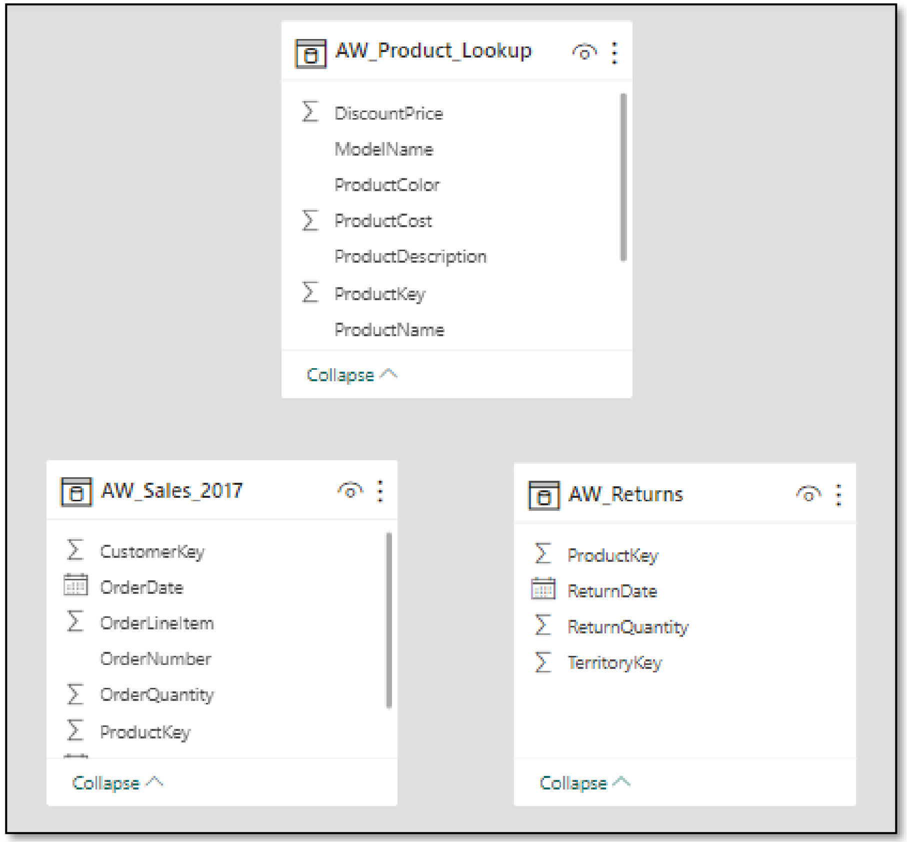
CREATING DATA MODEL

What is a Data Model



AW_Product_Lookup

- DiscountPrice
- ModelName
- ProductColor
- ProductCost
- ProductDescription
- ProductKey
- ProductName

AW_Sales_2017

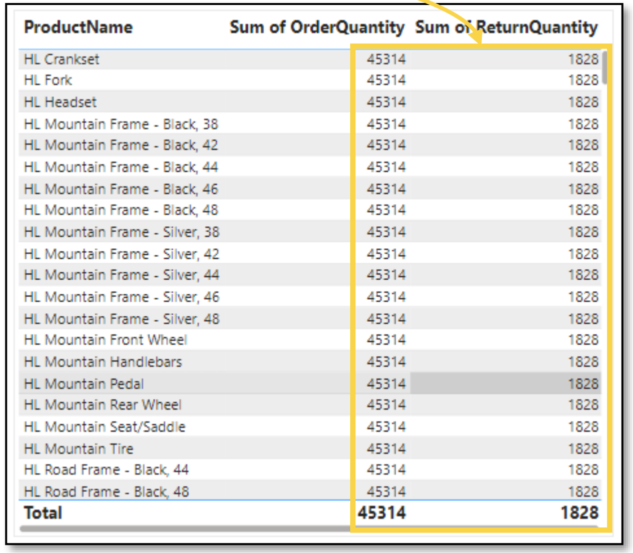
- CustomerKey
- OrderDate
- OrderLineItem
- OrderNumber
- OrderQuantity
- ProductKey

AW_Returns

- ProductKey
- ReturnDate
- ReturnQuantity
- TerritoryKey

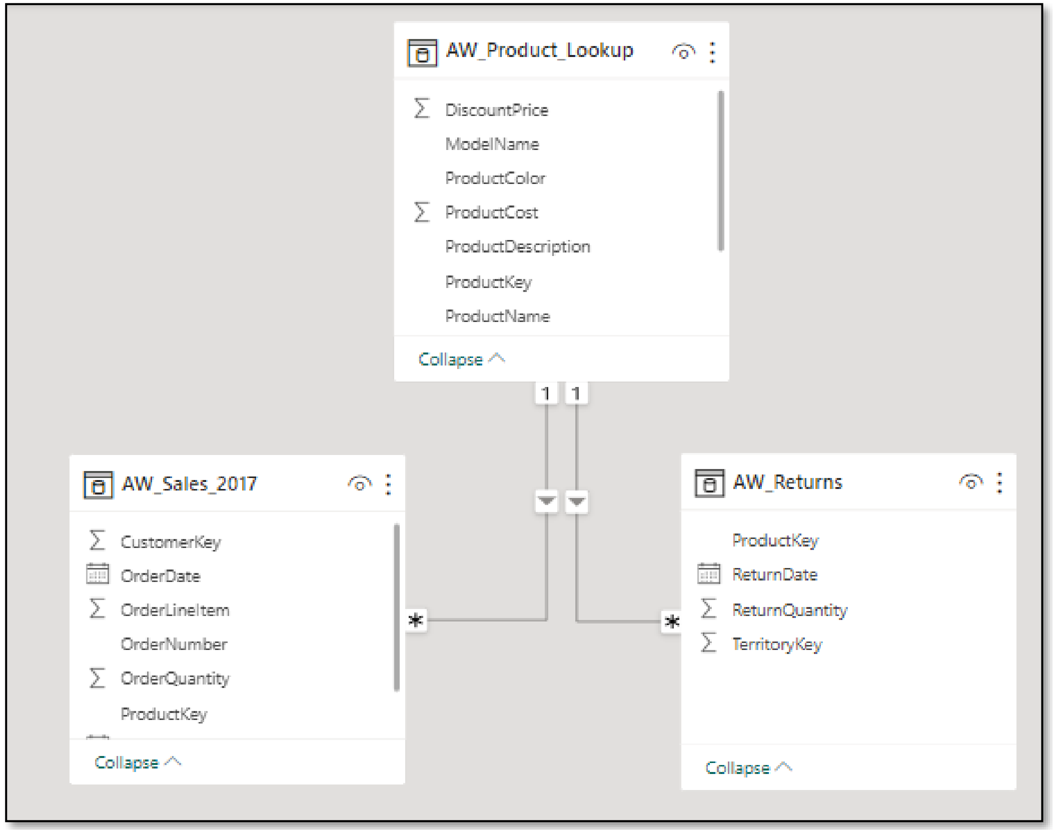
This **IS NOT** a data Model ☹

- This is a collection of independent tables, which share no connections or relationships
- If you tried to visualize Orders and Returns by Product, this is what you'd get



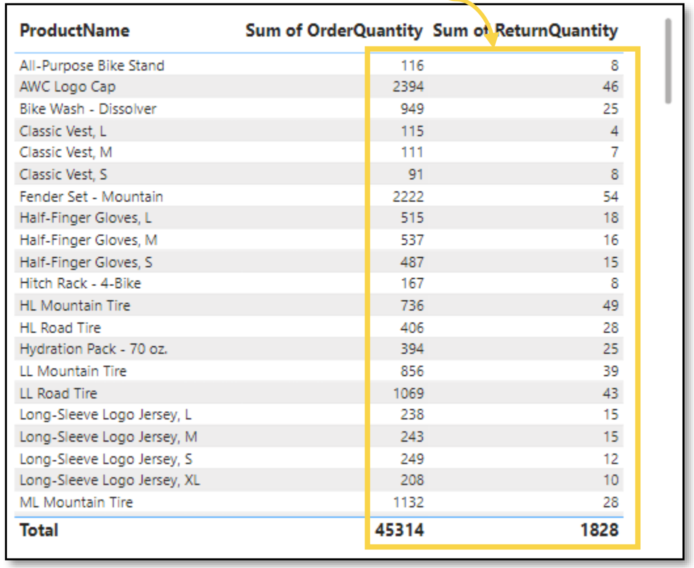
ProductName	Sum of OrderQuantity	Sum of ReturnQuantity
HL Crankset	45314	1828
HL Fork	45314	1828
HL Headset	45314	1828
HL Mountain Frame - Black, 38	45314	1828
HL Mountain Frame - Black, 42	45314	1828
HL Mountain Frame - Black, 44	45314	1828
HL Mountain Frame - Black, 46	45314	1828
HL Mountain Frame - Black, 48	45314	1828
HL Mountain Frame - Silver, 38	45314	1828
HL Mountain Frame - Silver, 42	45314	1828
HL Mountain Frame - Silver, 44	45314	1828
HL Mountain Frame - Silver, 46	45314	1828
HL Mountain Frame - Silver, 48	45314	1828
HL Mountain Front Wheel	45314	1828
HL Mountain Handlebars	45314	1828
HL Mountain Pedal	45314	1828
HL Mountain Rear Wheel	45314	1828
HL Mountain Seat/Saddle	45314	1828
HL Mountain Tire	45314	1828
HL Road Frame - Black, 44	45314	1828
HL Road Frame - Black, 48	45314	1828
Total	**45314**	**1828**

What is a Data Model



This **IS** a data model! 😀

- The tables are connected via relationships, based on the common ProductKey
- Now the Sales and Returns tables know how to filter using fields from the Product table!



