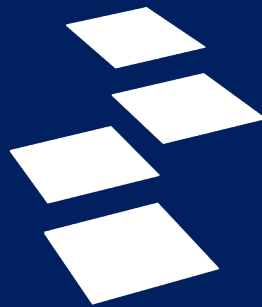


# Fundamentals of Data Modeling

Labs



JOURNEYTEAM

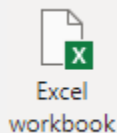
# Guide to the Data Modeling Labs


This workbook contains Labs for the JourneyTEAM Fundamentals of Data Modeling course.

The associated data file used in these Labs can be downloaded at [BusinessIntelligence/Example Tables - v2.xlsx at master · JourneyTEAM-Public/BusinessIntelligence \(github.com\)](https://github.com/JourneyTEAM-Public/BusinessIntelligence/blob/master/Example%20Tables%20-%20v2.xlsx)

You will need that data in a Power BI Desktop file (.pbix). To do so, follow these steps:

1. Open a new Power BI Desktop file
2. Click *Excel Workbook* in the top ribbon



3. In the new window, navigate to the excel file downloaded for this course
4. Double-click on that file
5. Select the table "FlatTable" (you will see a check mark once selected)  
 FlatTable
6. Click OK

The Labs are meant to be done in order. Skipping one or more Labs may make future Labs impossible to complete. Each lab contains step-by-step instructions. Follow the steps as closely as you can. If something looks off at any point, retrace your steps to see if you misapplied a previous step. If this does solve the problem, please let us know.

Steps are written with the assumption that you are learning as you do the labs (future lab's steps assume you learned the terms used in previous labs/steps), so try to learn while you do the labs (as if you needed more motivation, right?).

Enjoy the course!

## Tables Lab

### Introduction

We will begin by creating a Measures Table then breaking up (Normalizing) a Flat Table into a Fact Table and multiple Dimension Tables. With those normalized tables, we will set the groundwork for a Relational Model that we will use throughout the course. We will use Power Query, Power BI's extract, transform, and load (ETL) tool, to perform the Normalization.

Your data sources may not contain Flat Tables in the future, we are using one in this course to show how Normalizing is done and to explore the differences between Fact and Dimension Tables.

### Create a Measures Table

1. Left-click *Enter Data* in the top ribbon (under Home tab)
2. Double left-click *Column1*
3. Rename column to "Hide Me" (see result below)



Create Table		
	Hide Me	+
1		
+		

4. At the bottom left of the Create Table pop-up window, change the name of the table to "MeasureTable" (see result below)


Name:	MeasureTable
-------	--------------

5. Left-click *OK*

You just created a new table and chose all the data inside of it. Nice. This is a method rarely used within Power BI, because a file sourced outside of Power BI is often easier to update (and have the updates show up after each refresh). However, this is perfect for creating a Measure Table, which will be used to organize Measures.

## Normalize the FlatTable

### Disable the Load of FlatTable

Go to the Power Query Editor by selecting this icon  above *Transform Data* in the top ribbon.

1. In the Queries Pane on the left, left-click the word "FlatTable"  FlatTable

Left-clicking a query in the Queries Pane displays the selected query (in this case a table) in the middle of the screen. In the FlatTable displayed there are several columns that represent dimensions or slices of sales data (e.g., product, supplier, dates, etc.). We will use this table and those columns to create our Dimension Tables.

2. Right-click *FlatTable*  FlatTable
3. Left-click *Enable Load* from the drop-down menu

This ensures that the table will not be pushed to the Data Model when a refresh occurs, but could still be used in other queries (as we will do shortly)

*FlatTable* is now italicized, indicating the table's load has been disabled.



Enable Load will now be unchecked in the drop-down menu.

### Create a Product Dimension Table

1. Right-click the *FlatTable*
2. Left-click *Reference* within the drop-down menu

This will create a new table in the Queries Pane called Flat Table (2)

3. Left-click *Flat Table (2)* in the Queries Pane

The table now on display on the screen (FlatTable (2)) is an exact replica of the FlatTable; however, this table is enabled and will be pushed to the Data Model once you apply the changes in Power Query.

4. While holding Control on your keyboard, left-click the headers of the columns *product* and *product\_type* to select them (see results below)

$A_C^B$ product	$A_C^B$ product_type	$A_C^B$ supplier	$A_C^B$ country	$A_C^B$ sales_person
Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
4 distinct, 0 unique	3 distinct, 0 unique	3 distinct, 0 unique	3 distinct, 0 unique	5 distinct, 1 unique
Basketball Jersey	Clothing	Adidas	China	Jessica J.
Basketball Jersey	Clothing	Nike	USA	Jon H.
Basketball Jersey	Clothing	Under Armour	Taiwan	Jose K.
Soccer Ball	Ball	Adidas	China	Jaina Q.
Soccer Ball	Ball	Nike	USA	Jessica J.

- While both columns are selected, right-click the header of the column *product*
- Left-click *Remove Other Columns* from the drop-down menu (see results below)

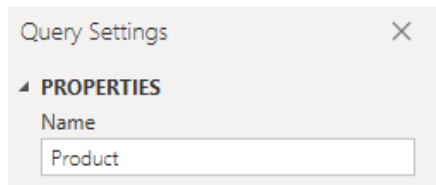
$A_C^B$ product	$A_C^B$ product_type
Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
4 distinct, 0 unique	3 distinct, 0 unique
Basketball Jersey	Clothing
Basketball Jersey	Clothing
Basketball Jersey	Clothing
Soccer Ball	Ball
Soccer Ball	Ball

- Select the column *product*
- Right-click the header of the column *product*
- Left-click *Remove Duplicates* from the drop-down menu (see results below)

$A_C^B$ product	$A_C^B$ product_type
Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
4 distinct, 4 unique	3 distinct, 2 unique
Basketball Jersey	Clothing
Soccer Ball	Ball
Football Helmet	Equipment
Basketball Ball	Ball

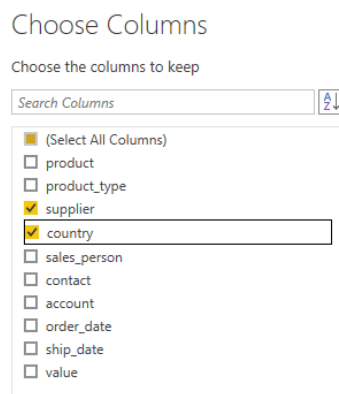
product\_type is a categorization of product; therefore, product is the lowest granularity we need between the two columns. It is important to understand the required granularity of a Dimension Table so that we can ensure we capture all the necessary combinations of values in the columns of the table. More on this later.

