOT NULL
, [Quarter] int NOT NULL
, CONSTRAINT PK_Dates PRIMARY KEY CLUSTERED (DateID)
)


CREATE TABLE [dbo].[SalesFact] (
[DateKey] [int] NULL,
[SalesKey] [int] NULL,
[TerritoryKey] [int] NULL,
[SalesPersonKey] [int] NULL,
[ProductKey] [int] NULL,
[CustomerKey] [int] NULL,
```

```
)


CREATE TABLE [dbo].[SalesFact] (
[DateKey] [int] NULL,
[SalesKey] [int] NULL,
[TerritoryKey] [int] NULL,
[SalesPersonKey] [int] NULL,
[ProductKey] [int] NULL,
[CustomerKey] [int] NULL,
[QuantitySold] [int] NULL,
[SoldPrice] [decimal] NULL,
) ON [PRIMARY]
```

Below are the views I created to get the data for each dimension

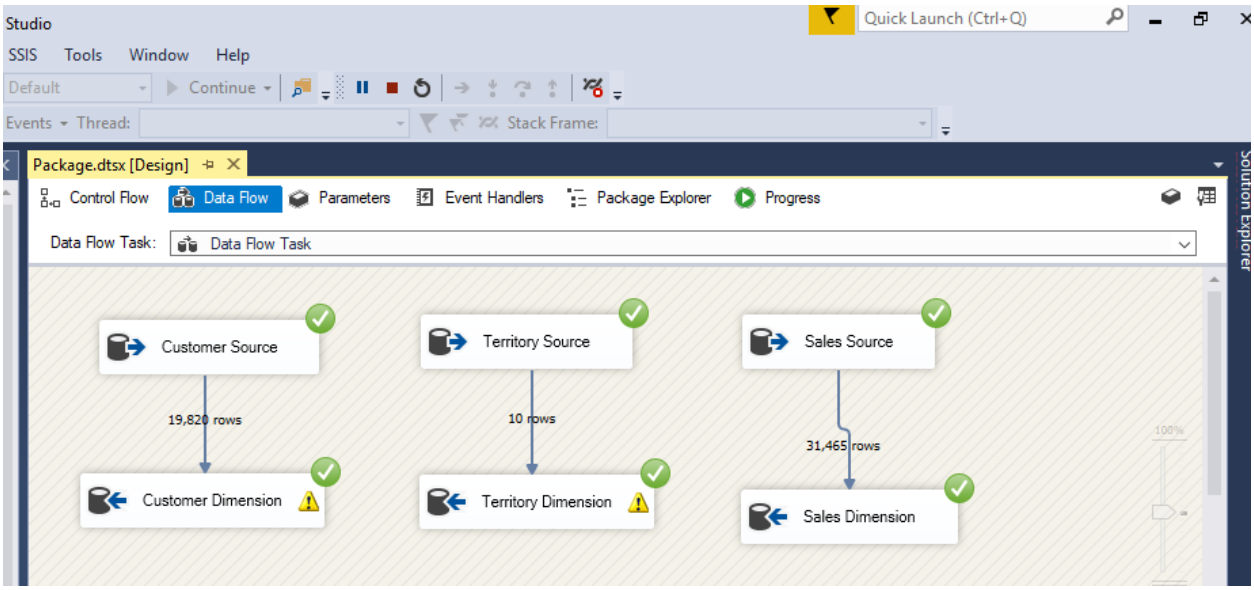
SQLQuery_Dimensio...ql - not connected* | SQLQuery9.sql - not connected*

```
CREATE VIEW [CustomerDim] AS
Select CustomerID, StoreID, PersonID, Person_Flag, D.Gender, D.Education, D.HomeownerFlag, D
From Sales.Customer as C
FULL OUTER JOIN Person.Person AS P on C.PersonID=P.BusinessEntityID
FULL OUTER JOIN Sales.vPersonDemographics AS D on D.BusinessEntityID=P.BusinessEntityID
WHERE CustomerID IS NOT NULL


--Territory Dimension
CREATE VIEW [TerritoryDim] AS
Select t.TerritoryID, t.Name
FROM Sales.SalesTerritory as t

--Sales Person Dim
CREATE VIEW [SalesPersonDim] AS