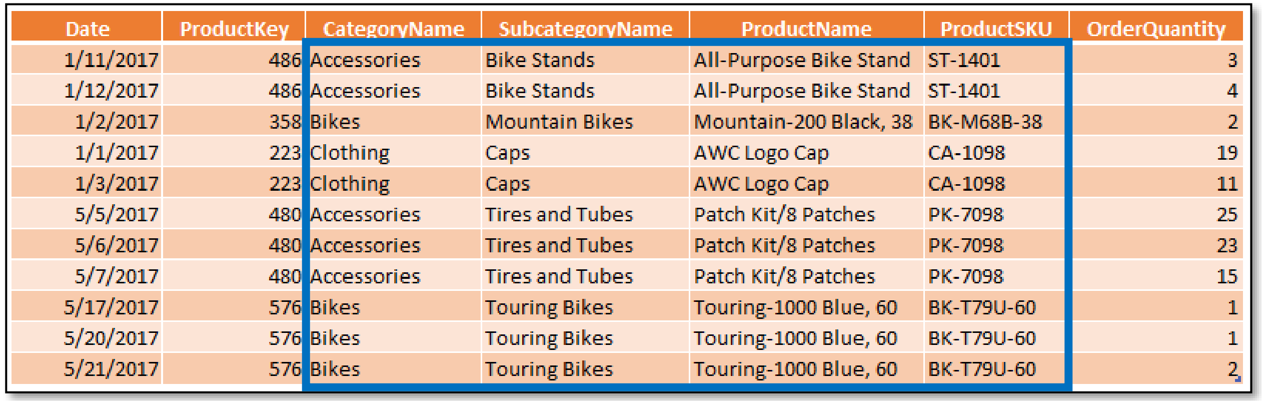
ProductName	Sum of OrderQuantity	Sum of ReturnQuantity
All-Purpose Bike Stand	116	8
AWC Logo Cap	2394	46
Bike Wash - Dissolver	949	25
Classic Vest, L	115	4
Classic Vest, M	111	7
Classic Vest, S	91	8
Fender Set - Mountain	2222	54
Half-Finger Gloves, L	515	18
Half-Finger Gloves, M	537	16
Half-Finger Gloves, S	487	15
Hitch Rack - 4-Bike	167	8
HL Mountain Tire	736	49
HL Road Tire	406	28
Hydration Pack - 70 oz.	394	25
LL Mountain Tire	856	39
LL Road Tire	1069	43
Long-Sleeve Logo Jersey, L	238	15
Long-Sleeve Logo Jersey, M	243	15
Long-Sleeve Logo Jersey, S	249	12
Long-Sleeve Logo Jersey, XL	208	10
ML Mountain Tire	1132	28
Total	**45314**	**1828**

Database Normalization

Normalization is the process of organizing the tables and columns in a relational database to reduce redundancy and preserve data integrity. It’s commonly used to

- **Eliminate redundant data** to decrease table sizes and improve processing speed & efficiency
- **Minimize errors and anomalies** from data modifications (inserting, updating or deleting records)
- **Simplify queries** and structure the database for meaningful analysis

TIP: In a normalized database, each table should serve a **distinct** and **specific** purpose (example: product information, dates, transaction records, customer attributes, etc.)



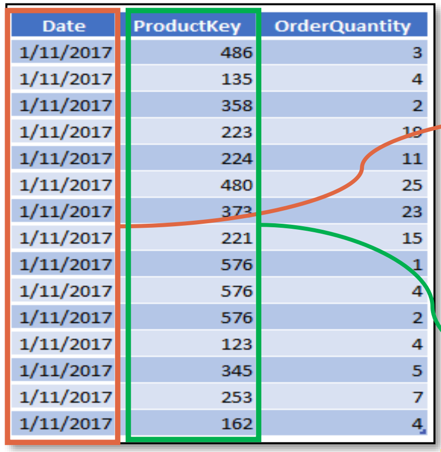
Date	ProductKey	CategoryName	SubcategoryName	ProductName	ProductSKU	OrderQuantity
1/11/2017	486	Accessories	Bike Stands	All-Purpose Bike Stand	ST-1401	3
1/12/2017	486	Accessories	Bike Stands	All-Purpose Bike Stand	ST-1401	4
1/2/2017	358	Bikes	Mountain Bikes	Mountain-200 Black, 38	BK-M68B-38	2
1/1/2017	223	Clothing	Caps	AWC Logo Cap	CA-1098	19
1/3/2017	223	Clothing	Caps	AWC Logo Cap	CA-1098	11
5/5/2017	480	Accessories	Tires and Tubes	Patch Kit/8 Patches	PK-7098	25
5/6/2017	480	Accessories	Tires and Tubes	Patch Kit/8 Patches	PK-7098	23
5/7/2017	480	Accessories	Tires and Tubes	Patch Kit/8 Patches	PK-7098	15
5/17/2017	576	Bikes	Touring Bikes	Touring-1000 Blue, 60	BK-T79U-60	1
5/20/2017	576	Bikes	Touring Bikes	Touring-1000 Blue, 60	BK-T79U-60	1
5/21/2017	576	Bikes	Touring Bikes	Touring-1000 Blue, 60	BK-T79U-60	2

This may not seem critical now, but minor inefficiencies can become major problems as databases scale in size!

Data Tables vs Lookup Tables

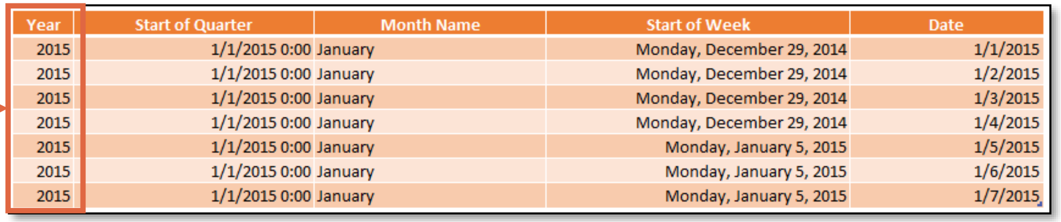
Models generally contain two types of tables: **data** (or “fact”) **tables**, and lookup (or “dimension”) tables

- **Data tables** contain numbers or values, typically at a granular level, with ID or “key” columns that can be used to create table relationships
- **Lookup tables** provide descriptive, often text-based attributes about each dimension in a table



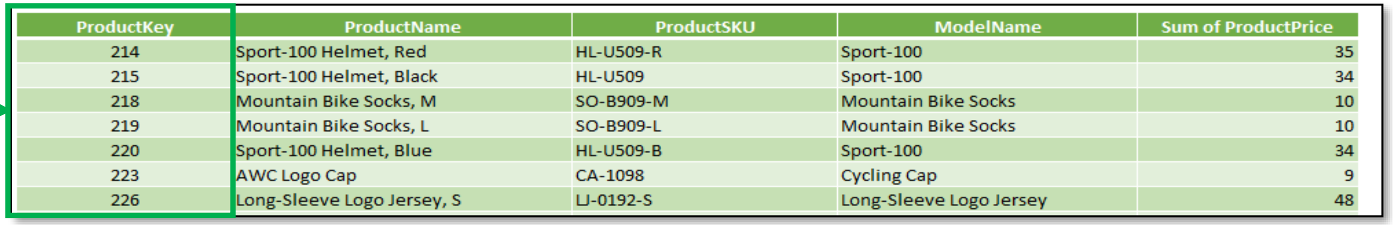
Date	ProductKey	OrderQuantity
1/11/2017	486	3
1/11/2017	135	4
1/11/2017	358	2
1/11/2017	223	10
1/11/2017	224	11
1/11/2017	480	25
1/11/2017	373	23
1/11/2017	221	15
1/11/2017	576	1
1/11/2017	576	4
1/11/2017	576	2
1/11/2017	123	4
1/11/2017	345	5
1/11/2017	253	7
1/11/2017	162	4

This **Data Table** contains **“quantity”** values, and connects to lookup tables via the **“date”** and **“product key”** columns.