- In the Query Settings Pane found on the far right of the page, change the Name field to "Product" (see results below)



### Create a Supplier Dimension Table

1. Repeat Steps 1-3 of **Create a Product Dimension Table** above
2. Left-click the icon above *Choose Columns* on the top ribbon (under Home tab)
3. In the drop-down menu, left-click (*Select All Columns*) at the top of the menu
4. Left-click *supplier* and *country* (see results below)



5. Left-click *OK*

That was another method to Remove Other Columns, a very useful function in Power Query. This method also leaves just the columns selected.

6. While holding control, select the *supplier* and *country* columns
7. Left-click *Remove Rows* in the top ribbon (under the Home tab)
8. Left-click *Remove Duplicates* from the drop-down menu (see results below)



supplier	country
Adidas	China
Nike	USA
Under Armour	Tiawan
Under Armour	USA

This was another method for removing duplicate rows. In this case, because we had multiple columns selected when removing duplicates, all the variations of the values between the selected columns were kept—all rows with repeat combinations of the selected columns were removed.

This is important here because we had suppliers located in more than one country. Removing duplicates on just supplier would have eliminated combinations of supplier and country. Therefore, the granularity required here was the unique combinations of suppliers and countries. Understanding granularity of tables is essential for normalization (and data modeling generally)

9. Double left-click the word FlatTable (2) in the Queries Pane
10. Change the name of the table to "Supplier"

## Create a SalesPerson Dimension Table

1. Repeat Steps 1-3 of **Create a Product Dimension Table** above
2. Using whatever method you prefer, select the *sales\_person* column and remove the other columns
3. Using whatever method you prefer, select the *sales\_person* column and remove duplicate rows
4. Change the table name to "SalesPerson" (see final result below)

sales_person
Jessica J.
Jon H.
Jose K.
Jaina Q.
Joe K.

### Create an Account Dimension Table

1. Repeat Steps 1-3 of **Create a Product Dimension Table** above
2. Select the *contact*, *store*, and *account* columns and remove the other columns
3. Select those three columns again and remove duplicate rows
4. Change the table name to Account (see final result below)

A <sub>C</sub> contact	A <sub>C</sub> store_no	A <sub>C</sub> account
<div> <div>Valid 100%</div> <div>Error 0%</div> <div>Empty 0%</div> </div> <div>13 distinct, 3 unique</div>	<div> <div>Valid 100%</div> <div>Error 0%</div> <div>Empty 0%</div> </div> <div>36 distinct, 33 unique</div>	<div> <div>Valid 100%</div> <div>Error 0%</div> <div>Empty 0%</div> </div> <div>5 distinct, 0 unique</div>
Quinn K.	DSG 1	Dick's Sporting Goods
Pierre M.	DSG 101	Dick's Sporting Goods
Limon L.	Sears 201	Sears
Larry I.	T 490	Target
Sim S.	T 95	Target

### Create a Sales Fact Table

1. Repeat Steps 1-2 of **Create a Product Dimension Table** above
2. Change the name of the table to "Sales"

Fact Tables are typically at the granularity of transactions or events. In this case, each record (or row) represents the individual sales of products (this is often called the Sales Line level). Because the FlatTable already represents this granularity, no granularity changes are required. As you would expect, this means the Fact Table is going to have a lot more rows than Dimension Tables, which is almost always the case.

No column removal is necessary yet because each column may be required for building Surrogate Keys, something we will be doing in the next lab. Fact Tables should have as few columns as possible—but this will have to wait for future labs.



## Column (Fields) Lab

### Introduction

Columns, or sometimes called Fields, are the lifeblood and foundation, of Power BI. Every table, relationship, and visual are built using columns. Without columns, there is no Power BI (well, no useful Power BI). Understanding columns, how they are created, and how they are used is crucial for successful Data Model development.

In this lab you will:


1. Create Surrogate Keys; columns that will be used for Relationships between tables
2. Create new columns that can be used in the model to slice your data in different ways
3. Change column Data Types

### Create Surrogate Keys

Surrogate Keys are columns created during the transformation process or after being pulled from the source system. They are used for three major reasons: (1) Increasing performance speed, (2) managing granularity, and (3) reducing the size of the model. Each of those principles will be explored more later.

You will now create multiple Surrogate Keys on the Dimension Tables then merge them onto the Fact Table—so they can be used in relationships between the tables in the next Lab.

#### *Create Product, Supplier, SalesPerson, and Account Surrogate Keys*

1. Left-click *Product* in the Queries Pane
2. Left-click the *Add Column* tab at top of the top ribbon
3. Left-click *Index Column* in the top ribbon  *Index Column* (not the drop-down arrow)


You just created a column called *Index* which provides a unique number for each row in the table. On a Dimension Table, like this is, each row will have a unique (or non-repeating) Index number because each row represents each unique value within the selected granularity (products). Surrogate Keys always originate in Dimension Tables.

4. Double left-click the *Index* column header and rename to "Product ID"
5. Create Surrogate Keys on the *Supplier*, *SalesPerson*, and *Account* tables (using Steps 1-4 above as your guide)

#### *Add Keys to Fact Table*

1. Left-click *Sales* in the Queries Pane



- Left-click the *Home* tab on top of the top ribbon
- Left-click *Merge Queries* in the top ribbon  **Merge Queries** (not the drop-down arrow)
- Left-click the empty drop-down menu in the middle-left of the pop-up window
- Left-click *Supplier*
- While holding Control on your keyboard, left-click the headers of the columns *supplier* then *country* in the top table of the pop-up window (see result below)

supplier 1	country 2
Adidas	China
Nike	USA
Under Armour	Tiawan

Power BI Data Models only allows relationships between one column of a table to one column of another table. Power Query allows relationships via multiple columns between tables. Surrogate Keys allow us to make a single column key between tables while using multiple columns, which is what you're doing here. By holding Control while selecting columns, you have chosen, in this case, two columns to connect the tables by. This is essential because connecting via just *country* or just *supplier* would miss at least one unique combination of *supplier* and *country*.

- While holding Control on your keyboard, left-click the headers of the columns *supplier* then *country* in bottom table of the pop-up window (see result below)

Sales

product	product_type	supplier 1	country 2	sales_person	contact	store_no	account
Basketball Jersey	Clothing	Adidas	China	Jessica J.	Quinn K.	DSG 1	Dick's Sporting Ge
Basketball Jersey	Clothing	Nike	USA	Jon H.	Pierre M.	DSG 101	Dick's Sporting Ge
Basketball Jersey	Clothing	Under Armour	Tiawan	Jose K.	Limon L.	Sears 201	Sears
Soccer Ball	Ball	Adidas	China	Jaina Q.	Larry I.	T 490	Target

Supplier

supplier 1	country 2	Supplier ID
Adidas	China	0
Nike	USA	1
Under Armour	Tiawan	2
Under Armour	USA	3
Adidas	Mexico	4

- Left-click OK

Merging creates a column on the far right of the originating table (Sales). If you scroll to the right within the table, you will see a column called *Supplier.1*. Merging connects two tables together using the keys we identified in the pop-up window. Now that the tables are connected, we can pull data from the connected table (Supplier) to the base table (Sales).

