Power BI - Data Modelling - p4 - Lecture 12

ASSIGNMENT: TABLE RELATIONSHIPS

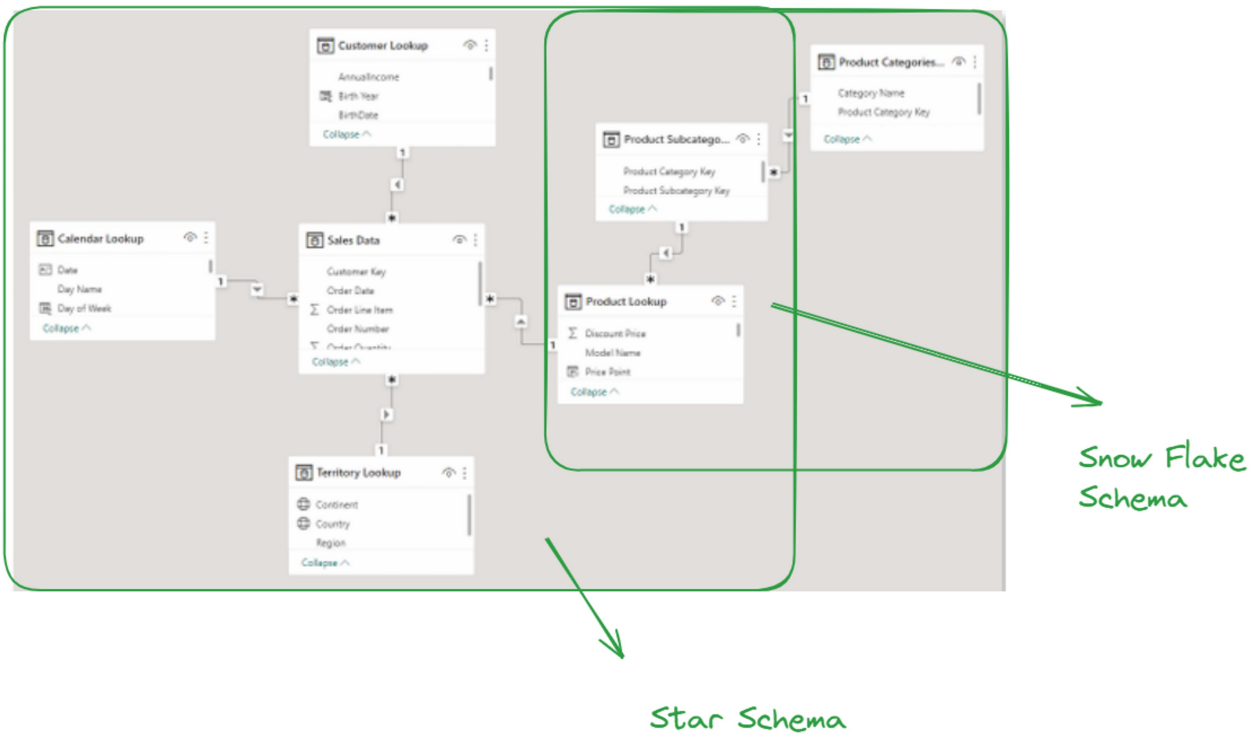
1. Delete all existing table relationships.

2. Create a star schema by creating relationships between the Sales, Calendar, Customer, Product and Territories tables.

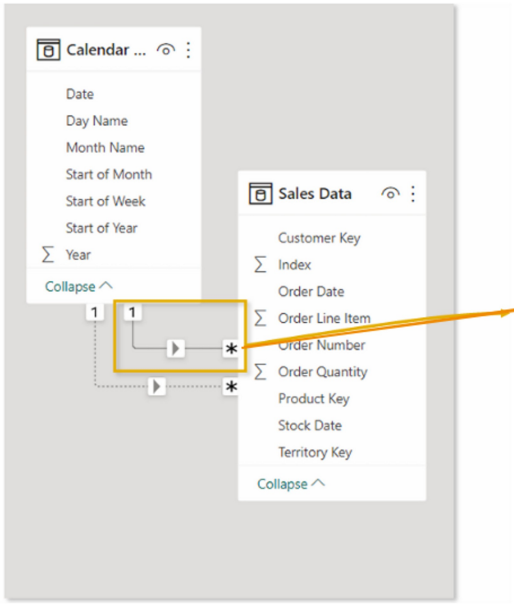
3. Connect all three product tables (Product, Subcategory, Category) in a snowflake schema.