Year	Start of Quarter	Month Name	Start of Week	Date
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/1/2015
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/2/2015
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/3/2015
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/4/2015
2015	1/1/2015 0:00	January	Monday, January 5, 2015	1/5/2015
2015	1/1/2015 0:00	January	Monday, January 5, 2015	1/6/2015
2015	1/1/2015 0:00	January	Monday, January 5, 2015	1/7/2015

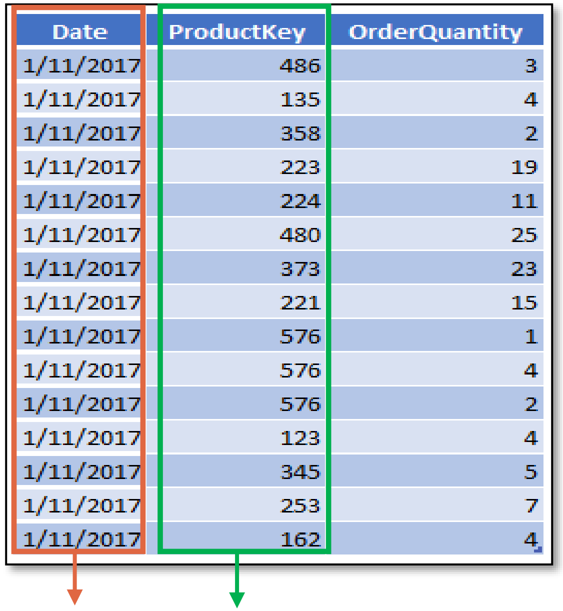
This **Product Lookup** table provides additional attributes about each **product** (product name, sku, model name, price, etc.)



ProductKey	ProductName	ProductSKU	ModelName	Sum of ProductPrice
214	Sport-100 Helmet, Red	HL-U509-R	Sport-100	35
215	Sport-100 Helmet, Black	HL-U509	Sport-100	34
218	Mountain Bike Socks, M	SO-B909-M	Mountain Bike Socks	10
219	Mountain Bike Socks, L	SO-B909-L	Mountain Bike Socks	10
220	Sport-100 Helmet, Blue	HL-U509-B	Sport-100	34
223	AWC Logo Cap	CA-1098	Cycling Cap	9
226	Long-Sleeve Logo Jersey, S	LJ-0192-S	Long-Sleeve Logo Jersey	48

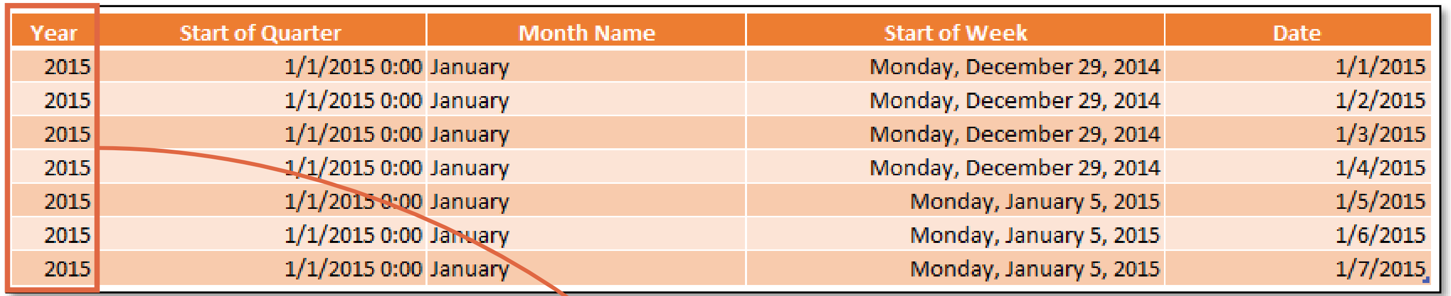
This **Product Lookup** table provides additional attributes about each **product** (product name, sku, model name, price, etc.)

Primary vs Foreign Keys

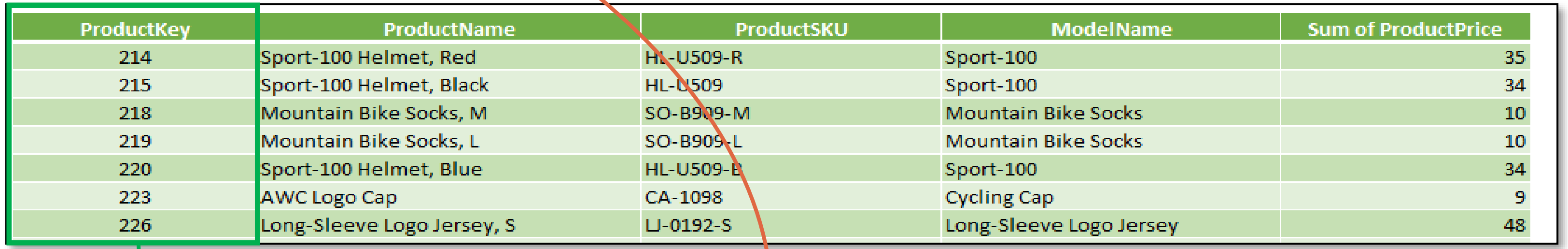


Date	ProductKey	OrderQuantity
1/11/2017	486	3
1/11/2017	135	4
1/11/2017	358	2
1/11/2017	223	19
1/11/2017	224	11
1/11/2017	480	25
1/11/2017	373	23
1/11/2017	221	15
1/11/2017	576	1
1/11/2017	576	4
1/11/2017	576	2
1/11/2017	123	4
1/11/2017	345	5
1/11/2017	253	7
1/11/2017	162	4

These columns are **foreign keys**; they contain multiple instances of each value, and are used to match the **primary keys** in related lookup tables.



Year	Start of Quarter	Month Name	Start of Week	Date
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/1/2015
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/2/2015
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/3/2015
2015	1/1/2015 0:00	January	Monday, December 29, 2014	1/4/2015
2015	1/1/2015 0:00	January	Monday, January 5, 2015	1/5/2015
2015	1/1/2015 0:00	January	Monday, January 5, 2015	1/6/2015
2015	1/1/2015 0:00	January	Monday, January 5, 2015	1/7/2015



ProductKey	ProductName	ProductSKU	ModelName	Sum of ProductPrice
214	Sport-100 Helmet, Red	HL-U509-R	Sport-100	35
215	Sport-100 Helmet, Black	HL-U509	Sport-100	34
218	Mountain Bike Socks, M	SO-B909-M	Mountain Bike Socks	10
219	Mountain Bike Socks, L	SO-B909-L	Mountain Bike Socks	10
220	Sport-100 Helmet, Blue	HL-U509-B	Sport-100	34
223	AWC Logo Cap	CA-1098	Cycling Cap	9
226	Long-Sleeve Logo Jersey, S	LJ-0192-S	Long-Sleeve Logo Jersey	48

These columns are **primary keys**; they uniquely identify each row of a table, and match the **foreign keys** in related data tables