9. Left click the Expand icon at the right of the *Supplier.1* column header



10. Within the pop-up window, de-select all by left-clicking *Select All Columns*
11. Left-click *Supplier ID*
12. Left-click *OK*

You just pulled the Surrogate Key we created on the Supplier Dimension Table onto the Sales Table. In the next Lab, you will use these columns to create a relationship between the two tables.

13. Pull the Surrogate Keys from *Product*, *SalesPerson*, and *Account* onto *Sales* (using steps 1-12 as your guide).

### Create New Columns

Filtering and slicing in Power BI is done using columns. You can create new slicers and filters for your reports by creating new or calculated columns on tables. You will now create a Sales Team column and a Sales Region column you will use, in a future Lab, to slice the data by Sales Teams or Sales Region.

With few exceptions, calculated columns like these are only created when new filter or slicer columns are needed within a report. Calculations for creating new numbers (e.g., Sales Amount, ratios, etc.) should almost always be done using measures instead. New columns create data that must be stored within the Data Model, which uses precious memory. This will be explained in more depth later.

#### Create Sales Team Column

1. Left-click *SalesPerson* in the Queries Pane
2. If not already selected, left-click the tab *Add Column* on the top ribbon
3. Left-click *Conditional Column* in the top ribbon
4. In the ensuing pop-up window, make the following changes:

**Add Conditional Column**



Add a conditional column that is computed from the other columns or values.

New column name  
Sales Team

	Column Name	Operator	Value	Output
If	sales_person	equals	ABC 123 Jessica J.	ABC 123 Jessica
Else If	sales_person	equals	ABC 123 Jon H.	ABC 123 Jessica

Add Clause

Else  
ABC 123 Jaina

With this column we signal which sales people belong to which sales team (Jessica's or Jaina's). These teams are of course subject to change. To change it in the future, left-click the gear to the right of the step this creates. **Added Conditional Column**  

With few exceptions, Calculated Columns like these are only created when new filter or slicer columns are needed within a report. Calculations for creating new numbers (e.g., Sales Amount, ratios, etc.) should almost always be done using measures instead. This will be explained in more depth later in the course.

5. Left-click OK

### Create Sales Region Column

1. Follow *Create Sales Team Column* Steps 1-3
2. In the ensuing pop-up window, make the following changes:

**Add Conditional Column**

Add a conditional column that is computed from the other columns or values.

New column name  
Sales Region

	Column Name	Operator	Value	Output
If	Sales Team	equals	ABC 123 Jessica	ABC 123 North

Add Clause

Else  
ABC 123 South

3. Left-click OK

These calculated columns can now be used as filters or slicers in your report. Huzzah.

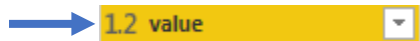
### Change Column Data Types

The Data Type selected for a column determines how it is displayed and stored within the Data Model. You should select a Data Type for all columns within Power Query before pressing Close & Apply. And you should select Data Types that minimize the data required for storage and capture all the data needed to be displayed in the report. For example, if you do not need decimals, use Whole Number as the Data Type (this will automatically round the decimals to the nearest whole number). If your column contains text, it is going to be the Text Data Type, and that is okay (you cannot make text a number by changing the type)...it will just take up more space. This is one reason why we normalize tables—pushing filter/slicer columns that are usually Text to Dimension Tables where there are far fewer rows than Fact Tables.

Here are two basic principles to consider: (1) each character of a value must be stored (e.g., 100 is bigger than 10.00001 from a math perspective, but much smaller from a storage perspective (3 vs. 8 characters)); the fewer characters the better from a storage perspective; (2) text characters require more storage than numbers; Power BI stores data in numbers, to store text, Power BI essentially creates a dictionary for itself, which also must be stored. This is one reason we use Surrogate Keys rather than the Text fields they replace. These concepts will be explained in greater detail later.


### Change value Data Type

1. Left-click *Sales* in the Queries Pane
2. Left-click the 1.2 icon on the left side of the *value* column header
3. Left-click *Fixed Decimal Number* from the drop-down menu




You have now rounded the decimal numbers in the *values* column to the nearest hundredth. The extra decimal numbers beyond hundredths are usually more granular than required for reporting. When that's the case, use the Fixed Decimal Number Data Type to minimize characters and, therefore, the memory used.

### Change *order\_date* Data Type

1. Left-click the Date icon at the far left of the *order\_date* column header 
2. Left-click *Date/Time* within the drop-down menu

Notice how the switch created a significant number of characters on each of the values within the column. Also note how repetitive these added characters are (all are 12:00:00 AM); this is because you have forced more granularity on the values when the granularity did not previously exist—so Power BI makes it up. Because this is so repetitive and totally made up, there is no value to the addition. Even if the the time aspect was unique to each value, you may consider removing it because of the amount of storage the time element requires. If time is required, this column should be divided into two columns: Date and Time to reduce cardinality.

In this case, the data is uselessly repetitive, so we will remove the time element of the column.

3. Left-click the Date/Time icon at the left side of the *order\_date* column header 
4. Left-click *Date* from the drop-down


## Relationships Lab

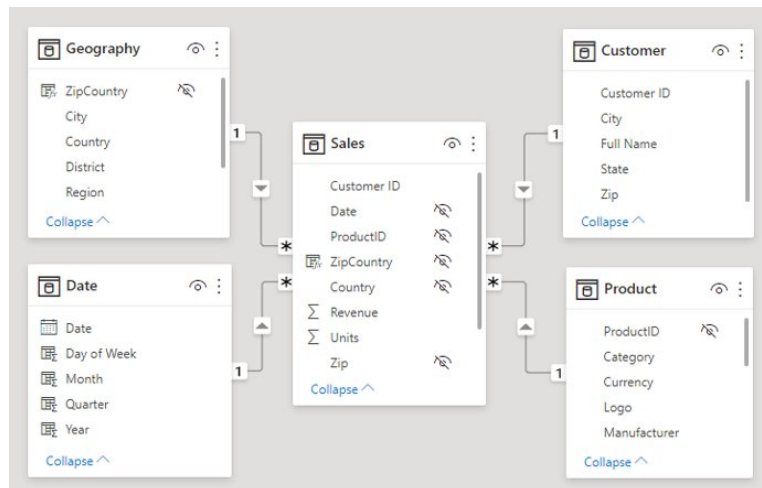
### Introduction

Relationships and relational models are one of the superpowers of Power BI. Relationships not only allow tables to talk to one another, they allow users to significantly reduce the amount of data required to model data, which provides significant performance advantages.