Select DISTINCT SalesPersonID, SalesQuota
FROM Sales.SalesOrderHeader As SH
INNER JOIN Sales.SalesPerson AS SP on SH.SalesPersonID=SP.BusinessEntityID
WHERE SalesQuota IS NOT NULL

--Sales Dimension
CREATE VIEW [SalesDim] AS
Select SalesOrderID, OnlineOrderFlag
FROM Sales.SalesOrderHeader
```

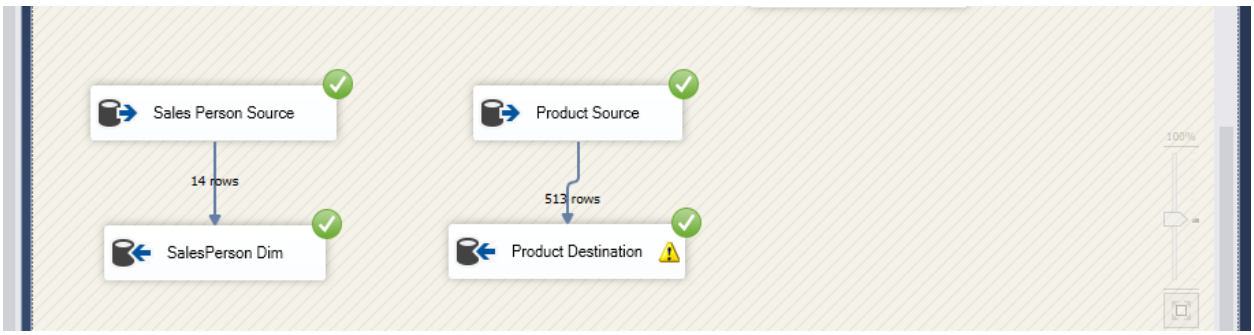


The time dimension was populated with a SQL script.

Entire Date Flow Example for populating dimensions

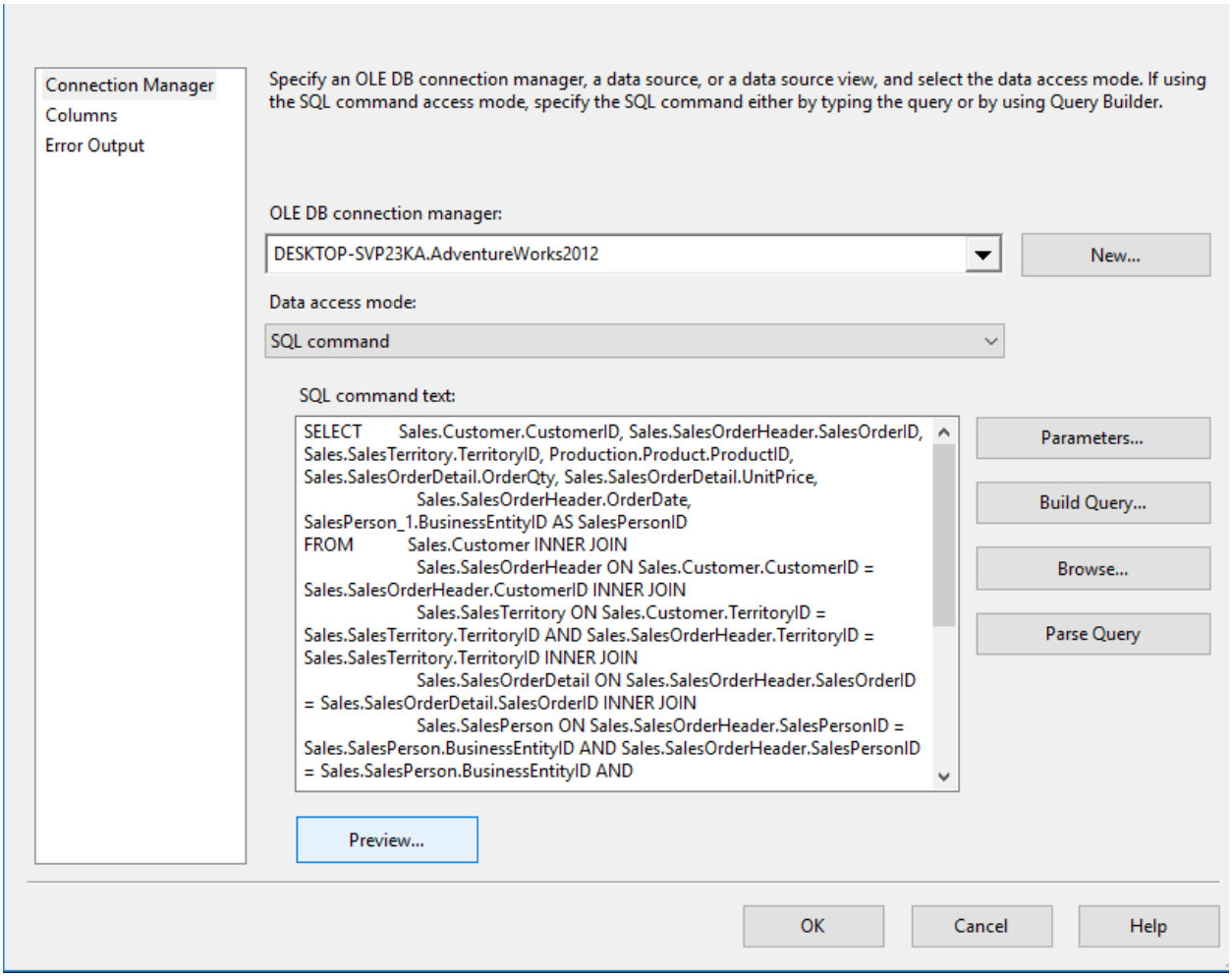


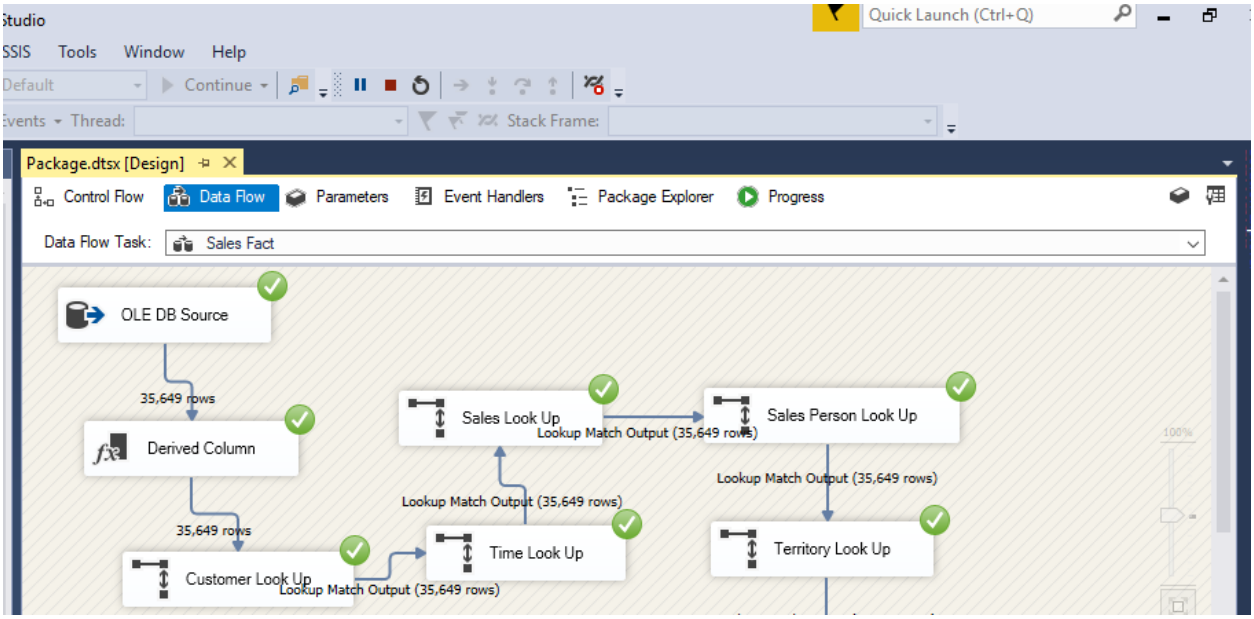
Here, again, the little yellow triangles were just warnings about labels possibly being truncated.