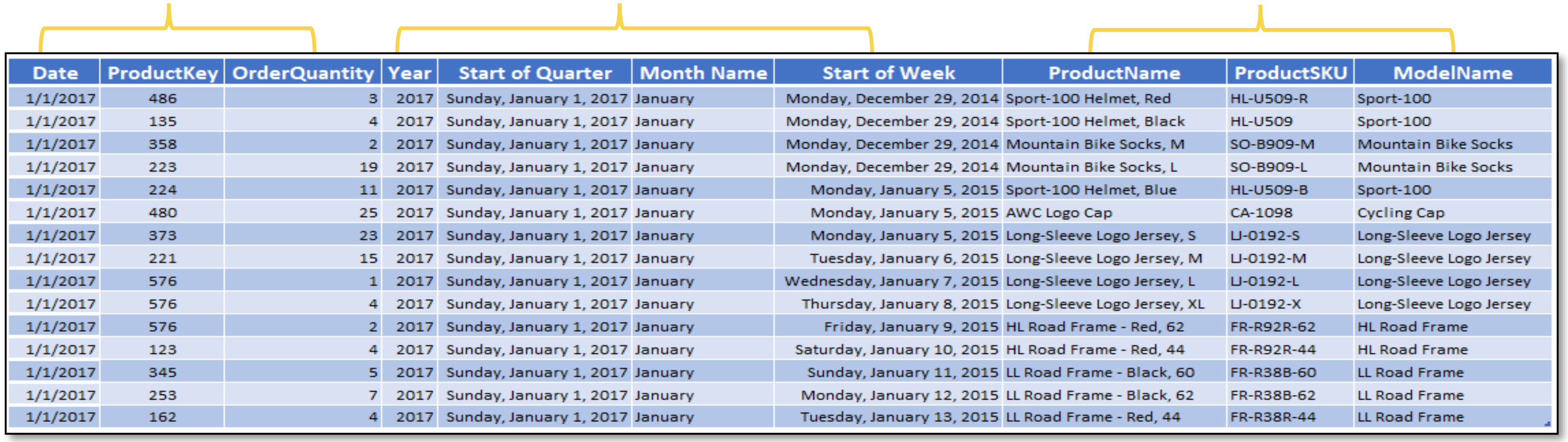
Relationship vs Merged Tables

Can’t I just **merge queries** or use **LOOKUP** or **RELATED** functions to pull those attributes into the fact table itself, so that I have everything in one place ??

Original **Fact Table** fields

Attributes from Table **Calendar Lookup** table

Attributes from Table **Product Lookup** table



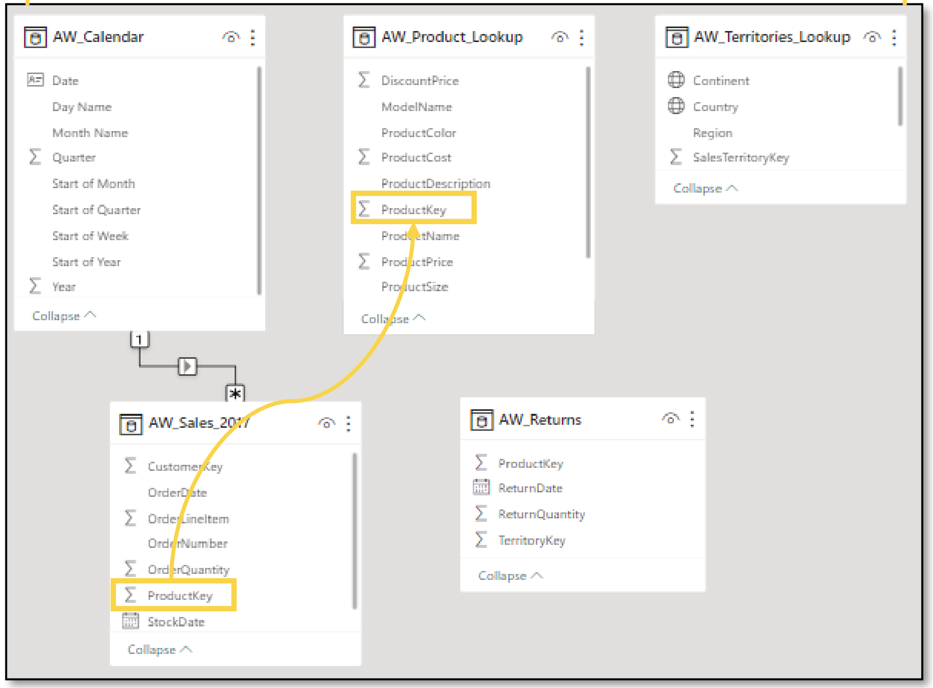
Date	ProductKey	OrderQuantity	Year	Start of Quarter	Month Name	Start of Week	ProductName	ProductSKU	ModelName
1/1/2017	486	3	2017	Sunday, January 1, 2017	January	Monday, December 29, 2014	Sport-100 Helmet, Red	HL-U509-R	Sport-100
1/1/2017	135	4	2017	Sunday, January 1, 2017	January	Monday, December 29, 2014	Sport-100 Helmet, Black	HL-U509	Sport-100
1/1/2017	358	2	2017	Sunday, January 1, 2017	January	Monday, December 29, 2014	Mountain Bike Socks, M	SO-B909-M	Mountain Bike Socks
1/1/2017	223	19	2017	Sunday, January 1, 2017	January	Monday, December 29, 2014	Mountain Bike Socks, L	SO-B909-L	Mountain Bike Socks
1/1/2017	224	11	2017	Sunday, January 1, 2017	January	Monday, January 5, 2015	Sport-100 Helmet, Blue	HL-U509-B	Sport-100
1/1/2017	480	25	2017	Sunday, January 1, 2017	January	Monday, January 5, 2015	AWC Logo Cap	CA-1098	Cycling Cap
1/1/2017	373	23	2017	Sunday, January 1, 2017	January	Monday, January 5, 2015	Long-Sleeve Logo Jersey, S	LJ-0192-S	Long-Sleeve Logo Jersey
1/1/2017	221	15	2017	Sunday, January 1, 2017	January	Tuesday, January 6, 2015	Long-Sleeve Logo Jersey, M	LJ-0192-M	Long-Sleeve Logo Jersey
1/1/2017	576	1	2017	Sunday, January 1, 2017	January	Wednesday, January 7, 2015	Long-Sleeve Logo Jersey, L	LJ-0192-L	Long-Sleeve Logo Jersey
1/1/2017	576	4	2017	Sunday, January 1, 2017	January	Thursday, January 8, 2015	Long-Sleeve Logo Jersey, XL	LJ-0192-X	Long-Sleeve Logo Jersey
1/1/2017	576	2	2017	Sunday, January 1, 2017	January	Friday, January 9, 2015	HL Road Frame - Red, 62	FR-R92R-62	HL Road Frame
1/1/2017	123	4	2017	Sunday, January 1, 2017	January	Saturday, January 10, 2015	HL Road Frame - Red, 44	FR-R92R-44	HL Road Frame
1/1/2017	345	5	2017	Sunday, January 1, 2017	January	Sunday, January 11, 2015	LL Road Frame - Black, 60	FR-R38B-60	LL Road Frame
1/1/2017	253	7	2017	Sunday, January 1, 2017	January	Monday, January 12, 2015	LL Road Frame - Black, 62	FR-R38B-62	LL Road Frame
1/1/2017	162	4	2017	Sunday, January 1, 2017	January	Tuesday, January 13, 2015	LL Road Frame - Red, 44	FR-R38R-44	LL Road Frame

Sure you can, but it’s **inefficient**!

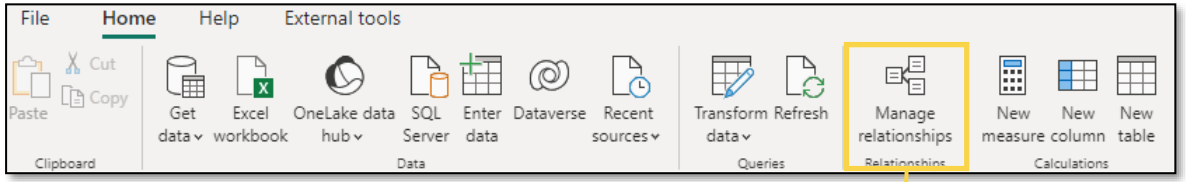
- Merging data in this way creates **redundant data** and utilizes **significantly more memory and processing power** than creating relationships between multiple small tables

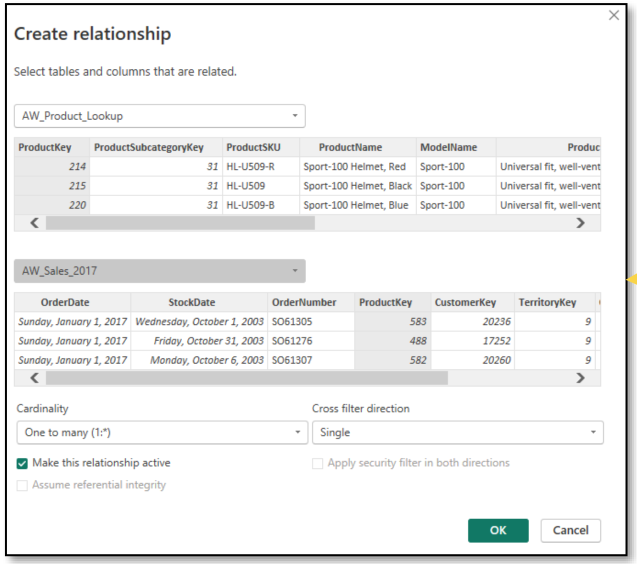
Creating Table Relationship

Option 1: Click and drag to connect primary and foreign keys within the **Relationships** pane



Option 2: Add or detect relationships using the **"Manage Relationships"** dialog box





Create relationship

Select tables and columns that are related.