Behind the scenes and on the fly, Power BI uses relationships to tack tables on to one another. This would essentially be a JOIN in SQL or a Merge in Power Query. All the work you did to normalize the FlatTable in the first Lab is, in essence, reversed behind the scenes; all the related columns that we sent to new tables are brought back onto a single, unseen table. In some circles this process is called Table Extension. Table Extension, which is accomplished via Relationships, is the process that allows one table to interact with or filter another.

You may be thinking: "If Power BI denormalizes tables in the background, why in the world did we normalize the FlatTable like we did?" Great question. Microsoft has not been very explicit about the details, but it has definitively stated that using a Star Schema (see image below), or Fact and Dimension Tables, is the most performant approach to Data Modeling in Power BI. This infers that the memory gains (or reduction of data) that come from normalizing tables in this way outweighs the computing costs of denormalizing tables in the background. This concept will be explored in more depth later.

Relationships are managed from the Model View  within Power BI Desktop. In that view you can see the tables that make up the model and the relationships between them (see below). Without relationships between tables, the tables cannot filter one another, but can still be used in a report (as demonstrated in the demo).



There are several types and variations of relationships that can be used. The most performant is a One-Directional, One-to-Many relationship (each of the relationships above are this type of relationship).

In this lab you will create relationships between the tables we have created and explore different types of Relationships available within Power BI.

### Relationship Building

1. Left-click the *Home* tab at the top left of the top ribbon

2. Left-click *Close & Apply*  at the left side of the top ribbon (not the drop-down)

*Close & Apply* closes the Power Query Editor window and applies the changes you had made there to the Data Model in Power BI. The pop-up refresh screen you will now see captures this process, which includes: (1) pulling data from the sources, (2) running the transformations on that data, (3) compressing the data, and (4) pulling the data (importing the data) into Power BI.

3. Once the refresh is complete, left-click the Model View icon  on the far left pane

Power BI's AI made some relationships for us. If you hover over a relationship, or line, the keys used for the relationship will be highlighted on each table. As you may have noted, the AI did not do a perfect job, although it wasn't too far off. We want to use the Surrogate Key columns we previously created as our keys. Relationships can be created by dragging a field one one table to a field on another table. Because we already have established Relationships, you will use another method.





4. Left-click *Manage Relationships* in the top ribbon
5. Left-click a Relationship where the two associated fields are not one of the ID columns we had previously created
6. Left-click *Edit*
7. Left-click the associated ID columns in both the top and the bottom tables within the pop-up window
8. Left-click *OK*
9. Repeat Steps 5-8 for each of the Relationships that are not using the Surrogate Keys we previously created (see result below)

Manage relationships		
Active	From: Table (Column)	To: Table (Column)
<input checked="" type="checkbox"/>	Sales (Account ID)	Account (Account ID)
<input checked="" type="checkbox"/>	Sales (Product ID)	Product (Product ID)
<input checked="" type="checkbox"/>	Sales (Sales Person ID)	SalesPerson (Sales Person ID)
<input checked="" type="checkbox"/>	Sales (Supplier ID)	Supplier (Supplier ID)

10. Left-click *Close*

As the AI so kindly pointed out for us, Surrogate Keys are not required for making relationships. Any two columns on any two tables that have matching values can be used to make an effective relationship between tables (no matter the Data Type, though the Data Type must match). That means we could have, for example, used the sales\_person column on the Sales Person and Sales tables to create a relationship between them; however, we have found that when working with more data relationships that use columns with the Text Data Type can slow visual refreshes and the development process. Surrogate Keys usually also take much less space in the model, which also helps performance.

### OPTIONAL: Understand Relationships

There are several variations of Relationships available in Power BI. Variations fall into a few main concepts: (1) Key Cardinality (2) Filter Direction, and (3) State.

**Key Cardinality.** There are two sides to every relationship and each side can only be designated as either Many or One. This designation is determined by the uniqueness of the values within the key column used in the relationship. In the Model View, the uniqueness of the keys involved in a relationship are represented by either a "1" or a star "\*" next to the

associated table. The “1” indicates that each value within that table’s key column (the column used in the relationship) is unique (i.e., only show up on one row on the table), and a star “\*” indicates that one or more values within the column are repeated (i.e., show up in more than one row on the table).

Dimension Tables should almost always be on the One side of a relationship. This is because, as we ensured in an earlier lab, each row in a dimension represents a unique thing within the context of the dimension (e.g., In your Product Dimension Table, each row represents a unique product. And in your Supplier Dimension Table, each row represents a unique combination of supplier and country. Each of the things only show up on one row—because they are unique).

Fact Tables, on the other hand, will almost always be on the Many side of relationships. This is because Fact Tables do not typically represent unique dimensions, rather they represent unique transactions. And transactions usually contain several dimensional attributes or keys that repeat from one transaction to another (e.g., Each row in your Sales Fact Table represent a sale, which contains the sales person, product, account, etc. associated with that transaction. It’s likely that a sales person, a product, or an account will be associated with more than one sale, meaning they will show up on multiple rows). Because the values within the key columns on the Sales table are not unique (show up on more than one row), this side of the relationship is declared as Many.


**Filter Direction.** The direction of a Relationship determines which table is tacked on to the other (e.g., Table 1 tacked onto Table 2, or visa-versa, or both). The process is commonly known as Table Extension (see the Introduction to this Lab above). A table cannot filter another table unless there is an active relationship that allows it to (or a Measure is used to work around that). In the Model View, the direction of the filter is shown by an arrow found on the relationship’s line.

By default, most relationships filter from the One side to the Many side (usually Dimension Table to Fact Table). This is largely because how Relational Models are built (Dimension Tables are built to filter Fact Tables), but also because performing a join or a merge of a table on the many-side to the table on the one-side can completely change the size of the table and complexity of the table by not just adding columns, but rows, as well (the merged table will now have as many rows as the table on the many side had). However, Bi-Directional Relationships are available in Power BI, but they should be avoided.

**State.** Relationships can be either Active or Inactive. In the Model View, Active relationships are represented by solid lines and Inactive relationships are represented by dotted lines.

Active relationships determine the filtering a user experiences when using slicers, visuals, and filters in the Report View. Inactive relationships do not influence filtering *unless* activated within a Measure. Less performant/Star Schema-violating relationships should be set as Inactive whenever possible and activated within a Measure so they are only used when required.