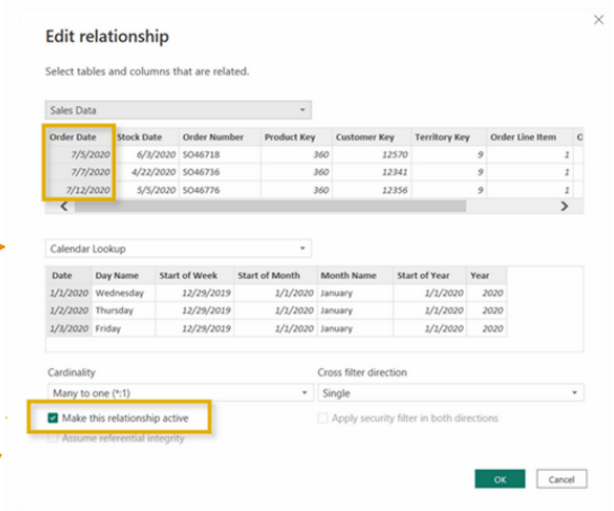
4. Use the matrix visual to confirm that you can filter Order Quantity values using fields from each dimension table.

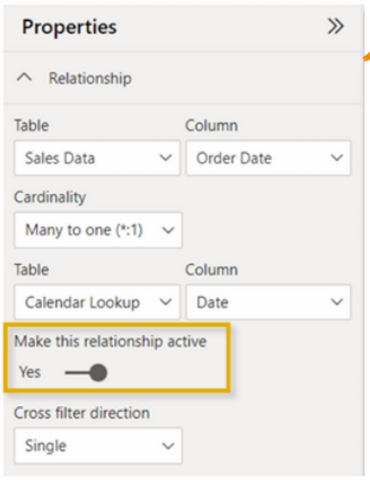
Solution Preview



ACTIVE & INACTIVE RELATIONSHIPS



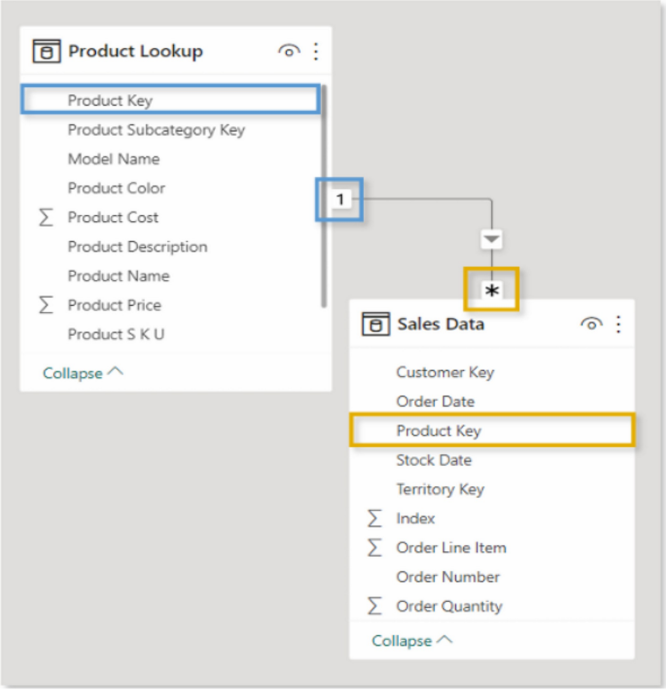




The Sales Data table contains two date fields (Order Date & Stock Date), but there can only be one active relationship to the Date key in the Calendar table.