AW_Product_Lookup

ProductKey	ProductSubcategoryKey	ProductSKU	ProductName	ModelName	Produc
214	31	HL-U509-R	Sport-100 Helmet, Red	Sport-100	Universal fit, well-vent
215	31	HL-U509	Sport-100 Helmet, Black	Sport-100	Universal fit, well-vent
220	31	HL-U509-B	Sport-100 Helmet, Blue	Sport-100	Universal fit, well-vent

AW_Sales_2017

OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey
Sunday, January 1, 2017	Wednesday, October 1, 2003	SO61305	583	20236	9
Sunday, January 1, 2017	Friday, October 31, 2003	SO61276	488	17252	9
Sunday, January 1, 2017	Monday, October 6, 2003	SO61307	582	20260	9

Cardinality: One to many (1:*)

Cross filter direction: Single

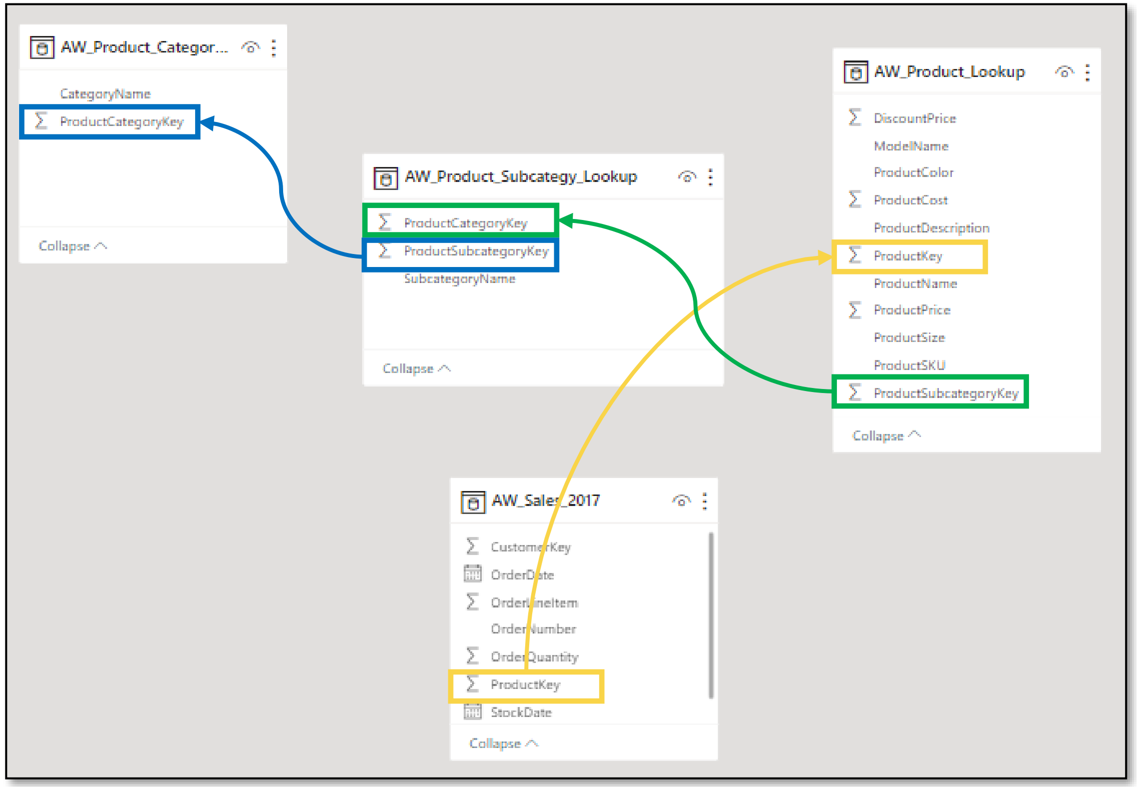
☑ Make this relationship active

☐ Apply security filter in both directions

☐ Assume referential integrity

OK Cancel

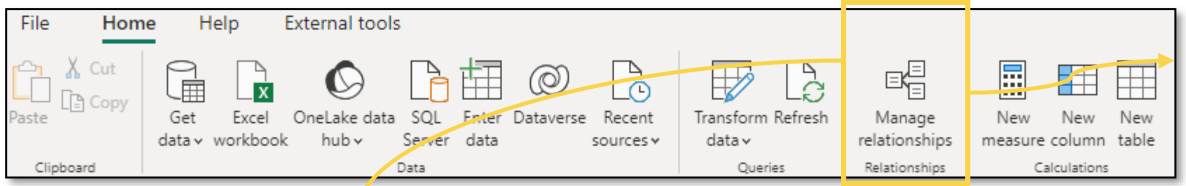
Creating Snowflake Schemas

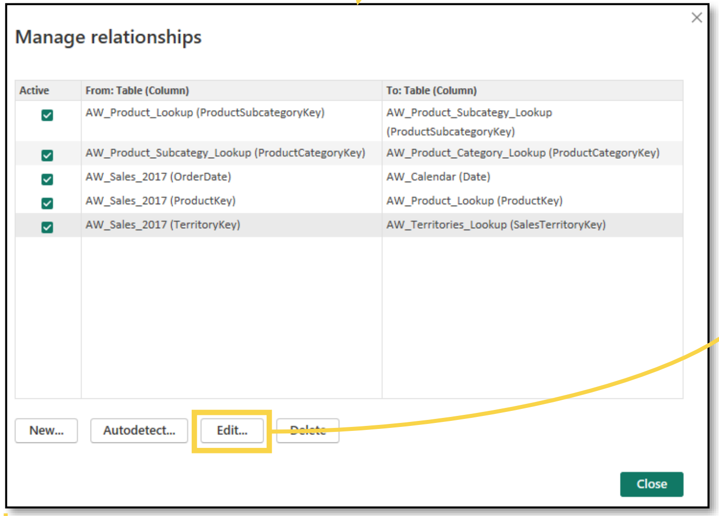


The **AW_Sales_2017** table can connect to AW_Product_Lookup using the ProductKey field, but cannot connect directly to the **AW_Product_Subcategories_Lookup** or **AW_Product_Categories_Lookup** tables

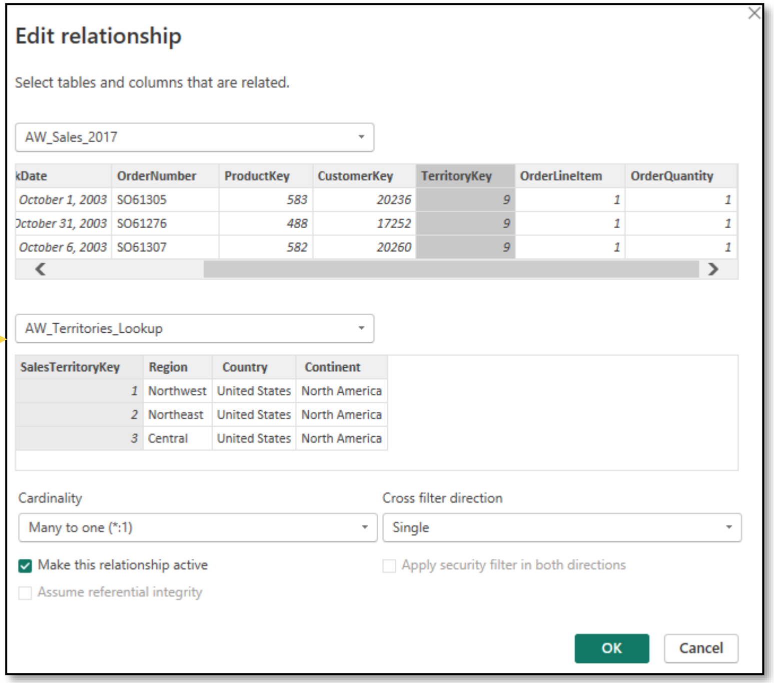
By creating relationships from **AW_Products_Lookup** to **AW_Product_Subcategories_Lookup** (using ProductSubcategoryKey) and **AW_Product_Subcategories_Lookup** to **AW_Product_Categories_Lookup** (using ProductCategoryKey), we have essentially connected **AW_Sales_2017** to each lookup table; filter context will now flow all the way down the chain.