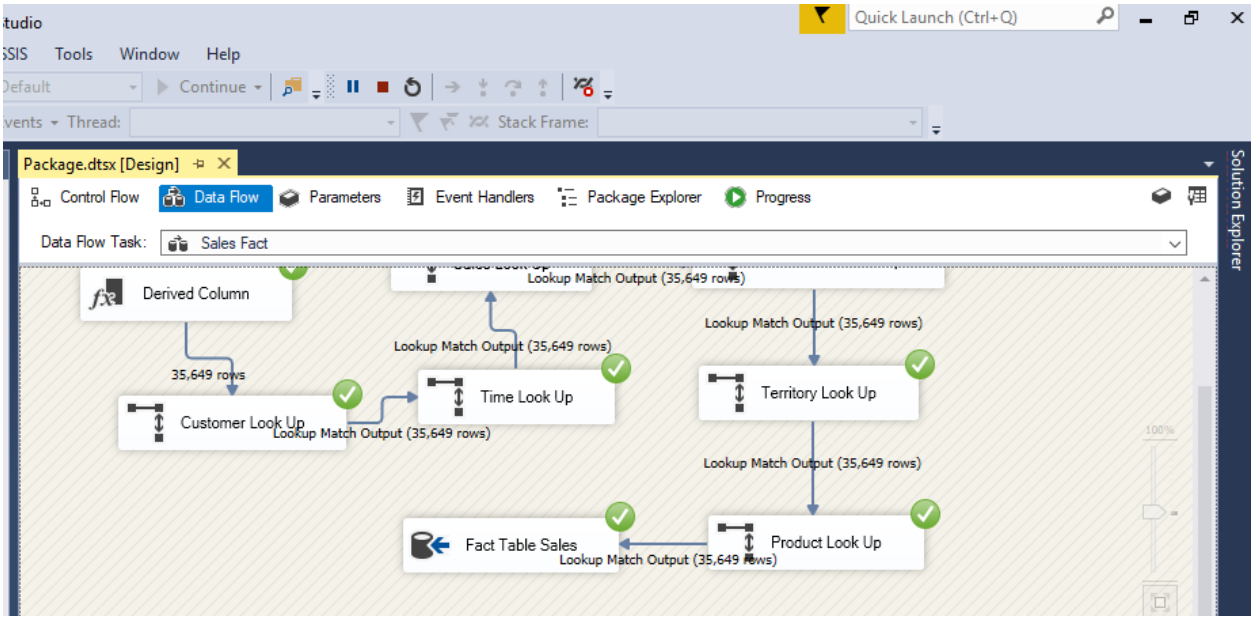


Now that we have our dimensions properly loaded, we need to prepare the fact table so it can be used for queries to answer the business questions we want to answer. To do this, I used lookups in SSIS to populate the fact table. The steps of this process are outlined via screenshots and short explanations.

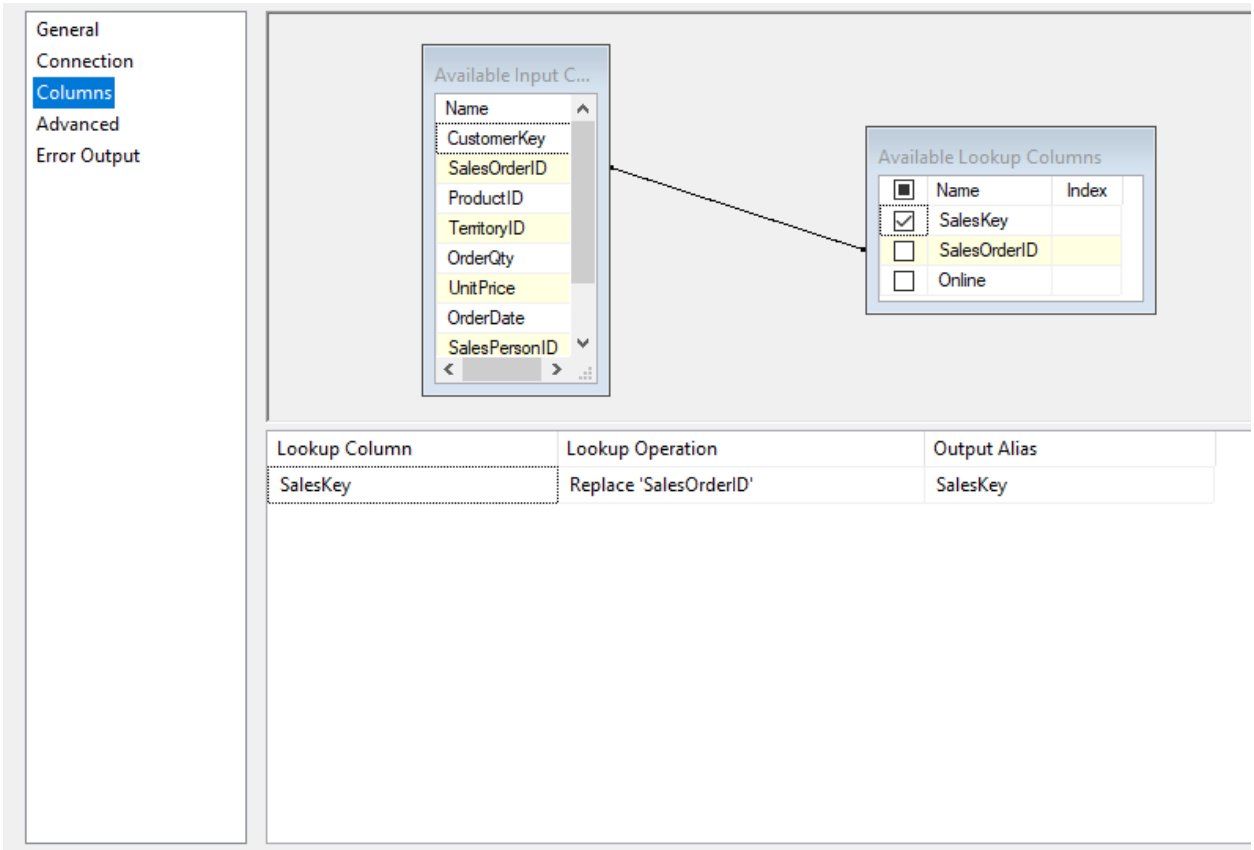
We must create a data source first. From this data source, all lookups to will be performed and any transformations necessary will also be performed (date conversion to map to date dimension). The query below brings everything into one large table. This will be our main data source in the data flow to produce our fact table.