You can set relationships to active or inactive from either the Edit Relationships dialog box or the Properties (you must deactivate one before activating another)

RELATIONSHIP CARDINALITY



Cardinality refers to the uniqueness of values in a column.

- Ideally, all relationships in the data model should follow a one-to-many cardinality: one instance of each primary key, and many instances of each foreign key

In this example there is only ONE instance of each Product Key in the Product table (noted by a "1"), since each row contains attributes of a single product (name, SKU, description, price, etc.)

There are MANY instances of each Product Key in the Sales table (noted by an asterisk *), since there are multiple sales for each product

ONE-TO-ONE CARDINALITY

Product Lookup

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

Price Lookup

product_id	product_price
4	$3.64
5	$2.19
7	$2.61
8	$2.59

- Connecting the two tables above using product_id creates a one-to-one relationship, since each product ID only appears once in each table

- This isn't necessarily a "bad" relationship, but you can simplify the model by merging the tables into a single, valid dimension table

product_id	product_name	product_sku	product_price
4	Washington Cream Soda	64412155747	$3.64
5	Washington Diet Soda	85561191439	$2.19
7	Washington Diet Cola	20191444754	$2.61
8	Washington Orange Juice	89770532250	$2.59

NOTE: this still respects the rules of normalization, since all rows are unique and capture product-specific attributes

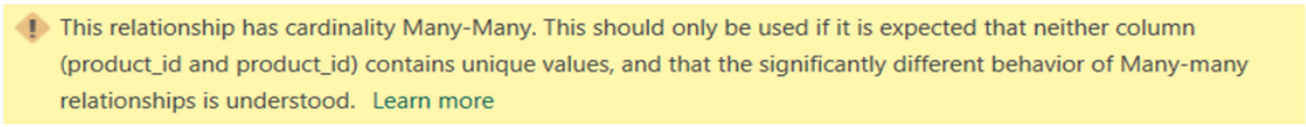
MANY-TO-MANY CARDINALITY

Product Lookup

product_id	product_name	product_sku
4	Washington Cream Soda	64412155747
4	Washington Diet Cream Soda	81727382373
5	Washington Diet Soda	85561191439
7	Washington Diet Cola	20191444754
8	Washington Orange Juice	89770532250

Sales

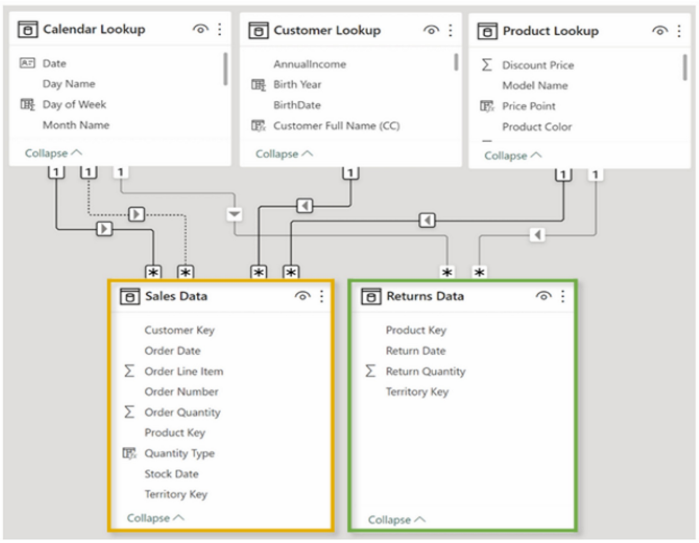
date	product_id	transactions
1/1/2017	4	12
1/2/2017	4	9
1/3/2017	4	11
1/1/2017	5	16
1/2/2017	5	19
1/1/2017	7	11



- If we try to connect the tables above using product_id, we'll get a many-to-many relationship warning since there are multiple instances of product_id in both tables.

- Even if we force this relationship, how would we know which product was actually sold on each date – Cream Soda or Diet Cream Soda?