Managing & Editing Relationships




Manage relationships

Active	From: Table (Column)	To: Table (Column)
☑	AW_Product_Lookup (ProductSubcategoryKey)	AW_Product_Subcategy_Lookup (ProductSubcategoryKey)
☑	AW_Product_Subcategy_Lookup (ProductCategoryKey)	AW_Product_Category_Lookup (ProductCategoryKey)
☑	AW_Sales_2017 (OrderDate)	AW_Calendar (Date)
☑	AW_Sales_2017 (ProductKey)	AW_Product_Lookup (ProductKey)
☑	AW_Sales_2017 (TerritoryKey)	AW_Territories_Lookup (SalesTerritoryKey)


Edit relationship

Select tables and columns that are related.

AW_Sales_2017

kDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity
October 1, 2003	SO61305	583	20236	9	1	1
)ctober 31, 2003	SO61276	488	17252	9	1	1
October 6, 2003	SO61307	582	20260	9	1	1

AW_Territories_Lookup

SalesTerritoryKey	Region	Country	Continent
1	Northwest	United States	North America
2	Northeast	United States	North America
3	Central	United States	North America

Cardinality: Many to one (*:1)

Cross filter direction: Single

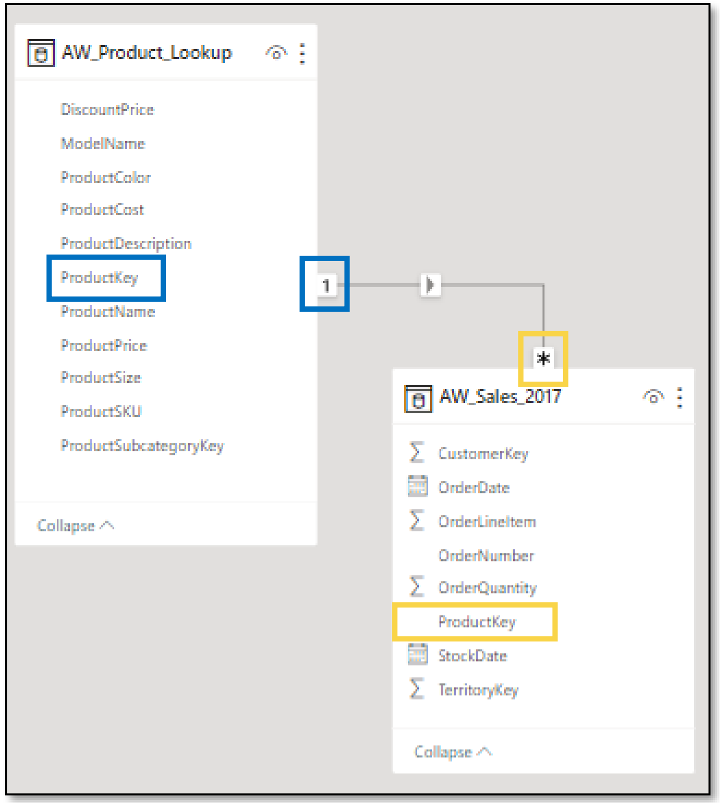
☑ Make this relationship active

□ Apply security filter in both directions

□ Assume referential integrity

The **"Manage Relationships"** dialog box allows you to **add**, **edit**, or **delete** table relationships

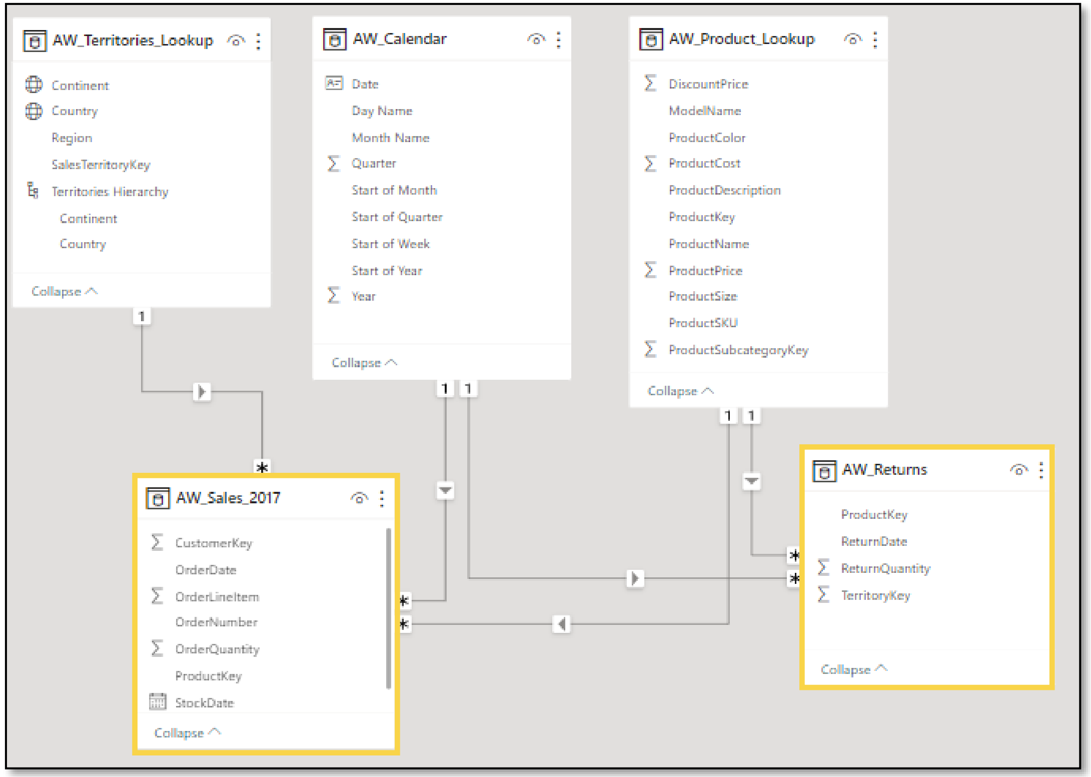
Relationship Cardinality



Cardinality refers to the uniqueness of values in a column

- For our purposes, all relationships in the data model should follow a **"one-to-many"** cardinality; **one** instance of each primary key, but potentially **many** instances of each foreign key

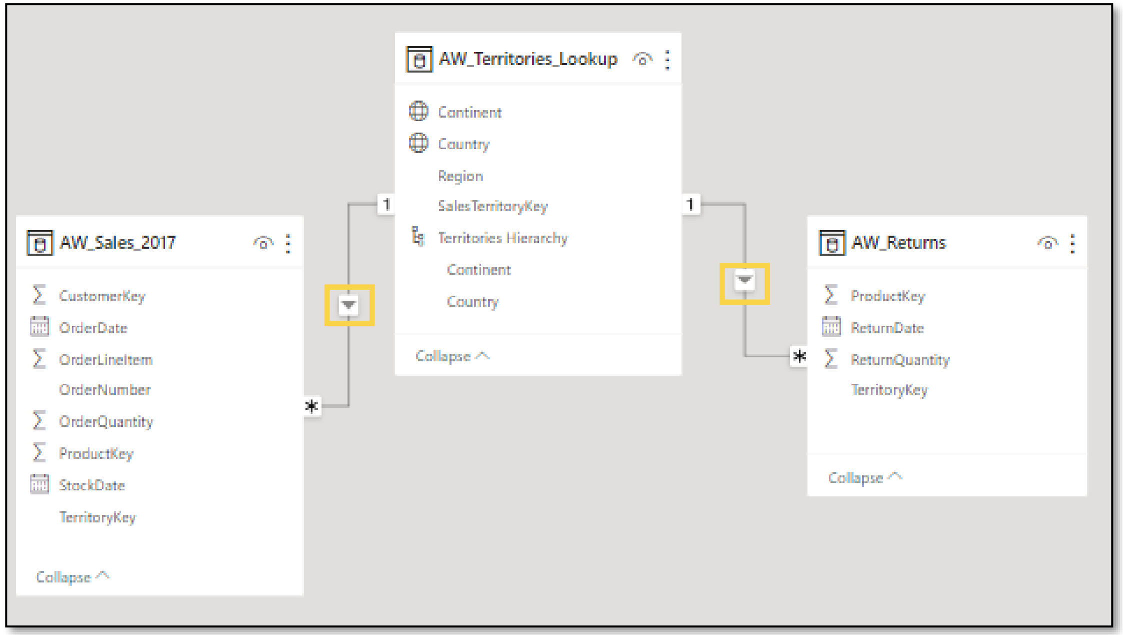
Connecting Multiple Data Tables



This model contains two data tables: **AW_Sales_Data_2017** and **AW_Returns**

- Note that the Returns table connects to **AW_Calendar** and **AW_Product_Lookup** just like the Sales table, but without a *CustomerKey* field it cannot be joined to **AW_Territoties_Lookup.**
- In other words, we know which **product** was returned and on which **date**, but nothing about which **territories** made the return