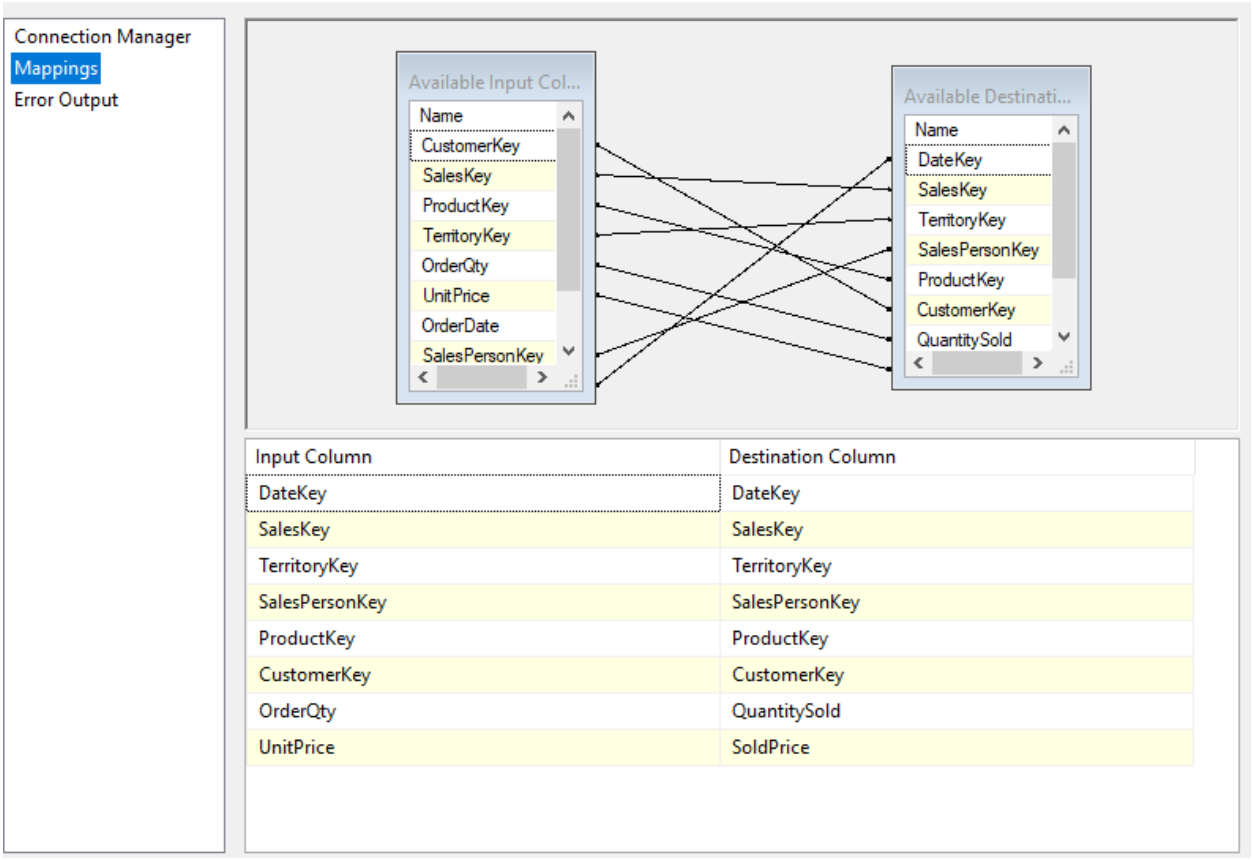

Quick Launch (Ctrl+Q)
SSIS Tools Window Help
Continue
Stack Frame:
Package.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Sales Fact
OLE DB Source
35,649 rows
Derived Column
35,649 rows
Customer Look Up
Lookup Match Output (35,649 rows)
Time Look Up
Lookup Match Output (35,649 rows)
Sales Look Up
Lookup Match Output (35,649 rows)
Sales Person Look Up
Lookup Match Output (35,649 rows)
Territory Look Up



Here is an example of how the look up function should look when mapping the columns. Here, the Sales look up is used as an example.