CONNECTING MULTIPLE FACT TABLES



This model contains two fact tables: Sales Data and Returns Data

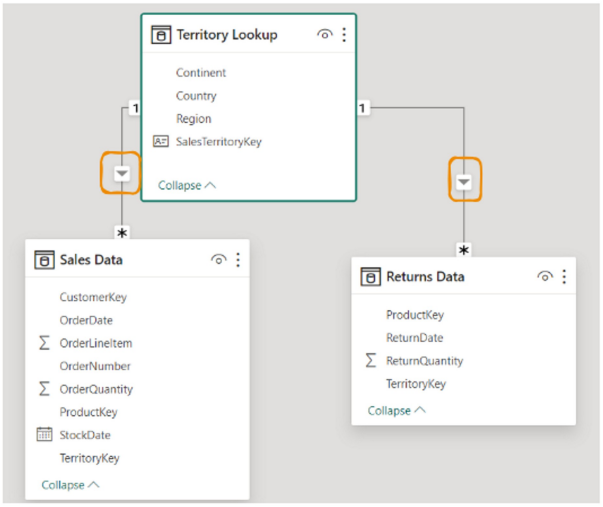
- Since there is no primary/foreign key relationship, we can't connect them directly to each other.

- But we can connect each fact table to related lookups, which allows us to filter both sales and returns data using fields from any shared lookup tables.

- We can view orders and returns by product since both tables relate to Product Lookup, but we can't view returns by customer since no relationship exists.

Generally speaking, fact tables should connect through shared dimension tables, not directly to each other

FILTER CONTEXT & FLOW



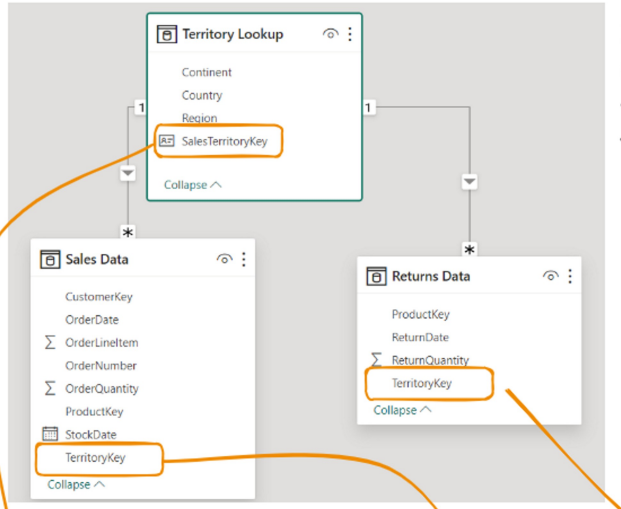
Here we have two data tables (Sales Data and Returns Data), connected to Territory Lookup

The arrows show the filter direction, and point from the one (1) side of the relationship to the many (*) side

- When you filter a table, that filter context is passed to any related "downstream" tables, following the arrow's direction

- Filter context CANNOT flow "upstream"

PRO TIP: Arrange lookup tables above fact tables in your model as a visual reminder that filters always flow downstream



In this model, the only way to filter both Sales and Returns data by Territory is to use the Territory Key from the lookup table, which is upstream and related to both fact tables.

- Filtering using Territory Key from the Sales table yields incorrect Returns values, since the filter context can't flow to any other table

- Filtering using Territory Key from the Returns table yields incorrect Sales values, and is limited to territories that exist in the returns table

TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	[illegible]	[illegible]
6	[illegible]	[illegible]
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

Filtering by Territory Lookup[Territory Key]