Filter Flow

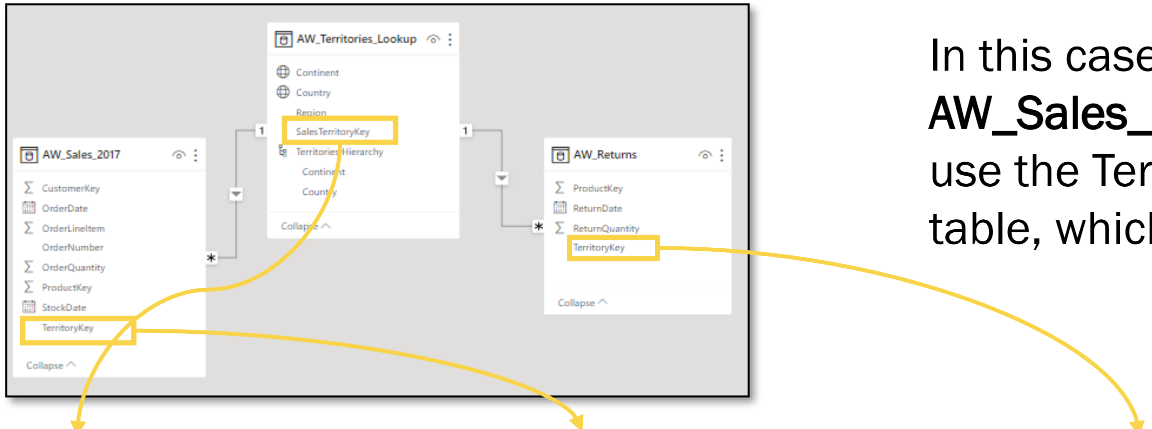


Here we have two data tables (**AW_Sales_Data_2017** and **AW_Returns**)

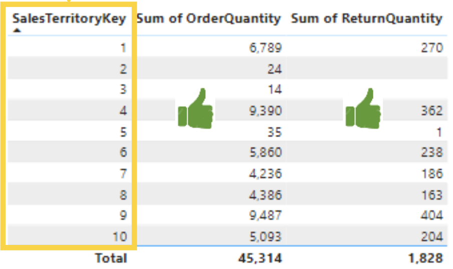
Note the filter directions (shown as arrows) in each relationship; by default, **these will point from the "one" side of the relationship** (lookups) **to the "many" side** (data)

- When you filter a table, that filter context is passed along to all related "*downstream*" tables (following the direction of the arrow)
- Filters cannot flow "*upstream*"

Filter Flow (Cont.)

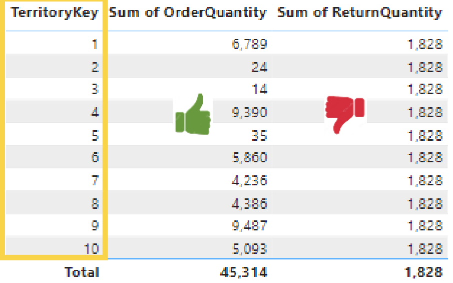


In this case, the only valid way filter both **AW_Sales_2017** and **AW_Returns** data by Territory is to use the TerritoryKey field from the **AW_Territory_Lookup** table, which is upstream and related to both data tables



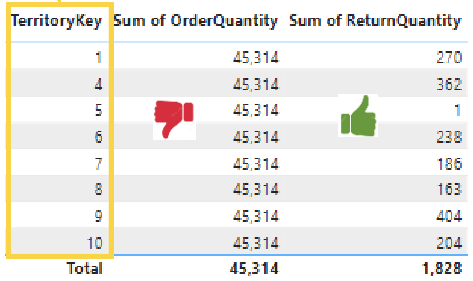
SalesTerritoryKey	Sum of OrderQuantity 👍	Sum of ReturnQuantity 👍
1	6,789	270
2	24	
3	14	
4	9,390	362
5	35	1
6	5,860	238
7	4,236	186
8	4,386	163
9	9,487	404
10	5,093	204
Total	**45,314**	**1,828**

1) Filtering using TerritoryKey from the **AW_Territory_Lookup** table



TerritoryKey	Sum of OrderQuantity 👍	Sum of ReturnQuantity 👎
1	6,789	1,828
2	24	1,828
3	14	1,828
4	9,390	1,828
5	35	1,828
6	5,860	1,828
7	4,236	1,828
8	4,386	1,828
9	9,487	1,828
10	5,093	1,828
Total	**45,314**	**1,828**

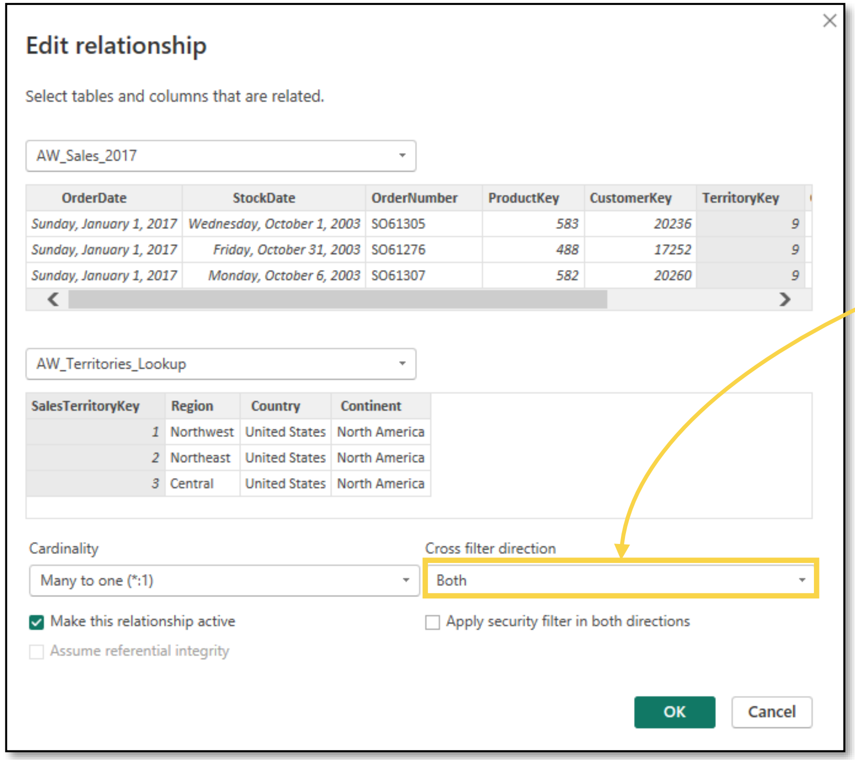
2) Filtering using TerritoryKey from the **AW_Sales_2017** table



TerritoryKey	Sum of OrderQuantity 👎	Sum of ReturnQuantity 👍
1	45,314	270
4	45,314	362
5	45,314	1
6	45,314	238
7	45,314	186
8	45,314	163
9	45,314	404
10	45,314	204
Total	**45,314**	**1,828**

3. Filtering using TerritoryKey from the **AW_Returns** table

Two-Way Filters



Edit relationship

Select tables and columns that are related.