### Key Cardinality


1. Left-click on the Model View icon  found on the far left pane
2. Right-click the Relationship line between the Sales and Product tables
3. In the drop-down, left-click *Properties*
4. In the pop-up window, left-click *product\_type* within the Product table (see example below)

Product		
Product ID	product	product_type
0	Basketball Jersey	Clothing
1	Soccer Ball	Ball
2	Football Helmet	Equipment


5. Left-click *OK*

You just changed to key that will be used on the Product table for the relationship. More importantly, for the purpose of this lab, you just changed the Cardinality of the key: it is now Many rather than One. This means the selected key, *product\_type*, has at least one value that appears in more than one row. You will explore this shortly.


Note that this key is an entirely different Data Type than the key used on the Sales table (Text vs. Number). And there are no matching values between the two columns. For a Relationship to work, the types of the two columns must align (e.g., Text to Text, Number to Number) and there must be matching values. However, Power BI will still allow that Relationship to exist—though it will not work.

6. Left-click the Data View icon  found on the far left pane.
7. Left-click the *Product* table in the Fields Pane on the far right.
8. Observe the *product\_type* column and try to find the value that appears on more than one row (Hint: "Ball")

Just that one repeating value turned this column into a Many key. That's all it takes, no matter the amount of rows on the table. Let's take a look at how poor relationships like this one can affect visuals.

9. Left-click the Report View  found on the far left pane
10. Left-click *Relational Model Test* page tab found at the bottom left

The table in the middle blew up. Well, it's still there, but it doesn't work anymore. This is because the table is attempting to use the Relationship we just changed to be useless (different Data Types and no matching values). So even though Power BI allows you to create bad relationships, there are consequences. Now let's fix this fella.


11. Left-click on the Model View icon  found on the far left pane
12. Right-click the Relationship line between the Sales and Product tables
13. In the drop-down, left-click *Properties*
14. In the pop-up window, left-click *Product ID* within the Product table

And with that, the Cardinality of the key has been restored to One. Note that within this column, *Product ID*, there are no repeating values. That is why it's Cardinality is One—just the way we want it.

Note: Many-to-Many Relationships can be quite problematic from a performance perspective and should be avoided. There are several methods for addressing these Relationships. We've already implemented one of the best: Surrogate Keys. The others will not be addressed directly in this course. (These other options will be addressed in future courses. Till then, see the [Microsoft Documentation](#) if other options are required

15. Left-click OK


### Filter Direction

1. Left-click the Report View  found on the far left pane
2. Left-click the *Relational Model Test* page tab at the bottom left of the screen
3. Left-click *Kroger* in the account\_contact slicer at the top left of the canvas (area where visuals can be placed)
4. Observe the values within the column *supplier* in the table in the middle of the canvas, only Adidas and Under Armour appear there (see example below)


account	contact	store_no	product	product_type	sales_person	supplier	country	order_date	value
Kroger	Kolby G.	K 2100	Basketball Ball	Ball	Jon H.	Adidas	China	1/21/2019	\$9.68
Kroger	Kolby G.	K 2100	Soccer Ball	Ball	Jaina Q.	Adidas	China	2/10/2019	\$9.81
Kroger	Kolby G.	K 432	Soccer Ball	Ball	Jon H.	Under Armour	England	3/2/2019	\$9.82
Kroger	Kolby G.	K 89	Soccer Ball	Ball	Jon H.	Under Armour	USA	3/2/2019	\$9.81
<b>Total</b>									<b>\$39.12</b>

- Now observe *supplier\_country* slicer to the top right of the table; Adidas, Nike, and Under Armour appear there

This may seem unremarkable. But the slicer is not being filtered by the results of the table, hence why the results of the table do not affect it. This is because the table that provides the fields populating the slicer (Table: Supplier; Fields: supplier and country) filters the Sales table and not the other way around. Let's look in the Model View.

- Left-click on the Model View icon  found on the far left pane
- Observe the relationship between Supplier and Sales (the arrow points from the Supplier table to the Sales table)

Now let's see what happens if we make this a Bi-Directional Relationship.

- Right-click the Relationship line between the Supplier and Sales tables
- Left-click *Properties* from the drop-down menu
- Near the bottom right of the pop-up window, under Cross filter direction, select *Both* from the drop-down menu
- Left-click OK
- Left-click the Report View 
- Left-click the *Relational Model Test* page tab at the bottom left of the screen
- Ensure *Kroger* is selected (and the only selection) in the account, store\_no, contact slicer at the top left
- Observe that the top right slicer is now being filtered by the results of the table: Nike is no longer available in the slicer

Now let's jump back to the model view to see what's happening behind the scenes.

- Left-click on the Model View icon 


Note that the slicer where we selected Kroger is coming from the Account table, which, according to the arrow on the Relationship, is filtering the Sales table (as we clearly observed earlier in the Report View). Now, we can also see that Sales is now filtering Supplier, the table that is providing us the supplier slicer in the Report View (the arrows on the relationship are now pointing both directions). This means that whatever filters the Sales table, will now filter through the Sales table onto the Supplier table, as we observed in the Report View. Or in other words, by providing a filter from

Account, you are filtering the Sales table, which then filters the Supplier table. Yeah, relationships can get ugly fast.


Although this is a cool functionality, Bi-Directional Filters are not performant and should be avoided. This is because Bi-Directional Filters are essentially merging both tables twice, once each direction, and then looking for matches as filters are passed. And merges from a many-side to a one-side can really expand not just the columns of the table, but also the rows (a Fact table merged onto a Dimension table will make the Dimension table have as many rows as the Fact table). Avoid that if you can, as it may cause performance issues either now or after your data has grown. Time to fix it.

17. Right-click the Relationship line between the Supplier and Sales tables
18. Left-click *Properties* from the drop-down menu
19. Near the bottom right of the pop-up window, under Cross filter direction, select *Single* from the drop-down menu
20. Left-click *OK*

### State

1. Left-click on the Model View icon 
2. Right-click the Relationship line between the Account and Sales tables
3. Left-click *Properties* from the drop-down menu
4. Near the bottom left of the pop-up window, left-click (uncheck) *Make this relationship active*
5. Left-click *OK*

You just made this relationship Inactive. Now let's go see the impact.


6. Left-click the Report View 
7. Left-click the *Relational Model Test* page tab

You broke it! Okay, I made you do it. The table broke because the table was utilizing the relationship we disabled to bring Account columns onto the Sales table (as observed previously). Without the relationship, and the fields remaining on the table in the middle of the canvas, Power BI is trying, but failing to perform what is being asked of it—thus the failure.