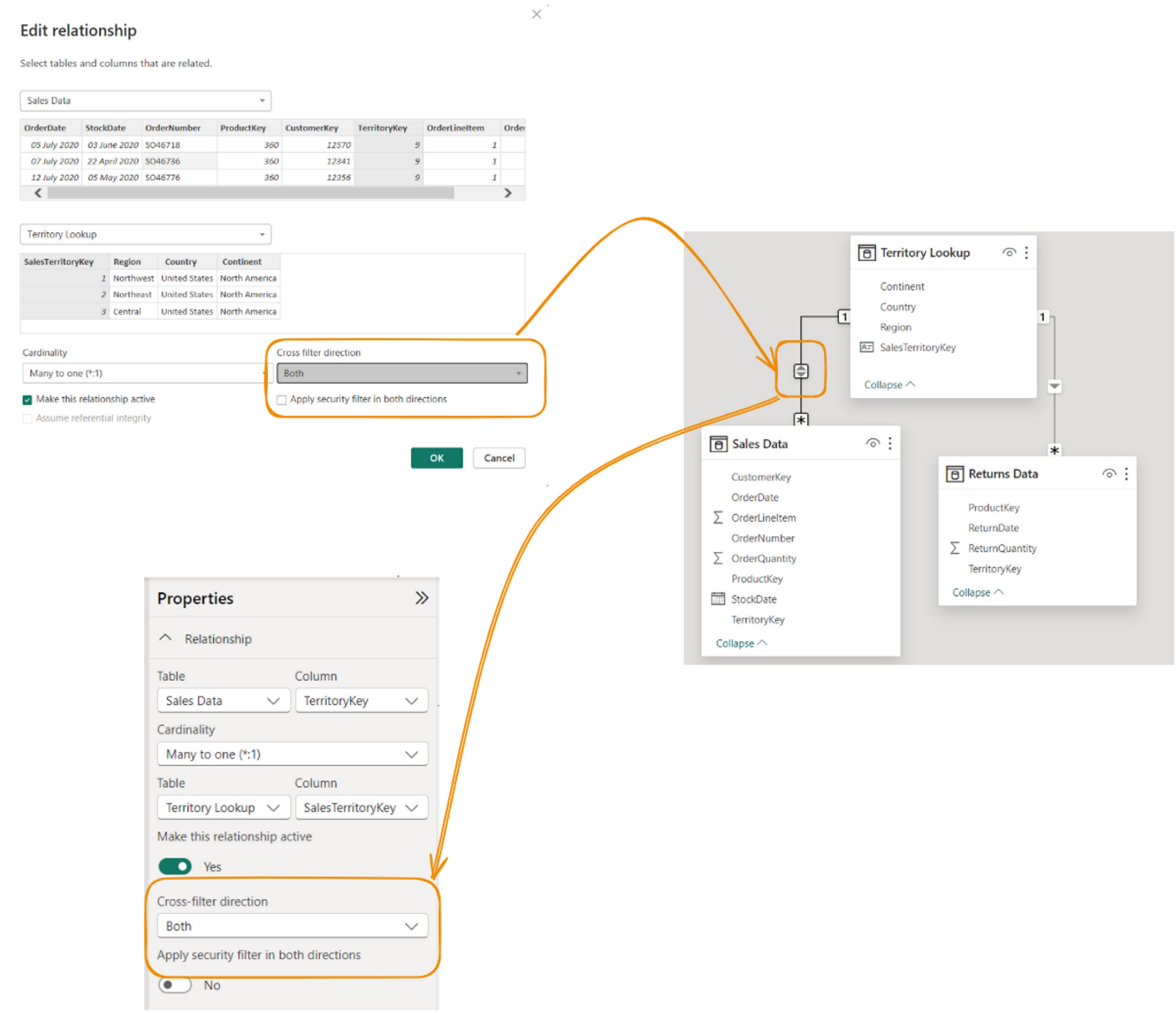
TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	1,828
2	40	1,828
3	30	1,828
4	17,191	1,828
5	[illegible]	[illegible]
6	[illegible]	[illegible]
7	7,862	1,828
8	7,950	1,828
9	17,951	1,828
10	9,694	1,828
Total	**84,174**	**1,828**

Filtering by Sales Data[Territory Key]

TerritoryKey	OrderQuantity	ReturnQuantity
1	84,174	270
4	84,174	362
5	84,174	1
6	[illegible]	[illegible]
7	[illegible]	[illegible]
8	84,174	163
9	84,174	404
10	84,174	204
Total	**84,174**	**1,828**