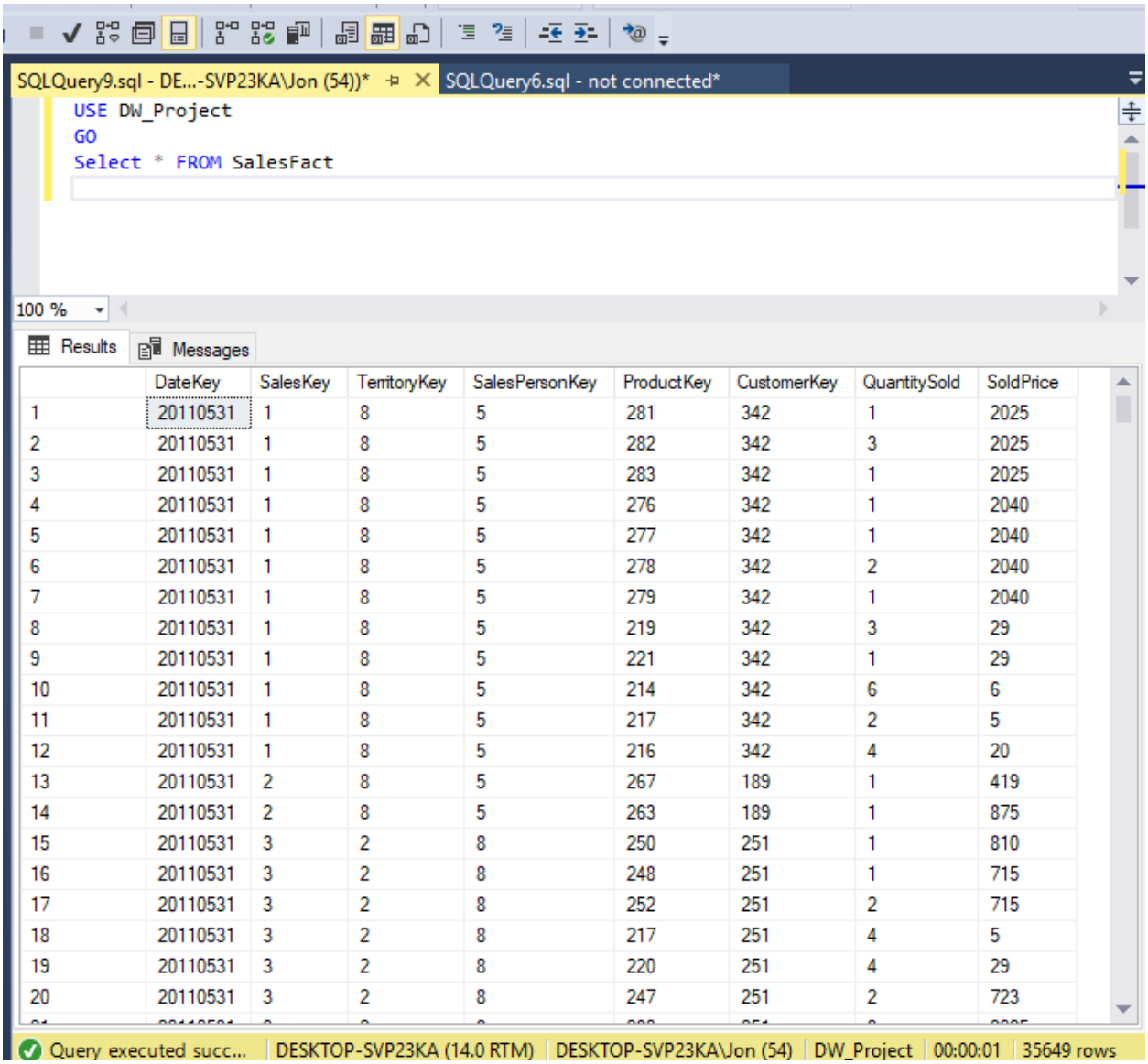


And below is what the final mappings in the sales fact table look like.