Note: Relationships are a big deal in Power BI. Changing relationships mid-flight will likely break visuals you previously created. That's because field references, which

define visuals and what they show, rely deeply on Relationships. One more reason to get your Data Model set as early as possible in the development process.


Let's take off the fields from Account and do some other exploring.

8. Left-click the main table in the middle of the canvas
9. Left-click the Fields Section Icon  of the Visualizations Pane
10. Remove the fields associated with the Account table by left-clicking the "x" on the right of *account*, *contact*, and *store\_no* at the top of the list of fields pulled into the selected visual
11. Left-click any item in the account, store\_no, contact slicer at the top left of the canvas

Nothing is happening on the main table. That's because the Account table, the origin of the fields in the slicer, are no longer filtering the Sales table. Yeah, Relationships are important.

Without writing specific DAX code in a measure, an Inactive relationship is useless. We will not cover how to write that code in this course, but in a future course.

Let's take this back to how it was.

12. Left-click the main table on the canvas
13. Left-click the arrow next to *Account* in the Fields Pane
14. Left-click *account*, *store\_no*, and *contact* to add them back to the table
15. Left-click on the Model View icon 
16. Right-click the Relationship line between the Account and Sales tables
17. Left-click *Properties* from the drop-down menu
18. Near the bottom left of the pop-up window, left-click (check) *Make this relationship active*
19. Left-click OK

## Measures Lab

### Introduction


Measures play an important role in creating an amazing, scalable Power BI Data Model. The biggest advantage they provide is that, while calculating or creating data that can be seen in visuals, they add a nominal amount of data to your model. In fact, the only data that Measures add to your Data Model is the code of the Measures themselves. And the mere ability to create Measures can help us remove columns from our Data Models be created in Measures rather than columns, which expand the size of our Data Models.

Measures, like columns, can be accessed in the Fields Pane. And, like columns, are found in the Fields Pane under tables. However, unlike columns, they are not truly part of the table they are listed under in the Fields Pane. Measures are table agnostic. You previously created a Measures Table to store and organize our Measures. You will use that in this Lab.

In this Lab you will create a measure, then if desired, you can explore the affects of a Measure on a Data Model.

### Create a Measure

#### Create a Measure



1. Left-click the Report View 
2. Left-click the *Data Model Test* page tab
3. Right-click the word *MeasureTable* in the Fields Pane
4. Left-click *New Measure* in the drop-down menu
5. In the Formula Bar that just opened at the top of the canvas, replace "Measure =" with "Sales Amount = SUM([Sales]value)"
6. Left-click the check mark to the left of the Formula Bar

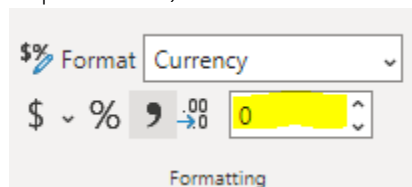
Boom. Measure created. This calculation will sum (or summarize) the values in the values column within the Filter Context (Note: this term will be explored ad nauseum in the DAX courses). In short this means if I drop this measure into a column chart that has Country on the axis, and the sum of values will be done within the filter of each country on the axis, or if I change a slicer on the page, this summary will adjust to that filter.

Now add the measure to a visual and see what happens.

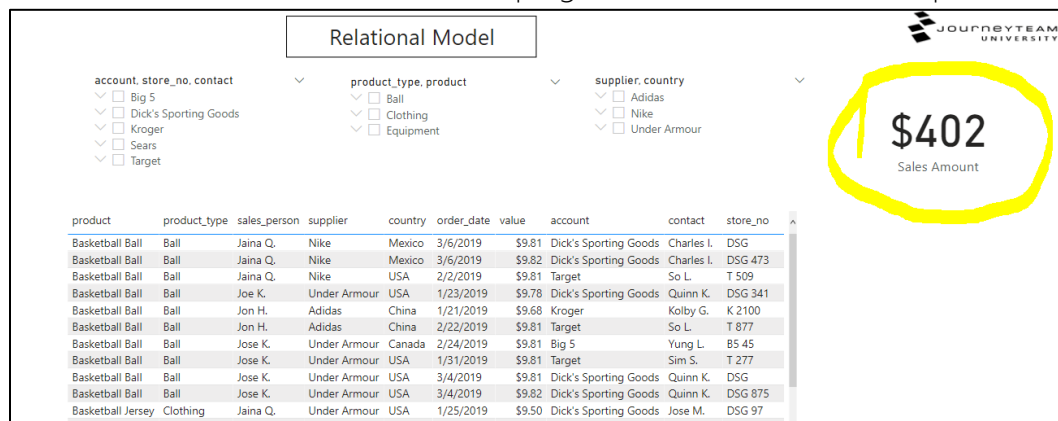


## Test the Measure

1. Left-click the Report View 
2. Left-click the *Data Model Test* page tab
3. Left-click the Card Visual icon  in the Visualization Pane on the right side of the page
4. With the new visual on the canvas still selected, click the box to the left of your new measure, *Sales Amount*, in the Fields Pane
5. Left-click the word *Sales Amount* in the Fields Pane
6. In the Measure Tools tab of the ribbon at the top of the page, adjust the number of decimal places to 0 (see example below)



7. Re-size the visual and move it near the top right of the canvas (see example below)



Note that the Card visual is now displaying the total of the value column in the main table. That means the Measure is working as planned—it is summing up all of the value column's values (that's not confusing, right?). Importantly, the Measure is accomplishing this without adding any data to your Data Model. Let's take a look.

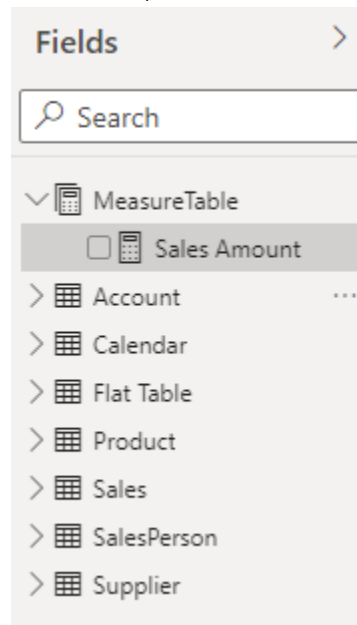
8. Left-click the Data View icon 
9. Left-click the word *Measures Table* in the Fields Pane

Observe that there is no column or values on the MeasuresTable—even though our new Measure is adding that data to our visual. Measures do not add data to Data Models (other than the code written to define them), but they still create values that can be seen in our visualizations. More on this later.