AW_Sales_2017

OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey
Sunday, January 1, 2017	*Wednesday, October 1, 2003*	SO61305	*583*	*20236*	*9*
Sunday, January 1, 2017	*Friday, October 31, 2003*	SO61276	*488*	*17252*	*9*
Sunday, January 1, 2017	*Monday, October 6, 2003*	SO61307	*582*	*20260*	*9*

AW_Territories_Lookup

SalesTerritoryKey	Region	Country	Continent
1	Northwest	United States	North America
2	Northeast	United States	North America
3	Central	United States	North America

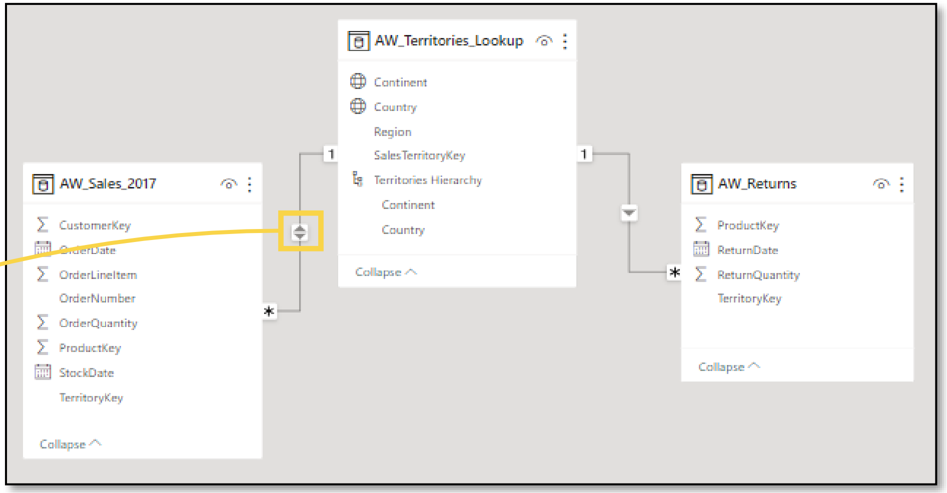
Cardinality: Many to one (*:1)

Cross filter direction: Both

☑ Make this relationship active

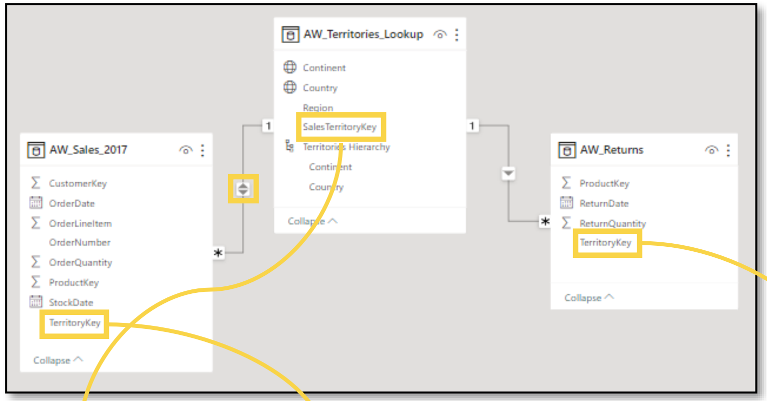
□ Apply security filter in both directions

□ Assume referential integrity



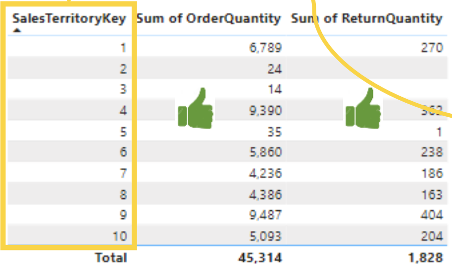
Updating the filter direction between **AW_Sales_2017** and **AW_Territories_Lookup** from "*Single*" to "*Both*" allows filter context to flow both ways

Two-Way Filters (Cont.)



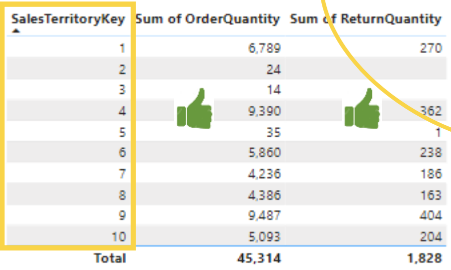
With two-way cross-filtering enabled between the **AW_Sales_2017** and **AW_Territories_Lookup** tables, we now see correct values using TerritoryKey from either table

- The filter context for **AW_Sales_2017**[*TerritoryKey*] now passes up to the **AW_Territories_Lookup**, and then down to the **AW_Returns** table
- Note that we still see incorrect values when filtering using *TerritoryKey* from the **AW_Returns** table, since the filter context is isolated to that single table



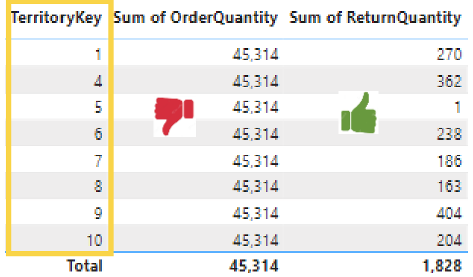
SalesTerritoryKey	Sum of OrderQuantity	Sum of ReturnQuantity
1	6,789	270
2	24	
3	14	
4	9,390	362
5	35	1
6	5,860	238
7	4,236	186
8	4,386	163
9	9,487	404
10	5,093	204
Total	**45,314**	**1,828**

1) Filtering using TerritoryKey from the **AW_Territory_Lookup** table



SalesTerritoryKey	Sum of OrderQuantity	Sum of ReturnQuantity
1	6,789	270
2	24	
3	14	
4	9,390	362
5	35	1
6	5,860	238
7	4,236	186
8	4,386	163
9	9,487	404
10	5,093	204
Total	**45,314**	**1,828**

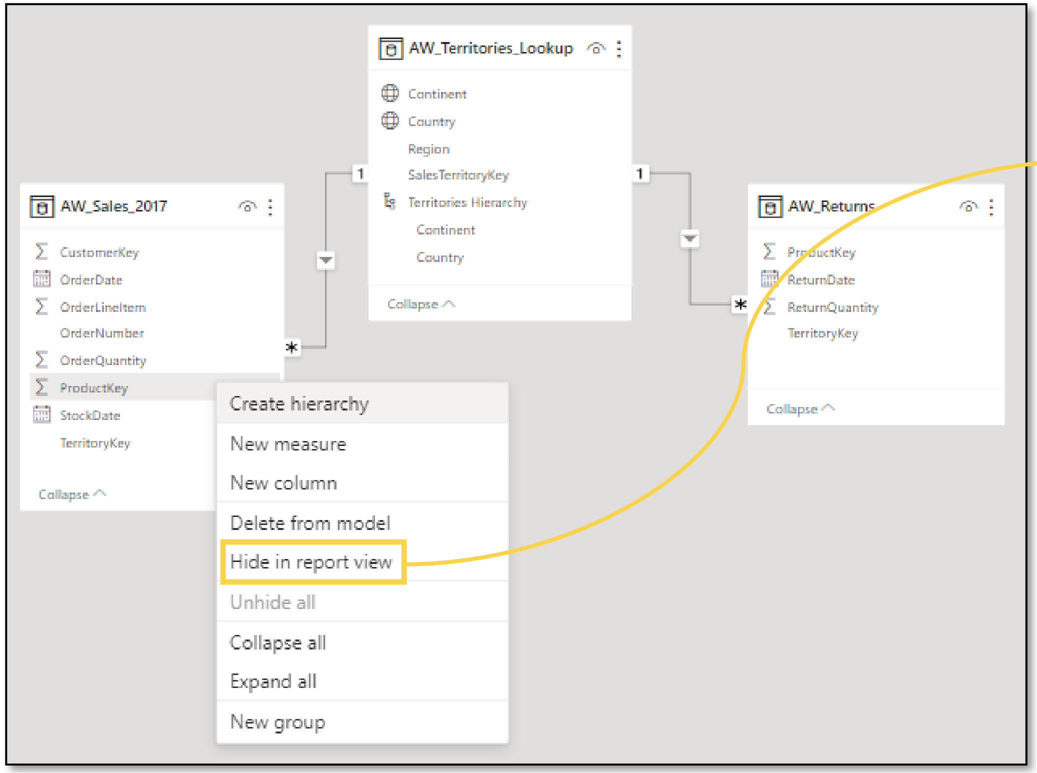
2) Filtering using TerritoryKey from the **AW_Sales_2017** table



TerritoryKey	Sum of OrderQuantity	Sum of ReturnQuantity
1	45,314	270
4	45,314	362
5	45,314	1
6	45,314	238
7	45,314	186
8	45,314	163
9	45,314	404
10	45,314	204
Total	**45,314**	**1,828**

3. Filtering using TerritoryKey from the **AW_Returns** table

Hiding Fields From Report View



Hiding fields from Report View makes them inaccessible from the Report tab (although they can still be accessed within the **Data** or **Relationships** views)

PRO TIP:
Hide the **foreign key columns** in your data tables to force users to filter using the **primary keys** in the lookup tables