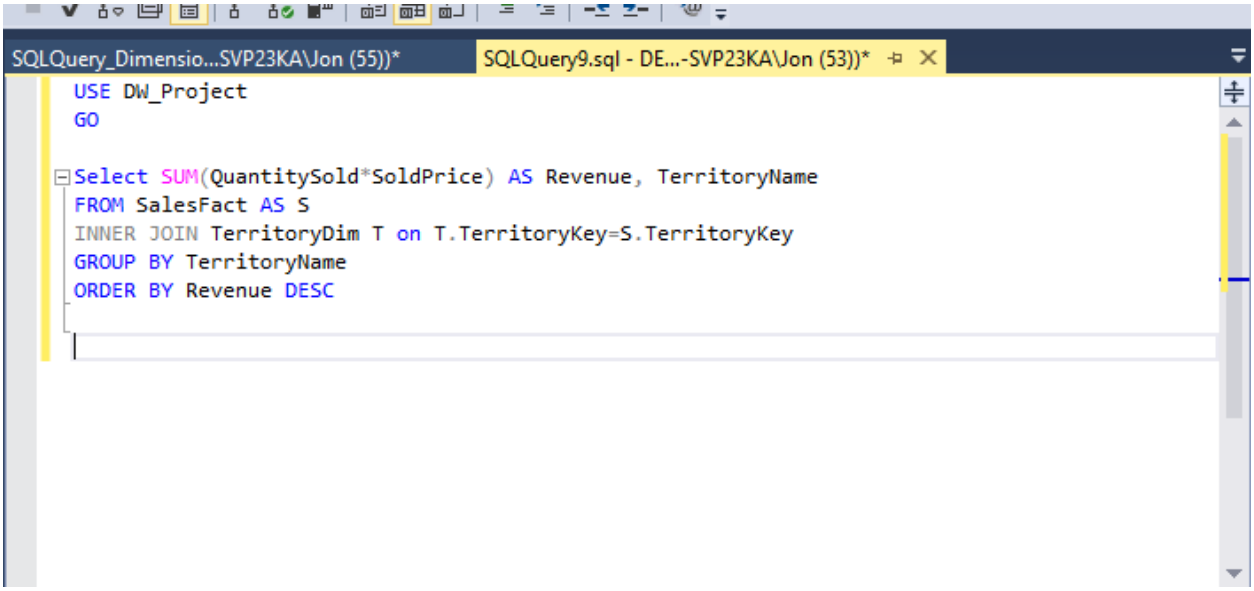
Input Column	Destination Column
DateKey	DateKey
SalesKey	SalesKey
TerritoryKey	TerritoryKey
SalesPersonKey	SalesPersonKey
ProductKey	ProductKey
CustomerKey	CustomerKey
OrderQty	QuantitySold
UnitPrice	SoldPrice

Also, please note that in the above workflow, you will see a derived column transformation. This was due to the fact that I had to make sure that the data formats matched between my source data and destination. Originally, the formats were different so when I tried doing the look up using the date, I got an error message saying that the look up could not be done. After performing this transformation and ensuring that the date formats matched, the look ups worked fine and I proceeded with the rest of the workflow.