### *Finalize The Measures Table*

Now that you have a Measure created in the Measures Table, that table is ready to become a true Measures Table.

1. Expand the MeasureTable by left-clicking the arrow to the left of the word *MeasureTable* in the Fields Pane
2. Once expanded and the columns or fields of the table are visible beneath it, right-click the word *Hide Me*
3. In the drop-down menu, left-click *Hide* (see result below)



The MeasureTable jumped to the top of the tables! And the icon in front of it changed to indicate this is now an official Measures Table. So, in short, to make a Measures Table, there must be at least one Measure associated with it, and all of the columns on that table must be hidden.

## Performant Lab A

### Introduction

In this Lab you will improve the long-term performance of your report(s) using this Data Model by:

1. Removing unnecessary columns and rows
2. Summarizing data to lowest required granularity, and
3. Maximizing compression by sorting columns

### Removing Unnecessary Columns and Rows

1. Under the Home tab of the top ribbon, left-click the *Transform Data* icon  to take us back to the Power Query Editor
2. In the Power Query Editor, left-click *Sales* in the Query pane on the left side

Thanks to the Surrogate Keys you previously created, you no longer need most of the columns on this table. Using those Keys in Relationships in the Data Model, we effectively bring all of the related tables onto this table through the Table Extension process. Therefore, we can remove the now obsolete columns.

3. While holding Control on your keyboard, left-click *Product ID*, *Account ID*, *Sales Person ID*, *Supplier ID*, *value*, *ship\_date*, and *order\_date*
4. While those columns are still selected, right-click any of the columns header
5. Left-click *Remove Other Columns* from the ensuing drop-down menu

You just removed all the columns you did not have selected. This is a handy way to remove columns you don't need. By doing this you successfully took all of that unnecessary data off of your Data Model!

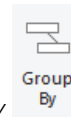
Remember, removing data from your Data Model is one of the three levers you have to affect the speed and longevity of your reports. Removing unnecessary data like this should become second nature for you.

### Summarize Data to Lowest Required Granularity

Assume that all the reports that will use this data will only require sales data at monthly level. This opens the door for some significant data reduction. Let's take our Sales table to the monthly granularity.

1. In the Power Query Editor, left-click *Sales* in the Query pane on the left side

The bottom left of the page signifies this table has 82 rows: **8 COLUMNS, 82 ROWS**. By summarizing sales to the monthly level, you will reduce the number of rows, and therefore the amount of data in the Data Model.



2. Under the Home tab in the top ribbon, left click *Group By*
3. In the ensuing pop-up window, left click *Advanced* and make the following changes:

**Group By**

Specify the columns to group by and one or more outputs.

☐ Basic ☒ **Advanced**

New column name	Operation	Column
<input type="text" value="Value"/>	<input type="text" value="Sum"/>	<input type="text" value="value"/>

4. Left-click *OK*

Notice there are now 41 rows. This method halved the data in the table. Depending on the table, summarizing to a higher granularity like this can provide significant gains from a data minimization perspective. However, we lost the specificity of the exact day the order and the shipments occurred. While this can be a very effective data reduction technique, you will want to be sure you do not need the granularity you sacrifice to get here.

Notice that you used the Surrogate Keys as part of the grouping; this ensured that you retained the dimension-related granularity we previously established. Had you only selected Order Start of Month, you would have reduced the number of rows significantly (one row per month) but would have lost the keys that allow you to slice that data by the relevant dimensions. This is very important to think through when you summarize data in a table.

### Maximize Compression by Sorting Columns

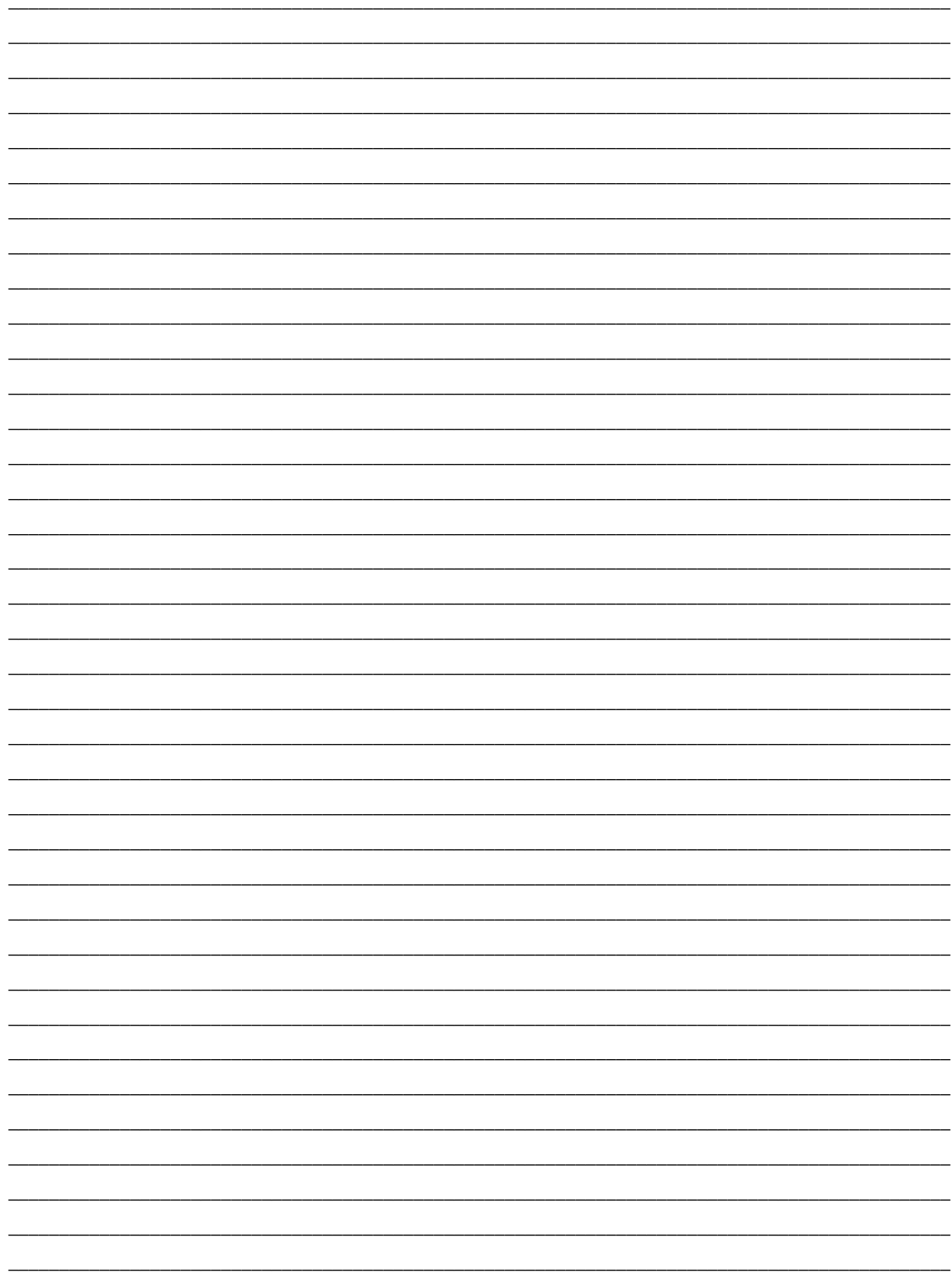
As explained previously, Power BI takes repeating values in each column and compresses them into smaller values to (1) decrease the amount of storage required to maintain the data and (2) potentially speed up the relationships that use that column (I have not received a firm confirmation on this from Microsoft at the time of writing). Next you will implement some sorting to ensure you maximize the compression and gain the two benefits of decreasing the model size and improving the Data Model's infrastructure.