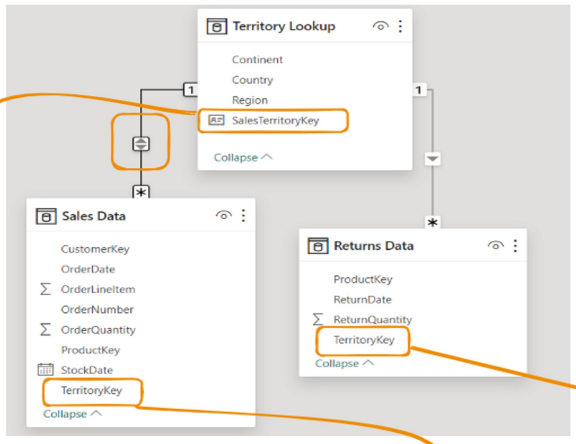
Filtering by Returns Data[Territory Key]

BI-DIRECTIONAL FILTERS



Updating the cross-filter direction from Single to Both allows filter context to flow in either direction

- In this example, filters applied to the Sales table can pass up to the Territory Lookup table, then down to Returns



With two-way cross-filtering enabled between Sales and Territory, we now see correct values using Territory Key from either table

- Filter context can now pass up to the Territory Lookup table, then downstream to Returns

- However, we still see incorrect values when filtering using Territory Key from the Returns table, since the filter context is isolated to that single table

TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	49	1
6	10,894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

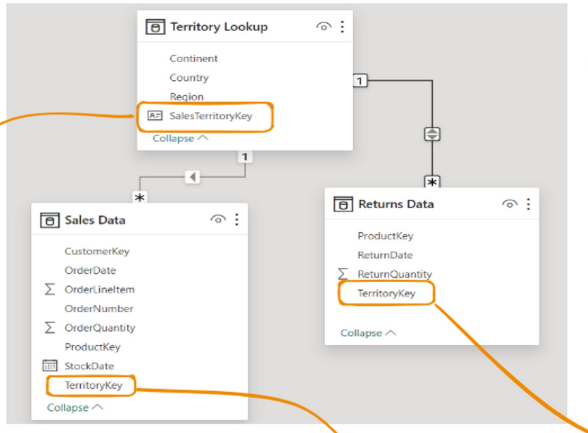
Filtering by Territory Lookup[Territory Key]

TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	49	1
6	10,894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

Filtering by Sales Data[Territory Key]

TerritoryKey	OrderQuantity	ReturnQuantity
1	84,174	270
4	84,174	362
5	84,174	1
6	84,174	238
7	84,174	186
8	84,174	163
9	84,174	404
10	84,174	204
Total	**84,174**	**1,828**

Filtering by Returns Data[Territory Key]



In this case, we've enabled two-way cross-filtering between the Returns and Territory tables

- As expected, we now see incorrect values when filtering using Territory Key from the Sales table, since the filter context is isolated to that single table

- While the values appear to be correct when filtering using Territory Key from the Returns table, we're missing sales data from any territories that didn't appear in the returns table (specifically Territories 2 & 3)

TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
2	40	
3	30	
4	17,191	362
5	49	1
6	10,894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

Filtering by Territory Lookup[Territory Key]

TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	1,828
2	40	1,828
3	30	1,828
4	17,191	1,828
5	49	1,828
6	10,894	1,828
7	7,862	1,828
8	7,950	1,828
9	17,951	1,828
10	9,694	1,828
Total	**84,174**	**1,828**

Filtering by Sales Data[Territory Key]

TerritoryKey	OrderQuantity	ReturnQuantity
1	12,513	270
4	17,191	362
5	49	1
6	10,894	238
7	7,862	186
8	7,950	163
9	17,951	404
10	9,694	204
Total	**84,174**	**1,828**

Filtering by Returns Data[Territory Key]

Territories 2 & 3 don't exist in the Returns table, so they aren't included in the filter context that passes to Territory Lookup and Sales