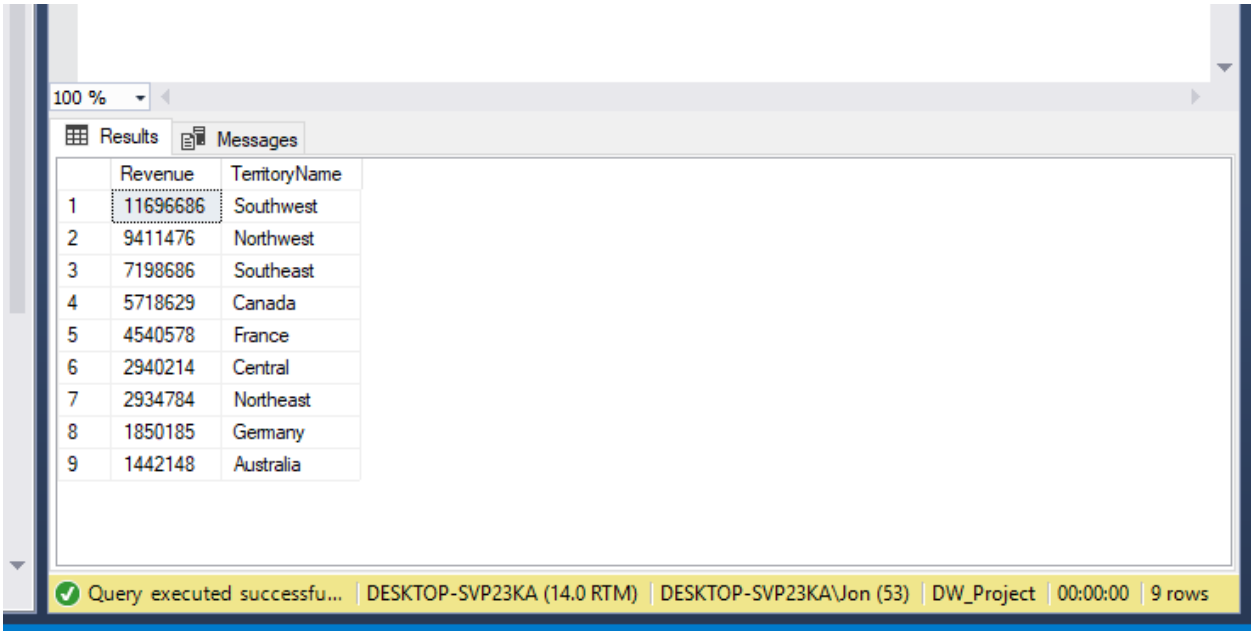


	DateKey	SalesKey	TerritoryKey	SalesPersonKey	ProductKey	CustomerKey	QuantitySold	SoldPrice
1	20110531	1	8	5	281	342	1	2025
2	20110531	1	8	5	282	342	3	2025
3	20110531	1	8	5	283	342	1	2025
4	20110531	1	8	5	276	342	1	2040
5	20110531	1	8	5	277	342	1	2040
6	20110531	1	8	5	278	342	2	2040
7	20110531	1	8	5	279	342	1	2040
8	20110531	1	8	5	219	342	3	29
9	20110531	1	8	5	221	342	1	29
10	20110531	1	8	5	214	342	6	6
11	20110531	1	8	5	217	342	2	5
12	20110531	1	8	5	216	342	4	20
13	20110531	2	8	5	267	189	1	419
14	20110531	2	8	5	263	189	1	875
15	20110531	3	2	8	250	251	1	810
16	20110531	3	2	8	248	251	1	715
17	20110531	3	2	8	252	251	2	715
18	20110531	3	2	8	217	251	4	5
19	20110531	3	2	8	220	251	4	29
20	20110531	3	2	8	247	251	2	723

So how may we want to utilize this fact table? Well, as an example, suppose as a business, we wanted to know more about the total sales of our products and where total sales are strongest or weakest. Let's have a look at the query below:



Here, we have a query that will give us total revenue for a given territory. After running this query, we get the output below.



Here, we can see that sales appear strongest in the southwest United States and weakest in Australia. This could be because the number of stores is much less in Australia or there could be many other reasons. But this is just one example of how we can utilize this data warehouse to gain insights about our business. We could run similar queries to get more detailed information about which products are selling where as well as sales breakdowns by gender or marital status. This could inform how we market different items to certain customers based on their demographics.

Lesson Learned:

- The things I learned from this project are numerous. Starting off, I was a bit too ambitious and felt that using a technology like Python to do ETL would be preferable. After reading quite a bit, I decided I'd better use the standard technologies for a first time data warehouse builder like myself. I spent a great number of hours just learning about SSMS and SSIS technologies as these were brand new to me before this class. It's one thing to know and understand dimensional modeling and the ETL process from a conceptual point of view, but it's an entirely new world learning the appropriate technologies which help you apply those concepts in the real world.
- It's very important to really understand the data before building your dimensional model. This is something I always firmly believe before doing any sort of analysis or really any work with data. I really had to study the Adventure Works database ERD as well as looking up various resources for it like data dictionaries. The ERD, while it may be enough for experienced warehousing professionals, didn't quite give me enough information about the different entities. Maybe this is something I need to improve on but I also spent a great deal of time understanding the relationships between the tables so that when I built my dimensional model, it would actually make sense and would allow the final data warehouse to provide some value to a business owner (even though this is a fictitious company and not a real-world scenario).
- In terms of next steps in the learning process, I think it would be of great value to understand how to connect this data warehouse to something like Tableau or some other interactive visualization tool like D3 or Shiny in R. Then you could really show-off the business value of the warehouse in a way that most people could see and understand easily. I may try to do this with the warehouse I've created here. I would have liked to have done it here but I just didn't quite have the time to figure that extra piece out.