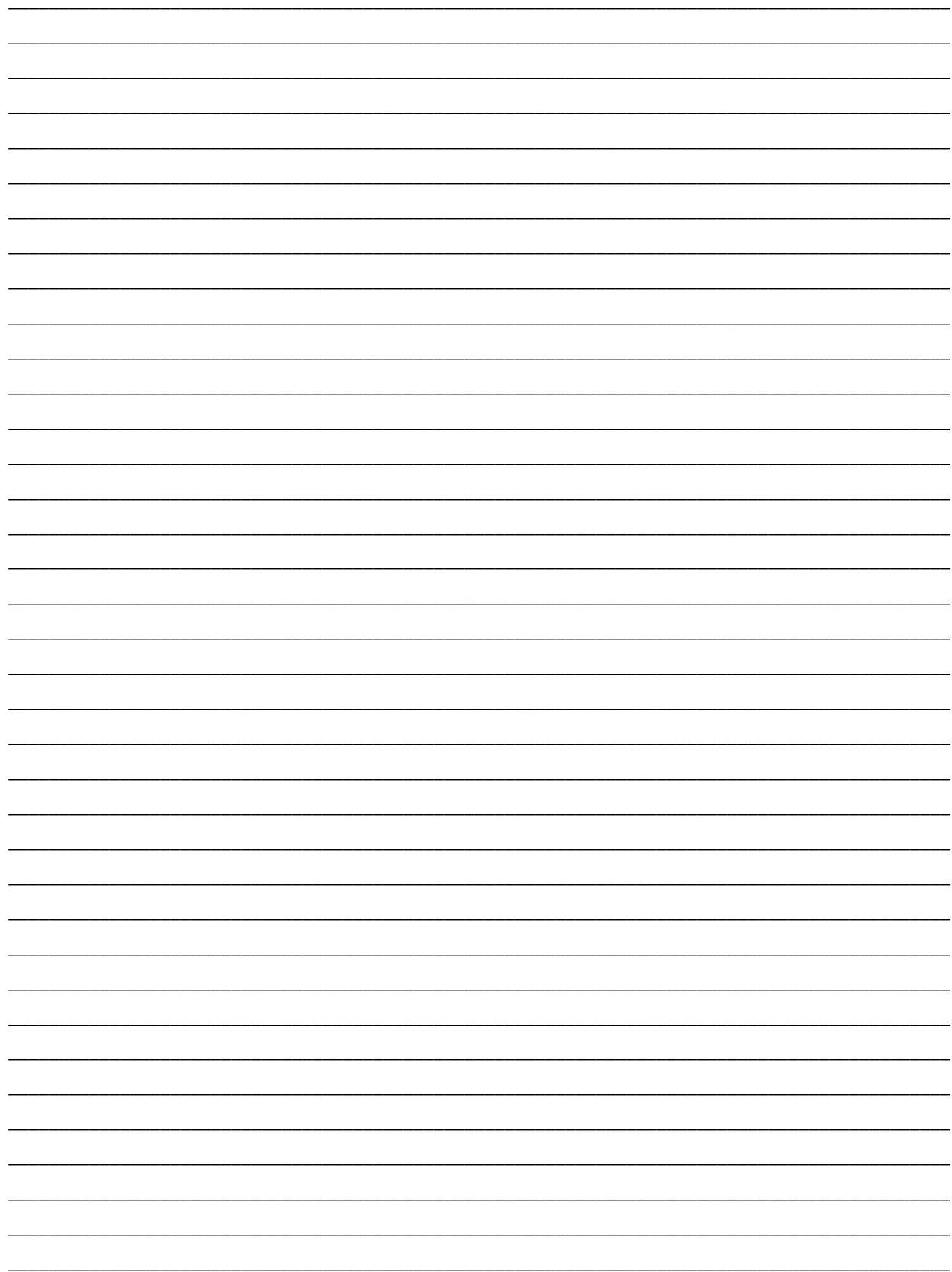
Before you start, here are a few principles to guide your sorting and compression efforts: (1) Find the columns that will give you the biggest gains (usually text columns with the fewest distinct values like Country or Region), (2) find which columns would allow the effective sorting of other columns (e.g., sorting Country first may support an effective sort of Region, which will likely allow an effective sort of State), (3) start by sorting the column that will provide the biggest gain and allow for the effective sorting of the most other columns, (4) proceed to sort the next columns with the previous principle in mind. Note: Power Query will sort as many columns as you tell it to and it will respect the order in which you perform the sorting.

Feel free to go through each table and determine what you think is best for each table before following the steps below. After each instruction, reasoning will be provided.

1. In the Power Query Editor, left-click *Sales* in the Queries pane
2. Left-click the drop-down arrow in the *Order Start of Month* column header
3. In the drop-down menu, left-click *Sort Ascending*

This table has no text columns with few repeating values, which are the obvious first targets. In fact, this table no longer has any text columns. Columns like the value column, which have very few repeating values, are rarely candidates for a good sort. And because the ID columns aren't related to each other in any way, sorting them wouldn't necessarily promote better sorting within each other.





# Other Classes Taught by JourneyTEAM

## Power BI

- Fundamentals of DAX
- Intermediate DAX
- Advanced DAX
- Dashboard in a Day
- Advanced Dashboard in a Day
- Admin in a Day
- Advanced Visualizations

## Power Automate

- Flow in a Day
- Advanced Flow in a Day

## Power Apps

- Power Apps in a Day
- Advanced Power Apps in a Day

## Power Virtual Agent

- Virtual Agent in a Day





JOURNEYTEAM